J Cryptol (2016) 29:514–551
DOI: 10.1007/s00145-015-9200-x

Journal of CRYPTOLOGY

Leakage-Resilient Cryptography from Minimal Assumptions

Carmit Hazay
Bar-Ilan University, Ramat-Gan, Israel
carmit.hazay@gmail.com; carmit.hazay@biu.ac.il

Adriana López-Alt
New York University, New York, NY, USA
adrilopo@gmail.com

Hoeteck Wee*
CNRS and ENS, Paris, France
wee@di.ens.fr

Daniel Wichs†
Northeastern University, Boston, MA, USA
wichs@ccs.neu.edu

Communicated by Eike Kiltz.

Received 12 June 2013
Online publication 6 March 2015

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

* Research conducted while at George Washington University, supported by NSF CAREER Award CNS-1237429.

† Research conducted while at IBM Research, T. J. Watson.

poor in comparison with 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 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 theoretic 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., [1,6,26,36,37,46–48] 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 [32]), where memory contents, including secret-key information, are not 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 [3,12,15,24]. 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 [2–5,7–9,11–13,17,25,28,31,34,38,42]. 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" $f(\mathsf{sk})$ 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 [10,16,19,22,29,33,39,41,45] 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 many cryptographic primitives, including all of the ones discussed in this work, are generically resilient against small amounts of leakage. In particular, every instantiation of such primitives 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 be of too much 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. In this work, whenever we talk about leakage-resilient schemes, we refer to schemes of this type that can tolerate larger amounts of leakage beyond the generic bound. 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 [4,7,42], 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 [3,17,38] 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-key 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:

[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 [23] for a general treatment of this question.

- 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 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 computational Diffie-Hellman (CDH).

Leakage Amount Versus 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* [16,42], where we restrict the amount of entropy loss caused by the leakage rather than restricting its length. We also show that our public/symmetric-key encryption schemes provide resilience to *"after-the-fact"* leakage as defined by Halevi and Lin [31]. In particular, if the attacker can choose to learn some arbitrary ℓ_{post} bits of leakage on the secret-key adaptively *after* seeing 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* [3,12,15,24], 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.

1.1. *Overview of Our Techniques*

Our starting point is a result of Naor and Segev [42] (journal version [43]), which constructs leakage-resilient public-key encryption from any *hash-proof system* (HPS) [14]. As observed in [2,42], 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 that, surprisingly, wHPS can be constructed

[2] This weaker variant of HPS was discussed implicitly but not defined formally in [42]. The work of [2] explicitly defined a notion of "identity-based HPS" which corresponds to an extension of our notion of wHPS

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 pseudorandom 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 now briefly describe what wHPS is, 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) of PK and c^*. 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.

Footnote 2 continued
to the identity-based setting. In both works, the distinction between the "weak" and "full" notions of HPS was not considered important beyond simplifying exposition, and all of the given instantiations in these works even achieve the "full" notion. In other words, although these works notice that weak HPS is sufficient, they do not get any extra benefits from this observation.

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 [42].

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.

The *input indistinguishability* property of the above construction 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 [2,3,9,35,40], 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 message-authentication codes departs somewhat from this abstraction and requires additional ideas.

1.2. *Organization*

In Sect. 2, we describe our notation and define the concept of a *leakage oracle*, which we use to formalize leakage attacks. We also state several useful lemmas on entropy and extractors. In Sect. 3, we give our results for leakage-resilient public-key encryption via the intermediate abstraction of a weak hash-proof system (wHPS). In Sect. 4, we give our results for leakage-resilient symmetric-key encryption via a symmetric-key wHPS. In Sect. 5, we turn to the construction of leakage-resilient message-authentication codes. Lastly, in Sect. 6, we present extensions of our results to entropy-bounded leakage, after-the-fact leakage, and the bounded-retrieval model.

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 let $\mathbf{SD}(X, Y)$ denote their statistical distance. 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 $\mathcal{O}^{\ell}_{\mathsf{SK}}(\cdot)$, is parameterized by a secret key SK and a leakage parameter ℓ. Each query to the leakage oracle consists of a function $f_i : \{0,1\}^{|\mathsf{SK}|} \to \{0,1\}^{\ell_i}$ (represented by a circuit), to which the oracle answers with $f_i(\mathsf{SK})$.[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 $\sum_{i=1}^{q} \ell_i \leq \ell$. In other words, the total number of bits output by the oracle is bounded by ℓ.

2.1. *Entropy and Extractors*

Definition 2.1. (*Min-Entropy*) The *min-entropy* of a random variable X, denoted as $\mathbf{H}_\infty(X)$ is defined as $\mathbf{H}_\infty(X) \stackrel{\mathsf{def}}{=} -\log(\max_x \Pr[X = x])$.

Definition 2.2. (*Average-Conditional Min-Entropy* [21]) The *average-conditional min-entropy* of a random variable X conditioned on a correlated variable Z, denoted as $\mathbf{H}_\infty(X \mid Z)$ is defined as

$$\begin{aligned}\mathbf{H}_\infty(X \mid Z) &\stackrel{\mathsf{def}}{=} -\log\left(\mathbb{E}_{z\leftarrow Z}\left[\max_x \Pr[X = x|Z = z]\right]\right)\\ &= -\log\left(\mathbb{E}_{z\leftarrow Z}\left[2^{\mathbf{H}_\infty[X|Z=z]}\right]\right).\end{aligned}$$

This notion of conditional min-entropy measures the best guess for X by an adversary that may observe an *average-case* correlated variable Z. That is, for all (inefficient) functions $\mathcal{A}$, we have $\Pr[\mathcal{A}(Z) = X] \leq 2^{-\mathbf{H}_\infty(X|Z)}$, and there is some $\mathcal{A}$ achieving equality.

Lemma 2.3. [21] *Let X, Y, Z be arbitrarily correlated random variables where the support of Y has at most 2^ℓ elements. Then $\mathbf{H}_\infty(X|(Y, Z)) \geq \mathbf{H}_\infty(X|Z) - \ell$. In particular, $\mathbf{H}_\infty(X|Y) \geq \mathbf{H}_\infty(X) - \ell$.*

[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 give the following definition of randomness extractors [44], which is somewhat stronger than the usual one and is also called an *average-case strong extractor* [21].

Definition 2.4. (*Randomness Extractor*) An efficient function $\mathsf{Ext} : \mathcal{X} \times \mathcal{S} \to \mathcal{Y}$ is a (v, ε)-*extractor* if for all (correlated) random variables X, Z such that the support of X is $\mathcal{X}$ and $\mathbf{H}_\infty(X \mid Z) \geq v$, we have $\mathbf{SD}(\ (Z, S, \mathsf{Ext}(X; S))\ ,\ (Z, S, Y)\ ) \leq \varepsilon$, where S (also called the *seed*) and Y are distributed uniformly and independently over their domains $\mathcal{S}, \mathcal{Y}$ respectively.

Theorem 2.5. [21,44] *Let* $\mathcal{H} = \{\ h_s : \mathcal{X} \to \mathcal{Y}\ \}_{s \in \mathcal{S}}$ *be a* universal family of hash functions *meaning that for all* $x \neq x' \in \mathcal{X}$ *we have* $\Pr_{s \leftarrow \mathcal{S}}[h_s(x) = h_s(x')] \leq \frac{1}{|\mathcal{Y}|}$. *Then* $\mathsf{Ext}(x; s) \stackrel{\text{def}}{=} h_s(x)$, *is a* (v, ε)-*extractor for any parameter* $v \geq \log|\mathcal{Y}| + 2\log(1/\varepsilon)$.

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 [4,42].

Definition 3.1. (*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 (PK, SK) in the support of $\mathsf{LR.Gen}(1^\lambda)$ and all messages $\text{M} \in \mathcal{M}$, $\mathsf{LR.Dec}(\text{SK}, \mathsf{LR.Enc}(\text{PK}, \text{M})) = \text{M}$.
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 $(\text{PK}, \text{SK}) \leftarrow \mathsf{LR.Gen}(1^\lambda)$ and gives PK to $\mathcal{A}$.
Leakage Queries:	$\mathcal{A}$ is given access to the leakage oracle $\mathcal{O}^\ell_{\text{SK}}(\cdot)$. Without loss of generality, we can assume that $\mathcal{A}$ queries $\mathcal{O}^\ell_{\text{SK}}(\cdot)$ only once with a function f whose output is ℓ bits.
Challenge:	$\mathcal{A}$ chooses two plaintexts $\text{M}_0, \text{M}_1 \in \mathcal{M}$ and gives these to the challenger. The challenger chooses a random bit $b \leftarrow \{0, 1\}$, and sends $c^* \leftarrow \mathsf{LR.Enc}(\text{PK}, \text{M}_b)$ to $\mathcal{A}$. The attacker $\mathcal{A}$ outputs a bit b'.

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

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

Remarks. Notice that the attacker is only given access to the leakage oracle prior to receiving the challenge ciphertext. This is a necessary restriction as otherwise, he could leak (e.g.,) the first bit of the plaintext and easily win the distinguishing game. See the extensions in Sect. 6.3 for a meaningful definition of "after-the-fact" leakage, which can occur after observing the challenge ciphertext. Our default definition also does not allow

leakage on the randomness used by the key-generation algorithm, but we will give some positive results for this variant later on.

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 [42] and a formal definition of [2], who considered a similar notion in the "identity-based" setting. As described in the introduction, we will think of a wHPS as a special type of key encapsulation mechanism with additional properties.

Definition 3.2. 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 public/secret-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 $k \in \mathcal{K}$.

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

Correctness: For all $(\mathrm{PK}, \mathrm{SK})$ in the range of $\mathsf{Gen}(1^\lambda)$,

$$\Pr\left[k = k' \;\middle|\; \begin{array}{c}(c, k) \leftarrow \mathsf{Encap}(\mathrm{PK}) \\ k' = \mathsf{Decap}(c, \mathrm{SK})\end{array}\right] = 1.$$

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, a valid 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. We now describe the construction of leakage-resilient PKE from hash-proof systems due to Naor and Segev [42]. We essentially follow the construction and proof from that work, while formalizing that the weaker security of wHPS is sufficient.

Let $(\mathsf{wHPS.Gen}, \mathsf{wHPS.Encap}, \mathsf{wHPS.Encap}^*, \mathsf{wHPS.Decap})$ be a wHPS with output set $\mathcal{K}$, and let $\mathsf{Ext} : \mathcal{K} \times \mathcal{S} \to \mathcal{M}$ be a $(\log|\mathcal{K}| - \ell, \varepsilon)$-extractor, where $\mathcal{K}$, $\mathcal{S}$, $\mathcal{M}$ are efficient ensembles, $\ell = \ell(\lambda)$ is some parameter and $\varepsilon = \varepsilon(\lambda) = \mathrm{negl}(\lambda)$ is negligible. Further, assume that $\mathcal{M}$ is an additive group (e.g., bit strings under XOR). We define an encryption scheme with message space $\mathcal{M}$ as follows:

- $\mathsf{LR.Gen}(1^\lambda)$: Output $(\mathrm{PK}, \mathrm{SK}) \leftarrow \mathsf{wHPS.Gen}(1^\lambda)$.
- $\mathsf{LR.Enc}(\mathrm{PK}, \mathrm{M})$: Sample a seed $s \leftarrow \mathcal{S}$ and output $c = (s, c_0, c_1)$, where:

$$(c_0, k) \leftarrow \mathsf{wHPS.Encap}(\mathrm{PK}), \quad c_1 = \mathrm{M} + \mathsf{Ext}(k; s)$$

- $\mathsf{LR.Dec}(\mathrm{SK}, c)$: Parse $c = (s, c_0, c_1)$ and set $k := \mathsf{wHPS.Decap}(\mathrm{SK}, c_0)$. Output $\mathrm{M} := c_1 - \mathsf{Ext}(k; s)$.

Theorem 3.3. *The encryption scheme* $(\mathsf{LR.Gen}, \mathsf{LR.Enc}, \mathsf{LR.Dec})$ *described above is an* $\ell(\lambda)$*-leakage-resilient public-key encryption.*

Proof. Correctness of the encryption scheme follows directly from the correctness of the weak hash-proof system. We argue security using a series of games argument.

Game 0: This is the security game defined in Definition 3.1. In this game, the adversary's view consists of $(\mathrm{PK}, f(\mathrm{SK}), s, c_0, c_1)$, where $(\mathrm{PK}, \mathrm{SK}) \leftarrow \mathsf{wHPS.Gen}(1^\lambda)$, $s \leftarrow \mathcal{S}$ and:

$$(c_0, k) \leftarrow \mathsf{wHPS.Encap}(\mathrm{PK}), c_1 \leftarrow \mathrm{M}_b + \mathsf{Ext}(k; s).$$

The leakage function f is chosen by the attacker adaptively based on PK (recall that, w.l.o.g., the attacker chooses a single leakage function with ℓ bit output).

Game 1: In this game, we change how c_1 is computed. The challenger now computes:

$$(c_0, k_0) \leftarrow \mathsf{wHPS.Encap}(\mathrm{PK}), \quad k_1 \leftarrow \mathsf{wHPS.Decap}(c_0, \mathrm{SK}),$$
$$c_1 \leftarrow \mathrm{M}_b + \mathsf{Ext}(k_1; s)$$

The only difference between Game 0 and Game 1 is the use of k_0 versus k_1. However, by the *correctness* of the wHPS, we know that $k_0 = k_1$. Therefore, Games 0 and 1 are identical.

Game 2: In this game, we change how c_0 is computed. Instead of letting c_0 be a valid ciphertext computed using $\mathsf{wHPS.Encap}$, we now let it be an invalid ciphertext computed with $\mathsf{wHPS.Encap}^*$:

$$c_0 \leftarrow \mathsf{wHPS.Encap}^*(\mathrm{PK}), \quad k \leftarrow \mathsf{wHPS.Decap}(c_0, \mathrm{SK}),$$
$$c_1 \leftarrow \mathrm{M}_b + \mathsf{Ext}(k; s)$$

We claim that Games 1 and 2 are indistinguishable by the *ciphertext indistinguishability* property of the weak hash-proof system. Indeed, we know that

valid and invalid ciphertexts are indistinguishable even given the *entire* secret key SK, and therefore certainly given only the output of the leakage query $f(\mathrm{SK})$.

Game 3: Finally, we change how c_1 is computed.

$$c_0 \leftarrow \mathsf{wHPS.Encap}^*(\mathrm{PK}), \quad R \leftarrow \mathcal{M},$$
$$c_1 \leftarrow \mathrm{M}_b + R$$

Let $\mathrm{PK}, c_0, k = \mathsf{wHPS.Decap}(c_0, \mathrm{SK}), f(\mathrm{SK})$ be (correlated) random variables distributed as in Game 2 (since the function f is chosen adaptively depending on PK, we can think of it as a correlated random variable as well). By *smoothness*, we know that k is uniform over $\mathcal{K}$ even given PK and c_0: that is, $\mathbf{H}_\infty(k \mid \mathrm{PK}, c_0) = \log(|\mathcal{K}|)$. By Lemma 2.3, since the domain of $f(\mathrm{SK})$ is $\{0,1\}^\ell$, we know

$$\mathbf{H}_\infty(k \mid \mathrm{PK}, c_0, f(\mathrm{SK})) \geq \log(|\mathcal{K}|) - \ell.$$

(This holds even if f is adaptively chosen after seeing PK.) Since Ext is an (average-case, strong) $(\log(|\mathcal{K}|) - \ell, \varepsilon)$-extractor for $\varepsilon = \mathrm{negl}(\lambda)$, we conclude that $\mathsf{Ext}(k; s)$ is ε-close to a uniformly random R, even given PK, c_0, $f(\mathrm{SK})$. Thus Games 2 and 3 are statistically close.

Observe that the view of $\mathcal{A}$ in Game 3 is *independent* of both M_b and the challenge bit b. Therefore, the advantage of $\mathcal{A}$ in Game 3 is 0. We can thus conclude that the advantage of $\mathcal{A}$ in Game 0 is negligible in λ. □

3.2. *Constructing weak Hash-Proof Systems from any PKE*

In this section, we present a weak hash proof system starting from any semantically secure public-key encryption scheme. 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, where we take several independent copies of this scheme to get a larger output. The construction below generalizes the simple 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).

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

- $\mathsf{wHPS.Encap}^*(\mathrm{PK})$: 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 $c = \{c_i\}_{i\in[m]}$. Output $k = \mathsf{PKE.Dec}(\mathrm{SK}_t, c_t)$.

Theorem 3.4. *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* $\mathcal{K} = \mathbb{Z}_m$.

Proof. It is clear that the construction satisfies the *correctness* property of wHPS.

For the *ciphertext indistinguishability* property of wHPS, we need to prove that

$$\begin{aligned}&(\mathrm{PK}, \mathrm{SK}, c = \{c_i \leftarrow \mathsf{PKE.Enc}(\mathrm{PK}_i, k)\}_{i\in[m]})\\ &\quad\approx_c (\mathrm{PK}, \mathrm{SK}, c^* = \{c_i^* \leftarrow \mathsf{PKE.Enc}(\mathrm{PK}_i, k + i)\}_{i\in[m]})\end{aligned}$$

where $k \leftarrow \mathbb{Z}_m$. Firstly, since for a fixed t and random $k \leftarrow \mathbb{Z}_m$ the distribution of k and $k + t$ are equivalent, we can rewrite the left-hand side as $\{c_i \leftarrow \mathsf{PKE.Enc}(\mathrm{PK}_i, k + t)\}$. Notice that c_t and c_t^* are both identically distributed (even conditioned on PK, SK) and encrypt a random value $k + t$. Therefore, the difference in the above distributions lies in the ciphertexts $\{c_i\}_{i\neq t}$ and $\{c_i^*\}_{i\neq t}$ where the left-hand-ones encrypt $k + t$ and the right-hand-ones encrypt $k + i$. But, by the semantic security of the underlying PKE, this is computationally indistinguishable even given $\mathrm{PK} = (\mathrm{PK}_1, \ldots, \mathrm{PK}_m)$, $\mathrm{SK} = (t, \mathrm{SK}_t)$ and $c_t = c_t^*$ (which together determine k). Formally, we proceed via $m - 1$ hybrid arguments where in each hybrid game, we change $c_i \leftarrow \mathsf{PKE.Enc}(\mathrm{PK}_i, k + t)$ to $c_i^* \leftarrow \mathsf{PKE.Enc}(\mathrm{PK}_i, k + i)$ for $i \in [m] \setminus \{t\}$. This implies that in the ith hybrid game the first i ciphertexts are sampled according to $\mathsf{wHPS.Encap}^*(\cdot)$, whereas the remaining ciphertexts are sampled according to $\mathsf{wHPS.Encap}(\cdot)$. The reduction follows easily by obtaining the ith ciphertext via an encryption oracle while computing the rest of the ciphertexts using the public-keys within PK, which are independent of PK_i.

For the *smoothness* property of wHPS, we notice that given PK and an invalid ciphertext $c^* = \{c_i^* \leftarrow \mathsf{PKE.Enc}(\mathrm{PK}_i, k+i)\}$, the decapsulated value $k^* = \mathsf{wHPS.Decap}(\mathrm{SK}, c^*) = \mathsf{PKE.Dec}(\mathrm{SK}_t, c_t^*) = k + t$ is uniformly random in $\mathbb{Z}_m$ over the choice of index t contained in the secret-key, and therefore independent of PK, c^*. □

Output Amplification via Parallel Repetition. The above construction gives us a public-key wHPS with a polynomial-sized output domain $\mathcal{K} = \mathbb{Z}_m$, meaning that the entropy of the output is only logarithmic. Unfortunately, we cannot use this scheme directly with Theorem 3.3 to get a meaningful leakage-resilient PKE, since we do not 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. In particular, let $\Pi = (\mathsf{Gen}, \mathsf{Encap}, \mathsf{Encap}^*, \mathsf{Decap})$ be any wHPS with output domain $\mathcal{K}$. For any integer n, we can define the n-wise *parallel repetition scheme* $\Pi^n = (\mathsf{Gen}_n, \mathsf{Encap}_n, \mathsf{Encap}_n^*, \mathsf{Decap}_n)$ consisting of n independent copies of Π as follows:

- $\mathsf{Gen}_n(1^\lambda)$: outputs $\mathrm{PK} = (\mathrm{PK}_1, \ldots, \mathrm{PK}_n)$, $\mathrm{SK} = (\mathrm{SK}_1, \ldots, \mathrm{SK}_n)$ where $\{(\mathrm{PK}_i, \mathrm{SK}_i) \leftarrow \mathsf{Gen}(1^\lambda)\}$.

- $\mathsf{Encap}_n(\mathrm{PK})$: outputs $c = (c_1, \ldots, c_n), k = (k_1, \ldots, k_n)$ where $\{ (c_i, k_i) \leftarrow \mathsf{Encap}(\mathrm{PK}_i) \}$.
- $\mathsf{Encap}^*_n(\mathrm{PK})$: outputs $c^* = (c^*_1, \ldots, c^*_n)$ where $\{ c^*_i \leftarrow \mathsf{Encap}^*(\mathrm{PK}_i) \}$.
- $\mathsf{Decap}_n(\mathrm{SK}, c = (c_1, \ldots, c_n))$: outputs $(k_1, \ldots, k_n)$ where $\{ k_i = \mathsf{Decap}(\mathrm{SK}_i, c_i) \}$.

Then, the scheme Π^n is a valid wHPS scheme with (bigger) output domain $\mathcal{K}^n$. In other words, the output entropy is multiplied by a factor of n.

Theorem 3.5. *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 as defined above. Then Π^n is a wHPS with output domain $\mathcal{K}^n$.*

Proof. Correctness follows immediately. The ciphertext indistinguishability and smoothness properties of Π^n follow from those of Π by a simple hybrid argument over the indices $i \in [n]$. □

Summary and Parameters. We now saw how to construct a wHPS with small output size from any semantically secure PKE (Theorem 3.4), how to amplify the output size of a wHPS (Theorem 3.5), and how to go from a wHPS to a leakage-resilient encryption scheme (Theorem 3.3). Putting these results together, if we start with any PKE with secret-key size s, take our basic construction of wHPS with parameter m and apply n-wise parallel repetition, we get an ℓ-LR-PKE 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 follows.

Theorem 3.6. *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.*

Proof. Take our construction of wHPS from PKE (Theorem 3.4) with some polynomial parameter $m = m(\lambda)$, and apply parallel repetition (Theorem 3.5) with some polynomial parameter $n = n(\lambda) > 4\lambda$. This gives us a wHPS with output space $\mathcal{K} = (\mathbb{Z}_m)^n$ and key size $n(s + \lceil \log m \rceil) < 2ns$ (since $s > \log(m)$ for secure PKE). Applying our construction of LR-PKE from wHPS (Theorem 3.3) by using a universal hash function with λ-bit output as the extractor (Theorem 2.5), we get an ℓ-LR-PKE with leakage-bound $\ell = n \log(m) - 2\lambda > \frac{1}{2} n \log(m)$ and leakage rate $\alpha > \log(m)/4s$. Therefore, by choosing sufficiently large polynomials n, m, we can achieve the claim of the theorem, where n mainly influences the leakage amount and m mainly influences the leakage rate. This gives us a leakage-resilient PKE where the message size is λ bits, but we can always apply hybrid encryption to expand this to any desired polynomial. □

Leakage During Key Generation. Recall that our definition of leakage-resilient PKE only considered leakage on the secret-key SK and *not* on the randomness of the key-generation process. We note that we can also achieve the latter type of security, under an additional assumption. In particular, we need a wHPS scheme for which the

ciphertext indistinguishability property holds even if the attacker gets the full randomness of key generation (rather than just the secret-key), and the proof of Theorem 3.3 goes through as before. Our current construction of wHPS does *not* have this property. However, we can modify it slightly to get this property as follows: instead of sampling all m key pairs $\{(\mathrm{PK}_i, \mathrm{SK}_i)\}$ of the underlying PKE in full and storing only a single secret-key (t, SK_t), we sample the values PK_i for $i \neq t$ *obliviously*, without the corresponding secret-keys. This requires that our PKE supports an oblivious public-key sampling procedure, as defined in [20]. Although this property is not known to hold generically for every PKE, it can be obtained under various assumptions, such as CDH and RSA [20], for which we do not have prior leakage resilience results.

4. Leakage-Resilient Weak PRFs 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 pseudorandom. 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 4.1. (*Leakage-Resilient weak PRF* (*LR-wPRF*)) 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. 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 $K \leftarrow \mathcal{K}_\lambda$. The game then proceeds as follows.
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]
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 then outputs a bit b'.

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

[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 $F_K(\$)$.

Remarks on the Definition. Notice that the above definition implicitly already requires the input domain $|\mathcal{X}|$ to be super-polynomial to ensure that the challenge point X^* is *fresh* and $F_K(X^*)$ was not given out in the learning stage. Therefore, this definition rules out information-theoretic solutions for small input domains. We find this definition to be the easiest to use in our applications.

In the setting of no leakage ($\ell = 0$), we call a function $\mathcal{F}$ satisfying the above definition a *standard wPRF*. Such wPRFs exist assuming only that one-way functions exist [27,30]. The work of [45] (see also [23]) shows that *any* wPRF is also an ℓ-LR-wPRF for a logarithmic $\ell = O(\log(\lambda))$. Constructions of ℓ-LR-wPRF for larger ℓ can be derived from prior works on leakage-resilient public-key encryption [42], but only under strong public-key assumptions such as DDH. There are no prior-known constructions of ℓ-LR-wPRF for any super-logarithmic ℓ under any *symmetric-key assumptions*, such as the existence of one-way functions or collision-resistant hash functions.

We note that the given definition of LR-wPRF security also implies a *multi-challenge* variant where, during the challenge stage, the attacker is given arbitrarily many tuples $(X_1^*, Y_1^*), \ldots, (X_q^*, Y_q^*)$ which are either all pseudorandom with $X_i^* \leftarrow \mathcal{X}$, $Y_i^* = F_K(X_i^*)$ or all truly random with $(X_i^*, Y_i^*) \leftarrow \mathcal{X} \times \mathcal{Y}$. This follows by a simple hybrid argument.

From wPRF to CPA Encryption. We can use a LR-wPRF to construct leakage-resilient CPA-secure (LR-CPA) symmetric-key encryption. Since this construction is relatively obvious, we only sketch it and let the reader fill in the details. The security definition of ℓ-LR-CPA symmetric-key encryption consists of an initial learning stage where an attacker can adaptively ask arbitrary chosen-plaintext encryption queries interleaved with leakage queries to the leakage oracle $\mathcal{O}^{\ell}_{\mathrm{SK}}(\cdot)$. Later, after getting the challenge ciphertext, the attacker can ask for additional chosen-plaintext encryption queries *but not* leakage queries.

Assume that $\mathcal{F} = \{ F_K : \mathcal{X} \to \mathcal{Y}\}_{K \in \mathcal{K}}$ is a wPRF where the output domain $\mathcal{Y}$ is an additive group (e.g., bit strings under XOR). Then, we can encrypt a message $\mathrm{M} \in \mathcal{Y}$ via $\mathsf{Enc}_K(\mathrm{M}) = (X, F_K(X) + \mathrm{M})$ where $X \leftarrow \mathcal{X}$ comes from the random coins of the encryption. Decryption is obvious. We claim that if $\mathcal{F}$ is an ℓ-leakage-resilient wPRF, then the above encryption scheme is ℓ-LR-CPA secure. This simply follows by replacing the value $F_K(X)$ in the challenge ciphertext by a uniformly random and independent value, and arguing that this change is indistinguishable by the ℓ-LR-wPRF security of $\mathcal{F}$.

We note that the above scheme also remains secure even if the attacker can observe leakage that depends jointly on the secret-key and on the randomness used to answer CPA queries during initial the learning stage (but not the randomness used to create the challenge ciphertext). This simply follows since the scheme is *public-coin*, meaning that the randomness X used by the encryption process is provided in-the-clear by the ciphertext, and therefore, the attacker's future leakage queries in the initial learning stage can depend on X.

4.1. *Leakage Resilience Via Symmetric-Key wHPS*

Defining Symmetric-Key wHPS. Toward the goal of constructing a leakage-resilient 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 from Sect. 3.1. 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, analogously to the way we defined a public-key wHPS as a key encapsulation mechanism (KEM) with 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 $X_i \leftarrow \mathsf{Dist}_1$ (valid) and a random choice of $X^* \leftarrow \mathsf{Dist}_2$ (invalid), 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. Notice that this implies that there must be many keys K that are consistent with the values $\{(X_i, F_K(X_i))\}$. The additional distributions $\mathsf{Dist}_1, \mathsf{Dist}_2$ are *both* only used in the context of the security definitions and proofs, and never in the actual schemes, where we always sample $X \leftarrow \mathcal{X}$ uniformly at random. In the definition, we will also allow an additional secret "sampling key" samK which is needed in order to efficiently sample from the distributions $\mathsf{Dist}_1, \mathsf{Dist}_2$.

Definition 4.2. (*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 $K \in \mathcal{K}$ and 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 statistically equivalent:

$$\begin{aligned}&(X_1, \ldots, X_q, Y_1, \ldots, Y_q, X^*, Y^*)\\ &\equiv (X_1, \ldots, X_q, Y_1, \ldots, Y_q, X^*, U)\end{aligned}$$

where the distributions are defined by $K \leftarrow \mathcal{K}_\lambda, \mathsf{samK} \leftarrow \mathsf{SamGen}(K), \{X_i \leftarrow \mathsf{Dist}_1(\mathsf{samK}), Y_i := F_K(X_i)\}_{i\in[q]}$, $X^* \leftarrow \mathsf{Dist}_2(\mathsf{samK})$, $Y^* = F_K(X^*)$, 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 consistent key K.

Constructing LR-wPRF from SwHPS. We now construct a leakage-resilient wPRF from any symmetric-key wHPS. This is analogous to our results in the public-key setting (Theorem 3.3). In particular, we simply apply an extractor to the output of the symmetric-key wHPS.

Theorem 4.3. *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 $(\log(|\mathcal{Y}|) - \ell(\lambda), \epsilon(\lambda))$-extractor for some negligible $\epsilon(\lambda)$. 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.*

Proof. We prove the theorem via a hybrid argument over several games defined below.

Game 0. This game corresponds to the wPRF security game (Definition 4.1) where the challenger uses the bit $b = 0$, meaning that the challenge tuple is *pseudorandom*. In more detail, all wPRF queries during the learning stage are answered by choosing the input $(X_i, S_i) \leftarrow \mathcal{X} \times \mathcal{S}$ at random and setting the output $Z_i = F'_K((X_i, S_i)) = \mathsf{Ext}(F_K(X_i), S_i)$. The challenge tuple $((X^*, S^*), Z^*)$ is chosen the same way.

Game 1. In this game, we rely on the symmetric-key wHPS properties of $\mathcal{F}$ to change the distribution of all the $\{X_i\}$ values during the learning stage to come from Dist_1 and the X^* value in the challenge to come from Dist_2. In particular, during initialization, the challenger still chooses $K \leftarrow \mathcal{K}$, but now also choose $\mathsf{samK} \leftarrow \mathsf{SamGen}(K)$. During the learning stage, whenever answering wPRF queries, the challenger now chooses $X_i \leftarrow \mathsf{Dist}_1(\mathsf{samK})$ and then $S_i \leftarrow \mathcal{S}$, $Z_i = F'_K((X_i, S_i))$ as before. During the challenge stage, the challenger now chooses $X^* \leftarrow \mathsf{Dist}_2(\mathsf{samK})$ and again completes it with $S^* \leftarrow \mathcal{S}$, $Z^* = F'_K((X^*, S^*))$ as before.

We argue that Game 0 and 1 are computationally indistinguishable by the *input indistinguishability* property of the symmetric-key wHPS. Let q be the total number of wPRF queries that $\mathcal{A}$ makes during the learning stages. Then, we have a reduction which takes a tuple $(K, X_1, \ldots, X_q, X^*)$ as input and uses this to answer leakage and wPRF queries for $\mathcal{A}$ during the wPRF game and to form the challenge. If $\{X_i\}$, X^* are chosen uniformly at random (Dist_0), then this perfectly simulates Game 0 and if they are chosen via $\{X_i \leftarrow \mathsf{Dist}_1(\mathsf{samK})\}$, $X^* \leftarrow \mathsf{Dist}_2(\mathsf{samK})$ then this perfectly simulates Game 1.

Game 2. In this game, the challenge tuple $((X^*, S^*), Z^*)$ is chosen by selecting $Z^* \leftarrow \mathcal{Z}$ uniformly at random. All other values are computed as in Game 1.

Notice that in both Games 1,2 we choose $X_i \leftarrow \mathsf{Dist}_1(\mathsf{samK})$ during the learning stage and the challenge $X^* \leftarrow \mathsf{Dist}_2(\mathsf{samK})$. Let us define $Y^* = F_K(X^*)$. We can rely on the *smoothness* property of symmetric-key wHPS to claim that Y^* is uniformly random even given X^* and all of the wPRF query responses that the attacker sees in the learning stage, denoted by

$$P = (\{X_i, S_i, Z_i = F'_K((X_i, S_i))\}_{i\in[q]}),$$

Therefore, we know that $\mathbf{H}_\infty(Y^*|X^*, P) = \log(|\mathcal{Y}|)$. Let $L \in \{0,1\}^{\ell(\lambda)}$ denote the response(s) of the leakage oracle $\mathcal{O}^\ell_K(\cdot)$ during the learning stage (recall that we can assume the attacker just makes a single query at the end of the learning stage). Therefore, by Lemma 2.3, we also have

$$\mathbf{H}_\infty(Y^* \mid X^*, P, L) \geq \mathbf{H}_\infty(Y^* \mid X^*, P) - \ell(\lambda) = \log(|\mathcal{Y}|) - \ell(\lambda).$$

Therefore, conditioned on (X^*, L, P), the value Y^* has $\log(|\mathcal{Y}|) - \ell(\lambda)$ bits of entropy. Finally, we can rely on the security of extractors to get the statistical indistinguishability:

$$(L, P, X^*, S^*, Z^* = \mathsf{Ext}(Y^*; S^*)) \approx_s (L, P, X^*, S^*, Z^* \leftarrow \mathcal{Z})$$

where we rely on the fact that the seed $S^* \leftarrow \mathcal{S}$ is chosen at random, independently of P, L. Therefore, even conditioned on everything the attacker sees in the learning stage, and on the challenge input (X^*, S^*) the value $Z^* = F_{K'}((X^*, S^*)) = \mathsf{Ext}(Y^*; S^*)$ is statistically indistinguishable from uniform. This proves the indistinguishability of Games 1 and 2.

Game 3. This game corresponds to the wPRF security game (Definition 4.1) where the challenger uses the bit $b = 1$. In particular, the difference from Game 2 is that we switch the distribution of all of the inputs $\{X_i\}$ seen during the learning stage, and the input X^* used in the challenge, back to being chosen uniformly at random (Dist_0) via $\{X_i \leftarrow \mathcal{X}\}$ and $X^* \leftarrow \mathcal{X}$. All of the Z_i values during the learning stage are still chosen as wPRF outputs $Z_i = F'_K((X_i, S_i))$ and the challenge Z^* is still chosen uniformly at random via $Z^* \leftarrow \mathcal{Z}$ (as in Game 2).

Games 2 and 3 are computationally indistinguishable by the *input indistinguishability* property of the symmetric-key wHPS. The reduction is essentially the same as the one showing the indistinguishability of Games 0 and 1.

□

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. As in the public-key setting, we begin by constructing a simple SwHPS with a short output size, and then amplify the output size via parallel repetition.

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:

[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.

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

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 $\text{M} \leftarrow \mathbb{Z}_m$. Given the wPRF $\mathcal{F}_{weak}$ and the resulting encryption scheme $(\mathsf{Enc}, \mathsf{Dec})$ as above, we define the symmetric-key wHPS system:

$$\mathcal{F}_{SwHPS} = \{F_K : \mathcal{C}^m \to \mathbb{Z}_m\}_{K \in ([m] \times \mathcal{K})}$$
$$\text{where } F_{(K=(t,k))}(X = (c_1, \ldots, c_m)) := \mathsf{Dec}_k(c_t).$$

Notice that we can efficiently sample uniformly random inputs from the domain $\mathcal{C}^m$ of F_K (without knowing K), which corresponds to sampling from the distribution Dist_0 (see Definition 4.2). 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 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 4.4. *Assuming $\mathcal{F}_{weak}$ is a standard wPRF, the function family $\mathcal{F}_{SwHPS}$ as defined above is a symmetric-key wHPS.*

Proof. Let us start with the *input indistinguishability* property of symmetric-key wHPS, showing the indistinguishability of many samples from the various distributions Dist_0 (uniform) Dist_1, (valid) and Dist_2 (invalid) even when given K in full. For this, it helps to think of the three distributions in an alternate way. In all three of them, the ciphertext $c_t \leftarrow \mathcal{C}$ is just chosen uniformly at random, and we can define $r := \mathsf{Dec}_k(c_t)$. The distributions only differ in how we choose c_i for $i \neq t$. In *Distribution 0*, we choose all the other $c_i \leftarrow \mathcal{C}$ to be random as well, in *Distribution 1* we set $c_i \leftarrow \mathsf{Enc}_{k_i}(r)$, and in *Distribution 2* we set $c_i \leftarrow \mathsf{Enc}_{k_i}(r + i - t)$. This alternate description is completely equivalent to the distributions defined above. Now, it is easy to see that for $i \neq t$, the ciphertexts c_i from any of these distributions are indistinguishable from the uniform distribution over $\mathcal{C}$ (by the security of wPRF $\mathcal{F}_{weak}$ with the key k_i) even if

the attacker knows $K = (t, k)$ and the choice of r. This even holds if the attacker can see arbitrarily many other samples from these various distributions. This gives us input indistinguishability.

Next, let us show the *smoothness* property of symmetric-key wHPS. We need to show that given arbitrarily many values $\mathsf{Valid} = \{X_i \leftarrow \mathsf{Dist}_1(\mathsf{samK}), F_K(X_i)\}$ and some challenge point $X^* \leftarrow \mathsf{Dist}_2(\mathsf{samK})$, the value $F_K(X^*)$ is (perfectly) uniformly random and independent of Valid. Actually, we show that this holds even if the attacker were also given samK in full. Notice that the value $\mathsf{samK} = (k_1, \ldots, k_m)$ already completely determines $F_K(X_i)$ when $X_i \leftarrow \mathsf{Dist}_1(\mathsf{samK})$ because we can write $F_K(X_i = (c_1, \ldots, c_m)) = \mathsf{Dec}_{k_1}(c_1)$. Therefore, the choice of the index t in the key $K = (t, k)$ is uniform and independent of $\mathsf{Valid}, \mathsf{samK}, X^*$. On the other hand, when $X^* \leftarrow \mathsf{Dist}_2(\mathsf{samK})$ then we can write:

$$F_{(t,k)}(X^* = (c_1^*, \ldots, c_m^*)) = \mathsf{Dec}_{k_t}(c_t^*) = r + t$$

which is uniformly random in $\mathbb{Z}_m$ over the choice of t and hence independent of $\mathsf{Valid}, \mathsf{samK}, X^*$. □

Output Amplification via Parallel Repetition. The above construction of a symmetric-key wHPS only gets us a polynomial-sized output domain, which means that the outputs only have $O(\log(\lambda))$ entropy. When using symmetric-key wHPS to get leakage-resilient wPRF, we cannot use this to extract even one bit. Therefore, we somehow need to amplify the output domain and entropy. As in the public-key setting, this turns out to be very easy with symmetric-key wHPS, just by using "parallel repetition" where we concatenate several independent copies together. More specifically, we have the following theorem.

Theorem 4.5. *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.

Proof. The proof essentially follows directly from the definition of SwHPS. By the definition, there are some algorithms $\mathsf{SamGen}, \mathsf{Dist}_1, \mathsf{Dist}_2$ for the scheme $\mathcal{F}$ that satisfy the input indistinguishability and smoothness properties. We naturally define the algorithms $\mathsf{SamGen}^n, \mathsf{Dist}_1^n, \mathsf{Dist}_2^n$ as:

$\mathsf{samK} \leftarrow \mathsf{SamGen}^n(K)$.	Parse $K = (k_1, \ldots, k_n)$ and sample $\{\mathsf{samK}_i \leftarrow \mathsf{SamGen}(k_i) :\in [n]\}$. Output $\mathsf{samK} = (\mathsf{samK}_1, \ldots, \mathsf{samK}_n)$.
$X \leftarrow \mathsf{Dist}^n_{b \in \{1,2\}}(\mathsf{samK})$.	Sample $\{x_i \leftarrow \mathsf{Dist}_b(\mathsf{samK}_i) : i \in [n]\}$. Output $X = (x_1, \ldots, x_n)$.

Now, we want to show that the input indistinguishability and smoothness properties holds for $\mathcal{F}^n$ with the above algorithms.

- For *input indistinguishability* of $\mathcal{F}^n$, we want to show that 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:

$$\begin{aligned}&(K=(k_1,\ldots,k_n),X_1=(x_{1,1},\ldots,x_{1,n})\ldots,X_q=(x_{q,1},\ldots,x_{q,n}))\\&\approx_c(K=(k_1,\ldots,k_n),X'_1=(x'_{1,1},\ldots,x'_{1,n})\ldots,X'_q=(x'_{q,1},\ldots,x'_{q,n}))\end{aligned}$$

where $K \leftarrow (\mathcal{K}_\lambda)^n, \mathsf{samK} \leftarrow \mathsf{SamGen}^n(K), \{X_i \leftarrow \mathsf{Dist}^n_{b_i}(\mathsf{samK})\}, \{X'_i \leftarrow \mathsf{Dist}^n_{b'_i}(\mathsf{samK})\}$. This follows via a sequence of n hybrid steps, where we use the input indistinguishability property of the "small" scheme in position $j \in [n]$ to switch from sampling $\{x_{i,j} \leftarrow \mathsf{Dist}_{b_1} : i \in [q]\}$ to sampling $\{x'_{i,j} \leftarrow \mathsf{Dist}_{b'_1} : i \in [q]\}$.

- For the *smoothness* property of $\mathcal{F}^n$, we want to show that for any polynomial $q = q(\lambda)$ the following distributions are statistically equivalent:

$$\begin{aligned}&(X_1,\ldots,X_q,Y_1,\ldots,Y_q,X^*=(x^*_1,\ldots,x^*_n),Y^*=(y^*_1,\ldots,y^*_n)\\&\equiv(X_1,\ldots,X_q,Y_1,\ldots,Y_q,X^*=(x^*_1,\ldots,x^*_n),U=(u_1,\ldots,u_n))\end{aligned}$$

where the distributions are defined by $K \leftarrow (\mathcal{K})^n, \mathsf{samK} \leftarrow \mathsf{SamGen}^n(K)$, $\{X_i \leftarrow \mathsf{Dist}^n_1(\mathsf{samK})\}$, $X^* \leftarrow \mathsf{Dist}^n_2(\mathsf{samK})$, $\{Y_i = F_K(X_i)\}$, $Y^* = F_K(X^*)$, and $U \leftarrow (\mathcal{Y})^n$. This too follows by a sequence of n hybrid steps where we use the smoothness property of the "small" scheme in position $j \in [n]$ to switch $y^*_j = f_{k_j}(x^*_j)$ to $u_j \leftarrow \mathcal{Y}$. □

Putting it All Together. We now summarize our final results by combining all of the ingredients we developed so far in this section.

Theorem 4.6. *Assuming the existence of one-way functions, there exist $\ell(\lambda)$-LR-wPRFs and $\ell(\lambda)$-LR-CPA symmetric-key encryption schemes for any arbitrarily large polynomial $\ell(\lambda)$. Furthermore, assuming the existence of standard wPRFs with key size $s(\lambda)$, the above schemes exist for any* leakage rate $\alpha(\lambda) = O\left(\frac{\log(\lambda)}{s(\lambda)}\right)$.

Proof. Let $m = m(\lambda)$ be some polynomial parameter, which is a power of 2. The existence of standard one-way functions implies the existence of a standard wPRF family

$$\mathcal{F}_{wPRF} = \left\{F_K : \{0,1\}^{i(\lambda)} \to \{0,1\}^{\log(m(\lambda))}\right\}_{K\in\{0,1\}^{s(\lambda)}}$$

for some super-logarithmic input size $i = i(\lambda) = \omega(\log(\lambda))$, by the classic results of [27,30]. Using our basic construction of SwHPS (Theorem 4.4) with parameter $m = m(\lambda)$ and parallel repetition with some parameter $n = n(\lambda) > 4\lambda$, we get a symmetric-key wHPS with key size $n(s + \log(m)) < 2ns$ and output size $n \log m$ (in bits). Using our construction of leakage-resilient wPRFs (Theorem 4.3) and employing a universal hash function with λ-bit output as the extractor, we get an $\ell(\lambda)$-LR-wPRF with leakage-bound $\ell(\lambda) = n \log m - 2\lambda > \frac{1}{2} n \log m$, leakage rate $\alpha = \log(m)/4s$, and output size λ. Therefore, by choosing sufficiently large polynomials n, m we can achieve the claim of the theorem, where n mainly influences the leakage amount and m mainly influences the leakage rate. This gives us a leakage-resilient wPRF with λ-bit output, but we can

easily amplify this to any number of bits without affecting the secret-key size or leakage amount using a standard (non-leakage-resilient) pseudorandom generator (PRG). The results for CPA encryption follow from those for wPRF. □

5. Leakage-Resilient Message Authentication

We now construct leakage-resilient message-authentication codes (LR-MACs) from the minimal assumption that one-way functions exist.

5.1. *Definitions*

We begin by defining leakage-resilient MACs. We give a definition with several meaningful variants and show interesting connections between them.

Definition 5.1. (*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* ($\ell - LR - MAC$) 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 define two useful variants.

Definition 5.2. (*nvq-MAC, SU-MAC*) We define two variants of the LR-MAC security game:

- *No-Verification-Query (nvq-MAC):* In this weaker security notion, the attacker does not get access to the verification oracle $\mathsf{Ver}_K(\cdot,\cdot)$ during the learning stage. (The default notion in Definition 5.1 allows unlimited verification queries.)
- *Strongly Unforgeable MAC (SU-MAC):* In this stronger security notion, the attacker cannot come up with a new tag for a previously authenticated message. Formally, we redefine the winning condition so that the attacker *wins* on a forgery (M^*, σ^*) as long as: *the tagging oracle never returned the tag* σ^* *on any prior tagging query with the message* M^* *and* $\mathsf{Ver}_K(\mathrm{M}^*, \sigma^*) = 1$.

(The default notion in Definition 5.1 requires that the tagging oracle is never queried with M^*.)

We also consider a combination of both of the above variants (strong unforgeability against a no-verification-query attack), which we call an SU-nvq-MAC.

We note that "no verification query (nvq)" security is a weaker but already meaningful notion, which may be useful in many practical scenarios. For example, we can insist that whenever the user detects an invalid tag, she stops using the scheme in the future. That way, the attacker does not get to make verification queries beyond a single forgery attempt. We also note that any nvq-MAC which is ℓ-leakage-resilient, is also secure as long as the attacker can make up to $q < \ell$ verification queries and gets up to $(\ell - q)$ bits of leakage, since each verification query can be thought of as an additional bit of leakage on the secret-key. Of course, putting *any* a-priori bound on the number of allowed verification queries may be too restrictive in some scenarios, and therefore, we still desire a construction achieving full MAC security with unlimited verification queries rather than just nvq security.

Relations Between Definitions. Unfortunately, there are examples of nvq-MACs which are completely *insecure* in the setting of unlimited verification queries. In other words, it is *not* the case that nvq-MAC security is equivalent to full MAC security. However, we now show that when it comes to *strongly unforgeable* SU-MACs, the above equivalence does hold and there is no distinction between only considering a "no-verification-query" attack and a general attack in which unlimited verification queries are allowed. In other words, we show that the seemingly weaker notion of an SU-nvq-MAC security is actually equivalent to SU-MAC security. The above equivalence even holds in the setting of leakage.

Theorem 5.3. *Let* Π *be any* ℓ*-leakage-resilient SU-nvq-MAC (strongly unforgeable MAC against a no-verification-query attack). Then,* Π *is also an* ℓ*-leakage-resilient SU-MAC (strongly unforgeable MAC against unlimited verification queries).*

Proof. Let $\mathcal{A}$ be an attacker against the ℓ-leakage-resilient SU-MAC security of some scheme $(\mathsf{Tag}, \mathsf{Ver})$. Without loss of generality, assume that $\mathcal{A}$ asks at most q verification queries and always asks a verification query (m^*, σ^*) for the values that it later submits as its forgery. We call a verification query (m, σ) "fresh" if no prior tagging query with the message m returned σ.

We show how to convert $\mathcal{A}$ into an attacker $\mathcal{B}$ that breaks the ℓ-leakage-resilient SU-nvq-MAC security of the scheme. In other words, $\mathcal{B}$ manages to break strong unforgeability without making *any* verification queries during the learning stage.

The attacker $\mathcal{B}$ chooses a random index $i \leftarrow [q]$. It runs $\mathcal{A}$ and gives it access to its own tagging and leakage oracles during the learning stage, and simulates the verification oracle for $\mathcal{A}$ as follows. For the first $i - 1$ verification queries that $\mathcal{A}$ makes, $\mathcal{B}$ simply checks if the query is "fresh" and if so responds with 0 (rejects) else with 1 (accepts). For the ith verification query (m, σ) asked by $\mathcal{A}$, the attacker $\mathcal{B}$ submits (m, σ) as its forgery attempt.

Let E_j be the event that jth verification query made by $\mathcal{A}$ in the original SU-MAC security game is the *first* one that would qualify as a valid forgery (it is "fresh" and the signature verifies). For $j = i$, the probability of E_j happening is exactly the same in the

original SU-MAC security game and in the simulation. This is because the simulation only differs in how it responds to verification queries, but in both cases if the event E_j occurs, then the first $j-1$ verification queries simply reject. Recall that we assume $\mathcal{A}$ always submits its forgery as a verification query, and hence, one of the events E_j must occur if $\mathcal{A}$ wins the game. Letting $\mathbf{i}$ be the random index chosen by $\mathcal{B}$ we have:

$$\Pr[\mathcal{B} \text{ wins }] = \Pr\left[\bigcup_{j=1}^{q}(E_j \wedge (\mathbf{i}=j))\right] = \sum_{j=1}^{q}\Pr[E_j \wedge (\mathbf{i}=j)]$$
$$= \frac{1}{q}\sum_{j=1}^{q}\Pr[E_j] \geq \frac{1}{q}\Pr[\bigcup_{j=1}^{q}E_j] \geq \frac{1}{q}\Pr[\mathcal{A} \text{ wins }].$$

where we rely on the fact that the events E_j are disjoint and that the choice of $\mathbf{i}$ is random and independent of E_j.

Therefore, the advantage of the attacker $\mathcal{B}$ in the SU-nvq-MAC game is only a factor of q smaller than that of $\mathcal{A}$ in the SU-MAC game, proving the theorem. □

5.2. *Leakage-Resilient MAC with "No Verification Query" Security*

We begin by presenting a simple construction of a leakage-resilient nvq-MAC, meaning that it only achieves security against a no-verification-query (nvq) attack. In Sect. 5.3, we will then show how to "upgrade" any such MAC into a fully secure (and even strongly unforgeable) MAC.

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 the keys $K \in \mathcal{K}^{MAC}$ 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:

- $\mathsf{Tag}_K(\text{M})$: 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:

$$\sigma_{i,j} := \begin{cases} f_{k_i}(r||\text{M}) & \text{if } j = t_i \\ y \leftarrow \mathcal{Y} & \text{otherwise} \end{cases}$$

  In other words, each row $i \in [n]$ of the matrix $\{\sigma_{i,j}\}$ contains one pseudorandom value under the key k_i in the column t_i, and the rest of the row is truly random.
- $\mathsf{Ver}_K(\text{M}, \sigma)$: Parse $\sigma = (r, \{\sigma_{i,j}\})$. For all $i \in [n]$, check that $f_{k_i}(r||\text{M}) = \sigma_{i,t_i}$.

Security Intuition. Although we do not formally frame the above construction inside of an abstraction (as we did with wHPS for encryption), it is useful to explain its security 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 $\sigma_{i,j} := f_{k_{i,j}}(r||\text{M})$ 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 $\text{M}^*, \sigma^* = (r^*, \{\sigma^*_{i,j}\})$ is *valid* if there is some pair (i, j) with $j \neq t_i$ such that $\sigma^*_{i,j} = f_{k_{i,j}}(r^*||\text{M}^*)$, 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 $(r^*||\text{M}^*)$*, 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 is no easier than guessing the value $T = (t_1, \ldots, t_n)$ *since the only pairs* (i, j) *for which* $\sigma^*_{i,j} = f_{k_{i,j}}(r^*||\text{M}^*)$ *are ones where* $j = t_i$*. But* T *has* $n\log(m)$ *bits of entropy and is completely independent of the outputs of the alternate tagging 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 5.4. *If* $\mathcal{F}_{\text{prf}}$ *is a PRF family with parameters as above, then the given construction is an* $\ell(\lambda)$*-leakage-resilient nvq-MAC for any* $\ell(\lambda) = n(\lambda)\log(m(\lambda)) - \omega(\log(\lambda))$.

Proof. Let Game 0 be the original ℓ-LR nvq-MAC security game with some PPT attacker $\mathcal{A}$ (as in Definition 5.1). Let Game 1 be a modified version of the game where the challenger picks an entire $n \times m$ matrix of random PRF keys $\{k_{i,j} \leftarrow \mathcal{K} : i \in [n], j \in [m]\}$ at the beginning of the game and sets the MAC key $K = ((k_{1,t_1}, t_1), \ldots, (k_{n,t_n}, t_n))$ by picking the indices $\{t_i \leftarrow [m]\}$ randomly. During the learning stage, whenever answering some tagging query with message M for $\mathcal{A}$, the challenger now computes $\sigma = \{\sigma_{i,j} = f_{k_{i,j}}(r||\text{M})\}$. In other words, the only change from Game 0 is that, in Game 1, the values $\sigma_{i,j}$ are now all chosen pseudorandomly under different PRF keys. We rely on the PRF security of $f_{k_{i,j}}(\cdot)$ for $i \in [n]$, $j \in [m] \setminus \{t_i\}$ to claim that Games 0 and 1 are indistinguishable and the attacker's winning probability does not change between them. This holds even if the attacker were given the MAC key K in full. (Here we rely on the fact that the random nonce r chosen in each tagging query is likely to be fresh even if the message M has been queried before, and therefore, the PRF inputs $(r||\text{M})$ are always fresh.) Therefore, we have

$$\Pr[\mathcal{A} \text{ wins in Game 1}] \geq \Pr[\mathcal{A} \text{ wins in Game 0}] - \text{negl}(\lambda).$$

In Game 1, define the event E occurs if, in the attacker's forgery $\text{M}^*, \sigma^* = (r^*, \{\sigma^*_{i,j}\})$, there is some pair $(i, j) \in [n] \times [m]$ with $j \neq t_i$ such that $f_{k_{i,j}}(r^*||\text{M}^*) = \sigma^*_{i,j}$. In other

words, the attacker correctly predicts one of the PRF values in the matrix, outside of the special positions (i, t_i).

On the one hand, we claim that the attacker is unlikely to win by causing E to occur:

$$\Pr[\mathcal{A} \text{ wins in Game 1} \wedge E] \leq \Pr[E] \leq \text{negl}(\lambda).$$

This follows by the security of the PRF, where we replace the functions $f_{k_{i,j}}$ for $j \neq t_i$ by truly random functions $f_{i,j}$. Since the forgery message M^* is fresh, the attacker does not get any information about $f_{i,j}(r^*||\text{M}^*)$ throughout the game, and therefore, the probability that $f_{i,j}(r^*||\text{M}^*) = \sigma^*_{i,j}$ is at most $\frac{1}{|\mathcal{Y}|} = \text{negl}(\lambda)$. The inequality then follows by taking a union bound over all such pairs (i, j) with $j \neq t_i$.

On the other hand, we also claim that the attacker is unlikely to win while ensuring that E does not occur:

$$\Pr[\mathcal{A} \text{ wins in Game 1} \wedge \neg E] \leq 2^{-n\log(m)+\ell} \leq \text{negl}(\lambda).$$

This follows by an information-theoretic argument, showing that $\mathcal{A}$ does not have enough information about the indices $T = (t_1, \ldots, t_n)$ to *only* predict the correct outputs $\sigma^*_{i,j} = f_{k_{i,j}}(r^*||\text{M}^*)$ when $j = t_i$. Notice that in Game 1, the challenger's responses to the tagging queries only depend on the values $\{k_{i,j}\}$ and is completely independent of T. Therefore, the attacker's view in Game 1 contains only ℓ bits of information about T from its leakage queries. However, if the event "$\mathcal{A}$ wins in Game 1 $\wedge \neg E$" occurs, it means that the attacker's forgery has the property that the only tuples (i, j) for which $f_{k_{i,j}}(r^*||\text{M}^*) = \sigma^*_{i,j}$ are ones where $j = t_i$. In this case, we can completely recover T from the attacker's forgery given $\{k_{i,j}\}$. Since the min-entropy of T given the set $\{k_{i,j}\}$ and the view of $\mathcal{A}$ in Game 1 is

$$\mathbf{H}_\infty(T \mid \{k_{i,j}\}, \text{view } \mathcal{A}) \geq \mathbf{H}_\infty(T \mid \{k_{i,j}\}) - \ell \geq n\log(m) - \ell$$

the above probability is bounded by $2^{-n\log(m)+\ell} \leq \text{negl}(\lambda)$ as claimed.

Putting the above claims together, we get $\Pr[\mathcal{A} \text{ wins in Game 1}] \leq \text{negl}(\lambda)$ and therefore:

$$\Pr[\mathcal{A} \text{ wins in Game 0}] \leq \Pr[\mathcal{A} \text{ wins in Game 1}] + \text{negl}(\lambda) \leq \text{negl}(\lambda)$$

which proves the theorem as claimed. □

Insecurity with verification queries. The main problem with allowing verification queries in the above construction is that the attacker can completely learn the indices $t_1, \ldots, t_n$ in the secret-key. In particular, the attacker can make a single tagging query on some message M and get back the tag $\sigma = (r, \{\sigma_{i,j}\})$. Then for each pair $(i, j) \in [n]\times[m]$, the attacker can modify only the component $\sigma_{i,j}$ of σ (say, flip the last bit of it) and make a verification query to check whether the modified tag is still valid for the message M. If not, then the attacker learns that $t_i = j$. By making sufficiently many such verification queries, the attacker then recovers $(t_1, \ldots, t_n)$. Although this by itself does not lead to a concrete attack on the leakage-resilient security of the scheme, it completely breaks our proof technique which relied on the fact that the attacker does not gain information

about $(t_1, \ldots, t_n)$ other than through leakage queries. Next, we show how to generically convert a leakage-resilient nvq-MAC into a strongly unforgeable SU-MAC that allows verification queries.

5.3. *Leakage-Resilient Strongly Unforgeable MAC*

We now show how to convert any leakage-resilient nvq-MAC (no-verification-query) into an SU-MAC (strongly unforgeable), which is the strongest notion of security we defined. Outside of the setting of leakage resilience, a transformation along these lines was given in the work of Dodis et al. [18]. Unfortunately, this transformation requires additional components in the secret-key and is not known to be secure in the presence of leakage. We give an alternative transformation, which we can prove secure even in the presence of leakage. The construction is described as follows:

- Let $(\mathsf{nvqTag}, \mathsf{nvqVer})$ be an $\ell(\lambda)$-leakage-resilient nvq-MAC (no-verification-query security) with key space $\mathcal{K}^{nvq}$ and tag space $\mathcal{T}$.
- Let $\mathcal{F} = \{f_k : \mathcal{X} \to \mathcal{Y}\}_{k \in \mathcal{K}^{hps}}$ be a symmetric-key wHPS with key space $\mathcal{K}^{hps}$.
- Let $\mathcal{H} = \{h_k : \mathcal{T} \to \mathcal{Z}\}_{k \in \mathcal{Y}}$ be a $2^{-u(\lambda)}$-secure (information-theoretic) *one-time-MAC*, meaning that: for all (computationally unbounded) $\mathcal{A}$ and all $\psi \in \mathcal{T}$ we have

$$\Pr[h_k(\psi') = \tau' \wedge \psi' \neq \psi \mid k \leftarrow \mathcal{Y}, \tau := h_k(\psi), (\psi', \tau') \leftarrow \mathcal{A}(\tau)] \leq 2^{-u(\lambda)}.$$

  We assume $u(\lambda)$ is super-logarithmic.

Define the MAC $(\mathsf{Tag}, \mathsf{Ver})$ with key space $\mathcal{K} = \mathcal{K}^{nvq} \times \mathcal{K}^{hps}$ as follows.

- $\mathsf{Tag}_K(\mathrm{M})$: Parse $K = (k_1, k_2)$. Choose $x \leftarrow \mathcal{X}$, $\psi \leftarrow \mathsf{nvqTag}_{k_1}(x||\mathrm{M})$. Compute $k_3 := f_{k_2}(x)$, $\tau := h_{k_3}(\psi)$. Output $\sigma := (x, \psi, \tau)$.
- $\mathsf{Ver}_K(\mathrm{M}, \sigma)$: Parse $K = (k_1, k_2), \sigma = (x, \psi, \tau)$. Compute $k_3 := f_{k_2}(x)$. If $\mathsf{nvqVer}_{k_1}((x||\mathrm{M}), \psi) = 1$ and $h_{k_3}(\psi) = \tau$ output 1 else output 0.

Security Intuition. We show that the above construction is a leakage-resilient strongly unforgeable MAC (SU-MAC). Recall that, by Theorem 5.3, we only need to prove SU-nvq security (strong unforgeability against a no-verification-query attack) and we get security against unlimited verification queries for free. Let us consider an attack where the adversary gets some tag $\sigma = (x, \psi, \tau)$ for a message M under the above MAC and attempts to come up with a modified tag $\sigma^* = (x^*, \psi^*, \tau^*) \neq \sigma$ for the same message (this appears to be the most illustrative case).

If the attacker modifies $x^* \neq x$, then ψ^* would be a valid tag for a *new* message $(x^*||\mathrm{M})$ under the original nvq-MAC, which would violate its security. Therefore, assume that the attacker leaves $x^* = x$ the same. If $\psi^* = \psi$ is also left the same, then we must have $\tau^* = \tau = h_{f_{k_2}}(\psi^*)$ in order for the tag to verify, in which case the entire tag $\sigma^* = \sigma$ has been unchanged and hence is not a valid forgery. On the other hand, if the attacker changes $\psi^* \neq \psi$, then he cannot come up with a valid $\tau^* = h_{k_3}(\psi^*)$ given only $\tau = h_{k_3}(\psi)$ since $k_3 = f_{k_2}(x)$ is pseudorandom and $h_{k_3}(\cdot)$ is a one-time MAC. The main difficulty lies in proving that the above holds in the setting of leakage, if the attacker can leak information about k_2 *after seeing* x. Here, we use the fact that $f_{k_2}(\cdot)$

is a symmetric-key wHPS to argue that this information can decrease the entropy of $k_3 = f_{k_2}(x)$ by at most ℓ bits, which is insufficient to break one-time MAC security. Implicitly, we are relying on the fact that a (symmetric-key) wHPS is secure against *after-the-fact leakage*, a property that we examine in further detail in Sect. 6.3.

Theorem 5.5. *Let* (nvqTag, nvqVer) *be an* $\ell(\lambda)$*-leakage-resilient nvq-MAC (secure against a no-verification-query attack). Let* $\mathcal{F}$ *be a symmetric-key wHPS and* $\mathcal{H}$ *a* $2^{-u(\lambda)}$*-secure (information-theoretic) one-time MAC with parameters as described above. Then, the above construction of* (Tag, Ver) *is* $\ell'(\lambda)$*-leakage-resilient SU-MAC (strongly unforgeable with unlimited verification queries) for any* $\ell'(\lambda) = \min\{\ell(\lambda), u(\lambda) - \omega(\log(\lambda))\}$.

Proof. By Theorem 5.3, we only need to prove SU-nvq-MAC security (strongly unforgeable security against a "no-verification-query" attack), and full SU-MAC security (against unlimited verification queries) follows automatically. Let $\mathcal{A}$ be a PPT attacker against the ℓ'-leakage-resilient SU-nvq-MAC security game (as in Definitions 5.1, 5.2) of the scheme (Tag, Ver). Assume that $\mathcal{A}$ makes at most q queries to the tagging oracle throughout the game. We define the following events within the context of the security game:

- The event Win occurs if $\mathcal{A}$ wins the security game against the challenger. In other words, it outputs a valid forgery where the combination of (M^*, σ^*) is fresh, meaning that no prior tagging query with message M^* returned the tag σ^*.
- The event Match_j to occur if the attacker's final forgery attempt is of the form $\mathrm{M}^*, \sigma^* = (x^*, \psi^*, \tau^*)$ and the jth query to the tagging oracle is for a matching message $\mathrm{M}_j = \mathrm{M}^*$ and the response is $\sigma_j = (x_j, \psi_j, \tau_j)$ with matching $x_j = x^*$ (the other components may not match).
- We define the event $\mathsf{Match} := \bigcup_{j=1}^{q} \mathsf{Match}_j$. In other words, the attacker's final forgery attempt is of the form $\mathrm{M}^*, \sigma^* = (x^*, \psi^*, \tau^*)$ such that *some* prior query to the tagging oracle has a matching message M^* and is answered with a tag $\sigma = (x, \psi, \tau)$ having a matching $x = x^*$.

Firstly, we wish to show that

$$\Pr[\mathsf{Win} \wedge \mathsf{Match}] = \Pr[\mathsf{Win}] - \Pr[\mathsf{Win} \wedge \neg\mathsf{Match}] \geq \Pr[\mathsf{Win}] - \mathrm{negl}(\lambda) \tag{5.1}$$

We show this by relying on the security of the underlying nvq-MAC. Notice that whenever $\mathsf{Win} \wedge \neg\mathsf{Match}$ occurs, the tag ψ^* is a valid tag of the message $(x^*||\mathrm{M}^*)$ under the nvq-MAC, and the nvqTag algorithm was previously never executed with $(x^*||\mathrm{M}^*)$ as an input during the course of the game. Therefore, we have a simple reduction that breaks the ℓ'-LR nvq-MAC security of the underlying nvq-MAC with probability $\Pr[\mathsf{Win} \wedge \neg\mathsf{Match}]$, meaning that the latter must be negligible.

Next, we wish to show that

$$\forall j \in [q] : \ \Pr[\mathsf{Win} \wedge \mathsf{Match}_j] \leq \mathrm{negl}(\lambda) \tag{5.2}$$

To show this, we need to rely on the security of the symmetric-key wHPS (SwHPS). Let us consider a modified version of the security experiment. The challenger samples a SwHPS

sampling key $\mathsf{samK} \leftarrow \mathsf{SamGen}(k_2)$ at the beginning of the game. All tagging queries *other than* the jth one are answered by choosing the component $x \leftarrow \mathsf{Dist}_1(\mathsf{samK})$ from the valid distribution and the jth tagging query are answered by choosing $x \leftarrow \mathsf{Dist}_2(\mathsf{samK})$ from the invalid distribution. The remainder of the tagging process (aside from how x is chosen) remains the same. By the input indistinguishability of the SwHPS, these two experiments are indistinguishable even when given the MAC secret-key $K = (k_1, k_2)$ in full. Let $\mathsf{Win}', \mathsf{Match}'_j$ be the analogous events in the modified experiment. Since these events can be described as efficient predicates of the attacker's view and the MAC key K in the experiment, we have

$$\left| \Pr[\mathsf{Win} \wedge \mathsf{Match}_j] - \Pr[\mathsf{Win}' \wedge \mathsf{Match}'_j] \right| \leq \mathrm{negl}(\lambda)$$

Hence, to prove Eq. (5.2) it suffices to show $\Pr[\mathsf{Win}' \wedge \mathsf{Match}'_j] \leq \mathrm{negl}(\lambda)$ in the modified experiment. Assume that Match'_j occurs, so that the jth tagging query/response is $(\mathrm{M}, \sigma = (x, \psi, \tau))$ and the attacker's forgery attempt is $(\mathrm{M}, \sigma^* = (x, \psi^*, \tau^*))$ where the values M, x match. Let $k_3 = f_{k_2}(x)$. If $\psi = \psi^*$ also match between the two tags, then the forgery can only be accepting if $\tau^* = h_{k_3}(\psi) = \tau$ meaning that the tags $\sigma^* = \sigma$ must match entirely. In this case, the attacker automatically loses. Hence, if the event $\mathsf{Win}' \wedge \mathsf{Match}'$ occurs, we must also have $\psi \neq \psi^*$. By the *smoothness* of the SwHPS, the value k_3 is uniformly random given the entire view of the attacker during the game *aside* from the value $\tau = h_{k_3}(\psi)$ and the outputs of the leakage oracle. Therefore, the probability of $\mathsf{Win}' \wedge \mathsf{Match}'_j$ is at most the probability of breaking the security of the one-time MAC given ℓ bits of leakage on the (otherwise uniformly random) secret-key k_3. Since the one-time MAC is $2^{-u(\lambda)}$-secure without any leakage, it is also at least $2^{\ell(\lambda)-u(\lambda)}$ secure given ℓ bits of leakage (since we can always guess such leakage with probability $2^{-\ell(\lambda)}$). Therefore, we can upper-bound $\Pr[\mathsf{Win}' \wedge \mathsf{Match}'_j] \leq 2^{\ell(\lambda)-u(\lambda)} = \mathrm{negl}(\lambda)$. As we argued, this also proves Eq. (5.2).

Taking a union bound, Eq. (5.2) shows that $\Pr[\mathsf{Win} \wedge \mathsf{Match}] \leq \sum_{j=1}^{q} \Pr[\mathsf{Win} \wedge \mathsf{Match}_j] \leq \mathrm{negl}(\lambda)$. Combining this with Eq. (5.1), we get $\Pr[\mathsf{Win}] \leq \mathrm{negl}(\lambda)$ meaning that the attacker $\mathcal{A}$ has a negligible probability of winning the ℓ'-leakage-resilient SU-nvq-MAC security game against the MAC $(\mathsf{Tag}, \mathsf{Ver})$, as we wanted to show. □

Putting it All Together. We now combine the results from this section to state our main theorem for leakage-resilient message-authentication codes.

Theorem 5.6. *Assuming the existence of one-way functions, there exist $\ell(\lambda)$-leakage-resilient strongly unforgeable MACs for arbitrarily large polynomial $\ell(\lambda)$. Furthermore, assuming there exist standard PRFs with variable-length input size, output size λ, and key size $s(\lambda)$, the above schemes exist for any* leakage rate $\alpha(\lambda) = O\left(\frac{\log(\lambda)}{s(\lambda)}\right)$.

Proof. First, we use our construction of leakage-resilient nvq-MACs from PRFs (Theorem 5.4) with polynomial parameters $n = n(\lambda) > 4\lambda$ and $m = m(\lambda) > 4$ (assume m is a perfect power of 2). If the PRF key size is $s = s(\lambda)$, we get an ℓ-leakage-resilient nvq-MACs with leakage-bound $\ell = n\log(m) - \lambda > \frac{1}{2}n\log(m)$, key size $n(s + \log(m)) < 2ns$ and tag size $nm\lambda + \lambda$ (in bits).

We then use our construction for upgrading nvq-MAC security to SU-MAC security Theorem 5.5 using a wHPS and a one-time MAC constructed as follows:

- We use a one-time MAC with key size $2w$ and output size w and input size tw defined by

$$h_{r,s}(a_1, \ldots, a_t) := \sum_{i=1}^{t} a_i \cdot r^i + s$$

  where all operations are over the field $\mathbb{F}_{2^w}$. We set $w := n\log(m)$ and $t := m\lambda$. The security is given by $t/2^w$ which we can write as 2^{-u} for $u = w - \log(t) = n\log(m) - (\log(m) + \log(\lambda))$.
- We use an SwHPS with key size $n(s + \log(m)) < 2ns$ and output size $2n\log m$ (in bits) from the proof of Theorem 4.6, under the same PRF assumption.

Note that the output size of the wHPS matches the key size of the one-time MAC as needed. Also, the input size of the one-time MAC is

$$tw = (m\lambda)n\log(m) \geq 2nm\lambda \geq nm\lambda + \lambda$$

and therefore large enough so that we can feed it the tags of the nvq-MAC as inputs (as needed).

By Theorem 5.5, this construction is ℓ'-leakage-resilient for $\ell' = \min(\ell, u - \lambda) \geq n\log(m) - 2\lambda > \frac{1}{2}n\log(m)$ and has key size $< 4ns$. Therefore, the leakage rate is $\alpha > \log(m)/8s$. By choosing sufficiently large polynomials n, m, we can achieve the claim of the theorem, where n mainly influences the leakage amount ℓ' and m mainly influences the leakage rate α. □

6. Extensions

6.1. *Entropy-Bounded Leakage*

The notion of entropy-bounded leakage was first suggested by Naor and Segev [42] to capture scenarios where the *length* of the leakage obtained by the attacker is much longer than the length of the secret-key, but its relevant information content is still small. More specifically, in the model of *entropy-bounded leakage*, we restrict the amount of entropy that can be lost by seeing the output of the leakage function, rather than bounding its output size. There are several (similar but not equivalent) ways to model entropy-bounded leakage and the concept of entropy loss (see [9,11,16,42]). We follow [16] and consider the entropy loss over the uniform distribution as the measure of leakiness, since this measure turns out to be very robust and easy to use. We recall that we consider an average min-entropy notion, formally defined in Definition 2.2.

Definition 6.1. [16] A probabilistic function $h : \{0,1\}^* \to \{0,1\}^*$ is ℓ-*leaky* if, for all $n \in \mathbb{N}$, we have $\mathbf{H}_\infty(\mathcal{U}_n \mid h(\mathcal{U}_n)) \geq n - \ell$, where $\mathcal{U}_n$ is the uniform distribution over $\{0,1\}^n$.

When we define the ℓ-leakage-resilient security of various primitives in the entropy-bounded leakage model, the attacker can adaptively query the "leakage oracle" with arbitrary functions h_i, each of which is ℓ_i-leaky, as long as the total leakiness is bounded by $\sum_i \ell_i \leq \ell$.[7]

Notice that by Lemma 2.3, a length-bounded function $h : \{0,1\}^* \to \{0,1\}^\ell$ is ℓ-leaky. Thus, entropy-bounded leakage provides a generalization of length-bounded leakage. Furthermore, as was shown in [16], if a function is ℓ-leaky (decreases the entropy of the uniform distribution by at most ℓ bits), then it decreases the entropy of *every* distribution by at most ℓ bits. Moreover, the definition composes nicely and an adversary that adaptively chooses several ℓ_i-leaky functions, only learns $\sum_i \ell_i$ bits of information.

We claim that all of our results, both in the public-key and in the symmetric-key setting, hold if we consider entropy-bounded leakage instead of length-bounded leakage. This essentially follows immediately from our proofs of security, where we only used the entropy loss of the leakage to argue security.

6.2. *Bounded-Retrieval Model*

Motivation. The Bounded-Retrieval Model (BRM) [2,3,12,15,24] attempts to address the issue of *system compromise*, where an attacker can download large amounts of data from a compromised system storing cryptographic keys. In some cases, we may still be able to assume that the attacker is constrained, and the amount of data that he can download is bounded by some sufficiently huge bound ℓ (e.g., on the order of Gigabytes). This may be a reasonable assumption if the attacker's bandwidth is bounded, downloading more data is not a cost-effective attack, or the system can detect and prevent larger amounts of leakage. The main idea of the BRM is to use leakage resilience to maintain security in this scenario.

BRM Requirements. On a high level, the BRM requires *efficient* leakage-resilient cryptosystems that can tolerate *huge* amounts of leakage ℓ. In other words, the BRM places additional efficiency requirements on leakage-resilient schemes. If we want to instantiate a scheme so as to tolerate ℓ bits of leakage, then the secret-key size must be sufficiently large ($|\text{SK}| > \ell$) so that the attacker cannot leak the key in full. However, in the BRM, we insist that all other efficiency parameters of the scheme (the computation time of all algorithms, the public-key sizes, ciphertext sizes etc.) must remain small and essentially independent of ℓ. In other words, if the leakage-bound ℓ grows to the order of Gigabytes, the secret-key will need to grow as well, but the cryptographic scheme should otherwise remain efficient and usable. One outcome of this requirement is that schemes in the BRM cannot even access their entire (huge) secret-key during each cryptographic operation.

Prior Work on Encryption in the BRM. The work of Alwen et al. [2] shows how to obtain encryption schemes in the BRM from *identity-based weak hash-proof systems*

[7] Since we cannot efficiently measure the amount of "leakiness" of a function, the leakage oracle cannot efficiently verify the above condition. Instead, we simply insist that the attacker satisfies this condition and is agnostic to how this is ensured. In other words, we quantify over all attackers that satisfy the above condition.

(IB-wHPS).[8] Such schemes were then constructed under several concrete assumptions including LWE (+ random oracle), QR (+ random oracle), and the "q-truncated augmented bilinear Diffie-Hellman exponent (q-TABDHE) assumption." We now show how to achieve BRM encryption under general assumptions, by leveraging the ideas behind our constructions of public-key and symmetric-key wHPS from Sects. 3.2 and 4.2. First, let us give an overview of how IB-wHPS is used to construct encryption schemes in the BRM.

Random Sampling. The main idea behind all prior constructions in the BRM is to randomly sample some subset of the secret-key bits which will be relevant for the current cryptographic operation. That way, even if the attacker observed some leakage in the past, we are still likely to sample bits of the secret-key that have high entropy. We review this idea in more detail for the example of public-key encryption studied in [2] (a similar idea also applies to symmetric-key encryption).

Let $\mathcal{H}$ be some wHPS with a small secret-key (e.g., our basic construction in Sect. 3.2). We can use it to construct an efficient ℓ-leakage-resilient public-key encryption $\mathcal{E}$, where the bound ℓ and the secret-key can be made arbitrarily large while maintaining low computation cost. The public/secret-key pair of $\mathcal{E}$ consists of n independent public/secret keys of $\mathcal{H}$, i.e., $\mathrm{PK} = (\mathrm{PK}_1, \ldots, \mathrm{PK}_n)$, $\mathrm{SK} = (\mathrm{SK}_1, \ldots, \mathrm{SK}_n)$, where n is proportional to the leakage-bound ℓ. So far, this is the same as parallel repetition of $\mathcal{H}$. However, when we encrypt under $\mathcal{E}$, we now sample a random subset of $t \ll n$ indices $I \subseteq [n]$, $|I| = t$ and only compute $(c_i, k_i) \leftarrow \mathsf{wHPS.Encap}(\mathrm{PK}_i)$ for the indices $i \in I$. We then apply an extractor to the values $\{k_i\}_{i \in I}$ and use the output as a one-time-pad to encrypt an arbitrary message. Intuitively, when we switch from valid to invalid encapsulation, the collection of $\{k_i\}_{i \in I}$ is likely to have entropy even given the attacker's leakage. This happens because the choice of I is random and hence the attacker's leakage is unlikely to have been concentrated on the positions $\{\mathrm{SK}_i\}_{i \in I}$. This type of intuition is formalized in the work of Vadhan on locally computable extractors [50]. The computation time of encryption/decryption in $\mathcal{E}$ is only proportional to t, which is small and independent of ℓ. Unfortunately, the encryption scheme $\mathcal{E}$ still has a huge public-key (in the symmetric-key setting, a variant of the above idea using symmetric-key wHPS is already sufficient).

Reducing Public-Key Size. To reduce the public-key size, the work of [2] introduces a notion of identity-based wHPS (IB-wHPS). Informally, IB-wHPS is a generalization of wHPS, where the $\mathsf{wHPS.Encap}$, $\mathsf{wHPS.Encap}^*$ algorithms take a master public key MPK and an identity ID, and the $\mathsf{wHPS.Decap}$ algorithm takes in a secret-key $\mathrm{SK}_{\mathsf{ID}}$ corresponding to identity ID. See [2] for a formal definition. Using an IB-wHPS $\mathcal{H}$, we can construct an encryption scheme $\mathcal{E}$ in the BRM as follows. The public-key of $\mathcal{E}$ is set to the master public-key MPK of $\mathcal{H}$. The secret-key of $\mathcal{E}$ consists of n identity secret-keys $\mathrm{SK} = (\mathrm{SK}_{\mathsf{ID}_1}, \ldots, \mathrm{SK}_{\mathsf{ID}_n})$ corresponding to some n fixed identities. Encryption and decryption work essentially the same way as before. To encrypt, we first choose t random indices $I = (i_1, \ldots, i_t)$, and compute $(c_j, k_j) \leftarrow \mathsf{wHPS.Encap}(\mathrm{MPK}, \mathsf{ID}_{i_j})$ for the t relevant identities. We then apply an extractor with a random seed s to get

[8] Although that work does not use the term "weak" HPS, the abstraction there matches the natural extension of our notion of wHPS to the identity-based setting.

$r = \mathsf{Ext}((k_1, \ldots, k_t); s)$ and use r as a one-time-pad. In particular, the ciphertext for a message M is given by $C = (I, c_1, \ldots, c_t, r + \mathrm{M})$. See [2] for a formal description of IB-wHPS and an analysis of this abstract approach.

Our Observations. Firstly, we can directly plug in our construction of symmetric-key wHPS into the above framework to get symmetric-key encryption schemes in the BRM under general assumptions. Similarly, we can directly use our public-key wHPS to get a public-key encryption scheme in a relaxed version of the BRM, where the public-key size is large. To get a short public-key, we need to have an IB-wHPS. We notice that we can use our techniques to construct IB-wHPS generically from *any* identity-based encryption (IBE) scheme. Assume that the identities of the underlying IBE scheme have the form $\mathsf{ID} = (i, j)$ where $i, j \in \mathbb{N}$. The identities in our IB-wHPS scheme will have the form $\mathsf{ID} = i \in \mathbb{N}$ and the secret-key of the IB-wHPS for identity i will be a pair $(t, \mathrm{SK}_{i,t})$ where $t \leftarrow [m]$ is random and $\mathrm{SK}_{i,t}$ is an IBE secret-key for identity (i, t). Notice that the IB-wHPS secret-key for each identity i has at least $\log(m)$ bit of entropy depending on the choice of t. To encapsulate toward identity i, we create m IBE ciphertexts $(c_1, \ldots, c_m)$ toward the identities $(i, 1), \ldots (i, m)$, respectively. In a valid encapsulation, all ciphertexts encrypt the same random value $k \leftarrow [m]$, and in an invalid encapsulation, they encrypt different values, where c_i encrypts $k + i \pmod m$. The analysis showing that this scheme satisfies IB-wHPS essentially follows the proof of Theorem 3.4. Therefore, we get public-key encryption in the BRM assuming the existence of *any* standard IBE scheme. It is not known whether this assumption is minimal.

6.3. *After-the-Fact Leakage*

In this section, we consider the notion of *after-the-fact* leakage-resilient encryption defined by Halevi and Lin [31]. Loosely speaking, after-the-fact leakage-resilient security implies that an attacker who gets to observe ℓ_{POST} bits of leakage on the secret-key adaptively after seing the challenge ciphertext learns at most ℓ_{POST} bits of information about the plaintext (in contrast, standard leakage-resilient security of encryption implies that seing ℓ bits of leakage on the secret-key before seing the challenge ciphertext will not help reveal any information about the encrypted message). For simplicity, we will assume that the message M is picked at random from $\{0, 1\}^m$. We formulate the notion of after-the-fact leakage by defining two games: a *real* versus a *simulated* one. The real game has two phases of leakage: prior to the challenge phase and afterward. Therefore, the leakage oracle $\mathcal{O}_{\mathrm{SK}}^{\ell_{\mathrm{PRE}}, \ell_{\mathrm{POST}}}(\cdot)$ is parameterized by a secret-key SK, two leakage parameters $\ell_{\mathrm{PRE}}, \ell_{\mathrm{POST}}$, and a security parameter λ. Formally, we define the following two games:

The Real Game. Given the parameters $\lambda, \ell_{\mathrm{PRE}}, \ell_{\mathrm{POST}}$ and an encryption scheme $\Psi = (\mathsf{LR.Gen}, \mathsf{LR.Enc}, \mathsf{LR.Dec})$, the real game is defined as follows:

Key Generation: The challenger chooses at random a plaintext $M^{\mathrm{RL}} \leftarrow \{0, 1\}^m$. The challenger also runs $(\mathrm{PK}, \mathrm{SK}) \leftarrow \mathsf{LR.Gen}(1^\lambda)$ and gives PK to $\mathcal{A}$.

Pre-Challenge Leakage Queries:	$\mathcal{A}$ is given access to the leakage oracle $\mathcal{O}_{\mathrm{SK}}^{\ell_{\mathrm{PRE}},\ell_{\mathrm{POST}}}(\cdot)$. Note that we can assume without loss of generality that $\mathcal{A}$ queries $\mathcal{O}_{\mathrm{SK}}^{\ell_{\mathrm{PRE}},\ell_{\mathrm{POST}}}(\cdot)$ only once with a function f whose output is ℓ_{PRE} bits.
Challenge:	The challenger sends $c^* \leftarrow \mathsf{LR.Enc}(\mathrm{PK}, M^{\mathrm{RL}})$ to $\mathcal{A}$.
Post-Challenge Leakage Queries:	$\mathcal{A}$ is given access to the leakage oracle $\mathcal{O}_{\mathrm{SK}}^{\ell_{\mathrm{PRE}},\ell_{\mathrm{POST}}}(\cdot)$. Again, we can assume without loss of generality that $\mathcal{A}$ queries $\mathcal{O}_{\mathrm{SK}}^{\ell_{\mathrm{PRE}},\ell_{\mathrm{POST}}}(\cdot)$ only once with a function f whose output is ℓ_{POST} bits.

Denote by $\mathbf{View}_{\mathcal{A}}^{\mathrm{RL}}(\Psi)$ the random variable describing the view of the adversary $\mathcal{A}$ in the real game.

The Simulated Game. In the simulated game the challenger is replaced by a simulator $\mathcal{S}$ that gets a uniformly chosen plaintext M^{SM} as input and simulates the interaction with $\mathcal{A}$ conditioned on this plaintext. The view of $\mathcal{A}$ when interacting with $\mathcal{S}$ is denoted by $\mathbf{View}_{\mathcal{A},\mathcal{S}}^{\mathrm{SM}}(\Psi)$.

We continue with a definition of after-the-fact leakage-resilient encryption.

Definition 6.2. (*After-the-Fact Leakage-Resilient Encryption*) We say that an encryption scheme $\Psi = (\mathsf{LR.Gen}, \mathsf{LR.Enc}, \mathsf{LR.Dec})$ is $(\ell_{\mathrm{PRE}}, \ell_{\mathrm{POST}})$-*after-the-fact leakage resilient* if there exists a simulator $\mathcal{S}$, such that for all PPT adversaries $\mathcal{A}$ the following properties are satisfied:

- $(\mathbf{View}_{\mathcal{A}}^{\mathrm{RL}}(\Psi), M^{\mathrm{RL}}) \approx_c (\mathbf{View}_{\mathcal{A},\mathcal{S}}^{\mathrm{SM}}(\Psi), M^{\mathrm{SM}})$.
- $\mathbf{H}_{\infty}(M^{\mathrm{SM}} | \mathbf{View}_{\mathcal{A},\mathcal{S}}^{\mathrm{SM}}(\Psi)) \geq m - \ell_{\mathrm{POST}}$.

Constructions. Halevi and Lin [31] showed how to achieve *after-the-fact* leakage-resilient PKE from hash-proof systems. It is easy to see that their proof remains the same if we only have a *weak* hash-proof system (wHPS). The main idea is that the simulator uses *invalid* encapsulation algorithm so as to inject real (information-theoretic) entropy into the ciphertext. Moreover, we can also naturally define and construct after-the-fact leakage-resilient symmetric-key CPA secure encryption from symmetric-key wHPS using the same techniques.

7. 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 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] D. Agrawal, B. Archambeault, J.R. Rao, P. Rohatgi, The EM side-channel(s). in B.S. Kaliski Jr., Ç.K. Koç, C. Paar, editors, *CHES*, Lecture Notes in Computer Science, vol. 2523 (Springer, Berlin) pp. 29–45, August 13–15 2002

[2] J. Alwen, Y. Dodis, M. Naor, G. Segev, S. Walfish, D. Wichs, Public-key encryption in the bounded-retrieval model. in H. Gilbert, editor, *Advances in Cryptology—EUROCRYPT 2010*, Lecture Notes in Computer Science, vol. 6110 (Springer, Berlin, 2010), pp. 113–134

[3] J. Alwen, Y. Dodis, D. Wichs, Leakage-resilient public-key cryptography in the bounded-retrieval model. in *Advances in Cryptology (CRYPTO) 2009, 29th Annual International Cryptology Conference* (Santa Barbara, CA, 2009), pp. 36–54

[4] A. Akavia, S. Goldwasser, V. Vaikuntanathan, Simultaneous hardcore bits and cryptography against memory attacks. in O. Reingold, editor, *Sixth Theory of Cryptography Conference—TCC 2007*, Lecture Notes in Computer Science, vol. 5444 (Springer, Berlin, 2009)

[5] N. Bitansky, R. Canetti, S. Halevi, Leakage-tolerant interactive protocols. in *9th Theory of Cryptography Conference (TCC)* (Taormina, Sicily, 2012), pp. 266–284

[6] H. Bar-El, Known attacks against smartcards, 2003. Last accessed: August 26, 2009. http://www.hbarel.com/publications/Known_Attacks_Against_Smartcards.pdf

[7] Z. Brakerski, S. Goldwasser, Circular and leakage resilient public-key encryption under subgroup indistinguishability—(or: Quadratic residuosity strikes back). in T. Rabin, editor, *CRYPTO*, Lecture Notes in Computer Science, vol. 6223 (Springer, Berlin, 2010), pp. 1–20

[8] M. Braverman, A. Hassidim, Y.T. Kalai, Leaky pseudo-entropy functions. in B. Chazelle, editor, *ICS*, (Tsinghua University Press, 2011), pp. 353–366

[9] Z. Brakerski, Y.T. Kalai, A parallel repetition theorem for leakage resilience. in Cramer [15], pp. 248–265

[10] Z. Brakerski, J. Katz, Y. Kalai, V. Vaikuntanathan, Overcoming the hole in the bucket: Public-key cryptography against resilient to continual memory leakage. in *FOCS* [36], pp. 501–510

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

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

[13] S.S.M. Chow, Y. Dodis, Y. Rouselakis, B. Waters, Practical leakage-resilient identity-based encryption from simple assumptions. in E. Al-Shaer, A.D. Keromytis, V. Shmatikov, editors, *ACM Conference on Computer and Communications Security, (ACM, 2010)*, pp. 152–161

[14] R. Cramer, V. Shoup, Universal hash proofs and a paradigm for adaptive chosen ciphertext secure public-key encryption. in L. Knudsen, editor, *Advances in Cryptology—EUROCRYPT 2002*, LNCS, vol. 2332 (Springer, Berlin) pp. 45–64, 28 April–2 May 2002

[15] G. Di Crescenzo, R. J. Lipton, S. Walfish, Perfectly secure password protocols in the bounded retrieval model. in *Third Theory of Cryptography Conference (TCC)* (New York, NY, 2006), pp. 225–244

[16] Y. Dodis, K. Haralambiev, A. López-Alt, D. Wichs, Cryptography against continuous memory attacks. in *51th Annual (IEEE) Symposium on Foundations of Computer Science (FOCS)* (Las Vegas, NV, 2010), pp. 511–520

[17] Y. Dodis, K. Haralambiev, A. López-Alt, D. Wichs, Efficient public-key cryptography in the presence of key leakage. in M. Abe, editor, *ASIACRYPT*, Lecture Notes in Computer Science, vol. 6477 (Springer, Berlin, 2010), pp. 613–631

[18] Y. Dodis, E. Kiltz, K. Pietrzak, D. Wichs, Message authentication, revisited. in D. Pointcheval, T. Johansson, editor *EUROCRYPT*, Lecture Notes in Computer Science, vol. 7237, (Springer, Berlin 2012), pp. 355–374

[19] Y. Dodis, A.B. Lewko, B. Waters, D. Wichs, Storing secrets on continually leaky devices. in R. Ostrovsky, editor, *FOCS*, (IEEE, 2011), pp. 688–697

[20] I. Damgård, J.B. Nielsen, Improved non-committing encryption schemes based on a general complexity assumption. in M. Bellare, editor, *CRYPTO*, Lecture Notes in Computer Science, vol. 1880 (Springer, Berlin, 2000), pp. 432–450

[21] Y. Dodis, R. Ostrovsky, L. Reyzin, A. Smith, Fuzzy extractors: How to generate strong keys from biometrics and other noisy data. *SIAM Journal on Computing*, 38(1):97–139 (2008)
[22] S. Dziembowski, K. Pietrzak, Leakage-resilient cryptography. in *49th Symposium on Foundations of Computer Science, Philadelphia, PA, USA* (IEEE Computer Society) pp. 293–302, 25–28 October 2008
[23] Y. Dodis, Y. Yu, Overcoming weak expectations. in *ITW*, 2012. http://www.cs.nyu.edu/dodis/ps/weak-expe.pdf
[24] S. Dziembowski, Intrusion-resilience via the bounded-storage model. in *Third Theory of Cryptography Conference (TCC)* (New York, NY, 2006), pp. 207–224
[25] S. Dziembowski, Intrusion-resilience via the bounded-storage model. in Halevi and Rabin [34], pp. 207–224
[26] ECRYPT, Side channel cryptanalysis lounge. Last accessed: May 1, 2011. http://www.emsec.rub.de/research/projects/sclounge/
[27] O. Goldreich, S. Goldwasser, S. Micali, How to construct random functions. *Journal of the ACM*, 33(4):792–807, October 1986
[28] S. Goldwasser, Y.T. Kalai, C. Peikert, V. Vaikuntanathan, Robustness of the learning with errors assumption. in A.C.-C. Yao editor, *ICS*, (Tsinghua University Press, 2010), pp. 230–240
[29] S. Goldwasser, G.N. Rothblum, How to compute in the presence of leakage. *Electronic Colloquium on Computational Complexity (ECCC)*, 19:10, 2012
[30] J. Håstad, R. Impagliazzo, L.A. Levin, M. Luby, Construction of pseudorandom generator from any one-way function. *SIAM Journal on Computing*, 28(4):1364–1396, 1999
[31] S. Halevi, H. Lin, After-the-fact leakage in public-key encryption. in *8th Theory of Cryptography Conference (TCC)* (Providence, RI, 2011), pp. 107–124
[32] J.A. Halderman, S.D. Schoen, N. Heninger, W. Clarkson, W. Paul, J.A. Calandrino, A.J. Feldman, J. Appelbaum, E.W. Felten, Lest we remember: Cold-boot attacks on encryption keys. *Commun. ACM*, 52(5):91–98, 2009
[33] Y. Ishai, A. Sahai, D. Wagner, Private circuits: Securing hardware against probing attacks. in D. Boneh, editor, *Advances in Cryptology—CRYPTO 2003*, LNCS, vol. 2729 (Springer, Berlin, 2003)
[34] A. Jain, S. Garg, A. Sahai, Leakage-resilient zero knowledge. in *Advances in Cryptology–CRYPTO 2011, 31st Annual Cryptology Conference* (Santa Barbara, CA, 2011), pp. 297–315
[35] A. Jain, K. Pietrzak, Parallel repetition for leakage resilience amplification revisited. in *8th Theory of Cryptography Conference (TCC)* (Providence, RI, 2011), pp. 58–69
[36] P. Kocher, J. Jaffe, B. Jun, Differential power analysis. in M. Wiener, editor, *Advances in Cryptology—CRYPTO'99*, LNCS, vol. 1666, (Springer, Berlin) 15–19 August 1999, pp. 388–397
[37] P. Kocher, Timing attacks on implementations of Diffie-Hellman, RSA, DSS, and other systems. in N. Koblitz, editor, *Advances in Cryptology-CRYPTO 96*, LNCS, vol. 1109 (Springer, Berlin) 18–22 August 1996, pp. 104–113
[38] J. Katz, V. Vaikuntanathan, Signature schemes with bounded leakage resilience. in M. Matsui, editor, *Advances in Cryptology—ASIACRYPT 2009*, LNCS (Springer, Berlin, 2009), to appear
[39] A.B. Lewko, M. Lewko, B. Waters, How to leak on key updates. in *Proceedings of the 43rd ACM Symposium on Theory of Computing (STOC)* (San Jose, CA, 2011), pp. 725–734
[40] A. Lewko, B. Waters, On the insecurity of parallel repetition for leakage resilience. in *51th Annual IEEE Symposium on Foundations of Computer Science (FOCS)* (Las Vegas, NV, 2010), pp. 521–530
[41] S. Micali, L. Reyzin, Physically observable cryptography (extended abstract). in M. Naor, editor, *First Theory of Cryptography Conference—TCC 2004*, LNCS, vol. 2951 (Springer, Berlin), February 19–21 2004, pp. 278–296
[42] M. Naor, G. Segev, Public-key cryptosystems resilient to key leakage. in Halevi [31], pp. 18–35
[43] M. Naor, G. Segev, Public-key cryptosystems resilient to key leakage. *SIAM Journal on Computing*, 41(4):772–814, 2012. A preliminary version appeared in *Advances in Cryptology—CRYPTO'09*, pp. 18–35, 2009
[44] N. Nisan, D. Zuckerman, Randomness is linear in space. *Journal of Computer and System Sciences*, 52(1):43–53, 1996
[45] K. Pietrzak, A leakage-resilient mode of operation. in A. Joux, editor, *Advances in Cryptology—EUROCRYPT 2009*, LNCS, vol. 5479 (Springer, Berlin, 2009) pp. 462–482
[46] J.-J. Quisquater. F. Koene, 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

[47] J.-J. Quisquater, D. Samyde, Electromagnetic analysis (ema): Measures and counter-measures for smart cards. in I. Attali, T.P. Jensen, editors, *E-smart*, LNCS, vol. 2140 (Springer, Berlin), September 19–21 2001, pp. 200–210

[48] Reliable Computing Laboratory, Boston University. Side channel attacks database. http://www.sidechannelattacks.com. Last accessed: August 26, 2009

[49] F.-X. Standaert, How leaky is an extractor? in M. Abdalla, P.S.L.M. Barreto, editor, *LATINCRYPT*, Lecture Notes in Computer Science, vol. 6212 (Springer, Berlin, 2010), pp. 294–304

[50] S.P. Vadhan, On constructing locally computable extractors and cryptosystems in the bounded storage model. in D. Boneh, editor, *CRYPTO*, Lecture Notes in Computer Science, vol. 2729 (Springer, Berlin, 2003), pp. 61–77