Leakage-Resilient Cryptography from the Inner-Product Extractor

Stefan Dziembowski[1,⋆] and Sebastian Faust[2,⋆⋆]

[1] University of Warsaw and Sapienza University of Rome
[2] Aarhus University

Abstract. We present a generic method to secure various widely-used cryptosystems against *arbitrary* side-channel leakage, as long as the leakage adheres three restrictions: first, it is bounded per observation but in total can be arbitrary large. Second, memory parts leak *independently*, and, third, the randomness that is used for certain operations comes from a simple (non-uniform) distribution.

As a fundamental building block, we construct a scheme to store a cryptographic secret such that it remains *information theoretically* hidden, even given arbitrary continuous leakage from the storage. To this end, we use a randomized encoding and develop a method to securely *refresh* these encodings even in the presence of leakage. We then show that our encoding scheme exhibits an efficient additive homomorphism which can be used to protect important cryptographic tasks such as identification, signing and encryption. More precisely, we propose *efficient* implementations of the Okamoto identification scheme, and of an ElGamal-based cryptosystem with security against continuous leakage, as long as the leakage adheres the above mentioned restrictions. We prove security of the Okamoto scheme under the DL assumption and *CCA2 security* of our encryption scheme under the DDH assumption.

1 Introduction

In the last years, a large body of work attempts to analyze the effectiveness of side-channel countermeasures in a mathematically rigorous way. These works propose a physical model incorporating a (mostly broad) class of side-channel attacks and design new cryptographic schemes that provably withstand them under certain assumptions about the physical hardware (see, e.g., [24,11,12,16,9,5,23] and many more). By now we have seen new constructions for many important cryptographic primitives such as digital signature and public key encryption

⋆ The European Research Council has provided financial support to the first author of this paper under the European Community's Seventh Framework Programme (FP7/2007-2013) / ERC grant agreement no CNTM-207908.

⋆⋆ Sebastian Faust acknowledges support from the Danish National Research Foundation and The National Science Foundation of China (under the grant 61061130540) for the Sino-Danish Center for the Theory of Interactive Computation, within part of this work was performed. Part of this work was done while being at KU Leuven.

schemes that are provably secure against surprisingly broad classes of leakage attacks.

Unfortunately, most of these new constructions are rather complicated non-standard schemes, often relying on a heavy cryptographic machinery, which makes them less appealing for implementations on computationally limited devices. In this work, we take a different approach: instead of developing new cryptographic schemes, we ask the natural question whether standard, widely-used cryptosystems can be implemented *efficiently* such that they remain secure in the presence of continuous bounded leakage. We answer this question affirmatively, and show a *generic* way that "compiles" various common cryptosystems into schemes that remain secure against a broad class of leakage attacks.

Similar to earlier work, we make certain restrictions on the leakage. We follow the work of Dziembowski and Pietrzak [11], and allow the leakage to be arbitrary as long as the following two restrictions are satisfied:

1. **Bounded leakage:** the amount of leakage in each round is bounded to λ bits (but overall can be arbitrary large).
2. **Independent leakage:** the computation can be structured into rounds, where each such round leaks independently (we define the notion of a "round" below).

Formally, this is modeled by letting the adversary in each round choose a polynomial time computable leakage function f with range $\{0,1\}^\lambda$, and then giving her $f(\tau)$ where τ is all the data that has been accessed during the current round. In addition to these two restrictions, we require that our device has access to a source of correlated randomness generated in a *leak-free* way – e.g., computed by a simple leak free component. We elaborate in the following on our leakage restrictions.

On the bounded leakage assumption. Most recent work on leakage resilient cryptography requires that the leakage is bounded per observation to some fraction of the secret key. This models the observation that in practice many side-channel attacks only exploit a polylogarithmic amount of information, and typically require thousands of observations until the single key can be recovered. This is, for instance, the case for DPA-based attacks where the power consumption is modeled by a weighted sum of the computation's intermediate values. We would like to mention that all our results also remain true in the *entropy loss model*, i.e., we do not necessarily require that the leakage is bounded to λ bits, but rather only need that the min entropy of the state remains sufficiently high even after given the leakage.

On independent leakages. In this paper, we assume that the memory of the device is divided into three parts L, R and C where (L, C) and (R, C) leak independently. To use the independent leakage assumption, we structure the computation into rounds, where each round only accesses either (L, C) or (R, C). Similar assumptions have been used in several works [24,11,27,21,12].

On leak-free components. We require that devices that implement our schemes have access to a source of correlated randomness sampled in a leak-free way. Such a source can, for instance, be implemented by a probabilistic leak-free component that outputs the correlated randomness. Of course, the assumption of a leak-free component is a strong requirement on the hardware, but let us argue why in our particular case it still may be a feasible assumption. As in earlier works that made use of leak-free components [15,13,20,16], we require that our component leaks from its outputs, but the leakage function is oblivious to its internals. To be more concrete, in the simplest case our component $\mathcal{O}$ outputs two random vectors $A, B \leftarrow \mathbb{F}^n$ (with $\mathbb{F}$ being a finite field and n being a statistical security parameter) such that their inner product is 0, i.e., $\sum_i A_i \cdot B_i = 0$. We require that A gets stored on one part of the memory, while B gets stored on the other, thus, we require that A and B leak independently.

Our component $\mathcal{O}$ exhibits several properties that are beneficial for implementations. First, $\mathcal{O}$ is simple and small. It can be implemented in size linear in n, as one simply needs to sample uniformly at random vectors A and $(B_1, \ldots, B_{n-1})$ and computes the last element B_n such that $\sum_i A_i \cdot B_i = 0$.[1] Second, $\mathcal{O}$ is used in a very limited way, namely, it is needed only when the secret key gets refreshed (cf. Section 1.2 for further discussion on this). Finally, $\mathcal{O}$ does not take any inputs, and hence its computation is completely independent of the actual computation (e.g., encryption or signing) that is carried out by the device. This not only allows to test the component independently from the actual cryptoscheme that is implemented, but moreover makes it much harder to attack by side-channel analysis, as successful attacks usually require some choice (or at least knowledge) over the inputs.

1.1 Leakage Resilient Standard Cryptographic Schemes

While in the last years tremendous progress has been made in the design of new cryptographic schemes with built-in leakage resilience, two common criticisms are frequently brought up:

1. Cryptographic schemes are rarely used stand-alone, but more often are part of an industrial standard. Even if desirable, it is unlikely that in the near future these standards will be adjusted to include recent scientific progress.
2. Many of the current leakage resilient cryptoschemes are complicated, rely on non-standard complexity assumptions and are often rather inefficient.

In this work, we are interested in techniques that allow for efficient leakage resilient implementations of widely-used cryptographic schemes. Before we given an overview of our contributions in the next section, we discuss some related literature that considered a similar question.

LEAKAGE RESILIENT CIRCUIT COMPILERS. One fundamental question in leakage resilient cryptography is whether *any* computation can be implemented in a way that resists certain side-channel leakages. This question has been studied in a series of works [19,13,20,16] and dates back to the work of Ishai et al. [19]. In

[1] For simplicity, we assume that L_n is non-zero.

particular, the works of Juma and Vahlis [20] and Goldwasser and Rothblum [16] study the question whether any computation can be implemented in a way that withstands arbitrary polynomial-time computable leakages. As a building block they use a public-key encryption scheme and encrypt the entire computation of the circuit. More precisely, the approach of Juma and Vahlis makes use of fully homomorphic encryption, while Goldwasser and Rothblum generate for each Boolean wire of the circuit a new key pair and encrypt the current value on the wire using the corresponding key. We would like to emphasize that all circuit compilers (except for the one of Ishai et al.) require leak-free components. Notice also that the work of Goldwasser and Rothblum and Juma and Vahlis requires the independent leakage assumption.

Leakage Resilient ElGamal. While circuit compilers allow to secure any (cryptographic) computation against leakage, they typically suffer from a large efficiency overhead. A recent work of Kiltz and Pietrzak [21] makes progress in this direction. The authors show that certain standard cryptographic schemes can be implemented *efficiently* in a leakage resilient way. The main weakness of this work is that the security proof is given in the generic group model.

1.2 Our Contribution

In this paper, we show a *generic* method to implement various standard cryptographic schemes that are provably secure in the above described leakage model. More precisely, we propose an efficient and simple implementation of the Okamoto authentication/signature scheme and of an ElGamal-based encryption scheme, and prove the security of our implementations under continuous leakage attacks. We also discuss why our techniques are fairly general and may find applications for the secure implementation of various other cryptographic schemes. As a fundamental tool, we introduce an information theoretically secure scheme to refresh an encoded secret in the presence of continuous leakage. We detail on our results below.

Leakage Resilient Refreshing of Encoded Secrets. Recently, Davi et al. [8] introduced the notion of leakage resilient storage (LRS). An LRS encodes a secret S such that given partial knowledge about the encoding an adversary does not obtain any knowledge about the encoded secret S. One of their instantiations relies on the inner product two-source extractor introduced in the seminal work of Chor and Goldreich [7]. In this scheme the secret S is encoded as a pair $(L, R) \in \mathbb{F}^n \times \mathbb{F}^n$, where $\mathbb{F}$ is some finite field, and $\langle L, R \rangle := \sum_i L_i \cdot R_i = S$. Unfortunately, the construction of Davi et al. has one important weakness: it can trivially be broken if an adversary continuously leaks from the two parts L and R. The first contribution of this paper is to propose an efficient refreshing scheme for the inner product based encoding.

This is achieved by dividing the memory of the device into three parts L, R and C, where initially (L, R) are chosen uniformly subject to the constraint that $\langle L, R \rangle = S$, and C is empty. Our refreshing scheme Refresh takes as input (L, R) and outputs a fresh encoding (L', R') of S. The computation of Refresh will be structured into several rounds, where in each round we only touch either (L, C) or (R, C), but never L and R at the same time. We will allow the adversary to adaptively leak a bounded amount of information from (L, C) and (R, C). In fact, this is the only assumption we make, i.e., we do not require that the rounds of the computation leak independently. Since in our protocol the third part C is only used to "communicate" information between L and R, we will usually describe our schemes in form of a 2-party protocol: one party, P_{L}, is controlling L, while the second party, P_{R}, holds R. The third part C is used to store messages that are exchanged between the parties. Hence, instead of saying that we allow the adversary to retrieve information from (L, C) and (R, C), we can say that the leakage functions take as inputs all variables that are in the view of P_{L} or P_{R}.

Our protocol for the refreshing uses the following basic idea. Suppose initially P_{L} holds L and P_{R} holds R with $\langle L, R\rangle = S$, then we proceed as follows:

1. P_{L} chooses a vector X that is orthogonal to L, i.e., $\langle L, X\rangle = 0$, and sends it over to P_{R}.
2. P_{R} computes $R' := R + X$ and chooses a vector Y that is orthogonal to R' and sends it over to P_{L}.
3. P_{L} computes $L' := L + Y$.

The output of the protocol is (L', R'). By simple linear algebra it follows that $\langle L, R\rangle = \langle L', R'\rangle = S$. One may hope that the above scheme achieves security in the presence of continuous leakage. Perhaps counterintuitive, we show in the full version of this paper that this simple protocol can be broken if the leakage function can be evaluated on (L, X, Y) and (R, X, Y). To avoid this attack, we introduce a method for P_{L} to send a random X to P_{R} in an "oblivious" way, i.e., without actually learning anything about X, besides the fact that X is orthogonal to L (and symmetrically a similar protocol for P_{R} sending Y to P_{L}). We propose an efficient protocol that achieves this property by making use of our source of correlated randomness $(A, B) \leftarrow \mathcal{O}$. Notice that even given access to such a distribution, the refreshing of an encoded secret is a non-trivial task, as, e.g., just computing $L' = L + A$ and $R' = R + B$ does not preserve the secret.

The protocol that we eventually construct in Figure 1 solves actually a more general problem: we will consider schemes for storing vectors $S \in \mathbb{F}^m$, and the encoding of a secret S will be a random pair (L, R) where L is a vector of length n and R is an $n \times m$-matrix (where $n \gg m$ is some parameter), and $S = L \cdot R$.

Leakage Resilient Authentication and Signatures. We then use our protocol for refreshing an encoded secret as a building block to efficiently implement standard authentication and signature schemes. More concretely, we show that under the DL assumption a simple implementation of the widely-used Okamoto authentication scheme is secure against impersonation attacks even if the prover's computation leaks continuously. Using the standard Fiat-Shamir heuristic, we can turn our protocol into a leakage resilient signature scheme.

At a high level, our transformation of the standard Okamoto scheme encodes the original secret keys with our inner product based encoding scheme. Then, we carry out the computation of the prover in "encoded form", and finally after each execution of the prover, we refresh the encoded secrets using our leakage resilient refreshing scheme. To carry out the computation of the prover in an encoded, we use the following two observations about the inner product based encoding:

1. it exhibits an additive homomorphism, i.e., if we encode two secrets S_1, S_2 as (L, Q) and (L, R), then $(L, Q + R)$ represents an encoding of $S_1 + S_2$. Moreover, if Q and R are stored on the same memory part, then this computation can be carried out in a leakage resilient way.
2. for two secrets S_1 and S_2 and two group generators g_1 and g_2, it allows to compute $g_1^{S_1} \cdot g_2^{S_2}$ in a leakage-resilient way. To illustrate this, suppose that S_1 is encoded by (L, Q) and S_2 is encoded by (L, R). A protocol to compute $g_1^{S_1} \cdot g_2^{S_2}$ proceeds then as follows. P_{R} computes the vector $A := g_1^Q g_2^R = \left(g_1^{Q_1} g_2^{R_1}, \ldots, g_1^{Q_n} g_2^{R_n}\right)$ and sends it over to P_{L}. Next, P_{L} computes the vector $B := A^L = (A_1^{L_1}, \ldots, A_n^{L_n})$ and finally it computes $g_1^{S_1} g_2^{S_2} = \prod_i B_i$.

Together with our scheme for refreshing the inner product encoding, these both basic components suffice to implement the standard Okamoto authentication scheme in a leakage resilient way (cf. Section 4).

Leakage Resilient CCA2-secure encryption. As a third contribution, we show that a simple and efficient variant of the ElGamal cryptosystem can be proven to be CCA2 secure in the RO model even if the computation from the decryption process leaks continuously. We would like to emphasize that we allow the leakage to *depend* on the target ciphertext. We achieve this by exploiting the independent leakage assumption and carry out the computation using the above described protocol for secure exponentiation. We would like to note that even though our scheme uses a simulation sound (SS) NIZK, our construction is rather efficient, as SS-NIZKs can be implemented efficiently via the Fiat-Shamir heuristic. Notice that the Fiat-Shamir heuristic is the only place where the random oracle assumption is used.

A general paradigm for leakage resilient implementations. We observe that our methods for implementing cryptographic schemes is fairly general. Indeed, the two main properties that we require are

1. The secret key of the cryptosystem is an element in a finite field, and the scheme computes only a linear function of the secret keys, and
2. The secret key is hidden information theoretically even given the transcript that an adversary obtains when interacting with the cryptosystem.

Various other cryptosystems satisfy these properties. For instance, we can use our techniques to construct a (rather inefficient) leakage resilient CCA2-secure encryption scheme that is provably secure in the *standard model.*

Comparison to Other Related Work We would like to mention that in a series of important recent works [9,5,23,22,4] *new* schemes for leakage resilient signing and encryption (CPA-secure) have been proposed. While these works have an obvious advantage over our work by considering a more powerful leakage model, we would like to point out that these schemes are non-standard, rather inefficient and rely on non-standard assumptions. Very recently, Dodis et al. [10] introduced a method for storing and refreshing a secret. Their construction does not require leak-free components, but is rather inefficient and relies on computational assumptions. Moreover, it is not clear if it can be used for other purposes such as implementing standard cryptosystems.

2 Preliminaries

For a natural number n the set $\{1, \dots, n\}$ will be denoted by $[n]$. If X is a random variable then we write $x \leftarrow X$ for the value that the random variable takes when sampled according to the distribution of X. In this paper, we will slightly abuse notation and also denote by X the probability distribution on the range of the variable. V is a row vector, and we denote by V^{T} its transposition. We let $\mathbb{F}$ be a finite field and for $m, n \in \mathbb{N}$, let $\mathbb{F}^{m \times n}$ denote the set of $m \times n$-matrices over $\mathbb{F}$. Typically, we use M_i to denote the column vectors of the matrix M. For a matrix $M \in \mathbb{F}^{m \times n}$ and an m bit vector $V \in \mathbb{F}^m$ we denote by $V \cdot M$ the n-element vector that results from matrix multiplication of V and M. For a natural number n by (0^n) we will denote the vector $(0, \dots, 0)$ of length n. We will often use the set of non-singular $m \times m$ matrices denoted by $\mathsf{NonSing}^{m \times m}(\mathbb{F}) \subset \mathbb{F}^{m \times m}$.

Let in the rest of this work n be the statistical and k be the computational security parameter. Let $\mathbb{G}$ be a group of prime order p such that $\log_2(p) \geq k$. We denote by $(p, \mathbb{G}) \leftarrow \mathsf{G}$ a group sampling algorithm. Let g be a generator of $\mathbb{G}$, then for a (column/row) vector $A \in \mathbb{Z}_p^n$ we denote by g^A the vector $C = (g^{A_1}, \dots, g^{A_n})$. Furthermore, let C^B be the vector $(g^{A_1 B_1}, \dots, g^{A_n B_n})$.

Let X_0, X_1 be random variables distributed over $\mathcal{X}$ and Y be a random variable over a set $\mathcal{Y}$, then we define the statistical distance between X_0 and X_1 as $\Delta(X_0; X_1) = \sum_{x \in \mathcal{X}} 1/2 |\Pr[X_0 = x] - \Pr[X_1 = x]|$. Moreover, let $\Delta(X_0; X_1|Y) \overset{\mathsf{def}}{=} \Delta((Y, X_0); (Y, X_1))$ be the statistical distance conditioned on Y.

2.1 Model of Leakage

In this work, we assume that the memory of a physical device is split into two parts, which leak *independently*. We model this in form of a *leakage game*, where the adversary can *adaptively* learn information from each part of the memory. More formally, let $L, R \in \{0,1\}^s$ be the two parts of the memory, then for a parameter $\lambda \in \mathbb{N}$, we define a λ-*leakage game* played between an adaptive adversary $\mathcal{A}$ – called a λ-*limited adversary* – and a *leakage oracle* $\Omega(L, R)$ as follows. For some $t \in \mathbb{N}$, the adversary $\mathcal{A}$ can adaptively issue a sequence $\{(f_i, x_i)\}_{i=1}^t$ of requests to the oracle $\Omega(L, R)$, where $x_i \in \{L, R\}$ and $f_i : \{0,1\}^s \to \{0,1\}^{\lambda_i}$. For the ith query the oracle replies with $f_i(x_i)$. The only restriction is that in total the adversary does not learn more than λ bits from each L and R. In the following, let $\mathit{Out}(\mathcal{A}, \Omega(L, R))$ be the output of $\mathcal{A}$ at the end of this game. Without loss of generality, we assume that $\mathit{Out}(\mathcal{A}, \Omega(L, R)) := (f_1(x_1), \dots, f_t(x_t))$.

Leakage from Computation. So far, we discussed how to model leakage from the memory of a device, where the memory is split into two parts (L, R). If the physical device carries out "some computation" using its memory (L, R), then this computation leaks information to the adversary. We model this in form of a two-party protocol $\Pi = (P_\mathsf{L}, P_\mathsf{R})$ executed between the parties P_L and P_R.

Initially, the party P_L holds L, while P_R holds R. The execution of Π with initial inputs L and R, denoted by $\Pi(L, R)$, proceeds in rounds. In each round one player is active and sends messages to the other one. These messages can depend on his input (i.e., his initial state), his local randomness, and the messages that he received in earlier rounds. Additionally, the *user* of the protocol (or the adversary – in case the user is malicious) may interact with the protocol, i.e., he may receive messages from the players and send messages to them. For simplicity, we assume that messages that are sent by the user to the protocol are delivered to both parties P_L and P_R. At the end of the protocol's execution, the players P_L and P_R (resp.) may output a value L' and R' (resp.). These outputs may be viewed as the *new* internal state of the protocol.

One natural way to describe the leakage of the computation (and memory) of such a protocol is to allow the adversary to adaptively pick at the beginning of each round a leakage function f and give $f(\mathsf{state})$ to the adversary. Here, state contains the initial state of the active party, its local randomness and the messages sent and received during this round. Indeed, we allow the adversary to learn such leakages. To ease description, we consider however a stronger model, and use the concept of a leakage game introduced earlier in this section. More precisely, for player $P_x \in \{P_\mathsf{L}, P_\mathsf{R}\}$, we denote the local randomness that is used by P_x as ρ_x, and all the messages that are received *or* sent (including the messages from the user of the protocol) by M_x. At any point in time, we allow the adversary $\mathcal{A}$ to play a λ-leakage game against the leakage oracle $\Omega((L, \rho_\mathsf{L}, M_\mathsf{L}); (R, \rho_\mathsf{R}, M_\mathsf{R}))$. A technical problem may arise if $\mathcal{A}$ asks for leakages *before* sending regular messages to the players. In such a case parts of M_x may be undefined, and for simplicity, we will set them to constant 0. For some initial state (L, R), we denote the output of $\mathcal{A}$ after this process with $\mathcal{A} \leftrightarrows (\Pi(L, R) \rightarrow (L', R'))$.

As we are interested in the continuous leakage setting, we will mostly consider an adversary that runs in many executions of $\mathcal{A} \leftrightarrows (\Pi(L, R) \rightarrow (L', R'))$. For the ith execution of the protocol $\Pi(L^{i-1}, R^{i-1})$, we will write

$$\mathcal{A} \leftrightarrows \left(\Pi(L^{i-1}, R^{i-1}) \rightarrow (L^i, R^i)\right),$$

where the current initial state of this round is (L^{i-1}, R^{i-1}) and the new state of P_L and P_R will be (L^i, R^i). After $\mathcal{A} \leftrightarrows \left(\Pi(L^{i-1}, R^{i-1}) \rightarrow (L^i, R^i)\right)$, we assume that the players P_L and P_R erase their current state except for their new state L^i and R^i, respectively. For the ith execution of $\mathcal{A} \leftrightarrows \left(\Pi(L^{i-1}, R^{i-1}) \rightarrow (L^i, R^i)\right)$, we let the adversary interact with the leakage oracle $\Omega((L^{i-1}, \rho^i_\mathsf{L}, M^i_\mathsf{L}); (R^{i-1}, \rho^i_\mathsf{R}, M^i_\mathsf{R}))$. If $\mathcal{A}$ is a λ-limited adversary, then we allow him to learn up to λ bits from the oracle in each such execution.

2.2 Leakage-Resilient Storage

A leakage-resilient storage (LRS) $\Phi = (\mathsf{Encode}, \mathsf{Decode})$ allows to store a secret in an "encoded form" such that even given leakage from the encoding no adversary learns information about the encoded values. A simple LRS for the independent leakage model can be based on two source extractors. More precisely, an LRS for the independent leakage model is defined for message space $\mathcal{M}$ and encoding space $\mathcal{L} \times \mathcal{R}$ as follows:

– $\mathsf{Encode} : \mathcal{M} \to \mathcal{L} \times \mathcal{R}$ is a probabilistic, efficiently computable function and
– $\mathsf{Decode} : \mathcal{L} \times \mathcal{R} \to \mathcal{M}$ is a deterministic, efficiently computable function such that for every $S \in \mathcal{M}$ we have $\mathsf{Decode}(\mathsf{Encode}(S)) = S$.

An LRS Φ is said to be (λ, ϵ)-secure, if for any $S, S' \in \mathcal{M}$ and any λ-limited adversary $\mathcal{A}$, we have

$$\Delta(\mathit{Out}(\mathcal{A}, \Omega(L, R)); \mathit{Out}(\mathcal{A}, \Omega(L', R'))) \leq \epsilon,$$

where $(L, R) := \mathsf{Encode}(S)$ and $(L', R') := \mathsf{Encode}(S')$.

We consider a leakage-resilient storage scheme that allows to efficiently store elements $S \in \mathbb{F}^m$ for some $m \in \mathbb{N}$. Namely, we propose $\Phi_{\mathbb{F}}^{n,m} = (\mathsf{Encode}_{\mathbb{F}}^{n,m}, \mathsf{Decode}_{\mathbb{F}}^{n,m})$ defined as follows:

– $\mathsf{Encode}_{\mathbb{F}}^{n,m}(S)$ first selects $L \leftarrow \mathbb{F}^n \setminus \{(0^n)\}$ at random, and then samples $R \leftarrow \mathbb{F}^{n \times m}$ such that $L \cdot R = S$. It outputs (L, R).
– $\mathsf{Decode}_{\mathbb{F}}^{n,m}(L, R)$ outputs $L \cdot R$.

The following lemma shows that $\Phi_{\mathbb{F}}^{n,m}$ is a secure LRS. The proof uses the fact that an inner product over a finite field is a two-source extractor [7,28] and appears in the full version.

Lemma 1. *Let $m, n \in \mathbb{N}$ with $m < n$ and let $\mathbb{F}$ such that $|\mathbb{F}| = \Omega(n)$. For any $1/2 > \delta > 0, \gamma > 0$ the LRS $\Phi_{\mathbb{F}}^{n,m}$ as defined above is (λ, ϵ)-secure, with $\lambda = (1/2 - \delta) n \log |\mathbb{F}| - \log \gamma^{-1}$ and $\epsilon = 2m(|\mathbb{F}|^{m+1/2-n\delta} + |\mathbb{F}^m| \gamma)$.*

The following is an instantiation of Lemma 1 for concrete parameters.

Corollary 1. *Suppose $|\mathbb{F}| = \Omega(n)$ and $m < n/20$. Then, LRS $\Phi_{\mathbb{F}}^{n,m}$ is $(0.3 \cdot |\mathbb{F}^n|, \mathit{negl}(n))$-secure, for some negligible function negl.*

3 Leakage-Resilient Refreshing of LRS

For a secret S and a leakage resilient storage $\Phi = (\mathsf{Encode}, \mathsf{Decode})$ with message space $\mathcal{M}$, we develop a probabilistic protocol $(L', R') \leftarrow \mathsf{Refresh}(L, R)$ that securely refreshes $(L, R) \leftarrow \mathsf{Encode}(S)$, even when the adversary can *continuously* observe the computation from the refreshing process. The only additional assumption that we make is that the protocol has access to a simple leak-free source $\mathcal{O}$ of correlated randomness.

Initially, P_{L} holds L and P_{R} holds R. At any point during the execution of the protocol, the adversary can interact with a leakage oracle and learn information about the internal state of P_{L} and P_{R}. At the end the players output the "refreshed" encoding (L', R'), i.e., the new state of the protocol. Notice that the only way in which the adversary can "interact" with the protocol is via the leakage oracle.

For *correctness*, we require that $\mathsf{Decode}(L, R) = \mathsf{Decode}(L', R')$ Informally, for security, we require that no λ-limited adversary can learn any significant information about S (for some parameter $\lambda \in \mathbb{N}$). We will define the security of the refreshing protocol using an indistinguishability notion. Intuitively, the definition says that for any two secrets $S, S' \in \mathcal{M}$ the view (i.e., the leakage) resulting from the execution of the refreshing of secret S is statistically close to the view from the refreshing of secret S'. Before we formally define security of our refreshing, we consider the following experiment, which runs the refreshing protocol for ℓ rounds and lets the adversary play a leakage game in each round. For a protocol Π, an LRS Φ, a λ-bounded adversary $\mathcal{A}$, $\ell \in \mathbb{N}$ and $S \in \mathcal{M}$, we have $\mathsf{Exp}_{(\Pi,\Phi)}(\mathcal{A}, S, \ell)$:

1. For a secret S, we generate the initial encoding as $(L^0, R^0) \leftarrow \mathsf{Encode}(S)$.
2. For $i = 1$ to ℓ run $\mathcal{A}$ against the *ith round* of the refreshing protocol: $\mathcal{A} \leftrightarrows \big(\Pi(L^{i-1}, R^{i-1}) \rightarrow (L^i, R^i)\big)$.
3. Return whatever $\mathcal{A}$ outputs.

Wlog. we assume that $\mathcal{A}$ outputs just a single bit $b \in \{0, 1\}$. To simplify notation, we will sometimes omit to specify Φ in $\mathsf{Exp}_{(\Pi,\Phi)}(\mathcal{A}, S, \ell)$ explicitly. We are now ready to define security of a refreshing protocol.

Definition 1 (A $(\ell, \lambda, \epsilon)$-refreshing protocol). *For a LRS $\Phi = (\mathsf{Encode}, \mathsf{Decode})$ with message space $\mathcal{M}$, a refreshing protocol $(\mathsf{Refresh}, \Phi)$ is $(\ell, \lambda, \epsilon)$-*secure*, if for every λ-limited adversary $\mathcal{A}$ and any two secrets $S, S' \in \mathcal{M}$, we have that $\Delta(\mathsf{Exp}_{(\mathsf{Refresh},\Phi)}(\mathcal{A}, S, \ell); \mathsf{Exp}_{(\mathsf{Refresh},\Phi)}(\mathcal{A}, S', \ell)) \leq \epsilon$.*

In the rest of this section, we construct a secure refreshing protocol for the LRS scheme $\Phi_{\mathbb{F}}^{n,m} = (\mathsf{Encode}_{\mathbb{F}}^{n,m}, \mathsf{Decode}_{\mathbb{F}}^{n,m})$ from Section 2.2. Our protocol can refresh an encoding $(L, R) \leftarrow \mathsf{Encode}_{\mathbb{F}}^{n,m}(S)$ any polynomial number of times, and guarantees security for λ being a constant fraction of the length of L and R (cf. Theorem 1 and Corollary 2 for the concrete parameters). To ease notation, we often omit to specify $\Phi_{\mathbb{F}}^{n,m}$ when talking about the refreshing protocol $(\mathsf{Refresh}_{\mathbb{F}}^{n,m}, \Phi_{\mathbb{F}}^{n,m})$ and just write $\mathsf{Refresh}$.

As outlined in the introduction, we assume that the players have access to a non-uniform source of randomness. More precisely, they may access an oracle $\mathcal{O}$ that samples pairs $(A, B) \in \mathbb{F}^n \times \mathsf{NonSing}^{n \times m}(\mathbb{F})$ such that $A \neq (0^n)$ and $A \cdot B = (0^m)$. In each iteration the players will sample the oracle twice: once for refreshing the share of P_{R} (denote the sampled pair by (A, B)), and once for refreshing the share of P_{L} (denote the sampled pair by $(\tilde{A}, \tilde{B})$). The protocol is depicted on Fig. 1. To understand the main idea behind the protocol, the reader may initially disregard the checks (in Steps 1 and 4) that L and R' have full

Protocol $(L', R') \leftarrow \mathsf{Refresh}_{\mathbb{F}}^{n,m}(L, R)$**:**

Input (L, R)**:** $L \in \mathbb{F}^n$ is given to P_{L} and $R \in \mathbb{F}^{n \times m}$ is given to P_{R}.

Refreshing the share of P_{R}**:**

1. If L does not have a full rank then the players abort. Let $(A, B) \leftarrow \mathcal{O}$ and give A to P_{L} and B to P_{R}.
2. Player P_{L} generates a random non-singular matrix $M \in \mathbb{F}^{n \times n}$ such that $L \cdot M = A$ and sends it to P_{R}.
3. Player P_{R} sets $X := M \cdot B$ and $R' := R + X$.

Refreshing the share of P_{L}**:**

4. If R' does not have a full rank then the players abort. Let $(\tilde{A}, \tilde{B}) \leftarrow \mathcal{O}$ and give $\tilde{A}$ to P_{L} and $\tilde{B}$ to P_{R}.
5. Player P_{R} generates a random non-singular matrix $\tilde{M} \in \mathbb{F}^{n \times n}$ such that $\tilde{M} \cdot R' = \tilde{B}$ and sends it to P_{L}.
6. Player P_{L} sets $Y := \tilde{A} \cdot \tilde{M}$ and $L' := L + Y$.

Output: The players output (L', R').

The adversary plays a λ-leakage game against:
$\Omega\left((L, A, M, \tilde{A}, \tilde{M}) \; ; \; (R, B, M, \tilde{B}, \tilde{M})\right)$

Fig. 1. Protocol $\mathsf{Refresh}_{\mathbb{F}}^{n,m}$. The oracle $\mathcal{O}$ samples randomly pairs $(A, B) \in \mathbb{F}^n \times \mathsf{NonSing}^{n \times m}(\mathbb{F})$ such that $A \neq (0^n)$ and $A \cdot B = (0^m)$. The text in the frame describes the leakage game played by the adversary. Note that sampling the random matrices in Steps 2 and 5 can be done efficiently.

rank (these checks were introduced only to facilitate the proof and only occur with very small probability). The reader may also initially assume that $m = 1$ (the case of $m > 1$ is a simple generalization of the $m = 1$ case). The main idea of our protocol is that first the players generate the value $X \in \mathbb{F}^{n \times m}$ such that $L \cdot X = (0^m)$, and then in Steps 3 the player P_{R} sets $R' := R + X$ (note that, by simple linear algebra $L \cdot R' = L \cdot (R + X) = L \cdot R + L \cdot X = L \cdot R$). Symmetrically, later, the players generate $Y \in \mathbb{F}^n$ such that $Y \cdot R' = (0^m)$ and set (in Step 6) $L' = L + Y$. By a similar reasoning as before we have $L' \cdot R' = L \cdot R' (= L \cdot R)$. The above analysis gives us the correctness of our protocol.

Lemma 2 (Correctness of the refreshing). *Assuming that the players* P_{L} *and* P_{R} *did not abort, we have for any* $S \in \mathbb{F}^m$*:* $\mathsf{Decode}_{\mathbb{F}}^{n,m}(\mathsf{Refresh}_{\mathbb{F}}^{n,m}(S)) = S$.

We now state our main theorem which shows that the protocol $\mathsf{Refresh}_{\mathbb{F}}^{n,m}$ from Figure 1 satisfies Definition 1. In the full version of this paper, we show that our refreshing is secure even if the adversary has some (not necessarily short) auxiliary information about the encoding.

Theorem 1 (Security of $\mathsf{Refresh}_{\mathbb{F}}^{n,m}$**).** *Let* $m/3 \leq n$*,* $n \geq 16$ *and* $\ell \in \mathbb{N}$*. Let* n, m *and* $\mathbb{F}$ *be such that* $\Phi_{\mathbb{F}}^{n,m}$ *is* (λ, ϵ)*-secure (for some* λ *and* ϵ*). The protocol* $\mathsf{Refresh}_{\mathbb{F}}^{n,m}$ *is a* $(\ell, \lambda/2 - 1, \epsilon')$-refreshing protocol *for an LRS* $\Phi_{\mathbb{F}}^{n,m}$ *with* $\epsilon' := 2\ell \left|\mathbb{F}\right|^m (3 \left|\mathbb{F}\right|^m \epsilon + m \left|\mathbb{F}\right|^{-n-1})$.

For the proof of this theorem, we will need to show that any adversary $\mathcal{A}$ that interacts for ℓ iterations with the refreshing experiment $\mathsf{Exp}_{\mathsf{Refresh}}$ (as given in

Definition 1), will only gain a negligible (in n) amount of information about the encoded secret S. Notice that this in particular means that $\mathcal{A}$'s interaction with the leakage oracle given in the frame of Figure 1 will not provide the adversary with information on the encoded secret. More formally, we will show that for every $(\lambda/2-1)$-limited $\mathcal{A}$ and every S, S' we have:

$$\Delta(\mathsf{Exp}_{\mathsf{Refresh}}(\mathcal{A}, S, \ell); \mathsf{Exp}_{\mathsf{Refresh}}(\mathcal{A}, S', \ell)) \leq 2\ell \left|\mathbb{F}\right|^m \left(3\left|\mathbb{F}\right|^m \epsilon + m\left|\mathbb{F}\right|^{-n-1}\right). \quad (1)$$

This will be proven using the standard technique called the "hybrid argument" by creating a sequence of "hybrid distributions". We will show that the first distribution in this sequence is statistically very close to $\mathsf{Exp}_{\mathsf{Refresh}}(\mathcal{A}, S, \ell)$, while the latter is close to $\mathsf{Exp}_{\mathsf{Refresh}}(\mathcal{A}, S', \ell)$. Moreover, each two consecutive distributions in the sequence will be statistically close. Hence, by applying the triangle inequality multiple times, we will obtain that $\mathsf{Exp}_{\mathsf{Refresh}}(\mathcal{A}, S, \ell)$ and $\mathsf{Exp}_{\mathsf{Refresh}}(\mathcal{A}, S', \ell)$ are close. The proof of the theorem is deferred to the full version of this paper. Combining Theorem 1 with Corollary 1 we get the following.

Corollary 2. *Let $n \in \mathbb{N}$ be the security parameter. Suppose $|\mathbb{F}| = \Omega(n)$ and let $m = o(n)$. Then $\mathsf{Refresh}_{\mathbb{F}}^{n,m}$ is a $(\ell, 0.15 \cdot n\log(|\mathbb{F}|) - 1, negl(n))$*-refreshing protocol *for the LRS $\Phi_{\mathbb{F}}^{n,m}$, where ℓ is a polynomial in n and $negl(n)$ is some negligible function.*

4 Identification and Signature Schemes

In an identification scheme ID a prover attempts to prove its identity to a verifier. For a security parameter k, ID consists out of three PPT algorithms $\mathsf{ID} = (\mathsf{KeyGen}, \mathcal{P}, \mathcal{V})$:

- $(pk, sk) \leftarrow \mathsf{KeyGen}(1^k)$: It outputs the public parameters of the scheme and a valid key pair.
- $(\mathcal{P}(pk, sk), \mathcal{V}(pk))$: An interactive protocol in which $\mathcal{P}$ tries to convince $\mathcal{V}$ of its identity by using his secret key sk. The verifier $\mathcal{V}$ outputs either *accept* or *reject*.

We require that ID is *complete*. This means that an honest prover will always be accepted by the verifier. The standard security definition of an identification scheme ID considers a polynomial-time adversary $\mathcal{A}$ that inputs the public key pk and interacts with the prover $\mathcal{P}(pk, sk)$ playing the role of a verifier. Then, $\mathcal{A}$ tries to impersonate $\mathcal{P}(pk, sk)$ by engaging in an interaction with $\mathcal{V}(pk)$. We say that the scheme is *secure* if every polynomial-time adversary $\mathcal{A}$ impersonates the prover with only negligible probability.

We extend this standard security to incorporate leakage from the prover's computation. To this end, we let the adversary take the role of $\mathcal{V}$ in the execution of the protocol $(\mathcal{P}(pk, sk), \mathcal{V}(pk))$ and allow him to obtain leakage from the prover's execution. We denote a single execution of this process by $\mathcal{A} \leftrightarrows \big(\mathcal{P}(sk) \to sk'\big)$, where sk' may be the updated key.

Definition 2 (Security against Leakage and Impersonation Attacks (ID-LEAK security)). *Let $k \in \mathbb{N}$ be the security parameter. An identification scheme* $\mathsf{ID} = (\mathsf{KeyGen}, \mathcal{P}, \mathcal{V})$ *is $\lambda(k)$-ID-LEAK secure if for any PPT $\lambda(k)$-limited adversary $\mathcal{A}$ it holds that the experiment below outputs* 1 *with probability at most $negl(k)$:*

1. *The challenger samples $(pk, sk^0) \leftarrow \mathsf{KeyGen}(1^k)$ and gives pk to $\mathcal{A}$.*
2. *Repeat for $i = 0 \dots poly(k)$ times: $\mathcal{A} \leftrightarrows \left(\mathcal{P}(sk^i) \rightarrow sk^{i+1}\right)$, where in each execution the adversary can interact with the honest prover and gets up to $\lambda(k)$ bits about the current secret state sk^i and the randomness that is used.*
3. *$\mathcal{A}$ impersonates the prover and interacts with $\mathcal{V}(pk)$. If $\mathcal{V}(pk)$ accepts, then output* 1*; otherwise output* 0.

Notice that the adversary is allowed to obtain λ bits of information for *each* execution of the identification protocol.

4.1 A Construction of a Leakage-Resilient Identification Protocol

Our construction is based on the standard Okamoto identification scheme [25]. Let g_1 and g_2 be two generators of $\mathbb{G}$ such that $\alpha = \log_{g_1}(g_2)$ is unknown. The secret key sk is equal to $(x_1, x_2) \leftarrow \mathbb{Z}_p^2$ and the public key pk is $g_1^{x_1} \cdot g_2^{x_2}$.

1. $\mathcal{P}$ chooses $(w_1, w_2) \leftarrow \mathbb{Z}_p^2$, computes $a := g_1^{w_1} g_2^{w_2}$, and sends a to $\mathcal{V}$.
2. $\mathcal{V}$ chooses $c \leftarrow \mathbb{Z}_p$ and sends it to $\mathcal{P}$.
3. $\mathcal{P}$ computes $z_1 := w_1 + cx_1$ and $z_2 := w_2 + cx_2$ and sends (z_1, z_2) to $\mathcal{V}$.
4. $\mathcal{V}$ accepts if and only if $g_1^{z_1} g_2^{z_2} \overset{?}{=} a \cdot pk^c$.

We next describe how to implement the Okamoto scheme such that it remains secure even if the computation of the prover is carried out on a leaky device. Verification is as in the standard Okamoto scheme, while the key generation and the computation of the prover is adjusted to protect against leakage attacks. More precisely, instead of using $(x_1, x_2) \in \mathbb{Z}_p^2$ as secret key, we store $(L, (R_1, R_2)) \leftarrow \mathsf{Encode}_{\mathbb{F}}^{n,2}(x_1, x_2)$ and implement the computation of the prover as a two-party protocol run between $P_{\mathsf{L}}(L)$ and $P_{\mathsf{R}}(R_1, R_2)$. To this end, we will use the fact that the Okamoto identification protocol *only* requires to compute a linear function of the encoded secret key. The protocol is given in Figure 2.

Finally, we will combine our identification protocol with our protocol for refreshing to construct an identification scheme $\mathsf{Oka} = (\mathsf{KeyGen}, \mathcal{P}, \mathcal{V}, \mathsf{Refresh}_{\mathbb{Z}_p}^{n,2})$ that is ID-LEAK secure. More precisely, in the ith execution of $(\mathcal{P}(pk, (L, R)), \mathcal{V}(pk))$ after Step 5 in Figure 2, we execute $(L^{i+1}, R^{i+1}) \leftarrow \mathsf{Refresh}_{\mathbb{Z}_p}^{n,2}(L^i, R^i)$ and set the prover's secret key for the next round to $sk^{i+1} := (L^{i+1}, R^{i+1})$. Notice that in such a case, we include into the leakage oracle from the figure the variables that are used by the refreshing and let the adversary interact in each round with the following leakage oracle:

$$\Omega\left((L^i, U, Z, A, M, \tilde{A}, \tilde{M})\ ;\ (R^i, W, A, M, \tilde{A}, \tilde{M})\right).$$

Key generation $\mathsf{KeyGen}(1^k)$:

Sample $(p, \mathbb{G}) \leftarrow \mathsf{G}(1^k)$, generators $g_1, g_2 \leftarrow \mathbb{G}$, $S = (x_1, x_2) \leftarrow \mathbb{Z}_p^2$ and $(L, R) \leftarrow \mathsf{Encode}_{\mathbb{Z}_p}^{n,2}(S)$. Set $sk = (L, R)$ and $pk = (p, g_1, g_2, h := g_1^{x_1} g_2^{x_2})$.

The identification protocol $(\mathcal{P}(pk, (L, R)), \mathcal{V}(pk))$

Input for prover (L, R): L is given to P_L and R is given to P_R.

Prover $\mathcal{P}(pk, (L, R))$:	**Verifier $\mathcal{V}(pk)$:**
1. P_R samples $(W_1, W_2) \leftarrow \mathbb{Z}_p^{2n}$, computes $U := g_1^{W_1} \odot g_2^{W_2}$ and sets $W := (W_1^\mathsf{T}, W_2^\mathsf{T})$. The vector U is sent to P_L ($\odot$ is component-wise multiplication of vectors).	
2. P_L computes $V = U^L$ and $a = \prod_i V_i$. The value a is sent to $\mathcal{V}$.	
	3. Senc $c \leftarrow \mathbb{Z}_p$ to $\mathcal{P}$.
4. P_R computes the $n \times 2$ matrix $Z := W + cR$ and sends it to P_L.	
5. P_L computes $(z_1, z_2) = L \cdot Z$. The values (z_1, z_2) are given to $\mathcal{V}$.	
At any time, the adversary can play a λ-leakage game against: $\Omega\left((L, U, Z)\ ;\ (R, W)\right)$. We set $Z = 0$ for leakage queries that are asked *before* c is fixed.	
	6. Accept iff $g_1^{z_1} g_2^{z_2} = a h^c$.

Fig. 2. The key generation algorithm and the protocol $(\mathcal{P}(pk, (L, R)), \mathcal{V}(pk))$ for identification. $(\mathcal{P}(pk, (L, R)), \mathcal{V}(pk))$ is an interactive protocol between a prover $\mathcal{P}$ and a verifier $\mathcal{V}$.

It is easy to see that the above protocol satisfies the completeness property. This is due to the correctness of the refreshing protocol, and the fact that messages that are exchanged by the parties $\mathcal{P}$ and $\mathcal{V}$ in Figure 2 are as in the original Okamoto protocol. The security of our protocol Oka is proven in the following theorem.

Theorem 2. $\mathsf{Oka} = (\mathsf{KeyGen}, \mathcal{P}, \mathcal{V}, \mathsf{Refresh}_{\mathbb{Z}_p}^{n,2})$ *is* $((0.15 \cdot n - 3) \log p - 1)$*-ID-LEAK secure, if the DL assumption holds.*

The proof follows from the following three observations:

1. We first consider a single execution of the protocol $(\mathcal{P}(pk, (L, R)), \mathcal{V}(pk))$ from Figure 2 and prove a simple property in the information theoretic setting. Namely, we show that the there exists an (unbounded) simulator with access to a leakage oracle $\Omega(L^*, R^*)$ can simulate $\mathcal{A}(pk)$'s view in $\mathcal{A} \leftrightarrows (\mathcal{P}(L, R)) \rightarrow (L, R))$. In this step the analysis neglects the leakage from the refreshing process as we consider only a *single* run of the protocol.

2. We next consider the setting where unbounded $\mathcal{A}$ runs in many iterations of $\mathcal{A} \leftrightarrows \big(\mathcal{P}(L^i, R^i)) \rightarrow (L^{i+1}, R^{i+1})\big)$, where we also take into account that the refreshing of (L^i, R^i) leaks information. We will combine our results from the last section with the simulator defined in 1 to show that *any* unbounded adversary will only learn a negligible amount of information about the secret key.
3. Finally, we will argue why this proves the ID-Leak security of our scheme. To this end, we rely on a recent result of Dodis et al. [2], which shows security of the original Okamoto scheme for keys sampled from a high average min-entropy source.

Leakage Resilient Signatures It is well known fact that the Okamoto identification protocol can be turned into a signature scheme using the Fiat-Shamir heuristic. Similarly, we can turn the scheme from Figure 2 into a leakage resilient signature scheme which can be proven secure against continuous leakage attacks in the random oracle model under the DL assumption.

5 Leakage Resilient Encryption

In this section, we construct an efficient encryption schemes that is secure against continuous leakage attacks. Our construction is based on a variant of the ElGamal cryptosystem and is proven secure against adaptive chosen message and leakage attacks (CCLA2) in the Random Oracle model.

5.1 Definitions

For security parameter k a public-key encryption scheme $\mathsf{PKE} = (\mathsf{KeyGen}, \mathsf{Encr}, \mathsf{Decr})$ consists of three PPT algorithms.

– $(pk, sk) \leftarrow \mathsf{KeyGen}(1^k)$: It outputs a valid public/secret key pair.
– $c \leftarrow \mathsf{Encr}(pk, m)$: That is, a probabilistic algorithm that on input some message m and the public key pk outputs a ciphertext $c = \mathsf{Encr}(pk, m)$.
– $m = \mathsf{Decr}(sk, c)$: The decryption algorithm takes as input the secret key sk and a ciphertext c such that for any m we have $m = \mathsf{Decr}(sk, \mathsf{Encr}(pk, m))$.

To define security we allow the adversary to query the decryption oracle on some chosen ciphertext c, and additionally allow him to obtain a bounded amount of leakage from the decryption process. This may be repeated many times, hence, eventually the adversary may learn a large amount of information. Formally, we define security against adaptive chosen ciphertext and leakage attacks (IND-CCLA2 security) as follows.

Definition 3 (Security against Chosen Ciphertext Leakage Attacks (CCLA2-secure)). *Let $k \in \mathbb{N}$ be the security parameter. A public-key encryption scheme* $\mathsf{PKE} = (\mathsf{KeyGen}, \mathsf{Encr}, \mathsf{Decr})$ *is $\lambda(k)$-IND-CCLA2 secure if for any PPT $\lambda(k)$-limited adversary $\mathcal{A}$ the probability that the experiment below outputs* 1 *is at most* $1/2 + negl(k)$.

1. *Sample* $b \leftarrow \{0,1\}$ *and* $(pk, sk) \leftarrow \mathsf{KeyGen}(1^k)$*. Give* pk *to* $\mathcal{A}$*.*
2. *Repeat until* $\mathcal{A}(1^k)$ *outputs* (m_0, m_1)*:* $\mathcal{A}(1^k) \leftrightarrows \big(\mathsf{Decr}(sk, c) \rightarrow sk'\big)$*, where for each decryption query* c *the adversary additionally retrieves up to* $\lambda(k)$ *bits about the current secret state* sk*. Set the key for the next round to* $sk := sk'$*.*
3. *The challenger computes* $c^* \leftarrow \mathsf{Encr}(pk, m_b)$ *and gives it to* $\mathcal{A}$*.*
4. *Repeat until* $\mathcal{A}(1^k)$ *outputs* b'*:* $\mathcal{A}(1^k) \leftrightarrows \big(\mathsf{Decr}(sk, c) \rightarrow sk'\big)$*, where for each decryption query* $c \neq c^*$ *the adversary additionally retrieves up to* $\lambda(k)$ *bits about the current secret state* sk*. Set the key for the next round to* $sk := sk'$*.*
5. *If* $b = b'$ *then output* 1*; otherwise output* 0*.*

The weaker notion of CCLA1-security can be obtained by omitting Step 4 in the experiment above.

5.2 Efficient IND-CCLA2-secure Encryption

An important tool of our encryption scheme is a simulation-sound (SS) NIZK. Informally, a NIZK proof system is said to be simulation sound, if any adversary has negligible advantage in breaking soundness (i.e., forging an accepting proof for an invalid statement), *even* after seeing a bounded number of proofs for (in)valid statements. We refer the reader to [3,29] for the formal definition of NIZKs and simulation soundness. SS-NIZKs can be instantiated in the common random string model using the Groth-Sahai proof system [18] and the techniques of [17]. Unfortunately, this results into an impractical scheme. In contrast, in the random oracle model using the Fiat-Shamir heuristic [14] simulation soundness can be achieved efficiently. In particular, it has been proven in [1] that the standard Chaum-Pedersen protocol [6] for proving equivalence of discrete logarithms can be turned into a SS-NIZK using the Fiat-Shamir heuristic. Let in the following $(\mathsf{Prov}, \mathsf{Ver})$ denote such a non-interactive proof system for proving the equivalence of discrete logarithms.

Our scheme can be viewed as a leakage-resilient implementation of the following simple variant of the ElGamal encryption scheme using the above simulation sound NIZK. Let g_1, g_2 be two generators of a prime order p group $\mathbb{G}$. Let $sk = (x_1, x_2) \in \mathbb{Z}_p^2$ be the secret key and $pk = (g_1, g_2, h = g_1^{x_1} \cdot g_2^{x_2})$ the public key. To encrypt a message $m \in \mathbb{G}$, pick uniformly $r \leftarrow \mathbb{Z}_p$ and compute $c = (u := g_1^r, v := g_2^r, w := h^r m, \pi)$, where $\pi := \mathsf{Prov}(u, v, r)$ is a NIZK proof of $\log_{g_1}(u) = \log_{g_2}(v)$. To decrypt $c = (u, v, w, \pi)$, verify the NIZK, and if it accepts, output $w \cdot (u^{-x_1} \cdot v^{-x_2})$.

It can easily be shown that this scheme achieves standard CCA2 security in the RO model. In this section, we will show how to *implement* this scheme such that it remains secure even if the decryption continuously leaks information. Similar to our transformation of the Okamoto scheme, we store the secret key (x_1, x_2) as $(L, R) \leftarrow \mathsf{Encode}_{\mathbb{F}}^{n,2}(x_1, x_2)$ and implement the computation of the decryption process as a two-party protocol run between $P_{\mathsf{L}}(L)$ and $P_{\mathsf{R}}(R)$. The protocol for key generation and decryption is given in Figure 3. Finally, we will combine the protocol from Figure 3 with our refreshing protocol from Section 3 to construct an encryption scheme $\mathsf{PKE} = (\mathsf{KeyGen}, \mathsf{Encr}, \mathsf{Decr}, \mathsf{Refresh}_{\mathbb{Z}_p}^{n,2})$ that is CCLA2 secure.

Key generation $\mathsf{KeyGen}(1^k)$:

Let $(p, \mathbb{G}) \leftarrow \mathsf{G}(1^k)$, $g_1, g_2 \leftarrow \mathbb{G}$, $S = (x_1, x_2) \leftarrow \mathbb{Z}_p^2$ and $(L, R) \leftarrow \mathsf{Encode}_{\mathbb{Z}_p}^{n,2}(S)$. Let $sk = (L, R)$ and $pk = (p, g_1, g_2, h := g_1^{x_1} g_2^{x_2})$.

Encryption $\mathsf{Encr}(pk, m)$:

Sample $r \leftarrow \mathbb{Z}_p$ uniformly at random and compute $c = (u := g_1^r, v := g_2^r, w := h^r m)$. Run the NIZK prover $\mathsf{Prov}(u, v, r)$ to obtain a proof π for $\log_{g_1}(u) = \log_{g_2}(v)$. Return (c, π).

The protocol for decryption $\mathsf{Decr}(sk, c)$:

Input for decryption $sk := (L, R)$: L is given to P_{L} and R is given to P_{R}.

Both parties obtain c and parse it as (u, v, w, π). If $\mathsf{Ver}(u, v, \pi) = \textit{reject}$ then abort; otherwise proceed as follows:

1. P_{R} computes the vector $U := u^{R_1} \odot v^{R_2}$. U is sent to P_{L} ($\odot$ denotes component-wise multiplication of vectors).
2. P_{L} computes $V = U^{-L}$ and outputs $w \prod_i V_i$.

Notice that we can omit the leakage from the verification of the NIZK as it only includes publicly known values. At any time, the adversary can play a λ-leakage game against: $\Omega\left((L, U)\ ;\ R\right)$.

Fig. 3. Our public-key encryption scheme PKE

The security analysis follows the outline given in the last section. We first show that the leakage from a single decryption query can be simulated in a perfect way with just access to a leakage oracle $\Omega(L^*, R^*)$. For this simulation to go through, we require that an adversary can only observe leakage from operations that involve the secret key, *if* the decryption oracle is queried on a valid ciphertexts. We call a ciphertext *valid*, if $\log_{g_1}(u) = log_{g_2}(v)$ holds. Notice that this is also the reason why we need NIZKs and cannot use the standard techniques to get CCA1/2 security based on hash proof systems. In the next step, we show that even when the adversary can continuously obtain leakage from the decryption, he will not be able to learn information about the encoded secret key. To this end, we will combine the scheme from Figure 3 with our refreshing protocol $\mathsf{Refresh}_{\mathbb{Z}_p}^{n,2}$. In the following theorem, we show IND-CCLA2 security of our scheme.

Theorem 3. PKE *is* $(0.15 \cdot n \log p - 1)$*-IND-CCLA2 secure in the random oracle model, if the DDH assumption holds.*

6 A General Paradigm for Leakage-Resilient Cryptographic Schemes

In the last sections, we proposed leakage-resilient implementations of standard cryptographic schemes. Namely, we showed how to implement the standard Okamoto identification scheme and a variant of the ElGamal encryption scheme such that they satisfy strong security guarantees even under continuous leakage attacks. The security proof of both schemes relied on very similar observations, namely:

1. The underlying cryptographic scheme (e.g., the Okamoto scheme or the El-Gamal variant) computes only a linear function of the secret key. Notice that in the examples of the last section the linear function was computed in the exponen. This is not a problem as long as the computation can be carried out efficiently. This was indeed the case for the schemes of the last sections.
2. The secret key is hidden information theoretically even given the protocol transcript that an adversary obtains when interacting with the underlying cryptographic scheme. In the protocols from the last section, for instance, the secret key (x_1, x_2) was information theoretically hidden even given the corresponding public key. Furthermore, for the Okamoto scheme this holds even given (a, z_1, z_2), which were sent by the prover to the verifier.

Various other cryptographic schemes satisfy the above properties, and hence can be made secure against continuous leakage attacks. For instance, the Pedersen commitment scheme [26], which is information-theoretically hiding and at the same time only requires to compute a linear function of its secrets.[2] Another example of the above paradigm is a variant of the linear Cramer-Shoup cryptosystem as presented in [30]. Notice that as in the encryption scheme from Section 5, this requires to use as a check for the validity of the ciphertexts a NIZK proof system. One can instantiate such a NIZK in the standard model using the Groth-Sahai proof system [18]. This gives us an efficient CCLA1-secure public-key encryption scheme in the standard model, and a rather inefficient CCLA2-secure scheme using the extensions of [17]. We suggest that many other standard cryptographic schemes can be proven secure following the ideas that were presented in this paper.

Acknowledgments. The authors are grateful to Francesco Davi, Yevgeniy Dodis, Krzysztof Pietrzak, Leonid Reyzin and Daniele Venturi for helpful discussions on the problem of leakage-resilient refreshing.

References

1. Abdalla, M., Boyen, X., Chevalier, C., Pointcheval, D.: Distributed Public-Key Cryptography from Weak Secrets. In: Jarecki, S., Tsudik, G. (eds.) PKC 2009. LNCS, vol. 5443, pp. 139–159. Springer, Heidelberg (2009)
2. Alwen, J., Dodis, Y., Wichs, D.: Leakage-Resilient Public-Key Cryptography in the Bounded-Retrieval Model. In: Halevi, S. (ed.) CRYPTO 2009. LNCS, vol. 5677, pp. 36–54. Springer, Heidelberg (2009)
3. Blum, M., Feldman, P., Micali, S.: Non-interactive zero-knowledge and its applications (extended abstract). In: STOC, pp. 103–112 (1988)
4. Boyle, E., Segev, G., Wichs, D.: Fully Leakage-Resilient Signatures. In: Paterson, K.G. (ed.) EUROCRYPT 2011. LNCS, vol. 6632, pp. 89–108. Springer, Heidelberg (2011)

[2] Notice that we computed a Pedersen commitment as part of the prover's protocol in our implementation of the Okamoto scheme.

5. Brakerski, Z., Kalai, Y.T., Katz, J., Vaikuntanathan, V.: Overcoming the hole in the bucket: Public-key cryptography resilient to continual memory leakage. In: FOCS, pp. 501–510 (2010)
6. Chaum, D., Pedersen, T.P.: Wallet Databases with Observers. In: Brickell, E.F. (ed.) CRYPTO 1992. LNCS, vol. 740, pp. 89–105. Springer, Heidelberg (1993)
7. Chor, B., Goldreich, O.: Unbiased bits from sources of weak randomness and probabilistic communication complexity. SIAM J. Comput. 17(2), 230–261 (1988)
8. Davì, F., Dziembowski, S., Venturi, D.: Leakage-Resilient Storage. In: Garay, J.A., De Prisco, R. (eds.) SCN 2010. LNCS, vol. 6280, pp. 121–137. Springer, Heidelberg (2010)
9. Dodis, Y., Haralambiev, K., López-Alt, A., Wichs, D.: Cryptography against continuous memory attacks. In: FOCS, pp. 511–520 (2010)
10. Dodis, Y., Lewko, A., Waters, B., Wichs, D.: How to store a secret on continually leaky devices. Accepted to FOCS 2011 (2011)
11. Dziembowski, S., Pietrzak, K.: Leakage-resilient cryptography. In: FOCS 2008: Proceedings of the 49th Annual IEEE Symposium on Foundations of Computer Science. IEEE Computer Society, Washington, DC, USA (2008)
12. Faust, S., Kiltz, E., Pietrzak, K., Rothblum, G.N.: Leakage-Resilient Signatures. In: Micciancio, D. (ed.) TCC 2010. LNCS, vol. 5978, pp. 343–360. Springer, Heidelberg (2010)
13. Faust, S., Rabin, T., Reyzin, L., Tromer, E., Vaikuntanathan, V.: Protecting Circuits from Leakage: the Computationally-Bounded and Noisy Cases. In: Gilbert, H. (ed.) EUROCRYPT 2010. LNCS, vol. 6110, pp. 135–156. Springer, Heidelberg (2010)
14. Fiat, A., Shamir, A.: How to Prove Yourself: Practical Solutions to Identification and Signature Problems. In: Odlyzko, A.M. (ed.) CRYPTO 1986. LNCS, vol. 263, pp. 186–194. Springer, Heidelberg (1987)
15. Goldwasser, S., Kalai, Y.T., Rothblum, G.N.: One-Time Programs. In: Wagner, D. (ed.) CRYPTO 2008. LNCS, vol. 5157, pp. 39–56. Springer, Heidelberg (2008)
16. Goldwasser, S., Rothblum, G.N.: Securing Computation against Continuous Leakage. In: Rabin, T. (ed.) CRYPTO 2010. LNCS, vol. 6223, pp. 59–79. Springer, Heidelberg (2010)
17. Groth, J.: Simulation-Sound NIZK Proofs for a Practical Language and Constant Size Group Signatures. In: Lai, X., Chen, K. (eds.) ASIACRYPT 2006. LNCS, vol. 4284, pp. 444–459. Springer, Heidelberg (2006)
18. Groth, J., Sahai, A.: Efficient Non-interactive Proof Systems for Bilinear Groups. In: Smart, N.P. (ed.) EUROCRYPT 2008. LNCS, vol. 4965, pp. 415–432. Springer, Heidelberg (2008)
19. Ishai, Y., Sahai, A., Wagner, D.: Private Circuits: Securing Hardware against Probing Attacks. In: Boneh, D. (ed.) CRYPTO 2003. LNCS, vol. 2729, pp. 463–481. Springer, Heidelberg (2003)
20. Juma, A., Vahlis, Y.: Protecting Cryptographic Keys against Continual Leakage. In: Rabin, T. (ed.) CRYPTO 2010. LNCS, vol. 6223, pp. 41–58. Springer, Heidelberg (2010)
21. Kiltz, E., Pietrzak, K.: Leakage Resilient ElGamal Encryption. In: Abe, M. (ed.) ASIACRYPT 2010. LNCS, vol. 6477, pp. 595–612. Springer, Heidelberg (2010)
22. Lewko, A., Rouselakis, Y., Waters, B.: Achieving Leakage Resilience through Dual System Encryption. In: Ishai, Y. (ed.) TCC 2011. LNCS, vol. 6597, pp. 70–88. Springer, Heidelberg (2011)
23. Lewko, A., Lewko, M., Waters, B.: How to leak on key updates. In: To Appear at STOC 2011 (2011)

24. Micali, S., Reyzin, L.: Physically Observable Cryptography (Extended Abstract). In: Naor, M. (ed.) TCC 2004. LNCS, vol. 2951, pp. 278–296. Springer, Heidelberg (2004)
25. Okamoto, T.: Provably Secure and Practical Identification Schemes and Corresponding Signature Schemes. In: Brickell, E.F. (ed.) CRYPTO 1992. LNCS, vol. 740, pp. 31–53. Springer, Heidelberg (1993)
26. Pedersen, T.P.: Non-interactive and Information-Theoretic Secure Verifiable Secret Sharing. In: Feigenbaum, J. (ed.) CRYPTO 1991. LNCS, vol. 576, pp. 129–140. Springer, Heidelberg (1992)
27. Pietrzak, K.: A Leakage-Resilient Mode of Operation. In: Joux, A. (ed.) EUROCRYPT 2009. LNCS, vol. 5479, pp. 462–482. Springer, Heidelberg (2009)
28. Rao, A.: An exposition of bourgain's 2-source extractor. Electronic Colloquium on Computational Complexity (ECCC) 14(034) (2007)
29. Sahai, A.: Non-malleable non-interactive zero knowledge and adaptive chosen-ciphertext security. In: FOCS, pp. 543–553 (1999)
30. Shacham, H.: A cramer-shoup encryption scheme from the linear assumption and from progressively weaker linear variants. Cryptology ePrint Archive, Report 2007/074 (2007), `http://eprint.iacr.org/`