CS670: A2 Solutions

Prashant Kumar, 231110036

March 29, 2024

Solution-1: Secure Multiparty Computation Using Cards

Protocol Setup:

- Five cards are used: three aces and two kings.
- Alice and Bob each have a King and an Ace, and one extra Ace card is commonly used by both Alice and Bob.

Protocol Steps:

1. If Alice likes Bob, she places her cards face down in the order: King, Ace.
2. If Alice does not like Bob, she arranges her cards as: Ace, King.
3. If Bob does not like Alice, he arranges his cards as: King, Ace.
4. If Bob likes Alice, he arranges his cards as: Ace, King.
5. An Ace is then placed face down between their cards, creating a sequence: Alice's cards, middle Ace, and Bob's cards.
6. The cards undergo a cyclic shuffle, with some cards from the bottom moved to the top while maintaining order. This process repeats until both parties are satisfied. (this can be done like once Alice does the shuffle privately and then Bob does the shuffle privately so both Alice and Bob are unaware of each other's shuffle)
7. Finally, the cards arrangement after cycle shuffle is shown, and the interpretation of result is as follows.

Interpretation:

- If two Kings appear together during the cyclic arrangement, it indicates mutual interest between Alice and Bob, and they should go on the date.
- If two Kings do not appear together, it suggests that one or both parties are not interested (Here all other cases are having equal likelihood)

Example Scenario (Both likes each other):

1. Alice likes Bob, so she arranges her cards as: King, Ace.
2. Bob likes Alice, so he arranges his cards as: Ace, King.
3. With an Ace placed between their cards, the sequence becomes: King Ace Ace Ace King (all face down).
4. Despite shuffling, the order remains unchanged because moving a group of cards causes their positions to shift relative to the cards remaining at the top while original card and neighbors within the group cycle are together.
5. Finite Possibilities: Each card within the moving group eventually progresses through all possible positions relative to the others.
6. Hence, two Kings will always come close together, indicating mutual interest.

Alternate Scenarios:

- If at least one party does not like the other, an Ace will be placed between the Kings, leading to different arrangements such as:
  - King Ace Ace King Ace (Alice likes Bob not)
  - Ace King Ace Ace King (Alice does not like Bob)
  - Ace King Ace King Ace (Both don't like each other)
- In such cases, shuffling prevents two Kings from being neighbors, signaling a lack of mutual interest.

Solution-2: Secure Multiparty Computation on Secret Shares

I will refer Du-Atallah Protocol in my solution (treating it as a function) so I'm including Du-Atallah Protocol here-
Du-Atallah Protocol: First recall the Du-Atallah protocol for multiplication. P0 holds $(x0, y0)$ and P1 holds $(x1, y1)$. They compute $z0$ and $z1$ respectively such that $z0 + z1 = (x0 + x1) \cdot (y0 + y1)$. The protocol works as follows:

1. The protocol begins with a (trusted) dealer $P2$ sampling five random values $(X0, X1, Y0, Y1, \alpha)$.
2. $P2$ sends $(X0, Y0, X0 \cdot Y1 + \alpha)$ to $P0$ and $P2$ sends $(X1, Y1, X1 \cdot Y0 - \alpha)$ to $P1$.
3. $P0$ sends $(x0 + X0)$ and $(y0 + Y0)$ to $P1$. Similarly, $P1$ sends $(x1 + X1)$ and $(y1 + Y1)$ to $P0$.
4. Next, $P0$ computes, $z0 \leftarrow x0 \cdot (y0 + (y1 + Y1)) - Y0 \cdot (x1 + X1) + (X0 \cdot Y1 + \alpha)$.
5. Next, $P1$ computes, $z1 \leftarrow x1 \cdot (y1 + (y0 + Y0)) - Y1 \cdot (x0 + X0) + (X1 \cdot Y0 - \alpha)$.
6. Now observe, $z0 + z1 = (x0 + x1) \cdot (y0 + y1)$.

Part 1: Communication Time Optimization
Total Communication Time in optimized scenario:
Assumption:
In optimized scenario both party can send the message simultaneously if needed (assuming dual-mode communication for optimized implementation) and a party can receive from multiple senders (as real world receiving devices have receiver buffer to implement this)

Given in question that the time to send a message from P_i to P_j is $x_{i,j}$.
Communication steps and corresponding time:

- Assuming P2 has already sampled random values and computed respective values to start the protocol.
- $P2$ sends $(X0, Y0, X0 \cdot Y1 + \alpha)$ to $P0$ and time for that will be $3 * x_{2,0}$.
- Now $P2$ computes the respective values for sending to $P1$ meanwhile notice that $P1$ has the required values so it can compute $(x0 + X0)$ and $(y0 + Y0)$ (negligible time) and start sending these to $P1$.
- $P2$ sends $(X1, Y1, X1 \cdot Y0 - \alpha)$ to $P1$ and simultaneously $P0$ sends $(x0 + X0)$ and $(y0 + Y0)$ to $P1$ time for this step will be $max(3 * x_{2,1}, 2 * x_{0,1})$ where max denotes the maximum value of both arguments.
- Now when $P1$ has received the required values from $P2$ it can also start computing values to be send to $P0$.
- $P1$ sends $(x1 + X1)$ and $(y1 + Y1)$ to $P0$ and time for this will be $2 * x_{1,0}$.
- Now the communication part is done, and if they need to transfer more message the process can start again by $P2$, meanwhile results will be calculated to respective parties.

$$\text{Total Time} = 3 * x_{2,0} + max(3 * x_{2,1}, 2 * x_{0,1}) + 2 * x_{1,0}$$

Part 2: MPC Protocol for Dot Product with Vector Inputs Assumptions:

- Assume n dimension Vectors: $\vec{x} = (x_0, x_1, ..., x_{n-1})$, $\vec{y} = (y_0, y_1, ..., y_{n-1})$.
- Shares: Party Pi holds additive shares $x_{i,j}$ and $y_{i,j}$ for each element of vector $\vec{x}$ and $\vec{y}$. (for e.g. shares $x_{0,5}$ and $y_{0,5}$ are the shares of x_5 and y_5 respectively held by $P0$)
- Trusted Dealer: P2
- Assuming I can do the MPC for for Du-Atallah Protocol such that if 2 parties have shares $(x0, y0)$ and $(x1, y1)$ then at the end of protocol it results that respective parties have shares $z0$ and $z1$ such that $z0 + z1 = (x0 + y0) \cdot (y0 + y1)$. (protocol for this is included above)

Objective: At the end of the protocol party $P0$ and $P1$ should have shares $z0$ and $z1$ such that $z0 + z1 = (x0 \cdot y0 + x1 \cdot y1 ... xn - 1 \cdot yn - 1)$.
Protocol Steps:

- For i=0 to n-1
  1. Perform Du-Atallah multiplication on the shares $x_{0,i}$ and $y_{0,i}$ held by party $P0$ and shares $x_{1,i}$ and $y_{1,i}$ held by party $P1$. (here trusted dealer presence is assumed inherently)
  2. At the end of Du-Atallah multiplication party $P0$ has $Z_{0,i}$ and party $P1$ has $Z_{1,i}$ such that $Z_{0,i} + Z_{1,i} = (x_{0,i} + x_{1,i}) \cdot (y_{0,i} + y_{1,i}) = (xi \cdot yi)$
- P0 computes its share of the dot product as: $z_0 = \sum_{i=0}^{n-1} z_{i,0}$
- P1 computes its share of the dot product as: $z_1 = \sum_{i=0}^{n-1} z_{i,1}$
- Now observe that $z0 + z1 = (x0 \cdot y0 + x1 \cdot y1 ... xn - 1 \cdot yn - 1)$, which is result we wanted.

Part 3: Implementing an XORif Functionality
Trusted Dealer (P2):

- Sample random values: $X0$, $X1$, $Y0$, $Y1$, α
- $P2$ to $P0$: Send $(X0, Y0, X0 \text{ AND } Y1 \oplus \alpha)$ to $P0$
- $P2$ to $P1$: Send $(X1, Y1, X1 \text{ AND } Y0 \oplus -\alpha)$ to $P1$
- $P0$ to $P1$: Send $(x0 \oplus X0, y0 \oplus Y0)$ to $P1$
- $P1$ to $P0$: Send $(x1 \oplus X1, y1 \oplus Y1)$ to $P0$

Computations:

- **P0 Computes:**

$$z0 \leftarrow x0 \text{ AND } (y0 \oplus (y1 \oplus Y1)) \oplus Y0 \text{ AND } (x1 \oplus X1) \oplus (X0 \text{ AND } Y1 \oplus \alpha)$$

- **P1 Computes:**

$$z1 \leftarrow x1 \text{ AND } (y1 \oplus (y0 \oplus Y0)) \oplus Y1 \text{ AND } (x0 \oplus X0) \oplus (X1 \text{ AND } Y0 \oplus -\alpha)$$

Result:

- $z0 \oplus z1 = (x0 \oplus x1)\text{AND}(y0 \oplus y1)$

Goal: Show that $z0 \oplus z1 = (x0 \oplus x1)(y0 \oplus y1)$ **Explanation:**

- **P0's Calculation:**

$$z0 = x0(y0 \oplus y1 \oplus Y1) \oplus Y0(x1 \oplus X1) \oplus (X0Y1 \oplus \alpha)$$

- **P1's Calculation:**

$$z1 = x1(y1 \oplus y0 \oplus Y0) \oplus Y1(x0 \oplus X0) \oplus (X1Y0 \oplus -\alpha)$$

- **Computing $z0 \oplus z1$:**

$$\begin{aligned} z0 \oplus z1 &= (\text{Calculation for } z0) \oplus (\text{Calculation for } z1) \\ &= (x0y0 \oplus x0y1 \oplus x0Y1 \oplus x1Y0 \oplus x1X1 \oplus X0Y1 \oplus \alpha) \\ &\quad \oplus (x1y1 \oplus x1y0 \oplus x1Y0 \oplus x0Y1 \oplus x0X0 \oplus X1Y0 \oplus -\alpha) \\ &= x0y0 \oplus x0y1 \oplus x1y0 \oplus x1y1 \oplus x0X0 \oplus x1X1 \oplus 2\alpha \\ &= (x0 \oplus x1)(y0 \oplus y1) \oplus x0X0 \oplus x1X1 \oplus 2\alpha \\ &= (x0 \oplus x1)(y0 \oplus y1) \quad (\text{Since } xiXi \oplus xiXi \oplus 2\alpha = 0) \end{aligned}$$

finally:

- First, we compute $(b0 \oplus b1)(y0 \oplus y1)$ and store it into A.
- Then, using A (divided into shares and computing $z0$ and $z1$), we compute $(x0 \oplus x1) \cdot A$.
- If $b0 \oplus b1 = 1$, then $A = y0 \oplus y1$; otherwise, $A = 0$.
- So the final answer would be:
  - $z0 \oplus z1 = x0 \oplus x1 \oplus y0 \oplus y1$ for $b0 \oplus b1 = 1$
  - Otherwise, $z0 \oplus z1 = x0 \oplus x1$

Solution-3: Optimizations in Garbled Circuits

Part 1:

the gates that can be optimized to use only two cipher texts instead of the standard four are: **Note:**
1. I'm not including the NOT gate in the answer because it is trivial and there will be no logical reasons behind this as ask in the question. similarly question states for exactly 2 cipher texts but XOR gate requires 0 cipher text so please ignore XOR gate if that is the case.

XOR Gates: because of the FreeXOR optimization, XOR gates don't require a garbled table at all. Since the output is easily calculated from the input wire values and their difference, no cipher texts are needed for transmission.

AND Gates: Techniques like half-gates can reduce the needed cipher texts. By splitting the AND gate and ensuring one input is always known (either by the generator or the evaluator), only a single 'active' cipher text is needed. This cipher text determines the output, allowing for a two-cipher text representation.

Part 2:

One-Time Pads (OTPs) aren't a good fit because they are designed for a single use with a unique key. In garbled circuits, the same wire label (which acts as the key) has to encrypt multiple values within a gate. This repeated use compromises the security OTPs are meant to provide.

Point-and-permute helps because it limits how many times the same key is reused (because we only share the target cipher text in this) and hence OTP is used once also evaluator can decrypt only shared cipher text using OTP.