PROJECT REPORT

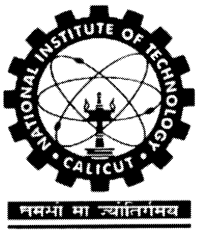
Pseudorandom Generator based on the Directed Hamiltonian Path Problem

Submitted in partial fulfilment of
the requirements for the award of the degree of

Bachelor of Technology
in
Computer Science and Engineering

Submitted by

Girish R Varma
CS04B025
Under the guidance of
Mr. Muralikrishnan K



Department of Computer Science & Engineering
National Institute of Technology Calicut
Kerala - 673601
2008

National Institute of Technology Calicut
Department of Computer Science & Engineering

Certified that this Project Report entitled

Pseudorandom Generator based on the Directed Hamiltonian Path Problem

is a bonafide record of the work carried out by

Girish R Varma
CS04B025

in partial fulfilment of
the requirements for the award of the degree of
Bachelor of Technology
in
Computer Science and Engineering

under our guidance

Mr. Muralikrishnan K
Lecturer
Dept.of Computer Science & Engineering

Dr.M.P.Sebastian
Professor and Head
Dept.of Computer Science & Engineering

Contents

1 Introduction **4**
- 1.1 An Incomplete Model . . . 4
- 1.2 Revised Model . . . 4
- 1.3 Security of the Model . . . 5
  - 1.3.1 Information Theoretic Definition . . . 5
  - 1.3.2 Complexity Theoretic Definition . . . 5
- 1.4 Some Common Terms . . . 5
  - 1.4.1 Efficient and Intractable . . . 5
  - 1.4.2 Negligible and Noticeable . . . 5
- 1.5 Notations . . . 6

2 One-Way Functions **7**
- 2.1 Hard-Core Predicate . . . 7
- 2.2 Hard-Core Predicates for any One-Way Function . . . 7

3 Construction of One-Way Functions based on NP-Hard Problems **10**
- 3.1 Functions that are Hard on Average but Easy with Auxiliary Input . . . 10
- 3.2 Construction based on HAMPATH . . . 10

4 Pseudorandom Generator **12**
- 4.1 Equivalence of Pseudorandom Generator with different Expansion Factors . . . 12
- 4.2 Psuedorandom Generators based on One-Way Functions . . . 14

5 Conclusion **16**

References **17**

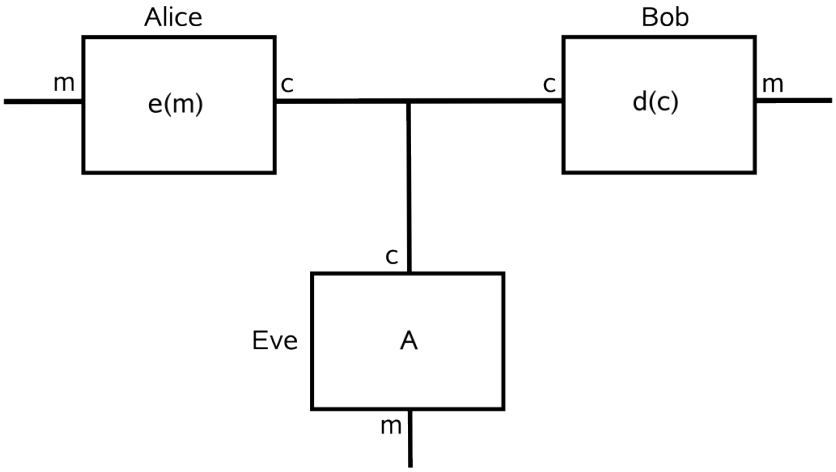
Abstract

This report is a study of one-way functions and pseudorandom generators, which forms the primitives in construction of cryptographic systems. Simpler arguments proving equivalence of one-way functions and pseudorandom generators are presented[1][2]. Also a one-way function is constructed based on the HAMPATH(directed hamiltonian path probem). It is found that this requires a strong assumption of hardness on average of HAMPATH. Basing the one-wayness of the function constructed on weaker assumption is left for future work.

1 Introduction

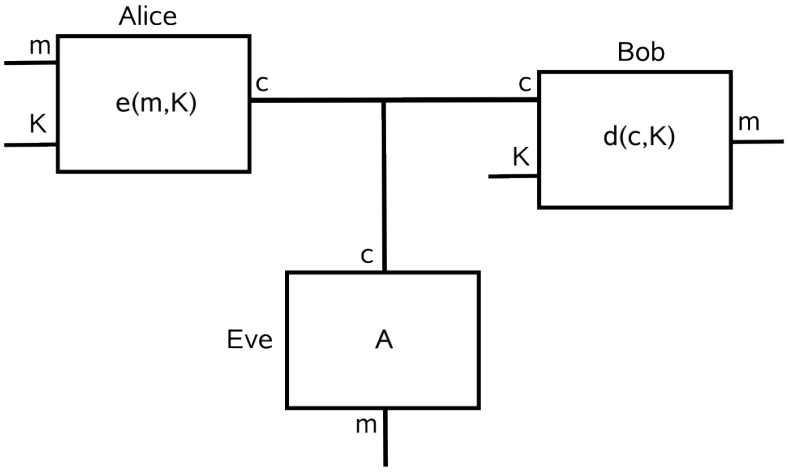
One of the basic problems that cryptography attempts to find a solution is described as follows[3]. Alice is sending a message to Bob, and wants to do it secretly. Bob wants to make sure that the message he receives was sent by Alice. The message is sent over an insecure channel, where Eve (eavesdropper) is listening and she tries to get the message.

1.1 An Incomplete Model



So Alice encrypts (using encryption function e) the message also called plain text (m) and converts it to a cipher text(c). Bob decrypts (using decryption function d) the cipher text and converts it back to message. The problem with this model is that e and d might be known to Eve. So Eve cannot be stopped from obtaining m.The model is incomplete.

1.2 Revised Model



Assume that Alice and Bob know some secret which is not known to Eve. This secret is called the key (K). The functions e and d now also takes K as an input. The functions are such that the message encrypted using a particular key can only be decrypted using the same key. This model is popularly called Private Key or Symmetric Encryption.

A variation of this is the Public Key Encryption where the encryption key(or public key) and decryption key(or private key) are different. Each party who want to communicate, has a private key known only to him and a public key known to every one. To send a message to

Bob, Alice encrypts m using his public key. Then c can be decrypted only using his private key, known only to him. It requires an additional restriction that the Bobs private key should not be feasibly computable from his public key.

1.3 Security of the Model

Security of the model above is defined in two ways:

1.3.1 Information Theoretic Definition

Given c, Eve should not be able to gain any information about m. Shannon proved that it can be done by using a random key equal to the size of the message. Each bit of the message is XOR-ed with the corresponding bit of the key. This technique is called One Time Pad (OTP). The problem with this technique is that the size of messages can be very large. It is inconvenient to have large key values.

1.3.2 Complexity Theoretic Definition

Assume Eve has limited computational capability. Given c, Eve should not be able to obtain m by efficient methods. Construction of a model as above without having the key size increasing is possible using this assumption. It requires structures known as one-way functions and pseudorandom generators.

1.4 Some Common Terms

1.4.1 Efficient and Intractable

Computations that can be done by probabilistic polynomial time Turing Machines are considered efficient. Those that cannot be done by them are considered intractable. The adversarial tasks that we are interested, is such that if a solution is found, it can easily be verified($\in NP$). So cryptographic systems based on complexity theory also assumes that $\exists L \in NP \setminus BPP$. Therefore $P \neq NP$ is necessary but not sufficient[4].

1.4.2 Negligible and Noticeable

A function is said to be negligible if

$$\forall k \; \exists n_0 \; \forall n > n_0, \;\; f(n) \leq 1/n^k \tag{1.1}$$

If an algorithm has negligible probability of success, repeating the algorithm polynomial number of times, yields a new algorithm with negligible probability of success.

A function is said to be noticeable if

$$\exists k \; \exists n_0 \; \forall n > n_0, \;\; f(n) \geq \frac{1}{n^k} \tag{1.2}$$

Functions may be neither negligible nor noticeable.

1.5 Notations

- Here a random variable is used to denote a mapping from some sample space to a set of strings.
- U_n denotes a random variable uniformly distributed in $\{0,1\}^n$.
- If A is a probabilistic polynomial time algorithm, f is a function $f : \{0,1\}^* \rightarrow \{0,1\}^*$, then
$$Pr[A(f(U_n)) = U_n]$$
denotes the following
$$\sum_{x \in \{0,1\}^n} Pr[U_n = x] * Pr[A(f(x)) = x]$$
That is for some deterministic $A^{'}$, a polynomial $p(n)$ number of coin tosses,
$$\sum_{x \in \{0,1\}^n} Pr[U_n = x] * \frac{\left(\sum_{c \in \{0,1\}^{p(n)}} \psi[A^{'}(f(x), c) = x]\right)}{2^{p(n)}}$$
where ψ denotes the indicater random variable.

2 One-Way Functions

In simple terms, they are functions that are easy to compute but hard on average to invert[1]. For simplicity, i am restricting to functions that are length regular($\forall x \in \{0,1\}^n,\ |f(x)|$ remains same). To avoid functions that decrease the length of the input exponentially, i restrict that length of x may be computed easily from that of $f(x)$.

Definition 2.1. A function $f : \{0,1\}^* \to \{0,1\}^*$ is said to be one-way if

1. f is deterministic polynomial time computable.
2. $\exists$ (a function l, polynomial time algorithm A) $\forall x$,
$$l(|x|) = |f(x)| \bigwedge A(l(|x|)) = |x|$$
3. $\forall$(probabilistic polynomial time algorithms A, k) $\exists n_0\ \forall n > n_0$,
$$Pr[A(f(U_n)) = U_n] < \frac{1}{n^k} \tag{2.1}$$

2.1 Hard-Core Predicate

A hard-core predicate with respect to a function f, is a function with 1 bit output that is easy to compute from the input of f, but can be computed only with negligible advantage on the average from the output.

Definition 2.2. A predicate $b : \{0,1\}^* \to \{0,1\}$ is called a hard-core of a function f if

1. b is deterministic polynomial time computable.
2. $\forall$(probabilistic polynomial time algorithms A, k) $\exists n_0\ \forall n > n_0$,
$$Pr[A(f(U_n)) = b(U_n)] < \frac{1}{2} + \frac{1}{n^k} \tag{2.2}$$

2.2 Hard-Core Predicates for any One-Way Function

Theorem 2.3. *Let f be an arbitrary one-way function, and let g be defined by $g(x,r) = (f(x),r)$, where $|x| = |r|$. Let $b(x,r) = \bigoplus_{i=1}^{n} x_i \cdot r_i$ (the inner product mod 2 of the binary vectors x and r). Then the predicate b is a hard-core of the function g.*

The function g is one-way, as inverting f can be reduced to inverting g. The theorem states that the exclusive or of a random subset of bits of x, cannot be found by efficient means given $f(x)$ and the subset itself.

Proof. The proof follows by contradiction. Let G be a probabilistic polynomial time algorithm(that was not supposed to exist) for computing $b(x,r)$ from $f(x)$ and r, with noticeable probability. Then

$$Pr[G(f(U_n), R_n) = b(U_n, R_n)] = \frac{1}{2} + \varepsilon(n) \bigwedge \varepsilon(n) > \frac{1}{n^k} \tag{2.3}$$

where R_n is uniformly distributed in $\{0,1\}^n$. Using G it is possible to construct an efficient algorithm, G' for inverting g, resulting in a contradiction.

We analyze G' only for a noticeable fraction of $x \in \{0,1\}^n$(denoted as S_n) and we prove that G' inverts f with noticeable probability for $x \in S_n$. This is enough to show that G' inverts f with noticeable probability.

Lemma 2.4. *Let $S_n \subseteq \{0,1\}^n$ be the set of x such that,*

$$Pr[G(f(x), R_n) = b(x, R_n)] \geq \frac{1}{2} + \frac{\varepsilon(n)}{2}$$

Then $|S_n| \geq \frac{\varepsilon(n)}{2}$.

Loosely speaking, this means for a noticeable fraction of inputs, the advantage in finding $b(x, R_n)$ is noticeable.

Proof. Let $s(x) = Pr[G(f(x), R_n) = b(x, R_n)]$ and $t = \frac{1}{2} + \frac{\varepsilon(n)}{2}$. Then $E(s(X)) = \frac{1}{2} + \varepsilon(n)$.

$$\begin{aligned} & E(s(X)) \leq t * 1 + 1 * Pr[s(X) \geq t] \\ & \Leftrightarrow Pr[s(X) \geq t] \geq E(s(X)) - t \\ & \Leftrightarrow Pr[s(X) \geq \frac{1}{2} + \frac{\varepsilon(n)}{2}] \geq \frac{\varepsilon(n)}{2} \end{aligned} \tag{2.4}$$

□

Lemma 2.5.

$$b(x, \alpha) \bigoplus b(x, \beta) = b\left(x, \alpha \bigoplus \beta\right)$$

Proof.

$$b(x, \alpha) \bigoplus b(x, \beta) = \left(\bigoplus_{i=1}^{n} x_i \cdot \alpha_i\right) \bigoplus \left(\bigoplus_{i=1}^{n} x_i \cdot \beta_i\right) = \bigoplus_{i=1}^{n} x_i \cdot \left(\alpha_i \bigoplus \beta_i\right) = b\left(x, \alpha \bigoplus \beta\right)$$

The equations given below follows from Claim 2.
Especially, if e_i denote a n bit string with all 0 but 1 in only the ith position, then

$$b(x, r) \bigoplus b\left(x, r \bigoplus e_i\right) = b(x, e_i) = x_i$$

Also, for $i \in \{1..l\}$, if $\forall i,\ b(x, s_i) = \sigma_i$, then for any $J \subseteq \{1, ..l\}$, let $r^J = \bigoplus_{j \in J} s_j$ and $\rho^J = \bigoplus_{j \in J} \sigma_j$.

$$b(x, r^J) = b\left(x, \bigoplus_{j \in J} s_j\right) = \bigoplus_{j \in J} b(x, s_j) = \bigoplus_{j \in J} \sigma_j = \rho^J$$

Loosely speaking, this means by guessing only $\sigma_1..\sigma_l$, each equal to $b(x, s_1)..b(x, s_l)$, we get $2^l - 1$ samples(one for each subset of $\{1, ..l\}$), with $b(x, r^J) = \rho^J$. Each r^J so obtained are pairwise independent if $s_1..s_l$ are uniformly and independently selected. This is true because, for each $J_1 \neq J_2 \subseteq \{1, ..l\},\ J_1 \subset J_2 \bigvee J_2 \subset J_1 \bigvee$ *no subset relation exists*. So $\exists s_i \in J_1 \notin J_2$ or vice versa. Since s_i's are xor-ed, r^{J_1} and r^{J_2} are independent.

□

Lemma 2.6. *If each x_i is obtained with probability atleast $\frac{1}{2} + \frac{1}{2p(n)}$, then with $m = 2np^2(n)$ pair-wise independent trials, the probability can be amplified to $1 - \frac{1}{2n}$. Then x can be obtained with probability atleast $1 - \frac{n}{2n} = \frac{1}{2}$.*

Proof. Let X_j denote whether the ith trial was successful in obtaining x_i. Using Chebyshev's inequality,

$$\begin{aligned} Pr[failure] &= Pr\left[\sum X_j \leq \frac{m}{2}\right] \\ &\leq Pr\left[\left|\sum X_j - (\frac{1}{2} + \frac{1}{2p(n)})m\right| \geq \frac{m}{2p(n)}\right] \\ &\leq \frac{Var[X_j]}{(\frac{1}{2p(n)})^2 * m} \\ &= Var[X_j]/(\frac{1}{2p(n)})^2 * (2np^2(n)) \\ &= \frac{1/4}{n/2} \\ &= \frac{1}{2n} \end{aligned} \tag{2.5}$$

□

Using the above lemmas we can write an algorithm for inverting f.

Algorithm 2.1 $G^{'}(y)$

1: $l \leftarrow log(np^2(n) + 1)$.
2: Uniformly and independently select $s_1..s_l$ and $\sigma_1..\sigma_l$.
3: **for all** $J \subseteq \{1, ..l\}$ **do**
4: $\quad r^J \leftarrow \bigoplus_{j \in J} s_j$
5: $\quad \rho^J \leftarrow \bigoplus_{j \in J} \sigma_j$
6: $\quad$ **for all** $i \in \{1, ..n\}$ **do**
7: $\quad\quad z_i^J \leftarrow G(y, r^J) \bigoplus \rho^J$
8: $\quad$ **end for**
9: **end for**
10: $x_i \leftarrow$ majority value of z_i's.
11: **return** $x_1...x_n$.

All $\sigma_i = b(x, s_i)$ with probability $2^{-l} = \frac{1}{2np^2(n)+1}$. For $\varepsilon(n)/2$ fraction of x's, this algorithm inverts f with probability $\frac{1}{2} * \frac{1}{2np^2(n)+1}$. Therefore

$$\begin{aligned} Pr[G^{'}(f(U_n)) = U_n] &\geq \frac{1}{2} * \frac{1}{2np^2(n) + 1} * \frac{\varepsilon(n)}{2} \\ &= \frac{1}{8np^3(n) + 4p(n)} \end{aligned}$$

Hence f is not one-way which is a contradiction.

□

3 Construction of One-Way Functions based on NP-Hard Problems

For constructing one-way functions based on NP-Hard Problems it is useful to consider the following structure

3.1 Functions that are Hard on Average but Easy with Auxiliary Input

Definition 3.1. A function $h : D^{Gen} \rightarrow \{0,1\}^*$ is hard on average but easy with auxiliary input if

1. $\exists$ (probabilistic polynomial time algorithm Gen, polynomial time algorithm A) such that $A(x,y) = h(x)$ for every (x,y) that is a possible output of $Gen(1^n)$ for some n
2. $\forall$ (probabilistic polynomial time algorithm $A^{'}, k$) $\exists n_0$ $\forall n > n_0$

$$Pr[A^{'}(U_{D_n^{Gen}}) = h(U_{D_n^{Gen}})] < 1/n^k$$

3. where D^{Gen} is the set of all x that is the first output of $Gen(1^n)$ for some n and $D_n^{Gen} = D^{Gen} \cap \{0,1\}^n$.

Theorem 3.2. *If there exists a function that is hard on average but easy with auxiliary input, then we can define a function that is one-way from the coins used by Gen to x.*

Proof. Let f be a function mapping the coin tosses of Gen to its first output. Let us assume that f is not one-way ie $\exists$ a probabilistic polynomial time algorithm A such that

$$Pr[A(f(U_n)) = U_n] > \frac{1}{n^k}$$

Then we can have an algorithm for computing $h(x)$ with noticeable probability using A to find the coin tosses of Gen. Then execute Gen with the specific coin toss deterministically to obtain (x,y) and use $A^{'}$ to compute $h(x)$.

□

3.2 Construction based on HAMPATH

Here a one-way function is constructed based on the HAMPATH problem.

$$\text{HAMPATH} = \{< G, s, t >:\ G \text{ is a directed graph with a hamiltonian path from } s \text{ to } t\}$$

Let $h(< G, s, t >) = (< G, s, t >, 1)$ if $< G, s, t > \in$ HAMPATH else $(< G, s, t >, 0)$. Also the domain of h only consists of those $< G, s, t >$ which is a possible first output of $Gen(1^n)$ for some n.The algorithms Gen and A, required for definition of h are described below.

Algorithm 3.1 $Gen(1^n)$

for $i = 1$ to n **do**
  $Adj[i-1, i] = 1$
end for
for $i = 1$ to n **do**
  for $j = i + 2$ to n **do**
    $c =$ TossCoin()
    $Adj[i, j] = c$
  end for
end for
$c =$ TossCoin()
if $c = 1$ **then**
  return $(< G, 1, n >, < 1, \ldots, n >)$
else
  Uniformly choose $j \in \{1, \ldots, n\}$
  $Adj[j-1, j] = 0$
  for $i = 1$ to n **do**
    if $i \neq j$ **then**
      $c =$ TossCoin()
      $Adj[j-1, j] = c$
    end if
  end for
  return $(< G, 1, n >, < 0, \ldots, 0 >)$
end if

Algorithm 3.2 $A(< G, s, t >, \pi)$

if π is a valid hamiltonian path from s to t in G **then**
  return $(< G, s, t >, 1)$
else
  return $(< G, s, t >, 0)$
end if

Remark 3.3. HAMPATH $\notin BPP$ implies $\exists\ x$ such that for even the best probabilistic polynomial time algorithms for HAMPATH

$$Pr[\text{the algorithm succeeds for } x] < \frac{2}{3}$$

This only means that $\exists x$ that may be hard instances. For hardness on average, we need this value averaged over all x to be negligible. So the assumption that HAMPATH $\in NP \setminus BPP$ seems to be insufficient for proving the hardness on average of the constructed function. Though it will be interesting if HAMPATH $\in NP \setminus BPP \rightarrow \exists$ a one way function or the more general question $\exists L \in NP \setminus BPP \rightarrow \exists$ a one way function.

Hence it is required to directly assume hardness on average for h. ie, $\forall$(probabilistic polynomial time algorithms $A^{'},\ k$) $\exists n_0\ \forall n > n_0$,

$$Pr[A^{'}(U_n) = h(U_n)] < \frac{1}{n^k}$$

Then we can define a function as proved in Theorem 3.2 from the coin tosses made by Gen to its first output. This function is one-way.

4 Pseudorandom Generator

Definition 4.1. A **pseudorandom generator** is a deterministic polynomial-time algorithm G satisfying the following two conditions:

1. Expansion: $\exists$ (a function l and a polynomial time algorithm A), such that $\forall s$,

$$l(|s|) = |G(s)| \bigwedge A(l(|s|)) = |s|$$

2. Pseudorandomness: $\forall$(Probabilistic Polynomial Time Algorithm D, k) $\exists n_0$ $\forall n > n_0$

$$|Pr[D(G(U_n)) = 1] - Pr[D(U_n) = 1]| < \frac{1}{n^k}$$

The function l is called the expansion factor of G.

4.1 Equivalence of Pseudorandom Generator with different Expansion Factors

Theorem 4.2. *For any polynomial $p(n)$,*
$\exists$ pseudorandom generator of expansion $l(n) = n+1 \Leftrightarrow \exists$ pseudorandom generator of expansion $l^{'}(n) = p(n)$.

Proof. The backward direction is trivial. The $p(n)$ bit output just need to be stripped at $n+1$ bits. For proving the forward direction, we construct a pseudorandom generator of expansion $p(n)$ from one with $n+1$.

Let G be a pseudorandom generator of expansion $l(n) = n+1$. Therefore $\forall$(probabilistic polynomial time algorithms D, k) $\exists n_0$ $\forall n > n_0$,

$$|Pr[D(G(U_n)) = 1] - Pr[D(U_{n+1}) = 1]| < \frac{1}{n^k} \tag{4.1}$$

Consider the following algorithm, From the algorithm $G^{'}$ it follows that

Algorithm 4.1 $G^{'}(s)$

```
1: Let s_0 = s
2: for i = 1 to p(n) do
3:     σ_i s_i = G(s_{i-1})
4: end for
5: return σ_1..σ_{p(n)}
```

$$\text{pref}_{j+1}\left(G^{'}(x)\right) = \text{pref}_1\left(G(x)\right) \cdot \text{pref}_j\left(G^{'}\left(\text{suff}_n(G(x))\right)\right) \tag{4.2}$$

where $\text{pref}_j(\alpha)$ denotes the j bit prefix of α.
For the purpose of the proof, let

$$f_{p(n)-k}(\alpha) = \text{pref}_1(\alpha) \cdot \text{pref}_{p(n)-k-1}\left(G^{'}\left(\text{suff}_n(\alpha)\right)\right) \tag{4.3}$$

Lemma 4.3. *$G^{'}$ is a pseudorandom generator with expansion factor $p(n)$.*

Proof. The proof follows by a method known as the hybrid technique. In this method we define $p(n)+1$ hybrid random variables, numbered from 0 to $p(n)(H_k^n\ 0 \leq k \leq p(n))$. The extreme ones($H_0^n$ and $H_{p(n)}^n$), being equal to $U_{p(n)}$ and $G^{'}(U_n)$. Then it is shown that a polynomial time distinguisher for extreme hybrids will imply a polynomial time distinguisher for neighboring hybrids(H_k^n and H_{k+1}^n) and this will in turn imply a polynomial time distinguisher for $G(U_n)$ and U_{n+1}, which is a contradiction.

□

Let

$$H_k^n = U_k \cdot \text{pref}_{p(n)-k}\left(G^{'}(U_n^{'})\right) \tag{4.4}$$

So

$$H_0^n = G^{'}(U_n) \text{ and } H_{p(n)}^n = U_{p(n)}$$

Also,

$$\begin{aligned} H_k^n &= U_k \cdot \text{pref}_{p(n)-k}\left(G^{'}(U_n^{'})\right) \\ &= U_k \cdot \text{pref}_1\left(G(U_n^{'})\right) \cdot \text{pref}_{p(n)-k-1}\left(G^{'}\left(\text{suff}_n\left(G(U_n^{'})\right)\right)\right) \\ &= U_k \cdot f_{p(n)-k}\left(G(U_n^{'})\right) \end{aligned} \tag{4.5}$$

where $\text{suff}_j(\alpha)$ denotes the j bit suffix of α.

$$\begin{aligned} H_{k+1}^n &= U_{k+1} \cdot \text{pref}_{p(n)-k-1}\left(G^{'}(U_n^{'})\right) \\ &= U_k \cdot \text{pref}_1\left(U_{n+1}^{'}\right) \cdot \text{pref}_{p(n)-k-1}\left(G^{'}\left(\text{suff}_n\left(U_{n+1}^{'}\right)\right)\right) \\ &= U_k \cdot f_{p(n)-k}\left(U_{n+1}^{'}\right) \end{aligned} \tag{4.6}$$

For the purpose of getting a contradiction, let $D^{'}$ be a polynomial time distinguishing algorithm such that $\exists k^{'}\ \forall n > n_0$

$$\left|Pr[D^{'}(G^{'}(U_n)) = 1] - Pr[D^{'}(U_{p(n)}) = 1]\right| > \frac{1}{n^{k^{'}}}$$

Consider the following algorithm, From the construction it follows that,

Algorithm 4.2 $D(\alpha_{n+1})$

1: Uniformly select $k \in \{0, 1, .., p(n)-1\}$
2: Uniformly select $\beta \in \{0,1\}^k$
3: **return** $D^{'}(\beta \cdot f_{p(n)-k}(\alpha))$

$$Pr[D(\alpha) = 1] = \frac{1}{p(n)} \sum_{k=0}^{p(n)-1} Pr[D^{'}(U_k \cdot f_{p(n)-k}(\alpha)) = 1]$$

Lemma 4.4. *D can distinguish $G(U_n)$ and U_{n+1} with noticeable probability.*

Proof.

$$Pr[D(G(U_n)) = 1] = \frac{1}{p(n)} \sum_{k=0}^{p(n)-1} Pr[D^{'}(U_k \cdot f_{p(n)-k}(G(U_n))) = 1]$$

$$= \frac{1}{p(n)} \sum_{k=0}^{p(n)-1} Pr[D^{'}(H_k^n) = 1]$$

$$Pr[D(U_{n+1}) = 1] = \frac{1}{p(n)} \sum_{k=0}^{p(n)-1} Pr[D^{'}(U_k \cdot f_{p(n)-k}(U_{n+1})) = 1]$$

$$= \frac{1}{p(n)} \sum_{k=0}^{p(n)-1} Pr[D^{'}(H_{k+1}^n) = 1]$$

$$Pr[D(G(U_n)) = 1] - Pr[D(U_{n+1}) = 1] = \frac{1}{p(n)} \Big(Pr[D^{'}(H_0^n) = 1] - Pr[D^{'}(H_{p(n)}^n) = 1] \Big)$$

$$> \frac{1}{p(n)} * \frac{1}{n^{k'}}$$

□

□

4.2 Psuedorandom Generators based on One-Way Functions

Theorem 4.5. *If $f : \{0,1\}^n \to \{0,1\}^n$ is an one-one one-way function and b be a hard-core predicate for f, then $G(x) = f(x) \cdot b(x)$ is pseudorandom generator with expansion factor $l(n) = n + 1$.*

Proof. We prove that the task of distinguishing $G(U_n)$ from U_{n+1} is equivalent to the task of distinguishing

$$E^1 = f(U_n) \cdot b(U_n) \text{ and } E^2 = f(U_n) \cdot \overline{b}(U_n) \tag{4.7}$$

where $\overline{b}(U_n) = 1 - b(U_n)$. Then we construct an algorithm A which finds $b(U_n)$ from $f(U_n)$ with noticeable advantage using the distinguisher for E^1 and E^2.

Lemma 4.6. *D is a algorithm that is able to distinguish E^1 and E^2 with noticeable probability $\Leftrightarrow$ D is an algorithm that is able to distinguish $G(U_n)$ from U_{n+1} with noticeable probability.*

Proof. By definition, $G(U_n)$ and E^1 are identically distributed. U_{n+1} is identically distributed to $f(U_n) \cdot U_1^{'}$ as f is one-one. Also $U_1^{'}$ is uniformly distributed in the set $\{b(U_n), \overline{b}\,(U_n)\}$. Therefore U_{n+1} is distributed identically to the distribution obtained by taking E^1 with probability $\frac{1}{2}$ and E^2 with probability $\frac{1}{2}$.

$$Pr[D(G(U_n)) = 1] = Pr[D(E^1) = 1] \tag{4.8}$$

$$Pr[D(U_{n+1}) = 1] = \frac{1}{2} Pr[D(E^1) = 1] + \frac{1}{2} Pr[D(E^2) = 1] \tag{4.9}$$

$$Pr[D(G(U_n)) = 1] - Pr[D(U_{n+1}) = 1] = \frac{1}{2} \Big(Pr[D(E^1) = 1] - Pr[D(E^2) = 1] \Big) \tag{4.10}$$

Hence the claim is proved.

□

For the purpose of contradiction, let us assume that D is an algorithm able to distinguish E^1 and E^2 with noticeable probability. Consider the following algorithm,

Algorithm 4.3 $A(y = f(x))$

1: Uniformly select $\sigma \in \{0, 1\}$
2: **if** $D(y \cdot \sigma) = 1$ **then**
3: **return** σ
4: **else**
5: **return** $1 - \sigma$
6: **end if**

Lemma 4.7. *A can find $b(U_n)$ given $f(U_n)$ with noticeable advantage.*

Proof.

$$\begin{aligned} Pr[A(f(U_n)) = b(U_n)] \;=\; & Pr[D(f(U_n) \cdot U_1^{'}) = 1 \;\&\; U_1^{'} = b(U_n)] \\ & + Pr[D(f(U_n) \cdot U_1^{'}) = 0 \;\&\; 1 - U_1^{'} = b(U_n)] \\ = \; & Pr[D(f(U_n) \cdot b(U_n)) = 1 \;\&\; U_1^{'} = b(U_n)] \\ & + \; Pr[D(f(U_n) \cdot \overline{b}(U_n)) = 0 \;\&\; U_1^{'} = \overline{b}(U_n)] \\ = \; & \frac{1}{2} Pr[D(f(U_n) \cdot b(U_n)) = 1] \;+\; \frac{1}{2}\left(1 - Pr[D(f(U_n) \cdot \overline{b}(U_n)) = 1]\right) \\ = \; & \frac{1}{2} \;+\; \frac{1}{2} * \left(Pr[D(f(U_n) \cdot b(U_n)) = 1] - Pr[D(f(U_n) \cdot \overline{b}(U_n)) = 1]\right) \\ > \; & \frac{1}{2} \;+\; \frac{1}{2n^k} \end{aligned} \tag{4.11}$$

□

□

5 Conclusion

Simpler definitions for one-way functions and pseudorandom generators were presented and studied. Also simpler proofs for existence of hard-core predicates for one-way functions, construction pseudorandom generators from one-way functions using hard-core predicates and equivalence of pseudorandom generators with different expansion factors were presented[1].

A one-way function was defined based on HAMPATH $\in NP$-Hard. It was found that the assumption HAMPATH $\in NP \setminus BPP$ is not sufficient for this purpose. So the hardness on average for HAMPATH was assumed. Basing one-wayness of the constructed function on weaker assumption is left for future work. Also the papers[5][6], gives more results on basing one-way functions on NP-Hard problems.

References

[1] O. Goldriech, *Foundations of Cryptography Volume 1 : Basic Tools.* Cambridge University Press, 2001.

[2] S. Arora and B. Barak, *Complexity Theory: A Modern Approach.* http://www.cs.princeton.edu/theory/complexity/book.pdf.

[3] H. Delfs and H. Knebl, *Introduction to Cryptography: Principles and Applications.* Springer, 2002.

[4] M. Sipser, *Introduction to the Theory of Computation.* Thompson Brook/Cole, 2002.

[5] A. Bogdanov and L. Trevisan, "On worst-case to average-case reductions for np problems," *SIAM Journal on Computing*, vol. 36, no. 4, pp. 1119–1159, 2007.

[6] G. Akavia, Goldreich and Moshkovitz, "On basing one-way functions on np-hardness," *Annual ACM Symposium on Theory of Computing*, vol. 38, pp. 701–710, 2006.

Index

cipher text, 4
cryptography, 4

decryption function, 4

efficient, 5
encryption function, 4
expansion factor, 12

HAMPATH, 10
hard on average but easy with auxiliary input, 10
hard-core predicate, 7
hybrid technique, 13

intractable, 5

key, 4

negligible, 5
noticeable, 5

One Time Pad, 5
one-way, 7

plain text, 4
private key, 4
Private Key or Symmetric Encryption, 4
pseudorandom generator, 12
public key, 4
Public Key Encryption, 4