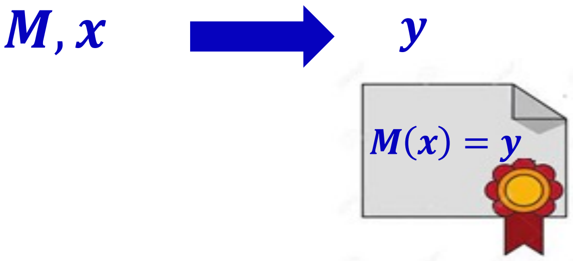
The Evolution of Proofs in Computer Science and the Existence of SNARGs

Lecture 17

Efficient Verification of Computation



$\boldsymbol{M, x} \longrightarrow \boldsymbol{y}$

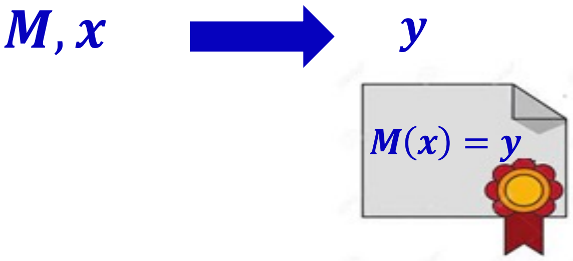
$\boldsymbol{M(x) = y}$

Completeness: If $M(x) = y$ within time T, then a valid certificate for $y = M(x)$ is computable in time $\approx T$, of size $\ll T$, and verifiable in time $\ll T$.

Soundness: If $M(x) \neq y$ then it is "practically impossible" to generate a valid certificate.

If adv succeeds then it can break a cryptographic assumption

Efficient Verification of Computation



$\boldsymbol{M, x} \longrightarrow \boldsymbol{y}$

$\boldsymbol{M(x) = y}$

Completeness: If $M(x) = y$ within time T, then a valid certificate for $y = M(x)$ is computable in time $\approx T$, of size $\ll T$, and verifiable in time $\ll T$.

Soundness: If $M(x) \neq y$ then it is "practically impossible" to generate a valid certificate.

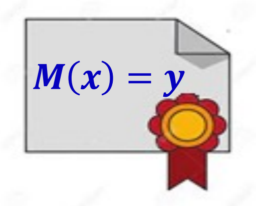
Needed! Otherwise, such a scheme would imply $\boldsymbol{DTIME(T) \subseteq NTIME(\ll T)}$

If adv succeeds then it can break a cryptographic assumption

Efficient Verification of Computation

Common Random String (CRS)

$M, x \longrightarrow y$



Completeness: If $M(x) = y$ within time T, then a valid certificate for $y = M(x)$ time $\approx T$, of size $\ll T$, and verifiable in time $\ll T$.

computational

Soundness: If $M(x) \neq y$ then it is "practically impossible" to generate a valid certificate.

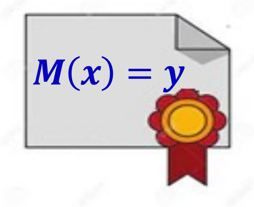
Needed! Otherwise, such a scheme would imply $DTIME(T) \subseteq NTIME(\ll T)$

If adv succeeds then it can break a cryptographic assumption

Efficient Verification of Computation

Common Random String (CRS)

$$\boldsymbol{M, x} \longrightarrow \boldsymbol{y}$$



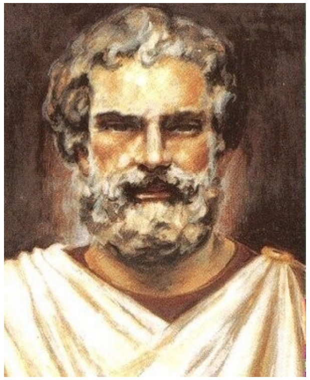
Completeness: If $M(x) = y$ within time T, then a valid certificate for $y = M(x)$ time $\approx T$, of size $\ll T$, and verifiable in time $\ll T$.

computational

Soundness: If $M(x) \neq y$ then it is “practically impossible” to generate a valid certificate.

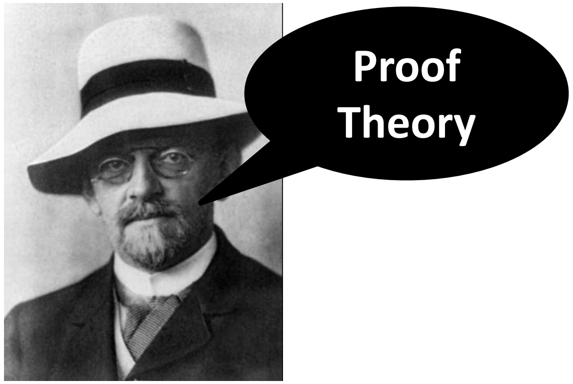
Succinct Non-interactive Argument (SNARG)

What is a Proof?

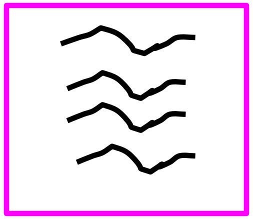


Thales (600BCE)

Euclid (300BCE)



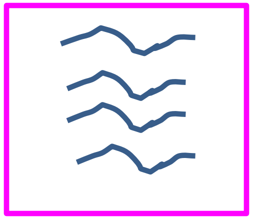
Hilbert (19th century)



Zero-Knowledge Proofs

[Goldwasser-Micali-Rackoff85]

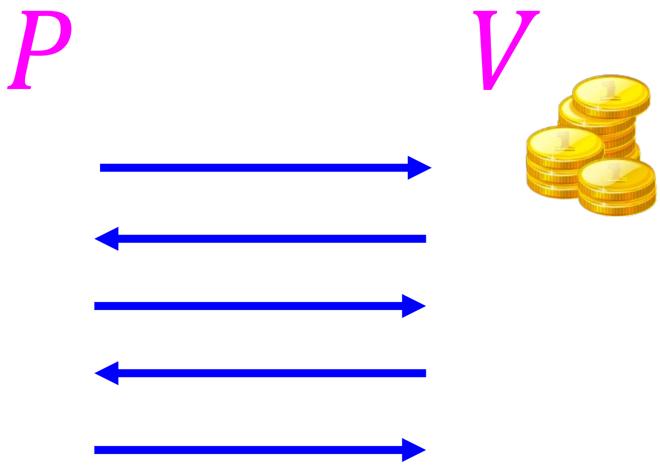
Proofs that reveal **no information** beyond the validity of the statement



This is information

Interactive Proofs

[Goldwasser-Micali-Rackoff85]

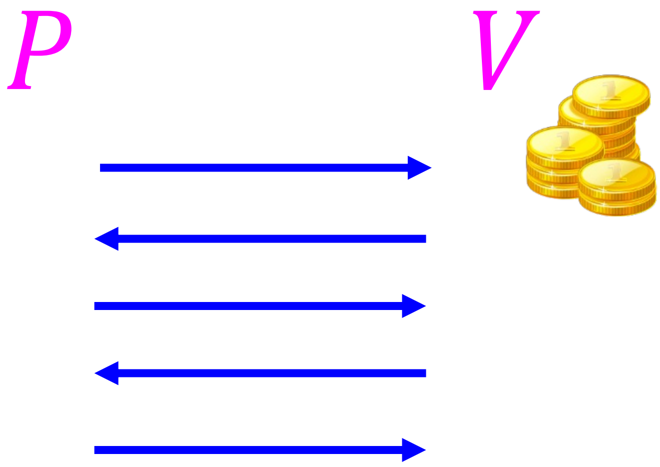


Completeness: P can convince V to accept a true statement with probability 1 (over V's coin tosses)

Soundness: A prover cannot convince V to accept a false statement except with exponentially small probability (over V's coin tosses)

Interactive Proofs

[Goldwasser-Micali-Rackoff85]

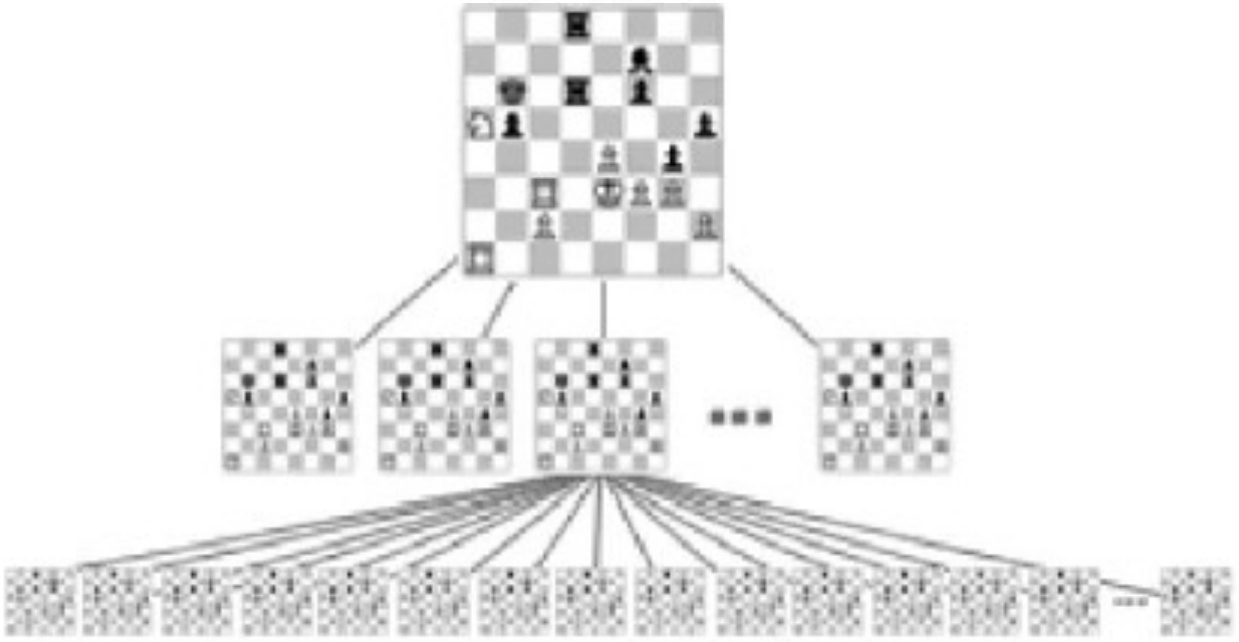


Theorem [Goldreich-Micali-Wigderson87]: Every statement that has a classical proof has zero-knowledge interactive proof, assuming one-way functions exist.

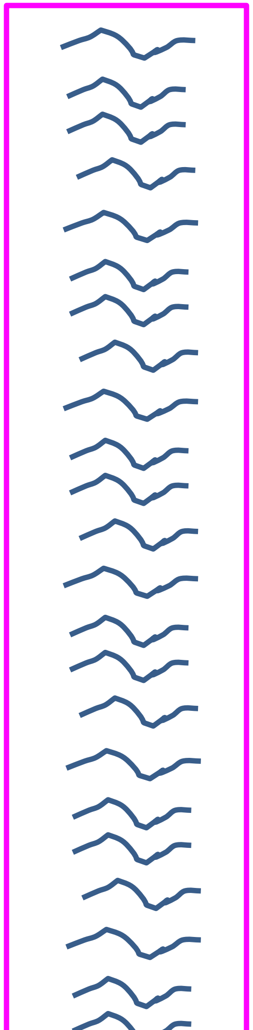
Interactive Proofs are Shorter!

[Lund-Fortnow-Karloff-Nissan90, Shamir90]

Example: Chess



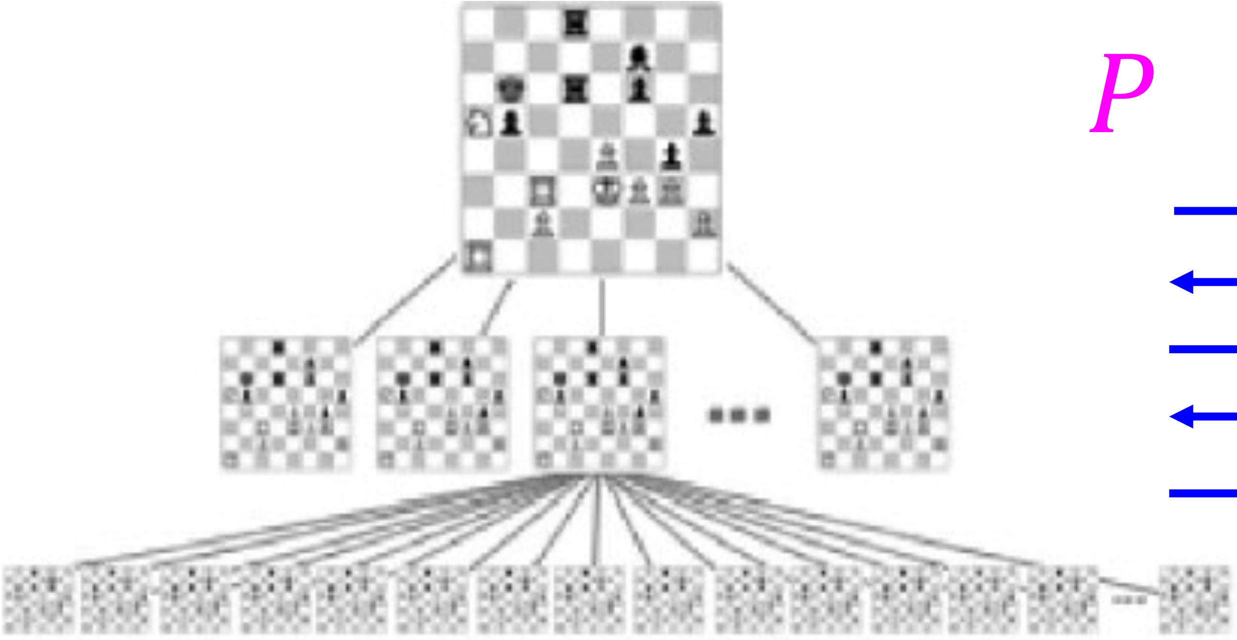
Classical proof



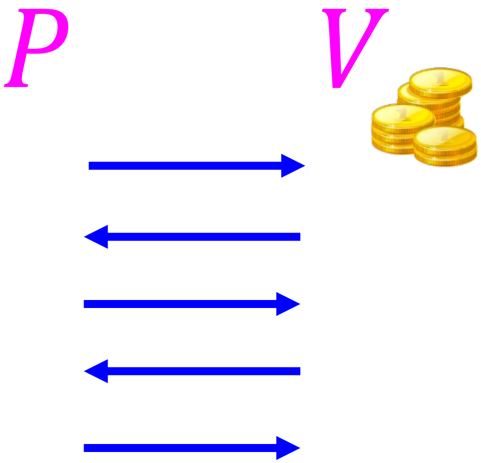
Interactive Proofs are Shorter!

[Lund-Fortnow-Karloff-Nissan90, Shamir90]

Example: Chess



Succinct interactive proof



Theorem:

$IP = PSPACE$

verification time ≈ space

[BenOr-Goldwasser-Kilian-Wigderson]:

Do there exist ZK proofs unconditionally?

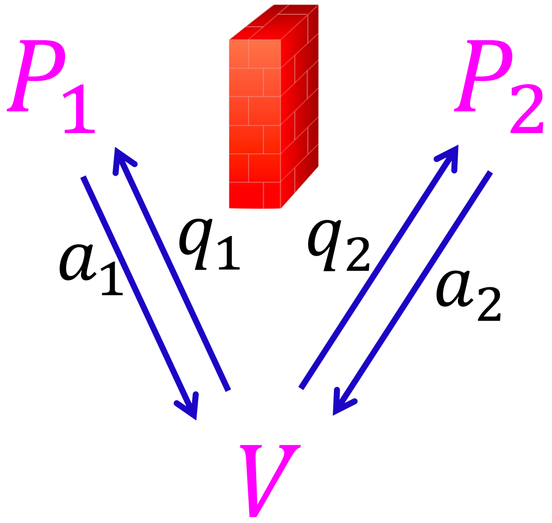
(Without assuming one-way functions)

Not in general!

Unless the polynomial hierarchy collapses

Multi-Prover Interactive Proofs

[BenOr-Goldwasser-Kilian-Wigderson88]

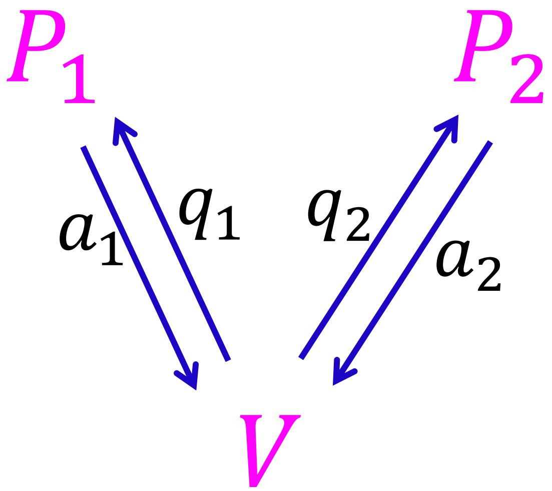


Completeness: P_1 and P_2 can convince V to accept a true statement with probability 1 (over V's coin tosses)

Soundness: Non-communicating provers cannot convince V to accept a false statement, except with exponentially small probability (over V's coin tosses)

Multi-Prover Interactive Proofs

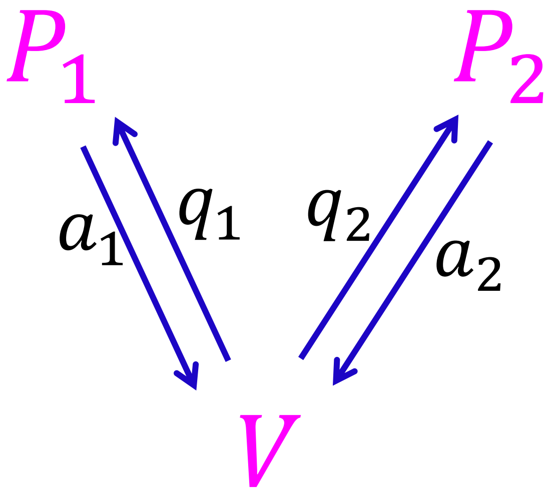
[BenOr-Goldwasser-Kilian-Wigderson88]



Theorem: Every statement that has a proof has an **unconditional zero-knowledge** proof!

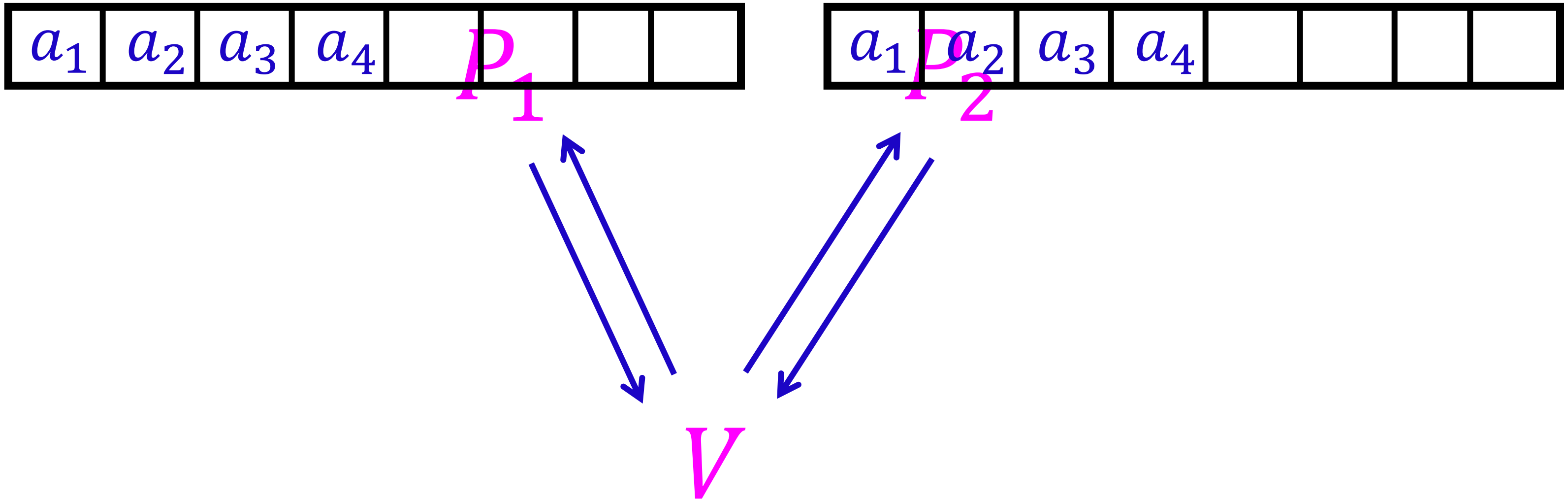
Multi-Prover Interactive Proofs

[BenOr-Goldwasser-Kilian-Wigderson88]



Theorem [Babai-Fortnow-Lund90]: Any proof can be made exponentially shorter with a 2-prover interactive proof!

[Fortnow-Rompel-Sipser88]:



a_1 a_2 a_3 a_4

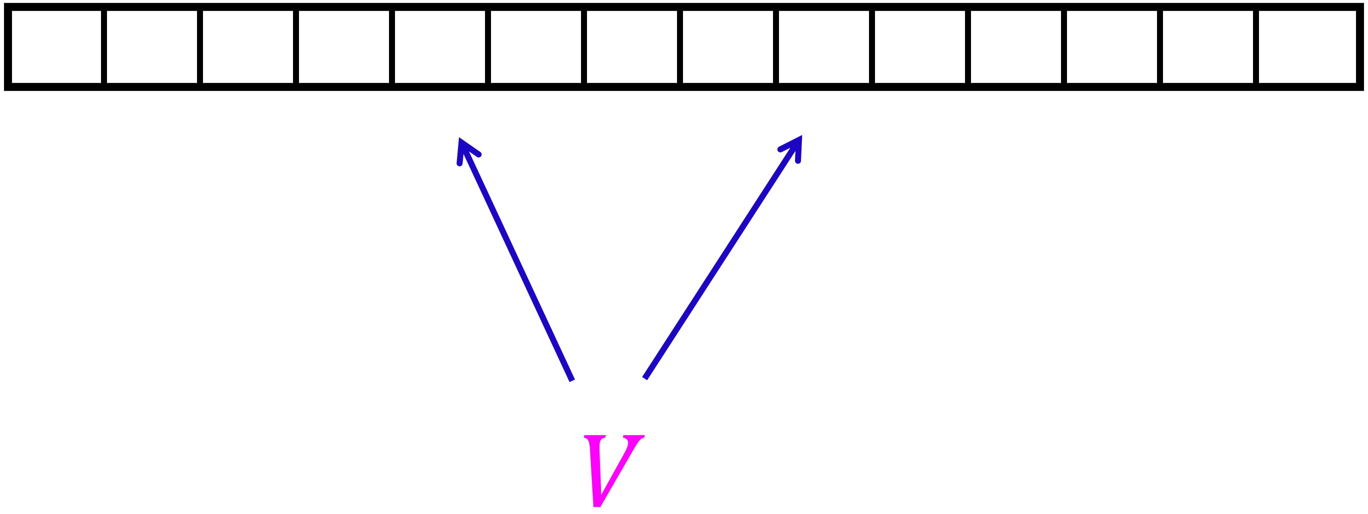
a_1 a_2 a_3 a_4

P_1

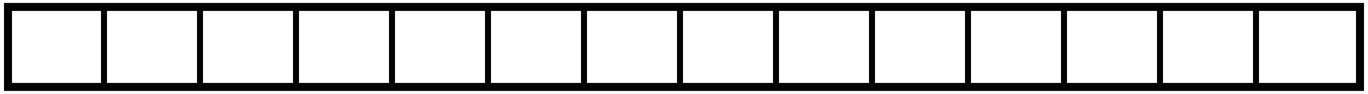
P_2

V

Probabilistically Checkable Proofs



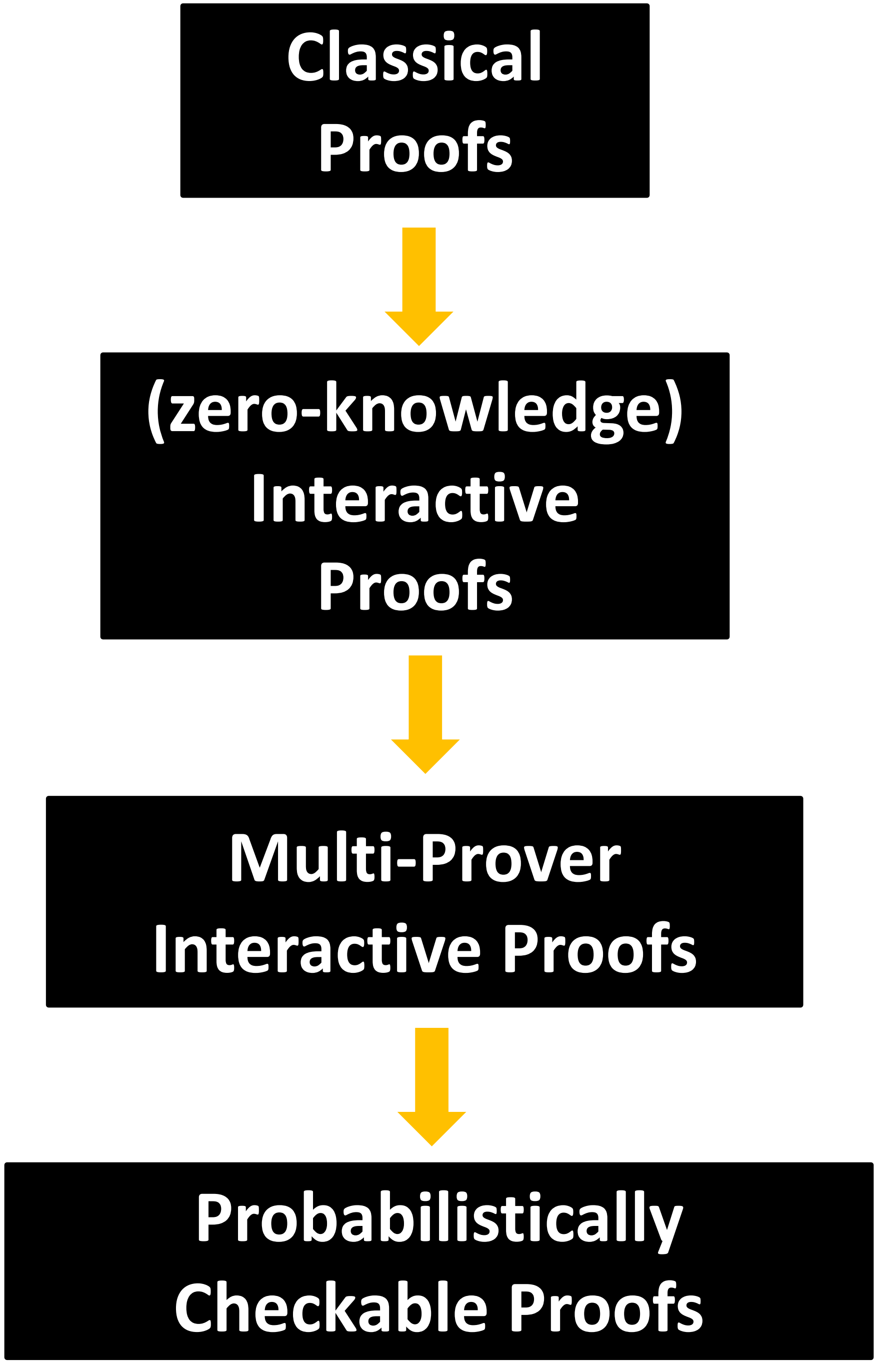
Probabilistically Checkable Proofs



[Feige-Goldwasser-Lovasz-Safra-Szegedy91, Babai-Fortnow-Levin-Szegedy91, Arora-Safra92, Arora-Lund-Mutwani-Sudan-Szegedy92]

PCP Theorem:

Every proof can be converted to a probabilistically checkable one (of almost same size) that can be verified by reading only constant number of its bits.



Classical Proofs

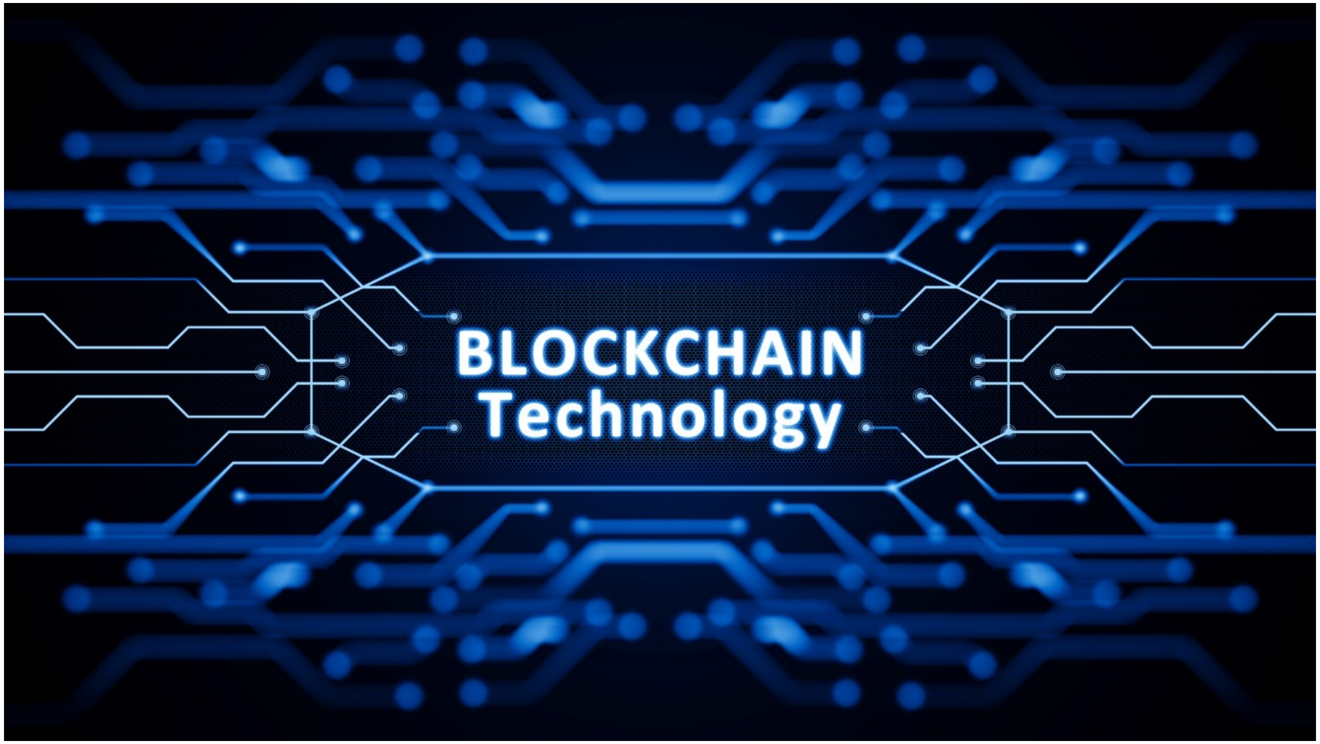
(zero-knowledge) Interactive Proofs

Multi-Prover Interactive Proofs

Probabilistically Checkable Proofs

Fast Forward to Today’s Reality

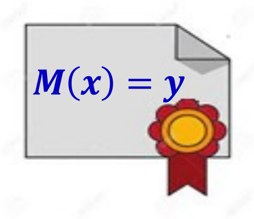
Fast Forward to Today's Reality



Succinct Non-Interactive Argument (SNARG)

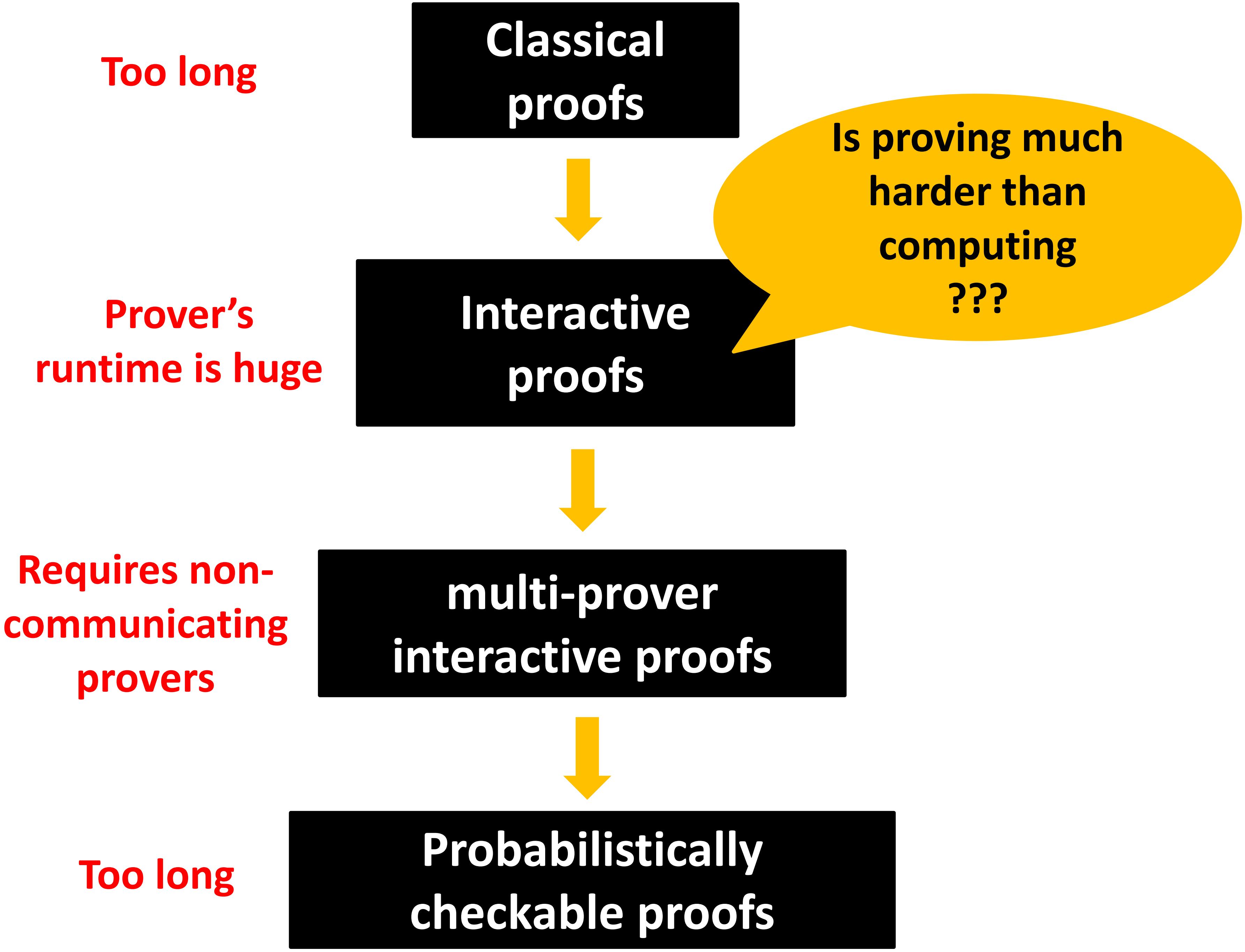
Common Random String (CRS)

$$M, x \longrightarrow y$$



Succinct Proofs

A **succinct proof** that my transaction is valid!



Too long

Classical proofs

Is proving much harder than computing ???

Prover’s runtime is huge

Interactive proofs

Requires non-communicating provers

multi-prover interactive proofs

Too long

Probabilistically checkable proofs

Doubly Efficient Interactive Proofs

A doubly efficient Interactive proof for proving correctness of a computation satisfies:

Prover runtime ≈ computation runtime

Verifier runtime ≈ |input|

Focus: Polynomial-time computations!

Doubly Efficient Interactive Proofs

[Goldwasser-K-Rothblum08]:

Doubly efficient interactive proofs for **depth bounded** computations

(communication complexity grows with the **depth**)

[Reingold-Rothblum-Rothblum15]:

Doubly efficient interactive proofs for **space bounded** computations

(communication complexity grows with the **space**, and with $\mathbf{time}^{\epsilon}$, ϵ small const.)

Non-Interactive Delegation scheme for all functions!

[Kilian92, Micali94]

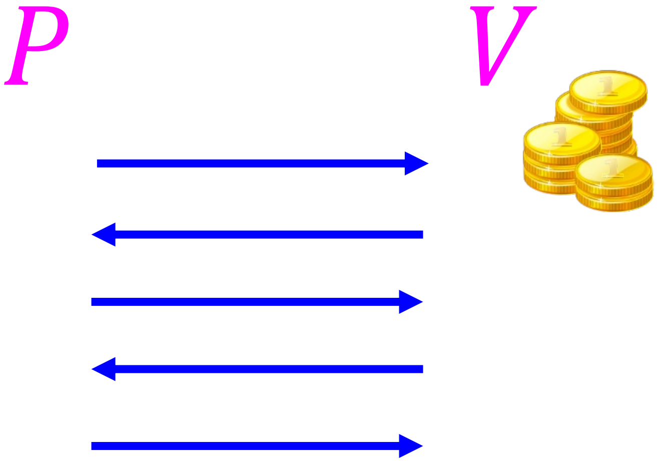
Non-Interactive Delegation scheme for all functions!

[Kilian92, Micali94]

Relax soundness to hold only against **polynomial time adversaries**

Interactive Proofs

[Goldwasser-Micali-Rackoff85]



P V

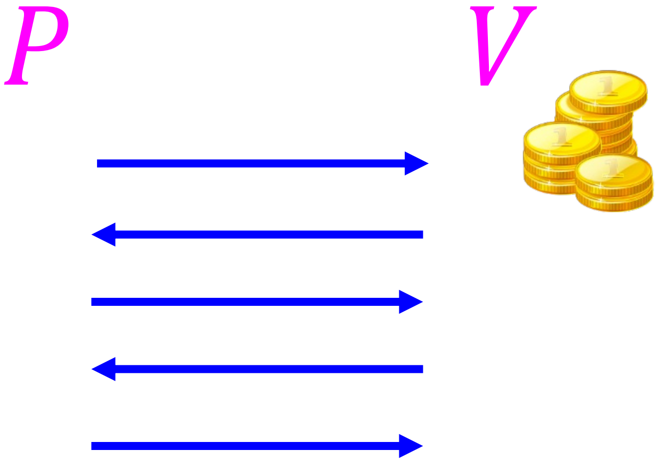
Completeness: P can convince V to accept a true statement with probability 1 (over V's coin tosses)

Soundness: A prover cannot convince V to accept a false statement except with exponentially small probability (over V's coin tosses)

Computationally-Sound Interactive Proofs

[Brassard-Chaum-Creapeau88]

Arguments



Computationally bounded

convince V to accept a true statement with

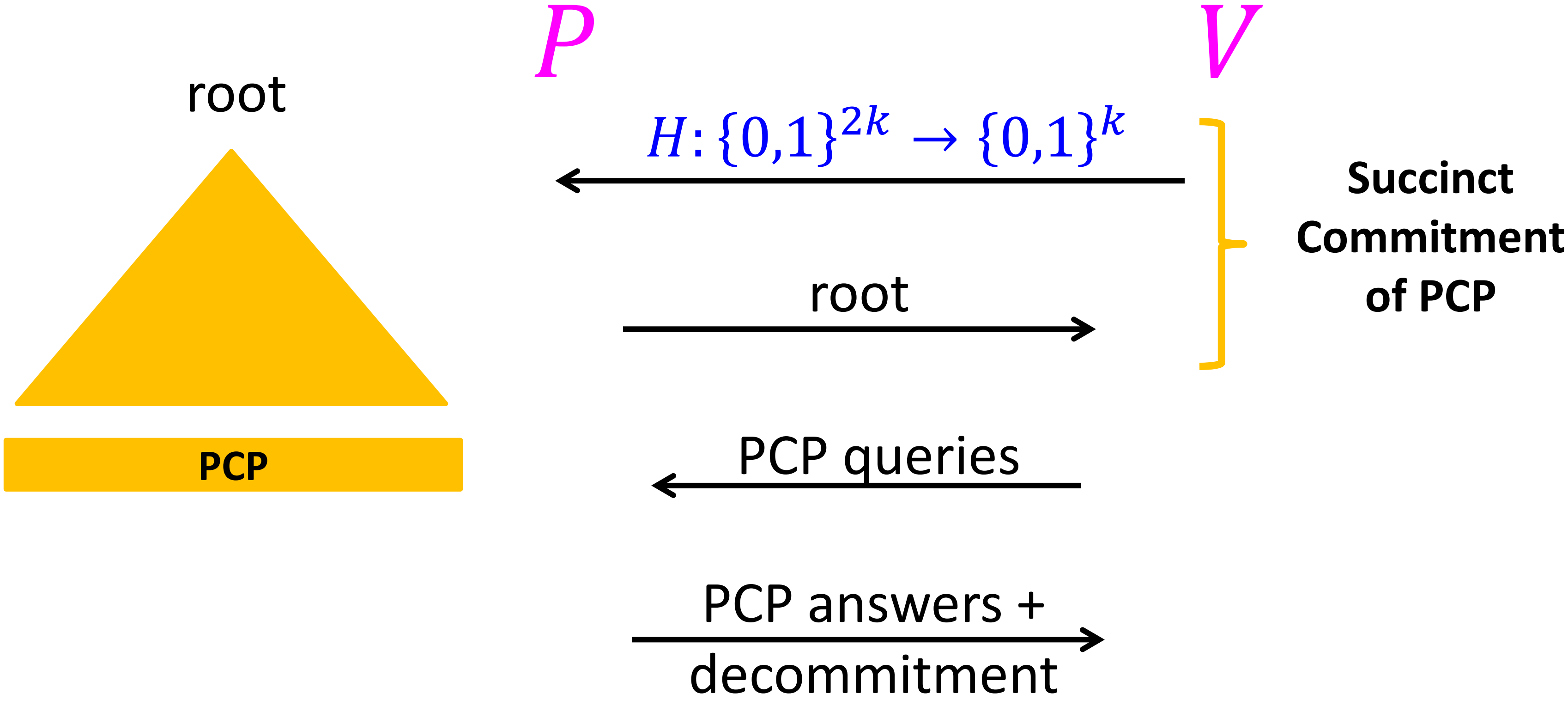
s coin tosses)

Soundness: A prover cannot convince V to accept a false statement except with exponentially small probability (over V's coin tosses)

Succinct Interactive Arguments

[Kilian92, Micali94]

Convert any **PCP** into a **succinct interactive argument**



root

PCP

P

V

$H: \{0,1\}^{2k} \rightarrow \{0,1\}^{k}$

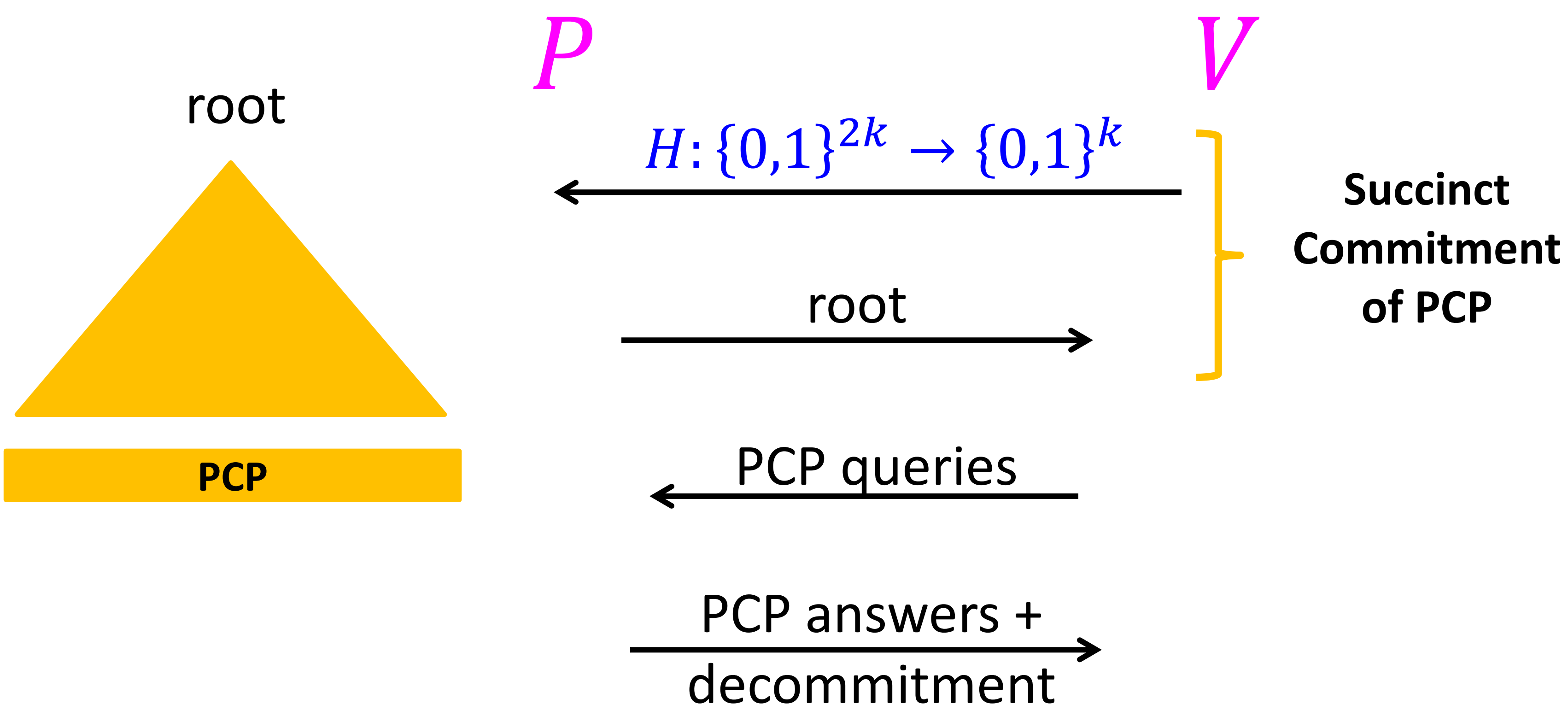
root

Succinct Commitment of PCP

PCP queries

PCP answers + decommitment

Theorem: This scheme is **sound** against cheating provers that **cannot find collisions in *H***
(i.e., cannot find $x_1 \neq x_2$ such that $H(x_1) = H(x_2)$)



root

PCP

P

V

$H: \{0,1\}^{2k} \rightarrow \{0,1\}^k$

root

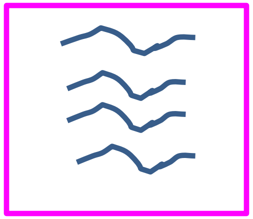
Succinct Commitment of PCP

PCP queries

PCP answers + decommitment

Succinct Non-Interactive Arguments (SNARGs)

Common random string (CRS)

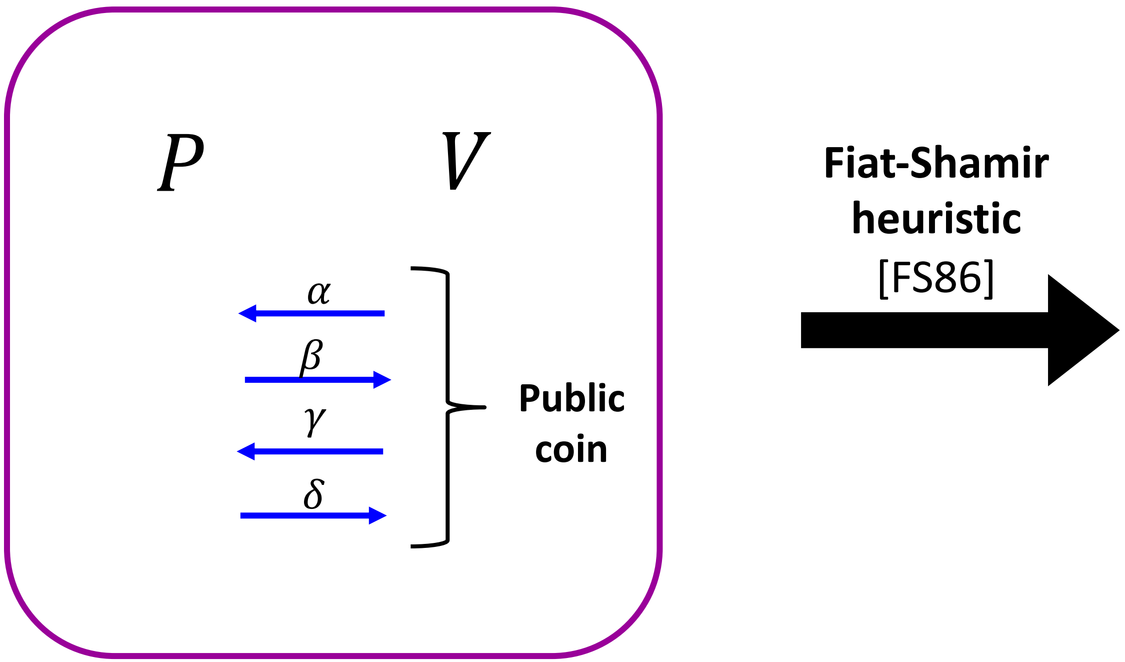


Guarantee: Given CRS, it is computationally hard to generate a proof of a false statement

Succinct Non-Interactive Arguments (SNARGs)

Apply **Fiat-Shamir Paradigm** to **eliminate interaction** from interactive schemes

From Succinct Interactive Schemes to SNARGs



P V

α

β

γ

δ

Public coin

Fiat-Shamir heuristic

[FS86]

SNARG

$(\boldsymbol{\alpha}, \boldsymbol{\beta}, \boldsymbol{\gamma}, \boldsymbol{\delta})$

s.t. $\beta = H(\alpha)$ and $\delta = H(\alpha, \beta, \gamma)$

The (In)Security of the Fiat-Shamir Heuristic

[FS86]

Proposed as a heuristic for converting identification schemes into signature schemes.

In practice: ✓

In theory: ✗

One of the most widely used signature scheme (ECDSA) is based on this heuristic

The (In)Security of the Fiat-Shamir Heuristic

[FS86]

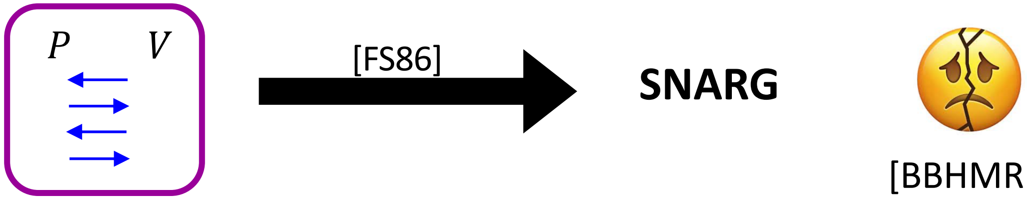
Proposed as a heuristic for converting identification schemes into signature schemes.

In practice: ✓

In theory: ✗

Computational soundness

First SNARG construction: Kilian92, Micali94



P V

[FS86]

SNARG

[BBHMR19]

The (In)Security of the Fiat-Shamir Heuristic

[FS86]

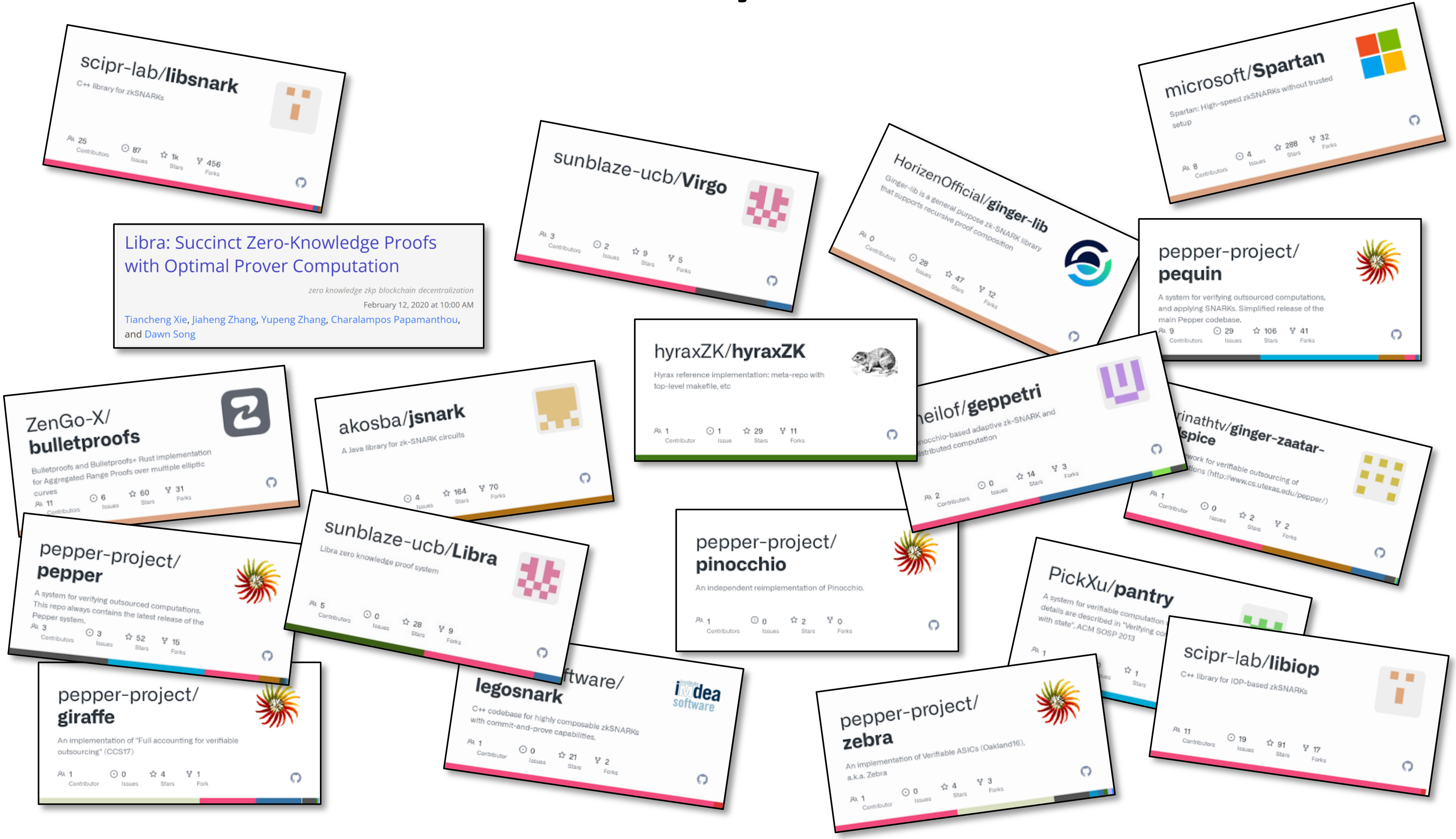
Is this heuristic secure when applied to statistically sound proofs??

Yes, under very strong cryptographic assumptions 🙂

Yes, for the GKR protocol under LWE or DDH 🙂

Yes, for some specific succinct interactive arguments under LWE!

From Theory to Practice



scipr-lab/**libsnark**
C++ library for zkSNARKs
25 Contributors · 87 Issues · 1k Stars · 456 Forks

Libra: Succinct Zero-Knowledge Proofs with Optimal Prover Computation
zero knowledge zkp blockchain decentralization
February 12, 2020 at 10:00 AM
Tiancheng Xie, Jiaheng Zhang, Yupeng Zhang, Charalampos Papamanthou, and Dawn Song

sunblaze-ucb/**Virgo**
3 Contributors · 2 Issues · 9 Stars · 5 Forks

HorizenOfficial/**ginger-lib**
Ginger-lib is a general purpose zk-SNARK library that supports recursive proof composition
0 Contributors · 28 Issues · 47 Stars · 12 Forks

microsoft/**Spartan**
Spartan: High-speed zkSNARKs without trusted setup
8 Contributors · 4 Issues · 288 Stars · 32 Forks

pepper-project/**pequin**
A system for verifying outsourced computations, and applying SNARKs. Simplified release of the main Pepper codebase.
9 Contributors · 29 Issues · 106 Stars · 41 Forks

hyraxZK/**hyraxZK**
Hyrax reference implementation: meta-repo with top-level makefile, etc
1 Contributor · 1 Issue · 29 Stars · 11 Forks

heilof/**geppetri**
nocchio-based adaptive zk-SNARK and
istributed computation
2 Contributors · 0 Issues · 14 Stars · 3 Forks

ZenGo-X/**bulletproofs**
Bulletproofs and Bulletproofs+ Rust implementation for Aggregated Range Proofs over multiple elliptic curves
11 Contributors · 6 Issues · 60 Stars · 31 Forks

akosba/**jsnark**
A Java library for zk-SNARK circuits
4 Issues · 164 Stars · 70 Forks

rinathtv/**ginger-zaatar-**
spice
work for verifiable outsourcing of
tions (http://www.cs.utexas.edu/pepper/)
1 Contributor · 0 Issues · 2 Stars · 2 Forks

pepper-project/**pepper**
A system for verifying outsourced computations. This repo always contains the latest release of the Pepper system.
3 Contributors · 3 Issues · 52 Stars · 15 Forks

sunblaze-ucb/**Libra**
Libra zero knowledge proof system
5 Contributors · 0 Issues · 28 Stars · 9 Forks

pepper-project/**pinocchio**
An independent reimplementation of Pinocchio.
1 Contributors · 0 Issues · 2 Stars · 0 Forks

PickXu/**pantry**
A system for verifiable computation
details are described in "Verifying co
with state", ACM SOSP 2013
1 · 1 Stars

pepper-project/**giraffe**
An implementation of "Full accounting for verifiable outsourcing" (CCS17)
1 Contributor · 0 Issues · 4 Stars · 1 Fork

ftware/
legosnark
C++ codebase for highly composable zkSNARKs with commit-and-prove capabilities.
1 Contributor · 0 Issues · 21 Stars · 2 Forks

pepper-project/**zebra**
An implementation of Verifiable ASICs (Oakland16), a.k.a. Zebra
1 Contributor · 0 Issues · 4 Stars · 3 Forks

scipr-lab/**libiop**
C++ library for IOP-based zkSNARKs
11 Contributors · 19 Issues · 91 Stars · 17 Forks

From Theory to Deployment

Zcash

STARKWARE

ethereum

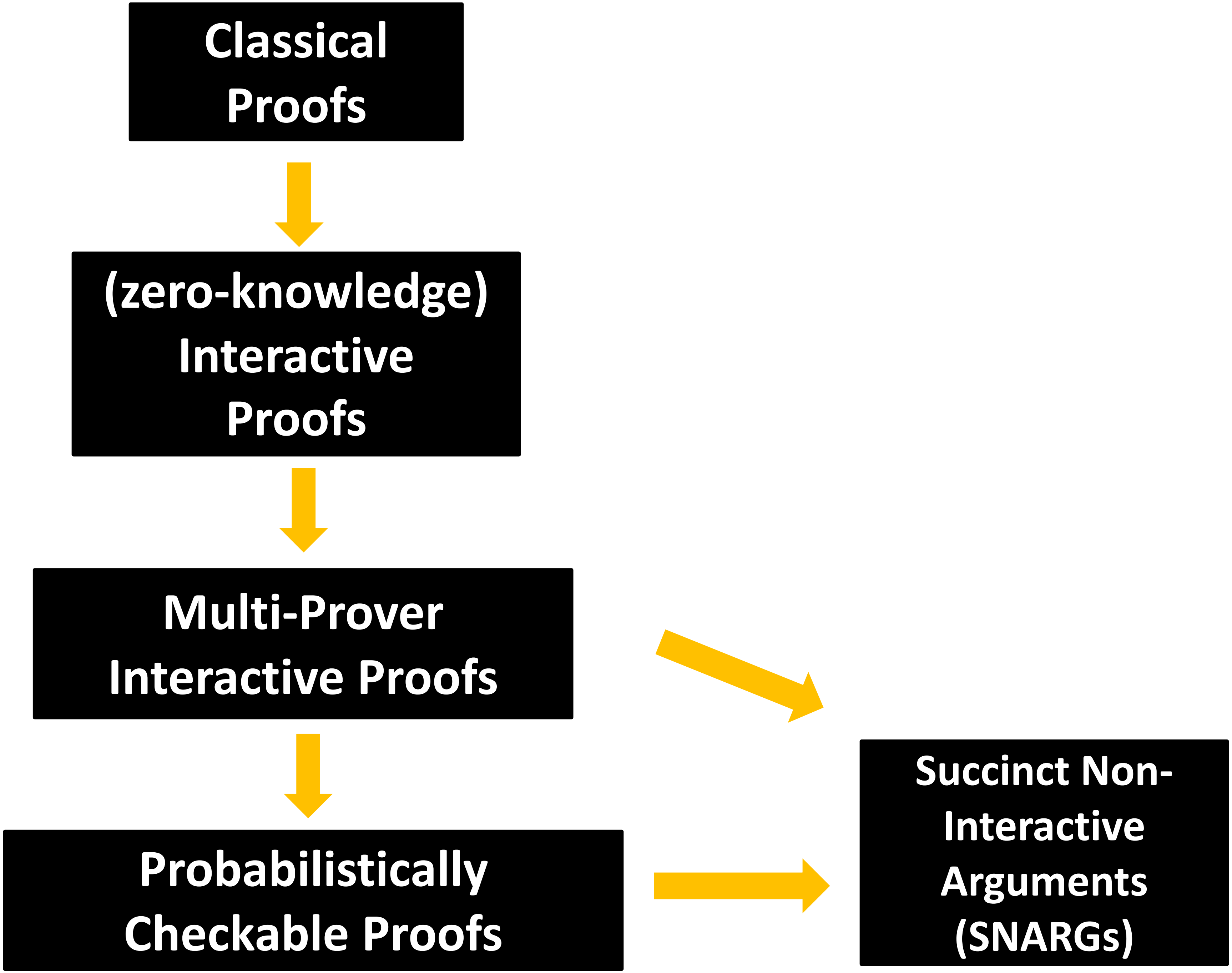
O(1) Labs

Aleo

DARPA DEFENSE ADVANCED RESEARCH PROJECTS AGENCY

Securing Information for Encrypted Verification and Evaluation (SIEVE)

ZKPROOF



Classical Proofs

(zero-knowledge) Interactive Proofs

Multi-Prover Interactive Proofs

Probabilistically Checkable Proofs

Succinct Non-Interactive Arguments (SNARGs)

THANK
YOU