ECE 443/518 – Computer Cyber Security
Lecture 18 Secure Multi-Party Computation

Professor Jia Wang
Department of Electrical and Computer Engineering
Illinois Institute of Technology

October 31, 2022

Outline

Reading Assignment

- This lecture: Secure Multi-Party Computation
- Next lecture: ICS 2-7,14

Outline

Secure Multi-Party Computation

Garbled Circuit

Secure Multi-Party Computation

- Assume there are n honest-but-curious parties $1, 2, \ldots, n$.
- Each party k possesses a secret value v_k.
- Together they compute $f = F(v_1, v_2, \ldots, v_n)$.
  - For a well-known function F.
- Confidentiality: secret remains secret.
  - Any party k should only learn f from the computation, but nothing more about secrets of other parties.
- Ignore integrity issues.

Examples

- Voting
  - Secret from every party: 0 or 1
  - F computes the summation.
  - Every party learns only f, the number of 1's.
  - If there is only two parties, it is OK for each of them to actually learn other's secret since that can be derived from f and their own secret.
- Salary comparison
  - Secret from every party: a number representing salary.
  - F computes the maximum.
  - Every party learns only f, the highest salary.
  - But you can't learn other's salary even if there is only two parties.
- Let's examine the case between two parties Alice and Bob.
  - How could you represent arbitrary computations?

Outline

Secure Multi-Party Computation

Garbled Circuit

Circuit

- Encode secrets from Alice and Bob, as well as the result f from the computation, all as binary strings.
- F then becomes a boolean function.
  - Implemented as a boolean circuit.
- In particular, a combinational circuit.
  - Whose size is proportional to the effort to compute F.
  - We will not distinguish F from its combinational circuit implementation.

NAND

- Secret from Alice: $a \in \{0, 1\}$
- Secret from Bob: $b \in \{0, 1\}$
- Can they compute $f = NAND(a, b)$ without revealing their own secrets?
  - If we could further extend this to any input bits and any number of NAND gates, then we could handle arbitrary combinational circuits.
- Note that for $f = NAND(a, b)$, if Bob chooses $b = 1$ then he can learn a from f.
  - This is allowed per definition of Secure Multi-Party Computation.
  - Not a concern if Bob chooses $b = 0$, or the circuit is much more complicated.

Idea of Garbled Circuit

- A collaboration between Alice and Bob.
- The garbler Alice garbles the circuit.
  - By encrypting every wire and every gate truth table.
  - Send Bob the garbled circuit.
  - Send Bob her input bits (encrypted).
  - Help Bob to encrypt his input bits.
- The evaluator Bob evalutes the garbled circuit.
  - Compute with encrypted boolean values.
  - Communicate with Alice to reveal the output bits.

Encrypting Wires

- For any wire W, Alice generates a random selection bit S_w.
- Then, Alice generates two random strings W_0 and W_1.
  - W_0 represents 0 and starts with S_w.
  - W_1 represents 1 and starts with $1 - S_w$.
- For the circuit $O = NAND(A, B)$, there are three wires.

Wire	Selection Bit	0	1
O	S_O	$O_0 = S_O \cdots$	$O_1 = (1 - S_O) \cdots$
A	S_A	$A_0 = S_A \cdots$	$A_1 = (1 - S_A) \cdots$
B	S_B	$B_0 = S_B \cdots$	$B_1 = (1 - S_B) \cdots$

Encrypting Wires (Cont.)

- For example, let's use 5 bits for each wire.

Wire	Selection Bit	0	1
O	$S_O = 1$	$O_0 = 10001 = 17$	$O_1 = 00101 = 5$
A	$S_A = 0$	$A_0 = 00110 = 6$	$A_1 = 10000 = 16$
B	$S_B = 1$	$B_0 = 10010 = 18$	$B_1 = 00010 = 2$

- Alice cannot send Bob the above table.
- However, for the computation to proceed, Bob need to know A_a and B_b, and then calculate O_f.
  - Without knowing a, b. (Note that Bob knows b for the simple NAND example we are working with. However, for more complex circuits, we should require Bob to not know b.)
  - Also without knowing f at this point, so that if there are multiple levels of gates, the computation may continue.

Garbled Circuit

S(A)	S(B)	E(O)
$S_A = 0$	$S_B = 1$	$e_{A_0,B_0}(O_1) = 6 + 18 + 5 \bmod 32 = 29$
$S_A = 0$	$1 - S_B = 0$	$e_{A_0,B_1}(O_1) = 6 + 2 + 5 \bmod 32 = 13$
$1 - S_A = 1$	$S_B = 1$	$e_{A_1,B_0}(O_1) = 16 + 18 + 5 \bmod 32 = 7$
$1 - S_A = 1$	$1 - S_B = 0$	$e_{A_1,B_1}(O_0) = 16 + 2 + 17 \bmod 32 = 3$

- With the help of an encryption function $e()$, Alice encrypts every gate truth table.
  - e will take A and B as key and O as the plaintext.
  - Subscripts are the actual boolean values, e.g. for A_0 and B_0, we should use O_1 because 0 NAND 0 = 1.
  - Let's use $e_{A||B}(O) = A + B + O \bmod 32$ for our example.
- Bob decrypts with this table to obtain O_f from A_a and B_b.
  - Using the first bit of A_a and B_b to identify the row.
  - E.g. for $A_a = 16$ and $B_b = 18$, the third row.
- Issue: if the table rows follow this particular order,
  - Bob will learn S_A and S_B and know what A and B represent.

Garbled Circuit (Cont.)

- Alice sorts the rows into $S(A)S(B) = 00, 01, 10, 11$.

S(A)	S(B)	E(O)
0	0	13
0	1	29
1	0	3
1	1	7

- With $A_a = 16$ and $B_b = 18$, Bob can decrypt to obtain $O_f = d_{16||18}(10) = 7 - 16 - 18 \bmod 32 = 5$
  - Locate the row with $E(O) = 7$ using the first bits of A and B.
  - Though Bob has no idea what a, b and f are.
- Could Bob learn other A, B, or O from the table?
  - We will answer this question in Homework 4 using an argument similar to OTP.

Input and Output

- For input wires,
  - Alice sends Bob A_a.
  - Alice uses OT to send Bob B_b.
    - Obviously Bob doesn't want Alice to know b.
- Once Bob calculates O_f, Alice tells what is f.
- Alice has no need to send Bob A_{1-a}.
- Could Alice also send Bob B_{1-b} to avoid using OT?
  - Alice cannot send Bob B_{1-b}.
  - Otherwise Bob can compute $O_{f'}$ from A_a and B_{1-b} and then $a = O_{f'} \oplus O_f$ since $f' = NAND(a, 1-b)$.

A More Complicated Circuit

- What about more complicated circuits?
  - E.g. $f = NAND(NAND(a, b), NAND(c, d))$ where Alice provides a and c while Bob provides b and d.
- Identify wires and gates before encrypting them.
  - Wires: A, B, C, D, X, Y, Z
  - Gate 1: $X = NAND(A, B)$
  - Gate 2: $Y = NAND(C, D)$
  - Gate 3: $Z = NAND(X, Y)$

The Garbler Alice: Encrypting Wires

Wire	Selection Bit	0	1
A	$S_A = 0$	$A_0 = 00110 = 6$	$A_1 = 10000 = 16$
B	$S_B = 1$	$B_0 = 10010 = 18$	$B_1 = 00010 = 2$
C	$S_C = 1$	$C_0 = 10100 = 20$	$C_1 = 00001 = 1$
D	$S_D = 1$	$D_0 = 11001 = 25$	$D_1 = 00111 = 7$
X	$S_X = 0$	$X_0 = 00111 = 7$	$X_1 = 11111 = 31$
Y	$S_Y = 0$	$Y_0 = 00000 = 0$	$Y_1 = 10101 = 21$
Z	$S_Z = 1$	$Z_0 = 10001 = 17$	$Z_1 = 00101 = 5$

The Garbler Alice: Encrypting Truth Tables

Gate 1		
S(A)	S(B)	E(X)
$S_A = 0$	$S_B = 1$	$e_{A_0,B_0}(X_1) = 6 + 18 + 31 \bmod 32 = 23$
$S_A = 0$	$1 - S_B = 0$	$e_{A_0,B_1}(X_1) = 6 + 2 + 31 \bmod 32 = 8$
$1 - S_A = 1$	$S_B = 1$	$e_{A_1,B_0}(X_1) = 16 + 18 + 31 \bmod 32 = 1$
$1 - S_A = 1$	$1 - S_B = 0$	$e_{A_1,B_1}(X_0) = 16 + 2 + 7 \bmod 32 = 25$

Gate 2		
S(C)	S(D)	E(Y)
$S_C = 1$	$S_D = 1$	$e_{C_0,D_0}(Y_1) = 20 + 25 + 21 \bmod 32 = 2$
$S_C = 1$	$1 - S_D = 0$	$e_{C_0,D_1}(Y_1) = 20 + 7 + 21 \bmod 32 = 16$
$1 - S_C = 0$	$S_D = 1$	$e_{C_1,D_0}(Y_1) = 1 + 25 + 21 \bmod 32 = 15$
$1 - S_C = 0$	$1 - S_D = 0$	$e_{C_1,D_1}(Y_0) = 1 + 7 + 0 \bmod 32 = 8$

Gate 3		
S(X)	S(Y)	E(Z)
$S_X = 0$	$S_Y = 0$	$e_{X_0,Y_0}(Z_1) = 7 + 0 + 5 \bmod 32 = 12$
$S_X = 0$	$1 - S_Y = 1$	$e_{X_0,Y_1}(Z_1) = 7 + 21 + 5 \bmod 32 = 1$
$1 - S_X = 1$	$S_Y = 0$	$e_{X_1,Y_0}(Z_1) = 31 + 0 + 5 \bmod 32 = 4$
$1 - S_X = 1$	$1 - S_Y = 1$	$e_{X_1,Y_1}(Z_0) = 31 + 21 + 17 \bmod 32 = 5$

The Evaluator Bob

- The garbled circuit sent by Alice

Gate 1		Gate 2		Gate 3	
S(A) S(B)	E(X)	S(C) S(D)	E(Y)	S(X) S(Y)	E(Z)
0 0	8	0 0	8	0 0	12
0 1	23	0 1	15	0 1	1
1 0	25	1 0	16	1 0	4
1 1	1	1 1	2	1 1	5

- Alice sends her inputs: $A_a = 16,\ C_c = 20$
- Alice sends Bob's inputs via OT: $B_1 = 2,\ D_1 = 7$
- Bob's calculation
  - $X_x = 25 - 16 - 2 \bmod 32 = 7$
  - $Y_y = 16 - 20 - 7 \bmod 32 = 21$
  - $Z_z = 1 - 7 - 21 \bmod 32 = 5$
- After Bob shares $Z_z = 5$ with Alice, both party learn the result $f = 1$.

Discussions

- The mechanism works with arbitrary number of NAND gates, and thus any combinational circuits.
  - Bob can evalute each gate following the topological ordering, without knowing what each gate inputs and gate output mean.
- Overall, there is constant amount of computation and communication per each NAND gate.
  - Efficient in theory.
- A lot of ongoing research to improve its practical performance
  - The mechanism works with arbitrary gates and some gates lile XOR and NOT can be implemented much more easily.

Summary

- Secure two-party computation via garbled circuit.