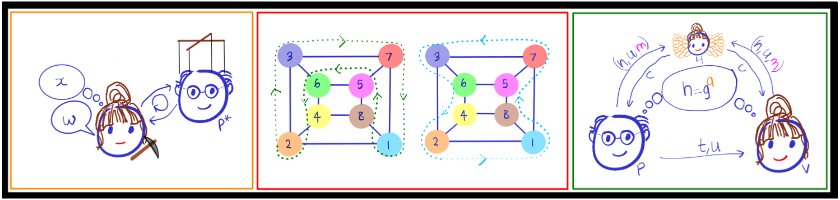
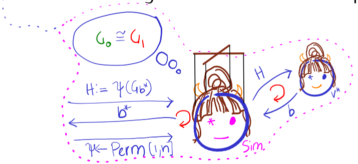


CS783: Theoretical Foundations of Cryptography

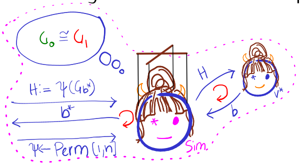
Lecture 16 (04/Oct/24)

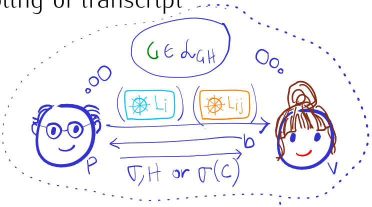
Instructor: Chethan Kamath

- Malicious-verifier perfect ZKP for GI
  - Simulator was expected polynomial-time
  - Takeaway: out of order sampling of transcript



- Malicious-verifier perfect ZKP for GI
  - Simulator was expected polynomial-time
  - Takeaway: out of order sampling of transcript

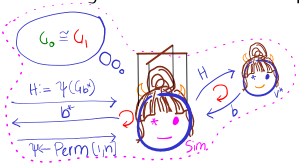


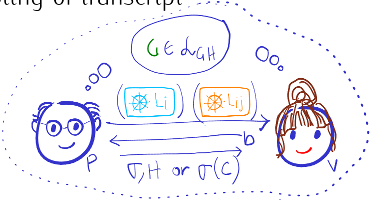


- (Computational) ZKP for **NP**
  - Blum's protocol for Graph Hamiltonicity using lockers
  - Locker computationally hides ⇒ ZK

Recall from Last Lecture

- Malicious-verifier perfect ZKP for GI
  - Simulator was expected polynomial-time
  - Takeaway: out of order sampling of transcript



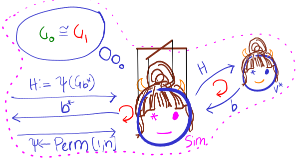


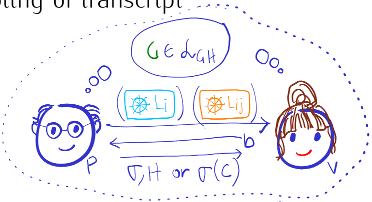
- (Computational) ZKP for **NP**
  - Blum's protocol for Graph Hamiltonicity using lockers
  - Locker computationally hides ⇒ ZK
  - What about perfect/statistical ZKP for **NP**?

    ⚠ Not possible (unless polynomial hierarchy collapses)!

Recall from Last Lecture

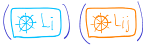
- Malicious-verifier perfect ZKP for GI
  - Simulator was expected polynomial-time
  - Takeaway: out of order sampling of transcript





- (Computational) ZKP for **NP**
  - Blum's protocol for Graph Hamiltonicity using lockers
  - Locker computationally hides ⇒ ZK
  - What about perfect/statistical ZKP for **NP**?

    ⚠ Not possible (unless polynomial hierarchy collapses)!

- Commitment schemes: digital lockers 

  - Non-interactive constructions from PKE and OWP
  - Two-message construction from PRG ← OWF

Commitment Schemes are Digital Lockers

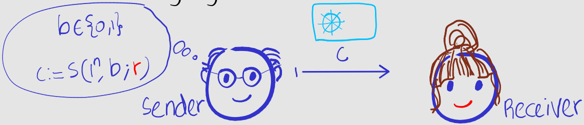
Defintion 1

A (non-interactive) bit commitment scheme is a pair of algorithms (S, R) *with the following syntax:*

Commitment Schemes are Digital Lockers

Defintion 1

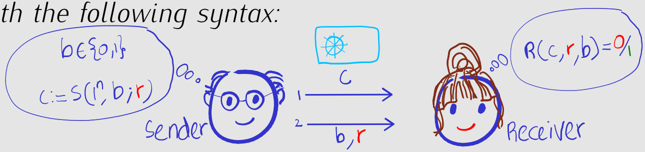
A (non-interactive) bit commitment scheme is a pair of algorithms (S, R) *with the following syntax:*



Commitment Schemes are Digital Lockers

Defintion 1

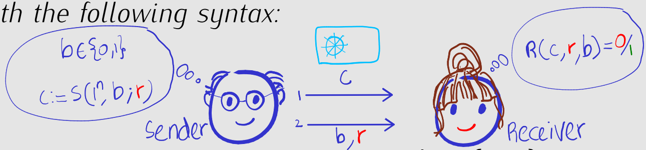
A (non-interactive) bit commitment scheme is a pair of algorithms (S, R) *with the following syntax:*



Commitment Schemes are Digital Lockers

Defintion 1

A (non-interactive) bit commitment scheme is a pair of algorithms (S, R) *with the following syntax:*



- *Correctness: for all* $n \in \mathbb{N}$ *and inputs* $b \in \{0, 1\}$*:*

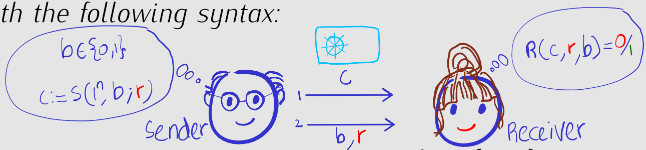
$$\Pr_r\left[R(S(1^n, b; r), r, b) = 1\right] = 1$$

- *Computational hiding: PPT adversary cannot distinguish commitment to* 0 *from commitment to* 1

Commitment Schemes are Digital Lockers

Defintion 1

A (non-interactive) bit commitment scheme is a pair of algorithms (S, R) *with the following syntax:*



- *Correctness: for all* $n \in \mathbb{N}$ *and inputs* $b \in \{0,1\}$*:*

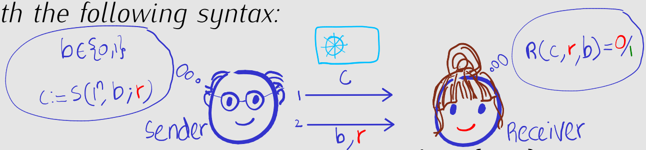
$$\Pr_r\left[R(S(1^n,b;r),r,b)=1\right]=1$$

- *Computational hiding: PPT adversary cannot distinguish commitment to* 0 *from commitment to* 1
- *Perfect binding: for any* $c \in \{0,1\}^*$*, there do not exist openings* $r_0, r_1 \in \{0,1\}^*$ *such that* $\mathsf{R}(c, r_0, 0) = \mathsf{R}(c, r_1, 1) = 1$

Commitment Schemes are Digital Lockers

Defintion 1

A (non-interactive) bit commitment scheme is a pair of algorithms (S, R) *with the following syntax:*



- *Correctness: for all* $n \in \mathbb{N}$ *and inputs* $b \in \{0, 1\}$*:*

$$\Pr_r\left[R(S(1^n, b; r), r, b) = 1\right] = 1$$

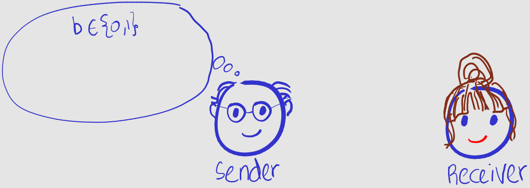
- *Computational hiding: PPT adversary cannot distinguish commitment to* 0 *from commitment to* 1
- *Perfect binding: for any* $c \in \{0, 1\}^*$*, there do not exist openings* $r_0, r_1 \in \{0, 1\}^*$ *such that* $\mathsf{R}(c, r_0, 0) = \mathsf{R}(c, r_1, 1) = 1$

- In general the commit phase can be interactive $\leftrightarrows$

Bit Commitment ← OWP

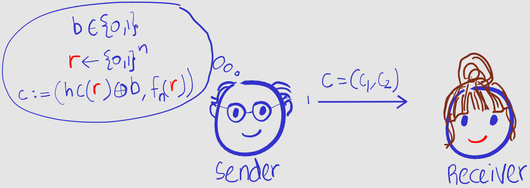
Construction 1 (OWP $f_n : \{0,1\}^n \to \{0,1\}^n \to$ *bit*-commitment Σ)

- *Recall: every (leaky)* f_n *has hard-core predicate* $\mathrm{hc} : \{0,1\}^n \to \{0,1\}$



Construction 1 (OWP $\mathsf{f}_n : \{0,1\}^n \to \{0,1\}^n \to$ *bit*-commitment Σ)

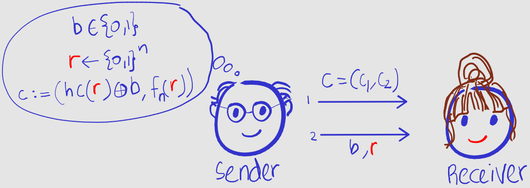
- *Recall: every (leaky)* f_n *has hard-core predicate* $\mathsf{hc} : \{0,1\}^n \to \{0,1\}$



Bit Commitment ← OWP

Construction 1 (OWP $f_n : \{0,1\}^n \to \{0,1\}^n \to$ *bit*-commitment Σ)

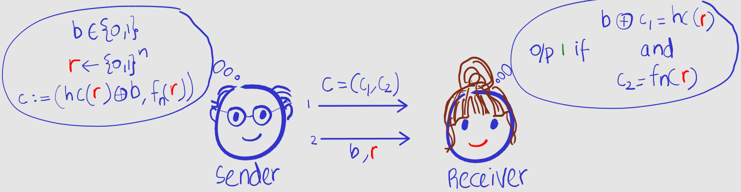
- *Recall: every (leaky)* f_n *has hard-core predicate* $hc : \{0,1\}^n \to \{0,1\}$



Bit Commitment $\leftarrow$ OWP

Construction 1 (OWP $f_n : \{0,1\}^n \to \{0,1\}^n \to$ *bit*-commitment Σ)

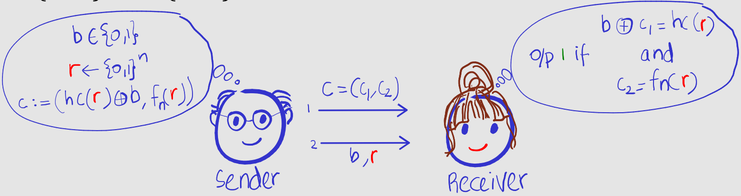
- *Recall: every (leaky)* f_n *has hard-core predicate* $hc : \{0,1\}^n \to \{0,1\}$



Bit Commitment ← OWP

Construction 1 (OWP $f_n : \{0,1\}^n \to \{0,1\}^n \to$ *bit*-commitment Σ)

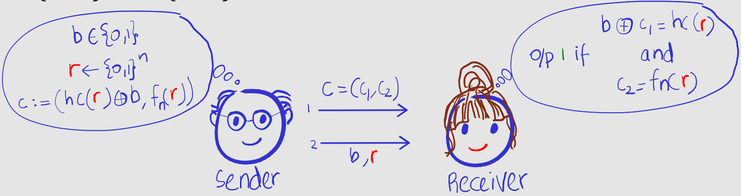
- *Recall: every (leaky)* f_n *has hard-core predicate* $\mathrm{hc} : \{0,1\}^n \to \{0,1\}$



- Security of hard-core predicate hc ⇒ computational hiding
- f_n permutation ⇒ perfect binding

Construction 1 (OWP $f_n : \{0,1\}^n \to \{0,1\}^n \to$ *bit*-commitment Σ)

- *Recall: every (leaky)* f_n *has hard-core predicate* $hc : \{0,1\}^n \to \{0,1\}$



- Security of hard-core predicate hc ⇒ computational hiding
- f_n permutation ⇒ perfect binding

Exercise 1

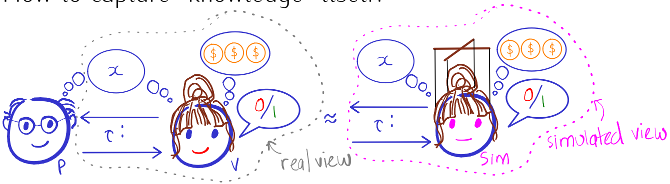
1. *Formally describe the construction, and write down the proof*
2. *Given a bit-commitment, construct a commitment for* $\{0,1\}^\ell$

Plan for Today's Lecture...

- Proof of knowledge (PoK): soundness $\xrightarrow{++}$ *knowledge soundness*

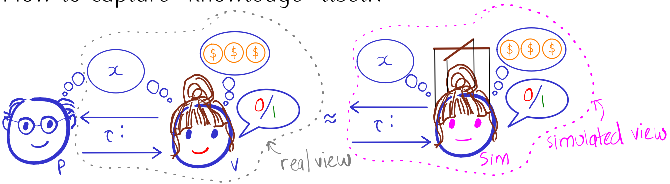
Plan for Today's Lecture...

- Proof of knowledge (PoK): soundness $\xrightarrow{++}$ *knowledge soundness*
  - We have captured "gaining knowledge" via simulator
  - How to capture "knowledge" itself?

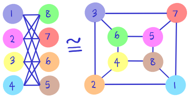


Plan for Today's Lecture...

- Proof of knowledge (PoK): soundness $\xrightarrow{++}$ *knowledge soundness*
  - We have captured "gaining knowledge" via simulator
  - How to capture "knowledge" itself?

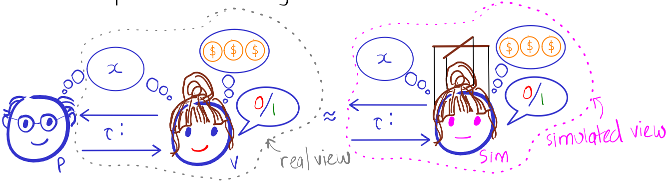


- Zero-knowledge PoK for
  1. Graph Isomorphism
  2. Discrete-log problem: Schnorr's protocol

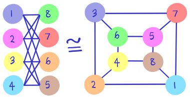


Plan for Today's Lecture...

- Proof of knowledge (PoK): soundness $\xrightarrow{++}$ *knowledge soundness*
  - We have captured "gaining knowledge" via simulator
  - How to capture "knowledge" itself?



- Zero-knowledge PoK for
  1. Graph Isomorphism
  2. Discrete-log problem: Schnorr's protocol



- Fiat-Shamir Transform

  - Interactive protocol $\xrightarrow{\text{Random Oracle}}$ non-interactive protocol
  - Digital signature from Schnorr's protocol

Plan for Today's Lecture

Plan for Today's Lecture

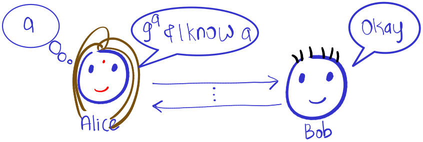
Recall Definition of Zero-Knowledge Proof (for NP)

- Completeness
- Soundness
- Zero-knowledge (ZK)

Recall Definition of Zero-Knowledge Proof (for NP)

- Completeness
- Soundness
- Zero-knowledge (ZK)

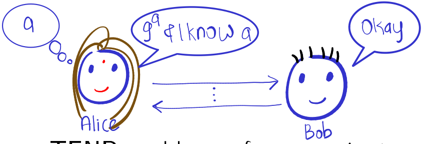
- In some situations, stronger guarantee is needed: V should be convinced that P *knows* a witness
  - Identification, e.g., for ElGamal PKE in cyclic group $\mathbb{G}$
    - Public key is $h := g^a$ and secret key is the discrete log a
    - Owner has to prove they *possess* a (such an a always exists)

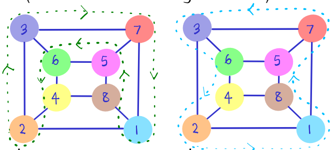


Recall Definition of Zero-Knowledge Proof (for NP)

- Completeness
- Soundness
- Zero-knowledge (ZK)

- In some situations, stronger guarantee is needed: V should be convinced that P *knows* a witness
  - Identification, e.g., for ElGamal PKE in cyclic group $\mathbb{G}$
    - Public key is $h := g^a$ and secret key is the discrete log a
    - Owner has to prove they *possess* a (such an a always exists)





  - TFNP problems: for every instance there exists a solution
    - Smith: given 3-regular graph with a Ham. cycle, find one more
    - Solver wants to prove they have *found* the second Ham. cycle

How to Quantify Knowledge?

- For defining ZK, we only quantified "gain of knowledge"
  - "V gains no knowledge" if anything V can *compute* after the interaction with P, it could have computed *without it*

How to Quantify Knowledge?

- For defining ZK, we only quantified “gain of knowledge”
  - “V gains no knowledge” if anything V can *compute* after the interaction with P, it could have computed *without it*
  - Formalised via “simulation paradigm”: V’s view can be *efficiently simulated* given only the instance x

How to Quantify Knowledge?

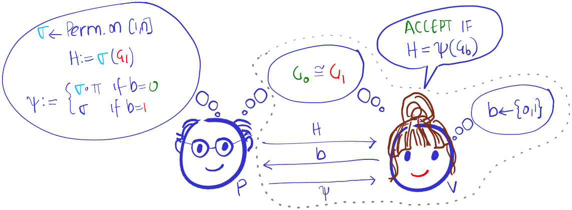
- For defining ZK, we only quantified "gain of knowledge"
  - "V gains no knowledge" if anything V can *compute* after the interaction with P, it could have computed *without it*
  - Formalised via "simulation paradigm": V's view can be *efficiently simulated* given only the instance x

? How would you quantify "knowledge" itself?

How to Quantify Knowledge?

- For defining ZK, we only quantified "gain of knowledge"
  - "V gains no knowledge" if anything V can *compute* after the interaction with P, it could have computed *without it*
  - Formalised via "simulation paradigm": V's view can be *efficiently simulated* given only the instance x
- How would you quantify "knowledge" itself?
  - For a student: get hold of student, hold viva, extract answers

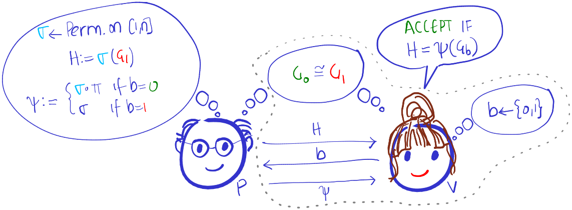
- For defining ZK, we only quantified "gain of knowledge"
  - "V gains no knowledge" if anything V can *compute* after the interaction with P, it could have computed *without it*
  - Formalised via "simulation paradigm": V's view can be *efficiently simulated* given only the instance x
- ? How would you quantify "knowledge" itself?
  - For a student: get hold of student, hold viva, extract answers



  - For P in Π_{GI}?

How to Quantify Knowledge?

- For defining ZK, we only quantified "gain of knowledge"
  - "V gains no knowledge" if anything V can *compute* after the interaction with P, it could have computed *without it*
  - Formalised via "simulation paradigm": V's view can be *efficiently simulated* given only the instance x
- How would you quantify "knowledge" itself?
  - For a student: get hold of student, hold viva, extract answers



  - For P in Π_{GI}? Should be possible to *efficiently extract* isomorphism π given access to P
  - In general, for NP: should be possible to extract a witness w

Let's Define Zero-Knowledge Proof of Knowledge...

Defintion 2 (ZKPoK)

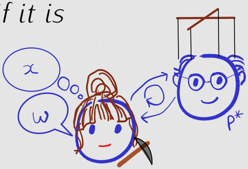
An interactive protocol $\Pi = (\mathsf{P}, \mathsf{V})$ *for an* NP *language* $\mathcal{L}$ *is a zero-knowledge proof of knowledge if it is*

1. *Complete*
2. *Zero knowledge*

Let's Define Zero-Knowledge Proof of Knowledge...

Defintion 2 (ZKPoK)

An interactive protocol $\Pi = (\mathsf{P}, \mathsf{V})$ *for an* NP *language* $\mathcal{L}$ *is a zero-knowledge proof of knowledge if it is*



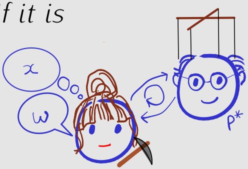
1. *Complete*
2. *Zero knowledge*
3. *Knowledge sound:*
   - $\exists$ *expected polynomial-time extractor* Ext *such that*
   - $\forall$ *prover* P^* *and instance* $\boldsymbol{x}$*:*

$$\Pr_{w \leftarrow \mathsf{Ext}^{\mathsf{P}^*}(x)}[w \textit{ is a witness for } \boldsymbol{x}] \geq \Pr[1 \leftarrow \langle \mathsf{P}^*, \mathsf{V} \rangle(x)] - \epsilon_k$$

Let's Define Zero-Knowledge Proof of Knowledge...

Defintion 2 (ZKPoK)

An interactive protocol $\Pi = (\mathsf{P}, \mathsf{V})$ *for an* NP *language* $\mathcal{L}$ *is a zero-knowledge proof of knowledge if it is*



1. *Complete*
2. *Zero knowledge*
3. *Knowledge sound:*
   - $\exists$ *expected polynomial-time extractor* Ext *such that*
   - $\forall$ *prover* P^* *and instance* $\boldsymbol{x}$*:*

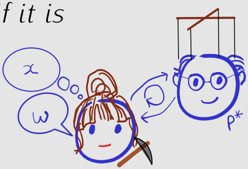
"knowledge error"

$$\Pr_{w \leftarrow \mathsf{Ext}^{\mathsf{P}^*}(x)}[w \textit{ is a witness for } x] \geq \Pr[1 \leftarrow \langle \mathsf{P}^*, \mathsf{V} \rangle(x)] - \epsilon_k$$

Let's Define Zero-Knowledge Proof of Knowledge...

Defintion 2 (ZKPoK)

An interactive protocol $\Pi = (\mathsf{P}, \mathsf{V})$ *for an* NP *language* $\mathcal{L}$ *is a zero-knowledge proof of knowledge if it is*



1. *Complete*
2. *Zero knowledge*
3. *Knowledge sound:*
   - $\exists$ *expected polynomial-time extractor* Ext *such that*
   - $\forall$ *prover* P^* *and instance* x*:*

"knowledge error"

$$\Pr_{w \leftarrow \mathsf{Ext}^{\mathsf{P}^*}(x)}[w \text{ is a witness for } x] \geq \Pr[1 \leftarrow \langle \mathsf{P}^*, \mathsf{V}\rangle(x)] - \epsilon_k$$

- Trivial if we omit either of 2 or 3

- Ext must do something *more* than V, e.g. "rewind" P^*

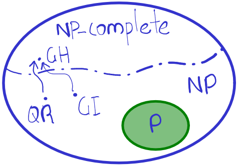
Let's Define Zero-Knowledge Proof of Knowledge...

Exercise 2 (PoK implies soundness)

Show that if an IP has knowledge error at most ϵ_k then its soundness error $\epsilon_s \leq \epsilon_k$.

Exercise 3

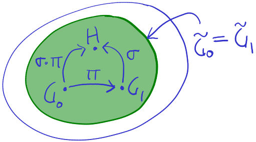
Does this notion make sense beyond NP*?*



Plan for Today's Lecture

Observation: transitivity of isomorphism

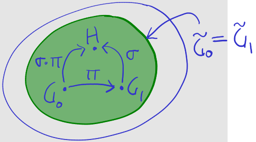
- $G_0 \cong G_1 \Rightarrow$ if $G_1 \cong H$ then $G_0 \cong H$

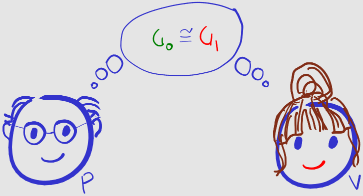


Observation: transitivity of isomorphism

- $G_0 \cong G_1 \Rightarrow$ if $G_1 \cong H$ then $G_0 \cong H$

Protocol 1 ($\Pi_{GI} = (\mathsf{P}, \mathsf{V})$: IP for $\mathcal{L}_{GI}$)





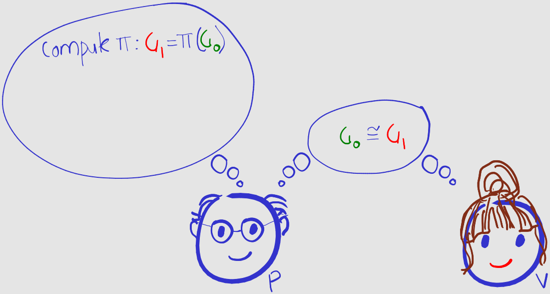
1. P *"commits" by sending a random* H *s.t.* $G_1 \cong H$
2. *For* $b \leftarrow \{0, 1\}$, V *challenges* P *to "reveal"* ψ *s.t.* $G_b \cong H$
3. V *accepts if* $\psi(G_b) = H$

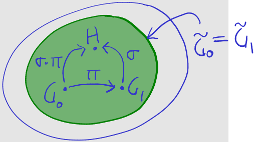
Recall Π_{GI}: ZKP for GI...

Observation: transitivity of isomorphism

- $G_0 \cong G_1 \Rightarrow$ if $G_1 \cong H$ then $G_0 \cong H$

Protocol 1 ($\Pi_{\mathrm{GI}} = (\mathsf{P}, \mathsf{V})$: IP for $\mathcal{L}_{\mathrm{GI}}$)





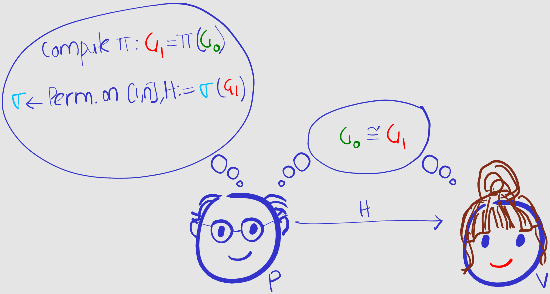
1. P *"commits" by sending a random* H *s.t.* $G_1 \cong H$
2. *For* $b \leftarrow \{0,1\}$, V *challenges* P *to "reveal"* ψ *s.t.* $G_b \cong H$
3. V *accepts if* $\psi(G_b) = H$

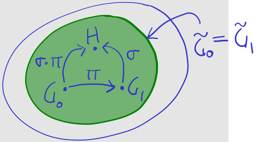
Recall Π_{GI}: ZKP for GI...

Observation: transitivity of isomorphism

- $G_0 \cong G_1 \Rightarrow$ if $G_1 \cong H$ then $G_0 \cong H$

Protocol 1 ($\Pi_{GI} = (\mathsf{P}, \mathsf{V})$: IP for $\mathcal{L}_{GI}$)





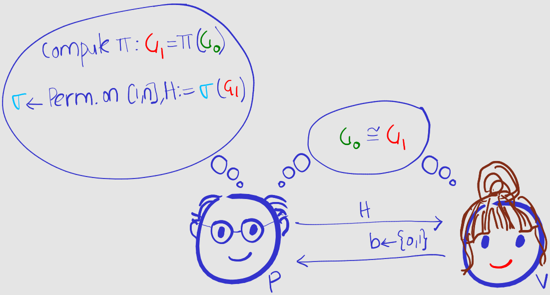
1. P *"commits" by sending a random H s.t. $G_1 \cong H$*
2. *For $b \leftarrow \{0,1\}$,* V *challenges* P *to "reveal" ψ s.t. $G_b \cong H$*
3. V *accepts if $\psi(G_b) = H$*

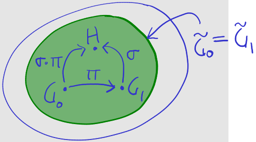
Recall Π_{GI}: ZKP for GI...

Observation: transitivity of isomorphism

- $G_0 \cong G_1 \Rightarrow$ if $G_1 \cong H$ then $G_0 \cong H$

Protocol 1 ($\Pi_{\mathrm{GI}} = (\mathrm{P}, \mathrm{V})$: IP for $\mathcal{L}_{\mathrm{GI}}$)





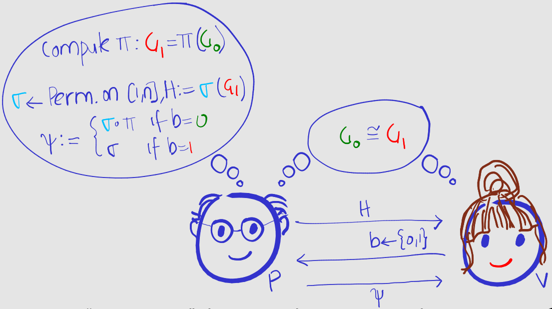
1. P *"commits" by sending a random* H *s.t.* $G_1 \cong H$
2. *For* $b \leftarrow \{0,1\}$, V *challenges* P *to "reveal"* ψ *s.t.* $G_b \cong H$
3. V *accepts if* $\psi(G_b) = H$

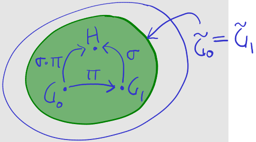
Recall Π_{GI}: ZKP for GI...

Observation: transitivity of isomorphism

- $G_0 \cong G_1 \Rightarrow$ if $G_1 \cong H$ then $G_0 \cong H$

Protocol 1 ($\Pi_{GI} = (P, V)$: IP for $\mathcal{L}_{GI}$)



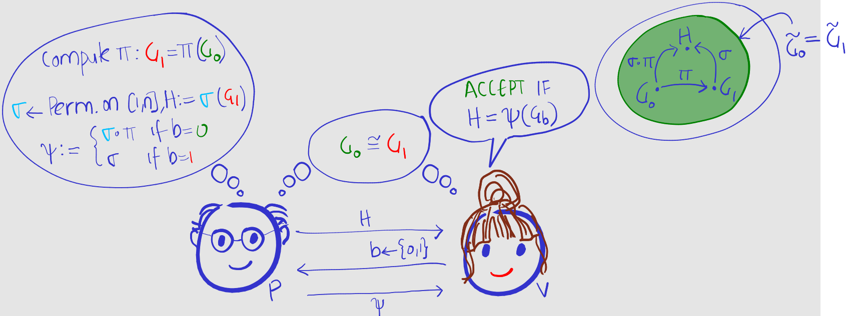


1. P *"commits" by sending a random* H *s.t.* $G_1 \cong H$
2. *For* $b \leftarrow \{0, 1\}$, V *challenges* P *to "reveal"* ψ *s.t.* $G_b \cong H$
3. V *accepts if* $\psi(G_b) = H$

Observation: transitivity of isomorphism

- $G_0 \cong G_1 \Rightarrow$ if $G_1 \cong H$ then $G_0 \cong H$

Protocol 1 ($\Pi_{\mathrm{GI}} = (\mathsf{P}, \mathsf{V})$: IP for $\mathcal{L}_{\mathrm{GI}}$)



1. P *"commits" by sending a random H s.t. $G_1 \cong H$*
2. *For $b \leftarrow \{0,1\}$,* V *challenges* P *to "reveal" ψ s.t. $G_b \cong H$*
3. V *accepts if $\psi(G_b) = H$*

How to Extract π from P^*?

Theorem 1

Π_{GI} *is a ZKPoK for* $\mathcal{L}_{\mathrm{GI}}$ *with* $\epsilon_k \leq 1/2$

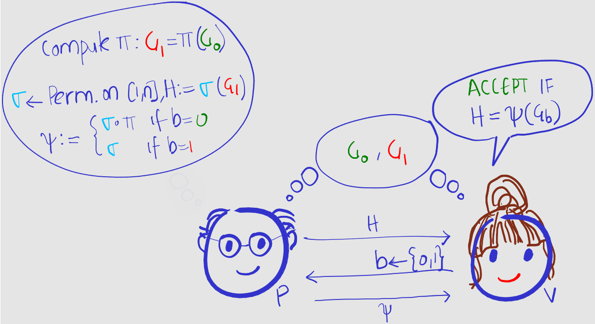
How to Extract π from P*?

Theorem 1

Π_{GI} is a ZKPoK for $\mathcal{L}_{\mathrm{GI}}$ with $\epsilon_k \leq 1/2$

response to challenge 1 response to challenge 0

Proof (of PoK) Hint $\psi_1^{-1} \circ \psi_0 = \sigma^{-1} \circ \sigma \circ \pi = \pi$.



$\square$

Theorem 1

Π_{GI} is a ZKPoK for $\mathcal{L}_{GI}$ with $\epsilon_k \leq 1/2$

response to challenge 1 response to challenge 0

Proof (of PoK) Hint $\psi_1^{-1} \circ \psi_0 = \sigma^{-1} \circ \sigma \circ \pi = \pi$.

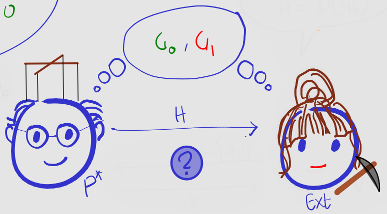
Extraction strategy $Ext^{P^*}(G_0, G_1)$

1) Invoke P^* on (G_0, G_1) to obtain H

Compute π: $G_1 = \pi(G_0)$

$\sigma \leftarrow$ Perm. on $[1,n]$, $H := \sigma(G_1)$

$\psi := \begin{cases} \sigma \circ \pi & \text{if } b=0 \\ \sigma & \text{if } b=1 \end{cases}$



$\square$

How to Extract π from P*?

Theorem 1

Π_{GI} *is a ZKPoK for* $\mathcal{L}_{\mathrm{GI}}$ *with* $\epsilon_k \leq 1/2$

response to challenge 1 response to challenge 0

Proof (of PoK) Hint $\psi_1^{-1} \circ \psi_0 = \sigma^{-1} \circ \sigma \circ \pi = \pi$.

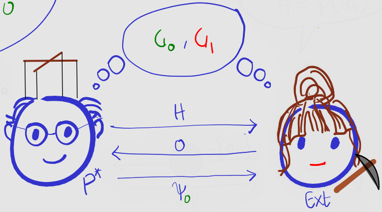
Compute π: $G_1 = \pi(G_0)$

$\sigma \leftarrow$ Perm. on $[1,n]$, $H := \sigma(G_1)$

$\psi := \begin{cases} \sigma \circ \pi & \text{if } b = 0 \\ \sigma & \text{if } b = 1 \end{cases}$

Extraction strategy $\mathrm{Ext}^{P^*}(G_0, G_1)$

1) Invoke P^* on (G_0, G_1) to obtain H
2) Challenge on 0 to get ψ_0



□

How to Extract π from P^*?

Theorem 1

Π_{GI} is a ZKPoK for $\mathcal{L}_{GI}$ with $\epsilon_k \leq 1/2$

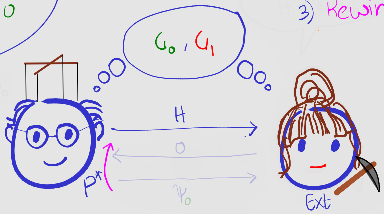
response to challenge 1 / response to challenge 0

Proof (of PoK) Hint $\psi_1^{-1} \circ \psi_0 = \sigma^{-1} \circ \sigma \circ \pi = \pi$.

Compute π: $G_1 = \pi(G_0)$

$\sigma \leftarrow$ Perm. on $[1,n]$, $H := \sigma(G_1)$

$\psi := \begin{cases} \sigma \circ \pi & \text{if } b=0 \\ \sigma & \text{if } b=1 \end{cases}$



Extraction strategy $\mathrm{Ext}^{P^*}(G_0, G_1)$

1) Invoke P^* on (G_0, G_1) to obtain H
2) Challenge on 0 to get ψ_0
3) Rewind P^* to end of 1)

$\square$

How to Extract π from P*?

Theorem 1

Π_{GI} is a ZKPoK for $\mathcal{L}_{GI}$ with $\epsilon_k \le 1/2$

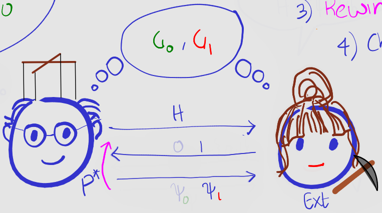
response to challenge 1   response to challenge 0

Proof (of PoK) Hint $\psi_1^{-1} \circ \psi_0 = \sigma^{-1} \circ \sigma \circ \pi = \pi$.

Compute $\pi : G_1 = \pi(G_0)$

$\sigma \leftarrow$ Perm. on $[1,n]$, $H := \sigma(G_1)$

$\psi := \begin{cases} \sigma \circ \pi & \text{if } b=0 \\ \sigma & \text{if } b=1 \end{cases}$



Extraction strategy $\mathrm{Ext}^{P^*}(G_0, G_1)$

1) Invoke P^* on (G_0, G_1) to obtain H
2) Challenge on 0 to get ψ_0
3) Rewind P^* to end of 1)
4) Challenge on 1 to get ψ_1

$\square$

Theorem 1

Π_{GI} *is a ZKPoK for* $\mathcal{L}_{GI}$ *with* $\epsilon_k \leq 1/2$

response to challenge 1 $\qquad$ response to challenge 0

Proof (of PoK) Hint $\psi_1^{-1} \circ \psi_0 = \sigma^{-1} \circ \sigma \circ \pi = \pi$.

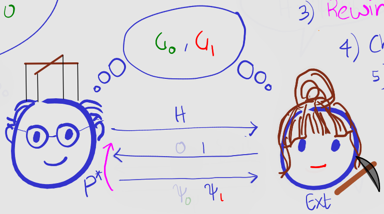
Compute π: $G_1 = \pi(G_0)$

$\sigma \leftarrow$ Perm. on $[1,n]$, $H := \sigma(G_1)$

$\psi := \begin{cases} \sigma \circ \pi & \text{if } b=0 \\ \sigma & \text{if } b=1 \end{cases}$

Extraction strategy $\mathrm{Ext}^{P^*}(G_0, G_1)$

1) Invoke P^* on (G_0, G_1) to obtain H
2) Challenge on 0 to get ψ_0
3) Rewind P^* to end of 1)
4) Challenge on 1 to get ψ_1
5) Output $\psi_1^{-1} \circ \psi_0$



$\square$

Theorem 1

Π_{GI} is a ZKPoK for $\mathcal{L}_{GI}$ with $\epsilon_k \leq 1/2$

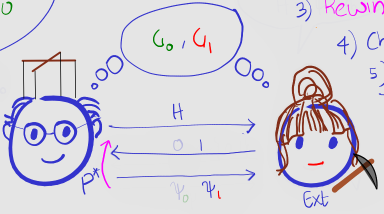
response to challenge 1 response to challenge 0

Proof (of PoK) Hint $\psi_1^{-1} \circ \psi_0 = \sigma^{-1} \circ \sigma \circ \pi = \pi.$

Compute π: $G_1 = \pi(G_0)$

$\sigma \leftarrow$ Perm. on $[1,n]$, $H := \sigma(G_1)$

$\psi := \begin{cases} \sigma \circ \pi & \text{if } b=0 \\ \sigma & \text{if } b=1 \end{cases}$



Extraction strategy $\mathrm{Ext}^{P^*}(G_0, G_1)$

1) Invoke P* on (G_0, G_1) to obtain H
2) Challenge on 0 to get ψ_0
3) Rewind P* to end of 1)
4) Challenge on 1 to get ψ_1
5) Output $\psi_1^{-1} \circ \psi_0$

V accepts 2) & 4)

$\Downarrow$

$H = \psi_0(G_0) = \psi_1(G_1)$

$\Downarrow$

$G_1 = \psi_1^{-1} \circ \psi_0(G_0)$ $\square$

How to Extract π from P^*?

Theorem 1

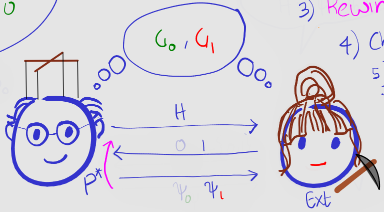
Π_{GI} is a ZKPoK for $\mathcal{L}_{GI}$ with $\epsilon_k \leq 1/2$

Proof (of PoK) Hint $\underbrace{\psi_1^{-1}}_{\text{response to challenge 1}} \circ \underbrace{\psi_0}_{\text{response to challenge 0}} = \sigma^{-1} \circ \sigma \circ \pi = \pi.$

Compute π: $G_1 = \pi(G_0)$

$\sigma \leftarrow$ Perm. on $[1,n]$, $H := \sigma(G_1)$

$\psi := \begin{cases} \sigma \circ \pi & \text{if } b = 0 \\ \sigma & \text{if } b = 1 \end{cases}$



Extraction strategy $\mathrm{Ext}^{P^*}(G_0, G_1)$

1) Invoke P^* on (G_0, G_1) to obtain H
2) Challenge on 0 to get ψ_0
3) Rewind P^* to end of 1)
4) Challenge on 1 to get ψ_1
5) Output $\psi_1^{-1} \circ \psi_0$

V accepts 2) & 4)

$\Downarrow$

$H = \psi_0(G_0) = \psi_1(G_1)$

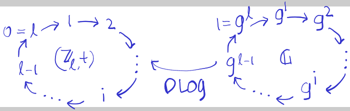
$\Downarrow$

$G_1 = \psi_1^{-1} \circ \psi_0(G_0)$ $\square$

Exercise 4

Analyse strategy for P^* with $\Pr[1 \leftarrow \langle P^*, V\rangle(G_0, G_1)] = 1/2 + 1/n$

- Recall:

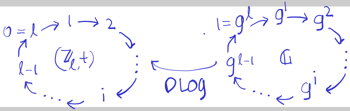


Defintion 3 (DLog problem in prime-order $\mathbb{G}$ w.r.to S)

- *Input:*
  1. $(\mathbb{G}, p, g)$ *sampled by a group sampler* $\mathsf{S}(1^n)$
  2. $h := g^a$ *for* $a \leftarrow \mathbb{Z}_p$
- *Solution:* a

ZKPoK for DLog: Schnorr's Protocol...

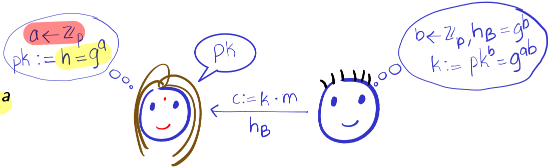
- Recall:



Defintion 3 (DLog problem in prime-order $\mathbb{G}$ w.r.to S)

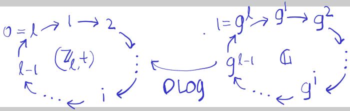
- *Input:*
  1. $(\mathbb{G}, p, g)$ *sampled by a group sampler* $\mathsf{S}(1^n)$
  2. $h := g^a$ *for* $a \leftarrow \mathbb{Z}_p$
- *Solution:* a

- ElGamal PKE:
  - Public key: $h := g^a$
  - Secret key: a



ZKPoK for DLog: Schnorr's Protocol...

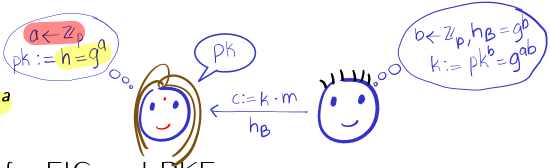
- Recall:



Defintion 3 (DLog problem in prime-order $\mathbb{G}$ w.r.to S)

- *Input:*
  1. $(\mathbb{G}, p, g)$ *sampled by a group sampler* $\mathsf{S}(1^n)$
  2. $h := g^a$ *for* $a \leftarrow \mathbb{Z}_p$
- *Solution:* a

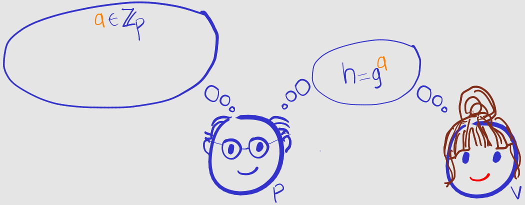
- ElGamal PKE:
  - Public key: $h := g^a$
  - Secret key: a



- Identification protocol for ElGamal PKE:
  - ZKP: owner of h proves possession of a without revealing it
  - PoK: without knowledge of a, verifier cannot be convinced

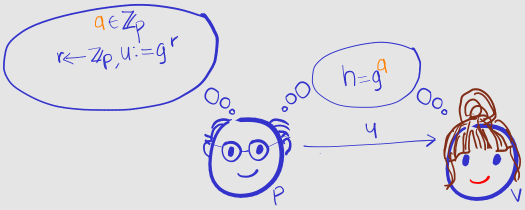
ZKPoK for DLog: Schnorr's Protocol...

Protocol 2 (Π_{DLog}: Schnorr's protocol)

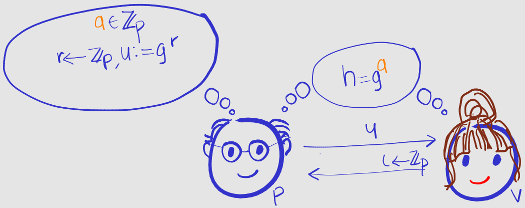


ZKPoK for DLog: Schnorr's Protocol...

Protocol 2 (Π_{DLog}: Schnorr's protocol)

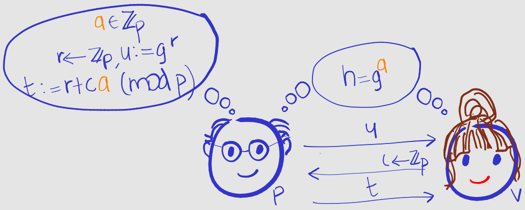


Protocol 2 (Π_{DLog}: Schnorr's protocol)



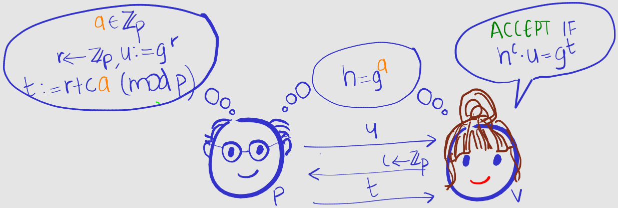
ZKPoK for DLog: Schnorr's Protocol...

Protocol 2 (Π_{DLog}: Schnorr's protocol)



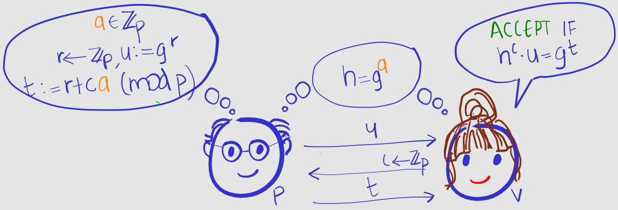
ZKPoK for DLog: Schnorr's Protocol...

Protocol 2 (Π_{DLog}: Schnorr's protocol)



ZKPoK for DLog: Schnorr's Protocol...

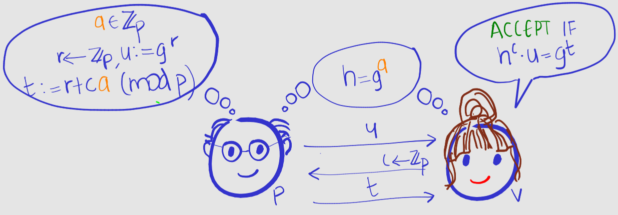
Protocol 2 (Π_{DLog}: Schnorr's protocol)



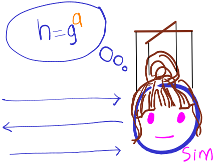
- Completeness: $h^c \cdot u = (g^a)^c \cdot g^r = g^{ac+r} = g^t$ (by group axioms)

ZKPoK for DLog: Schnorr's Protocol...

Protocol 2 (Π_{DLog}: Schnorr's protocol)

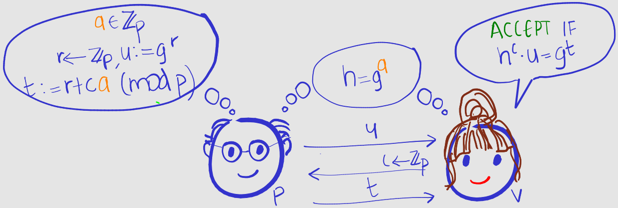


- Completeness: $h^c \cdot u = (g^a)^c \cdot g^r = g^{ac+r} = g^t$ (by group axioms)
- Honest-verifier ZK: out of order sampling, again

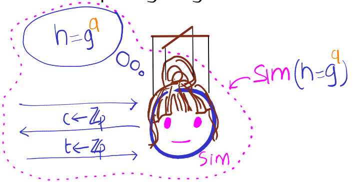


ZKPoK for DLog: Schnorr's Protocol...

Protocol 2 (Π_{DLog}: Schnorr's protocol)

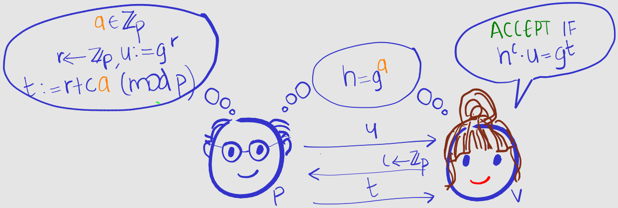


- Completeness: $h^c \cdot u = (g^a)^c \cdot g^r = g^{ac+r} = g^t$ (by group axioms)
- Honest-verifier ZK: out of order sampling, again



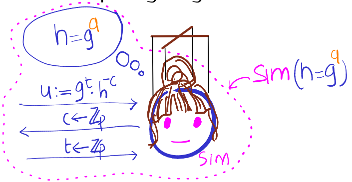
ZKPoK for DLog: Schnorr's Protocol...

Protocol 2 (Π_{DLog}: Schnorr's protocol)



- Completeness: $h^c \cdot u = (g^a)^c \cdot g^r = g^{ac+r} = g^t$ (by group axioms)
- Honest-verifier ZK: out of order sampling, again

Distributed identically to view since $g^t \cdot h^{-c} = g^{t-ac}$ is random

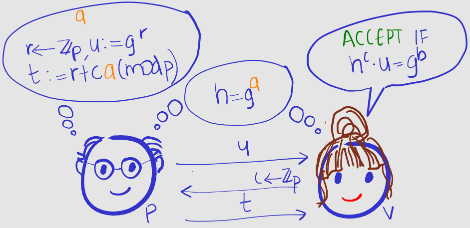


How to Extract a from P*?

Theorem 2

Π_{DLog} *is a PoK for* $\mathcal{L}_{\mathrm{DLog}}$ *with* $\epsilon_k \leq 1/p$

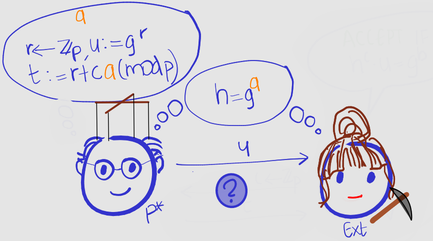
Proof (of PoK) Hint Obtain two eqns of form $t = r + ca \bmod p$.



Theorem 2

Π_{DLog} *is a PoK for* $\mathcal{L}_{\mathrm{DLog}}$ *with* $\epsilon_k \leq 1/p$

Proof (of PoK) Hint Obtain two eqns of form $t = r + ca \bmod p$.



Extraction strategy $\mathrm{Ext}^{P^*}(h)$

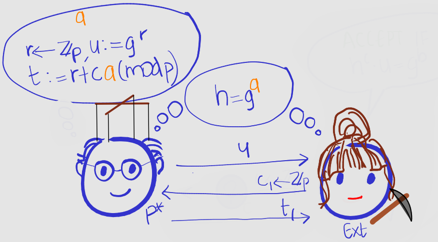
1) Invoke P^* on h to obtain u

How to Extract a from P^*?

Theorem 2

Π_{DLog} is a PoK for $\mathcal{L}_{\text{DLog}}$ with $\epsilon_k \le 1/p$

Proof (of PoK) Hint Obtain two eqns of form $t = r + ca \bmod p$.



Extraction strategy $\mathrm{Ext}^{P^*}(h)$

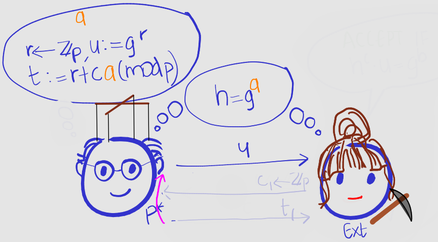
1) Invoke P^* on h to obtain u
2) Challenge on $c_1 \leftarrow \mathbb{Z}_p$ to get t_1

Theorem 2

Π_{DLog} *is a PoK for* $\mathcal{L}_{\mathrm{DLog}}$ *with* $\epsilon_k \leq 1/p$

Proof (of PoK)

Hint Obtain two eqns of form $t = r + ca \bmod p$.



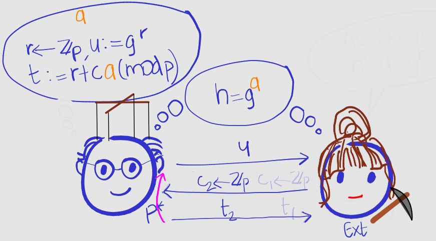
Extraction strategy $\mathrm{Ext}^{P^*}(h)$

1) Invoke P^* on h to obtain u
2) Challenge on $c_1 \leftarrow \mathbb{Z}_p$ to get t_1
3) Rewind P to 1)

Theorem 2

Π_{DLog} *is a PoK for* $\mathcal{L}_{\mathrm{DLog}}$ *with* $\epsilon_k \leq 1/p$

Proof (of PoK) Hint Obtain two eqns of form $t = r + ca \bmod p$.



Extraction strategy $\mathrm{Ext}^{P^*}(h)$

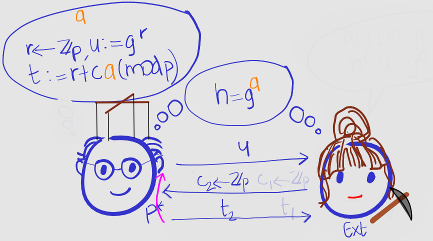
1) Invoke P^* on h to obtain u
2) Challenge on $c_1 \leftarrow \mathbb{Z}_p$ to get t_1
3) Rewind P to 1)
4) Challenge on $c_2 \leftarrow \mathbb{Z}_p$ to get t_2

How to Extract a from P*?

Theorem 2

Π_{DLog} is a PoK for $\mathcal{L}_{\mathrm{DLog}}$ with $\epsilon_k \leq 1/p$

Proof (of PoK) Hint Obtain two eqns of form $t = r + ca \bmod p$.



Extraction strategy $\mathrm{Ext}^{P^*}(h)$

1) Invoke P^* on h to obtain u
2) Challenge on $c_1 \leftarrow \mathbb{Z}_p$ to get t_1
3) Rewind P to 1)
4) Challenge on $c_2 \leftarrow \mathbb{Z}_p$ to get t_2
5) Output $t_1 - t_2 / c_1 - c_2$

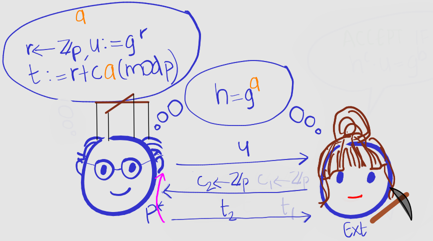
V accepts both executions $\Rightarrow g^{t_1} = u \cdot h^{c_1}$ & $g^{t_2} = u \cdot h^{c_2}$

$\Rightarrow g^{t_1 - t_2} = h^{c_1 - c_2} \Rightarrow a = \dfrac{t_1 - t_2}{c_1 - c_2}$

Theorem 2

Π_{DLog} *is a PoK for* $\mathcal{L}_{\mathrm{DLog}}$ *with* $\epsilon_k \leq 1/p$

Proof (of PoK)   Hint Obtain two eqns of form $t = r + ca \bmod p$.



Extraction strategy $\mathrm{Ext}^{P^*}(h)$

1) Invoke P^* on h to obtain u
2) Challenge on $c_1 \leftarrow \mathbb{Z}_p$ to get t_1
3) Rewind P to 1)
4) Challenge on $c_2 \leftarrow \mathbb{Z}_p$ to get t_2
5) Output $t_1 - t_2 / c_1 - c_2$

V accepts both executions $\Rightarrow g^{t_1} = u \cdot h^{c_1}$ & $g^{t_2} = u \cdot h^{c_2}$

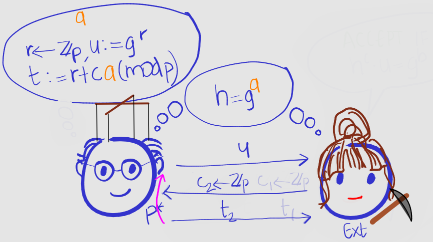
Extraction error $1/p$ $\Leftarrow$ Fails if $c_1 = c_2$ $\not\Rightarrow$ $g^{t_1 - t_2} = h^{c_1 - c_2} \Rightarrow a = \frac{t_1 - t_2}{c_1 - c_2}$

How to Extract a from P^*?

Theorem 2

Π_{DLog} is a PoK for $\mathcal{L}_{\mathrm{DLog}}$ with $\epsilon_k \leq 1/p$

Proof (of PoK) Hint Obtain two eqns of form $t = r + ca \bmod p$.



Extraction strategy $\mathrm{Ext}^{P^*}(h)$

1) Invoke P^* on h to obtain u
2) Challenge on $c_1 \leftarrow \mathbb{Z}_p$ to get t_1
3) Rewind P to 1)
4) Challenge on $c_2 \leftarrow \mathbb{Z}_p$ to get t_2
5) Output $t_1 - t_2 / c_1 - c_2$

V accepts both executions $\Rightarrow g^{t_1} = u \cdot h^{c_1}$ & $g^{t_2} = u \cdot h^{c_2}$

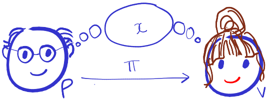
Extraction error $1/p$ $\Leftarrow$ Fails iff $c_1 = c_2$ $\not\Rightarrow$ $g^{t_1 - t_2} = h^{c_1 - c_2} \Rightarrow a = \frac{t_1 - t_2}{c_1 - c_2}$ $\square$

Exercise 5 ("Rewinding lemma")

Analyse strategy for P^* with $\Pr[1 \leftarrow \langle P^*, V \rangle(h)] = 1/p + 1/n$

Plan for Today's Lecture

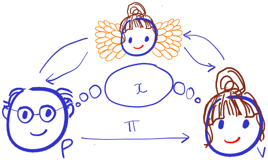
Non-Interactive Zero-Knowledge (NIZK)



Exercise 6 (Exercise 5, Lecture 14)

If $\mathcal{L}$ has a non-interactive ZKP $\Pi = (\mathsf{P}, \mathsf{V})$ then $\mathcal{L} \in \mathsf{BPP}$

Non-Interactive Zero-Knowledge (NIZK)



Exercise 6 (Exercise 5, Lecture 14)

If $\mathcal{L}$ has a non-interactive ZKP $\Pi = (\mathsf{P}, \mathsf{V})$ then $\mathcal{L} \in \mathsf{BPP}$

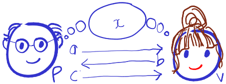
- One way around: NIZK in random oracle model (ROM)
  - ROM: All parties **P**, **V**, **Sim** and **Ext** can access to random function H in the sky
  - **Sim** and **Ext** can program H

NIZK in ROM via Fiat-Shamir Transform

- Public-coin interactive protocol $\xrightarrow{ROM}$ non-interactive protocol
  - Public coin: verifier's messages are just random coins
    - E.g., Π_{DLog} (Schnorr's protocol) and Π_{GI}

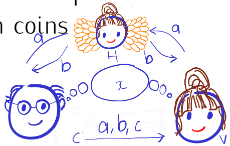
NIZK in ROM via Fiat-Shamir Transform

- Public-coin interactive protocol $\xrightarrow{\textit{ROM}}$ non-interactive protocol
  - Public coin: verifier's messages are just random coins
    - E.g., Π_{DLog} (Schnorr's protocol) and Π_{GI}
  - Idea: "replace" verifier with random oracle H



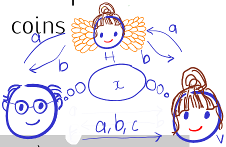
NIZK in ROM via Fiat-Shamir Transform

- Public-coin interactive protocol $\xrightarrow{ROM}$ non-interactive protocol
  - Public coin: verifier's messages are just random coins
    - E.g., Π_{DLog} (Schnorr's protocol) and Π_{GI}
  - Idea: "replace" verifier with random oracle H

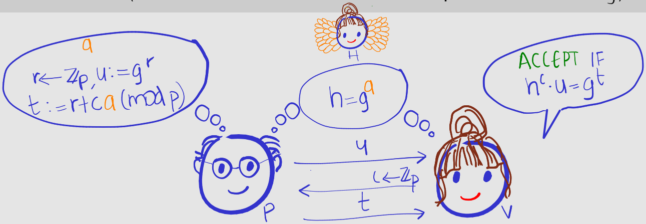


NIZK in ROM via Fiat-Shamir Transform

- Public-coin interactive protocol $\xrightarrow{ROM}$ non-interactive protocol
  - Public coin: verifier's messages are just random coins
    - E.g., Π_{DLog} (Schnorr's protocol) and Π_{GI}
  - Idea: "replace" verifier with random oracle H

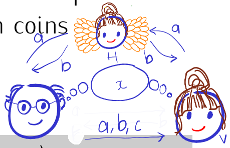


Construction 2 (Schnorr's non-interactive protocol N_{DLog})

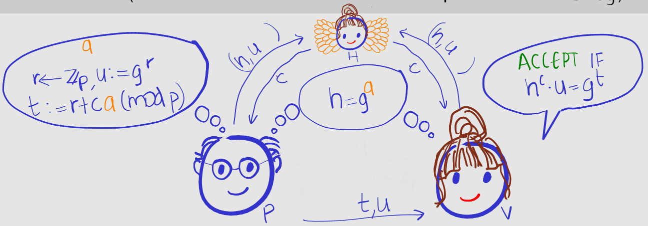


NIZK in ROM via Fiat-Shamir Transform

- Public-coin interactive protocol $\xrightarrow{ROM}$ non-interactive protocol
  - Public coin: verifier's messages are just random coins
    - E.g., Π_{DLog} (Schnorr's protocol) and Π_{GI}
  - Idea: "replace" verifier with random oracle H

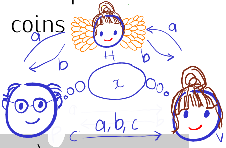


Construction 2 (Schnorr's non-interactive protocol N_{DLog})

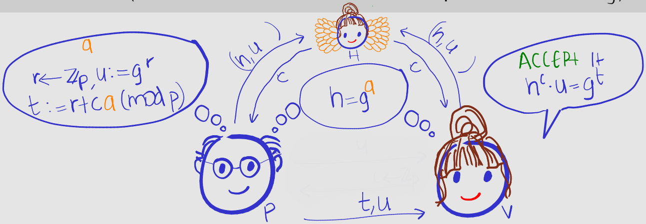


NIZK in ROM via Fiat-Shamir Transform

- Public-coin interactive protocol $\xrightarrow{ROM}$ non-interactive protocol
  - Public coin: verifier's messages are just random coins
    - E.g., Π_{DLog} (Schnorr's protocol) and Π_{GI}
  - Idea: "replace" verifier with random oracle H



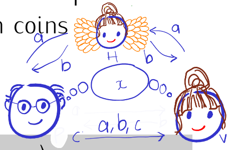
Construction 2 (Schnorr's non-interactive protocol N_{DLog})



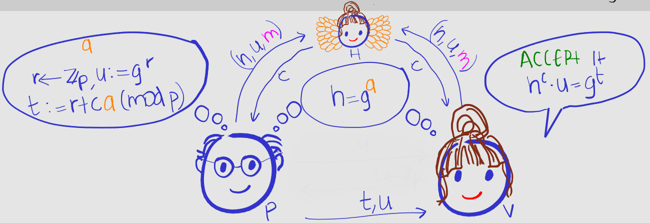
- N_{DLog} can be shown to be NIZK(PoK) in ROM

NIZK in ROM via Fiat-Shamir Transform

- Public-coin interactive protocol $\xrightarrow{\textit{ROM}}$ non-interactive protocol
  - Public coin: verifier's messages are just random coins
    - E.g., Π_{DLog} (Schnorr's protocol) and Π_{GI}
  - Idea: "replace" verifier with random oracle H

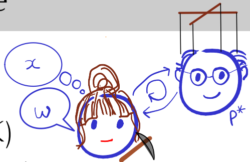


Construction 2 (Schnorr's non-interactive protocol N_{DLog})



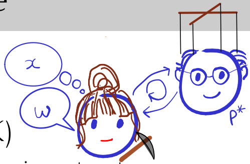
- N_{DLog} can be shown to be NIZK(PoK) in ROM
- Tweak N_{DLog} to get signature: include message $\boldsymbol{m}$ in hash
  - Closely-related to DSA

To Recap Today's Lecture

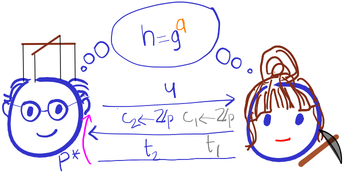


- Zero-knowledge proofs of knowledge (ZKPoK)
  - Quantified what "knowing something" means via extractors

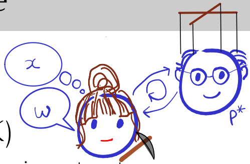
To Recap Today's Lecture



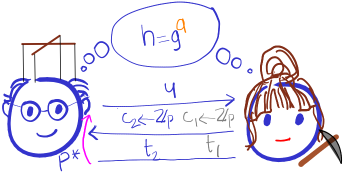
- Zero-knowledge proofs of knowledge (ZKPoK)
  - Quantified what "knowing something" means via extractors
- Examples
  1. ZKPoK for Graph Isomorphism (GI)
  2. ZKPoK for the discrete-log problem: Schnorr's protocol
  3. Key tool: rewinding the prover



To Recap Today's Lecture



- Zero-knowledge proofs of knowledge (ZKPoK)
  - Quantified what "knowing something" means via extractors
- Examples
  1. ZKPoK for Graph Isomorphism (GI)
  2. ZKPoK for the discrete-log problem: Schnorr's protocol
  3. Key tool: rewinding the prover



- Fiat-Shamir Transform

  - NIZK in random oracle model (ROM)
  - Digital signature from DLog in ROM

Next Lecture

- Task 6: private computation of two-party functions
- Security: extending the simulation paradigm
- Perfectly-secure private computation of linear functions
- Impossibility of perfect security for general functions

References

1 [Gol01, §4.7] for details of today's lecture
2 [GMR89] for definitional and philosophical discussion on ZK
3 NIZK was introduced [BFM88]
4 Fiat-Shamir Transform was introduced in [FS87]
5 The constructions of commitment scheme from OWP and PRG is from [GMW91] and [Nao90]

Manuel Blum, Paul Feldman, and Silvio Micali.
Non-interactive zero-knowledge and its applications (extended abstract).
In *20th ACM STOC*, pages 103–112. ACM Press, May 1988.

Amos Fiat and Adi Shamir.
How to prove yourself: Practical solutions to identification and signature problems.
In Andrew M. Odlyzko, editor, *CRYPTO'86*, volume 263 of *LNCS*, pages 186–194. Springer, Heidelberg, August 1987.

Shafi Goldwasser, Silvio Micali, and Charles Rackoff.
The knowledge complexity of interactive proof systems.
SIAM J. Comput., 18(1):186–208, 1989.

Oded Goldreich, Silvio Micali, and Avi Wigderson.
Proofs that yield nothing but their validity for all languages in NP have zero-knowledge proof systems.
J. ACM, 38(3):691–729, 1991.

Oded Goldreich.
The Foundations of Cryptography - Volume 1: Basic Techniques.
Cambridge University Press, 2001.

Moni Naor.
Bit commitment using pseudo-randomness.
In Gilles Brassard, editor, *CRYPTO'89*, volume 435 of *LNCS*, pages 128–136. Springer, Heidelberg, August 1990.