Secure Function Evaluation

Multi-Party Computation

\( n \) players \( P_i \) want to jointly evaluate \( y_i = f_i(x_1, \ldots, x_n) \), for public functions \( f_i \) so that

- \( x_i \) is the private input of \( P_i \)
- \( P_i \) eventually learns \( y_i = f_i(x_1, \ldots, x_n) \)
- \( \ldots \) and nothing else about \( x_j \) for \( j \neq i \)

Security Notions

- Privacy
- Correctness
- Fairness (much harder to get)
Secure Function Evaluation

**t-Privacy**

If t parties collude, they cannot learn more on the other inputs than from their own/known inputs and outputs.

Note that the knowledge of \( y_i \) can leak some information on the \( x_j \)'s.

**Security Models**

- **Honest-but-curious**: all the players follow the protocol honestly, but the adversary knows all the inputs/outputs from \( t \) users.
- **Malicious users**: the adversary controls a fixed set of \( t \) players.
- **Dynamic adversary**: the adversary dynamically chooses the (up to) \( t \) players it controls.

**Electronic Voting**

**Private Evaluation of the Sum**

For all \( i: x_i \in \{0, 1\} \) and \( f_i(x_1, \ldots, x_n) = \sum_j x_j \)

**Example (Homomorphic Encryption)**

- \( P_i \) encrypts \( C_i = E(x_i) \) with an additively homomorphic encryption scheme.
- They all compute \( C = E(\sum_P x_i) \).
- They jointly decrypt \( C \) to get \( y = \sum_x x_i \) using a distributed decryption.

**Electronic Voting**

**Privacy: Limitations**

In case of unanimity (i.e. \( \sum x_i = n \)), one learns all the \( x_i \)'s, even in the honest-but-curious setting.

This is not a weakness of the protocol, but of the functionality: one should just reveal the winner.

**Replay Attacks**

A malicious adversary could try to amplify \( P_1 \)'s vote, replaying its message \( C_1 \) by \( t \) corrupted players: this can leak \( P_1 \)'s vote \( x_1 \).

This can be avoided with non-malleable encryption.
Secure 2-Party Computation

The 2-party particular case: on Alice’s input $x$ and Bob’s input $y$,
Alice gets $f(x, y)$ and Bob gets $g(x, y)$, but nothing else

Equality Test

Alice owns a value $x \in [A, B]$ and Bob owns a value $y \in [A, B]$,
in the end, they both learn whether $x = y$ or not

With Homomorphic Encryption

- Alice encrypts $C = E(x)$
  with an additively homomorphic encryption scheme
- Bob computes $C^0 = E(r(x - y))$, for a random element $r$
- Alice computes $C^0 = E(r^0(x - y))$, for a random element $r^0$
- They jointly decrypt $C^\oplus_i$: the value is 0 iff $x = y$ (or random)

Yao Millionaires’ Problem

Alice owns an integer $x$ and Bob owns an integer $y$,
in the end, they both learn whether $x \leq y$ or not

Equality Test

Alice owns a value $x$ and Bob owns a value $y$,
in the end, they both learn whether $x = y$ or not

Yao Millionaires’ Problem

Alice owns an integer $x \in [0, 2^n[$ and Bob owns an integer $y \in [0, 2^n[$,
in the end, they both learn whether $x \leq y$ or not

Equalities Test

We fill and order the sets by length: $\bar{T}_x = \{T_x^1 \}$ and $\bar{T}_y = \{T_y^0 \}$ where
for $i = 0, \ldots, n$:
- if $x_i = 0$, $X_i = 2^n$, otherwise $X_i = x_{n-1} \ldots x_i \in [0, 2^{n-i}[$
- if $y_i = 1$, $Y_i = 2^n + 1$, otherwise $Y_i = y_{n-1} \ldots y_i + 1 \in [0, 2^{n-i}[$

$x > y \iff \exists i < n, X_i = Y_i$

With Homomorphic Encryption

- Alice encrypts $C_i = E(X_i)$
  with an additively homomorphic encryption scheme
- Bob computes $C_i^0 = E(r_i (X_i - Y_i))$, for random elements $r_i$
  and sends them in random order
- Alice computes $C_i^\oplus = E(\prod r_i^0 (X_i - Y_i))$, for random elements $r_i^0$
- They jointly decrypt $C_i^\oplus$: one value is 0 iff $x > y$
Outline

1 Secure Function Evaluation
   - Introduction
   - Examples
   - Malicious Setting

2 Oblivious Transfer

3 Garbled Circuits

GMW Compiler

GMW Compiler

- Commitment of the inputs
- Secure coin tossing
- Zero-knowledge proofs of correct behavior

Secure 2-Party Computation

The 2-party particular case: on Alice's input $x$ and Bob's input $y$, Alice gets $f(x, y)$ and Bob gets $g(x, y)$, but nothing else.

Oblivious Transfer

Alice owns two values $x_0, x_1$ and Bob owns a bit $b \in \{0, 1\}$, so that in the end, Bob learns $x_b$ and Alice gets nothing:

$x = (x_0, x_1)$ and $y = b$, then $f((x_0, x_1), b) = x_b$ and $g((x_0, x_1), b) = \perp$

Oblivious Transfer is equivalent to Secure 2-Party Computation

From an Oblivious Transfer Protocol, one can implement any 2-Party Secure Function Evaluation.
**Oblivious Transfer**

**Example (Bellare-Micali’s Construction – 1992)**

In a discrete logarithm setting \((G, g, p)\), for \(x_0, x_1 \in G\)

- Alice chooses \(c \leftarrow G\) and sends it to Bob
- Bob chooses \(k \leftarrow Z_p\), sets \(pk_b \leftarrow g^k\) and \(pk_{1-b} \leftarrow c/pk_b\), and sends \((pk_0, pk_1)\) to Alice
- Alice checks \(pk_0 \cdot pk_1 = c\)
- Alice encrypts \(x_i\) under \(pk_i\) for \(i = 0, 1\) with ElGamal:
  \[ C_i \leftarrow g^{r_i} \text{ and } C_i^0 \leftarrow x_i \cdot pk_i^{r_i} \text{, for } r_i \leftarrow Z_p \]
- Bob can decrypt \((C_b, C_b^0)\) using \(k\)

Because of the random \(c\) (unknown discrete logarithm), Bob should not be able to infer any information about \(x_{1-b}\)

This is provably secure in the **honest-but-curious setting**

**Example (Naor-Pinkas Construction – 2000)**

In a discrete logarithm setting \((G, g, p)\), for \(x_0, x_1 \in G\)

- Bob chooses \(r, s, t \leftarrow Z_p\), sets \(X \leftarrow g^r\), \(Y \leftarrow g^s\), \(Z_b \leftarrow g^{rs}\), \(Z_{1-b} \leftarrow g^t\), and sends \((X, Y, Z_0, Z_1)\) to Bob
- Alice checks \(Z_0 \neq Z_1\), and re-randomizes the tuples:
  \[ T_0 \leftarrow (X, Y_0^0 = Y_0 u_0 g^{v_0}, Z_0^0 = Z_0 u_0 X_0^0) \text{ and } \]
  \[ T_1 \leftarrow (X, Y_1^0 = Y_1 u_1 g^{v_1}, Z_1^0 = Z_1 u_1 X_1), \text{ for } u_0, v_0, u_1, v_1 \leftarrow Z_p \]
- Alice encrypts \(x_i\) under \(T_i\):
  \[ C_i = Y_i^0 \text{ and } C_i^0 = x_i \cdot Z_i^0 \]
- Bob can decrypt \((C_b, C_b^0)\) using \(r\)

The re-randomization keeps the DH-tuple \(T_b\), but perfectly removes information in \(T_{1-b}\)

This is provably secure in the **malicious setting**
Boolean Circuit

Boolean circuit, Alice’s inputs \((x_1, x_2, x_3)\), and Bob’s inputs \((y_1, y_2, y_3)\):

They both learn \(z\) in the end, but nothing else.

Garbled Circuit

Alice converts the circuit into a generic circuit: 1-input or 2-input gates:

1-Input Garbled Gate
For the gate \(A\) (not): 4 random secret keys \(I^0_A, I^1_A, O^0_A, O^1_A\)

\[
A = \begin{bmatrix} 1 & 0 \\ 0 & 0 \\ 0 & 1 \\ 0 & 1 \end{bmatrix}
\]

\[
A = 10 \:
C^0_A = Encrypt(I^0_A, O^1_A) \quad C^1_A = Encrypt(I^1_A, O^0_A)
\]

2-Input Garbled Gate
For the gate \(B\) (and): 8 random secret keys \(I^0_B, I^1_B, J^0_B, J^1_B, O^0_B, O^1_B\)

\[
B = \begin{bmatrix} 0 & 0 \\ 0 & 1 \\ 1 & 0 \\ 1 & 1 \end{bmatrix}
\]

\[
B = 00 \:
C^0_B = Encrypt(I^0_B || J^0_B, O^0_B) \quad C^{01}_B = Encrypt(I^0_B || J^1_B, O^0_B) \quad C^{10}_B = Encrypt(I^1_B || J^0_B, O^0_B) \quad C^{11}_B = Encrypt(I^1_B || J^1_B, O^0_B)
\]

Garbled Gates

Alice generates the garbled gates.
Alice's Inputs

Alice publishes the ciphertexts in random order for each gate

Alice publishes the keys corresponding to her inputs:
- For $x_1$, she sends $I^{x_1}_A$
- For $x_2$, she sends $J^{x_2}_B$
- For $x_3$, she sends $J^{x_3}_C$

Bob's Inputs

$$Y_1 \rightarrow A \rightarrow Y_A$$

$$A = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix} : C^0_A = \text{Encrypt}(I^0_A, O^1_A) \quad C^1_A = \text{Encrypt}(I^1_A, O^0_A)$$

Oblivious Transfer

Alice owns $I^0_A, I^1_A$ and Bob owns $y_1 \in \{0, 1\}$
- Using an OT, Bob gets $I^{y_1}_A$, while Alice learns nothing
- From the ciphertexts $(C^b_A)_b$, Bob gets $O^{y_b}_A$

Bob's Inputs

$$B = \begin{bmatrix} 0 & 0 \\ 0 & 1 \end{bmatrix} : C^0_B = \text{Encrypt}(I^0_B \| J^0_B, O^0_B) \quad C^0_1 = \text{Encrypt}(I^0_B \| J^1_B, O^0_B)$$

$$C^1_0 = \text{Encrypt}(I^1_B \| J^0_B, O^0_B) \quad C^1_1 = \text{Encrypt}(I^1_B \| J^1_B, O^0_B)$$

Oblivious Transfer

Alice owns $I^0_B, I^1_B$, and Bob owns $y_2 \in \{0, 1\}$
- Using an OT, Bob gets $I^{y_2}_B$, while Alice learns nothing
- Bob additionally knows $J^{x_2}_B$
- From the ciphertexts $(C^{bb'}_B)_{bb'}$, Bob gets $O^{y_b}_B$

Internal Garbled Gates

For the gate $E$ (or): 2 new random secret keys $O^0_E, O^1_E$
while $I^0_E \leftarrow O^0_A, I^1_E \leftarrow O^1_A, J^0_E \leftarrow O^0_B, J^1_E \leftarrow O^1_B$

$$E = \begin{bmatrix} 0 & 1 \\ 1 & 1 \end{bmatrix} : C^{00}_E = \text{Encrypt}(I^0_E \| J^0_E, O^0_E) \quad C^{01}_E = \text{Encrypt}(I^0_E \| J^1_E, O^1_E)$$

$$C^{10}_E = \text{Encrypt}(I^1_E \| J^0_E, O^1_E) \quad C^{11}_E = \text{Encrypt}(I^1_E \| J^1_E, O^1_E)$$
Evaluation of Internal Gates

\( E = \begin{bmatrix} 0 & 1 \\ 1 & 1 \end{bmatrix} \)

- \( C_{E}^{00} = \text{Encrypt}(I_{0}^{E} || J_{0}^{E}, O_{E}^{i}) \)
- \( C_{E}^{01} = \text{Encrypt}(I_{0}^{E} || J_{1}^{E}, O_{E}^{i}) \)
- \( C_{E}^{10} = \text{Encrypt}(I_{1}^{E} || J_{0}^{E}, O_{E}^{i}) \)
- \( C_{E}^{11} = \text{Encrypt}(I_{1}^{E} || J_{1}^{E}, O_{E}^{i}) \)

**Evaluation of Gate E**

Bob knows \( I_{E}^{A} = O_{A}^{i} \) and \( J_{E}^{B} = O_{B}^{i} \)

From the ciphertexts \( (C_{E}^{bb'})_{bb'} \), Bob gets \( O_{E}^{i} \)

Output Garbled Gates

Output Garbled Gate

For the gate \( G \) (or):

- \( I_{G}^{0} \leftarrow O_{E}^{i}, I_{G}^{1} \leftarrow O_{F}^{f}, J_{G}^{0} \leftarrow O_{E}^{i}, J_{G}^{1} \leftarrow O_{F}^{f} \)

\( G = \begin{bmatrix} 0 & 1 \\ 1 & 1 \end{bmatrix} \)

- \( C_{G}^{00} = \text{Encrypt}(I_{G}^{0} || J_{0}^{G}, 0) \)
- \( C_{G}^{01} = \text{Encrypt}(I_{G}^{0} || J_{1}^{G}, 1) \)
- \( C_{G}^{10} = \text{Encrypt}(I_{G}^{1} || J_{0}^{G}, 1) \)
- \( C_{G}^{11} = \text{Encrypt}(I_{G}^{1} || J_{1}^{G}, 1) \)

**Evaluation of Gate G**

Bob knows \( I_{G}^{F} = O_{E}^{f} \) and \( J_{G}^{F} = O_{F}^{f} \)

From the ciphertexts \( (C_{G}^{bb'})_{bb'} \), Bob gets \( z \in \{0, 1\} \)

Bob can then transmit \( z \) to Alice

Outline

1. Secure Function Evaluation
2. Oblivious Transfer
3. Garbled Circuits
   - Introduction
   - Garbled Circuits
   - Correctness
The previous construction assumes that

- Bob extracts the correct plaintext among the multiple candidates
  \[ \Rightarrow \text{Redundancy is added to the plaintext} \]
  (or authenticated encryption)

They have to trust each other

- Alice correctly builds garbled gates: the ciphertexts are correct
  \[ \Rightarrow \text{Cut-and-choose technique} \]

- Alice plays the oblivious transfer protocols with correct inputs
  \[ \Rightarrow \text{Inputs are committed, checked during the cut-and-choose,} \]
  and ZK proofs are done during the OT

- Bob sends back the correct value \( z \)
  \[ \Rightarrow \text{Random tags are appended to the final results 0 and 1} \]
  that Bob cannot guess