CS6222 Cryptography

Topic: PRGs
Lecturer: Wei-Kai Lin (TA: Arup Sarker)

Date: Sep 10, 2024
Scriber: Adarsha Poudel

---

1 Pseudorandom Generators (PRGs)

PRGs are foundational in cryptography, as they are used to construct various cryptographic primitives such as pseudorandom functions (PRFs) and symmetric key encryption schemes. The expanding property ensures that the generator can produce more pseudorandom bits than the input, which is crucial for many cryptographic applications.

Additionally, PRGs enhance the efficiency of the one-time pad by reducing the required key length. In a traditional one-time pad, the key must be as long as the message, but with a PRG, a shorter seed (key) can be expanded to the necessary length, making it feasible to use a key that is a fraction of the input length, significantly improving practical usability.

1.1 Definition of PRG

A function $g : \{0,1\}^* \to \{0,1\}^*$ is a *Pseudorandom Generator (PRG)* if it satisfies the following properties:

- **Efficiently Computable:** The function g is efficiently computable, meaning it can be computed in polynomial time relative to the size of its input.
- **Expanding:** The PRG expands its input, i.e., for every input $x \in \{0,1\}^*$, the length of the output is strictly greater than the length of the input:

  $$|g(x)| = l(n) > |x|$$

  where $l(n)$ denotes the length of the output, and $n = |x|$.
- **Pseudorandom:** The output $g(x)$ is pseudorandom, meaning it is computationally indistinguishable from a truly random string of the same length. Formally, for every probabilistic polynomial-time algorithm A, the probability that A distinguishes $g(x)$ from a uniform random string of length $l(n)$ is negligible:

  $$\left|\Pr[A(g(x)) = 1] - \Pr[A(U_{l(n)}) = 1]\right| < \epsilon(n)$$

  where $U_{l(n)}$ denotes the uniform distribution over strings of length $l(n)$, and $\epsilon(n) = \frac{1}{p(n)}$ is a negligible function for some polynomial $p(n)$.

1.2 Theorem: PRG Expansion

Theorem: Suppose g is a PRG such that $|g(x)| = |x| + 1$. Then, there exists a PRG G such that $|G(x)| = 2|x|$.

1.2.1 G(x) generation

The idea behind the construction of G from $g(x)$ is as follows:

Given an initial seed s of length n, the generator g can be used to obtain $n+1$ pseudorandom bits. One of the $n+1$ bits may be output, and the remaining n bits can be used once again as a seed for g. The reason these n bits can be reused as a seed is that they are pseudorandom. This process can be repeated $2n$ times to achieve the desired length.

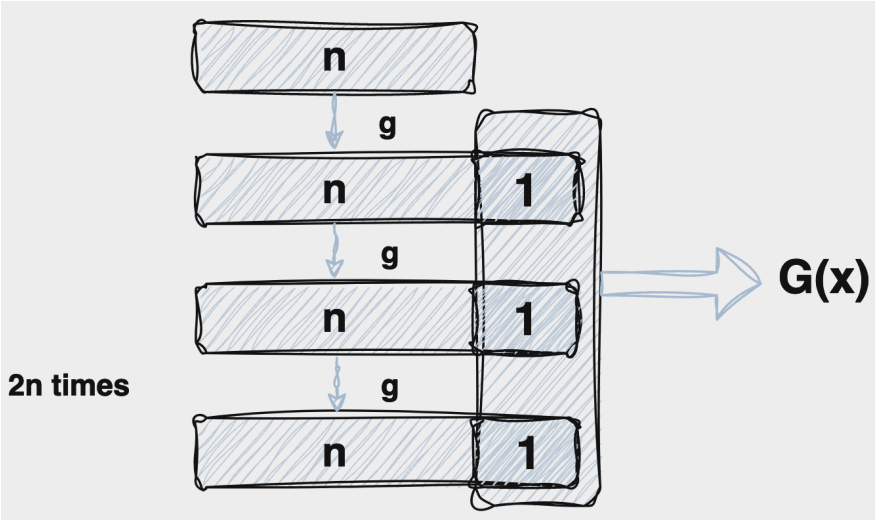

Figure 1: Construction of $G(x)$ by applying g $2n$ times to achieve the desired expansion.

1.2.2 Security Proof Using Hybrid Lemma

Step 1: Assumption for Contradiction Assume, for contradiction, that G is not a pseudorandom generator. This implies that there exists a non-uniform probabilistic polynomial-time (NUPPT) adversary A and a polynomial $p(n)$ such that for $n \in \mathbb{N}$, A can distinguish the output of $G(x)$ from a uniform distribution U_{2n} with probability greater than $\frac{1}{p(n)}$. Formally, we have:

$$|\Pr[A(G(x)) = 1] - \Pr[A(U_{2n}) = 1]| > \frac{1}{p(n)}.$$

This inequality indicates that A can distinguish between the pseudorandom output of G and a truly random string of length $2n$ with a non-negligible advantage.

Step 2: Construction of Adversary B To leverage the distinguishing capability of A, we construct a new adversary B that uses A to challenge the pseudorandomness of g.

- **Objective of B**: Adversary B aims to distinguish the output of g from a truly random string, thus testing g's pseudorandomness.

- **Operation of B**:
  1. B receives an input $t \in \{0,1\}^{n+1}$, which is either generated by $g(x)$ for some $x \in \{0,1\}^n$ or is uniformly random.
  2. B simulates a sequence that resembles the hybrids H_i (defined below) by embedding t into specific positions corresponding to the hybrids.
  3. B then runs A on this crafted input, effectively "hardwiring" the bits of t into the sequence in a manner that matches H_i.

Step 3: Hybrid Distributions We define a sequence of hybrid distributions $H_0, H_1, \ldots, H_{2n}$ to facilitate the application of the Hybrid Lemma:

- H_0: Represents the fully random distribution U_{2n}.
- H_{2n}: Corresponds to the distribution of the output of $G(x)$.
- **Intermediate Hybrids H_i**:
  - Each H_i is constructed such that the first i bits are truly random, while the remaining $2n - i$ bits are produced by iteratively applying g, as in the construction of G.

Step 4: Proving Indistinguishability of Consecutive Hybrids $H_i \approx H_{i+1}$

1. **Indistinguishability Argument**:
   - For each pair (H_i, H_{i+1}), the only difference lies in the replacement of one bit: H_i has the i-th bit as pseudorandom, while in H_{i+1}, this bit is replaced by a truly random bit.
   - The goal is to demonstrate that no efficient adversary can distinguish between H_i and H_{i+1}.
2. **Using Adversary B**:
   - Construct adversary B so that it can distinguish whether a given input t is more consistent with the pseudorandom or random nature of the hybrids.
   - B utilizes A by preparing inputs that simulate H_i and H_{i+1} and feeds these to A.
   - If A successfully distinguishes between these hybrids, then B exploits this to distinguish the output of g from truly random, violating g's pseudorandomness assumption.
3. **Formal Indistinguishability Proof**:
   - For any efficient distinguisher D:

$$|\Pr[D(H_i) = 1] - \Pr[D(H_{i+1}) = 1]| \leq \epsilon(n),$$

     where $\epsilon(n)$ is negligible.
   - Summing these negligible differences across all $2n$ hybrids results in a cumulative difference of $2n \cdot \epsilon(n)$, which remains negligible.

Step 5: Applying the Hybrid Lemma Applying the Hybrid Lemma, we conclude:

$$H_0 \approx H_1 \approx \ldots \approx H_{2n}.$$

Thus, H_0, which is fully random, is indistinguishable from H_{2n}, the output distribution of $G(x)$. Therefore, $G(x)$ is indistinguishable from a truly random string.

Step 6: Reduction Using B and Concluding the Contradiction

1. **Reduction to g's Indistinguishability**:

   - To directly challenge g's pseudorandomness, we construct B to leverage A's distinguishing power. Specifically, B is designed to take an input $t \in \{0,1\}^{n+1}$ (either from $g(x)$ or uniformly random) and simulate inputs for A that reflect the intermediate hybrid distributions H_i.
   - Mathematically, B operates as follows:

     $$\text{Given } t \in \{0,1\}^{n+1}, B \text{ constructs an input } \tilde{x} \text{ for } A \text{ such that:}$$

     - If $t = g(x)$ for some $x \in \{0,1\}^n$, then $\tilde{x}$ is structured to resemble a hybrid distribution close to H_j. - If t is uniformly random, $\tilde{x}$ mimics the structure of H_{j+1}.
   - The reduction uses the property:

     $$|\Pr[A(H_j) = 1] - \Pr[A(H_{j+1}) = 1]| \geq \frac{1}{p(n)}.$$

     If A can distinguish $G(x)$, then the above expression implies that B can distinguish between $g(x)$ and a random string of the same length, directly challenging the assumption that g is a PRG.
   - **Formal Reduction Analysis**:
     - Suppose $t = g(x)$. Then:

       $$\Pr[B(t) = 1] = \Pr[A(H_j) = 1].$$

     - If t is uniformly random:

       $$\Pr[B(t) = 1] = \Pr[A(H_{j+1}) = 1].$$

     - Hence:

       $$|\Pr[B(g(x)) = 1] - \Pr[B(U_{n+1}) = 1]| = |\Pr[A(H_j) = 1] - \Pr[A(H_{j+1}) = 1]| > \frac{1}{p(n)}.$$

     - This inequality indicates that B distinguishes $g(x)$ from random with a non-negligible advantage, contradicting the definition of g being a PRG.

2. **Final Contradiction**:

   - Since B successfully distinguishes $g(x)$ from a uniformly random string with a probability greater than $\frac{1}{p(n)}$, we conclude that g cannot be pseudorandom.
   - This directly contradicts our initial assumption that g is a PRG, which implies that the only consistent conclusion is that the assumption of G not being pseudorandom is false.
   - Therefore, G must indeed be a pseudorandom generator, as any adversary's success in distinguishing it from uniform would imply a breach in g's pseudorandomness, which is not the case.