Lecture 11: From Practice to Theory

Guest Lecturer: Alex Lombardi

SIMONS INSTITUTE for the Theory of Computing

Berkeley UNIVERSITY OF CALIFORNIA

Zero Knowledge Proofs

Instructors: Dan Boneh, Shafi Goldwasser, Dawn Song, Justin Thaler, Yupeng Zhang





Stanford University

Berkeley UNIVERSITY OF CALIFORNIA

GEORGETOWN UNIVERSITY

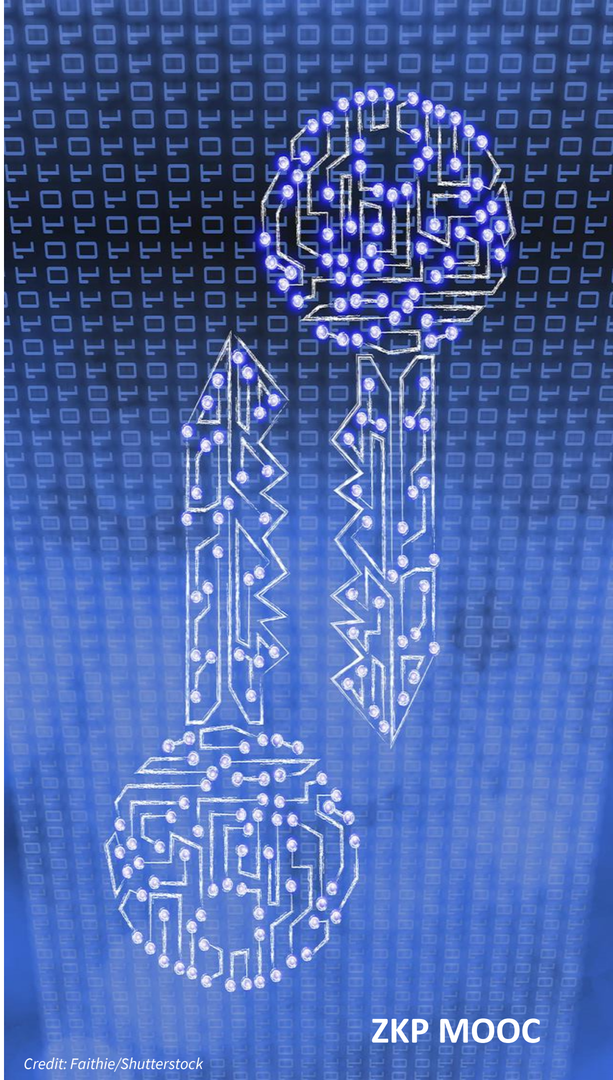
TEXAS A&M UNIVERSITY

Authentication

Blockchains and cryptocurrencies

secure multiparty computation

What does theoretical research on proof systems look like?



Credit: Faithie/Shutterstock

ZKP MOOC

Theoretical Research on Cryptographic Proofs

Feasibility (do they exist in principle?)

- SNAR(G/K)s, other protocols (ZK, WI, WH, etc.)
- Strong attack models (Concurrent? Quantum?)

Theoretical Research on Cryptographic Proofs

Feasibility (do they exist in principle?)

- SNAR(G/K)s, other protocols (ZK, WI, WH, etc.)
- Strong attack models (Concurrent? Quantum?)

Minimize Assumptions (to the extent possible)

- Trusted setup (CRS/URS/plain model)
- Security reduction based on simple, well-studied, falsifiable assumptions.

Theoretical Research on Cryptographic Proofs

Feasibility (do they exist in principle?)

- SNAR(G/K)s, other protocols (ZK, WI, WH, etc.)
- Strong attack models (Concurrent? Quantum?)

Minimize Assumptions (to the extent possible)

- Trusted setup (CRS/URS/plain model)
- Security reduction based on simple, well-studied, falsifiable assumptions.

Improve efficiency

- Amount of communication, number of rounds
- Prover/verifier efficiency

Theoretical Research on Cryptographic Proofs

Feasibility (do they exist in principle?)

- SNAR(G/K)s, other protocols (ZK, WI, WH, etc.)
- Strong attack models (Concurrent? Quantum?)

+Applications

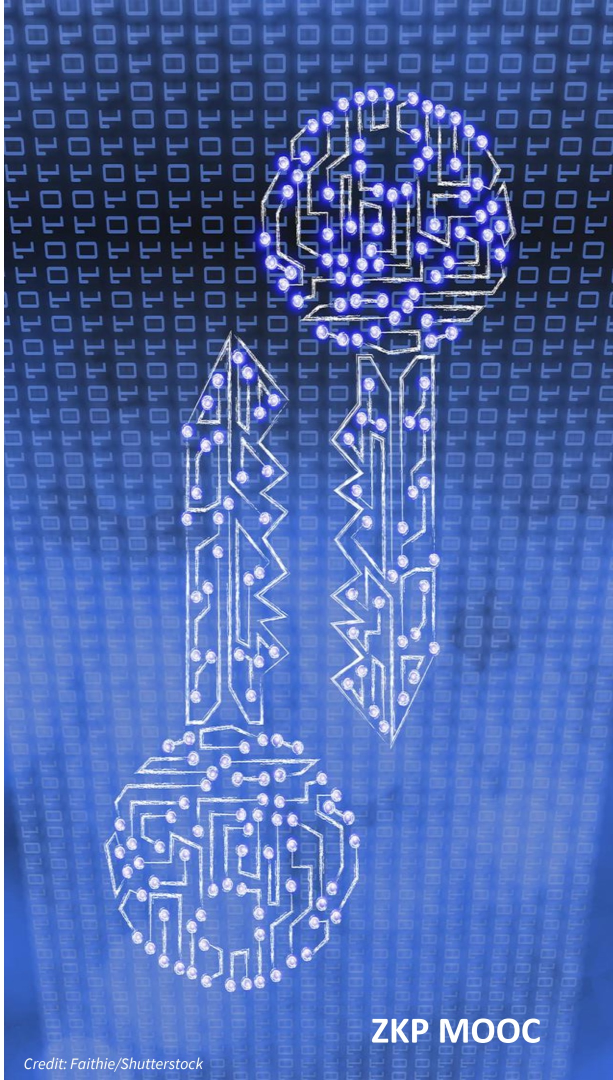
Minimize Assumptions (to the extent possible)

- Trusted setup (CRS/URS/plain model)
- Security reduction based on simple, well-studied, falsifiable assumptions.

Improve efficiency

- Amount of communication, number of rounds
- Prover/verifier efficiency

Example: Interactive ZK



Credit: Faithie/Shutterstock

Interactive Zero-Knowledge Protocols

- No trusted setup allowed.
  - Security against Malicious verifier hard to guarantee.

Interactive Zero-Knowledge Protocols

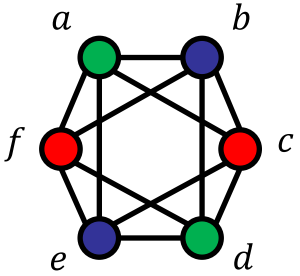
- No trusted setup allowed.
  - Security against Malicious verifier hard to guarantee.
- Lecture 1: ZK for NP [GMW86] with inverse poly soundness error. How do we reduce the error?

Interactive Zero-Knowledge Protocols

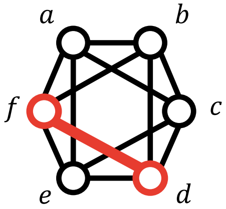
- No trusted setup allowed.
  - Security against Malicious verifier hard to guarantee.
- Lecture 1: ZK for NP [GMW86] with inverse poly soundness error. How do we reduce the error?
  - **Sequential repetition** works (but very inefficient).
  - **Parallel repetition** reduces soundness error but *may not* preserve ZK! Let’s see why:

Zero Knowledge Proofs for NP

Claim: This graph has a 3-coloring.

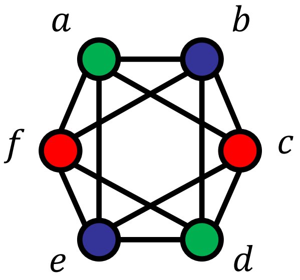


P



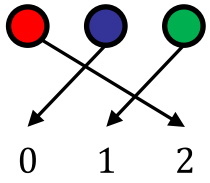
V

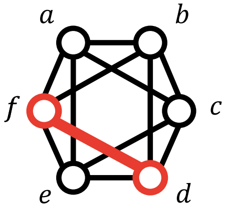
Zero Knowledge Proofs for NP



P

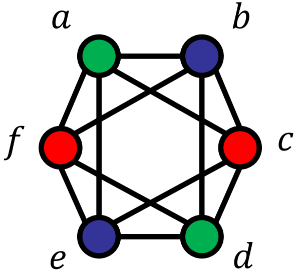
1) Randomize colors





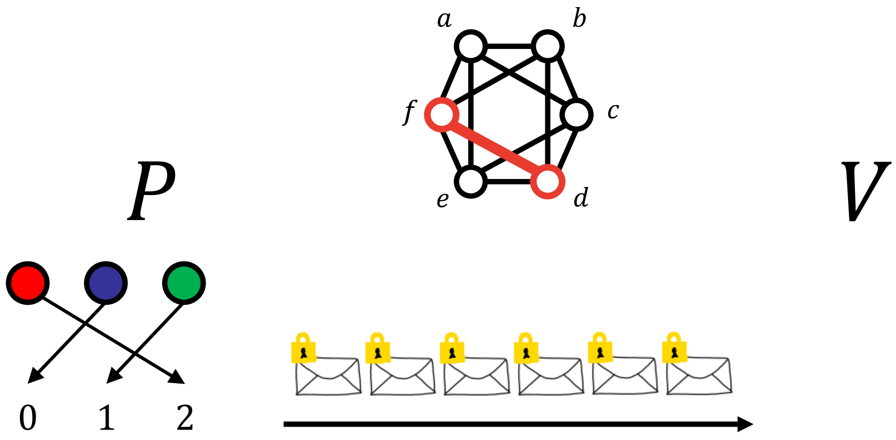
V

Zero Knowledge Proofs for NP

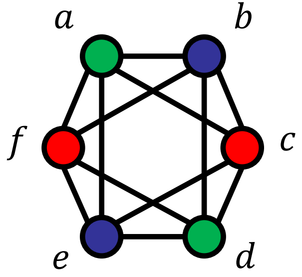


1) Randomize colors

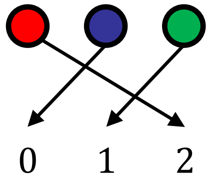
2) Commit



Zero Knowledge Proofs for NP

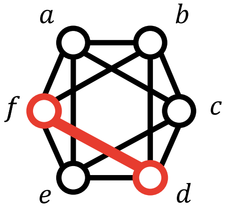


1) Randomize colors



2) Commit

P

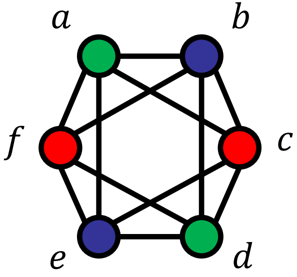


(d, f)

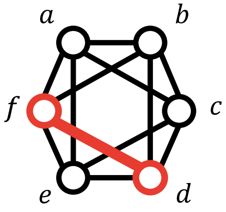
V

1) Sample a **challenge** edge.

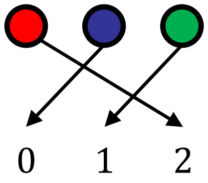
Zero Knowledge Proofs for NP



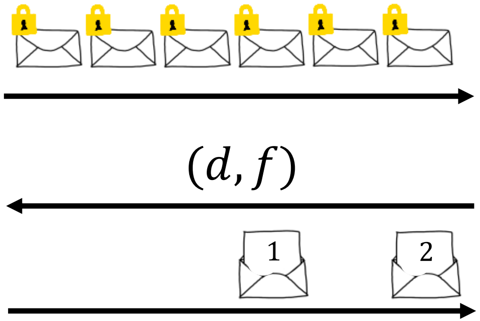
P



V



2) Commit



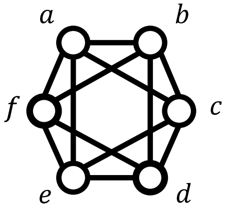
1) Sample a **challenge** edge.

3) Reveal edge colors

2) Accept if colors are different.

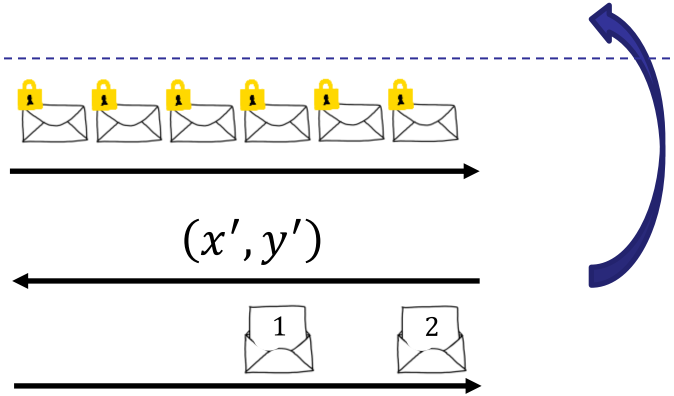
Zero Knowledge Proofs for NP

ZK Simulator: guess Verifier’s challenge in advance, and **rewind** if the guess was wrong.



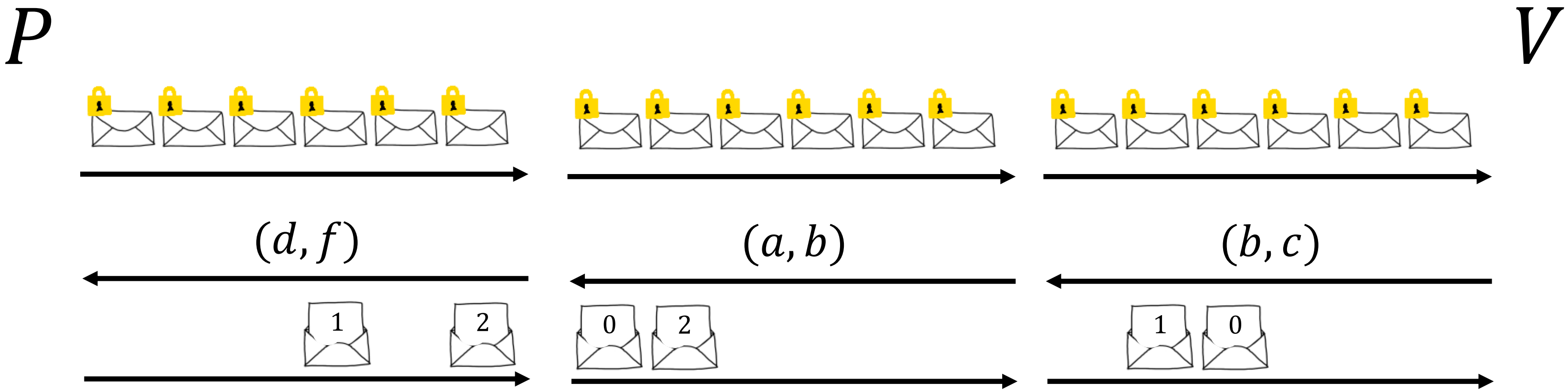
V^*

1) Guess (x, y)
2) Pick two random bits
3) Commit



If $(x, y) \neq (x', y')$

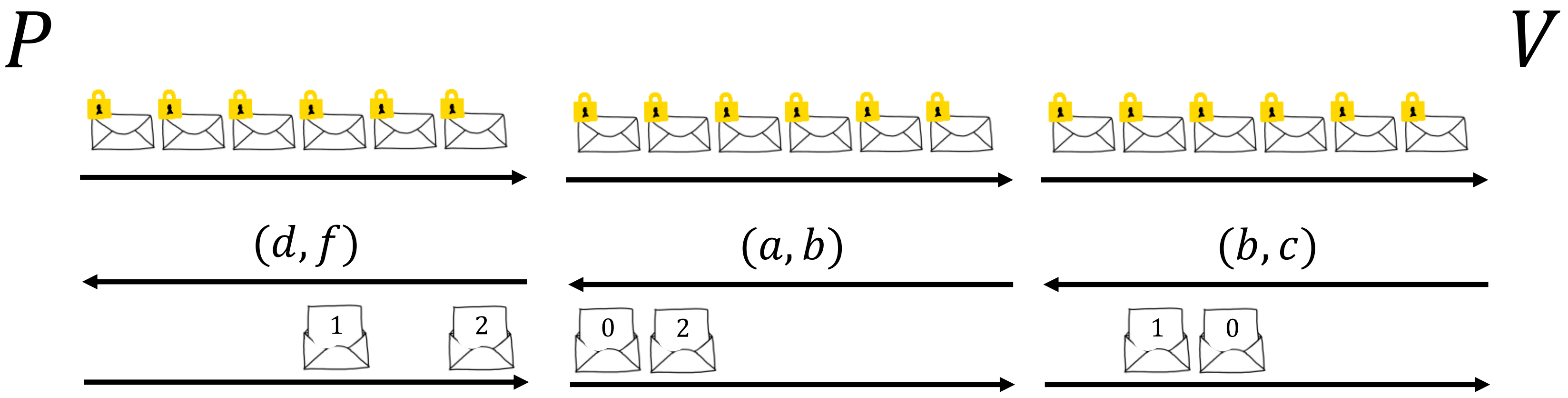
Zero Knowledge Proofs for NP



If there are t repetitions, over 2^t possible challenges to guess from!

Would take exponential time.

Zero Knowledge Proofs for NP



In fact, it turns out that this protocol really shouldn't be ZK!

[DNRS99]: If you can do Fiat-Shamir for Π, then Π wasn't malicious-verifier ZK.

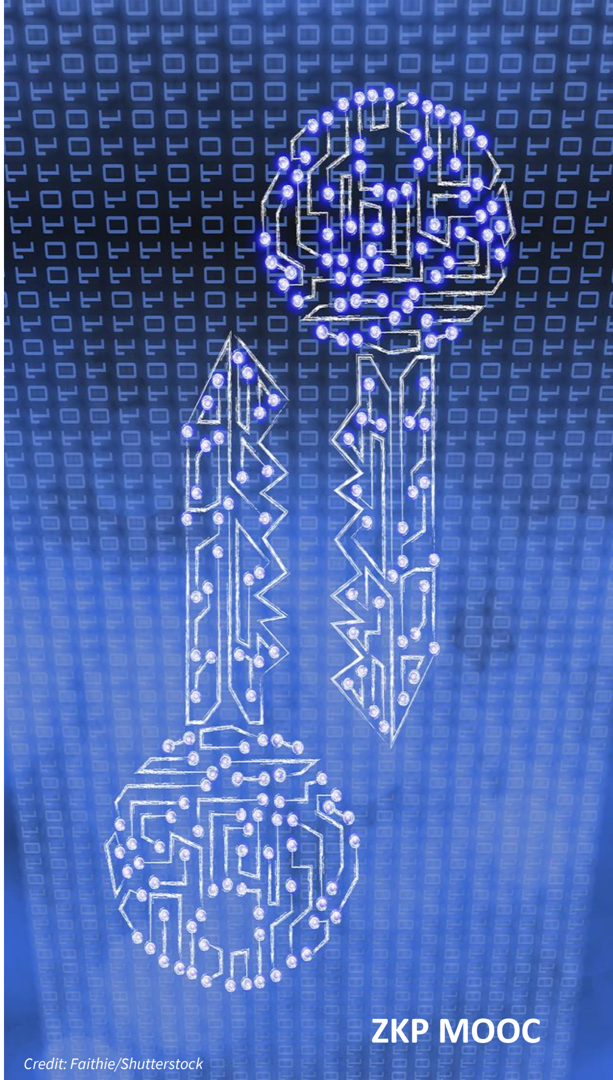
Interactive Zero-Knowledge Protocols

- No trusted setup allowed.
  - Security against Malicious verifier hard to guarantee.
- Many lines of research devoted to understanding the feasibility of interactive ZK.

Interactive Zero-Knowledge Protocols

- No trusted setup allowed.
  - Security against Malicious verifier hard to guarantee.
- Many lines of research devoted to understanding the feasibility of interactive ZK.
    - **How many communication rounds?** [BKP18] suggests that you can do it in 3.
    - **How efficient can you make the prover?** [IKOS07, …]
    - **Stronger forms of security:** quantum attacks, concurrency

Main Topics:
Fiat-Shamir and SNARGs



Credit: Faithie/Shutterstock

Succinct Non-Interactive Arguments (SNARGs)

$$x, \mathrm{crs}$$

$$P(w) \xrightarrow{\pi} V$$

- Completeness: if $x \in L$, V **accepts** honest P with probability $1 - \mathrm{negl}$
- Computational Soundness: if $x \notin L$, for all efficient P^*, V **rejects** w.p. $1 - \mathrm{negl}$
- Succinctness: proof has length $\mathrm{poly}(\lambda, \log(|x| + |w|))$ and verification is **fast**.

Succinct Non-Interactive Arguments (SNARGs)

$$x, \mathrm{crs}$$

$$P(w) \xrightarrow{\quad \pi \quad} V$$

- Completeness: if $x \in L$, V **accepts** honest P with probability $1 - \mathrm{negl}$
- Computational Soundness: if $x \notin L$, for all efficient P^*, V **rejects** w.p. $1 - \mathrm{negl}$
- Succinctness: proof has length $\mathrm{poly}(\lambda, \log(|x| + |w|))$ and verification is **fast**.

This class so far: constructions of SNARGs using IOPs and a random oracle.

The Fiat-Shamir Transform

Powerful, general proposal for removing interaction.

Interactive

$\boldsymbol{P}$ $\xrightarrow{\alpha}$ $\boldsymbol{V}$

$\xleftarrow{\beta}$

$\xrightarrow{\gamma}$

Non-Interactive

Hash Function h

$\boldsymbol{P}$ $\boldsymbol{V}$

Compute $\beta = h(\alpha)$ $\xrightarrow{\alpha, \beta, \gamma}$

If h is modeled as a random oracle, securely compiles any constant-round public coin protocol.

The Fiat-Shamir Transform

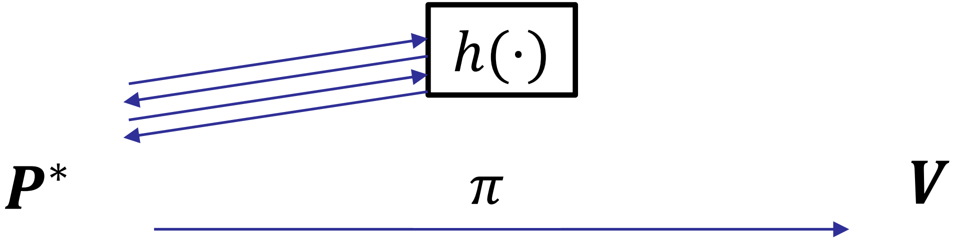
What does that mean?

If h is modeled as a random oracle, securely compiles any constant-round public coin protocol.

The Random Oracle Model [BR93]

Assumption about the structure of an attack on a hash function h:

"The best you can do is treat h as a black box in your attack."

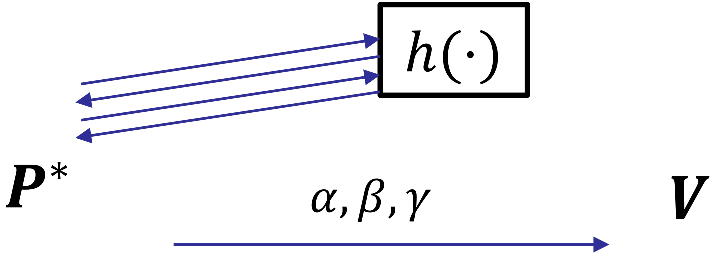


Under such an assumption, $h(\cdot)$ can be thought of as a random function.

Fiat-Shamir in the ROM

Claim: Fiat-Shamir for constant-round protocols is secure in the ROM

Proof (3 message case):

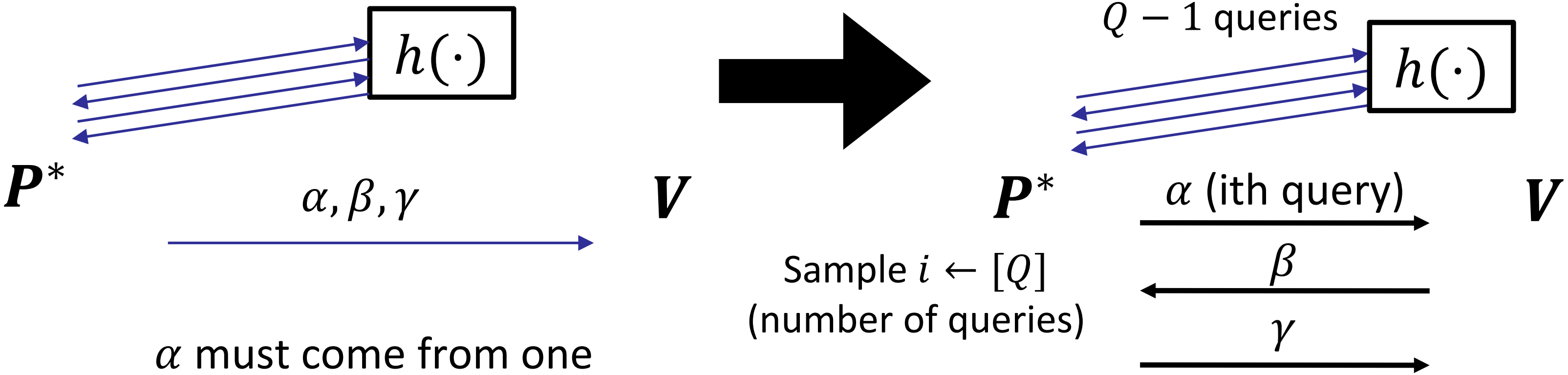


α must come from one of the oracle queries

Fiat-Shamir in the ROM

Claim: Fiat-Shamir for constant-round protocols is secure in the ROM

Proof (3 message case):

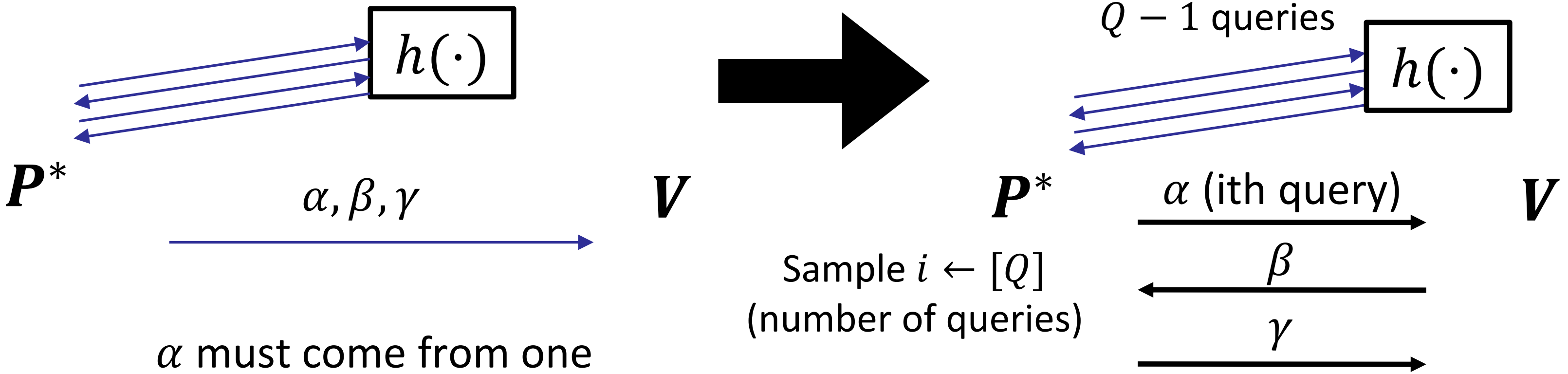


α must come from one of the oracle queries

Fiat-Shamir in the ROM

Claim: Fiat-Shamir for constant-round protocols is secure in the ROM

Proof (3 message case):



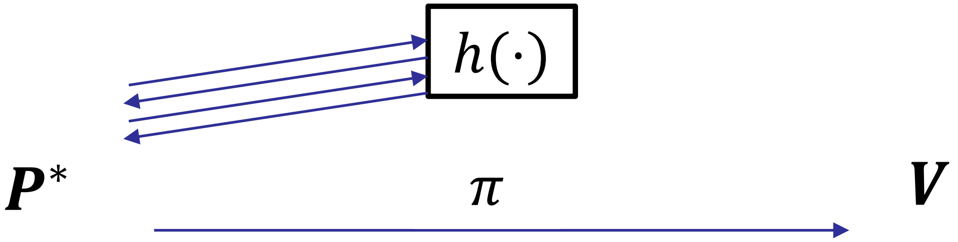
α must come from one of the oracle queries

$1/Q$ security loss

The Random Oracle Model [BR93]

Assumption about the structure of an attack on a hash function h:

"The best you can do is treat h as a black box in your attack."

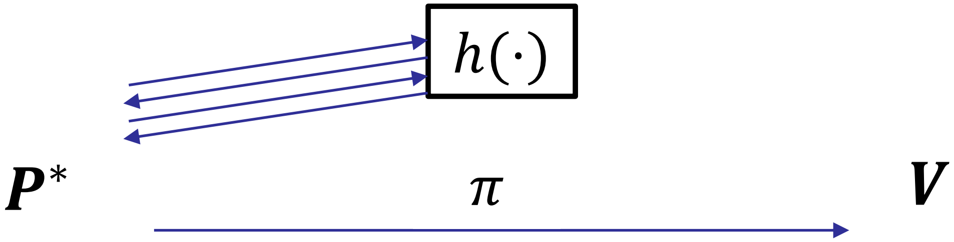


Under such an assumption, $h(\cdot)$ can be thought of as a random function.

The Random Oracle Model [BR93]

Assumption about the structure of an attack on a hash function h:

"The best you can do is treat h as a black box in your attack."

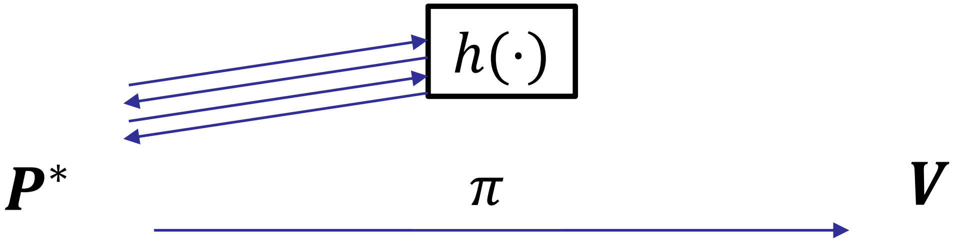


In practice, $h(\cdot)$ is instantiated with (e.g.) SHA256, possibly salted.

The Random Oracle Model [BR93]

Assumption about the structure of an attack on a hash function h:

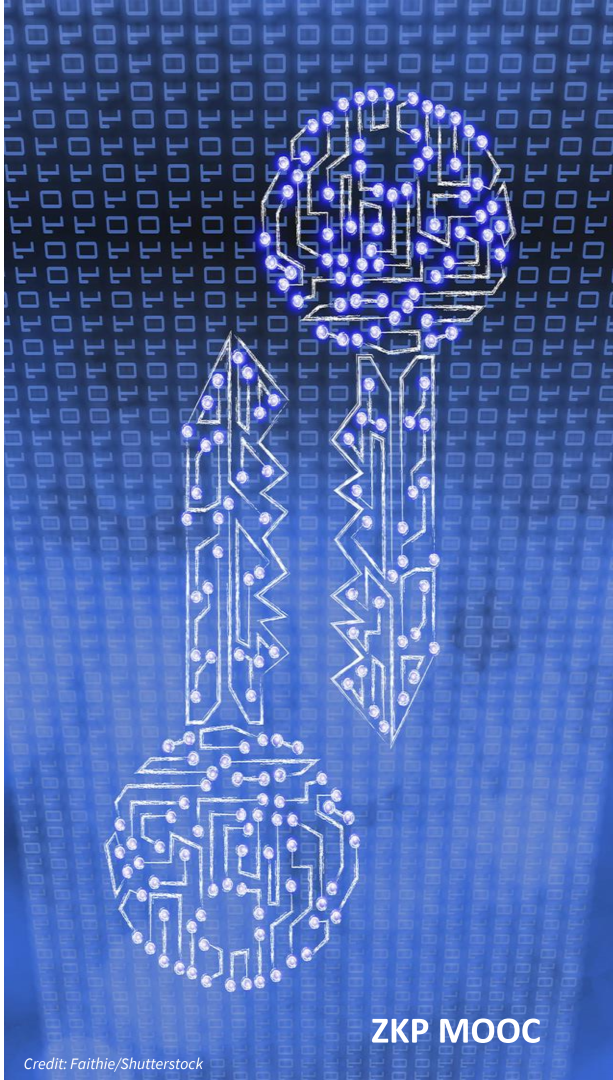
"The best you can do is treat h as a black box in your attack."



No matter what, $h(\cdot)$ is instantiated with a public efficient algorithm.

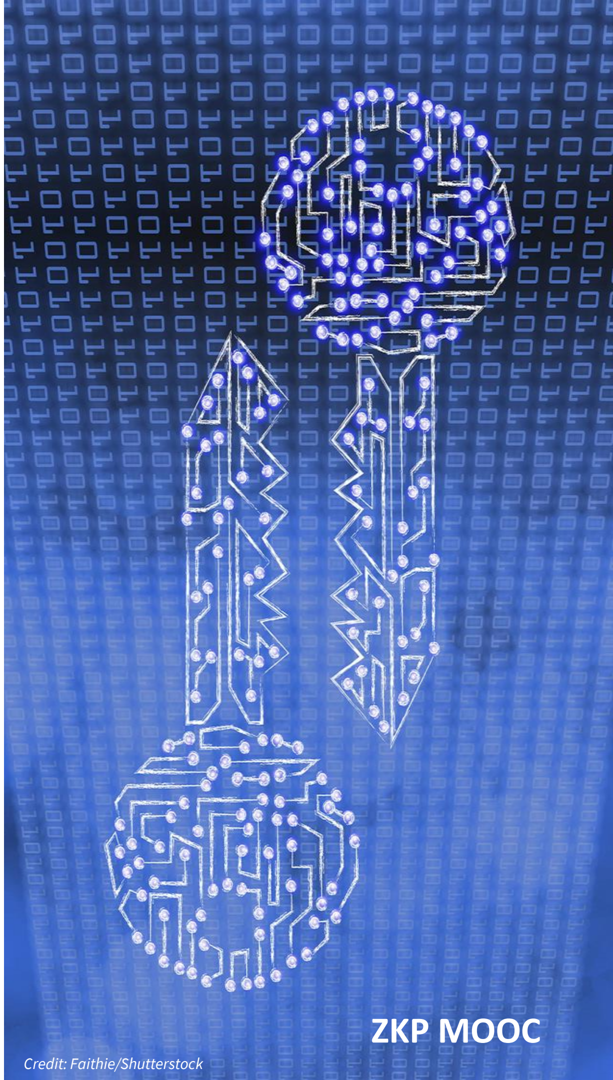
Obvious (theoretical) problem:

Public efficient algorithms can’t compute random functions



Credit: Faithie/Shutterstock

Next: example of an uninstantiable random oracle property [CGH98]



Credit: Faithie/Shutterstock

Random Oracles Do Not Exist

Fix a function $f\colon \{0,1\}^* \to \{0,1\}^\lambda$

We say that a hash function h is Correlation Intractable (CI) for f if it is hard to find x such that $h(x) = f(x)$

$\forall$ PPT A,

$$\Pr_{\substack{h \leftarrow H \\ x \leftarrow A(h)}} [h(x) = f(x)] = \mathrm{negl}$$

Random Oracles Do Not Exist

For any fixed f, a RO is CI for f.

Why? Each query x to the RO produces a random output y, which is equal to $f(x)$ with probability $2^{-\lambda}$.

Random Oracles Do Not Exist

Claim [CGH98]: $\exists f$ such that for any (efficient) hash family H, H fails to be CI for f!

Random Oracles Do Not Exist

Claim [CGH98]: $\exists f$ such that for any (efficient) hash family H, H fails to be CI for f!

$f(x)$: interpret x as a program P and output $P(x)$.

Random Oracles Do Not Exist

Claim [CGH98]: $\exists f$ such that for any (efficient) hash family H, H fails to be CI for f!

$f(x)$: interpret x as a program P and output $P(x)$.

Given $h \leftarrow H$, attack sets $x = \langle h \rangle$ to be a description of h. Then,

$$f(x) = P(x) = P(\langle h \rangle) = h(\langle h \rangle) = h(x).$$

Random Oracles Do Not Exist

Is this a reasonable counterexample?

- Hash function/random oracle must be able to hash inputs of arbitrary length. CI with bounded inputs might exist!

Random Oracles Do Not Exist

Is this a reasonable counterexample?

- Hash function/random oracle must be able to hash inputs of arbitrary length. CI with bounded inputs might exist!
  - [Barak01,GK03] apply to fixed-input length hash functions.

Random Oracles Do Not Exist

Is this a reasonable counterexample?

- Hash function/random oracle must be able to hash inputs of arbitrary length. CI with bounded inputs might exist!
  - [Barak01,GK03] apply to fixed-input length hash functions.

Theorem [Barak '01, Goldwasser-Kalai '03]: $\exists$ interactive protocol Π such that Π_{FS} is ROM-secure but insecure for any efficiently computable H (e.g. SHA-3).

Random Oracles Do Not Exist

Is this a reasonable counterexample?

- Hash function/random oracle must be able to hash inputs of arbitrary length. CI with bounded inputs might exist!
  - [Barak01,GK03] apply to fixed-input length hash functions.
- Security property broken by running the hash function on its own description. Is this practically relevant?

Random Oracles Do Not Exist

Is this a reasonable counterexample?

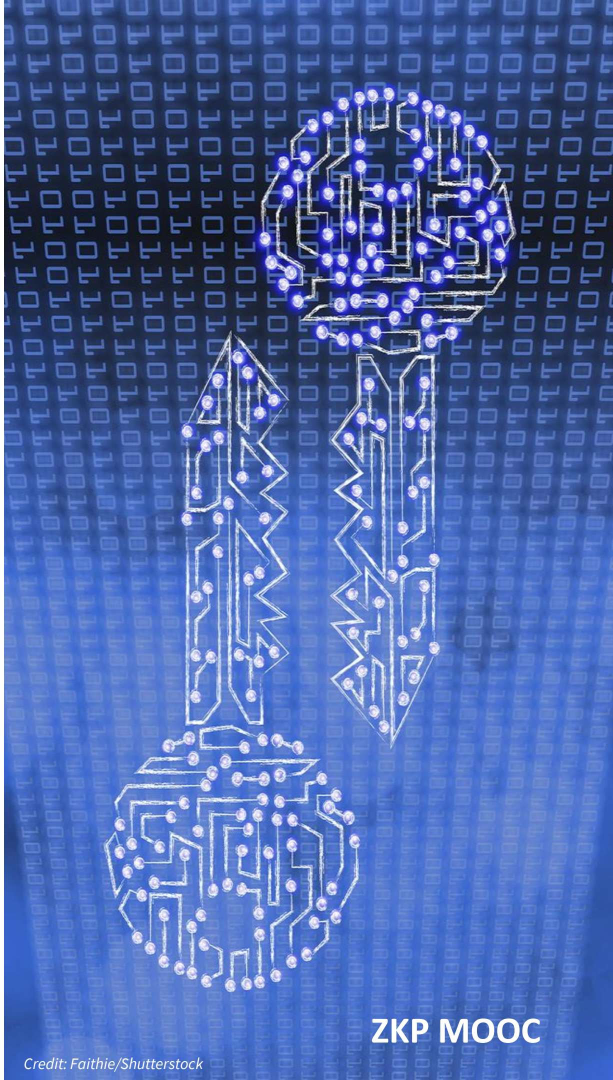
- Hash function/random oracle must be able to hash inputs of arbitrary length. CI with bounded inputs might exist!
  - [Barak01,GK03] apply to fixed-input length hash functions.
- Security property broken by running the hash function on its own description. Is this practically relevant?
  - Recursive SNARKs do something of this flavor.

Random Oracles Do Not Exist

Is this a reasonable counterexample?

- Hash function/random oracle must be able to hash inputs of arbitrary length. CI with bounded inputs might exist!
  - [Barak01,GK03] apply to fixed-input length hash functions.
- Does NOT imply RO-based SNARKs are broken in practice.
  - But it does imply a lack of theoretical understanding.

What can we do without random oracles?



Credit: Faithie/Shutterstock

Falsifiable Assumptions

Prove security assuming that some concrete algorithmic task is infeasible:

- Computing discrete logarithms is hard.
- Solving random noisy linear equations (LWE) is hard.
- SHA256 is collision-resistant.

Falsifiable Assumptions

Many cryptographic constructions use random oracles to get better efficiency, but *can* be based on falsifiable assumptions.

- CCA-secure public key encryption.
- Identity-based encryption.
- Non-interactive zero knowledge.

Falsifiable Assumptions

Can (ZK-)SNARKs for NP be built based on falsifiable assumptions?

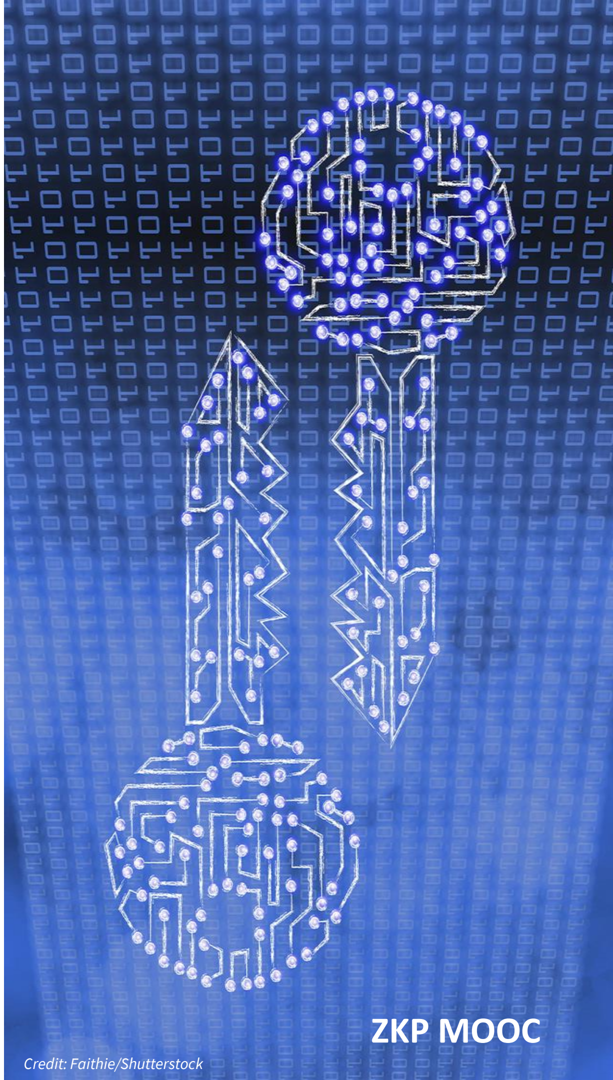
- (minor caveats but) No!
- No way to extract a long witness from a short proof. Need assumption (RO, “knowledge assumption”) that guarantees adversary “knows” a long string given a short commitment.

Falsifiable Assumptions

Can (ZK-)SNARGs for NP be built based on falsifiable assumptions?

- It's complicated. (We don't know)
- Significant barriers [Gentry-Wichs '11]
- The community is still trying to understand this.

Rest of today: SNARGs for limited computations from falsifiable assumptions (LWE)



Two tools/techniques

- Correlation-intractable hash functions [CCHLRRW19,PS19,HLR21]
  - Used to instantiate Fiat-Shamir without random oracles, for “nice enough” interactive protocols.
- Somewhere extractable commitments [HW15]
  - Used to make a “nice enough” interactive protocol
    - Special variant of the typical IOP-based approach.

Correlation Intractability

A hash family H is CI for f if $\forall$ PPT A,

$$\Pr_{\substack{h \leftarrow H \\ x \leftarrow A(h)}} [h(x) = f(x)] = \text{negl}$$

Correlation Intractability

A hash family H is CI for binary relation R if $\forall$ PPT A,

$$\Pr_{\substack{h \leftarrow H \\ x \leftarrow A(h)}} \left[\left(x, h(x) \right) \in R \right] = \mathrm{negl}$$

Correlation Intractability

A hash family H is CI for f if $\forall$ PPT A,

$$\Pr_{\substack{h \leftarrow H \\ x \leftarrow A(h)}} [h(x) = f(x)] = \text{negl}$$

- Weren't these impossible to build?
  - Restrict to fixed input length (necessary)
  - Restrict to fixed running time on f (unclear if necessary)

CI Construction

Here's a simple construction [CLW18] using Fully Homomorphic Encryption (FHE)

$$x \xrightarrow[\mathrm{Enc}]{\mathrm{pk}} \boxed{x} \xrightarrow[\mathrm{Eval}]{\mathrm{pk},\, f} \boxed{f(x)} \xrightarrow[\mathrm{Dec}]{\mathrm{sk}} f(x)$$

CI Construction

$$\langle h \rangle = (\mathrm{pk}, \mathrm{Enc}(g)\ )$$

Real hash key: $g \equiv 0$ (or a uniform random string – nobody can tell)

$$h(x) = \mathrm{Eval}(x, \mathrm{Enc}(g)\ ) = \mathrm{Enc}(g(x))$$

Key point: g is hidden to everyone! We consider different g to prove security.

Security Analysis

Suppose an attacker, given $\langle h \rangle$, finds x such that $h(x) = f(x)$.

Key idea: let $g^*(x) = \mathrm{Dec}(f(x)) + 1$. We know that $\mathrm{Enc}(g) \approx \mathrm{Enc}(g^*)$ if the encryption scheme is (circular-)secure.

$$h(x) = \mathrm{Eval}(x, \mathrm{Enc}(g^*)) = \mathrm{Enc}(g^*(x))$$

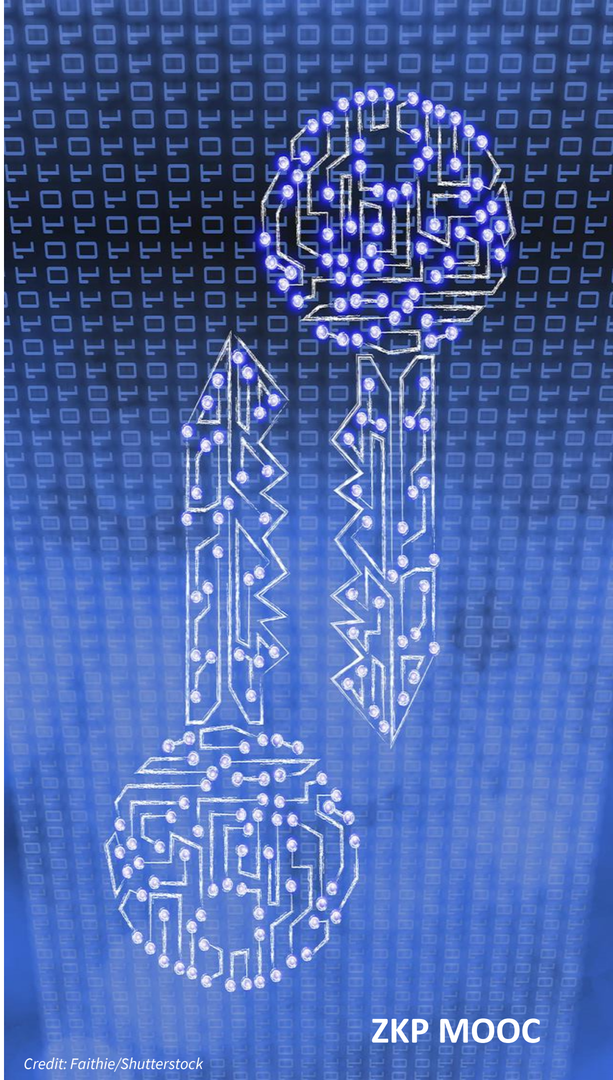
$\mathrm{Dec}(f(x)) = \mathrm{Dec}(h(x)) = g^*(x) = \mathrm{Dec}(f(x)) + 1$. Impossible!

Correlation Intractability: what we know

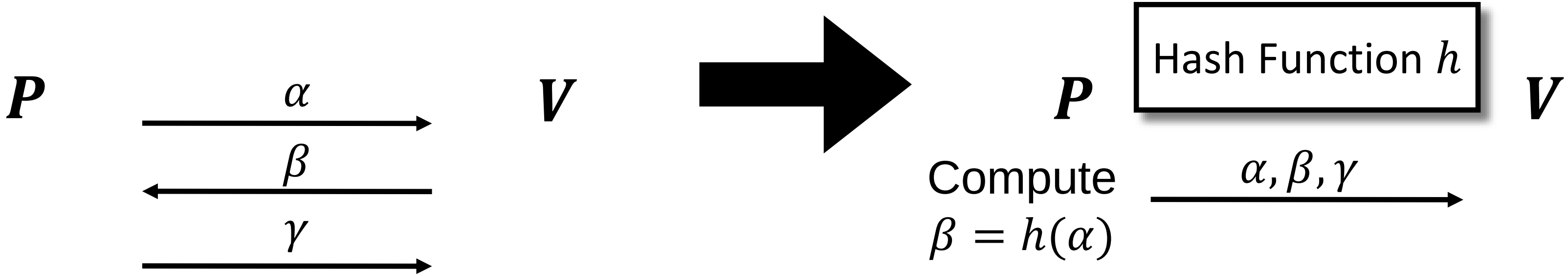
H is CI for R if $\forall$ PPT A, $\Pr_{\substack{h \leftarrow H \\ x \leftarrow A(h)}}\left[\left(x, h(x)\right) \in R\right] = \mathrm{negl}$

- Constructions for efficiently computable functions:
    - From LWE ([CLW18,PS19,LV22])
    - From DDH (JJ21)
- Construction [HLR21] for (efficient) relations with “product structure”

How do we use CI to instantiate Fiat-Shamir?



Avoid the “Bad Challenges”



Def: Given false claim x and a first message α, a challenge β is “bad” if there exists a prover message γ making V accept.

We want to say: if the (3 message) interactive protocol is sound, then (for all x, α) most β are not bad. True for *statistically sound* IPs.

Avoid the "Bad Challenges"

$$P \xrightarrow{\alpha} V$$
$$P \xleftarrow{\beta} V$$
$$P \xrightarrow{\gamma} V$$

$$\Longrightarrow$$

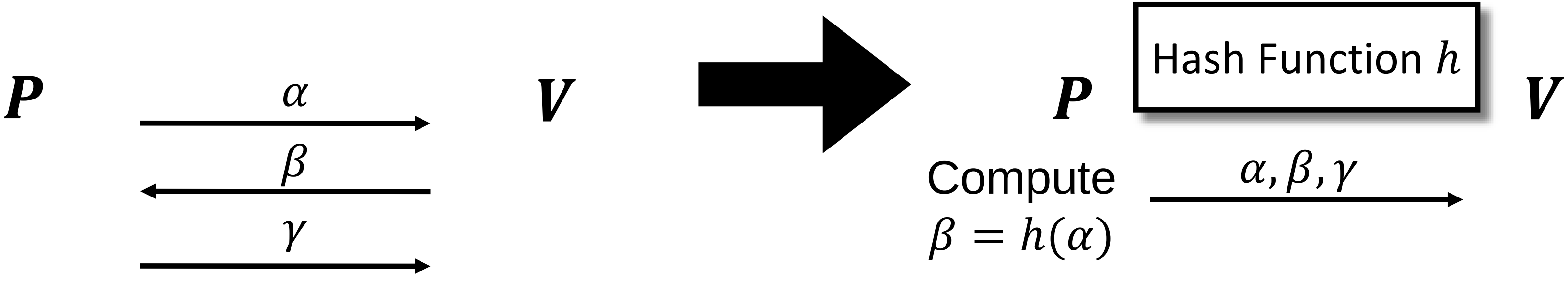
Hash Function h

P Compute $\beta = h(\alpha)$ $\xrightarrow{\alpha, \beta, \gamma}$ V

Exactly what CI is good for! Define relation $R_x = \{(\alpha, \beta) : \beta \text{ is bad}\}$. Then if h is CI for R_x (when $x \notin L$), Π_{FS} is sound using h!

Protocols with more than 3 messages: round-by-round soundness (each round has a type of "bad challenge" to avoid).

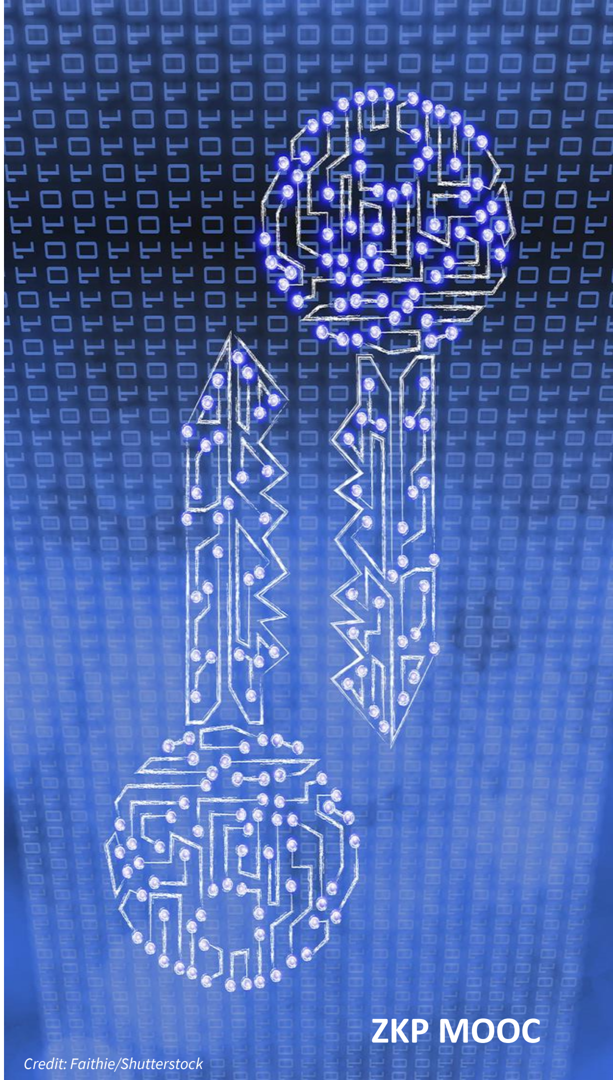
Avoid the “Bad Challenges”



Main challenges:

1) Sometimes our IP doesn’t have statistical soundness.
2) We can only build CI for relations R that can be decided efficiently.

Important example: SNARGs via IOPs (PCPs)



Credit: Faithie/Shutterstock

SNARGs from PCPs [Kilian, Micali]

$P(x, w)$ $\qquad\qquad$ $V(x)$

Compute (long) proof string π from (x, w)

$\xrightarrow{\mathrm{Com}(\pi)}$

r (describes location set S) $\longleftarrow$ $\quad r \leftarrow \{0,1\}^\lambda$

$\xrightarrow{\text{Open } \pi_S}$ $\quad$ Verify opening, check consistency of π_S

Candidate SNARG: apply Fiat-Shamir to this protocol!

Simplified (less efficient) version of modern SNARKs you've learned about.

SNARGs from PCPs [Kilian, Micali]

$P(x, w)$ | $V(x)$

Compute (long) proof string π from (x, w)

$\xrightarrow{\mathrm{Com}(\pi)}$

$\xleftarrow{r \text{ (describes location set } S)}$ $r \leftarrow \{0,1\}^\lambda$

$\xrightarrow{\text{Open } \pi_S}$

Verify opening, check consistency of π_S

Not statistically sound, so it's not clear how to analyze FS without random oracles.

SNARGs for Batch NP

$P(x_1, \ldots, x_k, w_1, \ldots, w_k)$ $\xrightarrow{\pi}$ $V(x_1, \ldots, x_k)$

- Completeness: if $x_i \in L$ for all i, V **accepts** honest P
- Computational Soundness: if $x_i \notin L$ for some i, for all efficient P^*, V **rejects.**
- Succinctness: proof has length $\text{poly}(\lambda, |w|, \log k)$

Surprisingly powerful (implies SNARGs for P, etc.)

Interactive Batch Arguments from PCPs [CJJ21]

$P(x_1, \dots, x_k, w_1, \dots, w_k)$ $V(x_1, \dots, x_k)$

$$\xrightarrow{\mathrm{Com}(\pi_1, \dots, \pi_k)}$$

$$\xleftarrow{r \text{ (describes location set } S)}$$

$r \leftarrow \{0,1\}^\lambda$

$$\xrightarrow{\text{Open } \pi_{1,S}, \dots \pi_{k,S}}$$

Verify opening, check consistency of π_S

Interactive Batch Arguments from PCPs [CJJ21]

$P(x_1, \dots, x_k, w_1, \dots, w_k)$ $\qquad$ $V(x_1, \dots, x_k)$

$\xrightarrow{\mathrm{Com}(\pi_1, \dots, \pi_k)}$

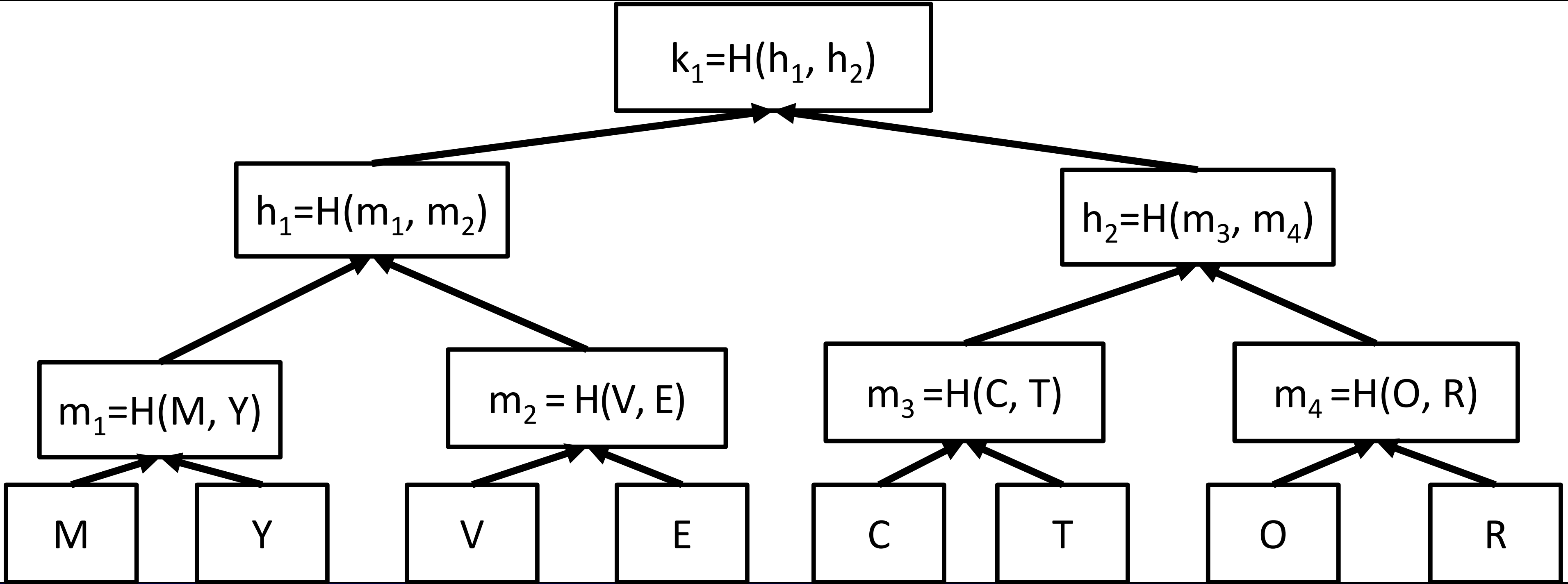
$\xleftarrow{r \text{ (describes location set } S\text{)}}$ $\qquad$ $r \leftarrow \{0,1\}^\lambda$

$\xrightarrow{\text{Open } \pi_{1,S}, \dots \pi_{k,S}}$ $\qquad$ Verify opening, check consistency of π_S

Choose Com to be *statistically binding* on one out of k proofs (π_1)

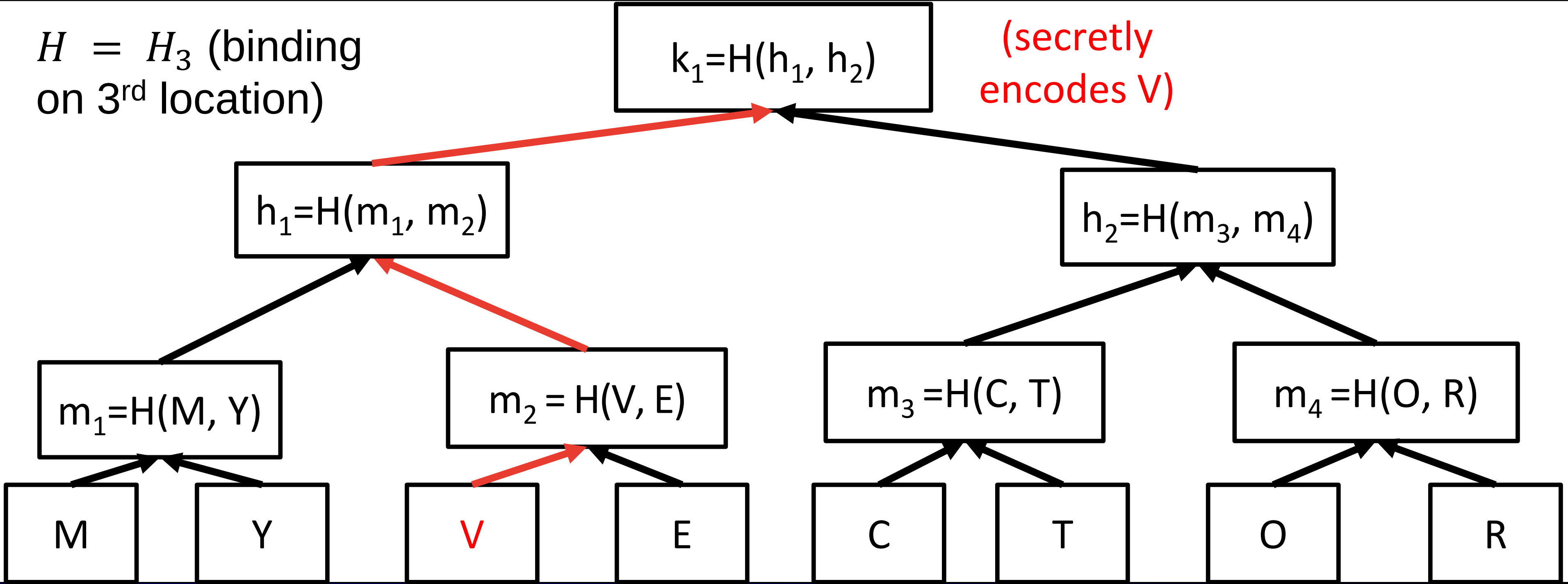
If x_i is false, protocol is now statistically sound! (π_1 is fixed)

SSB Commitments



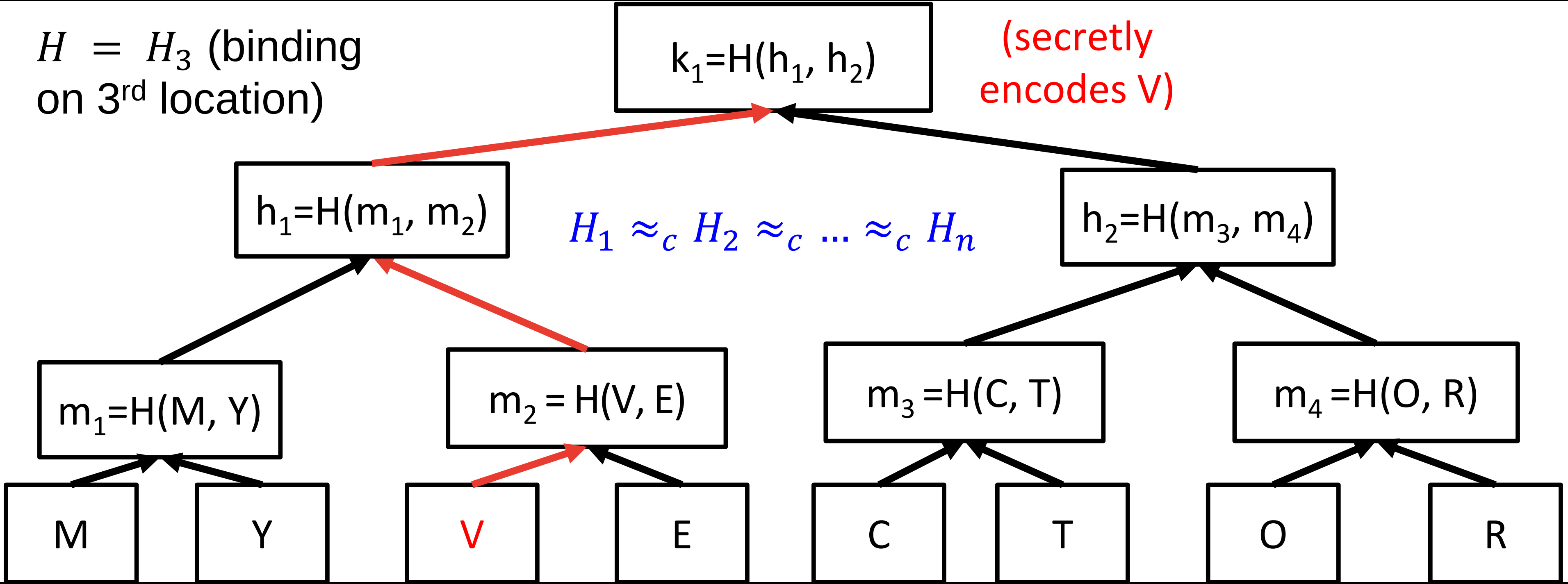
SSB Commitments

$H = H_3$ (binding on 3rd location)



SSB Commitments

$H = H_3$ (binding on 3rd location)



$H_1 \approx_c H_2 \approx_c \ldots \approx_c H_n$

Interactive Batch Arguments from PCPs [CJJ21]

$P(x_1, \dots, x_k, w_1, \dots, w_k)$ $\qquad$ $V(x_1, \dots, x_k)$

$$P \xrightarrow{\mathrm{Com}(\pi_1, \dots, \pi_k)} V$$

$$P \xleftarrow{r \text{ (describes location set } S\text{)}} V \qquad r \leftarrow \{0,1\}^\lambda$$

$$P \xrightarrow{\text{Open } \pi_{1,S}, \dots \pi_{k,S}} V$$

Verify opening, check consistency of π_S

Choose Com to be *statistically binding* on one out of k proofs (π_1)

If x_i is false, protocol is now statistically sound! (π_1 is fixed)

Interactive Batch Arguments from PCPs [CJJ21]

$P(x_1, \ldots, x_k, w_1, \ldots, w_k)$ $V(x_1, \ldots, x_k)$

$\xrightarrow{\mathrm{Com}(\pi_1, \ldots, \pi_k)}$

$\xleftarrow{r \text{ (describes location set } S)}$ $r \leftarrow \{0,1\}^\lambda$

$\xrightarrow{\text{Open } \pi_{1,S}, \ldots \pi_{k,S}}$ Verify opening, check consistency of π_S

Choose Com to be *statistically binding* on one out of k proofs (π_k)

If x_i is false, protocol is now statistically sound! (π_k is fixed)

Batch Arguments from PCPs [CJJ21]

$P(x_1, \dots, x_k, w_1, \dots, w_k)$ $\qquad\qquad$ $V(x_1, \dots, x_k)$

$$P \xrightarrow{\mathrm{Com}(\pi_1, \dots, \pi_k)} V$$

$$P \xleftarrow{r \text{ (describes location set } S)} V \qquad r \leftarrow \{0,1\}^\lambda$$

$$P \xrightarrow{\text{Open } \pi_{1,S}, \dots \pi_{k,S}} V$$

Verify opening, check consistency of π_S

With some work, can use CI hash functions to compile this protocol.

Succinctness: $|w| \cdot \lambda + k \cdot \lambda$, but can be reduced to $|w| \cdot \lambda$ by recursing.

Summary of Fiat-Shamir without RO

- Use hash functions that are CI for appropriate functions/relations
  - [CCHLRRW19,PS19,BKM20,JJ21,HLR21]
- Carefully show that FS-soundness for protocols of interest follows from compatible forms of CI
    - [CCHLRRW19]: (non-succinct) NIZK
    - [JKKZ21]: non-interactive sumcheck protocol
    - [CJJ21]: batch NP arguments

Summary of Fiat-Shamir without RO

Open problems:

- Characterize which protocols can be FS-compiled (we know it doesn't work in general [Bar01, GK03])
- SNARGs for NP from falsifiable assumptions?

END OF LECTURE