1. Definitions and Foundations

1.1. **Pseudorandom generators.** Modern cryptography can be viewed as the study of hardness. In a typical cryptographic application, there is some task that is "easy" or "efficient" for the honest user, but is "hard" or "difficult" for the illegitimate user. For example, take the case of a (secret-key) encryption scheme. It should be easy for the honest users (those who possess the secret key) to encrypt and decrypt messages. However, it should be difficult for an illegitimate user (those who do not possess the secret key) to learn information about a message given only the ciphertext. In this lecture, we will introduce several (closely-related) notions that underpin symmetric cryptography: one-way functions (OWFs), pseudorandom generators (PRGs), pseudorandom functions (PRFs), pseudorandom permutations (PRPs). We begin with a discussion of pseudorandom generators. Intuitively, a PRG takes as input a short seed (random) and expands it into a longer random-looking string. PRGs are also commonly referred to as stream ciphers. We formalize this notion in the following definition:

Definition 1.1 (Pseudorandom Generator)**.** Let $\lambda \in \mathbb{N}$ be a security parameter. A pseudorandom generator (PRG) is an efficiently computable function $G_\lambda : \{0,1\}^\lambda \to \{0,1\}^{\ell(\lambda)}$ where $\ell(\lambda)$ is a fixed polynomial in λ and $\ell(\lambda) > \lambda$. A PRG is *secure* if for all efficient adversaries $\mathcal{A}$, there is a negligible function $\varepsilon(\lambda) = \text{negl}(\lambda)$ such that

$$\text{PRG Adv}[A,G] = \left|\Pr[s \xleftarrow{R} \{0,1\}^\lambda : \mathcal{A}(G_\lambda(s)) = 1] - \Pr[z \xleftarrow{R} \{0,1\}^{\ell(\lambda)} : \mathcal{A}(z) = 1]\right| = \varepsilon(\lambda).$$

(World 0) (World 1)

Let us interpret this definition:

- We write $x \xleftarrow{R} \{0,1\}^\lambda$ to denote that the input x is sampled uniformly at random from $\{0,1\}^\lambda$. More generally, for a finite set S, we write $x \xleftarrow{R} S$ to denote a uniformly random draw from S.
- We say that a function f is "efficiently computable" (or more simply, "efficient") if it can be computed by a *probabilistic polynomial-time* algorithm (in the length of its input).
- We say that a function ε is "negligible" in a parameter λ, denoted $\text{negl}(\lambda)$, if $\varepsilon(\lambda) = o(1/\lambda^c)$ for all constants $c \in \mathbb{N}$. In other words, ε is smaller than *any* inverse polynomial function of λ. We will often refer to λ as a *security parameter*.
- We often refer to ℓ as the expansion factor of the PRG.
- Intuitively, a PRG is secure if no efficient adversary can distinguish the output of a PRG from a uniformly random string of the same length, except with vanishingly small probability (as a function of the security parameter λ). Note that restricting the adversary to be efficient is critical; otherwise, the definition is *impossible* to satisfy.
- A PRG is more accurately viewed as a *family* of functions, where there is one function G_λ for each security parameter $\lambda \in \mathbb{N}$. We sometimes denote this by $G = \{G_\lambda\}_{\lambda \in \mathbb{N}}$. For notational convenience, we will drop the explicit dependence on the security parameter λ when it is not essential.

Another way to state the PRG security definition is through the lens of computational indistinguishability. We begin by defining what it means for two distributions to be computationally indistinguishable.

Definition 1.2. Let $\lambda \in \mathbb{N}$ be a security parameter. Let $\mathcal{D}_1 = \{\mathcal{D}_{1,\lambda}\}_{\lambda \in \mathbb{N}}$ and $\mathcal{D}_2 = \{\mathcal{D}_{2,\lambda}\}_{\lambda \in \mathbb{N}}$ be two ensembles of distributions. We say that $\mathcal{D}_1$ and $\mathcal{D}_2$ are computationally indistinguishable, denoted $\mathcal{D}_1 \overset{c}{\approx} \mathcal{D}_2$ if for all efficient adversaries:

$$\left|\Pr[x \leftarrow \mathcal{D}_1 : \mathcal{A}(1^\lambda, x)] - \Pr[x \leftarrow \mathcal{D}_2 : \mathcal{A}(1^\lambda, x)]\right| = \text{negl}(\lambda).$$

As before, for notational conciseness in the sequel, we will omit the explicit dependence on the security parameter λ unless it is critical for understanding.

Intuitively, two distributions $\mathcal{D}_1$ and $\mathcal{D}_2$ are computationally indistinguishable if no efficient adversary can tell whether it received a sample from the first distribution $\mathcal{D}_1$ or the second distribution $\mathcal{D}_2$ (except with negligibly small probability as a function of the security parameter). Note that in the above definition, the adversary's input includes the *unary* encoding of the security parameter λ. This is to enable the distinguisher to run in time $\text{poly}(\lambda)$. Otherwise, if the outputs of the distributions $\mathcal{D}_1$ and $\mathcal{D}_2$ are short (e.g., $\log \lambda$-bits long), then we are constraining the adversary to run in polylogarithmic time. We can now reformulate the definition of a secure PRG in the language of computational indistinguishability. In particular, we say that a

function G is a secure PRG (with expansion factor $\ell(\lambda)$) if the following holds:

$$\left\{s \xleftarrow{R} \{0,1\}^\lambda : G(s)\right\} \overset{c}{\approx} \left\{z \xleftarrow{R} \{0,1\}^{\ell(\lambda)} : z\right\}.$$

Remark 1.3 (One-Time Encryption using PRGs)**.** It is very straightforward to build a (symmetric) one-time encryption scheme using a PRG. Suppose we want to encrypt $\ell = \ell(\lambda)$-bit messages. The encryption scheme relies on a PRG with expansion factor $\ell(\lambda)$ and the key is a PRG seed $s \in \{0,1\}^\lambda$. The encryption of a message $m \in \{0,1\}^{\ell(\lambda)}$ is then $m \oplus G(s)$. Security of the PRG means that $G(s)$ looks like a uniformly random string, in which case, the encryption scheme is precisely the *one-time pad*, which provides *perfect secrecy*. The limitation of course is that this encryption scheme is *one-time*.

1.2. **The Blum-Micali PRG.** A natural question to ask is whether secure PRGs exist. Unfortunately, answering this question in the affirmative also shows $\mathsf{P} \neq \mathsf{NP}$. It turns out that proving the existence of PRGs requires proving a stronger result than $\mathsf{P} \neq \mathsf{NP}$—in particular, that *one-way functions* exist.[1] In this class, we operate (at the minimum) under the assumption that one-way functions exist. Using one-way permutations (a slight strengthening of one-way functions), it is straightforward to construct a PRG that extends the input by a single bit (the same is possible starting from a one-way function, but with a more complicated construction).

Suppose now that we have a PRG G that extends the seed by a single bit, that is, $G_\lambda \colon \{0,1\}^\lambda \to \{0,1\}^{\lambda+1}$. A natural question is whether we can use G to build a *many-bit* PRG? Given a secure PRG G that expands the seed by a single bit, it is possible to construct a PRG with an arbitrary $\mathrm{poly}(\lambda)$-expansion factor using the Blum-Micali construction:

Construction 1.4 (Blum-Micali)**.** Let $G_\lambda \colon \{0,1\}^\lambda \to \{0,1\}^{\lambda+1}$ be a secure PRG that extends the input seed by a single bit. Fix any polynomial $\ell(\lambda) = \mathrm{poly}(\lambda)$. We can use G_λ to construct a PRG H_λ with expansion $\ell(\lambda)$ as follows. On input a seed $s_0 \in \{0,1\}^\lambda$, H_λ computes $(s_i, b_i) \leftarrow G(s_{i-1})$ for $i \in [\ell]$. Finally, H_λ outputs the bits $(b_1, \ldots, b_\ell) \in \{0,1\}^\ell$.

Theorem 1.5. *Fix any polynomial $\ell = \ell(\lambda)$. If $G = \{G_\lambda\}_{\lambda \in \mathbb{N}}$ is a secure PRG that expands the seed by a single bit, then $H = \{H_\lambda\}_{\lambda \in \mathbb{N}}$ is a secure PRG with expansion factor $\ell(\lambda)$.*

To prove Theorem 1.5, we introduce a technique called a *hybrid argument*. The basic idea of a hybrid argument is as follows. Suppose we have two (ensembles of) distributions $\mathcal{D}_1$ and $\mathcal{D}_2$, and we want to show that $\mathcal{D}_1 \overset{c}{\approx} \mathcal{D}_2$. Suppose moreover that we can identify an "intermediate" distribution $\mathcal{D}'$ and we are able to show that $\mathcal{D}_1 \overset{c}{\approx} \mathcal{D}'$ and $\mathcal{D}' \overset{c}{\approx} \mathcal{D}_2$. We claim that this implies $\mathcal{D}_1 \overset{c}{\approx} \mathcal{D}_2$. To see this, take any efficient algorithm $\mathcal{A}$. We need to show that

$$\left|\Pr[x \leftarrow \mathcal{D}_1 : \mathcal{A}(1^\lambda, x) = 1] - \Pr[x \leftarrow \mathcal{D}_2 : \mathcal{A}(1^\lambda, x) = 1]\right| = \mathrm{negl}(\lambda).$$

We write $p_1 = \Pr[x \leftarrow \mathcal{D}_1 : \mathcal{A}(1^\lambda, x) = 1]$ to denote the probability that $\mathcal{A}$ outputs 1 when given a sample from $\mathcal{D}_1$. We define p_2 and p' accordingly (for distributions $\mathcal{D}_2$ and $\mathcal{D}'$, respectively). We can now appeal to the triangle inequality to argue

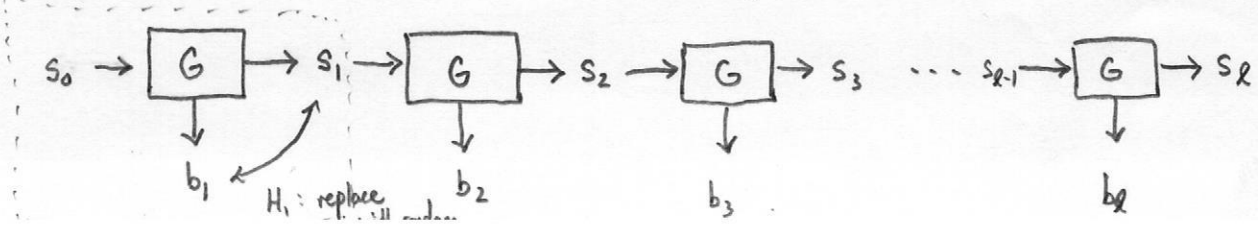
$$|p_1 - p_2| = |p_1 - p' + p' - p_2| \leq |p_1 - p'| + |p_2 - p'| = \mathrm{negl}(\lambda),$$

using the fact that $\mathcal{D}_1 \overset{c}{\approx} \mathcal{D}'$ and $\mathcal{D}' \overset{c}{\approx} \mathcal{D}_2$. More generally, if we have $n = \mathrm{poly}(\lambda)$ intermediate distributions $\mathcal{D}_0, \mathcal{D}_1, \ldots, \mathcal{D}_n$ such that $\mathcal{D}_{i-1} \overset{c}{\approx} \mathcal{D}_i$ for all $i \in [n]$, then $\mathcal{D}_0 \overset{c}{\approx} \mathcal{D}_n$. Note that this only applies if n is a *polynomial* function in λ. With this background, we can now prove the security of the Blum-Micali distribution.

Proof of Theorem 1.5. We define a sequence of ℓ hybrid distributions $\mathsf{Hyb}_0, \ldots, \mathsf{Hyb}_\ell$, where distribution Hyb_i is the following distribution:

(1) Sample $b_1, \ldots, b_i \xleftarrow{R} \{0,1\}$ and $s_i \xleftarrow{R} \{0,1\}^\lambda$.
(2) For $\ell \geq j > i$, let $(s_j, b_j) \leftarrow G_\lambda(s_{j-1})$.
(3) Output $b_1, \ldots, b_\ell$.

[1] It is entirely possible that $\mathsf{P} \neq \mathsf{NP}$, and yet, one-way functions, and correspondingly, cryptography does not exist.



By construction, Hyb_0 is the output of the Blum-Micali construction, while Hyb_ℓ is the distribution where the output is a truly random ℓ-bit string. Each pair of intermediate distributions is computationally indistinguishable (by PRG security of G). Since $\ell = \text{poly}(\lambda)$, we conclude that $\mathsf{Hyb}_0 \overset{c}{\approx} \mathsf{Hyb}_\ell$, and so H is a secure PRG. □

1.3. **Pseudorandom functions.** In Remark 1.3, we argued that PRGs (with sufficient expansion factor) can be used to derive a secure one-time encryption scheme. The natural question is how to build many-time encryption schemes (i.e., CPA-secure symmetric encryption). As you may recall, a useful cryptographic primitive for building many-time encryption schemes (as well as many other cryptographic primitives) is the notion of a pseudorandom function (PRF). We first recall the definition of a PRF.

Definition 1.6 (Pseudorandom Function)**.** A pseudorandom function $F\colon \mathcal{K} \times \mathcal{X} \to \mathcal{Y}$ on key-space $\mathcal{K}$, domain $\mathcal{X}$, and range $\mathcal{Y}$ is an efficiently-computable function $F\colon \mathcal{K} \times \mathcal{X} \to \mathcal{Y}$ such that for all efficient adversaries $\mathcal{A}$:

$$\mathsf{PRFAdv}[\mathcal{A}, F] = \left|\Pr[k \xleftarrow{R} \mathcal{K} : \mathcal{A}^{F(k,\cdot)}(1^\lambda) = 1] - \Pr[f \xleftarrow{R} \mathsf{Funs}[\mathcal{X}, \mathcal{Y}] : \mathcal{A}^{f(\cdot)}(1^\lambda) = 1]\right| = \text{negl}(\lambda).$$

In particular, a PRF is secure if no efficient adversary can distinguish the outputs of the PRF (on adaptively-chosen queries) from that of a truly random function. It is easy to see that a PRF can be used to build CPA-secure encryption schemes (e.g., using nonce-based counter mode).

1.4. **From PRGs to PRFs.** Having defined the notion of a PRF, the natural next question is whether PRFs exist. A seminal result by Goldreich, Goldwasser, and Micali showed how PRFs can be constructed from any *length-doubling* PRG (which in turn, can be instantiated from any one-way function). The Goldreich-Goldwasser-Micali (GGM) construction is a tree-based construction. Let $G : \{0,1\}^\lambda \to \{0,1\}^{2\lambda}$ be a length-doubling PRG. On input a seed $s \in \{0,1\}^\lambda$, $G(s)$ outputs a string $(s_0, s_1) \in \{0,1\}^{2\lambda}$. For notational convenience, we will write $G(s) \to (s_0, s_1) = (G_0(s), G_1(s))$. The GGM construction of a PRF over a domain $\{0,1\}^n$ operates as follows. On input an input $x = x_1 \cdots x_n \in \{0,1\}^n$ and a key $k \in \{0,1\}^\lambda$, the PRF $F\colon \{0,1\}^\lambda \times \{0,1\}^n \to \{0,1\}^\lambda$ is defined as $F(k,x) = G_{x_n}(G_{x_{n-1}}(\cdots G_{x_1}(s)\cdots))$. Security of this construction reduces to the security of the PRG and can be formally shown using a hybrid argument.

The GGM construction can be used to build a PRF on any domain $\{0,1\}^n$ where $n = \text{poly}(\lambda)$. The range of the above construction is always $\{0,1\}^\lambda$. How do we extend to an arbitrary range? If we want the PRF output to be $\{0,1\}^\ell$ where $\ell(\lambda) < \lambda$, then we can simply truncate the PRF (why does this work?). On the other hand, if $\ell(\lambda) > \lambda$, we can use $F(k,x)$ as the input to a PRG that outputs $\ell(\lambda)$ bits (security of this construction again follows by a simple hybrid argument).

1.5. **From PRFs to PRGs.** Once we have a PRF, it is very straightforward to construct a PRG by simply evaluating the PRF in "counter mode." Suppose we have a one-bit PRF $F\colon \{0,1\}^\lambda \times \{0,1\}^n \to \{0,1\}$. We construct a PRG $G\colon \{0,1\}^\lambda \to \{0,1\}^{\ell(\lambda)}$ as follows: $G(k) = F(k,0) \| F(k,1) \| \cdots \| F(k,\ell)$. This construction works as long as $n > \log \ell$. Security of this construction directly reduces to the security of the underlying PRF (no hybrid needed).

PRF Security :

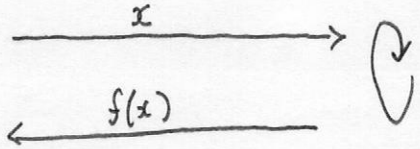
adversary

challenger

World 0: $k \leftarrow \mathcal{K}$

$f \leftarrow \mathsf{PRF}(k, \cdot)$

World 1: $f \xleftarrow{R} \mathsf{Fun}[\mathcal{X}, \mathcal{Y}]$



$L \in \{0,1\}$