J. Cryptol. (2013) 26: 513–558
DOI: 10.1007/s00145-012-9136-3

Journal of
CRYPTOLOGY

Fully Leakage-Resilient Signatures*

Elette Boyle†
Department of Mathematics, Massachusetts Institute of Technology, Cambridge, MA 02139, USA
eboyle@mit.edu

Gil Segev‡
Microsoft Research, Mountain View, CA 94043, USA
gil.segev@microsoft.com

Daniel Wichs§
Department of Computer Science, New York University, New York, NY 10012, USA

Communicated by Jonathan Katz

Received 10 April 2012
Online publication 31 October 2012

Abstract. A signature scheme is *fully leakage resilient* (Katz and Vaikuntanathan, ASIACRYPT'09) if it is existentially unforgeable under an adaptive chosen-message attack even in a setting where an adversary may obtain bounded (yet arbitrary) leakage information on *all intermediate values that are used throughout the lifetime of the system*. This is a strong and meaningful notion of security that captures a wide range of side-channel attacks.

One of the main challenges in constructing fully leakage-resilient signature schemes is dealing with leakage that may depend on the random bits used by the signing algorithm, and constructions of such schemes are known only in the random-oracle model. Moreover, even in the random-oracle model, known schemes are only resilient to leakage of less than half the length of their signing key.

In this paper we construct the first *fully* leakage-resilient signature schemes without random oracles. We present a scheme that is resilient to any leakage of length $(1 - o(1))L$ bits, where L is the length of the signing key. Our approach relies on generic cryptographic primitives, and at the same time admits rather efficient instantiations based on specific number-theoretic assumptions. In addition, we show that our approach extends to the continual-leakage model, recently introduced by Dodis, Haralambiev, Lopez-Alt and Wichs (FOCS'10), and by Brakerski, Tauman Kalai, Katz and

* Solicited from Eurocrypt 2011. A preliminary version of this work appeared in *Advances in Cryptology—EUROCRYPT'11*, pp. 89–108, 2011.

† Research of E. Boyle was supported by the US National Defense Science and Engineering Graduate Fellowship. This work was partially completed while visiting the Weizmann Institute of Science.

‡ This work was partially completed while G. Segev was a Ph.D. student at the Weizmann Institute of Science, and supported by the Adams Fellowship Program of the Israel Academy of Sciences and Humanities.

§ This work of D. Wichs was partially completed while visiting the Weizmann Institute of Science.

Vaikuntanathan (FOCS'10). In this model the signing key is allowed to be refreshed, while its corresponding verification key remains fixed, and the amount of leakage is assumed to be bounded only in between any two successive key refreshes.

Key words. Leakage-resilient cryptography, Signature schemes.

1. Introduction

One of the main goals of research in the foundations of cryptography is designing systems that withstand adversarial behavior. Given a cryptographic task, such as public-key encryption, one must formalize an attack model specifying a class of adversaries, and define a notion of security capturing what it means to break the system. Within such a framework, it is then possible to rigorously analyze the security of cryptographic systems.

Starting with the seminal work of Goldwasser and Micali [29], various and increasingly strong attack models and notions of security have been proposed. Over the years, however, theoreticians and practitioners began to notice that a large class of realistic attacks, called *side-channel attacks*, are not captured by the existing models. In such attacks, the adversary may learn some additional information regarding the internal secret state of a system, by measuring various properties resulting from specific *physical* implementations (e.g., timing information, detection of internal faults, electromagnetic radiation, power consumption etc. [8,11,46,47]). As a result, it has become an important research agenda to extend the standard models to capture such side-channel attacks, and to design cryptographic systems whose security guarantees can be rigorously analyzed and clearly stated in these stronger models. Our work focuses on the model of *memory attacks*, and its *bounded-leakage* and *continual-leakage* variants, which we describe next (several other models are described in Sect. 1.2).

Memory Attacks: Bounded-Leakage and Continual-Leakage The model of *memory attacks* was introduced by Akavia, Goldwasser, and Vaikuntanathan [1]. Its main premise is that the adversary can learn *arbitrary* information regarding the secret state of a system, subject only to the constraint that the *amount* of information learned is somehow bounded. More precisely, the adversary can adaptively select *arbitrary poly-time computable* functions $f_i : \{0,1\}^* \to \{0,1\}^{\lambda_i}$ and learn the value of f_i applied to the internal state of the system, subject only to some constraint on the output sizes λ_i.

The work of [1] assumes that there is an a priori determined *leakage bound* λ, which bounds the *overall* amount of information learned by the adversary throughout the entire lifetime of the system to be $\sum_i \lambda_i \leq \lambda$. We call this the *bounded-leakage model*. Usually the leakage bound λ is also related to the secret-key size, so that a *relatively* large fraction $\lambda/|sk|$ of the secret key can be leaked. A great deal of research has gone into devising various cryptographic primitives in this model, such as public-key and identity-based encryption schemes, signature schemes, and more (see, for example, [2,3,13,19,43,51,55]).

A drawback of the bounded-leakage model is that, if a system is being used continually for a sufficiently long time, then the amount of leakage observed by the attacker may exceed any a priori determined leakage bound. Hence, we would like to bound the *rate* of leakage rather than the *overall amount* of leakage. If we do not bound the overall

leakage, then any static piece of information that stays unmodified on the system can eventually be fully recovered by the adversary. Hence the secret keys of such systems must be periodically *refreshed*.

Recently, Dodis et al. [18] and Brakerski et al. [15] suggested the *continual-leakage model*, in which a scheme periodically *self-refreshes* its internal secret key, while the corresponding public key remains fixed. In this model, only the amount of leakage seen by the adversary *in between any two successive refreshes* is assumed to be a priori bounded by some leakage bound λ.[1] However, there is no a priori bound on the overall amount of information seen by the adversary throughout the lifetime of the system.

We note that in both the bounded-leakage model and the continual-leakage model the adversary may be able to learn partial, but yet *arbitrary*, information on the *entire* secret key. This is in contrast with other models, where either the leakage is assumed to be of "low complexity" (such as AC^0 circuits) [25,40], or certain secret values are assumed to be leak-free [23,24,31,42,54,59].

Leakage-Resilient Signature Schemes In this paper we study the security of signature schemes in the bounded-leakage and continual-leakage models. Signature schemes in the bounded-leakage model were proposed by Alwen, Dodis, and Wichs [3] and by Katz and Vaikuntanathan [43], who focused mainly on leakage of (*only*) *the signing key* of the scheme. Specifically, a signature scheme is leakage-resilient in the bounded-leakage model if it is existentially unforgeable against an adaptive chosen-message attack [30] even when adversarially chosen functions of the signing key are leaked in an adaptive fashion. Signature schemes satisfying this notion of security were constructed both based on generic cryptographic primitives in the standard model [43] and based on the Fiat–Shamir transform [26] in the random-oracle model [3,43].

Although this notion of leakage resilience already captures some attacks, it does not fully capture general leakage attacks, which may depend on the *entire internal state* of the system. In particular, the problem is that both of the signature scheme constructions from [3,43] are *randomized* and hence the internal state includes, in addition to the secret key, all of the random coins used by the signing algorithm.[2] The prior schemes may therefore be vulnerable to leakage attacks that (also) depend on this randomness.

This was already noted by Katz and Vaikuntanathan [43], who put forward the stricter notion of a *fully leakage-resilient* signature schemes (in the bounded-leakage model). This notion requires a signature scheme to remain existentially unforgeable under an adaptive chosen-message attack even when the adversary obtains bounded leakage information on *all intermediate values* used by the signer throughout the lifetime of the system, including the secret keys *and* internal random coins (the notion can be naturally extended to the continual-leakage model [15,18]). This stronger notion seems to better capture real attacks, relying on e.g. timing or power consumption patterns, since these likely *do* depend on the internal randomness.

[1] If the time between refreshing is fixed, we can think of this as bounding the *rate* of leakage.

[2] No known deterministic or public-coin constructions of leakage-resilient signatures are known. Without leakage, the signing algorithm of any signature scheme can be made deterministic by using, as its random coins, the output of a pseudorandom function (PRF) applied to the message, where the seed of the PRF is made part of the secret key. However, in the setting of key leakage, this transformation may no longer be secure since the seed to the PRF can also leak.

Currently, however, the known constructions of fully leakage-resilient signature schemes are proven secure only in the random-oracle model [3,15,18,43]. Moreover, even in the random-oracle model, known schemes are either resilient to leakage of at most half the length of the signing key [3,18,43], or require refreshing of the signing key after every few invocation of the signing algorithm, even when no leakage occurs [15] (this is required even in the bounded-leakage model, where refreshing is not part of the typical functionality). In the standard model, only constructions of "one-time" signatures[3] from [43] are known to be fully leakage resilient.

In a concurrent and independent work, Malkin, Teranishi, Vahlis and Yung [52] propose an alternative signature scheme in the continual-leakage model. Although the two schemes appear very different at first, they can be seen as separate instantiations of a common strategy, which we will explain shortly.

1.1. *Our Contributions*

We construct the first fully leakage-resilient signature schemes without random oracles. We first present a scheme in the bounded-leakage model that is resilient to any leakage of $(1 - o(1))L$ bits, where L is the bit-length of the signing key. Our scheme is based on generic cryptographic primitives, and is inspired by the approach of Katz and Vaikuntanathan [43] (although their scheme is resilient to leakage from the signing key only). Moreover, we show that our construction can be instantiated based on specific number-theoretic assumptions to yield a rather *efficient* scheme.

We then extend our approach to the continual-leakage model by relying on any *continual leakage-resilient one-way relation*, a primitive recently introduced by Dodis, Haralambiev, Lopez-Alt and Wichs [18]. Our resulting signature scheme construction inherits the leakage-resilience properties of the underlying one-way relation with respect to leakage allowed between successive key updates and during the refreshing algorithm. Instantiating our scheme with the construction of the one-way relations from [63] we can get signature schemes that are resilient to any leakage of length $(1 - o(1))L$ bits between any refreshes and a logarithmic number of bits during the refreshing, under the decisional linear assumption. Alternatively, using a construction of one-way relations from [20], we get a signature scheme that is secure for up to some $O(L)$ bits of leakage in-between and during each refreshing, under the same assumption.

Finally, we note that our approach yields the first separation between the bounded-leakage model and the noisy leakage model, which was formalized by Naor and Segev [55] and later refined by Dodis et al. [18, Definition 7.2]. Noisy leakage (also known as entropy-bounded leakage) is a realistic generalization of length-bounded leakage, in which the leakage is not necessarily of bounded length, and it is only guaranteed that the secret key still has some min-entropy even given the leakage. This settles an open problem posed by Naor and Segev (see Sect. 8 for a more elaborated discussion).

1.2. *Related Work*

Various constructions of leakage-resilient signature schemes have been proposed so far. In this section we describe the different leakage models and discuss the security guar-

[3] Such schemes can only be used to sign a single message (or, more generally, some a priori bound t on the number of messages).

antees that are satisfied by these constructions. In what follows we denote by L the bit-length of the signing key.

The Limited-Complexity Leakage Model Some of the initial works in leakage-resilient cryptography concentrated on specific and limited forms of leakage. Ishai et al. [40] showed how to securely implement any efficiently computable function in the presence of an adversary who can probe the values of a constant number of wires. Faust et al. [25] demonstrated a related construction allowing leakage functions with bounded output, belonging to a low complexity class, such as AC^0. Unfortunately, to accommodate such generality in the underlying function, constructions of this type so far handled only weak forms of leakage, which seem rather far from capturing many known side-channel attacks.

The "Only Computation Leaks" Model Micali and Reyzin [54] introduced a model in which the complexity of leakage functions is unrestricted, the overall amount of leakage is unbounded, but leakage is assumed to only occur on values currently accessed during a computation. Explicitly, in each step of computation, the adversary can adaptively select a polynomial-time leakage function f with bounded output to be applied to current active values. Values are active in a computation step if they are accessed by the cryptographic system; all other values stored in memory but not accessed during the time-step are assumed to be leak-free for the step. This model lends itself to techniques that split a computation into two halves, maintaining only one half as active in any given time-step, as proposed by Dziembowski and Pietrzak [23,59] within the construction of leakage-resilient stream ciphers.

Faust et al. [24] constructed a stateful, tree-based signature scheme in the "only computation leaks" model based on any 3-time secure signature scheme. Their construction can handle $\ell/3$ bits of leakage per signature, where the underlying scheme is resilient to ℓ bits of leakage overall. Implementing this with currently known constructions for the underlying scheme (see [3,43]) yields leakage resilience approaching $L/36$ bits, based on one-way functions.

Goldwasser and Rothblum [31] recently presented a way to compile any cryptographic algorithm into one that resists leakage bounded by a constant fraction of the secret key size in the "only computation leaks" model, based on the DDH assumption and the existence of a simple secure hardware component. Using different techniques, Juma and Vahlis [42] showed how to encapsulate a secret key and compute on it in a leakage-resilient fashion within the "only computation leaks" model using fully homomorphic encryption and leak-free hardware tokens.

The "only computation leaks" paradigm is powerful in the sense that it allows any efficiently computable leakage functions, and a large amount of overall leakage. However, it rests on the assumption that values not immediately being used in computation are completely leak-free, which is not always valid. Specifically, this model does not capture known attacks in which inactive values in memory are still vulnerable, such as the cold-boot attacks of Halderman et al. [35] (see also the improvements of Heninger and Shacham [37]), or measurements which can detect the physical processes used to maintain values in memory.

The Bounded-Leakage Model In the bounded-leakage model, as discussed above, signature schemes resilient to leakage depending only on the signing key have been constructed in the random-oracle model [3] and in the standard model (i.e., without random oracles) [43], handling leakage of up to $(1 - o(1))L$ bits (see also the work of Dodis et al. [19] that focuses on efficient instantiations of such schemes).

Many-time signature schemes resistant to leakage from the entire secret state have been constructed only in the random-oracle model. Alwen et al. [3] and Katz and Vaikuntanathan [43] presented a scheme using the Fiat–Shamir transform [26] with any second-preimage resistant function. Both works concurrently proposed the same scheme, but leakage of randomness was only analyzed in the latter [43]. Due to the use of the Fiat–Shamir transform, the construction is only resilient to leakage approaching length $L/2$. A more efficient version, allowing the same amount of leakage, was given by Dodis et al. [19].

The only existing constructions of fully leakage-resistant signatures in the standard model are a pair of one-time signature schemes due to Katz and Vaikuntanathan [43]. Their first scheme is based on the existence of any one-way function, and is resilient to any leakage of length less than $L/4$ bits. The second is based on specific number-theoretic assumptions, and is resilient to any leakage of length less than $L/2$ bits. Both schemes extend to t-time schemes with leakage less than $\Theta(L/t)$. Since these schemes have a deterministic signing algorithm, the secret state of the signer consists only of the secret key, and thus full leakage resilience simply reduces to the weaker notion of resilience against key leakage.

The Continual-Leakage Model Notions of leakage-resilient signatures were naturally extended to the continual-leakage setting by Dodis et al. [18] and Brakerski et al. [15], as described above. Dodis et al. [18] presented two signature schemes in the continual-leakage model by using their abstracted concept of a continual leakage-resilient one-way relation together with known schemes from the bounded-leakage model. The first scheme is a modified version of the scheme of Katz and Vaikuntanathan [43], and is resilient to leakage only from the signing key. Their second scheme uses the Fiat–Shamir transform in the random-oracle model, and allows up to $L/2$ bits of leakage in each round from the entire current secret state of the signer.

Brakerski et al. [15] also presented two signature schemes in this model. The first scheme is based on the one of Katz and Vaikuntanathan [43] as in [18], and is resilient to leakage only from the signing key. The second scheme is in the random-oracle model, and is secure against leakage from the entire secret state of the signer. The amount of leakage that can be tolerated is $(1/2 - o(1))L$ bits per key update based on the decisional linear assumption (this was later improved by Tauman Kalai et al. [63] to $(1 - o(1))L$ bits), and $(1 - o(1))L$ bits based on the SXDH assumption. In their scheme, each signature leaks information regarding the secret key, regardless of whether the adversary makes leakage queries. This continual leakage of information through signature queries means the signing key must be refreshed after every few invocation of the signing algorithm. Thus, the construction does not yield a scheme in the bounded-leakage model, where refreshing is not part of the typical functionality of a signature scheme, and is less efficient in the continual-leakage model.

Subsequent to the present work, Garg et al. [28] showed that the adaptively secure UC NIZKs of Groth et al. [33] possess leakage-resilient properties, which in turn yield

leakage-resilient signature schemes in both the bounded- and continual-leakage models. Their construction additionally relies on (and inherits the leakage-resilient properties of) any underlying leakage-resilient one-way relation. The scheme also remains secure in the noisy leakage model of Naor and Segev [55].

1.3. *Overview of Our Approach*

In this section we present an overview of our approach for constructing fully leakage-resilient signature schemes. We focus here on our construction in the bounded-leakage model, as it already emphasizes the main ideas underlying our approach, and we refer the reader to Sect. 7 for an overview of our construction in the continual-leakage model. We begin by describing more clearly the notion of a fully leakage-resilient signature scheme in the bounded-leakage model. Then, we briefly describe the leakage-resilient signature scheme of Katz and Vaikuntanathan [43], which serves as our starting point, and explain the main challenges in constructing *fully* leakage-resilient signature schemes. The main part of this overview then focuses on our construction.

Modeling Fully Leakage-Resilient Signature Schemes A signature scheme is fully leakage-resilient in the bounded-leakage model if it is existentially unforgeable against an adversary that can obtain both signatures on any message of her choice, and bounded leakage information on all intermediate values used by the signer throughout the lifetime of the system.

This is formalized by considering an experiment that involves a signer and an adversary. First, the signer invokes the key-generation algorithm and obtains a verification key vk and a signing key sk. At this point, a value state is initialized to contain the random coins that were used by the key-generation algorithm. The adversary is given the verification key vk and can adaptively submit two types of query: *signing queries*, and *leakage queries*. A signing query consists of a message m, and is answered by invoking the signing algorithm with the signing key and the message. Following each such query, the random coins that were used by the signing algorithm are added to the state. A leakage query consists of a leakage function f, and is answered by applying f to the value state. The leakage functions have to be efficiently computable, and the sum of their output lengths has to be upper bounded by a predetermined parameter λ. The adversary is successful if she outputs a pair (m^*, σ^*), where m^* is a message with which she did not issue a signing query, and σ^* is a valid signature on m^* with respect to vk. We refer the reader to Sect. 3 for a formal definition.

The Katz–Vaikuntanathan Scheme The Katz–Vaikuntanathan signature scheme [43] relies on a second-preimage resistant (SPR) function $\mathsf{F} : \{0,1\}^{\mu(n)} \to \{0,1\}^{\kappa(n)}$ (for some $\kappa(n) < \mu(n)$), a CPA-secure public-key encryption scheme, and a (unbounded simulation-sound) NIZK proof system.[4] The signing key is a random $x \in \{0,1\}^{\mu(n)}$,

[4] A function F is second-preimage resistant if, given a random input x it is hard to find $x' \neq x$ such that $\mathsf{F}(x') = \mathsf{F}(x)$. See Definition 2.2 in Sect. 2. We note that when F is only assumed to be a one-way function, the scheme may not always be resilient to leakage, but it is nevertheless existentially unforgeable under an adaptive chosen-message attack. In this case the scheme can be viewed as a variant of the Bellare–Goldwasser signature scheme [6].

and the verification key is a triplet $(y = \mathsf{F}(x), pk, \mathsf{crs})$, where pk is a public key for the encryption scheme, and crs is a common-reference string for the proof system. A signature on a message m consists of a ciphertext c which is an encryption of $m||x$ using pk, and a proof that the ciphertext c is indeed an encryption of $m||x'$, for some $x' \in \mathsf{F}^{-1}(y)$.[5]

This scheme is leakage resilient in the bounded-leakage model. That is, it satisfies the weaker variant of the above notion of security, where the leakage is allowed to depend on the signing key only. The security of the scheme is based on three main properties:

1. A typical verification key has many possible secret keys. Specifically, the set $\mathsf{F}^{-1}(y)$ is of size roughly $2^{\mu(n)-\kappa(n)}$.
2. The "real" signatures of the scheme are *computationally indistinguishable* from "fake" signatures, which are *statistically independent* of the signing key. This follows from the semantic security of the encryption scheme and from the zero knowledge of the proof system. Specifically, a "fake" signature on a message m can be produced by encrypting $m||0^n$, and then using the NIZK simulator to generate the proof.
3. Given the decryption key corresponding to pk, any valid forgery produced by the adversary can be used to extract a preimage x' of y. This follows from the soundness of the proof system, which guarantees that the adversary's forgery is a "real" signature[6] and therefore the corresponding ciphertext can be decrypted to a valid preimage x'.

These three properties are used to prove the security of the scheme as follows. Assume there is an adversary that breaks the scheme. Then, given a random pre-image x of y, we can run this adversary and (by the third property) extract some valid preimage x' from the adversary's signing forgery with a reasonable probability. This would break second-preimage resistance of F as long as we can argue that $x' \neq x$. To do so, we use the second property to replace "real signatures" with "fake signatures" without affecting the probability of recovering some valid preimage x'. But now, the signing queries do not reveal any additional information regarding x, given y. So the only correlated information on x that the adversary sees is the value $y = F(x)$ of size $\kappa(n)$ and the leakage of size λ. Therefore, if $\lambda \leq \mu(n) - \kappa(n) - \omega(\log(n))$, then the adversary has (information-theoretically) super-logarithmic uncertainty about the value of x and hence the probability of extracting $x' = x$ from her forgery is negligible.

The Main Challenges The security proof of the Katz–Vaikuntanathan scheme relies on the argument that, given many signatures of chosen messages and λ bits of leakage from the signing key x, the value x is still hard to guess by the adversary. However, when the leakage may depend also on the randomness used by the signing algorithm, this is no longer true, and in fact the scheme is insecure in general. The main problem is that, in the above argument, we crucially used the ability to switch "real" signatures for "fake"

[5] Katz and Vaikuntanathan show that it is actually possible to encrypt only x (instead of $m||x$), and include m as a label in the statement that is proved using the NIZK proof system. However, for making this informal description more intuitive, we consider here an encryption of both m and x.

[6] In fact, a stronger notion called simulation-soundness is required, because the adversary gets to see several fake proofs before generating her signature.

signatures. This step, in turn, relied on the security of the encryption scheme and the zero-knowledge property of the proofs. However, we cannot rely on these properties if the adversary can also leak on the random coins of the encryption scheme and the proof system! Consider, for example, an instantiation of the scheme with a CPA-secure encryption scheme defined as $\mathsf{Enc}_{pk}(m||x) = (\mathsf{Enc}'_{pk}(s), \mathsf{PRG}(s) \oplus (m||x))$, where Enc' is a CPA-secure encryption scheme, and PRG is a pseudorandom generator that is applied on a random seed s. Leaking the seed s, whose length may be arbitrarily shorter than λ, completely reveals the signing key x. A similar instantiation for the proof system can be shown to have a similar effect when the leakage may depend on the randomness used by the prover.[7]

Our Approach A natural observation is that the above problems can be avoided if the "real" and "fake" signatures cannot be distinguished *even* given the random coins used to generate them. Remember that fake signatures are statistically independent of the secret key x, while real signatures allow us to extract some preimage using an appropriate trapdoor (decryption key).

The first idea toward achieving the above is to replace the (unbounded simulation-sound) NIZK proof system with a *statistical non-interactive witness-indistinguishable (SNIWI) argument system*. On one hand we relax the (unbounded simulation-sound) zero-knowledge property to *witness indistinguishability*, and on the other hand we require that proofs generated using different witnesses are *statistically* indistinguishable from each other. In particular, this guarantees that *even* a correctly generated proof is statistically independent of the witness (in our case the signing key x) used to generate it.

The harder part lies in getting an encryption scheme where the ciphertexts are independent of the message (in our case, the signing key x) that they encrypt. In particular, this clearly contradicts the decryptability of a ciphertext. We could imagine using known lossy encryption schemes, where the encryption key pk can be generated in one of two indistinguishable modes: "*injective*" mode which allows for decryptability, and "*lossy*" mode where ciphertexts statistically hide the message. But remember that we need to satisfy the following two properties simultaneously: (1) the ability to answer the adversary's signing queries with fake signatures that reveal no information regarding x, (2) the ability to extract a witness x' from the adversary's forgery. By setting the pk to be in either injective or lossy mode, we can achieve either property, but not at the same time! The main tool used in resolving this conflict is to design a *partitioned-lossy encryption scheme*, where the encryption of some messages is lossy while that of others is injective.

A Selectively Unforgeable Signature Scheme For the reader's intuition, we first show how to achieve a weaker notion of signature security that we refer to as *selective unforgeability under a chosen-message attack*. For this notion, we assume the adversary specifies the message m^* on which she plans to forge a signature in advance, before receiving the verification key. The signing queries and leakage are still adaptive.

[7] Note that even a leakage function with only one output bit can be easily used to distinguish an encryption of $m||x$ from an encryption of $m||0^n$, or to distinguish the prover of the proof system from the simulator of the proof system. Thus, technically speaking, it seems that at no point in time during the various experiments of the security proof it is possible to change the way signing queries are answered.

To achieve this notion of security, we introduce the concept of an *all-lossy-but-one* (*ALBO*) *public-key encryption scheme*. This is a tag-based public-key encryption scheme, where the encryption procedure takes as input a tag t in addition to the message. The key-generation procedure takes as input a special tag t^* and produces a key pair (pk, sk) such that encrypting under the tag t^* allows for efficient decryption with sk, but encryption under any other tag $t \neq t^*$ statistically hides the encrypted message. We call t^* the *injective* tag, and any other tag a *lossy* tag.[8] The only computational requirement is that the public key hides the injective tag t^* that was used for its generation.

We now modify the Katz–Vaikuntanathan signature scheme by using an ALBO encryption scheme instead of a standard CPA-secure scheme. To sign m, we encrypt (only) the signing key x under the tag $t = m$. We use a SNIWI argument system instead of a simulation-sound NIZK to generate the proof. To argue security, we note that since the adversary's forgery message m^* is chosen ahead of time, we can generate the encryption key pk such that $t^* = m^*$ is the only injective tag, without affecting the adversary's ability to forge—this change is indistinguishable even given full view of the signing key x and randomness of signing. Now we are in a situation where all the signing queries for $m \neq m^*$ yield signatures which are statistically independent of the signing key x, while the forgery can be used to extract some preimage x'. Therefore, we can argue as before: the bounded leakage on the secret key x and randomness of signing is short enough that x must have entropy left given this leakage, and therefore the outcome $x' = x$ is unlikely.

The Full Scheme So far we described our approach as leading to the rather weak notion of selective unforgeability under a chosen-message attack. Our actual scheme is fully leakage-resilient according to the stronger notion that was discussed in the beginning of this section (i.e., where the adversary is allowed to adaptively choose m^* after seing vk and responses to all signing and leakage queries).

We note that, in the random-oracle model, there is a simple generic transformation from selective security to full security by signing the output of the random oracle applied to the message. Alternatively, in the standard model, there is a simple transformation with exponential security loss by simply "guessing" the forgery: this can yield fully secure schemes under some exponential hardness assumptions by using complexity-leveraging. Lastly, there is a completely generic transformation due to [14] (abstracting a non-generic approach of [38]) by hashing the message with a chameleon hash function [49] and signing each prefix of the hash separately. Unfortunately, this results in long signatures. All of these generic techniques also work in the setting of full-leakage resilience. We present an alternative that does not suffer from the above disadvantages.

For our actual scheme, we follow the approach of Boneh and Boyen [9] for transforming selectively secure identity-based encryption schemes into fully secure ones using an admissible hash function (see Sect. 2.6). This relies on a slightly more refined "partitioning strategy" than the "all-but-one" strategy used for the selectively secure scheme. In particular, we introduce the notion of a $\mathcal{R}$*-lossy public-key encryption scheme*. This

[8] We note that our notion is the opposite of the notion of an all-but-one lossy trapdoor function [58], where there is one lossy tag and all the other tags are injective.

is a generalization of an ALBO encryption scheme where the set of possible tags is partitioned into injective tags and lossy tags according to a relation $\mathcal{R}$ (in particular, there may be more than one injective tag). The main idea of this approach is to ensure that, with a non-negligible probability, all of the adversary's signing queries will fall into the "lossy" partition, while the forgery falls into the "injective" partition.

Comparison to [52] An alternate way to view our combination of a SNIWI paired with a partitioned lossy encryption is as a tag-based proof system that is partitioned to be extractable for some tags and statistically witness indistinguishable for others. Our main result shows how to build fully leakage-resilient signatures from such a proof system. The work of [52] can be seen as an alternate instantiation of this strategy which relies on Groth–Sahai NIZKs [34]. These NIZKs are either statistically witness indistinguishable or extractable depending on the choice of the CRS. In the construction of [52], the actual CRS for each use of the Groth–Sahai NIZK is derived from the tag in a clever way (using the Waters hash [64]) so as to give an alternate method for partitioning lossy/extractable tags.

1.4. *Paper Organization*

In Sect. 2 we introduce some preliminaries and notation. Section 3 contains a discussion of the models of leakage resilience considered in this paper. In Sect. 4 we define and construct $\mathcal{R}$-lossy public-key encryption schemes, a tool used in our constructions. Section 5 contains the construction and security proof of our signature scheme in the bounded-leakage model. In Sect. 6 we present a specific instantiation of our scheme based on the decisional linear assumption. In Sect. 7 we extend our scheme to the continual-leakage model. Finally, in Sect. 8 we discuss several concluding remarks and open problems.

2. Preliminaries

In this section we present some basic notions, definitions, and tools that are used in our constructions.

2.1. *Statistical Distance, Min-Entropy, and Average Min-Entropy*

The *statistical distance* between two random variables X and Y over a finite domain Ω is defined as $\mathrm{SD}(X,Y) = \frac{1}{2}\sum_{\omega\in\Omega} |\Pr[X=\omega] - \Pr[Y=\omega]|$. We say that two variables are ϵ-*close*, and write $X \approx_\epsilon Y$, if their statistical distance is at most ϵ. The *min-entropy* of a random variable X is $\mathrm{H}_\infty(X) = -\log(\max_x \Pr[X=x])$. Dodis et al. [21] formalized the notion of *average min-entropy* that captures the remaining unpredictability of a random variable X conditioned on the value of a random variable Y, which is defined as follows:

$$\widetilde{\mathrm{H}}_\infty(X|Y) = -\log\bigl(E_{y\leftarrow Y}\bigl[2^{-\mathrm{H}_\infty(X|Y=y)}\bigr]\bigr).$$

The following bound on average min-entropy was proved in [21]:

Lemma 2.1 ([21]). *For any random variables X, Y and Z, if Y has at most 2^λ possible values then*

$$\widetilde{\mathrm{H}}_\infty(X|Y,Z) \geq \widetilde{\mathrm{H}}_\infty(X|Z) - \lambda.$$

2.2. *The DDH, SXDH, and d-Linear Assumptions*

Let $\mathsf{GroupGen}$ be a probabilistic polynomial-time algorithm that takes as input a security parameter 1^n, and outputs a triplet $(\mathbb{G}, p, g)$ where $\mathbb{G}$ is a group of order p that is generated by $g \in \mathbb{G}$, and p is an n-bit prime number. In addition, let $\mathsf{BilinearGroupGen}$ be a probabilistic polynomial-time algorithm that takes a security parameter 1^n as input and outputs a 5-tuple $(\mathbb{G}, \mathbb{G}_T, p, g, e)$, where $\mathbb{G}$ and $\mathbb{G}_T$ are groups of order p, the group $\mathbb{G}$ is generated by g, and $e : \mathbb{G} \times \mathbb{G} \to \mathbb{G}_T$ is a bilinear map.

The Decisional Diffie–Hellman Assumption The decisional Diffie–Hellman (DDH) assumption is that the ensembles $\{(\mathbb{G}, g_1, g_2, g_1^r, g_2^r)\}_{n\in\mathbb{N}}$ and $\{(\mathbb{G}, g_1, g_2, g_1^{r_1}, g_2^{r_2})\}_{n\in\mathbb{N}}$ are computationally indistinguishable, where $(\mathbb{G}, p, g) \leftarrow \mathsf{GroupGen}(1^n)$, and the elements $g_1, g_2 \in \mathbb{G}$ and $r, r_1, r_2 \in \mathbb{Z}_p$ are chosen independently and uniformly at random.

The d-Linear Assumption Boneh, Boyen, and Shacham [10] introduced the decisional linear assumption, intended to take the place of DDH in groups where DDH is easy (specifically, in bilinear groups). They showed that the hardness of DDH implies the hardness of the decisional linear assumption, but at least in generic groups (see, for example, [41,62]), the decisional linear assumption remains hard even if DDH is easy. The DDH and the decisional linear assumptions naturally generalize to the family of (decisional) d-linear assumptions [45,61], where for every $d \geq 1$ the d-linear assumption is that the ensembles

$$\left\{\left(\mathbb{G}, g_1, \ldots, g_d, g_{d+1}, g_1^{r_1}, \ldots, g_d^{r_d}, g_{d+1}^{\sum_{i=1}^{d} r_i}\right)\right\}_{n\in\mathbb{N}}$$
$$\left\{\left(\mathbb{G}, g_1, \ldots, g_d, g_{d+1}, g_1^{r_1}, \ldots, g_d^{r_d}, g_{d+1}^{r_{d+1}}\right)\right\}_{n\in\mathbb{N}},$$

are computationally indistinguishable, where $(\mathbb{G}, p, g) \leftarrow \mathsf{GroupGen}(1^n)$, and the elements $g_1, \ldots, g_{d+1} \in \mathbb{G}$ and $r_1, \ldots, r_{d+1} \in \mathbb{Z}_p$ are chosen independently and uniformly at random.

Note that DDH is the 1-linear assumption, and that decisional linear assumption is the 2-linear assumption. These assumptions are progressively weaker: the hardness of d-linear implies the hardness of $(d+1)$-linear, but in generic groups $(d+1)$-linear remains hard even if d-linear is easy.

The SXDH Assumption Some parts of our construction will rely on bilinear groups containing a pairing $e : \mathbb{G}_1 \times \mathbb{G}_2 \to \mathbb{G}_3$. In this case, the DDH (1-linear) assumption can only hold for some *asymmetric* pairings where the groups $\mathbb{G}_1, \mathbb{G}_2$ are different from each other and there is no efficient homomorphism between them. In other words, when dealing with bilinear group, the DDH assumption is a strong assumption which may only hold in restricted settings. We therefore use the pairing-terminology, calling it SXDH (symmetric external DDH) to distinguish from the non-pairing setting (we refer

the reader to [4], for example, for more details). On the other hand, the weaker d-linear assumptions for $d \geq 2$ are believed to hold for most cryptographically used bilinear groups. Therefore, when using bilinear groups, it is preferable to build schemes under the decisional linear assumption, or higher values of $d > 2$.

2.3. *Second-Preimage Resistance*

A family of efficiently computable functions is a pair of polynomial-time algorithms $(\mathsf{KeyGen}, \mathsf{F})$, where KeyGen is a probabilistic algorithm that on input 1^n outputs a description $s \in \{0,1\}^*$ of a function $\mathsf{F}(s,\cdot) : \{0,1\}^{\mu(n)} \to \{0,1\}^{\kappa(n)}$. Such a family is *second-preimage resistant* (SPR) if given a randomly chosen input $x \in \{0,1\}^{\mu(n)}$ and a description of a randomly chosen function $s \leftarrow \mathsf{KeyGen}(1^n)$, it is computationally infeasible to find an input $x' \in \{0,1\}^{\mu(n)}$ such that $x' \neq x$ and $\mathsf{F}(s,x) = \mathsf{F}(s,x')$. This is a weakening of the notion of a family of universal one-way hash functions introduced by Naor and Yung [56], in which the input x is allowed to be chosen in an adversarial manner (but still independent of the function description s).

Definition 2.2 (Second-Preimage Resistance). A family $\mathcal{F} = (\mathsf{KeyGen}, \mathsf{F})$ of efficiently computable functions is *second-preimage resistant* if for any probabilistic polynomial-time algorithm $\mathcal{A}$ is holds that

$$\Pr\left[\mathsf{F}_s(x') = \mathsf{F}_s(x) \wedge x' \neq x \;\middle|\; \begin{array}{c} s \leftarrow \mathsf{KeyGen}(1^n), x \leftarrow \{0,1\}^{\mu(n)} \\ x' \leftarrow \mathcal{A}(s,x) \end{array}\right] < \nu(n),$$

for some negligible function $\nu(n)$, where the probability is taken over the choice of $x \leftarrow \{0,1\}^{\mu(n)}$ and over the internal randomness of KeyGen and $\mathcal{A}$.

In addition, we say that $\mathcal{F} = (\mathsf{KeyGen}, \mathsf{F})$ is a family of *public-coin* second-preimage resistant functions, if it satisfies Definition 2.2 even when the algorithm $\mathcal{A}$ takes as input also the internal randomness that was used by $\mathsf{KeyGen}(1^n)$ for sampling the function. We refer the reader to [39] for more details of public-coin hash functions.

For any integer functions $\mu(n)$ and $\kappa(n)$ that are polynomially related, the existence of universal one-way hash functions (and therefore also of second-preimage resistant functions) with domain $\{0,1\}^{\mu(n)}$ and range $\{0,1\}^{\kappa(n)}$ is known to be equivalent to that of one-way functions [60]. As noted by Katz and Vaikuntanathan [43], standard constructions of universal one-way hash functions are public coin. In practice, such public-coin functions can be constructed rather easily from various number-theoretic assumptions. For example, if the discrete log problem is hard in some group $\mathbb{G}$ of prime order p, the family of functions $f_{g_1,\ldots,g_k} : \mathbb{Z}_p^k \to \mathbb{G}$ defined as $f_{g_1,\ldots,g_k}(x_1,\ldots,x_k) = \prod_{i=1}^k g_i^{x_i}$ is second-preimage resistant (and even collision resistant), where $g_1,\ldots,g_k \in \mathbb{G}$ are chosen uniformly and independently at random by the key-generation algorithm.

We note that for public-coin SPR functions, there is actually no need for an explicit key-generation algorithm. Without loss of generality one can define a single function $\mathsf{F}'_r(x) = (r, \mathsf{F}_s(x))$, where $s = \mathsf{KeyGen}(1^n; r)$, and this is also SPR with the same amount of "lossiness" as the family $\mathcal{F}$.

2.4. *Statistical Non-Interactive Witness-Indistinguishable Argument Systems*

A non-interactive argument system for a language L with witness relation R_L is a triplet of algorithms $(\mathsf{CRSGen}, \mathsf{P}, \mathsf{V})$, where CRSGen is an algorithm generating a common reference string crs, and P and V are the prover and verifier algorithms, respectively. The prover takes as input a triplet (crs, x, w), where $(x, w) \in R_L$, and outputs a proof π. The verifier takes as input a triplet (crs, x, π) and either accepts or rejects. In this paper we consider a setting where all three algorithms run in probabilistic polynomial time. The two requirements of an argument system are completeness and soundness with respect to efficient cheating provers. Informally, for every $(x, w) \in R_L$ the prover generates proofs that are always accepted by the verifier, and for every $x \notin L$ any efficient cheating prover has only a negligible probability of convincing the verifier to accept. An argument system is called statistical witness indistinguishable if for any $x \in L$ and any two witnesses $w_0 \neq w_1$ such that $(x, w_0), (x, w_1) \in R_L$, the proofs generated by $\mathsf{P}(\mathsf{crs}, x, w_0)$ and $\mathsf{P}(\mathsf{crs}, x, w_1)$ are statistically indistinguishable given the common reference string.

Definition 2.3 (SNIWI Argument System). A *statistical non-interactive witness-indistinguishable argument system* for a language L with witness relation R_L is a triplet of probabilistic polynomial-time algorithms $(\mathsf{CRSGen}, \mathsf{P}, \mathsf{V})$ such that the following properties hold:

1. *Perfect completeness:* For every $(x, w) \in R_L$ we have

$$\Pr\left[\mathsf{V}\left(\mathsf{crs}, x, \mathsf{P}(\mathsf{crs}, x, w)\right) = 1\right] = 1,$$

   where $\mathsf{crs} \leftarrow \mathsf{CRSGen}(1^n)$, and the probability is taken over the internal randomness of CRSGen, P, and V.
2. *Adaptive soundness:* For every probabilistic polynomial-time prover P^* we have

$$\Pr\left[\mathsf{V}(\mathsf{crs}, x, \pi) = 1 \wedge x \notin L \mid \mathsf{crs} \leftarrow \mathsf{CRSGen}\left(1^n\right), (x, \pi) \leftarrow \mathsf{P}^*\left(1^n, \mathsf{crs}\right)\right] < \nu(n),$$

   for some negligible function $\nu(n)$.
3. *Statistical witness indistinguishability:* There exists a probabilistic polynomial-time algorithm CRSGen_{WI} such that:
   - The distributions $\{\mathsf{CRSGen}(1^n)\}$ and $\{\mathsf{CRSGen}_{WI}(1^n)\}$ are *computationally indistinguishable*.
   - For any triplet (x, w_0, w_1) such that $(x, w_0) \in R_L$ and $(x, w_1) \in R_L$, the distributions $\{\mathsf{crs}, \mathsf{P}(\mathsf{crs}, x, w_0)\}$ and $\{\mathsf{crs}, \mathsf{P}(\mathsf{crs}, x, w_1)\}$ are *statistically indistinguishable*, when $\mathsf{crs} \leftarrow \mathsf{CRSGen}_{WI}(1^n)$.

For our construction we are interested in SNIWI argument systems for NP. Such an argument system is implied by the construction of Groth, Ostrovsky and Sahai [33] that satisfies the stronger notion of a perfect non-interactive zero-knowledge argument system. Their construction can be based on the hardness of either the decisional subgroup problem [12] or the decisional linear problem [10]. As pointed out by Groth et al. we note that in their linear-based construction the algorithm CRSGen admits oblivious

sampling (specifically, the distribution of the common reference string is statistically close to the uniform distribution), which is a technical property that is required for our construction in the bounded-leakage model.

2.5. *Lossy Public-Key Encryption*

A lossy public-key encryption scheme is a public-key encryption scheme in which public keys can be generated in two modes that are computationally indistinguishable: an *injective* mode in which ciphertexts can be decrypted using a corresponding secret key, and a *lossy* mode in which ciphertexts statistically hide the encrypted messages. Lossy public-key encryption was recently found useful for various applications (see, for example, [7,48,57]), and its existence was shown to be equivalent to 2-move statistical semi-honest oblivious transfer via rather simple and efficient black-box reductions [36]. In particular, this implies that it can be realized based also on any other primitive that is known to imply such oblivious transfer protocols, including in particular homomorphic encryption, 2-move private information retrieval, and lossy trapdoor functions. Thus, lossy public-key encryption schemes can be constructed based on the hardness of various number-theoretic problems, such as Decisional Diffie–Hellman (and, more generally, decisional d-linear), quadratic residuosity, composite residuosity, learning with errors, and more.

Definition 2.4 (Lossy PKE). A *lossy public-key encryption scheme* is a 4-tuple of probabilistic polynomial-time algorithms $(\mathsf{KeyGen}_0, \mathsf{KeyGen}_1, \mathsf{Enc}, \mathsf{Dec})$ such that:

1. *Lossy key generation:* $\mathsf{KeyGen}_0(1^n)$ outputs a public key pk.
2. *Injective key generation:* $\mathsf{KeyGen}_1(1^n)$ outputs a secret key sk and a public key pk.
3. *Lossiness under lossy keys:* For every public key pk produced by $\mathsf{KeyGen}_0(1^n)$, and for every two messages $m_0, m_1 \in \{0,1\}^{\ell(n)}$, the statistical distance between the distributions $\mathsf{Enc}_{pk}(m_0)$ and $\mathsf{Enc}_{pk}(m_1)$ is negligible in n.
4. *Decryption under injective keys:* For every message $m \in \{0,1\}^{\ell(n)}$ we have

$$\Pr\left[\mathsf{Dec}_{sk}\left(\mathsf{Enc}_{pk}(m)\right) = m\right] > 1 - \nu(n),$$

   for some negligible function $\nu(n)$, where $(sk, pk) \leftarrow \mathsf{KeyGen}_1(1^n)$, and the probability is taken over the internal randomness of KeyGen_1, Enc and Dec.
5. *Indistinguishability of injective and lossy public keys:* The two ensembles $\{pk : pk \leftarrow \mathsf{KeyGen}_0(1^n)\}_{n\in\mathbb{N}}$ and $\{pk : (sk, pk) \leftarrow \mathsf{KeyGen}_1(1^n)\}_{n\in\mathbb{N}}$ are computationally indistinguishable.

For our application we need to be able to obliviously sample public descriptions that are computationally indistinguishable from those produced by KeyGen_0 and KeyGen_1. Specifically, we require that the public descriptions that are produced by KeyGen_0 and KeyGen_1 are computationally indistinguishable from the uniform distribution. This holds, for example, in the construction of Peikert et al. [57] (based on the DDH assumption—see also [7]), and in the constructions resulting from the lossy trapdoor functions of Peikert and Waters [58] (based on the DDH and LWE assumptions) and of Freeman et al. [27] (based on the d-linear assumption).

2.6. *Admissible Hash Functions*

The concept of an *admissible hash function* was first defined by Boneh and Boyen [9] to convert a natural selectively secure identity-based encryption scheme into a fully secure one. In this paper we use such hash functions in a similar manner to convert a selectively secure signature scheme (where the adversary declares the message to be forged ahead of time, before receiving the verification key) into a fully secure one. The main idea of an admissible hash function is that it allows the reduction in the proof of security to secretly partition the message space into two subsets, which we will label as red (R) and blue (B), such that there is a noticeable probability that all of the messages in the adversary's signing queries will be in the blue set, but the forgery will be on a message in the red set. This is useful if the simulator can efficiently answer signing queries in the blue set, yet break some hard problem given a valid forgery on a message from the red set. Our exposition and definition of admissible hash function follow that of Cash, Hofheinz, Kiltz, and Peikert [16].

For $K \in \{0,1,\perp\}^{\tau(n)}$, we define the function $F_K : \{0,1\}^{\tau(n)} \to \{\mathtt{R},\mathtt{B}\}$ which "colors" the space $\{0,1\}^{\tau(n)}$ of tags in the following way:

$$F_K(y) := \begin{cases} \mathtt{R} & \text{if } \forall i \in \{1,\ldots,\tau(n)\} : K_i = y_i \text{ or } K_i = \perp \\ \mathtt{B} & \text{otherwise.} \end{cases}$$

For any $u = u(n) < \tau(n)$, we let $\mathcal{K}_{u,n}$ denote the uniform distribution over $\{0,1,\perp\}^{\tau(n)}$ *conditioned on* exactly u positions having $\perp$ values. (Note that if K is chosen from $\mathcal{K}_{u,n}$, then the map $F_K(\cdot)$ colors exactly 2^u values red.) We would like to pick a distribution $\mathcal{K}_{u,n}$ for choosing K so that there is a noticeable probability for any set of tags $y_0,\ldots,y_q$ of y_0 being colored "red" and all other tags being colored "blue". Unfortunately, this cannot happen if we allow all tags. Instead, we will need to rely on a special hash function the maps messages x to tags y.

Let $\mathcal{H} = \{\mathcal{H}_n\}_{n\in\mathbb{N}}$ be a hash-function ensemble, where each $H \in \mathcal{H}_n$ is a polynomial-time computable function $H : \{0,1\}^* \to \{0,1\}^{\tau(n)}$. For each $H \in \mathcal{H}_n$, we define the function $F_{K,H} : \{0,1\}^* \to \{\mathtt{R},\mathtt{B}\}$, which "colors" the space $\{0,1\}^*$ according to $F_{K,H}(x) = F_K(H(x))$.

Definition 2.5 (Admissible Hash Function [9,16]). We say that $\mathcal{H}$ is an *admissible hash-function ensemble* if for every $H \in \mathcal{H}$ there exists a set $\mathbf{bad}_H$ of string-tuples such that the following two properties hold:

- For every probabilistic polynomial-time algorithm $\mathcal{A}$ there exists a negligible function $\nu(n)$ satisfying

$$\Pr\left[(x_0,\ldots,x_q) \in \mathbf{bad}_H \mid H \leftarrow \mathcal{H}_n, (x_0,\ldots,x_q) \leftarrow \mathcal{A}\left(1^n, H\right)\right] \le \nu(n).$$

- For every polynomial $q = q(n)$ there is a polynomial $p = p(n)$ and an efficiently computable $u = u(n)$ such that, for every $H \in \mathcal{H}_n$ and $(x_0,\ldots,x_q) \notin \mathbf{bad}_H$ with $x_0 \notin \{x_1,\ldots,x_q\}$, we have

$$\Pr_{K \leftarrow \mathcal{K}_{u,n}}\left[F_{K,H}(x_0) = \mathtt{R} \wedge F_{K,H}(x_1) = \cdots = F_{K,H}(x_q) = \mathtt{B}\right] \ge \frac{1}{p(n)}.$$

We note that for the application to identity-based encryption [9,16] the bad sets $\mathbf{bad}_H$ are required to be efficiently recognizable, but this is not required for our application. In addition, we say that $\mathcal{H}$ is a *public-coin* admissible hash-function ensemble, if it satisfies Definition 2.5 even when the algorithm $\mathcal{A}$ takes as input also the internal randomness that was used by $\mathsf{KeyGen}(1^n)$ for sampling the function.

The work of Boneh and Boyen [9] shows how to construct admissible hash functions from collision-resistant hash functions. Moreover, if the underlying collision-resistant hash functions are public coin, then so are the resulting admissible hash functions. As already mentioned in Sect. 2.3, public-coin collision-resistant hash functions can be constructed rather easily from various number-theoretic assumptions.

3. Modeling Leakage-Resilient Signature Schemes

A signature scheme is a triplet $(\mathsf{KeyGen}, \mathsf{Sign}, \mathsf{Verify})$ of probabilistic polynomial-time algorithms with syntax:

- $(vk, sk) \leftarrow \mathsf{KeyGen}(1^n)$ outputs a verification key and signing key.
- $\sigma \leftarrow \mathsf{Sign}_{sk}(m)$ signs a message m using the singing key sk.
- $\mathsf{Verify}_{vk}(m, \sigma) \in \{0, 1\}$ outputs a bit deciding whether σ is a valid signature for m.

We require perfect correctness, which states that for any valid key pair (vk, sk) output by KeyGen and any message $m \in \{0, 1\}^*$ we have $\mathsf{Verify}_{vk}(m, \mathsf{Sign}_{sk}(m)) = 1$.

We now define fully leakage-resilient signature security in the two different models: the bounded-leakage model (see Sect. 3.1) and the continual-leakage model (see Sect. 3.2).

3.1. *The Bounded-Leakage Model*

A signature scheme is fully leakage-resilient (FLR) in the bounded-leakage model if it is existentially unforgeable against an adversary that can obtain both signatures on any message of her choice, and bounded leakage information on all intermediate values used by the key-generation algorithm and the signer throughout the lifetime of the system. To model this, we define a variable state which includes all secret state used by the system so far. Initially, we set state to be the random coins of the KeyGen algorithm (note that we do not need to explicitly add sk to the state, since it can be easily computed from it by any leakage function). On each signing query made by the adversary, we append the random coins of the signing algorithm to the state. The adversary can leak arbitrary information regarding state as long as the amount is overall-bounded.

Definition 3.1 (FLR Security—Bounded Leakage). A signature scheme $\Pi = (\mathsf{KeyGen}, \mathsf{Sign}, \mathsf{Verify})$ is λ-*fully leakage-resilient in the bounded-leakage model* if for any probabilistic polynomial-time adversary $\mathcal{A}$ we see that $\Pr[\mathsf{Success}^{\lambda\text{-}\mathsf{FLR}}_{\Pi,\mathcal{A}}(n)]$ is negligible in n, where the event $\mathsf{Success}^{\lambda\text{-}\mathsf{FLR}}_{\Pi,\mathcal{A}}(n)$ is defined via the following experiment:

1. Sample $r \leftarrow \{0, 1\}^*$, compute $(vk, sk) = \mathsf{KeyGen}(1^n; r)$, and set $\mathsf{state} = \{r\}$.
2. The adversary $\mathcal{A}$ receives as input the pair $(1^n, vk)$, and can adaptively query a signing oracle and a leakage oracle that are defined as follows:

- *Signing queries.* The signing oracle receives as input a message m_i, samples $r_i \leftarrow \{0,1\}^*$, and then computes $\sigma_i \leftarrow \mathsf{Sign}_{sk}(m_i; r_i)$. It updates $\mathsf{state} := \mathsf{state} \cup \{r_i\}$ and outputs σ_i.
- *Leakage queries.* The leakage oracle receives as input a description of an efficiently computable function $f_j : \{0,1\}^* \to \{0,1\}^{\lambda_j}$, and outputs $f_j(\mathsf{state})$. We call λ_j the *output length* of the jth leakage function.

3. The adversary $\mathcal{A}$ outputs a pair (m^*, σ^*).
4. $\mathsf{Success}^{\lambda\text{-}\mathsf{FLR}}_{\Pi,\mathcal{A}}(n)$ denotes the event in which:

- $\mathsf{Verify}_{vk}(m^*, \sigma^*) = 1$.
- m^* was not queried to the signing oracle.
- The sum of output lengths of all leakage functions is at most $\lambda(n)$.

3.2. *The Continual-Leakage Model*

In the continual-leakage model, a signature scheme also includes an additional "key-refresh algorithm" $sk' \leftarrow \mathsf{Refresh}_{pk}(sk)$, which the signer can use at any point in time to refresh his signing key. Each new signing key sk' produced by the key-refresh algorithm is functionally equivalent to the original key. We imagine a setting where the signer periodically updates her signing key, while the adversary is continuously leaking information regarding the state of the system. We model this with as an attack which consists of arbitrarily many leakage *epochs*, during each of which the adversary can learn an additional λ bits of information regarding the current state of the system. In the beginning of the first epoch, the set state consists of just the signing key sk produced by key-generation algorithm. During each epoch, the adversary can adaptively issue signing queries, where the randomness of signing algorithm is appended to the set state, and leakage queries for up to λ bits of information about the state. At any point in time the adversary can move to the next epoch by issuing a key-refresh query, which results in the set state being reset[9] to $sk \leftarrow \mathsf{Refresh}_{pk}(sk)$. Notice that, since there is no bound on the number of epochs, there is also no bound on the overall amount of leakage the adversary can learn during the attack game.

Definition 3.2 (CFLR Security—Continual Leakage). A signature scheme $\Pi = (\mathsf{KeyGen}, \mathsf{Refresh}, \mathsf{Sign}, \mathsf{Verify})$ is *λ-fully leakage-resilient in the continual-leakage model (CFLR)* if for any probabilistic polynomial-time adversary $\mathcal{A}$ we find that $\Pr[\mathsf{Success}^{\lambda\text{-}\mathsf{CFLR}}_{\Pi,\mathcal{A}}(n)]$ is negligible in n, where the event $\mathsf{Success}^{\lambda\text{-}\mathsf{CFLR}}_{\Pi,\mathcal{A}}(n)$ is defined via the following experiment:

1. Sample $(vk, sk) \leftarrow \mathsf{KeyGen}(1^n)$, and set $\mathsf{state} := \{sk\}$ and $L = 0$.
2. The adversary $\mathcal{A}$ receives as input the pair $(1^n, vk)$, and can adaptively issue the following types of queries:

- *Signing queries.* The signing oracle receives as input a message m_i, samples $r_i \leftarrow \{0,1\}^*$, and then computes $\sigma_i \leftarrow \mathsf{Sign}_{sk}(m_i; r_i)$. It updates $\mathsf{state} := \mathsf{state} \cup \{r_i\}$ and outputs σ_i.

[9] The necessary requirement that the state is reset models the ability of the honest parties to *erase/overwrite* their prior signing key and the randomness used in prior executions during a key refresh.

- *Leakage queries.* The leakage oracle receives as input a description of an efficiently computable function $f_j : \{0,1\}^* \to \{0,1\}^{\lambda_j}$. If $L + \lambda_j \leq \lambda(n)$ then it outputs $f_j(\mathsf{state})$, and updates $L := L + \lambda_j$. Otherwise it outputs $\perp$.
- *Key-refresh queries.* On a key-refresh query, the challenger samples randomness r and the signing key is refreshed by sampling $sk' \leftarrow \mathsf{Refresh}_{pk}(sk; r)$ and setting $sk := sk'$. In addition, we reset $\mathsf{state} := \{sk\}$ and $L := 0$.

3. The adversary $\mathcal{A}$ outputs a pair (m^*, σ^*).
4. $\mathsf{Success}^{\lambda\text{-}\mathsf{FLR}}_{\Pi,\mathcal{A}}(n)$ denotes the event in which:
   - $\mathsf{Verify}_{vk}(m^*, \sigma^*) = 1$.
   - m^* was never queried to the signing oracle.

We also define a stronger variant of the definition that provides *leakage-of-refreshing security*, by modifying the challenger so that, during a key-refresh query, it sets $\mathsf{state} := sk||r$ to include the new secret key sk and the randomness r used during the refreshing.

Note that our definition for the continual-leakage model does not consider leakage of the randomness used by the key-generation algorithm.[10] As shown in [15] and [18], any scheme that satisfies the basic definition (without leakage on key-generation or refreshing) with some super-logarithmic $\lambda(n)$, is also generically secure if an additional $O(\log(n))$ bits are leaked about the randomness of key-generation and *each of* the key-refresh executions. However, it seems that protecting against leakage during key generations (and to a lesser degree, during refreshing) is less crucial then protecting against leakage during the signing operations, since the former may be conducted "off-line" in a controlled environment, while the latter occur during the normal everyday execution of the scheme.

4. $\mathcal{R}$-Lossy Public-Key Encryption

In this section we introduce the notion of an $\mathcal{R}$-lossy public-key encryption scheme. Informally, such a scheme is a tag-based public-key encryption scheme where the set of possible tags is partitioned into two subsets: *injective* tags, and *lossy* tags. When a message is encrypted under an injective tag, the resulting ciphertext can be correctly decrypted using the secret key. On the other hand, when encrypted under a lossy tag, the ciphertext statistically hides the message. The partitioning of the tags in defined by a binary relation $\mathcal{R} \subseteq \mathcal{K} \times \mathcal{T}$: the key-generation algorithm receives as input an *initialization value* $K \in \mathcal{K}$ and this partitions the set tags $\mathcal{T}$ so that $t \in \mathcal{T}$ is injective if and only if $(K, t) \in \mathcal{R}$. More, formally, we require that the relation $\mathcal{R} \subseteq \mathcal{K} \times \mathcal{T}$ consists of a sequence of efficiently (in n) recognizable sub-relations $\mathcal{R}_n \subseteq \mathcal{K}_n \times \mathcal{T}_n$.

The only computational requirement of an $\mathcal{R}$-lossy public-key encryption scheme is that the public key of the encryption scheme hides the initialization value K. That is, public keys produced by different initialization values are computationally indistinguishable.

[10] This is in contrast to our bounded-leakage definition, which does consider such leakage.

Definition 4.1 ($\mathcal{R}$-Lossy PKE). Let $\mathcal{R} \subseteq \mathcal{K} \times \mathcal{T}$ be an efficiently computable binary relation. An *$\mathcal{R}$-lossy public-key encryption scheme* is a triplet of probabilistic polynomial-time algorithms $(\mathsf{KeyGen}, \mathsf{Enc}, \mathsf{Dec})$ such that:

1. *Key generation:* For any initialization value $K \in \mathcal{K}_n$, the algorithm $\mathsf{KeyGen}(1^n, K)$ outputs a secret key sk and a public key pk.
2. *Decryption under injective tags:* For any initialization value $K \in \mathcal{K}_n$ and tag $t \in \mathcal{T}_n$ such that $(K, t) \in \mathcal{R}_n$, and for any message $m \in \{0,1\}^{\ell(n)}$, we have
$$\Pr\left[\mathsf{Dec}^t_{sk}\left(\mathsf{Enc}^t_{pk}(m)\right) = m\right] > 1 - \nu(n),$$
for some negligible function $\nu(n)$, where $(sk, pk) \leftarrow \mathsf{KeyGen}(1^n, K)$, and the probability is taken over the internal randomness of KeyGen, Enc and Dec.
3. *Lossiness under lossy tags:* For any initialization value $K \in \mathcal{K}_n$ and tag $t \in \mathcal{T}_n$ such that $(K, t) \notin \mathcal{R}_n$, for every pair (sk, pk) of keys produced by $\mathsf{KeyGen}(1^n, K)$, and for every two messages $m_0, m_1 \in \{0,1\}^{\ell(n)}$, the distributions $\mathsf{Enc}^t_{pk}(m_0)$ and $\mathsf{Enc}^t_{pk}(m_1)$ are statistically indistinguishable.
4. *Indistinguishability of initialization values:* For every sequence $\{(K_n, K'_n)\}_{n\in\mathbb{N}}$ such that $K_n, K'_n \in \mathcal{K}_n$, the two ensembles $\{pk : (sk, pk) \leftarrow \mathsf{KeyGen}(1^n, K_n)\}_{n\in\mathbb{N}}$ and $\{pk : (sk, pk) \leftarrow \mathsf{KeyGen}(1^n, K'_n)\}_{n\in\mathbb{N}}$ are computationally indistinguishable.

As with the other primitives that are used in our construction, we need to be able to obliviously sample public keys in a way that is computationally indistinguishable from those produced by $\mathsf{KeyGen}(1^n, \cdot)$. Specifically, we require that there exists a sequence of initialization values $\{K_n\}_{n\in\mathbb{N}}$ such that the ensemble $\{pk : (sk, pk) \leftarrow \mathsf{KeyGen}(1^n, K_n)\}_{n\in\mathbb{N}}$ is computationally indistinguishable from the uniform distribution over $\{0,1\}^*$. Note that by the indistinguishability of initialization values property defined above, this in fact holds for every sequence $\{K_n\}_{n\in\mathbb{N}}$.

For our constructions of fully leakage-resilient signature schemes we consider two relations: the equality relation $\mathcal{R}^{\mathsf{EQ}}$, and the more general "bit-matching" relation $\mathcal{R}^{\mathsf{BM}}$ that is defined below.

The Relation $\mathcal{R}^{EQ}$ The relation $\mathcal{R}^{\mathsf{EQ}}$ is the equality relation for binary tags of length $\tau(n)$ bits. That is, $\mathcal{K}_n = \mathcal{T}_n = \{0,1\}^{\tau(n)}$, and $(K, t) \in \mathcal{R}^{\mathsf{EQ}}_n$ if and only if $K = t$. An $\mathcal{R}^{\mathsf{EQ}}$-lossy encryption is just an *all-but-one-lossy* (ALBO) public-key encryption scheme, a primitive discussed in the introduction. In this case there is one injective tag, corresponding to the value of K used during initialization, and all the other tags are lossy.

The Relation $\mathcal{R}^{BM}$ The bit-matching relation $\mathcal{R}^{\mathsf{BM}}$ is a generalization of equality, which allows for more complex partitions. For $\mathcal{K}_n = \{0,1,\perp\}^{\tau(n)}$, $\mathcal{T}_n = \{0,1\}^{\tau(n)}$ define $(K, t) \in \mathcal{R}^{\mathsf{BM}}_n \subseteq \mathcal{K}_n \times \mathcal{T}_n$ if and only if for every $i \in \{1, \ldots, \tau(n)\}$ we have $K_i = t_i$ or $K_i = \perp$. That is, given some fixed initialization value K, the set of injective tags t are exactly those whose bits match K in all positions i for which $K_i \neq \perp$. Notice that, if K does not contain any $\perp$ symbols, then there is a *single* injective tag $t = K$ and all other tags are lossy. Therefore $\mathcal{R}^{\mathsf{BM}}$-lossy encryption is a strict generalization of $\mathcal{R}^{\mathsf{EQ}}$-lossy encryption.

In our signature scheme construction, the $\mathcal{R}^{\mathsf{BM}}$-lossy encryption will be used in combination with an *admissible hash function* (discussed in Sect. 2.6). The admissible hash function gives us a way to map messages to encryption tags such that, with high probability over an appropriate distribution of K, all signing queries map to lossy tags while the forgery maps to an injective tag.

Constructions We propose two constructions of $\mathcal{R}^{\mathsf{BM}}$-lossy public-key encryption schemes.[11] Our first construction is rather generic and is based on any lossy public-key encryption scheme (recall Sect. 2.5). In turn, this implies $\mathcal{R}^{\mathsf{BM}}$-lossy public-key encryption schemes can be based on a variety of number-theoretic assumptions. Our second construction is based on a specific number-theoretic assumption (the DDH assumption)[12] and is significantly more efficient than our generic construction.

4.1. *A Generic Construction of $\mathcal{R}^{\mathsf{BM}}$-Lossy PKE from Lossy PKE*

Let $\Pi = (\mathsf{KeyGen}_0, \mathsf{KeyGen}_1, \mathsf{Enc}, \mathsf{Dec})$ be any lossy public-key encryption scheme. The key-generation algorithm of our $\mathcal{R}^{\mathsf{BM}}$-lossy public-key encryption scheme samples $\tau(n)$ pairs of public keys of the scheme Π. Each such pair is of one out of three possible types according to the symbols of the initialization value $K \in \{0, 1, \bot\}^{\tau(n)}$. For every $i \in \{1, \ldots, \tau(n)\}$, if $K_i = 0$ then the ith pair consists of a lossy key and an injective key, if $K_i = 1$ then the ith pair consists of an injective and a lossy key (i.e., the order is reversed), and if $K_i = \bot$ then ith pair consists of two injective keys.[13] A message is encrypted under a tag $t \in \{0, 1\}^{\tau(n)}$ by using a $\tau(n)$-out-of-$\tau(n)$ (information-theoretic) secret-sharing scheme to share the message, and then encrypting the ith share using one of the keys from the ith pair of keys according to the ith bit of t. More formally, consider the following encryption scheme $\Pi' = (\mathsf{KeyGen}', \mathsf{Enc}', \mathsf{Dec}')$:

- *Key generation:* On input 1^n and an initialization value $K = K_1 \cdots K_{\tau(n)} \in \{0, 1, \bot\}^{\tau(n)}$, for every $1 \leq i \leq \tau(n)$ the algorithm KeyGen' produces a pair $((sk_{i,0}, pk_{i,0}), (sk_{i,1}, pk_{i,1}))$ as follows:
  - If $K_i = 0$ then it samples $(sk_{i,0}, pk_{i,0}) \leftarrow \mathsf{KeyGen}_1(1^n)$, $pk_{i,1} \leftarrow \mathsf{KeyGen}_0(1^n)$, and sets $sk_{i,1} = \bot$.
  - If $K_i = 1$ then it samples $pk_{i,0} \leftarrow \mathsf{KeyGen}_0(1^n)$, and $(sk_{i,1}, pk_{i,1}) \leftarrow \mathsf{KeyGen}_1(1^n)$, and sets $sk_{i,0} = \bot$.
  - If $K_i = \bot$ then it samples $(sk_{i,0}, pk_{i,0}) \leftarrow \mathsf{KeyGen}_1(1^n)$ and $(sk_{i,1}, pk_{i,1}) \leftarrow \mathsf{KeyGen}_1(1^n)$.

  It then outputs the pair (sk, pk) defined as

$$sk = \left(K, \left\{(sk_{i,0}, sk_{i,1})\right\}_{i=1}^{\tau(n)}\right),$$

$$pk = \left(\left\{(pk_{i,0}, pk_{i,1})\right\}_{i=1}^{\tau(n)}\right).$$

[11] We note that rather straightforward variants of these constructions yield $\mathcal{R}^{\mathsf{EQ}}$-lossy public-key encryption schemes.

[12] Our construction easily generalizes to rely on the d-linear assumption for any $d \geq 1$.

[13] Observe that if the underlying lossy encryption scheme allows oblivious sampling of public keys, then so does our construction.

- *Encryption:* On input a public key pk of the above form, a tag $t = t_1 \cdots t_{\tau(n)} \in \{0,1\}^{\tau(n)}$ and a message $m \in \{0,1\}^{\ell(n)}$, the algorithm Enc' uses a $\tau(n)$-out-of-$\tau(n)$ (information-theoretic) secret-sharing scheme to compute shares $(m_1, \ldots, m_{\tau(n)})$ of m, and outputs a ciphertext c defined as

$$c = \left(\mathsf{Enc}_{pk_{1,t_1}}(m_1), \ldots, \mathsf{Enc}_{pk_{\tau(n),t_{\tau(n)}}}(m_{\tau(n)})\right).$$

- *Decryption:* On input a secret key sk of the above form, a tag $t = t_1 \cdots t_{\tau(n)} \in \{0,1\}^{\tau(n)}$ and a ciphertext $c = (c_1, \ldots, c_{\tau(n)})$, the algorithm Dec' proceeds as follows. If $(K, t) \notin \mathcal{R}^{\mathsf{BM}}$ (i.e., t is a lossy tag) then it outputs $\perp$. Otherwise (i.e., t is an injective tag), for every $1 \le i \le \tau(n)$ it computes $m_i = \mathsf{Dec}_{sk_{i,t_i}}(c_i)$, and uses the reconstruction procedure of the secret sharing scheme to output the message m corresponding to the shares $(m_1, \ldots, m_{\tau(n)})$.

Theorem 4.2. *If $\Pi = (\mathsf{KeyGen}_0, \mathsf{KeyGen}_1, \mathsf{Enc}, \mathsf{Dec})$ is a lossy public-key encryption scheme, then $\Pi' = (\mathsf{KeyGen}', \mathsf{Enc}', \mathsf{Dec}')$ is an $\mathcal{R}^{\mathsf{BM}}$-lossy public-key encryption scheme.*

Proof. Indistinguishability of initialization values follows directly from the indistinguishability of lossy and injective public keys of the underlying lossy encryption scheme via a straightforward hybrid argument. The correctness of the decryption algorithm under injective tags follows from the fact that when encrypting a message under an injective tag t (i.e., for every $i \in \{1, \ldots, \tau(n)\}$ we have $K_i = \perp$ or $K_i = t_i$), each share of the message is encrypted under an injective public key of the underlying lossy encryption scheme. Lossiness of encryption under lossy tags follows from the fact that when encrypting a message under any lossy tag t (i.e., there exists some $i \in \{1, \ldots, \tau(n)\}$ for which $K_i \neq \perp$ and $K_i \neq t_i$), at least one of the shares is encrypted using a lossy public key. This guarantees that the ciphertext corresponding to this share is statistically indistinguishable from an encryption of any other share under the same lossy public key. Thus, the $\tau(n)$-out-of-$\tau(n)$ (information-theoretic) secret-sharing scheme implies that encryptions of any two messages under any lossy tag are statistically indistinguishable. □

4.2. *A More Efficient Construction of $\mathcal{R}^{BM}$-Lossy PKE Based on DDH*

We now present a specific DDH-based scheme with better efficiency than the generic construction from Sect. 4.1. In this scheme, the public key still consists of $\tau(n)$ pairs, where $\tau(n)$ is the length of the tags, but each ciphertext consists of only two group elements. This scheme satisfies the requirements of Definition 4.1 with overwhelming probability over the internal randomness of the key-generation algorithm (oblivious sampling of public keys is always guaranteed), which will be sufficient for the security of our constructions in this paper. We refer to a scheme that satisfies this slightly weaker guarantee as an *almost-always* $\mathcal{R}$-lossy public-key encryption scheme. Consider the following encryption scheme $\Pi_{\mathsf{DDH}} = (\mathsf{KeyGen}_{\mathsf{DDH}}, \mathsf{Enc}_{\mathsf{DDH}}, \mathsf{Dec}_{\mathsf{DDH}})$:

- *Key generation:* On input 1^n and an initialization value $K = K_1 \cdots K_{\tau(n)} \in \{0,1,\perp\}^{\tau(n)}$, the algorithm $\mathsf{KeyGen}_{\mathsf{DDH}}$ samples $(\mathbb{G}, p, g) \leftarrow \mathsf{GroupGen}(1^n)$, together with a uniformly distributed element $h \leftarrow \mathbb{G}$. Then, for every $1 \le i \le \tau(n)$

it samples $\alpha_{i,0}, \beta_{i,0}, \alpha_{i,1}, \beta_{i,1} \leftarrow \mathbb{Z}_p$ uniformly and independently at random, and continues as follows:
- If $K_i = 0$ then it sets $\alpha_{i,0} = \beta_{i,0}$.
- If $K_i = 1$ then it sets $\alpha_{i,1} = \beta_{i,1}$.
- If $K_i = \perp$ then it sets $\alpha_{i,0} = \beta_{i,0}$ and $\alpha_{i,1} = \beta_{i,1}$.

Finally, for every $1 \leq i \leq \tau(n)$ and $b \in \{0,1\}$ it sets $(u_{i,b}, v_{i,b}) = (g^{\alpha_{i,b}}, h^{\beta_{i,b}})$, and outputs the pair (sk, pk) defined as

$$sk = \left(K, \left\{(\alpha_{i,0}, \beta_{i,0}), (\alpha_{i,1}, \beta_{i,1})\right\}_{i=1}^{\tau(n)}\right),$$
$$pk = \left(g, h, \left\{(u_{i,0}, v_{i,0}), (u_{i,1}, v_{i,1})\right\}_{i=1}^{\tau(n)}\right).$$

- *Encryption:* On input a public key pk of the above form, a tag $t = t_1 \cdots t_{\tau(n)} \in \{0,1\}^{\tau(n)}$ and a message $m \in \mathbb{G}$, the algorithm $\mathsf{Enc}_{\mathsf{DDH}}$ chooses $r, r' \in \mathbb{Z}_p$ uniformly and independently at random, computes

$$u_t = \prod_{i=1}^{\tau(n)} u_{i,t_i}, \qquad v_t = \prod_{i=1}^{\tau(n)} v_{i,t_i}, \qquad c_1 = g^r h^{r'}, \qquad c_2 = (u_t)^r (v_t)^{r'} \cdot m,$$

and outputs the ciphertext (c_1, c_2).
- *Decryption:* On input a secret key sk of the above form, a tag $t = t_1 \cdots t_{\tau(n)} \in \{0,1\}^{\tau(n)}$ and a ciphertext $c = (c_1, c_2)$, the algorithm $\mathsf{Dec}_{\mathsf{DDH}}$ proceeds as follows. If $(K, t) \notin \mathcal{R}^{\mathsf{BM}}$ (i.e., t is a lossy tag) then it outputs $\perp$. Otherwise (i.e., t is an injective tag), it outputs the message m defined as

$$m = c_2 \cdot \left(c_1^{\sum_{i=1}^{\tau(n)} \alpha_{i,t_i}}\right)^{-1}.$$

Theorem 4.3. *Assuming the hardness of the DDH problem, then* $\Pi_{\mathsf{DDH}} = (\mathsf{KeyGen}_{\mathsf{DDH}}, \mathsf{Enc}_{\mathsf{DDH}}, \mathsf{Dec}_{\mathsf{DDH}})$ *is an almost-always* $\mathcal{R}^{\mathsf{BM}}$*-lossy public-key encryption scheme for tags of length* $\tau(n) \leq \log p - \omega(\log n)$.

Proof. Indistinguishability of initialization values follows directly from the hardness of the DDH problem and a standard hybrid argument. In addition, (perfect) correctness of the decryption algorithm under injective tags follows from the fact that for any injective tag $t = t_1 \cdots t_{\tau(n)}$ (i.e., for every $i \in \{1, \ldots, \tau(n)\}$ we have $K_i = \perp$ or $K_i = t_i$) we have $\alpha_{i,t_i} = \beta_{i,t_i}$ for every $i \in \{1, \ldots, \tau(n)\}$. Therefore, for any ciphertext (c_1, c_2) that

is produced by encrypting a message m under an injective tag t we have

$$\begin{aligned}
c_2 \cdot \left(c_1^{\sum_{i=1}^{\tau(n)} \alpha_{i,t_i}}\right)^{-1} &= (u_t)^r (v_t)^{r'} \cdot m \cdot \left(\left(g^r h^{r'}\right)^{\sum_{i=1}^{\tau(n)} \alpha_{i,t_i}}\right)^{-1} \\
&= \left(\prod_{i=1}^{\tau(n)} u_{i,t_i}\right)^r \left(\prod_{i=1}^{\tau(n)} v_{i,t_i}\right)^{r'} \cdot m \cdot \left(\left(g^r h^{r'}\right)^{\sum_{i=1}^{\tau(n)} \alpha_{i,t_i}}\right)^{-1} \\
&= \left(\prod_{i=1}^{\tau(n)} g^{\alpha_{i,t_i}}\right)^r \left(\prod_{i=1}^{\tau(n)} h^{\beta_{i,t_i}}\right)^{r'} \cdot m \cdot \left(\left(g^r h^{r'}\right)^{\sum_{i=1}^{\tau(n)} \alpha_{i,t_i}}\right)^{-1} \\
&= \left(\prod_{i=1}^{\tau(n)} g^{\alpha_{i,t_i}}\right)^r \left(\prod_{i=1}^{\tau(n)} h^{\alpha_{i,t_i}}\right)^{r'} \cdot m \cdot \left(\left(g^r h^{r'}\right)^{\sum_{i=1}^{\tau(n)} \alpha_{i,t_i}}\right)^{-1} \\
&= \left(g^r h^{r'}\right)^{\sum_{i=1}^{\tau(n)} \alpha_{i,t_i}} \cdot m \cdot \left(\left(g^r h^{r'}\right)^{\sum_{i=1}^{\tau(n)} \alpha_{i,t_i}}\right)^{-1} \\
&= m.
\end{aligned}$$

Finally, we prove that lossiness under lossy tags holds with probability at least $1 - 2^{\tau(n)}/p$ over the internal randomness of the key-generation algorithm. Fix an initialization value $K \in \{0, 1, \perp\}^{\tau(n)}$, and a lossy tag $t \in \{0,1\}^{\tau(n)}$ (i.e., there exists some $i \in \{1, \ldots, \tau(n)\}$ for which $K_i \neq \perp$ and $K_i \neq t_i$). Then, there exists some $i \in \{1, \ldots, \tau(n)\}$ for which the values α_{i,t_i} and β_{i,t_i} are uniformly and *independently* chosen, and therefore the values u_t and v_t as defined by the encryption algorithm are independently and uniformly distributed in $\mathbb{G}$. Thus, with probability $1 - 1/p$, we see that (g, h, u_t, v_t) is not a DDH tuple (i.e., $\log_g(u_t) \neq \log_h(v_t)$). In this case, the elements $g^r h^{r'}$ and $(u_t)^r (v_t)^{r'}$ are also independently and uniformly distributed in $\mathbb{G}$ over the choice of r and r' (see, for example, [57, Lemma 4]). This implies that a ciphertext (c_1, c_2) encrypted under a lossy tag t carries no information on the message. This holds for any specific lossy tag t, and therefore a union bound guarantees that this holds for all lossy tags with probability at least $1 - 2^{\tau(n)}/p$. □

5. A Signature Scheme in the Bounded-Leakage Model

In this section we present a fully leakage-resilient signature scheme in the bounded-leakage model (see Definition 3.1). We use the following primitives in a generic manner:

- Let $\mathcal{F} = (\mathsf{KeyGen}_{\mathsf{SPR}}, \mathsf{F})$ be a family of public-coin second-preimage resistant functions $\mathsf{F}_s(\cdot) : \{0,1\}^{\mu(n)} \to \{0,1\}^{\kappa(n)}$ for some $\kappa(n) < \mu(n)$ (see Sect. 2.3).
- Let $\mathcal{H}$ be a public-coin admissible hash function ensemble (see Sect. 2.6).
- Let $\mathcal{E} = (\mathsf{KeyGen}_{\mathcal{R}^{\mathsf{BM}}}, \mathsf{Enc}, \mathsf{Dec})$ be an $\mathcal{R}^{\mathsf{BM}}$-lossy public-key encryption scheme (see Sect. 4).
- Let $\Pi = (\mathsf{CRSGen}, \mathsf{P}, \mathsf{V})$ be a SNIWI argument system for the language

$$L = \left\{(s, y, pk, t, C) : \exists x, \omega \text{ st } C = \mathsf{Enc}_{pk}^t(x; \omega) \text{ and } \mathsf{F}_s(x) = y\right\}$$

(see Sect. 2.4).

We assume that the distribution of public keys and common-reference strings produced by the algorithms $\mathsf{KeyGen}_{\mathcal{R}^{\mathsf{BM}}}$ and CRSGen, respectively, are computationally indistinguishable from the uniform distribution over the appropriate domains.[14] Define the signature scheme $\mathcal{S} = (\mathsf{KeyGen}, \mathsf{Sign}, \mathsf{Verify})$:

- *Key generation:* On input 1^n, the algorithm KeyGen samples a uniformly distributed $x \leftarrow \{0,1\}^{\mu(n)}$, a function description $s \leftarrow \mathsf{KeyGen}_{\mathsf{SPR}}(1^n)$ from the SPR family, and computes $y = \mathsf{F}_s(x)$. Then, it samples a description of an admissible hash function $H \leftarrow \mathcal{H}_n$, and samples $pk \leftarrow \{0,1\}^*$ and $\mathsf{crs} \leftarrow \{0,1\}^*$ to be used as a public key for the $\mathcal{R}^{\mathsf{BM}}$-lossy encryption scheme and a common-reference string for the SNIWI argument system, respectively. It outputs the signing key $sk = x$ and the verification key $vk = (s, y, H, pk, \mathsf{crs})$.
- *Signing:* On input message m, the algorithm Sign computes an encryption $C = \mathsf{Enc}_{pk}^{H(m)}(x; \omega)$ of x under the tag $H(m)$ using fresh randomness ω. Then, it invokes the prover of the argument system to obtain a proof $\pi \leftarrow \mathsf{P}(\mathsf{crs}, (s, y, pk, H(m), C), (x, \omega))$, and outputs the signature (C, π).
- *Verifying:* On input message m and signature $\sigma = (C, \pi)$, the algorithm Verify invokes the verifier of the argument system and outputs 1 if and only if $\mathsf{V}(\mathsf{crs}, (s, y, pk, H(m), C), \pi) = 1$.

Theorem 5.1. *Assuming the existence of the schemes $\mathcal{F}$, $\mathcal{H}$, $\mathcal{E}$ and Π with properties described above, the scheme $\mathcal{S} = (\mathsf{KeyGen}, \mathsf{Sign}, \mathsf{Verify})$ is λ-fully leakage-resilient in the bounded-leakage model for any $\lambda = \mu(n) - \kappa(n) - \omega(\log n)$. The relative leakage is given by $\lambda/|sk| \approx (1 - \kappa(n)/\mu(n)) = (1 - o(1))$ for an appropriate choice of $\kappa(n) = o(\mu(n))$.*

Before turning to the formal proof of Theorem 5.1 we first provide a high-level outline of the main ideas. Suppose there is an adversary who breaks the security of the scheme. We can then use the adversary to break the security of the SPR function as follows. Choose a random crs for the SNIWI argument honestly, and a (pk, sk) pair for $\mathcal{R}^{\mathsf{BM}}$-lossy public-key encryption using an initialization value K sampled from an appropriate distribution (dictated by the admissible hash function, depending on the number of signing queries the adversary makes). Given a random challenge x from the SPR challenger, we embed $y = \mathsf{F}(x)$, crs, pk into the verification key and then run the forging adversary, using x to answer all its signing/leakage queries. If the adversary's forgery is on a message m^* that corresponds to an injective tag of the encryption scheme, then we use sk to decrypt a (hopefully second preimage) x' from the adversary's forged signature. We argue that, with polynomial probability, we do recover a *second preimage* $x' \neq x$, using the following steps:

- Using the partitioning argument of Boneh–Boyen [9], there is a noticeable probability that the all of the adversary's signing queries correspond to "lossy" tags while the forgery corresponds to an "injective" tag. Here we rely on the property that the initialization value K is hidden by the public-key. We call an execution where the above occurs a "good execution."

[14] More generally, we just require "oblivious" sampling, but we will assume uniform distribution for simplicity. See Sect. 2.

- In a good execution, the adversary's forgery can be decrypted to a valid preimage $x' \in \mathsf{F}^{-1}(y)$, by the soundness of the SNIWI argument.
- Information-theoretically, the probability of $x' = x$ in a good execution is negligible, since the adversary just does not have enough information regarding x. That is, the signature-query responses are independent of x, and the leakage-query responses and the verification key y are too short. This is formalized with an entropy argument.

In terms of efficiency, in Sect. 6 we present a specific *efficient* instantiation of the scheme based on the linear assumption. In particular, the signature will consist of a *constant* number of group elements. However, our verification keys are rather large, consisting of $\Omega(n)$ group elements.

Proof of Theorem 5.1. Fix a probabilistic polynomial-time adversary $\mathcal{A}$, and let $q = q(n)$ be a polynomial upper bound on the number of signing queries made by $\mathcal{A}$ in any execution. Without loss of generality we assume that $\mathcal{A}$ always submits q signing queries, and we denote them by $m_1, \ldots, m_q$. Let $\mathcal{K}_{u,n}$ be the distribution from Definition 2.5 for the appropriate setting of $u = u(n)$ corresponding to q. The proof consists of a sequence of experiments. We analyze several events within the context of these experiments; events with the same name but different subscript are defined analogously, but within the context of the experiment indicated by the subscript. □

Experiment 0. This is the original experiment in the definition of λ-fully leakage-resilience, as described in Sect. 3. Note that the initial state is $\mathsf{state} = \{x, r_s, r_H, pk, \mathsf{crs}\}$, where r_s and r_H denote the randomness used to generate s and H, respectively. In this experiment we also sample $K \leftarrow \mathcal{K}_{u,n}$, but this value is not used by the challenger in any way.

Let Forge_0 be the event that the signature (C^*, π^*) produced by $\mathcal{A}$ at the conclusion of Experiment 0 is valid, and that $m^* \neq m_i$ for every $i \in \{1, \ldots, q\}$. In addition, let $\mathsf{CorrectHash}_0$ be the event that all signature queries $m_1, \ldots, m_q$ made by $\mathcal{A}$ during the course of Experiment 0 fall into the set of "Blue" tags defined by K, while the forged message m^* falls into the set of "Red" tags. That is, in the notation of Sect. 2.6, we have $F_{K,H}(m_1) = \cdots = F_{K,H}(m_q) = \mathtt{B}$ and $F_{K,H}(m^*) = \mathtt{R}$.

Claim 5.2. *There exists a polynomial $p(n)$ such that*

$$\Pr[\mathsf{Forge}_0 \wedge \mathsf{CorrectHash}_0] \geq \Pr[\mathsf{Forge}_0]/p(n) - \mathsf{negl}(n).$$

Proof. Using the notation of Definition 2.5, let Bad be the event $(m^*, m_1, \ldots, m_q) \in \mathbf{bad}_H$. Then Definition 2.5 clearly implies that $\Pr[\mathsf{Bad}] \leq \mathsf{negl}(n)$. Let $p(\cdot)$ be as in Definition 2.5. Then

$$\begin{aligned}
\Pr[\mathsf{Forge}_0 \wedge \mathsf{CorrectHash}_0] &\geq \Pr[\mathsf{Forge}_0 \wedge \mathsf{CorrectHash}_0 \wedge \neg\mathsf{Bad}] \\
&\geq \Pr[\mathsf{Forge}_0 \wedge \neg\mathsf{Bad}] \Pr[\mathsf{CorrectHash}_0 \mid \mathsf{Forge}_0 \wedge \neg\mathsf{Bad}] \\
&\geq \big(\Pr[\mathsf{Forge}_0] - \mathsf{negl}(n)\big)/p(n) \qquad (5.1)
\end{aligned}$$

where (5.1) follows by the definition of admissible hash functions, and since the choice of K in this experiment is independent of the events Forge_0 and Bad. □

Experiment 1. Modify Experiment 0 as follows. The key-generation algorithm does not sample pk for encryption obliviously. Instead, it uses the value K and samples $(pk, sk_E) \leftarrow \mathsf{KeyGen}_{\mathcal{R}^{\mathsf{BM}}}(1^n, K)$. Note that from the adversary's point of view the initial state remains $\mathsf{state} = \{x, r_s, r_H, pk, \mathsf{crs}\}$. Define the events Forge_1 and $\mathsf{CorrectHash}_1$ analogously to Experiment 0.

Claim 5.3. $\Pr[\mathsf{Forge}_1 \wedge \mathsf{CorrectHash}_1] \geq \Pr[\mathsf{Forge}_0 \wedge \mathsf{CorrectHash}_0] - \mathsf{negl}(n)$.

Proof. This follows by the "indistinguishability of initialization values" property of the $\mathcal{R}^{\mathsf{BM}}$-lossy public-key encryption scheme. That is, even if the choice of K is known to the adversary, it is impossible to distinguish between the case where pk is chosen obliviously (Experiment 0) and where it is chosen using the initialization value K (Experiment 1). Since the occurrence of the events Forge, $\mathsf{CorrectHash}$ can be easily computed by the attacker using its view in the experiments and the choice of K, the probabilities of these events cannot change more than negligibly between Experiments 0 and 1. □

Define the event $\mathsf{Extract}_1$ to be the event that Forge_1 and $\mathsf{CorrectHash}_1$ hold and, in addition, the signature $\sigma^* = (C^*, \pi^*)$ on the message m^* produced by $\mathcal{A}$ at the conclusion of Experiment 1 is such that $x' = \mathsf{Dec}_{sk_E}^{H(m^*)}(C^*)$ satisfies $\mathsf{F}_s(x') = y$.

Claim 5.4. $\Pr[\mathsf{Extract}_1] \geq \Pr[\mathsf{Forge}_1 \wedge \mathsf{CorrectHash}_1] - \mathsf{negl}(n)$.

Proof. Follows by the adaptive soundness of the SNIWI argument. That is, when $\mathsf{Forge}_1 \wedge \mathsf{CorrectHash}_1$ occurs but $\mathsf{Extract}_1$ does not, the proof π^* is that of a false statement. □

Define the event $\mathsf{SameExtract}_1$ to be the event that $\mathsf{Extract}_1$ occurs and that $x' = x$, where $x' = \mathsf{Dec}_{sk_E}^{H(m^*)}(C^*)$ is the extracted value and x is the secret-key used by the challenger.

Claim 5.5. $\Pr[\mathsf{SameExtract}_1] \geq \Pr[\mathsf{Extract}_1] - \mathsf{negl}(n)$.

Proof. Follows by the second pre-image resistance (SPR) of $(\mathsf{KeyGen}, \mathsf{F})$. That is, if the probability of $\mathsf{Extract}_1 \wedge \neg\mathsf{SameExtract}_1$ occurring is noticeable, than we can break the SPR-security by using the SPR-challenge (s, x) to run Experiment 1 and recovering $x' \neq x$ such that $\mathsf{F}_s(x) = \mathsf{F}_s(x')$ from the adversary's forgery (with noticeable probability). □

Experiment 2. Modify how the crs of the SNIWI argument system is generated, by using the procedure $\mathsf{CRSGen}_{WI}(1^n)$ (see Definition 2.3) instead of obliviously sampling the crs (which corresponds to $\mathsf{CRSGen}(1^n)$).

Claim 5.6. $\Pr[\mathsf{SameExtract}_2] \geq \Pr[\mathsf{SameExtract}_1] - \mathsf{negl}(n)$.

Proof. Follows by the computational indistinguishability of the distributions $\mathsf{CRSGen}(1^n)$ and $\mathsf{CRSGen}_{WI}(1^n)$. □

We now want to show that, in Experiment 2, the only information that the adversary learns about the secret-key x is from the leakage-queries, while the signature queries do not (statistically) reveal additional information. To do so, we introduce Experiments 3–6. From now on, all of our arguments will be solely *information-theoretic*, and hence we do not mind that the following experiments will *no longer be efficient*.

Experiment 3. Modify Experiment 2 by changing the response of the signing oracle. For any ciphertext-plaintext pair (C, z), define $\mathsf{Rand}_{pk}^{H(m)}(C, z)$ to be the weighted distribution $\{r : C = \mathsf{Enc}_{pk}^{H(m)}(z; r)\}$ of random values which give C as an encryption of z. When responding to a signing query m, the oracle first generates an encryption $C = \mathsf{Enc}_{pk}^{H(m)}(x; r_E)$ as in the previous experiments. It then samples a new value for the randomness $r'_E \leftarrow \mathsf{Rand}_{pk}^{H(m)}(C, x)$ and uses this value in the place of r_E in the state update process and as the witness for the proof π. Explicitly, the output signature is (C, π), where

$$C = \mathsf{Enc}_{pk}^{H(m)}(x; r_E), \qquad \pi = \mathsf{P}\big(\mathsf{crs}, \big(s, y, pk, H(m), C\big), \big(x, r'_E\big); r_\pi\big),$$

and the state is updated as $\mathsf{state} \leftarrow \mathsf{state} \cup \{r'_E, r_\pi\}$.

Note that, in the description of Experiment 3, the challenger is *no longer efficient*. However, the distribution of this modified experiment is identical to the original; this step merely introduces the randomness r'_E as a function of C and x, as opposed to viewing the ciphertext C as a function of x and r'_E. Therefore we get the following claim.

Claim 5.7. $\Pr[\mathsf{SameExtract}_3] = \Pr[\mathsf{SameExtract}_2]$.

Experiment 4. Again modify the response of the signing oracle, this time replacing the encryption C of x in each signature with a new encryption C' of a uniformly chosen preimage x' of y under $\mathsf{F}_s(\cdot)$. Explicitly, for each signature query m,

1. Choose x' uniformly at random subject to $\mathsf{F}_s(x') = y$.
2. Sample $r_E \leftarrow \{0,1\}^*$ and compute $C' \leftarrow \mathsf{Enc}_{pk}^{H(m)}(x'; r_E)$.
3. Sample $r'_E \leftarrow \mathsf{Rand}_{pk}^{H(m)}(C', x)$. Note this is with respect to the original x, and if $\mathsf{Rand}_{pk}^{H(m)}(C', x) = \emptyset$ then the experiment terminates.
4. Sample $r_\pi \leftarrow \{0,1\}^*$ and compute $\pi' = \mathsf{P}(\mathsf{crs}, (s, y, pk, H(m), C'), (x, r'_E); r_\pi)$.
5. Output the signature (C', π'), where

$$C' = \mathsf{Enc}_{pk}^{m}\big(x'; r_E\big), \qquad \pi' = \mathsf{P}\big(\mathsf{crs}, \big(s, y, pk, H(m), C'\big), \big(x, r'_E\big); r_\pi\big),$$

and update $\mathsf{state} \leftarrow \mathsf{state} \cup \{r'_E, r_\pi\}$.

Claim 5.8. $\Pr[\mathsf{SameExtract}_4] \geq \Pr[\mathsf{SameExtract}_3] - \mathsf{negl}(n)$.

Proof. Define View_i be the view of $\mathcal{A}$ in Experiment i, consisting of its random coins and observed values of the verification key, queried signatures, and leakage values.

Let $C \leftarrow \mathsf{Enc}_{pk}^{H(m_i)}(x)$ and $C' \leftarrow \mathsf{Enc}_{pk}^{H(m_i)}(x')$. Recall that $\mathsf{CorrectHash}_3$ implies the adversary's signature queries m_i correspond to lossy tags for the $\mathcal{R}^{\mathsf{BM}}$-lossy scheme. Thus, for each i the distributions (pk, C) and (pk, C') are statistically indistinguishable, say δ close. In particular, this also implies $\mathsf{Rand}_{pk}^{H(m_i)}(C, x) \neq \emptyset$ with overwhelming probability. This indistinguishability remains true even if the value of x is known.

The remainder of the adversary's view is composed of the verification key, the proofs π_i from the queried signatures, and the leakage-function evaluations $\mathsf{Leakage} = \bigcup_j f_j(\mathsf{state}_j)$. The verification key $vk = (s, y, H, pk, \mathsf{crs})$ can be computed purely as a function of x and randomness. And, in Experiments 3 and 4, the proofs π_i and leakage values are computed as a function of vk, x, independent randomness r_π, and r'_E, which in turn is selected as a function of x and the corresponding ciphertext C or C'. For each i the joint distribution $(x, C, vk, \pi_i, \mathsf{Leakage})$ must then be δ-close to that of $(x, C', vk, \pi_i, \mathsf{Leakage})$. If the adversary makes $q = \mathsf{poly}(n)$ signature queries, we will then have $(x, \mathsf{View}_3) \approx_{q\delta} (x, \mathsf{View}_4)$. Hence, $\Pr[\mathsf{SameExtract}_3] \leq \Pr[\mathsf{SameExtract}_4] + q\delta$. □

Experiment 5. Modify the response of the signing oracle by performing Steps 1–4 as in Experiment 4, then continuing as follows. Analogous to the distribution that was defined above for sampling r'_E for the $\mathcal{R}^{\mathsf{BM}}$-lossy public-key encryption scheme, let $\mathsf{Rand}_{H(m)}(\pi, x, r_E)$ be the weighted distribution $\{r : \pi = \mathsf{P}(\mathsf{crs}, (s, y, pk, H(m), C), (x, r_E); r)\}$.

5. Sample $r'_\pi \leftarrow \mathsf{Rand}_{H(m)}(\pi', x, r'_E)$.
6. Output the signature (C', π'), where

$$C' = \mathsf{Enc}_{pk}^{m}\big(x'; r_E\big), \qquad \pi' = \mathsf{P}\big(\mathsf{crs}, \big(s, y, pk, H(m), C'\big), \big(x, r'_E\big); r_\pi\big)$$

as before, but update $\mathsf{state} \leftarrow \mathsf{state} \cup \{r'_E, r'_\pi\}$ using r'_π.

Again, the distributions of Experiments 4 and 5 are identical and hence we get the following:

Claim 5.9. $\Pr[\mathsf{SameExtract}_5] = \Pr[\mathsf{SameExtract}_4]$.

Experiment 6. Modify the response of the signing oracle by replacing the proof with one using witness (x', r_E) instead of (x, r'_E). Explicitly, perform Steps 1–3 as in Experiment 3, then continue as follows.

4. Sample $r_\pi \leftarrow \{0,1\}^*$ and compute $\pi'' = \mathsf{P}(\mathsf{crs}, (s, y, pk, H(m), C'), (x', r_E); r_\pi)$.
5. Sample $r'_\pi \leftarrow \mathsf{Rand}_{H(m)}(\pi'', x, r'_E)$. If $\mathsf{Rand}_{H(m)}(\pi'', x, r'_E) = \emptyset$, the experiment terminates.

6. Output the signature (C', π''), where

$$C' = \mathsf{Enc}^{m}_{pk}\big(x'; r_E\big), \qquad \pi'' = \mathsf{P}\big(\mathsf{crs}, \big(s, y, pk, H(m), C'\big), \big(x', r_E\big); r_\pi\big),$$

and update $\mathsf{state} \leftarrow \mathsf{state} \cup \{r'_E, r'_\pi\}$.

Claim 5.10. $\Pr[\mathsf{SameExtract}_6] \geq \Pr[\mathsf{SameExtract}_5] - \mathsf{negl}(n)$.

Proof. The proof is analogous to that of Claim 5.8. Specifically, let

$$\pi' \leftarrow \mathsf{P}\big(\mathsf{crs}, \big(s, y, pk, H(m), C'\big), \big(x, r'_E\big); r_\pi\big)$$
$$\pi'' \leftarrow \mathsf{P}\big(\mathsf{crs}, \big(s, y, pk, H(m), C'\big), \big(x', r_E\big); r_\pi\big),$$

then by the statistical witness indistinguishability of the argument system, the distributions of π' and π'' are δ-close for some negligible δ, even if x is known. In particular, this implies $\mathsf{Rand}_{H(m)}(\pi'', x, r'_E) \neq \emptyset$ with overwhelming probability. Since the leakage is computed on r'_π, which is selected as a function of the proof and x, the joint distributions of

$$\big(x, vk, C', \pi', \mathsf{Leakage}\big(x, \pi'\big)\big) \approx_\delta \big(x, vk, C', \pi'', \mathsf{Leakage}\big(x, \pi''\big)\big)$$

must be δ-close. Thus, if the adversary makes q signature queries, we will have $(x, \mathsf{View}_5) \approx_{q\delta} (x, \mathsf{View}_6)$, and so $\mathsf{View}_5 \approx_{q\delta} \mathsf{View}_6$. □

As a final step in the proof, we show that x still possesses high average min-entropy conditioned on the view of $\mathcal{A}$ within Experiment 6.

Claim 5.11. $\widetilde{\mathrm{H}}_\infty(x|\mathsf{View}_6) \geq \mu(n) - \kappa(n) - \lambda$.

Proof. We consider how the average min-entropy of x decreases as the experiment progresses. At the beginning of the experiment (i.e., before the adversary submits any queries), the view of the adversary consists of the verification key vk and its own random coins $\mathsf{RandCoins}$. The experiment proceeds with a series of queries made by $\mathcal{A}$ to the leakage and signing oracles. Let $\mathsf{View}_6^{(j)}$ denote the view of the adversary within Experiment 6 after j such queries (thus $\mathsf{View}_6^{(0)} = vk||\mathsf{RandCoins}$, where $||$ denotes concatenation).

For each leakage query f_j with a λ_j-bit output, we have $\mathsf{View}_6^{(j)} = \mathsf{View}_6^{(j-1)}||f_j(\mathsf{state})$, and therefore Lemma 2.1 guarantees that $\widetilde{\mathrm{H}}_\infty(x|\mathsf{View}_6^{(j)}) \geq \widetilde{\mathrm{H}}_\infty(x|\mathsf{View}_6^{(j-1)}) - \lambda_j$. For each signature query with a message m_j, we have $\mathsf{View}_6^{(j)} = \mathsf{View}_6^{(j-1)}||m_j||\sigma_j$. First, note that the message m_j is chosen by the adversary as a function of $\mathsf{View}_6^{(j-1)}$. Second, note that in Experiment 6, each signature σ_j is computed as a function of y which is independent of x given $\mathsf{View}_6^{(j-1)}$. Therefore, the pair (m_j, σ_j) does not reduce the average min-entropy of x given $\mathsf{View}_6^{(j-1)}$. That is, $\widetilde{\mathrm{H}}_\infty(x|\mathsf{View}_6^{(j)}) = \widetilde{\mathrm{H}}_\infty(x|\mathsf{View}_6^{(j-1)})$. Thus, at the conclusion of Experiment 6 we have

$$\widetilde{\mathrm{H}}_\infty(x|\mathsf{View}_6) \geq \widetilde{\mathrm{H}}_\infty\big(\mathsf{View}_6^{(0)}\big) - \sum \lambda_j \geq \widetilde{\mathrm{H}}_\infty(x|vk, \mathsf{RandCoins}) - \lambda.$$

Now, the random coins $\mathsf{RandCoins}$ of the adversary are independent of x, and the same holds for all components of the verification key vk except $y = \mathsf{F}_s(x)$. This means $\widetilde{\mathrm{H}}_\infty(x|vk, \mathsf{RandCoins}) = \widetilde{\mathrm{H}}_\infty(x|\mathsf{F}_s(x))$. Finally, since $\mathsf{F}_s(\cdot) : \{0,1\}^{\mu(n)} \to \{0,1\}^{\kappa(n)}$, we have $\widetilde{\mathrm{H}}_\infty(x|\mathsf{F}_s(x)) \geq \mu(n) - \kappa(n)$ by Lemma 2.1. Hence, it follows that $\widetilde{\mathrm{H}}_\infty(x|\mathsf{View}_6) \geq \widetilde{\mathrm{H}}_\infty(x|\mathsf{F}_s(x)) - \lambda \geq \mu(n) - \kappa(n) - \lambda$. □

Combining Claims 5.2–5.10 above, if $\Pr[\mathsf{Forge}_0]$ is non-negligible then $\Pr[\mathsf{SameExtract}_6]$ is also non-negligible. That is, in Experiment 6, $\mathcal{A}$ is able to produce a valid signature (C^*, π^*) for which $x = \mathsf{Dec}_{sk_E}^{H(m^*)}(C^*)$ with non-negligible probability. But such a signature uniquely determines the value of x. By Claim 5.11, if $\lambda \leq \mu(n) - \kappa(n) - \omega(\log n)$, then x still has super-logarithmic average min-entropy given the view of $\mathcal{A}$ in Experiment 6, which means this is not possible even for a computationally unbounded $\mathcal{A}$. Explicitly, from the definition of average min-entropy, $\Pr[\mathsf{SameExtract}_6] \leq 2^{-(\mu(n)-\kappa(n)-\lambda)} \leq 2^{-\omega(\log n)} = \mathsf{negl}(n)$. Therefore, the event Forge_0 occurs with only a negligible probability, and this concludes the proof of Theorem 5.1. □

6. An Efficient Instantiation Based on the Linear Assumption

In this section we show that our construction, which is based on generic cryptographic primitives, can be instantiated based on specific number-theoretic assumptions to yield a rather efficient scheme. That is, we demonstrate that our approach is not only of theoretical interest (due to the argument system for general NP languages), but may also be of practical interest. We follow the approach of Dodis et al. [19] who presented rather efficient instantiations of the leakage-resilient signature scheme of Katz and Vaikuntanathan [43] using the proof system of Groth and Sahai [34]. For our scheme, this means that all of its building blocks have to be instantiated efficiently, and expressed in a form such that the resulting NP language fits the proof system of Groth and Sahai. Here we present an instantiation based on the linear assumption, and we note that an additional instantiation can be based on the seemingly less standard SXDH assumption as in [19]. In what follows we present linear-based instantiations of a family of SPR functions, and of an $\mathcal{R}^{\mathsf{BM}}$-lossy public-key encryption scheme. We then briefly describe the proof system of Groth and Sahai [34] that we use as a SNIWI argument system.

We note that our construction can be instantiated with any public-coin admissible hash function ensemble. As discussed in Sect. 2.6, such functions can be constructed based on public-coin collision-resistant hash functions, which in turn can be constructed based on the discrete log assumption in some group $\mathbb{G}$ of prime order p (which follows from the linear assumption).

6.1. *A Linear-Based Family of SPR Functions*

The following family $\mathcal{F} = (\mathsf{KeyGen}, \mathsf{F})$ of functions is based on the SPR relation of Dodis et al. [19, Appendix C.2.1].

- *Key generation:* On input 1^n the algorithm KeyGen begins by sampling $(\mathbb{G}, \mathbb{G}_T, p, g, e) \leftarrow \mathsf{BilinearGroupGen}(1^n)$. Then, it samples $s = (h_1, \ldots, h_{k(n)}, h'_1, \ldots, h'_{k(n)}) \leftarrow \mathbb{G}^{2k(n)}$ uniformly at random to be used as a public description of a function $\mathsf{F}(s, \cdot) : \mathbb{G}^{k(n)} \to \mathbb{G}_T^2$.

- *Evaluation:* On input a public description s of the above form and an input $(g_1, \ldots, g_{k(n)}) \in \mathbb{G}^{k(n)}$, the algorithm F outputs

$$\mathsf{F}\big(s, (g_1, \ldots, g_{k(n)})\big) = \left(\prod_{i=1}^{k(n)} e(h_i, g_i), \prod_{i=1}^{k(n)} e\big(h'_i, g_i\big) \right).$$

The following theorem establishes the SPR-security of these functions. The proof is essentially identical to the security proof of the SPR relation of Dodis et al. [19, Claim C.2], and is therefore omitted.

Theorem 6.1. *Assuming the hardness of the decisional linear problem, then* $\mathcal{F} = (\mathsf{KeyGen}, \mathsf{F})$ *is a family of SPR functions.*

6.2. *A Linear-Based* $\mathcal{R}^{\mathsf{BM}}$*-Lossy Public-Key Encryption Scheme*

We present a natural generalization of the scheme described in Sect. 4.2. Consider the following encryption scheme $\Pi_{\mathsf{Lin}} = (\mathsf{KeyGen}_{\mathsf{Lin}}, \mathsf{Enc}_{\mathsf{Lin}}, \mathsf{Dec}_{\mathsf{Lin}})$:

- *Key generation:* On input 1^n and an initialization value $K = K_1 \cdots K_{\tau(n)} \in \{0, 1, \perp\}^{\tau(n)}$, the algorithm $\mathsf{KeyGen}_{\mathsf{Lin}}$ samples $(\mathbb{G}, p, g) \leftarrow \mathsf{GroupGen}(1^n)$, together with three independently and uniformly chosen elements $g_1, g_2, g_3 \leftarrow \mathbb{G}$. Then, for every $1 \le i \le \tau(n)$ it samples $\alpha_{i,0}, \beta_{i,0}, \gamma_{i,0}, \alpha_{i,1}, \beta_{i,1}, \gamma_{i,1} \leftarrow \mathbb{Z}_p$ uniformly and independently at random, and continues as follows:
  - If $K_i = 0$ then it sets $\gamma_{i,1} = \alpha_{i,1} + \beta_{i,1}$.
  - If $K_i = 1$ then it sets $\gamma_{i,0} = \alpha_{i,0} + \beta_{i,0}$.
  - If $K_i = \perp$ then it sets $\gamma_{i,0} = \alpha_{i,0} + \beta_{i,0}$ and $\gamma_{i,1} = \alpha_{i,1} + \beta_{i,1}$.

  Finally, for every $1 \le i \le \tau(n)$ and $b \in \{0, 1\}$ it sets $(u_{i,b}, v_{i,b}, w_{i,b}) = (g_1^{\alpha_{i,b}}, g_2^{\beta_{i,b}}, g_3^{\gamma_{i,b}})$, and outputs the pair (sk, pk) defined as

$$sk = \Big(K, \big\{(\alpha_{i,0}, \beta_{i,0}), (\alpha_{i,1}, \beta_{i,1})\big\}_{i=1}^{\tau(n)}\Big),$$

$$pk = \Big(g_1, g_2, g_3, \big\{(u_{i,0}, v_{i,0}, w_{i,0}), (u_{i,1}, v_{i,1}, w_{i,1})\big\}_{i=1}^{\tau(n)}\Big).$$

- *Encryption:* On input a public key pk of the above form, a tag $t = t_1 \cdots t_{\tau(n)} \in \{0, 1\}^{\tau(n)}$ and a message $m \in \mathbb{G}$, the algorithm $\mathsf{Enc}_{\mathsf{Lin}}$ chooses $r, r', r'' \in \mathbb{Z}_p$ uniformly and independently at random, computes

$$u_t = \prod_{i=1}^{\tau(n)} u_{i,t_i}, \qquad v_t = \prod_{i=1}^{\tau(n)} v_{i,t_i}, \qquad w_t = \prod_{i=1}^{\tau(n)} w_{i,t_i},$$

$$c_1 = g_1^r g_3^{r''}, \qquad c_2 = g_2^{r'} g_3^{r''}, \qquad c_3 = (u_t)^r (v_t)^{r'} (w_t)^{r''} \cdot m,$$

  and outputs the ciphertext (c_1, c_2, c_3).
- *Decryption:* On input a secret key sk of the above form, a tag $t = t_1 \cdots t_{\tau(n)} \in \{0, 1\}^{\tau(n)}$ and a ciphertext $c = (c_1, c_2, c_3)$, the algorithm $\mathsf{Dec}_{\mathsf{Lin}}$ proceeds as follows. If $(K, t) \notin \mathcal{R}^{\mathsf{BM}}$ (i.e., t is a lossy tag) then it outputs $\perp$. Otherwise (i.e., t is

an injective tag), it outputs the message m defined as

$$m = c_3 \cdot \left(c_1^{\sum_{i=1}^{\tau(n)} \alpha_{i,t_i}} c_2^{\sum_{i=1}^{\tau(n)} \beta_{i,t_i}}\right)^{-1}.$$

Theorem 6.2. *Assuming the hardness of the decisional linear problem, then* $\Pi_{\mathsf{Lin}} = (\mathsf{KeyGen}_{\mathsf{Lin}}, \mathsf{Enc}_{\mathsf{Lin}}, \mathsf{Dec}_{\mathsf{Lin}})$ *is an almost-always* $\mathcal{R}^{\mathsf{BM}}$*-lossy public-key encryption scheme for tags of length* $\tau(n) \leq \log p - \omega(\log n)$.

Proof. Indistinguishability of initialization values follows directly from the hardness of the decisional linear problem and a standard hybrid argument. In addition, (perfect) correctness of the decryption algorithm under injective tags follows from the fact that for any injective tag $t = t_1 \cdots t_{\tau(n)}$ (i.e., for every $i \in \{1, \ldots, \tau(n)\}$ we have $K_i \neq t_i$) we have $\gamma_{i,t_i} = \alpha_{i,t_i} + \beta_{i,t_i}$ for every $i \in \{1, \ldots, \tau(n)\}$. Therefore, for any ciphertext (c_1, c_2, c_3) that is produced by encrypting a message m under an injective tag t we have

$$\begin{aligned}
&c_3 \cdot \left(c_1^{\sum_{i=1}^{\tau(n)} \alpha_{i,t_i}} c_2^{\sum_{i=1}^{\tau(n)} \beta_{i,t_i}}\right)^{-1} \\
&= (u_t)^r (v_t)^{r'} (w_t)^{r''} \cdot m \cdot \left(\left(g_1^r g_3^{r''}\right)^{\sum_{i=1}^{\tau(n)} \alpha_{i,t_i}} \left(g_2^{r'} g_3^{r''}\right)^{\sum_{i=1}^{\tau(n)} \beta_{i,t_i}}\right)^{-1} \\
&= \left(\prod_{i=1}^{\tau(n)} u_{i,t_i}\right)^r \left(\prod_{i=1}^{\tau(n)} v_{i,t_i}\right)^{r'} \left(\prod_{i=1}^{\tau(n)} w_{i,t_i}\right)^{r''} \\
&\quad \cdot m \cdot \left(g_1^{r \sum_{i=1}^{\tau(n)} \alpha_{i,t_i}} g_2^{r' \sum_{i=1}^{\tau(n)} \beta_{i,t_i}} g_3^{r'' \sum_{i=1}^{\tau(n)} (\alpha_{i,t_i} + \beta_{i,t_i})}\right)^{-1} \\
&= \left(g_1^{r \sum_{i=1}^{\tau(n)} \alpha_{i,t_i}} g_2^{r' \sum_{i=1}^{\tau(n)} \beta_{i,t_i}} g_3^{r'' \sum_{i=1}^{\tau(n)} \gamma_{i,t_i}}\right) \\
&\quad \cdot m \cdot \left(g_1^{r \sum_{i=1}^{\tau(n)} \alpha_{i,t_i}} g_2^{r' \sum_{i=1}^{\tau(n)} \beta_{i,t_i}} g_3^{r'' \sum_{i=1}^{\tau(n)} (\alpha_{i,t_i} + \beta_{i,t_i})}\right)^{-1} \\
&= m.
\end{aligned}$$

Finally, we prove that lossiness under lossy tags holds with probability at least $1 - 2^{\tau(n)}/p$ over the internal randomness of the key-generation algorithm. Fix an initialization value $K \in \{0, 1, \perp\}^{\tau(n)}$, and a lossy tag $t \in \{0, 1\}^{\tau(n)}$ (i.e., there exists some $i \in \{1, \ldots, \tau(n)\}$ for which $K_i = t_i$). Then, there exists some $i \in \{1, \ldots, \tau(n)\}$ for which the values α_{i,t_i}, β_{i,t_i}, and γ_{i,t_i} are uniformly and *independently* chosen, and therefore the values u_t, v_t, and w_t as defined by the encryption algorithm are independently and uniformly distributed in $\mathbb{G}$. Thus, with probability $1 - 1/p$, we find that $(g_1, g_2, g_3, u_t, v_t, w_t)$ is not an instance of the linear problem (i.e., $\log_{g_3}(w_t) \neq \log_{g_1}(u_t) + \log_{g_2}(w_t)$). In this case, the triplet $(g_1^r g_3^{r''}, g_2^{r'} g_3^{r''}, (u_t)^r (v_t)^{r'} (w_t)^{r''})$ is uniformly distributed in $\mathbb{G}^3$ over the choice of r, r', and r'' (this is a natural generalization of [57, Lemma 4]). This implies that a ciphertext (c_1, c_2, c_3) encrypted under a lossy tag t carries no information on the message. This holds for any specific lossy tag t, and therefore a union bound guarantees that this holds for all lossy tags with probability at least $1 - 2^{\tau(n)}/p$. □

6.3. *A Linear-Based SNIWI Argument System*

In this section we briefly review the proof system of Groth and Sahai [34] for proving that a system of equations is satisfiable. We mainly follow the exposition of Dodis et al. [19], and refer the reader to [34] for more details. We note that in the Groth–Sahai proof system, there are two computationally indistinguishable methods for generating the common reference string: one (called real) that yields perfect soundness, and another (called simulated) that yields perfect witness indistinguishability. By using the simulated common reference string we can thus use their system as a SNIWI argument system. We consider here the cases of one-sided multi-exponentiation equations and one-sided pairing-product equations, as these are the cases that arise from our linear-based constructions in Appendices 6.1 and 6.2.

The CRS-Generation Algorithm On input 1^n the algorithm CRSGen samples $(\mathbb{G}, \mathbb{G}_T, p, g, e) \leftarrow \mathsf{BilinearGroupGen}(1^n)$, together with three independently and uniformly distributed elements $u_0, u_1, u_2 \leftarrow \mathbb{G}$, and sets

$$\vec{u}_1 = (u_0, u_1, 1),$$
$$\vec{u}_2 = (u_0, 1, u_2).$$

Then, it samples $\vec{u}_0 \leftarrow \mathbb{G}^3 \backslash \mathfrak{U}$ and $\vec{u} \leftarrow \mathfrak{U}$ independently and uniformly at random, where $\mathfrak{U} = \{(u_0^{\alpha+\beta}, u_1^{\alpha}, u_2^{\beta}) : \alpha, \beta \in \mathbb{Z}_p\}$. It outputs $\mathsf{crs} = (\vec{u}_0, \vec{u}_1, \vec{u}_2, \vec{u})$.

Dealing with One-Sided Multi-Exponentiation Equations For every equation of the form

$$\prod_{i=1}^{k} g_i^{\chi_i} = g_0,$$

where $g_0, g_1, \ldots, g_k \in \mathbb{G}$ are constants, and $\chi_1, \ldots, \chi_k \in \mathbb{Z}_p$ are variables (i.e., the χ_i's are the satisfying assignment), the prover begins by committing to each of the χ_i's. The commitment to each $\chi_i \in \mathbb{Z}_p$ is defined as $\vec{\gamma}_i = \vec{u}^{\chi_i} \prod_{j=1}^{2} \vec{u}_j^{\,t_i^{(j)}}$, where $\vec{t}_i = (t_i^{(1)}, t_i^{(2)}) \leftarrow \mathbb{Z}_p^2$ is sampled uniformly at random. Then, the prover computes the group elements p_1 and p_2 that are defined as

$$p_j = \prod_{i=1}^{k} g_i^{t_i^{(j)}}, \quad j = 1, 2.$$

In turn, the verifier accepts if and only if for every such equation we have

$$\prod_{i=1}^{k} E(\vec{\gamma}_i, g_i) = E(\vec{u}, g_0) \prod_{j=1}^{2} E(\vec{u}_j, p_j),$$

where $E : \mathbb{G}^3 \times \mathbb{G} \to \mathbb{G}_T^3$ is defined as $E((\alpha_0, \alpha_1, \alpha_2), \beta) = (e(\alpha_0, \beta), e(\alpha_1, \beta), e(\alpha_2, \beta))$. Note that for a set of r such equations with a witness of size k, the proof consists of $3k + 2r$ group elements.

Dealing with One-Sided Pairing-Product Equations For every equation of the form

$$\prod_{i=1}^{k} e(h_i, x_i) = T,$$

where $h_1, \ldots, h_k \in \mathbb{G}$ and $T \in \mathbb{G}_T$ are constants, and $x_1, \ldots, x_k \in \mathbb{G}$ are variables (i.e., the x_i's are the satisfying assignment), the prover begins by committing to each of the x_i's. The commitment to each $x_i \in \mathbb{G}$ is defined as $\vec{\delta}_i = (x_i, 1, 1) \prod_{j=0}^{2} \vec{u}_j^{\,s_i^{(j)}}$, where $\vec{s}_i = (s_i^{(0)}, s_i^{(1)}, s_i^{(2)}) \leftarrow \mathbb{Z}_p^3$ is sampled uniformly at random, and vector multiplication is defined component-wise. Then, the prover computes the group elements p_0, p_1, and p_2 that are defined as

$$p_j = \prod_{i=1}^{k} h_i^{s_i^{(j)}}, \quad j = 0, 1, 2.$$

In turn, the verifier accepts if and only if for every such equation we have

$$\prod_{i=1}^{k} E(h_i, \vec{\delta}_i) = (T, 1, 1) \prod_{j=0}^{2} E(p_j, \vec{u}_j),$$

where $E : \mathbb{G} \times \mathbb{G}^3 \to \mathbb{G}_T^3$ is defined as $E(\alpha, (\beta_0, \beta_1, \beta_2)) = (e(\alpha, \beta_0), e(\alpha, \beta_1), e(\alpha, \beta_2))$. Note that for a set of r such equations with a witness of size k, the proof consists of $3(k + r)$ group elements.

7. A Signature Scheme in the Continual-Leakage Model

In this section, we extend our approach to the continual-leakage model. In order to do this, in Sect. 7.1 we first introduce an alternative and more general measure of leakage called "entropy leakage." Instead of measuring leakage in terms of the output length of a leakage function, we look at the entropy loss that such a function causes to a random input. In Sect. 7.2, we offer a generalized explanation of our scheme in the bounded-leakage model as a construction of leakage-resilient signatures from *leakage-resilient one-way functions* (LR-OWF). This explanation is only meant to build intuition for our construction in the continual setting, and hence the exposition will be rather informal. Finally, in Sect. 7.3, we show that our construction generalizes to constructing fully leakage-resilient signatures in the continual-leakage model from *continuous-leakage-resilient one-way relations* (in the entropy leakage sense), whose instantiations were given in [15,18,20,50,63].

7.1. *Entropy Leakage*

So far, we have measured the *amount* of leakage that the adversary learns from a function f via the output length of f. We call this *length-bounded leakage*. However, this is not the most general way of measuring leakage. As an alternative, we could consider measuring the amount of leakage via the *entropy loss* to the input of f, given the

output of f. In particular, we want our definitions to allow long leakage as long as it does not reveal too much *useful* information. The idea of measuring the entropy loss of a leakage function is due to Naor and Segev [55], but our definition is closer to that of Dodis et al. [18]. In particular, we measure the amount of information leaked by a function $f : \{0,1\}^* \to \{0,1\}^*$ in terms of the amount of entropy that it reduces from a *uniformly distributed* random variable. As shown in [18], this definition has some nice composability properties.

Definition 7.1 (λ-*Entropy Leaky* Functions). A (possibly randomized) efficiently computable function $f : \{0,1\}^* \to \{0,1\}^*$ is λ-*entropy leaky* if there exists some (possibly inefficiently computable) function f' such that:

- For all $x \in \{0,1\}^*$, $f(x) \approx_s f'(x)$ (over the randomness of f and f').
- For all integers $n \geq 1$, $\widetilde{\mathrm{H}}_\infty(U_n \mid f'(U_n)) \geq n - \lambda$, where U_n is the uniform distribution over $\{0,1\}^n$.

Notice that any function $f : \{0,1\}^* \to \{0,1\}^\lambda$ with λ-bit output is clearly λ-entropy leaky. However, there are also functions which are λ-entropy leaky but whose output lengths can be arbitrarily long. Therefore, resilience to λ bits of entropy leakage is a seemingly stronger notion of security than resilience to λ bits of length-bounded leakage (see also the discussion in Sect. 8). Also notice that a function f which is λ-entropy leaky may not, on its own, satisfy $\widetilde{\mathrm{H}}_\infty(U_n \mid f(U_n)) \geq n - \lambda$. For example, consider a function f which just output x with probability $n^{-\log(n)}$ and otherwise outputs 1. Then this function is 0-leaky according to our definition since it is statistically close to the function f' that always outputs 1 and $\widetilde{\mathrm{H}}_\infty(U_n \mid f'(U_n)) = n$. However, for the function f itself, we only have $\widetilde{\mathrm{H}}_\infty(U_n \mid f(U_n)) \leq \log^2(n)$.

The main reason is for our definition of "entropy leaky" functions is that it turns out to be more robust in some scenarios. In particular, it allows us to prove the leakage-resilient security of a signature scheme from the leakage-resilient one-way function, which we do next.

7.2. *A Generalized Explanation: FLR Signatures from LR-OWFs*

Recall that our original signature scheme is based on a second-preimage resistant (SPR) function $(\mathsf{KeyGen}, \mathsf{F})$. The verification key contains an image $y = \mathsf{F}_s(x)$ of the secret key, and each signature is essentially a proof that the signer knows a preimage of y under F_s. The important property of the SPR family is that one can easily evaluate $\mathsf{F}_s(x)$ given x (to generate the verification key), but an adversary cannot extract a preimage of y, even given "leakage" information on x learned via leakage and signature queries during the security experiment. It turns out this is the only property we need. In what follows, we abstract out the role of the SPR family in our construction to something satisfying the more general notion of a *leakage-resilient one-way function*, for a class of leakage that captures these oracle responses. As this section is only meant to provide intuition, the discussion will be somewhat informal.

LR One-Way Functions We define the notion of a *leakage-resilient one-way function* (LR-OWF) F in the following way. Given the image $y = \mathsf{F}(x)$ of a random value x in

the domain, together with λ bits of leakage on x, it is computationally hard to produce any preimage x' of y under F. We can define this notion with respect to either length-bounded leakage or entropy-bounded leakage (which is more general).

Definition 7.2 (LR-OWF). A collection $(\mathsf{KeyGen}, \mathsf{F})$ is a λ-*leakage-resilient one-way function (LR-OWF)* with respect to *entropy-bounded leakage* (respectively, *length-bounded leakage*) if for any probabilistic polynomial-time algorithm $\mathcal{A}$ and efficiently computable function g which is λ-entropy leaky (respectively, has λ-bit outputs) there exists a negligible function $\nu(\cdot)$ such that

$$\Pr\big[\mathsf{F}_s(x') = y \mid s \leftarrow \mathsf{KeyGen}(1^n), x \leftarrow \{0,1\}^{\mu(n)}, \\ y = \mathsf{F}_s(x), x' \leftarrow \mathcal{A}(s, y, g(x))\big] \leq \nu(n).$$

It is easy to show (as done in [19], generalizing the ideas of [3,43]) that a second-preimage resistant (SPR) function with $\mu(n)$-bit inputs and $\kappa(n)$-bit outputs is also a λ-leakage-resilient one-way function for $\lambda(n) = \mu(n) - \kappa(n) - \omega(\log n)$. Essentially, this is because a random x has entropy even given $y = \mathsf{F}_s(x)$ and some λ bits of leakage on x. Therefore, if there exists an algorithm that is able to produce an x' such that $\mathsf{F}_s(x') = y$ given only this information, then with noticeable probability $x' \neq x$, and we can use this adversary to break the second-preimage resistance. For the above argument, it does not matter if we consider entropy-bounded or length-bounded leakage. Therefore, we get the following observation.

Observation 7.3. *An second-preimage resistant function* $(\mathsf{KeyGen}, \mathsf{F})$ *with* $\mu(n)$*-bit inputs and* $\kappa(n)$*-bit outputs is a LR-OWF with respect to* $\lambda(n)$*-entropy-bounded leakage, where* $\lambda(n) = \mu(n) - \kappa(n) - \omega(\log n)$.

FLR Signatures from LR-OWFs We now informally explain our signature scheme from the bounded-leakage model (Sect. 5) in a slightly more general manner. Recall that our construction was based on an SPR function $(\mathsf{KeyGen}, \mathsf{F})$, where $y = \mathsf{F}_s(x)$ was in the verification key and x was the secret key. To sign a message m, we used a $\mathcal{R}^{\mathsf{BM}}$-lossy encryption scheme to encrypt x under label $H(m)$ and proved that the ciphertext is formed correctly using a SNIWI argument system. In general, we do not need $(\mathsf{KeyGen}, \mathsf{F})$ to be an SPR function, but rather any LR-OWF with respect to λ-entropy-bounded leakage.

Observation 7.4. *The construction from Sect. 5 is a* λ*-FLR signature scheme in the bounded-leakage model, when instantiated with* any *LR-OWF with respect to* λ*-entropy-bounded leakage.*

The original proof of security (Theorem 5.1) extends naturally to this more general case. To give the main intuition, let us look at the proof. Once we move from Experiment 0 to Experiment 1, there is a good chance of all the signing queries being "lossy" and the forgery still decrypting to some x^* with $\mathsf{F}_s(x^*) = \mathsf{F}_s(x) = y$, where y is in the verification key and x is in the secret key. The main idea of the rest of the proof is that, if this is the case, then the entire view of the adversary (the leakage-queries and

the signing queries) does not reduce the entropy of x significantly. Therefore, we can think of the view of the adversary as being "entropy-bounded" leakage (but not "length-bounded" leakage, since we do not know how to efficiently compress it). A successful forgery in this setting means the adversary only receives entropy-bounded leakage on x but still manages to produce (an encryption of) some pre-image x^*, thus breaking one-way security.

Although the above generalization does not seem significant at first, this view of our basic construction will make it easier to extend to the setting of continuous leakage, which we will do in the next section.

7.3. *Extension to Continuous Leakage*

We begin by generalizing the concept of a leakage-resilient one-way function to the continual leakage setting, following [18]. First, we relax the requirement that x is uniformly random and $y = \mathsf{F}(x)$ is a deterministic function of x. Instead, we define a one-way *relation* $(\mathsf{KeyGen}, \mathsf{R})$, where the KeyGen algorithm generates pairs $(y, x) \in \mathsf{R}$ simultaneously using internal randomness. The security property is similar to that of a one-way function: given a randomly generated y and leakage on x, it should be hard to find x' such that $(y, x') \in \mathsf{R}$.[15] One can keep in mind an example where x and y are a secret key and public key for a cryptographic system, and $(y, x) \in \mathsf{R}$ when x is a proper secret key corresponding to y.

Second, we add an algorithm $\mathsf{Refresh}$ that allows us to refresh the secret x. That is, if $(y, x) \in \mathsf{R}$ and $x' \leftarrow \mathsf{Refresh}(x)$ then $(y, x') \in \mathsf{R}$. The main goal of the refresh algorithm is to allow for continual leakage on the secret key. The user starts off with a key x_0 produced by the key-generation algorithm and periodically updates it by running $x_i \leftarrow \mathsf{Refresh}(x_{i-1})$. The adversary can continually receive partial leakage on *every* version x_i of the key that the user ever creates (so that the total amount of leakage learned is unbounded). Nevertheless, at no point in this process should the adversary be able to come up with some x' such that $(y, x') \in \mathsf{R}$.

Third, we consider "entropy-bounded" rather than "length-bounded" leakage. The formal definition appears below.

Definition 7.5 (CLR-OWR: Similar to [18]). A continuous-leakage-resilient one-way relation (CLR-OWR) consists of three poly-time procedures $(\mathsf{KeyGen}, \mathsf{Refresh}, \mathsf{R})$ with syntax:

- *Key Generation:* $\mathsf{KeyGen}(1^n)$ outputs a public key y and a secret key x.
- *Key Refreshing:* $\mathsf{Refresh}_y(x)$ outputs a refreshed secret key x'.
- *Relation Testing:* $\mathsf{R}(y, x)$ outputs 1 if and only if the pair (y, x) is "valid" and satisfies the relation.

We say that the relation is *continuous-leakage-resilient (CLR) with respect to* λ*-entropy-bounded leakage* if it satisfies the following properties:

[15] In the leak-free setting, any one-way relation $(\mathsf{KeyGen}, \mathsf{R})$ can easily be turned into a one-way function $y = \mathsf{KeyGen}(r)$ mapping the random coins of KeyGen to the value y it produces. In the setting of leakage, this transformation no longer holds since leakage on r gives more information than leakage on x.

Correctness: For any polynomial $q = q(n)$, if we sample $(y, x) \leftarrow \mathsf{KeyGen}(1^n)$, $x_1 \leftarrow \mathsf{Refresh}_y(x), \ldots, x_q \leftarrow \mathsf{Refresh}_y(x_{q-1})$, then, with overwhelming probability, $\mathsf{R}(y, x) = \mathsf{R}(y, x_1) = \cdots = \mathsf{R}(y, x_q) = 1$.

Security: For any PPT adversary $\mathcal{A}$, we have $\Pr[\mathcal{A} \text{ wins }] \leq \mathsf{negl}(n)$ in the following game:

- The challenger chooses $(y, x) \leftarrow \mathsf{KeyGen}(1^n)$ and gives y to $\mathcal{A}$.
- The adversary $\mathcal{A}$ runs for arbitrarily many *leakage rounds* $i = 1, 2, \ldots$. In each round i, the adversary chooses a leakage function $f_i : \{0,1\}^* \to \{0,1\}^*$ and learns $f_i(x_i)$. The next-round secret key is sampled as $x_{i+1} \leftarrow \mathsf{Refresh}_y(x_i; r_i)$ with fresh randomness r_i.
- The adversary *wins* if at some point it produces a value x^* such that $\mathsf{R}(y, x^*) = 1$ *and* each of the leakage functions f_i are λ-entropy leaky.

We say that the relation also satisfies a stronger *leakage-of-refreshing security* if security holds for a modification of the above game where the attacker learns leakage $f_i(x_i; r_{i-1})$ on the secret key x_i and the randomness r_{i-1} used during the refreshing.

The results of [15,18] show how to construct CLR-OWR under the linear assumption in bilinear groups. The leakage rate was later improved by the work of [63]. These works mostly consider leakage in between refreshing, although they can also be shown to allow some small amount (logarithmic in the security parameter) leakage on the randomness of the refreshing process itself. This issue was later improved on by the works of [20, 50] which allow a constant fraction of the key and the randomness of refreshing to leak. Although the main definitions in all of these works are with respect to length-bounded leakage, they easily extend to entropy leakage. We state the results of those works as follows:

Claim 7.6 ([20,63]). *For any polynomial $\lambda(\cdot)$ and any constant $\epsilon > 0$, there exist $\lambda(n)$-CLR-OWR schemes where the relative leakage (ratio of leakage to secret key size) is $\lambda/|sk| = (1 - \epsilon)$ under the linear assumption in bilinear groups. Under the same assumption and for any polynomial $\lambda(\cdot)$, there also exist $\lambda(n)$-CLR-OWR with the stronger "leakage-of-refreshing security", where, for some constant $c > 0$, the relative leakage is $\lambda/|sk| > c$.*

Given a CLR-OWR $(\mathsf{KeyGen}_{\mathsf{OWR}}, \mathsf{Refresh}, \mathsf{R})$, we can naturally generalize our construction of signatures from Sect. 5. Namely, the CLR-OWR will take the place of the second-preimage resistant function in the original construction, analogous to the use of the bounded-leakage-resilient one-way function in the scheme described in the previous subsection. Explicitly, let $(\mathsf{KeyGen}_{\mathcal{R}^{\mathsf{BM}}}, \mathsf{Enc}, \mathsf{Dec})$ be an $\mathcal{R}^{\mathsf{BM}}$-lossy public-key encryption scheme, and let $(\mathsf{CRSGen}, \mathsf{P}, \mathsf{V})$ be a SNIWI argument system for the language

$$L = \left\{(s, y, pk, t, C) : \exists x, \omega \text{ st } C = \mathsf{Enc}^t_{pk}(x; \omega) \text{ and } \mathsf{R}(y, x) = 1\right\}.$$

Consider the following signature scheme $(\mathsf{KeyGen}, \mathsf{Refresh}, \mathsf{Sign}, \mathsf{Verify})$:

- *Key Generation:* Sample $(pk, \cdot) \leftarrow \mathsf{KeyGen}_{\mathcal{R}^{\mathsf{BM}}}(1^n)$, $(y, x) \leftarrow \mathsf{KeyGen}_{\mathsf{OWR}}(1^n)$ and $\mathsf{crs} \leftarrow \mathsf{CRSGen}(1^n)$. Output $vk = (\mathsf{crs}, pk, y)$ and $sk = x$.

- *Key Refreshing:* Use the Refresh procedure of the CLR-OWR to compute $x' \leftarrow \mathsf{Refresh}_y(x)$.
- *Signing:* On input message m, the algorithm Sign computes an encryption $C = \mathsf{Enc}^m_{pk}(x;\omega)$ of x under the tag m using fresh randomness ω. Then, it invokes the prover of the SNIWI argument system to obtain a proof $\pi \leftarrow \mathsf{P}(\mathsf{crs}, (y, pk, m, C), (x, \omega))$, and outputs the signature (C, π).
- *Verifying:* On input message m and signature $\sigma = (C, \pi)$, the algorithm Verify invokes the verifier of the SNIWI argument system and outputs 1 if and only if $\mathsf{V}(\mathsf{crs}, (y, pk, m, C), \pi) = 1$.

Theorem 7.7. *Assume that, in the above construction, the relation is a* $(\lambda(n)+1)$*-CLR-OWR, the encryption is* $\mathcal{R}^{\mathsf{BM}}$*-lossy, and the argument system is a SNIWI. Then the above signature scheme is* $\lambda(n)$*-fully leakage-resilient in the continual leakage model. If the relation has leakage-of-refreshing security, then so does the resulting signature scheme.*

Proof of Theorem 7.7. The structure of the proof is essentially identical to the proof of Theorem 5.1. We will refer to the prior proof liberally, with a focus on the main differences. Assume that an adversary $\mathcal{A}$ breaks the signature security with a noticeable probability. We define Experiment 0 to be the continual fully leakage-resilient security game for signatures (see Definition 3.2) and Experiment 1 to be the modified game where the $\mathcal{R}^{\mathsf{BM}}$-lossy encryption scheme is initialized with an initialization value $K \leftarrow \mathcal{K}_{u,n}$ for an appropriate u corresponding to the number of signature queries q that the attacker makes. This mirrors Experiments 0 and 1 in the proof of Theorem 5.1, and the same proof shows that these experiments are indistinguishable. We can define the event $\mathsf{Extract}_1$ to occur when:

- All of the signing queries fall into the "blue"/lossy set and the forgery message falls into the "red"/injective set as defined by the initialization value K. (This corresponds to the event $\mathsf{CorrectHash}_1$ defined in the proof of Theorem 5.1.)
- The ciphertext portion of the forgery decrypts to a valid x^* for which $\mathsf{R}(y, x^*) = 1$.

As in the proof of Theorem 5.1 (specifically, Claim 5.4), the event $\mathsf{Extract}_1$ occurs with a noticeable probability. We now show how to use an adversary $\mathcal{A}$ from Experiment 1 to break the security of the CLR-OWR. Our reduction $\mathcal{B}$ samples the crs of the SNIWI argument system and the public/secret key (pk, sk) of the encryption scheme, as in Experiment 1. Recall that the adversary $\mathcal{A}$ expects to run in many epochs (periods between issuing a key-refresh query). The view of $\mathcal{A}$ during each epoch i consists of his random coins together with the signing queries and leakage queries issued during that epoch. The main idea is that the reduction $\mathcal{B}$ can simulate this view for $\mathcal{A}$ by learning a single leakage function g_i on the secret key x_i of the CLR-OWR in each epoch. The selection of g_i (described below) will ensure that:

1. The simulation perfectly matches Experiment 1. In particular, the event $\mathsf{Extract}_1$ occurs with a non-negligible probability.
2. If the event $\mathsf{Extract}_1$ occurs, then every function g_i queried by $\mathcal{B}$ is at most $(\lambda+1)$-entropy leaky.

When the event $\mathsf{Extract}_1$ occurs, then $\mathcal{B}$ can decrypt the ciphertext portion of $\mathcal{A}$'s forgery to some correct x^* such that $\mathsf{R}(y, x^*) = 1$, and win the CLR-OWR security game. Therefore the above two requirements ensure that this occurs with a non-negligible probability, which leads to a contradiction.

We are left to describe how $\mathcal{B}$ chooses the leakage functions so as to satisfy conditions (1) and (2). In epoch i, the function $g_i : \{0,1\}^* \to \{0,1\}^*$ includes, in its description, the entire view (including the random coins) of the adversary $\mathcal{A}$ up to the start of epoch i, along with the verification key $vk = (\mathsf{crs}, pk, y)$ of the signature scheme. The function $g_i(x_i)$ first checks whether $\mathsf{R}(y, x_i) = 1$ and, if not, returns a 0. Otherwise, it internally runs the code of $\mathcal{A}$ for that epoch, and uses the current secret key x_i (and internal random coins) to answer the leakage queries and the signing queries. The output of g_i consists of all the answers to the various queries asked by $\mathcal{A}$ during the epoch.

It is easy to see that this leakage can be used by $\mathcal{B}$ to (perfectly) simulate the epoch to $\mathcal{A}$, so we satisfy requirement (1). For requirement (2), note that the *output length* of g_i is long (possibly much longer than the secret key), since it includes the queried signatures. However, when the event $\mathsf{Extract}_1$ occurs, all the signing queries correspond to lossy tags of the encryption scheme, and hence do not reveal information regarding x. In particular, we can define an (inefficient) leakage function g_i' so that (for fixed x), $g_i(x) \approx_s g_i'(x)$ are statistically close, and the signature portion of $g_i'(x)$ perfectly hides x given y. This function g_i' precisely corresponds to the (inefficiently) generated responses to signature and leakage queries within Experiment 5 of the proof of Theorem 5.1.

As shown in the proof of Theorem 5.1, the only entropy loss induced by this g_i on x given y is due to the output corresponding to $\mathcal{A}$'s leakage queries, and *not* to the signature queries. Of course, for a uniformly random x, we also learn if $R(y, x) = 1$, which can reveal up to one additional bit of entropy. Therefore, since $\mathcal{A}$'s leakage queries were limited to being λ-entropy leaky, when $\mathsf{Extract}_1$ occurs, the function g_i is $(\lambda + 1)$-entropy leaky, proving (2). □

7.4. *Comparison to the Scheme of Brakerski et al.* [15]

Our construction in the continual-leakage model shares some similarities with the scheme presented by Brakerski et al. [15] in the random-oracle model. But, as we now discuss, the two are conceptually quite different. A signature in the Brakerski et al. scheme is composed of two parts: a form of "encryption" of the signing key using lossy trapdoor functions (LTDFs), and a short non-interactive argument that the encryption is formed correctly. The "encryption" portion of their signature is formed by applying message-dependent branches of a LTDF to secret shares of the signing key. Brakerski et al. prove that if the signing key is updated every few signatures, then this signature scheme satisfies a weak notion of unforgeability, where the adversary is required to specify the target forgery message prior to learning the verification key. They then use two known transformations (see [38,49]) to convert the scheme into one that is unforgeable in the more standard sense.

In the Brakerski et al. scheme, each signature leaks information regarding the signing key regardless of whether the adversary makes leakage queries. Each LTDF encryption as presented reveals an amount of information equal to the lossy parameter of the

LTDF. In addition, each non-interactive proof is treated entirely as leakage—i.e., revealing information on the signing key equal to its full length. This framework of leaking information in each signature has some unfortunate consequences. In order to keep the amount of leakage per signature small enough to maintain security, their construction requires short-length proofs, which are only known to exist within the random-oracle model [5,44,53]. In addition, since each signature reveals new information on the current signing key, the signer *must* update the signing key after every couple signatures, even if no side-channel leakage occurs. This means their construction does not yield a scheme in the bounded-leakage model where key refreshing is not a standard part of the model, and that execution within the continual-leakage model is rather inefficient. Requiring frequent key refreshes thus puts a strong restriction on the model, as the key-refreshing operation must be performed in a secure, nearly leakage-free environment.

Using the generic signature transformations to go from weak unforgeability to standard unforgeability also induces a drop in the efficiency of their scheme. For instance, one transformation requires signing each prefix of the original message, thus growing the overall signature size by a factor of the message length [38].

Our construction avoids these practical issues through two main technical differences. First, our combination of $\mathcal{R}$-lossy encryption with a SNIWI argument system (in the standard model) yields signatures which statistically reveals *no information* about the signing key (see the related discussion in Sect. 8). Second, we are able to bypass the generic signature transformations in an efficient fashion by extending the technique of admissible hash functions [9]. Essentially, the two transformations used by Brakerski et al. are replaced by simply hashing the message with a special hash function before signing. Using these new techniques, we are able to construct a scheme that is more efficient, no longer relies on the random-oracle model, and can withstand a greater fraction of leakage ($(1 - o(1))L$ as opposed to $(1/2 - o(1))L$ based on the linear assumption).

8. Concluding Remarks and Open Problems

Deterministic Leakage-Resilient Signatures An alternative approach for constructing fully leakage-resilient signature schemes is constructing a signature scheme that is resilient to leakage from the signing key, and has a deterministic signing algorithm (this is indeed the idea underlying the fully leakage-resilient *one-time* signature schemes of Katz and Vaikuntanathan [43]). In general, the signing algorithm of any signature scheme can be made deterministic by using as its random coins the output of a pseudorandom function applied to the message. This requires, however, that the signing key will include also the key of the pseudorandom function, and therefore it is not clear that such a transformation can preserve leakage resilience.

Length-Bounded Leakage vs. Entropy-Bounded Leakage In some scenarios it is not always possible to assume that the total amount of leakage is upper bounded by λ bits, where λ is less than the length of the secret key. This motivated the approach of Naor and Segev [55] (later refined by Dodis et al. [18, Definition 7.2]) who considered the more general notion of *noisy leakage* (also known as *entropy-bounded leakage*—see Sect. 7.1), in which the leakage is not necessarily of bounded length, but is guaranteed to reduce the average min-entropy of the secret key by at most λ. Although our schemes are

secure with respect to length-bounded leakage, they are in fact insecure with respect to entropy-bounded leakage. This seems to be the first separation between length-bounded and entropy-bounded leakage, and settles an open problem posed by Naor and Segev.

Specifically, in our schemes the public key for the $\mathcal{R}^{\mathsf{BM}}$-lossy encryption scheme is sampled obliviously as a uniformly random string $pk \in \{0,1\}^*$. For our specific constructions based on the DDH or linear assumptions (see Sects. 4.2 and 6.2), this can be easily seen to imply that with an overwhelming probability all possible tags for the $\mathcal{R}^{\mathsf{BM}}$-lossy scheme are lossy. An analysis almost identical to that presented in the security proofs of our schemes then shows that a leakage function that simply outputs a signature on any message m^* is a valid leakage function with respect to entropy-bounded leakage (yet clearly invalid with respect to length-bounded leakage).

Modeling Hard-to-Invert Leakage for Signature Schemes So far signature schemes were considered with respect to leakage with an information-theoretic guarantee: even after seeing the leakage, the signing key still has a certain amount of min-entropy. In the setting of public-key encryption a more general model was formalized by only assuming that the decryption key cannot be efficiently recovered given the leakage (see [13,17,22,32] and the references therein). For signature schemes, however, due to the interaction between the adversary and the signer, it is not clear how to meaningfully formalize such an attack model. It would be interesting to formalize hard-to-invert leakage for signature schemes (especially when any intermediate value may leak, and not only the signing key), and to construct schemes that are leakage resilient in such a model.

Acknowledgements

We thank Moni Naor, Brent Waters, and the anonymous referees for many useful comments on this work.

References

[1] A. Akavia, S. Goldwasser, V. Vaikuntanathan, Simultaneous hardcore bits and cryptography against memory attacks, in *Proceedings of the 6th Theory of Cryptography Conference* (2009), pp. 474–495

[2] J. Alwen, Y. Dodis, M. Naor, G. Segev, S. Walfish, D. Wichs, Public-key encryption in the bounded-retrieval model, in *Advances in Cryptology—EUROCRYPT'10* (2010), pp. 113–134

[3] J. Alwen, Y. Dodis, D. Wichs, Leakage-resilient public-key cryptography in the bounded-retrieval model, in *Advances in Cryptology—CRYPTO'09* (2009), pp. 36–54

[4] G. Ateniese, J. Camenisch, B. de Medeiros, Untraceable RFID tags via insubvertible encryption, in *Proceedings of the 12th ACM Conference on Computer and Communications Security* (2005), pp. 92–101

[5] B. Barak, O. Goldreich, Universal arguments and their applications. *SIAM J. Comput.* **38**(5), 1661–1694 (2008)

[6] M. Bellare, S. Goldwasser, New paradigms for digital signatures and message authentication based on non-interative zero knowledge proofs, in *Advances in Cryptology—CRYPTO'89* (1989), pp. 194–211

[7] M. Bellare, D. Hofheinz, S. Yilek, Possibility and impossibility results for encryption and commitment secure under selective opening, in *Advances in Cryptology—EUROCRYPT'09* (2009), pp. 1–35

[8] E. Biham, A. Shamir, Differential fault analysis of secret key cryptosystems, in *Advances in Cryptology—CRYPTO'97* (1997), pp. 513–525

[9] D. Boneh, X. Boyen, Secure identity based encryption without random oracles, in *Advances in Cryptology—CRYPTO'04* (2004), pp. 443–459

[10] D. Boneh, X. Boyen, H. Shacham, Short group signatures, in *Advances in Cryptology—CRYPTO'04* (2004), pp. 41–55
[11] D. Boneh, R.A. DeMillo, R.J. Lipton, On the importance of checking cryptographic protocols for faults, in *Advances in Cryptology—EUROCRYPT'97* (1997), pp. 37–51
[12] D. Boneh, E.-J. Goh, K. Nissim, Evaluating 2-DNF formulas on ciphertexts, in *Proceedings of the 2nd Theory of Cryptography Conference* (2005), pp. 325–341
[13] Z. Brakerski, S. Goldwasser, Circular and leakage resilient public-key encryption under subgroup indistinguishability (or: quadratic residuosity strikes back), in *Advances in Cryptology—CRYPTO'10* (2010), pp. 1–20
[14] Z. Brakerski, Y. Tauman Kalai, A framework for efficient signatures, ring signatures and identity based encryption in the standard model. Cryptology ePrint Archive, Report 2010/086, 2010
[15] Z. Brakerski, Y. Tauman Kalai, J. Katz, V. Vaikuntanathan, Cryptography resilient to continual memory leakage, in *Proceedings of the 51st Annual IEEE Symposium on Foundations of Computer Science* (2010), pp. 501–510
[16] D. Cash, D. Hofheinz, E. Kiltz, C. Peikert, Bonsai trees, or how to delegate a lattice basis, in *Advances in Cryptology—EUROCRYPT'10* (2010), pp. 523–552
[17] Y. Dodis, S. Goldwasser, Y. Tauman Kalai, C. Peikert, V. Vaikuntanathan, Public-key encryption schemes with auxiliary inputs, in *Proceedings of the 7th Theory of Cryptography Conference* (2010), pp. 361–381
[18] Y. Dodis, K. Haralambiev, A. Lopez-Alt, D. Wichs, Cryptography against continuous memory attacks, in *Proceedings of the 51st Annual IEEE Symposium on Foundations of Computer Science* (2010), pp. 511–520
[19] Y. Dodis, K. Haralambiev, A. Lopez-Alt, D. Wichs, Efficient public-key cryptography in the presence of key leakage, in *Advances in Cryptology—ASIACRYPT'10* (2010), pp. 613–631
[20] Y. Dodis, A.B. Lewko, B. Waters, D. Wichs, Storing secrets on continually leaky devices, in *Proceedings of the 52nd Annual IEEE Symposium on Foundations of Computer Science* (2011), pp. 688–697
[21] Y. Dodis, R. Ostrovsky, L. Reyzin, A. Smith, Fuzzy extractors: how to generate strong keys from biometrics and other noisy data. *SIAM J. Comput.* **38**(1), 97–139 (2008)
[22] Y. Dodis, Y. Tauman Kalai, S. Lovett, On cryptography with auxiliary input, in *Proceedings of the 41st Annual ACM Symposium on Theory of Computing* (2009), pp. 621–630
[23] S. Dziembowski, K. Pietrzak, Leakage-resilient cryptography, in *Proceedings of the 49th Annual IEEE Symposium on Foundations of Computer Science* (2008), pp. 293–302
[24] S. Faust, E. Kiltz, K. Pietrzak, G.N. Rothblum, Leakage-resilient signatures, in *Proceedings of the 7th Theory of Cryptography Conference* (2010), pp. 343–360
[25] S. Faust, T. Rabin, L. Reyzin, E. Tromer, V. Vaikuntanathan, Protecting circuits from leakage: the computationally-bounded and noisy cases, in *Advances in Cryptology—EUROCRYPT'10* (2010), pp. 135–156
[26] A. Fiat, A. Shamir, How to prove yourself: practical solutions to identification and signature problems, in *Advances in Cryptology—CRYPTO'86* (1986), pp. 186–194
[27] D.M. Freeman, O. Goldreich, E. Kiltz, A. Rosen, G. Segev, More constructions of lossy and correlation-secure trapdoor functions, in *Proceedings of the 13th International Conference on Practice and Theory in Public Key Cryptography* (2010), pp. 279–295
[28] S. Garg, A. Jain, A. Sahai, Leakage-resilient zero knowledge, in *Advances in Cryptology—CRYPTO'11* (2011), pp. 297–315
[29] S. Goldwasser, S. Micali, Probabilistic encryption. *J. Comput. Syst. Sci.* **28**(2), 270–299 (1984)
[30] S. Goldwasser, S. Micali, R.L. Rivest, A digital signature scheme secure against adaptive chosen-message attacks. *SIAM J. Comput.* **17**(2), 281–308 (1988)
[31] S. Goldwasser, G. Rothblum, How to play mental solitaire under continuous side-channels: a completeness theorem using secure hardware, in *Advances in Cryptology—CRYPTO'10* (2010), pp. 59–79
[32] S. Goldwasser, Y. Tauman Kalai, C. Peikert, V. Vaikuntanathan, Robustness of the learning with errors assumption, in *Proceedings of the 1st Symposium on Innovations in Computer Science* (2010), pp. 230–240
[33] J. Groth, R. Ostrovsky, A. Sahai, Perfect non-interactive zero knowledge for NP, in *Advance in Cryptology—EUROCRYPT'06* (2006), pp. 339–358
[34] J. Groth, A. Sahai, Efficient non-interactive proof systems for bilinear groups, in *Advances in Cryptology—EUROCRYPT'08* (2008), pp. 415–432

[35] J.A. Halderman, S.D. Schoen, N. Heninger, W. Clarkson, W. Paul, J.A. Calandrino, A.J. Feldman, J. Appelbaum, E.W. Felten, Lest we remember: cold boot attacks on encryption keys, in *Proceedings of the 17th USENIX Security Symposium* (2008), pp. 45–60
[36] B. Hemenway, B. Libert, R. Ostrovsky, D. Vergnaud, Lossy encryption: constructions from general assumptions and efficient selective opening chosen ciphertext security, in *Advances in Cryptology—ASIACRYPT'11* (2011), pp. 70–88
[37] N. Heninger, H. Shacham, Reconstructing RSA private keys from random key bits, in *Advances in Cryptology—CRYPTO'09* (2009), pp. 1–17
[38] S. Hohenberger, B. Waters, Short and stateless signatures from the RSA assumption, in *Advances in Cryptology—CRYPTO'09* (2009), pp. 654–670
[39] C.-Y. Hsiao, L. Reyzin, Finding collisions on a public road, or do secure hash functions need secret coins, in *Advances in Cryptology—CRYPTO'04* (2004), pp. 92–105
[40] Y. Ishai, A. Sahai, D. Wagner, Private circuits: securing hardware against probing attacks, in *Advances in Cryptology—CRYPTO'03* (2003), pp. 463–481
[41] A. Joux, K. Nguyen, Separating decision Diffie–Hellman from computational Diffie–Hellman in cryptographic groups. *J. Cryptol.* **16**(4), 239–247 (2003)
[42] A. Juma, Y. Vahlis, On protecting cryptographic keys against side-channel attacks, in *Advances in Cryptology—CRYPTO'10* (2010), pp. 41–58
[43] J. Katz, V. Vaikuntanathan, Signature schemes with bounded leakage resilience, in *Advances in Cryptology—ASIACRYPT'09* (2009), pp. 703–720
[44] J. Kilian, A note on efficient zero-knowledge proofs and arguments (extended abstract), in *Proceedings of the 22nd Annual ACM Symposium on Theory of Computing* (1992), pp. 723–732
[45] E. Kiltz, Chosen-ciphertext secure key-encapsulation based on gap hashed Diffie–Hellman, in *Proceedings of the 10th International Conference on Practice and Theory in Public-Key Cryptography* (2007), pp. 282–297
[46] P.C. Kocher, Timing attacks on implementations of Diffie–Hellman, RSA, DSS, and other systems, in *Advances in Cryptology—CRYPTO'96* (1996), pp. 104–113
[47] P.C. Kocher, J. Jaffe, B. Jun, Differential power analysis, in *Advances in Cryptology—CRYPTO'99* (1999), pp. 388–397
[48] G. Kol, M. Naor, Cryptography and game theory: designing protocols for exchanging information, in *Proceedings of the 5th Theory of Cryptography Conference* (2008), pp. 320–339
[49] H. Krawczyk, T. Rabin, Chameleon signatures, in *Proceedings of the Network and Distributed System Security Symposium (NDSS)* (2000)
[50] A.B. Lewko, M. Lewko, B. Waters, How to leak on key updates, in *Proceedings of the 43rd Annaul ACM Symposium on Theory of Computing* (2011), pp. 725–734
[51] V. Lyubashevsky, A. Palacio, G. Segev, Public-key cryptographic primitives provably as secure as subset sum, in *Proceedings of the 7th Theory of Cryptography Conference* (2010), pp. 382–400
[52] T. Malkin, I. Teranishi, Y. Vahlis, M. Yung, Signatures resilient to continual leakage on memory and computation, in *Proceedings of the 8th Theory of Cryptography Conference* (2011), pp. 89–106
[53] S. Micali, Computationally sound proofs. *SIAM J. Comput.* **30**(4), 1253–1298 (2000)
[54] S. Micali, L. Reyzin, Physically observable cryptography, in *Proceedings of the 1st Theory of Cryptography Conference* (2004), pp. 278–296
[55] M. Naor, G. Segev, Public-key cryptosystems resilient to key leakage, in *Advances in Cryptology—CRYPTO'09* (2009), pp. 18–35
[56] M. Naor, M. Yung, Universal one-way hash functions and their cryptographic applications, in *Proceedings of the 21st Annual ACM Symposium on Theory of Computing* (1989), pp. 33–43
[57] C. Peikert, V. Vaikuntanathan, B. Waters, A framework for efficient and composable oblivious transfer, in *Advances in Cryptology—CRYPTO'08* (2008), pp. 554–571
[58] C. Peikert, B. Waters, Lossy trapdoor functions and their applications. *SIAM J. Comput.* **40**(6), 1803–1844 (2011)
[59] K. Pietrzak, A leakage-resilient mode of operation, in *Advances in Cryptology—EUROCRYPT'09* (2009), pp. 462–482
[60] J. Rompel, One-way functions are necessary and sufficient for secure signatures, in *Proceedings of the 22nd Annual ACM Symposium on Theory of Computing* (1990), pp. 387–394
[61] H. Shacham, A Cramer–Shoup encryption scheme from the linear assumption and from progressively weaker linear variants. Cryptology ePrint Archive, Report 2007/074 (2007)

[62] V. Shoup, Lower bounds for discrete logarithms and related problems, in *Advances in Cryptology—EUROCRYPT'97* (1997), pp. 256–266

[63] Y. Tauman Kalai, B. Kanukurthi, A. Sahai, Cryptography with tamperable and leaky memory, in *Advances in Cryptology—CRYPTO'11* (2011), pp. 373–390

[64] B. Waters, Efficient identity-based encryption without random oracles, in *Advances in Cryptology—EUROCRYPT'05* (2005), pp. 114–127