A Parallel Repetition Theorem for Leakage Resilience

Zvika Brakerski[1] and Yael Tauman Kalai[2]

[1] Stanford University, zvika@stanford.edu.
[2] Microsoft Research, yael@microsoft.com.

Abstract. A leakage resilient encryption scheme is one which stays secure even against an attacker that obtains a bounded amount of side information on the secret key (say λ bits of "leakage"). A fundamental question is whether parallel repetition amplifies leakage resilience. Namely, if we secret share our message, and encrypt the shares under two independent keys, will the resulting scheme be resilient to 2λ bits of leakage?

Surprisingly, Lewko and Waters (FOCS 2010) showed that this is false. They gave an example of a public-key encryption scheme that is (CPA) resilient to λ bits of leakage, and yet its 2-repetition is not resilient to even $(1+\epsilon)\lambda$ bits of leakage. In their counter-example, the repeated schemes share secretly generated public parameters.

In this work, we show that under a reasonable strengthening of the definition of leakage resilience (one that captures known proof techniques for achieving non-trivial leakage resilience), parallel repetition *does* in fact amplify leakage (for CPA security). In particular, if fresh public parameters are used for each copy of the Lewko-Waters scheme, then their negative result does not hold, and leakage is amplified by parallel repetition.

More generally, given t schemes that are resilient to $\lambda_1, \ldots, \lambda_t$ bits of leakage, respectfully, we show that their direct product is resilient to $\sum(\lambda_i - 1)$ bits. We present our amplification theorem in a general framework that applies other cryptographic primitives as well.

1 Introduction

In recent years, motivated by a large variety of real-world physical attacks, there has been a major effort by the cryptographic community to construct schemes that are resilient to leakage from the secret keys. This successful line of work gave rise to many constructions of leakage-resilient cryptographic primitives, including stream ciphers [11, 19], signature schemes [15, 12], symmetric and public-key encryption schemes [1, 18, 10, 9], as well as more complicated primitives.

A natural question to ask is: Does parallel repetition amplify leakage? More concretely, suppose we are given a public-key encryption scheme $\mathcal{E}$ that remains secure even if λ bits about the secret key are leaked. Is it possible to amplify the leakage-resilience to $t\lambda$ by taking t copies of $\mathcal{E}$, and encrypting a message m by secret sharing it, and encrypting the i^{th} share using $\mathcal{E}_i$ (we denote the resulting

scheme by $\mathcal{E}^t$)? Using an appropriate definition of parallel repetition, a similar question can be asked for signatures.

Alwen, Dodis, and Wichs [3] and Alwen, Dodis, Naor, Segev, Walfish and Wichs [2] were able to amplify leakage resilience for particular schemes, using the specific properties of these schemes. They raised the fundamental question of whether leakage resilience can *always* be amplified by parallel repetition. They predicted that such a result will be hard or even impossible to prove under the known definitions.

Recently, Lewko and Waters [16] gave a striking negative result, giving an example of a public-key encryption scheme that is resilient to λ bits of leakage but whose 2 repetition is not resilient to even $(1+\epsilon)\lambda$ bits. This was followed by a work of Jain and Pietrzak [14] who presented a signature scheme where increasing the number of repetitions does not improve the leakage resilience at all. We elaborate on these negative results (and on how they go hand-in-hand with our positive results) in Section 1.2.

1.1 Our Results

We give positive results, by proving direct product theorems for leakage resilience. In particular, we show that parallel repetition does amplify the leakage resilience (almost) as expected.

The leakage model we consider is based on the "*noisy* leakage" model of Naor and Segev [18].[3] In this model, "legal" leakage functions are poly-size circuits that reduce the min-entropy of the secret key by at most λ. A scheme is said to be *λ-leakage resilient* if every PPT adversary, that asks for a "legal" leakage function, breaks the scheme with only negligible probability.

In this work, we consider a slightly relaxed leakage model. Instead of requiring the leakage function to *always* reduce the min-entropy of sk by at most λ, we require that it should be hard to break the scheme on those leakage values that do reduce the min-entropy by at most λ. In other words, we consider a *point-wise* definition: We say that a scheme is *point-wise λ-leakage resilient* if for any PPT adversary, that asks for a poly-size leakage function L, the probability that both the leakage value $y \leftarrow L(pk, sk)$ reduces the min-entropy of sk by at most λ, and that $\mathcal{A}(pk, y)$ breaks the scheme, is negligible.

We believe that this leakage model is of independent interest, as it captures our "intent" better: As long as the secret key is left with enough min-entropy, the scheme is secure. Moreover, we note that all known constructions that are λ-leakage resilient are also point-wise λ-leakage resilient (including [18, 15, 9, 5]). We elaborate on this in Section 4.

At first it may seem that point-wise leakage is equivalent to noisy leakage. However, the difficulty is that it may be hard to determine whether a leakage value $y \leftarrow L(pk, sk)$ indeed reduces the min-entropy of sk by at most λ. If this

[3] While "entropic leakage" may be a more suitable name for this model, we stick with the terminology of [18] for historic reasons.

was efficiently determined, then indeed we would have a reduction between the two models.

For technical reasons (see Section 1.3), we need to further relax our leakage model for our results to go through. We consider two (incomparable) relaxations.

First Relaxation: Almost λ-Leakage. In the first relaxation, instead of requiring that sk has high min-entropy (given pk, y), we require that it is statistically close to a random variable with high min-entropy. A scheme that is secure in this model is said to be *point-wise almost λ-leakage resilient*. We can prove a direct product theorem of any *constant* number of repetitions under this definition.

Theorem 1. *Let $c \in \mathbb{N}$ be a constant, and for every $i \in [c]$, let $\mathcal{E}_i$ be a point-wise almost λ_i-leakage-resilient public-key encryption scheme. Then, $\mathcal{E}_1 \times \ldots \times \mathcal{E}_c$ is point-wise almost λ-leakage-resilient, where $\lambda = \sum_{i=1}^{c}(\lambda_i - 1)$.*

We refer the reader to Section 1.3 and Section 5 for more details.

Second Relaxation: Leakage with Small Advice. In the second relaxation, we give the adversary an additional logarithmic (in the security parameter) number of bits of (possibly hard to compute) advice (quite surprisingly, we were unable to reduce this model to the point-wise λ-leakage model). A scheme that is secure in this model is said to be *point-wise λ-leakage resilient with logarithmic advice*. We can prove a direct product theorem of any *polynomial* number of repetitions under this definition.

We note that it is not clear what it means to have t different leakage resilient schemes when t is super constant, since there is a different number of schemes for each value of the security parameter. While one can come up with a proper definition (involving a generation algorithm that, for every value of the security parameter, gets i and implements $\mathcal{E}_i$), for the sake of clarity, we choose to state the theorem below only for parallel repetition of the same scheme.

Theorem 2. *Let $t = t(k)$ be a polynomial in the security parameter. Let $\mathcal{E}$ be a public-key encryption scheme that is point-wise λ-leakage resilient with logarithmic advice. Then $\mathcal{E}^t$ is point-wise $t(\lambda - 1)$-leakage resilient with logarithmic advice.*

We refer the reader to Section 1.3 for an overview of the proof, and to Section 6 for more details.

The Relation Between our Models. Interestingly, we are not able to show that our relaxations are equivalent to one another, nor to show that they are implied by (plain) point-wise leakage resilience. This is surprising since in the bounded leakage model,[4] a negligible change in the secret-key distribution, or adding a logarithmic number of hard to compute bits, does not change the model. In

[4] Where the leakage function's output is required to be bounded by λ bits, as opposed to our requirement that the secret key has high residual entropy.

a nutshell, the reason that this does not carry to our models, is that having high min-entropy is not an efficiently verifiable condition, and that statistical indistinguishability does not preserve min-entropy.

We are able to show, however, that point-wise λ-leakage resilience implies λ-bounded leakage resilience (for the same value of λ), and thus in particular, our relaxed models also imply bounded leakage resilience. We note that proving the above is somewhat nontrivial since we do not want to suffer a degradation in λ. We refer the reader to Section 3.3 for a formal presentation.

Our Models and Current Proof Techniques. We show that for essentially all known schemes that are resilient to non-trivial leakage (i.e. super-logarithmic in the hardness of the underlying problem), amplification of leakage resilience via parallel repetition works. Specifically, this includes the Lewko-Waters counterexample, if the public parameters are chosen independently for each copy of the scheme. In order to do this, we identify a proof template that is used in all leakage resilience proofs, and show that this template is strong enough to prove point-wise leakage resilience, as well as our relaxed notions. See Section 4 for the full details.

The Lewko-Waters counterexample uses its public parameters in a very particular way that makes the argument not go through (see below).

1.2 Prior Work

As we claimed above, all known leakage resilient schemes are proved using the same proof template, and remain secure under our leakage models. This implies that parallel repetition should amplify security for all known schemes, which does not seem to coincide with the negative results of [16, 14]. We explain this alleged discrepancy below.

The Lewko-Waters Counterexample. Lewko and Waters [16] construct a public key encryption scheme that is resilient to non-trivial length-bounded leakage, and prove that parallel repetition does not amplify its leakage resilience. However, the copies of their encryption scheme share public parameters: They are all using the same bilinear group. Their scheme, like all other schemes we are aware of, is (computationally indistinguishable from) point-wise leakage resilient and our theorems imply that parallel repetition does amplify its resilience to leakage. This is true so long as the public parameters are generated anew for each copy of the scheme: In our proof, we need to be able to sample key pairs for the scheme in question. Lewko and Waters use the public parameters in an extremely pathological (and clever!) way: The public parameters enable to generate keys for their actual scheme, but not for the computationally indistinguishable scheme where leakage resilience is actually proven. However, if we consider the generation of public parameters as a part of the key generation process, then new key pairs can always be generated, and parallel repetition works.

The Jain-Pietrzak Counterexample. Jain and Pietrzak [14] give a negative result for signature schemes. They take any secure signature scheme and change it so that if the message to be signed belongs to a set H, then the signature algorithm simply outputs the entire secret key. The set H is computationally hard to hit (given only the public key), and thus the scheme remains secure. It follows that the scheme remains secure also given leakage of length $O(\log k)$, where k is the security parameter (more generally, if the underlying problem is 2^λ hard, then the scheme is resilient to $\sim \lambda$ bits of leakage).

They prove that parallel repetition fails, by proving that if the scheme is repeated t times, for some large enough t, then the leakage can in fact give enough information to find a message m that belongs to all the sets H_i, and thus break security completely. They start with a result that relies on common public parameters: a common (seeded) hash function. Then, they suggest to remove this public parameter by replacing the seeded hash function with an explicit hash function, such as SHA256. However, this explicit hash function is also, in some sense, a joint *non-uniform* public parameter.

This counterexample heavily relies on the "help" of the signing oracle when breaking the repeated scheme. The paper also presents a construction of a CCA encryption scheme, where they use the decryption oracle to break the parallel repetition system.

In general, signature schemes are not covered by our amplification theorems. Our theorems (and proofs) only cover public key primitives where the challenger in the security game does not need to know the secret key (beyond providing the adversary with the leakage value). Our results do extend to schemes such as signature schemes or CCA encryption schemes, if they have the property that the challenger (i.e., the signing oracle or the decryption oracle) can be efficiently simulated given only the public key (or given very little information about the secret key), in a way that is computationally indistinguishable even given the leakage. For example, the signature scheme of Katz and Vaikuntanathan [15] has this property, and thus its leakage resilience is amplified by parallel repetition. Whether our techniques can be applied to other leakage resilient signature schemes (e.g. [4, 17, 13]) is an interesting question that we leave for further research.

1.3 Overview of Our Techniques

In what follows we give a high-level overview of our proofs. For the sake of simplicity, we focus on the case of two-fold parallel repetition. Let $\mathcal{E}$ be any λ-leakage resilient encryption scheme. Our goal is to prove that the scheme $\mathcal{E}^2$ is 2λ-leakage resilient. For technical reasons, in our actual proof, we manage to show that $\mathcal{E}^2$ is $(2\lambda - 1)$-leakage resilient (in both our leakage models).

Our proof is by reduction: Suppose there exists an adversary $\mathcal{B}$ for the parallel repetition scheme $\mathcal{E}^2$ that leaks $L(pk_1, pk_2, sk_1, sk_2)$, where L reduces the min-entropy of (sk_1, sk_2) by at most $2\lambda - 1$. We construct an adversary $\mathcal{A}$, that uses $\mathcal{B}$ to break security of $\mathcal{E}$, and uses a leakage function L' that reduces the min-entropy of the secret key by at most λ.

Intuitively (and, as we will show, falsely), it does not seem too hard to show such a reduction. It only makes sense that when the pair (sk_1, sk_2) looses 2λ bits of entropy, then at least one of the secret keys sk_1, sk_2 "loses" at most λ bits (otherwise the total loss should be more than 2λ). Therefore the adversary $\mathcal{A}$ can sample a key pair by itself and "plant" it either as (pk_1, sk_1) or as (pk_2, sk_2) (at random). Namely, $\mathcal{A}$ will sample a random $i \in \{1, 2\}$, and uniformly sample (pk_i, sk_i), the key pair of the scheme we actually wish to attack will play the role of (pk_{3-i}, sk_{3-i}). Upon receiving a leakage function $L(\cdot)$ from $\mathcal{B}$, the adversary $\mathcal{A}$ will plug the known (pk_i, sk_i) into the function and thus obtain L' to be sent to the challenger. Upon receiving a response from the challenger, it is forwarded back to $\mathcal{B}$, which can then break security with noticeable probability. Notice that $\mathcal{B}$'s view in the game is identical to its view in the repeated game against $\mathcal{E}^2$, and thus it still breaks the security with the same probability. The only worry is whether the function L' only reduces the key entropy by the allowed amount, which is unfortunately not the case. Assume that L leaks some 2λ bits on the bit-wise XOR $sk_1 \oplus sk_2$. Then when plugging in a known sk_i, the resulting L' still leaks 2λ bits on sk_{3-i}.

To solve this problem, we must prevent $\mathcal{A}$ from knowing sk_i. This is achieved by having the key pair (pk_i, sk_i) sampled by the leakage function L', rather than by $\mathcal{A}$. Namely, $L'(pk, sk)$ is now defined as follows: First, sample (pk_i, sk_i) and set $(pk_{3-i}, sk_{3-i}) = (pk, sk)$. Then run $y \leftarrow L(pk_1, pk_2, sk_1, sk_2)$ to obtain the leakage value. Lastly, output (y, pk_1, pk_2). Given the output of L', the adversary $\mathcal{A}$ can forward the value y to $\mathcal{B}$, that uses it to break the scheme, all without ever being exposed to the value of sk_i.

This seems to give $\mathcal{A}$ the least amount of information possible, so we should hope that now we can prove that the entropy of sk is reduced by at most λ. However, again, this is not true. Suppose that with probability $1/2$, the leakage function L outputs 2λ bits about sk_1 and with probability $1/2$ it outputs 2λ bits about sk_2. In this case, L indeed reduces the min-entropy of (sk_1, sk_2) by 2λ, and yet for every $i \in \{1, 2\}$ the leakage function $L'(pk, sk)$ reduces the min-entropy of sk by essentially 2λ as well, and thus is not a valid leakage function for the one shot game.

This abnormality results, to some extent, from using min-entropy (as opposed to Shannon entropy) as our entropy measure: If $L'(pk, sk)$ outputs both $y = L(pk_1, pk_2, sk_1, sk_2)$ and sk_{3-i}, then it would indeed leak at most λ bits on sk (with probability $1/2$). The fact that we have *less* information, namely sk_i is not known, might actually *decrease* the min-entropy of the key.

We arrive at a conflict: On one hand, knowing sk_i is a problem, but on the other, not knowing it seems to also be a problem. We show that revealing sk_i only in some cases, enables to prove parallel repetition. We use a simple lemma (Lemma A.1), which essentially shows how to "split-up" the joint min-entropy of two random variables. More precisely, it says that there is a subset S of all possible secret keys sk_1, such that for every $sk_1 \in S$, the the random variable $sk_2|sk_1$ has high min-entropy. Moreover, given the additional bit of information

that $sk_1 \notin S$, causes sk_1 to have high min-entropy (which decreases as the size of S shrinks).

We proceed by a specific analysis for each of our two relaxed models. For explanatory reasons, we first discuss leakage with advice (our second relaxation) and then go back to the almost leakage resilience model (our first relaxation).

Point-Wise λ-Leakage with Advice. In this model, the adversary $\mathcal{A}$ will leak $L'(pk, sk)$, which is a randomized leakage function, defined by choosing a random $\tau \in \{1, 2\}$, setting $(sk_\tau, pk_\tau) = (sk, pk)$, choosing a new fresh key pair (sk_i, pk_i), where $i = 3 - \tau$, and outputting $L(pk_1, pk_2, sk_1, sk_2)$. In addition, it will use one bit of advice which is whether $sk_i \in S$. If so, the leakage function $L'(pk, sk)$ outputs sk_i in addition to $L(pk_1, pk_2, sk_1, sk_2)$, and otherwise it outputs only $L(pk_1, pk_2, sk_1, sk_2)$. Now we can prove that indeed, for many pairs (pk, sk), the leakage $L'(pk, sk)$ leaks at most λ bits about sk (and $\mathcal{B}$ breaks $\mathcal{E}^2$ on the corresponding keys).

Note that the leakage function L' sometimes leaks more than it should. Namely, in some cases the value $y \leftarrow L'(pk, sk)$ reduces the min-entropy of sk by λ; but in other cases it reduces the min-entropy of sk by more than λ,[5] and in these cases it is an invalid leakage function. For this reason, we need to consider the *point-wise* λ-leakage definition. In addition, note that L' used only one bit of additional advice. Therefore when going from $\mathcal{E}$ to $\mathcal{E}^t$ the reduction uses $\log t$ bits of advice.

Point-Wise Almost λ-Leakage. In this model, the idea of the reduction is the following: The adversary $\mathcal{A}$ will leak $L'(sk, pk)$, which is a randomized leakage function, defined by choosing a random $\tau \in \{1, 2\}$, setting $(sk_\tau, pk_\tau) = (sk, pk)$, choosing a new fresh key pair (sk_i, pk_i), where $i = 3 - \tau$, and outputting $L(pk_1, pk_2, sk_1, sk_2)$, and in addition with probability $1/2$ outputting sk_i.

As in the model with advice, the leakage function L' might leak more than λ bits about sk, and thus we use the point-wise definition. In the analysis, we distinguish between the case that the set S is noticeable and the case that it is negligible. In the former, with non-negligible probability the leakage function L' will sample $sk_i \in S$ and will output it. In this case the leakage function is legal. If the set S is negligible, we claim the distribution of the secret key sk_τ is statistically close to the distribution of sk_τ conditioned on the event that $sk_i \notin S$ (as this event happens only with negligible probability). Therefore, if L' did not output the secret key sk_i, the secret key sk_τ is statistically close to a distribution with high enough min-entropy. Due to this analysis, we need to relax our leakage model *almost* λ-leakage resilient.

Since the analysis in this model is asymptotic, we are not able to extend it beyond a constant number of repetitions. See discussion in Section 5.

[5] This happens when the set S is very small, yet $sk_\tau \in S$.

1.4 Paper Organization

We define our generalized notion of public-key primitives in Section 2, where we also define parallel repetition and leakage attacks on such primitives. Our model of point-wise leakage resilience is presented in Section 3. In Section 4 we explain why all known leakage resilient schemes are also point-wise leakage-resilient.

Our parallel repetition theorems for a constant number of repetitions and for a polynomial number of repetitions are presented in Sections 5 and 6, respectively. In Section 7 we discuss what our theorems imply for schemes that are only computationally indistinguishable from being secure in our model. Appendix A contains the min-entropy splitting lemma that is used for all our proofs.

Due to space limitations, some proofs are omitted from this extended abstract and can be found in the full version [6].

2 Public-Key Primitives, Parallel-Repetition, Leakage Attacks

In this section we give a definition of a *public key primitive* which generalizes one-way relations and public-key encryption under chosen plaintext attack (CPA). We then show how to define parallel repetition with respect to public-key primitives in a way that, again, generalizes the intuitive notions of parallel repetition for either one-way relations or public-key encryption.

2.1 A Unified Framework for Public-Key Primitives

We use the following formalization that generalizes both one-way relations and public-key encryption.

Definition 2.1 (public-key primitive). *A* public-key primitive $\mathcal{E} = (G, V)$ *is a pair of* PPT *algorithms such that*

- *The* key generator G *generates a pair of secret and public keys:* $(sk, pk) \leftarrow G(1^k)$.
- *The* verifier V *is an oracle machine such that* $V^{\mathcal{O}(pk)}(pk)$ *either accepts or rejects.*

Definition 2.2 (secure public-key primitive). *A public-key primitive* $\mathcal{E} = (G, V)$ *is* secure *if for any* PPT *oracle* break*, it holds that*

$$\Pr_{(sk,pk)\leftarrow G(1^k)}[V^{\mathsf{break}(pk)}(pk)] = \mathrm{negl}(k) \ .$$

To be concrete, for one-way relations, the breaker needs to send a candidate secret key sk (= inversion of the public key), and the verifier runs the relation's verification procedure. To see why public key encryption can be stated in these terms, requires some work. The reason it is not immediate is that typically, we would consider the interaction between the verifier and the breaker, to be the following: The verifier gives the breaker a challenge ciphertext $\mathsf{Enc}_{pk}(b)$, and he

accepts if the breaker succeeds in guessing b. However, the breaker can clearly cause the verifier to accept with probability $1/2$, where we need to ensure that the breaker succeeds only with negligible probability. This technical annoyance can be fixed by considering the game where the verifier sends $\text{poly}(k)$ challenge ciphertexts to the breaker, each encrypting a random bit. The breaker succeeds if it succeeded in guessing significantly more than $1/2$ of the bits encrypted. The formal definition and precise analysis are much more cumbersome. The proof appears in the full version [6].

Note that our verifier (which corresponds to the challenger in "security game" based definitions) only gets the public key as input and not the secret key. If the secret key was also given, then all public-key encryption schemes, signature schemes, and one-way relations, would trivially fit into this framework. However, in this work, we only consider primitives where the verifier V does not use the secret key sk to verify, but uses only the public key pk. An example of such a primitive is public-key encryption (under CPA). However, signature schemes or CCA secure encryption schemes do not fall into this category, since for these primitives the verifier in the definition above does need to know the secret key sk in order to simulate the signing oracle, in the case of signature schemes, and to simulate the decryption oracle, in the case of CCA encryption schemes.

2.2 Parallel Repetition

Definition 2.3 (t-parallel repetition). *For any public-key primitive $\mathcal{E} = (G, V)$ and any $t \in \mathbb{N}$, its t-*parallel repetition*, denoted $\mathcal{E}^t = (G^t, V^t)$, is in itself a public-key primitive defined as follows*

- *The key generator $(sk^t, pk^t) \leftarrow G^t(1^k)$ generates $(sk_i, pk_i) \leftarrow G(1^k)$ for all $i \in [t]$ and outputs $sk^t \triangleq (sk_1, \ldots, sk_t)$, $pk^t \triangleq (pk_1, \ldots, pk_t)$.*
- *The verifier $(V^t)^{\mathcal{O}(pk^t)}(pk^t)$, runs $V^{\mathcal{O}(pk^t, i)}(pk_i)$ for all $i \in [t]$, and accepts if and only if they all accept.*

A direct product of t schemes $\mathcal{E}_1 \times \cdots \times \mathcal{E}_t$ is defined similarly.

While it is straightforward that our definition captures the notion of parallel repetition for one-way relations (where the goal is to find legal pre-images for all input public-keys), let us be a little more explicit about how the above captures parallel repetition for public-key encryption.

Lemma 2.4. *Let $\mathcal{E} = (G, V)$ be a public-key primitive that represents a public-key encryption scheme and let $t \in \mathbb{N}$. Then there exists a public key encryption scheme that is represented by $\mathcal{E}^t$.*

Moreover, this scheme is obtained by secret sharing the message into t shares and encrypting share i with pk_i. To decrypt, decrypt all shares and restore the message.

The proof is straightforward and is omitted.

2.3 Leakage Attacks

In this section, we generalize the notion of leakage attacks to our public-key primitive framework. Note that we do not define what it means for a scheme to be secure, only present a model for an attack.

Definition 2.5 (leakage attack). *We consider adversaries of the form* $\mathcal{A} = (\mathsf{leak}_{\mathcal{A}}, \mathsf{break}_{\mathcal{A}})$*, where* $\mathsf{leak}_{\mathcal{A}}$*,* $\mathsf{break}_{\mathcal{A}}$ *are (possibly randomized) functions. We refer to* $\mathsf{leak}_{\mathcal{A}}$ *as the* leakage function *and to* $\mathsf{break}_{\mathcal{A}}$ *as the* breaker*.*

A leakage attack *of an adversary* $\mathcal{A} = (\mathsf{leak}_{\mathcal{A}}, \mathsf{break}_{\mathcal{A}})$ *on a public-key primitive* $\mathcal{E} = (G, V)$ *(with security parameter* k*) is the following process.*

- **Initialize:** *Generate a key pair* $(sk, pk) \xleftarrow{\$} G(1^k)$*.*
- **Leak:** *Apply the leakage function on the key pair to obtain the* leakage value $y \leftarrow \mathsf{leak}_{\mathcal{A}}(pk, sk)$*.*
- **Break:** $\mathcal{A}$ succeeds *if* $V^{\mathsf{break}(pk,y)}(pk)$ *accepts.*

3 Point-Wise Leakage Resilience

In this work, we consider "noisy leakage" functions, which are only allowed to reduce the (average) min-entropy of the secret key by a bounded amount. However, we relax the min-entropy restriction, and consider a *point-wise* definition, where we require that the specific leakage value is legal (as opposed to requiring that the leakage function is *always* legal).

We define our new model below. Then, in Sections 3.1, 3.2, we present two relaxed versions of point-wise leakage resilience that we need in order to prove our parallel repetition theorems. Finally, in Section 3.3 we show that all of these notions are strictly stronger than the old bounded-leakage model of [1]. Namely, security w.r.t. to our definitions imply, as a special case, security w.r.t. bounded leakage.

Definition 3.1 (point-wise λ-leakage). *Let* $\mathcal{E} = (G, V)$ *be a public key primitive. A possibly randomized leakage function* leak *is* λ-leaky at point (pk, y)*, where* pk *is a public key and* y *is a leakage value (in the image of* leak*), if*

$$\mathbf{H}_{\infty}(S_{pk,y}) \geq \mathbf{H}_{\infty}(S_{pk}) - \lambda \ ,$$

where S_{pk} *is the distribution of secret keys conditioned on the public key being* pk*, and* $S_{pk,y}$ *is the distribution of secret keys conditioned on both the public key being* pk *and on* $\mathsf{leak}(pk, sk) = y$*.*

Definition 3.2 (point-wise λ-leakage resilience). *A public-key primitive* $\mathcal{E} = (G, V)$ *is* point-wise λ-leakage-resilient *if for any* PPT *adversary* $\mathcal{A}$*, where* $\mathcal{A} = (\mathsf{leak}_{\mathcal{A}}, \mathsf{break}_{\mathcal{A}})$*, it holds that*

$$\mathrm{Adv}_{\mathcal{E},\lambda}[\mathcal{A}] \triangleq \Pr\left[(\mathsf{leak}_{\mathcal{A}} \text{ is } \lambda\text{-leaky at } (pk, y)) \wedge (\mathcal{A}(pk, y) \text{ succeeds})\right] = \mathrm{negl}(k) \ ,$$

where the probability is taken over $(sk, pk) \leftarrow G(1^k)$*, over the random coin tosses of* $\mathcal{A} = (\mathsf{leak}_{\mathcal{A}}, \mathsf{break}_{\mathcal{A}})$*, and over the random coin tosses of the verifier in the verification game.*

In order to obtain our direct product theorems for leakage resilience, we relax the point-wise leakage resilience definition in two (incomparable) ways.

3.1 First Relaxation: Almost Leakage Resilience

In this relaxation, instead of requiring that sk has high min-entropy conditioned on pk and $y = \mathsf{leak}(pk, sk)$, we require that the distribution of sk (conditioned on pk, y) is statistically close to one that has high min-entropy.

Definition 3.3 (close to λ-leaky). *A leakage function* leak *is* μ-close to λ-leaky at point (pk, y) *if there exists a distribution* $\tilde{S}_{pk,y}$ *that is* μ*-close to* $S_{pk,y}$ *and*

$$\mathbf{H}_\infty(\tilde{S}_{pk,y}) \geq \mathbf{H}_\infty(S_{pk}) - \lambda \ .$$

Definition 3.4 (resilience to almost λ-leakage). $\mathcal{E} = (G, V)$ *is* point-wise almost λ-leakage-resilient *if for any* PPT *adversary* $\mathcal{A} = (\mathsf{leak}_{\mathcal{A}}, \mathsf{break}_{\mathcal{A}})$ *and for any negligible function* μ*, it holds that*

$$\mathrm{Adv}_{\mathcal{E},\lambda,\mu}[\mathcal{A}] \triangleq \Pr\left[(\mathsf{leak}_{\mathcal{A}} \text{ is } \mu\text{-close to } \lambda\text{-leaky at } (pk, y)) \wedge (\mathcal{A}(pk, y) \text{ succeeds})\right] = \mathrm{negl}(k) \ .$$

where the probability is taken over $(sk, pk) \leftarrow G(1^k)$*, over the random coin tosses of* $\mathcal{A} = (\mathsf{leak}_{\mathcal{A}}, \mathsf{break}_{\mathcal{A}})$*, and over the random coin tosses of the verifier in the verification game.*

Under this definition we obtain a direct-product theorem for constant number of repetitions.

3.2 Second Relaxation: Leakage Resilience with Advice

To obtain a direct-product theorem for a super-constant number of repetitions, we use a slightly different (and incomparable) model, where we do not allow statistical closeness, but rather allow the attacker to get a logarithmic number of bits of (possibly inefficient) advice.

Definition 3.5 (PPT$_{\text{-}a}$). *We say that a function* f *is* PPT$_{\text{-}a}$ *computable if the function* $f_{\text{-}a}$*, defined below, is* PPT *computable. The function* $f_{\text{-}a}$ *is identical to* f*, except that the last* a *bits of its output are truncated.*

We say that an adversary $\mathcal{A} = (\mathsf{leak}_{\mathcal{A}}, \mathsf{break}_{\mathcal{A}})$ *is a* PPT$_{\text{-}a}$ *adversary if* $\mathsf{leak}_{\mathcal{A}}$ *is* PPT$_{\text{-}a}$ *computable and* $\mathsf{break}_{\mathcal{A}}$ *is* PPT *computable.*

Definition 3.6 (point-wise λ-leakage with advice). *A public-key primitive* $\mathcal{E} = (G, V)$ *is resilient to* point-wise λ-leakage and logarithmic advice *if for any* PPT$_{\text{-}O(\log k)}$ *adversary* $\mathcal{A} = (\mathsf{leak}_{\mathcal{A}}, \mathsf{break}_{\mathcal{A}})$ *it holds that*

$$\mathrm{Adv}_{\mathcal{E},\lambda}[\mathcal{A}] \triangleq \Pr\left[(\mathsf{leak}_{\mathcal{A}} \text{ is } \lambda\text{-leaky at } (pk, y)) \wedge (\mathcal{A}(pk, y) \text{ succeeds})\right] = \mathrm{negl}(k) \ ,$$

where the probability is taken over $(sk, pk) \leftarrow G(1^k)$*, over the random coin tosses of* $\mathcal{A} = (\mathsf{leak}_{\mathcal{A}}, \mathsf{break}_{\mathcal{A}})$*, and over the random coin tosses of the verifier in the verification game.*

3.3 Relation to Bounded Leakage

To conclude, we prove that point-wise λ-leakage resilience implies the basic form of λ-bounded leakage. A proof sketch appears in the full version [6].

Definition 3.7 ([1]). *A public-key primitive $\mathcal{E} = (G, V)$ is* λ-bounded leakage resilient *if any* PPT *adversary $\mathcal{A} = (\mathsf{leak}_{\mathcal{A}}, \mathsf{break}_{\mathcal{A}})$ for which the output of $\mathsf{leak}_{\mathcal{A}}$ is at most λ bits, succeeds with negligible probability.*

Lemma 3.8. *If $\mathcal{E} = (G, V)$ is point-wise λ-leakage resilient then it is also λ-bounded leakage resilient.*

We note that point-wise almost λ-leakage resilience, and λ-leakage resilience with logarithmic advice, are stronger notions of security (they give the adversary more power) and thus the above immediately applies to these notions as well.

4 Why Known Schemes are Point-Wise Leakage Resilient

In this section, we show that leakage resilience is amplified by parallel repetition for, essentially, all known schemes that are resilient to non-trivial (i.e. super-logarithmic) leakage. To show this, we sketch a proof template that is shared among all (non trivial) leakage resilient results, and we show that this proof template proves security also w.r.t. our leakage models (the point-wise almost λ-leakage model, and the point-wise λ-leakage with logarithmic advice model).

The Proof Template. The proof template for proving leakage resilience is very simple, and works in two hybrid steps. Recall that the adversary first gets a pair $(pk, y = L(pk, sk))$, where L is a poly-size leakage function chosen by $\mathcal{A}$. Then it chooses messages m_0, m_1 and gets a challenge ciphertext $c_b \leftarrow \mathsf{Enc}_{pk}(m_b)$. The adversary wins if it guesses the bit b correctly.

The first step in the template is to replace the challenge c_b with an "illegally" generated ciphertext c_b^*, such that $(sk, pk, c_b) \overset{c}{\approx} (sk, pk, c_b^*)$ (and it is efficient to generate c_b^* given sk, pk, b). Due to computational indistinguishability, the adversary's success probability should remain unchanged. We note that there is no entropy involved in this part, only a requirement that L is efficiently computable.

The second step is completely information theoretic: It is proven that if the distribution of the secret key conditioned on pk, y, which we denote by $S_{pk,y}$, has sufficient min-entropy, then c_b^* carries no information on b (or, more precisely, that conditioned on the view of the adversary, b is statistically close to uniform). Therefore, no adversary can guess its value with non-negligible advantage.

Point-Wise Leakage Resilience. The above proof template also proves point-wise leakage resilience. The second step of the hybrid works in a point-wise manner and therefore we only need to worry about the first step. In the first step, clearly computational indistinguishability still holds, but proving that the point-wise advantage remains unchanged is a bit harder, since we cannot efficiently check

the point-wise advantage. Nevertheless, we argue that if the advantage of $\mathcal{A}$ is non-negligible, then it drops by a factor of at most two. Such a claim is sufficient for the next level of the template.

To see why this is the case, consider an adversary $\mathcal{A}$ that has non-negligible point-wise advantage ϵ when given (pk, y, c_b), but less than $\epsilon/2$ when given (pk, y, c_b^*). Recall that the advantage measures the probability of both $\mathcal{A}$ succeeding (in the verification game) and pk, y being point-wise λ-leaky. It follows that with non-negligible probability over pk, y, the conditional success probability of $\mathcal{A}$, conditioned on pk, y, drops by at least $\epsilon/4$ (otherwise the advantage, which measures over a subset of the pk, y, couldn't have dropped).

A distinguisher $\mathcal{B}(sk, pk, c_b/c_b^*)$ is defined as follows: First, compute the leakage $y{:=}L(sk, pk)$. Then generate many samples of c_b/c_b^* and use them to evaluate the success probability of $\mathcal{A}$ conditioned on pk, y in the two cases. If indeed pk, y are such that the success probability drops, use $\mathcal{A}$ to distinguish between the two cases. If no noticeable change in the success probability was noticed, then output a random guess. Putting it all together, we get a polynomial distinguisher between (pk, y, c_b) and (pk, y, c_b^*), in contradiction to the hardness assumption.

We note that this is true even if y is not fully known to the distinguisher: say $O(\log k)$ bits of y are not known, the distinguisher can still try all options and check if for either one the success probability changes by $\epsilon/4$.

Our Relaxed Models of Point-Wise Leakage Resilience. Our first relaxation, of allowing the secret key to be statistically close to λ-leakage resilient, only effects the second step of the template. We can still argue that b is statistically close to uniform by adding another hybrid where the conditional distribution $S_{pk,y}$ is replaced with a statistically indistinguishable $\tilde{S}_{pk,y}$ that has high min-entropy.

Our second relaxation, of allowing logarithmic advice, goes into the first step (this is the only step where we care about the complexity of L). As we explained above, our argument works even if a logarithmic part of the leakage value is not known. Therefore we will use only the efficient part of the leakage function and computational indistinguishability will still hold.

Computationally Indistinguishable Schemes. For some schemes, such as [1, 16], leakage resilient is proven by showing that they are computationally indistinguishable from another scheme which, in turn, is proven leakage resilient using the template. We show in Section 7 that this still implies that parallel repetition amplifies leakage.

5 Direct-Product Theorem for a Constant Number of Repetitions

In this section, we prove a direct-product theorem for a constant number of repetitions, w.r.t. point-wise almost leakage-resilience as defined in Section 3.1.

Theorem 5.1. *Let $c \in \mathbb{N}$ be a constant, and for every $i \in [c]$, let $\mathcal{E}_i = (G_i, V_i)$ be a point-wise almost λ_i-leakage-resilient public-key primitive. Then, $\mathcal{E}_1 \times \ldots \times \mathcal{E}_c$ is point-wise almost λ-leakage-resilient, where $\lambda = \sum_{i=1}^{c}(\lambda_i - 1)$.*

It suffices to prove this theorem for $c = 2$, and apply it successively. In order to simplify notation, we prove it for the case of parallel repetition, where $\mathcal{E}_1 = \mathcal{E}_2$, the proof extends readily to the case of direct product.

Lemma 5.2. *Let $\mathcal{E} = (G, V)$ be a point-wise almost λ-leakage-resilient public-key primitive. Then $\mathcal{E}^2$ is point-wise almost $(2\lambda - 1)$-leakage-resilient.*

Before we present the outline of the proof, let us make a few remarks.

1. Note that there is a loss of one bit in the amplification. Namely, we go from λ to $(2\lambda - 1)$ instead of just 2λ. While some loss in the parameters is implied by our techniques, more detailed analysis can show that the composed scheme is in fact $(2\lambda - \delta)$-leakage resilient for any $\delta(k) = 1/\mathrm{poly}(k)$. Thus the loss incurred is less than a single bit. As our result is qualitative in nature, we chose not to overload with the additional complication.
2. While at first glance one could imagine that Theorem 5.1 should extend beyond constant c, we were unable to prove such an argument. The reason is that super-constant repetition gives a different scheme for each value of the security parameter. This means that we cannot use Theorem 5.1 as black-box. More importantly, our proof techniques rely on the asymptotic behavior of the scheme so we were not able to even change the proof to apply for a super-constant number of repetitions.
   A result for the more general case of any polynomial number of repetitions is presented, in the slightly different and incomparable "advice" model, in Section 6.
   Finally, we remark that known negative results for security of parallel repetition are already effective for a constant number of repetitions. Thus our result contrasts them even for this case.

Proof overview of Lemma 5.2. We consider an adversary $\mathcal{B}$ that succeeds in the parallel repetition game, and construct an adversary $\mathcal{A}$ that succeeds in the single instance game. The straightforward proof strategy would be to "plant" the "real" key pair, that is given as input to $\mathcal{A}$, as one of the key pairs that are input to $\mathcal{B}$, and sample the other pair uniformly.[6] In such case, the input to $\mathcal{B}$ is distributed identically to the parallel repetition case and indeed $\mathcal{B}$ will succeed with noticeable probability. However, we may no longer be able to claim that our leakage leaves sufficient entropy in the secret key. We are guaranteed by the functionality of $\mathcal{B}$ that the key pair (sk_1, sk_2) is left with sufficient min-entropy but it is still possible that neither sk_1 nor sk_2 have *any* min-entropy by themselves.

[6] We note that even this step is impossible when relying on "secretly generated" public parameters as in the scheme presented in [16] (or rather, the scheme that is computationally indistinguishable to theirs and actually has entropic leakage resilient).

To solve the above we use Lemma A.1, which essentially says how to split-up the joint entropy of two random variables. Specifically it says that either sk_1 or $sk_2|sk_1$ will have sufficient min-entropy, depending on whether sk_1 belongs to a hard-to-recognize set R, and conditioned on the knowledge of whether $sk_1 \in R$. Namely, either $sk_1|\mathbb{1}_{sk_1 \in R}$ or $sk_2|(sk_1, \mathbb{1}_{sk_1 \in R})$ have high min-entropy. If we could compute the bit $\mathbb{1}_{sk_1 \in R}$, we would have been done (and indeed if we are allowed one bit of inefficient leakage, an easier proof follows, see Section 6). Since this is impossible, we turn to case analysis:

Obviously, if $\Pr[sk_1 \in R] = \text{negl}(k)$, then we can always guess that $\mathbb{1}_{sk_1 \in R} = 0$ and be right almost always. This implies that in such case $sk_2|sk_1$ is statistically indistinguishable from having high min-entropy, as we wanted.

For the second case, if $\Pr[sk_1 \in R] \geq 1/\text{poly}(k)$, then $sk_2|(sk_1, \mathbb{1}_{sk_1 \in R})$ will have high min-entropy for a noticeable part of the time. To complete the analysis here, we notice that

$$\mathbf{H}_\infty(sk_2|(sk_1, \mathbb{1}_{sk_1 \in R})) = \mathbf{H}_\infty(sk_2|sk_1).$$

This is because R is a well defined set and thus $\mathbb{1}_{sk_1 \in R}$ is a deterministic (though hard to compute) function of sk_1. It follows that $sk_2|sk_1$ will have high min-entropy for a noticeable fraction of the time, which completes the proof.

For the formal proof, see the full version [6].

6 Direct-Product Theorem for Polynomially Many Repetitions

In this section we present a direct product theorem that applies to any polynomial number of repetitions. This theorem is relative to the advice model defined in Section 3.2. For the sake of simplicity, we will assume that the number of repetitions is a power of 2, although the same techniques can be used for any number.

Theorem 6.1. *Let $\mathcal{E} = (G, V)$ be a public-key primitive that is resilient to point-wise λ-leakage and logarithmic advice. Let $t = t(k)$ be a polynomially bounded function of the security parameter such that $t(k)$ is always a power of 2. Then $\mathcal{E}^t$ is resilient to point-wise $t(\lambda - 1)$-leakage and logarithmic adivce.*

Towards proving the theorem, we present the following lemma, which is a parameterized special case of the above theorem, and will imply the theorem by successive applications.

Lemma 6.2. *For any public-key primitive $\mathcal{E} = (G, V)$ and any $\text{PPT}_{\text{-}a}$ adversary $\mathcal{B} = (\mathsf{leak}_{\mathcal{B}}, \mathsf{break}_{\mathcal{B}})$ for $\mathcal{E}^2$, there exists a $\text{PPT}_{\text{-}(a+1)}$ adversary $\mathcal{A} = (\mathsf{leak}_{\mathcal{A}}, \mathsf{break}_{\mathcal{A}})$ for $\mathcal{E}$, such that for all k,*

$$\text{Adv}_{\mathcal{E},\lambda}[\mathcal{A}] \geq (1/4) \cdot \text{Adv}_{\mathcal{E}^2,(2\lambda-1)}[\mathcal{B}] \ .$$

The theorem immediately follows by applying the lemma $\log t$ times. See proofs in the full version [6].

7 Leakage from Computationally Indistinguishable Schemes

Our definition of point-wise leakage resilience is based on the residual min-entropy of the secret key, conditioned on the leakage value. In the literature, starting with [18], this is referred to as "resilience to *noisy* leakage". It is self evident that schemes where the public key is an injective function of the secret key cannot be proven leakage resilient in this respect. This is because even leaking the secret key in its entirety, which obviously breaks security, does not reduce its min entropy conditioned on the public key (the conditional min-entropy is 0 to begin with, and it stays 0 after the leakage). We do know, however, of such injective public-key encryption schemes that are proven to be leakage resilient with respect to the weaker notion of "length bounded leakage". There, the restriction on the leakage function is that it has bounded length. Notable examples are the scheme of [1] and the scheme of [16] (which was introduced as a counterexample for parallel repetition of length-bounded leakage resilience, see Section 1.2). While at first glance it may seem that our result is completely powerless with regards to such schemes, we show in this section that for all known schemes, and specifically for the schemes of [1, 16], our theorem in fact *does* imply parallel repetition.

The key observation upon revisiting the proofs of security of [1, 16], is that in both cases, the proof is by presenting a second scheme in which the key distribution is computationally indistinguishable from the original scheme (but may have undesired features such as worse efficiency of key generation), and proving that this second scheme is resilient to leakage of bounded length. This implies that the original scheme is resilient to bounded leakage as well (since otherwise one can distinguish the key generation processes). The second scheme, in these two cases, is in fact resilient to noisy leakage. Furthermore, the second scheme in the two cases adheres to our notion of point-wise leakage resilience.

In light of the above, we put forth the following corollary of Theorems 5.1 and 6.1.

Corollary 7.1. *Let $\mathcal{E} = (G, V)$ be a public-key primitive, and let G' be such that $G(1^k) \overset{c}{\approx} G'(1^k)$. Then:*

1. *If $\mathcal{E}' = (G', V)$ is point-wise almost λ-leakage resilient, then $\mathcal{E}^t$ is $t \cdot (\lambda - 1)$-bounded leakage resilient for any constant $t \in \mathbb{N}$.*
2. *If $\mathcal{E}' = (G', V)$ is point-wise λ-leakage resilient with logarithmic advice, then $\mathcal{E}^t$ is $t \cdot (\lambda - 1)$-bounded leakage resilient for any polynomial $t = t(k)$.*

Proof. The proof of the two parts is almost identical: We use either Theorem 5.1 or Theorem 6.1 to show that $(\mathcal{E}')^t$ is point-wise almost $t \cdot (\lambda - 1)$-leakage resilient, or, respectively, leakage resilient with logarithmic advice. By Lemma 3.8, this means that $(\mathcal{E}')^t$ is $t \cdot (\lambda - 1)$-bounded leakage resilient.

By a hybrid argument $(G')^t(1^k) \stackrel{c}{\approx} G^t(1^k)$.[7] Therefore, it must be that $\mathcal{E}^t$ is also $t \cdot (\lambda - 1)$-bounded leakage resilient (otherwise there is a distinguisher between the key generators). This completes the proof.

Using Corollary 7.1, we can show that t-parallel repetition of the schemes of [1, 16] indeed amplifies their leakage resilience.

References

[1] A. Akavia, S. Goldwasser, and V. Vaikuntanathan. Simultaneous hardcore bits and cryptography against memory attacks. In O. Reingold, editor, *TCC*, volume 5444 of *Lecture Notes in Computer Science*, pages 474–495. Springer, 2009.

[2] J. Alwen, Y. Dodis, M. Naor, G. Segev, S. Walfish, and D. Wichs. Public-key encryption in the bounded-retrieval model. In H. Gilbert, editor, *EUROCRYPT*, volume 6110 of *Lecture Notes in Computer Science*, pages 113–134. Springer, 2010.

[3] J. Alwen, Y. Dodis, and D. Wichs. Leakage-resilient public-key cryptography in the bounded-retrieval model. In S. Halevi, editor, *CRYPTO*, volume 5677 of *Lecture Notes in Computer Science*, pages 36–54. Springer, 2009.

[4] E. Boyle, G. Segev, and D. Wichs. Fully leakage-resilient signatures. In K. G. Paterson, editor, *EUROCRYPT*, volume 6632 of *Lecture Notes in Computer Science*, pages 89–108. Springer, 2011.

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

[6] Z. Brakerski and Y. T. Kalai. A parallel repetition theorem for leakage resilience. Cryptology ePrint Archive, Report 2011/250, 2011. `http://eprint.iacr.org/`.

[7] Z. Brakerski and G. Segev. Personal communication, 2009.

[8] I. B. Damgård, S. Fehr, R. Renner, L. Salvail, and C. Schaffner. A tight high-order entropic quantum uncertainty relation with applications. In *Proceedings of the 27th annual international cryptology conference on Advances in cryptology*, CRYPTO'07, pages 360–378, Berlin, Heidelberg, 2007. Springer-Verlag.

[9] Y. Dodis, S. Goldwasser, Y. T. Kalai, C. Peikert, and V. Vaikuntanathan. Public-key encryption schemes with auxiliary inputs. In D. Micciancio, editor, *TCC*, volume 5978 of *Lecture Notes in Computer Science*, pages 361–381. Springer, 2010.

[10] Y. Dodis, Y. T. Kalai, and S. Lovett. On cryptography with auxiliary input. In M. Mitzenmacher, editor, *STOC*, pages 621–630. ACM, 2009.

[11] S. Dziembowski and K. Pietrzak. Leakage-resilient cryptography. In *FOCS*, pages 293–302. IEEE Computer Society, 2008.

[12] S. Faust, E. Kiltz, K. Pietrzak, and G. N. Rothblum. Leakage-resilient signatures. In D. Micciancio, editor, *TCC*, volume 5978 of *Lecture Notes in Computer Science*, pages 343–360. Springer, 2010.

[13] S. Garg, A. Jain, and A. Sahai. Leakage-resilient zero knowledge. In P. Rogaway, editor, *CRYPTO*, volume 6841 of *Lecture Notes in Computer Science*, pages 297–315. Springer, 2011.

[7] It is important to note that this will not be necessarily true if either G or G' relies on secret parameters. This relates to our discussion of the result of [16] in Section 1.2.

[14] A. Jain and K. Pietrzak. Parallel repetition for leakage resilience amplification revisited. In Y. Ishai, editor, *TCC*, volume 6597 of *Lecture Notes in Computer Science*, pages 58–69. Springer, 2011.
[15] J. Katz and V. Vaikuntanathan. Signature schemes with bounded leakage resilience. In M. Matsui, editor, *ASIACRYPT*, volume 5912 of *Lecture Notes in Computer Science*, pages 703–720. Springer, 2009.
[16] A. B. Lewko and B. Waters. On the insecurity of parallel repetition for leakage resilience. In L. Trevisan, editor, *FOCS*, pages 521–530. IEEE Computer Society, 2010.
[17] T. Malkin, I. Teranishi, Y. Vahlis, and M. Yung. Signatures resilient to continual leakage on memory and computation. In Y. Ishai, editor, *TCC*, volume 6597 of *Lecture Notes in Computer Science*, pages 89–106. Springer, 2011.
[18] M. Naor and G. Segev. Public-key cryptosystems resilient to key leakage. In S. Halevi, editor, *CRYPTO*, volume 5677 of *Lecture Notes in Computer Science*, pages 18–35. Springer, 2009.
[19] K. Pietrzak. A leakage-resilient mode of operation. In A. Joux, editor, *EUROCRYPT*, volume 5479 of *Lecture Notes in Computer Science*, pages 462–482. Springer, 2009.
[20] J. Wullschleger. Oblivious-transfer amplification. In M. Naor, editor, *EUROCRYPT*, volume 4515 of *Lecture Notes in Computer Science*, pages 555–572. Springer, 2007.

A How to Split Min-Entropy

We present a lemma that shows that the joint min-entropy of two random variables can be split between them under some condition. Variants of this lemma appeared in previous works (e.g. [8, 20]), this formulation is from [7].

Lemma A.1 (min-entropy split). *Let X, Y be such that $\mathbf{H}_\infty(X, Y) \geq a + b$, for $a, b > 0$. Then there exists a set R_X, which is a subset of the support of X such that both:*

1. *For all $x \in R_X$, it holds that $\mathbf{H}_\infty(Y|X = x) \geq b$.*
2. *$\mathbf{H}_\infty(X|X \notin R_X) \geq a - \log(1/\epsilon)$, where $\epsilon \triangleq \Pr[X \notin R_X]$.*

Proof. Define

$$R_X \triangleq \{x : \Pr[X = x] \geq 2^{-a}\} .$$

Then for all $x \in R_X$ and for all y, it holds that $\Pr[Y = y|X = x] \leq 2^{-b}$, and thus $\mathbf{H}_\infty(Y|X = x) \geq b$. In addition, $\mathbf{H}_\infty(X|X \notin R_X) \geq a + \log \Pr[X \notin R_X]$, i.e. $\mathbf{H}_\infty(X|X \notin R_X) \geq a - \log(1/\epsilon)$.