Fundamentals of Cryptography Homework 9

周书予

2000013060@stu.pku.edu.cn

December 24, 2022

Problem 1 1-out-of-t Oblivious Transfer

Part A

The protocol goes like this:

- The sender prepares t random messages $r_1, \cdots, r_t$ uniformly sampled from $\{0,1\}^\ell$.
- The given 1-out-of-2 OT protocol is used for t rounds. In round i, the sender prepares the following two inputs for the OT protocol:

$$(r_i, \bigoplus_{j<i} r_j \oplus m_i)$$

- If the receiver wants to learn m_x, he or she requires the former message in the first $x-1$ rounds, and the later message in round x, which means he or she can learn

$$r_1, \cdots, r_{x-1}, \bigoplus_{j<x} r_j \oplus m_x$$

  helping him or her reveal m_x.

The correctness for this protocol is obvious.

The view of the sender can be simulated by $t\mathsf{OT.Sim}_S$, which runs $\mathsf{OT.Sim}_S$ on t pairs of inputs prepared by the sender independently.

The view of the receiver can also be simulated by $t\mathsf{OT.Sim}_R$. In the first $x-1$ rounds, the (semi-honest) receiver can only require the former message, and $t\mathsf{OT.Sim}_R$ simply samples $r \leftarrow \$$ and returns $\mathsf{OT.Sim}_R(r, 0)$ to the receiver. Of course it should remember all r-s. When round x comes, the receiver requires the later message, the $t\mathsf{OT.Sim}_R$ will return $\mathsf{OT.Sim}_R(m_x \oplus R, 1)$, where R denotes the XOR sum of all r-s. In the remaining rounds, it simply returns $\mathsf{OT.Sim}_R(\$, b)$ for the require bit b from the receiver.

Thus, the protocol is secure against semi-honest sender and semi-honest receiver.

Problem 2 Oblivious Linear Function Evalution

Part B

Security of Receiver: computationally secure

With input a, b, the simulator for the sender $\mathsf{Sim}_S(a, b)$ should output view somehow indistinguishable from $\mathsf{View}_S(a, b, x)$.

According to the protocol, the simulator Sim_S should send to the sender pk the public key and c the encryption for x. Without knowing x, the simulator simply sends $c = \mathsf{Enc}(pk, 0)$, or just the encryption for some random element in $\mathcal{R}$, which is computationally indistinguishable from $\mathsf{Enc}(pk, x)$ as long as the public-key encryption scheme is CPA-secure.

So we have $\mathsf{Sim}_S(a, b) \approx_c \mathsf{View}_S(a, b, x)$, which suggests computational security of the receiver.

Security of Sender: perfectly secure

We need to build a simulator Sim_R such that $\mathsf{Sim}_R(x, ax + b) \approx \mathsf{View}_R(a, b, x)$.

Sim_R can encrypt $ax+b$ directly and send it back to the receiver, so this security is perfect.

Part C

With Alice having $x_1, y_1 \in \mathcal{R}$ and Bob having $x_2, y_2 \in \mathcal{R}$, the arithmetic analog of GMW protocol should give Alice $z_1 \in \mathcal{R}$ and Bob $z_2 \in \mathcal{R}$, such that

$$z_1 + z_2 = (x_1 + x_2)(y_1 + y_2)$$

In this new protocol, Alice should sample $r_1 \leftarrow \mathcal{R}$ and Bob should sample $r_2 \leftarrow \mathcal{R}$, both uniformly at random.

Alice should set the OLE parameters $a = x_1, b = r_1$. Bob should make query for $x = y_2$ and learn $ax + b = x_1y_2 + r_1$, without any other information. After that, Bob set $z_2 = (x_1y_2 + r_1) + (x_2y_2 - r_2)$.

Similarly Alice can also learn $x_2y_1 + r_2$ without any other information, and set $z_1 = (x_2y_1 + r_2) + (x_1y_1 - r_1)$.

It is not hard to see that $z_1 + z_2 = (x_1 + x_2)(y_1 + y_2)$, which means this protocol is correct.

Problem 3 Information-Theoretic Garbled Circuits

Part A

Notice that the gate table is indexed by $(x_{j_1} \oplus \alpha_{j_1}, x_{j_2} \oplus \alpha_{j_2})$ which hides x_{j_1}, x_{j_2}, so the table no longer needs to be random shuffled.

When evaluating this modified garbled circuit, use the first bit of two garbled input $L_{j_1,x_{j_1}}$ and $L_{j_2,x_{j_2}}$ to index the gate table, and decrypt only one table entry to reveal L_{i,x_i}.

Part B

The biggest issue when replacing the CPA-secure encryption scheme with a perfect secure encryption scheme directly is that some $L_{i,b}$-s are used as keys in the gate table more than once.

To fix this, we need to ensure that such $L_{i,b}$-s use different bits in encryption. Specifically, let's say a gate g with input wires j_1, j_2 and output wire i, then, if the length of $L_{i,0}, L_{i,1}$ is k-bit, we need $2k$ bits in $L_{j_1,0}, L_{j_1,1}, L_{j_2,0}, L_{j_2,1}$ **dedicated for wire** i.

We should now analyse the total length of all $L_{i,b}$-s. Generally speaking, the length of $L_{i,b}$ should equal to twice the sum of length of all $L_{j,b}$-s where j is a "descendant" of i, plus one, which stands for the mask bit.

We call a sequence of wires $i_1, \cdots, i_d$ a *path* of depth d in circuit C, if

- for all $1 \leqslant j < d$, there is a gate in C which takes i_j as input wire and i_{j+1} as output wire.

It can be proved by induction that, for any wire i in C, its garbled string $L_{i,0}, L_{i,1}$ is of length

$$\sum_d 2^{d-1}[\# \text{ path of depth } d \text{ strarting with wire } i]$$

For circuit with depth $O(\log n)$, the total length of $L_{i,b}$-s is $\text{poly}(n)$. They can be garbled efficiently.

Part C

First, Sim samples random strings as L_{i,x_i} for each $i \in [1, n]$.

Then, it generates gate table for each gate g. Let's say for a gate g with input wires j_1, j_2 and output wire i, since $L_{j_1,x_{j_1}}, L_{j_2,x_{j_2}}$ are known, Sim knows which entry in gate table encrypts L_{i,x_i}. For other entries, it should simulate since at least one key to encrypt is unknown to itself.

Specifically, to simulate $\mathsf{Enc}(k_1, \mathsf{Enc}(k_2, p))$, if k_1 is unknown, Sim should use \$ to simulate, while if k_1 is known but k_2 unknown, it should use $\mathsf{Enc}(k_1, \$)$ instead to simulate.

CPA-security or perfect security of the encryption scheme ensures the computational or perfect indistinguishability of the simulator.

Problem 4: Randomized Encoding

Part A

$\hat{f}$ takes x, r as input. It generates $\tilde{C}$ the garbled circuit of f, $L_{i,b}$ the garbled strings, using r as its random tape, and outputs $\tilde{C}$ together with $L_{1,x_1}, \cdots, L_{n,x_n}$.

For correctness, notice that garbled circuit can be simulated efficiently, thus efficient decoding algorithm which retrieves $f(x)$ can be easily built.

For security, it is guaranteed by the security of Yao's garbled circuit.

For less complexity (less circuit depth here, specifically), notice that all gate tables can be computed in parallel, so the circuit to compute garbled circuit is of $O(1)$ depth, since all $L_{j_1,b}, L_{j_2,b}, L_{i,b}$-s are known in advance.

Part B

Protocol: Each party randomly picks a $r_i \leftarrow \mathcal{R}$, and then uses Π_g protocol computing $g((x_1, r_1), \cdots, (x_n, r_n)) = \hat{f}((x_1, \cdots, x_n), r_1 + \cdots + r_n)$, and finally decode $\hat{f}((x_1, \cdots, x_n), r_1 + \cdots + r_n)$ to get $f(x_1, \cdots, x_n)$.

The correctness of the new protocol is obvious.

Since Π_g is computationally secure against semi-honest adversary corrupting up to t parties, the new protocol is also computationally secure against semi-honest adversary corrupting up to t parties.

Problem 5: Threshold Secret Sharing Lower Bound

Part A

Let x_0, x_1 denote two distinct elements from $\mathcal{X}$.

Let $S_{i,0}$ denote the i-th share when the secret is x_0. Notice that $S_{i,0}$ is a random variable over $\mathcal{S}_i$. Similarly we define $S_{i,1}$.

Since only one share can reveal no information about the secret, the distribution of $S_{i,0}$ and $S_{i,1}$ should be identical. Furthermore, we can get

$$\Pr\left[S_{i,0} = S_{i,1}\right] \geqslant \frac{1}{|\mathcal{S}_i|}$$

Also notice that it is impossible for both $S_{i,0} = S_{i,1}$ and $S_{j,0} = S_{j,1}$, since otherwise, in some circumstances it could be impossible to reveal the secret x (would feel confused between x_0 and x_1).

Thus,

$$\begin{aligned} 1 \geqslant \Pr\left[\exists i, S_{i,0} = S_{i,1}\right] &= \sum_{i=1}^{n} \Pr\left[S_{i,0} = S_{i,1}\right] \\ &\geqslant \sum_{i=1}^{n} \frac{1}{|\mathcal{S}_i|} \end{aligned}$$

So we have proved that $\sum_{i=1}^{n} \frac{1}{|\mathcal{S}_i|} \leqslant 1$. Function $f(x) = \log(1/x)$ is a convex function, by Jensen's Inequality,

$$\sum_i \log|\mathcal{S}_i| = \sum_i f(1/|\mathcal{S}_i|) \geqslant nf\left(\frac{\sum_i 1/|\mathcal{S}_i|}{n}\right) \geqslant nf\left(\frac{1}{n}\right) = n\log n$$

Part B

We convert a t-out-of-n secret sharing to a 2-out-of-n secret sharing, and try to follow the proof in **Part A**.

Let $A \subseteq [n]$ be a set of parties of size $t-2$. After receiving the share, all members in A boardcast its share to every other parties. It can be proved that the remaining $n-t+2$ parties are in a 2-out-of-n secret sharing with perfect correctness and perfect security.

Let $B = [n] \setminus A$. Following the proof in **Part A** we have

$$\sum_{i \in B} \log|\mathcal{S}_i| \geqslant (n-t+2)\log(n-t+2)$$

which means $\sum_i \log|\mathcal{S}_i| = \Omega((n-t)\log(n-t))$.