Lecture 4: PRFs, CPA-Secure Encryption, and GGM

MIT - 6.5620
Lecture 4 (September 21, 2026)

> ***Warning:*** This document is a rough draft and may contain bugs. Please feel free to email us with corrections.

Recap

Last lecture we covered the following:

1. We defined the notion of a pseudorandom generator.

   Definition 1. An efficient (poly-time computable) deterministic function $G : \{0,1\}^\lambda \to \{0,1\}^{n(\lambda)}$ is said to be a pseudorandom generator if the following two conditions hold:

   - It is expanding; i.e., $n(\lambda) > \lambda$.
   - It is pseudorandom, i.e.

     $$\{G(U_\lambda)\}_{\lambda\in\mathbb{N}} \approx \{U_{n(\lambda)}\}_{\lambda\in\mathbb{N}},$$

     where U_ℓ is the uniform distribution over $\{0,1\}^\ell$.

2. We proved that pseudorandomenss is equivalent to the following "next-bit unpredictability" property.

   Definition 2. An expanding function $G : \{0,1\}^\lambda \to \{0,1\}^{n(\lambda)}$ is next-bit unpredictable if for every poly-size adversary $\mathcal{A}$ there exists a negligible function μ such that for every $\lambda \in \mathbb{N}$ and every $i \in [n(\lambda)]$

   $$\Pr_{U\leftarrow\{0,1\}^\lambda}[\mathcal{A}(G(U_\lambda)_{[i-1]}) = G(U_\lambda)_i] = 1/2 + \mu(\lambda)$$

   where $G(U_\lambda)_{[i-1]}$ denotes the first $i-1$ bits of $G(U_\lambda)$ and $G(U_\lambda)_i$ denotes the i'th bit of $G(U_\lambda)$.

*Pseudorandom Functions (*PRF*)*

At the end of the previous lecture we briefly introduced pseudorandom functions. We begin by recalling their definition.

Definition 3 (Pseudorandom function)**.** An efficiently computable pseudorandom function family $F = \{F_\lambda\}_{\lambda\in\mathbb{N}}$, where for every $\lambda \in \mathbb{N}$,

$F_\lambda : \mathcal{K}_\lambda \times \mathcal{X}_\lambda \to \mathcal{Y}_\lambda$, has the property that for every poly-size $\mathcal{A}$ there exists a negligible function μ such that for every $\lambda \in \mathbb{N}$,

$$|\Pr_{k \leftarrow \mathcal{K}_\lambda}[\mathcal{A}^{F_\lambda(k,\cdot)}(1^\lambda) = 1] - \Pr_{R_\lambda}[\mathcal{A}^{R_\lambda(\cdot)}(1^\lambda) = 1]| \leq \mu(\lambda)$$

where R_λ is a truly random function $R_\lambda : \mathcal{X}_\lambda \to \mathcal{Y}_\lambda$. For concreteness, we think of $\mathcal{X} = \{0,1\}^n$ and $\mathcal{Y} = \{0,1\}$.

We return to secret-key encryption and show how pseudorandom functions allow us to securely encrypt polynomially many messages without maintaining state.

Security against Chosen-Plaintext Attacks

We now strengthen the one-message security definition from Lecture 2. We allow the adversary to obtain encryptions of polynomially many messages of its choice, chosen adaptively based on previously seen ciphertexts. This notion is called security against chosen-plaintext attacks (CPA security).

Definition 4. An encryption scheme $(\mathsf{Enc}, \mathsf{Dec})$ is said to be secure against *adaptively chosen plaintext attacks* (CPA secure) if for every PPT adversary $\mathcal{A}$ there exists a negligible function μ such that for every $\lambda \in \mathbb{N}$, $\mathcal{A}$ wins in the following game with probability at most $\frac{1}{2} + \mu(\lambda)$:

- The challenger chooses a key $k \leftarrow \mathcal{K}_\lambda$.
- The adversary $\mathcal{A}$ given 1^λ chooses a message $m_i \in \mathcal{M}_\lambda$ and receives $c_i \leftarrow \mathsf{Enc}_\lambda(k, m_i)$.

  This step can be repeated polynomially many times.
- The adversary $\mathcal{A}$ chooses $m_0, m_1 \in \mathcal{M}_\lambda$.
- The challenger chooses a random bit $b \leftarrow \{0,1\}$, generates $c \leftarrow \mathsf{Enc}(k, m_b)$, and sends the ciphertext c to the adversary.
- The adversary given c outputs a bit b'.

We say that $\mathcal{A}$ wins if $b' = b$.

Remark. Note that it suffices to construct a CPA secure encryption scheme for single bit messages; i.e., for $\mathcal{M} = \{0,1\}$. The reason is that we can encrypt arbitrarily long messages bit-by-bit. Therefore, for simplicity, in what follows we focus on constructing a CPA secure encryption scheme with $\mathcal{M} = \{0,1\}$.

Constructing a CPA*-Secure Encryption Scheme*

We first observe that if we could expand the secret key to an arbitrarily long pad then we could use it to encrypt all our messages,

each time using a fresh part of the pad. This scheme would result in a stateful scheme since we need to remember which part of the pad was already used.

It turns out that this is inherent! Specifically, there does not exist a CPA secure encryption scheme where the encryption algorithm is deterministic (unless it is stateful)! The reason is that it is always easy to distinguish between $(\mathsf{Enc}(k,m),\mathsf{Enc}(k,m))$ and $(\mathsf{Enc}(k,m),\mathsf{Enc}(k,m'))$ for randomly chosen $m,m' \in \mathcal{M}_\lambda$. Indeed, in all our CPA secure encryption schemes the encryption algorithm is *randomized*.

Continuing with our intuition above, if we could expand our secret key to an extremely long pad, then each time we encrypt a message, we can use a random part of the pad, and hope that the pad is long enough that we never reuse the same part. The question is how can we expand our secret key into such a long pad *efficiently*?

A pseudorandom function gives us exactly such an implicitly represented long pad. Let $F = \{F_\lambda\}_{\lambda\in\mathbb{N}}$ be a PRF family where

$$F_\lambda : \{0,1\}^\lambda \times \{0,1\}^\lambda \to \{0,1\}.$$

We define

$$\mathsf{Enc}_\lambda(k,m;r) = (r, m \oplus F_\lambda(k,r)),$$

where $r \xleftarrow{R} \{0,1\}^\lambda$, and

$$\mathsf{Dec}_\lambda(k,(r,c)) = F_\lambda(k,r) \oplus c.$$

Theorem 5. *The encryption scheme defined above is* CPA *secure assuming the underlying function F is a* PRF*.*

Proof. The CPA security follows immediately from the definition of a PRF. Specifically, suppose there exists a poly-size $\mathcal{A}$ and there exists a non-negligible ϵ such that $\mathcal{A}$ wins in the CPA game with probability at least $\frac{1}{2} + \epsilon(\lambda)$. We construct a poly-size $\mathcal{B}$ that breaks the PRF security of F.

$\mathcal{B}^O(1^\lambda)$ distinguishes between the case that $O = F_\lambda(k,\cdot)$ and the case that $O = R_\lambda$, as follows:

1. Run the adversary $\mathcal{A}$ in the CPA game, while emulating the "challenger" in the CPA security game, as follows:
   (a) Every time $\mathcal{A}$ requests an encryption of a message $m_i \in \{0,1\}$, sample a random $r_i \leftarrow \{0,1\}^\lambda$ and return the ciphertext $(r_i, O(r_i) \oplus m_i)$.
   (b) When $\mathcal{A}$ sends the two challenge messages $m_0, m_1 \in \{0,1\}$, choose a random $b \leftarrow \{0,1\}$ and $r \leftarrow \{0,1\}^\lambda$, and send $c = (r, \mathcal{O}(r) \oplus m_b)$.

2. Denote the output of $\mathcal{A}$ by b'.

3. If $b' = b$ then guess pseudorandom (say output 1) and otherwise guess random (say output 0).

By definition of $\mathcal{B}$, and by our assumption that $\mathcal{A}$ wins the security game with probability $\frac{1}{2} + \epsilon(\lambda)$, we conclude that

$$\Pr[\mathcal{B}^{F_\lambda(k,\cdot)}(1^\lambda) = 1] \geq \frac{1}{2} + \epsilon(\lambda).$$

On the other hand, we argue that there exists a negligible function μ such that for every $\lambda \in \mathbb{N}$

$$\Pr[\mathcal{B}^{R_\lambda}(1^\lambda) = 1] = \frac{1}{2} + \mu(\lambda).$$

For the latter, denote by E the event that the challenge ciphertext (r, c_b) encrypting m_b satisfies that r is different from all the random strings $\{r_i\}$ used to encrypt the messages $\{m_i\}$ chosen by $\mathcal{A}$. Then

$$\begin{aligned}
&\Pr[\mathcal{B}^{R_\lambda}(1^\lambda) = 1] = \\
&\Pr[\mathcal{B}^{R_\lambda}(1^\lambda) = 1 |\ E] \cdot \Pr[E] + \Pr[\mathcal{B}^{R_\lambda}(1^\lambda) = 1 |\ \neg E] \cdot \Pr[\neg E] \leq \\
&\Pr[\mathcal{B}^{R_\lambda}(1^\lambda) = 1 |\ E] \cdot \Pr[E] + \mu(\lambda) \leq \\
&\Pr[\mathcal{B}^{R_\lambda}(1^\lambda) = 1 |\ E] + \mu(\lambda) = \\
&\frac{1}{2} + \mu(\lambda).
\end{aligned}$$

□

Thus, pseudorandom functions give us CPA-secure secret-key encryption. Our next goal is to construct pseudorandom functions from the primitive we already have: pseudorandom generators.

PRF *Construction*

We next show how to construct a PRF from any PRG. This is a beautiful construction proposed by Goldreich, Goldwasser and Micali [1].

Note that PRFs and PRGs are very similar objects. They both take a short random seed $k \leftarrow \{0,1\}^\lambda$ and generate many pseudorandom bits out of it. The difference is that a PRG expands λ bits to $n(\lambda) \leq \text{poly}(\lambda)$ bits whereas a PRF can in principle expand by 2^λ bits:

$$\left(F(k, 0^\lambda), \ldots, F(k, 1^\lambda)\right).$$

One can think of a PRF as a PRG with random access.

The GGM Construction. Recall that last week we showed how to increase the stretch of a PRG

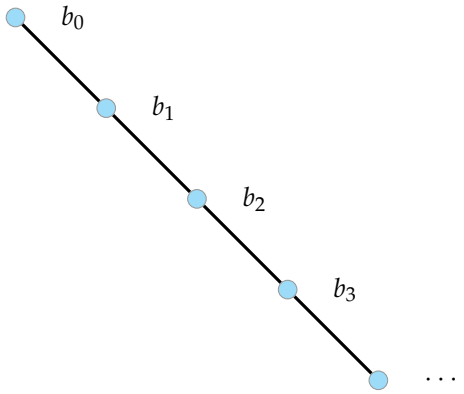
$$G : \{0,1\}^\lambda \to \{0,1\}^{\lambda+1}$$

into a PRG with arbitrary stretch

$$G^k : \{0,1\}^\lambda \to \{0,1\}^{\lambda+k}$$

The problem is that accessing the $\lambda + i$th bit of $G^k(x)$ takes time $\geq i$. The reason is that this construction increases expansion via a line. .

Recall from lecture 3 that we write $G(s_i) = s_{i+1} \circ b_i$, where s_{i+1} is the new λ-bit state and b_i is the output bit



The idea of GGM is to use a tree structure as opposed to a line structure to have a more efficient random access. By the stretch-extension theorem from the previous lecture, we may assume that we have a length-doubling PRG $G : \{0,1\}^\lambda \to \{0,1\}^{2\lambda}$. Denote by

$$G(x) = (G_0(x), G_1(x))$$

where for every $b \in \{0,1\}$

$$G_b : \{0,1\}^\lambda \to \{0,1\}^\lambda.$$

Similarly, for every $b_1, b_2 \in \{0,1\}$ let

$$G_{b_1,b_2} : \{0,1\}^\lambda \to \{0,1\}^\lambda$$

defined by

$$G_{b_1,b_2}(x) = G_{b_2}(G_{b_1}(x)).$$

More generally, for every $b_1, \ldots, b_i \in \{0,1\}$ let

$$G_{b_1,\ldots,b_i} : \{0,1\}^\lambda \to \{0,1\}^\lambda$$

defined by

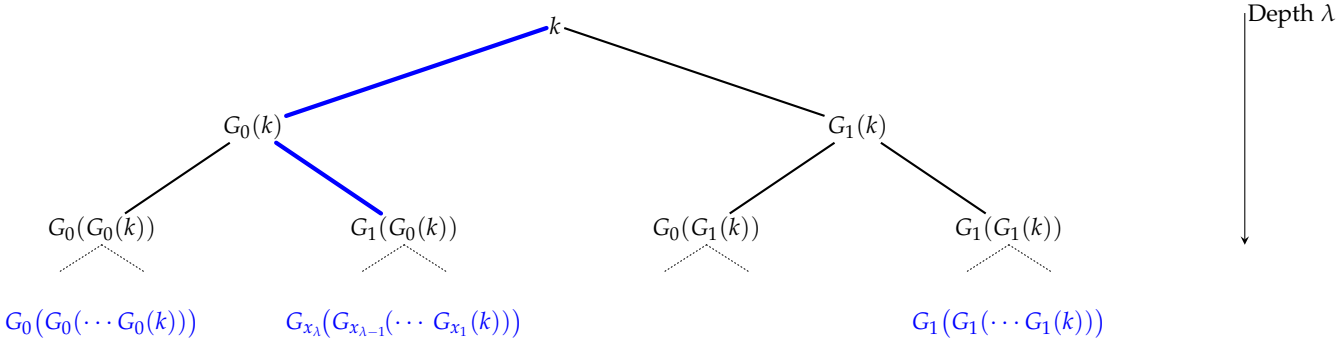
$$G_{b_1,\ldots,b_i}(x) = G_{b_i}(\ldots(G_{b_1}(x))\ldots).$$

For $x = x_1 \cdots x_\lambda \in \{0,1\}^\lambda$, define

$$F(k,x) = G_{x_1,\ldots,x_\lambda}(k) = G_{x_\lambda}\left(\cdots G_{x_2}(G_{x_1}(k))\cdots\right).$$

Goldreich–Goldwasser–Micali PRF

Construction: Let $G(s) = G_0(s) \,\|\, G_1(s)$ where $G_0(s)$ and $G_1(s)$ are both λ bits.



Each path/leaf labeled by $x \in \{0,1\}^\lambda$ corresponds to $F(k,x)$.

Theorem 6. *The GGM construction is a* PRF.

The proof of this theorem makes use of the following lemma.

Lemma 7. *If $G : \{0,1\}^\lambda \to \{0,1\}^{n(\lambda)}$ is a* PRG *then for every polynomial $\ell : \mathbb{N} \to \mathbb{N}$ it holds that*

$$\{G(k_1),\ldots,G(k_{\ell(\lambda)})\}_{\lambda\in\mathbb{N}} \approx \{U_{n(\lambda)\cdot\ell(\lambda)}\}_{\lambda\in\mathbb{N}}$$

where $k_1,\ldots,k_{\ell(\lambda)} \leftarrow \{0,1\}^\lambda$.

The proof of this lemma follows from a standard hybrid argument.

Proof of Lemma 7. The proof is by a hybrid argument. Suppose for contradiction that there exists a poly-size $\mathcal{A}$ and a non-negligible function ϵ such that for every $\lambda \in \mathbb{N}$

$$|\Pr[\mathcal{A}(G(k_1),\ldots,G(k_{\ell(\lambda)}) = 1] - \Pr[\mathcal{A}(U_{n(\lambda)\cdot\ell(\lambda))=1}| \geq \epsilon(\lambda).$$

For every $\lambda \in \mathbb{N}$, and for every $i \in \{0,1,\ldots,\ell(\lambda)\}$, denote by

$$H_{\lambda,i} = (G(k_1),\ldots,G(k_i),U_{n(\lambda)\cdot(\ell(\lambda)-i)}).$$

Then for every $\lambda \in \mathbb{N}$,

$$|\Pr[\mathcal{A}(G(k_1),\ldots,G(k_{\ell(\lambda)}) = 1] - \Pr[\mathcal{A}(U_{n(\lambda)\cdot\ell(\lambda)}) = 1]| =$$
$$|\Pr[\mathcal{A}(H_{\lambda,\ell}) = 1] - \Pr[\mathcal{A}(H_{\lambda,0}) = 1] =$$
$$|\sum_{i=1}^{\ell} \Pr[\mathcal{A}(H_{\lambda,i}) = 1] - \Pr[\mathcal{A}(H_{\lambda,i-1}) = 1]| \leq$$
$$\sum_{i=1}^{\ell} |\Pr[\mathcal{A}(H_i) = 1] - \Pr[\mathcal{A}(H_{i-1}) = 1]|$$

Thus, for every $\lambda \in \mathbb{N}$ there exists $i \in [\ell(\lambda)]$ such that

$$|\Pr[\mathcal{A}(H_i) = 1] - \Pr[\mathcal{A}(H_{i-1}) = 1]| \geq \frac{\epsilon(\lambda)}{\ell(\lambda)}$$

We use this to break the security of the PRG G. Specifically, we construct a poly-size $\mathcal{B}$ that is given as input x, which is either distributed as $G(U_\lambda)$ or as $U_{n(\lambda)}$, and proceeds as follows:

1. Sample $k_1, \ldots, k_{i-1} \leftarrow \{0,1\}^\lambda$.
2. Sample independently $u_{i+1}, \ldots, u_{\ell(\lambda)} \xleftarrow{R} \{0,1\}^{n(\lambda)}$.
3. Output $\mathcal{A}\left(G(k_1), \ldots, G(k_{i-1}), x, u_{i+1}, \ldots, u_{\ell(\lambda)}\right)$.

If $x = G(U_\lambda)$, then the input to $\mathcal{A}$ is distributed according to $H_{\lambda,i}$, whereas if $x = U_{n(\lambda)}$, it is distributed according to $H_{\lambda,i-1}$. Thus $\mathcal{B}$ distinguishes $G(U_\lambda)$ from $U_{n(\lambda)}$ with advantage at least

$$\frac{\epsilon(\lambda)}{\ell(\lambda)},$$

which is non-negligible since ℓ is polynomial, contradicting the pseudorandomness of G. □

We next show how to use Lemma 7 to prove Theorem 6.

Proof of Theorem 6. The proof is via a hybrid argument on the layers of the tree. Suppose there exists a poly-size $\mathcal{A}$ and a non-negligible ϵ such that for every $\lambda \in \mathbb{N}$

$$|\Pr[\mathcal{A}^{F(k,\cdot)}(1^\lambda) = 1] - \Pr[\mathcal{A}^{R_\lambda}(1^\lambda) = 1]| \geq \epsilon$$

In what follows, we denote by $F_0 = F(k, \cdot)$. Let F_1 be the function that is similar to F_0 except that it replaces the first layer of the tree (from the root) with a uniform layer. Namely, $(G_0(k), G_1(k))$ is replaced with truly random (k_0, k_1). By the security of the PRG there exists a negligible function μ_1

$$|\Pr[\mathcal{A}^{F_0(k,\cdot)}(1^\lambda) = 1] - \Pr[\mathcal{A}^{F_1(k,\cdot)}(1^\lambda) = 1]| \leq \mu_1.$$

More generally, let F_i be the function that replaces the i'th layer of the tree (from the root) with truly random values. Note that $F_\lambda = R(\cdot)$. By a hybrid argument there exists $i \in [\lambda]$ such that for every $\lambda \in \mathbb{N}$

$$|\Pr[\mathcal{A}^{F_{i-1}}(1^\lambda) = 1] - \Pr[\mathcal{A}^{F_i}(1^\lambda) = 1]| \geq \frac{\epsilon}{\lambda}$$

We argue that this contradicts Lemma 7.

Note that the number of nodes in the ith layer is 2^i, which may be super-polynomial. Thus we cannot contradict Lemma 7 with $\ell = 2^i$

(indeed, this lemma is false with super-polynomial ℓ!). Instead, rather than assigning a string to each node in the i'th layer, we will only assign strings to nodes that are queried by $\mathcal{A}$.

Specifically, let $q = q(\lambda)$ be an upper bound on the number of oracle calls that $\mathcal{A}$ makes. We construct a poly-size adversary $\mathcal{B}$ such that for every $\lambda \in \mathbb{N}$,

$$\left|\Pr[\mathcal{B}(G(k_1),\ldots,G(k_q))=1] - \Pr[\mathcal{B}(U_{2\lambda q(\lambda)})=1]\right| \geq \frac{\epsilon}{\lambda}. \qquad (1)$$

$\mathcal{B}$ on input $(x_{1,0}, x_{1,1}, \ldots, x_{q,0}, x_{q,1})$ emulates $\mathcal{A}(1^\lambda)$ as follows:

1. Upon receiving the j'th oracle call from $\mathcal{A}$ do the following:
   (a) Denote the oracle query by $r \in \{0,1\}^\lambda$.
   (b) Denote by $r_{[i-1]} \in \{0,1\}^{i-1}$ the first $i-1$ bits of r. Similarly denote by $r_{[i]} \in \{0,1\}^i$ the first i bits of r.
   (c) If $k_{r_{[i-1]},0}$ and $k_{r_{[i-1]},1}$ have not been previously defined, then set $k_{r_{[i-1]},0} = x_{j,0}$ and $k_{r_{[i-1]},1} = x_{j,1}$.
   (d) Compute $y = G_{r_\lambda}\left(G_{r_{\lambda-1}}\left(\ldots G_{r_{i+1}}\left(k_{r_{[i]}}\right)\ldots\right)\right)$ and return y to $\mathcal{A}$.
2. When $\mathcal{A}$ terminates, output whatever $\mathcal{A}$ outputs.

It is easy to see that if $(x_{1,0}, x_{1,1}, \ldots, x_{q,0}, x_{q,1})$ is uniformly distributed in $\{0,1\}^{\lambda\cdot 2q}$ then

$$\Pr[\mathcal{B}(x_{1,0}, x_{1,1}, \ldots, x_{q,0}, x_{q,1})=1] = \Pr[\mathcal{A}^{F_i}(1^\lambda)=1].$$

On the other hand, if $(x_{1,0}, x_{1,1}, \ldots, x_{q,0}, x_{q,1})$ is distributed as $(G(k_1),\ldots,G(k_q))$ for random $k_1,\ldots,k_q \leftarrow \{0,1\}^\lambda$, then

$$\Pr[\mathcal{B}(x_{1,0}, x_{1,1}, \ldots, x_{q,0}, x_{q,1})=1] = \Pr[\mathcal{A}^{F_{i-1}}(1^\lambda)=1].$$

Thus, by Equation (1), we conclude that

$$|\Pr[\mathcal{B}(G(k_1),\ldots,G(k_q))=1] - \Pr[\mathcal{B}(U_{\lambda\cdot 2q})=1]| \geq \frac{\epsilon}{\lambda}$$

contradicting Lemma 7. □

References

[1] Oded Goldreich, Shafi Goldwasser, and Silvio Micali. How to construct random functions. *Journal of the ACM*, 33(4):792–807, 1986.