Zero-Knowledge Proofs IV

601.442/642 Modern Cryptography

2nd April 2026

Recap: Commitment Schemes

Commitment Scheme

Let $\ell := \ell(\lambda)$ be a polynomial. A PPT algorithm Com is a commitment scheme for ℓ-bit strings if it satisfies the following properties.

- **Hiding:** For every $v_0, v_1 \in \{0,1\}^\ell$, we have

$$\{\mathsf{Com}(v_0\ ;\ r) : r \xleftarrow{\$} \{0,1\}^\lambda\}$$

$$\overset{c}{\approx}\ \{\mathsf{Com}(v_1\ ;\ r) : r \xleftarrow{\$} \{0,1\}^\lambda\}.$$

- **Binding:** For all $v_0, v_1 \in \{0,1\}^\ell$ and $r_0, r_1 \in \{0,1\}^\lambda$, such that $v_0 \neq v_1$, we have

$$\mathsf{Com}(v_0\ ;\ r_0) \neq \mathsf{Com}(v_1\ ;\ r_1).$$

Recap: Bit Commitments from OWP

Theorem: If one-way permutations exist, then there exists a commitment scheme for 1-bit messages.

Proof:

Let $f : \{0,1\}^\lambda \to \{0,1\}^\lambda$ be a OWP and let $\mathsf{hc} : \{0,1\}^\lambda \to \{0,1\}$ be the hardcore predicate for f.

$$\mathsf{Com}(b\ ;\ r) = \Big(f(r),\ \mathsf{hc}(r) \oplus b\Big)$$

Can also be constructed from OWFs, but requires a bit more effort.

Lemma: A bit commitment implies a commitment scheme for polynomial length bit strings.

Simply commit to each bit separately.

ZKP for Graph 3-Coloring

Statement: $G = (V, E)$ where $|V| = n = \text{poly}(\lambda)$.

Prover's NP witness: Color assignment $\phi : V \to \{1, 2, 3\}$.

ZKP for Graph 3-Coloring

Repeat the following procedure $n|E|$ times.

- $P \to V$: P samples $\psi \xleftarrow{\$} \text{Perm}_3$ uniformly at random. For each vertex $v_i \in V$ it computes $\text{color}_i := \psi\left(\phi(v_i)\right)$ and

  $$c_i := \text{Com}(\text{color}_i)$$

  and sends $(c_1, \ldots, c_n)$ to V.

- $V \to P$: V samples a random edge (i, j) and sends it to P.

- $P \to V$: P opens c_i and c_j to reveal color_i and color_j.

- V: If the openings are valid and $\text{color}_i \neq \text{color}_j$, then V continues to the next iteration. Else, it rejects.

If V does not reject in any iteration, it accepts the proof.

ZKP for Graph 3-Coloring

Statement: $G = (V, E)$ where $|V| = n = \text{poly}(\lambda)$.

Prover's NP witness: Color assignment $\phi : V \to \{1, 2, 3\}$.

ZKP for Graph 3-Coloring

~~Repeat the following procedure $n|E|$ times.~~

- $P \to V$: P samples $\psi \xleftarrow{\$} \text{Perm}_3$ uniformly at random. For each vertex $v_i \in V$ it computes $\text{color}_i := \psi\left(\phi(v_i)\right)$ and

$$c_i := \text{Com}(\text{color}_i)$$

  and sends $(c_1, \ldots, c_n)$ to V.

- $V \to P$: V samples a random edge (i, j) and sends it to P.

- $P \to V$: P opens c_i and c_j to reveal color_i and color_j.

- V: If the openings are valid and $\text{color}_i \neq \text{color}_j$, then V continues to the next iteration. Else, it rejects.

If V does not reject in any iteration, it accepts the proof.

Simulator S

ZKP for Graph 3-Coloring

Statement: $G = (V, E)$ where $|V| = n = \text{poly}(\lambda)$.

Prover's NP witness: Color assignment $\phi : V \to \{1, 2, 3\}$.

ZKP for Graph 3-Coloring

~~Repeat the following procedure $n|E|$ times.~~

- $P \to V$: P samples $\psi \xleftarrow{\$} \text{Perm}_3$ uniformly at random. For each vertex $v_i \in V$ it computes $\text{color}_i := \psi\left(\phi(v_i)\right)$ and

$$c_i := \text{Com}(\text{color}_i)$$

  and sends $(c_1, \ldots, c_n)$ to V.

- $V \to P$: V samples a random edge (i, j) and sends it to P.

- $P \to V$: P opens c_i and c_j to reveal color_i and color_j.

- V: If the openings are valid and $\text{color}_i \neq \text{color}_j$, then V continues to the next iteration. Else, it rejects.

If V does not reject in any iteration, it accepts the proof.

Simulator S

- Sample $(i', j') \xleftarrow{\$} E$ and colors $\text{color}_{i'}, \text{color}_{j'} \xleftarrow{\$} \{1, 2, 3\}$ such that $\text{color}_{i'} \neq \text{color}_{j'}$. Set every other $\text{color}_i = 1$.

ZKP for Graph 3-Coloring

Statement: $G = (V, E)$ where $|V| = n = \text{poly}(\lambda)$.

Prover's NP witness: Color assignment $\phi : V \to \{1, 2, 3\}$.

ZKP for Graph 3-Coloring

~~Repeat the following procedure $n|E|$ times.~~

- $P \to V$: P samples $\psi \xleftarrow{\$} \text{Perm}_3$ uniformly at random. For each vertex $v_i \in V$ it computes $\text{color}_i := \psi\left(\phi(v_i)\right)$ and
$$c_i := \text{Com}(\text{color}_i)$$
and sends $(c_1, \ldots, c_n)$ to V.
- $V \to P$: V samples a random edge (i, j) and sends it to P.
- $P \to V$: P opens c_i and c_j to reveal color_i and color_j.
- V: If the openings are valid and $\text{color}_i \neq \text{color}_j$, then V continues to the next iteration. Else, it rejects.

If V does not reject in any iteration, it accepts the proof.

Simulator S

- Sample $(i', j') \xleftarrow{\$} E$ and colors $\text{color}_{i'}, \text{color}_{j'} \xleftarrow{\$} \{1, 2, 3\}$ such that $\text{color}_{i'} \neq \text{color}_{j'}$. Set every other $\text{color}_i = 1$.
- For every $i \in \{1, \ldots, n\}$, compute $c_i \leftarrow \text{Com}(\text{color}_i)$.

ZKP for Graph 3-Coloring

Statement: $G = (V, E)$ where $|V| = n = \text{poly}(\lambda)$.

Prover's NP witness: Color assignment $\phi : V \to \{1, 2, 3\}$.

ZKP for Graph 3-Coloring

~~Repeat the following procedure $n|E|$ times.~~

- $P \to V$: P samples $\psi \xleftarrow{\$} \text{Perm}_3$ uniformly at random. For each vertex $v_i \in V$ it computes $\text{color}_i := \psi\left(\phi(v_i)\right)$ and
$$c_i := \text{Com}(\text{color}_i)$$
and sends $(c_1, \ldots, c_n)$ to V.
- $V \to P$: V samples a random edge (i, j) and sends it to P.
- $P \to V$: P opens c_i and c_j to reveal color_i and color_j.
- V: If the openings are valid and $\text{color}_i \neq \text{color}_j$, then V continues to the next iteration. Else, it rejects.

If V does not reject in any iteration, it accepts the proof.

Simulator S

- Sample $(i', j') \xleftarrow{\$} E$ and colors $\text{color}_{i'}, \text{color}_{j'} \xleftarrow{\$} \{1, 2, 3\}$ such that $\text{color}_{i'} \neq \text{color}_{j'}$. Set every other $\text{color}_i = 1$.
- For every $i \in \{1, \ldots, n\}$, compute $c_i \leftarrow \text{Com}(\text{color}_i)$.
- Execute $\widehat{V}$ and send it $(c_1, \ldots, c_n)$. Let its challenge be $(i, j) \in E$.

ZKP for Graph 3-Coloring

Statement: $G = (V, E)$ where $|V| = n = \text{poly}(\lambda)$.

Prover's NP witness: Color assignment $\phi : V \to \{1, 2, 3\}$.

ZKP for Graph 3-Coloring

~~Repeat the following procedure $n|E|$ times.~~

- $P \to V$: P samples $\psi \xleftarrow{\$} \text{Perm}_3$ uniformly at random. For each vertex $v_i \in V$ it computes $\text{color}_i := \psi\left(\phi(v_i)\right)$ and
$$c_i := \text{Com}(\text{color}_i)$$
and sends $(c_1, \ldots, c_n)$ to V.
- $V \to P$: V samples a random edge (i, j) and sends it to P.
- $P \to V$: P opens c_i and c_j to reveal color_i and color_j.
- V: If the openings are valid and $\text{color}_i \neq \text{color}_j$, then V continues to the next iteration. Else, it rejects.

If V does not reject in any iteration, it accepts the proof.

Simulator S

- Sample $(i', j') \xleftarrow{\$} E$ and colors $\text{color}_{i'}, \text{color}_{j'} \xleftarrow{\$} \{1, 2, 3\}$ such that $\text{color}_{i'} \neq \text{color}_{j'}$. Set every other $\text{color}_i = 1$.
- For every $i \in \{1, \ldots, n\}$, compute $c_i \leftarrow \text{Com}(\text{color}_i)$.
- Execute $\widehat{V}$ and send it $(c_1, \ldots, c_n)$. Let its challenge be $(i, j) \in E$.
- If $(i, j) = (i', j')$, then output $(c_1, \ldots, c_n)$, (i, j), and the opening for $c_{i'}$ and $c_{j'}$ as the verifier's view. Else, restart the above process.

ZKP for Graph 3-Coloring

Statement: $G = (V, E)$ where $|V| = n = \text{poly}(\lambda)$.

Prover's NP witness: Color assignment $\phi : V \to \{1, 2, 3\}$.

ZKP for Graph 3-Coloring

~~Repeat the following procedure $n|E|$ times.~~

- $P \to V$: P samples $\psi \xleftarrow{\$} \text{Perm}_3$ uniformly at random. For each vertex $v_i \in V$ it computes $\text{color}_i := \psi\left(\phi(v_i)\right)$ and
$$c_i := \text{Com}(\text{color}_i)$$
and sends $(c_1, \ldots, c_n)$ to V.
- $V \to P$: V samples a random edge (i, j) and sends it to P.
- $P \to V$: P opens c_i and c_j to reveal color_i and color_j.
- V: If the openings are valid and $\text{color}_i \neq \text{color}_j$, then V continues to the next iteration. Else, it rejects.

If V does not reject in any iteration, it accepts the proof.

Simulator S

- Sample $(i', j') \xleftarrow{\$} E$ and colors $\text{color}_{i'}, \text{color}_{j'} \xleftarrow{\$} \{1, 2, 3\}$ such that $\text{color}_{i'} \neq \text{color}_{j'}$. Set every other $\text{color}_i = 1$.
- For every $i \in \{1, \ldots, n\}$, compute $c_i \leftarrow \text{Com}(\text{color}_i)$.
- Execute $\widehat{V}$ and send it $(c_1, \ldots, c_n)$. Let its challenge be $(i, j) \in E$.
- If $(i, j) = (i', j')$, then output $(c_1, \ldots, c_n)$, (i, j), and the opening for $c_{i'}$ and $c_{j'}$ as the verifier's view. Else, restart the above process.
- If simulation has not succeeded after $n|E|$ attempts, then output $\perp$.

ZKP for Graph 3-Coloring

ZKP for Graph 3-Coloring

- $P \to V$: P samples $\psi \xleftarrow{\$} \mathsf{Perm}_3$ uniformly at random. For each vertex $v_i \in V$ it computes $\mathsf{color}_i := \psi\left(\phi(v_i)\right)$ and
$$c_i := \mathsf{Com}(\mathsf{color}_i)$$
and sends $(c_1, \ldots, c_n)$ to V.
- $V \to P$: $\widehat{V}$ samples a random edge (i, j) and sends it to P.
- $P \to V$: P opens c_i and c_j to reveal color_i and color_j.

Simulator S

- Sample $(i', j') \xleftarrow{\$} E$ and colors $\mathsf{color}_{i'}, \mathsf{color}_{j'} \xleftarrow{\$} \{1, 2, 3\}$ such that $\mathsf{color}_{i'} \neq \mathsf{color}_{j'}$. Set every other $\mathsf{color}_i = 1$.
- For every $i \in \{1, \ldots, n\}$, compute $c_i \leftarrow \mathsf{Com}(\mathsf{color}_i)$.
- Execute $\widehat{V}$ and send it $(c_1, \ldots, c_n)$. Let its challenge be $(i, j) \in E$.
- If $(i, j) = (i', j')$, then output $(c_1, \ldots, c_n)$, (i, j), and the opening for $c_{i'}$ and $c_{j'}$ as the verifier's view. Else, restart the above process.
- If simulation has not succeeded after $n|E|$ attempts, then output $\perp$.

ZKP for Graph 3-Coloring

Goal: Show that the view of $\hat{V}$ in the protocol is indistinguishable from simulated view.

ZKP for Graph 3-Coloring

- $P \to V$: P samples $\psi \xleftarrow{\$} \mathsf{Perm}_3$ uniformly at random. For each vertex $v_i \in V$ it computes $\mathsf{color}_i := \psi\left(\phi(v_i)\right)$ and
$$c_i := \mathsf{Com}(\mathsf{color}_i)$$
and sends $(c_1, \ldots, c_n)$ to V.
- $V \to P$: $\widehat{V}$ samples a random edge (i, j) and sends it to P.
- $P \to V$: P opens c_i and c_j to reveal color_i and color_j.

Simulator S

- Sample $(i', j') \xleftarrow{\$} E$ and colors $\mathsf{color}_{i'}, \mathsf{color}_{j'} \xleftarrow{\$} \{1, 2, 3\}$ such that $\mathsf{color}_{i'} \neq \mathsf{color}_{j'}$. Set every other $\mathsf{color}_i = 1$.
- For every $i \in \{1, \ldots, n\}$, compute $c_i \leftarrow \mathsf{Com}(\mathsf{color}_i)$.
- Execute $\widehat{V}$ and send it $(c_1, \ldots, c_n)$. Let its challenge be $(i, j) \in E$.
- If $(i, j) = (i', j')$, then output $(c_1, \ldots, c_n)$, (i, j), and the opening for $c_{i'}$ and $c_{j'}$ as the verifier's view. Else, restart the above process.
- If simulation has not succeeded after $n|E|$ attempts, then output $\perp$.

ZKP for Graph 3-Coloring

Hybrid 0

- $\psi \xleftarrow{\$} \mathsf{Perm}_3$
- $\forall i, \mathsf{color}_i := \psi\left(\phi(v_i)\right)$
- $\forall i, c_i := \mathsf{Com}(\mathsf{color}_i)$
- $(i, j) \leftarrow \widehat{V}(c_1, \ldots, c_n)$

Output

$(c_1, \ldots, c_n)$ (i, j) Openings for c_i and c_j

This hybrid is $\widehat{V}$'s view in a real execution of the protocol.

ZKP for Graph 3-Coloring

Hybrid 0

- $\psi \xleftarrow{\$} \mathsf{Perm}_3$
- $\forall i, \mathsf{color}_i := \psi\left(\phi(v_i)\right)$
- $\forall i, c_i := \mathsf{Com}(\mathsf{color}_i)$
- $(i, j) \leftarrow \widehat{V}(c_1, \ldots, c_n)$

Output

$(c_1, \ldots, c_n)$ (i, j) Openings for c_i and c_j

This hybrid is $\widehat{V}$'s view in a real execution of the protocol.

Generating this distribution requires ϕ.

ZKP for Graph 3-Coloring

Hybrid 1

If $(i, j) \neq (i', j')$

At most $n|E|$ times

- $\psi \xleftarrow{\$} \text{Perm}_3$
- $(i', j') \xleftarrow{\$} E$
- $\forall i, \text{color}_i := \psi\left(\phi(v_i)\right)$
- $\forall i, c_i := \text{Com}(\text{color}_i)$
- $(i, j) \leftarrow \widehat{V}(c_1, \ldots, c_n)$

Output

$(c_1, \ldots, c_n)$ (i, j) Openings for c_i and c_j

This hybrid is similar to the real world prover, but keeps trying until it guesses $\widehat{V}$'s response.

ZKP for Graph 3-Coloring

Hybrid 1

- $\psi \xleftarrow{\$} \text{Perm}_3$
- $(i', j') \xleftarrow{\$} E$
- $\forall i, \text{color}_i := \psi\left(\phi(v_i)\right)$
- $\forall i, c_i := \text{Com}(\text{color}_i)$
- $(i, j) \leftarrow \widehat{V}(c_1, \ldots, c_n)$

If $(i, j) \neq (i', j')$

At most $n|E|$ times

Output

$(c_1, \ldots, c_n)$ (i, j) Openings for c_i and c_j

This hybrid is similar to the real world prover, but keeps trying until it guesses $\widehat{V}$'s response.

Claim: Hybrid 0 $\overset{s}{\approx}$ Hybrid 1.

ZKP for Graph 3-Coloring

Hybrid 1

If $(i,j) \neq (i',j')$

At most $n|E|$ times

- $\psi \xleftarrow{\$} \text{Perm}_3$
- $(i',j') \xleftarrow{\$} E$
- $\forall i, \text{color}_i := \psi\left(\phi(v_i)\right)$
- $\forall i, c_i := \text{Com}(\text{color}_i)$
- $(i,j) \leftarrow \widehat{V}(c_1, \ldots, c_n)$

Output

$(c_1, \ldots, c_n)$ (i,j) Openings for c_i and c_j

This hybrid is similar to the real world prover, but keeps trying until it guesses $\widehat{V}$'s response.

Claim: Hybrid 0 $\overset{s}{\approx}$ Hybrid 1.

The hybrids are identical conditioned on $(i,j) = (i',j')$ since (i',j') is sampled uniformly at random.

ZKP for Graph 3-Coloring

Hybrid 1

If $(i,j) \neq (i',j')$

At most $n|E|$ times

- $\psi \xleftarrow{\$} \text{Perm}_3$
- $(i',j') \xleftarrow{\$} E$
- $\forall i, \text{color}_i := \psi\left(\phi(v_i)\right)$
- $\forall i, c_i := \text{Com}(\text{color}_i)$
- $(i,j) \leftarrow \widehat{V}(c_1, \ldots, c_n)$

Output

$(c_1, \ldots, c_n)$ (i,j) Openings for c_i and c_j

This hybrid is similar to the real world prover, but keeps trying until it guesses $\widehat{V}$'s response.

Claim: Hybrid 0 $\overset{s}{\approx}$ Hybrid 1.

The hybrids are identical conditioned on $(i,j) = (i',j')$ since (i',j') is sampled uniformly at random.

Therefore, we need to argue that the output is not $\perp$ with overwhelming probability.

ZKP for Graph 3-Coloring

Hybrid 1

If $(i,j) \neq (i',j')$

At most $n|E|$ times

- $\psi \xleftarrow{\$} \text{Perm}_3$
- $(i',j') \xleftarrow{\$} E$
- $\forall i, \text{color}_i := \psi\left(\phi(v_i)\right)$
- $\forall i, c_i := \text{Com}(\text{color}_i)$
- $(i,j) \leftarrow \widehat{V}(c_1, \ldots, c_n)$

Output

$(c_1, \ldots, c_n)$ (i,j) Openings for c_i and c_j

This hybrid is similar to the real world prover, but keeps trying until it guesses $\widehat{V}$'s response.

Claim: Hybrid 0 $\overset{s}{\approx}$ Hybrid 1.

The hybrids are identical conditioned on $(i,j) = (i',j')$ since (i',j') is sampled uniformly at random.

Therefore, we need to argue that the output is not $\perp$ with overwhelming probability.

(i,j) is independent of (i',j') since $(c_1, \ldots, c_n)$ is independent of (i',j').

ZKP for Graph 3-Coloring

Hybrid 1

- $\psi \xleftarrow{\$} \text{Perm}_3$
- $(i', j') \xleftarrow{\$} E$
- $\forall i, \text{color}_i := \psi\left(\phi(v_i)\right)$
- $\forall i, c_i := \text{Com}(\text{color}_i)$
- $(i, j) \leftarrow \widehat{V}(c_1, \ldots, c_n)$

If $(i, j) \neq (i', j')$

At most $n|E|$ times

Output

$(c_1, \ldots, c_n)$ (i, j) Openings for c_i and c_j

This hybrid is similar to the real world prover, but keeps trying until it guesses $\widehat{V}$'s response.

Claim: Hybrid 0 $\overset{s}{\approx}$ Hybrid 1.

The hybrids are identical conditioned on $(i, j) = (i', j')$ since (i', j') is sampled uniformly at random.

Therefore, we need to argue that the output is not $\perp$ with overwhelming probability.

(i, j) is independent of (i', j') since $(c_1, \ldots, c_n)$ is independent of (i', j').

Therefore, probability of not terminating after $n|E|$ attempts is

$$\left(1 - \frac{1}{|E|}\right)^{n|E|} \approx e^{-n} = \text{negl}(\lambda).$$

ZKP for Graph 3-Coloring

Hybrid 1

If $(i,j) \neq (i',j')$

At most $n|E|$ times

- $\psi \xleftarrow{\$} \mathsf{Perm}_3$
- $(i',j') \xleftarrow{\$} E$
- $\forall i, \mathsf{color}_i := \psi\left(\phi(v_i)\right)$
- $\forall i, c_i := \mathsf{Com}(\mathsf{color}_i)$
- $(i,j) \leftarrow \widehat{V}(c_1, \ldots, c_n)$

Output

$(c_1, \ldots, c_n)$ (i,j) Openings for c_i and c_j

This hybrid is similar to the real world prover, but keeps trying until it guesses $\widehat{V}$'s response.

Claim: Hybrid 0 $\overset{s}{\approx}$ Hybrid 1.

The hybrids are identical conditioned on $(i,j) = (i',j')$ since (i',j') is sampled uniformly at random.

Therefore, we need to argue that the output is not $\perp$ with overwhelming probability.

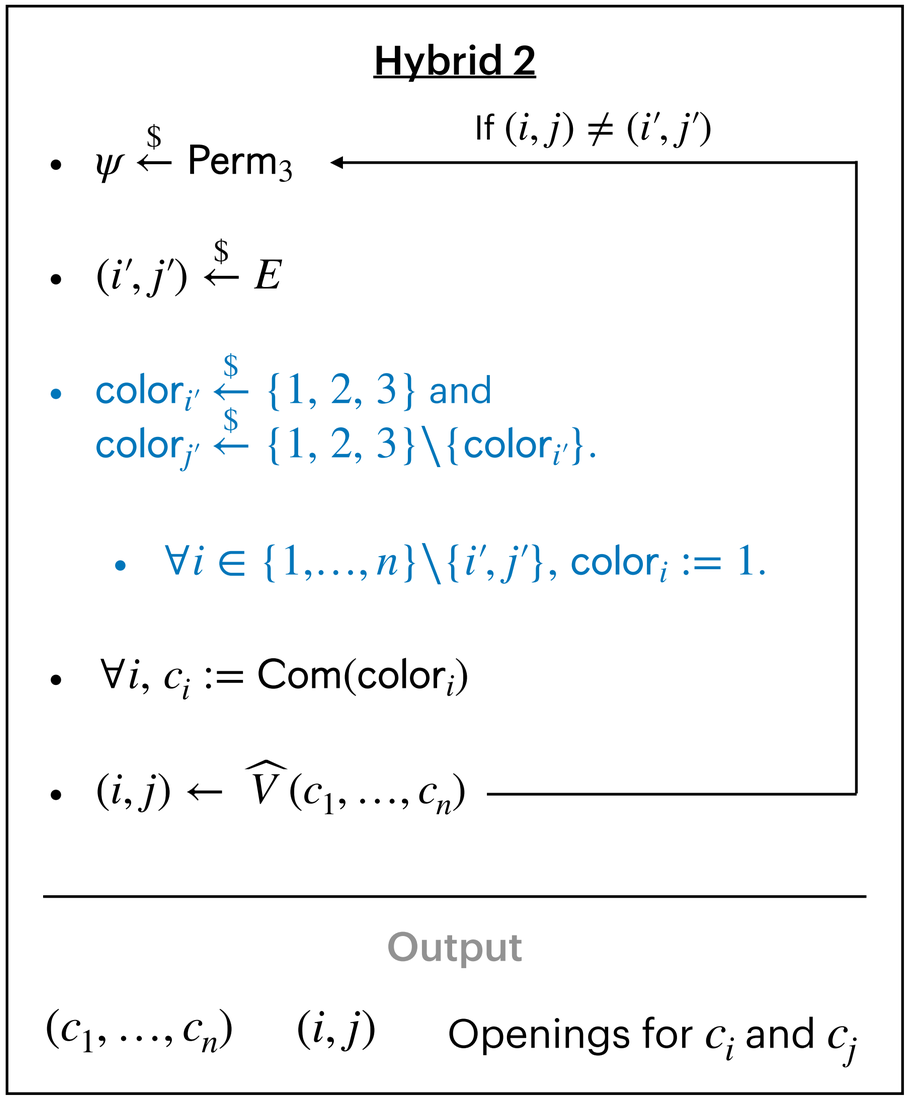
(i,j) is independent of (i',j') since $(c_1, \ldots, c_n)$ is independent of (i',j').

Therefore, probability of not terminating after $n|E|$ attempts is

$$\left(1 - \frac{1}{|E|}\right)^{n|E|} \approx e^{-n} = \mathsf{negl}(\lambda).$$

This hybrid still requires ϕ.

ZKP for Graph 3-Coloring



This hybrid is the simulator S.

ZKP for Graph 3-Coloring

Hybrid 2

If $(i,j) \neq (i',j')$

- $\psi \xleftarrow{\$} \mathsf{Perm}_3$
- $(i',j') \xleftarrow{\$} E$
- $\mathsf{color}_{i'} \xleftarrow{\$} \{1,2,3\}$ and $\mathsf{color}_{j'} \xleftarrow{\$} \{1,2,3\}\backslash\{\mathsf{color}_{i'}\}$.
  - $\forall i \in \{1,\ldots,n\}\backslash\{i',j'\}$, $\mathsf{color}_i := 1$.
- $\forall i$, $c_i := \mathsf{Com}(\mathsf{color}_i)$
- $(i,j) \leftarrow \widehat{V}(c_1,\ldots,c_n)$

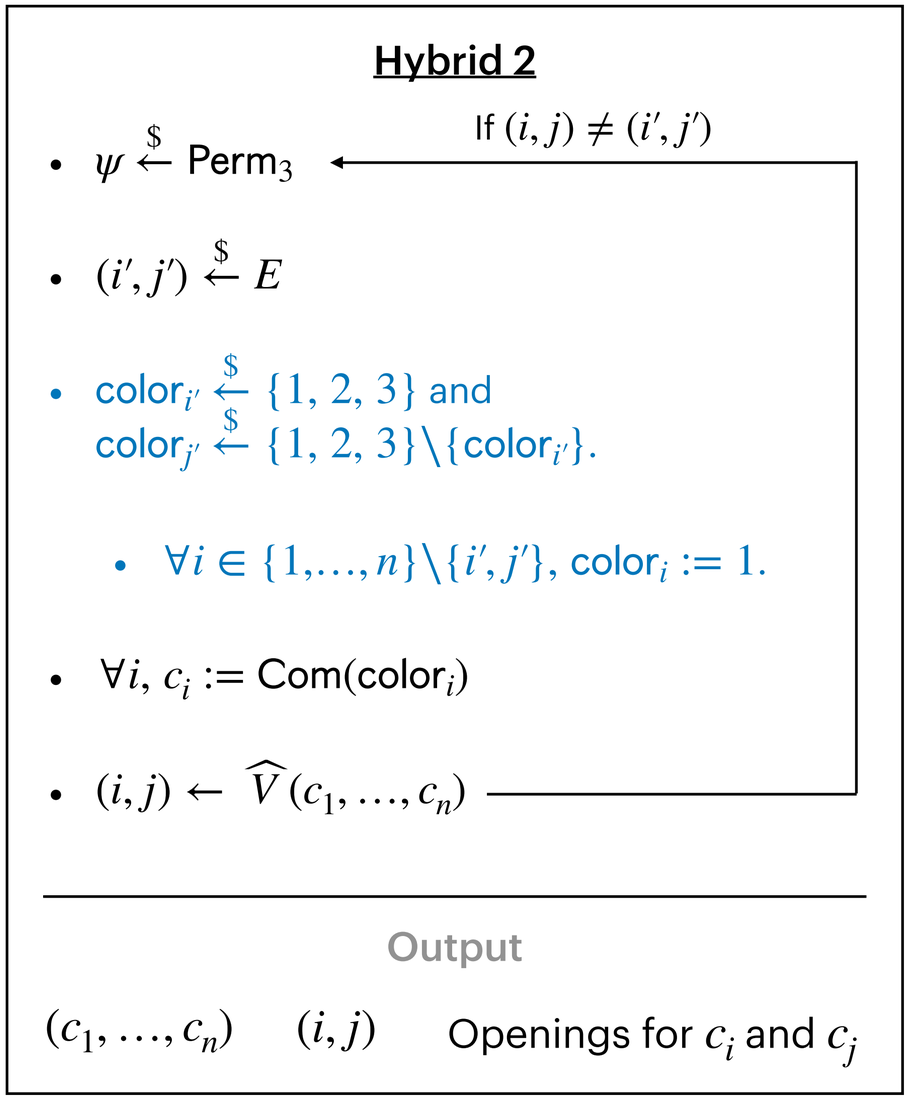
Output

$(c_1,\ldots,c_n)$ (i,j) Openings for c_i and c_j

This hybrid is the simulator S.

Claim: Hybrid 1 $\overset{c}{\approx}$ Hybrid 2.

ZKP for Graph 3-Coloring



This hybrid is the simulator S.

Claim: Hybrid 1 $\overset{c}{\approx}$ Hybrid 2.

Distribution of $\{\text{color}_i\}_i$ now depends on (i',j').

ZKP for Graph 3-Coloring

Hybrid 2

If $(i,j) \neq (i',j')$

- $\psi \xleftarrow{\$} \mathsf{Perm}_3$
- $(i',j') \xleftarrow{\$} E$
- $\mathsf{color}_{i'} \xleftarrow{\$} \{1,2,3\}$ and $\mathsf{color}_{j'} \xleftarrow{\$} \{1,2,3\} \backslash \{\mathsf{color}_{i'}\}$.
  - $\forall i \in \{1,\ldots,n\} \backslash \{i',j'\}$, $\mathsf{color}_i := 1$.
- $\forall i$, $c_i := \mathsf{Com}(\mathsf{color}_i)$
- $(i,j) \leftarrow \widehat{V}(c_1,\ldots,c_n)$

Output

$(c_1,\ldots,c_n)$ (i,j) Openings for c_i and c_j

This hybrid is the simulator S.

Claim: Hybrid 1 $\overset{c}{\approx}$ Hybrid 2.

Distribution of $\{\mathsf{color}_i\}_i$ now depends on (i',j').

However, by hiding property of commitment scheme, $\{c_i\}_i$ is computationally indistinguishable across hybrids.

ZKP for Graph 3-Coloring

Hybrid 2

If $(i,j) \neq (i',j')$

- $\psi \xleftarrow{\$} \text{Perm}_3$
- $(i',j') \xleftarrow{\$} E$
- $\text{color}_{i'} \xleftarrow{\$} \{1,2,3\}$ and $\text{color}_{j'} \xleftarrow{\$} \{1,2,3\} \backslash \{\text{color}_{i'}\}$.
  - $\forall i \in \{1,\ldots,n\} \backslash \{i',j'\}$, $\text{color}_i := 1$.
- $\forall i$, $c_i := \text{Com}(\text{color}_i)$
- $(i,j) \leftarrow \widehat{V}(c_1,\ldots,c_n)$

Output

$(c_1,\ldots,c_n)$ (i,j) Openings for c_i and c_j

This hybrid is the simulator S.

Claim: Hybrid 1 $\overset{c}{\approx}$ Hybrid 2.

Distribution of $\{\text{color}_i\}_i$ now depends on (i',j').

However, by hiding property of commitment scheme, $\{c_i\}_i$ is computationally indistinguishable across hybrids.

$\Longrightarrow$ (i,j) is computationally indistinguishable across hybrids.

Probability that $(i,j) = (i',j')$ is negligibly close across hybrids.

ZKP for Graph 3-Coloring

Hybrid 2

If $(i,j) \neq (i',j')$

- $\psi \xleftarrow{\$} \text{Perm}_3$
- $(i',j') \xleftarrow{\$} E$
- $\text{color}_{i'} \xleftarrow{\$} \{1,2,3\}$ and $\text{color}_{j'} \xleftarrow{\$} \{1,2,3\} \backslash \{\text{color}_{i'}\}$.
  - $\forall i \in \{1,\ldots,n\} \backslash \{i',j'\}$, $\text{color}_i := 1$.
- $\forall i$, $c_i := \text{Com}(\text{color}_i)$
- $(i,j) \leftarrow \widehat{V}(c_1,\ldots,c_n)$

Output

$(c_1,\ldots,c_n)$ (i,j) Openings for c_i and c_j

This hybrid is the simulator S.

Claim: Hybrid 1 $\overset{c}{\approx}$ Hybrid 2.

Distribution of $\{\text{color}_i\}_i$ now depends on (i',j').

However, by hiding property of commitment scheme, $\{c_i\}_i$ is computationally indistinguishable across hybrids.

$\Longrightarrow$ (i,j) is computationally indistinguishable across hybrids.

Probability that $(i,j) = (i',j')$ is negligibly close across hybrids.

This hybrid no longer requires ϕ.

ZKP for Graph 3-Coloring

View in Real Execution

- $\psi \xleftarrow{\$} \text{Perm}_3$
- $\forall i, \text{color}_i := \psi\left(\phi(v_i)\right)$
- $\forall i, c_i := \text{Com}(\text{color}_i)$
- $(i, j) \leftarrow \widehat{V}(c_1, \ldots, c_n)$

Output

$(c_1, \ldots, c_n)$ (i, j) Openings for c_i and c_j

$\overset{c}{\approx}$

Simulated View

If $(i, j) \neq (i', j')$

- $\psi \xleftarrow{\$} \text{Perm}_3$
- $(i', j') \xleftarrow{\$} E$
- $\text{color}_{i'} \xleftarrow{\$} \{1, 2, 3\}$ and $\text{color}_{j'} \xleftarrow{\$} \{1, 2, 3\} \backslash \{\text{color}_{i'}\}$.
  - $\forall i \in \{1, \ldots, n\} \backslash \{i', j'\}, \text{color}_i := 1.$
- $\forall i, c_i := \text{Com}(\text{color}_i)$
- $(i, j) \leftarrow \widehat{V}(c_1, \ldots, c_n)$

Output

$(c_1, \ldots, c_n)$ (i, j) Openings for c_i and c_j

Round Complexity of ZKP

Statement: $G = (V, E)$ where $|V| = n = \text{poly}(\lambda)$.

Prover's NP witness: Color assignment $\phi : V \to \{1, 2, 3\}$.

ZKP for Graph 3-Coloring

Repeat the following procedure $n|E|$ times.

- $P \to V$: P samples $\psi \xleftarrow{\$} \text{Perm}_3$ uniformly at random. For each vertex $v_i \in V$ it computes $\text{color}_i := \psi\left(\phi(v_i)\right)$ and

  $$c_i := \text{Com}(\text{color}_i)$$

  and sends $(c_1, \ldots, c_n)$ to V.

- $V \to P$: V samples a random edge (i, j) and sends it to P.

- $P \to V$: P opens c_i and c_j to reveal color_i and color_j.

- V: If the openings are valid and $\text{color}_i \neq \text{color}_j$, then V continues to the next iteration. Else, it rejects.

If V does not reject in any iteration, it accepts the proof.

Requires $3 \cdot n|E|$ rounds.

Round Complexity of ZKP

Statement: $G = (V, E)$ where $|V| = n = \text{poly}(\lambda)$.

Prover's NP witness: Color assignment $\phi : V \to \{1, 2, 3\}$.

ZKP for Graph 3-Coloring

Repeat the following procedure $n|E|$ times.

- $P \to V$: P samples $\psi \xleftarrow{\$} \text{Perm}_3$ uniformly at random. For each vertex $v_i \in V$ it computes $\text{color}_i := \psi\left(\phi(v_i)\right)$ and

  $$c_i := \text{Com}(\text{color}_i)$$

  and sends $(c_1, \ldots, c_n)$ to V.

- $V \to P$: V samples a random edge (i, j) and sends it to P.

- $P \to V$: P opens c_i and c_j to reveal color_i and color_j.

- V: If the openings are valid and $\text{color}_i \neq \text{color}_j$, then V continues to the next iteration. Else, it rejects.

If V does not reject in any iteration, it accepts the proof.

Requires $3 \cdot n|E|$ rounds.

Can we run all repetitions in parallel to get a 3-round ZKP?

Round Complexity of ZKP

Statement: $G = (V, E)$ where $|V| = n = \text{poly}(\lambda)$.

Prover's NP witness: Color assignment $\phi : V \to \{1, 2, 3\}$.

ZKP for Graph 3-Coloring

Repeat the following procedure $n|E|$ times.

- $P \to V$: P samples $\psi \xleftarrow{\$} \text{Perm}_3$ uniformly at random. For each vertex $v_i \in V$ it computes $\text{color}_i := \psi\left(\phi(v_i)\right)$ and

$$c_i := \text{Com}(\text{color}_i)$$

  and sends $(c_1, \ldots, c_n)$ to V.

- $V \to P$: V samples a random edge (i, j) and sends it to P.

- $P \to V$: P opens c_i and c_j to reveal color_i and color_j.

- V: If the openings are valid and $\text{color}_i \neq \text{color}_j$, then V continues to the next iteration. Else, it rejects.

If V does not reject in any iteration, it accepts the proof.

Requires $3 \cdot n|E|$ rounds.

Can we run all repetitions in parallel to get a 3-round ZKP?

Might not preserve ZK in general!

Theorem [Goldreich-Krawczyk'90]**:** There exist ZKPs whose parallel repetition is not zero knowledge.

Round Complexity of ZKP

Theorem [Goldreich-Krawczyk'90]**:** There exist ZKPs whose parallel repetition is not zero knowledge.

Round Complexity of ZKP

Theorem [Goldreich-Krawczyk'90]**:** There exist ZKPs whose parallel repetition is not zero knowledge.

While amplifying soundness by parallel repetition might not preserve ZK, we can still construct constant-round ZKP using different techniques.

Theorem [Goldreich-Kahan'95]**:** There is a constant-round ZKP for graph 3-coloring assuming the existence of collision resistance hash functions.

Round Complexity of ZKP

Theorem [Goldreich-Krawczyk'90]**:** There exist ZKPs whose parallel repetition is not zero knowledge.

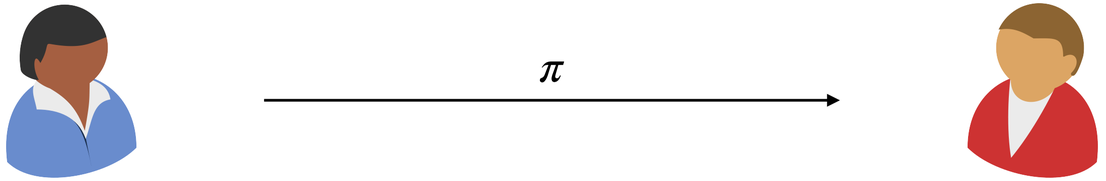
While amplifying soundness by parallel repetition might not preserve ZK, we can still construct constant-round ZKP using different techniques.

Theorem [Goldreich-Kahan'95]**:** There is a constant-round ZKP for graph 3-coloring assuming the existence of collision resistance hash functions.

Can we construct a one-message ZKP?

Non-Interactive Zero-Knowledge Proofs

$x \in L$

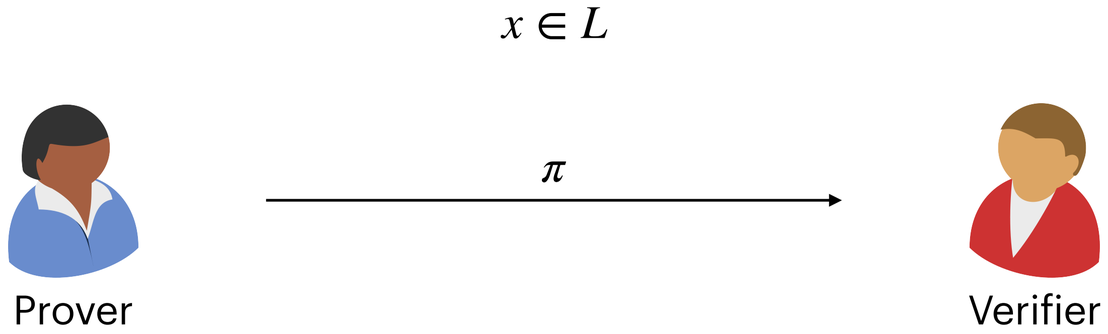


Prover

Verifier

We want completeness, soundness and zero-knowledge.

Non-Interactive Zero-Knowledge Proofs

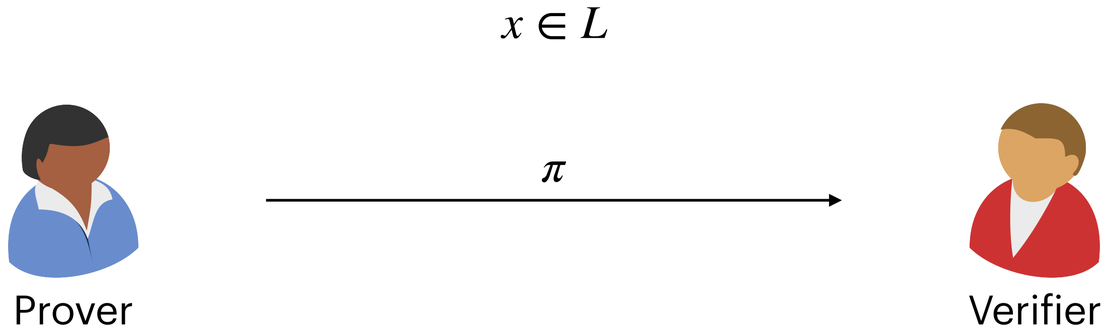


We want completeness, soundness and zero-knowledge.

Interaction is necessary for ZK!

Theorem: If a language L has a one-message ZKP, then $L \in$ BPP.

Non-Interactive Zero-Knowledge Proofs



We want completeness, soundness and zero-knowledge.

Interaction is necessary for ZK!

Theorem: If a language L has a one-message ZKP, then $L \in$ BPP.

Only languages that can be decided by a PPT algorithm have one-message ZKP.

Non-Interactive Zero-Knowledge Proofs

Theorem: If a language L has a one-message ZKP, then $L \in$ BPP.

Non-Interactive Zero-Knowledge Proofs

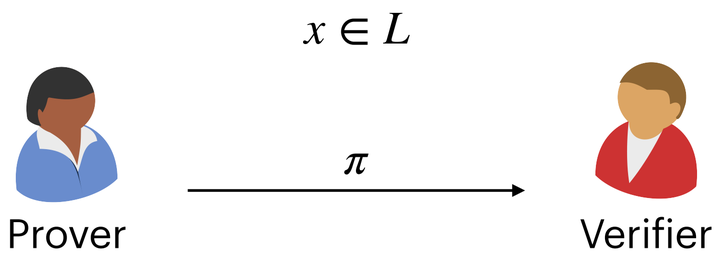
Theorem: If a language L has a one-message ZKP, then $L \in$ BPP.

Proof. Assume for the sake of contradiction there exists a one-message ZKP for $L \notin$ BPP.

Non-Interactive Zero-Knowledge Proofs

Theorem: If a language L has a one-message ZKP, then $L \in$ BPP.

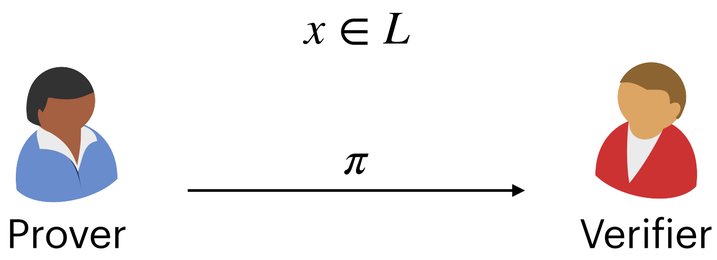
Proof. Assume for the sake of contradiction there exists a one-message ZKP for $L \notin$ BPP.



Non-Interactive Zero-Knowledge Proofs

Theorem: If a language L has a one-message ZKP, then $L \in$ BPP.

Proof. Assume for the sake of contradiction there exists a one-message ZKP for $L \notin$ BPP.

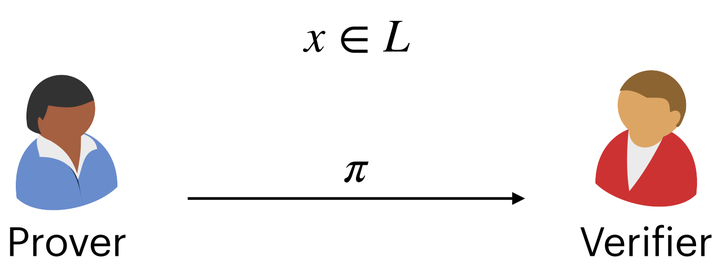


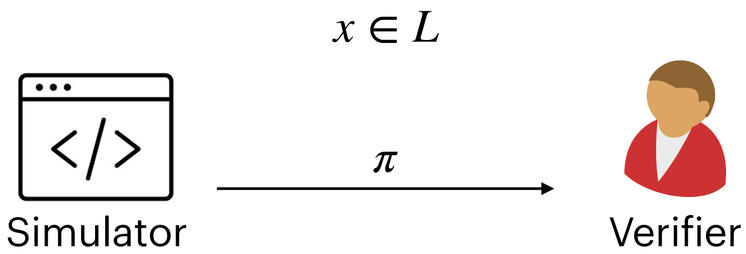
Verifier accepts π when $x \in L$ (completeness).

Non-Interactive Zero-Knowledge Proofs

Theorem: If a language L has a one-message ZKP, then $L \in$ BPP.

Proof. Assume for the sake of contradiction there exists a one-message ZKP for $L \notin$ BPP.



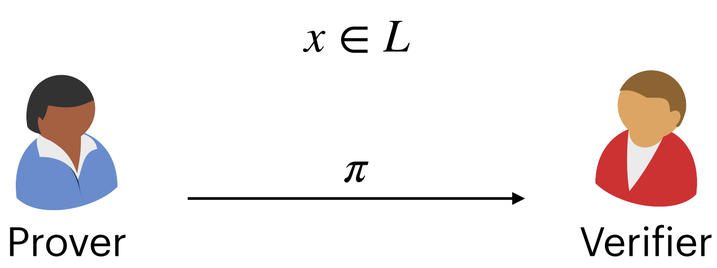


Verifier accepts π when $x \in L$ (completeness).

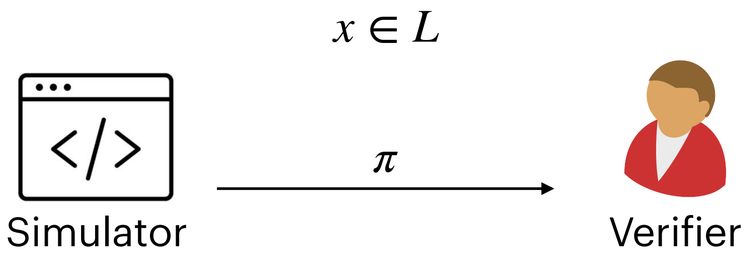
Non-Interactive Zero-Knowledge Proofs

Theorem: If a language L has a one-message ZKP, then $L \in$ BPP.

Proof. Assume for the sake of contradiction there exists a one-message ZKP for $L \notin$ BPP.



Verifier accepts π when $x \in L$ (completeness).

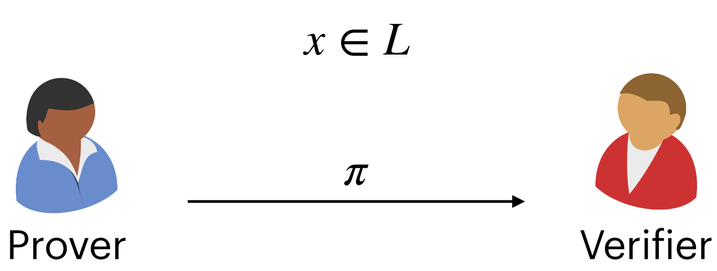


Verifier accepts π when $x \in L$ (zero-knowledge).

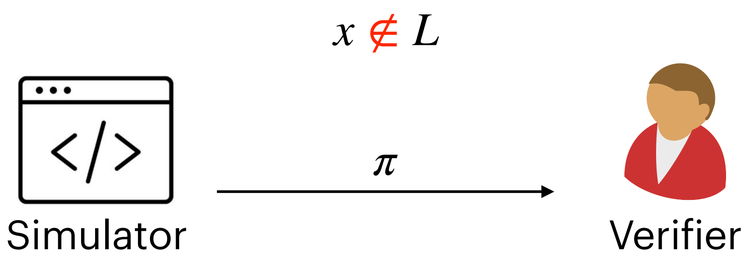
Non-Interactive Zero-Knowledge Proofs

Theorem: If a language L has a one-message ZKP, then $L \in$ BPP.

Proof. Assume for the sake of contradiction there exists a one-message ZKP for $L \notin$ BPP.



Verifier accepts π when $x \in L$ (completeness).

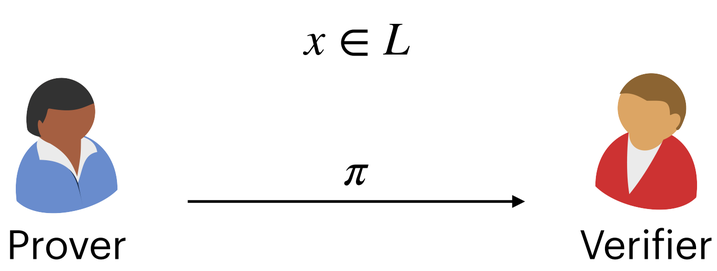


Claim: Verifier accepts π.

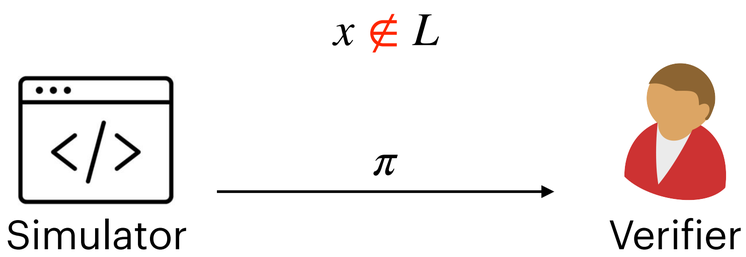
Non-Interactive Zero-Knowledge Proofs

Theorem: If a language L has a one-message ZKP, then $L \in$ BPP.

Proof. Assume for the sake of contradiction there exists a one-message ZKP for $L \notin$ BPP.



Verifier accepts π when $x \in L$ (completeness).



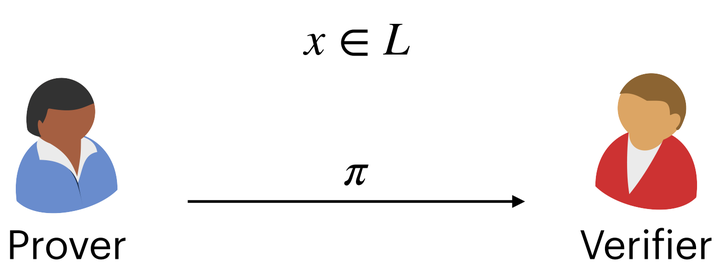
Claim: Verifier accepts π.

Why?

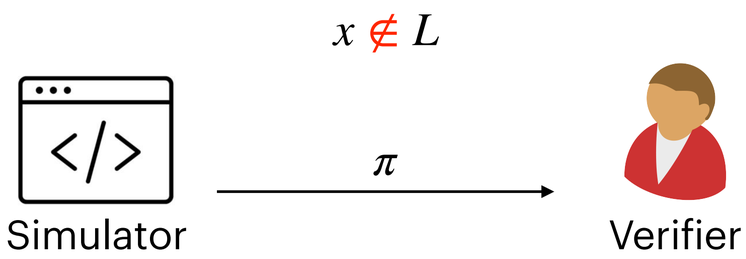
Non-Interactive Zero-Knowledge Proofs

Theorem: If a language L has a one-message ZKP, then $L \in$ BPP.

Proof. Assume for the sake of contradiction there exists a one-message ZKP for $L \notin$ BPP.



Verifier accepts π when $x \in L$ (completeness).



Claim: Verifier accepts π.

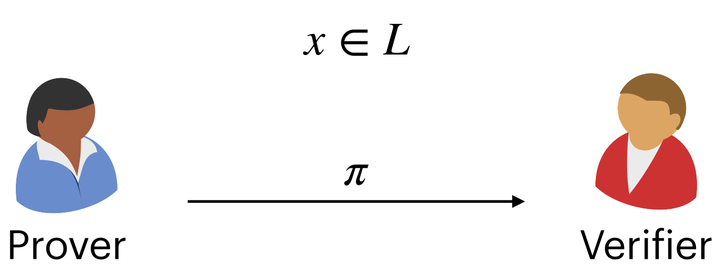
Why?

S and V are PPT algorithms.

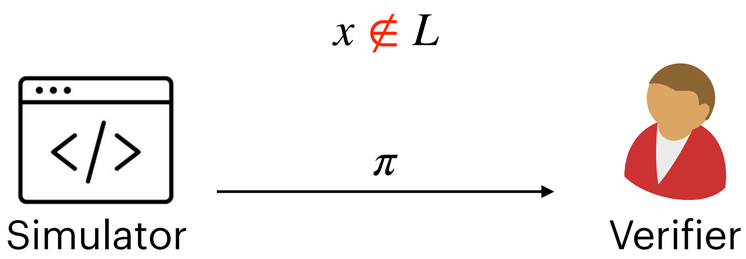
Non-Interactive Zero-Knowledge Proofs

Theorem: If a language L has a one-message ZKP, then $L \in$ BPP.

Proof. Assume for the sake of contradiction there exists a one-message ZKP for $L \notin$ BPP.



Verifier accepts π when $x \in L$ (completeness).



Claim: Verifier accepts π.

Why?

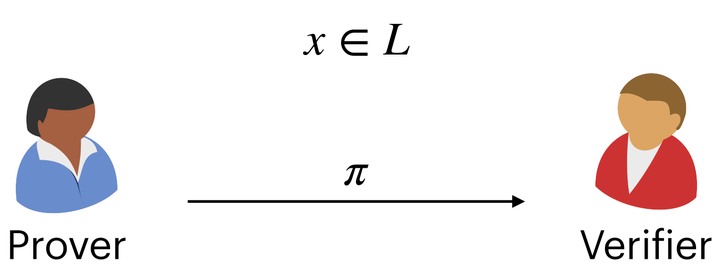
S and V are PPT algorithms.

If $L \notin$ BPP then neither S nor V can decide if $x \in L$.

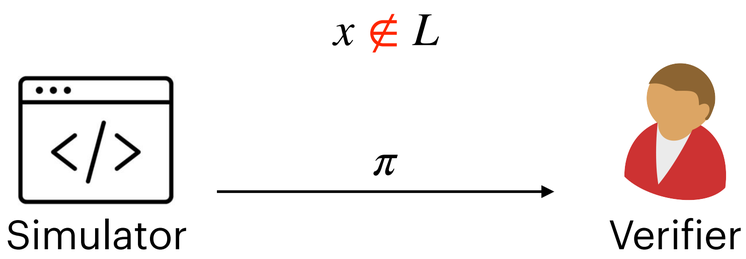
Non-Interactive Zero-Knowledge Proofs

Theorem: If a language L has a one-message ZKP, then $L \in$ BPP.

Proof. Assume for the sake of contradiction there exists a one-message ZKP for $L \notin$ BPP.



Verifier accepts π when $x \in L$ (completeness).



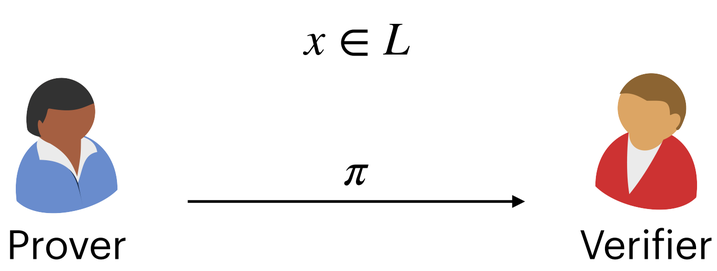
Claim: Verifier accepts π.

But the simulator is proving false statements!

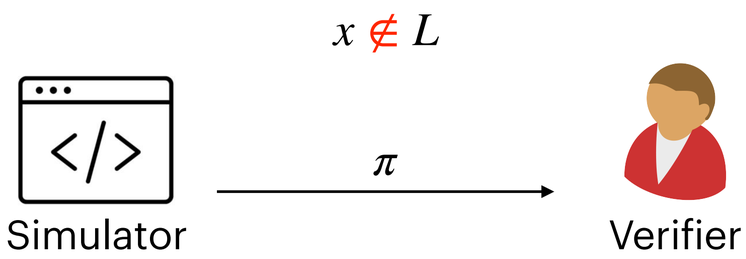
Non-Interactive Zero-Knowledge Proofs

Theorem: If a language L has a one-message ZKP, then $L \in$ BPP.

Proof. Assume for the sake of contradiction there exists a one-message ZKP for $L \notin$ BPP.



Verifier accepts π when $x \in L$ (completeness).



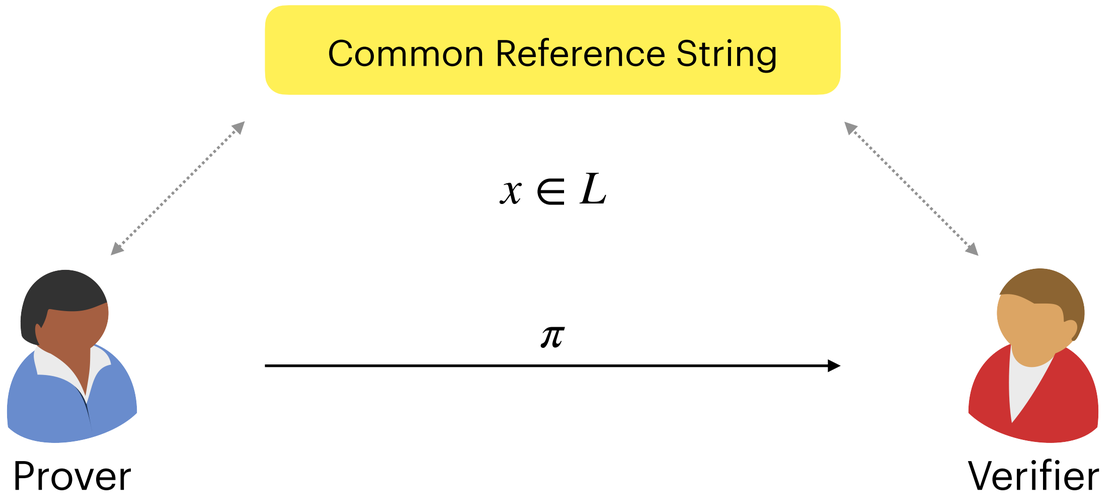
Claim: Verifier accepts π.

But the simulator is proving false statements!

The proof system is not sound!

Non-Interactive Zero-Knowledge Proofs

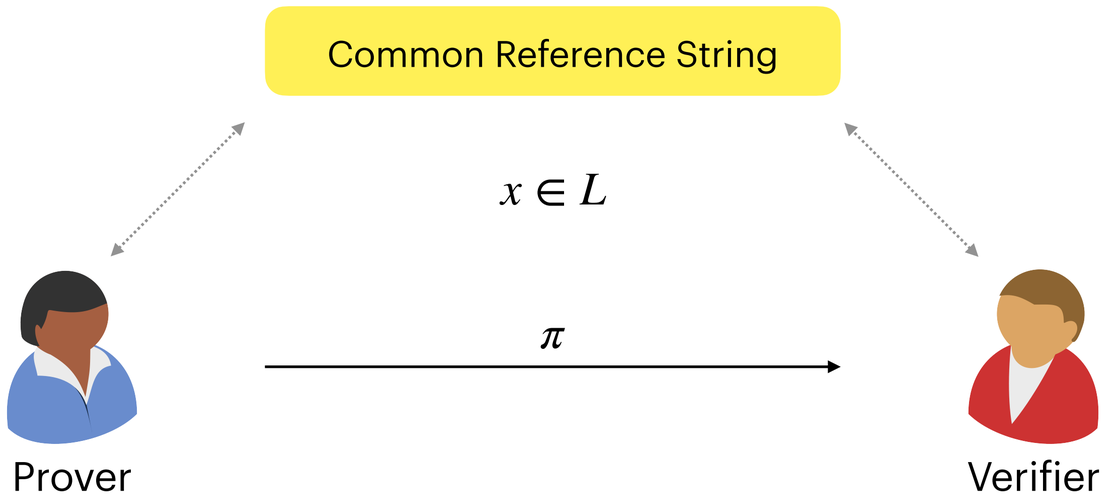
Define NIZKs in the common reference string model.



We want completeness, soundness and zero-knowledge.

Non-Interactive Zero-Knowledge Proofs

Define NIZKs in the common reference string model.

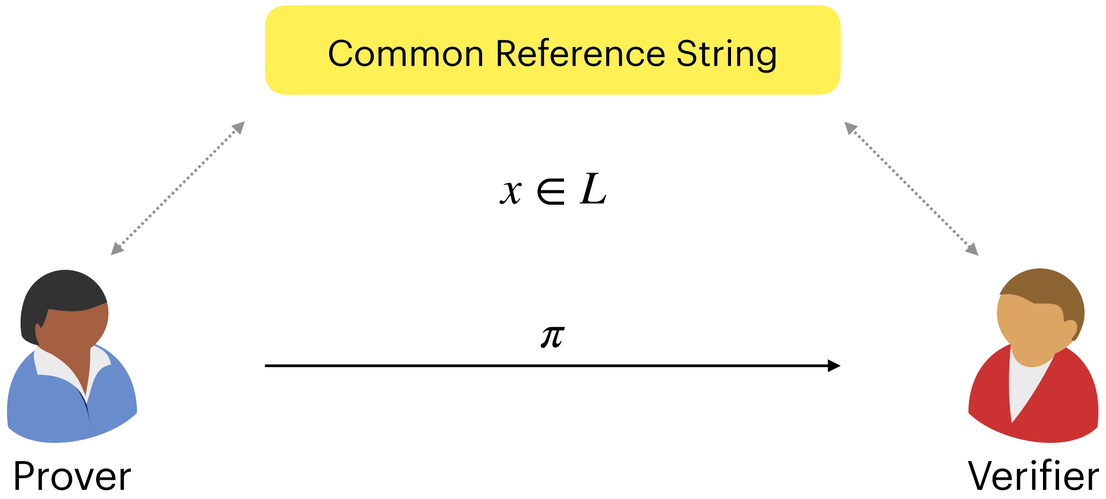


- Two approaches
    1. NIZKs for NP in the standard model assuming hardness of factoring, DDH, LWE etc.,

We want completeness, soundness and zero-knowledge.

Non-Interactive Zero-Knowledge Proofs

Define NIZKs in the common reference string model.

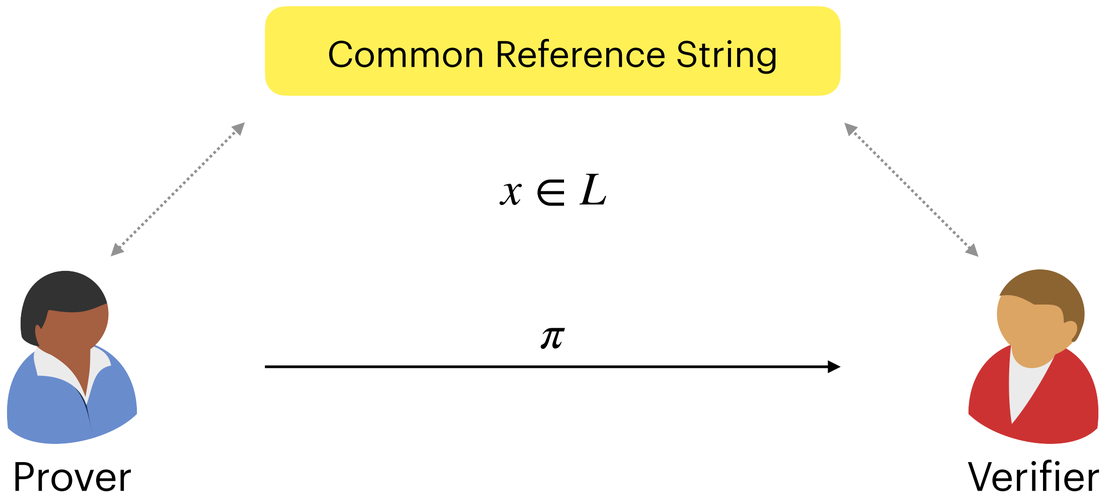


- Two approaches
  1. NIZKs for NP in the standard model assuming hardness of factoring, DDH, LWE etc.,
  2. NIZKs in the random oracle model.

We want completeness, soundness and zero-knowledge.

Non-Interactive Zero-Knowledge Proofs

Define NIZKs in the common reference string model.

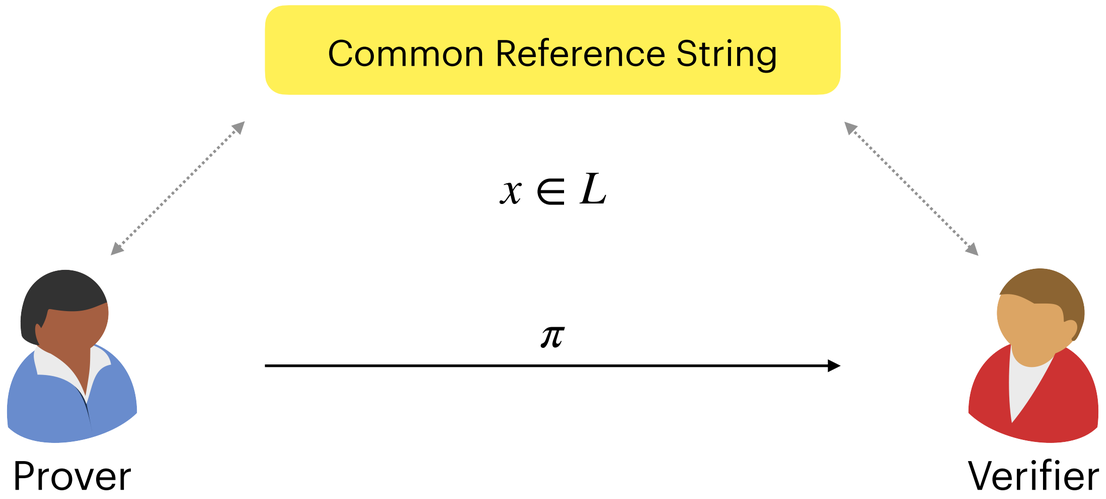


- Two approaches
  1. NIZKs for NP in the standard model assuming hardness of factoring, DDH, LWE etc.,
  2. NIZKs in the random oracle model.
     - CRS describes a hash function.

We want completeness, soundness and zero-knowledge.

Non-Interactive Zero-Knowledge Proofs

Define NIZKs in the common reference string model.

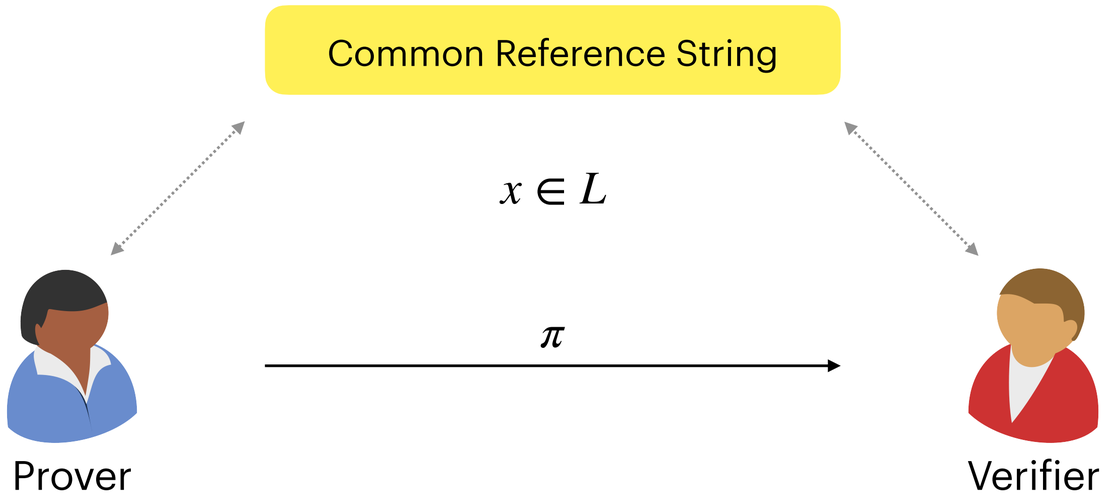


- Two approaches
  1. NIZKs for NP in the standard model assuming hardness of factoring, DDH, LWE etc.,
  2. NIZKs in the random oracle model.
     - CRS describes a hash function.
     - Assume the hash function is a random oracle (heuristic!)

We want completeness, soundness and zero-knowledge.

Non-Interactive Zero-Knowledge Proofs

Define NIZKs in the common reference string model.



We want completeness, soundness and zero-knowledge.

- Two approaches
  1. NIZKs for NP in the standard model assuming hardness of factoring, DDH, LWE etc.,
  2. NIZKs in the random oracle model.
     - CRS describes a hash function.
     - Assume the hash function is a random oracle (heuristic!)
- **Fiat-Shamir Transform:** Converting any public-coin ZKP into a NIZK in the random oracle model.
  - In a public-coin ZKP, all verifier messages are uniformly random.

The Fiat-Shamir Transform

Applied to ZKP for Graph 3-Coloring

Statement $x : G = (V, E)$

Prover

Verifier

Let $\phi : V \to \{1, 2, 3\}$ be a 3-coloring of G.

$\psi \xleftarrow{\$} \mathsf{Perm}_3$.

$\forall i \in \{1, \ldots, n\}$:

$\mathsf{color}_i := \psi\left(\phi(v_i)\right)$

$c_i := \mathsf{Com}\left(\mathsf{color}_i\right)$

$c_1, \ldots, c_n$ →

Sample a random edge $(i, j) \xleftarrow{\$} E$.

← (i, j)

Openings for c_i, c_j →

Check if openings are valid and if $\mathsf{color}_i \neq \mathsf{color}_j$.

The Fiat-Shamir Transform

Applied to ZKP for Graph 3-Coloring

Statement $x : G = (V, E)$

Prover

Verifier

Let $\phi : V \to \{1, 2, 3\}$ be a 3-coloring of G.

$\psi \xleftarrow{\$} \mathsf{Perm}_3$.

$\forall i \in \{1, \ldots, n\}$:

$\mathsf{color}_i := \psi\left(\phi(v_i)\right)$

$c_i := \mathsf{Com}\left(\mathsf{color}_i\right)$

$c_1, \ldots, c_n$ →

Sample a random edge $(i, j) \xleftarrow{\$} E$.

← (i, j)

Openings for c_i, c_j →

Check if openings are valid and if $\mathsf{color}_i \neq \mathsf{color}_j$.

The Fiat-Shamir Transform

Applied to ZKP for Graph 3-Coloring

Statement $x : G = (V, E)$

Prover

Verifier

This ZKP is public-coin!

Let $\phi : V \to \{1, 2, 3\}$ be a 3-coloring of G.

$\psi \xleftarrow{\$} \mathsf{Perm}_3$.

$\forall i \in \{1, \ldots, n\}$:

$\mathsf{color}_i := \psi\left(\phi(v_i)\right)$

$c_i := \mathsf{Com}\left(\mathsf{color}_i\right)$

$c_1, \ldots, c_n$ →

Sample a random edge $(i, j) \xleftarrow{\$} E$.

← (i, j)

Openings for c_i, c_j →

Check if openings are valid and if $\mathsf{color}_i \neq \mathsf{color}_j$.

The Fiat-Shamir Transform

Applied to ZKP for Graph 3-Coloring

Statement $x : G = (V, E)$

Prover

Verifier

This ZKP is public-coin!

The verifier's samples a random edge after the prover has committed to a coloring.

Let $\phi : V \to \{1, 2, 3\}$ be a 3-coloring of G.

$\psi \xleftarrow{\$} \mathsf{Perm}_3$.

$\forall i \in \{1, \ldots, n\}$:

$\mathsf{color}_i := \psi\left(\phi(v_i)\right)$

$c_i := \mathsf{Com}\left(\mathsf{color}_i\right)$

$c_1, \ldots, c_n$ →

Sample a random edge $(i, j) \xleftarrow{\$} E$.

← (i, j)

Openings for c_i, c_j →

Check if openings are valid and if $\mathsf{color}_i \neq \mathsf{color}_j$.

The Fiat-Shamir Transform

Applied to ZKP for Graph 3-Coloring

Statement $x : G = (V, E)$

Prover

Verifier

This ZKP is public-coin!

The verifier's samples a random edge after the prover has committed to a coloring.

Let $\phi : V \to \{1, 2, 3\}$ be a 3-coloring of G.

$\psi \xleftarrow{\$} \mathsf{Perm}_3$.

$\forall i \in \{1, \ldots, n\}$:

$\mathsf{color}_i := \psi\left(\phi(v_i)\right)$

$c_i := \mathsf{Com}\left(\mathsf{color}_i\right)$

$(i, j) := H(x, c_1, \ldots, c_n)$

The Fiat-Shamir Transform

Applied to ZKP for Graph 3-Coloring

Statement $x : G = (V, E)$

Prover

Verifier

This ZKP is public-coin!

The verifier's samples a random edge after the prover has committed to a coloring.

Let $\phi : V \to \{1, 2, 3\}$ be a 3-coloring of G.

$\psi \xleftarrow{\$} \mathsf{Perm}_3$.

$\forall i \in \{1, \ldots, n\}$:

$\mathsf{color}_i := \psi\left(\phi(v_i)\right)$

$c_i := \mathsf{Com}\left(\mathsf{color}_i\right)$

$(i, j) := H(x, c_1, \ldots, c_n)$

If the hash function is like a "random function", it outputs uniformly random (i, j) based on the prover's first message.

The Fiat-Shamir Transform

Applied to ZKP for Graph 3-Coloring

Statement $x : G = (V, E)$

Prover

Verifier

This ZKP is public-coin!

The verifier's samples a random edge after the prover has committed to a coloring.

Let $\phi : V \to \{1, 2, 3\}$ be a 3-coloring of G.

$\psi \xleftarrow{\$} \text{Perm}_3$.

$\forall i \in \{1, \ldots, n\}$:

$\text{color}_i := \psi\left(\phi(v_i)\right)$

$c_i := \text{Com}\left(\text{color}_i\right)$

$$\pi = \begin{pmatrix} c_1, \ldots, c_n \\ \text{opening for } c_i, c_j \end{pmatrix} \longrightarrow$$

$(i, j) := H(x, c_1, \ldots, c_n)$

If the hash function is like a "random function", it outputs uniformly random (i, j) based on the prover's first message.

The Fiat-Shamir Transform

Applied to ZKP for Graph 3-Coloring

Statement $x : G = (V, E)$

Prover

Verifier

This ZKP is public-coin!

The verifier's samples a random edge after the prover has committed to a coloring.

Let $\phi : V \to \{1, 2, 3\}$ be a 3-coloring of G.

$\psi \xleftarrow{\$} \text{Perm}_3$.

$\forall i \in \{1, \ldots, n\}$:

$\text{color}_i := \psi\left(\phi(v_i)\right)$

$c_i := \text{Com}\left(\text{color}_i\right)$

$$\pi = \begin{pmatrix} c_1, \ldots, c_n \\ \text{opening for } c_i, c_j \end{pmatrix} \longrightarrow$$

$(i, j) := H(x, c_1, \ldots, c_n)$

$(i, j) := H(x, c_1, \ldots, c_n)$

If the hash function is like a "random function", it outputs uniformly random (i, j) based on the prover's first message.

The Fiat-Shamir Transform

Applied to ZKP for Graph 3-Coloring

Statement $x : G = (V, E)$

Prover

Verifier

This ZKP is public-coin!

The verifier's samples a random edge after the prover has committed to a coloring.

Let $\phi : V \to \{1, 2, 3\}$ be a 3-coloring of G.

$\psi \xleftarrow{\$} \text{Perm}_3$.

$\forall i \in \{1, \ldots, n\}$:

$\text{color}_i := \psi\left(\phi(v_i)\right)$

$c_i := \text{Com}\left(\text{color}_i\right)$

$\pi = \left(\begin{matrix} c_1, \ldots, c_n \\ \text{opening for } c_i, c_j \end{matrix}\right)$

$(i, j) := H(x, c_1, \ldots, c_n)$

$(i, j) := H(x, c_1, \ldots, c_n)$

Check if openings are valid and if $\text{color}_i \neq \text{color}_j$.

If the hash function is like a "random function", it outputs uniformly random (i, j) based on the prover's first message.

The Fiat-Shamir Transform

Applied to ZKP for Graph 3-Coloring

Statement $x : G = (V, E)$

Prover

Verifier

Let $\phi : V \to \{1, 2, 3\}$ be a 3-coloring of G.

$\psi \xleftarrow{\$} \mathsf{Perm}_3$.

$\forall i \in \{1, \ldots, n\}$:

$\mathsf{color}_i := \psi\left(\phi(v_i)\right)$

$c_i := \mathsf{Com}\left(\mathsf{color}_i\right)$

$(i, j) := H(x, c_1, \ldots, c_n)$

$\pi = \left(\begin{matrix} c_1, \ldots, c_n \\ \text{opening for } c_i, c_j \end{matrix}\right)$

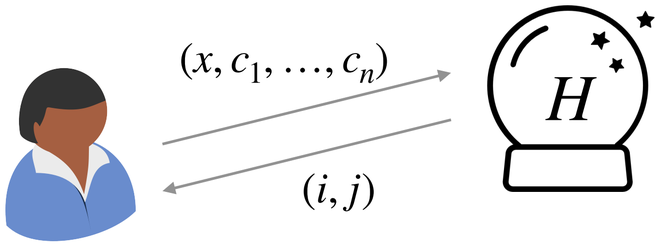
$(i, j) := H(x, c_1, \ldots, c_n)$

Check if openings are valid and if $\mathsf{color}_i \neq \mathsf{color}_j$.

The Fiat-Shamir Transform

Applied to ZKP for Graph 3-Coloring

Statement $x : G = (V, E)$



Prover

Verifier

Let $\phi : V \to \{1, 2, 3\}$ be a 3-coloring of G.

$\psi \xleftarrow{\$} \mathsf{Perm}_3$.

$\forall i \in \{1,\ldots,n\}$:

$\mathsf{color}_i := \psi\left(\phi(v_i)\right)$

$c_i := \mathsf{Com}\left(\mathsf{color}_i\right)$

$$\pi = \left(\begin{matrix} c_1, \ldots, c_n \\ \text{opening for } c_i, c_j \end{matrix}\right) \longrightarrow$$

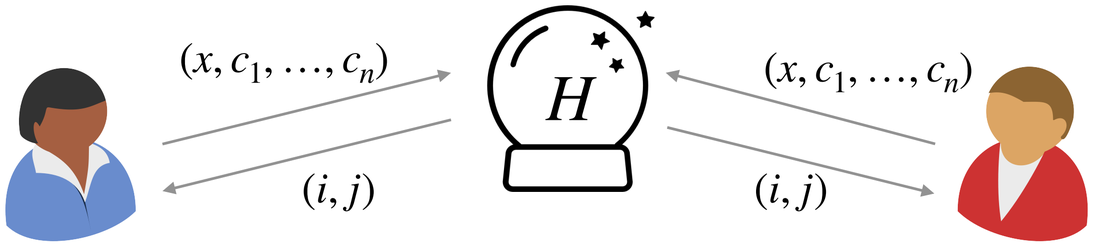
$(i, j) := H(x, c_1, \ldots, c_n)$

Check if openings are valid and if $\mathsf{color}_i \neq \mathsf{color}_j$.

The Fiat-Shamir Transform

Applied to ZKP for Graph 3-Coloring

Statement $x : G = (V, E)$



Let $\phi : V \to \{1, 2, 3\}$ be a 3-coloring of G.

$\psi \xleftarrow{\$} \mathsf{Perm}_3$.

$\forall i \in \{1,\ldots,n\}$:

$\mathsf{color}_i := \psi\left(\phi(v_i)\right)$

$c_i := \mathsf{Com}\left(\mathsf{color}_i\right)$

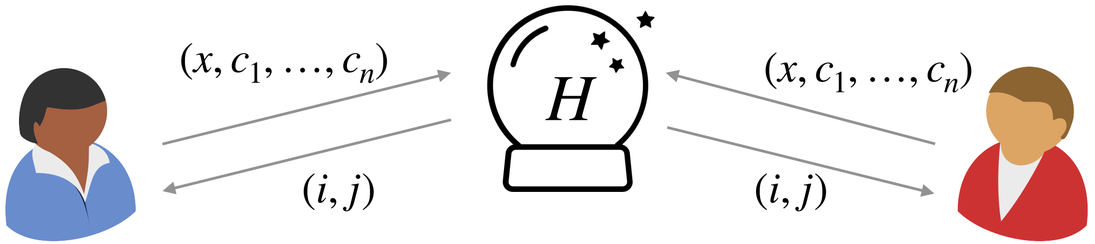
$$\pi = \left(\begin{matrix} c_1, \ldots, c_n \\ \text{opening for } c_i, c_j \end{matrix}\right) \longrightarrow$$

Check if openings are valid and if $\mathsf{color}_i \neq \mathsf{color}_j$.

The Fiat-Shamir Transform

Applied to ZKP for Graph 3-Coloring

Statement $x : G = (V, E)$



Let $\phi : V \to \{1, 2, 3\}$ be a 3-coloring of G.

$\psi \xleftarrow{\$} \mathsf{Perm}_3$.

$\forall i \in \{1, \ldots, n\}$:

$\mathsf{color}_i := \psi\left(\phi(v_i)\right)$

$c_i := \mathsf{Com}\left(\mathsf{color}_i\right)$

$$\pi = \left(\begin{matrix} c_1, \ldots, c_n \\ \text{opening for } c_i, c_j \end{matrix}\right) \longrightarrow$$

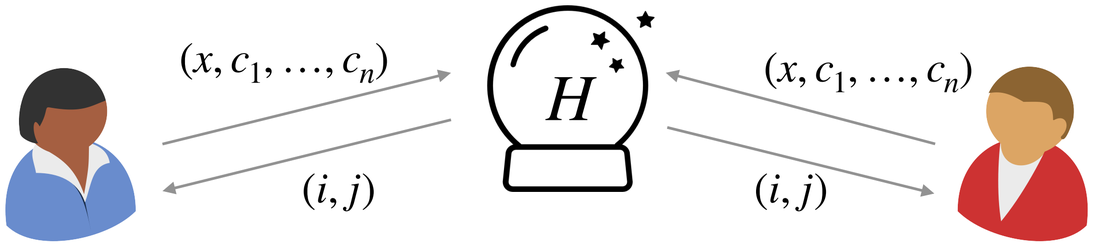
Check if openings are valid and if $\mathsf{color}_i \neq \mathsf{color}_j$.

The random oracle is like an honest verifier. It always returns uniformly random edges.

The Fiat-Shamir Transform

Applied to ZKP for Graph 3-Coloring

Statement $x : G = (V, E)$



- Is this protocol sound?

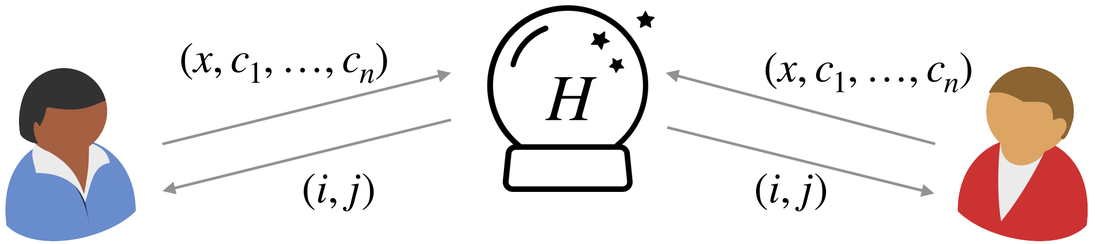
$$\pi = \begin{pmatrix} c_1, \ldots, c_n \\ \text{opening for } c_i, c_j \end{pmatrix} \longrightarrow$$

Check if openings are valid and if $\text{color}_i \neq \text{color}_j$.

The Fiat-Shamir Transform

Applied to ZKP for Graph 3-Coloring

Statement $x : G = (V, E)$



- Is this protocol sound?
- No! Malicious prover can always break soundness.

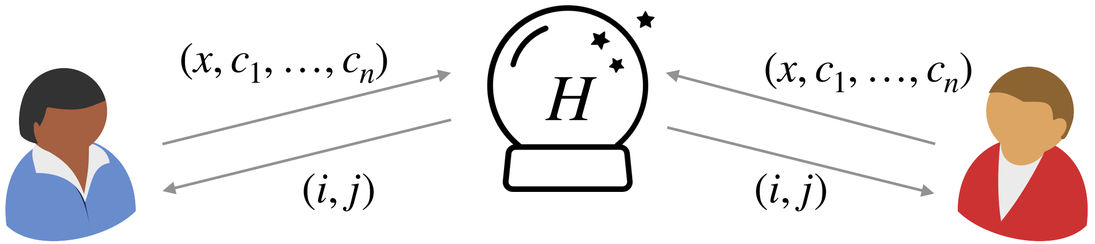
$$\pi = \begin{pmatrix} c_1, \ldots, c_n \\ \text{opening for } c_i, c_j \end{pmatrix} \longrightarrow$$

Check if openings are valid and if $\text{color}_i \neq \text{color}_j$.

The Fiat-Shamir Transform

Applied to ZKP for Graph 3-Coloring

Statement $x : G = (V, E)$



- Is this protocol sound?
- No! Malicious prover can always break soundness.
  - Consider simulator's strategy from interactive case.

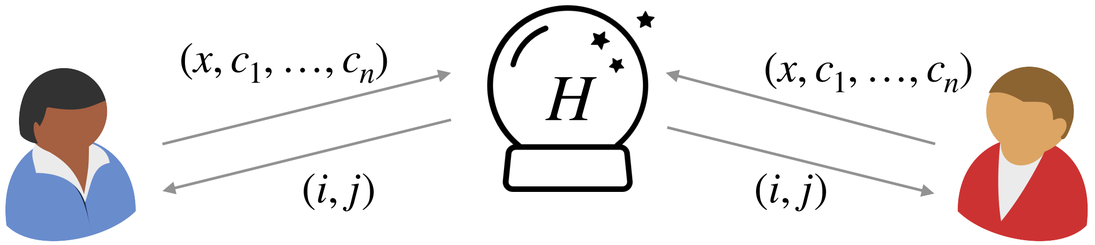
$$\pi = \begin{pmatrix} c_1, \ldots, c_n \\ \text{opening for } c_i, c_j \end{pmatrix} \longrightarrow$$

Check if openings are valid and if $\text{color}_i \neq \text{color}_j$.

The Fiat-Shamir Transform

Applied to ZKP for Graph 3-Coloring

Statement $x : G = (V, E)$



- Is this protocol sound?
- No! Malicious prover can always break soundness.
  - Consider simulator's strategy from interactive case.
- How can we amplify soundness?

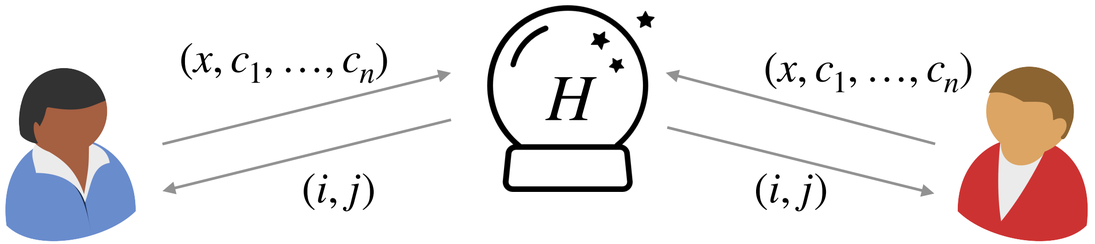
$$\pi = \begin{pmatrix} c_1, \ldots, c_n \\ \text{opening for } c_i, c_j \end{pmatrix} \longrightarrow$$

Check if openings are valid and if $\text{color}_i \neq \text{color}_j$.

The Fiat-Shamir Transform

Applied to ZKP for Graph 3-Coloring

Statement $x : G = (V, E)$



- Is this protocol sound?
- No! Malicious prover can always break soundness.
  - Consider simulator's strategy from interactive case.
- How can we amplify soundness?
  - Can we use sequential repetition?

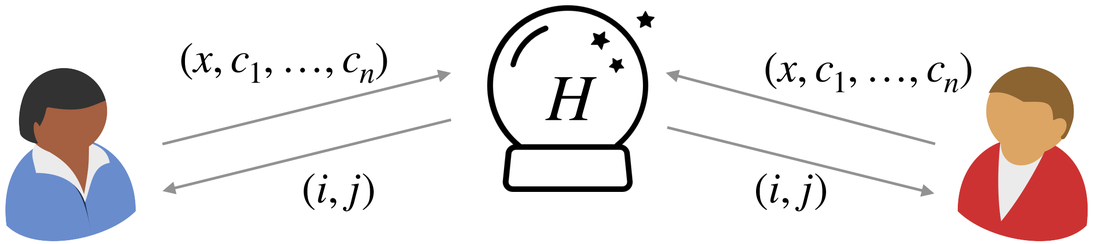
$$\pi = \begin{pmatrix} c_1, \ldots, c_n \\ \text{opening for } c_i, c_j \end{pmatrix} \longrightarrow$$

Check if openings are valid and if $\text{color}_i \neq \text{color}_j$.

The Fiat-Shamir Transform

Applied to ZKP for Graph 3-Coloring

Statement $x : G = (V, E)$



- Is this protocol sound?
- No! Malicious prover can always break soundness.
  - Consider simulator's strategy from interactive case.
- How can we amplify soundness?
  - Can we use sequential repetition? No!

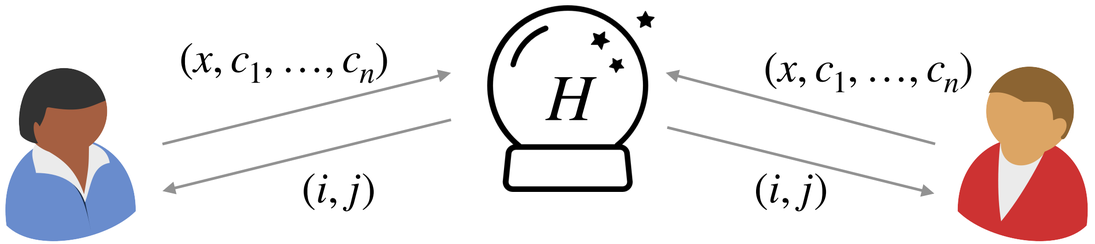
$$\pi = \begin{pmatrix} c_1, \ldots, c_n \\ \text{opening for } c_i, c_j \end{pmatrix} \longrightarrow$$

Check if openings are valid and if $\text{color}_i \neq \text{color}_j$.

The Fiat-Shamir Transform

Applied to ZKP for Graph 3-Coloring

Statement $x : G = (V, E)$



- Is this protocol sound?
- No! Malicious prover can always break soundness.
  - Consider simulator's strategy from interactive case.
- How can we amplify soundness?
  - Can we use sequential repetition? No!

$$\pi = \begin{pmatrix} c_1, \ldots, c_n \\ \text{opening for } c_i, c_j \end{pmatrix} \longrightarrow$$

Check if openings are valid and if $\text{color}_i \neq \text{color}_j$.

Solution: Use parallel repetition!

The Fiat-Shamir Transform

Applied to ZKP for Graph 3-Coloring

Statement $x : G = (V, E)$

Prover

Verifier

Let $\phi : V \to \{1, 2, 3\}$ be a 3-coloring of G.

The Fiat-Shamir Transform

Applied to ZKP for Graph 3-Coloring

Statement $x : G = (V, E)$

Prover

Verifier

Let $\phi : V \to \{1, 2, 3\}$ be a 3-coloring of G.

$\psi_1, \ldots, \psi_\lambda \xleftarrow{\$} \mathsf{Perm}_3$

The Fiat-Shamir Transform

Applied to ZKP for Graph 3-Coloring

Statement $x : G = (V, E)$

Prover

Verifier

Let $\phi : V \to \{1, 2, 3\}$ be a 3-coloring of G.

$\psi_1, \ldots, \psi_\lambda \xleftarrow{\$} \mathsf{Perm}_3$

$\forall i \in \{1,\ldots,n\}, k \in \{1,\ldots,\lambda\} :$

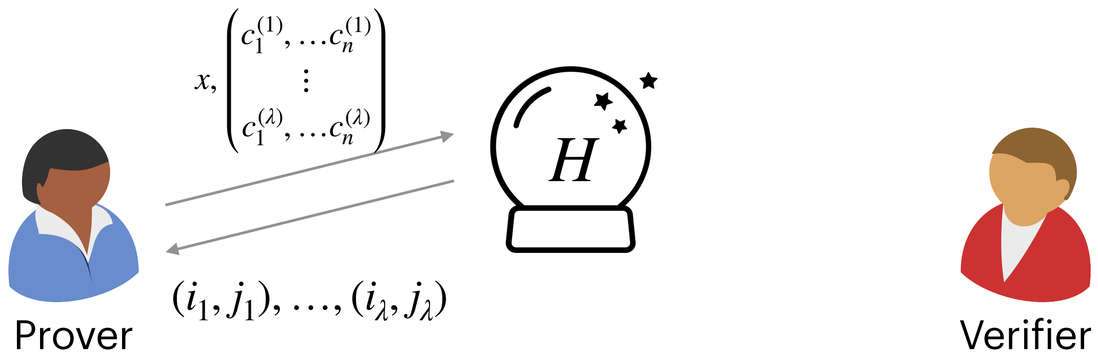
$\mathsf{color}_i^{(k)} := \psi_k\left(\phi(v_i)\right)$

$c_i^{(k)} := \mathsf{Com}\left(\mathsf{color}_i^{(k)}\right)$

The Fiat-Shamir Transform

Applied to ZKP for Graph 3-Coloring

Statement $x : G = (V, E)$



Let $\phi : V \to \{1, 2, 3\}$ be a 3-coloring of G.

$\psi_1, \ldots, \psi_\lambda \xleftarrow{\$} \mathsf{Perm}_3$

$\forall i \in \{1,\ldots,n\}, k \in \{1,\ldots,\lambda\} :$

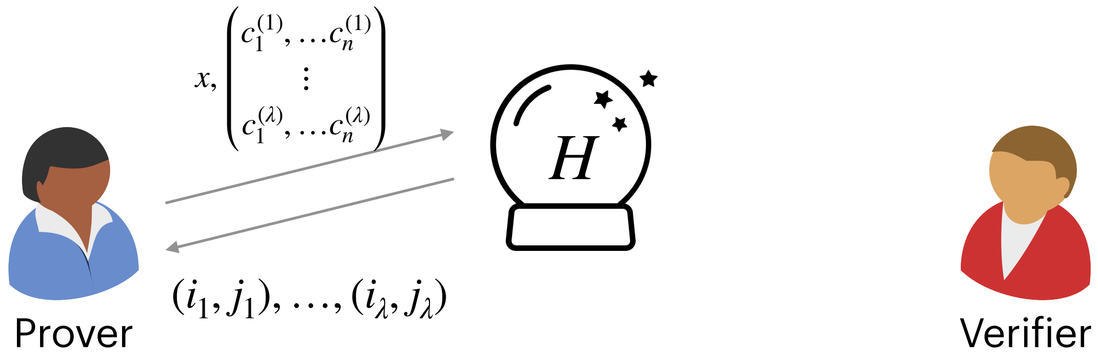
$\mathsf{color}_i^{(k)} := \psi_k\left(\phi(v_i)\right)$

$c_i^{(k)} := \mathsf{Com}\left(\mathsf{color}_i^{(k)}\right)$

The Fiat-Shamir Transform

Applied to ZKP for Graph 3-Coloring

Statement $x : G = (V, E)$



Let $\phi : V \to \{1, 2, 3\}$ be a 3-coloring of G.

$\psi_1, \ldots, \psi_\lambda \xleftarrow{\$} \mathsf{Perm}_3$

$\forall i \in \{1,\ldots,n\}, k \in \{1,\ldots,\lambda\} :$

$\mathsf{color}_i^{(k)} := \psi_k\left(\phi(v_i)\right)$

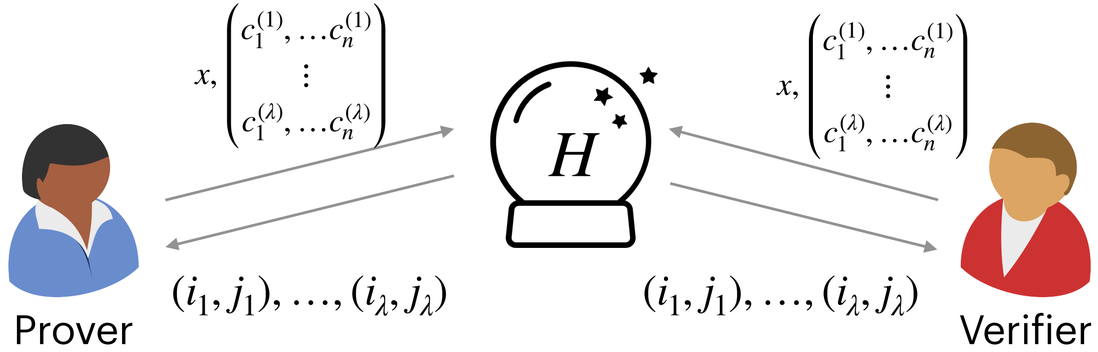
$c_i^{(k)} := \mathsf{Com}\left(\mathsf{color}_i^{(k)}\right)$

$$\pi = \begin{pmatrix} c_1^{(1)}, \ldots c_n^{(1)} \\ \text{opening for } c_{i_1}^{(1)}, c_{j_1}^{(1)} \\ \vdots \\ c_1^{(\lambda)}, \ldots c_n^{(\lambda)} \\ \text{opening for } c_{i_\lambda}^{(\lambda)}, c_{j_\lambda}^{(\lambda)} \end{pmatrix} \longrightarrow$$

The Fiat-Shamir Transform

Applied to ZKP for Graph 3-Coloring

Statement $x : G = (V, E)$



Let $\phi : V \to \{1, 2, 3\}$ be a 3-coloring of G.

$\psi_1, \ldots, \psi_\lambda \xleftarrow{\$} \mathsf{Perm}_3$

$\forall i \in \{1,\ldots,n\}, k \in \{1,\ldots,\lambda\}$:

$\mathsf{color}_i^{(k)} := \psi_k\left(\phi(v_i)\right)$

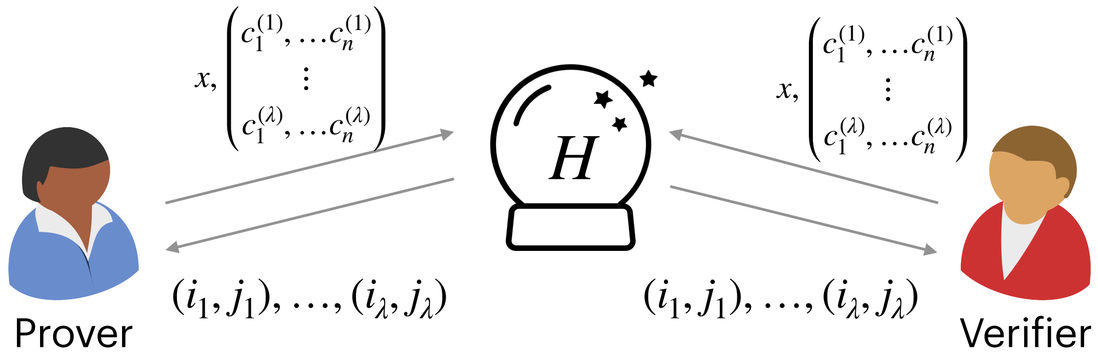
$c_i^{(k)} := \mathsf{Com}\left(\mathsf{color}_i^{(k)}\right)$

$$\pi = \begin{pmatrix} c_1^{(1)}, \ldots c_n^{(1)} \\ \text{opening for } c_{i_1}^{(1)}, c_{j_1}^{(1)} \\ \vdots \\ c_1^{(\lambda)}, \ldots c_n^{(\lambda)} \\ \text{opening for } c_{i_\lambda}^{(\lambda)}, c_{j_\lambda}^{(\lambda)} \end{pmatrix}$$

The Fiat-Shamir Transform

Applied to ZKP for Graph 3-Coloring

Statement $x : G = (V, E)$



Let $\phi : V \to \{1, 2, 3\}$ be a 3-coloring of G.

$\psi_1, \ldots, \psi_\lambda \xleftarrow{\$} \mathsf{Perm}_3$

$\forall i \in \{1,\ldots,n\}, k \in \{1,\ldots,\lambda\}$:

$\mathsf{color}_i^{(k)} := \psi_k\left(\phi(v_i)\right)$

$c_i^{(k)} := \mathsf{Com}\left(\mathsf{color}_i^{(k)}\right)$

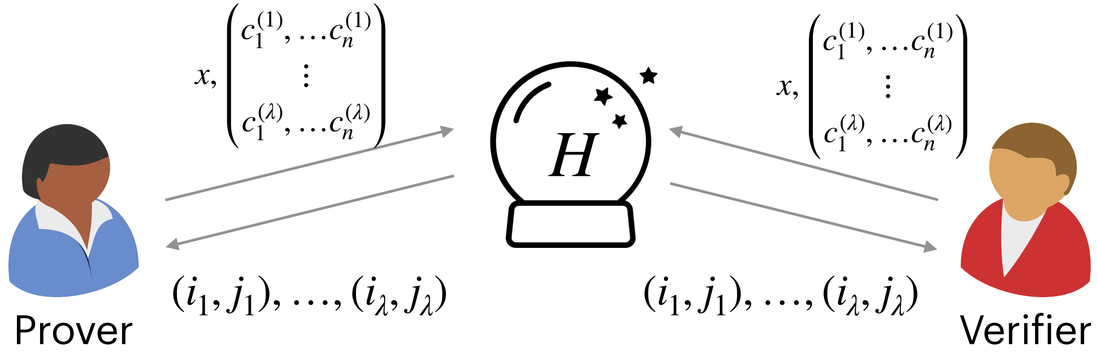
$$\pi = \begin{pmatrix} c_1^{(1)}, \ldots c_n^{(1)} \\ \text{opening for } c_{i_1}^{(1)}, c_{j_1}^{(1)} \\ \vdots \\ c_1^{(\lambda)}, \ldots c_n^{(\lambda)} \\ \text{opening for } c_{i_\lambda}^{(\lambda)}, c_{j_\lambda}^{(\lambda)} \end{pmatrix} \longrightarrow$$

Check if all openings are valid and if $\mathsf{color}_i^{(k)} \neq \mathsf{color}_j^{(k)}$.

The Fiat-Shamir Transform

Applied to ZKP for Graph 3-Coloring

Statement $x : G = (V, E)$



Let $\phi : V \to \{1, 2, 3\}$ be a 3-coloring of G.

$\psi_1, \dots, \psi_\lambda \xleftarrow{\$} \mathsf{Perm}_3$

$\forall i \in \{1, \dots, n\}, k \in \{1, \dots, \lambda\}$:

$\mathsf{color}_i^{(k)} := \psi_k\left(\phi(v_i)\right)$

$c_i^{(k)} := \mathsf{Com}\left(\mathsf{color}_i^{(k)}\right)$

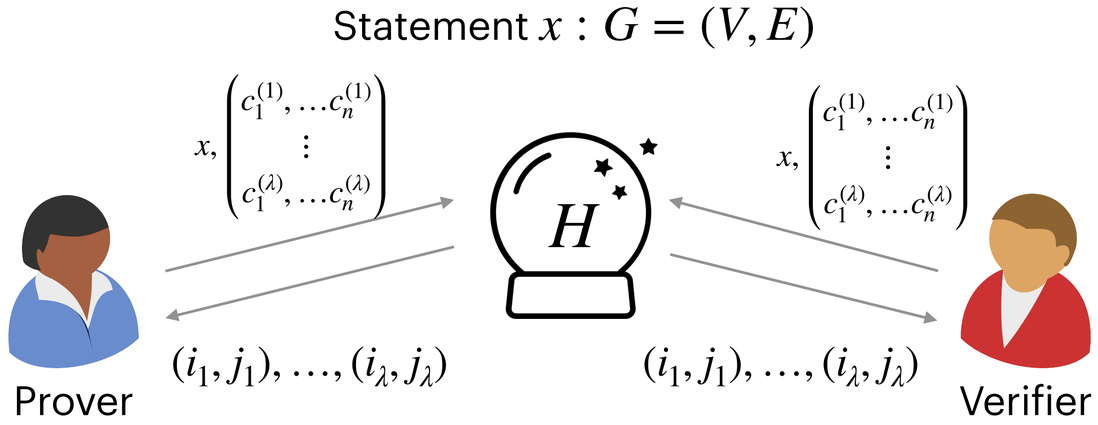
$$\pi = \begin{pmatrix} c_1^{(1)}, \dots c_n^{(1)} \\ \text{opening for } c_{i_1}^{(1)}, c_{j_1}^{(1)} \\ \vdots \\ c_1^{(\lambda)}, \dots c_n^{(\lambda)} \\ \text{opening for } c_{i_\lambda}^{(\lambda)}, c_{j_\lambda}^{(\lambda)} \end{pmatrix} \longrightarrow$$

Soundness:

Check if all openings are valid and if $\mathsf{color}_i^{(k)} \neq \mathsf{color}_j^{(k)}$.

The Fiat-Shamir Transform

Applied to ZKP for Graph 3-Coloring



Let $\phi : V \to \{1, 2, 3\}$ be a 3-coloring of G.

$\psi_1, \ldots, \psi_\lambda \xleftarrow{\$} \mathsf{Perm}_3$

$\forall i \in \{1,\ldots,n\}, k \in \{1,\ldots,\lambda\}$:

$\mathsf{color}_i^{(k)} := \psi_k\left(\phi(v_i)\right)$

$c_i^{(k)} := \mathsf{Com}\left(\mathsf{color}_i^{(k)}\right)$

$$\pi = \begin{pmatrix} c_1^{(1)}, \ldots c_n^{(1)} \\ \text{opening for } c_{i_1}^{(1)}, c_{j_1}^{(1)} \\ \vdots \\ c_1^{(\lambda)}, \ldots c_n^{(\lambda)} \\ \text{opening for } c_{i_\lambda}^{(\lambda)}, c_{j_\lambda}^{(\lambda)} \end{pmatrix} \longrightarrow$$

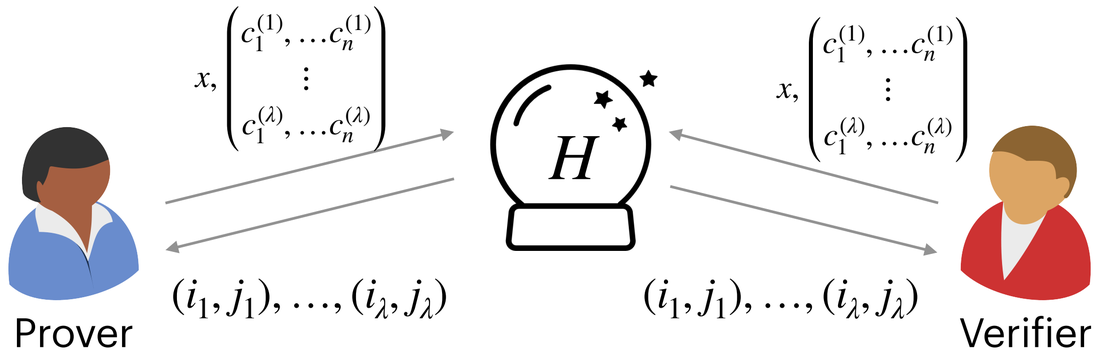
Soundness: Malicious prover would have to guess all challenge edges to cheat.

Check if all openings are valid and if $\mathsf{color}_i^{(k)} \neq \mathsf{color}_j^{(k)}$.

The Fiat-Shamir Transform

Applied to ZKP for Graph 3-Coloring

Statement $x : G = (V, E)$



Let $\phi : V \to \{1, 2, 3\}$ be a 3-coloring of G.

$\psi_1, \dots, \psi_\lambda \xleftarrow{\$} \mathsf{Perm}_3$

$\forall i \in \{1,\dots,n\}, k \in \{1,\dots,\lambda\}$:

$\mathsf{color}_i^{(k)} := \psi_k\left(\phi(v_i)\right)$

$c_i^{(k)} := \mathsf{Com}\left(\mathsf{color}_i^{(k)}\right)$

$$\pi = \begin{pmatrix} c_1^{(1)}, \dots c_n^{(1)} \\ \text{opening for } c_{i_1}^{(1)}, c_{j_1}^{(1)} \\ \vdots \\ c_1^{(\lambda)}, \dots c_n^{(\lambda)} \\ \text{opening for } c_{i_\lambda}^{(\lambda)}, c_{j_\lambda}^{(\lambda)} \end{pmatrix} \longrightarrow$$

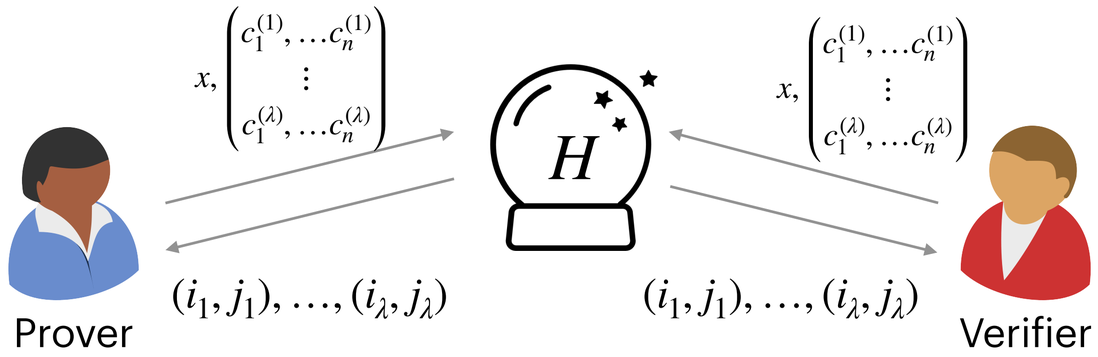
Zero Knowledge:

Check if all openings are valid and if $\mathsf{color}_i^{(k)} \neq \mathsf{color}_j^{(k)}$.

The Fiat-Shamir Transform

Applied to ZKP for Graph 3-Coloring

Statement $x : G = (V, E)$



Let $\phi : V \to \{1, 2, 3\}$ be a 3-coloring of G.

$\psi_1, \ldots, \psi_\lambda \xleftarrow{\$} \mathsf{Perm}_3$

$\forall i \in \{1, \ldots, n\}, k \in \{1, \ldots, \lambda\}$:

$\mathsf{color}_i^{(k)} := \psi_k\left(\phi(v_i)\right)$

$c_i^{(k)} := \mathsf{Com}\left(\mathsf{color}_i^{(k)}\right)$

$$\pi = \begin{pmatrix} c_1^{(1)}, \ldots c_n^{(1)} \\ \text{opening for } c_{i_1}^{(1)}, c_{j_1}^{(1)} \\ \vdots \\ c_1^{(\lambda)}, \ldots c_n^{(\lambda)} \\ \text{opening for } c_{i_\lambda}^{(\lambda)}, c_{j_\lambda}^{(\lambda)} \end{pmatrix} \longrightarrow$$

Check if all openings are valid and if $\mathsf{color}_i^{(k)} \neq \mathsf{color}_j^{(k)}$.

Zero Knowledge: Prover never interacts with the malicious verifier and the random oracle is like an “honest verifier”.

In the proof, the simulator can program the random oracle.

Zero-Knowledge Proof of Knowledge

Until now we have considered decision problems:

Either the statement is true ($x \in L$) or false ($x \notin L$).

Zero-Knowledge Proof of Knowledge

Until now we have considered decision problems:

Either the statement is true ($x \in L$) or false ($x \notin L$).

(Either there exists a 3-coloring or there does not exist a 3-coloring.)

Zero-Knowledge Proof of Knowledge

Until now we have considered decision problems:

Either the statement is true ($x \in L$) or false ($x \notin L$).

(Either there exists a 3-coloring or there does not exist a 3-coloring.)

Wants to prove that she has the secret key s s.t. h $= g^s$.

Prover

$$x = (\mathbb{G}, g, p, h)$$

Verifier

Zero-Knowledge Proof of Knowledge

Until now we have considered decision problems:

Either the statement is true ($x \in L$) or false ($x \notin L$).

(Either there exists a 3-coloring or there does not exist a 3-coloring.)

Wants to prove that she has the secret key s s.t. h $= g^s$.

Prover

$x = (\mathbb{G}, g, p, h)$

Verifier

"There exists s such that $h = g^s$."

Zero-Knowledge Proof of Knowledge

Until now we have considered decision problems:

Either the statement is true ($x \in L$) or false ($x \notin L$).

(Either there exists a 3-coloring or there does not exist a 3-coloring.)

Wants to prove that she has the secret key s s.t. h $= g^s$.

Prover

$x = (\mathbb{G}, g, p, h)$

Verifier

"There exists s such that $h = g^s$."

- This statement is always true since g generates $\mathbb{G}$.

Zero-Knowledge Proof of Knowledge

Until now we have considered decision problems:

Either the statement is true ($x \in L$) or false ($x \notin L$).

(Either there exists a 3-coloring or there does not exist a 3-coloring.)

Wants to prove that she has the secret key s s.t. $h = g^s$.

Prover

$$x = (\mathbb{G}, g, p, h)$$

Verifier

"There exists s such that $h = g^s$."

- This statement is always true since g generates $\mathbb{G}$.
- This is not an interesting decision problem.

Zero-Knowledge Proof of Knowledge

Until now we have considered decision problems:

Either the statement is true ($x \in L$) or false ($x \notin L$).

(Either there exists a 3-coloring or there does not exist a 3-coloring.)

Wants to prove that she has the secret key s s.t. $h = g^s$.

Prover

$$x = (\mathbb{G}, g, p, h)$$

Verifier

"There exists s such that $h = g^s$."

- This statement is always true since g generates $\mathbb{G}$.
- This is not an interesting decision problem.
- It is a search problem.

Zero-Knowledge Proof of Knowledge

Until now we have considered decision problems:

Either the statement is true ($x \in L$) or false ($x \notin L$).

(Either there exists a 3-coloring or there does not exist a 3-coloring.)

Wants to prove that she has the secret key s s.t. h $= g^s$.

Prover

$x = (\mathbb{G}, g, p, h)$

Verifier

"I know s such that $h = g^s$."

Zero-Knowledge Proof of Knowledge

Until now we have considered decision problems:

Either the statement is true ($x \in L$) or false ($x \notin L$).

(Either there exists a 3-coloring or there does not exist a 3-coloring.)

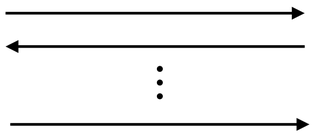
Wants to prove that she has the secret key s s.t. h $= g^s$.

Prover

$x = (\mathbb{G}, g, p, h)$

Verifier

"I know s such that $h = g^s$."



Zero-Knowledge Proof of Knowledge

Until now we have considered decision problems:

Either the statement is true ($x \in L$) or false ($x \notin L$).

(Either there exists a 3-coloring or there does not exist a 3-coloring.)

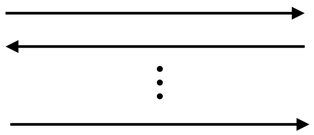
Wants to prove that she has the secret key s s.t. h $= g^s$.

Prover

$x = (\mathbb{G}, g, p, h)$

Verifier

"I know s such that $h = g^s$."



"I'm convinced you know the solution."

Zero-Knowledge Proof of Knowledge

Until now we have considered decision problems:

Either the statement is true ($x \in L$) or false ($x \notin L$).

(Either there exists a 3-coloring or there does not exist a 3-coloring.)

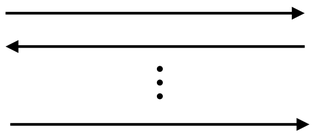
Wants to prove that she has the secret key s s.t. h $= g^s$.

Prover

$x = (\mathbb{G}, g, p, h)$

Verifier

"I know s such that $h = g^s$."



"I'm convinced you know the solution."

Completeness: The verifier accepts when it interacts with the prover run with input s.

Zero-Knowledge Proof of Knowledge

Until now we have considered decision problems:

Either the statement is true ($x \in L$) or false ($x \notin L$).

(Either there exists a 3-coloring or there does not exist a 3-coloring.)

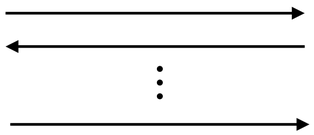
Wants to prove that she has the secret key s s.t. $h = g^s$.

Prover

$x = (\mathbb{G}, g, p, h)$

Verifier

"I know s such that $h = g^s$."



"I'm convinced you know the solution."

Completeness: The verifier accepts when it interacts with the prover run with input s.

Zero-Knowledge: For every (possibly malicious) verifier, there is a simulator whose output is indistinguishable from the verifier's view in an interaction with the prover.

Zero-Knowledge Proof of Knowledge

Until now we have considered decision problems:

Either the statement is true ($x \in L$) or false ($x \notin L$).

(Either there exists a 3-coloring or there does not exist a 3-coloring.)

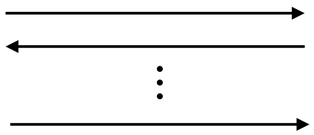
Wants to prove that she has the secret key s s.t. h $= g^s$.

Prover

$x = (\mathbb{G}, g, p, h)$

Verifier

"I know s such that $h = g^s$."



"I'm convinced you know the solution."

Completeness: The verifier accepts when it interacts with the prover run with input s.

Soundness: What does it mean to say a machine *knows* something?

Zero-Knowledge: For every (possibly malicious) verifier, there is a simulator whose output is indistinguishable from the verifier's view in an interaction with the prover.

Zero-Knowledge Proof of Knowledge

Until now we have considered decision problems:

Either the statement is true ($x \in L$) or false ($x \notin L$).

(Either there exists a 3-coloring or there does not exist a 3-coloring.)

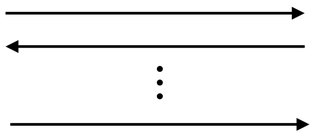
Wants to prove that she has the secret key s s.t. $h = g^s$.

Prover

$x = (\mathbb{G}, g, p, h)$

Verifier

"I know s such that $h = g^s$."



"I'm convinced you know the solution."

Completeness: The verifier accepts when it interacts with the prover run with input s.

Soundness: If a (possibly malicious) prover convinces the verifier then there is an efficient algorithm E, called the extractor, that outputs the witness s.

Zero-Knowledge: For every (possibly malicious) verifier, there is a simulator whose output is indistinguishable from the verifier's view in an interaction with the prover.

Zero-Knowledge Proof of Knowledge

Until now we have considered decision problems:

Either the statement is true ($x \in L$) or false ($x \notin L$).

(Either there exists a 3-coloring or there does not exist a 3-coloring.)

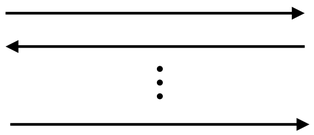
Wants to prove that she has the secret key s s.t. h $= g^s$.

Prover

$x = (\mathbb{G}, g, p, h)$

Verifier

"I know s such that $h = g^s$."



"I'm convinced you know the solution."

Completeness: The verifier accepts when it interacts with the prover run with input s.

Soundness: If a (possibly malicious) prover convinces the verifier then there is an efficient algorithm E, called the extractor, that outputs the witness s.

We will not formally define extractors in this course.

Zero-Knowledge: For every (possibly malicious) verifier, there is a simulator whose output is indistinguishable from the verifier's view in an interaction with the prover.

Zero-Knowledge Proof of Knowledge

Theorem [Goldreich-Micali-Wigderson'87]**:** Assuming OWFs exist, any language in NP has a zero-knowledge proof of knowledge.

Additional Slides

ZK Proof of Knowledge for Discrete Log

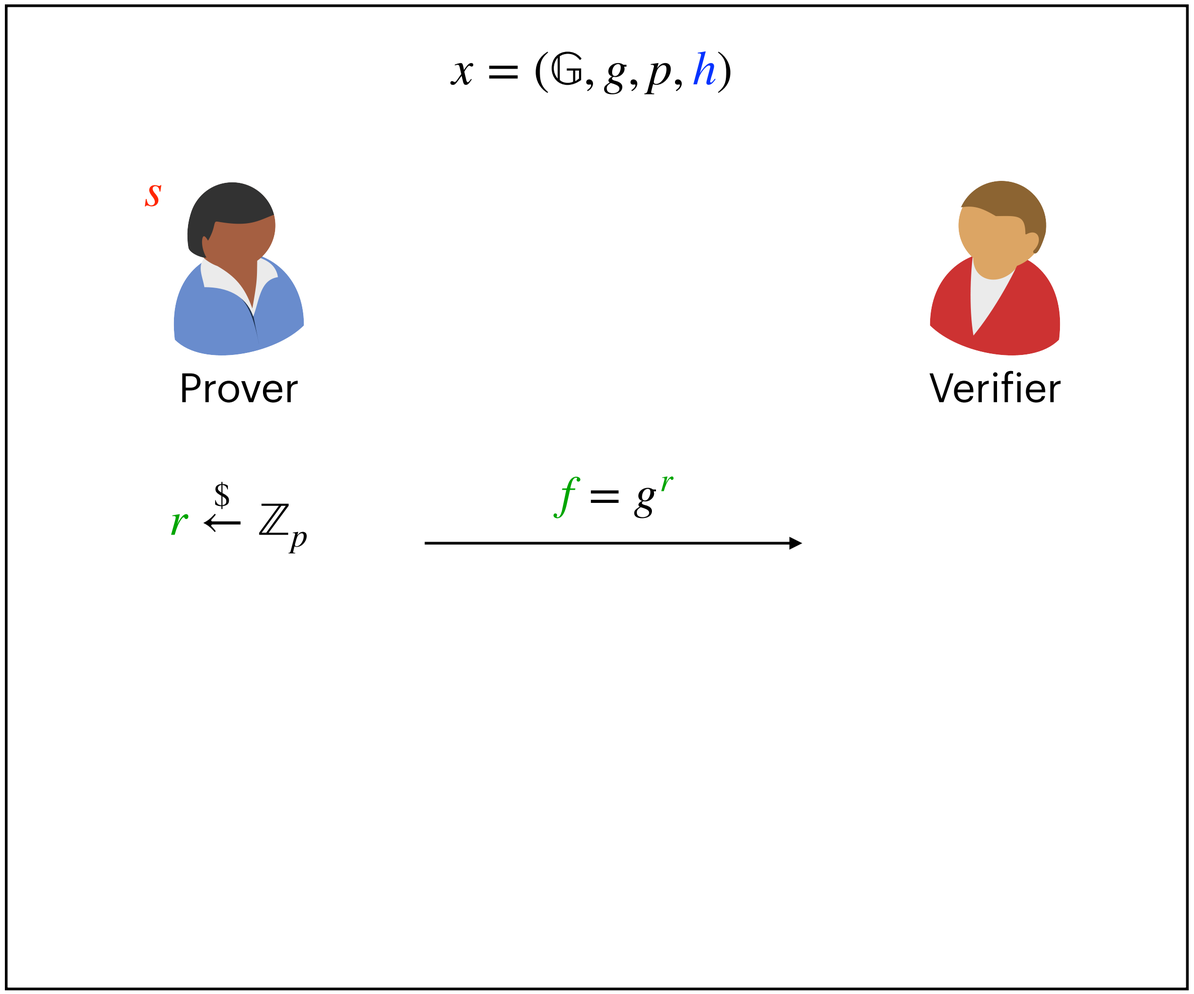
$x = (\mathbb{G}, g, p, h)$

s

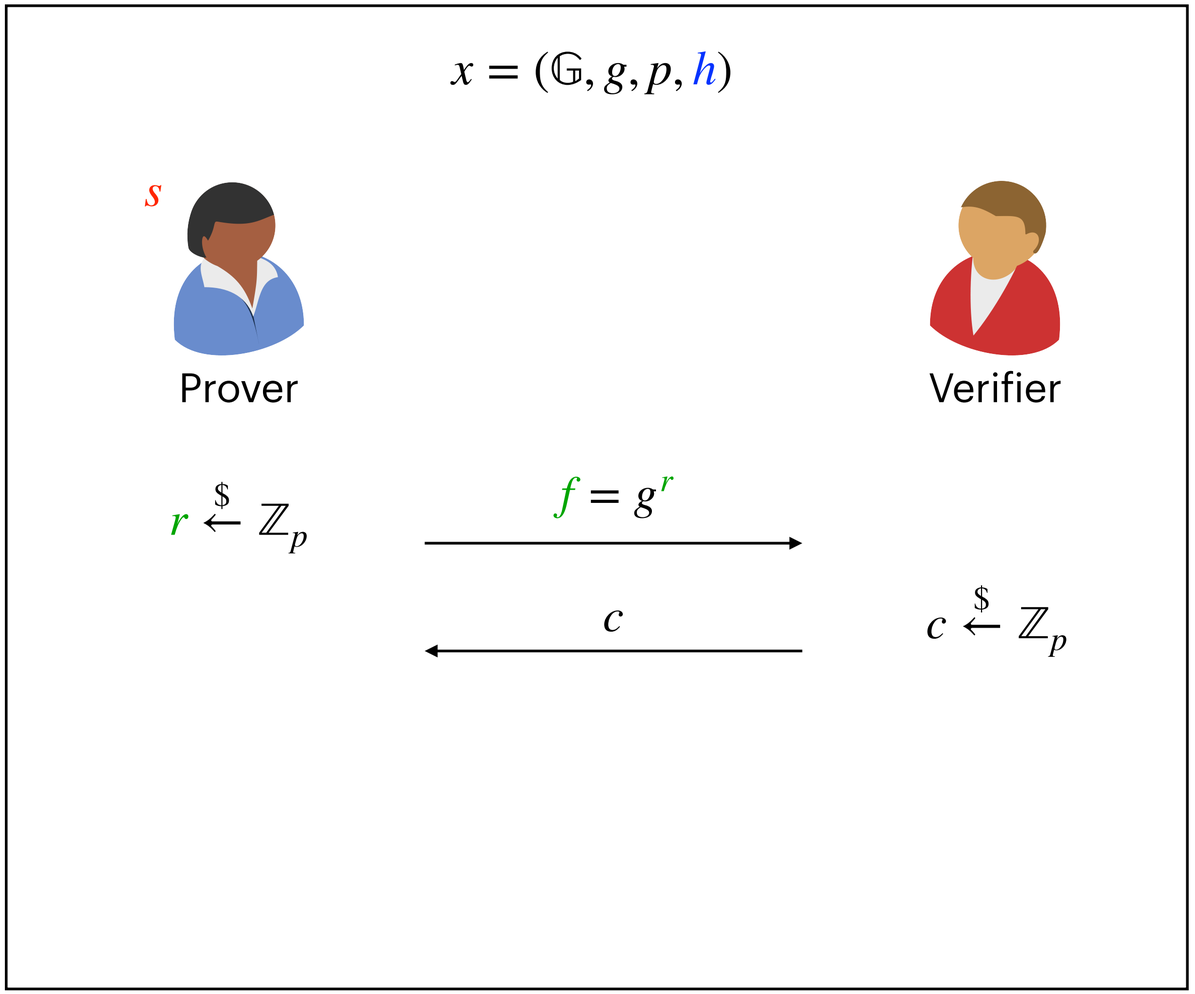
Prover

Verifier

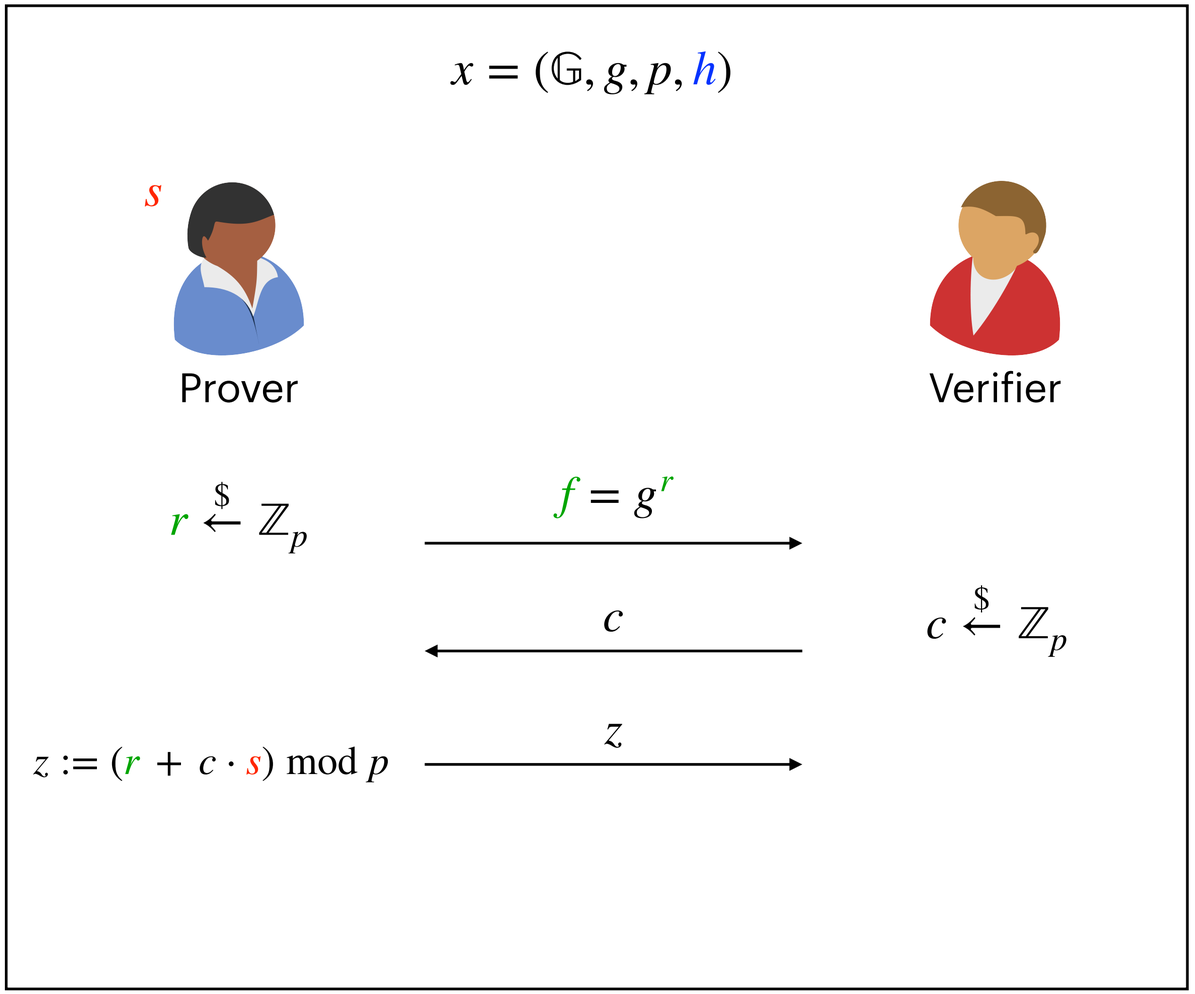
ZK Proof of Knowledge for Discrete Log



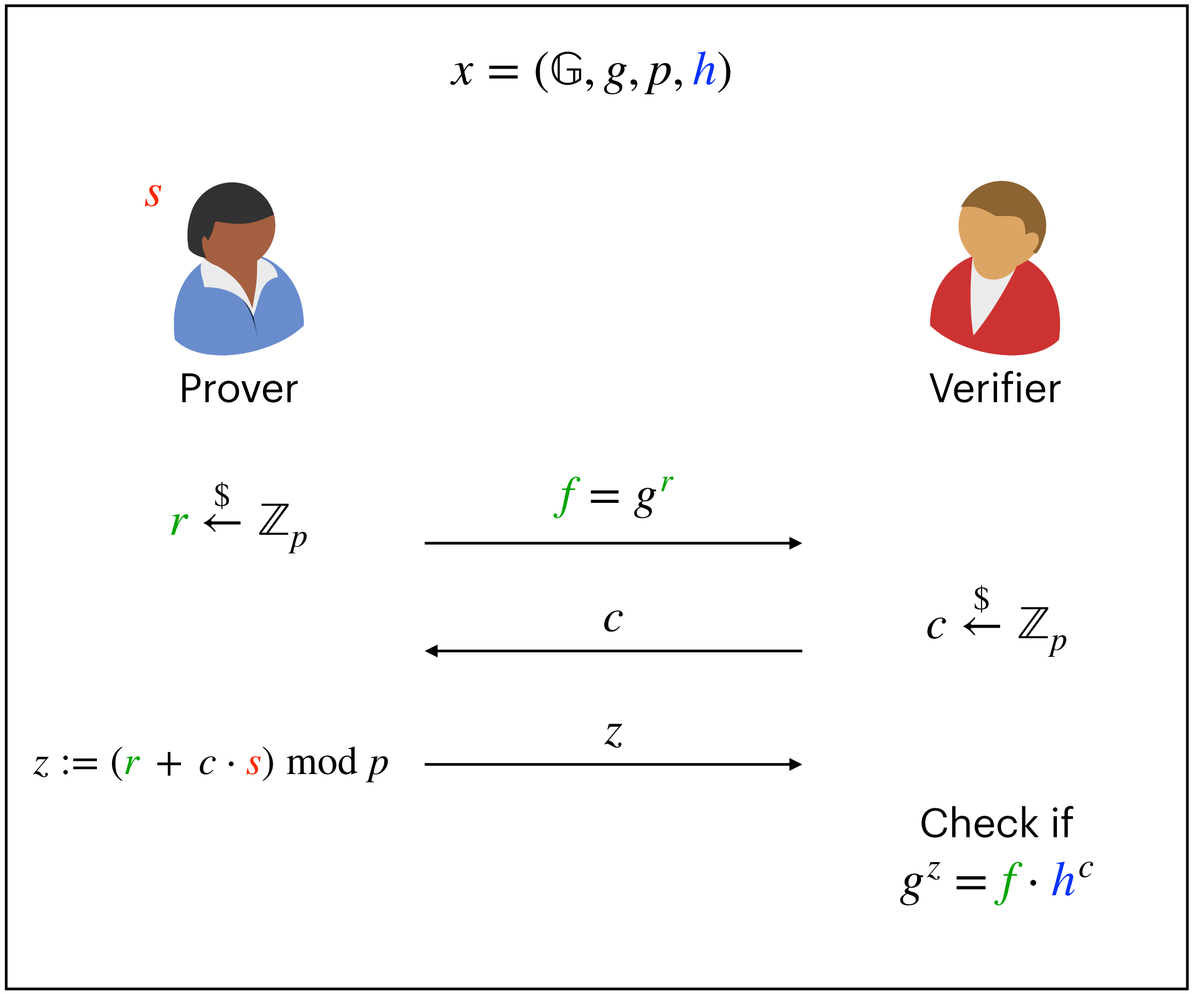
ZK Proof of Knowledge for Discrete Log



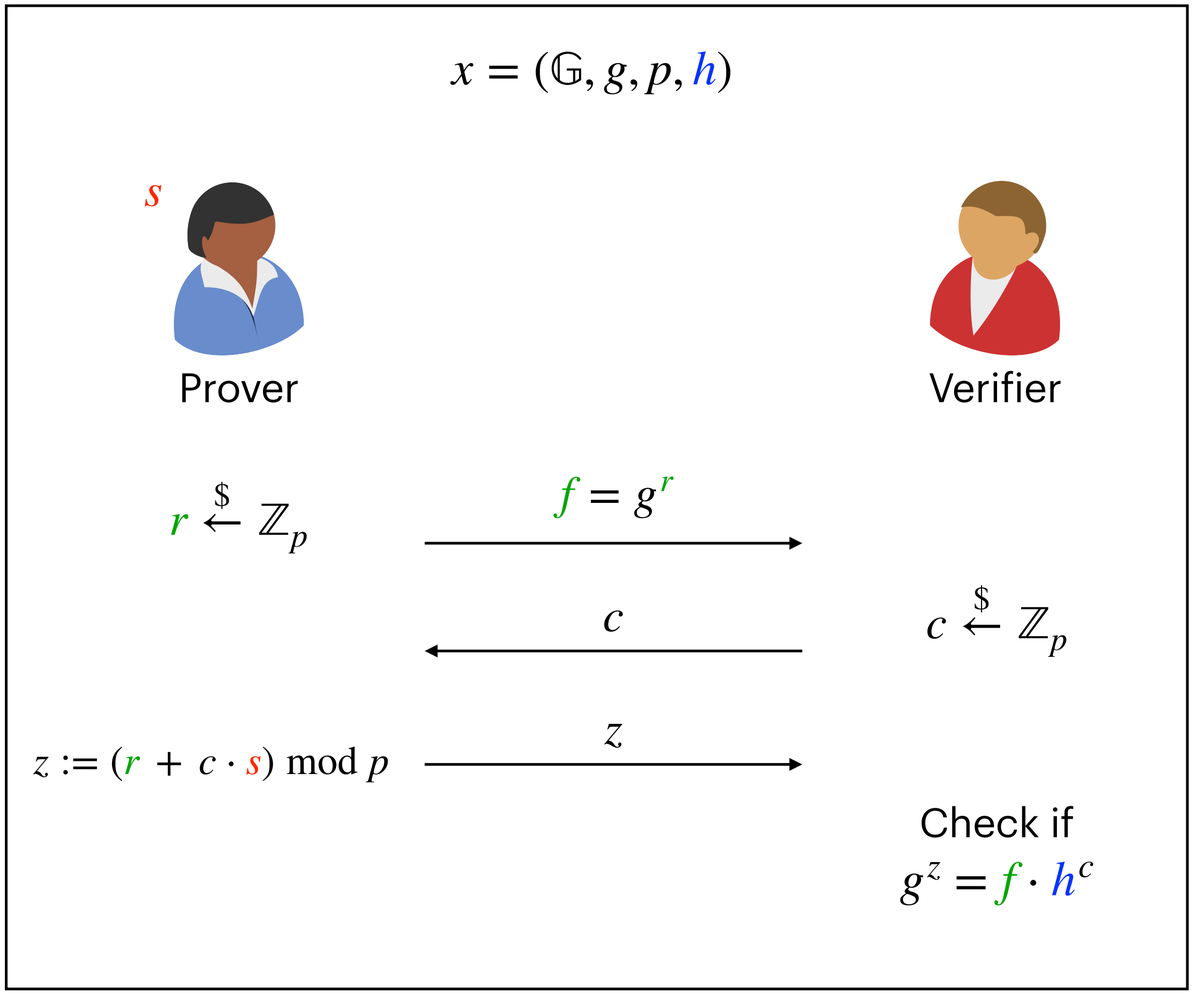
ZK Proof of Knowledge for Discrete Log



ZK Proof of Knowledge for Discrete Log



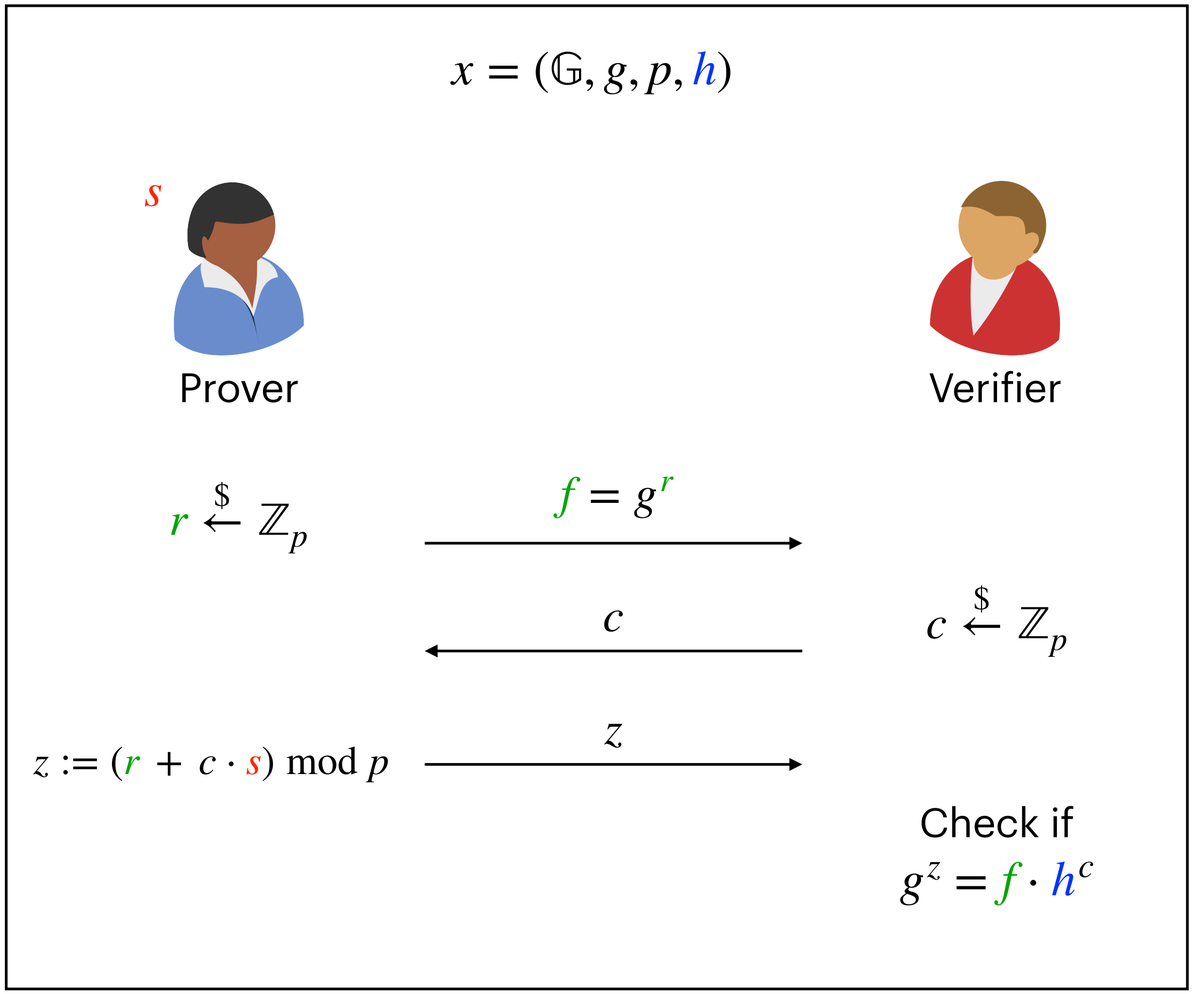
ZK Proof of Knowledge for Discrete Log



Completeness

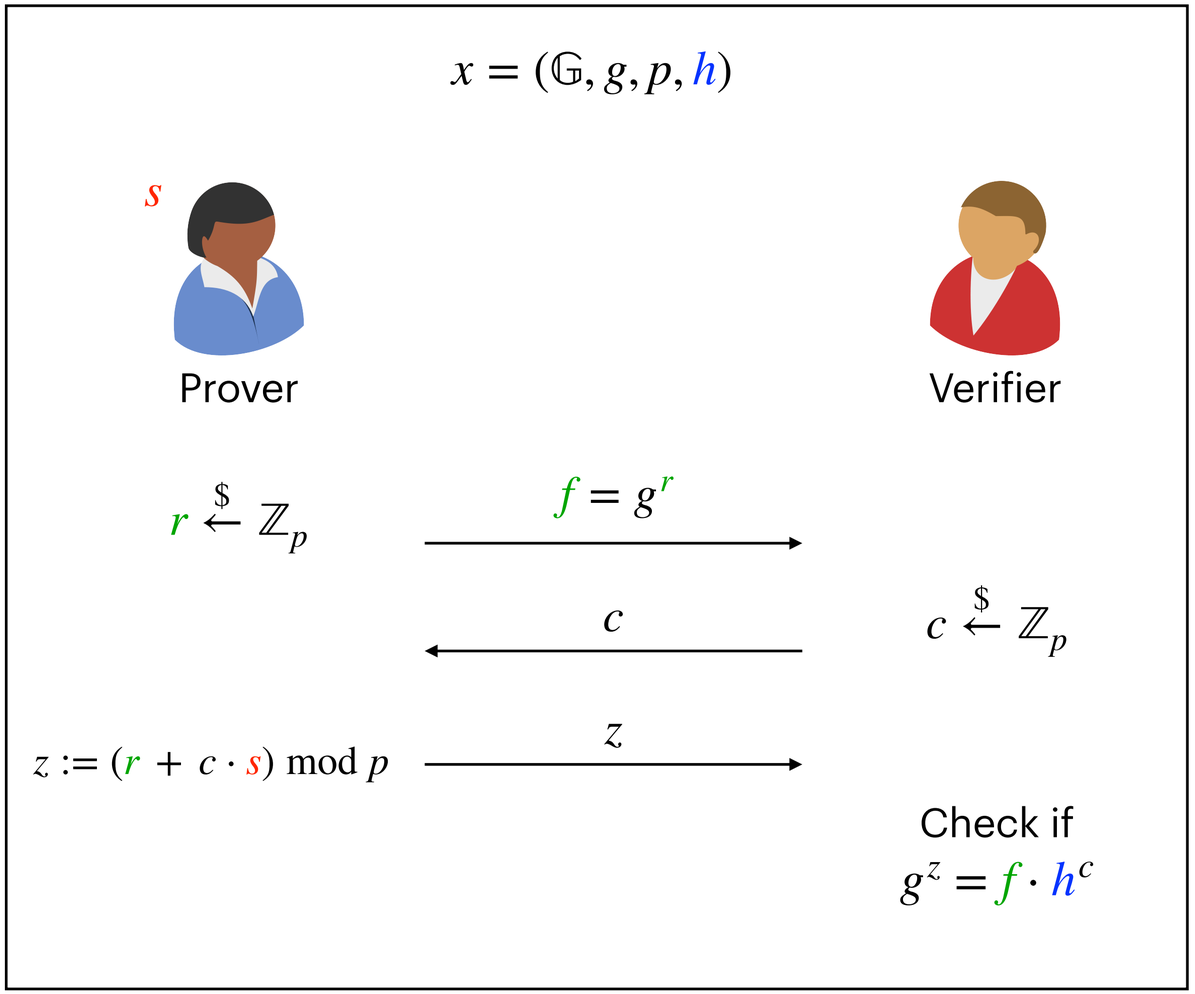
$$g^z = g^{r+c\cdot s} = g^r \cdot (g^s)^c = f \cdot h^c .$$

ZK Proof of Knowledge for Discrete Log



Zero-Knowledge

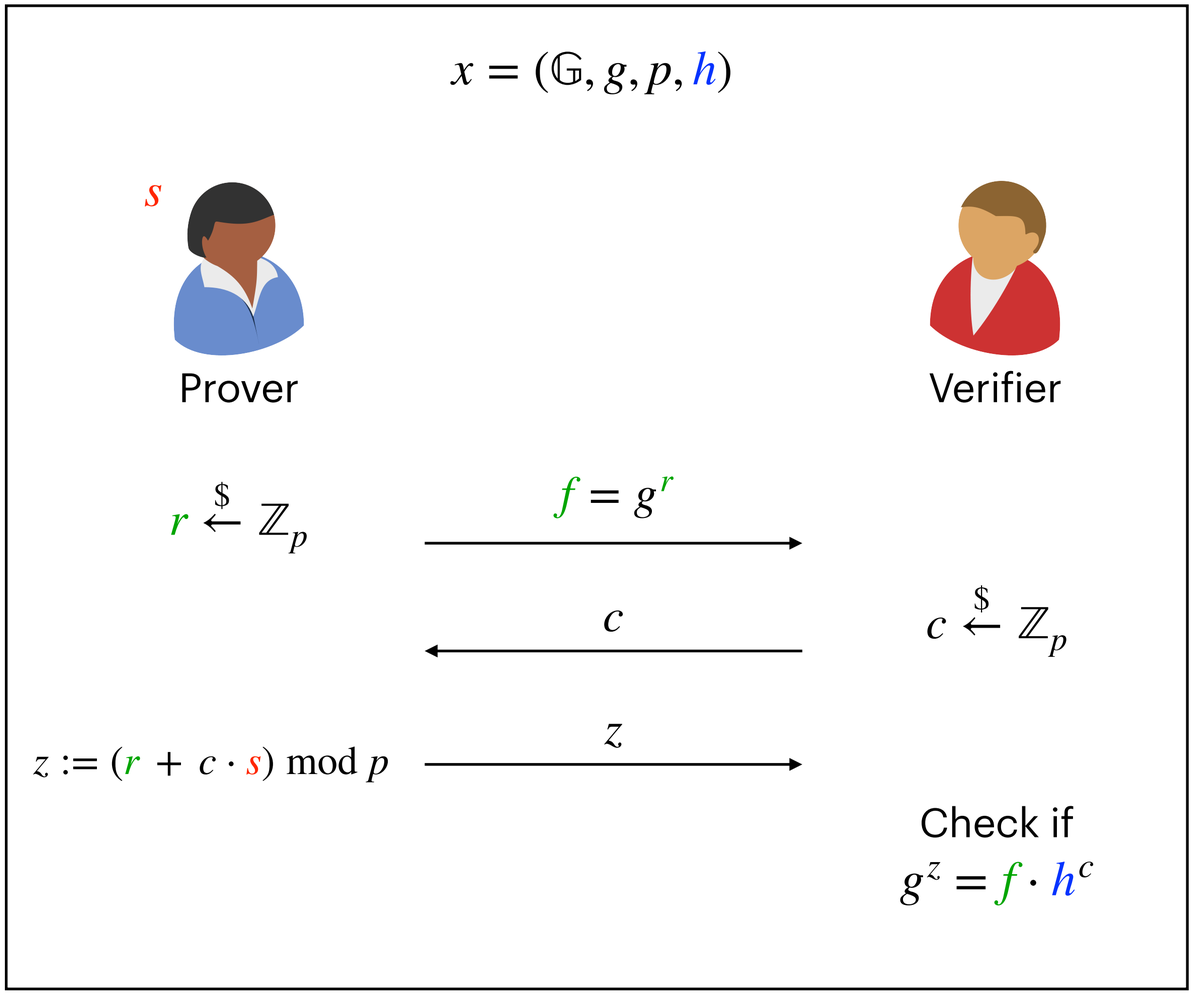
ZK Proof of Knowledge for Discrete Log



Zero-Knowledge

Not clear how to prove zero-knowledge against arbitrary malicious verifiers.

ZK Proof of Knowledge for Discrete Log

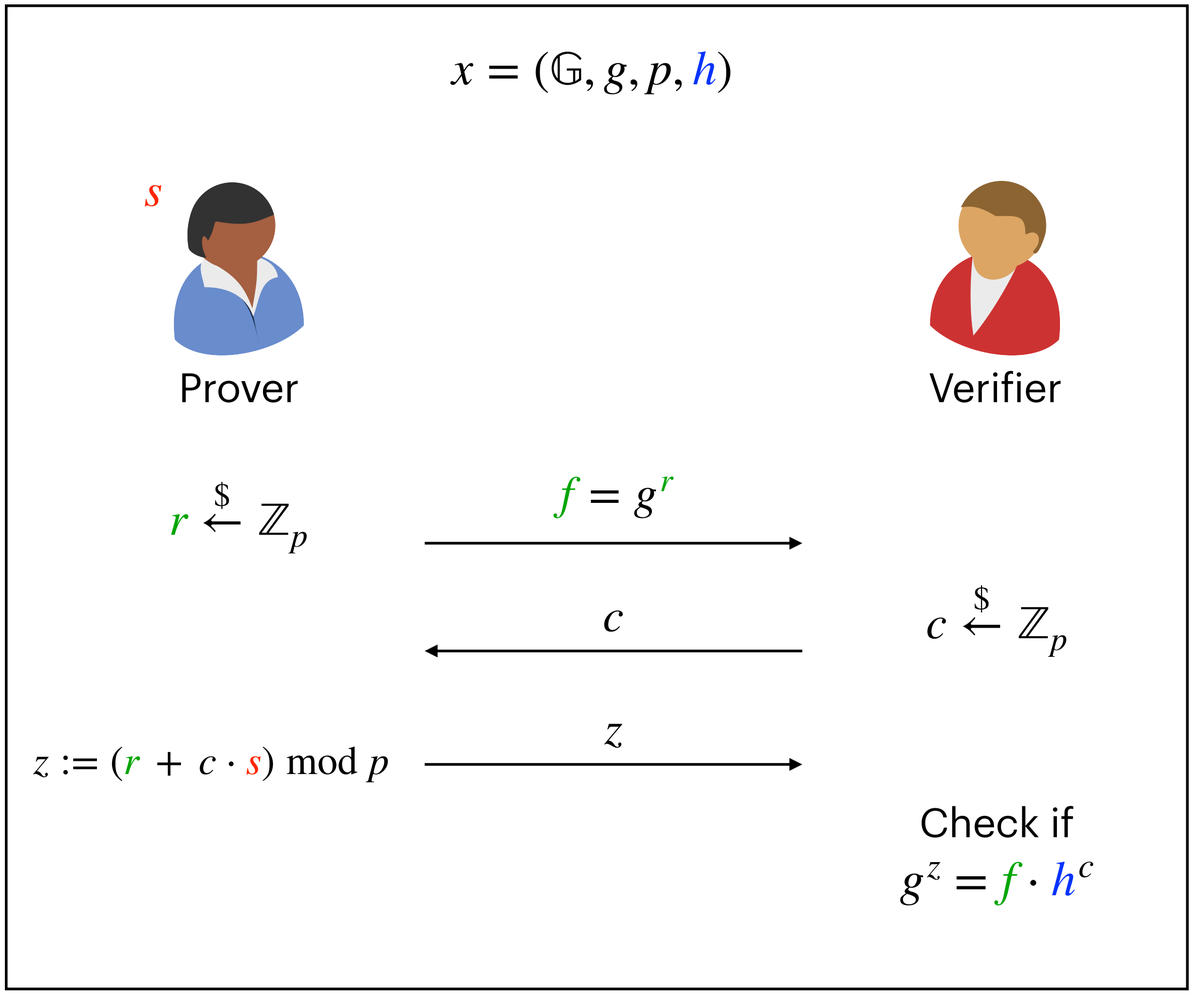


Zero-Knowledge

Not clear how to prove zero-knowledge against arbitrary malicious verifiers.

We will prove a weaker guarantee: an honest verifier will not gain knowledge by interacting with the prover.

ZK Proof of Knowledge for Discrete Log



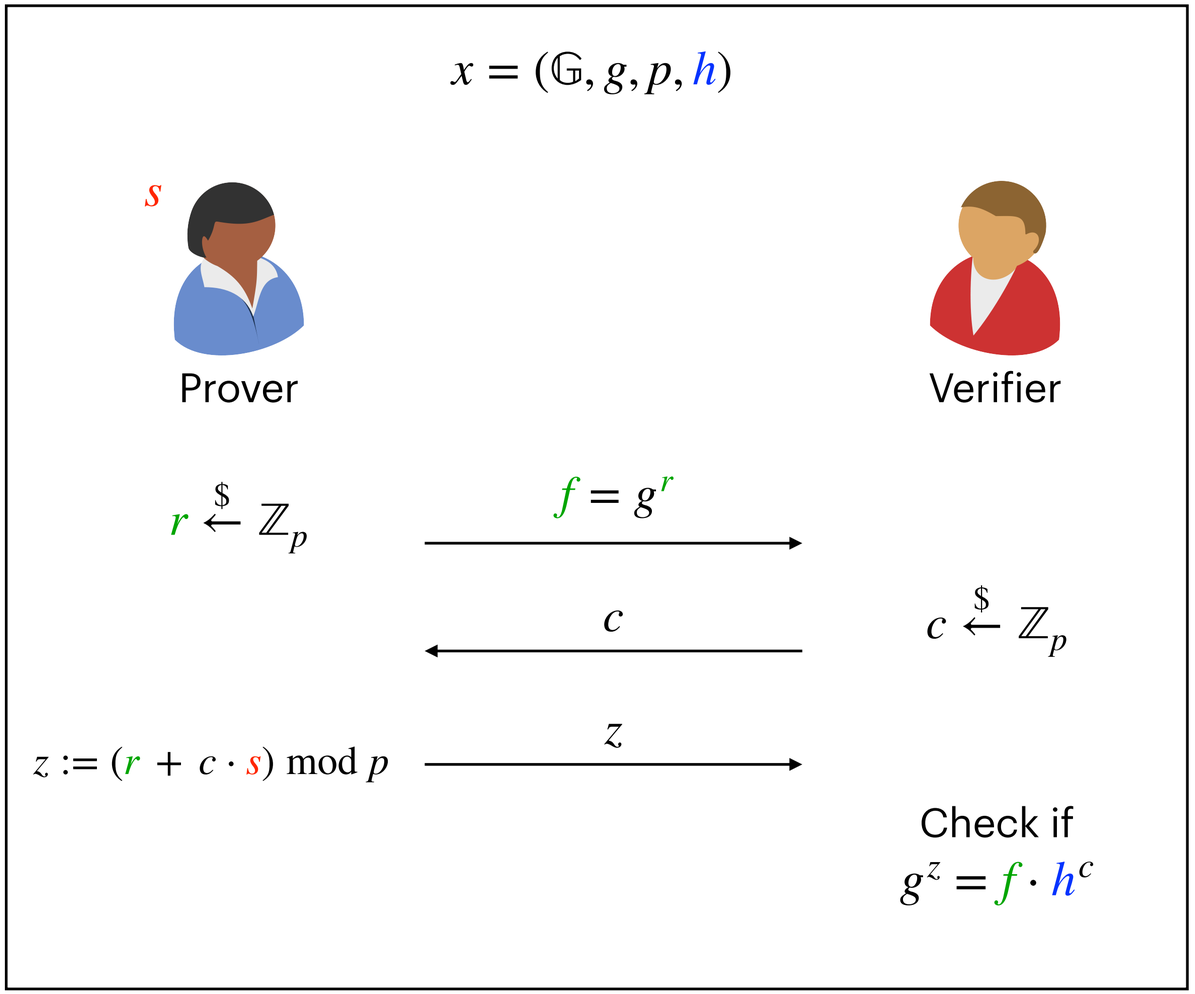
Zero-Knowledge

Not clear how to prove zero-knowledge against arbitrary malicious verifiers.

We will prove a weaker guarantee: an honest verifier will not gain knowledge by interacting with the prover.

Honest-Verifier Zero-Knowledge: There is a simulator whose output is indistinguishable from the honest verifier's view in the protocol.

ZK Proof of Knowledge for Discrete Log



Zero-Knowledge

Not clear how to prove zero-knowledge against arbitrary malicious verifiers.

We will prove a weaker guarantee: an honest verifier will not gain knowledge by interacting with the prover.

Honest-Verifier Zero-Knowledge: There is a simulator whose output is indistinguishable from the honest verifier's view in the protocol.

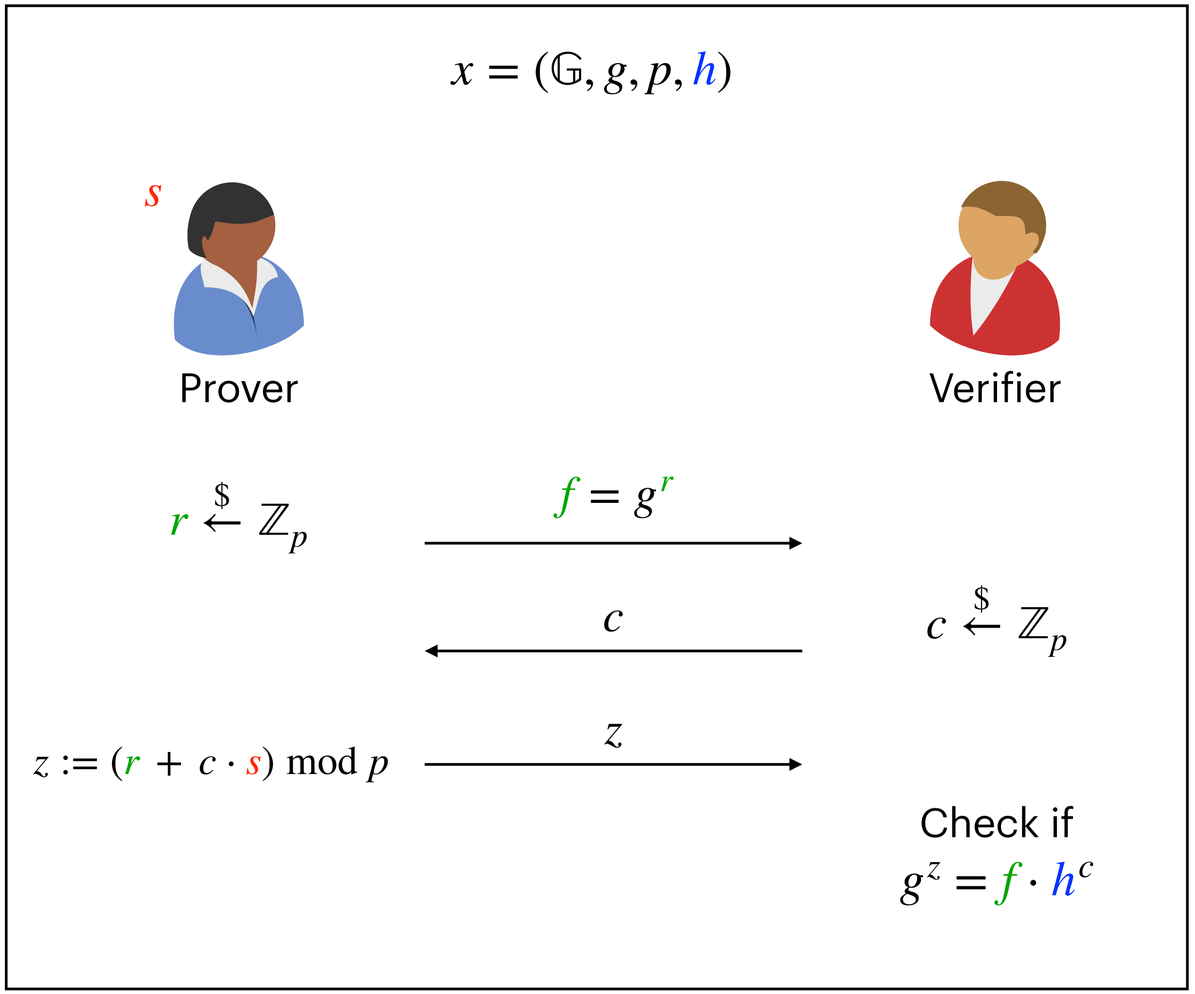
Simulator

$z, c \xleftarrow{\$} \mathbb{Z}_p$

$f = \dfrac{g^z}{h^c}$

Output (f, c, z).

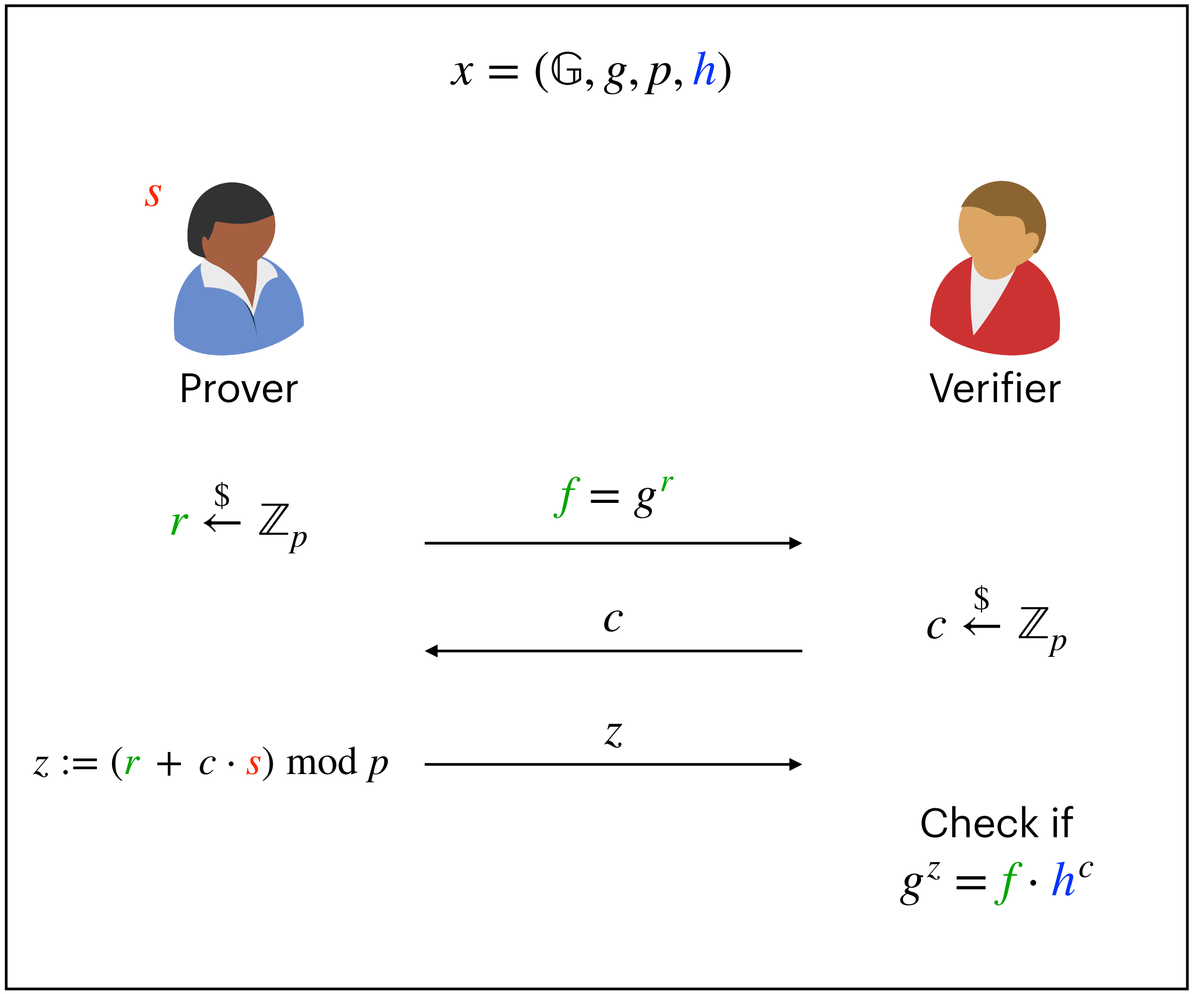
ZK Proof of Knowledge for Discrete Log



Soundness

Prover

ZK Proof of Knowledge for Discrete Log



Soundness


Prover

We will consider the simpler case where the malicious prover convinces the verifier with probability 1.

ZK Proof of Knowledge for Discrete Log

$x = (\mathbb{G}, g, p, h)$

Prover

s

Verifier

$r \xleftarrow{\$} \mathbb{Z}_p$

$f = g^r$

c

$c \xleftarrow{\$} \mathbb{Z}_p$

$z := (r + c \cdot s) \bmod p$

z

Check if

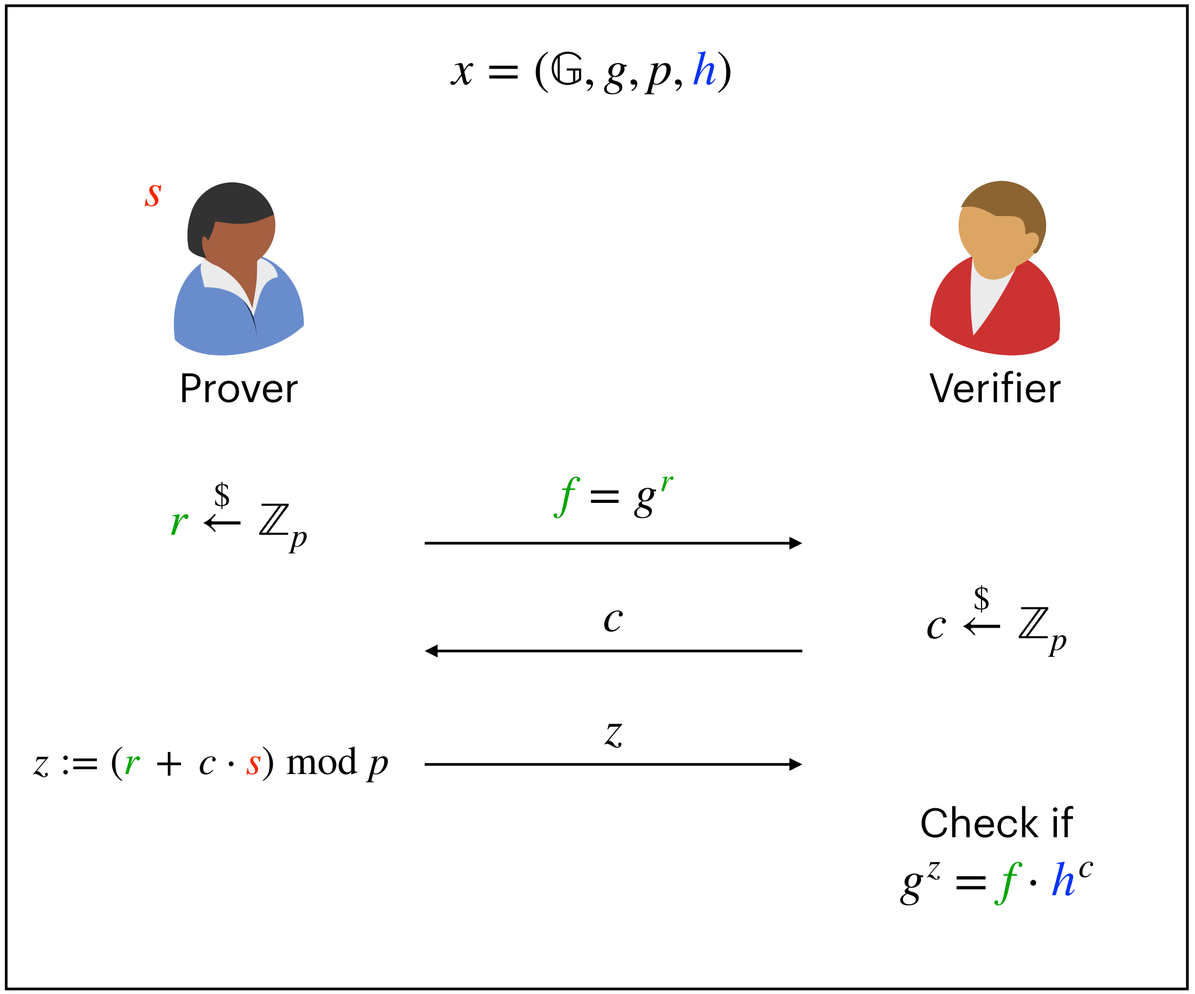
$g^z = f \cdot h^c$

Soundness

Prover

Extractor

ZK Proof of Knowledge for Discrete Log



Soundness


Prover


Extractor

Extract the witness while making the prover believe it is talking to the verifier.

ZK Proof of Knowledge for Discrete Log

$x = (\mathbb{G}, g, p, h)$

Prover

s

Verifier

$r \xleftarrow{\$} \mathbb{Z}_p$

$f = g^r$

c

$c \xleftarrow{\$} \mathbb{Z}_p$

$z := (r + c \cdot s) \bmod p$

z

Check if

$g^z = f \cdot h^c$

Soundness

Prover

Extractor

ZK Proof of Knowledge for Discrete Log

$x = (\mathbb{G}, g, p, h)$

Prover

Verifier

s

$r \xleftarrow{\$} \mathbb{Z}_p$

$f = g^r$ →

← c

$c \xleftarrow{\$} \mathbb{Z}_p$

$z := (r + c \cdot s) \bmod p$

z →

Check if $g^z = f \cdot h^c$

Soundness

Prover

f →

Extractor

ZK Proof of Knowledge for Discrete Log

$x = (\mathbb{G}, g, p, h)$

Prover

s

Verifier

$r \xleftarrow{\$} \mathbb{Z}_p$

$f = g^r$ →

← c

$c \xleftarrow{\$} \mathbb{Z}_p$

$z := (r + c \cdot s) \bmod p$

z →

Check if $g^z = f \cdot h^c$

Soundness

Prover

Extractor

f →

← c_1

$c_1 \xleftarrow{\$} \mathbb{Z}_p$

ZK Proof of Knowledge for Discrete Log

$x = (\mathbb{G}, g, p, h)$

s

Prover

Verifier

$r \xleftarrow{\$} \mathbb{Z}_p$

$f = g^r$ →

← c

$c \xleftarrow{\$} \mathbb{Z}_p$

$z := (r + c \cdot s) \bmod p$

z →

Check if $g^z = f \cdot h^c$

Soundness

Prover

Extractor

f →

← c_1

$c_1 \xleftarrow{\$} \mathbb{Z}_p$

z_1 →

$g^{z_1} = f \cdot h^{c_1}$

ZK Proof of Knowledge for Discrete Log

$x = (\mathbb{G}, g, p, h)$

s

Prover

Verifier

$r \xleftarrow{\$} \mathbb{Z}_p$

$f = g^r$ →

← c

$c \xleftarrow{\$} \mathbb{Z}_p$

$z := (r + c \cdot s) \bmod p$

z →

Check if $g^z = f \cdot h^c$

Soundness

Prover

(f, c_1, z_1)

$g^{z_1} = f \cdot h^{c_1}$

Extractor

f →

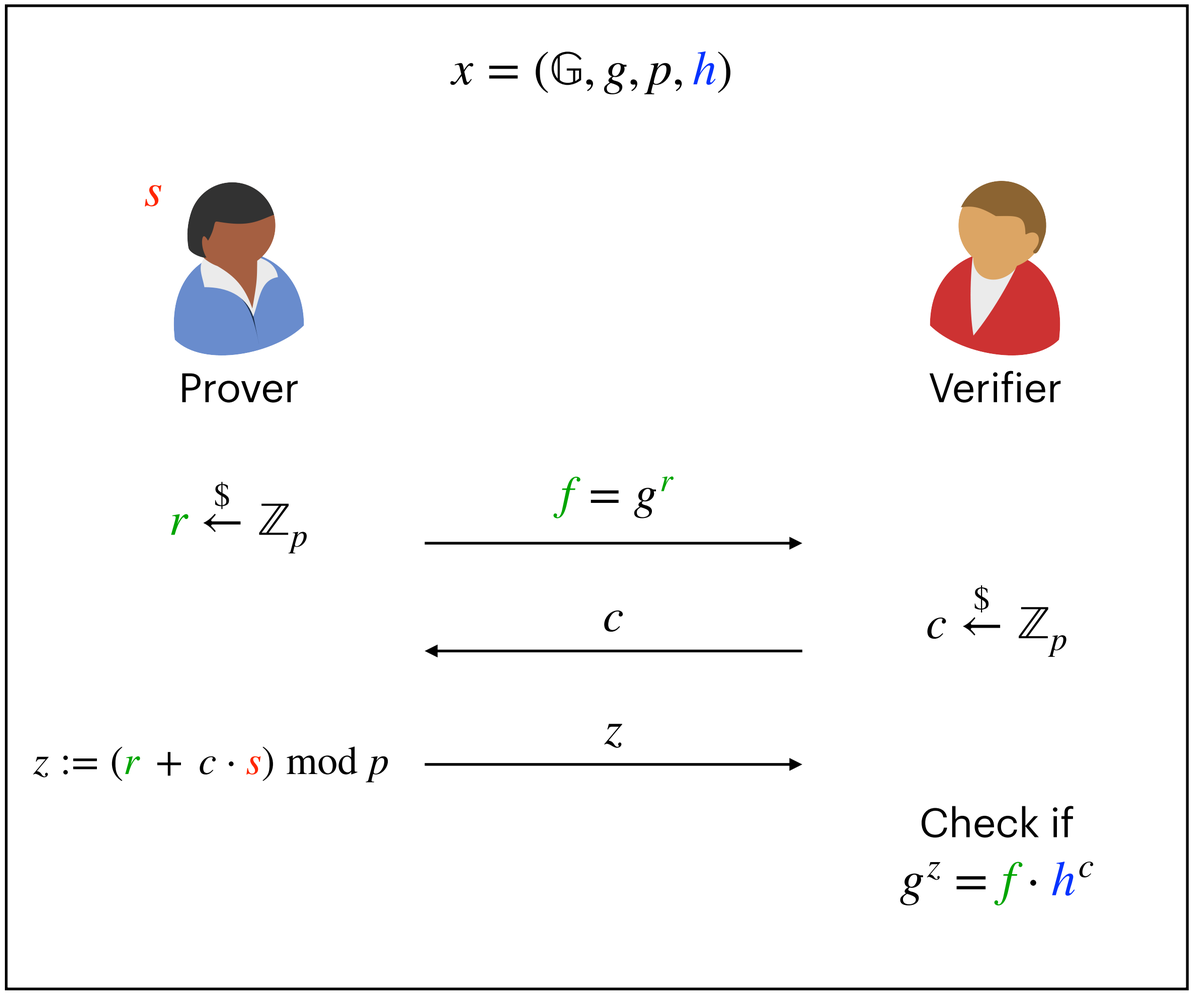
← c_1

$c_1 \xleftarrow{\$} \mathbb{Z}_p$

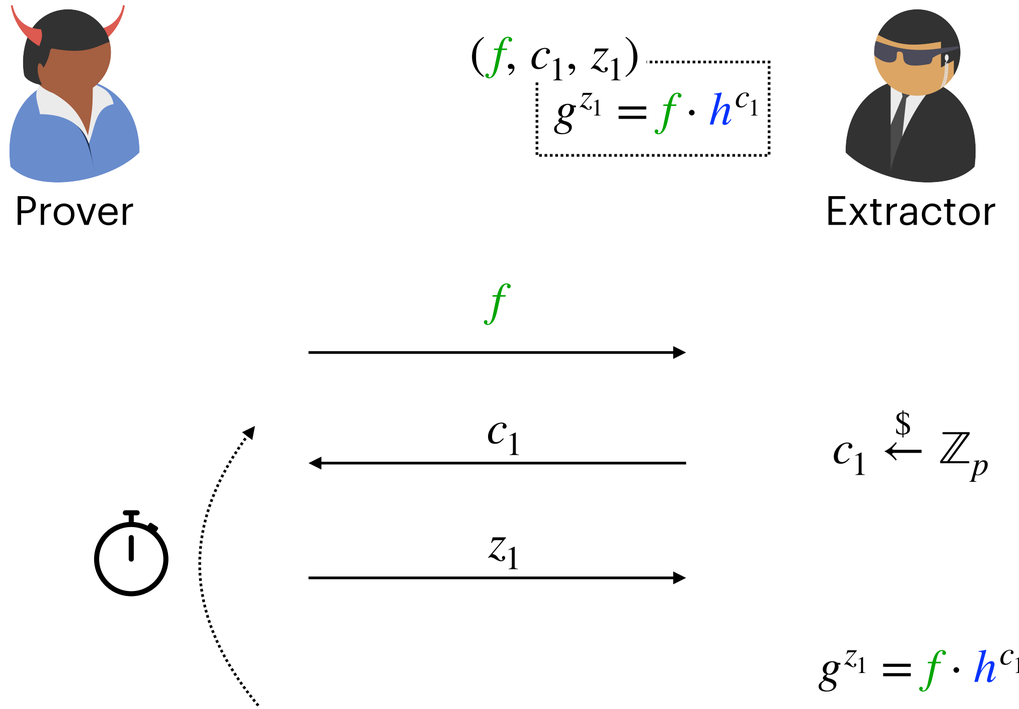
z_1 →

$g^{z_1} = f \cdot h^{c_1}$

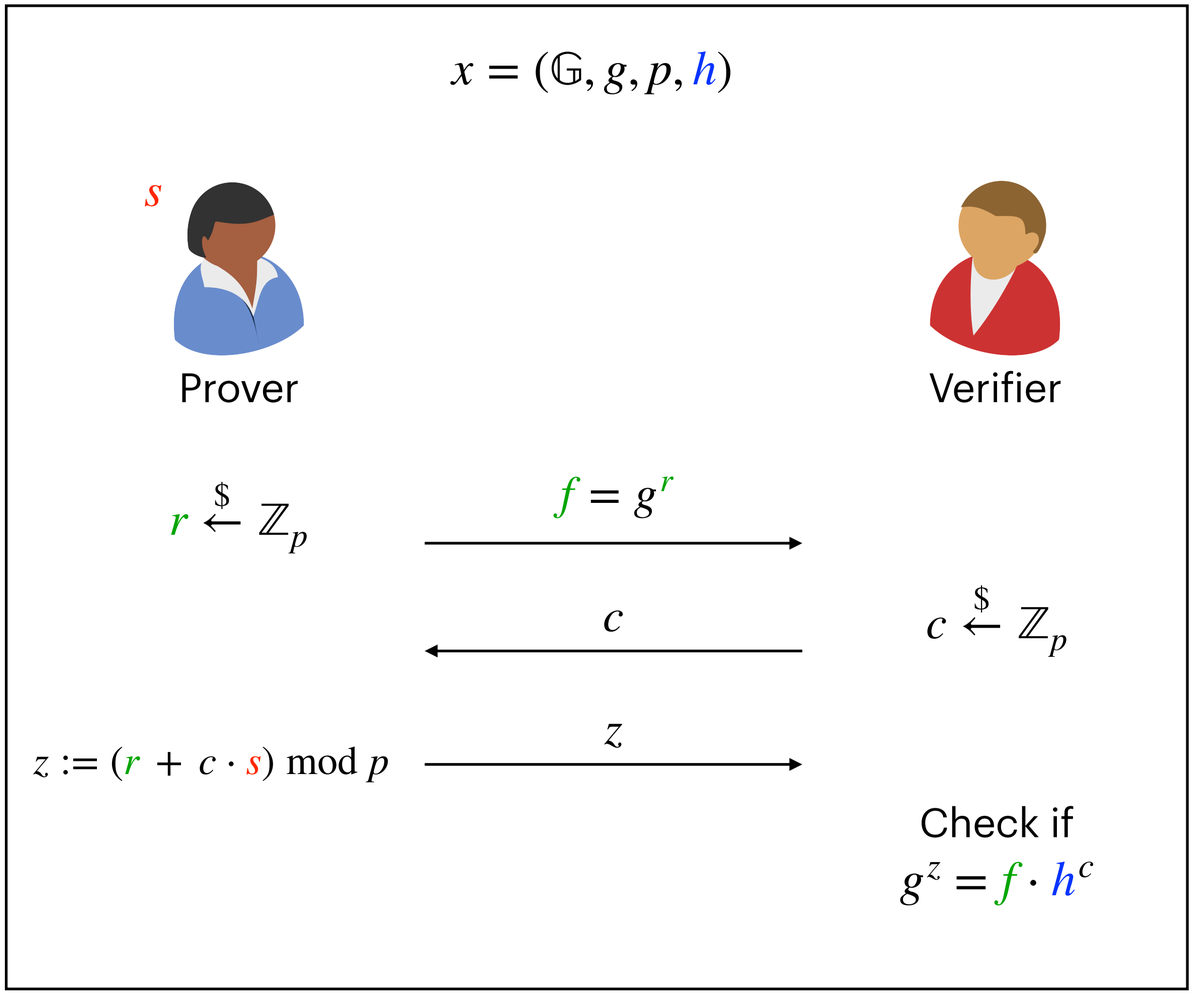
ZK Proof of Knowledge for Discrete Log



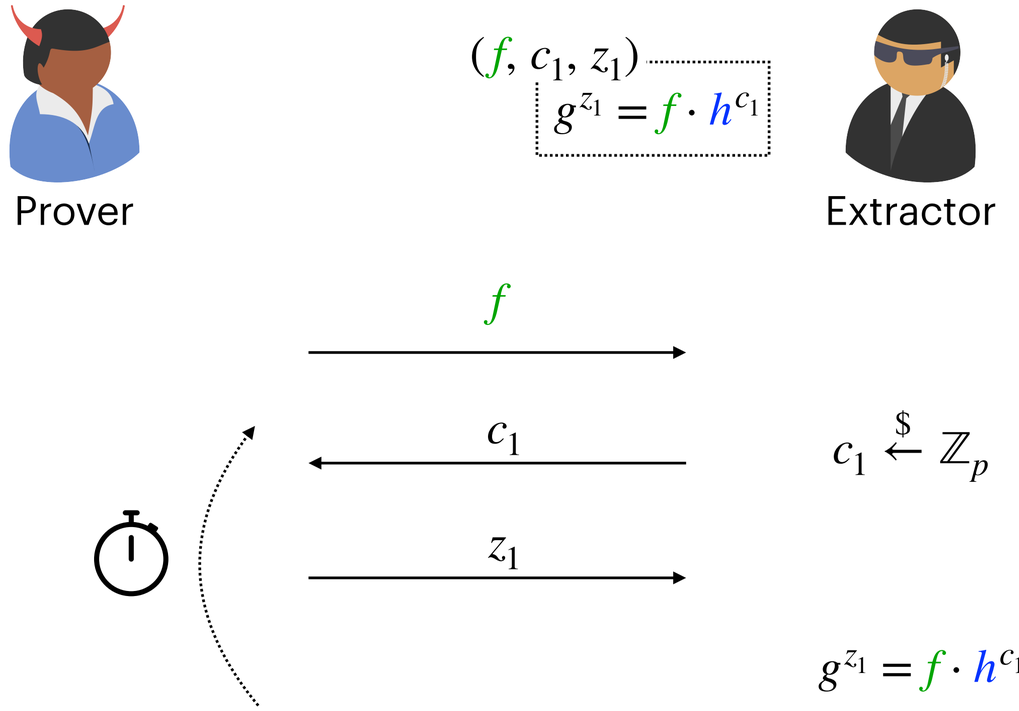
Soundness



ZK Proof of Knowledge for Discrete Log



Soundness



Rewind prover to the first message.

The prover is simply a program. A snapshot of its state can be saved and copied at any point during its execution.

ZK Proof of Knowledge for Discrete Log

$x = (\mathbb{G}, g, p, h)$

s

Prover

Verifier

$r \xleftarrow{\$} \mathbb{Z}_p$

$f = g^r$ →

← c

$c \xleftarrow{\$} \mathbb{Z}_p$

$z := (r + c \cdot s) \bmod p$

z →

Check if $g^z = f \cdot h^c$

Soundness

Prover

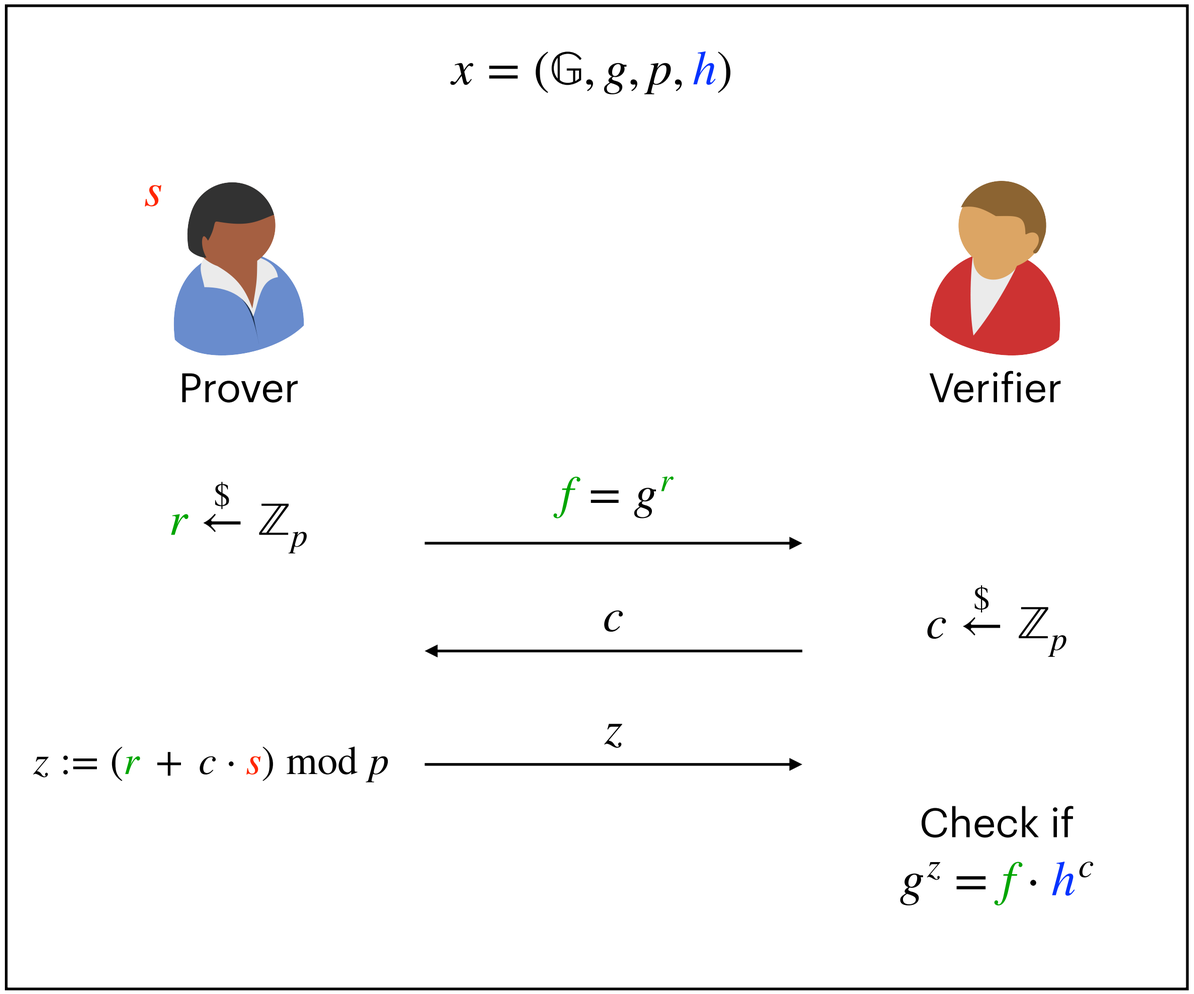
f →

(f, c_1, z_1)

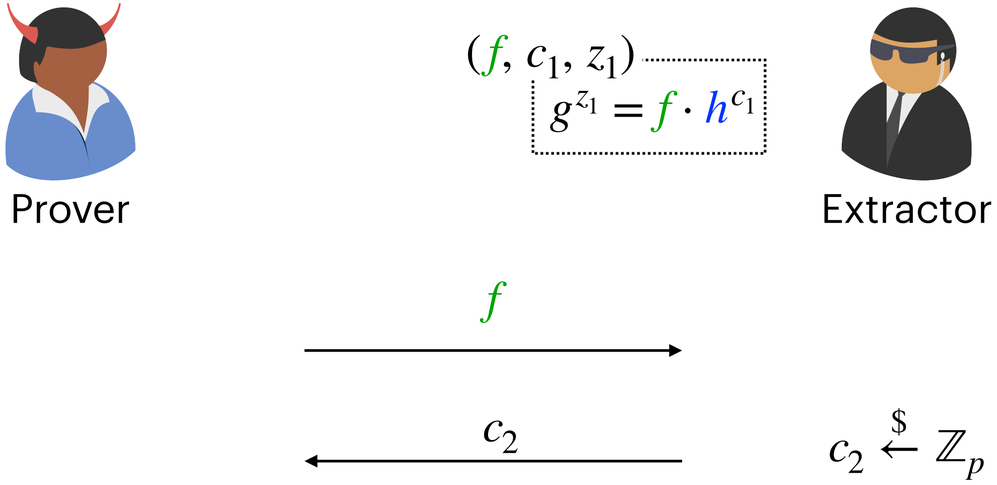
$g^{z_1} = f \cdot h^{c_1}$

Extractor

ZK Proof of Knowledge for Discrete Log



Soundness



ZK Proof of Knowledge for Discrete Log

$x = (\mathbb{G}, g, p, h)$

s

Prover

Verifier

$r \xleftarrow{\$} \mathbb{Z}_p$

$f = g^r$ →

← c

$c \xleftarrow{\$} \mathbb{Z}_p$

$z := (r + c \cdot s) \bmod p$

z →

Check if $g^z = f \cdot h^c$

Soundness

Prover

(f, c_1, z_1)

$g^{z_1} = f \cdot h^{c_1}$

Extractor

f →

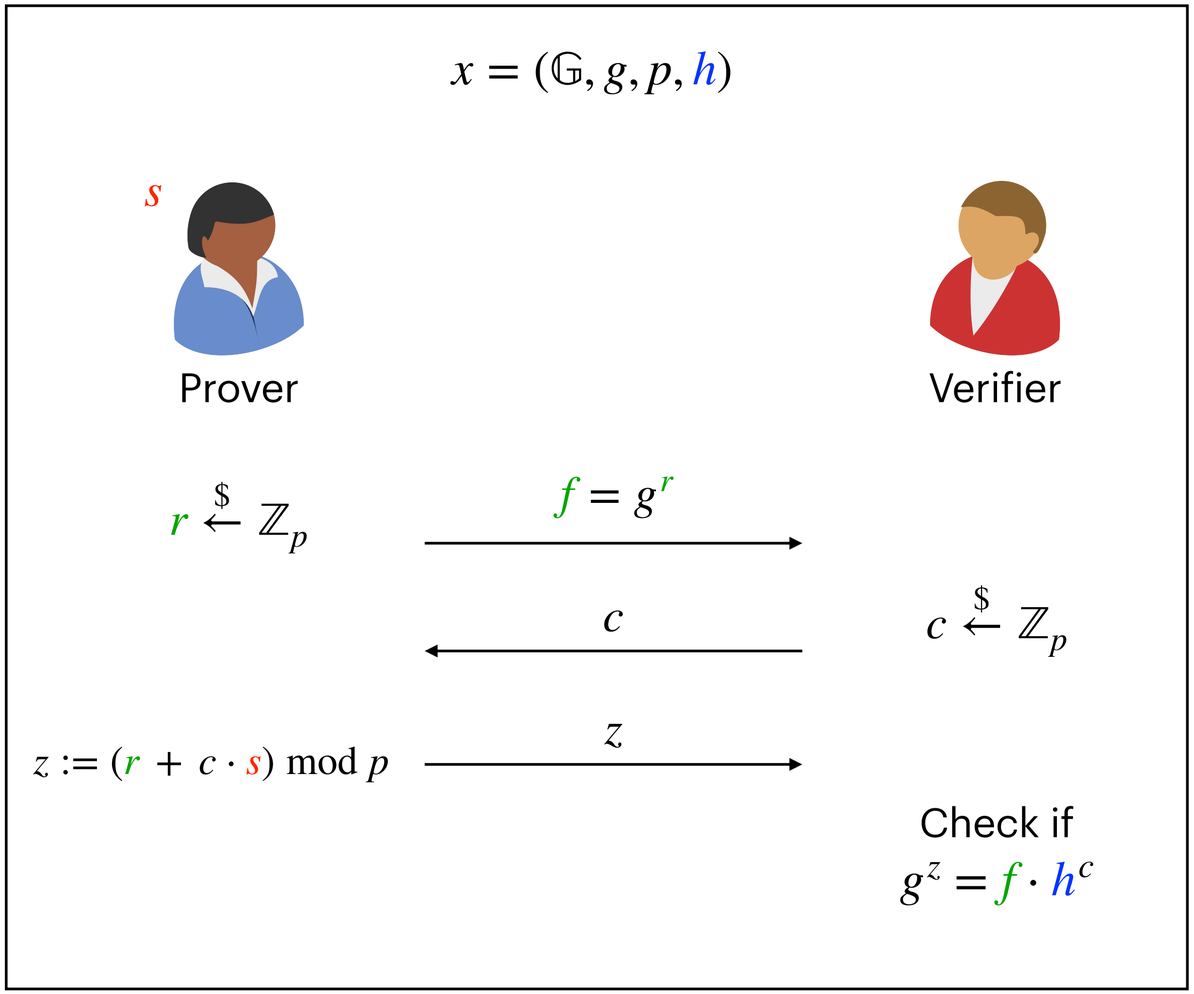
← c_2

$c_2 \xleftarrow{\$} \mathbb{Z}_p$

z_2 →

$g^{z_2} = f \cdot h^{c_2}$

ZK Proof of Knowledge for Discrete Log



Soundness


Prover


Extractor

(f, c_1, z_1)
$g^{z_1} = f \cdot h^{c_1}$
(f, c_2, z_2)
$g^{z_2} = f \cdot h^{c_2}$

f

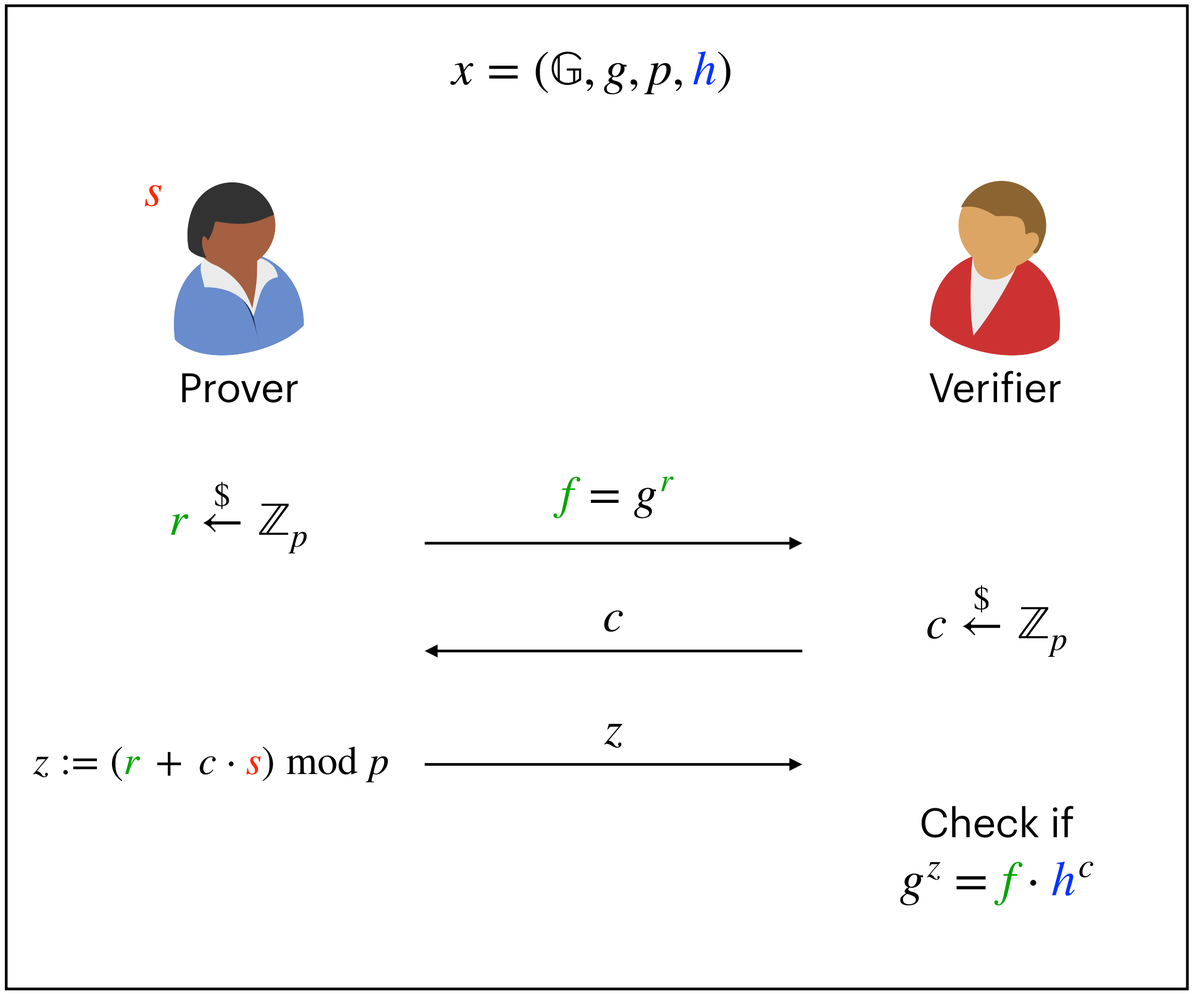
c_2

$c_2 \xleftarrow{\$} \mathbb{Z}_p$

z_2

$g^{z_2} = f \cdot h^{c_2}$

ZK Proof of Knowledge for Discrete Log




Prover

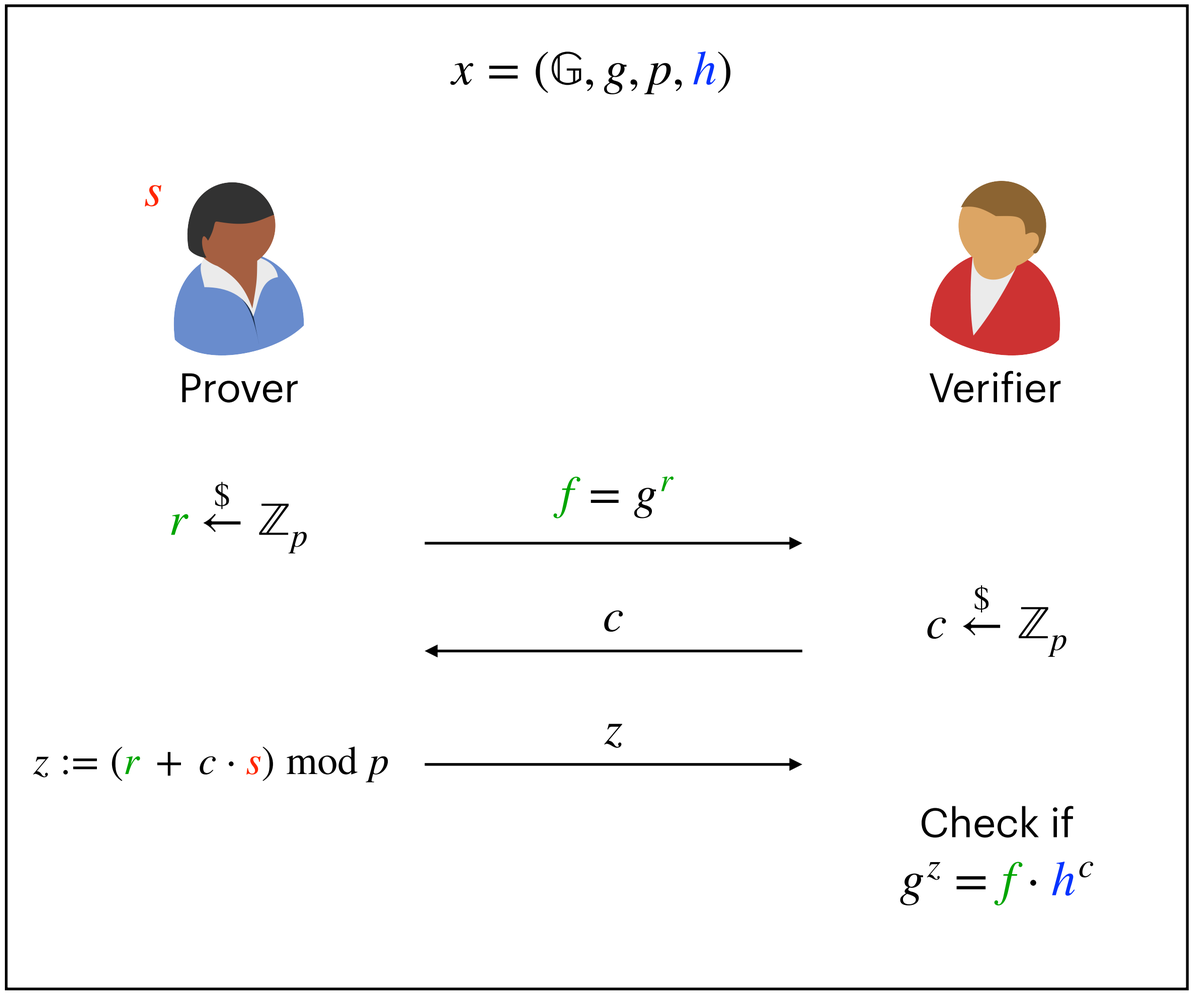
Soundness

(f, c_1, z_1)
$g^{z_1} = f \cdot h^{c_1}$

(f, c_2, z_2)
$g^{z_2} = f \cdot h^{c_2}$


Extractor

ZK Proof of Knowledge for Discrete Log




Prover

Soundness

(f, c_1, z_1)
$g^{z_1} = f \cdot h^{c_1}$

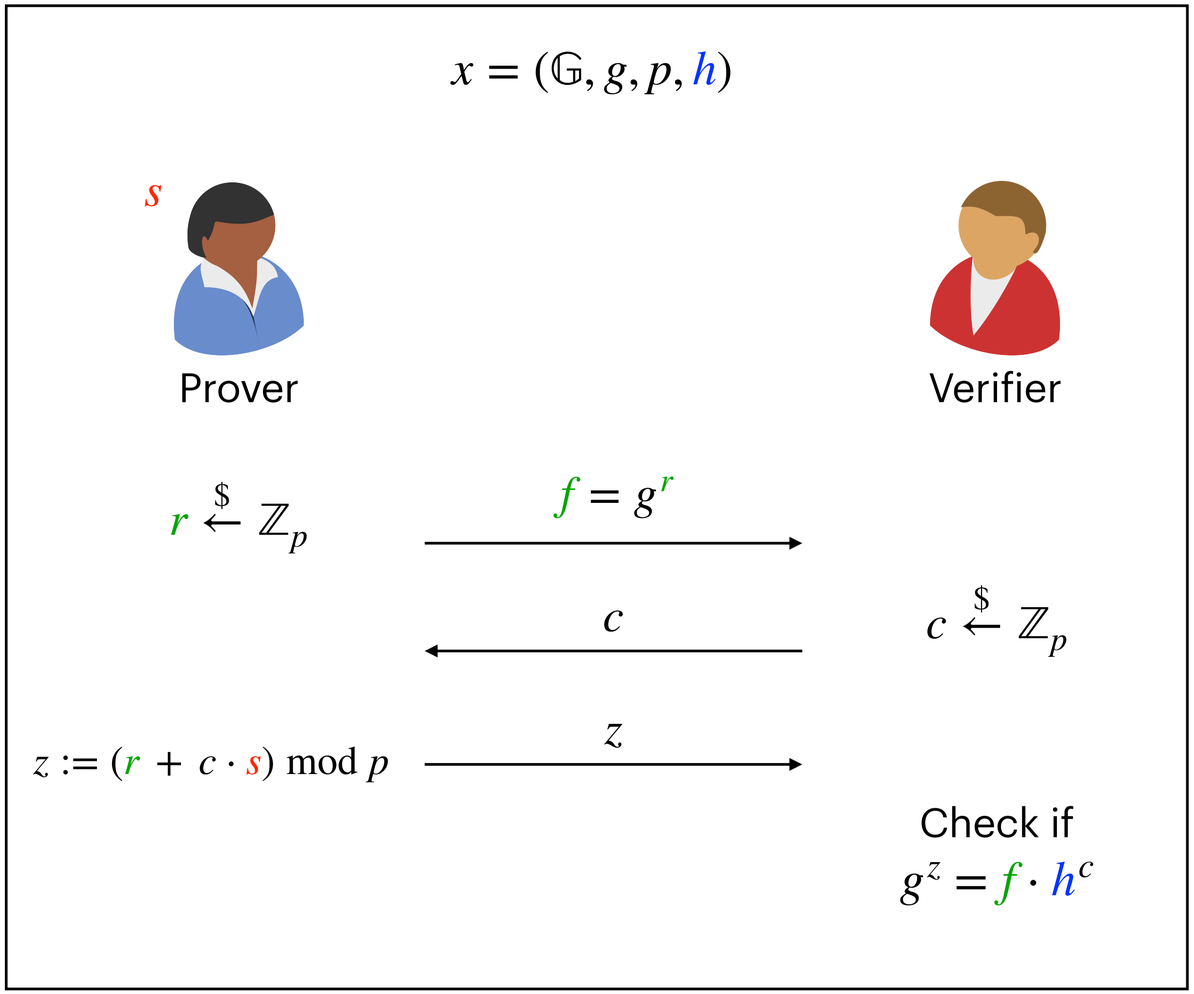
(f, c_2, z_2)
$g^{z_2} = f \cdot h^{c_2}$


Extractor

$\Downarrow$

$g^{z_1 - z_2} = h^{c_1 - c_2}$

ZK Proof of Knowledge for Discrete Log




Prover

Soundness

(f, c_1, z_1) $g^{z_1} = f \cdot h^{c_1}$

(f, c_2, z_2) $g^{z_2} = f \cdot h^{c_2}$


Extractor

$\Downarrow$

$g^{z_1 - z_2} = h^{c_1 - c_2}$

Output $s = \dfrac{z_1 - z_2}{c_1 - c_2} \pmod p$

ZK Proof of Knowledge for Discrete Log

$x = (\mathbb{G}, g, p, h)$

Prover

Verifier

s

$r \xleftarrow{\$} \mathbb{Z}_p$

$f = g^r$ →

← c

$c \xleftarrow{\$} \mathbb{Z}_p$

$z := (r + c \cdot s) \bmod p$

z →

Check if $g^z = f \cdot h^c$

Prover

Soundness

(f, c_1, z_1) $g^{z_1} = f \cdot h^{c_1}$

(f, c_2, z_2) $g^{z_2} = f \cdot h^{c_2}$

Extractor

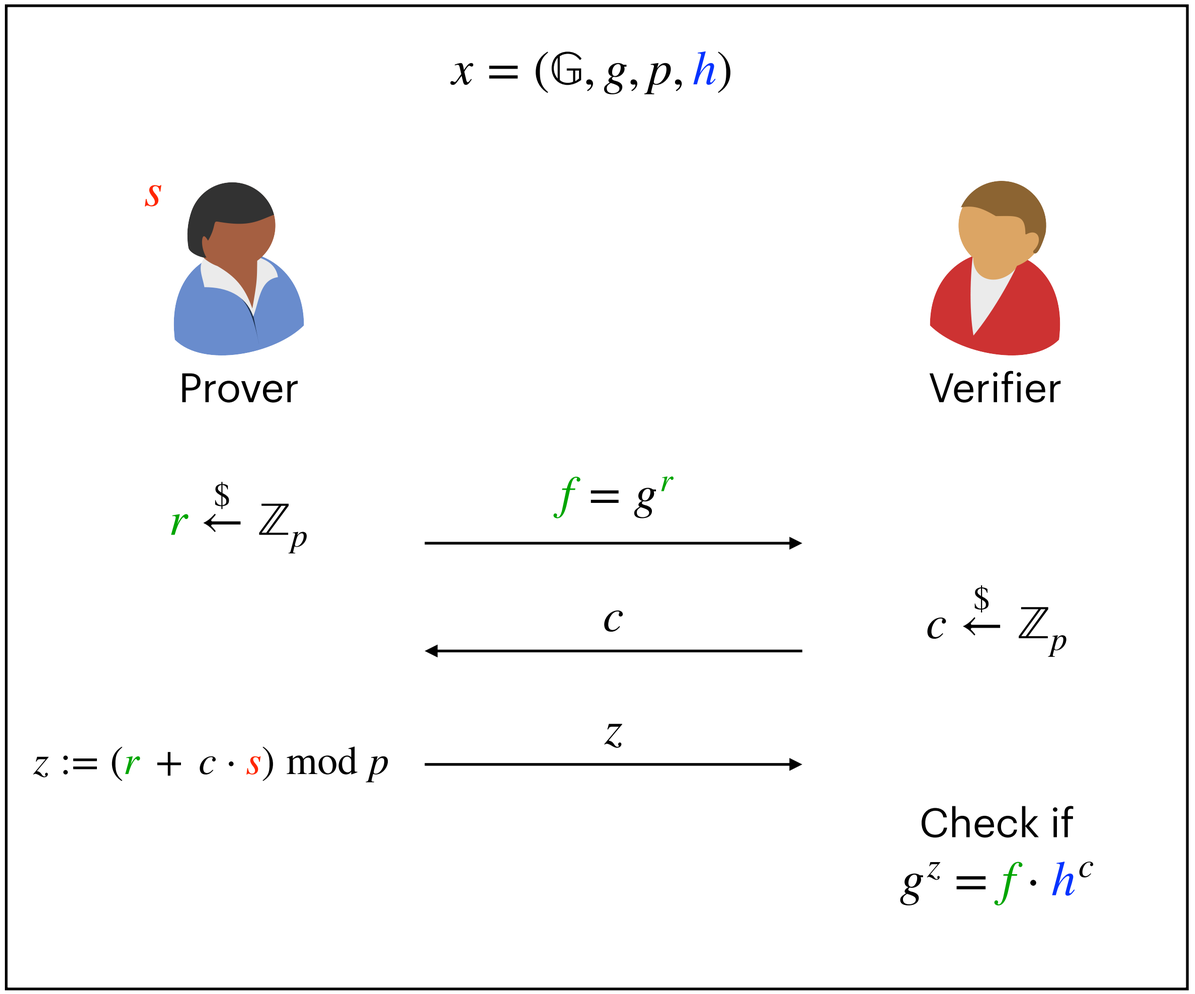
$\Downarrow$

$g^{z_1 - z_2} = h^{c_1 - c_2}$

Output $s = \dfrac{z_1 - z_2}{c_1 - c_2} \pmod p$

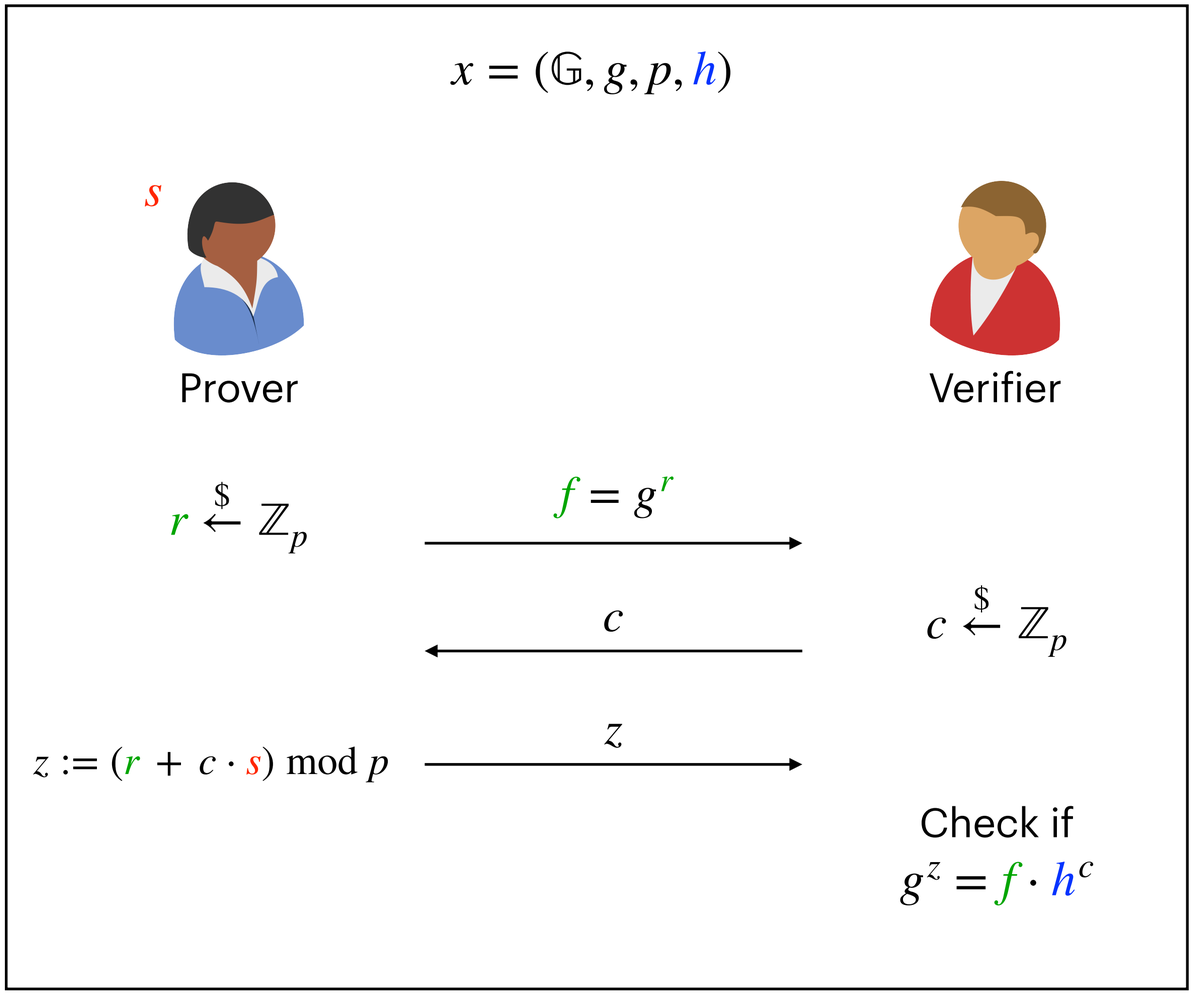
Since $p \in O(2^\lambda)$ probability that $c_1 = c_2$ is negligible.

NIZK Proof of Knowledge for Discrete Log



This proof is public-coin!

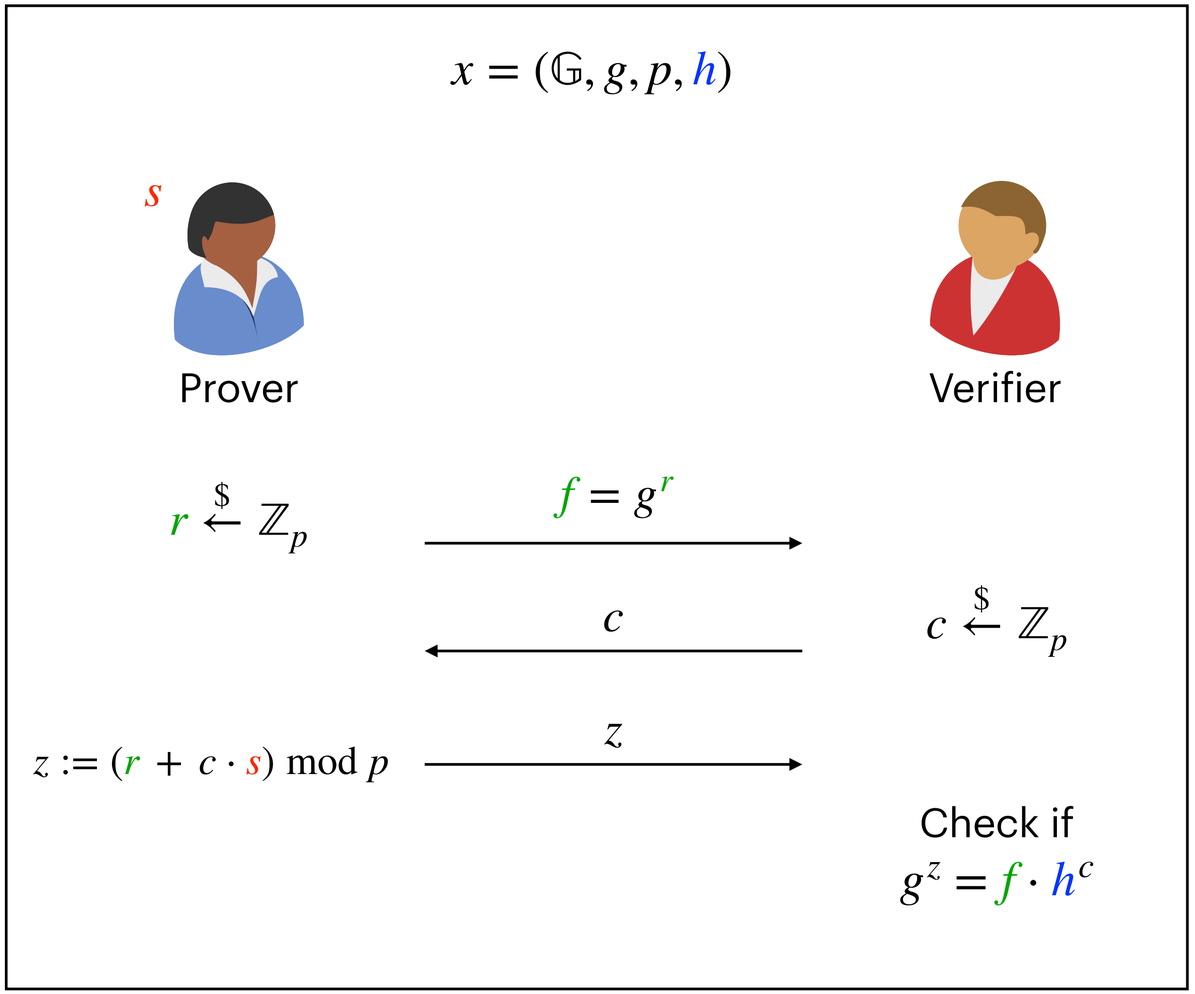
NIZK Proof of Knowledge for Discrete Log



This proof is public-coin!

We can use Fiat-Shamir to transform it into a NIZK in the random oracle model.

NIZK Proof of Knowledge for Discrete Log



This proof is public-coin!

We can use Fiat-Shamir to transform it into a NIZK in the random oracle model.

The NIZK will in fact be a proof of knowledge!

NIZK Proof of Knowledge for Discrete Log

$x = (\mathbb{G}, g, p, h)$

s

Prover

Verifier

$r \xleftarrow{\$} \mathbb{Z}_p$

$f = g^r$

$c := H(x, f)$

$z := (r + c \cdot s) \bmod p$

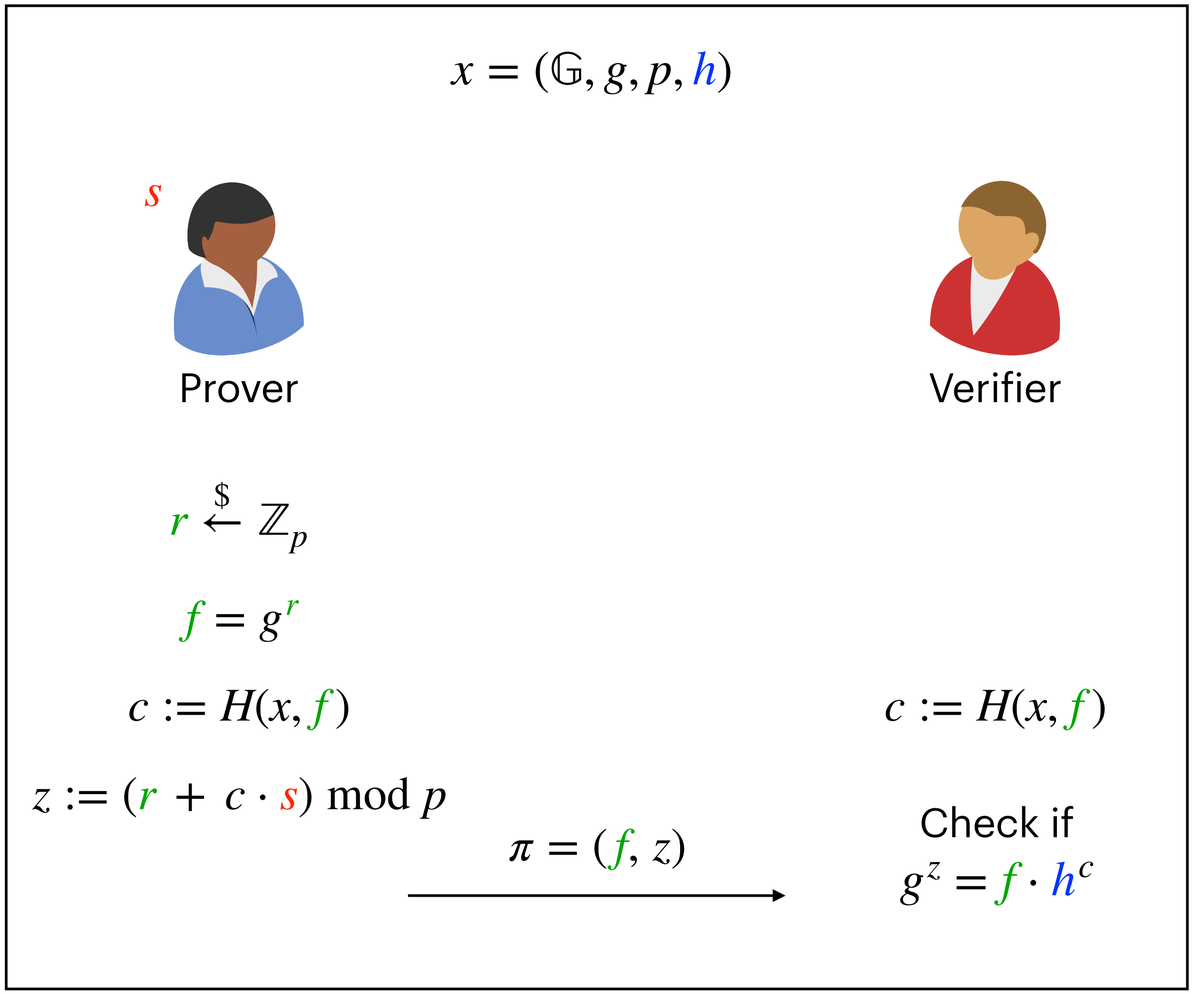
$\pi = (f, z)$

$c := H(x, f)$

Check if

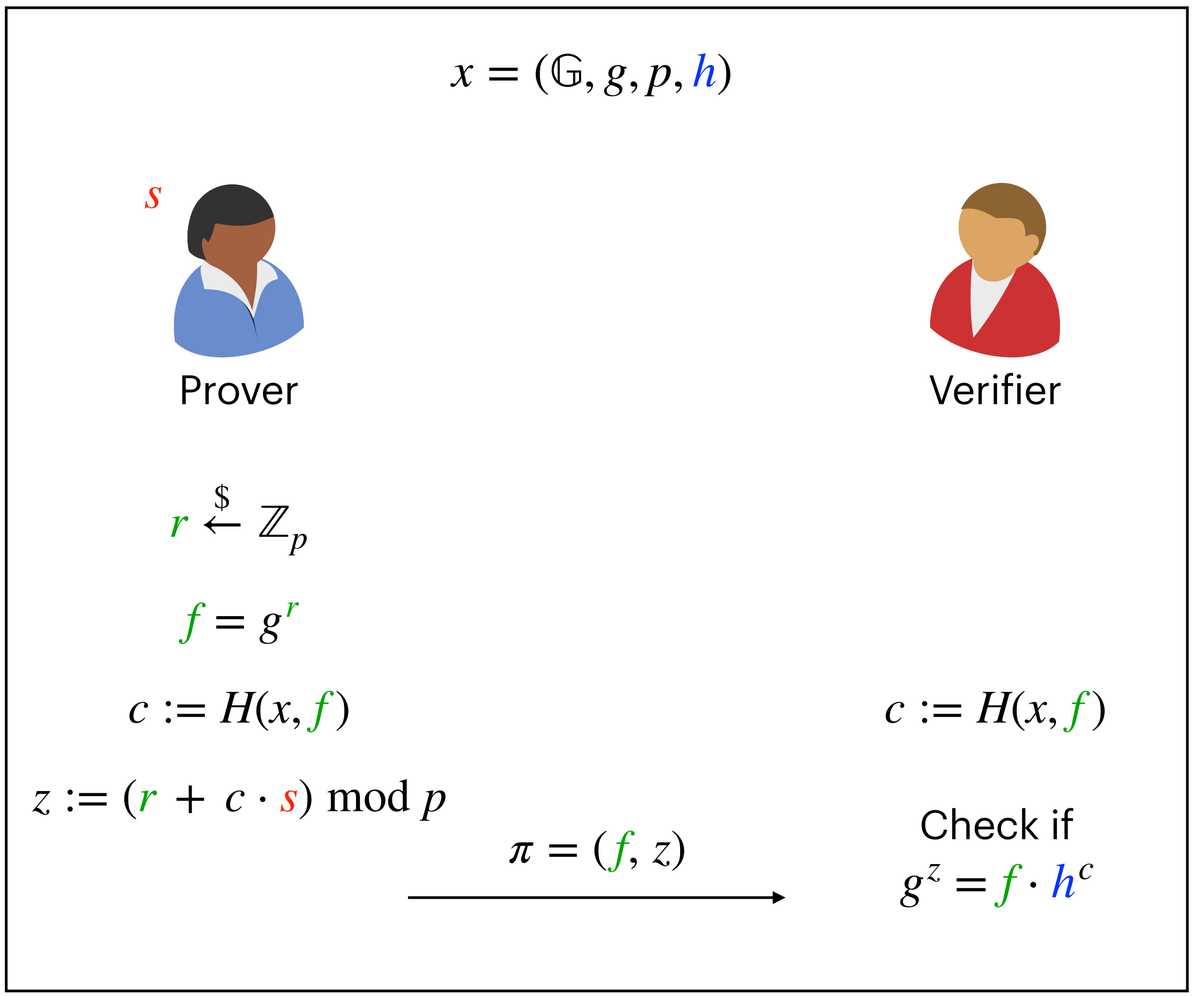
$g^z = f \cdot h^c$

NIZK Proof of Knowledge for Discrete Log



This proves that Alice knows the secret key s for the public key h.

NIZK Proof of Knowledge for Discrete Log



This proves that Alice knows the secret key s for the public key h.

Can be converted into a signature scheme by additionally including the message to be signed in the hash.

Schnorr Signature Scheme

$x = (\mathbb{G}, g, p, h)$

s

Prover

Verifier

$r \xleftarrow{\$} \mathbb{Z}_p$

$f = g^r$

$c := H(x, m, f)$ $\qquad$ $c := H(x, m, f)$

$z := (r + c \cdot s) \bmod p$

$\pi = (f, z)$ $\longrightarrow$

Check if $g^z = f \cdot h^c$

This proves that Alice knows the secret key s for the public key h.

Can be converted into a signature scheme by additionally including the message to be signed in the hash.