Leakage-Resilient Cryptography: A Survey of Recent Advances
RESEARCH EXAM, Spring 2010

Petros Mol

UC San Diego
pmol@cs.ucsd.edu

Abstract. Side-channel attacks represent a very frequent and severe type of attack against implementations of cryptographic protocols. Most countermeasures proposed until recently are ad-hoc, offer only partial remedy and fail to capture the problem in its entirety. In light of this, the last few years the cryptographic community has tried to set the theoretical foundations in order to formally address the problem of side-channel attacks. These efforts led to the development of Leakage Resilient Cryptography the goal of which is to design cryptographic protocols that remain secure in the presence of arbitrary, yet bounded, information about the secret key. In this survey, we review recent advances towards this direction. We first present an abstract and general framework that captures a wide range of side-channel attacks. We then present some of the most influential models from the literature as special cases of the general framework and describe how standard (leakage-free) security notions translate in the presence of leakage. Finally, we discuss the extent to which practical attacking scenarios are captured by the existing models and suggest some interesting directions for future research.

1 Introduction

The problem. In traditional Cryptography, primitives are treated as mathematical objects with a predefined (well-restricted) interface between the primitive and the user/adversary. Based on this view, cryptographers have constructed a plethora of cryptographic primitives (CPA/CCA secure encryption schemes, identification schemes, unforgeable signatures, etc.) from various computational hardness assumptions (factoring, hard lattice problems, DDH, etc.). These primitives are provably secure in the following sense: if there exists an efficient adversary that breaks primitive Π with some ("sufficiently large") probability, then one can use this adversary to solve a problem $\mathcal{P}$ which is considered hard. Given the state of the art for efficient algorithms, we have enough confidence that the problem $\mathcal{P}$ cannot be solved efficiently which in turn implies that no efficient adversary for Π exists.

Cryptography, however, should be developped for actual deployment in real-world applications and not solely for theoretical purposes. In this new setting, the actual interaction between the primitive and the adversary depends not only on the mathematical description of the primitive, but also on its implementation and the specifics of the physical device on which the primitive is implemented. The information about the primitive leaked to the adversary goes well beyond that predicted by the designer and, accumulatively, can allow the adversary break an, otherwise secure, primitive. Several examples of such "side-channel" attacks exist in the literature (see the main types below). We emphasize that such dangers stem from the fact that the existing models fail to capture adversaries that attack the *implementation* (as opposed to the abstract logic) of a primitive.

In search of countermeasures, one can try to prevent side-channel attacks by modifying the implementation or securing hardware. This leads to a trial and error approachwhere an implementation is made secure against a certain type of attack only before a new more effective attack appears. Leakage Resilient Cryptography adopts a different viewpoint by trying to provide provably secure primitives in the presence of a wide range of side-channel information.

Types of Side Channel Attacks. Below we give a brief description of the most common types of side-channel attacks along with some countermeasures proposed in the literature. For more details the reader is reffered to [11, 20] and the references therein.

- *Power Analysis Attacks:* In this kind of attacks, the adversary gets side information by measuring the power consumption of a cryptographic device. In cryptographic implementations where the execution path depends on the processed data, the power trace can reveal the sequence of instructions executed and hence leak information about the secret key. Various examples of power analysis attacks are given by Kocher et. al. [15].
- *Timing Attacks:* Here the adverary uses the running time of the execution of a protocol as side-channel information. The adversary knows a set of messages[1] as well as the running time the cryptographic device needs to process them. He can use these running times and potentially (partial) knowledge of the implementation in order to derive information about the secret key. The reader is reffered to [14] for examples of timing against known cryptographic primitives.
- *Fault Injection Attacks:* These attacks fall into the broader class of tampering attacks. The adversary forces the device to perform erroneous operations (i.e. by flipping some bits in a register) . If the implementation is not robust against fault injection, then an erroneous operation might leak information for the secret key.
- *Memory Attacks:* This type of attack was recently introduced by Halderman et. al. [10]. It is based on the fact that DRAM cells retain their state for long intervals even if the computer is unplugged. Hence an attacker with physical access to the machine can read the content of a fraction of the cells and deduce useful information about the secret key. The implications of the attack are even more severe due to the fact that the DRAM bits default to known "ground state". Halderman et. al. studied the effect of these attacks against DES,AES and RSA as well as against known Disk Encryption Systems.

Implementation-level countermeasures. The most common approach for protecting against power analysis attacks is to decrease the Signal-to-Noise ratio (SNR). The goal here is to somehow obfuscate the computations taking place so that the adversary gets much less useful information. Some possible ways to do that is by performing random power consuming operations or by randomizing the execution sequence.

For timing attacks, the most common approach is to make the execution time independent of the secret key. Another way to protect against timing attacks is by blinding the input; informally, this can be done by first randomizing the input (via an efficient and invertible randomized transformation), then perform operations on the randomized inputs and then return the correct output by "undoing" the transformation. The benefit of this approach is that it decorrelates the inputs from the intermediate values and computations. However, both approaches result in significantly slower implementations.

Protecting against fault injection attacks is usually achieved by performing consistency checks. For example, one can run the algorithm (or some steps of the it) twice and check whether the results coincide. The importance of these consistency checks have been highlighted in [2]. Again, this protection comes at the cost of increasing the running time (or even the hardware components that are used).

Finally, the effects of memory attacks can be mitigated by obfuscating the key (by either encrypting memory contents or by applying a transfornation to the part of memory that contains the key). Another possible countermeasure is to avoid precomputations or redundancies (this of course

[1] message in this context does not necessarily mean plaintext; it might as well be a ciphertext, a signature or any other message (string) processed during the execution of the protocol.

won't prevent the attack altogether, it will only reduce its severity). Again the countermeasures incur significant performance slowdown.

Leakage Resilient Cryptography. The preceding analysis indicates that countermeasures against side-channel attacks based on hardware or implementation-level solutions are usually ad-hoc (lack formal security analysis), costly both in terms of deployment and in terms of performance and can only provide partial remedy to the problem. Usually they just result in increasing the running time or equivalently decreasing the efficiency of the attack and not in preventing it altogether. The hardware/implementation-level solutions target a specific side-channel attack and need to be revised and fixed every time a more efficient side-channel attacks appears.

In this survey we focus on software-level solutions by considering cryptographic protocols that provably resist some of the aforementoned side-channel attacks. Leakage Resilient Cryptography doesn't concern itself with preventing side channel attacks. Trying to predict all different ways an attacker can gain side-channel information and prevent him by securing the hardware or changing the implementation seems like an unachievable goal. Leakage resilient cryptography on the other hand takes for granted the existence of side-channel information and tries to build primitives that provably withstand attacks where such information is availible. In order to capture many possible side-channel adversaries and abstract out the details of the hardware or the implementation, side-information is abstractly modeled as a function f that accounts for all the leakage information the adversary has access to. The main theoretical question of leakage resilient cryptography is the following: for which families of functions $\mathcal{F}$ and which primitives, one can provide constructions that provably tolerate any side-channel attack whose leakage can be modeled as a function $f \in \mathcal{F}$. From a practical point of view, we would like the constructions to be secure with respect to as large a class of functions as possible that at the same time capture many realistic attacks. On the other hand, in order for leakage resilient cryptography to have practical implications, we need to get a better understanding of which leakage functions arise in practice and also be able to design hardware whose leakage can be tolerated by our constructions.

Roadmap. In Section 2 we give some mathematical notation and background. Section 3 describes a general framework for modeling side-channel attacks abstractly and presents the main models derived as special restrictions of this general framework along with a comparison between them. The subsequent sections present concrete security definitions and an overview of the construction of some cryptographic primitives in each of the derived models . In particular, Section 4 gives an overview of available cryptographic primitives in the continuous leakage model, Section 5 reviews security against memory attacks whereas Section 6 studies security in the presence of auxiliary inputs. We conclude in Section 7 with a summary and some interesting future research directions.

2 Preliminaries

Notation. Let X be a random variable, $\mathcal{X}$ a set (domain) and $\mathcal{D}_X$ a distribution. Then $X \overset{\mathcal{D}_X}{\leftarrow} \mathcal{X}$ indicates that X is distributed over $\mathcal{X}$ according to $\mathcal{D}_X$. In particular $X \overset{\$}{\leftarrow} \mathcal{X}$ means that X is chosen uniformly at random from $\mathcal{X}$ whereas $X \leftarrow Y$ denotes assignment. We also use $negl(n)$ to denote the set of all functions that are negligible in n, that is

$$negl(n) = \{f(n) \,\big|\, \forall \text{ polynomial } p,\ \exists\, n_0 \text{ s.t. } \forall n > n_0,\ f(n) < \frac{1}{p(n)}\}$$

Finally $poly(n)$ denotes the set of functions that are upper bounded by a polynomial $p(n)$.

Probability Background. We review quickly some basic facts from probability. Let X, Y be two (discrete) random variables distributed over a countable set $\mathcal{V}$ according to $\mathcal{D}_X$ and $\mathcal{D}_Y$ respectively. The *statistical distance* between X and Y (or between $\mathcal{D}_X$ and $\mathcal{D}_Y$) is defined as

$$\Delta(X,Y) = \frac{1}{2}\sum_{v\in\mathcal{V}} |\Pr[\,X = v\,] - \Pr[\,Y = v\,]|$$

One important fact that stems directly from the above defintion is that the statistical distance cannot increase by applying a (possibly randomized) function f , that is

$$\Delta(f(X), f(Y)) \leq \Delta(X,Y) \tag{1}$$

Since we are interested in the asymptotic characteristics of distributions we extend the above definitions to probability ensembles. For two random variable ensembles $\mathcal{X} = \{X_\lambda\}_{\lambda\in\mathbb{N}}$ and $\mathcal{Y} = \{Y_\lambda\}_{\lambda\in\mathbb{N}}$ indexed by a (security) parameter λ, we say that $\mathcal{X}$ and $\mathcal{Y}$ are *statistically indistinguishable* (and denote $\mathcal{X} \overset{s}{\approx} \mathcal{Y}$) if $\Delta(X_\lambda, Y_\lambda) = negl(\lambda)$. Likewise, we say that $\mathcal{X}$ and $\mathcal{Y}$ are computationally indistinguishable (and denote $\mathcal{X} \overset{c}{\approx} \mathcal{Y}$) if

$$|\Pr[\,\mathcal{A}(X_\lambda) = 1\,] - \Pr[\,\mathcal{A}(Y_\lambda) = 1\,]| = negl(\lambda)$$

for any Probabilistic Polynomial Time (PPT in short) algorithm $\mathcal{A}$ (where the probability is taken over the randomness of $\mathcal{A}$ and the random variables X_λ, Y_λ).

For a random variable X defined over a domain $\mathcal{X}$, we define its *min-entropy* as

$$\mathrm{H}_\infty(X) = -\log(\max_{x\in\mathcal{X}} \Pr[\,X = x\,]).$$

where $\max_{x\in\mathcal{X}} \Pr[\,X = x\,] = 2^{-\mathrm{H}_\infty(X)}$ denotes the *predictability* of the random variable X.

Another useful notion of entropy is the *average min-entropy* (defined in [6]) of a random variable X (given Y) which is defined as follows:

$$\tilde{\mathrm{H}}_\infty(X|Y) = -\log\big(\mathop{\mathbf{E}}_{y\leftarrow Y}\big[2^{-\mathrm{H}_\infty(X\,|\,Y=y)}\big]\big) = -\log\big(\mathop{\mathbf{E}}_{y\leftarrow Y}\big[\max_{x\in\mathcal{X}} \Pr[\,X = x \mid Y = y\,]\big]\big)$$

The average min-entropy expresses the average maximum probability of predicting X given Y.

Cryptography Background. We now review some basic definitions from Cryptography.

Definition 1 (Public-Key Encryption Scheme). *A public-key encryption scheme is a triplet $\mathcal{AE} = (\mathcal{K}, \mathcal{E}, \mathcal{D})$ of PPT algorithms. $\mathcal{K}$, on input the security parameter 1^λ, outputs a pair of keys (pk, sk). $\mathcal{E}$ gets as its input the public key pk and a message $m \in \mathcal{M}$ (for some message space $\mathcal{M}$) and outputs a ciphertext c. The decryption algorithm $\mathcal{D}$ on input the secret key sk and a ciphertext c, outputs a message m or $\perp$ (failure). It is required that $\Pr[\,\mathcal{D}(sk, \mathcal{E}(pk, m)) \neq m\,] = negl(\lambda)$, where the probability is taken over the randomness of $\mathcal{K}, \mathcal{E}$ and $\mathcal{D}$.*

Definition 2 (Pseudorandom Function (PRF)). *Let $R_{n,m}$ be the set of functions with domain $\{0,1\}^n$ and range $\{0,1\}^m$. Let also $F : \{0,1\}^k \times \{0,1\}^n \to \{0,1\}^m$ be an efficient, keyed function. F is said to be pseudorandom if $\forall$ PPT distinguisher D and all polynomials $p()$, for sufficiently large n's*

$$\left|\Pr\left[\,D^{F(K,\cdot)}(1^n) = 1\,\right] - \Pr\left[\,D^{r(\cdot)}(1^n) = 1\,\right]\right| \leq \frac{1}{p(n)} \tag{2}$$

where $K \overset{\$}{\leftarrow} \{0,1\}^k$ and $r \overset{\$}{\leftarrow} R_{n,m}$.
If (2) holds when D is given query access to the oracle (either the F oracle or the r oracle), we say that F is a **strong** *PRF.*
If on the other hand (2) holds only for distinguishers D that are given outputs of the oracle on random inputs $x \in \{0,1\}^n$, we say that F is a **weak** *PRF.*

3 Modeling Side-Channel Attacks

Even though side-channel attacks against cryptographic protocols have appeared in practice for more than twenty years, it was only a few years ago that the theoretical basis for a formal treatment was set. In their seminal paper, Micali and Reyzin [16] give a general framework for modeling side-channel attacks. Their goal was to give a model that captures the strongest and most flexible[2] adversaries while leaving some room for the deployment of cryptography. A main contribution of their model is the separation of the logic of a primitive and its implementation which allows for a distinction between physical and non-physical attacks. Accordingly, the security of a primitive is considered against adversaries that have 2 components: (a) a computational component (against the logic of the primitive) and (b) a physical component that accounts for the collection of the side-channel information.

In what follows, we present (a slight relaxation of) the Micali-Reyzin model. We try to give as general a framework as possible and subsequently present the most popular models appearing in the literature as special cases of the general framework. Following [16], for an implementation $\mathcal{I}$, we use a function $f_{\mathcal{I}}(\cdot)$ to model the leakage (side-information) available to the adversary. $f_{\mathcal{I}}$ is implementation specific and is a function of (i) the secret (internal) state S of the protocol, (ii) an input Y provided by the adversary (usually Y determines the information the adversary wants to collect) and (iii) randomness R. The last input corresponds to the environmental noise and is outside the control of the adversary or the designer. If the overall computation takes m steps, then at step i, the adversary can get $f_{\mathcal{I}}(S_i, Y_i, R_i)$. For ease of presentation, we will omit R and consider instead a randomized function f. Also Y_i can be omitted by incorporating it to the description of the function. Since we will be treating general classes of functions, we can omit the index $\mathcal{I}$ (in this case any claim for a family of functions $\mathcal{F}$ can be interpreted as "any implementation whose leakage function belongs in $\mathcal{F}$...."). So we will write $f_{Y_i}(S_i)$ (or simply $f_i(S_i)$) instead of $f(S_i, Y_i, R_i)$. We note here that in order for f to be meaningful for cryptography, it must be the case that there exist physical machines (and implementations) whose leakage function doesn't completely leak (at least computationally) the whole secret internal state. Since this is mainly a hardware issue, Micali and Reyzin state this explicitly as a physical assumption.

Assumption 1 (Fundamental Physical Assumption [16]) *There exist physical implementations of cryptographic primitives whose leakage function doesn't reveal the entire internal state of the primitive.*

3.1 Models

Given the above formalization for the leakage, we can describe the main theoretical models for side-channel atacks that appear in the literature as special cases of the Micali-Reyzin framework. Each model can be derived by adding restrictions on either the input of the function, the output of the function or the function itself (by restricting the family $\mathcal{F}$ in which it belongs). Regarding the set (family) $\mathcal{F}$ of functions that f belongs to, in this survey we only consider *arbitrary* functions (some examples of side-channel attacks against restricted functions can be found in [3, 12]). The only restriction we put on the leakage function f is that it is computable in time polynomial in its input size.[3] As for the size of the output of f, it can either by bounded (and even smaller than the size of the secret material) or unbounded (this models continuous leakage). Also the effective input to the leakage function f can be either the entire secret state S or part of the secret state.

[2] As we will see later, some of the axioms introduced by Micali and Reyzin restrict the types of attacks captured by their model to a subset of the attacks that might be encountered in practice.

[3] Given the practical attacking scenarios modeled by a leakage function, this restriction seems natural.

Based on the restrictions imposed on (input/output of) the leakage function, we present below the most influential models that have appeared in the literature the last few years. Each model has its own strengths and weaknesses and is appropriate for specific attacking scenarios and inadequate for others.

Unbounded (continuous) computational-leakage In this setting the overall leakage might be unbounded; In particular, it can be larger than the size of the secret state. The overall execution of the cryptographic protocol is splitted into rounds. Before the ith round an attacker chooses (adaptively) a leakage function f_i and after the execution of the round, it receives $f_i(S_{i-1})$ (where S_{i-1} is the internal state right before the execution of the ith round). In order for the models to be useful for cryptographic constructions, the output of f_i per round should be bounded (even though the overall leakage accross all rounds can be unbounded). Another necessary restriction on the leakage function is that it should respect one of the axioms of the Micali-Reyzin model [16], namely "Only Computation Leaks Information" (henceforth denoted OCLI). More specificaly, at each round i, the internal state S_i can be partitioned into S_i^a and S_i^u where S_i^a (resp. S_i^u) denotes the part of the secret state that is accessed (resp. not accessed) during the computation at round i. In this model, the leakage function takes as input only S_i^a and hence no information on the unaccessed part of the state can leak to the adversary.

This model captures successfully side-channel attacks in which leakage results from computation. Practical attacks of this type include power analysis, eletromagnetic radiation and timing attacks (see previous section for more details on these attacks). Constructions of cryptograhic primitives secure in this model include stream ciphers [7, 19], a block cipher [18] and signature schemes [8]. We give the high-level ideas underlying some of these constructions in Section 4.

Memory-Leakage Attacks The results of Halderman et al. [10] suggests that the axiom "Only Computation Leaks Information" falls short of capturing a wide range of devastating attacking scenarios. In these attacks (henceforth termed Memory-attacks following [1]) information about the entire secret state can leak even if no computation takes place. Conceivably, no construction can provide meaningful security unless one further restricts the leakage function. The restriction imposed is that the overall size of the output of the leakage function f (taken over all rounds of execution) is upper bounded by a parameter ℓ (the leakage) which should be, roughly speaking, smaller than the size of the secret material S.[4] Constructions secure against memory attacks are stateless and include private and public-key encryption schemes [9, 1, 17], IBE schemes [1] as well as signature schemes [13]. Section 5 gives an overview of constructions proven secure in this model.

Leakage as Auxiliary Inputs Dodis, Kalai and Lovett [5] study the security of cryptographic primitives in the presence of auxiliary inputs. In this attacking scenario, the output of the leakage function cannot only be larger than the size of the secret key but can also fully reveal it *information-theoretically.* Also leakage can happen even in the absence of computation. In that sense, the auxiliary input model seems more general than the previous two models. However, the requirement on the leakage function is that it is exponentially hard[5] for any PPT adversary to recover S given $f(S)$ (see Section 6 for a more formal treatment.) Costructions of secure primitives in the auxiliary input model include the private schemes from [5, 9] and the public key schemes from [4].

[4] Strictly speaking, as pointed out by Naor and Segev [17], a sufficient requirement is that S is not revealed *information-theoretically* given $f(S)$ even if $|f(S)| > |S|$.

[5] The exponential hardness requirement has been somewhat relaxed in [4].

3.2 Comparison/Discussion

The main formalizations presented above are incomparable in the sense that no formalization is a strict generalization of another. Each model might or might not be appropriate depending on the practical attack scenario in consideration. For instance, the continuous leakage model captures computational side-channel attacks where collection of side information happens accumulatively accross multiple rounds of execution of a protocol. However, it fails to capture memory attacks. All the attacks proved in the continuous leakage model rely crucially on the fact that parts of the secret key cannot be leaked if they don't contribute to the computation. On the other hand, the memory attacks model requires that the total number of bits leaked during the attack is upper bounded by the length of the secret key. Auxiliary inputs model relaxes somewhat this requirement by allowing leakage functions with output longer than the secret key size. From a proof-technique viewpoint, the auxiliary input model is the computational analogue of the memory attacks model. Instead of remaining (statistical) entropy which is the requirement in the memory attacks model, the auxiliary input model requires (roughly speaking) that the secret key has high computational entropy even given $f(S)$. This translates to the *computational* assumption that $f(S)$ is exponentially hard to invert. We note here that a drwaback of the auxiliary inputs model is that given a function f modelling the leakage of a piece of hardware, there is no way to check whether f belongs to the set of functions that are (exponentially) hard to invert.

3.3 Defining Security in the Presence of Side-Channel Attacks

In order to construct primitives that are secure against side-channel attacks, we first need to formally define what security in the presence of side-channels means. Informally, almost all definitions in the presence of leakage follow from their leakage-free counterparts by extending the adversary that attacks the primitive to a stronger or, more precisely, a more informed one. The security definition, modeled as a game (interaction or experiment), considers an adversary which, besides the information normally given to him through the interface considered in the original (leakage-free) security game, gets in addition the outputs of the leakage functions $f_i(s)$ which he can adaptively specify during the attack. We consider a primitive secure if the adversary, in this new interaction, has "small" success probability in "breaking" the primitive.[6] In the following sections, we give define formally a few important cryptographic primitives in each leakage model.

4 Security Against Continuous Leakage Attacks

In this setting, we consider stateful cryptographic protocols that execute in rounds. After the execution of the i-th round, the adversary besides the information naturally given to him via the standard interaction, gets to see the output of a leakage function f_i (adaptively chosen by him) on input state S_{i-1} (and possibly randomness r_i if the primitive is randomized). Again, security against arbitrary leakage functions can be achieved only if $|f_i(S_{i-1})| < |S_{i-1}|$. However, unlike memory attacks, the overall leakage across all rounds of execution can be much larger than the size of the state. This model seems to give too much power to the adversary. Indeed consider a deterministic stateful primitive and assume that the output of f_i is just a single bit. Let M be an upper bound on the bitsize of the internal state. The adversary before the ith step adaptively chooses as the leakage function the one that computes state S_M and outputs its ith bit (notice here that given the *entire* state S_{i-1} as input, a function f can fully compute S_M). After M rounds the adversary can fully recover the entire state S_M at which point the security of the primitive is

[6] "small" and "breaking" take different meanings depending on the specific primitive under consideration.

completely broken. We thus need to add an extra requirement, namely the domain of f_i at each round i is the *active* part S_{i-1}^a of S_{i-1}, that is the part of the state that is actually used in order to compute S_i from S_{i-1}. This requirement follows the "Only Computation Leaks Information" axiom of Micali and Reyzin.

In the continuous leakage model, folclore primitives from traditional cryptography might no longer be secure. Such an example is (strong) pseudorandom functions (PRFs, see Definition 2) Consider a PRF $F : \{0,1\}^k \times \{0,1\}^n \to \{0,1\}^m$ and let $f : \{0,1\}^k \to \{0,1\}^\ell$ be the leakage function. An adversary can distinguish $F(K, X)$ from $R(X)$ (for a random function $R : \{0,1\}^n \to \{0,1\}^m$) by querying the leakage oracle with $f(K)$ that outputs the last ℓ bits of $F(K, X)$.[7] Pietrzak [19] showed that if one considers *weak* PRFs (where the adversary gets $F(K, X)$ for randomly chosen X) pseudorandomness can be proved in the presence of bounded leakage (with somewhat worse security guarantees due to the leakage). The leakage resilience of weak PRFs, allows one to use them as a building block for more advanced primitives. One such primitive, namely leakage-resilient stream ciphers, was presented by Pietrzak [19] (this essentially simplifies the stream-cipher construction of Dziembowski and Pietrzak [7]).

The first public-key primitive resilient to continuous leakage in the standard model was constructed by Faust et. al. [8]. In particular, the authors construct stateful leakage resilient signatures that can withstand a bounded amount of arbitrary leakage per invocation. The building block of their construction is a 3-time signature scheme that is secure against bounded memory attacks (i.e. [13]) that tolerates ℓ_{total} bits of *overall* leakage. Faust. et. al. present a tree-based construction that transforms any such signature scheme to a stateful signature scheme that can tolerate $\frac{\ell_{total}}{3}$ bits of leakage *per invocation*. When instantiated with a scheme from [13], the aforementioned tree-based construction yields an efficient signature scheme that can resist $\left(\frac{1}{36} - \epsilon\right) |S_i^a|$ bits of leakage per invocation (where $|S_i^a|$ is the active state at each invocation).

5 Security Against Bounded Memory Attacks

In this section, we present formal definitions of security against memory leakage attacks. Recall that in memory attacks, the leakage function can be any *arbitrary polynomially computable* function that takes as input the entire secret information.[8] The only restriction that we put is that the information meant to be hidden from the attacker, is not completely revealed *information-theoretically*. Formally, if s denotes the secret information and f is the leakage function, we require that $\mathrm{H}_\infty(s \mid f(s))$ is sufficiently large.

Akavia et. al. [1] were the first to define security against leakage attacks in the public key setting. Their definition is a natural extension of the standard notion of indistinguishability under chosen-plaintext-attacks (IND-CPA). Roughly, the adversary, in addition to all the information provided in the standard IND-CPA interaction, gets access to a leakage oracle that, on input an arbitrary function f_i returns $f_i(sk)$. If the adversary queries the leakage oracle Q times by providing functions $f_1, f_2, ..., f_Q$, the bounded leakage requirement translates to

$$\left| \sum_{i=1}^{Q} f_i(sk) \right| \leq \ell \qquad \text{for some bound } \ell$$

[7] Notice that this attack exploits the fact that in the security game for strong PRFs, the adversary can give an input X of his choice.

[8] In this section we will only define stateless primitives. Secret information in this setting means either the secret key alone (for public-key encryption schemes) or the secret key and the used randomness (for signature schemes).

Depending on whether the leakage functions are chosen by the adversary before or after he sees the public key pk, Akavia et. al. distinguish between non-adaptive (also termed weak key-leakage attack in [17]) and adaptive memory attacks respectively. Below we give a formal definition.

Let $\mathcal{O}_{sk}(\cdot)$ be a leakage oracle that takes as input a function f with domain the secret key space $\mathcal{SK}$ and range $\{0,1\}^*$. An adversary $\mathcal{A}$ has two components (corresponding to phases of the attack). In the first phase (component $\mathcal{A}_1$), the adversary is given access to a leakage oracle $\mathcal{O}_{sk}(\cdot)$. In the second phase (component $\mathcal{A}_2$) he tries to guess which bit the challenge ciphertext corresponds to (see Table 1). We say that an adverary $\mathcal{A}$ is ℓ-bounded if the sum of the output lengths that the leakage oracle returns on inputs the f_i's is at most ℓ.

Definition 3 (Chosen-plaintext memory attacks). *Let $\Pi = (\mathcal{K}, \mathcal{E}, \mathcal{D})$ be a public-key encryption scheme. We say that Π is ℓ-NAMA-CPA (resp. ℓ-AMA-CPA) secure if $\forall$ PPT ℓ-bounded adversary $\mathcal{A} = (\mathcal{A}_1, \mathcal{A}_2)$ the advantage of $\mathcal{A}$ in the left (resp. right) game of Table 1 is negligible in the security parameter, where the advantages are defined as*

$$\mathbf{Adv}_{\Pi}^{\text{nama-cpa}}(\mathcal{A}) = \left| \Pr\left[\, \mathcal{A} \textit{ wins NAMA game} \,\right] - \frac{1}{2} \right| \quad \textit{and}$$
$$\mathbf{Adv}_{\Pi}^{\text{ama-cpa}}(\mathcal{A}) = \left| \Pr\left[\, \mathcal{A} \textit{ wins AMA game} \,\right] - \frac{1}{2} \right|$$

Non-adaptive memory attacks (NAMA)	Adaptive memory attacks (AMA)
1. $(sk, pk) \leftarrow \mathcal{K}(1^n)$.	1. $(sk, pk) \leftarrow \mathcal{K}(1^n)$.
2. $(m_0, m_1, state) \leftarrow \mathcal{A}_1(pk, f(sk))$ $(\lvert m_0\rvert = \lvert m_1\rvert)$	2. $(m_0, m_1, state) \leftarrow \mathcal{A}_1^{\mathcal{O}_{sk}(\cdot)}(pk)$ $(\lvert m_0\rvert = \lvert m_1\rvert)$
3. $c^* \leftarrow \mathcal{E}_{pk}(m_b)$	3. $c^* \leftarrow \mathcal{E}_{pk}(m_b)$
4. $b' \leftarrow \mathcal{A}_2(c^*, state)$	4. $b' \leftarrow \mathcal{A}_2(c^*, state)$
5. if $b' = b$ return 1, else return 0	5. if $b' = b$ return 1, else return 0

Table 1. Definitions of IND-CPA security against memory attacks

Public-Key Encryption. Akavia et. al. [1] show that, under the LWE assumption, Regev's initial scheme [21] is secure against memory attacks with large amounts of leakage. Naor and Segev [17] observed that any semantically secure encryption scheme can be efficiently transformed to one that is secure against *non-adaptive* memory attacks where $|sk|(1 - o(1))$ bits of the secret key are leaked. They also present a generic and efficient construction of public-key encryption schemes that are resilient to adaptive memory attacks of up to $|sk|(1-o(1))$ bits. In addition, they extend the definitions of security in the chosen-ciphertext attack (CCA) setting. This is done in a straightforward way by providing the adversary with a decryption oracle in the first phase of the attack (step.2, Table 1) for CCA1 security or in both phases (steps 2 and 4,Table 1) for CCA2 security. The authors also provide two efficient constructions that are CCA1 and CCA2 secure where the leakage can be as large as $|sk|(\frac{1}{4} - o(1))$ and $|sk|(\frac{1}{6} - o(1))$ bits respectively.

Signature Schemes. Katz and Vaikuntanathan [13] extend the definitions of existential unforgeability under chosen-message attacks for signature to the leakage setting. Besides the standard queries to the signing oracle, the adversary can also query a leakage oracle on leakage functions f_i. Again the requirement hear is that the sum of the outputs from the leakage oracle is bounded by ℓ.

Interestingly [13] consider f_is that besides the secret key sk can also take as input the randomness used to produce the signatures. Signature schemes that are secure against these stronger types of attacks are called *fully* ℓ-leakage resilient (as opposed to (simple) ℓ-leakage resilient when the adversary can only get information about the secret key. The authors give constructions of both fully and simple leakage resilient schemes under standard assumptions (where the fully resilient schemes can withstand smaller amounts of leakage and allow the adversary only a limited-fixed in advance- number of queries to the signing oracle).

6 Security Against Auxiliary-Input Attacks

Auxiliary input attacks deviate from the memory attack setting in that the output of the leakage function f can be unbounded. However, it is still required that any PPT adversary can recever sk from $f(sk)$ with probability at most $2^{-\ell(\cdot)}$ for some function $\ell(\cdot)$. In the language of assumptions, the secret key is computationally unpredictable given $f(sk)$ whereas in the memory attack model, sk is needed to be information-theoretically hidden given $f(sk)$.

The study of lekage attacks in the auxiliary input model was initiated by Dodis et. al. [5] in the symmetric key setting. The leakage functions consider are those that are exponentially hard to invert.

Definition 4 (def.1 from [5]). *A polynomial time computable function $f : \{0,1\}^n \to \{0,1\}^*$ is exponentially hard to invert if there exists a constant c such that for every PPT adversary $\mathcal{A}$ and for sufficiently large n*

$$\Pr_{x \in \{0,1\}^n} [\, \mathcal{A}(f(x)) = x \,] \leq 2^{-cn}$$

The security definitions for symmetric encryption in the auxiliary input model are a straightforward modification of the corresponding definitions from the memory attacks model (see Section 5) and hence we omit to give them explicitly[9]. Instead of a bounded leakage function f, the adversary gets to see $f(k)$ for an exponentially hard to invert function f. Apart from defining security with auxiliary inputs , Dodis et al. provide the first constructions of CPA and CCA secure symmetric key encryption schemes that are secure in th epresence of exponentially hard-to-invert auxiliary inputs. The underlying hardness assumption of all their constructions is a generalization of the decision version of Learning Parity with Noise (LPN) problem.

In the public-key setting, security in the auxiliary input model was defined and studied by Dodis et. al. [4]. The definition again is a straightforward modification of the corresponding definition from the (adaptive) memory attack setting

Definition 5 (CPA with auxiliary inputs, [4]). *Let $\mathcal{F}$ be a family of functions and $\Pi = (\mathcal{K}, \mathcal{E}, \mathcal{D})$ a public-key encryption scheme. We say that Π is secure with respect to auxiliary inputs from $\mathcal{F}$ if $\forall$ PPT adversary $\mathcal{A} = (\mathcal{A}_1, \mathcal{A}_2)$ and function $f \in \mathcal{F}$ the advantage of $\mathcal{A}$ in the game of Table 2 is negligible in the security parameter, where the advantage is defined*

$$\mathbf{Adv}^{\text{ai-cpa}}_{\Pi}(\mathcal{A}) = \left| \Pr [\, \mathcal{A} \text{ wins } AUX \text{ game} \,] - \frac{1}{2} \right|$$

[9] Of course, since we are in the symmetric key setting, there is no public key and both $\mathcal{A}_1$ and $\mathcal{A}_2$ have access to an encryption oracle.

```
Auxiliary input attacks (AI)
1. $(sk, pk) \leftarrow \mathcal{K}(1^n)$.
2. $(m_0, m_1, state) \leftarrow \mathcal{A}_1(pk, f(sk, pk))$ $(|m_0| = |m_1|)$
3. $c^* \leftarrow \mathcal{E}_{pk}(m_b)$
4. $b' \leftarrow \mathcal{A}_2(c^*, state)$
5. if $b' = b$ return 1, else return 0
```

Table 2. Definition of IND-CPA security against auxiliary input attacks

In [4], they consider two new families of functions $\mathcal{F}$ parametrized by a function $\ell(k)$. Formally the new classes considered are defined as follows[10]

$$\mathcal{F}_{un}(\ell(k)) = \Big\{ f : \{0,1\}^{|pk|+|sk|} \to \{0,1\}^* \,\big|\, \forall\, PPT \text{ algorithm } \mathcal{I},\ \Pr[\, \mathcal{I}(f(sk,pk)) = sk \,] \leq \ell(k) \Big\}$$
$$\mathcal{F}_{pk-un}(\ell(k)) = \Big\{ f : \{0,1\}^{|pk|+|sk|} \to \{0,1\}^* \,\big|\, \forall\, PPT \text{ algorithm } \mathcal{I},\ \Pr[\, \mathcal{I}(f(sk,pk),pk) = sk \,] \leq \ell(k) \Big\}$$

where the probability is taken over the randomness of $\mathcal{K}(1^n)$ (and the internal randomness of $\mathcal{I}$). Given these families, the authors define $\ell(k)$-*AI-CPA* (resp. $\ell(k)$-*wAI-CPA*) secure encryption schemes as those that are secure with respect to leakage functions from $\mathcal{F}_{un}(\ell(k))$ (resp. $\mathcal{F}_{pk\text{-}un}(\ell(k))$)

Besides its mathematical niche, this formalization allows one to compare and relate security notions from memory leakage and auxiliary input model (see [4, Lemma 3.1]). In light of these new definitions, the authors provide constructions that are provably secure when the auxiliary input function is hard to invert with probability as high as $2^{-k^{\epsilon}}$ for any $\epsilon > 0$. The ideal in the auxiliary input model would be to prove security for auxiliary input functions that are only invertible with $\ell(k)$ probability by a PPT adversary for *any* $\ell(k) \in negl(k)$.

7 Conclusions

7.1 Summary

Designing primitives that are secure in the presence of leakage is a difficult but not impossible task. The last few years, the cryptographic community has put a lot of effort in constructing leakage resilient primitives. As the foundations for a theoretical treatment of the subject have been set, we expect that within the next years more and more leakage resilient primitives will be constructed that will tolerate richer and richer families $\mathcal{F}$ of leakage functions.

At this point it is instructive to ask how the development of this new field can affect practical cryptography. We believe that there is still lot of research to be done from the engineering and hardware design community so that the actual physical devices on which the protocols run, leak information within the range tolerated by existing provably leakage resilient constructions. Even if arguing about the latter seems a very hard task (after all, how can one prove or even claim that a smart card leaks up to ℓ bits ?), this approach leaves hardware designers with a much clearer engineering goal.

[10] f is always considered polynomial time computable.

7.2 Open Problems

One major question is how incorporating leakage resiliense to an implementation affects its performance. In practice, many implementations deviate from the mathematical description of the protocol for efficiency reasons. One such example is adding redundant information about the secret key in order to optimize operations. How is leakage resilience affected in the presence of redundant keys?

Another direction has to do with flexibility. In the vast majority of the available constructions either the design of the primitive or its security depend on the maximum tolerated leakage. Ideally, one would like to design a protocol that is leakage resilient but as secure and efficient as its leakage-free counterpart whenever no leakage takes place. This problem was addressed to some extent in [9]. The authors provide a symmetric encryption scheme whose parameters are independent of the maximum tolerated leakage. However, in the complete absense of leakage the security guarantees of the scheme are significantly worse than a scheme that is initially designed to be secure in the leakage-free setting.

References

1. Adi Akavia, Shafi Goldwasser, and Vinod Vaikuntanathan. Simultaneous Hardcore Bits and Cryptography against Memory Attacks. In Omer Reingold, editor, *TCC*, volume 5444 of *Lecture Notes in Computer Science*, pages 474–495. Springer, 2009.
2. Dan Boneh, Richard A. DeMillo, and Richard J. Lipton. On the Importance of Checking Cryptographic Protocols for Faults. In *EUROCRYPT'97: Proceedings of the 16th annual international conference on Theory and application of cryptographic techniques*, pages 37–51, Berlin, Heidelberg, 1997. Springer-Verlag.
3. Ran Canetti, Yevgeniy Dodis, Shai Halevi, Eyal Kushilevitz, and Amit Sahai. Exposure-Resilient Functions and All-or-Nothing Transforms. In *EUROCRYPT*, pages 453–469, 2000.
4. Yevgeniy Dodis, Shafi Goldwasser, Yael Tauman Kalai, Chris Peikert, and Vinod Vaikuntanathan. Public-Key Encryption Schemes with Auxiliary Inputs. In Daniele Micciancio, editor, *TCC*, volume 5978 of *Lecture Notes in Computer Science*, pages 361–381. Springer, 2010.
5. Yevgeniy Dodis, Yael Tauman Kalai, and Shachar Lovett. On Cryptography with Auxiliary Input. In *STOC*, pages 621–630, 2009.
6. Yevgeniy Dodis, Rafail Ostrovsky, Leonid Reyzin, and Adam Smith. Fuzzy Extractors: How to Generate Strong Keys from Biometrics and Other Noisy Data. *SIAM J. Comput.*, 38(1):97–139, 2008. Preliminary Version in EUROCRYPT 2004.
7. Stefan Dziembowski and Krzysztof Pietrzak. Leakage-Resilient Cryptography. In *FOCS*, pages 293–302, 2008.
8. Sebastian Faust, Eike Kiltz, Krzysztof Pietrzak, and Guy N. Rothblum. Leakage-Resilient Signatures. In *Theory of Cryptography, 7th Theory of Cryptography Conference, TCC 2010, Zurich, Switzerland, February 9-11, 2010. Proceedings*, pages 343–360, 2010.
9. Shafi Goldwasser, Yael Tauman Kalai, Chris Peikert, and Vinod Vaikuntanathan. Robustness of the Learning with Errors Assumption. 2010.
10. J. Alex Halderman, Seth D. Schoen, Nadia Heninger, William Clarkson, William Paul, Joseph A. Calandrino, Ariel J. Feldman, Jacob Appelbaum, and Edward W. Felten. Lest We Remember: Cold Boot Attacks on Encryption Keys. In Paul C. van Oorschot, editor, *USENIX Security Symposium*, pages 45–60. USENIX Association, 2008.
11. Erwin Hess, Norbert Janssen, Bernd Meyer, and Torsten Schtze. Information Leakage Attacks Against Smart Card. In *in EUROSMART Security Conference*, pages 55–64, 2000.
12. Yuval Ishai, Amit Sahai, and David Wagner. Private Circuits: Securing Hardware against Probing Attacks. In *CRYPTO*, pages 463–481, 2003.
13. Jonathan Katz and Vinod Vaikuntanathan. Signature Schemes with Bounded Leakage Resilience. In Mitsuru Matsui, editor, *ASIACRYPT*, volume 5912 of *Lecture Notes in Computer Science*, pages 703–720. Springer, 2009.
14. Paul C. Kocher. Timing Attacks on Implementations of Diffie-Hellman, RSA, DSS, and Other Systems. In *CRYPTO '96: Proceedings of the 16th Annual International Cryptology Conference on Advances in Cryptology*, pages 104–113, London, UK, 1996. Springer-Verlag.
15. Paul C. Kocher, Joshua Jaffe, and Benjamin Jun. Differential Power Analysis. In *CRYPTO '99: Proceedings of the 19th Annual International Cryptology Conference on Advances in Cryptology*, pages 388–397, London, UK, 1999. Springer-Verlag.
16. Silvio Micali and Leonid Reyzin. Physically Observable Cryptography (Extended Abstract). In Moni Naor, editor, *TCC*, volume 2951 of *Lecture Notes in Computer Science*, pages 278–296. Springer, 2004.
17. Moni Naor and Gil Segev. Public-Key Cryptosystems Resilient to Key Leakage. In Shai Halevi, editor, *CRYPTO*, volume 5677 of *Lecture Notes in Computer Science*, pages 18–35. Springer, 2009.
18. Christophe Petit, François-Xavier Standaert, Olivier Pereira, Tal Malkin, and Moti Yung. A Block Cipher Based Pseudo Random Number Generator Secure Against Side-Channel Key Recovery. In Masayuki Abe and Virgil D. Gligor, editors, *ASIACCS*, pages 56–65. ACM, 2008.
19. Krzysztof Pietrzak. A Leakage-Resilient Mode of Operation. In *EUROCRYPT*, pages 462–482, 2009.
20. Jean-Jacques Quisquater and Math Rizk. Side channel attacks: State of the art. Technical report, October 2002.
21. Oded Regev. On Lattices, Learning with Errors, Random Linear Codes and Cryptography. In *STOC*, pages 84–93, 2005.