Pseudorandom Generators

Alexander Vakhitov

Saint Petersburg State University, Mathematics and Mechanics

2nd April 2005

Outline of Part I

1. Main Approach to Pseudorandom Generator Proofs
   - Already Introduced Concepts
   - Our Task
   - Computational Indistinguishability
   - Pseudorandom Functions
   - Using Pseudorandom Generator

Outline of Part II

2. Blum-Blum-Shub generator
   - One-Way Function for BBS
   - Hard Bit and the best of BBS

Outline of Part III

Part I

Main Approach

One-Way Function

- *One-way function* is used in the construction of pseudorandom generator.
- Informally, f is one-way if it is easy to compute but hard to invert.
- If $P = NP$, then there are no one-way functions
- It is not ever known if $P \neq NP$ implies there are one-way functions.

One-Way Function

Example

Examples of one-way functions

- Discrete logarithm problem (x^e mod n) for a large prime n
- Factoring a product of two large primes
- Nonnumber theoretic functions,including coding theory problems

One-Way Function

Definition

A function $f : \{0,1\}^* \to \{0,1\}^*$ is called one-way if hold:

1. easy to evaluate: There exist a polynomial-time algorithm computing $f(x)$ from every $x \in \{0,1\}^*$
2. hard to invert: For every probabilistic polynomial-time algorithm A, every polynomial p, and all sufficiently large n,

$$Pr[A(f(x), 1^n) \in f^{-1}(f(x))] < \frac{1}{p(n)}$$

where the probability is taken uniformly over all possible choices of $x \in \{0,1\}^n$ and all the possible outcomes of the internal coin tosses in A.

Hidden Bit

Definition

A polynomial-time computable predicate $b : \{0,1\}^* \rightarrow \{0,1\}$ is called a hard-core (hidden bit) of a function f if for every probabilistic polynomial-time algorithm A, every positive polynomial p, and all sufficiently large n,

$$Pr[A(f(x)) = b(x)] < \frac{1}{2} + \frac{1}{p(n)}$$

where the probability is taken uniformly over all possible choices of $x \in \{0,1\}^n$ and all the possible outcomes of the internal coin tosses in A.

Hiding Information

- There are 2 agents **A** and **B** exchanging with message m
- Shannon (1943) proved:
  fully secure encryption system can exist if the size of the secret information S which **A** *and* **B** *agree on prior is as large as the number of secret bits to be ever exchanged remotely using the encryption system.*

$$\mathbf{A} \overset{m}{\underset{S}{\Longrightarrow}} \mathbf{B}$$

Pseudorandom Generator Intuitively

Definition

Pseudorandom Generator is a deterministic program used to generate a long sequence of bit which look like random sequences, given as input a short random sequence (the input seed).

r truly random, G - PSRG, $\Rightarrow G(r)$ "looks like random" and $|G(r)| \gg |r|$

Way of Using in Cryptography

Example

A ···r···> **B**

$c = G(r) \oplus m$ **B**

A $\Longrightarrow$ c **B**

A $m = c \oplus G(r)$

Citation

Indistinguishable things are identical
(or should be considered as identical)
The Principle of Identity of Indiscernibles,
G.W.Leibnitz (1646-1714)
taken from:
Foundations of Cryptography - a Primer,
O. Goldreich

Computational Indistinguishability

Definition

We say that bit string sets $X = \{X_n\}_{n \in \mathbb{N}}$ and $Y = \{Y_n\}_{n \in \mathbb{N}}$ are computationally indistinguishable if for every probabilistic polynomial-time algorithm A, every polynomial p, and all sufficiently large n,

$$|Pr[A(X_n) = 1] - Pr[A(Y_n) = 1]| < \frac{1}{p(n)}$$

where the probabilities are taken over the relevant distribution (X or Y) and over the internal coin tosses of algorithm A.

Hybrid Argument Method

Method Construction

- Assume that we have multiple samples of distributions X and Y (that is, $\{\{X_n\}\}_m$ and $\{\{Y_n\}\}_m$ for $n, m \in \mathbb{N}$;
- Consider sequence of samples $H_i = \{X_1, \ldots, X_i, Y_{i+1} \ldots Y_s\}$ for some $s \in \mathbb{N}$ - length of a hybrid H_i;
- Distinguishing H_0 and H_s yields a procedure for distinguishing H_i from H_{i+1} for randomly chosen i (if D distinguishes X from Y, then it also distinguishes a pair of neighboring hybrids);
- Then, we can build distinguisher D' for a single sample (S), which choses i randomly, generates i samples $\{X_k\}$ from X and other samples $\{Y_k\}$ from Y, makes a sequence $\{X_1, \ldots, X_i, S, Y_1, \ldots\}$ and runs D on it.

Pseudorandom Generator

Definition

Let $l : \mathbb{N} \rightarrow \mathbb{N}$ satisfy $l(n) > n \forall n \in \mathbb{N}$. A pseudorandom generator, with stretch function l, is a (deterministic) polynomial-time algorithm G satisfying:

1. $\forall s \in \{0,1\}^*$, it holds that $|G(s)| = l(|s|)$
2. $\{G(U_n)\}_{n \in \mathbb{N}}$ and $\{U_{l(n)}\}_{n \in \mathbb{N}}$ are computationally indistinguishable, where U_m denotes the uniform distribution over $\{0,1\}^m$.

Pseudorandom Generator

Definition

Let $l : \mathbb{N} \to \mathbb{N}$ satisfy $l(n) > n \forall n \in \mathbb{N}$. A pseudorandom generator, with stretch function l, is a (deterministic) polynomial-time algorithm G satisfying:

1. $\forall s \in \{0,1\}^*$, it holds that $|G(s)| = l(|s|)$
2. $\{G(U_n)\}_{n\in\mathbb{N}}$ and $\{U_{l(n)}\}_{n\in\mathbb{N}}$ are computationally indistinguishable, where U_m denotes the uniform distribution over $\{0,1\}^m$.

Simple generator

Example

If we have a injective (one-to-one) one-way function $f : \{0,1\}^n \rightarrow \{0,1\}^{l_n}$ and $b : \{0,1\}^n \rightarrow \{0,1\}$ is a hidden bit of f then we can build a pseudorandom generator in a such way:

$$G(x) =< b(x), b(f(x)), b(f(f(x))), \ldots, b(f^{l(|x|)}(x)) >$$

Simple generator

Theorem

Following conditions are equivalent:

- The distribution X, in our case it is $\{G(U_n)\}_n \in \mathbb{N}$, is computationally indistinguishable from a uniform distribution on $\{U_{l(n)}\}_{n \in \mathbb{N}}$
- The distribution X is unpredictable in polynomial-time; no feasible algorithm, given a prefix of sequence, can guess the next bit with a sufficient advantage over $\frac{1}{2}$

Theorem proof

Theorem

Following conditions are equivalent:

1. The distribution X, is computationally indistinguishable from a uniform distribution
2. The distribution X is unpredictable in polynomial-time;

Proof

- Pseudorandomness implies polynomial-time unpredictability
- Let's prove the inverse:

Theorem proof

Theorem

Following conditions are equivalent:

1. The distribution X, is computationally indistinguishable from a uniform distribution
2. The distribution X is unpredictable in polynomial-time;

Proof

- Pseudorandomness implies polynomial-time unpredictability
- Let's prove the inverse:

Theorem proof

Proof of 2 $\Rightarrow$ 1

- Suppose that exists algorithm $A : |Pr[A(x) = 1] - Pr[G(x) = 1]| > \epsilon, \epsilon > 0$;
- Reverse $G'(s) = G(s)_{l(|s|),\dots,1} =< b(f^{l(|x|)}(x)), \dots, b(x) >$
- choose a random k, consider H_k is a *hybrid* built from G'(X) and $U_{l(n)}$ (then $G'(X) = H_n$ and $y = H_0$);
- Given $b(f^{l-1}(x)), \dots, b(f^{l-k}(x))$ A predicts $b(f^{l-k-1}(x))$
- x is chosen from U_n then given y=f(x) one can predict b(x) by invoking A on input $b(f^{k-1}(y)) \cdots b(y) = b(f^k(x)) \cdots b(f(x))$ which is polynomial-time computable from y.

Theorem 2

Pseudorandom generators exist iff one-way functions exist

Proof

Given a pseudorandom generator (stretching in a factor 2) we consider the function f(x,y)=G(x) and see that an algorithm which inverts f also distinguishes between $G(U_n)$ and U_{2n}

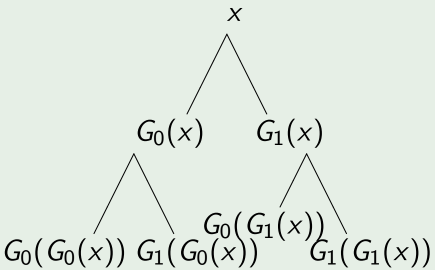
Construct a pseudorandom function

Definition

$f_s(x) : \{0,1\}^n \to \{0,1\}^n$ is pseudorandom function if it is infeasible to distinguish values of f_s for a random uniformly chosen s from values of truly random function $F : \{0,1\}^n \to \{0,1\}^n$

Example

PSRG G stretches in a factor of 2: $G(x) =< G_0(x), G_1(x) >$; then let's build a binary tree:



Ways to use Pseudorandom Generator

- randomized ciphers and stream ciphers
- randomized algorithms simulation and removing random steps from program execution
- computer modeling in general

Ways to use Pseudorandom Generator

- randomized ciphers and stream ciphers
- randomized algorithms simulation and removing random steps from program execution
- computer modeling in general

Ways to use Pseudorandom Generator

- randomized ciphers and stream ciphers
- randomized algorithms simulation and removing random steps from program execution
- computer modeling in general

Part II

BBS Generator

One-Way Function for BBS

Let's look at $f_{BBS}(x) = x^2 \mod n, n = pq$ for primes p and q congruent to 3 modulo 4.

Solving $a \equiv x^2 \mod n$

$a \equiv x^2 \equiv (-x)^2 \mod p$, and
$a \equiv (-y)^2 \equiv y^2 \mod q$
Then there are four solutions for $a \equiv z^2 \mod n$ $(\pm cx \pm dy)$, where

$$c \equiv \begin{cases} 1 \mod p \\ 0 \mod q \end{cases} \quad d \equiv \begin{cases} 1 \mod q \\ 0 \mod p \end{cases}$$

One-Way Function for BBS

Squaring on $\mathbb{Z}_{n=pq}$ where $p \equiv q \equiv 3 \bmod 4$
$a^{p-1} \equiv 1 \rightarrow \sqrt{a} \equiv a^{\frac{p-1}{2}}$, if $p \equiv 3 \bmod 4 \rightarrow$
$a^{\frac{p-1}{2}} \equiv a^{2m+1}$ - unique square root in
$Q_p = \{4m + 3 \bmod p\} \subset \mathbb{Z}_p$; Squaring is a permutation on Q_p
(every square has a unique square root, which is itself a square).

Hard Bit and the best of BBS

Claim

The least significant bit of x is a hard bit for the one-way function f_{BBS}

Direct computing of bits

$G_{BBS}(x)_{\{}j\} = lsb(x^{2^j} \bmod n) = lsb(x^{\alpha} \bmod \phi(n)$ where
$\phi(n) = (p-1)(q-1)$
$\mathrm{G}_{BBS}(x)_{\{j\}}$ is computed in time $O(\max\{|x|^3, |x|^2 \log j\})$

Part III

Generator Construction

Polynomial Parameter

Definition

Parameter k_n is called polynomial if there is a constant $c > 0$ such that $\forall n \in \mathbb{N}$

$$\frac{1}{cn^c} \leq k_n \leq cn^c$$

k_n is called **P**-time polynomial parameter if in addition there is a constant $c' > 0$ such that $\forall n$, k_n is computable in time at most $c' n^{c'}$

Function Ensemble

Definition

Let $f : \{0,1\}^{t_n} \to \{0,1\}^{l_n}$ denote a function ensemble, where t_n and l_n are integer-valued **P**-time polynomial parameters and where f with respect to n is a function mapping $\{0,1\}^{t_n}$ to $\{0,1\}^{l_n}$.

- f is injective $\Rightarrow$ one-to-one function ensemble
- f is injective and $l_n = t_n \Rightarrow$ permutation ensemble
- $f : \{0,1\}^{t_n} \times \{0,1\}^{l_n} \to \{0,1\}^{m_n} \Rightarrow$ ensemble with 2 inputs

At most every primitive in the paper (pseudorandom generator, one-way function, hidden bit) will be discussed here as a function ensemble.

Adversary and security definition

- Function ensemble may be broken (in some sense) by another function ensemble.
- For instance, adversary tries to break one-way function.
- The ability of breaking something is measured by time-success ratio.

Definition

Adversary A is a function ensemble, it is breaking another function ensemble f. The time-success ratio of A for f $\mathbf{R}_{\mathbf{t_n}} = T_n / sp_n(A)$, where t_n is the length of the private input to f, T_n is the worst-case running time of A.

Shannon Entropy

Definition

Let D be a distribution on a set S. We define the information of x with respect to d to be $I_D(x) = -\log(D(x))$; Let X be a random value with distribution D ($X \in_D S$ The Shannon Entropy of D is $H(D) = E[I_D(X)]$

Computational Entropy

Definition

Let $f : \{0,1\}^{t_n} \rightarrow \{0,1\}^{l_n}$ be a **P**-time function ensemble and let s_n be a polynomial parameter. Then f has **R**-secure computational entropy s_n if there is a **P**-time function ensemble $f' : \{0,1\}^{m_n} \rightarrow \{0,1\}^{l_n}$ such that $f(U_{t_n})$ and $f'(U_{m_n})$ are **R**-secure computationally indistinguishable and $\mathbf{H}(f'(U_{m_n})) \geq s_n$.

Construction steps

- Any one-way function
- False-Entropy Generator

Definition

Let $f : \{0,1\}^{t_n} \to \{0,1\}^{l_n}$ be a **P**-time function ensemble and let s_n be a polynomial parameter. Then f is an **R**-secure false-entropy generator with false entropy s_n if $f(U_{t_n})$ has **R**-secure computational entropy $H(f(U_{t_n})) + s_n$.

False-entropy generator concept is that it's computational entropy g(X) is significantly greater than the Shannon entropy of g(X).

Construction steps

- Pseudoentropy generator

Definition

Let $f : \{0,1\}^{t_n} \rightarrow \{0,1\}^{l_n}$ be a **P**-time function ensemble and let s_n be a polynomial parameter. Then f is an **R**-secure pseudoentropy generator with pseudoentropy s_n if $f(U_{t_n})$ has **R**-secure computational entropy $t_n + s_n$.

Pseudoentropy generator concept is that it's computational entropy g(X) is significantly greater than the Shannon entropy of X.

- Pseudorandom generator

My sources

1. A Pseudorandom Generator From Any One-Way Function by J. Hastad,R. Implagliazzo, L. Levin and M. Luby
2. Lecture notes On Cryptography by S. Goldwasser and M. Bellare
3. Foundations of Cryptography - A Primer by O. Goldreich

Next-bit Test

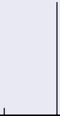
1. Remember the algorithm which was able to distinguish between hybrids and look at next definition:
A is called a next-bit test for a bit string generator if for any generated string S it can predict from a prefix $S_{1 \dots p}$ S_{p+1} bit of the string with some probability $\frac{1}{2}$
Can a human test some generator?
This is a string 011 011 101 100 100 011 110 ???

Linear Feedback Register

2. We have a simple Linear Feedback Shift Register. Build a tree of pseudorandom functions for it and tell, how we can use such functions in a telephone coin flip problem.

Distributions and Entropy

3. We have a histogram of 2 distributions. Tell me, for which distribution entropy is higher? what means entropy in this case?



Break a classical pseudorandom scheme

4. We have $\sqrt{5} = 10.001111000110111\ldots$ it seems quite random. But it's an insecure generator. Try to prove it!