Lecture 22
Outline

OR Composition, continued

Pairing-based cryptography
OR Composition, Continued
Encrypting bits

**Prover**

NP language = \{ (g,h,v,w) : \exists (b,c) \text{ such that } b \in \{0,1\} \text{ and } v = g^c, w = h^c.g^b \}

\[
\begin{align*}
g, h, v, w &= g^a, g^b, g^{ab}.g^d \\
&\text{or} \\
g, h, v, w &= g^a, g^b, g^{ab}.g^d
\end{align*}
\]

**Verifier**

\[
\begin{align*}
(g, h, v, w) \text{ is a DDH tuple: Chaum-Pedersen} \\
(g, h, v, w) \text{ is a DDH tuple}
\end{align*}
\]
OR Composition

Prover

Veriﬁer

Goal: Prove that \( w \) such that \((x_0, w) \) in \( R_0 \) or \((x_1, w) \) in \( R_1 \)

Suppose we have a protocol \((P_0, V_0)\) for \( R_0 \), and a protocol \((P_1, V_1)\) for \( R_1 \)

Can we combine them to obtain a protocol for \( R_0 \) OR \( R_1 \)?

What about letting the prover simulate exactly one of them?
OR Composition

Prover

Goal: Prove that \( w \) such that \( (x_0, w) \) in \( R_0 \) or \( (x_1, w) \) in \( R_1 \)

Verifier

1. Prover first sends \( \exists w \ (x_0, w) \in R_0 \)

2. Prover sends \( \exists w \ (x_1, w) \in R_1 \)

Verifier checks \( |ch_1| = |ch_0| = |ch_1| \)

Check: \( ch_0 \oplus ch_1 = ch \)

Complete protocol
OR Composition

Prover

Goal: Prove that w such that \((x_0,w)\) in \(R_0\) or \((x_1,w)\) in \(R_1\)

Verifier

\begin{align*}
\text{Before rewinding,} & \quad \text{FIRST MESSAGE} \quad \rightarrow \\
\text{RESEND} & \quad \text{and send} \quad \leftarrow \\
\end{align*}

\(\begin{align*}
\text{CH} & \quad \rightarrow \\
\text{CH}_0 & \quad \rightarrow \\
\text{CH}_1 & \quad \rightarrow \\
\text{CH}_0' & \quad \rightarrow \\
\text{CH}_1' & \quad \rightarrow \\
\end{align*}\)

\(\text{either CH}_0' \neq \text{CH}_0 \text{ or CH}_1' = \text{CH}_1.\)
OR Composition: Zero-Knowledge

Prover

Goal: Prove that \( w \) such that \((x_0, w) \) in \( R_0 \) or \((x_1, w) \) in \( R_1 \)

Guess \( ch \) in advance

\[ \Rightarrow \text{Sample random } ch_0, ch_1, \text{ s.t. } ch_0 + ch_1 = ch. \]

Simulate \( \Pi_1 \) given \( ch_0 \)

Verifier

Simulate \( \Pi_2 \) given \( ch_1 \)
OR Composition

**Prover**

Goal: Prove that \( w \) such that \((x_0, w) \) in \( R_0 \) or \((x_1, w) \) in \( R_1 \)

\( g, h, v, w \equiv DDH \)

\( R, T \)

\( \overline{P} \)

\( \overline{P'} \)

**Verifier**

\( g, h, v, w \equiv DDH \)

\( \overline{R}, \overline{T} \)

\( \overline{P} \)

\( \overline{P'} \)

FALSE
Application: Encrypting bits

Prover

NP language = \{(g,h,v,w) : \exists (b,c) \text{ such that } b \in \{0,1\} \text{ and } v = g^c, w = h^c.g^b\}
Non-Interactive Zero-Knowledge

Prover

Verifier

"commit" $\rightarrow$

\begin{align*}
\text{"challenge"} & \leftarrow [n \text{ bit}] \\
\text{"response"} & \rightarrow
\end{align*}
Non-Interactive Zero-Knowledge: Fiat-Shamir

Soundness.

Prover

Verifier

"commit" -> RO

"challenge" = H("commit")

"response"
Non-Interactive Zero-Knowledge: Fiat-Shamir

**Prover**

Sim fixes challenge \( r \), first, randomly.

Then computes "commit" message:

\[
\begin{align*}
R, T \\
H(R, T) \\
S = r + p \cdot b
\end{align*}
\]

**Verifier**
Pairings
Pairing-based cryptography

- So far, we’ve looked at hard problems like discrete log, CDH, HDH, DDH in groups
- Certain groups have an additional structure
- Let $G_0, G_1, G_T$ be 3 cyclic groups of prime order
  where $g_0 \in G_0$ and $g_1 \in G_1$ are generators
- A pairing is an efficiently computable function $e: G_0 \times G_1 \rightarrow G_T$ such that:
  1. $g_T = e(g_0, g_1)$ is a generator of $G_T$
  2. For all $(u, u') \in G_0$ and $(v, v') \in G_1,$
     $e(u.u', v) = e(u,v).e(u',v)$ and $e(u, v.v') = e(u,v).e(u,v')$
Pairing-based cryptography

- A pairing is an efficiently computable function $e: G_0 \times G_1 \rightarrow G_T$ such that:

  1. $g_T = e(g_0, g_1)$ is a generator of $G_T$

  2. For all $(u, u') \in G_0$ and $(v, v') \in G_1$,

        $e(u, u', v) = e(u, v).e(u', v)$ and $e(u, v, v') = e(u, v).e(u, v')$

- Consequences: $e(g_0^a, g_1^b) = $
Pairing-based cryptography

• A pairing is an efficiently computable function $e: G_0 \times G_1 \rightarrow G_T$ such that:

  1. $g_T = e(g_0, g_1)$ is a generator of $G_T$

  2. For all $(u, u') \in G_0$ and $(v, v') \in G_1$,

     
     $e (u.u' , v) = e(u,v).e(u',v)$ and $e (u , v.v' ) = e(u,v).e(u,v')$

• Consequences:  when $G_0 = G_1$, 