University of Illinois, Urbana Champaign
CS 598DK Special Topics in Cryptography

Instructor: Dakshita Khurana
Scribe: Ye Zhang
Date: September 4, 2019

LECTURE
3

Hard-core predicate - II, Commitment scheme

3.1 Introduction

In the previous lecture, we introduced OWF (One-Way Function) which is easy to compute but hard to invert. We defined negligible function and Probabilistic Polynomial Time Turing machine to describe "hard" precisely. However, there is one problem left: the output of OWF $f(x)$ is not uniformly random, it can reveal some information about x according to different constructions. So we are curious that is there any feature of x that OWF really hides? In other words, if we give the adversary $f(x)$, is there some $B(x)$ that the adversary cannot predict? In other words, can we find $B(x)$ so that guessing $B(x)$ *even given* $f(x)$ is as difficult as inverting OWF $f(x)$?

In this lecture, we introduce the notion and construction of a *Boolean* Hard-core predicate function $B(x)$, that satisfies the relationship above. Furthermore, we use B(x) to build a coin tossing game, and use commitment scheme to formalize this. Then, we give a formal definition of commitments, and discuss its security properties (binding and hiding). A commitment scheme is one of the most important cryptographic primitives, and it is used in zero-knowledge proof and so on.

3.2 Notation

This set of scribe notes will use the following notations:

- **negl**(k) : Negligible Function
- **nuPPT** : non-uniform Probabilistic Polynomial Time
- $x \xleftarrow{\$} \{0,1\}^k$: uniformly sample a k-bit length binary string
- $< x_1, x_2 >$: inner product of x_1 and x_2

3.3 Hard-core predicate

The intuition of hard-core predicate is that we want to use a OWF to construct a boolean predicate function that outputs a single bit which is hard to predict. In other words, given the output of a OWF, the possibility of predicting the output of the boolean predicate function correctly is at most $\frac{1}{2} + \mathbf{negl}(k)$ [1].

DEFINITION 3.1. A hard-core predicate of a one-way function of a OWF $f\{0,1\}^* \to \{0,1\}^*$ is a boolean predicate $B\{0,1\}^* \to \{0,1\}$ s.t.

- $\forall x$, $B(x)$ is easy to compute (i.e. can be computed in time polynomial in $|x|$).
- $\forall$**nuPPT** A , $\forall c \in \mathbb{N}$ $\exists k_0 \in \mathbb{N}$ s.t. $\forall k > k_0$

$$Pr[A(1^k, f(x)) = B(x) : x \xleftarrow{\$} \{0,1\}^k] = \frac{1}{2} + \mathbf{negl}(k)$$

Notice that we usually use non-uniform adversaries in cryptography.

3.4 Construction of a Hard Core Predicate

Given a OWF $f : \{0,1\}^k \to \{0,1\}^{m(k)}$, we define a new OWF $g : \{0,1\}^{2k} \to \{0,1\}^{m(k)+k}$ as

$$g(x) = f(x_1)||x_2$$

where $x = x_1||x_2, |x| = 2k, |x_1| = |x_2| = k$.

Note that $g(x)$ is still a OWF since inverting $g(x)$ is at least as hard as inverting $f(x_1)$. We construct a hard-core predicate for g as follows:

$$B_g(x) \equiv \langle x_1, x_2 \rangle \pmod 2$$

We state without proof that $B_g(x)$ is a hard-core predicate for g. We don't give a proof of this statement in this lecture, and we refer readers to read the proof from Goldreich-Levin paper [2] or read simplified version from other notes [1]. We just give some comments about how this is proved. The proof proceeds as follows:

Suppose $\exists$**nuPPT** A and a polynomial $p(\cdot)$ s.t.

$$Pr[A(g(x)) = B_g(x)] = \frac{1}{2} + \frac{1}{p(k)}$$

Then, [2] use A to build A' s.t.

$$Pr[A'(g(x)) \in g^{-1}(g(x))] = \frac{1}{\mathsf{poly}(p(k))}$$

However such A' violates the defination of OWF, so we get a contradiction. In fact, according to [1], building A' from A is easiest if A can predict $B_g(x)$ with probability 1, a little harder if A predicts $B_g(x)$ with probability $\frac{3}{4} + \mathbf{negl}(k)$ and finally they analyze the real case where A predicts $B_g(x)$ with probability $\frac{1}{2} + \mathbf{negl}(k)$.

[1] Notice that without any information, the possibility of predicating a single random bit correctly is still $\frac{1}{2}$, we use $\mathbf{negl}(k)$ to allow a very slight computationally bounded adversary, which could, eg, come from inverting the one-way function with probability $\mathbf{negl}(k)$.

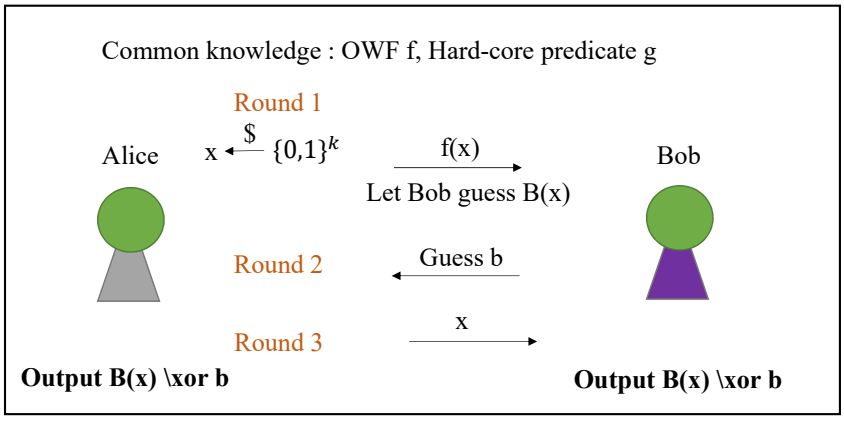

FIGURE 3.1: Using hardcore bit $B(\cdot)$ to build coin tossing

3.5 Application of Hard Core Bit: Coin Tossing

Suppose Alice and Bob live on opposite corners of the world, and want to toss an unbiased coin. Basically, they want to generate a single bit of uniform randomness, that should be 0 with probability (close to) $\frac{1}{2}$, and 1 with probability (close to) $\frac{1}{2}$. They do not want to trust any third party.

Can we design a protocol to ensure that they end up with a bit that is unbiased, as long as one of them is not cheating?

A starting point could be to have both Alice and Bob toss random coins individually, that is, Alice samples $r_A \xleftarrow{\$} \{0,1\}$, and Bob samples $r_B \xleftarrow{\$} \{0,1\}$. Then, the resulting coin could be $r = r_A \oplus r_B$. As long as r_A and r_B are *independent*, and at least one out of r_A and r_B is uniformly random, r is uniformly random.

The problem is how they should communicate this independent randomness to each other. If Alice sends r_A to Bob, Bob can set r_B *depending* on the value r_A (eg, $r_B = (1 \oplus r_A)$), to bias the value r (in this case, Bob can ensure that $r = r_A \oplus r_B = 1$). Similarly, if Bob speaks first, Alice can set r_A to bias r.

But we want to ensure that neither of them can bias the resulting coin, despite the fact that they have to communicate sequentially. So we need Alice to send something that binds her to her r_A, without exactly revealing the value r_A to Bob.

A solution is presented in figure 3.1. Specifically, Alice samples $x \leftarrow \{0,1\}^k$, and sends $f(x)$ to Bob. If Bob guesses $B(x)$ correctly, then both parties output $r = 0$, and otherwise both parties output $r = 1$.

Since Bob cannot guess $B(x)$ given only $f(x)$, this implies that Bob cannot bias the resulting coin.

Note that this scheme has some problems. Alice can still cheats if she knows x and $\overline{x}$ s.t.

$$B(x) \neq B(\overline{x}) \quad but \quad f(x) = f(\overline{x})$$

We need to ensure that Alice cannot change $B(x)$ once she sees Bob's guess. This can be done by ensuring that the one-way function is *injective*. That is, for every y in the image of the one-way function, $f^{-1}(y)$ is unique. This ensures that once Alice sends $y = f(x)$, she cannot find $x' \neq x$ such that $y = f(x')$.

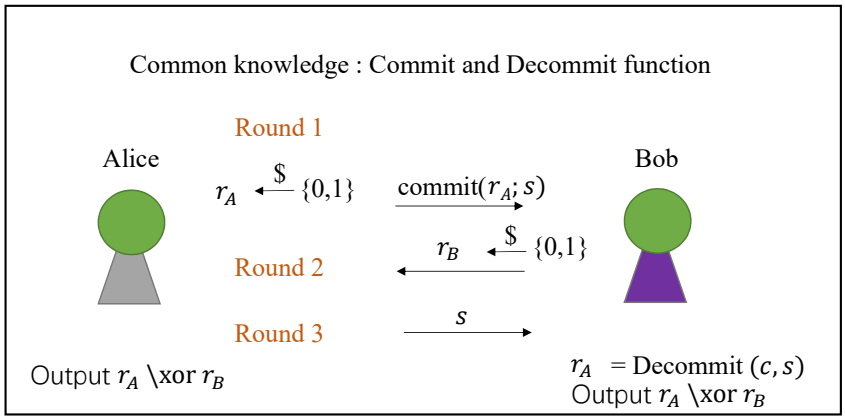

FIGURE 3.2: Using commitment scheme to build coin tossing

3.6 Coin Tossing via Commitments

A different (but equivalent) way to think about solving the coin tossing problem, is via a commitment scheme. Going back to the example where Alice and Bob sample r_A and r_B independently and uniformly at random and try to output $r_A \oplus r_B$, we will consider the following scenario. Let's say Alice could write down r_A and lock it into a box. Then she could send the box to Bob, and Bob would sample and send Alice r_B. Finally, given r_B, Alice would "open" her commitment and Bob would see r_A. They would both output $r = r_A \oplus r_B$.

A commitment scheme helps hide a value and allows the committer to open it anytime she wants. But the committer cannot change her mind once she sends the commitment. Intuitively it acts like a lock box with a key that can be sent later. When we commit to a value, we put the value into the lock box, and this value can't be changed anymore. A coin tossing protocol using a commitment scheme is formalized in Figure 3.2.

Before defining a commitment, we define computational indistinguishability first.

DEFINITION 3.2. (Computational Indistinguishability) Two distributions D_0 and D_1 are said to be computational indistinguishable, if

$$\forall nuPPT\ A \quad |\underset{x \leftarrow D_0}{Pr}[A(1^k, x) = 1] - \underset{y \leftarrow D_1}{Pr}[A(1^k, y) = 1]| \leq neql(k)$$

This is often denoted by $D_0 \approx_c D_1$.

We now give a formal definition of bit commitment scheme:

DEFINITION 3.3. A bit Commitment scheme has two algorithms:

$$Commit(1^k, m; r) \to c \quad (m \in \{0,1\})$$

$$Decommit(1^k, c; r) \to m\ or\ \perp \quad (m \in \{0,1\})$$

The **correctness** of the commitment scheme is:

$$\forall m, \forall r, \quad Decommit(1^k, Commit(1^k, m; r), r) = m$$

It has two properities **binding** and **hiding**.

- **Hiding**:

$$Commit(1^k, 0, r) \approx_c Commit(1^k, 1, r)$$

or equivalently,

$$\forall nuPPT\, A,\ |Pr[A(1^k, Commit(1^k, 0, r)) = 1] - Pr[A(1^k, Commit(1^k, 1, r)) = 1]| = negl(k)$$

- **Binding**:

$$\forall c, \forall r_1, \forall r_2$$
$$if \quad Decommit(1^k, c, r_1) = m_1 \neq \perp, Decommit(1^k, c, r_1) = m_2 \neq \perp$$
$$Then \quad m_1 = m_2$$

So hiding means the distribution of commitment of 0 and 1 are computationally indistinguishable. In other words, the commitment leaks no information about the message to a computationally bounded adversary.

Binding is for guaranteeing that once he sends the commitment, the committer should not be able to change what they committed to. As defined above, binding is a statistical property that holds even for computationally unbounded committers.

3.7 Construction of a Commitment Scheme from Hard Core Bit

Question : Can we use hard-core predicate $B(x)$ to construct a bit commitment ?

A straightforward way is to try to compare the protocols in Figure 3.1 and 3.2. Comparing the two schemes, we know that the injective one-way function $f(x)$ acts like a commitment to the bit $B(x)$ (note: $f(x)$ is not a commitment to x, since $f(x)$ could reveal some bits of x).

To commit to a specific bit $b \in \{0, 1\}$, the committer can just sample x repeatedly until $B(x) = b$, and then output $f(x)$ as a commitment to b. An equivalent way of saying this is to take an injective one-way function f and define, for $m \in \{0, 1\}$,

$$\mathsf{Commit}(m; r) = f(x)||(B(x) \oplus m)$$

$\mathsf{Decommit}(c, r)$ parses $c = (y||z)$ and outputs $(y \oplus B(r))$ if and only if $f(r) = y$

Intuitively, we use $B(x)$ as a one-time pad to hide m.

Hiding. Any nuPPT adversary A that breaks hiding predicts the hardcore bit $B(x)$ given $f(x)$, with probability better than $\frac{1}{2} + \mathsf{negl}(K)$. Therefore, by unpredictability of the hardcore bit, the scheme satisfies hiding.

Binding. Since f is injective, given $c = (y||z)$, there exists a single value x such that $y = f(x)$, and therefore $B(x)$ is also unique. Since $B(x)$ is unique, $z \oplus B(x)$ can be a single unique message.

Acknowledgement

These scribe notes were prepared by editing a light modification of the template designed by Alexander Sherstov.

References

[1] The goldreich-levin theorem : Notes from rafael pass's class. http://www.cs.cornell.edu/courses/cs687/2006fa/lectures/lecture11.pdf.

[2] O. Goldreich and L. A. Levin. A hard-core predicate for all one-way functions. In *Proceedings of the twenty-first annual ACM symposium on Theory of computing*, pages 25–32. ACM, 1989.