CSA E0 235: Cryptography February 29, 2016

Scribe for Lecture 12

Instructor: Arpita Patra *Submitted by: Anupam Sanghi*

1 Introduction

In the last lecture, we saw a method of constructing a Pseudorandom Function (PRF) using a Pseudorandom Generator (PRG) and also proved that our construction indeed gives a PRF. In this lecture, we start by revisiting the concept of One-Way Functions (OWF) and One-Way Permutations (OWP). Later, we will see one of the crucial aspects of OWF, known as Hard-Core Predicate (HCP).

2 One-Way Functions (OWF)

An OWF is a function that is easy to compute on every input, but is (almost always) hard to invert given the image of a random input. Since we are interested in a computational task that is almost always hard to solve, the hard-to-invert requirement is formalized by saying that a polynomial-time adversary will fail to invert the function (i.e., find *some* preimage), except with negligible probability.

2.1 The Invert Experiment

The experiment $Invert_{\mathcal{A},f}(n)$ (shown in Figure 1) proceeds as follows:

- The challenger $\mathcal{C}$ picks a random $x \in_R \{0,1\}^n$ from the domain of the one-way function f and computes $y = f(x)$.
- The adversary $\mathcal{A}$ is given y as the input and it has to guess a preimage x' of y.
- $\mathcal{C}$ verifies whether x' is indeed correct or not. If the guess is correct, i.e. $f(x') = y$, then $\mathcal{A}$ wins and the game's output is 1, else $\mathcal{A}$ loses and the game's output is 0.

Remark

1. Note that it is sufficient for $\mathcal{A}$ to find any preimage of y. It need not be the original x.
2. It is always possible to succeed with negligible probability, by just guessing a preimage of the appropriate length.
3. Given exponential-time, it is always possible to search the entire domain of a preimage. Thus, existence of one-way functions is inherently an assumption about *computational complexity* and *computational hardness.* That is, it considers a problem that can be solved in principle. However, it cannot be solved efficiently.

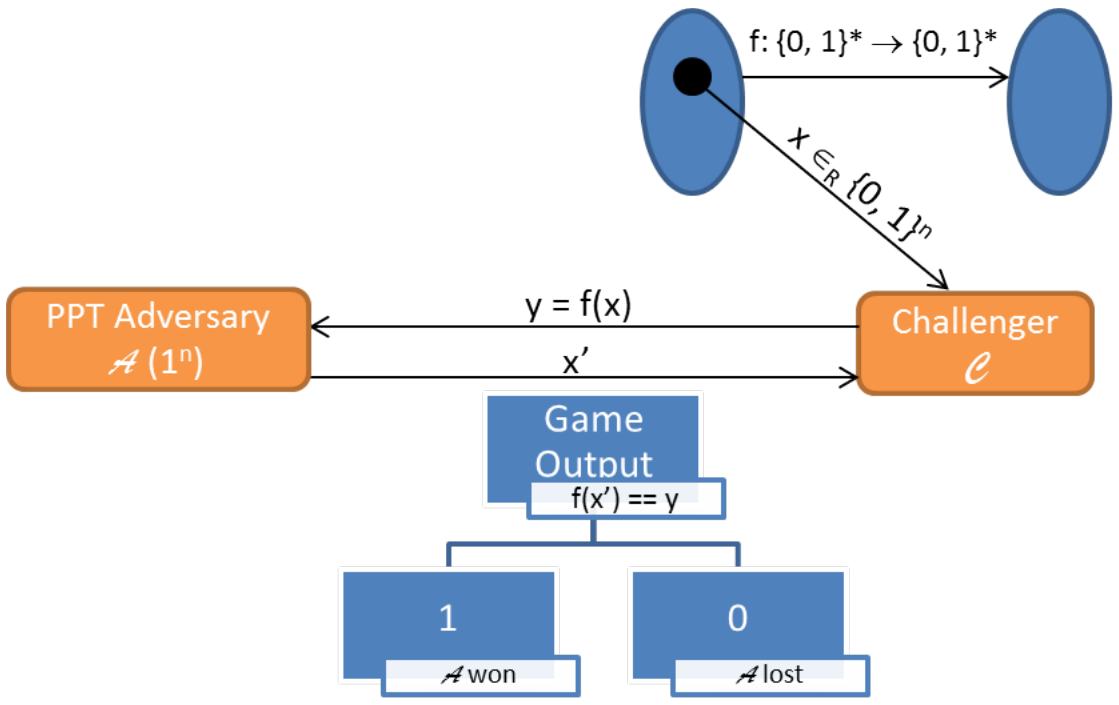


Figure 1: Invert Experiment

2.2 OWF: Mathematical Formulation

Formally, we can define OWF as follows:

Definition (one-way functions): *A function $f : \{0,1\}^* \rightarrow \{0,1\}^*$ is* ***one-way****, if the following two conditions hold:*

1. Easy to compute: *There exists a polynomial-time algorithm M_f such that on input any $x \in \{0,1\}^*$, M_f outputs $f(x)$ (i.e., $M_f(x) = f(x)$ for every x).*

2. Hard to invert: *For every probabilistic polynomial-time (PPT) inverting algorithm $\mathcal{A}$, there exists a negligible function* ***negl*** *such that*

$$Pr[Invert_{\mathcal{A},f}(n) = 1] \leq negl(n) \approx Pr[\mathcal{A}(f(x), 1^n) \in f^{-1}(f(x))]_{x \leftarrow \{0,1\}^n} \leq negl(n) \quad (1)$$

where the probability is taken over the uniform[1] *choice of $x \in \{0,1\}^n$ and the random coin tosses of $\mathcal{A}$.*

2.3 Non-One-Way Functions

A function that is not one-way is not necessarily easy to invert all the time (or even "often"). Rather, the converse of the Definition 1 is that there exists a PPT adversary $\mathcal{A}$ that inverts

[1] We stress that it is only guaranteed that a one-way function is hard to invert when the input is *uniformly distributed*

$f(x)$ for $x \in \{0,1\}^n$ with at least non-negligible probability. This actually means, that there exists a positive polynomial $p(.)$ such that for *infinitely many n's*,

$$Pr[\mathcal{A}(f(x), 1^n) \in f^{-1}(f(x))]_{x \leftarrow \{0,1\}^n} \geq 1/p(n) \tag{2}$$

Examples

1. Consider f such that,

$$Pr[\mathcal{A}(f(x), 1^n) \in f^{-1}(f(x))]_{x \leftarrow \{0,1\}^n} \begin{cases} > 1/n^{10} & \text{when } n \text{ is even} \\ \leq negl(n) & \text{when } n \text{ is odd} \end{cases}$$

   This is a <u>non-one-way</u> function, because as per the above definition, the adversary can invert the function value with a non negligible probability for all the even n's, which are infinitely many.

2. $f(x, y) = xy$, where $x, y \in \mathcal{N}$

$$Pr[\mathcal{A}(f(x, y), 1^n) \in f^{-1}(f(x, y))] \geq 3/4$$

   Here, $xy \in \{0,1\}^n$ when is even, we can always find a valid preimage i.e., $(2, xy/2)$. The product of x and y will be even when either x is even, or y is even, or both. Therefore, the probability of inverting the function is at least 3/4 (non-negligible) for all values of n. Hence, the given function is <u>non-one-way</u>.

3. $f(x) = x_1 x_2 ... x_{n-1}$, where $x \in \{0,1\}^n$

$$Pr[\mathcal{A}(f(x), 1^n) \in f^{-1}(f(x))]_{x \leftarrow \{0,1\}^n} = 1$$

   Here, irrespective of the last bit of x being 0 or 1, the function value will always be same. Therefore, we can always find a preimage for a given function value. Hence, this is also a <u>non-one-way</u> function.

2.4 Existence of OWFs

There is no unconditional proof of the existence of OWFs but there is a strong belief that they do. Therefore, the existence of OWF is a conjecture.

- The existence of OWF will imply $\mathcal{P} \neq \mathcal{NP}$ as the former suggests every PPT must fail to solve *almost always* for any random input and the latter suggests every PPT algorithm must fail to solve at least for one input. The vice-versa is not true as being $\mathcal{NP} - complete$ is not enough to be a candidate OWF. Hence, believing OWFs exist is much more than believing $\mathcal{P} \neq \mathcal{NP}$.

- $\mathcal{P} = \mathcal{NP}$ imply non-existence of OWF because if $\mathcal{P} = \mathcal{NP}$, then all the PPT algorithms will always be able to solve the problem and hence the existence of OWF will not be possible. However, it is easy to see that the vice-versa is not true, i.e., non-existence of OWF does not imply $\mathcal{P} = \mathcal{NP}$.

Conclusion

- OWF $\Rightarrow \mathcal{P} \neq \mathcal{NP}$
- $\mathcal{P} \neq \mathcal{NP} \not\Rightarrow$ OWF
- no OWF $\not\Rightarrow \mathcal{P} = \mathcal{NP}$
- $\mathcal{P} = \mathcal{NP} \Rightarrow$ no OWF

2.5 Candidate OWFs

The assumption of the existence of OWF is based on some very natural computational problems that have yet to yield polynomial-time algorithms. Some of these problems, that also form our candidate OWFs are:

1. *Subset-Sum Problem*: This is defined by

$$f(x_1, ..., x_n, J) = (x_1, ..., x_n, \Sigma_{j \in J} x_j)$$

   where all x_i's are of length n, and J is a subset of $\{1, ..., n\}$. Note that when given an image $(x_1, ..., x_n, y)$ of this function, the task of inverting it is exactly that of finding a subset J' of $\{1, ..., n\}$ such that $\Sigma_{j \in J} x_j = y$.

2. *Integer Factorization*: This problem relates to the difficulty of finding the prime factors of a number that is the product of long uniformly distributed primes of similar length. This leads us to define the function:

$$f(x, y) = x \cdot y : x, y \in Primes, |x| = |y|$$

2.6 One-Way Permutations (OWP)

We will often be interested in one-way functions with specific properties. One particular category of interest is that of one-way functions that are bijective. We call such functions "one-way permutations".

Function $f : \{0,1\}^* \to \{0,1\}^*$ is *length-preserving* if $|f(x)| = |x|$ for all x, i.e., the size of the image and preimage are the same.

Definition (one-way permutation): *Function f is a OWP if it is a OWF and is length-preserving and has one-to-one mapping.*

Remark An interesting property of OWP is that any value y uniquely determines its preimage x. This is due to the fact that a permutation f is bijection, and so there exists only one preimage. Thus, even though y fully determines x, it is still hard to find x in polynomial-time.

3 Hard-Core Predicates (HCP)

By the definition, a one-way function is hard to invert, i.e., given a value $y = f(x)$, the value of x is unknown to any polynomial-time inverting algorithm. Thus, f hides information about x, even when $f(x)$ is known. But, this does not mean that nothing about x can be determined from $f(x)$. Indeed, an OWF may yield a lot of information about its input, and yet still be hard to invert. Let us see the following example:

Example Let f be an OWF and define $g(x_1, x_2) = (f(x_1), x_2)$, where $|x_1| = |x_2|$. It is easy to see that g is also OWF (else inverting g can be used to invert f), even though it reveals half of its input.

Therefore, we can conclude that f certainly hides *something* about x. This is exactly captured by *hard-core predicate.* A hard-core predicate of a function f is a boolean function $hc : \{0,1\}^* \to \{0,1\}$, outputting a single bit with the following property: If f is one-way, then upon input f(x) it is infeasible to correctly guess hc(x) with any non-negligible advantage above 1/2.

3.1 HCP: Mathematical Formulation

Formally, we can define HCP as follows:

Definition (hard-core predicate): *A polynomial-time computable predicate hc:* $\{0,1\}^* \to \{0,1\}$ *is called a hard-core of a function f if for every PPT algorithm $\mathcal{A}$, there exists a negligible function **negl** such that*

$$Pr[\mathcal{A}(f(x)) = hc(x)]_{x \leftarrow \{0,1\}^n} \leq 1/2 + negl(n) \tag{3}$$

where the probability is taken over the uniform choice of x in $\{0,1\}^n$ *and the random coin tosses of $\mathcal{A}$.*

Remark

1. Note that it is always possible to compute hc(x) correctly with probability 1/2 by just randomly guessing it.
2. HCP may exist even for functions that are <u>not</u> one-way. For example, let f be the following non-OWF:
$$f(x_1, ..., x_n) = (x_1, ..., x_{n-1})$$
then HCP for f can be defined as follows:
$$hc(x_1, ..., x_n) = x_n$$
It is easy to see that for a random x, given $f(x_1, ..., x_n)$, $hc(x_1, ..., x_n)$ can be guessed with probability 1/2. Therefore, f need not be one-way to define HCP, though we are not interested in such functions.

HCP for Permutations

Let $f : \{0,1\}^* \to \{0,1\}^*$ be a permutation and let $hc : \{0,1\}^* \to \{0,1\}$ be a boolean function. Then hc is a hard-core predicate for the permutation f if for every PPT algorithm $\mathcal{A}$, there exists a negligible function ***negl*** *such that*

$$Pr[\mathcal{A}(f(x)) = hc(x)]_{x \leftarrow \{0,1\}^n} \leq 1/2 + negl(n)$$

Remark A hard-core predicate cannot exist for a permutation that is not one-way. In fact, this is true in general for any *one-one function* because for such functions, the value $f(x)$ fully determines x. Now, if f is not one-way, then it can be inverted, revealing the unique preimage x. Thus, $hc(x)$ can be computed from $f(x)$ with some non-negligible advantage.

Finding HCPs for OWF/OWP is not Simple

Consider a for a moment the candidate hard-core predicate defined as $hc(x) = \bigoplus_{i=1}^{n} x_i$ where $x_1, ..., x_n$ denote the bits of x. The intuition behind why this function "should" be a hard-core predicate is that if f cannot be inverted, then $f(x)$ must hide at least one of the bits x_i of its preimage. Then, the exclusive-or of all of the bits of x must be hard to compute. Despite its appeal, this argument is incorrect. Specifically, given a one-way function f, define the function $g(x) = (f(x), \bigoplus_{i=1}^{n} x_i)$. It is not hard to show that g is one-way. However, it is clear that $g(x)$ does not hide the value of $hc(x) = \bigoplus_{i=1}^{n} x_i$, because this is part of its output. Therefore, $hc(x)$ is not always hard-core predicate. Actually, it can be shown that for every given predicate hc, there exists a one-way function for which hc is not a hard-core of f.

3.2 HCP from every OWF

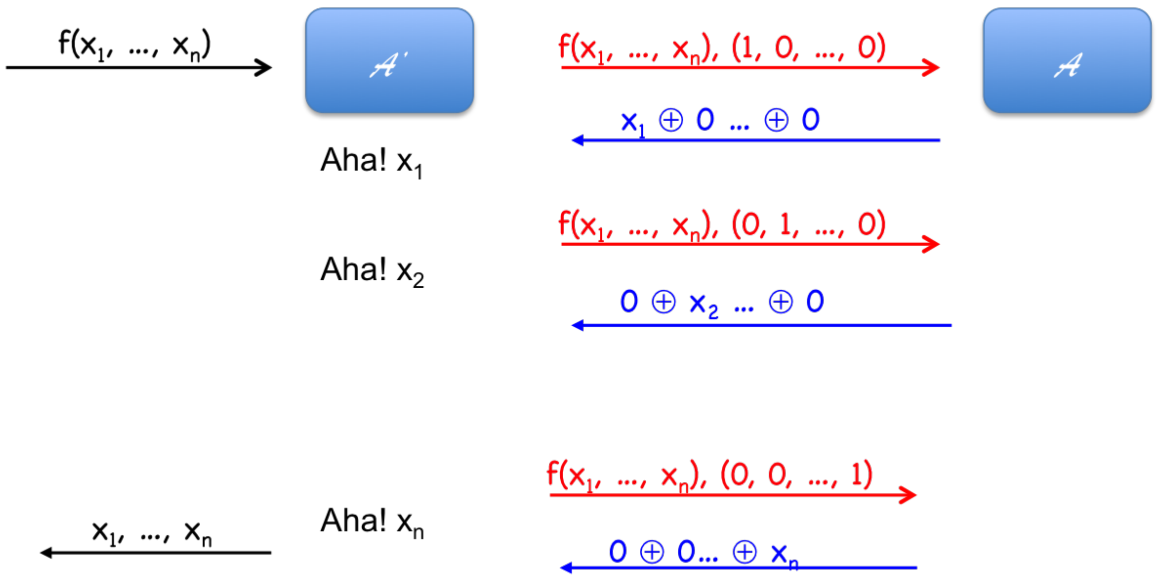
There is no proof yet that HCP exists for a given OWF. However given a OWF f, we can construct another OWF g and its HCP. Though, this is a weaker statement, yet it suffices our purpose. We have the following theorem:

Theorem 1 *(Goldreich-Levin): Let f be an OWF/OWP and define $g(x, r) = (f(x), r)$. Then, the function $hc(x, r) = \bigoplus_{i=1}^{n} x_i \cdot r_i$, where $x = x_1, ..., x_n$ and $r = r_1, ..., r_n$, is a hard-core predicate of g.*

We now proceed to prove the above theorem. Due to the complexity of the proof, we have divided the proof in three stages beginning with the most simplistic case and ending with the full proof. The complete proof is quite complex, which is why as part of this course, we shall discuss only the first two stages.

3.2.1 The Most Simplistic Case

We shall first show that if there exists a PPT adversary $\mathcal{A}$ that always successfully guesses $hc(x, r)$ given $(f(x), r)$, then it is possible to invert f in polynomial-time Given the assumption that f is a one-way permutation (for simplicity we are taking OWP; for OWF also the proof will be same), it follows that no such adversary $\mathcal{A}$ exists.



f(x1, ..., xn)
A'
Aha! x1
Aha! x2
Aha! xn
x1, ..., xn
f(x1, ..., xn), (1, 0, ..., 0)
x1 ⊕ 0 ... ⊕ 0
f(x1, ..., xn), (0, 1, ..., 0)
0 ⊕ x2 ... ⊕ 0
f(x1, ..., xn), (0, 0, ..., 1)
0 ⊕ 0... ⊕ xn
A

Figure 2: Simple Case Game

Proposition 2 *Let f and hc be as in the above theorem. If there exists a PPT adversary $\mathcal{A}$ such that*

$$Pr[\mathcal{A}(f(x),r) = hc(x,r)]_{x,r\leftarrow\{0,1\}^n} = 1, \forall n$$

then there exists a PPT adversary $\mathcal{A}'$ such that

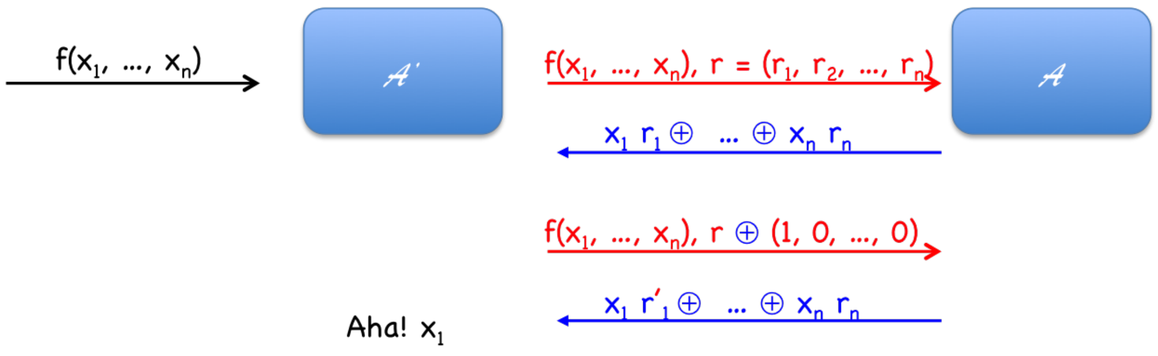
$$Pr[\mathcal{A}'(f(x)) = x]_{x\leftarrow\{0,1\}^n} = 1, \forall n$$

Proof Let us construct an adversary $\mathcal{A}'$ as follows. Let e^i denote the length-n vector that contains zeroes in all positions, except for the i^{th} position where it contains one. Then, upon input y, adversary $\mathcal{A}'$ invokes $\mathcal{A}$ with input (y, e^i) for $i = i, ..., n$. That is, for $y = f(x)$, $\mathcal{A}'$ effectively invokes $\mathcal{A}$ with input $g(x, e^i)$. Since $\mathcal{A}$ always succeeds in guessing hc, we have that for each i, $\mathcal{A}$ outputs $hc(x, e^i)$. However,

$$hc(x,e^i) = \bigoplus_{i=i}^{n} x_j \cdot e^i_j = x_i \cdot e^i_j + \bigoplus_{j\neq i} x_j \cdot e^i_j = x_i$$

where the last equality is because $e^i_i = 1$, but for all $j \neq i$ it holds that $e^i_j = 0$. We therefore conclude that $\mathcal{A}'$ obtains the bits $x_1, ..., x_n$. By the assumption on $\mathcal{A}$, we have that $f(x) = y$, and so $\mathcal{A}'$ successfully inverts f. Figure 2 denotes this game. ■

Of course, this proof is true only when we have an adversary that can always invert the function, which we know is not true for one-way functions. Also, the power of adversary can be used only when $r's$ are taken randomly and the above experiment does not take $r's$ randomly. Therefore, we take a step ahead in our proof by moving closer to our ultimate objective.



$f(x_1, \ldots, x_n)$
$\mathcal{A}'$
$f(x_1, \ldots, x_n), r = (r_1, r_2, \ldots, r_n)$
$\mathcal{A}$
$x_1 r_1 \oplus \ldots \oplus x_n r_n$
$f(x_1, \ldots, x_n), r \oplus (1, 0, \ldots, 0)$
$x_1 r'_1 \oplus \ldots \oplus x_n r_n$
Aha! x_1

Figure 3: Moderate Case Game

3.2.2 The Moderately Simple Case

Here we will show that it is hard to guess $hc(x,r)$ with probability that is non-negligibly greater than $3/4$. For this, we shall prove the following:

Proposition 3 *Let f and hc be as in the theorem. If there exists a PPT adversary $\mathcal{A}$ and a polynomial $p(.)$ such that for infinitely many $n's$*

$$Pr[\mathcal{A}(f(x),r) = hc(x,r)]_{x,r \leftarrow \{0,1\}^n} \geq 3/4 + 1/p(n)$$

then there exists a PPT adversary $\mathcal{A}'$ such that for infinitely many $n's$

$$Pr[\mathcal{A}'(f(x)) = x]_{x \leftarrow \{0,1\}^n} \geq 1/4p(n)$$

Proof This proof is based on the observation that for every $r \in \{0,1\}^n$, the values $hc(x, r \bigoplus e^i)$ and $hc(x,r)$ together can be used to derive the i^{th} bit of x. This can be shown as follows:

$$hc(x,r) \bigoplus hc(x, r \bigoplus e^i) = (\bigoplus_{j=1}^{n} x_j \cdot r_j) \bigoplus (\bigoplus_{j=1}^{n} x_j \cdot (r_j \bigoplus e^i)) = x_i \cdot r_i \bigoplus x_i \cdot (r_i \bigoplus 1) = x_i$$

where the second equality is because for all $j \neq i$, the value $x_j \cdot r_j \bigoplus x_j \cdot r_j$ appears and therefore is canceled out. This game is shown in Figure 3. Note that here the $r's$ in both the queries are random (though not independent). But the success of this experiment requires that both the queries return correct answers, which might not be true as the success probability of the adversary is < 1. To proceed, let us first state the following two lemmas

Lemma 4 *If we have $\mathcal{A}$, $p(n)$ such that for infinitely many $n's$*

$$Pr[\mathcal{A}(f(x),r) = hc(x,r)]_{x,r \leftarrow \{0,1\}^n} \geq 3/4 + 1/p(n)$$

then, then there exists a set $S \subseteq \{0,1\}^n$ of size $2^n/2p(n)$ such that for every x in S and every i,

$$Pr[\mathcal{A}(f(x),r) = hc(x,r) \wedge \mathcal{A}(f(x), r \bigoplus e^i) = hc(x, r \bigoplus e^i)]_{r \leftarrow \{0,1\}^n} \geq 1/2 + 1/p(n)$$

The above lemma is simply trying to say that, if we have an adversary $\mathcal{A}$ as in our proposition, then the probability that the two queries in the game will be correct for a random choice of r (that is, the probability of guessing i^{th} bit of x correctly) is greater than or equal to $1/2 + 1/p(n)$ for every x in the set S.

Lemma 5 *If we have $\mathcal{A}$, $p(n)$, $S \subseteq \{0,1\}^n$ of size $2^n/2p(n)$ such that for every x in S and every i,*

$$Pr[\mathcal{A}(f(x), r) = hc(x, r) \wedge \mathcal{A}(f(x), r \bigoplus e^i) = hc(x, r \bigoplus e^i)]_{r \leftarrow \{0,1\}^n} \geq 1/2 + 1/p(n)$$

then we can construct an adversary $\mathcal{A}'$ such that

$$Pr[\mathcal{A}'(f(x)) = x]_{x \leftarrow \{0,1\}^n} \geq 1/4p(n)$$

Note that if we prove lemma 4 and lemma 5, then our proposition will be proved.

We shall first prove lemma 5 in three steps, that is we shall state and prove three claims that will prove the lemma.

Proof The claims are as follows:

Claim 6 *There exists an adversary $\mathcal{A}'$ such that $Pr[\mathcal{A}'$ correctly guesses i^{th} bit of $x] \geq 1 - 1/2n$ for every x in S.*

Claim 7 *There exists an adversary $\mathcal{A}'$ such that $Pr[\mathcal{A}'$ correctly guesses entire $x] \geq 1/2$ for every x in S.*

Claim 8 *There exists an adversary $\mathcal{A}'$ such that $Pr[\mathcal{A}'(f(x)) = x]_{x \leftarrow \{0,1\}^n} \geq 1/4p(n)$*

The last claim is precisely what we need to prove for lemma 5. Let us first assume that the claim 6 is true. Given that, we shall first prove claim 7.

Proof We have,
$Pr[\mathcal{A}'$ correctly guesses i^{th} bit of $x] \geq 1 - 1/2n$ for every x in S
$\Rightarrow Pr[\mathcal{A}'$ incorrectly guesses i^{th} bit of $x] \leq 1/2n$.
Applying Union-Bound on this we get,
$Pr[\mathcal{A}'$ incorrectly guesses any bit of $x] \leq 1/2n + 1/2n...$ n times $= 1/2$
$\Rightarrow Pr[\mathcal{A}'$ correctly guesses entire $x] \geq 1/2$ ■

Now, given claim 7, we can prove claim 8 as follows:

Proof We can easily see that,
$Pr[\mathcal{A}'(f(x)) = x]_{x \leftarrow \{0,1\}^n}$
$\geq Pr[\mathcal{A}'(f(x)) = x | x \in S]_{x \leftarrow \{0,1\}^n} \cdot Pr[x \in S]_{x \leftarrow \{0,1\}^n}$
$\geq 1/2 \cdot 1/2p(n)$ (from claim 7)
$= 1/4p(n)$ ■

Now, proving claim 6 simply requires us to show how can we improve the probability of $\mathcal{A}'$ for guessing x_i so that it is correct with probability at least $1 - 1/2n$. The idea behind the improved procedure is to run the original procedure many times. Then, since the correct answer is obtained more often than the correct one, it holds that over many trials the majority result will b e correct with high probability. This can be proven using Chernoff bound. The Chernoff bound gives us:

$Pr[\mathcal{A}'$ incorrectly guesses i^{th} bit of $x] \leq e^{m/2(p(n))^2}$

Here, m denotes the number of trails. So by equating $e^{m/2(p(n))^2} = 1/2n$, we can get m. Hence, using the above Probabilistic Boosting Technique, we get the claim 6 proved. Therefore, we conclude that the lemma 5 has been proven. ■

Now, we need to prove lemma 4 in order to finally prove our proposition. To prove lemma 4 we shall make two claims and proving them, we shall complete our proof for the lemma.

Proof The claims are as follows:

Claim 9 *There exists a set $S \subseteq \{0,1\}^n$ of size $2^n/2p(n)$ such that for every x in S*

$$Pr[\mathcal{A}(f(x), r) = hc(x, r)]_{r \leftarrow \{0,1\}^n} \geq 3/4 + 1/2p(n)$$

Claim 10 *There exists a set $S \subseteq \{0,1\}^n$ of size $2^n/2p(n)$ such that for every x in S and every i,*

$$Pr[\mathcal{A}(f(x), r) = hc(x, r) \wedge \mathcal{A}(f(x), r \bigoplus e^i) = hc(x, r \bigoplus e^i)]_{r \leftarrow \{0,1\}^n} \geq 1/2 + 1/p(n)$$

Let us assume claim 9 holds, then let us first prove claim 10.

Proof We have,
$Pr[\mathcal{A}(f(x), r) = hc(x, r)]_{r \leftarrow \{0,1\}^n} \geq 3/4 + 1/2p(n)$
$\Rightarrow Pr[\mathcal{A}(f(x), r) \neq hc(x, r)]_{r \leftarrow \{0,1\}^n} \leq 1/4 - 1/2p(n)$
Similarly, we can say
$Pr[\mathcal{A}(f(x), r \bigoplus e^i) \neq hc(x, r \bigoplus e^i)]_{r \leftarrow \{0,1\}^n} \leq 1/4 - 1/2p(n)$
Applying Union Bound, we get
$Pr[\mathcal{A}(f(x), r) \neq hc(x, r) \vee \mathcal{A}(f(x), r \bigoplus e^i) \neq hc(x, r \bigoplus e^i)]_{r \leftarrow \{0,1\}^n} \leq (1/4 - 1/2p(n)) + (1/4 - 1/2p(n)) = 1/2 - 1/p(n)$
$\Rightarrow Pr[\mathcal{A}(f(x), r) = hc(x, r) \wedge \mathcal{A}(f(x), r \bigoplus e^i) = hc(x, r \bigoplus e^i)]_{r \leftarrow \{0,1\}^n} \geq 1/2 + 1/p(n)$ ■

Now, we shall prove claim 9 to finally prove our proposition.

Proof
$Pr[\mathcal{A}(f(x), r) = hc(x, r)]_{x, r \leftarrow \{0,1\}^n}$
$= 1/2^n \Sigma_x Pr[\mathcal{A}(f(x_i), r) = hc(x_i, r)]_{r \leftarrow \{0,1\}^n}$

$= 1/2^n(\Sigma_{x \in S} Pr[\mathcal{A}(f(x_i), r) = hc(x_i, r)]_{r \leftarrow \{0,1\}^n} + \Sigma_{x \notin S} Pr[\mathcal{A}(f(x_i), r) = hc(x_i, r)]_{r \leftarrow \{0,1\}^n})$
$\leq |S|/2^n + 1/2^n \Sigma_{x \notin S}(3/4 + 1/2p(n))$
$\leq |S|/2^n + (3/4 + 1/2p(n))$
We have, $Pr[\mathcal{A}(f(x), r) = hc(x, r)]_{x, r \leftarrow \{0,1\}^n} \geq 3/4 + 1/p(n)$
Therefore,
$|S|/2^n + (3/4 + 1/2p(n)) \geq 3/4 + 1/p(n)$
which gives, $|S| \geq 2^n/2p(n)$ ■

■

■

This proves our proposition. Due to complexity of the full proof, we shall not prove the last stage of the theorem.

4 Conclusion

In this lecture we studied that given an OWF/OWP f, we can construct another OWF/OWP g and its hard-core predicate hc. In the next lecture we shall see that if we have such a g and hc, then we can construct a PRG $G : \{0,1\}^n \rightarrow \{0,1\}^{n+1}$ from them and give a formal proof that the G obtained from our construction is indeed a PRG. Also, we shall extend our construction by generating another PRG $\{0,1\}^n \rightarrow \{0,1\}^{n+l(n)}$ using G and for this construction also we will prove that the generator thus obtained is indeed a PRG.

5 References

1. Jonathan Katz, Yehuda Lindell. *Introduction to modern cryptography.* CRC Press, 2014.

2. Arpita Patra. http://drona.csa.iisc.ernet.in/ arpita/Cryptography16.html