Zero-Knowledge Proofs

Lecture 15
Interactive Proofs
Interactive Proofs
Interactive Proofs

Prover wants to convince verifier that x has some property
**Interactive Proofs**

*Prover* wants to convince *verifier* that $x$ has some property

i.e. $x$ is in “language” $L$
Interactive Proofs

*Prover* wants to convince *verifier* that \( x \) has some property

i.e. \( x \) is in “language” \( L \)
Interactive Proofs

*Prover* wants to convince *verifier* that $x$ has some property

i.e. $x$ is in “language” $L$
Interactive Proofs

*Prover* wants to convince *verifier* that $x$ has some property

i.e. $x$ is in “language” $L$
Interactive Proofs

*Prover* wants to convince *verifier* that $x$ has some property

i.e. $x$ is in "language" $L$
Interactive Proofs

*Prover* wants to convince *verifier* that \( x \) has some property

i.e. \( x \) is in “language” \( L \)

All powerful prover, computationally bounded verifier (for now)
Interactive Proofs
Interactive Proofs

Completeness
Interactive Proofs

Completeness

If $x$ in $L$, honest Prover will convince honest Verifier
Interactive Proofs

Completeness
- If x in L, honest Prover will convince honest Verifier

Soundness
Interactive Proofs

Completeness
- If $x$ in $L$, honest Prover will convince honest Verifier

Soundness
- If $x$ not in $L$, honest Verifier won’t accept any purported proof
Interactive Proofs

Completeness
- If $x$ in $L$, honest Prover will convince honest Verifier

Soundness
- If $x$ not in $L$, honest Verifier won’t accept any purported proof
Interactive Proofs

Completeness
If $x \in L$, honest Prover will convince honest Verifier

Soundness
If $x \notin L$, honest Verifier won’t accept any purported proof
Interactive Proofs

Completeness
- If \( x \in L \), honest Prover will convince honest Verifier

Soundness
- If \( x \not\in L \), honest Verifier won’t accept any purported proof

Yeah right!
Interactive Proofs

Completeness

If $x \in L$, honest Prover will convince honest Verifier

Soundness

If $x$ not in $L$, honest Verifier won’t accept any purported proof
Interactive Proofs

Completeness
- If \( x \in L \), honest Prover will convince honest Verifier

Soundness
- If \( x \not\in L \), honest Verifier won’t accept any purported proof

\( x \in L \)

yeah right!

Reject!
An Example

Coke in bottle or can
An Example

Coke in bottle or can

Prover claims: coke in bottle and coke in can are different
An Example

Coke in bottle or can

- Prover claims: coke in bottle and coke in can are different

IP protocol:
An Example

Coke in bottle or can

Prover claims: coke in bottle and coke in can are different

IP protocol:

Pour into from can or bottle
An Example

Coke in bottle or can

- Prover claims: coke in bottle and coke in can are different

IP protocol:

Pour into from can or bottle
An Example

Coke in bottle or can

- Prover claims: coke in bottle and coke in can are different

IP protocol:
- prover tells whether cup was filled from can or bottle

Pour into from can or bottle
An Example

Coke in bottle or can

- Prover claims: coke in bottle and coke in can are different

IP protocol:

- prover tells whether cup was filled from can or bottle
An Example

Coke in bottle or can

Prover claims: coke in bottle and coke in can are different

IP protocol:

prover tells whether cup was filled from can or bottle

repeat till verifier is convinced

Pour into from can or bottle

can/bottle
An Example

Graph Non-Isomorphism

- Prover claims: $G_0$ not isomorphic to $G_1$

IP protocol:

- prover tells whether $G^*$ is an isomorphism of $G_0$ or $G_1$
- repeat till verifier is convinced

Set $G^*$ to be $\pi(G_0)$ or $\pi(G_1)$ ($\pi$ random)
An Example

Graph Non-Isomorphism

- Prover claims: $G_0$ not isomorphic to $G_1$

IP protocol:

- prover tells whether $G^*$ is an isomorphism of $G_0$ or $G_1$
- repeat till verifier is convinced

Isomorphism: Same graph can be represented as a matrix in different ways:

$$
\begin{align*}
0 & 1 & 0 & 1 \\
0 & 1 & 0 & 1
\end{align*}
$$

e.g., $G_0 = \begin{pmatrix} 1 & 0 & 0 & 1 \\ 0 & 0 & 0 & 1 \end{pmatrix}$ & $G_1 = \begin{pmatrix} 1 & 0 & 1 & 1 \\ 0 & 1 & 0 & 0 \\ 1 & 1 & 1 & 0 \\ 1 & 1 & 0 & 0 \end{pmatrix}$

both are isomorphic to the graph represented by the drawing
An Example

Graph Non-Isomorphism

Prover claims: $G_0$ not isomorphic to $G_1$

IP protocol:

- prover tells whether $G^*$ is an isomorphism of $G_0$ or $G_1$
- repeat till verifier is convinced

Isomorphism: Same graph can be represented as a matrix in different ways:

\[
\begin{pmatrix}
0 & 1 & 0 & 1 \\
0 & 1 & 0 & 1 \\
0 & 0 & 0 & 1 \\
1 & 1 & 1 & 0
\end{pmatrix}
\]

\[
\begin{pmatrix}
0 & 1 & 0 & 1 \\
0 & 1 & 0 & 1 \\
0 & 0 & 1 & 0 \\
1 & 1 & 0 & 0
\end{pmatrix}
\]

Both are isomorphic to the graph represented by the drawing.
Graph Non-Isomorphism

Prover claims: $G_0$ not isomorphic to $G_1$

IP protocol:
- prover tells whether $G^*$ is an isomorphism of $G_0$ or $G_1$
- repeat till verifier is convinced

Isomorphism: Same graph can be represented as a matrix in different ways:

$$
\begin{pmatrix}
0 & 1 & 0 & 1 \\
1 & 0 & 0 & 1 \\
0 & 0 & 0 & 1 \\
1 & 1 & 1 & 0
\end{pmatrix}
\quad \text{and} \quad
\begin{pmatrix}
0 & 1 & 0 & 1 \\
0 & 0 & 0 & 1 \\
1 & 0 & 0 & 1 \\
1 & 1 & 1 & 0
\end{pmatrix}
$$

both are isomorphic to the graph represented by the drawing

Set $G^*$ to be $\pi(G_0)$ or $\pi(G_1)$ ($\pi$ random)
Proofs for NP languages

Prove to me!

$x \in L$
Prove to me! $x \in L$

Proving membership in an NP language $L$
Prove to me!

Proving membership in an **NP** language $L$

$x \in L \text{ iff } \exists w \ R(x, w) = 1 \text{ (for } R \text{ in } P)$
Proofs for NP languages

Proving membership in an NP language $L$

$x \in L$ iff $\exists w \ R(x,w) = 1$ (for $R$ in P)

e.g. Graph Isomorphism

Prove to me!
Proofs for NP languages

Proving membership in an \( \text{NP} \) language \( L \)

\[ x \in L \text{ iff } \exists w \ R(x,w)=1 \text{ (for } R \text{ in } \text{P}) \]

e.g. Graph Isomorphism

IP protocol:

Prove to me!

\( x \in L \)

w
Proofs for NP languages

Proving membership in an NP language $L$

$x \in L$ iff $\exists w \ R(x,w)=1$ (for $R$ in $P$)

- e.g. Graph Isomorphism

IP protocol:

- prover sends $w$ (non-interactive)
Prove to me!

$\forall x \in L$ iff $\exists w \ R(x,w)=1$ (for $R \in \mathbf{P}$)

e.g. Graph Isomorphism

**IP protocol:**

prover sends $w$ (non-interactive)
Prove to me!

$x \in L$ iff $\exists w \ R(x,w)=1$ (for $R$ in $P$)

e.g. Graph Isomorphism

**IP protocol:**

prover sends $w$ (non-interactive)
Proving membership in an \textbf{NP} language $L$

$x \in L$ iff \( \exists w \ R(x,w)=1 \) (for $R$ in $\mathbf{P}$)

- e.g. Graph Isomorphism

**IP protocol:**

prover sends $w$ (non-interactive)

\textbf{NP} is the class of languages which have non-interactive and deterministic proof-systems
Proofs for NP languages

Proving membership in an NP language $L$

$x \in L$ iff $\exists w \; R(x,w)=1$ (for $R$ in $P$)

- e.g. Graph Isomorphism

**IP protocol:**
- prover sends $w$ (non-interactive)

What if prover doesn’t want to reveal $w$?

**NP** is the class of languages which have non-interactive and deterministic proof-systems
Zero-Knowledge Proofs
Zero-Knowledge Proofs

Verifier should not gain *any* knowledge from the honest prover
Verifier should not gain *any* knowledge from the honest prover except whether \( x \) is in \( L \)
Zero-Knowledge Proofs

Verifier should not gain *any* knowledge from the honest prover except whether x is in L.
Zero-Knowledge Proofs

Verifier should not gain *any* knowledge from the honest prover

except whether $x$ is in $L$

$x \in L$
Zero-Knowledge Proofs

Verifier should not gain \textit{any} knowledge from the honest prover except whether $x$ is in $L$
Zero-Knowledge Proofs

Verifier should not gain any knowledge from the honest prover except whether $x$ is in $L$
Zero-Knowledge Proofs

Verifier should not gain *any* knowledge from the honest prover except whether $x$ is in $L$.
Zero-Knowledge Proofs

Verifier should not gain \textit{any} knowledge from the honest prover except whether $x$ is in L.
Zero-Knowledge Proofs

Verifier should not gain *any* knowledge from the honest prover except whether \( x \) is in \( L \).

How to formalize this?

\[ x \in L \]

Prove to me! wonder what \( f(w) \) is...
Zero-Knowledge Proofs

Verifier should not gain *any* knowledge from the honest prover except whether $x$ is in $L$

How to formalize this?

Simulation!
An Example
An Example

Graph Isomorphism
An Example

Graph Isomorphism

\((G_0, G_1)\) in \(L\) iff there exists an isomorphism \(\sigma\) such that \(\sigma(G_0) = G_1\)
An Example

Graph Isomorphism

\((G_0, G_1)\) in \(L\) iff there exists an isomorphism \(\sigma\) such that \(\sigma(G_0) = G_1\)

IP protocol: send \(\sigma\)
An Example

**Graph Isomorphism**

\[(G_0, G_1) \text{ in } L \text{ iff there exists an isomorphism } \sigma \text{ such that } \sigma(G_0) = G_1\]

IP protocol: send \(\sigma\)

ZK protocol?
An Example

**Graph Isomorphism**

\((G_0, G_1)\) in L iff there exists an isomorphism \(\sigma\) such that \(\sigma(G_0) = G_1\)

- IP protocol: send \(\sigma\)
- ZK protocol?
An Example

Graph Isomorphism

$(G_0, G_1)$ in L iff there exists an isomorphism $\sigma$ such that $\sigma(G_0) = G_1$

IP protocol: send $\sigma$

ZK protocol?
An Example

Graph Isomorphism

\((G_0, G_1)\) in L iff there exists an isomorphism \(\sigma\) such that \(\sigma(G_0) = G_1\)

IP protocol: send \(\sigma\)

ZK protocol?
An Example

Graph Isomorphism

\((G_0, G_1)\) in \(L\) iff there exists an isomorphism \(\sigma\) such that \(\sigma(G_0) = G_1\)

- IP protocol: send \(\sigma\)
- ZK protocol?
An Example

Graph Isomorphism

\((G_0, G_1) \text{ in L iff there exists an isomorphism } \sigma \text{ such that } \sigma(G_0) = G_1\)

IP protocol: send \(\sigma\)

ZK protocol?

\[ G^* := \pi(G_1) \text{ (random } \pi) \]

if \(b=1\), \(\pi^* := \pi\)

if \(b=0\), \(\pi^* := \pi \circ \sigma\)

random bit \(b\)
An Example

Graph Isomorphism

\((G_0, G_1)\) in L iff there exists an isomorphism \(\sigma\) such that \(\sigma(G_0) = G_1\)

IP protocol: send \(\sigma\)

ZK protocol?

\[ G^* := \pi(G_1) \quad \text{(random } \pi) \]

\[ \text{if } b = 1, \quad \pi^* := \pi \]

\[ \text{if } b = 0, \quad \pi^* := \pi \circ \sigma \]
An Example

Graph Isomorphism

\((G_0, G_1)\) in \(L\) iff there exists an isomorphism \(\sigma\) such that \(\sigma(G_0) = G_1\)

IP protocol: send \(\sigma\)

ZK protocol?

\(G^* := \pi(G_1)\) (random \(\pi\))

if \(b=1\), \(\pi^* := \pi\)

if \(b=0\), \(\pi^* := \pi \circ \sigma\)
An Example

\[ G^* := \pi(G_1) \text{ (random } \pi) \]

if \( b = 1 \), \( \pi^* := \pi \)
if \( b = 0 \), \( \pi^* := \pi \circ \sigma \)

\( G^* = \pi^*(G_b) ? \)

random bit \( b \)
An Example

Why is this convincing?

\[ G^* := \pi(G_1) \] (random \( \pi \))

if \( b = 1 \), \( \pi^* := \pi \)
if \( b = 0 \), \( \pi^* := \pi \circ \sigma \)

\[ G^* = \pi^*(G_b)? \]
An Example

Why is this convincing?

- If prover can answer both b’s for the same G* then $G_0 \sim G_1$

$G^* := \pi(G_1)$ (random $\pi$)

if $b=1$, $\pi^* := \pi$
if $b=0$, $\pi^* := \pi \circ \sigma$

$G^* = \pi^*(G_b)$?
An Example

Why is this convincing?

- If prover can answer both b’s for the same $G^*$ then $G_0 \sim G_1$
- Otherwise, testing on a random b will leave prover stuck w.p. 1/2
An Example

Why is this convincing?

- If prover can answer both $b$’s for the same $G^*$ then $G_0 \sim G_1$
- Otherwise, testing on a random $b$ will leave prover stuck w.p. 1/2

Why ZK?

$G^* \leftarrow \pi(G_1)$ (random $\pi$)

- if $b=1$, $\pi^* := \pi$
- if $b=0$, $\pi^* := \pi \circ \sigma$

$G^* = \pi^*(G_b)$?
**An Example**

**Why is this convincing?**
- If prover can answer both b’s for the same G* then $G_0 \sim G_1$
- Otherwise, testing on a random b will leave prover stuck w.p. 1/2

**Why ZK?**
- Verifier’s view: random b and $\pi^*$ s.t. $G^* = \pi^*(G_b)$

$G^* := \pi(G_1)$ (random $\pi$)

if $b=1$, $\pi^* := \pi$
if $b=0$, $\pi^* := \pi \circ \sigma$

$G^* = \pi^*(G_b)$?
An Example

Why is this convincing?
- If prover can answer both b's for the same $G^*$ then $G_0 \sim G_1$
- Otherwise, testing on a random b will leave prover stuck w.p. 1/2

Why ZK?
- Verifier's view: random b and $\pi^*$ s.t. $G^* = \pi^*(G_b)$
- Which he could have generated by himself (whether $G_0 \sim G_1$ or not)
Zero-Knowledge Proofs
Zero-Knowledge Proofs

Interactive Proof
Zero-Knowledge Proofs

- Interactive Proof
- Complete and Sound
Zero-Knowledge Proofs

- Interactive Proof
  - Complete and Sound

ZK Property:
Zero-Knowledge Proofs

- Interactive Proof
- Complete and Sound

ZK Property:
Zero-Knowledge Proofs

- Interactive Proof
- Complete and Sound

ZK Property:
Zero-Knowledge Proofs

Interactive Proof
- Complete and Sound

ZK Property:
Zero-Knowledge Proofs

- Interactive Proof
- Complete and Sound
- ZK Property:
Zero-Knowledge Proofs

Interactive Proof

Complete and Sound

ZK Property:

Verifier’s view could have been “simulated”
Zero-Knowledge Proofs

Interactive Proof

- Complete and Sound

ZK Property:

- Verifier’s view could have been “simulated”
Zero-Knowledge Proofs

Interactive Proof
- Complete and Sound

ZK Property:
- Verifier’s view could have been “simulated”
Zero-Knowledge Proofs

Interactive Proof

- Complete and Sound

ZK Property:

- Verifier’s view could have been “simulated”
Zero-Knowledge Proofs

Interactive Proof

Complete and Sound

ZK Property:

Verifier’s view could have been “simulated”
Zero-Knowledge Proofs

Interactive Proof
- Complete and Sound

ZK Property:
- Verifier’s view could have been “simulated”
- For every adversarial strategy, there exists a simulation strategy
ZK Property (in other pict's)

Secure (and correct) if:

∀ output of is distributed identically in REAL and IDEAL
ZK Property (in other pict's)

Secure (and correct) if:

∀ output of

is distributed identically in REAL and IDEAL
ZK Property (in other pict’s)

Secure (and correct) if:

∀ \exists s.t.

output of is distributed identically in REAL and IDEAL
ZK Property (in other pict’s)

Classical definition uses simulation only for corrupt receiver;

Secure (and correct) if:

\[ \forall x \quad \exists \quad \text{s.t.} \quad \forall \quad \text{output of is distributed identically in REAL and IDEAL} \]
ZK Property (in other pict’s)

Classical definition uses simulation only for corrupt receiver; and uses only standalone security: Environment gets only a transcript at the end.

Secure (and correct) if:

∀ \exists \text{s.t.} output of is distributed identically in REAL and IDEAL.
Secure (and correct) if:

∀ output of is distributed identically in REAL and IDEAL
SIM ZK

- SIM-ZK would require simulation also when prover is corrupt

Secure (and correct) if:
\[ \forall x, w \ni \exists s.t. \forall output of is distributed identically in REAL and IDEAL \]
SIM ZK

- SIM-ZK would require simulation also when prover is corrupt
- Then simulator is a witness extractor

Secure (and correct) if:
\[ \forall x, w \exists s.t. \forall \text{output of is distributed identically in REAL and IDEAL} \]
SIM ZK

- SIM-ZK would require simulation also when prover is corrupt
- Then simulator is a witness extractor
- Adding this (in standalone setting) makes it a Proof of Knowledge

Secure (and correct) if:
\[
\forall \exists \text{s.t.} \forall \text{output of is distributed identically in REAL and IDEAL}
\]
Results
Results

IP and ZK defined [GMR’85]
Results

- IP and ZK defined [GMR’85]
- ZK for all NP languages [GMW’86]
Results

- IP and ZK defined [GMR’85]
- ZK for all NP languages [GMW’86]
  - Assuming one-way functions exist
Results

- IP and ZK defined [GMR’85]
- ZK for all NP languages [GMW’86]
  - Assuming one-way functions exist
- ZK for all of IP [BGGHKMR’88]
Results

- IP and ZK defined [GMR’85]
- ZK for all NP languages [GMW’86]
  Assuming one-way functions exist
- ZK for all of IP [BGGHKMR’88]
  Everything that can be proven can be proven in zero-knowledge! (Assuming OWF)
Results

- IP and ZK defined [GMR’85]
- ZK for all NP languages [GMW’86]
  - Assuming one-way functions exist
- ZK for all of IP [BGGHKMR’88]
  - Everything that can be proven can be proven in zero-knowledge! (Assuming OWF)
- Variants (for NP)
Results

- IP and ZK defined [GMR’85]
- ZK for all NP languages [GMW’86]
  Assuming one-way functions exist
- ZK for all of IP [BGGHKMR’88]
  Everything that can be proven can be proven in zero-knowledge! (Assuming OWF)
- Variants (for NP)
  ZKPoK, Statistical ZK Arguments, O(1)-round ZK, ...
A ZK Proof for Graph Colorability
A ZK Proof for Graph Colorability
A ZK Proof for Graph Colorability

Uses a commitment protocol as a subroutine
A ZK Proof for Graph Colorability

Uses a commitment protocol as a subroutine
A ZK Proof for Graph Colorability

Uses a commitment protocol as a subroutine

Use random colors

G, coloring
A ZK Proof for Graph Colorability

Uses a commitment protocol as a subroutine
A ZK Proof for Graph Colorability

Uses a commitment protocol as a subroutine
A ZK Proof for Graph Colorability

Uses a commitment protocol as a subroutine

- Pick a random edge
- Distinct colors?
  - Use random colors
  - $G, \text{coloring}$
  - Reveal edge
  - Committed

Pick random edge
- Distinct colors?
A ZK Proof for Graph Colorability

Uses a commitment protocol as a subroutine.
A ZK Proof for Graph Colorability

- Uses a commitment protocol as a subroutine
- At least $1/m$ probability of catching a wrong proof

![Diagram](image-url)
A ZK Proof for Graph Colorability

- Uses a commitment protocol as a subroutine
- At least 1/m probability of catching a wrong proof
- Soundness amplification: Repeat say mk times (with independent color permutations)
A Commitment Protocol
A Commitment Protocol

Using a OWP $f$ and a hardcore predicate for it $B$
A Commitment Protocol

- Using a OWP $f$ and a hardcore predicate for it $B$
- Satisfies only classical (IND) security, in terms of hiding and binding
Using a OWP $f$ and a hardcore predicate for it $B$

Satisfies only classical (IND) security, in terms of hiding and binding
A Commitment Protocol

Using a OWP $f$ and a hardcore predicate for it $B$

Satisfies only classical (IND) security, in terms of hiding and binding
A Commitment Protocol

- Using an OWP $f$ and a hardcore predicate for it $B$
- Satisfies only classical (IND) security, in terms of hiding and binding

\[ f(x), b \oplus B(x) \]
Using a OWP $f$ and a hardcore predicate for it $B$
Satisfies only classical (IND) security, in terms of hiding and binding
A Commitment Protocol

Using a OWP $f$ and a hardcore predicate for it $B$

Satisfies only classical (IND) security, in terms of hiding and binding
Using a OWP $f$ and a hardcore predicate for it $B$

Satisfies only classical (IND) security, in terms of hiding and binding

A Commitment Protocol

$f(x)$, $b \oplus B(x)$
A Commitment Protocol

Using a OWP $f$ and a hardcore predicate for it $B$

Satisfies only classical (IND) security, in terms of hiding and binding
Using a OWP $f$ and a hardcore predicate for it $B$
Satisfies only classical (IND) security, in terms of hiding and binding
Using a OWP $f$ and a hardcore predicate for it $B$.

Satisfies only classical (IND) security, in terms of hiding and binding.
A Commitment Protocol

Using a OWP $f$ and a hardcore predicate for it

Satisfies only classical (IND) security, in terms of hiding and binding

Perfectly binding because $f$ is a permutation

Using a OWP $f$ and a hardcore predicate for it $B$

Satisfies only classical (IND) security, in terms of hiding and binding

Perfectly binding because $f$ is a permutation
A Commitment Protocol

Using a OWP $f$ and a hardcore predicate for it $B$

Satisfies only classical (IND) security, in terms of hiding and binding

Perfectly binding because $f$ is a permutation

Hiding because $B(x)$ is pseudorandom given $f(x)$
ZK Proofs: What for?
ZK Proofs: What for?

Authentication
ZK Proofs: What for?

Authentication

- Using ZK Proof of Knowledge
ZK Proofs: What for?

- Authentication
- Using ZK Proof of Knowledge
- Canonical use: As a tool in larger protocols
ZK Proofs: What for?

- Authentication
  - Using ZK Proof of Knowledge
- Canonical use: As a tool in larger protocols
  - To enforce “honest behavior” in protocols
ZK Proofs: What for?

- Authentication
  - Using ZK Proof of Knowledge

- Canonical use: As a tool in larger protocols
  - To enforce “honest behavior” in protocols
  - At each step prove in ZK it was done as prescribed
ZK Proofs: What for?

Authentication

Using ZK Proof of Knowledge

Canonical use: As a tool in larger protocols

To enforce “honest behavior” in protocols

At each step prove in ZK it was done as prescribed
ZK Proofs: What for?

Authentication

- Using ZK Proof of Knowledge

Canonical use: As a tool in larger protocols

- To enforce “honest behavior” in protocols

- At each step prove in ZK it was done as prescribed

Prove to me $x_1$ is what you should have sent me now
ZK Proofs: What for?

- Authentication
  - Using ZK Proof of Knowledge
- Canonical use: As a tool in larger protocols
  - To enforce “honest behavior” in protocols
  - At each step prove in ZK it was done as prescribed

Prove to me $x_1$ is what you should have sent me now
ZK Proofs: What for?

- Authentication
  - Using ZK Proof of Knowledge

- Canonical use: As a tool in larger protocols
  - To enforce "honest behavior" in protocols
  - At each step prove in ZK it was done as prescribed

Prove to me $x_1$ is what you should have sent me now.

OK
ZK Proofs: What for?

Authentication
- Using ZK Proof of Knowledge

Canonical use: As a tool in larger protocols
- To enforce “honest behavior” in protocols
- At each step prove in ZK it was done as prescribed

Prove to me $x_1$ is what you should have sent me now

OK
ZK Proofs: What for?

Authentication

Using ZK Proof of Knowledge

Canonical use: As a tool in larger protocols

To enforce “honest behavior” in protocols

At each step prove in ZK it was done as prescribed

Prove to me $x_1$ is what you should have sent me now

Prove $y_1$ is what...

OK
ZK Proofs: What for?

Authentication
Using ZK Proof of Knowledge

Canonical use: As a tool in larger protocols
To enforce “honest behavior” in protocols
At each step prove in ZK it was done as prescribed

Prove to me $x_1$ is what you should have sent me now

Prove $y_1$ is what...
ZK Proofs: What for?

Authentication
- Using ZK Proof of Knowledge

Canonical use: As a tool in larger protocols
- To enforce “honest behavior” in protocols
- At each step prove in ZK it was done as prescribed

Prove to me $x_1$ is what you should have sent me now

Prove $y_1$ is what...

OK
ZK Proofs: What for?

Authentication

Using ZK Proof of Knowledge

Canonical use: As a tool in larger protocols

To enforce “honest behavior” in protocols

At each step prove in ZK it was done as prescribed

Prove to me \( x_1 \) is what you should have sent me now

Prove \( y_1 \) is what...

\( x_1 \) \( y_1 \) \( x_2 \)
ZK Proofs: What for?

Authentication

Using ZK Proof of Knowledge

Canonical use: As a tool in larger protocols

To enforce “honest behavior” in protocols

At each step prove in ZK it was done as prescribed