Leakage-Resilient Cryptography from Puncturable Primitives and Obfuscation

Yu Chen[1,2,3], Yuyu Wang[4,5,6*], and Hong-Sheng Zhou[7]

[1] State Key Laboratory of Information Security, Institute of Information Engineering, Chinese Academy of Sciences, Beijing 100093, China
[2] State Key Laboratory of Cryptology, P.O. Box 5159, Beijing 100878, China
[3] School of Cyber Security, University of Chinese Academy of Sciences
chenyu@iie.ac.cn
[4] Tokyo Institute of Technology, Tokyo, Japan
[5] IOHK, Hong Kong, China
[6] National Institute of Advanced Industrial Science and Technology, Tokyo, Japan
wang.y.ar@m.titech.ac.jp
[7] Virginia Commonwealth University, Richmond, USA
hszhou@vcu.edu

Abstract. In this work, we develop a framework for building leakage-resilient cryptosystems in the bounded leakage model from puncturable primitives and indistinguishability obfuscation ($i\mathcal{O}$). The major insight of our work is that various types of puncturable pseudorandom functions (PRFs) can achieve leakage resilience on an obfuscated street.

First, we build leakage-resilient weak PRFs from weak puncturable PRFs and $i\mathcal{O}$, which readily imply leakage-resilient secret-key encryption. Then, we build leakage-resilient publicly evaluable PRFs (PEPRFs) from puncturable PEPRFs and $i\mathcal{O}$, which readily imply leakage-resilient key encapsulation mechanism and thus public-key encryption. As a building block of independent interest, we realize puncturable PEPRFs from either newly introduced puncturable objects such as puncturable trapdoor functions and puncturable extractable hash proof systems or existing puncturable PRFs with $i\mathcal{O}$. Finally, we construct the first leakage-resilient public-coin signature from selective puncturable PRFs, leakage-resilient one-way functions and $i\mathcal{O}$. This settles the open problem posed by Boyle, Segev, and Wichs (Eurocrypt 2011).

By further assuming the existence of lossy functions, all the above constructions achieve optimal leakage rate of $1 - o(1)$. Such a leakage rate is not known to be achievable for weak PRFs, PEPRFs and public-coin signatures before. This also resolves the open problem posed by Dachman-Soled, Gordon, Liu, O'Neill, and Zhou (PKC 2016, JOC 2018).

1 Introduction

A main line in cryptography is to design cryptosystems in security models that capture a wide range of possible attacks. Based on the idealized assumption that software/hardware implementations of cryptosystems perfectly hide the internal secrets, traditional security models (following the seminal work of Goldwasser

* Corresponding author

and Micali [GM84]) only give an adversary "black-box" access to cryptosystems. However, advancements of cryptanalysis indicate that such an idealized assumption is false in real world: an adversary can launch a variety of *key leakage attacks* (such as [Koc96, BDL97, BS97, KJJ99, HSH+08]) to get some partial information about secret keys.

To thwart key leakage attacks in a systematic manner, the research community has paid extensive efforts on the design of provably secure leakage-resilient cryptosystems in the last decade, spreading from basic primitives (including one-way functions, pseudorandom functions, message authentication codes, encryptions, and signatures) to advanced protocols (including identifications, authenticated key agreements, and zero-knowledge proof systems).

Leakage models. Briefly speaking, leakage models are defined by strengthening standard models with a leakage oracle $\mathcal{O}_{\mathsf{leak}}(\cdot)$, from which an adversary can (adaptively) specify a series of *leakage functions* $f_i : \{0,1\}^* \to \{0,1\}^{\ell_i}$ and learn the result of f_i applied to the internal secret state. Over the years, several leakage models have been proposed in the literature, differing in the specifications of f_i. In this work we focus on a simple yet general model called *bounded leakage* model, introduced by Akavia et al. [AGV09]. In the bounded leakage model, all secrets in memory are subject to leakage, i.e., the input of f_i could be entire secret key sk, while f_i could be arbitrary subjected to the natural restriction that $\sum_i \ell_i$ is bounded by some parameter ℓ, called the leakage bound. The leakage rate is defined as the ratio of ℓ to the secret key size $|sk|$, i.e., $\ell/|sk|$. Obviously, the optimal leakage rate is $1 - o(1)$ since otherwise the adversary can trivially learn the entire secret via querying $\mathcal{O}_{\mathsf{leak}}(\cdot)$.

To date, the bounded leakage model has been widely adopted in many works [NS09, KV09, CDRW10, BG10, GKPV10, HL11, BK12, BCH12]. The results from the bounded leakage model are usually used as building blocks for leakage-resilient schemes in more complex leakage models.

Approach towards leakage resilience. From the perspective of provable security, the main technical hurdle to achieve leakage-resilience is that the reduction must be able to handle leakage queries w.r.t. arbitrary functions chosen from $\mathcal{L}$, where $\mathcal{L}$ is the ensemble of admissible leakage functions. This seemingly stipulates that the reduction should know the secret key while typically this is not the case because the underlying intractable problems is usually embedded in the secret key. This intuition has been formalized as "useless attacker paradox" in [Wic13]. Prior works overcome this paradox by taking the following two approaches.

One approach is directly resorting to *leakage-resilient* assumptions (which might be well packed as advanced assumptions). Following this approach, the reduction can easily handle leakage queries by simply forwarding them to its own challenger. Goldwasser et al. [GKPV10] proved that the LWE assumption itself is leakage-resilient and then built a leakage-resilient secret-key encryption from it. Akavia et al. [AGV09] proved that meaningful and meaningless public keys are computationally indistinguishable even in the presence of se-

cret key leakage based on the LWE assumption, and then utilized this leakage-resilient "assumption" to show that Regev's PKE [Reg05] is actually leakage-resilient. Katz and Vaikuntanathan [KV09] built a leakage-resilient signature from universal one-way hash functions (UOWHFs)[8] together with PKE and simulation-sound non-interactive zero knowledge (NIZK) proof system, where the UOWHFs are actually used as leakage-resilient one-way functions. Similar strategy is also adopted for constructing other leakage-resilient signature schemes [DHLW10, BSW11, MTVY11].

Another approach is combining *key detached* strategy and *leakage-resilient* facts/assumptions, which is mainly used in the constructions of leakage-resilient PKE. Informally, the key detached strategy means the underlying intractable problems are not embedded to the secret keys, but to the ciphertexts. Following this approach, the reduction can easily handle key leakage queries by either owning the secret key or relying on leakage-resilient assumptions. Naor and Segev [NS09] utilized hash proof system (HPS) as a powerful tool to construct leakage-resilient PKE. In the security proof, valid ciphertexts are first switched to invalid ones (such switching is computationally indistinguishable even given the whole secret key because the underlying subset membership problem and secret keys are detached) to ensure that the hash proof π has high min-entropy, then the leftover hash lemma is used to prove the session key of the form $\mathsf{ext}(\pi, s)$ is random even in the presence of bounded key leakage.[9] Subsequently, Alwen et al. [ADN+10] and Hazay et al. [HLWW13] extended HPS to the identity-based and symmetric-key setting respectively, and used them to construct leakage-resilient identity-based encryption and secret-key encryption. Dodis et al. [DGK+10] constructed leakage-resilient PKE in the auxiliary input model via a similar method. In the security proof, valid ciphertexts are also first switched to invalid ones, then the generalized Goldreich-Levin theorem is used to argue that the session key of the form $\mathsf{hc}(sk)$ is pseudorandom even given auxiliary-input of the secret key sk.[10]

1.1 Motivation

So far, a broad range of leakage-resilient cryptographic schemes under various leakage models have been proposed in the literature. Nevertheless, several interesting problems are still left open around lower-level, "workhorse" primitives like SKE, PKE, and signature under the basic bounded leakage model.

For leakage-resilient SKE, the task can be reduced to constructing leakage-resilient weak PRFs (wPRFs) in the bounded leakage model. However, the lit-

[8] This is sometimes called second pre-image resistant functions.

[9] Leftover hash lemma could be interpreted as a leakage-resilient fact, which stipulates $\mathsf{ext}(x, s)$ is close to uniform even given a correlated value z, as long as s is a random seed chosen independently and x still has high min-entropy given leakage z.

[10] Goldreich-Levin theorem can be interpreted as a leakage-resilient assumption, which states that if h is one-way then $\mathsf{hc}(x)$ is pseudorandom even in the presence of $h(x)$. Here hc serves as a computational randomness extractor and $h(x)$ could be viewed as leakage on x.

erature on this topic is sparse. [Pie09, DY13] showed that any wPRF is already leakage-resilient for a logarithmic leakage bound $\ell = O(\log \lambda)$. Hazay et al. [HLWW13] built leakage-resilient wPRF from any one-way functions. Their construction only requires minimal assumption, but its leakage rate is $O(\log(\lambda)/|sk|)$, which is rather poor. To date, essentially nothing better was known for generic construction of leakage-resilient SKE with optimal leakage rate, beyond simply using leakage-resilient PKE in the symmetric-key setting.

For leakage-resilient PKE, existing constructions [AGV09, BG10, DGK+10, NS09, ADN+10] are based on either specific assumptions such as LWE, DDH, DCR, QR, or somewhat more generally the hash proof systems[11]. It is intriguing to know if there is a generic construction. In particular, whether the generic constructions of PKE based on trapdoor functions/relations [PW08, RS09, KMO10, Wee10] can be made leakage-resilient is still unclear. On the other hand, semantic security against chosen-ciphertext attacks (CCA) is the strongest notion for PKE in the traditional security model [GM84]. Several previous works [NS09, LWZ13, QL13, QL14, CQX18] studied how to achieve leakage-resilience and CCA security simultaneously via dedicated composition of separate techniques. Nevertheless, no prior work considered the orthogonal problem: whether we can acquire leakage-resilience from CCA security. We observe that in the CCA security experiment, responses to decryption queries can be viewed as a certain form of key leakage (the leakage function f is tied to decryption algorithm but with unbounded output length). It is interesting to know whether there is a general connection between the two important security notions for PKE.

For leakage-resilient signature, achieving *fully leakage-resilience* is of particular interest since it better captures real attacks [KV09]. This notion requires a signature to remain existentially unforgeable under chosen-message attacks even when an adversary obtains bounded leakage information on all intermediate states, including the secret keys and internal random coins. Clearly, if the signing procedure is deterministic or public-coin[12], standard leakage resilience automatically implies fully leakage resilience. To date, all the known fully leakage-resilient signature schemes [BSW11, MTVY11, LLW11, GJS11] in the standard model are randomized and secret-coin. The existence of leakage-resilient deterministic or public-coin signature is unclear and was left as an open problem by Boyle et al. [BSW11]. Earlier, the leakage-resilient signature scheme by Katz and Vaikuntanathan [KV09] is deterministic but only "one-time" secure. Recently, Wang et al. [WMHT16] proposed a leakage-resilient public-coin signature scheme. However, their construction is only secure against selective leakage attacks, i.e., an

[11] Following current conventions, we do not regard hash proof systems [CS02] as a general assumption.

[12] A signature is *secret-coin* if its security breaks down when the randomness used in the signing procedure is revealed. On the contrary, a signature is *public-coin* if it stays secure even when the random coins used in the signing procedure are revealed (i.e., provided in-the-clear by the signature). In other words, public-coin signature is secure even when the *entire* random coins used for signing are leaked.

adversary has to declare the leakage function before seeing the verification key. Besides, their construction requires differing-input obfuscation [BGI+12], whose existence is seriously cast in doubt [GGHW14, BSW16]. From this perspective, the problem posed by Boyle et al. [BSW11] is still largely open.

1.2 Our Contributions

With the preceding discussion in mind, in this work we focus on generic constructions of leakage-resilient encryption and signature in the bounded leakage model. The major insight of our work is that various kinds of puncturable PRFs can achieve leakage-resilience on an obfuscated street. We summarize our main results (depicted in Figure 1) as below.

Leakage-resilient SKE. As shown in [HLWW13], the classic construction of CPA-secure SKE from wPRF is leakage-resilience-preserving. So, we restrict our attention to constructing leakage-resilient wPRFs. Towards this goal, in Section 3.2 we first put forward a new notion called weak puncturable PRFs (wPPRFs), which could be thought of as the puncturable version of wPRFs. We then show wPPRFs and selective puncturable PRFs (sPPRFs) [SW14] imply each other, while the latter is implied by the GGM-tree based PRFs [GGM86]. Finally, in Section 3.3 we build leakage-resilient wPRFs from wPPRFs and $i\mathcal{O}$.

Leakage-resilient KEM. The KEM-DEM paradigm (here KEM stands for key encapsulation mechanism, DEM stands for data encapsulation mechanism) is a modular and efficient approach for building PKE. In the leakage setting, one can build a leakage-resilient PKE by combining a leakage-resilient KEM with a standard DEM. In the rest of this work, we only focus on the construction of leakage-resilient KEM. Chen and Zhang [CZ14] put forward the notion of publicly evaluable PRFs (PEPRFs), which encompasses almost all the known constructions of KEM. We observe that leakage-resilient PEPRFs naturally imply leakage-resilient KEM. So, the task is reduced to acquiring leakage resilience for PEPRFs.

To this end, in Section 4.2 we first put forward the notion of puncturable PEPRFs, then build leakage-resilient PEPRFs from puncturable PEPRFs and $i\mathcal{O}$ in Section 4.3. Moreover, we instantiate puncturable PEPRFs from either newly introduced primitives such as puncturable trapdoor functions and puncturable extractable hash proof systems, or existing puncturable PRFs with $i\mathcal{O}$.

This result provides a unified framework for constructing leakage-resilient KEM, which not only clarifies and encompasses the construction by Dachman-Soled et al. [DGL+16, Section 5.1], but also indicates that the PKE constructions based on "puncturable" trapdoor functions/relations (which in turn implied by correlated-product trapdoor functions [RS09] or extractable hash proof systems [Wee10] with puncturable property) can be made leakage resilient! Recently, Matsuda and Hanaoka [MH15] introduced a new primitive called puncturable KEM (PKEM), which captures a common pattern towards CCA security underlying many constructions of CCA-secure PKE. We remark that PPEPRFs

imply PKEM with perfect strong punctured decapsulation soundness. This result establishes a somewhat surprising connection between CCA security and leakage resilience, that is, CCA security obtained along the puncturable road can be converted to leakage-resilience in a non-black-box manner via obfuscation.

Leakage-resilient signature. In Section 5, we show how to build leakage-resilient signature from selective puncturable PRFs, $i\mathcal{O}$, and leakage-resilient one-way functions. Our basic scheme is deterministic but only achieves selective security[13]. To attain adaptive security, several bootstrapping techniques can be used without compromising leakage resilience. More precisely, one can either use the magic method enabled by extremely lossy function [Zha16], obtaining the first deterministic leakage-resilient signature scheme, or apply the "prefix-guessing technique" [HW09, RW14], yielding the first public-coin leakage-resilient signature scheme. We postpone the details to the full version [CWZ18].

We highlight that in our construction the signature size is exactly the output size of a puncturable PRF[14], which is very close to the leakage bound. Clearly, signature size cannot be shorter than leakage bound, since otherwise an adversary can directly obtain a forged signature from leakage. In this sense, our constructions also enjoy the almost optimal signature size.

All the basic constructions described above can tolerate L bits of leakage for any polynomial L of security parameter λ. However, the leakage rate is low due to the fact that secret keys are obfuscated programs, which could be very huge. By further assuming the existence of lossy functions [PW08], we can remarkably shrink the size of secret keys and achieve optimal leakage rate $1 - o(1)$. Such a leakage rate is not known to be achievable for weak PRFs, PEPRFs and deterministic/public-coin signatures before.

1.3 Overview of Our Techniques

As we summarized before, a common theme of the two main approaches towards leakage resilience in the literature is that the reduction always try to simulate leakage oracle *perfectly*, i.e., answering leakage queries with *real* leakage. To do so, we have to either rely on leakage-resilient assumptions or resort to sophisticated design with specific structure. It is interesting to investigate the possibility of simulating leakage oracle *computationally*, namely answering leakage queries with *simulated* leakage, as long as it is computationally indistinguishable from *real* leakage. This would possibly lend new techniques to address the unsolved problems in leakage-resilient cryptography.

Very recently, Dachman-Soled et al. [DGL+16] discovered powerful applications of $i\mathcal{O}$ to leakage-resilient cryptography. In the continual leakage model, they presented an $i\mathcal{O}$-based compiler that transforms any public-key encryption

[13] In selective security model, the adversary must declare the message m^* on which it will make a forgery before seeing the verification key, but then can adaptively make signing queries on messages distinct from m^*.

[14] In the case of our adaptively secure construction, a signature additionally contains a public coin of size λ^c for any constant $c < 1$.

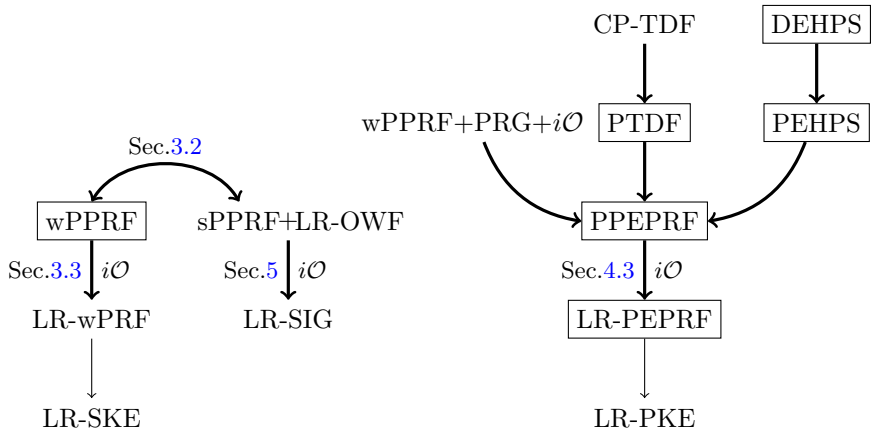

Fig. 1. The bold lines and rectangles denote our contributions (the thin lines denote those that are straightforward or follow readily from previous work).

or signature scheme with *consecutive* continual leakage-resilience to continual leakage resilience allowing leakage on key updates. In the bounded leakage model, they showed how to modify the Sahai-Waters PKE to be leakage-resilient. We observe that their work essentially embodies the idea of simulating leakage oracle computationally.

Simulate leakage via obfuscation. At the heart of our leakage-resilient encryptions and signatures is a general approach of simulating leakages enabled by puncturable primitives and obfuscation, which is largely inspired by the leakage-resilient variant of Sahai-Waters PKE due to Dachman-Soled et al. [DGL+16]. Next, we first distill and extend the idea underlying the work of [DGL+16], then carry out a systematic study of its applicability to leakage-resilient cryptography.

Recall that the common technical hurdle towards leakage resilience is to handle leakage queries. As opposed to the naive strategy of answering leakage queries with real secret keys, another promising strategy is simulating leakage with "faked" secret keys. By the composition lemma, as long as the faked secret keys are indistinguishable from the real ones, the simulated leakages are also indistinguishable from the real leakages because all leakage functions are efficiently computable.

Our approach adopts the second strategy. First, a secret key sk of any cryptographic scheme can always be expressed as a program Eval with sk hardwired. If a cryptographic scheme is puncturable (e.g., puncturable PRFs), then the reduction may build a functional-equivalent program Eval$'$ with sk_{x^*} and y^* hardwired, where sk_{x^*} is the punctured secret key at input x^* and $y^* = \text{Eval}(x^*)$. Secondly, note that indistinguishability obfuscation preserves functionality and guarantees that the obfuscations of any two functional-equivalent programs are computationally indistinguishable. Therefore, by setting the new secret key as $i\mathcal{O}(\text{Eval})$, the reduction is able to simulate leakage queries with $i\mathcal{O}(\text{Eval}')$. This

approach abstracts the high-level idea of how to acquire leakage-resilience when puncturable primitives meet $i\mathcal{O}$.

Obfuscate key and extract randomness. Our leakage-resilient encryptions exactly follow this approach. In a nutshell, we use $i\mathcal{O}$ to compile weak (resp. publicly evaluable) puncturable PRFs into leakage-resilient weak (resp. publicly evaluable) PRFs, which immediately yield leakage-resilient secret-key (resp. public-key) encryption.

To best illustrate our approach, we focus here on the secret-key setting, as it already emphasizes the main ideas underlying our approach. Starting from a weak puncturable PRF $F : K \times X \to \{0,1\}^n$, we use $i\mathcal{O}$ to compile it into a leakage-resilient weak PRF with a randomness extractor $\mathsf{ext} : \{0,1\}^n \times S \to Z$. The construction is instructive: (1) generate a secret key k for F, then create a program Eval with k hardwired, which on input x and s outputs $\mathsf{ext}(F_k(x), s)$; (2) set the secret key as $i\mathcal{O}(\text{Eval})$. This defines a weak PRF $\hat{F} : \{0,1\}^n \times S \to Z$. To establish security, the hybrid argument starts with the real game for leakage-resilient wPRF, where leakage and evaluation queries are handled with real secret key $i\mathcal{O}(\text{Eval})$. In the next game, the challenger picks the challenge input x^* and s^* at the very beginning, create a program Eval$'$ with the same input-output behavior as Eval, where k_{x^*} is the punctured key for k w.r.t. x^* and $y^* = F_k(x^*)$. The leakage and evaluation queries are thus handled with $i\mathcal{O}(\text{Eval}')$. Such modifications are undetectable by the security of $i\mathcal{O}$. In the final game, the challenger switches y^* from $F_k(x^*)$ to a random value. This transition is undetectable by the weak pseudorandomness of the starting puncturable PRF. An important fact is that the responses to evaluation queries are determined by k_{x^*}, and thus do not leak any information about y^*. Now, we can argue the desired security in purely information-theoretic way. By appropriate parameter choice, y^* still retains high min-entropy in the presence of leakage, and thus the value $\mathsf{ext}(y^*, s^*)$ is statistically close to uniform distribution.

In the public-key setting, our construction essentially follows the same approach. We use $i\mathcal{O}$ to compile puncturable PEPRF into leakage-resilient PEPRF, which readily yield leakage-resilient CPA-secure KEM. The main technical novelty lies in realizing puncturable PEPRFs from a variety of puncturable primitives. More precisely, we build puncturable PEPRFs from: (1) newly introduced notion of puncturable TDFs, which is in turn implied by correlated-product TDFs [RS09]; (2) newly introduced notion of puncturable EHPS, which is implied by EHPS [Wee10] satisfying derivable property; (3) selective puncturable PRFs, pseudorandom generator, and $i\mathcal{O}$ (adapted from the Sahai-Waters PKE [SW14]). This provides us a unified method to build leakage-resilient KEM from various puncturable primitives and $i\mathcal{O}$.

Obfuscate key and translate leakage. Along our approach towards leakage resilience, we investigate the possibility of building leakage-resilient signature from puncturable primitives and $i\mathcal{O}$. We choose the short "hash-and-sign" selectively secure signature by Sahai and Waters [SW14] as our starting point, since it inherits the puncturable property from its underlying selective puncturable

PRFs. To best illustrate the idea of our adaption, we first briefly review the Sahai-Waters signature scheme.

The Sahai-Waters signature is essentially a PRF-based MAC with public verifiability. The signing key sk is simply a secret key of selective puncturable PRF (sPPRF), and the signature on m is $\sigma \leftarrow F_k(m)$. The verification key vk is set as $i\mathcal{O}(\text{Vefy})$ where Vefy is a program that can check the MAC publicly. To excise out the information about $F_k(m^*)$ (here m^* denotes the target message), Vefy computes $g(F_k(m))$ and compares the result for equality to $g(\sigma)$, where g is a one-way function and σ is the claimed signature on m. To establish security, the hybrid argument starts with the real game for selective signature. The intermediate hybrid game builds an equivalent verification program using a punctured key k_{m^*} and $y^* \leftarrow g(\sigma^*)$ where $\sigma^* = F_k(m^*)$. The final hybrid game replaces σ^* with a random value. The first transition is undetectable by the security of $i\mathcal{O}$, while the second transition is undetectable by the pseudorandomness of sPPRF. In the final game, no PPT adversary is able to output a valid forgery (find the preimage σ^*) with non-negligible advantage by the one-wayness of g.

Following the new approach of simulating leakage, a tempting idea to make the Sahai-Waters signature leakage-resilient is setting the signing key as $i\mathcal{O}(\text{Sign})$, where Sign is a program that on input m outputs $F_k(m)$. Among the transitions of hybrid games, Sign is replaced by Sign$'$ (with k_{m^*} and σ^* hardwired). In this way, leakage and signing queries can be handled with "faked" signing key. However, we are unable to reduce the leakage-resilient unforgeability to the one-wayness of g. This is because in addition to $y^* = g(\sigma^*)$ revealed in vk, the information of σ^* may also be leaked via leakage queries on signing key $i\mathcal{O}(\text{Sign}')$. Therefore, the security proof breaks down in the final game, i.e., the reduction has to build Sign$'$ while σ^* is unknown.[15] We overcome this obstacle by using *leakage-resilient* OWF to replace standard OWF. Briefly, OWF is *leakage-resilient* if one-wayness remains in the presence of certain leakage on the preimage. Also observe that a leakage function f about the signing key $i\mathcal{O}(\text{Sign}')$ can be efficiently translated to leakage about σ^*, since both f and $i\mathcal{O}$ are efficiently computable. With such enhancement, in the final game the reduction can handle signing queries using k_{m^*} and handle leakage queries on signing key $i\mathcal{O}(\text{Sign}')$ by translating them to leakage queries on preimage σ^* to the underlying leakage-resilient OWF. See Section 5 for technical details.

Improving leakage rate via lossy functions. Applying the above approach in a straightforward manner will incur poor leakage rate, because the secret keys are obfuscated programs, which could be very large.

In [DGL+16], the authors showed how to modify their basic leakage-resilient PKE construction to achieve optimal leakage rate. Next, we briefly revisit their technique in the context of our construction of leakage-resilient wPRF. Now, the key generation algorithm works as follows: (1) pick a random key k_e for a SKE scheme and generate a dummy ciphertext $ct \leftarrow \mathsf{Enc}(k_e, 0^n)$ as the secret

[15] Note that this dilemma does not occur in the case of encryption, since the argument in the final game is information-theoretic.

key sk; (2) pick a collision-resistant hash h and compute $\eta^* \leftarrow h(ct)$; (3) pick a random key k for the underlying weak PRF, obfuscate a program Eval and store the obfuscated result C_{eval} into public parameters. Here, the program Eval is hardwired with k and t^*, which on input sk and (x, s) outputs $\mathsf{ext}(F_k(x), s)$ if and only if $h(sk) = \eta^*$. Intuitively, ct acts as a trigger of C_{eval}, which only works when $h(ct)$ matches η^*. In this way, the size of secret key is greatly reduced.

In the security proof, the first game is the real game. In the next game, ct is switched to an encryption of the PRF value $y^* \leftarrow F_k(x^*)$. This modification is undetectable by the semantic security of SKE. Then, C_{eval} is switched to C'_{eval}, which is an obfuscation of program Eval'. With k_e and a punctured PRF key k_{x^*} hardwired, Eval' works if and only if the hash value of its input ct matches η^*. When $h(ct) = t^*$, it evaluates with k_{x^*} if $x \neq x^*$, otherwise it evaluates after decrypting ct to y^*. In the final game, y^* is switched to a uniformly random value. The rest security analysis is routine. A subtle problem arised is that now Eval and Eval' have differing inputs, because h is compressing and thus a collision ct' (i.e., $h(ct') = \eta^* = h(ct)$) that encrypts a value $y' \neq y^*$ is likely to exist. Therefore, they have to rely on public-coin differing-input obfuscation [IPS15], which is stronger than indistinguishability obfuscation.

As analyzed above, the usage of CRHF leads to the reliance on differing-input obfuscation, while the choice of CRHF seems necessary to ensure that η^* only leaks partial information about y^* (encrypted in ct), which is crucial to achieve high leakage rate. Can we achieve higher leakage rate without resorting to differing-input obfuscation? The answer is affirmative. Our idea is to replace CRHFs with lossy functions [PW08]. In the real construction, h is generated as an injective function. By this choice, η^* uniquely fixes its preimage ct and thus the value y^*, With this setting, Eval and Eval' agree on all inputs, and $i\mathcal{O}$ suffices to guarantee the switching from Eval to Eval' is undetectable. To argue the high leakage rate we can attain, in the last game we switches h to a lossy function that significantly lose the information about y^*. By the security of lossy functions, this change is undetectable. Clearly, in the last game y^* still maintains sufficiently large min-entropy even in the presence of η^* and leakage. By appropriate choice of parameter, optimal leakage rate is achievable. The above technique carries over to the constructions of leakage-resilient PEPRF and signature as well.

We believe that the our technique of improving leakage rate by interplaying $i\mathcal{O}$ with lossy functions will also be instructive for avoiding using differing-input obfuscation in other places.

1.4 Related Work

Leakage models. Several leakage models have been proposed in the literature. In the seminal work, Micali and Reyzin [MR04] initiated the formal study of side-channel attacks by introducing the "only computation leaks information" model. Unfortunately, it fails to capture many practical leakage attacks, such as the cold-boot attack of [HSH+08].

To capture more general side-channel attacks known as memory attacks, Akavia et al. [AGV09] introduced the bounded leakage model, in which the ad-

versary can obtain arbitrary length-bounded leakage. The follow-up works considered various strengthens to accommodate more complex and general leakage scenarios. Naor and Segev [NS09] generalized the bounded leakage model to noisy leakage model (also known as entropy leakage model), where length-bounded leakage is relaxed to entropy-bounded leakage. Alwen et al. [ADW09, ADN+10] suggested the bounded-retrieval model, which imposes an additional requirement that the tolerated leakage amount can grow by proportionally expanding the secret key without increasing the size of public key, or computation/bandwidth efficiency. Dodis et al. [DHLAW10] and Brakerski et al. [BKKV10] introduced the continual leakage model for public-key schemes, where the secret key can be periodically self-refreshed while the public key remains the same. This model allows bounded leakage between any two successive refreshes without a-priori bound on the overall amount of leakage throughout the lifetime of the system.

The bottomline of the bounded leakage model and its variants interpret the following restriction on the leakage: it is information-theoretically impossible to recover the secret key from the leakage. Dodis et al. [DKL09, DGK+10] introduced the auxiliary input model (AIM), in which the total amount of leakage could be unbounded, as long as the secret key remains hard-to-invert given the leakage (but even if the secret key is fully determined in an information-theoretic sense). As noted in [KV09], a drawback of this model is that given some collection of leakage functions $\{f_i\}$ there is no way to tell, in general, whether they satisfy the stated requirement or not. Furthermore, existing constructions in this model require super-polynomial hardness assumptions.

Leakage-resilient cryptosystems. There is a large body of constructions of leakage-resilient cryptosystems in various models. In the bounded leakage model, there are OWF [KV09, Kom16], MAC and SKE [HLWW13], PKE [AGV09, NS09, LWZ13, QL13, QL14, CQX18], IBE [AGV09, ADN+10, CDRW10], signature [KV09, ADW09], AKE [ADW09], and zero-knowledge proofs [GJS11]. In the continual leakage model, there are PKE [DHLAW10, BKKV10], IBE [LRW11, YCZY12, YXZ+15], and signature [BSW11, MTVY11, LLW11]. In the auxiliary input model, there are SKE [DKL09], PKE [DGK+10], and signature [WMHT16].

2 Preliminaries

Notation. For a distribution or random variable X, we write $x \xleftarrow{R} X$ to denote the operation of sampling a random x according to X. For a set X, we use $x \xleftarrow{R} X$ to denote the operation of sampling x uniformly at random from X, and use $|X|$ to denote its size. We use U_X to denote the uniform distribution over X. For a positive integer n, we use $[n]$ to denote the set $\{1, \ldots, n\}$. Unless described otherwise, all quantities are implicitly functions of a security parameter denoted λ. We say that a quantity is negligible, written $\mathsf{negl}(\lambda)$, if it vanishes faster than the inverse of any polynomial in λ. A probabilistic polynomial time (PPT) algorithm is a randomized algorithm that runs in time $\mathsf{poly}(\lambda)$. If $\mathcal{A}$ is a randomized algorithm, we write $z \leftarrow \mathcal{A}(x_1, \ldots, x_n; r)$ to indicate that $\mathcal{A}$ outputs

z on inputs $(x_1, \ldots, x_n)$ and random coins r. For notational clarity we usually omit r and write $z \leftarrow \mathcal{A}(x_1, \ldots, x_n)$.

Due to space limitations, we postpone the background of randomness extraction, definitions of lossy functions, leakage-resilient one-way functions/symmetric encryption/key encapsulation mechanism/signature to the full version [CWZ18].

2.1 Puncturable Pseudorandom Functions

Puncturable PRFs (PPRFs) is the simplest type of constrained PRFs [KPTZ13, BW13, BGI14]. In a PPRF, the constrained key is associated with an element $x^* \in X$, which allows evaluation on all elements $x \neq x^*$. Next, we recall the definition and security notion of PPRFs from [SW14] as below.

Definition 1 (PPRFs). *A PPRF $F : K \times X \to Y$ consists of four polynomial time algorithms:*

- $\mathsf{Gen}(\lambda)$*: on input λ, output public parameter pp and a secret key $k \xleftarrow{R} K$. pp will be used as an implicit input of* PrivEval*,* Puncture *and* PuncEval*.*
- $\mathsf{PrivEval}(k, x)$*: on input a secret key k and $x \in X$, output $F(k, x)$.*
- $\mathsf{Puncture}(k, x^*)$*: on input a secret key k and $x^* \in X$, output a punctured key $k(\{x^*\})$.*[16]
- $\mathsf{PuncEval}(k_{x^*}, x)$*: on input a punctured key k_{x^*} and an element $x \in X$, output $F(k, x)$ if $x \neq x^*$ and a special reject symbol $\perp$ otherwise.*

For ease of notation, we write k_{x^*} to represent $k(\{x^*\})$, write $F_k(x)$ and $F(k, x)$ interchangeably and write $F_{k_{x^*}}(x)$ or $F(k_{x^*}, x)$ to represent $\mathsf{PuncEval}(k_{x^*}, x)$*.*

Sahai and Waters [SW14] defined selective pseudorandomness for PPRFs, which is weaker than full pseudorandomness in that the adversary must commit to the target input x^* even before seeing the public parameter.

Selective pseudorandomness. Let $\mathcal{A} = (\mathcal{A}_1, \mathcal{A}_2)$ be an adversary against PPRFs and define its advantage in the following experiment:

$$\mathsf{Adv}_{\mathcal{A}}(\lambda) = \Pr\left[\beta = \beta' : \begin{array}{l} (state, x^*) \leftarrow \mathcal{A}_1(\lambda); \\ (pp, k) \leftarrow \mathsf{Gen}(\lambda); \\ k_{x^*} \leftarrow \mathsf{Puncture}(k, x^*); \\ y_0^* \leftarrow F_k(x^*), y_1^* \xleftarrow{R} Y; \\ \beta \xleftarrow{R} \{0, 1\}; \\ \beta' \leftarrow \mathcal{A}_2(state, pp, k_{x^*}, y_\beta^*); \end{array}\right] - \frac{1}{2}.$$

A PPRF is said to be selectively pseudorandom if for any PPT adversary $\mathcal{A}$ its advantage defined as above is negligible in λ. For simplicity, we refer to selectively pseudorandom PPRFs as sPPRFs. sPPRFs with fixed-length domain are easily obtained from the GGM tree-based PRFs [GGM86], as observed in [BW13, BGI14, KPTZ13]. Ramchen and Waters [RW14] also showed the existence of sPPRFs with variable-length domain.

[16] Without loss of generality, we assume that $k(\{x^*\})$ includes the information of x^* in plain.

2.2 Indistinguishability Obfuscation for Circuits

We recall the definition and security notion of indistinguishability obfuscator from [GGH+13] as below.

Definition 2 (Indistinguishability Obfuscator ($i\mathcal{O}$)). *A uniform PPT machine $i\mathcal{O}$ is called an indistinguishability obfuscator for a circuit class $\{\mathcal{C}_\lambda\}$ if the following conditions are satisfied:*

- *(Preserving Functionality) For all security parameter $\lambda \in \mathbb{N}$, for all $C \in \mathcal{C}_\lambda$, and for all inputs $x \in \{0,1\}^*$, we have:*

$$\Pr[C'(x) = C(x) : C' \leftarrow i\mathcal{O}(\lambda, C)] = 1$$

- *(Indistinguishability of Obfuscation) For any PPT adversaries $(\mathcal{S}, \mathcal{D})$, there exists a negligible function α such that the following holds: if $\Pr[\forall x, C_0(x) = C_1(x) : (C_0, C_1, aux) \leftarrow \mathcal{S}(\lambda)] \geq 1 - \alpha(\lambda)$, then we have:*

$$|\Pr[\mathcal{D}(aux, i\mathcal{O}(\lambda, C_0)) = 1] - \Pr[\mathcal{D}(aux, i\mathcal{O}(\lambda, C_1)) = 1]| \leq \alpha(\lambda)$$

3 Leakage-Resilient SKE

We begin this section by recalling the notion of leakage-resilient wPRFs and their application in building leakage-resilient CPA-secure SKE from [HLWW13]. We then introduce a new notion called weak puncturable PRFs (weak PPRFs), and show how to compile weak PPRFs to leakage-resilient wPRFs via $i\mathcal{O}$.

3.1 Leakage-Resilient Weak PRFs

Standard PRFs require full pseudorandomness: given polynomially many *arbitrarily* inputs $x_1, \ldots, x_q$, the outputs $F_k(x_1), \ldots, F_k(x_q)$ look pseudorandom. Sometimes, the full power of PRFs is not needed and it is sufficient to have weak PRFs which only claim weak pseudorandomness, where pseudorandomness holds for *uniformly random* choice of inputs $\{x_i\}$. The corresponding leakage-resilient notion requires that weak pseudorandomness holds even if the adversary can learn some leakage about the secret key k. Now, we recall the formal definition of leakage-resilient weak pseudorandomness from [HLWW13].

Leakage-resilient weak pseudorandomness. Let $\mathcal{A} = (\mathcal{A}_1, \mathcal{A}_2)$ be a PPT adversary against PRFs and define its advantage in the following experiment.

$$\mathsf{Adv}_{\mathcal{A}}(\lambda) = \Pr\left[\beta = \beta' : \begin{array}{l} (pp, k) \leftarrow \mathsf{Gen}(\lambda); \\ state \leftarrow \mathcal{A}_1^{\mathcal{O}_{\mathsf{leak}}(\cdot), \mathcal{O}_{\mathsf{eval}}(\$)}(pp); \\ x^* \xleftarrow{R} X; \\ y_0^* \leftarrow F_k(x^*), y_1^* \xleftarrow{R} Y; \\ \beta \xleftarrow{R} \{0,1\}; \\ \beta' \leftarrow \mathcal{A}_2^{\mathcal{O}_{\mathsf{eval}}(\$)}(state, x^*, y_\beta^*); \end{array}\right] - \frac{1}{2}.$$

Here $\mathcal{O}_{\mathsf{leak}}(\cdot)$ is a leakage oracle that on input leakage function $f : K \to \{0,1\}^*$ returns $f(k)$, subjected to the restriction that the sum of its output lengths is at most ℓ. $\mathcal{O}_{\mathsf{eval}}(\$)$ is an evaluation oracle that does not take any input and on each invocation, chooses a freshly random $x \in X$ and outputs $(x, F_k(x))$. A PRF is ℓ-leakage-resilient weakly pseudorandom if no PPT adversary has non-negligible advantage in the above experiment.

Remark 1. As pointed out in [HLWW13], since the adversary can always learn a few bits of $F_k(x)$ for some x of its choice (via leakage query), we cannot hope to achieve full pseudorandomness in the presence of leakage, and hence setting for weak pseudorandomness is a natural choice.

Leakage-resilient SKE. The construction of LR CPA-secure SKE from LR wPRF is obvious. We sketch the construction from [HLWW13] for completeness. Assume $F : K \times X \to Y$ is a leakage-resilient wPRF, whose range Y is an additive group (e.g., bit-strings under XOR). The secret key is exactly the key of the underlying wPRF. To encrypt a message $m \in Y$, one samples $x \xleftarrow{R} X$ and outputs the ciphertext $(x, F_k(x) + m)$. The decryption process is obvious. The desired LR CPA security of SKE follows readily from the LR weak pseudorandomness of the wPRF.

3.2 Weak Puncturable PRFs

Towards the construction of leakage-resilient wPRFs, we put forward a new notion called weak PPRFs by introducing weak pseudorandomness for PPRFs. We show that weak PPRFs and selective PPRFs imply each other, while the latter is directly implied by the GGM-tree based PRFs [GGM86].

Next, we formally introduce weak pseudorandomness for PPRFs, which differs from selective pseudorandomness (cf. definition in Section 2.1) in that the target input x^* is uniformly chosen by the challenger, rather than being arbitrarily chosen by the adversary before seeing the public parameter.

Weak pseudorandomness. Let $\mathcal{A} = (\mathcal{A}_1, \mathcal{A}_2)$ be an adversary against PPRFs and define its advantage in the following experiment:

$$\mathsf{Adv}_{\mathcal{A}}(\lambda) = \Pr\left[\beta = \beta' : \begin{array}{l} (pp, k) \leftarrow \mathsf{Gen}(\lambda); \\ x^* \xleftarrow{R} X; \\ k_{x^*} \leftarrow \mathsf{Puncture}(k, x^*); \\ y_0^* \leftarrow F_k(x^*), y_1^* \xleftarrow{R} Y; \\ \beta \xleftarrow{R} \{0,1\}; \\ \beta' \leftarrow \mathcal{A}(pp, x^*, k_{x^*}, y_\beta^*); \end{array}\right] - \frac{1}{2}.$$

A PPRF is weakly pseudorandom if no PPT adversary has non-negligible advantage in the above experiment. For simplicity, we refer to weakly pseudorandom PPRFs as wPPRFs.

Interestingly, we show that wPPRFs and sPPRFs imply each other.

Theorem 1. *wPPRFs and sPPRFs imply each other.*

Proof. We first show that "wPPRFs imply sPPRFs" by building sPPRFs from wPPRFs. Let $F : K \times X \to Y$ be a wPPRF, we build a sPPRF $\hat{F} : K \times X \to Y$ from F as below.

- $\mathsf{Gen}(\lambda)$: run $(pp, k) \leftarrow F.\mathsf{Gen}(\lambda)$, pick $r^* \xleftarrow{R} X$, set $\hat{pp} = (pp, r^*)$ and k as the secret key.
- $\mathsf{PrivEval}(k, x)$: on input k and x, output $\hat{y} \leftarrow F_k(x + r^*)$ via computing $F.\mathsf{PrivEval}(k, x + r^*)$. This algorithm defines $\hat{F}_k(x) := F_k(x + r^*)$.
- $\mathsf{Puncture}(k, x^*)$: compute $k_{x^*+r^*} \leftarrow F.\mathsf{Puncture}(k, x^* + r^*)$, output $\hat{k}_{x^*} = k_{x^*+r^*}$.
- $\mathsf{PuncEval}(\hat{k}_{x^*}, x)$: parse $\hat{k}_{x^*}$ as $k_{x^*+r^*}$, if $x \neq x^*$ output $y \leftarrow F_{k_{x^*+r^*}}(x + r^*)$ via computing $F.\mathsf{PuncEval}(k_{x^*+r^*}, x + r^*)$, else output $\perp$.

We now reduce the selective pseudorandomness of the above construction to the weak pseudorandomness of the underlying wPPRF. Let $\mathcal{A}$ be an adversary against sPPRF with advantage $\mathsf{Adv}_{\mathcal{A}}(\lambda)$, we build an adversary $\mathcal{B}$ that breaks wPPRF with the same advantage. $\mathcal{B}$ interacts with $\mathcal{A}$ in the selective pseudorandomness experiment of sPPRF as below:

1. Commit: $\mathcal{A}$ submits its target input $\hat{x^*}$.
2. Setup and Challenge: $\mathcal{B}$ invokes its wPPRF challenger and receives back the wPPRF challenge instance $(pp, k_{x^*}, x^*, y^*_\beta)$ where x^* is randomly chosen from X, y^*_β is either $F_k(x^*)$ if $\beta = 0$ or randomly chosen from Y if $\beta = 1$. $\mathcal{B}$ then sets $r^* = x^* - \hat{x^*}$, $\hat{pp} = (pp, r^*)$, $\hat{k}_{\hat{x^*}} = k_{x^*}$, sends $(\hat{pp}, \hat{k}_{\hat{x^*}}, y^*_\beta)$ to $\mathcal{A}$ as the sPPRF challenge.
3. Guess: $\mathcal{A}$ outputs its guess β' for β and $\mathcal{B}$ forwards β' to its own challenger.

Note that x^* is distributed uniformly at random over X, thereby so is r^*. According to the construction, the punctured key $\hat{k}_{\hat{x^*}}$ at point $\hat{x^*}$ in sPPRF $\hat{F}$ equals the punctured key $k_{\hat{x^*}+r^*} = k_{x^*}$ at point x^* in wPPRF F. Therefore, $\mathcal{B}$'s simulation is perfect and has the same advantage as $\mathcal{A}$. This proves the forward implication.

The reverse direction that "sPPRFs imply wPPRFs" follows by a simple reduction of weak pseudorandomness to selective pseudorandomness. Let $\mathcal{A}$ be an adversary against wPPRF with advantage $\mathsf{Adv}_{\mathcal{A}}(\lambda)$, we build an adversary $\mathcal{B}$ that breaks sPPRF with the same advantage. $\mathcal{B}$ interacts with $\mathcal{A}$ in the weak pseudorandomness experiment of wPPRF as below:

1. Setup and Challenge: $\mathcal{B}$ picks $x^* \xleftarrow{R} X$ and submits x^* to its own sPPRF challenger. Upon receiving back (pp, k_{x^*}, y^*_β) where y_β is either $F_k(x^*)$ if $\beta = 0$ or randomly chosen from Y if $\beta = 1$, $\mathcal{B}$ sends $(pp, x^*, k_{x^*}, y^*_\beta)$ to $\mathcal{A}$ as the wPPRF challenge.
2. Guess: $\mathcal{A}$ outputs its guess β' for β and $\mathcal{B}$ forwards β' to its own challenger.

Note that x^* is distributed uniformly over X. Therefore, $\mathcal{B}$'s simulation is perfect and has the same advantage as $\mathcal{A}$. This proves the inverse implication.

The theorem immediately follows.

3.3 Leakage-Resilient wPRFs from wPPRFs and $i\mathcal{O}$

Now, we show how to construct leakage-resilient wPRFs from wPPRFs and $i\mathcal{O}$. Let $F : K \times X \to Y$ be a wPPRF, $i\mathcal{O}$ be an indistinguishability obfuscator, and $\mathsf{ext} : Y \times S \to Z$ be an average-case (n, ϵ)-strong extractor. In what follows, we build a LR wPRF $\hat{F} : \hat{K} \times \hat{X} \to Z$, where $\hat{X} = X \times S$.

- $\mathsf{Gen}(\lambda)$: run $(pp, k) \leftarrow F.\mathsf{Gen}(\lambda)$, output pp and $\hat{k} \leftarrow i\mathcal{O}(\text{PrivEval})$, where PrivEval is the program defined in Figure 2.
- $\mathsf{PrivEval}(\hat{k}, \hat{x})$: on input $\hat{k}$ and $\hat{x} = (x, s) \in X \times S$, output $y \leftarrow \hat{k}(x, s)$. This algorithm implicitly defines $\hat{F}_{\hat{k}}(\hat{x}) := \mathsf{ext}(F_k(x), s)$.

PrivEval

Constants: wPPRF key k
Input: $\hat{x} = (x, s)$

1. Output $z \leftarrow \mathsf{ext}(F_k(x), s)$.

Fig. 2. Program PrivEval. This program is appropriately padded to the maximum of the size of itself and program PrivEval* defined in Figure 3.

PrivEval*

Constants: wPPRF punctured key k_{x^*}, x^*, y^*
Input: $\hat{x} = (x, s)$

1. If $x = x^*$, output $z \leftarrow \mathsf{ext}(y^*, s)$.
2. Else, output $z \leftarrow \mathsf{ext}(F_{k_{x^*}}(x), s)$.

Fig. 3. Program PrivEval*

Theorem 2. *If F is a secure wPPRF, $i\mathcal{O}$ is indistinguishably secure, ext is an average-case (n, ϵ)-strong extractor, the above construction is a ℓ-LR wPRF as long as $\ell \leq \log |Y| - n$.*

Proof. We proceed via a sequence of games. Let S_i be the event that $\mathcal{A}$ wins in Game i.

Game 0. This game is the standard leakage-resilient weak pseudorandomness game for wPRFs. $\mathcal{CH}$ interacts with $\mathcal{A}$ as below:

Setup: $\mathcal{CH}$ runs $(pp, k) \leftarrow F.\mathsf{Gen}(\lambda)$, creates $\hat{k} \leftarrow i\mathcal{O}(\text{PrivEval})$, where the program PrivEval is defined in Figure 2. $\mathcal{CH}$ then sends pp to $\mathcal{A}$.
Phase 1: $\mathcal{A}$ can make evaluation queries and leakage queries. For each evaluation query, $\mathcal{CH}$ chooses $x \xleftarrow{R} X$ and $s \xleftarrow{R} S$ and returns $(x, s, \hat{k}(x, s))$. For each leakage query $\langle f \rangle$, $\mathcal{CH}$ responds with $f(\hat{k})$.

Challenge: $\mathcal{CH}$ chooses $x^* \xleftarrow{R} X$, $s^* \xleftarrow{R} S$ and computes $y^* \leftarrow F_k(x^*)$, then computes $z_0^* \leftarrow \mathsf{ext}(y^*, s^*)$, picks $z_1^* \xleftarrow{R} Z$ and $\beta \xleftarrow{R} \{0,1\}$, sends z_β^* to $\mathcal{A}$.
Phase 2: $\mathcal{A}$ continues to make evaluation queries. $\mathcal{CH}$ responds the same way as in Phase 1.
Guess: $\mathcal{A}$ outputs its guess β' for β and wins if $\beta' = \beta$.

According to the definition, we have:

$$\mathsf{Adv}_{\mathcal{A}}(\lambda) = |\Pr[S_0] - 1/2|$$

Game 1. Same as Game 0 except that $\mathcal{CH}$ chooses $x^* \xleftarrow{R} X$, $s^* \xleftarrow{R} S$ and computes $y^* \leftarrow F_k(x^*)$ in the Setup stage. This change is only conceptual and thus we have:

$$\Pr[S_1] = \Pr[S_0]$$

Game 2. Same as Game 1 except that $\mathcal{CH}$ directly aborts when handling evaluation queries for $x = x^*$.

Let E be the event that there exists one random sample x that equals x^* when $\mathcal{CH}$ emulates evaluation oracle. Clearly, if E never happens, then Game 1 and Game 2 are identical. Suppose $\mathcal{A}$ makes at most q_e evaluation queries. Since $\mathcal{A}$ is a PPT adversary, q_e is bounded by a polynomial in λ. Therefore, $\Pr[E] \leq q_e/|X| \leq \mathsf{negl}(\lambda)$, we have:

$$|\Pr[S_2] - \Pr[S_1]| \leq \Pr[E] \leq \mathsf{negl}(\lambda)$$

Game 3. Same as Game 2 except that $\mathcal{CH}$ computes $k_{x^*} \leftarrow F.\mathsf{Puncture}(k, x^*)$, $y^* \leftarrow F_k(x^*)$, and creates $\hat{k} \leftarrow i\mathcal{O}(\mathrm{PrivEval}^*)$ in the Setup stage. Here, the program $\mathrm{PrivEval}^*$ (defined in Figure 3) is built from constants k_{x^*}, x^*, y^*.

By the correctness of wPPRFs, the two programs PrivEval and $\mathrm{PrivEval}^*$ agree on all inputs. By the security of $i\mathcal{O}$, we have:

$$|\Pr[S_3] - \Pr[S_2]| \leq \mathsf{Adv}_{\mathcal{A}}^{i\mathcal{O}}$$

Game 4. Same as Game 3 except that $\mathcal{CH}$ picks $y^* \xleftarrow{R} Y$ rather than setting $y^* \leftarrow F_k(x^*)$ in the Setup stage.

By a simple reduction to the weak pseudorandomness of wPPRFs, this modification is undetectable for all PPT adversaries. Thus, we have:

$$|\Pr[S_4] - \Pr[S_3]| \leq \mathsf{Adv}_{\mathcal{A}}^{\mathrm{wPPRF}}$$

Game 5. Same as Game 4 except that $\mathcal{CH}$ picks $z_0^* \xleftarrow{R} Z$ rather than setting $z_0^* \leftarrow \mathsf{ext}(y^*, s^*)$ in the Challenge stage.

We denote by V the set of public parameter pp, (x^*, s^*), the responses to all evaluation queries (determined by k_{x^*}), z_1^* and β. In both Game 4 and Game 5, y^* is uniformly chosen from Y (independent of V), thus $\mathsf{H}_\infty(y^*|V) = \log|Y|$. Observe that $\mathcal{A}$ also obtains at most ℓ bits leakage on $\hat{k}$ (denote by $leak$) which is correlated to y^*, it follows by the chain rule that $\tilde{\mathsf{H}}_\infty(y^*|(V, leak)) \geq \mathsf{H}_\infty(y^*|V) -$

$\ell = \log |Y| - \ell$. Since ext is an average-case (n, ϵ)-strong extractor, we conclude that $\mathsf{ext}(y^*, s^*)$ is ϵ-close to a uniformly random $z_0^* \xleftarrow{R} Z$, even given V and leakage. Note that $\mathcal{A}$'s view in Game 4 and Game 5 is fully determined by z_0^*, V and $leak$, while V and $leak$ are distributed identically in Game 4 and Game 5. Thereby, $\mathcal{A}$'s view in Game 4 and Game 5 are $\epsilon/2$-close. Thus, we have:

$$|\Pr[S_5] - \Pr[S_4]| \leq \epsilon/2 \leq \mathsf{negl}(\lambda)$$

In Game 5, both z_0^* and z_1^* are randomly chosen from Z. Therefore, we have:

$$\Pr[S_5] = 1/2$$

Putting all the above together, the theorem immediately follows. □

We have sketched how to achieve optimal leakage rate in Section 1.3. To avoid repetition, we omit the details here.

Comparison with prior constructions. [Pie09, DY13] showed that any wPRF is already leakage-resilient for a logarithmic leakage bound $\ell = O(\log \lambda)$. Hazay et al. [HLWW13] showed a black-box construction of LR wPRF from any wPRF $F : K \times X \to Y$. Their construction takes two steps: (1) construct symmetric-key weak HPS from wPRF; (2) build LR wPRF by parallel repetition of symmetric-key weak HPS. The consequence is that it is not flexible and efficient. To make the output size larger than $n \log |Y|$, they have to invoke n independent copies of the basic wPRF, and the domain size must be larger than $n(|X| + \log |Y|)$. Besides, its leakage rate is rather poor, say, $O(\log(\lambda)/|k|)$. In contrast, our construction enjoys flexible parameter choice and optimal leakage rate, which is benefited from the non-black-box use of underlying wPPRF via $i\mathcal{O}$.

4 Leakage-Resilient KEM

We begin this section by formally defining leakage-resilient PEPRFs. We then show that leakage-resilient PEPRFs naturally yield leakage-resilient KEM. Towards achieving leakage-resilience for PEPRFs, we first introduce a new notion called puncturable PEPRFs, and construct them from various puncturable primitives, which we believe is of independent interest. Finally, we show how to compile puncturable PEPRFs to leakage-resilient PEPRFs via $i\mathcal{O}$.

4.1 Leakage-Resilient PEPRFs

Chen and Zhang [CZ14] put forwarded the notion of PEPRFs, which is best viewed as a counterpart of weak PRFs in the public-key setting. In PEPRFs, each secret key is associated with a public key, and there is a collection of $\mathcal{NP}$ languages (indexed by public key) defined over domain. For any element in the language, in addition to evaluating its PRF value using secret key, one can also evaluate it publicly with public key and the associated witness.

PEPRFs neatly capture the essence of KEM, and they can be instantiated from either specific assumptions or more general assumptions such as (extractable) hash proof systems and trapdoor functions. In what follows, we recall the standard definition of PEPRFs from [CZ14] and proceed to introduce leakage resilience for them.

Definition 3 (PEPRFs). *Let $L = \{L_{pk}\}_{pk \in PK}$ be a collection of $\mathcal{NP}$ languages defined over X. A PEPRF $F : SK \times X \to Y \cup \perp$[17] for L consists of three polynomial time algorithms as below:*

- $\mathsf{Gen}(\lambda)$*: on input λ, output a public key pk and a secret key sk.*
- $\mathsf{PrivEval}(sk, x)$*: on input sk and $x \in X$, output $y \leftarrow F_{sk}(x) \in Y \cup \perp$.*
- $\mathsf{PubEval}(pk, x, w)$*: on input pk and $x \in L_{pk}$ together with a witness w, output $y \leftarrow F_{sk}(x) \in Y$.*

To be applicable, L is required to be efficiently samplable, i.e., for each $pk \in PK$, there exists an efficient sampling algorithm $\mathsf{SampRel}$ *that on input pk outputs a random element $x \in L_{pk}$ together with a witness w.*

Leakage-resilient weak pseudorandomness. Let $\mathcal{A}$ be an adversary against PEPRFs and define its advantage as below:

$$\mathsf{Adv}_{\mathcal{A}}(\lambda) = \Pr\left[\beta' = \beta : \begin{array}{l} (pk, sk) \leftarrow \mathsf{Gen}(\lambda); \\ state \leftarrow \mathcal{A}^{\mathcal{O}_{\mathsf{leak}}(\cdot)}(pk); \\ (x^*, w^*) \leftarrow \mathsf{SampRel}(pk); \\ y_0^* \leftarrow F_{sk}(x^*), y_1^* \xleftarrow{\mathrm{R}} Y; \\ \beta \xleftarrow{\mathrm{R}} \{0,1\}; \\ \beta' \leftarrow \mathcal{A}(pk, x^*, y_\beta^*); \end{array}\right] - \frac{1}{2}.$$

Here $\mathcal{O}_{\mathsf{leak}}(\cdot)$ is a leakage oracle that on input $f : SK \to \{0,1\}^*$ returns $f(sk)$, subjected to the restriction that the sum of its output lengths is at most ℓ. A PEPRF is ℓ-leakage-resilient weakly pseudorandom if no PPT adversary has non-negligible advantage in the above experiment. As pointed out in [CZ14], full pseudorandomness is impossible due to the publicly evaluable property.

Leakage-Resilient KEM. [CZ14] showed that weakly pseudorandom PEPRF naturally imply CPA-secure KEM. We observe that this implication applies in the leakage setting as well. We sketch the construction here for completeness. Assume $F : SK \times X \to Y$ is a leakage-resilient PEPRF for $L = \{L_{pk}\}_{pk \in PK}$, where the range Y is an additive group. The key pair is exactly the key pair of the underlying PEPRF. To encrypt a message $m \in Y$, one picks $x \xleftarrow{\mathrm{R}} L_{pk}$ with a witness w, computes $k \leftarrow \mathsf{PubEval}(pk, x, w)$ and outputs ciphertext $(x, k + m)$. The decryption process re-computes k via $\mathsf{PrivEval}(k, x)$. The LR CPA security of KEM readily follows from the LR weak pseudorandomness of the underlying PEPRF. The resulting LR CPA-secure KEM can be boosted to LR CPA-secure PKE by combining data encapsulation mechanism (DEM) with appropriate security properties [CS02].

[17] In a PEPRF, when the input x is not in L_{pk}, its PRF value $F_{sk}(x)$ may not be well defined and will be denoted by a distinguished symbol $\perp$.

4.2 Puncturable PEPRFs

To construct leakage-resilient PEPRFs, we first introduce the puncturable version of PEPRFs, called puncturable PEPRFs (PPEPRFs), which could also be viewed as an extension of PPRFs in the public-key setting. We formally define PPEPRFs as below and postpone their realizations to the full version [CWZ18].

Definition 4 (PPEPRFs). *Let $L = \{L_{pk}\}$ be a collection of $\mathcal{NP}$ languages defined over X. A PPEPRF $F : SK \times X \to Y \cup \perp$ for L consists of the following polynomial time algorithms:*

- $\mathsf{Gen}(\lambda)$*: on input λ, output a public key pk and a secret key sk.*
- $\mathsf{PrivEval}(sk, x)$*: on input sk and $x \in X$, output $y \leftarrow F_{sk}(x) \in Y \cup \perp$.*
- $\mathsf{Puncture}(sk, x^*)$*: on input sk and $x^* \in L_{pk}$, output a punctured key sk_{x^*}.*
- $\mathsf{PuncEval}(sk_{x^*}, x)$*: on input a punctured key sk_{x^*} and $x \neq x^*$, output $y \leftarrow F_{sk}(x) \in Y \cup \perp$.*
- $\mathsf{PubEval}(pk, x, w)$*: on input pk and $x \in L_{pk}$ together with a witness w, output $y \leftarrow F_{sk}(x) \in Y$.*

For security, we require that weak pseudorandomness remains even when the adversary is given a punctured secret key.

Weak pseudorandomness. Let $\mathcal{A}$ be an adversary against PPEPRFs and define its advantage as below:

$$\mathsf{Adv}_{\mathcal{A}}(\lambda) = \Pr\left[\beta' = \beta : \begin{array}{l} (pk, sk) \leftarrow \mathsf{Gen}(\lambda); \\ (x^*, w^*) \leftarrow \mathsf{SampRel}(pk); \\ sk_{x^*} \leftarrow \mathsf{Puncture}(sk, x^*); \\ y_0^* \leftarrow F_{sk}(x^*), y_1^* \xleftarrow{\mathrm{R}} Y; \\ \beta \xleftarrow{\mathrm{R}} \{0,1\}; \\ \beta' \leftarrow \mathcal{A}(pk, sk_{x^*}, x^*, y_\beta^*); \end{array}\right] - \frac{1}{2}.$$

A PPEPRF is weakly pseudorandom if for any PPT adversary $\mathcal{A}$ its advantage in the above experiment is negligible in λ.

4.3 Leakage-Resilient PEPRFs from PPEPRFs and $i\mathcal{O}$

Let $F : SK \times X \to Y \cup \perp$ be a PPEPRF for $L = \{L_{pk}\}_{pk \in PK}$, $i\mathcal{O}$ be an indistinguishability obfuscation, and $\mathsf{ext} : Y \times S \to Z$ be an average-case (n, ϵ)-strong extractor. Without loss of generality, we assume that $Y = \{0,1\}^\rho$. In what follows, we build a leakage-resilient PEPRF $\hat{F} : \hat{SK} \times \hat{X} \to Z \cup \perp$ for $\hat{L} = \{\hat{L}_{pk}\}_{pk \in PK}$, where $\hat{X} = X \times S$ and $\hat{L}_{pk} = \{\hat{x} = (x, s) : x \in L_{pk} \wedge s \in S\}$. According to the definition of $\hat{L}$, a witness w for $x \in L_{pk}$ is also a witness for $\hat{x} = (x, s) \in \hat{L}_{pk}$, where s could be any seed from S.

- $\mathsf{Gen}(\lambda)$: run $F.\mathsf{Gen}(\lambda)$ to obtain (pk, sk), create $\hat{sk} \leftarrow i\mathcal{O}(\mathrm{PrivEval})$, where the program PrivEval is defined in Figure 4; output $(pk, \hat{sk})$.

– $\mathsf{PrivEval}(\hat{sk}, \hat{x})$: on input $\hat{sk}$ and $\hat{x} = (x, s) \in \hat{X}$, output $\hat{y} \leftarrow \hat{sk}(\hat{x})$. This actually defines $\hat{F}_{\hat{sk}}(\hat{x}) := \mathsf{ext}(F_{sk}(x), s)$, where $\hat{x} = (x, s)$.
– $\mathsf{PubEval}(pk, \hat{x}, w)$: on input pk, $\hat{x} = (x, s) \in \hat{L}_{pk}$ and a witness w for $\hat{x}$, compute $y \leftarrow F_{sk}(x)$ via $F.\mathsf{PubEval}(pk, x, w)$, output $\hat{y} \leftarrow \mathsf{ext}(y, s)$.

PrivEval

Constants: PPEPRF secret key sk
Input: $\hat{x} = (x, s)$

1. Output $\mathsf{ext}(F_{sk}(x), s)$.

Fig. 4. Program PrivEval. The program is appropriately padded to the maximum of the size of itself and program PrivEval* described in Figure 5.

PrivEval*

Constants: PPEPRF punctured secret key sk_{x^*}, x^* and y^*
Input: $\hat{x} = (x, s)$

1. If $x = x^*$, output $\mathsf{ext}(y^*, s)$.
2. Else, output $\mathsf{ext}(F_{sk_{x^*}}(x), s)$.

Fig. 5. Program PrivEval*

Theorem 3. *If F is a secure PPEPRF, $i\mathcal{O}$ is indistinguishably secure, and* ext *is an average-case (n, ϵ)-strong extractor, the above PEPRF construction is ℓ-leakage-resilient weakly pseudorandom as long as $\ell \leq \rho - n$.*

The security proof is somewhat similar to that in Section 3. We postpone the details to the full version [CWZ18].

4.4 Construction with Improved Leakage Rate

The leakage rate of the above basic construction is low. Next, we show how to modify it to achieve optimal leakage rate. We need two extra primitives: (1) an IND-CPA secure SKE with message space $\{0,1\}^\rho$ and ciphertext space $\{0,1\}^v$; (2) a family of (v, τ)-lossy functions. The construction is as below.

– $\mathsf{Gen}(\lambda)$: run $(pk, sk) \leftarrow F.\mathsf{Gen}(\lambda)$, $h \leftarrow \mathrm{LF}.\mathsf{GenInj}(\lambda)$, $k_e \leftarrow \mathrm{SKE}.\mathsf{Gen}(\lambda)$, generate a dummy ciphertext $ct \leftarrow \mathrm{SKE}.\mathsf{Enc}(k_e, 0^\rho)$ as $\hat{sk}$, compute $\eta^* \leftarrow h(ct)$, create $C_{\mathrm{eval}} \leftarrow i\mathcal{O}(\mathrm{PrivEval})$ (here the program PrivEval is defined in Figure 6 and η^* acts as its trigger), set $\hat{pk} = (pk, C_{\mathrm{eval}})$, output $(\hat{pk}, \hat{sk})$.

– $\mathsf{PrivEval}(\hat{sk}, \hat{x})$: on input $\hat{sk}$ and $\hat{x} = (x, s) \in \hat{X}$, output $\hat{y} \leftarrow C_{\text{eval}}(\hat{sk}, \hat{x})$. This actually defines $\hat{F}_{\hat{sk}}(\hat{x}) := \mathsf{ext}(F_{sk}(x), s)$, where $\hat{x} = (x, s)$.
– $\mathsf{PubEval}(\hat{pk}, \hat{x}, w)$: on input $\hat{pk} = (pk, C_{\text{eval}}, t)$, $\hat{x} = (x, s) \in \hat{L}_{pk}$ and a witness w for $\hat{x}$, compute $y \leftarrow F_{sk}(x)$ via $F.\mathsf{PubEval}(pk, x, w)$, output $\hat{y} \leftarrow \mathsf{ext}(y, s)$.

PrivEval

Constants: PPEPRF secret key sk, η^*
Input: $\hat{sk}$, $\hat{x} = (x, s)$

1. If $h(\hat{sk}) \neq \eta^*$, output $\perp$.
2. Else, output $\mathsf{ext}(F_{sk}(x), s)$.

Fig. 6. Program PrivEval. This program is appropriately padded to the maximum of the size of itself and the program PrivEval* described in Figure 7.

PrivEval*

Constants: PPEPRF punctured secret key sk_{x^*}, k_e, x^* and η^*
Input: $\hat{sk}$, $\hat{x} = (x, s)$

1. If $h(\hat{sk}) \neq \eta^*$, output $\perp$.
2. If $x = x^*$, set $y^* \leftarrow \text{SKE}.\mathsf{Dec}(k_e, \hat{sk})$, output $\mathsf{ext}(y^*, s)$.
3. Else, output $\mathsf{ext}(F_{sk_{x^*}}(x), s)$.

Fig. 7. Program PuncEval

Theorem 4. *If F is a secure PPEPRF, $i\mathcal{O}$ is indistinguishably secure, SKE is an IND-CPA secure secret-key encryption, LF is a family of (v, τ)-lossy functions, ext is an average-case (n, ϵ)-strong extractor. the above PEPRF construction is ℓ-leakage-resilient weakly pseudorandom as long as $\ell \leq \rho - n - \tau$.*

Proof. By appropriate parameter choice (e.g. setting $v = \rho + o(\rho)$, $n = o(\rho)$, $\tau = o(v)$), we have $|\hat{sk}| = v = \rho + o(\rho)$ and $\ell = \rho - o(\rho)$ and thus the leakage rate is optimal.

We proceed via a sequence of games. Let S_i be the event that $\mathcal{A}$ succeeds in Game i.

Game 0. This is the standard leakage-resilient weak pseudorandomness game for PEPRFs. $\mathcal{CH}$ interacts with $\mathcal{A}$ as below.

1. Setup: $\mathcal{CH}$ runs $(pk, sk) \leftarrow F.\mathsf{Gen}(\lambda)$, $h \leftarrow \text{LF}.\mathsf{GenInj}(\lambda)$, samples $k_e \leftarrow \text{SKE}.\mathsf{Gen}(\lambda)$, generates a dummy ciphertext $ct \leftarrow \text{SKE}.\mathsf{Enc}(k_e, 0^\rho)$ as $\hat{sk}$,

computes $\eta^* \leftarrow h(ct)$, creates $C_{\text{eval}} \leftarrow i\mathcal{O}(\text{PrivEval})$. $\mathcal{CH}$ sets $\hat{pk} = (pk, C_{\text{eval}})$ and sends it to $\mathcal{A}$.

2. Leakage Query: Upon receiving leakage query $\langle f \rangle$, $\mathcal{CH}$ responds with $f(\hat{sk})$ as long as the total leakage is less than ℓ.
3. Challenge: $\mathcal{CH}$ samples $(x^*, w^*) \leftarrow \mathsf{SampRel}(pk)$, picks $s^* \xleftarrow{R} S$, computes $y^* \leftarrow F_{sk}(x^*)$ via $F.\mathsf{PubEval}(pk, x^*, w^*)$, $z_0^* \leftarrow \mathsf{ext}(y^*, s^*)$, samples $z_1^* \xleftarrow{R} Z$, $\beta \xleftarrow{R} \{0,1\}$. Finally, $\mathcal{CH}$ sends $\hat{x^*} = (x^*, s^*)$ and z_β^* to $\mathcal{A}$.
4. Guess: $\mathcal{A}$ outputs a guess β' for β and wins if $\beta' = \beta$.

According to the definition, we have:

$$\mathsf{Adv}_{\mathcal{A}}(\lambda) = |\Pr[S_0] - 1/2|$$

Game 1. Same as Game 0 except that $\mathcal{CH}$ samples x^*, w^* and computes $y^* \leftarrow F_{sk}(x^*)$ in the Setup stage. This change is purely conceptual and thus we have:

$$\Pr[S_1] = \Pr[S_0]$$

Game 2. Same as Game 1 except that $\mathcal{CH}$ computes $ct \leftarrow \mathrm{SKE}.\mathsf{Enc}(k_e, y^*)$ rather than $ct \leftarrow \mathrm{SKE}.\mathsf{Enc}(k_e, 0^\rho)$ in the Setup stage. By a direct reduction to the IND-CPA security of SKE, we have:

$$|\Pr[S_2] - \Pr[S_1]| \leq \mathsf{Adv}_{\mathcal{A}}^{\mathrm{SKE}}$$

Game 3. Same as Game 2 except that $\mathcal{CH}$ computes $sk_{x^*} \leftarrow F.\mathsf{Puncture}(sk, x^*)$ and creates $C_{\text{eval}} \leftarrow i\mathcal{O}(\text{PrivEval})$ in the Setup stage. Here, the program PrivEval* (defined in Figure 7) is built from constants (sk_{x^*}, x^*, y^*).

By the injectivity of h and the correctness of SKE and PPEPRF, the two programs PrivEval and PuncPriv agree on all inputs. By a direct reduction to the security of $i\mathcal{O}$, we conclude that:

$$|\Pr[S_3] - \Pr[S_2]| \leq \mathsf{Adv}_{\mathcal{A}}^{i\mathcal{O}}$$

Game 4. Same as Game 3 except that in the Setup stage $\mathcal{CH}$ picks $y^* \xleftarrow{R} Y$ rather than setting $y^* \leftarrow F_{sk}(x^*)$.

Assuming the weak pseudorandomness of the underlying PPEPRF, this modification is undetectable by all PPT adversaries. Thus, we have:

$$|\Pr[S_4] - \Pr[S_3]| \leq \mathsf{Adv}_{\mathcal{A}}^{\mathrm{PPEPRF}}$$

Game 5. Same as Game 4 except that $\mathcal{CH}$ samples a lossy function h via $\mathrm{LF}.\mathsf{GenLossy}(\lambda)$ rather than sampling an injective function in the Setup stage. By a direct reduction to the security of lossy functions, we conclude that:

$$|\Pr[S_5] - \Pr[S_4]| \leq \mathsf{Adv}_{\mathcal{A}}^{\mathrm{LF}}$$

Game 6. Same as Game 5 except that $\mathcal{CH}$ picks $z_0^* \xleftarrow{R} Z$ rather than setting $z_0^* \leftarrow \mathsf{ext}(y^*, s^*)$ in the Challenge stage.

We denote by V the set of public key $\hat{pk} = (pk, C_{\text{eval}})$, x^* and s^*. In both Game 5 and Game 6, y^* is uniformly chosen from Y (independent of sk_{x^*}, x^* and s^*) and but is correlated to η^* which has at most 2^τ values, we have $\mathsf{H}_\infty(y^*|V) \geq \rho - \tau$ by the chain rule. Observe that $\mathcal{A}$ also obtains at most ℓ bits leakage on $\hat{sk}$ (denote by $leak$) which is correlated to y^*, it follows by the chain rule that $\tilde{\mathsf{H}}_\infty(y^*|(V, leak)) \geq \mathsf{H}_\infty(y^*|V) - \ell = \rho - \tau - \ell$, which is greater than n by the parameter choice. Since ext is an average-case (n, ϵ)-strong extractor, we conclude that $\mathsf{ext}(y^*, s^*)$ is ϵ-close to a uniformly random $z_0^* \in Z$, even given V and leakage. Observe that $\mathcal{A}$'s view in Game 5 and Game 6 are fully determined by z_0^*, z_1^*, β^*, V and $leak$, while z_1^*, β^*, V and $leak$ are distributed identically in Game 5 and Game 6. Thereby, $\mathcal{A}$'s view in Game 5 and Game 6 are $\epsilon/2$-close. Thus, we have:

$$|\Pr[S_6] - \Pr[S_5]| \leq \epsilon/2 \leq \mathsf{negl}(\lambda)$$

In Game 6, both z_0^* and z_1^* are randomly chosen from Z. Therefore, we have:

$$\Pr[S_6] = 1/2$$

Putting all the above together, the theorem immediately follows. □

5 Leakage-Resilient Signature

To best illustrate our idea, in the section we only present the construction with selective security. We postpone the constructions with adaptive security and optimal leakage rate to the full version [CWZ18].

5.1 Selective Construction from sPPRFs, Leakage-Resilient OWFs and $i\mathcal{O}$

Let $F : K \times M \to \{0,1\}^n$ be a sPPRF, $i\mathcal{O}$ be an indistinguishability obfuscator, $g : \{0,1\}^n \to \{0,1\}^\mu$ be a leakage-resilient OWF. We build a leakage-resilient signature as below.

- $\mathsf{Gen}(\lambda)$: run $(pp, k) \leftarrow F.\mathsf{Gen}(\lambda)$, create $sk \leftarrow i\mathcal{O}(\text{Sign})$ and $vk \leftarrow i\mathcal{O}(\text{Verify})$. The programs Sign and Verify are defined in Figure 8 and Figure 10 respectively.
- $\mathsf{Sign}(sk, m)$: output $\sigma \leftarrow sk(m)$.
- $\mathsf{Verify}(vk, m, \sigma)$: output $vk(m, \sigma)$.

Theorem 5. *If F is a secure sPPRF, $i\mathcal{O}$ is indistinguishably secure, g is ℓ-leakage-resilient one-way, the above construction is ℓ-leakage-resilient EUF-CMA in the selective sense.*

Proof. We proceed via a sequence of games. Let S_i be the probability that $\mathcal{A}$ wins in Game i.

Sign

Constants: sPPRF key k
Input: message m

1. Compute $\sigma \leftarrow F(k, m)$.

Fig. 8. Program Sign. This program is appropriately padded to the maximum of the size of itself and program Sign* defined in Figure 9.

Sign*

Constants: sPPRF punctured key k_{m^*}, m^*, σ^*
Input: message m

1. If $m = m^*$, output σ^*.
2. Else, output $\sigma \leftarrow F(k_{m^*}, m)$.

Fig. 9. Program Sign*

Verify

Constants: sPPRF key k
Input: message m and signature σ

1. Test if $g(\sigma) = g(F(k, m))$, output 1 if true and 0 if false.

Fig. 10. Program Verify. This program is appropriately padded to the maximum of the size of itself and the program Verify* defined in Figure 11.

Verify*

Constants: sPPRF punctured key k_{m^*}, m^* and y^*
Input: message m and signature σ

1. If $m = m^*$, test whether $g(\sigma) = y^*$. Output 1 if true and 0 if false.
2. Else, test if $g(\sigma) = g(F(k_{m^*}, m))$. Output 1 if true and 0 if false.

Fig. 11. Program Verify*

Game 0. This is the standard leakage-resilient selective EUF-CMA game for signature. $\mathcal{CH}$ interacts with $\mathcal{A}$ as follows:

1. Commit: $\mathcal{A}$ submits the target message m^* to $\mathcal{CH}$.
2. Setup: $\mathcal{CH}$ runs $(pp, k) \leftarrow F.\mathsf{Gen}(\lambda)$, creates $sk \leftarrow i\mathcal{O}(\text{Sign})$, $vk \leftarrow i\mathcal{O}(\text{Verify})$. $\mathcal{CH}$ sends vk to $\mathcal{A}$.
3. Signing Query: Upon receiving signing query $\langle m \rangle \neq \langle m^* \rangle$, $\mathcal{CH}$ responds with $\sigma \leftarrow sk(m)$.
4. Leakage Query: Upon receiving leakage query $\langle f \rangle$, $\mathcal{CH}$ responds with $f(sk)$.
5. Forge: $\mathcal{A}$ outputs a forgery σ' and wins if $\mathsf{Verify}(vk, m^*, \sigma') = 1$.

According to the definition of $\mathcal{A}$, we have:

$$\mathsf{Adv}_{\mathcal{A}}(\lambda) = \Pr[S_0]$$

Game 1. Same as Game 0 except that in the Setup stage $\mathcal{CH}$ computes $\sigma^* \leftarrow F(k, m^*)$, $y^* \leftarrow g(\sigma^*)$, and $k_{m^*} \leftarrow F.\mathsf{Puncture}(k, m^*)$, creates $vk \leftarrow i\mathcal{O}(\text{Verify}^*)$, where the program Verify* is defined in Figure 11.

It is easy to check that the programs Verify and Verify* agree on all inputs. By the security of $i\mathcal{O}$, we have:

$$|\Pr[S_1] - \Pr[S_0]| \leq \mathsf{Adv}^{i\mathcal{O}}_{\mathcal{A}}$$

Game 2. Same as Game 1 except that $\mathcal{CH}$ uses k_{m^*} to handle signing queries, i.e., returning $\sigma \leftarrow F(k_{m^*}, m)$ for $m \neq m^*$.

By the correctness of sPPRF, Game 1 and Game 2 are identical in $\mathcal{A}$'s view. Thus, we have:

$$\Pr[S_2] = \Pr[S_1]$$

Game 3. Same as Game 2 except that $\mathcal{CH}$ creates $sk \leftarrow i\mathcal{O}(\text{Sign}^*)$ in the Setup stage. Here the program Sign* (defined in Figure 9) is built from constants k_{m^*}, m^* and σ^*.

It is easy to check that the two programs Sign and Sign* agree on all inputs. By the security of $i\mathcal{O}$, we have:

$$|\Pr[S_3] - \Pr[S_2]| \leq \mathsf{Adv}^{i\mathcal{O}}_{\mathcal{A}}$$

Game 4. Same as Game 3 except that in Setup stage $\mathcal{CH}$ picks $\sigma^* \xleftarrow{R} \{0,1\}^n$ rather than setting $\sigma^* \leftarrow F(k, m^*)$.

By the selective pseudorandomness of sPPRF, we have:

$$|\Pr[S_4] - \Pr[S_3]| \leq \mathsf{Adv}^{\text{sPPRF}}_{\mathcal{A}} \tag{1}$$

It remains to analyze $\Pr[S_4]$. We have the following claim.

Claim. If g is an ℓ-leakage-resilient OWF, then the advantage of any PPT adversary in Game 4 is negligible in λ.

Proof. Let $\mathcal{A}$ be a PPT adversary wins Game 4 with advantage $\mathsf{Adv}_{\mathcal{A}}(\lambda)$. We construct an adversary $\mathcal{B}$ that breaks the assumed leakage-resilient one-wayness of g with the same advantage, implying that $\Pr[S_4]$ must be negligible.

Given (g, y^*) where $y^* \leftarrow g(\sigma^*)$ for some $\sigma^* \xleftarrow{R} \{0,1\}^n$, $\mathcal{B}$ interacts with $\mathcal{A}$ in Game 4 with the aim to output σ' such that $g(\sigma') = y^*$.

1. Commit: $\mathcal{A}$ submits the target message m^* to $\mathcal{CH}$.
2. Setup: $\mathcal{B}$ runs $(pp, k) \leftarrow F.\mathsf{Gen}(\lambda)$, computes $k_{m^*} \leftarrow F.\mathsf{Puncture}(k, m^*)$, creates $vk \leftarrow i\mathcal{O}(\text{Verify}^*)$, and sends vk to $\mathcal{A}$. $\mathcal{B}$ also picks random coins r used for obfuscating the program Sign* (with constants k_{m^*}, m^*, σ^* hardwired) for later simulation. Note that the constant σ^* is unknown to $\mathcal{B}$.

3. Signing Query: Upon receiving signing query $\langle m \rangle \neq \langle m^* \rangle$, $\mathcal{B}$ responds with $\sigma \leftarrow F(k_{m^*}, m)$ using k_{m^*}.
4. Leakage Query: Note that the signing key $sk \leftarrow i\mathcal{O}(\mathsf{Sign}^*_{k_{m^*},m^*,\sigma^*}; r)$ could be viewed as the value of some function $\psi(\cdot)$ at point σ^*, where $\psi(\cdot)$ on input σ outputs $i\mathcal{O}(\mathsf{Sign}^*_{k_{m^*},m^*,\sigma}; r)$. Since $i\mathcal{O}$ is efficiently computable, so is $\psi(\cdot)$. Based on this observation, $\mathcal{B}$ can transform any leakage queries on sk to leakage queries on σ^*. Upon receiving leakage query $\langle f \rangle$, $\mathcal{B}$ makes leakage query $\langle f \circ \psi \rangle$ to its own challenger and forwards the reply to $\mathcal{A}$.
5. Forge: $\mathcal{A}$ outputs a forgery σ' and wins if $\mathsf{Verify}(vk, m^*, \sigma') = 1$.

Finally, $\mathcal{B}$ forwards σ' to its challenger. It is straightforward to verify that $\mathcal{B}$'s simulation for Game 4 is perfect. If $\mathcal{A}$ succeeds, according to the definition of algorithm Verify in Game 4, σ' is indeed a preimage of y^* under g, thus $\mathcal{B}$ also succeeds. This proves the claim.

Putting all the above together, the theorem immediately follows. □

Acknowledgments. We thank the anonymous reviewers of Asiacrypt 2018 for their helpful comments. The first author is supported by National Natural Science Foundation of China (Grant No. 61772522), Youth Innovation Promotion Association CAS, Key Research Program of Frontier Sciences, CAS (Grant No. QYZDB-SSW-SYS035). The second author is partially supported by Nomura Research Institute, JST CREST JPMJCR14D6, JST OPERA. The third author is partially supported by NSF grant 1801470.

References

[ADN+10] Joël Alwen, Yevgeniy Dodis, Moni Naor, Gil Segev, Shabsi Walfish, and Daniel Wichs. Public-key encryption in the bounded-retrieval model. In *EUROCRYPT*, pages 113–134, 2010.

[ADW09] Joël Alwen, Yevgeniy Dodis, and Daniel Wichs. Leakage-resilient public-key cryptography in the bounded-retrieval model. In *CRYPTO*, pages 36–54, 2009.

[AGV09] Adi Akavia, Shafi Goldwasser, and Vinod Vaikuntanathan. Simultaneous hardcore bits and cryptography against memory attacks. In *TCC*, pages 474–495, 2009.

[BCH12] Nir Bitansky, Ran Canetti, and Shai Halevi. Leakage-tolerant interactive protocols. In *TCC*, pages 266–284, 2012.

[BDL97] Dan Boneh, Richard A. DeMillo, and Richard J. Lipton. On the importance of checking cryptographic protocols for faults. In *EUROCRYPT*, pages 37–51, 1997.

[BG10] Zvika Brakerski and Shafi Goldwasser. Circular and leakage resilient public-key encryption under subgroup indistinguishability - (or: Quadratic residuosity strikes back). In *CRYPTO*, pages 1–20, 2010.

[BGI+12] Boaz Barak, Oded Goldreich, Russell Impagliazzo, Steven Rudich, Amit Sahai, Salil P. Vadhan, and Ke Yang. On the (im)possibility of obfuscating programs. *J. ACM*, 59(2):6, 2012.

[BGI14] Elette Boyle, Shafi Goldwasser, and Ioana Ivan. Functional signatures and pseudorandom functions. In *PKC*, pages 501–519, 2014.

[BK12] Zvika Brakerski and Yael Tauman Kalai. A parallel repetition theorem for leakage resilience. In *TCC*, pages 248–265, 2012.

[BKKV10] Zvika Brakerski, Yael Tauman Kalai, Jonathan Katz, and Vinod Vaikuntanathan. Overcoming the hole in the bucket: Public-key cryptography resilient to continual memory leakage. In *FOCS*, pages 501–510, 2010.

[BS97] Eli Biham and Adi Shamir. Differential fault analysis of secret key cryptosystems. In *CRYPTO*, pages 513–525, 1997.

[BSW11] Elette Boyle, Gil Segev, and Daniel Wichs. Fully leakage-resilient signatures. In *EUROCRYPT*, pages 89–108, 2011.

[BSW16] Mihir Bellare, Igors Stepanovs, and Brent Waters. New negative results on differing-inputs obfuscation. In *EUROCRYPT*, pages 792–821, 2016.

[BW13] Dan Boneh and Brent Waters. Constrained pseudorandom functions and their applications. In *ASIACRYPT*, pages 280–300, 2013.

[CDRW10] Sherman S. M. Chow, Yevgeniy Dodis, Yannis Rouselakis, and Brent Waters. Practical leakage-resilient identity-based encryption from simple assumptions. In *ACM CCS*, pages 152–161, 2010.

[CQX18] Yu Chen, Baodong Qin, and Haiyang Xue. Regularly lossy functions and their applications. In *CT-RSA*, 2018.

[CS02] Ronald Cramer and Victor Shoup. Universal hash proofs and a paradigm for adaptive chosen ciphertext secure public-key encryption. In *EUROCRYPT*, pages 45–64, 2002.

[CWZ18] Yu Chen, Yuyu Wang, and Hong-Sheng Zhou. Leakage-resilient cryptography from puncturable primitives and obfuscation, 2018. http://eprint.iacr.org/2018/781.

[CZ14] Yu Chen and Zongyang Zhang. Publicly evaluable pseudorandom functions and their applications. In *SCN*, pages 115–134, 2014.

[DGK+10] Yevgeniy Dodis, Shafi Goldwasser, Yael Tauman Kalai, Chris Peikert, and Vinod Vaikuntanathan. Public-key encryption schemes with auxiliary inputs. In *TCC*, pages 361–381, 2010.

[DGL+16] Dana Dachman-Soled, S. Dov Gordon, Feng-Hao Liu, Adam O'Neill, and Hong-Sheng Zhou. Leakage-resilient public-key encryption from obfuscation. In *PKC*, pages 101–128, 2016.

[DHLAW10] Yevgeniy Dodis, Kristiyan Haralambiev, Adriana López-Alt, and Daniel Wichs. Cryptography against continuous memory attacks. In *FOCS*, pages 511–520, 2010.

[DHLW10] Yevgeniy Dodis, Kristiyan Haralambiev, Adriana López-Alt, and Daniel Wichs. Efficient public-key cryptography in the presence of key leakage. In *ASIACRYPT*, pages 613–631, 2010.

[DKL09] Yevgeniy Dodis, Yael Tauman Kalai, and Shachar Lovett. On cryptography with auxiliary input. In *STOC*, pages 621–630, 2009.

[DY13] Yevgeniy Dodis and Yu Yu. Overcoming weak expectations. In *TCC*, pages 1–22, 2013.

[GGH+13] Sanjam Garg, Craig Gentry, Shai Halevi, Mariana Raykova, Amit Sahai, and Brent Waters. Candidate indistinguishability obfuscation and

functional encryption for all circuits. In *FOCS*, pages 40–49, 2013.

[GGHW14] Sanjam Garg, Craig Gentry, Shai Halevi, and Daniel Wichs. On the implausibility of differing-inputs obfuscation and extractable witness encryption with auxiliary input. In *CRYPTO*, pages 518–535, 2014.

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

[GJS11] Sanjam Garg, Abhishek Jain, and Amit Sahai. Leakage-resilient zero knowledge. In *CRYPTO*, pages 297–315, 2011.

[GKPV10] Shafi Goldwasser, Yael Tauman Kalai, Chris Peikert, and Vinod Vaikuntanathan. Robustness of the learning with errors assumption. In *ICS*, pages 230–240, 2010.

[GM84] Shafi Goldwasser and Silvio Micali. Probabilistic encryption. *J. Comput. Syst. Sci.*, 28(2):270–299, 1984.

[HL11] Shai Halevi and Huijia Lin. After-the-fact leakage in public-key encryption. In *TCC*, pages 107–124, 2011.

[HLWW13] Carmit Hazay, Adriana López-Alt, Hoeteck Wee, and Daniel Wichs. Leakage-resilient cryptography from minimal assumptions. In *EUROCRYPT*, pages 160–176, 2013.

[HSH+08] J. Alex Halderman, Seth D. Schoen, Nadia Heninger, William Clarkson, William Paul, Joseph A. Calandrino, Ariel J. Feldman, Jacob Appelbaum, and Edward W. Felten. Lest we remember: Cold boot attacks on encryption keys. In *USENIX Security Symposium*, pages 45–60, 2008.

[HW09] Susan Hohenberger and Brent Waters. Short and stateless signatures from the RSA assumption. In *CRYPTO*, pages 654–670, 2009.

[IPS15] Yuval Ishai, Omkant Pandey, and Amit Sahai. Public-coin differing-inputs obfuscation and its applications. In *TCC*, pages 668–697, 2015.

[KJJ99] Paul C. Kocher, Joshua Jaffe, and Benjamin Jun. Differential power analysis. In *CRYPTO*, pages 388–397, 1999.

[KMO10] Eike Kiltz, Payman Mohassel, and Adam O'Neill. Adaptive trapdoor functions and chosen-ciphertext security. In *EUROCRYPT*, pages 673–692, 2010.

[Koc96] Paul C. Kocher. Timing attacks on implementations of Diffie-Hellman, RSA, DSS, and other systems. In *CRYPTO*, pages 104–113, 1996.

[Kom16] Ilan Komargodski. Leakage resilient one-way functions: The auxiliary-input setting. In *TCC*, pages 139–158, 2016.

[KPTZ13] Aggelos Kiayias, Stavros Papadopoulos, Nikos Triandopoulos, and Thomas Zacharias. Delegatable pseudorandom functions and applications. In *ACM CCS*, pages 669–684, 2013.

[KV09] Jonathan Katz and Vinod Vaikuntanathan. Signature schemes with bounded leakage resilience. In *ASIACRYPT*, pages 703–720, 2009.

[LLW11] Allison B. Lewko, Mark Lewko, and Brent Waters. How to leak on key updates. In *STOC*, pages 725–734, 2011.

[LRW11] Allison B. Lewko, Yannis Rouselakis, and Brent Waters. Achieving leakage resilience through dual system encryption. In *TCC*, pages 70–88, 2011.

[LWZ13] Shengli Liu, Jian Weng, and Yunlei Zhao. Efficient public key cryptosystem resilient to key leakage chosen ciphertext attacks. In *CT-RSA*, pages 84–100, 2013.

[MH15] Takahiro Matsuda and Goichiro Hanaoka. Constructing and understanding chosen ciphertext security via puncturable key encapsulation mechanisms. In *TCC*, pages 561–590, 2015.

[MR04] Silvio Micali and Leonid Reyzin. Physically observable cryptography (extended abstract). In *TCC*, pages 278–296, 2004.

[MTVY11] Tal Malkin, Isamu Teranishi, Yevgeniy Vahlis, and Moti Yung. Signatures resilient to continual leakage on memory and computation. In *TCC*, pages 89–106, 2011.

[NS09] Moni Naor and Gil Segev. Public-key cryptosystems resilient to key leakage. In *CRYPTO*, pages 18–35, 2009.

[Pie09] Krzysztof Pietrzak. A leakage-resilient mode of operation. In *Advances in Cryptology - EUROCRYPT 2009*, pages 462–482, 2009.

[PW08] Chris Peikert and Brent Waters. Lossy trapdoor functions and their applications. In *STOC*, pages 187–196, 2008.

[QL13] Baodong Qin and Shengli Liu. Leakage-resilient chosen-ciphertext secure public-key encryption from hash proof system and one-time lossy filter. In *ASIACRYPT*, pages 381–400, 2013.

[QL14] Baodong Qin and Shengli Liu. Leakage-flexible cca-secure public-key encryption: Simple construction and free of pairing. In *PKC*, pages 19–36, 2014.

[Reg05] Oded Regev. On lattices, learning with errors, random linear codes, and cryptography. In *STOC*, pages 84–93, 2005.

[RS09] Alon Rosen and Gil Segev. Chosen-ciphertext security via correlated products. In *TCC*, pages 419–436, 2009.

[RW14] Kim Ramchen and Brent Waters. Fully secure and fast signing from obfuscation. In *ACM CCS*, pages 659–673, 2014.

[SW14] Amit Sahai and Brent Waters. How to use indistinguishability obfuscation: deniable encryption, and more. In *STOC*, pages 475–484, 2014.

[Wee10] Hoeteck Wee. Efficient chosen-ciphertext security via extractable hash proofs. In *CRYPTO*, pages 314–332, 2010.

[Wic13] Daniel Wichs. Barriers in cryptography with weak, correlated and leaky sources. In *Innovations in Theoretical Computer Science, ITCS*, pages 111–126, 2013.

[WMHT16] Yuyu Wang, Takahiro Matsuda, Goichiro Hanaoka, and Keisuke Tanaka. Signatures resilient to uninvertible leakage. In *SCN*, pages 372–390, 2016.

[YCZY12] Tsz Hon Yuen, Sherman S. M. Chow, Ye Zhang, and Siu-Ming Yiu. Identity-based encryption resilient to continual auxiliary leakage. In *EUROCRYPT*, pages 117–134, 2012.

[YXZ+15] Rupeng Yang, Qiuliang Xu, Yongbin Zhou, Rui Zhang, Chengyu Hu, and Zuoxia Yu. Updatable hash proof system and its applications. In *ESORICS*, pages 266–285, 2015.

[Zha16] Mark Zhandry. The Magic of ELFs. In *CRYPTO*, pages 479–508, 2016.