

# Zero Knowledge Proofs

Sushmita Ruj

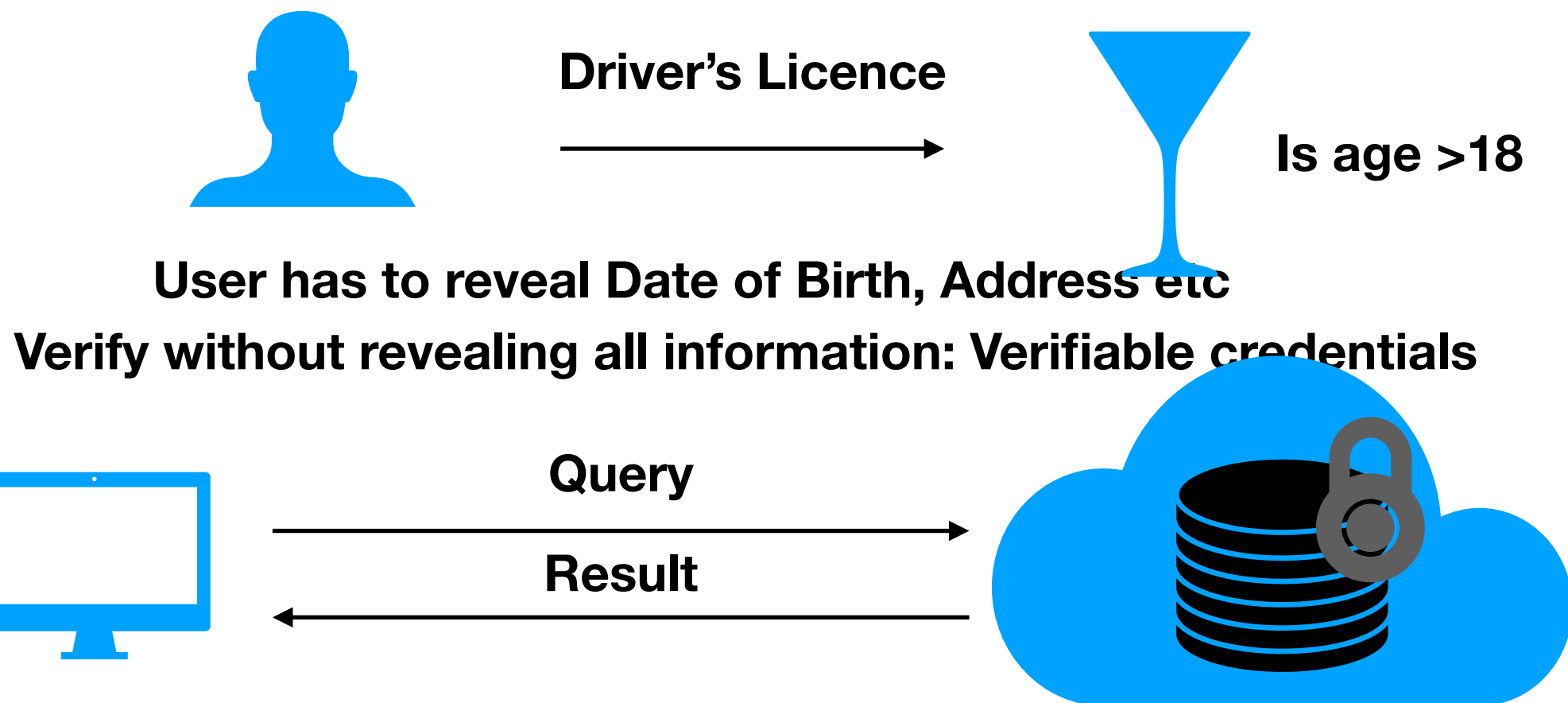
# Recap

- Blockchain History
- Blockchain Basics
- Attacks on Blockchains
- Privacy
- Open Questions

# This Lecture

- Need for Zero-knowledge proofs
- ZKP Foundations
- SNARKS: Building Blocks and Design
- ZKP Applications Conclusion and open problems

# Proofs $\leftrightarrow$ Verifiability

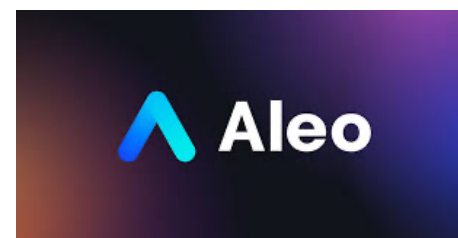


**Prove I have enough money in my bank account to buy a car worth \$X,  
without revealing my bank balance?**

**How do I verify a ML algorithm generates the correct models**

**Proofs should be correct and efficient, efficient generation, verification and short**

# Blockchain Companies building on ZKP



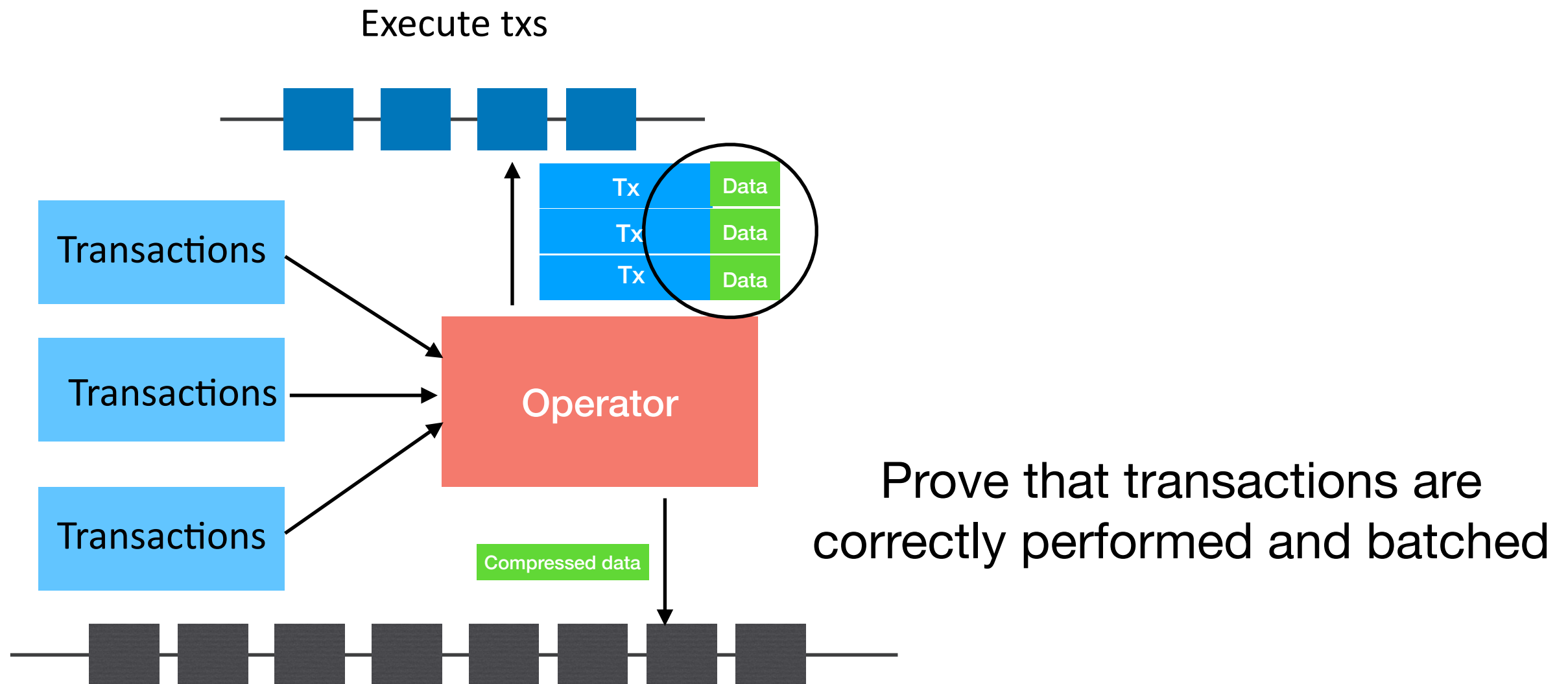
# Application 1: Privacy Preserving Transactions

- Hide Value of Transaction
- Hide sender/receiver
- Zcash, Tornado Cash (attacked), Aleo

# Application 1: Privacy Preserving Transactions

- Hide Value of Transaction
- Hide sender/receiver
- Zcash, Tornado Cash (attacked)

# Application 2: Rollups for Scalability





# Zero-Knowledge Proofs Foundations

# Proofs

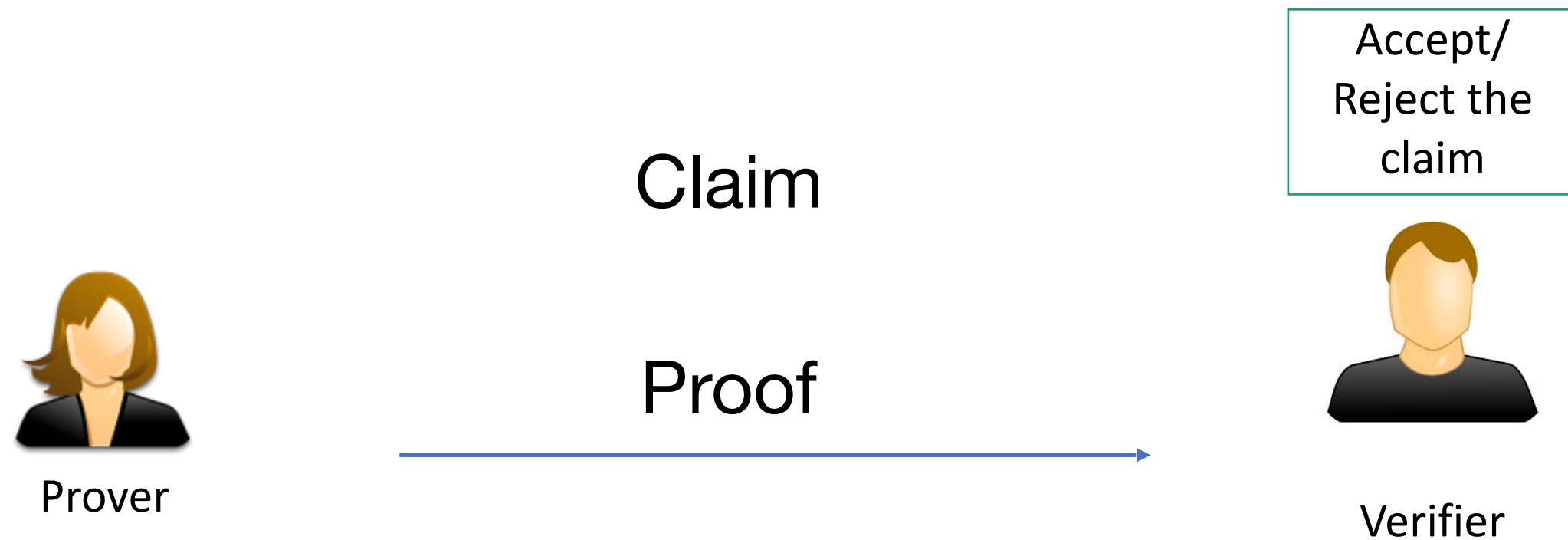
- A method to establish the truth

- Legal
- Authoritative
- Scientific
- Philosophical
- Mathematical

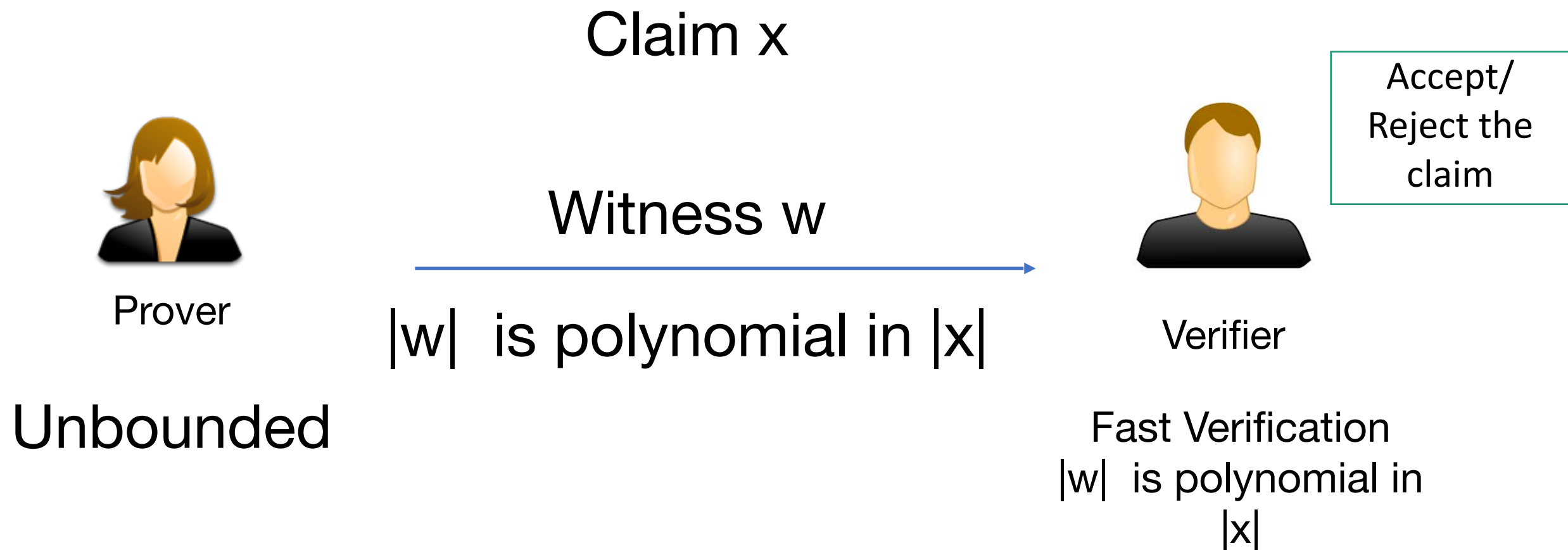
Axioms  $\rightarrow \rightarrow \dots \rightarrow$  Propositions

- Probabilistic, Interactive

# Proofs

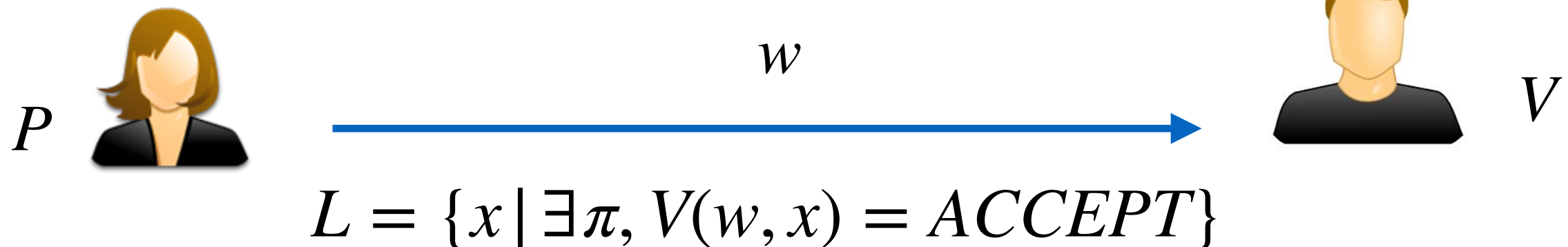


# Efficient Proofs (NP-Proofs)



# Proof Systems

$P$  wants to prove the following statement  
 $x \in L$  for some language  $L \subseteq \Sigma^*$



Definition: A proof system for membership in  $L$  is an algorithm  $V$ , such that  $\forall x$  :

**Completeness:** If  $x \in L$ , then  $\exists w, V(w, x) = \text{ACCEPT}$

**Soundness:** If  $x \notin L$ , then  $\forall w, V(w, x) = \text{REJECT}$

Babai: Trading Group Theory for Randomness, 1985

Goldwasser-Micali-Rackoff: The Knowledge Complexity of Interactive Proof-Systems, 1985

# NP Proof Systems

- Efficient Verification meaning polynomial time verification

Definition: A **NP proof system** for membership in  $L$  is an algorithm  $V$ , such that  $\forall x$  :

**Completeness:** If  $x \in L$ , then

$\exists w, (|w| = \text{poly}(|x|)), V(w, x) = \text{ACCEPT}$

**Soundness:** If  $x \notin L$ , then  $\forall w, V(w, x) = \text{REJECT}$

**Efficiency:**  $V(w, x)$  halts after at most  $\text{poly}(|x|)$  steps

- $V$ 's running time is measured in terms of  $|x|$ , length of input  $x$
- $\text{poly}(|x|) = |x|^c$ , for some constant  $c$
- Necessarily,  $|\pi| = \text{poly}(|x|)$

# Proofs leaking secrets

$$QR_N = \{x \mid x \text{ is a quadratic residue modulo } N\}$$

Prove that  $x \in QR_N$

$P$



I know  $w$  and  
I can prove it

$w$

$V$



Checks  $x \stackrel{?}{=} w^2 \pmod N$

Reveals information to  $V$ , that  $V$  could not have computed

# “No knowledge is leaked”

- $V$  didn't learn  $w$
- $V$  didn't learn any bit of  $w$
- $V$  didn't learn any information about  $w$
- $V$  didn't learn any information at all (except  $x \in QR_N$ )

When would we say that  $V$  *did* learn something?

If following the interaction  $V$  could compute something it could have not computed without it!

Zero-knowledge: whatever is computed following the interaction could have been computed without it.

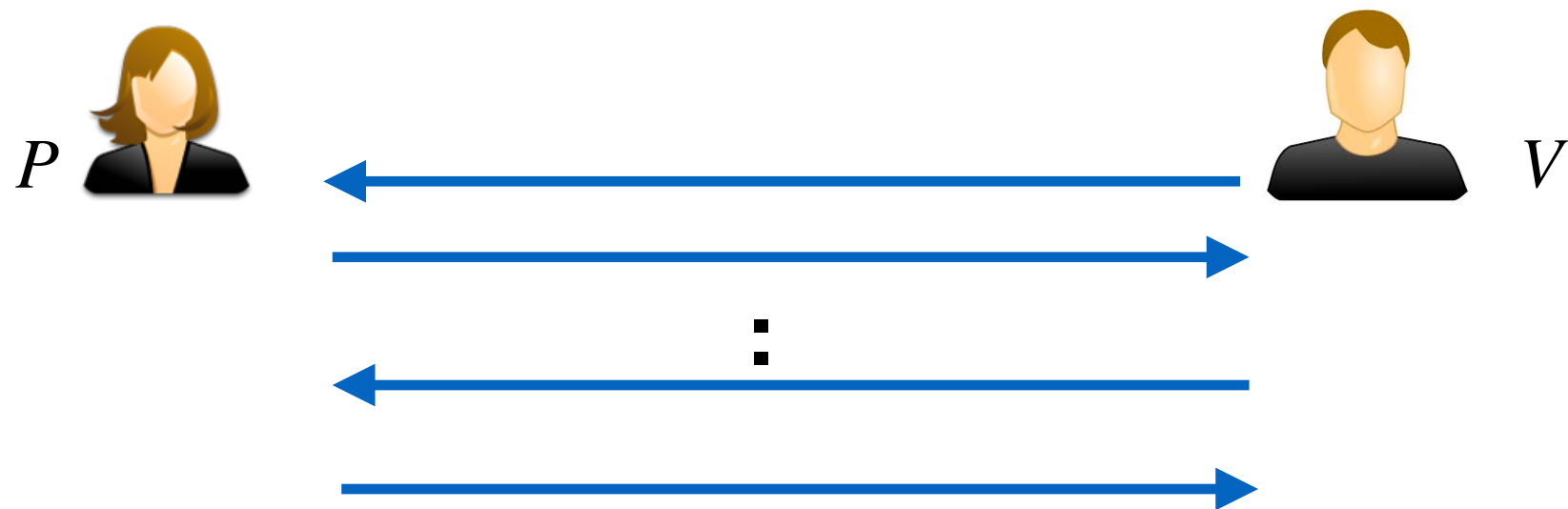
Goldreich-Micali-Wigderson 1991: Every set in NP has a Zero-Knowledge Interactive proof



# Paradoxical Proofs

- Zero-knowledge proofs: A proof that does not leak information
- Probabilistically checkable proofs: Proofs need not be read in their entirety

# Interactive and Probabilistic Proofs



- Interactive: Verifier engages in an interaction rather than reading the proof
- Verifier is randomised and can make errors with small probability

# Interactive Proofs



Completeness:  $P$  convinces  $V$  that the  $x \in L$

Computational Soundness: A Cheating prover can't convince  $V$  to accept  $x \notin L$  (except with negligible probability)

# Interactive Proofs for QR

$$QR_N = \{x \mid x \text{ is a quadratic residue modulo } N\}$$

Prove that  $x \in QR_N$

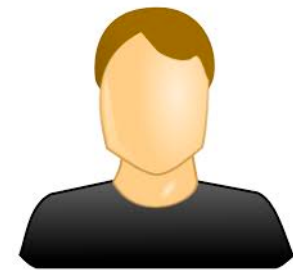
$P$



I know  $w$  and  
I can prove it

$w$

$V$

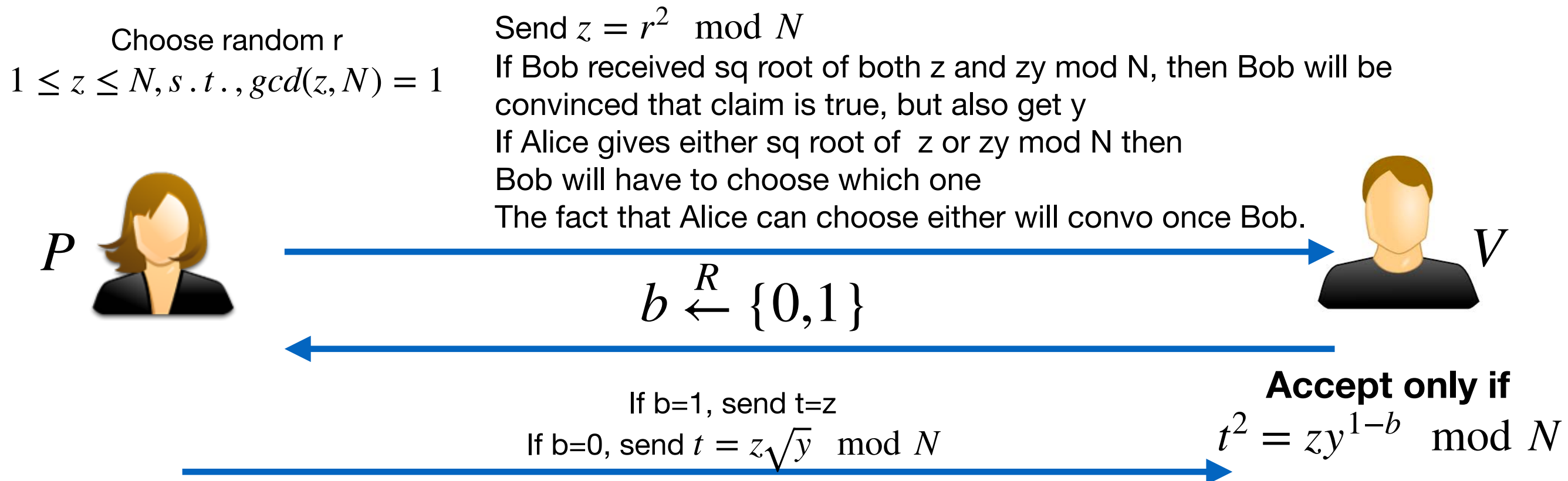


Checks  $x \stackrel{?}{=} w^2 \pmod N$

Reveals information to  $V$ , that  $V$  could not have computed

# Example

Prove Alice knows  $y$  such that  $y = x^2 \pmod N$



- Completeness: If Claim is true then  $V$  Accepts
- Soundness: If claim is false  $\forall$  Provers  $P, Pr[V \text{ accepts}] \leq 1/2$
- Run multiple times (say 1000) , the above probability negl

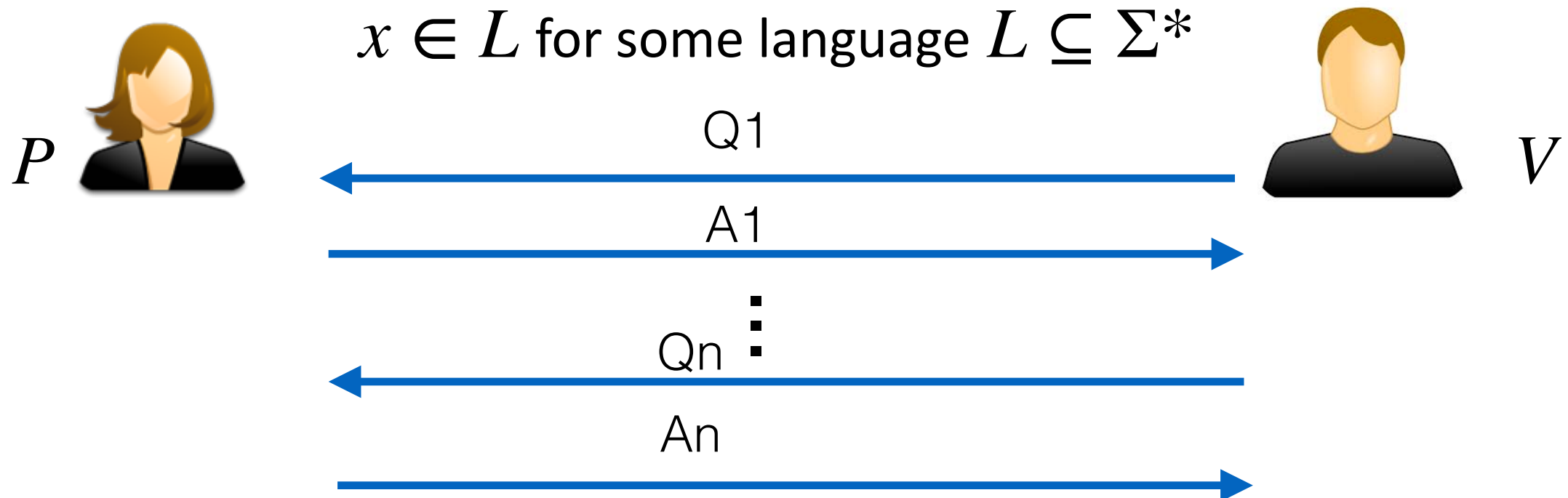
# Important idea

- Multiple rounds
- Proof consists of 2 parts: seeing either part on its own conveys no information; seeing both parts imply 100% correctness.
- Verifier chooses at random which of the two parts of the proof he wants the prover to give him. The ability of the prover to provide either part, convinces the verifier

# Interactive Proofs (IP)

$P$  wants to prove the following statement

$x \in L$  for some language  $L \subseteq \Sigma^*$



$V$  is a Probabilistic Polynomial time (PPT)

ACCEPT/REJECT

For any common input  $x$ , let:

$$Pr[(P, V) \text{ accepts } x] \stackrel{\Delta}{=} Pr_r[(P, V)(x, r) = ACCEPT]$$

Babai: Trading Group Theory for Randomness 1985

Goldwasser-Micali-Rackoff (GMR85): The Knowledge Complexity of Interactive Proof-Systems, 1985

# Interactive Proof Systems

Definition[GMR85]: An **Interactive *proof system*** for membership in  $L$  is a PPT algorithm  $V$  and a function  $P$ , such that  $\forall x$  :

Completeness: If  $x \in L$ , then,  $Pr[(P, V) \text{ accepts } x] = 1$

Soundness: If  $x \notin L$ , then  $\forall$  *cheating provers*  $P^*$  s.t.,  $Pr[(P^*, V) \text{ accepts } x]$  is negl

Completeness and soundness can be bounded by any  $c : \mathbb{N} \rightarrow [0,1]$  and  $s : \mathbb{N} \rightarrow [0,1]$  as long as

- $c(|x|) \geq 1/2 + 1/\text{poly}(|x|)$
  - $s(|x|) < 1/2 - 1/\text{poly}(|x|)$
- $\text{poly}(|x|)$  Independent repetitions implies  $c(|x|) - s(|x|) \geq 1 - 2^{-\text{poly}(|x|)}$

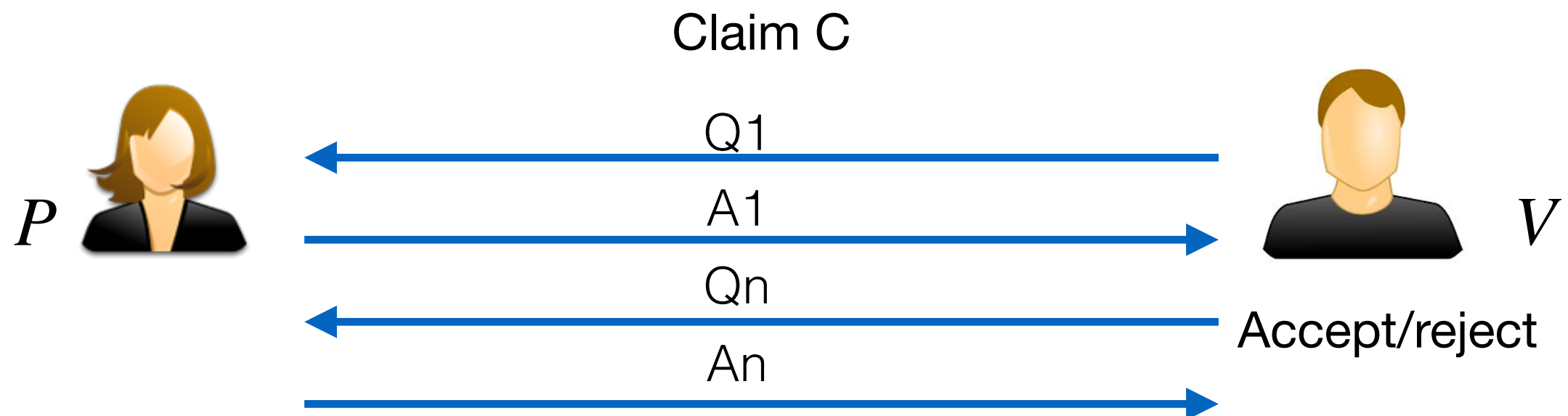
**Definition:** class of languages  $IP = \{L \text{ for which there is an interactive proof}\}$



# Zero Knowledge

- For True Statements and every verifier
- What a verifier can compute before interaction
- The Verifier can compute after interaction

# Verifier's View



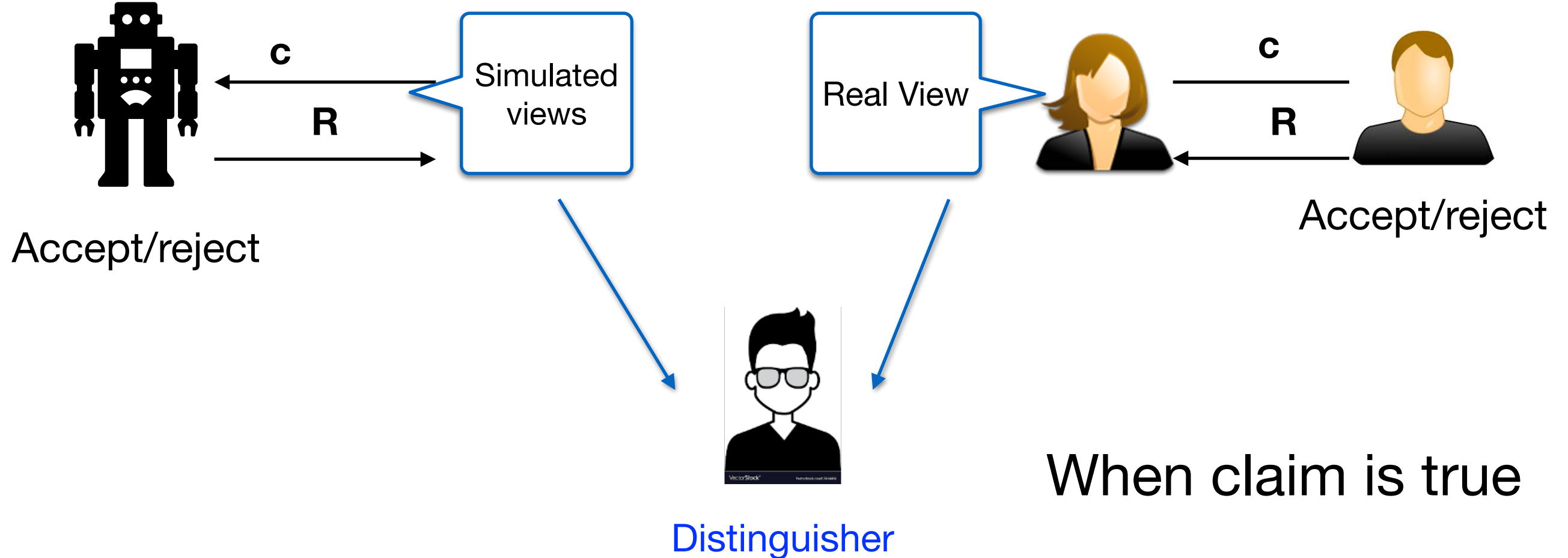
After interaction  $V$  learned  $C$  is true/false

A view of interaction includes coin toss + transcript

$\text{View}_V[P, V] = \{Q_1, A_1, Q_2, A_2, \dots, Q_n, A_n, \text{coins of } V\}$

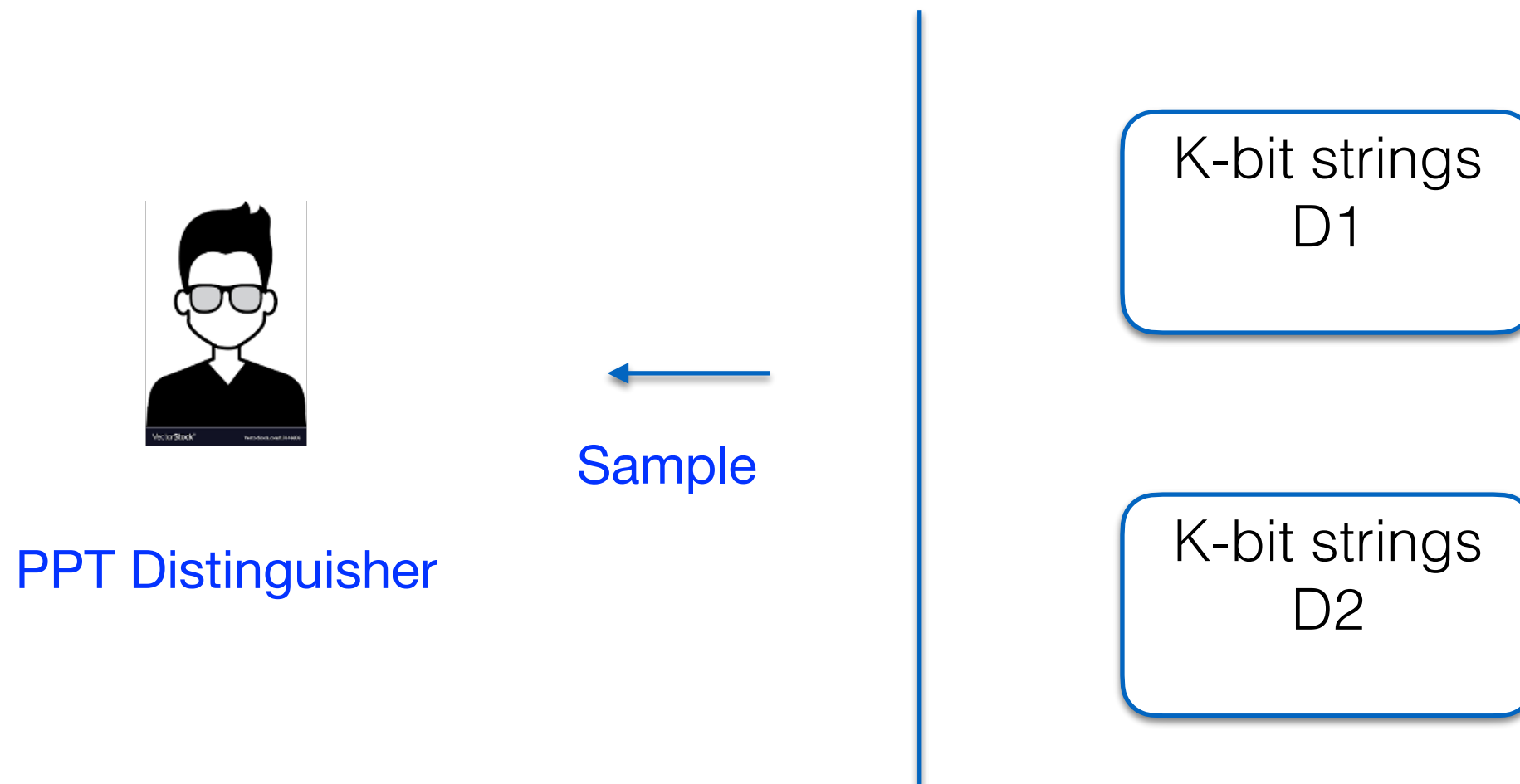
Probability distribution over coins of  $P$  and  $V$

# Simulation Paradigm



Distinguisher's view is nothing new, if he could have simulated on its own s.t  
'simulated view' and 'real-view' are **computationally-Indistinguishable**

# Computational Indistinguishability



If no “distinguisher” can tell apart two different probability distributions they are “effectively the same”.

For all distinguisher algorithms  $D$ , even after receiving a polynomial number of samples from  $D_b$ ,  $\text{Prob}[D \text{ guesses } b] < 1/2 + \text{negl}$

# Zero-Knowledge Proof

An Interactive Protocol  $(P, V)$  is zero-knowledge for a language  $L$  if there exists a **PPT** algorithm  $\text{Sim}$  (a simulator) such that **for every  $x \in L$** , the following two probability distributions are **poly-time** indistinguishable:

1.  $\text{view}_V(P, V)[x, 1^\lambda] = \{(Q_1, A_1, Q_2, A_2, \dots, \text{coins of } V)\}$
2.  $\text{Sim}(x, 1^\lambda)$  (over coins of  $V$  and  $P$ )

Definition:  $(P, V)$  is a zero-knowledge interactive protocol if it is complete, sound and zero-knowledge

# If Verifier is Not Honest

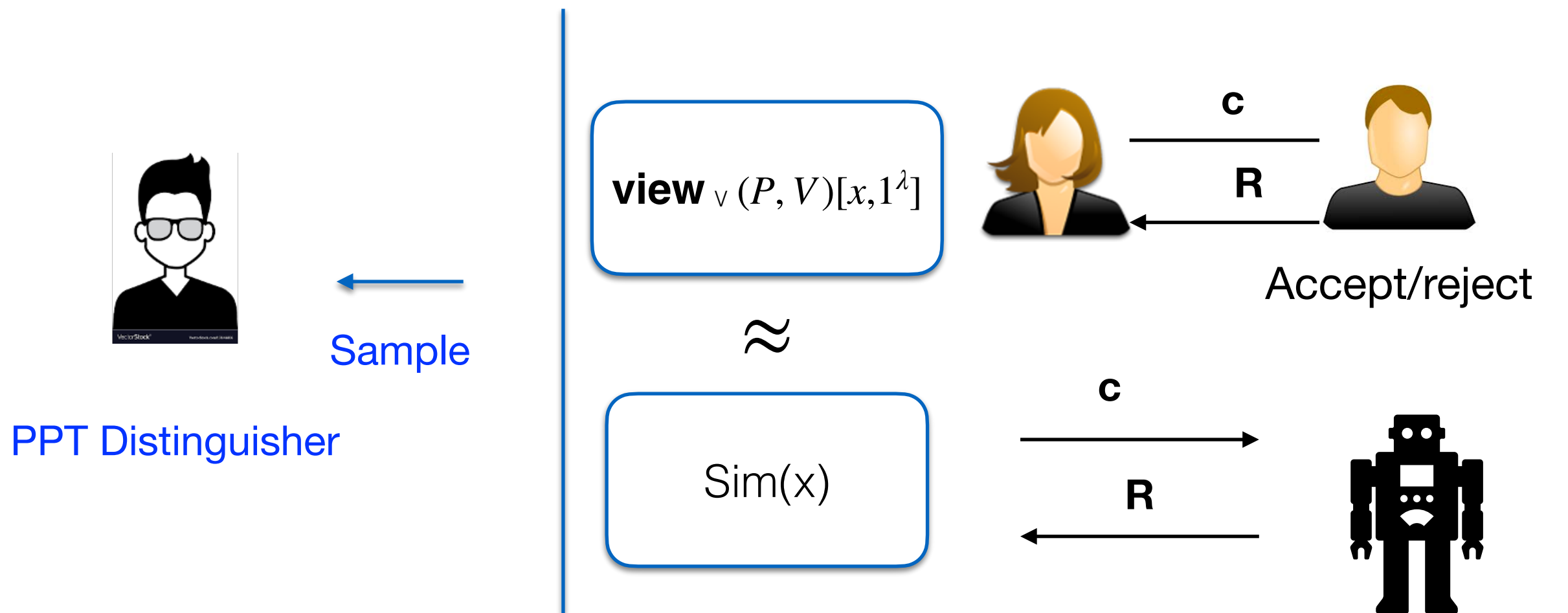
An Interactive Protocol  $(P, V)$  is honest verifier zero-knowledge for a language  $L$  if there exists a PPT algorithm  $\text{Sim}$  s.t. for every  $x \in L$ ,

$$\text{view}_V(P, V)[x, 1^\lambda] \approx \text{Sim}(x, 1^\lambda)$$

An Interactive Protocol  $(P, V)$  is zero-knowledge for a language  $L$  if for every PPT  $V^*$ , there exists a polynomial time algorithm  $\text{Sim}$  s.t. for every  $x \in L$ ,

$$\text{view}_V(P, V)[x, 1^\lambda] \approx \text{Sim}(x, 1^\lambda)$$

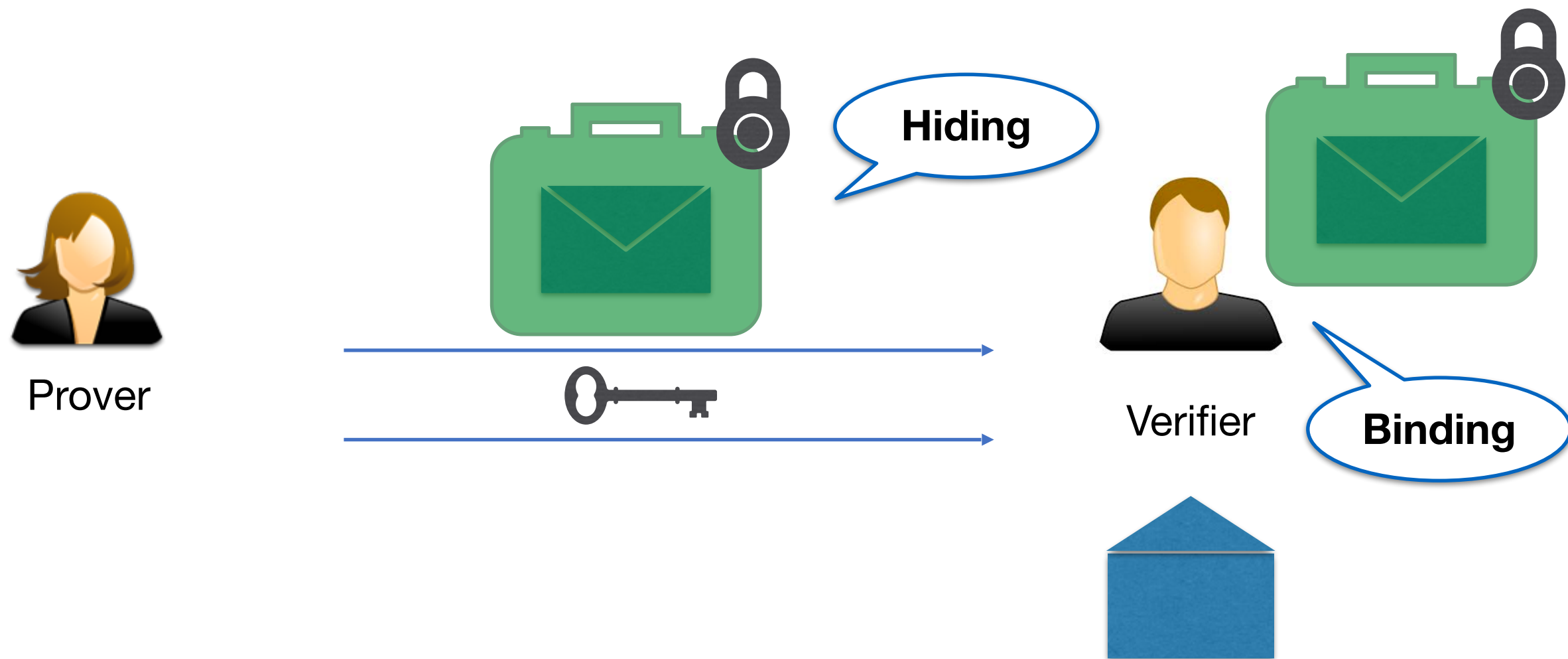
# Perfect ZK



Verifier's view can be exactly simulated

Simulated view  $\approx$  Real Views

# Commitment






# Zero Knowledge for all of NP

Theorem[GMW86, Naor]: If one-way functions exist, then every  $L$  in NP has computational ZK interactive proofs.

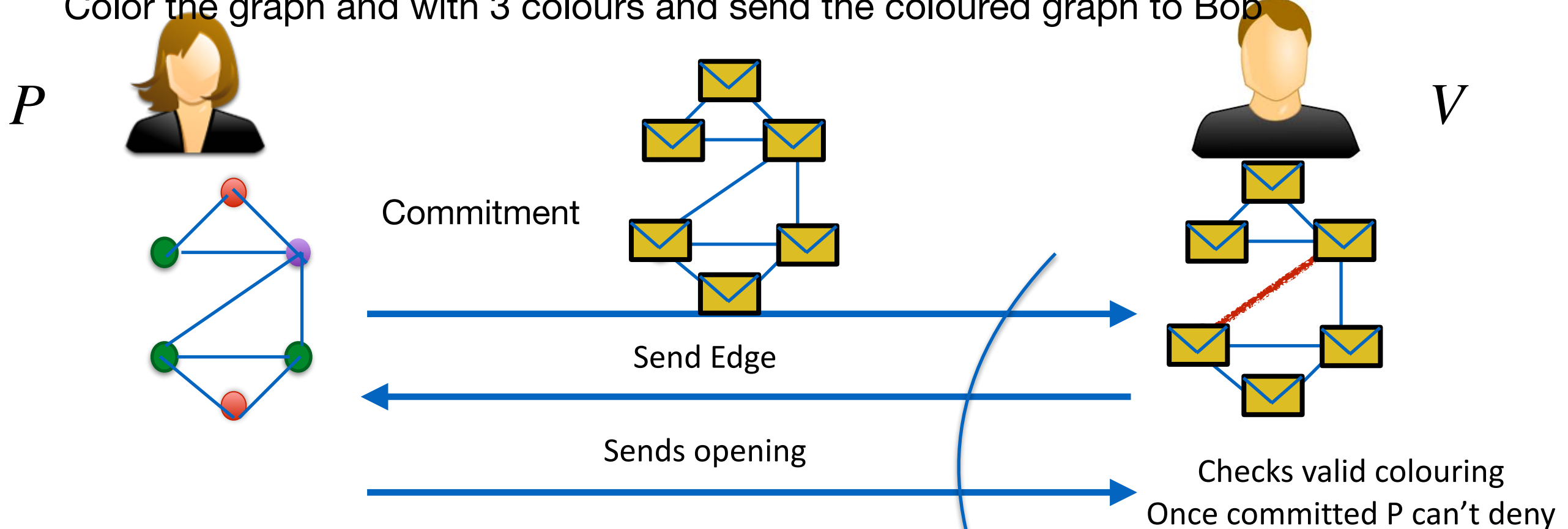
## Ideas of the proof:

1.[GMW87] Show that an NP-Complete Problem has a ZK interactive Proof if bit commitments exist

2.[Naor] One Way functions  bit commitment protocol exist

# ZK Proof for Graph Colouring

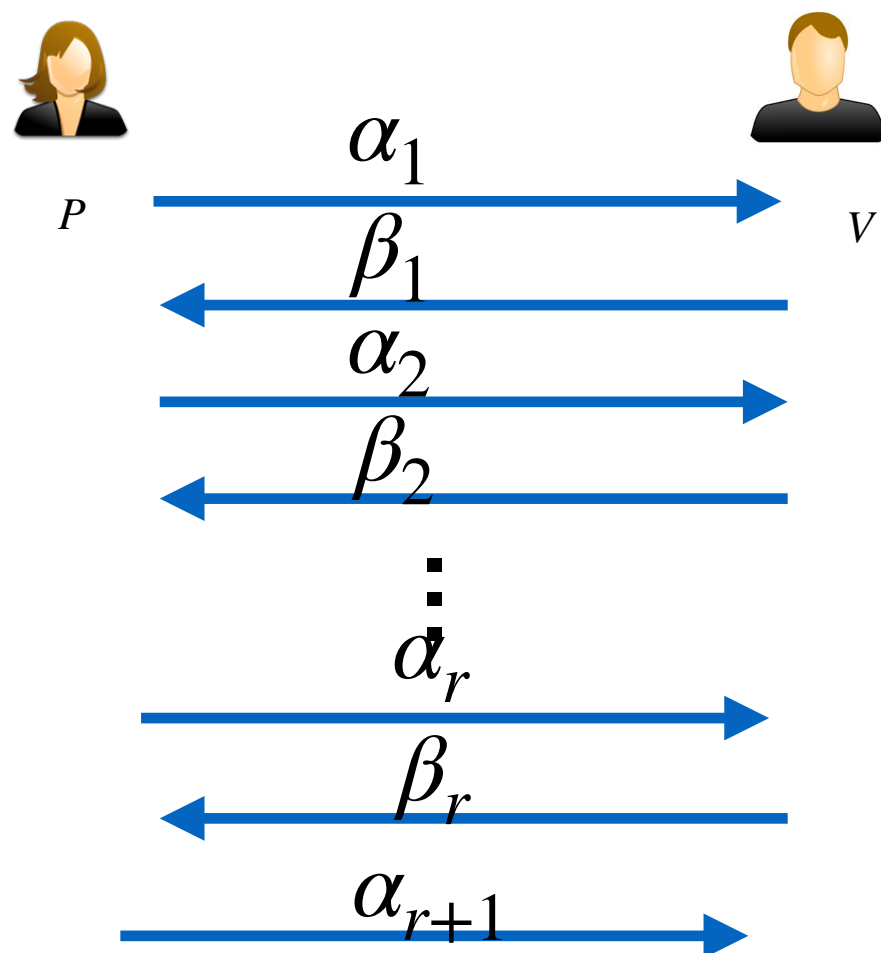
- Alice wants to prove that the nodes of a graph can be coloured with at most 3 colours, so that adjacent nodes have different colours
- Color the graph and with 3 colours and send the coloured graph to Bob



Repeat many times, every time with independent colour permutations

# Fiat-Shamir Transform

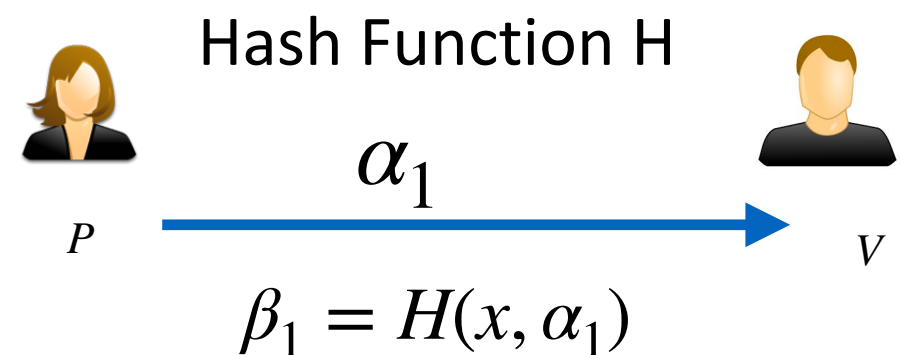
Public-coin interactive argument



Each  $\beta_i$  uniformly random

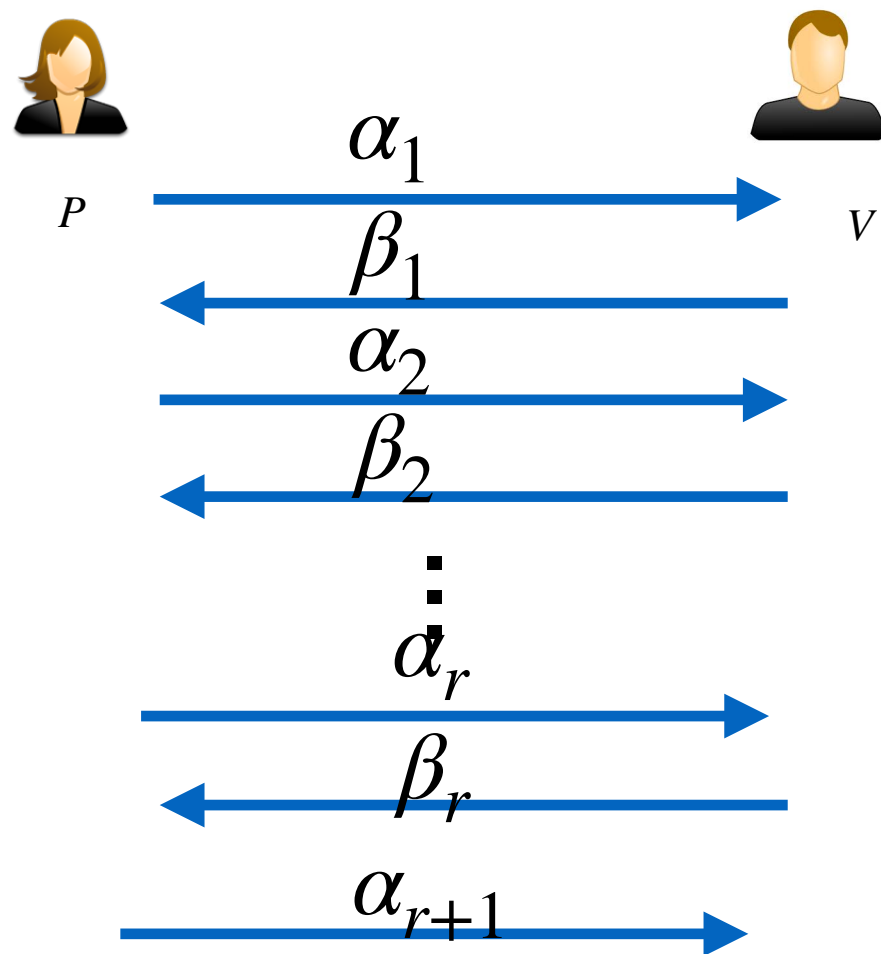


Public-coin non-interactive argument



# Fiat-Shamir Transform

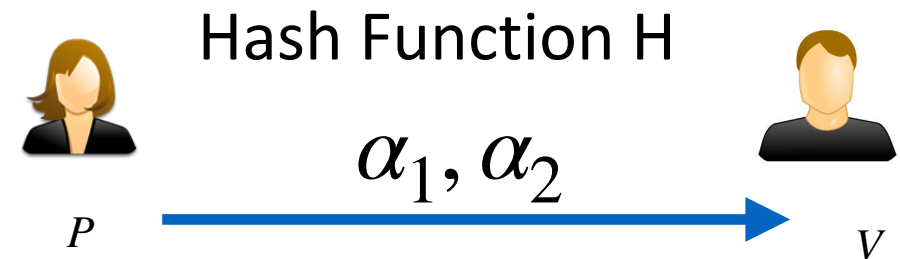
Public-coin interactive argument



Each  $\beta_i$  uniformly random



Public-coin interactive argument

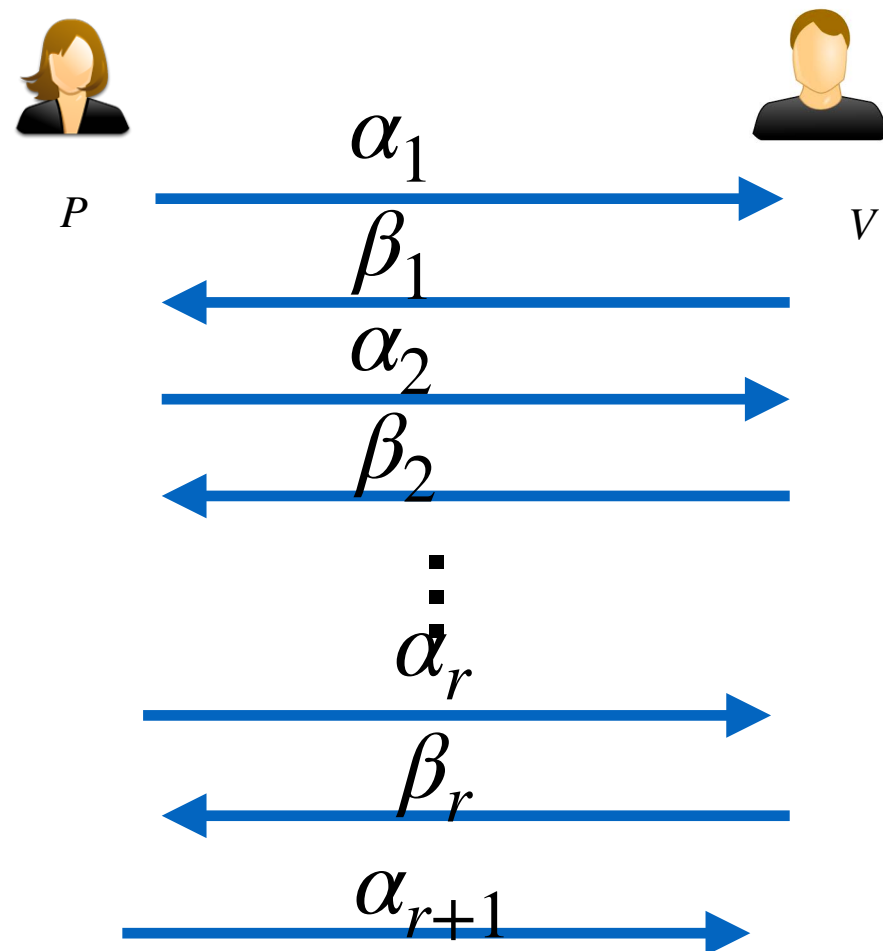


$$\beta_1 = H(x, \alpha_1)$$

$$\beta_2 = H(x, \alpha_1, \alpha_2)$$

# Fiat-Shamir Transform

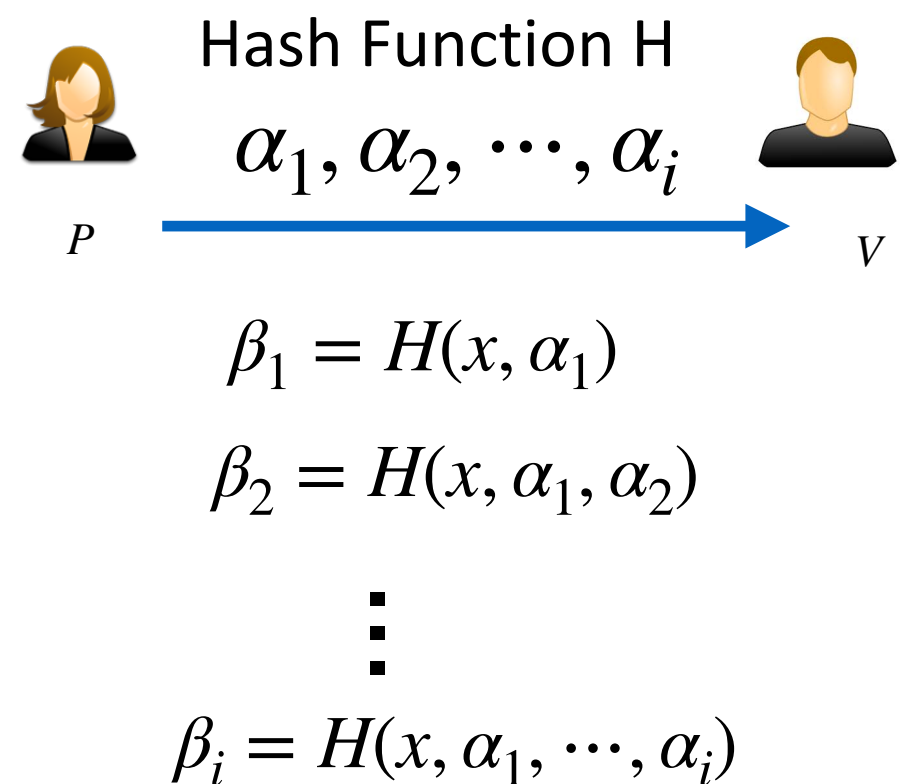
Public-coin interactive argument



Each  $\beta_i$  uniformly random



Non-Interactive argument



# ZK-SNARK

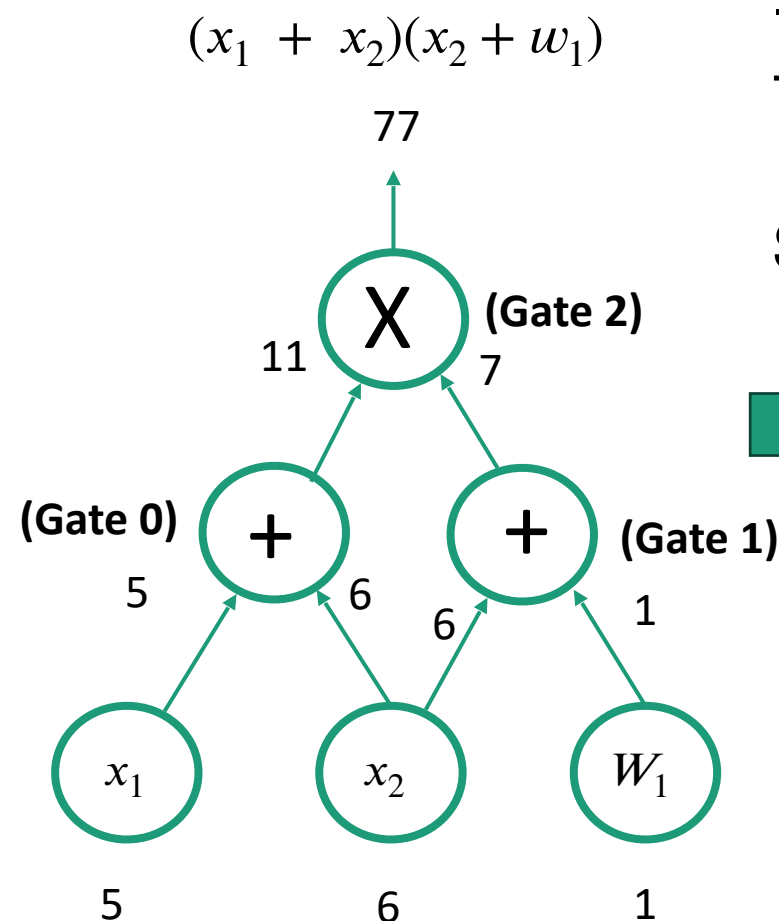
- **Succinct Non-Interactive Argument of Knowledge (SNARK)**
- **Succinct** : Proofs are short and can be verified much faster than verifying the claim from the original witness (e.g., the colouring)
- **Argument** is just a In a proof, "computationally sound proofs".
- In proofs: soundness holds against a computationally unbounded prover and in an argument, the soundness only holds against a polynomially bounded prover....
- **Knowledge**: Not just proving that there is a witness (e.g., colouring) but that the prover "knows" such a colouring

# Arithmetic Circuits

Fix a finite field  $\mathbb{F} = \{0, 1, \dots, p-1\}$

**Arithmetic circuit**  $\mathbb{C} : \mathbb{F}^n \rightarrow \mathbb{F}$

- A directed acyclic graph (DAG)
- Defines an n-variable polynomial with a evaluation formula



**Size of the circuit** is  $C = \# \text{gates}$

Input	5	6	1
Gate 0:	5	6	11
Gate 1:	6	1	7
Gate 2:	11	7	77

Left  
input

Right  
input

Output

Compile circuit to a computation trace (Arithmetization)

# Arithmetic Circuits: Example

Fix a finite field  $\mathbb{F} = \{0, 1, \dots, p-1\}$

**Arithmetic circuit**  $\mathbb{C} : \mathbb{F}^n \rightarrow \mathbb{F}$

- A directed acyclic graph (DAG)
- Defines an n-variable polynomial with a evaluation formula

**Size of the circuit** is  $C = \# \text{gates}$

$C_{\text{SHA}}(h, m) : \text{Outputs } 0 \text{ if } \text{SHA256}(m) = h \text{ and } \neq 0 \text{ otherwise}$

$C_{\text{SHA}}(h, m) = (h - \text{SHA256}(m)), \quad |C_{\text{SHA}}| \approx 20K$

$C_{\text{sig}}(\text{pk}, m, \sigma) : \text{Outputs } 0 \text{ if } \sigma \text{ is a valid ECDA signature with } P$   
 $\neq 0 \text{ otherwise}$



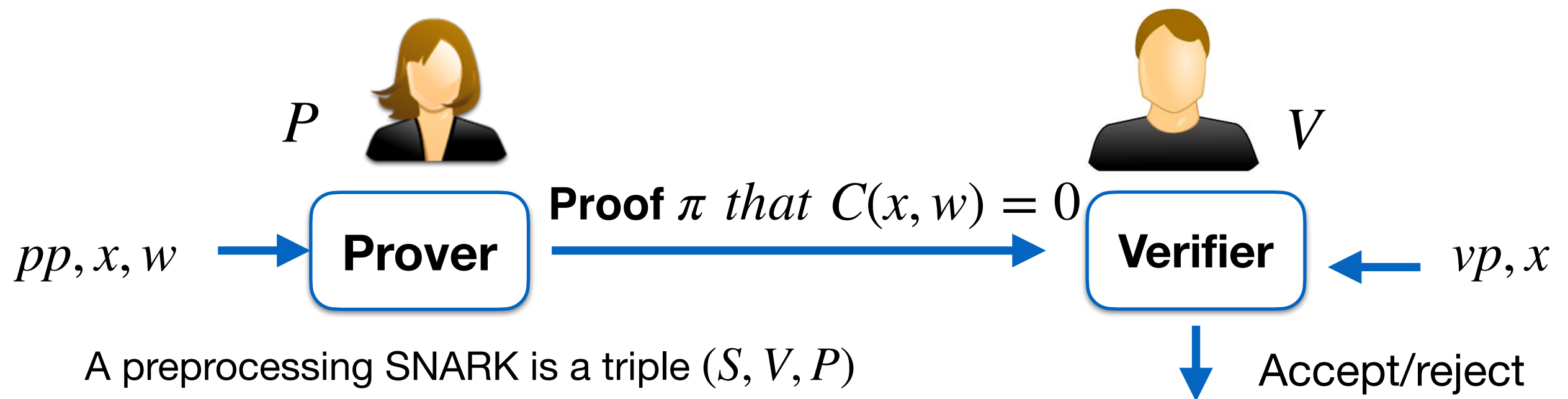
# Preprocessing NARKS

Public arithmetic Circuit  $C(x, w) \rightarrow \mathbb{F}$

$x$  is the public input

$w$  is the secret witness

Preprocessing/setup step :  $S(C) \rightarrow (pp, vp)$  (public parameters)



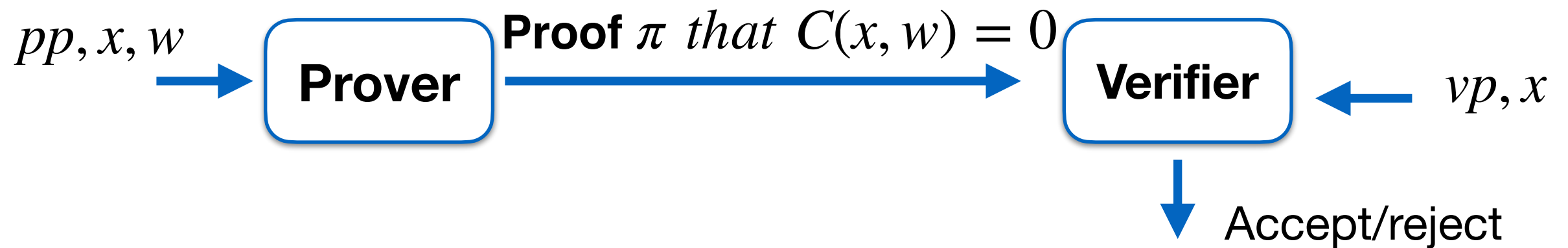
A preprocessing SNARK is a triple  $(S, V, P)$

$S(C) \rightarrow (pp, vp)$

$P(pp, x, w) \rightarrow \pi$

$V(vp, x, \pi) \rightarrow \text{Accept/Reject}$

# Definition



Completeness:

$$\forall x, w : C(x, w) = 0 \implies \Pr[V(vp, x, P(pp, x, w)) = \text{Accept}] = 1$$

**Knowledge Soundness:**

$V$  accepts  $\implies V$  knows  $w, s.t. C(x, w) = 0$ , and there exists an extractor which can extract  $w$  from  $P$

(Zero Knowledge (optional):  $C(pp, vp, x, w, \pi)$  reveals nothing about  $w$ )

# Succinct Arguments of Knowledge

A Succinct NARK is a triple  $(S, V, P)$  , such that

$S(C) \rightarrow (pp, vp)$  (Short parameters)

$P(pp, x, w) \rightarrow \pi$     Short:  $|\pi|$  is  $O(\log(|C|))$

$V(vp, x, \pi) \rightarrow \text{Accept/Reject}$

Verification is fast  $O_\lambda(|x|, \log(|C|))$

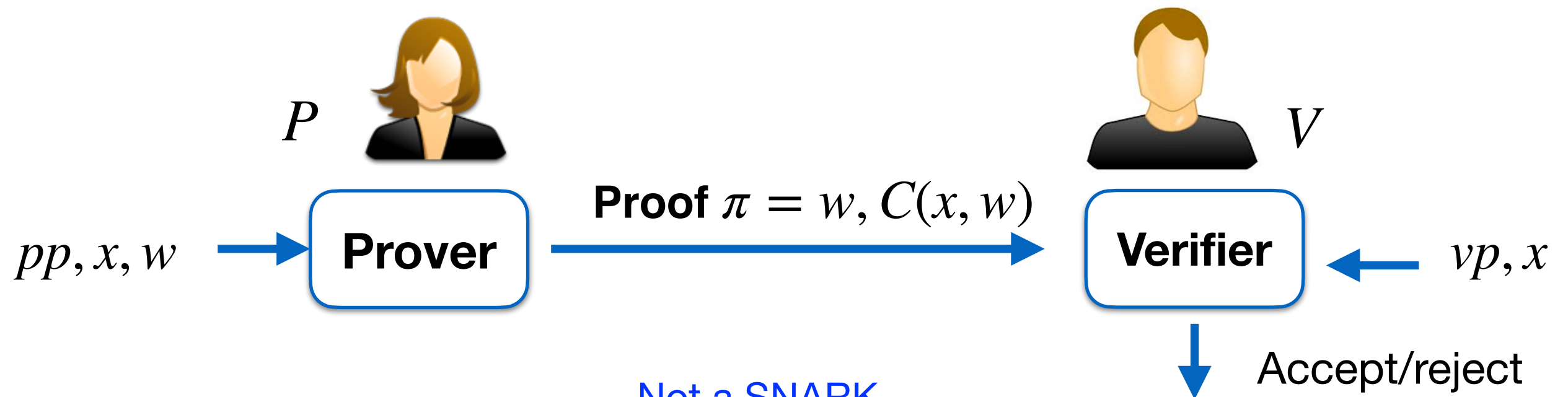
# SNARKS

Public arithmetic Circuit  $C(x, w) \rightarrow \mathbb{F}$

$x$  is the public input

$w$  is the secret witness

Preprocessing/setup step :  $S(C) \rightarrow (pp, vp)$  (public parameters)



$w$  can be long

Verifier computes  $C(x, w)$  which can be “slow”

Witness  $w$  is revealed, not Zero knowledge

# Preprocessing SNARKs:

## Desirable properties

Setup step :  $S(C, r) \rightarrow (pp, vp)$  (r random)

### Trusted Setup per circuit:

r should be kept secret from Prover

### Trusted but universal (updatable) setup:

Secret r is independent of Circuit C.

Step 1: Init step  $S_{init}(\lambda, r) \rightarrow gp$

Step 2:  $S_{index}(gp, C) \rightarrow (pp, vp)$

### Transparent setup: S(C)

Does not use secret r (no trusted setup)

Desirable

Specially important for  
blockchain application

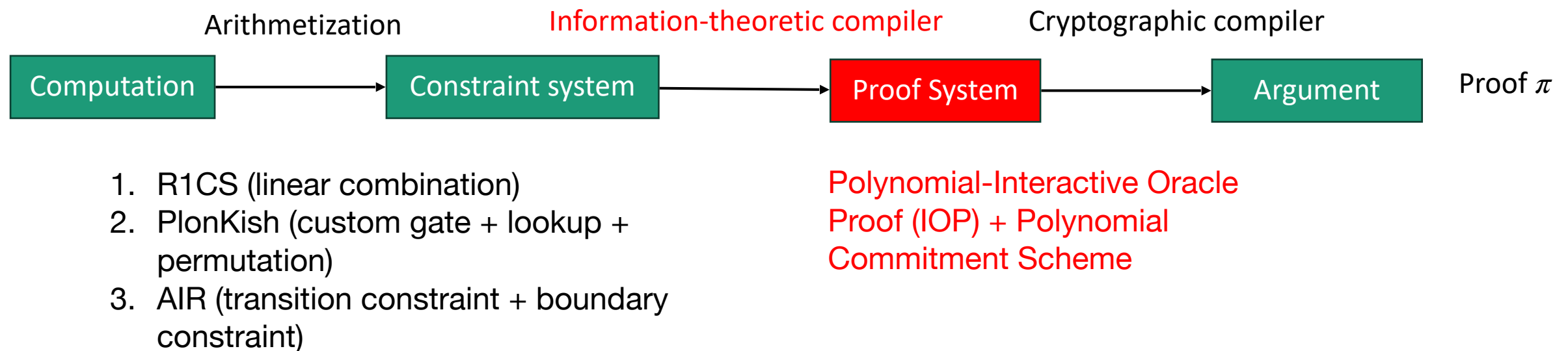
# SNARKs

ZKP	Trusted-Setup	Post Quantum
Groth16	Trusted per circuit	No
Ligero	Universal Trusted	Yes
Aurora	Transparent	Yes
Plonk	Universal Trusted	No
Sonic	Universal Trusted	No
Marlin	Universal Trusted	No
Plonky2	Transparent	No

# SNARK Construction

# Computing SNARKs

- Add zero-knowledge property -> ZK-SNARK
- Pipeline of SNARK:





# General Constructions of SNARKs

Polynomial Commitment Schemes (PCS)

Cryptographic Object

+

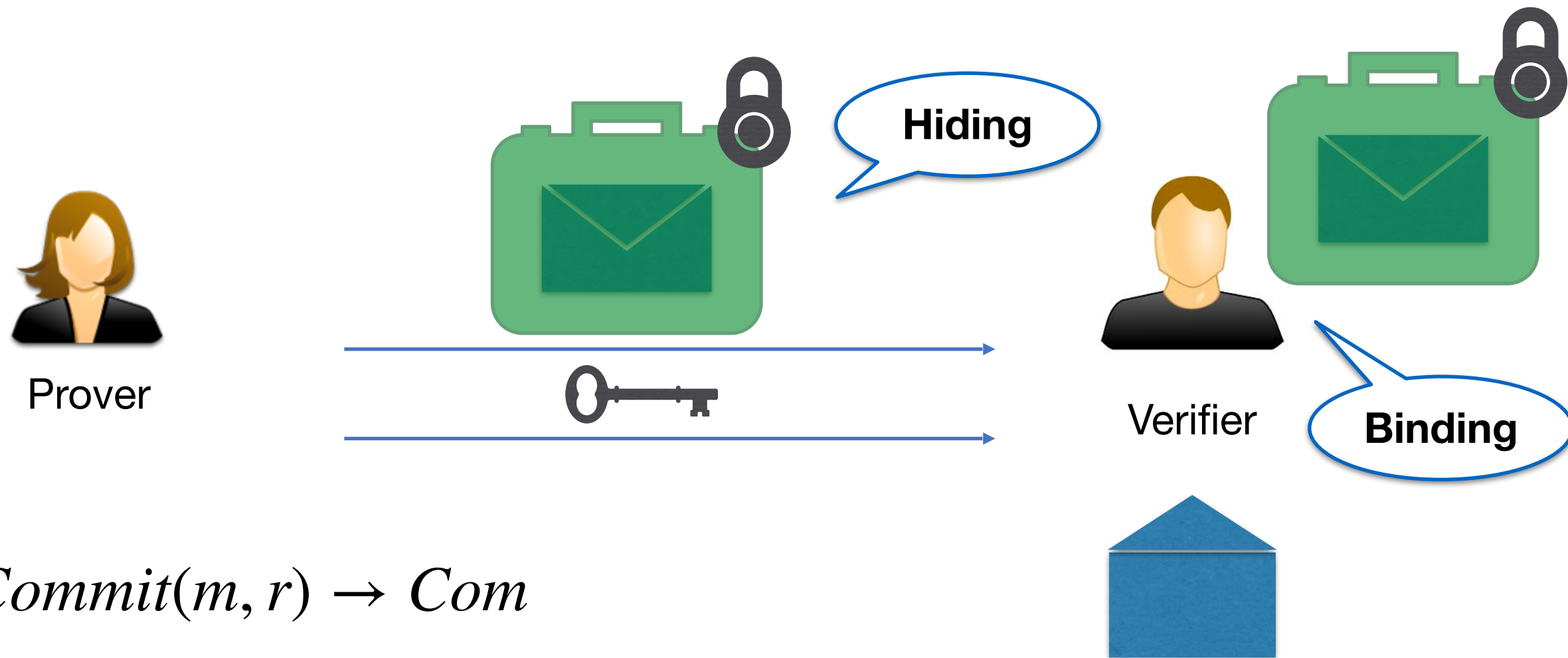
Interactive oracle proofs (IOP)

Information Theoretic Object



**SNARKs for General Circuits**

# Commitment



$Commit(m, r) \rightarrow Com$

$Verify(m, com, r) \rightarrow Accept/Reject$

**Hiding:** Com reveals nothing about m

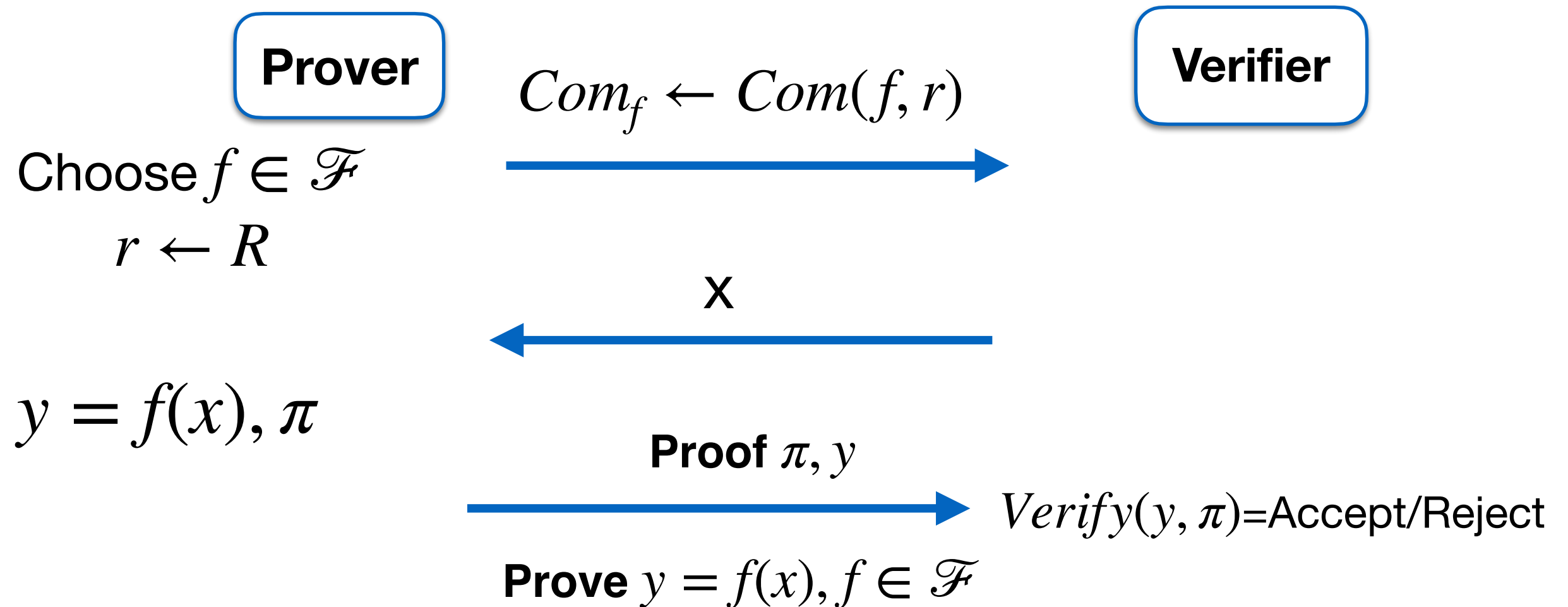
**Binding :** Cannot produce Com such that there are more than one openings

# Construction

- Fix a hash function  $H : \mathcal{M} \times \mathcal{R} \rightarrow T$
- $Commit(m, r) : Com = H(m, r)$
- $Verify(m, Com, r) : Accept \quad if \quad Com = H(m, r)$
- Should have hiding and binding property for suitable Hash function

# Committing to a Function

Let  $\mathcal{F} = f : X \rightarrow Y$  be a family of functions



# Functional Commitments

- **Vector Commitments:** Commit of a vector

$$\vec{v} = (v_1, v_2, \dots, v_d) \in \mathbb{F}_p^d, \text{ open } f_{\vec{v}}(i) = v_i$$

- **Polynomial Commitments:** Commit to a univariate function

$$f(X) \in \mathbb{F}_p^{\leq d}[X], d \text{ is the degree of the polynomial}$$

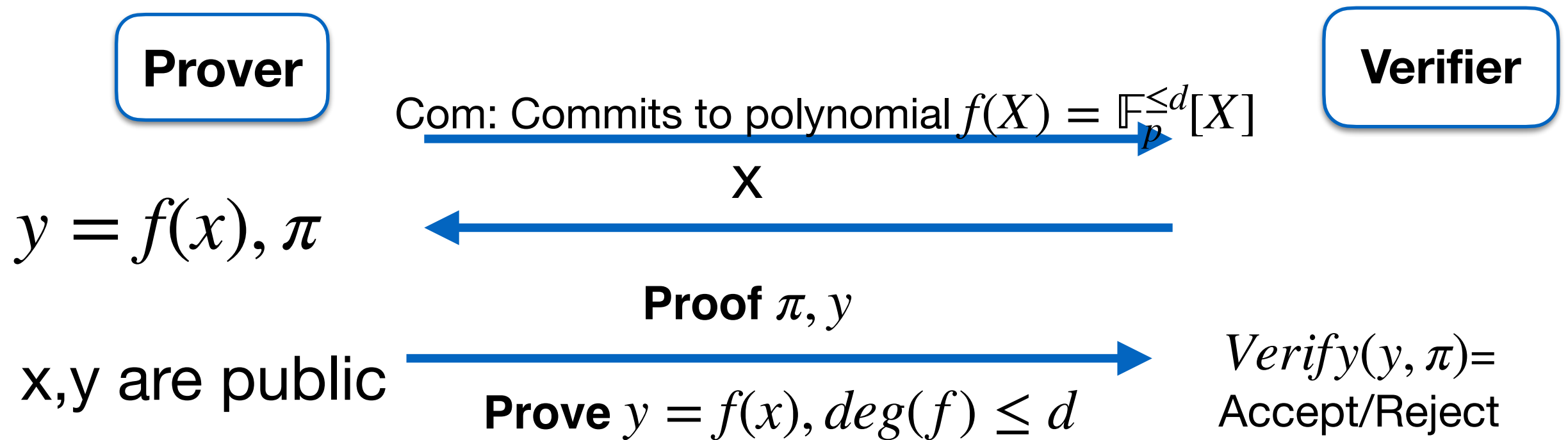
- **Multilinear Commitment:** Commit to a multivariate function

$$f(X) \in \mathbb{F}_p^{\leq 1}[X_1, X_2, \dots, x_k]$$

- **Inner Product Commitments (Inner product arguments IPA):**

$$\text{Commit to } \vec{v} \in \mathbb{F}_p^d \quad \text{Open an inner product } f_{\vec{v}}(\vec{u}) = (\vec{v}, \vec{u})$$

# Polynomial Commitments



Examples:

**Bilinear Group based:** KGZ'10 (Trusted Setup), DORY'20-Transparent setup

**Hash Functions (Post Quantum):** FRI (used in STARKs)

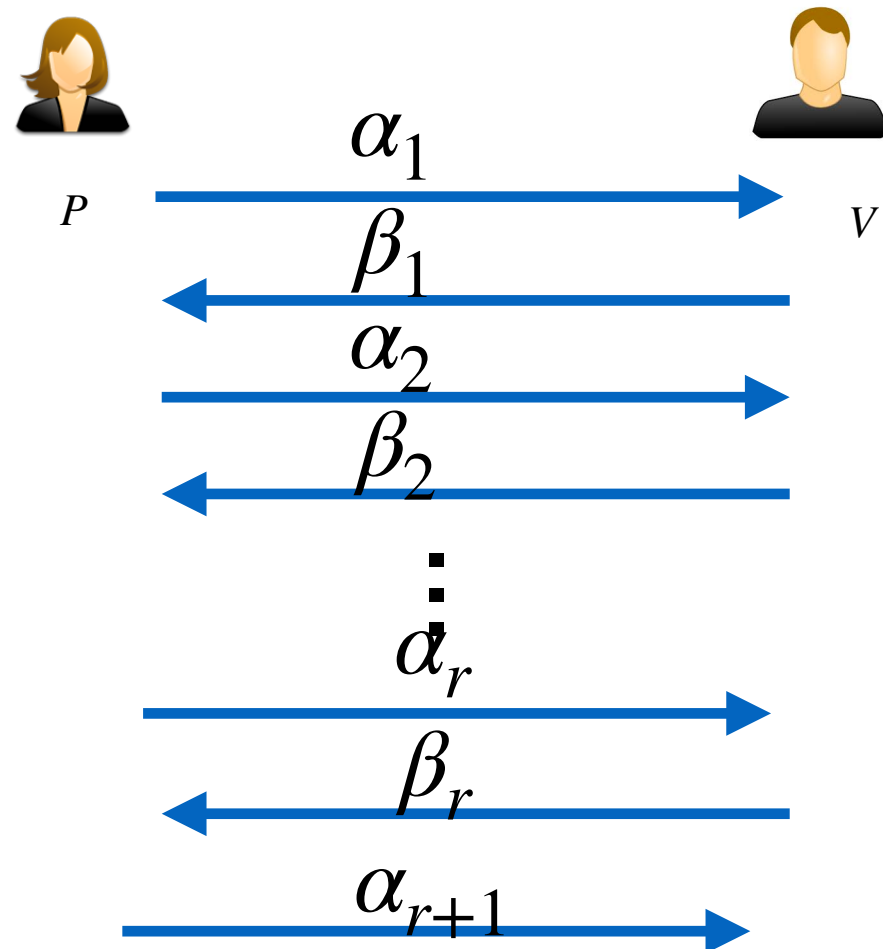
**Elliptic curves:** Bulletproofs, Short proofs but verifier time is  $O(d)$

# Important Observation from Schwartz–Zippel Lemma

- Let  $f(X) \in \mathbb{F}_p^{\leq d}[X]$  be a non-zero polynomial
- For  $r \xleftarrow{\$} \mathbb{F}_p : \Pr[f(r) = 0] = d/p$
- If  $p \approx 2^{255}, d \leq 2^{40}$ ,  $d/p$  is negl.
- We can say that for  $r \xleftarrow{\$} \mathbb{F}_p$ , if  $f(r) = 0$ , then  $f$  is identically zero with high probability
- This serves as a simple test for a committed polynomial
- The Schwartz-Zippel lemma holds even for multivariate polynomials ( $d$  the total degree of  $f$ )
- Let  $f, g \in \mathbb{F}_p^d[X]$ , for  $r \xleftarrow{\$} \mathbb{F}_p$ , if  $f(r) = g(r) \implies f(r) - g(r) = 0$ , then  $f = g$  w.h.p.
- Tests two polynomials are equal

# Fiat-Shamir Transform

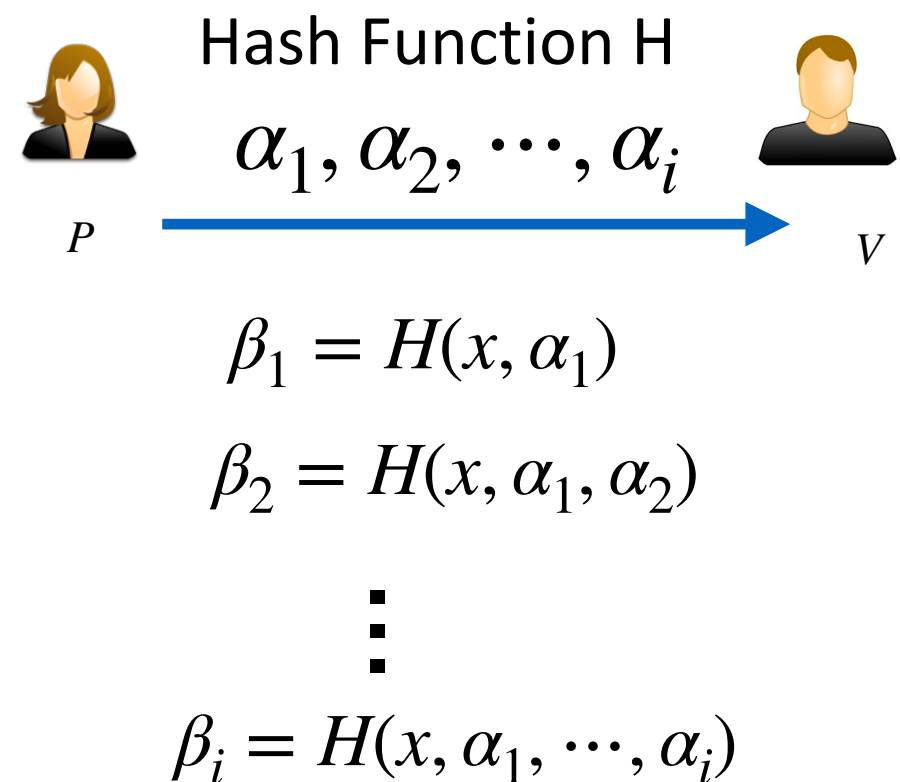
Public-coin interactive argument



Each  $\beta_i$  uniformly random

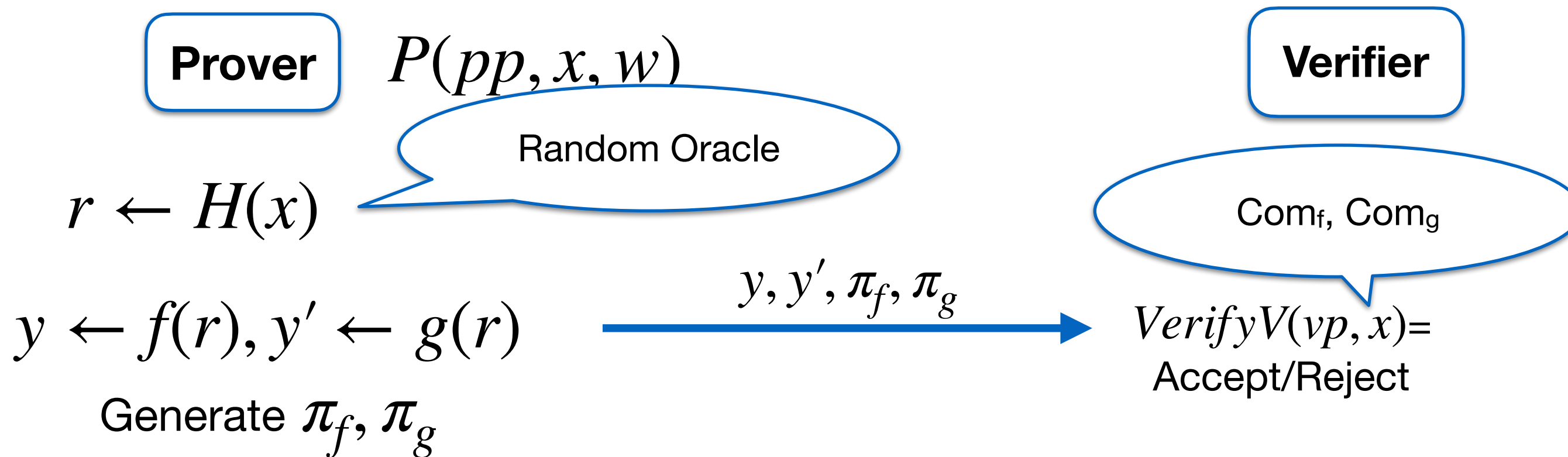


Non-Interactive argument





# Non-interactive Polynomial Equality Testing



# Interactive Oracle Proof

- Goal: Boost functional commitments to generate SNARKs for general circuits

Converts Polynomial commitment scheme  
 $f(X) = \mathbb{F}_p^d[X]$  to

SNARK for any circuit  $C$ , where  $|C| < d$

Ben-Sasson-Chiesa-Spooner'16

# F-IOP

- Let  $C(w,x)$  be an arithmetic Circuit, let  $x \in \mathbb{F}_p^n$
- $\mathcal{F}$  – *IOP* is a proof system that proves  $\exists w : C(x, w) = 0$
- $Setup(C) \rightarrow pp, vp = (f_0^o, f_1^o, \dots, f_s^o)$  (oracles for function in  $\mathcal{F}$  which will be instantiated with commitments)

# $\mathcal{F}$ – *IOP*: Proving $C(w,x)=0$

$P(pp, x, w)$

**Prover**

$V(vp, x)$

**Verifier**

Oracle for  $f_1 \in \mathcal{F}$

$r_1$

$r_1 \stackrel{\$}{\leftarrow} R$

Oracle for  $f_2 \in \mathcal{F}$

$r_2$

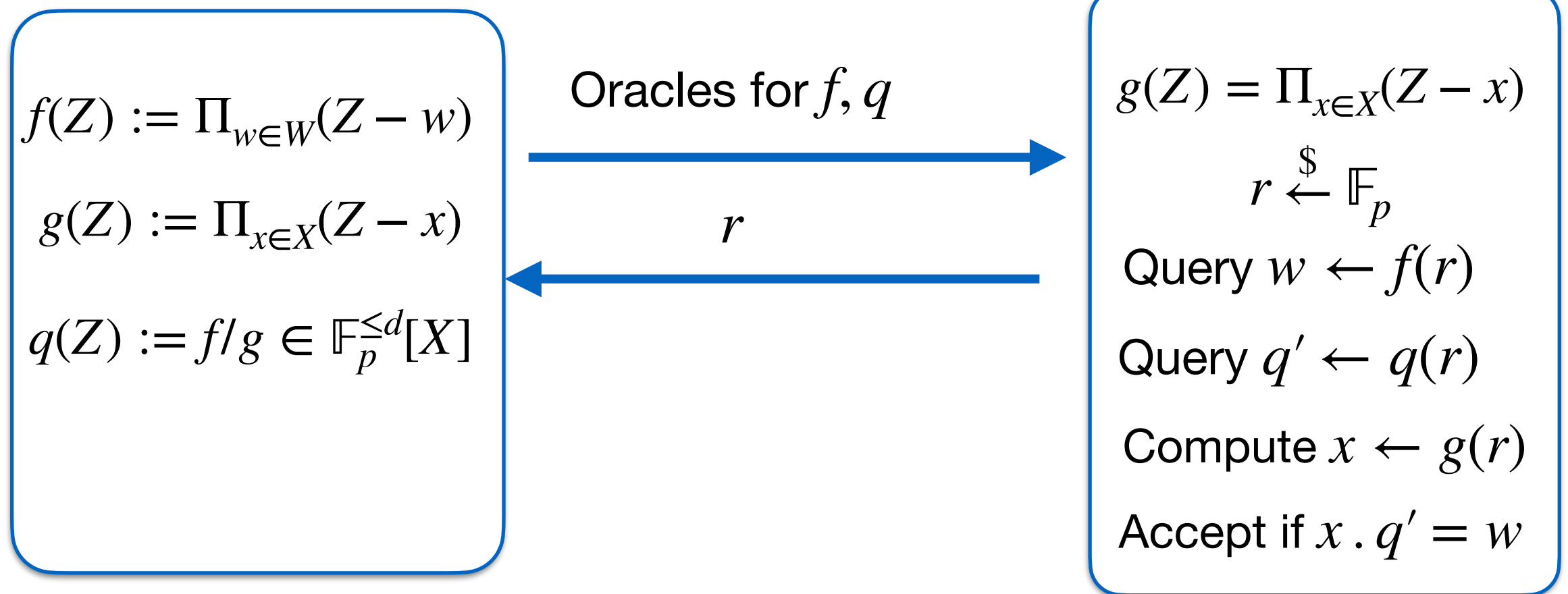
Oracle for  $f_t \in \mathcal{F}$

$r_t$

$Verify^{f_0^o, f_1^o, \dots, f_s^o}(x, r_1, r_2, \dots, r_t)$

# Example

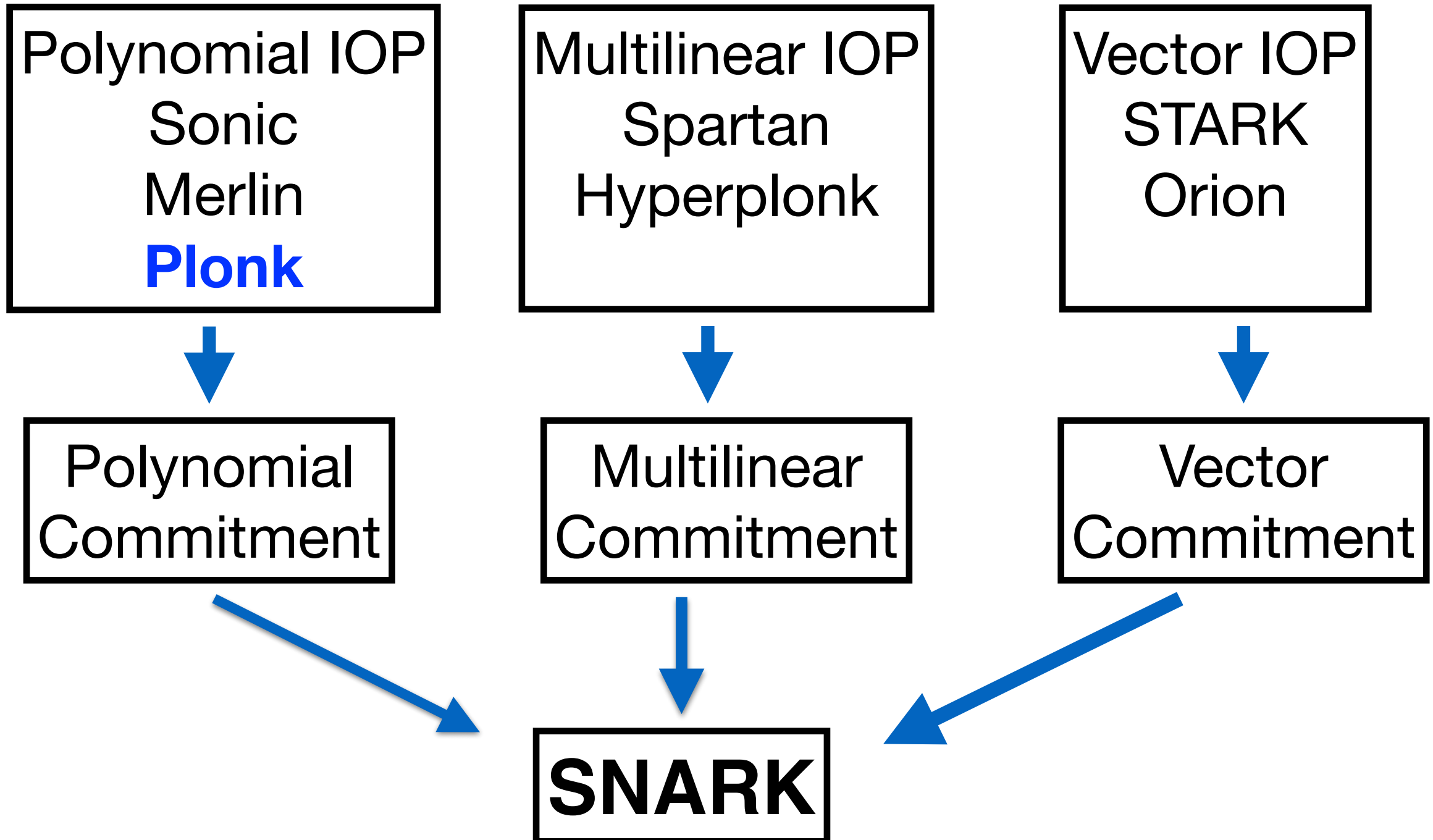
$$P(pp, x, w) \quad C(X, W) = 0, \text{ s.t. } X \subseteq W \subseteq \mathbb{F}_p \quad V(vp, x)$$



Knowledge Soundness:  $V$  accepts if  $f = g \cdot q$  w.h.p  $\implies X \subseteq W$

Extractor( $X, f, q, r$ ) : Output witness  $W$  by computing all roots of  $f(Z)$

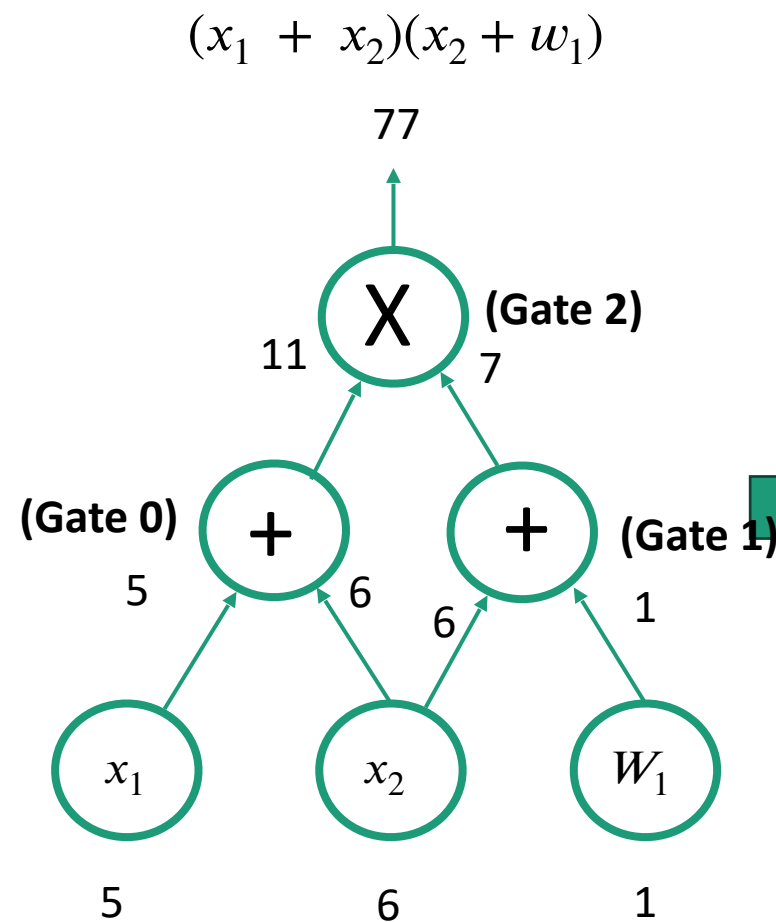
# Candidate IOPs



# Proof Gadgets

- Let  $S$  be a subset of  $\mathbb{F}_p$  of size  $k$
- Let  $\mathbb{F}_p^{\leq d}[X]$  ( $d \geq k$ ), verifier has oracle access of  $f$
- Construct Poly IOP for the following tasks
- Zero-Test: Prove that  $f$  is identically zero on  $S$
- Sum-Check: prove that  $\sum_{a \in S} f(a) = 0$
- Product-Check: prove that  $\prod_{a \in S} f(a) = 1 \dots$

# Plonk High level Idea



$$|c| = \# \text{gates}, \text{Inputs} = |x| + |W|$$

$$d = 3|C| + |I|$$

Input	5	6	1
Gate 0:	5	6	11
Gate 1:	6	1	7
Gate 2:	11	7	77

Left input      Right input      Output

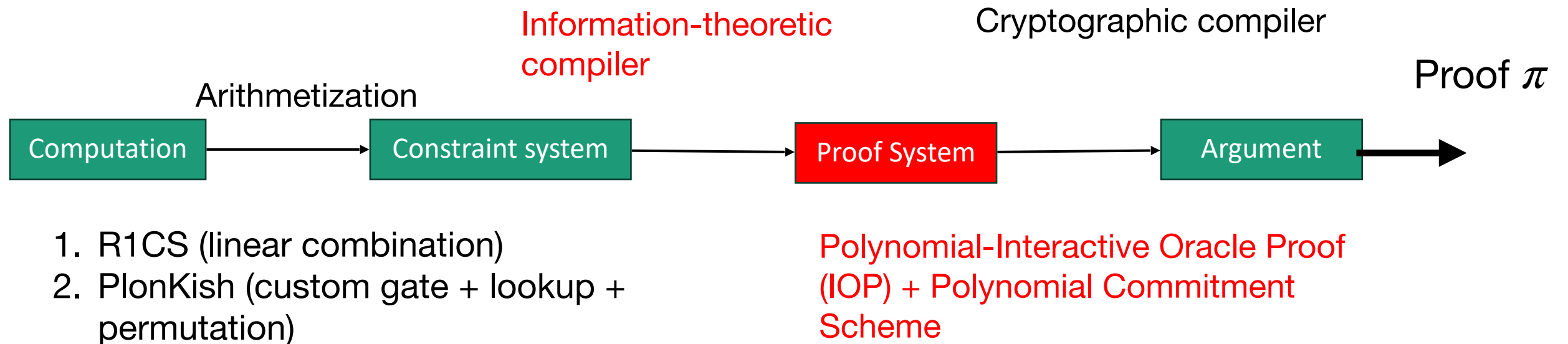
1. T encodes the correct inputs
2. Every gate is evaluated correctly
3. The wiring is implemented correctly
4. The output gate is 0

Compile circuit to a computation trace  
(Arithmetization)

Encode the trace as a polynomial T



# Computing SNARKs



# Some Libraries

- Circom library
- GNARK
- Zokrates

# Applications

# Blockchain Applications

- Proof of solvency of exchanges
- Proving historical facts in zero-knowledge
- Privacy preserving blockchains like Monero
- Payment Channel Networks
- Self Sovereign Identities on Blockchains
- Many more....

Thank you!

# Interactive Proof Systems

Definition[GMR85]: An **Interactive *proof system*** for membership in  $L$  is a PPT algorithm  $V$  and a function  $P$ , such that  $\forall x$  :

Completeness: If  $x \in L$ , then,  $Pr[(P, V) \text{ accepts } x] \geq 2/3$

Soundness: If  $x \notin L$ , then  $\exists P^*$  s.t.,  $Pr[(P, V) \text{ accepts } x] \leq 1/3$

Completeness and soundness can be bounded by any  $c : \mathbb{N} \rightarrow [0,1]$  and  $s : \mathbb{N} \rightarrow [0,1]$  as long as

- $c(|x|) \geq 1/2 + 1/\text{poly}(|x|)$
- $s(|x|) < 1/2 - 1/\text{poly}(|x|)$

$\text{poly}(|x|)$  Independent repetitions implies  $c(|x|) - s(|x|) \geq 1 - 2^{-\text{poly}(|x|)}$

# Succinct Arguments of Knowledge

A Succinct NARK is a triple  $(S, V, P)$  , such that

$S(C) \rightarrow (pp, vp)$  for prover and verifier

$P(pp, x, w) \rightarrow \pi$  Short:  $|\pi|$  is sublinear in  $|x|$

$V(vp, x, \pi) \rightarrow \text{Accept/Reject}$

Verification is fast  $O_\lambda(|x|, \text{sublinear } |C|)$

# Reason For Interest

**Babai-Fortnow-Levin-Szegedy 1991:**

*In this setup, a single reliable PC can monitor the operation of a herd of supercomputers working with unreliable software.*

*“Checking Computations in Polylogarithmic Time”*