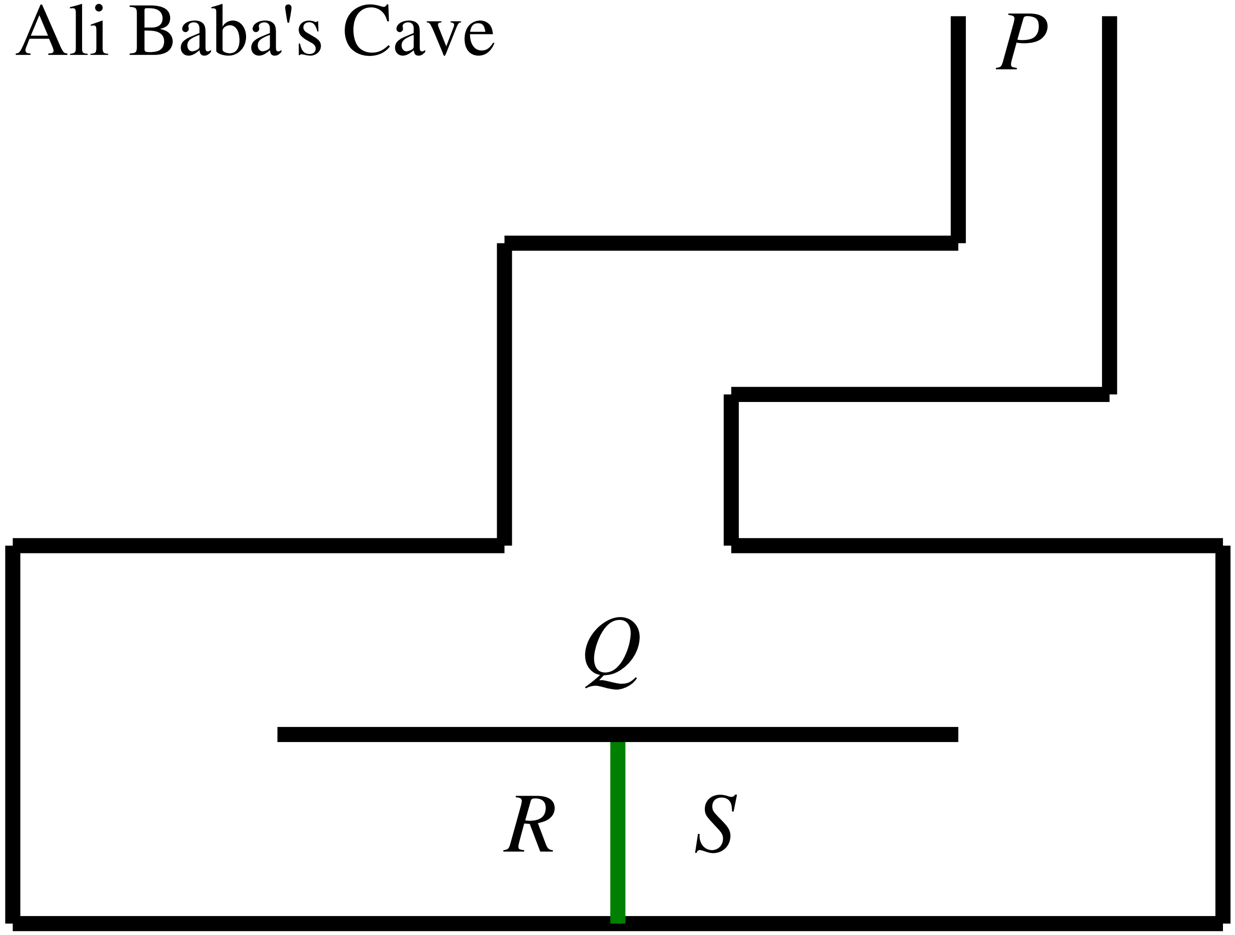
Interactive and Zero Knowledge Proofs

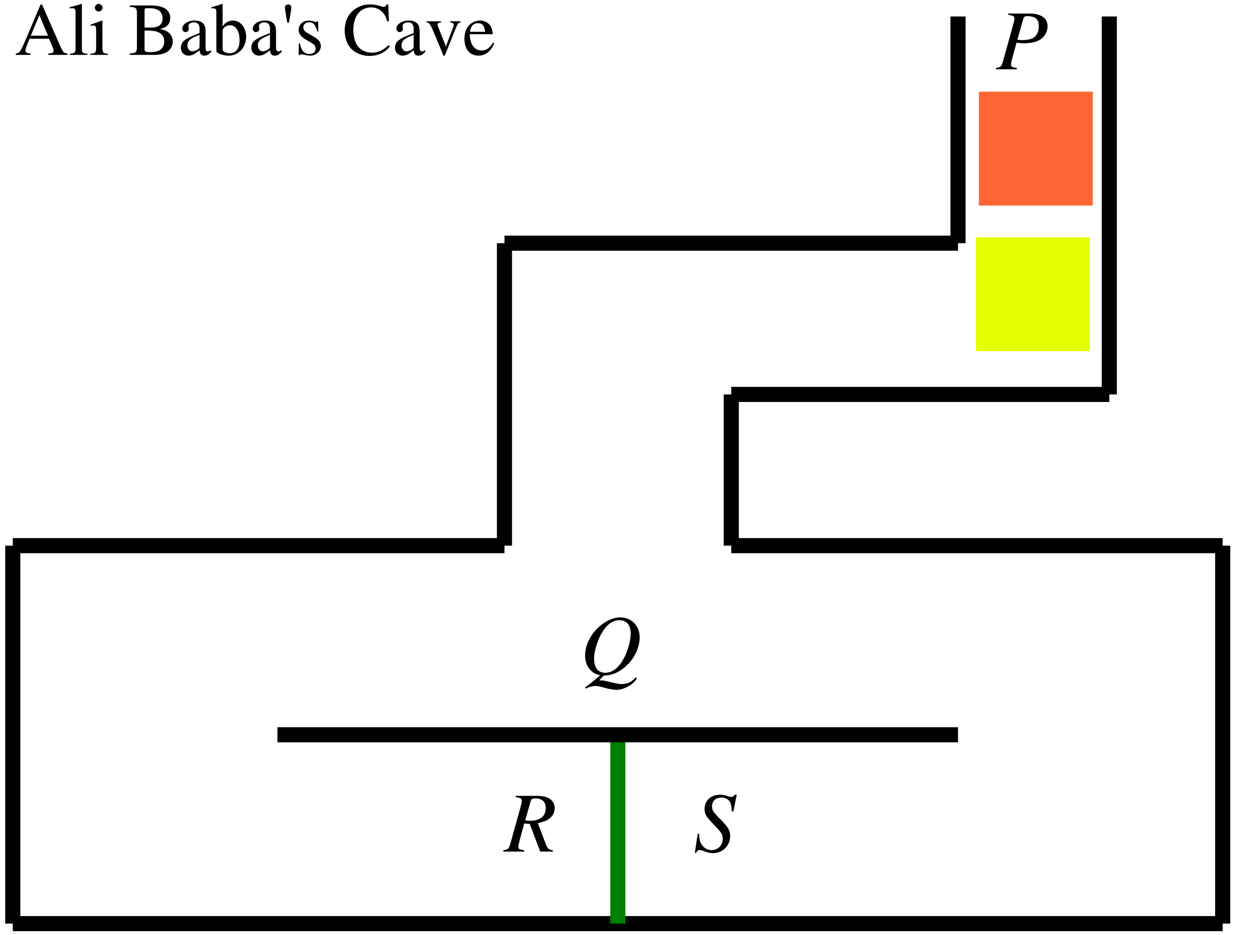
Interactive and Zero Knowledge Proofs

Example: Ali Baba's Cave



Interactive and Zero Knowledge Proofs

Example: Ali Baba's Cave

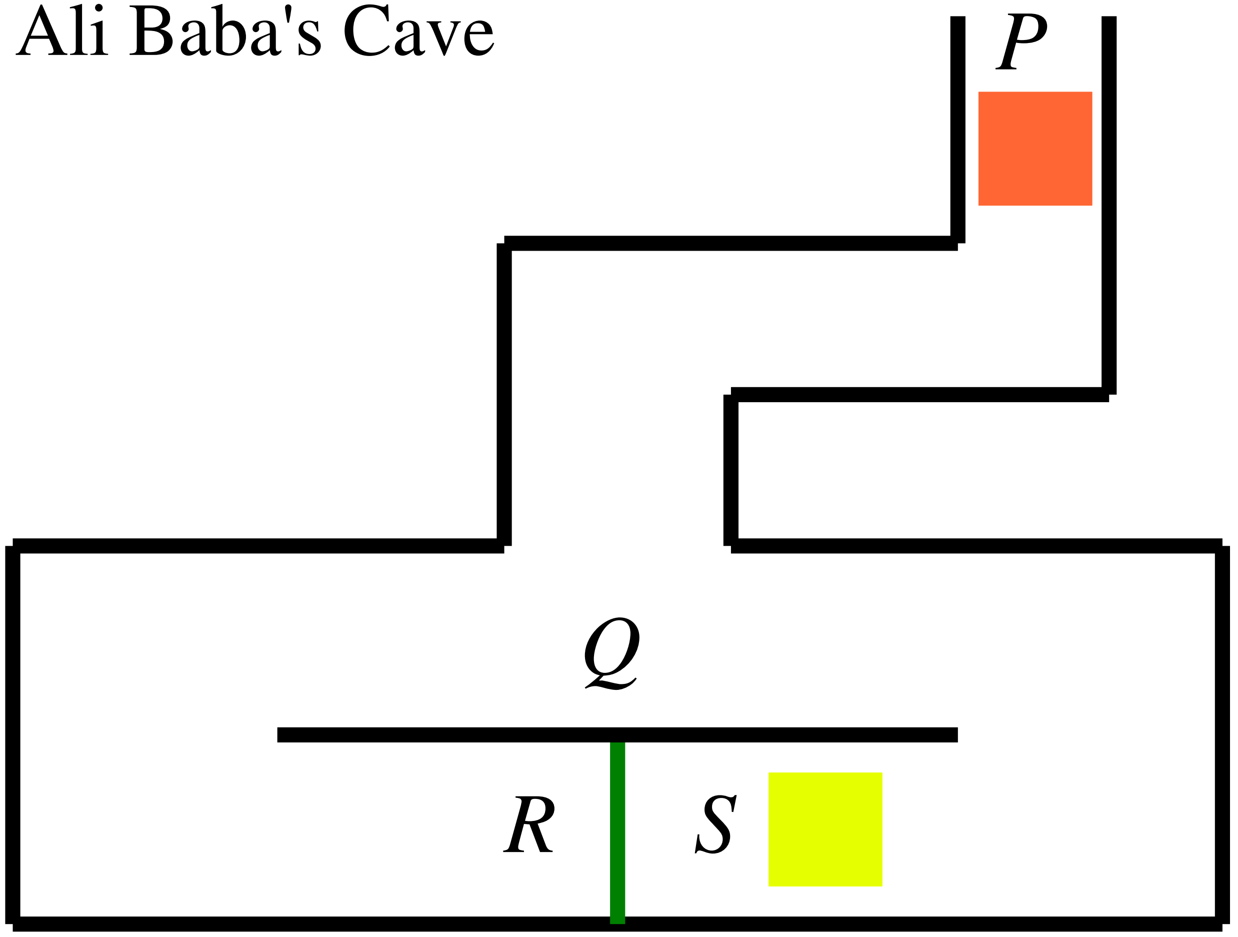


Initially:

Prover 🟨 and Verifier 🟧 at the mouth of the cave. Neither can see deep into the cave. From Q a player cannot see either R or S. Prover proves it knows the secret words that will open the door at green line, deep inside the cave, but without telling what they are.

Interactive and Zero Knowledge Proofs

Example: Ali Baba's Cave

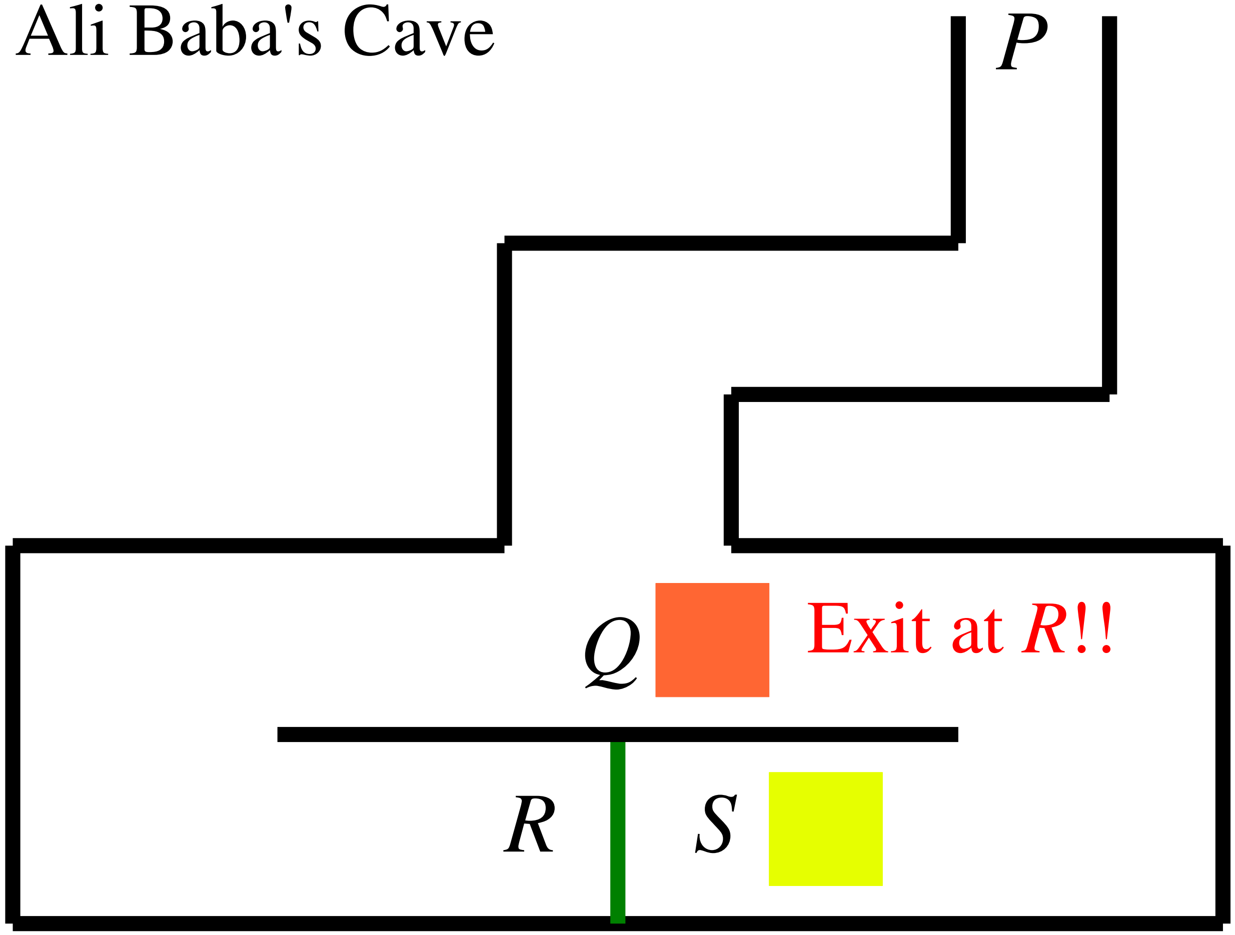


Round:

Prover's commitment is to visit R or S while verifier waits at P

Interactive and Zero Knowledge Proofs

Example: Ali Baba's Cave



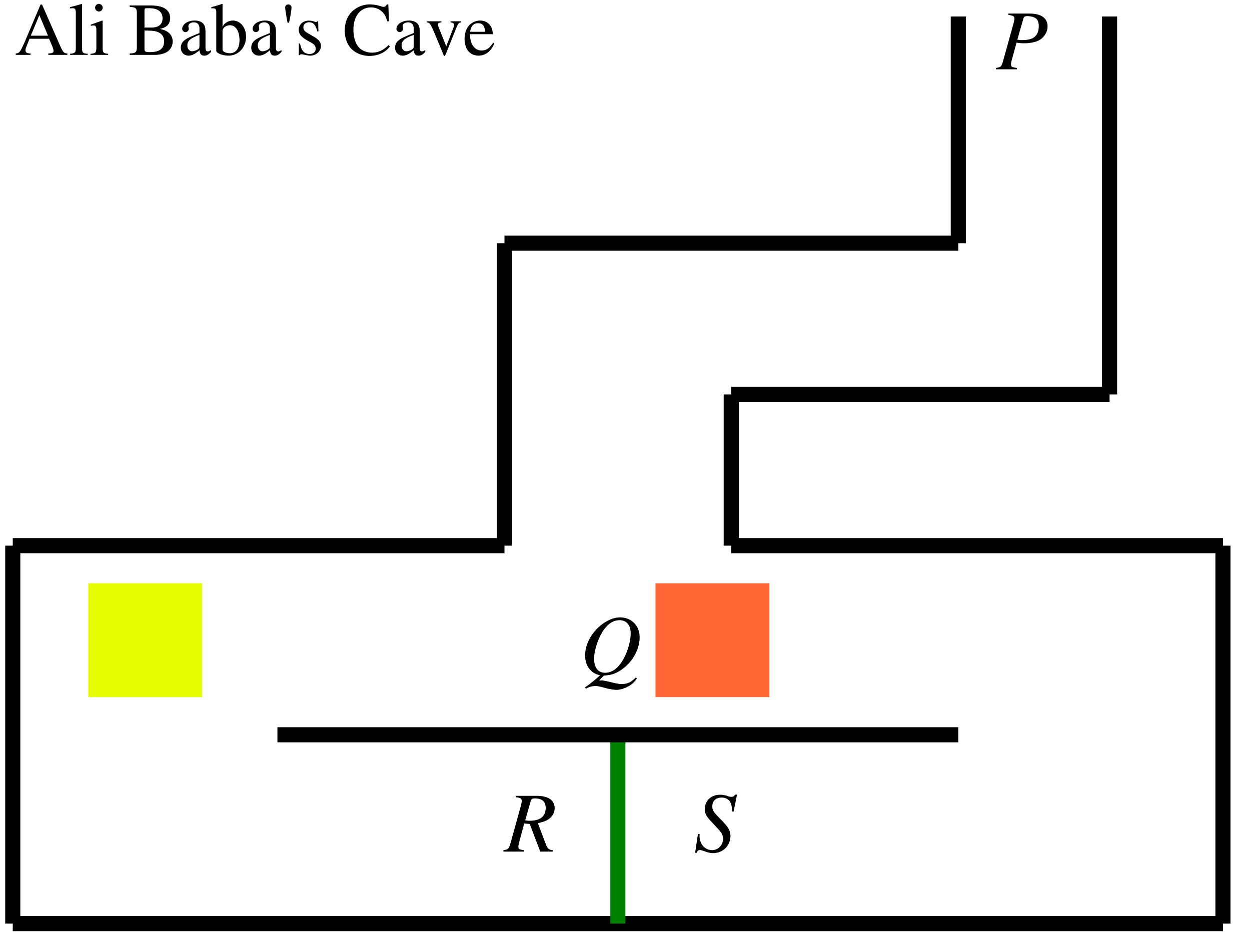
Round:

Prover's commitment is to visit *R* or *S* while verifier waits at *P*

Verifier's challenge is to walk to *Q* and ask prover to exit at *R* or *S*

Interactive and Zero Knowledge Proofs

Example: Ali Baba's Cave



Round:

Prover's commitment is to visit *R* or *S* while verifier waits at *P*

Verifier's challenge is to walk to *Q* and ask prover to exit at *R* or *S*

Prover's response is to do as verifier says

Many Rounds: certain prover does not know or pretty sure it does

Interactive and Zero Knowledge Proofs

A protocol between two parties in which one party, called the ***prover*** tries to prove a certain fact to the other party, called the ***verifier***. Used for authentication and identification.

The fact to prove usually is related to the prover's identity, say the prover's secret key.

The following properties are important:

1. **Completeness** - the verifier always accepts the proof if the fact is true and both parties follow the protocol.
2. **Soundness** - the verifier always rejects the proof if the fact is false, as long as the verifier follows the protocol.
3. **Zero-Knowledge** - verifier learns nothing else about the fact being proved from the prover that could not be learned without the prover, regardless of following the protocol. Verifier cannot even prove the fact to anyone later.

Interactive and Zero Knowledge Proofs

How do you know you have a Zero Knowledge proof?

The verifier can produce a simulation of the transactions even if the prover does not know the fact to be proved. The simulation can be handed to a third party who cannot tell whether the simulation is real or fake.

How can the verifier do that?

The verifier video tapes the transactions and throws out any bad frames and presto the rest looks to anyone like a transaction proving the fact.

Interactive and Zero Knowledge Proofs

A Round - a commitment message from the prover,
a challenge from the verifier,
a response to the challenge from the prover.

The protocol may repeat for several rounds. Based on the prover's responses in all the rounds, the verifier decides whether to accept or reject the proof.

Interactive and Zero Knowledge Proofs

How do we know Ali Baba's protocol is a zero-knowledge proof?

The proof can be performed efficiently by a simulator that has no idea of what the proof is.

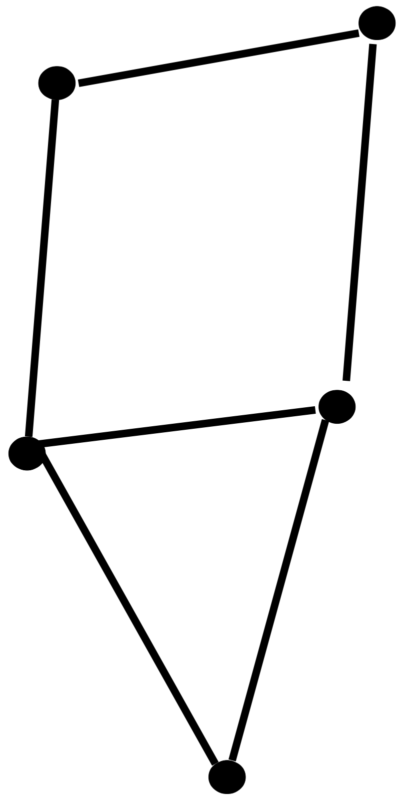
The verifier video tapes the movements of the prover and the verifier assuming the prover *does not know* the secret words.

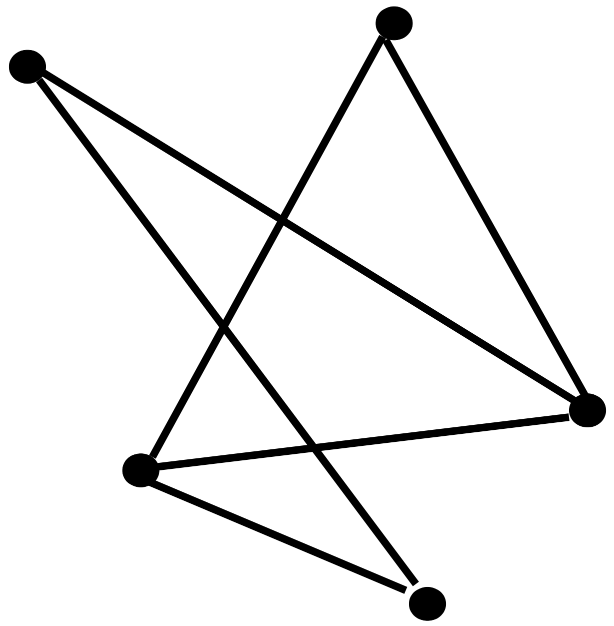
Some of the rounds show the prover unable to find the correct exit. Those rounds are deleted from the video tape.

The result is a sequence of rounds that appear to show the prover *does know* the secret words.

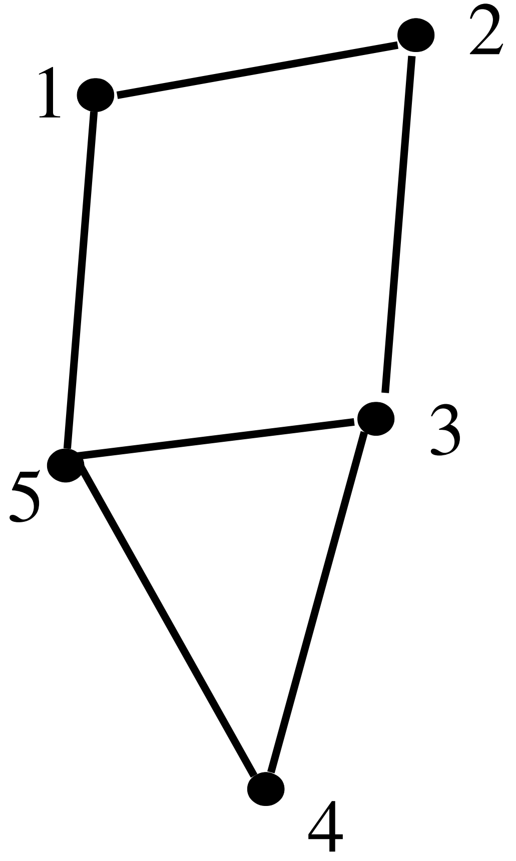
Hence no knowledge can be extracted from the video tape.
There is no knowledge in the recording of the original protocol.

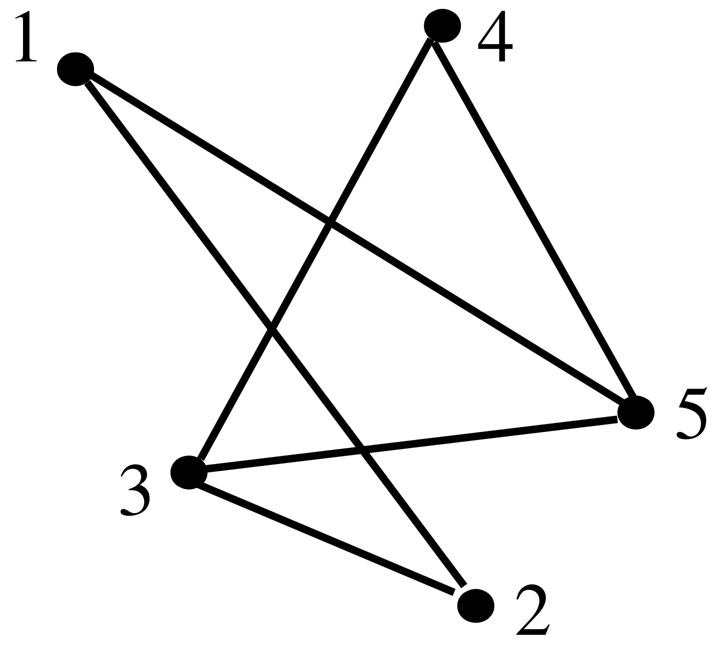
Graph Isomorphism & Zero Knowledge Proofs



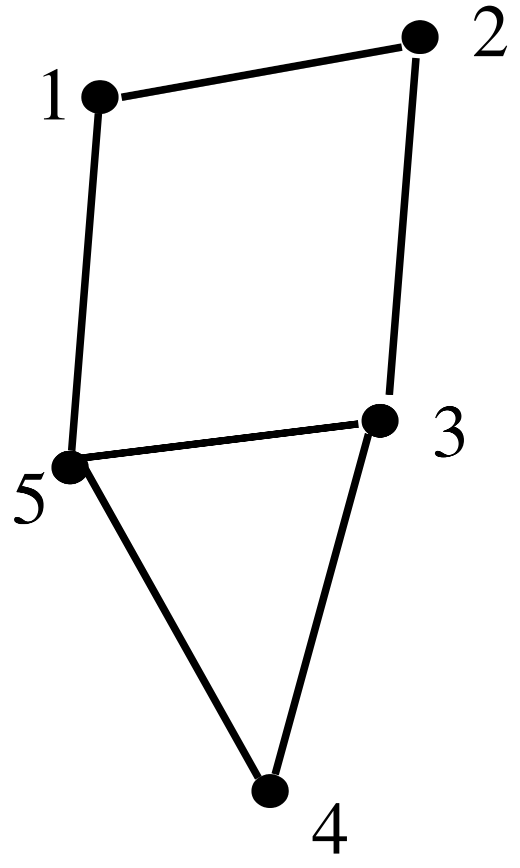


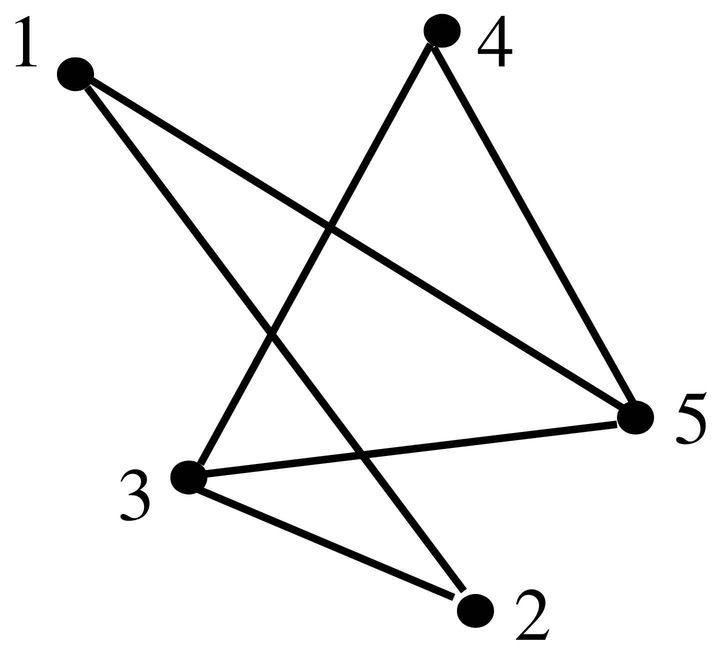
Graph Isomorphism & Zero Knowledge Proofs





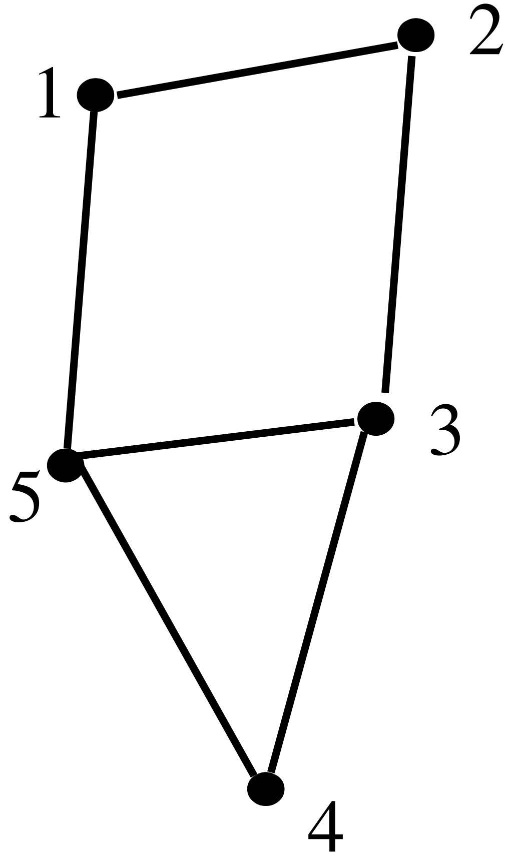
Graph Isomorphism & Zero Knowledge Proofs

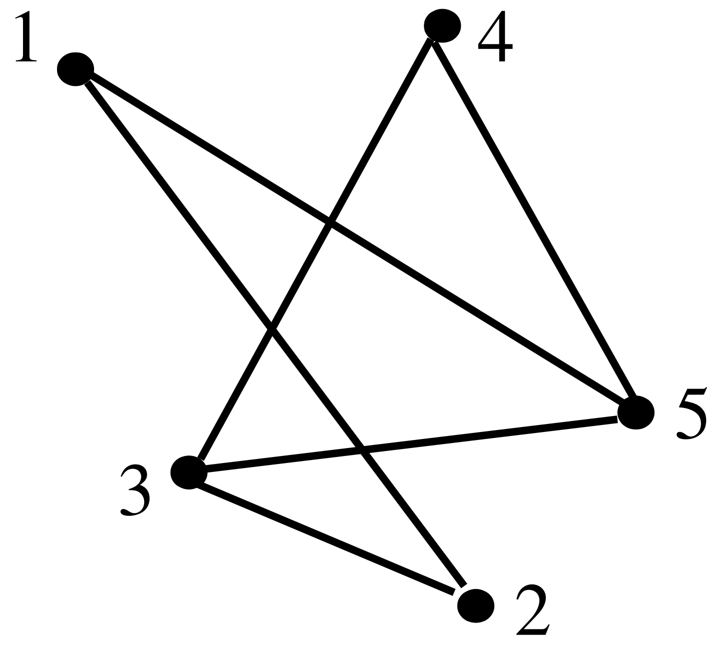




It is **really hard** to determine whether two graphs are isomorphic

Graph Isomorphism & Zero Knowledge Proofs





It is **really hard** to determine whether two graphs are isomorphic
But, if someone hands you a vertex mapping, it is easy to check!!!

Graph Isomorphism - Zero Knowledge Proof

Graph Isomorphism - Zero Knowledge Proof

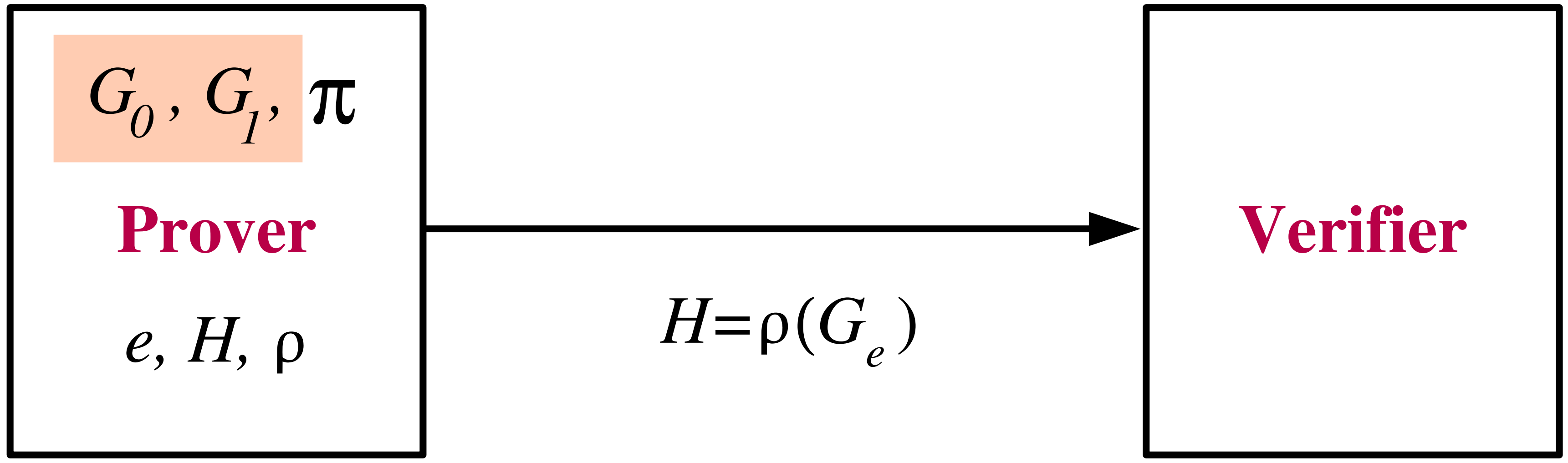
G_0 , G_1, π
Prover
Verifier

Generate two isomorphic graphs, G_0 and G_1 of n vertices,

where $G_1 = \pi(G_0)$.

Keep π secret, publish the graphs.

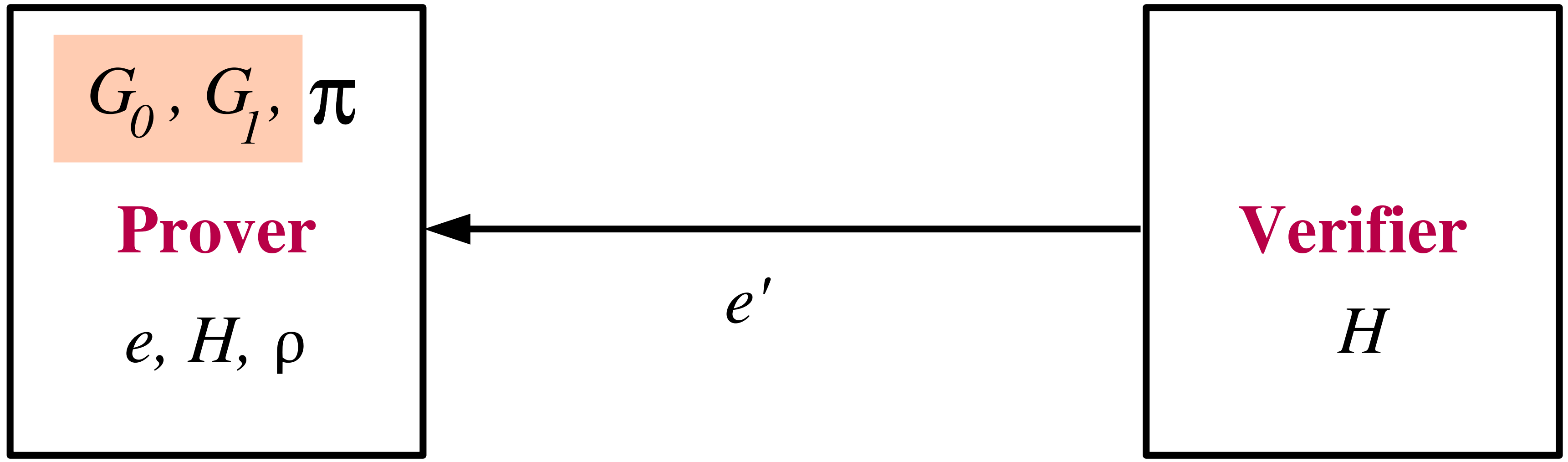
Graph Isomorphism - Zero Knowledge Proof



Protocol:

Prover: Generate 2nd perm ρ, compute $H=\rho(G_e)$, select $e \in \{0,1\}$

Graph Isomorphism - Zero Knowledge Proof

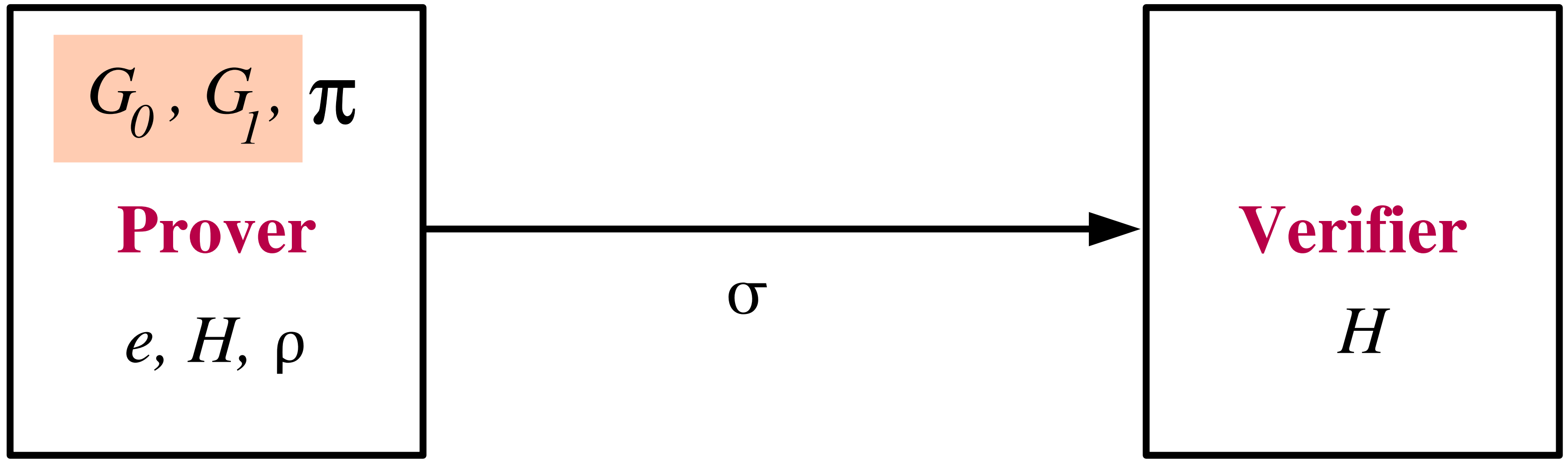


Protocol:

Prover: Generate 2nd perm ρ, compute $H = \rho(G_e)$, select $e \in \{0,1\}$

Verifier: Select $e' \in \{0,1\}$, ask prover to prove H isomorphic to $G_{e'}$

Graph Isomorphism - Zero Knowledge Proof



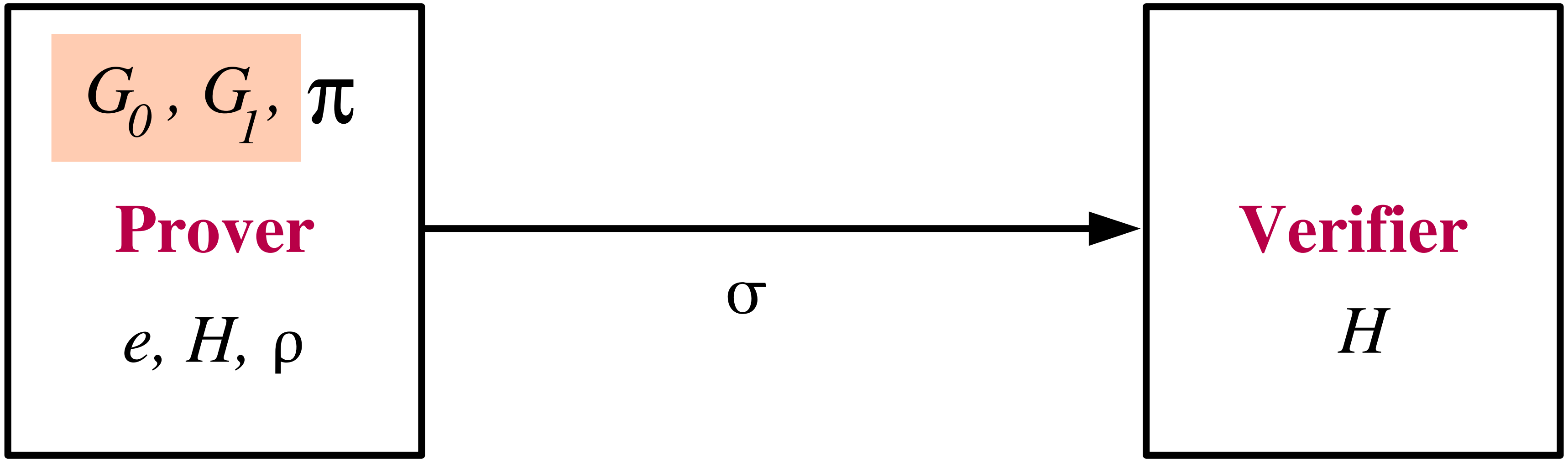
Protocol:

Prover: Generate 2nd perm ρ, compute $H=\rho(G_e)$, select $e \in \{0,1\}$

Verifier: Select $e' \in \{0,1\}$, ask prover to prove H isomorphic to $G_{e'}$

Prover: Compute $\sigma = \begin{cases} \rho & \text{if } e' = e \\ \rho \circ \pi^{-1} & \text{if } e' = 1 \text{ and } e = 0 \\ \rho \circ \pi & \text{if } e' = 0 \text{ and } e = 1 \end{cases}$

Graph Isomorphism - Zero Knowledge Proof



Verifier: Checks $\sigma(G_{e'}) = H$

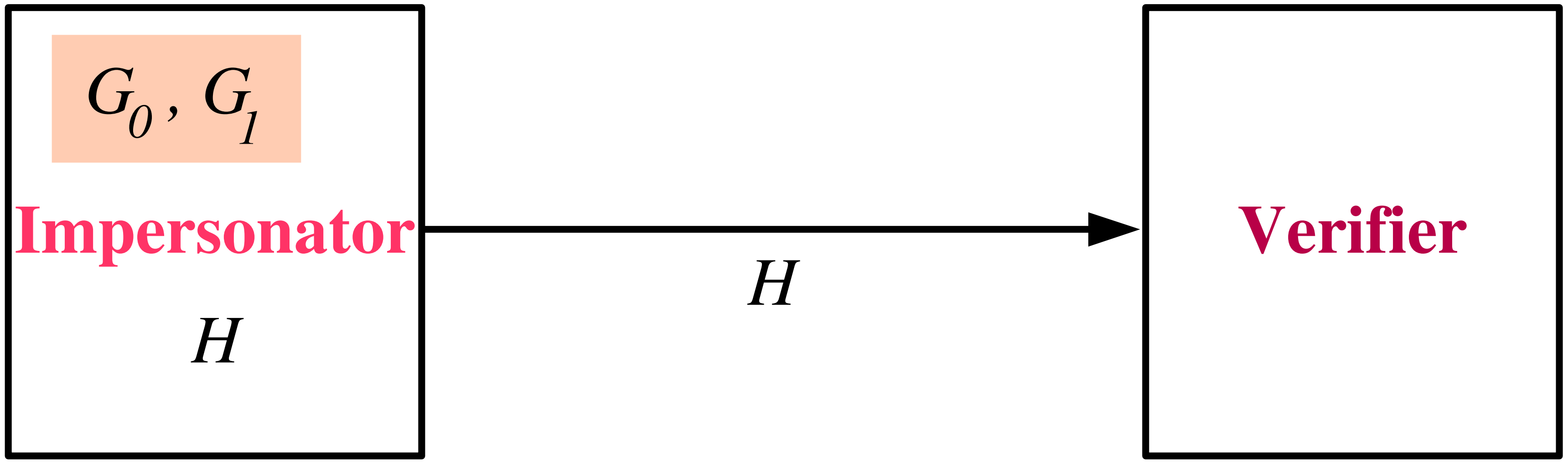
Protocol:

Prover: Generate 2nd perm ρ, compute H=$\rho(G_e)$, select $e \in \{0,1\}$

Verifier: Select $e' \in \{0,1\}$, ask prover to prove H isomorphic to $G_{e'}$

Prover: Compute $\sigma = \begin{cases} \rho & \text{if } e' = e \\ \rho \circ \pi^{-1} & \text{if } e' = 1 \text{ and } e = 0 \\ \rho \circ \pi & \text{if } e' = 0 \text{ and } e = 1 \end{cases}$

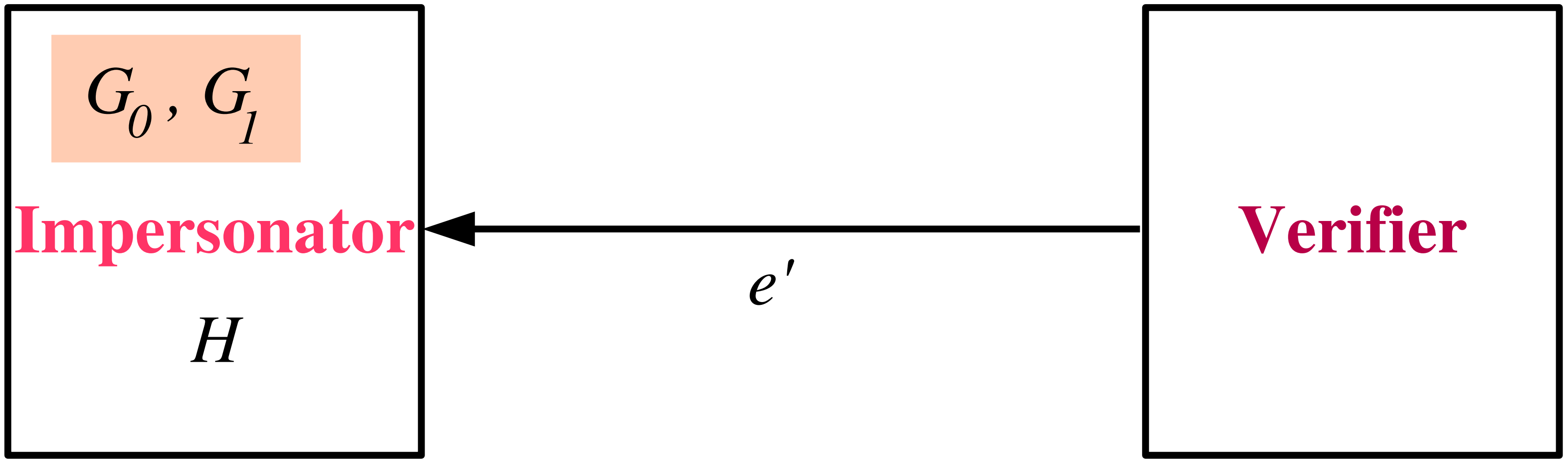
Graph Isomorphism - Zero Knowledge Proof



Protocol:

Impersonator: Generate ρ, compute H=ρ(G_e), select e ∈{0,1}

Graph Isomorphism - Zero Knowledge Proof

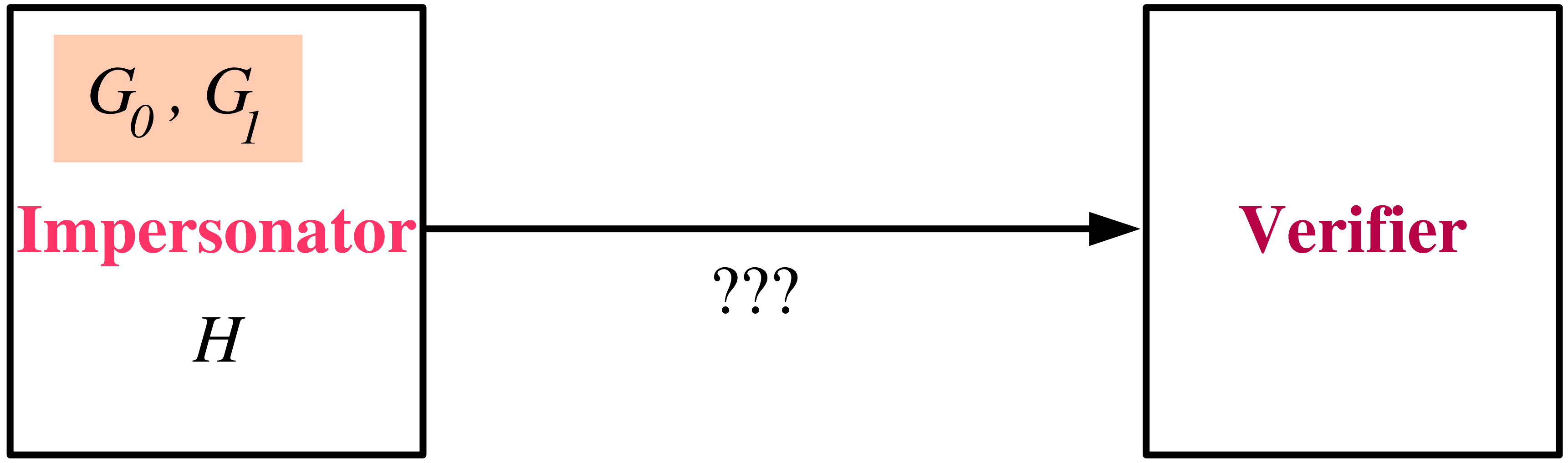


Protocol:

Impersonator: Generate ρ, compute H=ρ(G_e), select $e \in \{0,1\}$

Verifier: Select $e' \in \{0,1\}$, ask to prove H isomorphic to $G_{e'}$

Graph Isomorphism - Zero Knowledge Proof



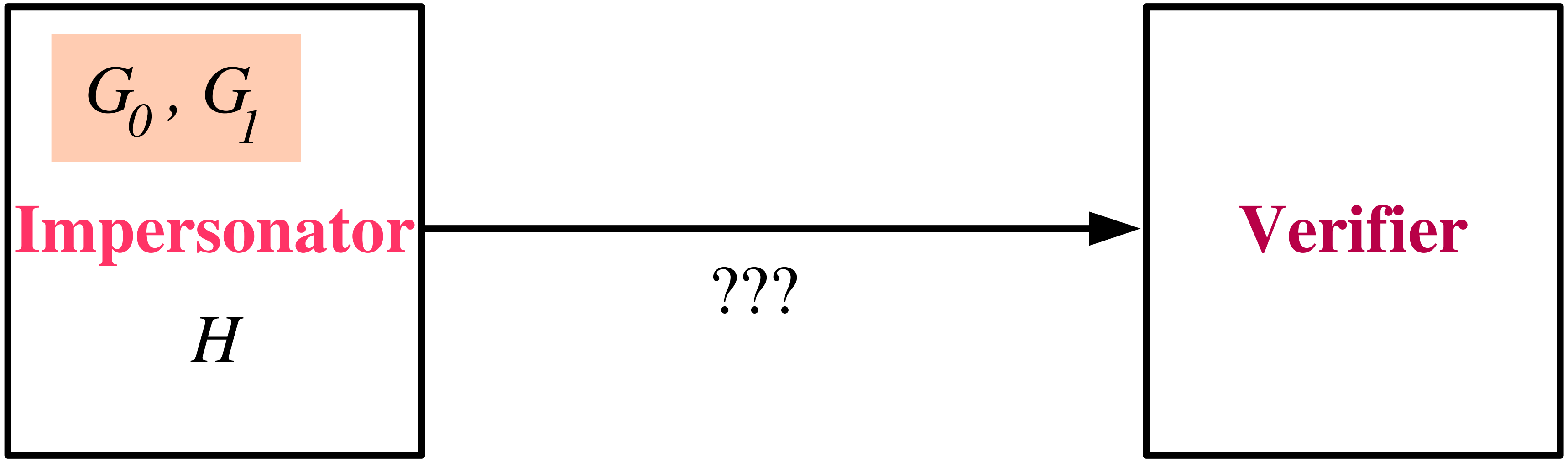
Protocol:

Impersonator: Generate ρ, compute $H=\rho(G_e)$, select $e \in \{0,1\}$

Verifier: Select $e' \in \{0,1\}$, ask to prove H isomorphic to $G_{e'}$

Impersonator: Cannot compute σ if $e' \neq e$, does not know π

Graph Isomorphism - Zero Knowledge Proof



Protocol:

Impersonator: Generate ρ, compute H=ρ(G_e), select $e \in \{0,1\}$

Verifier: Select $e' \in \{0,1\}$, ask to prove H isomorphic to $G_{e'}$

Impersonator: Cannot compute σ if $e' \neq e$, does not know π
could have seen a previous σ sent by the prover
but with probability 1/2 it would be the wrong one

Feige-Fiat-Shamir Zero Knowledge Proof

Based on difficulty of computing square roots mod a composite n

Given two large primes p, q and $n = p*q$, computing $\sqrt{x}$ mod n is very hard without knowing p, q

But there exist efficient algorithms for computing square roots modulo a prime number, and therefore $\sqrt{x}$ mod n can be computed efficiently if p and q are known

Feige-Fiat-Shamir Zero Knowledge Proof

Feige-Fiat-Shamir Zero Knowledge Proof

p, q, S

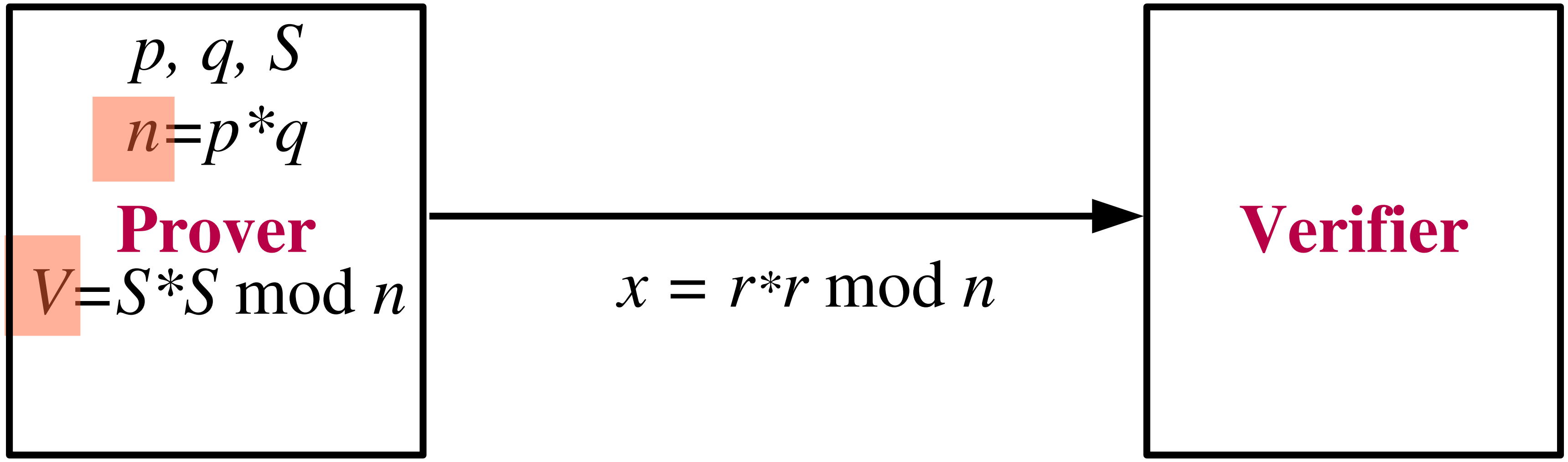
$n=p*q$

Prover

$V=S*S \bmod n$

Verifier

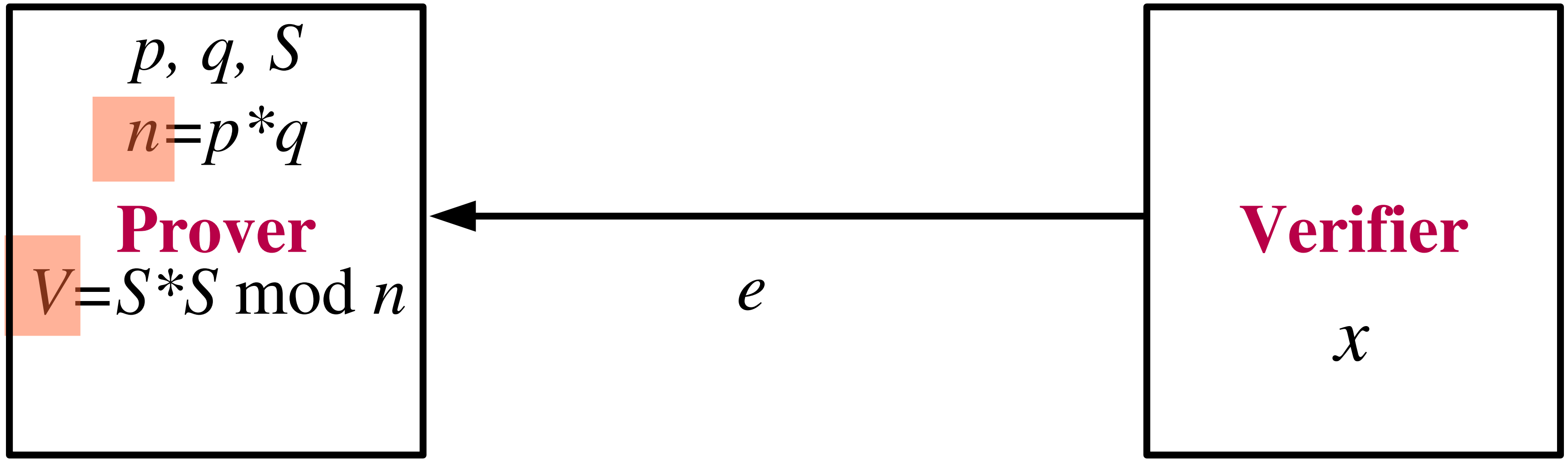
Feige-Fiat-Shamir Zero Knowledge Proof



Protocol:

Prover: Generate random r, send $x = r*r \bmod n$

Feige-Fiat-Shamir Zero Knowledge Proof

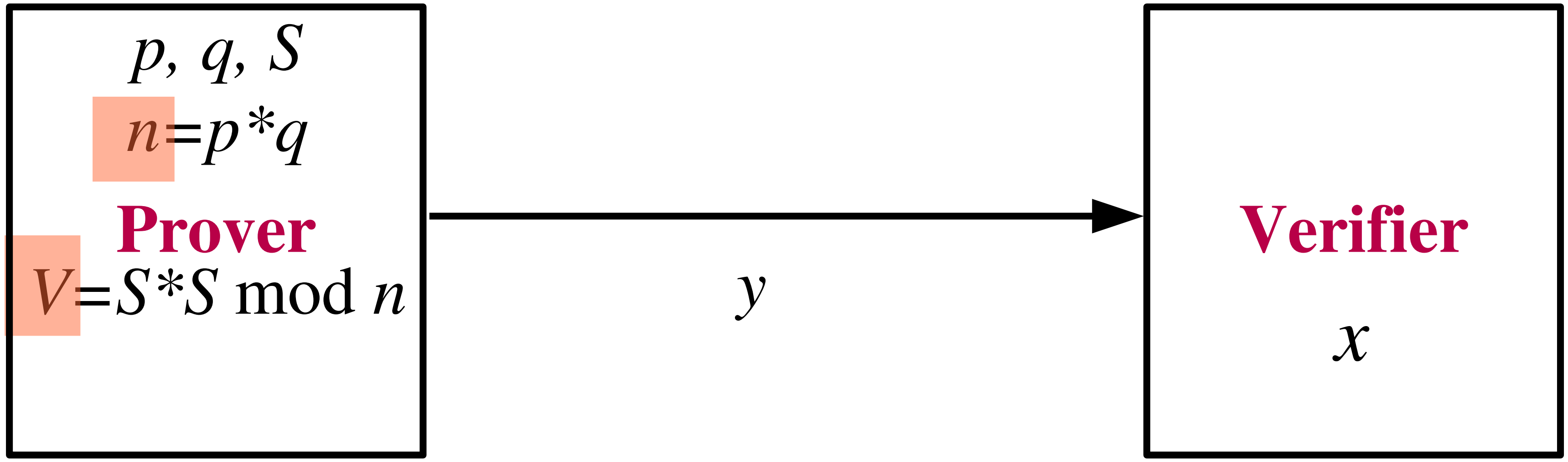


Protocol:

Prover: Generate random r, send $x = r*r \bmod n$

Verifier: Select $e \in \{0,1\}$, ask to prove it knows $\sqrt{x} \bmod n$

Feige-Fiat-Shamir Zero Knowledge Proof



Protocol:

Prover: Generate random r, send $x = r*r \bmod n$

Verifier: Select $e \in \{0,1\}$, ask to prove it knows $\sqrt{x} \bmod n$

Prover: Send $y = r*S^e \bmod n$

Feige-Fiat-Shamir Zero Knowledge Proof

p, q, S

$n = p*q$

Prover

$V = S*S \bmod n$

Verifier

x

Protocol:

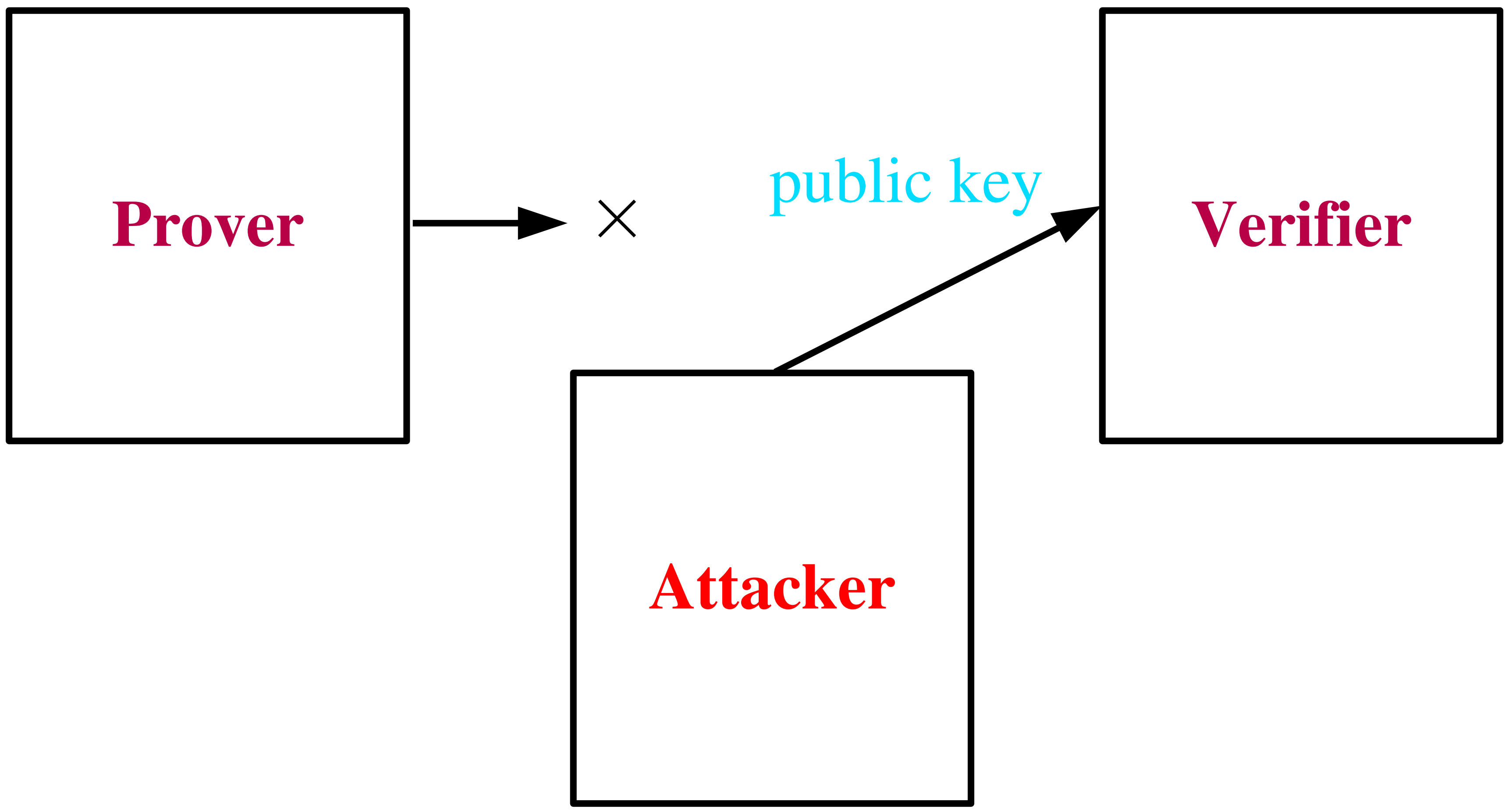
Prover: Generate random r, send $x = r*r \bmod n$

Verifier: Select $e \in \{0,1\}$, ask to prove it knows $\sqrt{x} \bmod n$

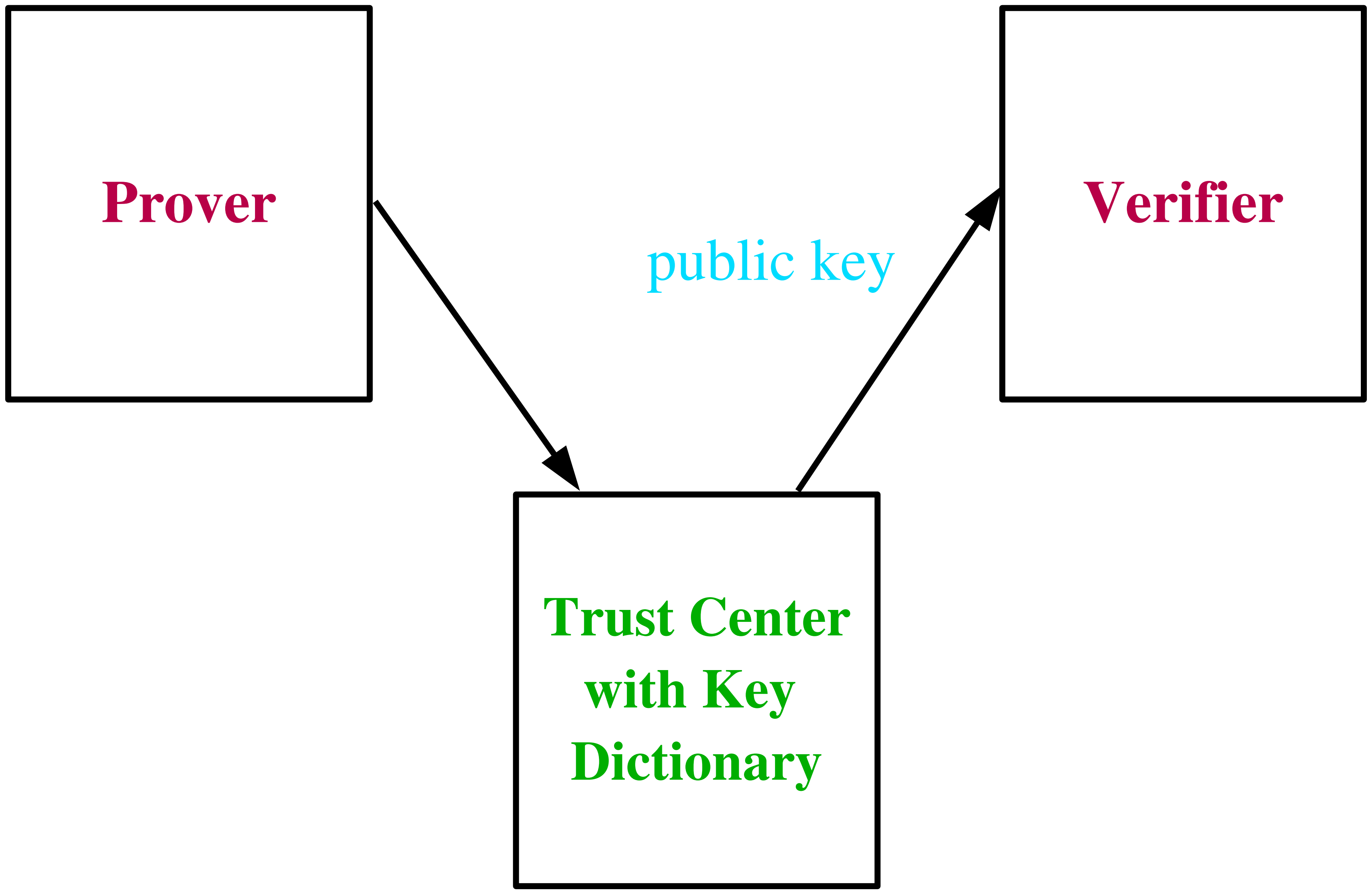
Prover: Send $y = r*S^e \bmod n$

Verifier: Checks $y*y = x*V^e \bmod n$

Security Problems



Security Problems



Security Problems

Even better: need to use trust center only for key generation

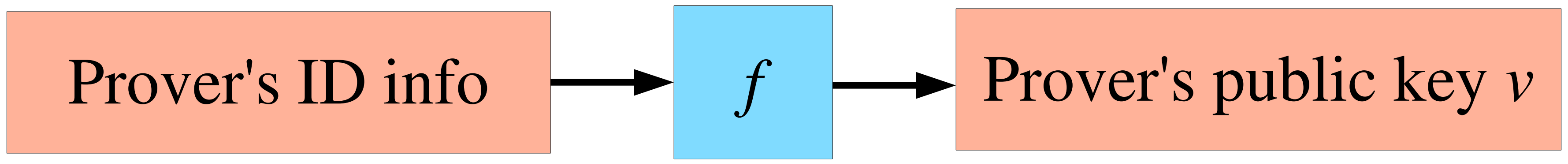
Trust Center does the following one time:

- Generates primes p, q, and computes $n=p*q$
- Publishes n, keeps p, q secret
- Defines and publishes a one-way hash function f

A Prover visits the Trust Center for a Zero-Knowledge ID

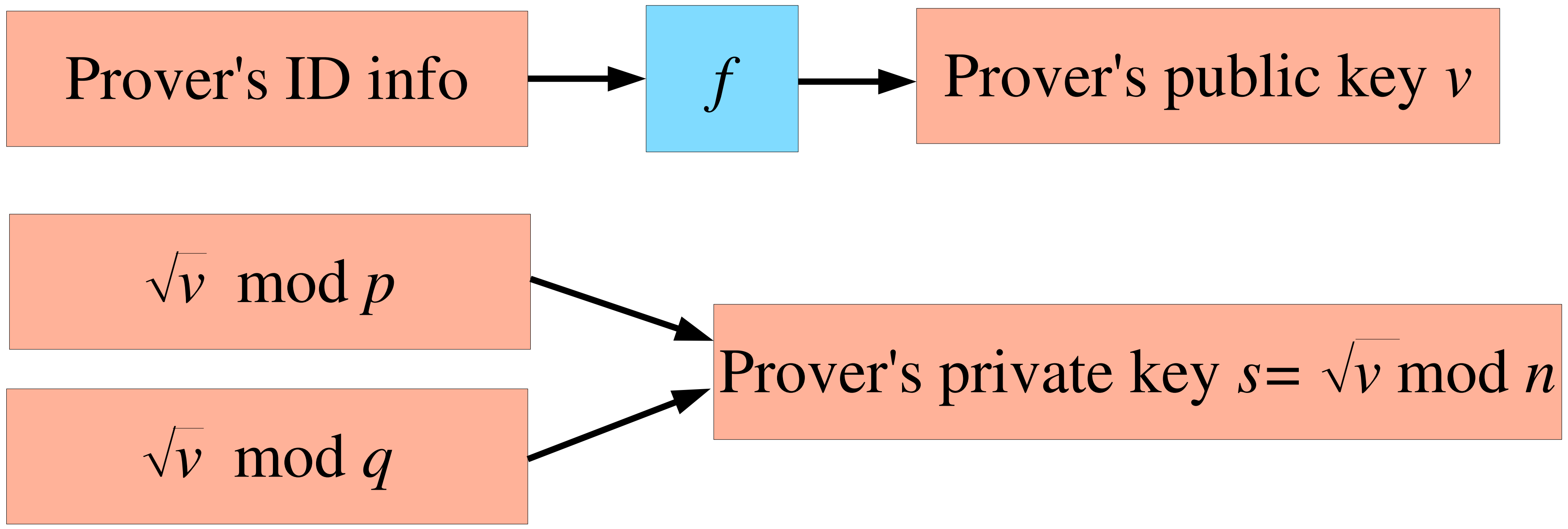
Security Problems

At the Trust Center:



Security Problems

At the Trust Center:



Security Problems

At the Trust Center:

$$\begin{tikzcd}
\text{Prover's ID info} \arrow[r] & f \arrow[r] & \text{Prover's public key } v \\
\sqrt{v}\ \bmod p \arrow[dr] & & \\
 & \text{Prover's private key } s = \sqrt{v} \bmod n & \\
\sqrt{v}\ \bmod q \arrow[ur] & & \\
\sqrt{v}\ \bmod n \arrow[dr] & & \\
 & \text{Prover's Certified Data} & \\
\text{Prover's ID info} \arrow[ur] & &
\end{tikzcd}$$

Security Problems

At the Verifier:

$$\begin{tikzcd}\boxed{\text{Prover's ID info}} \arrow[r] & \boxed{f} \arrow[r] & \boxed{\text{Prover's public key } v}\end{tikzcd}$$

Then run the Zero-Knowledge Authentication Scheme