Mise en Situation

Suppose Alice knows a secret $S$

- You want to check that Alice knows the secret
- How can Alice convince you she does?
Mise en Situation

Suppose Alice knows a secret $S$

- You want to check that Alice knows the secret
- How can Alice convince you she does?

... without actually revealing $S$!
Example 1: The Magic Cave

Consider a cave looking as follows: [Picture missing]

- Alice knows the magic word to open the door
Example 1: The Magic Cave

Consider a cave looking as follows: [Picture missing]

- Alice knows the magic word to open the door

To convince Bob that she knows the magic word

- She goes into the cave, picks a passage at random
Example 1: The Magic Cave

Consider a cave looking as follows: [Picture missing]

- Alice knows the magic word to open the door

To convince Bob that she knows the magic word

- She goes into the cave, picks a passage at random
- Bob screams to come out the left (or right) passage
Example 1: The Magic Cave

Consider a cave looking as follows: [Picture missing]

- Alice knows the magic word to open the door

To convince Bob that she knows the magic word

- She goes into the cave, picks a passage at random
- Bob screams to come out the left (or right) passage
- Alice, knowing the magic word can do it
Example 1: The Magic Cave

Consider a cave looking as follows: [Picture missing]

- Alice knows the magic word to open the door
To convince Bob that she knows the magic word
- She goes into the cave, picks a passage at random
- Bob screams to come out the left (or right) passage
- Alice, knowing the magic word can do it
- If she doesn’t know it, she has 50/50 of being right
Example 1: The Magic Cave

Consider a cave looking as follows: [Picture missing]

- Alice knows the magic word to open the door

To convince Bob that she knows the magic word

- She goes into the cave, picks a passage at random
- Bob screams to come out the left (or right) passage
- Alice, knowing the magic word can do it
- If she doesn’t know it, she has 50/50 of being right

Repeat until Bob is convinced
Example 2: Rubik’s Cube

Bob has a scrambled Rubik’s cube

- Alice knows how to unscramble that cube
Example 2: Rubik’s Cube

Bob has a scrambled Rubik’s cube

- Alice knows how to unscramble that cube

To convince Bob that she knows how to unscramble it

- Bob gives her the scrambled cube
Example 2: Rubik’s Cube

Bob has a scrambled Rubik’s cube

- Alice knows how to unscramble that cube

To convince Bob that she knows how to unscramble it

- Bob gives her the scrambled cube
- She secretly scrambles it further (remembering how)
Example 2: Rubik’s Cube

Bob has a scrambled Rubik’s cube

• Alice knows how to unscramble that cube

To convince Bob that she knows how to unscramble it

• Bob gives her the scrambled cube
• She secretly scrambles it further (remembering how)
• Bob asks her to either: unscramble the cube now, or restore the original scrambling
Example 2: Rubik’s Cube

Bob has a scrambled Rubik’s cube

- Alice knows how to unscramble that cube

To convince Bob that she knows how to unscramble it

- Bob gives her the scrambled cube
- She secretly scrambles it further (remembering how)
- Bob asks her to either: unscramble the cube now, or restore the original scrambling
- Alice can do either if she knows how to unscramble the original cube; not otherwise
Zero Knowledge Protocols

Introduced by Goldwasser, Micali, and Rackoff in 1985
- Refined and explored by Goldreich, Micali, and Wigderson in 1986

There is a constantly changing definition of zero knowledge protocols and many papers are still coming out
- We will remain informal here
The Setup

The **Prover**
- has a secret
- Usually a probabilistic polynomial time (interactive) Turing machine
  - Sometimes completely unconstrained

The **Verifier**
- Usually a probabilistic polynomial time (interactive) Turing machine

No limits on the number of rounds of communication
Properties

Completeness

• A prover who knows the secret (honest prover) can prove it with probability 1

Soundness

• The probability that a cheating prover can get away with it can be made arbitrarily small

Zero Knowledge

• If the prover knows the secret, no verifier learns anything beyond that fact
Properties

Completeness

- A prover who knows the secret (honest prover) can prove it with probability 1.

Soundness

- The probability that a cheating prover can get away with it can be made arbitrarily small.

Zero Knowledge

- If the prover knows the secret, no verifier learns anything beyond that fact.

More precisely:

- ... does not learn anything useful beyond that fact.
Applications

Zero-knowledge protocols can be used when secret knowledge too sensitive to reveal needs to be verified

- Key authentication
- PIN numbers
- Smart cards
Example 3: Discrete Log

P wants to convince V that $\alpha^k = \beta$ for some $k$ in $[0..\lambda]$

- $\alpha, \beta$ known
Example 3: Discrete Log

P wants to convince V that $\alpha^k = \beta$ for some $k$ in $[0..\lambda]$

- $\alpha$, $\beta$ known

Assume this is a group for which the discrete log problem is hard, as usual.

The secret here is the $k$
Example 3: Discrete Log

P wants to convince V that $\alpha^k = \beta$ for some $k$ in $[0..\lambda]$

- $\alpha, \beta$ known

\[
P \quad \alpha^j \quad V
\]

rand $j \in [0, \lambda - 1]$
Example 3: Discrete Log

P wants to convince V that $\alpha^k = \beta$ for some $k$ in $[0..\lambda]$

- $\alpha, \beta$ known

P \hspace{2cm} \alpha^j \hspace{2cm} V

rand $j \in [0,\lambda-1]$

You want to avoid 0 or 1 here (why?)

So pick $j \in [j_0,\lambda-1]$ where $1 < j_0 \leq \lambda-1$
Example 3: Discrete Log

P wants to convince V that $\alpha^k = \beta$ for some $k$ in $[0..\lambda]$

- $\alpha$, $\beta$ known

\[
P \xrightarrow{\alpha^j} V \xleftarrow{i}
\]

$\text{rand } j \in [0,\lambda-1]$  $\text{rand } i \in \{0,1\}$

As remarked during lecture: this should really be $i$ chosen at random in $[1..\lambda-1]$
Example 3: Discrete Log

\( P \) wants to convince \( V \) that \( \alpha^k = \beta \) for some \( k \) in \([0..\lambda]\)

- \( \alpha, \beta \) known

\[
\begin{align*}
\text{rand } j &\in [0,\lambda-1] \\
\alpha^j &\rightarrow \\
\text{rand } i &\in \{0,1\}
\end{align*}
\]

\[
\begin{align*}
j + ik \mod \lambda &\rightarrow \\
j + ik \mod \lambda &\leftarrow
\end{align*}
\]
Example 3: Discrete Log

P wants to convince V that $\alpha^k = \beta$ for some $k$ in $[0..\lambda]$

- $\alpha, \beta$ known

<table>
<thead>
<tr>
<th>P</th>
<th>$\alpha^j$</th>
<th>V</th>
</tr>
</thead>
<tbody>
<tr>
<td>rand $j \in [0, \lambda - 1]$</td>
<td>i</td>
<td>rand $i \in {0, 1}$</td>
</tr>
</tbody>
</table>

$\alpha^j \cdot \alpha^{ik} = \alpha^j \beta^i$?
Example 3: Discrete Log

P wants to convince V that $\alpha^k = \beta$ for some $k$ in $[0..\lambda]$

- $\alpha$, $\beta$ known

P

$\text{rand } j \in [0, \lambda - 1]$

$\alpha^j$

$\text{rand } i \in \{0, 1\}$

V

$j + ik \mod \lambda$

When you repeat the protocol (to help convince verifier) make sure you pick a different random $j$ every time
Example 4: Graph 3-Coloring

G a known graph, Prover has a (secret) 3-coloring

- Wants to convince Verifier she has one
Example 4: Graph 3-Coloring

G a known graph, Prover has a (secret) 3-coloring

- Wants to convince Verifier she has one

\[ \text{P} \quad \text{encrypted recoloring} \quad \text{V} \]

\text{rand recoloring of G}

(one key per node)
Example 4: Graph 3-Coloring

G a known graph, Prover has a (secret) 3-coloring

- Wants to convince Verifier she has one

Prover (P) has a (secret) 3-coloring of G. They want to convince Verifier (V) of this fact. Prover encrypts a recoloring of G, with one key per node, and sends a recoloring of an edge chosen at random in the graph. Verifier checks the recoloring and, if it is consistent, sends a confirmation back to Prover.
Example 4: Graph 3-Coloring

G a known graph, Prover has a (secret) 3-coloring

- Wants to convince Verifier she has one

<table>
<thead>
<tr>
<th>P</th>
<th>rand recoloring of G</th>
<th>encrypted recoloring</th>
<th>rand i,j ∈ Nodes</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td>(one key per node)</td>
<td></td>
</tr>
<tr>
<td>i,j</td>
<td>keys for i and j colors</td>
<td>color(i) ≠ color(j)</td>
<td></td>
</tr>
</tbody>
</table>
Example 4: Graph 3-Coloring

G a known graph, Prover has a (secret) 3-coloring

- Wants to convince Verifier she has one

When you repeat the protocol (to help convince verifier) make sure you pick a different coloring every iteration.
Example 5: Hamiltonian Path

G a known graph, Prover has a (secret) Hamiltonian path

- Wants to convince Verifier she has one
Example 5: Hamiltonian Path

G a known graph, Prover has a (secret) Hamiltonian path

- Wants to convince Verifier she has one

\[ P \quad \rightarrow \quad H \quad \rightarrow \quad V \]

rand isomorphic copy H of G
(\pi is the matching)
Example 5: Hamiltonian Path

G a known graph, Prover has a (secret) Hamiltonian path

- Wants to convince Verifier she has one

\[
P \xrightarrow{\text{rand}} H \xrightarrow{\text{choice}} V
\]

\(P\) rand isomorphic copy \(H\) of \(G\) 
(\(\pi\) is the matching)

\(H\) choice

\(V\) rand \{give me \(\pi\), give me Hamiltonian path in \(H\}\)
Example 5: Hamiltonian Path

G a known graph, Prover has a (secret) Hamiltonian path

- Wants to convince Verifier she has one

\[
P \quad \text{rand isomorphic copy } H \text{ of } G \quad \text{choice} \quad \text{rand \{give me } \pi, \text{ give me Hamiltonian path in } H\}\]

\[
\text{requested answer} \quad \text{check iso or check path}
\]
Example 5: Hamiltonian Path

Even if V knows H, it is hard to reconstruct \( \pi \) from G and H.

(Although no one knows quite how hard...)

rand \{give me \( \pi \), give me Hamiltonian path in H\}

check iso or check path

G and \( \pi \) (at) Hamiltonian path

doesn't

rand isomorphic copy H of G

(\( \pi \) is the matching)
Commitment Scheme

A key ingredient in many zero knowledge protocols

- Interesting in its own right

How do you flip a coin in real life?

(1) Bob "calls" the coin flip

(2) Alice flips the coin, and if Bob's call is correct, he wins, otherwise Alice does
Flipping a Coin Over the Phone

How do you do this over the telephone?

- Bob cannot trust Alice to reply honestly

Need **commitment**:

- A value of 0 or 1 is committed to by encrypting it or hashing it with a one-way function to get a “blob”
- We can verify the commitment by “unwrapping” this blob after revealing the key
Flipping a Coin Over the Phone

How do you do this over the telephone?

- Bob cannot trust Alice to reply honestly

1. Bob "calls" the coin flip and tells Alice only a commitment to his call
2. Alice flips the coin and reports the result
3. Bob reveals what he committed to; if that matches the coin result Alice reported, Bob wins
Flipping a Coin Over the Phone

For Alice to be able to skew the results in her favor, she must be able to understand the call hidden in Bob's commitment, so if the commitment scheme is a good one, Alice cannot affect the results.

Similarly, Bob cannot affect the result if he cannot change the value he commits to.

(3) Bob reveals what he committed to; if that matches the coin result Alice reported, Bob wins
Bit Commitment Properties

Concealment:

- Receiver cannot determine the value of the bit from the “blob”

Binding:

- Sender cannot open the “blob” as both a zero and a one
Given an instance of an NP-complete problem
Given an instance of an NP-complete problem

- Prover generates a new isomorphic instance based on the original one
ZK from NP-Complete Problems

Given an instance of an NP-complete problem

- Prover generates a new isomorphic instance based on the original one
- Prover commit the solution to the new problem to Verifier with a commitment protocol
Given an instance of an NP-complete problem

- Prover generates a new isomorphic instance based on the original one
- Prover commits the solution to the new problem to Verifier with a commitment protocol
- Verifier can challenge Prover with one of the questions:
  - Prove the two instances are isomorphic
  - Or show me the solution to the new instance
**ZK from NP-Complete Problems**

Given an instance of an NP-complete problem

- Prover generates a new isomorphic instance based on the original one
- Prover commits the solution to the new problem to Verifier with a commitment protocol
- Verifier can challenge Prover with one of the questions:
  - Prove the two instances are isomorphic
  - Or show me the solution to the new instance

As usual, repeat procedure until Verifier is satisfied.
ZK from NP-Complete Problems

Given an instance of an NP-complete problem

- Prover generates a new isomorphic instance based on the original one
- Prover commits the solution to the new problem to the Verifier with a commitment protocol
- Verifier can challenge Prover with one of the questions:
  - Prove the two instances are isomorphic
  - Or show me the solution to the new instance

Tricky bit:
Verifier should not be able to transfer a solution back to the original instance
Graph Isomorphism

$G_0$ and $G_1$ are known graphs.

Prover knows a (secret) isomorphism $\pi$ between them.
Graph Isomorphism

$G_0$ and $G_1$ are known graphs.

Prover knows a (secret) isomorphism $\pi$ between them.

$P$ wants to convince $V$ that she knows the isomorphism
Graph Isomorphism

$G_0$ and $G_1$ are known graphs.

Prover knows a (secret) isomorphism $\pi$ between them.

\[ \sigma_0 = \mu \]
\[ \sigma_1 = \mu \circ \pi^{-1} \]
Graph Isomorphism

$G_0$ and $G_1$ are known graphs.

Prover knows a (secret) isomorphism $\pi$ between them.

$P$

random $H$ and isomorphism $\mu$

between $G_0$ and $H$

$\sigma_0 = \mu$

$\sigma_1 = \mu \circ \pi^{-1}$

$H$

$V$

rand $i \in \{0,1\}$
$G_0$ and $G_1$ are known graphs.

Prover knows a (secret) isomorphism $\pi$ between them.

- Random $H$ and isomorphism $\mu$ between $G_0$ and $H$

\[
\sigma_0 = \mu \\
\sigma_1 = \mu \circ \pi^{-1}
\]

Check that $\sigma_i$ is an isomorphism between $G_i$ and $H$
More about NPC Problems

Every NPC problem yields a zero knowledge protocol

- Assumes existence of one-way functions
- Or existence of an encryption scheme
  - Basically, for commitment scheme

Variant that does not require such an assumption:

- Use multiple independent provers instead of only one, allowing the verifier to validate prover results against each other to avoid being misled.
ZK Proofs of Identity

If a private key is used as an identity, we can use a zero-knowledge proof for identity

- Chess Master problem: When Alice is proving her identity to a malicious node, the malicious node may be proving to a third party
- Cf wormhole attacks on wireless networks

Proposed solutions:

- Accurately synchronized clocks