CSA E0 235: Cryptography 1st March, 2016

Scribe for Lecture 12

Instructor: Arpita Patra *Submitted by: Prateek Yadav*

1 Today's Lecture

In today's Lecture we will Recall One-Way Functions(OWF) & One-way Permutations(OWP). We will look at the questions that, Do OWF/OWP exist? We will look at some candidate OWF and some Existing functions which are supposed to be One-way till Date.

After this we will define the notion of *Hard-Core Predicates* of OWF, It's Definition, Non-triviality and Partial Proof of the Goldreich-Levin theorem which states that if there exists a one-way function, then there exists a one-way function with *hard-core predicate*. The Goldreich- Levin theorem can be used to show that existence of a one-way function implies the existence of a PRG with expansion factor $n + 1$.

2 One-Way Functions(OWF)

A one-way function $f : \{0,1\}^* \to \{0,1\}^*$ is easy to compute, yet hard to invert. Given a *random* value of the function, it should be impossible for an algorithm that runs in polynomial time to find the pre-image of the given value.

Definition. A function $f : \{0,1\}^n \to \{0,1\}^n$ is said to be *one-way* if it is polynomially computable and for every PPT algorithm $\mathcal{A}$, there is a negligible function negl(n) such that

$$\Pr[\mathcal{A}(f(x), 1^n) \in f^{-1}(f(x))] \leq \text{negl}(n) \text{ for } x \text{ chosen uniformly from } \{0,1\}^n$$

If an adversary $\mathcal{A}$with unbounded computational power is considered, then no one-way functions would exist, since $\mathcal{A}$ could brute-force over the domain to find a pre-image for any given input.

We can create a game based on this definition $\mathsf{Invert}_{\mathcal{A},f}(n)$:

1. Choose a value x at random from $\{0,1\}^n$ and compute $y = f(x)$ and gives this value to an adversary $\mathcal{A}$.
2. $\mathcal{A}$ returns a value x' back.
3. If $f(x') = y$, then the game outputs 1, else the game outputs 0.
4. f is *one-way* if it is polynomially computable and for every probabilistic polynomial-time adversary $\Pr[\mathsf{Invert}_{\mathcal{A},f}(n) = 1] \leq \text{negl}(n)$.

One-way functions are widely believed to exist because polynomial time inverting algorithms have not been found for several functions which are considered to be one-way.

1. **Integer Factorisation:** If x and y are two equal length primes, then define $f(x, y) := xy$.
2. Define $f : \{0,1\}^n \to \{0,1\}^n$ by $f(x_1 \dots x_n) := x_1 x_2 \dots x_{n-1}$. f is not an OWF since for the adversary $\mathcal{A}$ which returns $x_1 x_2 \dots x_{n-1} \,||\, 0$ upon being given the value $x_1 \dots x_n$, $\Pr[\mathsf{Invert}_{\mathcal{A},f}(n) = 1] = 1$.
3. Suppose $f : \{0,1\}^n \to \{0,1\}^n$ is polynomially computable and for every PPT algorithm $\mathcal{A}$, there is a negligible function $\mathrm{negl}(n)$ such that
$$\Pr[\mathcal{A}(f(x), 1^n) \in f^{-1}(f(x))] \begin{cases} \leq \mathrm{negl}(n) & \text{if } n \text{ is even} \\ > \frac{1}{n^{100}} & \text{if } n \text{ is odd} \end{cases}$$
4. **Subset Sum:** Let $S = \{x_1, x_2, \dots, x_n\}$ be a set of equal length integers and let $\mathfrak{S}$ be its power set. Then define $f : \mathfrak{S} \to \{0,1\}^n$ by $f(J) := \sum_{x \in J} x \pmod{2^n}$.

Amongst the four constructions above we have shown that 2 and 3 are not OWF and can be inverted, while the first and the fourth constructions are suitable to be called a OWF.

For f to be an OWF, every PPT algorithm must fail to solve *almost every* instance of the inversion problem. We have no proof that a one-way function exists. If the inversion of a function f is NP-complete for example, it does not immediately implies that f is an OWF, for the inversion problem to be NP-complete, we only require that for every PPT algorithm there is a *single* instance of the problem which it fails to solve. So the follwing results hold true:

1. Existence of an OWF $\Rightarrow \mathsf{P} \neq \mathsf{NP}$.
2. $\mathsf{P} \neq \mathsf{NP} \not\Rightarrow$ existence of an OWF.
3. $\mathsf{P} = \mathsf{NP} \not\Rightarrow$ No OWF exist.
4. No OWF exist $\not\Rightarrow \mathsf{P} = \mathsf{NP}$.
5. Belief in the existence of an OWF is stronger than than the belief that $\mathsf{P} \neq \mathsf{NP}$.

It is not known whether the existence of a one-way function implies the existence of a one-way permutation.

The theorems we are going to prove for one-way permutations will also be true for one-way functions but the proof will be a bit more complicated to work with.

Definition. A one-way function that is length-preserving and one-to-one is called a one-way permutation. If f is a one-way permutation, then any value y has a unique preimage $x = f^{-1}(y)$

3 Hard-Core Predicates

By OWF one might get the impression that nothing about x can be determined from $f(x)$ in polynomial time. This is not necessarily the case. Indeed, it is possible for $f(x)$ to "leak" a lot of information about x even if f is one-way. Our Goal here is to construct a PRG G using a one-way function f. A natural attempt at this is to let $G(s) = f(s) \mid\mid r(s)$ where r is a Boolean function. If this is a PRG, it must be that $r(x)$ cannot be efficiently determined from $f(x)$. If we can find such a r, we might be able to construct a PRG with expansion factor of one and then run it polynomial number of times to compute a PRG with $p(n)$ expansion factor.

Suppose f is an OWF. Given $f(x)$, it might be possible to determine a large portion of x even though f is an OWF:

Illustration: If f is an OWF, then so is g, if $g(x, r) := (f(x), r)$ and $|x| = |r|$.

On the other hand, since f is an OWF, it is difficult to determine x *completely* given $f(x)$. Therefore there is something about x that is hidden even if know $f(x)$. To formalise this expectation we introduce the concept of Hard-core predicates.

A $hard - corepredicate$ is a single bit of information about x than every adversary finds difficult to guess when given access to $f(x)$ only.

Definition. A function $\mathsf{hc} : \{0,1\}^n \to \{0,1\}$ is said to be a *hard-core predicate* for a function of f if it is polynomially computable and for every PPT algorithm $\mathcal{A}$, there is a negligible function $\text{negl}(n)$ such that

$$\Pr[\mathcal{A}(f(x), 1^n) = \mathsf{hc}(x)] \leq \frac{1}{2} + \text{negl}(n) \text{ for } x \text{ chosen uniformly from } \{0,1\}^n$$

For Hard-Core predicate of a function to exist the function f need not be one way, many non OWF's have hard core predicates.

Illustration: Consider the non-OWF $f : \{0,1\}^n \to \{0,1\}^n$ defined by $f(x_1 \dots x_n) := x_1 x_2 \dots x_{n-1}$. Then $\mathsf{hc}(x) := x_n$ is a hard-core predicate for f.

If a function f is bijective and has a hard-core predicate, then f must be a one-way permutation because if it were not one-way, one could invert to find the unique pre-image and then use it to compute the hard-core predicate for the function f at x.

Finding a hard-core predicate for a OWF is not easy. The illustration below will enlighten this fact.

Illustration: If f is an OWF, then so is g, if $g(x_1 \dots x_n) = f(x_1 \dots x_n) \mid\mid x_1 \oplus \dots \oplus x_n$. Hence h defined by $h(x_1 \dots x_n) := x_1 \oplus \dots \oplus x_n$ is not a hard-core predicate for g. In fact, the very same construction allows us to show that given any Boolean function h, there exists a OWF g such that h is not a hard-core predicate for g.

The fact that whether every one-way function has a hard-core predicate associated with it

is not proven, but we can prove that if there exist a one-way function, then we can construct a one-way function with hard-core predicate. We will prove this later, that if f is a one-way function with hard-core predicate hc, then $G(s) = f(s) \mid\mid \mathsf{hc}(s)$ is a PRG with expansion factor of one. This will allow us to show that if we assume existance of OWF then it is sufficient to prove all the assumption that we have made during the course related to PRF, PRP, PRG etc.

4 Goldreich-Levin Theorem

In this section we will state the Goldreich-Levin Theorem and then provide a partial proof for it. The theorem is one of the toughest and most complicated proof involved in the entire set of Cryptographic proof's.

Theorem (Goldreich-Levin Theorem). Let f be a one-way permutation and define g by $g(x, r) = (f(x), r)$ where $x = x_1 \dots x_n$ and $r = r_1 \dots r_n$. Then the function $\mathsf{hc}(x, r) := r_1 x_1 \oplus \cdots \oplus r_n x_n$ is a hard-core predicate for g.

One can attempt to prove the theorem by proving the contrapositive, i.e., if hc is not a hard-core predicate, then f is not an OWP. To show this, one must demonstrate that if there exists a PPT algorithm $\mathcal{A}$ such that $\Pr[\mathcal{A}(f(x), r) = \mathsf{hc}(x, r)] \geq \frac{1}{2} + \frac{1}{p(n)}$, then there exists a PPT algorithm $\mathcal{A}_1$ such that $\Pr[\mathcal{A}_1(f(x), 1^n) = x] \geq \frac{1}{p_1(n)}$. We will not be able to prove this. Instead we shall prove the following weaker statement:

Theorem. If there exists a PPT algorithm $\mathcal{A}$ such that $\Pr[\mathcal{A}(f(x), r) = \mathsf{hc}(x, r)] \geq \frac{3}{4} + \frac{1}{p(n)}$, then there exists a PPT algorithm $\mathcal{A}_1$ such that $\Pr[\mathcal{A}_1(f(x), 1^n) = x] \geq \frac{1}{p_1(n)}$.

Proof. First consider the following 'toy' problem: If there was an $\mathcal{A}$ which could *always* compute $\mathsf{hc}(x, r)$ given $f(x)$ and r, then could f possibly be one way?

No. Consider $\mathcal{A}_1$ which acts as follows: it gets the values of $f(x)$, $x = x_1 \dots x_n$, from its verifier and sends $(f(x), (1, 0, \dots, 0))$ to $\mathcal{A}$. Then $\mathcal{A}$ would compute $\mathsf{hc}(x, r) = x_1$ and send it back to $\mathcal{A}_1$. Similarly, $\mathcal{A}_1$ can determine the other bits of x and send x to his verifier, so f cannot be an OWP.

But we only have an $\mathcal{A}$ which can determine $\mathsf{hc}(x, r)$ with a certain probability. So what can we do instead? Consider an $r \in \{0, 1\}^n$ and let r^i be r with the i^{th} bit flipped. It is straight forward to check that $x_i = \mathsf{hc}(x, r^i) \oplus \mathsf{hc}(x, r)$. Hence given $\mathsf{hc}(x, r^i)$ and $\mathsf{hc}(x, r)$ correctly, $\mathcal{A}_1$ can determine x_i.

Consider an x for which a random choice of r results in $\mathcal{A}$ returning $\mathsf{hc}(x, r)$ and $\mathsf{hc}(x, r^i)$ $(\forall\ i)$ correctly with a 'high' probability. Then $\mathcal{A}_1$ can compute each x_i and intuitively, should be able to invert $f(x)$ with a high probability. If we could show that there are sufficiently many x for which this is true, then $\mathcal{A}_1$ can invert f with a non-negligible probability and we would be done. In this direction, we have the following lemma.

Lemma. If there exists a PPT algorithm $\mathcal{A}$ such that $\Pr[\mathcal{A}(f(x), r) = \mathsf{hc}(x, r)] \geq \frac{3}{4} + \frac{1}{p(n)}$, then there exists a set $S \subset \{0,1\}^n$ of size at least $\frac{2^n}{2p(n)}$ such that for every x_0 in S and every i

$$\Pr[\mathcal{A}(f(x_0), r) = \mathsf{hc}(x_0, r) \text{ and } \mathcal{A}(f(x_0), r^i) = \mathsf{hc}(x_0, r^i)] \geq \frac{1}{2} + \frac{1}{p(n)}$$

where the probability is taken over the random choice of r and any randomness used by $\mathcal{A}$.

Proof. Consider the *maximal*[1] set S such that for every x_0 in S

$$\Pr[\mathcal{A}(f(x_0), r) = \mathsf{hc}(x_0, r)] \geq \frac{3}{4} + \frac{1}{2p(n)}$$

for a random choice of r. A maximal set exists, since there is at least one such set and the number of sets is finite. Then note the following:

$$\begin{aligned}
\Pr[\mathcal{A}(f(x), r) = \mathsf{hc}(x, r)] &= \frac{1}{2^n} \sum_{x_0 \in \{0,1\}^n} \Pr[\mathcal{A}(f(x_0), r) = \mathsf{hc}(x_0, r)] \\
&= \frac{1}{2^n} \sum_{x_0 \in S \text{ and } x_0 \notin S} \Pr[\mathcal{A}(f(x_0), r) = \mathsf{hc}(x_0, r)] \\
&\leq \frac{|S|}{2^n} + \frac{1}{2^n} \sum_{x_0 \notin S} \frac{3}{4} + \frac{1}{2p(n)} \\
&\leq \frac{|S|}{2^n} + \frac{3}{4} + \frac{1}{2p(n)}
\end{aligned}$$

But $\Pr[\mathcal{A}(f(x), r) = \mathsf{hc}(x, r)] \geq \frac{3}{4} + \frac{1}{p(n)}$. Hence $\frac{|S|}{2^n} + \frac{3}{4} + \frac{1}{2p(n)} \geq \frac{3}{4} + \frac{1}{p(n)} \implies |S| \geq \frac{2^n}{2p(n)}$. So there is a set S of size at least $\frac{2^n}{2p(n)}$ such that for every x_0 in S, $\Pr[\mathcal{A}(f(x_0), r) = \mathsf{hc}(x_0, r)] \geq \frac{3}{4} + \frac{1}{2p(n)}$

Now observe the following: for every x_0 in S $\Pr[\mathcal{A}(f(x_0), r) \neq \mathsf{hc}(x_0, r)] \leq \frac{1}{4} - \frac{1}{2p(n)}$ where the probability is taken over the random choice of r as well. This means, in particular, that we can replace r with r^i in the statement above. It follows that for every $x_0 \in S$

$$\Pr[\mathcal{A}(f(x_0), r) \neq \mathsf{hc}(x_0, r) \text{ or } \mathcal{A}(f(x_0), r^i) \neq \mathsf{hc}(x_0, r^i)] \leq 2\left(\frac{1}{4} - \frac{1}{2p(n)}\right)$$

by the union bound. The required result follows immediately from taking the complement of the above event. □

The lemma implies that for every $x_0 \in S$ $\mathcal{A}_1$ can ask $\mathcal{A}$ just two questions and determine x_i with a probability of at least $\frac{1}{2} + \frac{1}{p(n)}$. But if $\mathcal{A}_1$ were to ask more questions (by choosing different r's) and then pick the value of x_i suggested by the *majority* of the answers, then $\mathcal{A}_1$ could obtain the right answer with a much a higher probability. More formally, we have the following:

[1] This was not mentioned in class, but is needed for the proof that follows and is given in the book.

Lemma. $\mathcal{A}_1$ can determine x_i with a probability of at least $1 - \frac{1}{2n}$ while asking only polynomially many questions.

Proof. Suppose X_i's, $i = 1$ to $i = 2s + 1$, are indicator random variables with probability of success $k = \frac{1}{2} + \frac{1}{p(n)}$. Let $\delta = 1 - \frac{s}{(2s+1)k}$. Note that $\delta > 0$. Further let $\mu = \mathbb{E}\left[\sum X_i\right] = (2s + 1)k$ The probability that the majority of them are 0 is equal to the probability that $\sum X_i \leq s$. But note the following:

$$\begin{aligned}
\Pr\left[\sum X_i \leq s\right] &= \Pr\left[\sum X_i \leq (2s+1)k \cdot \frac{s}{(2s+1)k}\right] \\
&= \Pr\left[\sum X_i \leq \mu \cdot \frac{s}{(2s+1)k}\right] \\
&= \Pr\left[\sum X_i \leq \mu \cdot (1-\delta)\right] \\
&\leq e^{\frac{-\mu\delta^2}{2}}
\end{aligned}$$

where the last inequality follows from an application of the Chernoff bound. But if $s > cp(n)^3$ for some constant c, a straight forward computation shows that $e^{\frac{-\mu\delta^2}{2}} \leq \frac{1}{2n}$. We think of the X_i's as random variables that are lower bounds for the random variables which indicate the probability of $\mathcal{A}_1$ determining a bit of x. $\mathcal{A}_1$ only needs to ask s pairs of questions to determine x_i with a probability of at least $1 - \frac{1}{2n}$. □

Hence there exists $\mathcal{A}_1$ such that the probability that $\mathcal{A}_1$ does not determine x_i correctly is at most $\frac{1}{2n}$. By the union bound, the probability that such an $\mathcal{A}_1$ does not determine x_i correctly *for some* i is at most $n \cdot \frac{1}{2n} = \frac{1}{2}$. Hence there exists an $\mathcal{A}_1$ which can determine the inverse of an element in S with probability at least $\frac{1}{2}$.

Since the size of S is at least $\frac{2^n}{2p(n)}$, a random string from $\{0,1\}^n$ lies in S with probability at least $\frac{1}{2^n} \cdot \frac{2^n}{2p(n)} = \frac{1}{2p(n)}$. Since $\mathcal{A}_1$ can invert an element of S with probability at least $\frac{1}{2}$, $\mathcal{A}_1$ can invert a random string from $\{0,1\}^n$ with probability at least $\frac{1}{4p(n)} = \frac{1}{p_1(n)}$ as required. □

A more sophisticated argument involving the use of an algorithm to construct 'nice' sets can be used to show the full Goldreich-Levin theorem. We do not cover it due to a lack of time.