Formal Proofs in Cryptography

Somitra Sanadhya

February 1, 2025

Part-1: Perfect Security

Perfect Security

- In cryptography, a cryptosystem is said to have **perfect security** if an adversary gains no additional information about the plaintext after observing the ciphertext.
- Contents of this part:
  - Different definitions of perfect security
  - Equivalence of these definitions
  - A practical issue in using a perfectly secure encryption scheme

Notation and Cryptosystem Components

- Let $\mathcal{K}$ denote the set of keys.
- Let $\mathcal{M}$ denote the set of messages (plaintexts).
- Let $\mathcal{C}$ denote the set of ciphertexts.
- $E_k : \mathcal{M} \to \mathcal{C}$ is the encryption function using key k.
- $D_k : \mathcal{C} \to \mathcal{M}$ is the decryption function using key k.

Assumption: Encryption is done with a randomly selected key k from $\mathcal{K}$ according to some probability distribution.

Definition 1: Ciphertext Distribution Independence

Definition

A cryptosystem is said to be **perfectly secure** if for all messages $m_1, m_2 \in \mathcal{M}$ and for all ciphertexts $c \in \mathcal{C}$:

$$P(C = c \mid M = m_1) = P(C = c \mid M = m_2).$$

- This definition states that the probability of obtaining any ciphertext c is independent of the original message m.
- Thus, observing the ciphertext provides no clue as to which message was encrypted.

Definition 2: Shannon's Definition

Definition

A cryptosystem achieves perfect security if:

$$P(M = m \mid C = c) = P(M = m) \quad \text{for all } m \in \mathcal{M} \text{ and } c \in \mathcal{C}.$$

- This definition implies that the a-posteriori probability of any message m, given the ciphertext c, is the same as the a-priori probability of m.
- In other words, the ciphertext c does not change our beliefs about the likelihood of any message.

Definition 3: Key-Message Relation

Definition

For every plaintext m and every ciphertext c, there exists a key k such that:

$$E_k(m) = c.$$

- This ensures that no matter what message and ciphertext are chosen, there is always a key that could have produced c from m.
- It highlights that the encryption process does not favor any particular plaintext-ciphertext pairing.

Proof: Equivalence of Definition 1 and Definition 2

Goal: Show that the condition

$$P(C = c \mid M = m_1) = P(C = c \mid M = m_2) \quad \forall m_1, m_2 \in \mathcal{M},\ c \in \mathcal{C},$$

is equivalent to Shannon's definition:

$$P(M = m \mid C = c) = P(M = m) \quad \forall m \in \mathcal{M},\ c \in \mathcal{C}.$$

Proof:

- **Assume:** $P(M \mid C) = P(M)$. By Bayes' theorem:

$$P(M = m \mid C = c) = \frac{P(C = c \mid M = m)P(M = m)}{P(C = c)}.$$

  Setting $P(M = m \mid C = c) = P(M = m)$ gives:

$$\frac{P(C = c \mid M = m)P(M = m)}{P(C = c)} = P(M = m)$$

$$\implies \quad P(C = c \mid M = m) = P(C = c).$$

- This shows that the ciphertext distribution is independent of the message.

… contd.

- Conversely, if $P(C = c \mid M = m) = P(C = c)$ holds, then by applying Bayes' theorem in reverse, we obtain:

$$P(M = m \mid C = c) = \frac{P(C = c)P(M = m)}{P(C = c)} = P(M = m).$$

- This proves the equivalence of definitions 1 and 2.

Proof: Key-Message Relation and Perfect Security

- The key-message relation (every message-ciphertext pair can be produced by some key) ensures that there is no bias in the encryption process.
- This condition guarantees that for any given m and c, the probability $P(C = c \mid M = m)$ is the same because the set of keys that map m to c does not depend on m.
- Hence, if this condition is met, it is another way of stating that $P(C = c \mid M = m)$ is independent of m, which is equivalent to the other definitions of perfect security.

Practical problem in using Perfect Security

Theorem

For any perfectly secure encryption scheme,

$$|K| \geq |M|$$

i.e. the keyspace should be larger than the message space.

- We will prove this statement by contrapositive (i.e. to prove $A \Rightarrow B$ we show that $\neg B \Rightarrow \neg A$.
- Assume that $|K| < |M|$, and we have a ciphertext $c \in C$.
- Construct the set $X = \{\forall k \in K, Dec(k, c)\}$, i.e. decrypt c with all possible keys.
- Clearly $|X| \leq |K|$. But we already had $|K| < |M|$. This means there exists a message $m_0 \notin X$, and another message $m_1 \in X$.
- Hence $\Pr[C = c | M = m_0] = 0$, but $\Pr[C = c | M = m_1] \neq 0$.
- Hence the scheme can't be perfectly secure.

OTP

- OTP : One Time Pad encryption scheme.
- $M = K = C = \{0,1\}^n$
- KeyGen: generate a key $k \in K$ uniformly at random. That is, for any key k, we have that $\Pr[K = k] = \frac{1}{2^n}$.
- $Enc(k, m) = m \oplus k$
- $Dec(k, c) = c \oplus k$

OTP is perfectly secure

Theorem

OTP is a perfectly secure encryption scheme.

Proof.

Left as an exercise. $\square$

Summary and Conclusion

- We have reviewed multiple definitions of perfect security:
  1. Independence of ciphertext distribution from the plaintext.
  2. Shannon's definition: $P(M \mid C) = P(M)$.
  3. Key-message relation: For any m and c, there is a key k such that $E_k(m) = c$.
- We provided formal proofs demonstrating the equivalence of these definitions.
- Perfect security requires key space to be larger than message space, i.e. to send an n-bit message, the communicating parties will need to share a secret key of size $\geq n$ bits.
- While the One-Time Pad is a classic example of a system with perfect security, practical constraints usually prevent its widespread use.

Part-2: Pre-requisites for the proofs that follow

Contents

- In this part, we discuss the following
  - Negligible functions
  - Computational Security
  - Proving security or insecurity of a cryptographic scheme
  - Reduction proofs

Negligible Functions: Motivation

- In cryptography, we are interested in the security of systems against adversaries.
- We say an adversary is **efficient** if it runs in polynomial time.
- Security is defined in an asymptotic sense, where an adversary's advantage should vanish as the security parameter n increases.

Definition of Negligible Functions

Definition

A function $\mu : \mathbb{N} \to \mathbb{R}^+$ is **negligible** if for every positive polynomial $p(n)$ there exists an $n_0 \in \mathbb{N}$ such that

$$\mu(n) < \frac{1}{p(n)} \quad \text{for all } n \geq n_0.$$

- Informally, a function is negligible if it decreases faster than the inverse of any polynomial (beyond some threshold n_0).
- Try to compare this definition with asymptotic runtime definition for algorithms that you may have already studied.

Why do we define negligible functions this way?

- We permit "efficient" adversaries, i.e. adversaries who are allowed to run in time $poly(n)$ where n is the security parameter.
- If the adversary is able to "break" a scheme with probability p, then, potentially, she can increase her probability of breaking the scheme by running the attack algorithm $poly(n)$ times.
- Hence, we want that the probability of breaking the scheme should be such that even if it is multiplied by a polynomial in n, it still remains negligible.
- The following two statements are left as exercises for you. If $\epsilon_1(n)$ and $\epsilon_2(n)$ are two negligible functions then
  1. $\epsilon_1(n) + \epsilon_2(n)$ is also negligible.
  2. $\epsilon_1(n) \times \epsilon_2(n)$ is also negligible.

Negligible Functions

- $\mu(n) = 2^{-n}$ is negligible because for any polynomial $p(n)$, there exists n_0 such that

$$2^{-n} < \frac{1}{p(n)} \quad \text{for all } n \geq n_0.$$

- $\mu(n) = n^{-c}$ for some constant $c > 0$ is **not** negligible because it only decays polynomially.

Computational Security

- Consider the following two attacks against an encryption scheme:
  1. Given a $c \in C$, the adversary $\mathcal{A}$ runs $Dec(k, c)$ for all keys $k \in K$. Assuming that $\mathcal{A}$ has the ability to identify which is the correct plaintext (e.g. by the syntax of the message), $\mathcal{A}$ wins with probability 1 in breaking the scheme. Effort required is 2^n if the key size is n bits.
  2. $\mathcal{A}$ guesses the secret key and finds the correct message by decerypting the ciphertext with that key. Success probability in this case $= 1/2^n$ but runtime $= 1$.
- We would like to consider the cases in between these two extremes.
- We will restrict $\mathcal{A}$ to run their attack algorithm in time $poly(n)$ only. And then we would like their success probability of breaking the scheme to be $\epsilon(n)$, which should be negligible in security parameter n.
- This setting is called the "computational security" model.

Proving security or insecurity of a cryptographic scheme

- If a scheme is insecure then showing one attack against it is enough.
- If a scheme is secure that absence of certain attacks is not enough. We need to “formally” prove the security of the scheme.
- Usually, we will follow the “reduction proof” approach to prove the security of various schemes.

What is a Reduction Proof?

- A **reduction proof** is a technique used to demonstrate that breaking a cryptographic scheme is at least as hard as solving a well-known hard problem.
- The idea is to show that if an adversary can break the scheme with non-negligible advantage, then we can construct an algorithm that solves the hard problem with non-negligible probability.

Basic Structure of a Reduction Proof

1. **Assumption:** Assume there exists an adversary $\mathcal{A}$ that breaks the cryptosystem with non-negligible advantage.
2. **Construction:** Build a new algorithm $\mathcal{B}$ that uses $\mathcal{A}$ as a subroutine.
3. **Contradiction:** Show that $\mathcal{B}$ solves a known hard problem with non-negligible probability, contradicting its assumed hardness.

Example: Reduction in the Context of IND-CPA Security

- **IND-CPA Security:** A scheme is IND-CPA secure if no polynomial-time adversary can distinguish encryptions of any two chosen plaintexts with non-negligible advantage.
- **Reduction Idea:** Suppose there exists an adversary $\mathcal{A}$ that achieves a non-negligible advantage $\epsilon(n)$ in breaking the scheme.
- **Constructing $\mathcal{B}$:** We use $\mathcal{A}$ to build an algorithm $\mathcal{B}$ that can, for example, solve a hard problem.

Reduction Proof Outline

Step 1: Setup

- Given an instance of a hard problem (e.g., PRG instance), algorithm $\mathcal{B}$ prepares the input for the adversary $\mathcal{A}$ by simulating the cryptosystem.

Step 2: Interaction with $\mathcal{A}$

- $\mathcal{B}$ runs $\mathcal{A}$ on this simulated environment.
- $\mathcal{A}$ outputs a guess or decision.

Step 3: Conclusion

- Based on $\mathcal{A}$'s output and some additional computation, $\mathcal{B}$ decides whether the original instance was from the hard problem's distribution or not.
- The non-negligible advantage of $\mathcal{A}$ translates to a non-negligible advantage for $\mathcal{B}$, thus contradicting the hardness assumption.

Formalizing the Advantage

Adversary's Advantage

The advantage $\epsilon(n)$ of an adversary $\mathcal{A}$ is defined as:

$$\epsilon(n) = \left| \Pr[\mathcal{A} \text{ wins}] - \frac{1}{2} \right|.$$

- For a secure scheme, we require that $\epsilon(n)$ is a negligible function.
- A reduction shows that if $\epsilon(n)$ were non-negligible, then one could solve a hard problem with probability at least $\epsilon(n)$, leading to a contradiction.

Putting It All Together

- Negligible functions quantify what it means for an adversary's advantage to be insignificant.
- Reduction proofs allow us to leverage the assumed hardness of computational problems to argue the security of cryptographic schemes.
- The common theme is that any non-negligible advantage in breaking the scheme leads to a contradiction with established hardness assumptions.

Part-3: PRG

Contents

- In this part, we will cover the following:
  - Formal definition of a PRG
  - Formal definition of an IND-CPA secure encryption scheme
  - Using PRG to construct an IND-CPA secure encrytion scheme. To prove that the scheme is secure, we will show a reduction proof.
  - Finally, we will show some examples of constructing secure and insecure PRG's from a given secure PRG.

What is a Pseudorandom Generator?

- A **pseudorandom generator** (PRG) is a deterministic polynomial-time algorithm G that takes a short, uniformly random seed s and outputs a longer string $G(s)$.
- Formally, if $s \in \{0,1\}^n$ then:

  $$G : \{0,1\}^n \to \{0,1\}^{\ell(n)},$$

  where $\ell(n) > n$.
- The output of G is *indistinguishable* from a truly random string of length $\ell(n)$ by any polynomial-time algorithm.

Properties of a PRG

- **Expansion:** The function G stretches a short seed into a longer string.
- **Pseudorandomness:** For any polynomial-time distinguisher D, the difference in probability between D distinguishing $G(s)$ and a truly random string is negligible:

$$\left| \Pr_{s \leftarrow \{0,1\}^n}[D(G(s)) = 1] - \Pr_{r \leftarrow \{0,1\}^{\ell(n)}}[D(r) = 1] \right| < \mathrm{negl}(n).$$

The Encryption Scheme

Setup: Let $G : \{0,1\}^n \to \{0,1\}^{\ell(n)}$ be a PRG.

Key Generation:

- The secret key k is chosen uniformly at random from $\{0,1\}^n$.

Encryption:

$$\text{To encrypt a message } m \in \{0,1\}^{\ell(n)} : \quad c = m \oplus G(k),$$

where $\oplus$ denotes bitwise XOR.

Decryption:

$$\text{Given } c \text{ and key } k, \quad m = c \oplus G(k).$$

Correctness of the Scheme

- The decryption recovers the original message because:

  $$c \oplus G(k) = (m \oplus G(k)) \oplus G(k) = m \oplus (G(k) \oplus G(k)) = m \oplus 0 = m.$$

- Thus, the scheme is correct.

Security Notion: IND-CPA

- We wish to prove that the encryption scheme is IND-CPA secure, meaning that no efficient adversary can distinguish the encryptions of two messages of their choice.
- In the IND-CPA experiment, the adversary chooses two messages $m_0, m_1 \in \{0,1\}^{\ell(n)}$. A random bit b is chosen, and the challenge ciphertext is:

$$c = m_b \oplus G(k).$$

- The adversary then outputs a guess b' for b. The scheme is secure if the adversary's advantage

$$\epsilon(n) = \left| \Pr[b' = b] - \frac{1}{2} \right|$$

is negligible.

Reduction to the PRG Security

- We will now use the "reduction proof" paradigm that we studied in the previous part.
- The setup will be as follows:
  - We want to prove that (Security of PRG) $\Rightarrow$ (Secure encryption scheme). That is, the scheme is secure if the construction uses a PRG.
  - To achieve this, we will assume that we have an adversary $\mathcal{A}$ who can break the security of the encryption scheme.
  - Then we will show that this $\mathcal{A}$ can be used as a subroutine to construct an adversary $\mathcal{B}$ which breaks the security of the PRG itself.
  - This will prove that the scheme is secure if it uses a PRG.

Reduction to the PRG Security ... contd.

- Assume there exists an adversary $\mathcal{A}$ that can distinguish encryptions with a non-negligible advantage $\epsilon(n)$. (i.e. breaks the security of the encryption scheme).
- We construct a distinguisher $\mathcal{D}$ that uses $\mathcal{A}$ to distinguish the output of G from truly random strings. (i.e. breaks the security of the PRG)

- The distinguisher $\mathcal{D}$ is given a string w of length $\ell(n)$ which is either $G(k)$ for a random k or a truly random string.
- $\mathcal{D}$ will call $\mathcal{A}$ as a subroutine. But $\mathcal{A}$ works for an encryption scheme, not a PRG. Therefore, $\mathcal{D}$ will setup an instance of an encryption scheme first, and then use $\mathcal{A}$ internally.

Constructing the Distinguisher $\mathcal{D}$

1. $\mathcal{D}$ receives $w \in \{0,1\}^{\ell(n)}$.
2. $\mathcal{D}$ chooses two messages $m_0, m_1 \in \{0,1\}^{\ell(n)}$ (these can be chosen arbitrarily).
3. It then simulates the IND-CPA challenger by computing:

   $$c = m_b \oplus w,$$

   where $b \in \{0,1\}$ is chosen uniformly at random.
4. $\mathcal{D}$ gives c, m_0, m_1 to the adversary $\mathcal{A}$ and receives a guess b'.
5. If $b' = b$, then $\mathcal{D}$ outputs 1 (i.e., it believes w was pseudorandom); otherwise, it outputs 0 (believing that w was random).

Analysis of $\mathcal{D}$

- **Case 1:** $w = G(k)$ for some random key k.
  In this case, $c = m_b \oplus G(k)$ is a valid encryption of m_b.
  Hence, by the assumption on $\mathcal{A}$:

$$\Pr[\mathcal{D} \text{ outputs } 1] = \Pr[b' = b] = \frac{1}{2} + \epsilon(n).$$

- **Case 2:** w is truly random.
  In this case, c is independent of m_b (since w is uniform).
  Therefore, no matter what $\mathcal{A}$ does:

$$\Pr[\mathcal{D} \text{ outputs } 1] = \frac{1}{2}.$$

Conclusion: $\mathcal{D}$ distinguishes between $G(k)$ and a truly random string with advantage $\epsilon(n)$, which is non-negligible. This contradicts the security of the PRG.

Conclusion

- We constructed an encryption scheme based on a PRG where the ciphertext is computed as $c = m \oplus G(k)$.
- Under the assumption that G is a secure PRG, the encryption scheme is IND-CPA secure.
- The proof uses a reduction: any adversary breaking the encryption scheme would imply a distinguisher for the PRG, contradicting its assumed pseudorandomness.

Example 1: Iterative Expansion

- **Construction:** Given a $G : \{0,1\}^n \to \{0,1\}^{2n}$, construct $G_1 : \{0,1\}^n \to \{0,1\}^{3n}$ by iterating G.
- **Idea:** Use the output of G to generate a new seed and concatenate the outputs.
- **Procedure:**
  1. Let $s_0 \in \{0,1\}^n$.
  2. Compute $G(s_0) = x_0 \,\|\, x_1$ where $x_0, x_1 \in \{0,1\}^n$.
  3. Compute $G(x_1) = x_3 \,\|\, x_4$ where $x_3, x_4 \in \{0,1\}^n$.
  4. Output $G_1(s) = x_0 \| x_3 \| x_4$.
- **Security:** Under the assumption that G is secure, the concatenated output is indistinguishable from random.
- Formal proof is given after a few slides. First, we would like to develop some intuition about the PRG's.

Example 2: XOR of Two Independent PRG Outputs

- **Construction:** Define $G_2(s_1, s_2) = G(s_1) \oplus G(s_2)$, where s_1 and s_2 are independent seeds.
- **Idea:** The XOR of two independent pseudorandom strings is pseudorandom.
- **Security Argument:**
  - Since both $G(s_1)$ and $G(s_2)$ are indistinguishable from random, their XOR is also indistinguishable from a random string.
  - Any efficient distinguisher for G_2 would contradict the security of G.

Example 3: Concatenating with a Constant

- **Construction:** Define $H_1(s) = G(s) \| 0^{\ell(n)}$ (concatenate $G(s)$ with a fixed string of zeros).
- **Problem:**
  - The constant portion $0^{\ell(n)}$ is easily identifiable and not random.
  - A distinguisher can check whether the last $\ell(n)$ bits are all zeros, which occurs with probability 1 in $H_1(s)$ but with negligible probability for a truly random string.
- **Conclusion:** H_1 is insecure as a PRG.

Example 4: Repeating the Output

- **Construction:** Define $H_2(s) = G(s) \| G(s)$ (concatenate the same PRG output twice).
- **Problem:**
  - The repetition in $H_2(s)$ creates a structural pattern.
  - A distinguisher can check if the first half of the output is identical to the second half. For $H_2(s)$ this check always passes, while for a truly random string it passes only with negligible probability.
- **Conclusion:** H_2 is insecure since the redundancy is easily exploitable.

Part-4: The hybrid argument

What is a Hybrid Argument?

- A **hybrid argument** is a proof technique used to show that two distributions (or cryptographic games) are indistinguishable.
- The idea is to define a series of intermediate distributions (hybrids) between the real and ideal distributions.
- If each consecutive pair of hybrids is indistinguishable, then the first and the last are also indistinguishable.

How to Construct Hybrid Distributions

- Identify the original distribution/game H_0 and the target (ideal) distribution/game H_k.
- Define a sequence of hybrids $\mathsf{H}_0, \mathsf{H}_1, \ldots, \mathsf{H}_k$ such that:

$$\mathsf{H}_0 \approx \mathsf{H}_1 \approx \cdots \approx \mathsf{H}_k.$$

- Prove that for each i the difference between H_i and H_{i+1} is negligible or zero.

Example: PRG-Based Encryption Scheme

- Consider an encryption scheme based on a pseudorandom generator (PRG) G.
- Let the encryption of a message m be:

$$c = m \oplus G(k),$$

  where k is a secret key.
- To prove IND-CPA security, we can use a hybrid argument:
  - H_0: $c = m \oplus G(k)$ (real encryption).
  - H_1: $c = m \oplus r$ where r is chosen uniformly at random.
- If $G(k)$ is indistinguishable from random, then $\mathrm{H}_0 \approx \mathrm{H}_1$. Since $m \oplus r$ is also uniformly random (for fixed m), the scheme is secure.

Analysis and Transitivity of Indistinguishability

- The hybrid argument relies on the **transitivity** of indistinguishability:

$$\text{If } \mathsf{H}_0 \approx \mathsf{H}_1 \text{ and } \mathsf{H}_1 \approx \mathsf{H}_2, \text{ then } \mathsf{H}_0 \approx \mathsf{H}_2.$$

- By breaking a complex proof into smaller hybrid steps, we can focus on proving that each small change is undetectable.
- The overall security follows from the fact that a polynomial-time adversary cannot distinguish between the endpoints of the hybrid sequence.

Summary

- The hybrid argument is a powerful and versatile technique in cryptographic proofs.
- It simplifies the task of proving indistinguishability by bridging the gap between two distributions through intermediate steps.
- This technique is widely used in security proofs for encryption schemes, digital signatures, and other cryptographic primitives.

Coming back to Example 1: The iterative PRG

Given

A secure pseudorandom generator $G : \{0,1\}^n \to \{0,1\}^{2n}$. For any seed $s \in \{0,1\}^n$, write:

$$G(s) = x_1 \| x_2,$$

where $x_1, x_2 \in \{0,1\}^n$.

Further Computation

Compute:

$$G(x_2) = x_3 \| x_4, \quad \text{with } x_3, x_4 \in \{0,1\}^n.$$

Definition of G_1

Define the $3n$-bit PRG:

$$G_1(s) = x_1 \| x_3 \| x_4.$$

Security Goal

- We wish to show that if G is a secure PRG, then the function $G_1 : \{0,1\}^n \to \{0,1\}^{3n}$ is also a secure PRG.
- That is, for any polynomial-time distinguisher $\mathcal{A}$, the advantage in distinguishing $G_1(s)$ from a uniformly random string in $\{0,1\}^{3n}$ is negligible.

Overview of the Hybrid Argument

- We define three hybrids:
  - H_0: The real output $G_1(s) = x_1 \| x_3 \| x_4$.
  - H_1: Replace x_1 by a uniform random string $r_1 \in \{0,1\}^n$; output $r_1 \| x_3 \| x_4$.
  - H_2: Replace $x_3 \| x_4$ by a uniform random string $r_2 \in \{0,1\}^{2n}$; output $r_1 \| r_2$.
- Finally, note that H_2 is distributed uniformly over $\{0,1\}^{3n}$.

Hybrid H_0 and H_1

- We take H_0 as the real construction and H_2 as a construction which produces uniformly random strings.
- H_0: $G_1(s) = x_1 \| x_3 \| x_4$, where

$$G(s) = x_1 \| x_2 \quad \text{and} \quad G(x_2) = x_3 \| x_4.$$

- H_1: Replace x_1 with a uniformly random $r_1 \in \{0,1\}^n$ but keep x_3 and x_4 as before:

$$H_1 = r_1 \| x_3 \| x_4.$$

- **Claim:** No PPT distinguisher can distinguish H_0 from H_1 with non-negligible advantage, otherwise we could break the pseudorandomness of the first n-bit block of $G(s)$.

Hybrid H_1 and H_2

- H_1: $r_1 \| x_3 \| x_4$, where r_1 is uniformly random and $x_3 \| x_4 = G(x_2)$ is pseudorandom.
- H_2: Replace $x_3 \| x_4$ with a uniformly random string $r_2 \in \{0,1\}^{2n}$:
$$\mathsf{H}_2 = r_1 \| r_2.$$
- **Claim:** If there were a distinguisher that could distinguish H_1 from H_2 with non-negligible advantage, then we could break the security of G on input x_2.

Concluding the Hybrid Argument

- By the transitivity of indistinguishability:

$$\mathsf{H}_0 \approx \mathsf{H}_1 \quad \text{and} \quad \mathsf{H}_1 \approx \mathsf{H}_2.$$

- Since H_2 is uniformly distributed in $\{0,1\}^{3n}$, it follows that the output of $G_1(s)$ is indistinguishable from uniform.
- Therefore, G_1 is a secure PRG.
- Now, we formalize this intuition into a formal proof by filling the missing details about the adversary.

Reduction: From a Distinguisher for G_1 to a Breaker for G

- Suppose there exists a PPT distinguisher $\mathcal{A}$ with non-negligible advantage $\epsilon(n)$ in distinguishing $G_1(s)$ from a uniform string in $\{0,1\}^{3n}$.
- We construct a PPT algorithm $\mathcal{B}$ that breaks the pseudorandomness of G by distinguishing either:
  1. The first n bits x_1 in $G(s) = x_1 \| x_2$, or
  2. The output $G(x_2) = x_3 \| x_4$.
- $\mathcal{B}$ uses $\mathcal{A}$ as a subroutine in the corresponding hybrid transition.

Case 1: If $\mathcal{A}$ distinguishes H_0 from H_1, then $\mathcal{B}$ distinguishes the first block x_1 of $G(s)$ from uniform.

Case 2: If $\mathcal{A}$ distinguishes H_1 from H_2, then $\mathcal{B}$ distinguishes the output $G(x_2)$ from uniform.

Conclusion of the Reduction Proof

- In either case, a non-negligible advantage for $\mathcal{A}$ would yield a non-negligible advantage for $\mathcal{B}$ in breaking the security of G.
- Since G is assumed secure, such a PPT algorithm $\mathcal{A}$ cannot exist.
- Therefore, G_1 is a secure PRG.

Summary and Final Remarks

- We constructed a $3n$-bit PRG $G_1(s) = x_1 \| x_3 \| x_4$ using two calls to a secure PRG G.
- A hybrid argument was employed, replacing portions of the output with truly random bits step by step.
- The reduction shows that any advantage in distinguishing G_1 from uniform implies an advantage against G, contradicting its security.
- Hence, G_1 is secure.

Part-5: The security of PRF based encryption

Encryption Scheme Overview

- **Setup:** Let $f : \mathcal{K} \times \mathcal{R} \to \{0,1\}^n$ be a secure pseudorandom function (PRF).
- **Encryption:** To encrypt a message $m \in \{0,1\}^n$:
  - Choose a random $r \in \mathcal{R}$ (with $\mathcal{R}$ typically $\{0,1\}^n$).
  - Compute $c = f(k,r) \oplus m$.
  - Output ciphertext (r,c).
- **Decryption:** Given ciphertext (r,c) and key k:
  - Recover $m = c \oplus f(k,r)$.

IND-CPA Security

- The goal is to prove that the above encryption scheme is IND-CPA secure.
- Informally, no polynomial-time adversary can distinguish between the encryptions of any two chosen messages.
- We will prove that if f is a secure PRF, then the encryption scheme is IND-CPA secure.

Hybrid Argument Overview

- We use a hybrid argument to bridge the real encryption scheme with an ideal scheme that is perfectly secure.
- The key idea is to replace the PRF $f(k, \cdot)$ with a truly random function.
- If an adversary could distinguish the real scheme from the ideal one, then we could build a distinguisher for the PRF.

Hybrid Definitions

- Hybrid_0: The real encryption scheme.

$$\text{Enc}_k(m) = \big(r,\ f(k,r) \oplus m\big)$$

- Hybrid_1: Replace $f(k,\cdot)$ with a truly random function $F(\cdot)$. Thus, the encryption becomes:

$$\text{Enc}'(m) = \big(r,\ F(r) \oplus m\big).$$

Indistinguishability Between Hybrids

- By the security of the PRF f, no PPT distinguisher can tell apart $f(k, \cdot)$ from a truly random function $F(\cdot)$.
- Hence, the outputs in Hybrid_0 and Hybrid_1 are indistinguishable.
- Formally, if there exists an adversary $\mathcal{A}$ that can distinguish between these two hybrids with non-negligible advantage, then we can construct a distinguisher for the PRF f.

Security of the Ideal Scheme

- Consider Hybrid_1:
$$(r, F(r) \oplus m).$$
- Given that F is a truly random function and r is uniformly random:
  - $F(r)$ is uniformly random.
  - $F(r) \oplus m$ is a one-time pad encryption of m.
- Therefore, Hybrid_1 provides perfect secrecy (i.e., it is IND-CPA secure).

Concluding the Hybrid Argument

- We have:
$$\mathsf{Hybrid}_0 \approx \mathsf{Hybrid}_1,$$
and Hybrid_1 is perfectly secure.
- Thus, the real encryption scheme is IND-CPA secure.
- If an adversary $\mathcal{A}$ were to break the encryption scheme, then it would also break the PRF security of f, contradicting the assumption that f is a secure PRF.

Summary

- We defined an encryption scheme using a PRF f as:

$$E_k(m) = \big(r,\ f(k,r) \oplus m\big).$$

- The proof uses a hybrid argument, replacing f with a truly random function.
- The ideal scheme is equivalent to a one-time pad encryption, which is perfectly secure.
- Hence, the security of the PRF implies the IND-CPA security of the encryption scheme.

Further Reading

- Jonathan Katz and Yehuda Lindell, *Introduction to Modern Cryptography*.