Today’s topics:
- OR Proofs
- Trapdoor Commits
- Applications

1 OR Proofs

An OR protocol is a protocol in which you only need a witness for one of two protocols to be validated.

**Assumption 1** \( \Sigma_1 \) and \( \Sigma_2 \) are two \( \Sigma \)-protocols for relations \( R_1 \) and \( R_2 \).

**Definition 2** Define \( R_{OR}(x, (y_1, y_2)) = 1 \iff R_1(x, y_1) = 1 \) or \( R_2(x, y_2) = 1 \)

This defined the language \( L_{OR} = \{(y_1, y_2)|\exists b \in \{0,1\} \land R_b(x, y_b) = 1\} \)

1.1 \( \Sigma_{OR} \) for \( R_{OR} \)

We now give the \( \Sigma \)-protocol for the OR relation \( R_{OR} \). Let \( \bar{b} = 3 - b \).

<table>
<thead>
<tr>
<th>P(( x_b ) for ( b \in (1, 2) ))</th>
<th>V(( y_1, y_2 ))</th>
</tr>
</thead>
<tbody>
<tr>
<td>( a_b \leftarrow \text{honest via } x_b )</td>
<td>( a_1, a_2 )</td>
</tr>
<tr>
<td>( (a_b, c_b, z_b) \leftarrow \text{sim}_b(y_b) )</td>
<td>( \text{Sample } c \leftarrow G )</td>
</tr>
</tbody>
</table>

Set \( c_b = c - c_b \)

Honestly generate \( z_b \) for \( (a_b, c_b) \) \( \leftarrow \text{sim}_b(c_b) \)

1.2 Proof this is a \( \Sigma \)-protocol

- Special HVZK : Define the simulator \( \text{sim}_{OR}(c) \) as follows.
  - \( c_1 \leftarrow G \)
  - \( c_2 = c - c_1 \)
  - \( (a_1, z_1) \leftarrow \text{sim}_1(c_1) \)
\begin{itemize}
  \item \((a_2, z_2) \leftarrow \text{sim}_2(c_2)\)
\end{itemize}

It is clear that the transcript \((a = (a_1, a_2), z = (c_1, c_2, z_1, z_2))\) is distributed identically to the real protocol.

- **Special Soundness**
  - We are given accepting \(((a_1, a_2), c, (c_1, c_2, z_1, z_2), c', (c'_1, c'_2, z'_1, z'_2))\), where \(c \neq c'\).
  - \((c_1 + c_2 = c) \neq (c'_1 + c'_2 = c)\) implies either \(c_1 \neq c'_1\), or \(c_2 \neq c'_2\), or both.
  - If \(c_\alpha \neq c'_\alpha\), then use Special Soundness of \(\Sigma_\alpha\) to extract the witness for \(R_\alpha\), which is also the witness for \(R_{OR}\).

As a corollary, \(\Sigma_{OR}\) is WI since it is a \(\Sigma\)-protocols.

\begin{tabular}{|l|
\hline
Remark 1: Makes sense even if \(y_b \notin R_b\).  \\
Remark 2: You can generalize to \(R_1 \lor R_2 \lor R_3 \lor \ldots \lor R_t\). The complexity is \(\approx t\).  \\
Remark 3: \(\Sigma_{AND}\) is also easy (reuse same \(c\) for both relations).  \\
Thus, can compute a \(\Sigma\)-protocol for any “monotone function”.  \\
\hline
\end{tabular}

2 Application to Active ID Protocols

**Assumption 3** \(f: 1-1\ OWF\ having\ a\ \Sigma\text{-protocol. }\ (\ ex: \ \text{Schnorr: } f(x) = g^x; \ \text{GQ: } f(x) = x^e mod n)\).

From the properties of \(\Sigma\)-protocols (see last lecture) we have passive security. (HW: Active security?)

**Idea 1** Take two copies of \(\Sigma\), \(\Sigma_1\), \(\Sigma_2\) and run the \(OR\) proof using them.

- Select \(x_1, x_2 \leftarrow \$
- \(y_1 = f(x_1)\)
- \(y_2 = f(x_2)\)
- \(Sk = x_b\ Pk = (y_1, y_2)\) and choose a \(\Sigma_{OR}\).

\begin{tabular}{|l|
\hline
Notice: \(\exists\) two possible witnesses and WI implies \(V^*\) doesn’t know which one is used for \(\forall\ Pk(y_1, y_2)\). By last lecture, this implies that \(\Sigma_{OR}\) is actively secure.  \\
Note: This is not LR, there is only 1 bit of information.  \\
Exercise: Compare Okamoto.  \\
\hline
\end{tabular}
2.1 Private Sharing of a resource

Alice and Bob want to share a resource $R$ (printer, etc.) but they do not trust each other (they do not want to share a password.) They would also like to make it impossible to distinguish which of them has accessed the resource. Based on the properties of OR-protocols, Alice and Bob can each have their own password and still have anonymous access to the resource.

- Alice($P_{kA}, S_{kA}$) $\leftarrow \Sigma_A$
- Bob($P_{kB}, S_{kB}$) $\leftarrow \Sigma_B$
- To gain access to the resource, run $\Sigma_{A\lor B}$.

2.2 Off-the-Record Designated Proofs

Alice has a weakness $x$ for $y$ in language $C$. Alice wants to show Bob that she knows this weakness without giving the weakness away, i.e. not reveal the weakness to Eve.

- Option 1: Run $\Sigma$-protocol for $C$.
  - Only secure if you are talking to Bob who is an HVZK
  - Alice might be talking to Eve.
- Option 2: If Bob has ($P_{kB}, S_{kB}$) & $\exists \Sigma_B$ Alice runs $\Sigma_{OR}$ of $\Sigma$ and $\Sigma_B$. Where $\Sigma$ is a $\Sigma$-protocol on $C$.

The idea of option 2 is that Alice can convince Bob that she knows the weakness, because otherwise it implies that she knows Bob’s secret key. It also becomes impossible for Bob to convince anyone else - he would have to convince them that she DOESN’T know his secret key.

Special Case: Off-the-Record (OTR) Conversation

Alice($S_{kA}, P_{kA}$) for a signature scheme $S$ and $S$ has a $\Sigma$-protocol $\Sigma_A$ such that $L_m = \"I know Signature(m)\"$ for $\forall m$. Apply the same reasoning as OTR Designated Proofs to this, she can show Bob she sent the message, but no one else will believe him claiming she sent it.

3 Trapdoor Commitments

A Trapdoor Commitment is a special "commitment" scheme in which, from an observer’s point of view, appears to be a regular commitment scheme, but with a special Trapdoor Key ($T_k$) you can "Equivocate": change the message which you have "committed" yourself to.
Commitment Scheme Refresher

**Definition 4**  Commitment Scheme is a triple of algorithms (Setup, Com, Open )

- **Setup** \( (1^\lambda) \rightarrow Ck \) (Defines message space \( \mu \))
- **Com** \( Ck(c) \rightarrow (a, d) \) where \( a \) is the "commitment" and \( d \) is the "decommitment" (key)
- **Open** \( Ck(a, d) \rightarrow \bar{c} \in \mu \lor \{ \bot \} \)

**Hiding:** \( \forall c_0, c_1 \in \mu \rightarrow a(c_0) \approx a(c_1) \)

**Perfect Hiding:** \( a(c_0) \equiv a(c_1) \)

**Binding:** For a random \( Ck \) it is "hard to find" (\( a,d,d' \)) such that

- \( \text{Open}_{Ck}(a, d) = c \)
- \( \text{Open}_{Ck}(a, d') = c' \)
- \( c, c' \neq \bot \land c \neq c' \)

**3.1 Construction**

Let \( R(x, y) \) be any hard relation having a \( \Sigma \)-protocol.

- **TSetup** \( (1^\lambda) \rightarrow (Ck, Tk) \) replaces **Setup** \( (1^\lambda) \).
- **Hiding:** \( \forall c_0, c_1 (a(c_0), Tk) \equiv (a(c_1), Tk) \)
- **Equivocability:** \( \exists \) two algorithms FCom, Equiv:
  - \( \text{FCom}_{Ck}(Tk) \rightarrow (\tilde{a}, \text{state}) \)
  - \( \text{Equiv}_{Ck}(\tilde{a}, c, \text{state}, Tk) \rightarrow \tilde{d}_c \) such that \( \text{Open}(\tilde{a}, \tilde{d}) = c \)

- Even with the Tk, this is still strong:
  - \( \forall c | \text{Com}(c) \rightarrow (a, d) \equiv \text{fake}(\tilde{a}, \tilde{d}) \)
• decommitment ← (c, z)
• Open((a, (c, z))) → valid if (a, c, z) is valid.
• Strong Hiding: ∀c a(c) ≡ Prover’s a but Prover’s a is independent of c.
• Binding: Exactly Special Soundness : (a, (c, z)), (c’, z’) → hard
• FComCk(x) → Generates an honest a and honest state. (Oblivious: True whenever first flow is independent of x)
• Equiv(a, c, state, x) → honest z → You can create any $d_c = (c, z)$

This produces a valid commitment scheme:

$Ck = y$ This simply defines the message space. We just have to select a $TSetup$ who’s range is the desired message space. This produces the same message space as $Setup$.

$ComCk(c)$: $(a, z) ← sim^*(c)$ For a give challenge $c$, $sim^*$ will produce an $(a, z)$ pair which fall into the $(a, c, z)$ triple of a $\Sigma$-protocol. By Special HVZK of $\Sigma$-protocols, this is indistinguishable from a real $(a, z)$ pair.

$FComCk(x)$ Is actually the honest generation of an $(a, z)$ pair for the $\Sigma$-protocol, which is indistinguishable from the last item.

$Equiv(a, c, state, x) →$ honest $z$ Because this is actually a $\Sigma$-protocol run as intended (first send $a$, then receive $c$, then generate a $z$) This will produce a valid $d ← (c, z)$ That is again, indistinguishable from the $(c, z)$ pair produced by $ComCk(c)$.

### 3.1.1 Example: Schnorr’s

Note: GQ as an exercise.

![Schnorr's Commitment](image)

- $Com(z) ← Z_q \mid a = y^c \ast g^z$ (Redersen’s commitment: $(y, g)$ are random generators)
- $FCom(g, y) \mid r ← Z_q \mid a = g^r \forall r$
- $Equiv(a, c, r, x) \mid z = r - c \ast x$
3.2 Applications

3.2.1 Offline-Online Signatures

Notice that $\text{Equiv}$ is often faster than $\text{Com}/\text{FCom}$ or most other cryptographic operations. How do we take advantage of this? Split signing into two sections:

<table>
<thead>
<tr>
<th>Offline</th>
<th>Online</th>
</tr>
</thead>
<tbody>
<tr>
<td>Does not have message $m$ - does most of the work.</td>
<td>$m$ comes - little work</td>
</tr>
<tr>
<td>$S = (\text{Gen}, \text{Sig}, \text{Ver})$</td>
<td>$\langle \text{Sig}(a), d \rangle$</td>
</tr>
<tr>
<td>TD Commitment: $S' = (\text{Gen}', \text{Sig}', \text{Ver}')$</td>
<td>$\text{Sig}(a)$ is generated offline, $a$ may be fake.</td>
</tr>
<tr>
<td>$\text{Gen}'(1^\lambda)</td>
<td>(Sk, V k) \leftarrow \text{Gen}(1^\lambda)$</td>
</tr>
<tr>
<td>$(Ck, Tk) \leftarrow \text{TSetup}(1^\lambda)$</td>
<td></td>
</tr>
<tr>
<td>$Sk' = (Sk, Tk)$ $Vk' = (V k, C k)$</td>
<td></td>
</tr>
<tr>
<td>$\text{Sig}(m) : (\bar{a}, \text{state}), \text{FCom}(Tk)$</td>
<td></td>
</tr>
<tr>
<td>$\sigma = \text{Sig}_{Sk}(\bar{a})$</td>
<td></td>
</tr>
<tr>
<td>$d_m \leftarrow \text{Equiv}(\bar{a}, m, Tk, \text{state})$</td>
<td></td>
</tr>
<tr>
<td>Output($\sigma, \bar{a}, d_m$)</td>
<td></td>
</tr>
</tbody>
</table>

What we are doing here is not signing the message itself, as this is a computationally expensive operation that can only be done when you have the message. Instead what we are doing is signing a random commitment (one to which we have a Trapdoor Key,) which can be done offline before we have any messages, and then once a message arrives we are equivocating a decommitment that will open the signed commitment into the message we wish to sign.

**Lemma 5** If $T = \text{Trapdoor Commitment}$ and $S = \text{UF-KMA}$ (Existentially Unforgeable under Known Message Attack) $\rightarrow S' = \text{UF-CMA}$ (Existentially Unforgeable under Chosen Message Attack)

3.2.2 Off-the-Record Communication

<table>
<thead>
<tr>
<th>Alice(Sk, V k)</th>
<th>Bob(Ck, Tk)</th>
</tr>
</thead>
<tbody>
<tr>
<td>$(a, d) \leftarrow \text{Com}_{Ck_B}(m)$</td>
<td>$\sigma \leftarrow \text{Sig}_{Sk}(a)$ $\xrightarrow{(a,d,\sigma)}$ Bob verifies.</td>
</tr>
</tbody>
</table>

Bob believes Alice sent $m$. (Otherwise his $Tk$ is known.) No one will believe Bob saying Alice sent $m$. (He can Equivocate $m$ into anything he wants: $d' \leftarrow \text{Bob}(a, m', d, Tk)$)

Reverse State Reconstruction

Given any valid $(a, m, d)$ and $Tk \rightarrow$ produce state such that $(a, \text{state}) \equiv F\text{Comm}(Tk)$. i.e. for Schnorr, $\text{state} = r$ and $r$ can be reconstructed: $r = z + c \ast x$. 

L4-6