Privacy with Leakage from Falsifiable Assumptions

Francesco Berti[1], Itamar Levi[2]
francesco.berti@mil.be,itamar.levi@biu.ac.il

[1] Department of Mathematics, Royal Military Academy, Rue Hobbema 8, B 1000 Brussels, Belgium
[2] Bar-Ilan University, Ramat-Gan, Israel

Abstract. An inherent challenge in cryptography is finding realistic and meaningful models and assumptions. Such assumptions must be falsifiable. For symmetric-key cryptography in the standard model (i.e., black-box without leakage), it is conjectured that we have identified the right assumptions: pseudorandom functions and permutations (PRFs and PRPs). However, when considering side-channel attacks (SCAs) that exploit physical side-channel leakage, the answer is less clear. Moreover, the requirements for *integrity* and *privacy* may differ in the leakage-aware SCA setting. For *integrity*: due to advances in the so-called *unbounded leakage* model, it is sufficient to claim integrity based only on black-box properties by instantiating a single strongly unpredictable-with-leakage (tweakable) block cipher. For *privacy*: one approach relied on *simulatability*, proposed by Yu and Standaert at CRYPTO 2013. That is, it should be possible to simulate leakage traces without access to the underlying secret key. Unfortunately, simulating the leakage behavior of a complex electronic system and arguing that distinguishing simulation from reality is hard has proved highly challenging (e.g., Galea et al., ASIACRYPT 2014, and Deperz et al., Cascade 2026).

In this paper, we propose a new model and assumption. We isolate a key component, namely the key-refreshing process, and provide a security definition for it that is both falsifiable in practice and usable in security proofs: *secure refreshing*. Like simulatability, this assumption is designed to capture the security of the key-refreshing process. Rather than relying on a simulator, however, we formalize security through a direct refreshing experiment that can be practically falsified by evaluation laboratories. Therefore, our approach avoids many of the concrete problems and challenges associated with previous models. Furthermore, inspired by this new assumption (and its accompanying model), and to provide an additional security assumption for the strongly SCA-protected *leak-free* component, we introduce *strongly protected secure refreshing*. As one concrete example, using these assumptions, we prove the privacy of the well-known PSV encryption scheme. We provide the first privacy proof based only on *falsifiable assumptions* and, importantly, without assuming that the strongly protected component is *leak-free* which can effectively translate to significant computational cost reduction.

Keywords: Falsifiable-assumptions, Leakage-resilient cryptography, Leak-Free Computation, Privacy, Secure-Refreshing, Simulatability, Symmetric key

1 Introduction

In modern cryptography, it is a foundational principle that the security of a scheme should rely solely on the secrecy of its key, rather than on the obscurity of its design [29]. This principle, motivates the use of well-defined cryptographic primitives grounded in explicit and testable assumptions. In the setting of symmetric cryptography, where communicating parties share a common secret key, widely accepted security notions include pseudo-random functions (PRFs) and pseudo-random permutations (PRPs). These abstractions capture the idea that a keyed function should be computationally indistinguishable from a truly random function or permutation. Importantly, there exist practical constructions, such as the Advanced Encryption Standard (AES), that are widely believed to realize these notions in practice. Consequently, many cryptographic schemes can be analyzed and proven secure under the assumption that their underlying primitives behave as ideal PRFs or PRPs, and that the only secret component is the key itself. A key strength of these assumptions lies in their *falsifiability*: their validity can be challenged and potentially refuted through the discovery of concrete attacks. This makes them both scientifically meaningful and practically robust as a foundation for secure system design.

However, such an abstraction is only valid in the *black-box scenario*, i.e., the adversary interacts with the encryption scheme only by choosing its inputs and seeing its outputs. Conversely, for adversaries having access to devices: performing the crypotographic computations on a physical (typically electronic) device, can observe the physical quantities involved in these computations, as time, electro-magnetic radiation and instantaneous power consumption, the so called *side-channel*s. A side-channel attack can reveal the key even with only a single execution measurement [30,32,44], denoted in the jargon by a *trace* or *leakage*.

Countermeasures against such leakage-aided attacks works at different abstraction levels (e.g., [6]). Importantly each countermeasure typically necessitates different assumptions and different models to base security claims: 1) at the hardware level it is possible to reduce the side-channel information the adversary can retrieve, e.g., [49,37]; 2) at the implementation level, the decurity achieved by hardware-level countermeasures can be amplified, for example via secret-sharing (i.e.,*masking*) and shuffling [14,50,33]; 3) As the two previous countermeasures are quite expensive (a PRF implementation protected with a high-order *masking* scheme can induce significant electronic-performance overheads [24,45,13]), there is a complementary line of research: designing primitives and modes-of-operation that are inherently more secure. This is typically denoted as *mode-level* SCA-security, from seminal ideas [31,42,38], later formalized in the framework of leakage-resilient cryptography [22]. Following this line of research, many efficient Pseudorandom Number Generators (PRGs) and Stream Ciphers

have been proposed [43,52,47,48,23,1,21,28]. Concretely, to achieve SCA-security they crucially rely on key evolution, called also *key-refreshing*. Therefore, they are secure as long as the leakage is "bounded", and this is achieved by frequent key refreshes allowing an adversary to obtain at most few traces per key used. Following this line of research, Pereira et al. [41] introduced the concept of *leveled implementation* to create a MAC and an encryption scheme, PSV (see Fig. 1) that is leakage resistant: They assumed that there exists a strongly-SCA-protected primitive [3], usually a PRF or a PRP that is so well protected (e.g., protected with a high order masking scheme) that nothing leaks. Thus, we call and model it as *leak-free*. Note that from a security perspective, this is quite a questionable and restrictive model [4]. This primitive is the only one that uses the *master key* (that is, the key of the scheme) and produces an *ephemeral key* that is used as a key to a CBC-MAC or a leakage-resilient stream cipher. In this way, we obtain reasonably efficient and secure schemes, because we use the strongly-protected (and thus very slow) PRF only once per encryption/tag-generation query, leaving the bulk of the computations to the much more efficient CBC-MAC or stream-cipher (the encryption scheme) that uses primitives that are mildly protected against SCAs. Such a mode level countermeasure has been very popular with many proposed schemes using it: MAC, Encryption and Authenticated Encryption schemes [11,12,5,8,9,19,18,25,28].

Defining security in the presence of leakage requires modeling it. There are two main models *memory leakage* and *circuit leakage*. In both cases, the leakage an adversary obtains is represented via a *leakage function*. In the memory leakage model, the leakage function takes as input the secret state of the circuit [2]. In the circuit leakage model, it is assumed that side-channel information comes from the values created during a computation (the famous paradigm "Only Computation Leaks", OCL [39]), therefore, the leakage function takes as input the complete state of the circuit including internal computations [39,27]. The choice of the leakage function is a crucial and daunting task: it should not give more information than what actually leaks (as this would prevent the possibility that a scheme is proved to be secure[5]). But it should also model realistically what an adversary can actually obtain with her measurements. Unfortunately, providing a rigorous model is hard: the trace is the result of complex internal *analog* interactions (Maxwell's electromagnetism and Kerchoff's current and voltage equations) happening typically within billions of transistors devices. Due to the complexity of concrete electronic circuits (plurality of complex wires and gates topologies) and their physical interactions, finding, and then computing, a leakage function that maps the initial state of the device (that contains the key and the input of the scheme) to the actual leakage trace seems out of reach.

[3] Typically, in the jargon we omit the SCA and use *strongly-protected* to be understood from the context

[4] I.e., how much SCA security should be embedded so as to claim leak-freeness? Can this claim always be verified or falsified?

[5] For example, if the leakage function reveals the whole key with only a single trace

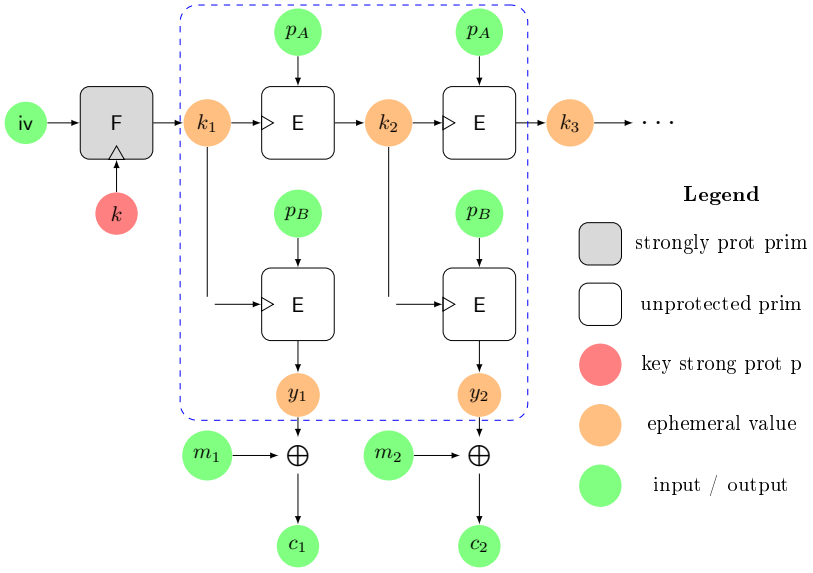


Fig. 1: The PSV encryption scheme [41]. Note that the master key is only used by the strongly protected block cipher F, while the mildly protected block ciphers E use the ephemeral keys $k_1, \ldots$. Each of these keys is used only twice. It uses the stream-cipher introduced in [52], that we highlight in the blue box.

Various proposition has been proposed for modeling the leakage-function, *bounded length for the output* [39], *indistinguishability*: that is, given the leakage, the secret state retains enough entropy [22], *hard-to-invert*: that is, given the leakage, the adversary cannot retrieve the secret state [20]. When multiple primitive calls are used in a scheme (consider for example the case of the stream cipher depicted in Fig. 1) the situation is even more complex: Assume that the leakage function reveals one bit of information per block-cipher call (therefore, we are in the bounded lenght output case, and we have the strictest possible bound). If we do not put any additional bound on the leakage function the PSV construction is insecure: It is enough to consider a leakage function that on input $(p_A; k_i)$ generates the following leakage associated to the computation $\mathsf{E}_{k_i}(p_A)$, $k_{i,128}$, the i^{th}-bit of the the 128^{th} refreshed key, k_{128}[6]. Therefore, k_{128} would be completely exposed, rendering the stream cipher insecure (note that from k_i and p_A it is possible to compute all following keys). On the other hand, such an attack seems irrealistic to do in practice, as all side-channel attacks try to recover information about the states used in the current computations. Notably, against a block-cipher, the usual target are the first and/or the last rounds due to rapid SPN (Substitution Permutation Nerwork) code-diffusion, see as an example [15,34,46,51,52]: it shows first that constructions which seem to be secure

[6] We assume that the block-cipher has 128-bit keys for simplicity, as AES does

cannot be proved to be secure even in one of the most constraining models (leakage function bounded to a single bit), on the other hand, this plausible insecurity is often exposed only due to leakage functions that are associated with unrealistic attacks.

Therefore, to prove the security of a cryptographic scheme based on rekeying many different attempts have emerged in the literature. First 1), the *alternating approach* [22]: substantially three different keys are refreshed, but during each step, one key is idle (and therefore, cannot be input of the leakage function). In this way, the leakage function cannot predict future keys because it does not have access to the idle key. Second 2): Since we need to prevent unrealist attacks, Yu et al. [52] have proposed a leakage-function model that matches real attacks on block-ciphers (i.e. on the first and/or on the last rounds): they assume that the leakage function of the blockcipher computation $y = \mathsf{E}_k(x)$, $\mathsf{L}_\mathsf{E}(x;k)$ has the following shape: $\mathsf{L}_\mathsf{E}(x;k) = (\mathsf{L}_\mathsf{E}^{in}(x;k), \mathsf{L}_\mathsf{E}^{out}(y;k))$ That is, the adversary receives a function of the manipulation of the input and secret key, which is done in the first rounds of block-ciphers, and a function of the manipulation of the output and secret key, which is done in the last rounds. Furthermore, they assume that having access to such leakage functions, an adversary cannot *guess* the key used by the block-cipher q times (this is the so-called *q-unpredictability with leakage* [$-\mathsf{upL}q$]). Finally, they model the block-cipher as an ideal cipher, and that the leakage-functions do not have oracle access to it (this hypothesis is realistic as the leakage functions is supposed to represent *natural* physical leakage, not a computation). In this way, the previous attack is prevented as the leakage function cannot compute the future keys since this a quarry to the ideal-cipher. Such an approach has found much success in the literature, being used to assess the security of Authenticated-Encryption, AE, schemes [8] and oblivious transfer schemes [10], for example.

On the other hand, using an ideal assumption, the ideal cipher, poses some problems: First, the ideal-cipher model constitutes a non-standard assumption in any leakage-aware context. Second, with leakage the adversary is facing an actual implementation of a block-cipher, whose leakage depends on the actual computations, but it is only modeled as an ideal object (which is far from the case): leakage is generated throughout the computation of (e.g.) a block-cipher on all internal states and it carry correlative information within the leakage typically on a long history of states. To prevent the use of such idealistic assumptions which are clearly restrictive, Standaert et al. [47] introduced the *simulatability assumption*: a simulator S can create a leakage trace indistinguishable from the real leakage trace $\mathsf{L}_\mathsf{E}(x;k)$ of the computation $y = \mathsf{E}_k(x)$ using only the input x and the output y (in addition the adversary can obtain a leakage trace of the generation of k). Since this simulator does not have access to the key used by the block-cipher, the existence of such a simulator allows us to replace the actual outputs of the block-cipher with random values in an indistinguishable way. If it is possible to build such a simulator it would allow to construct and prove the security of encryption schemes in the presence of leakage, see [41] for example. Unfortunately, up to now, there exists no simulator that is able to produce such

an indistinguishable leakage trace. The original simulator proposed by Standaert et al. [47], that mixes two leakage traces, $\mathsf{L}(x; k^*)$ and $\mathsf{L}(x'; k^*)$ where $y = \mathsf{E}_{k^*}(x')$ for a random key k^* gluing them in the middle (or transitioning gradually from one to another), has been broken by Longo et al. [36] using the trace consistency: there are coupling effects between operations that are close in time or area that this splitting and concatenate approach loses. This has been confirmed by a recent work by Deprez et al. [17] which worked on newer technology. As the authors [17] claim to perfectly simulate leakage we need "an extremely accurate modeling of the leakage function". But the exact modeling of the leakage function is precisely what the simulatability approach wants to avoid. This is illustrated in Fig. 2.

Finally, Deprez et al. [17] propose to combine the split and concatenate simulator with a theoretical simulator that comes from the line of research that reduces the noisy leakage model to the bounded leakage model [40,4]. On the other hand, this line of research only ensures the *existence* of a simulator for noisy leakage using a certain amount of bounded leakage. Therefore no simulator according to this approach has been built and we cannot falsify the simulatable assumption.

Therefore, theoretically we are still left with a significant problem of finding an appropriate replacement for simulatability or realistic leakage modeling in the standard model *that can be falsified by a laboratory.*

In addition to modeling challenges, in this paper we also touch the problem of the assumption for the strongly protected primitive, the *leak-free*ness assumption. In all the papers where *privacy* is considered, it is assumed that the strongly protected primitive is leak-free, see for example [41,8,12,5]. This is not the case for *integrity* [7] where Berti et al. [7] has introduced *strongly unpredictable with leakage*, SUP–L2, which assumes that with leakage, it is hard to predict a fresh and valid couple input/output for the strongly protected blockcipher. This notion is in the standard model and is falsifiable, as compared with *leak-free*ness claims. Although the first MAC, Hash-then-BC, $\tau = \mathsf{F}_k(\mathsf{H}(m))$, where in verification we invert the BC F and we check whether $\mathsf{H}(m) \stackrel{?}{=} \mathsf{F}_k^{-1}(\tau)$ models the hash function as a random oracle[8] assumes the random oracle model for technical reason [7], it has been proved that a MAC can achieve *integrity* in the presence of leakage only using a SUP–L2 tweakable block-cipher (and a collision resistant hash function) [9]. Therefore, we have found a good falsifiable assumption for strongly protected primitives when our goal is integrity. This is because we are interested that an adversary cannot *completely predict* a good tag. We still lack an appropriate falsifiable assumption when our goal is confidentiality since we want the ciphertext to look random. If an adversary can obtain some bits of information about a tag, this is not a problem; on the contrary a single bit of

[7] With ambiguity we use *integrity* and *authenticity* where it is clear from the context.

[8] Note that we do not care how much the hash function leaks, because its input is chosen by the adversary in the unforgeability game, therefore the adversary can compute its output by himself.

leakage about the plaintext (or considering PSV, Fig. 1 a single bit of the pseudorandom stream $y_1, y_2, \ldots$) is enough to break confidentiality. We depict these assumptions in Fig. 2.

1.1 Our Contribution

In this paper, we provide two assumptions to solve the two aforementioned problems: *secure refreshing*, SecRef and *strongly protected secure refreshing*, SpSecRef. These assumptions can be used to model and claim security of a construction in regards with leakage, and later on be used in experimental laboratories to opt for falsification [9]. Instead of working in the mode-level, using claims over block-cipher calls, our idea is to give a security definition regarding the basic structure of refreshing:

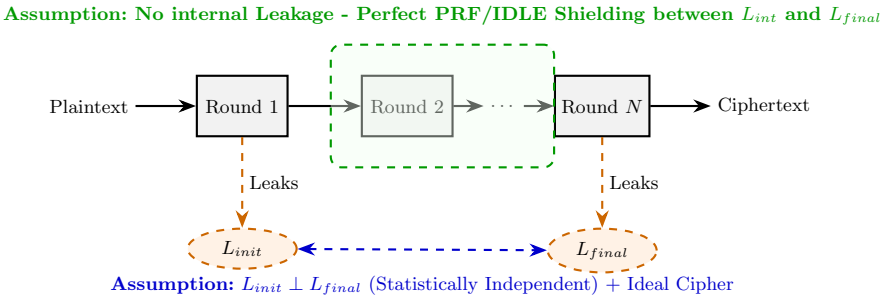
- **To replace simulatability:** we introduce the SecRef assumption- an adversary cannot distinguish via leakage if in the refreshing we are using, the actual key produced by the previous call to the block-cipher or a random key. Looking at Fig. 1, we consider the four calls to the weakly-protected block-cipher E. We assume that it is indistinguishable whether we use $\mathsf{E}_{k_2}(\cdot)$ or $\mathsf{E}_{\bar{k}_2}(\cdot)$ to compute k_3 and y_2, where $\bar{k}_2$ is a randomly picked key. That is, the leakage from the computation $k_2 = \mathsf{E}_{k_1}(p_A)$ is independent indistinguishable from the leakage of the computations $k_3 = \mathsf{E}_{k_2}(p_A)$ and $y_2 = \mathsf{E}_{k_2}(p_B)$.
- **To replace the leak-free assumption:** we look at the concatenation of a strongly protected primitive and a weakly protected primitive. The SpSecRef assumption assumes that it is indistinguishable if the weakly protected primitive uses or not the output of the strongly protected primitive. We assume that the adversary can see many of these concatenations but every concatenation only once (therefore there is no reuse of the ephemeral keys). Looking at Fig. 1, we are considering the concatenation $k_1 = \mathsf{F}_k(\mathsf{iv})$ with $k_2 = \mathsf{E}_{k_1}(p_A)$. We claim that we can replace k_1 in the latter computation, with a randomly picked key $\bar{k}_1$. As we use F many times (with different inputs), we claim that we can do this replacement for every execution of F. This means that the leakage of F does not reveal any information about the key used k, the master key, even if the adversary has access to multiple execution of this refreshing step.

We abstract and illustrate these indistinguishability definitions in Fig. 3.

Furthermore, we propose an experiment that can be made by an evaluation laboratory to assess if the implementations of the weakly protected primitive (and of the strongly protected primitive) achieves them.

We prove the privacy in the presence of leakage of the PSV encryption scheme (introduced by Pereira et al. [41]) using these two assumptions.

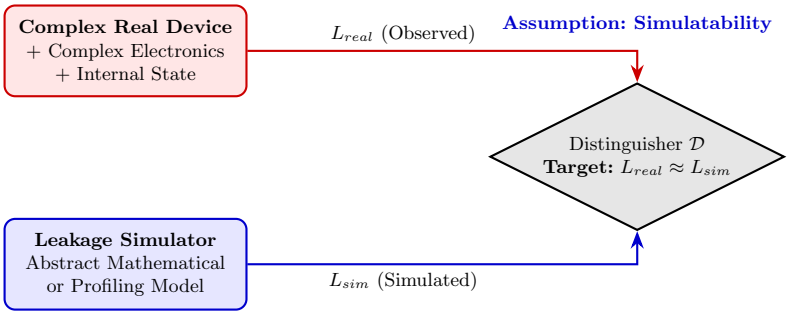
[9] Similarly to linear cryptanalysis on a block-cipher modeled as a PRP and tested upon with indistinguishability claims



Reality vs. Assumption – Y. Yu et al. (CCS 2010):

× **Unrealistic Silence:** In physical devices, *all* rounds leak, not just initial and final rounds.
× **Unrealistic Independence:** Physical leakages across different rounds are correlated and dependent on the underlying state/hardware.
× **Lack of Graceful Degradation:** The security proofs fail catastrophically if the middle rounds exhibit even partial leakage, which is unavoidable in practice.
× **Use of the Ideal Cipher Model:** This definition works *only* if we model the block-cipher as ideal.

(a) Assumption of confined, independent leakages restricted strictly to outer block cipher rounds.



Reality vs. Assumption – F. Standaert et al. (CRYPTO 2013):

× **Simulator Construction is Hard:** Achieving indistinguishability ($L_{real} \approx L_{sim}$) for complex electronic systems is practically hard [36,17]
× **Physical Variability:** Unavoidable manufacturing process variations prevent a generic simulator from accurately mirroring specific hardware instances.
× **Variability of Real and Model:** Process variations but also temporal and environmental inconsistencies. Operating conditions like temperature, voltage fluctuations, and component aging alter the leakage profile dynamically, making it nearly impossible to mimic consistently.

(b) Assumption of empirically verifiable simulatability generating indistinguishable leakage.

Fig. 2: Comparison of theoretical leakage assumptions against practical hardware realities in leakage-resilient cryptography.

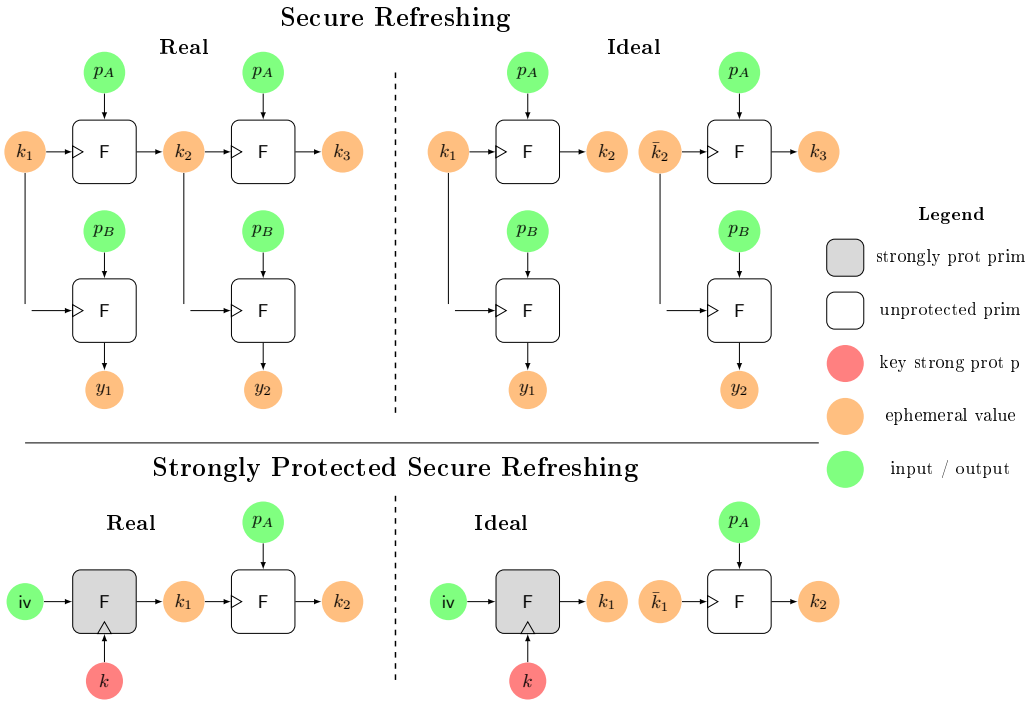


Fig. 3: The secure refreshing and the strongly protected secure refreshing security definitions. Note that the real part of the secure refreshing corresponds to the dotted blue box in Fig. 1, while the strongly protected secure refreshing corresponds to the generation on the first ephemeral key and its refresh in PSV.

2 Background

Notations. Let $\{0,1\}^n$ be the set of all n-bit strings. Given a string x, we denote with $|x|$ its length. Given $L \in \mathbb{N}$, $l \leq |x|$, we denote with $\pi_L(x)$, the string composed by the L leftmost bits of x. We denote that x is picked uniformly at random from the set $\mathcal{X}$, with $x \xleftarrow{\$} \mathcal{X}$.

When we *parse* a string x into n-bit blocks, we divide x into $(x_1, \ldots, x_l)$ with $|x_1| = \ldots = |x_{l-1}| = n$ and $0 < |x_l| \leq n$, where $x = x_1 \| \ldots \| x_l$ ($x \| y$ denotes the concatenation of the strings x and y).

A $(q_1, \ldots, q_d, t)$-adversary A is a probabilistic algorithm that can make q_i queries to oracle $\mathcal{O}_i$ and runs in time bounded by t. By $y \leftarrow \mathsf{A}^{\mathcal{O}_1,\ldots,\mathcal{O}_d}(x)$, we denote that the adversary A, on input x, with access to oracles $\mathcal{O}_1, \ldots, \mathcal{O}_d$, outputs y. Let Alg be a probabilistic algorithm.

2.1 Cryptographic Primitives used - Block-Ciphers (BC)

Block-ciphers. We use block-ciphers (BC) to generate pseudorandom values. It is a family of functions $\mathsf{F} : \mathcal{K} \times \{0,1\}^n \to \{0,1\}^n$ is a family of functions s.t. for all

$k \in \mathcal{K}$, the function $\mathsf{F}(k,\cdot) : \{0,1\}^n \to \{0,1\}^n$ is a permutation. For simplicity, we often write $\mathsf{F}_k(x)$ instead of $\mathsf{F}(k,x)$. We denote the inverse of F_k by F_k^{-1}.

We want that a BC behaves as a pseudo random function (PRF):

Definition 1. *A* BC $\mathsf{F} : \mathcal{K} \times \mathcal{X} \to \{0,1\}^n$ *is a* (q,t,ϵ)-PRF *(*Pseudo Random Function*) if for any* (q,t)*-adversary* A

$$|\Pr[1 \leftarrow \mathsf{A}^{\mathsf{F}_k(\cdot)}] - \Pr[1 \leftarrow \mathsf{A}^{\mathsf{f}(\cdot)}] \leq \epsilon$$

where $k \xleftarrow{\$} \mathcal{K}$*, and* $\mathsf{f} \xleftarrow{\$} \mathcal{FUNC}$*, where* $\mathcal{FUNC}$ *is the set of functions with the same signature as* F_k*, that is, the functions* $\mathsf{f} : \mathcal{X} \to \{0,1\}^n$*.*

2.2 Cryptographic Primitives used - (IV-based) Encryption Schemes

For encryption, we use iv-based encryption schemes:

Definition 2. *An* iv-based encryption scheme (ivE) *is a triple of algorithms* $\Pi = (\mathsf{Gen}, \mathsf{Enc}, \mathsf{Dec})$ *where*

- Gen*, a* key-generation *algorithm, samples a key* k *from the key space* $\mathcal{K}$*.*
- Enc *is an* encryption *algorithm that, on input a key* $k \in \mathcal{K}$*, an initialization vector* $\mathsf{iv} \in \mathcal{IV}$*, and a message* $m \in \mathcal{M}$*, outputs a ciphertext* $c \in \mathcal{C}$*, denoted* $c \leftarrow \mathsf{Enc}_k(\mathsf{iv}, m)$*.*
- Dec *is a deterministic* decryption *algorithm that, on input a key* $k \in \mathcal{K}$*, an* $\mathsf{iv} \in \mathcal{IV}$*, and a ciphertext* $c \in \mathcal{C}$*, outputs a message* $m \in \mathcal{M}$ *or* $\perp$ *("invalid"), denoted* $\perp / m = \mathsf{Dec}_k(\mathsf{iv}, c)$*.*

For security, we assume that the initialization vector is uniformly sampled.

Definition 3. *An* ivE*-scheme* $\Pi = (\mathsf{Gen}, \mathsf{Enc}, \mathsf{Dec})$ *is* (q,t,ϵ)-ivE-secure *if* $\forall$ (q,t)*-adversary* A

$$|\Pr[1 \leftarrow \mathsf{A}^{\mathsf{Enc}_k^{\$}}] - \Pr[1 \leftarrow \mathsf{A}^{\$}]| \leq \epsilon,$$

where $\mathsf{Enc}_k^{\$}(m)$ *is an oracle that on input* m*, picks* $\mathsf{iv} \xleftarrow{\$} \mathcal{IV}$ *and outputs* $(\mathsf{iv}, \mathsf{Enc}_k(\mathsf{iv}, m))$*, and the oracle* \$*, on input* m*, samples* $\mathsf{iv} \xleftarrow{\$} \mathcal{IV}$ *and outputs* (iv, c) *with* $c \xleftarrow{\$} \{0,1\}^{|\mathsf{Enc}_k(\mathsf{iv},m)|}$*.*

2.3 Leakage

When the adversary has physical access to the electronic devices where cryptographic algorithms are implented in, she may measure physical quantities involved during the computations, such as instantaneous power consumption, execution time, or electromagnetic/radio-frequency (RF) radiation, etc. [30,32,44,16]. These observations (also called *leakage*) can give significative information about the inputs (even about the key).

Notation. We use the suffix L to denote that an oracle leaks (i.e., OL, for the oracle O). The leakage is obtained via the *leakage function* $\mathsf{L}_{\mathsf{O}}(x;k)$.

We say that a security definition is *black-box* when there is no leakage.

2.4 *Privacy* in the Presence of Leakage.

Modifying the standard real-vs-ideal game (see Def.3) by adding leakage to both oracles ($\mathsf{Enc}^{\$}$ and \$) is problematic as we need to create an ideal leakage [41]. Therefore, following Pereira et al. [41], we start from the standard CPA-security (Chosen-Plaintext Attack) [29], adding leakage to the game. In this way, the adversary has to distinguish between two real encryptions whose leakage can be collected:

Definition 4. *An* ivE*-scheme* $\Pi = (\mathsf{Gen}, \mathsf{Enc}, \mathsf{Dec})$ *is* (q_L, q_E, t, ϵ)-CPAL-secure (CPA with leakage)*, if for any* (q_L, q_E, t)*-adversary* $\mathsf{A} = (\mathsf{A}_1, \mathsf{A}_2)$

$$\Big| \Pr[b = b' \mid b' \leftarrow \mathsf{A}_2^{\mathsf{L}(\cdot,\cdot;\cdot),\mathsf{Enc}^{\$}\mathsf{L}_k(\cdot)}((\mathsf{iv}^*, c^*), \ell^*, st),\ (c^*, \ell^*) = \mathsf{EncL}_k(\mathsf{iv}^*, m_b),$$
$$\mathsf{iv}^* \xleftarrow{\$} \mathcal{IV}, (st, m_0, m_1) \leftarrow \mathsf{A}_1^{\mathsf{L}(\cdot,\cdot;\cdot),\mathsf{Enc}^{\$}\mathsf{L}_k(\cdot)}\ s.t.\ |m_0| = |m_1|, b \xleftarrow{\$} \{0,1\}] - \frac{1}{2} \Big| \leq \epsilon$$

where $\mathsf{L}(\cdot,\cdot;\cdot)$ *is an oracle used to model the leakage;* A_1 *is a* $(q_{1,L}, q_{1,E}, t_1)$*-adversary,* A_2 *is a* $(q_{2,L}, q_{2,E}, t_2)$*-adversary, with* $q_{1,L} + q_{2,L} \leq q_L$, $q_{1,E} + q_{2,E} \leq q_E$ *and* $t_1 + t_2 \leq t$.

We use the oracle $\mathsf{L}(\cdot,\cdot;\cdot)$ to model the profiling (or pre-processing) phase that an adversary can do [41]. In this phase, the adversary has access to a copy of the target device. She has full control of the device (notably she can choose the keys). In this way, she can try to understand what the leakage reveals about the message and the key. Therefore, she can choose all the inputs (message, iv *and the key*) in these queries, clearly with the aim of accurately *model* the leakage.

Note that in this definition, as in all others where the adversary performs multiple steps, i.e. $\mathsf{A} = (\mathsf{A}_1, \mathsf{A}_2)$, we assume that in each step the adversary can send some information to the following step. We represent this with the string *st*.

We do not follow the IND−CLPA approach by Barwell et al. [3] because they assume that the challenge encryption (that is the one producing c^* in the previous definition) does not leak [26].

2.5 PSV

Pereira et al. [41] started from the leakage resilient stream cipher from Standaert et al. [47]. This scheme is based on a block-cipher E and follows the rekeying principle. In this scheme, every key k_i is called *ephemeral* because it is used only twice as a key for E: once to generate a new key (i.e., refresh it), $k_{i+1} = \mathsf{E}_{k_i}(p_A)$, and once to generate a (pseudo) random stream block, $y_i = \mathsf{E}_{k_i}(p_B)$ (p_A and p_B are two public constants, $p_A \neq p_B$).

The PSV encryption scheme uses a master key k as the key of the strongly protected block-cipher F. PSV is an ivE, therefore there is always a random input, the iv. It uses the strongly protected block-cipher F_k and the iv to generate the first ephemeral key k_1 of the previously described stream cipher [47]. To encrypt the message m, it XORes it to the stream produced.

Algorithm 1 The leakage-resistant encryption scheme PSV [41].

It uses a leak-free BC $\mathsf{F} : \mathcal{K} \times \{0,1\}^n \rightarrow \{0,1\}^n$, and a weakly protected BC $\mathsf{E} : \{0,1\}^n \times \{0,1\}^n \rightarrow \{0,1\}^n$.

Gen:

- $k \xleftarrow{\$} \mathcal{K}$
- $p_A, p_B \xleftarrow{\$} \{0,1\}^n$

$\mathsf{Enc}_k(m)$:

- Parse m in $m_1, ..., m_l$
- $\mathsf{iv} \xleftarrow{\$} \{0,1\}^n$
- $k_1 = \mathsf{F}_k(\mathsf{iv})$
- For $i = 1, ..., l$
  - $y_i = \mathsf{E}_{k_i}(p_B)$
  - $c_i = \pi_{|m_i|}(y_i) \oplus m_i$
  - $k_{i+1} = \mathsf{E}_{k_i}(p_A)$
- $c = (c_1, \ldots, c_l)$
- Return (iv, c)

$\mathsf{Dec}_k(\mathsf{iv}, c)$:

- Parse c in $c_1, ..., c_l$
- $k_1 = \mathsf{F}_k(\mathsf{iv})$
- For $i = 1, ..., l$
  - $y_i = \mathsf{E}_{k_i}(p_B)$
  - $m_i = \pi_{|c_i|}(y_i) \oplus c_i$
  - $k_{i+1} = \mathsf{E}_{k_i}(p_A)$
- Return $m = (m_1, \ldots, m_l)$

We describe formally PSV in Fig. 1 and in Alg. 1.

PSV has been proved CPAL secure in the presence of leakage assuming that F is a leak-free PRF, E provides simulatable leakage (and an assumption for the leakage of the XOR computation) [41]. This proof does not mean that the advantage of an adversary against PSV is negligible, but only that we can reduce the CPAL security of the whole PSV scheme to the previous aforementioned properties.

3 New Reductions for Blockciphers in the Presence of Leakage

We introduce our new assumption for leakage of the block-cipher. We start considering PSV (see Fig. 1).

We observe that it is based on a refreshing scheme. Our key observation is the following:

Key Reduction

We assume that it is indistinguishable whether the scheme is refreshing the internal keys (k_i) or it uses a random key.
We show that the security of the whole scheme with leakage can be reduced to a verifiable security argument - if it is indistinguishable with leakage whether the scheme is refreshing the internal keys (k_i) or using random keys.

Definition 5. *The implementation of the blockcipher* E *provides a* (q, t, ϵ)-secure refreshing *(*SecRef*) if for any* (q, t)*-adversary* A*,*

$$|\Pr[\mathsf{SecRef}_{\mathsf{E},\mathsf{L},\mathsf{A}^{\mathsf{L}(\cdot,\cdot)}} = 1] - 1/2| \leq \epsilon.$$

The SecRef *experiment is defined in Tab. 1.*

The $\mathsf{SecRef}_{\mathsf{E},\mathsf{L},\mathsf{A}^{\mathsf{L}(\cdot,\cdot)}}$ experiment	
Initialization: $k_1, \bar{k}_2 \xleftarrow{\$} \{0,1\}^n$ $p_A, p_B \leftarrow \{0,1\}^n, p_A \neq p_B$ $(k_2, \ell^1) = \mathsf{EL}_{k_1}(p_A)$ $(y_1, \ell^2) = \mathsf{EL}_{k_1}(p_B)$ $b \xleftarrow{\$} \{0,1\}$ Finalization: $b' \leftarrow \mathsf{A}^{\mathsf{L}(\cdot,\cdot),\mathsf{EL}(\cdot)}(\ell^1, y_1, \ell^2, p_A, p_B)$ If $b = b'$ Return 1 Return 0	Oracle $\mathsf{EL}(x)$: If $b = 0$ $(y, \ell) = \mathsf{EL}_{k_2}(x)$ Else $(y, \ell) = \mathsf{EL}_{\bar{k}_2}(x)$ Return (y, ℓ)

Table 1: The SecRef experiment. The choices of the indexes for the ephemeral keys is done to make them coherent with the PSV Fig. 1 (and Fig. 3).

Assumption for the initial stage of the rekeying process in PSV*:* Owing the structure of PSV, we clearly need a special consideration for the start of the chain which is topologically different, i.e., in previous works the goal was to prove and define security when the main refreshed key is output by a strongly protected blockcipher, and concretely in order to prove security of PSV a *leak-free* assumption was necessary.

Key Advantage

In this work our goal is to weaken this hard assumption (which may induce high implementation cost). We propose to use the following assumption, stemming from the following rational: if it is hard to distinguish with a single leakage-trace associated with the single output generated by F (namely k_1), which is the output of a Nonce-based encryption (therefore only a single leakage trace assumption) from the output generated at random, then we may retain confidentiality with leakage, without the need of instantiating a leak-free component. In practice, such *leak-free*ness is typically achieved with a very high order masking implementation and is highly expensive. Our requirement from F is much weaker or, as we anticipate, would be easier to achieve by implementers.

In this case, we will instantiate F with these *looser* requirements, which we provide a falsifiable in-lab test procedure. And because we proved that it is indistinguishable from random, even with leakage, we can model that as it is immediately refreshed, moving forward and propagating throughout the scheme in our proof. As we show in the next section, this security requirement cannot be applied straightforwardly to PSV, because the first ephemeral key in the i^{th} encryption query k_1^{i}[10] is used once, while the first ephemeral key k_1 is used twice (once to generate y_1 and once for k_2). But, it is enough to see k^i as a first ephemeral key k_0, then refresh it, so that $\bar{k} = k_1$, and from k_1 use the encryption part of PSV. We detail the modified scheme as PSV2 in Alg. 2 and in Fig. 4.

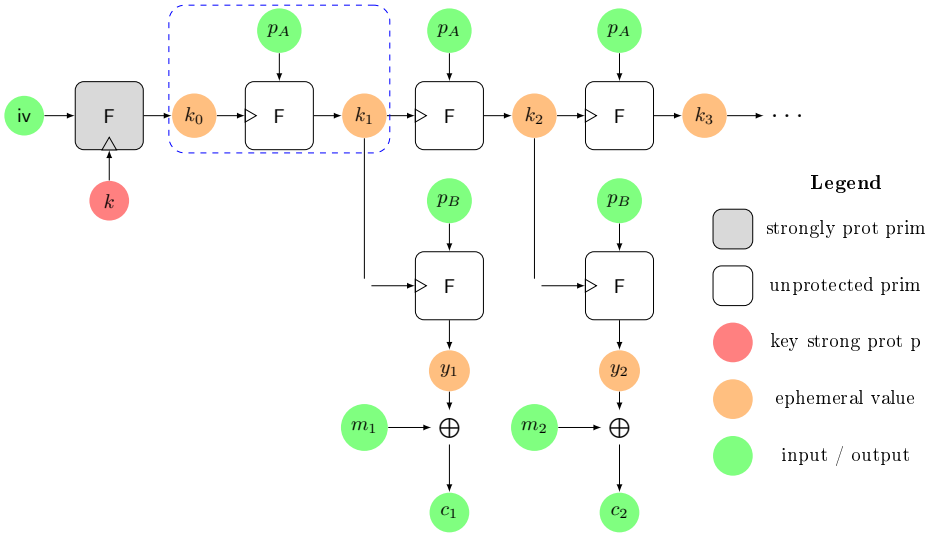


Fig. 4: The PSV2 encryption scheme (Alg. 2). We highlight the differences with PSV (Fig. 1).

Definition 6. *The implementations of the strongly protected (Sp) blockcipher* F *and the less protected blockcipher* E *provides a* (q, t, ϵ)- strongly protected secure refreshing *(*SpSecRef*) if for any* (q, t)*-adversary* A*,*

$$|\Pr[\mathsf{SpSecRef}_{\mathsf{F},\mathsf{E},\mathsf{L},\mathsf{A}^{\mathsf{L}(\cdot,\cdot)}} = 1] - 1/2| \leq \epsilon.$$

The SecRef *experiment is defined in Tab. 2.*

Note that the previous definition does not require that the leakages ℓ_1, ℓ_2, and those obtained from the leaking queries to E via L *provide no information.*

[10] We use the superscript to denote to which encryption query the ephemeral key refers to.

Algorithm 2 The leakage-resistant encryption scheme PSV2 (a modification of PSV). We highlight in blue the changes with respect to PSV.

It uses a strongly protected BC $\mathsf{F} : \mathcal{K} \times \{0,1\}^n \to \{0,1\}^n$, and a weakly protected BC $\mathsf{E} : \{0,1\}^n \times \{0,1\}^n \to \{0,1\}^n$.

Gen:

- $k \xleftarrow{\$} \mathcal{K}$
- $p_A, p_B \xleftarrow{\$} \{0,1\}^n$

$\mathsf{Enc}_k(m)$:

- Parse m in $m_1, ..., m_l$
- $\mathsf{iv} \xleftarrow{\$} \{0,1\}^n$
- $k_0 = \mathsf{F}_k(\mathsf{iv})$
- $k_1 = \mathsf{E}_{k_0}(p_A)$
- For $i = 1, ..., l$
  - $y_i = \mathsf{E}_{k_i}(p_B)$
  - $c_i = \pi_{|m_i|}(y_i) \oplus m_i$
  - $k_{i+1} = \mathsf{E}_{k_i}(p_A)$
- $c = (c_1, \ldots, c_l)$
- Return (iv, c)

$\mathsf{Dec}_k(\mathsf{iv}, c)$:

- Parse c in $c_1, ..., c_l$
- $k_0 = \mathsf{F}_k(\mathsf{iv})$
- $k_1 = \mathsf{E}_{k_0}(p_A)$
- For $i = 1, ..., l$
  - $y_i = \mathsf{E}_{k_i}(p_B)$
  - $m_i = \pi_{|c_i|}(y_i) \oplus c_i$
  - $k_{i+1} = \mathsf{E}_{k_i}(p_A)$
- Return $m = (m_1, \ldots, m_l)$

The $\mathsf{SpSecRef}_{\mathsf{F},\mathsf{E},\mathsf{L},\mathsf{A}^{\mathsf{L}(\cdot,\cdot)}}$ experiment	
Initialization: $k \xleftarrow{\$} \{0,1\}^n$ $\mathcal{K}, \mathcal{N} \leftarrow \emptyset$ $b \xleftarrow{\$} \{0,1\}$ $i = 1$ Finalization: $b' \leftarrow \mathsf{A}^{\mathsf{L}(\cdot,\cdot),\mathsf{EL}(\cdot,\cdot),\mathsf{FL}_k(x)}$ If $b = b'$ Return 1 Return 0	Oracle $\mathsf{EL}(i, x)$: If k_1^i not defined Return $\perp$ $(y, \ell) = \mathsf{EL}_{k_1^i}(x)$ Return (y, ℓ) Oracle $\mathsf{FL}_k(x)$: If $x \in \mathcal{N}$ Return $\perp$ $(\ell, k_1) = \mathsf{F}_k(x)$ If $b = 0$ $k_1^i = k_1$ If $b = 1$ $k_1^i \xleftarrow{\$} \{0,1\}^n$ $\mathcal{K} \leftarrow \mathcal{K} \cup \{k_1^i\}, \quad \mathcal{N} \leftarrow \mathcal{N} \cup \{x\}$ $i = i + 1$ Return ℓ

Table 2: The SpSecRef experiment.

Rather, the definition requires that the information provided by these leakages *does not allow the adversary to correlate the generation of k_2 with its subsequent use.*[11]

With respect to simulators for simulatability [46,36,17], we note that such simulators are potentially vulnerable to attacks because they effectively combine physically "different" measurements. In addition to potential inconsistencies in the logical propagation of the simulated leakage, these measurements may also exhibit physical and environmental inconsistencies. In our setting, we propose a mechanism that prevents this artifact and allows the "real" and "ideal" cases to be evaluated in the same environment and within the same experiment. Furthermore, if the experiment is conducted in a *fixed* environment, where the internal selection between k_2 and $\bar{k}_2$ is not possible for simulation or verification purposes (as discussed in detail in Sec. 3.1), we believe that detecting cross-trace inconsistencies is substantially more difficult in practice, both from a physical and an experimental perspective. In our setting, the adversary has to identify inconsistencies across **three** different traces (one generating a key and two subsequently using that key), rather than inconsistencies within a single trace, as is the case when the leakage is simulated with the simulatability framework.

Compositional Scheme-level Arguments - Assumption for the XOR*:* Similarly to other works which aim to touch mode-level compositional arguments with leakage, we need to base security arguments on some assumptions relating to other building blocks in the scheme, such as vector XOR operations, and sometimes their composition with other primitives [41,8]. For example in PSV, we XOR a message block m_i with a random block of the stream, y_i. Therefore an adversary, which receives $c_i = y_i \oplus m_i$, can combine the leakage from the generation of $y_i (= \mathsf{E}_{k_i}(p_B))$ and from the XOR operation $y_i \oplus m_i$. Assumptions like that are popular in the literature, see for example [41,8,35].

Essentially, we require that an adversary cannot distinguish whether a single block (or stage) of PSV is encrypting the i^{th} message block from m_0 or from m_1. We need to give the adversary the leakage of the generation of y_i (and therefore also of the computation of $k_{i+1} = \mathsf{E}_{k_i}(p_A)$), since from k_i, the adversary can compute y_i (as p_B is a public constant). This is formalized by adapting the

[11] For example, suppose that ℓ_1 leaks the first 10 bits of k_2 (we omit ℓ_2 for simplicity, as we expect it to provide no information about k_2, but only about the key k_1 used to generate k_2), while EL leaks the last 10 bits of the key used by EL. In this case, the adversary cannot distinguish whether EL uses k_2 or $\bar{k}_2$. Similarly, suppose that ℓ_1 reveals the Hamming weight of k_2, while EL leaks the first 10 bits of the key used by EL. Then, the adversary cannot distinguish whether EL uses k_2 or $\bar{k}_2$ except with negligible probability. Clearly, if $HW(k_2)$ is extreme, i.e., close to 0 or n, the Hamming weight may provide sufficient information to distinguish the keys. However, in such cases, the Hamming weight itself already significantly reduces the entropy of k_2, but this happens with negligible probability.

LORL2 assumption [8][12] to our model for weakly protected blockciphers (the resulting definition is similar to [41]).

Definition 7. *The implementation of the blockipher* E *and of the consecutive* $\oplus$ *provides* (t, ϵ)-indistinguishability with leakage (XORInd) *if forall* t*-adversary* $\mathsf{A} = (\mathsf{A}_1, \mathsf{A}_2)$

$$
\begin{aligned}
|\Pr[b' = b \mid b' \leftarrow \mathsf{A}_2^{\mathsf{L}(\cdot,\cdot)}(k_1, c, \ell^1, \ell^2, \ell^3, \mathsf{st}),\ (c, \ell^3) \leftarrow y \oplus_{\mathsf{L}}\ m_b,\ (y, \ell^2) \leftarrow \mathsf{EL}_k(p_B),\\
(k_1, \ell^1) \leftarrow \mathsf{EL}_k(p_A),\ (m_0, m_1, \mathsf{st}) \leftarrow \mathsf{A}_1^{\mathsf{L}(\cdot,\cdot)}(p_A, p_B, n),\quad s.t.\ |m_0| = |m_1| = n,\\
b \xleftarrow{\$} \{0,1\},\ k \xleftarrow{\$} \{0,1\}^n,\ p_A, p_B \leftarrow \{0,1\}^n,\quad s.t.\ p_A \neq p_B] - 1/2| \leq \epsilon.
\end{aligned}
$$

Essentially, the previous definition is the eavesdropper with leakage security of a single block of PSV.

As pointed out in [8], it is hard to expect that an adversary has negligible advantage in distinguishing the two messages, but we have reduced the security of the whole scheme to the security of one of the simplest manipulation of the message: XORing it with a pseudorandom block.

For simplicity, we have assumed that dropping bits (what we do when we XOR the last message using the $\pi_{|m_l|}$ does not leak any information). We assume this since we can assume that these additional bits are leaked and this does not give any information about the other bits.

3.1 On the Falsifiability of our Assumptions

As one of the main objectives of this work, it can and should be possible to falsify the proposed assumption by an evaluation laboratory. We next list an explicit/example test-sketch that can be used for that purpose.

For SecRef*:* The test proceeds in two steps:

- 1) Without leakage: A random key k_1 is picked, and $k_2 = \mathsf{E}_{k_1}(p_A)$ is computed, as well as a random key $\bar{k}_2$ is picked. Then, k_2 and $\bar{k}_2$ are stored in two memory cells M_0 and M_1. Then, a bit b is sampled. To compute the oracle $\mathsf{EL}(\cdot)$, we require that the value stored in M_b is used as a key.
- 2) With leakage: The device is tested without using the information about the memory cells (or the device is given to another tester who does not know what has been done without leakage, in phase-1). To avoid the scenario that it would be possible to distinguish the two cases by looking at the usage of memory cells, we assume that the value k_2 computed during this phase-2 (and therefore about which the adversary has leakage) is stored in a memory cell (say) M_3 different from M_0 and M_1.
- **Challenge:** the evaluation lab is essentially challenged with the distinguishing game and to understand whether $\mathsf{EL}(\cdot)$ is using as a key k_2 (which suits the generation leakage it has collected) or $\bar{k}_2$.

[12] LORL2 has been used to prove the privacy of TEDT (an Authenticated Encryption Scheme that uses PSV inside) in the presence of leakage. We need to adapt it because that definition was in the ideal cipher model, following the work of [52].

The idea is that the test should be setup in a way reflecting the adversarial scenario which should be left with distinguishing the previous situations looking only at the outputs of E and its leakage, and not by exploiting the memory use which is an artifact. And in such a scenario she should not be able to attack PSV.

Note that the proposed "modifications" are not required in the actual implementation of any scheme. They are introduced solely for the purpose of experimentally evaluating our assumption. More precisely, the additional registers are needed only during the evaluation procedure. Without them, if EL directly uses k_2, an evaluator could trivially observe that the register storing k_2 during its generation is later reused by EL. Conversely, in the modified experiment (i.e., $b = 1$), the evaluator would observe either that the register containing k_2 is overwritten before the call to EL, or that EL uses a different register altogether. Such observations would allow the evaluator to distinguish the two experiments solely by monitoring register usage, without relying on leakage from the cryptographic computations themselves. Since, in the "real" implementation, the first scenario naturally occurs, we eliminate this artificial distinguisher during the evaluation by precomputing k_2 and letting EL operate on a separate register containing either k_2 or $\bar{k}_2$, rather than on the register produced during initialization. Another possible option for such an evaluation lab is to exclude the register READ leakage section, though it is less *clean*.

For $\mathsf{SpSecRef}$*:*

- 1) Without leakage: A random key k is picked, a vector of q random ivs', $[\mathsf{iv}^1, \ldots, \mathsf{iv}^q]$ and $k_1^i = \mathsf{F}_k(\mathsf{iv}^i)$ is computed for all i. Then, two arrays of memory cells M_0 and M_1 containing the memory cells $M_{i,j}$ ($(i,j) \in \{0,1\} \times \{1, \ldots, q\}$) are taken, and a random key $\bar{k}_2$ is picked. k_1^i is stored in M_0^i and $\bar{k}_1^i \in M_1^i$. Then, a bit b is sampled. To compute the oracle $\mathsf{EL}(i, \cdot)$, we require that the value stored in $M_{i,b}$ is used as a key.
- 2) With leakage: The device is tested without using the information about the memory cells. To avoid the scenario that it would be possible to distinguish the two cases by looking at the usage of memory cells, we assume that the ephemeral keys k_0^i computed during this phase are stored in a memory arrays of cells M_3 different from M_0 and M_1.

4 Proving Privacy Based on the New Assumptions

In this section, we prove the security of PSV using our assumptions.

Theorem 1. *Let* F *be a leak-free blockcipher that is a* $(q+1, t_1, \epsilon_{\mathsf{PRF}})$*-secure* PRF*. Let the implementation of the weakly protected blockipher* E *and of the consecutive* $\oplus$ *be* $(2, t_2, \epsilon_{\mathsf{SecRef}})$-$\mathsf{SecRef}$ *and* $(t_3, \epsilon_{\mathsf{XORInd}})$-$\mathsf{XORInd}$*, then,* PSV *encrypting at most* L *blocks messages, is* (q, t, ϵ)-CPAL *secure, with*

$$\epsilon_{\mathsf{PRF}} + L\epsilon_{\mathsf{SecRef}} + L/2\epsilon_{\mathsf{XORInd}},$$

$t_1 = (q+1)L(2t_{\mathsf{E}} + t_{\oplus})$, $t_2 = (q+1)L(2t_{\mathsf{E}} + t_{\oplus}) - (4t_E + 2t_{\oplus})$, *and* $t_3 = (q+1)L(2t_{\mathsf{E}} + t_{\oplus}) - (2t_E + t_{\oplus})$, *where* t_{E} *is the time needed to evaluate once* E *and to collect its leakage,* $t_{\oplus}$ *is the time needed to XOR 2 strings of at most* n *bits and to collect the corresponding leakage.*

Idea of the proof. We observe that the adversary cannot obtain any useful information from leakage except during the challenge encryption query (that is, if we give the adversary all others ephemeral keys the adversary cannot do anything better). The idea of the proof is to replace F_k with a random function (by hypothesis F is a leak-free secure PRF), then, we can replace one after the other the ephemeral keys in the challenge encryption query with random ones. This allows us to replace the encryption of each block of m_0 with the encryption of the corresponding block of m_1 because we are exactly in the case of the XORInd security definition.

Proof. We prove this using a sequence of games. We denote with E_i the event that at the end of Game i, the adversary A outputs 1.

Game 0. It is the CPAL game where the bit picked is 0.

Game 1. It is Game 0 where we replace the outputs of the leak-free PRF output with random values.

Transition between Game 0 and Game 1. To bound $|\Pr[E_0] - \Pr[E_1]|$, we build a PRF-adversary $(q+1, t_1)$-B against F.

The adversary B against the PRF F. Since F is a leak-free PRF, we can see B as black-box adversary. Its description is straightforward: At the start of the game B has oracle access to either the oracle $\mathsf{F}(\cdot)$ or $\$(\cdot)$ and it has to distinguish them. She picks two distinct values p_A and p_B in $\{0,1\}^n$ which she sends to A.

When A performs one of her encryption queries on input m, B proceeds as follows:

- 1) she picks a random value $\mathsf{iv} \xleftarrow{\$} \{0,1\}^n$;
- 2) she queries her oracle on input iv obtaining k_1;
- 3) she parses m in n-bit blocks, $m = (m_1, \ldots, m_l)$ (with $l \leq L$) and for each $j = 1, \ldots, l$:
  - $3i$) she computes $y_j = \mathsf{E}_{k_j}(p_B)$, collecting the leakage $\ell_{j,1} = \mathsf{L}_E(p_B; k_j)$;
  - $3ii$) she computes $c_j = y_j \oplus m_j$, collecting the leakage $\ell_{j,2} = \mathsf{L}_{\oplus}(y_j, m_j)$ [if $j = l$ $c_j = \pi_{|m_j|}(y_j) \oplus m_j$ and $\ell_{j,2} = \mathsf{L}_{\oplus}(\pi_{|m_j|}(y_j), m_j)$], and
  - $3iii$) she computes $k_{j+1} = \mathsf{E}_{k_j}(p_A)$, collecting the leakage $\ell_{j,3} = \mathsf{L}_E(p_A; k_j)$. She sets $\ell_j := (\ell_{j,1}, \ell_{j,2}, \ell_{j,3})$;
- 4) Finally, she returns $c = (c_1, \ldots, c_l)$ and the leakage $\ell = (\ell_1, \ldots, \ell_l)$. This needs one oracle query and needs time $2lt_{\mathsf{E}} + lt_{\oplus} \leq L(2t_{\mathsf{E}} + t_{\oplus})$.

When A_1 outputs (st, m_0, m_1), B proceeds as in an encryption query where the adversary has asked to encrypt m_0. Furthermore, in addition to the ciphertext c^* and the leakage ℓ^* computed in this way, she also sends st to A_2. For this, B needs one oracle query and time at most $L(2t_{\mathsf{E}} + t_{\oplus})$.

When A_2 outputs her output bit, B simply returns the same output bit as A_2. Therefore B does at most $q+1$ oracle queries and needs time at most $(q+1)L(2t_{\mathsf{E}}+t_{\oplus})=t_1$.

Transition between Game 0 and Game 1 We observe that if the oracle B interacts with is $\mathsf{F}_k(\cdot)$, B perfectly simulates Game 0 for A, otherwise Game 1. Since F is $(q+1,t_1,\epsilon_{\mathsf{PRF}})$-$\mathsf{PRF}$ and B is a $(q+1,t_1)$-adversary, then

$$|\Pr[E_0]-\Pr[E_1]|\leq\epsilon_{\mathsf{PRF}}.$$

Game 2 It is Game 1 where we replace in the challenge encryption query (the one encrypting either m_0 or m_1) all the ephemeral keys with random keys.

Transition between Game 1 and Game 2 We use a sequence of L games: Game $1^0,\ldots,1^L$. Let Game 1^0 be Game 1.

Game 1^i It is Game 1^{i-1} where, instead of using k_i to compute y_i and k_{i+1}, a random key $\bar{k}_i$ is used.[13] Let E_{1^i} be the event that A outputs 1 at the end of this game.

Transition between Game 1^i and Game 1^{i+1}. To bound $|\Pr[E_{1^i}]-\Pr[E_{1^{i+1}}]|$ we consider two cases: a) if $i=0$, then $\Pr[E_{1^0}]=\Pr[E_{1^1}]$ because there is no leakage (k_1 is obtained via a leak-free PRF) and k_1 is thus considered random; b) otherwise, we build a SecRef-adversary C^i.

The SecRef-adversary C^i: C^i has to distinguish whether she has oracle access to $\mathsf{EL}_{k_i}(\cdot)$ or to $\mathsf{EL}_{\bar{k}_i}(\cdot)$. At the start of the game C^i receives $(\ell^1,c,\ell^2,p_A,p_B)$. She forwards (p_A,p_B) to A.

For every encryption query, C^i follows the behaviour of B except for step 2): C^i picks a random value k_1. Therefore, for every encryption query C^i needs time $2lt_{\mathsf{E}}+lt_{\oplus}\leq L(2t_{\mathsf{E}}+t_{\oplus})$ (we assume that picking a random value requires no time).

When A_1 outputs (st,m^0,m^1), C^i proceeds as follows:

- 1) she picks a random value $\mathsf{iv}\xleftarrow{\$}\{0,1\}^n$;
- 2) she parses m^0 in n-bit blocks, $m^0=(m_1,\ldots,m_l)$ (with $l\leq L$).
- 3) for $j=1,\ldots,i-2$
  - $3i$) she picks a random value $\bar{k}_j$ she computes $y_j=\mathsf{E}_{\bar{k}_j}(p_B)$, collecting the leakage $\ell_{j,1}=\mathsf{L}_E(p_B;\bar{k}_j)$;
  - $3ii$) she computes $c_j=y_j\oplus m_j$, collecting the leakage $\ell_{j,2}=\mathsf{L}_{\oplus}(y_j,m_j)$, and
  - $3iii$) she computes $\bar{k}_{j+1}=\mathsf{E}_{\bar{k}_j}(p_A)$, collecting the leakage $\ell_{j,3}=\mathsf{L}_E(p_A;\bar{k}_j)$. She sets $\ell_j:=(\ell_{j,1},\ell_{j,2},\ell_{j,3})$;

  Then 4)
  - $4i$) she sets $y_{i-1}:=c$, $\ell_{i-1,1}:=\ell_1$;
  - $4ii$) she computes $c_{i-1}=y_{i-1}\oplus m_{i-1}$, collecting the leakage $\ell_{i-1,2}=\mathsf{L}_{\oplus}(y_{i-1},m_{i-1})$, and
  - $4iii$) she sets $\ell_{i-1,3}:=\ell^2$ and $\ell_{i-1}:=(\ell_{i-1,1},\ell_{i-1,2},\ell_{i-1,3})$.

[13] $\bar{k}_i$ corresponds to $\bar{k}$ in Tab. 1, while k_i corresponds to $\tilde{k}$ and k_{i-1} to k.

After that 5),

- $5i$) she queries her oracle on input p_B obtaining $(y_i, \ell_{i,1})$;
- $5ii$) she computes $c_i = y_i \oplus m_i$, collecting the leakage $\ell_{i,2} = \mathsf{L}_{\oplus}(y_i, m_i)$ [if $i = l$ $c_i = \pi_{|m_i|}(y_i) \oplus m_i$ and $\ell_{i,2} = \mathsf{L}_{\oplus}(\pi_{|m_i|}(y_i), m_i)$], and
- $5iii$) she queries her oracle on input p_A obtaining $(k_{i+1}, \ell_{i,3})$, and she sets $\ell_i := (\ell_{i,1}, \ell_{i,2}, \ell_{i,3})$;

– 6) Then, for all $j = i+1, \ldots, l$:

- $6i$) she computes $y_j = \mathsf{E}_{k_j}(p_B)$, collecting the leakage $\ell_{j,1} = \mathsf{L}_E(p_B; k_j)$;
- $6ii$) she computes $c_j = y_j \oplus m_j$, collecting the leakage $\ell_{j,2} = \mathsf{L}_{\oplus}(y_j, m_j)$ [if $j = l$ $c_j = \pi_{|m_j|}(y_j) \oplus m_j$ and $\ell_{j,2} = \mathsf{L}_{\oplus}(\pi_{|m_j|}(y_j), m_j)$], and
- $6iii$) she computes $k_{j+1} = \mathsf{E}_{k_j}(p_A)$, collecting the leakage $\ell_{j,3} = \mathsf{L}_E(p_A; k_j)$. She sets $\ell_j := (\ell_{j,1}, \ell_{j,2}, \ell_{j,3})$;

– 7) Finally, she returns $c = (c_1, \ldots, c_l)$ and the leakage $\ell = (\ell_1, \ldots, \ell_l)$ to A_2 and forwards st. This needs two oracle queries and needs time $(l-2)(2t_{\mathsf{E}} + t_{\oplus}) \leq (L-2)(2t_{\mathsf{E}} + t_{\oplus})$.

When A_2 outputs her output bit, C^i simply returns the same output bit as A_2. Therefore C^i queries twice her oracle and needs time at most $(q+1)L(2t_{\mathsf{E}} + t_{\oplus}) - (4t_E + 2t_{\oplus}) = t_2$.

Bounding $|\Pr[E_{1^i}] - \Pr[E_{1^{i+1}}]|$**.** Let us assume that the inputs (ℓ^1, c, ℓ^2) that C^i receives are computed as follows: $k_{i-1}, \bar{k}_i \xleftarrow{\$} \{0,1\}^n$ (they correspond to k and $\bar{k}$, respectively in Tab. 1), $(\tilde{k}_i, \ell^1) = \mathsf{EL}_{k_{i-1}}(p_A)$, $(y_i, \ell^2) = \mathsf{EL}_{k_{i-1}}(p_B)$. If the oracle C^i interacts uses $\tilde{k}_i$ (that would correspond to a honest execution of a block of PSV) as its key, then C^i simulates Game 1^i for A, otherwise, it uses the randomly picked key $\bar{k}_i$, and C^i simulates Game 1^{i+1}. Therefore as any challenger can do the previous choices,

$$|\Pr[E_{1^i}] - \Pr[E_{1^{i+1}}]| \leq \epsilon_{\mathsf{SecRef}}.$$

Bounding $|\Pr[E_1] - \Pr[E_2]|$**.** Since Game 1 is Game 1^0 and Game 2 is Game 1^L, then

$$|\Pr[E_1] - \Pr[E_2]| = |\Pr[E_{1^0}] - \Pr[E_{1^L}]| \leq \sum_{i=1}^{L} |\Pr[E_{1^i}] - \Pr[E_{1^{i+1}}]| = L\epsilon_{\mathsf{SecRef}}.$$

Game 3. It is Game 2, where we encrypt message m_1 instead of m_0.

Transition between Game 2 and Game 3 We use a sequence of L games: Game $2^0, \ldots, 2^L$. Let Game 2^0 be Game 2.

Game 2^i It is Game 2^{i-1} where we encrypt $m_{1,i}$ instead of $m_{0,i}$. Let E_{2^i} be the event that A outputs 1 at the end of this game.

Transition between Game 2^i **and Game** 2^{i+1}**.** To bound $|\Pr[E_{2^i}] - \Pr[E_{2^{i+1}}]|$ we build a XORInd-adversary D^i.

The t_3**-**XORInd**-adversary** D^i**:** At the start of the game, two distinct constants p_A and p_B are chosen. D receives as input (p_A, p_B, n), where n is the security parameter.

For every encryption query, D^i follows the behaviour of C^i. Therefore, for every encryption query D^i needs time $2lt_{\mathsf{E}} + lt_{\oplus} \leq L(2t_{\mathsf{E}} + t_{\oplus})$ (we assume that picking a random value requires no time).

For the challenge encryption query, when A_1 outputs (st, m^0, m^1), D_1^i proceeds as follows:

- 1) she picks a random value $\mathsf{iv} \xleftarrow{\$} \{0,1\}^n$;
- 2) she parses m^0 and m^1 in n-bit blocks, $m^0 = (m_1^0, \ldots, m_l^0)$, and $m^1 = (m_1^1, \ldots, m_l^1)$ (with $l \leq L$).
- 3) for $j = 1, \ldots, i-1$
  - $3i$) she picks a random value $\bar{k}_j$ she computes $y_j = \mathsf{E}_{\bar{k}_j}(p_B)$, collecting the leakage $\ell_{j,1} = \mathsf{L}_E(p_B; \bar{k}_j)$;
  - $3ii$) she computes $c_j = y_j \oplus m_j^1$, collecting the leakage $\ell_{j,2} = \mathsf{L}_{\oplus}(y_j, m_j^1)$, and
  - $3iii$) she computes $\bar{k}_{j+1} = \mathsf{E}_{\bar{k}_j}(p_A)$, collecting the leakage $\ell_{j,3} = \mathsf{L}_E(p_A; \bar{k}_j)$. She sets $\ell_j := (\ell_{j,1}, \ell_{j,2}, \ell_{j,3})$;
- 4) she outputs $(m_i^0, m_1^i, \mathsf{st}')$, where $\mathsf{st}' = (\mathsf{st}, m^0, m^1, c_1, \ldots, c_{i-1})$, obtaining $(k_{i+1}, c_i, \ell_{i,1}, \ell_{i,2}, \ell_{i,3})$;
- 5) for $j = i+1, \ldots, l$
  - $5i$) she picks a random value $\bar{k}_j$ she computes $y_j = \mathsf{E}_{\bar{k}_j}(p_B)$, collecting the leakage $\ell_{j,1} = \mathsf{L}_E(p_B; \bar{k}_j)$;
  - $5ii$) she computes $c_j = y_j \oplus m_j^0$, collecting the leakage $\ell_{j,2} = \mathsf{L}_{\oplus}(y_j, m_j^0)$ [if $j = l$ $c_j = \pi_{|m_j^0|}(y_j) \oplus m_j^0$ and $\ell_{j,2} = \mathsf{L}_{\oplus}(\pi_{|m_j^0|}(y_j), m_j^0)$], and
  - $5iii$) she computes $\bar{k}_{j+1} = \mathsf{E}_{\bar{k}_j}(p_A)$, collecting the leakage $\ell_{j,3} = \mathsf{L}_E(p_A; \bar{k}_j)$. She sets $\ell_j := (\ell_{j,1}, \ell_{j,2}, \ell_{j,3})$;
- 6) Finally, she returns $c = (c_1, \ldots, c_l)$ and the leakage $\ell = (\ell_1, \ldots, \ell_l)$ to A_2 and forwards st.

This needs two oracle queries and needs time $(l-2)(2t_{\mathsf{E}}+t_{\oplus}) \leq (L-2)(2t_{\mathsf{E}}+t_{\oplus})$.

When A_2 outputs her output bit, D_2^i simply returns the same output bit as A_2. Therefore D^i needs time at most $(q+1)L(2t_{\mathsf{E}} + t_{\oplus}) - (2t_E + t_{\oplus}) = t_3$.

Bounding $|\Pr[E_{2^i}] - \Pr[E_{2^{i+1}}]|$. If the oracle D^i interacts encrypts m_i^0, then D^i simulates Game 2^i for A, otherwise Game 2^{i+1}. Therefore, since D^i is a t_3-adversary and the implementation of the blockipher E and of the consecutive $\oplus$ provides (t, ϵ)-XORInd,

$$|\Pr[E_{2^i}] - \Pr[E_{2^{i+1}}]| \leq \epsilon_{\mathsf{XORInd}}.$$

Bounding $|\Pr[E_2] - \Pr[E_3]|$. Since Game 2 is Game 2^0 and Game 3 is Game 2^L, then

$$|\Pr[E_2] - \Pr[E_3]| = |\Pr[E_{2^0}] - \Pr[E_{2^L}]| \leq \sum_{i=1}^{L} |\Pr[E_{2^i}] - \Pr[E_{2^{i+1}}]| = L\epsilon_{\mathsf{XORInd}}.$$

Game 4. It is Game 3, where, in the challenge encryption query, instead of using the random key $\bar{k}_i$ to compute y_i and k_{i+1}, the key k_i is used. As this is

the transition between Game 1 and Game 2 done in the opposite order, then, we can bound

$$|\Pr[E_3] - \Pr[E_4]| \leq L\epsilon_{\mathsf{SecRef}}.$$

Game 5. It is Game 4 where we replace the random k_1 (which in Game 1 replaced the output of the leak-free PRF) with the real output of the leak-free PRF, As this is the transition between Game 0 and Game 1 done in the opposite order, then, we can bound

$$|\Pr[E_4] - \Pr[E_5]| \leq \epsilon_{\mathsf{PRF}}.$$

Concluding the proof. Putting everything together, since the probability that an adversary wins the CPAL game is $1/2|\Pr[1 \leftarrow \mathsf{A}(c^{*,0})] - \Pr[1 \leftarrow \mathsf{A}(c^{*,1})]|$, where $c^{*,i}$ denotes that in the challenge encryption query we have encrypted m^i, then

$$\epsilon \leq 1/2|\Pr[E_0] - \Pr[E_5]| \leq 1/2\sum_{i=0}^{4}|\Pr[E_i] - \Pr[E_{i+1}]|$$
$$\leq 1/2(2\epsilon_{\mathsf{PRF}} + 2L\epsilon_{\mathsf{SecRef}} + L\epsilon_{\mathsf{XORInd}}) = \epsilon_{\mathsf{PRF}} + L\epsilon_{\mathsf{SecRef}} + L/2\epsilon_{\mathsf{XORInd}}.$$

Now, we prove the security of the modified PSV2 construction assuming that the implementation of the strongly protected F and of the the block-cipher E provides stronlgy protected secure refreshing (Def. 6):

Theorem 2. *Let the implementation of the strongly protected* F *blockcipher and of the less protected blockcipher* E *provides* $(q+1, t_1, \epsilon_{\mathsf{SpSecRef}})$-SpSecRef. *Let the implementation of the weakly protected blockipher* E *and of the consecutive* $\oplus$ *be* $(2, t_2, \epsilon_{\mathsf{SecRef}})$-SecRef *and* $(t_3, \epsilon_{\mathsf{XORInd}})$-XORInd, *then,* PSV *encrypting at most* L *blocks messages, is* (q, t, ϵ)-CPAL *secure, with*

$$\epsilon_{\mathsf{SpSecRef}} + L\epsilon_{\mathsf{SecRef}} + L/2\epsilon_{\mathsf{XORInd}},$$

$t_1 = (q+1)L(2t_{\mathsf{E}} + t_{\oplus})$, $t_2 = (q+1)L(2t_{\mathsf{E}} + t_{\oplus}) - (4t_E + 2t_{\oplus})$, *and* $t_3 = (q+1)L(2t_{\mathsf{E}} + t_{\oplus}) - (2t_E + t_{\oplus})$, *where* t_{E} *is the time needed to evaluate once* E *and to collect its leakage,* $t_{\oplus}$ *is the time needed to XOR 2 strings of at most* n *bits and to collect the corresponding leakage.*

Idea of the proof. With respect to Thm. 1, the only difference is that F is no more assumed to be leak-free. But the modification we did to obtain PSV2 are such that we can immediately replace all first ephemeral keys k_0^i with random ones using the SpSecRef security definition. From there, the proof is the same as the proof of Thm, 1

Proof. We prove this using a sequence of games. We denote with E_i the event that at the end of Game i, the adversary A outputs 1. **Game 0.** It is the CPAL game where the bit picked is 0.

Game 1. It is Game 0 where we replace the outputs of the F with random values.

Transition between Game 0 and Game 1. To bound $|\Pr[E_0] - \Pr[E_1]|$, we build a $\mathsf{SpSecRef}$-adversary $(q+1, t_1)$-B against F and E.

The $(q+1, t_1)$-$\mathsf{SpSecRef}$-adversary B. At the start of the game B picks two distinct values p_A and p_B in $\{0,1\}^n$, that she forwards to A. B has oracle access to an oracle that is implemented with $\mathsf{F}_k(\cdot)$. This oracle generates the keys $k_0^1, \ldots, k_0^q, k_0^*$ either via $\mathsf{F}_k(\cdot)$ or by random sampling. After any of these keys is generated B has oracle access to an oracle that implements $\mathsf{E}_{k_0^j}$.

When A does one of her encryption queries, we say the j^{th} on input m, B proceeds as follows:

- 1) she picks a random value $\mathsf{iv}^j \xleftarrow{\$} \{0,1\}^n$;
- 2) she queries her oracle on input iv^j obtaining $\ell_{0,1}$;
- 3) she queries her oracle $\mathsf{E}_{k_0^j}$ on input p_A obtaining k_1 and the leakage $\ell_{0,2}$; she sets $\ell_0 := (\ell_{0,1}, \ell_{0,2})$;
- 4) she parses m in n-bit blocks, $m = (m_1, \ldots, m_l)$ (with $l \leq L$) and for each $j = 1, \ldots, l$:
  - $3i$) she computes $y_j = \mathsf{E}_{k_j}(p_B)$, collecting the leakage $\ell_{j,1} = \mathsf{L}_E(p_B; k_j)$;
  - $3ii$) she computes $c_j = y_j \oplus m_j$, collecting the leakage $\ell_{j,2} = \mathsf{L}_{\oplus}(y_j, m_j)$ [if $j = l$ $c_j = \pi_{|m_j|}(y_j) \oplus m_j$ and $\ell_{j,2} = \mathsf{L}_{\oplus}(\pi_{|m_j|}(y_j), m_j)$], and
  - $3iii$) she computes $k_{j+1} = \mathsf{E}_{k_j}(p_A)$, collecting the leakage $\ell_{j,3} = \mathsf{L}_E(p_A; k_j)$. She sets $\ell_j := (\ell_{j,1}, \ell_{j,2}, \ell_{j,3})$;
- 4) Finally, she returns $c = (c_1, \ldots, c_l)$ and the leakage $\ell = (\ell_0, \ell_1, \ldots, \ell_l)$.

This needs one oracle query to F and one to $\mathsf{E}_{k_0^j}$ and needs time $2lt_{\mathsf{E}} + lt_{\oplus} \leq L(2t_{\mathsf{E}} + t_{\oplus})$.

When A_1 outputs (st, m_0, m_1), B proceeds as in an encryption query where the adversary has asked to encrypt m_0. Furthermore, in addition to the ciphertext c^* and the leakage ℓ^* computed in this way, she also sends st to A_2. For this, B needs one oracle query and time at most $L(2t_{\mathsf{E}} + t_{\oplus})$.

When A_2 outputs her output bit, B simply returns the same output bit as A_2. Therefore B does at most $q+1$ oracle queries and needs time at most $(q+1)L(2t_{\mathsf{E}} + t_{\oplus}) = t_1$.

Transition between Game 0 and Game 1 We observe that if the oracle B interacts sets the key used by the random oracle as $\mathsf{F}_k(y^j)$, B perfectly simulates Game 0 for A, otherwise Game 1. Since F is $(q+1, t_1, \epsilon_{\mathsf{SpSecRef}})$-$\mathsf{PRF}$ and B is a $(q+1, t_1)$-adversary, then

$$|\Pr[E_0] - \Pr[E_1]| \leq \mathsf{SpSecRef}.$$

Game 2. It is Game 1, where, in the challenge encryption query, we encrypt message m_1 instead of m_0. Since the transition between Game 1 and Game is the exact same transition between Game 1 and Game 4 in the previous proof (Thm. 1), as in both cases the ephemeral key k_1 is random, then,

$$|\Pr[E_1] - \Pr[E_2]| \leq 2L\epsilon_{\mathsf{SecRef}} + L\epsilon_{\mathsf{XORInd}}.$$

Game 3. It is Game 2 where we replace the random k_0 (which in Game 1 replaced the output of the stronlgy protected F) with the real output of F, As

this is the transition between Game 0 and Game 1 done in the opposite order, then, we can bound

$$|\Pr[E_2] - \Pr[E_3]| \leq \epsilon_{\mathsf{SpSecRef}}.$$

Concluding the proof. Putting everything together, since the probability that an adversary wins the CPAL game is $1/2|\Pr[1 \leftarrow \mathsf{A}(c^{*,0})] - \Pr[1 \leftarrow \mathsf{A}(c^{*,1})]|$, where $c^{*,i}$ denotes that in the challenge encryption query we have encrypted m^i, then

$$\epsilon \leq 1/2|\Pr[E_0] - \Pr[E_3]| \leq 1/2\sum_{i=0}^{3}|\Pr[E_i] - \Pr[E_{i+1}]|$$

$$\leq 1/2(2\epsilon_{\mathsf{SpSecRef}} + 2L\epsilon_{\mathsf{SecRef}} + L\epsilon_{\mathsf{XORInd}}) = \epsilon_{\mathsf{SpSecRef}} + L\epsilon_{\mathsf{SecRef}} + L/2\epsilon_{\mathsf{XORInd}}.$$

5 Conclusion and Future Work

In this paper, our goal is to provide a security property to prove the security of refreshing schemes in the standard model. This security property can be tested and falsified. Furthermore, we have given, to the best of our knowledge, the first security assumption that can replace the leak-free assumption for the strongly protected block cipher, that can be falsified and based on which we can prove confidentiality, and hopefully to reduce the prohibitive cost from leak-free components.

Future work. Several research directions remain open:

- Verifying in practice that our security definitions can be achieved by existing implementations of block ciphers and to design new ones able to offer to meet our model with lower cost.
- We believe that the proposed definitions can be applied to other schemes and can be the base on which the security of other schemes can be proved.
- We should try to find similar definitions when the adversary can also inject faults.

Acknowledgments

Itamar Levi and Francesco Berti were partly supported by the Israel Science Foundation (ISF), grants 2569/21 and 3276/25. Part of this work was carried out while Francesco Berti was at Bar Ilan University.

References

1. M. Abdalla, S. Belaïd, and P. Fouque. Leakage-resilient symmetric encryption via re-keying. In G. Bertoni and J. Coron, editors, *Cryptographic Hardware and Embedded Systems - CHES 2013 - 15th International Workshop, Santa Barbara, CA, USA, August 20-23, 2013. Proceedings*, volume 8086 of *Lecture Notes in Computer Science*, pages 471–488. Springer, 2013.

2. A. Akavia, S. Goldwasser, and V. Vaikuntanathan. Simultaneous hardcore bits and cryptography against memory attacks. In O. 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. G. Barwell, D. P. Martin, E. Oswald, and M. Stam. Authenticated encryption in the face of protocol and side channel leakage. In T. Takagi and T. Peyrin, editors, *Advances in Cryptology - ASIACRYPT 2017 - 23rd International Conference on the Theory and Applications of Cryptology and Information Security, Hong Kong, China, December 3-7, 2017, Proceedings, Part I*, Lecture Notes in Computer Science, pages 693–723. Springer, 2017.
4. J. Béguinot, A. Mukherjee, M. Obresmki, J. Ribeiro, L. Roy, F. Standaert, and D. Venturi. Simulating noisy leakage with bounded leakage: Simpler, better, faster. *IACR Cryptol. ePrint Arch.*, 2026:357, 2026.
5. D. Bellizia, F. Berti, O. Bronchain, G. Cassiers, S. Duval, C. Guo, G. Leander, G. Leurent, I. Levi, C. Momin, O. Pereira, T. Peters, F. Standaert, B. Udvarhelyi, and F. Wiemer. Spook: Sponge-based leakage-resistant authenticated encryption with a masked tweakable block cipher. *IACR Trans. Symmetric Cryptol.*, 2020(S1):295–349, 2020.
6. D. Bellizia, O. Bronchain, G. Cassiers, V. Grosso, C. Guo, C. Momin, O. Pereira, T. Peters, and F. Standaert. Mode-level vs. implementation-level physical security in symmetric cryptography - A practical guide through the leakage-resistance jungle. In D. Micciancio and T. Ristenpart, editors, *Advances in Cryptology - CRYPTO 2020 - 40th Annual International Cryptology Conference, CRYPTO 2020, Santa Barbara, CA, USA, August 17-21, 2020, Proceedings, Part I*, volume 12170 of *Lecture Notes in Computer Science*, pages 369–400. Springer, 2020.
7. F. Berti, C. Guo, O. Pereira, T. Peters, and F. Standaert. Strong authenticity with leakage under weak and falsifiable physical assumptions. In Z. Liu and M. Yung, editors, *Information Security and Cryptology - 15th International Conference, Inscrypt 2019, Nanjing, China, December 6-8, 2019, Revised Selected Papers*, volume 12020 of *Lecture Notes in Computer Science*, pages 517–532. Springer, 2019.
8. F. Berti, C. Guo, O. Pereira, T. Peters, and F. Standaert. Tedt, a leakage-resist AEAD mode for high physical security applications. *IACR Trans. Cryptogr. Hardw. Embed. Syst.*, 2020(1):256–320, 2020.
9. F. Berti, C. Guo, T. Peters, and F. Standaert. Efficient leakage-resilient macs without idealized assumptions. In M. Tibouchi and H. Wang, editors, *Advances in Cryptology - ASIACRYPT 2021 - 27th International Conference on the Theory and Application of Cryptology and Information Security, Singapore, December 6-10, 2021, Proceedings, Part II*, volume 13091 of *Lecture Notes in Computer Science*, pages 95–123. Springer, 2021.
10. F. Berti, C. Hazay, and I. Levi. sflr-sfot: Leakage-resilient oblivious transfer. In C. Galdi and D. H. Phan, editors, *Security and Cryptography for Networks - 14th International Conference, SCN 2024, Amalfi, Italy, September 11-13, 2024, Proceedings, Part I*, volume 14973 of *Lecture Notes in Computer Science*, pages 182–204. Springer, 2024.
11. F. Berti, F. Koeune, O. Pereira, T. Peters, and F. Standaert. Ciphertext integrity with misuse and leakage: Definition and efficient constructions with symmetric primitives. In J. Kim, G. Ahn, S. Kim, Y. Kim, J. López, and T. Kim, editors, *Proceedings of the 2018 on Asia Conference on Computer and Communications*

Security, AsiaCCS 2018, Incheon, Republic of Korea, June 04-08, 2018, pages 37–50. ACM, 2018.

12. F. Berti, O. Pereira, and F. Standaert. Reducing the cost of authenticity with leakages: a \mathsf CIML2 -secure \mathsf AE scheme with one call to a strongly protected tweakable block cipher. In J. Buchmann, A. Nitaj, and T. Rachidi, editors, *Progress in Cryptology - AFRICACRYPT 2019 - 11th International Conference on Cryptology in Africa, Rabat, Morocco, July 9-11, 2019, Proceedings*, volume 11627 of *Lecture Notes in Computer Science*, pages 229–249. Springer, 2019.
13. G. Cassiers, B. Grégoire, I. Levi, and F.-X. Standaert. Hardware private circuits: From trivial composition to full verification. *IEEE Transactions on Computers*, 70(10):1677–1690, 2020.
14. S. Chari, C. S. Jutla, J. R. Rao, and P. Rohatgi. Towards sound approaches to counteract power-analysis attacks. In M. J. Wiener, editor, *Advances in Cryptology - CRYPTO '99, 19th Annual International Cryptology Conference, Santa Barbara, California, USA, August 15-19, 1999, Proceedings*, volume 1666 of *Lecture Notes in Computer Science*, pages 398–412. Springer, 1999.
15. N. Costes and M. Stam. Pincering SKINNY by exploiting slow diffusion enhancing differential power analysis with cluster graph inference. *IACR Trans. Cryptogr. Hardw. Embed. Syst.*, 2023(4):460–492, 2023.
16. E. Danieli, M. Goldzweig, M. Avital, and I. Levi. Revealing the secrets of radio embedded systems: Extraction of raw information via RF. *IEEE Trans. Inf. Forensics Secur.*, 19:2066–2081, 2024.
17. E. Deprez, C. Momin, and F.-X. Standaert. Simulatable leakage, revisited. In *Constructive Approaches for Security Analysis and Design of Embedded Systems - First International Conference, CASCADE 2026, Regensburg, Germany, March 30- April 1, 2026, Proceedings*, Lecture Notes in Computer Science, 2026.
18. C. Dobraunig, M. Eichlseder, S. Mangard, F. Mendel, B. Mennink, R. Primas, and T. Unterluggauer. Isap v2.0. *IACR Trans. Symmetric Cryptol.*, 2020(S1):390–416, 2020.
19. C. Dobraunig, M. Eichlseder, S. Mangard, F. Mendel, and T. Unterluggauer. ISAP - towards side-channel secure authenticated encryption. *IACR Trans. Symmetric Cryptol.*, 2017(1):80–105, 2017.
20. Y. Dodis, Y. T. Kalai, and S. Lovett. On cryptography with auxiliary input. In M. Mitzenmacher, editor, *Proceedings of the 41st Annual ACM Symposium on Theory of Computing, STOC 2009, Bethesda, MD, USA, May 31 - June 2, 2009*, pages 621–630. ACM, 2009.
21. Y. Dodis and K. Pietrzak. Leakage-resilient pseudorandom functions and side-channel attacks on feistel networks. In T. Rabin, editor, *Advances in Cryptology - CRYPTO 2010, 30th Annual Cryptology Conference, Santa Barbara, CA, USA, August 15-19, 2010. Proceedings*, volume 6223 of *Lecture Notes in Computer Science*, pages 21–40. Springer, 2010.
22. S. Dziembowski and K. Pietrzak. Leakage-resilient cryptography. In *49th Annual IEEE Symposium on Foundations of Computer Science, FOCS 2008, Philadelphia, PA, USA, October 25-28, 2008*, pages 293–302. IEEE Computer Society, 2008.
23. S. Faust, K. Pietrzak, and J. Schipper. Practical leakage-resilient symmetric cryptography. In E. Prouff and P. Schaumont, editors, *Cryptographic Hardware and Embedded Systems - CHES 2012 - 14th International Workshop, Leuven, Belgium, September 9-12, 2012. Proceedings*, volume 7428 of *Lecture Notes in Computer Science*, pages 213–232. Springer, 2012.

24. D. Goudarzi and M. Rivain. How fast can higher-order masking be in software? In J. Coron and J. B. Nielsen, editors, *Advances in Cryptology - EUROCRYPT 2017 - 36th Annual International Conference on the Theory and Applications of Cryptographic Techniques, Paris, France, April 30 - May 4, 2017, Proceedings, Part I*, volume 10210 of *Lecture Notes in Computer Science*, pages 567–597, 2017.
25. H. Groß, E. Wenger, C. Dobraunig, and C. Ehrenhöfer. Ascon hardware implementations and side-channel evaluation. *Microprocess. Microsystems*, 52:470–479, 2017.
26. C. Guo, O. Pereira, T. Peters, and F. Standaert. Authenticated encryption with nonce misuse and physical leakage: Definitions, separation results and first construction - (extended abstract). In P. Schwabe and N. Thériault, editors, *Progress in Cryptology - LATINCRYPT 2019 - 6th International Conference on Cryptology and Information Security in Latin America, Santiago de Chile, Chile, October 2-4, 2019, Proceedings*, volume 11774 of *Lecture Notes in Computer Science*, pages 150–172. Springer, 2019.
27. A. Juma and Y. Vahlis. Protecting cryptographic keys against continual leakage. In T. Rabin, editor, *Advances in Cryptology - CRYPTO 2010, 30th Annual Cryptology Conference, Santa Barbara, CA, USA, August 15-19, 2010. Proceedings*, volume 6223 of *Lecture Notes in Computer Science*, pages 41–58. Springer, 2010.
28. Y. T. Kalai and L. Reyzin. A survey of leakage-resilient cryptography. In O. Goldreich, editor, *Providing Sound Foundations for Cryptography: On the Work of Shafi Goldwasser and Silvio Micali*, pages 727–794. ACM, 2019.
29. J. Katz and Y. Lindell. *Introduction to Modern Cryptography, Second Edition*. CRC Press, 2014.
30. P. C. Kocher. Timing attacks on implementations of diffie-hellman, rsa, dss, and other systems. In N. Koblitz, editor, *Advances in Cryptology - CRYPTO '96, 16th Annual International Cryptology Conference, Santa Barbara, California, USA, August 18-22, 1996, Proceedings*, volume 1109 of *Lecture Notes in Computer Science*, pages 104–113. Springer, 1996.
31. P. C. Kocher. Leak-resistant cryptographic indexed key update, Mar. 25 2003. US Patent 6,539,092.
32. P. C. Kocher, J. Jaffe, and B. Jun. Differential power analysis. In M. J. Wiener, editor, *Advances in Cryptology - CRYPTO '99, 19th Annual International Cryptology Conference, Santa Barbara, California, USA, August 15-19, 1999, Proceedings*, volume 1666 of *Lecture Notes in Computer Science*, pages 388–397. Springer, 1999.
33. I. Levi, D. Bellizia, and F.-X. Standaert. Beyond algorithmic noise or how to shuffle parallel implementations? *International Journal of Circuit Theory and Applications*, 48(5):674–695, 2020.
34. I. Levi and C. Hazay. Garbled circuits from an SCA perspective free XOR can be quite expensive. . *IACR Trans. Cryptogr. Hardw. Embed. Syst.*, 2023(2):54–79, 2023.
35. R. Li, Y. Sun, C. Guo, F. Standaert, W. Wang, and X. Wang. Leakage-resilient circuit garbling. In B. Luo, X. Liao, J. Xu, E. Kirda, and D. Lie, editors, *Proceedings of the 2024 on ACM SIGSAC Conference on Computer and Communications Security, CCS 2024, Salt Lake City, UT, USA, October 14-18, 2024*, pages 780–794. ACM, 2024.
36. J. Longo, D. P. Martin, E. Oswald, D. Page, M. Stam, and M. Tunstall. Simulatable leakage: Analysis, pitfalls, and new constructions. In P. Sarkar and T. Iwata, editors, *Advances in Cryptology - ASIACRYPT 2014 - 20th International Conference on the Theory and Application of Cryptology and Information Security,*

Kaoshiung, Taiwan, R.O.C., December 7-11, 2014. Proceedings, Part I, volume 8873 of *Lecture Notes in Computer Science*, pages 223–242. Springer, 2014.
37. S. Mangard. Hardware countermeasures against DPA ? A statistical analysis of their effectiveness. In T. Okamoto, editor, *Topics in Cryptology - CT-RSA 2004, The Cryptographers' Track at the RSA Conference 2004, San Francisco, CA, USA, February 23-27, 2004, Proceedings*, volume 2964 of *Lecture Notes in Computer Science*, pages 222–235. Springer, 2004.
38. M. Medwed, F. Standaert, J. Großschädl, and F. Regazzoni. Fresh re-keying: Security against side-channel and fault attacks for low-cost devices. In D. J. Bernstein and T. Lange, editors, *Progress in Cryptology - AFRICACRYPT 2010, Third International Conference on Cryptology in Africa, Stellenbosch, South Africa, May 3-6, 2010. Proceedings*, volume 6055 of *Lecture Notes in Computer Science*, pages 279–296. Springer, 2010.
39. S. Micali and L. Reyzin. Physically observable cryptography (extended abstract). In M. Naor, editor, *Theory of Cryptography, First Theory of Cryptography Conference, TCC 2004, Cambridge, MA, USA, February 19-21, 2004, Proceedings*, volume 2951 of *Lecture Notes in Computer Science*, pages 278–296. Springer, 2004.
40. M. Obremski, J. Ribeiro, L. Roy, F. Standaert, and D. Venturi. Improved reductions from noisy to bounded and probing leakages via hockey-stick divergences. In L. Reyzin and D. Stebila, editors, *Advances in Cryptology - CRYPTO 2024 - 44th Annual International Cryptology Conference, Santa Barbara, CA, USA, August 18-22, 2024, Proceedings, Part VI*, Lecture Notes in Computer Science, pages 461–491. Springer, 2024.
41. O. Pereira, F. Standaert, and S. Vivek. Leakage-resilient authentication and encryption from symmetric cryptographic primitives. In I. Ray, N. Li, and C. Kruegel, editors, *Proceedings of the 22nd ACM SIGSAC Conference on Computer and Communications Security, Denver, CO, USA, October 12-16, 2015*, pages 96–108. ACM, 2015.
42. C. Petit, F. Standaert, O. Pereira, T. Malkin, and M. Yung. A block cipher based pseudo random number generator secure against side-channel key recovery. In M. Abe and V. D. Gligor, editors, *Proceedings of the 2008 ACM Symposium on Information, Computer and Communications Security, ASIACCS 2008, Tokyo, Japan, March 18-20, 2008*, pages 56–65. ACM, 2008.
43. K. Pietrzak. A leakage-resilient mode of operation. In A. 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.
44. J. Quisquater and D. Samyde. Electromagnetic analysis (EMA): measures and counter-measures for smart cards. In I. Attali and T. P. Jensen, editors, *Smart Card Programming and Security, International Conference on Research in Smart Cards, E-smart 2001, Cannes, France, September 19-21, 2001, Proceedings*, volume 2140 of *Lecture Notes in Computer Science*, pages 200–210. Springer, 2001.
45. D. Salomon and I. Levi. Masksimd-lib: on the performance gap of a generic c optimized assembly and wide vector extensions for masked software with an ascon-p test case. *Journal of cryptographic engineering*, 13(3):325–342, 2023.
46. M. Staib and A. Moradi. Deep learning side-channel collision attack. *IACR Trans. Cryptogr. Hardw. Embed. Syst.*, 2023(3):422–444, 2023.
47. F. Standaert, O. Pereira, and Y. Yu. Leakage-resilient symmetric cryptography under empirically verifiable assumptions. In R. Canetti and J. A. Garay, editors,

Advances in Cryptology - CRYPTO 2013 - 33rd Annual Cryptology Conference, Santa Barbara, CA, USA, August 18-22, 2013. Proceedings, Part I, volume 8042 of *Lecture Notes in Computer Science*, pages 335–352. Springer, 2013.

48. F. Standaert, O. Pereira, Y. Yu, J. Quisquater, M. Yung, and E. Oswald. Leakage resilient cryptography in practice. In A. Sadeghi and D. Naccache, editors, *Towards Hardware-Intrinsic Security - Foundations and Practice*, Information Security and Cryptography, pages 99–134. Springer, 2010.
49. K. Tiri and I. Verbauwhede. A logic level design methodology for a secure DPA resistant ASIC or FPGA implementation. In *2004 Design, Automation and Test in Europe Conference and Exposition (DATE 2004), 16-20 February 2004, Paris, France*, pages 246–251. IEEE Computer Society, 2004.
50. N. Veyrat-Charvillon, M. Medwed, S. Kerckhof, and F. Standaert. Shuffling against side-channel attacks: A comprehensive study with cautionary note. In X. Wang and K. Sako, editors, *Advances in Cryptology - ASIACRYPT 2012 - 18th International Conference on the Theory and Application of Cryptology and Information Security, Beijing, China, December 2-6, 2012. Proceedings*, volume 7658 of *Lecture Notes in Computer Science*, pages 740–757. Springer, 2012.
51. S. You, M. G. Kuhn, S. Sarkar, and F. Hao. Low trace-count template attacks on 32-bit implementations of ASCON AEAD. *IACR Trans. Cryptogr. Hardw. Embed. Syst.*, 2023(4):344–366, 2023.
52. Y. Yu, F. Standaert, O. Pereira, and M. Yung. Practical leakage-resilient pseudorandom generators. In E. Al-Shaer, A. D. Keromytis, and V. Shmatikov, editors, *Proceedings of the 17th ACM Conference on Computer and Communications Security, CCS 2010, Chicago, Illinois, USA, October 4-8, 2010*, pages 141–151. ACM, 2010.