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

Joël Alwen * Yevgeniy Dodis† Daniel Wichs‡

May 20, 2009

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 leakage-resilient AKA in the Random-Oracle model. This protocol ensures that session keys are private and authentic even if (1) the adversary leaks a large fraction of the long-term secret keys of both users prior to the protocol execution and (2) the adversary completely learns the long-term secret keys after the protocol execution. In particular, our AKA protocol provides qualitatively stronger privacy guarantees than leakage-resilient public-encryption schemes (constructed in prior and concurrent works), since such schemes necessarily become 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 $\ell(1 + \delta) + O(\lambda)$.
  In particular, the attacker can learn an approximately $(1 - \delta)$ fraction of the secret key.
- Public key size is $O(\lambda)$, and independent of ℓ.
- Communication complexity is $O(\lambda/\delta)$, and independent of ℓ.
- All computation reads at most $O(\lambda/\delta^2)$ locations of the secret key, independently 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.

*Computer Science Dept. NYU. Email: jalwen@cs.nyu.edu.
†Computer Science Dept. NYU. Email: dodis@cs.nyu.edu.
‡Computer Science Dept. NYU. Email: wichs@cs.nyu.edu.

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 [Koc96, BDL97, BS97, KJJ99, QS01, GMO01], 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. [HSH$^+$08], 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 [HSH$^+$08] (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 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) [CLW06, Dzi06], 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)* protocols where we can achieve *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 [AGV09]. 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 [Oka92] using m generators, denoted Okamoto_m^λ, 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 [Oka92] based on RSA, and the Ong-Schnorr [OS90] 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 $\text{DirProd}_{n,m,t}^\lambda$, 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 $\text{CompDirProd}_{n,m,t}^\lambda$, has essentially the same communication complexity as the basic (non-BRM) Okamoto scheme even though the allowed leakage ℓ can be made arbitrarily large.

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:

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

Entries (except ℓ) represent # of group elements. Security parameter is λ. Parameters n, m, t offer flexibility.

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

- For the first scheme ($\text{Okamoto}^{\lambda}_{m}$), 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 ($\text{DirProd}^{\lambda}_{n,m,t}, \text{CompDirProd}^{\lambda}_{n,m,t}$), 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 $\ell \approx (1-\delta)\lvert\mathsf{sk}\rvert$ 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 [BBR88, Mau92b, BBCM95] shows how two users, who *share* a weak secret, can agree on a uniformly random key in the presence of a passive attacker. The works of [MW97, RW03, DKRS06, KR09, DW09] extend this to active attacks, and the works of [Mau92a, AR99, ADR02, Lu02, Vad04] extended this to the case of *huge* secrets (motivated by the Bounded Storage Model, but also applicable to the BRM). Such information-theoretically secure schemes can only be used *once* to convert a shared secret, which may have been partially compromised by leakage attacks, into a *single* uniform session-key.

In the computational setting, users can agree on *arbitrarily many* session-keys using Password Authenticated Key Agreement (PAKE) [BM93, BPR00, BMP00, KOY01, GL06], 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 [CLW06, Dzi06]. In particular, Dziembowski [Dzi06] 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 [CDD+07]. Other symmetric-key applications, such as password authentication and secret sharing, were studied in the BRM setting by [CLW06] and [DP07], respectively. We also note that *non-interactive* symmetric key encryption schemes from weakly-secret keys were constructed implicitly in [Pie09] (based on weak pseudorandom functions) and explicitly in [DKL09] 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. [AGV09], which showed that Regev's public-key encryption scheme [Reg05] (based on lattices) is leakage-resilient. Recently, and concurrently with our work, the results of [NS09, KV09] present several new constructions of of public key encryption schemes for this setting, based on other (non-lattice) assumptions, tolerating more leakage, achieving CCA-2 security and allowing for stronger "auxiliary-input" attacks (described subsequently). The main two drawbacks of these works, which we address with our result, are that (1) non-interactive encryption schemes inherently become insecure if the adversary can perform leakage attacks *after* seeing a ciphertext and (2) the proposed encryption schemes are only secure with respect to relative leakage and it is unclear how to extend them to the BRM setting.

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* [CDH+00, DSS01] 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 [MR04] and studied further by Dziembowski and Pietrzak [DP08, Pie09], 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. [ISW03, IPSW06].

Lastly, the recent works [DKL09, KV09] 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

All proofs missing from the main body of the paper are relegated to Appendix B.

Entropy and Predictability. We review information-theoretic definitions for entropy, along with some of our generalizations and lemmas which we will rely on in the paper.

Definition 2.1. *The* min-entropy *of a random variable X is* $\mathbf{H}_\infty(X) \stackrel{def}{=} -\log(\max_x \Pr[X = x])$.

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 [DORS08], offered a natural generalization of min-entropy, called the *(average) conditional min-entropy of X conditioned on Z*, defined as $\widetilde{\mathbf{H}}_\infty(X|Z) \stackrel{\text{def}}{=} -\log(\mathbf{E}_z \max_x \Pr[X = x|Z = z])$. We can rephrase this definition in terms of predictors $\mathcal{A}$ that are given some information Z (presumably correlated with X), so $\widetilde{\mathbf{H}}_\infty(X|Z) = -\log(\max_{\mathcal{A}} \Pr[\mathcal{A}(Z) = X])$. 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 (think of this as a challenger in an attack game). We consider the predictability of X by a predictor $\mathcal{A}^{\mathcal{E}(\cdot)}$ which can act arbitrarily in the experiment with the challenger.

Definition 2.2. *The* conditional min-entropy *of a random variable X, conditioned on the experiment $\mathcal{E}$ is $\widetilde{\mathbf{H}}_\infty(X \mid \mathcal{E}) \stackrel{def}{=} -\log(\max_{\mathcal{A}} \Pr[\mathcal{A}^{\mathcal{E}(\cdot)}() = X])$. 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 $\widetilde{\mathbf{H}}_\infty(X \mid \mathcal{E})$.*

In Appendix A, we present several new results connecting entropy and the (approximate) list-decoding of error-correcting codes. These results help us prove the security of our constructions for the Bounded-Retrieval Model.

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 where: (1) the prover $\mathcal{P}(x, w)$ sends an initial message a, (2) the verifier $\mathcal{V}(x)$ sends a uniformly random challenge c, (3) the prover $\mathcal{P}(x, w)$ sends a response z. The verifier $\mathcal{V}(x)$ either *accepts* or *rejects* the conversation by computing some predicate of the instance x and the conversation (a, c, z). We require that Σ-protocols satisfy the following three properties:

1. *Perfect Completeness:* For any $(x, w) \in \mathcal{R}$, the execution $\{\mathcal{P}(x, w) \rightleftharpoons \mathcal{V}(x)\}$ is always accepting.
2. *Special Soundness:* There is an efficient algorithm such that, given an instance x and two accepting conversations for x: (a, c, z), (a, c', z') where $c \neq c'$, the algorithm outputs w such that $(x, w) \in \mathcal{R}$.
3. *Perfect Honest Verifier Zero Knowledge (*HVZK*):* There is a PPT simulator $\mathcal{S}$ such that, for any $(x, w) \in \mathcal{R}$, the simulator $\mathcal{S}(x)$ produces conversations (a, c, z) which are *identically distributed* to the conversations produced by an honest execution $\{\mathcal{P}(x, w) \rightleftharpoons \mathcal{V}(x)\}$.

As was shown in [CDS94], the HVZK property implies *witness indistinguishability.* Here, we rephrase essentially the same property in a slightly different manner. We show that, oracle access to a prover $\mathcal{P}(x, w)$ does not decrease the entropy of w *in any experiment* in which x is given to the predictor.

Lemma 2.1. *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 $\widetilde{\mathbf{H}}_\infty(W|\mathcal{E}_2) = \widetilde{\mathbf{H}}_\infty(W|\mathcal{E}_1)$.*

Hardness Assumptions. We review several of the standard hardness assumptions for prime-order groups. Let $\mathcal{G}(1^\lambda)$ be 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. Each of the following assumptions is made for some specific groups $\mathbb{G}$ and algorithms $\mathcal{G}$ which we do not specify. The *discrete-logarithm* (DL) assumption for groups $\mathbb{G}$ states that $\Pr[\mathcal{A}(\mathbb{G}, g^x) = x \mid \mathbb{G} \leftarrow_R \mathcal{G}(1^\lambda), x \leftarrow_R \mathbb{Z}_p] \leq \mathrm{negl}(\lambda)$. The *computational Diffie-Hellman* (CDH) assumption for groups $\mathbb{G}$ states that $\Pr[\mathcal{A}(\mathbb{G}, g^x, g^y) = g^{xy} \mid \mathbb{G} \leftarrow_R \mathcal{G}(1^\lambda), x, y \leftarrow_R \mathbb{Z}_p] \leq \mathrm{negl}(\lambda)$. The *decisional Diffie-Hellman* (DDH) assumption for groups $\mathbb{G}$ states that the tuples $(\mathbb{G}, g^x, g^y, g^{xy})$ and $(\mathbb{G}, g^x, g^y, g^z)$ are computationally indistinguishable, where x, y, z are uniformly random over $\mathbb{Z}_p$ and $\mathbb{G} \leftarrow \mathcal{G}(1^\lambda)$. Lastly, the *Gap Diffie-Hellman* (GDH) assumption for groups $\mathbb{G}$ states that the CDH assumption holds for these groups, but DDH is easy. In particular, we require that there is a poly-time algorithm which, on input $(\mathbb{G}, g, g^x, g^y, h)$ outputs 1 iff $h = g^{xy}$. For example, this is the case for bilinear groups, where the bilinear map allows us to solve DDH easily, but CDH is still believed to be 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. Intuitively, we would like to capture the fact that the adversary can compute arbitrary efficient functions of the secret key as long as the *total* number of bits learned is *bounded* by some parameter ℓ. These *leakage functions* can be chosen adaptively, based on the results of prior leakage attacks and any other events that may take place during the attack game.[1] The following definition formalizes the above concept.

Definition 3.1. *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) a* leakage function $h_i : \{0,1\}^* \to \{0,1\}^{\alpha_i}$ *(i.e. a Turing Machine whose output tape is at most* α_i *bits). 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. The oracle computes the function* $h_i(\mathsf{sk})$ *for at most* $\mathsf{poly}(\lambda)$ *steps and, if the computation completes, responds with the output. Otherwise, it responds with the dummy value* 1^{α_i}*.*

We now show that, in any experiment $\mathcal{E}$, adding access to the leakage oracle $\mathcal{O}^{\lambda,\ell}_{\mathsf{sk}}(\cdot)$ decreases the entropy of sk by at most ℓ bits.

Lemma 3.1. *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 $(\mathsf{ParamGen}, \mathsf{KeyGen}, \mathcal{P}, \mathcal{V})$:

[1] For example, in a *chosen message attack* on a signature scheme, the adversary may adaptively choose its leakage functions based on the signatures it gets from the signing oracle.

$\mathsf{params} \leftarrow \mathsf{ParamGen}(1^\lambda)$**:** 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).[2]

$\mathcal{P}(\mathsf{pk}, \mathsf{help}, \mathsf{sk}), \mathcal{V}(\mathsf{pk})$**:** 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 the security of ID schemes for adversaries that can perform leakage attacks against the secret key sk. 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.

<u>$\mathsf{IDPRE}_\ell^\lambda(\mathcal{A}), \quad \mathsf{IDANY}_\ell^\lambda(\mathcal{A})$</u>

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.

Definition 4.1. *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

We now show that the Okamoto identification scheme from [Oka92] 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

[2]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.

$\mathsf{ParamGen}(1^\lambda):$ 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 $\mathsf{sk} = (x_1, \ldots, x_m) \leftarrow_R (\mathbb{Z}_p)^m$ and set $\mathsf{pk} = \prod_{j=1}^m \{g_j\}^{x_j}$. 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 $\mathcal{V}$.
$\mathbf{A}()$: Choose $\overline{y} = (y_1, \ldots, y_m) \leftarrow_R (\mathbb{Z}_p)^m$, compute $a = \prod_{j=1}^m g_j{}^{y_j}$. 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 $\prod_{j=1}^m g_j^{z_j} \stackrel{?}{=} a(\mathsf{pk})^c$.

Figure 1: The Okamoto_m^λ identification scheme.

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, which are outlined in the following lemma.

Lemma 4.1. *The following three properties hold for the* Okamoto_m^λ *ID scheme:*

(1) It is difficult to find a public key pk *and two* different *secret keys* $\mathsf{sk}' \neq \mathsf{sk}$ *for* pk*. In particular, under the* DL *assumption, for any PPT adversary* $\mathcal{A}$*:*

$$\Pr[\mathsf{sk}' \neq \mathsf{sk} \text{ and } (\mathsf{pk}, \mathsf{sk}'), (\mathsf{pk}, \mathsf{sk}) \in \mathcal{R} \mid (\mathsf{pk}, \mathsf{sk}, \mathsf{sk}') \leftarrow \mathcal{A}(\mathsf{params}), \mathsf{params} \leftarrow \mathsf{ParamGen}(1^\lambda)] \leq \mathrm{negl}(\lambda).$$

(2) The protocol $\mathcal{P}, \mathcal{V}$ *is* Σ*-protocol for the relation* $\mathcal{R}$*.*

(3) Thinking of key pairs $(\mathsf{pk}, \mathsf{sk})$ *as random variables* $(\mathsf{PK}, \mathsf{SK})$*, we get* $\widetilde{\mathbf{H}}_\infty(\mathsf{SK}|\mathsf{PK}) \geq (m-1)\log(p)$*.*

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

Theorem 4.1. *Under the* DL *assumption,* Okamoto_m^λ *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. The (perfect) completeness property is easy to verify. For security, we start with pre-impersonation leakage. Assume that there is an adversary $\mathcal{A}$ which runs in time t (including the run-time of the queries for $\mathcal{O}_{\mathsf{sk}}^{\lambda,\ell}(\cdot)$) and has advantage ε in the game $\mathsf{IDPRE}_\ell^\lambda(\mathcal{A})$. Then, we construct an adversary $\mathcal{B}$ which runs in time $\approx 2t$ and

$$\Pr[\mathsf{sk}' \neq \mathsf{sk} \text{ and } (\mathsf{pk}, \mathsf{sk}'), (\mathsf{pk}, \mathsf{sk}) \in \mathcal{R} \mid (\mathsf{pk}, \mathsf{sk}, \mathsf{sk}') \leftarrow \mathcal{B}(\mathsf{params}), \mathsf{params} \leftarrow \mathsf{ParamGen}(1^\lambda)] \geq \varepsilon^2 - \frac{1}{p} - 2^{-\lambda}.$$

The adversary $\mathcal{B}$ chooses a random pair $(\mathsf{pk}, \mathsf{sk}) \leftarrow \mathsf{KeyGen}()$ and runs as the challenger for $\mathcal{A}$ in the game $\mathsf{IDPRE}_\ell^\lambda(\mathcal{A})$. In particular, $\mathcal{B}$ gives pk to $\mathcal{A}$ and uses sk to construct the leakage oracle $\mathcal{O}_{\mathsf{sk}}^{\lambda,\ell}(\cdot)$ and a prover $\mathcal{P}(\mathsf{pk}, \mathsf{sk})$ for $\mathcal{A}$ to interact with. When $\mathcal{A}$ reaches the impersonation stage, $\mathcal{B}$ chooses a random challenge $c \leftarrow_R \mathbb{Z}_p$ which results in the conversation (a, c, z). Then $\mathcal{B}$ rewinds $\mathcal{A}$, sends an fresh random challenge $c' \leftarrow_R \mathbb{Z}_p$, and gets a conversation (a, c', z'). If both conversations (a, c, z), (a, c', z') are accepting and $c \neq c'$ then, using the special soundness property, $\mathcal{B}$ (always) extracts a witness sk' such that $(\mathsf{pk}, \mathsf{sk}') \in \mathcal{R}$. Let E_1 be the event that the above occurs and let E_2 be the event that $\mathsf{sk}' = \mathsf{sk}$.

Claim 4.1. *The probability of the event E_1 is* $\Pr[E_1] \geq \varepsilon^2 - 1/p$.

Proof. By assumption, the overall advantage of $\mathcal{A}$ is ε. Let ε_v be the advantage of $\mathcal{A}$ conditioned on a particular *execution* of the attack game up until the challenge phase (i.e. the coins of $\mathcal{A}$, the responses from $\mathcal{O}^{\lambda,\ell}_{\mathsf{sk}}(\cdot)$ and $\mathcal{P}(\mathsf{pk}, \mathsf{sk})$). Let V be a random variable for such executions v. We have $\Pr[E_1 \mid V = v] \geq \varepsilon_v^2 - 1/p$, since the probability that two conversations are accepting is ε_v^2, and the probability that $c = c'$ is at most $1/p$. Therefore

$$\Pr[E_1] = \sum_v \Pr[E_1 | V = v] \Pr[V = v] \geq \mathbf{E}[\varepsilon_v^2 - 1/p] \geq \mathbf{E}[\varepsilon_v^2] - 1/p \geq (\mathbf{E}[\varepsilon_v])^2 - 1/p \geq \varepsilon^2 - 1/p$$

which completes the proof of the claim. □

Claim 4.2. *The probability of E_2 is* $\Pr[E_2] \leq 2^{-\lambda}$.

Proof. We can think of the process which produces the secret key as a predictor that runs in an experiment $\mathcal{E}_0$, in which it gets access to the oracles $\mathcal{O}^{\lambda,\ell}_{\mathsf{SK}}(\cdot)$, $\mathcal{P}(\mathsf{PK}, \mathsf{SK})$ and public key PK. In particular, our specific predictor runs the adversary $\mathcal{A}$ once, rewinds it, and gives it an alternate challenge. Therefore, we can bound $\Pr[E_2] \leq 2^{-\widetilde{\mathbf{H}}_\infty(\mathsf{SK} \mid \mathcal{E}_0)}$. Let $\mathcal{E}_1$ be the same experiment as $\mathcal{E}_0$, except that the predictor does not get access to $\mathcal{O}^{\lambda,\ell}_{\mathsf{SK}}(\cdot)$, 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

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

where the first inequality follows by Lemma 3.1, the second one by Lemma 2.1, and the last one by part (3) of Lemma 4.1. The claim follows since $\ell \leq (m-1)\log(p) - \lambda$. □

(continuing the proof of Theorem 4.1) Given the two claims, the first part of the theorem (pre-impersonation leakage) follows by noticing that the success probability of the attacker $\mathcal{B}$ is:

$$\Pr[E_1 \wedge \neg E_2] \geq \varepsilon^2 - 1/p - 2^{-\lambda} \geq \varepsilon^2 + \mathrm{negl}(\lambda).$$

The proof for anytime security is the same with one small modification. Now, adversary $\mathcal{A}$ can make (all of its) queries *after* receiving the challenge c and choose its leakage functions depending on c. Therefore, when it is rewound and given a fresh challenge c' it may attempt to make fresh *new and different* queries to $\mathcal{O}^{\lambda,\ell}_{\mathsf{sk}}(\cdot)$. Hence, for anytime leakage, the predictor in experiment $\mathcal{E}_0$ (in Claim 4.2) really gets access to a leakage oracle with leakege parameter $2\ell'$ if the adversary $\mathcal{A}$ has access to one with leakege ℓ'. Therefore, for anytime attacks, the amount of allowed leakege is halved. □

4.3 Construction 2: Adding Flexibility Through Direct-Products

We now propose a construction of an 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 Okamoto^λ_m scheme in parallel. In particular, the secret key will be a (possibly huge) 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 $\mathsf{pk}[i]$ (after which point sigk is never used again and deleted from memory) and sets $\mathsf{help} = (\sigma[1], \ldots, \sigma[n])$. We could then define a 4 rounds protocol, where the verifier begins by giving t random indices $(r_1, \ldots, r_t) \in [n]^t$ to the prover. Then the prover and verifier

$\mathsf{ParamGen}(1^\lambda)$**:** 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 $\mathsf{sk} = (\mathsf{sk}[1], \ldots, \mathsf{sk}[n])$, $\mathsf{help} = (\mathsf{pk}[1], \ldots, \mathsf{pk}[n], \sigma[1], \ldots, \sigma[n])$ and output $(\mathsf{pk}, \mathsf{sk}, \mathsf{help})$.†

$\mathcal{P}, \mathcal{V}$**:** 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 pk_i. That is, $\mathbf{Ver}_{\mathsf{pk}_i}(a_i, c^*, \overline{z}_i) \stackrel{?}{=} Accept$.
(II) The signatures σ_i for $r_i||\mathsf{pk}_i$ verify under pk. That is $\mathsf{SigVer}_{\mathsf{pk}}(r_i||\mathsf{pk}_i, \sigma_i) \stackrel{?}{=} Accept$.

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

Figure 2: The $\mathrm{DirProd}^\lambda_{n,m,t}$ identification scheme.

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 will be a three round scheme where the indices $r_1, \ldots, r_t$ are sent by the verifier with the challenge (we rely on the fact that the first messages a of the generalized Okamoto scheme do 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 are independent of the indices $r_1, \ldots, r_t$, which are only chosen later during the impersonation stage. In Appendix A 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 [IJK06], 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.

Notice 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 4.2. *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* $\delta = \frac{1}{m}(1 + \frac{\log(n)}{\lambda} + \frac{4}{n}) + \frac{2\lambda}{t} \approx \frac{1}{m} + O(\lambda/t)$.

4.4 Construction 3: Saving Communication using Compressed Direct-Products

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 $\mathsf{pk}[r_1], \dots, \mathsf{pk}[r_t]$ 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 signature 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 the previous construction, the secret key $\mathsf{sk} = (\mathsf{sk}[1], \dots, \mathsf{sk}[n])$ 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 $\mathsf{sk}[r_1], \dots, \mathsf{sk}[r_t]$ used by the protocol execution. However, instead of running parallel versions of the Okamoto_m^λ 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 $\mathsf{pk}[r_1], \dots, \mathsf{pk}[r_t]$ alone, so that the value pk^* does 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 secret keys $\{\mathsf{sk}[r_i] = (x_1[r_i], \dots, x_m[r_i])\}_{i \in [t]}$ into a single key $\mathsf{sk}^* = (x_1^*, \dots, x_m^*)$, 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 $x_j[r_1] || \dots || x_j[r_t]$. In Appendix A, 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 $\mathsf{pk}[r_1], \dots, \mathsf{pk}[r_t]$ using e as requested)? Recall that in construction 2 we used signatures to ensure that the prover uses the right public keys $\mathsf{pk}[r_i]$. Unfortunately, the prover cannot store a signature for every possible key pk^* (corresponding to all choices of $r_1, \dots, r_t$ and e) since there are exponentially many. The prover *could* store signatures of the individual public keys $\mathsf{pk}[i]$, but then, would have to send all of the individual component public keys and their signatures to validate pk^*, which defeats the whole point of compression! What we really need is a signature scheme, where the signatures $\sigma[i]$ of individual keys $\mathsf{pk}[i]$ can be combined into a *short authenticator* σ^*, which validates pk^*.

The idea of combining signatures of component blocks to validate some function over the blocks was used in the work of Shacham and Waters [SW08], and we borrow the techniques of that work. In particular, we (also) use a modification of the BLS signature scheme ([BLS01]) to compute the component signatures $\sigma[i]$. Unfortunately, our authenticators σ^* do not validate keys pk^* *on their own*. Instead, we analyze our scheme as a unit, and show that the adversary cannot do *both*: (1) come up with a public key pk^* authenticated by σ^* and (2) run the generalized Okamoto scheme using this key pk^*. Note that the use of (modified) BLS signatures requires us to work in Gap Diffie-Hellman (GDH) groups where the CDH problem is believed to be hard, but DDH tuples can be identified efficiently (see Preliminaries). In addition, the use of BLS signatures requires us to rely on the Random Oracle Model for security. 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.

$\mathsf{ParamGen}(1^\lambda)$**:** Let $(p, G, g) \leftarrow \mathcal{G}(1^\lambda)$, $g_1, \ldots, g_m, u \leftarrow_R G$. Set $\mathsf{params} = (p, G, g_1, \ldots, g_m, u)$.

$\mathsf{KeyGen}()$**:** Choose $s \leftarrow_R \mathbb{Z}_p$ and set $\mathsf{pk} = v = u^s$.
For $i = 1, \ldots, n$ set: $(\mathsf{pk}[i], \mathsf{sk}[i] = (x_1[i], \ldots, x_m[i]) \leftarrow_R \mathbf{Gen}()$, $\sigma[i] = (H(i)\mathsf{pk}[i])^s$
Set $\mathsf{sk} = (\mathsf{sk}[1], \ldots, \mathsf{sk}[n])$, $\mathsf{help} = (\mathsf{pk}[1], \ldots, \mathsf{pk}[n], \sigma[1], \ldots, \sigma[n])$ and 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 $\mathsf{sk}^* = (x_1^*, \ldots, x_m^*)$ by setting $x_j^* = \sum_{i=1}^t (x_j[r_i]) e^{i-1}$ for $j = 1, \ldots, t$.
Set $\mathsf{pk}^* = \prod_{i=1}^t \mathsf{pk}[r_i]^{(e^{i-1})}$, $\quad \sigma^* = \prod_{i=1}^t \sigma[r_i]^{(e^{i-1})}$, $\quad \overline{z} = \mathbf{Z}_{\mathsf{sk}^*}(c^*, \overline{y})$.
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, $\mathbf{Ver}_{\mathsf{pk}^*}(a, c^*, \overline{z}) = Accept$.
(II) The value $(u, v, (\mathsf{pk}^* \prod_{i=1}^t H(r_i)^{e^{i-1}}), \sigma^*)$ is a DDH tuple.

†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.

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

Theorem 4.3. *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* $\delta = \frac{1}{m}(1 + \frac{\log(n)}{\lambda} + \frac{10}{n}) + \frac{6\lambda}{t} \approx \frac{1}{m} + O(\lambda/t)$.

5 Existentially and Entropically Unforgeable Signatures

In this section we consider 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.* An attack against entropic unforgeability is only valid if, the attacker manages to forge a message which is chosen from some distribution of significant entropy, *after* the adversary finishes interacting with the leakage oracle. 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 algorithms $(\mathsf{ParamGen}, \mathsf{KeyGen}, \mathsf{Sign}, \mathsf{Verify})$. The $\mathsf{ParamGen}$ algorithm produces some public parameters, params, which can be shared by all the users and are (implicitly) available to all of the other algorithms. The KeyGen algorithm produces a key tuple $(\mathsf{verk}, \mathsf{help}, \mathsf{sigk})$ where help has the same semantics as in ID schemes: it is given to the adversary and need not be kept secret, but it is also not needed by honest users to verify signatures. Lastly $\mathsf{Sign}_{\mathsf{sigk},\mathsf{help}}, \mathsf{Verify}_{\mathsf{verk}}$ are the standard signing and verification algorithms. 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 *entropic unforgeability attack game* $\mathsf{EUG}_\ell^\lambda$ appears in Figure 4. We use $\mathcal{S}_{\mathsf{sigk}}(\cdot)$ to denote the *signing oracle* which accepts arbitrary messages $m \in \{0,1\}^*$ as input, and outputs $\sigma = \mathsf{Sign}_{\mathsf{sigk},\mathsf{help}}(m)$.

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

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).

Figure 4: Entropic Unforgeability Attack Game

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 known ahead of time by $\mathcal{A}_1$. When discussing signature schemes in the random-oracle model, we assume that the adversary $\mathcal{A}_1, \mathcal{A}_2$ *and* the leakage functions f submitted to $\mathcal{O}_{\mathsf{sk}}^{\lambda,\ell}(\cdot)$, all have access to the random oracle.

Definition 5.1. *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.*[3] *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 $\mathcal{A}$.*

We now show that the Fiat-Shamir heuristic [FS86] can be used to construct entropically (resp. existentially) unforgeable signature schemes secure against ℓ bits of key leakage, from ID schemes secure against pre-impersonation (resp. anytime) leakage of ℓ bits. Recall that, for 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 5.1. *Let $\mathsf{ID} = (\mathsf{ParamGen}, \mathsf{KeyGen}, \mathcal{P}, \mathcal{V})$ by a public coins ID scheme consisting of three rounds of interaction initiated by the prover. Let $\mathsf{Sig} = (\mathsf{ParamGen}, \mathsf{KeyGen}, \mathsf{Sign}, \mathsf{Verify})$ 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 Okamoto$_m^\lambda$ identification scheme, analyzed in Theorem 4.1, 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

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

the BRM setting, we can instead use $\mathrm{CompDirProd}^{\lambda}_{n,m,t}$, analyzed in Theorem 4.3 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 Authenticated Key Agreement

Using our leakage-resilient entropically-unforgeable signatures, we construct a leakage-resilient key agreement protocol eSIG-DH. The details of the model, security notion, construction and proof are in Appendix C. Bellow we a give high level overview of the result.

Model & Security Notion: We adapt the notion of *SK-security with perfect forward secrecy* from Canetti and Krawczyk in [CK01]. In particular, we prove security in the unauthenticated-links (UM) model with erasures where we consider a "man-in-the-middle" adversary playing against multiple concurrent sessions between n players $\mathcal{P}_1, \ldots, \mathcal{P}_n$. Further, every session is initiated by the adversary which also chooses the intended peer with whom the session owner wishes to establish a shared session key. Any such efficient adversary is called a KA-adversary.

The main modification we make to the [CK01] model is to augment the abilities of a KA-adversary $\mathcal{A}$ with an attack called a leak query. In particular we give $\mathcal{A}$ access to n leakage oracles $\mathcal{O}^{\lambda,\ell}_{\mathsf{sk}_i}(\cdot)$ where sk_i is the secret key of player $\mathcal{P}_i$ which can be called at any time, adaptively, and in any order. Intuitively our definition guarantees the security of a session key for any session θ between players $\mathcal{P}_i$ and $\mathcal{P}_j$ as long as:

1. Before session θ is initiated, $\mathcal{A}$ learns at most ℓ bits through leak query attacks against $\mathcal{P}_i$ and ℓ bits in leak query attacks against $\mathcal{P}_j$.
2. During session θ, $\mathcal{A}$ makes no leak query attacks against either player.

In particular, we require *perfect forward secrecy*: the key exchanged in θ remains secure even if $\mathcal{A}$ learns the entire long-term secrets and internal states of $\mathcal{P}_i$ and $\mathcal{P}_j$ after θ is complete and the session key has been deleted from their memory. (This security property is called.)

The Protocol: Our construction is essentially an instantiation of the SIG-DH construction of [CK01]. In SIG-DH a passive key agreement protocol (namely the Diffie-Hellman protocol) is authenticated with a signature scheme. We only modify the construction by plugging in our entropically-secure signatures with leakage ℓ. Since each party signs a random group element chosen by the other party (and hence having entropy), our entropically-unforgeable signatures are sufficient for this setting. We refer to the resulting protocol as eSIG-DH and the details of a slightly optimized construction are given in Figure 6.

Theorem 6.1. *Let* Sig *be a entropically-unforgeable signature scheme with leakage ℓ. Then, under the* DDH *assumption,* eSIG-DH *is an ℓ-*SK-secure KA *protocol with perfect forward secrecy.*

We notice that the above theorem constructs a KA protocol which essentially preserves the efficiency of the underlying signature scheme. In particular, plugging in our specific instantiation, we get authenticated key agreement protocols with the parameters claimed in Table 1.

Remark 6.1 (Generalization of Diffie-Hellman). We note that, although the construction of eSIG-DH makes use of a Diffie-Hellman key agreement protocol, we can use any Key Encapsulation Mechanism (formalized in [CS04]) in its place. More generally, it seems that any passively secure key agreement protocol could be used. However, in general, this may require that players supply each other with random challenges to be included with each message signed.

Remark 6.2 (Using existentially-unforgeable signatures). Another option is to instantiate $\mathsf{eSIG\text{-}DH}$ with a leakage-resilient, existentially-unforgeable signature scheme. This allows for the strengthening of the security notion of the resulting AKA, in that security is maintained even if the adversary gets to perform leakage attacks *during* a protocol execution.In particular, this allows us to guarantee the security of sessions against which a $\mathsf{leak\ query}$ is made, provided that ephemeral state (and the agreed upon session-key) is not leaked. On the other hand, this requires that the amount of information leaked is smaller then the length of a signature, and thus is not suitable for the BRM setting.

7 Invisible Key Updates

We notice that our schemes also allow for efficient updates of the secret key, using an externally stored "master update key". 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 helper $\mathsf{help} = (\mathsf{help}[1], \ldots, \mathsf{help}[n])$ of sizes n, and then deleted immediately afterwards. However, we can store this "mater key" on a separate external device which is not susceptible to leakage, as a "mater update key". Then, to update the secret-key database, we simply overwrite the current secret-keys (and helper values) with the "next" n values. That is, $\mathsf{sk} := (\mathsf{sk}[nk+1], \ldots, \mathsf{sk}[n(k+1)]), \mathsf{help} = (\mathsf{help}[nk+1], \ldots, \mathsf{help}[n(k+1)])$ 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 for which the adversary can successfully run an impersonation attack. The signature schemes and AKA protocols are then just based on the new ID scheme. Notice that our 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.

References

[ADR02] Yonatan Aumann, Yan Zong Ding, and Michael O. Rabin. Everlasting security in the bounded storage model. *IEEE Transactions on Information Theory*, 48(6):1668–1680, 2002.

[AGV09] A. Akavia, S. Goldwasser, and V. Vaikuntanathan. Simultaneous hardcore bits and cryptography against memory attacks. In Omer Reingold, editor, *Theory of Cryptography — TCC 2009*, volume 5444 of *Lecture Notes in Computer Science*. Springer-Verlag, 2009.

[AR99] Yonatan Aumann and Michael O. Rabin. Information theoretically secure communication in the limited storage space model. In Wiener [Wie99], pages 65–79.

[BBCM95] Charles H. Bennett, Gilles Brassard, Claude Crépeau, and Ueli M. Maurer. Generalized privacy amplification. *IEEE Transactions on Information Theory*, 41(6):1915–1923, 1995.

[BBR88] Charles H. Bennett, Gilles Brassard, and Jean-Marc Robert. Privacy amplification by public discussion. *SIAM J. Comput.*, 17(2):210–229, 1988.

[BDL97] Dan Boneh, Richard A. DeMillo, and Richard J. Lipton. On the importance of checking cryptographic protocols for faults (extended abstract). In *EUROCRYPT*, pages 37–51, 1997.

[BLS01] Dan Boneh, Ben Lynn, and Hovav Shacham. Short signatures from the weil pairing. In Colin Boyd, editor, *ASIACRYPT*, volume 2248 of *Lecture Notes in Computer Science*, pages 514–532. Springer, 2001.

[BM93] Steven M. Bellovin and Michael 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.

[BMP00] Victor Boyko, Philip D. MacKenzie, and Sarvar Patel. Provably secure password-authenticated key exchange using diffie-hellman. In *EUROCRYPT*, pages 156–171, 2000.

[BPR00] Mihir Bellare, David Pointcheval, and Phillip Rogaway. Authenticated key exchange secure against dictionary attacks. In *EUROCRYPT*, pages 139–155, 2000.

[BS97] Eli Biham and Adi Shamir. Differential fault analysis of secret key cryptosystems. In Burton S. Kaliski Jr., editor, *CRYPTO*, volume 1294 of *Lecture Notes in Computer Science*, pages 513–525. Springer, 1997.

[CDD+07] David Cash, Yan Zong Ding, Yevgeniy Dodis, Wenke Lee, Richard J. Lipton, and Shabsi Walfish. Intrusion-resilient key exchange in the bounded retrieval model. In Salil P. Vadhan, editor, *TCC*, volume 4392 of *Lecture Notes in Computer Science*, pages 479–498. Springer, 2007.

[CDH+00] Ran Canetti, Yevgeniy Dodis, Shai Halevi, Eyal Kushilevitz, and Amit Sahai. Exposure-resilient functions and all-or-nothing transforms. In *EUROCRYPT*, pages 453–469, 2000.

[CDS94] Ronald Cramer, Ivan Damgård, and Berry Schoenmakers. Proofs of partial knowledge and simplified design of witness hiding protocols. In Yvo Desmedt, editor, *CRYPTO*, volume 839 of *Lecture Notes in Computer Science*, pages 174–187. Springer, 1994.

[CK01] Ran Canetti and Hugo Krawczyk. Analysis of key-exchange protocols and their use for building secure channels. In Pfitzmann [Pfi01], pages 453–474.

[CK02] Ran Canetti and Hugo Krawczyk. Security analysis of ike's signature-based key-exchange protocol. In Yung [Yun02], pages 143–161.

[CLW06] Giovanni Di Crescenzo, Richard J. Lipton, and Shabsi Walfish. Perfectly secure password protocols in the bounded retrieval model. In Halevi and Rabin [HR06], pages 225–244.

[CS04] Ronald Cramer and Victor Shoup. Design and analysis of practical public-key encryption schemes secure against adaptive chosen ciphertext attack. *SIAM J. Comput.*, 33(1):167–226, 2004.

[DKL09] Yevgeniy Dodis, Yael Tauman Kalai, and Shachar Lovett. On cryptography with auxiliary input. In *STOC*, 2009. To Appear.

[DKRS06] Yevgeniy Dodis, Jonathan Katz, Leonid Reyzin, and Adam Smith. Robust fuzzy extractors and authenticated key agreement from close secrets. In *CRYPTO*, pages 232–250, 2006.

[DORS08] Yevgeniy Dodis, Rafail Ostrovsky, Leonid Reyzin, and Adam Smith. Fuzzy extractors: How to generate strong keys from biometrics and other noisy data. *SIAM J. Comput.*, 38(1):97–139, 2008.

[DP07] Stefan Dziembowski and Krzysztof Pietrzak. Intrusion-resilient secret sharing. In *FOCS*, pages 227–237. IEEE Computer Society, 2007.

[DP08] Stefan Dziembowski and Krzysztof Pietrzak. Leakage-resilient cryptography. In *FOCS*, pages 293–302. IEEE Computer Society, 2008.

[DSS01] Yevgeniy Dodis, Amit Sahai, and Adam Smith. On perfect and adaptive security in exposure-resilient cryptography. In *EUROCRYPT*, pages 301–324, 2001.

[DW09] Yevgeniy Dodis and Daniel Wichs. Non-malleable extractors and symmetric key cryptography from weak secrets. In *STOC*, 2009. To Appear. Full version at `http://eprint.iacr.org/2008/503`.

[Dzi06] Stefan Dziembowski. Intrusion-resilience via the bounded-storage model. In Halevi and Rabin [HR06], pages 207–224.

[FS86] Amos Fiat and Adi Shamir. How to prove yourself: Practical solutions to identification and signature problems. In Andrew M. Odlyzko, editor, *CRYPTO*, volume 263 of *Lecture Notes in Computer Science*, pages 186–194. Springer, 1986.

[GL06] Oded Goldreich and Yehuda Lindell. Session-key generation using human passwords only. *J. Cryptology*, 19(3):241–340, 2006.

[GMO01] Karine Gandolfi, Christophe Mourtel, and Francis Olivier. Electromagnetic analysis: Concrete results. In Çetin Kaya Koç, David Naccache, and Christof Paar, editors, *CHES*, volume 2162 of *Lecture Notes in Computer Science*, pages 251–261. Springer, 2001.

[Gur04] Venkatesan Guruswami. *List Decoding of Error-Correcting Codes (Winning Thesis of the 2002 ACM Doctoral Dissertation Competition)*, volume 3282 of *Lecture Notes in Computer Science*. Springer, 2004.

[HR06] Shai Halevi and Tal Rabin, editors. *Theory of Cryptography, Third Theory of Cryptography Conference, TCC 2006, New York, NY, USA, March 4-7, 2006, Proceedings*, volume 3876 of *Lecture Notes in Computer Science*. Springer, 2006.

[HSH+08] J. Alex Halderman, Seth D. Schoen, Nadia Heninger, William Clarkson, William Paul, Joseph A. Calandrino, Ariel J. Feldman, Jacob Appelbaum, and Edward W. Felten. Lest we remember: Cold boot attacks on encryption keys. In Paul C. van Oorschot, editor, *USENIX Security Symposium*, pages 45–60. USENIX Association, 2008.

[IJK06] Russell Impagliazzo, Ragesh Jaiswal, and Valentine Kabanets. Approximately list-decoding direct product codes and uniform hardness amplification. In *FOCS*, pages 187–196. IEEE Computer Society, 2006.

[IPSW06] Yuval Ishai, Manoj Prabhakaran, Amit Sahai, and David Wagner. Private circuits ii: Keeping secrets in tamperable circuits. In Serge Vaudenay, editor, *EUROCRYPT*, volume 4004 of *Lecture Notes in Computer Science*, pages 308–327. Springer, 2006.

[ISW03] Yuval Ishai, Amit Sahai, and David Wagner. Private circuits: Securing hardware against probing attacks. In Dan Boneh, editor, *CRYPTO*, volume 2729 of *Lecture Notes in Computer Science*, pages 463–481. Springer, 2003.

[KJJ99] Paul C. Kocher, Joshua Jaffe, and Benjamin Jun. Differential power analysis. In Wiener [Wie99], pages 388–397.

[Koc96] Paul C. Kocher. Timing attacks on implementations of diffie-hellman, rsa, dss, and other systems. In Neal Koblitz, editor, *CRYPTO*, volume 1109 of *Lecture Notes in Computer Science*, pages 104–113. Springer, 1996.

[KOY01] Jonathan Katz, Rafail Ostrovsky, and Moti Yung. Efficient password-authenticated key exchange using human-memorable passwords. In Pfitzmann [Pfi01], pages 475–494.

[KR09] Bhavana Kanukurthi and Leonid Reyzin. Key agreement from close secrets over unsecured channels. In *EUROCRYPT*, 2009. To Appear. Full version at http://eprint.iacr.org/2008/494.

[Kra05] Hugo Krawczyk. Hmqv: A high-performance secure diffie-hellman protocol. In Victor Shoup, editor, *CRYPTO*, volume 3621 of *Lecture Notes in Computer Science*, pages 546–566. Springer, 2005.

[KV09] Yael Tauman Kalai and Vinod Vaikuntanathan. Public-key encryption schemes with auxiliary inputs and applications. 2009. Personal Communication.

[Lu02] Chi-Jen Lu. Hyper-encryption against space-bounded adversaries from on-line strong extractors. In Yung [Yun02], pages 257–271.

[Mau92a] Ueli M. Maurer. Conditionally-perfect secrecy and a provably-secure randomized cipher. *J. Cryptology*, 5(1):53–66, 1992.

[Mau92b] Ueli M. Maurer. Protocols for secret key agreement by public discussion based on common information. In *CRYPTO*, pages 461–470, 1992.

[MR04] Silvio Micali and Leonid Reyzin. Physically observable cryptography (extended abstract). In Moni Naor, editor, *TCC*, volume 2951 of *Lecture Notes in Computer Science*, pages 278–296. Springer, 2004.

[MW97] Ueli M. Maurer and Stefan Wolf. Privacy amplification secure against active adversaries. In *CRYPTO*, pages 307–321, 1997.

[NS09] Moni Naor and Gil Segev. Public-key cryptosystems resilient to key leakage. In *Cryptology ePrint Archive, Report 2009/105*, 2009. `http://eprint.iacr.org/2009/105`.

[NZ96] Noam Nisan and David Zuckerman. Randomness is linear in space. *J. Comput. Syst. Sci.*, 52(1):43–52, 1996.

[Oka92] Tatsuaki Okamoto. Provably secure and practical identification schemes and corresponding signature schemes. In Ernest F. Brickell, editor, *CRYPTO*, volume 740 of *Lecture Notes in Computer Science*, pages 31–53. Springer, 1992.

[OS90] H. Ong and Claus-Peter Schnorr. Fast signature generation with a fiat shamir-like scheme. In *EUROCRYPT*, pages 432–440, 1990.

[Pfi01] Birgit Pfitzmann, editor. *Advances in Cryptology - EUROCRYPT 2001, International Conference on the Theory and Application of Cryptographic Techniques, Innsbruck, Austria, May 6-10, 2001, Proceeding*, volume 2045 of *Lecture Notes in Computer Science*. Springer, 2001.

[Pie09] Krzysztof Pietrzak. A leakage-resilient mode of operation. In *Eurocrypt 2009, Cologne, Germany*, 2009.

[QS01] Jean-Jacques Quisquater and David Samyde. Electromagnetic analysis (ema): Measures and counter-measures for smart cards. In Isabelle Attali and Thomas P. Jensen, editors, *E-smart*, volume 2140 of *Lecture Notes in Computer Science*, pages 200–210. Springer, 2001.

[Reg05] Oded Regev. On lattices, learning with errors, random linear codes, and cryptography. In Harold N. Gabow and Ronald Fagin, editors, *STOC*, pages 84–93. ACM, 2005.

[RW03] Renato Renner and Stefan Wolf. Unconditional authenticity and privacy from an arbitrarily weak secret. In *CRYPTO*, pages 78–95, 2003.

[SW08] Hovav Shacham and Brent Waters. Compact proofs of retrievability. In Josef Pieprzyk, editor, *ASIACRYPT*, volume 5350 of *Lecture Notes in Computer Science*, pages 90–107. Springer, 2008.

[Vad04] Salil P. Vadhan. Constructing locally computable extractors and cryptosystems in the bounded-storage model. *J. Cryptology*, 17(1):43–77, 2004.

[Wie99] Michael J. Wiener, editor. *Advances in Cryptology - CRYPTO '99, 19th Annual International Cryptology Conference, Santa Barbara, California, USA, August 15-19, 1999, Proceedings*, volume 1666 of *Lecture Notes in Computer Science*. Springer, 1999.

[Yun02] Moti Yung, editor. *Advances in Cryptology - CRYPTO 2002, 22nd Annual International Cryptology Conference, Santa Barbara, California, USA, August 18-22, 2002, Proceedings*, volume 2442 of *Lecture Notes in Computer Science*. Springer, 2002.

A Entropy Preservation

Our analysis of entropy preservation is mostly based on the notion of *approximately list-decodable* codes. We take the following definition from [IJK06].

Definition A.1. *An error-correcting code* $\mathrm{enc} : \Sigma_1^n \to \Sigma_2^N$ *is* (ε, δ, L)-approximately list-decodable *if for any value* $\tilde{c} \in \Sigma_2^N$ *there exists a collection of at most* L source messages $m_1, \ldots, m_L \in \Sigma_1^n$ *such that for every* codeword $c = \mathrm{enc}(m)$ *that agrees with* $\tilde{c}$ *in at least an* ε *fraction of positions, there exists an index* $i \in [L]$ *such that the* hamming distance $d(m, m_i) \leq \delta k$*.*

For a value $c \in \Sigma^N$ we use the notation $c[i]$ to denote the ith symbol of c. The following general lemma will be the main technical tool for analyzing entropy preservation for our constructions.

Lemma A.1. *Let* $\mathrm{enc} : \Sigma_1^n \to \Sigma_2^N$ *be* (ε, δ, L)*-approximately list decodable. Let* X *be a random variable over* Σ_1^n*, and* $\mathcal{E}_1$ *be an arbitrary experiment such that* $\widetilde{\mathbf{H}}_\infty(X \mid \mathcal{E}_1) \geq \log(L/\varepsilon) + \delta n(\log(|\Sigma_1|) + \log(n))$*. Let* $\mathcal{E}_2$ *be an experiment which first runs* $\mathcal{E}_1$ *and then chooses* R *uniformly at random over* $[N]$ *and gives* R *to* $\mathcal{A}$ *. Then* $\widetilde{\mathbf{H}}_\infty(\mathrm{enc}(X)[R] \mid \mathcal{E}_2) \geq \log(1/\varepsilon) - 1$*.*

Proof. We argue the contrapositive. Assume that $\widetilde{\mathbf{H}}_\infty(\mathrm{enc}(X)[R] \mid \mathcal{E}_2) < \log(1/\varepsilon) - 1$. Then there is a statistical predictor $\mathcal{A}$ such that $\Pr[\mathcal{A}^{\mathcal{E}_2} = \mathrm{enc}(X)[R]] > 2\varepsilon$. Let us consider pairs (X, V) where V denotes a random variable for the execution of $\mathcal{A}^{\mathcal{E}_1}$ – i.e everything until the point where R_1 is chosen. We say that a pair (x, v) is "good" if $\Pr[\mathcal{A}^{\mathrm{exp}_2} = \mathrm{enc}(X)[R] \mid X = x, V = v] \geq \varepsilon$ (where the probability is taken entirely over R). We define the event $\mathbf{G}$ to be the event that the random variables (X, V) are a "good" pair. Then, using standard good-bad analysis, we get $\Pr[\mathbf{G}] > \varepsilon$.
We now define a machine $\mathcal{B}$ which runs in the experiment $\mathcal{E}_1$ and acts as follows:

1. Run $\mathcal{A}$ with the oracle $\mathcal{E}_1(\cdot)$. At the conclusion, of this process, run $\mathcal{A}$ all N possible values of R. This results in some N predictions $\tilde{c} = (\tilde{c}[1], \ldots, \tilde{c}[N])$. Approximately list-decode $\tilde{c}$ to a set of L values $m_1, \ldots, m_L$ as defined in Definition A.1.

2. Choose m_i uniformly at random from this list.

3. Choose a random subset $S \subset [n]$ of size δn.

4. Output a value $m \in \Sigma_1^n$ by choosing $m[j]$ uniformly at random from Σ_1 for each $j \in S$, and setting $m[j] := m_i[j]$ for each $j \notin S$.

We define the events: E_1 is the event (X, V) is a good pair, E_2 is the event that $d(\mathrm{enc}(X), m_i) \leq \delta n$, E_3 is the event that $m_i = \mathrm{enc}(X)$ for each $j \notin S$, and E_4 is the event that $m = \mathrm{enc}(X)$. Then

$$\begin{aligned}
\Pr[\mathcal{B}^{\mathcal{E}_1(\cdot)} = X] = \Pr[E_4] \quad &\geq \quad \Pr[E_1]\Pr[E_2 \mid E_1]\Pr[E_3 \mid E_1, E_2]\Pr[E_4 \mid E_3, E_2, E_1] \\
&\geq \quad \Pr[\mathbf{G}]\left(\frac{1}{L}\right)\left(\frac{1}{n}\right)^{\delta n}\left(\frac{1}{|\Sigma|}\right)^{\delta n} \\
&> \quad \left(\frac{\varepsilon}{L}\right)2^{-\delta n(\log(|\Sigma_1|)+\log(n))}
\end{aligned} \tag{1}$$

where (1) follows by noting:

- $\Pr[E_2 \mid E_1] \geq 1/L$. The event E_1 implies that $\mathcal{A}$ answers an ε fraction of the positions $\tilde{c}[i]$ correctly and therefore at least one of the values $m_1, \ldots, m_L$ is of distance δn from X.

- $\Pr[E_3 \mid E_2, E_1]\left(\frac{1}{n}\right)^{\delta n}$. If E_2 occurs then there at most δn incorrect positions in m_i. Therefore, at least one choice of δn indices includes all of such positions.

- $\Pr[E_4 \mid E_3, E_2, E_1] \geq \left(\frac{1}{|\Sigma|}\right)^{\delta n}$. This is the event that $m[j] \in \Sigma$ is chosen correctly for each $j \in S$. There are exactly δn independent choices, each uniform over the set Σ.

Therefore $\widetilde{\mathbf{H}}_\infty(X|\mathcal{E}_1) < \log(L/\varepsilon) + \delta n(\log(|\Sigma_1|) + \log(n))$, concluding the proof. □

We now apply the above lemma to some specific encodings. The first such code is the "direct-product" code $\mathsf{DirProd} : \Sigma^n \times [n]^t \to \Sigma^t$ which takes a value $x = (x[1], \ldots, x[n]) \in \Sigma^n$ and indices $(r_1, \ldots, r_t) \in [n]^t$ and outputs $(x[r_1], \ldots, x[r_t])$. Interestingly, the idea of using direct products is reminiscent of lemmas in [NZ96, Vad04] where it is shown that, for the special case where the alphabet Σ consists of bits, the entropy rate is preserved under direct product codes (in fact, these lemmas allow for the use of randomness-efficient oblivious samplers). Our lemma, on the other hand, is proven for arbitrary alphabets, and is more suitable for our needs. First we recall that direct-product codes are good approximately list decodable codes.

Lemma A.2 (from [IJK06]). *Let* enc *be the t-wise direct product code over an initial alphabet Σ. Then, for every $\varepsilon > 0, \delta \geq 2\log(1/\varepsilon)/t$ the code* enc *is (ε, δ, L)-approximately list-decodable where $L \leq 1/(\varepsilon^2 - (1-\delta)^t) \leq 1/(\varepsilon^2 - e^{-\delta t}) \leq 4/\varepsilon^2$.*

Now, taking the above lemma in conjunction with Lemma A.1, we get the following parameters.

Lemma A.3. *Let* $\mathsf{DirProd}$ *be the t-wise direct product code over an alphabet Σ of size q. Let X be a random variable over Σ_1^n and $\mathcal{E}_1$ be an arbitrary experiment. Define $\mathcal{E}_2$ to be the experiment where, at the conclusion of $\mathcal{E}_1$, a uniformly random index $R \in [N]$ is chosen and given to the predictor. Then:*
(1) For any $c > 0$, if $\widetilde{\mathbf{H}}_\infty(X \mid \mathcal{E}_1) \geq \frac{2c}{t} n(\log(q) + \log(n)) + 3c + 5$ then $\widetilde{\mathbf{H}}_\infty(\mathsf{DirProd}(X, R) \mid \mathcal{E}_2) > c$.
(2) For alphabets of size $q = p^m$, assume $\widetilde{\mathbf{H}}_\infty(X \mid \mathcal{E}_1) \geq n(m-1)\log(p) - \ell$ where $\ell \leq (1-\delta)nm\log(p)$ for some δ. Then $\widetilde{\mathbf{H}}_\infty(\mathsf{DirProd}(X, R) \mid \mathcal{E}_2) > c$ if $\delta \geq [\frac{2c}{t} + \frac{(1+\log(n)/\log(p))}{m} + \frac{3c+5}{nm\log(p)}]$.

Proof. By Lemma A.2, the t-wise direct product code is (ε, δ, L)-approximately list decodable for any $\varepsilon \geq 0$ and any $\delta \geq 2\log(1/\varepsilon)/t$ with $L = 4/\varepsilon^2$. Setting $c = \log(1/\varepsilon) - 1$, $\delta = (2c-2)/t$, we see that the first part of the corollary follows from Lemma A.1.
For the second part, $\widetilde{\mathbf{H}}_\infty(\mathsf{DirProd}(X, R) \mid \mathcal{E}_2) \geq c$ is implied by

$$\widetilde{\mathbf{H}}_\infty(X \mid \mathcal{E}_1) \geq n(m-1)\log(p) - \ell \geq (2c/t)n(m\log(p) + \log(n)) + 3c + 5$$

which is in turn implied by

$$\begin{aligned} \ell &\leq n(m-1)\log(p) - (2c/t)n(m\log(p) + \log(p)) - 3c - 5 \\ &= (1-\delta)(nm\log(p)) \end{aligned}$$

□

We can also apply our main entropy lemma (Lemma A.1) to standard error correcting codes with (very large) distance D. First we recall that such codes are good *list-decodable* codes.

Lemma A.4 (Johnson Bound from [Gur04]). *For any $[N, k, D]_q$ code over an alphabet Σ, any $r \in \Sigma^N$ the number of codewords that agree with r in at least an $\varepsilon \geq \sqrt{2\frac{N-D}{N}}$ fraction of positions is bounded by $L = \frac{2}{\varepsilon^2}$.*

Now, taking the above lemma in conjunction with Lemma A.1 we get the following parameters.

Corollary A.1. *Let* enc *be an error-correcting code with distance D, X be a random variable over Σ_1^n, and $\mathcal{E}_1$ be an arbitrary experiment. Define $\mathcal{E}_2$ to be the experiment where, at the conclusion of $\mathcal{E}_1$, a uniformly random index $R \in [N]$ is chosen and given to the predictor. Then for any $c > 0$, if $\widetilde{\mathbf{H}}_\infty(X \mid \mathcal{E}_1) \geq 3c + 1$ then $\widetilde{\mathbf{H}}_\infty(\mathrm{enc}(X)[R] \mid \mathcal{E}_2) \geq \min\left(c, \frac{1}{2}(\log(\frac{N}{N-D}) - 1)\right)$.*

Proof. The Johnson bound shows that enc is $(\varepsilon, 0, L)$ list decodable for any $\varepsilon \geq \sqrt{2\frac{N-D}{N}}$ with $L = \frac{2}{\varepsilon^2}$. Therefore, by Lemma A.1, if $\widetilde{\mathbf{H}}_\infty(X|\mathcal{E}_1) \geq 3\log(1/\varepsilon) + 1$ then $\widetilde{\mathbf{H}}_\infty(\mathrm{enc}(X)[R] \mid Z, R) \geq \log(1/\varepsilon) - 1$. Setting $c = \log(1/\varepsilon)$ we see that the theorem holds for any $c \leq \log\left(\sqrt{\frac{N}{2(N-D)}}\right) \leq \frac{1}{2}(\log(N/(N-D)) - 1)$. □

Lastly, we consider the *compressed direct-product code* $\mathsf{CompDirProd}$ which is used in the ID scheme of Construction 3. This code may not appear very natural from an information theoretic sense, but it ends up being useful in our constructions where we must match information theoretic codes with computational assumptions. We define the function $\mathsf{CompDirProd} : \mathbb{F}^{n \times m} \times ([n]^t \times \mathbb{F}) \rightarrow \mathbb{F}^m$, over fields $\mathbb{F}$ of size p. It is actually easier to define $\mathsf{CompDirProd}$ as a composition of the direct product function $\mathsf{DirProd} : (\mathbb{F}^m)^n \times [n]^t \to (\mathbb{F}^m)^t$ (over the alphabet $\mathbb{F}^m$) and a $\mathsf{Comp} : (\mathbb{F}^{t\times m}) \times \mathbb{F} \to (\mathbb{F})^m$ defined by $\mathsf{Comp}(x, e) = (w_1, \ldots, w_m)$ where $w_j = \sum_{i=1}^{t}(x[i][j])e^{i-1}$ is the eth symbol of the Reed-Solomon encoding of column j of the matrix x. Looking at Comp as an error correcting code (where the $\mathsf{Comp}(x, e)$ defines the eth symbol of the encoding of x), any t positions in the codeword uniquely determine the source value x and thus the distance of this code is $D \geq p - t + 1$. This allows us to use Lemma A.3 and Corollary A.1 to analyze the function $\mathsf{CompDirProd}$.

Lemma A.5. *Let $\mathsf{CompDirProd} : \mathbb{F}^{n\times m} \times ([n]^t \times \mathbb{F}) \to \mathbb{F}^m$ be as above, and let $\mathcal{E}_1$ be an arbitrary experiment such that $\widetilde{\mathbf{H}}_\infty(X \mid \mathcal{E}_1) \geq n(m-1)\log(p) - \ell$ for $\ell \leq (1-\delta)nm\log(p)$ for some δ. Define $\mathcal{E}_2$ to be the experiment where $\mathcal{E}_1$ is run first, and at its conclusion, a uniformly random index $R = (R_1, R_2) \in ([n]^t \times \mathbb{F})$ is chosen and given to the predictor. Then $\widetilde{\mathbf{H}}_\infty(\mathsf{CompDirProd}(X, R) \mid \mathcal{E}_2) \geq \min(c, (\log(p) - \log(t) - 1)/2)$ if $\delta \geq [\frac{6c+2}{t} + \frac{(1+\log(n)/\log(p))}{m} + \frac{9c+8}{nm\log(p)}]$.*

Proof. Let us define $\mathcal{E}_{1.5}$ where the experiment $\mathcal{E}_1$ is run, and then only the value R_1 is sent. By Lemma A.3, we get $\widetilde{\mathbf{H}}_\infty(\mathsf{DirProd}(X, R_1) \mid \mathcal{E}_{1.5}) \geq c'$ where $c' = 3c + 1$. Then, recalling that $\mathsf{CompDirProd}(X, (R_1, R_2)) = \mathsf{Comp}(\mathsf{DirProd}(X, R_1), R_2)$ we get,

$$\widetilde{\mathbf{H}}_\infty(\mathsf{CompDirProd}(X, (R_1, R_2)) \mid \mathcal{E}_2) = \widetilde{\mathbf{H}}_\infty(\mathsf{Comp}(\mathsf{DirProd}(X, R_1), R_2) \mid \mathcal{E}_2) \geq \min(c, (\log(p) - \log(t) - 1)/2)$$

where the last inequality follows from Corollary A.1, applied to $X' = \mathsf{DirProd}(X, R_1)$. □

B Proofs Omitted From Main Body

B.1 Lemma 2.1

Proof of Lemma 2.1. By the (perfect) HVZK assumption, for any $(x, w) \in \mathcal{R}$, there is a simulator $\mathcal{S}_{\mathsf{HVZK}}(x)$ which defines a joint distribution (A, C, Z) on conversations (a, c, z) produced by $\{\mathcal{P}(x, w) \rightleftharpoons \mathcal{V}(x)\}$ (honest prover, verifier). We claim that, for any $(x, w) \in \mathcal{R}$ and any arbitrary malicious verifier $\mathcal{V}'(x, w)$, the distribution $\{\mathcal{P}(x, w) \rightleftharpoons \mathcal{V}'(x, w)\}$ can be simulated (inefficiently) by the on-line simulator $\mathcal{S}(x)$ in the following manner:
(1) $\mathcal{S}(x)$ chooses $a \leftarrow_R A$ and sends it to $\mathcal{V}'(x, w)$ (2) $\mathcal{S}(x)$ gets a challenge c (3) $\mathcal{S}(x)$ samples $z \leftarrow_R (Z|A = a, C = c)$ and sends z to $\mathcal{V}'(x, w)$.
The simulation by $\mathcal{S}(x)$ perfectly matches the actual execution since (1) by the perfect HVZK property, $\mathcal{S}(x)$ samples a from the same distribution as $\mathcal{P}(x, w)$ (2) by the perfect HVZK property, for any particular choice of $A = a, C = c$, the value z is sampled from the same distribution as that of $\mathcal{P}(x, w)$ computing z honestly, given that the first message was a and the challenge was c.
Since the simulator $\mathcal{S}(x)$ can replace the oracle $\mathcal{P}(x, w)$ in any experiment in which the predictor $\mathcal{A}$ is given x, the entropy of W is the same in experiments $\mathcal{E}_1$ and $\mathcal{E}_2$. □

B.2 Lemma 3.1

Proof of Lemma 3.1. Assume that for some predictor $\mathcal{A}$, $\Pr[\mathcal{A}^{\mathcal{E}_2(\cdot)} = \mathsf{SK}] \geq \varepsilon$. Then we can construct a predictor $\mathcal{B}^{\mathcal{E}_1(\cdot)}$ which runs $\mathcal{A}$ internally and, each time $\mathcal{A}$ attempts to query $\mathcal{O}^{\lambda,\ell}_{\mathsf{SK}}(\cdot)$ with a leakage function that has range $\{0,1\}^{\alpha_i}$, the predictor $\mathcal{B}$ just gives a random string of length α_i to $\mathcal{A}$. Let G be the event that $\mathcal{B}$ gives the same answers as $\mathcal{O}^{\lambda,\ell}_{\mathsf{SK}}(\cdot)$ would give on these queries. Then $\Pr[G] \geq 2^{-\ell}$ since $\mathcal{B}$ has to guess at most ℓ bits during the execution. Therefore $\Pr[\mathcal{B}^{\mathcal{E}_1(\cdot)} = \mathsf{SK}] \geq \Pr[\mathcal{B}^{\mathcal{E}_1(\cdot)} = \mathsf{SK} \mid G]\Pr[G] \geq \Pr[\mathcal{A}^{\mathcal{E}_2(\cdot)} = \mathsf{SK}]2^{-\ell} \geq \varepsilon 2^{-\ell}$. □

B.3 Lemma 4.1

Proof of Lemma 4.1. For (1) we show a reduction. Assume that an adversary $\mathcal{A}$ runs in time t and solves $\mathcal{A}(\mathsf{params}) = (\mathsf{pk}, \mathsf{sk}, \mathsf{sk}')$ with $\mathsf{sk}' \neq \mathsf{sk}$ and $(\mathsf{pk}, \mathsf{sk}), (\mathsf{pk}, \mathsf{sk}') \in \mathcal{R}$ with probability ε. We use $\mathcal{A}$ to construct an adversary $\mathcal{B}$ which breaks the DL problem in time $\approx t$ and with probability ε/m. In particular, $\mathcal{B}$ receives a challenge $(g, G, p), g^x$ and attempts to compute x. The attacker $\mathcal{B}$ computes generators $g_1, \ldots, g_m$ by choosing a random index $\rho \in [m]$, setting $g_\rho := g^x$, and setting $g_j := g^{a_j}$ for uniformly random $a_j \leftarrow \mathbb{Z}_p$ for all $j \neq \rho$. Then $\mathcal{B}$ calls $\mathcal{A}$ on input $\mathsf{params} := (p, G, g_1, \ldots, g_m)$. Assume that $\mathcal{A}$ succeeds in outputting $\mathsf{pk}, \mathsf{sk} = (x_1, \ldots, x_m), \mathsf{sk}' = (x'_1, \ldots, x'_m)$ such that $\mathsf{sk}' \neq \mathsf{sk}$ and $(\mathsf{pk}, \mathsf{sk}), (\mathsf{pk}, \mathsf{sk}') \in \mathcal{R}$. Then, with probability at least $1/m$, we get $x_\rho \neq x'_\rho$ (since at least one $x_i \neq x'_i$ and the choice of ρ is independent of $\mathcal{A}$'s view). If this is the case, then:

$$
\begin{aligned}
& g^{\sum_{i=1, i\neq\rho}^m a_i x_i} (g^x)^{x_\rho} = \prod g_i^{x_i} = \prod g_i^{x'_i} = g^{\sum_{i=1, i\neq\rho}^m a_i x'_i} (g^x)^{x'_\rho} \\
\Rightarrow \quad & \sum_{i=1, i\neq\rho}^{m} a_i x_i + x x_\rho = \sum_{i=1, i\neq\rho}^{m} a_i x'_i + x x'_\rho \\
\Rightarrow \quad & x = \sum_{i=1, i\neq\rho}^{m} a_i (x'_i - x_i)/(x_\rho - x'_\rho)
\end{aligned}
$$

so $\mathcal{B}$ can easily compute x by computing the right-hand side of the above equation.

For (2), we show perfect completeness, special soundness and (perfect) HVZK. Completeness is obvious. For special soundness we note that for any two conversations (a, c, z), (a, c', z') which are both accepting for an instance pk and $c \neq c'$, we get:

$$
\begin{aligned}
& \prod_{i=1}^{m} g_i^{z_i} = a h^c \text{ and } \prod_{i=1}^{m} g_i^{z'_i} = a h^{c'} \\
\Rightarrow \quad & \prod_{i=1}^{m} g_i^{z_i - z'_i} = h^{c - c'} \\
\Rightarrow \quad & \prod_{i=1}^{m} g_i^{(z_i - z'_i)/(c - c')} = h
\end{aligned}
$$

So we see that it is easy to compute $x'_i = (z_i - z'_i)/(c - c')$ yielding a witness $\mathsf{sk}' = (x'_1, \ldots, x'_m)$ such that $(\mathsf{pk}, \mathsf{sk}') \in \mathcal{R}$. For the (perfect) HVZK property, we have a simulator $\mathcal{S}(\mathsf{pk})$ which chooses $c, z_1, \ldots, z_m$ at random from $\mathbb{Z}_p$ and sets $a := (\prod_{i=1}^m g_i^{z_i})/h^c$. For any $(\mathsf{pk}, \mathsf{sk}) \in \mathcal{R}$, the distributions $\mathcal{S}(\mathsf{pk})$ and $\{\mathcal{P}(\mathsf{pk}, \mathsf{sk}) \rightleftharpoons \mathcal{V}(\mathsf{pk})\}$ are both uniformly random over all values $(a, c, z = (z_1, \ldots, z_m))$ such that $\prod_{i=1}^m g_i^{z_i} = a h^c$. Therefore the simulation perfectly matches the honest conversation. This completes the proof of (2).

Part (3) simply follows from the fact that, for any value of pk, there are p^{m-1} possibilities for sk, each of which is equally likely. □

B.4 Theorem 4.2

Proof of Theorem 4.2. Assume that an adversary $\mathcal{A}$ runs in time t and succeeds with probability ε in the attack hame $\mathsf{IDPRE}_\ell^\lambda(\mathcal{A})$. Then, we construct an adversary $\mathcal{B}$ which runs in time $\approx 2t$ and

$$
\Pr\left[\mathsf{sk}^* \neq \hat{\mathsf{sk}}^* \text{ and } (\mathsf{pk}^*, \mathsf{sk}^*), (\mathsf{pk}^*, \hat{\mathsf{sk}}^*) \in \mathcal{R} \;\middle|\; \begin{array}{c} \mathsf{params} \leftarrow \mathsf{ParamGen}(1^\lambda) \\ (\mathsf{pk}^*, \mathsf{sk}^*, \hat{\mathsf{sk}}^*) \leftarrow \mathcal{B}(\mathsf{params}) \end{array} \right] \geq \varepsilon^2 - \frac{1}{p} - 2^{-\lambda}.
$$

The adversary $\mathcal{B}$ gets params and runs $(\mathsf{pk}, \mathsf{help}, \mathsf{sk}) \leftarrow \mathsf{KeyGen}()$. Then it gives $\mathsf{pk}, \mathsf{help}$ to the adversary $\mathcal{A}$, and runs the attack game $\mathsf{IDPRE}_\ell^\lambda(\mathcal{A})$, by acting as the challenger for $\mathcal{A}$ and providing with access to $\mathcal{O}_{\mathsf{sk}}^{\lambda,\ell}(\cdot)$ and $\mathcal{P}(\mathsf{pk},\mathsf{help},\mathsf{sk})$. Let $c_1 = (r_1,\dots,r_t,c_1^*)$ be the challenge that was chosen during this run. Then $\mathcal{B}$ rewinds $\mathcal{A}$ and gives it the challenge $c_2 = (r_1,\dots,r_t,c_2^*)$ for a fresh value $c_2^* \leftarrow \mathbb{Z}_p$ and gets a second response. Let E_1 be the event that both runs are accepting and $c_1^* \neq c_2^*$. Then, by the same analysis as in Claim 4.1, $\Pr[E_1] \geq \varepsilon^2 - 1/p$. Let E_2 be the event that the public keys $\mathsf{pk}_1,\dots,\mathsf{pk}_t$ sent by $\mathcal{A}$ match the actual public keys $\mathsf{pk}[r_1],\dots,\mathsf{pk}[r_t]$ (which were chosen by $\mathcal{B}$ during key generation) in both conversations. Then, since an incorrect public key $\mathsf{pk}_i \neq \mathsf{pk}[r_i]$ implies a forgery on the signature scheme, $\Pr[E_1 \wedge \neg E_2] \leq \mathrm{negl}(\lambda)$ and so $\Pr[E_1 \wedge E_2] \geq \varepsilon^2 - 1/p - \mathrm{negl}(\lambda)$. Finally assume E_1, E_2 occur. Then let

$$\left((a_1^{(1)},\dots,a_t^{(1)}), c_1 = (r_1,\dots,r_t,c_1^*), (z_1^{(1)},\dots,z_t^{(1)})\right) \ , \ \left((a_1^{(2)},\dots,a_t^{(2)}), c_2 = (r_1,\dots,r_t,c_2^*), (z_1^{(2)},\dots,z_t^{(2)})\right)$$

be the two accepting conversations (without public keys and signatures). Using the knowledge soundness of the generalized Okamoto scheme, these conversations can be used to efficiently retrieve secret keys $(\hat{\mathsf{sk}}_1,\dots,\hat{\mathsf{sk}}_t)$ for the public keys $(\mathsf{pk}[r_1],\dots,\mathsf{pk}[r_t])$. Let E_3 be the even that $(\hat{\mathsf{sk}}_1,\dots,\hat{\mathsf{sk}}_t) = (\mathsf{sk}[r_1],\dots,\mathsf{sk}[r_t])$, i.e. that the extracted secret keys perfectly match the ones chosen by $\mathcal{B}$ during key generation.

We claim that $\Pr[E_3] \leq 2^{-\lambda}$. For this argument, we use the fact that the constructed ID scheme is HVZK for the relation
$\mathcal{R}' = \{((\mathsf{pk},\mathsf{help}),\mathsf{sk}) \mid (\mathsf{pk}[i],\mathsf{sk}[i]) \in R, \text{ for } i = 1,\dots,n\}$. This is easy to check using the HVZK simulator for the underlying generalized Okamoto scheme.

Claim B.1. *The probability of E_3 is at most $\Pr[E_3] \leq 2^{-\lambda}$.*

Proof. Let $\mathcal{E}_1$ be the experiment in which a predictor is given the value $\mathsf{PK},\mathsf{HELP}$ and oracle access to $\mathcal{O}_{\mathsf{SK}}^{\lambda,\ell}(\cdot)$, $\mathcal{P}(\mathsf{PK},\mathsf{SK})$. Then (reusing the same analysis from Claim 4.2) we get $\widetilde{\mathbf{H}}_\infty(\mathsf{SK} \mid \mathcal{E}_1) \geq \widetilde{\mathbf{H}}_\infty(\mathsf{SK} \mid \mathsf{PK}) - \ell \geq n(m-1)\log(p) - \ell$. We can think of the rewinding process as an experiment $\mathcal{E}_2$, which first runs $\mathcal{E}_1$ and then all oracle access is taken away and the predictor gets a value $R = (R_1,\dots,R_t)$, which is random on $[n]^t$. We can think of $(\mathsf{SK}[R_1],\dots,\mathsf{SK}[R_t]) = \mathsf{DirProd}(\mathsf{SK},R)$ where $\mathsf{DirProd}$ is the direct product function which, on input a vector and indices, outputs the value of the vector at the specified indices. Now we use our analysis in Appendix A, and in particular Lemma A.3, we get $\widetilde{\mathbf{H}}_\infty((\mathsf{SK}[R_1],\dots,\mathsf{SK}[R_t]) \mid \mathcal{E}_2) \geq \lambda$. Therefore, the probability of E_3 is bounded by $2^{-\lambda}$. □

Using the above claim, $\Pr[E_1 \wedge E_2 \wedge \neg E_3] \geq \varepsilon^2 - \mathrm{negl}(\lambda)$. If $E_1 \wedge E_2 \wedge \neg E_3$ occurs, than at least one of the extracted keys is different than one of the generated keys – i.e. $\mathsf{sk}[r_i] \neq \hat{\mathsf{sk}}_i$. In this case, the adversary $\mathcal{B}$ succeeds in its attack game, by outputting the tuple $(\mathsf{pk}[r_i],\mathsf{sk}[r_i],\hat{\mathsf{sk}}_i)$. □

B.5 Theorem 4.3

Lemma B.1. *The following two properties hold for the* $\mathrm{CompDirProd}_{n,m,t}^{\lambda}$ *construction:*

1. *The protocol $(\mathcal{P},\mathcal{V})$ is* HVZK *for the relation:* $\mathcal{R}' = \{((\mathsf{pk},\mathsf{help}),\mathsf{sk}) \mid (\mathsf{pk}[i],\mathsf{sk}[i]) \in \mathcal{R}$ *for* $i = 1,\dots,n\}$.
2. $\widetilde{\mathbf{H}}_\infty(\mathsf{SK} \mid \mathsf{PK},\mathsf{HELP}) \geq n(m-1)\log(p)$.

Proof of Lemma B.1. The HVZK simulator $\mathcal{S}(\mathsf{pk},\mathsf{help})$ chooses $r_1,\dots,r_t,e$ randomly, and computes the values pk^*,σ^* using the helper string help, by running the computation specified in step (3) of Figure 3. Then $\mathcal{S}$ just runs the simulator for the generalized Okamoto scheme with the key pk^* to produce the values $(a,c^*,\overline{z})$. Putting these values together completes a conversation which follows the specified distribution. The entropy calculation in (2) follows from the fact that, for each value of $\mathsf{PK},\mathsf{HELP}$ there are $p^{n(m-1)}$ possible values for SK, each of which is equally likely. □

Proof of Theorem 4.3. First we show (perfect) completeness. We notice that

$$\mathsf{pk}^* = \prod_{i=1}^{t} \mathsf{pk}[r_i]^{(e^{i-1})} = \prod_{j=1}^{m} g_j^{(x_j^*)}$$

and so $(\mathsf{pk}^*, \mathsf{sk}^* = (x_1^*, \ldots, x_m^*))$ is a valid Okamoto key-pair. Hence, by the completeness of Okamoto, the conversation $(a, c^*, \overline{z})$ is accepting. Also,

$$\sigma^* = \prod_{i=1}^{t} \sigma[r_i]^{(e^{i-1})} = \left(\mathsf{pk}^* \prod_{i=1}^{t} H(r_i)^{(e^{i-1})}\right)^s$$

and so $(u, v = u^s, (\mathsf{pk}^* \prod_{i=1}^{t} H(r_i)^{e^{i-1}}), \sigma^*)$ is a DDH tuple. Therefore both verification conditions are always satisfied and the verifier always accepts an honest proof.

For security, assume that there is an adversary $\mathcal{A}$ which runs in time t (including the run-time of the queries for $\mathcal{O}_{\mathsf{sk}}^{\lambda,\ell}(\cdot)$) and has advantage ε in the game $\mathsf{IDPRE}_\ell^\lambda(\mathcal{A})$. Consider the following *rewinding process*. First the game $\mathsf{IDPRE}_\ell^\lambda(\mathcal{A})$ is first run to completion. Assume that, during this run, $\mathcal{A}$ received the challenge $c_1 = (r_1, \ldots, r_t, e, c_1^*)$ in the impersonation stage. Then, $\mathcal{A}$ is rewound and given a new challenge $c_2 = (r_1, \ldots, r_t, e, c_2^*)$ for a fresh c_2^*, chosen uniformly at randomly from $\mathbb{Z}_p$ (but the other components remain the same in c_1, c_2). Let

$$(a, c_1 = (r_1, \ldots, r_t, e, c_1^*), (\overline{z}^{(1)}, \mathsf{pk}_1^*, \sigma_1^*)) \quad , \quad (a, c_2 = (r_1, \ldots, r_t, e, c_2^*), (\overline{z}^{(2)}, \mathsf{pk}_2^*, \sigma_2^*))$$

be the two *received conversations* produced by $\mathcal{A}$ during this rewinding process. Let E_1 be the event that both conversations are accepting (meaning that that they satisfy properties (I),(II) defined in Figure 3) *and* that $c_1^* \neq c_2^*$. Then, following the same analysis as in the proof of Claim 4.1, the probability of E_1 is at least ε' where $\varepsilon' \geq \varepsilon^2 - 1/p$. We define $\mathsf{pk}^*, \mathsf{sk}^* = (x_1^*, \ldots, x_m^*), \sigma^*$ to be the values corresponding to an *honest conversation*, which would be produced by an honest prover $\mathcal{P}(\mathsf{pk}, \mathsf{sk})$ on challenges of the form $(r_1, \ldots, r_t, e, \cdot)$. Let E_2 be the event that $(\mathsf{pk}_1^*)^{c_1^*}/(\mathsf{pk}_2^*)^{c_2^*} \neq (\mathsf{pk}^*)^{c_1^* - c_2^*}$. Then either $\Pr[E_1 \wedge E_2] \geq \varepsilon'/2$ (case A) or $\Pr[E_1 \wedge \neg E_2] \geq \varepsilon'/2$ (case B). We break up the rest of the analysis by these two cases. For both cases we use the fact that, if condition (I) holds, then:

$$a(\mathsf{pk}_1^*)^{c_1^*} = \prod_{j=1}^{m} g_j^{z_j^{(1)}}, \quad a(\mathsf{pk}_2^*)^{c_2^*} = \prod_{j=1}^{m} g_j^{z_j^{(2)}} \quad \Longrightarrow \quad \frac{(\mathsf{pk}_1^*)^{c_1^*}}{(\mathsf{pk}_2^*)^{c_2^*}} = \prod_{j=1}^{m} g_j^{z_j^{(1)} - z_j^{(2)}} \tag{2}$$

Case A: We show that the adversary $\mathcal{A}$ can be used to break the CDH problem. Let $\mathcal{B}$ be a CDH adversary, which receives, as a challenge, the description of a group G of order p and (random) elements $u, v = u^s, g$ in G. Then $\mathcal{B}$ chooses system parameters by sampling $g_j = g^{\gamma_j}$ for $\gamma_j \leftarrow \mathbb{Z}_p$. It sets $\mathsf{params} = (p, G, g_1, \ldots, g_m, u)$, $\mathsf{pk} = v$. The secret database $\mathsf{sk} = (\mathsf{sk}[1], \ldots, \mathsf{sk}[n])$ is chosen by setting $(\mathsf{pk}[i], \mathsf{sk}[i]) \leftarrow_R \mathbf{Gen}()$. Lastly, the helper values $\mathsf{help} = (\sigma[1], \ldots, \sigma[n])$ are chosen by programming the random oracle $H(i) := u^{\beta_i}/\mathsf{pk}[i]$ for a random $\beta_i \leftarrow_R \mathbb{Z}_p$ and setting $\sigma[i] := v^{\beta_i} = (H(i)\mathsf{pk}[i])^s$ for $i = 1, \ldots, n$. Then $\mathcal{B}$ gives the parameters params, and the keys $(\mathsf{pk}, \mathsf{sk}, \mathsf{help})$ to $\mathcal{A}$ and runs the rewinding process. Since all values chosen by $\mathcal{B}$ come from the same distribution as in the original process, the event $E_1 \wedge E_2$ occurs with probability $\varepsilon'/2$. If this is the case, then property (II) holds for both received conversations, as well as the honest values $\mathsf{pk}^*, \sigma^*, \mathsf{sk}^*$ (which are known to $\mathcal{B}$) and so

$$\sigma^* = (\mathsf{pk}^* w)^s, \quad \sigma_1^* = (\mathsf{pk}_1^* w)^s, \quad \sigma_2^* = (\mathsf{pk}_2^* w)^s \quad \text{where} \quad w = \prod_{i=1}^{t} H(r_i)^{e^{i-1}}.$$

This implies

$$\frac{(\sigma_1^*)^{c_1^*}}{(\sigma_2^*)^{c_2^*}} = \left(w^{c_1^* - c_2^*} \frac{(\mathsf{pk}_1^*)^{c_1^*}}{(\mathsf{pk}_2^*)^{c_2^*}}\right)^s, \quad (\sigma^*)^{c_1^* - c_2^*} = \left(w^{c_1^* - c_2^*} (\mathsf{pk}^*)^{c_1^* - c_2^*}\right)^s.$$

Dividing the two equations by each other, the adversary $\mathcal{B}$ produces a value $\hat{\sigma}$ such that:

$$\hat{\sigma} = \left(\left(\frac{(\mathsf{pk}_1^*)^{c_1^*}}{(\mathsf{pk}_2^*)^{c_2^*}}\right)\left(\frac{1}{(\mathsf{pk}^*)^{c_1^*-c_2^*}}\right)\right)^s = \left(\prod_{j=1}^m g_j^{z_j^{(1)}-z_j^{(2)}-x_j^*(c_1^*-c_2^*)}\right)^s$$

where $z_j^{(1)}, z_j^{(2)}, x_j^*$ are all known to $\mathcal{B}$. Now, using the fact that $\mathcal{B}$ knows γ_j such that $g_j = g^{\gamma_j}$, $\mathcal{B}$ can rewrite the above equation as $\hat{\sigma} = (g^\omega)^s$ for some known value ω. Moreover, the event E_2 implies that $\omega \neq 0$. Therefore, $\mathcal{B}$ can efficiently compute $(\hat{\sigma})^{1/\omega} = g^s$ and thus solves the CDH problem with probability at least $\varepsilon'/2$.

Case B: We show how to use $\mathcal{A}$ to construct an adversary $\mathcal{B}$, which, on input $(p, G, g_1, \dots, g_m)$ computes $(\mathsf{pk}^*, \mathsf{sk}^*, \hat{\mathsf{sk}}^*)$ such that $(\mathsf{pk}^*, \mathsf{sk}^*), (\mathsf{pk}^*, \hat{\mathsf{sk}}^*) \in \mathcal{R}$ and $\mathsf{sk}^* \neq \hat{\mathsf{sk}}^*$. As shown in Lemma 4.1, this is equivalent to solving DL (and hence CDH). The adversary $\mathcal{B}$ chooses an element $u \leftarrow_R G$ and sets $\mathsf{params} = (p, G, g_1, \dots, g_m, u)$. It then runs the rewinding process with the parameters params and following everything else as specified: (1) choosing $(\mathsf{pk}, \mathsf{sk}, \mathsf{help}) \leftarrow \mathsf{KeyGen}()$, (2) running as $\mathcal{P}(\mathsf{pk}, \mathsf{sk})$ and as $\mathcal{O}_{\mathsf{sk}}^{\lambda,\ell}(\cdot)$ with $\mathcal{A}$, (3) rewinding to get two conversations with different challenges c_1^*, c_2^*. Assume that the event $E_1 \wedge \neg E_2$ occurs. Then $(\mathsf{pk}^*)^{c_1^*-c_2^*} = \prod_{j=1}^m g_j^{z_j^{(1)}-z_j^{(2)}}$ by equation (2). Therefore, $\mathcal{B}$ can efficiently compute the values $\hat{x}_j^* \stackrel{\text{def}}{=} (z_j^{(1)} - z_j^{(2)})/(c_1^* - c_2^*)$ and the witness $\hat{\mathsf{sk}}^* = (\hat{x}_1^*, \dots, \hat{x}_m^*)$ for pk^*. We call this the *extracted* witness. However, $\mathcal{B}$ also has the *honest witness* sk^* for pk^*, which it can compute from sk. Let E_3 be the event that $\hat{\mathsf{sk}}^* = \mathsf{sk}^*$.

Claim B.2. *The probability of E_3 is at most* $\Pr[E_3] \leq 2^{-\Omega(\lambda)}$.

Proof. Let $\mathcal{E}_1$ be the experiment in which a predictor is given the value $\mathsf{PK}, \mathsf{HELP}$ and oracle access to $\mathcal{O}_{\mathsf{SK}}^{\lambda,\ell}(\cdot)$, $\mathcal{P}(\mathsf{PK}, \mathsf{SK})$. Then (reusing the same analysis from Claim 4.2) we get $\widetilde{\mathbf{H}}_\infty(\mathsf{SK} \mid \mathcal{E}_1) \geq \widetilde{\mathbf{H}}_\infty(\mathsf{SK} \mid \mathsf{PK}) - \ell \geq n(m-1)\log(p) - \ell$. We can think of the rewinding process as an experiment $\mathcal{E}_2$, where the predictor first runs $\mathcal{E}_1$, and then is given a random value R over $\left([n]^t \times \mathbb{Z}_p\right)$ (i.e. a random variable for the values $(r_1, \dots, r_t, e)$). We want to show that it is unlikely that a predictor can outputs SK^* in this experiment, where SK^* is the compressed key. Here we use one of our entropy preservation analysis from Appendix A, where we analyze exactly the compression function called $\mathsf{CompDirProd}$ such that $\mathsf{SK}^* = \mathsf{CompDirProd}(\mathsf{SK}, R)$. In particular, Lemma A.5 shows that $\widetilde{\mathbf{H}}_\infty(\mathsf{SK} \mid \mathcal{E}_1) \geq n(m-1)\log(p) - \ell$ implies that $\widetilde{\mathbf{H}}_\infty(SK^* \mid \mathcal{E}_2) \geq \min(\lambda, \frac{\log(p)-\log(t)-1}{2}) = \Omega(\lambda)$ when $\ell \leq (1-\delta)nm\log(p)$. Therefore $\Pr[E_2] \leq 2^{-\widetilde{\mathbf{H}}_\infty(\mathsf{SK}^* \mid \mathcal{E}_2)} \leq 2^{-\Omega(\lambda)}$.

□

Therefore, the probability of $E_1 \wedge \neg E_2 \wedge \neg E_3$ is at least $\varepsilon' - 2^{-\Omega(\lambda)}$ and, with this probability, the adversary $\mathcal{B}$ solves the discrete logarithm problem. □

B.6 Theorem 5.1

Proof of Theorem 5.1. We begin by proving the theorem for the pre-impersonation leakage case. Given an s-entropic forger $\mathcal{A} = (\mathcal{A}_1, \mathcal{A}_2)$ which has advantage ε in $\mathsf{EUG}_\ell^\lambda$ we construct an ID scheme attacker $\mathcal{B}$ which has probability polynomial in ε at breaking the security of the ID scheme.

We now show that the probability that $\mathcal{B}$ convinces verifier $\mathcal{V}$ during the Impersonation Stage is polynomially related to the advantage of $\mathcal{A}$. For an execution of $\mathcal{A}$ let S be the event that $\mathcal{A}$ outputs a signature $(\hat{a}, \hat{z})$ of a new message m. Then by definition, for a random real world execution we have that $\Pr[S] = \varepsilon$ is exactly the advantage of $\mathcal{A}$ in the signature attack game.

Reduction $\mathcal{B}$

The attacker $\mathcal{B}$ gets input $(\mathsf{params}, \mathsf{pk}, \mathsf{help})$ and oracle access to the leakage oracle $\mathcal{O}_{\mathsf{sk}}^{\lambda,\ell}(\cdot)$ and to a prover $\mathcal{P}(\mathsf{pk}, \mathsf{help}, \mathsf{sk})$. Its goal is to win the $\mathsf{IDPRE}_\ell^\lambda(\mathcal{B})$ attack game (see Definition 4.1).

1. Let q be an upper bound on the number Random Oracle queries made by $\mathcal{A}$. $\mathcal{B}$ selects a random index $\rho \leftarrow [q]$. $\mathcal{B}$ interacts with $\mathcal{P}(\mathsf{pk}, \mathsf{help}, \mathsf{sk})$ using uniformly random challenges c_i to obtain a larger number of conversations of the form (a_i, c_i, z_i). After this point there is no more interaction with $\mathcal{P}$. $\mathcal{B}$ initializes $\mathcal{A}_1$ with input $(\mathsf{params}, \mathsf{pk}, \mathsf{help})$. Eventually $\mathcal{B}$ also passes the hint v from $\mathcal{A}_1$ to $\mathcal{A}_2$.
2. $\mathcal{B}$ simulates the oracles $\mathcal{S}_{\mathsf{sk}}(\cdot), \mathcal{O}_{\mathsf{sk}}^{\lambda,\ell}(\cdot)$ as well as the Random Oracle H for $\mathcal{A}$ as follows:

   Random Oracle Queries (expect query ρ): For each Random Oracle query x if $H(x)$ has already been defined, output it. Otherwise interpret $x = (a, m)$ and see if list of conversations includes one of the form (a, c_j, z_j). If so then define $H(x) := c_j$, associate the conversation (a, c_j, z_j) with x and remove it from the list. Output c_j. If no conversation matches a then select a random response c, define $H(x) := c$ and output c.

   Signature Queries: On input m take the next conversation (a, c, z) from the list and check if $H(x = (a, m))$ is defined. If so there must be a conversation (a, c', z') associated with x. Respond with signature (a, z'). Otherwise define $H(x) := c$, associate conversation (a, c, z) with x, remove (a, c, z) from the list and output signature (a, z).

   Random Oracle Query ρ: When the ρ-th Random Oracle query $x = (a, m)$ is made, if $H(x)$ has already been defined then $\mathcal{B}$ outputs **abort-1** and terminates. Otherwise $\mathcal{B}$ enters the impersonation stage of ID attack game $\mathsf{IDPRE}_\ell^\lambda$ and loses access to the leakage oracle. It starts a fresh interaction with the honest verifier $\mathcal{V}(\mathsf{pk})$. $\mathcal{B}$ interprets x as $x = (a, m)$ and sends a to $\mathcal{V}(\mathsf{pk})$ receiving c in response which it outputs as the response to it's ρ-th query.

   Leakage Queries Queries: If a leakage query is made after the ρ-th Random Oracle query then $\mathcal{B}$ outputs **abort-2** and terminates. Otherwise it forwards the leakage query to it's leakage oracle and outputs the response.
3. Eventually $\mathcal{A}_2$ outputs a message m and signature (a, z). If the signature is valid and (a, m) was the ρ-th Random Oracle query then $\mathcal{B}$ sends z to $\mathcal{V}(\mathsf{pk})$ and terminates.

Figure 5: Reduction from Sig attacker to ID attacker.

For an execution of $\mathcal{A}$ let S_1 (occurring with probability ε_1) be the event that S occurs and, at some point, $\mathcal{A}$ queries the random oracle on $\hat{x} = (\hat{a}, m)$. Suppose event S occurs but event S_1 does *not*, then with overwhelming probability the conversation $(\hat{a}, H(\hat{x}), \hat{z})$ is not an accepting conversation since otherwise $\mathcal{A}$ could be used to break the soundness of the ID scheme. (In particular $\mathcal{A}$, on input only the public key, could generate a valid third flow message for a first flow of it's choosing without even seeing the challenge from the verifier.) Therefore we have that $\varepsilon_1 \leq \varepsilon - \mathrm{negl}$ for some negligible function negl in λ.

In an execution where S_1 occurs let ρ' be the index of the first Random Oracle query of the form $(\cdot, m)$. Consider an execution of $\mathcal{A}$ emulated exactly as in the real world except that $\mathcal{B}$ first guesses $\rho \leftarrow [q]$. Let S_2 (occurring with probability ε_2) be the event that S_1 occurs and $\mathcal{B}$ guesses $\rho = \rho'$. Since the view of $\mathcal{A}$ is independent of the value of ρ which is chosen uniformly we have that $\varepsilon_2 = \varepsilon_1/q = (\varepsilon - \mathrm{negl})/q$.

Now consider an execution of $\mathcal{A}$ emulated by $\mathcal{B}$ as described in Figure 5. Let S_3 (occurring with probability ε_3) be the event that S_2 occurs and $\mathcal{B}$ does not output *abort-2* and let S_3' be the event that S_2 occurs but $\mathcal{B}$ *does* output abort-2. Then $\Pr[S_2] = \Pr[S_2 \wedge S_3] + \Pr[S_2 \wedge S_3'] = \Pr[S_3] + \Pr[S_3']$. If event S_3 occurs then the ρ-th query must have been made by $\mathcal{A}_1$ since $\mathcal{A}_2$ can not make leakage queries. Because $\mathcal{A}$ is an s-entropic adversary this can happen with at most probability 2^{-s}. Thus we have that we have $\varepsilon_2 \leq \varepsilon_3 + 2^{-s}$.

Further we argue that in such an emulation, conditioned on event S_3 occurring, the view of an adversary $\mathcal{A}$ is identically distributed to a real world execution. This follows from two observations. First, all Random Oracle queries in the emulation are answered consistently with uniformly random and independent responses. Further these are also consistent with the responses to the signature oracle. The second observation is that the responses to $\mathcal{A}$'s signature queries have the same distribution as in a real world execution. In particular the values of a and z used in the signatures are selected independently of the message and according to the honest provers algorithm for a fresh random challenge.

Therefore we can conclude that $\mathcal{A}$ behaves exactly as in real world executions which implies $\varepsilon_3 \geq \varepsilon_2 - 2^{-s} = \varepsilon_1/q - 2^{-s} = (\varepsilon - \text{negl})q - 2^{-s}$ which is polynomial related to ε for $s \geq \lambda$. Note that emulated executions where event S_3 occurs correspond exactly to the executions of $\mathcal{B}$ when it convinces $\mathcal{V}$ during the Impersonation Stage of the ID attack game. Thus the proof for security with pre-impersonation leakage is complete.

The only difference in the proof of security with any-time leakage is that in the reduction $\mathcal{B}$ we remove the condition for outputting *abort-2*. That is leakage queries are always forwarded to the leakage oracle regardless of when they occur. (Recall that in the attack game for any-time leakage $\mathcal{B}$ retains access to the leakage oracle even during the Impersonation Stage.) Therefore the only change to the above analysis is that the probability of event S_3' is 0 and so we can bound $\varepsilon_3 \geq \varepsilon_1/q$ which is also polynomial in ε. □

C Authenticated Key Agreement

In this section we give a definition for authenticated key agreements secure against side channel leakage which is derived from the notion of SK-security of Canetti and Krawczyk in [CK01] by simplifying the presentation of the model for our needs while augmenting the adversaries capabilities with calls to an appropriate side channel oracle which can be evaluated on the long-term secret of players. Next we give a construction eSIG-DH and prove it secure.

Communication Model & Sessions: We prove security in the unauthenticated-links (UM) model of [CK01]. That is we consider an internet like setting with PKI setup; an asynchronous network of n players $\{\mathcal{P}_1, \ldots, \mathcal{P}_n\}$ with multiple concurrent sessions where players are message driven probabilistic Turing Machines. The PKI is a public file associating players with keys.[4] Each time a player participates in a new instance of a key agreement (KA) protocol it creates a new session of which it is the owner. Each such session is associated with a unique session ID sid and a peer (the player with whom the owner intends to agree on a key). For a given session the owner is assigned one of two roles: initiator or responder. An important notion for defining security of KA protocols is that of a matching sessions.[5] A pair of sessions S_0 and S_1 are called matching if:

1. The sid are equal.
2. The role of the owner of S_0 is different from the role of the owner of S_1.
3. The peer of the owner of S_0 is the owner of S_1 and vice versa.

Following [Kra05], remaining consistent with [CK01], we make the following two simplifying assumptions.

[4] In particular this is the weakest form of PKI where no properties of the keys including knowledge of the secret key are verified by the registration server.

[5] Intuitively, the security of a session key derived during a particular KA protocol can not be guaranteed for one player if the other has been corrupted.

1. Sessions are initiated at a player by an unspecified higher level protocol which provides the intended peer and role. (This is called the "pre-specified peer model" in [CK02])
2. sid's will take the form $(\hat{A}, \hat{B}, Out, In)$ where $\hat{A}$ is the owner's identifier, $\hat{B}$ is the peer's identifier, Out are the outgoing messages and In the incoming messages.

A session can be marked as complete at which point the owner outputs the session-key and deletes all session specific state from memory. Finally, a complete session can be marked as expired which means that the session-key is deleted as well. Intuitively, if security for the session key is guaranteed after it has expired even if the long-term secret was subsequently given to the attacker then this captures *perfect forward secrecy* (*PFS*).

Adversarial Model and Exposures: The adversary in the UM is a classic "man-in-the-middle" adversary. More precisely a KA-adversary $\mathcal{A}$ is an active adversary with full control over the routing and scheduling of messages. $\mathcal{A}$ initiates all session's and may at any point launch a session exposure attack against a specified session which is then called exposed. A session exposure attack may be any one of the following types:

state-reveal query: These are attacks are launched against incomplete sessions and result in the adversary learning all session specific internal state of the owner. This does *not* include long-term secrets.

session key query: These attacks are launched against complete sessions and result in the adversary learning the value of the owner's seesion-key.

corrupt player: These attacks are launched against a specified player and result in the adversary learning all secrets of that player. In particular $\mathcal{A}$ learns all session-keys of unexpired sessions, internal states of incomplete sessions, and long-term secrets. Further the owner is called corrupted, all future actions of that player are assumed to be under the control of the adversary and all unexpired sessions of that owner are also exposed.

Leakage: We now diverge from the original model of [CK01] and introduce a fourth type of attack modeled by giving the adversary access a the leakage oracle for each player. In particular an ℓ-KA-adversary $\mathcal{A}$ is a KA-adversary such that for every player P with secret key sk, $\mathcal{A}$ has access to the leakage oracle $\mathcal{O}_{\mathsf{sk}}^{\lambda,\ell}(\cdot)$. $\mathcal{A}$ can then launch the following additional attack:

leak query: At any time, for any player P where P has secret key sk, $\mathcal{A}$ can submit a query $\mathcal{O}_{\mathsf{sk}}^{\lambda,\ell}(\cdot)$. Any currently incomplete sessions where owner $= P$ are subsequently marked as exposed.

Secure KA Protocols Analogously to [CK01] we can now define security for KA protocols as the inability of any efficient ℓ-KA-adversary $\mathcal{A}$ to distinguish the session-key from a random key for any complete test-session of it's choosing amongst all complete unexposed unexpired sessions.

At any point during it's execution $\mathcal{A}$ can select a test-session (i.e. the challenge session) from amongst all currently complete but unexposed and unexpired sessions. The challenger selects a random bit b and if $b = 0$ it hands the session-key v_0 to $\mathcal{A}$. Otherwise it selects a fresh random and independent session-key v_1 from the same distribution as that generated by the protocol and hands this to $\mathcal{A}$. After receiving v_b, $\mathcal{A}$ continues execution (with the restriction that it can not expose the test-session) until it finally outputs a guess bit b'. We call the quantity $\left|\Pr[b = b'] - \frac{1}{2}\right|$ the advantage of $\mathcal{A}$ and call as test-session winning if $b = b'$.

Definition C.1. *Let $\mathcal{A}$ be an efficient ℓ-KA-adversary. A KA protocol π in the UM model is called* ℓ-SK-Secure with PFS *if for all such $\mathcal{A}$ running against π we have that:*

Completeness: *If two uncorrupted parties complete* matching sessions *then they both output the same key.*

Privacy: *The advantage of $\mathcal{A}$ is a negligible function in λ.*

C.1 Construction

We now give the protocol eSIG-DH in figure Figure 6 constructed from an entropically-unforgeable signature scheme $\mathsf{Sig} = (\mathsf{KeyGen}, \mathsf{Sign}, \mathsf{Verify})$ with leakage ℓ and Diffie-Hellman key agreement (along the lines of the SIG-DH protocol in [CK01]).

Public Parameters: $\mathbb{G} = (g, G, p)$ a DDH group with generator g and order p.
Common Input: signature verification keys $(\mathsf{verk}_i, \mathsf{verk}_j)$

Initiator $\mathcal{P}_i(\mathsf{sigk}_i)$		Responder $\mathcal{P}_j(\mathsf{sigk}_j)$
$a \leftarrow_R \mathbb{Z}_p$, $\alpha = g^a$ register session $(\mathcal{P}_j, \alpha)$	$\xrightarrow{\mathcal{P}_i, \alpha}$	
	$\xleftarrow{\mathcal{P}_j, \beta, \sigma_j}$	$b \leftarrow_R \mathbb{Z}_p$, $\beta = g^b$ $\sigma_j = \mathsf{Sign}_{\mathsf{sigk}_j}(\mathcal{P}_i, \mathcal{P}_j, \alpha, \beta)$ register session $(\mathcal{P}_i, \alpha, \beta)$
$\sigma_i = \mathsf{Sign}_{\mathsf{sigk}_i}(\mathcal{P}_j, \mathcal{P}_i, \alpha, \beta)$ output $\mathsf{peer} = \mathcal{P}_j$, $\mathsf{sid} = (\alpha, \beta)$, output session key $\gamma_i = \beta^a$ delete a	$\xrightarrow{\mathcal{P}_i, \sigma_i}$	
mark session complete		output peer $=\mathcal{P}_j$, sid $=(\alpha, \beta)$ output session key $\gamma_i = \beta^a$ delete b mark session complete

Sanity Checks:

1. If σ_j is not a valid signature of $(\mathcal{P}_i, \mathcal{P}_j, \alpha, \beta)$, under verk_j then $\mathcal{P}_i$ ignores the message. Similarly for σ_i and $\mathcal{P}_j$.
2. If $\mathcal{P}_i$ receives a round-2 message σ_i of $(\mathcal{P}_i, \mathcal{P}_j, \alpha, \beta)$ but has not registered a session $(\mathcal{P}_j, \alpha)$ then ignore the message. Similarly $\mathcal{P}_j$ and σ_j.

Figure 6: Protocol eSIG-DH

For completeness we restate the Theorem 6.1 bellow.

Theorem (6.1). *Let* Sig *be an entropically-unforgeable signature scheme with leakage ℓ, then* eSIG-DH *is an ℓ-*SK-secure KA *protocol with PFS in the* UM *under the* DDH *assumption.*

Proof of Theorem 6.1. To see that eSIG-DH is complete consider a pair of uncorrupted parties $\mathcal{P}_i$ and $\mathcal{P}_j$ which have completed matching sessions. As the sessions match they have the same $\mathsf{sid} = (\alpha, \beta)$. Therefore the session keys were computed as $\gamma_i = \beta^a$ by $\mathcal{P}_i$ and $\alpha^b = \gamma_j$ by $\mathcal{P}_j$. Thus we have that $\gamma_i = \gamma_j$.

The proof that eSIG-DH is private is more involved. Assume that the ℓ-KA-adversary $\mathcal{A}$ can break privacy of eSIG-DH with non-negligible probability. $\mathcal{A}$ can be one of two types of adversary. In the following, for an execution of $\mathcal{A}$ against a set of players, we call a signature σ under player $\mathcal{P}$'s key *new* if $\mathcal{P}$ has not been corrupted and has never produced a signature of m.

T1: There is a non-negligible probability that in a winning test-session the owner receives a new signature for σ_i or σ_j.

T2: There is negligible probability that in a winning test-session the owner receives a new signature.

In the proof of Claim C.1 we describe efficient reductions $\mathcal{B}_1$ from adversaries of type T1 to the entropic-unforgeability of Sig. In the proof of Claim C.2 we give an efficient reduction $\mathcal{B}_2$ from a type T2 adversary which breaks the DDH. Thus a reduction $\mathcal{B}$ which selects uniformly between each of these three strategies can either break the signature scheme or break the DDH proving theorem Theorem 6.1.

It remains to show the two claims.

Claim C.1. *There is a efficient reduction from a type T1 attacker to the entropic-unforgeability of* Sig *with leakage ℓ.*

Proof. Let $\mathcal{A}$ be a type T1 attacker and l be an upper bound on the number of sessions started by $\mathcal{A}$. We construct a reduction $\mathcal{B}_1 = (\mathcal{C}_1, \mathcal{C}_2)$ to the entropic-unforgeability of Sig.

$\mathcal{B}_1$ emulates a run of $\mathcal{A}$ against n players internally in two phases. $\mathcal{B}_1$ begins with $\mathcal{C}_1$ which guesses which session will be the test-session in the emulation of $\mathcal{A}$. Once it receives α for that session it's passes the current state of the emulation to the second half $\mathcal{C}_2$. $\mathcal{C}_2$ then selects a random challenge α (or β) and continues the emulation until σ_i (or σ_j) has been sent for that session. We describe the details of $\mathcal{B}_1$ strategy in Figure 7.

We argue that for a type T1 attacker $\mathcal{A}$, the probability of $\mathcal{B}_1$ forging is polynomially related to the probability ε that $\mathcal{A}$ forges a signature during eSIG-DH. Further let δ be the probability that in a winning test-session $\mathcal{A}$ delivers a signature of a new message to the owner. For an execution of $\mathcal{A}$ let event S_1 (occurring with probability ε_1) be the event that $\mathcal{A}$ that the execution is winning and that $\mathcal{A}$ delivers a signature of a new message to the owner of the test-session. By definition, for a real world execution we have that $\varepsilon_1 \geq \varepsilon\delta$.

Now consider an execution of $\mathcal{A}$ emulated by $\mathcal{B}_1$ completely honestly (i.e. exactly as in the real world) with the exception that before starting $\mathcal{B}_1$ selects a random $r \leftarrow [l]$ and a random player index $i \in [n]$. Let the even S_2 (occurring with probability ε_2) that S_1 occurs and that r-th session is the test-session and that the peer for that session is $\mathcal{P}_i$. Since the view of $\mathcal{A}$ is independent of both r and i we have that $\varepsilon_2 = \frac{\varepsilon_1}{ln}$.

Consider an execution of $\mathcal{A}$ emulated by $\mathcal{B}_1$ as described in Figure 7. We note that the view of $\mathcal{A}$ in such an emulation is indistinguishable to it's view in a real world execution as the signing and leakage oracles are used to answer any queries pertaining to sigk and all other operations are performed exactly as in the real world. In particular the value of $\hat{\beta}$ is computed (by $\mathcal{C}_2$) exactly as would be done by $\mathcal{P}_i$. Therefore $\mathcal{A}$ will behave exactly as in a real world execution and we have that $\Pr[S_2] = \varepsilon_2$ also for the emulation.

For such an emulation let event S_3 (occurring with probability ε_3) be the probability that S_2 occurs and that $\mathcal{C}_2$ does not performs a leak query against $\mathcal{P}$. By definition, for event S_3 to occur so must event S_1 which implies that the execution was winning. By definition of a ℓ-SK-security this implies that no leak query was launched against $\mathcal{P}$ after the test-session was initialized and before it was marked complete. However $\mathcal{C}_2$ is run by $\mathcal{B}_1$ exactly in this window and so by assumption $\mathcal{C}_2$ could not have made such a query. Therefore we have that $\Pr[S_3] = \varepsilon_2$.

Note that emulations where S_3 occurs are exactly the emulations where $\mathcal{B}_1$ wins the signature attack game $\mathsf{EUG}_\ell^\lambda$. Further we argue that $\mathcal{B}_1$ is an λ-entropic adversary. If the r-th session is marked complete it must be that the signature supplied to the owner by $\mathcal{A}$ was accepted. If $\mathcal{P}$ is the initiator

<u>Reduction $\mathcal{B}_1 = (\mathcal{C}_1, \mathcal{C}_2)$</u>

1. Forger $\mathcal{C}_1^{\mathcal{S}_{\mathsf{sigk}}(\cdot), \mathcal{O}_{\mathsf{sk}}^{\lambda,\ell}(\cdot)}$ takes input verk and has access to the signature and leakage oracles. It select a random player $\mathcal{P} \in \{\mathcal{P}_1, \ldots, \mathcal{P}_n\}$ and a random session $r \leftarrow [1, l]$ (guessing that this will be the test-session and $\mathcal{P}$ will be the peer).
2. Forger $\mathcal{C}_1$ runs $\mathcal{A}$ internally against honest players $\{\mathcal{P}_1, \ldots, \mathcal{P}_n\}$ with the caveat that rather then generating a fresh signature key pair for $\mathcal{P}$, $\mathcal{C}_1$ uses verk as $\mathcal{P}$'s public key. It signs all signatures of $\mathcal{P}$ are with calls to $\mathcal{S}_{\mathsf{sigk}}(\cdot)$ and any leak query is answered with $\mathcal{O}_{\mathsf{sigk}}^{\ell,\lambda}(\cdot)$. Besides that $\mathcal{P}$ follows the honest algorithm until the r-rth session. Once the r-th session is initialized there are two cases depending on the role of $\mathcal{P}$:

   $\mathcal{P}$ is the initiator: In this case $\mathcal{B}$ will attempt to forge using σ_i.

   (a) Once the round-1 message $(\mathcal{P}_i = \mathcal{P}, \hat{\alpha})$ has been received by $\mathcal{P}_j$ $\mathcal{C}_1$ outputs the entire state of the current emulation encoded in string v. This is given to $\mathcal{C}_2^{\mathcal{S}_{\mathsf{sigk}}(\cdot)}$ which gets a fresh random tape r from the $\mathsf{EUG}_\ell^\lambda$ challenger and resumes the emulation at the same point where $\mathcal{C}_1$ left off. It replaces the random tape of $\mathcal{P}_j$ with r and continues the emulation faithfully. (Note that $\hat{\beta}$ the value of β chosen by $\mathcal{P}_j$ for the round-2 message of the r-th session depends only on r.) Signature queries to $\mathcal{P}$ are answered with calls to the signing oracle.

   (b) Once the round-3 message $(\mathcal{P}_i, \sigma_i)$ is received return signature σ_i and message $(\mathcal{P}_j, \mathcal{P}_i, \hat{\alpha}, \hat{\beta})$ to the challenger in $\mathsf{EUG}_\ell^\lambda(\mathcal{B}_1)$ and terminate.

   $\mathcal{P}$ is the responder: In this case $\mathcal{B}$ will attempt to forge using σ_j

   (a) Once $\mathcal{A}$ instructs $\mathcal{P}_i$ to initialize the r-th session $\mathcal{C}_1$ outputs the current state of the emulation encoded in the string v. This is given to $\mathcal{C}_2^{\mathcal{S}_{\mathsf{sigk}}(\cdot)}$ which gets a fresh random tape r from the $\mathsf{EUG}_\ell^\lambda$ challenger and resumes the emulation at the same point where $\mathcal{C}_1$ left off. It replaces the random tape of $\mathcal{P}_j$ with r and continues the emulation faithfully. (Note that $\hat{\alpha}$ the value of β chosen by $\mathcal{P}_j$ for the round-2 message of the r-th session depends only on r.) Signature queries to $\mathcal{P}$ are answered with calls to the signing oracle.

   (b) Once the round-2 message $(\mathcal{P}_i, \sigma_i)$ is received return signature σ_i and message $(\mathcal{P}_j, \mathcal{P}_i, \hat{\alpha}, \hat{\beta})$ to the $\mathsf{EUG}_\ell^\lambda$ challenger and terminate.

<u>Abort Conditions</u>

1. If $\mathcal{A}$ terminates during the emulation before completing the r-th session then $\mathcal{B}_1$ aborts.
2. If the r-th session is not the test-session then $\mathcal{B}_1$ aborts.
3. If for the r-th session the $\mathcal{P}$ is not the peer then abort.
4. If a leak query is made by $\mathcal{A}$ against $\mathcal{P}$ during the part of the emulation run by $\mathcal{C}_2$ then $\mathcal{B}_1$ aborts.

Figure 7: Forger $\mathcal{B}_1 = (\mathcal{C}_1, \mathcal{C}_2)$

then the message $\mathsf{m} = (\mathcal{P}_i, \mathcal{P}_j = \mathcal{P}, \hat{\alpha}, \beta')$ outputted by $\mathcal{C}_2$ is such that $\beta' = \hat{\beta}$. Further $\hat{\beta}$ was chosen by $\mathcal{C}_2$ independently of the view of $\mathcal{C}_1$. An analogous statement holds for α' in the m outputted by $\mathcal{C}_2$ when $\mathcal{P}$ is the responder. For security parameter λ we have that $\log(p) \geq \lambda$ and so $\widetilde{\mathbf{H}}_\infty(\mathsf{MSG}_{\mathcal{C}_2} | \mathsf{View}_{\mathcal{C}_1}) \geq \lambda$ making $\mathcal{B}_1$ a λ-entropic adversary. Therefore it's $\mathcal{B}_1$'s advantage is at least $\frac{\varepsilon\delta}{ln}$ which, by assumption of $\mathcal{A}$ being a type T1 attacker is a polynomial in ε.

This completes the proof of Claim C.1 □

Claim C.2. *There is a efficient reduction from a type T3 attacker to* DDH.

Proof. To prove the claim we consider two subcases. Either the test-session is selected

We construct reduction $\mathcal{B}_2$ (outlined in Figure 8 which has input the tuple $G = (\mathbb{G}, \alpha^*, \beta^*, \gamma^*)$. The idea is that $\mathcal{B}_2$ guess the test-session and force α^* and β^* to be played. That way the adversary has a non-negligible advantage at winning the session if and only if the input was a DDH tuple.

Reduction $\mathcal{B}_2(G, \mathcal{A})$

1. Select a random a random session $r \leftarrow [1, l]$.
2. Run $\mathcal{A}$ internally against honest players $\{\mathcal{P}_1, \ldots, \mathcal{P}_n\}$.
3. If $\mathcal{A}$ terminates before finishing the r-th session then abort.
4. In round 1 of the r-th session set $\alpha = \alpha^*$ and in round 2 set $\beta = \beta^*$.
5. If $\mathcal{A}$ does not select the r-th session as the test-session then abort. Otherwise hand γ^* to $\mathcal{A}$ as the challenge key.
6. When $\mathcal{A}$ responds with $b' = 0$ then output "True" otherwise output "False".

Figure 8: Reduction $\mathcal{B}_2$ to breaking the DDH.

The analysis is relatively straightforward. Let ε be the probability that a real world execution of $\mathcal{A}$ is winning. Further let negl be a negligible function in λ representing the probability that in a random execution, $\mathcal{A}$ sends a signature of a new message to the owner of the test-session. Let $r \leftarrow [q]$ be selected uniformly independently. For an execution of $\mathcal{A}$ let event S_1 (occurring with probability ε_1) be the event that the r-th session is the test-session. Then by definition $\varepsilon_1 = l^{-1}$.

Now consider an emulation of $\mathcal{A}$ run by $\mathcal{B}_2$ as described in Figure 8. Note that until the challenge key for the test-session is given to $\mathcal{A}$ it's view remains independent of the value of r. Therefore also for the emulation $\Pr[S_1] = \varepsilon_1$.

Let event S_2 (occurring with probability ε_2) be event that S_1 occurs and that the execution would be winning if $\mathcal{A}$ were given the real session key with probability 1/2. Then we have that $\varepsilon_2 = \varepsilon_1 \varepsilon$. Further let S_3 be the event that a new signature was generated by $\mathcal{A}$ for the test-session. Then we have that $\Pr[S_2] = \Pr[S_2 \wedge S_3] + \Pr[S_2 \wedge \neg S_3]$. We can bound $\Pr[S_2 \wedge S_3] = \Pr[S_3] \Pr[S_2 | S_3] \leq \text{negl}$ and so $\Pr[S_4 = (S_2 \wedge \neg S_3)] \geq \varepsilon_2 - \text{negl} = (\varepsilon l^{-1}) - \text{negl}$.

Notice that conditioned on event S_4 occurring we have that sid $= (\hat{\alpha}, \hat{\beta})$ and so if G is a DDH tuple then γ^* is distributed exactly as the session key for the test-session would be. Otherwise γ^* is a random element from the set of all such keys. Therefore the output bit of $\mathcal{A}$ on input γ^* is identically distributed to a real world execution where event S_4 has occurred. But emulations of $\mathcal{B}_2$ where S_4 occurs are exactly the ones where $\mathcal{B}_2$ breaks DDH. There for $\mathcal{B}_2$ has advantage (εl^{-1})negl which is polynomial in ε. $\square$

This concludes the proof of Theorem 6.1. $\square$