A Survey of Leakage-Resilient Cryptography*

Yael Tauman Kalai
Microsoft Research, MIT
yael@microsoft.com

Leonid Reyzin
Boston University
reyzin@bu.edu

March 15, 2019

Abstract

In the past 15 years, cryptography has made considerable progress in expanding the adversarial attack model to cover side-channel attacks, and has built schemes to provably defend against some of them. This survey covers the main models and results in this so-called "leakage-resilient" cryptography.

1 Introduction

In most theoretical work on cryptography, parties are afforded complete privacy for their local computations. An adversary may, perhaps, be able to obtain a signature on a chosen plaintext or a decryption of a chosen ciphertext, but typically the signing or decryption process itself is assumed to be entirely hidden from the adversary. In particular, the only information correlated with the secret key that the theoretical adversary can obtain is typically confined to well-defined interfaces, such as signing or decrypting. Such an adversary is sometimes called a "black-box" attacker.

Work in modern cryptography—much of it pioneered by Shafi Goldwasser and Silvio Micali—demonstrated that it is possible to provably (based on certain computational complexity assumptions) defend against black-box attackers for large classes of cryptographic tasks, such as pseudorandom generation [BM82, BM84, GGM84, GGM86], encryption [GM82, GM84], signatures [GMR84, GMR88], zero-knowledge proofs [GMR85, GMR89, GMW86, GMW91], and secure multiparty computation [GMW87, BGW88].

Real adversaries, unfortunately, do not always respect such clean abstraction boundaries. A variety of successful *side-channel attacks* have demonstrated that information about the secret key and the internal state of a computation can leak out to a determined adversary. These attacks exploit the fact that every cryptographic algorithm is ultimately implemented on a physical device that affects the environment around it in measurable ways. To mention just a few prominent examples, attacks have exploited the time taken by a particular implementation of a cryptographic algorithm [Koc96], the amount of power consumed [KJJ99], or the electromagnetic radiation [AARR03]. So-called "cold boot" attacks [HSH+08, HSH+09] have been used to recover some fraction of a cryptographic secret key given physical access to a powered-off device. More

*This work will appear as a chapter in a forthcoming book titled Providing Sound Foundations for Cryptography: On the work of Shafi Goldwasser and Silvio Micali, edited by Oded Goldreich

recent attacks [LSG+18, KHF+19] allow processes to violate isolation boundaries and read information from other processes on the same machine — even those in secure enclaves [BMW+18]. In other words, the real adversary may not be black-box.

The emergence of side-channel attacks caused the cryptographic community to re-evaluate the black-box adversary model and to create new adversary models and provably secure designs. This line of work became known as "leakage-resilient cryptography." Shafi Goldwasser and Silvio Micali were again prominent in this effort, both because their past work on black-box security informed models for leakage-resilience, and because they themselves proposed models that formalize side-channel leakage and designed leakage-resilient schemes.

In this survey we cover some of the work on leakage-resilient cryptography. It is important to emphasize that our selection is biased toward more theoretical and foundational works. Even among those, our choices are necessarily biased by work we know. The field is vast and rapidly growing: as of Februrary 2019, Google Scholar finds over 400 papers with the phrase "leakage-resilient" or "leakage resilience" in the title, and about 2800 with the phrase "leakage-resilient" in the paper (98% of them published after 2006).

We do not address the vast literature dealing with adversaries who actively tamper with the memory or computation of the honest parties, rather than merely observe it (see, e.g., [GLM+04, IPSW06, DPW10, FPV11, LL12, FMVW14, JW15, FMNV15, DLSZ15]), even though it is, of course, connected to the literature on leakage resilience, and often includes leakage-resilience as one of its goals.

We apologize in advance to authors whose work we could not include and to readers who will be left to discover other work on their own.

Because leakage-resilient cryptography is a relatively young subset of cryptography, the gap between theory and practice is fairly large. This gap manifests itself in the debates about the practical relevance of theoretical models and the inefficiencies of provably secure constructions. This survey focuses on more theoretical work. An excellent source of more applied research in this field is the Conference on Cryptographic Hardware and Embedded Systems (CHES) and the journal IACR Transactions on Cryptographic Hardware and Embedded Systems (TCHES).

A bibliographic note: For most papers, we cite the conference version. In the few cases we are aware of the journal version, we cite it, as well. Many papers we cite have full versions that were too long to appear in conference proceedings, easily found through an on-line search, more often than not posted on `https://eprint.iacr.org`. These full versions sometimes correct errors that appeared in the conference version.

1.1 Early Works

Early works, such as work on oblivious RAM [GO96], threshold [DF90] and proactive [HJJ+97] cryptography, forward [Gün90, BM99] and intrusion-resilient [IR02] security, can be thought of, in hindsight, as works on leakage resilience. There are many other examples, too numerous to mention here.

We now elaborate on two particular lines of work. The first of these considers leakage of some of the bits of the secret key. The second one considers leakage during computation.

Leaking Bits from Keys Motivated by the problem of key exposure, Canetti et al. [CDH+00], followed by Dodis, Sahai, and Smith [DSS01], proposed an approach of storing a cryptographic key

in a redundant form, so that the key remains hidden even when some of the stored bits are leaked to the adversary. They introduced the notion of an "exposure-resilient function" and showed a connection to "all-or-nothing transforms" [Riv97, Boy99]. See [Dod00] for a detailed exposition of these results. These results were limited to leakage that consisted of subsets of bits of the stored secret, rather than more general functions of it.

This line of work was generalized by the long sequence of works on *memory leakage*, pioneered by Dziembowski [Dzi06], Di Crescenzo, Lipton, and Walfish [DLW06], and Akavia, Goldwasser, and Vaikuntanathan [AGV09], who considered *arbitrary* (poly-time computable) partial leakage from memory. We elaborate on these works in Section 1.2 and Section 2.

Leakage from Computation Chari et al. [CJRR99] considered a formal model of attacks in which every bit produced in a computation (i.e., every wire of a circuit) can be measured by the adversary, but each measurement has noise (their model was informed, in particular, by the differential power analysis attacks of [KJJ99]). Independently, Goubin and Patarin [GP99], also concerned about differential power analysis attacks, considered how to keep individual wire values in a smart-card circuit independent of the secret key. Both papers suggested the following countermeasure: represent each bit b by k random bits whose exclusive-or is equal to b (this approach is also known as XOR-secret sharing or boolean masking). Chari et al. [CJRR99] showed that, given the noisy reading of all k shares of b, the adversary can distinguish $b = 0$ from $b = 1$ only with advantage that is exponentially small in k. They did not, however, show how to compute on shared versions of bits. In contrast, Goubin and Patarin [GP99] showed how to compute certain functions using the shared versions of bits, but without a formal model in which to argue security.

Precise models and provable approaches to handling leakage from computation were pioneered by the works of Ishai, Sahai, and Wagner [ISW03] and Micali and Reyzin [MR04]. We discuss this line of work in Section 1.2 and Section 4.

1.2 Formalisms of Leakage-Resilient Cryptography

We coarsely divide the works on leakage-resilient cryptography into two strands. The first of these considers leakage from memory, while the second considers leakage during computation.

Memory Leakage In most common models of memory leakage, the adversary is usually allowed obtain an *arbitrary* polynomial-time computable but bounded-length leakage on the secret key. The goal is to build cryptographic schemes that remain secure even if this partial information about the secret key is available to the adversary.

Dziembowski [Dzi06] and Di Crescenzo, Lipton, and Walfish [DLW06] defined the term *bounded retrieval model*, which assumes that the adversary can obtain at most K bits of information about the secret key, for some (absolute, large) value K. The secret key is allowed to be larger than K, as long as the efficiency of the scheme is not negatively affected: the running times of the relevant algorithms should grow only polylogarithmically with K. They constructed leakage-resilient symmetric password and authentication protocols in this model.

Akavia, Goldwasser and Vaikuntanathan [AGV09] considered arbitrary leakage in the public-key setting. They considered the so-called *bounded memory leakage*, in which the amount of leakage is not an absolute value but rather is expressed as a function of the secret-key size (but growing the key is expensive, because the running times of the relevant algorithms can grow polynomially

with the key size). Public-key schemes in the bounded retrieval model of [Dzi06, DLW06] were also subsequently constructed [ADW09]. The bounded memory leakage model was later generalized to so-called *auxiliary input leakage* [DKL09]. In this model, leakage is not necessarily bounded in size: the only requirement is the minimum necessary for any security to remain, namely, that the secret should remain computationally hidden even given the leakage. Memory leakage was also generalized to the continual setting [BKKV10, DHLW10a], in which the secret key is periodically updated, *without updating the public key*, and it is assumed that there is bounded memory leakage within each time period, but there is no bound on the overall leakage.

We elaborate on this line of work in Section 2.

Computation Leakage The line of work on leakage from computation considers the situation in which side-channel information comes from the intermediate values created during a computation, rather than only from the secret itself. Sometimes memory leakage models discussed above can also model leakage of intermediate values created during a computation, because these values are just functions of the secret memory. However, this approach to modeling leakage from computation fails whenever secret randomness is used during a computation (though a few papers on memory leakage do model leakage from secret randomness; see Section 2 for details).

There are even more important distinctions between the models of memory and computation leakage. Memory leakage models most typically consider one-time leakage (but see Section 2.5 for exceptions), while computational leakage models typically consider continual leakage over multiple uses of the secret key, forcing constructions to update the secret memory in order to maintain security. On the other hand, computation leakage models usually place more restrictions on the allowed leakage, such as, for example, assuming that different components of a computation that are separated in space or in time leak independently (i.e., the adversary can obtain separate leakage functions of some intermediate values, but not a joint function of them all), or that some memory does not leak at all. This is in contrast to memory leakage models, which usually allow the leakage to be an arbitrary (bounded) function of the entire secret.

Ishai, Sahai, and Wagner [ISW03] built on the work of Chari et al. [CJRR99] to model leakage from wires of a circuit. In the model of [ISW03], the computation is performed by a clocked circuit with a secret state (for example, a circuit implementing a block cipher with a secret key). The circuit is run repeatedly on various inputs, producing outputs and possibly also updating the state. The adversary is able to provide inputs and observe outputs as well as the exact values of some internal wires during the computation. This model and its variants resulted in a long line of work that we survey in Section 4.3.

Micali and Reyzin [MR04] gave a more general model of leakage during computation. They modeled computation as proceeding in steps, and allowed the adversary to obtain different side-channel information at each step. Specifically, they described their model in terms of Random-Access Machines (RAMs, which are Turing Machines augmented with addressable memory) rather than circuits, although circuit variants of their model were considered later. In this model, an adversary is able to specify a leakage function (from a class of available functions) at each step of the computation. The function is applied to the current state of the computing machine and the output is given to the adversary, who uses this information to specify the function for the next step. In order to enable security against such general attacks, Micali and Reyzin assumed the existence of secure storage that is not given to the leakage function. That is, values can leak when being computed on and being read from or written to memory; but once they are in memory, the leakage

function has no access to them. This assumption became known as "Only Computation Leaks Information," commonly abbreviated as OCL. This assumption was generalized in later work, as discussed in Sections 3 and 4 (see, in particular, Section 4.1). The power of this assumption comes from enabling constructions that separate computation into two or more components that leak independently, as shown in [DP08] (see Section 4.2.2).

We elaborate on leakage from computation in Section 4.

1.3 Roadmap

In this survey, we address the two strands of works on leakage-resilient cryptography: "leakage from memory" (Section 2) and "leakage from computation" (Section 4).

We emphasize that this division is not perfect. Some papers consider both memory and computational leakage. In addition, some papers on memory leakage use results on computational leakage, and vice versa. Nevertheless, we feel this division is helpful for systematizing knowledge in this area.

There is yet another category of papers on "leakage-resilient storage". This category lies in between the two categories described above. It considers the problem of storage, rather than computation, and thus considers leakage from memory. However, papers in this category typically restrict the leakage function in the same way as works in the "computational leakage" category do: the stored secret is separated into components, and leakage functions are applied separately to each component, but never jointly to all of them. The works in this category are described in Section 3.

We assume that the readers possesses a solid background in cryptography and is familiar with such concepts as CPA-secure encryption, zero-knowledge proofs, and secure multi-party computation. We assume the reader is reasonably comfortable with commonly used tools, such as randomness extractors[1] and pseudorandom generators[2].

2 Memory Leakage

The main goal of works discussed in this section is to build cryptographic schemes that can remain secure even if some partial information about the secret key is available to the adversary. It is important to recall the basic fact that the adversarial inability to recover the full secret key is a necessary, but not a sufficient, condition for the security of a cryptographic construction.

2.1 The Models for Memory Leakage

As already mentioned in Section 1.2, Dziembowski [Dzi06] and Di Crescenzo, Lipton, and Walfish [DLW06] considered arbitrary leakage from memory, proposing the *bounded retrieval model*. In this model, the adversary can obtain an arbitrary polynomial-time computable leakage function of the secret key, but the output size of this leakage function is bounded. Security is achieved by

[1] The notion of a seeded randomness extractor, introduced by Nisan and Zuckerman [NZ96], is defined as follows: A function $\mathsf{Ext} : \{0,1\}^n \times \{0,1\}^d \to \{0,1\}^\ell$ is said to be a (k, ε) extractor if for any random variable X over $\{0,1\}^n$ with min-entropy k, and for a uniformly chosen $r \leftarrow \{0,1\}^d$, it holds that $(r, \mathsf{Ext}(x, r))$ is ε-statistically close to a uniform string over $\{0,1\}^{d+\ell}$.

[2] The notion of a cryptographic pseudorandom generator (PRG), introduced in [BM82, Yao82, BM84], is defined as follows: A function $G : \{0,1\}^k \to \{0,1\}^\ell$ is a PRG if, for a uniform secret s, the output $G(s)$ is computationally indistinguishable from a uniform string over $\{0,1\}^\ell$.

making the secret key longer than this leakage length bound. While in most cryptographic schemes long secret keys would translate into long running times, this model requires that essentially the only price for increased leakage should be increased secret storage: the running time of the parties should grow only logarithmically with the leakage length bound. In particular, the parties do not need to access the entire long secret key for each operation. We discuss this model and relevant constructions in Section 2.4. Initially, works in the bounded retrieval model achieved only symmetric-key cryptographic constructions, because growing the secret key size while maintaining the public key the same presents a challenge.

In the public key setting, Akavia, Goldwasser, and Vaikuntanathan [AGV09] considered arbitrary leakage from the secret key, defining the term *bounded memory leakage*, also known as *relative memory leakage*. In this model, similarly to the bounded retrieval model, the leakage function is an arbitrary bounded-output-length polynomial-time computable function; but the output length of this function is expressed as function of the key length (or, more generally, of the min-entropy of the key). Typically, the goal is to obtain security even if a large fraction of the secret key (or its min-entropy) is leaked. Unlike the bounded retrieval model, this model does not place any restrictions on running times, and thus increasing key size in order to allow more leakage (in absolute terms) will negatively affect the performance of most constructions. We elaborate on this model in Section 2.2.

Shortly after, Dodis, Kalai, and Lovett [DKL09] generalized the notion of bounded leakage to so-called *auxiliary input leakage*. In this model, the leakage function can have *unbounded* output length, and the only restriction is that given the leakage (and the public interface) it is (computationally) hard to find the secret key. This restriction seems to be the minimal necessary to achieve meaningful security, because no security remains if the secret key can be computed from the leakage. We elaborate on this model in Section 2.3.

Even though the auxiliary input leakage model seems the strongest possible for one-time leakage, it cannot protect against continual leakage, where the secret key is leaked continually few bits at a time, since in this case the secret key can eventually leak entirely. To handle leakage over the long term, the continual memory leakage model, defined by [BKKV10, DHLW10a], considers the setting in which the secret key is periodically updated, *without updating the public key*, and assumes that there is bounded memory leakage (in the sense of [AGV09]) within each time period, but there is no bound on the overall leakage. We elaborate on this line of work in Section 2.5.

We emphasize that in all four models mentioned above, each bit of leakage can be an *arbitrary* efficiently computable function of the secret key (with the minimal necessary restriction in the auxiliary input case). This is in contrast to the leakage models that are considered in Sections 3 and 4, where the leakage functions are restricted in some way (such as OCL, noisy, or low-complexity leakage).

In Sections 2.2-2.5, we define the foregoing leakage models and show constructions of specific leakage-resilient cryptographic systems. We emphasize that, in most cases, the leakage function is applied only to the *secret key* (and publicly available information, such as the public key), and no leakage occurs during computation. For example, leakage cannot depend on the secret randomness used during a computation. There are a few exceptions, starting from the work of Boyle, Segev, and Wichs [BSW11] (mentioned in Section 2.2 below), which constructs a signature scheme in the bounded memory leakage that is secure even if the leakage is applied to the secret key *and* the randomness used to generate a signature.

In Sections 2.2-2.5, we focus on constructing non-interactive cryptographic primitives, such as

leakage-resilient encryption schemes and signature schemes. In Section 2.6 we consider leakage-resilient interactive protocols, which are different from cryptographic schemes discussed in Sections 2.2-2.5, in that the leakage does not necessarily come from the secret key. Thus, in the setting of interactive protocols, it is more difficult to define security in the presence of leakage, since we have to account for leakage coming not from secret keys, which are meaningless on their own, but from protocol inputs (for example, witnesses to ZK statements), which carry meaningful private information.

2.2 Bounded Memory Leakage

As mentioned above, Akavia, Goldwasswer, and Vaikuntanathan [AGV09] introduced the notion of *bounded memory leakage*. They considered an adversarial model in which the adversary can request a bounded amount of leakage on the secret key, adaptively one bit at a time. Let κ be the length of the secret key sk and let $\alpha \in (0,1)$ be the allowed leakage fraction. In this model the adversary can make $\alpha\kappa$ oracle queries, where each query consists of a Boolean circuit $C : \{0,1\}^\kappa \to \{0,1\}$ and is answered by $C(\mathsf{sk})$. Each circuit can be chosen based on previous leakage information and other information known to the adversary from the public interface (such as the public key, known signatures, etc.). We note that the size of each circuit is obviously bounded by the running time of the adversary, and hence leakage functions have bounded complexity. If the adversary cannot break the scheme after at most $\alpha\kappa$ such leakage queries, then the scheme is said to be α-leakage-resilient.

As observed by [AGV09], any public key encryption scheme that is secure against adversaries running in time $2^{\alpha\kappa}$, is also α-leakage-resilient. Intuitively, this follows from the fact that if one can break the scheme with $L = L(\kappa)$ bits of leakage in time $T = T(\kappa)$, then one can break the scheme without any leakage in time $2^L \cdot T$. This observation was made in the context of Regev's public key encryption scheme [Reg05], but easily extends to any exponentially secure encryption scheme.

Naor and Segev [NS09] constructed a public key encryption scheme that is secure against bounded memory leakage under standard *polynomial-time* assumptions. They started with the observation that the circular secure scheme of Boneh et al. [BHHO08] is already leakage-resilient under the DDH assumption. More generally, they showed how to construct a leakage-resilient public key semantically secure encryption from any hash proof system [CS02], thus showing how build leakage-resilient encryption schemes on a variety of assumptions, such as the Quadratic Residuosity Assumption, DDH, and Nth Residuosity Assumption. Moreover, they prove that the Naor-Yung paradigm [NY90] is applicable in this setting, and thus obtain leakage-resilient encryption schemes that are CCA2-secure. These schemes are reslient to $1 - o(1)$ leakage rate.

These schemes (as well as schemes in followup work) have the following blueprint: The public key has exponentially many valid secret keys, so that even given the leakage (and the public key), the secret key still has high min-entropy. For example, in the encryption scheme of [BHHO08], the secret key is $(g_1, g_2, \ldots, g_\ell, s_1, s_2, \ldots, s_\ell)$, where $g_1, g_2, \ldots, g_\ell$ are random generators in a group G of prime order p, and $s_1, s_2, \ldots, s_\ell$ are all randomly chosen in $\mathbb{Z}_p$; the public key is $(g_1, g_2, \ldots, g_\ell, h)$ where $h = g_1^{s_1} \cdot g_2^{s_2} \cdot \cdots \cdot g_\ell^{s_\ell}$. In addition, there is an alternative mode for generating ciphertexts (used only in the proof of security), such that even given the entire secret key one cannot distinguish between an honestly generated ciphertext and one that is generated via the alternative mode. Importantly, if the secret key has sufficient min-entropy then a ciphertext generated via the alternative mode *information theoretically* hides the message.

For example, in the encryption scheme of [BHHO08], the correct ciphertext corresponding to a message m is of the form $(g_1^r, g_2^r, \ldots, g_\ell^r, (g_1^{s_1} \cdot g_2^{s_2} \cdot \cdots \cdot g_\ell^{s_\ell})^r \cdot m)$ for randomly chosen r

in $\mathbb{Z}_p$. In the alternative mode, the ciphertext is generated by $(g_1^{r_1}, g_2^{r_2}, \ldots, g_\ell^{r_\ell}, g_1^{s_1 \cdot r_1} \cdot g_2^{s_2 \cdot r_2} \cdot \cdots \cdot g_\ell^{s_\ell \cdot r_\ell} \cdot m)$, for randomly chosen $r_1, r_2, \ldots, r_\ell$ in $\mathbb{Z}_p$. By DDH, even given the secret key $(g_1, g_2, \ldots, g_\ell, s_1, s_2, \ldots, s_\ell)$, the correct and alternative ciphertexts are indistinguishable. The alternative ciphertext information-theoretically hides the message m, as long as sufficient min-entropy remains in the secret key after leakage, because for fixed $(g_1, g_2, \ldots, g_\ell)$, the mapping from $(s_1, s_2, \ldots, s_\ell, r_1, r_2, \ldots, r_\ell)$ to $g_1^{s_1 \cdot r_1} \cdot g_2^{s_2 \cdot r_2} \cdot \cdots \cdot g_\ell^{s_\ell \cdot r_\ell}$ is a strong randomness extractor when $(r_1, r, \ldots, r_\ell)$ is viewed as the seed and $(s_1, s_2, \ldots, s_\ell)$ is viewed as the source. Indeed, it was proven in [NS09] that this scheme is resilient to $1 - o(1)$ leakage rate, i.e., security holds even if all but $o(1)$-fraction of the secret key is leaked.

This blueprint (of analyzing security by showing indistinguishability to a setting where security holds information-theoretically) is used in many followup works, including constructions of leakage-resilient CCA secure encryption schemes, identity based encryption scheme, pseudo-random functions, and more. See, for example, [FKPR10, DHLW10b, BHK11, GV13a, FNV15].

We emphasize that typically, leakage-resilient encryption schemes assume that the leakage happens *before* the ciphertext is generated, and security is guaranteed only for future ciphertexts. Halevi and Lin [HL11] considered the model of *after-the-fact leakage*. They formulated the notion of *entropic* leakage-resilient public key encryption, which captures the intuition that as long as the entropy of the encrypted message is higher than the amount of leakage, the message still has some (pseudo) entropy left. They show that this notion is realized by the Naor-Segev constructions mentioned above. In order to achieve more traditional CPA security against after-the-fact leakage, they move to a weaker leakage model (so-called OCL model); we discuss this result and some follow-up work in Section 4.2.6, after the OCL model is introduced in Section 4.1.

Katz and Vaikuntanathan [KV09] showed how to construct a leakage-resilient signature scheme in the bounded memory leakage model. Loosely speaking, their blueprint is somewhat similar to the above: Start with a public verification key pk that has exponentially many secret keys associated with it. In particular, the public verification key contains a hash value $y = h(x)$ and the secret key contains the pre-image x.

Their first observation is that any target-collision-resistant hash function[3] h is leakage-resilient. Namely, given $y = h(x)$ and bounded (efficiently computable) leakage $L(x)$ on x, it is hard to invert h on y. The reason is that even given y and $L(x)$, x still has sufficient min-entropy, and thus if an adversary can invert y (given $L(x)$) then with high probability it will output $x' \neq x$ such that $h(x') = h(x)$ and $L(x') = L(x)$. Thus, this adversary can be used to break the target collision resistant property, which gives the adversary even more information (namely, all of x).

Their signature scheme has the property that an adversary that forges a signature must "know" a secret key corresponding to y (which is part of the public key). This is achieved by having the signature contain an encryption of x, along with a non-interactive zero-knowledge (NIZK) proof that indeed the ciphertext decrypts to a pre-image of y. We note that in order to make the proof go through, one needs to use what is known as a "simulation sound" NIZK [BFM88, Sah99]: When using the adversary to break the target collision resistance property, we need to provide this adversary with signatures to messages of its choice, and to ensure that the secret key still has high min-entropy; these signatures will contain a ciphertext that decrypts to 0 (rather than a valid secret key), along with a simulated NIZK. The simulation soundness guarantees that the adversary must still generate a ciphertext that decrypts to a secret key.

[3] A function h is target-collision-resistant (also known as universal one-way hash function) if given a random element x in the domain it is hard to find $x' \neq x$ such that $h(x) = h(x')$.

All the works mentioned above constructed leakage-resilient schemes based on specific number-theoretic assumptions. Hazay et al. [HLWW13, HLWW16] construct a leakage-resilient CPA-secure encryption scheme from *any* (not leakage-resilient) CPA-secure encryption scheme. Loosely speaking, Hazay et al. extend the work of Naor and Segev [NS09], and construct a leakage-resilient encryption scheme from any *weak hash-proof system*. In addition, they show how to build such weak hash-proof system from any CPA-secure encryption scheme. However, the leakage rate α in their resulting scheme is quite low. They also construct a leakage-resilient symmetric encryption scheme, weak PRF, and message authentication code from any one-way function. In addition, they extend their results to the after-the-fact leakage model of [HL11] mentioned above and to the bounded retrieval model (see Section 2.4).

We emphasize that in all the schemes mentioned above, the leakage is only a function of the secret key (and publicly available information, such as the corresponding public key). Boyle et al. [BSW11] (and followup works) constructed a signature scheme where the leakage can also depend on the randomness used to generate the signatures. This leakage model is somewhat reminiscent to the leakage models considered in Section 4, where the leakage occurs during computation. In particular, such leakage-resilient signature scheme must have the property that signatures hide the secret key, even given bounded leakage on the entire state of this computation.

2.3 Auxiliary Input Memory Leakage

Shortly after the formalization of bounded memory leakage, Dodis, Kalai, and Lovett [DKL09] formulated the notion of *auxiliary input* memory leakage. The motivation for this model is that in reality side-channel attacks can leak *many* bits about the secret key, more than the length of the secret key. Of course, if the secret key is fully computable from the leakage, all hope is lost. On the other hand, even if many bits are leaked, as long as the secret key is not computable from them, it may still be possible to build a secure cryptographic scheme.

Formally, the *auxiliary input* model considers any (efficiently computable) leakage function f applied to the secret key sk, even one with *long output*, as long as given $f(\mathsf{sk})$, together with other public information, it is computationally (sufficiently) hard to find a valid secret key. Namely, in this model, the adversary can choose an arbitrary leakage function $f : \{0,1\}^\kappa \to \{0,1\}^*$ (modelled as a Boolean circuit) to be applied to the entire secret key sk, so long as f is (sufficiently) hard to invert, given all the information known to the adversary, such as the public key. As above, security is required to hold even against adversaries that are given $f(\mathsf{sk})$. This function f can be adaptively chosen based on all the information known to the adversary.

Because this model requires only that the secret key should have computational secrecy given the leakage, it is more general than the bounded memory leakage model of Section 2.2, which requires that the secret key should have some information-theoretic uncertainty given the leakage. The auxiliary input leakage model attempts to consider the most general possible leakage that does not trivially break security. This model is inspired by the work of Canetti [Can97], which studies cryptography with auxiliary inputs in the context of perfect one-way functions.[4]

In their work, Dodis, Kalai, and Lovett [DKL09] constructed a *symmetric encryption scheme* secure against auxiliary input leakage, as long as the leakage function satisfies the condition that every polynomial size algorithm can invert it with probability at most $2^{-\epsilon n}$ for some constant

[4]We note that Goldwasser and Kalai [GK05] considered the auxiliary input model in the context of obfuscation. However, they obtained mainly negative results, demonstrating the impossibility of obfuscation with auxiliary input.

$\epsilon > 0$, where n is the length of the secret key. In what follows we outline the ideas behind their scheme. The first observation is that constructing a symmetric encryption scheme that is resilient to leakage seems to be much easier than constructing a public key one, since intuitively, one can apply a seeded extractor $\mathsf{Ext} : \{0,1\}^n \times \{0,1\}^d \to \{0,1\}^\ell$ to the (partially leaked) secret key, and use $\mathsf{Ext}(x,r)$ as the secret key, where r is a random seed that is appended to the ciphertext, so that the party decrypting this message could reconstruct the effective secret key $\mathsf{Ext}(x,r)$. We note that this general approach gives only one-time (or bounded-time) security; i.e., security holds only if the adversary is allowed to see only bounded number of ciphertexts. Indeed, if the adversary is given many pairs $(r_i, \mathsf{Ext}(\mathsf{sk}, r_i))$ then he may be able to efficiently reconstruct the secret key sk. However, we can obtain many-time security by adding some "noise," as we explain next.

Specifically, consider the inner product seeded extractor $\mathsf{Ext} : \{0,1\}^n \times \{0,1\}^n \to \{0,1\}$, defined by $\mathsf{Ext}(x,r) = \langle x, r\rangle$. When using this extractor in the approach above, with additional noise, we obtain the following symmetric encryption scheme: To encrypt a message $b \in \{0,1\}$ using a (partially leaked) secret key sk, choose a random $r \in \{0,1\}^n$ and let the ciphertext be $(r, \langle \mathsf{sk}, r\rangle \oplus e \oplus b)$, where e is 1 with small probability ϵ and is 0 otherwise. Note that this ciphertext has a decryption error of ϵ. This decryption error is overcome via repetition: Namely, an encryption of $b \in \{0,1\}$ will consist of many pairs $(r_i, \langle \mathsf{sk}, r_i\rangle \oplus e_i \oplus b)$, where each e_i is sampled independently and is 1 with small probability ϵ and is 0 otherwise. This is indeed a symmetric encryption, and its (many-time) security follows from the assumption that learning parity with noise (LPN) is hard. More importantly, one can argue that even if the secret key is partially leaked (and only has sufficiently high min-entropy), then this encryption remains secure. Intuitively, this follows from the fact that the inner product is an extractor.

Recall, however, that our goal is to prove that security holds given $f(\mathsf{sk})$, for any polynomial-time computable function f that is sufficiently hard-to-invert.[5] This follows from the hard-core predicate theorem of Goldreich and Levin [GL89], which asserts that for every one-way function $f : \{0,1\}^n \to \{0,1\}^*$, the pair $(r, \langle \mathsf{sk}, r\rangle)$ is computationally indistinguishable from uniform *even given* $f(\mathsf{sk})$.

The foregoing idea was carried over to the public key setting by Dodis et al. [DGK$^+$10], who constructed a *public-key* encryption scheme and proved that it is CPA secure against auxiliary inputs under the learning with errors (LWE) assumption. They proved leakage resilience against any sub-exponential hard-to-invert leakage function (i.e., any leakage function such that poly-size circuits can invert it with probability at most 2^{-n^ϵ} for some constant $\epsilon > 0$, where n is the size of the secret key).

They also showed that the BHHO encryption scheme [BHHO08], which was proven to be resilient to bounded memory leakage, is in fact CPA secure against such sub-expontentially hard-to-invert auxiliary inputs under the DDH assumption. Recall that the in the BHHO encryption scheme, the secret key is of the form $(g_1, g_2, \ldots, g_\ell, s_1, s_2, \ldots, s_\ell)$, where each g_i is randomly chosen from a group G of prime order p, and each s_i is randomly chosen from $\mathbb{Z}_p$, and the public key is $(g_1, g_2, \ldots, g_\ell, h)$ where $h = g_1^{s_1} \cdot g_2^{s_2} \cdot \cdots \cdot g_\ell^{s_\ell}$. The encryption of a message m is of the form $(g_1^r, g_2^r, \ldots, g_\ell^r, h^r \cdot m)$. As mentioned in Section 2.2, even given the secret key, this cipertext is indistinguishable from an alternative ciphertext of the form $(g_1^{r_1}, g_2^{r_2}, \ldots, g_\ell^{r_\ell}, \prod g_i^{r_i s_i} \cdot m)$, where $r_1, r_2 \ldots, r_\ell$ are all chosen randomly and independently in $\mathbb{Z}_p$. Denoting each $g_i = g^{\alpha_i}$, where g is an (arbitrary) generator of the group G, we note that the (alternative) ciphertext masks m with $g^{\langle r,s\rangle}$, where $r = (r_1, r_2, \ldots, r_\ell)$ and $s = (s_1, s_2, \ldots, s_\ell)$. Thus, the result of [DGK$^+$10] is obtained

[5] In particular, sk may have no min-entroypy conditioned on $f(\mathsf{sk})$.

by extending the Goldreich-Levin theorem to provide a hard-core value over large fields.

More generally, [DGK+10] proved that these schemes are secure against a richer class of leakage functions, for example, leakage functions that are polynomially hard-to-invert with probability $2^{-\text{polylog}(n)}$ (however, then the corresponding assumptions are the sub-exponential security of LWE/DDH). Following this work, Goldwasser et al. [GKPV10] used a similar approach to argue that the LWE assumption itself is robust to auxiliary inputs.

Brakerski and Goldwasser [BG10] showed how to construct a public-key encryption scheme secure against sub-exponentially hard-to-invert leakage, based on the Quadratic Residuosity (QR) and Decisional Composite Residuosity (DCR) hardness assumptions. Brakerski and Segev [BS11] considered the problem of deterministic public-key encryption in the presence of auxiliary leakage, and proposed several constructions based on the DDH assumption and subgroup indistinguishability assumptions.

Summary of the leakage models discussed so far. In Section 2.2 we defined bounded memory leakage, where the length of the leakage is bounded *relative* to the length of the secret key, which in turn depends on the security parameter. In Section 2.3 we defined the auxiliary input model, where the length of the leakage is arbitrary, but it is required that given this leakage (and other public information), finding the secret key should be hard. Unfortunately, the theoretical restrictions on the leakage function are unsupported by the bitter reality that the key may eventually leak completely over time. While at first glance it may seem impossible to do anything about this problem, as the auxiliary input leakage seems to impose the minimal necessary requirement on the leakage function, two approaches have been proposed to address it. The first is the *bounded retrieval* discussed in Section 2.4, and the second is the *continual memory leakage* model discussed in Section 2.5.

2.4 Bounded Retrieval Model

The bounded retrieval model (BRM), defined by Di Crescenzo, Lipton, and Walfish [DLW06] and Dziembowski [Dzi06], assumes that there is a bound B on the overall leakage. However, as opposed to the bounded memory leakage of Section 2.2, this bound is thought of as being extremely large, and in particular, can be significantly larger than the security parameter, and longer than the number of steps it takes to decrypt or sign. For security, the minimum requirement is that the secret key must be longer than B (else it could leak entirely); the goal of constructions in this model is to make sure that the efficiency of the system does not degrade with this bound B. That is, the goal of BRM is to protect against large amounts of leakage by making the secret key even larger, while ensuring that this necessary inefficiency in storage is essentially the only inefficiency of the system. This means that for every operation, honest users should have to read only a small portion of the secret (this property is called locality), and their computation and communication should not be much larger than in conventional cryptosystems. To put it differently, the bounded retrieval model studies the same problem as the bounded memory leakage model, but allows the users to increase their secret key size flexibly, so as to protect against large amounts of leakage, *without degrading other efficiency parameters*. This model is motivated by various malware attacks, in which a persistent virus may transmit a large amount of private data to a remote attacker.

As mentioned above, this model preceded the bounded leakage model, and the original work that introduced this model [DLW06, Dzi06] constructed leakage-resilient password and authentication

protocols. The work of Alwen, Dodis, and Wichs [ADW09] constructed leakage-resilient identification schemes, signature schemes, and authenticated key agreement protocols in this model, and shortly after, Alwen et al. [ADN+10] constructed a leakage-resilient public key encryption scheme in this model.

Loosely speaking, these schemes are constructed via a generic *leakage-resilience amplification* process. Namely, start with a leakage-resilient primitive in the bounded memory leakage model of Section 2.2 (also known as the relative leakage model and use it to construct a B-leakage-resilient primitive in the bounded retrieval model (for an arbitrary value of B).

The naive approach is to artificially inflate the security parameter to be larger than the bound B. This approach clearly does not satisfy the desired efficiency requirements. A better approach is to use *parallel repetition*. For the sake of concreteness, suppose we start with a public key encryption scheme that is secure in the relative leakage model (described in Section 2.2). As a first attempt at converting this scheme to the bounded retrieval model, store many secret keys $\mathsf{sk}_1, \ldots, \mathsf{sk}_N$, together with the corresponding public keys $\mathsf{pk}_1, \ldots, \mathsf{pk}_N$. To ensure that the ciphertext remains succinct, to encrypt a message m, choose a few random indices $i_1, \ldots, i_\kappa \in [N]$, secret share the message via a κ-out-of-κ secret sharing scheme (e.g., by choosing κ random messages $m_1, \ldots, m_\kappa$ such that $m = m_1 \oplus \cdots \oplus m_\kappa$), and output $(\mathsf{Enc}_{\mathsf{pk}_{i_1}}(m_1), \ldots, \mathsf{Enc}_{\mathsf{pk}_{i_\kappa}}(m_\kappa))$. Intuitively, even if ϵN bits are leaked, since the adversary does not know ahead of time which indices $i_1, \ldots, i_\kappa$ will be chosen during the ciphertext generation, at least one of the secret keys $\{\mathsf{sk}_{i_j}\}_{j \in [\kappa]}$ is likely to "still have sufficient min-entropy conditioned on the leakage", which in turn seems to imply that security holds. Unfortunately, formalizing this intuition is currently beyond reach, because the leakage can be a complex function of all keys $\mathsf{sk}_1, \ldots, \mathsf{sk}_N$.

Note that the ciphertext is small, independent of the absolute leakage bound B. However, the length of the public key $(\mathsf{pk}_1, \ldots, \mathsf{pk}_N)$ is large (and grows with B). This shortcoming is overcome by using an *identity based encryption* (IBE) scheme, as opposed to a standard encryption scheme. The public key of the parallel repetition scheme is simply the master public key of the IBE scheme. The secret key is the secret keys corresponding to N fixed IDs $\mathsf{ID}_1, \ldots, \mathsf{ID}_N$.

This scheme satisfies the required efficiency guarantees: the ciphertexts and the public key are succinct (do not grow with B), encryption is efficient, and decryption is efficient given random access to the secret key.

Security. Despite the intuition above, it turns out that this scheme is not necessarily secure. In particular, [ADN+10] construct an artificial IBE scheme for which this blueprint results in an insecure scheme. Loosely speaking, this IBE scheme has the property that given secret keys of many identities, one can compress these keys to a short "digest" (of size independent of B) such that from this digest one can reconstruct all the compressed secret keys. To get around this problem, [ADN+10] construct an IBE scheme with an additional special structure, which they call "identity-based hash proof system", and prove the security of the above blueprint if the IBE scheme used is an identity-based hash proof system. They construct such an identity-based hash proof system based on several standard assumptions (such as Quadratic Residuosity, Learning with Errors, and Bilinear Diffie-Hellman).

We refer the reader to Alwen, Dodis, and Wichs [ADW10] for a fantastic survey on the bounded retrieval model.

2.5 Continual Memory Leakage

The continual leakage model considers the setting in which the total leakage is *unbounded* and yet all the parameters of the scheme (including the length of the secret key) are bounded (and depend only on the security parameter). In particular, the leakage can eventually reveal as many bits as there are in the secret key, and we still want to argue security in this case. This seemingly impossible task is achieved by periodically *updating* the secret key, *without changing the public key*. Namely, as is often the case in leakage-resilient schemes, in this setting a public key pk has (exponentially) many secret keys associated with it. The initial secret key is sk_1; it is updated every time period, to $\mathsf{sk}_2, \mathsf{sk}_3$, etc., so that all the secret keys $\mathsf{sk}_1, \mathsf{sk}_2, \mathsf{sk}_3 \ldots$ correspond to the same public key pk. The security guarantee is that even if the adversary obtains *bounded* leakage on each sk_i (but unbounded leakage overall), the scheme remains secure.

Specifically, in the continual leakage model security holds even given $L_1(\mathsf{sk}_1), \ldots, L_N(\mathsf{sk}_N)$, where N is adversarially chosen, and $L_1, \ldots, L_N$ are adversarially chosen functions (represented as circuits) of bounded output length. Of course, for any security to hold, the output length of each L_i must be smaller than $|\mathsf{sk}_i|$.

The model was first considered by Brakerski et al. [BKKV10] and Dodis et al. [DHLW10a], who constructed public-key encryption and signature schemes that are secure even when the leakage length in each time period is a constant fraction $|\mathsf{sk}_i|$, under the decisional linear assumption in bilinear groups. These works allow no leakage during the key updates.[6]

The encryption scheme (constructed in [BKKV10]) is a variant of the BHHO encryption scheme, discussed above. Let the secret key be a random vector $s = (s_1, \ldots, s_\ell) \in \mathbb{Z}_p^\ell$. Let g be a generator of a group G of prime order p. Let $a = (a_1, \ldots, a_\ell)$ be a random element in $\mathbb{Z}_p^\ell$ such that the inner product $\langle a, s \rangle = 0$ modulo p, and the public key be $(g^{a_1}, \ldots, g^{a_\ell})$. To encrypt a bit 0, choose a random $r \in \mathbb{Z}_p$ and output $(g^{a_1 r}, \ldots, g^{a_\ell r})$, and to encrypt the bit 1 output a random element in G^ℓ. Decryption is done by raising the ciphertext to the power of $s = (s_1, \ldots, s_\ell)$ coordinate-wise, multiplying all the coordinates together, and outputting 0 if the resulting product is the identity element of G, and 1 otherwise.

This scheme is resilient to bounded memory leakage, and even to auxiliary input memory leakage, via a similar analysis to the ones outlined in Sections 2.2 and 2.3, respectively. However, it is not clear how to (efficiently) update the secret key, in order to make this scheme secure against continual memory leakage.

Given a secret key $s = (s_1, \ldots, s_\ell)$ and a public key $(g^{a_1}, \ldots, g^{a_\ell})$, we can efficiently update the secret key by choosing a random $\alpha \in \mathbb{Z}_p$ and setting the updated secret key to be $\alpha s = (\alpha s_1, \ldots, \alpha s_\ell)$. However, this scheme is not secure against continual memory leakage, since an adversary can, for example, normalize the secret key by dividing all the coordinates by the first coordinate, and leak on this normalized key, which remains unchanged.

To get around this attack, rather than setting the secret key to be $s = (s_1, \ldots, s_\ell)$, set it to be $g^s = (g^{s_1}, \ldots, g^{s_\ell})$. In order to maintain the ability to decrypt we need to rely on a group G with a bilinear map $e : G \times G \to G_T$. To decrypt, pair the ciphertext $(g^{y_1}, \ldots, g^{y_\ell})$ with the secret key$(g^{s_1}, \ldots, g^{s_\ell})$, to obtain $\prod_{i=1}^{\ell} e(g^{y_i}, g^{s_i})$, and output 0 if the value obtained is the identity element of G_T; otherwise output 1. To update the secret key, simply raise the secret key to the power of a random $\alpha \in \mathbb{Z}_p$ (coordinate by coordinate).

[6] More generally, these works are resilient to logarithmic amount of leakage during key updates. Very loosely speaking, this follows from the fact that such small quantity of leakage can be guessed with non-negligible probability and thus cannot be of much help to the adversary.

One can prove that this scheme is secure against continual leakage under the DDH assumption; however, this assumption is known to be false in groups with bilinear maps. This obstacle is bypassed by either considering an asymmetric map, and relying on the SXDH assumption, or setting the secret key to be a matrix with two rows, and relying on the decisional linear assumption.

To prove security, we rely on the fact that under the SXDH assumption (or the decisional linear assumption), an adversary cannot distinguish between the case that the updates are done as prescribed, and the case that they are done by choosing a fresh random secret s in the kernel of a, and raising it to the power of g; and this indistinguishability holds even given the secret key. Moreover, one can prove that if the key is updated in the alternative way described above, then security holds in the continual memory leakage model.

Leakage during updates. Lewko, Lewko, and Waters [LLW11] showed how to achieve constant leakage rate during key updates; the security of their scheme is under the subgroup decision assumption in composite order bilinear groups. This work was improved by Dodis et al. [DLWW11] and modified to achieve leakage-resilient storage (see Section 3).

Dachman-Soled et al. [DGL+16] showed a generic way to tolerate leakage during key updates. Specifically, they showed how to use obfuscation to compile any public-key encryption or signature scheme that satisfies a slight strengthening of continual memory leakage (which they refer to as "consecutive" memory leakage) but does not tolerate leakage on key updates, to one that is resilient to continual memory leakage *with leakage on key updates*.

Further strengthening the model. The continual leakage model was further strengthened in different ways. Yuen et al. [YCZY12] considered the *continual auxiliary input leakage* model, in which the leakage per time period is not required to be bounded in length, but rather can be an arbitrary *hard-to-invert* function of the secret key, like the leakage in Section 2.3. They construct identity-based encryption which is secure in this model, by applying a modified version of the Goldreich-Levin theorem, together with the ideas from [LLW11], of using dual system encryption systems for leakage-resilience.

Malkin et al. [MTVY11] consider continual memory leakage, where leakage can occur also during computations. They present a signature scheme that is resilient to continual leakage, where leakage can occur during the signing process, and thus the leakage is a function of both the secret key and the randomness used to sign a message. We discuss other signature schemes that can handle leakage during the signing process in Section 4.2.6.

Dziembowski, Kazana, and Wichs [DKW11] consider a combination of continual memory leakage with the bounded retrieval model described in Section 2.4, and construct schemes that are resilient against such leakage if the leakage function itself has limited space for its computation (see also Section 4.2.4 for more on their model).

2.6 Interactive Protocols

So far, we mainly focused on leakage-resilient cryptographic primitives, such as encryption schemes and signature schemes, with the goal of preserving the original security guarantees in the presence of leakage.

In this section, we extend the notion of leakage resilience to the context of *interactive protocols*. The initial works that construct leakage-resilient interactive protocols focused on specific tasks, such

as coin tossing [BGK11], zero-knowledge [GJS11, BCH12], secure message transmission, message authentication, commitment, and oblivious transfer [BCH12]. These works, as well as followup works, consider the setting where an adversary can obtain arbitrary (bounded) leakage on the entire state of *each* (honest) party during the entire protocol execution.

Boyle, Goldwasser, and Kalai [BGK11] constructed a coin tossing protocol with the standard security guarantee upgraded for leakage resilience: namely, even if the adversary leaks a constant fraction of the state of each (honest) party, she cannot distinguish the output from a random coin toss. In the context of zero-knowledge, it is easy to see that achieving similar leakage resilience under the standard zero-knowledge definition is simply impossible. For example, consider an adversary that leaks ℓ bits of information from the state of the prover, by leaking the first ℓ bits of the witness. Clearly, this adversary's view cannot be efficiently simulated (assuming these bits of the witness are hard to compute). Instead, the (concurrent) works of Garg, Jain, and Sahai [GJS11] and Bitansky, Canetti, and Halevi [BCH12] weaken the zero-knowledge condition in the leaky setting, to require that the protocol does not reveal any information beyond the validity of the statement *and the leakage obtained by the adversary*. Defining this formally is non-trivial, as we explain below.

Bitansky, Canetti, and Halevi [BCH12] presented a *general framework* for expressing security requirements of interactive protocols in the presence of arbitrary (poly-time) leakage. Noting that standard "ideal world" security, where the side-channel adversary does not learn more than the inputs and outputs of the malicious parties, is in general impossible, they defined the notion of *leakage tolerance*, as follows. Consider an adversary who leaks a total of ℓ bits of information from all the (honest) parties. A leakage-tolerant protocol ensures that such an adversary learns at most what can be learned in the *leaky ideal world*, in which the ideal-world adversary also gets ℓ bits of leakage.[7] Thus, a leakage tolerant protocol is one where the level of security gracefully degrades with the amount of leakage (which may develop over time).

In more detail, they consider a "real world" in which the adversary can get leakage on the entire state of any one party at any time (but cannot get joint leakage on the states of many parties). To account for the security degradation this leakage necessarily causes, they also allow the same amount of leakage in the "ideal world." More specifically, the leaky ideal model they consider is the so-called *individual leakage model*, which allows the ideal world adversary to obtain leakage on the input of each party *separately*, as long as the total number of bits leaked is at most ℓ.

Constructing leakage tolerant protocols is highly non-trivial. Intuitively, the initial difficulty is that we need to simulate the protocol *without knowing* the inputs of the honest parties and then later "explain" the leaked information. As observed in [GJS11, BCH12], this is reminiscent to the difficulty in constructing *adaptively secure* protocols. This connection was formalized in [NVZ13].

For example, consider the most basic task of message transmission. Typically, in order to transmit a message m securely, one encrypts m with a secure encryption scheme. However, note that given $\mathrm{Enc}(m;r)$ together with leakage $L(m;r)$, it may be possible to efficiently compute m, even if the amount of leakage is significantly smaller than the length of m. Bitansky, Canetti, and Halevi [BCH12] observe that if instead of using any secure encryption, one uses a *non-committing encryption* [CFGN96], then the message transmission becomes leakage tolerant.[8]

A non-committing encryption scheme, a concept that was developed for adaptively secure communication, allows one to generate a simulated (equivocal) ciphertext ct *without knowing a corresponding plaintext* and later given any plaintext m generate randomness r that explains this

[7] They formalize their notion in the UC framework, but in this survey we focus on the stand-alone setting.

[8] This observation was previously used in [BCG+11], in the context of constructing obfuscation with leaky hardware.

ciphertext; i.e., such that $\mathrm{ct} = \mathrm{Enc}(m; r)$. This ensures that the ciphertex does not leak additional information, beyond what is already leaked by the leakage function. Similar ideas were used in [BCH12] to construct leakage tolerant zero-knowledge, message authentication, commitment, and oblivious transfer protocols. In particular, to construct a leakage tolerant zero-knowledge protocol, rather than using a standard commitment scheme, they use equivocal commitments [FS90].

Ananth, Goyal, and Pandey [AGP14] extend the work of Garg, Jain, and Sahai [GJS11] (mentioned above) to the continual leakage setting. Namely, they construct an interactive proof for every language $L \in \mathsf{NP}$, such that any PPT verifier cannot learn a witness corresponding to $x \in L$, even after interacting *many times* with a prover who proves that $x \in L$ (for the *same* x), and even if in each such interaction a constant fraction of the prover's memory is leaked. Their formal requirement is that such an adversary cannot later convince an honest verifier that $x \in L$. Loosely speaking, this is done by encoding the witness using an encoding scheme that is robust to continual leakage.

General leakage-resilient MPC. While the works discussed above were for some specific interactive tasks, such as coin tossing and zero-knoweldge, the works Boyle et al. [BGJ+13, BGJK12] consider the task of constructing arbitrary two-party and multi-party secure computation that remain secure in the face of leakage. Namely, these works consider the setting where during the protocol execution, the state of the honest parties may be partially leaked. Clearly, one cannot hope to achieve "ideal world" security in the face of leakage, since the adversary can leak some of the bits of the input of the honest parties, and obtain information that is not leaked in the ideal world. To deal with this limitation, in [BGJ+13] the ideal world adversary is allowed to obtain some leakage. The difference between the model of [BGJ+13] and the leakage-tolerant model of [BCH12] discussed above is that [BGJ+13] allows both the real-world and the ideal-world leakage function to be a joint function of all the inputs, rather than locally computed for each party; in addition, [BGJ+13] allows the leakage length to be arbitrary (but the same in both the real and the ideal world). In contrast, the work of [BGJK12] does not allow leakage in the ideal world, but allows a leak-free preprocessing stage, where the secret inputs are pre-processed and shared among the parties before the adversary obtains any leakage. We now discuss these works in more detail.

Boyle et al. [BGJ+13] define the notion of multi-party protocols that are secure against *adaptive auxiliary information*. In their model, the adversary can corrupt an arbitrary subset of parties and, in addition, can learn arbitrary auxiliary information on the entire states of all honest parties (including their inputs and random coins), in an adaptive manner, throughout the protocol execution. There is no a priori bound on the amount of the auxiliary information that the adversary may be able to learn. Their protocol guarantees that for any amount of information the real-world adversary is able to (adaptively) acquire throughout the protocol, this "same amount" of auxiliary information is given to the ideal-world simulator, thus providing graceful degradation of security.[9]

For any (efficiently computable) functionality they construct a secure (two-party or multi-party) protocol that realizes this functionality securely against malicious adversaries in the presence of adaptive auxiliary input. Their protocols are in the common reference string model, and the security is based on the linear assumption over bilinear groups and on the nth residuosity assumption.

In [BGJK12], *continual memory leakage* was considered in the MPC setting. This is in contrast to [BGJ+13] and all the other leakage resilient protocols that were mentioned so far, which consider

[9] Note that it is not immediately apparent how to formalize this notion. We refer the reader to [BGJ+13] for details.

the single execution setting. [BGJK12] construct multi-party secure computation protocols that achieve standard ideal-world security (where no leakage is allowed in the ideal world) against real-world adversaries that may leak repeatedly from the secret state of each honest player separately, assuming a one-time leak-free preprocessing phase, and assuming the number of parties is large enough (larger than polylog(n), where n is the security parameter).

More specifically, they construct a multi-party computation (MPC) protocol that is secure even if a malicious adversary, in addition to corrupting $1 - \epsilon$ fraction of all parties for an arbitrarily small constant $\epsilon > 0$, can leak information about the secret state of each honest party. This leakage can be continual for an unbounded number of executions of the MPC protocol, computing different functions on the same or different set of inputs.

Interestingly, even though their MPC is secure against continual *memory* leakage, they achieve their result by relying on techniques from the *only computation leaks* (OCL) model (see Section 4.1). At a very high level, their basic idea is to run the MPC protocol of [BGJ$^+$13] that is resilient to adaptive auxiliary information, but rather than running the protocol on the underlying function, they run it on an OCL-compiled version of it. Roughly speaking, the OCL version has the property that local leakage does not leak any sensitive information. Therefore, even if all parties have leaked partial information at a certain point in the protocol execution, this leakage corresponds to local leakage in the underlying circuit, and since the underlying circuit is resilient to OCL leakage, no sensitive information is revealed.

This connection between continual memory leakage and the OCL model was further established in the work of Bitansky, Dachman-Soled, and Lin [BDL14]. Similarly to [BGJK12], they construct multi-party protocols in the continual leakage setting, but as opposed to requiring a leak-free *input-dependent* preprocessing phase, they only utilize a leak free *input-independent* preprocessing phase. As a result they can only achieve *leakage tolerance* (as opposed to leakage resilience). However, as opposed to [BGJ$^+$13], where the ideal world leakage is a *joint* function of all the inputs, in this work the real world leakage can be simulated by individually leaking on each party separately in the ideal world, thus giving a stronger security guarantee. Similarly to [BGJK12], their protocols are resilient to the corruption of $1 - \epsilon$ fraction of all parties for an arbitrarily small constant $\epsilon > 0$, where the number of parties grow with the security parameter.

Very recently, Benhamouda et al. [BDIR18] showed that in the honest-but-curious setting, and assuming the number of parties n is large enough, the GMW compiler [GMW87] implemented with a high-threshold version of the Shamir secret sharing scheme [Sha79], is robust against leakage one-time leakage in the preprocessing model. However, the leakage rate is quite small (roughly, $\frac{O(n)}{|C|}$ where C is the circuit the parties are computing). We refer the reader to Section 3 for further details.

3 Leakage from Storage

In this section, we consider the following generalization of exposure-resilient functions, mentioned in 1.1. Suppose a secret is encoded before being stored in memory; the adversary can repeatedly and adaptively apply a leakage function (from a set of allowed functions) to the encoding. The adversary's goal is to distinguish the stored secret from uniform. Thus, the security requirement for protecting the secret is stronger than in Section 2, where some information about the secret is allowed to leak as long as the leakage does not enable the adversary to break the underlying cryptographic scheme (e.g., encryption or signatures). On the other hand, the set of allowed

leakage functions, which will depend on the construction, will be generally more restricted than in Section 2.

This model, called "leakage-resilient storage," was introduced by Davi, Dziembowski, and Venturi [DDV10]. They propose two constructions, both secure only if the leakage is applied a bounded number of times (in their constructions, the encoding is not updated, which makes unbounded leakage impossible to achieve).

The first construction splits the stored secret into two components, and the assumption is that the two components leak independently (i.e., the two components are given to separate leakage functions rather than a single one; this model is known as the OCL model—see Section 4.1). Their construction uses a two-source extractor[10] 2-Ext as follows: To hide a secret $s \in \{0,1\}$, simply choose at random $u, v \in \{0,1\}^n$ such that 2-$\mathsf{Ext}(u,v) = s$, and store the string u in one component and the string v in the other.[11] The secret s is reconstructed by simply evaluating 2-Ext on the two stored strings u and v. This approach has proven quite fruitful, resulting, in particular, in the leakage-resilient encryption and signatures of [DF11] (Section 4.2.6) and circuit compilers of [DF12] (Section 4.3.4).

The second construction of [DDV10] does not require the leakage to be applied to two parts independently; rather, the leakage function is restricted to a limited complexity class. The idea is to use a deterministic extractor, instead of a two-source extractor. While deterministic extractors do not exist in general, Trevisan and Vadhan [TV00] constructed, for any polynomial time bound T, a deterministic extractor for sources that are sampleable in time T (and have sufficient min-entropy). Thus, if the leakage function is restricted to be computable in some a priori bounded time T (and its output length is also bounded), then one can store a secret s by simply choosing a random $u \in \{0,1\}$ such that $\mathsf{Ext}(u) = s$, where Ext is a deterministic extractor for T-time sampleableable distributions. Both constructions require no computational assumptions, except on the leakage function.

Protection against continual leakage requires the ability to update the stored secrets. In the OCL model (in which components leak independently), components should be updated before they leak too much information. Akavia, Goldwasser, and Hazay [AGH12] provide such a construction with two components, where the update requires interaction between the components. More generally, they construct a leakage-resilient public key encryption scheme, where the secret key is stored in two components, and the assumption is that the leakage on each component happens separately (we refer the reader to Section 4.2.6 for details). This scheme relies on computational assumptions; in particular it assumes that there exists a group with a bilinear map, for which the linear assumption holds and the Bilinear Decisional Diffie-Hellman assumption holds.

Eliminating communication during updates presents an additional challenge. This challenge was solved by Dodis et al. [DLWW11] (they also consider extensions to more than two components and allow full compromises of some). In their scheme, the updating of each component happens independently of the other, without the need for communication or synchronization. Technically,

[10] A two-source extractor produces an output that is close to uniformly random as long as the two sources are independent and each has sufficient entropy

[11] Storing a secret $s \in \{0,1\}^k$ that consists of many bits can be done in a bit-by-bit manner, but this approach can be secure only against $1/k$-fraction leakage of each component. To improve the leakage bound, we can use a two source extractor 2-Ext with k-bit outputs. However, it may be hard to choose at random $u, v \in \{0,1\}^n$ such that 2-$\mathsf{Ext}(u,v) = s$, since it may be hard to sample u and v given s. Instead, one can choose at random $u, v \in \{0,1\}^n$, let 2-$\mathsf{Ext}(u,v) = \mathsf{sk}$, encrypt the secret s using the secret key sk, and store (u, sk) in one component and store v in the other.

this work builds on [LLW11]: they encrypt the secret, store the ciphertext in one component and the secret key in the other component, and update both the key and the ciphertext, separately. This work also improves and simplifies the construction of [LLW11] for the continual leakage model (see Section 2.5). Their scheme assumes the existence of a group with a bilinear map, for which the linear assumption holds.

Faonio and Nielsen [FN17] consider the problem of leakage during the encoding process itself, to obtain so-called fully leakage-resilient codes. Leakage during the encoding process means that the secret cannot be completely protected; instead, the requirement is relaxed to leakage-tolerance of [BCH12] (see Section 2.6), in which the simulator is allowed to obtain some leakage on the secret.

Benhamouda et al. [BDIR18] consider storage of a secret in n shares produced via additive or high-threshold Shamir secret sharing over a prime field. Assuming each share leaks independently (i.e., in the n-component OCL model), they show that storage remains secure even if each share leaks about a quarter of its bits, for large enough n and field size. While this result requires many independently-leaking components, its advantage is that the secret sharing technique is standard, and readily usable in multiparty protocols. They use this result for secure computation (assuming leak-free preprocessing), in which each uncorrupted party can leak, once, a short function of its entire state.

Leakage-resilient storage is often an implicit ingredient in many constructions of leakage-resilient computation, because the master secret must be stored in a leakage-resilient way. Thus, many works discussed in Section 4 also provide some form of leakage-resilient storage.

4 Leakage from Computation

In this section, we consider leakage models that focus on adversary's access to the entire computation rather than just the secret memory. In general (with some exceptions, noted throughout this section), the goal of works discussed in this section is to protect against continual, rather than one-time, leakage. Thus, some models considered in this section are similar to models considered in Section 2.5, and some works could be placed into either section. On the other hand, the classes of leakage discussed in this section are typically more restricted than the classes of leakage discussed in Section 2.

The work on leakage from computation can be roughly divided into two categories: constructions of specific cryptographic primitives (Section 4.2) and general compilers that work for any cryptographic primitive and, in fact, for any computation (Section 4.3). There are, naturally, interactions between the two categories, and general compilation techniques are often applied to specific schemes, as we discuss throughout this section.

The most common leakage models are noisy or probabilistic leakage of each wire introduced in [CJRR99], wire-probing leakage of [ISW03], only-computation leaks (OCL) model of [MR04], and leakage of limited computational complexity introduced in [FRR+10]. There is considerable debate as to whether these models correctly capture actual side-channel attacks. Thus, heuristic, rather than fully provable, evaluation approaches are also common, because of the difficulty of capturing actual side-channel attacks with theoretical leakage models. We discuss these briefly in Section 4.4.

Because so many constructions are in the only-computation-leaks model, and because this model has slightly different variants and interpretations, we start by giving an overview of this model and its many versions.

4.1 The Only Computation Leaks (OCL) model

The general model of leakage during computation introduced by Micali and Reyzin [MR04] (see Section 1.2) contains one crucial assumption: the existence of leak-free memory. The model allows for values to be moved to that memory when they are not needed in a computation. Formally, the adversarial leakage function at each step of the computation takes as input the entire state of the Turing machine, including the values on its tapes, except the state of the leak-free memory. It is important to note, however, that leak-free memory does not mean leak-free values, because values in this leak-free memory cannot be used directly: they have to be read from the memory to the working tapes when needed for computation, and written from the working tapes into the leak-free memory when stored. Leakage functions have access to the values when they are on the working tapes and, in particular, during the reading and writing operations. (Recall that in the general model of [MR04], leakage functions come from some allowable class, and if the class is sufficiently limited, the adversary doesn't simply see whatever the leakage function sees.) A good analogy is a computer whose CPU, caches, and memory bus leak, but RAM doesn't. Alternatively, one can push the leak-free assumption one level lower in the memory hierarchy, and imagine a computer in which everything leaks except the hard disk. This assumption became known as "Only Computation Leaks Information," commonly abbreviated as OCL. See Section 4.2.1 for the first constructions in this model.

Dziembowski and Pietrzak [DP08] showed that the following special case of this general OCL model suffices to get strong results. In their model, the state of the computation is broken up into a few (specifically, three) parts. The computation proceeds in steps, and each step uses only some (specifically, two) of the parts. Each step leaks a bounded amount of information (specified by an adversarially chosen polynomial-time leakage function with a bounded output), and the part that is not used does not leak (i.e., is not given to the leakage function). See Section 4.2.2 for the first constructions in this model.

As pointed out by [DP08], the restriction on *when* each part leaks is not important for security; what is important, rather, is that the parts leak independently (i.e., any given leakage function does not have access to all of the parts at once), and only a bounded amount of leakage is available at each step of the computation. This independent leakage assumption became commonly used in many subsequent constructions of leakage-resilient cryptographic schemes (Section 4.2) and leakage-resilient storage (Section 3).

The OCL assumption was also used for the purpose of building general leakage-resilient circuit compilers in the style of [ISW03] (see 1.2 and 4.3.1) rather than specific cryptographic schemes. This line of work, discussed in Sections 4.3.4 and 4.3.5, assumes that the transformed computation can be broken up into parts that leak independently. Each part can leak an arbitrary (or, depending on the model, any polynomial-time) function of its state, as long as the output size of the function is bounded. Since the leakage function on each component is powerful enough to simulate the inner wires of the component, we do not need to provide the wires explicitly as inputs to the leakage function; it suffices to provide the inputs and the randomness used in each component. Thus, the situation for each component is similar to bounded memory leakage (see Sections 2.1 and 2.2), and techniques for protection against such leakage are often helpful in this setting.

This line of work can be interpreted in the original OCL model of [MR04], in which the CPU leaks and memory does not. Each component corresponds to reading some data from memory, performing the component's work on the CPU, and writing the data back. It can also be interpreted in the circuit model of computation (like the work of [ISW03]); the circuit is broken up into separate

topologically ordered components, and the leakage function specified by the adversary is limited to working separately on the wires of each component (again, for each component it suffices to give the leakage function only the the wires going into it and the randomness generated within it). The latter model is articulated in [GR10]. The connection between the models is explained in, for example, [GR15, Section 1.2].

Constructions in the OCL model can be also naturally viewed as protocols between two or more stateful parties; the adversary can obtain leakage from each party, but the leakage is independent for each party. Parties can correspond to circuit components in the previous paragraph, with inter-component wires modeled as inter-party communication. More generally, however, each party can be invoked more than once per execution of the protocol, and so there may be fewer parties than components (every invocation of a party corresponds to writing and reading non-leaking memory in the model of [MR04] and to a new circuit component in the model of [GR10]). The parties are assumed to be able to erase parts of their state that they are no longer using (else the adversary could obtain unbounded leakage about the first invocation by leaking information in subsequent invocations). This model is articulated in [DP08] and [JV10] for the two-party setting; the observation that the number of parties can be flexible is made in [DF12]. For some protocols, such as [DP08] and [JV10], communication between the parties is fully available to the adversary; for others, such as [DF12], it counts against the adversary's leakage allowance (the adversary can use the leakage function to compute sent messages; received messages are given as input to the leakage function of the receiving party).

Several papers observed that their constructions are secure against a stronger class of leakage functions than just OCL as defined in [MR04]: namely, leakage need not be restricted to computation. The adversary can obtain leakage from any of the parties at any time, repeatedly and adaptively, as long as the amount of leakage is bounded. This bound may be per party, as in [BCG$^+$11, DF12], or total, as in [GIM$^+$16]. This view is equivalent to having leakage computed by viruses that have infected all the parties but have limited ability to communicate with each other (virus communication messages correspond to the outputs of the leakage functions); [GIM$^+$16] call it "bounded-communication leakage" or BCL (note that "communication" here refers not to the computing parties, but to the leakage functions).

This connection between the OCL model and the multi-party protocol model was made more formal and exploited by several works (e.g., [BGJK12, BDL14, DDN15, DLZ15, BDIR18]—see Sections 2.6, 3, and 4.3.4).

It should be noted that the leakage functions in the OCL model need not necessarily be limited by the number of output bits, although this is how the limitation on the leakage functions is most commonly stated. What matters, informally, is the amount of useful information contained in the leakage. In particular, if the leakage is noisy, it may be able to hide information even if it's long (see, in particular, Section 4.3.5).

4.2 Specific Schemes

Because leakage can occur during every computation on a given secret key, the main challenge in most constructions discussed in this section is to evolve the secret key (while securely erasing the previous versions), so that repeated leakage of, for example, one key bit at a time cannot lead the adversary to discover the entire key. In this way, the problems considered in this section are often similar to the problems encountered in the continual memory leakage model discussed in Section 2.5. Such key evolution is generally harder to achieve for public-key primitives, because

the public key must remain the same as the secret key changes.

Similarly to works on the continual memory leakage model, most works discussed in this section assume that key generation is completely leak-free, and that secure erasure is possible — once erased, values do not leak. However, in contrast to continual memory leakage, most constructions discussed here assume OCL leakage model described in Section 4.1.

4.2.1 Pseudorandom Generators of [MR04]

Micali and Reyzin [MR04] showed constructions of leakage-resilient pseudorandom generators out of simpler leakage-resilient building blocks (such as leakage-resilient one-way permutations). These "physical reductions" are analogous to cryptographic reduction based on complexity-theoretic assumptions. This approach makes assumptions on the leakage of the building block as it processes data, but allows full leakage whenever other code is executed. The reasoning behind this approach is that it may be easier for hardware designers to protect a simple building block.

Specifically, the work of [MR04] shows that if the output of a length-preserving one-way function is indistinguishable from random even given the leakage, then the Blum-Micali [BM84] construction (specifically, iterating the one-way function) with the Goldreich-Levin [GL89] hardcore bit (used as an extractor to "remove" the leakage) is next-bit-unpredictable when the bits are output in reverse order. The same paper also showed that indistinguishability is harder to achieve than unpredictability. Subsequent work on unpredictable generators (which became known as "leakage-resilient stream ciphers") is discussed in Sections 4.2.2 and 4.2.3.

4.2.2 The Power of Only-Computation-Leaks: The Stream Cipher of [DP08]

The remarkable power of the only-computation-leaks (OCL) assumption was demonstrated by Dziembowski and Pietrzak [DP08], who built a stream cipher that provably provides leakage resilience based on very mild assumptions. In addition to the OCL assumption, they assume that a bounded number of bits is leaked during an evaluation of two basic cryptographic primitives: a pseudorandom generator and a randomness extractor. They do not make any other restrictions on the leakage function: in fact, like in the model of [MR04], the adversary can choose any leakage function to be applied to the currently used portion of the state, as long as it is efficiently computable and its output is not too long. More generally, the leakage function can have arbitrary output length, as long as the secret maintains (pseudo)entropy given the leakage.

The specific use of the OCL assumption in [DP08] is quite simple. The stream cipher proceeds in rounds, outputting a fresh string of pseudorandom bits in each round and evolving its state. The stream cipher state is stored in three variables: two variables M_0 and M_1 that are used and updated in alternate rounds (never together), and the third variable K that is used and updated in every round. The one variable not used in the current round is assumed not to leak (equivalently, is stored in non-leaky memory); formally, it is not given as input to the leakage function. The variable K that is used in every round can be fully public without compromising security.

Dziembowski and Pietrzak also pointed out that in their setting, the OCL assumption can be viewed simply as a restriction on the leakage function. Instead of assuming that some parts of the state do not leak, we can simply assume that a separate leakage function is applied to different parts the state. In other words, different parts of the state leak independently rather than jointly. This view of the OCL assumption was adopted by many subsequent works.

The construction of [DP08] works as follows. Let G be a pseudorandom generator (PRG). The nonsecret variable K is an extractor seed. In each round ℓ, K is used to extract three values from M_i (where $i = \ell \bmod 2$): the stream cipher output bits, a new value for the extractor seed K, and a PRG seed X. M_i is then replaced with $G(X)$. Note that in this construction, the extractor seed that is used for M_i is itself extracted from M_{1-i} in the previous round, using a seed extracted from M_i in the round before, and so on. This technique, introduced in [DP07], is known as alternating extraction. As already shown in [DP07], if M_0 and M_1 start with sufficient entropy, alternating extraction will keep producing uniform values even in the presence of leakage, as long as the leakage function does not get to see M_0 and M_1 simultaneously. Alternating extraction is not enough, however, because it works only until the information-theoretic entropy of M_0 and M_1 is exhausted. To make a stream cipher that outputs more random bits than its seed, Dziembowski and Pietrzak introduce the second ingredient: the PRG, which replaces limited information-theoretic entropy with as much computational entropy as needed. To prove security of the overall scheme, they had to prove that a PRG will work even in the presence of leakage (i.e., when the PRG seed X is not uniform to the adversary). This result, independently also shown in [RTTV08], became known as the "dense model theorem": it quantifies the amount of entropy in a PRG output given a certain amount of leakage from the PRG seed or computation (see [FR12] for an entropy-based formulation). We note that PRGs secure against specific leakage (rather than arbitrary bounded leakage of dense model theorem) have also been considered—e.g., [ISW03, IKL$^+$13].

Note that because the stream cipher never needs to output past values, the construction of [DP08] is able to update the secret state in a one-way fashion. This fact allows the construction of [DP08] to be more efficient than the construction of [ISW03], which is forced to create fresh randomized representations of the same logical secret state in order to allow for general computations, and thus must use fresh randomness at each iteration and work with a state that is represented via XOR-based secret sharing (also known as masking).

4.2.3 More Leakage-Resilient Stream Ciphers

Following the breakthrough result of [DP08], work continued on provably secure leakage-resilient symmetric encryption and pseudorandom objects, such as stream ciphers, pseudorandom functions (PRFs), and pseudorandom permutations (PRPs, also known as block ciphers). A number of results offered various tradeoffs between construction complexity, assumptions used, and security achieved. We briefly mention only some of the relevant work.

Pietrzak [Pie09] simplifies the construction of [DP08] by assuming a stronger underlying primitive (a so-called weak PRF instead of just a pseudorandom generator used in [DP08]).

Standaert et al. [SPY$^+$10] argued that a different leakage model than OCL may be more reflective of real side-channel attacks and may also improve efficiency of constructions. The difficulty in designing a good leakage model is that without sufficient restrictions on the leakage class, the adversarially supplied leakage function can perform a "precomputation" attack, in which the leakage function precomputes the value that the pseudorandom object would output far in the future, thus making the value no longer random-looking when it is finally output. To design a leakage class that is both reflective of reality and prevents these theoretical attacks is a difficult task (OCL is one such design). Standaert et al. suggested not allowing the adversary to choose the leakage function adaptively (as already suggested in [MR04]), or employing a random oracle that can be queried by the construction, but not by the leakage function. Both of these leakage models were considered by [YSPY10]; following the discovery by [FPS12] of a mistake in one of the proofs of [YSPY10], fixes

and further improvements were proposed by [YS13]. The random oracle of [YSPY10] is replaced by a so-called "simulatable leakage" assumption in [SPY13], where it is argued that though the assumption may seem strong, it is more realistic than length- or entropy-based restrictions on the leakage function; see [LMO+14] for a discussion on how to break various simulators and [FH15] for connections between simulatable leakage and other leakage-function restrictions.

Leakage-resilient pseudorandom generators "with input" (i.e., whose state can be continually updated by additional input) are considered in [ABP+15].

4.2.4 Leakage-Resilient Key Evolution

One-way key evolution, which is the main ingredient in leakage-resilient stream ciphers, was considered as a separate primitive by Dziembowski, Kazana, and Wichs [DKW11]. Like the authors of [YSPY10], they work in the random oracle model. However, they do not assume that the leakage function cannot evaluate the random oracle; instead, they assume the leakage function is space bounded, and use graph pebbling problems to protect against such leakage. They show applications of their construction to authentication and to obtaining security against continual leakage in the bounded retrieval model (see Sections 2.4 and 2.5). Their construction was improved by [SZ13].

4.2.5 Leakage-Resilient Block Ciphers, Encryption, and Authentication

A significant stumbling block for achieving efficient leakage-resilient constructions of PRFs, PRPs, and higher level symmetric primitives, such as encryption and authentication, is the fact that the secret state does not naturally evolve in the mathematical description of the primitive, in contrast to stream ciphers, which naturally evolve their secret state in a one-way fashion. The state does not naturally evolve for PRFs and PRPs because they need to repeatedly produce the same output on the same input. Higher-level primitives, such as encryption and authentication, have multiple participating parties who cannot be assumed to update the state synchronously (in particular, what was encrypted yesterday needs to still be decryptable today).

Such primitives are sometimes called "stateless" in the literature (which is a bit of a misnomer, because they have a secret state—they just don't change it), in contrast to "stateful" stream ciphers discussed above. If such a primitive is used repeatedly with the same secret state, and the leakage class is sufficiently rich, then the adversary will eventually obtain the entire secret state.

General compilers discussed in Section 4.3 can be used for any cryptographic primitive and, therefore, can be used to address this challenge. Some works have optimized general compilation techniques for particular symmetric primitives, especially block ciphers. We review these approaches in Sections 4.3.2, 4.3.5, and 4.4. For the remainder of this section, we focus on approaches that have less general applicability. Many of these approaches split the secret key into multiple parts that can evolve even when the secret key remains the same, and thus provide some form of secure storage (see Section 3) in such a way that the stored value can be used in the computation by the symmetric primitive.

Dodis and Pietrzak [DP10] get around the problem of evolving state for PRFs and PRPs by limiting the leakage class: they consider nonadaptive OCL leakage, in which the adversary must fix the leakage function in advance and keep it the same every time the PRF or PRP is invoked. They construct a PRF and a PRP that are reslient to such nonadaptive OCL leakage without the need for key evolution. They also show generic side-channel attacks on Feistel-based PRP constructions. Faust, Pietrzak, and Schipper [FPS12] consider models in which the adversary does not get to

choose the leakage function and/or the inputs adaptively, showing that these relaxations lead to more efficient constructions of PRFs and PRPs secure against OCL leakage.

Another way to get around the problem of evolving state is to force all participants to evolve it. In particular, leakage-resilient MACs in which both sides evolve the secret key were considered by Schipper [Sch10].

Some states can be easily split into multiple evolving components using algebraic techniques (instead of more traditional symmetric primitives), even when the underlying secret (which is never reconstructed) does not evolve. Following ideas from the public-key encryption scheme of [KP10] (discussed in Section 4.2.6), Martin et al. [MOSW15] use bilinear groups (in the generic group model) to construct a leakage-resilient MAC in the OCL model. The construction splits the secret into two parts multiplicatively and assumes the two parts leak independently. Since their scheme does not allow leakage during verification, it can be seen as a weaker variant of a PRF, with output that is unpredictable rather than pseudorandom. Barwell et al. [BMOS17] demonstrate both a PRF and a MAC that resists leakage during verification using a three-share variant of this construction. Note that bilinear pairings are considerably less efficient than typical block-cipher-based MAC constructions, though they are competitive with public-key schemes.

Andrychowicz, Masny, and Persichetti [AMP15], propose, as an application of their general compiler discussed in 4.3.4, a particularly efficient leakage-resilient implementation of interactive secret-key authentication protocol Lapin [HKL$^+$12]. The construction splits the secret into two parts that are assumed to leak independently, using the inner-product extractor (see Section 3) over large finite fields.

Pereira, Standaert, and Vivek [PSV15] obtain symmetric encryption and MACs by combining a leak-free block cipher in which the key does not evolve with a leaking primitive that evolves its key, emphasizing that the leak-free primitive is more expensive and thus used sparingly. The key of the leak-free block cipher is the master key of the entire scheme, and is used to generate temporary keys for the leaky primitive. The approach of generating temporary keys using a master key is sometimes called re-keying. While [PSV15] assume a leak-free primitive for re-keying, some works design leakage-resilient re-keying schemes: at each invocation, such a scheme generates a fresh key for a stream cipher and updates its own state. Re-keying was addressed in theory and practice well before leakage-resilient cryptography was formalized (e.g., [AB00, Koc03]); in the context of leakage-resilience, see [ABF13, DFH$^+$16], and references therein. The idea of combining a low-leakage (expensive to implement) primitive with a higher-leakage (inexpensive) one is sometimes called the "leveled leakage setting".

Authenticated symmetric encryption (which protects both secrecy and authenticity of the message against chosen-ciphertext attacks) presents more opportunities for leakage, because, in addition to leakage during computation, the decryption oracle may leak information about how exactly an invalid ciphertext failed to decrypt. This problem was addressed via generic composition of leakage-resilient PRFs, MACs, and symmetric encryption in [BMOS17], and via the leveled approach (as discussed in the previous paragraph) in a series of works (see [GPPS18] and references therein); some of these works also provide protection in case of poor randomness or nonce generation. One suggestion for implementing the expensive PRF is to use the bilinear-pairings based PRF construction of [BMOS17].

It's important to note that there is no consensus on the leakage model for symmetric encryption schemes, because a single bit of leakage about the plaintext trivially breaks the standard indistinguishability notion. Some works (e.g., [BMOS17]) prohibit leakage during the challenge phase;

others (e.g., [PSV15, GPPS18]) permit it, but provide designs that first hide the plaintext via some operation assumed to leak nothing useful.

4.2.6 Leakage-Resilient Public-Key Objects

Micali and Reyzin [MR04] construct the first leakage-resilient signature scheme in the OCL model. Specifically, the observe that the following classical stateful signature scheme is already leakage-resilient in the OCL model: the public key is the root for a Merkle tree [Mer88] of one-time public keys, where each one-time public key is for Lamport's one-time signature scheme [Lam79]. Leakage resilience in the OCL model is trivial, because the model assumes there is no leakage during key generation, and after key generation, there is no computation on secret values, except to output some of them as part of a signature. The proposed scheme requires an a priori bound on the total number of signatures that will ever be produced and key generation time that is proportional to that bound; it is also stateful.

Faust et al. [FKPR10] reduce key generation time and remove the a priori bound on the number of signatures by replacing the Merkle tree in the signatures of [MR04] with a signature tree. They observe each secret signing key is used at most three times (to sign two leaves and a message), and therefore if the underlying signature scheme is resilient against memory leakage that results from three signatures, the resulting tree-based signature scheme will be leakage-resilient in the OCL model. This signature scheme is still stateful, however.

Malkin et al. [MTVY11], building on techniques of [ADW09, KV09, BKKV10] for memory leakage (see Section 2.1), construct signature schemes that resist leakage during the signing process without the OCL assumption.

Kiltz and Pietrzak [KP10] construct a leakage-resilient public-key encryption scheme resistant against continual leakage in the OCL model (however, unlike the one-time leakage results discussed in the previous paragraph, in their model no leakage is allowed once the challenge ciphertext is given to the adversary). The main idea of their construction is as follows. Start with ElGamal encryption [ElG85], but use bilinear groups (i.e., a bilinear pairing operator e that takes two elements of a source group into a single element of a target group) in order to enable multiplicative sharing of the secret key. That is, instead of the usual secret key x, let the secret ket be g^x in the source group, where g is the group's generator. The public key is its image in the target group, $X = e(g^x, g)$. Encryption is the usual ElGamal, except the the first component is in the source group: an encryptor chooses a random r, outputs g^r, and uses X^r as a symmetric key to encrypt the message. Decryption is done by first computing $e(g^x, g^r) = e(g^x, g)^r = X^r$. To make this scheme leakage-resilient, multiplicatively share the secret key g^x into two shares stored in two separate components, and decrypt by working with each share separately within each component and multiplying the results. To obtain security against continual leakage, rerandomize these shares at every decryption. Both decryption and update require a single message between the two components. Note that to obtain security, it is essential for leakage resilience that x is stored in the exponent, because additive secret sharing of x could allow an adversary to obtain sensitive information about x via OCL leakage.

Kiltz and Pietrzak show that this scheme is CCA1-secure in the presence of OCL leakage (i.e., independent leakage from the two shares of the secret key) in the so-called *generic group model*, an idealized model in which group elements are assumed to have random representations that leak only equality information. Galindo et al. [GGL+16] show a software implementation of a variant of this scheme, and then evaluate the implementation to determine whether the amount of leakage is indeed sufficiently small per invocation, as required for security to hold.

Galindo and Vivek [GV13b, GV13a] and Tang et al. [TLNL14] adapt the approach of [KP10] to digital signatures, basing their schemes on identity-based encryption (IBE) and signatures schemes of [BB11, BLS04, Wat05, Sch91]; Wu, Tseng, and Huang [WTH16] extend it further to identity-based signatures.

Instead of multiplicative sharing of [KP10], Dziembowski and Faust [DF11] use the inner-product-based sharing introduced in the leakage-resilient storage work of [DDV10] (see Section 3) to construct CCA2-secure encryption (that handles even post-challenge leakage), identification schemes, and signature schemes in the OCL model. They build on ideas of [DDV10] and on work in the memory leakage model, such as [NS09] (see Section 2.2) and [ADW09] (see Section 2.4). Their schemes operate in a prime-order group with generators g_1, g_2; the secret key for each scheme is a pair of values x_1, x_2, and the public key is $g^{x_1 x_2}$ (thus ensuring, as in the continual memory leakage model of Section 2.5, that there are multiple secret keys for each public key). The secret key is shared into two parts, L and (R_1, R_2) (where L, R_1, R_2 are vectors), so that the inner product of L and R_i is x_i for $i = 1, 2$. The encryption scheme is similar to ElGamal [ElG85] (and similar to [NS09]), while the identification and signature schemes are based on those of Okamoto [Oka93] (which were analyzed in the bounded rertrieval model by [ADW09]). The most innovative part of this work is a two-message protocol to update the shares L and (R_1, R_2) in a way that ensures security even if the adversary can obtain leakage during the protocol. The protocol requires a leak-free component that samples pairs of values from a fixed, input-independent distribution (this assumption is considerably weaker than the assumption of leak-free updating made in the many works discussed in 2.5). The ideas of this work led to a general compiler by [DF12] discussed in Section 4.3.4.

Akavia, Goldwasser, and Hazay [AGH12] consider a model very similar to the two-component OCL model of [KP10] and [DF11]: there are two parties who hold shares of the secret and communicate over a public channel; the parties' secrets leak independently. In this model they construct CPA-secure public-key encryption and IBE, as well as CCA2-secure public-key encryption (using the IBE-to-CCA transformation of Boneh et al. [BCHK07]); no post-challenge leakage is allowed. They do not require idealized models or leak-free components. The main idea is to share the master secret key g^α of the Boneh-Boyen [BB04] IBE between the two parties via encryption that is similar to Naor-Segev [NS09], with one party holding the secret key and the other holding the ciphertext. Both decryption and share updates are accomplished by a two-party two-message protocol that (again) uses Naor-Segev-like encryption, relying on its homomorphic properties. This scheme can also be used for leakage-resilient storage (see Section 3), using the interactive updating protocol to update the stored shares.

Barthe et al. [BBE+18] show how to implement the lattice-based signature scheme of Güneysu, Lyubashevsky, and Pöppelmann [GLP12] in the wire-probing model of [ISW03], using many of the recent advances developed for masking-based circuit transformations (see Section 4.3.2), as well as developing additional techniques, such as conversion between masking modulo 2 and modulo a large prime.

We close this section by discussing a few works that address one-time leakage rather than continual leakage discussed above. (Most work addressing one-time leakage is discussed in Section 2; we single out the following works for this section because they work in the OCL model.) Halevi and Lin [HL11, Section 4], building on their result that the Naor-Segev [NS09] construction maintains entropic security against memory leakage even if it occurs after the challenge ciphertext is known to the adversary (see Section 2.2), show how to build a public encryption scheme in the 2-state

OCL model that is CPA-secure for one-time post-challenge leakage. The idea is to store two secret keys separately, use each of them to decrypt a random string, and use the inner product of the two random strings (which is a two-source extractor—see Section 3) to decrypt the message. Zhang, Chow, and Cao [ZCC15] show how to upgrade this scheme's security to CCA, as well as how to construct IBE schemes by building on techniques from [ADN+10]. Fujisaki et al. [FKN+15] show a similar upgrade to CCA security as well as security against leakage from the encryptor's randomness.

4.3 General Compilers

While Section 4.2 discussed specific cryptographic primitives, here we discuss general transformations to achieve leakage resilience for any computation. They are, of course, also applicable to the specific cryptographic goals discussed above, but often less efficient than the specific constructions.

The commonly used paradigm for general leakage-resilient compilers was introduced by Ishai, Sahai, and Wagner [ISW03] (see Section 1.2). To recap, they address the situation in which computation is performed by a clocked circuit with a secret state (for example, a circuit implementing a block cipher with a secret key). The circuit is run repeatedly on various inputs, producing outputs and possibly also updating the secret state. They consider adversaries who are able to provide inputs and observe outputs as well as observe some leakage function of the internal wires during the computation. The security goal is to build a circuit in such a way that the adversary learns nothing useful about the secret state from the leakage. The notion of "learning nothing useful" is defined by the existence of a simulator who faithfully simulates the leakage by observing only the input/output behavior. The initial secret state is stored in some specially encoded form and is assumed to be placed into the circuit without any leakage. In order to protect against repeated leakage on multiple inputs, constructions must update the secret state and erase the previous version, similar to constructions in Section 4.2.

General compilers achieve this security goal for any computation. The computation itself is specified by a stateful, but not leakage-resilient, circuit C. The goal of a compiler is to create a new circuit C' (and an encoding of the secret state) so that C' computes the same functionality as C and is leakage-resilient in the sense described above.

The specific leakage function considered by [ISW03] was wire probing: the adversary could obtain leakage from t wires. We discuss their construction in Section 4.3.1. We cover other transformations secure against wire-probing leakage in Section 4.3.2.

Following the introduction of general leakage functions in [MR04], researchers have considered other types of leakage. A folklore result, attributed to Impagliazzo by [GR15, Section 1], is that general leakage-resilient computation is impossible under even a single bit of leakage without some constraint on the leakage function, because of the general impossibility of black-box obfuscation [BGI+01] (the connection between leakage-resilient computation and obfuscation has been also explored by other works—see, e.g., [BCG+11]). Thus, some restrictions on the leakage functions, besides the amount of leakage, are necessary.

Transformations secure against a variety of leakage classes are discussed in Sections 4.3.3 (leakage of limited complexity), 4.3.4 (OCL leakage), and 4.3.5 (noisy and noisy OCL leakage).

Before proceeding, we should note the following folklore result (see, e.g., [BCG+11, Section 1.1]): to achieve a general compiler secure against some leakage, it often suffices to build a leakage-resilient construction for decryption of a fully-homomorphic encryption scheme. The secret state

can then be stored encrypted under such a scheme, and all computation and state update can be carried out encrypted until the output is needed.

4.3.1 The Compiler of [ISW03]

The transformation of [ISW03] is similar to the one in [CJRR99]: each wire carrying a bit b is a replaced by a bundle of $t+1$ wires carrying the boolean masking of b, i.e., $t+1$ bits whose exclusive-or is equal to b. The main technical tool is the design of a gadget for the logical AND operation: it takes two wire bundles for bits b_1 and b_2 and outputs a wire bundle for the bit $b_1 \cdot b_2$, in such a way that the adversary cannot learn anything by observing t wires, because the distribution of wire values is t-wise independent. The gadget is made up $\Theta(t^2)$ bit gates and uses $\Theta(t^2)$ random bits.

The secret state is stored encoded in the same way: each bit b is replaced by $t+1$ bits that XOR to b. Inputs are encoded and outputs are decoded to the same representation (leakage during encoding and decoding is not a concern, because the adversary is assumed to be able observe inputs and outputs). The encoded secret state is updated (rerandomized) before being stored again, whether the actual secret state changes or not.

As already mentioned, this construction is secure against continual leakage. At its core is a transformation secure against one-time leakage. Specifically, given a stateful circuit C, treat initial state as an additional input and the updated state as an additional output, resulting in a circuit $\tilde{C}$ that has state, but only inputs and outputs. The goal of a one-time-secure (also known as stateless) transformation is to transform $\tilde{C}$ into $\tilde{C}'$ that leaks nothing useful about its input. To enable such a transformation, we will allow $\tilde{C}'$ to receive its input already encoded, and to produce encoded outputs. The stateful C' that is secure against continual leakage is produced by taking $\tilde{C}'$, storing the encoded state in memory registers, and adding input encoding and output decoding.

One-time-secure (stateless) transformations are sometimes interesting on their own. They do not always result in secure transformations against continual leakage, because it is not always possible to update the secret state so that cumulative leakage does not add up to reveal it.

The transformation of Ishai, Sahai, and Wagner [ISW03] achieves perfect security. The authors also show more efficient transformations for large values of t that achieve statistical security, and a derandomized construction that achieves computational security.

4.3.2 Improved Compilers for Wire Probing Leakage

Considerable effort has been devoted to improving the compiler of [ISW03].

Many subsequent papers improved efficiency of [ISW03]. Some papers design special masking-friendly block ciphers (e.g., [PRC12], [GLSV15]) or more efficient masking techniques (see, e.g., [GM17], [GR17], [JS17], and references therein). Some consider automated synthesis and verification of masked circuits for specific computations—see, in particular, [BBD+15, BBD+16, BBP+16, Cor18, BGI+18, BGR18] and references therein (a good overview of this area is given in [BDF+17, Section 1.2]). Some reduce the amount of randomness used (e.g., [BBP+16, BBP+17, FPS17]). Some consider both Boolean masking and masking modulo a power of two (see [BCZ18] and references therein) or a large prime (see [BBE+18]); the ability to switch between the two gives more efficient implementations. Masking is not the only countermeasure used in this setting—see, e.g., [CRZ18] for a randomized table countermeasure and a discussion of other countermeasures used. Even though block cipher constructions are the primary goals of these works, many of them present

techniques of general applicability. Some works combine leakage-resilience with resilience to glitches (e.g., [FGP+18]).

Many of the works mentioned above try to optimize not only the circuit size, but also the amount of randomness. Ishai, Sahai, and Wagner [ISW03] showed that if we are willing to settle for computational, rather than information-theoretic security against leakage, then their construction can be fully derandomized (except for an initial random seed) with the help of a leakage-resilient pseudorandom generator that they construct. For the case of perfect security, the randomness complexity is improved from t^2 per gate to $t^{1+\epsilon}$ for the entire circuit in [IKL+13, AIS18], with the help of different leakage-resilient (so-called "robust") pseudorandom generators (t random bits are necessary according to [AIS18]).

A series of works by Balasch et al. (see [BFG+17] and references therein) considers so-called "inner-product" masking instead of boolean masking. It presents both general compilation techniques and applications to AES. This basic idea is similar to [ISW03]: replace wires with wire bundles, and gates with gadgets. However, this masking operates on words rather than bits, so, to start with, a "wire" carries b-bit elements of the finite field GF(2^b). Like in [FRR+10] (see Section 4.3.3), the masking operation replaces each such wire with a wire bundle whose inner product with a fixed vector (which is a system parameter) is equal to the wire value. We note that this usage of the inner product operation is different from how the inner product is used in [DF12] (see Section 4.3.4), where a wire is represented by two vectors whose dot product is equal to the wire's value, because in [DF12] both vectors are random, while in [BFG+17] one vector is a fixed parameter. The value of this fixed parameter is of little importance to the theoretical evaluation (as long it has no zero coordinates), but matters to the heuristic security evaluation: in addition to theoretical security evaluation, these and other similar works are evaluated in heuristic evaluation frameworks we discuss in Section 4.4.

On the more theoretical side, a number of works considered the problem of leakage rate (i.e., the ratio of leaking wires to total wires in the compiled circuit). Because the circuit size in the construction of [ISW03] increases by a factor of t^2 during compilation, the leakage rate is quite low and, in fact, decreases linearly as t increases. If the choice of leaking wires is not completely up to the adversary (for example, each wire leaks with some probability, or not too many wires leak in any particular region of the circuit), then the leakage rate can be improved to a constant [Ajt11, ADF16, AIS18].

4.3.3 Compilers for Leakage of Limited Complexity

Faust et al. [FRR+10, FRR+14] showed two compilers. Both compilers, in addition to the leakage-class restriction, assume the existence of certain leak-free hardware (which is input-independent), thus providing a reduction from a simple leak-free piece of hardware to a general leak-free circuit, in the spirit of [MR04]. The first compiler provides security against noisy leakage of *every* wire; we discuss it in Section 4.3.5. Here we focus on the second compiler of Faust et al., which is secure against a class of leakage functions that cannot decode a linear secret sharing scheme (the specific linear secret sharing scheme determined the class of leakage functions that could be tolerated). In particular, by using the same boolean masking as used by [ISW03], but different AND gadgets, the compiler achieves security against leakage functions in the complexity class AC^0 (i.e., leakage functions computable by unbounded fan-in constant-depth circuits with "and", "or", and "not" gates). It is not practical: to tolerate leakage of λ bits of information per round of execution, the circuit size has to increase by a multiplicative factor of more than λ^{12}. Its theoretical efficiency was

been improved in subsequent work [ADD+15], using techniques from multi-party computation (in particular, working over large fields and using packed secret sharing), although concrete parameters are not analyzed. It is improved to withstand more leakage, and, in a surprising application, used to construct zero-knowledge PCP by Ishai, Weiss, and Yang [IWY16].

Several subsequent papers improved protection against leakage functions from a restricted complexity class. Rothblum [Rot12] improved the AC^0-leakage compiler of [FRR+10] to remove the need for leak-free hardware, but at the cost of adding a computational hardness assumption. This transformation (which builds on the ideas of [GR12, GR15] discussed in 4.3.4) replaced the leak-free hardware with a leakage-resilient computation, and required changes to the wire-bundle encoding and gate gadgets in order to make simulation possible.

Miles and Viola [MV13] proposed a circuit transformation that resists more powerful classes of leakage functions, such as AC^0 augmented with gates that compute any symmetric function (including parity), and, under certain computational assumptions, the class TC^0 (i.e., leakage functions computable by unbounded fan-in constant-depth circuits with "threshold" and "not" gates). Their transformation follows the wire bundles and gadgets approach of prior work, but uses group operations over the alternating group A_5 instead of boolean masking for sharing each wire (and, of course, completely new gadgets). Miles [Mil14] extended this result to leakage functions in NC^1 (all leakage functions computable by polynomial-size logarithmic-depth constant fan-in circuits) under the assumption that $L \neq NC^1$. These compilers, like those of [FRR+10], require an input-independent leak-free hardware. While precise parameters are not analyzed, they do not seem to be in the realm of practical.

The above work is for continual leakage from stateful circuits. For the more limited case of one-time leakage from circuits without persistent state (see Section 4.3.1), Bogdanov et al. [BIVW16] showed that constructions secure against wire-probing leakage of t wires also achieve security against low-complexity leakage, where "low-complexity" means low approximate degree of the leakage function. The main technical insight is an equivalence between the notion of low approximate degree of a function and the function's inability to distinguish t-wise indistinguishable distributions (i.e., distributions whose projections on t symbols are identical). This result is similar to the result of [DDF14] for the connection between wire-probing and noisy leakage (see Section 4.3.5). Bogdanov et al. exploit the connection between secure multi-party computation and circuits resilient to wire-probing leakage (observed already in [ISW03]) to obtain new constructions of circuits resilient to one-time low-complexity leakage. However, it is not known how to extend their ideas to stateful circuits with security against continual leakage.

4.3.4 Compilers for OCL leakage

See Section 4.1 for a discussion of the "only computation leaks" (OCL) model and its variants.

Two general compilers in the OCL model were shown by Juma and Vahlis [JV10] and Goldwasser and Rothblum [GR10], using very different approaches.

Juma and Vahlis presented their result in two-component OCL model. One component stores the secret state encrypted under a public key for a fully homomorphic encryption scheme (FHE). The other component stores the FHE secret key. The facts that the two components leak separately and only a bounded amount are used to prove that information about the FHE plaintext is not accessible to the leakage function. In order to evaluate a circuit C, leakage-resilient computation is performed homomorphically under the cover of FHE by the first component; the result is then decrypted with the help of the second component. At the same time, fresh FHE keys are generated

to update the state of the second component, and the component's state is re-encrypted under these keys (using decryption under the cover of the FHE) to refresh the ciphertext. The amount of leakage per invocation that this construction can tolerate is logarithmic in the FHE security; the leakage function must be polynomial-time computable. The construction depends on an input-independent leak-free component that produces FHE ciphertexts for a fixed (e.g., all-zero) plaintext.

Goldwasser and Rothblum [GR10] divide the computation into many more independently leaking pieces — as many as gates in C. They use a leakage-resilient encryption scheme (with additional properties) as the underlying building block. They replace each wire value of the original circuit C with its ciphertext, and each gate of C with a gadget that takes ciphertexts as inputs and produces ciphertexts as outputs. In order to make the gadget leakage-resilient, they use the encryption scheme of [BHHO08, NS09] (see Section 2.2), slightly modified and augmented with (input-indepenent) leak-free hardware. The encryption keys are updated for each iteration. Under the assumption that each gadget leaks independently, the compiled circuit can tolerate a fixed amount of polynomial-time leakage per gadget. Thus, in contrast to circuit compilers described in Section 4.3.3 and the result of [JV10], the amount of leakage they can tolerate grows with the circuit size.

Dziembowski and Faust [DF12] and, independently, Goldwasser and Rothblum [GR12, GR15] eliminate the need for computational assumptions in [GR10], achieving security against arbitrarily complex (rather than only polynomial-time) leakage functions. Miles and Viola [MV13] provide another construction, by observing that their compiler against computationally-bounded leakage also provides security in the OCL model; however, it tolerates less leakage that the constructions of [DF12, GR12, GR15].

The compiler of [DF12], like prior work, assumes some leak-free hardware. It uses so-called "inner-product masking": each wire is represented by two vectors whose inner-product is equal to the wire value, as in the leakage-resilient storage of [DDV10] (see Section 3). Because the inner product function is a two-source extractor (which means the output is close to uniformly random as long as the two sources are independent and each has sufficient entropy), as long as the two vectors leak independently and not too much, the wire value is well-hidden. Gadgets that operate on the vectors are constructed with the help of (input-independent) leak-free hardware. This construction can be viewed in the circuit model, having $2n$ independently leaking components (where n is the number of wires in the original circuit). It can also be viewed as a two-party protocol, where each party keeps one of the two vectors for each wire, and the parties communicate for each gate. The latter view allows for much less leakage. The efficiency of this compiler has been improved by Andrychowicz et al. [ADD$^+$15].

The compiler of [GR12, GR15] eliminates not only computational assumptions, but also leak-free hardware, by replacing the computational encryption scheme of [GR10] with an information-theoretic one and replacing the leak-free components with leakage-resilient computation. Thus, the only remaining assumption is on the leakage function: that each component leaks independently, and the amount of leakage per component is bounded (it is also assumed, like in previous work, that the compilation itself, which is randomized and places the secret state into the circuit, doesn't leak; this assumption is shown necessary in [DDN15]). The number of components is the same as the number of gates in the original circuit.

Bitansky, Dachman-Soled, and Lin [BDL14] obtain a protocol with a constant number of independently-leaking components without computational assumptions or leak-free hardware. The number of parties is estimated to be about about 20 in [DLZ15]). Each component is invoked

a linear (in the circuit size) number of times. The main idea of the construction is to use the 2-component version of the compiler of [DF12], and replace the leak-free hardware by the leakage-resilient computation of [GR12, GR15].

Dachman-Soled, Liu, and Zhou [DLZ15] reduce the number of components even further—down to the optimal two—without relying on leak-free hardware, but at the cost of very strong computational assumptions. The technical idea behind their construction is to start with a two-component compiler that requires leak-free hardware (such as [JV10] or [DF12]) and then replace the leak-free hardware with a leakage-resilient two-party protocol. This protocol is what requires the computational assumption.

For the case of one-time security of stateless circuits (see Section 4.3.1), Goyal et al. [GIM+16] build compilers in the 2-component bounded-communication leakage model (which is a generalization of the OCL model; see Section 4.1). In this stateless setting, they are able to reduce the assumptions of [DLZ15] and increase efficiency compared to prior constructions, without resorting to leak-free hardware. The technical idea of the construction is a result that shows that protection against leakage functions that simply compute parities of wire values is essentially sufficient. It is not known how to extend this construction to protect against continual leakage in the stateful case.

Genkin, Ishai, and Weiss [GIW17] observe that leakage-resilient stateless circuits make sense as implementation to trusted third parties, in which multiple participants provide inputs and rely on the trusted third party to compute an output. While the party is trusted to compute the output correctly and not leak information deliberately, it may be under a side-channel attack by an adversary. This setting presents its own challenges not present in the usual stateful compilers (in particular, what happens if some participants provide invalidly encoded inputs). Building on the work of [GIM+16] for stateless compilers and the work of [IWY16], they show how these challenges can be overcome.

Most of the papers discussed above focus on the theory feasibility results and do not analyze the practical feasibility of their compilers. Further work is needed to make any of them practical.

On the more applied side, Andrychowicz, Masny, and Persichetti [AMP15] propose a two-component OCL compiler using inner-product masking over large finite fields (and some leak-free components), and apply it to the "Lapin" secret-key authentication protocol [HKL+12], producing a working implementation. They evaluate both the concrete leakage-resilient and concrete performance of their proposal, reporting a 30-fold slowdown over the standard version of Lapin for reasonable security parameters.

4.3.5 Compilers for Noisy and Noisy OCL Leakage

As already mentioned above, one of the compilers of Faust et al. [FRR+14] works in a noisy leakage model that is reminiscent of the noisy leakage model of Chari et al. [CJRR99]. Specifically, the assumption is that every wire's value is provided to the adversary, but each one is flipped independently with probability p. The compiler uses the same boolean masking as [CJRR99, ISW03], but builds AND gadgets differently. Unfortunately, the compiler is far from practical, requiring at least a million-fold increase in the circuit size even for small security parameters (in particular, to achieve security $2^{-\lambda}$ when the error probability for the leakage of each wire is $p \leq \frac{1}{2}$, the circuit size has to increase by a factor of more than $\max(10^5 \cdot \lambda^2, p^{-12}\lambda/100)$).

Subsequent work considered more general noisy leakage functions, many of them in a variant of the OCL model. In the version of the OCL model used in most works mentioned in 4.3.4,

the leakage can be an arbitrary polynomial-time function of the relevant portion of the state, but of limited output length. An objection to this model of leakage (raised in multiple forums; e.g., [SPY+10, SPY13]) is that it is both too strong and too weak. It is too strong because in reality, the physical side channels do not compute arbitrary polynomial-time functions, and ensuring protection against arbitrary polynomial-time leakage forces the designs to have unnecessary complexity. It is too weak because real side-channel attacks receive many bits of leakage—typically many more than the amount of secret state.

Addressing these objections, Prouff and Rivain [PR13] show a circuit compiler in the OCL model (with a linear number of independently leaking components), where the leakage from each component of the circuit reveals limited information (in the statistical sense of biasing the distribution) about the value being leaked. (Note that the model of power analysis attacks by Chari et al. [CJRR99], discussed in Section 1.1, has this property.) Their compiler uses additive secret sharing (also known as masking) for the wires, and gadgets similar to [ISW03, FRR+14] for multiplication; it is specialized to block ciphers that consist of s-box and linear operations, following the ideas of [CGP+12]. It uses some leak-free components. The security model of [PR13] is weaker than the model of [ISW03]; in particular, it does not provide the adversary with the input-output behavior of the circuit, but only with leakage under random inputs.

Duc, Dziembowski, and Faust [DDF14] show a much stronger compiler for the class of leakage functions considered in [PR13]. They demonstrate that the original compiler of [ISW03], without any leak-free components, and for arbitrary circuits, is also secure against noisy OCL leakage. Moreover, security holds for the strong definition of [ISW03], which allows the adversary to probe the input-output behavior of the circuit while obtaining side-channel leakage. They achieve this result by showing equivalence between noisy and wire-probing leakage; this equivalence has been used in subsequent works, as well. Duc, Faust, and Standaert [DFS15a, DDF19] further improve on the result by measuring the "noisiness" of statistical distance via a mutual information metric rather than statistical distance; it is argued that this metric is easier to estimate in practice. The quantitative bounds (relating the amount of noise to the security of the overall scheme) are further improved by Dziembowski, Faust, and Skórski [DFS15b, DFS16]. Andrychowicz, Dziembowski, and Faust [ADF16] and Goudrazi, Joux, and Rivain [GJR18] (using techniques from [ADD+15]) show how to improve the leakage rate and the efficiency of the transformed circuit.

4.4 Heuristic Security Evaluation of Leakage-Resilient Constructions

Much effort has also been devoted to understanding the security properties of masking in general and particularly in the context of block ciphers. As already mentioned, the [ISW03] compiler is secure against wire probing attacks that do not touch more than t wires. However, most realistic attacks with current technology do not obtain information about only a few wires; instead, they get noisy information about many wires. This kind of leakage is discussed in Section 4.3.5, in the simulatability framework of [ISW03]. However, simulatability is a very strong requirement, and is often unachievable within realistic efficiency constraints. Thus, researchers have used approaches based on evaluating the best known classes of attack strategies, in order to understand the security of designs for which the simulation proofs either do not exist or do not give meaningful security bounds. These approaches provide weaker security guarantees, because they do not consider all possible adversaries, but rather the best classes of adversaries known today. Nevertheless, they are often very useful for understanding the cost/benefit tradeoffs of various designs, and are used extensively in applied literature.

A prominent heuristic evaluation framework was put forward by [SMY09]. A large number of cryptographic designs and side-channel countermeasures have been evaluated in this framework (many of these are referenced in [DFS15a, Section 1]). A comparison between this approach and the more theoretical approach of [DP08, Pie09] (see Sections 4.2.2 and 4.2.3) is provided in [SPY+10]. An alternative evaluation framework was proposed in [WO11a, WO11b]. Some works combine provable and heuristic evaluations—see, e.g., [DFS15a]. The heuristic evaluation frameworks continue to evolve and mature; see [GS18] and references therein.

Barthe et al. [BDF+17] observe that side-channel attackers are often faced with the task of estimating statistical moments of random variables they receive as leakage functions. They therefore propose that the goal of a secure design it to make sure these moments, up to some order, are independent of the secret state of the circuit (the reasoning is that higher-order moments, which may be dependent, are very difficult to estimate). They relate their security goal to the wire-probing leakage of [ISW03] and argue that their model is particularly suitable for highly parallel (i.e., hardware rather than software) implementations.

Because of this survey's focus on approaches with a provable security foundation, we do not discuss heuristic evaluation frameworks in more detail, despite their strong impact on applied work.

5 Acknowledgements

We are deeply grateful to Shafi and Silvio for the intellectual gems they gave us, for their outstanding professional and personal mentorship, and for nurturing a thriving community of researchers that we are honored to call home.

We are also thankful to the authors of all the papers we surveyed. Many of them were patient enough to explain their results to us and to help put them in context.

We thank Oded Goldreich for a careful reading of our draft and excellent suggestions.

The work of Leonid Reyzin is supported, in part, by the US NSF grant 1422965.

References

[AARR03] Dakshi Agrawal, Bruce Archambeault, Josyula R. Rao, and Pankaj Rohatgi. The EM side-channel(s). In Burton S. Kaliski Jr., Çetin Kaya Koç, and Christof Paar, editors, *Cryptographic Hardware and Embedded Systems – CHES 2002*, volume 2523 of *Lecture Notes in Computer Science*, pages 29–45. Springer, Heidelberg, August 2003.

[AB00] Michel Abdalla and Mihir Bellare. Increasing the lifetime of a key: a comparative analysis of the security of re-keying techniques. In Tatsuaki Okamoto, editor, *Advances in Cryptology – ASIACRYPT 2000*, volume 1976 of *Lecture Notes in Computer Science*, pages 546–559. Springer, Heidelberg, December 2000.

[ABF13] Michel Abdalla, Sonia Belaïd, and Pierre-Alain Fouque. Leakage-resilient symmetric encryption via re-keying. In Guido Bertoni and Jean-Sébastien Coron, editors, *Cryptographic Hardware and Embedded Systems – CHES 2013*, volume 8086 of *Lecture Notes in Computer Science*, pages 471–488. Springer, Heidelberg, August 2013.

[ABP+15] Michel Abdalla, Sonia Belaïd, David Pointcheval, Sylvain Ruhault, and Damien Vergnaud. Robust pseudo-random number generators with input secure against side-

channel attacks. In Tal Malkin, Vladimir Kolesnikov, Allison Bishop Lewko, and Michalis Polychronakis, editors, *ACNS 15: 13th International Conference on Applied Cryptography and Network Security*, volume 9092 of *Lecture Notes in Computer Science*, pages 635–654. Springer, Heidelberg, June 2015.

[ADD+15] Marcin Andrychowicz, Ivan Damgård, Stefan Dziembowski, Sebastian Faust, and Antigoni Polychroniadou. Efficient leakage resilient circuit compilers. In Kaisa Nyberg, editor, *Topics in Cryptology – CT-RSA 2015*, volume 9048 of *Lecture Notes in Computer Science*, pages 311–329. Springer, Heidelberg, April 2015.

[ADF16] Marcin Andrychowicz, Stefan Dziembowski, and Sebastian Faust. Circuit compilers with $O(1/\log(n))$ leakage rate. In Marc Fischlin and Jean-Sébastien Coron, editors, *Advances in Cryptology – EUROCRYPT 2016, Part II*, volume 9666 of *Lecture Notes in Computer Science*, pages 586–615. Springer, Heidelberg, May 2016.

[ADN+10] Joël Alwen, Yevgeniy Dodis, Moni Naor, Gil Segev, Shabsi Walfish, and Daniel Wichs. Public-key encryption in the bounded-retrieval model. In Henri Gilbert, editor, *Advances in Cryptology – EUROCRYPT 2010*, volume 6110 of *Lecture Notes in Computer Science*, pages 113–134. Springer, Heidelberg, May / June 2010.

[ADW09] Joël Alwen, Yevgeniy Dodis, and Daniel Wichs. Leakage-resilient public-key cryptography in the bounded-retrieval model. In Shai Halevi, editor, *Advances in Cryptology – CRYPTO 2009*, volume 5677 of *Lecture Notes in Computer Science*, pages 36–54. Springer, Heidelberg, August 2009.

[ADW10] Joël Alwen, Yevgeniy Dodis, and Daniel Wichs. Survey: Leakage resilience and the bounded retrieval model. In Kaoru Kurosawa, editor, *ICITS 09: 4th International Conference on Information Theoretic Security*, volume 5973 of *Lecture Notes in Computer Science*, pages 1–18. Springer, Heidelberg, December 2010.

[AGH12] Adi Akavia, Shafi Goldwasser, and Carmit Hazay. Distributed public key schemes secure against continual leakage. In Darek Kowalski and Alessandro Panconesi, editors, *31st ACM Symposium Annual on Principles of Distributed Computing*, pages 155–164. Association for Computing Machinery, July 2012.

[AGP14] Prabhanjan Ananth, Vipul Goyal, and Omkant Pandey. Interactive proofs under continual memory leakage. In Juan A. Garay and Rosario Gennaro, editors, *Advances in Cryptology – CRYPTO 2014, Part II*, volume 8617 of *Lecture Notes in Computer Science*, pages 164–182. Springer, Heidelberg, August 2014.

[AGV09] Adi Akavia, Shafi Goldwasser, and Vinod Vaikuntanathan. Simultaneous hardcore bits and cryptography against memory attacks. In Omer Reingold, editor, *TCC 2009: 6th Theory of Cryptography Conference*, volume 5444 of *Lecture Notes in Computer Science*, pages 474–495. Springer, Heidelberg, March 2009.

[AIS18] Prabhanjan Ananth, Yuval Ishai, and Amit Sahai. Private circuits: A modular approach. In Hovav Shacham and Alexandra Boldyreva, editors, *Advances in Cryptology – CRYPTO 2018, Part III*, volume 10993 of *Lecture Notes in Computer Science*, pages 427–455. Springer, Heidelberg, August 2018.

[Ajt11] Miklós Ajtai. Secure computation with information leaking to an adversary. In Lance Fortnow and Salil P. Vadhan, editors, *43rd Annual ACM Symposium on Theory of Computing*, pages 715–724. ACM Press, June 2011.

[AMP15] Marcin Andrychowicz, Daniel Masny, and Edoardo Persichetti. Leakage-resilient cryptography over large finite fields: Theory and practice. In Tal Malkin, Vladimir Kolesnikov, Allison Bishop Lewko, and Michalis Polychronakis, editors, *ACNS 15: 13th International Conference on Applied Cryptography and Network Security*, volume 9092 of *Lecture Notes in Computer Science*, pages 655–674. Springer, Heidelberg, June 2015.

[BB04] Dan Boneh and Xavier Boyen. Secure identity based encryption without random oracles. In Matthew Franklin, editor, *Advances in Cryptology – CRYPTO 2004*, volume 3152 of *Lecture Notes in Computer Science*, pages 443–459. Springer, Heidelberg, August 2004.

[BB11] Dan Boneh and Xavier Boyen. Efficient selective identity-based encryption without random oracles. *Journal of Cryptology*, 24(4):659–693, October 2011.

[BBD+15] Gilles Barthe, Sonia Belaïd, François Dupressoir, Pierre-Alain Fouque, Benjamin Grégoire, and Pierre-Yves Strub. Verified proofs of higher-order masking. In Elisabeth Oswald and Marc Fischlin, editors, *Advances in Cryptology – EUROCRYPT 2015, Part I*, volume 9056 of *Lecture Notes in Computer Science*, pages 457–485. Springer, Heidelberg, April 2015.

[BBD+16] Gilles Barthe, Sonia Belaïd, François Dupressoir, Pierre-Alain Fouque, Benjamin Grégoire, Pierre-Yves Strub, and Rébecca Zucchini. Strong non-interference and type-directed higher-order masking. In Edgar R. Weippl, Stefan Katzenbeisser, Christopher Kruegel, Andrew C. Myers, and Shai Halevi, editors, *ACM CCS 16: 23rd Conference on Computer and Communications Security*, pages 116–129. ACM Press, October 2016.

[BBE+18] Gilles Barthe, Sonia Belaïd, Thomas Espitau, Pierre-Alain Fouque, Benjamin Grégoire, Mélissa Rossi, and Mehdi Tibouchi. Masking the GLP lattice-based signature scheme at any order. In Nielsen and Rijmen [NR18], pages 354–384.

[BBP+16] Sonia Belaïd, Fabrice Benhamouda, Alain Passelègue, Emmanuel Prouff, Adrian Thillard, and Damien Vergnaud. Randomness complexity of private circuits for multiplication. In Marc Fischlin and Jean-Sébastien Coron, editors, *Advances in Cryptology – EUROCRYPT 2016, Part II*, volume 9666 of *Lecture Notes in Computer Science*, pages 616–648. Springer, Heidelberg, May 2016.

[BBP+17] Sonia Belaïd, Fabrice Benhamouda, Alain Passelègue, Emmanuel Prouff, Adrian Thillard, and Damien Vergnaud. Private multiplication over finite fields. In Jonathan Katz and Hovav Shacham, editors, *Advances in Cryptology – CRYPTO 2017, Part III*, volume 10403 of *Lecture Notes in Computer Science*, pages 397–426. Springer, Heidelberg, August 2017.

[BCG+11] Nir Bitansky, Ran Canetti, Shafi Goldwasser, Shai Halevi, Yael Tauman Kalai, and Guy N. Rothblum. Program obfuscation with leaky hardware. In Dong Hoon Lee and Xiaoyun Wang, editors, *Advances in Cryptology – ASIACRYPT 2011*, volume 7073 of *Lecture Notes in Computer Science*, pages 722–739. Springer, Heidelberg, December 2011.

[BCH12] Nir Bitansky, Ran Canetti, and Shai Halevi. Leakage-tolerant interactive protocols. In Ronald Cramer, editor, *TCC 2012: 9th Theory of Cryptography Conference*, volume 7194 of *Lecture Notes in Computer Science*, pages 266–284. Springer, Heidelberg, March 2012.

[BCHK07] Dan Boneh, Ran Canetti, Shai Halevi, and Jonathan Katz. Chosen-ciphertext security from identity-based encryption. *SIAM J. Comput.*, 36(5):1301–1328, 2007.

[BCZ18] Luk Bettale, Jean-Sébastien Coron, and Rina Zeitoun. Improved high-order conversion from boolean to arithmetic masking. *IACR Trans. Cryptogr. Hardw. Embed. Syst.*, 2018(2):22–45, 2018.

[BDF+17] Gilles Barthe, François Dupressoir, Sebastian Faust, Benjamin Grégoire, François-Xavier Standaert, and Pierre-Yves Strub. Parallel implementations of masking schemes and the bounded moment leakage model. In Jean-Sébastien Coron and Jesper Buus Nielsen, editors, *Advances in Cryptology – EUROCRYPT 2017, Part I*, volume 10210 of *Lecture Notes in Computer Science*, pages 535–566. Springer, Heidelberg, April / May 2017.

[BDIR18] Fabrice Benhamouda, Akshay Degwekar, Yuval Ishai, and Tal Rabin. On the local leakage resilience of linear secret sharing schemes. In Hovav Shacham and Alexandra Boldyreva, editors, *Advances in Cryptology – CRYPTO 2018, Part I*, volume 10991 of *Lecture Notes in Computer Science*, pages 531–561. Springer, Heidelberg, August 2018.

[BDL14] Nir Bitansky, Dana Dachman-Soled, and Huijia Lin. Leakage-tolerant computation with input-independent preprocessing. In Juan A. Garay and Rosario Gennaro, editors, *Advances in Cryptology – CRYPTO 2014, Part II*, volume 8617 of *Lecture Notes in Computer Science*, pages 146–163. Springer, Heidelberg, August 2014.

[BFG+17] Josep Balasch, Sebastian Faust, Benedikt Gierlichs, Clara Paglialonga, and François-Xavier Standaert. Consolidating inner product masking. In Tsuyoshi Takagi and Thomas Peyrin, editors, *Advances in Cryptology – ASIACRYPT 2017, Part I*, volume 10624 of *Lecture Notes in Computer Science*, pages 724–754. Springer, Heidelberg, December 2017.

[BFM88] Manuel Blum, Paul Feldman, and Silvio Micali. Non-interactive zero-knowledge and its applications (extended abstract). In *20th Annual ACM Symposium on Theory of Computing*, pages 103–112. ACM Press, May 1988.

[BG10] Zvika Brakerski and Shafi Goldwasser. Circular and leakage resilient public-key encryption under subgroup indistinguishability - (or: Quadratic residuosity strikes back).

In Tal Rabin, editor, *Advances in Cryptology – CRYPTO 2010*, volume 6223 of *Lecture Notes in Computer Science*, pages 1–20. Springer, Heidelberg, August 2010.

[BGI+01] Boaz Barak, Oded Goldreich, Russell Impagliazzo, Steven Rudich, Amit Sahai, Salil P. Vadhan, and Ke Yang. On the (im)possibility of obfuscating programs. In Joe Kilian, editor, *Advances in Cryptology – CRYPTO 2001*, volume 2139 of *Lecture Notes in Computer Science*, pages 1–18. Springer, Heidelberg, August 2001.

[BGI+18] Roderick Bloem, Hannes Groß, Rinat Iusupov, Bettina Könighofer, Stefan Mangard, and Johannes Winter. Formal verification of masked hardware implementations in the presence of glitches. In Jesper Buus Nielsen and Vincent Rijmen, editors, *Advances in Cryptology – EUROCRYPT 2018, Part II*, volume 10821 of *Lecture Notes in Computer Science*, pages 321–353. Springer, Heidelberg, April / May 2018.

[BGJ+13] Elette Boyle, Sanjam Garg, Abhishek Jain, Yael Tauman Kalai, and Amit Sahai. Secure computation against adaptive auxiliary information. In Ran Canetti and Juan A. Garay, editors, *Advances in Cryptology – CRYPTO 2013, Part I*, volume 8042 of *Lecture Notes in Computer Science*, pages 316–334. Springer, Heidelberg, August 2013.

[BGJK12] Elette Boyle, Shafi Goldwasser, Abhishek Jain, and Yael Tauman Kalai. Multiparty computation secure against continual memory leakage. In Howard J. Karloff and Toniann Pitassi, editors, *44th Annual ACM Symposium on Theory of Computing*, pages 1235–1254. ACM Press, May 2012.

[BGK11] Elette Boyle, Shafi Goldwasser, and Yael Tauman Kalai. Leakage-resilient coin tossing. In David Peleg, editor, *Distributed Computing - 25th International Symposium, DISC 2011, Rome, Italy, September 20-22, 2011. Proceedings*, volume 6950 of *Lecture Notes in Computer Science*, pages 181–196. Springer, 2011.

[BGR18] Sonia Belaïd, Dahmun Goudarzi, and Matthieu Rivain. Tight private circuits: Achieving probing security with the least refreshing. Cryptology ePrint Archive, Report 2018/439, 2018. `https://eprint.iacr.org/2018/439`.

[BGW88] Michael Ben-Or, Shafi Goldwasser, and Avi Wigderson. Completeness theorems for non-cryptographic fault-tolerant distributed computation (extended abstract). In *20th Annual ACM Symposium on Theory of Computing*, pages 1–10. ACM Press, May 1988.

[BHHO08] Dan Boneh, Shai Halevi, Michael Hamburg, and Rafail Ostrovsky. Circular-secure encryption from decision Diffie-Hellman. In David Wagner, editor, *Advances in Cryptology – CRYPTO 2008*, volume 5157 of *Lecture Notes in Computer Science*, pages 108–125. Springer, Heidelberg, August 2008.

[BHK11] Mark Braverman, Avinatan Hassidim, and Yael Tauman Kalai. Leaky pseudo-entropy functions. In Bernard Chazelle, editor, *ICS 2011: 2nd Innovations in Computer Science*, pages 353–366. Tsinghua University Press, January 2011.

[BIVW16] Andrej Bogdanov, Yuval Ishai, Emanuele Viola, and Christopher Williamson. Bounded indistinguishability and the complexity of recovering secrets. In Matthew Robshaw and Jonathan Katz, editors, *Advances in Cryptology – CRYPTO 2016,*

Part III, volume 9816 of *Lecture Notes in Computer Science*, pages 593–618. Springer, Heidelberg, August 2016.

[BKKV10] Zvika Brakerski, Yael Tauman Kalai, Jonathan Katz, and Vinod Vaikuntanathan. Overcoming the hole in the bucket: Public-key cryptography resilient to continual memory leakage. In *51st Annual Symposium on Foundations of Computer Science*, pages 501–510. IEEE Computer Society Press, October 2010.

[BLS04] Dan Boneh, Ben Lynn, and Hovav Shacham. Short signatures from the Weil pairing. *Journal of Cryptology*, 17(4):297–319, September 2004.

[BM82] Manuel Blum and Silvio Micali. How to generate cryptographically strong sequences of pseudo random bits. In *23rd Annual Symposium on Foundations of Computer Science*, pages 112–117. IEEE Computer Society Press, November 1982.

[BM84] Manuel Blum and Silvio Micali. How to generate cryptographically strong sequences of pseudorandom bits. *SIAM Journal on Computing*, 13(4):850–864, 1984.

[BM99] Mihir Bellare and Sara K. Miner. A forward-secure digital signature scheme. In Michael J. Wiener, editor, *Advances in Cryptology – CRYPTO'99*, volume 1666 of *Lecture Notes in Computer Science*, pages 431–448. Springer, Heidelberg, August 1999.

[BMOS17] Guy Barwell, Daniel P. Martin, Elisabeth Oswald, and Martijn Stam. Authenticated encryption in the face of protocol and side channel leakage. In Tsuyoshi Takagi and Thomas Peyrin, editors, *Advances in Cryptology – ASIACRYPT 2017, Part I*, volume 10624 of *Lecture Notes in Computer Science*, pages 693–723. Springer, Heidelberg, December 2017.

[BMW+18] Jo Van Bulck, Marina Minkin, Ofir Weisse, Daniel Genkin, Baris Kasikci, Frank Piessens, Mark Silberstein, Thomas F. Wenisch, Yuval Yarom, and Raoul Strackx. Foreshadow: Extracting the keys to the intel SGX kingdom with transient out-of-order execution. In Enck and Felt [EF18], pages 991–1008.

[Boy99] Victor Boyko. On the security properties of OAEP as an all-or-nothing transform. In Michael J. Wiener, editor, *Advances in Cryptology – CRYPTO'99*, volume 1666 of *Lecture Notes in Computer Science*, pages 503–518. Springer, Heidelberg, August 1999.

[BS11] Zvika Brakerski and Gil Segev. Better security for deterministic public-key encryption: The auxiliary-input setting. In Phillip Rogaway, editor, *Advances in Cryptology – CRYPTO 2011*, volume 6841 of *Lecture Notes in Computer Science*, pages 543–560. Springer, Heidelberg, August 2011.

[BSW11] Elette Boyle, Gil Segev, and Daniel Wichs. Fully leakage-resilient signatures. In Kenneth G. Paterson, editor, *Advances in Cryptology – EUROCRYPT 2011*, volume 6632 of *Lecture Notes in Computer Science*, pages 89–108. Springer, Heidelberg, May 2011.

[Can97] Ran Canetti. Towards realizing random oracles: Hash functions that hide all partial information. In Burton S. Kaliski Jr., editor, *Advances in Cryptology – CRYPTO'97*, volume 1294 of *Lecture Notes in Computer Science*, pages 455–469. Springer, Heidelberg, August 1997.

[CDH+00] Ran Canetti, Yevgeniy Dodis, Shai Halevi, Eyal Kushilevitz, and Amit Sahai. Exposure-resilient functions and all-or-nothing transforms. In Bart Preneel, editor, *Advances in Cryptology – EUROCRYPT 2000*, volume 1807 of *Lecture Notes in Computer Science*, pages 453–469. Springer, Heidelberg, May 2000.

[CFGN96] Ran Canetti, Uriel Feige, Oded Goldreich, and Moni Naor. Adaptively secure multi-party computation. In *28th Annual ACM Symposium on Theory of Computing*, pages 639–648. ACM Press, May 1996.

[CGP+12] Claude Carlet, Louis Goubin, Emmanuel Prouff, Michaël Quisquater, and Matthieu Rivain. Higher-order masking schemes for S-boxes. In Anne Canteaut, editor, *Fast Software Encryption – FSE 2012*, volume 7549 of *Lecture Notes in Computer Science*, pages 366–384. Springer, Heidelberg, March 2012.

[CJRR99] Suresh Chari, Charanjit S. Jutla, Josyula R. Rao, and Pankaj Rohatgi. Towards sound approaches to counteract power-analysis attacks. In Michael J. Wiener, editor, *Advances in Cryptology – CRYPTO'99*, volume 1666 of *Lecture Notes in Computer Science*, pages 398–412. Springer, Heidelberg, August 1999.

[Cor18] Jean-Sébastien Coron. Formal verification of side-channel countermeasures via elementary circuit transformations. In Bart Preneel and Frederik Vercauteren, editors, *ACNS 18: 16th International Conference on Applied Cryptography and Network Security*, volume 10892 of *Lecture Notes in Computer Science*, pages 65–82. Springer, Heidelberg, July 2018.

[CRZ18] Jean-Sébastien Coron, Franck Rondepierre, and Rina Zeitoun. High order masking of look-up tables with common shares. *IACR Trans. Cryptogr. Hardw. Embed. Syst.*, 2018(1):40–72, 2018.

[CS02] Ronald Cramer and Victor Shoup. Universal hash proofs and a paradigm for adaptive chosen ciphertext secure public-key encryption. In Lars R. Knudsen, editor, *Advances in Cryptology – EUROCRYPT 2002*, volume 2332 of *Lecture Notes in Computer Science*, pages 45–64. Springer, Heidelberg, April / May 2002.

[DDF14] Alexandre Duc, Stefan Dziembowski, and Sebastian Faust. Unifying leakage models: From probing attacks to noisy leakage. In Phong Q. Nguyen and Elisabeth Oswald, editors, *Advances in Cryptology – EUROCRYPT 2014*, volume 8441 of *Lecture Notes in Computer Science*, pages 423–440. Springer, Heidelberg, May 2014.

[DDF19] Alexandre Duc, Stefan Dziembowski, and Sebastian Faust. Unifying leakage models: From probing attacks to noisy leakage. *Journal of Cryptology*, 32(1):151–177, January 2019.

[DDN15] Ivan Damgård, Frédéric Dupuis, and Jesper Buus Nielsen. On the orthogonal vector problem and the feasibility of unconditionally secure leakage-resilient computation. In Anja Lehmann and Stefan Wolf, editors, *ICITS 15: 8th International Conference on Information Theoretic Security*, volume 9063 of *Lecture Notes in Computer Science*, pages 87–104. Springer, Heidelberg, May 2015.

[DDV10] Francesco Davì, Stefan Dziembowski, and Daniele Venturi. Leakage-resilient storage. In Juan A. Garay and Roberto De Prisco, editors, *SCN 10: 7th International Conference on Security in Communication Networks*, volume 6280 of *Lecture Notes in Computer Science*, pages 121–137. Springer, Heidelberg, September 2010.

[DF90] Yvo Desmedt and Yair Frankel. Threshold cryptosystems. In Gilles Brassard, editor, *Advances in Cryptology – CRYPTO'89*, volume 435 of *Lecture Notes in Computer Science*, pages 307–315. Springer, Heidelberg, August 1990.

[DF11] Stefan Dziembowski and Sebastian Faust. Leakage-resilient cryptography from the inner-product extractor. In Dong Hoon Lee and Xiaoyun Wang, editors, *Advances in Cryptology – ASIACRYPT 2011*, volume 7073 of *Lecture Notes in Computer Science*, pages 702–721. Springer, Heidelberg, December 2011.

[DF12] Stefan Dziembowski and Sebastian Faust. Leakage-resilient circuits without computational assumptions. In Ronald Cramer, editor, *TCC 2012: 9th Theory of Cryptography Conference*, volume 7194 of *Lecture Notes in Computer Science*, pages 230–247. Springer, Heidelberg, March 2012.

[DFH+16] Stefan Dziembowski, Sebastian Faust, Gottfried Herold, Anthony Journault, Daniel Masny, and François-Xavier Standaert. Towards sound fresh re-keying with hard (physical) learning problems. In Matthew Robshaw and Jonathan Katz, editors, *Advances in Cryptology – CRYPTO 2016, Part II*, volume 9815 of *Lecture Notes in Computer Science*, pages 272–301. Springer, Heidelberg, August 2016.

[DFS15a] Alexandre Duc, Sebastian Faust, and François-Xavier Standaert. Making masking security proofs concrete - or how to evaluate the security of any leaking device. In Elisabeth Oswald and Marc Fischlin, editors, *Advances in Cryptology – EUROCRYPT 2015, Part I*, volume 9056 of *Lecture Notes in Computer Science*, pages 401–429. Springer, Heidelberg, April 2015.

[DFS15b] Stefan Dziembowski, Sebastian Faust, and Maciej Skorski. Noisy leakage revisited. In Elisabeth Oswald and Marc Fischlin, editors, *Advances in Cryptology – EUROCRYPT 2015, Part II*, volume 9057 of *Lecture Notes in Computer Science*, pages 159–188. Springer, Heidelberg, April 2015.

[DFS16] Stefan Dziembowski, Sebastian Faust, and Maciej Skórski. Optimal amplification of noisy leakages. In Eyal Kushilevitz and Tal Malkin, editors, *TCC 2016-A: 13th Theory of Cryptography Conference, Part II*, volume 9563 of *Lecture Notes in Computer Science*, pages 291–318. Springer, Heidelberg, January 2016.

[DGK+10] Yevgeniy Dodis, Shafi Goldwasser, Yael Tauman Kalai, Chris Peikert, and Vinod Vaikuntanathan. Public-key encryption schemes with auxiliary inputs. In Daniele

Micciancio, editor, *TCC 2010: 7th Theory of Cryptography Conference*, volume 5978 of *Lecture Notes in Computer Science*, pages 361–381. Springer, Heidelberg, February 2010.

[DGL+16] Dana Dachman-Soled, S. Dov Gordon, Feng-Hao Liu, Adam O'Neill, and Hong-Sheng Zhou. Leakage-resilient public-key encryption from obfuscation. In Chen-Mou Cheng, Kai-Min Chung, Giuseppe Persiano, and Bo-Yin Yang, editors, *PKC 2016: 19th International Conference on Theory and Practice of Public Key Cryptography, Part II*, volume 9615 of *Lecture Notes in Computer Science*, pages 101–128. Springer, Heidelberg, March 2016.

[DHLW10a] Yevgeniy Dodis, Kristiyan Haralambiev, Adriana López-Alt, and Daniel Wichs. Cryptography against continuous memory attacks. In *51st Annual Symposium on Foundations of Computer Science*, pages 511–520. IEEE Computer Society Press, October 2010.

[DHLW10b] Yevgeniy Dodis, Kristiyan Haralambiev, Adriana López-Alt, and Daniel Wichs. Efficient public-key cryptography in the presence of key leakage. In Masayuki Abe, editor, *Advances in Cryptology – ASIACRYPT 2010*, volume 6477 of *Lecture Notes in Computer Science*, pages 613–631. Springer, Heidelberg, December 2010.

[DKL09] Yevgeniy Dodis, Yael Tauman Kalai, and Shachar Lovett. On cryptography with auxiliary input. In Michael Mitzenmacher, editor, *41st Annual ACM Symposium on Theory of Computing*, pages 621–630. ACM Press, May / June 2009.

[DKW11] Stefan Dziembowski, Tomasz Kazana, and Daniel Wichs. Key-evolution schemes resilient to space-bounded leakage. In Phillip Rogaway, editor, *Advances in Cryptology – CRYPTO 2011*, volume 6841 of *Lecture Notes in Computer Science*, pages 335–353. Springer, Heidelberg, August 2011.

[DLSZ15] Dana Dachman-Soled, Feng-Hao Liu, Elaine Shi, and Hong-Sheng Zhou. Locally decodable and updatable non-malleable codes and their applications. In Yevgeniy Dodis and Jesper Buus Nielsen, editors, *TCC 2015: 12th Theory of Cryptography Conference, Part I*, volume 9014 of *Lecture Notes in Computer Science*, pages 427–450. Springer, Heidelberg, March 2015.

[DLW06] Giovanni Di Crescenzo, Richard J. Lipton, and Shabsi Walfish. Perfectly secure password protocols in the bounded retrieval model. In Shai Halevi and Tal Rabin, editors, *TCC 2006: 3rd Theory of Cryptography Conference*, volume 3876 of *Lecture Notes in Computer Science*, pages 225–244. Springer, Heidelberg, March 2006.

[DLWW11] Yevgeniy Dodis, Allison B. Lewko, Brent Waters, and Daniel Wichs. Storing secrets on continually leaky devices. In Rafail Ostrovsky, editor, *52nd Annual Symposium on Foundations of Computer Science*, pages 688–697. IEEE Computer Society Press, October 2011.

[DLZ15] Dana Dachman-Soled, Feng-Hao Liu, and Hong-Sheng Zhou. Leakage-resilient circuits revisited - optimal number of computing components without leak-free hardware. In Elisabeth Oswald and Marc Fischlin, editors, *Advances in Cryptology –*

EUROCRYPT 2015, Part II, volume 9057 of *Lecture Notes in Computer Science*, pages 131–158. Springer, Heidelberg, April 2015.

[Dod00] Yevgeniy Dodis. *Exposure-Resilient Cryptography*. PhD thesis, Massachusetts Institute of Technology, 2000.

[DP07] Stefan Dziembowski and Krzysztof Pietrzak. Intrusion-resilient secret sharing. In *48th Annual Symposium on Foundations of Computer Science*, pages 227–237. IEEE Computer Society Press, October 2007.

[DP08] Stefan Dziembowski and Krzysztof Pietrzak. Leakage-resilient cryptography. In *49th Annual Symposium on Foundations of Computer Science*, pages 293–302. IEEE Computer Society Press, October 2008.

[DP10] Yevgeniy Dodis and Krzysztof Pietrzak. Leakage-resilient pseudorandom functions and side-channel attacks on Feistel networks. In Tal Rabin, editor, *Advances in Cryptology – CRYPTO 2010*, volume 6223 of *Lecture Notes in Computer Science*, pages 21–40. Springer, Heidelberg, August 2010.

[DPW10] Stefan Dziembowski, Krzysztof Pietrzak, and Daniel Wichs. Non-malleable codes. In Andrew Chi-Chih Yao, editor, *ICS 2010: 1st Innovations in Computer Science*, pages 434–452. Tsinghua University Press, January 2010.

[DSS01] Yevgeniy Dodis, Amit Sahai, and Adam Smith. On perfect and adaptive security in exposure-resilient cryptography. In Birgit Pfitzmann, editor, *Advances in Cryptology – EUROCRYPT 2001*, volume 2045 of *Lecture Notes in Computer Science*, pages 301–324. Springer, Heidelberg, May 2001.

[Dzi06] Stefan Dziembowski. Intrusion-resilience via the bounded-storage model. In Shai Halevi and Tal Rabin, editors, *TCC 2006: 3rd Theory of Cryptography Conference*, volume 3876 of *Lecture Notes in Computer Science*, pages 207–224. Springer, Heidelberg, March 2006.

[EF18] William Enck and Adrienne Porter Felt, editors. *27th USENIX Security Symposium, USENIX Security 2018, Baltimore, MD, USA, August 15-17, 2018*. USENIX Association, 2018.

[ElG85] Taher ElGamal. A public key cryptosystem and a signature scheme based on discrete logarithms. *IEEE Transactions on Information Theory*, 31:469–472, 1985.

[FGP+18] Sebastian Faust, Vincent Grosso, Santos Merino Del Pozo, Clara Paglialonga, and François-Xavier Standaert. Composable masking schemes in the presence of physical defaults & the robust probing model. *IACR Transactions on Cryptographic Hardware and Embedded Systems*, 2018(3):89–120, 2018. `https://tches.iacr.org/index.php/TCHES/article/view/7270`.

[FH15] Benjamin Fuller and Ariel Hamlin. Unifying leakage classes: Simulatable leakage and pseudoentropy. In Anja Lehmann and Stefan Wolf, editors, *ICITS 15: 8th International Conference on Information Theoretic Security*, volume 9063 of *Lecture Notes in Computer Science*, pages 69–86. Springer, Heidelberg, May 2015.

[FKN+15] Eiichiro Fujisaki, Akinori Kawachi, Ryo Nishimaki, Keisuke Tanaka, and Kenji Yasunaga. Post-challenge leakage resilient public-key cryptosystem in split state model. *IEICE Transactions*, 98-A(3):853–862, 2015.

[FKPR10] Sebastian Faust, Eike Kiltz, Krzysztof Pietrzak, and Guy N. Rothblum. Leakage-resilient signatures. In Daniele Micciancio, editor, *TCC 2010: 7th Theory of Cryptography Conference*, volume 5978 of *Lecture Notes in Computer Science*, pages 343–360. Springer, Heidelberg, February 2010.

[FMNV15] Sebastian Faust, Pratyay Mukherjee, Jesper Buus Nielsen, and Daniele Venturi. A tamper and leakage resilient von neumann architecture. In Jonathan Katz, editor, *PKC 2015: 18th International Conference on Theory and Practice of Public Key Cryptography*, volume 9020 of *Lecture Notes in Computer Science*, pages 579–603. Springer, Heidelberg, March / April 2015.

[FMVW14] Sebastian Faust, Pratyay Mukherjee, Daniele Venturi, and Daniel Wichs. Efficient non-malleable codes and key-derivation for poly-size tampering circuits. In Phong Q. Nguyen and Elisabeth Oswald, editors, *Advances in Cryptology – EUROCRYPT 2014*, volume 8441 of *Lecture Notes in Computer Science*, pages 111–128. Springer, Heidelberg, May 2014.

[FN17] Antonio Faonio and Jesper Buus Nielsen. Fully leakage-resilient codes. In Serge Fehr, editor, *PKC 2017: 20th International Conference on Theory and Practice of Public Key Cryptography, Part I*, volume 10174 of *Lecture Notes in Computer Science*, pages 333–358. Springer, Heidelberg, March 2017.

[FNV15] Antonio Faonio, Jesper Buus Nielsen, and Daniele Venturi. Mind your coins: Fully leakage-resilient signatures with graceful degradation. In Magnús M. Halldórsson, Kazuo Iwama, Naoki Kobayashi, and Bettina Speckmann, editors, *ICALP 2015: 42nd International Colloquium on Automata, Languages and Programming, Part I*, volume 9134 of *Lecture Notes in Computer Science*, pages 456–468. Springer, Heidelberg, July 2015.

[FPS12] Sebastian Faust, Krzysztof Pietrzak, and Joachim Schipper. Practical leakage-resilient symmetric cryptography. In Emmanuel Prouff and Patrick Schaumont, editors, *Cryptographic Hardware and Embedded Systems – CHES 2012*, volume 7428 of *Lecture Notes in Computer Science*, pages 213–232. Springer, Heidelberg, September 2012.

[FPS17] Sebastian Faust, Clara Paglialonga, and Tobias Schneider. Amortizing randomness complexity in private circuits. In Tsuyoshi Takagi and Thomas Peyrin, editors, *Advances in Cryptology – ASIACRYPT 2017, Part I*, volume 10624 of *Lecture Notes in Computer Science*, pages 781–810. Springer, Heidelberg, December 2017.

[FPV11] Sebastian Faust, Krzysztof Pietrzak, and Daniele Venturi. Tamper-proof circuits: How to trade leakage for tamper-resilience. In Luca Aceto, Monika Henzinger, and Jiri Sgall, editors, *ICALP 2011: 38th International Colloquium on Automata, Languages and Programming, Part I*, volume 6755 of *Lecture Notes in Computer Science*, pages 391–402. Springer, Heidelberg, July 2011.

[FR12] Benjamin Fuller and Leonid Reyzin. Computational entropy and information leakage. Cryptology ePrint Archive, Report 2012/466, 2012. http://eprint.iacr.org/2012/466.

[FRR+10] Sebastian Faust, Tal Rabin, Leonid Reyzin, Eran Tromer, and Vinod Vaikuntanathan. Protecting circuits from leakage: the computationally-bounded and noisy cases. In Henri Gilbert, editor, *Advances in Cryptology – EUROCRYPT 2010*, volume 6110 of *Lecture Notes in Computer Science*, pages 135–156. Springer, Heidelberg, May / June 2010.

[FRR+14] Sebastian Faust, Tal Rabin, Leonid Reyzin, Eran Tromer, and Vinod Vaikuntanathan. Protecting circuits from computationally bounded and noisy leakage. *SIAM J. Comput.*, 43(5):1564–1614, 2014.

[FS90] Uriel Feige and Adi Shamir. Zero knowledge proofs of knowledge in two rounds. In Gilles Brassard, editor, *Advances in Cryptology – CRYPTO'89*, volume 435 of *Lecture Notes in Computer Science*, pages 526–544. Springer, Heidelberg, August 1990.

[GGL+16] David Galindo, Johann Großschädl, Zhe Liu, Praveen Kumar Vadnala, and Srinivas Vivek. Implementation of a leakage-resilient elgamal key encapsulation mechanism. *J. Cryptographic Engineering*, 6(3):229–238, 2016.

[GGM84] Oded Goldreich, Shafi Goldwasser, and Silvio Micali. On the cryptographic applications of random functions. In G. R. Blakley and David Chaum, editors, *Advances in Cryptology – CRYPTO'84*, volume 196 of *Lecture Notes in Computer Science*, pages 276–288. Springer, Heidelberg, August 1984.

[GGM86] Oded Goldreich, Shafi Goldwasser, and Silvio Micali. How to construct random functions. *Journal of the ACM*, 33(4):792–807, October 1986.

[GIM+16] Vipul Goyal, Yuval Ishai, Hemanta K. Maji, Amit Sahai, and Alexander A. Sherstov. Bounded-communication leakage resilience via parity-resilient circuits. In Irit Dinur, editor, *57th Annual Symposium on Foundations of Computer Science*, pages 1–10. IEEE Computer Society Press, October 2016.

[GIW17] Daniel Genkin, Yuval Ishai, and Mor Weiss. How to construct a leakage-resilient (stateless) trusted party. In Yael Kalai and Leonid Reyzin, editors, *TCC 2017: 15th Theory of Cryptography Conference, Part II*, volume 10678 of *Lecture Notes in Computer Science*, pages 209–244. Springer, Heidelberg, November 2017.

[GJR18] Dahmun Goudarzi, Antoine Joux, and Matthieu Rivain. How to securely compute with noisy leakage in quasilinear complexity. In Thomas Peyrin and Steven Galbraith, editors, *Advances in Cryptology – ASIACRYPT 2018, Part II*, volume 11273 of *Lecture Notes in Computer Science*, pages 547–574. Springer, Heidelberg, December 2018.

[GJS11] Sanjam Garg, Abhishek Jain, and Amit Sahai. Leakage-resilient zero knowledge. In Phillip Rogaway, editor, *Advances in Cryptology – CRYPTO 2011*, volume 6841 of *Lecture Notes in Computer Science*, pages 297–315. Springer, Heidelberg, August 2011.

[GK05] Shafi Goldwasser and Yael Tauman Kalai. On the impossibility of obfuscation with auxiliary input. In *46th Annual Symposium on Foundations of Computer Science*, pages 553–562. IEEE Computer Society Press, October 2005.

[GKPV10] Shafi Goldwasser, Yael Tauman Kalai, Chris Peikert, and Vinod Vaikuntanathan. Robustness of the learning with errors assumption. In Andrew Chi-Chih Yao, editor, *ICS 2010: 1st Innovations in Computer Science*, pages 230–240. Tsinghua University Press, January 2010.

[GL89] Oded Goldreich and Leonid A. Levin. A hard-core predicate for all one-way functions. In *21st Annual ACM Symposium on Theory of Computing*, pages 25–32. ACM Press, May 1989.

[GLM+04] Rosario Gennaro, Anna Lysyanskaya, Tal Malkin, Silvio Micali, and Tal Rabin. Algorithmic tamper-proof (ATP) security: Theoretical foundations for security against hardware tampering. In Moni Naor, editor, *TCC 2004: 1st Theory of Cryptography Conference*, volume 2951 of *Lecture Notes in Computer Science*, pages 258–277. Springer, Heidelberg, February 2004.

[GLP12] Tim Güneysu, Vadim Lyubashevsky, and Thomas Pöppelmann. Practical lattice-based cryptography: A signature scheme for embedded systems. In Emmanuel Prouff and Patrick Schaumont, editors, *Cryptographic Hardware and Embedded Systems – CHES 2012*, volume 7428 of *Lecture Notes in Computer Science*, pages 530–547. Springer, Heidelberg, September 2012.

[GLSV15] Vincent Grosso, Gaëtan Leurent, François-Xavier Standaert, and Kerem Varici. LS-designs: Bitslice encryption for efficient masked software implementations. In Carlos Cid and Christian Rechberger, editors, *Fast Software Encryption – FSE 2014*, volume 8540 of *Lecture Notes in Computer Science*, pages 18–37. Springer, Heidelberg, March 2015.

[GM82] Shafi Goldwasser and Silvio Micali. Probabilistic encryption and how to play mental poker keeping secret all partial information. In *14th Annual ACM Symposium on Theory of Computing*, pages 365–377. ACM Press, May 1982.

[GM84] Shafi Goldwasser and Silvio Micali. Probabilistic encryption. *Journal of Computer and System Sciences*, 28(2):270–299, 1984.

[GM17] Hannes Groß and Stefan Mangard. Reconciling d+1 masking in hardware and software. In Wieland Fischer and Naofumi Homma, editors, *Cryptographic Hardware and Embedded Systems – CHES 2017*, volume 10529 of *Lecture Notes in Computer Science*, pages 115–136. Springer, Heidelberg, September 2017.

[GMR84] Shafi Goldwasser, Silvio Micali, and Ronald L. Rivest. A "paradoxical" solution to the signature problem (abstract) (impromptu talk). In G. R. Blakley and David Chaum, editors, *Advances in Cryptology – CRYPTO'84*, volume 196 of *Lecture Notes in Computer Science*, page 467. Springer, Heidelberg, August 1984.

[GMR85] Shafi Goldwasser, Silvio Micali, and Charles Rackoff. The knowledge complexity of interactive proof-systems (extended abstract). In *17th Annual ACM Symposium on Theory of Computing*, pages 291–304. ACM Press, May 1985.

[GMR88] Shafi Goldwasser, Silvio Micali, and Ronald L. Rivest. A digital signature scheme secure against adaptive chosen-message attacks. *SIAM Journal on Computing*, 17(2):281–308, April 1988.

[GMR89] Shafi Goldwasser, Silvio Micali, and Charles Rackoff. The knowledge complexity of interactive proof systems. *SIAM Journal on Computing*, 18(1):186–208, 1989.

[GMW86] Oded Goldreich, Silvio Micali, and Avi Wigderson. Proofs that yield nothing but their validity and a methodology of cryptographic protocol design (extended abstract). In *27th Annual Symposium on Foundations of Computer Science*, pages 174–187. IEEE Computer Society Press, October 1986.

[GMW87] Oded Goldreich, Silvio Micali, and Avi Wigderson. How to play any mental game or A completeness theorem for protocols with honest majority. In Alfred Aho, editor, *19th Annual ACM Symposium on Theory of Computing*, pages 218–229. ACM Press, May 1987.

[GMW91] Oded Goldreich, Silvio Micali, and Avi Wigderson. Proofs that yield nothing but their validity or all languages in NP have zero-knowledge proof systems. *Journal of the ACM*, 38(3):691–729, 1991.

[GO96] Oded Goldreich and Rafail Ostrovsky. Software protection and simulation on oblivious rams. *J. ACM*, 43(3):431–473, 1996.

[GP99] Louis Goubin and Jacques Patarin. DES and differential power analysis (the "duplication" method). In Çetin Kaya Koç and Christof Paar, editors, *Cryptographic Hardware and Embedded Systems – CHES'99*, volume 1717 of *Lecture Notes in Computer Science*, pages 158–172. Springer, Heidelberg, August 1999.

[GPPS18] Chun Guo, Olivier Pereira, Thomas Peters, and François-Xavier Standaert. Leakage-resilient authenticated encryption with misuse in the leveled leakage setting: Definitions, separation results, and constructions. Cryptology ePrint Archive, Report 2018/484, 2018. `https://eprint.iacr.org/2018/484`.

[GR10] Shafi Goldwasser and Guy N. Rothblum. Securing computation against continuous leakage. In Tal Rabin, editor, *Advances in Cryptology – CRYPTO 2010*, volume 6223 of *Lecture Notes in Computer Science*, pages 59–79. Springer, Heidelberg, August 2010.

[GR12] Shafi Goldwasser and Guy N. Rothblum. How to compute in the presence of leakage. In *53rd Annual Symposium on Foundations of Computer Science*, pages 31–40. IEEE Computer Society Press, October 2012.

[GR15] Shafi Goldwasser and Guy N. Rothblum. How to compute in the presence of leakage. *SIAM J. Comput.*, 44(5):1480–1549, 2015.

[GR17] Dahmun Goudarzi and Matthieu Rivain. How fast can higher-order masking be in software? In Jean-Sébastien Coron and Jesper Buus Nielsen, editors, *Advances in Cryptology – EUROCRYPT 2017, Part I*, volume 10210 of *Lecture Notes in Computer Science*, pages 567–597. Springer, Heidelberg, April / May 2017.

[GS18] Vincent Grosso and François-Xavier Standaert. Masking proofs are tight and how to exploit it in security evaluations. In Jesper Buus Nielsen and Vincent Rijmen, editors, *Advances in Cryptology – EUROCRYPT 2018, Part II*, volume 10821 of *Lecture Notes in Computer Science*, pages 385–412. Springer, Heidelberg, April / May 2018.

[Gün90] Christoph G. Günther. An identity-based key-exchange protocol. In Jean-Jacques Quisquater and Joos Vandewalle, editors, *Advances in Cryptology – EUROCRYPT'89*, volume 434 of *Lecture Notes in Computer Science*, pages 29–37. Springer, Heidelberg, April 1990.

[GV13a] David Galindo and Srinivas Vivek. A leakage-resilient pairing-based variant of the Schnorr signature scheme. In Martijn Stam, editor, *14th IMA International Conference on Cryptography and Coding*, volume 8308 of *Lecture Notes in Computer Science*, pages 173–192. Springer, Heidelberg, December 2013.

[GV13b] David Galindo and Srinivas Vivek. A practical leakage-resilient signature scheme in the generic group model. In Lars R. Knudsen and Huapeng Wu, editors, *SAC 2012: 19th Annual International Workshop on Selected Areas in Cryptography*, volume 7707 of *Lecture Notes in Computer Science*, pages 50–65. Springer, Heidelberg, August 2013.

[HJJ+97] Amir Herzberg, Markus Jakobsson, Stanislaw Jarecki, Hugo Krawczyk, and Moti Yung. Proactive public key and signature systems. In *ACM CCS 97: 4th Conference on Computer and Communications Security*, pages 100–110. ACM Press, April 1997.

[HKL+12] Stefan Heyse, Eike Kiltz, Vadim Lyubashevsky, Christof Paar, and Krzysztof Pietrzak. Lapin: An efficient authentication protocol based on ring-LPN. In Anne Canteaut, editor, *Fast Software Encryption – FSE 2012*, volume 7549 of *Lecture Notes in Computer Science*, pages 346–365. Springer, Heidelberg, March 2012.

[HL11] Shai Halevi and Huijia Lin. After-the-fact leakage in public-key encryption. In Yuval Ishai, editor, *TCC 2011: 8th Theory of Cryptography Conference*, volume 6597 of *Lecture Notes in Computer Science*, pages 107–124. Springer, Heidelberg, March 2011.

[HLWW13] Carmit Hazay, Adriana López-Alt, Hoeteck Wee, and Daniel Wichs. Leakage-resilient cryptography from minimal assumptions. In Thomas Johansson and Phong Q. Nguyen, editors, *Advances in Cryptology – EUROCRYPT 2013*, volume 7881 of *Lecture Notes in Computer Science*, pages 160–176. Springer, Heidelberg, May 2013.

[HLWW16] Carmit Hazay, Adriana López-Alt, Hoeteck Wee, and Daniel Wichs. Leakage-resilient cryptography from minimal assumptions. *Journal of Cryptology*, 29(3):514–551, July 2016.

[HSH+08] 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, *Proceedings of the 17th USENIX Security Symposium, July 28-August 1, 2008, San Jose, CA, USA*, pages 45–60. USENIX Association, 2008.

[HSH+09] 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. *Commun. ACM*, 52(5):91–98, 2009.

[IKL+13] Yuval Ishai, Eyal Kushilevitz, Xin Li, Rafail Ostrovsky, Manoj Prabhakaran, Amit Sahai, and David Zuckerman. Robust pseudorandom generators. In Fedor V. Fomin, Rusins Freivalds, Marta Z. Kwiatkowska, and David Peleg, editors, *ICALP 2013: 40th International Colloquium on Automata, Languages and Programming, Part I*, volume 7965 of *Lecture Notes in Computer Science*, pages 576–588. Springer, Heidelberg, July 2013.

[IPSW06] Yuval Ishai, Manoj Prabhakaran, Amit Sahai, and David Wagner. Private circuits II: Keeping secrets in tamperable circuits. In Serge Vaudenay, editor, *Advances in Cryptology – EUROCRYPT 2006*, volume 4004 of *Lecture Notes in Computer Science*, pages 308–327. Springer, Heidelberg, May / June 2006.

[IR02] Gene Itkis and Leonid Reyzin. SiBIR: Signer-Base Intrusion-Resilient signatures. In Moti Yung, editor, *Advances in Cryptology – CRYPTO 2002*, volume 2442 of *Lecture Notes in Computer Science*, pages 499–514. Springer, Heidelberg, August 2002.

[ISW03] Yuval Ishai, Amit Sahai, and David Wagner. Private circuits: Securing hardware against probing attacks. In Dan Boneh, editor, *Advances in Cryptology – CRYPTO 2003*, volume 2729 of *Lecture Notes in Computer Science*, pages 463–481. Springer, Heidelberg, August 2003.

[IWY16] Yuval Ishai, Mor Weiss, and Guang Yang. Making the best of a leaky situation: Zero-knowledge PCPs from leakage-resilient circuits. In Eyal Kushilevitz and Tal Malkin, editors, *TCC 2016-A: 13th Theory of Cryptography Conference, Part II*, volume 9563 of *Lecture Notes in Computer Science*, pages 3–32. Springer, Heidelberg, January 2016.

[JS17] Anthony Journault and François-Xavier Standaert. Very high order masking: Efficient implementation and security evaluation. In Wieland Fischer and Naofumi Homma, editors, *Cryptographic Hardware and Embedded Systems – CHES 2017*, volume 10529 of *Lecture Notes in Computer Science*, pages 623–643. Springer, Heidelberg, September 2017.

[JV10] Ali Juma and Yevgeniy Vahlis. Protecting cryptographic keys against continual leakage. In Tal Rabin, editor, *Advances in Cryptology – CRYPTO 2010*, volume 6223 of *Lecture Notes in Computer Science*, pages 41–58. Springer, Heidelberg, August 2010.

[JW15] Zahra Jafargholi and Daniel Wichs. Tamper detection and continuous non-malleable codes. In Yevgeniy Dodis and Jesper Buus Nielsen, editors, *TCC 2015: 12th Theory of*

Cryptography Conference, Part I, volume 9014 of *Lecture Notes in Computer Science*, pages 451–480. Springer, Heidelberg, March 2015.

[KHF+19] Paul Kocher, Jann Horn, Anders Fogh, , Daniel Genkin, Daniel Gruss, Werner Haas, Mike Hamburg, Moritz Lipp, Stefan Mangard, Thomas Prescher, Michael Schwarz, and Yuval Yarom. Spectre attacks: Exploiting speculative execution. In *40th IEEE Symposium on Security and Privacy (S&P'19)*, 2019.

[KJJ99] Paul C. Kocher, Joshua Jaffe, and Benjamin Jun. Differential power analysis. In Michael J. Wiener, editor, *Advances in Cryptology – CRYPTO'99*, volume 1666 of *Lecture Notes in Computer Science*, pages 388–397. Springer, Heidelberg, August 1999.

[Koc96] Paul C. Kocher. Timing attacks on implementations of Diffie-Hellman, RSA, DSS, and other systems. In Neal Koblitz, editor, *Advances in Cryptology – CRYPTO'96*, volume 1109 of *Lecture Notes in Computer Science*, pages 104–113. Springer, Heidelberg, August 1996.

[Koc03] Paul C. Kocher. Leak-resistant cryptographic indexed key update, 2003. US Patent 65539092.

[KP10] Eike Kiltz and Krzysztof Pietrzak. Leakage resilient ElGamal encryption. In Masayuki Abe, editor, *Advances in Cryptology – ASIACRYPT 2010*, volume 6477 of *Lecture Notes in Computer Science*, pages 595–612. Springer, Heidelberg, December 2010.

[KV09] Jonathan Katz and Vinod Vaikuntanathan. Signature schemes with bounded leakage resilience. In Mitsuru Matsui, editor, *Advances in Cryptology – ASIACRYPT 2009*, volume 5912 of *Lecture Notes in Computer Science*, pages 703–720. Springer, Heidelberg, December 2009.

[Lam79] Leslie Lamport. Constructing digital signatures from a one-way function. Technical Report SRI-CSL-98, SRI International Computer Science Laboratory, October 1979.

[LL12] Feng-Hao Liu and Anna Lysyanskaya. Tamper and leakage resilience in the split-state model. In Reihaneh Safavi-Naini and Ran Canetti, editors, *Advances in Cryptology – CRYPTO 2012*, volume 7417 of *Lecture Notes in Computer Science*, pages 517–532. Springer, Heidelberg, August 2012.

[LLW11] Allison B. Lewko, Mark Lewko, and Brent Waters. How to leak on key updates. In Lance Fortnow and Salil P. Vadhan, editors, *43rd Annual ACM Symposium on Theory of Computing*, pages 725–734. ACM Press, June 2011.

[LMO+14] Jake Longo, Daniel P. Martin, Elisabeth Oswald, Daniel Page, Martijn Stam, and Michael Tunstall. Simulatable leakage: Analysis, pitfalls, and new constructions. In Palash Sarkar and Tetsu Iwata, editors, *Advances in Cryptology – ASIACRYPT 2014, Part I*, volume 8873 of *Lecture Notes in Computer Science*, pages 223–242. Springer, Heidelberg, December 2014.

[LSG+18] Moritz Lipp, Michael Schwarz, Daniel Gruss, Thomas Prescher, Werner Haas, Anders Fogh, Jann Horn, Stefan Mangard, Paul Kocher, Daniel Genkin, Yuval Yarom, and

Mike Hamburg. Meltdown: Reading kernel memory from user space. In Enck and Felt [EF18], pages 973–990.

[Mer88] Ralph C. Merkle. A digital signature based on a conventional encryption function. In Carl Pomerance, editor, *Advances in Cryptology – CRYPTO'87*, volume 293 of *Lecture Notes in Computer Science*, pages 369–378. Springer, Heidelberg, August 1988.

[Mil14] Eric Miles. Iterated group products and leakage resilience against NC1. In Moni Naor, editor, *ITCS 2014: 5th Conference on Innovations in Theoretical Computer Science*, pages 261–268. Association for Computing Machinery, January 2014.

[MOSW15] Daniel P. Martin, Elisabeth Oswald, Martijn Stam, and Marcin Wójcik. A leakage resilient MAC. In Jens Groth, editor, *15th IMA International Conference on Cryptography and Coding*, volume 9496 of *Lecture Notes in Computer Science*, pages 295–310. Springer, Heidelberg, December 2015.

[MR04] Silvio Micali and Leonid Reyzin. Physically observable cryptography (extended abstract). In Moni Naor, editor, *TCC 2004: 1st Theory of Cryptography Conference*, volume 2951 of *Lecture Notes in Computer Science*, pages 278–296. Springer, Heidelberg, February 2004.

[MTVY11] Tal Malkin, Isamu Teranishi, Yevgeniy Vahlis, and Moti Yung. Signatures resilient to continual leakage on memory and computation. In Yuval Ishai, editor, *TCC 2011: 8th Theory of Cryptography Conference*, volume 6597 of *Lecture Notes in Computer Science*, pages 89–106. Springer, Heidelberg, March 2011.

[MV13] Eric Miles and Emanuele Viola. Shielding circuits with groups. In Dan Boneh, Tim Roughgarden, and Joan Feigenbaum, editors, *45th Annual ACM Symposium on Theory of Computing*, pages 251–260. ACM Press, June 2013.

[NR18] Jesper Buus Nielsen and Vincent Rijmen, editors. *Advances in Cryptology - EUROCRYPT 2018 - 37th Annual International Conference on the Theory and Applications of Cryptographic Techniques, Tel Aviv, Israel, April 29 - May 3, 2018 Proceedings, Part II*, volume 10821 of *Lecture Notes in Computer Science*. Springer, 2018.

[NS09] Moni Naor and Gil Segev. Public-key cryptosystems resilient to key leakage. In Shai Halevi, editor, *Advances in Cryptology – CRYPTO 2009*, volume 5677 of *Lecture Notes in Computer Science*, pages 18–35. Springer, Heidelberg, August 2009.

[NVZ13] Jesper Buus Nielsen, Daniele Venturi, and Angela Zottarel. On the connection between leakage tolerance and adaptive security. In Kaoru Kurosawa and Goichiro Hanaoka, editors, *PKC 2013: 16th International Conference on Theory and Practice of Public Key Cryptography*, volume 7778 of *Lecture Notes in Computer Science*, pages 497–515. Springer, Heidelberg, February / March 2013.

[NY90] Moni Naor and Moti Yung. Public-key cryptosystems provably secure against chosen ciphertext attacks. In *22nd Annual ACM Symposium on Theory of Computing*, pages 427–437. ACM Press, May 1990.

[NZ96] Noam Nisan and David Zuckerman. Randomness is linear in space. *J. Comput. Syst. Sci.*, 52(1):43–52, 1996.

[Oka93] Tatsuaki Okamoto. Provably secure and practical identification schemes and corresponding signature schemes. In Ernest F. Brickell, editor, *Advances in Cryptology – CRYPTO'92*, volume 740 of *Lecture Notes in Computer Science*, pages 31–53. Springer, Heidelberg, August 1993.

[Pie09] Krzysztof Pietrzak. A leakage-resilient mode of operation. In Antoine Joux, editor, *Advances in Cryptology – EUROCRYPT 2009*, volume 5479 of *Lecture Notes in Computer Science*, pages 462–482. Springer, Heidelberg, April 2009.

[PR13] Emmanuel Prouff and Matthieu Rivain. Masking against side-channel attacks: A formal security proof. In Thomas Johansson and Phong Q. Nguyen, editors, *Advances in Cryptology – EUROCRYPT 2013*, volume 7881 of *Lecture Notes in Computer Science*, pages 142–159. Springer, Heidelberg, May 2013.

[PRC12] Gilles Piret, Thomas Roche, and Claude Carlet. PICARO - a block cipher allowing efficient higher-order side-channel resistance. In Feng Bao, Pierangela Samarati, and Jianying Zhou, editors, *ACNS 12: 10th International Conference on Applied Cryptography and Network Security*, volume 7341 of *Lecture Notes in Computer Science*, pages 311–328. Springer, Heidelberg, June 2012.

[PSV15] Olivier Pereira, François-Xavier Standaert, and Srinivas Vivek. Leakage-resilient authentication and encryption from symmetric cryptographic primitives. In Indrajit Ray, Ninghui Li, and Christopher Kruegel:, editors, *ACM CCS 15: 22nd Conference on Computer and Communications Security*, pages 96–108. ACM Press, October 2015.

[Reg05] Oded Regev. On lattices, learning with errors, random linear codes, and cryptography. In Harold N. Gabow and Ronald Fagin, editors, *37th Annual ACM Symposium on Theory of Computing*, pages 84–93. ACM Press, May 2005.

[Riv97] Ronald L. Rivest. All-or-nothing encryption and the package transform. In Eli Biham, editor, *Fast Software Encryption – FSE'97*, volume 1267 of *Lecture Notes in Computer Science*, pages 210–218. Springer, Heidelberg, January 1997.

[Rot12] Guy N. Rothblum. How to compute under $\mathcal{AC}^0$ leakage without secure hardware. In Reihaneh Safavi-Naini and Ran Canetti, editors, *Advances in Cryptology – CRYPTO 2012*, volume 7417 of *Lecture Notes in Computer Science*, pages 552–569. Springer, Heidelberg, August 2012.

[RTTV08] Omer Reingold, Luca Trevisan, Madhur Tulsiani, and Salil P. Vadhan. Dense subsets of pseudorandom sets. In *49th Annual Symposium on Foundations of Computer Science*, pages 76–85. IEEE Computer Society Press, October 2008.

[Sah99] Amit Sahai. Non-malleable non-interactive zero knowledge and adaptive chosen-ciphertext security. In *40th Annual Symposium on Foundations of Computer Science*, pages 543–553. IEEE Computer Society Press, October 1999.

[Sch91] Claus-Peter Schnorr. Efficient signature generation by smart cards. *Journal of Cryptology*, 4(3):161–174, January 1991.

[Sch10] Joachim Schipper. *Leakage-Resilient Authentication*. PhD thesis, Utrecht University, 2010.

[Sha79] Adi Shamir. How to share a secret. *Communications of the Association for Computing Machinery*, 22(11):612–613, November 1979.

[SMY09] François-Xavier Standaert, Tal Malkin, and Moti Yung. A unified framework for the analysis of side-channel key recovery attacks. In Antoine Joux, editor, *Advances in Cryptology – EUROCRYPT 2009*, volume 5479 of *Lecture Notes in Computer Science*, pages 443–461. Springer, Heidelberg, April 2009.

[SPY+10] François-Xavier Standaert, Olivier Pereira, Yu Yu, Jean-Jacques Quisquater, Moti Yung, and Elisabeth Oswald. Leakage resilient cryptography in practice. Information Security and Cryptography, pages 99–134. Springer, Heidelberg, 2010.

[SPY13] François-Xavier Standaert, Olivier Pereira, and Yu Yu. Leakage-resilient symmetric cryptography under empirically verifiable assumptions. In Ran Canetti and Juan A. Garay, editors, *Advances in Cryptology – CRYPTO 2013, Part I*, volume 8042 of *Lecture Notes in Computer Science*, pages 335–352. Springer, Heidelberg, August 2013.

[SZ13] Adam Smith and Ye Zhang. Near-linear time, leakage-resilient key evolution schemes from expander graphs. Cryptology ePrint Archive, Report 2013/864, 2013. `http://eprint.iacr.org/2013/864`.

[TLNL14] Fei Tang, Hongda Li, Qihua Niu, and Bei Liang. Efficient leakage-resilient signature schemes in the generic bilinear group model. In Xinyi Huang and Jianying Zhou, editors, *Information Security Practice and Experience - 10th International Conference, ISPEC 2014, Fuzhou, China, May 5-8, 2014. Proceedings*, volume 8434 of *Lecture Notes in Computer Science*, pages 418–432. Springer, 2014.

[TV00] Luca Trevisan and Salil P. Vadhan. Extracting randomness from samplable distributions. In *41st Annual Symposium on Foundations of Computer Science*, pages 32–42. IEEE Computer Society Press, November 2000.

[Wat05] Brent R. Waters. Efficient identity-based encryption without random oracles. In Ronald Cramer, editor, *Advances in Cryptology – EUROCRYPT 2005*, volume 3494 of *Lecture Notes in Computer Science*, pages 114–127. Springer, Heidelberg, May 2005.

[WO11a] Carolyn Whitnall and Elisabeth Oswald. A comprehensive evaluation of mutual information analysis using a fair evaluation framework. In Phillip Rogaway, editor, *Advances in Cryptology – CRYPTO 2011*, volume 6841 of *Lecture Notes in Computer Science*, pages 316–334. Springer, Heidelberg, August 2011.

[WO11b] Carolyn Whitnall and Elisabeth Oswald. A fair evaluation framework for comparing side-channel distinguishers. *J. Cryptographic Engineering*, 1(2):145–160, 2011.

[WTH16] Jui-Di Wu, Yuh-Min Tseng, and Sen-Shan Huang. Leakage-resilient id-based signature scheme in the generic bilinear group model. *Security and Communication Networks*, 9(17):3987–4001, 2016.

[Yao82] Andrew Chi-Chih Yao. Theory and applications of trapdoor functions (extended abstract). In *23rd Annual Symposium on Foundations of Computer Science*, pages 80–91. IEEE Computer Society Press, November 1982.

[YCZY12] Tsz Hon Yuen, Sherman S. M. Chow, Ye Zhang, and Siu Ming Yiu. Identity-based encryption resilient to continual auxiliary leakage. In David Pointcheval and Thomas Johansson, editors, *Advances in Cryptology – EUROCRYPT 2012*, volume 7237 of *Lecture Notes in Computer Science*, pages 117–134. Springer, Heidelberg, April 2012.

[YS13] Yu Yu and François-Xavier Standaert. Practical leakage-resilient pseudorandom objects with minimum public randomness. In Ed Dawson, editor, *Topics in Cryptology – CT-RSA 2013*, volume 7779 of *Lecture Notes in Computer Science*, pages 223–238. Springer, Heidelberg, February / March 2013.

[YSPY10] Yu Yu, François-Xavier Standaert, Olivier Pereira, and Moti Yung. Practical leakage-resilient pseudorandom generators. In Ehab Al-Shaer, Angelos D. Keromytis, and Vitaly Shmatikov, editors, *ACM CCS 10: 17th Conference on Computer and Communications Security*, pages 141–151. ACM Press, October 2010.

[ZCC15] Zongyang Zhang, Sherman S. M. Chow, and Zhenfu Cao. Post-challenge leakage in public-key encryption. *Theor. Comput. Sci.*, 572:25–49, 2015.