CMSC 858K — Introduction to Secure Computation November 11, 2013

Lecture 28

Lecturer: Jonathan Katz *Scribe(s): Xiong Fan*

1 Relaxations of Security

In standard definition of security, we assume that malicious parties can handle arbitrarily adversarial behaviors, even behavior that would be unlikely in the real world. However, in this lecture, we consider relaxation of security that we assume a attacker motivated by some rational behavior, and our system only defend against such attacks.

2 Rational Secret Sharing Scheme

Rational cryptography is a combination of cryptography and game theory. Here we first define the utility of parties informally (s is the secret to be shared):

Utility 3 : Most prefer exclusivity, i.e., they learn s, but other parties do not.

Utility 2 : Otherwise, prefer learning s rather than not.

Utility 1 : Neither party learns.

More formally, the process of the protocol should be:

1. s is chosen uniformly from domain D. We say that a party learns s if it outputs s at the end of the protocol.
2. Design some protocol π for reconstructing the secret s.
3. Running the protocol to completion should be a computational Nash equilibrium, i.e., if P_2 runs π honestly, we have expected utility of P_i^*, $P_i^* \leq 2 + negl(\cdot)$, for all PPT P_i^*.

The framework of protocol π in each iteration i is:

1. Parties compute a functionality $F_{seshare}((s_1, \sigma_1), (s_2, \sigma_2))$ (σ is the signature of the previous message) to obtain $((s_1', \sigma_1'), (s_2', \sigma_2'))$.
2. Simultaneously exchange (s_1', σ_1') and (s_2', σ_2').
3. Each party verifies the signature and computes $s^* = s_1' \oplus s_2'$.
4. If $s^* \in D$, output s^*.
5. If any cheating detected, then abort.
6. Otherwise, continue to the next iteration.

The description of the functionality $F_{seshare}((s_1, \sigma_1), (s_2, \sigma_2))$ is:

1. if signatures do not verify, output $\perp$.
2. With probability δ, choose uniform s'_1, s'_2, such that $s'_1 \oplus s'_2 = s_1 \oplus s_2$.
3. With probability $1 - \delta$, choose uniform s'_1, s'_2 such that $s'_1 \oplus s'_2 = 1$
4. Give (s'_1, σ'_1) to P_1, (s'_2, σ'_2) to P_2.

Claim 1 *For appropriate choice of δ_1, this protocol is computational Nash equilibrium.*

Proof Consider possible deviating behavior by P_1. If P_1 aborts during exchange,

- With probability δ, P_1 learns s and P_2 does not, which means P_1 gets utility 3.
- With probability $1 - \delta$, P_1 never learn s from the protocol, which means P_1 gets expected utility $\frac{1}{|D|} \cdot 3 + (1 - \frac{1}{|D|})$

■

Here is another protocol π' from Fuchsbauer et al.'09:

Pre-Processing Stage : Compute some functionality $F^*_{seshare}((s_1, \sigma_1), (s_2, \sigma_2))$ to obtain a sequence:

$$((s_{1,1}, \sigma_{1,1}), (s_{1,2}, \sigma_{1,2}), ..., (s_{1,n}, \sigma_{1,n}))$$

$$((s_{2,1}, \sigma_{2,1}), (s_{2,2}, \sigma_{2,2}), ..., (s_{2,n}, \sigma_{2,n}))$$

Iteration i : Simultaneously exchange $(s_{1,i}, \sigma_{1,i})$ and $(s_{2,i}, \sigma_{2,i})$.

Modify $F^*_{seshare}$:

1. Choose i according to a geometric distribution with parameter δ.
2. For all $j < i$, the shares will XOR to a uniform value in D, plus a 0 bit.
3. For $j = i$, the shares will XOR to s, plus a 0 bit.
4. for all $j > i$, the shares will XOR to s, plus a 1 bit.