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 \( C = E(\sum x_i) \)
- They jointly decrypt \( C \) to get \( y = \sum x_i \) 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 $x$ and Bob owns a value $y$,
in the end, they both learn whether $x = y$ or not

Yao Millionaires’ Problem

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

With Homomorphic Encryption

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

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

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

Theorem [Lin-Tzeng – 2005]

Given $x = x_{n-1} \ldots x_0, y = y_{n-1} \ldots y_0 \in \{0, 1\}^n$, and denoting

$$T_x^1 = \{x_{n-1} \ldots x_i | x_i = 1\}, \quad T_y^0 = \{y_{n-1} \ldots y_{i+1} | y_i = 0\}$$

$$x > y \iff T_x^1 \cap T_y^0 \neq \emptyset$$

We fill and order the sets by length: $\bar{T}_x^1 = \{X_i\}$ and $\bar{T}_y^0 = \{Y_i\}$ 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 $C_i' = E(r_i(X_i - Y_i))$, for random elements $r_i$
  and sends them in random order
- Alice computes $C_i'' = E(r_i'(X_i - Y_i))$, for random elements $r_i'$
- They jointly decrypt the $C_i''$s: one value is 0 iff $x > y$
Outline

1 Secure Function Evaluation
   - Introduction
   - Examples
   - Malicious Setting

2 Oblivious Transfer

3 Garbled Circuits

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.
Outline

1 Secure Function Evaluation
2 Oblivious Transfer
   - Definition
   - Examples
3 Garbled Circuits

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 \(k \leftarrow \mathbb{Z}_p\), sets \(pk_b \leftarrow g^k\) and \(pk_{1-b} \leftarrow c/pk_b\), and sends \((pk_0, pk_1)\) to Alice
- Alice checks \(pk_0 \cdot pk_1 = c\) and encrypts \(x_i\) under \(pk_i\) (for \(i = 0, 1\)) with ElGamal:
  \[ C_i \leftarrow g^{r_i} \text{ and } C_i' \leftarrow x_i \cdot pk_i^{r_i}, \text{ for } r_i \leftarrow \mathbb{Z}_p \]
- Bob can decrypt \((C_b, C_b')\) 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 \(r, s, t \leftarrow \mathbb{Z}_p\), 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:
  
  \[ T_0 \leftarrow (X, Y_0' = Y^{u_0} g^{v_0}, Z_0' = Z_0^{u_0} X^{v_0}) \text{ and} \]
  
  \[ T_1 \leftarrow (X, Y_1' = Y^{u_1} g^{v_1}, Z_1' = Z_1^{u_1} X^{v_1}), \text{ for } u_0, v_0, u_1, v_1 \leftarrow \mathbb{Z}_p \]
- Alice encrypts \(x_i\) under \(T_i\): \(C_i = Y_i'\) and \(C_i' = x_i \cdot Z_i'\)
- Bob can decrypt \((C_b, C_b')\) 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, 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.

---

**Outline**

1. Secure Function Evaluation
2. Oblivious Transfer
3. Garbled Circuits
   - Introduction
   - Garbled Circuits
   - Correctness

---

**Garbled Circuit**

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

**Garbled Gates**

Alice generates the garbled gates

1-Input Garbled Gate

For the gate A (not): 4 random secret keys \(I_A^0, I_A^1, O_A^0, O_A^1\)

\[
A = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix} \quad \text{not}
\]

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

\[
C = \begin{bmatrix} 0 & 1 \\ 1 & 1 \end{bmatrix} \quad \text{or}
\]

\[
D = \begin{bmatrix} 0 & 1 \\ 1 & 1 \end{bmatrix} \quad \text{line}
\]

\[
E = \begin{bmatrix} 0 & 1 \\ 0 & 1 \end{bmatrix} \quad \text{or}
\]

\[
F = \begin{bmatrix} 0 & 0 \\ 1 & 1 \end{bmatrix} \quad \text{and}
\]

\[
G = \begin{bmatrix} 0 & 1 \\ 1 & 1 \end{bmatrix} \quad \text{or}
\]

---

2-Input Garbled Gate

For the gate B (and): 8 random secret keys \(I_B^0, I_B^1, J_B^0, J_B^1, O_B^0, O_B^1\)

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

\[
C_B^{00} = \text{Encrypt}(I_B^0 || J_B^0, O_B^0)
\]

\[
C_B^{01} = \text{Encrypt}(I_B^1 || J_B^0, O_B^0)
\]

\[
C_B^{10} = \text{Encrypt}(I_B^0 || J_B^1, O_B^0)
\]

\[
C_B^{11} = \text{Encrypt}(I_B^1 || J_B^1, O_B^0)
\]
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

Bob’s Inputs

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

Oblivious Transfer

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

Internal Garbled Gates

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

Oblivious Transfer

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

Internal Garbled Gate

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

$$E = \begin{bmatrix} 0 & 1 \\ 1 & 1 \end{bmatrix} : C^0_E = \text{Encrypt}(I^0_E || J^0_E, O^0_E), C^0_E = \text{Encrypt}(I^0_E || J^1_E, O^1_E), C^1_E = \text{Encrypt}(I^1_E || J^0_E, O^1_E)$$
Evaluation of Internal Gates

\[ E = \begin{bmatrix} 0 & 1 \\ 1 & 1 \end{bmatrix} : C'^{00}_E = \text{Encrypt}(I'_0 \mathbin{||} J'_0, O'_0) \quad C'^{01}_E = \text{Encrypt}(I'_0 \mathbin{||} J'_1, O'_1) \]
\[ C'^{10}_E = \text{Encrypt}(I'_1 \mathbin{||} J'_0, O'_0) \quad C'^{11}_E = \text{Encrypt}(I'_1 \mathbin{||} J'_1, O'_1) \]

Evaluation of Gate E

Bob knows \( I'_A = O'_A \) and \( J'_B = O'_B \)
From the ciphertexts \( (C'^{bb'}_{bb'})_E \), Bob gets \( O'^{y'_E}_E \)

Output Garbled Gates

Output Garbled Gate

For the gate G (or): \( I'_G \leftarrow O'_G, I'_G \leftarrow O'_F, J'_G \leftarrow O'_G, J'_G \leftarrow O'_F \)

\[ G = \begin{bmatrix} 0 & 1 \\ 1 & 1 \end{bmatrix} : C'^{00}_G = \text{Encrypt}(I'_G \mathbin{||} J'_0, 0) \quad C'^{01}_G = \text{Encrypt}(I'_G \mathbin{||} J'_1, 1) \]
\[ C'^{10}_G = \text{Encrypt}(I'_1 \mathbin{||} J'_0, 1) \quad C'^{11}_G = \text{Encrypt}(I'_1 \mathbin{||} J'_1, 1) \]

Evaluation of Gate G

Bob knows \( I'_E = O'_E \) and \( J'_F = O'_F \)
From the ciphertexts \( (C'^{bb'}_{bb'})_G \), 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
Honest-but-Curious and Malicious

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