UP TO NOW
UP TO NOW

- Introduction to the field
UP TO NOW

- Introduction to the field
- Secure computation protocols
UP TO NOW

- Introduction to the field
- Secure computation protocols
- Introduction to malicious model
UP TO NOW

- Introduction to the field
- Secure computation protocols
- Introduction to malicious model
- Σ-protocols
THIS TIME
Σ-protocols: short reminder
THIS TIME

- Σ-protocols: short reminder
- Continuing with OR rule
THIS TIME

- Σ-protocols: short reminder
- Continuing with OR rule
- Σ-protocol for Boolean circuits / NP
THIS TIME

- Σ-protocols: short reminder
- Continuing with OR rule
- Σ-protocol for Boolean circuits / NP
- Constructing interactive zero knowledge protocols from Σ-protocols
THIS TIME

- $\Sigma$-protocols: short reminder
- Continuing with OR rule
- $\Sigma$-protocol for Boolean circuits / $\mathbf{NP}$
- Constructing interactive zero knowledge protocols from $\Sigma$-protocols
  - only beginning of this...
THIS TIME

❖ Σ-protocols: short reminder
❖ Continuing with OR rule
❖ Σ-protocol for Boolean circuits / NP
❖ Constructing interactive zero knowledge protocols from Σ-protocols
   ❖ only beginning of this...
❖ Useful tool: commitment schemes
THIS TIME

- Σ-protocols: short reminder
- Continuing with OR rule
- Σ-protocol for Boolean circuits / NP
- Constructing interactive zero knowledge protocols from Σ-protocols
  - only beginning of this...
- Useful tool: commitment schemes
- Σ-protocol = trapdoor commitment scheme (kind of)
REMINDER: Σ-PROTOCOLS
REMINDER: $\Sigma$-PROTOCOLS
REMINDER: $\Sigma$-PROTOCOLS

1st message: commitment $a$
REMINDER: Σ-PROTOCOLS

1st message: commitment $a$

2nd message: challenge $c$

$x, \omega$
REMINDER: $\Sigma$-PROTOCOLS

1st message: commitment $a$

2nd message: challenge $c$

3rd message: response $z$
REMINDER: Σ-PROTOCOLS

1st message: commitment $a$

2nd message: challenge $c$

3rd message: response $z$

Accepts iff prover knows $\omega$ such that $(x, \omega) \in R$
REMINDER: $\Sigma$-PROTOCOLS

- 1st message: commitment $a$
- 2nd message: challenge $c$
- 3rd message: response $z$

**Requirement:** $c$ is chosen from publicly known challenge set $C$ randomly. (Does not depend on $a$!)

**Terminology:** public coin protocol

Accepts iff prover knows $\omega$ such that $(x, \omega) \in R$
REMINDER: Σ-PROTOCOLS

1. Completeness
2. Special Soundness
3. Special Honest-Verifier ZK
Assume wlog that $P$ knows $\omega$ s.t. $R_1(x, \omega)$

Goal: construct protocol $(P, V)$ for $PK(\omega: R_1(x, \omega) \lor R_2(x, \omega))$
OR-PROOF

Assume wlog that $P$ knows $\omega$ s.t. $R_1(x, \omega)$

$(x, \omega) \vdash R_1(x, \omega)$

$\begin{align*}
a_1 &\leftarrow P_1(x; r) \\
c_2 &\leftarrow C \\
(a_2, z_2) &\leftarrow S_2(c_2)
\end{align*}$

Goal: construct protocol $(P, V)$ for PK $(\omega: R_1(x, \omega) \lor R_2(x, \omega))$
OR-PROOF

Assume wlog that $P$ knows $\omega$ s.t. $R_1(x, \omega)$

Goal: construct protocol $(P, V)$ for PK $(\omega: R_1(x, \omega) \lor R_2(x, \omega))$
Assume wlog that $P$ knows $\omega$ s.t. $R_1(x, \omega)$

$(x, \omega): R_1(x, \omega)$

$\begin{align*}
a_1 &\leftarrow P_1(x; r) \\
c_2 &\leftarrow C \\
(a_2, z_2) &\leftarrow S_2(c_2)
\end{align*}$

$c \leftarrow C = \{0, \ldots, |C| - 1\}$

$(a_1, a_2)$

$x$

Goal: construct protocol $(P, V)$ for PK $(\omega: R_1(x, \omega) \lor R_2(x, \omega))$
OR-PROC

\( (x, \omega): R_1(x, \omega) \)

Let \( \omega \) s.t. \( R_1(x, \omega) \)

\[ a_1 \leftarrow P_1(x; r) \]
\[ c_2 \leftarrow C \]
\[ (a_2, z_2) \leftarrow S_2(c_2) \]

\[ c_1 \leftarrow c - c_2 \mod |C| \]
\[ z_1 \leftarrow P_1(x, \omega, c_1; r) \]

Assume wlog that \( P \) knows \( \omega \) s.t. \( R_1(x, \omega) \)

\( x \)

Goal: construct protocol \((P, V)\) for PK \((\omega: R_1(x, \omega) \lor R_2(x, \omega))\)
OR-PROOF

\((x, \omega): R_1(x, \omega)\)

\(a_1 \leftarrow P_1(x; r)\)

\(c_2 \leftarrow C\)

\((a_2, z_2) \leftarrow S_2(c_2)\)

\(c \leftarrow C = \{0, ..., |C| - 1\}\)

\(c_1 \leftarrow c - c_2 \mod |C|\)

\(z_1 \leftarrow P_1(x, \omega, c_1; r)\)

\((c_1, z_1, z_2)\)

**Goal:** construct protocol \((P, V)\) for PK \((\omega: R_1(x, \omega) \lor R_2(x, \omega))\)

Assume wlog that \(P\) knows \(\omega\) s.t. \(R_1(x, \omega)\)
(x, ω): $R_1(x, ω)$

$\begin{align*}
a_1 &\leftarrow P_1(x; r) \\
c_2 &\leftarrow C \\
(a_2, z_2) &\leftarrow S_2(c_2)
\end{align*}$

$c \leftarrow C = \{0, \ldots, |C| - 1\}$

$c_1 \leftarrow c - c_2 \mod |C|$

$z_1 \leftarrow P_1(x, ω, c_1; r)$

$c_2 \leftarrow c - c_1 \mod |C|$

Accept if $c_1 < |C|$ and both $V_1(x, a_1, c_1, z_1)$ and $V_2(x, a_2, c_2, z_2)$ accept

Goal: construct protocol $(P, V)$ for PK $(ω: R_1(x, ω) \lor R_2(x, ω))$
SECURITY PROOF

✧ I will not give a full security proof, but it is simple
✧ **Completeness:** from completeness of first PK, and successful simulation of the second one
✧ **Special soundness:** OR-extractor runs extractors for both branches. **One** of them is successful, return this value
✧ **SHVZK:** since the first PK is SHVZK, and the second one is already simulated
POK: ELGAMAL PLAINTEXT IS BOOLEAN

\[ L = \{ (g, h, e_1, e_2), \text{s.t. } (e_1, e_2) = (g^\mu h^\rho, g^\rho) \text{ for some } \rho \in \mathbb{Z}_p, \mu \in \{0, 1\} \} \]

We depict prover when \( \mu = 0 \); \( \mu = 1 \) is dual.
POK: ELGAMAL Plaintext is Boolean

\[ L = \{(g, h, e_1, e_2), \text{s.t. } (e_1, e_2) = (g^\mu h^\rho, g^\rho) \text{ for some } \rho \in \mathbb{Z}_p, \mu \in \{0, 1\}\} \]

1. \( r \leftarrow \mathbb{Z}_p \)
2. \( (a_{11}, a_{12}) \leftarrow (h, g)^r \) // Real branch
3. \( c_2 \leftarrow C; z_2 \leftarrow \mathbb{Z}_p \) // Simulated branch
4. \( (a_{21}, a_{22}) \leftarrow (h, g)^{z_2} / (e_1 / g^1, e_2)^{c_2} \)

We depict prover when \( \mu = 0 \); \( \mu = 1 \) is dual

\((g, h, e_1, e_2), (\mu, \rho)\)
POK: ELGAMAL PLAINTEXT IS BOOLEAN

\[ L = \{ (g, h, e_1, e_2), \text{s.t.} (e_1, e_2) = (g^\mu h^\rho, g^\rho) \text{ for some } \rho \in \mathbb{Z}_p, \mu \in \{0, 1\} \} \]

1. \( r \leftarrow \mathbb{Z}_p \)
2. \((a_{11}, a_{12}) \leftarrow (h, g)^r \) // Real branch
3. \( c_2 \leftarrow C; z_2 \leftarrow \mathbb{Z}_p \) // Simulated branch
4. \((a_{21}, a_{22}) \leftarrow (h, g)^{z_2} / (e_1 / g^t, e_2)^{c_2} \)

We depict prover when \( \mu = 0; \mu = 1 \) is dual
POK: ELGAMAL PLAINTEXT IS BOOLEAN

\[ L = \{(g, h, e_1, e_2), \text{s.t. } (e_1, e_2) = (g^\mu h^q, g^q) \text{ for some } q \in \mathbb{Z}_p, \mu \in \{0, 1\}\} \]

1. \( r \leftarrow \mathbb{Z}_p \)
2. \( (a_{11}, a_{12}) \leftarrow (h, g)^r \text{ // Real branch} \)
3. \( c_2 \leftarrow C; z_2 \leftarrow \mathbb{Z}_p \text{ // Simulated branch} \)
4. \( (a_{21}, a_{22}) \leftarrow (h, g)^{z_2} / (e_1 / g^1, e_2)^{c_2} \)

We depict prover when \( \mu = 0 \); \( \mu = 1 \) is dual

\[ (g, h, e_1, e_2) \]

\[ (a_{11}, a_{12}, a_{21}, a_{22}) \]

\[ c \leftarrow C \]
POK: ELGAMAL PLAINTEXT IS BOOLEAN

\[ L = \{(g, h, e_1, e_2), \text{s.t. } (e_1, e_2) = (g^\mu h^\rho, g^\rho) \text{ for some } \rho, \mu \in \mathbb{Z}_p, \mu \in \{0, 1\}\} \]

1. \( r \leftarrow \mathbb{Z}_p \)
2. \( (a_{11}, a_{12}) \leftarrow (h, g)^r \) // Real branch
3. \( c_2 \leftarrow C; z_2 \leftarrow \mathbb{Z}_p \) // Simulated branch
4. \( (a_{21}, a_{22}) \leftarrow (h, g)^{z_2}/(e_1/g^1, e_2)^{c_2} \)

\( (g, h, e_1, e_2), (\mu, \rho) \rightarrow (a_{11}, a_{12}, a_{21}, a_{22}) \)

\( c \leftarrow C \)

\( c_1 \leftarrow c - c_2 \text{ mod } |C| \)

\( z_1 \leftarrow c_1 \rho + r_1 \)

We depict prover when \( \mu = 0; \mu = 1 \) is dual
POK: ELGamal Plaintext is Boolean

$L = \{(g, h, e_1, e_2), \text{s.t. } (e_1, e_2) = (g^\mu h^q, g^q) \text{ for some } q \in \mathbb{Z}_p, \mu \in \{0, 1\}\}$

1. $r \leftarrow \mathbb{Z}_p$
2. $(a_{11}, a_{12}) \leftarrow (h, g)^r$ // Real branch
3. $c_2 \leftarrow C; z_2 \leftarrow \mathbb{Z}_p$ // Simulated branch
4. $(a_{21}, a_{22}) \leftarrow (h, g)^{z_2} / (e_1 / g^1, e_2)^{c_2}$

$(g, h, e_1, e_2)$

$(a_{11}, a_{12}, a_{21}, a_{22})$

$c \leftarrow C$

$c_1 \leftarrow c - c_2 \mod |C|$

$z_1 \leftarrow c_1 \cdot q + r_1$

$(c_1, z)$

We depict prover when $\mu = 0$; $\mu = 1$ is dual.
POK: ELGAMAL PLAINTEXT IS BOOLEAN

\[ L = \{(g, h, e_1, e_2), \text{s.t. } (e_1, e_2) = (g^\mu h^q, g^q) \text{ for some } q \in \mathbb{Z}_p, \mu \in \{0, 1\}\} \]

1. \( r \leftarrow \mathbb{Z}_p \)
2. \((a_{11}, a_{12}) \leftarrow (h, g)^r \) // Real branch
3. \( c_2 \leftarrow C; z_2 \leftarrow \mathbb{Z}_p \) // Simulated branch
4. \((a_{21}, a_{22}) \leftarrow (h, g)^{z_2} / (e_1 / g^1, e_2)^{c_2} \)

We depict prover when \( \mu = 0 \); \( \mu = 1 \) is dual.

\( (g, h, e_1, e_2) \)

\( (a_{11}, a_{12}, a_{21}, a_{22}) \)

\( c \leftarrow C \)

\( c_1 \leftarrow c - c_2 \mod |C| \)

\( z_1 \leftarrow c_1 q + r_1 \)

\( (c_1, z) \)

\( c_2 \leftarrow c - c_1 \mod |C| \)

Accept if \( c_1 \in C, \)
\[ (h, g)^{z_1} = (a_{11}, a_{12})(e_1, e_2)^{c_1}, \]
\[ (h, g)^{z_2} = (a_{21}, a_{22})(e_1/g, e_2)^{c_2} \]
BETTER WITH ADDITIVE/VECTOR NOTATION

\[ L = \{ (g \in \mathbb{G}^2, \ e \in \mathbb{G}^2) : \exists (\mu \in \{0,1\}, \ \varrho), \ e = \Enc_g(\mu; \ \varrho) = \mu (g_0) + \varrho g \} \]

1. \( r_1 \leftarrow \mathbb{Z}_p \)
2. \( a_1 \leftarrow r_1 g \)
3. \( c_2 \leftarrow C; \ z_2 \leftarrow \mathbb{Z}_p \)
4. \( a_2 \leftarrow z_2 g - c_2 (e - (g_0)) \)

We depict prover when \( \mu = 0; \mu = 1 \) is dual

\( (g, e) \)

\( (a_1, a_2) \)

\( c \leftarrow C \)

\( c_1 \leftarrow c - c_2 \mod |C| \)

\( z_1 \leftarrow c_1 \varrho + r_1 \)

Accept if \( c_1 \in C, \)

\( z_1 g = c_1 e + a_1 \)

\( z_2 g = c_2 (e - (g_0)) + a_2 \)

Completeness: obvious
$z_1 g = c_1 e + a_1$

$z_1^* g = c_1^* e + a_1$

$\Rightarrow (z_1^* - z_1) g = (c_1^* - c_1) e$

$\Rightarrow (z_1^* - z_1)/(c_1^* - c_1) = e$

$\Rightarrow q = (z_1^* - z_1)/(c_1^* - c_1)$

$z_2 g = c_2 (e - \langle g \rangle_0) + a_2$

$z_2^* g = c_2^* (e - \langle g \rangle_0) + a_2$

$\Rightarrow (z_2^* - z_2) g = (c_2^* - c_2) (e - \langle g \rangle_0)$

$\Rightarrow (z_2^* - z_2)/(c_2^* - c_2) = e - \langle g \rangle_0$

$\Rightarrow q = (z_2^* - z_2)/(c_2^* - c_2)$
SECURITY PROOF

**Special Soundness:**

\[ z_1 g = c_1 e + a_1 \]
\[ z_1^* g = c_1^* e + a_1 \]

\[ \Rightarrow (z_1^* - z_1) g = (c_1^* - c_1) e \]

\[ \Rightarrow ((z_1^* - z_1) / (c_1^* - c_1)) g = e \]

\[ \Rightarrow \varrho = (z_1^* - z_1) / (c_1^* - c_1) \]

\[ z_2 g = c_2 (e - (g_0)) + a_2 \]
\[ z_2^* g = c_2^* (e - (g_0)) + a_2 \]

\[ \Rightarrow (z_2^* - z_2) g = (c_2^* - c_2) (e - (g_0)) \]

\[ \Rightarrow ((z_2^* - z_2) / (c_2^* - c_2)) g = e - (g_0) \]

\[ \Rightarrow \varrho = (z_2^* - z_2) / (c_2^* - c_2) \]
SPECIAL SOUNDNESS:

\[
\begin{align*}
    z_1 g &= c_1 \\
    z_1^* g &= c_1^* \\
    \Rightarrow (z_1^* - z_1) g &= (c_1^* - c_1) \\
    \Rightarrow ((z_1^* - z_1)/(c_1^* - c_1)) g &= e \\
    \Rightarrow q &= (z_1^* - z_1)/(c_1^* - c_1)
\end{align*}
\]

\[
K(a; c, c^*; (c_1, z), (c_1^*, z^*)):
\]

1. \( c_2 \leftarrow c - c_1; c_2^* \leftarrow c^* - c_1^* \)
2. If \( c_1^* \neq c_1 \)
   1. return \( q \leftarrow (z_1^* - z_1)/(c_1^* - c_1) \)
3. else // \( c_2^* \neq c_2 \)
   1. return \( q \leftarrow (z_2^* - z_2)/(c_2^* - c_2) \)
**SECURITY PROOF**

**SPECIAL SOUNDNESS:**

\[ z_1g = c \]
\[ z_1^*g = c \]

\[ \Rightarrow (z_1^* - z_1)g = 0 \]

\[ \Rightarrow ((z_1^* - z_1)/(c_1^* - c_1))g = e \]

\[ \Rightarrow q = (z_1^* - z_1)/(c_1^* - c_1) \]

**SIMULATION:**

**Sim** (g, e, c):

1. \( c_1 \leftarrow C, c_2 \leftarrow (c - c_1) \text{ mod } |C| \)
2. \( z_1, z_2 \leftarrow \mathbb{Z}_p \)
3. \( a_1 \leftarrow z_1g - c_1g; a_2 \leftarrow z_2g - c_2(e - (g_0)^{c_2}) \)
4. return \((a; c; (c_1, z))\)
Σ-PROTOCOLS FOR BOOLEAN CIRCUITS
Sigma-PROTOCOLS FOR BOOLEAN CIRCUITS

- Each Boolean circuit can be built from NAND gates
Each Boolean circuit can be built from NAND gates

\[ x \text{ NAND } y = 1 \text{ iff } x = 0 \text{ or } y = 0 \]
**Σ-PROTOCOLS FOR BOOLEAN CIRCUITS**

- Each Boolean circuit can be built from NAND gates
  - \( x \text{ NAND } y = 1 \) iff \( x = 0 \) or \( y = 0 \)
- Easy to verify that NAND is observed:
Σ-PROTOCOLS FOR BOOLEAN CIRCUITS

- Each Boolean circuit can be built from NAND gates
  - \( x \text{ NAND } y = 1 \) iff \( x = 0 \) or \( y = 0 \)
- Easy to verify that NAND is observed:
  - \( x \text{ NAND } y = z \) iff \( x + y + 2z - 2 \in \{0, 1\} \)

<table>
<thead>
<tr>
<th>( x, y, z )</th>
<th>( x + y + 2z - 2 )</th>
</tr>
</thead>
<tbody>
<tr>
<td>0, 0, 0</td>
<td>-2</td>
</tr>
<tr>
<td>0, 0, 1</td>
<td>0</td>
</tr>
<tr>
<td>0, 1, 0</td>
<td>-1</td>
</tr>
<tr>
<td>0, 1, 1</td>
<td>1</td>
</tr>
<tr>
<td>1, 0, 0</td>
<td>-1</td>
</tr>
<tr>
<td>1, 0, 1</td>
<td>1</td>
</tr>
<tr>
<td>1, 1, 0</td>
<td>0</td>
</tr>
<tr>
<td>1, 1, 1</td>
<td>2</td>
</tr>
</tbody>
</table>
Each Boolean circuit can be built from NAND gates
- \( x \text{ NAND } y = 1 \text{ iff } x = 0 \text{ or } y = 0 \)
- Easy to verify that NAND is observed:
  - \( x \text{ NAND } y = z \text{ iff } x + y + 2z - 2 \in \{0, 1\} \)

**Corollary.** Boolean proofs are sufficient to construct \( \Sigma \)-protocol for CIRCUIT-SAT
Each Boolean circuit can be built from NAND gates:

- \( x \text{ NAND } y = 1 \) iff \( x = 0 \) or \( y = 0 \)

Easy to verify that NAND is observed:

- \( x \text{ NAND } y = z \) iff \( x + y + 2z - 2 \in \{0, 1\} \)

**Corollary.** Boolean proofs are sufficient to construct \( \Sigma \)-protocol for CIRCUIT-SAT

**Proof.** Encrypt each wire value except the last one (which is 1). Prove that each wire value is Boolean. For each gate, prove that NAND is observed.
RECALL: SEMIHONEST MODEL

ω

f
RECALL: SEMIHONEST MODEL

\[ \omega \rightarrow x = \text{Encoded}(\omega) \rightarrow f \]
RECALL: SEMIHONEST MODEL

\[ x = \text{Encoded}(\omega) \]

\[ \text{Encoded}(f(\omega)) \]
RECALL: SEMIHONEST MODEL

\[ \omega \]

\[ x = \text{Encoded}(\omega) \]

\[ \text{Encoded}(f(\omega)) \]

\[ f \]

Decode, obtain \( f(\omega) \)
We know how to construct protocols for wide array of tasks, that are secure under the assumption that Alice's input belongs to some public set $S_1$.
HALF-WAY THERE: $\Sigma$-PROTOCOLS
HALF-WAY THERE: Σ-PROTOCOLS

ω

x = Encoded(ω), a of PK(ω ∈ S₁)

f
HALF-WAY THERE: $\Sigma$-PROTOCOLS

$x = \text{Encoded}(\omega), a \text{ of } \text{PK}(\omega \in S_1)$

Honesty chosen $c$
HALF-WAY THERE: $\Sigma$-PROTOCOLS

$\omega$  

$x = \text{Encoded}(\omega)$, $a$ of $\text{PK}(\omega \in S_1)$  

Honestly chosen $c$  

$z$ of $\text{PK}(\omega \in S_1)$  

$f$
HALF-WAY THERE: $\Sigma$-PROTOCOLS

$\omega$

$x = \text{Encoded}(\omega)$, $a$ of $\text{PK}(\omega \in S_1)$

Honestly chosen $c$

$z$ of $\text{PK}(\omega \in S_1)$

$f$

If $(a, c, z)$ is not an accepting view with input $x$, abort
HALF-WAY THERE: Σ-PROTOCOLS

\[ x = \text{Encoded}(\omega), \ a \text{ of } \text{PK}(\omega \in S_1) \]

\[ \text{Honesty chosen } c \]

\[ z \text{ of } \text{PK}(\omega \in S_1) \]

\[ \text{Encoded}(f(\omega)) \]

If \((a, c, z)\) is not an accepting view with input \(x\), abort
HALF-WAY THERE: $\Sigma$-PROTOCOLS

$\omega$:
- $x = \text{Encoded}(\omega)$, $a$ of $\text{PK}(\omega \in S_1)$
- Honestly chosen $c$
- $z$ of $\text{PK}(\omega \in S_1)$

$\text{Encode}(f(\omega))$

$f$:
- If $(a, c, z)$ is not an accepting view with input $x$, abort

Decode, obtain $f(\omega)$
HALF-WAY THERE: Σ-PROTOCOLS

Add a Σ-protocol that convinces Bob that $x \in L$, e.g., $L = \{x: x = \text{Encoded}(\omega) \text{ for some } \omega \in S_1\}$
GOAL: FULL ZK

\[ \omega \]

\[ x = \text{Encoded}(\omega), \text{ a of } \text{PK}(\omega \in S) \]

Arbitrary \( c \)

\[ z \text{ of } \text{PK}(\omega \in S_1) \]

Video

Add some additional steps...

Encoded(\( f(\omega) \))

If \((a, c, z)\) is not an accepting view with input \( x \), abort

Decode, obtain \( f(\omega) \)
**BASIC IDEA**

- **x, ω** (1st message: commitment a)
- **c** (2nd message: challenge)
- **z** (3rd message: response)

Σ-protocol is only zero knowledge when c is completely random. This is since we start simulating by picking c randomly, and then choose (z, a). It suffices for c to be independent of a: Alice’s best strategy is then to guess c.

Accepts iff prover knows ω such that (x, ω) ∈ R
BASIC IDEA

Σ-protocol is only zero knowledge when $c$ is completely random. This is since we start simulating by picking $c$ randomly, and then choose $(z, a)$. It suffices for $c$ to be independent of $a$: Alice’s best strategy is then to guess $c$.

Goal: guarantee $c$ is independent of $a$
QUIZ: HOW?
Question: how to guarantee $a$ and $c$ are mutually independent?
Question: how to guarantee $a$ and $c$ are mutually independent?

Hint:
Question: how to guarantee $a$ and $c$ are mutually independent?

Hint:

- Internet is asynchronous, so one message (say $c$) must be sent first, but in a "hidden" form
**Question:** how to guarantee \( a \) and \( c \) are mutually independent?

**Hint:**

- Internet is asynchronous, so one message (say \( c \)) must be sent first, but in a "hidden" form.
- Content only revealed after second message (say \( a \)) is sent.
FIRST IDEA: ENCRYPTION

$x, \omega, pk$

$x, pk, sk$
FIRST IDEA: ENCRYPTION

\[ x, \omega, \text{pk} \quad \text{C} \leftarrow \text{Enc}(c; r) \quad x, \text{pk}, \text{sk} \]
FIRST IDEA: ENCRYPTION

\[ x, \omega, pk \quad \rightarrow \quad C \leftarrow \text{Enc}(c; r) \quad \leftarrow \quad x, pk, sk \]
FIRST IDEA: ENCRYPTION

\[
x, \omega, pk \quad C \leftarrow \text{Enc}(c; r) \quad x, pk, sk
\]
FIRST IDEA: ENCRYPTION

\[ C \leftarrow \text{Enc}(c; r) \]

Abort if \( C \neq \text{Enc}(c; r) \)
FIRST IDEA: ENCRYPTION

\[ C \leftarrow \text{Enc}(c; r) \]

Abort if \( C \neq \text{Enc}(c; r) \)
**FIRST IDEA: ENCRYPTION**

- $x, \omega, \text{pk}$
- $C \leftarrow \text{Enc}(c; r)$
- $x, \text{pk}, \text{sk}$

Abort if $C \neq \text{Enc}(c; r)$

Accepts iff prover knows $\omega$ such that $(x, \omega) \in R$
FIRST IDEA: ENCRYPTION

Seems legit?

- $c$ is independent of $a$ since $x = \text{Enc}(\cdot)$ is sent to Alice first, and $x$ has unique decryption
- Bob cannot "change" $c$ later
- $a$ is independent of $c$ since due to IND-CPA security, $C$ reveals no information about $c$

$\begin{align*}
  x, \omega, \text{pk} & \quad \text{C} \leftarrow \text{Enc}(c; r) & \quad x, \text{pk}, \text{sk} \\
  a & \quad c, r & \\
  z & \quad \text{Accepts iff prover knows } \omega \text{ such that } (x, \omega) \in R
\end{align*}$

Abort if $C \neq \text{Enc}(c; r)$
FIRST IDEA: ENCRYPTION

x, ω, pk  →  C ← Enc(c; r)  →  x, pk, sk

Seems legit?

- c is independent of a since x = Enc (...) is sent to Alice first, and x has unique decryption
- Bob cannot "change" c later
- a is independent of c since due to IND-CPA security, C reveals no information about c

Abort if C ≠ Enc (c; r)

Accepts iff prover knows ω such that (x, ω) ∈ R

Those two properties are sufficient: no need to decrypt. Only ability to "open" encryption so one can verify what was inside.
FIRST IDEA: ENCRYPTION

$x, \omega, \text{pk}$ \quad $C \leftarrow \text{Enc}(c; r)$ \quad $x, \text{pk}, \text{sk}$

- $c$ is independent of $a$ since $x = \text{Enc}(\ldots)$ is sent to Alice first, and $x$ has unique decryption
- Bob cannot "change" $c$ later
- $a$ is independent of $c$ since due to IND-CPA security, $C$ reveals no information about $c$

Seems legit?

Accepts iff prover knows $\omega$ such that $(x, \omega) \in R$

Those two properties are sufficient: no need to decrypt. Only ability to "open" encryption so one can verify what was inside

Something weaker than encryption suffices
COMMITMENT SCHEME
A commitment scheme consists of three algorithms:
A commitment scheme consists of three algorithms:

- key generation $\text{Gen}(\ldots) \rightarrow \text{pk}$
A commitment scheme consists of three algorithms:

- key generation $\text{Gen}(\ldots) \rightarrow \text{pk}$
- commitment $\text{Com}_{\text{pk}}(c; r) \rightarrow C$
A commitment scheme consists of three algorithms:

- key generation $\text{Gen}(\ldots) \rightarrow \text{pk}$
- commitment $\text{Com}_{\text{pk}}(c; r) \rightarrow C$
- verification algorithm $\text{Ver}_{\text{pk}}(C; c, r) \in \{0, 1\}$
COMMITMENT SCHEME

pk ← Gen
COMMITMENT SCHEME

public key $pk$

$pk \leftarrow \text{Gen}$
COMMITMENT SCHEME

public key $\mathbf{pk}$

$c, r$

$\mathbf{pk} \leftarrow \text{Gen}$
COMMITMENT SCHEME

public key $pk$

$C \leftarrow \text{Com}_{pk} (c; r)$
COMMITMENT SCHEME

public key pk

\[ C \leftarrow \text{Com}_{pk} (c; r) \]

Store C
COMMITMENT SCHEME

public key $pk$

$C \leftarrow \text{Com}_{pk} (c; r)$

Store $C$

$(c, r)$

$c, r$
COMMITMENT SCHEME

public key $pk$

$pk \leftarrow \text{Gen}$

$c, r$

$(c, r)$

$C \leftarrow \text{Com}_{pk} (c; r)$

Store $C$

Output $\text{Ver}_{pk}(C; c, r)$
public key $pk$

$C \leftarrow \text{Com}_{pk} (c; r)$

Note: in some commitments schemes, Bob has to reveal some extra information on top of $c$ and $r$ (out of scope)
SECURITY GOALS OF COMMITMENT

- **Computational hiding (IND-CPA):**
  - given $c_0, c_1$ (chosen by adversary), $pk$, and $C = \text{Com}_{pk} (c_b; r)$, it is difficult to guess $b$

- **Perfect binding:**
  - for every $C$, there exists at most one $c$ such that $C = \text{Com}_{pk} (c; r)$ for some $r$
IND-CPA PKC = P.B. COMMITMENT

**Theorem.** Every IND-CPA secure cryptosystem is a perfectly binding and computationally hiding commitment scheme.

**Proof.** Obvious:

- perfect binding follows from unique decryption
- computational hiding follows from IND-CPA security
REFINED: P.B. COMMITMENT

\( x, \omega, pk \)

\( x, pk \)
REFINED: P.B. COMMITMENT

\[ x, \omega, \text{pk} \quad \rightarrow \quad C \leftarrow \text{Com}(c; r) \quad \rightarrow \quad x, \text{pk} \]
REFINED: P.B. COMMITMENT

\[ x, \omega, pk \quad \xrightarrow{\text{Com}(c; r)} \quad x, pk \]
REFINED: P.B. COMMITMENT

\[ x, \omega, pk \quad \xrightarrow{C \leftarrow \text{Com}(c; r)} \quad x, pk \]
REFINED: P.B. COMMITMENT

\[ x, \omega, pk \quad \xrightarrow{C \leftarrow \text{Com}(c; r)} \quad x, pk \]

Abort if \( \text{Ver}(C; c, r) = 0 \)
REFINED: P.B. COMMITMENT

\[ x, \omega, pk \quad \xrightarrow{\quad} \quad C \leftarrow \text{Com}(c; r) \quad \xrightarrow{\quad} \quad x, pk \quad \]

Abort if \( \text{Ver}(C; c, r) = 0 \)
REFINED: P.B. COMMITMENT

$(x, \omega, pk) \rightarrow C \leftarrow \text{Com}(c; r) \rightarrow (x, pk)$

Abort if $\text{Ver}(C; c, r) = 0$

Accepts iff prover knows $\omega$ such that $(x, \omega) \in R$
REFINED: P.B. COMMITMENT

$c$ is independent of $a$ since $C = \text{Com}(c; r)$ is sent to Alice first, and Com is perfectly binding

Bob cannot open $C$ to something different

$a$ is independent of $c$ since due to IND-CPA security, $C$ reveals no information about $c$

Accepts iff prover knows $\omega$ such that $(x, \omega) \in R$
REFINED: P.B. COMMITMENT

Seems legit?
- $c$ is independent of $a$ since $C = \text{Com}(c; r)$ is sent to Alice first, and Com is perfectly binding
- Bob cannot open $C$ to something different
- $a$ is independent of $c$ since due to IND-CPA security, $C$ reveals no information about $c$

Abort if $\text{Ver}(C; c, r) = 0$

Accepts iff prover knows $\omega$ such that $(x, \omega) \in R$

Can you find a problem?
SPECIAL SOUNDNESS

$x, \omega, pk$

$x, pk$
SPECIAL SOUNDNESS

\[ x, \omega, pk \]

\[ C \leftarrow \text{Com}(c; r) \]

\[ x, pk \]
SPECIAL SOUNDNESS

\[ x, \omega, pk \quad \rightarrow \quad C \leftarrow \text{Com}(c; r) \quad \rightarrow \quad x, pk \]
SPECIAL SOUNDNESS

\[ x, \omega, pk \quad \quad \quad \quad C \leftarrow \text{Com}(c; r) \quad \quad \quad \quad x, pk \]
SPECIAL SOUNDNESS

\[ x, \omega, pk \quad C \leftarrow \text{Com}(c; r) \quad x, pk \]

Abort if Ver\((C; c, r) = 0\)
SPECIAL SOUNDNESS

\[
x, \omega, \text{pk} \quad \rightarrow \quad C \leftarrow \text{Com}(c; r) \quad \rightarrow \quad \omega, \text{pk} \\
\begin{align*}
&\rightarrow \quad a \\
&\rightarrow \quad c, r \\
&\rightarrow \quad z \\
\end{align*}

\text{Abort if Ver}(C; c, r) = 0
$x, \omega, pk \quad C \leftarrow \text{Com}(c; r) \quad x, pk$

Abort if $\text{Ver}(C; c, r) = 0$

$c^* \neq c, r^*$
SPECIAL SOUNDNESS

\[ x, \omega, pk \]

\[ C \leftarrow \text{Com}(c; r) \]

\[ x, pk \]

Abort if \( \text{Ver}(C; c, r) = 0 \)

Abort if \( \text{Ver}(C; c^*, r^*) = 0 \)
SPECIAL SOUNDNESS

\[ x, \omega, pk \quad C \leftarrow \text{Com}(c; r) \quad x, pk \]

Abort if \( \text{Ver}(C; c, r) = 0 \)

Abort if \( \text{Ver}(C; c^*, r^*) = 0 \)
SPECIAL SOUNDNESS

\[ x, \omega, pk \quad C \leftarrow \text{Com}(c; r) \quad x, pk \]

Abort if \( \text{Ver}(C; c, r) = 0 \)

Abort if \( \text{Ver}(C; c^*, r^*) = 0 \)

Accepts iff both \((a, c, z)\) and \((a, c^*, z^*)\) are accepting views
SPECIAL SOUNDNESS

\[ x, \omega, pk \quad \text{C} \leftarrow \text{Com}(c; r) \quad x, pk \]

Abort if \( \text{Ver}(C; c, r) = 0 \)

Abort if \( \text{Ver}(C; c^*, r^*) = 0 \)

Accepts iff both \((a, c, z)\) and \((a, c^*, z^*)\) are accepting views

Can you find (a) problem?
SPECIAL SOUNDNESS

\[ x, \omega, pk \]

\[ C \leftarrow \text{Com}(c; r) \]

**Problem.** The commitment scheme is perfectly binding: it cannot be the case that \( \text{Ver}(C; c, r) \) and \( \text{Ver}(C; c^*, r^*) \) both accept if \( c \neq c^* \).

Accepts iff both \((a, c, z)\) and \((a, c^*, z^*)\) are accepting views.
SOLUTION?
SOLUTION?

- **Question:** what do do?
Question: what do do?

Hint 1: commitment scheme should not be perfectly binding
**Question:** what do do?

**Hint i:** commitment scheme should not be perfectly binding

For every $c \neq c^*$, $r$ there should exist $r^*$, such that
**SOLUTION?**

- **Question**: what do do?
- **Hint 1**: commitment scheme should not be perfectly binding
  - For every $c \neq c^*$, $r$ there should exist $r^*$, such that $\text{Com} (c; r) = \text{Com} (c^*; r^*)$
Question: what do do?

Hint 1: commitment scheme should not be perfectly binding

For every $c \neq c^*$, $r$ there should exist $r^*$, such that

$$\text{Com} (c; r) = \text{Com} (c^*; r^*)$$

Hint 2: finding such collusions should be easy for extractor, but difficult for a (malicious/honest) verifier
SOLUTION?

✧ **Question**: what do do?

✧ **Hint 1**: commitment scheme should not be perfectly binding

✧ For every $c \neq c^*$, $r$ there should exist $r^*$, such that

\[ \text{Com} (c; r) = \text{Com} (c^*; r^*) \]

✧ **Hint 2**: finding such collusions should be easy for extractor, but difficult for a (malicious/honest) verifier

We need (trapdoor) computational binding, where binding can be broken by a party who knows some "trapdoor"
TRAPDOOR COMMITMENT
TRAPDOOR COMMITMENT

- **Perfect hiding:**
TRAPDOOR COMMITMENT

- **Perfect hiding:**
  - Distribution of $\text{Com}_{pk}(m; r)$ does not depend on $m$
TRAPDOOR COMMITMENT

- **Perfect hiding:**
  - Distribution of $\text{Com}_{pk}(m; r)$ does not depend on $m$

- **Computational binding:**
TRAPDOOR COMMITMENT

- **Perfect hiding:**
  - Distribution of $\text{Com}_{pk}(m; r)$ does not depend on $m$

- **Computational binding:**
  - for every $m, r, m^* \neq m$, without knowing trapdoor
    sk, it is computationally hard to find $r^*$, such that
    $\text{Com}_{pk}(m; r) = \text{Com}_{pk}(m^*; r^*)$
TRAPDOOR COMMITMENT

- **Perfect hiding:**
  - Distribution of $\text{Com}_{\text{pk}} (m; r)$ does not depend on $m$

- **Computational binding:**
  - for every $m, r, m^* \neq m$, without knowing trapdoor sk, it is computationally hard to find $r^*$, such that $\text{Com}_{\text{pk}} (m; r) = \text{Com}_{\text{pk}} (m^*; r^*)$

- **Trapdoor:**
TRAPDOOR COMMITMENT

- **Perfect hiding:**
  - Distribution of $\text{Com}_{\text{pk}} (m; r)$ does not depend on $m$

- **Computational binding:**
  - for every $m, r, m^* \neq m$, without knowing trapdoor $sk$, it is computationally hard to find $r^*$, such that $\text{Com}_{\text{pk}} (m; r) = \text{Com}_{\text{pk}} (m^*; r^*)$

- **Trapdoor:**
  - given $sk, m, r, m^* \neq m$, it is computationally easy to find $r^*$ such that $\text{Com}_{\text{pk}} (m; r) = \text{Com}_{\text{pk}} (m^*; r^*)$
CHOICE OF COMMITMENT SCHEME
We use a commitment scheme that is constructed from any $\Sigma$-protocol for any hard language.
CHOICE OF COMMITMENT SCHEME

- We use a commitment scheme that is constructed from any $\Sigma$-protocol for any hard language.
- We use the fact that in the final protocol we only need to commit to random messages.
We use a commitment scheme that is constructed from any Σ-protocol for any hard language.

We use the fact that in the final protocol we only need to commit to random messages.

The committed message = c of original protocol.
**CHOICE OF COMMITMENT SCHEME**

- We use a commitment scheme that is constructed from any $\Sigma$-protocol for any hard language.
- We use the fact that in the final protocol we only need to commit to random messages.
- The committed message $= \mathbf{c}$ of original protocol.
- We use this commitment scheme since it fits well our goals.
Let $R = \{(x, w)\}$ be a relation, such that one can efficiently verify whether $(x, w) \in R$
Let $R = \{(x, w)\}$ be a relation, such that one can efficiently verify whether $(x, w) \in R$

Let $R(x) := \{w : (x, w) \in R\}$
Let $R = \{(x, w)\}$ be a relation, such that one can efficiently verify whether $(x, w) \in R$
Let $R(x) := \{w : (x, w) \in R\}$
Let $L := \{x : (\exists \text{short } w) R(x, w)\}$
HARD RELATION

Let $R = \{(x, w)\}$ be a relation, such that one can efficiently verify whether $(x, w) \in R$

Let $R(x) := \{w : (x, w) \in R\}$

Let $L := \{x : (\exists \text{short } w) \ R(x, w)\}$

Relation $R = \{(x, w)\}$ is **hard**, if given random $x \in L$, it is difficult to find a value $w \in R(x)$
Let $R = \{(x, w)\}$ be a relation, such that one can efficiently verify whether $(x, w) \in R$

Let $R(x) := \{w : (x, w) \in R\}$

Let $L := \{x : (\exists \text{short } w) R(x, w)\}$

Relation $R = \{(x, w)\}$ is hard, if given random $x \in L$, it is difficult to find a value $w \in R(x)$

Given $(x, w)$, it is efficient to verify if $(x, w) \in R$
Let \( R = \{(x, w)\} \) be a relation, such that one can efficiently verify whether \((x, w) \in R\)

Let \( R(x) := \{w : (x, w) \in R\} \)

Let \( L := \{x : (\exists \text{short } w) \ R(x, w)\} \)

Relation \( R = \{(x, w)\} \) is **hard**, if given random \( x \in L \), it is difficult to find a value \( w \in R(x) \)

Given \((x, w)\), it is efficient to verify if \((x, w) \in R\)

Example: \( R = \{(pk, sk)\} \)
COMMUNICATION FROM Σ-PROTOCOLS

Pick any hard relation \( R', (X, \Omega) \in R' \)
COMMITMENT FROM Σ-PROTOCOLS

public key $X$

Pick any hard relation $R', (X, \Omega) \in R'$
COMMITMENT FROM Σ-PROTOCOLS

public key $X$

Pick any hard relation $R', (X, \Omega) \in R'$

Use simulator of $PK(\Omega)$ to create accepting view $(A, c, Z)$
COMMITMENT FROM $\Sigma$-PROTOCOLS

public key $X$

Com $(c; Z) := A$

Pick any hard relation $R', (X, \Omega) \in R'$

Use simulator of $\text{PK}(\Omega)$ to create accepting view $(A, c, Z)$
COMMITMENT FROM Σ-PROTOCOLS

public key $X$

$\text{Com } (c; Z) := A$

Pick any hard relation $R', (X, \Omega) \in R'$

Use simulator of $\text{PK}(\Omega)$ to create accepting view $(A, c, Z)$

Store commitment $A$
COMMITMENT FROM Σ-PROTOCOLS

public key $X$

$\text{Com } (c; Z) := A$

Use simulator of $\text{PK}(\Omega)$ to create accepting view $(A, c, Z)$

Pick any hard relation $R', (X, \Omega) \in R'$

Store commitment $A$
COMMITMENT FROM $\Sigma$-PROTOCOLS

public key $X$

$\text{Com} \ (c; Z) := A$

Use simulator of $\text{PK}(\Omega)$ to create accepting view $(A, c, Z)$

Pick any hard relation $R', (X, \Omega) \in R'$

Store commitment $A$

$\text{Ver}(A; c, Z) = 1$ iff it is an accepting view of $\text{PK}(\Omega)$
EXAMPLE: DL BASED

Pick random $X \leftarrow g^\Omega$
EXAMPLE: DL BASED

Pick random $X \leftarrow g^\Omega$

public key $X$
EXAMPLE: DL BASED

public key $X$

Pick random $X \leftarrow g^\Omega$

$Z \leftarrow \mathbb{Z}_p$

$A \leftarrow X^{-c} g^Z$
EXAMPLE: DL BASED

public key $X$

Pick random $X \leftarrow g^\Omega$

$Z \leftarrow \$ \mathbb{Z}_p$

$A \leftarrow X^{-c} g^Z$

$\text{Com}_X (c; Z) := A$
EXAMPLE: DL BASED

Pick random $X \leftarrow g^\Omega$

Store commitment $A$

public key $X$

$Z \leftarrow \mathbb{Z}_p$

$A \leftarrow X^{-c} g^Z$

$\text{Com}_X (c; Z) := A$
EXAMPLE: DL BASED

\[ \text{Com}_X (c; Z) := A \]

\[
\begin{align*}
Z & \leftarrow \$ \mathbb{Z}_p \\
A & \leftarrow X^{-c} g^Z
\end{align*}
\]

Pick random \( X \leftarrow g^\Omega \)

Store commitment \( A \)

(c, Z)
EXAMPLE: DL BASED

\[ \text{Com}_X (c; Z) := A \]

\[ Z \leftarrow \$ \mathbb{Z}_p \]
\[ A \leftarrow X^{-c} g^Z \]

\[ \text{Ver}_X (A; c, Z) = 1 \text{ iff } A = X^c g^Z \]

Pick random \( X \leftarrow g^Q \)
Store commitment \( A \)

Public key \( X \)
EXAMPLE: DL BASED

EXAMPLE: DL BASED

\[ \text{Com}_X (c; Z) := A \]

\[ Z \leftarrow \mathbb{Z}_p \]
\[ A \leftarrow X^{-c} g^Z \]

Store commitment \( A \)

\[ \text{Ver}_X (A; c, Z) = 1 \text{ iff } A = X^{-c} g^Z \]

Pick random \( X \leftarrow g^\Omega \)

Note: a small variation of this commitment scheme is known as the Pedersen commitment.
Theorem. The commitment scheme of the last slide is computationally binding (if $R'$ is hard), perfectly hiding, and trapdoor.
Theorem. The commitment scheme of the last slide is computationally binding (if $R'$ is hard), perfectly hiding, and trapdoor

- *specially sound* $\Rightarrow$ **computational binding:** assume binding adversary outputs commitment collusion $(A, c, Z), (A, c^*, Z^*), c \neq c^*$, s.t. $\text{Ver}$ accepts both. Thus both are accepting views of $\Sigma$-protocol. We can use extractor to recover $\Omega$, thus $R'$ is **not** hard.
**Theorem.** The commitment scheme of the last slide is computationally binding (if \(R'\) is hard), perfectly hiding, and trapdoor

- **specially sound** => **computational binding:** assume binding adversary outputs commitment collusion \((A, c, Z), (A, c^*, Z^*)\), \(c \neq c^*\), s.t. Ver accepts both. Thus both are accepting views of \(\Sigma\)-protocol. We can use extractor to recover \(\Omega\), thus \(R'\) is **not** hard

- **SHVZK** => **perfect hiding:** follows since in real protocol, \(A\) is randomly chosen before \(c\) is chosen, and real protocol and simulator are indistinguishable
SECURITY OF THIS COMMITMENT

**Theorem.** The commitment scheme of the last slide is computationally binding (if $R'$ is hard), perfectly hiding, and trapdoor

- **specially sound** => **computational binding:** assume binding adversary outputs commitment collusion $(A, c, Z)$, $(A, c^*, Z^*)$, $c \neq c^*$, s.t. $Ver$ accepts both. Thus both are accepting views of $\Sigma$-protocol. We can use extractor to recover $\Omega$, thus $R'$ is **not** hard

- **SHVZK** => **perfect hiding:** follows since in real protocol, $A$ is randomly chosen before $c$ is chosen, and real protocol and simulator are indistinguishable

- **Completeness** => **trapdoor:** given $\Omega$, one can start $\Sigma$-protocol with any $A$, and then find $Z$ corresponding to any $c$ such that $(A, c, Z)$ is an accepting view
STUDY OUTCOMES
STUDY OUTCOMES

- \( \Sigma \)-protocol for Booleanity, Circuit-SAT
STUDY OUTCOMES

- Σ-protocol for Booleanity, Circuit-SAT
- Getting full ZK from Σ-protocols
NEXT LECTURE
NEXT LECTURE

✧ ZK from Sigma:
NEXT LECTURE

✧ ZK from Sigma:
  ✧ full construction
NEXT LECTURE

- **ZK from Sigma:**
  - full construction
- **Idea of NIZK**
NEXT LECTURE

- ZK from Sigma:
  - full construction
- Idea of NIZK
- Pairings:
ZK from Sigma:
- full construction

Idea of NIZK

Pairings:
- algebraically "one step up" from exponentiation
Next Lecture

- **ZK from Sigma:**
  - full construction

- **Idea of NIZK**

- **Pairings:**
  - algebraically "one step up" from exponentiation
  - instead of linear functions, allow to compute quadratic functions on ciphertexts, non-interactively
NEXT LECTURE

✦ **ZK from Sigma:**
  ✦ full construction

✦ **Idea of NIZK**

✦ **Pairings:**
  ✦ algebraically "one step up" from exponentiation
  ✦ instead of linear functions, allow to compute quadratic functions on ciphertexts, non-interactively
  ✦ Many, many applications - incl. **efficient** NIZK