Modern Public Key Cryptography

Efficient Zero-Knowledge

Daniel Kales
based on slides by David Derler

Graz, May 5, 2021

SCIENCE
PASSION
TECHNOLOGY

Outline

Efficient ZK Proofs of Knowledge

Efficient NIZK with Random Oracles

Efficient ZK for General Circuits

Recall: Zero Knowledge Proofs

NP-language L w.r.t. relation R

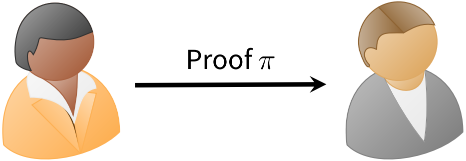
- $x \in L \iff \exists\, w : (x, w) \in R$

Non-interactive proof system

$(x, w) \in R$

$\pi \leftarrow \mathrm{Proof}(x, w)$

$x \overset{?}{\in} L$



$\checkmark / \times \leftarrow \mathrm{Verify}(x, \pi)$

Recall: Zero Knowledge Proofs contd'

Completeness

- Honestly computed proof for $(x, w) \in R$ will always verify

Soundness

- Infeasible to produce valid proof for $x \notin L$

Extractability

- Stronger variant of soundness
- Extract witness from valid proof (using trapdoor)

Recall: Zero Knowledge Proofs contd'

Witness Indistinguishability (WI)

- Distinguish proofs for same x w.r.t. different w, w'

Zero-Knowledge (ZK)

- Stronger variant of witness indistinguishability
- Simulate proofs without knowing w (using trapdoor)

Honest-Verifier Zero Knowledge

complete: honestly computed proofs must always verify

special-sound: dishonest proofs can only verify with negligible probability

(special) honest-verifier zero-knowledge: verifier learns nothing beyond validity of the proof

We consider Σ-protocols

- 3-move public coin HVZKPoK

Honest-Verifier Zero Knowledge

complete: honestly computed proofs must always verify

special-sound: dishonest proofs can only verify with negligible probability

(special) honest-verifier zero-knowledge: verifier learns nothing beyond validity of the proof

We consider Σ-protocols

- 3-move public coin HVZKPoK

Honest-Verifier Zero Knowledge

complete: honestly computed proofs must always verify

special-sound: dishonest proofs can only verify with negligible probability

(special) honest-verifier zero-knowledge: verifier learns nothing beyond validity of the proof

We consider Σ-protocols

- 3-move public coin HVZKPoK

Honest-Verifier Zero Knowledge

complete: honestly computed proofs must always verify

special-sound: dishonest proofs can only verify with negligible probability

(special) honest-verifier zero-knowledge: verifier learns nothing beyond validity of the proof

We consider Σ-protocols

- 3-move public coin HVZKPoK

Σ-protocols for DLOG (Schnorr proof)

Prove knowledge of dlog $k \in \mathbb{Z}_p$ in DL commitment $h = g^k$ of p-order group $G = \langle g \rangle$:

$\mathcal{P}(g, k)$		$\mathcal{V}(g, h)$
pick $r \xleftarrow{R} \mathbb{Z}_p$, $q \leftarrow g^r$	$\xrightarrow{q}$	
	$\xleftarrow{c}$	pick challenge $c \xleftarrow{R} \mathbb{Z}_p$
$z \leftarrow r + ck$	$\xrightarrow{z}$	$g^z \stackrel{?}{=} q \cdot h^c$

We write PoK$\{(\alpha) : h = g^\alpha\}$ to denote such a proof

Completeness?

Σ-protocols for DLOG (Schnorr proof)

Prove knowledge of dlog $k \in \mathbb{Z}_p$ in DL commitment $h = g^k$ of p-order group $G = \langle g \rangle$:

$\mathcal{P}(g, k)$		$\mathcal{V}(g, h)$
pick $r \xleftarrow{R} \mathbb{Z}_p$, $q \leftarrow g^r$	$\xrightarrow{q}$	
	$\xleftarrow{c}$	pick challenge $c \xleftarrow{R} \mathbb{Z}_p$
$z \leftarrow r + ck$	$\xrightarrow{z}$	$g^z \stackrel{?}{=} q \cdot h^c$

We write PoK$\{(\alpha) : h = g^\alpha\}$ to denote such a proof

Completeness?

Σ-protocols for DLOG (Schnorr proof)

Prove knowledge of dlog $k \in \mathbb{Z}_p$ in DL commitment $h = g^k$ of p-order group $G = \langle g \rangle$:

$$
\begin{array}{ccc}
\mathcal{P}(g, k) & & \mathcal{V}(g, h) \\
\hline
\text{pick } r \xleftarrow{R} \mathbb{Z}_p, \quad q \leftarrow g^r & \xrightarrow{q} & \\
 & \xleftarrow{c} & \text{pick challenge } c \xleftarrow{R} \mathbb{Z}_p \\
z \leftarrow r + ck & \xrightarrow{z} & g^z \stackrel{?}{=} q \cdot h^c
\end{array}
$$

We write $\mathrm{PoK}\{(\alpha) : h = g^{\alpha}\}$ to denote such a proof

Completeness?

Σ-protocols for DLOG (Schnorr proof)

Prove knowledge of dlog $k \in \mathbb{Z}_p$ in DL commitment $h = g^k$ of p-order group $G = \langle g \rangle$:

$$
\begin{array}{ccc}
\mathcal{P}(g, k) & & \mathcal{V}(g, h) \\
\hline
\text{pick } r \xleftarrow{R} \mathbb{Z}_p, \quad q \leftarrow g^r & \xrightarrow{q} & \\
& \xleftarrow{c} & \text{pick challenge } c \xleftarrow{R} \mathbb{Z}_p \\
z \leftarrow r + ck & \xrightarrow{z} & g^z \stackrel{?}{=} q \cdot h^c
\end{array}
$$

We write $\mathrm{PoK}\{ (\alpha) : h = g^{\alpha} \}$ to denote such a proof

Completeness?

Σ-protocols for DLOG (Schnorr proof)

Prove knowledge of dlog $k \in \mathbb{Z}_p$ in DL commitment $h = g^k$ of p-order group $G = \langle g \rangle$:

$$
\begin{array}{ccc}
\mathcal{P}(g, k) & & \mathcal{V}(g, h) \\
\hline
\text{pick } r \xleftarrow{R} \mathbb{Z}_p, \quad q \leftarrow g^r & \xrightarrow{q} & \\
& \xleftarrow{c} & \text{pick challenge } c \xleftarrow{R} \mathbb{Z}_p \\
z \leftarrow r + ck & \xrightarrow{z} & g^z \stackrel{?}{=} q \cdot h^c
\end{array}
$$

We write $\mathrm{PoK}\{(\alpha) : h = g^\alpha\}$ to denote such a proof

Completeness?

Σ-protocols for DLOG (Schnorr proof)

Prove knowledge of dlog $k \in \mathbb{Z}_p$ in DL commitment $h = g^k$ of p-order group $G = \langle g \rangle$:

$$\begin{array}{ccc} \mathcal{P}(g,k) & & \mathcal{V}(g,h) \\ \hline \text{pick } r \xleftarrow{R} \mathbb{Z}_p, \quad q \leftarrow g^r & \xrightarrow{q} & \\ & \xleftarrow{c} & \text{pick challenge } c \xleftarrow{R} \mathbb{Z}_p \\ z \leftarrow r + ck & \xrightarrow{z} & g^z \stackrel{?}{=} q \cdot h^c \end{array}$$

We write $\mathrm{PoK}\{(\alpha) : h = g^\alpha\}$ to denote such a proof

Completeness?

Σ-protocols (ctd.)

How is special soundness formalized?

- $\mathcal{P}^*$ can only answer correctly if c guessed!
  - If challenge space chosen large enough,

  $\Rightarrow$ soundness error negligible with one round
- Otherwise, we can extract secret ($\Rightarrow \mathcal{P}$ knows secret)!

Extraction for Schnorr protocol:

- After first showing, rewind $\mathcal{P}$ to step 2
- Two valid showings $(q, c, z), (q, c', z')$: $g^z = q \cdot h^c$ and $g^{z'} = q \cdot h^{c'}$

$\Rightarrow$ $g^{(z-kc)} = g^{(z'-kc')}$, i.e., $k = (z - z')(c - c')^{-1}$

Σ-protocols (ctd.)

How is special soundness formalized?

- $\mathcal{P}^*$ can only answer correctly if c guessed!
  - If challenge space chosen large enough,

  $\Rightarrow$ soundness error negligible with one round
- Otherwise, we can extract secret ($\Rightarrow \mathcal{P}$ knows secret)!

Extraction for Schnorr protocol:

- After first showing, rewind $\mathcal{P}$ to step 2
- Two valid showings $(q, c, z), (q, c', z')$: $g^z = q \cdot h^c$ and $g^{z'} = q \cdot h^{c'}$

$\Rightarrow$ $g^{(z-kc)} = g^{(z'-kc')}$, i.e., $k = (z - z')(c - c')^{-1}$

Σ-protocols (ctd.)

How to show (special) honest-verifier ZK?

- Interaction between $\mathcal{P}$ and $\mathcal{V}$ can be efficiently simulated (HVZK $\rightarrow \mathcal{S}$ does not use $\mathcal{V}^*$)

Simulation of Schnorr protocol

- Pick $c, z \xleftarrow{R} \mathbb{Z}_p$ and set $q \leftarrow g^z / g^c$
- (q, c, z) valid: $g^z = q \cdot g^c$
- (q, c, z) distributed like real interaction

- For special HVZK, $\mathcal{S}$ also gets c as input

Σ-protocols (ctd.)

How to show (special) honest-verifier ZK?

- Interaction between $\mathcal{P}$ and $\mathcal{V}$ can be efficiently simulated (HVZK $\rightarrow \mathcal{S}$ does not use $\mathcal{V}^*$)

Simulation of Schnorr protocol

- Pick $c, z \xleftarrow{R} \mathbb{Z}_p$ and set $q \leftarrow g^z / g^c$
- (q, c, z) valid: $g^z = q \cdot g^c$
- (q, c, z) distributed like real interaction

- For special HVZK, $\mathcal{S}$ also gets c as input

Σ-protocols (ctd.)

How to show (special) honest-verifier ZK?

- Interaction between $\mathcal{P}$ and $\mathcal{V}$ can be efficiently simulated (HVZK $\rightarrow \mathcal{S}$ does not use $\mathcal{V}^*$)

Simulation of Schnorr protocol

- Pick $c, z \xleftarrow{R} \mathbb{Z}_p$ and set $q \leftarrow g^z / g^c$
- (q, c, z) valid: $g^z = q \cdot g^c$
- (q, c, z) distributed like real interaction

- For special HVZK, $\mathcal{S}$ also gets c as input

Σ-protocols (ctd.)

Composition of Σ-protocols:

- Possible to prove more general relations by combining several protocol instances
- E.g. possible to prove relations:
  - AND,
  - OR,
  - EQ,
  - NEQ,
  - Interval, …
- Combination is again Σ-protocol (3-move structure)

Σ-protocols (ctd.)

Composition of Σ-protocols:

- Possible to prove more general relations by combining several protocol instances
- E.g. possible to prove relations:
  - AND,
  - OR,
  - EQ,
  - NEQ,
  - Interval, …
- Combination is again Σ-protocol (3-move structure)

Σ-protocols (ctd.)

Composition of Σ-protocols:

- Possible to prove more general relations by combining several protocol instances
- E.g. possible to prove relations:
  - AND,
  - OR,
  - EQ,
  - NEQ,
  - Interval, …
- Combination is again Σ-protocol (3-move structure)

Σ-protocols (Schnorr AND Proof)

Two values: $h_1 = g^{k_1}, h_2 = g^{k_2}$

$\mathcal{P}(g, k_1, k_2)$		$\mathcal{V}(g, h_1, h_2)$
pick $r_1, r_2 \xleftarrow{R} \mathbb{Z}_p$		
$q_1 \leftarrow g^{r_1},\ q_2 \leftarrow g^{r_2}$	$\xrightarrow{q_1, q_2}$	
	$\xleftarrow{c}$	pick $c \xleftarrow{R} \mathbb{Z}_p$
$z_1 \leftarrow r_1 + ck_1,\ z_2 \leftarrow r_2 + ck_2$	$\xrightarrow{z_1, z_2}$	$g^{z_1} \stackrel{?}{=} q_1 \cdot {h_1}^c$
		$g^{z_2} \stackrel{?}{=} q_2 \cdot {h_2}^c$

We write $\text{PoK}\left\{ (\alpha_1, \alpha_2) : h_1 = g^{\alpha_1} \wedge h_2 = g^{\alpha_2} \right\}$

Σ-protocols (Schnorr AND Proof)

Two values: $h_1 = g^{k_1}, h_2 = g^{k_2}$

$\mathcal{P}(g, k_1, k_2)$		$\mathcal{V}(g, h_1, h_2)$
pick $r_1, r_2 \xleftarrow{R} \mathbb{Z}_p$		
$q_1 \leftarrow g^{r_1},\ q_2 \leftarrow g^{r_2}$	$\xrightarrow{q_1, q_2}$	
	$\xleftarrow{c}$	pick $c \xleftarrow{R} \mathbb{Z}_p$
$z_1 \leftarrow r_1 + ck_1,\ z_2 \leftarrow r_2 + ck_2$	$\xrightarrow{z_1, z_2}$	$g^{z_1} \stackrel{?}{=} q_1 \cdot {h_1}^c$
		$g^{z_2} \stackrel{?}{=} q_2 \cdot {h_2}^c$

We write $\mathrm{PoK}\left\{ (\alpha_1, \alpha_2) : h_1 = g^{\alpha_1} \wedge h_2 = g^{\alpha_2} \right\}$

Σ-protocols (Schnorr AND Proof)

Two values: $h_1 = g^{k_1}, h_2 = g^{k_2}$

$$
\begin{array}{ccc}
\mathcal{P}(g, k_1, k_2) & & \mathcal{V}(g, h_1, h_2) \\
\hline
\text{pick } r_1, r_2 \xleftarrow{R} \mathbb{Z}_p & & \\
q_1 \leftarrow g^{r_1},\ q_2 \leftarrow g^{r_2} & \xrightarrow{q_1, q_2} & \\
& \xleftarrow{c} & \text{pick } c \xleftarrow{R} \mathbb{Z}_p \\
z_1 \leftarrow r_1 + ck_1,\ z_2 \leftarrow r_2 + ck_2 & \xrightarrow{z_1, z_2} & g^{z_1} \stackrel{?}{=} q_1 \cdot {h_1}^c \\
& & g^{z_2} \stackrel{?}{=} q_2 \cdot {h_2}^c
\end{array}
$$

We write $\mathrm{PoK}\left\{(\alpha_1, \alpha_2) : h_1 = g^{\alpha_1} \wedge h_2 = g^{\alpha_2}\right\}$

Σ-protocols (Schnorr AND Proof)

Two values: $h_1 = g^{k_1}, h_2 = g^{k_2}$

$$
\begin{array}{lcl}
\mathcal{P}(g, k_1, k_2) & & \mathcal{V}(g, h_1, h_2) \\
\hline
\text{pick } r_1, r_2 \xleftarrow{R} \mathbb{Z}_p & & \\
q_1 \leftarrow g^{r_1},\ q_2 \leftarrow g^{r_2} & \xrightarrow{q_1, q_2} & \\
& \xleftarrow{c} & \text{pick } c \xleftarrow{R} \mathbb{Z}_p \\
z_1 \leftarrow r_1 + ck_1,\ z_2 \leftarrow r_2 + ck_2 & \xrightarrow{z_1, z_2} & g^{z_1} \stackrel{?}{=} q_1 \cdot {h_1}^c \\
& & g^{z_2} \stackrel{?}{=} q_2 \cdot {h_2}^c
\end{array}
$$

We write $\mathrm{PoK}\left\{ \left(\alpha_1, \alpha_2\right) : h_1 = g^{\alpha_1} \wedge h_2 = g^{\alpha_2} \right\}$

Σ-protocols (Schnorr OR Proof)

Two values: $h_1 = g^{k_1}, h_2 = g^{k_2}$,

- where $\mathcal{P}$ only knows (w.l.o.g.) k_1

Two parallel proofs, where proof for k_2 is $\boxed{\text{simulated}}$:

$\mathcal{P}(g, k_1, h_2)$		$\mathcal{V}(g, h_1, h_2)$
$r_1, \boxed{c_2, z_2 \xleftarrow{R} \mathbb{Z}_p}$		
$q_1 \leftarrow g^{r_1}, \boxed{q_2 \leftarrow g^{z_2} / {h_2}^{c_2}}$	$\xrightarrow{q_1, q_2}$	
	$\xleftarrow{c}$	pick $c \xleftarrow{R} \mathbb{Z}_p$
$c_1 = c - c_2, z_1 = r_1 + c_1 k_1$	$\xrightarrow{c_1, c_2, z_1, z_2}$	$c \stackrel{?}{=} c_1 + c_2$
		$g^{z_1} \stackrel{?}{=} q_1 \cdot {h_1}^{c_1}$
		$g^{z_2} \stackrel{?}{=} q_2 \cdot {h_2}^{c_2}$

Σ-protocols (Schnorr OR Proof)

Two values: $h_1 = g^{k_1}, h_2 = g^{k_2}$,

- where $\mathcal{P}$ only knows (w.l.o.g.) k_1

Two parallel proofs, where proof for k_2 is $\boxed{\text{simulated}}$:

$\mathcal{P}(g, k_1, h_2)$		$\mathcal{V}(g, h_1, h_2)$
$r_1, \boxed{c_2, z_2 \xleftarrow{R} \mathbb{Z}_p}$		
$q_1 \leftarrow g^{r_1}, \boxed{q_2 \leftarrow g^{z_2} / {h_2}^{c_2}}$	$\xrightarrow{q_1, q_2}$	
	$\xleftarrow{c}$	pick $c \xleftarrow{R} \mathbb{Z}_p$
$c_1 = c - c_2,\ z_1 = r_1 + c_1 k_1$	$\xrightarrow{c_1, c_2, z_1, z_2}$	$c \stackrel{?}{=} c_1 + c_2$
		$g^{z_1} \stackrel{?}{=} q_1 \cdot {h_1}^{c_1}$
		$g^{z_2} \stackrel{?}{=} q_2 \cdot {h_2}^{c_2}$

Σ-protocols (Schnorr OR Proof)

Two values: $h_1 = g^{k_1}, h_2 = g^{k_2}$,

- where $\mathcal{P}$ only knows (w.l.o.g.) k_1

Two parallel proofs, where proof for k_2 is $\boxed{\text{simulated}}$:

$$
\begin{array}{ccc}
\mathcal{P}(g, k_1, h_2) & & \mathcal{V}(g, h_1, h_2) \\
\hline
r_1, \boxed{c_2, z_2 \xleftarrow{R} \mathbb{Z}_p} & & \\
q_1 \leftarrow g^{r_1}, \boxed{q_2 \leftarrow g^{z_2} / {h_2}^{c_2}} & \xrightarrow{q_1, q_2} & \\
& \xleftarrow{c} & \text{pick } c \xleftarrow{R} \mathbb{Z}_p \\
c_1 = c - c_2,\ z_1 = r_1 + c_1 k_1 & \xrightarrow{c_1, c_2, z_1, z_2} & c \stackrel{?}{=} c_1 + c_2 \\
& & g^{z_1} \stackrel{?}{=} q_1 \cdot {h_1}^{c_1} \\
& & g^{z_2} \stackrel{?}{=} q_2 \cdot {h_2}^{c_2}
\end{array}
$$

Σ-protocols (Schnorr OR Proof)

Two values: $h_1 = g^{k_1}, h_2 = g^{k_2}$,

- where $\mathcal{P}$ only knows (w.l.o.g.) k_1

Two parallel proofs, where proof for k_2 is simulated:

$\mathcal{P}(g, k_1, h_2)$		$\mathcal{V}(g, h_1, h_2)$
$r_1, \boxed{c_2, z_2 \xleftarrow{R} \mathbb{Z}_p}$		
$q_1 \leftarrow g^{r_1}, \boxed{q_2 \leftarrow g^{z_2} / {h_2}^{c_2}}$	$\xrightarrow{q_1, q_2}$	
	$\xleftarrow{c}$	pick $c \xleftarrow{R} \mathbb{Z}_p$
$c_1 = c - c_2,\ z_1 = r_1 + c_1 k_1$	$\xrightarrow{c_1, c_2, z_1, z_2}$	$c \stackrel{?}{=} c_1 + c_2$
		$g^{z_1} \stackrel{?}{=} q_1 \cdot {h_1}^{c_1}$
		$g^{z_2} \stackrel{?}{=} q_2 \cdot {h_2}^{c_2}$

Σ-protocols (Schnorr OR Proof)

Two values: $h_1 = g^{k_1}, h_2 = g^{k_2}$,

- where $\mathcal{P}$ only knows (w.l.o.g.) k_1

Two parallel proofs, where proof for k_2 is $\boxed{\text{simulated}}$:

$$
\begin{array}{ccc}
\mathcal{P}(g, k_1, h_2) & & \mathcal{V}(g, h_1, h_2) \\
\hline
r_1, \boxed{c_2, z_2 \xleftarrow{R} \mathbb{Z}_p} & & \\
q_1 \leftarrow g^{r_1}, \boxed{q_2 \leftarrow g^{z_2} / {h_2}^{c_2}} & \xrightarrow{q_1, q_2} & \\
& \xleftarrow{c} & \text{pick } c \xleftarrow{R} \mathbb{Z}_p \\
c_1 = c - c_2,\ z_1 = r_1 + c_1 k_1 & \xrightarrow{c_1, c_2, z_1, z_2} & c \stackrel{?}{=} c_1 + c_2 \\
& & g^{z_1} \stackrel{?}{=} q_1 \cdot {h_1}^{c_1} \\
& & g^{z_2} \stackrel{?}{=} q_2 \cdot {h_2}^{c_2}
\end{array}
$$

Σ-protocols (Pedersen Commitments)

Pedersen commitment $C = g^m \cdot h^r$ to $m \in \mathbb{Z}_p$

$\mathcal{P}(g,h,m,r)$		$\mathcal{V}(g,h,C)$
$r_1, r_2 \xleftarrow{R} \mathbb{Z}_p,\ q \leftarrow g^{r_1} \cdot h^{r_2}$	$\xrightarrow{q}$	
	$\xleftarrow{c}$	pick $c \xleftarrow{R} \mathbb{Z}_p$
$z_1 \leftarrow r_1 + cm,\ z_2 \leftarrow r_2 + cr$	$\xrightarrow{z_1, z_2}$	$g^{z_1} \cdot h^{z_2} \stackrel{?}{=} q \cdot C^c$

Completeness?

Σ-protocols (Pedersen Commitments)

Pedersen commitment $C = g^m \cdot h^r$ to $m \in \mathbb{Z}_p$

$\mathcal{P}(g, h, m, r)$		$\mathcal{V}(g, h, C)$
$r_1, r_2 \xleftarrow{R} \mathbb{Z}_p,\ q \leftarrow g^{r_1} \cdot h^{r_2}$	$\xrightarrow{q}$	
	$\xleftarrow{c}$	pick $c \xleftarrow{R} \mathbb{Z}_p$
$z_1 \leftarrow r_1 + cm,\ z_2 \leftarrow r_2 + cr$	$\xrightarrow{z_1, z_2}$	$g^{z_1} \cdot h^{z_2} \stackrel{?}{=} q \cdot C^c$

Completeness?

Σ-protocols (Pedersen Commitments)

Pedersen commitment $C = g^m \cdot h^r$ to $m \in \mathbb{Z}_p$

$$
\begin{array}{ccc}
\mathcal{P}(g,h,m,r) & & \mathcal{V}(g,h,C) \\
\hline
r_1, r_2 \xleftarrow{R} \mathbb{Z}_p,\ q \leftarrow g^{r_1} \cdot h^{r_2} & \xrightarrow{q} & \\
 & \xleftarrow{c} & \text{pick } c \xleftarrow{R} \mathbb{Z}_p \\
z_1 \leftarrow r_1 + cm,\ z_2 \leftarrow r_2 + cr & \xrightarrow{z_1, z_2} & g^{z_1} \cdot h^{z_2} \stackrel{?}{=} q \cdot C^c
\end{array}
$$

Completeness?

Non-Interactive PoKs (Fiat-Shamir Heuristic)

Goal: Make interactive proofs non-interactive

$\Rightarrow$ Then anyone can verify!

Idea: Let prover compute challenge c on its own

- s.t. challenge unpredictable

How? Use hash function on initial commitment q

Applications:

- NIZKPoKs by itself an application!
- Signature schemes from identification schemes

Non-Interactive PoKs (Fiat-Shamir Heuristic)

Goal: Make interactive proofs non-interactive

$\Rightarrow$ Then anyone can verify!

Idea: Let prover compute challenge c on its own

- s.t. challenge unpredictable

How? Use hash function on initial commitment q

Applications:

- NIZKPoKs by itself an application!
- Signature schemes from identification schemes

Non-Interactive PoKs (Fiat-Shamir Heuristic)

Goal: Make interactive proofs non-interactive

$\Rightarrow$ Then anyone can verify!

Idea: Let prover compute challenge c on its own

- s.t. challenge unpredictable

How? Use hash function on initial commitment q

Applications:

- NIZKPoKs by itself an application!
- Signature schemes from identification schemes

Non-Interactive PoKs (Fiat-Shamir Heuristic)

Goal: Make interactive proofs non-interactive

$\Rightarrow$ Then anyone can verify!

Idea: Let prover compute challenge c on its own

- s.t. challenge unpredictable

How? Use hash function on initial commitment q

Applications:

- NIZKPoKs by itself an application!
- Signature schemes from identification schemes

Schnorr Signature

Non-interactive Schnorr protocol

- + inclusion of message m into computation of challenge c!

$\Rightarrow$ Secure digital signature in ROM

Apply Fiat-Shamir:

- $q \leftarrow g^r$ as in Schnorr protocol
- Set challenge $c \leftarrow H(m \| q)$, where H hash function
- $z \leftarrow r + ck$ as in Schnorr protocol

If H is random-oracle, value c not predictable!

Schnorr Signature

Non-interactive Schnorr protocol

- + inclusion of message m into computation of challenge c!

$\Rightarrow$ Secure digital signature in ROM

Apply Fiat-Shamir:

- $q \leftarrow g^r$ as in Schnorr protocol
- Set challenge $c \leftarrow H(m \| q)$, where H hash function
- $z \leftarrow r + ck$ as in Schnorr protocol

If H is random-oracle, value c not predictable!

Schnorr Signature

Non-interactive Schnorr protocol

- + inclusion of message m into computation of challenge c!

$\Rightarrow$ Secure digital signature in ROM

Apply Fiat-Shamir:

- $q \leftarrow g^r$ as in Schnorr protocol
- Set challenge $c \leftarrow H(m \| q)$, where H hash function
- $z \leftarrow r + ck$ as in Schnorr protocol

If H is random-oracle, value c not predictable!

Schnorr Signature (ctd.)

Scheme

KeyGen(1^κ): Choose $\mathcal{G}^\kappa = (\mathbb{G}, p, g)$, $k \xleftarrow{R} \mathbb{Z}_p$, compute $h \leftarrow g^k$ and return $(sk, pk) \leftarrow (k, h)$

Sign(m, sk): Pick $r \xleftarrow{R} \mathbb{Z}_p^*$, compute $q \leftarrow g^r$, $c \leftarrow H(m \| q)$ and $z \leftarrow r + ck$ and output $\sigma \leftarrow (c, z)$

Verify(m, σ, pk): Return $[c = H(m \| g^z / h^c)]$

EUF-CMA secure in ROM based on DLP!

Notes

Is HVZK too weak in practice?

- Fiat-Shamir Heuristic
  - Verifier is forced to be honest
  - ZK in random oracle model
- Conversion for HVZK Σ-protocols to ZK ones [2]

Omega Protocols

- Online extractability instead of rewinding $\mathcal{P}$
- Compatible with the UC framework
- Tighter reductions

Notes

Is HVZK too weak in practice?

- Fiat-Shamir Heuristic
  - Verifier is forced to be honest
  - ZK in random oracle model
- Conversion for HVZK Σ-protocols to ZK ones [2]

Omega Protocols

- Online extractability instead of rewinding $\mathcal{P}$
- Compatible with the UC framework
- Tighter reductions

ZK for General Circuits

So far we have seen practically efficient proofs for statements regarding discrete logarithms.

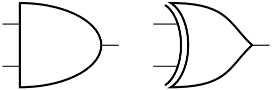
- Very useful in practice
- Building block in many useful protocols
  - secure voting schemes
  - anonymous transactions
  - anonymous credentials

What about arbitrary statements?

Interlude (Completeness of boolean circuits)

Any function computable in finite time can be expressed using a boolean circuit using 2-input gates.

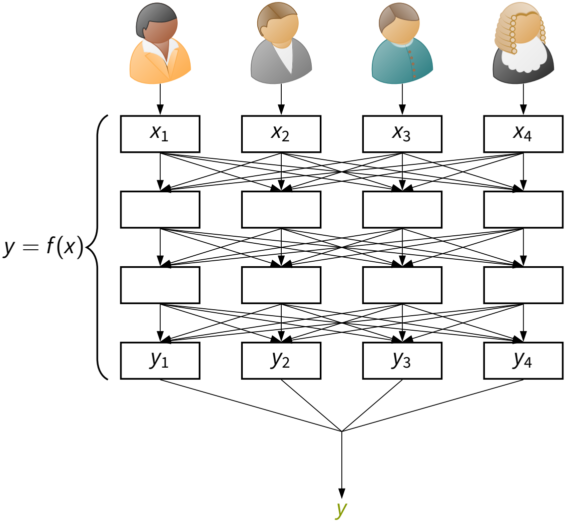
- You may have heard that the `NAND` gate is complete
- So is a combination of `AND` and `XOR` gates
  - This is nice because it maps to fundamental mathematical operations
  - Addition mod 2 $\equiv$ Binary XOR gate
  - Multiplication mod 2 $\equiv$ Binary AND gate



Multiparty Computation

A method to securely evaluate a public function between a number of parties, who hold private inputs.

- Many different protocols exists
  - Many work on a circuit representation of the function
  - Each gate corresponds to a “step” in the MPC protocol
  - Parties may need to communicate to evaluate a gate together
- $(n-1)$-privacy: even if all but one party collude, they cannot learn any information about the true values



MPC-in-the-Head Proof Systems

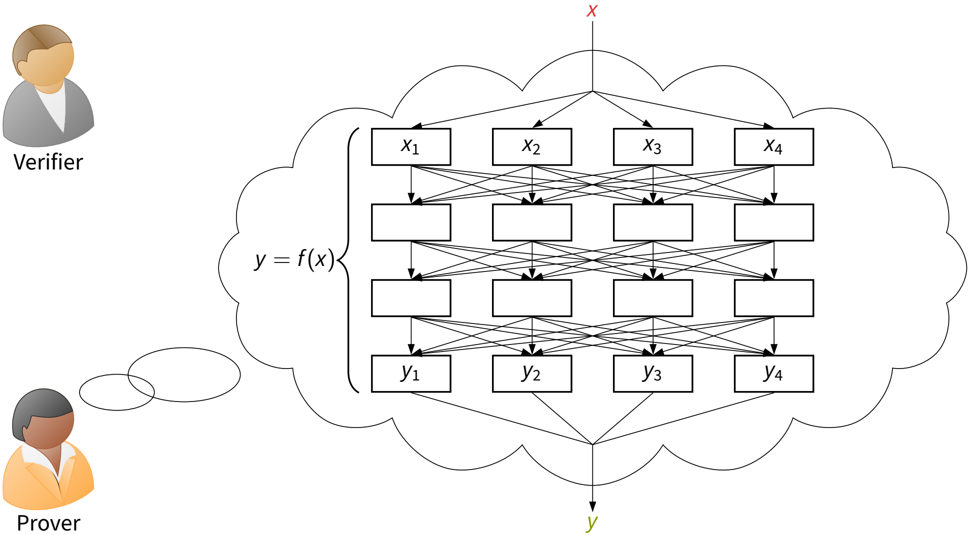
Thinking about Computations

MPC-in-the-Head Paradigm

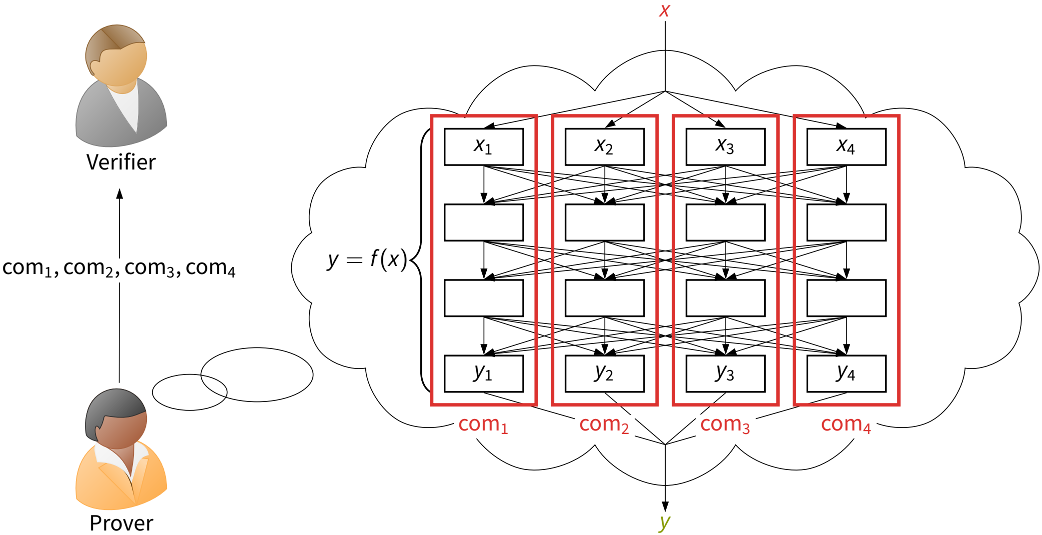
Technique by Ishai et al. (2008) to build a zero-knowledge proof system:

- Take a Multiparty Computation Protocol
- Simulate the evaluation of the function with N players
- Commit to the internal state and messages sent by the players
- Reveal a fraction of the internal states based on a random challenge
  - Not enough to leak any information about the real values
  - Enough that the consistency between the revealed parties can be verified
  - Gain some assurance that the remaining states are also ok

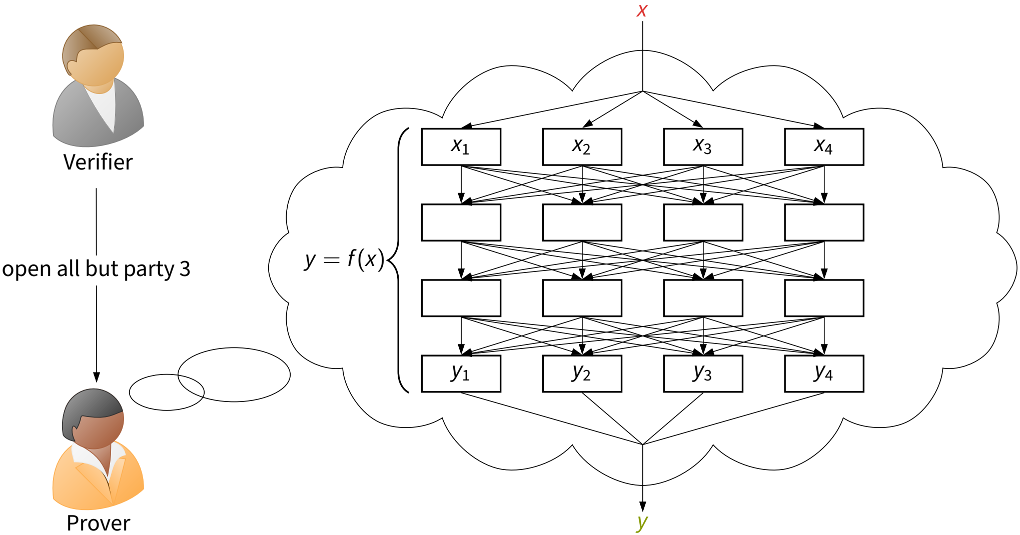
MPC-in-the-Head Paradigm (cont.)



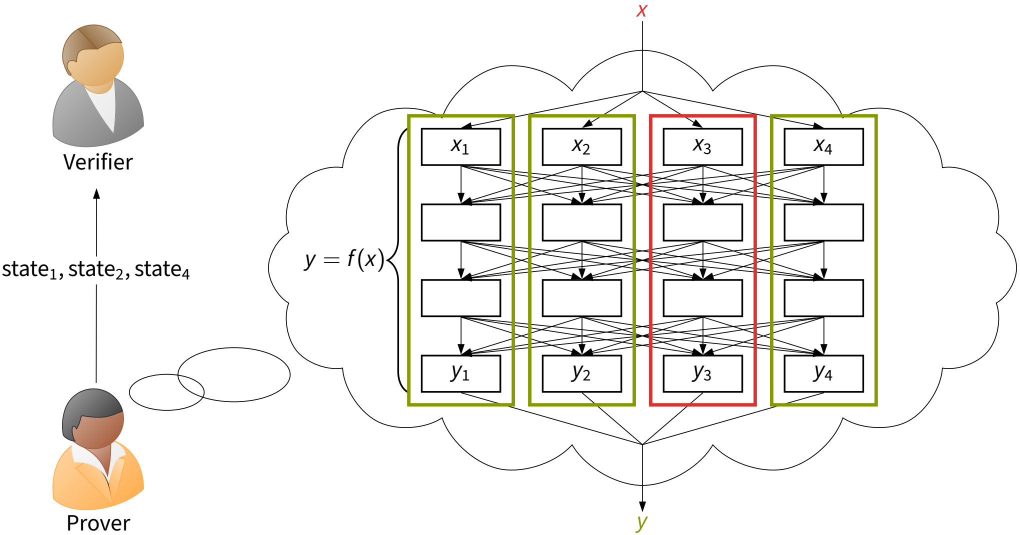
MPC-in-the-Head Paradigm (cont.)



MPC-in-the-Head Paradigm (cont.)



MPC-in-the-Head Paradigm (cont.)



MPCitH as a Sigma Protocol

Can view MPCitH protocol as a Σ-protocol:

- $\mathcal{P}_0$:
  - Prover simulates the MPC execution
  - Commits to state of all players
- $\mathcal{P}_1$:
  - Prover reveals all messages and internal states (except party ch)
- $\mathcal{V}$:
  - Verifier repeats execution with revealed parties
  - Verify consistency of revealed parties

Prover		Verifier
$\mathsf{com} \leftarrow \mathcal{P}_0(x)$	$\xrightarrow{\mathsf{com}}$	
		$\mathsf{ch} \xleftarrow{\$} \mathsf{ChS}(1^k)$
	$\xleftarrow{\mathsf{ch}}$	
$\mathsf{resp} \leftarrow \mathcal{P}_1(x, \mathsf{com}, \mathsf{ch})$		
	$\xrightarrow{\mathsf{resp}}$	
		$b \leftarrow \mathcal{V}(y, \mathsf{com}, \mathsf{ch}, \mathsf{resp})$

Non-Interactive MPCitH proofs

- Fiat-Shamir transformation
  - As seen above
  - Prover calculates challenge
  - Set challenge $c \leftarrow \mathcal{H}(\mathsf{com})$

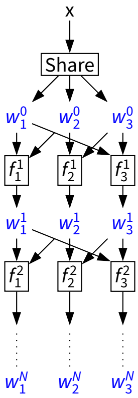
Prover		Verifier
$\mathsf{com} \leftarrow \mathcal{P}_0(x)$		
	$\xrightarrow{\mathsf{com}}$	
		$\mathsf{ch} \xleftarrow{\$} \mathsf{ChS}(1^k)$
	$\xleftarrow{\mathsf{ch}}$	
$\mathsf{resp} \leftarrow \mathcal{P}_1(x, \mathsf{com}, \mathsf{ch})$		
	$\xrightarrow{\mathsf{resp}}$	
		$b \leftarrow \mathcal{V}(y, \mathsf{com}, \mathsf{ch}, \mathsf{resp})$

Prover		Verifier
$\mathsf{com} \leftarrow \mathcal{P}_0(x)$		
	$\xrightarrow{\mathsf{com}}$	
$\mathsf{ch} \leftarrow \mathcal{H}(\mathsf{com})$		
$\mathsf{resp} \leftarrow \mathcal{P}_1(x, \mathsf{com}, \mathsf{ch})$		
	$\xrightarrow{\mathsf{resp}}$	
		$\mathsf{ch} \leftarrow \mathcal{H}(\mathsf{com})$
		$b \leftarrow \mathcal{V}(y, \mathsf{com}, \mathsf{ch}, \mathsf{resp})$

ZK for General Circuits [8, 5]

Instantiation of MPC-in-the-Head approach

1. (2,3)-decompose circuit into three shares
2. Revealing 2 parts reveals no information
3. Evaluate decomposed circuit per share
4. Commit to each evaluation
5. Challenger requests to open 2 of 3
6. Verifies consistency



Proof for $y = \text{SHA-256}(x)$: 13ms to create, 5ms to verify, $\approx$ 220 kilobytes

What you should know...

- Interactive Proof Systems
- Concept of Interactive ZK Proofs (Security Properties)
- Proofs of Knowledge:
  - Security Properties
  - Σ-protocols (Schnorr, compositions, ...)
  - Fiat-Shamir Transform
- Schnorr Signature Scheme
- Idea of ZK for General Circuits
  - MPC-in-the-Head

Questions?

Further Reading I

[1] Mihir Bellare and Oded Goldreich.

On defining proofs of knowledge.

In *Advances in Cryptology - CRYPTO '92, 12th Annual International Cryptology Conference, Santa Barbara, California, USA, August 16-20, 1992, Proceedings*, pages 390–420, 1992.

[2] Ronald Cramer, Ivan Damgård, and Philip D. MacKenzie.

Efficient zero-knowledge proofs of knowledge without intractability assumptions.

In *Public Key Cryptography, Third International Workshop on Practice and Theory in Public Key Cryptography, PKC 2000, Melbourne, Victoria, Australia, January 18-20, 2000, Proceedings*, pages 354–373, 2000.

[3] Ivan Damgard.

On Σ-protocols.

http://cs.au.dk/~ivan/Sigma.pdf.

[4] Juan A. Garay, Philip D. MacKenzie, and Ke Yang.

Strengthening zero-knowledge protocols using signatures.

J. Cryptology, 19(2):169–209, 2006.

Further Reading II

[5] Irene Giacomelli, Jesper Madsen, and Claudio Orlandi.

Zkboo: Faster zero-knowledge for boolean circuits.

In *USENIX Security*, 2016.

[6] Oded Goldreich.

Computational Complexity - A Conceptual Perspective.

Cambridge University Press, 2008.

[7] Jens Groth and Amit Sahai.

Efficient non-interactive proof systems for bilinear groups.

Cryptology ePrint Archive, Report 2007/155, 2007.

http://eprint.iacr.org/2007/155.

[8] Yuval Ishai, Eyal Kushilevitz, Rafail Ostrovsky, and Amit Sahai.

Zero-knowledge from secure multiparty computation.

In *Proceedings of the 39th Annual ACM Symposium on Theory of Computing, San Diego, California, USA, June 11-13, 2007*, pages 21–30, 2007.