SAPIENZA
UNIVERSITÀ DI ROMA

Input-shrinking functions: theory and application

Francesco Davì

Submitted to the Department of Computer Science
in partial fulfillment of the requirements for the
Degree of Doctor of Philosophy in Computer Science
at the Sapienza University of Rome

September 2011

SAPIENZA
UNIVERSITÀ DI ROMA

Input-shrinking functions: theory and application

Francesco Davì

Thesis Committee

Dr. Stefan Dziembowski (Advisor)
Prof. Luigi Vincenzo Mancini
Prof. Alessandro Mei

Reviewers

Prof. Miroslaw Kutylowski
Dr. Ivan Visconti

Submitted to the Department of Computer Science
in partial fulfillment of the requirements for the
Degree of Doctor of Philosophy in Computer Science
at the Sapienza University of Rome

September 2011

Author's address:
Francesco Davì
Computer Science Department
Sapienza, University of Rome
Via Salaria 113, 00198 Rome, Italy
e-mail: davi@di.uniroma1.it
www: http://www.dsi.uniroma1.it/~davi/

Notation

List of abbreviations

AES	Advanced Encryption Standard
AKE	Authenticated Key Exchange
AONT	All-Or-Nothing Transform
BRM	Bounded-Retrieval Model
BSM	Bounded-Storage Model
CBC	Cipher-Block Chaining
CDH	Computational Diffie-Hellman
CRS	Common Reference String
DDH	Decisional Diffie-Hellman
DH	Diffie-Hellman
DPA	Differential Power Analysis
DRAM	Dynamic Random-Access Memory
EKE	Encrypted Key Exchange
IC	Ideal Cipher
IKE	Internet Key Exchange
IV	Initial Vector
LRS	Leakage-Resilient Storage
MD	Message Digest
MITM	Man-in-the-middle
PAKE	Password-based Authenticated Key Exchange
PPT	Probabilistic Polynomial-Time
PRG	Pseudo-Random Generator
RO	Random Oracle
SHA	Secure Hash Algorithm
SID	Session Identifier
SPA	Simple Power Analysis
SSID	Sub-Session Identifier
UC	Universal Composability
WKE	Weak Key Exchange

Notation

Δ	statistical distance
$\mathbf{H}_\infty$	min-entropy
$\tilde{\mathbf{H}}_\infty$	average conditional min-entropy
Ext	extractor
Ext_2	two source extractor

$\mathcal{A}$	adversary/attacker
$\mathcal{F}$	ideal functionality
$\mathcal{H}, h$	hash function
Γ	class of leakage functions
λ	leakage
Λ	leakage function
$\mathsf{view}_{\mathcal{A}}$	the values retrieved by an adversary/attacker
ϵ, δ	adversarial advantage
k	security parameter
τ	internal state of a scheme
Φ	a LRS scheme
$\mathcal{K}$	key space
$\mathcal{E}, \mathcal{V}$	events
$\mathbb{F}$	finite field
GF	group field
$\mathcal{G}$	group sampling algorithm
g	group generator
p	order of group, usually prime
$\mathbb{G}$	group
pub	public Diffie-Hellman value
$priv$	private Diffie-Hellman value
X, Y	random variables
x, y	group or set elements
$\mathbf{Pr}$	probability measure
$poly(n)$	polynomial in n
$negl(n)$	negligible in n
$\mathbb{E}$	expected value
$\mathbb{R}, \mathbb{N}$	real and natural number
$[n]$	the natural numbers $\{1, \ldots, n\}$

Contents

List of Figures

List of Tables

Chapter 1

Introduction

For long time, the design of cryptographic schemes was based mostly on heuristic and a scheme was believed secure until someone was able to break it. Then the cryptographers' task was to fix the scheme, mainly providing an ad-hoc solution against the successful attack. These steps could be repeated several times and clearly this approach cannot guarantee security against every possible attacks, even completely new attacks, which should be the target of the design of secure cryptographic schemes.

From the middle of the past century, modern cryptography introduces the *provable security*, the idea to provide a formal description of the security and the adversarial models, to capture all the possible adversarial behaviors, and then to provide formal mathematical proofs of the security of a cryptographic scheme. In this setting, a cryptographic scheme is usually modeled as a *black-box*, i.e. an attacker can choose the input and observe the output of an execution of the protocol but she cannot retrieve any information about the internal state of the scheme.

In the last forty years, this approach developed very fast and attain to design a large number of cryptographic schemes that can be shown to be provably secure. Despite the mathematical proof of the security of such schemes, once they were implemented on some physical devices, it turned out that several of them have been broken. Indeed, by the end of the 90s, the cryptographers' community starts to realize that the adversarial black-box model considered so far does not take into account the possibility for an attacker to exploit the weakness of the real physical implementation of a protocol, which can allow the adversary to retrieve some extra leakage information, stepping out of the formalized model and resulting in breaking the scheme. Such attacks, in which an adversary can retrieve some leakage information during the execution of an implemented cryptographic protocol, are called *side-channel attacks*. The study of these attacks is changing the approach of the cryptographers' community to the design and

implementation of cryptographic schemes. Side-channel attacks exploit physical measurements on real devices and for this reason they have been studied mostly by the practitioners, who try to find countermeasures to the known attacks and to exploit new attacks. Again, the proposed solutions are mostly ad-hoc, often they come without a formal proof of security and cannot provide security against all possible attacks.

In the last years, also the theoreticians began to study the problem of cryptographic security in the presence of side-channel attacks, trying to formally model this leakage information in the provable secure setting. As a result, a new field was born: the *leakage-resilient cryptography*. In this area, the formal security requirements of the provable security hold and, in addition to the black-box interaction with the scheme, an attacker chooses some *leakage functions*, which model the side-channel information, and retrieves the value of these functions applied to the internal state of the cryptographic scheme during its execution on a physical device. Clearly, the class of leakage functions has to be restricted in some way, otherwise an attacker could obtain too much leakage information and easily break any scheme, but, clearly, to be meaningful this class should still cover most of the real world side-channel measurements.

Recently, several construction of cryptographic schemes in the leakage-resilient cryptography were showed together with formal proofs of their security, in models which differ in the class of leakage functions considered or in the way the leakage functions are applied to the internal state of the scheme. Some works consider very restricted class of leakage functions, while other ones consider a more general setting in which an attacker can choose any *input-shrinking* functions, i.e. such that the length in bits of the output is much less than the size in bits of the input. Furthermore, in some models the total amount of leakage retrieved by an attacker is bounded while in other ones it is bounded only in each time period but not in total (in this case, an update phase of the internal state of the scheme is needed, as otherwise after some number of time period an attacker could completely retrieve the entire internal state of the scheme).

The idea to reason about the partial key leakages by modeling them as input-shrinking functions originates from the Bounded-Retrieval Model (BRM). Originally the BRM was proposed as a method for protecting against computer viruses that may steal large amounts of data from the PCs: the main idea of the BRM is to construct schemes where the secret key is large and to assume that the adversary can retrieve the value of some input-shrinking functions applied to the secret key. The main differences between this setting and the models for the side-channel attacks come from the fact that the keys in the BRM are huge and hence:

- one has to design schemes where the honest user does not need to frequently

process the entire secret key;

- one can allow that some part of the secret key leaks each time the scheme is used.

Nevertheless in [68] it was observed that the BRM can be used to model the side-channel attacks and recently Alwen et al. [7] showed how to construct leakage-resilient public-key protocols in the BRM.

Leakage-resilient cryptography is growing very fast, providing a large number of cryptographic schemes that are provably secure even after the implementation on physical machines that may leak information. However, there is still a gap between the theoreticians' and the practitioners' point of view on this field, about how to meaningful model real world side-channel measurements and on which assumptions are not only meaningful in theory but also practical. A collaboration between the two cryptographic community, the theoreticians and the practitioners, is needed to further develop the area: sharing the knowledge is the key for better understand the side-channel attacks and to construct a provably secure physically implementable cryptography.

In this thesis, we contribute to this emerging field of the cryptography by studying the problem of secure data storage on hardware that may leak information, introducing a new primitive, a *leakage-resilient storage*, and showing two different constructions of such storage scheme provably secure against a class of leakage functions that can depend only on some restricted part of the memory and against a class of computationally weak leakage functions, e.g. functions computable by small circuits, respectively. Our results come with instantiations and analysis of concrete parameters. Furthermore, as second contribution, we present our implementation in C programming language, using the cryptographic library of the OpenSSL project, of a two-party Authenticated Key Exchange (AKE) protocol, which allows a client and a server, who share a huge secret file, to securely compute a shared key, providing client-to-server authentication, also in the presence of active attackers. Following the work of Cash et al. [42], we based our construction on a Weak Key Exchange (WKE) protocol, developed in the BRM, and a Password-based Authenticated Key Exchange (PAKE) protocol secure in the Universally Composable (UC) framework. The WKE protocol showed by Cash et al. uses an explicit construction of averaging sampler, which uses less random bits than the random choice but does not seem to be efficiently implementable in practice. In this thesis, we propose a WKE protocol similar but simpler than that one of [42]: our protocol uses more randomness than the Cash et al.'s one, as it simply uses random choice instead of averaging sampler, but we are able to show an efficient implementation of it. Moreover, we formally

adapt the security analysis of the WKE protocol of [42] to our WKE protocol. To complete our AKE protocol, we implement the PAKE protocol showed secure in the UC framework by Abdalla et al. [2], which is more efficient than the Canetti et al.'s UC-PAKE protocol [40] used in [42]. In our implementation of the WKE protocol, to achieve small constant communication complexity and amount of randomness, we rely on the Random Oracle (RO) model. However, we would like to note that in our implementation of the AKE protocol we need also a UC-PAKE protocol which already relies on RO, as it is impossible to achieve UC-PAKE in the standard model [40].

In our work we focus not only on the theoretical aspects of the area, providing formal models and proofs, but also on the practical ones, analyzing instantiations, concrete parameters and implementation of the proposed solutions, to contribute to bridge the gap between theory and practice in this field.

1.1 Roadmap

This thesis is organized as follows:

- Chapter 2 briefly introduces basic definitions and concepts used throughout the thesis;
- Chapter 3 gives an overview on the state of the art of leakage-resilient cryptography;
- Chapter 4 presents our leakage-resilient storage primitive and constructions;
- Chapter 5 discusses our implementation of an Authenticated Key Exchange protocol;
- Chapter 6 contains conclusion and future work.

Chapter 2

Preliminaries

In this chapter we briefly introduce some basic definitions and review some basic concepts that are used in this thesis.

2.1 Definitions

2.1.1 Statistical distance

Definition 1 (Statistical distance). *Given two random variables X_0, X_1 with values in $\mathcal{X}$, their statistical distance is defined as*

$$\Delta(X_0; X_1) \stackrel{\mathsf{def}}{=} \frac{1}{2} \sum_{x \in \mathcal{X}} |\boldsymbol{Pr}[X_0 = x] - \boldsymbol{Pr}[X_1 = x]\,|.$$

If a random variable X assumes values in $\{0,1\}^n$, then we let $d(X) \stackrel{\mathsf{def}}{=} \Delta(X; U_n)$ be the statistical distance[1] between X and the uniform distribution U_n over $\{0,1\}^n$. If $\Delta(X_0; X_1) \leq \epsilon$ we say that X_0 and X_1 are ϵ-close (or ϵ-indistinguishable). We also define the conditional statistical distance given a random variable Y as $\Delta(X_0; X_1|Y) \stackrel{\mathsf{def}}{=} \Delta(X_0, Y; X_1, Y)$ and $d(X|Y) \stackrel{\mathsf{def}}{=} \Delta(X, Y; U_n, Y)$.

2.1.2 Entropy

Definition 2 (Min-entropy). *Given a random variable X, the min-entropy of X is*

$$\mathbf{H}_\infty(X) \stackrel{\mathsf{def}}{=} \min_x(-\log(\boldsymbol{Pr}[X = x])) = -\log(\max_x(\boldsymbol{Pr}[X = x])).$$

[1] We will overload the symbols $\Delta(\cdot)$ and $d(\cdot)$ and sometimes apply them to the probability distributions instead of the random variables.

Given two random variables X, Y, the *conditional min-entropy* of X given Y is defined as

$$\mathbf{H}_\infty(X|Y) \stackrel{\mathsf{def}}{=} \mathbb{E}_{y \leftarrow Y}\mathbf{H}_\infty(X|Y=y) = \mathbb{E}_{y \leftarrow Y}(-\log(\max_x(\mathbf{Pr}\left[X=x|Y=y\right]))),$$

where $\mathbf{H}_\infty(X|Y=y)$ is the min-entropy of X after learning the value y of Y. However, this notion is not really useful from a cryptographic point of view. For cryptographic applications Dodis et al. [64] introduced the notion of *average conditional min-entropy*:

Definition 3 (Average conditional min-entropy)**.** *Given two random variables X, Y, the average conditional min-entropy of X given Y is*

$$\tilde{\mathbf{H}}_\infty(X|Y) \stackrel{\mathsf{def}}{=} -\log(\mathbb{E}_{y \leftarrow Y}(\max_x(\boldsymbol{Pr}\left[X=x|Y=y\right]))).$$

2.1.3 Extractors

A *randomness extractor* is a function that, given a long source of size n with high min-entropy and a short uniformly random seed, outputs a long sequence of size m, with $m < n$, that is statistically close to uniform.

Definition 4 (Extractor)**.** *A (k, ϵ)-extractor is a function*

$$\mathsf{Ext} : \{0,1\}^n \times \{0,1\}^d \to \{0,1\}^m$$

such that for every distribution X on $\{0,1\}^n$ with $\mathbf{H}_\infty(X) > k$ the distribution $\mathsf{Ext}(X, U_d)$ is ϵ-close to the uniform distribution on $\{0,1\}^m$.

In [44], Chor and Goldreich showed that it is not possible to use one function to extract randomness from every single high min-entropy source but it is possible from two independent high min-entropy sources. They introduced the two-source extractor:

Definition 5 (Two source extractor)**.** *A (k_0, k_1, ϵ)-two source extractor is a function*

$$\mathsf{Ext}_2 : \{0,1\}^{n_0} \times \{0,1\}^{n_1} \to \{0,1\}^m$$

such that for every two independent distributions X on $\{0,1\}^{n_0}$ with $\mathbf{H}_\infty(X) > k_0$ and Y on $\{0,1\}^{n_1}$ with $\mathbf{H}_\infty(Y) > k_1$ the distribution $\mathsf{Ext}_2(X, Y)$ is ϵ-close to the uniform distribution on $\{0,1\}^m$.

2.1.4 Universal Hash Functions

Definition 6 (ℓ-wise independent universal hash functions)**.** *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 universal 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 $n' \leq n$ we define

$$h_s(x) = \left(\sum_{i=0}^{\ell} s_i x^i\right)_{1\ldots n'}$$

(where $z_{1\ldots n'}$ denotes the set of n' first bits of z) then $\{h_s\}$ is a collection of ℓ-wise independent universal hash functions.

2.2 Perfect Security and Cryptographic Hardness Assumptions

In modern cryptography, to design a cryptographic scheme one has to formally define a security model and an adversarial model. A scheme is called *information-theoretically secure*, or *perfect secure*, if it is mathematically proven secure against a computationally unbounded adversary. However, most of the cryptographic constructions consider only "efficient" adversaries, modeled by *probabilistic polynomial-time* (PPT) Turing machines, which use some randomness and terminate after a polynomial number of steps, in the length of its input, or by bounded size circuits, where the size of a circuit is the number of its logic gates. This approach is called *computational security* and even if it is theoretically weaker than perfect security, it suffices for all practical purposes.

Computational security is based on notions from complexity theory and considers all the parameters involved in the computations as functions of a *security parameter* k, chosen by the honest parties when they run the protocol. Then, in this setting a scheme is formally proven secure against an adversary with running time polynomial in k and which can break the scheme only with *negligible*[2] probability in k, with large enough value of the security parameter k.

Most of the security proofs for modern cryptographic protocols rely on the assumption that some problems are hard to solve for computationally bounded

[2]A function $f : \mathbb{N} \to \mathbb{R}$ is negligible in k if for every polynomial $poly(\cdot)$ there exists an integer N such that for all $k > N$ it holds that $f(k) < 1/poly(k)$.

adversaries. Informally speaking , these proofs proceed by reduction, showing that if there exists an efficient (computationally bounded) adversary who breaks the scheme with no-negligible probability, then it is possible to efficiently use this adversary to break the problem that was assumed to be hard. Several of these hard problems come from number theory, as factoring prime numbers and computing discrete logarithms, and cryptographers developed several variety of assumptions based on them that are believed to be hard to solve in polynomial time. This model, in which the security proofs are based on some assumptions on the computational power of an adversary, is called the *standard model*.

We briefly define two standard assumptions that are used throughout this thesis. Let $\mathcal{G}(1^k)$ be a group sampling algorithms which outputs a tuple $(\mathbb{G}, p, g)$, where $\mathbb{G}$ is a cyclic group of order p and $g \in \mathbb{G}$ is a generator of $\mathbb{G}$.[3]

Definition 7 (Computational Diffie-Hellman (CDH) problem)**.** *Let x, y uniformly distributed over $\mathbb{Z}_p$ and $(\mathbb{G}, p, g) \leftarrow \mathcal{G}(1^k)$. The CDH problem is hard for $\mathbb{G}$ if for every adversary $\mathcal{A}$ there exists a negligible function $negl(\cdot)$ such that*

$$\boldsymbol{Pr}[\mathcal{A}(\mathbb{G}, p, g, g^x, g^y) = g^{xy}] \leq negl(k).$$

Definition 8 (Decisional Diffie-Hellman (DDH) problem)**.** *Let x, y, z uniformly distributed over $\mathbb{Z}_p$ and $(\mathbb{G}, p, g) \leftarrow \mathcal{G}(1^k)$. The DDH problem is hard for $\mathbb{G}$ if for every adversary $\mathcal{A}$ there exists a negligible function $negl(\cdot)$ such that*

$$|\boldsymbol{Pr}[\mathcal{A}(\mathbb{G}, p, g, g^x, g^y, g^{xy}) = 1] - \boldsymbol{Pr}[\mathcal{A}(\mathbb{G}, p, g, g^x, g^y, g^z) = 1]\,| \leq negl(k).$$

In other words, given the tuple $(\mathbb{G}, p, g)$ and the randomly chosen g^x and g^y, the CDH problem is to compute the Diffie-Hellman value g^{xy} while the DDH problem is to distinguish the Diffie-Hellman value g^{xy} from a randomly chosen group element g^z.

2.2.1 The Random Oracle Model

Several of the cryptographic schemes proven secure in the standard model are not practical and there exist cryptographic primitives that cannot be proven secure in this model. For these reasons, Bellare and Rogaway [14] introduced the *random oracle* (RO) *model*, an idealized model in which all the parties have access to an "oracle", a random function that outputs a uniformly random element for every input provided by the parties, but the parties have no knowledge about its internal behavior.

[3] In practice, prime order groups are desirable.

In practice, the random function is instantiated by a cryptographic hash function (as SHA-1), i.e. each party, instead of querying the oracle with a value, simply computes the cryptographic hash function of that value on its own. However, a proof in the RO model does not imply that the scheme will remain secure when the random oracle is implemented in practice. Indeed, some schemes (even if quite artificial) have been presented that are provably secure in the random oracle model but are insecure for any choices of the cryptographic hash function used to replace the oracle (see for example [37, 38]). Despite this limitation, a formal proof for a cryptographic scheme in the RO model could provide meaningful information about the scheme and its properties and could help to develop better schemes.

The Ideal Cipher Model. Another idealized model is the *ideal cipher* (IC) *model* ([78, 108, 50, 22]). In this model there exists a public random block cipher and all the parties can make encryption and decryption queries to it. Also in this model, some schemes have been showed that are secure in the IC model but becomes insecure once the ideal cipher is replaced by a real block cipher (see [21]). Nevertheless, as noted in [21], a proof in the IC model can still be useful as "it shows that a scheme is secure against generic attacks, that do not exploit specific weaknesses of the underlying block cipher". Coron et al. [46, 47] showed that the RO model and the IC model are equivalent.

2.3 Bounded-Retrieval Model

In [120] and [121], Maurer introduced the Bounded-Storage Model (BSM), which only assumes a bound on the storage capacity of an attacker, instead of the classical approach to bound her computational power. In particular, the BSM assumes that a large random string is available for a limited amount of time to all the parties who run the cryptographic scheme, but the quantity of information that an attacker can store is less than the size of the string and then she can only save partial information about it. Several works ([9, 73, 117, 151]) showed that in this setting the honest parties can generate, using a shared secret short string, a very long string such that an attacker has no information about it.

More formally, the parties share a short string key chosen uniformly at random from a key space $\mathcal{K}$ and a long random string $R \in \mathcal{R}$ (a randomizer) is available to all the parties. In order to obtain a string much longer than the shared one, all the parties apply a key-expansion function $f : \mathcal{R} \times \mathcal{K} \to \{0,1\}^n$, with $|R| \gg n \gg |key|$, to compute the expanded key $f(R, key)$. Clearly, the function f has to use only a small portion of R, as in this way the honest parties can compute it efficiently. An adversary can apply an arbitrary input-shrinking

function Λ to R, i.e. $\Lambda(R) < |R|$, and store the result. After this step, the adversary cannot access anymore the string R. A key-expansion function f is secure in the BSM if, with high probability, an adversary gains no information about the expanded key $f(R, key)$ even conditioned on the knowledge of $\Lambda(R)$ and key.[4]

In this model, several schemes were proven information-theoretically secure, as encryption ([9, 73, 117, 151, 53]) and key-agreement ([34, 72]).

Based on the BSM, the Bounded-Retrieval Model (BRM) was introduced independently by Dziembowski [68] and Di Crescenzo et al. [51]. In this model, the parties share a huge (several Gygabytes) random secret key K and an adversary is allowed to perform any computation on it but the quantity of information that she can retrieve is bounded, i.e. it is less than $|K|$ but it can even consist of some Gygabytes. This model captures the setting in which one wants to protect secret keys and to preserve security of cryptographic schemes on a computer infected (for a limited amount of time) by a malware, like a virus, which can communicate only a bounded amount of information to a malicious user, after performing any efficient computation on the entire internal state of the infected machine.

More precisely, an adversary can apply any input-shrinking function Λ to K and retrieve the result and after this step it is assumed that the parties are able to remove the malware program from their machine and then they use only a small portion of their shared secret key K to perform some cryptographic scheme. Informally, a scheme is secure in the BRM if the secret key K remains indistinguishable from random to an adversary even conditioned on the knowledge of the retrieved information $\Lambda(K)$.

The assumption that an adversary cannot retrieve more than some Gygabytes is quite practical as an honest user should be able to notice whether her machine is transmitting such huge amount of information to the external world, independently by her actions.

In the past years, authentication schemes [68, 51, 42] and secret-sharing [74] were proven secure in the BRM. Recently, Alwen et al. [7, 6] showed how to build the first public-key primitives (identification schemes, signatures, authenticated key agreement and encryption) in the BRM based on a variety of assumptions.

2.4 Universally Composable Security

In [35] Canetti introduced the Universally Composable (UC) framework in which security is based on an ideal functionality $\mathcal{F}$, which can be viewed as a trusted

[4] One assumes that the adversary knows key to prove a strong result but in practice key is kept secret by the parties.

party that interacts with a group of players to compute some given function f. The functionality $\mathcal{F}$ receives the input of each party, computes f on these values and gives back the output to the players. Hence, in this setting an attacker can only retrieve the data of the corrupted players but security is guaranteed for the honest parties.

Informally, a protocol proven secure in the UC framework remains secure even when used in arbitrary "environments", i.e. the behavior of a protocol which implements an "ideal functionality", under an adversarial attack, is not distinguishable from the behavior of a simulated attack on the "ideal functionality", by an "environment" that controls the parties who run the protocol, choosing inputs and observing outputs, during an attack. By this property, an "ideal functionality" can be used in any construction and then it can be replaced by an UC secure protocol, guaranteeing that the security analysis of the construction still holds.

In the UC framework there may exist several copies of the ideal functionality $\mathcal{F}$ running at the same time. Each copy has a unique session identifier (SID). The original UC theorem analyzes the security of a single protocol, but it does not consider the case in which several protocols share state and randomness.

In [41] Canetti and Rabin introduced the UC with joint state, which guarantees security in case different protocols share some common state. Informally, they define a multisession extension $\mathcal{F}^*$ of $\mathcal{F}$, which runs several executions of $\mathcal{F}$. Each copy of $\mathcal{F}$ is identified by a sub-session identifier (SSID).

Chapter 3

From Provable Security to Leakage-Resilient Cryptography

By the end of the 70s, modern cryptography introduced a new approach to the design of cryptographic schemes, requiring formal definitions and precise assumptions to achieve mathematical proofs of security in well-defined and meaningful models.
Previously, the security of a cryptographic scheme was based mostly on heuristic: a scheme was believed secure until someone was able to break it. Once a scheme was broken, the designers tried to fix it using an ad-hoc solution against the attack that successes in breaking the scheme. Such evolution of the cryptographic schemes is clearly dangerous: even providing security against all the known attacks might be not enough as there could be some adversarial behaviors not discovered yet that in this way are not considered in the design phase.

3.1 Provable Security

In 1984, Goldwasser and Micali [88] introduced the idea to design a cryptographic scheme by formalizing the adversarial model and the security model and then formally proving that, under some well-defined assumptions (see Section 2.2), no adversary can break the security definitions of the scheme. Such approach has been later called *provable security* as the cryptographic protocol now comes with a mathematical proof of their security.
Provable security models a cryptographic protocol as a *black-box*: an adversary can only know, or choose, the input and see the output of the protocol without obtaining any information about the internal state of the protocol during its execution. The security model of the protocol formally describes the interaction between the adversary and the black-box protocol, specifying what actions she

can perform and what she can learn. A security model can consist of a game between the adversary and the protocol or it can be simulation based. In the former case, the attack of the adversary is formalized by a game and the probability of the adversary to win such game represents the probability to break the protocol, hence the protocol is secure if this probability is negligible. In the simulation based security model, the behavior of a protocol under an adversarial attack is not distinguishable from the behavior of a simulated attack, in which a simulator with no access to the internal state of the protocol has to simulate the adversary. In this way, one shows that whatever the adversary learns from her attack can be derived by observing the execution of the protocol without knowing anything about its internal state. To show that a protocol is secure in this setting, one has to define a simulator such that its output distribution is indistinguishable from that one of the adversarial attack.

Despite the strong requirements of the provable security, from the mid-90s cryptographers began to discovered that once a secure protocol, even with a formal security definition and mathematical proof of security, is implemented on a physical device, it turns out that such implementation could be not secure anymore. Indeed, an adversary who wants to break the protocol can try to exploit the weakness of the physical device, which can leak some information about the implemented protocol that are not considered in the black-box model. An attack of this type, in which the adversary exploits the leakage information of a device during the execution of a cryptographic protocol implemented on it, is called a *side-channel* attack (in contrast to "main-channel" attacks which exploit the device as a black-box).

3.2 Side-Channel Attacks

In 1996, Kocher [112] showed that many implementations of secure cryptographic protocols can be broken by a *timing attack*, in which the attacker can measure the amount of time required by the private key operations during the execution of the protocol. The time required to compute some operations can depend on the secret key and the input of the protocol, therefore if an adversary is able to make reasonably accurate timing measurements on a vulnerable system then she could break the protocol with a computationally inexpensive attack.
Clearly, the measurements can contain some noise that could make the retrieved information useless to an attacker or could make hard to extract useful information from it. In his work, Kocher showed a statistical method to get rid of this problem, allowing the adversary to detect errors in the measurements and correct them (at the cost of a more expensive amount of resources used by the

attacker).
Moreover, Kocher noted that even if the protocol is implemented introducing a timer or just no meaningful operations to delay the response of the operations performed during the execution of the protocol (at the cost to slow down the execution, of course), an adversary can study the power consumption of the system to detect the real duration of any operation, as the power used by the protocol computation and the added operations (or during the delay) could be different, potentially allowing the attacker to still perform a timing attack.

In 1999, Kocher et al. [113] studied methods to analyze the power consumption measurements made on a tamper resistant device to obtain information about the secret data of the device. They introduced the *power analysis attack*, which exploits the correlation between the power consumptions and the operations performed during the execution of a cryptographic protocol on the device. The power analysis attack is the most studied side-channel attack and it is frequently used in practice, as it is relatively simple to perform but still it makes possible to completely discover the secret data of the system.
There exist two types of such attacks: *simple power analysis* (SPA) and *differential power analysis* (DPA).
SPA examines few power traces, which represents the current used by the system over the time, by visual inspection. Such analysis requires a limited access to the device but it needs the knowledge of the implementation of the device and works better if the operations executed on the device during the measurements are performed sequentially and the power consumption of every operation is known. Clearly, for an attacker could be hard to obtain such information about the device and the operations.
DPA does not require to know the implementation of the analyzed device and retrieves a small part of the secret data by studying the relation between power consumption and the operations performed at a given moment. To retrieve the entire secret key, the attacker has to repeat the attack choosing a different point in the time in which the computation depends on a different part of the secret data. DPA works even in presence of noise in the power traces, but it needs several measurements, therefore, in contrast to the SPA, it requires the control of the device for some time.

One of the most famous example of a successful side-channel attack, based on power analysis, is that one against the KeeLoq cipher [76], used in a remote keyless system for cars, which has been completely broken.

Other possible side-channel attacks exploit different leakage of the physical device, e.g. by measuring, during the execution of the target protocol, electromagnetic radiations [82, 134] and acoustic signals [144]. All these attacks are not invasive as they do not tamper the device but simply observe and measure the

leakage from it.
In contrast, invasive attacks [20, 25] allowed the adversary to observe "inside" the device, e.g. by measuring the power consumption directly from the wires, performing a so-called *probing attack*, or reading directly from the memory the contents of the cells, in a *cold boot attack*.
Probing attack, in which it is possible to simply read the contents carried by some wired of the device, required a rather complex tools to perform measurements, while cold boot attacks rely on the fact that dynamic random-access memory (DRAM) retains its contents for several seconds even after it is disconnected from power or from the machine on which is working. In [91], Halderman et al. showed that the information on the DRAM persist for minutes or even hours, if the device is frozen.

In the *fault attacks* [8, 25, 146] it is possible to introduce faults during the execution of the protocol on the device analyzed, e.g. by modifying the temperature, using radiation or changing some internal setting of the device as clock or power. Such attacks are really powerful as even only one single fault, a single change inserted in the computation, can allow an adversary to completely break an implementation of a protocol.

3.2.1 Countermeasures

A countermeasure against side-channel attacks tries to make the measurements of some physical properties of the device, during the execution of protocols on it, independent from the computation or to make harder to perform such measurements for an attacker.

On the hardware level, side-channel countermeasures do not depend on some specific implementation but rather they try to decrease the quantity of information that an attacker is able to retrieve in a measurement or is able to efficiently extract from the observed leakage of the physical device. To protect the device and to limit the physical access to it and its leakage, special hardware is used to shield its components but a good solution for an attack could not work properly against another type of side-channel, weakening the countermeasures.
Another countermeasure consists in adding noise to the adversarial observations of the device, e.g. by randomizing the power consumption of the system or the execution of the protocol's operations (inserting interruptions or dummy instructions). An adversary would then need to perform a larger amount of measurements to recover a better signal, increasing the cost of the attack.

At logical level, countermeasures to power analysis[1] are based on the concepts of *hiding* and *masking* (or blinding) [77, 123, 149], which try to make the power consumption of a device independent of the values computed during the execution of a protocol on that device.
Hiding techniques try to make the power consumption of a device constant but then a drawback is that the implementation always uses more power than what it needs in the computation, making this approach not practical on some resources-bounded device.
Masking techniques try to avoid correlation between the values computed during the computation of a protocol and the computation itself. In a circuit, the intermediate values computed by logical gates are bits and it is possible to mask them, for example, by xoring them with a random mask, generated by the circuit and potentially different for each invocation. The logical gates will then use masked values and corresponding masks instead of single bits and, as the masked bit is not correlated to the original one, the total power consumption is independent of the intermediate values of the computation, unless an attacker retrieves both the masked value and the mask, which allow her to simply know the corresponding original bit.
The number of random masks used in the implementation affects the efficiency of the countermeasure and the resistance to the attacks [43]: a single mask does not influence the efficiency of the operations performed but can easily be identified from the attacker by visual inspection of the power traces [141], while the best resilience to attacks is obtained by using a mask for each value but it is not efficient and then the system needs to deal with the problem of generating a large number of random masks.
The masking technique is applied also on the algorithmic level, to make the power consumption independent of the intermediate value of specific cryptographic protocols [90, 150, 129, 142].

A strong side-channel countermeasure at algorithmic level consists of updating the secret key used by the cryptographic protocol, for example, by simply applying to the key a cryptographic hash function, after every fixed number of operations or fixed amount of time. As a drawback, such technique requires an interaction between the parties who run the protocol with the same key to execute the update and needs to study also the leakage that can be measured by an attacker during the update phase.

In real implementations, some of the side-channel countermeasures introduced above are combined together, in particular logical and algorithmic techniques work under the assumption that an attacker is not able to retrieve the entire internal state of a device in a single measurement and the hardware countermea-

[1] See [119] for a complete treatment of power analysis attacks and countermeasures.

sures are the only ones which guarantee such requirement in practice.

3.3 Leakage-Resilient Cryptography

As illustrated in the previous section, the side-channels attacks and countermeasures are studied mostly by the practitioner's cryptographic community, which try to find new attacks and then to propose new countermeasures. Such ad-hoc approach produces cryptographic implementations that are secure against a some particular side-channel attacks, even all the known attacks, but they do not provide security against every kind of possible attacks, even attacks not known yet. Moreover, such solutions often do not come with a formal proof of their security, in contrast with the modern cryptography approach and the formal models of the provable security.

In the last years, the theoretician's cryptographic community began to focus on the side-channel attacks, on how to incorporate them in the models already known and on how to design new cryptographic schemes that are provably secure in this settings. The result of this collaboration between practitioners and theoreticians is the *leakage-resilient cryptography*, which deals with the problem of designing cryptographic protocols that are provably secure even after the implementation on machines that may leak information. In this setting, the leakage of a physical device is modeled by a class of functions and the formal proof of the security of a scheme is based on some assumptions on the physical implementation of the scheme. Clearly, it should be simpler to implement such assumptions than to build a black-box version of the scheme.

3.3.1 The Leakage Functions

In the leakage-resilient cryptography, an attacker can choose some functions from a large well-defined class and retrieve the value of such functions applied to the internal state of the implemented scheme during the computations, for which she can choose the input and observe the output, as in the black-box approach. The class of functions used to model the leakage has to be very realistic, i.e. it should cover all the attacks that an adversary can launch in the real world, but it has to be restricted in some way, as the class cannot contain any functions. For example, the leakage functions class cannot contain the identity function as otherwise an attacker can use it to trivially know the entire internal secret state of an implemented scheme and then, of course, any hope for security is gone.
Limiting the class of functions that an attacker can apply to retrieve side-channel information from a physical device makes sense also from the practical point of view. Indeed, an adversarial physical measurement usually consists of several

megabytes of information but many real attacks compress this measurement and exploit only a small fraction of the retrieved leakage. Moreover, most of the measurements could not reveal the secret state of a scheme to an attacker, as the measurements could contain noise or it could be computationally hard for the attacker to extract meaningful information from such measurements.
The results in this area can be categorized according to the class of leakage functions that the scheme covers: some works consider very restricted classes (e.g. in [98] the model assumes that the adversary can simply read-off some wires that represent the computation), while other ones consider more general leakages. A common general restriction on the class of functions used to model the leakage is to allow the adversary to choose any *input-shrinking* function, i.e. such that the length in bits of the output is much less than the size in bits of the input, the secret state in our case, which bounds the amount of leakage that the adversary can retrieve during an attack (e.g. see [75, 5, 7, 127, 131, 80]).
Some works ([75, 127, 131, 109, 159]) consider the possibility to relax this restriction by arguing that for their security proof it is enough to require that (a variant of) the min-entropy of the secret state of the scheme does not decrease too much conditioned on the leakage.[2] In this setting, there is not a limit on the amount of leakage that an attacker can retrieve but the restriction is less intuitive than simply bounding the leakage and it is not clear whether this approach is feasible in practice.

Another way to restrict the leakage consist to consider functions that are computationally bounded, e.g. functions computable by circuits of small size, or functions that are noisy, i.e. such that their output contains some noise. Such restrictions model several practical side-channel attacks, as allowing the adversary to choose any efficient functions seems to not always fit the real world.
In Chapter 4, we consider the problem of how to securely store a secret on a device that may leak information, introducing a new primitive, a *leakage-resilient storage* (LRS), and we show a construction of LRS which implies security in a computationally bounded leakage functions setting, in which, for example, the attacker can choose functions computable by circuits of a small size (see Section 4.3.2 for more details).

An attacker can choose the leakage functions adaptively or non-adaptively: in the former case the functions are chosen during the attack and the choice of each function is based on the knowledge of the outputs of the functions already applied, while in the latter case all the functions are chosen before starting the attack. Most of the works in this area consider adaptive adversaries, as this scenario is theoretically stronger then the non-adaptive one, but it can be hard

[2] As for min-entropy, λ bits of leakage cannot decrease it by more than λ bits.

to realize adaptivity in practice, as the leakage functions are often determined by the device and the measurement equipment.

3.3.2 Continual Leakage Model

The seminal work of Micali and Reyzin [124] introduced the framework of *physically observable cryptography*, the first formal model for the side-channel attacks. They observed that several provable security cryptographic schemes do not provide security anymore in the presence of leakage and proved the security of the implementation of some already known cryptographic schemes under some assumptions[3] on the physical world and the side-channel attacks.
The most important assumption stated in [124] is *only computation leaks information*, in which an attacker retrieves leakage information only from the internal state of a cryptographic scheme used in the computation for each time period (or each invocation of the scheme), while the parts of the internal state that are not accessed do not leak any information in that time period (or invocation). In this setting, even if the leakage in each time period has to be restricted in some way, the overall amount of information that an attacker can retrieve during the entire lifetime of the system is *unbounded*. Clearly, to achieve any security in this model, called the *continual leakage model* as a scheme can leak continually, the secret data of the system have to be updated periodically, to avoid that an attacker could retrieve too much information about them (as the total leakage is unbounded).
The assumption that the parts of the system's memory that are not involved in the current computation do not leak any information turned out to be not satisfied by several physical implementations, for example, the cold boot attack [91] showed that DRAM leaks information even if there is not computation on it and some practical attacks on not accessed memory were also demonstrated in [140]. Therefore, to produce secure schemes in this model one has to check whether the physical device on which the scheme is implemented satisfies the "only computation leaks information" assumption.
For this reason, some works (e.g. [75, 131, 109, 70]) try to relax this restriction by assuming that the part of the system's memory used and the part of the memory not used during the computation, leak independently. In this setting, in each time period an adversary chooses two functions and retrieves the value of one function applied to the active part of the system's memory in that time period and the value of the other function applied to the passive part of the memory. Then, an attacker can combine the information obtained from the two different

[3]Micali and Reyzin referred to these assumptions as "axioms" but some of them turned out to not hold for many real devices.

parts of the memory to evaluate a function of the global memory, in particular she can compute any linear combination of the retrieved leakage. However, recent works (e.g. [147]) showed that also this weaker restriction does not model some non-linear physical leakage.

In the continual leakage model, stream ciphers [75, 131], digital signature schemes [80], pseudorandom functions and permutations, in the non-adaptive setting [65], and public-key encryption, under some non-standard assumptions [109], have been built and formally proven secure.
Moreover, some general compilers have been proposed to securely realize any cryptographic primitives in this model by transforming any circuit in a leakage-resilient one [81, 100, 89], assuming the possibility to use some (simple) leak-free hardware components. These compilers represent a theoretical result of feasibility but they are rather inefficient in practice.
Recently, Dziembowski and Faust [70] proposed an extension in this model of the leakage-resilient storage primitive introduced in the Chapter 4 of this thesis, allowing for the refreshing of the stored data, assuming that different parts of the memory leak independently and using some simple leak-free components, and showed an efficient implementation of a public-key primitive based on their scheme.

3.3.3 Bounded Memory-Leakage Model

As showed in the previous section, assuming some restrictions on the type of leakage of a system's memory is controversial from a practical point of view. For this reason, the *bounded memory-leakage model*, complementary to the continual leakage model, has been introduced by Akavia et al. [5]. In this model, an attacker can retrieve side-channel information from the entire system's memory (for this reason such an attack is also called *memory attack*) but the overall amount of leakage is bounded (as the schemes built in this model does not provide methods to refresh their secret data). As for the previous model, the leakage is modeled by allowing the attacker to choose some functions from a large and well-defined class of functions, which should cover all the realistic side-channel attacks. However, the restriction of the amount of leakage that an attacker can retrieve during the entire lifetime of the system does not allow to model several practical attacks based on long term observation of the device with many measurements over time.

One of the first works in this model is that one of Ishai et al. [98], which deals with the problem of building general compilers for transforming a circuit in a new one, secure against an attacker who can read from a bounded number of wires during the computation.

Recently, in [5], the authors showed that some known provably secure schemes [137, 84] remain secure in the bounded memory-leakage model, under the same original assumptions.
In the last years, several cryptographic primitives have been formally proven secure in the bounded memory-leakage, as digital signature, in the random oracle model [7] and in the standard model [106, 58][4] and public-key encryption [127, 7, 87, 29, 58].
In Chapter 4, we show a construction of a *leakage-resilient storage*, a way to securely store data on a device that leaks information, and prove that it is information-theoretically secure in the bounded memory-leakage, where the restriction on the leakage functions consists in allowing the adversary to choose functions that are computationally weak, as for example, functions computable by circuits of small size (see Section 4.3.2 for more details).

3.3.4 Auxiliary Input Model

Dodis et al. [59] relaxed the assumptions on the leakage of the bounded memory-leakage model and introduced the *auxiliary input model*, in which "the leakage is not necessarily bounded in length, but it is (only) assumed to be computationally hard to recover the secret data from the leakage". As this model extends the bounded memory-leakage model, it is assumed that all the system's memory can leak, even the part not accessed during a computation.
This model covers a class of functions larger than the class considered in the bounded memory-leakage model, as the output of the leakage functions could be longer than the secret data size, and can be modeled using functions that are computationally hard to invert, i.e. such that given the leakage (the output of the function) a polynomial time adversary cannot efficiently compute the secret data of the scheme (the corresponding input of the function).

In [59], the authors showed a symmetric-key encryption scheme secure against exponentially hard to invert functions while Dodis et al. [56] and Brakerski and Goldwasser [29] showed public-key cryptographic schemes which are secure against sub-exponentially hard to invert functions.

The minimal assumption on the leakage of the auxiliary input model accurately fits the real world side-channel attacks as usually in practice the most difficult step in the measurements is the extraction of useful information, but

[4] In [106], Katz and Vaikuntanathan showed two constructions: one inefficient and another efficient but only *one-time*. In [58], Dodis et al. showed the first efficient digital signature scheme in the standard model in the bounded memory-leakage setting.

there are some cryptographic primitives that cannot be implemented in such model, as, for example, digital signature (see e.g. [79] for more details).

3.3.5 Continual Memory-Leakage Model

In 2010, Dodis et al. [57] and Brakerski et al. [30] concurrently introduced the new *continual memory-leakage model*, which combines the best features of both the continual leakage model and the bounded memory-leakage model: all the system's memory leaks and the amount of leakage that an attacker can retrieve is bounded only in each time period but it is unbounded during the entire lifetime of the system. As for the continual leakage model, secure schemes in this model have to deal with some methods to periodically update the secret data of the system.

In this model, Boyle et al. [28] constructed a digital signature scheme and several public-key cryptographic primitives have been proposed [57, 30, 115, 114]. Furthermore, Dodis et al. [63] introduced a method to securely store (and refresh) data on multiple physical devices which continually leak information about their entire internal state and showed a public-key encryption scheme based on it. However, all these schemes rely on non-standard assumptions or only allow for small fraction (constant or logarithmic) amount of leakage during the refreshing process of the secret data.

Chapter 4

Leakage-Resilient Storage

In this chapter we introduce a new primitive, a *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. This work has been presented at Security and Cryptography for Networks (SCN) 2010 and is a joint work with Stefan Dziembowski and Daniele Venturi [49].

4.1 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 [112], electromagnetic radiation [134, 82], power consumption [113], or even sound that the device is emitting [144] (see [133, 77] for more examples of such attacks).

4.1.1 Memory Leakages - Previous Work

Over 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 [124]). The main idea is to augment the standard security definition by allowing the adversary to learn the value of a chosen by her *leakage function* Λ on the internal data τ used by the cryptographic scheme. The results in this area can be categorized according to the class of leakage functions Λ that the model covers. Some papers consider very restricted classes (e.g. in [98] 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] allow the adversary to choose any function Λ that is *input-shrinking* (i.e. such that $|\Lambda(\tau)| \ll |\tau|$).

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 [124] (where it is stated as "Axiom 1", page 283), and the other papers that use it are [75, 131]. The schemes of [75, 131] are actually secure even if the total amount of information that leaks is greater than the memory size (this is possible since the memory contents is evolving during the computation). The other approach [5, 127, 106, 59, 56] is to assume that the memory may simply leak information, independently on the computation performed.

It may be questioned if the "only 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 [140]. More recently a class of *cold boot attacks* relying on the data remanence property was presented in [91].

A natural question to ask is whether there exist methods for storing data securely in the memory that may leak information. This is the main subject of this chapter.

4.1.2 Our Contribution

In this chapter 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 $\tau \stackrel{\mathsf{def}}{=} \mathsf{Encode}(m)$, and the *decoding* algorithm Decode is such that we always have $\mathsf{Decode}(\mathsf{Encode}(m)) = m$ (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 $\Lambda_1, \ldots, \Lambda_t$, and learn the values of

$$\Lambda_1(\tau), \ldots, \Lambda_t(\tau).$$

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 Section 4.2 for details). We assume that the Λ_i's are

elements of some fixed set Γ (that will be a parameter in the definition). Obviously, the larger Γ, the stronger is our definition, and we should aim at defining Γ in such a way that it covers all the attacks the adversary can launch in real-life. All the Γ's that we consider in this chapter contain at least the set of functions that read-off the individual bits of τ, hence we need to require that

$$\sum_{i=1}^{t} |\Lambda_i(\tau)| < |\tau|, \tag{4.1}$$

as otherwise the functions Λ_i could be chosen in such a way that $(\Lambda_1(\tau), \ldots, \Lambda_t(\tau)) = \tau$. This is essentially the input-shrinking property that, as discussed above, was already used in the literature.

LRS can also be viewed as a generalization of the All-Or-Nothing Transform (AONT) introduced in [139]. Indeed, the standard definition of AONT requires that it should be hard to reconstruct a message m if not all the bits of its encoding $\mathsf{Encode}(m)$ are known. LRS is defined more generally, with respect to a class Γ of functions. The security definition of LRS requires that it should be hard to reconstruct m even if some values $\Lambda_1(\mathsf{Encode}(m)), \ldots, \Lambda_t(\mathsf{Encode}(m))$ are known (where $\Lambda_1, \ldots, \Lambda_t \in \Gamma$), as long as the total length of $\Lambda_1(\mathsf{Encode}(m)), \ldots, \Lambda_t(\mathsf{Encode}(m))$ is smaller than some parameter λ. Therefore, AONT is a special case of LRS, where the leakage functions are projections of the individual bits.

Obviously, if we go to the extreme and simply allow the adversary to choose *any* (poly-time) functions Λ_i that satisfy (4.1) then there is no hope for any security, since the adversary could always choose Λ_1 in such a way that it simply calculates $\mathsf{Decode}(\tau)$ 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 functions 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 thesis we consider two natural choices of such Γ's and show LRS schemes information-theoretically secure in these settings relying on *deterministic extractors* [153, 23, 44, 45, 36]. In Section 4.3.1 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 on the memory cells that are physically far from each other. In Section 4.3.2 we construct a scheme that is

secure if the cardinality of Γ is restricted (but still it can be exponential in $|\tau|$). This construction implies security in the case when the set Γ consists of functions that are computable by Boolean circuits of a small size. Our construction is an adaptation of the technique already used (in a different context) in [148, 10].

The idea to model the leakages as functions from a small complexity class appeared already in [75], and was recently used in an independent work by Faust et al. [81] (we discuss the relationship between our work and [81] in Section 4.4). We also discuss (in Section 4.5) the connection between the problem of constructing leakage-resilient storage and a theory of compressibility of NP-instances [95].

4.1.3 Preliminaries

In this section we state the lemma that we need in the security proofs of this chapter.

The proof of the following lemma appears in Appendix A.1.1.

Lemma 1. *For every random variables X, Y and an event $\mathcal{E}$ we have*

$$d(X|Y = y \wedge \mathcal{E}) + \boldsymbol{Pr}\left[\overline{\mathcal{E}}\right] \geq d(X|Y). \tag{4.2}$$

The proofs of the following lemmata appear in the full version of [74].

Lemma 2 ([74])**.** *Let A, B be random variables where $A \in \mathcal{A}$. Then $\boldsymbol{Pr}[B = A] \leq d(A|B) + 1/|\mathcal{A}|$.*

Lemma 3 ([74])**.** *Let A, B be two random variables and let ϕ be any function. Then $d(A|B) \geq d(A|\phi(B))$.*

Lemma 4 ([74])**.** *Let A, B be independent random variables and consider a sequence $V_1, \ldots, V_i$ of random variables, where for some function ϕ, $V_i = \phi_i(C_i) = \phi(V_1, \ldots, V_{i-1}, C_i)$, with each C_i being either A or B. Then A and B are independent conditioned on $V_1, \ldots, V_i$, i.e. $I(A; B|V_1, \ldots, V_i) = 0$, where I denotes the Shannon's information.*[1]

The following lemma was proven in [10]:

Lemma 5 ([10])**.** *Let Y be an n-bit random variable with $\mathbf{H}_\infty(Y) \geq k$. Let $\{h_s\}_{s \in \mathcal{S}}$ be a collection of ℓ-wise independent universal 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*

$$u = \frac{\ell}{2}(k - \alpha - 2\log(1/\epsilon) - \log \ell + 2) - \alpha - 2. \tag{4.3}$$

[1] In [74] this lemma is stated in terms of a Markov chain.

We will also use the following standard fact whose proof appears in Appendix A.1.2.

Lemma 6. *Let X be a random variable uniformly distributed over $\{0,1\}^n$, and let W be a random variable that is independent on X. Let $f : \{0,1\}^* \to \{0,1\}^\lambda$. Then for every $k \in \mathbb{N}$ we have*

$$\boldsymbol{Pr}_{y:=f(X,W)}\left[\mathbf{H}_\infty(X|f(X,W)=y) \leq k\right] \leq 2^{\lambda+k-n}. \tag{4.4}$$

4.2 The Definition

Formally, a *leakage-resilient storage* (LRS) scheme is a pair $\Phi \stackrel{\mathsf{def}}{=} (\mathsf{Encode}, \mathsf{Decode})$, where

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

Security of such a scheme is defined as follows. Consider the following game between an adversary $\mathcal{A}$ and an oracle $\mathcal{O}$ (a similar game was used to define security of the Forward-Secure Storage (FSS) [69], the main difference being that (1) FSS had a secret key and (2) the FSS game had just one round)

1. The adversary chooses a pair of messages $m_0, m_1 \in \{0,1\}^\alpha$ and sends them to $\mathcal{O}$.
2. $\mathcal{O}$ chooses a random bit $b \in \{0,1\}$ and sets $\tau \stackrel{\mathsf{def}}{=} \mathsf{Encode}(m_b)$.
3. The following is executed t times, for $i = 1, \ldots, t$:
   (a) $\mathcal{A}$ selects a function $\Lambda_i : \{0,1\}^\beta \to \{0,1\}^{\lambda_i} \in \Gamma$, and sends it to $\mathcal{O}$,
   (b) $\mathcal{O}$ sends $\Lambda_i(\tau)$ to $\mathcal{A}$. We say that $\mathcal{A}$ *retrieved λ_i bits from τ*.
4. The adversary outputs b'. We say that she *won the game* if $b = b'$.

Such an adversary is called a (Γ, λ, t)-adversary if $\sum_{i=1}^t \lambda_i \leq \lambda$.
We say that Φ is $(\Gamma, \lambda, t, \epsilon)$-secure if no (Γ, λ, t)-adversary wins the game with probability greater than $\frac{1}{2} + \epsilon$.[2] We will drop t and say that Φ is $(\Gamma, \lambda, \epsilon)$-*secure*

[2] We say that Φ is *non-adaptively* $(\Gamma, \lambda, t, \epsilon)$-*secure* if the adversary wins the game with probability at most $\frac{1}{2} + \epsilon$, with the restriction that her choice of the functions Λ_i is non-adaptive (i.e. she has to choose all the Λ_i's in advance).

if the parameter t does not matter, i.e. if no (Γ, λ, t)-adversary wins the game with probability greater than $\frac{1}{2} + \epsilon$, for any t.
Unless explicitly stated otherwise, we will assume that the adversary is 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 $\mathcal{A}$ as above, let $\mathsf{view}_{\mathcal{A}}$ denote the vector of values that the adversary $\mathcal{A}$ retrieves from τ, i.e. $\mathsf{view}_{\mathcal{A}} \stackrel{\mathsf{def}}{=} (\Lambda_1(\tau), \ldots, \Lambda_t(\tau))$. Note that $|\mathsf{view}_{\mathcal{A}}| \leq \lambda$.

As argued in the introduction, LRS can be viewed as a generalization of the All-Or-Nothing Transform (AONT) introduced in [139] (see also e.g. [36] for a formal definition). In our framework AONT is simply a $(\Gamma_{\downarrow}, \lambda, \epsilon)$-secure LRS, $\Gamma_{\downarrow}$ being a set of functions Λ_i that leak some bits of the memory, i.e. the functions that have a form $\Lambda_i(\tau_1, \ldots, \tau_\beta) = \tau_i$, where ϵ is equal to 0 if we consider perfectly-secure AONT, or is some negligible value if we consider statistically-secure AONT.

4.2.1 A Weaker Definition

In our schemes, the encoding τ of a string $m \in \{0,1\}^\alpha$ is composed of two parts: (1) the randomness τ_{rand} used to encode the message and (2) the result of the encoding process, i. e. some value $f(\tau_{rand})$ xored with the message m (where f is some publicly-known function). More generally, one can assume that m is a member of some group $(\mathbb{G}, +)$ and f has a type $\{0,1\}^* \to \mathbb{G}$. In this case the encoding of a message m is $(\tau_{rand}, f(\tau_{rand}) + m)$.

For the sake of the security proofs in this chapter, we will consider a game that we call a *weak attack* in which $f(\tau_{rand}) + m$ is hidden from the adversary, and the Λ_i's are applied only to τ_{rand}. The adversary in this game will be called a *weak adversary* and denoted $\mathcal{A}_{weak}$, and we will say that the LRS scheme is *weakly* $(\Gamma, \lambda, t, \epsilon)$*-secure* if $d(f(\tau_{rand})|\mathsf{view}_{\mathcal{A}_{weak}}) \leq \epsilon$, for any $\mathcal{A}_{weak}$, where τ_{rand} is distributed uniformly over $\{0,1\}^n$.
We will say that Γ is *robust* if Γ is closed on the operation of fixing the second part of the input, i.e. if for every $\Lambda \in \Gamma$ and every $z \in \mathbb{G}$ we have that $\Lambda'(x) := \Lambda(x, z)$ is also a member of Γ.

The following lemma shows that a weakly-secure scheme is also secure according to the general definition.

Lemma 7. *Let Γ be an arbitrary robust set as above. For any λ, t and ϵ, if an encoding scheme is* weakly $(\Gamma, \lambda, t, \epsilon)$-secure *then it is also* $(\Gamma, \lambda, t, \epsilon \cdot 2^\alpha)$*-secure.*

Proof. Take some adversary $\mathcal{A}$ that wins the game described in Section 4.2 with

some probability $0.5 + \delta$. We construct a *weak* adversary $\mathcal{A}_{weak}$ such that

$$d(f(\tau_{rand})|Out_{\mathcal{A}_{weak}}) = \delta \cdot 2^{-\alpha}, \tag{4.5}$$

where $Out_{\mathcal{A}_{weak}}$ is some value that is a function of $\mathsf{view}_{\mathcal{A}_{weak}}$ (we will think of it as an output of the adversary $\mathcal{A}_{weak}$ at the end of the execution). Therefore, by Lemma 3, we will have that $d(f(\tau_{rand})|\mathsf{view}_{\mathcal{A}_{weak}}) \geq \delta \cdot 2^{-\alpha}$. After showing this we will be done, by setting $\delta := \epsilon \cdot 2^{\alpha}$.

The adversary $\mathcal{A}_{weak}$ 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, $\mathcal{A}_{weak}$ handles the requests issued by $\mathcal{A}$ in the following way. Recall that each request of $\mathcal{A}$ is a function $\Lambda_i : \{0,1\}^n \times \{0,1\}^{\alpha} \to \{0,1\}^{\lambda_i}$ that should be applied to τ. Each time such a request is issued, the adversary $\mathcal{A}_{weak}$ constructs a request Λ'_i defined for every τ_{rand} as follows:

$$\Lambda'_i(\tau_{rand}) := \Lambda_i(\tau_{rand}, z).$$

By the robustness of Γ we have that if $\Lambda_i \in \Gamma$ then also $\Lambda'_i \in \Gamma$. When the interaction is over and $\mathcal{A}$ outputs b', the adversary $\mathcal{A}_{weak}$ outputs $Out_{\mathcal{A}_{weak}} := z - m_{b'}$.

By Lemma 2 we have

$$\mathbf{Pr}\left[Out_{\mathcal{A}_{weak}} = f(\tau_{rand})\right] \leq 2^{-\alpha} + d(f(\tau_{rand})|Out_{\mathcal{A}_{weak}}). \tag{4.6}$$

Now suppose that for some $i \in \{0,1\}$ the following event $\mathcal{E}_i$ occurred: $z = m_i + f(\tau_{rand})$. In this case $\mathcal{A}_{weak}$ simply simulated the execution of $\mathcal{A}$ against the oracle $\mathcal{O}$ with $b = i$. Since z is chosen uniformly hence $\mathbf{Pr}[\mathcal{E}_0] = \mathbf{Pr}[\mathcal{E}_1] = 2^{-\alpha}$. Therefore the probability that $b' = b(= i)$ is equal to $0.5 + \delta$. Moreover, in this case (i.e. when $\mathcal{E}_0 \cup \mathcal{E}_1$ occurred and $b' = b$) we get that $Out_{\mathcal{A}_{weak}} = m_i + f(\tau_{rand}) - m_{b'}$, and therefore $Out_{\mathcal{A}_{weak}} = f(\tau_{rand})$. Hence we have

$$\begin{aligned} \mathbf{Pr}\left[Out_{\mathcal{A}_{weak}} = f(\tau_{rand})\right] &\geq \mathbf{Pr}\left[b = i \mid \mathcal{E}_0 \cup \mathcal{E}_1\right] \cdot \mathbf{Pr}\left[\mathcal{E}_0 \cup \mathcal{E}_1\right] \\ &= (0.5 + \delta) \cdot 2^{-\alpha+1} \\ &= 2^{-\alpha} + \delta \cdot 2^{-\alpha+1}. \end{aligned}$$

Combining it with (4.6) we get (4.5). □

4.3 The Implementations

In this section we consider two types of leakage functions Γ, and show LRS schemes secure against these Γ's relying on deterministic extractors [153, 23, 44, 45, 36].

In Section 4.3.1 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.
In Section 4.3.2 we construct a scheme that is secure if the cardinality of the set of functions that the adversary can choose is restricted.

4.3.1 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 $\mathcal{M}$ is divided into two parts $\mathcal{M}_0$ and $\mathcal{M}_1$, and each leakage function can be applied to one of the $\mathcal{M}_i$'s separately. In other words, the only restriction is that the adversary cannot choose leakage functions that depend simultaneously on both $\mathcal{M}_0$ and $\mathcal{M}_1$. More precisely, take some $\beta' < \beta$ and let $\tau = (\tau^0, \tau^1)$ where $\tau^0 \stackrel{\mathsf{def}}{=} (\tau_1, \ldots, \tau_{\beta'})$, and $\tau^1 \stackrel{\mathsf{def}}{=} (\tau_{\beta'+1}, \ldots, \tau_\beta)$. Let Γ_2 be the set of all functions Λ_i that "depend only on τ^0 or τ^1", i. e. they have a form

$$\Lambda_i(\tau) = \Lambda'_i(\tau^0),$$

or

$$\Lambda_i(\tau) = \Lambda'_i(\tau^1)$$

(for some Λ'_i). Of course, τ^0 and τ^1 do not need to be stored on two separate memory chips, and it is enough that it is simply impossible for the adversary to compute any function of τ^0 and τ^1 jointly. This may happen for example if τ^0 and τ^1 are stored on one chip, but are physically far from each other. Observe also that the class Γ_2 includes all the functions $\Lambda(\cdot)$ that have communication complexity λ (where λ is the bound on the total amount of bits that the adversary can retrieve). This includes for example the function that computes sum of the bits in (τ^0, τ^1) (as long as λ is at least logarithmic in the length of (τ^0, τ^1)).

The construction

One may observe that this model is very similar to the one of the two-party Intrusion-Resilient Secret Sharing (IRSS) of Dziembowski and Pietrzak (see [74], Section 2.1). The main difference is that the scheme of [74] has an additional property that the decoding function needs to access only small part of the encoded message. Since we do not need this property here, we can use in our construction a standard tool called *two source extractors* [44] (cf. Definition 5 in Section 2.1.3). Let $\mathsf{Ext}_2 : \{0,1\}^n \times \{0,1\}^n \to \{0,1\}^\alpha$ be a two source extractor

and $\Phi_2 \stackrel{\mathsf{def}}{=} (\mathsf{Encode}_2, \mathsf{Decode}_2)$. To encode a message $m \in \{0,1\}^\alpha$, we pick two n-bit strings R_0 and R_1 uniformly at random and we set

$$\tau = \mathsf{Encode}_2(m) = (\tau_{rand}, m^*) \stackrel{\mathsf{def}}{=} (R_0, R_1, \mathsf{Ext}_2(R_0, R_1) \oplus m)$$

and we store R_0 in the first part of the memory ($\mathcal{M}_0$), and $(R_1, \mathsf{Ext}_2(R_0, R_1) \oplus m)$ in the second part ($\mathcal{M}_1$). To decode it suffices to evaluate

$$\mathsf{Decode}_2(R_0, R_1, m^*) \stackrel{\mathsf{def}}{=} m^* \oplus \mathsf{Ext}_2(R_0, R_1).$$

We have the following lemma.

Lemma 8. *If* $\mathsf{Ext}_2 : \{0,1\}^n \times \{0,1\}^n \to \{0,1\}^\alpha$ *is a* (k, k, ϵ)*-two source extractor then* Φ_2 *is* $(\Gamma_2, \lambda, 2^\alpha \cdot \epsilon + 2^{1+\alpha+k+\lambda-n})$*-secure.*

Proof. First we show that Φ_2 is weakly secure against the adversary $\mathcal{A}_{weak}$ outlined in Section 4.2.1 (with $\tau_{rand} = (R_0, R_1)$) and then we use Lemma 7. Let $\mathcal{A}_{weak}$ be an adversary that can apply the leakage functions Λ_i only to τ_{rand} and denote with $\mathsf{view}_{\mathcal{A}_{weak}} = (\Lambda_1(\tau_{rand}), \ldots, \Lambda_t(\tau_{rand}))$ the view of the adversary after t queries to the oracle $\mathcal{O}$. We can now apply Lemma 4 (with $A = R_0$, $B = R_1$, $\phi_i = \Lambda_i$ and $V_i = \Lambda_i(\tau_{rand})$)[3] and conclude that R_0 and R_1 are independent given $\mathsf{view}_{\mathcal{A}_{weak}}$, i.e. $I(R_0; R_1 | \mathsf{view}_{\mathcal{A}_{weak}}) = 0$. Moreover by Lemma 6 we know that for each $i \in \{0,1\}$

$$\mathbf{Pr}_{y:=\mathsf{view}_{\mathcal{A}_{weak}}} [\mathbf{H}_\infty(R_i | \mathsf{view}_{\mathcal{A}_{weak}} = y) \leq k] \leq 2^{k+\lambda-n}.$$

Thus with probability at least $1 - 2^{1+k+\lambda-n}$ it happens that $y = \mathsf{view}_{\mathcal{A}_{weak}}$ is such that for both $i \in \{0,1\}$ we have $\mathbf{H}_\infty(R_i | \mathsf{view}_{\mathcal{A}_{weak}} = y) \geq k$. Let $\mathcal{V}$ denote the corresponding event. We clearly have that

$$d(\mathsf{Ext}_2(R_0, R_1) | \mathsf{view}_{\mathcal{A}_{weak}} = y \wedge \mathcal{V}) \leq \epsilon.$$

Hence, by Lemma 1 we get that $d(\mathsf{Ext}_2(R_0, R_1) | \mathsf{view}_{\mathcal{A}_{weak}}) \leq \epsilon + \mathbf{Pr}\left[\overline{\mathcal{V}}\right] = \epsilon + 2^{1+k+\lambda-n}$. Combining it with Lemma 7 we get that Φ_2 is $(\Gamma_2, \lambda, 2^\alpha \cdot \epsilon + 2^{1+\alpha+k+\lambda-n})$-secure. □

Instantiations

Several constructions [44, 143, 152, 54, 136] of a two-source extractor exist in the literature, and can be used in our scheme. Let $\mathbb{F}$ be a finite field and denote with $\mathsf{Ext}_{\mathsf{Had}} : \mathbb{F}^n \times \mathbb{F}^n \to \mathbb{F}$ the inner product in $\mathbb{F}$, denoted $\mathsf{Ext}_{\mathsf{Had}}(x, y) = \langle x, y \rangle$. As

[3] Clearly ϕ_i depends only on the values V_i that the adversary retrieved in the previous rounds.

shown in [136], for any $\delta > 0$, the function $\mathsf{Ext}_{\mathsf{Had}}$ is a $(k_{\mathsf{Had}}, k_{\mathsf{Had}}, \epsilon_{\mathsf{Had}})$-two source extractor, for $k_{\mathsf{Had}} > (1/2+\delta)n \log |\mathbb{F}|$ and $\epsilon_{\mathsf{Had}} = |\mathbb{F}|^{(n+1)/2} 2^{-k_{\mathsf{Had}}}$ (this generalizes previous results of Chor and Goldreich [44] and Vazirani [152]). Plugging it into the construction described above we get the following LRS scheme $\Phi_{\mathsf{Had}} = (\mathsf{Encode}_{\mathsf{Had}}, \mathsf{Decode}_{\mathsf{Had}})$ for encoding messages $m \in \mathbb{F}$:

$$\begin{array}{rcl} \mathsf{Encode}_{\mathsf{Had}}(m) & = & (r_0, r_1, \langle r_0, r_1 \rangle + m) \\ \mathsf{Decode}_{\mathsf{Had}}(r_0, r_1, m^*) & = & m^* - \langle r_0, r_1 \rangle. \end{array} \tag{4.7}$$

Observe that above, instead of using the xor we used the group operation in $\mathbb{F}$. This is ok, since, as explained in Section 4.2.1, one can transform a weakly-secure scheme into a standard one by using any group operation (not necessarily xor). Using Lemma 8 we get that Φ_{Had} is $(\Gamma_2, \lambda, |\mathbb{F}| \cdot \epsilon_{\mathsf{Had}} + |\mathbb{F}|^{1-n} 2^{1+k_{\mathsf{Had}}+\lambda})$-secure.

4.3.2 Functions that have small descriptions

The second case that we consider is when the only restriction on Γ is that it is a small set of robust 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.

A natural example of such a Γ is a set of *functions computable by Boolean circuits of a fixed size* (see e.g. [155] for an introduction to the complexity of Boolean circuits). Recall that the *size* of a Boolean circuit is the number ρ of its gates. Each gate G can be connected 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 $(\rho-1)(\rho-1) < \rho^2$ choices. Therefore there are at most $(\rho^2)^\rho = \rho^{2\rho}$ circuits of size ρ. Thus the circuits of size ρ can be described using $2\rho \log_2 \rho$ bits.

Several natural functions can be computed by Boolean circuits of a small size (see Section 3 of [155]). For example every symmetric function[4] 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 $(\Gamma_v, \lambda, t, \epsilon)$-secure LRS. Let $\{h_s : \{0,1\}^n \to \{0,1\}^\alpha\}_{s \in \mathcal{S}}$ be a collection of ℓ-wise independent universal hash functions (see Section 2.1.4). The scheme is parameterized by a value $s \in \mathcal{S}$. For any $s \in \mathcal{S}$ let $\Phi_s \stackrel{\mathsf{def}}{=} (\mathsf{Encode}_s, \mathsf{Decode}_s)$, being

$$\mathsf{Encode}_s(m) = (R, h_s(R) \oplus m),$$

[4] 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 Section 3.4 of [155].

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

$$\mathsf{Decode}_s(R, d) = h_s(R) \oplus d.$$

We point out that also the above construction can be interpreted in terms of deterministic extractors. Indeed, as shown in [148] (and in [10]), ℓ-wise independent universal hash functions are, with high probability, deterministic extractors for sources (with some min-entropy) that can be generated by an efficient sampling algorithm or circuit of a small size.[5] Stated in other words, an ℓ-wise independent universal hash function can be viewed as a function $\mathsf{Ext} : \{0,1\}^n \to \{0,1\}^\alpha$ with the following property: for every source $R \in \{0,1\}^n$ with min-entropy k which is samplable by a circuit of a small size, $\mathsf{Ext}(R)$ is close to uniform with high probability. The same construction was also used by Dodis et al. [66] in the context of AONT. Both [148] and [66] consider only the non-adaptive case. Here we show that this scheme is secure in the context of leakage-resilient storage. The following lemma states that with a good probability (over the choice of $s \in \mathcal{S}$) the scheme Φ_s is secure.

Lemma 9. *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 $(\Gamma_v, \lambda, t, 2^\alpha \cdot \epsilon + 2^{\alpha+k+\lambda-n})$-secure, for any $\lambda, k, t, v, \ell, \epsilon$ and ξ such that*

$$\xi = 2^{tv - \frac{\ell}{2}(k-\alpha-2\log(1/\epsilon)-\log \ell+2)+\alpha+2}. \tag{4.8}$$

In the lemma above k is a parameter, that in the proof will correspond to the min-entropy of R conditioned on the view of the adversary. Observe that we have a trade-off between $2^\alpha \cdot \epsilon + 2^{\alpha+k+\lambda-n}$ and ξ (larger k increases the first term, and decreases the second). The proof of this lemma is more involved and we present it in Section 4.3.2. 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 $(\Gamma_v, \lambda, t, 2^{-\mu})$-secure, for any λ, t, v, ℓ, μ and ξ such that*

$$\xi = 2^{tv - \frac{\ell}{2}(n-\lambda-3\mu-4\alpha-\log \ell-1)+\alpha+2}. \tag{4.9}$$

Proof. Set $\epsilon := 2^{-\alpha-\mu-1}$ and let $k := n - \mu - 1 - \alpha - \lambda$. Take Φ_s from Lemma 9. We have that

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

[5] The approach used in [148] is orthogonal to the one used in [44]: in the latter setting, distributions can be arbitrarily complex, but they have to satisfy a strong independence requirement; in the former setting distributions have to be samplable but can involve arbitrary dependencies.

and

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

which is equal to (4.9). □

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 $\mu := 24$ and $\alpha := 128$, and $n := 1024$. If we set $\chi := 10, \omega := 3/25$ then we can allow the adversary to retrieve at most 180 bits by setting $\ell = 278323$. With these settings we get $\xi \leq 4 \cdot 10^{-12}$.
If we consider a non-adaptive scenario, in which the adversary chooses a single leakage function (i.e. $t = 1$) and retrieves at most λ bits[6], then we obtain a better value for ℓ: for $\mu := 24$, $\alpha := 128$, $n := 1024$ and $\chi := 10$, we can allow the adversary to retrieve at most 180 bits by setting $\ell = 2203$. With these settings we get $\xi \leq 2 \cdot 10^{-28}$.

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

$$\mathsf{Encode}'(m) \stackrel{\mathsf{def}}{=} (s, \mathsf{Encode}_s(m)) \quad \text{and} \quad \mathsf{Decode}'(s, x) \stackrel{\mathsf{def}}{=} \mathsf{Decode}_s(x).$$

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

Proof of Lemma 9

We first show that Φ_s is weakly secure. Suppose that the adversary $\mathcal{A}_{weak}$ performs a weak attack against Φ_s. Let R be distributed uniformly over $\{0,1\}^n$. Then we show that for any $\epsilon > 0$ and for at least $1 - \xi$ fraction of $s \in \mathcal{S}$ we have

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

[6]This is equivalent to consider an adversary who chooses $t > 1$ leakage functions in advance, with the same total number of retrieved bits. Note that this scenario is theoretically weaker than the adaptive one but it is useful from a practical point of view.

where ξ is a function of t, v, ℓ, k, α and ϵ as defined in (4.8). Consider some fixed adversary $\mathcal{A}_{weak}$. Let $Good_{\mathcal{A}_{weak}}$ denote the event that $\mathbf{H}_\infty(R|\mathsf{view}_{\mathcal{A}_{weak}} = y) \geq k$, where $y := \mathsf{view}_{\mathcal{A}_{weak}}$. By Lemma 6 we get that $\mathbf{Pr}\left[\overline{Good_{\mathcal{A}_{weak}}}\right] \leq 2^{k+\lambda-n}$. On the other hand, we have

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

Therefore, by Lemma 5 we get that

$$\mathbf{Pr}_s\left[d(h_s(R)|\mathsf{view}_{\mathcal{A}_{weak}} = y, Good_{\mathcal{A}_{weak}}) \geq \epsilon\right] \leq 2^{-u}, \tag{4.10}$$

where $\mathbf{Pr}_s$ means that the probability is taken over the choice of $s \in \mathcal{S}$, and u is defined in (4.3). From Lemma 1 we get that (4.10) implies that

$$\mathbf{Pr}_s\left[d(h_s(R)|\mathsf{view}_{\mathcal{A}_{weak}}) \geq \epsilon + \mathbf{Pr}\left[\overline{Good_{\mathcal{A}_{weak}}}\right]\right] \leq 2^{-u},$$

which implies that

$$\mathbf{Pr}_s\left[d(h_s(R)|\mathsf{view}_{\mathcal{A}_{weak}}) \geq \epsilon'\right] \leq 2^{-u}, \tag{4.11}$$

where $\epsilon' := \epsilon + 2^{k+\lambda-n}$. Of course (4.11) holds just for a fixed adversary and to complete the proof we need to give a bound on the value

$$\max_{\mathcal{A}_{weak}}\left(\mathbf{Pr}_s\left[d(h_s(R)|\mathsf{view}_{\mathcal{A}_{weak}}) \geq \epsilon'\right]\right). \tag{4.12}$$

We will do it by applying a union-bound (over all $\mathcal{A}_{weak}$) to (4.11). However, since that the total number of different adversaries $\mathcal{A}_{weak}$ is doubly-exponential in λ,[7] we cannot do it in a straightforward way. Instead, we first observe that

$$\max_{\mathcal{A}_{weak}} \mathbf{Pr}_s\left[d(h_s(R)|\mathsf{view}_{\mathcal{A}_{weak}}) \geq \epsilon'\right] = \max_{\Lambda_1,\ldots,\Lambda_t} \mathbf{Pr}_s\left[d(h_s(R)|\Lambda_1(R),\ldots,\Lambda_t(R)) \geq \epsilon'\right]. \tag{4.13}$$

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

$$\max_{\mathcal{A}_{weak}}\left(\mathbf{Pr}_s\left[d(h_s(R)|\mathsf{view}_{\mathcal{A}_{weak}}) \geq \epsilon'\right]\right) \leq (2^v)^t \cdot 2^{-u} = 2^{tv-u}. \tag{4.14}$$

This completes the proof, since now using Lemma 7 we are done. □

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

4.4 Comparison with [81]

In an independent work Faust et al. [81] consider a problem of leakage-resilient computation. In their work, that can be viewed as an extension of the "private circuits" paper of [98], they provide a formal definition of a circuit computation that is secure against a class of leakages $\mathcal{L}_{TR}$ (cf. Def. 1 of [81]), and for certain classes $\mathcal{L}_{TR}$, they construct (Theorem 1, [81]) a generic transformation that, given any circuit C transforms it into another circuit C' that is secure against the leakages in $\mathcal{L}_{TR}$.

The main ingredient of their construction is a *linear* encoding scheme that is secure against leakages in some class $\mathcal{L}$. *Linearity* of the encoding means that the decoding function can be expressed as $\mathsf{Decode}(x_1, \ldots, x_\beta) = r_1 x_1 + \cdots + r_\beta x_\beta$, where $r_1, \ldots, r_\beta$ are constants from some field. Their definition of an encoding scheme is very similar to ours: essentially their *p-adaptive* $(\mathcal{L}, \tau, \epsilon)$*-leakage-indistinguishable encoding* is the same as our $(\mathcal{L}, \lambda, t, \epsilon)$-secure LRS scheme. The additional parameter τ, that they use indicates the running time of the adversary (that we do not consider in our work). On the other hand we use the parameter λ, that indicates the total amount of bits retrieved from the encoding, which is absent in [81].

We note that while the work of [81] has an obvious advantage over ours by considering not only secure storage, but also computation, our schemes cover different (and possibly more realistic) classes of leakage functions. In particular, both of the approaches in our work cover trivially the so-called *Hamming weight attacks* [107], which is a measurement frequently applied in practice, where the adversary is allowed to learn a sum of the bits, while the approach of [81] does not cover them.

4.5 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 [75] (footnote 3, page 295).

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 [95]. Informally, an NP-language L is *compressible* if every $x \in \{0,1\}^*$ can

be "compressed" to a much shorter string $\Lambda(x)$ (where Λ is some poly-time function, and $\lambda = |\Lambda(x)| \ll |x|$) such that an infinitely powerful machine M can determine if $x \in L$ just by looking at $\Lambda(x)$. Call this $(PTIME, \infty)$-λ-compressibility. As a natural generalization of this concept, one can consider any $(\mathcal{P}_0, \mathcal{P}_1)$-compressibility (where $\mathcal{P}_0$ and $\mathcal{P}_1$ are some complexity classes): in this setting we would require that $\Lambda \in \mathcal{P}_0$, 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 Section 4.2). Moreover, assume that the adversary is poly-time. Informally speaking what we are looking for, when constructing a Γ-secure LRS $\Phi = (\mathsf{Encode}, \mathsf{Decode})$ is a class of problems that are not $(\Gamma, PTIME)$-λ-compressible on average. More precisely, consider the language L of all valid encodings of some fixed message M. Of course, if this language is $(\Gamma, PTIME)$-λ-compressible with some probability ϵ then Φ cannot be $(\Gamma, \lambda, 1, \epsilon)$-secure (as otherwise the adversary could just choose Λ to be her leakage function). We think that investigate these connections is an interesting future research direction.

Chapter 5

Authenticated Key Exchange Implementation in the Bounded-Retrieval Model

In this chapter we present our implementation of an Authenticated Key Exchange protocol, secure in the bounded-retrieval model, in `C` programming language, using the cryptographic library of the OpenSSL project [128]. The code is available at the author's web site `http://www.dsi.uniroma1.it/~davi/`.

5.1 Introduction

5.1.1 Problem

In 1976, the seminal work of Whitfield Diffie and Martin Hellman "New directions in cryptography" [52] introduced the Diffie-Hellman (DH) key exchange protocol, which allows two parties, with no prior knowledge of each other, to securely derive a shared secret key over an insecure channel, relying on the CDH assumption (see Section 2.2). In the classical DH Key Exchange (see Figure 5.1), the public DH values of the parties are both sent in clear over the channel (i.e. without any encryption) and there is no authentication.

For this reason, the DH Key Exchange suffers of the man-in-the-middle (MITM) attack (see Figure 5.2), in which an adversary, who controls the channel between the parties, can force each party to share a valid secret key with her, while the parties believe to share this key among them.

To perform such attack, an adversary simply needs to intercept the public DH values sent from a party to the other and to send, instead of it, her valid public DH values to the parties. Although MITM attack looks simple, it is very serious, as, after such attack, the adversary can completely control all the

Setup:

- A finite cyclic group $\mathbb{G} = \langle g \rangle$ of order a prime number p.

Protocol:

Client **C**		Server **S**
$priv_{\mathbf{C}} \xleftarrow{R} [p-1]$		$priv_{\mathbf{S}} \xleftarrow{R} [p-1]$
$pub_{\mathbf{C}} := g^{priv_{\mathbf{C}}}$	$\xrightarrow{pub_{\mathbf{C}}}$	
	$\xleftarrow{pub_{\mathbf{S}}}$	$pub_{\mathbf{S}} := g^{priv_{\mathbf{S}}}$
$K_{\mathbf{C}} := pub_{\mathbf{S}}^{priv_{\mathbf{C}}}$		$K_{\mathbf{S}} := pub_{\mathbf{C}}^{priv_{\mathbf{S}}}$

Figure 5.1: Diffie-Hellman Key Exchange Protocol

Setup:

- A finite cyclic group $\mathbb{G} = \langle g \rangle$ of order a prime number p.

Protocol:

Client **C**		Adversary $\mathcal{A}$		Server **S**
$priv_{\mathbf{C}} \xleftarrow{R} [p-1]$		$priv1_{\mathcal{A}} \xleftarrow{R} [p-1]$		$priv_{\mathbf{S}} \xleftarrow{R} [p-1]$
		$priv2_{\mathcal{A}} \xleftarrow{R} [p-1]$		
$pub_{\mathbf{C}} := g^{priv_{\mathbf{C}}}$	$\xrightarrow{pub_{\mathbf{C}}}$			
	$\xleftarrow{pub1_{\mathcal{A}}}$	$pub1_{\mathcal{A}} := g^{priv1_{\mathcal{A}}}$		
		$pub2_{\mathcal{A}} := g^{priv2_{\mathcal{A}}}$	$\xrightarrow{pub2_{\mathcal{A}}}$	
			$\xleftarrow{pub_{\mathbf{S}}}$	$pub_{\mathbf{S}} := g^{priv_{\mathbf{S}}}$
$K_{\mathbf{C}} := pub_{\mathcal{A}}^{priv_{\mathbf{C}}}$		$K1_{\mathcal{A}} := pub_{\mathbf{C}}^{priv1_{\mathcal{A}}}$		
		$K2_{\mathcal{A}} := pub_{\mathbf{S}}^{priv2_{\mathcal{A}}}$		$K_{\mathbf{S}} := pub_{\mathcal{A}}^{priv_{\mathbf{S}}}$

Figure 5.2: Man-in-the-middle attack in the Diffie-Hellman Key Exchange Protocol

communication between the parties. A countermeasures to this weakness of the DH protocol is *authentication*, which means that the parties running the protocol have to authenticate each other (or only in one direction) to check if they are sharing the secret key computed by the protocol with the other party or with some malicious adversary, accepting the key in the former case while aborting it in the latter one.

A simple way to realize an Authenticated Key Exchange (AKE) protocol consists in providing the parties with a password, a weak key, eventually not uniformly distributed, human-memorizable and with low entropy. Using this password, the parties can run a Password-based Authenticated Key Exchange (PAKE) protocol to perform the AKE protocol. In such setting, an adversary could try to use the low entropy of the password to break the security of the protocol, performing a so called dictionary attack, in which the adversary try to determine the password by checking all the possible values. Indeed, as the dictionary, the set of all possible (or most possible) values for the password, has a small size, the adversary can try all these values as password, with non-negligible probability of finding the right one. A dictionary attack can be performed *on-line* or *off-line*. On-line means that the adversary repeatedly runs the protocol and interacts with the other parties to verify if her guess of the password is correct. Hence an on-line attack is not avoidable and to deal with such attack, the system administrators of the machines on which the protocol is run or the implementers of the protocol have to limit the number of attempts that a party can perform to authenticate or block a password after it is used for a certain number of failed attempts to authenticate. In an off-line attack, an adversary try to verify her guess for the password without running the protocol. A PAKE protocol has to be designed in a way such that an adversary can launch on-line dictionary attacks but it should be infeasible for her to verify her guess for the password off-line, without running the protocol.

5.1.2 Contribution

We implemented a two-party AKE protocol in `C` programming language, using the cryptographic library of the OpenSSL project [128]. Following the general paradigm introduced by Cash et al. [42], we based the AKE protocol on a Weak Key Exchange (WKE) protocol and a UC-PAKE protocol. Informally, a WKE protocol provides two parties with passwords (or weak keys) that are individually unpredictable, while a PAKE protocol allows two parties to securely derive a key over an insecure channel using a low-entropy password. In particular, we implemented:

1. in the Bounded Retrieval Model, a WKE protocol to output the password needed by the parties to run the PAKE protocol, similar but simpler than

that one showed in [42], which uses less random bits than our protocol but does not seem to be efficiently implementable in practice;

2. the two-party PAKE protocol, showed secure in the Universal Composability framework in [2], which is more efficient than the Canetti et al.'s UC-PAKE protocol [40] used in [42].

Moreover, as a contribution, we formally adapted the security analysis of the WKE protocol of [42] to our WKE protocol.

In our model, before running the protocols, the two parties, Client and Server, have to agree on a shared large secret key F, a file of n bits (some Gigabytes). They will use this file in the WKE protocol to compute the password for the PAKE protocol. Further, an active adversary $\mathcal{A}$ controls the channel between the parties and, before the parties run WKE and PAKE protocols, she can tamper the parties' machines and perform any efficient computation on the entire internal state of the parties, hence also the shared file F, but, as we work in the Bounded Retrieval Model (see Section 2.3), we assume that the adversary is communication bounded and can retrieve at most λ bits of information from the internal state of Client and Server. In contrast, in our implementation Client can choose how many bits of the shared file F she wants to use in the AKE protocol with Server.

Usually, a PAKE protocol deals with passwords with low entropy but in our setting the passwords output by the WKE protocol, and then used in the PAKE protocol, have high min-entropy from the adversary point of you. The important role of the PAKE protocol is to provide the client-to-server authentication: Client and Server will agree on a shared secret key after the PAKE protocol only if the passwords that they obtain at the end of the WKE protocol are equal. In fact, an adversary, who controls the channel and knows some information about the internal state of the parties and the shared file F, can adaptively correlate the passwords obtained by Client and Server at the end of the WKE protocol, without violate the individual unpredictability of the passwords.

Several security models do not cover the realistic scenario in which the parties run the protocol with different but related passwords. For this reason, we use a UC-PAKE protocol, as in the UC framework (see Section 2.4) there are not assumptions about the distribution of the passwords used by the parties who run the protocol.

Canetti et al. [40] showed that UC-PAKE protocols cannot be achieved in the standard model (see Section 2.2) and proposed a construction in the common reference string (CRS) model (used in the AKE protocol of [42]), in which the parties running the protocol agree, in a setup phase, to a shared short string.

Clearly, it is simple to implement a CRS in the BRM as in this model the parties already share a large file and hence they should simply generate the CRS and store it as a part of their shared string but the construction in [40] is less efficient than other known constructions, developed in the Random Oracle (RO) and Ideal Cipher (IC) models. Therefore, we decide to implement the UC-PAKE construction of Abdalla et al. [2], which is secure under the CDH intractability assumption and relies on the RO and IC models (see Section 2.2) and is more efficient than the protocol of [40]. We deal with the controversial issues to implement a RO and an IC with some concrete cryptographic hash functions and block cipher respectively [38], using the tools provided by the OpenSSL cryptographic library and following the standard choices showed in the RFC 5683 - Password-Authenticated Key Diffie-Hellman Exchange [33].

OpenSSL project. As stated on the OpenSSL project web page [128]: "The OpenSSL Project is a collaborative effort to develop a robust, commercial-grade, full-featured, and Open Source toolkit implementing the Secure Sockets Layer and Transport Layer Security protocols as well as a full-strength general purpose cryptography library". In this thesis, we use only the OpenSSL cryptographic library, the last version `1.0.0e` released on September 2011, which "implements a wide range of cryptographic algorithms used in various Internet standards".

5.1.3 Related works

Privacy amplification and *authenticated key exchange* based on weak keys that have some entropy were studied in [158, 122, 138, 60]. All the protocols showed therein are information-theoretic but they are not local, i.e. the parties need to use the whole secret key in their computation. Remind that, in our setting, Client can choose the number of bits of the shared password that the parties have to use when they run the protocol. Moreover, we will show at the end of Section 5.3.1 that the parties have to access a small portion of the shared file to achieve security against an adversary who can retrieve a very large quantity of bits, even 99% of the shared file.

In the *exposure-resilient* cryptography [36, 66, 103] an adversary have almost complete access to the secret key, but she can only learn the bits of the key (which is short) and she cannot perform arbitrary computation on it. Also the protocols proven secure in this setting are not local. To deal with key exposure, some stateful schemes have been introduced in which the key evolves during the time and an adversary can learn the whole key but the protocols update their secret key at each round. In this setting, *forward secure* schemes [11, 18, 39] provide security for all the rounds occurred before that the adversary tampers the system and learns the secret key but, once this happens, there is no more

security for the following rounds.

Key-insulated cryptography [61, 62] guarantees security for past and future rounds but it needs a master key, used by the parties to update their secret keys at every round and stored by a party that the adversary cannot corrupt. As an extension of such models, (two-party) *intrusion-resilient* schemes [99, 55] consider an adversary who can compromises both the parties but not in the same round.

In 2006, Dziembowski [68] constructed the first secure intrusion-resilient authenticated key exchange, in the BSM (see Section 2.3), using random oracles. Influenced by the work of Boyen et al. [26], in which the authors study the AKE protocol which uses biometric data and introduce "the idea of using low entropy intermediate keys as input of a PAKE protocol", Cash et al. [42] showed a general construction of an intrusion-resilient AKE protocol secure against static adversary and achieved an instantiation of such paradigm without random oracles, in the BRM.

In our implementation, we follow the general paradigm of [42] (implicitly used also in [68]), in which an AKE protocol is realized by performing a WKE protocol, which provides the parties with individually unpredictable passwords, and then a PAKE protocol, which uses such passwords as input and outputs a shared secret key only when both parties run the protocol with the same passwords.

In 1992 Bellovin and Merrit [19] introduced the notion of PAKE but only in 2000 Bellare et al. [12] and Boyko et al. [27] provided the first formalization of models and security of PAKE protocols. Both these works studies the Encrypted Key Exchange (EKE) protocol introduced in [19], a DH key exchange in which the public values of the parties are encrypted before to be sent from a party to the other, using the shared password as the key for the encryption and decryption operations. Bresson et al. [31] showed a formal security proof for a PAKE protocol in the model of [12], which extends the framework of Bellare and Rogaway [13, 15], relying on the RO model and the IC model, while Boyko et al. [27] provided such a proof relying only on the RO model, using the framework of Shoup [145].

There are also some works [86, 104, 105] that provide security proofs for PAKE protocol in the standard model, relying on the DDH assumption (see Section 2.2), but such protocols are not efficiently implementable.

A more recent version of the EKE protocol is the AuthA protocol[1] [16], modeled in [31] by the One-Encryption Key Exchange (OEKE) protocol, in which only the public value of the server is encrypted before to be sent to the client,

[1] Considered for standardization by the IEEE P1363 Standard working group on public-key cryptography [97]

while the first flow from the client to the server is sent in clear, without any encryption. In [32], Bresson et al. showed a construction of the AuthA protocol which does not rely on the IC model but uses a full-domain hash scheme, that is provable in the RO model, to perform the encryption process. Instead of using a block cipher, they encrypt the server's public value by multiplying it by a full-domain hash of the shared password.

In [40] Canetti et al. pointed out that the security models of [12] and [27] do not cover the realistic scenarios in which the parties run the protocol with "different but possibly correlated passwords" and then they introduced a new security model in the UC framework for PAKE protocol, in which there are "no assumptions on the distribution on the passwords" used by the parties who run the protocol. Moreover, in [40], the authors showed a PAKE protocol and proved its security in their new model, relying on standard number-theoretic assumptions, against static adversaries, in the CRS model. They also showed that it is impossible to achieve UC-PAKE in the standard model. Their protocol is based on those ones of Katz et al. [104] and Gennaro and Lindell [83] but it is not as efficient as some protocols proven secure in the models of [12] and [27], e.g. [31, 3, 104, 118]. Abdalla et al. [2] showed that the efficient PAKE protocol studied in [31], and proven secure in the model of [12], is also secure in the UC framework of Canetti et al. [40]. Indeed, in [2] the authors proven the security of the UC-PAKE protocol against adaptive adversaries under the CDH intractability assumption, in the RO model and the IC model.

5.2 Implementation

In this section, we describe the Authenticated Key Exchange protocol that we have implemented in `C` programming language, using the cryptographic library of the OpenSSL project. Our AKE is based on a WKE protocol and a UC-PAKE protocol, we show them in the next subsections.

5.2.1 Weak Key Exchange

In a WKE protocol (see Figure 5.3), two parties, Client and Server, sharing a source, simply choose a set of indexes from the source, send it to each other and compute a password as the concatenation of the bits of the source at the provided (ordered) indexes, first the Client's indexes and then the Server's ones.

The WKE construction of Cash et al. [42] makes use of an averaging sampler [17], which samples a small amount of bits from a large source in such a way that the min-entropy rate of the sampled bits is nearly the same of the large source.

Setup:

- $F \in_R \{0,1\}^n$: a large file (some Gigabytes) shared between Client and Server.
- $\texttt{DirProd} : \Sigma^n \times [n]^t \to \Sigma^t$, a function which takes a source $y = (y[1], \ldots, y[n]) \in \Sigma^n$ and indexes $(i_1, \ldots, i_t) \in [n]^t$ and outputs $(y[i_1], \ldots, y[i_t])$, the elements of the source at the provided indexes.

Protocol:

1. Client and Server choose random $I_C, I_S \in [n]^t$, respectively.

2. Client sends I_C to Server.
   Server receives a value I'_C and computes

   $$Pwd_S = \texttt{DirProd}(F, I'_C)||\texttt{DirProd}(F, I_S).$$

   Server outputs Pwd_S as her password for the PAKE protocol.

3. Server sends I_S to Client.
   Client receives a value I'_S and computes

   $$Pwd_C = \texttt{DirProd}(F, I_C)||\texttt{DirProd}(F, I'_S).$$

   Client outputs Pwd_C as her password for the PAKE protocol.

Figure 5.3: Weak Key Exchange Protocol

In their protocol, Cash et al. uses an explicit construction of averaging sampler showed by Vadhan [151], which uses less random bits than the random choice. However, an implementation of such sampler does not seem to be practical. In fact, it is not clear how to calculate the hidden constants of the Vadhan's result, as it is based on properties of expander graphs and their second eigenvalue for which only asymptotic values are known, and thus it is not possible to calculate the exact amount of random bits needed.

In our construction, to sample some bits from the large shared source, instead of using random walks on expander graphs, we use the random choice and then, to extract the selected bits from the source, we use the simple function Direct Product,[2] which takes as input a source and a set of indexes and outputs the elements of the source at the provided indexes. A result of Alwen et al. (Lemma A.3 in [7]) shows that, choosing appropriate parameters, the bits output by the function Direct Product have good min-entropy. It means that the shared passwords obtained by the parties at the end of the WKE protocol are individually unpredictable for the adversary, they will be equal if the adversary is passive while they could be different and correlated if the adversary performs some action on the messages exchanged between the parties, e.g. the adversary can block I_C or I_S and send a different, chosen by her, I'_C or I'_S.

Our WKE protocol is not forward secure, i.e. it does not provide security for passwords in case of future adversarial break in, as when an adversary compromises one of the party's machine, she could be able to retrieve (without violating the retrieval bound) the bits of the shared file which were used to compute the passwords in a previous execution of the WKE protocol. We show the analysis of our WKE protocol in Section 5.3.1.

Implementation details. To use the Direct Product function, the two parties running the WKE protocol have to choose uniformly at random the set of indexes needed by this function and then they should send to each other their own indexes. As the size of the shared file is some Gigabytes, the value of each index, which represents the position of a bit in the source, could be a large integer, around 10^{10}, thus we have to use the `C` "long long int" type to store it, i.e. 8 bytes. It means that to sample t bits from the n bits source, the parties have to use $64 * t$ random bits and they have to send these bits to each other.

To make our protocol more efficient in terms of random bits and communication complexity, we use a hash function to produce the indexes needed by the Direct Product function. Client and Server have to choose a cryptographic hash function $\mathcal{H}$ (for example SHA-1) and only a small random seed s, e.g. 4 bytes,

[2] Implicitly used also in [42]

and then they calculate respectively their t indexes $idx_1, \ldots, idx_t$ as follows:

$$idx_i = ((\mathcal{H}(i||s) \bmod 2^{64}) \bmod fileSizeBits) + 1, \tag{5.1}$$

i.e. as the decimal value of the least significant 8 bytes (so it can represent an integer between 0 and $2^{64} - 1$) of the digest output by the hash function applied to the number i concatenated to the seed s, modulo *fileSizeBits*, the size in bits of the shared file, plus 1. Hence, $\forall\ i \in [t]$, it holds that $idx_i \in [fileSizeBits]$, i.e. every index represents the position of a bit of the source. In this way, Client and Server uses only few random bits and they do not need to send all the indexes to the other party, as they can simply send their own seed and then they can calculate the indexes of the other party by their own, applying the hash function to the seed that they received.

We remind that the password is generated as the concatenation of the t indexes chosen by the Client and the t indexes chosen by the Server, where the indexes represent the position of bits in the shared file. Once Client and Server have computed the indexes, they both store them in two different arrays `indexes_client` and `indexes_server` and sort these arrays separately, using the `C`'s `qsort` function, which implements the "quick sort" algorithm[3] and then they repeat the following steps $2t$ times, the total number of indexes, until they compute the whole password:

1. read the smallest elements of the `indexes_client` and `indexes_server` arrays among the indexes not used yet[4];
2. calculate in which byte and in which bit of this byte of the shared file is the position corresponding at the selected index;
3. read this byte from the shared file;
4. check the value of the bit in the position corresponding at the selected index;
5. insert the value of the bit extracted from the shared file in the proper position in the password[5].

Therefore, the cost of the WKE protocol for each party is given by the sum of the following operations:

[3] From [116]: "The `qsort` function implementation might not be an in-place sort and might thereby use an extra amount of memory to store the array.".

[4] As the two arrays are sorted, it suffices to compare the first not already read element of both arrays.

[5] Client's indexes are inserted in the first half of the password while Server's ones are inserted in the second half, the first element of the corresponding indexes array in the first position of the proper half, the second element in the second position and so on.

- apply $2t$ times a hash function;
- apply 2 times a "quick sort" algorithm to array of length t;
- check the whole shared file once, to read the needed bytes from it[6].

To avoid to apply the hash function, the parties would need to use another method to compute the indexes, e.g. they could choose or compute their own indexes separately and then send them to each other. But in this way, the communication complexity of the protocol could be very expensive, as there is no limit on the number of indexes the parties can use, and to store each index are needed 64 bits (see the discussion at the beginning of this paragraph), while in our solution the parties need to send to each other 64 bits in total. Hence, our solution is very flexible, as Client can choose to use even all the bits of the shared source to create the password, tolerating in this way the maximum amount of leakage, with constant communication complexity, constant number of random bits used by each party and reading the whole shared file only once, at the cost, of course, to rely on the RO model. However, we would like to note that in our implementation of the AKE protocol we need also a UC-PAKE protocol which already relies on RO, as it is impossible to achieve UC-PAKE in the standard model [40].

To avoid to sort the arrays of indexes, the parties would need to read several times the shared file. This is the most expensive action, as to move to a selected position in a file we need to physically move the head of the disk on which the file is stored, hence we prefer to pay the cost of the sorting algorithm. One solution could be to map the shared file into the memory but the size of this file could be several Gigabytes, which could make this operation inefficient, and one should study also the additional leakage of such operation.

5.2.2 Password-based Authenticated Key Exchange

In our AKE protocol, we implement the two-party UC-PAKE protocol, proven secure in the RO and IC models, under the CDH assumption, by Abdalla et al. in [2]. In this protocol (see Figure 5.4), the parties, Client and Server, share a password and perform an one-flow encrypted Diffie-Hellman key exchange with one-side authentication. To run the protocol, the parties have to agree on some parameters:

1. A block cipher: $(\mathcal{E}_k, \mathcal{D}_k)$.
2. Two hash functions: $\mathcal{H}_0 : \{0,1\}^* \to \{0,1\}^{h_0}$ and $\mathcal{H}_1 : \{0,1\}^* \to \{0,1\}^{h_1}$.

[6] We need to read the file only once as the indexes are ordered and hence the following byte position to read is always greater than or equal to the previous one.

3. A finite cyclic group $\mathbb{G} = \langle g \rangle$ of order a prime number p, where the operation is denoted multiplicatively.[7]

Moreover, Client and Server share a sub-session identifier, which they can compute exchanging nonces, and a password. In our setting, before running the protocol Client chooses the parameters and then sends them to Server.

The protocol performs three flows. Client chooses a private DH random exponent $priv_{\mathbf{C}} \in [p-1]$, calculates the corresponding public DH value $pub_{\mathbf{C}} := g^{priv_{\mathbf{C}}}$ and sends it to Server, together with her name $\mathbf{C}$. Similarly, Server chooses a private DH random exponent $priv_{\mathbf{S}} \in [p-1]$ and calculates the corresponding public DH value $pub_{\mathbf{S}} := g^{priv_{\mathbf{S}}}$ but before to send it to Client, together with her name $\mathbf{S}$, Server encrypts it, using the password pwd and the sub-session identifier $ssid$ to compute a key for the block cipher. Such key can be computed by the Public-Key Cryptography Standards (PKCS) #5 Password-Based Key Derivation Function 2 (PBKDF2) with the Hash-based Message Authentication Code (HMAC) Secure Hash Algorithm (SHA-1) pseudo-random function [101]. Server computes the DH secret key $g^{priv_{\mathbf{C}} priv_{\mathbf{S}}}$ as $K_{\mathbf{S}} := pub_{\mathbf{C}}^{priv_{\mathbf{S}}}$.

To decrypt the public value of Server, Client apply the same PKCS5-PBKDF2-HMAC-SHA-1 function used by Server to pwd and $ssid$ to obtain the same key for the block cipher. Then Client computes the DH secret key $K_{\mathbf{C}} := pub_{\mathbf{S}}^{priv_{\mathbf{C}}}$, the secret key $sk_{\mathbf{C}} := \mathcal{H}_0(ssid||\mathbf{C}||\mathbf{S}||pub_{\mathbf{C}}||pub_{\mathbf{S}}||K_{\mathbf{C}})$ and an authentication tag $Auth := \mathcal{H}_1(ssid||\mathbf{C}||\mathbf{S}||pub_{\mathbf{C}}||pub_{\mathbf{S}}||K_{\mathbf{C}}))$. Client sends this tag to Server.

Then Server computes $\mathcal{H}_1(ssid||\mathbf{C}||\mathbf{S}||pub_{\mathbf{C}}||pub_{\mathbf{S}}||K_{\mathbf{S}}))$, if it is equal to $Auth$ then Server computes the secret key $sk_{\mathbf{S}} := \mathcal{H}_0(ssid||\mathbf{C}||\mathbf{S}||pub_{\mathbf{C}}||pub_{\mathbf{S}}||K_{\mathbf{S}})$ and the protocol succeeds in providing Client and Server with a secret key, otherwise Server produce an error message instead of the secret key. We discuss the analysis of this UC-PAKE protocol in Section 5.3.2.

Implementation details. In our implementation, as group $\mathbb{G}$ we use one of the Modular Exponential (MODP) Diffie-Hellman groups for Internet Key Exchange (IKE) showed in the the RFC2409 [94] and the RFC3526 [110]. By default, the order of the group is a 1024-bits prime number, but when Client runs the protocol, she can set it to 756, 1536, 2048, 3072, 4096, 6144 or 8192 bits. The generator is always the number 2.

OpenSSL provides the type `BIGNUM`, a good tool to deal with integer of large size, and functions which implement the DH Key Exchange, using the type `DH`. The type `BIGNUM` is used to hold a single integer and it dynamically allocates memory to store its data structures, hence it can manipulate numbers of arbitrary size. The type `DH` consists of several `BIGNUM` components, e.g. `g` the generator

[7] We can think of $\mathbb{G}\backslash\{1\}$, the set of the generators of $\mathbb{G}$, as the set $\{g^x | x \in \mathbb{Z}_q^*\}$.

Setup:

- Client and Server share a sub-session identifier *ssid* and a password *pwd*.
- A finite cyclic group $\mathbb{G} = \langle g \rangle$ of order a prime number p, where the operation is denoted multiplicatively.
- A block cipher: $(\mathcal{E}_k, \mathcal{D}_k)$.
- Two hash functions: $\mathcal{H}_0 : \{0,1\}^* \to \{0,1\}^{h_0}$ and $\mathcal{H}_1 : \{0,1\}^* \to \{0,1\}^{h_1}$.
- The Public-Key Cryptography Standards (PKCS) #5 Password-Based Key Derivation Function 2 (PBKDF2) with the Hash-based Message Authentication Code (HMAC) Secure Hash Algorithm (SHA-1) pseudo-random function, called PBKDF2 in what follow for brevity.

Protocol:

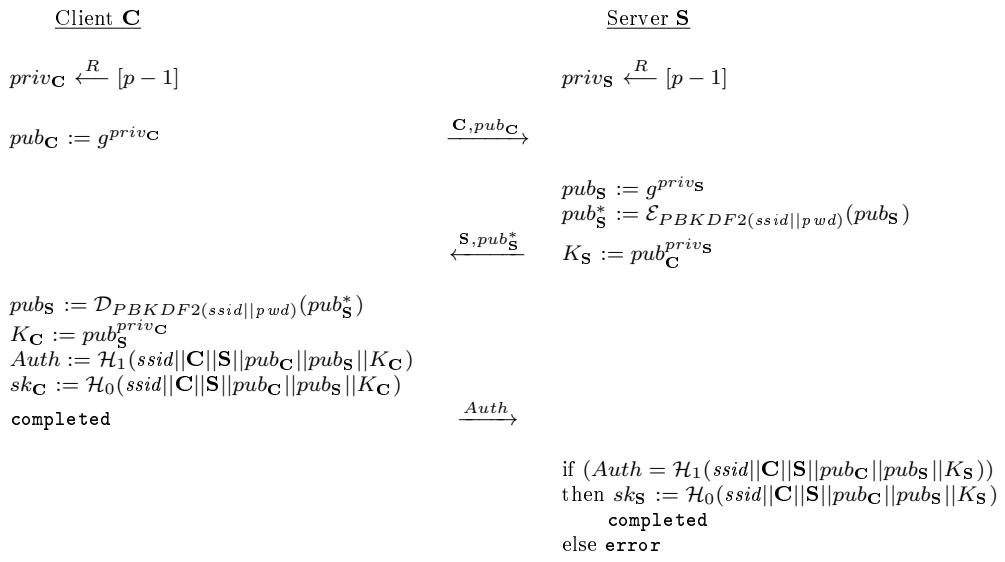


Figure 5.4: Password-Authenticated Key Exchange Protocol

of the the finite cyclic group in which the operation are performed, `p` the prime number order of this group, `priv_key` and `pub_key` the private and public DH values respectively.

In our implementation, we use the OpenSSL functions `DH_generate_key` and `DH_compute_key`. Given a `DH` structure with `p` and `g` set, `DH_generate_key` generates a valid private DH value and the corresponding public DH value. Given the private DH value of a party and the public DH value of the other one, `DH_compute_key` computes the shared secret DH key.

The OpenSSL functions `EVP_DigestInit_ex`, `EVP_DigestUpdate` and `EVP_DigestFinal_ex`[8] allow to hash an arbitrary number of strings of different size. To implement the $\mathcal{H}_0$ and $\mathcal{H}_1$ hash functions, which are applied to the message $z = ssid||\mathbf{C}||\mathbf{S}||pub_{\mathbf{C}}||pub_{\mathbf{S}}||K_j$, with $j \in \{\mathbf{C}, \mathbf{S}\}$ depending on which party computes it, we use a single cryptographic hash function h. Following the standard showed in [33], we calculate

$$\mathcal{H}_i(z) = h(i||length(z)||z||z),$$

for $i \in \{0, 1\}$, i.e. we compute the value of $\mathcal{H}_i(z)$ by simply applying the cryptographic hash function h to the concatenation of the index i, the value $length(z)$, the length in bit of the message z, and two times the message z. By the cryptographic hash function properties, even a small change in the input message drastically change the corresponding output, by the avalanche effect.

By default, we use SHA-1 as cryptographic hash function but it is possible to use also other functions, e.g. SHA-256 or MD5. In our implementation, Client can choose them as command line argument when she runs the protocol.

To perform encryption and decryption on the Server's public DH value, we use the OpenSSL `EVP_EncryptInit_ex`, `EVP_EncryptUpdate`, `EVP_EncryptFinal_ex` and `EVP_DecryptInit_ex`, `EVP_DecryptUpdate`, `EVP_DecryptFinal_ex` functions respectively[9].

To compute the key and the initial vector (IV) needed by these functions together with a cipher, we use the OpenSSL `PKCS5_PBKDF2_HMAC_SHA1` function, which implements the PKCS5-PBKDF2-HMAC-SHA-1 pseudo-random function [101]. It takes as input a secret string *pass*, a random string *salt* and a number of iteration *ic* and outputs key and IV for a block cipher. It is a deterministic function, so it produces same output when applied to same input, and therefore Client and Server can use it to obtain the same key and IV.

To the best of our knowledge, there is not yet an official documentation about this function. The input *salt* and *ic* increase the cost of producing the output

[8] To use these functions, one needs to initialize and then to clean up a message digest context `EVP_MD_CTX`.

[9] As for hashing, to use these functions, one needs to initialize and then clean up a cipher context `EVP_CIPHER_CTX`.

but also the cost of the difficulty of an attack. The random string *salt* is designed to avoid some attacks on password based encryption, e.g. if the same password is used multiple times or if password has low entropy, it should be a random string of at least 64 bits. The iteration counter *ic* makes the calculation of the key longer, to slow down brute force attacks, it should be a number greater than 1000.

In our setting, we do not use any *salt* as Client and Server cannot agree on the same one, keeping it secret to the adversary, without increasing the complexity of the protocol, and we set *ic* to be equal to 10000, it will increase the cost of brute-force attack significantly, without a noticeable impact for legitimate user. As secret string *pass*, we use the concatenation of the sub-session identifier *ssid* and the shared password *pwd*.

By default, as cipher we use the Advanced Encryption Standard (AES) with a key of 256 bits in the Cipher-block chaining (CBC) mode, but when Client runs the protocol, she can choose any other cipher, key length and mode of operation supported by OpenSSL.

5.2.3 Authenticated Key Exchange from Weak Key Exchange and Password-based Authenticated Key Exchange

Following the theoretical work of [42], in our implementation of the AKE protocol (see Figure 5.5), two parties, sharing a large file, run a WKE protocol to derive a shared password, with high min-entropy from the adversary point of view, and then they use this password in a UC-PAKE protocol to compute a shared secret key and to provide client-to-server authentication.

Before running the WKE protocol, Client selects the parameters of the protocols and sends them to Server, in one round. Our WKE construction, based on that one in [42], is a two round protocol and the PAKE protocol that we implemented from [2] is a three round protocol. Two more rounds are needed to exchange nonce, for a total of 8 rounds. However, we have moved the first round and the exchange of nonce in the WKE protocol, using three rounds less, for a total of 5 rounds. The AKE protocol construction showed in [42], uses 9 rounds but achieves two-side authentication. In our implementation, only client-to-server authentication is performed, as only Server receives an error message in case the parties do not meet the condition to share the same key. To achieve mutual authentication, we have to add just one round in the PAKE protocol but then we should carefully check if the security proofs given in [2] and [31] still hold. However, as stated in [2]: "client-authentication is usually enough in most cases and often results in more efficient protocols". We show the analysis of our

Setup:

- Client and Server share a large file.
- WKE: A Weak Key Exchange protocol.
- PAKE: A Universally Composable Password-based Authenticated Key Exchange protocol.

Protocol:

1. Client chooses the parameters of the protocols and sends them to Server.
2. Client and Server run the WKE protocol, obtaining the passwords Pwd_C and Pwd_S, respectively.
3. Client and Server exchange nonces, which are concatenated to form a sub-session identifier *ssid*.
4. Client and Server run the PAKE protocol with sub-session identifier *ssid*, using the passwords Pwd_C and Pwd_S as their inputs, respectively. Client obtains secret keys sk_C and Server obtains sk_S secret key or an error message.

Figure 5.5: Authenticated Key Exchange Protocol

AKE protocol in Section 5.3.3.

Implementation details. Following the OpenSSL standard, we set up a context object (`AKE_CTX`) that contains all the structures and variables that we need. In this way, we can initialize them all at once, hiding this process to a final user. To use block ciphers and hash functions in our code, both Client and Server need to load the internal OpenSSL table of digest algorithms and ciphers and then to clean it at the end of the process, using the OpenSSL `OpenSSL_add_all_algorithms` and `EVP_cleanup` functions respectively.

Client: Once Client has initialized the `AKE_CTX` context, she chooses the parameters of the protocols[10]:

- security parameter,

- leakage parameters,

- (number of bits of) group order,

- hash functions,

- cipher,

connects herself to Server by a stream socket[11] [157] and sends the parameters to Server.
After running WKE and PAKE protocols, Client needs to erase data and to free the memory that she has allocated in the `AKE_CTX` context.

Server: After the initialization of the `AKE_CTX` context, Server creates a stream socket and begins an infinite loop, waiting for connection from clients who wants to run an AKE protocol to derive a secret shared key. To handle multiple clients, Server creates a new process for each client who connects to the stream socket, using the `fork` function. Each new process is executed independently, has its own copy of the `AKE_CTX` context and exits once the secret shared key has been computed.

Once Server accepts a connection from a client, she receives the parameters sent by that client and checks the consistency of security and leakage parameters,

[10] Client inserts the parameters as command line arguments but she chooses leakage parameters interactively, once the program run, as they depend on security parameters and the size of the shared file.

[11] Server has already created it and waits for connection from clients.

to avoid malicious requests[12]. If the parameters pass the test, Server runs WKE and PAKE protocols and then, before to exit the process, she cleans the `AKE_CTX` context copy of that process, erasing the data and freeing the allocated memory.

The shared large file. In our setting, instead of remember an human-memorizable password, the parties have to store a large file. A different file is generated by a server in a setup phase for each client who wants to run an AKE protocol with that server. As we work in the BRM, the file size could be several Gigabytes but today the storage devices are pretty cheap and fast and it should not be a problem for a user to buy one of them suitable for her purpose. However, the server could provide the client with a USB pen drive which contains the file, when the client has to run the AKE protocol she simply needs to plug the USB pen drive into her machine. Moreover, in this case the adversary can retrieve some information about the file only when the USB pen drive is plugged into the user's machine. Of course, then the problem is moved to securely store the USB pen drive, preventing accidental lost or theft, that can occur more frequently with a USB pen drive than with a whole machine.

The server could use a pseudo-random generator (PRG)[13] and a short random seed to generate the long file but the process should be taken in a leakage-free contest and the resulting file will not be uniformly random, in contrast with our assumptions.

An important improvement for the AKE protocol would be the possibility to refresh the shared file as, even if the file size is large, it could happen that an adversary was able to compromise the user's machine several times between different running of the AKE protocol and to retrieve all the secret file or just enough to break the security of the scheme.

In [42], Cash et al. suggested a way for the parties to refresh the secret file by simply running the AKE protocol to share a short secret key, expanding it using a PRG and then xoring the long string obtained with the file and erasing the short secret key. All the process takes some time and the steps after the AKE protocol have to be performed only if the AKE protocol succeeds and in a leakage-free contest. If a PRG based on stream cipher is used, everything can be done without any needed of extra disk space. We remark that the problem to securely store data on an hardware device that may leak information is the subject of Chapter 4 of this thesis.

[12] In this way, we block an adversary who tries to convince a server to accept to run an AKE protocol with insecure parameters, i.e. parameters that could allow the adversary to retrieve some information about the shared password, while running the AKE protocol.

[13] Informally, a PRG is a deterministic function that expands a short random seed into a long pseudo-random string that cannot be distinguished efficiently from a truly random string.

Bandwith. In [51] Di Crescenzo et al. studied password protocols in the BRM and suggested to limit the server outgoing bandwidth to improve the resilience to an active adversary. Also in our setting we can think about limiting the bandwidth of the channel, as Client and Server need to exchange only a small quantity of bytes.

For example, in the default setting the parties exchange at most 778 bytes[14]:

- 330 bytes (at most) to send the parameters from Client to Server;
- 16 bytes in the WKE protocol:
  - 8 bytes from Client to Server to exchange *seed* and *nonce*;
  - 8 bytes from Server to Client to exchange *seed* and *nonce*.
- 432 bytes (at most) in the PAKE protocol:
  - 200 bytes from Client to Server to exchange *client's name length*, *client's name*, *client's public DH value length* and *client's public DH value*;
  - 216 bytes from Server to Client to exchange *server's name length*, *server's name*, *server's encrypted public DH value length* and *server's encrypted public DH value*;
  - 16 bytes from Client to Server to exchange *authentication tag*.

Therefore, even if we limit the bandwidth between Client and Server at 1024 bytes/sec, it does not affect the efficiency of the AKE protocol but, in a setting in which we tolerate a large amount of leakage, e.g. 99% of the file size[15], an adversary needs a large amount of time to complete the retrieval process:

- if file size is 1 GB, the adversary needs around 12 days;
- if file size is 10 GB, the adversary needs around 4 months;
- if file size is 50 GB, the adversary needs more than 1 year and half.

[14]The exact number of bytes depends by the length of the name of: the intput file, the hash functions and the cipher.

[15]We will see in the next section that parties running the AKE protocol can achieve such percentage of leakage tolerance by still accessing a small portion of the shared file, hence without affecting the efficiency of the protocol.

5.3 Analysis

5.3.1 Weak Key Exchange

In this section we adapt the security analysis of the WKE protocol of [42] to our WKE protocol (see Section 5.2.1). We do not introduce a formal security model as it would be equal to that one introduced in [42] but an interested reader can easily check that the result obtained in this section make our WKE protocol straightforwardly secure in that model.

In our construction of the WKE protocol, we use the Direct Product function $\texttt{DirProd}: \Sigma^n \times [n]^t \to \Sigma^t$, which takes as input a source $y = (y[1], \ldots, y[n]) \in \Sigma^n$ and a set of indexes $(i_1, \ldots, i_t) \in [n]^t$ and outputs $(y[i_1], \ldots, y[i_t])$, the elements of the source at the provided indexes.
In this section, we show that, choosing appropriate parameters, the elements output by the Direct Product function have a good min-entropy.

We need the following results:

Lemma 10 (Lemma 1 from [42]). *Let X and Y be any two (correlated) random variables. Suppose that $\mathbf{H}_\infty(X) \geq n$, and Y takes values in $\{0,1\}^r$. Then for every $\epsilon > 0$, with probability at least $1 - \epsilon$ over $y \leftarrow Y$, $\mathbf{H}_\infty(X|Y = y) \geq (n - r - \log(1/\epsilon))$.*

Lemma 11 (Lemma A.3 from [7]). *Let Σ be an alphabet of size q. Let X be a random variable over Σ^n and $\mathcal{E}_1$ be an arbitrary experiment. Define $\mathcal{E}_2$ to be the experiment where, at the conclusion of $\mathcal{E}_1$, a uniformly random $R \in [n]^t$ is chosen and given to the predictor. Then for any $c > 0$, if $\tilde{\mathbf{H}}_\infty(X|\mathcal{E}_1) \geq \frac{2cn}{t}(\log_2(q) + \log_2(n)) + 3c + 5$ then $\tilde{\mathbf{H}}_\infty(\texttt{DirProd}(X, R)|\mathcal{E}_2) > c$.*

Lemma 12 (Lemma 2.2 from [64]). *Let A, B be random variables. For any $\delta > 0$, $\mathbf{H}_\infty(A|B = b)$ is at least $\tilde{\mathbf{H}}_\infty(A|B) - \log_2(1/\delta)$ with probability at least $1 - \delta$ over the choice of b.*

Our result can be stated as follows:

Lemma 13. *Let $\texttt{DirProd}: \Sigma^n \times [n]^t \to \Sigma^t$ be as above and let F be a uniformly random string in $\{0,1\}^n$. Then, for every $0 < \beta, \epsilon < 1$, $t < n$, $\lambda < n$ and $k > 0$,*

$$\text{if } n - \lambda - \beta \geq \frac{2kn}{t}(1 + \log_2(n)) + 3k + 5$$

$$\text{then } \mathbf{H}_\infty(\texttt{DirProd}(F, I_C)|(I_C, view_{\mathcal{A}}) = (\mathcal{I}, \mathcal{V})) > k - \epsilon,$$

with probability $1 - 2^{-\epsilon}$, over the choice of $\mathcal{I} \in [n]^t$ and $\mathcal{V} \in \{0,1\}^\lambda$.

Proof. We assume that the file F is generated uniformly at random by Server and given to Client in a setup phase. As F is uniformly random it has min-entropy equals to its number of bits, i.e. $\mathbf{H}_\infty(F) = n$.

Before the beginning of the WKE protocol, the adversary can compromise the machines of Client and Server, perform any efficient computation on their internal states and the shared file F but, as we work in the BRM, she can retrieve at most λ bits of information.
We can think of this λ bits as the output of a circuit chosen by the adversary and applied to the entire internal state of Client and Server, so also I_C, I_S and F (cf. Figure 5.3). Our setting covers also the adaptive case, in which the adversary chooses adaptively several circuits and applies them sequentially once she intruded into the machines, the only restriction is that the total amount of bits output by the circuits and then retrieved by the adversary is at most λ. Hence, by Lemma 10, with probability $1 - 2^{-\beta}$ taken over the distribution of the adversary view $view_{\mathcal{A}} \leftarrow \{0,1\}^\lambda$, i.e. the vector of the λ bits retrieved by the adversary from Client and Server, it holds that $\mathbf{H}_\infty(F|view_{\mathcal{A}}) \geq n - \lambda - \beta$, for any β. It means that from the adversary's point of view the min-entropy of F decreases at most of $\lambda + \beta$, after the adversary learns $view_{\mathcal{A}}$, the vector of the retrieved λ bits, with probability $1 - 2^{-\beta}$.

In our setting we can think of the experiment $\mathcal{E}_1$ of Lemma 11 as the random variable $view_{\mathcal{A}}$, hence applying Lemma 11, we obtain that for any $k > 0$,

$$\text{if } \tilde{\mathbf{H}}_\infty(F|view_{\mathcal{A}}) \geq \frac{2kn}{t}(1 + \log_2(n)) + 3k + 5$$
$$\text{then } \tilde{\mathbf{H}}_\infty(\texttt{DirProd}(F, I)|(I, view_{\mathcal{A}})) > k, \tag{5.2}$$

where $I \in_R [n]^t$.[16]
As F is uniformly random and $view_{\mathcal{A}}$ is a vector of λ bits, we have that, from the adversary's point of view, $\tilde{\mathbf{H}}_\infty(F|view_{\mathcal{A}}) = \mathbf{H}_\infty(F|view_{\mathcal{A}}) \geq n - \lambda - \beta$. Then, fixing a value for k, equation (5.2) implies that

$$\text{if } n - \lambda - \beta \geq \frac{2kn}{t}(1 + \log_2(n)) + 3k + 5$$
$$\text{then } \tilde{\mathbf{H}}_\infty(\texttt{DirProd}(F, I_C)|(I_C, view_{\mathcal{A}})) > k. \tag{5.3}$$

Now, by fixing $\delta = 2^{-\epsilon}$ and applying Lemma 12, we obtain that

$$\mathbf{H}_\infty(\texttt{DirProd}(F, I_C)|(I_C, view_{\mathcal{A}}) = (\mathcal{I}, \mathcal{V})) \geq \tilde{\mathbf{H}}_\infty(\texttt{DirProd}(F, I_C)|(I_C, view_{\mathcal{A}})) - \epsilon \tag{5.4}$$

[16]Note that the adversary can learn the parties' seeds observing the channel and then she can simply calculate the corresponding indexes by applying the formula in equation (5.1). She can easily retrieve the length of the shared file when she compromises the parties' machines before they run the protocol.

with probability $1 - 2^{-\epsilon}$, over the choice of $\mathcal{I} \in [n]^t$ and $\mathcal{V} \in \{0,1\}^\lambda$.
Then, combining (5.3) and (5.4), we have that

$$\text{if } n - \lambda - \beta \geq \frac{2kn}{t}(1 + \log_2(n)) + 3k + 5$$

$$\text{then } \mathbf{H}_\infty(\texttt{DirProd}(F, I_C)|(I_C, view_\mathcal{A}) = (\mathcal{I}, \mathcal{V})) > k - \epsilon, \tag{5.5}$$

with probability $1 - 2^{-\epsilon}$, over the choice of $\mathcal{I} \in [n]^t$ and $\mathcal{V} \in \{0,1\}^\lambda$. □

In our implementation, we fix $k = 2\epsilon$, as in this way ϵ represents the lower bound of the min-entropy in equation (5.5) and it is the only security parameter of the protocol. Hence, we finally obtain:

Corollary 2. *Let* $\texttt{DirProd} : \Sigma^n \times [n]^t \to \Sigma^t$ *be as above and let* F *be a uniformly random string in* $\{0,1\}^n$. *Then, for every* $0 < \beta, \epsilon < 1$, $t < n$ *and* $\lambda < n$,

$$\textit{if } n - \lambda - \beta \geq \frac{4\epsilon n}{t}(1 + \log_2(n)) + 6\epsilon + 5 \tag{5.6}$$

$$\textit{then } \mathbf{H}_\infty(\texttt{DirProd}(F, I_C)|(I_C, view_\mathcal{A}) = (\mathcal{I}, \mathcal{V})) > \epsilon, \tag{5.7}$$

with probability $1 - 2^{-\epsilon}$, *over the choice of* $\mathcal{I} \in [n]^t$ *and* $\mathcal{V} \in \{0,1\}^\lambda$.

Therefore, even conditioned on the adversary's view (the λ bits retrieved from the internal state of the parties by the adversary) and the set of t indexes accessed in the file F by Client, if equation (5.6) holds, Pwd_C has min-entropy greater than ϵ (with probability $1 - 2^{-\epsilon}$), for any I'_S chosen by the adversary (cf. Figure 5.3). A similar argument applies to Pwd_S, therefore we conclude that, choosing appropriate parameters, the elements output by the Direct Product function have a good min-entropy from an adversary point of view, even after retrieving λ bit of information from the internal state of Client and Server (before of the execution of the WKE protocol). It means that the shared passwords obtained by the parties at the end of the WKE protocol are individually unpredictable for the adversary, they will be equal if the adversary is passive while they could be different and correlated if the adversary performs some action on the messages exchanged between the parties.

Concrete parameters and experimental evaluation

Fixing ϵ and β, we can choose the level of security that we want to achieve and then we can study the relation between λ, the number of bits retrieved by the adversary from the file F, and t, the number of bits of the file F that Client and Server use to compute their password. We simply set $\beta = 80$ to have that with very large probability $1 - 2^{-80}$, it holds that $\mathbf{H}_\infty(F|view_\mathcal{A}) \geq n - \lambda - 80$. This

choice does not affect the efficiency of the protocol as n is much greater than 80. In our implementation, by default, we set $\epsilon = 30$, obtaining that with probability $1 - 2^{-30}$ the min-entropy of Pwd_C and Pwd_S is greater than 30 (after the WKE protocol terminates and even if the adversary has retrieved λ bits from Client and Server and knows I_C and I_S), if

$$n - \lambda \geq \frac{120n}{t}(1 + \log_2(n)) + 265. \tag{5.8}$$

With this ϵ, setting $\lambda = 0$ in equation (5.8), we get that the minimum number of bits that Client and Server need to access in the file to allow leakage is

$$t_{min} = \left\lceil \frac{120n(1 + \log_2(n))}{n - 265} \right\rceil,$$

while setting $t = n$ we get that the maximum leakage tolerated[17] in bits is

$$\lambda_{max} = \lfloor n - 120 \log_2 n - 385 \rfloor.$$

However, Client can choose to set ϵ to any value greater than 30 when she runs the protocol. We note that the value k, which represents a lower bound of the average conditional min-entropy $\tilde{\mathbf{H}}_\infty(\mathtt{DirProd}(F, I)|(I, view_{\mathcal{A}}))$, is fixed to be equal to 2ϵ, as in this way equation (5.7) holds and ϵ is the only security parameter of the protocol (cf. Corollary 2).

Table 5.1 describes

- the values of t computed by applying equation 5.8
- the running time of the AKE protocol evaluated experimentally on an Intel(R) Core(TM) i5-2410M CPU @ 2.30GHz, with 4GB of RAM, under the 64-bits version of Ubuntu 11.04

in the standard setting $\epsilon = 30$ and in an higher level of security setting in which $\epsilon = 80$ with leakage λ equals to 99% of the size of the shared file.

From the analysis of Table 5.1, we can conclude that in our AKE protocol implementation it is possible to achieve a very high leakage resilience percentage still accessing a small portion of the shared file, even for $\epsilon = 80$, as Client and Server need to use only less than 310 KB[18] over the several Gigabytes of the shared file F to securely run the AKE protocol against an adversary who can retrieve even up to 99% of the file, as also showed in Figure 5.6.

In Figure 5.7, we show the experimental evaluation of the running time of

[17] Maximum leakage tolerated means that if the adversary retrieves more than this quantity of bits, in total both from Client and Server, then equation (5.6) does not hold and hence no more security is provided, as we cannot say anything about the min-entropy of Pwd_C and Pwd_S, even if Client and Server use all the bits of the shared file to compute their passwords.

[18] Client and Server access (at most) $2t$ bits in the file: t indexes chosen by Client and t indexes chosen by Server. Some indexes (even all of them) might be equal.

	$\epsilon = 30$ $\lambda = 99\%$ of n			$\epsilon = 80$ $\lambda = 99\%$ of n		
n (GB)	**t** (KB)	**WKE** (sec)	**PAKE** (sec)	**t** (KB)	**WKE** (sec)	**PAKE** (sec)
1	49,8	42,36	16,71	132,81	43,41	44,33
5	53,21	231,65	17,96	141,88	236,06	47,36
10	54,67	495,2	18,52	145,79	502,2	48,7
15	55,53	724,21	18,67	148,07	752,69	49,41
20	56,14	767,18	18,89	149,7	888,12	50,11
25	56,61	931,86	19,18	150,95	1067,24	50,46
30	56,99	1113,15	19,2	151,98	1286,26	50,84
35	57,32	1257,47	19,24	152,85	1414,54	51,08
40	57,6	1443,07	19,38	153,6	1604,41	51,35
45	57,85	1693,53	19,47	154,27	1795,34	51,37
50	58,07	1819,77	19,69	154,86	1881,3	51,88

Table 5.1: Values of t and running time of the WKE and PAKE protocols, in the standard setting $\epsilon = 30$ and in an higher level of security setting $\epsilon = 80$, for file size $1GB \leq n \leq 50GB$ and leakage $\lambda = 99\%$ of n.

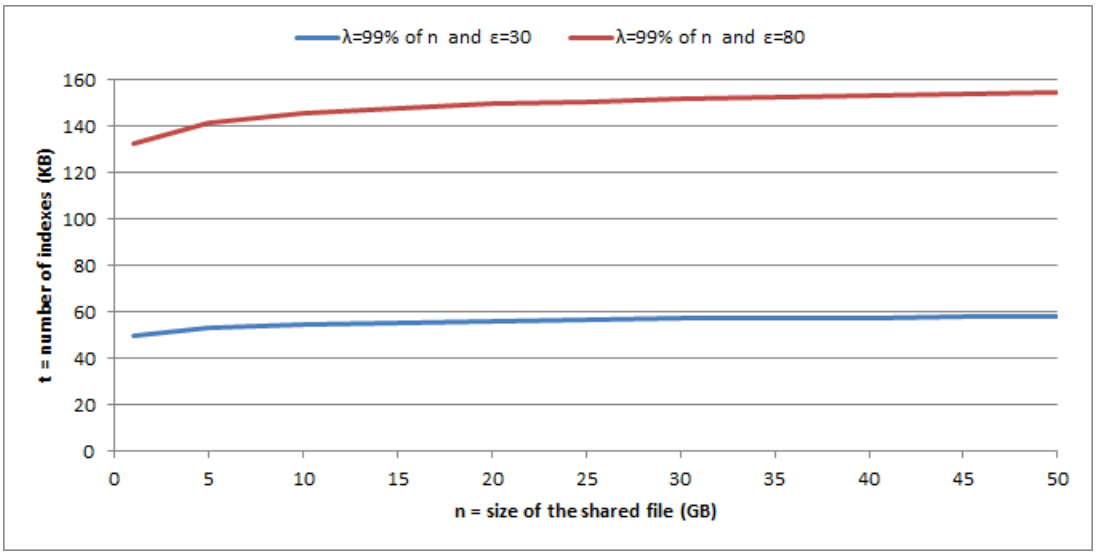
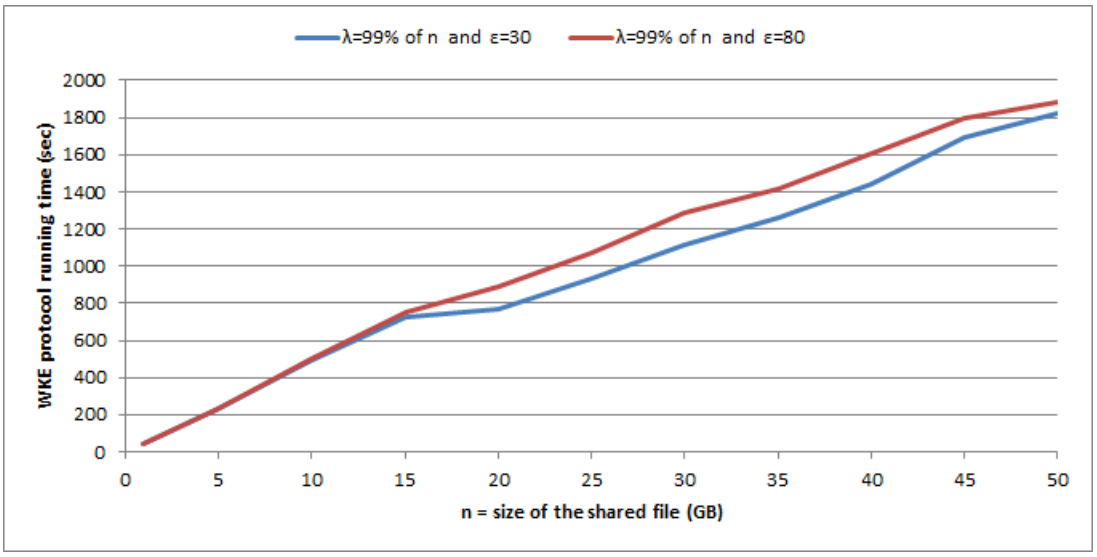
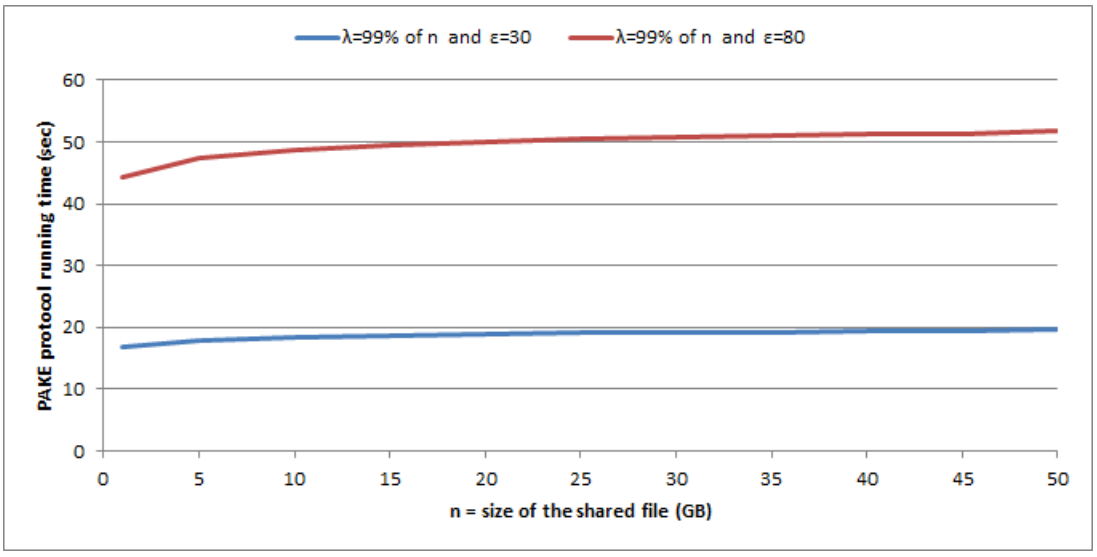


Figure 5.6: Comparison between the values of t in the standard setting $\epsilon = 30$ and in an higher level of security setting $\epsilon = 80$, for file size $1GB \leq n \leq 50GB$ and leakage $\lambda = 99\%$ of n.



(a) WKE protocol running time



(b) PAKE protocol running time

Figure 5.7: Experimental evaluation of the running time of the WKE (Figure 5.7(a)) and PAKE (Figure 5.7(b))protocols. Comparison between the standard setting $\epsilon = 30$ and an higher level of security setting $\epsilon = 80$, for file size $1GB \leq n \leq 50GB$ and leakage $\lambda = 99\%$ of n.

the WKE and PAKE protocols, comparing the standard setting $\epsilon = 30$ with an higher level of security setting $\epsilon = 80$, for file size $1GB \leq n \leq 50GB$ and leakage $\lambda = 99\%$ of n. From this comparison, we note that the request of an higher level of security does not affect significantly the efficiency of our AKE protocol. We would like to note that, despite the fact that the total amount of time needed to run the protocol looks expensive (around 30 minutes for large n), a very large amount of time is used to perform the WKE protocol, while the PAKE protocol is drastically faster (it takes only from few seconds to less than a minute). The WKE protocol is expensive, in terms of time, because it performs the accesses to the disk, the most expensive operation of our protocol. Therefore, to improve the efficiency of the AKE protocol it is sufficient to optimize this operation, e.g. using hardware which allows faster disk access time or mapping the whole shared file in the memory (but then one should study also the additional leakage of such operation). However, in section 5.3.3 we show that Client and Server needs to compute the shared keys only once, as the keys remain secure even if, after the end of the AKE protocol, the adversary retrieves all the shared file.

5.3.2 Password-based Authenticated Key Exchange

In our implementation of the AKE protocol, we use the UC-PAKE protocol of Abdalla et al. showed in [2]. As noted in [42], we need to use a UC-PAKE protocol instead of simply a standard PAKE protocol, as the passwords output by the WKE protocol and then used by the PAKE protocol, in our implementation of the AKE protocol, could be arbitrary correlated and, unlike more traditional models, in the UC framework there are "no assumptions on the distribution on the passwords" used by the parties.

In [2], the authors presented an "ideal functionality" $\mathcal{F}_{PAKE}^{CAuth}$ (see Figure 5.8) for PAKE protocol with client authentication (which extends that one in [40]) and then they proved that the protocol showed in Section 5.2.2 securely realizes such "ideal functionality"[19], against adaptive adversaries.

To provide a random oracle and an ideal cipher to the parties, in their proof Abdalla et al. use the ideal functionalities $\mathcal{F}_{RO}$ (already introduced in [96], see Figure 5.9) and $\mathcal{F}_{IC}$ (see Figure 5.10). Clearly, the random oracle model and the ideal model UC-emulates $\mathcal{F}_{RO}$ and $\mathcal{F}_{IC}$ respectively.

The functionality $\mathcal{F}_{PAKE}^{CAuth}$ provides that:

- the parties receive the same uniformly distributed random key if they share the same password and the adversary does not corrupt one of them and

[19] Technically, they prove that the protocol securely realizes the multi-session extension of the functionality, in the joint state version of the UC framework (see Section 2.4).

$\mathcal{F}_{PAKE}^{CAuth}$ owns a list L initially empty of values of the form (P_i, P_j, pwd).

- **Upon receiving a query (`NewSession`, $ssid$, P_i, P_j, pwd, `role`) from P_i:**
  - Send (`NewSession`, $ssid$, P_i, P_j, `role`) to $\mathcal{A}$.
  - If this is the first `NewSession` query, or if it is the second `NewSession` query and there is a record $(P_j, P_i, pwd', \overline{\texttt{role}}) \in L$, then record $(P_i, P_j, pwd, \texttt{role})$ in L and mark this record `fresh`.
- **Upon receiving a query (`TestPwd`, $ssid$, P_i, pwd') from the adversary $\mathcal{A}$:**
  If there exists a record of the form $(P_i, P_j, pwd, \texttt{role}) \in L$ which is `fresh`, then do:
  - If $pwd = pwd'$, mark the record `compromised` and reply to $\mathcal{A}$ with "correct guess".
  - If $pwd \neq pwd'$, mark the record `interrupted` and reply to $\mathcal{A}$ with "wrong guess".
- **Upon receiving a query (`NewKey`, $ssid$, P_i, sk) from $\mathcal{A}$, where $|sk| = k$:**
  If there is a record of the form $(P_i, P_j, pwd, \texttt{role}) \in L$, and this is the first `NewKey` query for P_i, then:
  If `role` = `client` :
  - If the session is `compromised`, or if one of the two players P_i or P_j is corrupted, then send $(ssid, sk)$ to P_i, record $(P_i, P_j, pwd, \texttt{client}, \texttt{completed})$ in L, as well as $(ssid, P_i, pwd, sk, \texttt{client}, \texttt{status}, \texttt{ready})$ (with `status` being the status of the session at that moment.
  - Else, if the session is `fresh` or `interrupted`, choose a random key sk' whose length is k and send $(ssid, sk')$ to P_i. Record $(P_i, P_j, pwd, \texttt{client}, \texttt{completed})$ in L, as well as $(ssid, P_i, pwd, sk', \texttt{client}, \texttt{status}, \texttt{ready})$ where `status` stands for `fresh` or `interrupted`;

  If `role` = `server` :
  - If the session is `compromised`, if one of the two players P_i or P_j is corrupted, and if there are two records of the form $(P_i, P_j, pwd, \texttt{server})$ and $(P_j, P_i, pwd', \texttt{client})$, set $s = sk$. Otherwise, if the session is `fresh` and there exists any recorded element of the form $(ssid, P_j, pwd', sk', \texttt{client}, \texttt{fresh}, \texttt{ready})$, set $s = sk'$.
    - If $pwd = pwd'$, send $(ssid, s)$ to P_i, record $(P_i, P_j, pwd, \texttt{server}, \texttt{completed})$ in L, as well as $(ssid, P_i, pwd, s, \texttt{server}, \texttt{status})$.
    - If $pwd \neq pwd'$, send $(ssid, \texttt{error})$ to P_i, record $(P_i, P_j, pwd, \texttt{server}, \texttt{completed})$ in L, as well as $(ssid, P_i, pwd, \texttt{server}, \texttt{status})$.
  - if the session is `fresh` and there does not exist any recorded element of the form $(ssid, P_j, pwd', sk', \texttt{client}, \texttt{fresh}, \texttt{ready})$, then do not do anything.
  - If the session is `interrupted` then send $(ssid, \texttt{error})$ to player P_i, and record in L $(P_i, P_j, pwd, \texttt{server}, \texttt{completed})$ and $(ssid, P_i, pwd, \texttt{server}, \texttt{error}, \texttt{completed})$.

Figure 5.8: Ideal functionality $\mathcal{F}_{PAKE}^{CAuth}$: it is parametrizes by a security parameter k. It interacts with an adversary $\mathcal{A}$ and a set of parties $P_1, \ldots, P_n$. (from [2])

The functionality $\mathcal{F}_{RO}$ proceeds as follows, running on security parameters k, with parties $P_1, \ldots, P_n$ and an adversary $\mathcal{A}$:

- $\mathcal{F}_{RO}$ keeps a list L (which is initially empty) of pairs of bitstrings.
- Upon receiving a value $(ssid, m)$ (with $m \in \{0,1\}^*$) from some party P_i or from $\mathcal{A}$, do:
  - If there is a pair $(m, \overline{v})$ for some $\overline{v} \in \{0,1\}^k$ in the list L, set $v := \overline{v}$.
  - If there is no such pair, choose uniformly $v \in \{0,1\}^k$ and store the pair $(m, v) \in L$.

  Once v is set, reply to the activating machine (i.e., either P_i or $\mathcal{A}$) with $(ssid, v)$.

Figure 5.9: Functionality $\mathcal{F}_{RO}$ (from [2])

she does not make any attempts to guess their passwords;

- Server receives an error message if the parties share different passwords or the adversary tries to guess the parties' passwords but fails (she has only one attempt) [20];

- the parties receive the same key chosen by the adversary if the adversary corrupts one party (and the parties share the same password) or she correctly guesses a party's password (with only one try allowed).

Moreover, $\mathcal{F}_{PAKE}^{CAuth}$ allows an adversary, who does not corrupt the players and who does not correctly guess their passwords, to learn only when the parties start the PAKE protocol. We note that even if the adversary simply fails to guess the password of one party, without compromises the protocol, then the Server aborts the protocol. In this way, the functionality provides the client-to-server authentication, as Server accepts a key only if Client shares the same one.

As the functionality of [2] does not provide the passwords to the parties, it models the case in which the parties run the protocol with different or correlated passwords.

[20] However, Client receives a uniformly distributed random key as the authentication is only client-to-server.

The functionality F_{IC} takes as input the security parameter k, and interacts with an adversary $\mathcal{A}$ and with a set of (dummy) parties $P_1, \ldots, P_n$ by means of these queries:

- F_{IC} keeps a (initially empty) list L containing 3-tuples of bitstrings and a number of (initially empty) sets $C_{key,ssid}$, $M_{key,ssid}$.
- **Upon receiving a query** $(ssid, ENC, key, m)$ **(with** $m \in \{0,1\}^k$**) from some party** P_i **or** $\mathcal{A}$**, do:**
  - If there is a 3-tuple $(key, m, \overline{c})$ for some $\overline{c} \in \{0,1\}^k$ in the list L, set $c := \overline{c}$.
  - If there is no such record, choose uniformly c in $\{0,1\}^k - C_{key,ssid}$ which is the set consisting of ciphertexts not already used with key and $ssid$. Next, it stores the 3-tuple $(key, m, c) \in L$ and sets $C_{key,ssid} \leftarrow C_{key,ssid} \cup \{c\}$.

  Once c is set, reply to the activating machine with $(ssid, c)$.
- **Upon receiving a query** $(ssid, DEC, key, c)$ **(with** $c \in \{0,1\}^k$**) from some party** P_i **or** $\mathcal{A}$**, do:**
  - If there is a 3-tuple $(key, \tilde{m}, c)$ for some $\tilde{m} \in \{0,1\}^k$ in the list L, set $m := \tilde{m}$.
  - If there is no such record, choose uniformly m in $\{0,1\}^k - M_{key,ssid}$ which is the set consisting of plaintexts not already used with key and $ssid$. Next, it stores the 3-tuple $(key, m, c) \in L$ and sets $M_{key,ssid} \leftarrow M_{key,ssid} \cup \{m\}$.

  Once m is set, reply to the activating machine with $(ssid, m)$.

Figure 5.10: Functionality $\mathcal{F}_{IC}$ (from [2])

5.3.3 Authenticated Key Exchange

In this section, following the proof sketched in [42], we show that the keys output by the AKE protocol of Figure 5.5 are indistinguishable from random to an adversary point of view (and equal), if the parties accept them.

In our AKE protocol, we use a UC-PAKE protocol, therefore we can substitute its execution with calls to the ideal functionality $\mathcal{F}_{PAKE}^{CAuth}$ of Figure 5.8, using Pwd_C and Pwd_S of Figure 5.3, the weak keys output by the WKE protocol, as the passwords for the `NewSession` query of P_i and P_j, respectively. By the definition of $\mathcal{F}_{PAKE}^{CAuth}$, only if an adversary corrupts a party or correctly guesses the password of one of the parties (before the end of the PAKE protocol) then she can choose the output keys of the parties. Otherwise, the keys output by the ideal functionality are indistinguishable from random to the adversary (or Server simply does not accept any key). To guess the passwords used by the parties in the PAKE protocol, the adversary has only one attempt, using the `TestPwd` query (cf. Figure 5.8). By the analysis of the WKE protocol (see Lemma 13), we know that the passwords output by the WKE protocol, and then used in the PAKE protocol, have (with high probability $1 - 2^{-\epsilon}$, where ϵ is the security parameter[21]) a min-entropy equals to ϵ conditioned on the adversary view before the call to $\mathcal{F}_{PAKE}^{CAuth}$, which is equal to the adversarial view after the WKE protocol plus the nonces exchanged between the parties to create the random $ssid$, and this holds even after that the adversary retrieves up to λ bits from the internal state of the parties and the shared file F, before the beginning of the WKE protocol. Therefore, an adversary can correctly guess in one attempt the passwords used in the PAKE protocol only with very low probability, chosen by Client. Then we can conclude that the keys sk_C and sk_S obtained by the parties at the end of the AKE protocol, are indistinguishable from random to the adversary. Moreover, these keys are equal as otherwise Server receives an error message (and rejects the key), providing the client-to-server authentication.

Finally, we note that even after the end of the AKE protocol, the adversary can never distinguish sk_C and sk_S from random. Indeed, if, after the end of the AKE protocol, an adversary retrieves the bits of the shared file used in the AKE protocol to compute the passwords output by the WKE protocol (without violating the retrieval bound), she can clearly compute the passwords but still she cannot distinguish the secret keys obtained by the parties from random, as they have been chosen at random by the functionality $\mathcal{F}_{PAKE}^{CAuth}$. If we look at the UC-PAKE protocol, it means that even if the adversary learns, after the execution of the protocol, the value of the passwords used in the protocol, she

[21] We remind that by default we set ϵ equal to 30 but Client can choose an arbitrary value, greater than 30, when she runs the protocol.

cannot distinguish the output keys from random, as the protocol, from her point of view, becomes a DH key exchange (see Figure 5.1), as now she can decrypt the Server's public DH value, and we rely on the CDH assumption (see Section 2.2).

Even if the adversary is given all the shared file, still the security guarantees of the completed AKE protocol hold but then, of course, there is no more security for future running of the protocol, as the retrieval bound requirement is gone.

Chapter 6

Conclusion and future work

The study of the side-channel attacks showed that the black-box model of the provable security approach is not enough for real-world implementation. Indeed, in the last years, several cryptographic schemes that were proven secure in the black-box model have been broken exploiting additional leakage information from the physical implementation of the scheme. Today, one of the main challenge of the cryptographic community is to combine the analysis of physical leakage with provable security, to design schemes that remains secure even after the implementation in the real world, where physical devices may leak information. After a period in which mostly the practitioners were involved in this study, exploiting side-channel measurements and countermeasures, recently also the theoretician began to proposed new formal models to extend the provable security to the side-channel attacks settings. However, in most cases, theoretical results are provable secure but are based on several assumptions on the type of leakage and often achieve complicated schemes, while practical works consider efficient but simple schemes, often without formal models and proofs. To further improve this research area and the design of a provable secure *and* efficiently implementable cryptography, the interaction between theoretician and practitioners is needed, for example, to better understand which theoretical assumptions are empirically verifiable and which constructions can be instantiated or implemented in practice.

We consider our work as a contribution to this emerging field, as a step to bridge the gap between theory and practice in this area, which is one of the topic on which the cryptographic community's interest is growing very fast.

In Chapter 4, we introduced a new primitive to securely store data on an hardware that may leak information. We showed the security of two constructions, respectively in the setting where

- the adversarial leakage functions are applied independently to different

parts of the system's memory;

- the size of the class of adversarial leakage functions is restricted, considering computationally bounded functions, e.g. functions computable by circuits of small size.

Our security proofs are information-theoretic, rely on deterministic extractors and are developed in the model where the total amount of leakage is bounded.

Our first construction is instantiated with two-source extractors and assumes that different parts of the memory leaks independently. This is a bit controversial assumption as, even if it captures any affine leakage functions, recent practical works (e.g. [147]) showed that this restriction does not model some non-linear physical leakage. However, as noted in [70], this limitation holds if one assumes that the logical gates of a system leaks independently and it could be less problematic when assuming that (at higher hardware level) different parts of the memory leaks independently.

Recently, Dziembowski and Faust [70] extended our result based on two-source extractors in the continual leakage model, where the total amount of leakage is unbounded (this is bounded only per time period) and there exists a way to refresh the internal state of the system. However, also this work relies on the assumption that different parts of the memory leaks independently and uses some leak-free hardware components.

We note that Dodis et al. [63] introduced a method for storing (and refreshing) a secret, shared between several devices (or different parts of a single device), in the continual memory-leakage model, where all the memory can leak and the total amount of leakage is unbounded, which does not use leak-free components, but it is rather inefficient and it is not information-theoretic secure, as it relies on non-standard assumptions. In particular, in [63] is shown that it is impossible to achieve an information-theoretically secure sharing storage scheme resilient to leakage in the continual memory-leakage model if during the refreshing phase there is no communication between the parties who share the secret data.

As future research direction, we think it is interesting to study whether it is possible to combine the results of [70] and [63]. Moreover, it is also worthy to improve those schemes, for example, allowing a larger quantity of leakage, in total or during the update phase, using weaker assumptions or avoiding the use of leak-free hardware components. An other research question is whether it is possible to obtain information-theoretically secure sharing schemes, as those one described in [63], if the parties who share the secret data can interactively

perform the refreshing phase.

Our second construction holds in the bounded memory-leakage model, where the total amount of leakage is bounded but all the memory can leak. There are still few works which consider computationally bounded leakage functions, even though it seems a very promising approach as it models practical attacks, which are mostly described by simple aggregate functions, and avoids artificial attacks not feasible in practice. Our construction come with formal security proofs, but the analysis of concrete parameters does not seem to allow for efficient feasibility of these schemes in practice.

As discussed in Section 4.5, we think that the idea to model the leakage as computationally bounded functions, instead of polynomial-time ones, and the study of realistic classes of such functions, is an interesting research direction, as it avoids to give too much computational power to an attacker and allows the analysis of real leakage measurement that are exploited in practice.

In Chapter 5, we discussed our implementation of an Authenticated Key Exchange protocol in the C programming language, using the cryptographic library of the OpenSSL project. The implemented AKE protocol is based on a Weak Key Exchange protocol, constructed following the work of [42] which uses less randomness but is not efficiently implementable, and on the Password-based Authenticated Key Exchange showed secure in the UC framework in [2], and allows a client and a server, who share a huge secret file, to securely compute a shared key, providing client-to-server authentication, also in the presence of active attackers.

Recently, Dziembowski et al. [71] introduced a new model, which assumes that "the leakage can be any arbitrary space bounded computation that can make random oracle calls itself", and showed a key-evolution scheme secure in this model. We think it is an interesting question whether it is possible to use the construction of [71] in the AKE protocol implemented in Chapter 5 to evolve the long shared key and to obtain security against an unbounded amount of total leakage.

Appendix A

Omitted Proofs

A.1 Proofs for Chapter 4

A.1.1 Proof of Lemma 1

Before showing this lemma let us first prove the following:

Lemma 14. *For every random variable X and events $\mathcal{E}, \mathcal{V}$ we have*

$$d(X|\mathcal{V}) \leq d(X|\mathcal{V} \wedge \mathcal{E}) + \boldsymbol{Pr}\left[\overline{\mathcal{E}}|\mathcal{V}\right]. \tag{A.1}$$

Proof. It is enough to show that

$$\Delta(P_{X|\mathcal{V}}; P_{X|\mathcal{V}\wedge\mathcal{E}}) \leq \mathbf{Pr}\left[\overline{\mathcal{E}}|\mathcal{V}\right]. \tag{A.2}$$

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

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

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

$$\{x : \mathbf{Pr}\left[X = x|\mathcal{V}\right] > \mathbf{Pr}\left[X = x|\mathcal{V} \wedge \mathcal{E}\right]\}.$$

We have that the left-hand side of (A.2) is equal to

$$\sum_{x\in\mathcal{F}} \mathbf{Pr}\left[X=x|\mathcal{V}\right] - \overbrace{\mathbf{Pr}\left[X=x|\mathcal{V}\wedge\mathcal{E}\right]}^{=\frac{\mathbf{Pr}[X=x\wedge\mathcal{E}|\mathcal{V}]}{\mathbf{Pr}[\mathcal{E}|\mathcal{V}]}\geq\mathbf{Pr}[X=x\wedge\mathcal{E}|\mathcal{V}]} \tag{A.3}$$

$$\leq \sum_{x\in\mathcal{F}} \mathbf{Pr}\left[X=x|\mathcal{V}\right] - \mathbf{Pr}\left[X=x\wedge\mathcal{E}|\mathcal{V}\right] \tag{A.4}$$

$$= \sum_{x\in\mathcal{F}} \mathbf{Pr}\left[X=x|\mathcal{V}\right] - \sum_{x\in\mathcal{F}} \mathbf{Pr}\left[X=x\wedge\mathcal{E}|\mathcal{V}\right] \tag{A.5}$$

$$= \mathbf{Pr}\left[X\in\mathcal{F}|\mathcal{V}\right] - \mathbf{Pr}\left[(X\in\mathcal{F})\wedge\mathcal{E}|\mathcal{V}\right] \tag{A.6}$$

$$\leq \mathbf{Pr}\left[\overline{\mathcal{E}}|\mathcal{V}\right]. \tag{A.7}$$

□

Proof of Lemma 1. The right-hand side of (4.2) is equal to

$$\sum_{y} d(X|Y=y)\cdot\mathbf{Pr}\left[Y=y\right], \tag{A.8}$$

and the left-hand side of (4.2) is equal to

$$\sum_{y} \left(d(X|(Y=y)\wedge\mathcal{E}) + \mathbf{Pr}\left[\overline{\mathcal{E}}|Y=y\right]\right)\cdot\mathbf{Pr}\left[Y=y\right]. \tag{A.9}$$

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

$$d(X|(Y=y)\wedge\mathcal{E}) + \mathbf{Pr}\left[\overline{\mathcal{E}}|Y=y\right] \geq d(X|Y=y).$$

This follows directly from Lemma 14, with $\mathcal{V}$ being the event that $Y=y$. □

A.1.2 Proof of Lemma 6

Proof. We prove that (4.4) holds for a fixed w. This clearly implies that (4.4) holds when W is a random variable independent on X. Since $|f(X,w)|\leq\lambda$, hence the number of all y's is at most equal to 2^λ. Therefore the number of x's for which there exists some y such that

$$|x : f(x,w)=y| \leq 2^k \tag{A.10}$$

holds is at most $2^{\lambda+k}$. Hence the probability that for a *random* X we have that (A.10) holds is at most $2^{\lambda+k-n}$. Since clearly if (A.10) does not hold then $\mathbf{H}_\infty(X|f(X,w)=y)>k$ we get that

$$\mathbb{P}_{y:=f(X,w)}(\mathbf{H}_\infty(X|f(X,w)=y)\leq k)\leq 2^{\lambda+k-n}.$$

Thus we are done. □

Bibliography

[1] *51th Annual IEEE Symposium on Foundations of Computer Science, FOCS 2010, October 23-26, 2010, Las Vegas, Nevada, USA* (2010), IEEE Computer Society.

[2] Abdalla, M., Catalano, D., Chevalier, C., and Pointcheval, D. Efficient two-party password-based key exchange protocols in the UC framework. In *CT-RSA* (2008), T. Malkin, Ed., vol. 4964 of *Lecture Notes in Computer Science*, Springer, pp. 335–351.

[3] Abdalla, M., and Pointcheval, D. Simple password-based encrypted key exchange protocols. In *CT-RSA* (2005), A. Menezes, Ed., vol. 3376 of *Lecture Notes in Computer Science*, Springer, pp. 191–208.

[4] Abe, M., Ed. *Advances in Cryptology - ASIACRYPT 2010 - 16th International Conference on the Theory and Application of Cryptology and Information Security, Singapore, December 5-9, 2010. Proceedings* (2010), vol. 6477 of *Lecture Notes in Computer Science*, Springer.

[5] Akavia, A., Goldwasser, S., and Vaikuntanathan, V. Simultaneous hardcore bits and cryptography against memory attacks. In *Theory of Cryptography, 6th Theory of Cryptography Conference, TCC 2009, San Francisco, CA, USA, March 15-17, 2009. Proceedings* (2009), O. Reingold, Ed., vol. 5444 of *Lecture Notes in Computer Science*, Springer, pp. 474–495.

[6] Alwen, J., Dodis, Y., Naor, M., Segev, G., Walfish, S., and Wichs, D. Public-key encryption in the bounded-retrieval model. In Gilbert [85], pp. 113–134.

[7] Alwen, J., Dodis, Y., and Wichs, D. Leakage-resilient public-key cryptography in the bounded-retrieval model. In Halevi [92], pp. 36–54.

[8] Anderson, R., and Kuhn, M. Tamper resistance: a cautionary note. In *WOEC'96: Proceedings of the 2nd conference on Proceedings of the Second*

USENIX Workshop on Electronic Commerce (Berkeley, CA, USA, 1996), USENIX Association, pp. 1–11.

[9] Aumann, Y., Ding, Y. Z., and Rabin, M. O. Everlasting security in the bounded storage model. *IEEE Transactions on Information Theory 48*, 6 (2002), 1668–1680.

[10] Barak, B., Shaltiel, R., and Tromer, E. True random number generators secure in a changing environment. In *CHES* (2003), C. D. Walter, Ç. K. Koç, and C. Paar, Eds., vol. 2779 of *Lecture Notes in Computer Science*, Springer, pp. 166–180.

[11] Bellare, M., and Miner, S. K. A forward-secure digital signature scheme. In Wiener [156], pp. 431–448.

[12] Bellare, M., Pointcheval, D., and Rogaway, P. Authenticated key exchange secure against dictionary attacks. In Preneel [132], pp. 139–155.

[13] Bellare, M., and Rogaway, P. Entity authentication and key distribution. In *CRYPTO* (1993), D. R. Stinson, Ed., vol. 773 of *Lecture Notes in Computer Science*, Springer, pp. 232–249.

[14] Bellare, M., and Rogaway, P. Random oracles are practical: a paradigm for designing efficient protocols. In *Proceedings of the 1st ACM conference on Computer and communications security* (New York, NY, USA, 1993), CCS '93, ACM, pp. 62–73.

[15] Bellare, M., and Rogaway, P. Provably secure session key distribution: the three party case. In *STOC '95: Proceedings of the Twenty-Seventh Annual ACM Symposium on Theory of Computing, 29 May-1 June 1995, Las Vegas, Nevada, USA* (1995), ACM, pp. 57–66.

[16] Bellare, M., and Rogaway, P. The AuthA protocol for password-based authenticated key exchange. In *Contributions to IEEE P1363* (2000).

[17] Bellare, M., and Rompel, J. Randomness-efficient oblivious sampling. In *FOCS '94: Proceedings of 35th Annual Symposium on Foundations of Computer Science, 20-22 November 1994, Santa Fe, New Mexico, USA* (1994), IEEE, pp. 276–287.

[18] Bellare, M., and Yee, B. S. Forward-security in private-key cryptography. In *CT-RSA* (2003), M. Joye, Ed., vol. 2612 of *Lecture Notes in Computer Science*, Springer, pp. 1–18.

[19] Bellovin, S. M., and Merritt, M. Augmented encrypted key exchange: A password-based protocol secure against dictionary attacks and password file compromise. In *ACM Conference on Computer and Communications Security* (1993), pp. 244–250.

[20] Biham, E., and Shamir, A. Differential cryptanalysis of des-like cryptosystems. *J. Cryptology 4*, 1 (1991), 3–72.

[21] Black, J. The ideal-cipher model, revisited: An uninstantiable blockcipher-based hash function. In *FSE* (2006), M. J. B. Robshaw, Ed., vol. 4047 of *Lecture Notes in Computer Science*, Springer, pp. 328–340.

[22] Black, J., Rogaway, P., and Shrimpton, T. Black-box analysis of the block-cipher-based hash-function constructions from PGV. In Yung [160], pp. 320–335.

[23] Blum, M. Independent unbiased coin flips from a correlated biased source: a finite state markov chain. In *FOCS '84: Proceedings of 25th Annual Symposium on Foundations of Computer Science, 24-26 October 1984, Singer Island, Florida, USA* (1984), IEEE, pp. 425–433.

[24] Boneh, D., Ed. *Advances in Cryptology - CRYPTO 2003, 23rd Annual International Cryptology Conference, Santa Barbara, California, USA, August 17-21, 2003, Proceedings* (2003), vol. 2729 of *Lecture Notes in Computer Science*, Springer.

[25] Boneh, D., DeMillo, R. A., and Lipton, R. J. On the importance of checking cryptographic protocols for faults (extended abstract). In *EUROCRYPT* (1997), pp. 37–51.

[26] Boyen, X., Dodis, Y., Katz, J., Ostrovsky, R., and Smith, A. Secure remote authentication using biometric data. In Cramer [48], pp. 147–163.

[27] Boyko, V., MacKenzie, P. D., and Patel, S. Provably secure password-authenticated key exchange using diffie-hellman. In Preneel [132], pp. 156–171.

[28] Boyle, E., Segev, G., and Wichs, D. Fully leakage-resilient signatures. In *EUROCRYPT* (2011), K. G. Paterson, Ed., vol. 6632 of *Lecture Notes in Computer Science*, Springer, pp. 89–108.

[29] Brakerski, Z., and Goldwasser, S. Circular and leakage resilient public-key encryption under subgroup indistinguishability (or: Quadratic residuosity strikes back). In Rabin [135], pp. 1–20.

[30] Brakerski, Z., Kalai, Y. T., Katz, J., and Vaikuntanathan, V. Overcoming the hole in the bucket: Public-key cryptography resilient to continual memory leakage. In *FOCS '10* [1], pp. 501–510.

[31] Bresson, E., Chevassut, O., and Pointcheval, D. Security proofs for an efficient password-based key exchange. In *ACM Conference on Computer and Communications Security* (2003), S. Jajodia, V. Atluri, and T. Jaeger, Eds., ACM, pp. 241–250.

[32] Bresson, E., Chevassut, O., and Pointcheval, D. New security results on encrypted key exchange. In *Public Key Cryptography* (2004), F. Bao, R. H. Deng, and J. Zhou, Eds., vol. 2947 of *Lecture Notes in Computer Science*, Springer, pp. 145–158.

[33] Brusilovsky, A., Faynberg, I., and Zeltsan, Z. Password-Authenticated Key (PAK) Diffie-Hellman Exchange RFC5683, February 2010.

[34] Cachin, C., and Maurer, U. M. Unconditional security against memory-bounded adversaries. In *CRYPTO* (1997), B. S. K. Jr., Ed., vol. 1294 of *Lecture Notes in Computer Science*, Springer, pp. 292–306.

[35] Canetti, R. Universally composable security: A new paradigm for cryptographic protocols. In *FOCS* (2001), pp. 136–145.

[36] Canetti, R., Dodis, Y., Halevi, S., Kushilevitz, E., and Sahai, A. Exposure-Resilient Functions and All-Or-Nothing Transforms. In *EUROCRYPT* (2000), pp. 453–469.

[37] Canetti, R., Goldreich, O., and Halevi, S. On the random-oracle methodology as applied to length-restricted signature schemes. In Naor [126], pp. 40–57.

[38] Canetti, R., Goldreich, O., and Halevi, S. The random oracle methodology, revisited. *J. ACM 51*, 4 (2004), 557–594.

[39] Canetti, R., Halevi, S., and Katz, J. A forward-secure public-key encryption scheme. *J. Cryptology 20*, 3 (2007), 265–294.

[40] Canetti, R., Halevi, S., Katz, J., Lindell, Y., and MacKenzie, P. D. Universally Composable password-based key exchange. In Cramer [48], pp. 404–421.

[41] Canetti, R., and Rabin, T. Universal composition with joint state. In Boneh [24], pp. 265–281.

[42] Cash, D., Ding, Y. Z., Dodis, Y., Lee, W., Lipton, R. J., and Walfish, S. Intrusion-resilient key exchange in the bounded retrieval model. In *TCC* (2007), S. P. Vadhan, Ed., vol. 4392 of *Lecture Notes in Computer Science*, Springer, pp. 479–498.

[43] Chari, S., Jutla, C. S., Rao, J. R., and Rohatgi, P. Towards sound approaches to counteract power-analysis attacks. In Wiener [156], pp. 398–412.

[44] Chor, B., and Goldreich, O. Unbiased bits from sources of weak randomness and probabilistic communication complexity. *SIAM J. Comput. 17*, 2 (1988), 230–261.

[45] Chor, B., Goldreich, O., Håstad, J., Friedman, J., Rudich, S., and Smolensky, R. The bit extraction problem of t-resilient functions (preliminary version). In *FOCS '85: Proceedings of 26th Annual Symposium on Foundations of Computer Science, 21-23 October 1985, Portland, Oregon, USA* (1985), IEEE, pp. 396–407.

[46] Coron, J.-S., Dodis, Y., Malinaud, C., and Puniya, P. Merkle-damgård revisited: How to construct a hash function. In *CRYPTO* (2005), V. Shoup, Ed., vol. 3621 of *Lecture Notes in Computer Science*, Springer, pp. 430–448.

[47] Coron, J.-S., Patarin, J., and Seurin, Y. The random oracle model and the ideal cipher model are equivalent. In Wagner [154], pp. 1–20.

[48] Cramer, R., Ed. *Advances in Cryptology - EUROCRYPT 2005, 24th Annual International Conference on the Theory and Applications of Cryptographic Techniques, Aarhus, Denmark, May 22-26, 2005, Proceedings* (2005), vol. 3494 of *Lecture Notes in Computer Science*, Springer.

[49] Davì, F., Dziembowski, S., and Venturi, D. Leakage-resilient storage. In *SCN* (2010), J. A. Garay and R. De Prisco, Eds., vol. 6280 of *Lecture Notes in Computer Science*, Springer, pp. 121–137.

[50] Desai, A. The security of all-or-nothing encryption: Protecting against exhaustive key search. In *CRYPTO* (2000), M. Bellare, Ed., vol. 1880 of *Lecture Notes in Computer Science*, Springer, pp. 359–375.

[51] Di Crescenzo, G., Lipton, R. J., and Walfish, S. Perfectly secure password protocols in the bounded retrieval model. In Halevi and Rabin [93], pp. 225–244.

[52] Diffie, W., and Hellman, M. E. New directions in cryptography. *IEEE Transactions on Information Theory 22*, 6 (November 1976), 644–654.

[53] Ding, Y. Z. Error correction in the bounded storage model. In *TCC* (2005), J. Kilian, Ed., vol. 3378 of *Lecture Notes in Computer Science*, Springer, pp. 578–599.

[54] Dodis, Y., Elbaz, A., Oliveira, R., and Raz, R. Improved randomness extraction from two independent sources. In *Approximation, Randomization, and Combinatorial Optimization, Algorithms and Techniques, 7th International Workshop on Approximation Algorithms for Combinatorial Optimization Problems, APPROX 2004, and 8th International Workshop on Randomization and Computation, RANDOM 2004, Cambridge, MA, USA, August 22-24, 2004, Proceedings* (2004), vol. 3122 of *Lecture Notes in Computer Science*, Springer, pp. 334–344.

[55] Dodis, Y., Franklin, M. K., Katz, J., Miyaji, A., and Yung, M. Intrusion-resilient public-key encryption. In *CT-RSA* (2003), pp. 19–32.

[56] Dodis, Y., Goldwasser, S., Kalai, Y. T., Peikert, C., and Vaikuntanathan, V. Public-key encryption schemes with auxiliary inputs. In Micciancio [125], pp. 361–381.

[57] Dodis, Y., Haralambiev, K., López-Alt, A., and Wichs, D. Cryptography against continuous memory attacks. In *FOCS '10* [1], pp. 511–520.

[58] Dodis, Y., Haralambiev, K., López-Alt, A., and Wichs, D. Efficient public-key cryptography in the presence of key leakage. In Abe [4], pp. 613–631.

[59] Dodis, Y., Kalai, Y. T., and Lovett, S. On cryptography with auxiliary input. In *STOC '09: Proceedings of the 41st annual ACM symposium on Theory of computing* (New York, NY, USA, 2009), ACM, pp. 621–630.

[60] Dodis, Y., Katz, J., Reyzin, L., and Smith, A. Robust fuzzy extractors and authenticated key agreement from close secrets. In Dwork [67], pp. 232–250.

[61] Dodis, Y., Katz, J., Xu, S., and Yung, M. Key-insulated public key cryptosystems. In *EUROCRYPT* (2002), L. R. Knudsen, Ed., vol. 2332 of *Lecture Notes in Computer Science*, Springer, pp. 65–82.

[62] Dodis, Y., Katz, J., Xu, S., and Yung, M. Strong key-insulated signature schemes. In *Public Key Cryptography* (2003), Y. Desmedt, Ed., vol. 2567 of *Lecture Notes in Computer Science*, Springer, pp. 130–144.

[63] Dodis, Y., Lewko, A., Waters, B., and Wichs, D. Storing secrets on continually leaky devices. Accepted at FOCS 2011. IACR Cryptology ePrint Archive, Report 2011/369, 2011. `http://eprint.iacr.org/2011/369`.

[64] Dodis, Y., Ostrovsky, R., Reyzin, L., and Smith, A. Fuzzy extractors: How to generate strong keys from biometrics and other noisy data. *SIAM J. Comput. 38*, 1 (2008), 97–139.

[65] Dodis, Y., and Pietrzak, K. Leakage-resilient pseudorandom functions and side-channel attacks on Feistel networks. In Rabin [135], pp. 21–40.

[66] Dodis, Y., Sahai, A., and Smith, A. On perfect and adaptive security in exposure-resilient cryptography. In Pfitzmann [130], pp. 301–324.

[67] Dwork, C., Ed. *Advances in Cryptology - CRYPTO 2006, 26th Annual International Cryptology Conference, Santa Barbara, California, USA, August 20-24, 2006, Proceedings* (2006), vol. 4117 of *Lecture Notes in Computer Science*, Springer.

[68] Dziembowski, S. Intrusion-resilience via the bounded-storage model. In Halevi and Rabin [93], pp. 207–224.

[69] Dziembowski, S. On forward-secure storage. In Dwork [67], pp. 251–270.

[70] Dziembowski, S., and Faust, S. Leakage-resilient cryptography from the inner-product extractor. In *ASIACRYPT* (2011), D. H. Lee and X. Wang, Eds., vol. 7073 of *Lecture Notes in Computer Science*, Springer, pp. 702–721.

[71] Dziembowski, S., Kazana, T., and Wichs, D. Key-evolution schemes resilient to space-bounded leakage. In *CRYPTO* (2011), P. Rogaway, Ed., vol. 6841 of *Lecture Notes in Computer Science*, Springer, pp. 335–353.

[72] Dziembowski, S., and Maurer, U. M. On generating the initial key in the bounded-storage model. In *EUROCRYPT* (2004), C. Cachin and J. Camenisch, Eds., vol. 3027 of *Lecture Notes in Computer Science*, Springer, pp. 126–137.

[73] Dziembowski, S., and Maurer, U. M. Optimal randomizer efficiency in the bounded-storage model. *J. Cryptology 17*, 1 (2004), 5–26.

[74] Dziembowski, S., and Pietrzak, K. Intrusion-resilient secret sharing. In *FOCS '07: Proceedings of 48th Annual IEEE Symposium on Foundations of Computer Science, October 20-23, 2007, Providence, RI, USA* (2007), IEEE Computer Society, pp. 227–237.

[75] Dziembowski, S., and Pietrzak, K. 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.

[76] Eisenbarth, T., Kasper, T., Moradi, A., Paar, C., Salmasizadeh, M., and Shalmani, M. T. M. On the power of power analysis in the real world: A complete break of the keeloqcode hopping scheme. In Wagner [154], pp. 203–220.

[77] European Network of Excellence in Cryptology (ECRYPT). The Side Channel Cryptanalysis Lounge. A webpage: `http://www.emsec.rub.de/research/projects/sclounge/` accessed on 9.12.2011.

[78] Even, S., and Mansour, Y. A construction of a cioher from a single pseudorandom permutation. In *ASIACRYPT* (1991), H. Imai, R. L. Rivest, and T. Matsumoto, Eds., vol. 739 of *Lecture Notes in Computer Science*, Springer, pp. 210–224.

[79] Faust, S. Provable security at implementation-level. PhD thesis, Katholieke Universiteit Leuven, 2010.

[80] Faust, S., Kiltz, E., Pietrzak, K., and Rothblum, G. N. Leakage-resilient signatures. In Micciancio [125], pp. 343–360.

[81] Faust, S., Rabin, T., Reyzin, L., Tromer, E., and Vaikuntanathan, V. Protecting circuits from leakage: the computationally-bounded and noisy cases. In Gilbert [85], pp. 135–156.

[82] Gandolfi, K., Mourtel, C., and Olivier, F. Electromagnetic analysis: Concrete results. In *CHES* (2001), Ç. K. Koç, D. Naccache, and C. Paar, Eds., vol. 2162 of *Lecture Notes in Computer Science*, Springer, pp. 251–261.

[83] Gennaro, R., and Lindell, Y. A framework for password-based authenticated key exchange. In *EUROCRYPT* (2003), E. Biham, Ed., vol. 2656 of *Lecture Notes in Computer Science*, Springer, pp. 524–543.

[84] Gentry, C., Peikert, C., and Vaikuntanathan, V. Trapdoors for hard lattices and new cryptographic constructions. In *STOC '08: Proceedings of the 40th Annual ACM Symposium on Theory of Computing, Victoria, British Columbia, Canada, May 17-20, 2008* (2008), C. Dwork, Ed., ACM, pp. 197–206.

[85] Gilbert, H., Ed. *Advances in Cryptology - EUROCRYPT 2010, 29th Annual International Conference on the Theory and Applications of Cryptographic Techniques, French Riviera, May 30 - June 3, 2010. Proceedings* (2010), vol. 6110 of *Lecture Notes in Computer Science*, Springer.

[86] Goldreich, O., and Lindell, Y. Session-key generation using human passwords only. In *CRYPTO* (2001), J. Kilian, Ed., vol. 2139 of *Lecture Notes in Computer Science*, Springer, pp. 408–432.

[87] Goldwasser, S., Kalai, Y. T., Peikert, C., and Vaikuntanathan, V. Robustness of the learning with errors assumption. In *ICS* (2010), A. C.-C. Yao, Ed., Tsinghua University Press, pp. 230–240.

[88] Goldwasser, S., and Micali, S. Probabilistic encryption. *J. Comput. Syst. Sci. 28*, 2 (1984), 270–299.

[89] Goldwasser, S., and Rothblum, G. N. Securing computation against continuous leakage. In Rabin [135], pp. 59–79.

[90] Golic, J. D., and Tymen, C. Multiplicative masking and power analysis of aes. In Kaliski Jr et al. [102], pp. 198–212.

[91] Halderman, A. J., Schoen, S. D., Heninger, N., Clarkson, W., Paul, W., Calandrino, J. A., Feldman, A. J., Appelbaum, J., and Felten, E. W. Lest we remember: cold boot attacks on encryption keys. *Commun. ACM 52*, 5 (2009), 91–98.

[92] Halevi, S., Ed. *Advances in Cryptology - CRYPTO 2009, 29th Annual International Cryptology Conference, Santa Barbara, CA, USA, August 16-20, 2009. Proceedings* (2009), vol. 5677 of *Lecture Notes in Computer Science*, Springer.

[93] Halevi, S., and Rabin, T., Eds. *Theory of Cryptography, Third Theory of Cryptography Conference, TCC 2006, New York, NY, USA, March 4-7, 2006, Proceedings* (2006), vol. 3876 of *Lecture Notes in Computer Science*, Springer.

[94] Harkins, D., and Carrel, D. The Internet Key Exchange (IKE). RFC2409, November 1998.

[95] Harnik, D., and Naor, M. On the compressibility of NP instances and cryptographic applications. In *FOCS '06: Proceedings of the 47th Annual IEEE Symposium on Foundations of Computer Science* (Washington, DC, USA, 2006), IEEE Computer Society, pp. 719–728.

[96] Hofheinz, D., and Müller-Quade, J. Universally composable commitments using random oracles. In Naor [126], pp. 58–76.

[97] IEEE Standard 1363.2 Study Group. Password-based public-key cryptography. Webpage: `http://grouper.ieee.org/groups/1363/passwdPK/` accessed on 9.12.2011.

[98] Ishai, Y., Sahai, A., and Wagner, D. Private Circuits: Securing Hardware against Probing Attacks. In *CRYPTO* (2003), pp. 463–481.

[99] Itkis, G., and Reyzin, L. Sibir: Signer-base intrusion-resilient signatures. In Yung [160], pp. 499–514.

[100] Juma, A., and Vahlis, Y. Protecting cryptographic keys against continual leakage. In Rabin [135], pp. 41–58.

[101] Kaliski, B. PKCS #5: Password-Based Cryptography Specification. Version 2.0. RFC2898, September 2000.

[102] Kaliski Jr, B. S., Koç, Ç. K., and Paar, C., Eds. *Cryptographic Hardware and Embedded Systems - CHES 2002, 4th International Workshop, Redwood Shores, CA, USA, August 13-15, 2002, Revised Papers* (2003), vol. 2523 of *Lecture Notes in Computer Science*, Springer.

[103] Kamp, J., and Zuckerman, D. Deterministic extractors for bit-fixing sources and exposure-resilient cryptography. *SIAM J. Comput. 36*, 5 (2007), 1231–1247.

[104] Katz, J., Ostrovsky, R., and Yung, M. Efficient password-authenticated key exchange using human-memorable passwords. In Pfitzmann [130], pp. 475–494.

[105] Katz, J., Ostrovsky, R., and Yung, M. Forward secrecy in password-only key exchange protocols. In *SCN* (2002), S. Cimato, C. Galdi, and G. Persiano, Eds., vol. 2576 of *Lecture Notes in Computer Science*, Springer, pp. 29–44.

[106] Katz, J., and Vaikuntanathan, V. Signature schemes with bounded leakage resilience. In *ASIACRYPT* (2009), M. Matsui, Ed., vol. 5912 of *Lecture Notes in Computer Science*, Springer, pp. 703–720.

[107] Kelsey, J., Schneier, B., Wagner, D., and Hall, C. Side channel cryptanalysis of product ciphers. *Journal of Computer Security 8* (2000), 141–158.

[108] Kilian, J., and Rogaway, P. How to protect DES against exhaustive key search. In Koblitz [111], pp. 252–267.

[109] Kiltz, E., and Pietrzak, K. Leakage resilient ElGamal encryption. In Abe [4], pp. 595–612.

[110] Kivinen, T., and Kojo, M. More Modular Exponential (MODP) Diffie-Hellman groups for Internet Key Exchange (IKE). RFC3526, May 2003.

[111] Koblitz, N., Ed. *Advances in Cryptology - CRYPTO '96, 16th Annual International Cryptology Conference, Santa Barbara, California, USA, August 18-22, 1996, Proceedings* (1996), vol. 1109 of *Lecture Notes in Computer Science*, Springer.

[112] Kocher, P. C. Timing attacks on implementations of Diffie-Hellman, RSA, DSS, and other systems. In Koblitz [111], pp. 104–113.

[113] Kocher, P. C., Jaffe, J., and Jun, B. Differential power analysis. In Wiener [156], pp. 388–397.

[114] Lewko, A. B., Lewko, M., and Waters, B. How to leak on key updates. In *STOC '11: Proceedings of the 43rd ACM Symposium on Theory of Computing, STOC 2011, San Jose, CA, USA, 6-8 June 2011* (2011), L. Fortnow and S. P. Vadhan, Eds., ACM, pp. 725–734.

[115] Lewko, A. B., Rouselakis, Y., and Waters, B. Achieving leakage resilience through dual system encryption. In *TCC* (2011), Y. Ishai, Ed., vol. 6597 of *Lecture Notes in Computer Science*, Springer, pp. 70–88.

[116] Loosemore, S., Stallman, R. M., McGrath, R., Oram, A., , and Drepper, U. The GNU C library reference manual. `http://www.gnu.org/s/libc/manual/` accessed on 9.12.2011.

[117] Lu, C.-J. Encryption against storage-bounded adversaries from on-line strong extractors. *J. Cryptology 17*, 1 (2004), 27–42.

[118] MacKenzie, P. The pak suite: Protocols for password-authenticated key exchange. In *IEEE P1363.2* (2002).

[119] Mangard, S., Oswald, E., and Popp, T. *Power analysis attacks - revealing the secrets of smart cards*. Springer, 2007.

[120] Maurer, U. M. A provably-secure strongly-randomized cipher. In *EUROCRYPT* (1990), pp. 361–373.

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

[122] Maurer, U. M., and Wolf, S. Secret-key agreement over unauthenticated public channels III: Privacy amplification. *IEEE Transactions on Information Theory 49*, 4 (2003), 839–851.

[123] Messerges, T. S., Dabbish, E. A., and Puhl, L. Method and apparatus for preventing information leakage attacks on a microelectronic assembly. Technical Report, US patent 6,295,606, September 2001.

[124] Micali, S., and Reyzin, L. Physically observable cryptography (extended abstract). In Naor [126], pp. 278–296.

[125] Micciancio, D., Ed. *Theory of Cryptography, 7th Theory of Cryptography Conference, TCC 2010, Zurich, Switzerland, February 9-11, 2010. Proceedings* (2010), vol. 5978 of *Lecture Notes in Computer Science*, Springer.

[126] Naor, M., Ed. *Theory of Cryptography, First Theory of Cryptography Conference, TCC 2004, Cambridge, MA, USA, February 19-21, 2004, Proceedings* (2004), vol. 2951 of *Lecture Notes in Computer Science*, Springer.

[127] Naor, M., and Segev, G. Public-key cryptosystems resilient to key leakage. In Halevi [92], pp. 18–35.

[128] OpenSSL project webpage:. `http://www.openssl.org` accessed on 9.12.2011.

[129] Oswald, E., Mangard, S., Pramstaller, N., and Rijmen, V. A side-channel analysis resistant description of the AES S-box. In *FSE* (2005), H. Gilbert and H. Handschuh, Eds., vol. 3557 of *Lecture Notes in Computer Science*, Springer, pp. 413–423.

[130] Pfitzmann, B., Ed. *Advances in Cryptology - EUROCRYPT 2001, International Conference on the Theory and Application of Cryptographic Techniques, Innsbruck, Austria, May 6-10, 2001, Proceeding* (2001), vol. 2045 of *Lecture Notes in Computer Science*, Springer.

[131] Pietrzak, K. A leakage-resilient mode of operation. In *Advances in Cryptology - EUROCRYPT 2009, 28th Annual International Conference on the Theory and Applications of Cryptographic Techniques, Cologne, Germany,*

April 26-30, 2009. Proceedings (2009), A. Joux, Ed., vol. 5479 of *Lecture Notes in Computer Science*, Springer, pp. 462–482.

[132] Preneel, B., Ed. *Advances in Cryptology - EUROCRYPT 2000, International Conference on the Theory and Application of Cryptographic Techniques, Bruges, Belgium, May 14-18, 2000, Proceeding* (2000), vol. 1807 of *Lecture Notes in Computer Science*, Springer.

[133] Quisquater, J.-J., and Koene, F. Side channel attacks: State of the art, October 2002. [77].

[134] Quisquater, J.-J., and Samyde, D. Electromagnetic analysis (EMA): Measures and counter-measures for smart cards. In *E-smart* (2001), I. Attali and T. P. Jensen, Eds., vol. 2140 of *Lecture Notes in Computer Science*, Springer, pp. 200–210.

[135] Rabin, T., Ed. *Advances in Cryptology - CRYPTO 2010, 30th Annual Cryptology Conference, Santa Barbara, CA, USA, August 15-19, 2010. Proceedings* (2010), vol. 6223 of *Lecture Notes in Computer Science*, Springer.

[136] Rao, A. An exposition of Bourgain's 2-source extractor. *Electronic Colloquium on Computational Complexity (ECCC) 14*, 034 (2007). `http://eccc.hpi-web.de/eccc-reports/2007/TR07-034/index.html` accessed on 9.12.2011.

[137] Regev, O. On lattices, learning with errors, random linear codes, and cryptography. In *STOC '05: Proceedings of the 37th Annual ACM Symposium on Theory of Computing, Baltimore, MD, USA, May 22-24, 2005* (2005), H. N. Gabow and R. Fagin, Eds., ACM, pp. 84–93.

[138] Renner, R., and Wolf, S. Unconditional authenticity and privacy from an arbitrarily weak secret. In Boneh [24], pp. 78–95.

[139] Rivest, R. L. All-or-nothing encryption and the package transform. In *Fast Software Encryption, 4th International Workshop, FSE '97, Haifa, Israel, January 20-22, 1997, Proceedings* (1997), E. Biham, Ed., vol. 1267 of *Lecture Notes in Computer Science*, Springer, pp. 210–218.

[140] Samyde, D., Skorobogatov, S., Anderson, R., and Quisquater, J.-J. On a new way to read data from memory. In *SISW '02: Proceedings of the First International IEEE Security in Storage Workshop* (Washington, DC, USA, 2002), IEEE Computer Society, p. 65.

[141] Schaumont, P., and Tiri, K. Masking and dual-rail logic don't add up. In *CHES* (2007), P. Paillier and I. Verbauwhede, Eds., vol. 4727 of *Lecture Notes in Computer Science*, Springer, pp. 95–106.

[142] Schramm, K., and Paar, C. Higher order masking of the aes. In *CT-RSA* (2006), D. Pointcheval, Ed., vol. 3860 of *Lecture Notes in Computer Science*, Springer, pp. 208–225.

[143] Shaltiel, R. How to get more mileage from randomness extractors. In *CCC '06: Proceedings of the 21st Annual IEEE Conference on Computational Complexity* (Washington, DC, USA, 2006), IEEE Computer Society, pp. 46–60.

[144] Shamir, A., and Tromer, E. Acoustic cryptanalysis. on nosy people and noisy machines. A webpage: `http://tau.ac.il/~tromer/acoustic/` accessed on 9.12.2011.

[145] Shoup, V. On formal models for secure key exchange (Version 4). Technical report RZ 3120. IBM Research Zurich, 1999.

[146] Skorobogatov, S. P., and Anderson, R. J. Optical fault induction attacks. In Kaliski Jr et al. [102], pp. 2–12.

[147] Standaert, F.-X., Pereira, O., Yu, Y., Quisquater, J.-J., Yung, M., and Oswald, E. *Leakage Resilient Cryptography in Practice.* Springer, November 2010, pp. 105–139.

[148] Trevisan, L., and Vadhan, S. P. Extracting randomness from samplable distributions. In *FOCS '00: Proceedings of 41th Annual Symposium on Foundations of Computer Science, 12-14 November 2000, Redondo Beach, California, USA* (2000), pp. 32–42.

[149] Trichina, E. Combinational logic design for AES subbyte transformation on masked data. Technical Report, IACR report, 2003.

[150] Trichina, E., De Seta, D., and Germani, L. Simplified adaptive multiplicative masking for aes. In Kaliski Jr et al. [102], pp. 187–197.

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

[152] Vazirani, U. V. Strong communication complexity or generating quasi-random sequences form two communicating semi-random sources. *Combinatorica 7*, 4 (1987), 375–392.

[153] VON NEUMANN, J. Various techniques used in connection with random digits. *J. Research Nat. Bur. Stand., Appl. Math. Series 12* (1951), 36–38.

[154] WAGNER, D., Ed. *Advances in Cryptology - CRYPTO 2008, 28th Annual International Cryptology Conference, Santa Barbara, CA, USA, August 17-21, 2008. Proceedings* (2008), vol. 5157 of *Lecture Notes in Computer Science*, Springer.

[155] WEGENER, I. *The Complexity of Boolean Functions*. John Wiley and Sons Ltd, and B. G. Teubner, 1987. See `http://ecc.hpi-web.de/static/books/The_Complexity_of_Boolean_Functions/` accessed on 9.12.2011.

[156] WIENER, M. J., Ed. *Advances in Cryptology - CRYPTO '99, 19th Annual International Cryptology Conference, Santa Barbara, California, USA, August 15-19, 1999, Proceedings* (1999), vol. 1666 of *Lecture Notes in Computer Science*, Springer.

[157] WIKIPEDIA. Internet socket. `http://en.wikipedia.org/wiki/Internet_socket` accessed on 9.12.2011.

[158] WOLF, S. Strong security against active attacks in information-theoretic secret-key agreement. In *ASIACRYPT* (1998), K. Ohta and D. Pei, Eds., vol. 1514 of *Lecture Notes in Computer Science*, Springer, pp. 405–419.

[159] YU, Y., STANDAERT, F.-X., PEREIRA, O., AND YUNG, M. Practical leakage-resilient pseudorandom generators. In *ACM Conference on Computer and Communications Security* (2010), E. Al-Shaer, A. D. Keromytis, and V. Shmatikov, Eds., ACM, pp. 141–151.

[160] YUNG, M., Ed. *Advances in Cryptology - CRYPTO 2002, 22nd Annual International Cryptology Conference, Santa Barbara, California, USA, August 18-22, 2002, Proceedings* (2002), vol. 2442 of *Lecture Notes in Computer Science*, Springer.