CS 4501-6501 Topics in Cryptography February 16, 2015

Lecture 10

Lecturer: Mohammad Mahmoody *Scribe: Tahseen Rabbani*

1 Delegation of Computation

We will be considering the scenario in which a weak entity (delegator) wishes to delegate a costly computation to a strong entity (worker). We will consider the case in which the weak entity wants to delegate the computation of $f(\text{x})$ for some polynomial-time computable function $f(\cdot)$.

1.1 Delegation Protocol

It is possible that the delegator may not trust the worker and requires that the worker prove the computation is correct. In this perspective, the delegator is a deterministic "verifier" and the worker is the "prover." The prover provides a solution y of $f(\text{x})$ and a proof π to convince the verifier of the truth of the solution, denoted x, belongs to some fixed language L. The verifier may either choose to accept or reject (x, y, π), respectively denoted $\text{Ver}(x, y, \pi) = 0 \vee 1$. By assumption, the verifier should have a running time significantly smaller than the time complexity of f, and the prover's running time should be in polynomial time in the time complexity of f.

We would like the verification process to be *sound* and *perfectly complete.* Perfect completeness means that if the worker provides (x, y, π) such that $y = f(\text{x})$, then $\Pr[\text{Ver}(x, y, \pi) = 1] = 1$. Soundness means that if the prover provides (x, y, π) such that $y \neq f(\text{x})$, then $\Pr[\text{Ver}(x, y, \pi) = 0]$ is "very high," which we will make more concrete in the next section.

1.2 $\mathcal{MA}$

We say that a language L is of class $\mathcal{MA}$ (for Merlin-Arthur) if there exists a polynomial-time deterministic Turing machine M and polynomials poly_1 and poly_2 such that for every input string of length $|x|$,

- if $x \in L$, then $\exists |w| \leq \text{poly}_1(|x|)$ such that $\forall y \leq \text{poly}_2(|x|)$, $\Pr[M(x, y, w) = 1] = 1$.
- if $x \notin L$ then $\forall |w| \leq \text{poly}_1(|x|)$, $\forall y \leq \text{poly}_2(|x|)$, $\Pr[M(x, y, w) = 0] = 1 - \mu(|x|)$,

where the negligible function $\mu(|x|)$ is one that is smaller than $1/\text{poly}(|x|)$. By letting $M = \text{Ver}$, it is easy to see how these conditions satisfy the properties of our previously described perfectly complete delegation protocol.

$\mathcal{MA}$ can be thought of as the randomized $\mathcal{NP}$ class, with the randomness referring to the strings $|y| < \text{poly}_2(|x|)$.

2 Interactive proofs

In this section, we will consider how our delegation protocol changes if interaction is allowed in the proof verification. First a reminder: $\mathcal{NP}$-completeness of SAT (or any other $\mathcal{NP}$-complete decision problem) means that there is an efficient mapping $\varphi_L(\cdot)$ such that $x \in L \Leftrightarrow \varphi_L(x) \in \text{SAT}$.

2.1 $\mathcal{IP}$ and Graph Non-Isomorphisms

For a pair of interactive algorithms (P, V), we denote their interaction over a common input x by $P \leftrightarrow V$, upon which V chooses to accept or reject x (which is the truth of some statement in some fixed language L).

We say that $L \in \mathcal{IP}$ if there exists a pair (P, V), where P is computationally unbounded, and V is running in probabilistic poly($|x|$), such that

- if $x \in L$ then $\Pr[V \leftrightarrow P$ accepts $x]$=1.
- if $x \notin L$, then for any P^* (even cheating), we have that $\Pr[V \leftrightarrow P^*$ rejects $x] = 1 - \mu(|x|)$.

In the context of an abstract delegation scheme, P would be a prover, and V would be the verifier.

An interesting example of a problem in $\mathcal{IP}$ is the Graph Non-Isomorphism problem. We say that a pair of graphs $(G_1, G_2) \in L$ if $G_1 \ncong G_2$. Of course, the only interesting case is when G_1 and G_2 have the same number of vertices. The interactive proof system is as follows:

1. Verifier randomly chooses $H \in \{G_1, G_2\}$. A permutation τ is applied to yield the graph $\tau H = \bar{H}$. This corresponds to a permuation of rows/columns in the adjacency matrix of H.
2. Verifier sends $\bar{H}$ to the prover.
3. WLOG, the prover replies with 0 if $\bar{H} \cong G_1$ and replies with 1 if $\bar{H} \cong G_2$.
4. Verifier accepts the reply if and only if the prover replies with the bit corresponding to the original selection (graph).

If $G_1 \ncong G_2$, the prover will always answer correctly. If $G_1 \cong G_2$, that is $(G_1, G_2) \in L$, then the prover (even when cheating) will fail to guess correctly with probability 1/2, since G_1 (G_2) is distributed with a copy of itself. Is it important to note that we can decrease the error from 1/2 to $1/2^k$ by k repetitions of the protocol.

2.2 PSPACE and the PCP Theorem

PSPACE is the set of all decision problems which can be solved by Turing machine using a polynomial-bit algorithm, i.e., problems which are computable with "efficient" memory.

Theorem 1. *$\mathcal{IP}$ = PSPACE*

This result is due to Adi Shamir [Sha92].

SAT, in fact, in PSPACE. If φ has n variables, our goal is to find out if there exists an assignment $x \in \{0, 1\}^n$ such that $\varphi(\mathrm{x}) = 1$. To try all possibilities, we don't need exponentially many bits; only n bits are needed. Furthemore, the SAT is trivially $\mathcal{IP}$ since it is $\mathcal{NP}$.

In our definition of $\mathcal{IP}$, it is possible for the prover to be running in large time complexity. Returning to our original delegation protocol, we require the prover to be running in polynomial time, so how can we use Shamir's theorem to aid us in this endeavor? Hopefully, problems which are computable with logarithmically many bits can be proven in polynomial time. This class of problems would not be terribly large, but it would be nontrivial, and we could employ our original delegation protocol over this class.

The first step towards achieving our delegation protocol is "short proofs" of a given statement. Short proofs are not always possible, but the following theorem shows us that we may be able to convince a verifier with only a small number of bits of the original proof.

Theorem 2 (PCP Theorem). *Every language $L \in \mathcal{NP}$ has a probabilistic verifier $\overline{Ver}$ such that*

- *if $x \in L$, there exists a convincing proof π which $\overline{Ver}$ accepts with probability 1.*
- *if $x \notin L$, $\overline{Ver}$ rejects every proof with probability at least 1/2.*

Furthermore, $\overline{Ver}$ examines only a constant number of bits k in π.

This result is given and proven in [**?**]. Johan Håstad gave the following improvement in [**?**].

Theorem 3. *Every language $L \in \mathcal{NP}$ has a probabilistic verifier $\overline{Ver}$ such that*

- *if $x \in L$, there exists a convincing proof π which $\overline{Ver}$ accepts with probability $\geq \frac{999}{1000}$.*
- *if $x \notin L$, $\overline{Ver}$ rejects every proof with probability at least 1/2.*

Furthermore, $\overline{Ver}$ examines only 3 bits in π.

References

[Sha92] A. Shamir. IP = PSPACE. *J. ACM*, 39:869–877, 1992.