Advanced Crypto 2024
Zero-Knowledge Proofs

Léo Colisson Palais

leo.colisson-palais@univ-grenoble-alpes.fr

`https://leo.colisson.me/teaching.html`

Zero-knowledge

Zero-Knowledge (ZK) Proof = prove a statement without revealing anything beyond the fact that the statement is true.

Applications ZK

Many applications:

- **Authentication**: “I know a secret x such that SHA3$(x) = y$”
- **Privacy-preserving blockchain**: “I can prove that this transaction is valid without revealing the sender, receiver, nor the amount of the transaction” (ZCash, see also smart contracts)
- **Multi-party computing**: “This circuit is an honesty-prepared garbled circuit, but I won’t reveal the keys of the circuit”
- **Sensitive data**: Say that the hash of your DNA (or medical record. . .) is signed by a trusted authority. Then you can prove to any insurance that you do not have a given genetic disorder without revealing your full DNA. Also works to prove that your salary is greater/lower than XXX without revealing it etc (needed by banks, housing allowance. . .).

Classical Zero-Knowledge

Classical Zero-Knowledge

Classical Zero-Knowledge

1			
		4	
		3	
	2		

Classical Zero-Knowledge

Classical Zero-Knowledge

1			
		4	
		3	
	2		

Classical Zero-Knowledge

1	4	2	3
2	3	4	1
4	1	3	2
3	2	1	4

Classical Zero-Knowledge

Classical Zero-Knowledge

Classical Zero-Knowledge

Classical Zero-Knowledge

Classical Zero-Knowledge

Classical Zero-Knowledge

Classical Zero-Knowledge

Generalizable in a non-interactive way to NP problems.

Issues

Still many questions:

- Sudoku are nice, but what else?
- How to replace physical cards?
- Can we make it fully non-interactive?
- Can we make the verification, e.g., logarithmic time?

ZK proofs for NP

NP

Definition (NP reminder)

A language $\mathcal{L} \subseteq \{0,1\}^*$ is said to be in the **NP** (nondeterministic polynomial time) class if there exists an efficient (polynomial time) Turing machine V such that $x \in \mathcal{L}$ iff there exists a **witness** w_x such that $V(x, w_x) = 1$ (we may write $x\mathcal{R}w_x$).

I.e. a problem is in NP if it is **easy to verify** a solution.

NP

Examples of NP problems:

- The language of Sudoku (of arbitrary size) with a solution is in NP:

	2	6						
						1	7	
		3	1		6			
	6			5		8		3
		9	2	6	1	7		
5		4		8			6	
			8		4	3		
	4	8						
						9	4	

$\Rightarrow$

1	2	6	5	7	8	4	3	9
4	8	5	9	3	2	1	7	6
7	9	3	1	4	6	5	8	2
2	6	1	4	5	7	8	9	3
8	3	9	2	6	1	7	5	4
5	7	4	3	8	9	2	6	1
6	5	2	8	9	4	3	1	7
9	4	8	7	1	3	6	2	5
3	1	7	6	2	5	9	4	8

(easy to verify)

- 3-SAT
- Graph coloring
- Hamiltonian path

NP

Examples of NP problems:

- The language of Sudoku (of arbitrary size) with a solution is in NP:

	2	6						
						1	7	
		3	1		6			
	6			5		8		3
		9	2	6	1	7		
5		4		8			6	
			8		4	3		
	4	8						
						9	4	

⇒

1	2	6	5	7	8	4	3	9
4	8	5	9	3	2	1	7	6
7	9	3	1	4	6	5	8	2
2	6	1	4	5	7	8	9	3
8	3	9	2	6	1	7	5	4
5	7	4	3	8	9	2	6	1
6	5	2	8	9	4	3	1	7
9	4	8	7	1	3	6	2	5
3	1	7	6	2	5	9	4	8

(easy to verify)

- 3-SAT ⇒ **NP-complete**
- Graph coloring ⇒ **NP-complete**
- Hamiltonian path ⇒ **NP-complete**

NP

Definition (NP complete)

A language $\mathcal{L}$ is **NP complete** if given access to an oracle $\mathcal{O}(x) := x \in \mathcal{L}$, one can efficiently tell if $x' \in \mathcal{L}'$ for any NP language $\mathcal{L}'$ and word x'.

ZK for NP

Theorem (informal)

For any NP language $\mathcal{L}$, there exists a zero-knowledge protocol to prove that a given word x belongs to $\mathcal{L}$. Notably, no information on the witness w_x is leaked to the prover.

Proof strategy:

$\mathcal{L} \longrightarrow$ SAT $\longrightarrow$ Hamiltonian path $\longrightarrow$ ZK for Hamiltonian path

ZK for NP, step 1: $\mathcal{L}$ to SAT

Definition (SAT)

A SAT (Boolean satisfiability) instance is defined by a conjunction of **clauses**, where each clause is the disjunction of multiple **literals** (a boolean variable or the negation of a boolean variable).
A SAT instance is said to be **satisfiable** if there exists an assignment making the final formula true.

E.g.:

- $(a \lor b) \land (\neg b \lor c \lor d) \land (a \lor \neg d)$
- $(a \lor \neg b \lor \neg c) \land (a \lor b \lor c) \land (b \lor \neg c)$

ZK for NP, step 1: $\mathcal{L}$ to SAT

First step: reduce $\mathcal{L}$ to a SAT instance (possible: SAT is NP complete and $\mathcal{L}$ is in NP). How?

$\Rightarrow$ **Tseytin transformation:**

- x is public, so we can consider the boolean circuit of the function $f(w) := V(x, w)$
- Add a new variable for each wire in the circuit of f (need to add new variables to avoid exponential increase in the number of clauses)
- For each gate g in the circuit of f, add new clauses to constraint the variable of the output wire o to be such that $o = g(i_1, \ldots, i_n)$ where $i_1, \ldots, i_n$ are the variable of the input wires of g. How to find the clauses?

ZK for NP, step 1: $\mathcal{L}$ to SAT

How to find the clauses to constraint $o = g(i_1, \ldots, i_n)$?

1. Method 1:
    - Rewrite $o \Leftrightarrow g(i_1, \ldots, i_n)$ as a boolean formula ϕ involving only $\wedge$, $\vee$ and $\neg$, using the fact that $a \Rightarrow b$ iff $b \vee \neg a$.
    - Express $\neg\phi$ as a disjunctive normal form, using first the Morgan laws ($\neg(A \vee B) = (\neg A) \wedge (\neg B)$ and $\neg(A \wedge B) = (\neg A) \vee (\neg B)$) to "push down" the negations, then distributivity laws ($(A \vee B) \wedge C = (A \wedge C) \vee (B \wedge C)$) to "push down" the conjunction.
    - Compute again the negation of $\neg\phi$ to recover ϕ (since $\neg\neg\phi = \phi$) using Morgan laws and simplification of double negation to get the conjunctive normal form of ϕ

ZK for NP, step 1: $\mathcal{L}$ to SAT

E.g. for $c = a \wedge b$ (we denote $\neg a$ as $\overline{a}$, $\wedge$ as multiplication and $\vee$ as addition since distributivity is easier to see with this notation):

$$\phi = (c \Leftrightarrow ab) = (c \Rightarrow ab)(ab \Rightarrow c) = (ab + \overline{c})(c + \overline{ab})$$

$$\overline{\phi} = \overline{(ab + \overline{c})(c + \overline{ab})} = \overline{ab + \overline{c}} + \overline{c + \overline{ab}} = \overline{ab}\,\overline{\overline{c}} + \overline{c}\,\overline{\overline{ab}} = (\overline{a} + \overline{b})c + \overline{c}ab = \overline{a}c + \overline{b}c + \overline{c}ab$$

$$\phi = \overline{\overline{\phi}} = \overline{\overline{a}c + \overline{b}c + \overline{c}ab} = (\overline{\overline{a}c})(\overline{\overline{b}c})(\overline{\overline{c}ab}) = (\overline{\overline{a}} + \overline{c})(\overline{\overline{b}} + \overline{c})(\overline{\overline{c}} + \overline{a} + \overline{b}) = (a + \overline{c})(b + \overline{c})(c + \overline{a} + \overline{b})$$

Hence we add the clauses $(a \vee \neg c) \wedge (b \vee \neg c) \wedge (c \vee \neg a \vee \neg b)$

Similarly, for an OR gate: $(a \vee b \vee \overline{c}) \wedge (\overline{a} \vee c) \wedge (\overline{b} \vee c)$

ZK for NP, step 1: $\mathcal{L}$ to SAT

How to find the clauses to constraint $o = g(i_1, \dots, i_n)$?

② Method 2:

- Write the truth table of $o \Leftrightarrow g(i_1, \dots, i_n)$
- Remark that the expression is true only if we are **not** in each line where the truth table is wrong: this directly gives a CNF by putting one clause per such line, where the literals are the negation of the assignments of this line.

E.g. for $c = a \wedge b$:

a	b	c	Truth value	Clauses to add
0	0	0	1	
0	0	1	0	$a \vee b \vee \neg c$
0	1	0	1	
0	1	1	0	$a \vee \neg b \vee \neg c$
1	0	0	1	
1	0	1	0	$\neg a \vee b \vee \neg c$
1	1	0	0	$\neg a \vee \neg b \vee c$
1	1	1	1	

(maybe not optimal, see also Karnaugh map)

ZK for NP, step 2: SAT to Hamiltonian path

Issue with SAT: no good way to do ZK directly on SAT.

$\Rightarrow$ **Turn SAT to Hamiltonian path!**

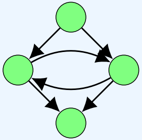
ZK for NP, step 2: SAT to Hamiltonian path

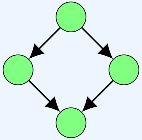
Definition (Hamiltonian path)

A **Hamiltonian path** in a directed graph $G = (V, E)$ is a path $P = (v_1, \dots, v_n)$ where $n = |V|$, i.e. a list of nodes such that for any i, $(v_i, v_{i+1}) \in E$, that visits all vertices in V exactly once (i.e. for all $i \neq i'$, $v_i \neq v_{i'}$). The decision version of the problem is to determine if there exists such a path.

?

Which graph(s) admit(s) an Hamiltonian path?





- Ⓐ None
- Ⓑ First one
- Ⓒ Second one
- Ⓓ Both

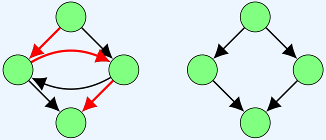
ZK for NP, step 2: SAT to Hamiltonian path

Definition (Hamiltonian path)

A **Hamiltonian path** in a directed graph $G = (V, E)$ is a path $P = (v_1, \ldots, v_n)$ where $n = |V|$, i.e. a list of nodes such that for any i, $(v_i, v_{i+1}) \in E$, that visits all vertices in V exactly once (i.e. for all $i \neq i'$, $v_i \neq v_{i'}$). The decision version of the problem is to determine if there exists such a path.

?

Which graph(s) admit(s) an Hamiltonian path?



- Ⓐ None
- Ⓑ First one ✓
- Ⓒ Second one
- Ⓓ Both

ZK for NP, step 2: SAT to Hamiltonian path

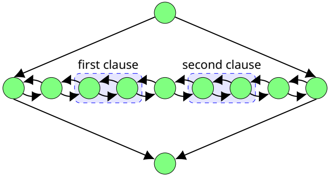
Theorem (Hamiltonian path is NP-complete)

For any SAT instance S, one can build in polynomial time a graph G_S that admits a Hamiltonian path iff S is satisfiable.

Instead of proving that a SAT instance is satisfiable, we can prove that a graph has a Hamiltonian path!

ZK for NP, step 2: SAT to Hamiltonian path

Step 1 construction: for each variable x, we create a diamond as follows, where the middle pattern repeats j times, where j is the number of clauses in S involving x:

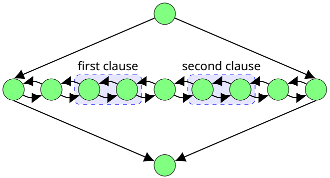


How many Hamiltonian paths can you find in this graph?

- Ⓐ 0
- Ⓑ 1
- Ⓒ 2
- Ⓓ 3 or more

ZK for NP, step 2: SAT to Hamiltonian path

Step 1 construction: for each variable x, we create a diamond as follows, where the middle pattern repeats j times, where j is the number of clauses in S involving x:

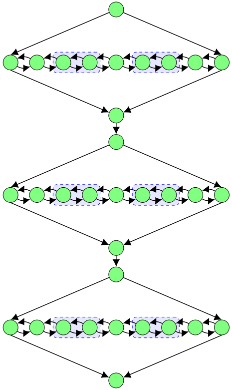


How many Hamiltonian paths can you find in this graph?

- Ⓐ 0
- Ⓑ 1
- Ⓒ 2 ✓
- Ⓓ 3 or more

ZK for NP, step 2: SAT to Hamiltonian path

Step 2 construction: we connect the diamonds as a chain (order does not matter)

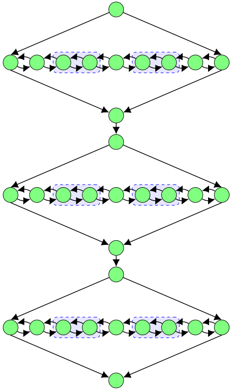


How many Hamiltonian paths can you find in this graph (suppose S has n variables)?

- Ⓐ 0
- Ⓑ n
- Ⓒ 2^n
- Ⓓ Other

ZK for NP, step 2: SAT to Hamiltonian path

Step 2 construction: we connect the diamonds as a chain (order does not matter)



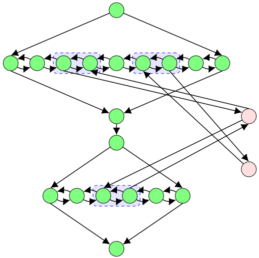
?

How many Hamiltonian paths can you find in this graph (suppose S has n variables)?

- Ⓐ 0
- Ⓑ n
- Ⓒ 2^n ✓
- Ⓓ Other

ZK for NP, step 2: SAT to Hamiltonian path

Last step construction: we add one node n_c per clause c, and for each variable x in c, we add two edges from this node to the two nodes a and b (a being to the left of b) of a free blue block in the diamond of x, where the direction is $a \to n_c \to b$ if the variable appears positively in the clause, and $b \to n_c \to a$ if the negation of x is in the clause.



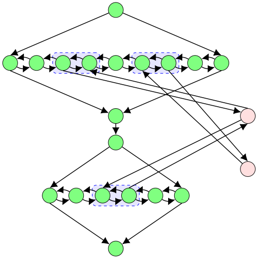
?

What is the formula encoded by the graph on the left?

- Ⓐ $(\neg a \vee \neg b) \wedge (a)$
- Ⓑ $(a \vee \neg b) \wedge (\neg a)$
- Ⓒ $(a \vee \neg b) \wedge (\neg a \vee \neg b)$
- Ⓓ $(a \vee \neg b) \wedge (\neg c)$
- Ⓔ $(a \wedge \neg b) \vee (\neg a)$

ZK for NP, step 2: SAT to Hamiltonian path

Last step construction: we add one node n_c per clause c, and for each variable x in c, we add two edges from this node to the two nodes a and b (a being to the left of b) of a free blue block in the diamond of x, where the direction is $a \to n_c \to b$ if the variable appears positively in the clause, and $b \to n_c \to a$ if the negation of x is in the clause.

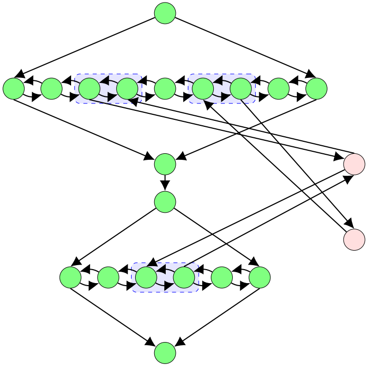


?

What is the formula encoded by the graph on the left?

- Ⓐ $(\neg a \lor \neg b) \land (a)$
- Ⓑ $(a \lor \neg b) \land (\neg a)$ ✓
- Ⓒ $(a \lor \neg b) \land (\neg a \lor \neg b)$
- Ⓓ $(a \lor \neg b) \land (\neg c)$
- Ⓔ $(a \land \neg b) \lor (\neg a)$

ZK for NP, step 2: SAT to Hamiltonian path



Claim

The resulting graph admits a Hamiltonian path iff S is satisfiable.

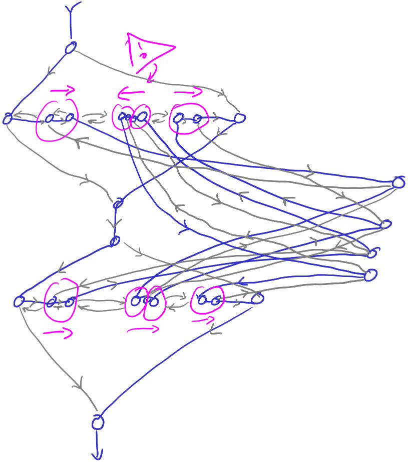
Proof skech:
$\Leftarrow$: quite easy
$\Rightarrow$: bit more technical: we must prove that all Hamiltonian paths have a "normal" form, i.e.:

- it visits the variables in order,
- all nodes of the variable are visited in a "Z" shape (two possible directions = interpret as true or false),
- if we leave one node in a blue box to a clause node, the next step is on the other node in the same blue box,

(Demonstration on board)

ZK for NP, step 2: SAT to Hamiltonian path

NB: **important to keep the "separation nodes"** between the blue boxes! Otherwise possible to find weird paths visiting the nodes in different directions:



Commitment

What is the cryptographic equivalent of the “cards” used in the sudoku game?

Commitment

What is the cryptographic equivalent of the “cards” used in the sudoku game?

$\Rightarrow$ **commitments!**

Commitments

Definition (Commitment)

Let $\mathsf{Commit}(x, r), \mathsf{Open}(c, x, r)$ be two probabilistic algorithms (implicitly depending on a security parameter λ). They are said to be a commitment if it is:

- **Correct**: for any x and r, $\mathsf{Open}(\mathsf{Commit}(x, r), x, r) = \top$
- **Hiding**: "Commitments reveal no info on x"
  For any x, x', and adversary $\mathcal{A}$,
  $$\left| \Pr_{\substack{r \xleftarrow{\$} \{0,1\}^\lambda \\ c \leftarrow \mathsf{Commit}(x,r)}} [\,\mathcal{A}(c) = 1\,] - \Pr_{\substack{r \xleftarrow{\$} \{0,1\}^\lambda \\ c \leftarrow \mathsf{Commit}(x',r)}} [\,\mathcal{A}(c) = 1\,] \right| \leq \mathsf{negl}(\lambda)$$
- **Binding**: "Hard to open to two different values"
  For any adversary $\mathcal{A}$,
  $$\Pr_{(c,x,r,x',r') \leftarrow \mathcal{A}(1^\lambda)} [\,\mathsf{Open}(c, x, r) = \mathsf{Open}(c, x', r') = \top \wedge x \neq x'\,] \leq \mathsf{negl}(\lambda)$$

Commitments

How to obtain commitments?

- **Method 1: Random Oracle model**:
  Commit$(x, r) = H(r \| x)$, Open$(c, x, r) = (c \overset{?}{=} H(r \| x))$
- **Method 2: One-way permutations (bit commitment)**:
  - $f : \{0,1\}^* \to \{0,1\}^*$
  - $p : \{0,1\}^* \to \{0,1\}$ hard-core predicate
    (hard to guess $p(x)$ given $f(x)$, exists thanks to the Goldreich-Levin theorem)
  - $x \in \{0,1\}$

  Commit$(x, r) = (f(r), p(r) \oplus x)$, Open$((y, b), x, r) = ((y, b) \overset{?}{=} (f(r), p(r) \oplus x))$
  (permutation needed for (statistical) binding, otherwise we need something like collision resistance)

Commitments

How to obtain commitments?

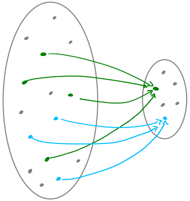
- **Method 3: PRG** (exists from one-way functions)
  - $G : \{0,1\}^* \to \{0,1\}^*$, such that $\forall s, |G(s)| = 3|s|$
  - We assume that the receiver sent a random $r_0 \overset{\$}{\leftarrow} \{0,1\}^{3n}$ before the commit phase
  - $x \in \{0,1\}$

Commit$(x,r) = G(r) \oplus (xr_0)$, Open$(c,x,r) = (G(r) \oplus (xr_0) \overset{?}{=} c)$

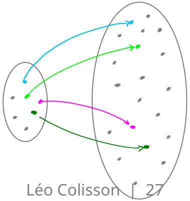
Commitments

There exists **no statistically hiding and statistically binding** commitment scheme, but there exists both:

- statistical hiding + computational binding (many-to-one hash function)



- computational hiding + statistical binding (injective hash function)



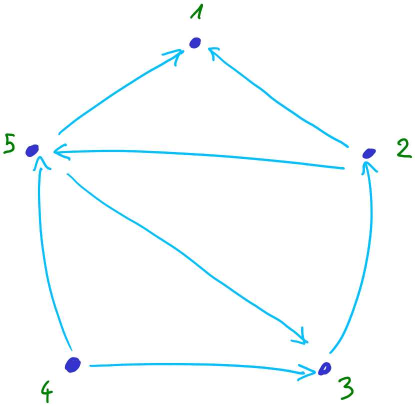
ZK for NP, step 3: ZK for Hamiltonian path

Claim (ZK for Hamiltonian path, informal)

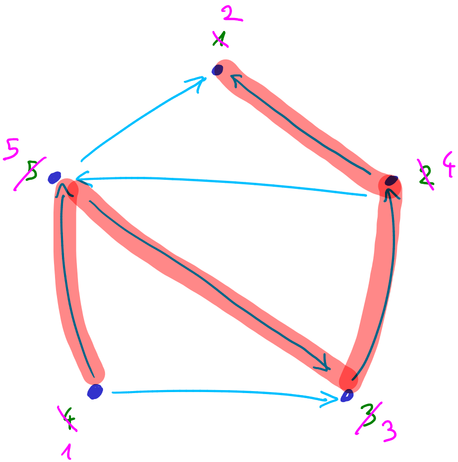
For any graph G, it is possible to prove that we know a Hamiltonian path for G without revealing anything about this path.

? Can you find how, based on the Sudoku example?

ZK proof of the Hamiltonian path (informal)



ZK proof of the Hamiltonian path (informal)

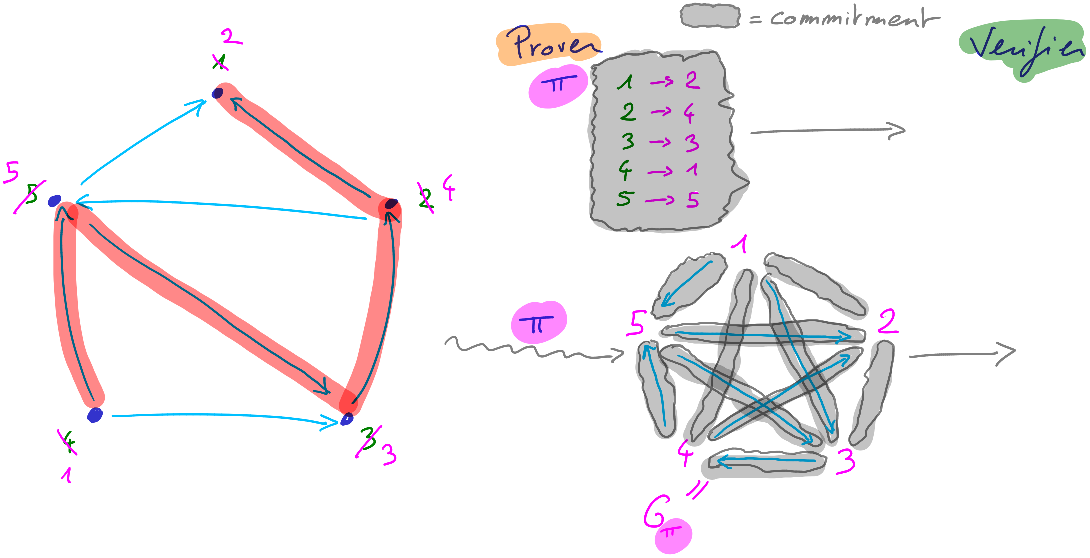


Prover

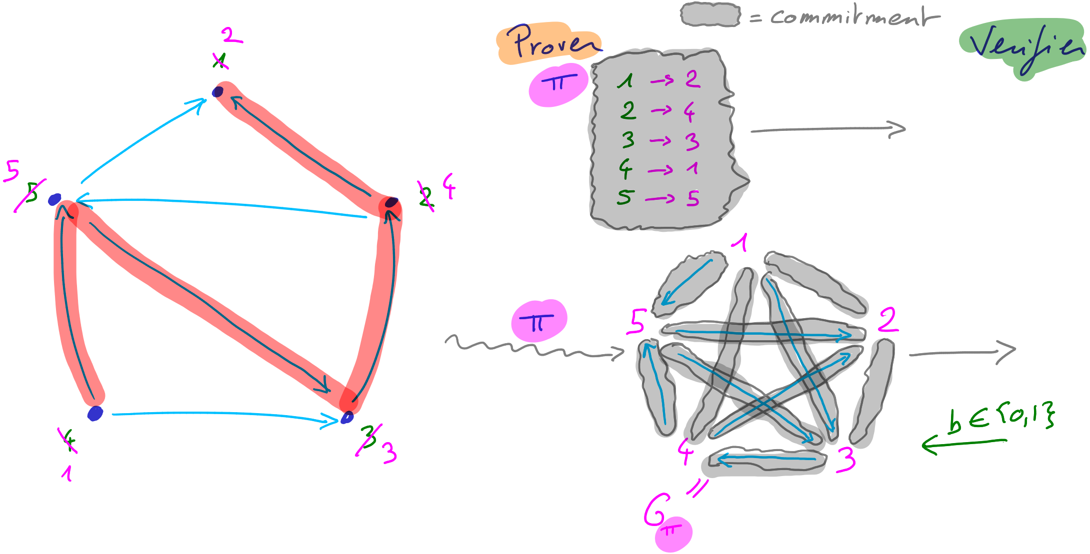
π

1	→	2
2	→	4
3	→	3
4	→	1
5	→	5

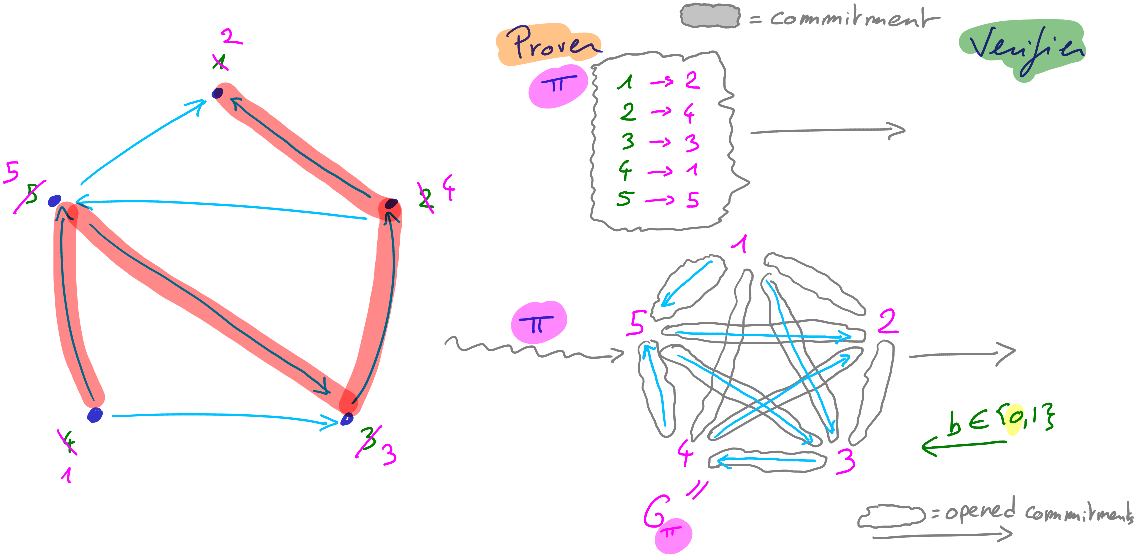
ZK proof of the Hamiltonian path (informal)



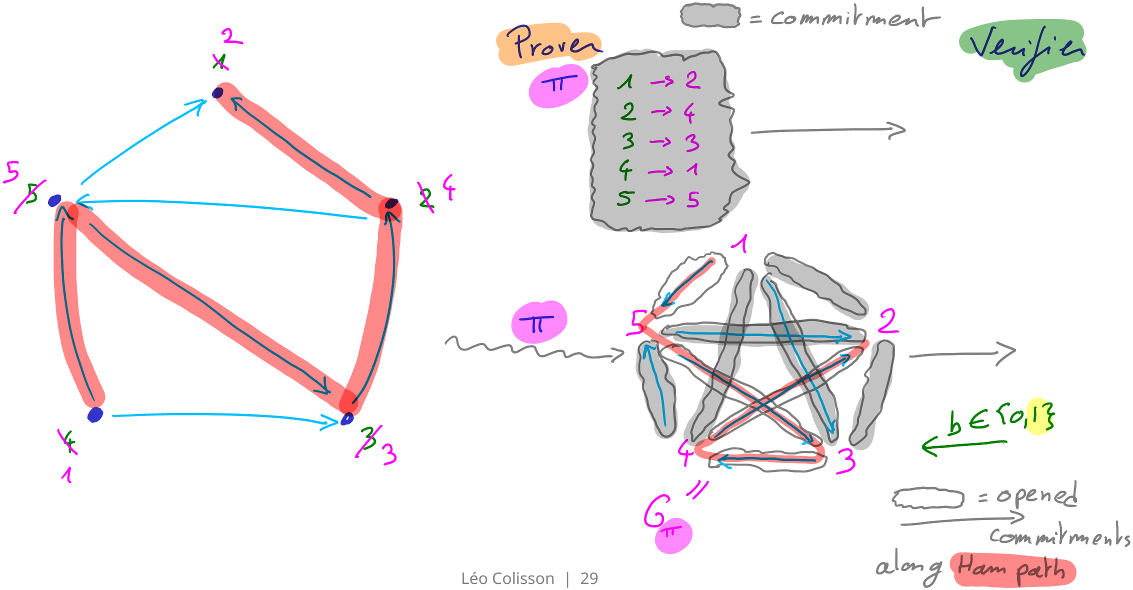
ZK proof of the Hamiltonian path (informal)



ZK proof of the Hamiltonian path (informal)



ZK proof of the Hamiltonian path (informal)



ZK proof of the Hamiltonian path

ZK-Ham protocol

Protocol $(V(G), P(G, V_H))$, where $G = (V_G, E_G)$ is a directed graph, and $V_H = (v_1, \dots, v_n)$ is a Hamiltonian path = repeat the following poly(λ) times:

1. The prover P picks a random permutation π on $\{1, \dots, n\}$, let M be the π-permuted adjacency matrix of G, i.e. $M_{(\pi(i),\pi(j))} = 1$ iff $(i, j) \in E$. P sends a commitment of each entry in M to V.
2. The verifier V picks a random bit $b \xleftarrow{\$} \{0, 1\}$ and sends it to P.
3.
   - if $b = 0$, P reveals π and opens all commitments. V verifies that they correspond to the π-permuted adjacency matrix of G.
   - if $b = 1$, P sends $(\pi(v_1), \dots, \pi(v_n))$ and only opens the commitments of M of the edges along this path. V verifies if all opening are valid and open to 1, and if all vertices are different.

(Note: instead of an adjacency matrix, we can also send the list of edges, but we need to shuffle them so that their position in the list reveals no information on the graph)

Formally defining ZK proofs

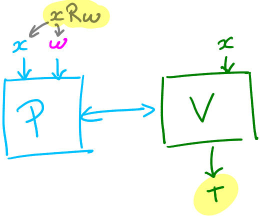
3 Goals

Formally defining ZK proofs

3 Goals

① **Correctness**

"Everyone honest
$\Rightarrow$ V accepts"

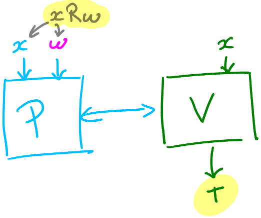


Formally defining ZK proofs

3 Goals

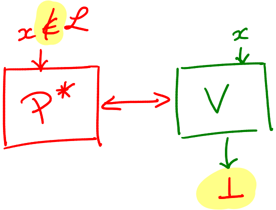
① Correctness

"Everyone honest
=> V accepts"



② Soundness

"Malicious Prover P^*
cannot convince V if $x \notin \mathcal{L}$"

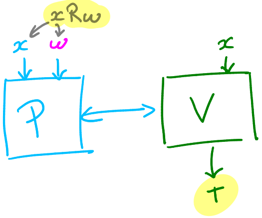


Formally defining ZK proofs

3 Goals

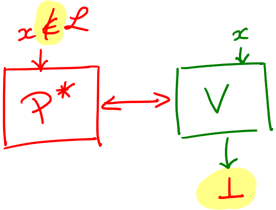
① Correctness

"Everyone honest $\Rightarrow$ V accepts"



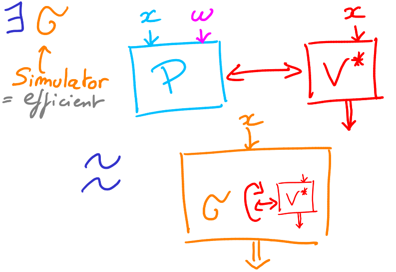
② Soundness

"Malicious Prover P^* cannot convince V if $x \notin \mathcal{L}$"



③ Zero-knowledge

"Malicious Verifier V^* learns nothing about the witness w":



Formally defining ZK proofs

Definition (ZK proof system)

A ZK proof system for a language $\mathcal{L}$ in NP, such that $x \in \mathcal{L} \Leftrightarrow \exists w, x\mathcal{R}w$, is defined by a protocol between an efficient verifier $V(x)$ (outputting either accept or reject) and a prover $P(x, w)$, such that the protocol is:

- **Correct**: if $\exists w, x\mathcal{R}w$, $V(x)$ always accepts after interacting with $P(x, w)$
- **Soundness**: if $x \notin \mathcal{L}$, $V(x)$ accepts with negligible probability after interacting with any malicious prover $P^*(x, w)$ (if P^* is restricted to be efficient, we often refer to this as an argument system instead of a proof system, but we will not make much distinction here)
- **Zero-Knowledge**: For any malicious efficient (if it is not restricted to be efficient, we refer to it as statistical ZK) verifier $V^*(x)$, there exists an efficient probabilistic algorithm S^* (that can depend arbitrarily on V^*), called **"simulator"**, such that for any $x\mathcal{R}w$, the output of $V^*(x)$ interacting with $P(x, w)$ is computationally indistinguishable from $S^*(x)$.

Formally defining ZK proofs

? Show that if a protocol is ZK for an NP-complete problem, and if $P \neq NP$, then V^* is, in particular, unable to recover the witness.

Formally defining ZK proofs

? Show that if a protocol is ZK for an NP-complete problem, and if $P \neq NP$, then V^* is, in particular, unable to recover the witness.

Idea: if $V^*(x)$ can output the witness w after interacting with $P(x, w)$, then $S^*(x)$ is also a witness (otherwise it is easy to distinguish both distributions by simply verifying if it is a valid witness). But $S^*(x)$ is efficient, which is absurd as the problem is NP complete, unless $P = NP$.

ZK proof of the Hamiltonian path

Theorem (ZK-Ham

The ZK protocol for the Hamiltonian path is zero-knowledge.

Proof: for the ZK part the proof needs to “rewind” the prover, details on white board and next slides. Details can also be found, e.g., in `https://courses.csail.mit.edu/6.857/2018/files/L22-ZK-Boaz.pdf`.

ZK proof of the Hamiltonian path

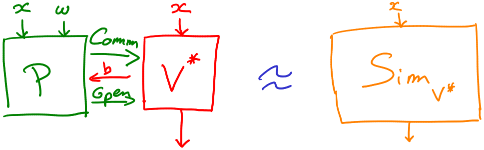
First focus one round ↖

Proof of the zero-knowledge:

ZK proof of the Hamiltonian path

First focus one round

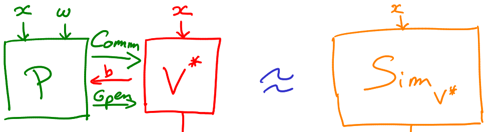
Proof of the zero-knowledge: Let V^* be a malicious verifier, and $x R w$. Want to show that $\exists$ poly Sim_{V^*} s.t.



ZK proof of the Hamiltonian path

First focus one round

Proof of the zero-knowledge: Let V^* be a malicious verifier, and $x R w$. Want to show that $\exists$ poly Sim_{V^*} s.t.



Guess challenge of V^*

Claim: we can define:

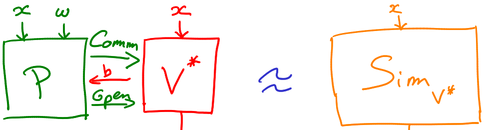
$\text{Sim}_{V^*}(x)$:

$\tilde{b} \leftarrow \{0,1\}$

ZK proof of the Hamiltonian path

First focus one round ↑

Proof of the zero-knowledge: Let V^* be a malicious verifier, and $x R w$. Want to show that $\exists$ poly Sim_{V^*} s.t.



Guess challenge of V^*

Claim: we can define:

$\mathrm{Sim}_{V^*}(x)$:

① $\tilde{b} \leftarrow \{0,1\}$

if $\tilde{b} = 0$:

Pick random π, send $(\mathrm{Comm}(\pi), \mathrm{Comm}(G_\pi))$ to simulated V^*, obtain challenge b.

Commit all entries one by one

Permuted adjacency matrix of G

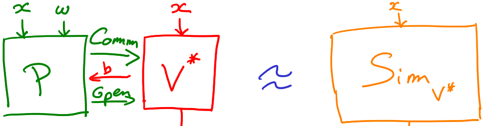
If $b = \tilde{b}$: Open all commitments to V^*

else: Goto ① (= rewind)

ZK proof of the Hamiltonian path

First focus one round

Proof of the zero-knowledge: Let V^* be a malicious verifier, and $x R w$. Want to show that $\exists$ poly Sim_{V^*} s.t.



Claim: we can define:

Guess challenge of V^*

Commit all entries one by one

Permuted adjacency matrix of G

Commit entries 1 by 1.

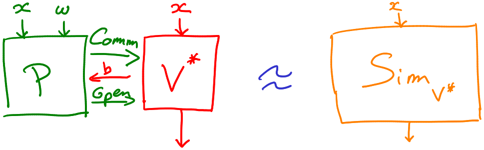
```
Sim_{V*}(x):
  ① $\tilde{b} \leftarrow \{0,1\}$
     if $\tilde{b} = 0$:
        Pick random $\pi$, send
           $(\mathrm{Comm}(\pi), \mathrm{Comm}(G_\pi))$
        to simulated $V^*$, obtain
        challenge $b$.
        If $b = \tilde{b}$: Open all commitments to $V^*$
        else: Goto ① (= rewind)
     if $\tilde{b} = 1$:
        Pick random $\pi$, send
           $(\mathrm{Comm}(\pi), \mathrm{Comm}\left(\begin{pmatrix} 1 & 1 & \cdots & 1 \\ \vdots & & & \vdots \\ 1 & \cdots & \cdots & 1 \end{pmatrix}\right))$
        to simulated $V^*$, obtain
        challenge $b$.
        If $b = \tilde{b}$: Open random Ham. path to $V^*$
        else: Goto ① (= rewind)
     return output $V^*$
```

ZK proof of the Hamiltonian path

First focus one round

Proof of the zero-knowledge:

Let V^* be a malicious verifier, and $x R w$.
Want to show that $\exists$ ppt Sim_{V^*} s.t.



Claim: we can define:

if $\tilde{b} = 1$:

Pick random π, send

$(Comm(\pi), Comm\left(\begin{pmatrix} 1 & 1 & \cdots & 1 \\ \vdots & & & \vdots \\ 1 & \cdots & \cdots & 1 \end{pmatrix}\right))$

Commit entries 1 by 1.

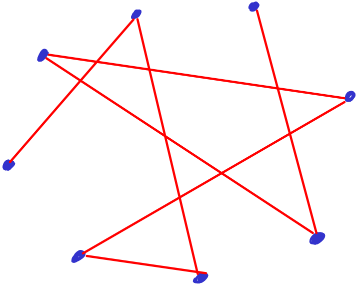
to simulated V^*, obtain challenge b.

If $b = \tilde{b}$: Open random Ham. path to V^*

else: Goto ① (= rewind)

return output V^*

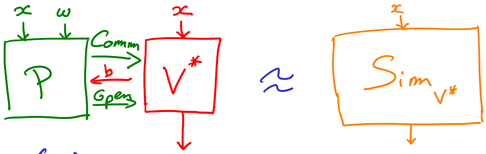
Trivial to do, e.g.:



ZK proof of the Hamiltonian path

First focus one round

Proof of the zero-knowledge: Let V^* be a malicious verifier, and $x R w$. Want to show that $\exists$ poly Sim_{V^*} s.t.



Claim 1: Sim_{V^*} runs in (expected) poly time => Show proba rewind $= 1/2 + \text{negl}$.

$=$ Proba V^* guesses $\tilde{b}$ given commitments of $f(\tilde{b})$.

$\text{Sim}_{V^*}(x)$:

① $\tilde{b} \leftarrow \{0,1\}$

if $\tilde{b} = 0$:

Pick random π, send $(\text{Comm}(\pi), \text{Comm}(G_\pi))$ to simulated V^*, obtain challenge b.

If $b = \tilde{b}$: Open all commitments to V^*

else: Goto ① (= rewind)

if $\tilde{b} = 1$:

Pick random π, send

$$\left(\text{Comm}(\pi), \text{Comm}\begin{pmatrix} 1 & 1 & \cdots & 1 \\ \vdots & & & \vdots \\ 1 & \cdots & \cdots & 1 \end{pmatrix}\right)$$

to simulated V^*, obtain challenge b.

If $b = \tilde{b}$: Open random Ham. path to V^*

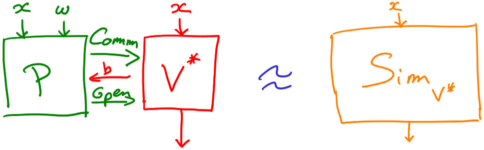
else: Goto ① (= rewind)

return output V^*

ZK proof of the Hamiltonian path

First focus one round

Proof of the zero-knowledge: Let V^* be a malicious verifier, and $x R w$. Want to show that $\exists$ poly Sim_{V^*} s.t.



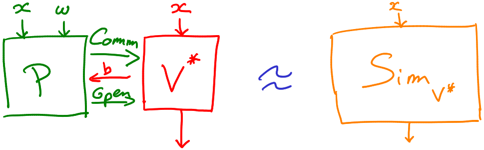
Claim 1: Sim_{V^*} runs in (expected) poly time => Show proba rewind $= 1/2 + \text{negl}$.

$=$ Proba V^* guesses $\tilde{b}$ given commitments of $f(\tilde{b})$.

ZK proof of the Hamiltonian path

First focus one round

Proof of the zero-knowledge:



Let V^* be a malicious verifier, and $x R w$.
Want to show that $\exists$ poly Sim_{V^*} s.t.

Claim 1: Sim_{V^*} runs in (expected) poly time $\Rightarrow$ Show $\underbrace{\text{proba rewind}}$ $= 1/2 + \text{negl}$.

$= \text{Proba } V^* \text{ guesses } \tilde{b} \text{ given commitments of } f(\tilde{b})$.

Def Comm is hiding iff

```
Guess($x_L, x_R$):
    $r \leftarrow \{0,1\}^\lambda$
    return Comm($r, x_L$)
```

$\Big\}\ \mathcal{L}_L$

$\approx$

```
Guess($x_L, x_R$):
    $r \leftarrow \{0,1\}^\lambda$
    return Comm($r, x_R$)
```

$\Big\}\ \mathcal{L}_R$

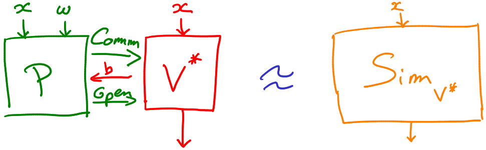
If $\Pr[V^*(f(\tilde{b})) = \tilde{b}] \geq \frac{1}{2} + \frac{1}{\text{poly}}$

Trivial to use V^* to distinguish $\mathcal{L}_L$ from $\mathcal{L}_R$:

ZK proof of the Hamiltonian path

First focus one round

Proof of the zero-knowledge: Let V^* be a malicious verifier, and $x R w$. Want to show that $\exists$ poly Sim_{V^*} s.t.



Def Comm is hiding iff

```
Guess($x_L, x_R$):
    $r \leftarrow \{0,1\}^\lambda$
    return Comm($r, x_L$)
```

$\mathcal{L}_L$

$\approx$

```
Guess($x_L, x_R$):
    $r \leftarrow \{0,1\}^\lambda$
    return Comm($r, x_R$)
```

$\mathcal{L}_R$

Claim 1: Sim_{V^*} runs in (expected) poly time $\Rightarrow$ Show proba rewind $= 1/2 + \text{negl}$.

$=$ Proba V^* guesses $\tilde{b}$ given commitments of $f(\tilde{b})$.

If $\Pr[V^*(f(\tilde{b})) = \tilde{b}] \geq \frac{1}{2} + \frac{1}{\text{poly}}$

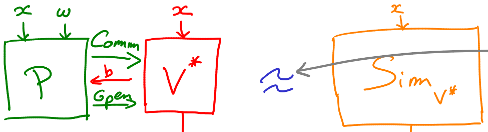
Trivial to use V^* to distinguish $\mathcal{L}_L$ from $\mathcal{L}_R$: for every commitment, call Guess(x_0, x_1), where x_0 is the object to commit when $b=0$, and x_1 when $b = 1$.

$\Rightarrow$ **Absurd** since commitments are hiding **!** So proba rewind $= 1/2 + \text{negl}$

ZK proof of the Hamiltonian path

First focus one round

Proof of the zero-knowledge: Let V^* be a malicious verifier, and $x R w$.
Want to show that $\exists$ poly Sim_{V^*} s.t.



Claim 2

Proof = rewrite Sim until we recover P→V.

Guess challenge of V^*

$\text{Sim}_{V^*}(x)$:

① $\tilde{b} \leftarrow \{0,1\}$

if $\tilde{b} = 0$:

Pick random π, send $(\text{Comm}(\pi), \text{Comm}(G_\pi))$ to simulated V^*, obtain challenge b.

Commit all entries one by one

Permuted adjacency matrix of G

If $b = \tilde{b}$: Open all commitments to V^*

else: Goto ① (= rewind)

if $\tilde{b} = 1$:

Pick random π, send $\left(\text{Comm}(\pi), \text{Comm}\begin{pmatrix} 1 & 1 & \cdots & 1 \\ \vdots & & & \vdots \\ 1 & \cdots & \cdots & 1 \end{pmatrix}\right)$ to simulated V^*, obtain challenge b.

Commit entries 1 by 1.

If $b = \tilde{b}$: Open random Ham. path to V^*

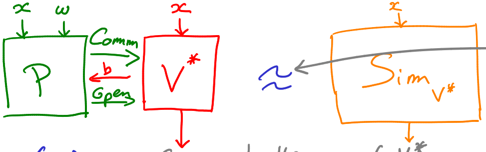
else: Goto ① (= rewind)

return output V^*

ZK proof of the Hamiltonian path

First focus one round

Proof of the zero-knowledge: Let V^* be a malicious verifier, and $x R w$.
Want to show that $\exists$ poly Sim_{V^*} s.t.



Claim 2

Proof = rewrite Sim until we recover P ↔ V.

Guess challenge of V^*

Commit all entries one by one

Permuted adjacency matrix of G

commit entries 1 by 1.

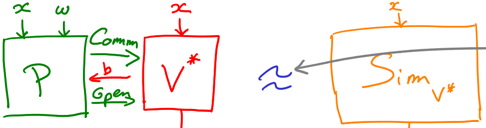
```
Sim_{V*}(x):
  ① $\tilde{b} \leftarrow \{0,1\}$
  if $\tilde{b} = 0$:
      Pick random $\pi$, send
          $(\mathrm{Comm}(\pi), \mathrm{Comm}(G_\pi))$
      to simulated $V^*$, obtain
      challenge $b$.
      If $b = \tilde{b}$: Open all commitments to $V^*$
      else: Goto ① (= rewind)
  if $\tilde{b} = 1$:
      Pick random $\pi$, send
          $\left(\mathrm{Comm}(\pi), \mathrm{Comm}\begin{pmatrix} 1 & 1 & \cdots & 1 \\ \vdots & & & \vdots \\ 1 & \cdots & \cdots & 1 \end{pmatrix}\right)$
      to simulated $V^*$, obtain
      challenge $b$.
      If $b = \tilde{b}$: Open ~~random Ham. path~~ Ham path of $G_\pi$ to $V^*$
      else: Goto ① (= rewind)
  return output $V^*$
```

ZK proof of the Hamiltonian path

First focus one round

Proof of the zero-knowledge:

Let V^* be a malicious verifier, and $x R w$.
Want to show that $\exists$ poly Sim_{V^*} s.t.



Claim 2

Proof = rewrite Sim until we recover $P \leftrightarrow V$.

Guess challenge of V^*

```
Sim_{V^*}(x):
    ① $\tilde{b} \leftarrow \{0,1\}$
    if $\tilde{b} = 0$:
        Pick random $\pi$, send
            $(\mathrm{Comm}(\pi), \mathrm{Comm}(G_\pi))$
        to simulated $V^*$, obtain
        challenge $b$.
        If $b = \tilde{b}$: Open all commitments to $V^*$
        else: Goto ① (= rewind)
    if $\tilde{b} = 1$:
        Pick random $\pi$, send
            $\left(\mathrm{Comm}(\pi), \mathrm{Comm}\left(\begin{pmatrix} 1 & 1 & \cdots & 1 \\ \vdots & & & \vdots \\ 1 & \cdots & & 1 \end{pmatrix}\right)\right)$
        to simulated $V^*$, obtain
        challenge $b$.
        If $b = \tilde{b}$: Open ~~random Ham. path~~ Ham path of $G_\pi$ to $V^*$
        else: Goto ① (= rewind)
    return output $V^*$
```

Commit all entries one by one

Permuted adjacency matrix of G

Commit entries 1 by 1.

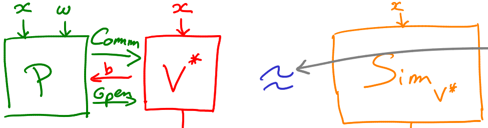
G_π → possible (hiding) ◇

ZK proof of the Hamiltonian path

First focus one round

Let V^* be a malicious verifier, and $x R w$.

Proof of the zero-knowledge: Want to show that $\exists$ poly Sim_{V^*} s.t.



Claim 2

Proof = rewrite Sim until we recover P ⟶ V.

Guess challenge of V^*

$Sim_{V^*}(x)$:

① $\tilde{b} \leftarrow \{0,1\}$

if $\tilde{b} = 0$:

Pick random π, send $(Comm(\pi), Comm(G_\pi))$ to simulated V^*, obtain challenge b.

Commit all entries one by one

Permuted adjacency matrix of G

If $b = \tilde{b}$: Open all commitments to V^*

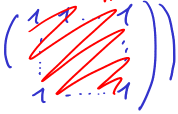
else: Goto ① (= rewind)

+ indep of $\tilde{b}$

if $\tilde{b} = 1$:

commit entries 1 by 1.

Pick random π, send $(Comm(\pi), Comm($



$))$ to simulated V^*, obtain challenge b.

G_π → possibly cheating

If $b = \tilde{b}$: Open ~~random Ham. path~~ Ham path of G_π to V^*

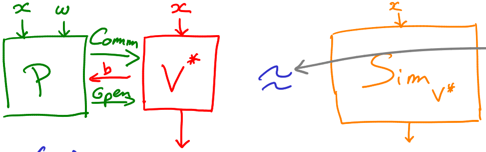
else: Goto ① (= rewind)

return output V^*

ZK proof of the Hamiltonian path

First focus one round

Proof of the zero-knowledge: Let V^* be a malicious verifier, and $x R w$. Want to show that $\exists$ poly Sim_{V^*} s.t.



Claim 2

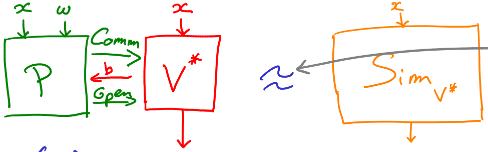
Proof = rewrite Sim until we recover P ⇄ V.

```
Sim_{V*}(x):
    ① Pick random $\pi$, send
        $(\mathrm{Comm}(\pi), \mathrm{Comm}(G_\pi))$
      to simulated $V^*$, obtain
      challenge $b$
    $\tilde{b} \leftarrow \{0,1\}$ → $b$ and $b'$ independent
    if $\tilde{b} = 0$:
        If $b = \tilde{b}$: Open all commitments to $V^*$
        else: Goto ① (= rewind)
    if $\tilde{b} = 1$:
        If $b = \tilde{b}$: Open ~~random Ham. path~~ Ham path of $G_\pi$
        else: Goto ① (= rewind)
    return output $V^*$
```

ZK proof of the Hamiltonian path

First focus one round

Proof of the zero-knowledge: Let V^* be a malicious verifier, and $x R w$. Want to show that $\exists$ poly Sim_{V^*} s.t.



Claim 2

Proof = rewrite Sim until we recover $P \leftrightarrow V$.

$\mathrm{Sim}_{V^*}(x)$:

① Pick random π, send $(\mathrm{Comm}(\pi), \mathrm{Comm}(G_\pi))$ to simulated V^*, obtain challenge b

$\tilde{b} \leftarrow \{0,1\}$

if $\tilde{b} = 0$: True

If ~~$b = \tilde{b}$~~ : Open all commitments to V^*

~~else : Goto ① (= rewind)~~

if $\tilde{b} = 1$: True

If ~~$b = \tilde{b}$~~ : Open ~~random Ham. path~~ Ham path of G_π to V^*

~~else : Goto ① (= rewind)~~

return output V^*

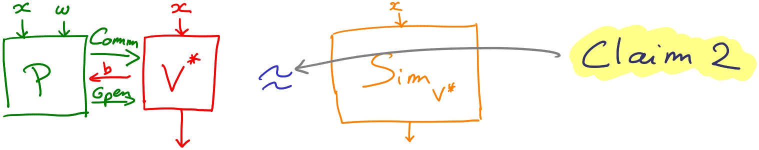
b and b' independent $\Rightarrow$ Same distribution if no rewind

$(\Pr(X = x \mid A) = \Pr(X = x)$ when A and X independent$)$

ZK proof of the Hamiltonian path

First focus one round

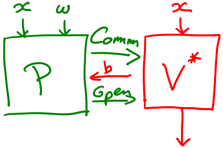
Proof of the zero-knowledge: Let V^* be a malicious verifier, and $x R w$. Want to show that $\exists$ poly Sim_{V^*} s.t.



Proof = rewrite Sim until we recover P ↔ V.

```
Pick random $\pi$, send
    $(\mathrm{Comm}(\pi), \mathrm{Comm}(G_\pi))$
to simulated $V^*$, obtain
challenge $b$
$\tilde{b} \leftarrow \{0,1\}$
if $\tilde{b} = 0$:
    Open all commitments to $V^*$
else Open Ham path of $G_\pi$ to $V^*$
return output of $V^*$
```

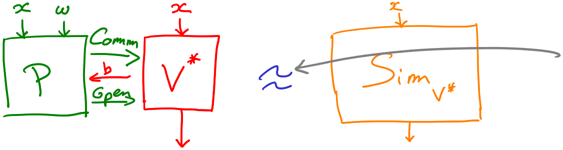
=



ZK proof of the Hamiltonian path

First focus one round

Proof of the zero-knowledge: Let V^* be a malicious verifier, and $x R w$. Want to show that $\exists$ poly Sim_{V^*} s.t.

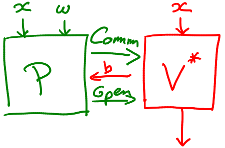


Claim 2

Proof = rewrite Sim until we recover P ⟷ V.

```
Pick random $\pi$, send
    $(\mathrm{Comm}(\pi), \mathrm{Comm}(G_\pi))$
to simulated $V^*$, obtain
challenge $b$
$\tilde{b} \leftarrow \{0,1\}$
if $\tilde{b} = 0$:
    Open all commitments to $V^*$
else
    Open Ham path of $G_\pi$ to $V^*$
return output of $V^*$
```

=

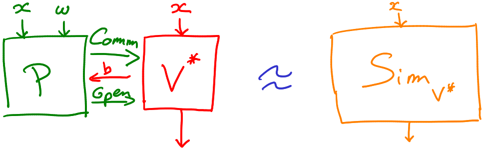


ZK proof of the Hamiltonian path

First focus one round

Proof of the zero-knowledge:

Let V^* be a malicious verifier, and $x R w$.
Want to show that $\exists$ poly Sim_{V^*} s.t.



n rounds

= simulate each round one by one (rewind to the beginning of the current round)

□

ZK proof of the Hamiltonian path

slightly easier proof

Proof of the soundness: 1) Prove $\frac{1}{2}$-soundness of 1 round assuming statistical binding

ZK proof of the Hamiltonian path

slightly easier proof

Proof of the soundness: 1) Prove $\frac{1}{2}$-soundness of 1 round assuming statistical binding

Contradiction: assume NOT $\frac{1}{2}$-sound.

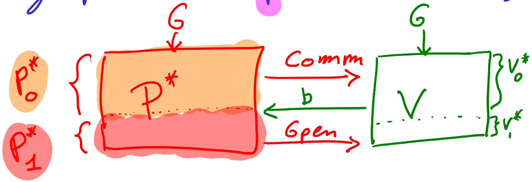
ZK proof of the Hamiltonian path

slightly easier proof

Proof of the soundness: 1) Prove $\frac{1}{2}$-soundness of 1 round assuming statistical binding

Contradiction: assume NOT $\frac{1}{2}$-sound. Then $\exists P^*$, and a non-Hamiltonian graph G s.t. $p := \Pr[\langle V(G), P^*(G)\rangle = \top] > \frac{1}{2}$

output V



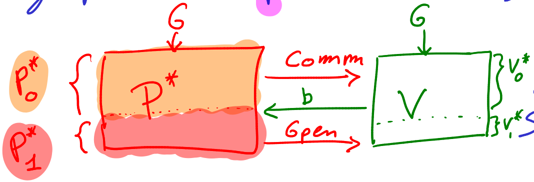
ZK proof of the Hamiltonian path

slightly easier proof

Proof of the soundness: 1) Prove $\frac{1}{2}$-soundness of 1 round assuming statistical binding

Contradiction: assume NOT $\frac{1}{2}$ sound. Then $\exists P^*$, and a non-Hamiltonian graph G s.t. $p := \Pr[\langle V(G), P^*(G)\rangle = \top] > \frac{1}{2}$

output V



WLOG, we can assume that P_0^* and P_1^* are deterministic (take max over randomness). So strategy of $P^*(G) :=$

- Comm } first message
- $Gpen_0$ } output when b=0
- $Gpen_1$ } output when b=1

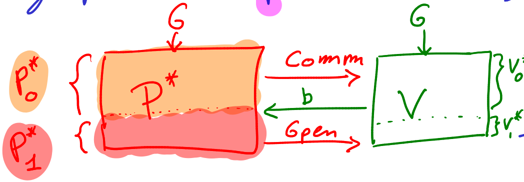
ZK proof of the Hamiltonian path

slightly easier proof

Proof of the soundness: 1) Prove $\frac{1}{2}$-soundness of 1 round assuming statistical binding

Contradiction: assume NOT $\frac{1}{2}$ sound. Then $\exists P^*$, and a non-Hamiltonian graph G s.t. $p := \Pr[\langle V(G), P^*(G)\rangle = \top] > \frac{1}{2}$

output V



WLOG, we can assume that P^*_0 and P^*_1 are deterministic (take max over randomness). So strategy of $P^*(G) :=$

- Comm } first message
- Gpen_0 } output when b=0
- Gpen_1 } output when b=1

By def. $p = \frac{1}{2}\left(\underbrace{\Pr[V \text{ accepts} \mid b = 0]}_{a_0} + \underbrace{\Pr[V \text{ accepts} \mid b = 1]}_{a_1}\right)$. Since V^*_1 is deterministic, $a_0 \in \{0,1\}$ and $a_1 \in \{0,1\}$. Since $p > \frac{1}{2}$, we have $a_0 = a_1 = 1$, i.e. V always accept.

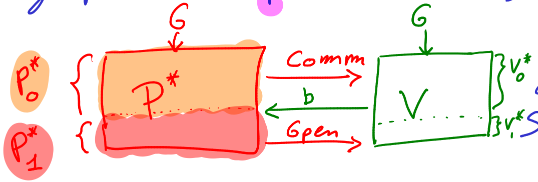
ZK proof of the Hamiltonian path

slightly easier proof

Proof of the soundness: 1) Prove $\frac{1}{2}$-soundness of 1 round assuming statistical binding

Contradiction: assume NOT $\frac{1}{2}$-sound. Then $\exists P^*$, and a non-Hamiltonian graph G s.t. $p := \Pr[\langle V(G), P^*(G)\rangle = \top] > \frac{1}{2}$

output V



WLOG, we can assume that P^*_0 and P^*_1 are deterministic (take max over randomness). So strategy of $P^*(G) :=$

- Comm } first message
- Open_0 } output when b=0
- Open_1 } output when b=1

By def. $p = \frac{1}{2}(\underbrace{\Pr[V \text{ accepts} \mid b=0]}_{a_0} + \underbrace{\Pr[V \text{ accepts} \mid b=1]}_{a_1})$. Since V^*_1 is deterministic, $a_0 \in \{0,1\}$ and $a_1 \in \{0,1\}$. Since $p > \frac{1}{2}$, we have $a_0 = a_1 = 1$, i.e. V always accept.

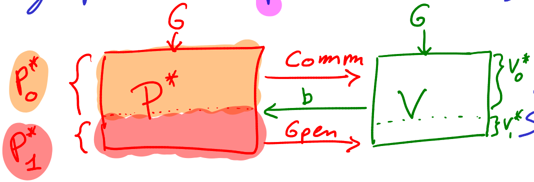
Comm is statistically binding => only one possible openning which is both a valid permutation of G (V accepts when b=0) and contains a Hamiltonian path (V accepts when b=1). Absurd! (G not Hamiltonian)

ZK proof of the Hamiltonian path

Proof of the soundness: 1) Prove $\frac{1}{2}$-soundness of 1 round assuming ~~statistical~~ binding

Contradiction: assume NOT $\frac{1}{2}$-sound. Then $\exists P^*$, and a non-Hamiltonian graph G s.t. $p := \Pr[\langle V(G), P^*(G)\rangle = \top] > \frac{1}{2}$

output V



WLOG, we can assume that P_0^* and P_1^* are deterministic (take max over randomness). So strategy of $P^*(G) :=$
- • Comm } first message
- • Open_0 } output when b=0
- • Open_1 } output when b=1

By def. $p = \frac{1}{2}\left(\underbrace{\Pr[V \text{ accepts} \mid b=0]}_{a_0} + \underbrace{\Pr[V \text{ accepts} \mid b=1]}_{a_1}\right)$. Since V_i^* is deterministic, $a_0 \in \{0,1\}$ and $a_1 \in \{0,1\}$. Since $p > \frac{1}{2}$, we have $a_0 = a_1 = 1$, i.e. V always accept.

Comm is ~~statistically~~ binding => 1st case: if Open_0 an Open_1 have identical openings: absurd (cf stat. binding)

2nd case: Open_0 and Open_1 have ≠ openings → Absurd (commitment is binding)

ZK proof of the Hamiltonian path

Proof of the soundness: 2) For n rounds :

At each round, the verifier accepts with proba $\leq \frac{1}{2}$.
(corollary last slide)

$$\Rightarrow \Pr[\langle V, P^* \rangle = accept] \leq \left(\frac{1}{2}\right)^n$$

$$= negl$$

$\square$

Proof of knowledge

How can we be sure that the prover **“knows” the secret?**

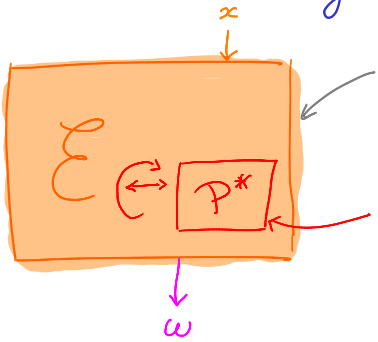
E.g.: For $y \in \mathbb{Z}_p^\times$, I can convince you that there exists x such that $g^x = y$ (e.g. g is a generator of $\mathbb{Z}_p^\times$, i.e. for all x dividing $p - 1$, $g^x \neq 1$), but I may not always know x (hardness of discrete log).

proof of membership $\neq$ proof of knowledge

How to define this notion formally?

Proof of knowledge

P^* "knows" ω if we can extract ω from the "source code" of P^*:

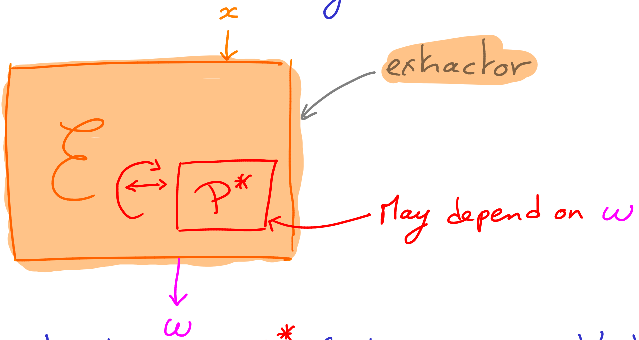


extractor

May depend on ω

Proof of knowledge

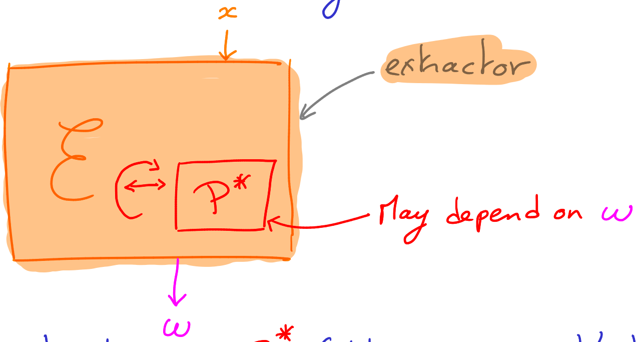
P^* "knows" ω if we can extract ω from the "source code" of P^*:

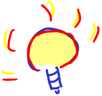


Pb: Cannot be true $\forall P^*$ (otherwise possible to efficiently solve NP problems)

Proof of knowledge

P^* "knows" ω if we can extract ω from the "source code" of P^*:





Pb Cannot be true $\forall P^*$ (otherwise possible to efficiently solve NP problems),

$\Rightarrow$ True for "convincing" P^*

Proof of knowledge

Definition (ZKPoK)

A ZK protocol for a language in NP (with relation $\mathcal{R}$) is said to be a **proof of knowledge** (ZKPoK) (with error $\kappa(\lambda)$) if there exists an efficient algorithm $\mathcal{E}$ given rewindable oracle access to P^*, called an **extractor**, such that for any x and prover P^*, if $\Pr[\langle P^*(x), V(x)\rangle = \top] > \kappa(|x|)$, $\mathcal{E}^{P^*}(x)$ returns a valid witness w ($x\mathcal{R}w$) in time $\frac{\text{poly}(\lambda)(|x|)}{\Pr[\langle P^*(x),V(x)\rangle=\top]-\kappa(|x|)}$

? Why isn't it contradicting the ZK property?

Proof of knowledge

Definition (ZKPoK)

A ZK protocol for a language in NP (with relation $\mathcal{R}$) is said to be a **proof of knowledge** (ZKPoK) (with error $\kappa(\lambda)$) if there exists an efficient algorithm $\mathcal{E}$ given rewindable oracle access to P^*, called an **extractor**, such that for any x and prover P^*, if $\Pr[\langle P^*(x), V(x)\rangle = \top] > \kappa(|x|)$, $\mathcal{E}^{P^*}(x)$ returns a valid witness w ($x\mathcal{R}w$) in time $\frac{\text{poly}(\lambda)(|x|)}{\Pr[\langle P^*(x),V(x)\rangle=\top]-\kappa(|x|)}$

Typically best trivial cheating proba (e.g $\kappa = \frac{1}{2^\lambda}$)

? Why isn't it contradicting the ZK property?

Proof of knowledge

Definition (ZKPoK)

A ZK protocol for a language in NP (with relation $\mathcal{R}$) is said to be a **proof of knowledge** (ZKPoK) (with error $\kappa(\lambda)$) if there exists an efficient algorithm $\mathcal{E}$ given rewindable oracle access to P^*, called an **extractor**, such that for any x and prover P^*, if $\Pr[\langle P^*(x), V(x)\rangle = \top] > \kappa(|x|)$, $\mathcal{E}^{P^*}(x)$ returns a valid witness w ($x\mathcal{R}w$) in time $\frac{\mathrm{poly}(\lambda)(|x|)}{\Pr[\langle P^*(x),V(x)\rangle=\top]-\kappa(|x|)}$

→ Typically best trivial cheating proba (e.g. $\kappa = \frac{1}{2^\lambda}$)

↳ If we can do much better = we find w!

Why isn't it contradicting the ZK property?

Proof of knowledge

Definition (ZKPoK)

A ZK protocol for a language in NP (with relation $\mathcal{R}$) is said to be a **proof of knowledge** (ZKPoK) (with error $\kappa(\lambda)$) if there exists an efficient algorithm $\mathcal{E}$ given rewindable oracle access to P^*, called an **extractor**, such that for any x and prover P^*, if $\Pr[\langle P^*(x), V(x)\rangle = \top] > \kappa(|x|)$, $\mathcal{E}^{P^*}(x)$ returns a valid witness w ($x\mathcal{R}w$) in time $\frac{\mathsf{poly}(\lambda)(|x|)}{\Pr[\langle P^*(x), V(x)\rangle = \top] - \kappa(|x|)}$

?

Why isn't it contradicting the ZK property?
The extractor can rewind P^* etc

ZK proof of the Hamiltonian path

Theorem (ZK-Ham

The ZK protocol for the Hamiltonian path is a zero-knowledge proof of knowledge.

Proof idea: the extractor plays the protocol honestly with $b = 0$, rewinds P^*, and then sends $b = 1$. This way it gets both a Hamiltonian path and π, so it can revert π on the Hamiltonian path to recover a Hamiltonian path on G.

When sending 2 challenges is enough to recover the witness = called **special soundness**

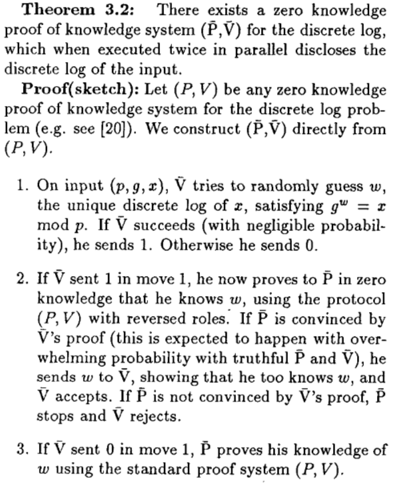
Reducing interactivity

Parallel repetition

For efficiency, tempting to repeat the ZK protocol for Hamiltonian path in parallel instead of sequentially.

1. **Wrong** in general: there exist ZK protocols secure when composed sequentially, but not in parallel [Feige, Shamir STOC 90] (see next slide)
2. **Unknown** for the protocol for Hamiltonian paths
3. **Known** for this protocol if the challenges of the verifier are random (semi-honest verifier) $\Rightarrow$ Fiat-shamir's construction has this property!

Parallel repetition



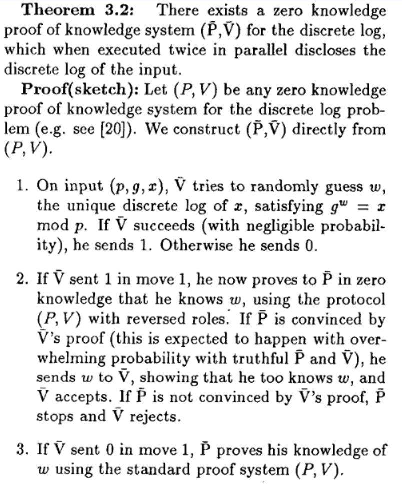
Theorem 3.2: There exists a zero knowledge proof of knowledge system $(\bar{P},\bar{V})$ for the discrete log, which when executed twice in parallel discloses the discrete log of the input.

Proof(sketch): Let (P, V) be any zero knowledge proof of knowledge system for the discrete log problem (e.g. see [20]). We construct $(\bar{P},\bar{V})$ directly from (P, V).

1. On input (p, g, x), $\bar{V}$ tries to randomly guess w, the unique discrete log of x, satisfying $g^w = x \mod p$. If $\bar{V}$ succeeds (with negligible probability), he sends 1. Otherwise he sends 0.
2. If $\bar{V}$ sent 1 in move 1, he now proves to $\bar{P}$ in zero knowledge that he knows w, using the protocol (P, V) with reversed roles. If $\bar{P}$ is convinced by $\bar{V}$'s proof (this is expected to happen with overwhelming probability with truthful $\bar{P}$ and $\bar{V}$), he sends w to $\bar{V}$, showing that he too knows w, and $\bar{V}$ accepts. If $\bar{P}$ is not convinced by $\bar{V}$'s proof, $\bar{P}$ stops and $\bar{V}$ rejects.
3. If $\bar{V}$ sent 0 in move 1, $\bar{P}$ proves his knowledge of w using the standard proof system (P, V).

Can you prove that this scheme is NOT Zero-Knowledge when composed in parallel twice?

Parallel repetition

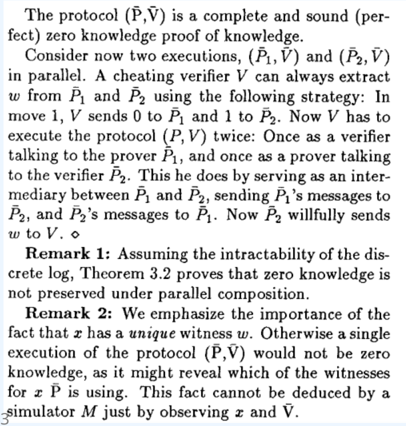


Theorem 3.2: There exists a zero knowledge proof of knowledge system ($\bar{P}$,$\bar{V}$) for the discrete log, which when executed twice in parallel discloses the discrete log of the input.

Proof(sketch): Let (P, V) be any zero knowledge proof of knowledge system for the discrete log problem (e.g. see [20]). We construct ($\bar{P}$,$\bar{V}$) directly from (P, V).

1. On input (p, g, x), $\bar{V}$ tries to randomly guess w, the unique discrete log of x, satisfying $g^w = x$ mod p. If $\bar{V}$ succeeds (with negligible probability), he sends 1. Otherwise he sends 0.
2. If $\bar{V}$ sent 1 in move 1, he now proves to $\bar{P}$ in zero knowledge that he knows w, using the protocol (P, V) with reversed roles. If $\bar{P}$ is convinced by $\bar{V}$'s proof (this is expected to happen with overwhelming probability with truthful $\bar{P}$ and $\bar{V}$), he sends w to $\bar{V}$, showing that he too knows w, and $\bar{V}$ accepts. If $\bar{P}$ is not convinced by $\bar{V}$'s proof, $\bar{P}$ stops and $\bar{V}$ rejects.
3. If $\bar{V}$ sent 0 in move 1, $\bar{P}$ proves his knowledge of w using the standard proof system (P, V).

Can you prove that this scheme is NOT Zero-Knowledge when composed in parallel twice?



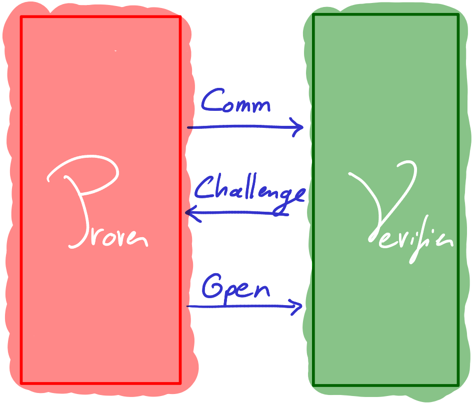
The protocol ($\bar{P}$,$\bar{V}$) is a complete and sound (perfect) zero knowledge proof of knowledge.

Consider now two executions, $(\bar{P}_1, \bar{V})$ and $(\bar{P}_2, \bar{V})$ in parallel. A cheating verifier V can always extract w from $\bar{P}_1$ and $\bar{P}_2$ using the following strategy: In move 1, V sends 0 to $\bar{P}_1$ and 1 to $\bar{P}_2$. Now V has to execute the protocol (P, V) twice: Once as a verifier talking to the prover $\bar{P}_1$, and once as a prover talking to the verifier $\bar{P}_2$. This he does by serving as an intermediary between $\bar{P}_1$ and $\bar{P}_2$, sending $\bar{P}_1$'s messages to $\bar{P}_2$, and $\bar{P}_2$'s messages to $\bar{P}_1$. Now $\bar{P}_2$ willfully sends w to V. ⋄

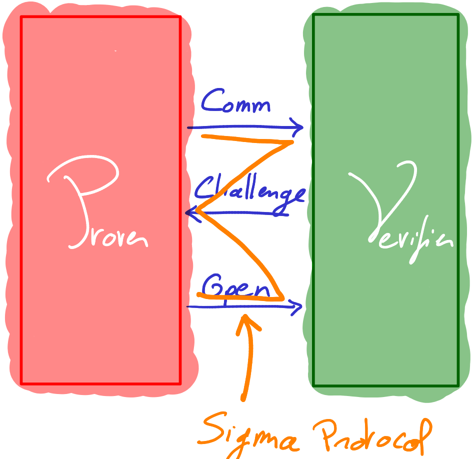
Remark 1: Assuming the intractability of the discrete log, Theorem 3.2 proves that zero knowledge is not preserved under parallel composition.

Remark 2: We emphasize the importance of the fact that x has a *unique* witness w. Otherwise a single execution of the protocol ($\bar{P}$,$\bar{V}$) would not be zero knowledge, as it might reveal which of the witnesses for x $\bar{P}$ is using. This fact cannot be deduced by a simulator M just by observing x and $\bar{V}$.

Sigma protocol



Sigma protocol

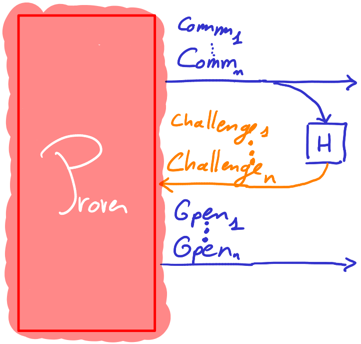


Fiat Shamir

How to make the protocol non-interactive (NIZK): **Fiat-Shamir** transform

1. Run the protocol in parallel
2. Replace the challenge with the hash of all commitments of first phase

FIAT-SHAMIR:



Fiat Shamir

Is it still secure if we hash the challenges one by one?

Fiat Shamir

How to prove security of the Fiat Shamir transform?

Fiat Shamir

How to prove security of the Fiat Shamir transform?

Claim: Impossible

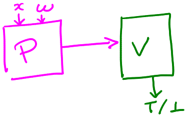
(in plain model)

Fiat Shamir

Claim: Impossible
(in plain model)

How to prove security of the Fiat Shamir transform?

In the non-interactive case:

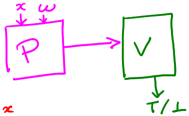


Fiat Shamir

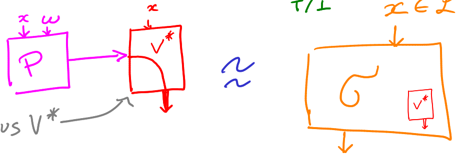
How to prove security of the Fiat Shamir transform?

Claim: Impossible
(in plain model)

In the non-interactive case:



⇒ ZK = ∃ $\mathcal{S}$ /



Particular case of malicious V*

Fiat Shamir

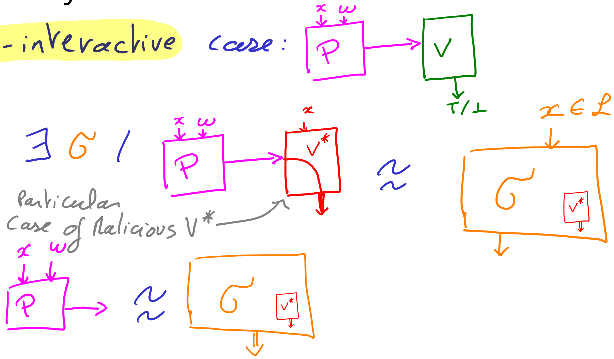
How to prove security of the Fiat Shamir transform?

Claim: Impossible

(in plain model)

In the non-interactive case:

⇒ ZK = ∃ σ /



Particular case of malicious V*

i.e.

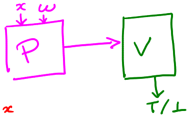
Fiat Shamir

How to prove security of the Fiat Shamir transform?

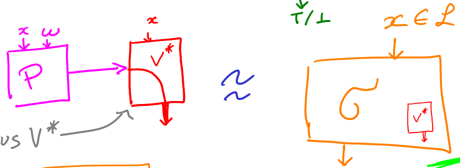
Claim: Impossible

(in plain model)

In the non-interactive case:

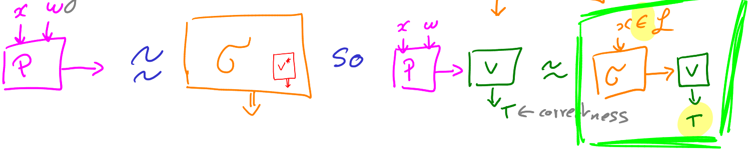


⇒ ZK = ∃ σ /



Particular case of Malicious V^*

i.e.



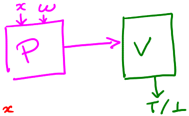
⊤ ← correctness

Fiat Shamir

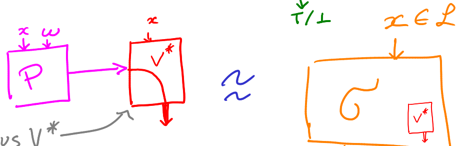
How to prove security of the Fiat Shamir transform?

Claim: Impossible (in plain model)

In the non-interactive case:

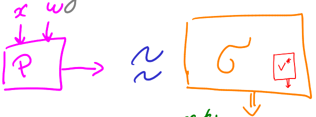


$\Rightarrow$ ZK $= \exists\, \sigma$ /

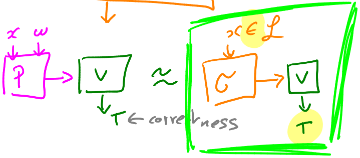


Particular case of Malicious V^*

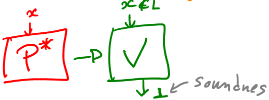
i.e.



So



$\Rightarrow$ Soundness $\forall P^*$,

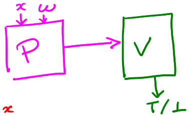


Fiat Shamir

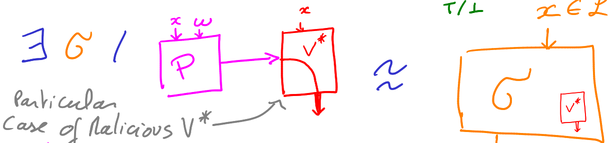
How to prove security of the Fiat Shamir transform?

Claim: Impossible
(in plain model)

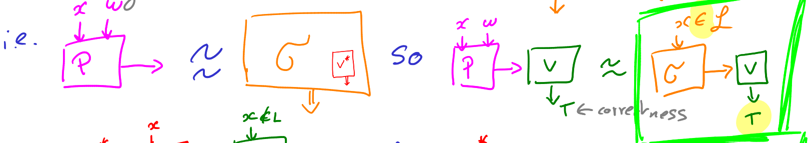
In the non-interactive case:



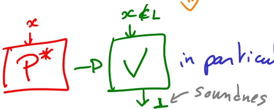
⇒ ZK = ∃ $\mathcal{S}$ /



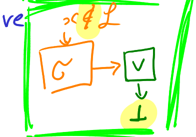
i.e.



⇒ **Soundness** ∀P*,



in particular if P* = $\mathcal{S}$ we have



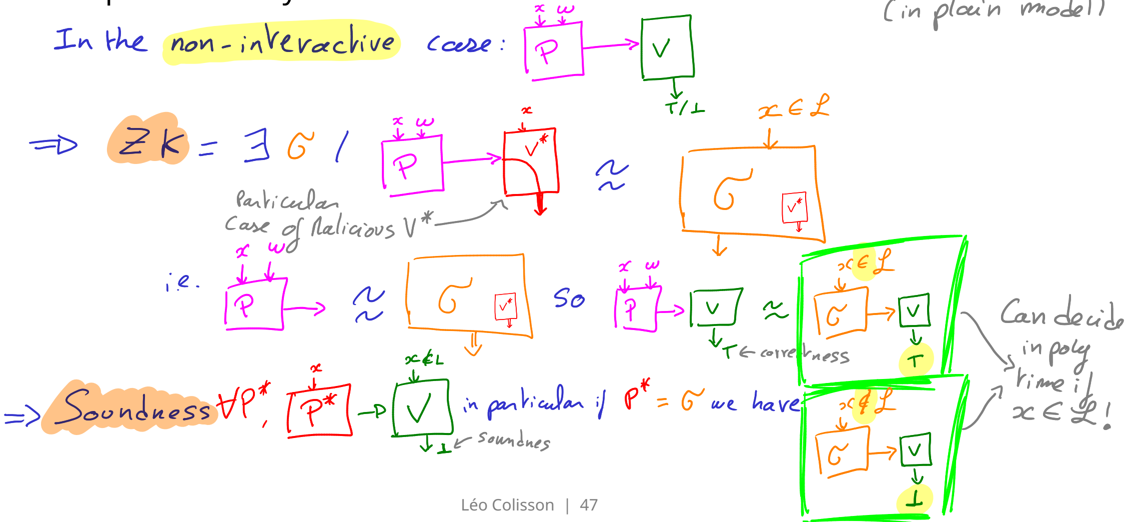
Fiat Shamir

How to prove security of the Fiat Shamir transform?

Claim: Impossible

(in plain model)

In the non-interactive case:



⇒ ZK = ∃ $\mathcal{S}$ /

Particular case of Malicious V*

i.e.

So

⊤ ← correctness

Can decide in poly time if $x \in \mathcal{L}$!

⇒ Soundness ∀P*,

$x \notin L$

⊥ ← soundness

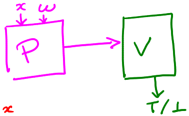
in particular if $P^* = \mathcal{S}$ we have

Fiat Shamir

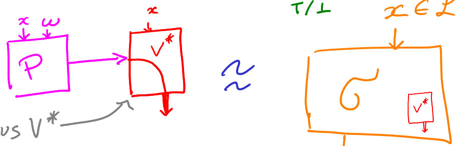
How to prove security of the Fiat Shamir transform?

Claim: Impossible
(in plain model)

In the non-interactive case:

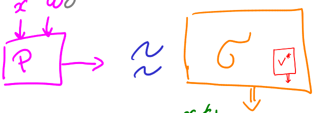


⇒ ZK = ∃ σ /

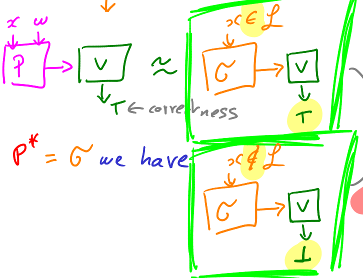


Particular case of Malicious V*

i.e.



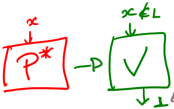
So



Can decide in poly time if x ∈ ℒ!

Absurd if P ≠ NP

⇒ Soundness ∀P*,

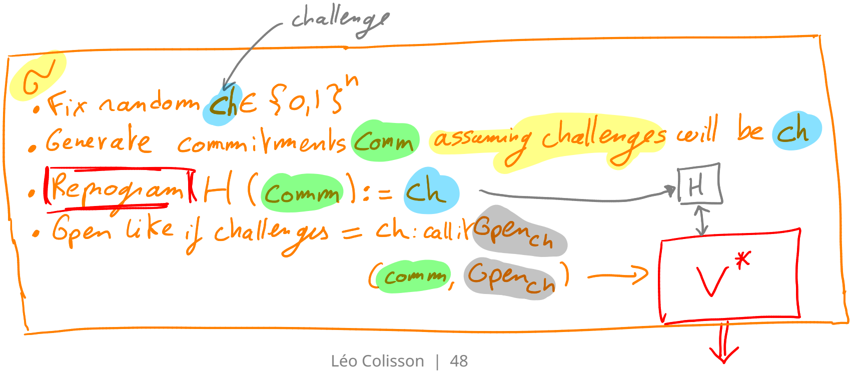


in particular if P* = σ we have

Fiat Shamir

How to prove security of the Fiat Shamir transform?
Solutions:

- Consider the Random Oracle Model
- The simulator can **reprogram** the oracle



challenge

S

- Fix random $ch \in \{0,1\}^n$
- Generate commitments Comm assuming challenges will be ch
- Reprogram $H(Comm) := ch$
- Open like if challenges = ch: call $Open_{ch}$

$(comm, Open_{ch}) \rightarrow V^*$

Efficiency?

	Universal	**Efficient**	**Simplicity**	**Post-quantum**
Hamiltonian path	✓	✗	✗	✓
Specialized approaches	✗	✓ ✓	✓	Depends
ZK-SNARK	✓	✓	✗ ✗	✗
ZK-STARK	✓	✓	✗ ✗	✓

Efficiency?

MANY REDUCTIONS
("KARP REDUCTIONS")
$\mathcal{L} \rightarrow SAT \rightarrow Ham$

	Universal	**Efficient**	**Simplicity**	**Post-quantum**
Hamiltonian path	✓	✗	✗	✓
Specialized approaches	✗	✓ ✓	✓	Depends
ZK-SNARK	✓	✓	✗ ✗	✗
ZK-STARK	✓	✓	✗ ✗	✓

Efficiency?

MANY REDUCTIONS
("KARP REDUCTIONS")
$\mathcal{L} \to$ SAT $\to$ Ham

	Universal	**Efficient**	**Simplicity**	**Post-quantum**
Hamiltonian path	✓	✗	✗	✓
Specialized approaches	✗	✓ ✓	✓	Depends
ZK-SNARK	✓	✓	✗ ✗	✗
ZK-STARK	✓	✓	✗ ✗	✓

→ CF. COURSE VANESSA

More efficient authentication & signature protocols

Zero-Knowledge proofs for discrete logarithm (DL)

Specialized solution: I know x such that $g^x = y$ (operations in $\mathbb{Z}_p^\times$ or arbitrary cyclic group G).

ZK for the discrete logarithm (DL)

Alice(p, g, y, x)		**Bob**(p, g, y)
$r \xleftarrow{\$} \mathbb{Z}_p^*$	$\xrightarrow{R := g^r}$	
	$\xleftarrow{b}$	$b \xleftarrow{\$} \{0,1\}$
$s := (r + bx) \mod (p-1)$	$\xrightarrow{s}$	
		$\mathbf{return}\ g^s \overset{?}{=} Ry^b$

Prove the correctness.

Zero-Knowledge proofs for discrete logarithm (DL)

Specialized solution: I know x such that $g^x = y$ (operations in $\mathbb{Z}_p^\times$ or arbitrary cyclic group G).

ZK for the discrete logarithm (DL)

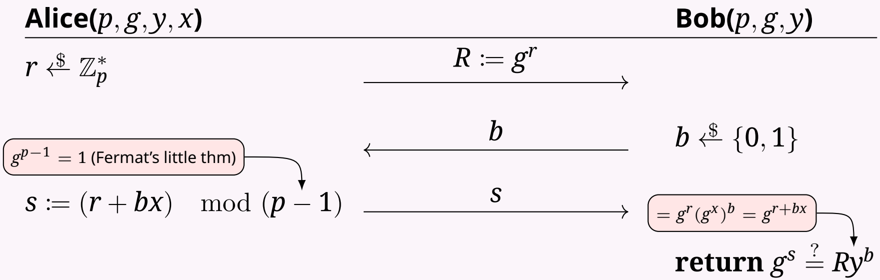
Alice(p, g, y, x)		**Bob**(p, g, y)
$r \xleftarrow{\$} \mathbb{Z}_p^*$	$\xrightarrow{\quad R := g^r \quad}$	
	$\xleftarrow{\quad b \quad}$	$b \xleftarrow{\$} \{0, 1\}$
$g^{p-1} = 1$ (Fermat's little thm) $s := (r + bx) \mod (p - 1)$	$\xrightarrow{\quad s \quad}$	
		$\textbf{return } g^s \stackrel{?}{=} R y^b$

? Prove the correctness.

Zero-Knowledge proofs for discrete logarithm (DL)

Specialized solution: I know x such that $g^x = y$ (operations in $\mathbb{Z}_p^\times$ or arbitrary cyclic group G).

ZK for the discrete logarithm (DL)



Alice(p, g, y, x)

Bob(p, g, y)

$r \xleftarrow{\$} \mathbb{Z}_p^*$

$\xrightarrow{R := g^r}$

$b \xleftarrow{\$} \{0, 1\}$

$\xleftarrow{b}$

$g^{p-1} = 1$ (Fermat's little thm)

$s := (r + bx) \mod (p-1)$

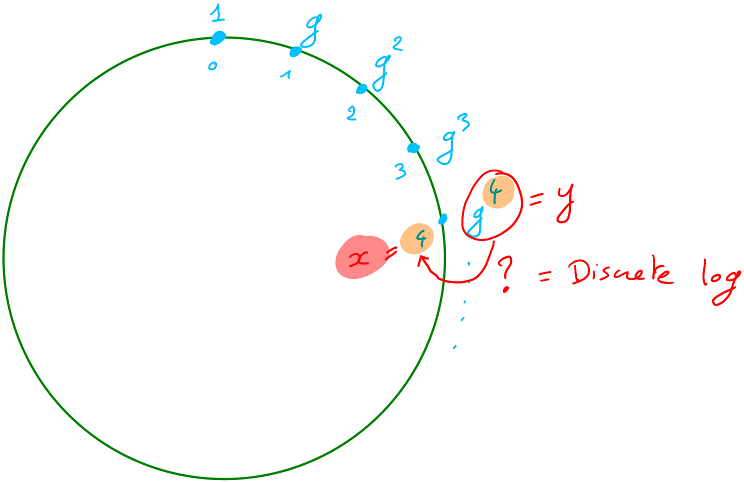
$\xrightarrow{s}$

$= g^r (g^x)^b = g^{r+bx}$

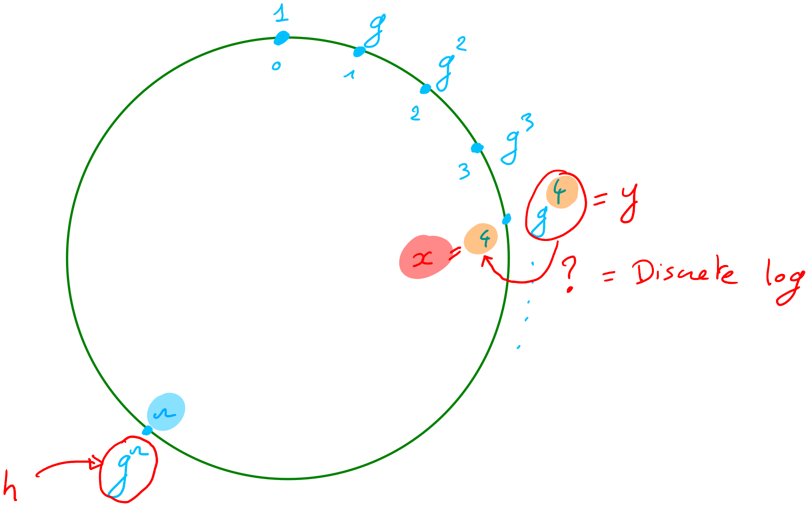
$\textbf{return } g^s \stackrel{?}{=} R y^b$

Prove the correctness.

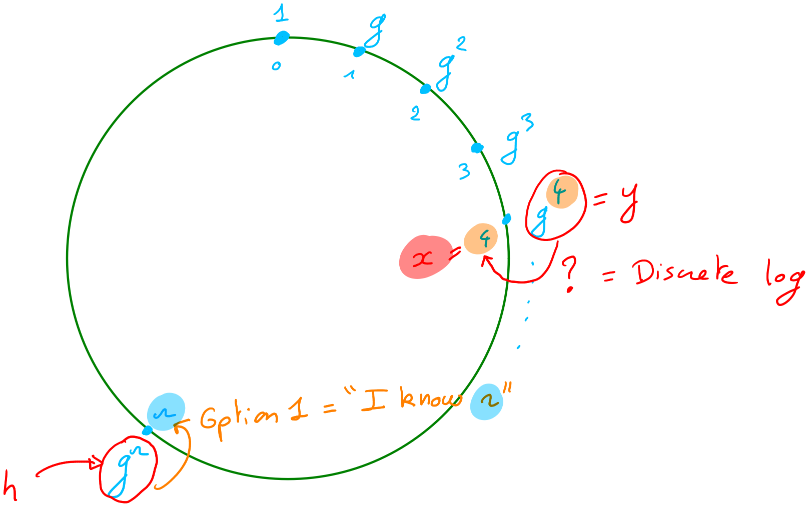
Zero-Knowledge proofs for discrete logarithm (DL)



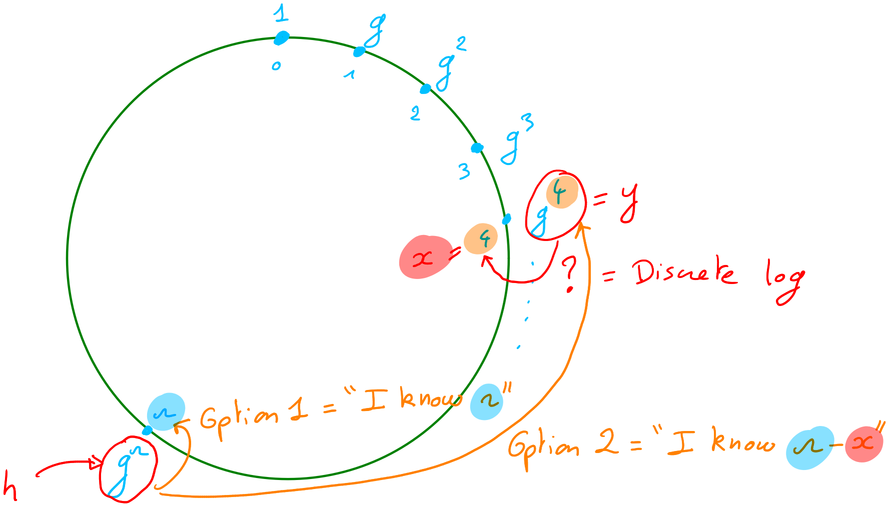
Zero-Knowledge proofs for discrete logarithm (DL)



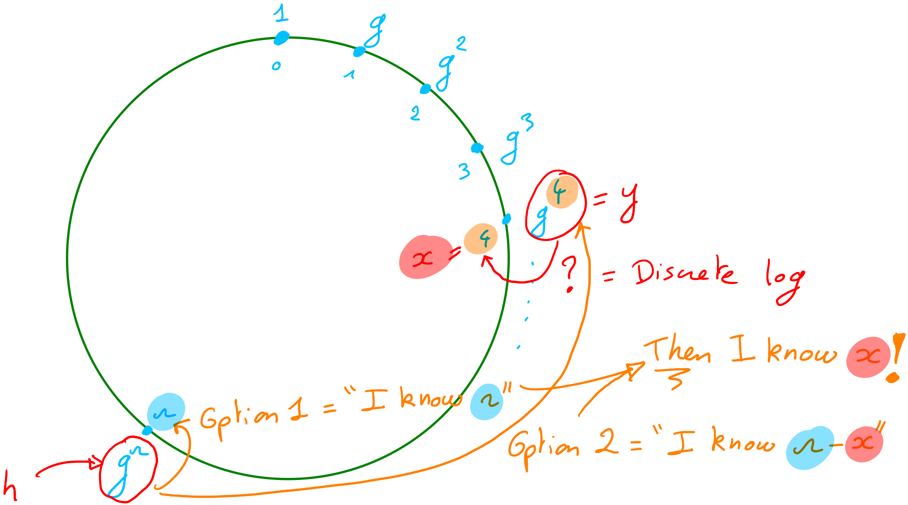
Zero-Knowledge proofs for discrete logarithm (DL)



Zero-Knowledge proofs for discrete logarithm (DL)



Zero-Knowledge proofs for discrete logarithm (DL)



Zero-Knowledge proofs for discrete logarithm (DL)

? Prove that this protocol is:

- ZK
- sound
- special sound

Schnorr signature

Problem of the above protocol: need n rounds to have security $\frac{1}{2^n}$. **Not very efficient**.

Schnorr signature = 1 **round** without (quite inneficient) Fiat Shamir!

$\Rightarrow$ Idea: **more than** 2 **challenges.**

Schnorr signature

Kept by some trusted authority, $g^x = y$

Schnorr authentication

Alice(p, g, y, x) **Bob**(p, g, y)

$r \xleftarrow{\$} \mathbb{Z}_p^*$ $\xrightarrow{R := g^r}$

$\xleftarrow{c}$ $c \xleftarrow{\$} \mathbb{Z}_p^*$

$s := (r + cx) \mod (p-1)$ $\xrightarrow{s}$

$\textbf{return } ??? \overset{?}{=} ???$

Find the verification procedure.

Schnorr signature

Kept by some trusted authority, $g^x = y$

Schnorr authentication

Alice(p, g, y, x) **Bob**(p, g, y)

$r \xleftarrow{\$} \mathbb{Z}_p^*$ $\xrightarrow{\quad R := g^r \quad}$

$\xleftarrow{\quad c \quad}$ $c \xleftarrow{\$} \mathbb{Z}_p^*$

$s := (r + cx) \mod (p-1)$ $\xrightarrow{\quad s \quad}$

return $g^s \overset{?}{=} Ry^c$

?

Find the verification procedure.

Schnorr signature

This allows someone to check if we interact with Alice, but two issues:

- this is interactive
- not a signature for now

$\Rightarrow$ Solution: Fiat-Shamir (one round) where the hash is based on the message to sign and commit.

Schnorr signature

Schnorr signature

Let $H \colon G \times \{0,1\}^* \to \mathbb{Z}_q^*$ be a hash function, m a message to sign and $y = g^x$ such that x is kept secret by Alice, and y is public.

Alice(p, g, y, x, m) **Bob(p, g, y, m)**

$r \xleftarrow{\$} \mathbb{Z}_p^*$

$R \coloneqq g^r$

$c \coloneqq H(R, m)$

$s \coloneqq (r + cx) \mod (p-1) \quad \xrightarrow{\quad (R, s) \quad}$

$\textbf{return } g^s \stackrel{?}{=} hy^{H(R,m)}$

Schnorr signature

Schnorr’s signature is used in real life, e.g. in the **Bitcoin** protocol (group: secp256k1 elliptic curve) to replace ECDSA:

- **Provably secure**: strongly unforgeable under chosen message attack (SUF-CMA) in the ROM assuming hardness of DL
- Can be generalized to **sign a message collaboratively** exploiting linearity

`https://github.com/bitcoin/bips/blob/master/bip-0340.mediawiki`

Goldreich-Levin construction

Goldreich-Levin

Earlier: **how to obtain bit commitment from one-way permutations**:

- $f : \{0,1\}^* \to \{0,1\}^*$
- $p : \{0,1\}^* \to \{0,1\}$ hard-core predicate
  (hard to guess $p(x)$ given $f(x)$, exists thanks to the Goldreich-Levin theorem)
- $x \in \{0,1\}$

Commit$(x,r) = (f(r), p(r) \oplus x)$, Open$((y,b),x,r) = ((y,b) \stackrel{?}{=} (f(r), p(r) \oplus x))$
(permutation needed for (statistical) binding, otherwise we need something like collision resistance)

Goldreich-Levin

Earlier: **how to obtain bit commitment from one-way permutations**:

- $f : \{0,1\}^* \to \{0,1\}^*$
- $p : \{0,1\}^* \to \{0,1\}$ hard-core predicate (hard to guess $p(x)$ given $f(x)$, exists thanks to the Goldreich-Levin theorem)
- $x \in \{0,1\}$

$\text{Commit}(x,r) = (f(r), p(r) \oplus x)$, $\text{Open}((y,b),x,r) = ((y,b) \overset{?}{=} (f(r), p(r) \oplus x))$
(permutation needed for (statistical) binding, otherwise we need something like collision resistance)

Goldreich-Levin

Hard to invert ≃ minimal assumption

One Way Function

Goldreich-Levin

Hard to invert

One Way Function

↳ Pb: some bits may not be secrets:

E.g: $f(b \| x) := b \| SHA(x)$

Goldreich-Levin

Hard to invert

One Way Function

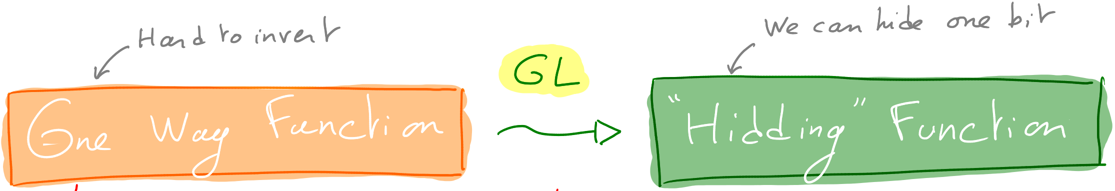
↳ Pb: some bits may not be secrets:

E.g: $f(b \| x) := b \| SHA(x)$

$f(b \| x)$ leaks b but is OWF

=> bad commitment scheme

Goldreich-Levin



↳ Pb: some bits may <u>not</u> be secrets:

E.g: $f(b \| x) := b \| SHA(x)$

$f(b\|x)$ leaks b but <u>is</u> OWF

=> bad commitment scheme

Goldreich-Levin

Hard to invert

One Way Function $\xrightarrow{GL}$ "Hidding" Function

We can hide one bit

↳ Pb: some bits may not be secrets:

E.g: $f(b \| x) := b \| SHA(x)$

$f(b \| x)$ leaks b but is OWF

=> bad commitment scheme

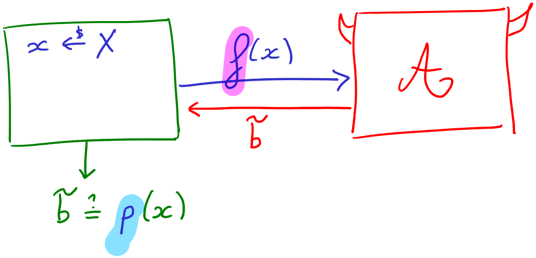
More precisely

= hardcore bit

Goldreich-Levin

Hardcore predicate

Two functions f and p: "Hard to guess $p(x)$ given $f(x)$":



Goldreich-Levin

Theorem (Goldreich-Levin)

Let f be an arbitrary one-way function, and let $f'(x,r) := (f(x),r)$ where $|x| = |r|$. Let $p(x,r) := \oplus_i(x_i r_i)$. Then p is a hardcore predicate for f'.

Proof sketch: By contradiction: For simplicity, assume there exists $\mathcal{A}(f'(x))$ that always guesses $g(x)$ correctly. Then, we can use $\mathcal{A}$ to invert f:

?

Show how $\mathcal{A}$ can be used to recover x from $y := f(x)$.

Goldreich-Levin

Theorem (Goldreich-Levin)

Let f be an arbitrary one-way function, and let $f'(x,r) := (f(x),r)$ where $|x| = |r|$. Let $p(x,r) := \oplus_i(x_i r_i)$. Then p is a hardcore predicate for f'.

Proof sketch: By contradiction: For simplicity, assume there exists $\mathcal{A}(f'(x))$ that always guesses $g(x)$ correctly. Then, we can use $\mathcal{A}$ to invert f:

Show how $\mathcal{A}$ can be used to recover x from $y := f(x)$.
We can recover x bit-by-bit:

1. First bit is $\mathcal{A}(y, 10\dots0) = g(x, 10\dots0) = x_1 \times 1 + x_2 \times 0 + \dots x_n \times 0 = x_1$
2. Second bit is $\mathcal{A}(y, 010\dots0)$, $\dots$
3. $\dots$
4. Last bit is $\mathcal{A}(y, 0\dots01)$ □

Goldreich Levin

Full proof: see *Foundation of Cryptography, Volume 1*, Oded Goldreich.