Foundations of Cryptography: Lecture 7

Lecture by Moni Naor, scribe notes by Noga Amit

December 6, 2021

1 PRFs for Authentication

Last time we talked about using **Pseudorandom Functions** (PRFs) for *Encryption*. Now we want to find out how can we use PRFs for *Authentication*. The settings are the same as before: we have Alice and Bob, Alice wants to authenticate and Bob asks her to prove that she's indeed Alice.

We may have many Bobs and everyone except Eve shares a secret key $s \in_R \{0,1\}^n$. Eve listens to the communication between Alice and Bob and then wants to convince some Bob that she's Alice. Eve wins if after seeing m sessions she manages to create another session. In other words, if the system is secure, then the only sessions created are sessions which were initiated by Alice.

We assume that each Bob has an ID: $Bob_1, Bob_2, Bob_3, ...$

The protocol for Bob_i is as follows:

- Bob sends $r_{b_i} \in_R \{0,1\}^n$.
- Alice sends $F_s(r_{b_i} \circ i)$.
- Bob calculates $F_s(r_{b_i} \circ i)$ and checks if it's equal to the message he received.

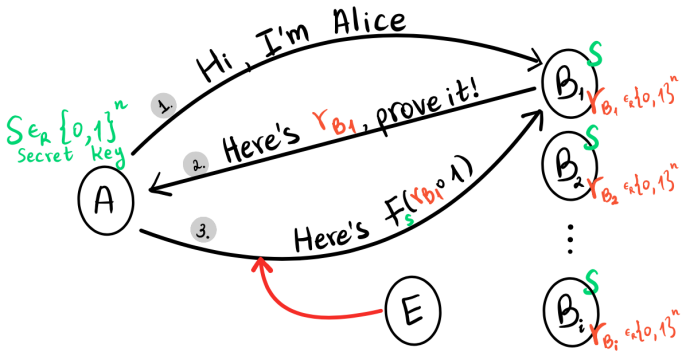


Figure 1: The Authentication protocol using a PRF

Security: Eve cannot guess the value $F_s(r_{b_j} \circ j)$, which with high probability has never appeared before: after m sessions, the probability of seeing a repetition is $\frac{\binom{m}{2}}{2^n}$. Therefore, if F is a truly random function, then she cannot guess it, and if it's a PRF, guessing would mean breaking the PRF.

Why should we concatenate Bob's index? To prevent a *Man in the Middle* attack. Suppose Alice wants to authenticate to Bob_1 and Eve wants to authenticate as Alice to Bob_2, and suppose that Eve controls the line. That way, whatever message Bob_2 is sending, Eve can divert it as if it's the message that Bob_1 is sending. Alice would think that she's talking to Bob_1 when she's actually talking to Bob_2.

Can we make this scheme into a public-key one? In other words, a scheme where Alice and the Bobs do not share a secret s but rather some public key derived from s (which is known only to Alice). The answer is *yes*, and we will discuss it when we will talk about zero-knowledge proofs. We can even make such schemes into signature schemes using the *Fiat-Shamir Heuristic*[1].

2 Domain Extension

In the same context, one may ask what can we do if Alice wants to send a message as well, instead of solely authenticate. That's simple: we can modify the second step of the protocol such that Alice sends $\langle m, F_s(m \circ r_{b_i} \circ i) \rangle$. However, in order to do so, we need to *expand the domain* of the PRF.
Generally speaking, there are two questions: how to expand the **domain** of a PRF and how to expand its **range**. The second one is easier; we can simply use two keys and apply it twice. In the future, we'll see how to generate two keys from only one key. However, extending the domain is harder:

Given a PRF $F_s : \{0,1\}^n \to \{0,1\}^n$*, we want to build* $F'_s : \{0,1\}^{2n} \to \{0,1\}^n$ *which would also be a PRF.*

Attempt 1: Interpret the input $x \in \{0,1\}^{2n}$ as $x = x_L \circ x_R$ and define $F'_s(x_L \circ x_R) = F_s(x_L) \oplus F_s(x_R)$.
Problem: $\forall x\ F'_s(x \circ x) = 0^n$.

Attempt 2: Use two secrets and define $F'_{s_1 s_2}(x_L \circ x_R) = F_{s_1}(x_L) \oplus F_{s_2}(x_R)$ in the same manner.
Problem: Linearity. Taking 3 messages that are linearly independent would reveal the image of a 4th message:

$$x_L^1 \circ x_R^1 \implies F_{s_1}(x_L^1) \oplus F_{s_2}(x_R^1)$$

[1] Fiat, Amos; Shamir, Adi (1987). "How To Prove Yourself: Practical Solutions to Identification and Signature Problems". Advances in Cryptology — CRYPTO' 86.

$$x_L^1 \circ x_R^2 \implies F_{s_1}(x_L^1) \oplus F_{s_2}(x_R^2)$$
$$x_L^2 \circ x_R^1 \implies F_{s_1}(x_L^2) \oplus F_{s_2}(x_R^1)$$

Would yield the image of $x_L^2 \circ x_R^2$ since the XOR of all of them sums up to zero.

Attempt 3: Define $F'_s(x_L \circ x_R) = F_s(x_L)$.
Problem: The same prefix would give the same result. Therefore, after seeing the image of a single message, one can get the image of any other message with the same prefix.

Solution: Define $F'_s(x_L \circ x_R) = F_s(F_s(x_L) \oplus x_R)$.
Security: The intuition behind its security is that the probability of a collision is very small. Formally, we should condition on a bad event which has a negligible probability. We define

$$BAD = \{\exists\ i \neq j :\ F_s(x_L^i) \oplus x_R^i = F_s(x_L^j) \oplus x_R^j\}$$

for two different inputs $x^i, x^j \in \{0,1\}^{2n}$ and m trials. When F_s is truly random,

$$\Pr[BAD] = \frac{\binom{m}{2}}{2^n}$$

which is negligible if n is large enough. Now, this already gives us security against non-adaptive attack, since if all messages are chosen ahead of time, the probability of breaking F'_s is exactly the probability of getting a collision, and we just saw that it's negligible. However, we also get security against adaptive attacks using the conditional probability on BAD:

$$\begin{aligned}
&\Pr[\text{breaking } F'_s] = \\
&\quad \Pr[\text{collision in } F'_s | BAD] * \Pr[BAD] + \Pr[\text{collision in } F'_s | \overline{BAD}] * \Pr[\overline{BAD}] \leq \\
&\quad \Pr[BAD] + \Pr[\text{collision in } F'_s \text{ for two distinct inputs}] * \Pr[\overline{BAD}] \approx \\
&\quad \Pr[BAD] + \frac{m}{2^n} = \textit{negligible}\,.
\end{aligned}$$

The last equality is followed by the fact that conditioned on $\overline{BAD}$ (i.e., no collisions occurred), the distribution of $\{F_s(x_L) \oplus x_R\}$ is close to uniform.
The key observation here is that the adversary does not see the inner inputs $F_s(x_L) \oplus x_R$; they are embedded inside the definition of F'_s.

What if we want to use this scheme ℓ times? We use a pairwise independent function[2]. This is called *Levin's Trick*: we add a pairwise independent function $h : \{0,1\}^{\ell n} \to \{0,1\}^n$ to the shared secret and apply

$$F'_{s,h}(x) = F_s(h(x))$$

That is the easiest solution for solving the *domain extension* problem. As before, the proof of security is based on defining a BAD event as a collision and conditioning on it.

[2] $h \in_R H$ and for all $x \neq x' \in \{0,1\}^{\ell n}, \Pr[h(x) = h(x')] \approx \frac{1}{2^{\ell n}}$.

3 PRFs for Encryption: revisited

Last time we saw how to use PRFs for encryption:

- Alice picks $r_A \in_R \{0,1\}^n$ and sends Bob $\langle r_A, F_s(r_A) \oplus m \rangle$.
- Bob gets $\langle r_A, c \rangle$, calculates $F_s(r_A)$ and decrypts $m = c \oplus F_s(r_A)$.

However, suppose Eve knows what m is, which makes sense in certain circumstances (e.g., Alice sends "Hello" to Bob every morning). If Eve is passive, there's not much she can do with this assumption; however if Eve is active, she can change the message easily by sending her new message (e.g. "Attack at dawn") XORed with Alice's original message, since

$$(F_s(r_A) \oplus \text{"Hello"}) \oplus (\text{"Hello"} \oplus \text{"Attack at dawn"}) = F_s(r_A) \oplus \text{"Attack at dawn"}$$

which would give Eve a legal ciphertext. To be secure against this kind of attacks, known as *malleability*, the simplest thing would be sending

$$\langle r_A, F_s(r_A) \oplus m, F_s(m) \rangle$$

and when Bob decrypts $\langle r_A, c, v \rangle$, he outputs $m = c \oplus F_s(r_A)$ only if $v = F_s(m)$.

4 Indistinguishability

Consider the following test: the adversary Eve chooses m_0, m_1, gets a ciphertext $c = Enc(m_b)$ for some $b \in_R \{0,1\}$ and has to guess b. We define two type of resistance against Eve's attack:

- **CPA** (Chosen Plaintext Attack): Eve can choose to see $Enc(m)$ as long as $m \neq m_0, m_1$.
- **CCA** (Chosen Ciphertext Attack):
  - **CCA1** (aka *Lunch Time*): Eve can choose to see $Dec(c)$ for every c before she gets the test.
  - **CCA2**: Eve can choose to see $Dec(c)$ even during the test, as long as $c \neq Enc(m_b)$.

4.1 CPA Security

We prove that the encryption scheme described in the previous section is **CPA-secure**. The proof essentially relies on the security of **One-time pad**: since the XOR operator is invertible, if $F_s(r_i)$ acts uniformly then so does $m_i \oplus F_s(r_i)$. The following figure shows that if the CPA-Attacker has advantage δ in guessing m'_b, then the PRF-Attacker has advantage $\frac{\delta}{2}$ in distinguishing F_s from a truly random function. This reduction yields that the security of the PRF implies CPA security for our encryption scheme.

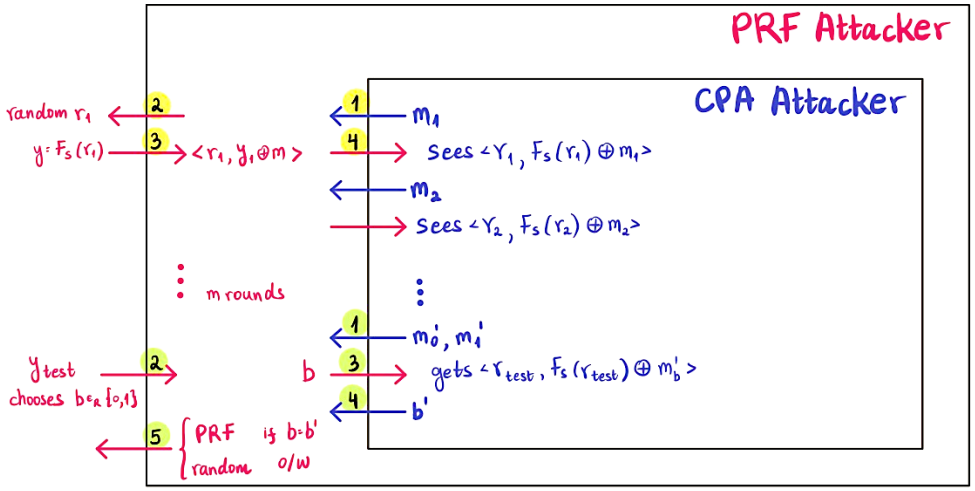


Figure 2: CPA/CCA1 Attacker to a PRF Attacker

4.2 CCA Security

We'll consider **CCA1**. It turns out that if we add $v = F_s(m)$ like we did earlier, we also get CCA1-security with the same proof. We argue that we cannot learn anything from the decryption function.
In order to do so, we need to extend the protocol by allowing Bob to return "fail" as output, which we indicate with a $\perp$. So, Alice sends

$$\langle r_A, F_s(r_A) \oplus m, F_s(m) \rangle$$

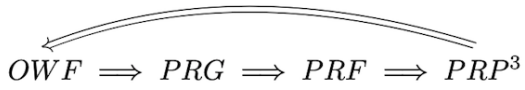
and when Bob decrypts $\langle r_A, c, v \rangle$ he outputs $\begin{cases} m = c \oplus F_s(r_A) & \text{if } v = F_s(m) \\ \perp & \text{otherwise} \end{cases}$
We claim that adding the $\perp$ doesn't leak information. Specifically, we claim that every message which Eve creates would end up in Bob outputting a $\perp$. The security follows directly from the same proof presented in Figure 2.

Why isn't this scheme CCA2-secure? Eve can flip all the bits of the given ciphertext, get the flipped message — since she can ask about any ciphertext different from the one in the test — flip the result and get the original message. That way, she may distinguish between every pair of messages.

5 PRGs

5.1 An overview of our related primitives

The last thing we haven't dealt with so far in our foregoing discussion about PRFs was *how* to construct one. Like in many other cases, it turns out that we can rely on the existence of a weaker primitive, called **PRG**, which would be defined next. But first, let's examine the hierarchy between them:



where the $\Longrightarrow$ is existential, e.g. we can existentially construct a PRG from a OWF.

5.2 A detour: OWFs from PRFs

We now turn to a direct construction of a OWF from a PRF. The construction would imply that if OWFs do not exist, then PRFs do not exist.

Attempt: Define $g(x) = F_s(x)$ where s is some known string.
Problem: If we take F which is actually a PRP then it's invertible, so g is invertible and therefore not one-way.

Solution: $g(x) = F_x('0') \circ F_x('1')$.
Security: The intuition is that one cannot expect to find an inverse to a truly random function. Formally, assume that we have two random strings $y_1 \circ y_2 \in \{0,1\}^n$, then $\Pr[\exists x.y_0 \circ y_1 = F_x('0') \circ F_x('1')] \leq \frac{1}{2^n}$. Therefore, if g is not a OWF, then one may distinguish between F and a truly random function, in contradiction to F being a PRF.

5.3 Definition

A poly-time function $G : \{0,1\}^n \to \{0,1\}^{m(n)}$, where $m(n) > n$, would be called a **PRG** if G's output on $s \in_R \{0,1\}^n$ is indistinguishable from $r \in_R \{0,1\}^{m(n)}$. s is the secret key and also called the *seed*.

5.4 Discussion

Note that G is not a function in the sense that we only run it once. However, we can think of G as a PRF of a very small domain, of $log(n)$ bits. Since the

[3] In general, PRP is an invertible PRF which models block ciphers like AES. We would learn about them in the next lectures.

domain is small, we can "ask" about all of it, which suggests another possible perspective to viewing G as the outputs $PRF(0), PRF(1), PRF(2), ...$

The inherent difference from a PRF is that we get the output at once: putting n bits and getting $m(n)$ bits with no back and forth.

The security relies on the fact that indeed $m(n) > n$: since

$$\Pr[\text{guessing the seed}] = \frac{2^n}{2^{m(n)}},$$

we have to make sure this probability is small enough.
For example, taking $m = 2n$ would do, since $\frac{2^n}{2^{2n}} = \frac{1}{2^n}$ which in negligible in n.
Next time we would build a PRF from a PRG.