Lecture 8: Privacy-Enhancing Technologies-2

-Zero Knowledge Proof

COMP 6712 Advanced Security and Privacy

Haiyang Xue

haiyang.xue@polyu.edu.hk

2024/3/11

Topic 2: Zero-knowledge proof

- Identification protocol and signature

- Sigma protocol

- Zero-knowledge proof
  - Zero knowledge proof for all NP
  - Non-interactive ZKP
  - zkSNARK and applications

Our aim

- We would like to know what is zero-knowledge proof
- We start from a special case, sigma protocol
- How can we construct zero-knowledge proof?
- What can we do with zero-knowledge proof?
- Recent development of zero-knowledge proof.

Typical proof

- In mathematics and in life, we often want to convince or prove things to others.
- Typically, if I know that X is true, and I want to convince you of that, I try to present all the facts I know and the inferences from that fact that imply that X is true.

- Ex: I know that 26781 is not a prime since it is 113×237,

  to prove to you that fact, I will present these factor and demonstrate that indeed $113 \times 237 = 26781$.

Why Zero-knowledge proof

- Byproduct of a proof is that you gained some knowledge,
- other than that you are now convinced that the statement is true.

- Ex: In the example before, not only are you convinced that 26781 is not a prime, but you also learned its factorization.

- A **zero knowledge proof** (Goldwasser, Micali, Rackoff 1982) tries to avoid it.
- Alice will prove to Bob that a statement X is true,
- Bob will completely convinced that X is true, but will not learn anything as a result of this process. That is, Bob will gain zero knowledge.

Mathematic problem

- Root of Quadratic equation
- $ax^2 + bx + c = 0$

- Solutions of this problem dates back to 2000 BC, Babylonian mathematicians give a preliminary solution.

- There are independent findings given by Babylonia, Egypt, Greece, China, and India.

- Now, we know $x = \frac{-b \pm \sqrt{b^2 + 2ac}}{2a}$

We assume

- Euclid would like to show to another mathematician he can find roots of all Quadratic equations,

Euclid | mathematician

Pick a, b, c

$$ax^2 + bx + c = 0$$

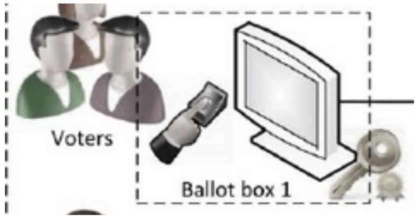
$$\xleftarrow{\quad a, b, c \quad}$$

$$x = \frac{-b \pm \sqrt{b^2 + 2ac}}{2a}$$

$$\xrightarrow{\quad x_1, x_2 \quad}$$

- BUT do not want to give any concrete solutions.(which adds "knowledge" to the mathematician)
- This is what zero-knowledge proof can solve

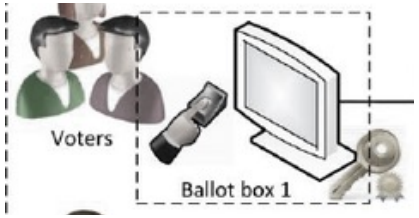
Applications: Electronic Voting (e-voting)



Candidates:
Alice,
Bob,
Tom,
Tony,
…

Alice, 0 or 1

Electronic Voting (e-voting)



Candidates:
Alice,
Bob,
Tom,
Tony,
…

ElGamal Enc for privacy
$G =< g >$
$pk := h = g^s, sk := s$

For Alice $g^{\beta_1}, h^{\beta_1} \cdot g^{b_1}$, where $b_1 = 0$ or 1

For Bob $g^{\beta_2}, h^{\beta_2} \cdot g^{b_2}$, where $b_2 = 0$ or 1

For Tony $g^{\beta_n}, h^{\beta_n} \cdot g^{b_n}$, where $b_2 = 0$ or 1

$\Pi g^{\beta_i}, \Pi(h^{\beta_i} \cdot g^{b_i})$ which is $g^{\Sigma\beta_i}, (h^{\Sigma\beta_i} \cdot g^{\Sigma b_i})$

an enc of $\sum b_i$

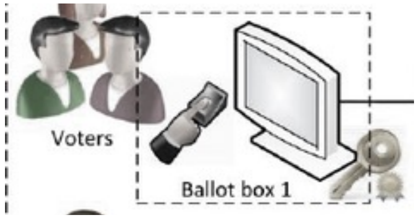
Electronic Voting (e-voting)

Candidates:
Alice,
Bob,
Tom,
Tony,
…

ElGamal Enc for privacy

$$G =< g >$$

$$pk := h = g^s, sk := s$$



For Alice $g^{\beta_1}, h^{\beta_1} \cdot g^{b_1}$

Cheating Voter $b_1 = 1000$

Thus, the voter needs to prove this is a ElGamal enc of 0 or 1
While no knowledge of b_1 is leaked

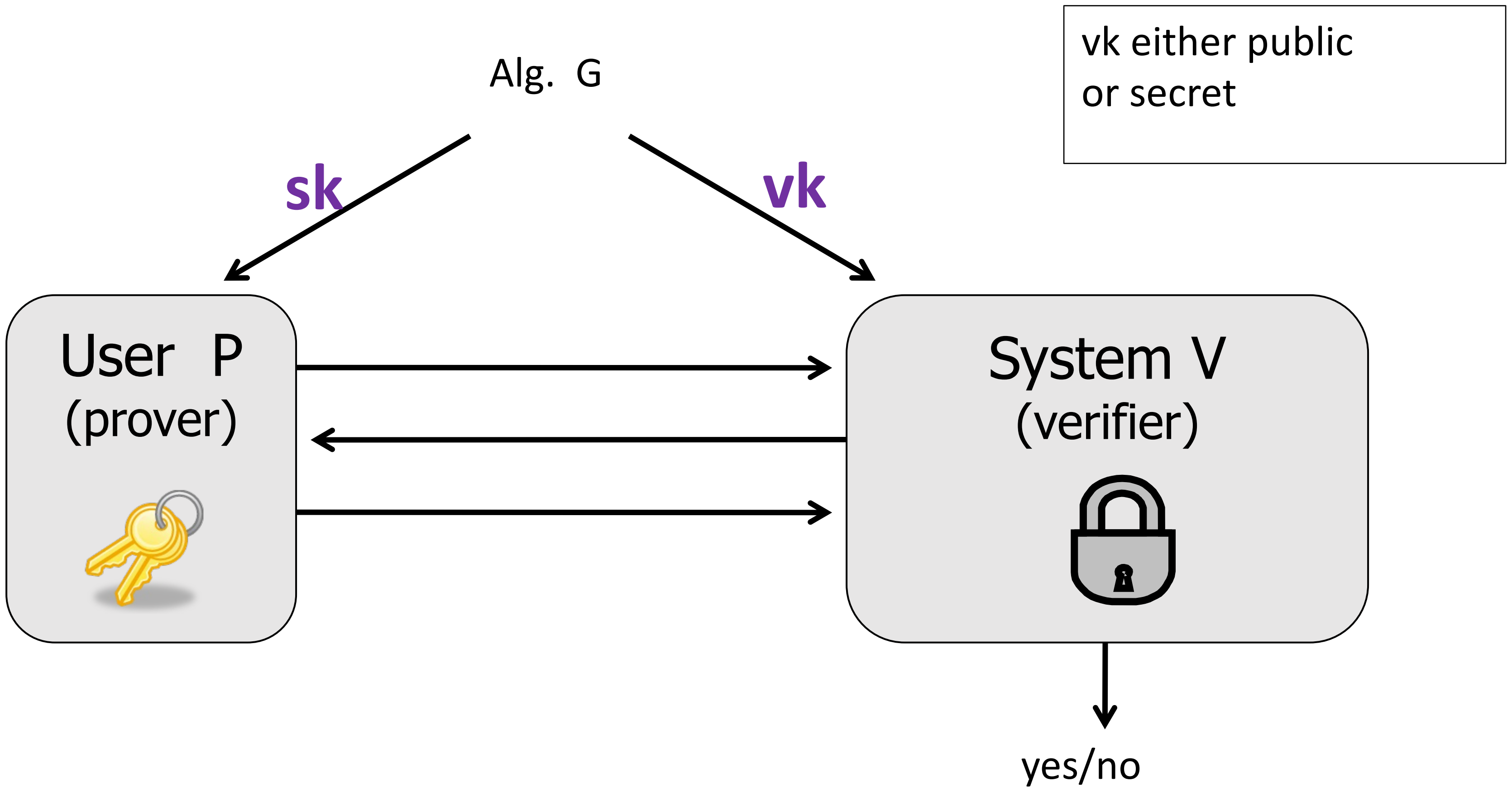
This is what Zero-knowledge proof can solve

Identification protocol

Identification protocol and signature

- ID for dl
- DDH
- Schnorr signatures

Identification/Authentication paradigm

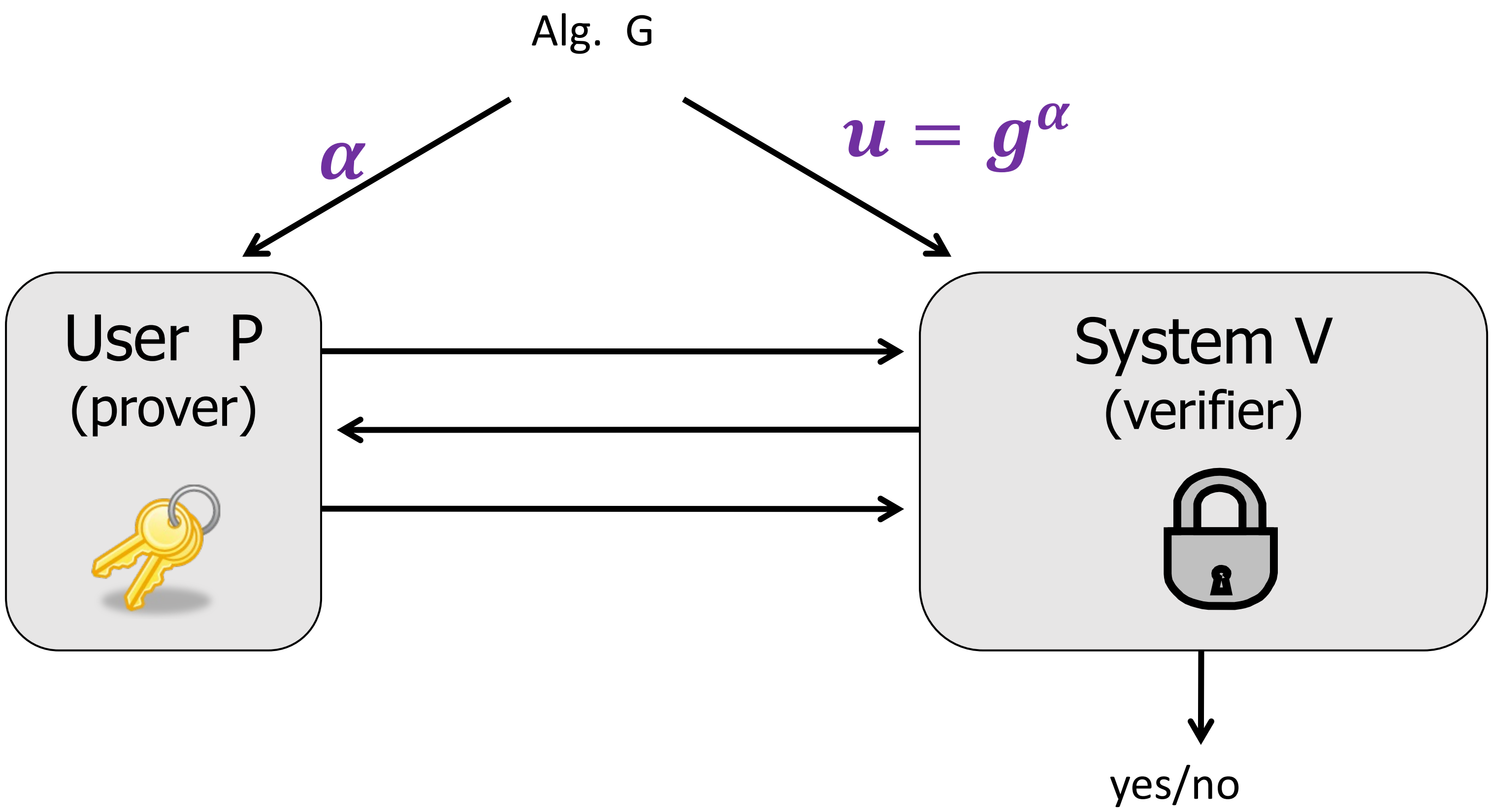


Password Auth. **sk = vk = pw**

Public key Auth. **sk, vk** is public key

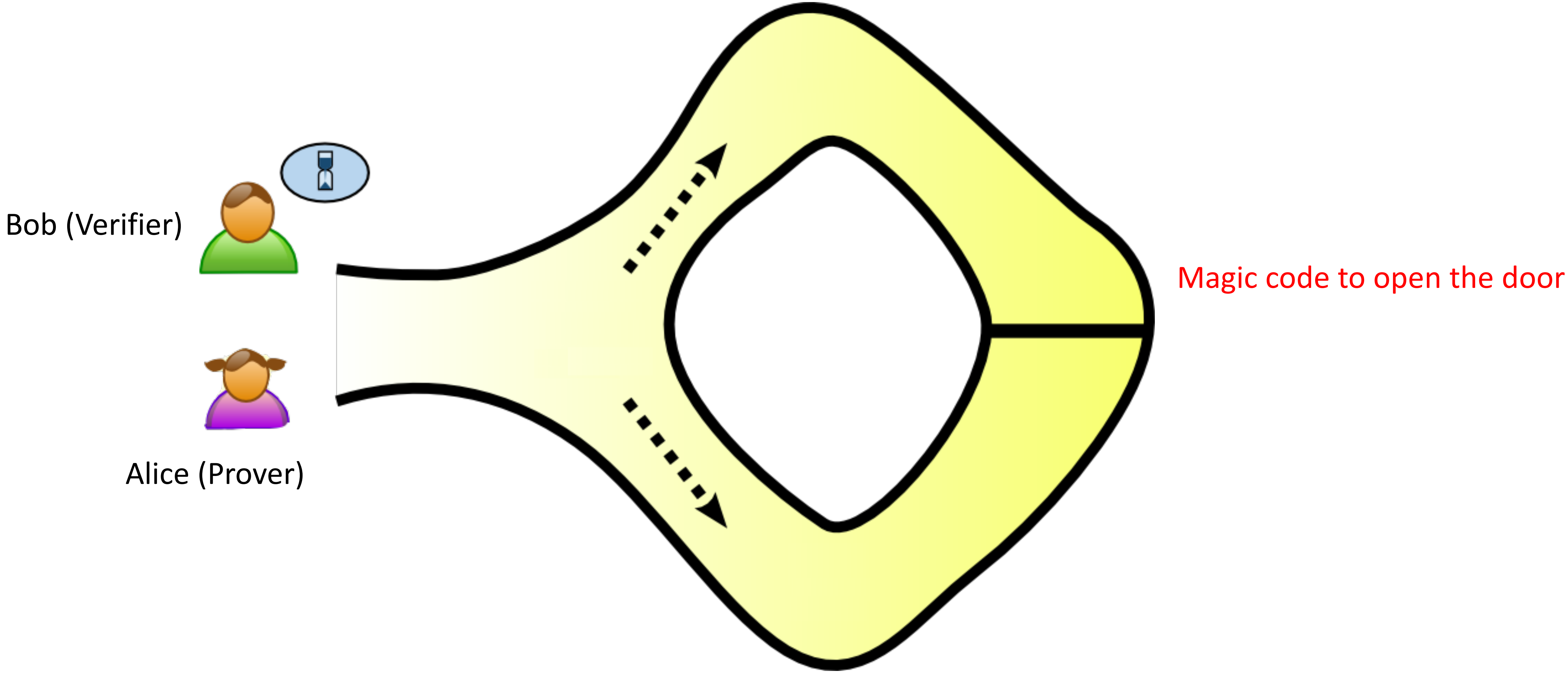
Identification/Authentication paradigm

$G =< g >, |G| = q$



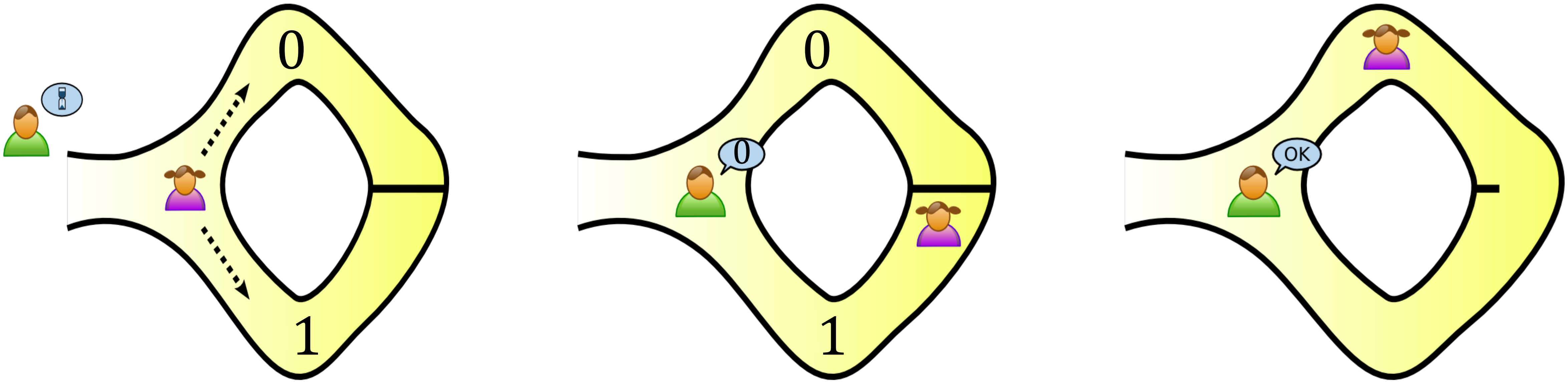
P proves the fact that "it knows α such that $u = g^{\alpha}$"
and nothing else is leaked.
How????????

A toy example: Ali Baba Cave



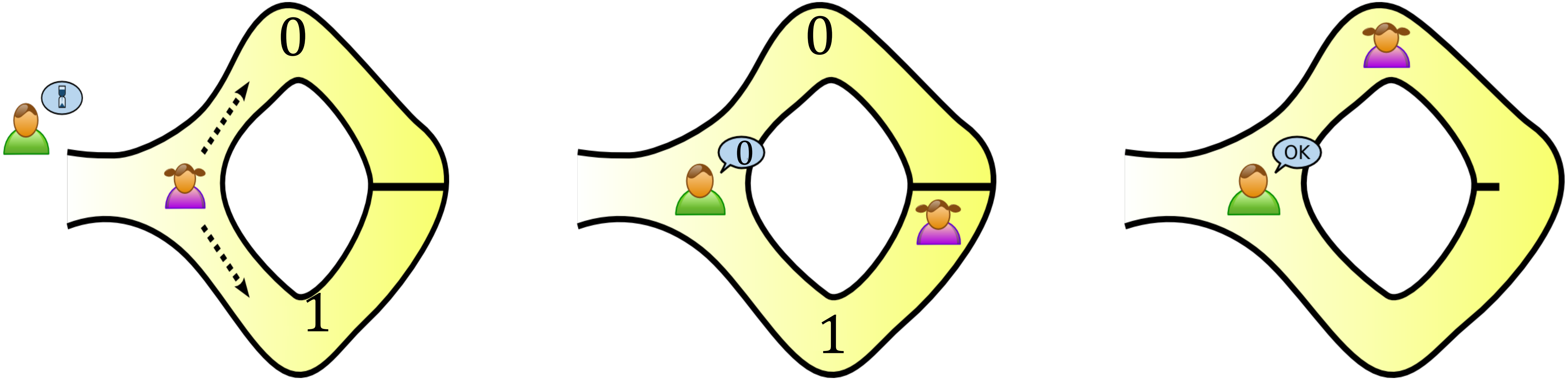
Goldwasser, Micali, Rackoff: The Knowledge Complexity of Interactive Proof-Systems (Extended Abstract)

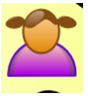
Alibaba Cave



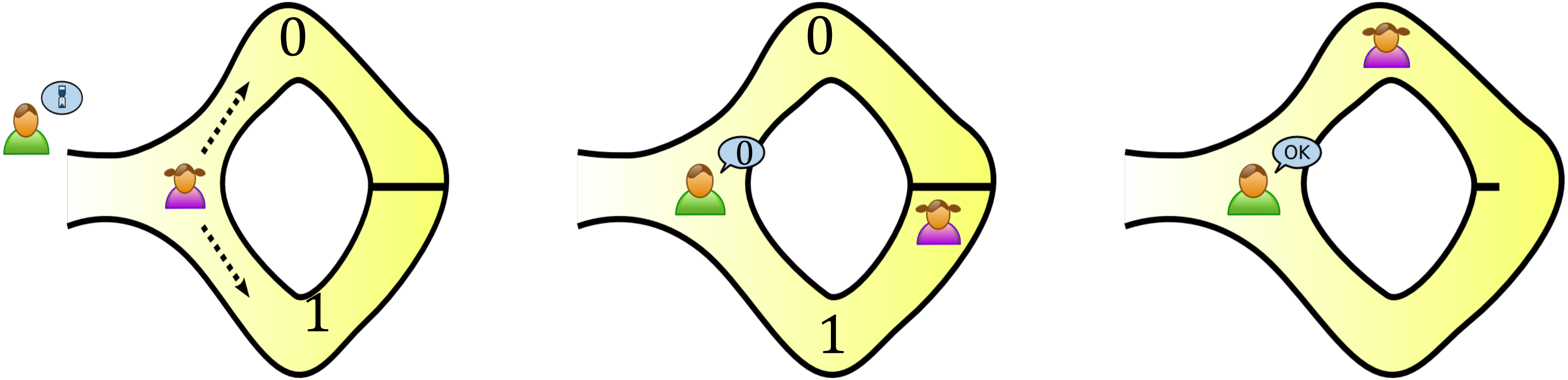
https://en.wikipedia.org/wiki/Zero-knowledge_proof

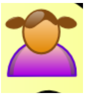
Alibaba Cave



- if  doesn't know the key, the proof was accepted with 1/2.
-  learns nothing about the magic code

Repeat the game n times



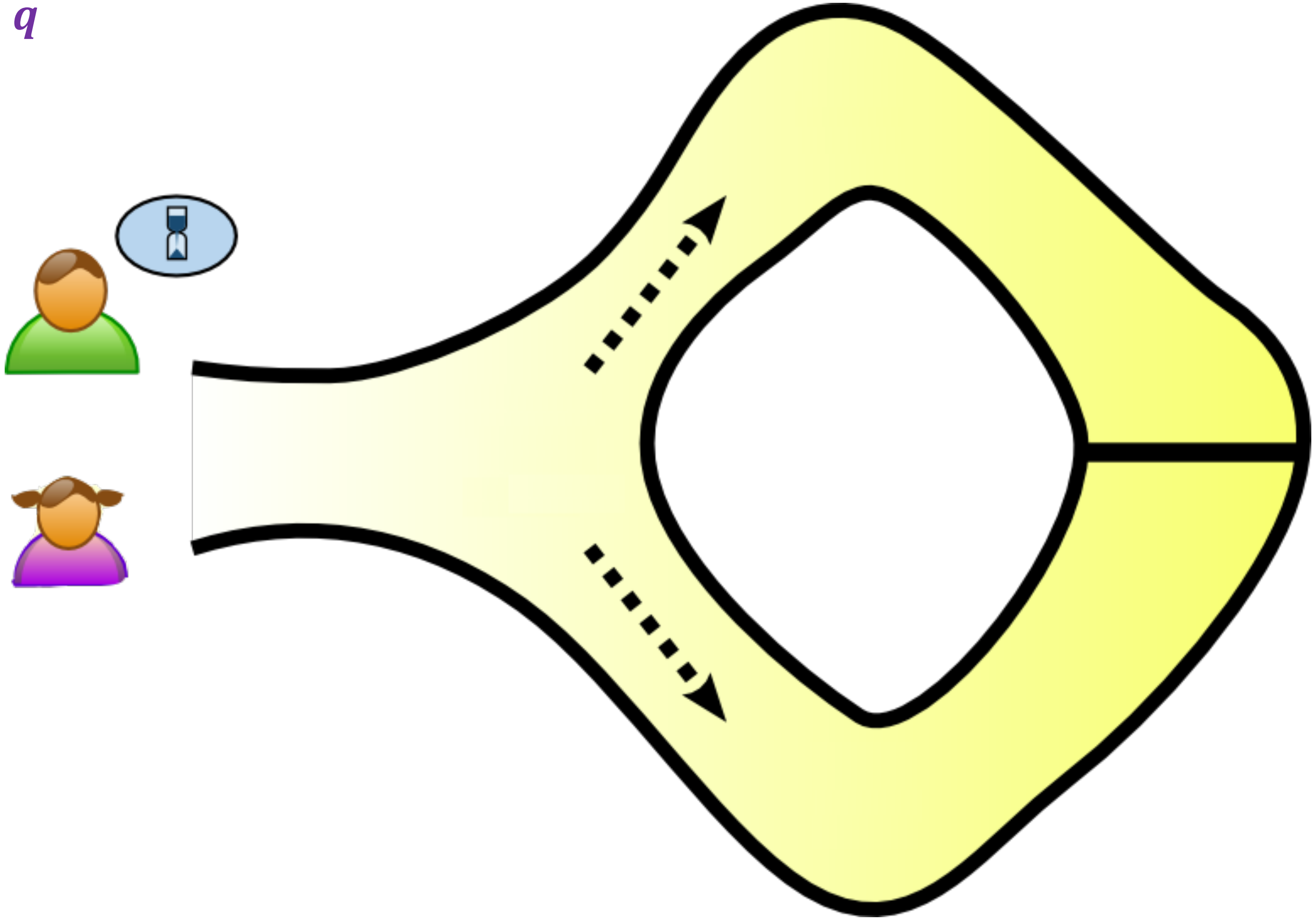
- if  does't know the key, the proof was accepted with $\frac{1}{2^n}$.
-  learns nothing about the magic code

Identification for Discrete logarithm

$G =< g >, |G| = q$

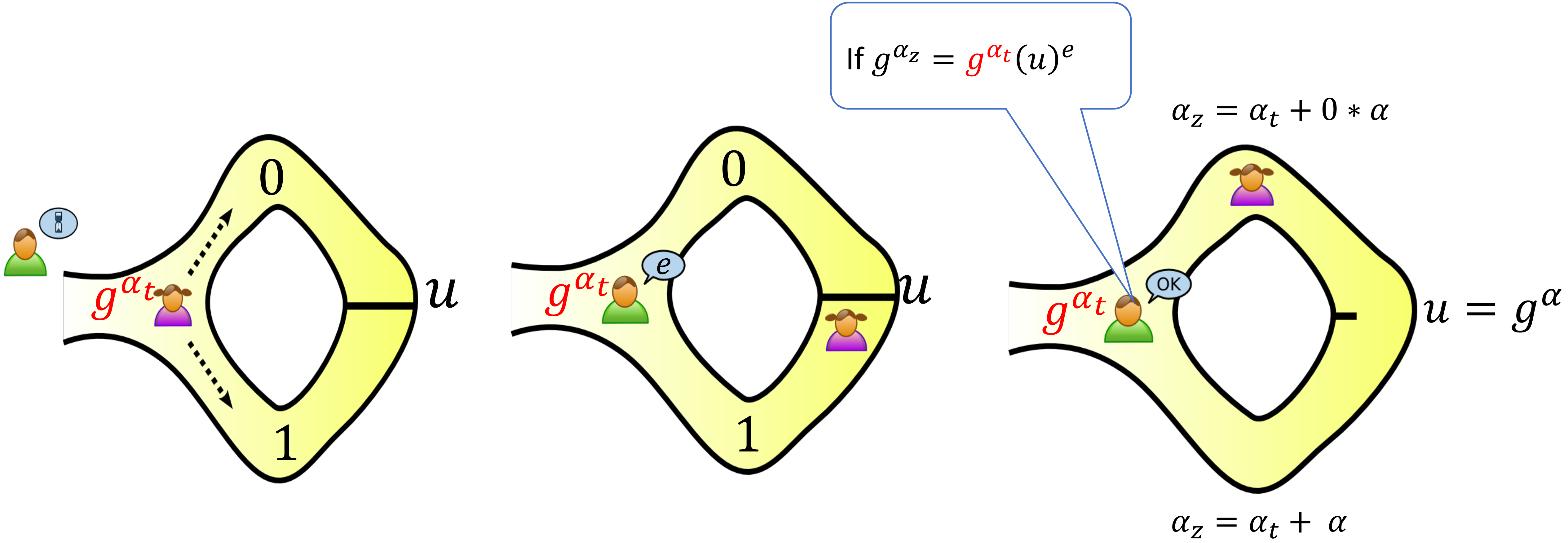
$g^a g^b = g^{a+b}$

$(g^a)^b = g^{ab}$



$u = g^{\alpha}$

Schnorr Identification

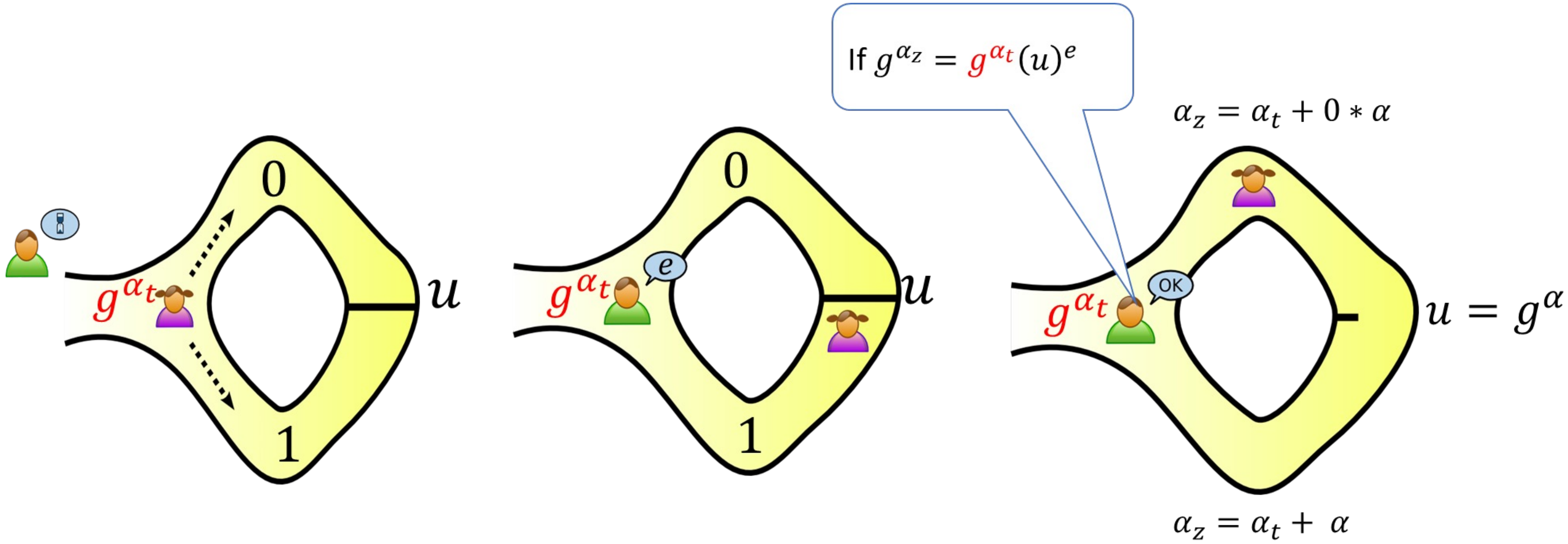


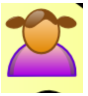
Alice commits to g^{α_t}

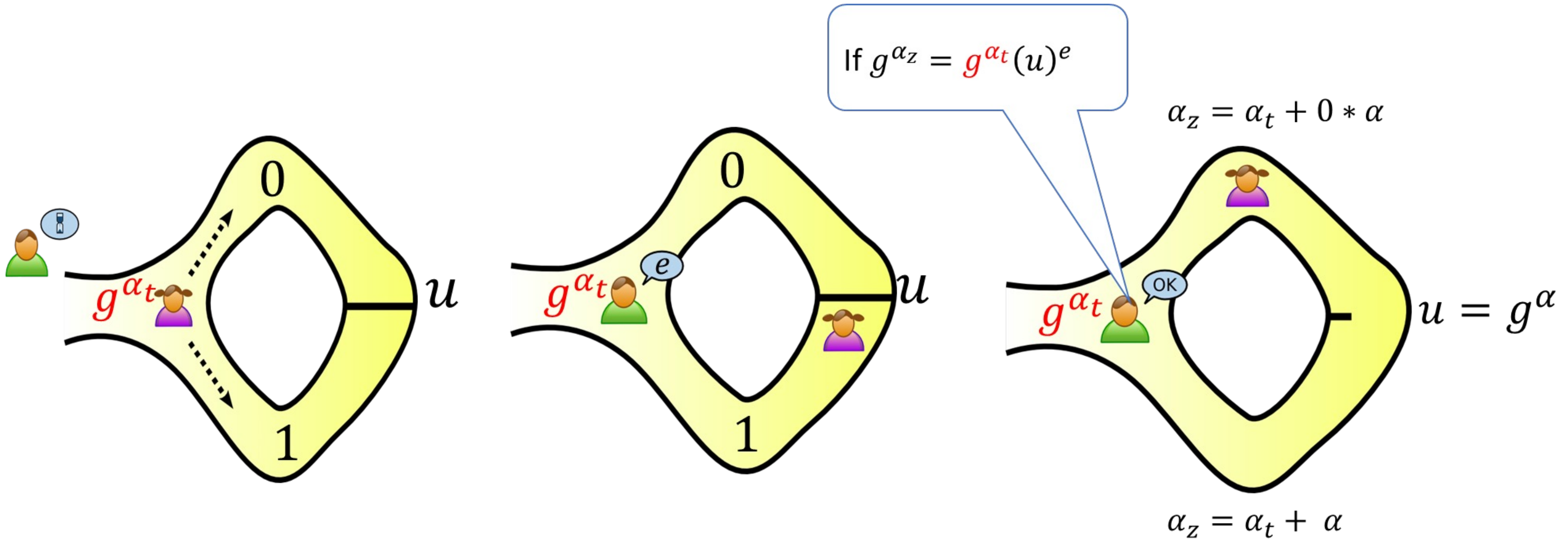
Bob chooses a challenge e

Alice responds with α_z

$Correctness\ g^z = g^{\alpha_t} g^{e\alpha} = g^{\alpha_t + e\alpha}$



- if  doesn't know the key, the proof was accepted with 1/2.
-  learns nothing about the magic code (α is covered by α_t)



- if  doesn't know the key, the proof was accepted with 1/2.
- Repeat the game n times, if … doesn't know the key, accepted with $1/2^n$.
- How about choose $e \leftarrow Z_q$, (q entrances rather than 2)?

Schnorr Identification

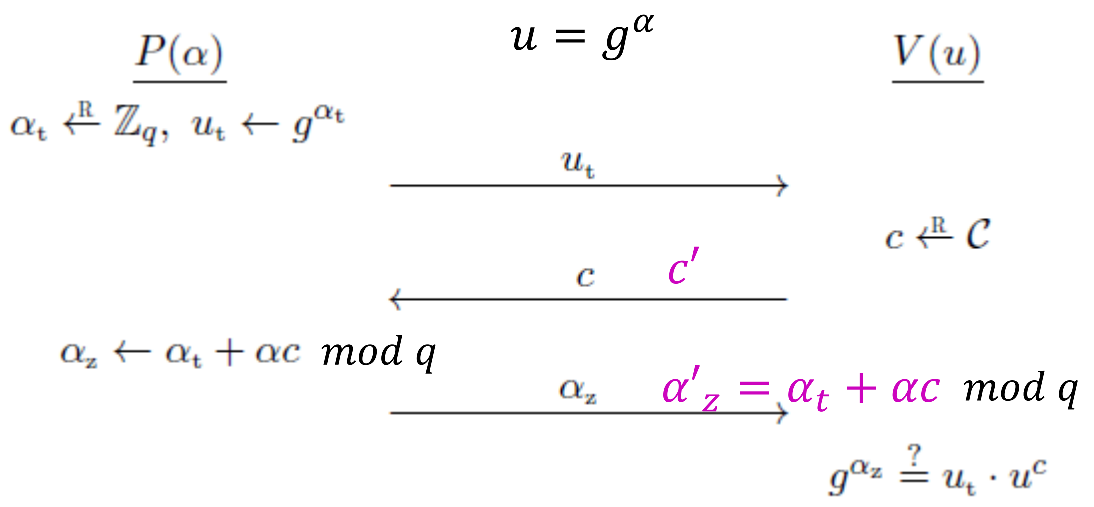
$$u = g^{\alpha}$$

$\underline{P(\alpha)}$		$\underline{V(u)}$
$\alpha_t \xleftarrow{R} \mathbb{Z}_q,\ u_t \leftarrow g^{\alpha_t}$		
	$\xrightarrow{\quad u_t \quad}$	
		$c \xleftarrow{R} \mathcal{C}$
	$\xleftarrow{\quad c \quad}$	
$\alpha_z \leftarrow \alpha_t + \alpha c \ \ mod\ q$		
	$\xrightarrow{\quad \alpha_z \quad}$	
		$g^{\alpha_z} \stackrel{?}{=} u_t \cdot u^c$

- Challenge space $\mathcal{C} = Z_q$
- Conversation: (u_t, c, α_z) is said to be valid if the verification passes

Direct Attacker

- An attacker without knowing α would like to pass the verification.



If the attacker can return valid respond α_z for a random c with probability ϵ

it can return valid respond α'_z for a random c' with probability $\epsilon - 1/q$ [Theorem 19.1, DS]

With c, c' and $\begin{cases} \alpha_z = \alpha_t + \alpha c \ mod\ q \\ \alpha'_z = \alpha_t + \alpha c' mod\ q \end{cases}$

we can find (or extract) α with probability $\epsilon(\epsilon - 1/q)$ (which is the discrete logarithm problem)

[DS] Dan Boneh and Victor Shoup, A Graduate Course in Applied Cryptography

What we have shown: “proof of knowledge”

- If someone passes the verification of Schnorr Identification,
- We must have the someone knows the discrete logarithm of $u = g^{\alpha}$

Eavesdropper Attacker

Actually, the attacker may see several valid conversations $(u_t^i, c^i, \alpha_z^i)_{i=1,2,3\ldots}$ does "proof of knowledge" hold?

$\underline{P(\alpha)}$ $\qquad u = g^{\alpha}$ $\qquad \underline{V(u)}$

$\alpha_t \xleftarrow{R} \mathbb{Z}_q, \; u_t \leftarrow g^{\alpha_t}$

$\xrightarrow{\quad u_t \quad}$

$c \xleftarrow{R} \mathcal{C}$

$\xleftarrow{\quad c \quad c' \quad}$

$\alpha_z \leftarrow \alpha_t + \alpha c \;\; mod \; q$

$\xrightarrow{\quad \alpha_z \quad}$ $\alpha'_z = \alpha_t + \alpha c \;\; mod \; q$

$g^{\alpha_z} \stackrel{?}{=} u_t \cdot u^c$

If the attacker can return valid respond α_z for a random c with probability ϵ

it can return valid respond α'_z for a random c' with probability $\epsilon - 1/q$ [Theorem 19.1, DS]

We can generate what Eav attacker learns $(u_t^i, c^i, \alpha_z^i)_{i=1,2,3\ldots}$

Sample $\alpha_z^i \leftarrow Z_q, c^i \leftarrow Z_q$ compute $u_t^i = g^{\alpha_z^i}/u^{c^i}$

With c, c' and $\begin{cases} \alpha_z = \alpha_t + \alpha c \; mod \; q \\ \alpha'_z = \alpha_t + \alpha c' mod \; q \end{cases}$

we can extract α with probability $\epsilon(\epsilon - 1/q)$ (which is the discrete logarithm problem)

What we have shown: honest verifier zero-knowledge

$$u = g^{\alpha}$$

$\underline{P(\alpha)}$		$\underline{V(u)}$
$\alpha_t \xleftarrow{R} \mathbb{Z}_q,\ u_t \leftarrow g^{\alpha_t}$		
	$\xrightarrow{\quad u_t \quad}$	
		$c \xleftarrow{R} \mathcal{C}$
	$\xleftarrow{\quad c \quad}$	
$\alpha_z \leftarrow \alpha_t + \alpha c \mod q$		
	$\xrightarrow{\quad \alpha_z \quad}$	
		$g^{\alpha_z} \stackrel{?}{=} u_t \cdot u^c$

We can generate what Eav attacker learns $\left(u_t^i, c^i, \alpha_z^i\right)_{i=1,2,3\ldots}$

Sample $\alpha_z^i \leftarrow Z_q, c^i \leftarrow Z_q$ compute $u_t^i = g^{\alpha_z^i}/u^{c^i}$

Honest verifier zero-knowledge says that:

without knowing the witness (discrete logarithm), we can generate (simulate) the valid transaction efficiently

Schnorr Identification

$$u = g^{\alpha}$$

$$\begin{array}{ccc}
\underline{P(\alpha)} & & \underline{V(u)} \\
\alpha_t \xleftarrow{R} \mathbb{Z}_q,\ u_t \leftarrow g^{\alpha_t} & & \\
& \xrightarrow{\quad u_t \quad} & \\
& & c \xleftarrow{R} \mathcal{C} \\
& \xleftarrow{\quad c \quad} & \\
\alpha_z \leftarrow \alpha_t + \alpha c \ \ mod\ q & & \\
& \xrightarrow{\quad \alpha_z \quad} & \\
& & g^{\alpha_z} \stackrel{?}{=} u_t \cdot u^c
\end{array}$$

- **Correctness(Completeness):** If P and V execute the protocol honestly, the proof is accepted.
- **Soundness (proof-of-knowledge):** If the proof is accepted, we can extract the witness (discrete log) α
- **Honest verifier zero-knowledge** says that: without knowing the witness (discrete logarithm), we can generate (simulate) the valid transaction efficiently

Identification protocol --- > Signature

$$\underline{P(\alpha)} \qquad u = g^{\alpha} \qquad \underline{V(u)}$$

$$\alpha_t \xleftarrow{R} \mathbb{Z}_q,\ u_t \leftarrow g^{\alpha_t}$$

$$\xrightarrow{\quad u_t \quad}$$

$$\boxed{c = Hash(m, u_t, u)}$$

$$\xleftarrow{\quad c \quad}$$

$$\alpha_z \leftarrow \alpha_t + \alpha c \ \ mod\ q$$

$$\xrightarrow{\quad \alpha_z \quad}$$

$$g^{\alpha_z} \stackrel{?}{=} u_t \cdot u^c$$

- The key generation
  - $\alpha \leftarrow Z_q, u = g^{\alpha}$
  - $sk = \alpha, vk = u$
- To sign m
  - $\alpha_t \leftarrow Z_q, u_t = g^{\alpha_t}$
  - $c = Hash(m, u_t, u)$
  - $\alpha_z = \alpha_t + \alpha c \ mod\ q$
  - Return $\sigma = (u_t, c, \alpha_t)$
- Verification
  - $g^{\alpha_z} =?\, u_t \cdot u^c$

Schnorr Signature is UF-CMA secure, under the discrete logarithm assumption

Identification protocol --- > Signature

$u = g^{\alpha}$

$\underline{P(\alpha)}$ $\qquad$ $\underline{V(u)}$

$\alpha_t \xleftarrow{R} \mathbb{Z}_q,\ u_t \leftarrow g^{\alpha_t}$

$\xrightarrow{\quad u_t \quad}$

$c = Hash(m, u_t, u)$

$\xleftarrow{\quad c \quad}$

$\alpha_z \leftarrow \alpha_t + \alpha c \quad mod\ q$

$\xrightarrow{\quad \alpha_z \quad}$

$g^{\alpha_z} \stackrel{?}{=} u_t \cdot u^c$

Soundness (discrete log) ⟶ Unforgeability

$Hash$ is random oracle

Honest verifier zero-knowledge ⟶ Chosen Message Attack

- The key generation
  - $\alpha \leftarrow Z_q, u = g^{\alpha}$
  - $sk = \alpha, vk = u$
- To sign m
  - $\alpha_t \leftarrow Z_q, u_t = g^{\alpha_t}$
  - $c = Hash(m, u_t, u)$
  - $\alpha_z = \alpha_t + \alpha c\ mod\ q$
  - Return $\sigma = (u_t, c, \alpha_t)$
- Verification
  - $g^{\alpha_z} =?\ u_t \cdot u^c$

History of Schnorr signature

- Schnorr invented Schnorr signature in 1989
- It was covered by U.S. Patent which expired in February 2008.
- In 1991, the National Institute of Standards (NIST) considered a number of viable candidates. Because the Schnorr system was protected by a patent, NIST opted for a more ad-hoc signature scheme: (EC)DSA
- Security: Schnorr > ECDSA
- Deployment: Schnorr < ECDSA

Schnorr, C. P. (1989). "Efficient Identification and Signatures for Smart Cards"

Identification for Decisional Diffie-Hellman ID_{DDH}

$$v = g^{\beta}, w = u^{\beta}$$

$$\begin{array}{lcl}
\underline{P(\beta, (u, v, w))} & & \underline{V(u, v, w)} \\
\beta_t \xleftarrow{R} \mathbb{Z}_q,\ v_t \leftarrow g^{\beta_t},\ w_t \leftarrow u^{\beta_t} & & \\
 & \xrightarrow{\quad v_t, w_t \quad} & \\
 & & c \xleftarrow{R} \mathcal{C} \\
 & \xleftarrow{\quad c \quad} & \\
\beta_z \leftarrow \beta_t + \beta c \ mod\ q & & \\
 & \xrightarrow{\quad \beta_z \quad} & \\
 & & g^{\beta_z} \stackrel{?}{=} v_t \cdot v^c \text{ and } u^{\beta_z} \stackrel{?}{=} w_t \cdot w^c
\end{array}$$

Given $(g, u, v = g^{\beta}, w = u^{\beta})$ with witness β, P wants to prove that it knows β

Identification for Decisional Diffie-Hellman (DDH)

Given $(g, u, v = g^{\beta}, w = u^{\beta})$ with witness β, P wants to prove that it knows β

$$v = g^{\beta}, w = u^{\beta}$$

$\underline{P(\beta, (u, v, w))}$		$\underline{V(u, v, w)}$
$\beta_t \xleftarrow{R} \mathbb{Z}_q,\ v_t \leftarrow g^{\beta_t},\ w_t \leftarrow u^{\beta_t}$		
	$\xrightarrow{\quad v_t,\, w_t \quad}$	
		$c \xleftarrow{R} \mathcal{C}$
	$\xleftarrow{\quad c \quad}$	
$\beta_z \leftarrow \beta_t + \beta c \ \ mod\ q$		
	$\xrightarrow{\quad \beta_z \quad}$	
		$g^{\beta_z} \stackrel{?}{=} v_t \cdot v^c$ and $u^{\beta_z} \stackrel{?}{=} w_t \cdot w^c$

- **Correctness(Completeness):** If P and V exact the protocol honestly, the proof is accepted.
- **Soundness (proof-of-knowledge):** If the proof is accepted, we can extract the witness (discrete log) α
- **Honest verifier zero-knowledge** says that: without knowing the witness (discrete logarithm), we can generate (simulate) the valid transaction efficiently

$$\boxed{\beta_z \leftarrow Z_q, c \leftarrow Z_q, v_t = \frac{g^{\beta_z}}{v^c}, u_t = g^{\beta_z}/u^c}$$

A short summary

- Identification protocol could be used to prove knowing something (discrete log)
- Without the fact of knowing something, nothing else is leaked
- Identification protocol could be used to build signature
- Identification protocols from discrete log and DDH

SIGMA protocol

SIGMA protocol

- Identification protocol is a special case of SIGMA protocol

- We first recall the language and corresponding relation

A NP language $L \coloneqq \{y \mid \exists\, x, s.t. (x, y) \in R\}$

Corresponding Relation R

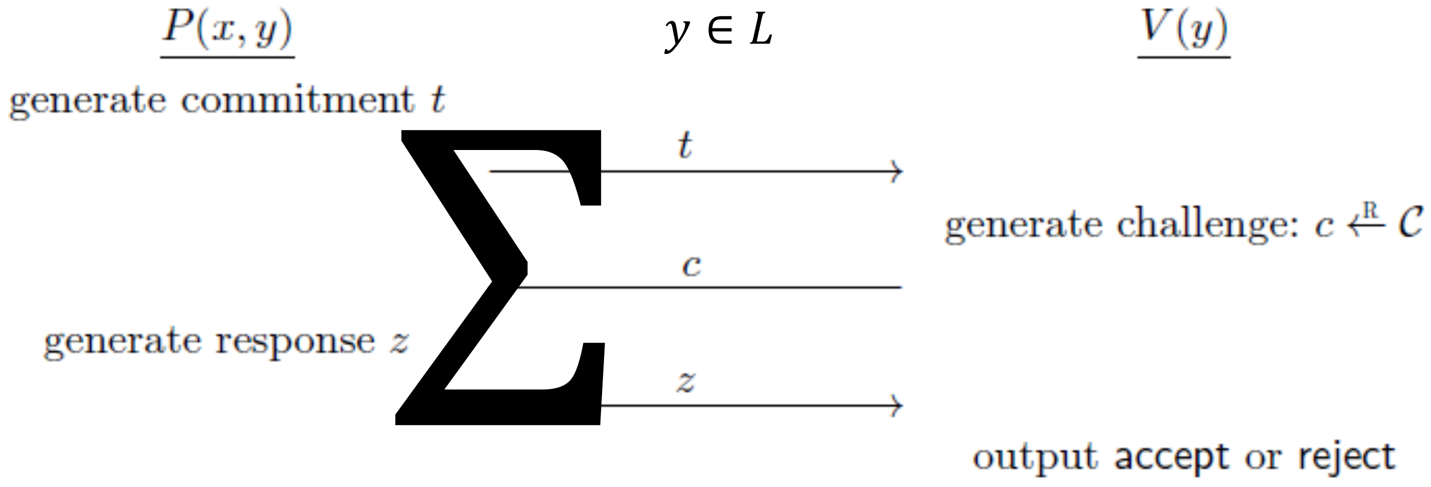
$y \in L$ if and only if $\exists$ withness x, such that $(x, y) \in R$

$(g, u, v, w) \in L_{DDH}$ iff $\exists$ witness β such that $v = g^{\beta}, w = u^{\beta}$

x is called the witness and y is called the statement

SIGMA protocol

- To proof that P knows witness x of statement y such that $(x, y) \in R$
- Sigma protocol runs as follows and



- **Correctness(Completeness):** If P and V execute the protocol honestly, the proof is accepted.
- **Special Soundness**: given valid transection (t, c, z) and (t, c', z'), we could extract x
- **Honest verifier zero-knowledge** says that: without knowing witness x, we can generate (simulate) the valid transaction efficiently for $y \in L$

Identification protocol is a special case of SIGMA

Schnorr, Discrete log relation $\mathcal{R} = \{\ (\alpha, u) \in \mathbb{Z}_q \times \mathbb{G} :\ \ g^{\alpha} = u\ \}$

DDH relation $\mathcal{R} := \left\{\ (\ \beta,\ (u, v, w)\ ) \in \mathbb{Z}_q \times \mathbb{G}^3 :\ v = g^{\beta} \text{ and } w = u^{\beta}\ \right\}$

Other relations

Given $G =< g >$ of order q, $h \in G$, and $u = g^{\alpha} h^{\beta} \in G$
with witness α, β, prove the following relation

$$\mathcal{R} = \left\{ \ \big(\ (\alpha, \beta), \ u \ \big) \in \mathbb{Z}_q^2 \times \mathbb{G} : \ g^{\alpha} h^{\beta} = u \ \right\}$$

Okamoto's protocol

$$\mathcal{R} = \left\{ \; (\; (\alpha,\beta),\; u \;) \in \mathbb{Z}_q^2 \times \mathbb{G} : \; g^{\alpha} h^{\beta} = u \; \right\}$$

$$\begin{array}{ccc}
\underline{P((\alpha,\beta),u)} & & \underline{V(u)} \\
\alpha_t \xleftarrow{R} \mathbb{Z}_q,\; \beta_t \xleftarrow{R} \mathbb{Z}_q,\; u_t \leftarrow g^{\alpha_t} h^{\beta_t} & & \\
& \xrightarrow{\quad u_t \quad} & \\
& & c \xleftarrow{R} \mathcal{C} \\
& \xleftarrow{\quad c \quad} & \\
\alpha_z \leftarrow \alpha_t + \alpha c \; mod \; q & & \\
\beta_z \leftarrow \beta_t + \beta c \; mod \; q & & \\
& \xrightarrow{\quad \alpha_z,\; \beta_z \quad} & \\
& & g^{\alpha_z} h^{\beta_z} \stackrel{?}{=} u_t \cdot u^c
\end{array}$$

Extension of Schnorr

- **Correctness(Completeness):** If P and V execute the protocol honestly, the proof is accepted.
- **Special Soundness**: given valid transection $(u_t, c, \alpha_z, \beta_z)$ and $(u_t, c', \alpha'_z, \beta'_z)$, we could extract α, β
- **Honest verifier zero-knowledge** says that: without knowing witness x, we can generate (simulate) the valid transaction efficiently for $y \in L$

AND composition of SIGAMA

Schnorr, Discrete log relation $\mathcal{R} = \{ \ (\alpha, u) \in \mathbb{Z}_q \times \mathbb{G} : \ g^{\alpha} = u \ \}$

How about prove $R_1 \wedge R_2 = \{ \ (x_1, x_2; h_1, h_2) \in Z_q^2 \times G^2 : h_1 = g^{x_1} \ and \ h_2 = g^{x_2} \}$

R_1 and R_2 are Discrete log relations

$G =< g >$ is group of order p

AND composition of SIGAMA: Parallel attempt

How about prove $R_1 \wedge R_2 = \{\, (x_1, x_2; h_1, h_2) \in Z_q^2 \times G^2 : h_1 = g^{x_1}\ and\ h_2 = g^{x_2} \}$

$h_1 = g^{x_1}\ and$

Prover		Verifier
$u_1, u_2 \in_R \mathbb{Z}_n$ $a_1 \leftarrow g^{u_1}$		
	$\xrightarrow{a_1,}$	
	$\xleftarrow{c_1,}$	$c_1, c_2 \in_R \mathbb{Z}_n$
$r_1 \leftarrow_n u_1 + c_1 x_1$		
	$\xrightarrow{r_1,}$	
		$g^{r_1} \stackrel{?}{=} a_1 h^{c_1}$

Run two Schnorr protocols independently???

AND composition of SIGAMA: Better solution

How about prove $R_1 \wedge R_2 = \{\, (x_1, x_2; h_1, h_2) \in Z_q^2 \times G^2 : h_1 = g^{x_1} \text{ and } h_2 = g^{x_2} \}$

$h_1 = g^{x_1} \text{ and } h_2 = g^{x_2}$

Prover		Verifier
$(x_1 = \log_g h_1, x_2 = \log_g h_2)$		
$u_1, u_2 \in_R \mathbb{Z}_n$		
$a_1 \leftarrow g^{u_1}$		
$a_2 \leftarrow g^{u_2}$	$\xrightarrow{a_1, a_2}$	
		$c \in_R \mathbb{Z}_n$
	$\xleftarrow{c}$	
$r_1 \leftarrow_n u_1 + c x_1 \mod q$		
$r_2 \leftarrow_n u_2 + c x_2 \mod q$	$\xrightarrow{r_1, r_2}$	
		$g^{r_1} \stackrel{?}{=} a_1 h_1^c$
		$g^{r_2} \stackrel{?}{=} a_2 h_2^c$

The same challenge is applied to two proofs

OR composition of SIGAMA

Schnorr, Discrete log $\mathcal{R} = \{\ (\alpha, u) \in \mathbb{Z}_q \times \mathbb{G} :\ \ g^{\alpha} = u\ \}$

AND Composition $R_1 \wedge R_2 = \{\ (x_1, x_2; h_1, h_2) \in Z_q^2 \times G^2 \colon h_1 = g^{x_1}\ and\ h_2 = g^{x_2}\}$

OR Composition $R_1 \vee R_2 = \{\ (x_1\ or\ x_2; h_1, h_2) \in Z_q\ \times G^2 \colon h_1 = g^{x_1}\ or\ h_2 = g^{x_2}\}$

R_1 and R_2 are Discrete log relations

OR composition of SIGAMA

How about prove $R_1 \vee R_2 = \{ (x_1 \ or \ x_2; h_1, h_2) \in Z_q \times G^2 \colon h_1 = g^{x_1} \ or \ h_2 = g^{x_2}\}$

The simulation

The real Schnorr

Prover $\quad h_1 = g^{x_1} \ or \ h_2 = g^{x_2} \quad$ Verifier

(using $x_2 = \log_g h_2$)

$c_1, r_1, u_2 \in_R \mathbb{Z}_n$

$a_1 \leftarrow g^{r_1} h_1^{-c_1}$

$a_2 \leftarrow g^{u_2}$

$\xrightarrow{a_1, a_2}$

$c \in_R \mathbb{Z}_n$

$\xleftarrow{c}$

$c_2 \leftarrow_n c - c_1$

$r_2 \leftarrow_n u_2 + c_2 x_2$

$\xrightarrow{c_1, c_2, r_1, r_2}$

$c_1 + c_2 \stackrel{?}{=}_n c$

$g^{r_1} \stackrel{?}{=} a_1 h_1^{c_1}$

$g^{r_2} \stackrel{?}{=} a_2 h_2^{c_2}$

- $c = c_1 + c_2$
- Simulate a valid transection for unknown witness but known challenge
- Generate the real Schnorr for known witness but unknown challenge

Question 1: 3 OR composition of SIGAMA

OR Composition $R_1 \vee R_2 = \{ (x_1\ or\ x_2; h_1, h_2) \in Z_q \times G^2 : h_1 = g^{x_1}\ or\ h_2 = g^{x_2} \}$

3OR Composition

$$R_1 \vee R_2 \vee R_3 = \{ (x_1, x_2\ or\ x_3; h_1, h_2, h_3) \in Z_q \times G^2 : \\ h_1 = g^{x_1}\ or\ h_2 = g^{x_2}\ or\ h_3 = g^{x_3} \}$$

- $c = c_1 + c_2 + c_3$
- Simulate two valid transections for unknown witness but known challenge
- Generate a real Schnorr for known witness but unknown challenge

R_1, R_2 and R_3 are Discrete log relations

Question 2: AND-OR composition of SIGAMA

AND Composition $R_1 \wedge R_2 = \{\, (x_1, x_2; h_1, h_2) \in Z_q^2 \times G^2 : h_1 = g^{x_1} \ and\ h_2 = g^{x_2} \}$

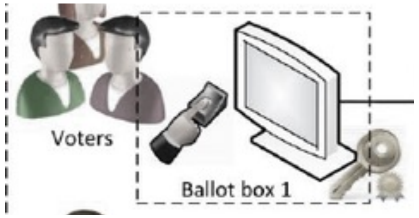
OR Composition $R_1 \vee R_2 = \{\, (x_1\ or\ x_2; h_1, h_2) \in Z_q \times G^2 : h_1 = g^{x_1}\ or\ h_2 = g^{x_2} \}$

How about relation $(R_1 \vee R_2) \wedge (R_3 \vee R_4)$

R_1, R_2, R_3 and R_4 are Discrete log relations

The second Assignment, I will give concrete requirement in next lecture.

Electronic Voting (e-voting)



Candidates:
Alice,
Bob,
Tom,
Tony,
…

ElGamal Enc for privacy

$$G = \langle g \rangle$$

$$pk := h = g^s, sk := s$$

For Alice $g^{\beta_1}, h^{\beta_1} \cdot g^{b_1}$

Cheating Voter $b_1 = 1000$

Thus, the voter needs to prove this is a ElGamal enc of 0 or 1
While no knowledge of b_1 is leaked

This is what Zero-knowledge proof can solve

OR-composition of ID_{DDH}

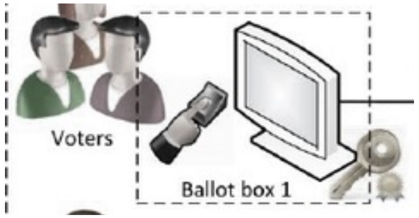
- We are ready to give such zero-knowledge proof
- Given $G =< g >, pk = u = g^s$
- and ciphertext $v = g^{\beta}, e = u^{\beta} \cdot g^{b}$
- Proof the following relation

$$\mathcal{R} := \left\{ \ (\ (b, \beta),\ (u, v, e)\ )\ :\ v = g^{\beta},\ \ e = u^{\beta} \cdot g^{b},\ \ b \in \{0, 1\} \ \right\}.$$

(u, v, e) is the encryption of 0 or 1 if and only if (g, u, v, e) is a DDH tuple or $(g, u, v, e/g)$ is a DDH tuple

We only need an OR-composition of ID_{DDH} to show that (g, u, v, e) is a DDH tuple or $(g, u, v, e/g)$ is a DDH tuple

Applications: e-voting



ElGamal Enc for privacy

$G =< g >$

$pk := u = g^s, sk := s$

For Alice $\quad v = g^{\beta_1}, e = h^{\beta_1} \cdot g^{b_1}$

Π

OR-composition proof Π of ID_{DDH} to show that
(g, u, v, e) is a DDH tuple or$(g, u, v, e/g)$ is a DDH tuple

A short summary: SIGMA protocol

- Identification protocol is a generalization of Identification protocol

- To proof that P knows witness x of statement y such that $(x, y) \in R$

- SIGMA for several relations
- OR and AND composition of SIGMA protocol

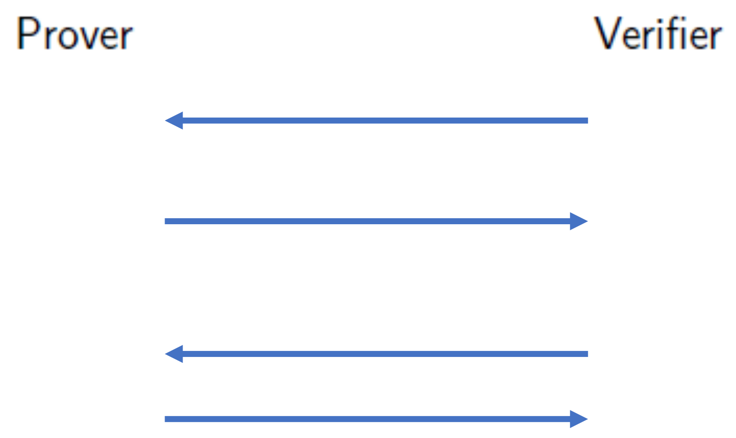
Applications: e-voting

Zero-knowledge proof

Zero-knowledge proof

- Zero-knowledge proof is an extension of SIGMA protocol
- The interactive is not necessary of 3-pass
- The soundness is not necessary of proof-of-knowledge
- The zero-knowledge should be hold for any verifier

$y \in L$ if and only if $\exists$ withness x, such that $(x, y) \in R$



- **Correctness(Completeness):** If $y \in L$, P and V execute the protocol honestly, the proof is accepted.
- **Soundness**: If $y \notin L$, for any (computational) P, V accepts with negligible probability
- **Zero-knowledge**: For any V, without knowing witness x, we can generate (simulate) the valid transaction efficiently for $y \in L$

Zero Knowledge Proof for NP language

- Let L be an NP language
- Prover with input (x, y) wants to prove that $y \in L$

➨ if $y \in L$, verifier accept

➨ if $y \notin L$, for any (PPT) prover, verifier will reject

➨ Zero-knowledge: any verifier learns nothing about the witness x

Zero Knowledge Proof (ZKP) for NP

Theorem [GMW86]

Commitment ---> ZKP for all of NP

[GMW86] O Goldreich, S Micali, A Wigderson, Proofs that yield nothing but their validity or all languages in NP have zero-knowledge proof systems, 1986

Zero Knowledge Proof for NP

- To prove that $\exists$ input x such that $C(x) = y$, where C is any polynomial size circuit.

- Circuit C could be:
  - $ax^2 + bx + c$
  - Polynomial function Poly(x)
  - Machine learning algorithms
  - Etc……..

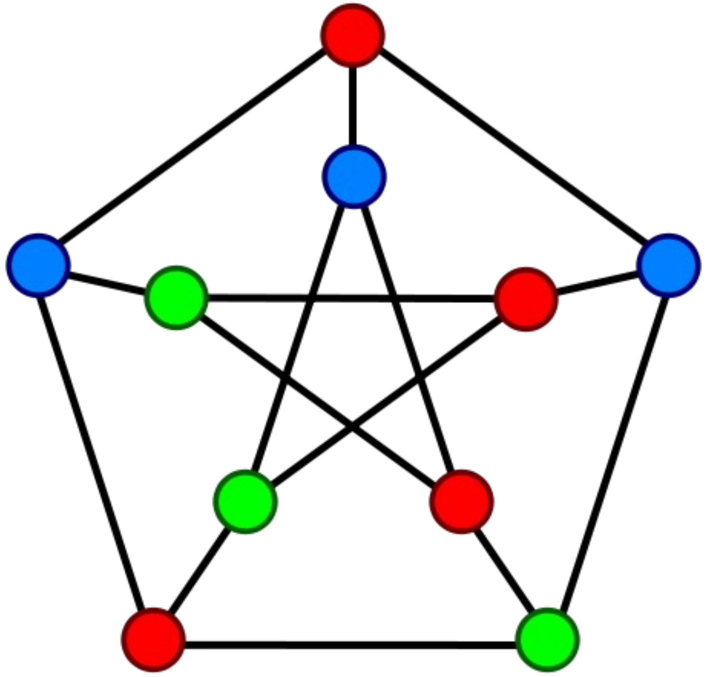
ZKP for 3-colorable Graphs

- Let G=(V, E) be graphs on n vertices and define V = $\{ v_1 , \ldots , v_n\}$ be the set of vertices, and E = $\{e_{i,j}\colon \exists\ edge\ e_{i,j}\ between\ v_i, v_j\}$ be the set of edges.

- we say that a graph G is 3-colorable (or G∈ $3COL$)

if there is a function $c : V \rightarrow \{R, G, B\}$

such that for every edge (v_i, v_j) ∈ E, $c(v_i) \neq c(v_j)$



ZKP for 3-colorable Graphs

- Why?

- The reason is that a protocol for $3COL$ actually implies a protocol for all languages in NP, since $3COL$ is NP-complete
- It means that we have a function **Reduce** that on input a NP language instance y, outputs a graph G such that

$$y \in L \text{ iff } G \in 3COL$$

what's more, there exists **Reduce'** on input witness x for $y \in L$ outputs witness for G $\in 3COL$

- This can be used for the prover to convert their proof for any NP into a proof for the $3COL$ protocol.

A tool: Commitment

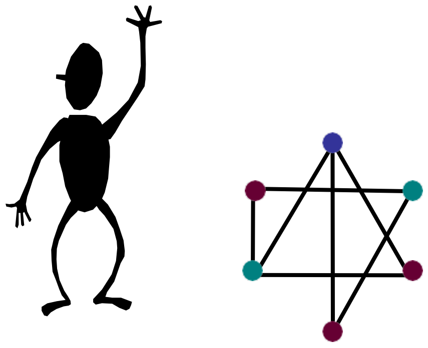
- A commitment Com is a 3-tuple algorithms (Setup, Commit, Verify)
  - Setup: Generate public parameters pp
  - Commit(m): Compute a commitment c to m with its opening d, and output c
  - Verify(c,m, d): indicate the validation of (m, d) with respect to commitment c
- A commitment could be statistical hiding and computational binding, or computational hiding and statistical hiding. For the first one

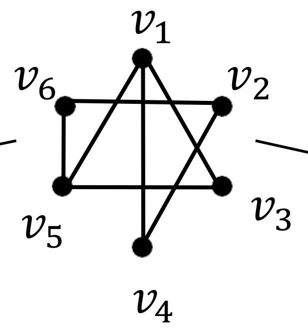
  - *Hiding*: For any $m, m' \in \mathcal{M}_{\text{com}}$, their commitments are statistical indistinguishable.
  - *Binding*: No probability polynomial time (PPT) adversary could open a commitment c on two different messages.

- Ex: Commit (m) as Hash(m, d) for randomness d, Hash could be SHA256
  - Hiding: random oracle of Hash
  - Biding: collision resistance

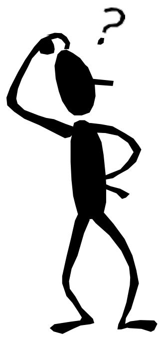
ZKP for 3-colorable Graphs

Prover





Verifier



1. Randomly permute coloring & send in Commitment

Commit(●)…Commit(●) →

2. Pick random edge.

← (1,4)

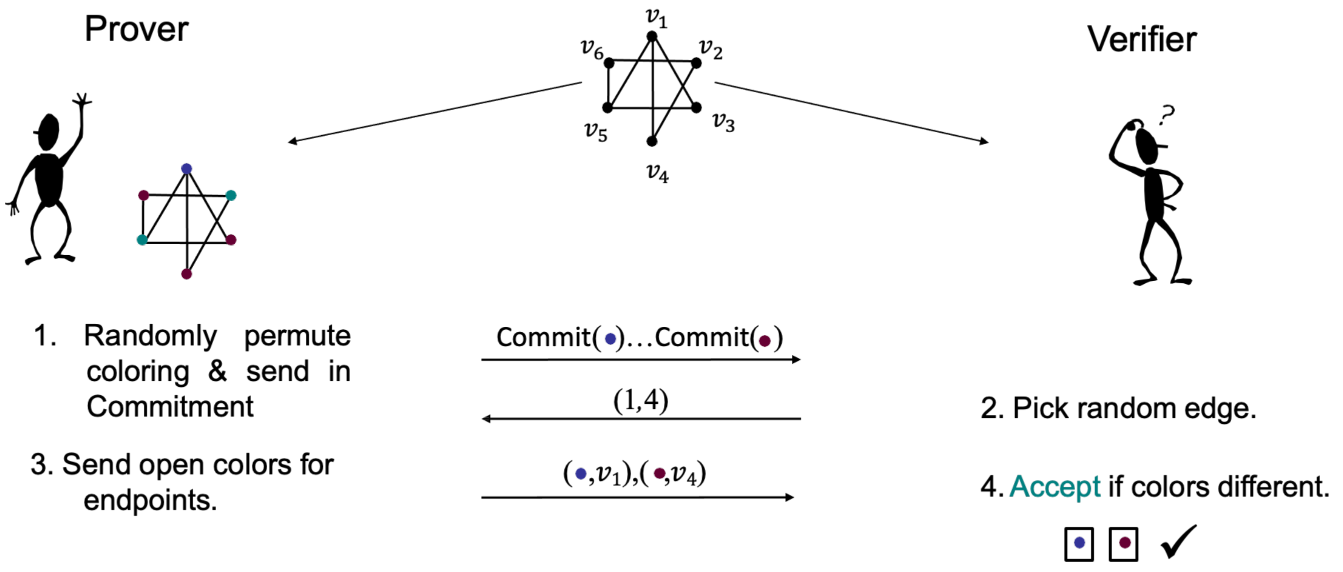
3. Send open colors for endpoints.

(●,v_1),(●,v_4) →

4. Accept if colors different.

☐ ☐ ✓

ZKP for 3-colorable Graphs



- **Correctness(Completeness):** easy.
- **Soundness:** If it is not 3-colorable, for any (computational) P, V accepts with probability less than $1 - 1/|E|$
- Implied by the biding of Commit

ZKP for 3-colorable Graphs

- **(Honest verifier) Zero-knowledge**:

- Step 1: Pick random index i, j
- Step 2: Commit$(0\ )$, …, Commit(0), and only two of them (with index i, j) are different R, G, or B
- When getting (i', j') from verifier, if $(i', j')\ =(i, j)$ open commit,

  otherwise return to Step 1

- Imply by the Hiding of Commit

A short Summary

Theorem [GMW86]
Commitment ---> ZKP for all of NP

[GMW86] O Goldreich, S Micali, A Wigderson, Proofs that yield nothing but their validity or all languages in NP have zero-knowledge proof systems, 1986

Non-interactive Zero Knowledge (NIZK)

- Non-interactive is better than interactive (latency)

- NIZK→signature, e-voting, etc.

- NIZK only exists for L in BPP, which is not interesting than NP

- However, with the setup of common random string,…
- Or random oracle…

Blum, Feldman, Micali. Non-interactive zero knowledge and its applications
Fiat, Shamir: How to prove yourself: practical solutions to identification and signature problems

NIZK assuming random oracle

$\underline{P(\alpha)}$ $\quad u = g^{\alpha} \quad$ $\underline{V(u)}$

$\alpha_t \xleftarrow{R} \mathbb{Z}_q,\ u_t \leftarrow g^{\alpha_t}$

$\xrightarrow{\quad u_t \quad}$

$c \xleftarrow{R} \mathcal{C}$

$\xleftarrow{\quad c \quad}$

$\alpha_z \leftarrow \alpha_t + \alpha c \mod q$

$\xrightarrow{\quad \alpha_z \quad}$

$g^{\alpha_z} \stackrel{?}{=} u_t \cdot u^c$

$c = H(u, u_t, m)$

$\longrightarrow$

Fiat-Shamir

NIZK (without m)
Signature (with m)

Blum, Feldman, Micali. Non-interactive zero knowledge and its applications
Fiat, Shamir: How to prove yourself: practical solutions to identification and signature problems

Succinct Non-Interactive Proof (zkSNARK)

- It is better if we have a very small (Succinct) proof
- And the verification of the proof is efficient.
- These proof is called Succinct Non-Interactive Proof (zkSNARK)

zk-SNARK/STARK

- Consider the complexity of Verifier.
- Could it be less than computing $R(x, w)$?????

- YES!!!!

PCP Theorem [AS,ALMSS,Dinur]:
NP statements have polynomial-size PCPs in which the verifier reads only O(1) bits.

– Can be made ZK with small overhead [KPT97,IW04]

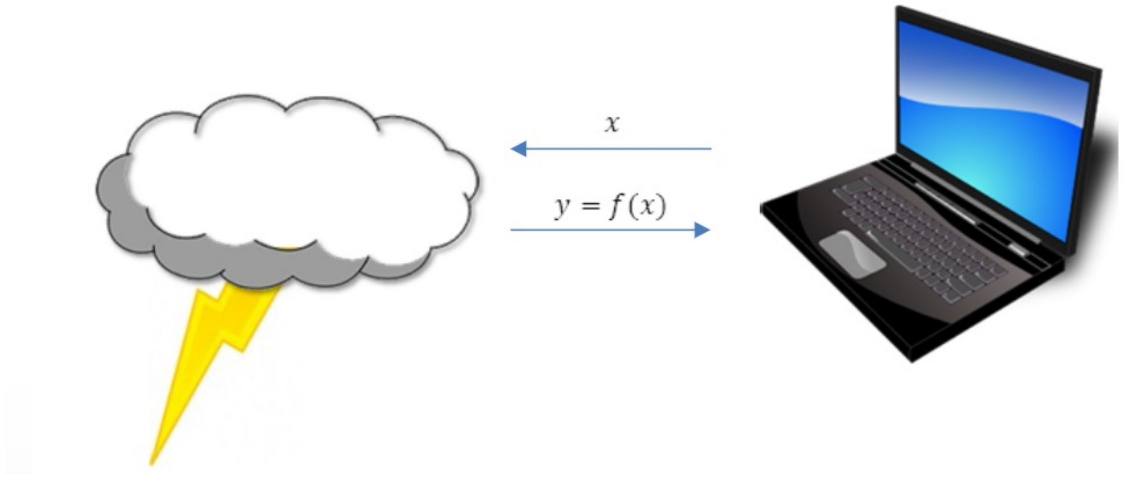
zkSNARK

- Verifiable Outsourcing computation

- Blockchain

Verifiable Outsourcing computation

We do not want to trust the cloud, but would like to use its power.



Cloud appends a zkSNARK Π to proof that $y = f(x)$

zk-SNARK/STARK

	SNARKs	STARKs	Bulletproofs
Algorithmic complexity: prover	O(N * log(N))	O(N * poly-log(N))	O(N * log(N))
Algorithmic complexity: verifier	~O(1)	O(poly-log(N))	O(N)
Communication complexity (proof size)	~O(1)	O(poly-log(N))	O(log(N))
- size estimate for 1 TX	Tx: 200 bytes, Key: 50 MB	45 kB	1.5 kb
- size estimate for 10.000 TX	Tx: 200 bytes, Key: 500 GB	135 kb	2.5 kb
Ethereum/EVM verification gas cost	~600k (Groth16)	~2.5M (estimate, no impl.)	N/A
Trusted setup required?	YES 😒	NO 😁	NO 😁
Post-quantum secure	NO 😒	YES 😁	NO 😒
Crypto assumptions	DLP + secure bilinear pairing 😒	Collision resistant hashes 😁	Discrete log 😏

Thank you