Randomness in Cryptography April 11, 2013

Lecture 13: Seed-Dependent Key Derivation

Lecturer: Yevgeniy Dodis *Scribe: Eric Miles*

In today's lecture, we study seeded key-derivation functions (KDFs) in the setting where the source of weak randomness X is dependent on the seed S, and we give two approaches to constructing KDFs in this setting. Then, we generalize the setting to one in which the attacker is given some additional "side information" that depends on X and S; we define a notion of condensers that are secure in the presence of such attackers, and show how they can be applied to instantiate the Fiat-Shamir heuristic.

1 Introduction

So far in this class, we have studied seeded key-derivation functions (KDF). Namely, we have constructed functions $\mathsf{KDF} : \{0,1\}^n \times \{0,1\}^v \to \{0,1\}^m$, and computed our secret key as $R := \mathsf{KDF}(X, S)$, where X is the weak source of randomness and S is a truly random seed. We have been implicitly using the following two assumptions.

(a) The source X is *independent* of the seed S.

(b) The seed S is *authentic*. (We will discuss formally what this means in future lectures.)

Assumption (a) could be problematic in a number of scenarios. For instance, in models for leakage-resilient cryptography it may be the case that the attacker can learn some information about S before influencing the distribution X. It could also be problematic in real-world random number generators: think of the weak randomness X being the temperature of the computer, which is affected by the computation producing the seed S.

In today's lecture, we will will build KDFs in scenarios where there is some dependence between the seed and the weak source of randomness.

2 Seed-dependent key derivation

To start, how should we model the seed dependency? We will think of having a probabilistic sampling algorithm A that sees the seed and produces the weak source of randomness, i.e. we will have $X \leftarrow A(S)$. As usual we will require that X has some level of entropy, e.g. $\mathbf{H}_\infty(X|S) \geq k$ for some parameter k.

It is not hard to see that constructions from previous lectures will not work "off the shelf" in this scenario. This is because we can use a sampler A to break a (standard) extractor $\mathsf{Ext} : \{0,1\}^n \times \{0,1\}^v \to \{0,1\}^m$ using essentially the same ideas that show deterministic extraction is impossible. Namely, given a seed $s \in \{0,1\}^v$, A will choose X uniformly over the set $\{x \mid \mathsf{Ext}(x, s)_1 = 0\}$ of strings for which the first bit of Ext's output is 0. This is a distribution of min-entropy essentially $n-1$ and yet for every s, $\mathsf{Ext}(A(s), s)$ is far from uniform. Even if we require that X is efficiently samplable, the sampler $A(s)$ can choose (say) n uniform values of $x \in \{0,1\}^n$, and output the first such that $\mathsf{Ext}(x,s)_1 = 0$ (or an arbitrary one if none satisfy this).

We will consider two interpretations of this impossibility result.

Interpretation 1: We cannot allow the sampler to run the extractor.

There are several interesting options for how to model such a restricted sampler. One is to use a so-called "split state" model, in which the sampler is made up of two (non-interacting) functions A_1 and A_2, the seed s is split as (s_1, s_2), and the source X is computed as $X := A_1(s_1) \circ A_2(s_2)$. Another, which we will examine now, is to restrict the running time of the sampler to be less than that of the extractor.

In fact, we have already seen in previous lectures how this can be helpful. Recall that we proved the following theorem.

Theorem 1 *Let $\mathcal{X} = \{X_i\}$ be a set of distributions over $\{0,1\}^n$ such that $|\mathcal{X}| \leq T$ and $\mathbf{H}_\infty(X_i) \geq k$ for all $X_i \in \mathcal{X}$. Then if $\mathcal{H} = \{h : \{0,1\}^n \to \{0,1\}^m\}$ is an $O(\log T)$-wise independent hash family where $m \leq k - 2\log(1/\varepsilon) - \log\log T - O(1)$, we have*

$$\Pr_{h \leftarrow \mathcal{H}}[\exists X_i \in \mathcal{X} \,:\, \mathsf{SD}(h(X_i), U_m \,|\, h) \geq \varepsilon] \leq 2^{-n}.$$

To apply this to our setting, note that (the description of) the function $h \in \mathcal{H}$ is the seed, and imagine that we restrict the sampler so that for every h, $A(h)$ outputs some distribution $X_i \in \mathcal{X}$. Then the above theorem guarantees that $\mathsf{Ext}(A(h), h)$ will be close to uniform with overwhelming probability over the choice of h.

The only question is how to enforce this restriction on the distributions that A outputs. To do this, we allow A to choose a circuit C_h of size $\leq t$ after seeing h, and then C_h is applied to uniformly random bits to output a sample. (An even more restricted option would be to allow a single size-t circuit with two inputs, where the first is set to h and the second is uniformly random.) As the number of circuits of size $\leq t$ is $2^{O(t \log t)}$, we can set $T = 2^{O(t \log t)}$ and apply the above theorem, with $\mathcal{X}$ being the set of all distributions of min-entropy $\geq k$ samplable by circuits of size $\leq t$, to guarantee that A can break the extractor with probability at most $\varepsilon + 2^{-n}$. Note that the time needed to compute the extractor (i.e. the hash function) is $\Omega(\log T) = \Omega(t \log t)$, which is indeed greater than the time t allowed to the sampler.

Interpretation 2: We cannot allow the KDF to be a good extractor.

Another way to circumvent the impossibility result is to allow the sampler to run in more time than the KDF, but to not require that the KDF computes an extractor. This will lead us to the notion of seed-dependent condensers, defined as follows.

DEFINITION 1 Let $c \in \{2, \infty\}$. A function $\mathsf{Cond} : \{0,1\}^n \times \{0,1\}^v \to \{0,1\}^m$ is a $(k, d, \varepsilon, t)_c$-*seed-dependent-condenser (SD-condenser)* if for every A running in time $\leq t$ that computes a distribution $X \leftarrow A(S)$ where $S \equiv U_v$ and $\mathbf{H}_c(X \,|\, S) \geq k$, there exists a distribution Y such that

1. $\mathsf{SD}(\mathsf{Cond}(X, S), Y \,|\, S) \leq \varepsilon$, and
2. $\mathbf{H}_c(Y \,|\, S) \geq m - d$.

$\diamondsuit$

A few notes are in order. First, the differences from the seed-independent setting are that the sampler A gets S and that A is restricted to run in time $\leq t$. When the entropy deficiency d is 0 we recover the notion of seed-dependent extractors, but as we have seen this necessitates $t_{\mathsf{Cond}} > t$ (where t_{Cond} denotes the running time of the condenser). Unlike the extractor setting, for condensers it may be possible to achieve $\varepsilon = 0$, i.e. it may be possible to have $\mathbf{H}_c(\mathsf{Cond}(X, S) \mid S) \geq m - d$. Finally, the attack mentioned at the beginning of this section essentially shows that we must have $d \geq \log(t/t_{\mathsf{Cond}})$, which is not too bad for us if we can make it tight. (See [1] for more details on this last bound.)

We conclude this section by recalling two lemmas from a previous lecture, which easily generalize to today's setting. By the "$(t, k)_c$-real model", we specify that the sampler A must run in time $\leq t$ and output a distribution X such that $\mathbf{H}_c(X \mid S) \geq k$.

Lemma 1 *If P is a (T, δ)-secure unpredictability application and* Cond *is a $(k, d, \varepsilon, t)_\infty$-SD-condenser, then P is (T, δ')-secure in the $(t, k)_\infty$-real model when using* Cond *as a KDF, where $\delta' \leq \varepsilon + \delta \cdot 2^d$.*

Lemma 2 *If P is a (T, σ)-square-secure application and* Cond *is a $(k, d, \varepsilon, t)_2$-SD-condenser, then P is (T, δ')-secure in the $(t, k)_2$-real model when using* Cond *as a KDF, where $\delta' \leq \varepsilon + \sqrt{\sigma \cdot 2^d}$.*

3 Seed-dependent condensers

In this section we show how to construct SD-condensers. Our main focus will be a direct, simple, and even "philosophically interesting" construction in the $\mathbf{H}_2$ setting. We will also mention a generic conversion from $\mathbf{H}_2$ SD-condensers to $\mathbf{H}_\infty$ SD-condensers. It is open to get better SD-condensers in the $\mathbf{H}_\infty$ setting than those resulting from this conversion, unlike in the seed-independent setting which we saw how to improve in previous lectures.

Recall that in the seed-independent setting, condensers were simply universal hash functions. Namely we had $\mathsf{Cond}(x, h) := h(x)$ where h was chosen uniformly from a set $\mathcal{H}$ such that $\Pr_{h \leftarrow \mathcal{H}}[h(x) \neq h(x')] \leq 2^{-m}$ for all $x \neq x'$. In the seed-dependent setting, we will instead use *collision-resistant* hash functions, defined next. It is natural that we should look to cryptographic hash functions to achieve SD-condensers, because we are trying to fool a large class of sampling algorithms (e.g. all efficiently samplable distributions). Furthermore, this gives some justification for the widespread practice of using (say) SHA-1 as a KDF in real-world applications; this is the "philosophically interesting" point mentioned earlier.

DEFINITION 2 A set $\mathcal{H} = \{h : \{0,1\}^n \to \{0,1\}^m\}$ is a (t, δ)-*collision-resistant (CR)* hash family if for all probabilistic B running in time $\leq t$

$$\Pr_{h \leftarrow \mathcal{H}}[B(h) \to (x_1, x_2) \ : \ h(x_1) = h(x_2) \wedge x_1 \neq x_2] \leq \delta.$$

$\mathcal{H}$ is (t, δ)-*independent-collision-resistant (iCR)* if for all probabilistic A running in time $\leq t$

$$\Pr_{h \leftarrow \mathcal{H}}\left[A^{(1)}(h) \to x_1, A^{(2)}(h) \to x_2 \ : \ h(x_1) = h(x_2) \wedge x_1 \neq x_2\right] \leq \delta$$

where $A^{(1)}$ and $A^{(2)}$ denote two independent executions of A. $\diamondsuit$

We give a few remarks on what can be achieved. It is not hard to see that every $(2t, \delta)$-CR hash family is also (t, δ)-iCR. If we allow non-uniform attackers then every (t, δ)-iCR hash family is also $(t, 2\delta)$-CR, because A can hardwire the best randomness for B and then choose randomly whether to output x_1 or x_2. By the standard "birthday attack", the best δ that we can hope to achieve for CR hash families is $\Omega(t^2/2^m)$. In fact, this is also the best δ that we can achieve for iCR hash families, as follows. Imagine an attacker A that chooses t uniformly random inputs, runs h on each, and if there is a collision $h(x) = h(x')$ then A randomly outputs one of x or x'. This attack succeeds with probability $\Omega(t^2/2^m)$ if we can guarantee that A chooses the same t inputs in both executions, which we can do by (non-uniformly) hardwiring A's randomness to the value that maximizes its success (though A still of course randomly outputs one of x or x').

The following lemma shows that iCR hash families suffice to construct SD-condensers in the $\mathbf{H}_2$ setting.

Lemma 3 *If $\mathcal{H} = \{h : \{0,1\}^n \to \{0,1\}^m\}$ is a (t, δ)-iCR hash family where $\delta = 2^{d-1}/2^m$, then $\mathsf{Cond}(x, h) := h(x)$ is a $(k, d, 0, t)_2$-SD-condenser for any $k \geq m - d + 1$.*

Proof: Let $A : \mathcal{H} \to \{0,1\}^n$ be any sampler running in time $\leq t$, and consider the following calculation where as usual Col denotes collision probability.

$$\begin{aligned}\mathsf{Col}(h(A(h)) \,|\, h) &= \Pr\left[A^{(1)}(h) \to x_1, A^{(2)}(h) \to x_2 \ : \ h(x_1) = h(x_2)\right] \\ &\leq \Pr[x_1 = x_2] + \Pr[h(x_1) = h(x_2) \ \wedge \ x_1 \neq x_2] \\ &\leq \mathsf{Col}(A(h) \,|\, h) + \delta.\end{aligned}$$

Observe that $\mathsf{Col}(A(h) \,|\, h) \leq 2^{-k} \leq 2^{d-1}/2^m$ by the assumption on the $\mathbf{H}_2$ entropy of A's output distribution. Since also $\delta \leq 2^{d-1}/2^m$, we have $\mathsf{Col}(h(A(h)) \,|\, h) \leq 2^d/2^m$ and thus Cond is a $(k, d, 0, t)_2$-SD-condenser. $\square$

Combining Lemmas 2, 3, and the preceding discussion gives the following corollary.

Corollary 4 *If P is a (t, σ)-square-secure application then, when using a $(2t, \delta)$-CR hash family as a KDF, P is (t, δ')-secure in the $(t, \log(1/\delta))_2$-real model, where $\delta' \leq \sqrt{\sigma \cdot \delta \cdot 2^{m+1}}$.*

In particular, if σ is negligible and t is polynomial, then δ' is negligible.

The following lemma gives a generic conversion from SD-condensers in the $\mathbf{H}_2$ setting to those in the $\mathbf{H}_\infty$ setting.

Lemma 5 *If Cond is a $(k, d, 0, t)_2$-SD-condenser, then for every $\gamma > 0$ Cond is also a $(k, d + \log(1/\gamma), \gamma, t)_\infty$-SD-condenser.*

4 Condensers with side-information

In this section we generalize the previous setting, and allow the sampler A to pass some additional information Z to the predictor P. More precisely, where before we had $A(S) \to$

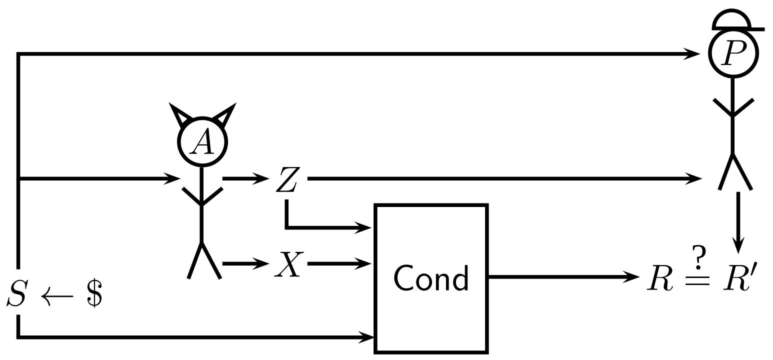

Figure 1: Condenser with side-information

X, now we will have $A(S) \to (X, Z)$ where $\mathbf{H}_\infty(X \mid S, Z) \geq k$ and Z is given to P in addition to the output R (which is either uniformly random or is $\mathsf{Cond}(X, S)$).

Unfortunately this is not achievable in general: $A(S)$ can choose X uniformly at random and send $Z := \mathsf{Cond}(X, S)$ to P. This satisfies $\mathbf{H}_\infty(X \mid S, Z) \geq n - m$, but clearly allows P to perfectly distinguish the output of the condenser. To get around this limitation, we also pass Z to the condenser. This notion, called a $(k, d, \varepsilon, t)_\infty$*-condenser with side information*, is depicted in Figure 1. Observe that if we can build such a condenser, then we can prove versions of Lemmas 1 and 2 in the "$(t, k)_\infty$-real model with side information", where $k = n - \ell$ and ℓ is the bit-length of the side information.

This is already non-trivial to achieve when the sampler A is restricted to choosing X uniformly at random, and for the rest of this section we will restrict ourselves to this special case. Note that we can thus view the jointly distributed (X, S) as a single uniform source X (which is input to both A and the condenser). The following definition formalizes the construction we wish to obtain, and is depicted in Figure 2.

Definition 3 A function $\mathsf{Cond} : \{0,1\}^n \times \{0,1\}^\ell \to \{0,1\}^m$ is an $(\ell, d, \varepsilon, t)$*-leaky condenser* if for every A running in time $\leq t$ that computes a distribution $Z \leftarrow A(X)$ where $X \equiv U_n$, there exists a distribution Y such that

1. $(Y, Z) \approx_\varepsilon (\mathsf{Cond}(X, Z), Z)$, and
2. $\mathbf{H}_\infty(Y \mid Z) \geq m - d$.

$\diamondsuit$

It is unknown if such leaky condensers exist, and obtaining a construction under any standard cryptographic assumption is an important open problem. Note that the construction from CR hash functions (Lemma 3) breaks down, because the second time around you need to run $A(h)$ conditioned on some fixed Z. It is even open to obtain a construction in the Random Oracle (RO) model; one difficulty here is specifying the right formalization for the min- and Rényi-entropy of the RO. Still, it seems reasonable to conjecture that leaky condensers (and condensers with side information) exist with roughly the same parameters as in the seed-dependent setting of Lemma 3.

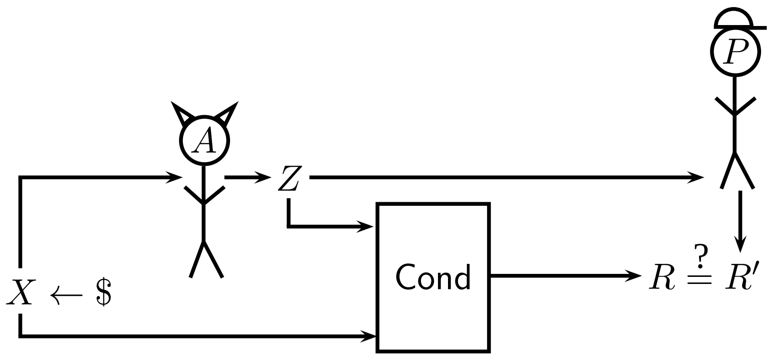

Figure 2: Leaky condenser

One potential approach (which according to the lecturer seems "likely to fail") is to use distributional collision-resistant hash functions, defined as follows.

Definition 4 A set $\mathcal{H} = \{h : \{0,1\}^n \to \{0,1\}^m\}$ is a (t,δ)-*distributional-collision-resistant (DCR)* hash family if for all probabilistic B running in time $\leq t$, the following holds. Sample $(x,z) \leftarrow B(h)$, and let X' be the marginal distribution on the first element of $B(h)$'s output conditioned on the second being z. Then, $\Pr[h(x) = h(X') \wedge x \neq X'] \leq \delta$. $\diamondsuit$

Question 1 *Can DCR hash families be used to construct leaky condensers, analogously to Lemma 3?*

Finally, we end today's lecture with an application of leaky condensers, namely instantiating the Fiat-Shamir heuristic.

4.1 The Fiat-Shamir heuristic

The Fiat-Shamir heuristic [2] is a construction for transforming an interactive, constant-round, public-coin protocol into a single-round (i.e. non-interactive) protocol. Its soundness can be proven in the RO model, but today we will see how leaky condensers can be used to instantiate the construction. We will focus on the following restricted case: transforming 3-round Σ-protocols (defined next) into 2-round public-coin protocols, or 1-round protocols in the Common Reference String (CRS) model.

Definition 5 Let L be a language consisting of pairs (α,β). A Σ-*protocol* for L, between a prover P who holds (α,β) and a verifier V who holds only α, is a 3-round protocol defined as follows. P first sends a commitment z, next V responds with a uniformly random challenge r, and finally P sends a response p.

The protocol is (t,δ)-*sound* if for every (possibly cheating) prover P^* who holds only α and runs in time $\leq t$, $V(\alpha)$ accepts with probability $\leq \delta$ after interacting with $P^*(\alpha)$, over a uniform choice of α. $\diamondsuit$

As an example, we now describe the well-known Schnorr protocol [3] for discrete-log. Let G be a cyclic group of order q, and let $g \in G$ be a generator. Then the language L is $\{(\alpha,\beta) \mid \alpha = g^\beta \in G;\ \beta \leq q\}$. The protocol is the following.

1. P chooses $\gamma \leq q$ uniformly at random, and sends $z := g^{\gamma}$.
2. V chooses $r \leq q$ uniformly at random and sends it.
3. P computes and sends $p := \gamma + r\beta \pmod q$.

   V accepts iff $g^p = z \cdot \alpha^r$.

This protocol is *zero-knowledge*, meaning that at the end V has learned nothing beyond the fact that P knows β. It also satisfies the following special soundness condition: for any two transcripts (z, r_1, p_1) and (z, r_2, p_2) with $r_1 \neq r_2$, if they both cause V to accept then β can be computed as $\beta = (p_1 - p_2)/(r_1 - r_2)$. Thus, under the assumption that no time-t algorithm can compute discrete-log in G except with probability $\leq \delta$ over a uniform choice of α, this protocol is (t, δ)-sound.

The Fiat-Shamir heuristic collapses any Σ-protocol (P, V) to a 2-round protocol (P', V') as follows. Let $\mathcal{H}$ be a family of functions to be specified. The protocol is the following.

1. V' chooses $h \leftarrow \mathcal{H}$ uniformly at random and sends it.
2. P' computes z as P would, computes $r := h(z)$, computes p from this r as P would, and sends (z, p).

   V' accepts iff V would on transcript (z, r, p). (Note that V' can compute r from z.)

Note that if the function h is specified by a shared random string (a.k.a. the CRS model), this can be made into a 1-round (non-interactive) protocol. The following question is open.

Question 2 *Can we construct hash families $\mathcal{H}$ such that (P', V') is sound for any sound (P, V)?*

Towards an answer to this question, let $z \in \{0,1\}^{\ell}$ and $r \in \{0,1\}^m$, and let $\mathsf{Cond} : \{0,1\}^n \times \{0,1\}^{\ell} \to \{0,1\}^m$ be an $(\ell, d, \varepsilon, t)_{\infty}$-leaky condenser for some n. Then let h be specified by a uniformly random $X \leftarrow U_n$, and compute $h(z) := \mathsf{Cond}(X, z)$.

For intuition as to why this works, consider a cheating prover P^*. P^* is given a random $X \in U_n$ (i.e. the description of h) and to successfully convince V' to accept, P^* must implicitly construct an accepting transcript $(z, r = \mathsf{Cond}(X, z), p)$. If we could guarantee that r was ε-close to the uniform distribution, then this would increase the protocol's soundness error by an additive factor of ε. Instead, viewing the transcript as the side information Z, the leaky-condenser property guarantees that r is ε-close to a distribution with entropy deficiency d. Thus the soundness error increases by an additive factor of ε and a multiplicative factor of 2^d.

This intuition can be made formal, as follows [1, Thm. 6.1].

Theorem 2 *If (P, V) is a (t, δ)-sound Σ-protocol with ℓ-bit commitments and m-bit challenges, and $\mathsf{Cond} : \{0,1\}^n \times \{0,1\}^{\ell} \to \{0,1\}^m$ is an $(\ell, d, \varepsilon, t)_{\infty}$-leaky condenser with $n = \ell$, then the protocol (P', V') using $h = \mathsf{Cond}$ as above is (t', δ')-sound for $\delta' = 2^d(\varepsilon + \delta)$ and $t' = t - O(\ell)$.*

The paper [1] also shows that some notion of leaky condensers are *necessary* for the soundness of the Fiat-Shamir heuristic, and thus provide the right abstraction for what type of construction is necessary.

We conclude with an open question.

Question 3 *Do $\mathbf{H}_2$ leaky-condensers suffice in the above construction?*

References

[1] Yevgeniy Dodis, Thomas Ristenpart, and Salil P. Vadhan. Randomness condensers for efficiently samplable, seed-dependent sources. In *TCC 2012*.

[2] Amos Fiat and Adi Shamir. How to prove yourself: practical solutions to identification and signature problems. In *CRYPTO 1986*.

[3] C.P. Schnorr. Efficient identification and signatures for smart cards. In *CRYPTO 1989*.