Randomness in Cryptography April 6, 2013

Lecture 12: Computational Key Derivation

Lecturer: Yevgeniy Dodis *Scribe: Benjamin Fuller*

In the previous lecture, we discussed methods of key derivation, in particular, how to beat the RT bound for square-friendly applications and defined the notion of unpredictability extractors. In this question, we'll ask if we can do any better by allowing computational assumptions. Such notions may be helpful for beating information-theoretic bounds for applications that consider only computational adversaries.

We'll begin with a recap of the information theoretic key derivation mechanisms from the previous class.

1 Last Class

If we are giving a high entropy source R, $\mathbf{H}_\infty(R) \geq k$ the problem of producing a uniform key is known as key derivation. We will consider an application P that achieves security δ when given a uniform key. We also consider a setting of parameters for $\delta = 2^{-64}$ and key length $m = 128$.

1. **Arbitrary Applications:** Our first options works for any application P but has the worst bounds. In setting we use a randomness extractor (e.g. a pairwise independent hash function through leftover hash lemma [11]). This gives us the following bound: $\delta' \leq \delta_{ALL} \stackrel{def}{=} \delta + \sqrt{2^{m-k}}$.

   Achieve $\delta' = O(\delta)$ when $k \geq m + 2\log 1/\delta$. We also consider a concrete parameters $\delta = 2^{-64}$ and $m = 128$. For these parameters, need $k = 256$.

   Achieve no security when $k = m$. Concrete parameters, achieve $\delta' = 1$.

2. **Square friendly applications:** Again we use LHL, but consider only square friendly applications. Let P be a square friendly application with security δ and square security $\sigma = O(\delta)$. Then we achieve real security $\delta' \leq \delta_{SQF} \stackrel{def}{=} \delta + \sqrt{\delta 2^{m-k}}$.

   Achieve $\delta' = O(\delta)$ when $k \geq m + \log 1/\delta$. Concrete parameters need $k = 192$.

   Achieve security $\delta' \approx \sqrt{\delta}$ when $k = m$. Concrete parameters achieve $\delta' = 2^{-32}$.

3. **Unpredictability applications:** This was the major new mechanism introduced in the previous class. We first showed the following theorem:

   Theorem 1 *Let $h : \{0,1\}^n \times \{0,1\}^\nu \to \{0,1\}^m$ be a $O(\log 1/\delta)$-independent hash. Then h is a (k, δ, δ')-unpredictability extractor (this is equivalent to a good $\mathbf{H}_\infty$-condensor) for $\delta' \leq \delta_{UNP} \stackrel{def}{=} \delta + O(\delta \log(1/\delta 2^{m-k}))$. This means for all unpredictability applications P with ideal security δ we achieve (k, δ')-security in the real model.*

An unpredictability extractor works for any unpredictability application P. Achieving security as above. In particular, this yields the following bounds:

Achieve $\delta' = O(\delta)$ when $k \geq m + \log\log 1/\delta$. Concrete parameters need $k = 138$.

Achieve security $\delta' = O(\delta \log 1/\delta)$ when $k = m$. Concrete parameters achieve $\delta' = 2^{-57.9}$.

4. **Heuristic Security:** We will use a random oracle as a reference point for comparison[1]. This is a heuristic KDF, we assume that h acts as a random oracle, then we achieve $\delta' \leq \delta_{RO} = \delta + q/2^k$ where q is the number of queries an adversary can make to the random oracle. For most applications $q \leq \delta 2^m$. This means that $\delta_{RO} = \delta + \delta 2^{m-k}$.

   Achieve $\delta' = O(\delta)$ when $k = m$. Concrete parameters need $k = 128$

   Achieve security $\delta' = O(\delta)$ when $k = m$. Concrete parameters achieve $\delta \approx 2^{-64}$.

Question: Can we beat these bounds using computational assumptions? We'll first consider whether these bounds are optimal in the information-theoretic setting.

2 Optimality of bounds for 1.-3.

2.1 LHL

We begin with the bound for an arbitrary application. Radhakrishnan and Ta-Shma [13] showed that to achieve a (k, ε) extractor then $\varepsilon \geq \sqrt{2^{m-k}}$. They showed for any candidate extractor Ext there exists a source X and a distinguisher D that fools it. In [13] both the source X and D are inefficient. It was an open question [3] whether both items must be inefficient. This is called the SRT conjecture. In [5], it was shown that having X be efficiently samplable does not change the bound. This result is unconditional in the sense that the bound still applies if Ext is not required to be efficient.

Open Question: Can the RT bound be overcome for inefficient sources and efficient distinguishers?

2.2 Square-Friendly LHL

For square-friendly applications can we achieve on $\delta' = O(\sqrt{\delta 2^{m-k}})$? In the previous lecture we showed this answer is no. That is, there is a SQF application P such that $\delta' = O(\sqrt{\delta 2^{m-k}})$ is tight for all Ext. In this setting as above we can make X efficiently samplable but D is inefficient. Recall the application was a square-friendly key derivation function.

2.3 Unpredictability Applications

In [5], the authors show that the bound of 3. is essentially optimal for unpredictability applications. They also show how to reduce seed length from $O(n \log 1/\delta)$ to $O(k \log k) \leq$

[1]Note that this technique only works for computationally secure applications. It does not work for applications like one time encryption or MAC.

$O(n \log n)$ We know that by combinatorial arguments that $O(\log n + \log 1/\delta)$ seed length is possible.

Open Question: Do there exists efficient schemes that achieve seed length $O(\log n + \log 1/\delta)$?

Open Question: Is this bound tight for natural applications P like CPA encryption or a weak PRF?

Instead of viewing the result of [5] as an unpredictability extractor we can view it as a randomness condenser that is $\mathsf{Ext}(X;S), S \approx_\delta (Y,S)$ where $\mathbf{H}_\infty(Y|S) \geq m - \ell$ where ℓ is entropy deficiency. Where $\ell = 0$ this is an extractor and subject to the same bounds as 1. when $\ell = 1$ then we only need $k \geq m + \log\log 1/\delta$ and when $\ell = \log\log 1/\delta + O(1)$ we get $k \geq m$. Thus, a slight entropy deficiency is very helpful. We now return to our original question, can we do beat these bounds in computational settings?

3 Computational Key Derivation

As described in [3], a natural replacement to an extractor is a computational extractor: DEFINITION 1 Let $\mathsf{Ext} : \{0,1\}^n \times \{0,1\}^\nu \to \{0,1\}^m$. Ext is a (t,k,ε)-computational extractor if for all t-bounded distinguishers D,

$$\Delta^D(U_m, \mathsf{Ext}(X;S)|S) \leq \varepsilon$$

We can also state this bound as $\mathsf{CD}_t(U_m, \mathsf{Ext}(X;S)|S) \leq \varepsilon$. ♢

Observation 2 *If P is a (t,δ)-secure application and Ext is a (t,k,ε)-computational extractor then using $\mathsf{Ext}(X;S)$ results in (t,δ')-real security for $\delta' = \delta + \varepsilon$.*

3.1 Extract-then-expand

A reasonable idea to build a computational extractor is to first build an extractor and then expand the output using a pseudorandom generator. We first define a pseudorandom generator:

DEFINITION 2 Let $G : \{0,1\}^{m'} \to \{0,1\}^m$ be a function. We say that G is a (t,ε)-pseudorandom generators if for all t-bounded $D, \Delta_D(G(U_{m'}), U_m) \leq \varepsilon$ ♢

Lemma 1 (Extract-then-expand [12]) *If $G : \{0,1\}^{m'} \to \{0,1\}^m$ is a (t,ε_{prg})-pseudorandom generator and $\mathsf{Ext}' : \{0,1\}^n \times \{0,1\}^\nu \to \{0,1\}^{m'}$ is a (k,ε_{ext})-extractor then $\mathsf{Ext}(x,s) = G(\mathsf{Ext}'(x,s))$ is a (t,k,ε') computational extractor where $\varepsilon' = \varepsilon_{ext} + \varepsilon_{prg}$.*

Consider an application P with ideal security δ. Using a computational extractor we achieve security $\delta' \leq \delta + \varepsilon_{ext} + \varepsilon_{prg}$. As before, we would like to set $\varepsilon_{ext} \approx \varepsilon_{prg} \approx \delta$ and find out what k is necessary. Recall for an extractor to get $\varepsilon_{ext} = \delta$ we need $k \geq m' + 2\log 1/\delta$.

We also need a (t,δ)-secure pseudorandom generator from m' to m bits. The good news is that once we can stretch from $m' \to m'+1$ we can stretch from m' to m with a little cost. So we just need a (t,δ)-secure PRG from $m' \to m'+1$. So how low can m' be?

Practice: A reasonable setting of $m' = 128$ (64 is considered too low). In this setting there is no need to expand our key, we can just directly use 128 bits. No explicit pseudorandom generator is necessary.

Theory: In [4], De, Tulsiani, and Trevisan show that m' must be at least $2\log 1/\varepsilon_{prg}$ when $t = O(m')$. They also provide a general (non-uniform) bound: $m' \geq 2\log 1/\varepsilon_{prg} + \log t/m'$.

These two results imply that for the extract-then-expand strategy to be helpful we need $k \geq 2\log 1/\delta + m' \geq 2\log 1/\delta + 2\log 1/\delta = 4\log 1/\delta$. So to achieve $\delta = 2^{-64}$ we need $k \geq 256$ to obtain arbitrary key length output.

Question 1 *Is extract-then-expand helpful?*

Yes: In the medium-to-high entropy regimes, extract-then-expand easily beats the RT bounds and achieves $m >> k - 2\log 1/\delta$.
No: Extract-then-expand fails in the low entropy regimes we most care about improving (i.e. when $k \leq 256$). We could also consider formulations other than $\delta = \varepsilon_{ext} = \varepsilon_{prg}$.

Aside: Is a pseudorandom generator (or a one-way function) necessary to beat the RT bound computationally?

Theorem 3 *Let s be a security parameter and let $\mathsf{Ext} : \{0,1\}^{n(\cdot)} \times \{0,1\}^{\nu(\cdot)} \to \{0,1\}^{m}$ be a family of $(t(s), k(s), \varepsilon(s))$-computational extractors. If $m >> k - 2\log 1/\varepsilon - \Omega(1)$ and $t(s) \geq \mathsf{poly}(s)$ and $\varepsilon(s) = \mathsf{negl}(s)$, then a (t', ε')-PRG exists for $t' \geq \mathsf{poly}(s)$ and $\varepsilon' = \mathsf{negl}(s)$.*

Proof:[Sketch] We will use the following result of Goldreich [10]:

Theorem 4 *[10] Let $c_1, c_2 > 0$ be two constants with $c_1 < c_2$. Let A, B be efficiently samplable distributions such that A, B are computationally indistinguishable (for all poly-t, $\Delta^t(A, B) < c_1\varepsilon$) and $\mathsf{SD}(A, B) \geq c_2\varepsilon$, then one-way functions (and thus pseudorandom generators exist)*[2]*.*

Using the above result we know that being the RT bound there must exist an efficiently samplable distribution X such that $\mathsf{SD}((S, \mathsf{Ext}(X; S)), (S, U_m)) > 5\varepsilon$. However, since we have a computational extractor:

$$\Delta^t((S, \mathsf{Ext}(X; S)), (S, U_m)) < \varepsilon.$$

These two statements satisfy the conditions of Theorem 4, thus, pseudorandom generators exist. □

Thus, we know in order to build a computational extractor we must build a pseudorandom generator. We now ask if we run a condenser instead of an extractor to get savings (recall our constructions of condensers are significantly better than constructions of extractors).

[2]The original statement of the theorem is that the statistical distance is $1/\mathsf{poly}$ while the computational distance is negl.

3.2 Condense-Expand-Extract

As we saw in the previous section, extract-then-expand requires a significant amount of starting entropy, we will try a new paradigm, extracting from a pseudoentropy generator, we start by providing definition of a computational condenser:

DEFINITION 3 An efficient function $\mathsf{PEG} : \{0,1\}^n \to \{0,1\}^m$ is a $(\frac{n-d'}{n} \to \frac{m-d}{m}, t, \varepsilon)$-pseudoentropy generator if for all X where $\mathbf{H}_\infty(X) \geq n - d'$ there exists a distribution Y where $\mathbf{H}_\infty(Y|S) \geq m - d$ and for all t-bounded D,

$$\Delta^D((\mathsf{PEG}(X,S),S),(Y,S)) \leq \varepsilon.$$

$\diamondsuit$

Note: It should be clear for a pseudoentropy generator $d \geq d'$ as the source can simply fix bits of the output of PEG. Thus, our goal is to increase the overall entropy not decrease the entropy gap. It is still an interesting object if we increase the entropy gap, that is: $d > d'$.

The hope is that a pseudoentropy generator can be built significantly efficiently than a computational extractor. Essentially, we want a "pseudorandom generator" that works when given any high entropy distribution. If we could build a pseudoentropy generator more efficiently than extract-then-expand we could then extract its output. The construction is below:

Theorem 5 *Let* $\mathsf{PEG} : \{0,1\}^n \to m'$ *be a* $(\frac{n-d'}{n} \to \frac{m'-d}{m'}, t, \varepsilon_{peg})$*-pseudoentropy generator and let* $\mathsf{Ext}' : \{0,1\}^{m'} \times \{0,1\}^\nu \to \{0,1\}^m$ *be a* $(m' - d, \varepsilon_{ext})$ *extractor, then* $\mathsf{Ext}(X;S) = \mathsf{Ext}'(\mathsf{PEG}(X);S)$ *is a* $(t, n - d', \varepsilon_{peg} + \varepsilon_{ext})$ *computational extractor.*

Thus, if we could build a pseudoentropy generator efficiently we may be able to avoid the problems encountered in the extract-then-expand paradigm. Note that a pseudoentropy generator is a difficult task, it expand the apparent entropy of all high entropy distributions. We'll start by asking how well pseudorandom generators perform on slightly deficient distributions. The hope is that we can then use a condenser (where constructions are significantly better than extractors) followed by a pseudorandom generator (to build a pseudoentropy generator).

Construction 1 *Let* $\mathsf{Cond} : \{0,1\}^n \times \{0,1\}^\mu \to \{0,1\}^{m_1}$ *be a* $(\frac{k}{n} \to \frac{m_1-d'}{m_1}, \varepsilon)$ *condenser and let* $G : \{0,1\}^{m_1} \to \{0,1\}^{m_2}$ *be a* (t, ε_{prg}) *pseudorandom generator. Define* $\mathsf{PEG}(x;s) = G(\mathsf{Cond}(x;s)), s$.

In order to show that Construction 1 is a pseudoentropy generator we must show that a pseudorandom generator still works on slightly deficient distributions. This is known as the dense model theorem and appears in several works [7, 14, 8, 9] (see Appendix B of [8] for a comparison).

Theorem 6 *Let* $G : \{0,1\}^{m_1} \to \{0,1\}^{m_2}$ *be a* (t, ε_{prg}) *pseudorandom generator. Let* $\varepsilon_{dense} > 0$ *be a parameter,* G *is a* $(\frac{m_1-d'}{m_1} \to \frac{m_2-d'}{m_2}, t\varepsilon^2_{dense}, \varepsilon 2^d + \varepsilon_{dense})$ *pseudoentropy generator.*

Proof:[Sketch] Let X be a distribution with $\mathbf{H}_\infty(X) \geq m_1 - d'$. The overall idea is to show if for any Y with entropy $\mathbf{H}_\infty(Y) \geq m_2 - d'$ there exists a distinguisher D for $G(X)$ and Y is also a good distinguisher for $G(U_{m_1})$ and U_{m_2}.

The proof proceeds in two stages: first we show if we have a distinguisher D that works for all high min-entropy distributions Y (this is a weaker statement, in the theorem a different distinguisher may work for each Y) then we show that D distinguishes $G(U_{m_1})$ and U_{m_2}. Second, we show how to build a distinguisher for all high min-entropy distributions Y out of distinguishers for each distribution Y. The second stage of the proof comes from [1, Theorem 5.2][3]. We focus on the first part. The proof proceeds roughly as follows,

1. Suppose D distinguishes $G(X)$ from any distribution Y of min-entropy $m_2 - d'$ with advantage ε' ($\varepsilon' = \varepsilon 2^{d'}$ the remainder of the loss in parameters occurs in the second part).

2. Show that one of two conditions holds:

   For all Y with min-entropy $m_2 - d'$, $\mathbb{E}[D(Y)]$ is lower than $\mathbb{E}[D(G(X)]$ by ε',

   Or for all such Y, $\mathbb{E}[D(Y)]$ is higher than $\mathbb{E}[D(G(X))]$ by at least ε'.

   This follows by convexity of min-entropy distributions: if there were a Y_1 Y_2 with $D(Y_1) < D(G(X))$ and $D(Y_2) > D(G(X))$ then we could build a distribution Y_3 out of Y_1, Y_2 with $D(Y_3) = D(G(X))$.

3. Assume the former without loss of generality. This initial step allows us to remove absolute values and to find a high-entropy distribution Y^+ on which $\mathbb{E}[D(Y^+)]$ is the highest.

4. Show that for every y outside of Y^+, D outputs 0, and that Y^+ is essentially flat (roughly, all points where D outputs 1 occur with the same probability). Use these two facts to show an upper bound on $\mathbb{E}[D(U_{m_2})]$.

5. Provide a lower bound on $\mathbb{E}[D(G(U_{m_1}))]$ based the performance of D on $G(X)$.

□

Parameter Settings There is a significant loss in Theorem 6. One possible option for parameters is to set $\varepsilon_{dense} = t^{-1/3}$ this makes $t' = t\varepsilon_{dense} = t^{1/3}$ and $\varepsilon' = \varepsilon 2^d + t^{-1/3}$. For most natural settings ε' will be dominated by the second term giving $t' = t^{1/3}$ and $\varepsilon' \approx t^{-1/3}$. Another possible option is to try keep $\varepsilon' = O(\varepsilon 2^{d'})$ this means setting $\varepsilon_{dense} = \varepsilon 2^{d'}$. This leads to settings $\varepsilon' = \varepsilon 2^{d'}$ and $t' = t(\varepsilon)^2 2^{2d'}$. For most reasonable settings of ε, t this is not a very meaningful security guarantee.

We now look back at Construction 1 using the analysis from Theorem 6:

Corollary 2 *Let P be an unpredictability application with ideal security δ. Let $\mathsf{Cond} : \{0,1\}^n \times \{0,1\}^\nu \to \{0,1\}^{m_1}$ be a $(\frac{k}{n} \to \frac{m_1 - \log\log 1/\delta}{m_1}, \delta)$ condenser and let $G : \{0,1\}^{m_1} \to \{0,1\}^{m_2}$ be a (t, ε_{prg}). Then using $G(\mathsf{Cond}(X;S)), S$ results in real security $\delta' = \delta + \varepsilon_{prg} \log 1/\delta + \varepsilon_{dense} + \delta = 2\delta + \varepsilon_{prg} \log 1/\delta + t^{-1/3}$ and $t' = t\varepsilon_{dense}^2 = t^{1/3}$ for $\varepsilon_{dense} = t^{-1/3}$.*

[3]There are several technical conditions in this theorem. The distinguisher is deterministic and output values in a range, much of the work in the first part of the proof is dealing with these conditions.

For practical parameters $k = 128, \varepsilon_{prg} = 2^{-64}, t = 2^{64}$ and $\delta = 2^{-64}$ this yields $\delta' = 2^{-64}2^6 + 2^{-64/3} \approx 2^{-21}$ the last term ε_{dense} is the term that makes this solution impractical.

Note: we could then extract from Construction 1 to yield a computational extractor but the loss in Theorem 6 prevents this from being very useful. This would make the construction applicable in all applications (not just unpredictability applications).

We now have two options for a computational extractor: extract-then-expand and condense-expand-extract. Extract-then-expand has decent parameters but only in the mid-to-high entropy regimes while parameter losses in condense-expand-extract make it impractical for use. We will now discuss our last technique to build a computational extractor.

3.3 wPRF's

Up until now we have used a truly random seed and stretching the input to an extractor. We will now try the opposite, using computational techniques to stretch the seed of an extractor. We begin by reviewing weak pseudorandom functions:

DEFINITION 4 Let $\mathcal{F}$ represent the set of all functions from $\{0,1\}^n \to \{0,1\}^m$. Let $\mathsf{KeyGen}(1^n) \to \{0,1\}^\nu$ be a function. The family of functions $f : \{0,1\}^n \times \{0,1\}^\nu \to \{0,1\}^m$ (indexed by s generated by KeyGen) is a (q,t,ε) if the following holds. Let $X_1, ..., X_q$ be uniformly sampled from $\{0,1\}^n$ for all t-bounded distinguishers D:

$$\Pr_{s \leftarrow \mathsf{KeyGen}(\cdot)}[D((X_1, f_s(X_1)), ..., (X_q, f_s(X_q)))] - \Pr[g \leftarrow \mathcal{F}][D((X_1, g(X_1)), ..., (X_q, f(X_q))] < \varepsilon.$$

♢

The double run trick works for wPRF's giving the following result:

Theorem 7 *[6, Theorem 3.2] Let F be a $(2q, 2t, \delta)$-wPRF in the ideal model. Then F is $(q,t,\delta))$-square secure and $(q,t,\sqrt{2^{d-1}\delta})$-secure in the $(m-d)$-real$_2$ model.*

Thus, a wPRF secure for 2 queries yields a key derivation function secure against one query in the ideal model (note that it beats the bounds in Theorem 6). This means wPRF's are computational Renyi entropy extractors. Thus, we try to construct an efficient wPRFsecure against two queries.

Lemma 3 *Let $G : \{0,1\}^m \to \{0,1\}^{2m}$ be a (t,ε)-pseudorandom generator and let $h : \{0,1\}^m \times \{0,1\}^{2m} \to \{0,1\}^m$ be a pairwise independent hash function (for example $h_{(a,b)}(s) = as + b$). Define the function $f_x(s) = h_{g(x)}(s)$. The following hold:*

1. *f is a $(t, 2, \varepsilon + \frac{1}{2^m})$-wPRF.*
2. *f is $(t, 1, \varepsilon + \frac{1}{2^m})$-square secure.*
3. *f is a $\left(t, m-d, \sqrt{2^d(\varepsilon + \frac{1}{2^m})}\right)$-computational extractor (for $\mathbf{H}_2$) that produces an m-bit key.*

Proof:[Sketch] Let $h : \{0,1\}^m \times \{0,1\}^{2m} \to \{0,1\}^m$ be any pairwise independent hash function. Then h is a $(\infty, 2, 2^{-m})$-secure wPRF. The only problem is that we need $2m$ bits as the seed for h. By a hybrid argument $h_{G(U_m)}(U_m)$ is a $(2t, 2, \varepsilon_{prg} + 2^{-m})$-wPRF. The lemma follows by application of Theorem 7. □

Notes: We have removed the dependence of t', ε' that occurred in Theorem 6, this allows for significantly better parameters in most cases. We also only require that $\mathbf{H}_2$ is high allowing for a larger class of sources.

We can also compose the efficient condensers we constructed in the previous lecture to produce a stronger key derivation function:

Corollary 4 *Let* $\mathsf{Cond} : \{0,1\}^n \times \{0,1\}^\mu \to \{0,1\}^m$ *be a* $(\frac{k}{n} \to \frac{m-d}{m}, \varepsilon)$*-condenser and let* $f : \{0,1\}^m \times \{0,1\}^m \to \{0,1\}^m$ *be the* wPRF *defined above. Let* $\mathsf{Ext} : \{0,1\}^n \times \{0,1\}^{\mu+m} \to \{0,1\}^m$ *be defined as* $g(x, s_1, s_2) = h_{G(\mathsf{Cond}(x;s_1))}(s_2)$*. Then* Ext *is a* $(t, k, \sqrt{2^d(\varepsilon_{prg} + \frac{1}{2^m})} + \varepsilon)$*-computational extractor. In particular, let* P *be an application with ideal security* δ*, using our optimal condenser we achieve real security* $\delta' = \delta + \sqrt{\varepsilon_{prg} 2^{m-k}}$*.*

References

[1] Boaz Barak, Ronen Shaltiel, and Avi Wigderson. Computational analogues of entropy. In *11th International Conference on Random Structures and Algorithms*, pages 200–215, 2003.

[2] Kai-Min Chung, Yael Tauman Kalai, Feng-Hao Liu, and Ran Raz. Memory delegation. In *CRYPTO*, pages 151–168, 2011.

[3] Dana Dachman-Soled, Rosario Gennaro, Hugo Krawczyk, and Tal Malkin. Computational Extractors and Pseudorandomness in TCC 2012.

[4] Anindya De, Luca Trevisan, and Madhur Tulsiani. Time space tradeoffs for attacks against one-way functions and prgs. Advances in CryptologyCRYPTO 2010. Springer Berlin Heidelberg, 2010. 649-665.

[5] Yevgeniy Dodis, Krzysztof Pietrzak, and Daniel Wichs. "Key Derivation Without Entropy Waste." (2013).

[6] Yevgeniy Dodis and Yu Yu. Overcoming weak expectations. Theory of Cryptography. Springer Berlin Heidelberg, 2013. 1-22.

[7] S. Dziembowski and K. Pietrzak. Leakage-Resilient cryptography. In *IEEE 49th Annual IEEE Symposium on Foundations of Computer Science, 2008.*, pages 293–302, 2008.

[8] Benjamin Fuller, Adam O'Neill, and Leonid Reyzin. A unified approach to deterministic encryption: New constructions and a connection to computational entropy. Cryptology ePrint Archive, Report 2012/005, 2012.

[9] Craig Gentry and Daniel Wichs. Separating succinct non-interactive arguments from all falsifiable assumptions. *STOC. ACM, New York*, pages 99–108, 2011.

[10] Oded Goldreich. A note on computational indistinguishability. Information Processing Letters 34.6 (1990): 277-281.

[11] Johan Håstad, Russell Impagliazzo, Leonid A. Levin, and Michael Luby. A pseudorandom generator from any one-way function. *SIAM J. Comput.*, 28(4):1364–1396, 1999.

[12] Hugo Krawczyk. Cryptographic extraction and key derivation: The HKDF scheme. Advances in CryptologyCRYPTO 2010. Springer Berlin Heidelberg, 2010. 631-648.

[13] Jaikumar Radhakrishnan and Amnon Ta-Shma. Bounds for dispersers, extractors, and depth-two superconcentrators. SIAM Journal on Discrete Mathematics 13.1 (2000): 2-24.

[14] O. Reingold, L. Trevisan, M. Tulsiani, and S. Vadhan. Dense Subsets of Pseudorandom Sets. In *2008 49th Annual IEEE Symposium on Foundations of Computer Science*, pages 76–85. IEEE, 2008.