6.876/18.426: Advanced Cryptography May 12, 2003

Lecture 22: Multi-Party Computation with Perfect Channels

Scribed by: Chun-Yun Hsiao
Guest Lecturer: Jonathan Herzog

1 Recap

In the previous two lectures, we've seen how to "securely" compute any (probabilistic) function. The general strategy is to first design a protocol secure in the *honest-but-curious* model,[1] and then force malicious players to follow the prescribed instructions by "compiling" the protocol into a robust one. The communication network we considered was a general one, i.e., any message sent was assumed to be known by the adversaries. The privacy relies on public-key encryption, thus it's *computationally* secure.

2 Perfect Channels

Today we consider a network where every pair of players has a private communication channel that cannot be tapped. Whether this model exists or not is more like a philosophical belief, but the protocol presented today is very clean and gives more intuition rather than technical complications. Furthermore, it's *information-theoretically* secure.

2.1 The Honest-But-Curious Protocol

As before, let n be the number of players, among which at most t may be corrupted. Here we only consider the static case, i.e., good (bad) guys remain good (bad) throughout the protocol. These t players may cooperate in any way they like as long as they are running the program specified by the protocol. For example, they must use a truly random string when they are told to. But they may, at any time, share with the other bad guys their private knowledge, including inputs, messages received, and random tape, etc. Of course, we assume that the bad guys know each other but the good guys don't.

The basic idea is simliar to what we saw in the last two lectures:

1. Split the inputs into shares; player P_i has (secret) share s_i.
2. Maintain invariant across gates (**AND** and **NOT**).
3. Reveal shares at the end.

The original paper [BGW88] deals with functions (to be computed) in a finite field. For simplicity, we focus on Boolean functions. We shall see shortly, however, we do require to work in a finite field of size greater than the number of players n.

[1] The definition will be treated later in class if time permits, otherwise is left as an exercise.

How do we share a bit in a way such that if t (or less) players cooperate, they get absolutely nothing, but if more than t players get together, they can derive the original bit? A degree t polynomial! We have seen this useful secret sharing scheme [Sha79] last time when we want to prevent malicious players from quitting before the protocol is completed.[2]

So player P_i, holding private input X_i, generates a random polynomial f of degree t with constant term equal to X_i. Send $f(\alpha_j)$ to P_j, for $1 \leq j \leq n$. Note that the α's are distinct constants and known to every player. This is where we require the size of the field to be greater than n.

It is not hard to see that if t (or less) players share the $f(\alpha_j)$'s, the constant term $f(0)$ is undetermined; it is equally likely to be 0 or 1, because the polynomial was chosen at random. And with more than t many $f(\alpha_j)$'s, f and thus $f(0)$ is determined.

The **invariant** we want to maintain is: *everyone has one point of a random polynomial f of degree t and $f(0)$ is the value of the wire* (consider the function as a Boolean circuit).

2.1.1 Invariant Across NOT Gate

To keep the invariant across **NOT** gate, we simply subtract f from 1: $f' \triangleq 1 - f$. Thus $f'(0) = 1 - f(0)$ as desired. So whenever there is a **NOT** gate, everyone subtract his share from 1: $f'(\alpha_i) \triangleq 1 - f(\alpha_i)$. Note that there is no communication needed.

Remark: The **NOT** gate is analogous to linear operators "+" and "× a constant" in [BGW88]; the **AND** gate is analogous to the binary operator "×."

2.1.2 Invariant Across AND Gate

The **AND** gate is a little more complicated. Naturally, we let $h \triangleq f \times g$, where f and g are two random polynomials of degree t (of course with constant term equal to a secret bit). There are two problems to overcome:

- h is not random.
- h is of degree $2t$; may exceed the size of the field later.

Remark: We can see that maintaining the invariant across **AND** gate is a *complete* problem for multi-party computation with perfect channels. This methodology is very common in computer science, as we all have seen it for the class NP. Another example is 1-out-of-4 OT from last lecture. The rule of thumb for solving a new problem is, to consider the most general case and see if we can find a well-defined complete problem. Then either solve it, which fortunately is the case for multi-party computation, or "claim" it as unsolvable in general, e.g. the NP-complete ones.

Injecting Randomness The first problem can be fixed by adding other degree $2t$ random polynomials: every player P_i chooses a degree $2t$ random polynomial q_i, where q_i has no constant term, and send $q_i(\alpha_j)$ to P_j. The new share for P_j becomes $f(\alpha_j) \times g(\alpha_j) + \sum_i q_i(\alpha_j)$. Now the players share a random polynomial of degree $2t$ with constant term

[2] In the model where no perfect channel is assumed.

equal to $f(0) \times g(0)$. Let $h'(x) = h'_0 + h'_1 x + \cdots + h'_{2t} x^{2t}$ be this polynomial, where h'_0 is the secret.

Reducing Degree To reduce the degree back to t, we simply drop all terms with exponent greater than t. Thus $k(x) \triangleq h'_0 + h'_1 x + \cdots + h'_t x^t$ is the polynomial to be shared. Note that $k(x)$ is random and $k(0) = h'_0 = f(0) \times g(0)$ is indeed the secret, but $k(\alpha_i) \neq h'(\alpha_i)$! We need one more step to correct the shares $k(\alpha_i)$.

Our goal is to derive (securely) the new shares $R \triangleq (k(\alpha_1), k(\alpha_2), \ldots, k(\alpha_n))$ from the temporary shares $S \triangleq (h'(\alpha_1), h'(\alpha_2), \ldots, h'(\alpha_n))$. It turns out that there exists a *constant* matrix A such that $R = S \cdot A$. By constant we mean that A does not depend on the inputs; it only depends on the α's, thus is publicly known in advance. Since matrix multiplication involves only linear operations, the scheme in the **NOT**-gate case can do this. More precicely, each player P_i shares his share by generating a random degree t polynomial $f^{(i)}$ with constant term equal to $h'(\alpha_i)$. So $f^{(\ell)}(\alpha_i)$ is the share sent from P_ℓ to P_i. Then for each i, j, player P_i sends $\sum_\ell A_{\ell j} \cdot f^{(\ell)}(\alpha_i)$ to player P_j. It is not hard to see that P_j gets a polynomial evaluated on n points. The constant term of this polynomial is his share $k(\alpha_j)$, which can be derived by interpolation. It remains to specify the matrix A.

Let H and K be n-dimensional row vectors: $H \triangleq (h'_0, h'_1, \ldots, h'_t, \ldots, h'_{2t}, 0, \ldots, 0)$ and $K \triangleq (h'_0, h'_1, \ldots, h'_t, 0, \ldots, 0)$. And let B and P be $n \times n$ matrices:

$$B \triangleq \begin{pmatrix} 1 & 1 & \cdots & 1 \\ \alpha_1 & \alpha_2 & \cdots & \alpha_n \\ \alpha_1^2 & \alpha_2^2 & \cdots & \alpha_n^2 \\ \vdots & \vdots & \ddots & \vdots \\ \alpha_1^{n-1} & \alpha_2^{n-1} & \cdots & \alpha_n^{n-1} \end{pmatrix}, \quad P \triangleq \begin{pmatrix} I_{t+1} & 0 \\ 0 & 0 \end{pmatrix},$$

where I_{t+1} is the $(t+1)$-dimensional identity matrix. Note that B is the Vandermonde matrix and in particular is invertible. P is the projection matrix. We have

$$\begin{aligned} S &= HB, \\ K &= HP, \\ R &= KB. \end{aligned}$$

Since B is invertible, we have

$$\begin{aligned} R &= KB \\ &= (HP)B \\ &= H(PB) \\ &= (SB^{-1})(PB) \\ &= S(B^{-1}PB). \end{aligned}$$

Let A be $B^{-1}PB$ and we are done.

3 Compiling the Protocol

The real beautiful part of the protocol with perfect channel, is the one that is secure against malicious players. Unfortunately, we have to see the compilation next time.

References

[BGW88] M. Ben-Or, S. Goldwasser, A. Wigderson. Completeness Theorems for Non-Cryptographic Fault-Tolerant Distributed Computation. Proceedings of 20th ACM Symposium on the Theory of Computing (STOC), pp. 1-10, 1988.

[GMW87] O. Goldreich, S. Micali, and A. Wigderson. How to Play Any Mental Game, or, A Completeness Theorem for Protocols with Honest Majority. Proceedings of 19th ACM Symposium on the Theory of Computing (STOC), pp. 218-229, 1987.

[Sha79] A. Shamir. How to Share a Secret. Communications of the ACM, pp. 612-613, 22(11), November 1979.