Does Fiat-Shamir Require a Cryptographic Hash Function?

Yilei Chen	(Visa Research)
Alex Lombardi	(MIT)
Fermi Ma	(Princeton and NTT Research)
Willy Quach	(Northeastern)

(Public-Coin) Interactive Protocols

[GMR85, B85]

P

V

(Public-Coin) Interactive Protocols

[GMR85, B85]

x is true

P

V

(Public-Coin) Interactive Protocols

[GMR85, B85]

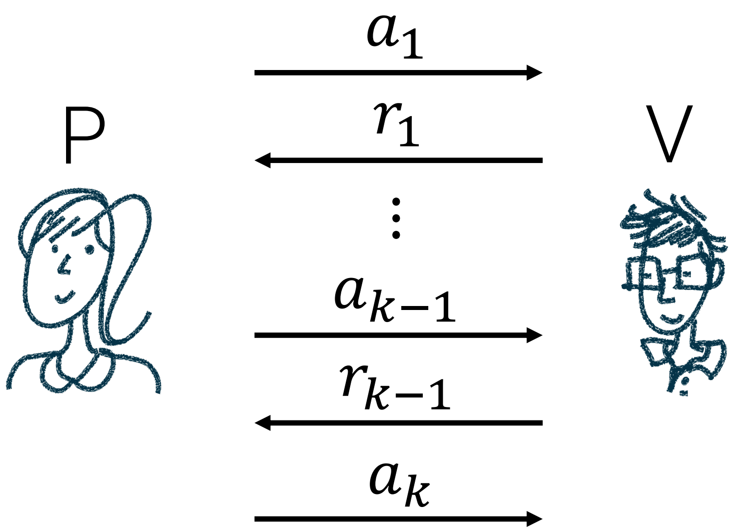
I know a
witness for x

P

V

(Public-Coin) Interactive Protocols

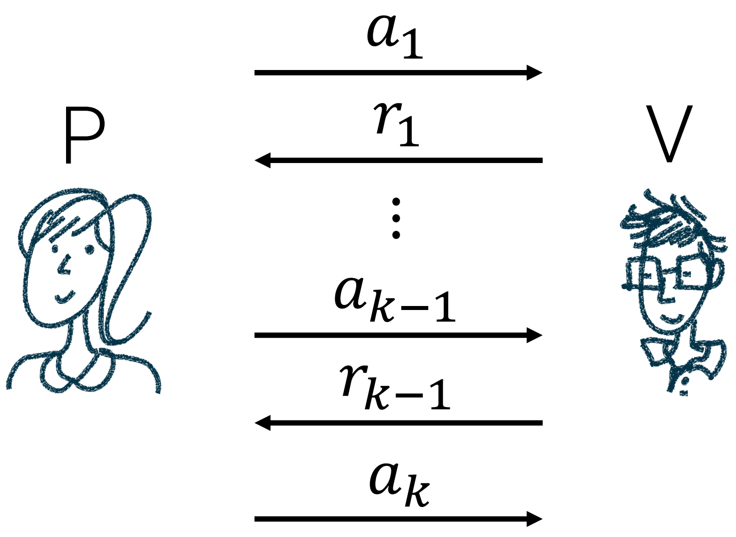
[GMR85, B85]



Public coin: each r_i uniformly random

(Public-Coin) Interactive Protocols

[GMR85, B85]



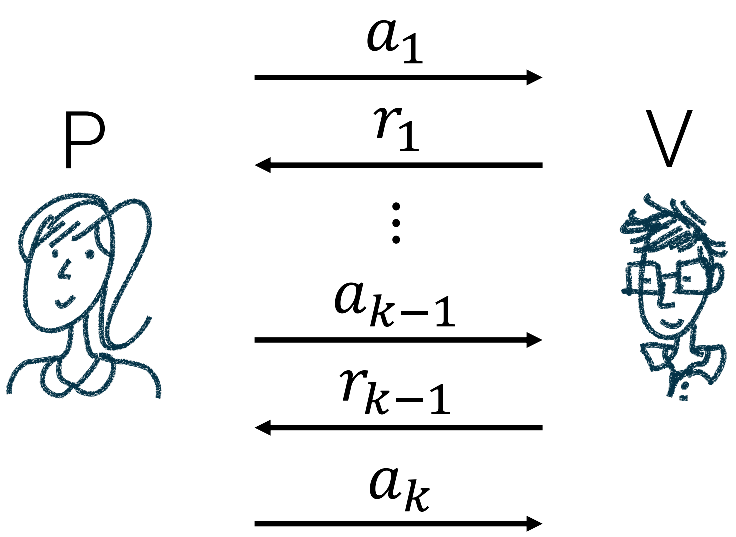
Completeness: If statement is true, verifier accepts w/ probability 1.

Soundness: If statement is false, verifier rejects w/ high probability, no matter what prover does.

Public coin: each r_i uniformly random

(Public-Coin) Interactive Protocols

[GMR85, B85]



Completeness: If statement is true, verifier accepts w/ probability 1.

Soundness: If statement is false, verifier rejects w/ high probability, no matter what prover does.

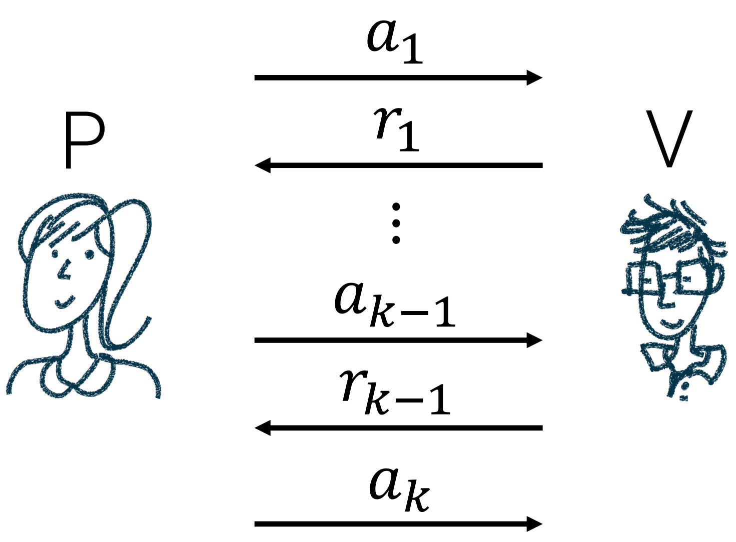
Public coin: each r_i uniformly random

Interaction is powerful [GS86, GMR89, GMW91, S92, K92, …]

IP = PSPACE, zero-knowledge, succinct arguments, etc.

(Public-Coin) Interactive Protocols

[GMR85, B85]



Completeness: If statement is true, verifier accepts w/ probability 1.

Soundness: If statement is false, verifier rejects w/ high probability, no matter what prover does.

Public coin: each r_i uniformly random

Interaction is powerful [GS86, GMR89, GMW91, S92, K92, …]

But do we always need it?

Fiat-Shamir Heuristic

[FS86]

Magical compiler that removes interaction from (public-coin) interactive protocols

Fiat-Shamir Heuristic

[FS86]

Magical compiler that removes interaction from (public-coin) interactive protocols

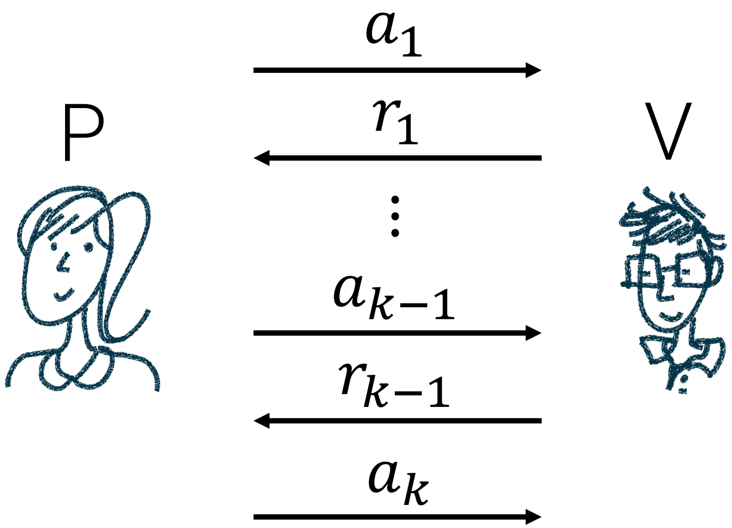
How? Replace random verifier messages with **hash** of previous messages

Fiat-Shamir Heuristic

[FS86]

Magical compiler that removes interaction from (public-coin) interactive protocols

How? Replace random verifier messages with **hash** of previous messages



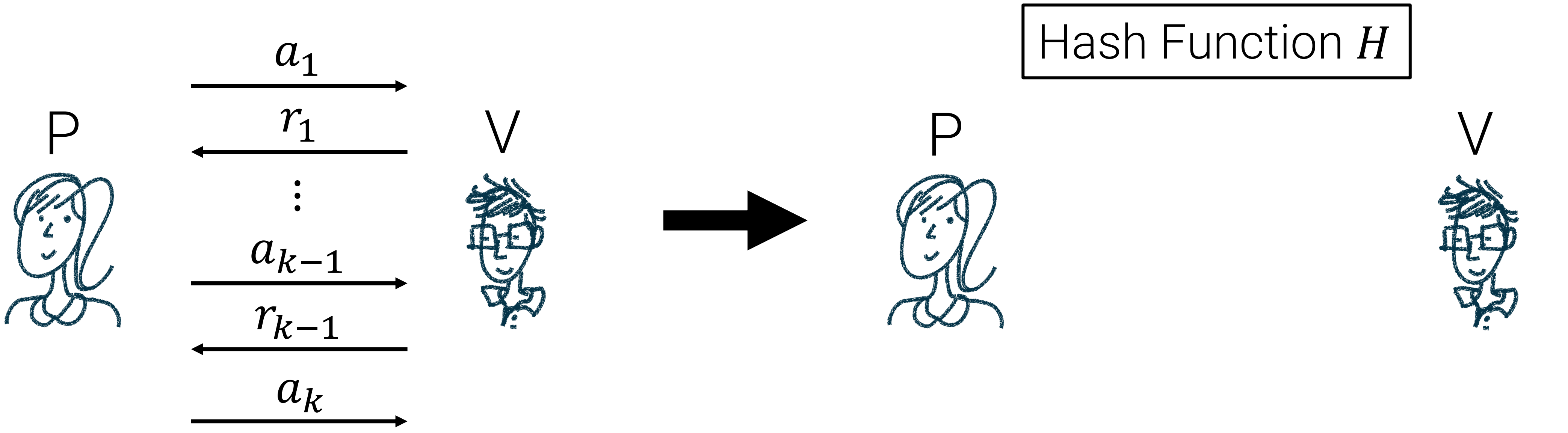
Public-Coin Interactive Protocol Π

Fiat-Shamir Heuristic

[FS86]

Magical compiler that removes interaction from (public-coin) interactive protocols

How? Replace random verifier messages with **hash** of previous messages



Public-Coin Interactive Protocol Π

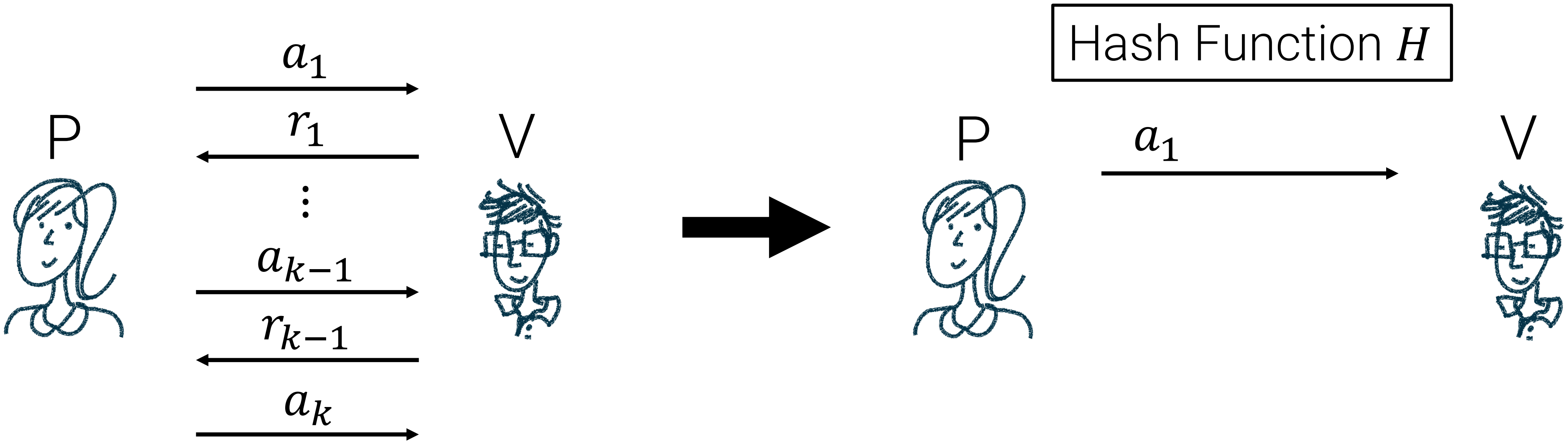
Non-Interactive Argument $FS_H(\Pi)$

Fiat-Shamir Heuristic

[FS86]

Magical compiler that removes interaction from (public-coin) interactive protocols

How? Replace random verifier messages with **hash** of previous messages



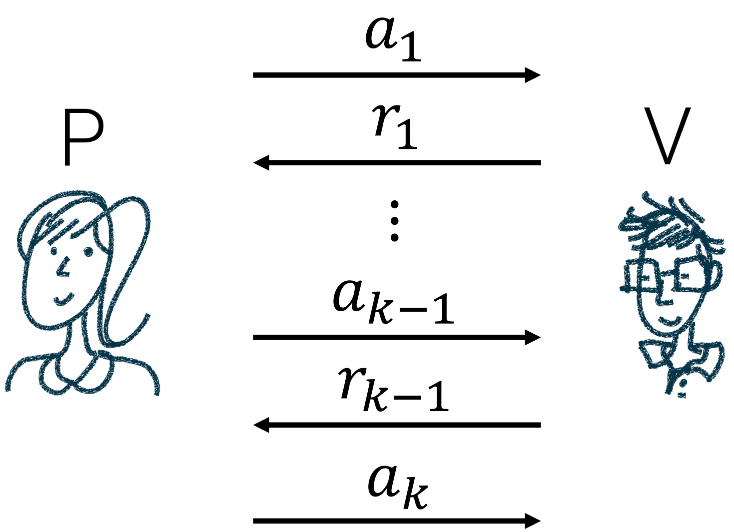
Public-Coin Interactive Protocol Π

Non-Interactive Argument $FS_H(\Pi)$

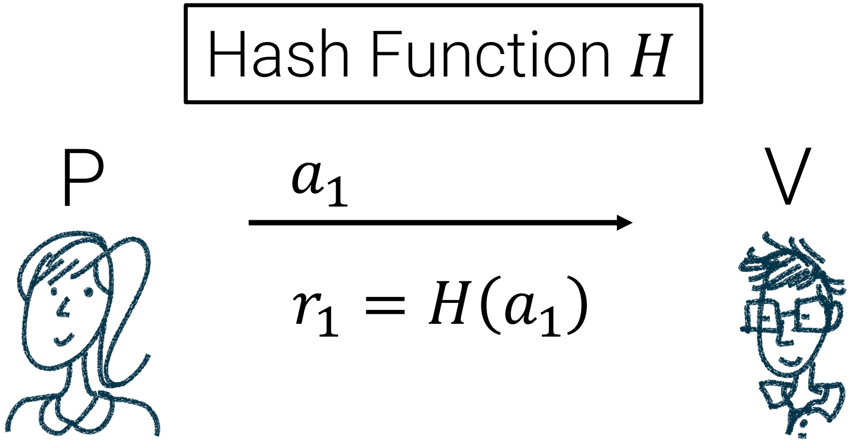
Fiat-Shamir Heuristic
[FS86]

Magical compiler that removes interaction from (public-coin) interactive protocols

How? Replace random verifier messages with **hash** of previous messages



Public-Coin Interactive Protocol Π



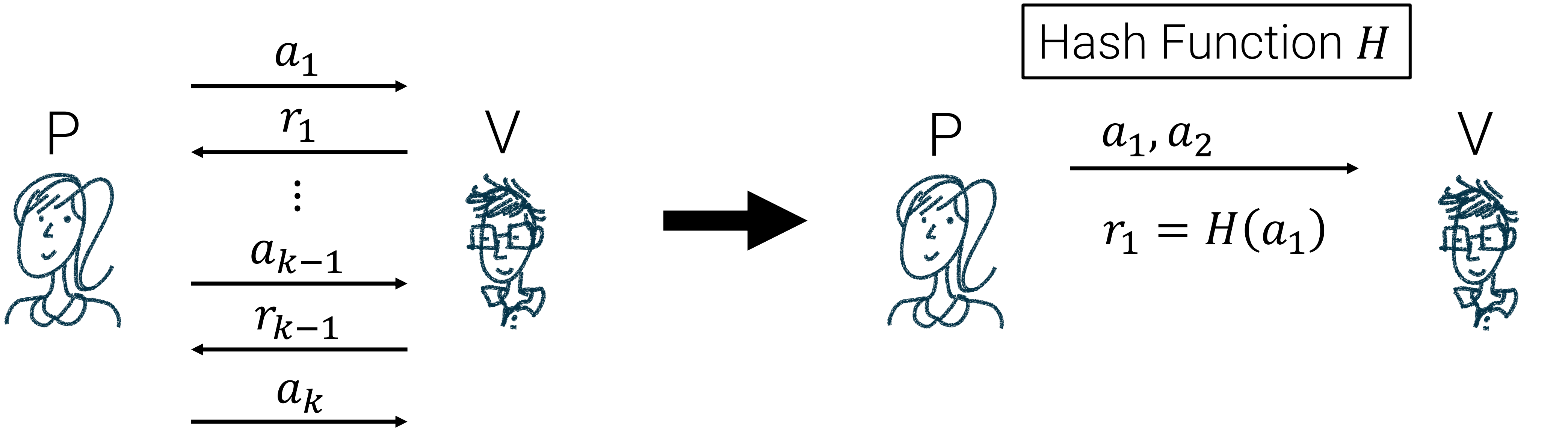
Non-Interactive Argument $FS_H(\Pi)$

Fiat-Shamir Heuristic

[FS86]

Magical compiler that removes interaction from (public-coin) interactive protocols

How? Replace random verifier messages with **hash** of previous messages



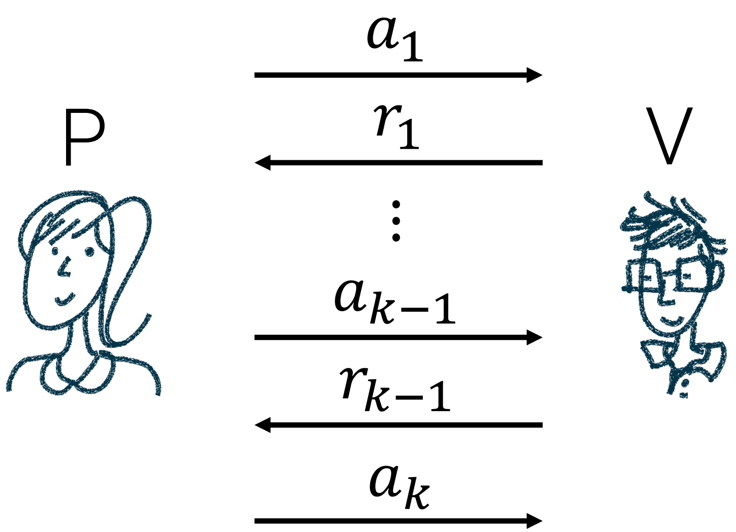
Public-Coin Interactive Protocol Π Non-Interactive Argument $FS_H(\Pi)$

Fiat-Shamir Heuristic

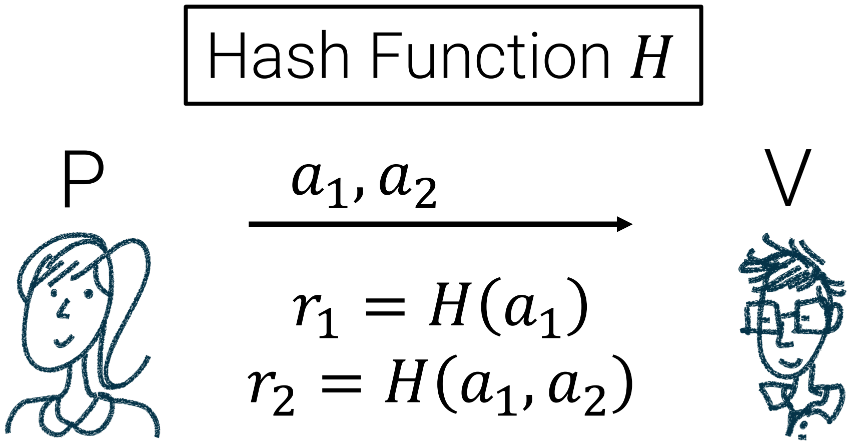
[FS86]

Magical compiler that removes interaction from (public-coin) interactive protocols

How? Replace random verifier messages with **hash** of previous messages



Public-Coin Interactive Protocol Π



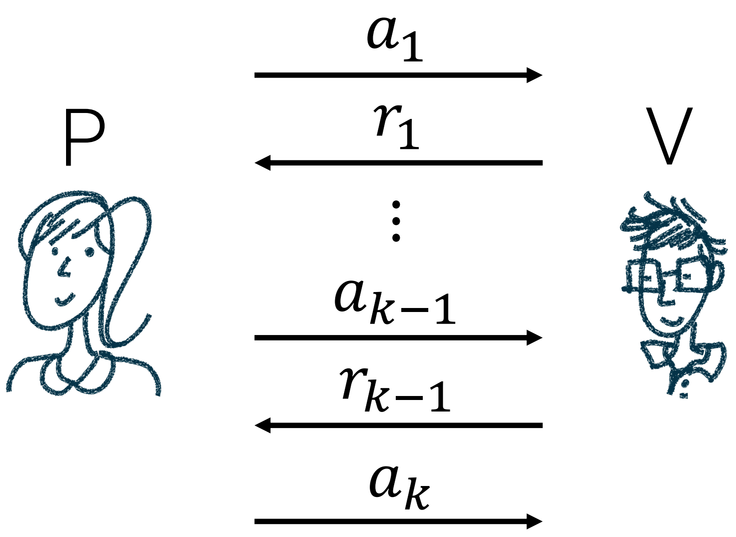
Non-Interactive Argument $FS_H(\Pi)$

Fiat-Shamir Heuristic

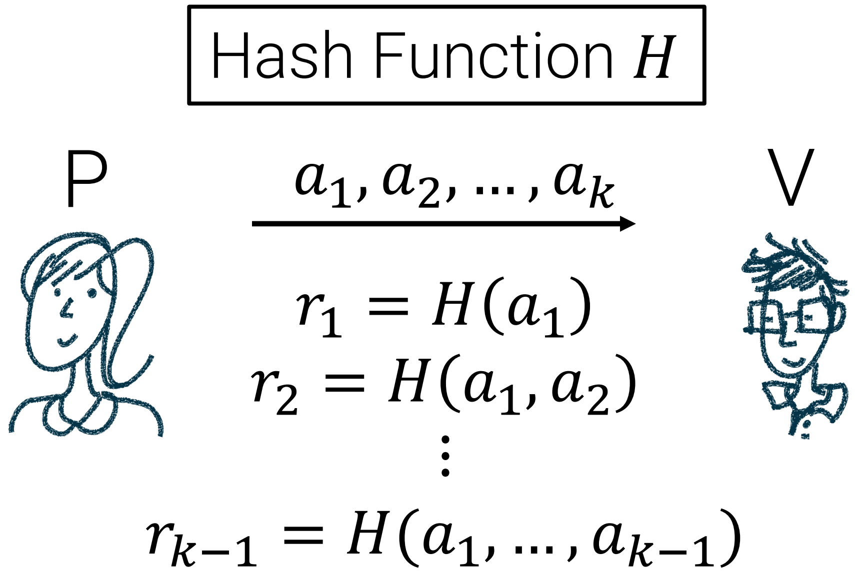
[FS86]

Magical compiler that removes interaction from (public-coin) interactive protocols

How? Replace random verifier messages with **hash** of previous messages

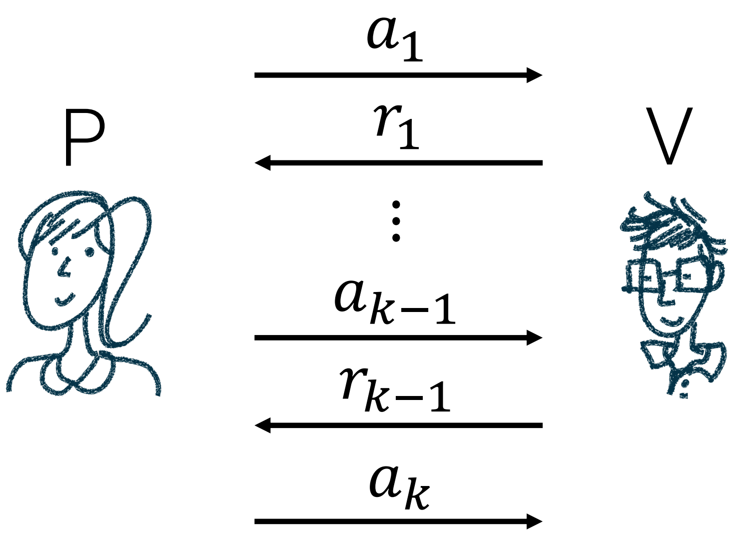


Public-Coin Interactive Protocol Π

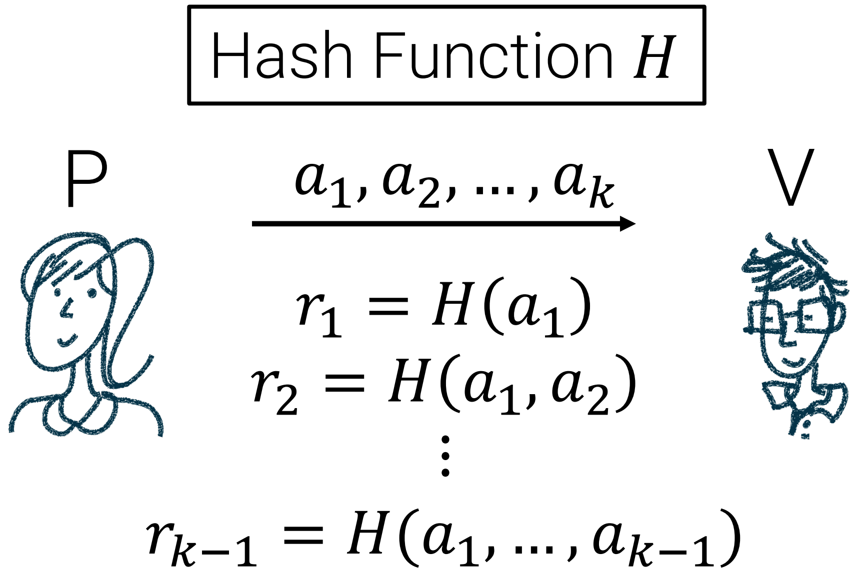


Non-Interactive Argument $FS_H(\Pi)$

When does Fiat-Shamir preserve soundness?



Public-Coin Interactive Protocol Π



Non-Interactive Argument $FS_H(\Pi)$

When does Fiat-Shamir preserve soundness?

- H is a random oracle (usually)

[FS86, BR93, PS96]

P

$$a_1 \longrightarrow$$
$$r_1 \longleftarrow$$
$$\vdots$$
$$a_{k-1} \longrightarrow$$
$$r_{k-1} \longleftarrow$$
$$a_k \longrightarrow$$

V

Public-Coin Interactive Protocol Π

Hash Function H

P

$$a_1, a_2, \dots, a_k \longrightarrow$$

$$r_1 = H(a_1)$$
$$r_2 = H(a_1, a_2)$$
$$\vdots$$
$$r_{k-1} = H(a_1, \dots, a_{k-1})$$

V

Non-Interactive Argument $FS_H(\Pi)$

When does Fiat-Shamir preserve soundness?

- H is a random oracle (usually)

[FS86, BR93, PS96]

- H is "correlation-intractable" (sometimes)

[CGH04, HMR08, CCR16, KRR17, CCRR18, CCHLRRW19, PS19, BKM20, LV20a, JKKZ20, LV20b …]

P

$\xrightarrow{a_1}$

$\xleftarrow{r_1}$

$\vdots$

$\xrightarrow{a_{k-1}}$

$\xleftarrow{r_{k-1}}$

$\xrightarrow{a_k}$

V

Public-Coin Interactive Protocol Π

Hash Function H

P

$\xrightarrow{a_1, a_2, \ldots, a_k}$

$$r_1 = H(a_1)$$
$$r_2 = H(a_1, a_2)$$
$$\vdots$$
$$r_{k-1} = H(a_1, \ldots, a_{k-1})$$

V

Non-Interactive Argument $FS_H(\Pi)$

Intuition: FS hash function should be complex/cryptographic. [Bellare-Rogaway93]

When instantiating a random oracle by a concrete function h, care must be taken first to ensure that h is adequately conservative in its design so as not to succumb to cryptanalytic attack, and second to ensure that h exposes no relevant "structure" attributable to its being defined from some lower-level primitive. Examples of both types of pitfalls are given in Section 6. As explained in that

Intuition: FS hash function should be complex/cryptographic. [Bellare-Rogaway93]

When instantiating a random oracle by a concrete function h, care must be taken first to ensure that h is adequately conservative in its design so as not to succumb to cryptanalytic attack, and second to ensure that h exposes no relevant "structure" attributable to its being defined from some lower-level primitive. Examples of both types of pitfalls are given in Section 6. As explained in that

What happens if the hash function exposes "structure"?

Intuition: FS hash function should be complex/cryptographic. [Bellare-Rogaway93]

When instantiating a random oracle by a concrete function h, care must be taken first to ensure that h is adequately conservative in its design so as not to succumb to cryptanalytic attack, and second to ensure that h exposes no relevant "structure" attributable to its being defined from some lower-level primitive. Examples of both types of pitfalls are given in Section 6. As explained in that

What happens if the hash function exposes "structure"?

This work: For some well-known protocols, soundness can still hold.

Result 1: Can compile some protocols* w/ ***simple, non-cryptographic***† FS hash functions.

* Examples:

- Lyubashevsky's ID protocol
- Schnorr's ID protocol
- Chaum-Pedersen protocol

† Examples:

- $H(x) = \mathrm{BitDecomp}(x)$
- $H(x) = ax + b\ (mod\ p)$

Result 1: Can compile some protocols* w/ ***simple, non-cryptographic***† FS hash functions.

Result 2: For many 3-message HVZK arguments‡, cryptographic FS hash function is ***necessary***.

* Examples:

- Lyubashevsky's ID protocol
- Schnorr's ID protocol
- Chaum-Pedersen protocol

† Examples:

- $H(x) = \mathrm{BitDecomp}(x)$
- $H(x) = ax + b\ (mod\ p)$

‡ Examples:

- Blum's Hamiltonicity protocol w/ parallel repetition
- GMW86 3-Coloring protocol w/ parallel repetition
- 1-bit challenge Schnorr w/ parallel repetition

Outline

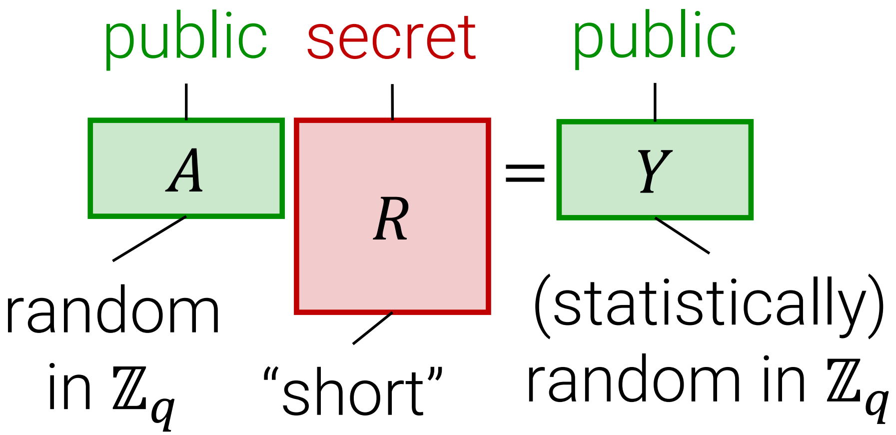
- Positive Results for Lyubashevsky
- Positive Results for Schnorr
- Negative Results

Outline

- Positive Results for Lyubashevsky
- Positive Results for Schnorr
- Negative Results

Review: Lyubashevsky's ID Protocol

[Lyu12]



Review: Lyubashevsky's ID Protocol

[Lyu12]

public — secret — public

$$A \cdot R = Y$$

A: random in $\mathbb{Z}_q$

R: "short"

Y: (statistically) random in $\mathbb{Z}_q$

I know a short pre-image of Y.

Review: Lyubashevsky's ID Protocol

[Lyu12]

public secret public

$A \; R = Y$

random in $\mathbb{Z}_q$ "short" (statistically) random in $\mathbb{Z}_q$

I know a short pre-image of Y.

$\alpha = A \; t$ "short"

Review: Lyubashevsky's ID Protocol

[Lyu12]

public secret public

$$\boxed{A}\ \boxed{R} = \boxed{Y}$$

random in $\mathbb{Z}_q$

"short"

(statistically) random in $\mathbb{Z}_q$

I know a short pre-image of Y.

$$\boxed{\alpha} = \boxed{A}\ \boxed{t}$$

"short"

$\longrightarrow$

$\boxed{c}$ random in {0,1}

$\longleftarrow$

Review: Lyubashevsky's ID Protocol

[Lyu12]

public | secret | public

$$A \, R = Y$$

A: random in $\mathbb{Z}_q$ (public)

R: "short" (secret)

Y: (statistically) random in $\mathbb{Z}_q$ (public)

I know a short pre-image of Y.

$$\alpha = A \, t$$

t: "short"

c: random in {0,1}

Review: Lyubashevsky's ID Protocol

[Lyu12]

public / secret / public

$$A \, R = Y$$

A: public, random in $\mathbb{Z}_q$

R: secret, "short"

Y: public, (statistically) random in $\mathbb{Z}_q$

I know a short pre-image of Y.

$$\alpha = A \, t$$

t: "short"

$\longrightarrow$

c: random in {0,1}

$\longleftarrow$

$$z = t + R \, c$$

$\longrightarrow$

Review: Lyubashevsky's ID Protocol
[Lyu12]

public secret public

$$A \, R = Y$$

random in $\mathbb{Z}_q$ "short" (statistically) random in $\mathbb{Z}_q$

I know a short pre-image of Y.

$$\alpha = A \, t$$

"short"

c random in {0,1}

$$z = t + R \, c$$

z independent of R by noise-flooding or rejection sampling

Review: Lyubashevsky's ID Protocol

[Lyu12]

public: A — random in $\mathbb{Z}_q$

secret: R — "short"

public: Y — (statistically) random in $\mathbb{Z}_q$

$$A R = Y$$

I know a short pre-image of Y.

$$\alpha = A t$$

t "short"

$\longrightarrow$

c random in {0,1}

$\longleftarrow$

$$z = t + R c$$

$\longrightarrow$

z independent of R by noise-flooding or rejection sampling

Accept if

$$A z = \alpha + Y c$$

and z is "short".

Review: Lyubashevsky's ID Protocol

[Lyu12]

$$A \, R = Y$$

A: public, random in $\mathbb{Z}_q$

R: secret, "short"

Y: public, (statistically) random in $\mathbb{Z}_q$

Soundness (Average Case)

If  accepts with good probability,  breaks SIS.

I know a short pre-image of Y.

$$\alpha = A \, t$$

t: "short"

$$c$$

c: random in {0,1}

$$z = t + R \, c$$

z independent of R by noise-flooding or rejection sampling

Accept if

$$A \, z = \alpha + Y \, c$$

and z is "short".

Review: Lyubashevsky's ID Protocol

[Lyu12]

public $\underbrace{A}_{\text{random in } \mathbb{Z}_q}$ secret $\underbrace{R}_{\text{"short"}}$ $=$ public $\underbrace{Y}_{\text{(statistically) random in } \mathbb{Z}_q}$

I know a short pre-image of Y.

$\alpha = A\, t$ — t "short"

$\longrightarrow$

c random in {0,1}

$\longleftarrow$

$z = t + R\, c$

$\longrightarrow$

z independent of R by noise-flooding or rejection sampling

Soundness (Average Case)

Run on $c \neq c'$, get

$$z = t + R\, c$$

$$z' = t + R\, c'$$

Accept if

$$A\, z = \alpha + Y\, c$$

and z is "short".

Review: Lyubashevsky's ID Protocol
[Lyu12]

public A — random in $\mathbb{Z}_q$

secret R — "short"

public Y — (statistically) random in $\mathbb{Z}_q$

$$A R = Y$$

I know a short pre-image of Y.

$$\alpha = A t$$

t "short"

c random in {0,1}

$$z = t + R c$$

z independent of R by noise-flooding or rejection sampling

Soundness (Average Case)

Subtract

$$z - z' = R\,(c - c')$$

Accept if

$$A z = \alpha + Y c$$

and z is "short".

Review: Lyubashevsky's ID Protocol

[Lyu12]

public: A — random in $\mathbb{Z}_q$; secret: R — "short"; public: Y — (statistically) random in $\mathbb{Z}_q$

$$A\,R = Y$$

I know a short pre-image of Y.

$$\alpha = A\,t$$

t — "short"

c — random in {0,1}

$$z = t + R\,c$$

z independent of R by noise-flooding or rejection sampling

Soundness (Average Case)

Multiply by A, rearrange:

$$\begin{bmatrix} A & -Y \end{bmatrix} \begin{bmatrix} z - z' \\ c - c' \end{bmatrix} = \boxed{0}$$

Accept if

$$A\,z = \alpha + Y\,c$$

and z is "short".

Review: Lyubashevsky's ID Protocol

[Lyu12]

$$\underbrace{A}_{\text{public; random in } \mathbb{Z}_q} \; \underbrace{R}_{\text{secret; "short"}} = \underbrace{Y}_{\text{public; (statistically) random in } \mathbb{Z}_q}$$

I know a short pre-image of Y.

$\alpha = A\,t$ — t "short"

c — random in {0,1}

$z = t + R\,c$

z independent of R by noise-flooding or rejection sampling

Honest Verifier ZK

Pick random c, z, set

$$\alpha = A\,z - Y\,c$$

Accept if

$$A\,z = \alpha + Y\,c$$

and z is "short".

Lyubashevsky + Fiat-Shamir

public · secret · public

$$A \, R = Y$$

A: random in $\mathbb{Z}_q$ · R: "short" · Y: (statistically) random in $\mathbb{Z}_q$

FS hash h: $x \longrightarrow y$

x: $\mathbb{Z}_q$ · y: $\{0,1\}$

I know a short pre-image of Y.

$$\alpha = A \, t$$

t: "short"

$$z = t + R \, h(\alpha)$$

Accept if

$$A \, z = \alpha + Y \, h(\alpha)$$

and z is "short".

Soundness: Must use h where hard

to find α and short z satisfying:

$$A\,z = \alpha + Y\,\square$$

random in $\mathbb{Z}_q$ $h(\alpha)$

Soundness: Must use h where hard to find α and short z satisfying:

$$A\,z = \alpha + Y\,h(\alpha)$$

random in $\mathbb{Z}_q$ $\qquad$ $h(\alpha)$

Key Idea: What if $\alpha = G\,h(\alpha)$?

Soundness: Must use h where hard to find α and short z satisfying:

$$A z = \alpha + Y h(\alpha)$$

random in $\mathbb{Z}_q$ $\qquad$ $h(\alpha)$

Key Idea: What if $\alpha = G \, h(\alpha)$?

For example, $h(\alpha) = BitDecomp(\alpha)$,

$$G = \begin{bmatrix} 1,2,4,\ldots & & & \\ & 1,2,4,\ldots. & & \\ & & \ddots & \\ & & & 1,2,4,\ldots \end{bmatrix}.$$

Soundness: Must use h where hard to find α and short z satisfying:

Key Idea: What if $\alpha = G\,h(\alpha)$?

$$A z = \alpha + Y\,h(\alpha) \longrightarrow A z = G\,h(\alpha) + Y\,h(\alpha)$$

random in $\mathbb{Z}_q$

Soundness: Must use h where hard to find α and short z satisfying:

$$A z = \alpha + Y\,h(\alpha)$$

random in $\mathbb{Z}_q$

Key Idea: What if $\alpha = G\,h(\alpha)$?

$$A z = G\,h(\alpha) + Y\,h(\alpha)$$

$$\begin{bmatrix} -A & G+Y \end{bmatrix} \begin{bmatrix} z \\ h(\alpha) \end{bmatrix} = \boxed{0}$$

random in $\mathbb{Z}_q$

Since $\begin{bmatrix} z \\ h(\alpha) \end{bmatrix}$ is short, hard to find under SIS!

Another Perspective: If you know R where $\underbrace{A}_{\text{random in } \mathbb{Z}_q} \underbrace{R}_{\text{"short"}} = \underbrace{Y}_{\text{(statistically) random in } \mathbb{Z}_q}$

then you can find short solution $\begin{bmatrix} z \\ h(\alpha) \end{bmatrix}$ to $\begin{bmatrix} -A & G+Y \end{bmatrix} \begin{bmatrix} z \\ h(\alpha) \end{bmatrix} = \boxed{0}$

Another Perspective: If you know R where

$$\underbrace{A}_{\text{random in } \mathbb{Z}_q}\ \underbrace{R}_{\text{“short”}} = \underbrace{Y}_{\text{(statistically) random in } \mathbb{Z}_q}$$

then you can find short solution $\begin{bmatrix} z \\ h(\alpha) \end{bmatrix}$ to

$$\begin{bmatrix} -A & G+Y \end{bmatrix} \begin{bmatrix} z \\ h(\alpha) \end{bmatrix} = \boxed{0}$$

This is exactly the MP12/LW15 lattice trapdoor!

In an alternate timeline, we could have ***discovered*** lattice trapdoors from trying to Fiat-Shamir Lyubashevsky’s protocol.

Theorem. If there exists $\boxed{G}$ such that $\boxed{\alpha} = \boxed{G}\,\boxed{h(\alpha)}$

then h is secure FS hash for [Lyu12] ID scheme (under SIS).

Theorem. If there exists $\boxed{G}$ such that $\boxed{\alpha} = \boxed{G}\,\boxed{h(\alpha)}$

then h is secure FS hash for [Lyu12] ID scheme (under SIS).

What does this say about signatures?

We have two approaches for constructing lattice-based signatures:

GPV08
(Preimage Sampleable Functions)

$f_A(x) = Ax$ where trapdoor T enables pre-image sampling.

Sign m by applying random oracle $RO(m)$ and use T to find preimage of $RO(m)$.

Fiat-Shamir + Lyubashevsky
("Lattice Signatures w/o Trapdoors")

Compile Lyubashevsky ID protocol into signature Fiat-Shamir.

We have two approaches for constructing lattice-based signatures:

GPV08
(Preimage Sampleable Functions)

$f_A(x) = Ax$ where trapdoor T enables pre-image sampling.

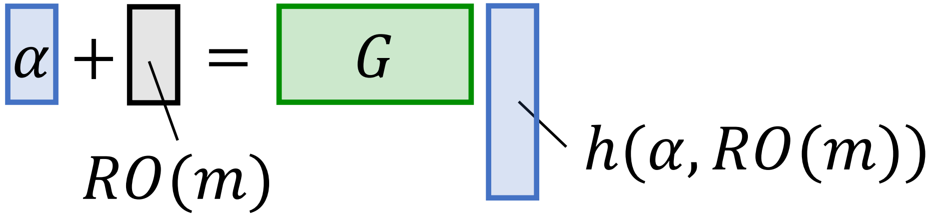
Sign m by applying random oracle $RO(m)$ and use T to find preimage of $RO(m)$.

Fiat-Shamir + Lyubashevsky
("Lattice Signatures w/o Trapdoors")

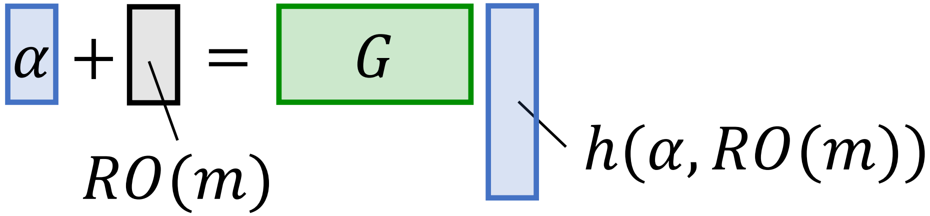
Compile Lyubashevsky ID protocol into signature Fiat-Shamir.

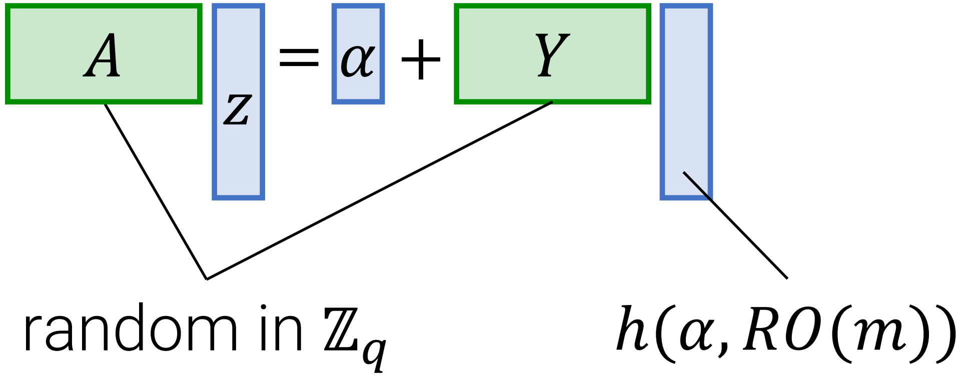
Claim. [GPV08] with [MP12] trapdoor can be viewed as Hash-and-Sign applied to $FS_h[\Pi_{Lyu}]$ where FS hash function is $h(\alpha, x) = G^{-1}(\alpha + x)$.

If $h\big(\alpha, RO(m)\big) = G^{-1}(\alpha + RO(m))$:

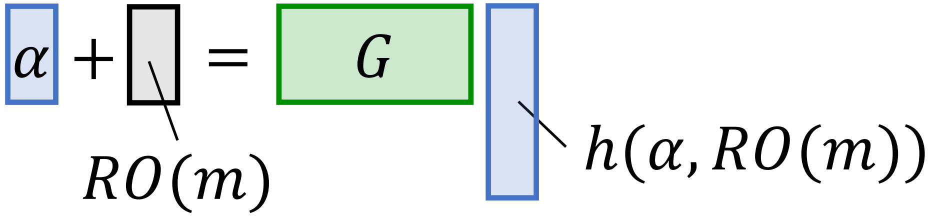


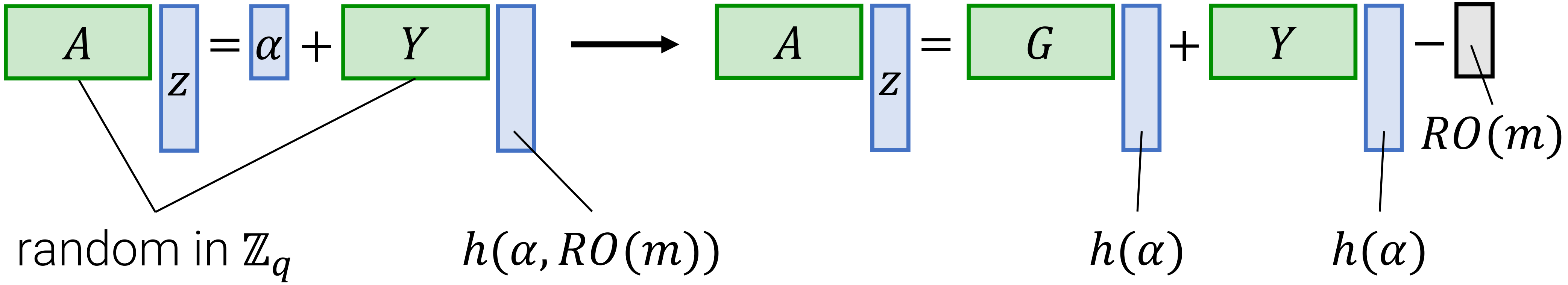
If $h\big(\alpha, RO(m)\big) = G^{-1}(\alpha + RO(m))$:



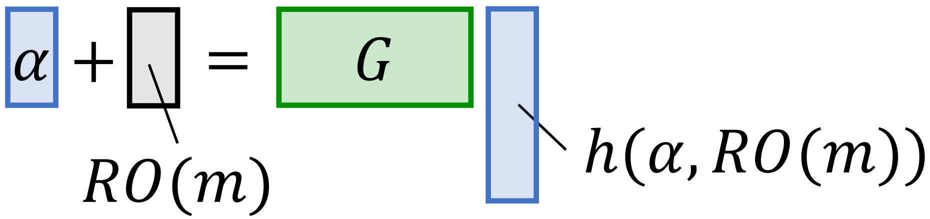


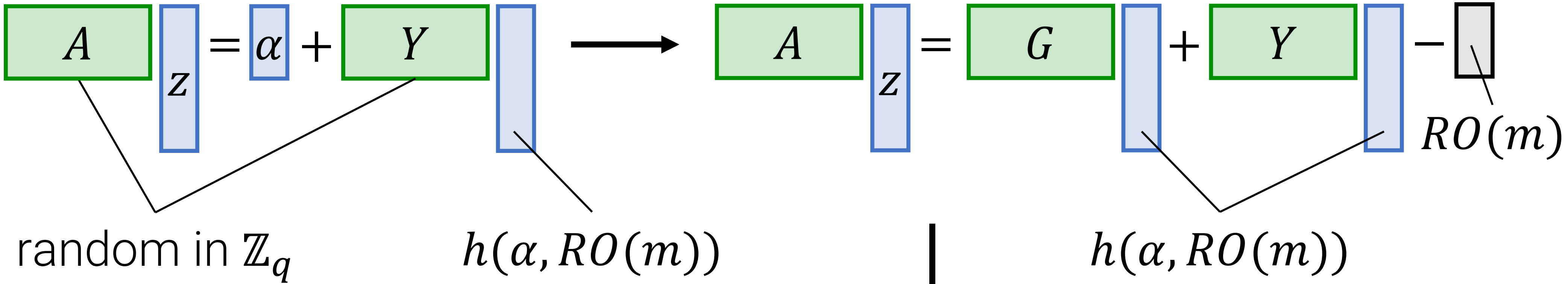
If $h\big(\alpha, RO(m)\big) = G^{-1}(\alpha + RO(m))$:

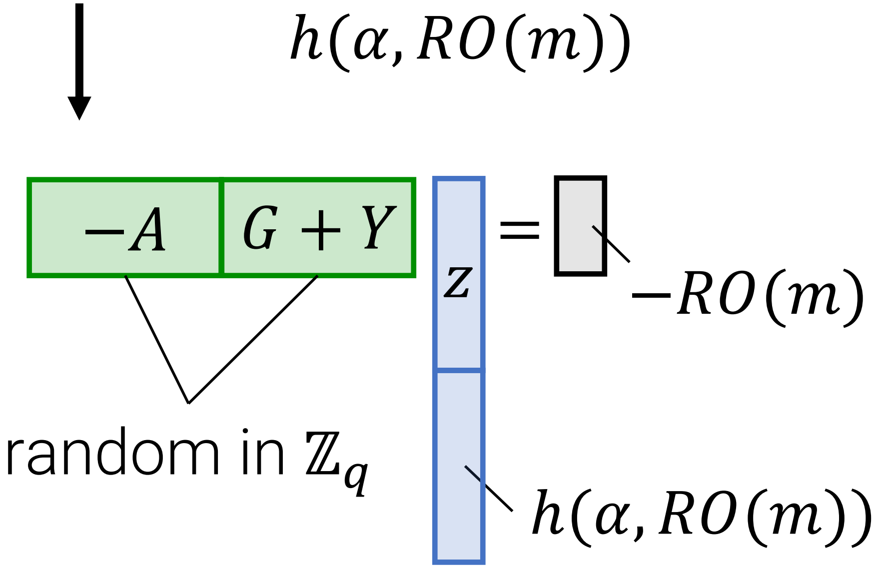




If $h\big(\alpha, RO(m)\big) = G^{-1}(\alpha + RO(m))$:







As in [GPV08], a signature is a preimage of $RO(m)$!

Outline

- Positive Results for Lyubashevsky
- Positive Results for Schnorr
- Negative Results

Review: Schnorr's ID Protocol [S91]

Group G of order p
with generator g

public g^x

Review: Schnorr's ID Protocol [S91]

Group G of order p
with generator g

I know x

public g^x

Review: Schnorr's ID Protocol [S91]

Group G of order p with generator g

I know x

public g^x

Sample random $r \leftarrow \mathbb{Z}_p$.

$\xrightarrow{g^r}$

Review: Schnorr's ID Protocol [S91]

Group G of order p
with generator g

I know x

public g^x

Sample random $r \leftarrow \mathbb{Z}_p$.

$\xrightarrow{g^r}$

$\xleftarrow{c}$

Sample random $c \leftarrow \mathbb{Z}_p$.

Review: Schnorr's ID Protocol [S91]

Group G of order p with generator g

I know x

public g^x

Sample random $r \leftarrow \mathbb{Z}_p$.

$\xrightarrow{g^r}$

$\xleftarrow{c}$

$\xrightarrow{z}$

Sample random $c \leftarrow \mathbb{Z}_p$.

Compute $z = r + cx$.

Accept if $g^z = g^r (g^x)^c$.

Review: Schnorr's ID Protocol [S91]

Group G of order p with generator g

I know x

public g^x

Sample random $r \leftarrow \mathbb{Z}_p$.

$\xrightarrow{g^r}$

$\xleftarrow{c}$

$\xrightarrow{z}$

Sample random $c \leftarrow \mathbb{Z}_p$.

Compute $z = r + cx$.

Accept if $g^z = g^r (g^x)^c$.

Proof of Knowledge: If  accepts w/ good probability, can extract x from .

(run  on $c_1 \neq c_2$; solve $z_1 = r + c_1 x$ and $z_2 = r + c_2 x$ for x)

Review: Schnorr's ID Protocol [S91]

Group G of order p with generator g

I know x

public g^x

Sample random $r \leftarrow \mathbb{Z}_p$.

$\xrightarrow{g^r}$

$\xleftarrow{c}$

$\xrightarrow{z}$

Sample random $c \leftarrow \mathbb{Z}_p$.

Compute $z = r + cx$.

Accept if $g^z = g^r (g^x)^c$.

Honest Verifier ZK: Can simulate honest verifier accepting transcripts.

(pick random c, z, set $g^r = g^z (g^x)^{-c}$).

Schnorr
+ Fiat-Shamir

Group G of order p with generator g

FS hash $H: G \to \mathbb{Z}_p$

public g^x

Sample random $r \leftarrow \mathbb{Z}_p$.

Compute $z = r + H(g^r)x$.

$\xrightarrow{g^r, z}$

Accept if $g^z = g^r (g^x)^{H(g^r)}$.

Schnorr
+ Fiat-Shamir

Group G of order p with generator g

FS hash $H: G \to \mathbb{Z}_p$

public g^x

Sample random $r \leftarrow \mathbb{Z}_p$.

Compute $z = r + H(g^r)x$.

$\xrightarrow{g^r, z}$

Accept if $g^z = g^r (g^x)^{H(g^r)}$.

Important Open Question: For what H is this sound?

Schnorr
+ Fiat-Shamir

Group G of order p
with generator g

FS hash
$H: G \to \mathbb{Z}_p$

public g^x

Sample random $r \leftarrow \mathbb{Z}_p$.

Compute $z = r + H(g^r)x$.

$\xrightarrow{g^r, z}$

Accept if $g^z = g^r (g^x)^{H(g^r)}$.

Important Open Question: For what H is this sound?

Let's ask a different question…

For what H is this *unsound*?

Schnorr
+ Fiat-Shamir

Group G of order p with generator g

FS hash $H: G \to \mathbb{Z}_p$

public g^x

Sample random $r \leftarrow \mathbb{Z}_p$.

Compute $z = r + H(g^r)x$.

$\xrightarrow{g^r, z}$

Accept if $g^z = g^r (g^x)^{H(g^r)}$.

Rephrased: For what H is it possible to break FS-Schnorr for *any* group G?

Schnorr
+ Fiat-Shamir

Group G of order p with generator g

FS hash $H: G \to \mathbb{Z}_p$

public g^x

Sample random $r \leftarrow \mathbb{Z}_p$.

Compute $z = r + H(g^r)x$.

$\xrightarrow{g^r, z}$

Accept if $g^z = g^r (g^x)^{H(g^r)}$.

Rephrased: For what H is it possible to break FS-Schnorr for ***any*** group G?

- Constant functions: If $H(g^r) = k$ for all g^r, set $g^r = (g^x)^{-k}$ and $z = 0$.

Schnorr + Fiat-Shamir

Group G of order p with generator g

FS hash $H: G \to \mathbb{Z}_p$

public g^x

Sample random $r \leftarrow \mathbb{Z}_p$.

Compute $z = r + H(g^r)x$.

$\xrightarrow{g^r, z}$

Accept if $g^z = g^r (g^x)^{H(g^r)}$.

Rephrased: For what H is it possible to break FS-Schnorr for *any* group G?

- Constant functions: If $H(g^r) = k$ for all g^r, set $g^r = (g^x)^{-k}$ and $z = 0$.
- "Constant on many inputs" : If $H(g^r) = k$ for ε fraction of g^r, same attack works with ε probability.

Schnorr
+ Fiat-Shamir

Group G of order p with generator g

FS hash $H: G \to \mathbb{Z}_p$

public g^x

Sample random $r \leftarrow \mathbb{Z}_p$.

Compute $z = r + H(g^r)x$.

$\xrightarrow{g^r, z}$

Accept if $g^z = g^r (g^x)^{H(g^r)}$.

FS-Schnorr always *insecure* for these H.

$H(g^r) = k$ for noticeable fraction of g^r

All functions H

Schnorr
+ Fiat-Shamir

Group G of order p with generator g

FS hash $H: G \to \mathbb{Z}_p$

public g^x

Sample random $r \leftarrow \mathbb{Z}_p$.

Compute $z = r + H(g^r)x$.

$\xrightarrow{g^r,\, z}$

Accept if $g^z = g^r (g^x)^{H(g^r)}$.

$H(g^r)$ has $\omega(log\ \lambda)$ min-entropy on random g^r

$H(g^r) = k$ for noticeable fraction of g^r

All functions H

FS-Schnorr always *insecure* for these H.

Schnorr
+ Fiat-Shamir

Group G of order p with generator g

FS hash $H : G \to \mathbb{Z}_p$

public g^x

Sample random $r \leftarrow \mathbb{Z}_p$.

Compute $z = r + H(g^r)x$.

$\xrightarrow{g^r, z}$

Accept if $g^z = g^r (g^x)^{H(g^r)}$.

Thm: FS-Schnorr *secure* in "Generic Group Model" for these H!

FS-Schnorr always *insecure* for these H.

$H(g^r)$ has $\omega(log\ \lambda)$ min-entropy on random g^r

$H(g^r) = k$ for noticeable fraction of g^r

All functions H

Aside: Generic Group Model (GGM) [N94,S97,M95]

Tagline: Idealized interface that ***only*** allows “honest” use of the group.

Aside: Generic Group Model (GGM) [N94,S97,M95]

Tagline: Idealized interface that *only* allows "honest" use of the group.

1) Sample random injection $\sigma: \mathbb{Z}_p \to \{0,1\}^\ell$.

GGM Oracle

x	$\sigma(x)$
0	10100100
1	01010111
$\vdots$	$\vdots$
$p-1$	10010110

Aside: Generic Group Model (GGM) [N94,S97,M95]

Tagline: Idealized interface that *only* allows "honest" use of the group.

1) Sample random injection $\sigma: \mathbb{Z}_p \to \{0,1\}^\ell$.
2) Replace each g^x with "label" $\sigma(x)$.

GGM Oracle

x	$\sigma(x)$
0	10100100
1	01010111
$\vdots$	$\vdots$
$p-1$	10010110

$$\xrightarrow{\sigma(x_1), \dots, \sigma(x_k)}$$

Aside: Generic Group Model (GGM) [N94,S97,M95]

Tagline: Idealized interface that *only* allows "honest" use of the group.

1) Sample random injection $\sigma: \mathbb{Z}_p \to \{0,1\}^\ell$.
2) Replace each g^x with "label" $\sigma(x)$.
3) Permit group operations via oracle queries

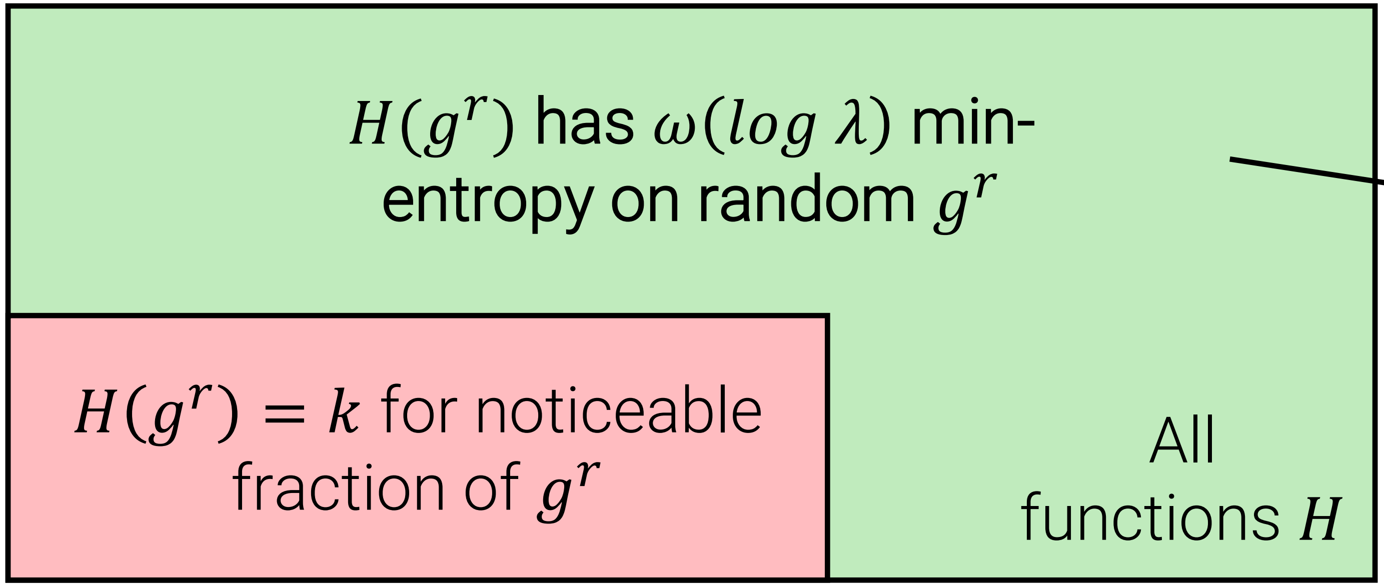
GGM Oracle

x	$\sigma(x)$
0	10100100
1	01010111
$\vdots$	$\vdots$
$p-1$	10010110

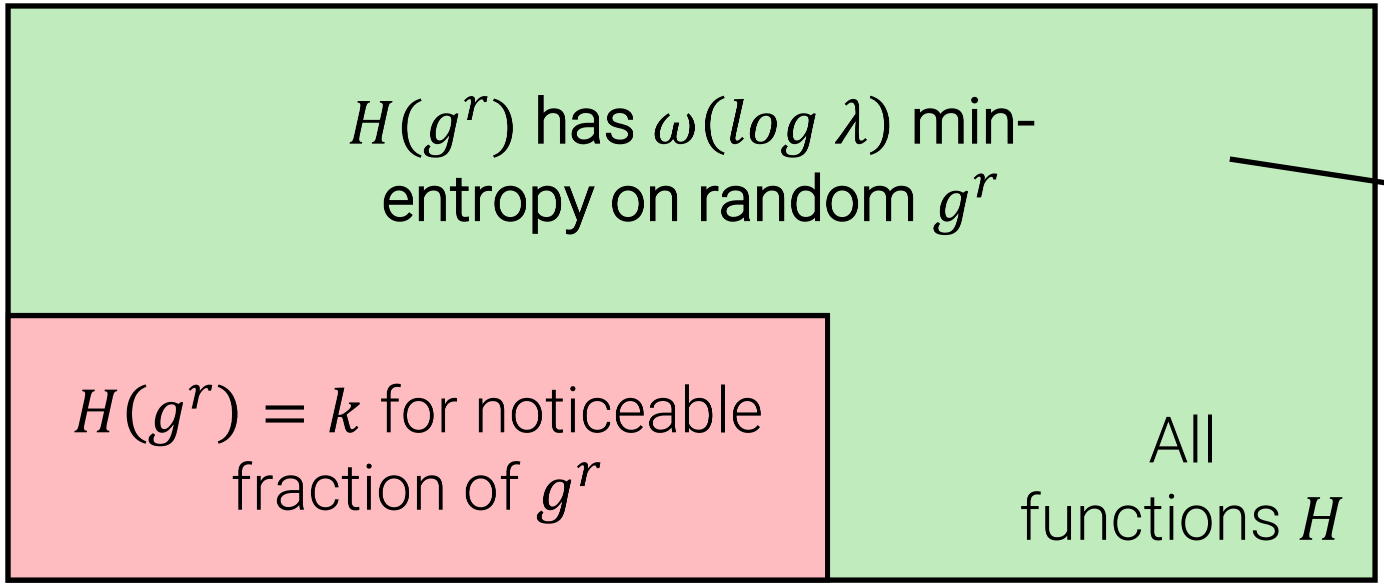
$\sigma(x_1), \ldots, \sigma(x_k)$ $\longrightarrow$

$\longleftarrow$ $\sigma(x), \sigma(y), a, b$

$\sigma(ax+by)$ $\longrightarrow$



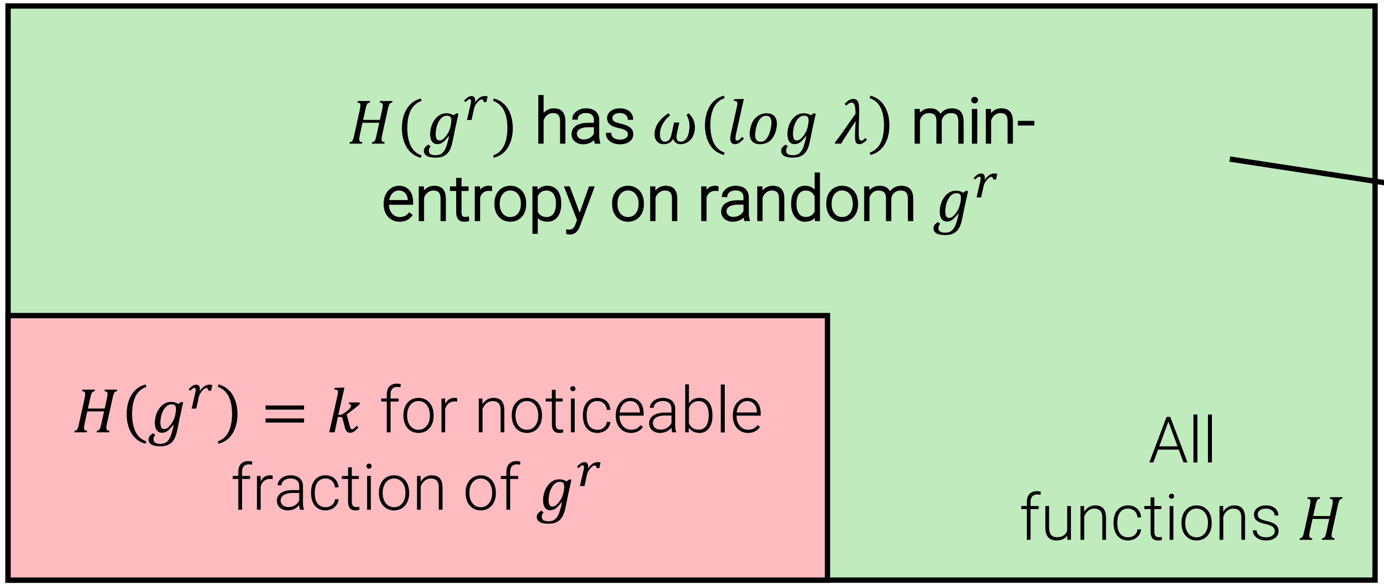
Theorem: FS-Schnorr *secure* in GGM for these H.
(captures all “reasonable” H)



Theorem: FS-Schnorr *secure* in GGM for these H.
(captures all "reasonable" H)

Example: $H: G \to \mathbb{Z}_p$ where

$H(g^r) :=$ "interpret g^r as a bit-string and reduce mod p"



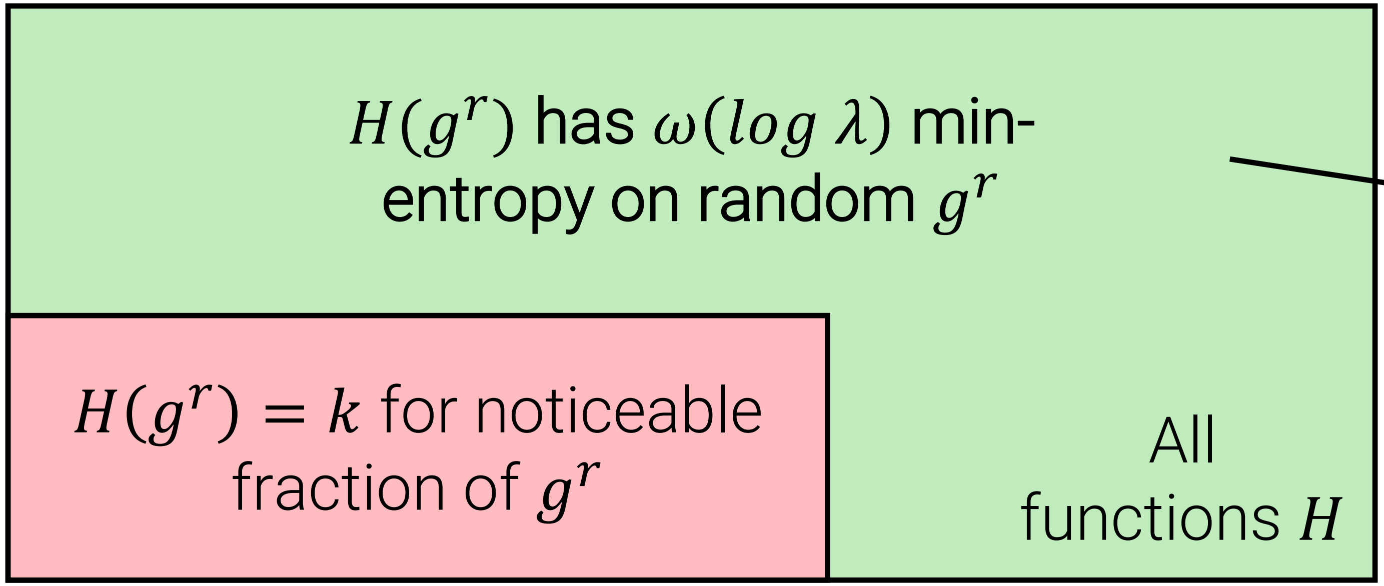
Theorem: FS-Schnorr *secure* in GGM for these H.
(captures all "reasonable" H)

Example: $H: G \to \mathbb{Z}_p$ where

$H(g^r) :=$ "interpret g^r as a bit-string and reduce mod p"

This extends to Schnorr signatures!*

*Similar to analysis by [NSW09]



Theorem: FS-Schnorr *secure* in GGM for these H.
(captures all "reasonable" H)

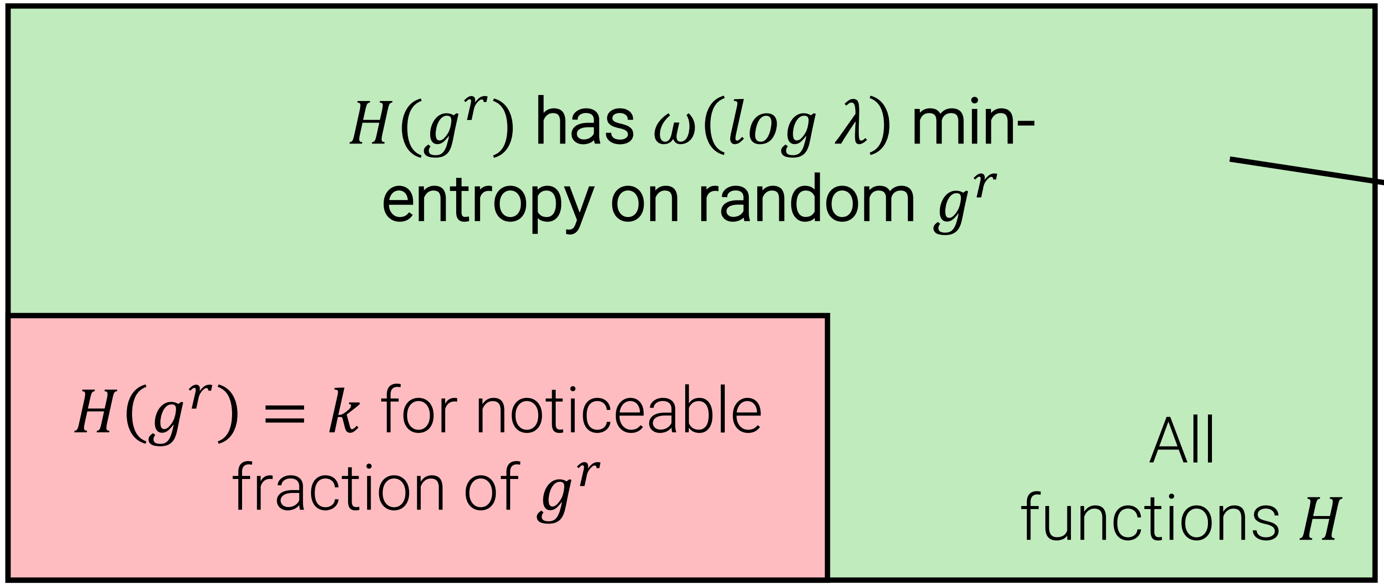
Example: $H: G \to \mathbb{Z}_p$ where

$H(g^r) :=$ "interpret g^r as a bit-string and reduce mod p"

This extends to Schnorr signatures!*

(Example) Theorem: Schnorr sigs are EUF-CMA *secure* in GGM for $H: G \times M \to \mathbb{Z}_p$ where $H(g^r, m) := g^r + m\ (mod\ p)$ if $|M|/p$ is negligible.

*Similar to analysis by [NSW09]



Theorem: FS-Schnorr *secure* in GGM for these H.
(captures all "reasonable" H)

Example: $H: G \to \mathbb{Z}_p$ where

$H(g^r) :=$ "interpret g^r as a bit-string and reduce mod p"

This extends to Schnorr signatures!*

(Example) Theorem: Schnorr sigs are EUF-CMA *secure* in GGM for $H: G \times M \to \mathbb{Z}_p$ where $H(g^r, m) := g^r + m\ (mod\ p)$ if $|M|/p$ is negligible.

*Similar to analysis by [NSW09]

…but the story doesn't end here

This H is insecure in practice!

(Example) Theorem: Schnorr sigs are EUF-CMA *secure* in GGM for $H: G \times M \to \mathbb{Z}_p$ where $H(g^r, m) \coloneqq g^r + m \ (mod\ p)$ if $|M|/p$ is negligible.

This H is insecure in practice!

Attack: We show a non-uniform attack on this signature scheme in **any concrete group**.

(Example) Theorem: Schnorr sigs are EUF-CMA *secure* in GGM for $H: G \times M \to \mathbb{Z}_p$ where $H(g^r, m) \coloneqq g^r + m \ (mod\ p)$ if $|M|/p$ is negligible.

This H is insecure in practice!

Attack: We show a non-uniform attack on this signature scheme in **any concrete group**.

(also applies to [NSW09] Schnorr signatures)

(Example) Theorem: Schnorr sigs are EUF-CMA *secure* in GGM for $H: G \times M \to \mathbb{Z}_p$ where $H(g^r, m) := g^r + m \ (mod\ p)$ if $|M|/p$ is negligible.

$$H(g^r, m) = g^r + m \ (mod\ p)$$

Group G of order p with generator g

Signing key g^x.

Recall: Valid signature on m is (g^r, z) where:

$$g^z = g^r (g^x)^{g^r + m\ (mod\ p)}$$

$$H(g^r, m) = g^r + m \ (mod\ p)$$

Group G of order p with generator g

Signing key g^x.

Recall: Valid signature on m is (g^r, z) where:

$$g^z = g^r (g^x)^{g^r + m \ (mod\ p)}$$

Non-Uniform Attack

- Advice: (m, r) where the bit-representation of g^r is $-m \ (mod\ p)$.
- Attack: Output $(m, g^r, z = r)$.

$$\boxed{H(g^r, m) = g^r + m \ (mod\ p)}$$

Group G of order p with generator g

Signing key g^x.

Recall: Valid signature on m is (g^r, z) where:

$$g^z = g^r (g^x)^{g^r + m \ (mod\ p)}$$

Non-Uniform Attack

- Advice: (m, r) where the bit-representation of g^r is $-m \ (mod\ p)$.
- Attack: Output $(m, g^r, z = r)$.

Over $\mathbb{Z}_p^\times$ and elliptic curve groups, this attack can be done *without* advice!

Problem: GGM fails to capture *non-uniform attacks*.

However, this is a known problem of the GGM, and we can (essentially) recover our positive results in the *preprocessing GGM*:

Problem: GGM fails to capture *non-uniform attacks*.

However, this is a known problem of the GGM, and we can (essentially) recover our positive results in the *preprocessing GGM*:

GGM Oracle

x	$\sigma(x)$
0	10100100
1	01010111
$\vdots$	$\vdots$
$p-1$	10010110

x	$\sigma(x)$
0	10100100
1	01010111
$\vdots$	$\vdots$
$p-1$	10010110

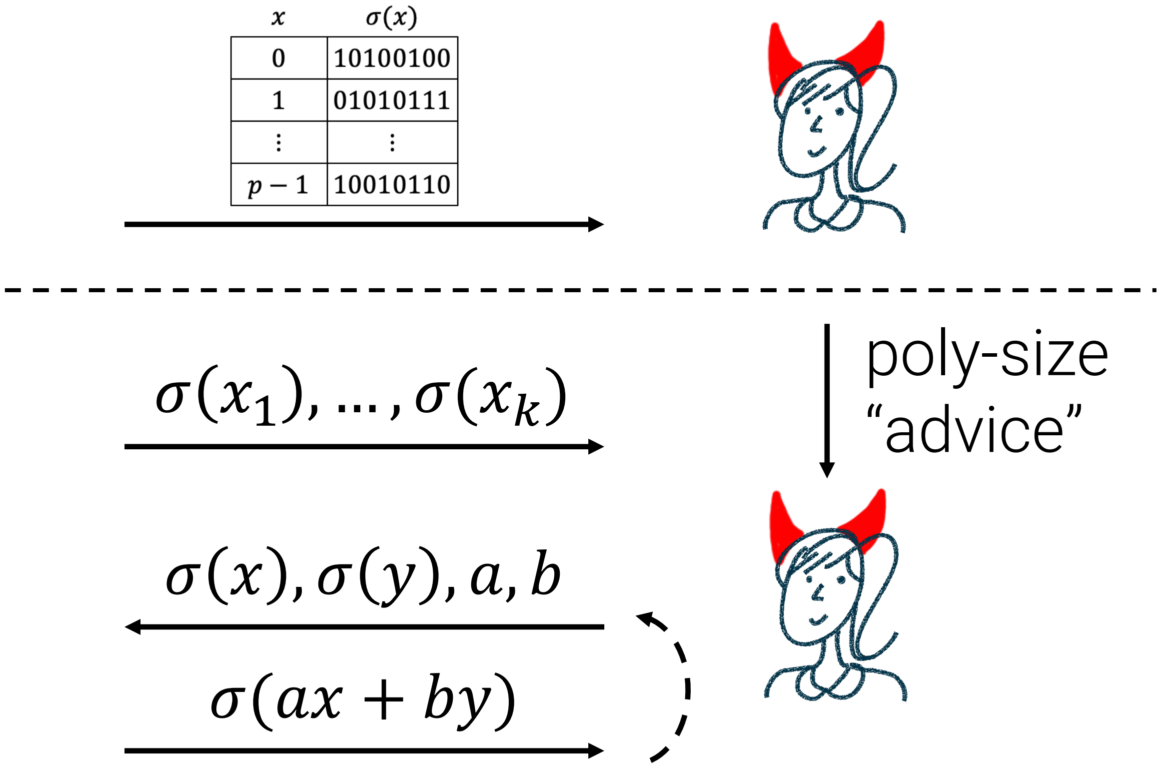
poly-size
"advice"

Problem: GGM fails to capture ***non-uniform attacks***.

However, this is a known problem of the GGM, and we can (essentially) recover our positive results in the ***preprocessing GGM***:

GGM Oracle

x	$\sigma(x)$
0	10100100
1	01010111
$\vdots$	$\vdots$
$p-1$	10010110



Theorem: Schnorr sigs are EUF-CMA *secure* in *preprocessing GGM* for $H: G \times M \to \mathbb{Z}_p$ where $H_k(g^r, m) \coloneqq g^r + m + k \ (mod\ p)$ if $|M|/p$ is negligible.

Uniformly random key $k \leftarrow \mathbb{Z}_p$ blocks generic non-uniform attacks

Theorem: Schnorr sigs are EUF-CMA *secure* in *preprocessing GGM* for $H: G \times M \to \mathbb{Z}_p$ where $H_k(g^r, m) \coloneqq g^r + m + k \ (mod\ p)$ if $|M|/p$ is negligible.

Uniformly random key $k \leftarrow \mathbb{Z}_p$ blocks generic non-uniform attacks

Conjecture: This scheme is secure if G is $\mathbb{Z}_p^\times$.
(not implied by generic analysis, but we haven't found any attacks)

Exercise. Break Schnorr sigs for short messages over $\mathbb{Z}_p^\times$ w/ this FS hash: $H_k(g^r, m) = g^r + m + k \ (mod\ p)$

Group: $\mathbb{Z}_p^\times$ with generator g

Message Space: $m \in M$ with $|M|/p$ negligible

Signing key: $sk \leftarrow \mathbb{Z}_p$

Verification key: $vk = (k, g^{sk})$ where $k \leftarrow \mathbb{Z}_p$

$\mathrm{Sign}(sk, m)$

- Sample $r \leftarrow \mathbb{Z}_p$. Let $z = r + (g^r + m + k) \cdot sk \ (mod\ p)$
- Output (g^r, z)

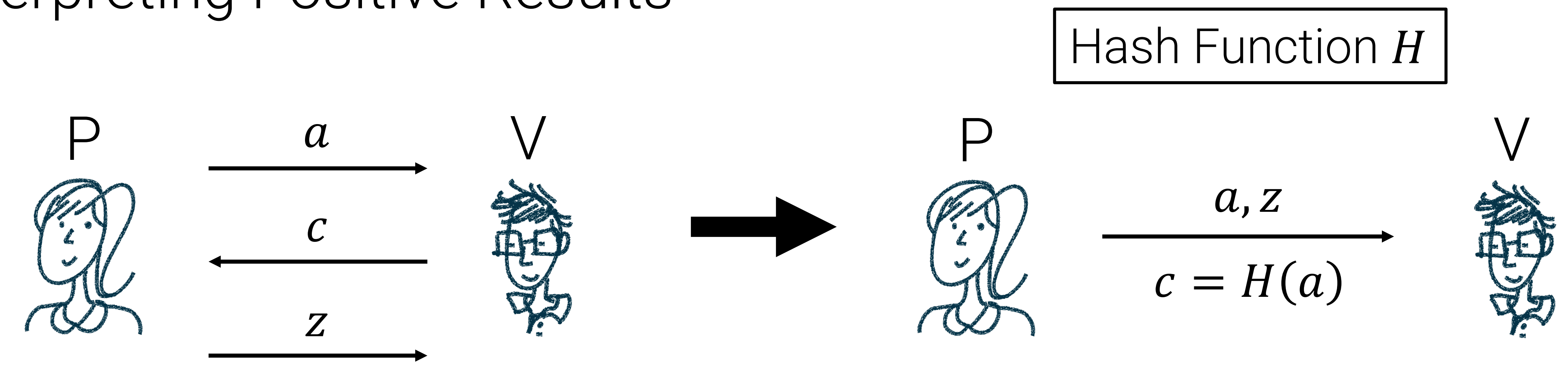
$\mathrm{Ver}(vk, m, (g^r, z))$

- Accept if $g^z = g^r \cdot (g^{sk})^{g^r + m + k} \ (mod\ p)$.

Warning: Our security analysis does not imply security in $\mathbb{Z}_p^\times$!

But unclear (to us) how to break EUF-CMA security.

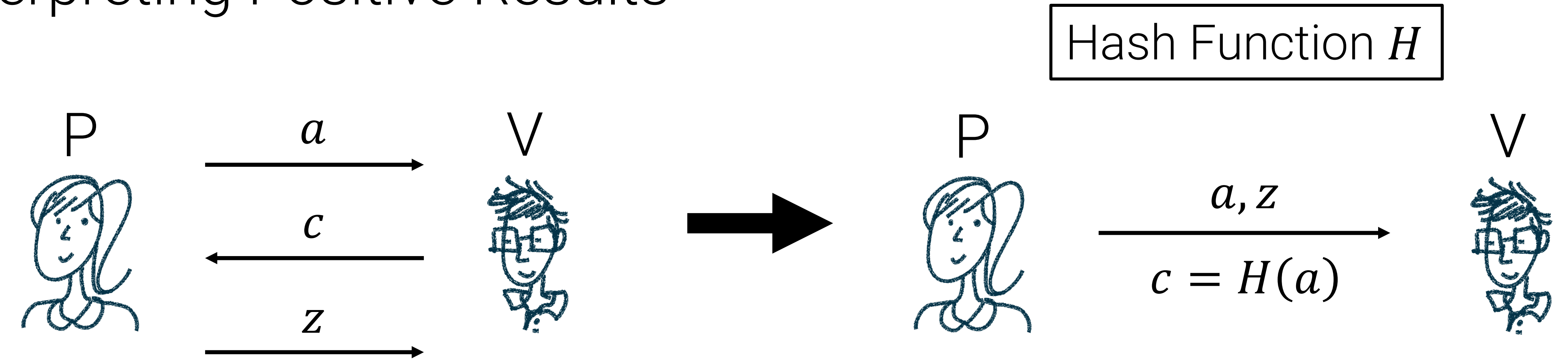
Interpreting Positive Results



Interactive Protocol Π Non-Interactive Protocol $FS_H[\Pi]$

In positive results, $FS_H[\Pi]$ soundness uses cryptography *already present* in Π.

Interpreting Positive Results



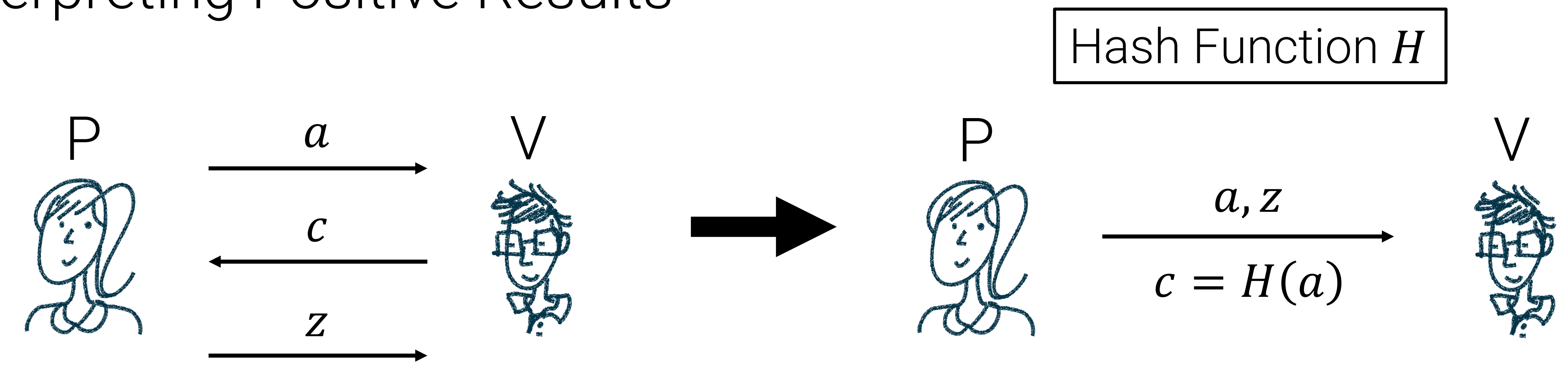
Interactive Protocol Π

Non-Interactive Protocol $FS_H[\Pi]$

In positive results, $FS_H[\Pi]$ soundness uses cryptography *already present* in Π.

- Π_{Sch} uses cryptographic groups; $FS_H[\Pi_{Sch}]$ soundness relies on generic hardness of the group.

Interpreting Positive Results



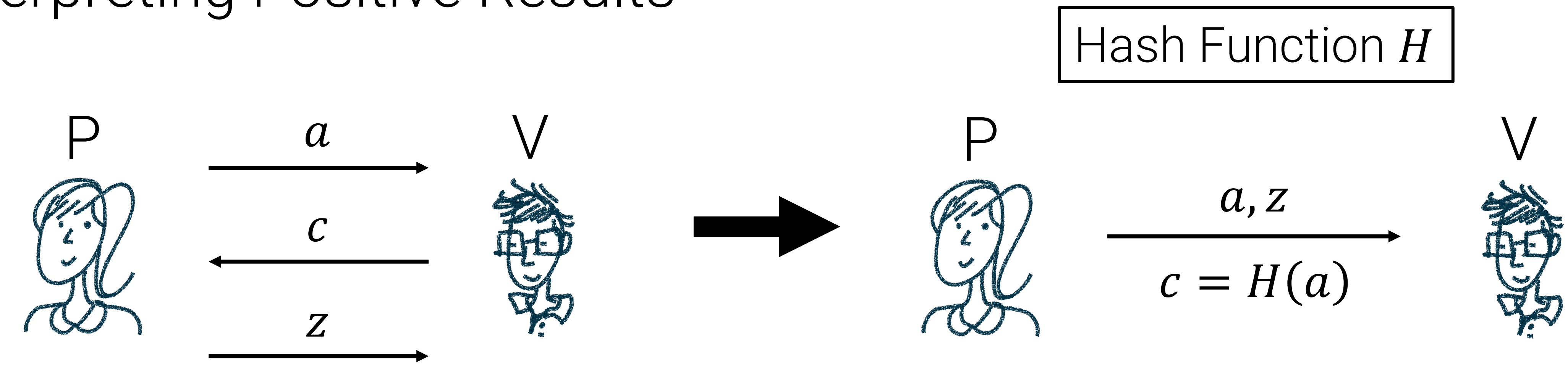
Interactive Protocol Π

Non-Interactive Protocol $FS_H[\Pi]$

In positive results, $FS_H[\Pi]$ soundness uses cryptography *already present* in Π.

- Π_{Sch} uses cryptographic groups; $FS_H[\Pi_{Sch}]$ soundness relies on generic hardness of the group.
- Π_{Lyu} uses lattices; $FS_H[\Pi_{Lyu}]$ soundness relies on SIS.

Interpreting Positive Results

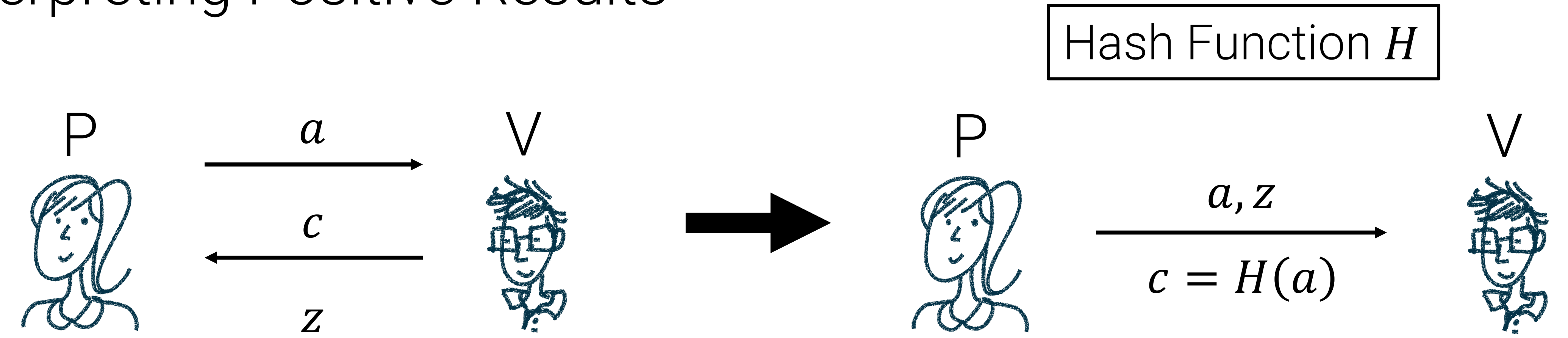


Interactive Protocol Π

Non-Interactive Protocol $FS_H[\Pi]$

This suggests a strategy: identify a security property related to Π that results in sound $FS_H[\Pi]$ for a simple/non-cryptographic H.

Interpreting Positive Results



Interactive Protocol Π

Non-Interactive Protocol $FS_H[\Pi]$

This suggests a strategy: identify a security property related to Π that results in sound $FS_H[\Pi]$ for a simple/non-cryptographic H.

When is it possible to do this?

Outline

- Positive Results for Lyubashevsky
- Positive Results for Schnorr
- Negative Results

Theorem. Let Π be a 3-message HVZK argument (or proof) with poly-size challenge space and let Π^t denote Π repeated t times in parallel.

Soundness of $FS_H[\Pi^t]$ **requires** H to satisfy a cryptographic security property.

Theorem. Let Π be a 3-message HVZK argument (or proof) with poly-size challenge space and let Π^t denote Π repeated t times in parallel.

Soundness of $FS_H[\Pi^t]$ requires H to satisfy a cryptographic security property.

- Blum's Hamiltonicity protocol
- GMW86 3-Coloring protocol
- 1-bit challenge Schnorr
- 1-bit challenge Lyubashevsky

Theorem. Let Π be a 3-message HVZK argument (or proof) with poly-size challenge space and let Π^t denote Π repeated t times in parallel.

Soundness of $FS_H[\Pi^t]$ requires H to satisfy a cryptographic security property.

- Blum's Hamiltonicity protocol
- GMW86 3-Coloring protocol
- 1-bit challenge Schnorr
- 1-bit challenge Lyubashevsky

Takeaway: FS without a cryptographic hash function requires large challenge space that is ***not*** obtained via parallel repetition of a protocol with a small challenge space.

Theorem. Let Π be a 3-message HVZK argument (or proof) with poly-size challenge space and let Π^t denote Π repeated t times in parallel.

Soundness of $FS_H[\Pi^t]$ requires H to satisfy a cryptographic security property.

- Blum's Hamiltonicity protocol

Recall: First message in Blum is a cryptographic commitment.

Even if the commitment is "ideal", the Fiat-Shamir hash function must be cryptographic.

Review: ZK Proof of Hamiltonicity [Blum86]

$P(G, \sigma)$

$V(G)$

$G' = \pi(G)$ for $\pi \leftarrow S_n$

Compute $a = Com(G')$

$\xrightarrow{\quad a \quad}$

$\xleftarrow{\quad b \in \{0,1\} \quad}$

$b = 0$: open G' and send π.

$b = 1$: open $\pi \circ \sigma$.

$\xrightarrow{\quad z \quad}$

Accept if:

$b = 0$: openings valid and $G' = \pi(G)$.

$b = 1$: openings valid and edge openings are 1.

Review: ZK Proof of Hamiltonicity [Blum86]

Random Oracle $\mathcal{O}$

$P(G, \sigma)$

Random Oracle $\mathcal{O}$

$V(G)$

$G' = \pi(G)$ for $\pi \leftarrow S_n$

Compute $a = Com(G')$
w/ $Com(x; r) = \mathcal{O}(x, r)$

$\xrightarrow{\quad a \quad}$

$\xleftarrow{\quad b \in \{0,1\} \quad}$

$b = 0$: open G' and send π.

$b = 1$: open $\pi \circ \sigma$.

$\xrightarrow{\quad z \quad}$

Accept if:

$b = 0$: openings valid and $G' = \pi(G)$.

$b = 1$: openings valid and edge openings are 1.

Review: ZK Proof of Hamiltonicity [Blum86]

Random Oracle $\mathcal{O}$

$P(G, \sigma)$

+ parallel repetition

Random Oracle $\mathcal{O}$

$V(G)$

$G' = \pi(G)$ for $\pi \leftarrow S_n$

Compute $a = Com(G')$
w/ $Com(x; r) = \mathcal{O}(x, r)$

$\xrightarrow{a_1, \dots, a_t}$

$\xleftarrow{b_1, \dots, b_t}$

$b = 0$: open G' and send π.

$b = 1$: open $\pi \circ \sigma$.

$\xrightarrow{z_1, \dots, z_t}$

Accept if:

$b = 0$: openings valid and $G' = \pi(G)$.

$b = 1$: openings valid and edge openings are 1.

Review: ZK Proof of Hamiltonicity [Blum86]

Random Oracle $\mathcal{O}$

$P(G, \sigma)$

FS hash H

What is a **bad** choice of H?

Random Oracle $\mathcal{O}$

$V(G)$

$G' = \pi(G)$ for $\pi \leftarrow S_n$

Compute $a = Com(G')$
w/ $Com(x; r) = \mathcal{O}(x, r)$

$a_1, \ldots, a_t$
$z_1, \ldots, z_t$ $\longrightarrow$

$b = 0$: open G' and send π.

$b = 1$: open $\pi \circ \sigma$.

$b_1, \ldots, b_t$
$= H(a_1, \ldots, a_t)$

Accept if:

$b = 0$: openings valid and $G' = \pi(G)$.

$b = 1$: openings valid and edge openings are 1.

Attacking an Insecure H: Suppose $H(a_1, \dots, a_t) = f(a_1), \dots, f(a_t)$.

Attacking an Insecure H: Suppose $H(a_1, \dots, a_t) = f(a_1), \dots, f(a_t)$.

Idea: Break each instance one-by-one.

Attacking an Insecure H: Suppose $H(a_1, \ldots, a_t) = f(a_1), \ldots, f(a_t)$.

Idea: Break each instance one-by-one.

- $b_{1,1} \leftarrow \{0,1\}$

Attacking an Insecure H: Suppose $H(a_1, \dots, a_t) = f(a_1), \dots, f(a_t)$.

Idea: Break each instance one-by-one.

- $b_{1,1} \leftarrow \{0,1\}$
- Compute $a_{1,1}$ that can open on challenge $b_{1,1}$.

Attacking an Insecure H: Suppose $H(a_1, \ldots, a_t) = f(a_1), \ldots, f(a_t)$.

Idea: Break each instance one-by-one.

- $b_{1,1} \leftarrow \{0,1\}$
- Compute $a_{1,1}$ that can open on challenge $b_{1,1}$.
- If $f(a_{1,1}) = b_{1,1}$ move on.

✓

Attacking an Insecure H: Suppose $H(a_1, \ldots, a_t) = f(a_1), \ldots, f(a_t)$.

Idea: Break each instance one-by-one.

- $b_{1,1} \leftarrow \{0,1\}$
- Compute $a_{1,1}$ that can open on challenge $b_{1,1}$.
- If $f\left(a_{1,1}\right) = b_{1,1}$ move on.

✓

- $b_{2,1} \leftarrow \{0,1\}$
- Compute $a_{2,1}$
- $f\left(a_{2,1}\right) \neq b_{2,1}$

X

Attacking an Insecure H: Suppose $H(a_1, \ldots, a_t) = f(a_1), \ldots, f(a_t)$.

Idea: Break each instance one-by-one.

- $b_{1,1} \leftarrow \{0,1\}$
- Compute $a_{1,1}$ that can open on challenge $b_{1,1}$.
- If $f(a_{1,1}) = b_{1,1}$ move on.

- $b_{2,1} \leftarrow \{0,1\}$
- Compute $a_{2,1}$
- $f(a_{2,1}) \neq b_{2,1}$

- $b_{2,2} \leftarrow \{0,1\}$
- Compute $a_{2,2}$
- $f(a_{2,2}) = b_{2,2}$

Attacking an Insecure H: Suppose $H(a_1, \dots, a_t) = f(a_1), \dots, f(a_t)$.

Idea: Break each instance one-by-one.

- $b_{1,1} \leftarrow \{0,1\}$
- Compute $a_{1,1}$ that can open on challenge $b_{1,1}$.
- If $f(a_{1,1}) = b_{1,1}$ move on.

✓

- $b_{2,1} \leftarrow \{0,1\}$
- Compute $a_{2,1}$
- $f(a_{2,1}) \neq b_{2,1}$

X

- $b_{2,2} \leftarrow \{0,1\}$
- Compute $a_{2,2}$
- $f(a_{2,2}) = b_{2,2}$

✓

- $b_{3,1} \leftarrow \{0,1\}$
- Compute $a_{3,1}$
- $f(a_{3,1}) = b_{3,1}$

✓

…

- $b_{t,1} \leftarrow \{0,1\}$
- Compute $a_{t,1}$
- $f(a_{t,1}) \neq b_{t,1}$

X

- $b_{t,2} \leftarrow \{0,1\}$
- Compute $a_{t,2}$
- $f(a_{t,2}) = b_{t,2}$

✓

Each $i = 1, \dots, t$ takes 2 tries in expectation

Attacking an Insecure H: Suppose $H(a_1, \dots, a_t) = f(a_1), \dots, f(a_t)$.

Idea: Break each instance one-by-one.

- $b_{1,1} \leftarrow \{0,1\}$
- Compute $a_{1,1}$ that can open on challenge $b_{1,1}$.
- If $f\left(\boxed{a_{1,1}}\right) = b_{1,1}$ move on.

- $b_{2,1} \leftarrow \{0,1\}$
- Compute $a_{2,1}$
- $f\left(a_{2,1}\right) \neq b_{2,1}$

- $b_{2,2} \leftarrow \{0,1\}$
- Compute $a_{2,2}$
- $f\left(\boxed{a_{2,2}}\right) = b_{2,2}$

- $b_{3,1} \leftarrow \{0,1\}$
- Compute $a_{3,1}$
- $f\left(\boxed{a_{3,1}}\right) = b_{3,1}$

…

- $b_{t,1} \leftarrow \{0,1\}$
- Compute $a_{t,1}$
- $f\left(a_{t,1}\right) \neq b_{t,1}$

- $b_{t,2} \leftarrow \{0,1\}$
- Compute $a_{t,2}$
- $f\left(\boxed{a_{t,2}}\right) = b_{t,2}$

Each $i = 1, \dots, t$ takes 2 tries in expectation.

Attacking an Insecure H: Suppose $H(a_1, \ldots, a_t) = f(a_1), \ldots, f(a_t)$.

$b_{1,1} \leftarrow \{0,1\}$ Commitment $a_{1,1}$	$b_{2,1} \leftarrow \{0,1\}$ Commitment $a_{2,1}$	$\cdots$	$b_{t,1} \leftarrow \{0,1\}$ Commitment $a_{t,1}$
$b_{1,2} \leftarrow \{0,1\}$ Commitment $a_{1,2}$	$b_{2,2} \leftarrow \{0,1\}$ Commitment $a_{2,2}$		
	$b_{2,3} \leftarrow \{0,1\}$ Commitment $a_{2,3}$		

Modify attack to always perform k tries for each i.

Attacking an Insecure H: Suppose $H(a_1, \dots, a_t) = f(a_1), \dots, f(a_t)$.

t columns

k rows

$b_{1,1} \leftarrow \{0,1\}$ Commitment $a_{1,1}$	$b_{2,1} \leftarrow \{0,1\}$ Commitment $a_{2,1}$	$\cdots$	$b_{t,1} \leftarrow \{0,1\}$ Commitment $a_{t,1}$
$b_{1,2} \leftarrow \{0,1\}$ Commitment $a_{1,2}$	$b_{2,2} \leftarrow \{0,1\}$ Commitment $a_{2,2}$	$\cdots$	$b_{t,2} \leftarrow \{0,1\}$ Commitment $a_{t,2}$
$\vdots$	$\vdots$		$\vdots$
$b_{1,k} \leftarrow \{0,1\}$ Commitment $a_{1,k}$	$b_{2,k} \leftarrow \{0,1\}$ Commitment $a_{2,k}$	$\cdots$	$b_{t,k} \leftarrow \{0,1\}$ Commitment $a_{t,k}$

Modify attack to always perform k tries for each i.

Attacking an Insecure H: Suppose $H(a_1, \ldots, a_t) = f(a_1), \ldots, f(a_t)$.

t columns

k rows

$b_{1,1} \leftarrow \{0,1\}$ Commitment $a_{1,1}$	$b_{2,1} \leftarrow \{0,1\}$ Commitment $a_{2,1}$	$\cdots$	$b_{t,1} \leftarrow \{0,1\}$ Commitment $a_{t,1}$
$b_{1,2} \leftarrow \{0,1\}$ Commitment $a_{1,2}$	$b_{2,2} \leftarrow \{0,1\}$ Commitment $a_{2,2}$	$\cdots$	$b_{t,2} \leftarrow \{0,1\}$ Commitment $a_{t,2}$
$\vdots$	$\vdots$		$\vdots$
$b_{1,k} \leftarrow \{0,1\}$ Commitment $a_{1,k}$	$b_{2,k} \leftarrow \{0,1\}$ Commitment $a_{2,k}$	$\cdots$	$b_{t,k} \leftarrow \{0,1\}$ Commitment $a_{t,k}$

If $k = \omega(\log t)$, w.h.p. can choose block j_i in each column i s.t.

$$H\left(a_{1,j_1}, \ldots, a_{t,j_t}\right) = b_{1,j_1}, \ldots, b_{t,j_t}$$

This generalizes to any H!

t columns

$\omega(t)$ rows

$b_{1,1} \leftarrow \{0,1\}$ Commitment $a_{1,1}$	$b_{2,1} \leftarrow \{0,1\}$ Commitment $a_{2,1}$	$\cdots$	$b_{t,1} \leftarrow \{0,1\}$ Commitment $a_{t,1}$
$b_{1,2} \leftarrow \{0,1\}$ Commitment $a_{1,2}$	$b_{2,2} \leftarrow \{0,1\}$ Commitment $a_{2,2}$	$\cdots$	$b_{t,2} \leftarrow \{0,1\}$ Commitment $a_{t,2}$
$\vdots$	$\vdots$		$\vdots$
$b_{1,k} \leftarrow \{0,1\}$ Commitment $a_{1,k}$	$b_{2,k} \leftarrow \{0,1\}$ Commitment $a_{2,k}$	$\cdots$	$b_{t,k} \leftarrow \{0,1\}$ Commitment $a_{t,k}$

This generalizes to any H!

t columns

$\omega(t)$ rows

$b_{1,1} \leftarrow \{0,1\}$ Commitment $a_{1,1}$	$b_{2,1} \leftarrow \{0,1\}$ Commitment $a_{2,1}$	$\cdots$	$b_{t,1} \leftarrow \{0,1\}$ Commitment $a_{t,1}$
$b_{1,2} \leftarrow \{0,1\}$ Commitment $a_{1,2}$	$b_{2,2} \leftarrow \{0,1\}$ Commitment $a_{2,2}$	$\cdots$	$b_{t,2} \leftarrow \{0,1\}$ Commitment $a_{t,2}$
$\vdots$	$\vdots$		$\vdots$
$b_{1,k} \leftarrow \{0,1\}$ Commitment $a_{1,k}$	$b_{2,k} \leftarrow \{0,1\}$ Commitment $a_{2,k}$	$\cdots$	$b_{t,k} \leftarrow \{0,1\}$ Commitment $a_{t,k}$

Lemma. For $\omega(t)$ rows, exists block j_i in each column i s.t.

$$H\left(a_{1,j_1}, \ldots, a_{t,j_t}\right) = b_{1,j_1}, \ldots, b_{t,j_t}$$

General attack on $FS_H[\Pi_{Blum}]$:

1) Sample grid of random bit/commitment pairs.

General attack on $FS_H[\Pi_{Blum}]$:

1) Sample grid of random bit/commitment pairs.

t columns

$\omega(t)$ rows

$b_{1,1} \leftarrow \{0,1\}$ Commitment $a_{1,1}$	$b_{2,1} \leftarrow \{0,1\}$ Commitment $a_{2,1}$	$\cdots$	$b_{t,1} \leftarrow \{0,1\}$ Commitment $a_{t,1}$
$b_{1,2} \leftarrow \{0,1\}$ Commitment $a_{1,2}$	$b_{2,2} \leftarrow \{0,1\}$ Commitment $a_{2,2}$	$\cdots$	$b_{t,2} \leftarrow \{0,1\}$ Commitment $a_{t,2}$
$\vdots$	$\vdots$		$\vdots$
$b_{1,k} \leftarrow \{0,1\}$ Commitment $a_{1,k}$	$b_{2,k} \leftarrow \{0,1\}$ Commitment $a_{2,k}$	$\cdots$	$b_{t,k} \leftarrow \{0,1\}$ Commitment $a_{t,k}$

General attack on $FS_H[\Pi_{Blum}]$:

1) Sample grid of random bit/commitment pairs.
2) Choose block j_i in column i s.t. $H(a_{1,j_1}, \ldots, a_{t,j_t}) = b_{1,j_1}, \ldots, b_{t,j_t}$.

t columns

$\omega(t)$ rows

$b_{1,1} \leftarrow \{0,1\}$ Commitment $a_{1,1}$	$b_{2,1} \leftarrow \{0,1\}$ Commitment $a_{2,1}$	$\cdots$	$b_{t,1} \leftarrow \{0,1\}$ Commitment $a_{t,1}$
$b_{1,2} \leftarrow \{0,1\}$ Commitment $a_{1,2}$	$b_{2,2} \leftarrow \{0,1\}$ Commitment $a_{2,2}$	$\cdots$	$b_{t,2} \leftarrow \{0,1\}$ Commitment $a_{t,2}$
$\vdots$	$\vdots$		$\vdots$
$b_{1,k} \leftarrow \{0,1\}$ Commitment $a_{1,k}$	$b_{2,k} \leftarrow \{0,1\}$ Commitment $a_{2,k}$	$\cdots$	$b_{t,k} \leftarrow \{0,1\}$ Commitment $a_{t,k}$

General attack on $FS_H[\Pi_{Blum}]$:

1) Sample grid of random bit/commitment pairs.
2) Choose block j_i in column i s.t. $H(a_{1,j_1}, \dots, a_{t,j_t}) = b_{1,j_1}, \dots, b_{t,j_t}$.
3) Open commitments.

t columns

$\omega(t)$ rows

$b_{1,1} \leftarrow \{0,1\}$ Commitment $a_{1,1}$	$b_{2,1} \leftarrow \{0,1\}$ Commitment $a_{2,1}$	$\cdots$	$b_{t,1} \leftarrow \{0,1\}$ Commitment $a_{t,1}$
$b_{1,2} \leftarrow \{0,1\}$ Commitment $a_{1,2}$	$b_{2,2} \leftarrow \{0,1\}$ Commitment $a_{2,2}$	$\cdots$	$b_{t,2} \leftarrow \{0,1\}$ Commitment $a_{t,2}$
$\vdots$	$\vdots$		$\vdots$
$b_{1,k} \leftarrow \{0,1\}$ Commitment $a_{1,k}$	$b_{2,k} \leftarrow \{0,1\}$ Commitment $a_{2,k}$	$\cdots$	$b_{t,k} \leftarrow \{0,1\}$ Commitment $a_{t,k}$

General attack on $FS_H[\Pi_{Blum}]$:

1) Sample grid of random bit/commitment pairs.
2) Choose block j_i in column i s.t. $H(a_{1,j_1}, \dots, a_{t,j_t}) = b_{1,j_1}, \dots, b_{t,j_t}$.
3) Open commitments.

Soundness of $FS_H[\Pi_{Blum}]$ requires computational hardness of (2).

H must be "mix-and-match resistant."

(requirement extends to any parallel repetition of 3-message HVZK argument with poly-size challenge space.)

Thanks!

eprint: 2020/915
slides: cs.princeton.edu/~fermim/

drawings by Eysa Lee