How to Commit to a Function

Dan Boneh Wilson Nguyen Alex Ozdemir

Stanford University

https://eprint.iacr.org/2021/1342

Cryptography: application areas

Traditional applications of Cryptography: **Internet Security**

⇒ secure communication: encryption, key exchange, etc.

More recent applications of Cryptography: **Blockchain applications**

⇒ data integrity: signatures, commitments, ZK proofs, …

The Science of Blockchains:

Economics, Distributed Systems, Cryptography, Prog. Lang.

The Science of Blockchain Conference 2022 (SBC'22)

Arrillaga Alumni Center, Stanford University

Aug. 29 - 31, 2022

Committing to a Function

Recall: basic commitment schemes

Two algorithms:

- $commit(m, r) \rightarrow$ ***com*** $(m \in \mathcal{M}, \quad r$ random in $\mathcal{R})$
- $verify(m,$ ***com***$, r) \rightarrow$ accept or reject

Basic properties:

- **binding**: cannot produce two valid openings for ***com***.
- **hiding**: ***com*** reveals nothing about committed data m

A standard construction

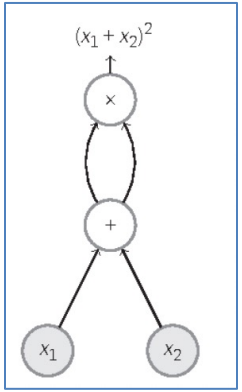
Fix a hash function $H: \mathcal{M} \times \mathcal{R} \rightarrow C$

commit(m, r):	**com** $\leftarrow H(m, r)$
verify$(m,$ **com**$, r)$:	accept if **com** = $H(m, r)$

Hiding and Binding for a suitable function H

Committing to a function

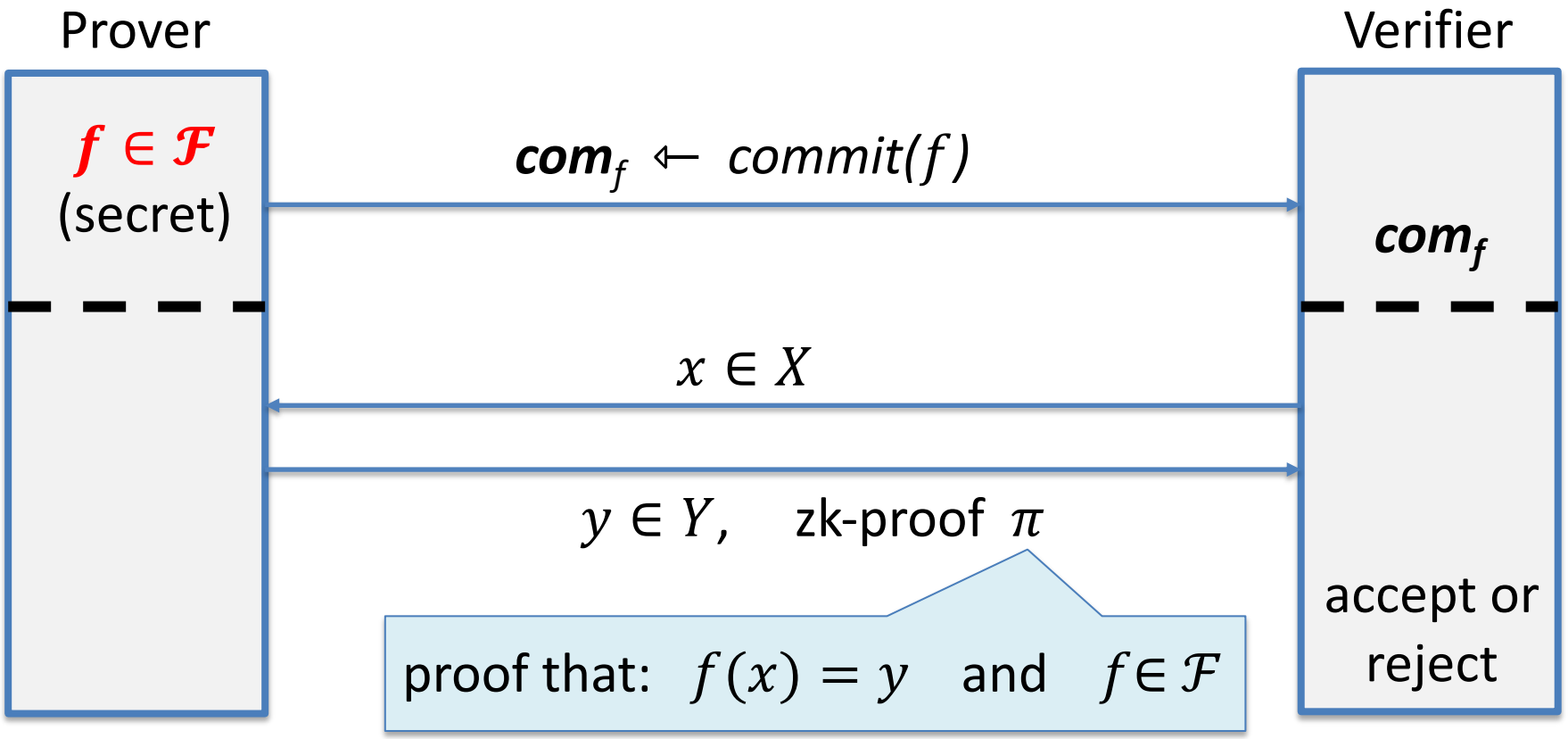
Fix a family of functions $\mathcal{F} = \{f: X \to Y\}$



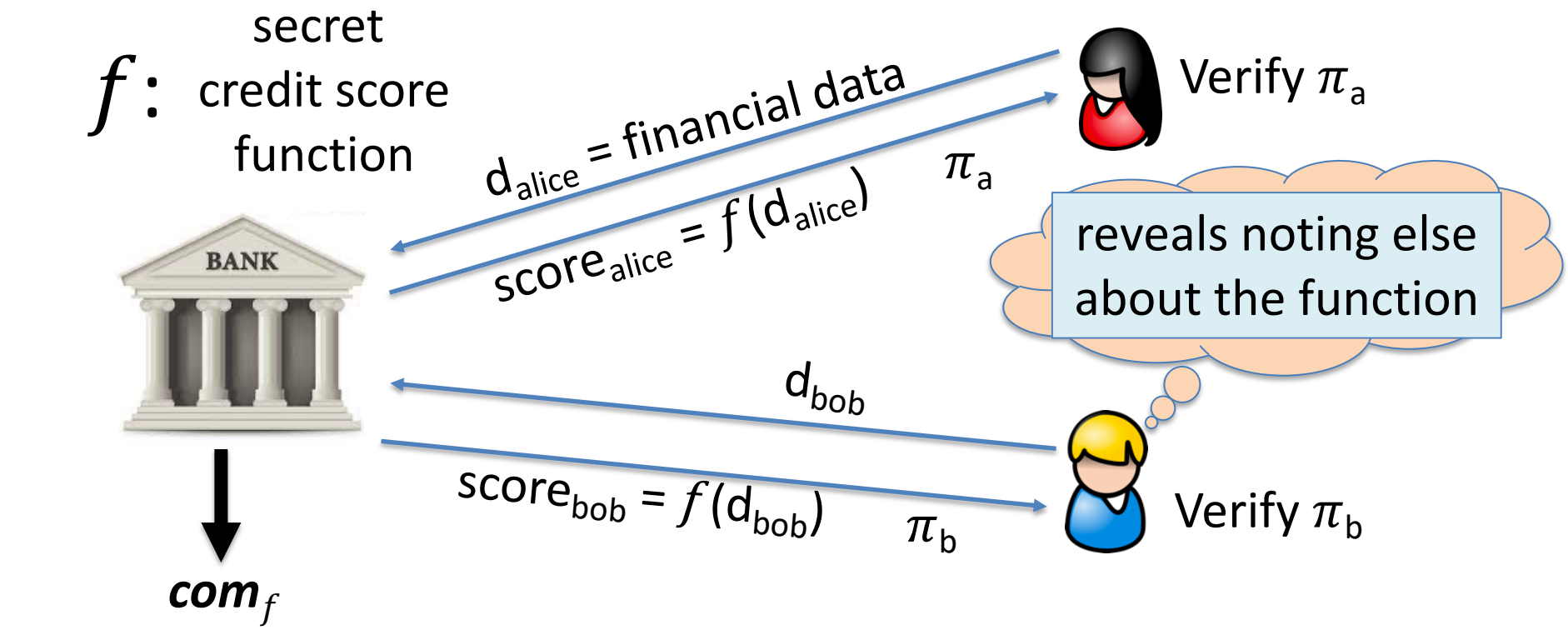
Examples:

- $\mathbb{F}_p^{(\leq d)}[X]$: all polynomials in $\mathbb{F}_p[X]$ of degree ≤ d
- $\mathcal{C}_s = \{f: \mathbb{F}_p^n \to \mathbb{F}_p\}$: arithmetic circuits of size ≤ s

Committing to a function



Applications: a verifiably uniform policy



… but is the function fair ??

Did Bank commit to a "fair" function ?

Suppose the bank commits to f. Is f fair ??

An unfair f(user,data):

if (user == 'dan') return("perfect score")

else return $\hat{f}$(data)

What is a fair function?

- Several criteria developed by the field of *algorithmic fairness*

Is the function “fair” ?

Option 1: **Trusted auditor(s)**

- Checks f and signs $\boldsymbol{com}_f$ if f is “fair”

Option 2: Zero knowledge **proof of fairness**

- Attach ZK proof π to $\boldsymbol{com}_f$. Anyone can verify proof.
- A good direction for future research

… back to committing to a function: syntax

A (function-hiding) **functional commitment** scheme for $\mathcal{F}$:

- *setup*$(\lambda) \rightarrow pp$, outputs public parameters pp
- *commit*$(pp, f, r) \rightarrow \boldsymbol{com_f}$ commitment to $f \in \mathcal{F}$ with $r \in \mathcal{R}$

  a **hiding** and **binding** commitment scheme for the set $\mathcal{F}$

- *eval*(Prover P, verifier V): for a given $\boldsymbol{com_f}$ and $x \in X$, $y \in Y$:

  P$(pp, f, x, y, r) \rightarrow$ succinct proof π

  V$(pp, \boldsymbol{com_f}, x, y, \pi) \rightarrow$ accept/reject

$f(x) = y$ and $f \in \mathcal{F}$ and *commit*$(pp, f, r) = \boldsymbol{com_f}$

Examples of functional commitments

Polynomial commitment schemes: [KZG'10, …]

- A functional commitment for the family $\mathcal{F} = \mathbb{F}_p^{(\leq d)}[X]$

  $\Longrightarrow$ prover commits to a univariate polynomial,
  later, can verifiably open the polynomial at any $x \in \mathbb{F}_p$

Our question: how to commit to a general arithmetic circuit?

An example polynomial commitment

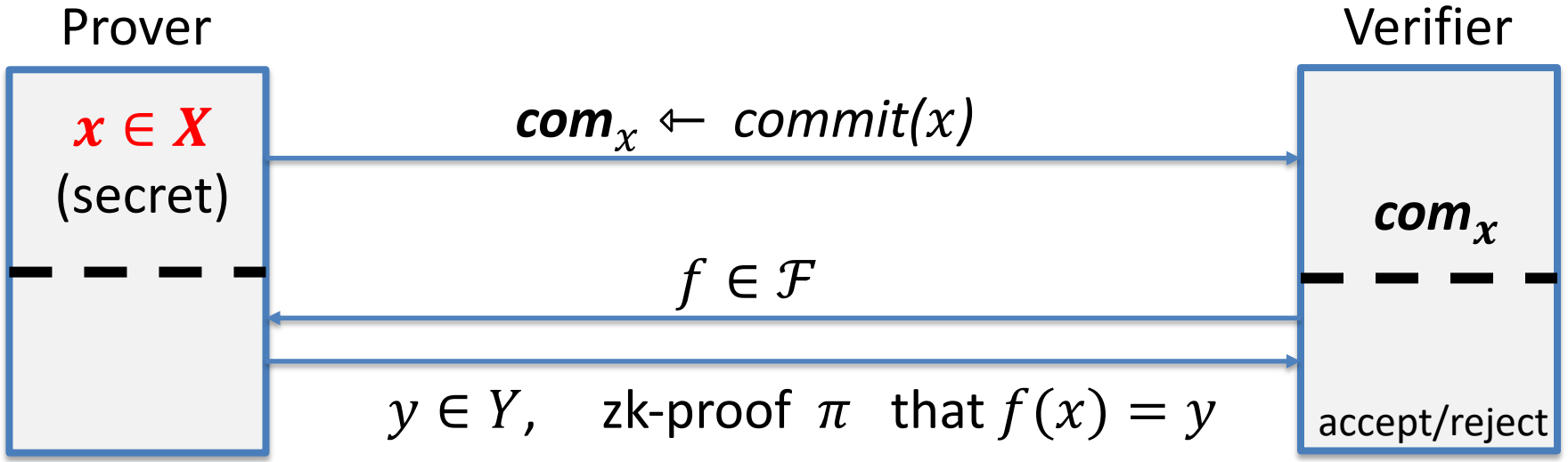
Dory: (eprint/2020/1274)

- transparent setup: no secret randomness in setup
- $\boldsymbol{com}_{\mathrm{f}}$ is a single group element (independent of degree d)
- eval proof size for $f \in \mathbb{F}_p^{(\leq d)}[X]$ is O(log d) group elements
- eval verify time is O(log d) Prover time: $O(d)$

Can we do the same for a general arithmetic circuit?

Note: a dual concept [Libert-Romana-Yung'16, …]

Input-hiding functional commitment for $\mathcal{F}$:

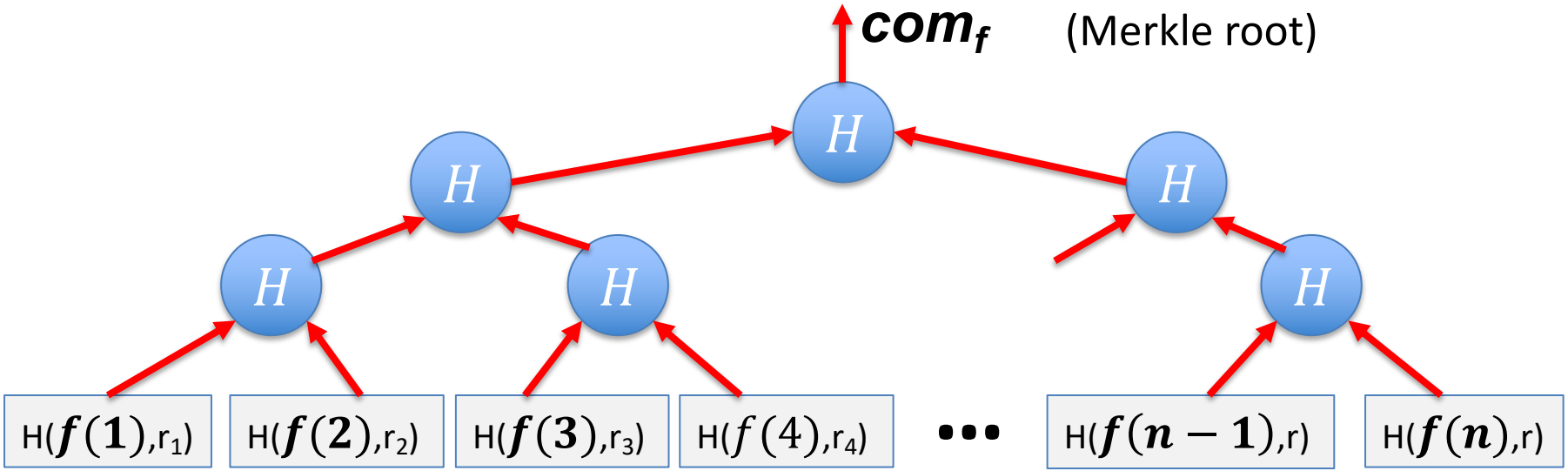


Efficiently implied by a zk-SNARK. Papers focus on weaker assumptions.

Constructing function-hiding functional commitments

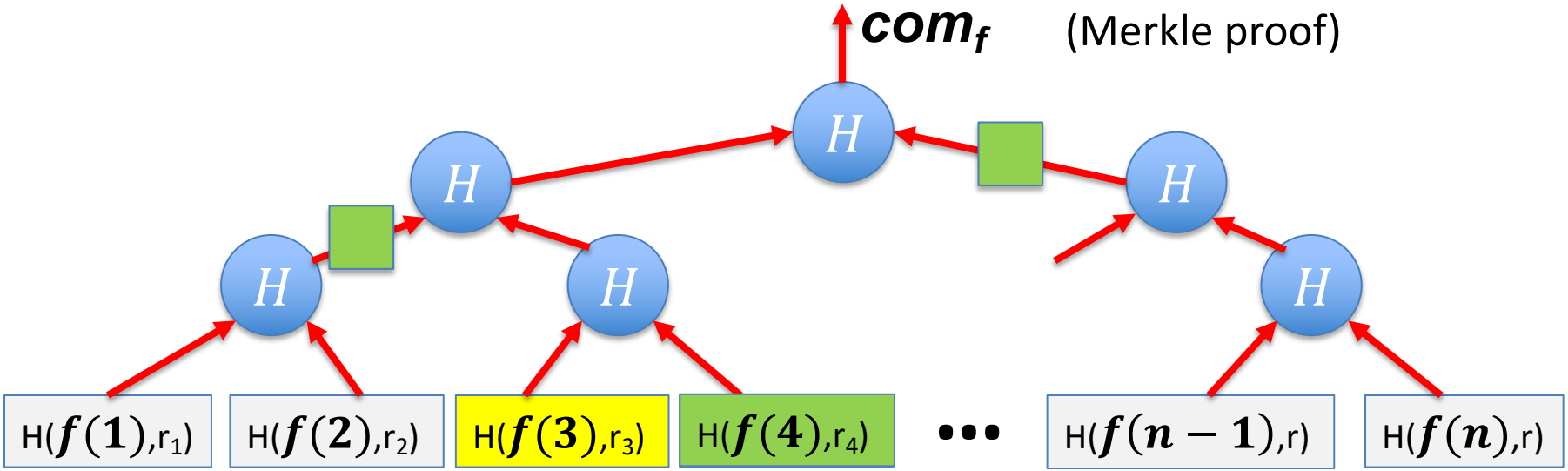
A trivial construction: small domain

$f: D \to \mathbb{F}_p$ where $|D| = \{1,2,\ldots,n\}$



A trivial construction: small domain

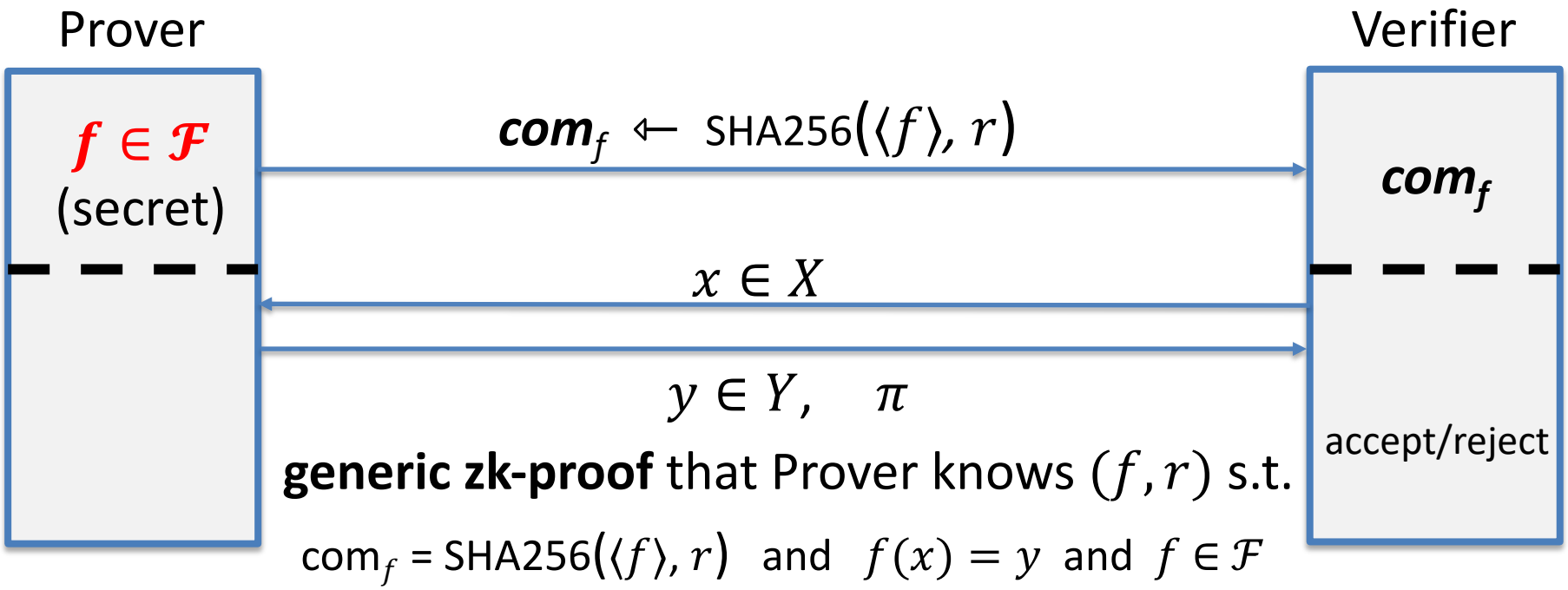
$f: D \to \mathbb{F}_p$ where $|D| = \{1,2,\ldots,n\}$



To prove $y = f(3)$ with a proof of $\log n$ size

What about functions with super-polynomial domain size??

Another trivial construction



The problem: inefficient! Can we do better?

Review: a preprocessing SNARK for a circuit C

A **succinct preprocessing argument system** is a triple (S, P, V):

- **S**$(C) \rightarrow$ public pa… over and verifier

proof that $C(\boldsymbol{st}, \boldsymbol{w}) = 0$

- **P**$(S_p, \boldsymbol{st}, \boldsymbol{w}) \rightarrow$ **short** proof π ;

$$|\pi| = O(\log(|C|), \lambda)$$

- **V**$(S_v, \boldsymbol{st}, \boldsymbol{\pi})$ **fast to verify** ;

time(V) = $O(|st|, \log(|C|), \lambda)$

short "summary" of C

why preprocess C??

Review: a preprocessing SNARK for a circuit C

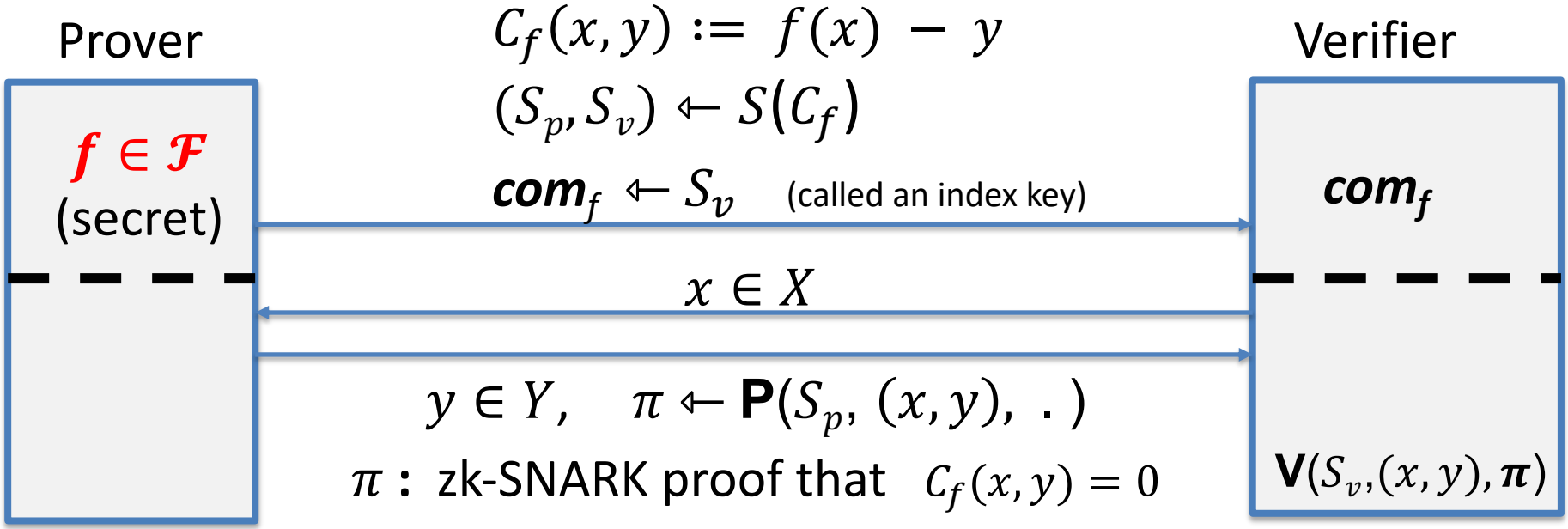
A **succinct preprocessing argument system** is a triple (S, P, V):

- **S**$(C) \rightarrow$ public parameters (S_p, S_v) for prover and verifier
- **P**$(S_p, \boldsymbol{st}, \boldsymbol{w}) \rightarrow$ **short** proof π ; $|\pi| = O(\mathbf{log}(|C|), \lambda)$
- **V**$(S_v, \boldsymbol{st}, \boldsymbol{\pi})$ **fast to verify** ; time(V) = $O(|st|, \mathbf{log}(|C|), \lambda)$

SNARK: (S, P, V) is **complete**, **knowledge sound**, and **succinct**

zk-SNARK: (S, P, V) is a SNARK and is **zero knowledge**

(Universal) preprocessing zk-SNARK $\Longrightarrow$ functional commitment?



Problems: (1) S_v may not be a <u>hiding</u> commitment to f

(2) what if C_f is malformed? (i.e. $f \notin \mathcal{F}$)

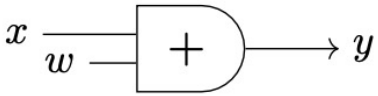
The problem

Problem 1: S_v may not be a hiding commitment to f

- Easily solved by adding a blinding step in $S(C_f, r)$ (see paper)

Problem 2: committed $C_f(x, y) := f(x) - y$ is malformed ($f \notin \mathcal{F}$)

hidden inputs

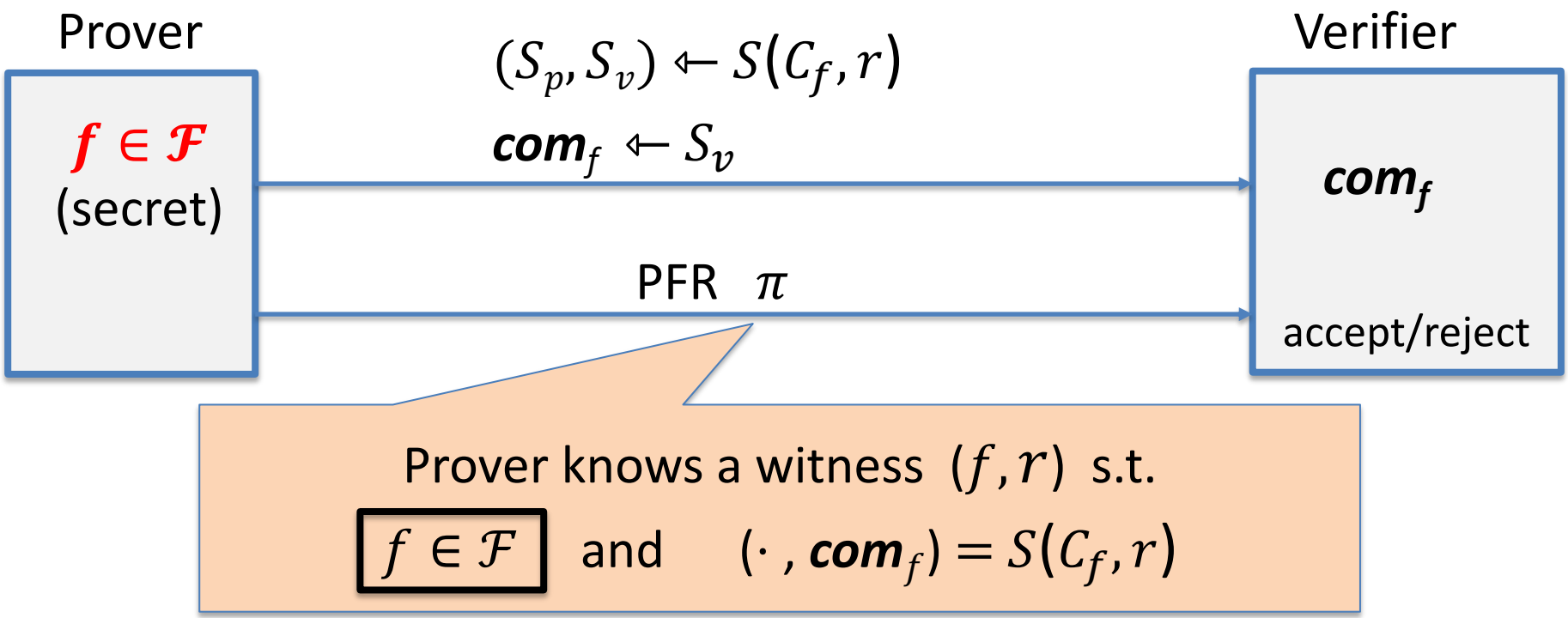
$f(x)$: 

more generally

Suppose: $C_f(0, y_0) = C_f(0, y_1) = 0$

Prover can prove $f(0) = y_0$ and $f(0) = y_1$!!

What to do? Proof of function relation (PFR)



Challenge: build efficient PFR for common preprocessing zk-SNARKs

The Marlin and Plonk zk-SNARKs [CHMMVW'19, GWC'19]

Plonk: circuit C_f is represented as an arithmetic circuit of size $\leq s$

- S_v: a poly. commitment to two univariate polynomials
- **Our PFR proves**: S_v commits to a DAG with no hidden inputs

Marlin: circuit C_f is represented as an **R1CS program** (A, B, C)

- S_v: a poly. commitment to nine univariate polynomials (unoptimized)
- **Our PFR (11KB) proves**: S_v commits to matrices A, B, C so that for every x there is a unique satisfying y

Marlin: Proof of function relation

R1CS program for a circuit C_f:

represent C_f as matrices $A, B, C \in \mathbb{F}_p^{n \times m}$ s.t.

$C_f(x, y) = 0$ iff $\exists w$: $(Av) \circ (Bv) = (Cv)$ where $v = \begin{bmatrix} x \\ w \\ y \end{bmatrix}$

Thm: (A, B, C) defines a function $f: X \nrightarrow Y$ whenever

$\boldsymbol{A}, \boldsymbol{B}$ **are lower triangular** and $\boldsymbol{C}$ **is diagonal**

$\Rightarrow$ need zk-SNARK that a committed matrix is lower triangular / diagonal

Marlin: Proof of function relation

Committing to a (sparse) matrix $A \in \mathbb{F}_p^{n \times m}$:

- Three polynomials $row,\ col,\ val \in \mathbb{F}_p[X]$

  $row(i),\ col(i),\ val(i)$: encode i-th non-zero element

- S_v contains poly. commitments to these three polynomials

Proof of diagonality: require that $commit(col) = commit(row)$

Proof of lower triangular: prove that $col(i) \leq row(i)$ for all $i \in \Omega$

How?

See paper

https://eprint.iacr.org/2021/1342

End result

com_f

Ensure orgs make decisions uniformly across a population

Lots more to do:

- Efficient ZK proof of *algorithmic fairness*
- Efficient proof of function relation for other SNARKs

THE END

https://eprint.iacr.org/2021/1342