# Zero-Knowledge Proofs for Classical Planning Problems

Augusto B. Corrêa, Clemens Büchner, Remo Christen

University of Basel, Switzerland

AAAI 2023

# Classical Planning

propositional classical planning tasks:

- propositional variables: $\{x, y\}$
- initial state: $\{x, \neg y\}$
- goal: $\{y\}$
- actions:

  $a_1$: $pre = \{\}$
  $\quad\quad eff = \{\neg x\}$

  $a_2$: $pre = \{\neg x\}$
  $\quad\quad eff = \{y\}$
- plan: $\pi = \langle (s_0, a_1, s_1), (s_1, a_2, s_2) \rangle$

# Planning without Revealing Solutions

I know a plan for some planning task.
But I won't show you.

Can I convince you that I really know a plan?
How do I do that without revealing anything about the plan?

# Planning without Revealing Solutions

I know a plan for some planning task.
But I won't show you.

Can I convince you that I really know a plan?
How do I do that without revealing anything about the plan?

Zero-Knowledge Proofs!

# Formally

Prover:

- claims to have a plan $\pi$ (somehow)
- wants to prove this without revealing anything about $\pi$

Verifier:

- wants to check that Prover is not lying
- do it efficiently

# Formally

Prover:

- claims to have a plan $\pi$ (somehow)
- wants to prove this without revealing anything about $\pi$

Verifier:

- wants to check that Prover is not lying
- do it efficiently

bounded plan existence:
Given a planning task $\Pi$ and $k \in \mathbb{N}$ such that $k \leq \mathit{poly}(|\Pi|)$, is there a plan $\pi$ for $\Pi$ with $|\pi| \leq k$?

# Formally

Prover:

- claims to have a plan $\pi$ (somehow)
- wants to prove this without revealing anything about $\pi$

Verifier:

- wants to check that Prover is not lying
- do it efficiently

bounded plan existence:
Given a planning task $\Pi$ and $k \in \mathbb{N}$ such that $k \leq \mathit{poly}(|\Pi|)$, is there a plan $\pi$ for $\Pi$ with $|\pi| \leq k$?

- no limitation on $k$ if you reduce to QBF protocols
- Prover needs to stronger than in our case; algebraic

# Properties

completeness: Verifier will never reject plan of an honest Prover

soundness: Verifier might be fooled with low probability

zero-knowledge: Verifier learns nothing about the plan

efficiency: protocol executed efficiently by Verifier and Prover

# Protocol Overview

overview of all steps:

- Prover obfuscates $\Pi$
- Verifier chooses between
  - verifying if obfuscation was done correctly
  - verifying one transition of the claimed plan $\pi$
- if the chosen verification succeeds, Verifier accepts $\pi$ otherwise, rejects

Prover communicates via encrypted messages

- opened using some key

- propositional variables: $\{x, y\}$
- initial state: $\{x, \neg y\}$
- goal: $\{y\}$
- actions:

$a_1$: $\quad pre = \{\}$
$\quad\quad eff = \{\neg x\}$

$a_2$: $\quad pre = \{\neg x\}$
$\quad\quad eff = \{y\}$

- plan: $\pi = \langle (s_0, a_1, s_1), (s_1, a_2, s_2) \rangle$

# Prover Obfuscates $\Pi$

transform $\Pi$ into $\hat{\Pi}$ such that it is hard to map $\hat{\Pi}$ back to $\Pi$

| variables | initial state | goal | actions | |
|---|---|---|---|---|
| $\{x, y\}$ | $\{x, \neg y\}$ | $\{y\}$ | $\{\} \rightarrow \{\neg x\}$ | $\{\neg x\} \rightarrow \{y\}$ |

# Prover Obfuscates $\Pi$

transform $\Pi$ into $\hat{\Pi}$ such that it is hard to map $\hat{\Pi}$ back to $\Pi$

| variables | initial state | goal | actions | |
|---|---|---|---|---|
| $\{x, y\}$ | $\{x, \neg y\}$ | $\{y\}$ | $\{\} \rightarrow \{\neg x\}$ | $\{\neg x\} \rightarrow \{y\}$ |

make actions indistinguishable:

| | | | | |
|---|---|---|---|---|
| $\{x, y, z\}$ | $\{x, \neg y, \neg z\}$ | $\{y\}$ | $\{\neg z\} \rightarrow \{\neg x\}$ | $\{\neg x\} \rightarrow \{y\}$ |

# Prover Obfuscates $\Pi$

transform $\Pi$ into $\hat{\Pi}$ such that it is hard to map $\hat{\Pi}$ back to $\Pi$

| variables | initial state | goal | actions | |
|---|---|---|---|---|
| $\{x, y\}$ | $\{x, \neg y\}$ | $\{y\}$ | $\{\} \rightarrow \{\neg x\}$ | $\{\neg x\} \rightarrow \{y\}$ |
| make actions indistinguishable: | | | | |
| $\{x, y, z\}$ | $\{x, \neg y, \neg z\}$ | $\{y\}$ | $\{\neg z\} \rightarrow \{\neg x\}$ | $\{\neg x\} \rightarrow \{y\}$ |
| change variable labels and sign: | | | | |
| $\{\chi, \upsilon, \zeta\}$ | $\{\chi, \upsilon, \neg\zeta\}$ | $\{\neg\upsilon\}$ | $\{\neg\zeta\} \rightarrow \{\neg\chi\}$ | $\{\neg\chi\} \rightarrow \{\neg\upsilon\}$ |

# Prover Obfuscates $\Pi$

transform $\Pi$ into $\hat{\Pi}$ such that it is hard to map $\hat{\Pi}$ back to $\Pi$

| variables | initial state | goal | actions | |
|---|---|---|---|---|
| $\{x, y\}$ | $\{x, \neg y\}$ | $\{y\}$ | $\{\} \rightarrow \{\neg x\}$ | $\{\neg x\} \rightarrow \{y\}$ |
| make actions indistinguishable: | | | | |
| $\{x, y, z\}$ | $\{x, \neg y, \neg z\}$ | $\{y\}$ | $\{\neg z\} \rightarrow \{\neg x\}$ | $\{\neg x\} \rightarrow \{y\}$ |
| change variable labels and sign: | | | | |
| $\{\chi, \upsilon, \zeta\}$ | $\{\chi, \upsilon, \neg\zeta\}$ | $\{\neg\upsilon\}$ | $\{\neg\zeta\} \rightarrow \{\neg\chi\}$ | $\{\neg\chi\} \rightarrow \{\neg\upsilon\}$ |
| add canonical initial and goal states: | | | | |
| (omitted) | | | | |

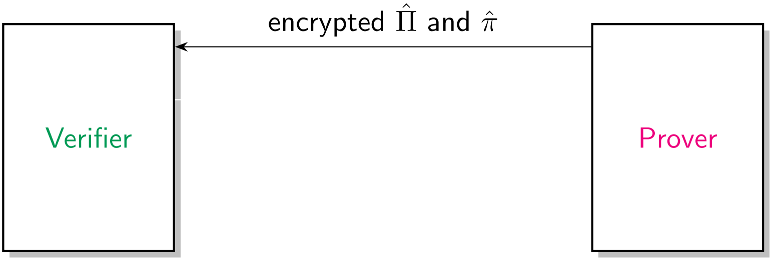
# Prover Obfuscates $\Pi$

transform $\Pi$ into $\hat{\Pi}$ such that it is hard to map $\hat{\Pi}$ back to $\Pi$

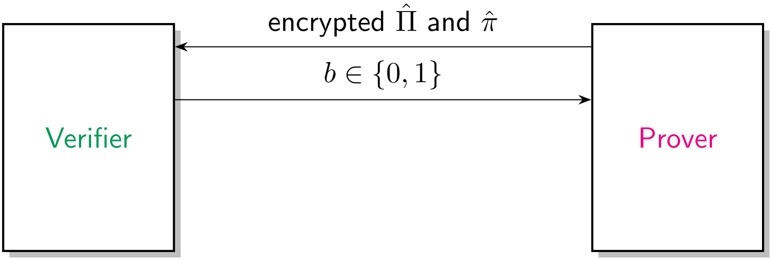
| variables | initial state | goal | actions | |
|---|---|---|---|---|
| $\{x, y\}$ | $\{x, \neg y\}$ | $\{y\}$ | $\{\} \rightarrow \{\neg x\}$ | $\{\neg x\} \rightarrow \{y\}$ |
| make actions indistinguishable: | | | | |
| $\{x, y, z\}$ | $\{x, \neg y, \neg z\}$ | $\{y\}$ | $\{\neg z\} \rightarrow \{\neg x\}$ | $\{\neg x\} \rightarrow \{y\}$ |
| change variable labels and sign: | | | | |
| $\{\chi, \upsilon, \zeta\}$ | $\{\chi, \upsilon, \neg\zeta\}$ | $\{\neg\upsilon\}$ | $\{\neg\zeta\} \rightarrow \{\neg\chi\}$ | $\{\neg\chi\} \rightarrow \{\neg\upsilon\}$ |
| add canonical initial and goal states: | | | | |
| (omitted) | | | | |

- change plan $\pi$ into $\hat{\pi}$
- done in polynomial time (details in the paper)

# Protocol



# Protocol

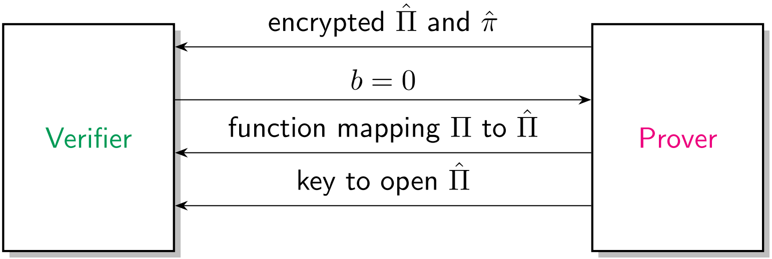


# Verifier's Random Choice

Verifier chooses a random value $b \in \{0, 1\}$

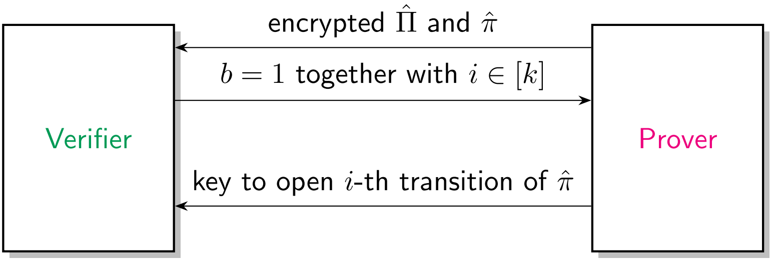
- if $b = 0$: verifier checks the transformation $\Pi \to \hat{\Pi}$
- if $b = 1$: verifier checks one randomly chosen transition of $\hat{\pi}$

# Protocol: $b = 0$



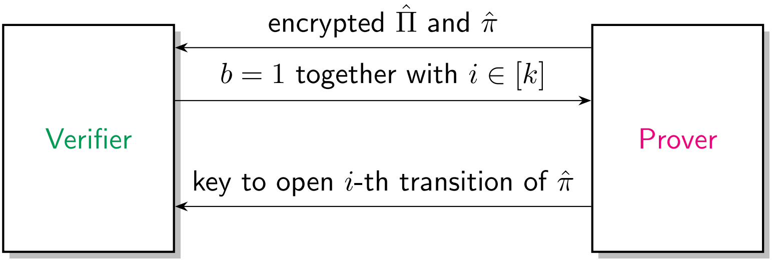

If sent function indeed maps $\Pi$ into $\hat{\Pi}$, Verifier **accepts**.
Otherwise, **rejects**.

# Protocol: $b = 1$




If transition $(s_{i-1}, a_i, s_i)$ is valid, Verifier **accepts**.
Otherwise, **rejects**.

# Protocol: $b = 1$




If transition $(s_{i-1}, a_i, s_i)$ is valid, Verifier **accepts**.
Otherwise, **rejects**.

in practice, a little bit harder:

- check if $s_0$ and $s_k$ are canonical initial and goal states
- check if $a_i$ is an action of $\hat{\Pi}$

# Properties (Revisited)

completeness: Verifier will never reject plan of an honest Prover

soundness: Verifier might be fooled with probability $1 - \frac{1}{2k}$

- run protocol multiple times to reduce this probability

zero-knowledge: Verifier learns nothing about the plan

- see paper for details

efficiency: protocol executed efficiently by Verifier and Prover

- constant number of messages per execution
- only polynomial-time computation

# Conclusion

zero-knowledge proofs for classical planning problems:

- how to prove that you have a plan without revealing it
- works for problems with polynomially long plans
- executed efficiently