18.425/6.875 Introduction to Cryptography

Lecture #22: December 9, 1991
Lecturer: Shafi Goldwasser
Scribe: Atul Shrivastava

Lecture Outline

Last time we studied two-party protocols for Coin Flipping, Key Exchange and Oblivious transfer and realized that secure two party protocols exist in the presence of one way functions. Today, we will adopt a more complexity theoretic view and study a special kind of two-party protocol known as the prover-verifier systems or Interactive Proof systems. We will study the power of Interactive Proof systems from the point of complexity theory. In this model the two parties involved are a Prover and a Verifier. Recently, more powerful form of interactive proof systems have been studied namely two (multi)-prover interactive proof systems. These will be covered in the following lecture.

1 Prover-Verifier Systems

Consider a two-party protocol (parties $A$ and $B$) in which both parties receive a common input (encrypted). Party $A$ is a probabilistic Turing Machine (thus it has access to a random coin or equivalently, it has its own private (unidirectional read-only) tape from which it reads a string of 0’s and 1’s). In addition it can communicate with party $B$ by sending and receiving messages. Figure 1 shows such a two-party system.

Figure 1: Prover-Verifier Model

Notice that we have placed no restriction on the computation power of $B$. The goal of the protocol is that $A$ wishes to decide language $L$. Thus, given input $x$ it has to output it’s decision (accept or reject $x$) depending on whether $x \in L$ or $x \notin L$. Since $A$ can interact with $B$, it can use the power of $B$ to help make it’s decision. But since $B$ can be an arbitrary party, $A$ needs to verify the assertions that $B$ makes during the protocol.

We know that the complexity class $NP$ can be formalized as the set of languages $L$ for which there exists polynomial time function $f$ s.t.

$$\begin{align*}
\exists w : f(x, w) &= T \quad \text{if } x \in L \\
\forall w : f(x, w) &= F \quad \text{if } x \notin L
\end{align*}$$

An Interactive proof system for $NP$ can thus be described in Figure 2.
As is clear, for the class $NP$ only one interaction between the prover and verifier is needed. The prover needs to send to the verifier the witness string $w$, namely a satisfying assignment for input $\phi$ (boolean formula). The verifier can then easily check that the formula is indeed satisfiable using $w$. Clearly the string $w$ is polynomial size and the verifier computation is also polynomial time. In fact, the verifier doesn’t need to toss coins in this system. The verifier above exhibits the polynomial time function $f$ for $L$ to be in $NP$.

**Interactive Proof Systems**

We can generalize the $NP$ prover-verifier system to multi-rounds i.e. the prover and verifier now interact over several rounds. At the end of the protocol, the verifier either accepts or rejects the input based on it’s local computation and the questions it asks the prover and the answers received in response. This scenario is shown in Figure 3.

Note that this model above can be easily formalized where $P$ and $V$ are actually Turing Machines whose computation in round $j$ depends upon the input $x$, the questions ($q_i$) asked by the verifier, the answers ($a_i$) given by the prover for all $j < i$. Moreover, it is also important to note that in general the verifier needs to be a probabilistic Turing Machine, otherwise multi-round interaction doesn’t give additional power when compared to the $NP$ prover-verifier protocol. Since the prover can guess the series of questions $q_i$ asked by the verifier (since verifier is deterministic) and can send just one answer $a = a_1 \ast a_2 \ast \cdots \ast a_t$, where $a_i$ corresponds to $q_i$ asked by the verifier. The verifier then just needs to verify the computation.

**Complexity Class IP**

We now restrict the computation power of the verifier to be a polynomial-time probabilistic Turing Machine. The class of languages accepted by such prover-verifier systems is called $IP$ (for Interactive proofs). More formally,

Let $L \subseteq \{0,1\}^*$. Then $L \in IP$ if there exists a polynomial-time probabilistic Turing Machine $V$ s.t.

$$\begin{align*}
\exists P \ Pr(V \text{ accepts } x) &= 1 \quad \text{if } x \in L \\
\forall P \ Pr(V \text{ accepts } x) &\leq 1/3 \quad \text{if } x \notin L
\end{align*}$$

For cryptographic purposes, we will also define complexity class $ZK$ (for Zero Knowledge) as $ZK = \{L : L \in IP \text{ and } L \text{ has zero knowledge proofs}\}$. The term
zero-knowledge refers to interactive proof protocols in which the prover does not reveal to the verifier anything about the proof showing $x \in L$. Thus it is clear that $ZK \subseteq IP$. We will illustrate a language $L \in ZK$ in the following example.

Example: Let $QR = \{(x, n) : \exists y \text{ such that } x = y^2 \text{ mod } n\}$. It is easy to see that $QR \in NP$, since the short witness of $x$ being a quadratic residue is the number $y$. We present a zero-knowledge interactive protocol in Figure 4 showing that $QR \in ZK$.

By repeating the above protocol hundred times, we get

$$\begin{align*}
Pr(V \text{ accepts } x \in L) &= 1 \\
Pr(V \text{ accepts } x \notin L) &\leq 2^{-100}
\end{align*}$$

Claim: $QR \in IP$.

Proof: Clear from the argument presented above.

Claim: $QR \in ZK$.

Proof: Consider the following Turing Machine $M_V = \text{"On input } (x, n), \text{ let } z \in_R Z_n^* \text{ and } \hat{c} \in_R \{0, 1\}. \text{ If } \hat{c} = 0 \text{ then let } w = z^2 \text{ mod } n \text{ else let } w = (z^2x^{-1}) \text{ mod } n. \text{ If } \hat{c} = c = 0 \text{ then output } z = \sqrt{w}. \text{ If } \hat{c} = c = 1 \text{ then output } z = \sqrt{wx}.\text{"}

Here $c$ is a boolean function computable by a polynomial time deterministic Turing Machine. Moreover, $c$ could depend on $w, x, n$ and past history of the Zero-knowledge protocol presented above. It can be shown that $M_V$ runs in expected polynomial time and has the same distribution on its output as $V$ in the Zero-knowledge protocol. For details of the proof, refer [1].

Based on the above Zero-Knowledge protocol, Shamir suggested the following password scheme:

Let the password be some integer $y$ known only to user $U$ trying to access a computer system $C$. $C$ knows $y^2 \text{ mod } n$. All arithmetic will be modulo $n$ known both to $U$ and $C$. The user will simulate $P$ while the computer simulates the verifier $V$. $U$ will convince $C$ of its correct identity without actually revealing $y$.

The protocol is as follows:

1. Repeat steps 2-5 for hundred times.

2. $U$ generates $z \in_R Z_n^*$ and sends $z^2 \text{ mod } n$ to $C$.

3. $C$ generates $b \in_R \{0, 1\}$ and sends $b$ to $U$.

4. if $b = 0$ then $U$ sends $z$ to $C$, else $U$ sends $z \cdot y \text{ mod } n$ to $C$. 

3
5. Let \( w \) be the integer received by \( C \). \( C \) can easily verify both cases i.e.
   
   (a) if \( b = 0 \) and \( w^2 = z^2 \mod n \) then continue else reject and halt.

   (b) if \( b = 1 \) and \( w^2 = z^2 \cdot y^2 \mod n \) then continue else reject and halt.

6. Accept iff no failure in any iteration.

   It is important to note that even if an evesdropper \( E \) is able to record a valid conversation between the \( U \) and \( C \), the probability of \( E \) being able to break into the system is very small \( (\leq 2^{-100}) \)!!

   We will not address the Zero-Knowledge property of \( IP \) protocols any further. Instead, our next theorem will show the computational power of \( IP \) protocols.

   We know that \( SAT = \{ \phi : \phi \) is a satisfiable boolean formula.\} \) is a complete problem for \( NP \). In fact, it has been shown that \( SAT \in ZK \) if there exist one way functions. However, it is not known whether \( \overline{SAT} \in NP \). In the remaining time, we will show that \( co - NP \subseteq IP \) by showing that \( \overline{SAT} \in IP \).

   **Theorem:** \( \overline{SAT} \in IP \).

   **Proof:** Define \#\( SAT = \{(\phi, k) : \phi \) has exactly \( k \) satisfying assignments \} \). We will in turn show that \#\( SAT \in IP \). The theorem follows. \( \square \)

   **Theorem:** \#\( SAT \in IP \).

   **Proof:** We assume wlog that \( \phi \) is in 3-CNF form. Given input \( (\phi, k) \), arithmetize \( \phi \) by replacing logical operators in \( \phi \) by arithmetic operators over a finite integer field \( F \) to obtain \( f_\phi \) as follows:

   - Replace \( x_1 \land x_2 \) by \( a_1 \cdot a_2 \)

   - Replace \( \neg x_1 \) by \( (1 - a_1) \)

   - Replace \( x_1 \lor x_2 \) by \( a_1 + a_2 - a_1 \cdot a_2 \)

   Note that \( x_i \)'s \( \in \{0, 1\} \) whereas \( a_i \)'s \( \in F \). Then, it is easy to prove that following claim.

   **Claim:** If \( \phi \) is satisfiable, then \( \exists l_1, \ldots, l_n \in \{0, 1\}^n \) s.t. \( f_\phi(l_1, \ldots, l_n) = 1 \). (otherwise \( \forall l_1, \ldots, l_n \in \{0, 1\}^n, f_\phi(l_1, \ldots, l_n) = 0 \).)

   Moreover, \( deg(f_\phi) \leq 3m \) with respect to any variable of \( f_\phi \). \( (\phi \) has \( m \) clauses), and \( |f_\phi| = O(m + n) \). Letting
A_\phi = \sum_{a \in \{0,1\}^n} f_\phi(a_1, \ldots, a_n)

It is clear that $A_\phi$ is the total number of satisfying assignments of $\phi$. All the above claims for $f_\phi, A_\phi$ hold over a finite integer field $F_p$ when $p > 2^n$ and $p$ is prime. As a last piece of notation, let

$$q_i(r_1, \ldots, r_{i-1}, y_i) = \sum_{a_{i+1} \in \{0,1\}, \ldots, a_n \in \{0,1\}} f_\phi(r_1, \ldots, r_{i-1}, y_i, a_{i+1}, \ldots, a_n)$$ (1)

for $r_1, \ldots, r_{i-1} \in F_p^*$. Then $q_1(0) + q_1(1) = A_\phi$. We now give the IP protocol in which $P$ proves to $V$ that $\phi$ has $k$ satisfying assignments.

1. $P$ sends to $V$ prime $p$ with a short certificate of primality (since $PRIMES \in NP$). $V$ verifies that $p$ is in fact a prime and $p > \max(2^n, 9nm)$ (otherwise rejects and halts).

2. (Round 1). $P$ sends to $V$ the polynomial $q_1(y_1)$. Since each $q_i$ is a polynomial of a single variable, $\deg(q_i) \leq 3m$. Thus $P$ can send $q_i$ to $V$ by sending its coefficients.

3. $V$ verifies $q_1(0) + q_1(1) = k$ (otherwise rejects and halts). Now $V$ needs to be convinced that $q_1$ is in fact the same polynomial as (1). In order to achieve this, $V$ sends $r_1 \in_R F_p^*$ to $P$.

4. $P$ sends to $V$ the polynomial $q_2(r_1, y_2)$.

5. $V$ checks that $q_2(r_1, 0) + q_2(r_1, 1) = q_1(r_1)$ (otherwise rejects and halts). Now $V$ needs to be convinced that $q_2$ is in fact the same polynomial as (1).

6. · · ·

7. (Round $i$). $P$ sends to $V$ the polynomial $q_i(r_1, \ldots, r_{i-1}, y_i)$.

8. $V$ checks that $q_i(r_1, \ldots, r_{i-1}, 0) + q_i(r_1, \ldots, r_{i-1}, 1) = q_{i-1}(r_1, \ldots, r_{i-1})$. Now $V$ wants to convinced that $q_i$ is infact the same polynomial as (1). $V$ sends $r_i \in_R F_p^*$ to $P$. 

5
10. (Round $n$). $P$ sends to $V$ the polynomial $q_n(r_1, \ldots, r_{n-1}, y_n)$.

11. $V$ checks that $q_n(r_1, \ldots, r_{n-1}, 0) + q_n(r_1, \ldots, r_{n-1}, 1) = q_{n-1}(r_1, \ldots, r_{n-1})$ (otherwise rejects and halts). Also now $V$ can easily check that $q_n$ is in fact the same polynomial as $f_\phi$ (by evaluating $q_n$ and $f_\phi$ on $3m + 1$ distinct points and accepting iff the two polynomials agree on each point).

**Claim:** The protocol above is an IP protocol for $\#SAT$.

**Proof:** Consider the case when $(\phi, k) \in \#SAT$. Since there is a prover $P$ which always sends the “right” things (i.e., prime $p$ and polynomials $q_i$ as in (1), the $V$ accepts $(\phi, k)$ with probability 1.

Now let $(\phi, k) \notin \#SAT$. We claim that it suffices to look at a $P$ who tries to maximize the probability that $V$ accepts. Then this $P$ will send the “wrong” polynomial $q_1$. Since $V$ chooses $r_1$ randomly in $F_p^*$, by choosing $p$ appropriately,

$$\Pr(V \text{ makes mistake in round } 1) \leq \frac{\deg(q_1)}{|F_p^*|} = \frac{3m}{|F_p^*|}$$

The same number also upper bounds the probability of making a mistake in any round $i$ (since $r_i$ is chosen independently for each round). Then the probability that $V$ makes a mistake during the protocol is upper bounded by the sum of the probability of making a mistake in any round, i.e.

$$\Pr(V \text{ makes a mistake}) \leq \sum_{i=1}^{n} \frac{3m}{|F_p^*|} = \frac{3mn}{|F_p^*|}$$

Thus, by choosing a prime $p$ such that $p > \max(2^n, 9mn)$ we get

$$\Pr(V \text{ accepts } (\phi, k) \notin \#SAT) \leq \frac{1}{3}$$

This concludes the proof that $\#SAT \in IP$.

This technique can be generalized to show that $PSPACE \subseteq IP$. In the next class, we will look at multi-prover systems (prover-verifier systems with multiple provers) and study the power of multi-prover systems. We will also see that with multi-prover systems, the size of the proofs is much shorter (compared to the $IP$ protocol presented above) i.e. proofs have length at most poly-logarithmic in the size of input.
References
