Purdue CS555: Cryptography
Lecture 5 Scribe Notes

Instructor: Hanshen Xiao
Teaching Assistant: Justin He

Fall 2025

Recap from Previous Lecture

- Preliminaries of Computational Complexity
- Model of Computation (Turing machines, RAM, circuits)
- Complexity Classes (P, NP, NPC, BPP)
- Reduction
- One-way Functions
- Construction of PRG from One-way Permutations
- Average and Worst Case Hardness

Topics Covered in This Lecture

1. Implications of One-way Functions
2. Construction of PRG (continued)
3. Construction of Hardcore Bit: Goldreich-Levin Theorem

1 Review: Key Concepts

1.1 Average-Case vs Worst-Case Hardness

Average-case hardness: For random secrets (with entropy) that the user fully controls, we can talk about average-case hardness, such as one-way functions:

$$\Pr[x \leftarrow \{0,1\}^n; y = F_n(x); x' \leftarrow A(1^n, y) : y = F_n(x')] \leq \mu(n)$$

Worst-case hardness: The meaning depends on context:

- *IND-CPA:* The adversary can arbitrarily select the challenge; worst-case indistinguishability holds for any input candidate pair
- *Lattice problems:* There exists some case that is hard to break

1.2 One-way Functions Imply P $\neq$ NP

Theorem 1. *If one-way functions exist, then $P \neq NP$.*

Proof. Assume f is one-way. Consider the language L of pairs (x^*, y) such that x^* is a prefix of some x satisfying $f(x) = y$.

Clearly $L \in NP$ (given witness x, we can verify in polynomial time).

If $P = NP$, we could use a polynomial-time decider D for L to invert y by finding the pre-image bit by bit: query (x_1, y), (x_1x_2, y), etc. This would contradict the one-wayness of f. $\square$

1.3 Hardcore Predicate (Recap)

Definition 1 (Hardcore Predicate)**.** *For any function family $F : \{0,1\}^n \to \{0,1\}^m$, a function $B : \{0,1\}^n \to \{0,1\}$ is a hardcore predicate if for every p.p.t. adversary A, there is a negligible function μ such that:*

$$\Pr[x \leftarrow \{0,1\}^n; y = F(x) : A(y) = B(x)] \leq \frac{1}{2} + \mu(n)$$

1.4 PRG Construction from OWP (Recap)

Theorem 2. *Let F be a one-way permutation and B an associated hardcore predicate for F. Define:*

$$G(x) = F(x) \| B(x)$$

Then G is a PRG (stretching input by one bit).

Proof Sketch. By next-bit unpredictability. Assume for contradiction that G is not a PRG. Then there exists a next-bit predictor D and polynomial p such that:

$$\Pr[x \leftarrow \{0,1\}^n; y = G(x) : D(y_{1\ldots n}) = y_{n+1}] \geq \frac{1}{2} + \frac{1}{p(n)}$$

Since $G(x) = F(x) \| B(x)$, this means:

$$\Pr[x \leftarrow \{0,1\}^n : D(F(x)) = B(x)] \geq \frac{1}{2} + \frac{1}{p(n)}$$

Therefore, D contradicts the hardcore property of B. □

2 The Goldreich-Levin Theorem

The key question: Does every one-way function have a hardcore predicate? The Goldreich-Levin (GL) theorem provides a powerful affirmative answer.

2.1 Statement of GL Theorem

Theorem 3 (Goldreich-Levin)**.** *Let $\{B_r : \{0,1\}^n \to \{0,1\}\}$ where*

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

be a collection of predicates (one for each r). Then, a random B_r is hardcore for every one-way function F. That is, for every one-way function F, every p.p.t. adversary A, there exists a negligible function μ such that:

$$\Pr[x \leftarrow \{0,1\}^n; r \leftarrow \{0,1\}^n : A(F(x), r) = B_r(x)] \leq \frac{1}{2} + \mu(n)$$

2.2 Alternative Interpretation

The GL theorem states that for every one-way function/permutation F, there is a related one-way function/permutation

$$F'(x, r) = (F(x), r)$$

which has a *deterministic* hardcore predicate. In particular, the predicate

$$B(x, r) = \langle r, x \rangle \bmod 2$$

is hardcore for F'.

This formulation is sufficient to construct PRGs from any one-way permutation.

3 Proof of the Goldreich-Levin Theorem

3.1 Setup and Strategy

Assumption (for contradiction): There exists a predictor P and polynomial p such that:

$$\Pr[x \leftarrow \{0,1\}^n; r \leftarrow \{0,1\}^n : P(F(x), r) = \langle r, x\rangle] \geq \frac{1}{2} + \frac{1}{p(n)}$$

Goal: Construct an inverter A for F such that:

$$\Pr[x \leftarrow \{0,1\}^n : A(F(x)) = x' : F(x') = F(x)] \geq \frac{1}{p'(n)}$$

for some polynomial p'.

3.2 Warm-up: Perfect Predictor

First, let's make our lives easier by assuming a *perfect* predictor P:

$$\Pr[x \leftarrow \{0,1\}^n; r \leftarrow \{0,1\}^n : P(F(x), r) = \langle r, x\rangle] = 1$$

Inverter Strategy: On input $y = F(x)$, run predictor P on inputs $(y, e_1), (y, e_2), \ldots, (y, e_n)$ where e_i are the unit vectors:

$$e_1 = 100\ldots0, \quad e_2 = 010\ldots0, \quad \ldots, \quad e_n = 00\ldots01$$

Since P is perfect, it returns $\langle e_i, x\rangle = x_i$, the i-th bit of x on the i-th invocation. Thus, we recover all of x.

3.3 Pretty Good Predictor and Linearity

Now assume a "pretty good" predictor:

$$\Pr[x \leftarrow \{0,1\}^n; r \leftarrow \{0,1\}^n : P(F(x), r) = \langle r, x\rangle] \geq \frac{3}{4} + \frac{1}{p(n)}$$

Claim 1 (Averaging Argument)**.** *For at least a $\frac{1}{2p(n)}$ fraction of the x,*

$$\Pr[r \leftarrow \{0,1\}^n : P(F(x), r) = \langle r, x\rangle] \geq \frac{3}{4} + \frac{1}{2p(n)}$$

Call these the "good x".

Key Idea - Linearity: Pick a random r and ask P to tell us $\langle r, x\rangle$ and $\langle r + e_i, x\rangle$. Subtract (XOR) the two answers to get:

$$\langle r + e_i, x\rangle - \langle r, x\rangle = \langle e_i, x\rangle = x_i$$

Analysis:

$$\begin{aligned}
\Pr[\text{we compute } x_i \text{ correctly}] &\geq \Pr[P \text{ predicts both } \langle r, x\rangle \text{ and } \langle r + e_i, x\rangle \text{ correctly}] \\
&= 1 - \Pr[P \text{ predicts } \langle r, x\rangle \text{ or } \langle r + e_i, x\rangle \text{ wrong}] \\
&\geq 1 - (\Pr[P \text{ predicts } \langle r, x\rangle \text{ wrong}] \\
&\qquad + \Pr[P \text{ predicts } \langle r + e_i, x\rangle \text{ wrong}]) \\
&\geq 1 - 2 \cdot \left(\frac{1}{4} - \frac{1}{2p(n)}\right) = \frac{1}{2} + \frac{1}{p(n)}
\end{aligned}$$

Inverter Algorithm:

1. Repeat $\log n \cdot p(n)$ times: pick random r, query $P(F(x), r)$ and $P(F(x), r + e_i)$, XOR to get a guess for x_i
2. Take the majority of all guesses as x_i
3. Repeat for each $i \in \{1, 2, \ldots, n\}$
4. Output the concatenation of all x_i

Analysis uses Chernoff bound and union bound.

3.4 The Challenge: Weak Predictor

The Real Assumption: The predictor satisfies only:

$$\Pr[x \leftarrow \{0,1\}^n; r \leftarrow \{0,1\}^n : P(F(x), r) = \langle r, x\rangle] \geq \frac{1}{2} + \frac{1}{2p(n)}$$

After averaging, for $\geq \frac{1}{2p(n)}$ fraction of the x:

$$\Pr[r \leftarrow \{0,1\}^n : P(F(x), r) = \langle r, x\rangle] \geq \frac{1}{2} + \frac{1}{2p(n)}$$

Problem with Previous Approach: The union bound analysis breaks down because we can no longer guarantee that both $\langle r, x\rangle$ and $\langle r + e_i, x\rangle$ are predicted correctly with high probability.

3.5 The Rackoff Trick: Using Advice

Key Insight (attributed to Charlie Rackoff): Imagine we have an oracle that tells us $\langle r, x\rangle$ for free. Then, we can:

1. Pick random r, ask oracle for $\langle r, x\rangle$
2. Ask P to predict $\langle r + e_i, x\rangle$
3. XOR the two to get x_i

Analysis:

$$\Pr[\text{we compute } x_i \text{ correctly}] \geq \Pr[P \text{ predicts } \langle r + e_i, x\rangle \text{ correctly}] \geq \frac{1}{2} + \frac{1}{2p(n)}$$

The Challenge: We don't have such an oracle. Solution: *guess* the values.

3.6 Parsimony in Guessing

Instead of guessing $\langle r, x\rangle$ for every r we use, we can be more parsimonious.

Strategy:

1. Pick random "seed vectors" $s_1, \ldots, s_{\log(m+1)}$ where $m = O(n \log n \cdot (p(n))^2)$
2. Guess $c_j = \langle s_j, x\rangle$ for all $j \in \{1, \ldots, \log(m+1)\}$
3. The probability that all guesses are correct is $\frac{1}{2^{\log(m+1)}} = \frac{1}{m+1}$

Key Construction: From the seed vectors, generate many r_i values.
Let $T_1, \ldots, T_m$ denote all possible non-empty subsets of $\{1, 2, \ldots, \log(m+1)\}$. Define:

$$r_i = \bigoplus_{j \in T_i} s_j \quad \text{and} \quad b_i = \bigoplus_{j \in T_i} c_j$$

Key Observation: If the guesses $c_1, \ldots, c_{\log(m+1)}$ are all correct, then so are $b_1, \ldots, b_m$. This is because:

$$b_i = \bigoplus_{j \in T_i} c_j = \bigoplus_{j \in T_i} \langle s_j, x\rangle = \left\langle \bigoplus_{j \in T_i} s_j, x \right\rangle = \langle r_i, x\rangle$$

3.7 The Complete Inverter Algorithm

Algorithm 1 OWF Inverter A

Input: $y = F(x)$ for unknown x
Generate random seed vectors $s_1, \ldots, s_{\log(m+1)}$
Generate random bits $c_1, \ldots, c_{\log(m+1)}$ (guesses for $\langle s_j, x\rangle$)
Derive $r_1, \ldots, r_m$ and bits $b_1, \ldots, b_m$ as: $r_i = \bigoplus_{j \in T_i} s_j$, $b_i = \bigoplus_{j \in T_i} c_j$
for $i = 1$ to n **do**
    Repeat $100n(p(n))^2$ times:
        Pick a random index $\ell \in \{1, \ldots, m\}$
        Ask P to predict $\langle r_\ell + e_i, x\rangle$, getting answer a
        Compute guess for x_i as: $g = a \oplus b_\ell$
    Compute majority of all guesses to determine x_i
end for
Output: $x = x_1 x_2 \cdots x_n$

3.8 Analysis of the Inverter

Conditioning: Let's condition on the guesses $c_1, \ldots, c_{\log(m+1)}$ being all correct.

Main Issue: The r_i values are *not* independent, so we cannot directly apply Chernoff bound.

Key Observation: The r_i values are *pairwise independent*. Therefore, we can apply Chebyshev's inequality.

3.8.1 Single Bit Analysis

Fix a bit position i. The probability that a single iteration of the inner loop gives the correct x_i is at least $\frac{1}{2} + \frac{1}{2p(n)}$.

Let E_ℓ be the event that the ℓ-th iteration gives the correct x_i.

The majority decision is correct if the number of events E_ℓ that occur is at least $\frac{m}{2} = 50n(p(n))^2$.

Expected number of correct guesses:

$$\mathbb{E}[\# \text{ correct}] = \left(\frac{1}{2} + \frac{1}{2p(n)}\right) \cdot 100n(p(n))^2 = 50n(p(n))^2 + 50np(n)$$

Variance (using pairwise independence):

$$\text{Var}[\# \text{ correct}] \approx \frac{1}{4} \cdot 100n(p(n))^2 = 25n(p(n))^2$$

By Chebyshev's inequality:

$$\begin{aligned}\Pr[\text{majority decision w.r.t. } x_i \text{ incorrect}] &\leq \frac{\text{Var}}{(\text{deviation})^2} \\ &\leq \frac{25n(p(n))^2}{(50np(n))^2} = \frac{1}{100n}\end{aligned}$$

By union bound over all n bits:

$$\Pr[\text{one of the } x_i \text{ is incorrect}] \leq n \cdot \frac{1}{100n} = \frac{1}{100}$$

Therefore, the inverter outputs the correct inverse with probability $p \geq 0.99$ (conditioned on guesses being correct and x being good).

3.9 Overall Success Probability

$$\begin{aligned}\Pr[\text{Inverter succeeds}] &\geq \Pr[\text{Inverter succeeds} \mid \text{all guesses correct, good } x] \\ &\quad \cdot \Pr[\text{all guesses correct}] \cdot \Pr[\text{good } x] \\ &= p \cdot \frac{1}{m+1} \cdot \frac{1}{2p(n)} \\ &= p \cdot \frac{1}{2n^2 p(n)^3}\end{aligned}$$

By our calculation, $p \geq 0.99$, so the inverter succeeds with non-negligible probability, contradicting the one-wayness of F. $\square$

Remark 1. We can also make the success probability $\approx \frac{1}{p(n)}$ by enumerating over all possible "guesses". Each guess results in a supposed inverse, but we can verify which is the actual inverse by checking if $F(x') = y$.

4 The Coding-Theoretic View

4.1 Hadamard Code Interpretation

The mapping $x \mapsto (\langle x, r \rangle)_{r \in \{0,1\}^n}$ can be viewed as a highly redundant, exponentially long encoding of x called the **Hadamard code**.

- $P(F(x), r)$ provides access to a *noisy* codeword
- The real proof = list-decoding algorithm for Hadamard code with error rate $\frac{1}{2} - \frac{1}{p(n)}$
- What we showed for the "pretty good predictor" = unique decoding algorithm for Hadamard code with error rate $\frac{1}{4} - \frac{1}{p(n)}$

4.2 General Connection: List-Decodable Codes

Framework (due to Impagliazzo and Sudan):

Let $x \to C(x)$ be an encoding.
Given: A corrupted codeword that is incorrect at $\frac{1}{2} - \epsilon$ fraction of locations.
List-decoder: Outputs a list $\{x_1, \ldots, x_m\}$ of possibilities for x.
Hardcore Predicate: Define $B_i(x) = C(x)_i$ (the i-th bit of the codeword).
How it works:

1. A hardcore-bit predictor gives access to a corrupted codeword
2. Run the list-decoder to get a list of possible inverses
3. Since the OWF is easy to compute, we can verify which candidates are actual inverses by checking if $F(x_j) = y$

This provides a general framework for constructing hardcore predicates from any list-decodable code!

5 Summary and Implications

5.1 Key Takeaways

- The Goldreich-Levin theorem shows that *every* one-way function has a hardcore predicate $\langle r, x \rangle$ for random r
- This means we can construct PRGs from *any* one-way permutation, not just those with known hardcore predicates
- The proof technique involves:
  - Averaging arguments to find "good" inputs
  - Exploiting linearity of inner products
  - Parsimonious guessing using pairwise independent sampling
  - Chebyshev's inequality for analysis
- The coding-theoretic view connects this to list-decoding of error-correcting codes
- This framework generalizes to other list-decodable codes (Impagliazzo-Sudan)

5.2 Significance

The GL theorem is one of the most important results in cryptography because:

1. It provides a *universal* hardcore predicate for all one-way functions
2. Combined with the OWP $\rightarrow$ PRG construction, it shows that one-way permutations are sufficient for constructing PRGs
3. The proof technique has influenced many subsequent results in cryptography and complexity theory
4. It demonstrates the power of combining probabilistic, algebraic, and coding-theoretic techniques