Lecture 4: Crypto Basics 1

Lecturer: Shafi Goldwasser **Date:** February 12, 2026
Scribe: Hara Moraitaki, Sky Pulling

Overview

This lecture introduces the foundational cryptographic primitives that underpin both modern cryptography and, as we shall see throughout the course, many results at the intersection of cryptography and machine learning. We begin with Shannon's information-theoretic notion of perfect secrecy, encounter its inherent limitations, and then transition to the computational framework of modern cryptography. We develop the theory of pseudorandom generators (PRGs) from one-way functions and hard-core predicates, culminating in a construction that overcomes Shannon's impossibility result by restricting to computationally bounded adversaries.

1. Secret Communication and Encryption Schemes

The fundamental problem of cryptography is enabling two parties, for the purposes of illustration traditionally called Alice and Bob, to communicate securely over an insecure channel in the presence of an adversary (Vincent/Eve). We formalize the tools for doing so via encryption schemes.

Remark 1.1 (Notational Convention). Throughout these notes, capital letters (M, K, C) denote random variables and their associated probability distributions. Lower-case letters (m, k, c) denote particular realizations. When a random variable appears in a probability statement, the probability is taken over the random choices of that variable.

Definition 1.1 (Encryption Scheme)**.** An *encryption scheme* is a triple of probabilistic algorithms $(\mathsf{Gen}, \mathsf{Enc}, \mathsf{Dec})$ defined over a message space $\mathcal{M}$, a key space $\mathcal{K}$, and a ciphertext space C:

1. **Key generation:** $\mathsf{Gen}(1^n)$ is a probabilistic algorithm that outputs a secret key K of length n, where input n is the security parameter. We want everything to be polynomial time in the security parameter.
2. **Encryption:** $\mathsf{Enc}(K, m)$ takes a key and a plaintext message $m \in \mathcal{M}$ and outputs a ciphertext $c \in C$.
3. **Decryption:** $\mathsf{Dec}(K, c)$ takes a key and a ciphertext and outputs a plaintext m'.

The scheme must satisfy correctness: for all $m \in \mathcal{M}$,

$$\mathsf{Dec}(K, \mathsf{Enc}(K, m)) = m.$$

Note: Sometimes later in the course, we loosen our definition of correctness to demand that

$\mathsf{Dec}(K, \mathsf{Enc}(K, m)) = m$ only be true with high probability, but for now we require the decryption to always work.

We associate probability spaces with each component: $\mathcal{K}$ is the key space (with $\Pr[K = k]$ determined by Gen), $\mathcal{M}$ is the message space (for example: English, all binary strings of a certain length), and C is the ciphertext space, with $\Pr[C = c]$ determined jointly by Gen, the message M, and the coins of Enc (the internal random bits used by the probabilistic encryption algorithm). The message M is itself treated as a random variable drawn from some distribution over $\mathcal{M}$. Ideally, a cryptographic scheme should work for all message spaces.

How is security defined? Security is always defined with respect to an adversary. Who is the adversary and what powers do they have?

Kerckhoffs's Principle

A cryptographic system should be secure even if everything about the system (the algorithms Gen, Enc, Dec) is known to the adversary, except for the secret key and the randomness used by legitimate parties. This principle, articulated by Auguste Kerckhoffs in the 19th century, is a cornerstone of modern cryptographic design.

Ciphertext-Only Attack Model. In the most basic adversarial setting, the adversary can observe ciphertexts c transmitted over an insecure channel but has no other access (e.g., no chosen-plaintext or chosen-ciphertext queries). All definitions of security in this lecture assume at least this level of adversarial access.

2. Shannon's Perfect Secrecy

Claude Shannon's 1949 paper [17] established the first rigorous mathematical framework for secrecy. In Shannon's model, the all-powerful adversary is *computationally unbounded*; the security analysis is purely information-theoretic. For now we will only discuss *passive* adversaries, but there also exist *active* adversaries, who may distort the ciphertext.

2.1. Definition: Perfect Secrecy

Definition 2.1 (Perfect Secrecy / Shannon Secrecy)**.** An encryption scheme $(\mathsf{Gen}, \mathsf{Enc}, \mathsf{Dec})$ satisfies *perfect secrecy* (Shannon secrecy) if for every probability distribution over the message space $\mathcal{M}$, every ciphertext $c \in C$, and every message $m \in \mathcal{M}$:

$$\Pr[M = m] = \Pr[M = m \mid \mathsf{Enc}(K, M) = c].$$

That is, the *a priori* probability of any message equals the *a posteriori* probability after observing the ciphertext.

For perfect secrecy, the ciphertext reveals no information whatsoever about the plaintext, regardless of the adversary's computational power. Note that the adversary does not appear explicitly in the definition but is implicitly present as the entity computing posterior probabilities.

2.2. Perfect Indistinguishability

An equivalent formulation frames security as a game between two "worlds."

Definition 2.2 (Perfect Indistinguishability)**.** An encryption scheme $(\mathsf{Gen}, \mathsf{Enc}, \mathsf{Dec})$ satisfies *perfect indistinguishability* if for every probability distribution over $\mathcal{M}$, for all pairs of messages $m_0, m_1 \in \mathcal{M}$, and for all $c \in C$:

$$\Pr\left[\mathsf{Enc}(K, m_0) = c\right] = \Pr\left[\mathsf{Enc}(K, m_1) = c\right],$$

where the probability is over $K \leftarrow \mathsf{Gen}(1^n)$ and the coins of Enc.

Equivalently, for any unbounded adversary Eve and any $m_0, m_1 \in \mathcal{M}$:

$$\Pr\left[\mathsf{Eve}(c) = b\right] = \tfrac{1}{2},$$

where $K \leftarrow \mathsf{Gen}(1^n)$, $b \leftarrow \{0,1\}$, and $c = \mathsf{Enc}(K, m_b)$. Eve receives the ciphertext c but cannot distinguish which of the two messages was encrypted with probability better than random guessing.

Theorem 2.1 (Equivalence)**.** *An encryption scheme satisfies perfect secrecy if and only if it satisfies perfect indistinguishability.*

Proof. We prove both directions using Bayes' theorem.

($\Rightarrow$) Perfect secrecy $\Rightarrow$ perfect indistinguishability.
Assume $\Pr[M = m] = \Pr[M = m \mid \mathsf{Enc}(K, M) = c]$ for all m, c. We want to show that $\Pr[\mathsf{Enc}(K, m_0) = c] = \Pr[\mathsf{Enc}(K, m_1) = c]$ for all m_0, m_1, c. As a reminder, the eavesdropper is assumed to know m_0, m_1, and c, but not the key K, and is trying to distinguish which message the ciphertext was encoded from.

By Bayes' theorem, for any m with $\Pr[M = m] > 0$:

$$\Pr\left[\mathsf{Enc}(K, M) = c \mid M = m\right] = \frac{\Pr\left[M = m \mid \mathsf{Enc}(K, M) = c\right] \cdot \Pr\left[\mathsf{Enc}(K, M) = c\right]}{\Pr\left[M = m\right]}.$$

Since perfect secrecy gives $\Pr[M = m \mid \mathsf{Enc}(K, M) = c] = \Pr[M = m]$, this simplifies to:

$$\Pr\left[\mathsf{Enc}(K, m) = c\right] = \Pr\left[\mathsf{Enc}(K, M) = c\right],$$

which is independent of m. In particular, $\Pr[\mathsf{Enc}(K, m_0) = c] = \Pr[\mathsf{Enc}(K, m_1) = c]$ for all m_0, m_1.

($\Leftarrow$) Perfect indistinguishability $\Rightarrow$ perfect secrecy.
Assume $\Pr[\mathsf{Enc}(K, m_0) = c] = \Pr[\mathsf{Enc}(K, m_1) = c]$ for all m_0, m_1, c. Let $p_c = \Pr[\mathsf{Enc}(K, m) = c]$ denote this common value (independent of m). By Bayes' theorem, for any m with $\Pr[M = m] > 0$:

$$\begin{aligned}\Pr\left[M = m \mid \mathsf{Enc}(K, M) = c\right] &= \frac{\Pr\left[\mathsf{Enc}(K, M) = c \mid M = m\right] \cdot \Pr\left[M = m\right]}{\Pr\left[\mathsf{Enc}(K, M) = c\right]} \\ &= \frac{p_c \cdot \Pr\left[M = m\right]}{\sum_{m'} p_c \cdot \Pr\left[M = m'\right]} = \frac{p_c \cdot \Pr\left[M = m\right]}{p_c} = \Pr\left[M = m\right].\end{aligned}$$

Thus the a posteriori distribution equals the a priori distribution, which is precisely perfect secrecy. □

2.3. The One-Time Pad

Shannon showed that perfect secrecy is achievable.

Definition 2.3 (One-Time Pad (OTP))**.** The one-time pad encryption scheme over $\{0,1\}^n$ is defined by:

- $\mathsf{Gen}(1^n)$: choose $K \leftarrow \{0,1\}^n$ uniformly at random.
- $\mathsf{Enc}(K, m) = K \oplus m$, where $\oplus$ denotes bitwise XOR.
- $\mathsf{Dec}(K, c) = K \oplus c$.

Correctness: $\mathsf{Dec}(K, \mathsf{Enc}(K, m)) = K \oplus (K \oplus m) = m$.
For example, given a key $K = 10111$ and a message $m = 00011$, the ciphertext is

$$c \;=\; K \oplus m \;=\; 10111 \oplus 00011 \;=\; 10100.$$

Decryption recovers the message: $m = c \oplus K$.

Perfect secrecy: For any fixed m and any c, there is exactly one key $K = c \oplus m$ such that $\mathsf{Enc}(K, m) = c$. Since K is uniform, $\Pr[\mathsf{Enc}(K, m) = c] = 2^{-n}$ for every m and c, which is independent of m.

However, the one-time pad has critical limitations:

1. **Key length:** The key must be as long as the message.
2. **Single use:** Reusing a key compromises security. If $c_1 = K \oplus m_1$ and $c_2 = K \oplus m_2$, then $c_1 \oplus c_2 = m_1 \oplus m_2$, leaking the XOR of plaintexts.
3. **Synchronization:** If multiple keys are used, the receiver must know which key corresponds to which ciphertext.

2.4. Shannon's Impossibility Theorem

Shannon proved that the one-time pad's key length is inherently necessary.

Shannon's Impossibility Theorem

Theorem 2.2 (Shannon's Theorem)**.** *For any encryption scheme satisfying perfect secrecy, $|\mathcal{K}| \geq |\mathcal{M}|$. That is, the key must be at least as long as the message.*

Proof sketch. Suppose $|\mathcal{K}| < |\mathcal{M}|$. Fix a ciphertext c with $\Pr[C = c] > 0$. The set of possible plaintexts given c is $\mathcal{M}(c) = \{m : \exists k \text{ s.t. } \mathsf{Dec}(k, c) = m\}$. Since $|\mathcal{M}(c)| \leq |\mathcal{K}| < |\mathcal{M}|$, there exists some $m^* \notin \mathcal{M}(c)$. Then $\Pr[M = m^* \mid C = c] = 0$, but if $\Pr[M = m^*] > 0$, perfect secrecy is violated. □

This impossibility tells us that in the information-theoretic setting, we cannot do better than the one-time pad. To build practical encryption, we must make one of the following compromises.

3. From Information-Theoretic to Computational Security

Recall from the first lecture the three strategies for circumventing impossibility results:

1. Weaken the adversary model (probabilistic polynomial-time algorithm).
2. Weaken the definition of security.
3. Introduce new assumptions.

The revolution of modern cryptography, initiated by Diffie and Hellman [6] and formalized by Goldwasser and Micali [10], applies all three. We restrict the adversary to *probabilistic polynomial time* (PPT), relax perfect secrecy to *computational indistinguishability*, and base security on *computational assumptions* (such as the existence of one-way functions).

3.1. Probabilistic Polynomial Time (PPT)

Definition 3.1 (Probabilistic Polynomial Time)**.** An algorithm A is *PPT* if:

1. A runs in time $O(n^c)$ for some constant $c > 0$, where n is the input length / security parameter.
2. A is randomized: it has access to a source of fair coin flips.

3.2. Negligible Functions

The notion of "negligible probability" is central to computational security.

Definition 3.2 (Negligible Function)**.** A function $\mu : \mathbb{N} \to \mathbb{R}$ is *negligible* if for every constant $c > 0$, there exists $n_0 \in \mathbb{N}$ such that for all $n > n_0$:

$$|\mu(n)| < \frac{1}{n^c}.$$

Intuitively, negligible functions decrease faster than the inverse of any polynomial. Common examples include 2^{-n}, $2^{-\sqrt{n}}$, $n^{-\log n}$, and not $\frac{1}{n}$ or $\frac{1}{n^{100}}$.

The operational significance is that events occurring with negligible probability "look like they never occur" to polynomial-time algorithms. If an adversary's advantage is negligible, no efficient strategy can meaningfully exploit it.

3.3. Computational Security of Encryption

The computational analogue of perfect indistinguishability relaxes the requirement from exact equality to "close enough for any efficient observer." This definition, due to Goldwasser and Micali [10], is one of the most important ideas in modern cryptography.

Definition 3.3 (Computational Security of Encryption)**.** An encryption scheme $(\mathsf{Gen}, \mathsf{Enc}, \mathsf{Dec})$ is *computationally secure* if for every PPT adversary Eve, every PPT sampleable message distribution M, and all $m_0, m_1 \in M(1^n)$:

$$|\Pr[\mathsf{Eve}(c) = 1 \mid c \leftarrow \mathsf{Enc}(K, m_1)] - \Pr[\mathsf{Eve}(c) = 1 \mid c \leftarrow \mathsf{Enc}(K, m_0)]| \leq \mathrm{negl}(n),$$

where $K \leftarrow \mathsf{Gen}(1^n)$.

The definition requires that the distributions $\mathsf{Enc}(K, m_0)$ and $\mathsf{Enc}(K, m_1)$ are *computationally indistinguishable*: no PPT algorithm can tell them apart.

Equivalently, in the "guessing game" formulation: for every PPT Eve, all m_0, m_1,

$$\Pr\left[\mathsf{Eve}(c) = b\right] \leq \frac{1}{2} + \mathrm{negl}(n),$$

where $K \leftarrow \mathsf{Gen}(1^n)$, $b \leftarrow \{0,1\}$, $c = \mathsf{Enc}(K, m_b)$.

3.4. Revised Encryption: Public-Key Setting

Moving to the computational setting also enables *public-key* encryption, as envisioned by Diffie and Hellman [6] and realized by Rivest, Shamir, and Adleman [16].

Definition 3.4 (Public-Key Encryption Scheme)**.** A public-key encryption scheme $(\mathsf{Gen}, \mathsf{Enc}, \mathsf{Dec})$ is a triple of PPT algorithms:

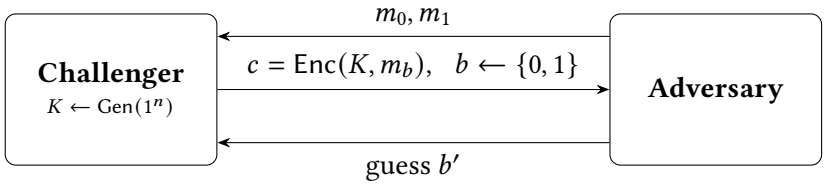
1. $\mathsf{Gen}(1^n)$ outputs a key pair $(\mathsf{pk}, \mathsf{sk})$, each of length n.
2. $\mathsf{Enc}(\mathsf{pk}, m)$ outputs ciphertext c.
3. $\mathsf{Dec}(\mathsf{sk}, c)$ outputs plaintext m'.

pk is a public key meant for encryption, and sk is a secret key meant for decryption. Correctness requires $\mathsf{Dec}(\mathsf{sk}, \mathsf{Enc}(\mathsf{pk}, m)) = m$ for all m, with high probability over the coins of Gen, Enc, and Dec.

The IND-CPA Game. This is often formalized as a challenger - adversary game. We will walk through an example of IND-CPA in a secret-key setting.

Remark 3.1. A public-key setting differs slightly. The adversary additionally receives pk before choosing m_0, m_1. The game is otherwise identical.

In the schema below, a challenger and adversary are interacting in a secret-key setting. The adversary will select an m_0 and m_1 that they believe they can distinguish between. The challenger will choose a key and encrypt either of the two messages. The scheme is secure if no PPT adversary can win the game by determining which message the challenger encrypted with probability non-negligibly better than $\frac{1}{2}$.



Wins if $b' = b$

New possibilities. With computational security, we can now ask and affirmatively answer questions impossible in the information-theoretic setting: (1) Can Alice and Bob meet once to agree on a short key and then securely exchange polynomially many messages? Yes. (2) Can Alice and Bob exchange messages without *ever* meeting (i.e., public-key cryptography)? Yes, via the constructions of Diffie–Hellman [6] and RSA [16].

4. Pseudorandom Generators

To overcome Shannon's impossibility theorem ($|\mathcal{K}| \geq |\mathcal{M}|$), we need a way to "stretch" a short key into a long pseudorandom string using a pseudorandom generator, a deterministic program that stretches a truly random short seed into a much longer sequence of bits that "looks random" to any efficient observer.

Notation. U_n (resp. U_m) denotes the uniform distribution on $\{0,1\}^n$ (resp. $\{0,1\}^m$). We use m as shorthand for $m(n)$.

Why Indistinguishability Matters

The indistinguishability definition (below) is powerful because of its universality: if $G(U_n)$ is indistinguishable from U_m, then $G(U_n)$ can replace true randomness in *any* polynomial-time application without affecting its behavior. If some application behaved differently with pseudorandom input, that application would itself be a distinguisher, contradicting the PRG definition.

4.1. Two Equivalent Definitions

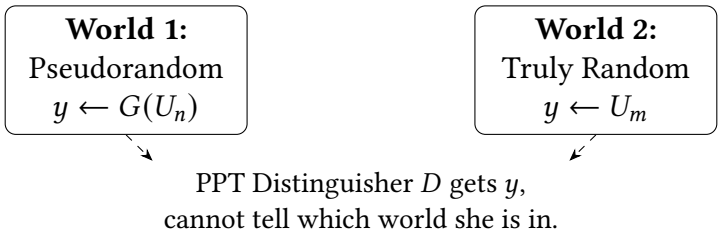
4.1.1 Indistinguishability (Passing All Statistical Tests)

Definition 4.1 (PRG: Indistinguishability)**.** A deterministic polynomial-time computable function $G : \{0,1\}^n \to \{0,1\}^m$ is (strong) pseudorandom generator if:

(a) $m = m(n) > n$ (expansion), and
(b) for every PPT algorithm D (a "statistical test"), there exists a negligible function negl such that:

$$|\Pr[D(G(U_n)) = 1] - \Pr[D(U_m) = 1]| \leq \text{negl}(n).$$

The distinguisher D receives a string y and must decide whether y was drawn from $G(U_n)$ (the "pseudorandom world") or from U_m (the "truly random world"). The definition requires that no efficient D can tell the difference.



4.1.2 Next-Bit Unpredictability

Definition 4.2 (PRG: Next-Bit Unpredictability)**.** A deterministic polynomial-time computable function $G : \{0,1\}^n \to \{0,1\}^m$ is a PRG if:

(a) $m > n$, and

(b) for every PPT algorithm PRED (a "next-bit predictor") and every $i \in [m]$, there exists a negligible function negl such that:

$$\Pr\left[\mathsf{PRED}(y_1 y_2 \cdots y_{i-1}) = y_i\right] < \frac{1}{2} + \mathrm{negl}(n),$$

where $y \leftarrow G(U_n)$ and y_j denotes the j-th bit of y.

If condition (b) holds, we say G "passes all next-bit tests."

4.2. Equivalence of the Two Definitions

Theorem 4.1. *A PRG G passes all polynomial-time statistical tests (Definition 4.1) if and only if it passes all polynomial-time next-bit tests (Definition 4.2).*

Proof sketch. ($\Rightarrow$) **Easy direction.** Any next-bit predictor PRED can be converted into a statistical test: given y, pick a random index i, run $\mathsf{PRED}(y_1 \cdots y_{i-1})$, and output 1 if the prediction matches y_i.

($\Leftarrow$) **Harder direction (Hybrid Argument).** Suppose G passes all next-bit tests but there exists a statistical test D that distinguishes $G(U_n)$ from U_m with non-negligible advantage. Define a sequence of $m+1$ hybrid distributions:

$$H_j = (y_1, y_2, \ldots, y_j, r_{j+1}, \ldots, r_m),$$

where $y \leftarrow G(U_n)$ and $r_{j+1}, \ldots, r_m \leftarrow \{0,1\}$ independently. Then $H_0 = U_m$ and $H_m = G(U_n)$. By the triangle inequality, D must distinguish some adjacent pair H_{j-1} and H_j with advantage at least ε/m (where ε is D's original advantage). This distinguisher yields a next-bit predictor for position j with non-negligible advantage, a contradiction. □

4.3. PRG Implies Short-Key Encryption

PRG ⇒ Short-Key Encryption

Theorem 4.2 (PRG ⇒ Computationally Secure Encryption)**.** *If $G : \{0,1\}^n \to \{0,1\}^{n+1}$ is a PRG, then the following encryption scheme is computationally secure:*

- $\mathsf{Gen}(1^n)$*: choose* $K \leftarrow \{0,1\}^n$.
- $\mathsf{Enc}(K, m)$ *for* $m \in \{0,1\}^{n+1}$*: output* $c = G(K) \oplus m$.
- $\mathsf{Dec}(K, c)$*: output* $G(K) \oplus c$.

This scheme encrypts $(n+1)$-bit messages using only an n-bit key, **breaking the Shannon barrier**. For longer messages, we simply use a PRG with larger stretch.

Proof. **Correctness:** $\mathsf{Dec}(K, c) = G(K) \oplus c = G(K) \oplus G(K) \oplus m = m$.

Security: By contradiction. In the "ideal world," replace $G(K)$ with a truly random string $r \leftarrow \{0,1\}^{n+1}$. Then $c = r \oplus m_b$ is uniformly distributed regardless of b, so no adversary can distinguish m_0 from m_1.

If Eve could distinguish $\mathsf{Enc}(K, m_0)$ from $\mathsf{Enc}(K, m_1)$ (with $G(K)$ as the pad), then Eve could distinguish $G(U_n)$ from U_{n+1}: given a challenge string $z \in \{0,1\}^{n+1}$, compute $z \oplus m_0$ and $z \oplus m_1$, and use Eve's distinguishing strategy. This contradicts the PRG assumption. □

5. Randomness: The Foundation

Randomness underpins both cryptography and machine learning:

In cryptography, keys must be unpredictable, and algorithms rely on additional randomness beyond the key. If the random bits are revealed or predictable, the entire security structure collapses.

In machine learning, randomness is crucial for training (SGD, ERM use random sampling) and for generation (e.g., in LLMs). This raises natural questions at the intersection: Can poor use of randomness lead to biased outcomes? Can specially designed PRGs enforce ML-desirable properties such as fairness or differential privacy? We will explore these connections in later lectures.

The PRG seed must be truly random, though generating true randomness is nontrivial in practice. Sources include specialized hardware (transistor noise), user input (biased but correctable via randomness extractors), and operating system entropy pools [4, 7, 18].

6. One-Way Functions

The central assumption of modern cryptography is the existence of one-way functions: functions that are easy to compute but hard to invert.

Definition 6.1 (One-Way Function)**.** A function $f : \{0,1\}^* \to \{0,1\}^*$ is a *one-way function (OWF)* if:

1. **Easy to evaluate:** There exists a PPT algorithm A such that $A(x) = f(x)$ for all x.
2. **Hard to invert:** For every PPT algorithm Inv and all sufficiently large n:

$$\Pr_{x \leftarrow \{0,1\}^n}[\mathsf{Inv}(1^n, f(x)) = x' \text{ s.t. } f(x') = f(x)] \leq \mathrm{negl}(n).$$

   where the probability is also over the coins of Inv. Note that there may exist multiple inverses x'.

Remark 6.1. An unbounded algorithm can always find an inverse (by exhaustive search). The hardness is specifically for *probabilistic polynomial-time* inverters, and it is an *average-case* hardness requirement (over random inputs).

Special cases:

- A *length-preserving* OWF satisfies $|f(x)| = |x|$ for all x.
- A *one-way permutation (OWP)* is a one-to-one, length-preserving OWF.

When working with a collection (family) of one-way functions $\{f_n : \{0,1\}^n \to \{0,1\}^{m(n)}\}_{n\in\mathbb{N}}$, we index by the security parameter n and require both efficient evaluation and hardness of inversion as above. A single function $f : \{0,1\}^* \to \{0,1\}^*$ can be viewed as a collection by restricting f to inputs in $\{0,1\}^n$.

6.1. Candidate One-Way Functions

One-way function candidates are abundant in number theory, coding theory, and combinatorics.

Example 6.2 (Subset Sum)**.** Define $f(a_1, \ldots, a_n, x_1, \ldots, x_n) = \left(a_1, \ldots, a_n, \ \sum_{i=1}^{n} x_i a_i \bmod 2^{n+1}\right)$, where a_i are random n-bit numbers and $x_i \in \{0,1\}$. Inverting f requires solving a random instance of the subset sum problem, which is believed to be hard on average.

Worst-Case vs. Average-Case Hardness

Subset Sum is NP-complete, but this is a worst-case statement. By contrast, one-wayness is an average-case requirement: the inverter must fail on a uniformly random input (here, over random $a_1, \ldots, a_n$ and the coins of x). The assumption $P \neq NP$ does not by itself imply that a given NP-complete problem is hard on the relevant input distribution, nor does it imply the existence of one-way functions. For a detailed discussion of the landscape of average-case hardness assumptions, see Impagliazzo's taxonomy of "five worlds" [13].

Example 6.3 (Rabin's Function)**.** $f_N(x) = x^2 \bmod N$, where $N = pq$ is an RSA modulus. Rabin [15] showed that inverting this function is as hard as factoring N.

Example 6.4 (RSA Function)**.** $f_N(x) = x^e \bmod N$, where $\gcd(e, \varphi(N)) = 1$. The RSA assumption [16] states that this is one-way without knowledge of the factorization of N.

Example 6.5 (Discrete Log (Diffie–Hellman))**.** $f_{g,p}(x) = g^x \bmod p$, where p is a large prime and g is a generator of $\mathbb{Z}_p^*$. The discrete logarithm assumption [6] states that this is one-way.

The existence of one-way functions is a widely believed but unproven assumption. It implies **P** $\neq$ **NP**, though the converse is not known: consistent with our current understanding, it may be that **P** $\neq$ **NP** yet no one-way functions exist (Impagliazzo's "Pessiland" [13]).

7. Hard-Core Predicates

A one-way function is hard to invert, but does the output necessarily hide *every* bit of the input? The answer is no—there exist one-way functions for which certain bits of the preimage are easy to compute.

Example 7.1. Consider a one-way function f and define

$$f'(x, b) = (f(x), b).$$

Then f' is still one-way, but the string b is trivially recoverable from $f'(x, b)$. In particular, by taking b to consist of (say) the first $|x|/2$ bits of the input, we obtain a one-way function for which the first half of the input bits can be efficiently recovered.

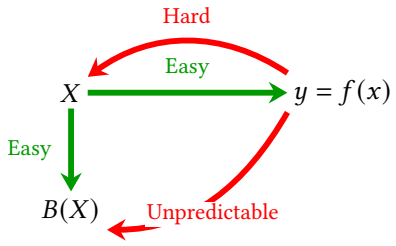
This motivates the notion of a hard-core predicate: a single bit of information about x that is provably hard to predict given $f(x)$.

Definition 7.1 (Hard-Core Predicate)**.** A Boolean predicate $B : \{0,1\}^* \to \{0,1\}$ is a *hard-core predicate* for a one-way function $f : \{0,1\}^* \to \{0,1\}^*$ if:

1. $B(x)$ is efficiently computable given x.
2. For every PPT predictor P:

$$\Pr_{x \leftarrow \{0,1\}^n}[P(1^n, f(x)) = B(x)] \leq \tfrac{1}{2} + \mathrm{negl}(n).$$

In other words, $B(x)$ is easy to compute given x (the forward direction) but computationally unpredictable given only $f(x)$ (the backward direction).



Remark 7.2. Some specific functions have known hard-core predicates: the least significant bit is hard-core for the RSA function $f_N(x) = x^e \bmod N$ [1], though it is *not* hard-core for the Diffie–Hellman function. For example, if there is a PPT algorithm P and a non-negligible $\varepsilon(n)$ such that for RSA,

$$\Pr_{x \leftarrow \mathbb{Z}_N^*}[P(x^e \bmod N) = \mathrm{LSB}(x)] \geq \tfrac{1}{2} + \varepsilon(n),$$

then one can build a PPT inverter A for RSA (using P as a subroutine) that recovers x from $x^e \bmod N$ with non-negligible probability. This raises the question: is there a *universal* hard-core predicate that works for every OWF?

7.1. The Goldreich–Levin Theorem

The Goldreich–Levin theorem provides an affirmative answer: every one-way function f can be transformed into a function $f'(x, r) = (f(x), r)$ that has an associated hard-core predicate. The key idea is to take a fresh random string $r \in \{0,1\}^n$ and look at the parity of a randomly selected subset of the bits of x, namely the mod-2 inner product $\langle r, x \rangle$. Intuitively, r specifies a subset of coordinates (those with $r_i = 1$), and $\langle r, x \rangle$ is the XOR of the corresponding bits of x.

The theorem shows that if an efficient adversary can predict this parity from $f(x)$ and r with noticeable advantage over $1/2$, then one can use that predictor as a subroutine to *recover* x from $f(x)$ with non-negligible probability, contradicting that f is one-way. In other words, for every one-way function f, the family of predicates $\{B_r\}_{r \in \{0,1\}^n}$ defined below is provably hard-core.

Beyond cryptography, the proof had major influence on coding theory: the reconstruction step can be viewed as a *list-decoding* procedure for the Hadamard code (and later helped inspire further

developments in list decoding). A downside is that the resulting hard-core bit is not a simple "ith bit" of x; it is a global parity of many bits, so extracting a specific coordinate of x from $f(x)$ is not what this theorem provides.

Goldreich–Levin Theorem

Theorem 7.3 (Goldreich–Levin). *For $r \in \{0,1\}^n$, define the predicate $B_r : \{0,1\}^n \to \{0,1\}$ by:*

$$B_r(x) = \langle r, x \rangle = \sum_{i=1}^{n} r_i x_i \bmod 2.$$

Then for every *one-way function f and every PPT algorithm A:*

$$\Pr_{\substack{x \leftarrow \{0,1\}^n \\ r \leftarrow \{0,1\}^n}} [A(f(x), r) = B_r(x)] \leq \tfrac{1}{2} + \mathrm{negl}(n).$$

Equivalently, $B(x, r) = \langle r, x \rangle$ is a deterministic hard-core predicate for the augmented function $f'(x, r) = (f(x), r)$. Note: there is no single fixed predicate $B(x)$ that is hard-core for every one-way function. Rather, Goldreich–Levin gives a universal construction: for each f, after augmenting to $f'(x, r) = (f(x), r)$, the predicate $B(x, r) = \langle r, x \rangle$ is hard-core for f'.

The following theorem is due to Goldreich and Levin [8].

Interpretation. There are two useful ways to view this result:

1. **Randomized hard-core bit:** A random inner product $\langle r, x \rangle$ is hard-core for any OWF when r is chosen uniformly at random.
2. **Deterministic hard-core bit via augmentation:** Define $f'(x, r) = (f(x), r)$. Then f' is also one-way (since f is), and $B(x, r) = \langle r, x \rangle$ is a deterministic hard-core predicate for f'.

The proof of the Goldreich–Levin theorem is a fundamental result in computational complexity. At a high level, it shows that if some algorithm A could predict $\langle r, x \rangle$ given $(f(x), r)$ with non-negligible advantage, then A could be used to reconstruct x entirely (inverting f), contradicting one-wayness. The reconstruction uses a clever self-correction technique exploiting the linearity of the inner product.

8. Constructing PRGs from One-Way Permutations

We now have all the ingredients to build a cryptographically strong PRG from a one-way permutation.

8.1. Naive Attempt and Its Failure

A natural first idea (credited to Adi Shamir): given a one-way permutation f, choose a random seed s, and output $f(s)$, $f^2(s)$, $f^3(s)$, ... in reverse order. Some intuition supports this: from $f^i(s)$, one cannot compute $f^{i-1}(s)$ (by one-wayness). However, while the *entire* preimage is hard

to recover, individual bits of $f^{-1}(y)$ may be predictable. Indeed, there exist one-way functions for which the first half of the inverse bits are easy to compute. Moreover, the output is also easily distinguishable from random: there is an efficient distinguisher that checks whether adjacent blocks are consistent under f. If the output is $(f^t(s), f^{t-1}(s), \ldots, f(s))$, then each adjacent pair satisfies the relation $f(z_{i+1}) = z_i$. A distinguisher can simply test this condition on all adjacent blocks, whereas a truly random sequence would satisfy it only with negligible probability. Thus, raw iteration of f is not enough.

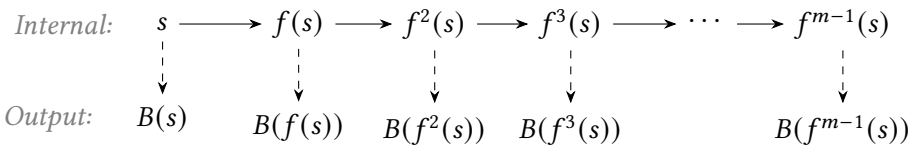
8.2. The Blum–Micali Construction

The solution, due to Blum and Micali [5], is to iterate f but output only the hard-core bit at each step.

Theorem 8.1 (OWP + Hard-Core Bit ⇒ PRG)**.** *Let $f : \{0,1\}^n \to \{0,1\}^n$ be a one-way permutation with hard-core predicate B. For any polynomial $P(\cdot)$, define $G : \{0,1\}^n \to \{0,1\}^{P(n)}$ by: on input seed $s \in \{0,1\}^n$,*

1. *Compute $f(s), f^2(s), \ldots, f^{P(n)-1}(s)$.*
2. *Output $B(s),\ B(f(s)),\ B(f^2(s)),\ \ldots,\ B(f^{P(n)-1}(s))$.*

Then G is a strong PRG.



Proof of Theorem 8.1. We show that G passes all next-bit tests (Definition 4.2), which by Theorem 4.1 implies it passes all statistical tests.

Suppose for contradiction that there exists a bit position $j < m$ (where $m = P(n)$) and a PPT next-bit predictor PRED such that:

$$\Pr_{y \leftarrow G(U_n)}\left[\mathsf{PRED}(y_1 \cdots y_{j-1}) = y_j\right] > \tfrac{1}{2} + \frac{1}{p(n)}.$$

for some polynomial p.

We construct a PPT algorithm P' that predicts $B(x)$ from $f(x)$, contradicting the hard-core property. Given $f(x)$, the predictor P' works as follows:

1. Compute $f^2(x), f^3(x), \ldots, f^j(x)$. (This is efficient since f is easy to evaluate.)
2. Compute the first $j-1$ output bits: $y_1 = B(f^j(x)),\ y_2 = B(f^{j-1}(x)),\ \ldots,\ y_{j-1} = B(f(x))$.
3. Output $\mathsf{PRED}(y_1, \ldots, y_{j-1})$.

We observe that the next bit y_j in the PRG output sequence is precisely $B(x)$! The sequence continues backward: the output is $B(s), B(f(s)), \ldots$, so if we are given $f(x)$ and compute forward, the bits $y_1, \ldots, y_{j-1}$ are exactly the bits that PRED expects to see.

The distribution is correct because f is a permutation, so when s is uniformly random, $x = f^{-(j)}(s)$ is also uniformly random. Thus $f(x)$ is uniformly distributed over $\{0,1\}^n$, and the bits $y_1, \ldots, y_{j-1}$ fed to PRED follow exactly the distribution that PRED was designed for.

Therefore:

$$\Pr\left[P'(f(x)) = B(x)\right] = \Pr\left[\mathsf{PRED}(y_1 \cdots y_{j-1}) = y_j\right] > \frac{1}{2} + \frac{1}{p(n)},$$

contradicting the hard-core property of B. □

Remark 8.2 (Role of f being a permutation). The proof crucially uses the fact that f is a permutation: this ensures that the distribution of the internal states (and hence the output bits) seen by PRED is exactly correct. For general one-way functions (which may be many-to-one), both the construction and the proof are far more delicate.

8.3. Using the Goldreich–Levin Hard-Core Bit

Combining with the Goldreich–Levin theorem (Theorem 7.3), we obtain an explicit PRG from any OWP f: choose a random $r \in \{0,1\}^n$ (which can be made public), and output:

$$G(s) = \big(\langle r, s\rangle,\ \langle r, f(s)\rangle,\ \langle r, f^2(s)\rangle,\ \ldots,\ \langle r, f^{m-1}(s)\rangle\big).$$

8.4. Length Extension: From One Extra Bit to Polynomial Stretch

Theorem 8.3. *If there exists a cryptographically strong PRG $g : \{0,1\}^n \to \{0,1\}^{n+1}$ that stretches by a single bit, then for any polynomial P, there exists a PRG $G : \{0,1\}^n \to \{0,1\}^{P(n)}$.*

Construction and proof idea. Write $g(x) = (g(x)|_{\le n},\ g(x)|_{n+1})$, where $g(x)|_{\le n}$ denotes the first n bits and $g(x)|_{n+1}$ denotes the $(n+1)$-th bit. Define G iteratively:

$$\begin{aligned} x_0 &= s, \\ x_{i+1} &= g(x_i)|_{\le n}, \quad b_i = g(x_i)|_{n+1}, \\ G(s) &= b_0\, b_1\, b_2\, \cdots\, b_{P(n)-1}. \end{aligned}$$

Security follows by a hybrid argument: define hybrid H_j where the first j output bits are pseudorandom (from the PRG chain) and the remaining bits are truly random. Adjacent hybrids H_j and H_{j+1} are indistinguishable by the single-bit PRG security, and there are polynomially many hybrids. □

8.5. From OWFs to PRGs (General Case)

The construction above requires a one-way *permutation*. The general result, meaning that the mere existence of *any* one-way function suffices, is much harder to prove.

The HILL Theorem: OWF ⇒ PRG

Theorem 8.4 (Håstad, Impagliazzo, Levin, Luby [12])**.** *If one-way functions exist, then for any polynomial P, cryptographically strong PRGs $G : \{0,1\}^n \to \{0,1\}^{P(n)}$ exist.*
This is the foundational result underpinning modern symmetric cryptography: the *mere existence* of any hard-to-invert function is sufficient to build a secure PRG with arbitrary stretch.

The HILL construction is significantly more involved and is one of the deepest results in the foundations of cryptography. A much simpler proof was recently given by Berman, Degwekar, Rothblum, and Vasudevan [2]. We refer the reader to the course website for details.

9. Connections to Machine Learning

Several results from this lecture connect directly to machine learning, and will be developed further in upcoming lectures.

Cryptographic hardness and learning. Kearns and Valiant [14] showed that the existence of one-way functions implies that certain concept classes (e.g., boolean formulae, finite automata) are not efficiently PAC-learnable: a learning algorithm for such a class could be used to invert a one-way function. Blum, Furst, Kearns, and Lipton [3] further developed this direction by basing cryptographic primitives on the hardness of specific learning problems (such as learning parity with noise).

Kerckhoffs's principle for ML. As noted in lecture, the principle that security should not rely on algorithm secrecy has a natural analogue: the robustness of ML systems should not depend on the secrecy of model architectures or training procedures. For an interesting perspective on this, see Hall et al. [11].

10. Summary and Looking Ahead

What we covered:

1. Defined encryption schemes and Shannon's perfect secrecy; proved the one-time pad achieves it, and proved Shannon's impossibility ($|\mathcal{K}| \geq |\mathcal{M}|$).
2. Transitioned to computational security: PPT adversaries, negligible functions, and computational indistinguishability [10].
3. Defined pseudorandom generators via two equivalent formulations (indistinguishability and next-bit unpredictability).
4. Defined one-way functions and some candidates (subset sum, RSA, Rabin, discrete log).
5. Defined hard-core predicates and stated the Goldreich–Levin theorem: every OWF can be augmented to have a hard-core bit.
6. Proved that OWP + hard-core bit ⇒ PRG (Blum–Micali construction), and stated the general HILL theorem (OWF ⇒ PRG).

Coming next (Lectures 5–6): Pseudorandom functions (PRFs) and their construction from PRGs via the GGM tree construction [9]; applications of PRFs.

References

[1] Werner Alexi, Benny Chor, Oded Goldreich, and Claus-Peter Schnorr. RSA and Rabin functions: Certain parts are as hard as the whole. *SIAM Journal on Computing*, 17(2):194–209, 1988.

[2] Itay Berman, Akshay Degwekar, Ron D. Rothblum, and Prashant Nalini Vasudevan. Counting unpredictable bits: A simple PRG from one-way functions. *SIAM Journal on Computing*, 52 (1):128–177, 2023.

[3] Avrim Blum, Merrick Furst, Michael Kearns, and Richard J. Lipton. Cryptographic primitives based on hard learning problems. In *Advances in Cryptology — CRYPTO '93*, pages 278–291, 1993.

[4] Manuel Blum. Independent unbiased coin flips from a correlated biased source—a finite state Markov chain. *Combinatorica*, 6(2):97–108, 1986.

[5] Manuel Blum and Silvio Micali. How to generate cryptographically strong sequences of pseudorandom bits. *SIAM Journal on Computing*, 13(4):850–864, 1984.

[6] Whitfield Diffie and Martin E. Hellman. New directions in cryptography. *IEEE Transactions on Information Theory*, 22(6):644–654, 1976.

[7] Peter Elias. The efficient construction of an unbiased random sequence. *The Annals of Mathematical Statistics*, 43(3):865–870, 1972.

[8] Oded Goldreich and Leonid A. Levin. A hard-core predicate for all one-way functions. In *Proceedings of the 21st ACM Symposium on Theory of Computing (STOC)*, pages 25–32, 1989.

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

[10] Shafi Goldwasser and Silvio Micali. Probabilistic encryption & how to play mental poker keeping secret all partial information. In *Proceedings of the 14th ACM Symposium on Theory of Computing (STOC)*, pages 365–377, 1982.

[11] Peter Hall, Olivia Mundahl, and Sunoo Park. The pitfalls of "security by obscurity" and what they mean for transparent ai. *arXiv preprint arXiv:2501.18669*, 2025.

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

[13] Russell Impagliazzo. A personal view of average-case complexity. In *Proceedings of the 10th Annual Structure in Complexity Theory Conference*, pages 134–147, 1995.

[14] Michael Kearns and Leslie G. Valiant. Cryptographic limitations on learning boolean formulae and finite automata. *Journal of the ACM*, 41(1):67–95, 1994.

[15] Michael O. Rabin. Digitalized signatures and public-key functions as intractable as factorization. Technical Report TR-212, MIT Laboratory for Computer Science, 1979.

[16] Ronald L. Rivest, Adi Shamir, and Leonard Adleman. A method for obtaining digital signatures and public-key cryptosystems. *Communications of the ACM*, 21(2):120–126, 1978.

[17] Claude E. Shannon. Communication theory of secrecy systems. *Bell System Technical Journal*, 28(4):656–715, 1949.

[18] John von Neumann. Various techniques used in connection with random digits. *National Bureau of Standards Applied Mathematics Series*, 12:36–38, 1951.