Garbled circuits and oblivious transfer with security against malicious adversaries

29.10.2019
Recall: garbled circuits

- Two parties: Garbler and Evaluator
- A boolean circuit $C$
- $G \rightarrow E$: keys corresponding to G’s input bits
- $G \rightarrow E$: keys corresponding to E’s input bits
  - Using oblivious transfer
- $G$ garbles the circuit, sends it to $E$
- $E$ evaluates the circuit with the keys he received
- $E$ sends results back to $G$
Security against malicious $E$

Let only $E$ receive the outputs of secure computation

- $E$ sends out messages only as part of oblivious transfer protocols
  - We’ll see maliciously secure oblivious transfer protocols pretty soon
- Security against passive $E \Rightarrow$ Security against active $E$

From single-party output to two-party output

- Output to $G$ is included in the output to $E$
  - Encrypted, then MAC-ed. Computed as part of $C$
  - $E$ sends it back to $G$
- Use OTP for encryption. And inf.-theor. secure MAC
  - $m \in \mathbb{F}$. $k = (a, b) \in \mathbb{F}^* \times \mathbb{F}$. $\text{MAC}_{a,b}(m) = am + b$
Some problems a malicious $G$ can cause

- Garbled circuit does not correspond to the original circuit
- Some gates are faulty
  - A key output by a gate (or an input) does not decrypt the next gate
  - Perhaps one of the keys is faulty and the other one is not. Whether $E$ stops or not, tells $G$ the value on this wire
Cut-and-choose

- $G$ has to prepare a complex object, according to specifications
- To make sure $G$ has correctly garbled the circuit:
  - $G$ garbles many copies of the circuit (let $s$ be the number)
  - $E$ select all but one of them
  - $G$ opens all selected copies
    - Hands all used randomness (i.e. the input keys) over to $E$
  - $E$ verifies that all opened copies are correct
  - $G$ and $E$ use the unopened copy for the actual computation
- $G$’s probability of successful cheating: $1/s$
Using many unopened copies

- Let us try like this:
  - $G$ garbles $s$ copies
  - $E$ selects $\phi s$ of them for opening ($0 < \phi < 1$)
    - Let $I \subset \{1, \ldots, s\}$ be the indices of opened copies
  - $G$ and $E$ use the remaining $(1 - \phi)s$ copies for actual computation
- $E$ learns the outputs of all unopened copies
- $E$ picks the outputs of the computation by majority vote
  - If $E$ catches $G$ cheating, he may not let him know
  - Indeed, catching may be dependent on the inputs of $E$
- New problem — all evaluation copies must receive same inputs
  - Both from $G$ and from $E$
What should $s$ and $\phi$ be?

- $G$ garbles $s$ copies of the circuit. Out of these $t$ copies are bad.
- $G$ wants that all $\phi s$ opened copies are good.
- $G$ also wants the majority of $(1 - \phi)s$ evaluated copies to be bad.
- Hence $t = (1 - \phi)s/2$
  - Less is too little.
  - More only increases the chance of getting caught.

- Now find the probability of $E$ catching a cheating $G$.
- Imagine that $E$ always selects the first $\phi s$ copies. But $G$ must randomly place the good and bad copies.

\[
\begin{array}{c|c}
\text{number of all possibilities} & \text{number of passing possibilities} \\
\binom{s}{(1-\phi)s/2} & \binom{(1-\phi)s}{(1-\phi)s/2}
\end{array}
\]
Probability of passing

A combinatorial exercise... 
Remember Stirling’s approximation: $n! \approx \sqrt{2\pi n} \cdot n^ne^{-n}$
for zero-knowledge proofs

- There is relation $ R \subseteq \{0, 1\}^* \times \{0, 1\}^*$, $ R \in P $
- Two parties: prover $ P $ and verifier $ V $
- $ P $ knows $ x, w \in \{0, 1\}^* $. $ V $ knows $ x $
- $ P $ wants to convince $ V $ that he knows $ w $, such that $(x, w) \in R$

Functionality $ \mathcal{F}_Z^R $

- Receive (prove, sessionId, $ x, w $) from $ P $. Ignore, if $(x, w) \notin R$
- Send (proofReceived, sessionId, $ |x| $) to $ A $
- Receive (sendProof, sessionId) from $ A $
- Send (proven, sessionId, $ x $) to $ V $

Such $ \mathcal{F} $-s are part of the “real system”
$\Sigma$-protocols

- $P$ has $x, w$. $V$ has $x$
- $P$ sends $\alpha$. $V$ responds with the challenge $\beta$. $P$ sends response $\gamma$. $V$ accepts or rejects.
- Completeness: if $(x, w) \in R$, then $V$ accepts
- Special soundness: if $(\alpha, \beta, \gamma)$ and $(\alpha, \beta', \gamma')$ are both accepting transcripts, then $w$ can be found from them.
- Simulatability: Given $(x, \beta)$, can generate $(\alpha, \gamma)$ so, that $(\alpha, \beta, \gamma)$ is indistinguishable from conversations between honest $P$ and $V$ on $x$

A $\Sigma$-protocol for $R$ can be converted to a ZK proof implementation for $R$
Diffie-Hellman tuples

- Let $\mathbb{G}$ be a cyclic group of size $q$ (with hard DH problem)
- $R \subseteq \mathbb{G}^4 \times \mathbb{Z}_q$ (DH tuples)

$((g, h, h_1, h_2), w) \in R \iff h_1 = g^w \land h_2 = h^w$

- There exists a $\Sigma$-protocol to show that $(h, h_1, h_2)$ is a DH tuple
  - $P$ picks $r \leftarrow \mathbb{Z}_q$, sends $(\alpha_1, \alpha_2) = (g^r, h^r)$ to $V$
  - $V$ responds with random $\beta \in \mathbb{Z}_q$
  - $P$ sends $\gamma = r + \beta w$ to $V$
  - $V$ accepts if $g^\gamma = \alpha_1 \cdot h_1^\beta$ and $h^\gamma = \alpha_2 \cdot h_2^\beta$

Exercise. Special soundness? Simulatability?
Σ-protocols for a subset of claims

- $P$ and $V$ have $x_1, \ldots, x_n$. Prover has $\{w_i\}_{i \in I}$, where $I \subseteq \{1, \ldots, n\}$, $|I| = k$, $I$ is private
- $P$ wants to show that he has witnesses for at least $k$ of $x_1, \ldots, x_n$
- $P$ randomly chooses $\beta_j \in \mathbb{F}$ for all $j \notin I$, simulates $\alpha_j, \gamma_j$.
- $P$ picks $\alpha_j$ for $j \in I$ as needed. Sends $\alpha = (\alpha_1, \ldots, \alpha_n)$ to $V$
- $V$ responds with $\beta \in \mathbb{F}$.
- $P$ picks polynomial $f$ so, that $f(0) = \beta$, $f(j) = \beta_j$ for all $j \notin I$ and $\deg f \leq n - k$
- $P$ defines $\beta_i = f(i)$ and computes the response $\gamma_i$ for all $i \in I$.
- $P$ sends $\gamma = (f, \gamma_1, \ldots, \gamma_n)$ to $V$
- $V$ checks $\deg f$ and $f(0)$, recomputes $\beta_i$, checks $\gamma_i$ for all $i$
Peikert et al.’s OT

- Group $\mathbb{G}$, generator $g_0$, size $q$. $S$ has $m_0, m_1 \in \mathbb{G}$. $R$ has $b \in \{0, 1\}$

- $R$ picks $y, \alpha, r \leftarrow \mathbb{Z}_q$. Computes

$$
g_1 = g_0^y \quad h_0 = g_0^\alpha \quad h_1 = g_1^{\alpha+1} \quad g' = g_b^r \quad h' = h_b^r$$

Sends them all to $S$ and proves $(g_0, g_1, h_0, h_1/g_1)$ is a DH tuple

- I.e. $(g_0, g_1, h_0, h_1)$ is not a DH tuple

- $S$ sends to $R$ pairs $(u_0, v_0) \leftarrow R(m_0, g_0, g', h_0, h')$ and $(u_1, v_1) \leftarrow R(m_1, g_1, g', h_1, h')$, where

$$R(m, w, x, y, z) = \text{pick } s, t \leftarrow \mathbb{Z}_g \text{ in } (w^sy^t, mx^sz^t)$$

- $R$ finds $m_b = v_b/u_b^r$
On correctness and security

- If \((g, g', h, h')\) is not a DH tuple, then \((g^s h^t, (g')^s (h')^t)\) is a random pair of group elements
- Hence one of \(m_0, m_1\) is masked with a random element
- The protocol is secure against malicious \(S\) and malicious \(R\)
- If \((g, g', h, h')\) is a DH tuple, then \((g^s h^t, (g')^s (h')^t)\) extends this DH tuple
- If \((g_0, g_1, h_0, h_1)\) had been a DH tuple, then \(R\) could have recovered both \(m_0\) and \(m_1\)
  \(\ldots\) using the knowledge about discrete logarithms of \(g_1, h_0\)
Cut-and-choose OT

- $S$ has $m_{0,1}, m_{1,1}, \ldots, m_{0,s}, m_{1,s}$. $R$ has $b_1, \ldots, b_s$ and $I \subseteq \{1, \ldots, s\}$ with $|I| \leq s/2$.

- As result, $S$ gets nothing. $R$ gets $m_{b_i,i}$ for all $i$. $R$ additionally gets $m_{1-b_i,i}$ for $i \in I$.

- Protocol:
  - $s$ parallel copies of Peikert’s OT
  - But $R$ proves that in only at least $s/2$ copies, $(g_0, g_1, h_0, h_1/g_1)$ is a DH tuple
  - In instances not in $I$, $R$ lets $(g_0, g_1, h_0, h_1)$ to be a DH tuple
Using cut-and-choose OT in GC

- There’s the circuit $C$. It has $\ell$ inputs of $E$
- $G$ garbles $s$ copies of the circuit

\[
\begin{pmatrix}
k_{0,1}^{(1)}, k_{1,1}^{(1)} & \ldots & k_{0,s}^{(1)}, k_{1,s}^{(1)} \\
\vdots & \ddots & \vdots \\
k_{0,1}^{(\ell)}, k_{1,1}^{(\ell)} & \ldots & k_{0,s}^{(\ell)}, k_{1,s}^{(\ell)}
\end{pmatrix}
\quad \text{G has}
\quad \begin{pmatrix}
b_{1}^{(1)} & \ldots & b_{s}^{(1)} & l^{(1)} \\
\vdots & \ddots & \vdots & \vdots \\
b_{1}^{(\ell)} & \ldots & b_{s}^{(\ell)} & l^{(\ell)}
\end{pmatrix}
\quad \text{E has}
\]

The values in each red box must be equal to each other. $E$ must prove to $G$ that this is the case.
Using cut-and-choose OT in GC

- There’s the circuit $C$. It has $\ell$ inputs of $E$
- $G$ garbles $s$ copies of the circuit

\[
\begin{pmatrix}
  k_{0,1}^{(1)}, k_{1,1}^{(1)} & \cdots & k_{0,s}^{(1)}, k_{1,s}^{(1)} \\
  \vdots & \ddots & \vdots \\
  k_{0,1}^{(\ell)}, k_{1,1}^{(\ell)} & \cdots & k_{0,s}^{(\ell)}, k_{1,s}^{(\ell)}
\end{pmatrix}
\]

- The values in each red box must be equal to each other
- $E$ must prove to $G$ that this is the case
Proving \( b_1 = \cdots = b_s \)

- \( R \) has \( b \in \{0, 1\} \) and \( I \subseteq \{1, \ldots, s\} \). \( S \) has
  \( m_{0,1}, m_{1,1}, \ldots, m_{0,s}, m_{1,s} \in \mathbb{G} \)

- \( R \) picks \( y, \alpha_1, \ldots, \alpha_s, r \leftarrow \mathbb{Z}_q \). Computes

\[
\begin{align*}
g_1 &= g_0^y \\
h_{0,i} &= g_0^{\alpha_i} \\
h_{1,i} &= g_1^{\alpha_i + [i \notin I]} \\
g' &= g_b^r \\
h'_i &= h_{b,i}^r
\end{align*}
\]

Sends them all to \( S \). Proves that \( (g_0, g_1, h_{0,i}, h_{1,i}/g_1) \) is a DH tuple for \( i \notin I \)

- Also proves that either "\( \forall i : (g_0, g', h_{0,i}, h'_i) \) is DH tuple" or "\( \forall i : (g_0, g', h_{1,i}, h'_i) \) is DH tuple"

- \( S \) sends to \( R \) pairs \( (u_{0,i}, v_{0,i}) \leftarrow R(m_0, g_0, g', h_{0,i}, h'_i) \) and \( (u_{1}, v_{1}) \leftarrow R(m_1, g_1, g', h_{1,i}, h'_i) \)

- \( R \) finds \( m_{b,i} = v_{b,i}/u_{b,i}^r \). Similarly finds \( m_{1-b,i} \) for \( i \in I \)
Proving the conjunctions of DH tuples

- P and V know \((g, h, u_1, v_1, \ldots, u_s, v_s)\)
- P knows \(a\), such that \(h = g^a\) and \(v_i = u_i^a\)
- V chooses random weights \(\omega_1, \ldots, \omega_s \leftarrow \mathbb{Z}_q\). Sends them to P
- Both compute
  
  \[
  u = \prod_{i=1}^{s} u_i^{\omega_i} \quad \text{and} \quad v = \prod_{i=1}^{s} v_i^{\omega_i}
  \]

- \(P\) proves to \(V\) that \((g, h, u, v)\) is a DH tuple

For the disjunction on previous slide, combine with the protocol for subset of claims
Also proving \( l_1 = \cdots = l_\ell \)

- \( R \) has \( b_1, \ldots, b_\ell \in \{0, 1\} \) and \( l \subseteq \{1, \ldots, s\} \). \( S \) has \( m^{(j)}_{0,i}, m^{(j)}_{1,i} \in \mathbb{G} \) for \( i \in \{1, \ldots, s\}, j \in \{1, \ldots, \ell\} \)
- \( R \) picks \( y, \alpha_1, \ldots, \alpha_s, r_1, \ldots, r_\ell \stackrel{\$}{\leftarrow} \mathbb{Z}_q \). Computes

\[
\begin{align*}
g_1 &= g_0^y \quad h_{0,i} = g_0^{\alpha_i} \quad h_{1,i} = g_1^{\alpha_i+[i \notin l]} \\
g(j) &= g_{b_j}^{r_j} \quad h_{(j),i} = h_{b,i}^{r_j}
\end{align*}
\]

Sends them all to \( S \). Proves that \((g_0, g_1, h_{0,i}, h_{1,i}/g_1)\) is a DH tuple for \( i \notin l \)
- Also proves for each \( j \) that either “\( \forall i : (g_0, g^{(j)}_i, h_{0,i}^{(j)}, h_{i,i}^{(j)}) \) is DH tuple” or “\( \forall i : (g_0, g^{(j)}_i, h_{i,i}, h_{i,i}^{(j)}) \) is DH tuple”
- \( \ldots \) continues as previously, for all \( j \in \{1, \ldots, \ell\} \)
- This was called **Batch single-choice cut-and-choose OT**
Choosing keys for $G$’s input wires

- Circuit $C$ is garbled $s$ times. Let $G$ have $\ell$ inputs
- $G$ chooses $a_{0,1}, a_{1,1}, \ldots, a_{0,\ell}, a_{1,\ell}, r_1, \ldots, r_s \in \mathbb{Z}_q$
- $G$ lets the input keys for the $j$-th input of the $i$-th garbled circuit be $g^{a_{0,j} \cdot r_i}$ and $g^{a_{1,j} \cdot r_i}$
- $G$ sends $g^{a_{0,1}}, g^{a_{1,1}}, \ldots, g^{a_{0,\ell}}, g^{a_{1,\ell}}, g^{r_1}, \ldots, g^{r_s}$ to $E$
- This commits $G$ to all of its keys in all circuits
The secure computation protocol

- $G$ garbles $s$ copies of the circuit [previous slide]
- $G$ and $E$ do a Batch single-choice cut-and-choose OT. $E$ learns his keys for his input in evaluation circuits, and all keys for his inputs in check circuits
- $E$ tells $G$, which circuits were check circuits. Proves it by sending both $k_{0,i}^{(1)}$ and $k_{1,i}^{(1)}$
- $G$ gives to $E$ the values $r_i$ for check circuits
- $E$ opens and verifies the correctness of all check circuits
- $G$ gives to $E$ the keys corresponding to $G$’s inputs in evaluation circuits. For each input, does a disjunction of conjunctions of DH tuple proofs
- $E$ evaluates the evaluation circuits and gets the result
Making sure all evaluation circuits are good

- After the real circuit, run a smaller cheat-detection circuit
  - G’s input: his actual input
  - E’s input: two different keys for the same output bit
  - Circuit compares E’s input with the (hardcoded) values of the keys for the outputs of the main circuit
    - Outputs, if there’s a match
    - If there’s a match, then also outputs G’s input

- Carefully schedule the protocols for main circuit and cheat-detection circuit

- For main circuit, each garbled copy is selected as check or evaluation circuit with probability 1/2
  - Independently of other copies
  - Changes the batch single-choice cut-and-choose OT protocol

- G successfully cheats only if all check circuits are good and all evaluation circuits are bad. Probability: $2^{-s}$
From $k$ R-OTs to $m$ R-OTs

**Sender**

- $c_1, \ldots, c_k$
- $s_{c_1,1}, \ldots, s_{c_k,k}$

**Receiver**

- $s_{0,1}, s_{1,1}, \ldots, s_{0,k}, s_{1,k}$

- $g_{d,j} := G(s_{d,j})$

- $b_1, \ldots, b_m \leftarrow \{0, 1\}$

- $u_j := g_{0,j} \oplus g_{1,j} \oplus b_1 \parallel \cdots \parallel b_m$

- $u_1, \ldots, u_k$

- $v_j := g_{c_j,j} \oplus c_j u_j$

- $r_{0,i} = H(i; v_1[i] \parallel \cdots \parallel v_k[i])$

- $r_{1,i} = H(i; v_1[i] \parallel \cdots \parallel v_k[i] \oplus c_1 \parallel \cdots \parallel c_k)$

- $r_{b_i,i} = H(i; g_{0,1}[i] \parallel \cdots \parallel g_{0,k}[i])$

**PRG** $G : \{0, 1\}^k \rightarrow \{0, 1\}^m$
Security against malicious parties

The original R-OT-s have to have security against malicious parties

Malicious sender
Already secure, because he does not do any extra moves

Malicious receiver

- Must use the same $b_1 \| \cdots \| b_m$ each time
  - $b_1 \| \cdots \| b_m = u_j \oplus g_{0,j} \oplus g_{1,j}$ for each $j$
- After sending $u_1, \ldots, u_k$, there will be an extra consistency check
Consistency checks

\[ b_1 \| \cdots \| b_m = u_i \oplus g_{0,i} \oplus g_{1,i} = u_j \oplus g_{0,j} \oplus g_{1,j} \]

\[ u_i \oplus u_j \oplus g_{c_i,i} \oplus g_{c_j,j} = g_{1-c_i,i} \oplus g_{1-c_j,j} \]

- For each \( i, j \in \{1, \ldots, k\} \), \( R \) sends \( h_{i,j}^{0,0} = H(g_{0,i} \oplus g_{0,j}) \),
  \( h_{i,j}^{0,1} = H(g_{0,i} \oplus g_{1,j}) \), \( h_{i,j}^{1,0} = H(g_{1,i} \oplus g_{0,j}) \), \( h_{i,j}^{1,1} = H(g_{1,i} \oplus g_{1,j}) \)
- \( S \) checks that

\[ H(g_{c_i,i} \oplus g_{c_j,j}) = h_{i,j}^{c_i,c_j} \]

\[ H(g_{c_i,i} \oplus g_{c_j,j} \oplus u_i \oplus u_j) = h_{i,j}^{1-c_i,1-c_j} \]
How $R$ could cheat?

- Suppose $R$ has $(b_1 \| \cdots \| b_m)_i \neq (b_1 \| \cdots \| b_m)_j$
- $R$ must compute $h_{i,j}^{c_i,c_j}$ correctly
  - Because $S$ checks $H(g_{c_i,i} \oplus g_{c_j,j}) = h_{i,j}^{c_i,c_j}$
- Hence $R$ must guess $c_i, c_j$
- Then he can put

$$h_{i,j}^{1-c_i,1-c_j} = H(g_{1-c_i,i} \oplus g_{1-c_j,j} \oplus (b_1 \| \cdots \| b_m)_i \oplus (b_1 \| \cdots \| b_m)_j)$$

- $R$ can try to guess $\rho$ bits. He’ll succeed with probability $2^{-\rho}$
- As $R$ now has the possibility to guess bits $c_1, \ldots, c_k$, we have to increase $k$ by $\rho$