Zero Knowledge Proofs

Sushmita Ruj

Recap

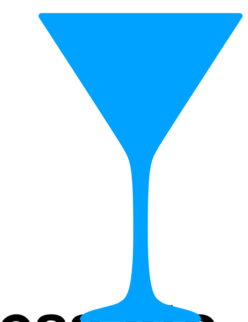
- Blockchain History
- Blockchain Basics
- Attacks on Blockchains
- Privacy
- Open Questions

This Lecture

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

Proofs <->Verifiability

Driver's Licence



Is age >18

User has to reveal Date of Birth, Address etc

Verify without revealing all information: Verifiable credentials

Query

Result

Prove that the result is correct, without disclosing the answers

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

IMMUTABLE

ZCASH

STARKWARE

Aleo

Aztec

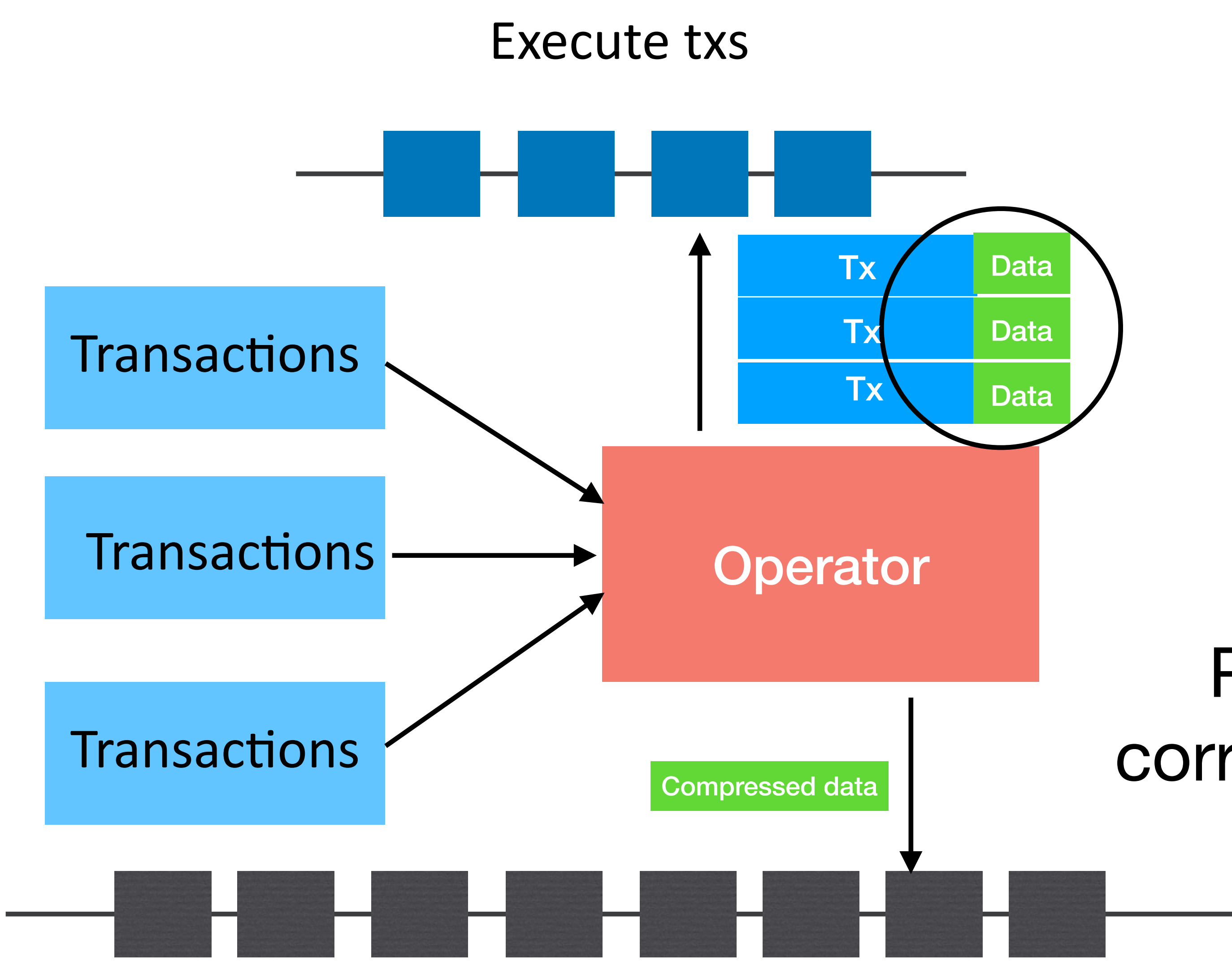
Application 1: Privacy Preserving Transactions

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

Application 1: Privacy Preserving Transactions

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

Application 2: Rollups for Scalability



Prove that transactions are correctly performed and batched

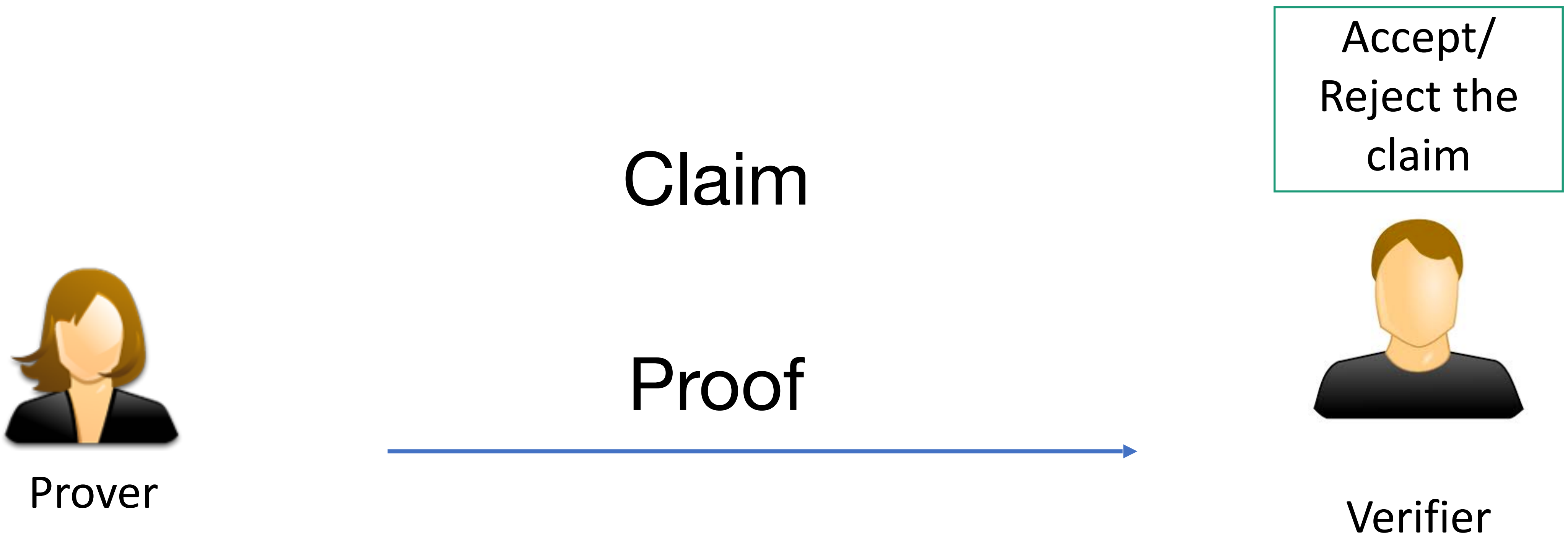
Zero-Knowledge Proofs Foundations

Proofs

- A method to establish the truth
  - Legal
  - Authoratitive
  - Scientific
  - Philosophical
  - Mathematical

    Axioms -> ->…-> Propositions
  - Probabilistic, Interactive

Proofs



Efficient Proofs (NP-Proofs)

Claim x

Accept/
Reject the
claim

Prover

Witness w

|w| is polynomial in |x|

Verifier

Unbounded

Fast Verification
|w| is polynomial in
|x|

Proof Systems

P wants to prove the following statement

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

$$L = \{x \,|\, \exists \pi, V(w, x) = ACCEPT\}$$

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

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

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

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

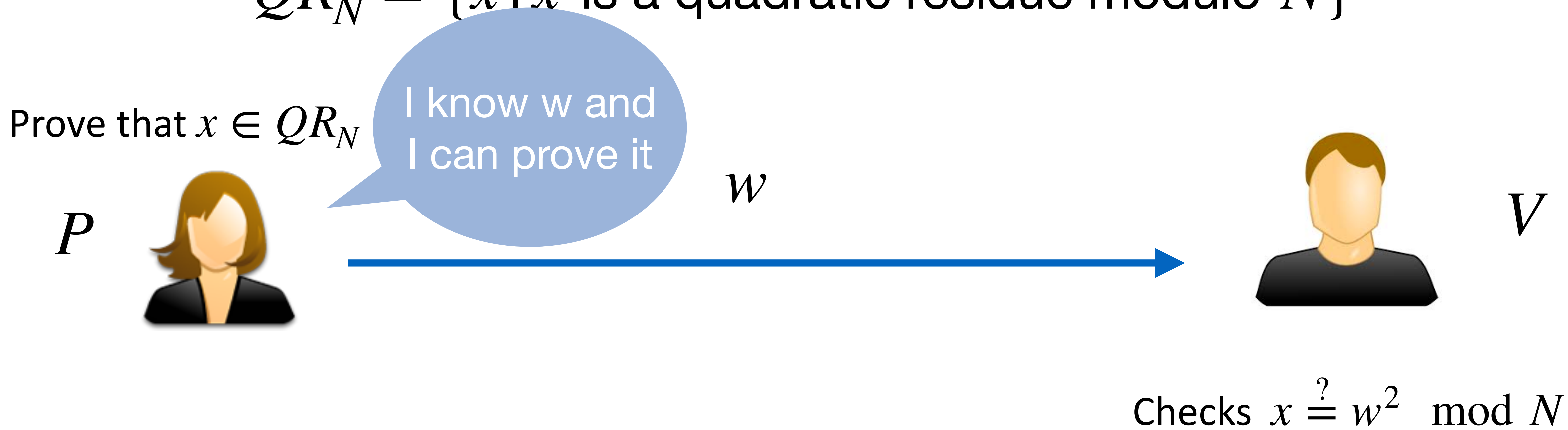
Soundness: If $x \notin L$, then $\forall w, V(w, x) = REJECT$

Efficiency: $V(w, x)$ halts after at most $poly(|x|)$ steps

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

Proofs leaking secrets

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



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 l*earn something?

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

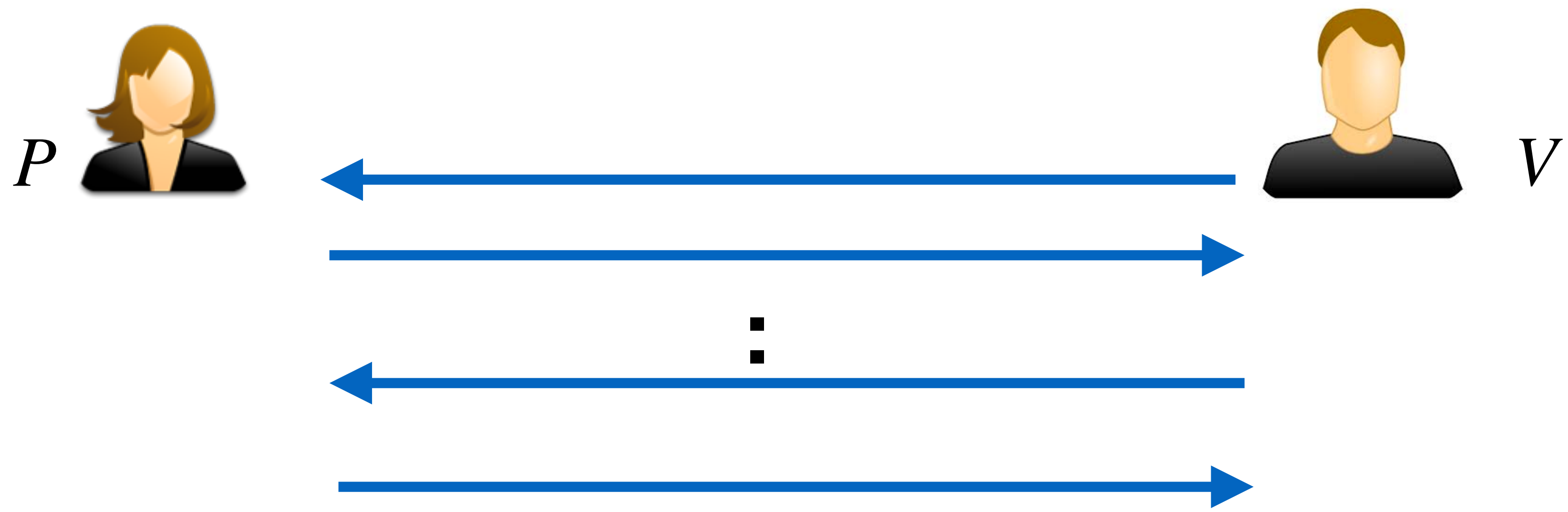
Zero-knowledge: whatever is computed following the interaction could have been computed without it.

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

Paradoxical Proofs

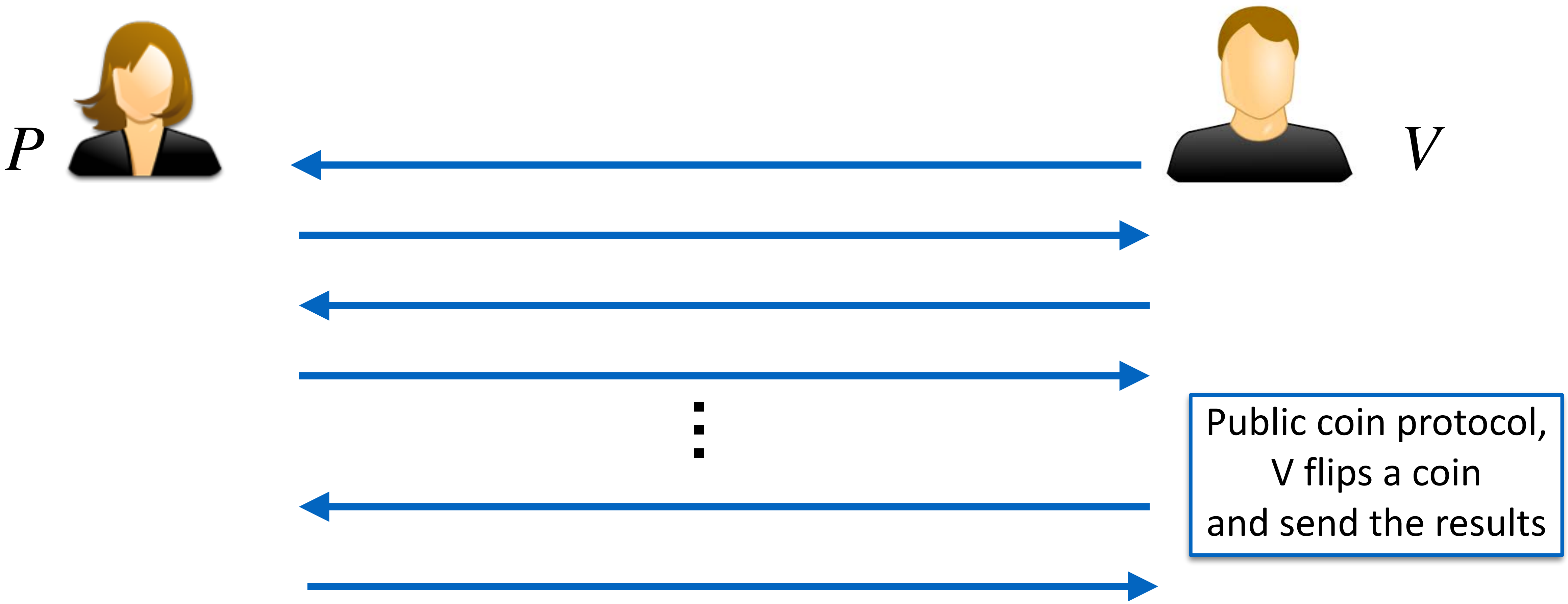
- Zero-knowledge proofs: A proof that does not leak information
- Probabilistically checkable proofs: Proofs need not be read in their entirety

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



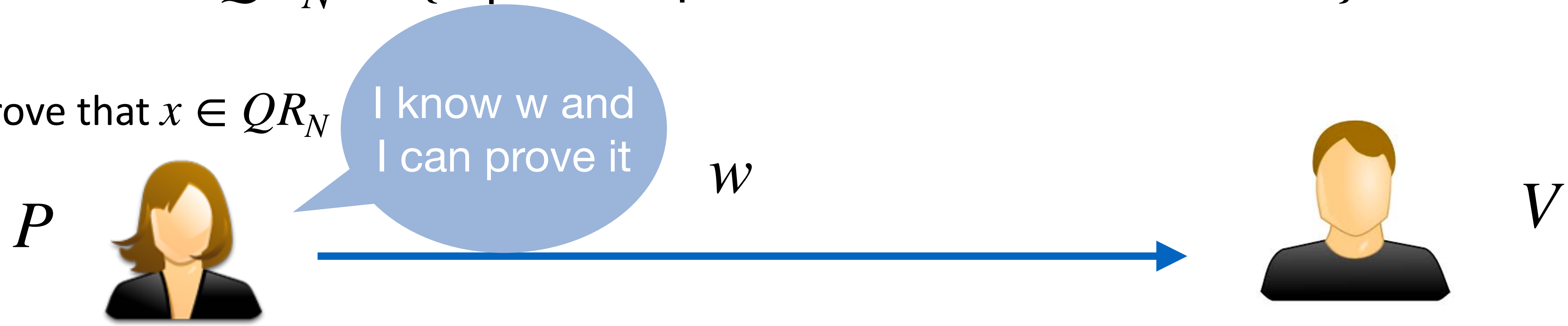
Completeness: P convinces V that the $x \in L$

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 $x \in QR_N$



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

Reveals information to V, that V could not have computed

Example

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

Choose random r

$1 \leq z \leq N, s.t., gcd(z, N) = 1$

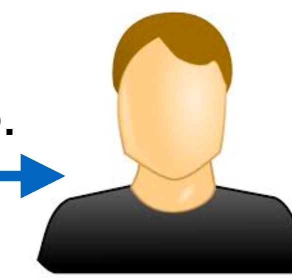
P

Send $z = r^2 \mod N$

If Bob received sq root of both z and zy mod N, then Bob will be convinced that claim is true, but also get y

If Alice gives either sq root of z or zy mod N then Bob will have to choose which one

The fact that Alice can choose either will convo once Bob.



V

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

If b=1, send t=z

If b=0, send $t = z\sqrt{y} \mod N$

Accept only if

$t^2 = zy^{1-b} \mod N$

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

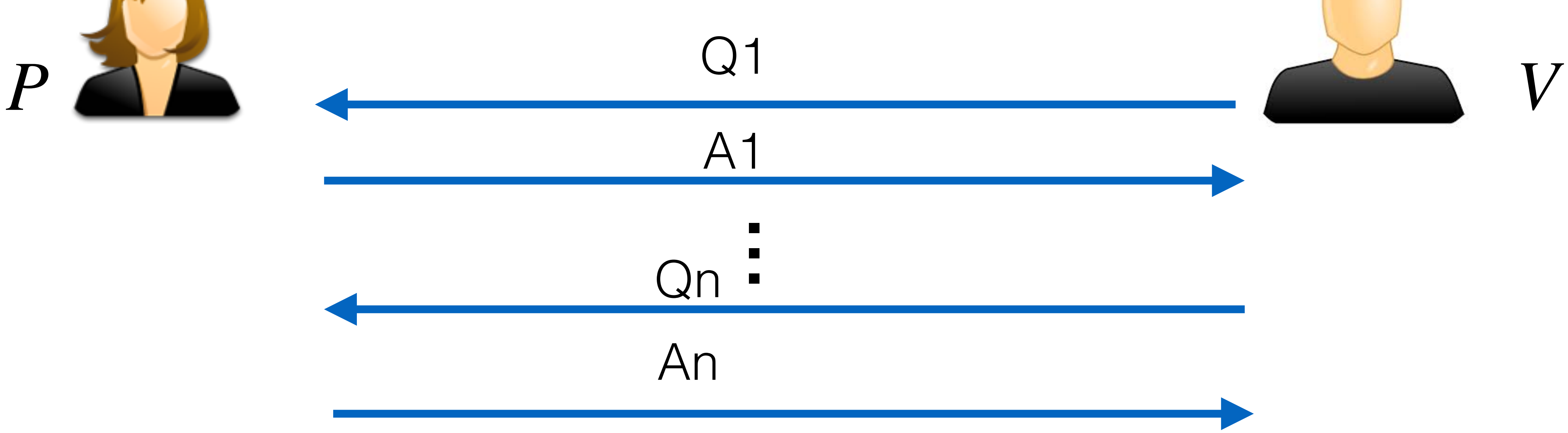
Important idea

- 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 $L \subseteq \Sigma^*$



ACCEPT/REJECT

V is a Probabilistic Polynomial time (PPT)

For any common input x, let:

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

Babai: Trading Group Theory for Randomness1985

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, $Pr[(P, V)$ accepts $x] = 1$

Soundness: If $x \notin L$, then $\forall$ $cheating \quad provers$ P^* s.t.,

$Pr[(P^*, V)$ accepts $x]$ is negl

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

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

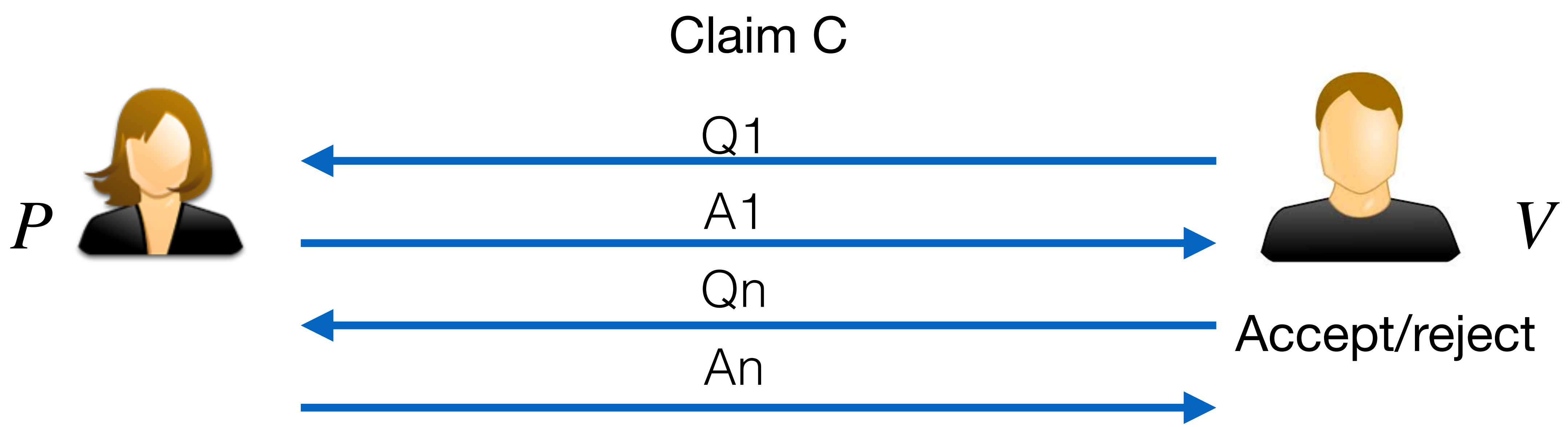
$poly(|x|)$ Independent repetitions implies $c(|x|) - s(|x|) \geq 1 - 2^{-poly(|x|)}$

Definition: class of languages IP= {L 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



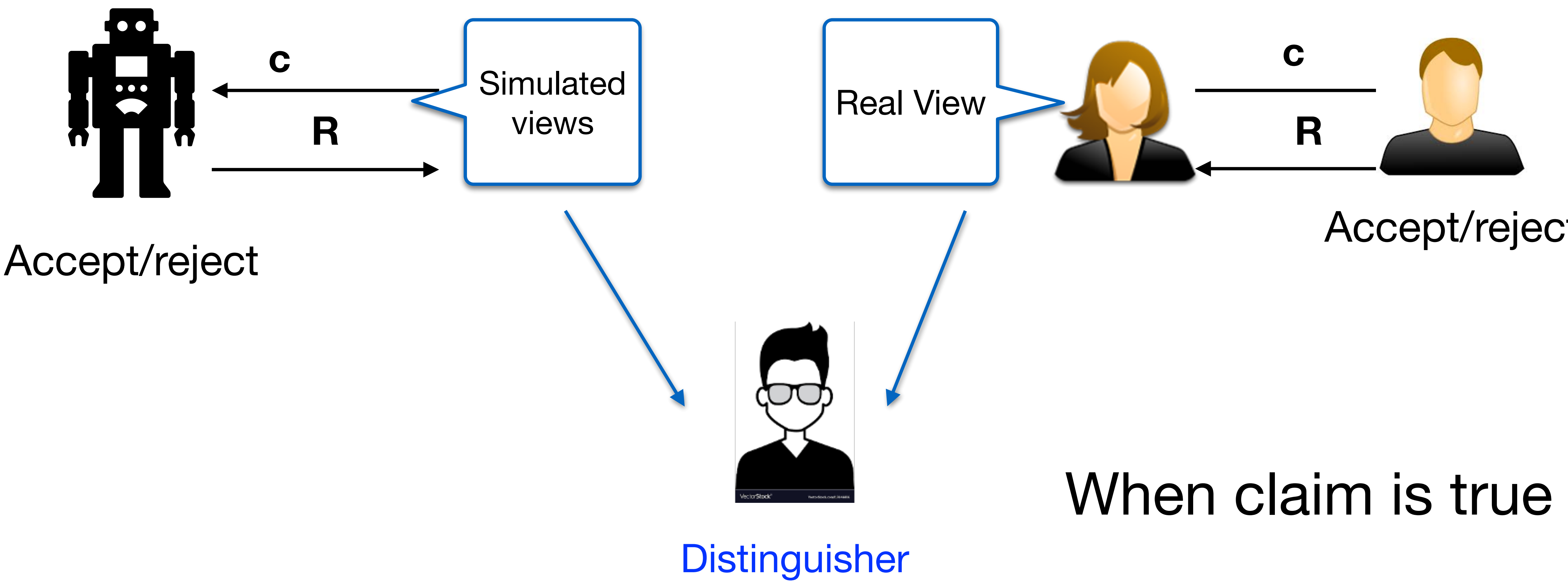
After interaction V learned C is true/false

A view of interaction includes coin toss + transcript

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

Probability distribution over coins of P and V

Simulation Paradigm



When claim is true

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

PPT Distinguisher

Sample

K-bit strings
D1

K-bit strings
D2

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, Prob[D guesses b] <1/2+negl

Zero-Knowledge Proof

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

1. $view_{\mathrm{V}}\,(P, V)[x, 1^{\lambda}]$ = {(Q1,A1,Q2,A2,…,coins of V)}
2. $Sim(x, 1^{\lambda})$ (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

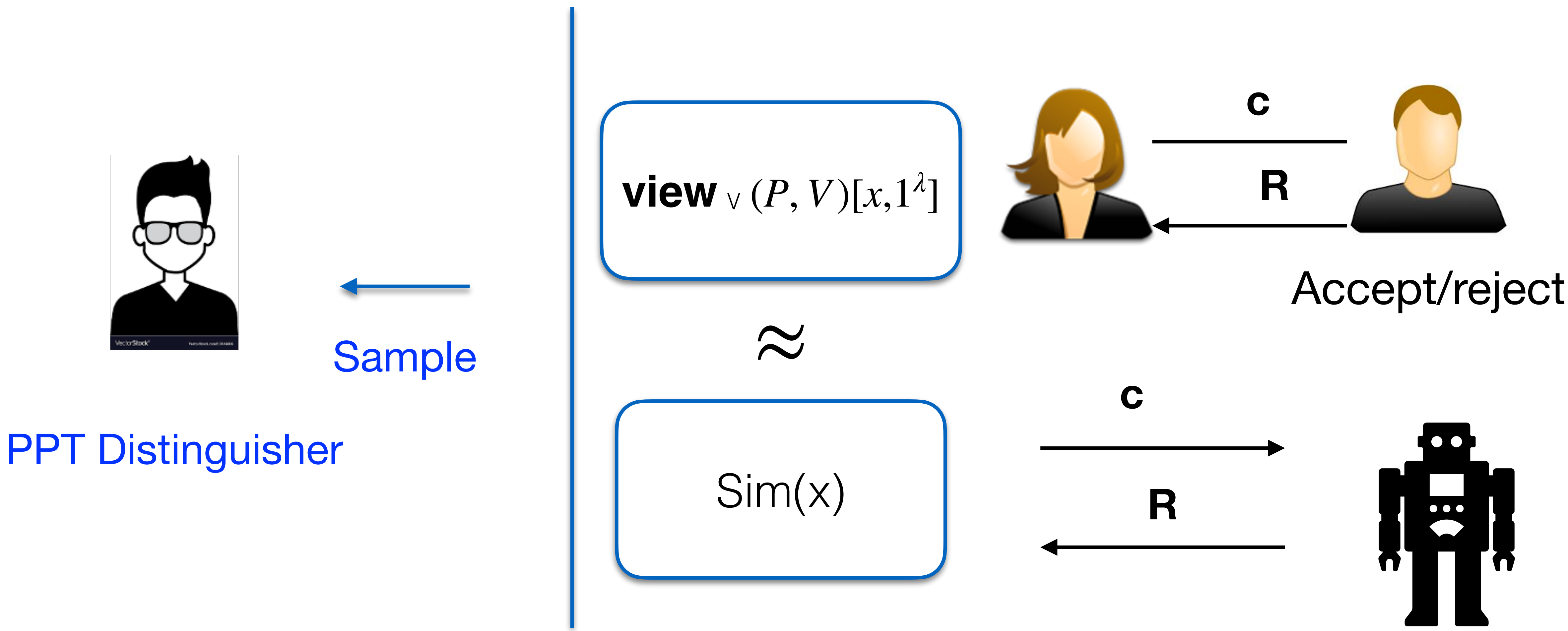
An Interactive Protocol (P,V) is honest verifier zero-knowledge for a language L if there exists a PPT algorithm Sim s.t. for every $\boldsymbol{x} \in \boldsymbol{L}$,

$$view_{\mathsf{V}}\,(P, V)[x, 1^{\lambda}] \approx Sim(x, 1^{\Lambda})$$

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

$$view_{\mathsf{V}}\,(P, V)[x, 1^{\lambda}] \approx Sim(x, 1^{\Lambda})$$

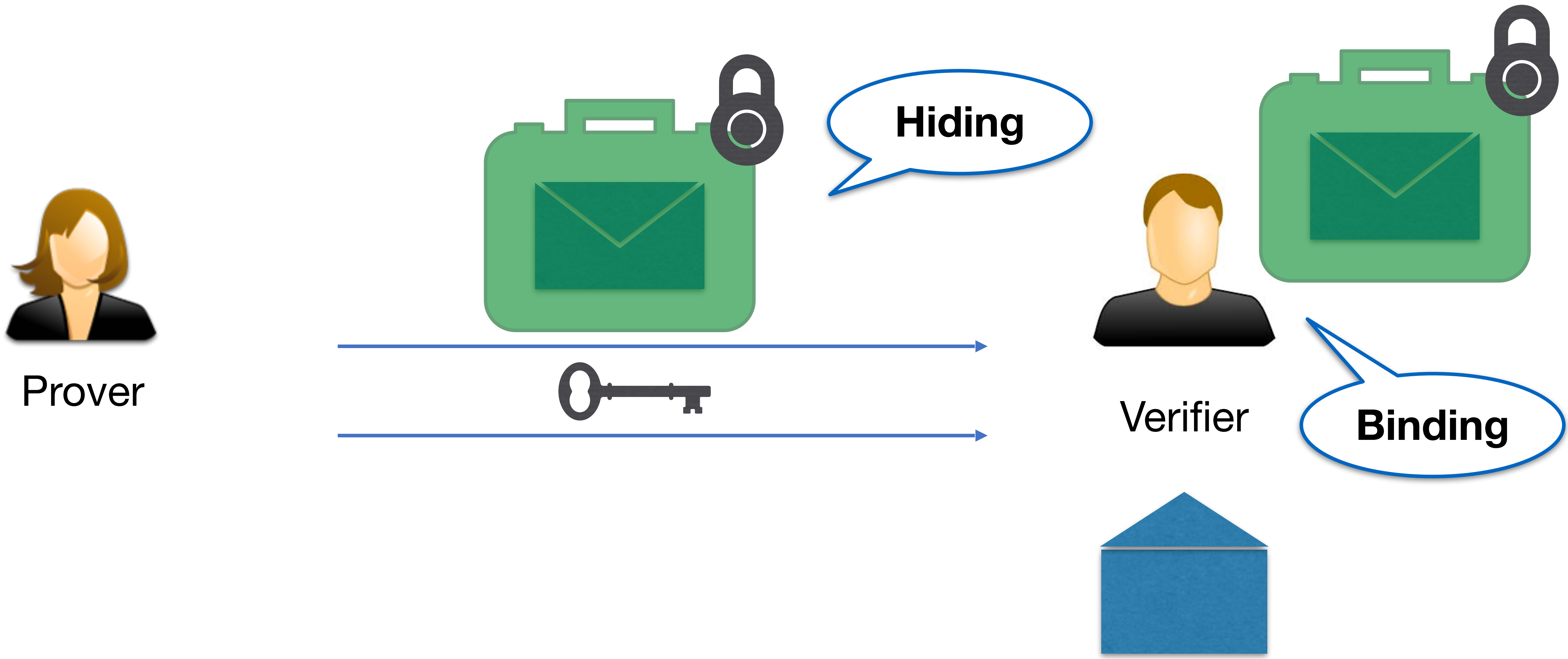
Perfect ZK



Verifier’s view can be exactly simulated

Simulated view ≈ 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.

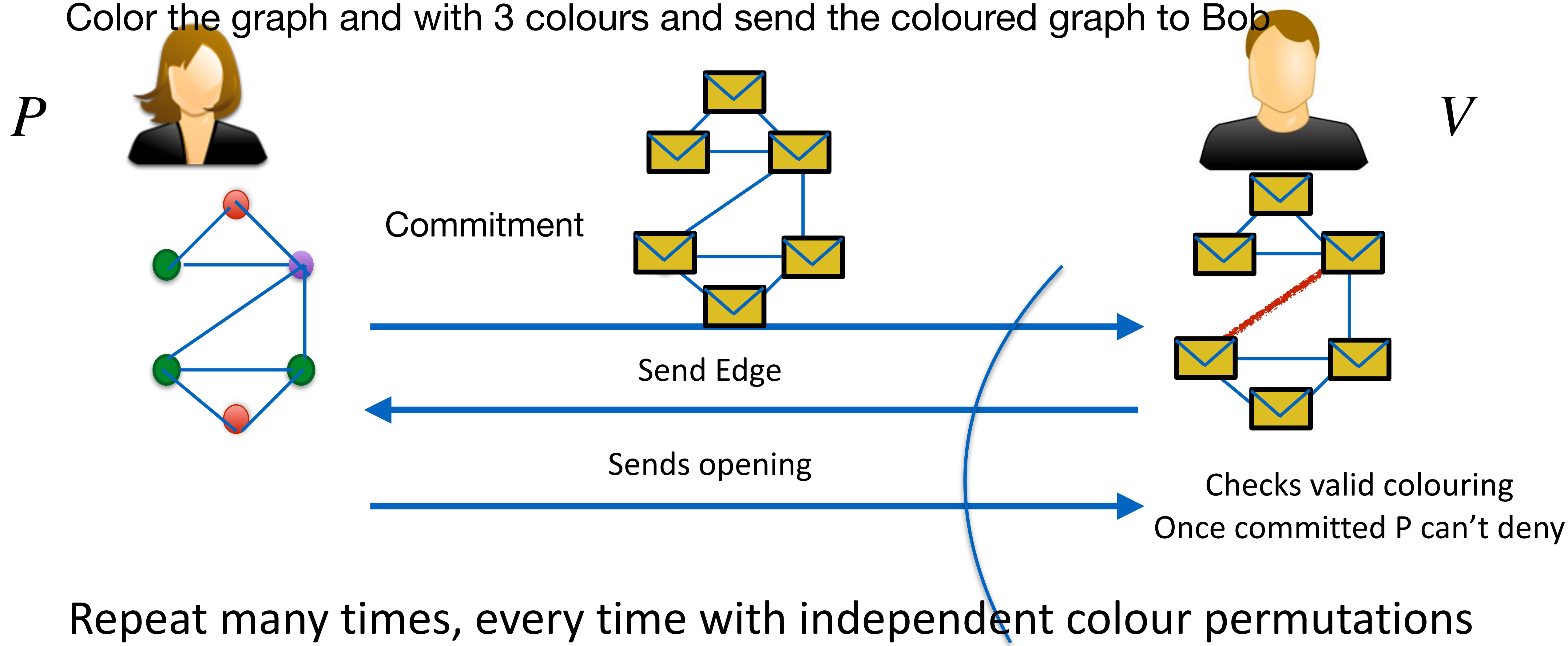
Ideas of the proof:

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

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

ZK Proof for Graph Colouring

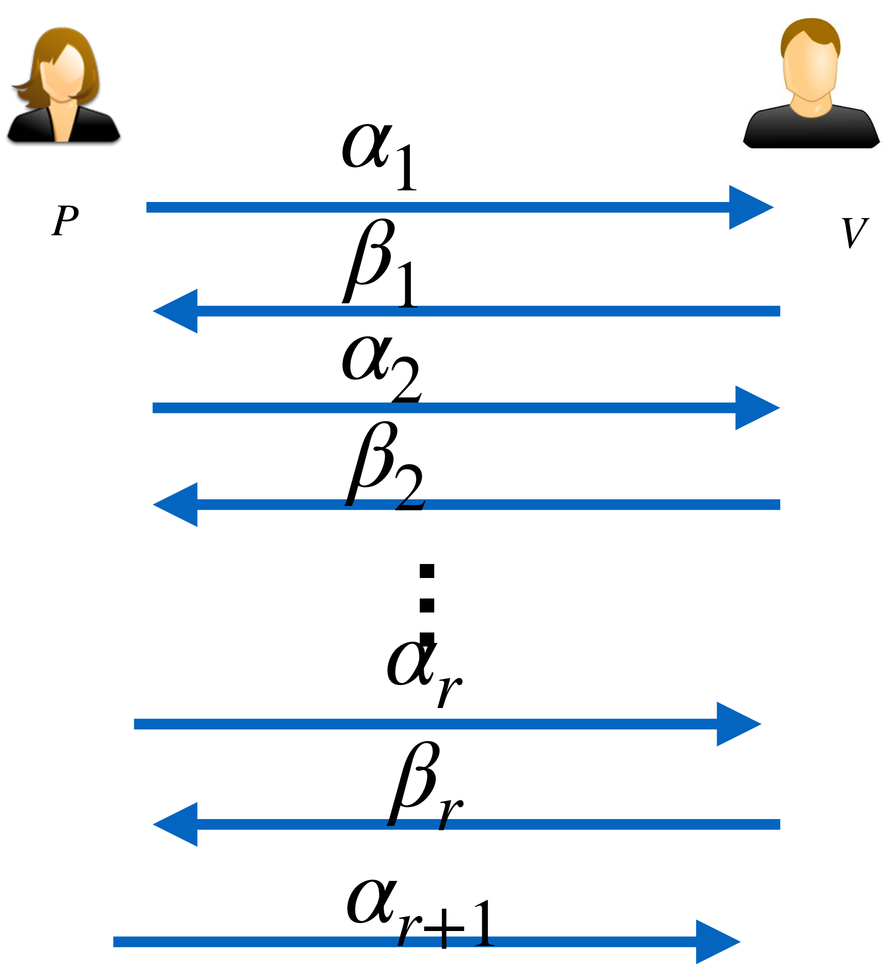
- Alice wants to prove that the nodes of a graph can be coloured with at most 3 colours, so that adjacent nodes have different colours
- Color the graph and with 3 colours and send the coloured graph to Bob



Repeat many times, every time with independent colour permutations

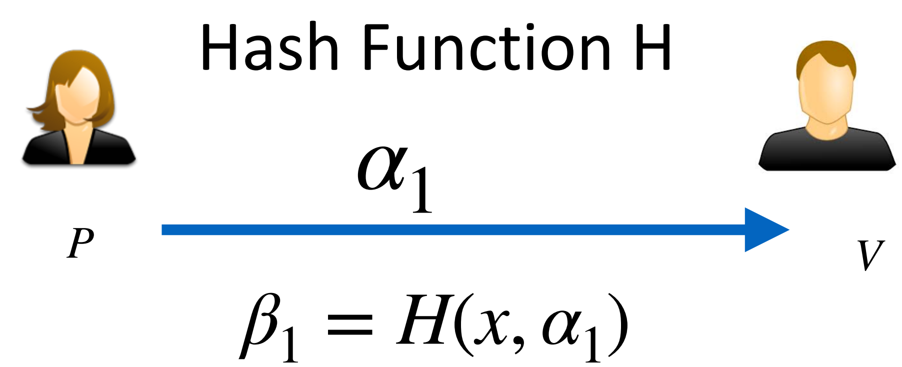
Fiat-Shamir Transform

Public-coin interactive argument



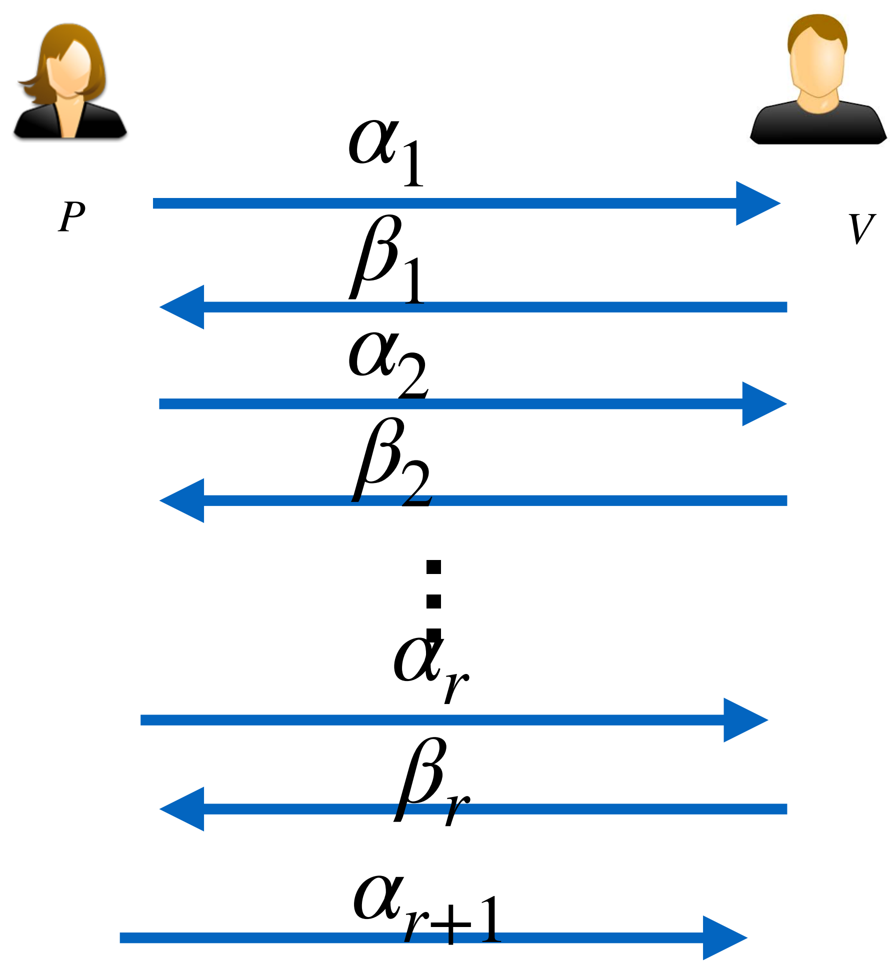
Each β_i uniformly random

Public-coin non-interactive argument



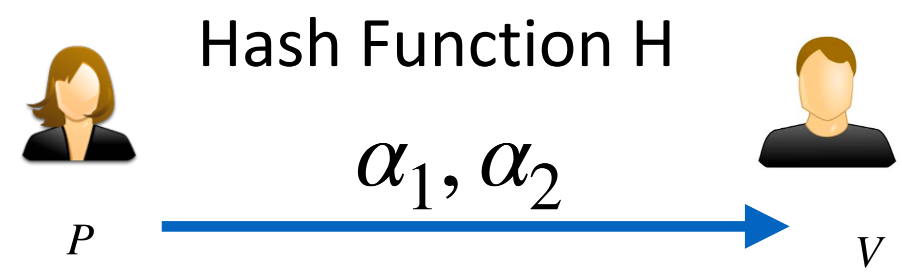
Fiat-Shamir Transform

Public-coin interactive argument



Each β_i uniformly random

Public-coin interactive argument

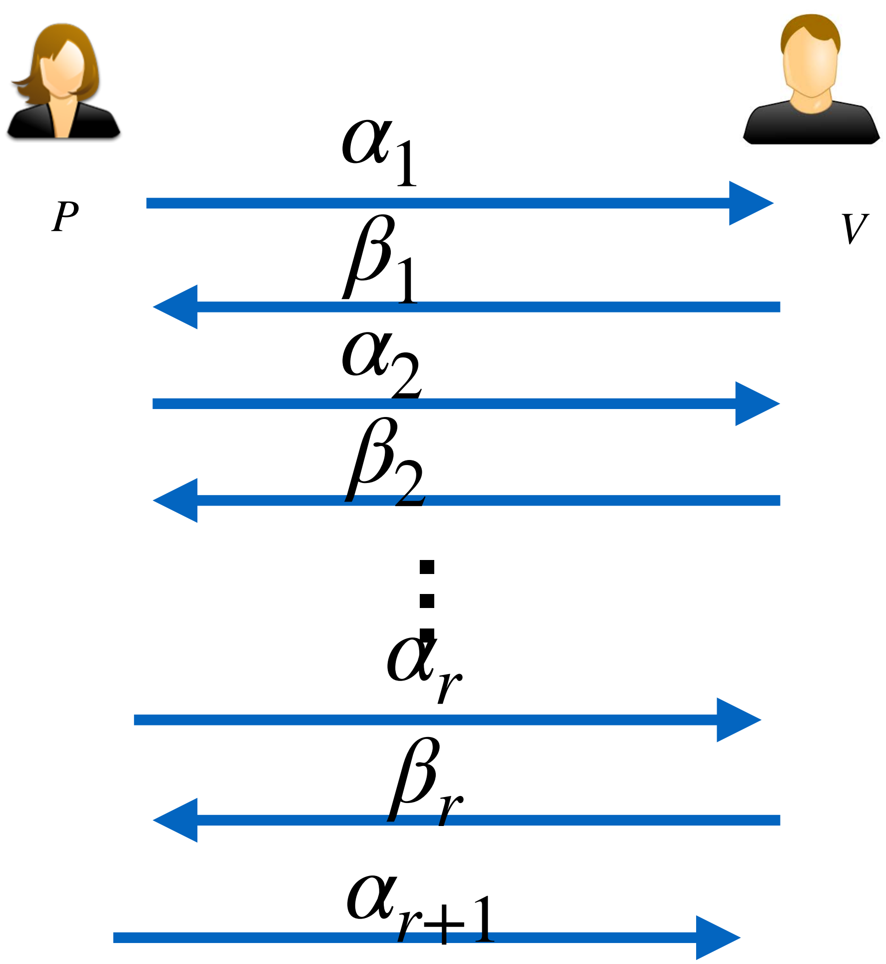


$$\beta_1 = H(x, \alpha_1)$$

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

Fiat-Shamir Transform

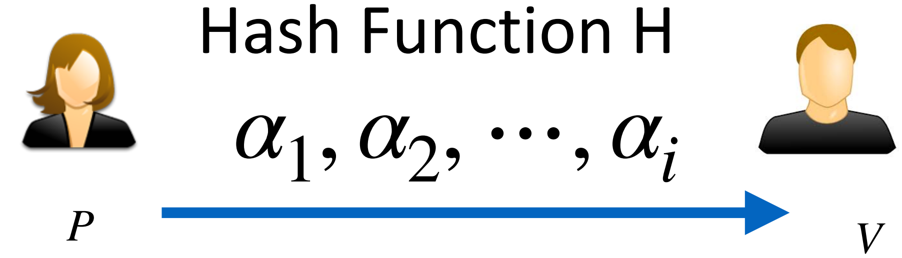
Public-coin interactive argument



Each β_i uniformly random

Non-Interactive argument

Hash Function H



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

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

$$\vdots$$

$$\beta_i = H(x, \alpha_1, \cdots, \alpha_i)$$

Identification Schemes

Prover | **Login/Password, ATM PIN,** | **Verifier**

sk | | **vk**

Accept/Reject

Attacks on Identification Schemes

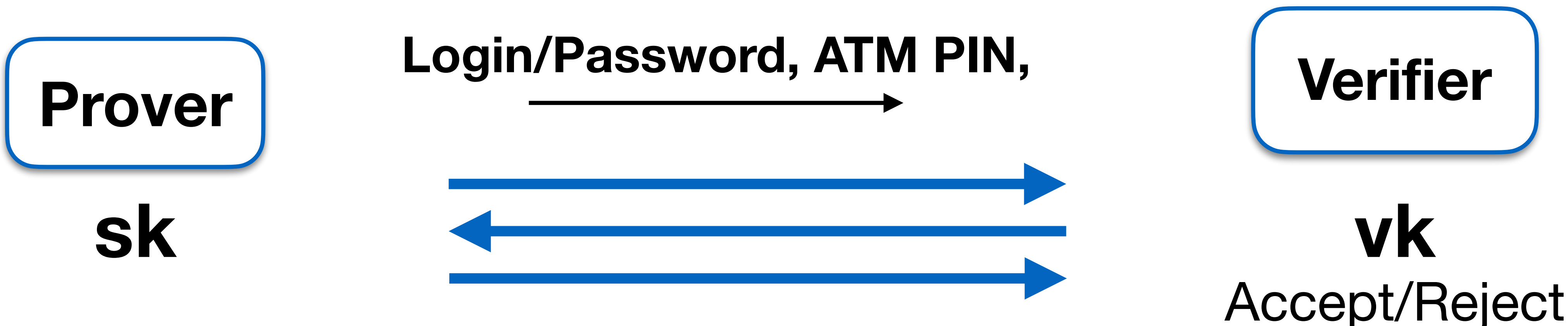
- Direct Attacks: Attacks are in close proximity
- Eavesdropping attacks: Eavesdropping on the communication channel, for example on the radio communication channel when a driver open a car with a fob
- Active Attacks: Interacts directly with the user (Eg a fake ATM machine)

Types of Identification Protocols

- Stateless VS Stateful: sk and vk changes (stateful) or might be fixed (stateless)

- One-sided Vs Mutual: Only prover authenticates itself (one-sided), or both authenticate one another (mutual)

Identification Schemes



An identification protocol is a triple I = (G; P; V).

- $(sk, vk) \leftarrow keyGen(1^{\lambda})$
- $P(sk) \rightarrow \pi$, P is an interactive protocol algorithm
- $V(\pi, vk) \rightarrow 0/1$, V an interactive protocol algorithm

Security : For all possible outputs (vk; sk) of G, if P is initialized with sk, and V is initialized with vk, then with probability 1, at the end of the interaction between P and V , V outputs accept.

Schnorr Identification

Let G be a cyclic group of prime order $\mathbb{Z}_q$,

g is a generator of G

Prover wants to verify it knows sk α, such that vk = $u = g^{\alpha}$

Prover		**Verifier**
$\alpha_t \xleftarrow{R} \mathbb{Z}_q$	$\xrightarrow{\quad u_t \quad}$
$u_t \leftarrow g^{\alpha_t}$	$\xleftarrow{\quad c \quad}$	$c \xleftarrow{R} C$
$\alpha_z \leftarrow \alpha_t + \alpha c$	$\xrightarrow{\quad \alpha_z \quad}$	$g^{\alpha_z} \stackrel{?}{=} u_t \, . \, u^c$

Schnorr's protocol is sometimes called a "proof of knowledge" of a discrete logarithm.

Identification Scheme to Signature Scheme

Let G be a cyclic group of prime order $\mathbb{Z}_q$, g is a generator of G

kenGen. $\alpha_t \xleftarrow{R} \mathbb{Z}_q \; u_t \leftarrow g^{\alpha_t}$ sk = α , vk = $u = g^{\alpha}$

Signer

$$\alpha_t \xleftarrow{R} \mathbb{Z}_q$$

$$u_t \leftarrow g^{\alpha_t}$$

$$c \leftarrow H(m, u_t)$$

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

$$\sigma = (u_t, \alpha_z) \longrightarrow$$

Verifier

$$c \leftarrow H(m, u_t)$$

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

Sigma Protocols

Prover
P(x,y)

Verifier
V(y)

Generate Commitment t $\xrightarrow{\ t\ }$

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

Generate response z $\xrightarrow{\ z\ }$ Output accept/reject

Special soundness: Let (P; V) be a Sigma protocol for $\mathscr{R} \subseteq \mathscr{X} \times \mathscr{X}$. We say that (P, V) provides special soundness if there is an efficient deterministic algorithm Ext, called a witness extractor, with the following property: whenever Ext is given as input a statement $y \in \mathscr{Y}$, and two accepting conversations (t, c, z) and (t,' c', z') with $c \neq c'$, algorithm Ext always outputs $x \in X$ such that $(x, y) \in \mathscr{R}$ (i.e., x is a witness for y).

ZK-SNARK

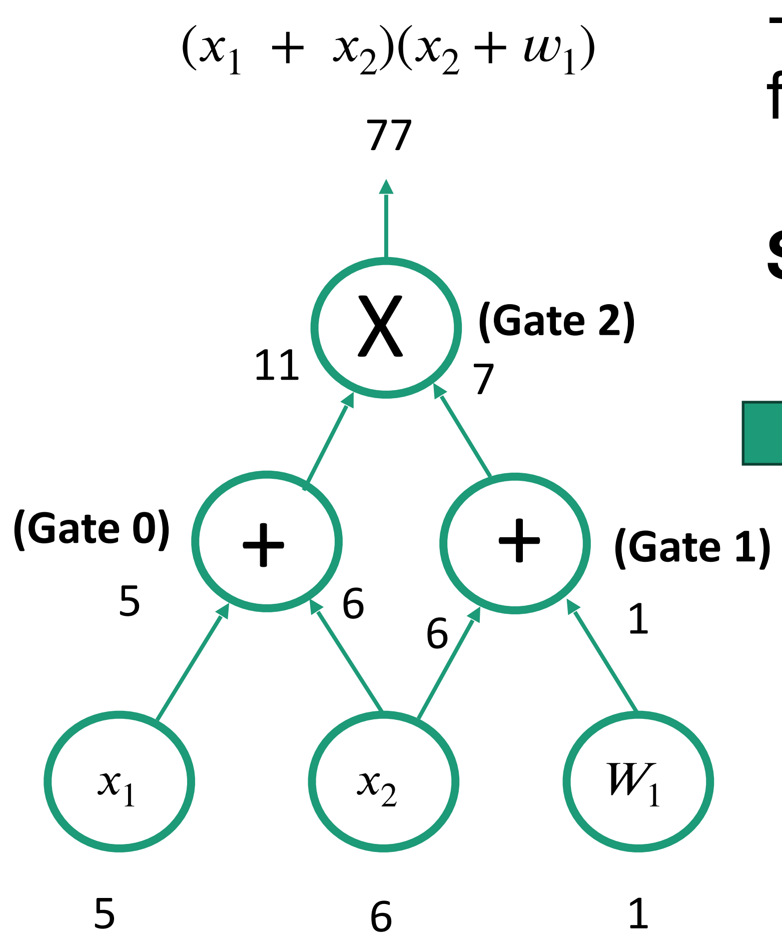
- **Succinct Non-Interactive Argument of Knowledge (SNARK)**
- **Succinct :** Proofs are short and can be verified much faster than verifying the claim from the original witness (e.g., the colouring)
- **Argument** is just a In a proof, "computationally sound proofs”.
- In proofs: soundness holds against 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 $\mathbb{F} = \{0,1,\cdots,p-1\}$

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

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



Size of the circuit is C = #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 $\mathbb{F} = \{0,1,\cdots,p-1\}$

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

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

Size of the circuit is C = #gates

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

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

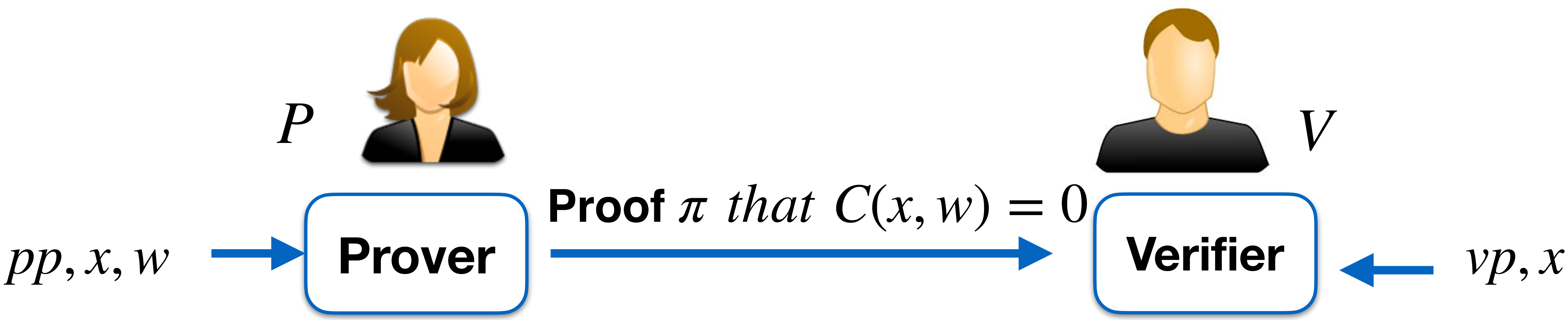
C_{sig}(pk,m,σ) : Outputs 0 if σ is a valid ECDA signature with P $\neq$0 otherwise

Preprocessing NARKS

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

x is the public input $\qquad$ w is the secret witness

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

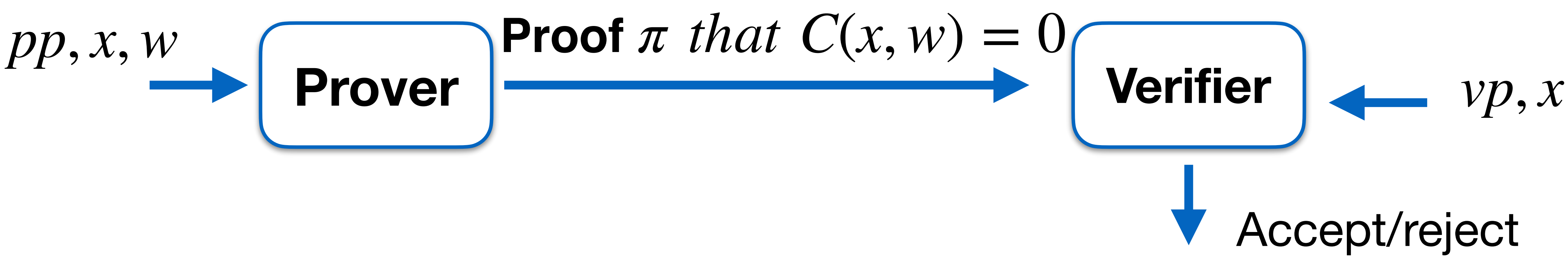


Accept/reject

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

$$S(C) \rightarrow (pp, vp)$$
$$P(pp, x, w) \rightarrow \pi$$
$$V(vp, x, \pi) \rightarrow Accept/Reject$$

Definition



Completeness:

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

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 $O\log(|C|)$

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

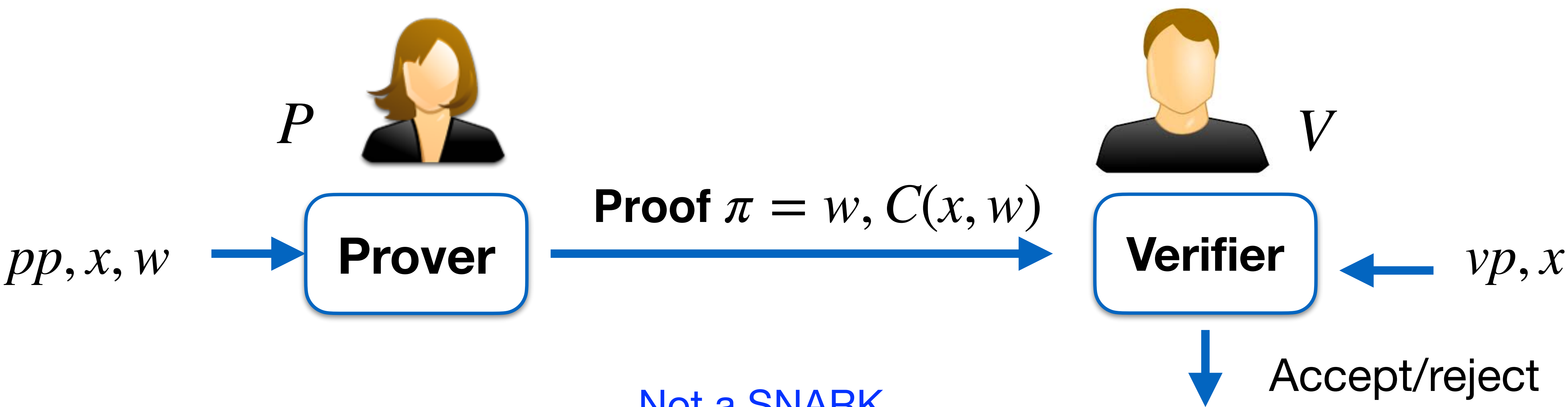
Verification is fast $O_\lambda(|x|, \log(|C|))$

SNARKS

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

x is the public input $\quad$ w is the secret witness

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



Not a SNARK

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

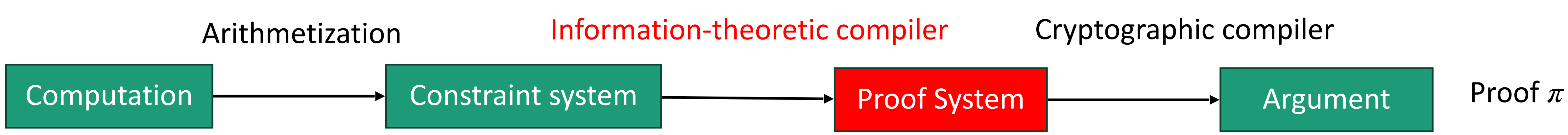
SNARKs

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:



1. R1CS (linear combination)
2. PlonKish (custom gate + lookup + permutation)
3. AIR (transition constraint + boundary constraint)

Polynomial-Interactive Oracle Proof (IOP) + Polynomial Commitment Scheme

General Constructions of SNARKs

Polynomial Commitment Schemes (PCS)

Cryptographic Object

+

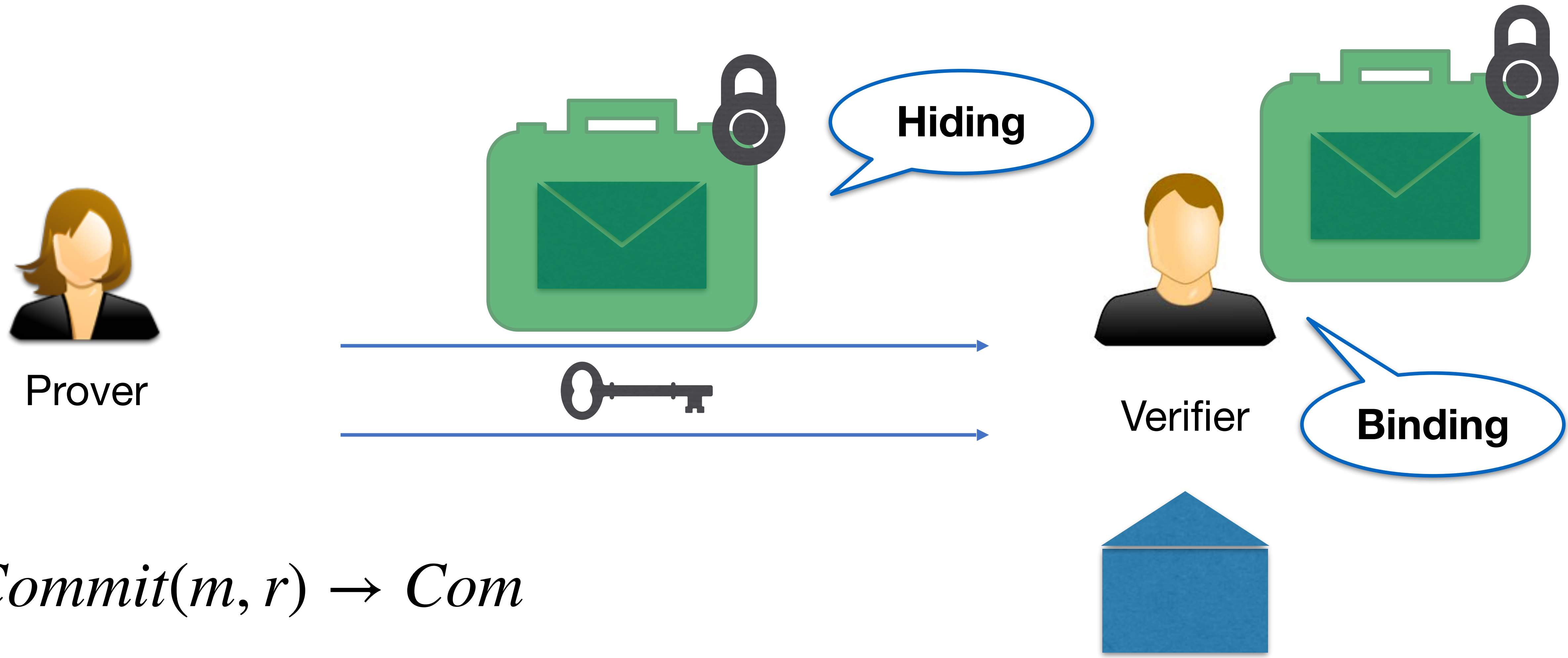
Interactive oracle proofs (IOP)

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 $H : \mathscr{M} \times \mathscr{R} \rightarrow T$

- $Commit(m, r) : Com = H(m, r)$

- $Verify(m, Com, r) : Accept \quad if \;\; Com = H(m, r)$

- Should have hiding and binding property for suitable Hash function

Committing to a Function

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

Prover

Verifier

Choose $f \in \mathscr{F}$

$r \leftarrow R$

$Com_f \leftarrow Com(f, r)$ →

X ←

$y = f(x), \pi$

Proof π, y →

Prove $y = f(x), f \in \mathscr{F}$

$Verify(y, \pi)$=Accept/Reject

Functional Commitments

- Vector Commitments: Commit of a vector $\overrightarrow{v} = (v_1, v_2, \cdots, v_d) \in \mathbb{F}_p^d$, open $f_{\overrightarrow{v}}(i) = v_i$
- Polynomial Commitments: Commit to a univariate function $f(X) = \mathbb{F}_p^{\leq d}[X]$, d is the degree of the polynomial
- Multilinear Commitment: Commit to a multivariate function $f(X) = \mathbb{F}_p^{\leq 1}[X_1, X_2, \cdots, x_k]$
- Inner Product Commitments (Inner product arguments IPA): Commit to $\overrightarrow{v} \in \mathbb{F}_p^d$ Open an inner product $f_{\overrightarrow{v}}(\overrightarrow{u}) = (\overrightarrow{v}, \overrightarrow{u})$

Polynomial Commitments

Prover

Verifier

Com: Commits to polynomial $f(X) = \mathbb{F}_p^{\leq d}[X]$

X

$y = f(x), \pi$

Proof π, y

x,y are public

Prove $y = f(x), deg(f) \leq d$

$Verify(y, \pi)$= Accept/Reject

Examples:

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

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

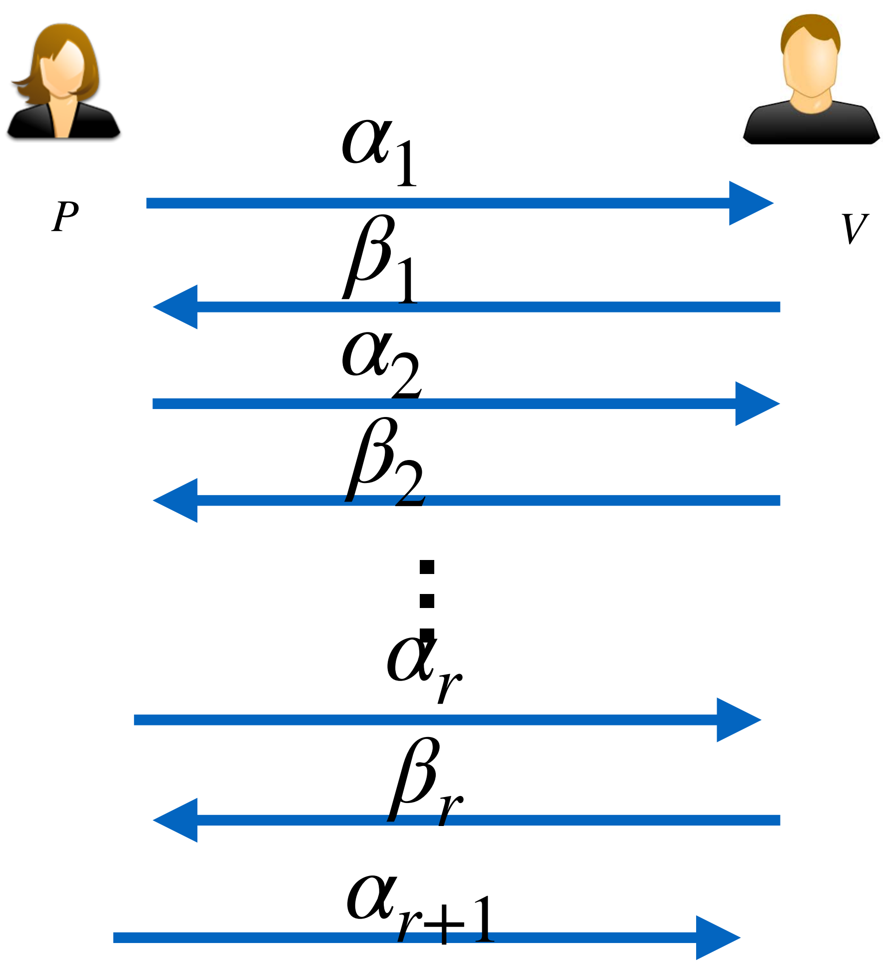
Elliptic curves: Bulletproofs, Short proofs but verifier time is O(d)

Important Observation from Schwartz–Zippel Lemma

- Let $f(X) = \mathbb{F}_p^{\leq d}[X]$ 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 servers 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 $f(r) = g(r) \implies f(r) - g(r) = 0, \, then \; f = g \quad w.h.p.$
- Tests two polynomials are equal

Fiat-Shamir Transform

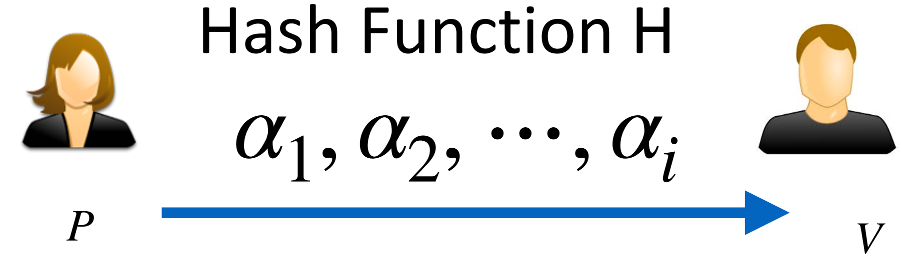
Public-coin interactive argument



Each β_i uniformly random

Non-Interactive argument

Hash Function H



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

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

$$\vdots$$

$$\beta_i = H(x, \alpha_1, \cdots, \alpha_i)$$

Non-interactive Polynomial Equality Testing

Prover $P(pp, x, w)$

Verifier

Random Oracle

$r \leftarrow H(x)$

$\text{Com}_f, \text{Com}_g$

$y \leftarrow f(r), y' \leftarrow g(r)$ $\xrightarrow{y, y', \pi_f, \pi_g}$ $VerifyV(vp, x)$= Accept/Reject

Generate π_f, π_g

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 $x \in \mathbb{F}_p^n$
- $\mathscr{F} - IOP$ is a proof system that proves $\exists w : C(x, w) = 0$
- $Setup(C) \rightarrow pp, vp = (f_0^o, f_1^o, \cdots, f_s^o)$ (oracles for function in $\mathscr{F}$ 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

$r_1 \xleftarrow{\$} R$

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

r_2

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

r_t

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

Example

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

Prover:

$f(Z) := \Pi_{w \in W}(Z - w)$

$g(Z) := \Pi_{x \in X}(Z - x)$

$q(Z) := f/g \in \mathbb{F}_p^{\leq d}[X]$

Oracles for f, q $\longrightarrow$

$\longleftarrow$ r

Verifier:

$g(Z) = \Pi_{x \in X}(Z - x)$

$r \xleftarrow{\$} \mathbb{F}_p$

Query $w \leftarrow f(r)$

Query $q' \leftarrow q(r)$

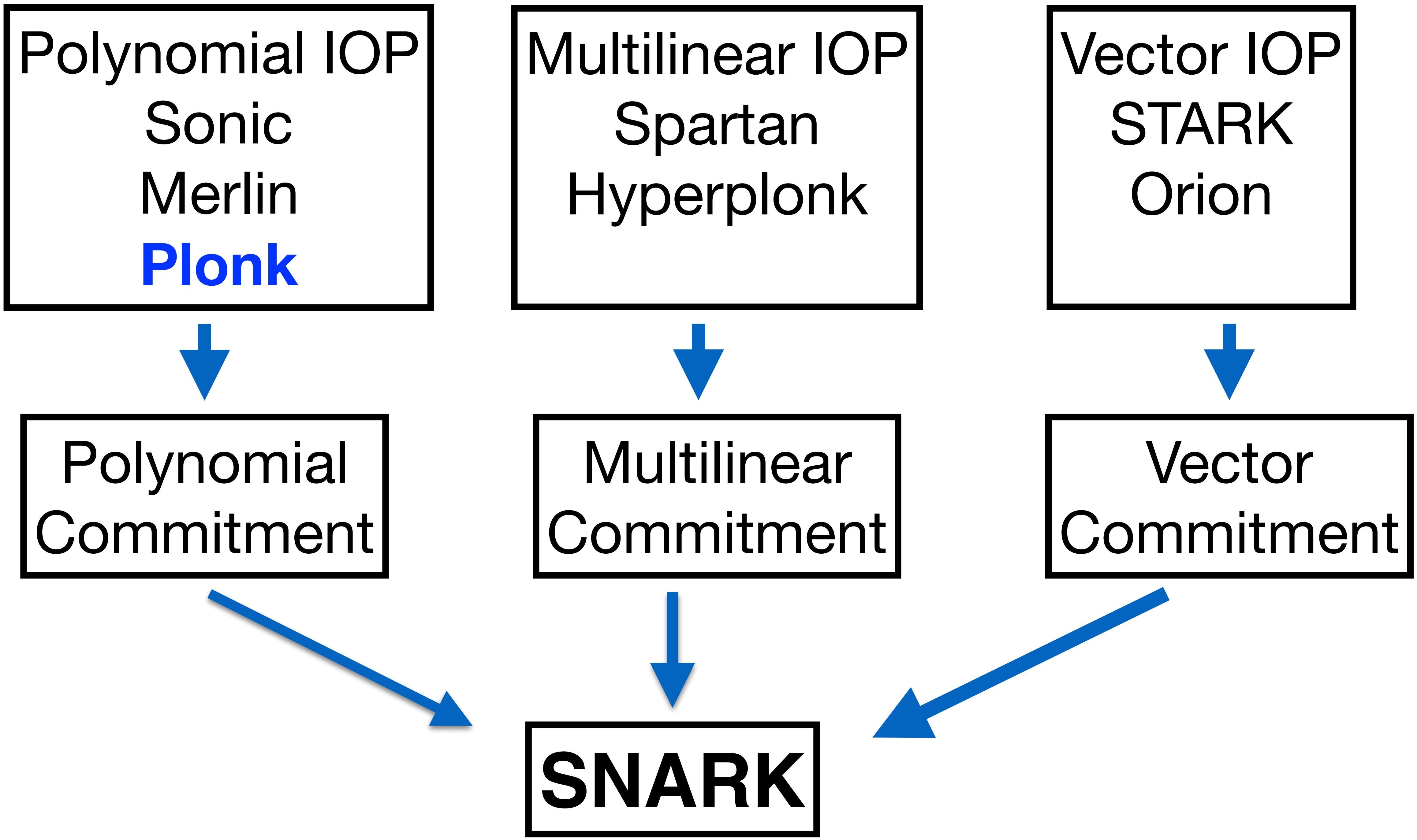
Compute $x \leftarrow g(r)$

Accept if $x . q' = w$

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

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

Candidate IOPs

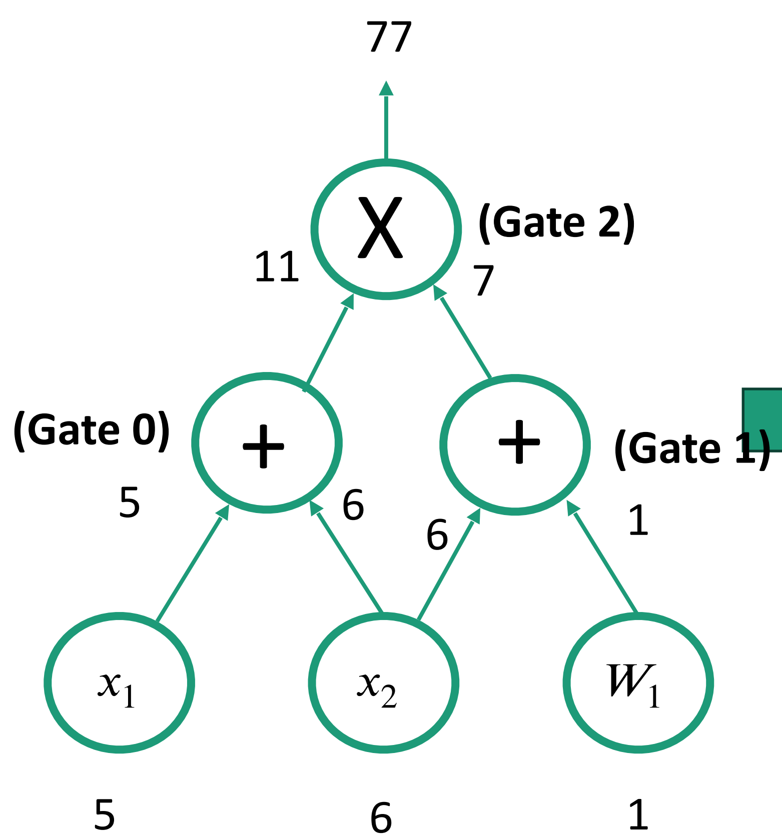


Proof Gadgets

- 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 $\Sigma_{a \in S} f(a) = 0$
- Product-Check: prove that $\Pi_{a \in S} f(a) = 1 \ldots$

Plonk High level Idea

$(x_1 + x_2)(x_2 + w_1)$



|c|=#gates, 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

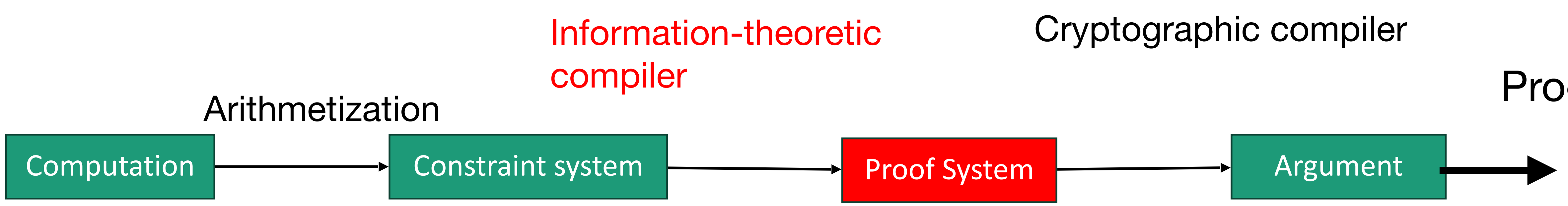
Left input   Right input   Output

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



1. R1CS (linear combination)
2. PlonKish (custom gate + lookup + permutation)

Polynomial-Interactive Oracle Proof (IOP) + Polynomial Commitment Scheme

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

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

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

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|

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

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

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”