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
ΤΗΣ ΤΙΜΗ

• 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

- $\Sigma$-protocols: short reminder
- Continuing with OR rule
- $\Sigma$-protocol for Boolean circuits / $\text{NP}$
- Constructing interactive zero knowledge protocols from $\Sigma$-protocols
REMINDER: $\Sigma$-PROTOCOLS
REMINDER: $\Sigma$-PROTOCOLS

$x, \omega$

$x$
REMINDER: $\Sigma$-PROTOCOLS

$x, \omega$

1st message: commitment $a$

$x$
REMINDER: $\Sigma$-PROTOCOLS

1st message: commitment $a$

2nd message: challenge $c$
REMINDER: Σ-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: Σ-PROTOCOLS**

$x, \omega$

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

1st message: commitment $a$

2nd message: challenge $c$

3rd message: response $z$

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

1. Completeness
2. Special Soundness
3. Special Honest-Verifier ZK (SHVZK)
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. $R_1(x, \omega)$

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

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. $R_1(x, \omega)$

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. $R_1(x, \omega)$

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

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

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

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

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

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

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

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

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

- \(c_1 \leftarrow c - c_2 \mod |C|\)
- \(z_1 \leftarrow P_1(x, \omega, c_1; r)\)
(x, ω): R_I(x, ω)

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

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

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

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

(z_1, z_2, c_1)

\[ x \]

\([a_1, a_2]\)

**Goal:** construct protocol \((P, V)\) for PK \((ω: R_I(x, ω) ∨ R_2(x, ω))\)
OR-PROOF

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

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

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

\(c_1 \leftarrow c - c_2 \text{ mod } \mid C \mid\)
\(z_1 \leftarrow P_1(x, \omega, c_1; r)\)

\((z_1, z_2, c_1)\)

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

\(\omega_{\lambda} s.t. R_1(x, \omega)\)
\(a_1 \leftarrow P_1(x; r)\)
\(c_2 \leftarrow C\)
\((a_2, z_2) \leftarrow S_2(c_2)\)

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

Accept if \(c_1 < \mid C \mid\) 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 \((\omega: R_1(x, \omega) \lor R_2(x, \omega))\)
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

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

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

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

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. \( (a_{21}, a_{22}) \leftarrow (h, g)^{z_2} / (e_1 / g^{\mu}, e_2)^{c_2} \)

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

\[ (g, h, e_1, e_2) \]
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. \((a_{11}, a_{12}) \leftarrow (h, g)^r \text{ // Real branch}\)
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
POK: ELGAMAL PLAINTEXT IS BOOLEAN

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

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. $(a_{21}, a_{22}) \leftarrow (h, g)^{z_2} / (e_1 / g^t, e_2)^{c_2}$

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

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

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

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

1. \[ r \leftarrow \mathbb{Z}_p \]
2. \[(a_{11}, a_{12}) \leftarrow (h, g)^r \text{ // Real branch} \]
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

\[ r \leftarrow \mathbb{Z}_p \]
\[ (a_{11}, a_{12}, a_{21}, a_{22}) \]
\[ c \leftarrow C \]
\[ c_1 \leftarrow c - c_2 \mod |C| \]
\[ z_1 \leftarrow c_1 \rho + r \]

\[(c_1, z)\]
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

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

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

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

\(c \leftarrow C\)

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

\(z_1 \leftarrow c_1 q + r\)

\((c_1, z)\)

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

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

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

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. \( a_2 \leftarrow z_2 g - c_2(e - (0)) \)

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

\( (g, e), (\mu, \varrho) \) \hspace{1cm} \( (g, e) \)

\( (c_1, z) \)

Completeness: obvious

\[ c_1 \leftarrow c - c_2 \text{ mod } |C| \]
\[ z_1 \leftarrow c_1 Q + r_1 \]

Accept if \( c_1 \in C, \)
\[ z_1 g = c_1 e + a_1 \]
\[ z_2 g = c_2(e - (0)) + a_2 \]
SECURITY PROOF

\[ z_1 g = c_1 e + a_1 \]
\[ z_1^* g = c_1^* e + a_1 \]
\[ \Rightarrow (z_1^* - z_1) g = (c_1^* - c_1) e \]
\[ \Rightarrow ((z_1^* - z_1)/(c_1^* - c_1)) g = e \]
\[ \Rightarrow \varrho = (z_1^* - z_1)/(c_1^* - c_1) \]

\[ z_2 g = c_2 (e - (1\ 0)) + a_2 \]
\[ z_2^* g = c_2^* (e - (1\ 0)) + a_2 \]
\[ \Rightarrow (z_2^* - z_2) g = (c_2^* - c_2) (e - (1\ 0)) \]
\[ \Rightarrow ((z_2^* - z_2)/(c_2^* - c_2)) g = e - (1\ 0) \]
\[ \Rightarrow \varrho = (z_2^* - z_2)/(c_2^* - c_2) \]
SECURITY PROOF

**Special Soundness:**

\[ z_1 g = c_1 e + a_1 \]
\[ z_1^* g = c_1^* e + a_1 \]
\[ \Rightarrow (z_1^* - z_1) g = (c_1^* - c_1) e \]
\[ \Rightarrow ((z_1^* - z_1)/(c_1^* - c_1)) g = e \]
\[ \Rightarrow \varrho = (z_1^* - z_1)/(c_1^* - c_1) \]

\[ z_2 g = c_2 (e - \begin{pmatrix} 1 \\ 0 \end{pmatrix}) + a_2 \]
\[ z_2^* g = c_2^* (e - \begin{pmatrix} 1 \\ 0 \end{pmatrix}) + a_2 \]
\[ \Rightarrow (z_2^* - z_2) g = (c_2^* - c_2) (e - \begin{pmatrix} 1 \\ 0 \end{pmatrix}) \]
\[ \Rightarrow ((z_2^* - z_2)/(c_2^* - c_2)) g = e - \begin{pmatrix} 1 \\ 0 \end{pmatrix} \]
\[ \Rightarrow \varrho = (z_2^* - z_2)/(c_2^* - c_2) \]
SECURITY PROOF

**Special soundness:**

\[
\begin{align*}
z_1 g &= c
\end{align*}
\]

\[
\begin{align*}
z_1^* g &= c
\end{align*}
\]

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

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

\[
\Rightarrow q = \frac{(z_1^* - z_1)}{(c_1^* - c_1)}
\]

**K \(a; c, c^*; (c_1, z), (c_2^*, z^*)\):**

1. \(c_2 \leftarrow c - c_1; c_2^* \leftarrow c^* - c_1^*\)

2. If \(c_1^* \neq c_1\)
   1. return \(q \leftarrow \frac{(z_1^* - z_1)}{(c_1^* - c_1)}\)

3. else // \(c_2^* \neq c_2\)
   1. return \(q \leftarrow \frac{(z_2^* - z_2)}{(c_2^* - c_2)}\)

\[
\Rightarrow (c - c_1, z - z_1) - (\frac{1}{0}) + a_2
\]

\[
\Rightarrow (c^* - c_1^*, z^* - z_1) - (\frac{1}{0}) + a_2
\]

\[
\Rightarrow (c - c_1, z - z_1, e - (\frac{1}{0})) - (\frac{1}{0}) + a_2
\]

\[
\Rightarrow (c^* - c_1^*, z^* - z_1, e^* - (\frac{1}{0})) - (\frac{1}{0}) + a_2
\]
**SECURITY PROOF**

**SPECIAL SOUNDNESS:**

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

**SIMULATION:**

\[ \text{Sim} (g, e, c): \]
1. \( c_1 \leftarrow C, c_2 \leftarrow (c - c_1) \mod |C| \)
2. \( z_1, z_2 \leftarrow \mathbb{Z}_p \)
3. \( a_1 \leftarrow z_1 g - c_1 g; a_2 \leftarrow z_2 g - c_2 (e - (1)_0) \)
4. \( \text{return } (a; c; z) \)
Σ-PROTOCOLS FOR BOOLEAN CIRCUITS
Each Boolean circuit can be built from NAND gates
Σ-PROTOCOLS FOR BOOLEAN CIRCUITS

- Each Boolean circuit can be built from NAND gates
  - $x \text{ NAND } y = 1$ iff $x = 0$ or $y = 0$
Σ-PROTOCOLS FOR BOOLEAN CIRCUITS

- 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:
Each Boolean circuit can be built from NAND gates

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

Easy to verify that NAND is observed:

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

**Corollary.** Boolean proofs are sufficient to construct Σ-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

\( \omega \)  

\( f \)
RECALL: SEMIHONEST MODEL

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

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

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

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

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

\[ \omega \rightarrow x = \text{Encoded}(\omega), a \text{ of } \text{PK}(\omega \in S_1) \rightarrow f \]
HALF-WAY THERE: Σ-PROTOCOLS

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

\( f \)

Honestly chosen \( c \)
HALF-WAY THERE: $\Sigma$-PROTOCOLS

$\omega$

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

$\text{Honestly chosen } c$

$f$

$z$ of $\text{PK}(\omega \in S_1)$
HALF-WAY THERE: $\Sigma$-PROTOCOLS

$\omega$

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

Honesty chosen $c$

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

$f$

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

\( \omega \)

- \( x = Encoded(\omega), a \) of PK(\( \omega \in S_1 \))

- Honestly chosen \( c \)

- \( z \) of PK(\( \omega \in S_1 \))

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

\( f \)

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

$\omega$

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

Honesty chosen $c$

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

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

$\text{Decode, obtain } f(\omega)$

$f$
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

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

Arbitrary \( c \)

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

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

Add some additional steps...

If \((a, c, z)\) is not an accepting view with input \(x\), abort.
The basic idea is as follows:

1. **1st message:** Commitment $a$
   - Prover commits to a value $a$.
   - $x, \omega$

2. **2nd message:** Challenge $c$
   - Verifier challenges the prover with a random challenge $c$.
   - The challenge $c$ is random.

3. **3rd message:** Response $z$
   - Prover responds with a value $z$.

The protocol is only zero knowledge when $c$ is completely random. This is because 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 $c$.

$\Sigma$-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 $\omega$ such that $(x, \omega) \in R$
**BASIC IDEA**

1st message: commitment $a$

2nd message: challenge $c$

3rd message: response $z$

- $\Sigma$-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:
QUIZ: HOW?

❖ **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
**QUIZ: HOW?**

✦ **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

\[ C \leftarrow \text{Enc}(c; r) \]
FIRST IDEA: ENCRYPTION

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

\[ C \leftarrow \text{Enc}(c; r) \]
FIRST IDEA: ENCRYPTION

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

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

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

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

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

\( x, \omega, pk \) \hspace{2cm} C \leftarrow \text{Enc}(c; r) \hspace{2cm} x, pk, sk

- \( c \) is independent of \( a \) since \( x = \text{Enc}(...) \) 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 \( c \)

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

Seems legit?
FIRST IDEA: ENCRYPTION

Seems legit?

- $c$ 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 $c$

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.
FIRST IDEA: ENCRYPTION

Seems legit?
- $c$ 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 $c$

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

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

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

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

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

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

Gen

pk ← Gen
COMMITMENT SCHEME

public key \( \text{pk} \)

\( \text{pk} \leftarrow \text{Gen} \)
COMMITMENT SCHEME

public key $pk$

$c, r$

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

public key \( pk \)

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

\( \text{pk} \leftarrow \text{Gen} \)
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 \( C \)

\( (c, r) \)
COMMITMENT SCHEME

public key $pk$

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

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

$c, r$

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

**public key pk**

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

**Store C**

\[ (c, r) \]

**Output Ver}_{pk}(C; 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 $r$
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

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

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

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

\( a \)

\( c, r \)
REFINED: P.B. COMMITMENT

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

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

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

\[ a \quad c, r \quad z \]

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

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

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

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

Can you find a problem?

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

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

\[ x, \omega, pk \]

\[ x, pk \]
SPECIAL SOUNDNESS

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

❖ Question: what do do?
Question: what do do?

Hint 1: commitment scheme should not be perfectly binding
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
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
**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:**
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 $\text{Com}_{pk}(m; r)$ 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:**
**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
CHOICE OF COMMITMENT SCHEME

- We use a commitment scheme that is constructed from any $\Sigma$-protocol for any hard language.
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.
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 = $c$ 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 = $c$ of original protocol.
- We use this commitment scheme since it fits well our goals.
HARD RELATION

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

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

Let $L := \{x : (\exists \text{short } w) R(x, w)\}$
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: \( R = \{(pk, sk)\} \)
COMMITMENT FROM Σ-PROTOCOLS

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

public key $X$

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 $X$

Com $(c; Z) := A$

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

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

public key $X$

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

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

Store commitment $A$

Pick any hard relation $R', (X, \Omega) \in R'$
Commitment from Σ-Protocols

- Public key \( X \)
- Com \((c; Z) := A\)
- Store commitment \( A \)
- 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 $(c; Z) := A$

Ver$(A; c, Z) = 1$ iff it is an accepting view of PK($\Omega$)

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

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

Store commitment $A$
EXAMPLE: DL BASED

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

public key $X$

Pick random $X \leftarrow g^\Omega$
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

public key $X$

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

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

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

Pick random $X$ $\leftarrow$ $g^\Omega$

Store commitment $A$
**EXAMPLE: DL BASED**

- Pick random $X \leftarrow g^{\Omega}$
- $Z \leftarrow \mathbb{Z}_p$
- $A \leftarrow X^{-c} g^Z$

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

- Store commitment $A$

- Public key $X$
EXAMPLE: DL BASED

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

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

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

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

Public key \( X \)

Store commitment \( A \)

\((c, Z)\)
EXAMPLE: DL BASED

Example Diagram:

- Pick random $X \leftarrow g^\Omega$
- $Z \leftarrow \$ \mathbb{Z}_p$
- $A \leftarrow X^{-c} g^Z$

Commitment:

$Com_X(c; Z) := A$

Verification:

$Ver_X(A; c, Z) = 1$ iff $A = X^c$

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.

\begin{itemize}
  \item **specially sound \Rightarrow computational binding:** assume adversary outputs \((A, c, Z), (A, c^*, Z^*), c \neq c^*, \) s.t. Ver accepts both. Thus both are accepting views. We can use extractor to recover \( \Omega \), thus \( R' \) is not hard.
\end{itemize}
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 $\Rightarrow$ computational binding:** assume adversary outputs $(A, c, Z)$, $(A, c^*, Z^*)$, $c \neq c^*$, s.t. Ver accepts both. Thus both are accepting views. We can use extractor to recover $\Omega$, thus $R'$ is not hard

- **SHVZK $\Rightarrow$ 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** $\Rightarrow$ **computational binding:** assume adversary outputs $(A, c, Z)$, $(A, c^*, Z^*)$, $c \neq c^*$, s.t. Ver accepts both. Thus both are accepting views. We can use extractor to recover $\Omega$, thus $R'$ is **not** hard

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

- **Completeness** $\Rightarrow$ **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
Σ-Protocol w/ Trapdoor Commitment

$x, \omega, X$

$x, X$
\[ x, \omega, X \quad c \leftarrow \$ C; A = \text{Com}(c; Z) \quad x, X \]
$\Sigma$-PROTOCOL W/ TRAPDOOR COMMITMENT

$x, \omega, X \quad c \leftarrow C; A = \text{Com}(c; Z) \quad x, X$
Σ-Protocol w/ Trapdoor Commitment

\[ x, \omega, X \quad \quad c \leftarrow C; A = \text{Com}(c; Z) \quad \quad x, X \]
Σ-PROTOCOL W/ TRAPDOOR COMMITMENT

\[ c \leftarrow \$ C; A = \text{Com}(c; Z) \]

Abort if \[ \text{Ver}(A; c, Z) = 0 \]
\textbf{$\Sigma$-PROTOCOL W/ TRAPDOOR COMMITMENT}

$x, \omega, X \quad c \leftarrow_s C; A = \text{Com}(c; Z) \quad x, X$

Abort if $\text{Ver}(A; c, Z) = 0$
Σ-PROTOCOL W/ TRAPDOOR COMMITMENT

$x, \omega, X 
\quad c \leftarrow \$ C; A = \text{Com}(c; Z) 
\quad x, X$

Abort if $\text{Ver}(A; c, Z) = 0$

Accept if $(a, c, z)$ is an accepting view of $\text{PK}(\omega)$
Σ-PROTOCOL W/ TRAPDOOR COMMITMENT

Seems legit?
- $c$ is independent of $a$ since $A = \text{Com}(c; Z)$ is sent to Alice first, and Com is binding
- $a$ is independent of $c$ since $A$ reveals no information about $c$
SOUNDNESS

$x, \omega, X$

$x, X, \Omega$
SOUNDNESS

\[ x, \omega, X \quad c \leftarrow C; A \leftarrow \text{Com}(c; Z) \quad x, X, \Omega \]
SOUNDNESS

$x, \omega, X \quad \leftarrow C; A \leftarrow \text{Com}(c; Z) \quad x, X, \Omega$
SOUNDNESS

\begin{align*}
x, \omega, X \quad & \quad c \leftarrow C; A \leftarrow \text{Com}(c; Z) \quad x, X, \Omega
\end{align*}
SOUNDNESS

\[ x, \omega, X \quad \quad c \leftarrow \$ C; A \leftarrow \text{Com}(c; Z) \quad \quad x, X, \Omega \]

Abort if \( \text{Ver}(A; c, Z) = 0 \)
SOUNDNESS

\[ x, \omega, X \quad c \leftarrow \$ C; A \leftarrow \text{Com}(c; Z) \quad x, X, \Omega \]

Abort if \( \text{Ver}(A; c, Z) = 0 \)
SOUNDNESS

\[ x, \omega, X \quad c \leftarrow s \quad C; A \leftarrow \text{Com}(c; Z) \quad x, X, \Omega \]

Abort if \( \text{Ver}(A; c, Z) = 0 \)

c* \neq c, Z*

\[ a \]

\[ c, Z \]

\[ z \]

\[ c, Z \]
\[ x, \omega, X \quad c \leftarrow \$ C; A \leftarrow \text{Com}(c; Z) \quad x, X, \Omega \]

Abort if \( \text{Ver}(A; c, Z) = 0 \)

Abort if \( \text{Ver}(A; c^*, Z^*) = 0 \)
SOUNDNESS

\[ x, \omega, X \quad c \leftarrow \$ C; A \leftarrow \text{Com}(c; Z) \quad x, X, \Omega \]

Abort if \( \text{Ver}(A; c, Z) = 0 \)

Abort if \( \text{Ver}(A; c^*, Z^*) = 0 \)
\[ x, \omega, X \quad c \leftarrow \$ C; A \leftarrow \text{Com}(c; Z) \quad x, X, \Omega \]

\[ a \]

\[ c, Z \]

\[ c^* \neq c, Z^* \]

Abort if \( \text{Ver}(A; c, Z) = 0 \)

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

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

\[ x, \omega, X \quad c \leftarrow C; A \leftarrow \text{Com}(c; Z) \quad x, X, \Omega \]

Abort if \( \text{Ver}(A; c, Z) = 0 \)

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

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

Can do due to knowledge of trapdoor
QUIZ: ARE WE DONE?
QUIZ: ARE WE DONE?

- **Guess:** are we done?
QUIZ: ARE WE DONE?

- **Guess:** are we done?
- **Answer:**
QUIZ: ARE WE DONE?

- **Guess:** are we done?

- **Answer:**
  - we checked: soundness is ok
QUIZ: ARE WE DONE?

❖ **Guess:** are we done?

❖ **Answer:**

❖ we checked: soundness is ok

❖ but what about ZK? let's check... (not done)
BASIC IDEA

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

Goal: guarantee $c$ is independent of $a$
PROBLEM WITH ZK

\[ x, \omega, X \quad A \leftarrow \text{Com}(c; Z) \quad x, X \]
PROBLEM WITH ZK

\[ x, \omega, X \quad A \leftarrow \text{Com}(c; Z) \quad x, X \]

\[ a \quad c, Z \quad z \]

\( S \) must be able to create accepting view \((A; a; (c, Z); z)\) **without** knowing \( \omega \)
PROBLEM WITH ZK

$x, \omega, X \quad A \leftarrow \text{Com}(c; Z) \quad x, X$

- $S$ must be able to create accepting view $(A; a; (c, Z); z)$ without knowing $\omega$
- It can use $S'$ of $\Sigma$-protocol; nothing else is known
**PROBLEM WITH ZK**

\[ x, \omega, X \quad \xrightarrow{A \leftarrow \text{Com}(c; Z)} \quad x, X \]

- \( S \) must be able to create accepting view \((A; a; (c, Z); z)\) **without** knowing \( \omega \)
- It can use \( S' \) of \( \Sigma \)-protocol; nothing else is known
  - \( S' \): for any \( c \), generate random \( z \), and then \( a \parallel \) out of order
PROBLEM WITH ZK

- $x, \omega, X$
- $A \leftarrow \text{Com}(c; Z)$
- $x, X$

- $S$ must be able to create accepting view $(A; a; (c, Z); z)$ without knowing $\omega$
- It can use $S'$ of $\Sigma$-protocol; nothing else is known
  - $S'$: for any $c$, generate random $z$, and then $a$ // out of order
  - since $c$ is random, choosing random $c$ guarantees correct distribution
PROBLEM WITH ZK

* S must be able to create accepting view \((A; a; (c, Z); z)\) without knowing \(\omega\)
* It can use \(S'\) of \(\Sigma\)-protocol; nothing else is known
  * \(S'\): for any \(c\), generate random \(z\), and then \(a \parallel\) out of order
  * since \(c\) is random, choosing random \(c\) guarantees correct distribution
* In the 4-round protocol, the verifier might try to cheat by choosing weird \(c\)
PROBLEM WITH ZK

- \( S \) must be able to create accepting view \((A; a; (c, Z); z)\) without knowing \( \omega \).
- It can use \( S' \) of \( \Sigma \)-protocol; nothing else is known.
  - \( S' \): for any \( c \), generate random \( z \), and then \( a \) // out of order.
  - since \( c \) is random, choosing random \( c \) guarantees correct distribution.
- In the 4-round protocol, the verifier might try to cheat by choosing weird \( c \).
- If \( c \) is not random then \( z \) is not random, thus this strategy does not work.
**PROBLEM WITH ZK**

- S must be able to create accepting view \((A; a; (c, Z); z)\) **without** knowing \(\omega\)
- It can use \(S'\) of \(\Sigma\)-protocol; nothing else is known
  - \(S'\): for any \(c\), generate random \(z\), and then \(a \parallel \) out of order
  - since \(c\) is random, choosing random \(c\) guarantees correct distribution
- In the 4-round protocol, the verifier might try to cheat by choosing weird \(c\)
- If \(c\) is not random then \(z\) is not random, thus this strategy does not work
- **Solution**: do in-order simulation but use some other “superpower”/trapdoor
REMINDER: BASIC SETTING

\[ A \leftarrow \text{Com}(c; Z) = \text{sim. } A \text{ of } \text{PK}(\Omega) \]

PK (I am P) ≈ PK (I am S)
IDEA: PK (I AM P OR S)

$X, x, \omega$  
$C \leftarrow \text{Com}(c; Z)$  
$x, X, \Omega$

$a$ of PK $(\omega \lor \Omega)$
$c, Z$
$z$ of PK $(\omega \lor \Omega)$

$X, x, \Omega$
$A \leftarrow \text{Com}(c; Z)$
$x, X, \Omega$

$a$ of PK $(\omega \lor \Omega)$
$c, Z$
$z$ of PK $(\omega \lor \Omega)$

Really are indistinguishable
IDEA
Let $V$ create $(X, \Omega) \in R'$, and send $X$ to $P$ with her first message
Let $V$ create $(X, \Omega) \in R'$, and send $X$ to $P$ with her first message.

Quiz: how can simulator obtain $\Omega$ from $V$?
Let $V$ create $(X, \Omega) \in R'$, and send $X$ to $P$ with her first message

**Quiz:** how can simulator obtain $\Omega$ from $V$?

**Answer:** we let $V$ to prove the knowledge of $\Omega$ to $P$
IDEA

- Let $V$ create $(X, \Omega) \in R'$, and send $X$ to $P$ with her first message.

- **Quiz**: how can simulator obtain $\Omega$ from $V$?

- **Answer**: we let $V$ to prove the knowledge of $\Omega$ to $P$.

- Simulator of ZK protocol uses extractor of $\Sigma$-protocol to extract $\Omega$ from $V$. 

4-ROUND ZERO KNOWLEDGE
4-ROUND ZERO KNOWLEDGE

$x, \omega$

$x, \text{ new } (X, \Omega)$
4-ROUND ZERO KNOWLEDGE

$x, \omega$ \quad \rightarrow \quad X, A$ of $\text{PK}(\Omega)$

$x, \text{new} (X, \Omega)$
4-ROUND ZERO KNOWLEDGE

\[ x, \omega \] of PK(\Omega)

\[ a \text{ of } PK(\omega \lor \Omega), C \text{ of } PK(\Omega) \]

\[ X, A \text{ of } PK(\Omega) \]

\[ x, \text{ new } (X, \Omega) \]
4-ROUND ZERO KNOWLEDGE

$X, A$ of $\text{PK}(\Omega)$

$x, \omega$

$a$ of $\text{PK}(\omega \lor \Omega)$, $C$ of $\text{PK}(\Omega)$

$Z$ of $\text{PK}(\Omega)$, $c$ of $\text{PK}(\omega \lor \Omega)$

$x, \text{new} (X, \Omega)$
4-ROUND ZERO KNOWLEDGE

Abort if \((A, C, Z)\) is not an accepting view of \(PK(\Omega)\)
4-ROUND ZERO KNOWLEDGE

$X, A$ of $PK(\Omega)$

$x, \omega$

$a$ of $PK(\omega \lor \Omega)$, $C$ of $PK(\Omega)$

$Z$ of $PK(\Omega)$, $c$ of $PK(\omega \lor \Omega)$

$x$, new $(X, \Omega)$

$z$ of $PK(\omega \lor \Omega)$

Abort if $(A, C, Z)$ is not an accepting view of $PK(\Omega)$
4-ROUND ZERO KNOWLEDGE

Accepts if \((a, c, z)\) is accepting view of \(\text{PK}(\omega \lor \Omega)\)

Abort if \((A, C, Z)\) is not an accepting view of \(\text{PK}(\Omega)\)

since \(X\) is used once, it does not matter if information about \(X\) leaks, thus \(\text{PK}(\Omega)\) can be HVZK

superpower: rewinding again
**Theorem.** Assume $\Sigma$-protocols for $\text{PK}(\omega)$ and $\text{PK}(\Omega)$ are complete, specially sound and SHVZK for language $L$ and language $L'$. Assume $L'$ is a hard language. Then the protocol from the previous slide is a computationally sound and perfectly zero knowledge 4-round proof for $L$. 
THEOREM. Assume $\Sigma$-protocols for $PK(\omega)$ and $PK(\Omega)$ are complete, specially sound and SHVZK for language $L$ and language $L'$. Assume $L'$ is a hard language. Then the protocol from the previous slide is a computationally sound and perfectly zero knowledge 4-round proof for $L$.

- **ZK:** Simulator uses extractor of $PK(\Omega)$ to obtain $\Omega$ and then executes $PK(\omega \lor \Omega)$

**superpower:** using rewinding in simulation
Theorem. Assume $\Sigma$-protocols for $\text{PK}(\omega)$ and $\text{PK}(\Omega)$ are complete, specially sound and SHVZK for language $L$ and language $L'$. Assume $L'$ is a hard language. Then the protocol from the previous slide is a computationally sound and perfectly zero knowledge 4-round proof for $L$.

- **ZK:** Simulator uses extractor of $\text{PK}(\Omega)$ to obtain $\Omega$ and then executes $\text{PK}(\omega \lor \Omega)$ superpower: using rewinding in simulation

- **comp. sound:** Assume prover succeeds in convincing verifier. Then we can use extractor to either extract $\omega$ or $\Omega$ from $P$. But since $L'$ is a hard language, $P$ knows $\Omega$ only with a negligible probability
REMARKS
**REMARKS**

❖ **Important:** $V$ can use *any* hard language $L'$
**REMARKS**

- **Important:** $V$ can use any hard language $L'$
  - just choose $L'$ that suits the rest of ZK proof
**REMARKS**

- **Important:** $V$ can use *any* hard language $L'$
  - just choose $L'$ that suits the rest of ZK proof
  - if ZK proof is about Paillier, use Paillier-based language, etc...
**REMARKS**

- **Important:** $V$ can use *any* hard language $L'$
  - just choose $L'$ that suits the rest of ZK proof
  - if ZK proof is about Paillier, use Paillier-based language, etc...
  - ... or just use something efficient (knowledge of DL)
**REMARKS**

- **Important**: $V$ can use *any* hard language $L'$
  - just choose $L'$ that suits the rest of ZK proof
  - if ZK proof is about Paillier, use Paillier-based language, etc...
  - ... or just use something efficient (knowledge of DL)
- We chose the concrete commitment scheme so that at the last step the number of modifications would be minimal
**REMARKS**

- **Important:** $V$ can use any hard language $L'$
  - just choose $L'$ that suits the rest of ZK proof
  - if ZK proof is about Paillier, use Paillier-based language, etc...
  - ... or just use something efficient (knowledge of DL)
- We chose the concrete commitment scheme so that at the last step the number of modifications would be minimal
- There are many other commitment schemes available
JUNGLE OF INTERACTIVE ZK
JUNGLE OF INTERACTIVE ZK

- Zero-knowledge has many parameters
JUNGLE OF INTERACTIVE ZK

- Zero-knowledge has many parameters
- Plain model, CRS model, random oracle model, bare public key model
JUNGLE OF INTERACTIVE ZK

- Zero-knowledge has many parameters
- Plain model, CRS model, random oracle model, bare public key model
- Standalone/concurrent/UC
Zero-knowledge has many parameters
Plain model, CRS model, random oracle model, bare public key model
Standalone/concurrent/UC

main problem: if several proofs are run in parallel, a message of one proof can be used in another proof. How to construct simulatable proofs becomes major hurdle
JUNGLE OF INTERACTIVE ZK

- Zero-knowledge has many parameters
- Plain model, CRS model, random oracle model, bare public key model
- Standalone/concurrent/UC

**main problem:** if several proofs are run in parallel, a message of one proof can be used in another proof. How to construct simulatable proofs becomes major hurdle

- Resettable
JUNGLE OF INTERACTIVE ZK

- Zero-knowledge has many parameters
- Plain model, CRS model, random oracle model, bare public key model
- Standalone/concurrent/UC

Main problem: if several proofs are run in parallel, a message of one proof can be used in another proof. How to construct simulatable proofs becomes a major hurdle.

- Resettable
- Computational/statistical/perfect soundness
JUNGLE OF INTERACTIVE ZK

- Zero-knowledge has many parameters
- Plain model, CRS model, random oracle model, bare public key model
- Standalone/concurrent/UC

**main problem:** if several proofs are run in parallel, a message of one proof can be used in another proof. How to construct simulatable proofs becomes major hurdle

- Resettable
- Computational/statistical/perfect soundness
- ...

JUNGLE OF INTERACTIVE ZK

- Zero-knowledge has many parameters
- Plain model, CRS model, random oracle model, bare public key model
- Standalone/concurrent/UC

Main problem: if several proofs are run in parallel, a message of one proof can be used in another proof. How to construct simulatable proofs becomes major hurdle

- Resettable
- Computational/statistical/perfect soundness
- ...

You do not have to remember those notions, it's a jungle
NIZK IN CRS MODEL
Several different efficient methodologies to construct non-interactive zero knowledge (NIZK) in "CRS model" are known.
NIZK IN CRS MODEL

- Several different efficient methodologies to construct non-interactive zero knowledge (NIZK) in "CRS model" are known
- Most of them use "pairings"
NIZK IN CRS MODEL

- Several different efficient methodologies to construct non-interactive zero knowledge (NIZK) in "CRS model" are known
- Most of them use "pairings"
- Will talk about pairings in the next lectures
Several different efficient methodologies to construct non-interactive zero knowledge (NIZK) in "CRS model" are known.

Most of them use "pairings"

Will talk about pairings in the next lectures

Will explain NIZK and CRS model after that
STUDY OUTCOMES
STUDY OUTCOMES

- Getting full ZK from $\Sigma$-protocols
STUDY OUTCOMES

- Getting full ZK from $\Sigma$-protocols
- Getting soundness - via commitment schemes
STUDY OUTCOMES

- Getting full ZK from $\Sigma$-protocols
- Getting soundness - via commitment schemes
- Getting ZK - via OR proof and additional $\Sigma$-protocol
STUDY OUTCOMES

- Getting full ZK from $\Sigma$-protocols
- Getting soundness - via commitment schemes
- Getting ZK - via OR proof and additional $\Sigma$-protocol
- We can now construct ZK for any language in NP
STUDY OUTCOMES

- Getting full ZK from Σ-protocols
- Getting soundness - via commitment schemes
- Getting ZK - via OR proof and additional Σ-protocol
- We can now construct ZK for any language in NP
  hence do crypto in malicious model...
STUDY OUTCOMES

- Getting full ZK from $\Sigma$-protocols
- Getting soundness - via commitment schemes
- Getting ZK - via OR proof and additional $\Sigma$-protocol
- We can now construct ZK for any language in NP
  - hence do crypto in malicious model...
  - though not necessarily efficiently
NEXT LECTURE

❖ Pairings:
Pairings:
- algebraically "one step up" from exponentiation
PAIRINGS:

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