Leftover Hash Lemma, Revisited

Boaz Barak[1], Yevgeniy Dodis[2], Hugo Krawczyk[3], Olivier Pereira[4], Krzysztof Pietrzak[5], Francois-Xavier Standaert[4], and Yu Yu[6]

[1] Microsoft Research New England. Email: boaz@microsoft.com.
[2] New York University. Email: dodis@cs.nyu.edu.
[3] IBM Research. Email: hugo@ee.technion.ac.il.
[4] Université Catholique de Louvain.
Email: {Olivier.Pereira,fstandae}@uclouvain.be.
[5] CWI Amsterdam. Email: pietrzak@cwi.nl.
[6] East China Normal University. Email: yuyu@yuyu.hk.

Abstract. The famous *Leftover Hash Lemma* (LHL) states that (almost) universal hash functions are good randomness extractors. Despite its numerous applications, LHL-based extractors suffer from the following two limitations:

- **Large Entropy Loss:** to extract v bits from distribution X of min-entropy m which are ε-close to uniform, one must set $v \le m - 2\log(1/\varepsilon)$, meaning that the entropy loss $L \overset{\text{def}}{=} m - v \ge 2\log(1/\varepsilon)$. For many applications, such entropy loss is too large.
- **Large Seed Length:** the seed length n of (almost) universal hash function required by the LHL must be at least $n \ge \min(u - v, v + 2\log(1/\varepsilon)) - O(1)$, where u is the length of the source, and must grow with the number of extracted bits.

Quite surprisingly, we show that both limitations of the LHL — large entropy loss and large seed — can be overcome (or, at least, mitigated) in various important scenarios. First, we show that entropy loss could be reduced to $L = \log(1/\varepsilon)$ for the setting of deriving secret keys for a wide range of cryptographic applications. Specifically, the security of these schemes with an LHL-derived key gracefully degrades from ε to at most $\varepsilon + \sqrt{\varepsilon 2^{-L}}$. (Notice that, unlike standard LHL, this bound is meaningful even when one extracts more bits than the min-entropy we have!) Based on these results we build a general *computational extractor* that enjoys low entropy loss and can be used to instantiate a generic key derivation function for *any* cryptographic application.

Second, we study the soundness of the natural *expand-then-extract* approach, where one uses a *pseudorandom generator* (PRG) to expand a short "input seed" S into a longer "output seed" S', and then use the resulting S' as the seed required by the LHL (or, more generally, by any randomness extractor). We show that, in general, the expand-then-extract approach is not sound if the Decisional Diffie-Hellman assumption is true. Despite that, we show that it is sound either: (1) when extracting a "small" (logarithmic in the security of the PRG) number of bits; or (2) in minicrypt. Implication (2) suggests that the expand-then-extract approach is likely secure when used with "practical" PRGs, despite lacking a reductionist proof of security!

1 Introduction

The famous Leftover Hash Lemma [18] (LHL; see also [18] for earlier formulations) has found a huge number of applications in many areas of cryptography and complexity theory. In its simplest form, it states that universal hash functions [7] are good (strong) randomness extractors [31]. Specifically, if X is a distribution of min-entropy m over some space $\mathcal{X}$, $\mathcal{H}$ is a family of universal functions (see Definition 2) from $\mathcal{X}$ to $\{0,1\}^v$, and H is a random member of $\mathcal{H}$, then, *even conditioned on the "seed"* H, the statistical distance between $H(X)$ and the uniform distribution U_v on $\{0,1\}^v$ is bounded by $\sqrt{2^{-L}}$, where $L \overset{\mathsf{def}}{=} m - v$. The parameter L is defined as the *entropy loss* and it measures the amount of min-entropy "sacrificed" in order to achieve good randomness extraction. Thus, no application can tell apart the "extracted" randomness $H(X)$ from uniform randomness U_v, with advantage greater than $\varepsilon \overset{\mathsf{def}}{=} \sqrt{2^{-L}}$, even if the seed H is published (as long as H is independent of X).

The LHL is extremely attractive for many reasons. First, and foremost, it leads to simple and efficient randomness extractors, and can be used in many applications requiring good secret randomness. One such major setting is that of cryptographic key derivation, which is needed in many situations, such as privacy amplification [4], Diffie-Hellman key exchange [14,25], biometrics [11,5] and random number generators from physical sources of randomness [3,2]. Second, many simple functions, such as the inner product or, more generally, matrix-vector multiplication, are universal. Such elegant functions have nice algebraic properties which can be used for other reasons beyond randomness extraction (for a few examples, see [26,10,29]). Third, many simple and efficient constructions of (almost) universal hash functions are known [7,37,30,24], making LHL-based extractors the most efficient extractors to date. Finally, LHL achieves the optimal value of the entropy loss $L = m - v$ sufficient to achieve the desired statistical distance ε. Specifically, LHL achieves $L = 2\log(1/\varepsilon)$, which is known to be the the smallest possible entropy loss for *any* extractor [34].

Despite these extremely attractive properties, LHL-based extractors are not necessarily applicable or sufficient in various situations. This is primarily due to the following two limitations of the LHL: large entropy loss and large seed.

Large Entropy Loss. In theory, the entropy loss of $2\log(1/\varepsilon)$ might appear quite insignificant, especially in the asymptotic sense. However, in practical situations it often becomes a deal-breaker, especially when applied to the setting of key derivation. In this case the main question is to determine the smallest min-entropy value m sufficient to extract a v-bit key with security ε. Minimizing this value m, which we call *startup entropy*, is often of critical importance, especially in entropy constrained scenarios, such as Diffie-Hellman key exchange (especially on elliptic curves) or biometrics. For example, for the Diffie-Hellman key exchange, the value m corresponds to the size of the elliptic curve group, which directly affects efficiency. This is one of the reasons why statistical extractors are often replaced in practice with heuristic constructions based on cryptographic hash functions.

Large Seed. Another significant hurdle in the use of LHL comes from the fact that universal hash functions require long description, which means that LHL-based extractors have long seeds. Indeed, Stinson [37] showed that (perfectly) universal hash functions require the length of the seed to be linear in the length of the source X. More generally, even "good-enough" *almost* universal hash functions for LHL require seeds of length at least $\min(|X| - v, v + 2\log(1/\varepsilon)) - O(1)$ [37], and, thus, must grow with the number of extracted bits. This large seed length makes it inconvenient in many applications of extractors (e.g., [6,35,25]), including any use of extractors for derandomization, where one must be able to enumerate over all the seeds efficiently.

Large (and variable-length) seeds are also inconvenient for standardized cryptographic applications where fixed-size keys, independent of the size of inputs, are favored (as in the case of block ciphers or cryptographic hash functions). When extractors are used in cryptographic settings, seeds are viewed as keys and hence fixed-size seeds are very desirable. In applications of extractors, where the attacker is assumed to be sufficiently limited as to not make the source X dependent on the seed (e.g., when extracting keys from biometrics, physical measurements or in the Diffie-Hellman key exchange), one might consider fixing a good *public* seed, and use it repeatedly with a fast provably secure extractor. As said, this is not possible with universal hash functions as their seed length must grow with the length of X.[1]

Our Results. Quite surprisingly, we show that both limitations of the LHL — large entropy loss and large seed — can be overcome or, at least, mitigated in various important scenarios. We describe these results below.

1.1 Reducing the Entropy Loss

At first, reducing the entropy loss L might seem impossible since we already mentioned that any extractor must have entropy loss $L \geq 2\log(1/\varepsilon) - O(1)$ [34]. However, the impossibility is for general applications of extractors, where we must ensure that the extracted string R cannot be distinguished from random by *any* statistical test D. In contrast, when extractors are used to derive *cryptographic keys*, we only care about *limited types* of statistical tests D. Concretely, the tests that correspond to the security game between the attacker A and the challenger C. For example, when deriving the key for a signature scheme, the only tests we care about correspond to the attacker seeing several signatures and then outputting a new signature. Namely, we only care that the probability of a successful forgery does not suddenly become non-negligible when the secret key is obtained using an extractor instead of being random. And since the signature scheme is assumed to be secure with a truly random key, we can restrict our attention to a very restricted class of statistical tests which almost never output 1.

[1] In theory one can build (non-LHL-based) extractors where the length n of the seed H is roughly logarithmic in the length of the source X (see [16,36] and many references therein). However, the resulting constructions are mainly of theoretical value and lose the extreme simplicity and efficiency of LHL-based extractors.

Similar restrictions on the distinguisher naturally arise for other cryptographic primitives, which gives us hope that the lower bound of [34] might be overcome in such settings.

Generalized LHL and Applications. Indeed, we derive a tighter form of the LHL, called *generalized LHL* (see Theorem 1), which non-trivially depends on the type of distinguisher D we care about. Our improved bound contains a novel term informally measuring the *standard deviation* of the distinguisher's advantage (the standard LHL is a particular case where this term is bounded by 1). Applying this new bound to the analysis of cryptographic functions, we obtain much tighter bounds for the security of a wide class cryptographic applications. These include key derivation for *all* "unpredictability" applications, such as signatures, MACs, one-way functions, identification schemes, etc. More surprisingly, they also include key derivation for some prominent "indistinguishability" applications that include all stateless encryption schemes, both CPA- and CCA-secure and in the public- and symmetric-key settings, as well as weak pseudorandom functions. Specifically, in each of these cases, denote by ε the security of the cryptographic primitive (i.e., the best success probability or advantage of an attacker with certain resources) when keyed with a perfectly random v-bit key, and by ε' the corresponding security value when the key is derived from an imperfect m-bit entropy source via the LHL. We show (recall that $L = m - v$ represents the entropy loss):

$$\varepsilon' \leq \varepsilon + \sqrt{\varepsilon 2^{v-m}} = \varepsilon + \sqrt{\varepsilon 2^{-L}} \qquad (1)$$

Comparing with Standard LHL. Let us first compare this bound with the regular $\varepsilon + \sqrt{2^{-L}}$ LHL bound. The latter required $L \geq 2\log(1/\varepsilon)$ to achieve the same type of security $O(\varepsilon)$ as with the ideal randomness. Using our improved bound, we show that only half of that amount, $L = \log(1/\varepsilon)$, already suffices. In fact, not only do we get improved bounds on the entropy loss L, but we also get meaningful security bounds for arbitrary values of L, even negative ones (when the entropy loss becomes *"entropy gain"*)! E.g., standard LHL does not give anything for $L \leq 0$, while we achieve significant $\varepsilon' \approx \sqrt{\varepsilon}$ security for $L = 0$ (no entropy loss!), and even start to "gracefully borrow" security from our application when we extract more bits than the min-entropy of the source, up to $L = -\log(1/\varepsilon)$ (i.e., $v = m + \log(1/\varepsilon)$).

Computational Extractor with Improved Loss. Although our improved bound, as stated, is not applicable to all cryptographic applications (the most important omission being pseudorandom functions and stream ciphers), in Section 3.2 we use our results to build *general-purpose* key derivation function for *any* (computationally-secure) cryptographic application, while providing the full entropy benefits derived from Equation (1). The scheme combines any LHL-based extractor with any (weak) pseudorandom function family.

1.2 Reducing the Seed Length

Expand-then-Extract. A natural idea to reduce the seed length is to use a *pseudorandom generator* (PRG) to expand a short "input seed" S into a longer "output seed" S', and then use the resulting S' as the seed required by the LHL, or, more generally, by any randomness extractor. Let us call this natural approach *expand-then-extract*. Of course, as long as one hides the short S and uses the long S' as the public seed for the extractor, the extracted bits are pseudorandom. But is it possible to ensure the pseudorandomness of the extractor's output if the actual short seed S is made *public*? Had this been the case, we would obtain efficient LHL-based extractors with a short seed and, moreover, an extractor whose seed length is independent of the length of the input, as desired for the practical scenarios discussed earlier.

Counter-Example. In Section 4.1 we show that the expand-then-extract approach will *not* work in general. We construct a simple PRG (which is secure under the *Decisional Diffie-Hellman* (DDH) assumption) and an extractor (which is a natural, perfectly universal hash function), where the output of the extractor — on any (e.g., even uniform) distribution — can be efficiently distinguished from random with probability close to 1, when given the short seed S used to generate the pseudorandom long seed S' for the extractor. Despite the above, we also show two positive results which nicely complement our counter-example.

Extracting Few Bits. First, in Section 4.2 we show that the expand-then-extract approach always works provided *the number of extracted bits v is "small"*. Here "small" means logarithmic in the security level of the PRG, which could range from $O(\log k)$ to $\Omega(k)$ (where k is the security parameter), depending on whether PRG is assumed to be polynomially or exponentially hard. Quite interestingly, in this case we do not even have to settle for pseudorandom bits: our small number v of extracted bits is actually *statistically* random, as long as the PRG is secure against circuits whose size is exponential in v. The intuition for this result comes from the fact that we can test, in time exponential in v, whether a given n-bit extractor seed s' is "good" or "bad" for our source X. We also know that most *random* long seeds $s' \leftarrow U_n$ must be good. Hence, by the PRG security, the same must be true for "most" *pseudorandom* seeds $s' \leftarrow \mathsf{Prg}(U_k)$, which is precisely what we need to show.

Security in minicrypt. Second, although our original counterexample is fairly simple and natural, it involves an assumption (DDH) from the "public-key world". In Section 4.3, we show, somewhat surprisingly, that such "public-key" type assumption is indeed *necessary* for any counter-example. We do this by showing that the expand-then-extract approach is sound in minicrypt [22] (i.e. in a hypothetical world where pseudorandom generators exist, but public-key cryptography does not). In particular, we construct a simple 2-message protocol (built from a PRG and an extractor) which constitutes a secure key-agreement protocol for any PRG/extractor combination for which the expand-then-extract approach is *insecure*. Since our protocol only has 2 messages, we even get semantically secure public-key encryption (PKE). Hence, since no such protocol/PKE exist in $\mathsf{minicrypt}$, expand-then-extract must be secure.

Practical Interpretation. This leads to the following *practical interpretation* of our results indicating that using the expand-then-extract approach with common pseudorandom primitives, such as AES, is secure in spite of a lack of direct (reductionist) proof of security. Indeed, consider the expand-then-extract scheme implemented via AES (in some stream cipher mode). Our results show that, if this extraction scheme fails, then we have found *a public-key encryption scheme that is provable secure based on the security of AES as a block cipher!* Moreover, the resulting PKE has a very restrictive form, where the secret key is a PRG seed S, and the public-key is the PRG output $S' = \mathsf{Prg}(S)$. (E.g., in the case of AES, the public key is simply the evaluation of AES on several distinct points.) As we argue in Section 4.3, the existence of such a PKE appears to be extremely unlikely, and would be a major breakthrough given current state-of-the-art. Thus, our results give strong evidence that the expand-then-extract approach might be secure "in practice", — when used with "fast" ciphers (like AES), — despite being (generally) insecure "in theory"!

We also remark that all our results elegantly handle side information Z the attacker might have about our source X (as advocated by [11], such "average-case" extractors are very handy in cryptographic applications), and also generalize to the case of *almost* universal hash functions.

1.3 Related Work

Hast [17] also observed that for certain cryptographic applications, the relevant attackers correspond to restricted classes of distinguishers, which allowed him to obtain improved security bounds when the Goldreich-Levin hardcore bit [15] is used as a "computational" randomness extractor. This result is incomparable to ours. On the one hand, we consider general (multi-bit) LHL-based extractors and not just the single bit inner-product function (which is the form of the Goldreich-Levin predicate). On the other hand, Hast was working in the computational setting, and had to make an explicit reduction from the distinguisher to the predictor of the source X, which is not required in our setting.

We also mentioned the notion of *slice extractors* defined by Radhakrishnan and Ta-Shma [34], which limits the type of statistical tests to "rare distinguishers". To the best of our understanding, this definition was not motivated by applications, but rather was a convenient "parametrization" on a road to other results. Still, this notion roughly correspond to the setting of key derivation for authentication applications, when the attacker rarely succeeds. Interestingly, [34] showed a lower bound for the entropy loss of slice extractors (which was lower than that of general extractors), and matched this lower bound by an existential construction. As it turns out, our improved LHL immediately gives a *constructive* way to match this lower bound, showing that LHL-based extractors are optimal slice extractors in terms of the entropy loss. This connection is outlined in more detail in the full version of this paper [1].

In a very different (non-cryptographic) context of building hash tables, Mitzenmacher and Vahdan [27] also observed that improved bounds on the "entropy loss" could be obtained when the standard deviation of the "distinguisher" is much less than 1. In their setting the entropy loss was the minimum entropy required from the input stream to hash well, and the distinguisher was the characteristic function of a set of occupied buckets.

We note that our "win-win" result for the expand-then-extract approach is similar in spirit to several other "win-win" results [13,32,12,33], where a (hypothetical) adversary for one task is turned into a "surprising useful" protocol for a seemingly unrelated task. Among the above, the result most similar to ours is [13], where a PRG is used to expand the key for "forward secure storage", which is a concept related to "locally computable" extractors.

On the more practical side of our results, particularly in what refers to key derivation, it is worth mentioning the work of [9,25] that analyze constructions of key derivation functions (KDFs) based on cryptographic hash functions. These constructions do not use standard, generic assumptions, such as pseudorandomness, but build on specific modes of operations on their compression function f, and rely on dedicated, sometimes idealized, assumptions. Under such idealized assumptions, these schemes support situations where the KDF needs to be modeled as a random oracle, or where the source only has "unpredictability entropy" [21]. On the other hand, outside of the random oracle heuristics, much of the analysis of [9,25] studied sufficient conditions on compression function f and/or the source input distribution, under which cryptographic hash functions are "universal enough" so as to apply the standard LHL. As such, these analyses suffer the same drawbacks as any other LHL-based extractor. In particular, our results regarding the improved entropy loss for LHL-based extractors should carry over to improve the results of [9,25], while our results on the expand-then-extract approach could be viewed as partial justification of the heuristic where a fixed-description-length compression function is replaced by random in most (but not all) of the analyses of [9,25].

2 Standard Leftover Hash Lemma

Notation. For a set S, we let U_S denote the uniform distribution over S. For an integer $v \in \mathbb{N}$, we let U_v denote the uniform distribution over $\{0,1\}^v$, the bit-strings of length v. For a distribution or random variable X we write $x \leftarrow X$ to denote the operation of sampling a random x according to X. For a set S, we write $s \leftarrow S$ as shorthand for $s \leftarrow U_S$.

Min-Entropy and Extractors. The min-entropy of a random variable X is defined as $\mathbf{H}_\infty(X) \stackrel{\text{def}}{=} -\log(\max_x \Pr[X = x])$. In cryptographic applications, one often uses the average min-entropy of a random variable X conditioned on another random variable Z. This is defined as

$$\widetilde{\mathbf{H}}_\infty(X|Z) \stackrel{\text{def}}{=} -\log \mathbb{E}_{z \leftarrow Z}\left[\max_x \Pr[X = x|Z = z]\right] = -\log \mathbb{E}_{z \leftarrow Z}\left[2^{-\mathbf{H}_\infty(X|Z=z)}\right]$$

where $\mathbb{E}_{z\leftarrow Z}$ denotes the expected value over $z \leftarrow Z$, and measures the worst-case predictability of X by an adversary that may observe a correlated variable Z.

We denote with $\Delta_D(X,Y)$ the advantage of a circuit D in distinguishing the random variables X, Y: $\Delta_D(X,Y) \stackrel{\mathrm{def}}{=} |\ \Pr[D(X)=1] - \Pr[D(Y)=1]\ |$. The *statistical distance* between two random variables X, Y is defined by

$$\mathsf{SD}(X,Y) \stackrel{\mathrm{def}}{=} \frac{1}{2}\sum_x |\Pr[X=x] - \Pr[Y=x]| = \max_D \quad \Delta_D(X,Y)$$

where the maximum is taken over all (potentially computationally unbounded) D. Given side information Z, we write $\Delta_D(X,Y|Z)$ and $\mathsf{SD}(X,Y|Z)$ as shorthands for $\Delta_D((X,Z),(Y,Z))$ and $\mathsf{SD}((X,Z),(Y,Z))$, respectively.[2]

An extractor [31] can be used to extract uniform randomness out of a weakly-random value which is only assumed to have sufficient min-entropy. Our definition follows that of [11], which is defined in terms of conditional min-entropy.

Definition 1 (Extractors). *An efficient function* $\mathsf{Ext} : \mathcal{X} \times \{0,1\}^n \to \{0,1\}^v$ *is an (average-case, strong)* (m,ε)-extractor *(for space* $\mathcal{X}$*), if for all* X, Z *such that* X *is distributed over* $\mathcal{X}$ *and* $\widetilde{\mathbf{H}}_\infty(X|Z) \geq m$*, we get*

$$\mathsf{SD}(\ \mathsf{Ext}(X;S)\ ,\ U_v \mid (S,Z)) \leq \varepsilon$$

where $S \equiv U_n$ *denotes the coins of* Ext *(called the* seed*). The value* $L = m - v$ *is called the* entropy loss *of* Ext*, and the value* n *is called the* seed length *of* Ext*.*

Universal hashing and Leftover Hash lemma. We now recall the definition of universal-hashing [7,37] and the leftover-hash lemma [18], which states that universal hash functions are also good extractors.

Definition 2 (ρ-Universal Hashing). *A family* $\mathcal{H}$ *of (deterministic) functions* $h : \mathcal{X} \to \{0,1\}^v$ *is a called* ρ-universal hash family *(on space* $\mathcal{X}$*), if for any* $x_1 \neq x_2 \in \mathcal{X}$ *we have* $\Pr_{h\leftarrow\mathcal{H}}[h(x_1) = h(x_2)] \leq \rho$*. When* $\rho = 1/2^v$*, we say that* $\mathcal{H}$ *is* universal*.*

We can finally state the Leftover Hash Lemma (LHL). (Multiple versions of this lemma have appeared; we use the formulation of [38, Theorem 8.1], augmented by [11, Lemma 2.4] for the conditional entropy case; see [18] and references therein for earlier formulations.)

Lemma 1 (Leftover-Hash Lemma). *Assume that the family* $\mathcal{H}$ *of functions* $h : \mathcal{X} \to \{0,1\}^v$ *is a* $\frac{1+\gamma}{2^v}$*-universal hash family. Then the extractor* $\mathsf{Ext}(x;h) \stackrel{def}{=} h(x)$*, where* h *is uniform over* $\mathcal{H}$*, is an* (m,ε)*-extractor, where* $\varepsilon = \frac{1}{2}\cdot\sqrt{\gamma + \frac{2^v}{2^m}} = \frac{1}{2}\cdot\sqrt{\gamma + \frac{1}{2^L}}$ *(recall,* $L = m - v$ *is the entropy loss). In particular,* $\frac{1+3\varepsilon^2}{2^v}$*-universal hash functions yield* $(v + 2\log(1/\varepsilon), \varepsilon)$*-extractors.*

[2] Notice, $\Delta_D(X,Y|Z) \leq \mathbb{E}_{z\leftarrow Z}[\Delta_D(X|_{Z=z}, Y|_{Z=z})]$, but $\mathsf{SD}(X,Y|Z) = \max_D \mathbb{E}_{z\leftarrow Z}[\Delta_D(X|_{Z=z}, Y|_{Z=z})]$.

3 Reducing the Entropy Loss

As we mentioned, the entropy loss of $2\log(1/\varepsilon)$ is optimal when one is concerned with general distinguishers D [34]. As we show, in various cryptographic scenarios we only care about a somewhat restrictive class of distinguishers, which will allow us to reduce the entropy loss for such applications.

Below, we state a generalization of the LHL (Theorem 1), which will include a novel term measuring the *standard deviation* of the distinguisher's advantage, and then derive some useful special cases (Theorem 2). In Section 3.1, we then apply our tighter bound to derive improved entropy loss bounds for various cryptographic applications, including bounds for all authentication applications (Theorem 3) and some privacy applications,including chosen plaintext secure encryption (Theorem 4) and weak PRFs. In Section 3.2 we further extend our results to get a generic key derivation function with improved entropy loss for *any* computationally secure application, including stream ciphers and PRFs.

Collision Probability and c-Variance. Given a distribution Y, its collision probability is $\mathsf{Col}(Y) \overset{\mathrm{def}}{=} \sum_y \Pr[Y=y]^2 \le 2^{-\mathbf{H}_\infty(Y)}$. Given a joint distribution (Y,Z), we let $\mathsf{Col}(Y|Z) \overset{\mathrm{def}}{=} \mathbb{E}_z[\mathsf{Col}(Y|Z=z)] \le 2^{-\widetilde{\mathbf{H}}_\infty(Y|Z)}$. We also use the notation $(U_{\mathcal{Y}}, Z)$ to denote the probability distribution which first samples $(y,z) \leftarrow (Y,Z)$, but then replaces y by an independent, uniform sample from $U_{\mathcal{Y}}$. Finally, given a random variable W and a constant c, we define its *c-variance* as $\mathsf{Var}_c[W] \overset{\mathrm{def}}{=} \mathbb{E}[(W-c)^2]$ and its *c-standard deviation* as $\sigma_c[W] \overset{\mathrm{def}}{=} \sqrt{\mathsf{Var}_c[W]}$. When $c = \mathbb{E}[W]$, we recover the standard notion of variance and standard deviation, $\mathsf{Var}[W]$ and $\sigma[W]$, and notice that these are the smallest possible values for $\mathsf{Var}_c[W]$ and $\sigma_c[W]$. Still, we will later find it easier to use (slightly weaker) bounds obtained with specific values of $c \in \{0, \frac{1}{2}\}$.

We start with an useful lemma of independent interest which generalizes Lemma 4.5.1 of [27].

Lemma 2. *Assume (Y,Z) is a pair of correlated random variables distribution on a set $\mathcal{Y}\times\mathcal{Z}$. Then for any (deterministic) real-valued function $f : \mathcal{Y}\times\mathcal{Z} \to \mathbb{R}$ and any constant c, we have*

$$|\ \mathbb{E}[f(Y,Z)] - \mathbb{E}[f(U_{\mathcal{Y}},Z)]\ | \ \ \le\ \ \sigma_c[f(U_{\mathcal{Y}},Z)] \cdot \sqrt{|\mathcal{Y}|\mathsf{Col}(Y|Z) - 1} \qquad (2)$$

The proof of this lemma can be found in the full version of this paper [1]. The useful feature of the bound given in Equation (2) comes from the fact that the value $\sigma_c[f(U_{\mathcal{Y}},Z)]$ does not depend on the actual distribution Y used to replace the uniform distribution, while the value $\sqrt{|\mathcal{Y}|\mathsf{Col}(Y|Z) - 1}$ does not depend on the function f whose average we are trying to preserve (by using Y in place of $U_{\mathcal{Y}}$). In particular, we get the following corollary which will allow us to eventually get all our improved bounds.

Theorem 1 (Generalized LHL). *Let (X,Z) be some joint distribution over $\mathcal{X}\times\mathcal{Z}$, $\mathcal{H} = \{h : \mathcal{X} \to \{0,1\}^v\}$ be a family of $\frac{1+\gamma}{2^v}$-universal hash functions, H be a random member of $\mathcal{H}$, and let $L \overset{def}{=} \widetilde{\mathbf{H}}_\infty(X|Z) - v$ be the entropy loss.*

Then, for any constant $c \in [0,1]$ and any (possibly probabilistic) distinguisher $D(r,h,z)$, we have

$$\Delta_D(H(X), U_v \mid (H,Z)) \le \sigma_c\left[\Pr_D[D(U_v,H,Z)=1]\right] \cdot \sqrt{\gamma + \frac{1}{2^L}} \tag{3}$$

The proof of the theorem can be found in the full version of this paper [1]. Equation (3) bounds the advantage of D in distinguishing real and extracted randomness using two terms. The second term (under the square root) depends on the universality of $\mathcal{H}$ and the entropy loss L (but not on D). The novel term is the c-standard deviation $\sigma_c[\Pr_D[D(U_v,H,Z)=1]]$ of D, which we will simply call c-standard deviation of D and denote $V(D,c)$. Intuitively, it measures how "concentrated" the expected output of D is to some value c in the "ideal setting" (when fed U_v rather than $H(X)$). We notice that for any D and any $c \in [0,1]$, the c-standard deviation $V(D,c) \le 1$. Plugging this trivial bound in Equation (3) removes the dependence on D, and (essentially)[3] gives us the statement of the standard LHL from Lemma 1. As mentioned, though, this forces the entropy loss to be at least $2\log(1/\varepsilon)$ to achieve security ε. Below, we show several special cases when we can upper bound $V(D,c)$ by roughly $\sqrt{\varepsilon}$, which means that the entropy loss L only needs to be roughly $\log(1/\varepsilon)$ (say, with perfectly universal $\mathcal{H}$) to achieve security ε.

Theorem 2. *Let (X,Z) be some joint distribution over $\mathcal{X} \times \mathcal{Z}$, $\mathcal{H} = \{h : \mathcal{X} \to \{0,1\}^v\}$ be a family of $\frac{1+\gamma}{2^v}$-universal hash functions, H be a random member of $\mathcal{H}$, $L \stackrel{def}{=} \widetilde{\mathbf{H}}_\infty(X|Z) - v$ be the entropy loss, and $D(r,h,z)$ be some (possibly probabilistic) distinguisher. Then, for each of the values ε defined in scenarios (a)-(c) below it holds:*

$$\Delta_D(H(X), U_v \mid (H,Z)) \le \sqrt{\varepsilon \cdot \left(\gamma + \frac{1}{2^L}\right)} \tag{4}$$

(a) Assume for some $c, \delta, \tau \in [0,1]$, $\varepsilon = \tau^2 + \delta$ and the following condition is satisfied:[4]

$$\Pr_{r \leftarrow U_v, h \leftarrow \mathcal{H}, z \leftarrow Z}[\ |\Pr_D[D(r,h,z)=1] - c| \ge \tau\ ] \le \delta \tag{5}$$

(b) Assume $\Pr[D(U_v,H,Z)=1] \le \varepsilon$ (where probability is taken over U_v, H, Z and the coins of D).

(c) For fixed r, h, and z, define the distinguisher $D'(r,h,z)$ as follows. First, make two independent samples $\tilde{d}, d \leftarrow D(r,h,z)$. Then, if $\tilde{d} = 1$, return d else return $(1-d)$. Assume further that $\Pr[D'(U_v,H,Z)=1] \le \frac{1}{2} + 2\epsilon$.

The proof of this Theorem is given in the full version [1]. In the following section, we demonstrate the use of Theorem 2 by concentrating on the important case of *key derivation* using LHL, where the value ε will essentially correspond to the "cryptographic security" of the application at hand.

[3] The exact bound claimed in Lemma 1 follows when $V(D,c) \le \frac{1}{2}$, which is true for $c = 1/2$.

[4] Note that the condition below implies $|\Pr[D(U_v,H,Z)=1] - c| \le \tau + \delta$.

3.1 Improved LHL for Key Derivation

Consider any cryptographic primitive P (e.g., signature, encryption, etc.), which uses randomness $R \in \{0,1\}^v$ to derive its secret (and, public, if needed) key(s). Without loss of generality, we can assume that R itself is the secret key. In the "ideal" setting, $R = U_v$ is perfectly uniform and independent from whatever side information Z available to the attacker. In the "real setting", the key owner has a randomness source, represented by a random variable X and possibly correlated with the attacker's side information Z. It then samples a universal hash function H using (fresh) *public* randomness and uses the extracted value $R = H(X)$ as its key. We would like to argue that if P is "ε-secure" in the ideal setting (against attackers with resources[5] less than T), then P is also "ε'-secure" in the real setting (against attackers with resources less than $T' \approx T$), where ε' is not much larger than ε. Of course, to have a hope of achieving this, $\mathcal{H}$ must be "universal-enough" and $L = \widetilde{\mathbf{H}}_\infty(X|Z) - v$ must "high-enough". To parameterize this, we will sometimes explicitly write (L, γ)-*real model* to denote the real model above, where $\mathcal{H}$ is $(1+\gamma)2^{-v}$-universal and $\widetilde{\mathbf{H}}_\infty(X|Z) \geq v + L$. We formalize this general setting as follows.

Abstract Security Games. We assume that the security of P is defined via an interactive game

between a probabilistic attacker $A(h, z)$ and a probabilistic challenger $C(r)$. Here one should think of h and z as particular values of the hash function and the side information, respectively, and r as a particular value used by the challenger in the key generation algorithm of P. We note that C only uses the secret key r and does not depend on h and z. In the ideal setting, where $r \leftarrow U_v$, the attacker A does not use the values h and z (and anyway the optimal values of h and z can be hardwired into A in the non-uniform model), yet, for notation convenience, we will still pass h and z to A even in the ideal setting.

At the end of the game, $C(r)$ outputs a bit b, where $b = 1$ indicates that the attacker "won the game". Since C is fixed by the definition of P (e.g., C runs the unforgeability game for signature or the semantic security game for encryption, etc.), we denote by $D_A(r, h, z)$ the (abstract) distinguisher which simulates the entire game between $A(h, z)$ and $C(r)$ and outputs the bit b, and by $\mathsf{Win}_A(r, h, z) = \Pr[D_A(r, h, z) = 1]$ the probability that $A(h, z)$ wins the game against $C(r)$. With this notation, the probability of winning the game in the "real setting" is given by the random variable $\mathsf{Win}_A(H(X), H, Z)$, and the same probability in the ideal setting becomes $\mathsf{Win}_A(U_v, H, Z)$. Moreover, the difference between these probabilities is simply the distinguishing advantage of D_A of telling apart real and extracted randomness when given H, Z:

$$|\mathsf{Win}_A(H(X), H, Z) - \mathsf{Win}_A(U_v, H, Z)| = \varDelta_{D_A}(H(X), U_v \mid (H, Z)) \qquad (6)$$

As we justify next, to argue the security of P in the real setting assuming its security in the ideal setting, it is sufficient for us to argue that the above

[5] We use the word "resource" to include all the efficiency measures we might care about, such as running time, circuit size, number of oracle queries, etc.

distinguishing advantage is "small" for all legal attackers A. And since the security of P will usually restrict the power of attackers A (hence, also the power of abstract distinguishers D_A), we may use the results of Theorem 2 to argue better bounds on the entropy loss $L = \widetilde{\mathbf{H}}_\infty(X|Z) - v$.

Definition 3. *Let $c = 0$ for unpredictability applications P (signature, MAC, one-way function, etc.) and $c = \frac{1}{2}$ for indistinguishability applications P (encryption, pseudorandom function/permutation, etc.). Fix also the $(1+\gamma)2^{-v}$-universal hash family $\mathcal{H}$ and the joint distribution (X, Z) satisfying $\widetilde{\mathbf{H}}_\infty(X|Z) \geq v + L$, so that the real and the ideal model are well-defined.*

We say that P is (T, ε)-secure in the ideal model if for all attackers A with resources less than T, we have $\mathsf{Win}_A(U_v, H, Z) \leq c + \varepsilon$.

Similarly, P is (T', ε')-secure in the real model if for all attackers A have resources less than T', we have $\mathsf{Win}_A(H(X), H, Z) \leq c + \varepsilon'$.

Triangle inequality coupled with Equation (6) immediately yields the following Corollary.

Lemma 3. *Fix L and γ defining the real and the ideal models. Assume P is (T, ε)-secure in the ideal model, and for all attackers A with resources less than T' (where $T' \leq T$) we have $\Delta_{D_A}(H(X), U_v \mid (H, Z)) \leq \delta$. Then P is $(T', \varepsilon + \delta)$-secure in the (L, γ)-real model.*

We are now ready to apply Lemma 3 and Theorem 2 to various cryptographic primitives P. Below, we let $c \in \{0, \frac{1}{2}\}$ be the constant governing the security of P (0 for unpredictability and 1/2 for indistinguishability applications). Due to space constraint, we leave the application of part (a) of Theorem 2 to so called "strongly secure" primitives, where (for some τ and δ) any attacker has advantage more than τ on at most δ the fraction of keys r, to the full version [1] of the paper, and move directly to other applications which use parts (b) and (c) of Theorem 2. We also give several concrete examples in the full version [1].

Improved Bound for Unpredictability Applications Recall, authentication applications correspond to $c = 0$, and include signature schemes, MACs, one-way functions/permutations, etc. In this case (T, ε)-security in the ideal model implies that for any T-bounded attacker A, $\mathbb{E}[\mathsf{Win}_A(U_v, H, Z)] \leq \varepsilon$. Recalling the definition of the abstract distinguisher D_A, this is the same as $\Pr[D_A(U_v, H, Z) = 1] \leq \varepsilon$, which is precisely the pre-condition for part (b) of Theorem 2. Thus, combining Equation (4) with Lemma 3, we immediate get:

Theorem 3. *Fix L and γ defining the real and the ideal models. Assume authentication primitive P (corresponding to $c = 0$) is (T, ε)-secure in the ideal model. Then P is (T, ε')-secure in the (L, γ)-real model, where*

$$\varepsilon' \leq \varepsilon + \sqrt{\varepsilon \left(\gamma + \frac{1}{2^L} \right)} \tag{7}$$

In particular, if $\gamma = 0$ and $L = \log(1/\varepsilon)$, then $\varepsilon' \le 2\varepsilon$. Moreover, when $\gamma = 0$, the security bound is meaningful even for negative *entropy "loss" $0 \ge L > -\log(1/\varepsilon)$, when one extracts more bits than the min-entropy $\widetilde{\mathbf{H}}_\infty(X|Z)$ and "borrows the security deficit" from the ideal security of P.*

Intuitively, Theorem 3 uses the fact that for authentication applications one only cares about distinguishers which almost never output 1, since the attacker almost never forges successfully.

Improved Bound for Some Indistinguishability Applications We now move to the more difficult case of indistinguishability applications, where $c = 1/2$. In general, we do not expect to outperform the standard LHL, as illustrated by the one-time pad example. Quite surprisingly, we show that the for a class of applications, including chosen plaintext attack (CPA) secure encryption, one can still get improved bounds as compared to the standard LHL. Specifically, as long as the primitive P allows the attacker A to "test" its success before playing the actual "challenge" from C, we still get significant savings. We start by defining the general type of security games where our technique applies.

Bit-Guessing Games. As usual the game is played by the attacker $A = A(h, z)$ and the challenger $C(r)$. The game can have an arbitrary structure, except the winning condition is determined as follows. At some point A asks C for a "challenge". C flips a random bit $b \in \{0, 1\}$, and sends A a value $e = e(b, r)$. The game continues in an arbitrary way and ends with A making a guess b'. A wins if $b = b'$.

So far, this still includes all standard indistinguishability games, including the one-time pad. The extra assumption we make is the following. For any valid attacker A having resources less than T' there exists another valid attacker A' (having somewhat larger resources $T \ge T'$) such that:

(1) The execution between A' and $C(r)$ defines four bits $b, b', \tilde{b}, \tilde{b}'$, such that the joint distribution of $(b, b', \tilde{b}, \tilde{b}')$ is the same as two *independent* tuples (b, b') obtained when A runs with $C(r)$.
(2) The bits b and b' are precisely the secret bit of C and the guess of A', so that A' wins iff $b = b'$.
(3) A' learns if $\tilde{b} = \tilde{b}'$ before outputting b'.

We will call such indistinguishability games (T', T)-*simulatable*. In the full version [1], we show a general result stating improved bounds on entropy loss for any (T', T)-simulatable application, and also show that CPA-secure (public- or symmetric-key) encryption schemes are simulatable, where, as expected, the "resources" T are roughly doubled compared to T'. (Intuitively, the attacker can run the challenger in "his head" and see if it won.) In particular, we get the following theorem as a corollary of this general result:

Theorem 4. *Fix L and γ defining the real and the ideal models, and set $\varepsilon' = \varepsilon + \sqrt{\varepsilon(\gamma + 2^{-L})}$.*

Assume P is public-key encryption scheme which is ε-secure, in the ideal model, against attackers with running time $2t + t_{enc}$, where t_{enc} is the runtime of the encryption process. Then P is ε'-secure, in the (L, γ)-real model, against attackers with running time t.

Similarly, assume P is a symmetric-key encryption scheme which is ε-secure, in the ideal model, against attackers with running time $2t + O(1)$ and making $2q + 1$ encryption queries. Then P is ε'-secure, in the (L, γ)-real model, against attackers with running time t and making q encryption queries.

Limitations and Extensions. Unfortunately, several other indistinguishability primitives, such as pseudorandom generators, functions or permutations, do not appear to be simulatable. The problem seems to be in verifying the winning condition (condition (3) of simulatability), since this has to be done with respect to the actual secret key r not known to the attacker. For PRFs (or PRPs), it is tempting to solve the problem by using an equivalent definition, where the attacker can learn the value of PRF at any point, but then, as a challenge, must distinguish the value of the PRF at an *un-queried* point from random. Although this variant allows the attacker to check the winning condition during the first "virtual" run, it now creates a different problem in that the challenge point during the second "actual" run might have been queried during the first run, making such an attacker A' invalid.

Interestingly, the above "fix" works for a useful relaxation of PRFs, known as *weak PRFs* (wPRFs). Here the attacker only gets to see the values of the PRF at several *random points*, and has to distinguish a new value from random. Assuming a large enough input domain, the probability of collision between the PRF values revealed in the first first run and challenged in the second run, is negligible, which allows to complete the (valid) simulation. Similarly, it works for a slightly stronger relaxation of PRFs, known as *random-challenge* PRFs. As with wPRFs, A gets as the challenge a real-or-random evaluations of the PRF at a random point, but can additionally query the PRF at arbitrary points different from the challenge point. In Section 3.2 we show that wPRFs are all we need to apply our results to a generic key derivation function.

3.2 A Generic Key Derivation Function

So far we have discussed the applications of our generalized Leftover Hash Lemma and Theorem 2 to the derivation of cryptographic keys in entropy-constraint environments for specific applications. Although our analysis covers many such applications, it does not cover all, including PRFs and stream ciphers. In this section we make a simple observation which allows us to overcome this limitation and design a *generic* key derivation function (KDFs) which is (computationally) secure for any cryptographic application while still enjoying the same entropy loss savings. The idea is to compose universal hash functions a *weak* PRF (wPRF), where the random input to the wPRF now becomes part of the extractor seed, and use the fact that wPRFs fall under the class of *simulatable* applications as defined in Section 3.1.

Specifically, we define the KDF on the basis of a $(1+\gamma)/2^v$-universal hash family $\mathcal{H}$ with v-bit outputs and a wPRF F taking a k-bit input w and a v-bit key r, and outputting a v-bit output $y = F_r(w)$, as follows. The public seed s for the KDF Ext is a pair $s = (h, w)$, where h is a random universal function from $\mathcal{H}$ and w is a random element in the domain of F. We then define $\mathsf{Ext}(x, (h, w)) \stackrel{\text{def}}{=} F_{h(x)}(w)$; i.e., the initially extracted value $h(x)$ is used as a wPRF key, which is then used to evaluate F on w.

We also notice that if one needs to extract multiple keys for several applications, we can simply use the output of our computational KDF as a seed of a regular PRG or PRF, since such applications are now "covered".

Remark 1. For the case of deriving multiple keys, as above, we notice that the wPRF step is not needed provided *all* the keys are for cryptographic applications covered by our technique (i.e., strongly secure, unpredictable, or simulatable primitives). Namely, in such a case we can directly use the initially extracted key $H(X)$ as a seed for the (regular) PRF/PRG to derive all the required keys. This allows us to avoid increasing the seed length by k bits, and saves one application of wPRF. The proof of this claim follows by a simple hybrid argument (which we omit). In general, though, the wPRF-based solution is preferable, as it adds considerable generality at a relatively moderate cost.

4 Reducing the Seed Length

In this section we study the soundness of the natural expand-then-extract approach described in the Introduction, showing our negative result in Section 4.1 and our two positive results in Section 4.2 and Section 4.3.

Negligible and Friends. We use k to denote a security parameter. A function $\mu : \mathbb{N} \to [0, 1]$ is *negligible* if for any $c > 0$ there is a k_0 such that $\mu(k) \leq 1/k^c$ for all $k \geq k_0$. To the contrary, μ is *non-negligible* if for some $c > 0$ we have $\mu(k) \geq 1/k^c$ for infinitely many k. Throughout, $\mathsf{negl}(k)$ denotes a negligible function in k.

A function $\tau(\cdot) : \mathbb{N} \to [0, 1]$ is *overwhelming* if $1-\tau(\cdot)$ is negligible. A function $\phi : \mathbb{N} \to [0, 1]$ is *noticeable* if for some $c > 0$ there is an k_0 such that $\phi(k) \geq 1/k^c$ for all $k \geq k_0$. Note that non-negligible is not the same as noticeable. For example, $\mu(k) \stackrel{\mathsf{def}}{=} k \bmod 2$ is non-negligible but not noticeable.

Computational Extractors and PRGs. Recall that with $\Delta_D(X, Y) \stackrel{\text{def}}{=} \mid \Pr[D(X) = 1] - \Pr[D(Y) = 1] \mid$ we denote the advantage of a circuit D in distinguishing the random variables X and Y. Let $\mathcal{D}_t$ denote the class of all probabilistic circuits of size t. With $\mathsf{CD}_t(X, Y) = \max_D \Delta_D(X, Y)$ we denote the *computational distance* of X and Y, here the maximum is over $D \in \mathcal{D}_t$. When $t = \infty$ in unbounded, we recover the notion of statistical distance $\mathsf{SD}(X, Y)$. When $X = X_k$ and $Y = Y_k$ are families of distributions indexed by the security parameter k, we will say that X and Y are computationally indistinguishable, denoted $X \approx_c Y$, if for every polynomial $t(.)$, $\mathsf{CD}_{t(k)}(X_k, Y_k) = \mathsf{negl}(k)$.

Definition 4 (Computational Extractor). *We say that an efficient function* $\mathsf{Ext} : \mathcal{X} \times \{0,1\}^n \to \{0,1\}^v$ *is an (average-case, strong)* **computational** *m-extractor (for space $\mathcal{X}$), if for all* efficiently samplable X, Z *such that X is distributed over $\mathcal{X}$ and $\widetilde{\mathbf{H}}_\infty(X|Z) \geq m$ (here $\mathcal{X}, n, v, m$ are all indexed by a security parameter k)*

$$(\mathsf{Ext}(X; S), S, Z) \approx_c (U_v, S, Z)$$

Definition 5 (Pseudorandom Generator). *A length increasing function* $\mathsf{Prg} : \{0,1\}^k \to \{0,1\}^n$ *is a pseudorandom generator (PRG) if*[6] $\mathsf{Prg}(U_k) \approx_c U_n$. *We also say that* Prg *is (T, δ)-secure if* $\mathsf{CD}_T(\mathsf{Prg}(U_k), U_n) \leq \delta$.

Computational Extractors with Short Seeds? We are ready to formalize our main question: *Is it safe to expand an extractor seed using a PRG?*

Hypothesis 1 *[Expand-then-Extract] If* Ext *is an $(m(k), \varepsilon(k))$-extractor with seed length $n(k)$ where $\varepsilon(.)$ is negligible and* $\mathsf{Prg} : \{0,1\}^k \to \{0,1\}^{n(k)}$ *is a pseudorandom generator, then* Ext' *defined as*

$$\mathsf{Ext}'(x; s) = \mathsf{Ext}(x; \mathsf{Prg}(s))$$

is a computational m-extractor.

4.1 Counter-Example: Expanding Seeds is Insecure in General

In this section we show that, unfortunately, Hypothesis 1 is wrong in general.

Theorem 5 (Hypothesis 1 wrong assuming DDH). *Under the DDH assumption, there exists a pseudorandom generator* $\mathsf{Prg}(.)$ *and a strong extractor* $\mathsf{Ext}(.;.)$ *(which is a perfectly universal hash function) such that* $\mathsf{Ext}'(x; s) \overset{\mathsf{def}}{=} \mathsf{Ext}(x; \mathsf{Prg}(s))$ *can be efficiently distinguished from uniform on any input distribution (i.e.* Ext' *is not a computational extractor.)*

Proof (of Theorem 5). Let $\mathcal{G}$ be a prime order p cyclic group with generator g where the DDH problem is hard. Then $\mathsf{Prg} : \mathbb{Z}_p^3 \to \mathcal{G}^6$ defined as

$$\mathsf{Prg}(a, b, c) = (g^a, g^b, g^{ab}, g^{ac}, g^{bc}, g^{abc})$$

is a a secure pseudorandom generator [28]. Let $\mathsf{Ext} : \mathbb{Z}_p^3 \times \mathcal{G}^6 \to \mathcal{G}^2$ be

$$\mathsf{Ext}((x, y, z); (A, B, C, D, E, F)) = (A^x B^y C^z, D^x E^y F^z)$$

It is easy to see that Ext is a perfectly universal hash function from $\mathbb{Z}_p^3 \to \mathcal{G}^2$ (and, by Lemma 1, strong $(2 \log p + 2 \log(1/\varepsilon), \varepsilon)$-extractor). Now consider the distribution

$$[\mathsf{Ext}((x, y, z); \mathsf{Prg}(a, b, c)) \ , \ (a, b, c)] = [(g^{ax} g^{by} g^{abz}, g^{acx} g^{bcy} g^{abcz}) \ , \ (a, b, c)] \quad (8)$$

[6] In order to avoid an extra parameter, we simply assume wlog that the seed length of our PRG is equal to the security parameter k.

The distribution (8) is not pseudorandom as any tuple $(\alpha, \beta), (a, b, c) \in \mathcal{G}^2 \times \mathbb{Z}_p^3$ of the form (8) satisfies $\alpha^c = \beta$, which can be efficiently verified, while a random distribution will satisfy this relation with probability $1/p$.

4.2 Expanding Seeds is Safe when Extracting Few Bits

By the following theorem, the expand-then-extract Hypothesis does hold, if the pseudorandom generator Prg used for expansion is sufficiently strong. The required hardness depends exponentially on the output length v of the extractor. The proof of the following Theorem is given in the full version of this paper [1].

Theorem 6. *Assume* $\mathsf{Ext} : \mathcal{X} \times \{0,1\}^n \to \{0,1\}^v$ *is a* (m, ε)-extractor *with running time* t_{Ext}*, and* $\mathsf{Prg} : \{0,1\}^k \to \{0,1\}^n$ *is a* $(T, \sqrt{\varepsilon})$*-pseudorandom generator, for some*

$$T \in O\left(2^{2v}(n+v)t_{\mathsf{Ext}}/\varepsilon\right) \tag{9}$$

Then $\mathsf{Ext}'(x; s) \overset{\mathsf{def}}{=} \mathsf{Ext}(x; \mathsf{Prg}(s))$ *is a* $(m, 4\sqrt{\varepsilon})$-extractor*. In particular, if the running time of* Ext *is polynomial in* k*, its error* $\varepsilon(k) = \mathsf{negl}(k)$*, its output size* $v = O(\log k)$*, and* Prg *is secure against polynomial (in* k*) size distinguishers, then* Ext' *is an* (m, ε')*-extractor for* $\varepsilon'(k) = \mathsf{negl}(k)$*.*

4.3 Expanding Seeds is Safe in Minicrypt

Before we can state the main result of this section, we need a few more definitions.

Bit-Agreement. *Bit-agreement* is a protocol between two efficient parties, which we refer to as Alice and Bob. They get the security parameter k in unary (denoted 1^k) as a common input and can communicate over an authentic channel. Finally, Alice and Bob output a bit b_{A} and b_{B}, respectively. The protocol has *correlation* $\epsilon = \epsilon(k)$, if for all k, $\Pr[b_{\mathsf{A}} = b_{\mathsf{B}}] \geq (1 + \epsilon(k))/2$. Furthermore, the protocol has security $\delta = \delta(k)$, if for every efficient adversary Eve, which can observe the whole communication C, and for all k, $\Pr[\mathsf{Eve}(1^k, C) = b_{\mathsf{B}}] \leq 1 - \delta(k)/2$.

Key-Agreement & PKE. If $\epsilon(\cdot)$ and $\delta(\cdot)$ are overwhelming then such a protocol achieves *key-agreement*. Using parallel repetition and privacy amplification, it is known [20,19] that any protocol which achieves bit-agreement with noticeable correlation $\epsilon(\cdot)$ and overwhelming security $\delta(\cdot)$ can be turned into a key-agreement protocol, without increasing the number of rounds. A 2-message key-agreement protocol is equivalent to public-key encryption (PKE). The proof of the following Theorem is given in the full version of this paper [1].

Theorem 7 (Hypothesis 1 holds in minicrypt). *If there exists a secure pseudorandom generator* Prg *and a strong extractor* Ext *where* $\mathsf{Ext}'(.;.) \overset{\mathsf{def}}{=} \mathsf{Ext}(\mathsf{Prg}(.);.)$ *is* <u>*not*</u> *a computational extractor, then the protocol from Figure 1 is a two-message bit-agreement protocol with noticeable correlation and overwhelming security (and thus implies PKE).*

Remark 2. In the above theorem not being a secure computational extractor means that there exists an efficient uniform D that can distinguish $\mathsf{Ext}(\mathsf{Prg}(.); .)$ with noticeable advantage (in the security parameter k). If $\mathsf{Ext}(\mathsf{Prg}(.); .)$ is only insecure against non-uniform adversaries, then also the resulting protocol (which uses D) will be non-uniform. If the distinguisher D only has non-negligible advantage (i.e. only works for infinitely many, but not all, security parameters k), then also the protocol will work for infinitely many k. This issue is inherent in win-win type results where an adversary is turned into a "useful" protocol [13,32,12,33]. It roots in the fact that in cryptography we usually put weak requirements for adversaries to be considered efficient (can be non-uniform and only have non-negligible advantage), whereas we usually require from practical algorithms to be uniform and secure for all (sufficiently large) security parameters.

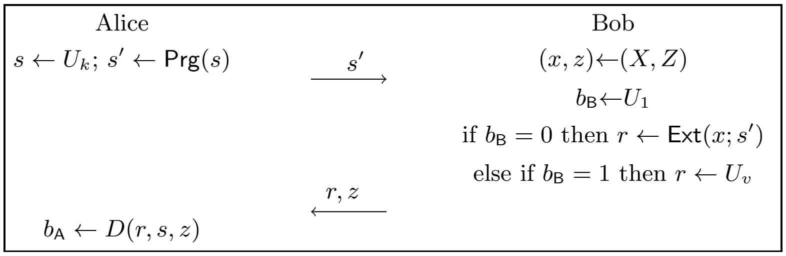
Alice
Bob
$s \leftarrow U_k;\ s' \leftarrow \mathsf{Prg}(s)$
s'
$(x, z) \leftarrow (X, Z)$
$b_\mathsf{B} \leftarrow U_1$
if $b_\mathsf{B} = 0$ then $r \leftarrow \mathsf{Ext}(x; s')$
else if $b_\mathsf{B} = 1$ then $r \leftarrow U_v$
r, z
$b_\mathsf{A} \leftarrow D(r, s, z)$

Fig. 1. A bit agreement protocol from any $\mathsf{Prg}, \mathsf{Ext}$ that constitute a counterexample (via distinguisher D) to Hypothesis 1.

We obtain the following corollary whose proof is immediate from Figure 1.

Corollary 1. *Assume* Prg *is a secure pseudorandom generator. Assume further that there exists no public-key encryption scheme (with non-negligible gap between security and decryption correctness) with pseudorandom ciphertexts, whose secret key is the seed of* Prg *and whose public key is the output of* Prg*. Then the expand-then-extract hypothesis is true for* Prg*.*

Discussion. Corollary 1 reduces the soundness of the expand-then-extract approach to the impossibility of constructing public-key encryption from the given PRG in a very particular way, where the ciphertexts are pseudorandom and, more importantly, the key-generation algorithm samples a random s and sets $sk = s, pk = \mathsf{Prg}(s)$. To the best of our knowledge, this impossibility assumption seems very likely for "practical" PRGs, such as AES. For example, not only there is no black-box construction of PKE (or key agreement) from a PRG alone, as shown by Impagliazzo and Rudich [23], but, in fact, it is entirely consistent with current knowledge that these two tasks are separable, in the sense that there is some computational model/complexity class (e.g., perhaps some extension of BQP or $AM \cap coAM$) that is powerful enough to break all public key schemes, but not powerful enough to break AES. If this is the case, then the AES-bases expand and extract scheme is secure with respect to all efficient input distributions and distingiuishers (even those that are based on public key tools such as factoring, lattices etc.). Moreover, we do not know any black-box construction of PKE from PRG and *any other* "cryptomania" assumption (like non-interactive zero-knowledge proofs, fully-homomorphic or identity-based encryption, etc.), where the public key of the PKE is simply the output of the PRG. To summarize, our results give strong evidence that the expand-then-extract approach is secure using any practical PRG (like AES), despite being (generally) insecure in theory.

Acknowledgements: We would like to thank Russell Impagliazzo and Ronen Shaltiel for useful discussions.

References

1. Boaz Barak, Yevgeniy Dodis, Hugo Krawczyk, Olivier Pereira, Krzysztof Pietrzak, Francois-Xavier Standaert, and Yu Yu. Leftover hash lemma, revisited. Cryptology ePrint Archive, Report 2011/088, 2011. http://eprint.iacr.org/.
2. Boaz Barak and Shai Halevi. A model and architecture for pseudo-random generation with applications to /dev/random. In ACM CCS 2005.
3. Boaz Barak, Ronen Shaltiel, and Eran Tromer. True random number generators secure in a changing environment. In CHES 2003.
4. Charles H. Bennett, Gilles Brassard, and Jean-Marc Robert. Privacy amplification by public discussion. *SIAM Journal on Computing*, 17(2):210–229, 1988.
5. Xavier Boyen, Yevgeniy Dodis, Jonathan Katz, Rafail Ostrovsky, and Adam Smith. Secure remote authentication using biometric data. In EUROCRYPT 2005.
6. Ran Canetti, Yevgeniy Dodis, Shai Halevi, Eyal Kushilevitz, and Amit Sahai. Exposure-resilient functions and all-or-nothing transforms. In Eurocrypt 2000.
7. J.L. Carter and M.N. Wegman. Universal classes of hash functions. *Journal of Computer and System Sciences*, 18:143–154, 1979.
8. Céline Chevalier, Pierre-Alain Fouque, David Pointcheval, and Sébastien Zimmer. Optimal randomness extraction from a diffie-hellman element. In EUROCRYPT 2009.
9. Yevgeniy Dodis, Rosario Gennaro, Johan Håstad, Hugo Krawczyk, and Tal Rabin. Randomness extraction and key derivation using the cbc, cascade and hmac modes. In CRYPTO 2004.
10. Yevgeniy Dodis, Jonathan Katz, Leonid Reyzin, and Adam Smith. Robust fuzzy extractors and authenticated key agreement from close secrets. In CRYPTO 2006.
11. Yevgeniy Dodis, Rafail Ostrovsky, Leonid Reyzin, and Adam Smith. Fuzzy extractors: How to generate strong keys from biometrics and other noisy data. *SIAM Journal on Computing*, 38(1):97–139, 2008.
12. Bella Dubrov and Yuval Ishai. On the randomness complexity of efficient sampling. In STOC 2006.
13. Stefan Dziembowski. On forward-secure storage. In CRYPTO 2006.
14. Rosario Gennaro, Hugo Krawczyk, and Tal Rabin. Secure hashed diffie-hellman over non-ddh groups. In EUROCRYPT 2004.

15. O. Goldreich and L. Levin. A hard-core predicate for all one-way functions. In STOC 1989.
16. V. Guruswami, C. Umans, and S. Vadhan. Unbalanced expanders and randomness extractors from parvaresh–vardy codes. *J. ACM*, 56(4), 2009.
17. Gustav Hast. Nearly one-sided tests and the goldreich?levin predicate. *J. Cryptology*, 17(3):209–229, 2004.
18. J. Håstad, R. Impagliazzo, L.A. Levin, and M. Luby. Construction of pseudorandom generator from any one-way function. *SIAM Journal on Computing*, 28(4):1364–1396, 1999.
19. Thomas Holenstein. Key agreement from weak bit agreement. In STOC 2005.
20. Thomas Holenstein. *Strengthening Key Agreement using Hard-Core Sets*. PhD thesis, ETH Zurich, Zurich, Switzerland, 2006.
21. Chun-Yuan Hsiao and Leonid Reyzin. Finding collisions on a public road, or do secure hash functions need secret coins. In CRYPTO 2004.
22. Russell Impagliazzo. A personal view of average-case complexity. In *Structure in Complexity Theory Conference*, pages 134–147, 1995.
23. Russell Impagliazzo and Steven Rudich. Limits on the provable consequences of one-way permutations. In STOC 1989.
24. Yuval Ishai, Eyal Kushilevitz, Rafail Ostrovsky, and Amit Sahai. Cryptography with constant computational overhead. In STOC 2008.
25. Hugo Krawczyk. Cryptographic Extraction and Key Derivation: The HKDF Scheme. In CRYPTO 2010.
26. Ueli Maurer and Stefan Wolf. Privacy amplification secure against active adversaries. In CRYPTO 1997.
27. Michael Mitzenmacher and Salil P. Vadhan. Why simple hash functions work: exploiting the entropy in a data stream. In SODA 2008.
28. Moni Naor and Omer Reingold. Number-theoretic constructions of efficient pseudorandom functions. In FOCS 1997.
29. Moni Naor and Gil Segev. Public-key cryptosystems resilient to key leakage. In CRYPTO 2009.
30. Wim Nevelsteen and Bart Preneel. Software performance of universal hash functions. In EUROCRYPT 1999.
31. Noam Nisan and David Zuckerman. Randomness is linear in space. *Journal of Computer and System Sciences*, 52(1):43–53, 1996.
32. Krzysztof Pietrzak. Composition implies adaptive security in minicrypt. In EUROCRYPT 2006.
33. Krzysztof Pietrzak and Johan Sjödin. Weak pseudorandom functions in minicrypt. In ICALP 2008.
34. Jaikumar Radhakrishnan and Amnon Ta-Shma. Bounds for dispersers, extractors, and depth-two superconcentrators. *SIAM Journal on Computing*, 13(1):2–24, 2000.
35. Renato Renner and Stefan Wolf. Unconditional authenticity and privacy from an arbitrarily weak secret. In CRYPTO 2003.
36. Ronen Shaltiel. Recent developments in explicit constructions of extractors. *Bulletin of the EATCS*, 77:67–95, 2002.
37. D. R. Stinson. Universal hashing and authentication codes. *Designs, Codes, and Cryptography*, 4(4):369–380, 1994.
38. D. R. Stinson. Universal hash families and the leftover hash lemma, and applications to cryptography and computing. *Journal of Combinatorial Mathematics and Combinatorial Computing*, 42:3–31, 2002. Available at http://www.cacr.math.uwaterloo.ca/~dstinson/publist.html.