Overcoming Weak Expectations

Yevgeniy Dodis[*] Yu Yu[†]

December 21, 2012

Abstract

Recently, there has been renewed interest in basing cryptographic primitives on weak secrets, where the only information about the secret is some non-trivial amount of (min-) entropy. From a formal point of view, such results require to upper bound the expectation of some function $f(X)$, where X is a weak source in question. We show an elementary inequality which essentially upper bounds such 'weak expectation' by two terms, the first of which is independent of f, while the second only depends on the 'variance' of f under uniform distribution. Quite remarkably, as relatively simple corollaries of this elementary inequality, we obtain some 'unexpected' results, in several cases noticeably simplifying/improving prior techniques for the same problem.

Examples include non-malleable extractors, leakage-resilient symmetric encryption, alternative to the dense model theorem, seed-dependent condensers and improved entropy loss for the leftover hash lemma.

Keywords: Leftover Hash Lemma, Randomness Extractors, Pseudorandom Generators, Key Derivation, Entropy Loss, Dense Model Theorem.

[*]Department of Computer Science, New York University. Email: dodis@cs.nyu.edu.

[†]Institute for Interdisciplinary Information Sciences, Tsinghua University. Email: yuyu@yuyu.hk.

1 Introduction

Formal cryptographic models take for granted the availability of perfect randomness. However, in reality we may only obtain 'weak' random sources that are far from uniform but only guaranteed with high unpredictability (formalized with min-entropy), such as biometric data [11, 4], physical sources [3, 2], secrets with partial leakage, and group elements from Diffie-Hellman key exchange [17, 19]. We refer to the former as ideal model and the latter as real model.

From a formal point of view, the standard (T, ε)-security (in the ideal model) of a cryptographic application P essentially requires that for any adversary A with resource[1] T, the expectation of $f(U_m)$ is upper bounded by ε, where function $f(r)$ denotes A's advantage conditioned on secret key being r, and U_m denotes uniform distribution over $\{0,1\}^m$. In the real model, keys are sampled from some non-uniform distribution R and thus the resulting security is the expected value of $f(R)$, which we call 'weak expectation'. We would hope that if P is (T, ε)-secure in the ideal setting, then P is also (T', ε') in the real setting by replacing U_m with R of sufficiently high min-entropy, where T' and ε' are not much worse than T and ε respectively.

In this paper, we present an elementary inequality that upper bounds the weak expectation of $f(R)$ by two terms: the first term only depends on the *entropy deficiency* (i.e. the difference between $m = length(R)$ and the amount of entropy it has), and the second is essentially the 'variance' of f under uniform distribution U_m. Quite surprisingly, some 'unexpected' results follow as simple corollaries of this inequality, such as non-malleable extractors [14, 10, 7, 20], leakage-resilient symmetric encryptions [23], alternative to the dense model theorem [28, 26, 15, 16], seed-dependent condensers [12] and improved entropy loss for the Leftover Hash Lemma (LHL) [1]. We provide a unified proof for these diversified problems and in many cases significantly simply and/or improve known techniques for the same problems.

Our Technique. Our main technique is heavily based on several tools introduced by Barak et al. [1] in the context of improving the "entropy loss" of the Leftover Hash Lemma [18]. This work concentrated on the setting of deriving (or extracting) a cryptographic key R from a weak source X, using public randomness S. The main observation of [1] in this context was the fact that, for a certain class of so called "square-friendly" applications P, — an informal notion later made more explicit by [12], and which we explain later — one can reduce the minimal entropy requirement on X by "borrowing" the security from P. The main insight of this work comes from "lifting" this important observation one level higher. Namely, "square-friendly" applications have the property of *directly tolerating weak keys* R. Informally, if P is "square-friendly" and (T, ε)-secure with uniform key U_m, then P is (T', ε')-secure with any weak key R having entropy deficiency (see above) d, where $T' \sim T$ and $\varepsilon' \sim 2^d \cdot \varepsilon$.[2]

Direct Applications. As mentioned above, this "obvious-in-retrospect" observation leads to several interesting (and sometimes unexpected!) consequences. First, by considering simple applications, such as information-theoretic one-time MACs [21, 9], we immediately obtain results which used to be proven directly, occasionally with elaborate analyses (essentially re-doing the elaborate analyses for the uniform case). Second, for some applications, such as weak pseudorandom functions and leakage-resilient symmetric-encryption, we obtain greatly improved results as compared to state-of-the-art [23] (and, again, with much simpler proofs). Third, we carefully design new "square-friendly" applications P, generally not studied in 'ideal' (uniform-key) setting — either because of their elementary analyses, or the lack of direct applications. However, by mechanically translating such 'uninteresting' applications P to the 'real' (weak-key) setting, we obtain natural and much more 'interesting' applications P'. Moreover, by 'blindly' applying our machinery, we get surprisingly non-trivial results about the 'real' security of such applications P'. For example, starting from a (carefully crafted) variant of pairwise

[1] 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.

[2] More precisely, $\varepsilon \cdot 2^d$ for unpredictability applications and $\sqrt{\varepsilon \cdot 2^d}$ for indistinguishability applications.

independent hash functions, we get the definition of so called *non-malleable extractors* [14]! Using our machinery, we obtain an *elementary* construction of such non-malleable extractors from 4-wise independent hash functions, which beats a much more elaborate construction recently found by [10].[3] Using a simpler example of P, — essentially a "one-wise" independent hash function, — we also get a cute (and, perhaps, surprising) variant of the Leftover Hash Lemma.

Applications to Key Derivation. Finally, we apply our improved understanding of 'real' security of "square-friendly" applications to the setting of key derivation, which was the original motivation of [1]. First, consider the case when the application P is "square-friendly". In this case, since P can directly tolerate keys R with small entropy deficiency d, our key derivation function h only needs to be a good *condenser* [24, 25] instead of being an *extractor*. Namely, instead of guaranteeing that $R = h(X; S)$ is nearly uniform (i.e., has 0 entropy deficiency), we only require that R has small entropy deficiency. This observation was recently made by [12] in a (somewhat advanced) "seed-dependent" setting for key derivation, where the distribution of X could depend on the "seed" S used to derive the final key R, and where non-trivial extraction is *impossible* [12, 29]. However, we observe that the same observation is true even in a more traditional "seed-independent" setting, where randomness extraction is possible (e.g., using LHL). In particular, since universal hash functions are good condensers for a wider range of parameters than extractors, we immediately get the same LHL improvements as [1]. Although this does not directly lead to further improvements, we believe our modular analysis better explains the results of [1], and also elegantly unifies the seed-independent [1] and the seed-dependent [12] settings. Indeed, the seed-dependent condenser considered by [12] is obtained from our construction by replacing a universal hash function by a collision-resistant hash function, and (literally!) changing one line in the proof (see Lemma 4.1 vs. Lemma 4.3).

More interestingly, we also look at the question of deriving keys for *all* (possibly "non-square-friendly") applications P. As follows from our results on seed-independent condensers (see Corollary 4.1), the question is the most challenging, and essentially reduces to the case, when the length of the source X is also m.[4] In this case, we use (appropriately long) public randomness S and a length-doubling pseudorandom generator (PRG) G on m-bit seed, to derive the following *"square-friendly" key derivation function* for P: compute $X' = G(X)$ and interpret the $2m$-bit value X' as the description of a pairwise independent hash function h from $|S|$ to m bits; then interpret S as the input to h; finally, set the final key $R = h(S) = h_{X'}(S)$.[5] Interestingly, this method is not only useful for "non-square-friendly" applications, but even for "square-friendly" applications with security $\varepsilon \gg \varepsilon_{\mathsf{prg}}$ (where $\varepsilon_{\mathsf{prg}}$ is the security of G against the same resources T as P).

PRGs with Weak Seeds. However, the above result is especially interesting when applied to PRGs themselves (i.e., $P = G$)! Namely, instead of using $G(X)$ directly as a pseudorandom string, which is not secure with weak seeds,[6] we evaluate a 'hash function' $h_{G(X)}$ on a public (random) input S, after which it is suddenly "safer" to start expanding the derived key R using G.

Prior to our work, the only alternative method of tolerating weak PRG seeds came from the "dense model theorem" [28, 26, 15, 16], which roughly states that the $2m$-bit output $X' = G(X)$ of a $(T, \varepsilon_{\mathsf{prg}})$-secure PRG G is (T', ε')-computationally close to having the same entropy deficiency $d \ll m$ as X (despite being twice as long). This means, for example, that one can now apply an m-bit extractor (e.g., LHL) h' to X' to derive the final m-bit key $R = h'(G(X))$. Unfortunately, a closer look shows

[3] The same final construction, with a very elegant direct proof, was independently discovered by Li [20]. As we explain later, our modular approach might have some advantages (beyond simplicity), such as obtaining more elaborate variants of non-malleable extractors, without the need to re-do the proof.

[4] Otherwise, one can apply a universal hash function to X with an m-bit output without affecting the "effective" entropy of the source.

[5] We mention that our result is similar in spirit to the works of [13, 8], who showed that *public* pairwise independent hash functions can save on the amount of *secret* randomness in some applications.

[6] E.g., if the first bit of X is constant, and the first bit of $G(X)$ equals to the first bit of X.

that not only ε' degrades by at least the (expected) factor 2^d as compared to $\varepsilon_{\mathsf{prg}}$, but also the time T' is much less than T: the most recent variant due to [16] has $T' \ll T^{1/3}$, while previous versions [15, 26] had $T' = T \cdot \mathsf{poly}(\varepsilon)$. In contrast, by replacing any extractor h' by a "special" extractor — a pairwise independent hash function h — and also swapping the roles of the key and the input, we can maintain nearly the same resources $T' \approx T$, but at the cost of potentially[7] increasing ε' from roughly $\varepsilon_{\mathsf{prg}} \cdot 2^d + T^{-1/3}$ to $\sqrt{\varepsilon_{\mathsf{prg}} \cdot 2^d}$. We believe such a tradeoff is quite favorable for many natural settings. Additionally, our approach is likely to use fewer public random bits S: $\log(1/\varepsilon_{\mathsf{prg}})$ bits vs. the seed length for an extractor extracting a constant fraction of min-entropy.

2 Preliminaries

Notations and Definitions. We use $s \leftarrow S$ to denote sampling an element s according to distribution S. The min-entropy of a random variable X is defined as $\mathbf{H}_\infty(X) \stackrel{\text{def}}{=} -\log(\max_x \Pr[X = x])$. We use $\mathsf{Col}(X)$ to denote the collision probability of X, i.e., $\mathsf{Col}(X) \stackrel{\text{def}}{=} \sum_x \Pr[X = x]^2 \leq 2^{-\mathbf{H}_\infty(X)}$, and collision entropy $\mathbf{H}_2(X) \stackrel{\text{def}}{=} -\log \mathsf{Col}(X) \geq \mathbf{H}_\infty(X)$. We also define average (aka conditional) collision entropy and average min-entropy of a random variable X conditioned on another random variable Z by

$$\mathbf{H}_2(X|Z) \stackrel{\text{def}}{=} -\log\left(\mathbb{E}_{z \leftarrow Z}\left[\sum_x \Pr[X = x|Z = z]^2 \right] \right) \tag{1}$$

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

respectively, where $\mathbb{E}_{z \leftarrow Z}$ denotes the expected value over $z \leftarrow Z$.

Let $c \in \{2, \infty\}$. We say that a distribution X over $\{0,1\}^m$ has *entropy deficiency* d (for a given entropy $\mathbf{H}_c$) conditioned on Z if $\mathbf{H}_c(X|Z) \geq m - d$. We also refer to the value $D \stackrel{\text{def}}{=} 2^d$ as *security deficiency* of X given Z (the reason for the name will be clear from our results).

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

$$\frac{1}{2} \sum_x |\Pr[X = x] - \Pr[Y = x]| = \max_C \ \Delta_C(X, Y)$$

We write $\mathsf{SD}(X, Y|Z)$ (resp. $\Delta_C(X, Y|Z)$) as shorthand for $\mathsf{SD}((X, Z), (Y, Z))$ (resp. $\Delta_C((X, Z), (Y, Z))$).

Abstract security games. We first define the general type of applications where our technique applies. The security of an application P can be defined via an interactive game between a probabilistic attacker A and a probabilistic challenger $\mathsf{C}(r)$, where C is fixed by the definition of P, and where the particular secret key r used by C is derived from U_m in the 'ideal' setting, and from some distribution R in the 'real' setting. The game can have an arbitrary structure, but at the end $\mathsf{C}(r)$ should output a bit, with output 1 indicating that A 'won' the game and 0 otherwise.

Given a particular key r, we define the *advantage* $f_{\mathsf{A}}(r)$ of A on r (against particular C fixed by P) as follows. For unpredictability games, $f_{\mathsf{A}}(r)$ is the expected value of $\mathsf{C}(r)$ taken over the internal coins of A and C, so that $f_{\mathsf{A}}(r) \in [0; 1]$; and for indistinguishability games, $f_{\mathsf{A}}(r)$ is the expectation of $\mathsf{C}(r) - 1/2$, so that $f_{\mathsf{A}}(r) \in [-1/2; 1/2]$. When A is clear from the context, we simply write $f(r)$.

We will refer to $|\mathbb{E}(f_{\mathsf{A}}(U_m))|$ as the advantage of A (in the ideal model). Similarly, for $c \in \{2, \infty\}$, we will refer to $\max_R |\mathbb{E}(f_{\mathsf{A}}(R))|$, taken over all distributions R with $\mathbf{H}_c(R) \geq m - d$, as the advantage of A in the $(m - d)$-real$_c$ model.

[7] Namely, our security is slightly worse only when $T^{-2/3} \ll \varepsilon_{\mathsf{prg}} \cdot 2^d$, and is never worse by more than $T^{1/6}$ factor irrespective of $\varepsilon_{\mathsf{prg}}$ and d, since $\varepsilon_{\mathsf{prg}} \cdot 2^d + T^{-1/3} \geq \sqrt{\varepsilon_{\mathsf{prg}} \cdot 2^d} \cdot T^{-1/6}$.

Definition 2.1 (Security) *An application P is* (T, ε)-secure *(in the ideal model) if the advantage of any T-bounded A is at most ε.*

An application P is (T', ε')-secure in the $(m-d)$-real$_c$ model *if the advantage of any T'-bounded A in the $(m-d)$-real$_c$ model is at most ε'.*

We note that a security result in the real$_2$ model is more desirable than (and implies) that in the real$_\infty$ model.

3 Overcoming Weak Expectations

Unpredictability applications. For unpredictability applications (with non-negative f), the following inequality implies that the security degrades at most by a factor of $D = 2^d$ compared with the ideal model (which is stated as Corollary 3.1), where d is the entropy deficiency.

Lemma 3.1 *For any (deterministic) real-valued function $f : \{0,1\}^m \to \mathbb{R}^+ \cup \{0\}$ and any random variable R with $\mathbf{H}_\infty(R) \geq m - d$, we have*

$$\mathbb{E}[f(R)] \;\leq\; 2^d \cdot \mathbb{E}[f(U_m)] \tag{3}$$

Proof.

$$\mathbb{E}[f(R)] \;=\; \sum_r \Pr[R = r] \cdot f(r) \;\leq\; 2^d \cdot \sum_r \frac{1}{2^m} \cdot f(r)$$

□

Corollary 3.1 *If an unpredictability application P is (T, ε)-secure in the ideal model, then P is $(T, 2^d \cdot \varepsilon)$-secure in the $(m-d)$-real$_\infty$ model.*

The above only applies to all "unpredictability" applications such as one-way functions, MACs and digital signatures. For a concrete example, consider the following one-time (information-theoretic) message authentication code (MAC). The m-bit key $r = (a, b)$, where $(a, b) \in \{0,1\}^{m/2}$, and the tag of an $m/2$-bit messages x is $\sigma_r(x) = a \cdot x + b$, where addition/multiplication is done in the field $GF[2^{m/2}]$. This MAC is easily known to have ideal security $\varepsilon = 2^{-m/2}$ [27], and was directly shown to have $(m-d)$-real$_\infty$ security $\varepsilon' = 2^{d-m/2}$ by [21]. A similar, but more involved direct analysis was given by [9] for the generalization of this MAC to longer messages (using well-known polynomial hashing [27]). In both cases, the analysis involved a relatively simple, yet slightly tricky probability derivation. This derivation, in hindsight inherited from the known ideal security proofs, carefully showed that learning the tag of any x does not help the attacker to select a new message $x' \neq x$ whose tag is correlated with the tag of x. Corollary 3.1 shows that both derivations did not have to be re-done when going from the 'ideal' to the 'real' model.

Indistinguishability applications. Unfortunately, Corollary 3.1 critically depends on the non-negativity of f, and is generally false when f can be negative, which happens for indistinguishability applications. In fact, for certain indistinguishability applications, such as one-time pad, pseudo-random-generators and functions (PRGs and PRFs), there exists R with $d = 1$ such that $\mathbb{E}[f(U_m)]$ is negligible (or even zero!) but $\mathbb{E}[f(R)] = 1/2$. For example, consider one-time pad encryption $e = x \oplus r$ of the message x using a key r, which has perfect security $\varepsilon = 0$ in the ideal model, when $r \leftarrow U_m$. However, imagine an imperfect key R, whose first bit is 0 and the remaining bits are uniform. Clearly, $\mathbf{H}_\infty(R) = m - 1$, but one can perfectly distinguish the encryptions of any two messages differing in the first bit, implying $(m-1)$-real$_\infty$ security $\varepsilon' = 1/2$. Fortunately, below we give another inequality for general f, which will be useful for other indistinguishability applications.

Lemma 3.2 *For any (deterministic) real-valued function $f : \{0,1\}^m \to \mathbb{R}$ and any random variable R with $\mathbf{H}_2(R) \geq m - d$, we have*

$$| \ \mathbb{E}[f(R)] \ | \ \leq \ \sqrt{2^d} \cdot \sqrt{\mathbb{E}[f(U_m)^2]} \tag{4}$$

Proof. Denote $p(r) = \Pr[R = r]$, and also recall the Cauchy-Schwartz inequality $|\sum a_i b_i| \leq \sqrt{(\sum a_i^2) \cdot (\sum b_i^2)}$. We have

$$| \ \mathbb{E}[f(R)] \ | = \left| \ \sum_r p(r) \cdot f(r) \ \right| \leq \sqrt{2^m \cdot \sum_r p(r)^2} \cdot \sqrt{\frac{1}{2^m} \sum_r f(r)^2} = \sqrt{2^d \cdot \mathbb{E}[f(U_m)^2]}$$

□

Lemma 3.2 upper bounds the (squared) weak expectation by the product of "security deficiency" $D = 2^d$ of R and $\mathbb{E}[f(U_m)^2]$. As with unpredictability applications, the value D comes to play due to the entropy deficiency of R, independent of f. Also, the second term $\mathbb{E}[f(U_m)^2]$ only depends on the uniform distribution (and not on R). However, it no longer bounds the ideal model security $\mathbb{E}[f(U_m)]$ of our application in consideration, but rather the expected square of the attacker's advantage. This leads us to the notion of *square security*, which was implicitly introduced by [1] and later made explicit by [12] in a more restricted context of key derivation.

Definition 3.1 (Square Security) *An application P is (T, σ)-square secure if for any T-bounded adversary A we have $\mathbb{E}[f(U_m)^2] \leq \sigma$, where $f(r)$ denotes A's advantage conditioned on key being r.*

Applying this definition to Lemma 3.2, we get the following general result.

Corollary 3.2 (Square security implies real model security) *If P is (T, σ)-square secure, then P is $(T, \sqrt{2^d \cdot \sigma})$-secure in the $(m - d)$-real$_2$ model.*

What applications have square security? More precisely, for which applications P does a good bound on standard security ε also imply a good bound on their square security? Let us call such applications *square-friendly*. We start with a few simple observations. First, all (T, ε)-secure unpredictability applications P are (T, ε)-square secure, since for non-negative f we have $\mathbb{E}[f(U_m)^2] \leq \mathbb{E}[f(U_m)]$. Hence, we immediately get $\sqrt{2^d \cdot \varepsilon}$-security in $(m - d)$-real$_2$ model for such applications. Notice, this bound is weaker than the $2^d \varepsilon$ bound in Corollary 3.1, although it applies whenever $\mathbf{H}_2(R) \geq m - d$ (instead of only when $\mathbf{H}_\infty(R) \geq m - d$, which is more restrictive). Still, we will find the seemingly weaker bound from Lemma 3.2 useful even for unpredictability applications, when we talk about key derivation functions in Section 4. This will precisely use the fact that Renyi entropy is a weaker restriction than min-entropy, making it easier to construct an appropriate key derivation function.

Moving to indistinguishability applications, it is known that PRGs, PRFs and one-time pads cannot have good square security (see [1]). Indeed, given our earlier counter-example for the one-time pad, a different result would contradict the bound in Corollary 3.2. To see this explicitly, consider a 1-bit one time pad encryption $e = x \oplus r$, where $x, r, e \in \{0, 1\}$ are the message, the key and the ciphertext, respectively. Consider also the attacker A who guesses that $x = e$. When the key $r = 0$, A is right and $f(0) = 1 - \frac{1}{2} = \frac{1}{2}$. Similarly, when the key $r = 1$, A is wrong and $f(1) = 0 - \frac{1}{2} = -\frac{1}{2}$. This gives perfect $\varepsilon = \mathbb{E}[f(U_1)] = 0$, but $\sigma = \mathbb{E}[f(U_1)^2] = \frac{1}{4}$.

Fortunately, there are still many interesting indistinguishability applications which are square-friendly, such as stateless chosen plaintext attack (CPA) secure encryption and weak pseudo-random functions (weak PRFs), as shown by [1]. These examples are shown using an elegant "double run" technique from [1]. In the following we give a slightly cleaner exposition of this technique, by decomposing (until Section 4) the core of this technique from the specifics of the key derivation setting. We also mention the "multi-run" extension of this technique, and then derive several new, somewhat unexpected examples, using several variants of q-wise independent hash functions.

3.1 Square-Friendly Applications via the Double-Run Trick

To make the exposition more intuitive, we start with a nice example of CPA-secure symmetric-key encryption schemes from [1], and later abstract and generalize the resulting technique.

Illustrating Example. Recall, for this application P the attacker "resources" $T = (t, q)$, where t is the running time of A and q is the total number of encryption queries made by A. More specifically, A is allowed to (adaptively) ask the challenger $\mathsf{C}(r)$ to produce (randomly generated) encryptions of $(q-1)$ arbitrary messages $s_1, \ldots, s_{q-1}$ under the secret key r, and (at any moment) one special "challenge" query (s_0^*, s_1^*). In response to this latter query, $\mathsf{C}(r)$ picks a random bit $b \in \{0,1\}$ and returns the encryption of s_b^*. Eventually, A outputs a bit b' and 'wins' if $b' = b$. As with other indistinguishability applications, the advantage $f(r)$ of A on key r is $\Pr[b = b'] - 1/2$.

Lemma 3.3 ([1]) *Assume P is a symmetric-key encryption scheme which is $((2t, 2q), 2\varepsilon)$-CPA-secure (in the ideal model). Then P is $((t, q), \varepsilon)$-square secure. Hence, standard CPA-security implies essentially the same level "CPA-square-security" (formally, with all parameters halved).*

Proof. It suffices to show that for any r and any attacker A with running time t and q queries, there exists another attacker B with running time roughly $2t$ and $2q$ queries, such that B's advantage on r is twice the squared advantage of A on r.

The strategy of B is to initialize two independent copies of A (with fresh randomness) — call them A_1 and A_2 — and run them one after another as follows. First, it first simulates a run of A_1 against the 'imaginary' challenger $\mathsf{C}_1(r)$, using q regular encryption queries to its own 'real' challenger $\mathsf{C}(r)$, to simulate both $(q-1)$ regular *and* 1 challenge queries of A_1 to $\mathsf{C}_1(r)$. In particular, the knowledge of the first challenge bit b_1 (which B chose himself) allows B to know whether or not A_1 succeeded in this simulated run. After the first simulated run is over, B now runs a second fresh copy A_2 of A "for real", now using A_2's challenge query (s_0^*, s_1^*) to C_2 as its own challenge query with C. This uses a total of $2q$ queries for B to complete both runs. Finally, if A_1 wins the game in its first run (against simulated C_1), then B returns A_2's answer in the second run unmodified; otherwise, B reverses the answer of A_2, interpreting the mistake of A_1 in the first run as an indication of a likely mistake of A_2 in the second run as well. In particular, irrespective of the sign of A's advantage ε below, we have

$$\Pr[\mathsf{B} \text{ wins}] \;=\; \Pr[\mathsf{A} \text{ wins twice}] \;+\; \Pr[\mathsf{A} \text{ loses twice}] = \left(\frac{1}{2} \pm \varepsilon\right)^2 + \left(\frac{1}{2} \mp \varepsilon\right)^2 = \frac{1}{2} + 2\varepsilon^2$$

□

The following theorem immediately follows from Corollary 3.2 and Lemma 3.3.

Theorem 3.1 *Assume P is a $((2t, 2q), 2\varepsilon)$-CPA secure symmetric-key encryption scheme in the ideal model. Then P is also $((t, q), \sqrt{2^d \cdot \varepsilon})$-secure in the $(m-d)$-real$_2$ model.*

Double-run trick. We now generalize this technique to any indistinguishability application P which we call (T', T, γ)-*simulatable*, slightly generalizing (and, in our opinion, simplifying) the related notion introduced by [1]. For syntactic convenience (and without loss of generality), we assume that in the security game for P the challenger $\mathsf{C}(r)$ chooses a random bit b, and the attacker A wins by outputting a bit $b' = b$ *without violating some failure predicate F*, where F is efficiently checkable by both A and C. For example, for the CPA encryption example from above, this failure predicate F is empty. In contrast, for the related notion of chosen ciphertext (CCA) security, F will be true if A asked C to decrypt the actual challenge ciphertext. Notice, since any A can efficiently check F, we could have assumed that no such A will violate F (we call such A *legal*). However, we will find our small convention slightly more convenient in the future, including the following definition.

Definition 3.2 *We say that an indistinguishability application P is* (T', T, γ)-simulatable, *if for any secret key r and any legal, T-bounded attacker A, there exists a (possibly illegal!) T'-bounded attacker B (for some $T' \geq T$) such that:*

(1) The execution between B and 'real' $\mathsf{C}(r)$ defines two independent *executions between a copy A_i of A and a 'simulated' challenger $\mathsf{C}_i(r)$, for $i = 1, 2$. In particular, except reusing the same r, $\mathsf{A}_1, \mathsf{C}_1(r), \mathsf{A}_2, \mathsf{C}_2(r)$ use fresh and independent randomness, including independent challenge bits b_1 and b_2.*

(2) The challenge b used by 'real' $\mathsf{C}(r)$ is equal to the challenge b_2 used by 'simulated' C_2.

(3) Before making its guess b' of the challenge bit b, B learns the values b_1, b'_1 and b'_2.

(4) The probability of B violating the failure predicate F is at most γ.

For example, the proof of Theorem 3.1 showed that any CPA-secure encryption is $(T' = (2t, 2q), T = (t, q), \gamma = 0)$-simulatable, since B indeed simulated two runs of A satisfying conditions (1)-(4) above. In particular, a straightforward abstraction of our proof shows the following:

Lemma 3.4 *Assume P is a (T', ε)-secure and (T', T, γ)-simulatable, then P is (T, σ)-square secure, where $\sigma \leq (\varepsilon + \gamma)/2$. In particular, by Corollary 3.2 P is $(T, \sqrt{2^{d-1}(\varepsilon + \gamma)})$-secure in the $(m-d)$-real$_2$ model.*

Multi-Run Extension. In the double-run game we use a test-run to estimate the sign of the advantage (whether it is positive or not), which advises attacker B whether or not to reverse A's answer in the real run. We can generalize this to a multi-run setting: the attacker B test-runs A for some odd $(2i+1)$ times, and takes a majority vote before the actual run, which gives B more accurate estimate on the sign of the advantage of A. Interestingly, with a different motivation in mind, this precise question was studied by Brakerski and Goldreich [5]. Translated to our vocabulary, to gain a factor $\alpha > 1$ in the square security (i.e., to show that $\sigma \leq \varepsilon/\alpha$), one needs to run the original distinguisher A for $\Theta(\alpha^2)$ times. Going back to Corollary 3.2, to get 'real' security $\varepsilon' = \frac{1}{\alpha} \cdot \sqrt{2^d \cdot \varepsilon}$, one needs to run A for $O(\alpha^4)$ times, therefore losing this factor in the allowed resources T. Although theoretically interesting, it appears that the best practical tradeoff is already achieved using the "double-run" trick itself, a conclusion shared by [5].

3.2 Applications to Weak Pseudorandom Functions and Extractors

Recall, weak PRFs [22] are close relatives of CPA-secure symmetric encryption, and relax the notion of (regular) PRFs. For future applications, we give a precise definition below.

Definition 3.3 ($((t, q), \delta)$-weak PRFs) *A family $\mathcal{H}$ of functions $\{h_r : \{0,1\}^n \to \{0,1\}^l \mid r \in \{0,1\}^m\}$ is $((t, q), \delta)$-secure weak PRF, if for any t-bounded attacker A, and random $s, s_1, \ldots, s_{q-1} \leftarrow U_n$ and $r \leftarrow U_m$, we have*

$$\Delta_{\mathsf{A}}(\ h_r(s),\ U_l \mid s\ ,\ s_1\ ,\ h_r(s_1),\ \cdots,\ s_{q-1},\ h_r(s_{q-1})\ ) \leq \delta$$

Notice, it is impossible to achieve $\delta = 0$ in this definition for $q > 1$, as there is always a small chance that $s \in \{s_1, \ldots, s_{q-1}\}$. Also, just like CPA-secure encryption, weak PRFs are easily seen to be $((2t, 2q), (t, q), 0)$-simulatable, since the 'outer' attacker B can choose its own bit b_1, and set the challenge value of the first run to be $h_r(s_q)$ if $b_1 = 0$, and uniform U_l otherwise. By Lemma 3.4, this means that

Theorem 3.2 *Assume P is a $((2t, 2q), \delta)$-secure weak PRF in the ideal model. Then P is $((t, q), \delta/2)$-square secure, as well as $((t, q), \sqrt{2^{d-1} \cdot \delta})$-secure in the $(m-d)$-real$_2$ model.*

Moreover, by applying the multi-run extension, if P is a $((O(\alpha^4 \cdot t), O(\alpha^4 \cdot q)), \delta)$-secure, then P is also $((t,q), \frac{1}{\alpha} \cdot \sqrt{2^d \cdot \delta})$-secure in the $(m-d)$-real$_2$ model. This results nicely improves (and simplifies!) a result of Pietrzak [23], who achieved security $\delta' \sim \delta \cdot 2^d$, but at a price of reducing the allowed running time t' and the number of queries q' by a *huge* factor $\mathsf{poly}(1/\delta') = \mathsf{poly}(2^d, 1/\delta)$. A comparable (actually, slightly better) result follows from our multi-run derivation above, by taking a very large value of $\alpha \sim 1/\sqrt{2^d \delta}$. Of course, such large α makes the resulting values t and q really low compared to the original t' and q'. Indeed, we believe the region of 'small' α, and especially the result of Theorem 3.2, is much more relevant for practical use.

While the security of weak PRFs with weak keys was already studied by [23, 1] with large q in mind, we obtain some expected results by concentrating on the most basic case of $q = 1$.

Application to Extractors. By looking at the k-real$_2$ security (where $k = m - d$) of weak PRFs for $q = 1$ and $t = \infty$, we essentially obtain the notion of extractors for Renyi entropy!

Definition 3.4 (Extractors) *We say that an efficient function* $\mathsf{Ext} : \{0,1\}^m \times \{0,1\}^n \to \{0,1\}^l$ *is a strong* (k, ε)*-extractor, if for all* R *(over* $\{0,1\}^m$*) with* $\mathbf{H}_2(R) \geq k$ *and for random* S *(uniform over* $\{0,1\}^n$*), we get*

$$\mathsf{SD}(\ \mathsf{Ext}(R;S)\ ,\ U_l \mid S) \leq \varepsilon$$

where coins $S \leftarrow U_n$ *is the random seed of* Ext. *The value* $L = k - l$ *is called the* entropy loss *of* Ext.

To apply Theorem 3.2 (with $q = 1$, $t = \infty$ and $k = m - d$) and obtain such extractors, all that remains is to build an $((\infty, 2), \delta)$-secure weak PRFs for a low value of δ, which is essentially a *pairwise independent* hash function on two random inputs:

$$\mathsf{SD}(\ h_r(s),\ U_l \mid s, s', h_r(s')\ ) \leq \delta \tag{5}$$

where $r \leftarrow U_m$ and $s, s' \leftarrow U_n$. For example, using any traditional pairwise independent hash function, which has the property that

$$\Pr_{r \leftarrow U_m}[h_r(s) = a\ \wedge\ h_r(s') = a'] = 2^{-2l} \tag{6}$$

for any $s \neq s'$ and any $a, a' \in \{0,1\}^l$, we achieve that the only case when one can distinguish $h_r(s)$ and $h_r(s')$ is when $s = s'$, which happens with probability 2^{-n}. In other words, pairwise independent hashing gives $\delta = 2^{-n}$, which, in turn, gives (by Theorem 3.2, with $q = 1$, $t = \infty$, $k = m - d$ and $\delta = 2^{-n}$):

Corollary 3.3 (Alternative LHL) *If* $\mathcal{H} \stackrel{\mathsf{def}}{=} \{h_r : \{0,1\}^n \to \{0,1\}^l \mid r \in \{0,1\}^m\}$ *is pairwise independent (i.e., satisfies Equation (6)), then* $\mathsf{Ext}(r;s) \stackrel{\mathsf{def}}{=} h_r(s)$ *is a strong* $(k, \sqrt{2^{m-k-n}})$*-extractor.*

To compare this result with the standard LHL [18], the optimal key length m for a family of pairwise independent hash functions from n to l bits (where $l \leq m/2$) is known to be $m = n + l$ (e.g., using Toeplitz matrices or "augmented" inner product discussed below). Plugging this to our bound in ε above, we get the same bound $\varepsilon = \sqrt{2^{l-k}} = 2^{-L/2}$ as the leftover hash lemma, where in both cases l is output size and k is the entropy of the source. Interestingly, standard leftover hashing uses universal $\mathcal{H}$ (see Definition 4.3 below), which is weaker, but sets $\mathsf{Ext}(r;s) = h_s(r)$, swapping the roles of source and seed. Also, while the LHL allows to push the output l arbitrarily close to the Renyi entropy k, the Alternative LHL above can only push l up to $m/2$, even when $m/2 < k \leq m$, making the standard LHL more preferable.

Finally, somewhat curiously, when l divides n, the following "augmented" inner product function $h_r(s)$ is *simultaneously* an optimal pairwise independent hash function when keyed by r, and an optimal universal function when keyed by s: $h_r(s) = r_1 \cdot s_1 + \ldots + r_p \cdot s_p + r_{p+1}$, where $p = n/l$, $r = (r_1, \ldots, r_{p+1})$, $s = (s_1, \ldots, s_p)$, and r_i and s_j are interpreted as elements of $GF[2^l]$. This peculiar "coincidence" will resurface again in Section 4.3, when we discuss the alternative to the dense-model theorem.

Computational Pairwise Independence. Continuing our exploration of $q = 1$ (whose square security follows from regular security of $q = 2$), information-theoretic pairwise independence requires that the length m of the key r is at least twice the length l of the function output $h_r(s)$. Looking ahead at the key derivation setting in Section 4.3, m will be equal to the security parameter, and we will need to achieve output length $l \geq m$, which is impossible information-theoretically. Instead, we observe that the result can be easily achieved computationally, by applying a length-doubling pseudorandom generator (PRG) $G : \{0,1\}^m \to \{0,1\}^{2m}$ first. Namely, a weak computationally pairwise independent hash function with an m-bit key *and* output can be obtained by first expanding the key r to $r' = G(r)$, and then using r' as the $2m$-bit key of (no longer impossible) pairwise independent hash function $h_{r'}$ with an m-bit output. We postpone further exploration of this computationally $((t, 2), \delta)$-secure weak PRF, and its application to key derivation, till Section 4.3.

3.3 Application to Non-Malleable Extractors

Having obtained randomness extractors for $q = 1$, we now continue our exploration for $q = 2$ (and larger values). First, however, we strengthen the security experiment for weak PRFs in order to obtain much stronger results. Indeed, while the "double-run" trick seems to require that the challenge input s is randomly chosen,[8] there seems to be no reason not to allow the attacker A to *choose* the input values $s_1, \ldots, s_{q-1}$, as long as all of them are different from the actual challenge s. In fact, we will even allow A to select $s_1, \ldots, s_{q-1}$ based on the challenge input s.[9]

The resulting notion, which we (for simplicity) only state for the information-theoretic case of $t = \infty$, is given below. Here the phrase that "$s_1 \ldots s_n$ can be arbitrarily correlated to s" means that the unbounded attacker A (implicit in the definition below) chooses $s_1 \ldots s_{q-1}$ as a function of s.

Definition 3.5 (weak (q, δ)-wise independence) *A family $\mathcal{H}$ of functions $\{h_r : \{0,1\}^n \to \{0,1\}^l \mid r \in \{0,1\}^m\}$ is* weakly (q, δ)-wise independent, *if for $r \leftarrow U_m$, $s \leftarrow U_n$, and for $s_1, \cdots, s_{q-1} \in \{0,1\}^n$ that are distinct from and arbitrarily correlated to s, we have*

$$\mathsf{SD}(\ h_r(s),\ U_l \mid s, h_r(s_1), \cdots, h_r(s_{q-1})\ ) \leq \delta$$

The failure event F happens when one of the points $s_i = s$.

Notice that, unlike weak with PRFs, here ideal security $\delta = 0$ is possible, since we explicitly require that $s \notin \{s_1 \ldots s_{q-1}\}$. In fact, a (perfectly) q-wise independent hash function (where the analog of Equation (6) holds for larger q) is also weakly $(q, 0)$-wise independent.

Also observe that we can naturally view the above definition as a game between a challenger C and the attacker A, where $(q-1)$ measures the "resources" of A (distinct from s points where he learns the true value of h_r), and δ is the advantage of distinguishing $h_r(s)$ from random. In particular, we can naturally define the (q, σ_q)-square security of $\mathcal{H}$ (with random key $r \leftarrow U_m$) and then use Corollary 3.2 to bound the security of $\mathcal{H}$ in the $(m-d)$-real$_2$ model, when using a weak key R with $\mathbf{H}_2(R) \geq m - d$. In fact, we can successfully apply the double-run trick above to show that $\mathcal{H}$ is $(2q, q, \gamma)$-simulatable, where, for the first time we have a non-zero failure probability $\gamma = q/2^n$. Indeed, to simulate the first virtual run of A, B simply chooses its own random point s and asks its value $h_r(s)$. The subtlety comes from the fact that both s and the the $(q-1)$-correlated values $s_1 \ldots s_{q-1}$ might accidentally collide with

[8] Otherwise, we arrive at the notion of of PRFs, which we know are not square-friendly.

[9] As mentioned later, we could even allow A to see the challenge $h_r(s)/U_l$ *and* s before selecting $s_1, \ldots, s_{q-1}$, although we do not formally pursue this direction.

the second (fortunately) random challenge s', making the resulting 'outer' attacker B illegal. Luckily, the probability that a random s' collides with any of these q values is at most $\gamma \leq q/2^n$, indeed.

Theorem 3.3 *If function family $\mathcal{H}$ is weakly $(2q, \delta)$-wise independent, then $\mathcal{H}$ is $(q, (\delta + q/2^n)/2)$-square secure, as well as weakly (q, ε)-wise independent in the $(m-d)$-real$_2$ model, where $\varepsilon = \sqrt{(\delta + \frac{q}{2^n}) \cdot 2^{d-1}}$.*

Non-malleable Extractors. Next, we consider the case of $q = 2$, where the notion of $(2, \varepsilon)$-wise independence in the $k = (m-d)$-real$_2$ model becomes a *non-malleable extractor* [14] (for Renyi entropy; the case $q = 1$ collapses to the setting of weak PRF considered in the previous section).

Definition 3.6 (Non-Malleable Extractors) *We say that an efficient function $\mathsf{nmExt} : \{0,1\}^m \times \{0,1\}^n \to \{0,1\}^l$ is a (k, ε)-*non-malleable extractor*, if for all R (over $\{0,1\}^m$) with $\mathbf{H}_2(R) \geq k$, for random S (uniform over $\{0,1\}^n$), and for all functions $g : \{0,1\}^n \to \{0,1\}^n$, s.t. $g(s) \neq s$ for all s, we get*

$$\mathsf{SD}(\ \mathsf{nmExt}(R; S)\ ,\ U_l \mid\ \ S, \mathsf{nmExt}(R; g(S))\ ) \leq \varepsilon$$

Applying Theorem 3.3 to (perfectly) 4-wise independent hash functions (i.e., $2q = 4$, $\delta = 0$, $k = m - d$), we get:

Corollary 3.4 (Non-Malleable Extractors) *If $\mathcal{H} \stackrel{\mathsf{def}}{=} \{h_r : \{0,1\}^n \to \{0,1\}^l \mid r \in \{0,1\}^m\}$ is 4-wise independent, then $\mathsf{nmExt}(r; s) \stackrel{\mathsf{def}}{=} h_r(s)$ is a $(k, \sqrt{2^{m-k-n}})$-non-malleable extractor.*

For a simple instantiation, let $\mathcal{H}$ be the following (optimal) 4-wise independent hash function with known parameters $n = m/2$ and $l = m/4$ (using BCH codes; see [20]). The key $r \in \{0,1\}^m$ is viewed as a tuple of 4 elements (r_1, r_2, r_3, r_4) in $GF[2^{m/4}] = GF[2^l]$, and a seed $s \in \{0,1\}^n \backslash 0^n$ is viewed as a non-zero point in $GF[2^n]$. Then, the m-bit value of $(s \| s^3)$ is viewed as 4 elements (s_1, s_2, s_3, s_4) in $GF[2^l]$, and the l-bit output of the function is set to $h_r(s) = r_1 \cdot s_1 + \ldots + r_4 \cdot s_4$. Using Corollary 3.4, this simple function is a $(k, \sqrt{2^{m/2-k}})$-non-malleable extractor with an output of size $l = m/4$. Quite surprisingly, this noticeably improves a much more complicated initial construction of non-malleable extractors of [10]. That result could only extract $l = k/2 - m/4 - \Omega(\log m) - \log(1/\varepsilon) \ll m/4$ bits, and relied on an unproved conjecture in number theory. In particular, even for one-bit output, it achieved slightly worse security $\varepsilon' = O(\mathsf{poly}(n)) \cdot \sqrt{2^{m/2-k}}$.

As mentioned earlier, the same final construction was independently discovered by Li [20] with a different, more direct proof. We believe that our modular approach might have some advantages (beyond simplicity). For example, in addition to generalizing to larger values of q (an observation also made by Li [20]), it appears that our approach seamlessly tolerates more elaborate variants of non-malleable extractors, such as when the points $s_1, \ldots, s_{q-1}$ could also depend on the challenge $h_r(U_m)/U_l$. It is not immediately clear if the same easily holds for the proof of [20].

3.4 Side Information

So far we presented our results assuming $\mathbf{H}_c(R) \geq m - d$ from the perspective of our attacker A. In some settings, such as the key derivation setting in Section 4 below, R itself is derived using some procedure, at the end of which the attacker A gets some side information S about R, but we know that $\mathbf{H}_c(R \mid S) \geq m - d$ (see Equation (1) and Equation (2) for definitions of conditional entropy). In other words, we only know that *on average*, over the choice of side information S, R has entropy at least $(m - d)$. One option to deal with such average-case setting is to use Markov's inequality to argue that $\mathbf{H}_c(R \mid S = s)$ is "almost" $(m-d)$, with "high probability" over the choice $s \leftarrow S$. Fortunately, though, all our results so far naturally extend directly to handle the average-case setting.

More formally, we define average-case $(m-d)$-real$_c$ model as follows. The pair (R,S) is distributed according to some joint distribution satisfying $\mathbf{H}_c(R \mid S) \geq m-d$. When the particular pair $(r,s) \leftarrow (R,S)$ is sampled, we now run the challenger $\mathsf{C}(r)$ against the attacker $\mathsf{A}(s)$, which defines the attacker's advantage $f_{\mathsf{A}}(r,s)$ on a particular r and s. The overall advantage of A against (R,S) is simply $|\mathbb{E}[f_{\mathsf{A}}(R,S)]|$, while $\max_{(R,S)} |\mathbb{E}[f_{\mathsf{A}}(R,S)]|$, taken over all distributions (R,S) with $\mathbf{H}_c(R \mid S) \geq m-d$, is now the the advantage of A in the (average-case) $(m-d)$-real$_c$ model. The ideal advantage of A is now defined as $\max_s |\mathbb{E}[f_{\mathsf{A}}(U_m,s)]|$, while the (ideal) square advantage is $\max_s \mathbb{E}[f^2_{\mathsf{A}}(U_m,s)]$.[10] The notions of (T,ε)-real/ideal security and (ideal) (T,σ)-square security can now be defined as before.

With these natural definitions in in mind, it is easy to see that all our proofs generalize seamlessly, provided that our basic Lemma 3.1 and Lemma 3.2 generalize. Fortunately, the latter is easily seen to be true:

Lemma 3.5 *For any real-valued function $f(r,s)$ and any random variables (R,S), where $|R| = m$:*

(a) If $\mathbf{H}_\infty(R \mid S) \geq m-d$ and $f \geq 0$, then $\mathbb{E}[f(R,S)] \leq 2^d \cdot \max_s \mathbb{E}[f(U_m,s)]$.

(b) If $\mathbf{H}_2(R \mid S) \geq m-d$, then $|\mathbb{E}[f(R,S)]| \leq \sqrt{2^d \cdot \mathbb{E}[f(U_m,S)^2]} \leq \sqrt{2^d \cdot \max_s \mathbb{E}[f(U_m,s)^2]}$.

Proof. Fix $c=\infty$ for part (a) and $c=2$ for part (b). Given s, define d_s by $\mathbf{H}_c(R \mid S=s) = m-d_s$. Then, from Equation (1) and Equation (2), we know that $\mathbb{E}_{s\leftarrow S}[2^{d_s}] \leq 2^d$.

For part (a), let $M_1 = \max_s \mathbb{E}[f(U_m,s)]$. Then,

$$\mathbb{E}[f(R,S)] = \mathbb{E}_{s\leftarrow S}[\mathbb{E}[f(R,S) \mid S=s]] \leq \mathbb{E}_{s\leftarrow S}[2^{d_s} \cdot \mathbb{E}[f(U_m,s)]] \leq \mathbb{E}_{s\leftarrow S}[2^{d_s}] \cdot M_1 \leq 2^d \cdot M_1$$

where the middle inequality $\mathbb{E}[f(R,S) \mid S=s] \leq 2^{d_s} \cdot \mathbb{E}[f(U_m,s)]$ follows from Lemma 3.1.

For part (b), we first show the weaker inequality $|\mathbb{E}[f(R,S)]| \leq \sqrt{2^d \cdot \max_s \mathbb{E}[f(U_m,s)^2]}$, by using Lemma 3.2 as a black-box. By Lemma 3.2 we know that $|\mathbb{E}[f(R,S) \mid S=s]| \leq \sqrt{2^{d_s} \cdot \mathbb{E}[f(U_m,s)^2]} \leq \sqrt{2^{d_s} \cdot M_2}$, where $M_2 = \max_s \mathbb{E}[f(U_m,s)^2]$. Taking expectation over $s \leftarrow S$, and using Jensen's inequality $\mathbb{E}[\sqrt{Z}] \leq \sqrt{\mathbb{E}[Z]}$, we get

$$|\mathbb{E}[f(R,S)]| \leq \mathbb{E}_{s\leftarrow S}[\ |\ \mathbb{E}[f(R,S) \mid S=s]\ |\ ] \leq \mathbb{E}_{s\leftarrow S}[\sqrt{2^{d_s} \cdot M_2}] \leq \sqrt{\mathbb{E}[2^{d_s}] \cdot M_2} \leq \sqrt{2^d \cdot M_2}$$

Notice, the weaker inequality is already sufficient for our purposes, since we define (average-case, ideal) square security by taking maximum over all s anyway. However, for elegance sake, we now show the stronger inequality $|\mathbb{E}[f(R,S)]| \leq \sqrt{2^d \cdot \mathbb{E}[f(U_m,S)^2]}$.

Denote $q(s) = \Pr[S=s]$ and $p(r|s) = \Pr[R=r \mid S=s]$. As in Lemma 3.2, we will use Cauchy-Schwartz inequality $|\sum a_i b_i| \leq \sqrt{(\sum a_i^2)\cdot(\sum b_i^2)}$, but will split the term $q(s)$ as $\sqrt{q(s)2^m} \cdot \sqrt{q(s)2^{-m}}$. When squared, it will give two terms $q(s)$, which is will be precisely what we want:

$$\begin{aligned}
|\ \mathbb{E}[f(R,S)]\ | \ &= \ \left| \sum_{s,r} q(s)\cdot p(r|s)\cdot f(r,s) \right| = \left| \sum_{s,r} (\sqrt{q(s)2^m}\cdot p(r|s))\cdot(\sqrt{q(s)2^{-m}}\cdot f(r,s)) \right| \\
&\leq \ \sqrt{2^m \cdot \sum_{s,r} q(s)\cdot p(r|s)^2} \cdot \sqrt{\frac{1}{2^m}\sum_{s,r} q(s)\cdot f(r,s)^2} = \sqrt{2^{m-\mathbf{H}_2(R|S)} \cdot \mathbb{E}[f(U_m,S)^2]} \\
&\leq \ \sqrt{2^d \cdot \mathbb{E}[f(U_m,S)^2]}
\end{aligned}$$

□

[10] One can think of s as the advice in the ideal model, which usually does not change anything since we consider non-uniform attackers.

4 Key Derivation Functions

So far we studied the security of various applications when their m-bit secret key is weak (i.e., has some entropy deficiency d). In many situations, one is given a source of randomness X of some, possibly different, length n and having some entropy k, and we need to first map it to the m-bit key R by means of some *key derivation function* (KDF) $h : \{0,1\}^n \to \{0,1\}^m$. As we will see, the source entropy k and the output length m play the most important role in this scenario, which leads to the following definition.

Definition 4.1 *We define* (k, m)-real$_c$ model *(for* $c \in \{2, \infty\}$*) as the key derivation setting, where a given KDF* h *with range* $\{0,1\}^m$ *is applied to any source* X *with* $\mathbf{H}_c(X) \geq k$*, to get a secret key* $R = h(X)$ *(for some application in question).*

As it turns out, in this level of generality deterministic key derivation is (essentially)[11] impossible for *all sources* of entropy k (see [12] for related discussion), so we will assume (and critically capitalize on) the existence of *public randomness* S. Depending on context, we will view such S either as a *seed* for $h(X; S)$, or as the description of h itself.

Having clarified the setting, we now turn to the question of designing such KDFs h for a given application P. First, when the source entropy $k \geq m + 2\log(1/\varepsilon)$, where ε is the desired security level for P, we can apply a good strong randomness extractor (e.g., by using LHL) to derive a key R which is (statistically) ε-close to U_m (even conditioned on the seed S). In practice, however, many sources do not have this much entropy, so we will consider the more challenging (and, often, more realistic) case when k is (noticeably) less than $m + 2\log(1/\varepsilon)$. We will divide our study intro three complementary approaches.

First, in Section 4.1 we will leverage the rich body results we obtained in Section 3 for dealing with "square-friendly" applications, and show that *randomness condensers* (instead of more demanding extractors) are precisely the right primitives to obtain improved key derivation results for all square-friendly applications. This will lead to the improved variant of LHL discovered by Barak et al. [1], but in a more modular and, arguably, intuitive manner. Interestingly, the parameters of standard extractors (i.e., standard LHL) will also "pop-up" to cover all (even non-square-friendly) applications when $k \geq m + 2\log(1/\varepsilon)$.

Second, in Section 4.2 we turn to a more challenging *seed-dependent* setting, considered by Dodis et al. [12], where the distribution on the source X could depend on the public seed S. This more or less follows the presentation of [12], and is included in this work mainly for completeness and modularity of exposition.

Finally, while the results of the previous subsections were interesting mainly from the perspective of the presentation, in Section 4.3 we consider the ("seed-independent") setting where the results of Section 4.1 lead to poor parameters. Namely, when the application P is either non-square-friendly, or has poor exact security ε to withstand the multiplicative 2^d loss incurred by our prior techniques. This will be done by capitalizing on our setting of public randomness to design a *square-friendly key derivation function* h. This has the advantage that the security of this key derivation step only needs to be analyzed with *uniform* X (i.e., in the ideal model), and our prior results will immediately imply the security of h in the real model. Moreover, instead of using the security of our final application P (which, as we said, leads to poor parameters), we will view the process of key derivation *as a new application* P' *of its own*! In particular, if the resulting key $R = h(X)$ will be pseudorandom, we can use it for any 'outer' P, irrespective of P's security or "square-friendliness".

We notice that a less optimized variant of this idea was already proposed by [1], who noticed that a weak PRF h_X— with public randomness S viewed as the input to h_X — is precisely the square-friendly

[11] Except maybe in the model of uniform adversaries, not considered here.

primitive we are looking for. In this work we take this observation one step further, by capitalizing on the fact that h only needs to be secure for *two queries* in the ideal model (and, hence, one query in the real model). This leads to a simple (computationally-secure) construction of such a KDF h using length-doubling PRGs, already mentioned at the end of Section 3.2. As an unexpected consequence, also mentioned in the Introduction, our new KDF will give us an interesting (and often more favorable) alternative to the dense model theorem [28, 26, 15, 16].

4.1 Condensers and Improved Leftover Hash Lemma

Recall, in the (k, m)-*real*$_c$ *model* we have an n-bit source X having $\mathbf{H}_c(X) \geq k$, and we wish to derive an m-bit key R from X. Moreover, the results on Section 3 — in particular Corollary 3.1 (for $c = \infty$) and Corollary 3.2 (for $c = 2$) — show that all we need from our KDF h is to ensure that $\mathbf{H}_c(R) \geq m - d$ to ensure security degradation of the order 2^d. Remembering the fact that our key derivation has a public seed S, which means that R should have entropy even given S. Fortunately, by the results of Section 3.4 all our results in Section 3 hold with respect to the side information S. Thus, we naturally arrive at the following definition.

Definition 4.2 (Condensers) *Let $c \in \{2, \infty\}$. We say that an efficient function $\mathsf{Cond} : \{0,1\}^n \times \{0,1\}^v \to \{0,1\}^m$ is a $(\frac{k}{n} \to \frac{m-d}{m})_c$-condenser if for $\mathbf{H}_c(X) \geq k$ and uniformly random S we have $\mathbf{H}_c(\ \mathsf{Cond}(X; S) \mid S\ ) \geq m - d$.*

Both $\mathbf{H}_\infty$- and $\mathbf{H}_2$- condensers are useful in cryptography. The former connects well with Lemma 3.1 (formally, its extension Lemma 3.5(a)) and Corollary 3.1, and the latter is more in line with Lemma 3.2 (formally, its extension Lemma 3.5(b)) and Corollary 3.2. In the sequel, though, we will only use $\mathbf{H}_2$ (and let $c = 2$ hereafter) since it seems to give stronger final bounds (even for unpredictability applications!), and applies to more cases (e.g. square-friendly indistinguishability applications). See [12] for more discussion.

We now recall the notion of universal hashing [6] and explicitly prove a well-known folklore[12] that universal hashing gives very good randomness condensers.

Definition 4.3 (Universal Hashing) *A family of functions $\mathcal{G} \stackrel{\mathsf{def}}{=} \{g_s : \{0,1\}^n \to \{0,1\}^m \mid s \in \{0,1\}^v\}$ is universal, if for any distinct $x_1, x_2 \in \{0,1\}^n$ we have*

$$\Pr_{s \leftarrow U_v}[g_s(x_1) = g_s(x_2)] = 2^{-m}$$

Lemma 4.1 *Universal hash function family $\mathcal{G} \stackrel{\mathsf{def}}{=} \{g_s : \{0,1\}^n \to \{0,1\}^m \mid s \in \{0,1\}^v\}$ defines a $(\frac{k}{n} \to \frac{m-d}{m})_2$-condenser $\mathsf{Cond}(x; s) \stackrel{\mathsf{def}}{=} g_s(x)$, where $2^d = 1 + 2^{m-k}$.*

Proof. We directly analyze the collision probability by estimating the probability that two independent samples X_1 and X_2 of X collide under g_S. The latter is done by conditioning on whether X_1 and X_2 collide among themselves, and using the universality of $\mathcal{G}$ to tackle the case of no collision:

$$\begin{aligned} \Pr[g_S(X_1) = g_S(X_2)] &\leq \Pr[X_1 = X_2] + \Pr[\ g_S(X_1) = g_S(X_2)\ \wedge\ X_1 \neq X_2] \\ &\leq 2^{-k} + 2^{-m} \ = \ 2^{-m} \cdot (2^{m-k} + 1) \ = \ 2^{d-m} \end{aligned}$$

□

Instead of composing this result with Lemma 3.2/Lemma 3.5(b), we use a slightly different version of these lemmas (whose proof is very similar as well, and is omitted) leading to improved final results.

[12] This argument is usually hidden inside the proof of the standard LHL, but here we find it worthy on its own.

Lemma 4.2 ([1]) *For any (deterministic) real-valued function $f : \{0,1\}^m \to \mathbb{R}$ and any random variable R with $\mathbf{H}_2(R) \geq m - d$, we have*

$$| \ \mathbb{E}[f(R)] - \mathbb{E}[f(U_m)] \ | \ \leq \ \sqrt{2^d - 1} \cdot \sqrt{\mathbb{E}[f(U_m)^2]} \tag{7}$$

More generally, when side information S is present, and $\mathbf{H}_2(R \mid S) \geq m - d$, we have:

$$| \ \mathbb{E}[f(R, S)] - \mathbb{E}[f(U_m, S)] \ | \ \leq \ \sqrt{2^d - 1} \cdot \sqrt{\mathbb{E}[f(U_m, S)^2]} \tag{8}$$

Corollary 4.1 (Using Universal Hashing as KDF) *If P is (T, ε)-secure and (T, σ)-square secure (in the ideal model), then using $R = g_s(X)$ makes P (T, ε')-secure in the (k, m)-real$_2$ model, where $\varepsilon' \leq \varepsilon + \sqrt{\sigma \cdot 2^{m-k}}$.*

Reduced entropy loss for leftover hash lemma. Recalling the notion of entropy loss $L \stackrel{\mathsf{def}}{=} k - m$, used in the earlier study of extractors, the bound of Corollary 4.1 can be rewritten as $\varepsilon' \leq \varepsilon + \sqrt{\sigma \cdot 2^{-L}}$. In particular, since any application has square-security $\sigma \leq 1$, we get $\varepsilon' \leq \varepsilon + \sqrt{2^{-L}}$. This implicitly recovers the traditional application of the LHL, which argues that entropy loss $L \geq 2 \log(1/\varepsilon)$ is enough to ensure comparable security $\varepsilon' \leq 2\varepsilon$ for any application P. More interestingly, we saw in Section 3 that many "square-friendly" applications, including all unpredictability applications and many indistinguishability applications, achieve $\sigma \approx \varepsilon$, in which case we get a bound $\varepsilon' \leq \varepsilon + \sqrt{\varepsilon \cdot 2^{-L}}$. Thus, to achieve $\varepsilon' \approx \varepsilon$, we only need to set $L = \log(1/\varepsilon)$ for such applications, saving $\log(1/\varepsilon)$ in the entropy requirement on X. More surprisingly, the resulting bound is meaningful even for negative L, in which case we are extracting more bits than the entropy k we have.

Finally, one can interpret Corollary 4.1 as the indication that the most challenging setting for key derivation occurs when the source X has length $n = m$, a result we will use in Section 4.3. Indeed, when $n = m$, the value $m - k$ is simply the *entropy deficiency* d of our source X. In particular, applying any of the results in Section 3 directly to X (without doing the key derivation) would incur a factor $2^d = 2^{m-k}$ loss in security. Using Corollary 4.1, we see that by applying an m-bit universal hash to any n-bit source X, we get the *same* security degradation 2^{m-k} as if the derived m-bit key R had the *same* entropy k as the original n-bit source X!

4.2 Seed-Dependent Key Derivation

We now generalize the notion of a condenser to the seed-dependent setting, where the adversarial sampler A can depend on the seed S but is computationally bounded. This challenging setting was considered by [29] in the context of seed-dependent *extractors*, where the authors made a pessimistic conclusion that the complexity of the seed-dependent extractor must be larger than that of the sampler A, making this notion not very useful for key derivation in practical applications. In contrast, we follow the result work of [12] who showed that (strong enough) collision-resistant hash functions (CRHFs) must be seed-dependent *condensers*, and thus can be used as KDFs for all square secure applications, despite having much smaller complexity than the complexity of the sampler A. This partially explains the use of CRHFs as KDFs in practical applications.

Definition 4.4 (Seed-Dependent Condensers) *An efficient function $\mathsf{Cond} : \{0,1\}^n \times \{0,1\}^v \to \{0,1\}^m$ is a $(\frac{k}{n} \to \frac{m-d}{m}, t)_2$-seed-dependent condenser if for all probabilistic adversaries A of size t who take a random seed $s \leftarrow U_v$ and output (using more coins) a sample $X \leftarrow \mathsf{A}(s)$ of entropy $\mathbf{H}_2(X|S) \geq k$, we have $\mathbf{H}_2(\ \mathsf{Cond}(X; S) \mid S \ ) \geq m - d$.*

Definition 4.5 (CRHF) *A family of hash functions $\mathcal{G} \stackrel{\mathsf{def}}{=} \{g_s : \{0,1\}^n \to \{0,1\}^m \mid s \in \{0,1\}^v\}$ is (t, δ)-collision-resistant if for any (non-uniform) attacker B of size t, we have*

$$\Pr[g_s(x_1) = g_s(x_2) \ \wedge \ x_1 \neq x_2] \leq \delta$$

where $s \leftarrow U_v$ and $(x_1, x_2) \leftarrow \mathsf{B}(s)$.

Lemma 4.3 (CRHFs are seed-dependent condensers) *A family of $(2t, \frac{D(t)}{2^m})$-collision-resistant hash functions $\mathcal{G} \stackrel{\mathsf{def}}{=} \{g_s : \{0,1\}^n \to \{0,1\}^m \mid s \in \{0,1\}^v\}$ defines a seed-dependent $(\frac{k}{n} \to \frac{m-d}{m}, t)_2$-condenser $\mathsf{Cond}(x; s) = g_s(x)$, where $2^d = 2^{m-k} + D(t)$.*

Proof. We estimate the collision probability of $\mathsf{A}(S)$ given S, but letting $\mathsf{A}(S)$ sample $X_1, X_2 \leftarrow \mathsf{A}(S)$, and bounding the probability of collision as follows:

$$\begin{aligned} \Pr[g_S(X_1) = g_S(X_2)] &\leq \Pr[X_1 = X_2] + \Pr[\; g_S(X_1) = g_S(X_2) \;\wedge\; X_1 {\neq} X_2 \;] \\ &\leq 2^{-k} + D(t) \cdot 2^{-m} \;=\; 2^{-m} \cdot (2^{m-k} + D(t)) \;=\; 2^{d-m} \end{aligned}$$

where $\Pr[\; g_S(X_1) = g_S(X_2) \;\wedge\; X_1 {\neq} X_2 \;] \leq D(t)/2^m$, since otherwise we can define an efficient collision-finding adversary $\mathsf{B}(S)$, who simply runs the sampler $\mathsf{A}(S)$ twice to get the collision (X_1, X_2). □

In the above, the entropy deficiency d is essentially the logarithm of $D(t)$, which is a function on the sampler's complexity t. We note $D(t) = \Omega(t^2)$ due to birthday attacks, and this bound can be achieved in the random oracle model. In general, it is reasonable to assume $D(t) = \mathsf{poly}(t)$ for strong enough CRHFs. Then, using the definition of condensers and Corollary 3.2, we get the following surprising result, which partially explains the prevalent use of CRHFs (which do not appear to have any extraction properties based on their definition) for key derivation:

Corollary 4.2 (Using CRHFs as KDFs) *If P is (T, σ)-square secure, $\{g_s\}$ is a family of $(2t, \frac{\mathsf{poly}(t)}{2^m})$-CRHFs, and X is a source produced by a sampler $\mathsf{A}(s)$ of complexity at most t and having $\mathbf{H}_2(X|S) \geq k \geq m - O(\log t)$, then using $R = g_s(X)$ makes P (T, ε')-secure, where $\varepsilon' \leq O(\sqrt{\sigma \cdot \mathsf{poly}(t)})$.*

From an asymptotic point of view, for square-friendly applications (e.g. CPA-secure encryptions, weak PRFs, unpredictability primitives) with negligible ideal ε (and hence negligible $\sigma \approx \varepsilon$), and all source samplers running in polynomial time t (all in the "security parameter"), we get negligible security $\varepsilon' {=} O(\sqrt{\varepsilon \cdot \mathsf{poly}(t)})$ in the real model.

4.3 Generic, Square-Friendly Key Derivation

Finally, we return to the "seed-independent" setting and turn to the question of generic key derivation for all applications P, by viewing the process of key derivation with public randomness as an application in itself! Indeed, we can imagine a game between the challenger $\mathsf{C}(x)$ and an attacker A, where $\mathsf{C}(x)$ sends the public randomness s in the first round, and then challenges A to distinguish the value $r = h(x; s)$ from uniform. In fact, this is nothing more than the weak PRF game for $q = 1$ considered in Section 3.2,[13] except we (confusingly) "renamed" our secret r by x, and the derived key $h_r(s)$ by $r = h(x; s)$ (the input s kept its letter)! In particular, we saw that the weak PRFs are square-friendly, which means that all we need to do (not counting letter translation) in order to apply Theorem 3.2 is to design a "good enough" $((t, 2), \delta)$-secure weak PRF *in the ideal model*.

To do so, let us examine the parameters we need. First, as explained at the end of Section 4.1, we will assume that our source length $n = m$ (since we can always apply Corollary 4.1 to more or less reduce to this case). Thus, we need a $((t, 2), \delta)$-secure weak PRF where both the key (now called x) and the output (now called r) are m-bit long. As explained at the end of Section 3.2, we cannot achieve this result information-theoretically. Instead, we return to the "computational pairwise independent" construction sketched at the end of Section 3.2, starting with a formal definition of a PRG.

[13] Alternatively, one can view this game as a *computational extractor*, where the extracted string r is only required to be pseudorandom.

Definition 4.6 (PRG) *We say that a function* $G : \{0,1\}^m \to \{0,1\}^{2m}$ *is a* $(2t, \varepsilon_{\mathsf{prg}})$*-secure PRG if for any* $2t$*-bounded attacker* A, $\Delta_{\mathsf{A}}(G(U_m), U_{2m}) \le \varepsilon_{\mathsf{prg}}$.

We now compose such a PRG G with any pairwise independent hash function (see Equation (6)) $h_y : \{0,1\}^p \to \{0,1\}^m$, where key length $|y| = 2m$ (we will set the input length $p \le m$ shortly). For example, viewing $y = (a, b)$, where $a, b \in GF[2^m]$ and $s \in \{0,1\}^p \subseteq GF[2^m]$, we could set $h_{a,b}(s) = a \cdot s + b$. Recall, as discussed in Section 3.2, the resulting family $\mathcal{H}$ is clearly a $((\infty, 2), 2^{-p})$-secure weak PRF, except its key y is too long ($2m$ bits instead of m). Therefore, we define the composed function $h'_x : \{0,1\}^p \to \{0,1\}^m$, with key $x \in \{0,1\}^m$, by $h'_x(s) = h_{G(x)}(s)$. A simple hybrid argument shows that the resulting hash family $\mathcal{H}'$ is a $((2t, 2), \varepsilon_{\mathsf{prg}} + 2^{-p})$-secure weak PRF. Combining this result with Theorem 3.2, we finally get our key derivation function from m-to-m bits:

Theorem 4.1 *If* G *is* $(2t, \varepsilon_{\mathsf{prg}})$*-secure and* $\mathcal{H}$ *is pairwise independent, then the* m*-to-*m*-bit key derivation function* $h'_x(s) = h_{G(x)}(s)$ *uses* p *bits of public randomness* s *and achieves* $(t, \sqrt{2^{d-1} \cdot (\varepsilon_{\mathsf{prg}} + 2^{-p})})$*-security in the* $(m-d, m)$*-real*$_2$*-model.*

In particular, if $p = \log(1/\varepsilon_{\mathsf{prg}})$*, then* $\mathcal{H}'$ *is* $(t, \sqrt{2^d \cdot \varepsilon_{\mathsf{prg}}})$*-secure in the* $(m-d, m)$*-real*$_2$*-model, and the derived key* R *can be used in any (even non-square-friendly)* (t, ε)*-secure application* P *needing an* m*-bit key, giving* $(t, \varepsilon + \sqrt{2^d \cdot \varepsilon_{\mathsf{prg}}})$*-security for* P *in the* $(m-d, m)$*-real*$_2$*-model.*

Notice, the latter bound might be beneficial not only for non-square-friendly applications, where no other options are available, but also for square-friendly applications where $\varepsilon \gg \varepsilon_{\mathsf{prg}}$. Also, the assumption $p \ge \log(1/\varepsilon_{\mathsf{prg}})$ is easy to achieve, since the standard "$a \cdot s + b$" pairwise independent hash function achieves $p = m$, and $m \gg \log(1/\varepsilon_{\mathsf{prg}})$ for any m-to-$2m$-bit PRG. Hence, we never need to use more than m bits of public randomness.

Generic Key Derivation. We now combine Theorem 4.1 and Corollary 4.1 to tackle the case of a general n-to-m-bit key derivation function. As before, we take a $(2t, \varepsilon_{\mathsf{prg}})$-secure PRG $G : \{0,1\}^m \to \{0,1\}^{2m}$ and a pairwise independent family $\mathcal{H} = \{h_y : \{0,1\}^p \to \{0,1\}^m \mid y \in \{0,1\}^{2m}\}$ with $p \ge \log(1/\varepsilon_{\mathsf{prg}})$, but now also use a universal family $\mathcal{G} = \{g_s : \{0,1\}^n \to \{0,1\}^m \mid s \in \{0,1\}^v\}$. Define the new hash family $\mathcal{H}' = \{h'_{s,s'} : \{0,1\}^n \to \{0,1\}^m \mid s \in \{0,1\}^v, s' \in \{0,1\}^p\}$ by $h'_{s,s'}(x) = h_{G(g_s(x))}(s')$.

Before stating our final bound, we claim that we can instantiate $\mathcal{H}'$ so that the amount of public randomness $p + v$ it is at most $\max(m, n)$. Indeed, the size of s' can be made $p = \log(1/\varepsilon_{\mathsf{prg}}) \le m$ bits. When $n \le m$, $\mathcal{G}$ can be keyless and simply have the "never-colliding" identity function, so $m = m + 0 = \max(m, n)$ total bits is enough. When $n \ge m$, the optimal key size of a universal hash family from n bits to m bits is $v = n - m$ (by Toeplitz matrices construction, or, when n is a multiple of m, the "augmented" inner product construction discussed in Section 3.2). This gives total number of bits at most $m + (n - m) = n = \max(m, n)$.

Using Corollary 4.1 to analyze the 'inner' n-to-m-bit KDF with parameters of the 'outer' m-to-m-bit KDF given by Theorem 4.1, we get:

Corollary 4.3 *If* G *is* $(2t, \varepsilon_{\mathsf{prg}})$*-secure PRG,* $\mathcal{H}$ *is pairwise independent and* $\mathcal{G}$ *is universal, then the function family* $\mathcal{H}'$ *above defines an* n*-to-*m*-bit key derivation function that uses* $p + v$ *bits of public randomness and achieves* $(t, \varepsilon_{\mathsf{prg}} + \sqrt{2^{m-k} \cdot \varepsilon_{\mathsf{prg}}})$*-security in the* (k, n)*-real*$_2$*-model.*

In particular, the derived key R *can be used in any (even non-square-friendly)* (t, ε)*-secure application* P *needing an* m*-bit key, giving* $(t, \varepsilon + \varepsilon_{\mathsf{prg}} + \sqrt{2^{m-k} \cdot \varepsilon_{\mathsf{prg}}})$*-security for* P *in the* (k, n)*-real*$_2$*-model.*

Moreover, $\mathcal{H}$ *and* $\mathcal{G}$ *can be instantiated so that the amount of public randomness* $p + v \le \max(m, n)$.

As before, the generic bound for P might be beneficial not only for non-square-friendly applications P, where no other options are available, but also for square-friendly applications where $\varepsilon \gg \varepsilon_{\mathsf{prg}}$.

Alternative to Dense Model Theorem. Finally, as mentioned in the Introduction, the most unexpected consequence occurs when we apply it to $P = G$ itself! In this case, while the initial value

$Y = G(X)$ need not be pseudorandom *at all* when $\mathbf{H}_2(X) \geq m - d$, our result in Theorem 4.1 implies that $G(h_{G(X)}(S))$ is $(t, \varepsilon_{\mathsf{prg}} + \sqrt{2^d \cdot \varepsilon_{\mathsf{prg}}})$-pseudorandom, even conditioned on S.

Theorem 4.2 (Alternative Dense Model Theorem) *Assume $G : \{0,1\}^m \to \{0,1\}^{2m}$ is a $(2t, \varepsilon_{\mathsf{prg}})$-secure PRG, $\mathcal{H} = \{h_y : \{0,1\}^p \to \{0,1\}^m \mid y \in \{0,1\}^{2m}\}$ is a pairwise independent family with $p \geq \log(1/\varepsilon_{\mathsf{prg}})$, X is any seed distribution over $\{0,1\}^m$ with $\mathbf{H}_2(X) \geq m - d$, and $S \leftarrow U_p$ is a public random string (independent of X). Then, for any t-bounded distinguishers A and B, we have*

$$\Delta_{\mathsf{A}}(\ h_{G(X)}(S)\ ,\ U_m \mid S\ ) \leq \sqrt{2^d \cdot \varepsilon_{\mathsf{prg}}} \quad \text{and} \quad \Delta_{\mathsf{B}}(\ G(h_{G(X)}(S))\ ,\ U_{2m} \mid S\ ) \leq \varepsilon_{\mathsf{prg}} + \sqrt{2^d \cdot \varepsilon_{\mathsf{prg}}}$$

In other words, $G(h_{G(X)}(S))$ is $(t, \varepsilon_{\mathsf{prg}} + \sqrt{2^d \cdot \varepsilon_{\mathsf{prg}}})$-pseudorandom conditioned on S.

We briefly compare to the results obtained via the standard dense model theorem [28, 26, 15, 16], which used to be the only alternative to dealing with pseudorandom generators having weak seeds. As a high-level summary, the best version of the previous approach [16] would show that $G(\mathsf{Ext}(G(X); S))$ is (t', ε')-pseudorandom, where $t' \approx t^{1/3}$, $\varepsilon' \approx 2^d \cdot \varepsilon_{\mathsf{prg}} + t^{-1/3}$, and Ext is a randomness extractor capable of extracting m bits from any $2m$-bit source Y with $\mathbf{H}_\infty(Y) \geq 2m - d$. Clearly, our time t is much better than $t^{1/3}$. As for security, our security $\varepsilon \approx \sqrt{2^d \cdot \varepsilon_{\mathsf{prg}}}$ is slightly worse only when $t^{-2/3} \ll \varepsilon_{\mathsf{prg}} \cdot 2^d$. Even in this case, we can see that

$$\varepsilon' \geq \max(\varepsilon^2, \varepsilon \cdot t^{-1/6})$$

The first inequality $\varepsilon' \geq \varepsilon^2$ is obvious, and the second inequality is true since $a + b \geq \sqrt{ab}$, which means that $\varepsilon' \approx 2^d \cdot \varepsilon_{\mathsf{prg}} + t^{-1/3} \geq \sqrt{2^d \cdot \varepsilon_{\mathsf{prg}}} \cdot t^{-1/6} \geq \varepsilon \cdot t^{-1/6}$. Thus, the original approach gains at most the square in the security, and never more than a multiplicative $t^{1/6}$ factor.

Additionally, our approach is likely to use fewer public random bits S, since $\log(1/\varepsilon_{\mathsf{prg}})$ is likely much less than the seed length of the required extractor Ext.

Concrete Instantiation. Finally, we observe the following peculiar coincidence: the most natural implementation of the extractor Ext for the previous dense-model approach and the pairwise independent hash function h_y in our alternative approach result in the *identical final constructions*. Namely, the simplest $2m$-to-m bit extractor Ext would be the universal[14] hash function $g_s(a, b) = a \cdot s + b$, which is also the same as the standard pairwise independent hash function $h_{a,b}(s) = a \cdot s + b$, where $a, b, s \in GF[2^m]$. Thus, if we let $G(x) = (a, b)$, both the previous and the new approach will use public randomness $s \in GF[2^m]$ to derive a pseudorandom m-bit key $r = a \cdot s + b$, after which applying the PRG will obviously result in a $2m$-bit pseudorandom value $G(r)$.

Curiously, despite resulting in identical constructions, the two supporting proofs are completely different (ours is *much* simpler) and yield very different (incomparable) time-security tradeoffs as well, as explained above. It would be interesting to find a unified proof, ideally giving a "continuous transition" from one security tradeoff to the other.

Acknowledgements: We would like to thank our co-authors in previous work [1, 12] for useful comments and suggestions.

Yevgeniy Dodis was supported by NSF Grants CNS-1065134, CNS-1065288, CNS-1017471, CNS-0831299 and Google Faculty Award. Yu Yu was supported by the National Basic Research Program of China Grant 2011CBA00300, 2011CBA00301, the National Natural Science Foundation of China Grant 61033001, 61172085, 61061130540, 61073174, 61103221, 11061130539, 61021004 and 60703031.

[14] Indeed, for any $(a, b) \neq (a', b')$, $\Pr_s[as + b = a's + b']$ is 2^{-m} if $a \neq a'$, and is 0 if $a = a'$ but $b \neq b'$.

References

[1] Boaz Barak, Yevgeniy Dodis, Hugo Krawczyk, Olivier Pereira, Krzysztof Pietrzak, François-Xavier Standaert, and Yu Yu. Leftover hash lemma, revisited. In Phillip Rogaway, editor, *CRYPTO*, LNCS, pages 1–20. Springer, 2011.

[2] Boaz Barak and Shai Halevi. A model and architecture for pseudo-random generation with applications to /dev/random. In *Proceedings of the 12th ACM Conference on Computer and Communication Security*, pages 203–212, 2005.

[3] Boaz Barak, Ronen Shaltiel, and Eran Tromer. True random number generators secure in a changing environment. In *Proceedings of the 5th Cryptographic Hardware and Embedded Systems*, pages 166–180, 2003.

[4] Xavier Boyen, Yevgeniy Dodis, Jonathan Katz, Rafail Ostrovsky, and Adam Smith. Secure remote authentication using biometric data. In Ronald Cramer, editor, *Advances in Cryptology—EUROCRYPT 2005*, volume 3494 of *LNCS*, pages 147–163. Springer-Verlag, 2005.

[5] Zvika Brakerski and Oded Goldreich. From absolute distinguishability to positive distinguishability. *Electronic Colloquium on Computational Complexity (ECCC)*, 16:31, 2009.

[6] J.L. Carter and M.N. Wegman. Universal classes of hash functions. *Journal of Computer and System Sciences*, 18:143–154, 1979.

[7] Gil Cohen, Ran Raz, and Gil Segev. Non-malleable extractors with short seeds and applications to privacy amplification. In *27th conference on computational complexity*, pages 110–124.

[8] Nenad Dedic, Danny Harnik, and Leonid Reyzin. Saving private randomness in one-way functions and pseudorandom generators. In *5th Theory of Cryptography Conference*, pages 607–625, 2008.

[9] Yevgeniy Dodis, Jonathan Katz, Leonid Reyzin, and Adam Smith. Robust fuzzy extractors and authenticated key agreement from close secrets. In Cynthia Dwork, editor, *Advances in Cryptology—CRYPTO 2006*, volume 4117 of *LNCS*, pages 232–250. Springer-Verlag, 20–24 August 2006.

[10] Yevgeniy Dodis, Xin Li, Trevor D. Wooley, and David Zuckerman. Privacy amplification and non-malleable extractors via character sums. In *Proceedings of the 52nd IEEE Symposium on Foundation of Computer Science*, pages 668–677, 2011.

[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] Yevgeniy Dodis, Thomas Ristenpart, and Salil P. Vadhan. Randomness condensers for efficiently samplable, seed-dependent sources. In *9th Theory of Cryptography Conference*, pages 618–635, 2012.

[13] Yevgeniy Dodis and Adam Smith. Correcting errors without leaking partial information. In *Proceedings of the Thirty-Seventh Annual ACM Symposium on Theory of Computing*, pages 654–663, Baltimore, Maryland, 22–24 May 2005.

[14] Yevgeniy Dodis and Daniel Wichs. Non-malleable extractors and symmetric key cryptography from weak secrets. In Michael Mitzenmacher, editor, *Proceedings of the 41st Annual ACM Symposium on Theory of Computing*, pages 601–610, Bethesda, MD, USA, 2009. ACM.

[15] Stefan Dziembowski and Krzysztof Pietrzak. Leakage-resilient cryptography. In *Proceedings of the 49th IEEE Symposium on Foundation of Computer Science*, pages 293–302, 2008.

[16] Benjamin Fuller, Adam O'Neill, and Leonid Reyzin. A unified approach to deterministic encryption: New constructions and a connection to computational entropy. In *9th Theory of Cryptography Conference*, pages 582–599, 2012.

[17] Rosario Gennaro, Hugo Krawczyk, and Tal Rabin. Secure hashed diffie-hellman over non-ddh groups. In Christian Cachin and Jan Camenisch, editors, *Advances in Cryptology—EUROCRYPT 2004*, volume 3027 of *LNCS*, pages 361–381. Springer-Verlag, 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] Hugo Krawczyk. Cryptographic Extraction and Key Derivation: The HKDF Scheme. In Tal Rabin, editor, *Advances in Cryptology - CRYPTO 2010*, volume 6223 of *LNCS*, pages 631–648. Springer-Verlag, 2010.

[20] Xin Li. Non-malleable extractors, two-source extractors and privacy amplification. In *Proceedings of the 53rd IEEE Symposium on Foundation of Computer Science*, pages xxx–xxx, 2012.

[21] Ueli Maurer and Stefan Wolf. Privacy amplification secure against active adversaries. In Burton S. Kaliski, Jr., editor, *Advances in Cryptology—CRYPTO '97*, volume 1294 of *LNCS*, pages 307–321. Springer-Verlag, 1997.

[22] Moni Naor and Omer Reingold. Synthesizers and their application to the parallel construction of pseudo-random functions. *J. Comput. Syst. Sci.*, 58(2):336–375, 1999.

[23] Krzysztof Pietrzak. A leakage-resilient mode of operation. In Antoine Joux, editor, *Advances in Cryptology - EUROCRYPT 2009*, volume 5479 of *LNCS*, pages 462–482. Springer-Verlag, 2009.

[24] Ran Raz and Omer Reingold. On recycling the randomness of states in space bounded computation. In *Proceedings of the 31st ACM Symposium on the Theory of Computing*, pages 159–168, 1999.

[25] Omer Reingold, Ronen Shaltiel, and Avi Wigderson. Extracting randomness via repeated condensing. *SIAM J. Comput.*, 35(5):1185–1209, 2006.

[26] Omer Reingold, Luca Trevisan, Madhur Tulsiani, and Salil P. Vadhan. Dense subsets of pseudorandom sets. In *Proceedings of the 49th IEEE Symposium on Foundation of Computer Science*, pages 76–85, 2008.

[27] D. R. Stinson. Universal hashing and authentication codes. *Designs, Codes, and Cryptography*, 4(4):369–380, 1994.

[28] Terence Tao and Tamar Ziegler. The primes contain arbitrarily long polynomial progressions, 2006. http://arxiv.org/abs/math.NT/0610050.

[29] Luca Trevisan and Salil Vadhan. Extracting randomness from samplable distributions. In *41st Annual Symposium on Foundations of Computer Science*, pages 32–42, Redondo Beach, California, November 2000. IEEE.