Lecture 22

2K Pok of DL and friends

\[ \begin{align*}
\text{Prover} & \quad \text{Verifier} \\
\alpha & \quad y = g^x \\
2K \text{ pok of } & \\
\alpha & \\
\end{align*} \]

Group \( G \) of order \( q \), prime, \( \langle g \rangle = G \)

\[ \text{Protocol} \]

\[ \begin{align*}
\text{Prover} & \quad \text{Verifier} \\
\gamma \leftarrow Z_q & \quad R \rightarrow c \leftarrow Z_q \\
R = g^\gamma & \\
\end{align*} \]

\[ s = r + c \alpha \]

\[ s \rightarrow g^s = R \cdot y^c \quad \text{with prob } \varepsilon = 1/\text{poly} \]

Verifier accepts.

Assume that we have access to a malicious prover \( \star \).
With access to prover \( \star \) we get 2 accepting conversations for the same \( R \):

\[ (R, c, s) \quad \text{and} \quad (R, c', s') \]

such that:

\[ \begin{align*}
g^s & = R \cdot y^c \\
g^s' & = R \cdot y^{c'} \end{align*} \]

\[ \text{divide them:} \quad g^{s-s'} = y^{c-c'} \quad \Rightarrow \quad y = g^{s-s'}/c-c' \]

Say we have a malicious verifier \( \star \), who computes:

\[ c = \text{MysteriousFunction}(R) \]

Simulator: choose \( R' \), send to verifier \( \star \) and find out \( c \).

Now, choose random \( s \leftarrow Z_q \) and solve for \( R = g^s / y^c \)

Send \( R \) to \( \star \), receive \( c \), respond with \( s \).

This can work if verifier is honest and \( c \in Z_q \) but if he is malicious then he may not give you a different \( c \) for the same \( R \).

\[ \text{In order for this to work we need the verifier to commit to "c" in advance.} \]

\[ \begin{align*}
\text{Prover} & \quad \text{Verifier} \\
\alpha & \quad y = g^x \\
h \leftarrow Z_q, h = g^\theta & \quad c \leftarrow Z_q \\
R \leftarrow & \\
(\text{commit} \circ \text{PedCommit}(c, \text{rand})) & \\
\text{rand} \leftarrow Z_q, \text{commit} = \text{Ped Commit}(c, \text{rand}) & \\
c, \text{rand} & \\
\end{align*} \]

\[ \text{Pederssen commitment} \]

\( PK = (g, h) \in G^2 \) of order \( q \)

\[ \text{Commit}(m, \text{rand}) = g^m \cdot h^{\text{rand}} \]

\[ \text{l} \leftarrow Z_q \]

- Unconditionally hiding: \( \text{Commit}(m, \text{rand}) = G \)
- Computationally binding: \( \text{if} \quad g^m \cdot \text{rand} = g^m' \cdot \text{rand}' \quad \Rightarrow \quad g^l = h^{m-m'} \)
ZK Simulator:
• choose $b$ at random, send $h = g^b$ to $V^*$
• receive commit from $V^*$, choose $R' \in C$, send to $V^*$
• receive $c'$ and rand' from $V^*$, pick $s \in \mathbb{Z}_q$, let $R = g^s / y^c$, reset $V^*$ sending it $R$; receive $(c', \text{rand})$ again. (If not try again with different random $s$). If $(c, \text{rand}) \neq (c', \text{rand'})$, fail.
Else, send $s, h$.

Knowledge extractor:
• Run the prover until accepting a conversation.
  Get $h = g^b$, $b$
  $s, R, s + g^s = R \cdot y^c$
  commit = $g^c \cdot \text{rand} \cdot c, \text{rand}$
• Pick random $c'$, solve for rand' $s+$
  $c + b \cdot \text{rand} = c' + b \cdot \text{rand'}$ (so, commit = $g^c \cdot \text{rand'})$
• Reset $P^*$ to 4th round, and send $(c', \text{rand'})$ instead.
• Receive $s', b$ s t $g^s = R \cdot y^{c'}$
  $g^{s-s'/c-c'} = y$

Say that on the OT protocol, we have a Simulator in the place of the recipient.

Simulator $m_0 \xrightarrow{m_0} \text{TTP} \xrightarrow{b} \text{Recipient}$
$\rightarrow$Constructing the second ZKPoK for OT.

Prover
$\begin{array}{c}
V_0, y, a, b \\
\text{or} \\
V_0 = g^x
\end{array}$

Verifier
$\begin{array}{c}
V_0, y
\end{array}$

ZK PoK of $\alpha, b \in \mathbb{Z}^*_3$
\textbf{Prover}

\[ r \in \mathbb{Z}_q, \quad s_b \in \mathbb{Z}_q, \quad c_b \in \mathbb{Z}_q \]

\[ R_b = g^r \]

\[ R_b, R_0 \rightarrow \]

\[ c \]

\[ C_b = c - c_b \]

\[ s_b = r + \alpha c_b \]

\[ c_0, c_1, s_0, s_1 \rightarrow \]

\[ c_0 + c_1 = c \]

\[ R_0 = g^{s_0} c_0 \]

\[ R_1 = g^s c_1 \]

\[ R_i = g^y i \]

\textbf{Verifier}

\[ R_0, R_1 \rightarrow \]

\[ c \]

Correctness is easy by inspection!