Leakage-Resilient Cryptography from Minimal Assumptions

Carmit Hazay[1,⋆], Adriana López-Alt[2,⋆⋆], Hoeteck Wee[3,⋆⋆⋆], and Daniel Wichs[4,†]

[1] Bar-Ilan University
[2] New York University
[3] George Washington University
[4] Northeastern University

Abstract. We present new constructions of leakage-resilient cryptosystems, which remain provably secure even if the attacker learns some arbitrary partial information about their internal secret key. For any polynomial ℓ, we can instantiate these schemes so as to tolerate up to ℓ bits of leakage. While there has been much prior work constructing such leakage-resilient cryptosystems under concrete number-theoretic and algebraic assumptions, we present the first schemes under general and minimal assumptions. In particular, we construct:

- Leakage-resilient *public-key encryption* from any standard public-key encryption.
- Leakage-resilient *weak pseudorandom functions*, *symmetric-key encryption*, and *message-authentication codes* from any one-way function.

These are the first constructions of leakage-resilient *symmetric-key* primitives that do not rely on *public-key* assumptions. We also get the first constructions of leakage-resilient public-key encryption from "search assumptions", such as the hardness of factoring or CDH. Although our schemes can tolerate arbitrarily large *amounts* of leakage, the tolerated *rate* of leakage (defined as the ratio of leakage-amount to key-size) is rather poor in comparison to prior results under specific assumptions.

As a building block of independent interest, we study a notion of *weak* hash-proof systems in the public-key and symmetric-key settings. While these inherit some of the interesting security properties of standard hash-proof systems, we can instantiate them under general assumptions.

1 Introduction

A central goal in cryptography is to base cryptosystems on intractability assumptions that are as weak and as general as possible; that way, if one

⋆ Research partially conducted while at Aarhus University.
⋆⋆ Supported by AT&T Labs Fellowship. Research partially conducted while at Aarhus University.
⋆⋆⋆ Supported by NSF CAREER Award CNS-1237429.
† Research conducted while at IBM Research, T.J. Watson.

T. Johansson and P. Nguyen (Eds.): EUROCRYPT 2013, LNCS 7881, pp. 160–176, 2013.

problem turns out to be susceptible to a new attack or if another turns out to yield better performance, we may readily replace the underlying problem in our cryptosystem. Another goal is to design cryptosystems in strong security models that account for a wide range of possible attacks. Our work lies at the intersection of these two areas, by studying leakage-resilient security under general and minimal assumptions.

Leakage Resilience. Leakage-resilient cryptosystems maintain their security even if an attacker can learn some partial information about the internal secret key. Aside from being a basic question of theoretical interest, the study of leakage-resilience is motivated by several real-world scenarios where information leaks. One such scenario involves side-channel attacks, where the physical attributes of a computing device (e.g., its power consumption, electromagnetic radiation, timing, temperature, acoustics, etc.) can reveal information about its internal secret state. See e.g., [1, 5, 24, 38, 39, 46–48] for many examples of such attacks that completely break otherwise secure cryptosystems. Another source of leakage occurs through imperfect erasures (such as in the *cold-boot* attack [27]), where memory contents, including secret key information, aren't properly erased and some partial information becomes available to an attacker. Another source of leakage occurs if the secret key is stored on a compromised system to which the attacker has remote access. As suggested in prior work, we can impede an attacker from retrieving the secret key in its entirety by making it deliberately huge (e.g., many gigabytes in length), but the attacker can still obtain some partial leakage [4, 12, 16, 21]. As yet another example, we may need to use a cryptosystem within the context of a larger protocol that intentionally leaks some information about the secret key as a part of its design. Leakage-resilience provides a powerful tool, allowing us to easily analyze the security of such constructions. In summary, we believe that leakage-resilience is an interesting and fundamental property worth studying because of its relevance to many diverse problems including (but not limited to) side-channel attacks.

Bounded-Leakage Model. There are several security models of leakage-resilience in the literature, differing in their specification of what information can become available to the attacker. In this work we will focus on a simple yet general model, called the *bounded-leakage* (or sometimes *memory leakage*) model, which has received much attention in recent years [2–4,6–9,11–13,18,22,25,29,35,37,43]. In this model, the attacker can learn arbitrary information about the secret key, as long as the total number of bits learned is bounded by some parameter ℓ, called the *leakage bound*. We formalize this security notion by giving the attacker access to a *leakage oracle* that she can repeatedly and adaptively query; each query to the oracle consists of a *leakage function* f and the oracle responds with the "leakage" $f(\mathrm{SK})$ computed on secret key SK. The leakage oracle is only restricted in the total number of bits that it outputs throughout its lifetime, which is bounded by ℓ. This model is particularly interesting because of its elegance and simplicity and its wide applicability to scenarios such as incomplete erasure, compromised systems, and information released by high-level protocols.

We note that several other models of leakage-resilience consider a more complex scenario, where information can leak continually over time, with no overall bound on the total amount of leakage. See [10,17,19,23,26,34,41,42,45] for some examples. These models may offer a more realistic view of side-channel attacks, where many measurements may be made by an attacker over time. Many of these works rely on results from the bounded-leakage model as basic building blocks. Therefore, we believe that a thorough understanding of the bounded-leakage model is a necessary, but perhaps not sufficient, prerequisite to understanding other more complex models. We mention that it remains debatable how accurately any of the above models reflects realistic side-channel attacks (see e.g., the discussion in [49]).

Prior Constructions. It turns out that essentially all cryptographic schemes are already resilient against small amounts of leakage. In particular, they can tolerate $\ell = O(\log(\lambda))$ bits of leakage, where λ is the security parameter, and schemes with stronger *exact security* can tolerate correspondingly larger amounts of leakage. Intuitively, this follows since we can correctly "guess" small leakage values with reasonable probability and hence they cannot help in an attack.[1]

Most prior research in leakage-resilient cryptography attempts to construct schemes that provably tolerate larger amounts of leakage, without making any strong exact-security assumptions on the underlying primitives. Ultimately, we aim to tolerate any polynomial leakage bound $\ell(\lambda)$ just by instantiating the scheme with a sufficiently large secret key. Prior to this work, we had such results for public-key encryption [2, 8, 43], under specific assumptions such as LWE, DDH, DCR, QR, or somewhat more generally, the existence of "hash-proof systems". We also had such results for signatures [4,18,37] assuming the existence of NIZKs and public-key encryption. Essentially nothing better was known for symmetric-key encryption or message-authentication codes, beyond simply using the corresponding public-key constructions in the symmetric setting.

Our Main Results. We present new constructions of several leakage-resilient cryptosystems under the *minimal assumption* that such cryptosystems exist in the standard setting, without any leakage. For any polynomial leakage-bound $\ell(\lambda)$ in the security parameter λ, we can instantiate these schemes so as to resit $\ell(\lambda)$ bits of leakage. In particular, we construct the following primitives:

- Leakage-resilient *public-key encryption* from any public-key encryption.
- Leakage-resilient *weak pseudorandom functions*, *symmetric-key encryption*, and *message-authentication codes* from any one-way function.

We only assume the underlying primitives satisfy the usual asymptotic notion of security, and do *not* require any stronger levels of exact security. These results give us the first constructions of leakage-resilient symmetric-key primitives

[1] This simple argument works for "unpredictability" applications such as signatures. A more subtle argument also works for many "indistinguishability" applications, including public-key encryption, weak-PRFs and symmetric-key CPA encryption (but not, e.g., one-time encryption). See [20] for a general treatment of this question.

that do not rely on public-key assumptions. They also give us the first constructions of leakage-resilient public-key encryption from several specific "search assumptions" such as the hardness of RSA, factoring, or CDH.

Leakage Amount vs. Rate. Although our schemes can tolerate an arbitrarily large polynomial *amount* of leakage ℓ, the tolerated *rate* of leakage (defined as the ratio of ℓ to the secret-key size) in these constructions is rather poor. In particular, the leakage rate in our schemes is $O(\log(\lambda)/s(\lambda))$ where $s(\lambda)$ is the secret-key size of the underlying non-leakage-resilient primitives. In contrast, the state-of-the-art constructions of leakage-resilient schemes from concrete number-theoretic assumptions such as DDH can usually achieve a $(1-o(1))$ leakage rate, meaning that almost the entire secret key can leak. Allowing higher leakage rates under general assumptions remains as an open problem.

Extensions of Our Results. We explore several extensions of our main results. Firstly, we show that all of the results also apply to an alternate notion of *entropy-bounded leakage* [17, 43], where we restrict the amount of entropy-loss caused by the leakage rather than restricting its length. Unlike length-bounded leakage, achieving resilience to even 1-bit of entropy-bounded leakage is non-trivial and does not follow from the standard security of a scheme. We also show that our public/symmetric key encryption schemes provide resilience to *"after-the-fact"* leakage as defined by Halevi and Lin [29]. In particular, if the attacker can choose to learn some arbitrary ℓ_{post} bits of leakage on the secret key adaptively *after* seing a challenge ciphertext, she learns no more than ℓ_{post} bits of information about the encrypted message (in contrast, if the leakage is independent of the challenge ciphertext, she learns nothing about the message). Lastly, we extend our results to the *bounded-retrieval model* [4,12,16,21], where we want to have efficient schemes tolerating huge amounts (many gigabytes) of leakage, meaning that the efficiency of the scheme should not degrade even as the leakage-bound ℓ increases. Since the secret-key size of such schemes must exceed ℓ and therefore also be huge, these schemes cannot even read their entire secret key during each cryptographic operation. This model is motivated by the problem of system compromise, where an attacker can download large amounts of data from a compromised system. Due to space limitations, we defer the extensions of our results to entropy-bounded leakage, after-the-fact leakage, and the bounded-retrieval model to the full version of this paper [31].

1.1 Overview of Our Techniques

Our starting point is a result of Naor and Segev [43] (journal version [44]), which constructs leakage-resilient public-key encryption from any *hash-proof system* (HPS) [15]. As observed in [4, 43], this construction does not require the full security notion of HPS and it turns out that a weaker variant, which we will call a *weak* HPS (wHPS), actually suffices.[2] As our first result we show

[2] Although this weaker variant was used by [4, 43] to simplify the exposition of HPS, neither work seemed to considered the differences between wHPS and full HPS as very significant.

that, surprisingly, wHPS can be constructed generically from any public-key encryption scheme. This is in contrast to the full notion of HPS, which we only know how to construct from concrete number-theoretic assumptions such as DDH, DCR or QR. This gives us our results for *public-key encryption*. Next, we also define a new and meaningful notion of a *symmetric-key* wHPS, which allows us to construct leakage-resilient *weak pseudo-random functions* and *symmetric-key encryption*. We show how to construct symmetric-key wHPS generically from any pseudorandom function (PRF), and hence only under the assumption that one-way functions exist. Lastly, we employ several additional ideas to construct leakage-resilient *message authentication codes*.

We briefly define wHPS, how it relates to leakage resilience, and how to construct it. We focus on the public-key setting since it is conceptually simpler.

Weak Hash-Proof Systems (wHPS). A weak hash-proof system (wHPS) can be thought of as a special type of *key-encapsulation mechanism*. It consists of:

- A public-key *encapsulation* algorithm $(c, k) \leftarrow \mathsf{Encap}(\mathrm{PK})$ that creates a ciphertext c encapsulating a random secret value k.
- A secret-key *decapsulation* algorithm $k = \mathsf{Decap}(\mathrm{SK}, c)$ that recovers k from the ciphertext c.

Within the security definition of wHPS, we also require an additional *invalid encapsulation* algorithm $c^* \leftarrow \mathsf{Encap}^*(\mathrm{PK})$, which is *not* used by honest parties. The scheme must satisfy the following:

- Ciphertext Indistinguishability: Valid ciphertexts $(c, \cdot) \leftarrow \mathsf{Encap}(\mathrm{PK})$ are computationally indistinguishable from invalid ciphertexts $c^* \leftarrow \mathsf{Encap}^*(\mathrm{PK})$, even given the secret key SK.
- Smoothness: Let $(\mathrm{PK}, \mathrm{SK})$ be a random wHPS key pair and $c^* \leftarrow \mathsf{Encap}^*(\mathrm{PK})$ be a random *invalid* ciphertext. Given PK and c^*, the output $k = \mathsf{Decap}(\mathrm{SK}, c^*)$ is uniformly random and independent (information theoretically). The randomness of k comes from the choice of the secret key SK consistent with PK, meaning that there must be multiple ones.

In other words, the secret key SK maintains real entropy even conditioned on PK, and this entropy is transferred to the output $k = \mathsf{Decap}(\mathrm{SK}, c^*)$ when we decapsulate a random invalid ciphertext c^*.

The above definition of wHPS departs from that of standard hash-proof systems in several ways, but most importantly, our "smoothness" property is defined for an *average-case* invalid ciphertext $c^* \leftarrow \mathsf{wHPS.Encap}^*(\mathrm{PK})$ rather than a worst-case choice of c^* from some invalid set. Indeed, this makes our definition unsuitable for applications dealing with chosen-ciphertext (CCA or even CCA-1) security, for which hash-proof systems were originally intended.

Leakage-Resilience from wHPS. Weak hash-proof systems are particularly suited for leakage-resilience. Assume the attacker gets a wHPS public-key PK and observes ℓ bits of leakage on the secret key SK. Later, the attacker sees a random valid ciphertext c computed via $(c, k) \leftarrow \mathsf{Encap}(\mathrm{PK})$; what has she learned about the hidden value k? Firstly, we can switch c to an invalid ciphertext

$c^* \leftarrow \mathsf{Encap}^*(\mathrm{PK})$ and define $k = \mathsf{Decap}(c^*, \mathrm{SK})$. This change is indistinguishable even given the secret key SK in full, and therefore also when only given leakage on SK. Secondly, because $k = \mathsf{Decap}(c^*, \mathrm{SK})$ is information-theoretically random even when given PK and c^*, the ℓ-bits of leakage that the attacker observes about SK can reduce the entropy of k by at most ℓ bits. Therefore, if k is sufficiently large, it still has high entropy given the view of the attacker, and we can easily convert it to a uniformly random value using a randomness extractor. The above argument closely follows that of [43].

Constructing wHPS. Our main result for public-key encryption is to construct wHPS from general assumptions. As a starting point, we give a very simple construction where the output $k \in \{0,1\}$ consists of a single bit. We do so given any standard public-key encryption (PKE) scheme, as follows:

- Choose two random PKE key-pairs $(\mathrm{PK}_0, \mathrm{SK}_0), (\mathrm{PK}_1, \mathrm{SK}_1)$ and define the wHPS public-key as $\mathrm{PK} = (\mathrm{PK}_0, \mathrm{PK}_1)$ and the wHPS secret key as $\mathrm{SK} = (b, \mathrm{SK}_b)$ where $b \leftarrow \{0,1\}$ is a random bit. Notice that, given PK, there are at least two possible consistent secret keys: $(0, \mathrm{SK}_0)$ and $(1, \mathrm{SK}_1)$.
- The valid encapsulation algorithm $(c, k) \leftarrow \mathsf{Encap}(\mathrm{PK})$ chooses a random bit $k \leftarrow \{0,1\}$ and sets $c = (c_0, c_1)$ where $c_0 \leftarrow \mathsf{PKE.Enc}(\mathrm{PK}_0, k), c_1 \leftarrow \mathsf{PKE.Enc}(\mathrm{PK}_1, k)$ both encrypt the *same* bit k.
- The invalid encapsulation algorithm $c^* \leftarrow \mathsf{Encap}^*(\mathrm{PK})$ chooses a random bit $k \leftarrow \{0,1\}$ and sets $c^* = (c_0, c_1)$ where $c_0 \leftarrow \mathsf{PKE.Enc}(\mathrm{PK}_0, k), c_1 \leftarrow \mathsf{PKE.Enc}(\mathrm{PK}_1, 1-k)$ encrypt *opposite* bits.
- The decapsulation algorithm $\mathsf{Decap}(\mathrm{SK}, c)$ takes $c = (c_0, c_1)$ and the secret key $\mathrm{SK} = (b, \mathrm{SK}_b)$, and outputs the decryption $\mathsf{PKE.Dec}(\mathrm{SK}_b, c_b)$ of the ciphertext c_b using the key SK_b.

Input indistinguishability follows since, even given the secret key $\mathrm{SK} = (b, \mathrm{SK}_b)$, the attacker cannot distinguish if the ciphertext c_{1-b} encrypts the same bit k as contained in c_b or the opposite bit $1-k$. The *smoothness* property follows since the decapsulation of a random invalid ciphertext $c^* = (c_0, c_1)$ is uniformly random over the choice of the secret-key bit b.

Amplifying wHPS. The above construction only gives us a wHPS with 1-bit output. However, we can easily amplify the output size of a wHPS to any arbitrary polynomial $n = n(\lambda)$, simply by taking n independent copies of the scheme in parallel. Notice that in the new scheme, there will be at least 2^n possible secret keys consistent with any public key, and the output of the wHPS on an invalid ciphertext will consist of n random and independent bits. Since the *amount* of tolerated leakage ℓ is roughly equal to the wHPS output-size n, we can set it to be arbitrarily high.

We note that the concept of amplifying leakage-resilience directly via parallel repetition has been suggested and explored in several works [3,4,9,36,40], with surprising counter-examples showing that it is not secure in general. In our special case, we only argue that parallel repetition amplifies the *output size* of a wHPS (which is trivial), and then use our connection between output size and leakage resilience to indirectly argue that the latter amplifies as well.

The above construction can tolerate roughly n bits of leakage by storing n decryption keys, meaning that the *rate* of leakage is roughly $1/s(\lambda)$, where $s(\lambda)$ is the size of the decryption key in the underlying PKE scheme. In our final construction, we show how to increase this to any $O(\log(\lambda)/s(\lambda))$ leakage rate. Getting an even higher rate remains as an open problem.

Symmetric-Key wHPS. In the second part of our work, we carry the above ideas over to the symmetric-key setting. To do so, we first define a notion of a symmetric-key wHPS analogously to our public-key wHPS. We can think of symmetric-key wHPS as a special type of pseudorandom function (PRF) $f_k(\cdot)$ with the following properties (simplified):

- INPUT INDISTINGUISHABILITY: There are two special distributions on the inputs x which we call *valid* and *invalid*, and which are indistinguishable from uniform even given the secret-key k.
- SMOOTHNESS: Given multiple inputs/outptus $\{(x, f_k(x))\}$ for various random *valid* x, and a random choice of an *invalid* input x^*, the output $f_k(x^*)$ is uniformly random and independent (information theoretically), where the randomness comes from the choice of a consistent key k.

In other words, the key k maintains real entropy even conditioned on seeing $f_k(x)$ for many random valid inputs x, but this entropy comes out when evaluating $f_k(x^*)$ at a random invalid input x^*.

We show how to use such symmetric-key wHPS schemes to construct leakage-resilient symmetric-key encryption and weak PRFs. We then construct symmetric-key wHPS generically from standard weak PRF, and therefore only assuming that one-way functions exist. Our construction of MACs departs somewhat from this abstraction and requires additional ideas.

2 Preliminaries

Notation. We let λ denote the security parameter. For an integer n, we let $[n]$ denote the set $\{1, \ldots, n\}$. For a randomized function f, we write $f(x; r)$ to denote the unique output of f on input x with random coins r. We write $f(x)$ to denote a random variable for the output of $f(x; r)$ over the random coins r. For a distribution or random variable X, we write $x \leftarrow X$ to denote the operation of sampling a random x according to X. For a set S, we write $s \leftarrow S$ to denote sampling s *uniformly at random* from S. For distributions X, Y, we write $X \equiv Y$ to mean that X, Y are identically distributed, $X \approx_s Y$ to mean that they are statistically close, and $X \approx_c Y$ to say that they are computationally indistinguishable. We let $\mathrm{negl}(\lambda)$ denote the set of all negligible function $\mu(\lambda) = \lambda^{-\omega(1)}$. We use calligraphic letters such as $\mathcal{X}$ to denote an *ensemble* of sets $\mathcal{X} = \{\mathcal{X}_\lambda\}_{\lambda \in \mathbb{N}}$. To simplify notation, we often exclude the subscript λ when clear from context, and write e.g. $x \leftarrow \mathcal{X}$ to denote $x \leftarrow \mathcal{X}_\lambda$. We say that an ensemble $\mathcal{X}$ is *efficient* if the operations of sampling a uniformly random $x \leftarrow \mathcal{X}_\lambda$ and testing $x \in \mathcal{X}_\lambda$ can be performed in $\mathsf{poly}(\lambda)$ time.

The Leakage Oracle. We model *leakage attacks* on a secret key SK by giving the adversary access to a *leakage oracle*, which he can adaptively access to

learn information about the secret key. The leakage oracle, denoted $\mathcal{O}^{\ell}_{\mathrm{SK}}(\cdot)$, is parameterized by a secret key SK and a leakage parameter ℓ. Each query to the leakage oracle consists of a function $f_i : \{0,1\}^{|\mathrm{SK}|} \to \{0,1\}^{\ell_i}$ (represented by a circuit), to which the oracle answers with $f_i(\mathrm{SK})$.[3] The oracle keeps track of the output sizes ℓ_i of all the leakage queries so far, and only responds to the qth leakage query if $\sum_{i=1}^{q} \ell_i \leq \ell$.

3 Leakage-Resilient Public-Key Encryption

We begin with a definition of leakage-resilient public-key encryption (PKE). Our definition is equivalent to that used by prior works [2,43].

Definition 1 (Leakage-Resilient PKE). *An $\ell(\lambda)$-leakage-resilient PKE consists of the algorithms $(\mathsf{LR.Gen}, \mathsf{LR.Enc}, \mathsf{LR.Dec})$ and a message space $\mathcal{M}$ satisfying the following properties:*

Correctness: *For all $(\mathrm{PK}, \mathrm{SK})$ in the support of $\mathsf{LR.Gen}(1^{\lambda})$ and all messages $\mathrm{M} \in \mathcal{M}$, $\mathsf{LR.Dec}(\mathrm{SK}, \mathsf{LR.Enc}(\mathrm{PK}, \mathrm{M})) = \mathrm{M}$.*

Semantic Security with ℓ-Leakage: *For all* PPT *adversaries $\mathcal{A}$, the advantage of $\mathcal{A}$ in the following game is negligible in λ:*

- **Key Generation:** *The challenger runs $(\mathrm{PK}, \mathrm{SK}) \leftarrow \mathsf{LR.Gen}(1^{\lambda})$ and gives PK to $\mathcal{A}$.*
- **Leakage Queries:** *$\mathcal{A}$ is given access to the leakage oracle $\mathcal{O}^{\ell}_{\mathrm{SK}}(\cdot)$. Without loss of generality, we can assume that $\mathcal{A}$ queries $\mathcal{O}^{\ell}_{\mathrm{SK}}(\cdot)$ only once with a function f whose output is ℓ bits.*
- **Challenge:** *$\mathcal{A}$ chooses two plaintexts $\mathrm{M}_0, \mathrm{M}_1 \in \mathcal{M}$ and gives these to the challenger. The challenger chooses a random bit $b \leftarrow \{0,1\}$, and sends $c^* \leftarrow \mathsf{LR.Enc}(\mathrm{PK}, \mathrm{M}_b)$ to $\mathcal{A}$. The attacker $\mathcal{A}$ outputs a bit b'.*

We define the advantage of $\mathcal{A}$ as $Adv_{\mathcal{A}}(\lambda) = \left|\Pr[b' = b] - \frac{1}{2}\right|$.

If an encryption scheme is *0-leakage-resilient* we refer to it as *semantically secure*.

3.1 Leakage-Resilience from Weak Hash-Proof Systems

We specify our notion of weak hash-proof systems (wHPS). Our definition essentially follows an informal description given in [43] and a formal definition of [3], who considered a similar notion in the "identity based" setting.

Definition 2. *A* weak hash-proof system (wHPS) *with output space $\mathcal{K}$ consists of four algorithms $(\mathsf{Gen}, \mathsf{Encap}, \mathsf{Encap}^*, \mathsf{Decap})$ with the following syntax:*

- $(\mathrm{PK}, \mathrm{SK}) \leftarrow \mathsf{Gen}(1^{\lambda})$*: Given security parameter λ, creates a key pair.*
- $(c, k) \leftarrow \mathsf{Encap}(\mathrm{PK})$*: Given a public key PK, creates a "valid" ciphertext c encapsulating $k \in \mathcal{K}$.*
- $c^* \leftarrow \mathsf{Encap}^*(\mathrm{PK})$*: Given a public key PK, creates an "invalid" ciphertext c^*.*
- $k = \mathsf{Decap}(c, \mathrm{SK})$*: Given a ciphertext c and secret key SK, deterministically recovers $k \in \mathcal{K}$.*

[3] We insist on a circuit representation to ensure that a poly-time attacker can only query poly-sized circuits, meaning that the leakage is poly-time computable.

We require a weak hash-proof system to satisfy the following properties:

Correctness: *For all* $(\mathrm{PK}, \mathrm{SK})$ *in the range of* $\mathsf{Gen}(1^\lambda)$*, and for* $(c, k) \leftarrow \mathsf{Encap}(\mathrm{PK})$*, we have* $k = \mathsf{Decap}(c, \mathrm{SK})$*.*

Ciphertext Indistinguishability: *If we sample* $(\mathrm{PK}, \mathrm{SK}) \leftarrow \mathsf{Gen}(1^\lambda)$*,* $(c, k) \leftarrow \mathsf{Encap}(\mathrm{PK})$*,* $c^* \leftarrow \mathsf{Encap}^*(\mathrm{PK})$*, we have the computational indistinguishability:* $(\mathrm{PK}, \mathrm{SK}, c) \approx_c (\mathrm{PK}, \mathrm{SK}, c^*)$*. In other words, avalid ciphertext* c *created with* Encap *is indistinguishable from an invalid ciphertext* c^* *created with* Encap^**,* even given the secret key SK*.*

Smoothness: *If we sample* $(\mathrm{PK}, \mathrm{SK}) \leftarrow \mathsf{Gen}(1^\lambda)$*,* $c^* \leftarrow \mathsf{Encap}^*(\mathrm{PK})$*,* $k \leftarrow \mathcal{K}$*, and set* $k^* = \mathsf{Decap}(c^*, \mathrm{SK})$*, we have the distributional equivalence:* $(\mathrm{PK}, c^*, k^*) \equiv (\mathrm{PK}, c^*, k)$*. In other words, the decapsulated value* $k^* = \mathsf{Decap}(c^*, \mathrm{SK})$ *is uniformly random over* $\mathcal{K}$ *and independent of* c^* *and* PK*. Since all of the randomness of* k^* *must therefore come from the choice of* SK*, this implicitly requires that there are many possible choices of* SK *for a fixed* PK*.*

Constructing LR-PKE from wHPS. In the full version [31], we show how to construct leakage-resilient PKE from wHPS by following the construction of Naor and Segev [43], while formalizing that the the weaker security of wHPS is sufficient. We apply an extractor to the output k of the wHPS, and then use the extracted randomness as a one-time-pad to encrypt a message of our choice.

3.2 Constructing Weak Hash-Proof Systems from Any PKE

We present a weak hash proof system starting from any semantically secure PKE. We begin by constructing a wHPS with a very small output-space $\mathcal{K} = \mathbb{Z}_m$ for some polynomial $m = m(\lambda)$. In other words, the entropy of the output is only $\log(m) = O(\log(\lambda))$ bits. We will then amplify this via parallel repetition. The construction below generalizes the scheme we described in the introduction, which corresponds to the special case of $m = 2$ and the output is only 1 bit. By increasing m, we get an improvement in the *leakage rate* of our scheme.

Basic Construction. Let $m = m(\lambda)$ be some polynomial parameter and let $\Pi = (\mathsf{PKE.Gen}, \mathsf{PKE.Enc}, \mathsf{PKE.Dec})$ be a public-key encryption scheme with message-space $\mathcal{M} \supseteq \mathbb{Z}_m$.[4] We construct a wHPS with output space $\mathcal{K} = \mathbb{Z}_m$ as follows:

- $\mathsf{wHPS.Gen}(1^\lambda)$: Generate m key pairs: $\{(\mathrm{PK}_i, \mathrm{SK}_i) \leftarrow \mathsf{PKE.Gen}(1^\lambda)\}_{i \in [m]}$. Sample a random $t \leftarrow [m]$. Output $\mathrm{SK} = (t, \mathrm{SK}_t), \mathrm{PK} = (\mathrm{PK}_1, \ldots, \mathrm{PK}_m)$.
- $\mathsf{wHPS.Encap}(\mathrm{PK})$: Choose $k \leftarrow \mathbb{Z}_m$, and set $c := \{c_i \leftarrow \mathsf{PKE.Enc}(\mathrm{PK}_i, k)\}_{i \in [m]}$. Output (c, k).
- $\mathsf{wHPS.Encap}^*(\mathrm{PK})$: Choose $k \leftarrow \mathbb{Z}_m$. Output $c^* = \{c_i^* \leftarrow \mathsf{PKE.Enc}(\mathrm{PK}_i, k + i)\}_{i \in [m]}$, where the addition $k + i$ is performed in the group $\mathbb{Z}_m$.
- $\mathsf{wHPS.Decap}(\mathrm{SK}, c)$: Parse $\mathrm{SK} = (t, \mathrm{SK}_t)$ and $c = \{c_i\}$. Output $k = \mathsf{PKE.Dec}(\mathrm{SK}_t, c_t)$.

Theorem 1. *If* $(\mathsf{PKE.Gen}, \mathsf{PKE.Enc}, \mathsf{PKE.Dec})$ *is a semantically secure public-key encryption scheme, then the construction above is a weak hash-proof system with output space* $\mathcal{K} = \mathbb{Z}_m$*.*

[4] We can set $\mathcal{M} = \{0,1\}^{\lceil \log(m) \rceil}$ and naturally interpret it as containing $\mathbb{Z}_m$.

Output Amplification Via Parallel Repetition. The above construction gives us a public-key wHPS with a polynomial-sized output domain $\mathcal{K} = \mathbb{Z}_m$, so that the entropy of the output is only logarithmic. Unfortunately, we cannot use this scheme directly to get a meaningful LR-PKE, since we don't even have enough entropy to extract a single bit! However, it turns out to be very simple to increase the output-length of a wHPS just by taking several independent copies.

Theorem 2. *Let Π be any wHPS with output-domain $\mathcal{K}$. Let $n = n(\lambda)$ be a polynomial and Π^n be the n-wise parallel-repetition of Π. Then Π^n is a wHPS with output-domain $\mathcal{K}^n$.*

By taking our basic construction of wHPS with parameter m and applying n-wise parallel-repetition, we get a scheme with leakage resilience $\ell \approx n \cdot \log(m)$ and secret-key size $\approx n \cdot s$, meaning that we get a leakage-rate $\alpha \approx \log(m)/s$. By taking a sufficiently large n and m, the following theorem.

Theorem 3. *Assume the existence of semantically-secure PKE with secret-key size $s = s(\lambda)$. Then, for any arbitrarily large polynomial $\ell = \ell(\lambda)$ and any $\alpha = \alpha(\lambda) = O\left(\frac{\log \lambda}{s(\lambda)}\right)$ there exists an ℓ-leakage-resilient PKE where the* leakage rate *(ratio of ℓ to secret key size) is α.*

4 Leakage-Resilient wPRF and Symmetric-Key Encryption

Defining LR-wPRF. We begin with the definition of a *Leakage-Resilient weak PRF (wPRF).* Recall that the standard notion of a wPRF tells us that, given arbitrarily many *uniformly random* inputs $x_1, \ldots, x_q$, the outputs of the wPRF $y_1 = f_k(x_1), \ldots, y_q = f_k(x_q)$ look pseudo-random. This is in contrast with standard PRFs where the above holds for a *worst-case (adversarial)* choice of inputs $\{x_i\}$. Our definition of leakage-resilient wPRF requires that wPRF security holds even if the attacker can leak some information about the secret key. In particular, any future output of the wPRF on a fresh random input will still look random. Note that, since the attacker can always leak a few bits of $f_k(x)$ for some x of his choice, we cannot hope to achieve full PRF security in the presence of leakage, and hence settling for wPRF security is a natural choice.

Definition 3 (Leakage-Resilient weak PRF (LR-wPRF)). *Let $\mathcal{X}, \mathcal{Y}, \mathcal{K}$ be efficient ensembles and let $\mathcal{F} = \{ F_K : \mathcal{X} \to \mathcal{Y}\}_{K \in \mathcal{K}}$ be some efficient function family. We say that $\mathcal{F}$ is an $\ell(\lambda)$-*leakage-resilient weak PRF (LR-wPRF) *if, for all PPT attackers $\mathcal{A}$ the advantage of $\mathcal{A}$ is negligible in the following game:*

Initialization: *The challenger chooses a random $K \leftarrow \mathcal{K}_\lambda$.*

Learning Stage: *The attacker $\mathcal{A}^{\mathcal{O}^\ell_K(\cdot), F_K(\$)}(1^\lambda)$ gets access to the* leakage oracle $\mathcal{O}^\ell_K(\cdot)$ *(allowing him to learn up to ℓ bits of information about K) and also the* wPRF oracle $F_K(\$)$ *which does not take any input and, on each invocation, chooses a freshly random $X \leftarrow \mathcal{X}$ and outputs $(X, F_K(X))$.*[5]

[5] Without loss of generality, we can also assume that the attacker only makes a single call to the leakage oracle $\mathcal{O}^\ell_K(\cdot)$ after making all of its calls to the wPRF oracle $F_K(\$)$.

Challenge Stage: *The challenger chooses a challenge bit* $b \leftarrow \{0,1\}$ *and a random input* $X^* \leftarrow \mathcal{X}$. *If* $b = 0$, *it sets* $Y^* := F_K(X^*)$ *and if* $b = 1$ *it chooses* $Y^* \leftarrow \mathcal{Y}$. *The challenger gives* (X^*, Y^*) *to* $\mathcal{A}$ *who outputs a bit* b'. *We define the advantage of the attacker* $\mathcal{A}$ *as* $Adv_{\mathcal{A}}(\lambda) = \left|\Pr[b' = b] - \frac{1}{2}\right|$.

In the setting of no leakage we call a function $\mathcal{F}$ *satisfying the above definition a* standard wPRF.

From wPRF to CPA Encryption. In the full version [31] we show how to construct leakage-resilient CPA-secure (LR-CPA) symmetric-key encryption from LR-wPRF.

4.1 Leakage-Resilience Via Symmetric-Key wHPS

Toward the goal of constructing a LR-wPRF, we define a new notion of a *symmetric-key weak hash-proof system* (SwHPS), which can be thought of as a symmetric-key version of wHPS. In particular, we define a symmetric-key wHPS as a type of wPRF family $\mathcal{F} = \{F_K : \mathcal{X} \to \mathcal{Y}\}_{K \in \mathcal{K}}$ with some special properties.

Other than being able to choose inputs $X \leftarrow \mathcal{X}$ uniformly at random from their domain (which we refer to as the distribution Dist_0), we can also define two additional distributions Dist_1 (valid), and Dist_2 (invalid) over the input-domain $\mathcal{X}$. We require that samples from these various distributions are indistinguishable even when given the secret key K. Furthermore, conditioned on seeing many pairs $\{(X_i, F_K(X_i))\}$ for many different $X_i \leftarrow \mathsf{Dist}_1$ and a random choice of $X^* \leftarrow \mathsf{Dist}_2$, the output of $F_K(X^*)$ will be *truly* random and independent, where the randomness comes from the choice of a consistent secret key K.

Definition 4 (Symmetric-Key wHPS). *Let* $\mathcal{X}, \mathcal{Y}, \mathcal{K}$ *be some efficient ensembles and let* $\mathcal{F} = \{ F_K : \mathcal{X} \to \mathcal{Y}\}_{K \in \mathcal{K}}$ *be some efficient function family with the following PPT algorithms:*

- $\mathsf{samK} \leftarrow \mathsf{SamGen}(K)$ *takes an input* $K \in \mathcal{K}$, *outputs a* sampling key samK.
- $X \leftarrow \mathsf{Dist}_1(\mathsf{samK}), X \leftarrow \mathsf{Dist}_2(\mathsf{samK})$ *are two distributions that sample* $X \in \mathcal{X}$ *using the sampling key* samK. *For convenience, we also define the distribution* $X \leftarrow \mathsf{Dist}_0(\mathsf{samK})$ *which just samples a uniformly random* $X \leftarrow \mathcal{X}$ *and ignores the sampling key* samK.

We say that $\mathcal{F}$ *is a* symmetric-key wHPS (SwHPS) *if it satisfies the following two properties:*

Input Indistinguishability. *For any polynomial* $q = q(\lambda)$ *and any choice of* $(b_1, \ldots, b_q), (b'_1, \ldots, b'_q) \in \{0,1,2\}^q$, *the following distributions are computationally indistinguishable:* $(K, X_1, \ldots, X_q) \approx_c (K, X'_1, \ldots, X'_q)$, *where* $K \leftarrow \mathcal{K}_\lambda, \mathsf{samK} \leftarrow \mathsf{SamGen}(K), \{X_i \leftarrow \mathsf{Dist}_{b_i}(\mathsf{samK})\}, \{X'_i \leftarrow \mathsf{Dist}_{b'_i}(\mathsf{samK})\}$.

Smoothness. *For any polynomial* $q = q(\lambda)$ *the following distributions are equivalent:* $(X_1, \ldots, X_q, Y_1, \ldots, Y_q, X^*, Y^*) \equiv (X_1, \ldots, X_q, Y_1, \ldots, Y_q, X^*, U)$, *where* $K \leftarrow \mathcal{K}_\lambda, \mathsf{samK} \leftarrow \mathsf{SamGen}(K), \{X_i \leftarrow \mathsf{Dist}_1(\mathsf{samK}), Y_i := F_K(X_i)\}_{i \in [q]}$, $X^* \leftarrow \mathsf{Dist}_2(\mathsf{samK})$, $Y^* = F_K(X^*)$, *and* $U \leftarrow \mathcal{Y}$. *In other words,* Y^* *is uniformly random and independent of the other elements, where the randomness comes from the choice of a key* K.

Constructing LR-wPRF from SwHPS. We now construct a leakage-resilient wPRF from any symmetric-key wHPS. As in the public-key setting, we simply apply an extractor to the output of the symmetric-key wHPS.

Theorem 4. *Assume that $\mathcal{X}, \mathcal{Y}, \mathcal{S}, \mathcal{Z}$ are efficient ensembles such that $\mathcal{F} = \{ F_K : \mathcal{X} \to \mathcal{Y}\}_{K \in \mathcal{K}}$ is a symmetric-key wHPS and $\mathsf{Ext} : \mathcal{Y} \times \mathcal{S} \to \mathcal{Z}$ is a $(\log(|\mathcal{Y}|) - \ell(\lambda), \varepsilon(\lambda))$-extractor for some negligible $\varepsilon(\lambda)$. Define the function family $\mathcal{F}' = \{F'_K : (\mathcal{X} \times \mathcal{S}) \to \mathcal{Z}\}_{K \in \mathcal{K}}$ via $F'_K((X, S)) := \mathsf{Ext}(F_K(X); S)$. Then $\mathcal{F}'$ is an $\ell(\lambda)$-LR wPRF.*

4.2 Constructing Symmetric-Key wHPS

We now construct symmetric-key wHPS (SwHPS) from any weak PRF, and therefore also from the mere existence of one-way functions.

Basic Construction. Let $m = m(\lambda)$ be some polynomial and let $\mathcal{F}_{weak} = \{f_k : \mathcal{X} \to \mathbb{Z}_m\}_{k \in \mathcal{K}}$ be a standard (0-LR) wPRF family.[6] Let $(\mathsf{Enc}, \mathsf{Dec})$ be a standard symmetric-key encryption scheme constructed from $\mathcal{F}_{weak}$ as follows:

- $\mathsf{Enc}_k(\mathrm{M})$: Choose $x \leftarrow \mathcal{X}$ and output $c = (x, f_k(x) + \mathrm{M})$, where the addition is performed in $\mathbb{Z}_m$.
- $\mathsf{Dec}_k(c = (x, z))$: Output $\mathrm{M} := z - f_k(x)$.

Notice that this encryption scheme has message space $\mathcal{M} = \mathbb{Z}_m$, ciphertext space $\mathcal{C} = (\mathcal{X} \times \mathbb{Z}_m)$ and key-space $\mathcal{K}$. A useful property of this encryption scheme is that we can obliviously sample $c \leftarrow \mathcal{C}$ without knowing the key k, and this induces the same distribution as encrypting a random $\mathrm{M} \leftarrow \mathbb{Z}_m$. Given the wPRF $\mathcal{F}_{weak}$ and the resulting encryption scheme $(\mathsf{Enc}, \mathsf{Dec})$ as above, we define the symmetric-key wHPS system: $\mathcal{F}_{SwHPS} = \{F_K : \mathcal{C}^m \to \mathbb{Z}_m\}_{K \in ([m] \times \mathcal{K})}$ where

$$F_{(K=(t,k))}(X = (c_1, \ldots, c_m)) := \mathsf{Dec}_k(c_t).$$

Notice that we can efficiently sample random inputs $X \leftarrow \mathcal{C}^m$ without knowing the key K. We define the additional algorithms needed for the definition of SwHPS as follows:

- $\mathsf{samK} \leftarrow \mathsf{SamGen}(K)$. Parse $K = (t, k)$. Choose $m - 1$ values $\{k_i \leftarrow \mathcal{K} : i \in [m], i \neq t\}$ and define $k_t := k$. Set $\mathsf{samK} := (k_1, \ldots, k_m)$.
- $X \leftarrow \mathsf{Dist}_1(\mathsf{samK})$ (*Valid*). Choose $r \leftarrow \mathbb{Z}_m$ and $\{c_i \leftarrow \mathsf{Enc}_{k_i}(r)\}_{i \in [m]}$. Output $X = (c_1, \ldots, c_m)$.
- $X \leftarrow \mathsf{Dist}_2(\mathsf{samK})$ (*Invalid*). Choose $r \leftarrow \mathbb{Z}_m$ and $\{c_i \leftarrow \mathsf{Enc}_{k_i}(r + i)\}_{i \in [m]}$ where the addition is performed in $\mathbb{Z}_m$. Output $X = (c_1, \ldots, c_m)$.

For a *valid* X all of the ciphertexts c_i decrypt to the *same* value r, and for an *invalid* X they all decrypt to *different* values $r + i$. It is easy to see that the distributions $\mathsf{Dist}_1, \mathsf{Dist}_2$ are indistinguishable from uniform (Dist_0) even given $K = (t, k)$ since the ciphertext c_t always *is* uniform on its own, and we cannot distinguish the ciphertexts $c_i : i \neq t$ from uniform by the security of the

[6] If m is a power of 2, then we can just identify the elements of $\mathbb{Z}_m$ with those of $\{0,1\}^{\log(m)}$ in a natural way. Therefore, the existence of such wPRFs does not require any special assumptions.

wPRF. Furthermore, given many values $\{X_i, F_K(X_i)\}$ where X_i is *valid*, we learn nothing (information theoretically) about the secret index t contained in $K = (t, k)$. Therefore, for a random *invalid* $X^* \leftarrow \mathsf{Dist}_2(\mathsf{samK})$, the output $F_K(X^*) = \mathsf{Dec}_{k_t}(c_t) = r + t$ is truly random and independent.

Theorem 5. *Assuming $\mathcal{F}_{weak}$ is a standard wPRF, the function family $\mathcal{F}_{SwHPS}$ as defined above is a symmetric-key wHPS.*

Output Amplification Via Parallel Repetition. The above construction only obtains a polynomial-sized output domain, which means that the outputs only have $O(\log(\lambda))$ entropy. We amplify the output domain and entropy by using "parallel repetition". Formally,

Theorem 6. *Assume that $\mathcal{F} = \{f_k : \mathcal{X} \to \mathcal{Y}\}_{k\in\mathcal{K}}$ is a symmetric-key wHPS and let $n = n(\lambda)$ be an arbitrary polynomial. Define $\mathcal{F}^n = \{F_K : \mathcal{X}^n \to \mathcal{Y}^n\}_{K\in\mathcal{K}^n}$ via $F_{(k_1,\ldots,k_n)}(x_1, \ldots, x_n) \stackrel{\text{def}}{=} (f_{k_1}(x_1), \ldots, f_{k_n}(x_n))$. Then $\mathcal{F}^n$ is also a symmetric-key wHPS, whose output is amplified by a factor of n.*

We summarize our final results by combining all the developed ingredients.

Theorem 7. *Assuming the existence of one-way functions, there exist $\ell(\lambda)$-LR-wPRFs and $\ell(\lambda)$-LR-CPA symmetric-key encryption schemes for any polynomial $\ell(\lambda)$. Furthermore, assuming the existence of standard wPRFs with key-size $s(\lambda)$, the above schemes exist for any* leakage rate $\alpha(\lambda) = O\left(\log(\lambda)/s(\lambda)\right)$.

5 Leakage-Resilient Message Authentication

In this section, we construct leakage-resilient message-authentication codes (LR-MACs) from the minimal assumption that one-way functions exit.

Definition 5 (Leakage-Resilient MAC). *A MAC consists of the algorithms* $(\mathsf{Tag}, \mathsf{Ver})$ *and an efficient ensemble $\mathcal{K}$ of secret-keys. For* correctness*, we require that for every message* $\mathrm{M} \in \{0,1\}^*$*, and every key $K \in \mathcal{K}$, and every correctly generated tag $\sigma \leftarrow \mathsf{Tag}_K(\mathrm{M})$, we have $\mathsf{Ver}_K(\mathrm{M}, \sigma) = 1$. For* security*, we consider the following game between an attacker $\mathcal{A}$ and a challenger:*

Initialization: *The challenger chooses a random key $K \leftarrow \mathcal{K}_\lambda$.*

Learning Stage: *The attacker $\mathcal{A}^{\mathcal{O}^{\ell}_K(\cdot),\mathsf{Tag}_K(\cdot),\mathsf{Ver}_K(\cdot,\cdot)}$ can adaptively ask arbitrary* leakage*,* tagging *and* verification *queries to its oracles.*

Forgery: *The attacker provides a* forgery (M^*, σ^*) *and* wins *if M^* was never given as an input to the tagging oracle $\mathsf{Tag}_K(\cdot)$ during the leaning stage and $\mathsf{Ver}_K(\mathrm{M}^*, \sigma^*) = 1$.*

*We say that such a scheme is an $\ell(\lambda)$-*leakage-resilient message authentication code (ℓ-LR-MAC) *if, for all PPT attackers $\mathcal{A}$, the probability that $\mathcal{A}$ wins in the above game is negligible.*

In addition to the above definition, we also define a weaker notion of security against *"no-verification-query attacks" (nvq-MAC)*, where the attacker does not get access to the verification oracle $\mathsf{Ver}_K(\cdot,\cdot)$ during the learning stage. See the full version [31] for additional discussion. As a starting step, we instantiate a leakage-resilient nvq-MAC and then upgrade it to achieve full security.

Constructing nvq-MACs. Let $\mathcal{F}_{\mathsf{prf}} = \{f_k : \{0,1\}^* \to \mathcal{Y}\}_{k \in \mathcal{K}}$ be a *pseudorandom function (PRF) family* with super-polynomial output domain $|\mathcal{Y}_\lambda| = \lambda^{\omega(1)}$. Let $n = n(\lambda)$, $m = m(\lambda)$ be arbitrary polynomials. We construct a MAC with key-space $\mathcal{K}^{MAC} = (\mathcal{K} \times [m])^n$, by parsing $K \in \mathcal{K}^{MAC}$ as $K = ((k_1, t_1), \ldots, (k_n, t_n))$ where $k_i \in \mathcal{K}, t_i \in [m]$. We define the algorithms $(\mathsf{Tag}, \mathsf{Ver})$ as follows:

- $\mathsf{Tag}_K(\mathrm{M})$: Parse $K = ((k_1, t_1), \ldots, (k_n, t_n))$. Choose a random nonce $r \leftarrow \{0,1\}^\lambda$ and output the tag $\sigma = (r, \{\sigma_{i,j}\})$ where $\{\sigma_{i,j}\}$ is an $n \times m$ matrix defined by:
$$\sigma_{i,j} := \begin{cases} f_{k_i}(r||\mathrm{M}) & \text{if } j = t_i \\ y \leftarrow \mathcal{Y} & \text{otherwise} \end{cases}$$
That is, each row $i \in [n]$ of the matrix $\{\sigma_{i,j}\}$ contains one pseudorandom value under key k_i in the column t_i, and the rest of the row is truly random.
- $\mathsf{Ver}_K(\mathrm{M}, \sigma)$: Parse $\sigma = (r, \{\sigma_{i,j}\})$. For all $i \in [n]$, check that $f_{k_i}(r||\mathrm{M}) = \sigma_{i,t_i}$.

Security Intuition. We explain the security of the above construction via several abstract properties. Firstly, we can define an *alternate tagging oracle*, which is computationally indistinguishable from the original one even given the secret key $K = ((k_1, t_1), \ldots, (k_n, t_n))$ in full. The alternate oracle initially chooses an entire $n \times m$ matrix of PRF keys $\{k_{i,j}\}$, where the keys $k_{i,t_i} := k_i$ are taken from K and the rest of the keys $k_{i,j}$ for $j \neq t_i$ are chosen randomly. When answering tagging queries, the alternate tagging oracle sets *all* of the values $\sigma_{i,j} := f_{k_{i,j}}(r||\mathrm{M})$ to be pseudorandom under the appropriate keys. Once we define the alternate tagging oracle, we can also define two types of forgeries: valid and invalid. A forgery $\mathrm{M}^*, \sigma^* = (r^*, \{\sigma^*_{i,j}\})$ is *valid* if there is some pair (i,j) with $j \neq t_i$ such that $\sigma^*_{i,j} = f_{k_{i,j}}(r^*||\mathrm{M}^*)$, and is *invalid* otherwise. We have the following properties:

(1) Given access to the alternate tagging oracle and the key K in full, it is computationally hard to come up with a *valid* accepting forgery. *(Doing so requires guessing a PRF output at some fresh point $(r^*||\mathrm{M}^*)$, for some PRF key $k_{i,j}$ which doesn't appear in K.)*

(2) Given access to the alternate tagging oracle but not the key K, the information-theoretic probability of outputting an *invalid* accepting forgery is $< 2^{-n\log(m)}$. *(Doing so requires guessing the indices $T = (t_1, \ldots, t_n)$ since the only pairs (i,j) for which $\sigma^*_{i,j} = f_{k_{i,j}}(r^*||\mathrm{M}^*)$ are when $j = t_i$. But T has $n\log(m)$ bits of entropy and is independent of the above oracle.)*

The above properties ensure leakage-resilience for up to $\ell = n\log(m) - \omega(\log(\lambda))$ bits of leakage on the key K. Given such leakage, producing a valid forgery becomes no easier, since it is already hard given K in full. On the other hand, the probability of producing an invalid forgery can go up by a factor of at most 2^ℓ, which remains negligible. We formalize this in the following theorem.

Theorem 8. *If $\mathcal{F}_{\mathsf{prf}}$ is a PRF family with parameters as above, then the given construction is an $\ell(\lambda)$-leakage-resilient nvq-MAC for any $\ell(\lambda) = n(\lambda)\log(m(\lambda)) - \omega(\log(\lambda))$.*

In the full version [31], we show how to upgrade nvq-MAC security to full MAC security. As a consequence, we get the following theorem.

Theorem 9. *Assuming the existence of one-way functions, there exist $\ell(\lambda)$-leakage-resilient MACs for any polynomial $\ell(\lambda)$. Furthermore, assuming the existence of PRFs with variable-length input-size, output size λ, and key-size $s(\lambda)$, such MACs exist for any* leakage rate $\alpha(\lambda) = O\left(\log(\lambda)/s(\lambda)\right)$.

6 Conclusions

We saw how to construct several leakage-resilient primitives under the minimal assumption that they exists in the standard setting without any leakage. Perhaps the main open question is to improve the *leakage rate* of such constructions (say, to some constant fraction of the secret key), or to provide black-box separations showing that this is not be possible. Another interesting open question is to construct leakage-resilient *signatures* under the minimal assumption that one-way functions exist. Lastly, it would be interesting to come up with other applications where *weak* hash-proof systems (wHPS) can replace standard HPS.

References

1. Agrawal, D., Archambeault, B., Rao, J.R., Rohatgi, P.: The EM side-channel(s). In: Kaliski Jr., B.S., Koç, Ç.K., Paar, C. (eds.) CHES 2002. LNCS, vol. 2523, pp. 29–45. Springer, Heidelberg (2003)
2. Akavia, A., Goldwasser, S., Vaikuntanathan, V.: Simultaneous hardcore bits and cryptography against memory attacks. In: Reingold, O. (ed.) TCC 2009. LNCS, vol. 5444, pp. 474–495. Springer, Heidelberg (2009)
3. Alwen, J., Dodis, Y., Naor, M., Segev, G., Walfish, S., Wichs, D.: Public-key encryption in the bounded-retrieval model. In: Gilbert, H. (ed.) EUROCRYPT 2010. LNCS, vol. 6110, pp. 113–134. Springer, Heidelberg (2010)
4. Alwen, J., Dodis, Y., Wichs, D.: Leakage-resilient public-key cryptography in the bounded-retrieval model. In: Halevi [28], pp. 36–54
5. Bar-El, H.: Known attacks against smartcards (2003), `http://www.hbarel.com/publications/Known_Attacks_Against_Smartcards.pdf` (last accessed: August 26, 2009)
6. Bitansky, N., Canetti, R., Halevi, S.: Leakage-tolerant interactive protocols. In: Cramer [14], pp. 266–284
7. Boyle, E., Segev, G., Wichs, D.: Fully leakage-resilient signatures. In: Paterson, K.G. (ed.) EUROCRYPT 2011. LNCS, vol. 6632, pp. 89–108. Springer, Heidelberg (2011)
8. Brakerski, Z., Goldwasser, S.: Circular and leakage resilient public-key encryption under subgroup indistinguishability (or: Quadratic residuosity strikes back). In: Rabin, T. (ed.) CRYPTO 2010. LNCS, vol. 6223, pp. 1–20. Springer, Heidelberg (2010)
9. Brakerski, Z., Kalai, Y.T.: A parallel repetition theorem for leakage resilience. In: Cramer [14], pp. 248–265
10. Brakerski, Z., Katz, J., Kalai, Y., Vaikuntanathan, V.: Overcomeing the hole in the bucket: Public-key cryptography against resilient to continual memory leakage. In: FOCS [32], pp. 501–510
11. Braverman, M., Hassidim, A., Kalai, Y.T.: Leaky pseudo-entropy functions. In: Chazelle, B. (ed.) ICS, pp. 353–366. Tsinghua University Press (2011)

12. Cash, D., Ding, Y.Z., Dodis, Y., Lee, W., Lipton, R.J., Walfish, S.: Intrusion-resilient key exchange in the bounded retrieval model. In: Vadhan, S.P. (ed.) TCC 2007. LNCS, vol. 4392, pp. 479–498. Springer, Heidelberg (2007)
13. Chow, S.S.M., Dodis, Y., Rouselakis, Y., Waters, B.: Practical leakage-resilient identity-based encryption from simple assumptions. In: Al-Shaer, E., Keromytis, A.D., Shmatikov, V. (eds.) ACM Conference on Computer and Communications Security, pp. 152–161. ACM (2010)
14. Cramer, R. (ed.): TCC 2012. LNCS, vol. 7194. Springer, Heidelberg (2012)
15. Cramer, R., Shoup, V.: Universal hash proofs and a paradigm for adaptive chosen ciphertext secure public-key encryption. In: Knudsen, L.R. (ed.) EUROCRYPT 2002. LNCS, vol. 2332, pp. 45–64. Springer, Heidelberg (2002)
16. Crescenzo, G.D., Lipton, R.J., Walfish, S.: Perfectly secure password protocols in the bounded retrieval model. In: Halevi and Rabin [30], pp. 225–244
17. Dodis, Y., Haralambiev, K., López-Alt, A., Wichs, D.: Cryptography against continuous memory attacks. In: FOCS [32], pp. 511–520
18. Dodis, Y., Haralambiev, K., López-Alt, A., Wichs, D.: Efficient public-key cryptography in the presence of key leakage. In: Abe, M. (ed.) ASIACRYPT 2010. LNCS, vol. 6477, pp. 613–631. Springer, Heidelberg (2010)
19. Dodis, Y., Lewko, A.B., Waters, B., Wichs, D.: Storing secrets on continually leaky devices. In: Ostrovsky, R. (ed.) FOCS, pp. 688–697. IEEE (2011)
20. Dodis, Y., Yu, Y.: Overcoming weak expectations. In: ITW (2012), http://www.cs.nyu.edu/~dodis/ps/weak-expe.pdf
21. Dziembowski, S.: Intrusion-resilience via the bounded-storage model. In: Halevi, S., Rabin, T. (eds.) TCC 2006. LNCS, vol. 3876, pp. 207–224. Springer, Heidelberg (2006)
22. Dziembowski, S.: Intrusion-resilience via the bounded-storage model. In: Halevi and Rabin [30], pp. 207–224
23. Dziembowski, S., Pietrzak, K.: Leakage-resilient cryptography. In: 49th Symposium on Foundations of Computer Science, October 25-28, pp. 293–302. IEEE Computer Society, Philadelphia (2008)
24. ECRYPT: Side channel cryptanalysis lounge, http://www.emsec.rub.de/research/projects/sclounge/ (last accessed: May 1, 2011)
25. Goldwasser, S., Kalai, Y.T., Peikert, C., Vaikuntanathan, V.: Robustness of the learning with errors assumption. In: Yao, A.C.C. (ed.) ICS, pp. 230–240. Tsinghua University Press (2010)
26. Goldwasser, S., Rothblum, G.N.: How to compute in the presence of leakage. In: Electronic Colloquium on Computational Complexity (ECCC), vol. 19, p. 10 (2012)
27. Halderman, J.A., Schoen, S.D., Heninger, N., Clarkson, W., Paul, W., Calandrino, J.A., Feldman, A.J., Appelbaum, J., Felten, E.W.: Lest we remember: cold-boot attacks on encryption keys. Commun. ACM 52(5), 91–98 (2009)
28. Halevi, S. (ed.): CRYPTO 2009. LNCS, vol. 5677. Springer, Heidelberg (2009)
29. Halevi, S., Lin, H.: After-the-fact leakage in public-key encryption. In: Ishai [33], pp. 107–124
30. Halevi, S., Rabin, T. (eds.): TCC 2006. LNCS, vol. 3876. Springer, Heidelberg (2006)
31. Hazay, C., López-Alt, A., Wee, H., Wichs, D.: Leakage-resilient cryptography from minimal assumptions. Cryptology ePrint Archive, Report 2012/604 (2012), http://eprint.iacr.org/
32. IEEE: 51th Symposium on Foundations of Computer Science, October 23-26 (2010)

33. Ishai, Y. (ed.): TCC 2011. LNCS, vol. 6597. Springer, Heidelberg (2011)
34. Ishai, Y., Sahai, A., Wagner, D.: Private circuits: Securing hardware against probing attacks. In: Boneh, D. (ed.) CRYPTO 2003. LNCS, vol. 2729, pp. 463–481. Springer, Heidelberg (2003)
35. Garg, S., Jain, A., Sahai, A.: Leakage-resilient zero knowledge. In: Rogaway, P. (ed.) CRYPTO 2011. LNCS, vol. 6841, pp. 297–315. Springer, Heidelberg (2011)
36. Jain, A., Pietrzak, K.: Parallel repetition for leakage resilience amplification revisited. In: Ishai [33], pp. 58–69
37. Katz, J., Vaikuntanathan, V.: Signature schemes with bounded leakage resilience. In: Matsui, M. (ed.) ASIACRYPT 2009. LNCS, vol. 5912, pp. 703–720. Springer, Heidelberg (2009)
38. Kocher, P.C.: Timing attacks on implementations of Diffie-Hellman, RSA, DSS, and other systems. In: Koblitz, N. (ed.) CRYPTO 1996. LNCS, vol. 1109, pp. 104–113. Springer, Heidelberg (1996)
39. Kocher, P., Jaffe, J., Jun, B.: Differential power analysis. In: Wiener, M. (ed.) CRYPTO 1999. LNCS, vol. 1666, pp. 388–397. Springer, Heidelberg (1999)
40. Lewko, A., Waters, B.: On the insecurity of parallel repetition for leakage resilience. In: FOCS [32], pp. 521–530
41. Lewko, A.B., Lewko, M., Waters, B.: How to leak on key updates. In: STOC (2011) (to appear)
42. Micali, S., Reyzin, L.: Physically observable cryptography (extended abstract). In: Naor, M. (ed.) TCC 2004. LNCS, vol. 2951, pp. 278–296. Springer, Heidelberg (2004)
43. Naor, M., Segev, G.: Public-key cryptosystems resilient to key leakage. In: Halevi [28], pp. 18–35
44. Naor, M., Segev, G.: Public-key cryptosystems resilient to key leakage. SIAM Journal on Computing 41(4), 772–814 (2012); a preliminary version appeared in Halevi, S. (ed.) CRYPTO 2009. LNCS, vol. 5677, pp. 18–35. Springer, Heidelberg (2009)
45. Pietrzak, K.: A leakage-resilient mode of operation. In: Joux, A. (ed.) EUROCRYPT 2009. LNCS, vol. 5479, pp. 462–482. Springer, Heidelberg (2009)
46. Quisquater, J.-J., Samyde, D.: ElectroMagnetic analysis (EMA): Measures and counter-measures for smart cards. In: Attali, S., Jensen, T. (eds.) E-smart 2001. LNCS, vol. 2140, pp. 200–210. Springer, Heidelberg (2001)
47. Quisquater, J.J., Koene, F.: Side channel attacks: State of the art (October 2002), `http://www.ipa.go.jp/security/enc/CRYPTREC/fy15/doc/1047_Side_Channel_report.pdf` (last accessed: August 26, 2009)
48. Reliable Computing Laboratory, Boston University: Side channel attacks database, `http://www.sidechannelattacks.com` (last accessed: August 26, 2009)
49. Standaert, F.X.: How leaky is an extractor? In: Abdalla, M., Barreto, P.S.L.M. (eds.) LATINCRYPT 2010. LNCS, vol. 6212, pp. 294–304. Springer, Heidelberg (2010)