Leakage-Resilient Public-Key Cryptography in the Bounded-Retrieval Model

Joël Alwen, Yevgeniy Dodis, and Daniel Wichs

Department of Computer Science, New York University.
{jalwen,dodis,wichs}@cs.nyu.edu

Abstract. We study the design of cryptographic primitives resilient to key-leakage attacks, where an attacker can repeatedly and adaptively learn information about the secret key, subject *only* to the constraint that the *overall amount* of such information is bounded by some parameter ℓ. We construct a variety of leakage-resilient public-key systems including the first known identification schemes (ID), signature schemes and authenticated key agreement protocols (AKA). Our main result is an efficient three-round AKA in the Random-Oracle Model, which is resilient to key-leakage attacks that can occur prior-to *and* after a protocol execution. Our AKA protocol can be used as an interactive encryption scheme with qualitatively stronger privacy guarantees than non-interactive encryption schemes (constructed in prior and concurrent works), which are inherently insecure if the adversary can perform leakage attacks after seing a ciphertext.

Moreover, our schemes can be flexibly extended to the *Bounded-Retrieval Model*, allowing us to tolerate very large absolute amount of adversarial leakage ℓ (potentially many gigabytes of information), *only* by increasing the size of the secret key and without any other loss of efficiency in communication or computation. Concretely, given any leakage parameter ℓ, security parameter λ, and any desired fraction $0 < \delta \leq 1$, our schemes have the following properties:

- Secret key size is $\ell(1 + \delta) + O(\lambda)$.
- Public key size is $O(\lambda)$, and independent of ℓ.
- Communication complexity is $O(\lambda/\delta)$, and independent of ℓ.
- Computation reads $O(\lambda/\delta^2)$ locations of the secret key, independent of ℓ.

Lastly, we show that our schemes allow for repeated *"invisible updates"* of the secret key, allowing us to tolerate up to ℓ bits of leakage in between any two updates, and an unlimited amount of leakage overall. These updates require that the parties can securely store a short "master update key" (e.g. on a separate secure device protected against leakage), which is only used for updates and not during protocol execution. The updates are invisible in the sense that a party can update its secret key at any point in time, *without* modifying the public key or notifying the other users.

1 Introduction

Traditionally, cryptographic systems rely on complete privacy of cryptographic keys. Unfortunately, this idealized assumption is often hard to satisfy in real systems. In many situations, the attacker might get some partial information about secret keys through means which were not anticipated by the designer of the system and, correspondingly, not taken into account when arguing its security. Such attacks, referred to as *key-leakage* *attacks*, come in a large variety. For example, this includes *side-channel* attacks [20, 21, 28], where an adversary observes some "physical output" of a computation (radiation, power, temperature, running time etc.) in addition to the "logical output" of the computation. Alternatively, this also includes the "cold-boot" attack of Halderman et al. [15], where an adversary can learn (imperfect) information about memory contents, even after a machine is powered down. Lastly, this can include various malware/virus/hacking attacks where the adversary can download arbitrary information from an attacked computer.

Given that one cannot hope to eliminate the problem of leakage attacks altogether, it is natural to design leakage-resilient cryptographic schemes which remain (provably) secure, even in the face of such attacks. To do so, we must first decide on an appropriate model of what information the adversary can learn during a leakage attack. In this work, we assume that the attacker can repeatedly and adaptively learn *arbitrary functions* of the secret key sk, as long as the *total* number of bits leaked during the lifetime of the system is bounded by some parameter ℓ. Due to its generality, this model seems to include a very large class of attacks mentioned above, and has recently attracted a lot of attention from the research community. In particular, this model simultaneously covers the following two typical scenarios, which seem to be treated differently in the existing literature.

Relative Leakage. Here, the secret key is chosen to be of some particular length s, which depends on the security parameter, and we assume that the leakage ℓ is bounded by some shrinking function of s; e.g., the attacker's leakage is less than half of the key-size. This assumption seems to be natural for modeling attacks where, no matter what the key-size is, the attacker gets some imperfect reading of the key. For example, this naturally models "cold boot attacks" attacks [15] (where the attacker might get part of the key stored in RAM) and "microwave" attacks (where the attacker manages to extract a corrupted copy of the key from a smart-card), among others.

Bounded-Retrieval Model (BRM). Here we assume that there is an external natural bound ℓ on the overall amount of information the attacker can learn throughout the lifetime of the system, particularly concentrating on the setting when ℓ can be extremely large. For example, the attacker may be able to repeatedly perform many side-channel attacks, each of which reveals a few bits of information about the key but, if the bandwidth of such attacks is relatively small, it may be infeasible, too time consuming, or simply not cost-affective for the adversary to learn "too much" information (say, more than 10 megabytes) overall. Alternatively, if an attacker hacks into a remote system (or infects it with some malware) it may again be infeasible/impractical for the attacker to download "too much" data (say, more than 10 gigabytes). In these situations the leakage bound ℓ is decided by external factors and one can only resist such attacks by making the secret key *intentionally large*, to dominate ℓ. Therefore, we want to be able to set the key size flexibly depending on the security parameter *and* the leakage bound ℓ. By itself, having large secret-keys might not be a big problem for usability, given the extremely cheap price of storage nowadays. Therefore, the main goal of this setting, usually refereed to as the *Bounded-Retrieval Model* (BRM) [8, 12], is to ensure that the *necessary* inefficiency in storage is essentially the *only* inefficiency that the users of the system incur. In particular, honest users should only have to read a small portion of the secret (this is called *locality*), and the public keys, communication and computation should not be much larger than in conventional cryptosystems. In particular, all efficiency parameters other than the secret-key size should *only* be proportional to the security parameter, and *not* the leakage bound ℓ.

To summarize, both leakage models (relative and BRM) study essentially the same technical question. However, the BRM setting additionally demands that: *users can increase their secret key size flexibly, so as to allow for an arbitrary large leakage bounds ℓ, but without degrading other efficiency parameters, such as computation, communication and locality.* This is the perspective we will take in this paper, treating both settings together, while striving to allow for the flexibility of the BRM.

NOTIONS OF SECURITY. Security with respect to key leakage attacks can be defined for nearly all cryptographic primitives (e.g. encryption, signatures, authenticated key agreement . . .) However, for many of the above primitives, there are natural limitations on the security notions that can be achieved in the presence of such attacks. For example, encryption schemes lose all privacy if the adversary can perform leakage attacks *after* seeing a ciphertext, since the leakage function can simply decrypt it and reveal some information about plaintext. Similarly, one cannot achieve *existential unforgeability* for signature schemes if the leakage bound ℓ is larger than the size of a single signature (as is the case in the BRM), since the adversary can simply leak the signature of a message of her choosing. These limitations do not seem to apply when considering interactive primitives, and therefore we choose to concentrate on *authenticated key agreement (AKA)*, which in turn allows for interactive encryption and authentication, and achieves *qualitatively* stronger security guarantees, even in the BRM.

1.1 Our Results

Our main result is the construction of a leakage-resilient public-key authenticated key agreement (AKA) protocol with the flexibility required by the BRM. We assume a public-key infrastructure where users have short public-keys and flexibly sized (potentially huge) secret keys. In a leakage-resilient AKA protocol, pairs of users agree on shared session-keys which are private and authentic, even if: (a) the attacker learns at most ℓ bits of information about the secret keys of both users *prior to* the protocol execution; (b) the attacker may learn the secret keys entirely *after* the protocol execution. In particular, condition (a) ensures that the adversary cannot *impersonate* an honest user, even after learning ℓ bits of leakage about that user's secret key. Since the shared session-keys can safely be used for encryption/authentication, a public-key AKA naturally yields *interactive* public-key encryption and authentication schemes which are secure under assumptions (a) and (b), and do not suffer from the inherent limitations of their non-interactive counterparts.

ROADMAP OF AKA CONSTRUCTION. Our construction of AKA is based on simpler primitives and, in particular, we also construct identification schemes (ID) and (non-interactive) signature schemes, which are of interest in their own right. The main technical portion of our paper will be the construction of ID schemes secure against leakage attacks. We then apply the Fiat-Shamir heuristic to obtain efficient leakage-resilient signature schemes in the random oracle (RO) model. Of course, our signature schemes cannot provide *existential unforgeability*, if the allowed leakage exceeds the size of even a single signature (which is usually the case in the BRM). Interestingly, we show how to achieve existential unforgeability under this necessary constraint, which resolves an open problem mentioned in [1]. For the BRM setting, which is our main point of interest, we must settle for a weaker, but still very useful security notion, that we call *entropic unforgeability*. Although an attacker may be able to forge signatures for a few messages after she performs a key-leakage attack, she should be unable to forge the signature of a random message sampled from any distribution of high enough min-entropy.

Finally, we use a standard construction of AKA based on Diffie-Hellman key exchange, in which the parties bind the protocol execution to a particular session and to their identities using signatures. We plug our entropically secure signature scheme into this construction to get leakage-resilient AKA. Intuitively, the usage of entropically secure signature will suffice, since each party only signs messages which are partially controlled by the other party, and happen to have entropy. We note that our constructions of authenticated key agreement from entropic signatures, and our constructions of such signatures from ID schemes, are extremely efficient and essentially preserve (1) the long-term public/secret key size, (2) the communication complexity, (3) the locality, and (4) the allowed leakage. Therefore, we apply most of our efforts to the construction of optimized, leakage-resilient ID schemes.

ID SCHEME CONSTRUCTIONS. We present three ID scheme constructions, which build on top of one another. First we notice that a generalization of the discrete-log based Okamoto ID scheme [25] using m generators, denoted $\mathrm{Okamoto}_m^\lambda$, is secure against leakage attacks where the allowed leakage is $\ell \approx (1 - \frac{1}{m})|\mathsf{sk}|$, and can be set arbitrarily close to the size of the secret key. Our argument relies on the following three simple properties of the scheme:

(1) Any adversarial prover that impersonates some identity must *know* a corresponding secret key for the identity's public key.

(2) For any Okamoto public key, there are (exponentially) many corresponding secret keys. Moreover, the actual secret key of the scheme maintains a high level of *information-theoretic entropy*, even when given: (a) the public key, (b) protocol executions between adversarial verifier and honest prover, and (c) ℓ bits of secret-key leakage.

(3) It is computationally infeasible to come up with an Okamoto public key and two *different* corresponding secret keys.

By property (1), an adversarial prover that successfully mounts impersonation attacks knows *some* secret key for the honest user's public key and, by property (2), this secret key is (information theoretically) unlikely to match the one possessed by the honest user, *even if the adversary got ℓ bits leakage*. Therefore, the adversarial prover together with the honest user can generate two different secret keys for a single Okamoto public key, contradicting property (3) and hence proving security. We note that several other identification schemes (e.g. an alternate construction by Okamoto [25] based on RSA, and the Ong-Schnorr [26] scheme based on factoring) also have the three mentioned properties and are therefore leakage-resilient as well.

While the (generalized) Okamoto scheme already provides an adequate solution for *relative* leakage, it cannot achieve large *absolute* leakage, without a proportional increase in communication complexity and locality. Therefore, we present two extensions of the Okamoto scheme which are suitable to the BRM setting. The first extension, denoted $\mathrm{DirProd}^{\lambda}_{n,m,t}$, is based on the idea of taking a "direct-product" of n basic Okamoto schemes, where the verifier will selects a small random subset of $t \ll n$ of these schemes, and executes the basic protocol for them in parallel. One can think of this as a simple form of *"leakage amplification"*, where we amplify the amount of allowed absolute leakage. Lastly, we improve the communication complexity of this second scheme still further (in the Random Oracle model), by showing how to use ideas from coding-theory and the special structure of the Okamoto scheme, to "securely compress" the t chosen Okamoto public keys into a *single* public key, and then running a *single* ID protocol for this key. Therefore, and quite remarkably, our third scheme, denoted $\mathrm{CompDirProd}^{\lambda}_{n,m,t}$, has essentially the same communication complexity as the basic (non-BRM) Okamoto scheme even though the allowed leakage ℓ can be made arbitrarily large.

Scheme	pub. params	pk	sk	help	Comm.	Loc.	Leakage ℓ (in bits)
$\mathrm{Okamoto}^{\lambda}_{m}$	m	1	m	0	$m + O(1)$	m	$(1-\delta)\lvert\mathsf{sk}\rvert$ $\delta \approx \frac{1}{m}$
$\mathrm{DirProd}^{\lambda}_{n,m,t}$	m	1	nm	n	$O(tm)$	tm	$(1-\delta)\lvert\mathsf{sk}\rvert$ $\delta \approx \left(\frac{1}{m} + O\left(\frac{\lambda}{t}\right)\right)$
$\mathrm{CompDirProd}^{\lambda}_{n,m,t}$	m	1	nm	n	$m + O(1)$	tm	$(1-\delta)\lvert\mathsf{sk}\rvert$ $\delta \approx \left(\frac{1}{m} + O\left(\frac{\lambda}{t}\right)\right)$

Table 1. Efficiency vs. Leakage Trafeoffs For Our ID, Sig, AKA schemes

OVERVIEW OF ACHIEVED PARAMETERS. We summarize the main parameters of the three ID scheme constructions (which translate into essentially the same parameters for the corresponding signatures and AKA protocols) in Table 1. The columns indicate the sizes of: the public parameters shared by all users, the public key, the secret key, a helper key (which is stored locally by each user, but does not have to be kept secret), the communication complexity per party (or signature size), the locality, and the allowed leakage ℓ. For simplicity, only the leakage parameter ℓ is measured in bits, and all other quantities are measures in group elements. The parameters m, n, t offer *flexibility* to meet the various desired settings of of absolute leakage ℓ and relative leakage $(1-\delta)$. In particular:

- For the first scheme ($\mathrm{Okamoto}^{\lambda}_{m}$), the only flexibility is in the number of generators m. Essentially to allow for relative leakage $(1-\delta)$ we can set $m \approx 1/\delta$ which gives us very practical schemes for reasonable settings of the relative leakage (e.g. $\delta = \frac{1}{2}$). However, to allow for a large absolute leakage ℓ, we must increase m still further (and proportionally with ℓ), which increases the communication, computation and size of public parameters to unreasonable levels.
- For the second and third scheme ($\mathrm{DirProd}^{\lambda}_{n,m,t}, \mathrm{CompDirProd}^{\lambda}_{n,m,t}$), we have the additional flexibility offered by parameters n (the number of stored copies of Okamoto key pairs) and t (the number of Okamoto keys used during a particular protocol). We notice that, by setting $m \approx 1/\delta, t \approx O(\lambda/\delta)$ we allow a relative leakage of $(1-\delta)$ and still get practical schemes with small public parameters, public key size, communication (especially in the third scheme), and locality. Moreover, we can then flexibly accommodate *any* value of the absolute leakage ℓ *only* by increasing n which *only* affects the size of the secret key.

INVISIBLE KEY UPDATES. Lastly, we mention a simple but powerful feature of our schemes. We introduce a method for users to periodically *update* their secret keys, so that the scheme remains secure as long as the adversary learns at most $\ell \approx (1-\delta)|\mathsf{sk}|$ bits of key leakage *in between updates*, but may learn leak significantly more than the size of the secret key *overall*. Our updates are *invisible* to the outside world, in the sense that the public keys remain unchanged and users do not need to know when or how often the secret keys of other users are updated in order to run an AKA protocol. For such updates, we require the use of a *"master update key"* which must be stored securely on an external storage device that is not susceptible to leakage attacks.

1.2 Related Work

WEAK SECRETS, SIDE-CHANNEL ATTACKS AND BRM. The model of key leakage attacks, as studied in this work, is very related to the study of cryptography with *weak secrets*. A weak secret is one which comes from some arbitrary distribution that has a sufficient level of (min-)entropy, and one can think of a secret key that has been partially compromised by leakage attacks as coming from such a distribution. Most of the prior work concerning weak secrets is specific to the *symmetric key setting* and much of this work is *information-theoretic in nature*. For example, the study of privacy-amplification [3, 22] shows how two users, who *share* a weak secret, can agree on a uniformly random key in the presence of a passive attacker. Such information-theoretically secure schemes can only be used *once* to convert a shared symmetric-key, which may have been partially compromised by leakage attacks, into a *single* uniform session-key.

In the computational symmetric-key setting, users can agree on *arbitrarily many* session-keys using Password Authenticated Key Agreement (PAKE) [2], where they use their shared weak (or partially compromised) secret key as the password. However, these solutions do not scale to the BRM, as they do not preserve low locality when the secret is large. The Bounded-Retrieval Model (BRM), where users have a huge secret key which is subject to large amounts of adversarial leakage, was introduced by [8, 12]. In particular, Dziembowski [12] constructed a *symmetric key* authenticated-key-agreement protocol for this setting in the Random-Oracle model. This was later extended to the standard model by [7]. We also note that *non-interactive* symmetric-key encryption schemes from weakly-secret keys were constructed implicitly in [27] (based on weak PRFs) and explicitly in [9] based on "learning parity with noise".

The only related prior work that considers leakage attacks in the *public-key* setting is a recent work of Akavia et al. [1], which showed that Regev's public-key encryption scheme [29] (based on lattices) is leakage-resilient. Several recent concurrent and independent works [24, 18, 19] also study leakage-resilient public-key primitives. The works of [24, 18] present several new constructions of of leakage-resilient public-key encryption schemes for this setting, based on more general (and non-lattice) assumptions, tolerating more leakage, achieving CCA-2 security and allowing for stronger "auxiliary-input" attacks (described subsequently). We note that all such non-interactive encryption schemes inherently become insecure if the adversary can perform leakage attacks *after* seeing a ciphertext. Existentially unforgeable leakage-resilient signatures were studied in the concurrent work of Katz [19], who independently discovered our $\mathrm{Okamoto}_m^\lambda$ construction (described above) of signatures in the Random Oracle model, as well as an alternative (albeit not practically efficient) instantiation of such signatures in the standard model. None of the prior or concurrent works in the public-key setting extend to the Bounded Retrieval Model.

OTHER MODELS OF ADVERSARIAL KEY COMPROMISE. It is worth describing several related models for key compromise which differ from the one used in this work. One possibility is to restrict the *type* of information that the adversary can learn about the secret key. For example, a line of work called *exposure resilient cryptography* [5, 11] studies a restricted class of adversarial leakage functions, where the adversary gets a *subset of the bits* of the secret key. In this setting, one can secure keys against leakage generically, by encoding them using an *all-or-nothing transform (AONT)*. We note that many natural side-channel attacks (e.g. learning the hamming weight of the key) and most malware attacks are not captured by this model.

Another line of work, initiated by Micali and Reyzin [23] and studied further by Dziembowski and Pietrzak [13, 27], designs various symmetric-key primitives under the axiom that "only computation leaks information". In these works, each stage of computation is assumed to leak some arbitrary shrinking function of (only) the data it accesses, but the adversary can observe computation *continuously*, and can learn an unbounded amount of such information overall. In particular, this model can protect against an adversary that continuously perform side-channel attacks (such as DPA attacks), each of which leaks some partial information (only) about the "current" computation. On the other hand, the axiom that "only computation leaks information" does not seem to apply to many other natural attacks, such as the memory/microwave attacks or virtually all malware/virus attacks. A related model, where the adversary can learn the values on some subset of wires during the evaluation of a circuit, was studied by Ishai et al. [17].

Lastly, the recent works [9, 18] study *auxiliary input*, where the adversary can learn functions $f(\mathsf{sk})$ of the secret key sk subject only to the constraint that such a function is *hard to invert*. This is a strictly stronger model than the one considered in this work, as such functions f can have output length larger than the size of the secret key and can reveal *all* of the statistical entropy of the key.

2 Preliminaries

ENTROPY AND PREDICTABILITY. We review the information-theoretic definition for entropy, along a new generalization useful for our paper.

Definition 1. *The* min-entropy *of a r. v.* X *is* $\mathbf{H}_\infty(X) \overset{def}{=} -\log(\max_x \Pr[X = x])$.

We can rephrase the above definition in terms of predictors $\mathcal{A}$. The min-entropy of a random variable X measures how well X can be predicted by the *best* predictor $\mathcal{A}$, i.e. $\mathbf{H}_\infty(X) = -\log(\max_{\mathcal{A}} \Pr[\mathcal{A}() = X])$, where the max is taken over all predictors without any requirement on efficiency. The work of [10], offered a natural generalization of min-entropy, called the *(average) conditional min-entropy of* X *conditioned on* Z, which can be defined as measuring the maximum predictability of X by a predictor that is given the value Z. In this paper, we generalize the notion of conditional min-entropy still further, to *interactive* predictors $\mathcal{A}$, which participate in some *randomized experiment* $\mathcal{E}$. We model experiments as interactions between $\mathcal{A}$ and a *challenger oracle* $\mathcal{E}(\cdot)$ which can be randomized, stateful and interactive. We consider the predictability of X by an arbitrary predictor $\mathcal{A}^{\mathcal{E}(\cdot)}$.

Definition 2. *The* conditional min-entropy *of a random variable X, conditioned on the experiment $\mathcal{E}$ is $\widetilde{\mathbf{H}}_\infty(X \mid \mathcal{E}) \overset{def}{=} -\log(\max_{\mathcal{A}} \Pr[\mathcal{A}^{\mathcal{E}(\cdot)}() = X])$. In the special case that $\mathcal{E}$ is a non-interactive experiment which simply outputs a random variable Z, we abuse notation and write $\widetilde{\mathbf{H}}_\infty(X \mid Z)$ to denote $\widetilde{\mathbf{H}}_\infty(X \mid \mathcal{E})$.*

REVIEW OF Σ-PROTOCOLS. Let $\mathcal{R}$ be a relation consisting of *instance, witness* pairs $(x, w) \in \mathcal{R}$ and let $L_R = \{x \mid \exists w, (x, w) \in \mathcal{R}\}$ be the *language* of $\mathcal{R}$. A Σ-protocol for $\mathcal{R}$ is a protocol between a PPT ITM prover $\mathcal{P}(x, w)$ and a PPT ITM verifier $\mathcal{V}(x)$, which proceeds in three rounds with conversations (a, c, z) initiated by the prover. We require that a Σ-protocol satisfies *perfect completeness*, *special soundness*, and *Honest Verifier Zero Knowledge*. In the full version of the paper, we prove the following lemma.

Lemma 1. *Let $(\mathcal{P}, \mathcal{V})$ be an* HVZK *protocol for the relation $\mathcal{R}$, and let (X, W) be random variables over $\mathcal{R}$. Let $\mathcal{E}_1$ be an arbitrary experiment in which $\mathcal{A}$ is given X at the start of the experiment, and let $\mathcal{E}_2$ be the same as $\mathcal{E}_1$, except that $\mathcal{A}$ is also given oracle access to $\mathcal{P}(X, W)$ throughout the experiment. Then $\widetilde{\mathbf{H}}_\infty(W|\mathcal{E}_2) = \widetilde{\mathbf{H}}_\infty(W|\mathcal{E}_1)$.*

PRIME-ORDERED GROUPS. We use the notation $\mathcal{G}(1^\lambda)$ to denote a group sampling algorithm which, on input 1^λ, outputs a tuple $\mathbb{G} = (p, G, g)$ where p is a prime, G is a (description of a) group of order p, and g is a generator of G. We will rely on the usual hardness assumptions: the discrete-logarithm (DL), computational Diffie-Hellman assumption (CDH) and decisional Diffie-Hellman (DDH) assumptions. We will also rely on the Gap Diffie-Hellman (GDH) assumption which state that for some groups, in which the DDH problem can be solved efficiently (for example using a bilinear map), the CDH problem is still hard.

3 Leakage Oracle

We model adversarial leakage attacks on a secret key sk, by giving the adversary access to a *leakage oracle*, which the adversary can (periodically) query to gain information about sk. This oracle is defined as follows.

Definition 3. *A* leakage oracle $\mathcal{O}^{\lambda,\ell}_{\mathsf{sk}}(\cdot)$ *is parameterized by a secret key* sk*, a* leakage parameter ℓ *and a security parameter λ. A query to the oracle consists of a (description of)* leakage function $h_i : \{0,1\}^* \to \{0,1\}^{\alpha_i}$*. The oracle $\mathcal{O}^{\lambda,\ell}_{\mathsf{sk}}(\cdot)$ checks if the sum of α_i, over all queries received so far, exceeds the leakage parameter ℓ and ignores the query if this is the case. Otherwise, the oracle computes the function $h_i(\mathsf{sk})$ for at most* poly(λ) *steps. If the computation completes, the oracle responds with the output and, otherwise, it responds with the dummy value 1^{α_i}.*

Since the cumulative output of leakage-oracle queries can be guessed with probability at least $2^{-\ell}$, the oracle can decrease the entropy of sk by at most ℓ bits in any experiment.

Lemma 2. *For any random variable* SK*, any experiment $\mathcal{E}_1$, let $\mathcal{E}_2$ be the experiment which is the same as $\mathcal{E}_1$, but also gives the predictor access to the leakage oracle $\mathcal{O}^{\lambda,\ell}_{\mathsf{SK}}(\cdot)$. Then $\widetilde{\mathbf{H}}_\infty(\mathsf{SK} \mid \mathcal{E}_2\ ) \geq \widetilde{\mathbf{H}}_\infty(\mathsf{SK} \mid \mathcal{E}_1) - \ell$.*

4 Identification Schemes

4.1 Definition

In an identification scheme, a prover attempts to prove its identity to a verifier. This proof should be convincing and non-transferable. More formally, an identification scheme consists of the four PPT algorithms $(\mathsf{ParamGen}, \mathsf{KeyGen}, \mathcal{P}, \mathcal{V})$:

$\mathsf{params} \leftarrow \mathsf{ParamGen}(1^\lambda)$**:** Outputs the public parameters of the scheme, which are common to all users. These parameters are available as inputs to $\mathsf{KeyGen}, \mathcal{P}, \mathcal{V}$, and we omit them from the descriptions.

$(\mathsf{pk}, \mathsf{help}, \mathsf{sk}) \leftarrow \mathsf{KeyGen}()$**:** Outputs the public key pk, a helper help and a secret key sk. The value help is analyzed as a public key with respect to security (i.e. it need not be kept secret and is given to the adversary) but is thought of as a secret key for *usability* (i.e. it is not used by honest verifiers).[1]

$\mathcal{P}(\mathsf{pk}, \mathsf{help}, \mathsf{sk}), \mathcal{V}(\mathsf{pk})$**:** These are the prover and verifier ITMs respectively. The verifier $\mathcal{V}$ outputs a judgement from one of $\{Accept, Reject\}$ at the conclusion of a protocol execution.

We require that an ID scheme is *complete*, so that in an interaction $\{\mathcal{P}(\mathsf{pk}, \mathsf{sk}) \rightleftharpoons \mathcal{V}(\mathsf{pk})\}$ between honest prover and honest verifier, the verifier *always accepts* the proof. We now formally define what it means for an ID scheme to be leakage resilient. As discussed, we will consider two separate security notions. The first notion, called *pre-impersonation leakage security*, is modeled by the attack game $\mathsf{IDPRE}_\ell^\lambda(\mathcal{A})$ and only allows the adversary to submit leakage queries *prior to* an impersonation attack, but not during one. The second notion, called *anytime leakage security*, is modeled by the attack game $\mathsf{IDANY}_\ell^\lambda(\mathcal{A})$ where the adversary can perform leakage attacks adaptively at any point in time, even during an impersonation attack. The two attack games are defined below and only differ in the impersonation stage.

$\underline{\mathsf{IDPRE}_\ell^\lambda(\mathcal{A}),\ \ \mathsf{IDANY}_\ell^\lambda(\mathcal{A})}$

1. Key Stage: Let $\mathsf{params} \leftarrow \mathsf{ParamGen}(1^\lambda)$, $(\mathsf{pk}, \mathsf{help}, \mathsf{sk}) \leftarrow \mathsf{KeyGen}()$ and give $(\mathsf{params}, \mathsf{pk}, \mathsf{help})$ to $\mathcal{A}$.

2. Test Stage: The adversary $\mathcal{A}^{\mathcal{O}_{\mathsf{sk}}^{\lambda,\ell}(\cdot), \mathcal{P}(\mathsf{pk},\mathsf{sk})}$ gets access to the leakage oracle $\mathcal{O}_{\mathsf{sk}}^{\lambda,\ell}(\cdot)$ and to an honest prover $\mathcal{P}(\mathsf{pk}, \mathsf{sk})$, modeled as an oracle that runs (arbitrarily many) proofs upon request.

3. Impersonation Stage: This stage is defined separately for the two games.

For $\mathsf{IDPRE}_\ell^\lambda(\mathcal{A})$**:** The adversary $\mathcal{A}$ *loses* access to the all oracles and runs a protocol $\{\mathcal{A} \rightleftharpoons \mathcal{V}(\mathsf{pk})\}$ with as an honest verifier.

For $\mathsf{IDANY}_\ell^\lambda(\mathcal{A})$**:** The adversary $\mathcal{A}^{\mathcal{O}_{\mathsf{sk}}^{\lambda,\ell}(\cdot)}$ *maintains* access to the leakage oracle $\mathcal{O}_{\mathsf{sk}}^{\lambda,\ell}(\cdot)$, but *not* the prover oracle $\mathcal{P}$, and runs a protocol $\{\mathcal{A}^{\mathcal{O}_{\mathsf{sk}}^{\lambda,\ell}(\cdot)} \rightleftharpoons \mathcal{V}(\mathsf{pk})\}$ with an honest verifier.

The *advantage* of an adversary $\mathcal{A}$ in the games $\mathsf{IDPRE}_\ell^\lambda(\mathcal{A})$, $\mathsf{IDANY}_\ell^\lambda(\mathcal{A})$ is the probability that the verifier $\mathcal{V}$ accepts in the impersonation stage.

[1] In some of our constructions, when sk is made intentionally huge, the size of help will become large as well, and thus it is important that this does *not* detract from the usability of the scheme by also increasing the size of the public key.

Definition 4. *Let* $(\mathsf{KeyGen}, \mathcal{P}, \mathcal{V})$ *be an identification scheme with perfect completeness, parameterized by security parameter* λ. *We say that the scheme is* secure with pre-impersonation leakage ℓ *if the advantage of any PPT adversary* $\mathcal{A}$ *in the attack game* $\mathsf{IDPRE}_{\ell}^{\lambda}(\mathcal{A})$ *is negligible in* λ. *We say that the scheme is* secure with anytime leakage ℓ *if the above also holds for the attack game* $\mathsf{IDANY}_{\ell}^{\lambda}(\mathcal{A})$.

4.2 Construction 1: Generalized Okamoto Scheme

```
$\mathsf{ParamGen}(1^\lambda)$ : Let $(p, G, g) \leftarrow \mathcal{G}(1^\lambda)$, $g_1, \ldots, g_m \leftarrow_R G$.
    Set $\mathsf{params} = (p, G, g_1, \ldots, g_m)$.
$\mathsf{KeyGen}()$: Let $\mathsf{sk} = (x_1, \ldots, x_m) \leftarrow_R (\mathbb{Z}_p)^m$, $\mathsf{pk} = \prod_{j=1}^{m} \{g_j\}^{x_j}$. Output $(\mathsf{pk}, \perp, \mathsf{sk})$.
$\mathcal{P}, \mathcal{V}$: The machines $\mathcal{P}, \mathcal{V}$ run the following protocol:
    (1) $\mathcal{P}$: Computes $(a, \overline{y}) \leftarrow \mathbf{A}()$ and sends $a$ to $\mathcal{V}$.
        $\mathbf{A}()$ : Let $\overline{y} = (y_1, \ldots, y_m) \leftarrow_R (\mathbb{Z}_p)^m$, $a = \prod_{j=1}^{m} {g_j}^{y_j}$. Output $(a, \overline{y})$.
    (2) $\mathcal{V}$: Choose $c \leftarrow_R \mathbb{Z}_p$ and send $c$ to $\mathcal{P}$.
    (3) $\mathcal{P}$: Compute $\overline{z} \leftarrow \mathbf{Z}_{\mathsf{sk}}(c, \overline{y})$ and send $\overline{z}$ to $\mathcal{V}$.
        $\mathbf{Z}_{\mathsf{sk}}(c, \overline{y})$: Compute $z_j := y_j + cx_j$ for $j = 1, \ldots, m$, output $\overline{z} := (z_1, \ldots, z_m)$.
    $\mathbf{Ver}_{\mathsf{pk}}(a, c, \overline{z})$: Output *Accept* iff $\prod_{j=1}^{m} g_j^{z_j} \overset{?}{=} a(\mathsf{pk})^c$.
```

Fig. 1. The $\mathrm{Okamoto}_m^\lambda$ identification scheme.

We now show that the Okamoto identification scheme from [25] is secure against key leakage attacks. The standard Okamoto scheme is defined with respect to two generators. Here, we describe a generalized version of the Okamoto scheme with m generators. Since we will re-use the basic components of the scheme as building-blocks for our more complicated schemes, we abstract away most of the computation of the scheme into the algorithms $(\mathbf{A}, \mathbf{Z}, \mathbf{Ver})$ which are used by $\mathcal{P}, \mathcal{V}$ to run the protocol as defined in Figure 1. To analyze the above scheme, we define the relation $\mathcal{R} = \{(\mathsf{pk}, \mathsf{sk}) | \mathsf{sk} = (x_1, \ldots, x_m), \mathsf{pk} = \prod_{j=1}^{m} g_j^{x_j}\}$. We will rely on only three properties of the relation $\mathcal{R}$ and the generalized Okamoto ID scheme, outlined in Lemma 3.

Lemma 3. *The following three properties hold for the* $\mathrm{Okamoto}_m^\lambda$ *ID scheme:*

(1) The protocol $(\mathcal{P}, \mathcal{V})$ *is a* Σ*-protocol for the relation* $\mathcal{R}$.
(2) Denoting key pairs as random variables, we get $\widetilde{\mathbf{H}}_\infty(\mathsf{SK}|\mathsf{PK}) \geq (m-1)\log(p)$.
(3) Under the discrete logarithm assumption, it is difficult to find a public key pk *and two* different *secret keys* $\mathsf{sk}' \neq \mathsf{sk}$ *for* pk. *In particular, for any PPT adversary* $\mathcal{A}$:

$$\Pr\left[\mathsf{sk}' \neq \mathsf{sk} \text{ and } (\mathsf{pk}, \mathsf{sk}'), (\mathsf{pk}, \mathsf{sk}) \in \mathcal{R} \;\middle|\; \begin{array}{c} (\mathsf{pk}, \mathsf{sk}, \mathsf{sk}') \leftarrow \mathcal{A}(\mathsf{params}) \\ \mathsf{params} \leftarrow \mathsf{ParamGen}(1^\lambda) \leq \mathrm{negl}(\lambda) \end{array}\right].$$

Using the properties in the above lemma, we show that the Okamoto ID scheme is secure against key leakage attacks.

Theorem 1. *Under the* DL *assumption,* $\mathrm{Okamoto}_m^\lambda$ *is a secure ID-scheme for pre-impersonation leakage of up to* $\ell = (m-1)\log(p) - \lambda \geq (1 - \frac{2}{m})|\mathsf{sk}|$ *bits. It is secure with anytime leakage of up to* $\ell' = \frac{1}{2}\ell$ *bits.*

Proof Sketch. Assume that there is an adversary $\mathcal{A}$ that has a non-negligible advantage in the pre-impersonation leakage attack game $\mathsf{IDPRE}_{\ell}^{\lambda}(\mathcal{A})$. Then there is a reduction which, for randomly chosen params, finds two distinct secret keys $\mathsf{sk}, \mathsf{sk}'$ for a single public-key pk (contradicting part (3) of Lemma 3). In particular, the reduction chooses a random $(\mathsf{pk}, \mathsf{sk})$ tuple and uses sk to simulate the leakage oracle, and the honest-prover oracle for the attacker $\mathcal{A}$ during the "test stage". Then, during the impersonation stage, the reduction runs $\mathcal{A}$ twice, with two randomly chosen challenges c, c' (using rewinding). There is a non-negligible probability that $\mathcal{A}$ produces two accepting conversations $(a, c, z), (a, c', z')$ with $c \neq c'$. Using the special soundness property of the Σ-protocol, the reduction uses these two conversation to recover a secret key sk'.

We must now analyze the probability of $\mathsf{sk} = \mathsf{sk}'$. We think of the reduction as an experiment $\mathcal{E}_0$ where $\mathcal{A}$ gets PK and access to the oracles $\mathcal{O}_{\mathsf{SK}}^{\lambda,\ell}(\cdot), \mathcal{P}(\mathsf{PK}, \mathsf{SK})$. Let $\mathcal{E}_1$ be the same experiment as $\mathcal{E}_0$, except that the predictor does not get access to $\mathcal{O}_{\mathsf{SK}}^{\lambda,\ell}(\cdot)$, and $\mathcal{E}_2$ be the same as $\mathcal{E}_1$ except that the predictor doesn't get access to $\mathcal{P}(\mathsf{PK}, \mathsf{SK})$ either (i.e. only gets PK). Then

$$\begin{aligned}\widetilde{\mathbf{H}}_\infty(\mathsf{SK} \mid \mathcal{E}_0) &\geq \widetilde{\mathbf{H}}_\infty(\mathsf{SK} \mid \mathcal{E}_1) - \ell \\ &\geq \widetilde{\mathbf{H}}_\infty(\mathsf{SK} \mid \mathcal{E}_2) - \ell = \widetilde{\mathbf{H}}_\infty(\mathsf{SK} \mid \mathsf{PK}) - \ell \\ &\geq (m-1)\log(p) - \ell \geq \lambda\end{aligned}$$

where the first inequality follows by Lemma 2, the second one by Lemma 1, and the last one by part (3) of Lemma 3. The probability of the reduction outputting $\mathsf{sk}' = \mathsf{sk}$ is therefore upper bounded by $2^{-\lambda}$ and hence, with non-negligible probability, the reduction produces two distinct secret keys $\mathsf{sk} \neq \mathsf{sk}'$.

For anytime leakage, $\mathcal{A}$ can make calls to the leakage oracle for ℓ' bits *even* during the impersonation stage. Since the reduction runs the impersonation stage twice on different challenges (by rewinding $\mathcal{A}$), the reduction needs $2\ell'$ bits of leakage. Therefore, we can only handle $\ell' = \frac{1}{2}\ell$ bits of anytime leakage. □

4.3 Construction 2: Adding Flexibility Through Direct-Products

We now propose a construction of a leakage-resilient ID scheme with pre-impersonation security, that is suitable for the BRM setting. In particular, it is possible to increase the allowed leakage ℓ arbitrarily without significantly affecting the communication and computation complexity, or even the size of the public key and public parameters. As we will see, some of the parameters in our construction are still sub-optimal, and we will get further efficiency gains in Section 4.4. However, the construction we present here is more natural and simpler to understand, and hence we present it first.

The main idea of our construction, is to run many copies of the $\mathrm{Okamoto}_m^\lambda$ scheme in parallel. In particular, the secret key will be a database $\mathsf{sk} = (\mathsf{sk}[1], \ldots, \mathsf{sk}[n])$ where each $\mathsf{sk}[i]$ is a secret key for the underlying generalized Okamoto scheme, and defines a corresponding public key $\mathsf{pk}[i]$. During key generation, the prover also chooses a key pair $(\mathsf{verk}, \mathsf{sigk})$ for a signature scheme and computes signatures $\sigma[i]$ for each public key $\mathsf{pk}[i]$ and sets the helper string to $\mathsf{help} = (\sigma[1], \ldots, \sigma[n])$ (after which point sigk is never used again and deleted from memory). We could then define a four-round protocol, where the verifier begins by giving t random indices $(r_1, \ldots, r_t) \in [n]^t$ to the

```
$\mathsf{ParamGen}(1^\lambda)$: Let $(p, G, g) \leftarrow \mathcal{G}(1^\lambda)$, $g_1, \ldots, g_m \leftarrow_R G$.
    Set $\mathsf{params} = (p, G, g_1, \ldots, g_m)$.
$\mathsf{KeyGen}()$: Choose $(\mathsf{verk}, \mathsf{sigk}) \leftarrow \mathsf{SigKeyGen}(1^\lambda)$ and set $\mathsf{pk} = \mathsf{verk}$.
    For $i = 1, \ldots, n$ set: $(\mathsf{sk}[i], \mathsf{pk}[i]) \leftarrow \mathbf{Gen}()$, $\sigma[i] = \mathsf{Sign}_{\mathsf{sigk}}(i||\mathsf{pk}[i])$.
    Set $\mathsf{sk} = (\mathsf{sk}[1], \ldots, \mathsf{sk}[n])$, $\mathsf{help} = (\sigma[1], \ldots, \sigma[n])$.
    Output $(\mathsf{pk}, \mathsf{sk}, \mathsf{help})$.$^\dagger$
$\mathcal{P}, \mathcal{V}$: The machines $\mathcal{P}, \mathcal{V}$ run the following protocol:
    (1) $\mathcal{P}$: For $i = 1, \ldots, t$: choose $(a_i, \overline{y}_i) \leftarrow \mathbf{A}()$. Send $(a_1, \ldots, a_t)$ to $\mathcal{V}$.
    (2) $\mathcal{V}$: Choose $t$ indices $(r_1, \ldots, r_t) \leftarrow_R [n]^t$, and $c^* \leftarrow_R \mathbb{Z}_p$.
        Send the challenge $c = (r_1, \ldots, r_t, c^*)$ to $\mathcal{P}$.
    (3) $\mathcal{P}$: For $i = 1, \ldots, t$: set $\mathsf{pk}_i = \mathsf{pk}[r_i]$, $\sigma_i = \sigma[r_i]$, $\overline{z}_i = \mathbf{Z}_{\mathsf{sk}[r_i]}(c^*, \overline{y}_i)$ and send
        $(\mathsf{pk}_i, \sigma_i, \overline{z}_i)$ to $\mathcal{V}$.
    $\mathcal{V}$ accepts iff, for $i = 1, \ldots, t$:
    (I) The conversation $(a_i, c^*, \overline{z}_i)$ is accepting for $\mathsf{pk}_i$. That is, $\mathbf{Ver}_{\mathsf{pk}_i}(a_i, c^*, \overline{z}_i) \overset{?}{=}$ $\mathit{Accept}$.
    (II) The signatures $\sigma_i$ for $r_i||\mathsf{pk}_i$ verify under $\mathsf{pk}$. That is $\mathsf{SigVer}_{\mathsf{pk}}(r_i||\mathsf{pk}_i, \sigma_i) \overset{?}{=}$ $\mathit{Accept}$.
```

† Note that the values $\mathsf{pk}[i]$ can be easily computed from $\mathsf{sk}[i]$ and thus need not be stored separately.

Fig. 2. The $\mathrm{DirProd}^{\lambda}_{n,m,t}$ identification scheme.

prover. Then the prover and verifier then execute t independent copies of $\mathrm{Okamoto}^{\lambda}_{m}$ (in parallel) for the public keys $\mathsf{pk}[r_1], \ldots, \mathsf{pk}[r_t]$, which the prover sends to the verifier along with their signatures $\sigma[r_i]$. Our actual construction is a three-round scheme where the indices $r_1, \ldots, r_t$ are sent by the verifier with the challenge and we rely on the fact that the first messages a of the generalized Okamoto scheme does not depend on the public key pk.

To analyze the security of the scheme, we notice that, in a pre-impersonation attack, the adversary's queries to the leakage oracle must be independent of the indices $r_1, \ldots, r_t$. In the full version, we show that, if sk has a significant amount of entropy at the beginning of the impersonation stage, then the random tuple $(\mathsf{sk}[r_1], \ldots, \mathsf{sk}[r_t])$ will *preserve* some significant amount of this entropy as well. This analysis is based on thinking of the tuples $(\mathsf{sk}[r_1], \ldots, \mathsf{sk}[r_t])$ as positions in an (exponentially) long *direct-product encoding* of sk. Such codes were defined and analyzed in [16], where it is shown that they are "approximately-list decodable". We show that this property implies entropy preservation in our sense. Our security analysis then relies on the fact that, if an adversarial prover can complete a proof on the challenge $r_1, \ldots, r_t$, then it must *know* the values $(\mathsf{sk}[r_1], \ldots, \mathsf{sk}[r_t])$ in their entirety, which is unlikely by our entropy argument.

We note that, although our discussion seems quite general, it is not clear that the main idea of our construction (taking direct products) would imply a general a compiler which converts an ID scheme with pre-impersonation leakage ℓ into one with "amplified" pre-impersonation leakage $\ell' \gg \ell$. Indeed, our argument is (crucially) information theoretic in the sense that we show that a random subset of secret keys still has (information theoretic) entropy after the adversary gets some key leakage. To translate this into a more general argument, we would need to somehow simulate an ℓ' bit leakage oracle for the entire key sk by accessing (many) ℓ-bit leakage oracles for the individual keys $\mathsf{sk}[i]$, which does not seem possible.

We present our construction, called $\mathrm{DirProd}^{\lambda}_{n,m,t}$ in Figure 2. The presentation is based on the algorithms $(\mathbf{Gen}, \mathbf{A}, \mathbf{Z}, \mathbf{Ver})$ where $\mathbf{Gen}$ is the key generation algorithm for the underlying Okamoto scheme, and $(\mathbf{A}, \mathbf{Z}, \mathbf{Ver})$ are the algorithms used by the prover and verifier as defined in Figure 1.

Theorem 2. *Assuming that* $(\mathsf{SigKeyGen}, \mathsf{Sign}, \mathsf{SigVer})$ *is an existentially secure signature scheme under chosen message attacks, and assuming the hardness of* DL*, the construction* $\mathrm{DirProd}^{\lambda}_{n,m,t}$ *is a secure ID-scheme for pre-impersonation leakage of up to* $\ell = (1-\delta)nm\log(p) = (1-\delta)|\mathsf{sk}|$ *bits where* $\delta = \frac{1}{m}(1 + \frac{\log(n)}{\lambda} + \frac{4}{n}) + \frac{2\lambda}{t}$ *which approaches* $\frac{1}{m} + O(\lambda/t)$*.*

4.4 Construction 3: Saving Communication using Compressed Direct-Products

$\mathsf{ParamGen}(1^{\lambda})$**:** Choose $(p, G, g) \leftarrow \mathcal{G}(1^{\lambda})$, $(g_1, \ldots, g_m, u) \leftarrow_R G^{m+1}$.
Set $\mathsf{params} = (p, G, g_1, \ldots, g_m, u)$.
$\mathsf{KeyGen}()$**:** Choose $s \leftarrow_R \mathbb{Z}_p$ and set $\mathsf{pk} = v = u^s$.
Choose $(\mathsf{pk}[i], \mathsf{sk}[i]) \leftarrow_R \mathbf{Gen}()$ and set $\sigma[i] = (H(i)\mathsf{pk}[i])^s$ for $i \in \{1, \ldots, n\}$.[†]
Set $\mathsf{sk} = (\mathsf{sk}[1], \ldots, \mathsf{sk}[n])$, $\mathsf{help} = (\sigma[1], \ldots, \sigma[n])$. Output $(\mathsf{pk}, \mathsf{sk}, \mathsf{help})$.
$\mathcal{P}, \mathcal{V}$**:** The machines $\mathcal{P}, \mathcal{V}$ run the following protocol:
(1) $\mathcal{P}$**:** Choose $(a, \overline{y}) \leftarrow \mathbf{A}()$ and send a to $\mathcal{V}$.
(2) $\mathcal{V}$**:** Choose t indices $(r_1, \ldots, r_t) \leftarrow_R [n]^t$, and $(c^*, e) \leftarrow_R (\mathbb{Z}_p)^2$.
Send the challenge $c = (r_1, \ldots, r_t, e, c^*)$ to $\mathcal{P}$. [‡]
(3) $\mathcal{P}$**:** Compute $\mathsf{sk}^* = (x^*_1, \ldots, x^*_m)$ where $\left\{x^*_j = \sum_{i=1}^{t}(x_j[r_i])e^{i-1}\right\}_{j\in\{1,\ldots,t\}}$.
Set $\mathsf{pk}^* = \prod_{i=1}^{t} \mathsf{pk}[r_i]^{\left(e^{i-1}\right)}$, $\sigma^* = \prod_{i=1}^{t} \sigma[r_i]^{\left(e^{i-1}\right)}$, $\overline{z} = \mathbf{Z}_{\mathsf{sk}^*}(c^*, \overline{y})$.
Send $(\mathsf{pk}^*, \sigma^*, \overline{z})$ to $\mathcal{V}$.
$\mathcal{V}$ accepts iff:
(I) The conversation $(a, c^*, \overline{z})$ is accepting for pk^*. That is, $\mathbf{Ver}_{\mathsf{pk}^*}(a, c^*, \overline{z}) = \mathit{Accept}$.
(II) The value $(u, v, (\mathsf{pk}^* \prod_{i=1}^{t} H(r_i)^{e^{i-1}}), \sigma^*)$ is a DDH tuple.

† Recall that we write $\mathsf{sk}[i] = (x_1[i], \ldots, x_m[i]) \in \mathbb{Z}_p^m$.

‡As stated, the challenge size is $t\log(n)$ which dominates the remaining communication. In the Random Oracle model, we can compress the challenge to a λ bit value, which is then expanded into the full challenge using the Random Oracle. This version matches the parameters claimed in Table 1.

Fig. 3. The $\mathrm{CompDirProd}^{\lambda}_{n,m,t}$ identification scheme.

As we saw, Construction 2 gives us flexibility, in the sense that we can increase the parameter n to allow for arbitrarily large leakage ℓ, without (significantly) affecting the size of the public key, the computation or the communication complexity. Unfortunately, even though these factors do not depend on n, the communication of the scheme is fairly large since it uses $t = O(\lambda)$ copies of the underlying Okamoto scheme. In fact, just having the prover send t public keys $\mathsf{pk}[r_1], \ldots, \mathsf{pk}[r_t]$ to the verifier in construction 2 already takes the communication complexity to $O(\lambda^2)$, which may be prohibitive. As we will see later, large communication complexity of the ID schemes will translate into long Fiat-Shamir signatures and, therefore, large communication complexity in our final authenticated key agreement protocols.

In this section, we show how to reduce the communication complexity of the ID scheme significantly. As in Construction 2, the secret key $\mathsf{sk} = (\mathsf{sk}[1], \ldots, \mathsf{sk}[n])$ is a (possibly huge) database of keys $\mathsf{sk}[i]$ for the underlying generalized Okamoto scheme, and the verifier selects a random set of t indices which define a set of t secret keys $\mathsf{sk}[r_1], \ldots, \mathsf{sk}[r_t]$ used by the protocol execution. However, instead of running parallel versions of the $\mathrm{Okamoto}_m^\lambda$ scheme for these keys individually, the prover now *compresses* them into a *single secret key* sk^* and then runs a *single copy* of the Okamoto scheme for the corresponding public key pk^*, which the prover sends to the verifier. The two important properties of this compression are: (1) it must be entropy preserving, in the sense that sk^* should be (information theoretically) unpredictable, assuming that there is sufficient entropy spread-out over the entire database sk and (2) the public key pk^* for the secret key sk^* can be computed from $\mathsf{pk}[r_1], \ldots, \mathsf{pk}[r_t]$ alone, so that the values pk^* do not decrease the entropy of the database sk.

Our compression function is based on the *Reed-Solomon Error-Correcting Code*. In particular, the verifier chooses a random value $e \in \mathbb{Z}_p$, and the prover compresses the t secret keys $\{\mathsf{sk}[r_i] = (x_1[r_i], \ldots, x_m[r_i])\}_{i \in \{1,\ldots,t\}}$ into a single key $\mathsf{sk}^* = (x_1^*, \ldots, x_m^*)$, where $x_j^* = \sum_{i=1}^{t} (x_j[r_i]) e^{(i-1)}$ is the e-th position in the Reed-Solomon encoding of the value $(x_j[r_1], \ldots, x_j[r_t])$. In the full version of this paper, we show that this compression function is entropy preserving. The corresponding public key pk^* for sk^* is just $\mathsf{pk}^* = \prod_{i=1}^{t} (\mathsf{pk}[r_i])^{\left(e^{i-1}\right)}$, which is easy to compute from the individual keys $\mathsf{pk}[r_i]$. Thus it satisfies the two properties we required.

Of course, there is one crucial problem we have not yet addressed: how does an honest verifier check that the compressed public key pk^* given by the prover is indeed the right one (i.e. corresponds to the correct combination of $\mathsf{pk}[r_1], \ldots, \mathsf{pk}[r_t]$ using e as requested)? We can no longer use signatures, as in construction 2, since the number of possibilities for pk^* is exponential. Instead, we use a modification of the BLS signature scheme ([4]) to compute "helper values" $\sigma[i]$, which can be efficiently combined into a short "authenticator" σ^*. The authenticator σ^* essentially ensures that the adversary sends the correct public key pk^*. We present our construction, in Figure 3. The presentation is based on the algorithms $(\mathbf{Gen}, \mathbf{A}, \mathbf{Z}, \mathbf{Ver})$ for the underlying generalized Okamoto scheme (see Figure 1). In addition, our construction relies on a hash function H modeled as a random oracle. The security of the scheme is formalized in Theorem 3. The proof appears in the full version of this paper, and requires a careful analysis, combining the authentication properties of the modified BLS signatures with the rewinding strategy for the Okamoto scheme.

Theorem 3. *Under the* GDH *assumption, the* $\mathrm{CompDirProd}_{n,m,t}^{\lambda}$ *scheme is a secure ID-scheme in the Random Oracle model, with pre-impersonation leakage of up to* $\ell = (1-\delta)nm\log(p) = (1-\delta)|\mathsf{sk}|$ *bits where* $\delta = \frac{1}{m}(1 + \frac{\log(n)}{\lambda} + \frac{10}{n}) + \frac{6\lambda}{t}$ *which approaches* $\frac{1}{m} + O(\lambda/t)$.

5 Existentially and Entropically Unforgeable Signatures

We now look at leakage-resilient signatures, where the adversary can (periodically) query a leakage oracle for up to ℓ bits of information about the secret key. Unfortunately, if ℓ is larger than the size of a single signature, it is clear that we cannot achieve the standard notion of *existential unforgeability* as the attacker can simply choose to learn the a signature of some message m as its leakage function. Therefore, to construct meaningful signature schemes in the BRM, we also define a new (weaker) security notion called *entropic unforgeability*, where an attacker should be unable to forge messages which are chosen from some distribution of significant entropy and given to the adversary only *after* the leakage attack. To further strengthen the attack game we let the forger select this distribution. This notion is useful since, in many practical scenarios, an attacker must be able to forge signatures for messages that are somehow beyond her control, in order to damage the security of the system.

A signature scheme consists of four algorithms: $(\mathsf{ParamGen}, \mathsf{KeyGen}, \mathsf{Sign}, \mathsf{Verify})$. To capture entropic unforgeability, we separate the attacker into two parts $\mathcal{A} = (\mathcal{A}_1, \mathcal{A}_2)$, where $\mathcal{A}_1$ runs during the first stage of the attack, with access to a leakage oracle and signing oracle. Once this stage is done, $\mathcal{A}_1$ can output an arbitrary hint for $\mathcal{A}_2$, who then attempts to forge the signature of some message while having access *only* to the signing oracle. The formal definition of the *unforgeability attack game* $\mathsf{EUG}_\ell^\lambda$ appears in Figure 4. We use $\mathcal{S}_{\mathsf{sigk}}(\cdot)$ to denote the *signing oracle*, which, on input $m \in \{0,1\}^*$, outputs $\sigma = \mathsf{Sign}_{\mathsf{sigk},\mathsf{help}}(m)$. We define the advantage of forger $\mathcal{A} = (\mathcal{A}_1, \mathcal{A}_2)$ to be the probability that $\mathsf{Verify}_{\mathsf{verk}}(\mathsf{m}, \sigma) = \mathit{Accept}$ and that the signing oracle was never queried with m. For entropic security, we also require that the output message m is chosen sufficiently randomly by $\mathcal{A}_2$, so that it could not have been predicted by $\mathcal{A}_1$.

$\underline{\mathsf{EUG}_\ell^\lambda}$

Initialization: The challenger selects $(\mathsf{verk}, \mathsf{help}, \mathsf{sigk}) \leftarrow \mathsf{KeyGen}(1^\lambda)$ and gives verk to the forger $\mathcal{A}_1$.

Signing & Leakage Queries: Adversary $\mathcal{A}_1^{\mathcal{O}_{\mathsf{sk}}^{\lambda,\ell}(\cdot), \mathcal{S}_{\mathsf{sigk}}(\cdot)}$ is given access to the signing oracle $\mathcal{S}_{\mathsf{sigk}}(\cdot)$ and leakage oracle $\mathcal{O}_{\mathsf{sk}}^{\lambda,\ell}(\cdot)$ and outputs an arbitrary hint $v \in \{0,1\}^*$.

Post-Leakage: Adversary $\mathcal{A}_2^{\mathcal{S}_{\mathsf{sigk}}(\cdot)}$ is given the hint v and access to (only) the signing oracle $\mathcal{S}_{\mathsf{sigk}}(\cdot)$. We parse the output of $\mathcal{A}_2$ as a message, signature pair (m, σ).

Fig. 4. Entropic/Existential Unforgeability Attack Game

Definition 5. *For an adversary $\mathcal{A} = (\mathcal{A}_1, \mathcal{A}_2)$, let $\mathsf{View}_{\mathcal{A}_1}$ be a random variable describing the view of $\mathcal{A}_1$ including its random coins and signing-oracle/leakage-oracle responses.*[2] *Let $\mathsf{MSG}_{\mathcal{A}_2}$ be the random variable describing the message output by $\mathcal{A}_2$ in $\mathsf{EUG}_\ell^\lambda$. We say that an adversary $\mathcal{A} = (\mathcal{A}_1, \mathcal{A}_2)$ is* entropic *if $\widetilde{\mathbf{H}}_\infty(\mathsf{MSG}_{\mathcal{A}_2} | \mathsf{View}_{\mathcal{A}_1}) \geq \lambda$ for security parameter λ. We say that a signature scheme $(\mathsf{KeyGen}, \mathsf{Verify}, \mathsf{Sign})$ is* existentially unforgeable *with leakage ℓ if the advantage of any PPT adversary $\mathcal{A} = (\mathcal{A}_1, \mathcal{A}_2)$ in the game $\mathsf{EUG}_\ell^\lambda(\mathcal{A})$ is negligible in λ. We say that the signature scheme is* entropically unforgeable *with leakage ℓ if the above only holds for* entropic *adversaries.*

We use the Fiat-Shamir heuristic [14] to construct entropically (resp. existentially) unforgeable signature schemes secure against ℓ bits of key leakage, from ID schemes

[2] In the Random Oracle Model, this also includes responses to Random Oracle queries.

secure against pre-impersonation (resp. anytime) leakage of ℓ bits. Recall that, for a three round ID scheme with flows (a, c, z), the Fiat-Shamir signature scheme defines a signature of a message m to be (a, z) such that the conversation $(a, H(a||m), z)$ is accepting. Here $H(\cdot)$ is hash function modeled as a Random Oracle.

Theorem 4. *Let* ID *by a public coins ID scheme consisting of three rounds of interaction and let* Sig *be the signature scheme produced by the Fiat-Shamir heuristic applied to* ID. *(1) If* ID *allows pre-impersonation leakage* ℓ*, then* Sig *is entropically unforgeable with leakage* ℓ*. (2) If* ID *allows anytime leakage* ℓ*, then* Sig *is existentially unforgeable with leakage* ℓ*.*

CONCRETE SCHEMES: Combining this theorem with the $\mathrm{Okamoto}_m^\lambda$ identification scheme, analyzed in Theorem 1, we obtain a (I) leakage-resilient existentially unforgeable signature scheme where ℓ approaches up to half the size of the secret key (and signature) and a (II) leakage-resilient entropically unforgeable signature scheme where ℓ approaches the size of the entire secret key (and signature). For the BRM setting, we can instead use $\mathrm{CompDirProd}_{n,m,t}^\lambda$, analyzed in Theorem 3, and get a (III) entropically unforgeable signatures, where ℓ approaches the size of the entire secret key, and can be made arbitrarily large without negatively impacting the other parameters (and can be much larger than the size of a signature).

6 Interactive Encryption, Authentication and AKA

Using leakage-resilient entropically unforgeable signatures, we can construct several *interactive* leakage-resilient primitives including encryption, authentication and authenticated key agreement. The security of these interactive primitives is preserved even if the adversary gets up to ℓ bits of leakage *prior to* the start of protocol execution, and the key is leaked entirely *after* the end of protocol execution.

For example consider the following simple two-round interactive authentication protocol. The verifier sends a random challenge r to the signer, who returns the signature $\sigma = \mathsf{Sign}_{\mathsf{sigk},\mathsf{help}}(m||r)$ for a message m. If no leakage occurs between the time where the verifier sends r and receives σ then, since r is random, entropic-unforgeability ensures that the adversary cannot forge the signature σ' of $m'||r$ for some $m' \neq m$.

Alternatively, consider the following simple three-round interactive encryption. The sender sends a random challenge r to the receiver, who in turn chooses a fresh temporary public/secret key pair $(\mathsf{pk}, \mathsf{sk})$ for a standard (non-leakage resilient) encryption scheme and sends $\mathsf{pk}, \sigma = \mathsf{Sign}_{\mathsf{sigk},\mathsf{help}}(\mathsf{pk}||r)$ to the sender. If the signature verifies, the sender sends an encryption of the message m under pk to the receiver, who decrypts it with sk and deletes all temporary state (including sk) immediately afterwards. This way, if no leakage occurs between the time that r is sent and sk is deleted, then entropic-unforgeability ensures that sk was chosen by the honest receiver, and so privacy is preserved since the adversary can never learn anything about sk.

Defining the security of interactive encryption and authentication schemes is a tedious process. Therefore, in the full version of the paper, we concentrate on the single primitive of Authenticated Key-Agreement (AKA), which allows for interactive encryption as well as authentication. We adapt the notion of *SK-security with perfect forward secrecy* from Canetti and Krawczyk in [6], and update it to model key-leakage attacks.

We then analyze a simple AKA construction from [6], which essentially consists of the Diffie-Hellman key-exchange protocol in which the parties sign the exchanged messages together with unique session information so as to bind the protocol execution to a particular session. We show that, if we employ leakage-resilient entropically secure signatures in this construction, then the resulting AKA is leakage-resilient as well.

7 Invisible Key Updates

Our schemes allow for efficient updates of the secret key, using an externally stored "master update key", so that the adversary is only limited to leaking ℓ bits between updates, but can get unlimited leakage overall. Since this is technically simple, we only give a high-level description of how this is done.

In particular, for our constructions $\mathrm{DirProd}^{\lambda}_{n,m,t}$ and $\mathrm{CompDirProd}^{\lambda}_{n,m,t}$, there is already a "master key" which is used to create a secret-key database of unbounded size – namely, the "master signing key" for generic signatures in $\mathrm{DirProd}^{\lambda}_{n,m,t}$ and for modified BLS signatures in $\mathrm{CompDirProd}^{\lambda}_{n,m,t}$. In our original descriptions, this master key is used once to create the secret-key database $\mathsf{sk} = (\mathsf{sk}[1], \ldots, \mathsf{sk}[n])$ and the helper $\mathsf{help} = (\mathsf{help}[1], \ldots, \mathsf{help}[n])$, and is then deleted immediately afterwards. However, we notice that the master key can really generate arbitrarily many secret keys $(\mathsf{sk}[1], \mathsf{sk}[2], \ldots)$ and corresponding helper strings $(\mathsf{help}[1], \mathsf{help}[2], \ldots)$.

To perform updates, we can store this "mater update key" on a separate external device, which is not susceptible to leakage, as a "mater update key". To update the secret-key database, we simply overwrite the current secret-keys and helper values with the "next" n values so that $\mathsf{sk} := (\mathsf{sk}[nk+1], \ldots, \mathsf{sk}[n(k+1)]), \mathsf{help} = (\mathsf{help}[nk+1], \ldots, \mathsf{help}[n(k+1)])$ after the kth update. To run an ID scheme, the prover simply sends the current index k to the verifier in the first flow of the protocol, and the verifier chooses the challenge indices in the range $[nk+1, n(k+1)]$. Note that an adversarial prover can send any index k' of his choosing. However, if the adversary learns at most ℓ bits in between any two updates, then there is *no* index k' for which the adversary can successfully run an impersonation attack.

The above updates for ID schemes translate to similar updates for our signature schemes and AKA protocols. Notice that these updates do not modify the public key, and the user has a completely free choice of when or how often the secret key is updated. However, it is important that the "master signing key" is stored securely and that the adversary cannot get any leakage of this key.

References

1. A. Akavia, S. Goldwasser, and V. Vaikuntanathan. Simultaneous hardcore bits and cryptography against memory attacks. In *TCC*, pages 474–495, 2009.
2. S. M. Bellovin and M. Merritt. Augmented encrypted key exchange: A password-based protocol secure against dictionary attacks and password file compromise. In *ACM Conference on Computer and Communications Security*, pages 244–250, 1993.
3. C. H. Bennett, G. Brassard, and J.-M. Robert. Privacy amplification by public discussion. *SIAM J. Comput.*, 17(2):210–229, 1988.
4. D. Boneh, B. Lynn, and H. Shacham. Short signatures from the weil pairing. In *ASIACRYPT*, pages 514–532, 2001.

5. R. Canetti, Y. Dodis, S. Halevi, E. Kushilevitz, and A. Sahai. Exposure-resilient functions and all-or-nothing transforms. In *EUROCRYPT*, pages 453–469, 2000.
6. R. Canetti and H. Krawczyk. Analysis of key-exchange protocols and their use for building secure channels. In *EUROCRYPT*, pages 453–474, 2001.
7. D. Cash, Y. Z. Ding, Y. Dodis, W. Lee, R. J. Lipton, and S. Walfish. Intrusion-resilient key exchange in the bounded retrieval model. In *TCC*, pages 479–498, 2007.
8. G. D. Crescenzo, R. J. Lipton, and S. Walfish. Perfectly secure password protocols in the bounded retrieval model. In *TCC*, pages 225–244, 2006.
9. Y. Dodis, Y. T. Kalai, and S. Lovett. On cryptography with auxiliary input. In *STOC*, 2009. To Appear.
10. Y. Dodis, R. Ostrovsky, L. Reyzin, and A. Smith. Fuzzy extractors: How to generate strong keys from biometrics and other noisy data. *SIAM J. Comput.*, 38(1):97–139, 2008.
11. Y. Dodis, A. Sahai, and A. Smith. On perfect and adaptive security in exposure-resilient cryptography. In *EUROCRYPT*, pages 301–324, 2001.
12. S. Dziembowski. Intrusion-resilience via the bounded-storage model. In *TCC*, pages 207–224, 2006.
13. S. Dziembowski and K. Pietrzak. Leakage-resilient cryptography. In *FOCS*, pages 293–302, 2008.
14. A. Fiat and A. Shamir. How to prove yourself: Practical solutions to identification and signature problems. In *CRYPTO*, pages 186–194, 1986.
15. J. A. Halderman, S. D. Schoen, N. Heninger, W. Clarkson, W. Paul, J. A. Calandrino, A. J. Feldman, J. Appelbaum, and E. W. Felten. Lest we remember: Cold boot attacks on encryption keys. In *USENIX Security Symposium*, pages 45–60, 2008.
16. R. Impagliazzo, R. Jaiswal, and V. Kabanets. Approximately list-decoding direct product codes and uniform hardness amplification. In *FOCS*, pages 187–196, 2006.
17. Y. Ishai, A. Sahai, and D. Wagner. Private circuits: Securing hardware against probing attacks. In *CRYPTO*, pages 463–481, 2003.
18. Y. T. Kalai and V. Vaikuntanathan. Public-key encryption schemes with auxiliary inputs and applications. 2009. Personal Communication.
19. J. Katz. Signature schemes with bounded leakage resilience. In *Cryptology ePrint Archive, Report 2009/220*, 2009. `http://eprint.iacr.org/2009/220`.
20. P. C. Kocher. Timing attacks on implementations of diffie-hellman, rsa, dss, and other systems. In *CRYPTO*, pages 104–113, 1996.
21. P. C. Kocher, J. Jaffe, and B. Jun. Differential power analysis. In *CRYPTO*, pages 388–397, 1999.
22. U. M. Maurer. Protocols for secret key agreement by public discussion based on common information. In *CRYPTO*, pages 461–470, 1992.
23. S. Micali and L. Reyzin. Physically observable cryptography (extended abstract). In *TCC*, pages 278–296, 2004.
24. M. Naor and G. Segev. Public-key cryptosystems resilient to key leakage. In *CRYPTO*, 2009. To Appear. Full version at `http://eprint.iacr.org/2009/105`.
25. T. Okamoto. Provably secure and practical identification schemes and corresponding signature schemes. In *CRYPTO*, pages 31–53, 1992.
26. H. Ong and C.-P. Schnorr. Fast signature generation with a fiat shamir-like scheme. In *EUROCRYPT*, pages 432–440, 1990.
27. K. Pietrzak. A leakage-resilient mode of operation. In *Eurocrypt 2009, Cologne, Germany*, 2009.
28. J.-J. Quisquater and D. Samyde. Electromagnetic analysis (ema): Measures and countermeasures for smart cards. In *E-smart*, pages 200–210, 2001.
29. O. Regev. On lattices, learning with errors, random linear codes, and cryptography. In *STOC*, pages 84–93, 2005.