Outline

Secure Computation

Garbled Circuits
Oblivious Transfer
Oblivious Transfer

S

\[ m_0, m_1 \]

R

b

postprocessing: \( m_b \)

Doesn't learn b.

 Doesn't learn \( m_{1-b} \)
OT: Security against Malicious Receivers

"Ideal Expmt"

Sim.

View Bob

m₀, m₁ → m₀, m₁

Cannot contain info. about m₁-b!

(1) Sim has to pretend to be Alice
(2) Sim has to "find" implicit b that Bob uses.
(3) Sim must generate view w/o knowledge of m₁-b

View Bob

"Real World/Expmt"

does not contain info. about m₁-b, by virtue of indistinguishability from the ideal view.
OT: Security against Malicious Senders

"Real World"

"Ideal World"
Use PKE with pseudorandom public keys.

Oblivious Transfer Secure Against Malicious Adversaries:

\[ \text{construction} \]

\[ \begin{align*}
\text{pt}_0, \text{pt}_1 \in \{0,1\} &  \\
\text{ct}_0 = \text{Enc}_{\text{pk}_0}(\text{pt}_0) &  \\
\text{ct}_1 = \text{Enc}_{\text{pk}_1}(\text{pt}_1) &  \\
\end{align*} \]

1) Sample \((pk, sk)\)
2) \(r_0, r_1\) s.t. \(H(r_0) \oplus r_i = pk\)
3) If \(b = 0\), \(s_0 = r_0, s_1 = r_1\),
   If \(b = 1\), \(s_0 = r_1, s_1 = r_0\)
[2019]

Oblivious Transfer Secure Against Malicious Adversaries:

\[ \text{PKE with pseudorandom public keys} \] \hspace{2cm} \text{CONSTRUCTION}

\[ \text{RANDOM ORACLE} \]

Bob: Has input \( b \)

\( \begin{align*}
& (S_0, S_1) \\
\end{align*} \)

\( \begin{align*}
& (pk, sk) \leftarrow G \hspace{1cm} 5) \text{ Set } pk := r_0 \oplus H(r) \\
\end{align*} \)

\( \begin{align*}
& ct_0, ct_1 \\
\end{align*} \)

\( \begin{align*}
& m_b \leftarrow \text{Dec}_{sk}(ct_b) \\
\end{align*} \)

\( \begin{align*}
& r_1 := H(r_0) \oplus pk \hspace{1cm} 4) \text{ Set } r_1 := H(r_0) \oplus pk \\
\end{align*} \)

\( \begin{align*}
& \text{Note: } pk = r_1 \oplus H(r) \\
\end{align*} \)

\( \begin{align*}
& \text{Sample } (pk, sk) \leftarrow G \hspace{1cm} 5) \text{ Set } pk := r_0 \oplus H(r) \\
\end{align*} \)

\( \begin{align*}
& s_0 \leftarrow r_0, s_1 \leftarrow r_1 \\
\end{align*} \)

If \( b = 0 \), then \( s_0, r_0, s_1, r_1 \)

If \( b = 1 \), then \( s_1, r_0, s_0, r_1 \)
Oblivious Transfer Secure Against Malicious Senders

Simulator's goal: Sample $\tilde{s}_0, \tilde{s}_i$ s.t. they know $(sk_0, sk_i)$ for $(pk_0, pk_i)$ where $pk_0 = H(\tilde{s}_0) \oplus s_i$ and $pk_i = H(s_i) \oplus \tilde{s}_0$.

Sim will first sample $(pk_0, sk_0) \leftarrow \text{Gen}$ and $(pk_i, sk_i) \leftarrow \text{Gen}$. Sample $(\tilde{s}_0, \tilde{s}_i)$ and set $H(\tilde{s}_0) = pk_0 \oplus \tilde{s}_i$. It will choose $\tilde{s}_i$ independently conditioned on $\tilde{s}_0$.
Oblivious Transfer Secure Against Malicious Senders
Oblivious Transfer Secure Against Malicious Receivers
Oblivious Transfer Secure Against Malicious Receivers

\[ \text{Alice: } m_0, m_1 \quad \text{end} \]
\[ \text{Bob: } b \]
\[ \text{Oracle: } m_0, m_1 \]
\[ \text{Adv: } S_0, S_1 \]

\[ \text{Alice: } pk_0 = H(S_0) \oplus S_1, \quad pk_1 = H(S_1) \oplus S_0 \]

Adv knows sk_B.

What is \( b \)?
Oblivious Transfer Secure Against Malicious RECEIVERS

Sim's algorithm:

1. Keep an ordered list of all of Bob's RO queries.
2. On receiving \((s_0, s_1)\):
   a. Both \((s_0, s_1)\) were not queried. Bob learns nothing, potentially.
   b. Only \(s_0\) was queried. Bob knows \(sk\) for \(pk = H(s_0) \oplus H(s_1)\).
   c. Only \(s_1\) was queried. Bob pot. knows \(sk\) for \(pk = H(s_1) \oplus H(s_0)\).
   d. Both were queried. Find which was queried first. If \(s_0\) was queried first, set \(r_1 = pk' \oplus H(r_0)\). If \(s_1\) was queried first, set \(r_1 = pk' \oplus H(r_0)\).
Given RO queries and $s_0$ and $s_1$, figure out what $b$ is.
Secure XOR

\[ f = \text{XOR} \]

\[ a \oplus b \]

\[ a \rightarrow b \]

\[ b \rightarrow a \]
Secure AND Computation from OT

\[a \land 0 = m_0\]
\[a \land 1 = m_1\]
\[m_c = a \land c = a \land b\]
Secure ω Computation from OT

\[
\begin{array}{cccc}
0 & 0 & 0 \\
0 & 1 & 1 \\
1 & 0 & 1 \\
1 & 1 & 1 \\
\end{array}
\]

\[a \lor b = m_c = a \lor c = a \lor b.\]
How about more general circuits?

Boolean Circuit (informal): a collection of logical gates that are connected by wires, where each wire represents a particular boolean value. Inputs to the circuit are placed onto input wires, these wires are fed through a series of logic gates, and eventually one or more outputs are produced.
General circuits?
Secret Sharing (2 out of 2)

\[\text{SS}(a) \rightarrow (a_1, a_2) \text{ s.t. } a_1 + a_2 = a.\]

[Shamir's Secret Sharing]

\[a \rightarrow (a_1, \ldots, a_n) \text{ s.t. any } t \text{-sized subset determines } a.\]
XOR on secret shared data
Alice & Bob distributed shares of inputs

\[ a \xrightarrow{ss} (a_1, a_2) \]

\[ a_2 \rightarrow b_1 \]

\[ b \xrightarrow{ss} (b_1, b_2) \]
XOR on secret-shared data

\[ f = \text{XOR} \]

\[ \begin{array}{c}
\text{END UP WITH}
\hline
\text{SHARES OF } f(a, b) = a \oplus b.
\hline
\text{d}_1, \text{d}_2 \text{ s.t. } \text{d}_1 \oplus \text{d}_2 = a \oplus b.
\end{array} \]

\[ \begin{array}{c}
\text{a}_1, \text{b}_1
\end{array} \]

\[ \begin{array}{c}
\text{a}_2, \text{b}_2
\end{array} \]
AND on secret-shared data

\[ a_1, a_2 \]

\[ (a_1, b_1) \]

\[ (a_1 \land b_1) \]

\[ d_1 \]

\[ (a_2, b_2) \]

\[ (a_2 \land b_2) \]

\[ b_1, b_2 \]

\[ (a_2^\prime, b_2^\prime) \]

\[ d_2 \]

s.t. \[ d_1 \oplus d_2 = (a_1 \land b_1) = (a_1 \oplus a_2) \land (b_1 \oplus b_2) \]

\[ \geq (a_1 \land b_1) \oplus (a_1 \land b_2) \oplus (a_2 \land b_1) \oplus (a_2 \land b_2) \]
on secret-shared data

(Alice, Bob) have \((c_1, c_2)\) as S.S. of \(c\).
End up with \((d_1, d_2)\) as S.S. of \(1-c\).
One of them (Bob) compute \((1-d_2)\).
General circuits on Shared Data