Leakage-Resilient Cryptography From the Inner-Product Extractor

Stefan Dziembowski[1]* 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.

* 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.

1 Introduction

Recently, 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., [30,16,17,21,10,6,29] and many more). By now we have seen new constructions for many important cryptographic primitives such as digital signature and public key encryption 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 [16], 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).

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.

1.1 Our Leakage Model

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. The assumption that different parts of the memory leak independently – originally put forward by Micali and Reyzin [30] as the physical axiom "only computation leaks information" – constitutes a spatial restriction of the leakage and has been used in several works [30,16,33,26,17] (and more). 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). One may object now that leakage is a global phenomenon. Indeed, this is true and many important leakage functions, for instance, the power consumption of a device modeled by the Hamming weight, are rather a global function of the computation's intermediate values. We would like to emphasize, however, that many relevant global leakage functions can be computed from just local leakages. This is not only true for the prominent Hamming weight leakage, but more generally, for *any affine* leakage function. Hence, independent leakages still allow for a broad class of practically relevant *global* leakage functions.

Recently, it has been pointed out that the independent leakage assumption does not capture so-called "coupling effects" that may occur between gates and add a non-linear term to the leakage function [38]. As such coupling effects typically occur between gates, it may contradict locality assumptions that are made at a very low architectural level – e.g., if we assume that each gate of the computation leaks independently. In our case, coupling effects may be much less dangerous, as we make the assumption of independent leakages at a much higher architectural level, i.e., we just assume that two parts of the memory leak independently.

ON LEAK-FREE COMPONENTS. We use a biased source of randomness that outputs 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. As in earlier works that made use of leak-free components [20,18,25,21], we require that our component leaks from its outputs, but the leakage function cannot view the internals of the ongoing computation. More concretely, 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 $\mathcal{O}$ 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 $\sum_i A_i \cdot B_i = 0$.[3] Second, we use $\mathcal{O}$ in a very limited way, namely, we need it only when we refresh the secret key (cf. Section 1.3 for further discussion on this). Finally, $\mathcal{O}$ does not take any inputs, and hence its computation is completely oblivious of the actual computation (e.g., encryption or signing) that is carried out by the device. Moreover, this property gives rise to an alternative implementation as pointed out by Faust et al. [18]: instead of assuming leak-free computation, we can have $\mathcal{O}$ simply read its output one by one from a pre-computed list. Thus, it suffices to have leak-proof one-time storage instead of leak-proof computation. This may be an option if the computation is performed only a bounded number of times.

We would also like to emphasize that a leak-free component that does not take any inputs is much harder to attack by side-channel analysis, as successful attacks usually require some choice (or at least knowledge) over the inputs that are taken by the device. Also, we would like to stress that such a component can be tested (regarding its side-channel resistance) independently of the environment in which it is going to be used eventually.

We will further discuss our model and how it relates to earlier work in the next sections.

1.2 Leakage Resilient Implementation of Standard Cryptographic Schemes

As outlined in the introduction, many recent works in leakage resilient cryptography design new cryptographic schemes that remain secure against certain (often very broad) classes of leakages. While the design of new cryptographic schemes with built-in leakage resilience is a very important research direction, we believe that most current results suffer from one of the following weaknesses:

1. For many security related tasks such as authentication or confidentiality of data, certain cryptographic schemes have become part of widely used international standards. Even if desirable, it is unlikely that in near future these standards will be adjusted to include recent scientific progress from leakage resilient cryptography.
2. Even though by now schemes that remain secure in surprisingly powerful leakage models have been proposed, they are often very complicated, rely on non-standard complexity assumptions and are often rather inefficient.

In this work, we take a different approach and propose general techniques that allow to implement efficiently standard cryptographic schemes that remain provably secure in the above described leakage model. Before

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

we given an overview of our contributions in the next section, we discuss some related literature that has dealt with similar questions before.

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 and dates back to the work of Ishai et al. [24]. They propose a circuit compiler that transforms any Boolean circuit into one that is secure against an adversary who reads off the values from a bounded number of wires. This work has recently been extended by Faust et al. [18] to consider larger classes of computationally bounded leakages – e.g., the leakage is modeled by an AC0 circuit that takes as input the entire state of the circuit. While these schemes only achieve security against restricted function classes (either probing attacks or AC0), the works of Juma and Vahlis [25] and Goldwasser and Rothblum [21] study the question whether any computation can be implemented in a way that withstands arbitrary polynomial-time computable leakages. As a building block these schemes use a public-key encryption scheme and essentially 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 two latter works require the independent leakage assumption: while Juma and Vahlis use – similar to us – a device whose memory is divided into two parts, Goldwasser and Rothblum assume that essentially every individual gate (of the original circuit) leaks independently.

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 [26] takes on this efficiency challenges and shows that certain standard cryptographic schemes can be implemented *efficiently* in a leakage resilient way. The authors propose an efficient "bilinear version" of the ElGamal encryption scheme that is CCA1-secure even if the computation from the decryption process leaks information. The main weakness of this work is that the security proof is given in the generic group model. That is, group elements are modeled as uniformly random values and to perform a group operation or compute a bilinear map one has to query an oracle. Such a proof only implies that there are no "generic" leakage attacks that would work on any underlying group, i.e., it shows security against any attack that is independent of the representation of the group elements.

1.3 Our contribution

We continue this line of research and show a generic method to implement various standard cryptographic schemes that are provably secure in the above described leakage model. More precisely, we propose efficient and simple implementations of the Okamoto authentication/signature scheme and show that standard security properties (such as existentially unforgeability) carry over under continuous leakage attacks. Moreover, we prove that a simple variant of the ElGamal encryption scheme is CCA2 secure in the random oracle model even if the decryption process leaks continuously. 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. [9] introduced the notion of leakage resilient storage (LRS). Such a scheme 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 [8]. Essentially, 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 avoid such attacks and propose an efficient refreshing scheme for the inner product based encoding.

To make such a refreshing secure against continuous leakage attacks, we divide 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 $\mathsf{Refresh}$ takes as input (L, R) and outputs a fresh encoding (L', R') of S. The computation of $\mathsf{Refresh}$ 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 could also hope that this scheme remains secure in the presence of continuous leakage attacks. Perhaps counterintuitive, we show (cf. Appendix D.1) that this simple protocol can be completely broken if the leakage function can be evaluated on (L, X, Y) and (R, X, Y). To prevent this attack, we need 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 of 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 in a way that withstands leakage attacks. 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, and hence, in a leakage resilient way, we make use of 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 our implementation is the *first* construction of a leakage resilient encryption scheme that allows the leakage to *depend* on the target ciphertext, i.e., we allow the adversary to obtain leakage *after* seeing the target ciphertext. We solve this obstacle by exploiting the independent leakage assumption, i.e., we encode the secret key as (L, R) and carry out the computation using the above described protocol for secure exponentiation. This together with our protocol for refreshing the inner product encoding yields a leakage resilient implementation of an ElGamal-based encryption scheme under the DDH assumption. 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. Moreover, notice that the Fiat-Shamir heuristic is the only place where the random oracle assumption is used, which in particular means that we do not make any additional restrictions on the class of leakage functions, as, e.g., leakage functions can query the random oracle at any point.

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 key in this field, 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*.

ON THE EFFICIENCY OF OUR SCHEMES. For a statistical security parameter n, we increase the secret key size and computation complexity by a factor of n compared to the underlying schemes. That is, instead of using two group elements as secret key, our implementations need to store $3n$ group elements. Moreover, if the underlying scheme needs to carry out 2 exponentiations our leakage resilient implementations requires $3n$ exponentiations. The public key size of both the underlying cryptosystem and our implementation is identical. As already discussed above, an alternative for implementing standard cryptosystems is via leakage resilient circuit compilers. These are, however, considerably less efficient due to the following three reasons:

1. Circuit compilers typically consider the question of how to implement Boolean circuits in a leakage resilient way. Hence, to implement, e.g., the Okamoto authentication, one first needs to "compile" the scheme into a Boolean circuit and only then implements it using the techniques from the leakage resilient circuit compilers. Our techniques are more direct and do not require such a detour via Boolean circuits.
2. Non-linear operations (e.g., AND that is heavily needed for exponentiations and multiplications) are significantly more expensive than the implementation of linear operations. More concretely, for any non-linear operation the efficiency loss is *quadratic* in the security parameter.

3. The current circuit compilers that protect against arbitrary polynomial-time computable leakage [25,21] either carry out the complete computation using a fully homomorphic encryption scheme, or they encrypt every bit on a wire with a new secret/public key pair. In contrast, we do not use any additional public key cryptography.

Of course, circuit compilers have an important advantage over our work. While we focus on certain cryptographic schemes, leakage resilient circuit compilers allow to implement *any* computation in a leakage resilient way. Hence, they may be used to implement, e.g., the AES in a leakage resilient way.

We would like to point out that the scheme of Kiltz and Pietrzak [26] is considerably more efficient than our scheme and results only into a constant loss in efficiency (compared to standard ElGamal). However, this comes at the price that the security proof is done in the generic group model. Moreover, we would like to mention that in this work we settle a question raised in [26]. We propose the *first* encryption scheme that is CCA2 secure in the presence of leakage attacks.

ON THE LEAK-FREE COMPONENT. In our work, we require access to correlated randomness that is sampled in a leak-free way. It is interesting to see how our requirement relates to earlier work that made use of similar assumptions [20,18,25,21]. We focus in the following on the results of Juma and Vahlis [25] and Goldwasser and Rothblum [21], which both work in a similar leakage model as we do. In the work of Juma and Vahlis, the leak-free component has to sample a public/secret key pair for a fully homomorphic encryption scheme together with two ciphertexts C and C' that are encryptions of 0. On the other hand, in the work of Goldwasser and Rothblum the leak free component takes as input a public key and a mode of operation, and either outputs a fresh encryption of 0 or an encryption of a random bit b. It is easy to see that the computation that is carried out by our leak-free component $\mathcal{O}$ is considerably simpler as it just involves a small number of simple operations. Moreover, we would like to emphasize that our component is *only* needed to refresh the secret key (and not for signing or decryption), while in Goldwasser and Rothblum the leak-free component is required $O(k)$ times for every NAND gate in the original circuit (here k is the security parameter). On the positive side, we would like to note, that the entire output of the leak-free component in [21] can leak jointly, while in our case the two parts A and B have to leak independently. The same is true for the component of Juma and Vahlis where the secret key cannot leak jointly with the ciphertexts.

COMPARISON TO OTHER RELATED WORK. We would like to mention that in a series of important recent works [10,6,29,28,5] *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 and rather inefficient. Furthermore, they rely on non-standard assumptions (e.g., subgroup decisional assumption), or only allow a small constant fraction (or even logarithmic fraction) of key leakage during the update process. Moreover, we would like to emphasize that the goal of our work is different: we are interested in the question whether *standard* cryptosystems can be efficiently implemented in a leakage resilient way. For instance, our method for storing and refreshing a secret key works generically and independently of the proposed scheme. We would like to note that Dodis et al. [12] recently 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. It is an interesting research question whether our results can be combined with the storage and refreshing scheme of [12].

2 Preliminaries

For a natural number n the set $\{1, \ldots, n\}$ will be denoted $[n]$. 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. A vector 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 $(0, \ldots, 0)$ 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 $C = (g^{A_1}, \ldots, g^{A_n})$. Furthermore, let C^B be the vector $(g^{A_1 B_1}, \ldots, g^{A_n B_n})$. In the following, we will often omit to explicitly specify the security parameter k, and assume that the information theoretic security parameter n is a function of k.

2.1 Basic Definitions from Information Theory

We denote with U_n the random variable with distribution uniform over $\{0,1\}^n$. Let X_0, X_1 be random variables distributed over $\mathcal{X}$ and let 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. For random variables X over $\mathcal{X}$ and Y over $\mathcal{Y}$ let $d(X) := \Delta(X; U)$ and $d(X|Y) := \Delta(X; U|Y)$ (where U is uniform and independent from Y). It is easy to verify that for any event $\mathcal{E}$ we have

$$d(X|Y) = \frac{1}{2} \sum_{x\in\mathcal{X}, y\in\mathcal{Y}} |\Pr[X = x \wedge Y = y \mid \mathcal{E}] - \Pr[Y = y \mid \mathcal{E}] / |\mathcal{X}||. \tag{1}$$

For random variables X, Y we define the min-entropy of X as $\mathsf{H}_\infty(X) = -\log \max_{x\in\mathcal{X}} \Pr[X = x]$ and the average min-entropy X given Y as [13]

$$\tilde{\mathsf{H}}_\infty(X|Y) := -\log\left(\mathbb{E}_{y\leftarrow Y}\left[2^{-\mathsf{H}_\infty(X|Y=y)}\right]\right).$$

We prove some basic information theoretic lemmata that will be used throughout the paper in Appendix A.

2.2 Leakage Model

As described in the introduction, in this work we will 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 $f_i : \{0,1\}^s \rightarrow \{0,1\}^{\lambda_i}$. For the ith query the oracle replies with $f_i(x_i)$ and we say that in this case the adversary $\mathcal{A}$ *retrieved the value* $f_i(x_i)$. The only restriction is that in total the adversary does not retrieve more than λ bits from each L and R. In the following, let $Out(\mathcal{A}, \Omega(L, R))$ be the output of $\mathcal{A}$ at the end of this game. Without loss of generality, we assume that $Out(\mathcal{A}, \Omega(L, R)) := (f_1(x_1), \ldots, f_t(x_t))$.

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), and this computation leaks information to the adversary, then we need a way to describe the leakage from such computation. As discussed in the introduction, we do this in form of a two-party protocol $\Pi = (P_{\mathsf{L}}, P_{\mathsf{R}})$, which is executed between the two parties P_{L} and P_{R} and an adversary is allowed to obtain partial information (the leakage) from the internal state of the players.

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 sends 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 and of course are not given to the adversary.

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 by her. Indeed, in our setting, we allow the adversary to learn such leakages. However, to simplify exposition, we consider actually 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 during the execution of $\Pi(L, R)$ 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 $\mathcal{A} \leftrightarrows (\Pi(L, R) \to (L', R'))$.

As we are interested in the continuous leakage setting, we will mostly consider an adversary that runs in many executions of $\mathcal{A} \leftrightarrows (\Pi(L, R) \to (L', R'))$. For the ith execution of the protocol $\Pi(L^{i-1}, R^{i-1})$, we will write $\mathcal{A} \leftrightarrows \big(\Pi(L^{i-1}, R^{i-1}) \to (L^i, R^i)\big)$, 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 $\mathcal{A} \leftrightarrows \big(\Pi(L^{i-1}, R^{i-1}) \to (L^i, R^i)\big)$, 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. More precisely, for the ith execution of $\mathcal{A} \leftrightarrows \big(\Pi(L^{i-1}, R^{i-1}) \to (L^i, R^i)\big)$, we let the adversary interact with the leakage oracle $\Omega((L^{i-1}, \rho^i_{\mathsf{L}}, M^i_{\mathsf{L}}); (R^{i-1}, \rho^i_{\mathsf{R}}, M^i_{\mathsf{R}}))$, where $(\rho^i_{\mathsf{L}}, \rho^i_{\mathsf{R}})$ denotes the randomness used during this execution, and $(M^i_{\mathsf{L}}, M^i_{\mathsf{R}})$ denotes the messages that the players send and receive. 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.

2.3 Leakage-resilient Storage

Davi et al. [9] recently introduced the notion of leakage-resilient storage (LRS) $\Phi = (\mathsf{Encode}, \mathsf{Decode})$. An LRS 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. One of the constructions that the authors propose uses two source extractors and can be shown to be secure in the independent leakage model. 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

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

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

In this paper, 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 $L \leftarrow \mathbb{F}^n \backslash \{(0^n)\}$ 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 [8,35] and appears in Appendix C.

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 $(0.3 \cdot |\mathbb{F}^n|, negl(n))$-secure, for some negligible function $negl$.*

Proof (of Corollary 1). In Lemma 1 set $\delta = 0.10$ and $\gamma := |\mathbb{F}|^{-n/10}$. It is easy to see that for such a choice of parameters ϵ (as defined in Lemma 1) is negligible. We also have that λ is equal to $0.4n \log |\mathbb{F}| - n \log |\mathbb{F}| / 10$, which is equal to $0.3 \cdot |\mathbb{F}^n|$. □

3 Leakage-Resilient Refreshing of LRS

An obvious drawback of an LRS is the fact that the *total* leakage from the memory is bounded by some small constant λ. Indeed, if an adversary can continuously learn information from the encoded secret $(L, R) \leftarrow \mathsf{Encode}(S)$, then after a few observations L and R are completely known, and the adversary can trivially break the LRS. To solve this problem, we need a method of "pumping" new randomness into the encoding. To this end, we show how to securely *refresh* an LRS encoding. More precisely, we will develop a probabilistic protocol $(L', R') \leftarrow \mathsf{Refresh}(L, R)$ that securely refreshes (L, R), 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.

For a secret S and a leakage-resilient storage $\Phi = (\mathsf{Encode}, \mathsf{Decode})$ with message space $\mathcal{M}$, the refreshing of an encoded secret $(L, R) \leftarrow \mathsf{Encode}(S)$ is done by a two-party protocol $(L', R') \leftarrow \mathsf{Refresh}(L, R)$. 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 there is no interaction between the refreshing protocol and the user of the protocol. In other words: the only way in which the adversary can "interact" with the protocol is via the leakage oracle.

For *correctness*, we require that $\mathsf{Decode}(L, R) = \mathsf{Decode}(L', R')$, i.e., the refreshed encoding decodes to the stored secret S. 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, an 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: $\mathcal{A} \leftrightarrows \big(\Pi(L^{i-1}, R^{i-1}) \to (L^i, R^i)\big)$.
3. Return whatever $\mathcal{A}$ outputs.

The experiment outputs whatever $\mathcal{A}$ outputs after interacting with Π for ℓ iterations (without loss of generation we can 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 2.3. 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). For ease of notation, we will often omit to specify the $\Phi_{\mathbb{F}}^{n,m}$ when talking about the refreshing protocol $(\mathsf{Refresh}_{\mathbb{F}}^{n,m}, \Phi_{\mathbb{F}}^{n,m})$ and just write $\mathsf{Refresh}_{\mathbb{F}}^{n,m}$ or $\mathsf{Refresh}$ when clear otherwise.

As already mentioned in the introduction, we will assume that the players have access to a non-uniform source of randomness. More precisely, they will 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 rank (these checks were introduced only to facilitate the proof and only occur with very small probability: cf. Lemma 3). 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 generation of X and Y is done in an "oblivious" way: the player P_{R} will learn X and the player P_{L} will learn Y, *but* X will be secret for P_{R} and Y will be secret for P_{R}. In Appendix D.1, we show that a simpler, and more natural protocol, in which X and Y are generated by P_{L} and P_{R} (resp.) and communicated to each other is actually *insecure*. We now first show correctness of our protocol from Figure 1.

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

Proof. As we assume that A and B have a full rank, and we check (in Steps 1 and Step 4) that L and R' have full rank, by Lemma 15 in Appendix B we can sample a random solution M and $\tilde{M}$ for the equations in Steps 2 and 5 (resp.). The rest of the proof follows by simple linear algebra:

$$L' \cdot R' = \left(L + \tilde{A} \cdot \tilde{M}\right) \cdot R' \tag{2}$$

$$= L \cdot R' + \tilde{A} \cdot \tilde{M} \cdot R' \tag{3}$$

$$= L \cdot R' + \tilde{A} \cdot \tilde{B} \tag{4}$$

$$= L \cdot R', \tag{5}$$

where (2) follows from the construction of the protocol (Step 6), Eq. (3) follows from the linearity of the inner product, Eq. (4) comes from the fact that $\tilde{M} \cdot R' = \tilde{B}$ (cf. Step 5), and (5) follows from $\tilde{A} \cdot \tilde{B} = (0, \ldots, 0)$.

Similarly (5) is equal to $L \cdot (R + M \cdot B) = L \cdot R + L \cdot M \cdot B$. By a similar reasoning as above we get that $L \cdot M \cdot B = A \cdot B = (0, \ldots, 0)$. Hence, (5) is equal to $L \cdot R$, which decodes to S. This finishes the proof. $\square$

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, when the protocol is executed. Note that by Lemma 15 in Appendix B, sampling the random matrices in Steps 2 and 5 can be done efficiently.

What remains is to show that protocol $\mathsf{Refresh}_{\mathbb{F}}^{n,m}$ from Figure 1 satisfies Definition 1. This is done in the following theorem.

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* $\epsilon' := 2\ell \,|\mathbb{F}|^m \,(3\,|\mathbb{F}|^m \,\epsilon + m\,|\mathbb{F}|^{-n-1})$.

To prove 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:

$$\Delta(\mathsf{Exp}_{\mathsf{Refresh}}(\mathcal{A}, S, \ell); \mathsf{Exp}_{\mathsf{Refresh}}(\mathcal{A}, S', \ell)) \leq 2\ell \,|\mathbb{F}|^m \,(3\,|\mathbb{F}|^m \,\epsilon + m\,|\mathbb{F}|^{-n-1}). \tag{6}$$

This will be proven using the standard technique called the "hybrid argument" by creating a sequence of "hybrid distributions" – denoted by $\mathbf{Hyb}_i(\mathcal{A}, S, S', \ell)$ and $\widetilde{\mathbf{Hyb}}_i(\mathcal{A}, S, S', \ell)$. We will show that the first distribution in this sequence is statistically very close to $\mathsf{Exp}_{\mathsf{Refresh}}(\mathcal{A}, S, \ell)$, and the last one is very close to $\mathsf{Exp}_{\mathsf{Refresh}}(\mathcal{A}, S', \ell)$. Moreover, each two consecutive distributions in the sequence will be very 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.

Following the notation of $\mathsf{Exp}_{\mathsf{Refresh}}$, we denote in the hybrid experiments $\mathbf{Hyb}_i$ and $\widetilde{\mathbf{Hyb}}_i$ the input to the jth execution of $\mathsf{Refresh}_{\mathbb{F}}^{n,m}$ by (L^{j-1}, R^{j-1}), and its output by (L^j, R^j). We can now define the hybrid distributions as follows. For each $i \in [n]$ let $\mathbf{Hyb}_i(\mathcal{A}, S, S', \ell)$ be defined in the same way as

$\mathsf{Exp}_{\mathsf{Refresh}}(\mathcal{A}, S, \ell)$ (i.e., run ℓ iterations of the protocol from Fig. 1) except that in the ith iteration in Step 2 for the refreshing of P_{R}'s share, instead of using the oracle $\mathcal{O}$ we use an oracle $\mathcal{O}'$ that samples (A, B) from the set

$$\{(A, B) \in \mathbb{F}^n \times \mathsf{NonSing}^{n\times m}(\mathbb{F}) : A \neq (0, \ldots, 0) \text{ and } A \cdot B = S' - S\}.$$

Observe that this means that for $j \leq i-1$ we have $L^j \cdot R^j = S$ since we proceed as in $\mathsf{Exp}_{\mathsf{Refresh}}(\mathcal{A}, S, \ell)$ for the first j executions of $\mathsf{Refresh}_{\mathbb{F}}^{n,m}$. As in the ith iteration for refreshing the share of P_{R} we use the oracle $\mathcal{O}'$ by simple calculation we get that

$$L^{i-1} \cdot R^i = L^{i-1} \cdot (R^{i-1} + X) = L^{i-1} \cdot (R^{i-1} + M \cdot B) = S + L^{i-1} \cdot M \cdot B = S + A \cdot B = S'.$$

Since from then on we continue with using the oracle $\mathcal{O}$, we get for $j \geq i$ that $L^j \cdot R^j = S'$. Similarly, let $\widetilde{\mathbf{Hyb}}_i(\mathcal{A}, S, S', \ell)$ be defined in the same way as $\mathsf{Exp}_{\mathsf{Refresh}}(\mathcal{A}, S, \ell)$ except that in the ith iteration of $\mathsf{Refresh}_{\mathbb{F}}^{n,m}$ for refreshing the share of P_{R} we sample $(\tilde{A}, \tilde{B})$ from the oracle $\mathcal{O}'$.

Before we show in Lemma 4 that these hybrid distributions are indeed close, let us prove a simple lemma about the event $\mathcal{Q}$ that the players abort in the protocol. Recall that in Figure 1 P_{L} or P_{R} may abort the execution of the protocol, in case that L (cf. Step 1) or R' (cf. Step 4) do not have full rank. To preserve correctness of the protocol, we reconstruct the secret in case of abortion. Obviously, in such a case the distance between the distributions will not be small anymore. Hence, we need to bound the probability that such an event occurs.

Lemma 3 (Probability of abort). *For any S, we have* $\Pr[\mathcal{Q}] \leq 2\ell m \cdot |\mathbb{F}|^{m-n}$ *in* $\mathsf{Exp}_{\mathsf{Refresh}}(\mathcal{A}, S, \ell)$.

Proof. For each j we define the event $\mathcal{E}(j)$ that the players abort in the jth iteration. This means that either L^{j-1} or R^j does not have full rank. Note that (L^0, R^0) are chosen uniformly at random subject to the constraint that they encode the value S. By the construction of our protocol, it is easy to verify that this carries over to each L^j and R^j. In other words for each $\mathcal{E}(j)$ we can view L^{j-1} as being chosen uniformly at random and each of the m columns of R^j as being chosen from an $n-1$ dimensional subspace of $\mathbb{F}^n$. Hence, from Lemma 16 the probability that $(R^j)^{\mathsf{T}}$ does not have a full rank is at most $m \cdot |\mathbb{F}|^{m-n}$. This carries over to R^j by Lemma 18 from the appendix. Clearly, L^j does not have a full rank only if it is equal to $(0, \ldots, 0)$, which happens with probability $|\mathbb{F}|^{-n} \leq m \cdot |\mathbb{F}|^{m-n}$. Therefore, we get from the union-bound that for each j we have $\Pr[\mathcal{E}(j)] \leq 2m \cdot |\mathbb{F}|^{m-n}$ and by applying again the union-bound, we get that $\Pr[\mathcal{Q}] \leq 2\ell \cdot m \cdot |\mathbb{F}|^{m-n}$. □

It is easy to see that the above Lemma 3 works also in case of the hybrid experiments, and hence for both $\mathbf{Hyb}_i(\mathcal{A}, S, S', \ell)$ and $\widetilde{\mathbf{Hyb}}_i(\mathcal{A}, S, S', \ell)$, we have

$$\Pr[\mathcal{Q}] \leq 2\ell m \cdot |\mathbb{F}|^{m-n}.$$

We now have the main technical lemma of this section.

Lemma 4. *For the parameters $\ell, n, m, \lambda, \epsilon, \mathbb{F}$ as in Theorem 1, for every $(\lambda/2 - 1)$-limited $\mathcal{A}$ and every S, S' we have the following:*

1. $\Delta\left(\mathsf{Exp}_{\mathsf{Refresh}}(\mathcal{A}, S, \ell)\, ;\, \mathbf{Hyb}_1(\mathcal{A}, S, S', \ell) \,|\, \overline{\mathcal{Q}}\right) \leq 2\,|\mathbb{F}|^{2m}\,\epsilon$,
2. *for every $i = 1, \ldots, \ell$ it holds that* $\Delta\left(\mathbf{Hyb}_i(\mathcal{A}, S, S', \ell)\, ;\, \widetilde{\mathbf{Hyb}}_i(\mathcal{A}, S, S', \ell) \,|\, \overline{\mathcal{Q}}\right) \leq 2\,|\mathbb{F}|^{2m}\,\epsilon$,
3. *for every $i = 1, \ldots, \ell - 1$ it holds that* $\Delta\left(\widetilde{\mathbf{Hyb}}_i(\mathcal{A}, S, S', \ell)\, ;\, \mathbf{Hyb}_{i+1}(\mathcal{A}, S, S', \ell) \,|\, \overline{\mathcal{Q}}\right) \leq 2\,|\mathbb{F}|^{2m}\,\epsilon$,
4. $\Delta\left(\widetilde{\mathbf{Hyb}}_\ell(\mathcal{A}, S, S', \ell)\, ;\, \mathsf{Exp}_{\mathsf{Refresh}}(\mathcal{A}, S', \ell) \,|\, \overline{\mathcal{Q}}\right) \leq 2\,|\mathbb{F}|^{2m}\,\epsilon$.

Proof. We show only the proof of Point 3. The proof of the remaining points is analogous. The table below represents the inner products of L^j and R^j in the hybrids $\widetilde{\mathbf{Hyb}}_i(\mathcal{A}, S, S', \ell)$ and $\mathbf{Hyb}_{i+1}(\mathcal{A}, S, S', \ell)$.

$j =$	1		2		$\cdots$	i		$i+1$		$i+2$		$\cdots$	ℓ
L^j	L^0	L^0	L^1	L^1	$\cdots$	L^{i-1}	L^{i-1}	L^i	L^i	L^{i+1}	L^{i+1}	$\cdots$	L^ℓ
R^j	R^0	R^1	R^1	R^2	$\cdots$	R^{i-1}	R^i	R^i	R^{i+1}	R^{i+1}	R^{i+2}	$\cdots$	R^ℓ
$(*)$ in $\mathbf{Hyb}_{i+1}(\mathcal{A}, S, S', \ell)$	S	S	S	S	$\cdots$	S	S	S	S'	S'	S'	$\cdots$	S'
$(**)$ in $\widetilde{\mathbf{Hyb}}_i(\mathcal{A}, S, S', \ell)$	S	S	S	S	$\cdots$	S	S	S'	S'	S'	S'	$\cdots$	S'

(7)

It is easy to see that the only difference between $\widetilde{\mathbf{Hyb}}_i(\mathcal{A}, S, S', \ell)$ and $\mathbf{Hyb}_{i+1}(\mathcal{A}, S, S', \ell)$ is that in the former $L^i \cdot R^i = S'$ and in the latter $L^i \cdot R^i = S$. We will show that for any $(\lambda/2 - 1)$-limited adversary $\mathcal{A}$

$$\Delta\left(\widetilde{\mathbf{Hyb}}_i(\mathcal{A}, S, S', \ell)\,;\,\mathbf{Hyb}_{i+1}(\mathcal{A}, S, S', \ell)|\overline{\mathcal{Q}}\right) \leq 2\,|\mathbb{F}|^{2m}\,\epsilon. \tag{8}$$

We prove this by contradiction. Suppose there exists an adversary $\mathcal{A}$ for which Eq. (8) does not hold, then we construct a λ-limited adversary $\mathcal{S}$, that we call the *simulator*, which will be able to break the (λ, ϵ)-security of $\Phi_{\mathbb{F}}^{n,m}$. $\mathcal{S}$ runs $\mathcal{A}$ as a sub-routine and simulates its environment according to either $\widetilde{\mathbf{Hyb}}_i(\mathcal{A}, S, S', \ell)$ or $\mathbf{Hyb}_{i+1}(\mathcal{A}, S, S', \ell)$ by just having access to its target oracle $\Omega(L, R)$. We will show that in case that $(L, R) \leftarrow \mathsf{Encode}_{\mathbb{F}}^{n,m}(S')$ the simulation of $\mathcal{A}$ is as $\widetilde{\mathbf{Hyb}}_i(\mathcal{A}, S, S', \ell)$ (cf. (**) in Table 7), while in case of $(L, R) \leftarrow \mathsf{Encode}_{\mathbb{F}}^{n,m}(S)$ the simulation is as in $\mathbf{Hyb}_{i+1}(\mathcal{A}, S, S', \ell)$ (cf. (*) in Table 7). To this end, $\mathcal{S}$ "plugs" the encoding (L, R) from its target oracle into (L^i, R^i), and uses access to its target oracle to simulate the leakage from $\mathsf{Exp}_{\mathsf{Refresh}}$ that depends on (L^i, R^i). One main difficulty is that $\mathsf{Exp}_{\mathsf{Refresh}}$ may run for many iterations, hence, allowing the adversary to learn a large amount of information, while on the other hand $\mathcal{S}$ only can retrieve up to λ bits from its target oracle. To solve this problem $\mathcal{S}$ will simulate most leakages "off-line", i.e., without using access to its target oracle. For the ith and $(i+1)$th execution, however, $\mathcal{S}$ will use access to its target oracle to make the leakages from these rounds consistent with the "off-line" leakages. Eventually, the simulator will output whatever $\mathcal{A}$ outputs, or *abort*, in which case it outputs $\perp$. We give the details below.

The adversary $\mathcal{S}$ will simulate $\mathcal{A}$ in the following way.

Pre-processing I: generating the L^j and R^j variables except for (L^i, R^i): He sets $(L^0, R^0) \leftarrow \mathsf{Encode}_{\mathbb{F}}^{n,m}(S)$. He performs $(i-1)$ iterations of the refreshing procedure, which results into $(L^0, R^0), \ldots, (L^{i-1}, R^{i-1})$ and all the messages that are generated during the execution of the protocol from Figure 1. He then sets $(L^{i+1}, R^{i+1}) \leftarrow \mathsf{Encode}_{\mathbb{F}}^{n,m}(S')$ and performs the remaining executions of the refreshing procedure. He stores all the variables used in these executions. Note, that the only thing that the simulator misses for the simulation of $\mathcal{A}$ are the variables used in the ith and $(i+1)$th iteration.

Pre-processing II: simulating the first $i - 1$ iterations: The simulator $\mathcal{S}$ starts $\mathcal{A}$ and simulates him on the variables that he generated previously for the first $(i - 1)$ iterations of the refreshing protocol. This can be done easily, as all the variables for the simulation are known.

Leakage of $(L^i, R^i) \leftarrow \mathsf{Refresh}_{\mathbb{F}}^{n,m}(L^{i-1}, R^{i-1})$ in the ith iteration: Now, the simulator simulates the leakage from phase i. Note that he does not know (L^i, R^i) – since it is equal to (L, R) he has access to it only via the leakage oracle. We first describe how he can do it if he knows (L^i, R^i) completely and later argue that it can be done also just by leaking limited amount of data from L^i and R^i.

First, $\mathcal{S}$ simulates the refreshing of P_{R}'s share. He sets

$$X := R^i - R^{i-1}. \tag{9}$$

He checks if $L^{i-1} \cdot X = (0, \ldots, 0)$ (or, equivalently: if $L^{i-1} \cdot R^i = S$). If not, then he aborts (let $\mathcal{Q}_1$ denote this event). Otherwise he chooses $M \in \mathsf{NonSing}^{n \times n}(\mathbb{F})$ uniformly at random and computes

$$B := M^{-1} \cdot X \tag{10}$$

He also computes

$$A := L^{i-1} \cdot M \tag{11}$$

Note that $A \cdot B := L^{i-1} \cdot M \cdot M^{-1} \cdot X = L^{i-1} \cdot X = (0, \ldots, 0)$.
Now, to simulate the refreshing of the share of P_{L}, he chooses a random $\tilde{M} \leftarrow \mathsf{NonSing}^{n \times n}(\mathbb{F})$. He computes

$$Y := L^i - L^{i-1}. \tag{12}$$

He sets:

$$\tilde{A} := Y \cdot \tilde{M}^{-1}. \tag{13}$$

He also computes

$$\tilde{B} := \tilde{M} \cdot R^i. \tag{14}$$

Note that we have

$$\tilde{A} \cdot \tilde{B} = \tilde{Y} \cdot \left(\tilde{M}\right)^{-1} \cdot \tilde{M} \cdot R^i = \tilde{Y} \cdot R^i = (L^i - L^{i-1}) \cdot R^i = \begin{cases} (0, \ldots, 0) & \text{if } L^i \cdot R^i = S \\ S' - S & \text{if } L^i \cdot R^i = S'. \end{cases} \tag{15}$$

The simulator also checks if the values $A, B, \tilde{A}$ and $\tilde{B}$ have a full rank (so that they look like generated by the oracle $\mathcal{O}$). If not, then he aborts. Call this event $\mathcal{Q}_2$.
Now, the simulator has all the variables needed to simulate the game between $\mathcal{A}$ and

$$\Omega\left((L^{i-1}, A, M, \tilde{A}, \tilde{M})\ ;\ (R^{i-1}, B, M, \tilde{B}, \tilde{M})\right).$$

The only problem, that we did not address so far, is that in reality the simulator has access to L^i and R^i only via the leakage oracle $\Omega(L; R)$. The main observation is now that the above simulation is done in such a way that the leakage function can compute (a) $(L^{i-1}, A, M, \tilde{A}, \tilde{M})$ just from $L^i := L$ and (b) $(R^{i-1}, B, M, \tilde{B}, \tilde{M})$ just from $R^i := R$.
First, observe that L^{i-1} and R^{i-1} are chosen by $\mathcal{S}$ in advance in the pre-processing I phase. Moreover, $\mathcal{S}$ can also choose M and $\tilde{M}$ beforehand, as they are just chosen randomly from $\mathsf{NonSing}^{n \times n}(\mathbb{F})$. Hence, $(L^{i-1}, R^{i-1}, M, \tilde{M})$ can be treated as constants and "hard-wired" into the leakage function that $\mathcal{A}$ issues to its leakage oracle in the ith iteration.
Now, to see (a) observe that A is a function of L^{i-1} and M (cf. (11)), which are "hard-wired" into the leakage function, and $\tilde{A}$ is a function of L^i, L^{i-1} and $\tilde{M}$ (cf. (12) and (13)), where L^{i-1} and $\tilde{M}$ are "hard-wired" and $L^i := L$ comes from the target oracle).
To see (b) observe that B is a function of M, R^i and R^{i-1} (cf. (9) and (10)), where M and R^{i-1} are "hard-wired", and $\tilde{B}$ is a function of $\tilde{M}$ (a "hard-wired" value), and R^i (cf. 14).
Hence, the $(\lambda/2 - 1)$-leakage game run by $\mathcal{A}$ at the ith iteration can be simulated by $\mathcal{S}$ using a $(\lambda/2)$-leakage game against $\Omega(L; R)$ (the "+1" overhead comes from the fact that we need a little bit of extra space in order to communicate the fact that the simulation aborted).

Leakage of $(L^{i+1}, R^{i+1}) \leftarrow \mathsf{Refresh}_{\mathbb{F}}^{n,m}(L^i, R^i)$ **in the** $(i+1)$**th iteration:** Simulating the leakage form phase $i + 1$ is done in a very similar way. First the simulator chooses random non-singular matrices M and $\tilde{M}$. He sets: $X := R^{i+1} - R^i$, and $B := M^{-1} \cdot X$. Then he calculates $A := L^i \cdot M$.

He then computes $Y := L^{i+1} - L^i$ and $\tilde{A} := Y \cdot \tilde{M}^{-1}$, and finally $\tilde{B} := \tilde{M} \cdot R^i$. He aborts in case $L^i \cdot R^{i+1} \neq S$ (denote this event by $\mathcal{Q}'_1$), or in case one of $A, B, \tilde{A}$ and $\tilde{B}$ does not have a full rank (denote this event by $\mathcal{Q}'_2$). Similarly to (15) it is easy to calculate that

$$A \cdot B = \begin{cases} (0, \dots, 0) & \text{if } L^i \cdot R^i = S' \\ S' - S & \text{if } L^i \cdot R^i = S. \end{cases} \tag{16}$$

Moreover, it is easy to check that, exactly as before, the $(\lambda/2 - 1)$-leakage game performed by $\mathcal{A}$ in the $(i+1)$th iteration can be simulated by $\mathcal{S}$ using a $(\lambda/2)$-leakage game against $\Omega(L; R)$.

Post-processing: simulating the remaining iterations: The simulator $\mathcal{S}$ simulates $\mathcal{A}$ on the variables that he generated previously for the $i+1$th,…,nth iteration. Exactly as in "pre-precessing II" he can do it since he generated the corresponding variables himself. At the end he outputs the output of $\mathcal{A}$.

We now prove that for any $b \in \{0, 1\}$ we have the following:

$$\Pr[Out(\mathcal{S}, \Omega(L,R)) = b \mid \overline{\mathcal{Q}_1 \vee \mathcal{Q}_2 \vee \mathcal{Q}'_1 \vee \mathcal{Q}'_2}] = \begin{cases} \Pr[\widetilde{\mathbf{Hyb}}_i(\mathcal{A}, S, S', \ell) = b \mid \overline{\mathcal{Q}}] & \text{if } L \cdot R = S \\ \Pr[\mathbf{Hyb}_{i+1}(\mathcal{A}, S, S', \ell) = b \mid \overline{\mathcal{Q}}] & \text{if } L \cdot R = S' \end{cases} \tag{17}$$

To show it, we assume that the players and the simulation did not abort (i.e., we have $\overline{\mathcal{Q}}$ and $\overline{\mathcal{Q}_1 \vee \mathcal{Q}_2 \vee \mathcal{Q}'_1 \vee \mathcal{Q}'_2}$). It is easy to see that the distributions of the variables created by the simulator and $\widetilde{\mathbf{Hyb}}_i(\mathcal{A}, S, S', \ell)$ and $\mathbf{Hyb}_{i+1}(\mathcal{A}, S, S', \ell)$ are equal for the first $i-1$ iterations, and for the iterations $i+2, \dots, n$. The only non-trivial things happen in the ith and $(i+1)$th iteration. The main difference between $\widetilde{\mathbf{Hyb}}_i(\mathcal{A}, S, S', \ell)$ and $\mathbf{Hyb}_{i+1}(\mathcal{A}, S, S', \ell)$ can be summarized as follows:

	$\widetilde{\mathbf{Hyb}}_i(\mathcal{A}, S, S', \ell)$	$\mathbf{Hyb}_{i+1}(\mathcal{A}, S, S', \ell)$
$\tilde{A} \cdot \tilde{B}$ in ith iteration	$S' - S$	$(0, \dots, 0)$
$A \cdot B$ in $(i+1)$th iteration	$(0, \dots, 0)$	$S' - S$

The variables created by our simulator indeed satisfy these relations, as shown on (15) and (16). It remains to show that the distribution of the variables generated by $\mathcal{S}$ when interacting with $\Omega(\mathsf{Encode}_{\mathbb{F}}^{n,m}(S))$ or $\Omega(\mathsf{Encode}_{\mathbb{F}}^{n,m}(S'))$ (resp.) is identical to $\mathbf{Hyb}_{i+1}(\mathcal{A}, S, S', \ell)$ or $\widetilde{\mathbf{Hyb}}_i(\mathcal{A}, S, S', \ell)$ (resp.). To this end, consider the execution of the ith iteration of $\mathsf{Refresh}_{\mathbb{F}}^{n,m}$ in $\widetilde{\mathbf{Hyb}}_i(\mathcal{A}, S, S', \ell)$ experiment and compare it to the simulation of the ith iteration assuming that $L^i \cdot R^i = S'$ (the remaining cases can be analyzed in a similar way).

ith iteration in $\widetilde{\mathbf{Hyb}}_i(\mathcal{A}, S, S', \ell)$:

$(A, B) \leftarrow \mathcal{O}$
M is a random non-singular matrix s. t. $L^{i-1}M = A$
$X := M \cdot B$
$R^i := R^{i-1} + X$

$(\tilde{A}, \tilde{B}) \leftarrow \mathcal{O}'$
$\tilde{M}$ is a random non-singular matrix s. t. $\tilde{M} \cdot R^i = \tilde{B}$
$Y := \tilde{A} \cdot \tilde{M}$
$L^i := L^{i-1} + Y$

ith iteration of simulation, with $L \cdot R = L^i \cdot R^i = S'$:

$X := R^i - R^{i-1}$
check if $L^{i-1} \cdot X = (0, \dots, 0)$
M is a random non-singular matrix
$A := L^{i-1} \cdot M$
$B := M^{-1} \cdot X$

$Y := L^i - L^{i-1}$
$\tilde{M}$ is a random non-singular matrix
$\tilde{A} := Y \cdot \tilde{M}^{-1}$
$\tilde{B} := \tilde{M} \cdot R^i$.

Let us now discuss why these methods of sampling the variables are identical if the abort events do not occur.

1. L^{i-1}, R^{i-1}, A and M: In both settings (L^{i-1}, R^{i-1}) are sampled in the same way. Together with Lemma 17 from the Appendix B we get that in both cases L^{i-1}, R^{i-1}, A and M are distributed identically.
2. B, X and R^i:
   - *Left hand side:* B comes from the oracle $\mathcal{O}$ and is a random matrix from $\mathsf{NonSing}^{n\times m}(\mathbb{F})$ subject to the constraint that $A \cdot B = (0, \ldots, 0)$. Then, we compute $X := M \cdot B$ and $R^i := R^{i-1} + X$.
   - *Right hand side:* Here, R^i and R^{i-1} are fixed and we compute $X := R^i - R^{i-1}$ and then $B := M^{-1} \cdot X$.

   It is easy to see that in the first case, X is a random $n \times m$ matrix subject to the constraint that its column vectors are orthogonal to L^{i-1}. In the second case X is computed from the matrices R^i and R^{i-1}, where $(L^{i-1}, R^{i-1}) \leftarrow \mathsf{Encode}_{\mathbb{F}}^{n,m}(S)$ and $(L^i, R^i) \leftarrow \mathsf{Encode}_{\mathbb{F}}^{n,m}(S')$ are sampled randomly and independently. Hence, $X := R^i - R^{i-1}$ is a random matrix. As on the right hand side we further assumed that $L^{i-1} \cdot X = (0, ...0)$ (i.e., the check does not fail), we have that also on the right hand side X is a random $n \times m$ matrix with column vectors that are orthogonal to L^{i-1}, which implies that both distributions are identical.
3. $(\tilde{A}, \tilde{B}), \tilde{M}$ and L^i: We can argue this similar to above.

It remains to bound the probability that the abort does not occur during the simulation.

Claim. In the above simulation we have:

$$\Pr\left[\overline{\mathcal{Q}_1} \wedge \overline{\mathcal{Q}_2} \wedge \overline{\mathcal{Q}_1'} \wedge \overline{\mathcal{Q}_2'}\right] \geq |\mathbb{F}|^{-2m}/2. \tag{18}$$

Proof. The event $\mathcal{Q}_1$ occurs if $L^{i-1} \cdot R^i \neq S$. Since clearly $L^{i-1} \cdot R^i$ has a uniform distribution over $|\mathbb{F}|^m$ and the same holds for $\mathcal{Q}_1'$, we have

$$\Pr[\overline{\mathcal{Q}}_1] = \Pr[\overline{\mathcal{Q}}_1'] = |\mathbb{F}|^{-m}. \tag{19}$$

We next define the events $\mathcal{Q}_A, \mathcal{Q}_B, \mathcal{Q}_{\tilde{A}}$ and $\mathcal{Q}_{\tilde{B}}$ as the events that the corresponding variables $A, B, \tilde{A}$ and $\tilde{B}$ do not have a full rank. Similarly, we define the events $\mathcal{Q}_A', \mathcal{Q}_B', \mathcal{Q}_{\tilde{A}}'$ and $\mathcal{Q}_{\tilde{B}}'$. As M is a full rank matrix and L^{i-1} is a random vector in $\mathbb{F}^n$, we get that A is chosen uniformly at random in $\mathbb{F}^n$, which gives us:

$$\Pr[\mathcal{Q}_A | \overline{\mathcal{Q}}_1 \wedge \overline{\mathcal{Q}}_1'] \leq |\mathbb{F}|^{-n}. \tag{20}$$

Let us next consider the event $\mathcal{Q}_B$. It is easy to see that the rows of X^{T} are chosen uniformly at random from the subspace of vectors that are orthogonal to L^{i-1}. Hence, by Lemma 16, it follows that X^{T} has rank m with probability at least $1 - m \cdot |\mathbb{F}|^{m-n}$. By Lemma 18, this implies that $\mathrm{rank}(X) = m$ with probability at least $1 - m \cdot |\mathbb{F}|^{m-n}$, and by the same lemma, we get that $\mathrm{rank}(B) = m$ with probability at least $1 - m \cdot |\mathbb{F}|^{m-n}$. As a similar argument works for $\mathcal{Q}_{\tilde{B}}$, we get:

$$\Pr[\mathcal{Q}_B | \overline{\mathcal{Q}}_1 \wedge \overline{\mathcal{Q}}_1'] \leq m \cdot |\mathbb{F}|^{m-n}, \tag{21}$$

$$\Pr[\mathcal{Q}_{\tilde{B}} | \overline{\mathcal{Q}}_1 \wedge \overline{\mathcal{Q}}_1'] \leq m \cdot |\mathbb{F}|^{m-n}. \tag{22}$$

Next, we consider $\mathcal{Q}_{\tilde{A}}$. As $\tilde{M}$ has full rank and conditioned on $\overline{\mathcal{Q}}_1$ the vector Y is a random vector from an $n - m$ dimensional subspace, we get

$$\Pr[\mathcal{Q}_{\tilde{A}} | \overline{\mathcal{Q}}_1 \wedge \overline{\mathcal{Q}}_1'] \leq |\mathbb{F}|^{m-n}. \tag{23}$$

The same arguments work for bounding $\mathcal{Q}'_A$, $\mathcal{Q}'_B$, $\mathcal{Q}'_{\tilde{A}}$ and $\mathcal{Q}'_{\tilde{B}}$. We now get:

$$\begin{aligned}
\Pr\left[\overline{\mathcal{Q}_1} \wedge \overline{\mathcal{Q}_2} \wedge \overline{\mathcal{Q}'_1} \wedge \overline{\mathcal{Q}'_2}\right] &= \Pr[\overline{\mathcal{Q}}_2 \wedge \overline{\mathcal{Q}}'_2 | \overline{\mathcal{Q}}_1 \wedge \overline{\mathcal{Q}}'_1] \cdot \Pr[\overline{\mathcal{Q}}_1] \cdot \Pr[\overline{\mathcal{Q}}'_1] && (24)\\
&= \Pr[\overline{\mathcal{Q}}_2 \wedge \overline{\mathcal{Q}}'_2 | \overline{\mathcal{Q}}_1 \wedge \overline{\mathcal{Q}}'_1] \cdot |\mathbb{F}|^{-2m} && (25)\\
&= (1 - \Pr[\mathcal{Q}_2 \vee \mathcal{Q}'_2 | \overline{\mathcal{Q}}_1 \wedge \overline{\mathcal{Q}}'_1]) \cdot |\mathbb{F}|^{-2m} && (26)\\
&\geq (1 - \Pr[\mathcal{Q}_A | \overline{\mathcal{Q}}_1 \wedge \overline{\mathcal{Q}}'_1] - \ldots - \Pr[\mathcal{Q}'_{\tilde{B}} | \overline{\mathcal{Q}}_1 \wedge \overline{\mathcal{Q}}'_1]) \cdot |\mathbb{F}|^{-2m} && (27)\\
&\geq (1 - 8m \cdot |\mathbb{F}|^{m-n}) \cdot |\mathbb{F}|^{-2m} && (28)\\
&\geq |\mathbb{F}|^{-2m}/2. && (29)
\end{aligned}$$

Eq. (24) follows from the fact that $\mathcal{Q}_1$ and $\mathcal{Q}'_1$ are independent and (25) from Eq. (20). Eq.(26) is a standard transformation and Eq.(27) follows from the union bound. Eq.(28) follows from Eq.(20)-(23) and $m \cdot |\mathbb{F}|^{m-n} \geq |\mathbb{F}|^{-n}$. Finally, Eq. 29 uses that $(1 - 8m \cdot |\mathbb{F}|^{m-n}) \geq 1/2$ when $m \leq n/2$ and $n \geq 16$. This concludes the proof. □

We are now ready to obtain a contradiction:

$$\begin{aligned}
&\Delta\left(Out(\mathcal{S}, \Omega(\mathsf{Encode}_{\mathbb{F}}^{n,m}(S)))\ ;\ Out(\mathcal{S}, \Omega(\mathsf{Encode}_{\mathbb{F}}^{n,m}(S')))\right)\\
&\geq \Delta\left(Out(\mathcal{S}, \Omega(\mathsf{Encode}_{\mathbb{F}}^{n,m}(S)))\ ;\ Out(\mathcal{S}, \Omega(\mathsf{Encode}_{\mathbb{F}}^{n,m}(S')))|\overline{\mathcal{Q}}_1 \wedge \overline{\mathcal{Q}}'_1 \wedge \overline{\mathcal{Q}}_2 \wedge \overline{\mathcal{Q}}'_2\right) \cdot \Pr[\overline{\mathcal{Q}}_1 \wedge \overline{\mathcal{Q}}'_1 \wedge \overline{\mathcal{Q}}_2 \wedge \overline{\mathcal{Q}}'_2]\\
&\geq \Delta\left(Out(\mathcal{S}, \Omega(\mathsf{Encode}_{\mathbb{F}}^{n,m}(S)))\ ;\ Out(\mathcal{S}, \Omega(\mathsf{Encode}_{\mathbb{F}}^{n,m}(S')))|\overline{\mathcal{Q}}_1 \wedge \overline{\mathcal{Q}}'_1 \wedge \overline{\mathcal{Q}}_2 \wedge \overline{\mathcal{Q}}'_2\right) \cdot |\mathbb{F}|^{-2m}/2\\
&\geq \Delta\left(\widetilde{\mathbf{Hyb}}_i(\mathcal{A}, S, S', \ell)\ ;\ \mathbf{Hyb}_{i+1}(\mathcal{A}, S, S', \ell)|\overline{\mathcal{Q}}\right) \cdot |\mathbb{F}|^{-2m}/2\\
&\geq 2\,|\mathbb{F}|^{2m} \epsilon \cdot |\mathbb{F}|^{-2m}/2 = \epsilon.
\end{aligned}$$

As further $\mathcal{S}$ retrieves in total λ bits from each party ($\lambda/2$ in phase i and $\lambda/2$ in phase $i+1$) we get a contradiction to the (λ, ϵ)-security of $\Phi_{\mathbb{F}}^{n,m}$, which concludes the proof of the lemma. □

Proof (of Theorem 1). As shown in Lemma 4 conditioned on $\overline{\mathcal{Q}}$ the distance between each consecutive distributions $\mathsf{Exp}_{\mathsf{Refresh}}(\mathcal{A}, S, \ell)$, $\mathbf{Hyb}_1(\mathcal{A}, S, S', \ell)$, $\widetilde{\mathbf{Hyb}}_1(\mathcal{A}, S, S', \ell)$,..., $\mathsf{Exp}_{\mathsf{Refresh}}(\mathcal{A}, S', \ell)$ is at most $2\,|\mathbb{F}|^{2m} \epsilon$. Since the sequence of distributions has length $2\ell+2$, we get by the triangle inequality (applied $2\ell+1$ times) that the distance between the first $\mathsf{Exp}_{\mathsf{Refresh}}(\mathcal{A}, S, \ell)$ and the last distribution $\mathsf{Exp}_{\mathsf{Refresh}}(\mathcal{A}, S', \ell)$ conditioned on $\overline{\mathcal{Q}}$ is at most $(2\ell+1)2\,|\mathbb{F}|^{2m} \epsilon \leq 6\ell\,|\mathbb{F}|^{2m} \epsilon$. With Lemma 3, we then get:

$$\begin{aligned}
\Delta(\mathsf{Exp}_{\mathsf{Refresh}}(\mathcal{A}, S, \ell); \mathsf{Exp}_{\mathsf{Refresh}}(\mathcal{A}, S', \ell)) &\leq (6\ell\,|\mathbb{F}|^{2m} \epsilon) \Pr[\overline{\mathcal{Q}}] + \Pr[\mathcal{Q}]\\
&\leq 6\ell\,|\mathbb{F}|^{2m} \epsilon + \Pr[\mathcal{Q}] \leq 6\ell\,|\mathbb{F}|^{2m} \epsilon + 2\ell m \cdot |\mathbb{F}|^{m-n-1}\\
&\leq 2\ell\,|\mathbb{F}|^{m} (3\,|\mathbb{F}|^{m} \epsilon + m\,|\mathbb{F}|^{-n-1}).
\end{aligned}$$

This proves (6), and hence shows the statement of the theorem. □

Combining this theorem 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.*

3.1 Security of Refreshing with Additional Auxiliary Information

Consider a secret $S \in \mathcal{M}^m$ and suppose we use our protocol in a setting where the secret is sampled from an affine subspace $Z \subseteq \mathcal{M}^m$ of dimension $m' < m$ that is known to the adversary beforehand.[4] In such a case our security definition says that an adversary that observes leakage from the refreshing cannot learn any additional information about S. In this section, we extend our security notion and show that the above statement holds even when the adversary obtains some auxiliary information on the encodings. More precisely, for an affine subspace Z of dimension m', let W be a $m \times (m - m')$-matrix of full rank and $Q \in \mathcal{M}^{m-m'}$ be a vector such that the solution space of $S \cdot W = Q$ defines the affine subspace Z. Then, we show that for any $S \in Z$ and $(L^0, R^0) \leftarrow \mathsf{Encode}(S)$, given $\mathsf{aux}^{i-1} = R^{i-1} \cdot W$ from the execution of $(L^i, R^i) \leftarrow \mathsf{Refresh}(L^{i-1}, R^{i-1})$ does not provide the adversary with information other than the fact that S belongs to the subspace Z. To define this more formally, we extend the experiment $\mathsf{Exp}_{\mathsf{Refresh}}(\mathcal{A}, S, \ell)$ from above in the following way:

- initially we give $\mathsf{aux}^0 = R^0 \cdot W$ to the adversary $\mathcal{A}$
- for each iteration in Step 2, we run $\mathcal{A}(W, Q) \leftrightarrows \big(\mathsf{Refresh}(L^{i-1}, R^{i-1}) \to (L^i, R^i)\big)$ and *additionally* give $\mathsf{aux}^i = R^i \cdot W$ to $\mathcal{A}$.

We denote this experiment with $\mathsf{Exp}^{\mathsf{aux}}_{\mathsf{Refresh}}(\mathcal{A}, S, \ell, W, Q)$, and consider the following extension of Definition 1.

Definition 2. *[*$(\ell, \lambda, \epsilon)$**-refreshing protocol with auxiliary information***] Let $Z \subseteq \mathcal{M}^m$ be an $m' < m$ dimensional affine subspace defined by W and Q as given above. For a LRS $\Phi = (\mathsf{Encode}, \mathsf{Decode})$ with message space $\mathcal{M}$, a refreshing protocol* Refresh *is a* $(\ell, \lambda, \epsilon)$-refreshing protocol with auxiliary information defined by (W, Q)*, if for every λ-limited adversary $\mathcal{A}$ and every pair of messages $S, S' \in Z$ we have that $\Delta(\mathsf{Exp}^{\mathsf{aux}}_{\mathsf{Refresh}}(\mathcal{A}, S, \ell, W, Q); \mathsf{Exp}^{\mathsf{aux}}_{\mathsf{Refresh}}(\mathcal{A}, S', \ell, W, Q)) \leq \epsilon$.*

The reader may ask why such an extended security definition is useful at all. Indeed, it turns out that such an extension is helpful when we want to apply our refreshing protocol to construct leakage-resilient cryptographic schemes. We will detail on this in the next sections.

Before we show that our refreshing protocol also satisfies this stronger notion of security, notice that since the adversary can choose the leakage function after learning aux^i, he can essentially leak from P_{L} the value of $L^i \cdot \mathsf{aux}^i = L^i \cdot R^i \cdot W = S \cdot W$ (if $m - m'$ is small), i.e., he can learn some linear function of the secret. The good news is that he knows $S \cdot W$ already because it is equal to Q! As it turns out, this is the only information that he will learn. Formally, we can show the following generalization of Theorem 1.

Theorem 2 (Generalization of Theorem 1). *Let $m/4 \leq n$, $n \geq 16$ and $\ell \in \mathbb{N}$. Let (W, Q) be as above defining an $m' < m$ dimensional affine subspace Z. Let n, m' and $\mathbb{F}$ be such that $\Phi^{n,m'}_{\mathbb{F}}$ is (λ, ϵ)-secure (for some λ and ϵ). The protocol* $\mathsf{Refresh}^{n,m}_{\mathbb{F}}$ *is a* $(\ell, \lambda/2 - 1, \epsilon')$-refreshing protocol with auxiliary information defined by (W, Q) *for $\Phi^{n,m}_{\mathbb{F}}$. Here, we have: $\epsilon' \leq 2\ell \left|\mathbb{F}\right|^m (3 \left|\mathbb{F}\right|^{2m} \epsilon + m \left|\mathbb{F}\right|^{-n-1})$.*

Proof (sketch). The proof is very similar to the proof of Theorem 1 and we only repeat the relevant details here. In the reduction of Theorem 1, we need to build a λ-limited simulator $\mathcal{S}$ that can simulate the view of a $(\lambda/2 - 1)$-limited adversary in $\mathsf{Exp}^{\mathsf{aux}}_{\mathsf{Refresh}}(\mathcal{A}, S, \ell, W, Q)$. To this end, let us first consider the case when the $m \times (m - m')$-matrix W has a special form (call it W'): namely, its 1 on its diagonal and otherwise 0. Note that knowledge of W' and Q with $S \cdot W'$ means that the first $m - m'$ coordinates of S are given to the adversary.

[4] In such cases, a natural idea to make the encoding more efficient is to fix some bijection $f : \mathcal{M}^{m'} \to Z$ and encode $x \in \mathcal{M}^{m'}$ instead of encoding $f(x)$. This is of course possible only if f is efficiently computable. As it turns out, in some cases such f cannot be efficiently computed. This happens e.g., if we use the encoded secrets in computationally-secure protocols (e.g., the Okamoto identification scheme — cf. Sect. 4).

Let $S, S' \in Z$ be two vectors which are equal on their first $m - m'$ coordinates, and we denote by $\tilde{S}$ and $\tilde{S}'$ the last m' coordinates of S and S', respectively. As in Theorem 1 consider $\widetilde{\mathbf{Hyb}}_i(\mathcal{A}, S, S', \ell)$ and $\mathbf{Hyb}_{i+1}(\mathcal{A}, S, S', \ell)$ which differ only in the ith and $(i+1)$th round. This means for the first $i-1$ executions of Refresh the simulator will refresh an encoding of S and for the last $(i+2)$ rounds, it refreshes and encoding of S'. Notice that for $j \in \{0, \ldots, i-1, i+2, \ldots, \ell\}$ the simulator $\mathcal{S}$ can trivially generate $\mathsf{aux}^j = R^j \cdot W'$, as R^j is known completely to $\mathcal{S}$.

For the simulation of the ith and $(i+1)$th execution $\mathcal{S}$ uses access to its target oracle $\Omega(L, R)$, where $(L, R) \leftarrow \mathsf{Encode}_{\mathbb{F}}^{n,m'}(\tilde{S})$ or $(L, R) \leftarrow \mathsf{Encode}_{\mathbb{F}}^{n,m'}(\tilde{S}')$. In the simulation $\mathcal{S}$ sets $L := L^i$ and the last m' columns of R^i to R. To sample the first $(m - m')$ columns of R^i it proceeds as follows. It samples uniformly at random vectors $R_1, \ldots, R_{m-m'}$ and checks with access to $\Omega(L, R)$, whether $L \cdot (R_1, \ldots, R_{m-m'}) = (S_1, \ldots, S_{m-m'})$. If it does not hold, then $\mathcal{S}$ aborts. Let $\overline{\mathcal{Q}}$ be the event that $\mathcal{S}$ does not abort. In the simulation $\mathcal{S}$ sets:

$$R^i := (R_1, \ldots, R_{m-m'}, R)\,,$$

and then proceeds as described in the proof of Theorem 1 to sample the remaining variables. Since $(R_1, \ldots, R_{m-m'})$ are known and W' has the special form as described above, $\mathcal{S}$ can easily compute $\mathsf{aux}^i = R^i \cdot W$.

It remains to analyze why the simulation as described above has the right distribution. It suffices to argue why (L^i, R^i) has the required distribution, as the remaining variables are sampled as in Theorem 1. This is easy to see as conditioned on $\overline{\mathcal{Q}}$ the first $(m - m')$ column vectors of R^i are sampled uniformly at random (and independently of R) subject to the constraint that $L \cdot (R_1, \ldots, R_{m-m'}) = (S_1, \ldots, S_{m-m'})$. Since $\Pr[\overline{\mathcal{Q}}] \geq \mathbb{F}^{-m+m'}$, we get together with the analysis from Theorem 1 that

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

The general case, i.e., when W is an arbitrary matrix, follows from the fact that every $W \in \mathbb{F}^{m \times (m-m')}$ can be transformed to this special form W' by multiplying it (from the left side) by some non-singular matrix $N \in \mathbb{F}^{m \times m}$ (let $W' = N \cdot W$). Hence, we can adjust the computation that is carried out by the simulator by multiplying the appropriate variables by N. □

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. The public key is known to both the prover $\mathcal{P}$ and the verifier $\mathcal{V}$.
- $(\mathcal{P}(pk, sk), \mathcal{V}(pk))$: 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. To define (black-box) *security* of an identification scheme ID, we consider a polynomial-time adversary $\mathcal{A}$ that gets the public key pk and interacts with the prover $\mathcal{P}(pk, sk)$ playing the role of a verifier. At the end of this interaction the adversary tries to impersonate $\mathcal{P}(pk, sk)$ by engaging in an interaction with $\mathcal{V}(sk)$. The adversary successfully impersonates the prover, if $\mathcal{V}(sk)$ outputs *accept*. We say that the scheme is *secure* if every polynomial-time adversary $\mathcal{A}$ impersonates the prover with only negligible probability.

We will now extend this standard security notion to capture leakage attacks on the prover, where the adversary obtains 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, besides exchanging messages with the prover, to obtain leakage from the prover's execution. We denote a single execution of this process by $\mathcal{A} \leftrightarrows \big(\mathcal{P}(sk) \rightarrow sk'\big)$, where sk' may be the updated key. In the definition below, we formalize security against such an adversary.

Definition 3 (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: $\mathcal{A} \leftrightarrows \big(\mathcal{P}(sk^i) \rightarrow sk^{i+1}\big)$, where in each execution the adversary can interact with the honest prover and retrieves 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. Hence, in total the adversary may learn $poly(k) \cdot \lambda(k)$ bits of information.

4.1 A Construction of a Leakage-Resilient Identification Protocol

Our construction is based on the standard Okamoto identification scheme [31], which works as follows. 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. The prover chooses $(w_1, w_2) \leftarrow \mathbb{Z}_p^2$, computes $a := g_1^{w_1} g_2^{w_2}$, and sends a to the verifier.
2. The verifier chooses $c \leftarrow \mathbb{Z}_p$ and sends it to the prover.
3. The prover computes $z_1 := w_1 + cx_1$ and $z_2 := w_2 + cx_2$ and sends (z_1, z_2) to the verifier.
4. The verifier accepts if and only if $g_1^{z_1} g_2^{z_2} \overset{?}{=} a \cdot pk^c$.

It has been shown that the Okamoto scheme is secure against impersonation attacks under the discrete logarithms assumption. We now 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. As outlined in the introduction such functions can be implemented in a "leakage-resilient way". The protocol is given in Figure 2.

Finally, we will combine our identification protocol with our refreshing protocol from Section 3 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, i.e., secure even against a polynomial number of observations. 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:

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

It is easy to see that the above protocol satisfies the completeness property. This is due to the soundness 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. To prove leakage-resilience of $\mathsf{Oka} = (\mathsf{KeyGen}, \mathcal{P}, \mathcal{V}, \mathsf{Refresh}_{\mathbb{Z}_p}^{n,2})$, we will proceed in three steps:

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

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}.

Computation of Prover $\mathcal{P}(pk, (L, R))$**:**

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 $\mathcal{V}$.

Computation of Verifier $\mathcal{V}(pk)$**:**

3. Pick $c \in \mathbb{Z}_p$ and send it 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 $\mathcal{V}$.

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 show that the there exists an (unbounded) simulator $\mathcal{S}(pk, \mathsf{aux})$ with access to a leakage oracle $\Omega(L^*, R^*)$ that, given "some" auxiliary information aux, can simulate $\mathcal{A}(pk)$'s view in $\mathcal{A} \leftrightarrows (\mathcal{P}(L, R)) \rightarrow (L', R'))$ (cf. Lemma 5). In this step the analysis neglects the leakage from the refreshing process.
2. We next consider the setting where unbounded $\mathcal{A}$ runs in many iterations of $\mathcal{A} \leftrightarrows \left(\mathcal{P}(L^i, R^i)) \rightarrow (L^{i+1}, R^{i+1})\right)$, 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 from Lemma 5 to show that *any* unbounded adversary will only learn a negligible amount of information about the secret key (cf. Lemma 6).
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. [3], which shows security of the original Okamoto scheme for keys sampled from a high average min-entropy source.

We now follow the three steps given above and start by proving a simple property of a single execution of the protocol $(\mathcal{P}(pk, sk), \mathcal{V}(pk))$ from Figure 2. Informally speaking, we show in the lemma below that for every unbounded adversary $\mathcal{A}$, there exists an unbounded simulator $\mathcal{S}$ that satisfies one of the following two properties:

1. $\mathcal{S}$ perfectly simulates the view of $\mathcal{A}$ in $\mathcal{A} \leftrightarrows (\mathcal{P}(pk, sk) \rightarrow (L, R))$ with $sk \leftarrow \mathsf{Encode}_{\mathbb{Z}_p}^{n,2}(x_1, x_2)$, or
2. $\mathcal{S}$ aborts the simulation, and outputs $\perp$. We denote the event that $\mathcal{S}$ aborts by $\mathcal{Q}$.

Lemma 5. *Let* $(pk, (L, R)) \leftarrow \mathsf{KeyGen}(1^k)$ *where* $(x_1, x_2) = \mathsf{Decode}_{\mathbb{Z}_p}^{n,2}(L, R)$. *Then for any (unbounded) λ-limited adversary $\mathcal{A}$, there exists a $(\lambda + 3\log p)$-limited simulator $\mathcal{S}$ with access to $\Omega(L^*, R^*)$ (here*

$(L^*, R^*) \leftarrow \mathsf{Encode}_{\mathbb{F}}^{n,1}(x_1)$*) and an event* $\mathcal{Q}$ *with* $\Pr\left[\overline{\mathcal{Q}}\right] \geq 1/p$ *such that for any* $b \in \{0,1\}$

$$\Pr[Out(\mathcal{S}(pk, \mathsf{aux}), \Omega(L^*, R^*)) = b|\overline{\mathcal{Q}}] = \Pr[\big(\mathcal{A}(pk) \leftrightarrows \big(\mathcal{P}(pk,(L,R)) \to (L',R')\big)\big) = b]. \tag{30}$$

Here, we have that $\mathsf{aux} = (1,\alpha)\cdot R$ *with* $\alpha = \log_{g_1}(g_2)$.

Notice that the simulator $\mathcal{S}$ only has leakage access to an encoding of a single value, namely to $(L^*, R^*) \leftarrow \mathsf{Encode}_{\mathbb{F}}^{n,1}(x_1)$, while he has to simulate the leakage of the refreshing of $(L,R) \leftarrow \mathsf{Encode}_{\mathbb{F}}^{n,2}(x_1,x_2)$. Notice, however, that $\mathcal{S}$ additionally knows $\mathsf{aux} = (1,\alpha)\cdot R$. From aux and R^* the simulator can easily compute a consistent encoding of x_2, as we will see in the proof below.

Proof. For an unbounded adversary $\mathcal{A}$, the information that is learnt by her contains (1) the messages that she receives from $\mathcal{P}$, i.e., the data that she learns via black-box access to the prover, and (2) the leakage from the execution of $(\mathcal{P}(pk,(L,R)), \mathcal{V}(pk))$. More formally, we can describe this information by:

1. Messages sent by $\mathcal{P}$ to the adversary:

$$\overbrace{\begin{pmatrix} 1 & \alpha & 0 & 0 \\ 0 & 0 & 1 & \alpha \\ c & 0 & 1 & 0 \\ 0 & c & 0 & 1 \end{pmatrix}}^{N} \cdot \begin{pmatrix} x_1 \\ x_2 \\ w_1 \\ w_2 \end{pmatrix} = \begin{pmatrix} \log_{g_1}(pk) \\ \log_{g_1}(a) \\ z_1 \\ z_2 \end{pmatrix}$$

   Recall that $\mathcal{A}$ is unbounded, hence she can compute discrete logarithms in $\mathbb{G}$.
2. Access to the leakage oracle via $\Omega\left((L,U,Z)\ ;\ (R,W)\right)$. Again, as $\mathcal{A}$ is unbounded, she can compute the discrete logarithm of U to the basis g_1 which is the vector $W_1 + \alpha W_2$. Hence, information theoretically it suffices to consider $\Omega\left((L, W_1+\alpha W_2, Z)\ ;\ (R,W)\right)$.

We need to show that $\mathcal{S}(pk, \mathsf{aux})$ can do a perfect simulation of this view with just access to the leakage oracle.

1. *Simulation of the leakage oracle as specified in 2:* To this end, we need to simulate the variables from the leakage oracle:

$$\Omega\left((L, W_1+\alpha W_2, Z)\ ;\ (R,W)\right), \tag{31}$$

   which is done with access to $\Omega(L^*, R^*)$ as follows:
   - *The encoding of the secret* (L,R)*:* $\mathcal{S}$ can simulate these variables by setting $L := L^*$ and the first column of R to $R_1 := R^*$. The second column is computed by $R_2 := (\mathsf{aux} - R^*)/\alpha$.
   - *The vectors* $W_1 + \alpha W_2$ *and the matrix* Z*:* As W_1 and W_2 are sampled uniformly at random, $\mathcal{S}$ can draw $Z \leftarrow \mathbb{Z}_p^{2n}$ uniformly at random. As the vector aux and the column vector of Z span a vector space of dimension 3 and $W_1 + \alpha W_2$ lies in this vector space, we can compute a consistent vector $W_1 + \alpha W_2$ as a linear combination of aux and Z.
   - *The matrix* W*:* Consider the following equation system:

$$N \cdot \begin{pmatrix} R^{\mathsf{T}} \\ W^{\mathsf{T}} \end{pmatrix} = \begin{pmatrix} \mathsf{aux} = R_1 + \alpha R_2 \\ W_1 + \alpha W_2 \\ Z_1 \\ Z_2 \end{pmatrix}. \tag{32}$$

   Here, the right side of the system is fixed and known to the simulator. Further, R is fixed as above, i.e., by the target oracle and aux. Hence, in this equation system it remains to compute the matrix W.

Since by our choice of $W_1 + \alpha W_2$ and Z the equation system is consistent and the last three rows of N are linearly independent, there exists a unique solution for (W_1, W_2) (recall that R is fixed which reduces the space of solutions to dimension 1). Since Z was sampled uniformly at random, it is easy to verify that also (W_1, W_2) is uniformly distributed conditioned on satisfying the above equation system. Hence, the distribution of the simulated W is as in the real experiment.

2. *Simulation of the messages sent by the prover as specified in 1:* As from the previous step the simulator knows $(W_1 + \alpha W_2, Z)$, he can query its leakage oracle to obtain $\log_{g_1}(a), z_1$ and z_2. Notice that this requires $\mathcal{S}$ to learn $3 \log p$ additional bits from its leakage oracle.

Finally, observe that since $\mathcal{A}$ may access its leakage oracle at any time (in particular before choosing c), but for the simulation from above we need to know the matrix N – in particular the value c, we let $\mathcal{S}$ simply guess c at the beginning of the simulation. If the guess was correct, then we perfectly simulated $\mathcal{A}(pk) \leftrightarrows (\mathcal{P}(pk, (L, R)) \rightarrow (L', R'))$; otherwise the event $\mathcal{Q}$ occurs. As this happens with probability at least $1/p$, we get the claim of the lemma. □

LEAKAGE FROM ℓ-EXECUTIONS. In the lemma above, we showed that the leakage from a single execution of the protocol can be perfectly simulated. We will now use this observation and prove that the leakage from several rounds will also not help in learning much about the encoded secret (cf. Step 2 in the above outline). This will require to refresh the encoded secret key periodically, as otherwise an adversary that continuously learns information from the device can trivially break the security. To this end, we consider $\mathsf{Oka} = (\mathsf{KeyGen}, \mathcal{P}, \mathcal{V}, \mathsf{Refresh}^{n,2}_{\mathbb{Z}_p})$ and assume that after each execution of the protocol from Figure 2, the prover executes $(L', R') \leftarrow \mathsf{Refresh}^{n,2}_{\mathbb{Z}_p}(L, R)$ and sets the new secret key (for the next execution) to (L', R'). We denote such a prover by $\mathcal{P}'$ and denote the ith execution of the identification protocol (with refreshing) by $(\mathcal{P}'(pk, (L^{i-1}, R^{i-1})), \mathcal{V}(pk))$.

Lemma 6. *Let $(pk, sk) \leftarrow \mathsf{KeyGen}(1^k)$, $\ell \in \mathbb{N}$, $\alpha = \log_{g_1}(g_2)$ and W the column vector $(1, \alpha)$. Suppose n, λ and ϵ are such that $\Phi^{n,1}_{Z_p}$ is (λ, ϵ)-secure. Then, for every $S = (x_1, x_2)$ and $S' = (x'_2, x'_2)$ that correspond to the same public key pk (cf. "key generation" on Fig. 2), and any $(\lambda/2 - 1 - 3\log p)$-limited adversary $\mathcal{A}$ we have that*

$$\Delta\left(\mathsf{Exp}^{\mathsf{aux}}_{\mathcal{P}'(pk)}(\mathcal{A}, S, \ell, W, pk); \mathsf{Exp}^{\mathsf{aux}}_{\mathcal{P}'(pk)}(\mathcal{A}, S', \ell, W, pk)\right) \le 2\ell p^7(3\epsilon + 2p^{-n-5}).$$

Proof. Notice that (W, pk) defines a 1-dimensional subspace $Z \subset (Z_p)^2$ that contains all the pairs (x_1, x_2) that correspond to the public key pk. For any $S, S' \in Z$ and any $(\lambda/2 - 1)$-limited adversary $\mathcal{A}$ we have by Theorem 2 for the refreshing of $\Phi^{n,2}_{\mathbb{Z}_p}$ that

$$\Delta\left(\mathsf{Exp}^{\mathsf{aux}}_{\mathsf{Refresh}}(\mathcal{A}, S, \ell, W, pk); \mathsf{Exp}^{\mathsf{aux}}_{\mathsf{Refresh}}(\mathcal{A}, S', \ell, W, pk)\right) \le 2\ell p^2(3p^4\epsilon + 2p^{-n-1}) = 2\ell p^6(3\epsilon + 2p^{-n-5}). \tag{33}$$

The above experiment considers only the leakage from the refreshing of (L^i, R^i). To combine this with leakage from the identification protocol, i.e., leakage from the prover $\mathcal{P}(pk)$ we use the simulation from Lemma 5. We can apply this lemma since (1) Eq. (33) is shown by reduction to the (λ, ϵ)-security of $\Phi^{n,1}_{Z_p}$ and (2) $\mathsf{aux}^i = R^i \cdot W$ is know to the simulator[5]. As the simulation from Lemma 5 is perfect, but fails with probability $1 - 1/p$, the error from Eq. (33) increases by at most a factor of p. Furthermore, notice that the simulation additionally needs to learn $3 \log p$ bits from its target oracle (for $\Phi^{n,1}_{Z_p}$), which gives us the claimed statement. □

The following corollary can be obtained from Lemma 6 in a similar way as Corollary 1 from Theorem 1

[5] Notice that this was the whole purpose of the extension presented in Theorem 2.

Corollary 3. *Let $n \in \mathbb{N}$ be the statistical security parameter. Furthermore, let $(pk, sk) \leftarrow \mathsf{KeyGen}(1^k)$, $\ell = poly(n)$, $\alpha = \log_{g_1}(g_2)$ and W be the column vector $(1, \alpha)$. For every $S = (x_1, x_2)$ and $S' = (x_2', x_2')$ that correspond to the same public key pk (cf. "key generation" on Fig. 2), and any $((0.15 \cdot n - 3)\log p - 1)$-limited adversary $\mathcal{A}$ we have that*

$$\Delta\left(\mathsf{Exp}^{\mathsf{aux}}_{\mathcal{P}'(pk)}(\mathcal{A}, S, \ell, W, pk); \mathsf{Exp}^{\mathsf{aux}}_{\mathcal{P}'(pk)}(\mathcal{A}, S', \ell, W, pk)\right) \leq negl(n).$$

We can now show that our scheme Oka satisfies the security notion of Definition 3.

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

Proof. It has been proven in Theorem 4.1 in [3] that the Okamoto scheme is secure against impersonation attacks under the DL assumption, even if the secret keys are sampled from some source with high average min entropy. From Corollary 3, we know that for any two uniformly sampled keys $S := (x_1, x_2), S' := (x_1', x_2')$ that correspond to the same public key pk and any $((0.15 \cdot n - 3)\log p - 1)$-limited adversary $\mathcal{A}$

$$\Delta\left(\mathsf{Exp}^{\mathsf{aux}}_{\mathcal{P}'(pk)}(\mathcal{A}, S, \ell, W, pk); \mathsf{Exp}^{\mathsf{aux}}_{\mathcal{P}'(pk)}(\mathcal{A}, S', \ell, W, pk)\right) \leq negl(n).$$

Hence, information theoretically an adversary does not learn much about the secret key S or S', respectively, by running in the experiment $\mathsf{Exp}^{\mathsf{aux}}_{\mathcal{P}'(pk)}$. More formally, by Lemma 9 in the appendix, this implies that the secret key has high average min-entropy even given the leakage and the public messages from the execution of the identification protocol (i.e., pk and a, z_1, z_2). This gives us with Theorem 4.1 in [3] that our scheme is $((0.15 \cdot n - 3)\log p - 1)$ $\mathsf{ID} - \mathsf{Leak}$-secure under the DL assumption. □

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 plaintext m we have $m = \mathsf{Decr}(sk, \mathsf{Encr}(pk, m))$.

The standard security notion for an encryption scheme is security against *chosen plaintext attacks* (IND-CPA). In a CPA attack against a public-key encryption scheme the adversary obtains pk, then picks two messages m_0, m_1 and has to guess the bit b on input $c^* = \mathsf{Encr}(pk, m_b)$. We can strengthen this notion by additionally allowing the adversary to ask for the decryption of chosen ciphertexts. A scheme that remains secure even if the adversary can ask for decryptions of chosen ciphertexts *prior* to seeing c^* is said to be secure against *non-adaptive chosen ciphertext attacks* (IND-CCA1 secure). If the adversary can additionally make decryption queries *after* seeing c^*, then the scheme is said to be secure against *adaptive chosen ciphertext attacks* (IND-CCA2 secure).

In the setting where computation leaks information, security against chosen plaintext and leakage attacks is not very interesting, as the adversary is not allowed to ask for decryption queries. On the other hand, the notion of CCA1/2-security can naturally be extended to the continuous leakage setting. Here, 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 4 (Security against Chosen Ciphertext Leakage Attacks (CCLA2)). *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. *The challenger samples $b \in \{0, 1\}$ and $(pk, sk) \leftarrow \mathsf{KeyGen}(1^k)$. It gives pk to $\mathcal{A}$.*
2. *Repeat until $\mathcal{A}(1^k)$ outputs (m_0, m_1): $\mathcal{A} \leftrightarrows \big(\mathsf{Decr}(sk, c) \to sk'\big)$, 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': $\mathcal{A} \leftrightarrows \big(\mathsf{Decr}(sk, c) \to sk'\big)$, 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. Notice that the adversary is allowed to obtain λ bits of information for *each* decryption query. Hence, in total the adversary may learn $poly(k) \cdot \lambda(k)$ bits of information.

5.2 An Efficient IND-CCLA2-secure Encryption Scheme in the Random Oracle Model

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 [4,36] 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 [23] and the techniques of [22]. Unfortunately, as pointed out by Dodis et al. [11], this results into an impractical scheme. In contrast, in the random oracle model using the Fiat-Shamir heuristic [19] simulation soundness can be achieved efficiently. In particular, it has been proven in [1] that the standard Chaum-Pedersen protocol [7] 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 obtained via the Fiat-Shamir heuristic. 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 and uses similar ideas as in our implementation of the Okamoto scheme. 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, i.e., secure even against a polynomial number of observations of the decryption process.

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). P_{L} runs the NIZK verifier $\mathsf{Ver}(u, v, \pi)$ to check that π is an accepting proof. If the verification fails, then output $\perp$ and stop; 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. The key generation KeyGen and the decryption process Decr of our public-key encryption scheme PKE.

SECURITY ANALYSIS OF $\mathsf{PKE} = (\mathsf{KeyGen}, \mathsf{Encr}, \mathsf{Decr}, \mathsf{Refresh}_{\mathbb{Z}_p}^{n,2})$. The formal security proof proceeds similar as the proof of security in the last section. We first show in Lemma 7 that the leakage from a single decryption query can be simulated in a perfect way with just access to a retrieving 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 $\mathsf{Refresh}_{\mathbb{Z}_p}^{n,2}$ to refresh the encoded secret key (L, R). Finally, we show in Theorem 4 that such a construction is IND-CCLA2 secure.

Lemma 7. *Let* $(pk, (L, R)) \leftarrow \mathsf{KeyGen}(1^k)$ *with* $(x_1, x_2) = \mathsf{Decode}_{\mathbb{Z}_p}^{n,2}(L, R)$*. Then for any (unbounded) λ-limited adversary $\mathcal{A}$, there exists a λ-limited simulator $\mathcal{S}$ with access to $\Omega(L^*, R^*)$ (here $(L^*, R^*) \leftarrow$*

$\mathsf{Encode}^{n,1}_{\mathbb{Z}_p}(x_1)$*) such that for any* $b \in \{0,1\}$

$$\Pr[Out(\mathcal{S}(pk, \mathsf{aux}), \Omega(L^*, R^*)) = b] = \Pr[(\mathcal{A}(pk) \leftrightarrows (\mathsf{Decr}((L,R),c) \to (L',R'))) = b], \tag{34}$$

where $\mathsf{aux} = (1,\alpha) \cdot R$ *with* $\alpha = \log_{g_1}(g_2)$*. Furthermore, we assume that* $c = (u,v,w,\pi)$ *is a valid ciphertext (i.e.,* $\log_{g_1}(u) = log_{g_2}(v)$*).*

Proof. The proof is very similar to the proof of Lemma 5. For the simulation to go through, it will be crucial that the adversary can only ask for decryptions of valid ciphertexts, i.e., in the following we consider $\log_{g_1}(u) = log_{g_2}(v) = r$. For an unbounded adversary $\mathcal{A}$, the information that is learnt from decryptions of ciphertexts $c = (u,v,w,\pi)$ can be described as follows:

1. The decrypted plaintext m that corresponds to the ciphertext c.
2. Leakage from the decryption: This involves the leakage from the verification of the NIZK π and the decryption of (u,v,w). As the verification of the NIZK only uses publicly known values, the simulation is trivial and will be omitted in the following. Hence, we can describe the leakage from the decryption by access to the following leakage oracle

$$\Omega\left((L, r(R_1 + \alpha R_2))\ ;\ R\right).$$

We need to show that $\mathcal{S}(pk, \mathsf{aux})$ can do a perfect simulation of the above described view with just access to its leakage oracle $\Omega(L^*, R^*)$. In fact, given the auxiliary information aux this simulation is trivial:

1. *Simulation of the leakage oracle as specified in Step 2:* We consider first how to simulate the leakage oracle. In the simulation, $\mathcal{S}$ identifies R_1 with R^* and can compute R_2 from R^* and aux. As $\mathcal{S}$ is unbounded, given the ciphertext c, he can compute $r = \log_{g_1}(u)$. Together with aux the simulator can compute $r(R_1 + \alpha R_2)$, which suffices to simulate the leakage queries.
2. *Simulation of the plaintext* $m = \mathsf{Decr}(sk, c)$*:* Again, as $\mathcal{S}$ knows r it can decrypt valid ciphertexts by $w \cdot h^{-r}$.

It is easy to see that for valid ciphertexts we get a perfect simulation. Further, since $\mathcal{S}$ does not need to learn extra information from its leakage oracle, we get the claimed result. □

In the lemma above, we showed that the leakage from a single execution of the decryption process can be perfectly simulated. We will now use this observation and prove that the leakage from several rounds will also not help in learning much about the encoded secret. This will require to refresh the encoded secret key periodically, as otherwise an adversary that continuously learns information from the device can trivially break the security. To this end, we consider $\mathsf{PKE} = (\mathsf{KeyGen}, \mathsf{Encr}, \mathsf{Decr}, \mathsf{Refresh}^{n,2}_{\mathbb{Z}_p})$ that executes after each run of the decryption process from Figure 3, the refreshing protocol $(L', R') \leftarrow \mathsf{Refresh}^{n,2}_{\mathbb{Z}_p}(L,R)$. The new secret key (i.e., for the next execution) is set to (L', R'). We denote such a decryption process by Decr'. We can now prove the following:

Lemma 8. *Let* $(pk, sk) \leftarrow \mathsf{KeyGen}(1^k)$*,* $\ell \in \mathbb{N}$*,* $\alpha = \log_{g_1}(g_2)$ *and* W *the column vector* $(1,\alpha)$*. Suppose* n, λ *and* ϵ *are such that* $\Phi^{n,1}_{Z_p}$ *is* (λ,ϵ)*-secure and that in the experiments below the decryption process* $\mathsf{Decr}'(\cdot)$ *is only run on valid ciphertext. Then, for every* $S = (x_1, x_2)$ *and* $S' = (x_2', x_2')$ *that corresponds to the same public key pk (cf. "key generation" on Fig. 3), and any* $(\lambda/2 - 1)$*-limited adversary* $\mathcal{A}$ *we have that*

$$\Delta\left(\mathsf{Exp}^{\mathsf{aux}}_{\mathsf{Decr}'(\cdot)}(\mathcal{A}, S, \ell, W, pk); \mathsf{Exp}^{\mathsf{aux}}_{\mathsf{Decr}'(\cdot)}(\mathcal{A}, S', \ell, W, pk)\right) \leq 2\ell p^6(3\epsilon + 2p^{-n-5}).$$

Proof (sketch). The proof is along the lines of Lemma 6. The only difference is that there is no need for the simulator to abort (recall that in the proof of the security of the Okamoto scheme the simulator needed to guess the value of c, that was later chosen by the simulated $\mathcal{A}$, and abort if he did not guess it correctly). Hence, we save a factor of p.

Similar to the last section, we can now obtain the following corollary that plugs-in concrete numbers.

Corollary 4. *Let $n \in \mathbb{N}$ be the statistical security parameter and suppose that in the experiments below the decryption process $\mathsf{Decr}'(\cdot)$ is only run on valid ciphertext. Furthermore, let $(pk, sk) \leftarrow \mathsf{KeyGen}(1^k)$, $\ell = poly(n)$, $\alpha = \log_{g_1}(g_2)$ and W be the column vector $(1, \alpha)$. For every $S = (x_1, x_2)$ and $S' = (x_2', x_2')$ that correspond to the same public key pk (cf. "key generation" on Fig. 3), and any $(0.15 \cdot n \log p - 1)$-limited adversary $\mathcal{A}$ we have that*

$$\Delta\left(\mathsf{Exp}^{\mathsf{aux}}_{\mathsf{Decr}'(\cdot)}(\mathcal{A}, S, \ell, W, pk); \mathsf{Exp}^{\mathsf{aux}}_{\mathsf{Decr}'(\cdot)}(\mathcal{A}, S', \ell, W, pk)\right) \le negl(n).$$

The above corollary says that an adversary that continuously observes the computation from the decryption process will not be able to learn any useful information about the encoded secret key. In the next theorem, we will show that such a property suffices to prove the IND-CCLA2 security of our scheme.

Theorem 4. PKE *is* $(0.15 \cdot n \log p - 1)$*-IND-CCLA2 secure in the random oracle model, if the DDH assumption holds.*

Proof. We prove this by a series of games. The games are all variants of the security experiment given in Definition 4. We will show that in the last game any adversary can guess the target bit b with probability at most $1/2$. As we will also show that the distances between the games are $negl(k)$ this will prove the statement of the theorem.

Game 0: This is as the original experiment as given in Definition 4. That is, a bit $b \leftarrow \{0, 1\}$ is sampled and the initial key $(pk, sk) \leftarrow \mathsf{KeyGen}(1^k)$ is generated and used to answer the decryption and leakage queries from $\mathsf{Decr}'(sk, c)$ (recall again that Decr' denotes the decryption process from Figure 3 followed by the key-refreshing with $\mathsf{Refresh}^{n,2}_{\mathbb{Z}_p}$). The target ciphertext c^* is generated by computing honestly $c^* = (u^*, v^*, w^*) \leftarrow \mathsf{Encr}(pk, m_b)$ and running the NIZK prover $\mathsf{Prov}(u^*, v^*, r^*)$ to obtain a proof π^* for $\log_{g_1}(u^*) = \log_{g_2}(v^*)$.

Game 1: This is as Game 1, but we generate the target ciphertext using the secret key. More precisely, we compute $c^* = (u^*, v^*, w^*, \pi^*)$ as follows: pick randomly $r \leftarrow \mathbb{Z}_p$ and compute $u^* = g_1^r$, $v^* = g_2^r$ and $w^* = (u^*)^{x_1} \cdot (v^*)^{x_2}$. Furthermore, compute π^* as above. Trivially, the distance between Game 0 and Game 1 is 0.

Game 2: This is as Game 1 (i.e., we simulate the answers to decryption and leakage queries honestly), but instead of running the NIZK prover that requires as input (u^*, v^*, r^*) (where r^* is the common discrete logarithm of u^* and v^*) to obtain the proof π^*, we run the NIZK simulator that only uses (u^*, v^*). By the zero-knowledge property of the NIZK the distance between the generated views in Game 1 and Game 2 is negligible.

Game 3: This is as Game 2, but we sample (u^*, v^*) randomly such that $\log_{g_1}(u^*) \neq \log_{g_2}(v^*)$. More precisely, as in Game 2, we sample the initial key $(pk, sk) \leftarrow \mathsf{KeyGen}(1^k)$. This allows us to answer the decryption queries and the corresponding leakage queries of the adversary. If $\mathcal{A}$ asks for a ciphertext of the messages (m_0, m_1), we reply with $c^* = (u^*, v^*, (u^*)^{x_1} \cdot (v^*)^{x_2} m_b, \pi^*)$. By the DDH assumption the distance between the views generated in Game 2 and Game 3 is negligible.

Game 4: This is as Game 3, but we reject to answer decryption queries for ciphertexts (u, v, w, π) with $\log_{g_1}(u) \neq \log_{g_2}(v)$. The views that are generated in Game 3 and Game 4 are close by the simulation soundness of the NIZK. Recall that simulation soundness guarantees that the adversary cannot generate accepting proofs of wrong statements, even if he has obtained accepting proofs of wrong statements earlier (this is the case as the target ciphertext c^* that the adversary sees in Step 3 below includes a proof of a wrong statement). To sum it up, Game 4 proceeds as follows:

1. Sample $b \leftarrow \{0, 1\}$ and a random (x_1, x_2) and compute $sk \leftarrow \mathsf{Encode}^{n,2}_{\mathbb{F}}(x_1, x_2)$. Compute pk from (x_1, x_2).

2. Answer valid decryption and leakage queries with sk until the adversary asks for (m_0, m_1).
3. Generate the target ciphertext $c^* = (u^*, v^*, w^*, \pi^*)$ by sampling (u^*, v^*) such that $\log_{g_1}(u^*) \neq \log_{g_2}(v^*)$, and computing $w^* = (u^*)^{x_1} \cdot (v^*)^{x_2} \cdot m_b$. Give (u^*, v^*, w^*, π^*) to the adversary.
4. Answer valid decryption and leakage queries with sk.

Game 5: This is as Game 4 with the following changes:

1. Sample a random (x_1, x_2) and compute pk from (x_1, x_2). Sample $(x_1', x_2') \leftarrow \{(y, z) | g_1^y \cdot g_2^z = g_1^{x_1} g_2^{x_2}\}$ uniformly at random and set $sk' \leftarrow \mathsf{Encode}_{\mathbb{F}}^{n,2}(x_1', x_2')$.
2. Answer the decryption and leakage queries with sk' until the adversary asks for (m_0, m_1).
3. Generate the target ciphertext c^* as in Game 3 by using (x_1, x_2).
4. Answer the decryption and leakage queries with sk'.

The distance between Game 4 and 5 is negligible by Lemma 8 (or the corresponding Corollary 4). More precisely, we have the following claim:

Claim. The distance between Game 4 and Game 5 is negligible in n.

Proof. We show this by reduction to Lemma 8 or the corresponding Corollary 4. Suppose we have given a distinguisher $\mathcal{D}$ for Game 4 and 5, and we will use it to construct an adversary $\mathcal{A}$ that breaks Corollary 4. $\mathcal{A}$ needs to simulate the view of the distinguisher $\mathcal{D}$ either in Game 4 or Game 5. To this end, it samples (x_1, x_2) at random, and samples $(x_1', x_2') \leftarrow \{(y, z) | g_1^y \cdot g_2^z = g_1^{x_1} g_2^{x_2}\}$ uniformly at random. Furthermore, it samples the corresponding public key pk. To answer the decryption and leakage queries, $\mathcal{A}$ asks its challenge oracle that either replies with the decryption and the corresponding leakage either by using $\mathsf{Decr}'(\mathsf{Encode}_{\mathbb{Z}_p}^{n,2}(x_1, x_2), \cdot)$ or by using $\mathsf{Decr}'(\mathsf{Encode}_{\mathbb{Z}_p}^{n,2}(x_1', x_2'), \cdot)$. Notice that here it is crucial that in Game 4 and 5 the adversary only asks for decryptions of valid cihpertexts, as the challenge oracle only replies to such queries. Next, $\mathcal{A}$ generates the target ciphertext c^* as described in Game 4 (which is identical to Game 5) and then it continues to answer decryption and leakage queries using its challenge oracle. It is easy to see that depending on whether $\mathcal{A}$'s target oracle uses $\mathsf{Decr}'(\mathsf{Encode}_{\mathbb{Z}_p}^{n,2}(x_1, x_2), \cdot)$ or $\mathsf{Decr}'(\mathsf{Encode}_{\mathbb{Z}_p}^{n,2}(x_1', x_2'), \cdot)$ the view of $\mathcal{D}$ is either as in Game 4 or Game 5. □

It remains to show that in Game 5 the adversary's view is independent of the bit b. Hence, the guessing probability in Game 5 is at most $1/2$.

Claim. The adversary's view in Game 5 is independent of the target bit b.

Proof. Consider the target ciphertext $c^* := (u^*, v^*, w^*, \pi^*)$, where $w^* = z^* \cdot m_b$ with $z^* = g_1^{r_1 x_1} \cdot g_2^{r_2 x_2}$ and $r_1 \neq r_2$. We will show that z^* acts as an information theoretically one-time pad. An (unbounded) adversary in Game 5 can compute from the public key pk the value $\log_{g_1}(h) = x_1 + \alpha \cdot x_2$. By the fact that in Game 5 the adversary only asks for valid cihpertexts (i.e., for a decryption query $c = (u, v, w, \pi)$ we have $\log_{g_1}(u) = \log_{g_2}(v)$) he does not learn more than this. Furthermore, since we used in Game 5 $sk' \leftarrow \mathsf{Encode}_{\mathbb{Z}_p}^{n,2}(x_1', x_2')$ to answer the leakage queries and (x_1', x_2') is chosen independent of (x_1, x_2), the adversary in Game 5 learns indeed no more than $x_1 + \alpha \cdot x_2$. To sum it up, together with the information from the target ciphertext, the adversary learns:

$$\begin{pmatrix} \log_{g_1}(h) \\ \log_{g_1}(z^*) \end{pmatrix} = \begin{pmatrix} 1 & \alpha \\ r_1 & r_2\alpha \end{pmatrix} \cdot \begin{pmatrix} x_1 \\ x_2 \end{pmatrix}.$$

Since $\alpha \neq 0$ and $r_1 \neq r_2$ the 2×2 matrix is non-singular, and hence $\log_{g_1}(h)$ and $\log_{g_1}(z^*)$ are linearly independent. □

As the distance between all games is negligible, we get the claimed result. □

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 ElGamal variant) computes only a linear function of the secret key. Notice that in the examples of the last section the linear function was essentially computed in the exponent of a group generator g. 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, this meant, for instance, that 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 during the execution of the identification protocol.

Various other cryptographic schemes satisfy the above properties, and hence can be made secure against continuous leakage attacks. For instance, the Pedersen commitment scheme [32], which is information-theoretically hiding and at the same time only requires to compute a linear function of its secrets.[6] Another example of the above paradigm is a variant of the linear Cramer-Shoup cryptosystem as presented in [37]. 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 [23]. 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 [22]. 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. Michel Abdalla, Xavier Boyen, Céline Chevalier, and David Pointcheval. Distributed public-key cryptography from weak secrets. In *Public Key Cryptography*, pages 139–159, 2009.
2. Adi Akavia, Shafi Goldwasser, and Vinod Vaikuntanathan. Simultaneous hardcore bits and cryptography against memory attacks. In Omer Reingold, editor, *Theory of Cryptography, 6th Theory of Cryptography Conference, TCC 2009, San Francisco, CA, USA, March 15-17, 2009. Proceedings*, volume 5444 of *Lecture Notes in Computer Science*, pages 474–495. Springer, 2009.
3. Joël Alwen, Yevgeniy Dodis, and Daniel Wichs. Leakage-resilient public-key cryptography in the bounded-retrieval model. In *CRYPTO*, pages 36–54, 2009.
4. Manuel Blum, Paul Feldman, and Silvio Micali. Non-interactive zero-knowledge and its applications (extended abstract). In *STOC*, pages 103–112, 1988.
5. Elette Boyle, Gil Segev, and Daniel Wichs. Fully leakage-resilient signatures. In *EUROCRYPT*, pages 89–108, 2011.
6. Zvika Brakerski, Yael Tauman Kalai, Jonathan Katz, and Vinod Vaikuntanathan. Overcoming the hole in the bucket: Public-key cryptography resilient to continual memory leakage. In *FOCS*, pages 501–510, 2010.
7. David Chaum and Torben P. Pedersen. Wallet databases with observers. In *CRYPTO*, pages 89–105, 1992.
8. Benny Chor and Oded Goldreich. Unbiased bits from sources of weak randomness and probabilistic communication complexity. *SIAM J. Comput.*, 17(2):230–261, 1988.
9. Francesco Davì, Stefan Dziembowski, and Daniele Venturi. Leakage-resilient storage. In *SCN*, volume 6280 of *Lecture Notes in Computer Science*, pages 121–137. Springer, 2010.

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

10. Yevgeniy Dodis, Kristiyan Haralambiev, Adriana López-Alt, and Daniel Wichs. Cryptography against continuous memory attacks. In *FOCS*, pages 511–520, 2010.
11. Yevgeniy Dodis, Kristiyan Haralambiev, Adriana López-Alt, and Daniel Wichs. Efficient public-key cryptography in the presence of key leakage. In *ASIACRYPT*, pages 613–631, 2010.
12. Yevgeniy Dodis, Allison Lewko, Brent Waters, and Daniel Wichs. How to store a secret on continually leaky devices. manuscript, 2011.
13. Yevgeniy Dodis, Rafail Ostrovsky, Leonid Reyzin, and Adam Smith. Fuzzy extractors: How to generate strong keys from biometrics and other noisy data. *SIAM Journal on Computing*, 38(1):97–139, 2008.
14. Stefan Dziembowski and Ueli Maurer. Optimal randomizer efficiency in the bounded-storage model. *Journal of Cryptology*, 17(1):5–26, January 2004. Conference version appeared in Proc. of STOC 2002.
15. Stefan Dziembowski and Krzysztof Pietrzak. Intrusion-resilient secret sharing. In *FOCS*, pages 227–237, 2007.
16. Stefan Dziembowski and Krzysztof Pietrzak. Leakage-resilient cryptography. In *FOCS '08: Proceedings of the 49th Annual IEEE Symposium on Foundations of Computer Science*, Washington, DC, USA, 2008. IEEE Computer Society.
17. Sebastian Faust, Eike Kiltz, Krzysztof Pietrzak, and Guy N. Rothblum. Leakage-resilient signatures. In Daniele Micciancio, editor, *Theory of Cryptography, 7th Theory of Cryptography Conference, TCC 2010, Zurich, Switzerland, February 9-11, 2010. Proceedings*, volume 5978 of *Lecture Notes in Computer Science*, pages 343–360. Springer, 2010.
18. Sebastian Faust, Tal Rabin, Leonid Reyzin, Eran Tromer, and Vinod Vaikuntanathan. Protecting circuits from leakage: the computationally-bounded and noisy cases. In *EUROCRYPT*, pages 135–156, 2010.
19. Amos Fiat and Adi Shamir. How to prove yourself: Practical solutions to identification and signature problems. In *CRYPTO*, pages 186–194, 1986.
20. Shafi Goldwasser, Yael Tauman Kalai, and Guy N. Rothblum. One-time programs. In David Wagner, editor, *Advances in Cryptology - CRYPTO 2008, 28th Annual International Cryptology Conference, Santa Barbara, CA, USA, August 17-21, 2008. Proceedings*, volume 5157 of *Lecture Notes in Computer Science*, pages 39–56. Springer, 2008.
21. Shafi Goldwasser and Guy N. Rothblum. Securing computation against continuous leakage. In *CRYPTO*, pages 59–79, 2010.
22. Jens Groth. Simulation-sound nizk proofs for a practical language and constant size group signatures. In *ASIACRYPT*, pages 444–459, 2006.
23. Jens Groth and Amit Sahai. Efficient non-interactive proof systems for bilinear groups. In *EUROCRYPT*, pages 415–432, 2008.
24. Yuval Ishai, Amit Sahai, and David Wagner. Private Circuits: Securing Hardware against Probing Attacks. In *CRYPTO*, pages 463–481, 2003.
25. Ali Juma and Yevgeniy Vahlis. Protecting cryptographic keys against continual leakage. In *CRYPTO*, pages 41–58, 2010.
26. Eike Kiltz and Krzysztof Pietrzak. Leakage resilient elgamal encryption. In *ASIACRYPT*, pages 595–612, 2010.
27. Paul C. Kocher, Joshua Jaffe, and Benjamin Jun. Differential power analysis. In *CRYPTO*, pages 388–397, 1999.
28. Allison Lewko, Mark Lewko, and Brent Waters. Achieving leakage resilience through dual system encryption. to appear at TCC 2011, 2011.
29. Allison Lewko, Mark Lewko, and Brent Waters. How to leak on key updates. to appear at STOC 2011, 2011.
30. 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.
31. Tatsuaki Okamoto. Provably secure and practical identification schemes and corresponding signature schemes. In *CRYPTO*, pages 31–53, 1992.
32. Torben P. Pedersen. Non-interactive and information-theoretic secure verifiable secret sharing. In *CRYPTO*, pages 129–140, 1991.
33. Krzysztof Pietrzak. A leakage-resilient mode of operation. In Antoine Joux, editor, *Advances in Cryptology - EUROCRYPT 2009, 28th Annual International Conference on the Theory and Applications of Cryptographic Techniques, Cologne, Germany, April 26-30, 2009. Proceedings*, volume 5479 of *Lecture Notes in Computer Science*, pages 462–482. Springer, 2009.
34. Jean-Jacques Quisquater and David Samyde. Electromagnetic analysis (ema): Measures and counter-measures for smart cards. In *E-smart*, pages 200–210, 2001.
35. Anup Rao. An exposition of bourgain's 2-source extractor. *Electronic Colloquium on Computational Complexity (ECCC)*, 14(034), 2007.
36. Amit Sahai. Non-malleable non-interactive zero knowledge and adaptive chosen-ciphertext security. In *FOCS*, pages 543–553, 1999.
37. Hovav Shacham. 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/`.
38. François-Xavier Standaert, Olivier Pereira, Yu Yu, Jean-Jacques Quisquater, Moti Yung, and Elisabeth Oswald. Leakage resilient cryptography in practice. *Towards Hardware Intrinsic Security: Foundation and Practice*, pages 105– 139, 2010.
39. Yu Yu, François-Xavier Standaert, Olivier Pereira, and Moti Yung. Practical leakage-resilient pseudorandom generators. In *ACM Conference on Computer and Communications Security*, pages 141–151, 2010.

A Information Theoretic Definitions and Lemmata

An important primitive in our work are randomness two-source extractors, which are formally defined as follows.

Definition 5 (Two-source Extractor [8]). *Let $\mathcal{L}, \mathcal{R}$ and $\mathcal{M}$ be three sets. A function $ext : \mathcal{L} \times \mathcal{R} \to \mathcal{M}$ is a* (k_0, k_1, ϵ)-two source extractor *if for random variables L over $\mathcal{L}$ and R over $\mathcal{R}$ with $\mathsf{H}_\infty(L) \geq k_0$ and $\mathsf{H}_\infty(R) \geq k_1$ we have that $d(ext(L,R)) \leq \epsilon$. It is a* strong (k_0, k_1, ϵ)*-two-source extractor if $d(ext(L,R)|L) \leq \epsilon$ and $d(ext(L,R)|R) \leq \epsilon$.*

We need some simple facts from information theory that are proven below.

Lemma 9. *Let $\epsilon > 0$ and let X, Y, Z be random variables such that $I(X; Z) = 0$, i.e. X and Z are independent, and $\Delta((X,Y);(X,Z)) \leq \epsilon$, then*

$$\tilde{\mathsf{H}}_\infty(X|Y,U) \geq \min(\mathsf{H}_\infty(X) - 1, \log \epsilon^{-1} - 1)$$

Proof. We prove this by contradiction. Suppose $\tilde{\mathsf{H}}_\infty(X|Y) < \min(\mathsf{H}_\infty(X) - 1, \log \epsilon^{-1} - 1)$, then we consider two cases: (a) $\mathsf{H}_\infty(X) \leq \log \epsilon^{-1}$ and (b) $\mathsf{H}_\infty(X) > \log \epsilon^{-1}$. Suppose (a) holds, then $\tilde{\mathsf{H}}_\infty(X|Y) < \mathsf{H}_\infty(X) - 1$, which means that there exists an algorithm $\mathcal{A}$ that given Y predicts X with probability at least $2^{-\mathsf{H}_\infty(X)+1}$, i.e.,

$$\Pr[\mathcal{A}(Y) = X] > 2^{-\mathsf{H}_\infty(X)+1}. \tag{35}$$

$$\tag{36}$$

Given such a $\mathcal{A}$ we can build a distinguisher D that contradicts $\Delta((X,Y);(X,Z)) \leq \epsilon$. D takes as input (X, U), where U is either Y or Z. It runs $\mathcal{A}(U)$ and if its output is equal to X it guesses 1; otherwise it returns 0. By Eq. 35, we get $\Pr[D(X,Y) = 1] > 2^{-\mathsf{H}_\infty(X)+1}$, while by independence of X and Z it follows that $\Pr[D(X,Z) = 1] = 2^{-\mathsf{H}_\infty(X)}$. Putting this together with $\mathsf{H}_\infty(X) \leq \log \epsilon^{-1}$, we get:

$$\Delta((X,Y);(X,Z)) > 2^{-\mathsf{H}_\infty(X)+1} - 2^{-\mathsf{H}_\infty(X)} \geq \epsilon,$$

which gives a contradiction.

Now suppose that (b) holds. In this case, we have $\tilde{\mathsf{H}}_\infty(X|Y) < \log \epsilon^{-1} - 1$, which gives us a predictor $\mathcal{A}$ with $\Pr[\mathcal{A}(Y) = X] > 2\epsilon$. With the same distinguisher as above, we get

$$\Delta((X,Y);(X,Z)) > 2\epsilon - 2^{-\mathsf{H}_\infty(X)} \geq 2\epsilon - \epsilon = \epsilon,$$

where the last inequality follows by (b). □

Lemma 10. *Let A, B be two random variables, then $\tilde{\mathsf{H}}_\infty(A|B) \geq \mathsf{H}_\infty(A,B) - \log |B|$.*

Proof.

$$\begin{aligned}
\tilde{\mathsf{H}}_\infty(A|B) &= -\log\left(\mathbb{E}_{b \leftarrow B} \max_a \Pr[A = a | b = b]\right) \\
&= -\log\left(\sum_b \max_a \Pr[A = a | B = b] \Pr[B = b]\right) \\
&= -\log\left(\sum_b \max_a \frac{\Pr[A = a, B = b]}{\Pr[B = b]} \Pr[B = b]\right) \\
&\geq -\log\left(|B| \cdot \max_{a,b} \Pr[A = a, B = b]\right) \\
&= -\log |B| - \log\left(\max_{a,b} \Pr[A = a, B = b]\right) \\
&= \mathsf{H}_\infty(A,B) - \log |B|.
\end{aligned}$$

□

Lemma 11. *Let A, B be two random variables, then $\mathsf{H}_\infty(A, B) \le \mathsf{H}_\infty(A) + \log |B|$.*

Proof.

$$\begin{aligned}
\mathsf{H}_\infty(A, B) &= -\log\left(\max_{a,b} \Pr[A = a, B = b]\right) \\
&= -\log\left(\max_{a,b} \Pr[A = a]\Pr[B = b|A = a]\right) \\
&\le -\log\left(\max_{a} \Pr[A = a]\,|B|^{-1}\right) \\
&= \mathsf{H}_\infty(A) + \log |B|\,.
\end{aligned}$$

□

We now recall some standard lemmata about the statistical distance (cf. e.g. [14]).

Lemma 12. *For any variables $X_1, \ldots, X_n$ we have $d(X_1, \ldots, X_n) \le \sum_i d(X_i|X_1, \ldots, X_{i-1})$.*

Lemma 13 (Triangle inequality for the statistical distance). *For any X, Y, Z we have that $\Delta(X; Z) \le \Delta(X; Y) + \Delta(Y; Z)$.*

Lemma 14. *For every X and Y and any function f we have $d(X|Y) \ge d(X|f(Y))$.*

B Basic Linear Algebra

Suppose $m < n$. Let $T \in \mathbb{F}^{m\times n}$ and $C \in \mathbb{F}^{m\times n}$. It is a well-known fact that if T and C have full rank then there always exists a solution $M \in \mathsf{NonSing}^{n\times n}(\mathbb{F})$ of the equation

$$T \cdot M = C. \tag{37}$$

Furthermore, we now show a method to sample such a solution uniformly at random.

Lemma 15. *Let T and C have full rank. There exists an efficient procedure that samples a solutions of the equation $T \cdot M = C$ uniformly at random.*

Proof. The procedure is given on Fig. 4. First, it is easy to see that this procedure indeed finds all the solutions

1. Let T' be an arbitrary non-singular $n \times n$-matrix such that the first m rows of T' are equal to T (since T has a full rank, such a matrix T' always exists).
2. Sample C' from a set of non-singular $n \times n$-matrices whose first m rows are equal to C and the other rows are chosen uniformly at random (since C has a full rank, such a matrix C' always exists)
3. Output $M' := (T')^{-1} C'$. Since M' is a product of two non-singular matrices hence M' is non-singular.

Fig. 4. A procedure for sampling a solution M of an equation $T \cdot M = C$. We assume that T and C have full rank.

of the $T \cdot M = C$ equation. This is because if M' is a solution of this equation, then for any T' (chosen is Step 1) we have that $C' = T' \cdot M'$ is a non-singular matrix (since it is a product of two non-singular matrices) such that its first m rows are equal to the first m rows of C (so C' belongs exactly to the set considered in Step 2 . Therefore, since in Step 3 we set $M' = (T')^{-1} C'$, hence this solution will always be found, and each solution has the same probability. □

Lemma 16. *Let $\ell, m, n \in \mathbb{N}$ be such that $m < n - \ell$. Consider the following way to sample a matrix $M \in \mathbb{F}^{m \times n}$: each row of M is sampled independently from an $n - \ell$ dimensional subspace of $\mathbb{F}^n$. The probability that a matrix M sampled in this way has full rank is at least $1 - m \cdot |\mathbb{F}|^{-n+m+\ell-1}$.*

Proof. For every $i \in [1, m]$, let $\mathcal{E}_i$ denote the event that the ith row of M does not belong to the linear sub-space spanned by the first $i-1$ rows of M. We clearly have $p = \Pr[\mathcal{E}_1] \cdot \dots \cdot \Pr[\mathcal{E}_m]$ (as all these events are independent). It is also easy to see that for each i we have $\Pr[\mathcal{E}_i] \geq 1 - |\mathbb{F}|^{-n+\ell+m-1}$. This is because for $i \leq m$ the space spanned by the first $i-1$ rows of M has a dimension at most $m-1$, and hence the probability that a random vector sampled from a $n - \ell$ dimensional subspace lies in the $m-1$ dimensional subspace is at least $|\mathbb{F}|^{-n+\ell+m-1}$. Hence $p \geq (1 - |\mathbb{F}|^{-n+\ell+m-1})^m$ which by Bernoulli's inequality is at least $1 - m\,|\mathbb{F}|^{-n+\ell+m-1}$. □

Notice that in the above Lemma if $\ell = 0$, then this means that $M \leftarrow \mathbb{F}^{m \times n}$ is sampled uniformly at random.

Lemma 17. *Consider the following experiments:*

Exp_0**:** *sample at random $T_0 \in \mathbb{F}^{m \times n}$ and $C_0 \in \mathbb{F}^{m \times n}$ of full rank, and then a random solution M_0 of $C_0 = T_0 \cdot M_0$,*

Exp_1**:** *sample at random $T_1 \in \mathbb{F}^{m \times n}$ and $M_1 \in \mathbb{F}^{n \times n}$ both of full rank, and then calculate $C_1 = T_1 \cdot M_1$.*

Then the distributions of (T_0, M_0, C_0) and (T_1, M_1, C_1) are identical.

Proof. Since C_0 is a function of T_0 and M_0, and C_1 is the same function of T_1 and M_1, it is enough to show that (T_0, M_0) and (T_1, M_1) have the same distribution. Variables (T_1, M_1) and T_0 have a uniform distribution, and therefore the only thing that remains to show is that the conditional distribution of M_0 given that T_0 is fixed is uniform. This easily follows from Lemma 15. Take any non-singular $k \times k$-matrix T_0' such that the first m rows of T_0' are equal to T_0. Let C_0' be a random non-singular $n \times n$-matrix whose first m rows are equal to C_0 and the other rows are chosen uniformly at random. Since C_0 is random, also C_0' is random. Therefore, since M_0' is a product of $(T_0')^{-1}$ and a random C_0', hence M_0' is also random.

We will also need the following well known facts from linear algebra.

Lemma 18. *Let A be a $(k \times m)$-matrix with rank m and B a $(m \times n)$-matrix then we have*

$$\mathrm{rank}(A \cdot B) = \mathrm{rank}(B) \tag{38}$$

$$\mathrm{rank}(A) = \mathrm{rank}(A^{\mathsf{T}}) \tag{39}$$

C Leakage-resilient storage

In this section we present the proof of Lemma 1. We start with the following auxiliary lemmata (that use lemmata and terms introduced in Appendix A)

C.1 Auxiliary lemmata

Lemma 19. *For any $\lambda \in \mathbb{N}$ and any λ-limited adversary $\mathcal{A}$, if L and R are independent, then L and R are independent conditioned on $Out(\mathcal{A}, \Omega(L, R))$, i.e.,*

$$I(L; R | Out(\mathcal{A}, \Omega(L, R))) = 0.$$

Proof. This directly follows from Lemma 4 in [15]. □

Lemma 20. *For $\delta > 0$, $\lambda \in \mathbb{N}$ and any λ-limited adversary $\mathcal{A}$, we have*

$$\Pr_{v \leftarrow Out(\mathcal{A}, \Omega(L,R))}[H_\infty(L | Out(\mathcal{A}, \Omega(L,R)) = v) \leq H_\infty(L) - \lambda - \log(1/\delta)] \leq \delta,$$
$$\Pr_{v \leftarrow Out(\mathcal{A}, \Omega(L,R))}[H_\infty(R | Out(\mathcal{A}, \Omega(L,R)) = v) \leq H_\infty(R) - \lambda - \log(1/\delta)] \leq \delta.$$

Proof. This directly follows from Lemma 2.2 in [13]. □

Consider any strong two-source extractor $ext : \mathcal{L} \times \mathcal{R} \to \mathcal{M}$, then we can define a $ext^m : \mathcal{L} \times \mathcal{R}^m \to \mathcal{M}^m$ as

$$ext^m(L, (R_1, \ldots, R_m)) = (ext(L, R_1), \ldots, ext(L, R_m)).$$

Lemma 21. *Let $ext : \mathcal{L} \times \mathcal{R} \to \mathcal{M}$ be a strong $(k_\mathcal{L}, k_\mathcal{R}, \epsilon)$-two source extractor (cf. Appendix A). For $m \in \mathbb{N}$ and any $\gamma > 0$ the function ext^m as defined above is a $(k_\mathcal{L}, (m-1)\log|\mathcal{R}| + k_\mathcal{R} + \log(1/\gamma), m(\epsilon + \gamma))$-two source extractor.*

Proof. We need to show that for sources L and $(R_1, \ldots, R_m)$ with $H_\infty(L) \geq k_\mathcal{L}$ and $H_\infty(R_1, \ldots, R_m) \geq (m-1)\log|\mathcal{R}| + k_\mathcal{R} + \log(1/\gamma)$, we have

$$d(ext(L, R_1), \ldots, ext(L, R_m)) \leq m(\epsilon + \gamma). \tag{40}$$

From Eq. (40), we get

$$\begin{aligned} d(ext(L, R_1), \ldots, ext(L, R_m)) &\leq \sum_{i \in [m]} d(ext(L, R_i) | ext(L, R_1), \ldots, ext(L, R_{i-1})) \\ &\leq \sum_{i \in [m]} d(ext(L, R_i) | L, R_1, \ldots, R_{i-1}). \end{aligned} \tag{41}$$

The first inequality follows by Lemma 12 and the second by Lemma 14 in Appendix B. We now need to upper-bound $d(ext(L, R_i) | L, R_1, \ldots, R_{i-1})$. To this end, from Lemma 11 in Appendix B, we have that for each $i \in [m]$:

$$\begin{aligned} \mathsf{H}_\infty(R_1, \ldots R_i) &\geq \mathsf{H}_\infty(R_1, \ldots, R_m) - (m-i)\log|\mathcal{R}| \\ &\geq (m-1)\log|\mathcal{R}| + k_\mathcal{R} + \log(1/\gamma) - (m-i)\log|\mathcal{R}| = (i-1)\log|\mathcal{R}| + k_\mathcal{R} + \log(1/\gamma). \end{aligned}$$

For $i \in [m]$ we use Lemma 10 in Appendix B, which gives us

$$\begin{aligned} \tilde{H}_\infty(R_i | R_1, \ldots, R_{i-1}) &\geq \mathsf{H}_\infty(R_1, \ldots R_i) - (i-1)\log|\mathcal{R}| \\ &\geq k_\mathcal{R} + \log(1/\gamma). \end{aligned}$$

Therefore, from Lemma 2.2 in [13] the probability that

$$H_\infty(R_i | R_1, \ldots, R_{i-1}) = (r_1, \ldots, r_{i-1}) \geq k_\mathcal{R} \tag{42}$$

is at least $1 - \gamma$. For each $i \in [m]$ let $\mathcal{E}(i)$ be an event that (42) holds, then as ext is a strong $(k_\mathcal{L}, k_\mathcal{R}, \epsilon)$-two extractor, for each $i \in [m]$ we get that $d(ext(L, R_i) | L, R_1, \ldots, R_{i-1}, \mathcal{E}(i)) \leq \epsilon$. From Lemma 2 in [9], we have

$$d(ext(L, R_i) | L, R_1, \ldots, R_{i-1}) \leq d(ext(L, R_i) | L, R_1, \ldots, R_{i-1}, \mathcal{E}(i)) + \Pr[\overline{\mathcal{E}(i)}] \leq \epsilon + \gamma.$$

With Eq. (41) this concludes the proof. □

Lemma 22. *Let $\gamma, \lambda > 0$ and let $ext : \mathcal{L}\times\mathcal{R} \to \mathcal{M}$ is an $(\log|\mathcal{L}|-\lambda-\log(1/\gamma)), \log|\mathcal{R}|-\lambda-\log(1/\gamma)), \epsilon)$-two source extractor satisfying the following property:*

If $L \leftarrow \mathcal{L}$ and $R \leftarrow \mathcal{R}$ are sampled uniformly at random, then $ext(L,R)$ is distributed uniformly over $\mathcal{M}$. (43)

Define an LRS $\Phi_{ext} = (\mathsf{Encode}_{ext}, \mathsf{Decode}_{ext})$ as follows:

- $\mathsf{Encode}_{ext} : \mathcal{M} \to \mathcal{L} \times \mathcal{R}$*: On input $S \in \mathcal{M}$ sample uniformly at random elements (L,R) such that $S = ext(L,R)$.*
- $\mathsf{Decode}_{ext} : \mathcal{L} \times \mathcal{R} \to \mathcal{M}$*:* $\mathsf{Decode}_{ext}(L,R) = ext(L,R) = S$*.*

Then Φ_{ext} is a $(\lambda, 2|\mathcal{M}|\epsilon - 2\gamma)$-secure LRS.

Proof. Suppose that (L,R) is sampled uniformly and independently from $\mathcal{L} \times \mathcal{R}$. Let $\mathcal{E}_{\mathcal{L}}$ be an event that

$$H_\infty(L|Out(\mathcal{A}, \Omega(L,R))) \le \log|\mathcal{L}| - \lambda - \log(1/\gamma)$$

and let $\mathcal{E}_{\mathcal{R}}$ be an event that

$$H_\infty(R|Out(\mathcal{A}, \Omega(L,R))) \le \log|\mathcal{R}| - \lambda - \log(1/\gamma).$$

From Lemma 20, we know that the probability of both of these events is at most γ, and hence, by the union-bound: $\Pr[\mathcal{E}_{\mathcal{L}} \cup \mathcal{E}_{\mathcal{R}}] \le 2\gamma$. Now, suppose $\mathcal{E} := \overline{\mathcal{E}_{\mathcal{L}} \cup \mathcal{E}_{\mathcal{R}}}$ occurred. For any $S, S' \in \mathcal{M}$ we have:

$$\begin{aligned}
\epsilon &\ge d(ext(L,R)|Out(\mathcal{A},\Omega(L,R)),\mathcal{E}) && (44)\\
&= \frac{1}{2}\sum_{S\in\mathcal{M}, w\in\{0,1\}^\lambda} |\Pr[ext(L,R) = S \wedge Out(\mathcal{A},\Omega(L,R)) = w \mid \mathcal{E}] - && (45)\\
&\qquad\qquad \Pr[Out(\mathcal{A},\Omega(L,R)) = w \mid \mathcal{E}]/|\mathcal{M}|\,| \\
&\ge \frac{1}{2}\sum_{w\in\{0,1\}^\lambda}\sum_{s\in\{S,S'\}} |\Pr[ext(L,R) = m \wedge Out(\mathcal{A},\Omega(L,R)) = w \mid \mathcal{E}] - \\
&\qquad\qquad \Pr[Out(\mathcal{A},\Omega(L,R)) = w \mid \mathcal{E}]/|\mathcal{M}|\,| \\
&\ge \frac{1}{2}\sum_{w\in\{0,1\}^\lambda} |\Pr[ext(L,R) = S \wedge Out(\mathcal{A},\Omega(L,R)) = w \mid \mathcal{E}] - && (46)\\
&\qquad\qquad \Pr\left[ext(L,R) = S' \wedge Out(\mathcal{A},\Omega(L,R)) = w \mid \mathcal{E}\right]| \\
&= \frac{1}{2|\mathcal{M}|}\sum_{w\in\{0,1\}^\lambda} |\Pr\left[Out(\mathcal{A},\Omega(L,R)) = w \mid ext(L,R) = S' \wedge \mathcal{E}\right] - && (47)\\
&\qquad\qquad \Pr\left[Out(\mathcal{A},\Omega(L,R)) = w \mid ext(L,R) = S' \wedge \mathcal{E}\right]| \\
&= \frac{\Delta(Out(\mathcal{A},\Omega(\mathsf{Encode}(S))); Out(\mathcal{A},\Omega(\mathsf{Encode}(S'))))|\mathcal{E})}{2|\mathcal{M}|} && (48)\\
&\ge \frac{\Delta(Out(\mathcal{A},\Omega(\mathsf{Encode}(S))); Out(\mathcal{A},\Omega(\mathsf{Encode}(S')))) - \Pr\left[\overline{\mathcal{E}}\right]}{2|\mathcal{M}|} && (49)\\
&\ge \frac{\Delta(Out(\mathcal{A},\Omega(\mathsf{Encode}(S))); Out(\mathcal{A},\Omega(\mathsf{Encode}(S')))) - 2\gamma}{2|\mathcal{M}|}. && (50)
\end{aligned}$$

Here, (44) follows from the definition of a two-source extractor, and the fact that L and R are independent given $Out(\mathcal{A}, \Omega(L,R))$ (this, in turn, follows from Lemma 19 and the fact that L and R were chosen independently at random, and hence $I(L,R) = 0$). Eq. (45) comes form the definition of the statistical distance (cf. (1) in Section 2.1), Eq. (46) comes from the triangle inequality, and (47) from the definition of the conditional probability and the fact that we assumed that ext satisfies Property (43). Again, (48) follows from the definition of the statistical distance, and (49) comes from the fact that the statistical distance is at most equal to 1. Clearly (50) implies that

$$\Delta(Out(\mathcal{A}, \Omega(\mathsf{Encode}(S))); Out(\mathcal{A}, \Omega(\mathsf{Encode}(S')))) \leq 2\,|\mathcal{M}|\,\epsilon - 2\gamma.$$

This finishes the proof. □

C.2 Proof of Lemma 1

In order to show Lemma 1 we use a technique that is an extension of the technique from [9]. In our proof we will use the fact that an inner product over a finite filed is a strong two-source extractor. This fact was first stated in the seminal work of Chor and Goldreich [8], where it was shown for a field $GF(2)$. Rao [35] extended this result and proved that the inner product over arbitrary finite fields $\mathbb{F}$ is a strong two-source extractor. More precisely, it was shown that for any finite field $\mathbb{F}$, any $\delta > 0$ and $n \in \mathbb{N}$ the function $ext^n_{\mathbb{F}} : \mathbb{F}^n \times \mathbb{F}^n \to \mathbb{F}$ (cf. [35], Theorem 3.1) defined as $ext^n_{\mathbb{F}}(L, R) = \langle L, R\rangle = \sum_i L_i R_i$ is a $(k_{ext}, \epsilon_{ext})$-two source extractor, for $\epsilon_{ext} = |\mathbb{F}|^{(n+1)/2}\, 2^{-k_{ext}}$ and $k_{ext} = (1/2 + \delta)n \log |\mathbb{F}|$, and $|\mathbb{F}| = \Omega(n)$. Here we generalize this fact in the following way.

Lemma 23. *For $n, m \in \mathbb{N}$ and a finite field $\mathbb{F}$ such that $|\mathbb{F}| = \Omega(n)$ define $ext^{n,m}_{\mathbb{F}} : \mathbb{F}^n \times \mathbb{F}^{n\times m} \to \mathbb{F}^m$ as $ext^{n,m}_{\mathbb{F}}(L, R) = L \cdot R$. For any $\delta > 0$ and $\gamma > 0$ we have that $ext^{n,m}_{\mathbb{F}}$ is a $((1/2 + \delta)n \log |\mathbb{F}|\,, (m - 1/2 + \delta)n \log |\mathbb{F}| + \log(\gamma^{-1}), m(|\mathbb{F}|^{(n+1)/2}\, 2^{-(1/2+\delta)n \log|\mathbb{F}|} + \gamma))$-extractor.*

Proof. Apply Lemma 21 to the extractor $ext^n_{\mathbb{F}}$. This can be done because we can view the function $ext^{n,m}_{\mathbb{F}} : \mathbb{F}^n \times \mathbb{F}^{n\times m} \to \mathbb{F}^m$ defined as $ext^{n,m}_{\mathbb{F}}(L, R) = L \cdot R$ as $\tilde{ext}^{n,m}_{\mathbb{F}} : \mathbb{F}^n \times (\mathbb{F}^n)^m \to \mathbb{F}^m$ defined as $\tilde{ext}^{n,m}_{\mathbb{F}}(L, (R_1, \ldots, R_m)) = (\langle L, R_1\rangle, \ldots, \langle L, R_m\rangle) = (ext^n_{\mathbb{F}}(L, R_1), \ldots, ext^n_{\mathbb{F}}(L, R_m))$. □

We now have the following.

Proof (of Lemma 1). We will combine Lemma 23 with Lemma 22. Recall that Lemma 22 states that every extractor can be converted into an LRS scheme, as long as it satisfies property (43). The extractor $ext^{n,m}_{\mathbb{F}}$ constructed in Lemma 23 does not satisfy (43) if it is consider over the domain $\mathbb{F}^n \times \mathbb{F}^{n,m}$. This is because for $L = (0^n)$ the value of $ext^{n,m}_{\mathbb{F}}(L, R)$ is equal to (0^m) for any R. It is easy to see, however, that if $ext^{n,m}_{\mathbb{F}}$ is considered over a restricted domain $(\mathbb{F}^n \setminus \{(0^n)\} \times \mathbb{F}^{n,m}$ then the property (43) is satisfied[7]. Therefore the LRS $\Phi_{ext^{n,m}_{\mathbb{F}}}$ (cf. Lemma 22) is (λ, ϵ)-secure (if considered over the restricted domain), with

$$\lambda = (1/2 - \delta)n \log |\mathbb{F}| - \log \gamma^{-1}$$
$$\epsilon = 2m(|\mathbb{F}|^{m+1/2-n\delta} + |\mathbb{F}^m|\,\gamma).$$

This finishes the proof since $\Phi_{ext^{n,m}_{\mathbb{F}}}$ is exactly the same as $\Phi^{n,m}_{\mathbb{F}}$. □

D Flawed constructions of a refreshing scheme

In this section we present insecure protocols for refreshing and attacks against them. We do it for two reasons: first, the insecure protocol $\mathsf{FlawedRefresh}$ is easier to understand and already outlines the basic idea of the more complicated (secure) refreshing scheme $\mathsf{Refresh}$. Second, on first sight $\mathsf{FlawedRefresh}$ looks like a secure method for refreshing the LRS. Somewhat surprisingly, as we will show in this section, the protocol $\mathsf{FlawedRefresh}$ can be completely broken (i.e., the adversary can recover the encoded secret S) with a simple, *non-adaptive* leakage attack (essentially, the only thing that we require is that the adversary can probe some parts of the memory when the computation takes place).

[7] This is because if $L \neq (0^n)$ then for every $X \in \mathbb{F}^m$ the set of solutions of $L \cdot R = X$ is an affine subspace of $\mathbb{F}^{n\times m}$ of dimension $m(n-1)$, and hence has a cardinality $|\mathbb{F}|^{m(n-1)}$.

D.1 A First Attempt for Refreshing the LRS $\Phi_{\mathbb{F}}^{n,m}$

To simplify exposition, we focus in this section on the case when $m = 1$. The protocol $\mathsf{FlawedRefresh}_{\mathbb{F}}^{n,1}(L, R)$ is given in Figure 5. The inner box inside of the protocol description is not part of the protocol, but only needed to illustrate the leakage game that the adversary can play when running the protocol $\mathsf{FlawedRefresh}_{\mathbb{F}}^{n,1}(L, R)$.

Protocol $(L', R') \leftarrow \mathsf{FlawedRefresh}_{\mathbb{F}}^{n,1}(L, R)$:

Input (L, R)**:** L is given to P_{L} and R is given to P_{R}.

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

1. P_{L} samples X such that $\langle X, L\rangle = 0$ and sends X to P_{R}.
2. P_{R} sets $R' := R + X$.

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

3. P_{R} samples Y such that $\langle Y, R'\rangle = 0$ and sends Y to P_{R}.
4. P_{L} sets $L' := L + Y$.

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

The adversary plays a λ-leakage game against the leakage oracle:

$$\Omega\left((L, X, Y)\ ;\ (R, X, Y)\right)$$

Fig. 5. The flawed protocol for refreshing $\mathsf{FlawedRefresh}_{\mathbb{F}}^{n,1}$. The text in frames describes the leakage game played by the adversary.

It is easy to see that the protocol satisfies the correctness property as

$$\langle L', R'\rangle = \langle L + Y, R\rangle = \langle L, R'\rangle + \langle Y, R'\rangle = \langle L, R'\rangle,$$

where the first equality comes from the construction of the protocol, the second one from the linearity of the inner product, and the last one from the fact that $\langle Y, R'\rangle = 0$. By a symmetric reasoning we get $\langle L, R'\rangle = \langle L, R\rangle$. Hence, the protocol is correct. Notice that the refreshing protocol from Figure 5 does not depend on any message sent by the adversary, thus, we may assume that the adversary plays the leakage game just before the end of the protocol. This game is depicted in the text in the frame on Fig. 5. Furthermore, notice that as L' and R' are functions of (L, X, Y) and (R, X, Y) respectively, we do not explicitly need to include them into the inputs of the oracle Ω.

At first glance, the protocol shown in Figure 5 looks like a promising candidate for a secure refreshing protocol, as the secrets L and R held by the players get completely "refreshed" by adding to them a high-entropy vectors Y and X (resp.). In the attack below, we will show that unfortunately such a simple refreshing protocol is not secure.

AN ATTACK AGAINST THE PROTOCOL $\mathsf{FlawedRefresh}$ FROM FIG. 5. In the following, if $V = (V_1, \ldots, V_n)$ is a vector then $V[a, \ldots, b]$ (for $1 \leq a \leq b \leq n$) will denote the vector $(V_a, \ldots, V_b)$. We show a $(2\,|\mathbb{F}|)$-limited adversary $\mathcal{A}$ that attacks $\mathsf{FlawedRefresh}_{\mathbb{F}}^{n,1}$ and recovers the encoded message S after $\ell = n$ rounds of the refreshing protocol. Let L^{j-1} and R^{j-1} denote the shares held by the players P_{L} and P_{R} at the beginning of the jth round. Let further X^j, Y^j denote the randomness sampled for the refreshing in the jth round of the protocol. This means that in the jth round the adversary can interact with the leakage oracle

$\Omega(L^{j-1}, X^j, Y^j; R^{j-1}, X^j, Y^j)$. We will show how a $(2\,|\mathbb{F}|)$-limited adversary $\mathcal{A}$ can by interaction with $\Omega(L^{j-1}, X^j, Y^j; R^{j-1}, X^j, Y^j)$ learn $W^j := \langle L^j[1,\dots,j], R^j[1,\dots,j]\rangle$. Clearly, after showing this we are done, as for $j := n$ the adversary can recover $S = W^n$.

The attack works by induction. Trivially for $j = 1$ we can compute W^1 by retrieving from the leakage oracle $L^1[1]$ and $R^1[1]$. For $j \geq 2$ assume that at the end of the $(j-1)$th iteration the adversary knows W^{j-1}, and we will show how the adversary can learn W^j at the end of the jth iteration. At the end of the jth iteration the adversary $\mathcal{A}$ retrieves the following from P_{L}: the value of $C^j_L = \langle L^{j-1}[1,\dots,j-1], X^j[1,\dots,j-1]\rangle$ and $D^j_L = L^j[j]$, and from P_{R} he retrieves $C^j_R := \langle Y^j[1,\dots,j-1], R^j[1,\dots,j-1]\rangle$, and $D^j_R = R^j[j]$. We now have:

$$\begin{aligned}
W^j &= \langle L^j[1,\dots,j], R^j[1,\dots,j]\rangle && (51)\\
&= \langle L^j[1,\dots,j-1], R^j[1,\dots,j-1]\rangle + L^j[j]\cdot R^j[j]\\
&= \langle L^j[1,\dots,j-1], R^j[1,\dots,j-1]\rangle + D^j_L D^j_R\\
&= \langle L^{j-1}[1,\dots,j-1], R^j[1,\dots,j-1]\rangle + \langle Y^j[1,\dots,j-1], R^j[1,\dots,j-1]\rangle + D^j_L D^j_R && (52)\\
&= \langle L^{j-1}[1,\dots,j-1], R^j[1,\dots,j-1]\rangle + C^j_R + D^j_L D^j_R\\
&= \langle L^{j-1}[1,\dots,j-1], R^{j-1}[1,\dots,j-1]\rangle + \langle L^{j-1}[1,\dots,j-1], X^j[1,\dots,j-1]\rangle + C^j_R + D^j_L D^j_R\\
&= W^{j-1} + C^j_L + C^j_R + D^j_L D^j_R, && (53)
\end{aligned}$$

where (52) and (53) come from the linearity of the inner product, and the rest are simple algebraic transformations. By assumption, the adversary knows W^{j-1} in (53). Therefore he can also calculate W^j.

A FIRST ATTEMPT TO FIXING THE PROBLEM. The main problem with the protocol from Fig. 5 is that the players P_{L} and P_{R} know the values of X and Y (resp.), and therefore they can calculate joint functions on (L, X, Y) and (R, X, Y) (resp.). Our method for overcoming this problem is to design a scheme, where X and Y are sent in an "oblivious" way: e.g. the player P_{R} will be able to learn a random X such that $\langle L, X\rangle = 0$, while P_{L} will not learn any additional information on X. One natural idea to implement this, is the following. In Step 1, instead of choosing X randomly, P_{L} chooses a random matrix M in such a way that each column of M is orthogonal to L, which can be expressed algebraically as:

$$L \cdot M = (0,\dots,0). \qquad (54)$$

(for simplicity assume that M is a $n \times n$-matrix of rank $n-1$). Then P_{L} sends M to P_{R}. In Step 2 player P_{R} calculates $X := M \cdot B$, where B is a random column vector of length n (in other words: X is a random linear combination of the columns of M), and sets $R' := R + X^{\mathsf{T}}$. By simple linear algebra we have $\langle L, X\rangle = \langle L, M\cdot B\rangle = L \cdot M \cdot B = (0,\dots,0)\cdot B$. Symmetrically, in Step 3 player P_{R} chooses a random $n\times n$-matrix $\tilde{M}$ (of rank $n-1$) such that $\tilde{M}\cdot R' = (0,\dots,0)$. Then, P_{R} sends $\tilde{M}$ to P_{R}. In Step 4 player P_{L} calculates $Y := \tilde{A}\cdot\tilde{M}$, where $\tilde{A}$ is a random column vector of length n, and then he sets $L' := L + Y$. Again, it can be easily verified that $\langle R', X\rangle = 0$. This protocol may look secure since obviously P_{L} does not learn X, as (from his point of view) it is a random element sampled from a linear space of vectors orthogonal to L. Unfortunately, now the problem appears on the other side, since P_{R}, after receiving M, can solve (54) and compute L (and similarly P_{L} can compute R from $\tilde{M}$). Hence, this protocol is not better than an (obviously not secure) solution where the players simply send their shares L and R to each other. We also note that attempts to repair this problem by decreasing the rank or the dimension of M and $\tilde{M}$ does not seem to lead to any solution.