Leakage-Resilient Storage

Francesco Davì and Stefan Dziembowski

University of Rome
La Sapienza

Abstract. We study a problem of secure date storage on hardware that may leak information. We introduce a new primitive, that we call *leakage-resilient storage* (LRS), which is an (unkeyed) scheme for encoding messages. The security of LRS is defined with respect to a class Γ of leakage functions.
We consider several natural choices of such Γ's: (1) a simple construction for the case when the adversary can just read-off some individual bits; (2) a construction where each leakage function can depends only on some restricted part of the memory; (3) a scheme that is secure if the number of functions that the adversary can choose is restricted (but still it can be exponential in the length of the encoding). This last construction implies security in the case when, the set Γ consists of functions that are computable by Boolean circuits of a small size.
We also discuss the connection between the problem of constructing leakage-resilient storage and a theory of the compressibility of NP-instances.

1 Introduction

Some of the most devastating attacks on cryptographic devices are those that break the actual physical implementation of the scheme, not its mathematical abstraction. These, so-called *side-channel attacks*, are based on the fact that the adversary may obtain some information about the internal data of the device by observing its running-time [14], electromagnetic radiation [22, 8], power consumption [15], or even sound that the device is emitting [25] (see [23, 19] for more examples of such attacks).

1.1 Memory leakages — previous work

In the last couple of years there has been a growing interest in the design of schemes that already on the abstract level guarantee that their physical implementation is secure against a large well-defined class of side-channel attacks (the pioneering paper in this area was [17], and some of the other papers in this area are: [11, 21, 6, 12, 4, 1, 18, 13, 26, 20]). The main idea is to augment the standard security definition by allowing the adversary to learn the value of a chosen by him *leakage function* g on the internal data Y used by the cryptographic scheme. The results in this area can be categorized according to the class of leakage functions g that the model covers. Some papers consider very restricted classes (e.g. in [11] the model assumes that the adversary can simply read-off some wires that represent the computation), while other ones consider more general leakages—e.g. [5, 2] allow the adversary to choose any function g that is *input-shrinking* (i.e. such that $|g(Y)| \ll |Y|$).

Another popular paradigm is to assume that *only computation leaks information*, i.e. the memory cells that do not take part in the computation (in a given time period) do not leak any information. The first paper to state this assumption is [17] (where it is stated as "Axiom 1", page 283), and the other papers that use it are [7, 21]. The other approach [12, 4, 1, 18, 13] is to assume that the memory may simply leak information, independently on the computation

performed. While the second approach may seem more general, it has to be noted that the first approach permits to construct schemes where the total amount of information that leaks can be greater than the memory size (this is possible since the memory contents in the schemes of [7, 21] is evolving during the computation).

It may be questioned if the "only the computation leaks information" paradigm is really relevant to the attack that the adversary can perform in real-life. In many situations memory may actually leak information, even if it is unaccessed. First of all, in modern computer systems it is hard to guarantee that a given part of memory really never gets accessed (for example the memory may be refreshed or moved to cache, etc.). Some practical attacks on unaccessed memory were also demonstrated in [24]. More recently a class of *cold boot attacks* relying on the data remanence property was presented in [9].

In [17] it is argued that

> "*Some form of security for unaccessed memory is mandatory.* For instance, if a small amount of information leakage from a stored secret occurs at every unit of time (e.g., if a given bit becomes 51% predictable within a day) then a patient enough adversary will eventually reconstruct the entire secret."

The goal of this paper is to introduce a formal framework to reason about what "some form of security for unaccessed memory" means, and to propose constructions secure under as weak assumptions as possible.

1.2 Our contribution

In this paper we introduce a new primitive, that we call *leakage-resilient storage*, which can be viewed as a secure storage scheme in the model where the physical memory may leak some side-channel information. A scheme like this consists of two poly-time algorithms *Encode* and *Decode*, where the *encoding* algorithm *Encode* takes as input a message m and produces as output a string $Y := \mathit{Encode}(m)$, and the decoding algorithm *Decode* is such that we always have $\mathit{Decode}(\mathit{Encode}(m)) = m$ (observe that these algorithms do not take as input any secret key).

Informally speaking, in the security definition we allow the adversary to adaptively choose a sequence of leakage functions $g_1, \ldots, g_t$, and learn the values of

$$g_1(Y), \ldots, g_t(Y).$$

We require that the adversary, after learning these values, should gain essentially no additional information on m (this is formalized using a standard indistinguishability game, see Sect. 2 for details). We assume that the g_i's are elements of some fixed set Γ (that will be a parameter in the definition). Obviously, the larger Γ is, the stronger is our definition, and we should aim at defining Γ in such a way that it covers all the attacks that the adversary can launch in real life. All the Γ's that we consider in this paper contain at least the set of functions that read-off the individual bits of Y (i.e. functions h such that $h(Y_1, \ldots, Y_\beta) = (Y_{i_1}, \ldots, Y_{i_j})$). Hence, of course, we need to require that

$$|g_1(Y)| + \cdots + |g_t(Y)| < |Y| \tag{1}$$

(as otherwise the adversary could simply choose the g_i's in such a way that $(g_1(Y), \ldots, g_t(Y)) = Y$). This is essentially the input-shrinking property that, as discussed above, was already used in the literature.

Obviously, if we go to the extreme and simply allow the adversary to choose *any* (poly-time) functions g_i that satisfy (1) then there is no hope for any security, since the adversary could always choose g_1 in such a way that it simply calculates $Decode(Y)$ and outputs some information about m (say: its first bit). Therefore Γ cannot contain the *Decode* function, and hence, we need to restrict Γ in some way.

Note that the assumption that Γ is a restricted class of function is actually very realistic. In practice, the leakage functions need to be computationally "simple": while it is plausible that the adversary can read-off the individual bits, or learn their sum, it seems very improbable that an attack based on measuring power consumption or electromagnetic radiation can directly give information about some more complicated functions of the secret bits.

In this paper we consider several natural choices of such Γ's. We start (Sect. 3.1) with a simple construction for the case when the adversary can just read-off some the individual bits. Then (Sect. 3.2) we describe a construction where each leakage function can depend only on some restricted part of the memory: either because it consists of two separate blocks, or because it is infeasible for the adversary to choose a function that depends from the memory cells that are physically far from each other. Finally (Sect. 3.3), we construct a scheme that is secure if the number of functions that the adversary can choose is restricted (but still it can be exponential in $|Y|$). This last construction implies security in the case when the set Γ consists of functions that are computable by Boolean circuits of a small size. We also discuss (in Sect. 5) the connection between the problem of constructing leakage-resilient storage and a theory of compressibility of NP-instances [10].

2 The definition

Formally, a *leakage-resilient storage (LRS)* scheme is a pair $(Encode, Decode)$, where

- *Encode* is a randomized, efficiently computable function $Encode : \{0,1\}^\alpha \to \{0,1\}^\beta$, and
- *Decode* is a deterministic, efficiently computable function $Decode : \{0,1\}^\beta \to \{0,1\}^\alpha$

Security of such a scheme is defined as follows. Let t be some parameter, and let Γ be a subset of the set of all functions $g : \{0,1\}^\beta \to \{0,1\}^*$. Consider the following game between an adversary $\mathcal{A}$ and an oracle Ω.

1. The adversary chooses a pair of messages $m^0, m^1 \in \{0,1\}^\alpha$ and sends them to Ω.
2. Ω chooses a random bit $b \in \{0,1\}$ and sets $X := Encode(m^b)$.
3. The following is executed t times, for $i = 1, \ldots, t$:
   (a) $\mathcal{A}$ selects a function $g_i : \{0,1\}^\beta \to \{0,1\}^{c_i} \in \Gamma$, and sends it to Ω,
   (b) Ω sends $g_i(X)$ to $\mathcal{A}$. We say that $\mathcal{A}$ *retrieved* c_i *bits from* X.
4. The adversary outputs b'. We say that he *won the game* if $b = b'$.

Such an adversary is called a (Γ, c, t)-adversary if $c_1 + \cdots + c_t \leq c$. We say that $(Encode, Decode)$ is (Γ, c, t, ϵ)-*secure* if no (Γ, c, t)-adversary wins the game with probability greater than $0.5 + \epsilon$. Unless explicitly stated otherwise, we will assume that the adversary in computationally-unbounded. In this case we assume that the adversary is deterministic. This can be done without loss of generality, since the unbounded adversary can always compute the optimal randomness. For an adversary $\mathcal{A}$ let $G_\mathcal{A}(Y)$ denote the vector of values that the adversary $\mathcal{A}$ retrieves from r, i.e. $G_\mathcal{A}(Y) := (g_1(Y), \ldots, g_t(Y))$. Note that $|G_\mathcal{A}(Y)| \leq c$.

2.1 Preliminaries

Let random variables X_0, X_1 be distributed over some set $\mathcal{X}$ and let Y be a random variable distributed over $\mathcal{Y}$. Define the *statistical distance between X_0 and X_1* as

$$\begin{aligned}\delta(X_0;X_1) &= \frac{1}{2}\sum_{x\in\mathcal{X}} |P(X_0=x) - P(X_1=x)| \\ &= \sum_{x:P(X_0=x)>P(X_1=x)} P(X_0=x) - P(X_1=x).\end{aligned}$$

If X is distributed over $\mathcal{X}$ then let $d(X) := \delta(X;U_{\mathcal{X}})$ denote the *statistical distance of X from a variable uniformly distributed over $\mathcal{X}$*. We will overload the symbols δ and d and sometimes apply them to the probability distributions instead of the random variables. If $\mathcal{E}$ is an event then $\delta(X_0;X_1|\mathcal{E}) = \delta(P_{X_0|\mathcal{E}};P_{X_1|\mathcal{E}})$ and $d(X|\mathcal{E}) = d(P_{X|\mathcal{E}})$. If Y is a random variable then $\delta(X_0;X_1|\mathcal{E},Y) = \sum_y \delta(X_0;X_1|\mathcal{E}\wedge(Y=y))\cdot P(\mathcal{E}\wedge(Y=y))$ and $d(X|\mathcal{E},Y) = \sum_y d(P_{X|\mathcal{E}\wedge(Y=y)})\cdot P(\mathcal{E}\wedge(Y=y))$. It is easy to verify that δ satisfies the *triangle inequality*: $\delta(A;B) \le \delta(A;C) + \delta(C;B)$.

A *min-entropy* of a random variable X is denoted as $H_\infty(x)$ and defined as $H_\infty(x) = \min_x(-\log P(X=x))$.

A family $\{H_s\}_{s\in\mathcal{S}}$ of functions $h_s : \mathcal{X} \to \mathcal{Y}$ is called a *collection of ℓ-wise independent hash functions* if for every set $\{x_1,\ldots,x_\ell\} \subseteq \mathcal{X}$ of ℓ elements, and a uniformly random $S \in \mathcal{S}$ we have that $(h_S(x_1),\ldots,h_S(x_\ell))$ is distributed uniformly over $\mathcal{Y}^\ell$. Several constructions of such functions exist in the literature. For example if $GF(2^n)$ is the field with 2^n elements, and for $s = (s_0,\ldots,s_\ell) \in GF(2^n)^{\ell+1}$ and every $m \le n$ we define

$$H_s(x) = \left(\sum_{i=0}^{\ell} s_i x^i\right)_{1\ldots m}$$

(where $x_{1\ldots m}$ denotes the set of m first bits of x) then $\{H_s\}$ is a collection of ℓ-wise independent hash functions.

3 The implementations

In this section we define several families of leakage functions Γ, and show LRS schemes secure against these Γ's.

3.1 Individual-bit leakages

We start with the following simple example. Let $\Gamma_\downarrow$ be the set of all functions g_i of a form

$$g_i(x_1,\ldots,x_\beta) = (x_{i_1},\ldots,x_{i_{c_i}}).$$

An $(\Gamma_\downarrow, \beta-1, \beta-1, 0)$-secure LRS scheme for sharing messages of length $\alpha = 1$ is simply an β-out-of-β secret sharing scheme based on xor. More precisely define $Encode_\oplus(b)$ as follows: $Encode^1_\oplus(b) := (x_1,\ldots,x_\beta)$, where $(x_1,\ldots,x_\beta)$ is a random vector such that $x_1\oplus\cdots\oplus x_\beta = b$. Decoding is defined as: $Decode^1_\oplus(x_1,\ldots,x_\beta) = x_1\oplus\cdots\oplus x_\beta$. The $(\Gamma_\downarrow, \beta-1, \beta-1, 0)$-security of this scheme follows from the fact that the value of $x_1\oplus\cdots\oplus x_\beta$ is uniformly random if at least one of the x_i's is unknown.

To encode messages $m = (m_1, \ldots, m_\alpha)$ of arbitrary length encode each bit m_i individually, i.e. set

$$\mathit{Encode}_\oplus((m_1, \ldots, m_\alpha)) = (Y_1, \ldots, Y_\alpha) = (\mathit{Encode}^1_\oplus(m_1), \ldots, \mathit{Encode}^1_\oplus(m_\alpha)),$$

and

$$\mathit{Decode}_\oplus(Y_1, \ldots, Y_\alpha) = (\mathit{Decode}^1_\oplus(Y_1), \ldots, \mathit{Decode}^1_\oplus(Y_\alpha))$$

(note that now the length of the encoding is $\alpha \cdot \beta$). It is easy to see that this scheme is also $(\Gamma_\downarrow, \beta - 1, \beta - 1, 0)$-secure.

3.2 Memory divided into two parts

Suppose that the encoding is stored on some physical storage device that consists of two separate chips, i.e. the memory x is divided into two parts x^0 and x^1, and each leakage function can be applied to one of the x^i's separately. In other words, the only restriction is that the adversary cannot choose leakage function that depend simultaneously on both x^0 and x^1. More precisely, suppose that β is even and let $x = (x^0, x^1)$ where $x^0 := (x_1, \ldots, x_{\beta/2})$, and $x^1 := (x_{(\beta/2)+1}, \ldots, x_\beta)$. Let Γ_2 be the set of all functions g_i that "depend only on x^0 or x^1", i.e. they have a form

$$g_i(x) = g'_i(x^0),$$

or

$$g_i(x) = g'_i(x^1)$$

(for some g'_i). This model is essentially the same as the one of the two-party Intrusion-Resilient Secret Sharing (IRSS) of Dziembowski and Pietrzak (see [6], Sect. 2.1). The notation that is used in [6] is different: the memory parts x^i are denoted with T_i (and T_0 is called a *share of Alice*, and T_1 is called a *share of Bob*) and the leakage functions are denoted with h_i. Moreover, the retrieval bound (denoted there with s) is applied to each share separately. Instead of bounding the number t of functions that the adversary can choose, [6] bound the number ℓ of times the adversary "hops" from Alice to Bob, which simply means that $\ell = t/2$ (let us assume that t is even). The encoding procedure is called in [6] *sharing of a secret*, and the decoding procedure is called *reconstructing of the secret*.

To construct such a scheme [6] use so called *Bounded-Storage Model (BSM)-secure* functions. A function $f : \{0,1\}^m \times \{0,1\}^n \to \{0,1\}^m$ is (ϵ_{BSM}, c)-*BSM-secure* if for every function $h : \{0,1\}^n \to \{0,1\}^c$ we have

$$d(f(K, R)|K, h(R)) \leq \epsilon_{BSM},$$

where $K \in \{0,1\}^m$ and $R \in \{0,1\}^n$ are uniformly random. An additional property that is usually required is that f should be computable only from a small part of the input R. Without this property (that in fact we do not need in this paper) any randomness extractor is a BSM-secure function (see [16, 27]).

Dziembowski and Pietrzak show how to construct a two-party IRSS scheme using such a BSM secure function. Their Theorem 1 implies the following:

Lemma 1. *Let f be an (ϵ_{BSM}, c)-BSM secure function. Then for every t one can construct an LRS scheme $(\mathit{Encode}_{\mathrm{IRSS}}, \mathit{Decode}_{\mathrm{IRSS}})$ that is $(\Gamma_2, c, t, 2^{\alpha+1} t \epsilon_{BSM})$-secure. The complexity of the encoding and the decoding procedures in this LRS scheme is linear in t.*

Proof. Theorem 1 of [6] shows how to construct a $(2^{\alpha+1}t\epsilon_{BSM}, t, c)$-secure two-party IRSS scheme $(share_{2,t/2}, reconstruct_{2,t/2})$ from any (ϵ_{BSM}, c)-BSM secure function. Let $Encode_{\mathrm{IRSS}}(M) = (T_1, T_2) = share_{2,t/2}(M)$ and let $Decode_{\mathrm{IRSS}}(T_1, T_2) = reconstruct_{2,t/2}(T_1, T_2)$. □

The (ϵ_{BSM}, c)-BSM-secure functions can be constructed for any c being a constant (smaller than 1) fraction of n. Moreover ϵ_{BSM} in these constructions is negligible, and can be made so small that for a constant α the value of $2^{\alpha} \cdot \epsilon_{BSM}$ is still negligible. Hence the $(\Gamma_2, c, t, \epsilon)$-secure LRS scheme can be constructed for c being a constant fraction of β, and ϵ being negligible.

Locally-computable functions Instead of assuming that x is stored on two separate chips it is enough to assume that the leakage functions g are computable *u-locally* (for some parameter u), i.e. each g can be written as

$$g(x_1, \ldots, x_\beta) = g'(x_j, \ldots, x_{j+u}),$$

for some g' and j such that $j + u \leq \beta$ (in other words: g is u-local in it depends on at most u consecutive bits of the input). Let Γ^u_{loc} denote the set of such functions. Observe that the individual-bit leakages with each $c_i = 1$ (Sect. 3.1)[1] can be considered as special case of the locally-computable functions (with $u = 0$). The practical motivation for this assumption is that it may be infeasible for the adversary to choose leakage functions that depend on the cells that are physically located far from each other on one memory chip. A scheme that is $(\Gamma^u_{loc}, c, t, \epsilon)$-secure can be trivially constructed from a $(\Gamma_2, c, t, \epsilon)$-secure scheme $(Encode_2, Decode_2)$: to encode a message m calculate $(x^0, x^1) := Encode_2(m)$ and output

$$x^0 || \overbrace{0, \ldots, 0}^{u} || x^1,$$

in other words: just separate x^0 from x^1 with u memory cells filled with zeros.

3.3 Functions that have small descriptions

Probably the most interesting case is when the only restriction on Γ is that it is a small set of functions: $|\Gamma| = 2^v$, where v is some parameter (that can be for example quadratic in β). One way to look at this family is to fix some method to describe the leakage functions as binary strings, and observe that the set of functions whose description has length v has exactly size 2^v.

For technical reasons, we introduce an additional restriction that the family Γ satisfies the following condition: if $g \in \Gamma$ then g' defined as

$$g'(x_1, \ldots, x_\beta) = g(x_1, \ldots, x_{i-1}, C, x_{i+1}, \ldots, x_\beta)$$

(for some constant $C \in \{0, 1\}$) is also a member of Γ. Such a family Γ will be called *robust*.

A natural example of such a robust Γ is a set of *functions computable by Boolean circuits of a fixed size* (see e.g. [28] for an introduction to the complexity of Boolean circuits). Recall that a *size* of a Boolean circuit is the number r of its gates. Each gate G can be connected

[1] Obviously in Sect. 3.1 without loss of generality we could consider only the functions g with output of length $c_i = 1$, since every $g \in \Gamma_{\downarrow}$ with output of length $c_i > 1$ can be computed by c_i functions with output of length 1.

with two other gates (G_1, G_2) (and we can assume that G is an AND gate if $G_1 \neq G_2$, and it is a NOT gate otherwise). Hence, for each gate we can have at most $(r-1)(r-1) < r^2$ choices. Therefore there are at most $(r^2)^r = r^{2r}$ circuits of size r. Thus the circuits of size r can be described using $u = 2r \log_2 r$ bits.

Several natural functions can be computed by Boolean circuits of a small size (see Sect. 3 of [28]). For example every symmetric function[2] can be computed by a circuit of a linear size (in its input).

Let Γ_v be any robust set of functions such that $|\Gamma_v| = 2^v$. We will now construct a $(\Gamma_v, c, t, \epsilon)$-secure LRS. Let $H = \{h_s : \{0,1\}^n \to \{0,1\}^\alpha\}_{s \in \mathcal{S}}$ be a collection of ℓ-wise independent hash functions. The scheme is parameterized by a value $s \in \mathcal{S}$. For any $s \in \mathcal{S}$ let $\Phi^s = (\mathit{Encode}^s, \mathit{Decode}^s)$, where

$$\mathit{Encode}^s(m) = (r, h_s(r) \oplus m),$$

where $r \in \{0,1\}^n$ is random. Let

$$\mathit{Decode}^s((r,d)) = h_s(r) \oplus d.$$

The following lemma states that with a good probability (over the choice of $s \in \mathcal{S}$) the scheme $(\mathit{Encode}^s, \mathit{Decode}^s)$ is secure.

Lemma 2. *Fix an arbitrary robust set Γ_v such that $|\Gamma_v| = 2^v$. For a randomly chosen s with probability at least $1 - \xi$ we have that Φ^s is $(\Gamma_v, c, t, 2^\alpha \cdot \epsilon + 2^{\alpha+k+c-n})$-secure, for any $c, k, t, v, \ell, \epsilon$ and ξ such that*

$$\xi = 2^{tv - \frac{\ell}{2}(k - \alpha - 2\log(1/\epsilon) - \log \ell + 2) + \alpha + 2}. \tag{2}$$

The proof of this lemma is more involved and we present it in Sect. 4. Let us first discuss this lemma for more concrete values of the parameters.

Corollary 1. *Fix an arbitrary robust set Γ_v such that $|\Gamma_v| = 2^v$. For a randomly chosen s with probability at least $1 - \xi$ we have that Φ^s is $(\Gamma_v, c, t, 2^{-\lambda})$-secure, for any $c, t, v, \ell, \lambda, \epsilon$ and ξ such that*

$$\xi = 2^{tv - \frac{\ell}{2}(n - c - 3\lambda - 4\alpha - \log \ell - 1) + \alpha + 2}. \tag{3}$$

Proof. Set $\epsilon := 2^{-\alpha-\lambda-1}$ and let $k := n - \lambda - 1 - \alpha - c$. Take Φ^s from Lemma 2. We have that

$$2^\alpha \cdot 2^{-\alpha-\lambda-1} + 2^{\alpha+k+c-n} \leq 2^{-\lambda-1} + 2^{-\lambda-1} \leq 2^{-\lambda},$$

and

$$\xi = 2^{tv - \frac{\ell}{2}((n-\lambda-1-\alpha-c) - \alpha - 2(\alpha+\lambda+1) - \log \ell + 2) + \alpha + 2}$$

which is equal to (3). □

Concrete values If we want to have security against circuits of size χn (for some constant $\chi > 1$) then the size of Γ is equal to $2^{2\chi n \log(\chi n)}$. If we apply it $t = \omega n$ times (for some constant $\omega < 1$) then $tv = 2\chi\omega n^2 \log(\chi n)$. To be more precise set $\lambda := 24$ and $\alpha := 128$, and $n = 1024$. If we set $\chi := 10$, $\omega := 3/25$ then we can allow the adversary to retrieve at most 352 bits by setting $\ell = 1000000$. With these settings we get $\xi = 4 \cdot 10^{-2516}$.

[2] A function is *symmetric* if its output does not depend on the permutation of the input bits. For example every function that just depends on the sum of the input bits is symmetric. See Sect. 3.4 of [28].

Practical considerations The parameter s can be public. Therefore if ξ is negligible, then for the real-life applications s can be just chosen once and for all by some trusted party. For example, one can assume that $s = H(0)||H(1)||\cdots$, where H is some hash function (this of course can be proven secure only in the random oracle model).

Alternatively, we could just assume that s is chosen independently each time $Encode^s$ is calculated, and becomes a part of the encoding. In other words we could define

$$Encode'(m) := (s, Encode^s(m)) \quad \text{and} \quad Decode'(s,x) := Decode^s(x).$$

Of course, in this way the length β of encoding gets larger, and hence if Γ_v is a family of circuits whose size r is some function of β, then v becomes much larger.

4 Proof of Lemma 2

4.1 Auxiliary lemmata

The following was proven in [6].

Lemma 3 ([6]). *Let A, B be random variables where $A \in \mathcal{A}$. Then $P(B = A) \leq d(A|B) + 1/|\mathcal{A}|$.*

We will also use the following lemma (proven in [3]):

Lemma 4 ([3]). *Let Y be an n-bit random variable with $\mathbf{H}_\infty(Y) \geq k$. Let $H = \{h_s\}_{s\in\mathcal{S}}$ be a collection of ℓ-wise independent hash functions $h_s : \{0,1\}^n \to \{0,1\}^\alpha$ (for $\ell \geq 2$). For at least $1 - 2^{-u}$ fraction of $s \in \mathcal{S}$, we have $d(h_s(Y)) \leq \epsilon$ for*

$$u = \frac{\ell}{2}(k - \alpha - 2\log(1/\epsilon) - \log\ell + 2) - \alpha - 2. \tag{4}$$

We will also use the following lemma

Lemma 5. *For every random variables X, Y and an event $\mathcal{E}$ we have*

$$d(X|Y) \leq d(X|Y,\mathcal{E}) + P\left(\overline{\mathcal{E}}\right). \tag{5}$$

Before showing this lemma let us first prove the following:

Lemma 6. *For every random variables X, Y and events $\mathcal{E}, \mathcal{H}$ we have*

$$d(X|\mathcal{H}) \leq d(X|\mathcal{H}\wedge\mathcal{E}) + P\left(\overline{\mathcal{E}}|\mathcal{H}\right). \tag{6}$$

Proof. It is enough to show that

$$\delta(P_{X|\mathcal{H}}; P_{X|\mathcal{H}\wedge\mathcal{E}}) \leq P\left(\overline{\mathcal{E}}|\mathcal{H}\right). \tag{7}$$

After showing this we will be done, since from the triangle inequality we have

$$\overbrace{\delta(P_{X|\mathcal{H}}; U_{\mathcal{X}})}^{=d(X|\mathcal{H})} \leq \overbrace{\delta(P_{X|\mathcal{H}\wedge\mathcal{E}}\ ;\ U_{\mathcal{X}})}^{=d(X|\mathcal{H}\wedge\mathcal{E})} + \delta(P_{X|\mathcal{H}}; P_{X|\mathcal{H}\wedge\mathcal{E}}),$$

where $U_{\mathcal{X}}$ denotes the uniform distribution over $\mathcal{X}$. Let $\mathcal{F}$ denote the set

$$\{x : P(X = x|\mathcal{H}) > P(X = x|\mathcal{H} \wedge \mathcal{E})\}.$$

We have that the left-hand side of (7) is equal to

$$\sum_{x\in\mathcal{F}} P(X = x|\mathcal{H}) - \overbrace{P(X = x|\mathcal{H} \wedge \mathcal{E})}^{=\frac{P(X=x\wedge\mathcal{E}|\mathcal{H})}{P(\mathcal{E}|\mathcal{H})}\geq P(X=x\wedge\mathcal{E}|\mathcal{H})}. \tag{8}$$

$$\leq \sum_{x\in\mathcal{F}} P(X = x|\mathcal{H}) - P(X = x \wedge \mathcal{E}|\mathcal{H}) \tag{9}$$

$$= \sum_{x\in\mathcal{F}} P(X = x|\mathcal{H}) - \sum_{x\in\mathcal{F}} P(X = x \wedge \mathcal{E}|\mathcal{H}) \tag{10}$$

$$= P(X \in \mathcal{F}|\mathcal{H}) - P((X \in \mathcal{F}) \wedge \mathcal{E}|\mathcal{H}) \tag{11}$$

$$\leq P\left(\overline{\mathcal{E}}|\mathcal{H}\right). \tag{12}$$

□

Proof (of Lemma 5). The left-hand side of (5) is equal to

$$\sum_y d(X|Y = y) \cdot P(Y = y), \tag{13}$$

and the right-hand side of (5) is equal to

$$\sum_y \left(d(X|(Y = y) \wedge \mathcal{E}) + P\left(\overline{\mathcal{E}}|Y = y\right)\right) \cdot P(Y = y). \tag{14}$$

To finish the proof it suffices to show that for every y we have

$$d(X|Y = y) \leq d(X|(Y = y) \wedge \mathcal{E}) + P\left(\overline{\mathcal{E}}|Y = y\right).$$

This follows directly from Lemma 6, with $\mathcal{H}$ being the event that $Y = y$. □

4.2 The proof

For the sake of the proof of Lemma 2 we will consider a game that we call a *weak attack* in which $h_s(r) \oplus m$ is hidden from the adversary, and the g_i's are applied only to r. In this game we assume that the adversary $\mathcal{A}_{weak}$, that we call a *weak adversary*, at the end of the interaction outputs some value $Out_{\mathcal{A}_{weak}}$, and we will say that the scheme Φ^s is *weakly* $(\Gamma_v, c, t, \epsilon)$*-secure* if $d(h_s(R)|Out_{\mathcal{A}_{weak}}) \leq \epsilon$, for any $\mathcal{A}_{weak}$,[3] where R is distributed uniformly over $\{0,1\}^n$. A similar proof strategy was used already in [6].

Lemma 7. *For any v, c, t, ϵ and s, if Φ^s is* weakly $(\Gamma_v, c, t, \epsilon)$-secure *then Φ^s is $(\Gamma_v, c, t, \epsilon \cdot 2^\alpha)$-secure.*

[3] Technically, $\mathcal{A}_{weak}$ is a (Γ'_v, c, t)-adversary, where $\Gamma'_v \subseteq \Gamma_v$ contains only such functions g_i, whose output does not depend on the second part (i.e. the final α bits) of the input. Since those functions depend only on the first n bits we will usually think of them as functions defined on $\{0,1\}^n$.

Proof. Take some adversary $\mathcal{A}$ that wins the game described in Sect. 2 with some probability $0.5 + \delta$. We construct a *weak* adversary $\mathcal{A}'$ such that

$$d(h_s(R)|Out_{\mathcal{A}'}) = \delta \cdot 2^{-\alpha}. \tag{15}$$

After showing this we will be done, by setting $\delta := \epsilon \cdot 2^{\alpha}$. The adversary $\mathcal{A}'$ works by simulating $\mathcal{A}$. First, it chooses a random string $z \in \{0,1\}^{\alpha}$ and it starts $\mathcal{A}$. Let m^0, m^1 be the messages that $\mathcal{A}$ outputs. Then, $\mathcal{A}'$ handles the requests issued by $\mathcal{A}$ in the following way. Recall that each request of $\mathcal{A}$ is a function $g_i : \{0,1\}^n \times \{0,1\}^{\alpha} \to \{0,1\}^{c_i}$ that should be applied to $(r, h_s(r) \oplus m)$. Each time such a request is issued, the adversary $\mathcal{A}'$ constructs a request g_i' defined for every r as follows:

$$g_i'(r) := g_i(r, z).$$

(By the robustness of Γ_v we have that if $g_i \in \Gamma_v$ then also $g_i' \in \Gamma_v$.) When the interaction is over and $\mathcal{A}$ outputs b the adversary $\mathcal{A}'$ outputs $Out_{\mathcal{A}'} := m^b \oplus z$. By Lemma 3 we have

$$P\left(Out_{\mathcal{A}'} = h_s(R)\right) \leq 2^{-\alpha} + d(h_s(R)|Out_{\mathcal{A}'}). \tag{16}$$

Now suppose that for some $i \in \{0,1\}$ the following event $\mathcal{X}^i$ occurred: $z = m^i \oplus h_s(r)$. In this case $\mathcal{A}'$ simply simulated the execution of $\mathcal{A}$ against the oracle Ω with $b = i$. Since z is chosen uniformly hence $P\left(\mathcal{X}^0\right) = P\left(\mathcal{X}^1\right) = 2^{-\alpha}$. Therefore the probability that $i = b$ is equal to $0.5 + \delta$. Moreover, in this case, $Out_{\mathcal{A}'} = m^b \oplus (m^i \oplus h_s(r))$, and therefore $Out_{\mathcal{A}'} = h_s(r)$ if and only if $b = i$. Hence we have

$$\begin{aligned} P\left(Out_{\mathcal{A}'} = h_s(r)\right) &\geq P\left(b = i \mid \mathcal{X}^0 \cup \mathcal{X}^1\right) \cdot P\left(\mathcal{X}^0 \cup \mathcal{X}^1\right) \\ &= (0.5 + \delta) \cdot 2^{-\alpha+1} \\ &= 2^{-\alpha} + \delta \cdot 2^{-\alpha+1}. \end{aligned}$$

Combining it with (16) we get (15). □

Before proving the lemma we show the following standard fact.

Lemma 8. *For every U*

$$P_{\boldsymbol{y}:=G_{\mathcal{A}}(R)}\left(|\{r : G_{\mathcal{A}}(r) = \boldsymbol{y}\}| \leq U\right) \leq 2^{c-n} \cdot U. \tag{17}$$

Proof. Since $|G_{\mathcal{A}}(r)| \leq c$, hence the number of all $\boldsymbol{y}$'s is at most equal to 2^c. Therefore the number of r's for which there exists some $\boldsymbol{y}$ such that

$$|\{r : G_{\mathcal{A}}(r) = \boldsymbol{y}\}| \leq U$$

holds is at most $2^c \cdot U$. Hence the probability that it exists for a *random* r is at most $2^{c-n} \cdot U$. Thus we are done. □

Lemma 9. *Suppose that the adversary $\mathcal{A}_{weak}$ performs a weak attack against Φ^s. Then, for any $\epsilon > 0$ and for at least $1 - \xi$ fraction of $s \in \mathcal{S}$ we have*

$$d(h_s(R)|G_{\mathcal{A}_{weak}}(R)) \leq \epsilon + 2^{k+c-n},$$

where ξ is a function of t, v, ℓ, k, and ϵ defined in (2).

Proof. Consider some fixed adversary $\mathcal{A}_{weak}$. Let $Good_{\mathcal{A}_{weak}}$ denote the event that $|\{r : G_{\mathcal{A}_{weak}}(r) = \boldsymbol{y}\}| \geq 2^k$, where $\boldsymbol{y} := G_{\mathcal{A}_{weak}}(R)$. By Lemma 8 (with $U := 2^k$) we get that $P\left(Good_{\mathcal{A}_{weak}}\right) \geq 1 - 2^{k+c-n}$. On the other hand, we have

$$\mathbf{H}_\infty(R|G_{\mathcal{A}_{weak}}(R), Good_{\mathcal{A}_{weak}}) \geq k.$$

Therefore, by Lemma 4 we get that

$$P_s\left(d(h_s(R)|G_{\mathcal{A}_{weak}}(R), Good_{\mathcal{A}_{weak}}) \geq \epsilon\right) \leq 2^{-u}, \tag{18}$$

where P_s means that the probability is taken over the choice of $s \in \mathcal{S}$, and u is defined in (4). From Lemma 5 we get that (18) implies that

$$P_s\left(d(h_s(R)|G_{\mathcal{A}_{weak}}(R)) \geq \epsilon + 1 - P\left(Good_{\mathcal{A}_{weak}}\right)\right) \leq 2^{-u},$$

which implies that

$$P_s(d(h_s(R)|G_{\mathcal{A}_{weak}}(R)) \geq \epsilon') \leq 2^{-u}, \tag{19}$$

where $\epsilon' := \epsilon + 2^{k+c-n}$. Of course (19) holds just for a fixed adversary and to complete the proof we need to give a bound on the value

$$\max_{\mathcal{A}_{weak}} \left(P_s\left(d(h_s(R)|G_{\mathcal{A}_{weak}}(R)) \geq \epsilon'\right)\right). \tag{20}$$

We will do it by applying a union-bound (over all $\mathcal{A}_{weak}$) to (19). However, since that the total number of different adversaries $\mathcal{A}_{weak}$ is doubly-exponential in c,[4] we cannot do it in a straightforward way. Instead, we first observe that

$$\max_{\mathcal{A}_{weak}} P_s\left(d(h_s(R)|G_{\mathcal{A}_{weak}}(R)) \geq \epsilon'\right) = \max_{g_1,\ldots,g_t} P_s\left(d(h_s(R)|g_1(R),\ldots,g_t(R)) \geq \epsilon'\right), \tag{21}$$

Since each $g_i \in \Gamma_v$, and $|\Gamma_v| = 2^v$ we get

$$\max_{\mathcal{A}_{weak}} \left(P_s\left(d(h_s(R)|G_{\mathcal{A}_{weak}}(R)) \geq \epsilon'\right)\right) \leq (2^v)^t \cdot 2^{-u} = 2^{tv-u}. \tag{22}$$

This completes the proof. □

Proof (of Lemma 2). Combine Lemmas 7 and 9. □

5 Connection with the theory of compressibility of NP-instances

We believe that in general the idea to model the leakage as functions from some low complexity class is worth investigating further, as it may lead to new applications of the circuit lower bounds. Interestingly, this is probably the first scenario ever considered in cryptography in which the computing power of the adversary is smaller than the computing power of the users (during some part of the attack). A similar observation was already made in [7] (footnote 3, page 295).

[4] This is because after retrieving c_i bits in the ith round the adversary can choose 2^v different functions g_{i+1}, hence in every round there are $2^{v \cdot 2^{c_i}}$.

It may also be worth exploring some interesting connections between this area and the theory of the compressibility of NP-instances of Harnik and Naor [10]. Informally, an NP-language L is *compressible* if every $x \in \{0,1\}^*$ can be "compressed" to a much shorter string $compress(X)$ (where g is some poly-time function, and $c = |compress(x)| \ll |x|$) such that an infinitely powerful machine M can determine if $x \in L$ just by looking at $compress(X)$. Call this $(PTIME, \infty)$-c-compressibility. Of course, one could generalize this concept and consider any $(\mathcal{P}_0, \mathcal{P}_1)$-compressibility (where $\mathcal{P}_0$ and $\mathcal{P}_1$ are some complexity classes): in this settings we would require that $g \in \mathcal{P}_0$, and the machine M operates in $\mathcal{P}_1$.

For simplicity in this section consider only the one-round LRS's i.e.: $t = 1$ (cf. game in Sect. 2). Moreover, assume that the adversary is poly-time. Informally speaking what we are looking for, when constructing a Γ-secure LRS $\Phi = (Encode, Decode)$ is a class of problems that are not $(\Gamma, PTIME)$-c-compressible on average. More precisely, consider the language L of all valid encodings of some fixed message M. Of course, if this language is $(\Gamma, PTIME)$-c-compressible with some probability ϵ then Φ cannot be $(\Gamma, c, 1, \epsilon)$-secure (as otherwise the adversary could just choose *compress* to be his leakage function). We leave investigating these connections as a future research direction.

References

1. Adi Akavia, Shafi Goldwasser, and Vinod Vaikuntanathan. Simultaneous hardcore bits and cryptography against memory attacks. In Omer Reingold, editor, *TCC*, volume 5444 of *Lecture Notes in Computer Science*, pages 474–495. Springer, 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. Boaz Barak, Ronen Shaltiel, and Eran Tromer. True random number generators secure in a changing environment. In Colin D. Walter, Çetin Kaya Koç, and Christof Paar, editors, *CHES*, volume 2779 of *Lecture Notes in Computer Science*, pages 166–180. Springer, 2003.
4. Yevgeniy Dodis, Yael Tauman Kalai, and Shachar Lovett. On cryptography with auxiliary input. In *Symposium on Theory of Computing (STOC)*, May 2009. to appear.
5. Stefan Dziembowski. Intrusion-resilience via the bounded-storage model. In Shai Halevi and Tal Rabin, editors, *TCC*, volume 3876 of *Lecture Notes in Computer Science*, pages 207–224. Springer, 2006.
6. Stefan Dziembowski and Krzysztof Pietrzak. Intrusion-resilient secret sharing. In *FOCS*, pages 227–237, 2007.
7. 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.
8. Karine Gandolfi, Christophe Mourtel, and Francis Olivier. Electromagnetic analysis: Concrete results. In *CHES*, pages 251–261, 2001.
9. Alex J. Halderman, Seth D. Schoen, Nadia Heninger, William Clarkson, William Paul, Joseph A. Calandrino, Ariel J. Feldman, Jacob Appelbaum, and Edward W. Felten. Lest we remember: cold-boot attacks on encryption keys. *Commun. ACM*, 52(5):91–98, 2009.
10. Danny Harnik and Moni Naor. On the compressibility of np instances and cryptographic applications. In *FOCS '06: Proceedings of the 47th Annual IEEE Symposium on Foundations of Computer Science*, pages 719–728, Washington, DC, USA, 2006. IEEE Computer Society.
11. Yuval Ishai, Amit Sahai, and David Wagner. Private Circuits: Securing Hardware against Probing Attacks. In *CRYPTO*, pages 463–481, 2003.
12. Yevgeniy Dodis Joel Alwen and Daniel Wichs. Leakage resilient public-key cryptography in the bounded retrieval model. In *Advances in Cryptology - CRYPTO*, August 2009. to appear.
13. Jonathan Katz. Signature schemes with bounded leakage resilience. Cryptology ePrint Archive, Report 2009/220, 2009. http://eprint.iacr.org/.

14. Paul C. Kocher. Timing attacks on implementations of diffie-hellman, rsa, dss, and other systems. In *CRYPTO*, pages 104–113, 1996.
15. Paul C. Kocher, Joshua Jaffe, and Benjamin Jun. Differential power analysis. In *CRYPTO*, pages 388–397, 1999.
16. Chi-Jen Lu. Encryption against storage-bounded adversaries from on-line strong extractors. *J. Cryptology*, 17(1):27–42, 2004.
17. 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.
18. Moni Naor and Gil Segev. Public-key cryptosystems resilient to key leakage. In *Advances in Cryptology - CRYPTO*, August 2009.
19. European Network of Excellence (ECRYPT). The side channel cryptanalysis lounge. http://www.crypto.ruhr-uni-bochum.de/en_sclounge.html. retrieved on 29.03.2008.
20. Christophe Petit, François-Xavier Standaert, Olivier Pereira, Tal Malkin, and Moti Yung. A block cipher based pseudo random number generator secure against side-channel key recovery. In Masayuki Abe and Virgil 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.
21. 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.
22. Jean-Jacques Quisquater and David Samyde. Electromagnetic analysis (ema): Measures and countermeasures for smart cards. In *E-smart*, pages 200–210, 2001.
23. Jean-Jaques Quisquater and Franois Koene. Side channel attacks: State of the art, October 2002. [19].
24. David Samyde, Sergei Skorobogatov, Ross Anderson, and Jean-Jacques Quisquater. On a new way to read data from memory. In *SISW '02: Proceedings of the First International IEEE Security in Storage Workshop*, page 65, Washington, DC, USA, 2002. IEEE Computer Society.
25. Adi Shamir and Eran Tromer. Acoustic cryptanalysis. on nosy people and noisy machines. A webpage: http://people.csail.mit.edu/tromer/acoustic/ accessed on 27.05.2009.
26. François-Xavier Standaert, Tal Malkin, and Moti Yung. A unified framework for the analysis of side-channel key recovery attacks. 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 443–461. Springer, 2009.
27. Salil P. Vadhan. Constructing locally computable extractors and cryptosystems in the bounded-storage model. *J. Cryptology*, 17(1):43–77, 2004.
28. Ingo Wegener. *The Complexity of Boolean Functions*. John Wiley and Sons Ltd, and B. G. Teubner, 1987. available at http://eccc.hpi-web.de/eccc-local/ECCC-Books/wegener_book_readme.html.