Improved Straight-Line Extraction in the Random Oracle Model

with Applications to Signature Aggregation

Yashvanth Kondi
abhi shelat

AARHUS UNIVERSITY
Northeastern University

This Work

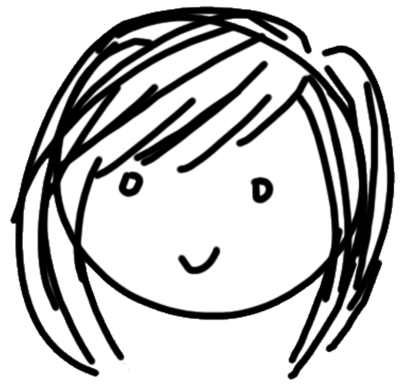
- We explore two dimensions of Fischlin's NIZKPoK compiler:
  - **Applicability**:
    Only proven for Sigma protocols with 'quasi-unique responses'
    (doesn't include logical OR, Pedersen commitment PoK, etc.)
    Folklore: "works anyway"

    1a) Contrary to folklore: attack on Witness Indistinguishability
    1b) Simple randomization fixes the problem
  - **Computation cost**:
    Usually the bottleneck — can we improve on it?

    2) Lower bound: Fischlin05 is optimal up to a small constant
    3) Application-specific optimization: 200× for EdDSA aggregation

Recap: Σ Protocol for Relation R

[Damgård 02]

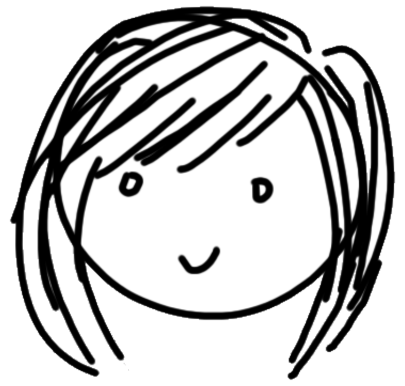


$P(X, w)$

$V(X)$

Recap: Σ Protocol for Relation R

[Damgård 02]



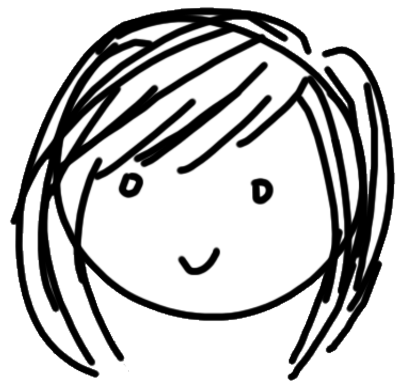
$P(X, w)$

$V(X)$

Commitment $\xrightarrow{a}$

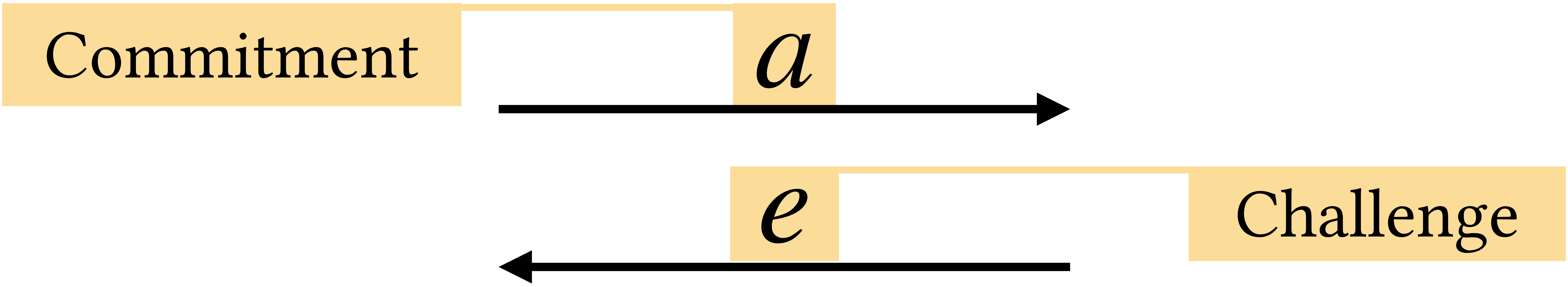
Recap: Σ Protocol for Relation R

[Damgård 02]



$P(X, w)$

$V(X)$

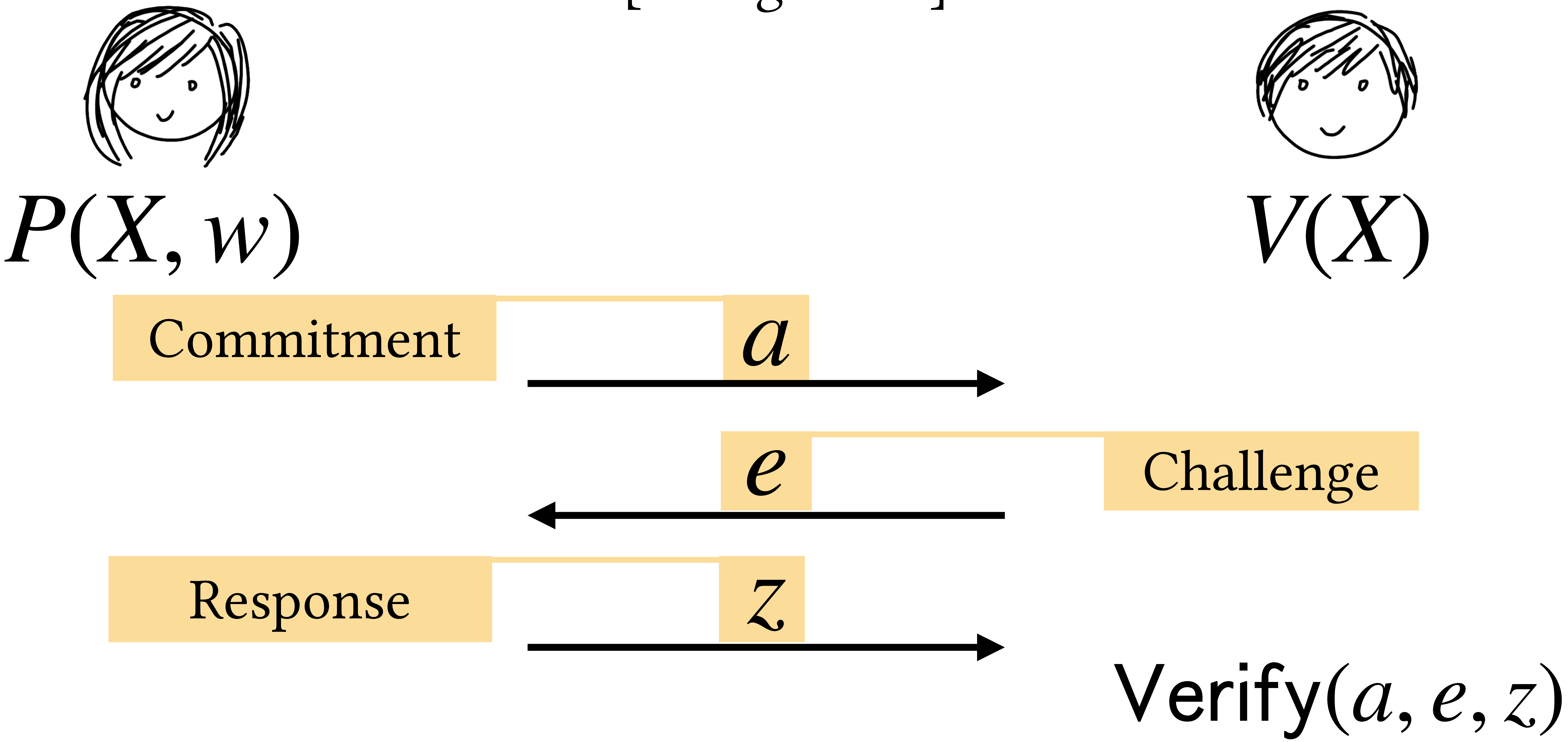


Commitment a →

← e Challenge

Recap: Σ Protocol for Relation R

[Damgård 02]



$P(X, w)$

$V(X)$

Commitment a $\longrightarrow$

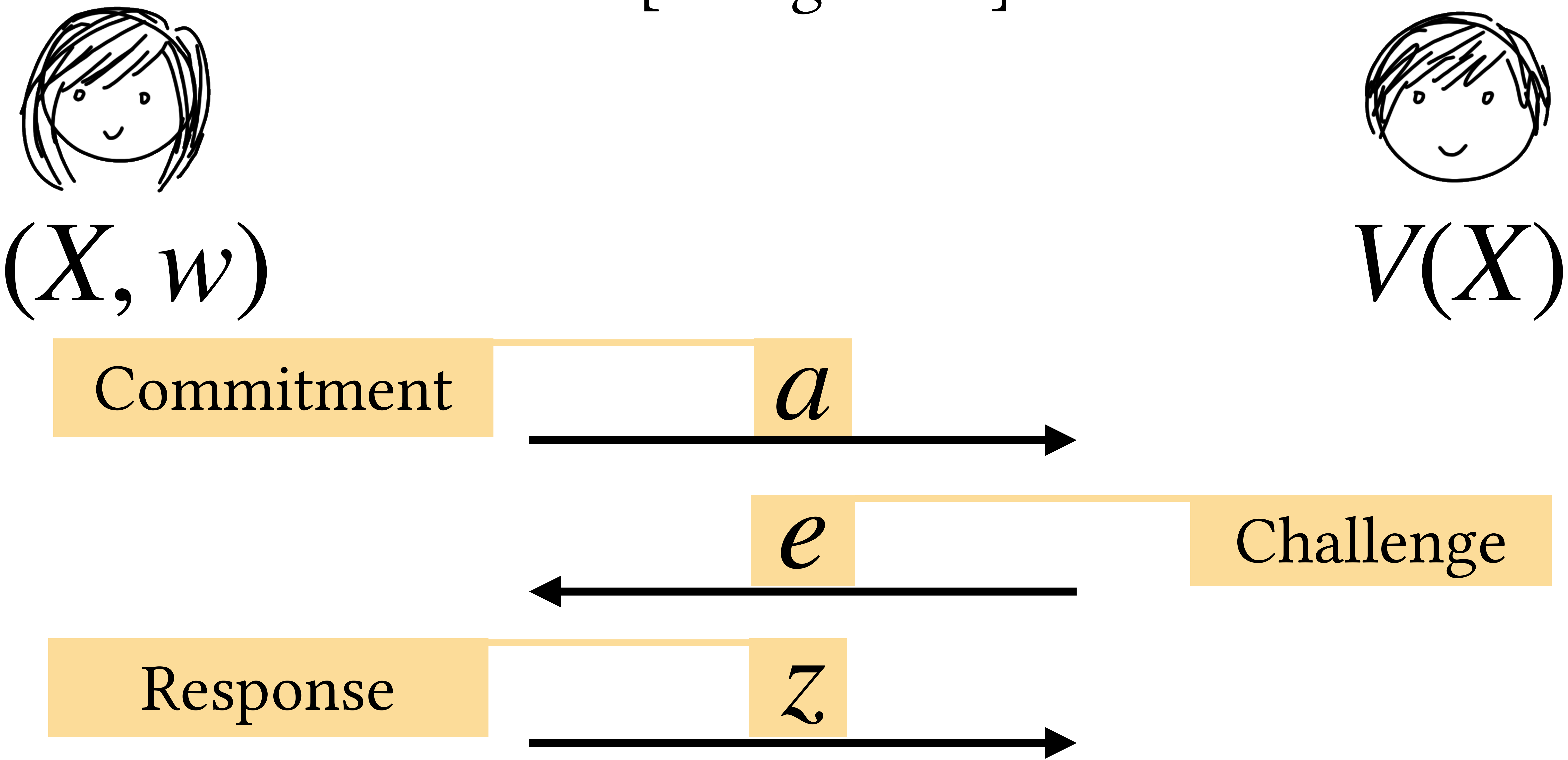
$\longleftarrow$ e Challenge

Response z $\longrightarrow$

$\mathsf{Verify}(a, e, z)$

Recap: Σ Protocol for Relation R

[Damgård 02]



$P(X, w)$

$V(X)$

Commitment a

e Challenge

Response z

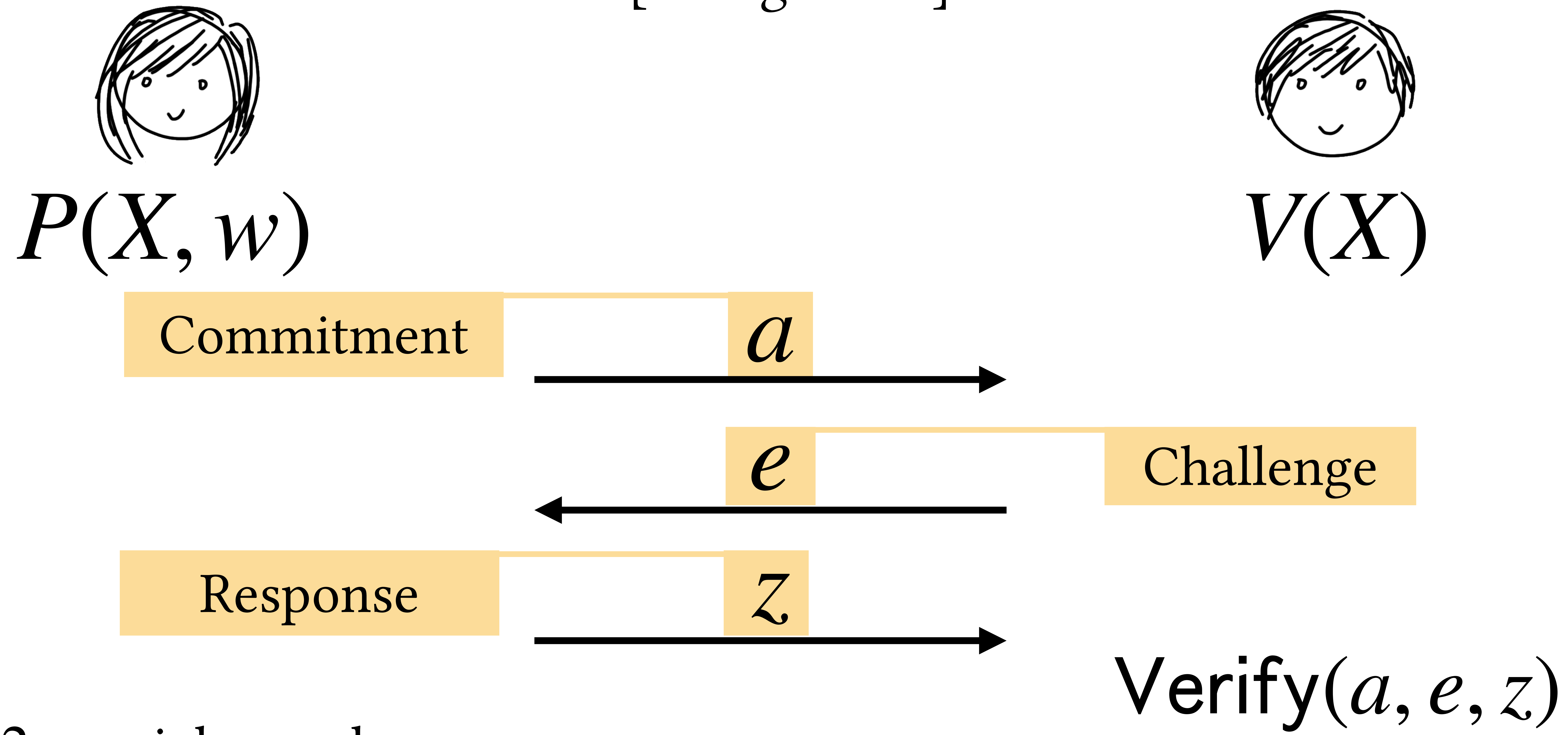
$\mathsf{Verify}(a, e, z)$

2-special soundness:

$$w \leftarrow \mathsf{Ext}(X, a, (e_1, z_1), (e_2, z_2)) \text{ such that } R(X, w) = 1$$

Recap: Σ Protocol for Relation R

[Damgård 02]



$P(X, w)$

$V(X)$

Commitment a →

e ← Challenge

Response z →

$\mathsf{Verify}(a, e, z)$

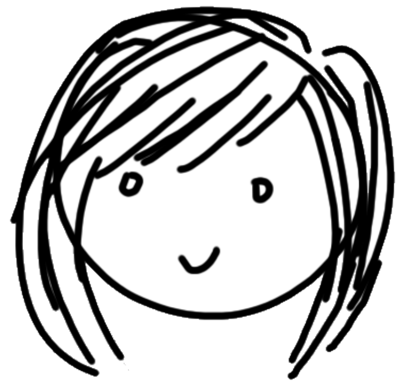
2-special soundness:

$w \leftarrow \mathsf{Ext}(X, a, (e_1, z_1), (e_2, z_2))$ such that $R(X, w) = 1$

Fixed commitment

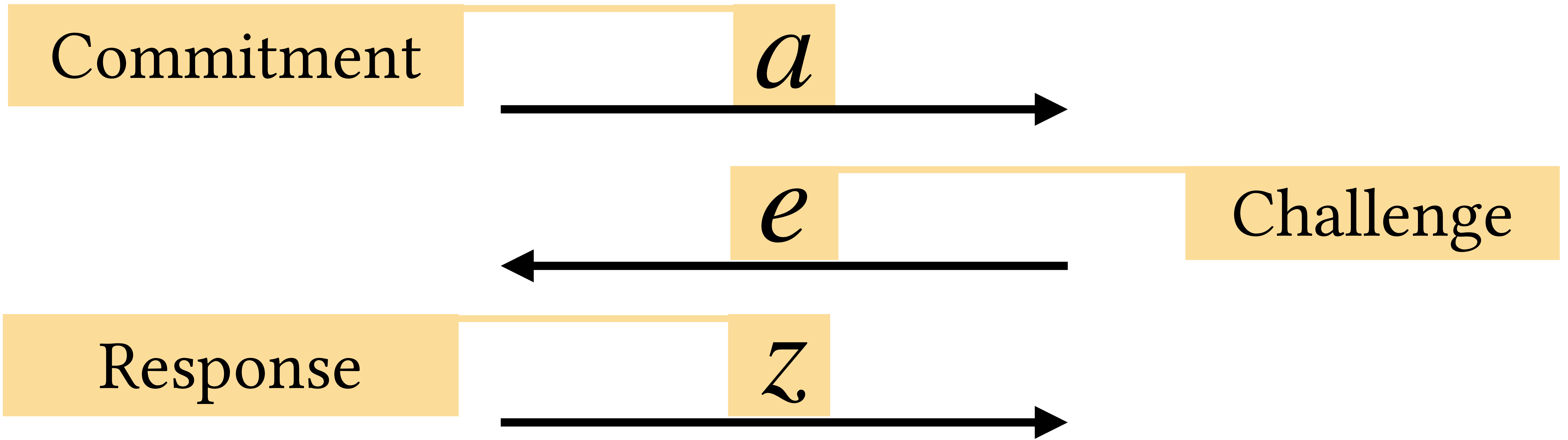
Recap: Σ Protocol for Relation R

[Damgård 02]



$P(X, w)$

$V(X)$



Commitment a

$$\xrightarrow{\quad a \quad}$$

e Challenge

$$\xleftarrow{\quad e \quad}$$

Response z

$$\xrightarrow{\quad z \quad}$$

$\mathsf{Verify}(a, e, z)$

2-special soundness:

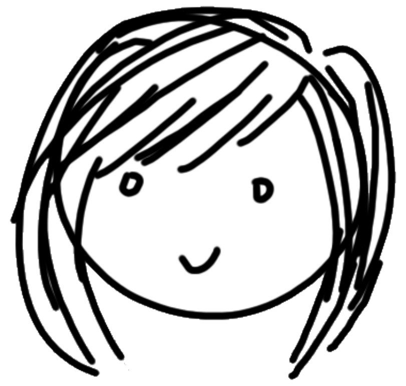
Varying (ch, resp) pairs

$w \leftarrow \mathsf{Ext}(X, a, (e_1, z_1), (e_2, z_2))$ such that $R(X, w) = 1$

Fixed commitment

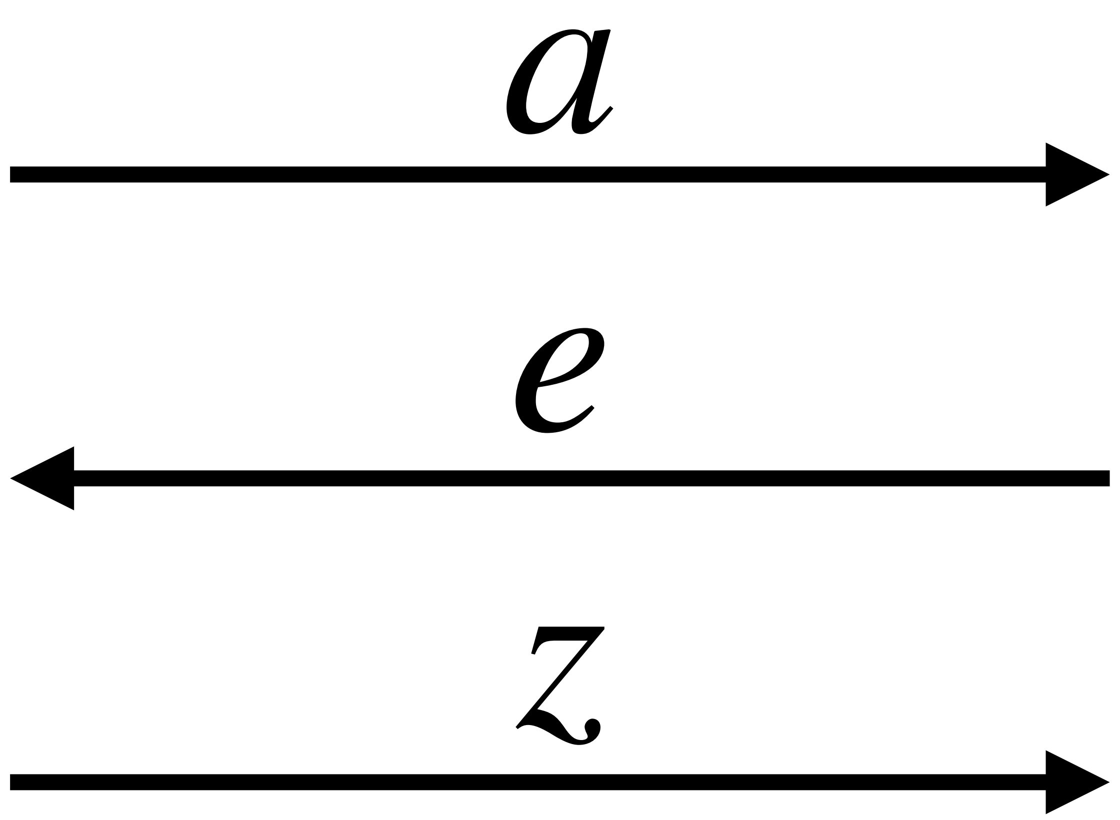
The Fiat-Shamir Transform

- [Fiat Shamir 87] provides a simple method to compile any public-coin protocol to a non-interactive proof, given a suitably chosen hash function



$P(X, w)$

$V(X)$



a

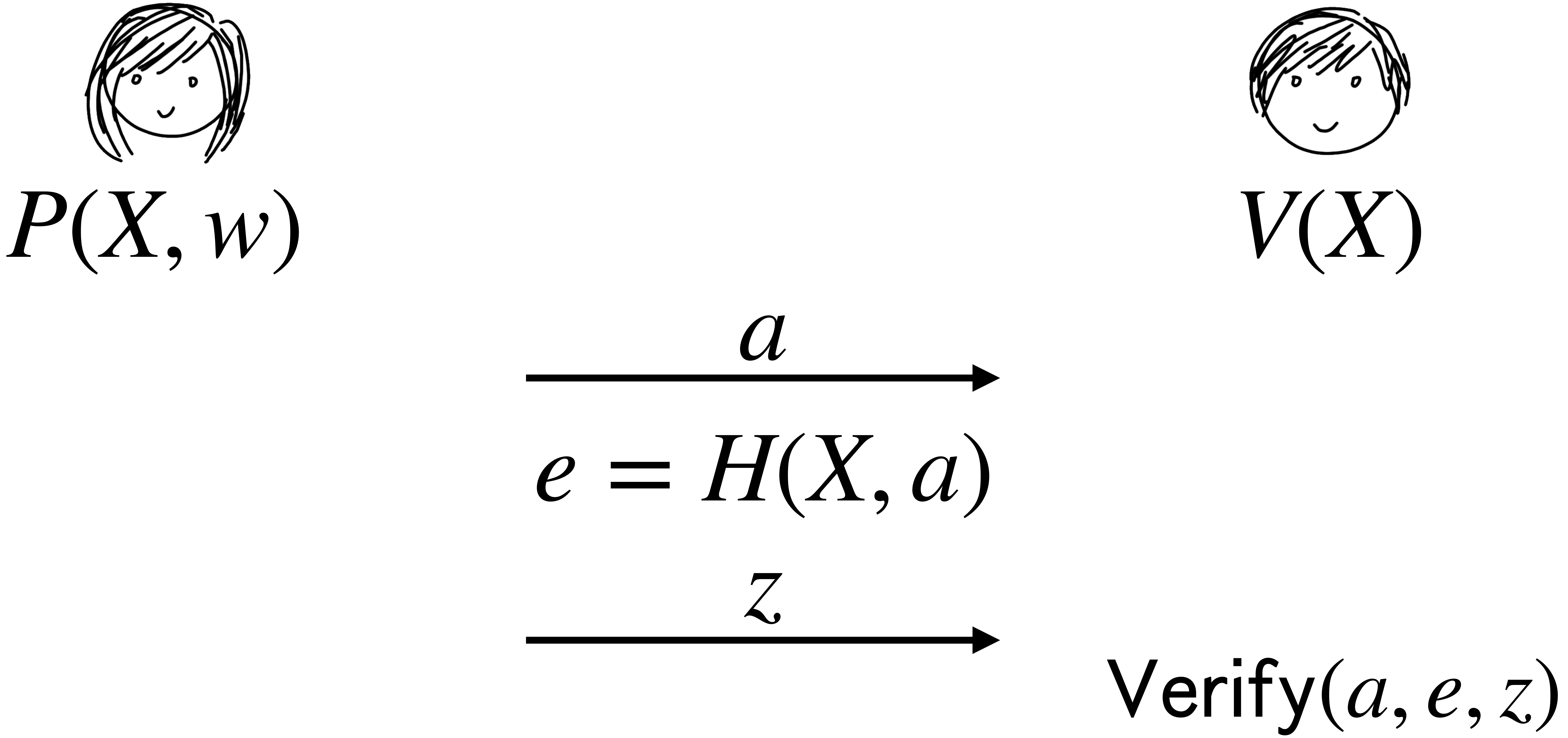
e

z

$\mathsf{Verify}(a, e, z)$

The Fiat-Shamir Transform

- [Fiat Shamir 87] provides a simple method to compile any public-coin protocol to a non-interactive proof, given a suitably chosen hash function



$P(X, w)$

$V(X)$

$$\xrightarrow{a}$$

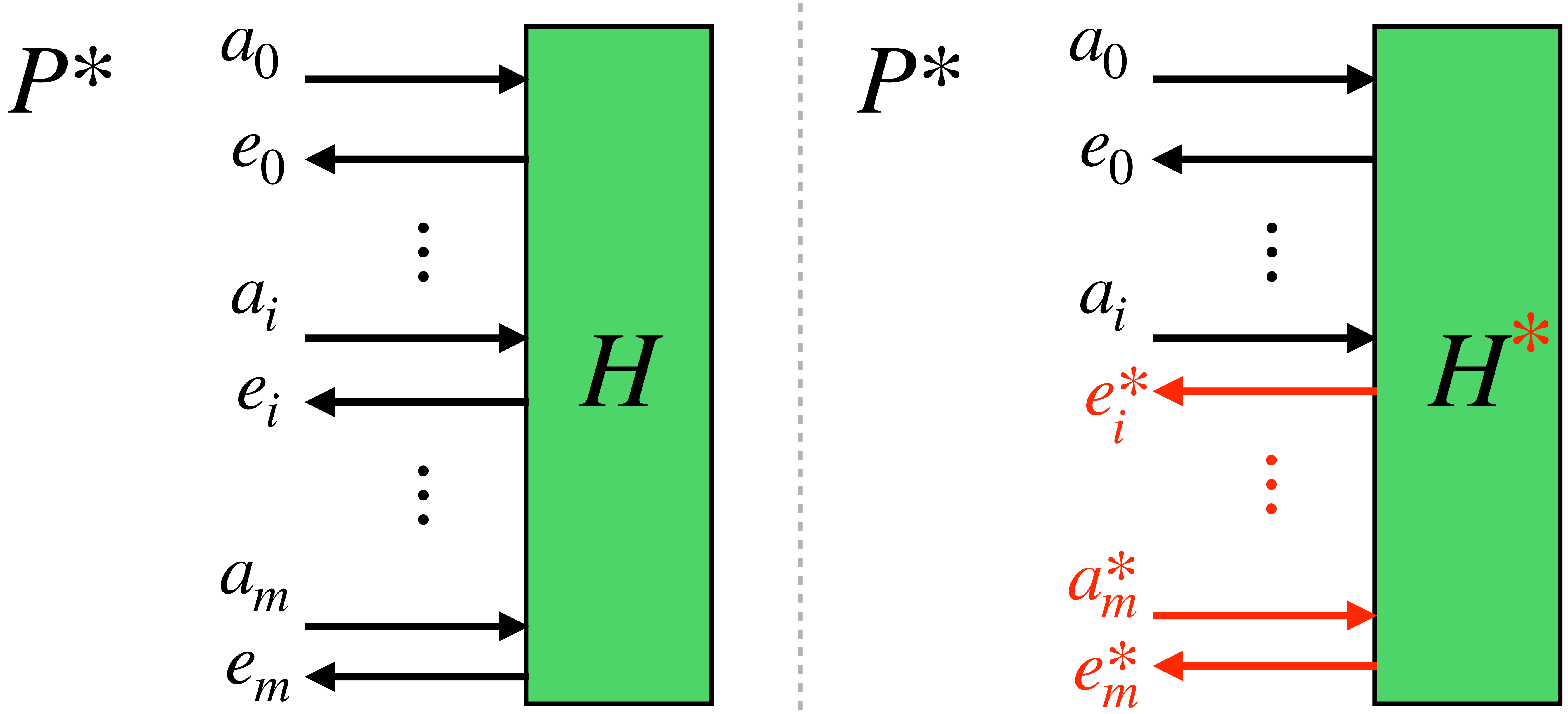
$$e = H(X, a)$$

$$\xrightarrow{z}$$

$\mathsf{Verify}(a, e, z)$

Fiat-Shamir: Security

- "Forking" extraction strategy in Random Oracle Model [Pointcheval Stern 96]:



$$\mathsf{Ext}\begin{pmatrix}(a_i, e_i) & (a_i, e_i) \\ \multicolumn{2}{c}{z_i, z_i^*}\end{pmatrix}$$

Outputs witness w

Output (a_i, e_i, z_i)

Output (a_i, e_i^*, z_i^*)

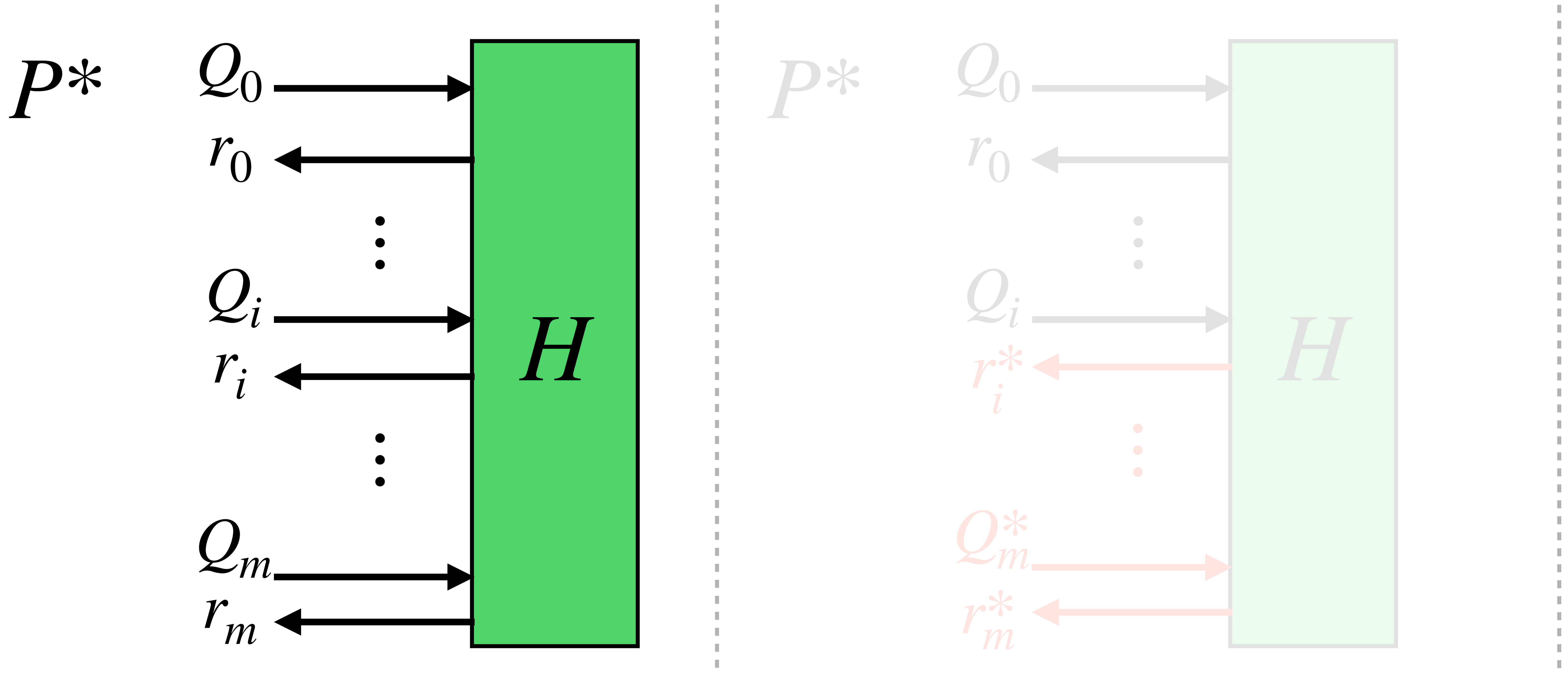
Probability of success: p p $\approx p^2$

Fiat-Shamir Compilation

- Advantages:
  - Simple to describe/implement
  - Very efficient; proving, verification cost exactly the same as input Σ-protocol
- Downsides:
  - Forking strategy does not compose; unclear how to prove concurrent security
  - Quadratic security loss

Straight-line Extraction

- Formalized by [Pass 03] in the Random Oracle Model:



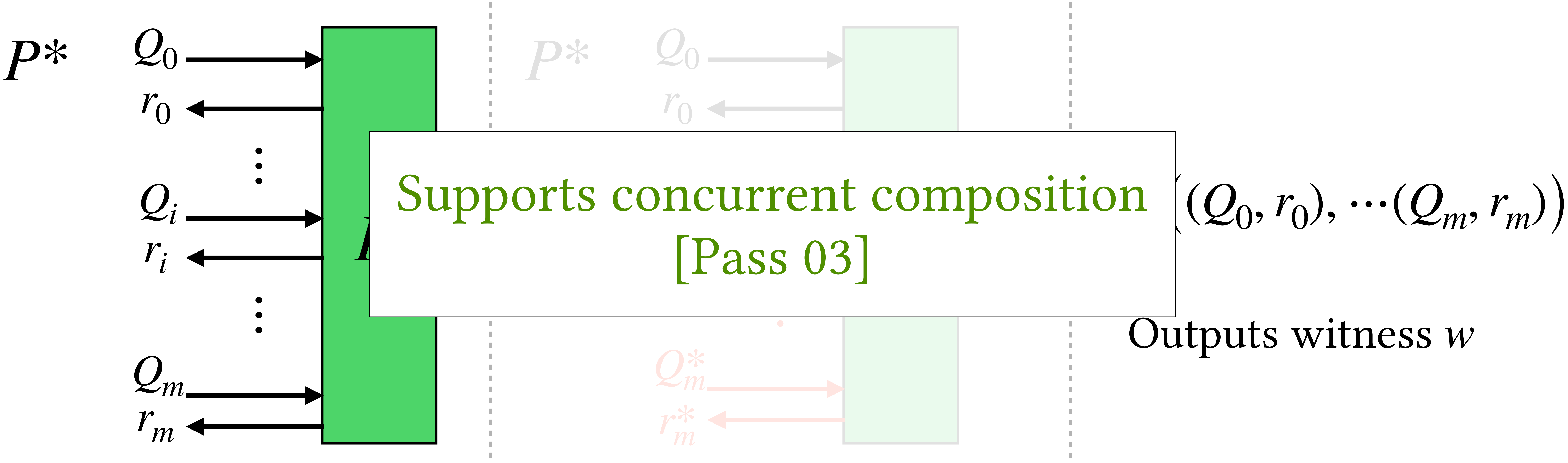
$\mathsf{Ext}\left((Q_0, r_0), \cdots (Q_m, r_m)\right)$

Outputs witness w

Probability of success: p p $\approx p$

Straight-line Extraction

- Formalized by [Pass 03] in the Random Oracle Model:



Supports concurrent composition [Pass 03]

Probability of success: p p $\approx p$

Straight-line Extraction

- Formalized by [Pass 03] in the Random Oracle Model:

Supports concurrent composition
[Pass 03]

Gave simple cut-and-choose construction

Fischlin’s Transformation

- [Fischlin 05] gave a straight-line extractable compiler that avoids cut-and-choose logistics through a clever “proof of work” type idea

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

$$\xrightarrow[a, e, z]{}$$

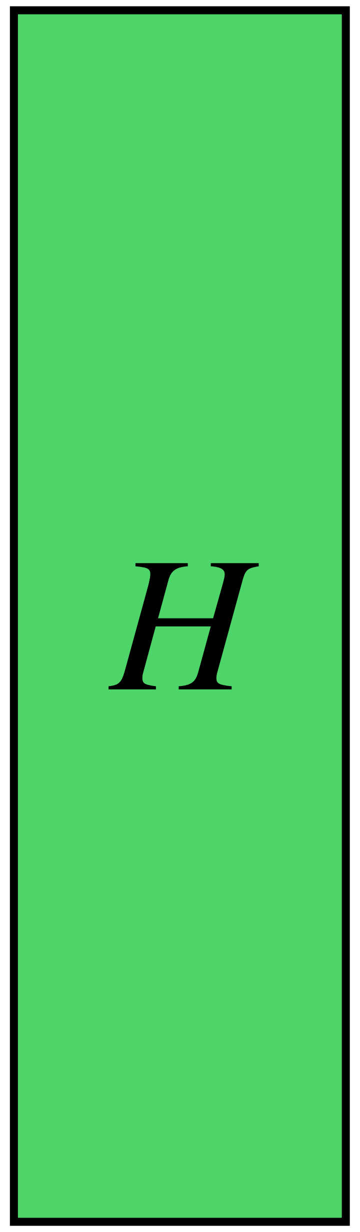
$$H(a, e, z) \overset{?}{=} 0$$

$$\mathsf{Verify}(a, e, z)$$

Fischlin's Transformation

- Let $\boxed{H}: \{0,1\}^* \mapsto \{0,1\}^\ell$ be a random oracle

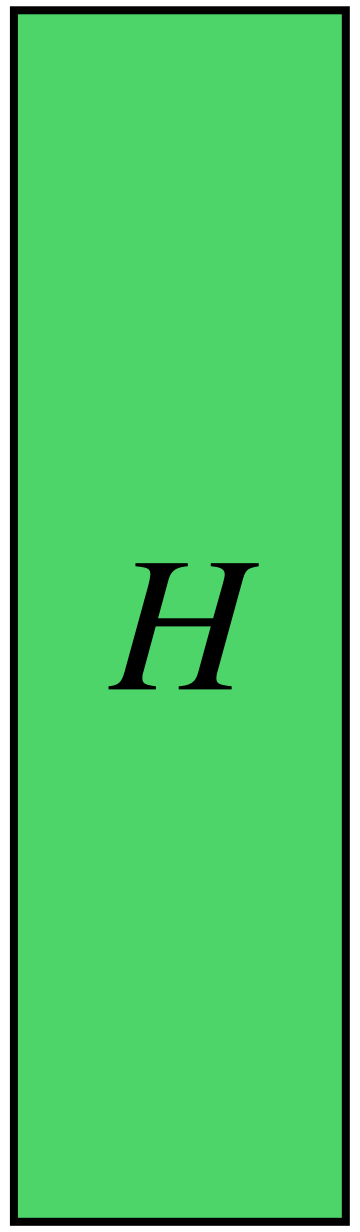
$P(X, w)$:



Fischlin's Transformation

- Let $\boxed{H}: \{0,1\}^* \mapsto \{0,1\}^{\ell}$ be a random oracle

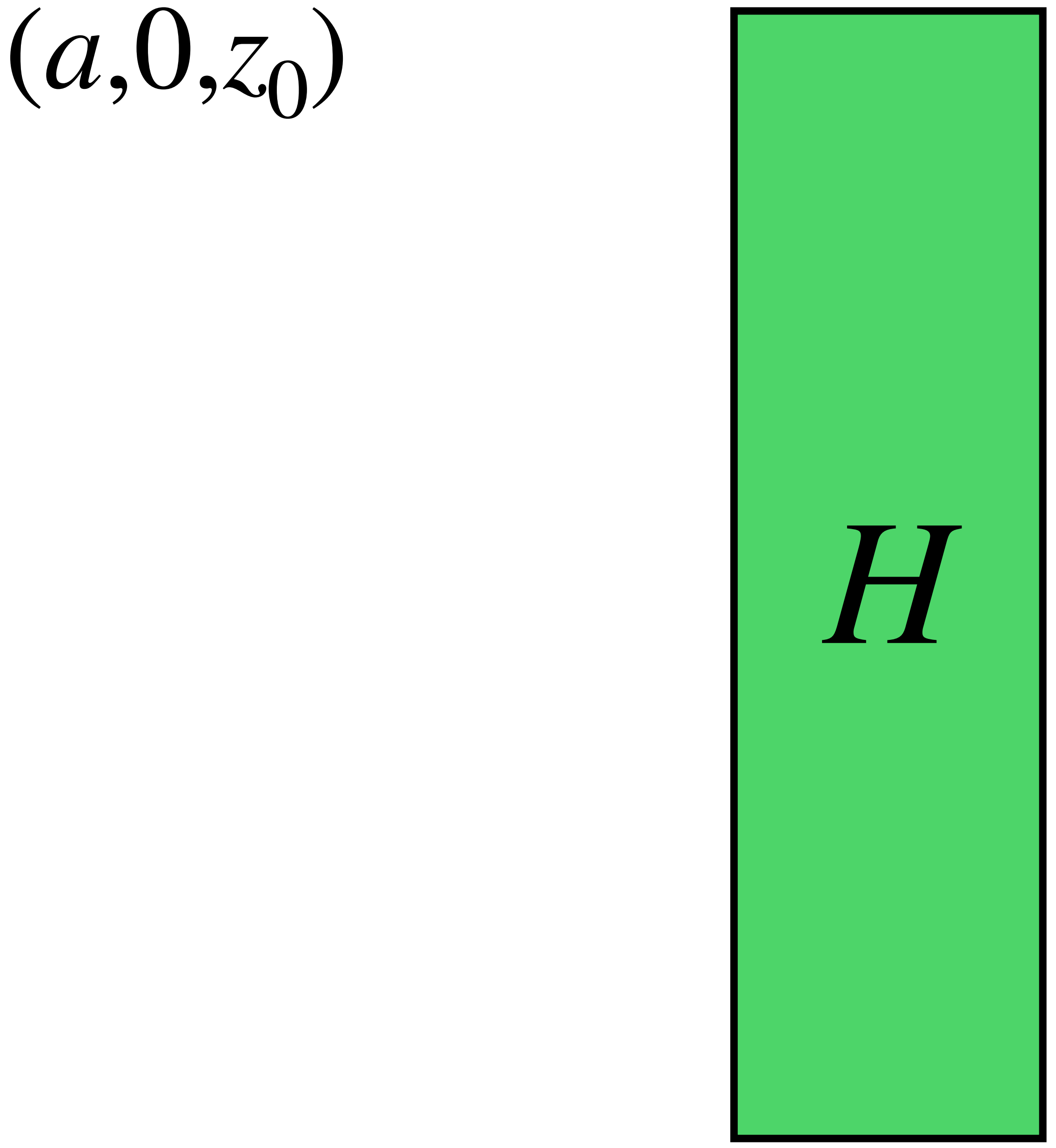
$P(X, w)$: Sample Σ-protocol first message 'a'



Fischlin's Transformation

- Let $\boxed{H}$: $\{0,1\}^* \mapsto \{0,1\}^\ell$ be a random oracle

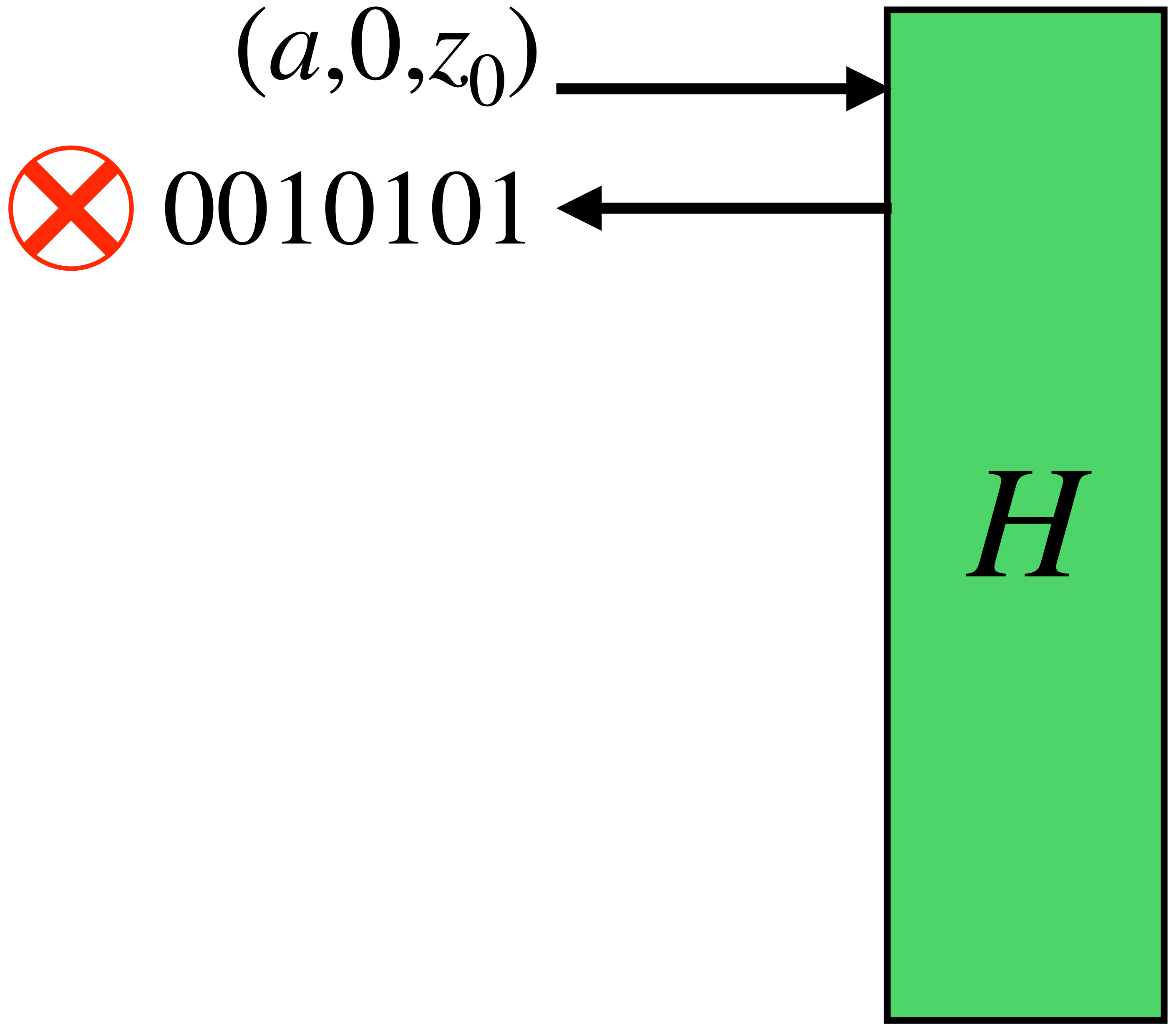
$P(X, w)$: Sample Σ-protocol first message 'a'


$(a,0,z_0)$

Fischlin's Transformation

- Let $\boxed{H}: \{0,1\}^* \mapsto \{0,1\}^\ell$ be a random oracle

$P(X, w)$: Sample Σ-protocol first message 'a'



$(a,0,z_0)$

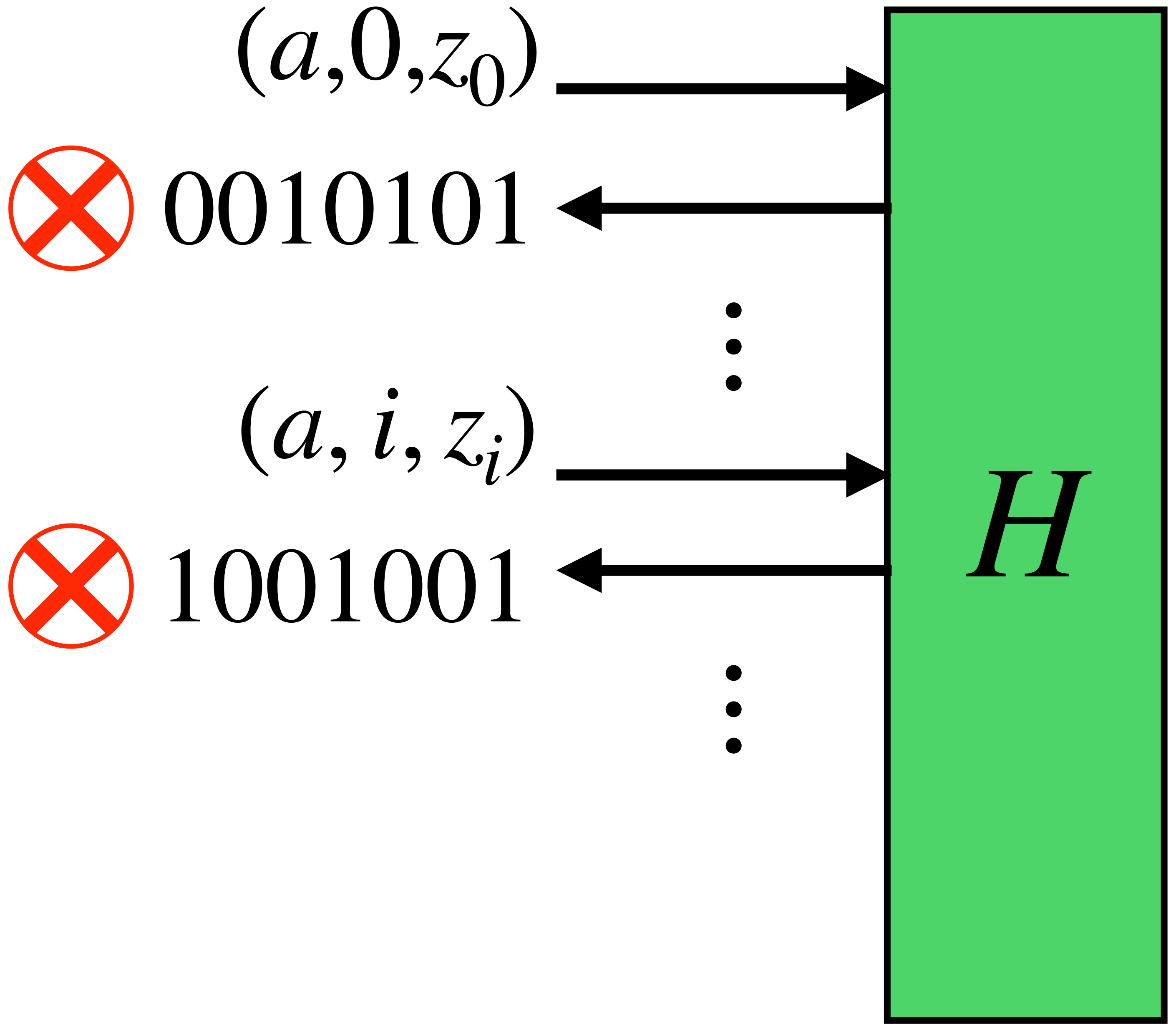
❌ 0010101

H

Fischlin's Transformation

- Let $\boxed{H}$: $\{0,1\}^* \mapsto \{0,1\}^\ell$ be a random oracle

$P(X, w)$: Sample Σ-protocol first message 'a'



$(a,0,z_0)$

⊗ 0010101

$\vdots$

(a, i, z_i)

⊗ 1001001

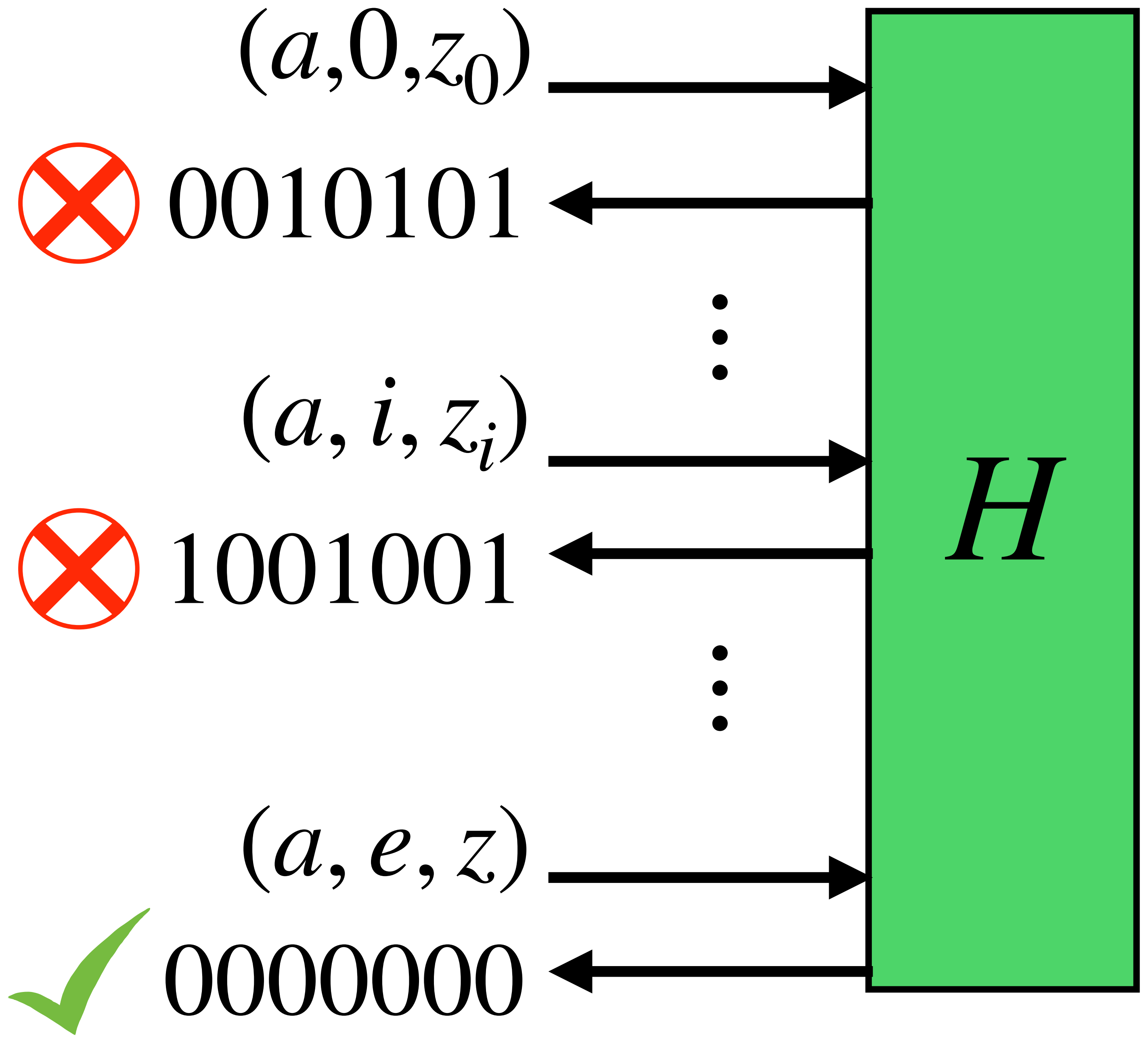
$\vdots$

H

Fischlin's Transformation

- Let $\boxed{H}: \{0,1\}^* \mapsto \{0,1\}^{\ell}$ be a random oracle

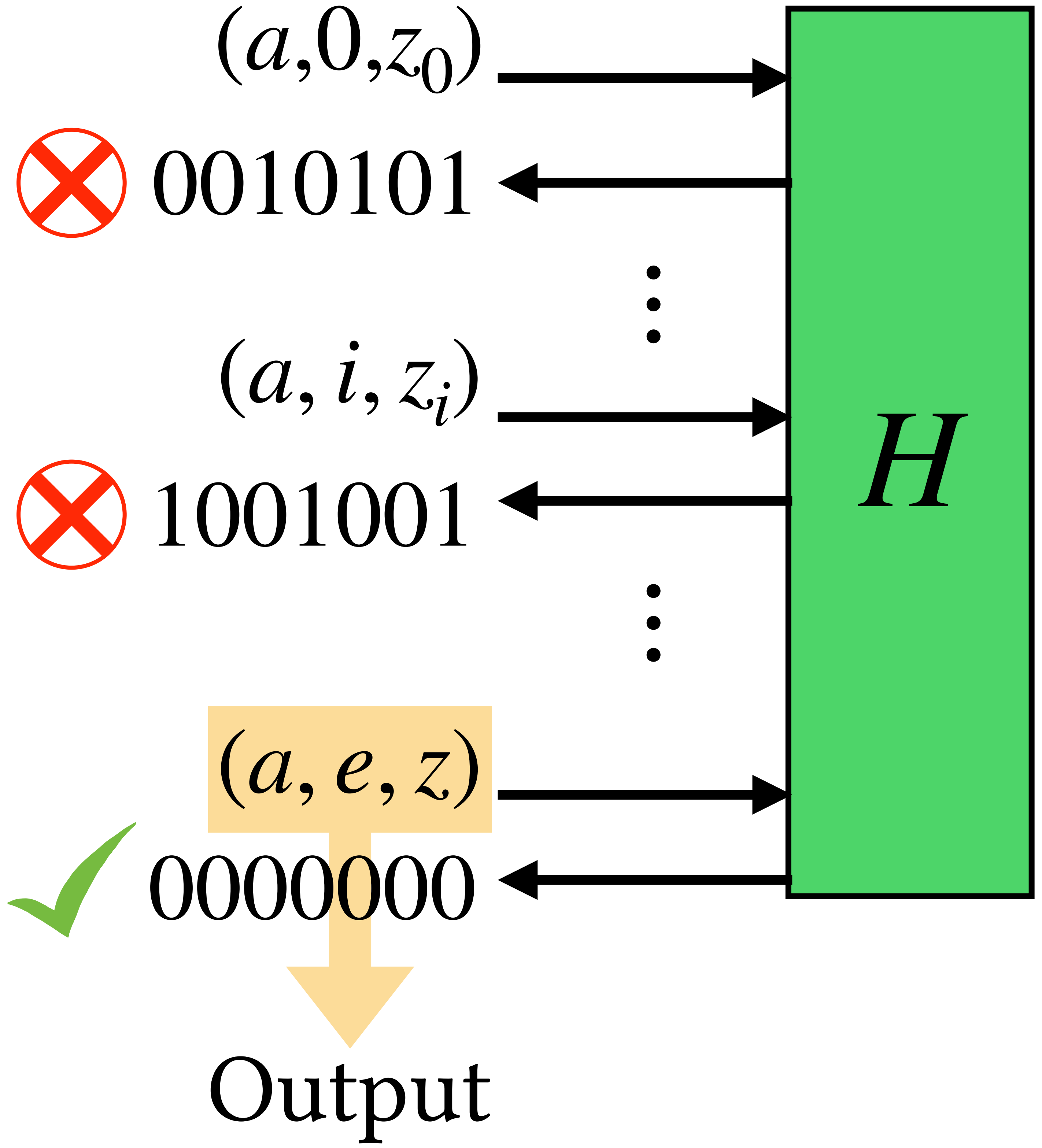
$P(X, w)$: Sample Σ-protocol first message 'a'



Fischlin's Transformation

- Let $\boxed{H}: \{0,1\}^* \mapsto \{0,1\}^{\ell}$ be a random oracle

$P(X, w)$: Sample Σ-protocol first message 'a'



$(a, 0, z_0)$

✗ 0010101

⋮

(a, i, z_i)

✗ 1001001

⋮

(a, e, z)

✓ 0000000

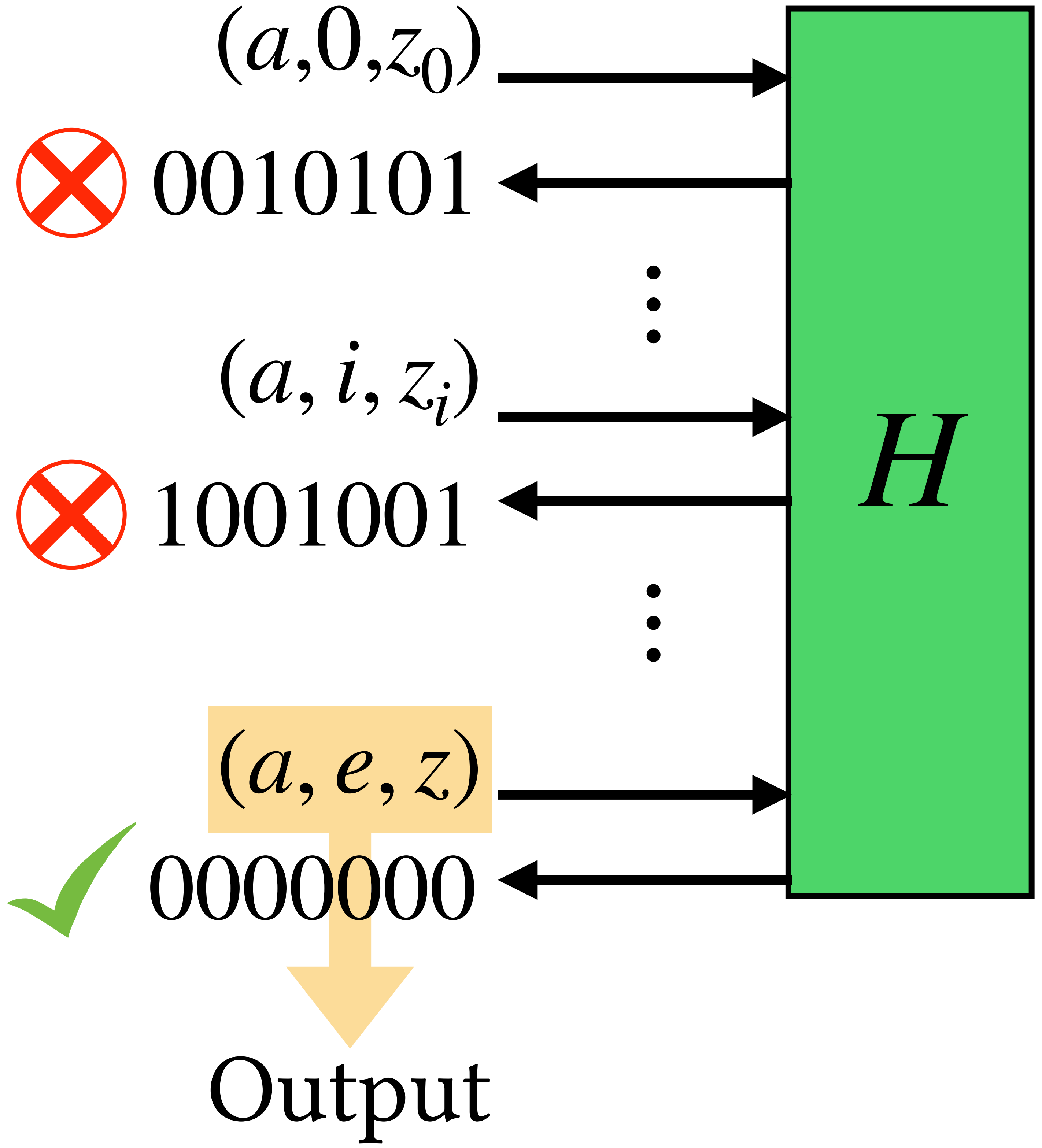
Output

H

Fischlin's Transformation

- Let $\boxed{H}$: $\{0,1\}^* \mapsto \{0,1\}^\ell$ be a random oracle

$P(X, w)$: Sample Σ-protocol first message 'a'



$(a,0,z_0)$

✗ 0010101

⋮

(a, i, z_i)

✗ 1001001

⋮

(a, e, z)

✓ 0000000

Output

H

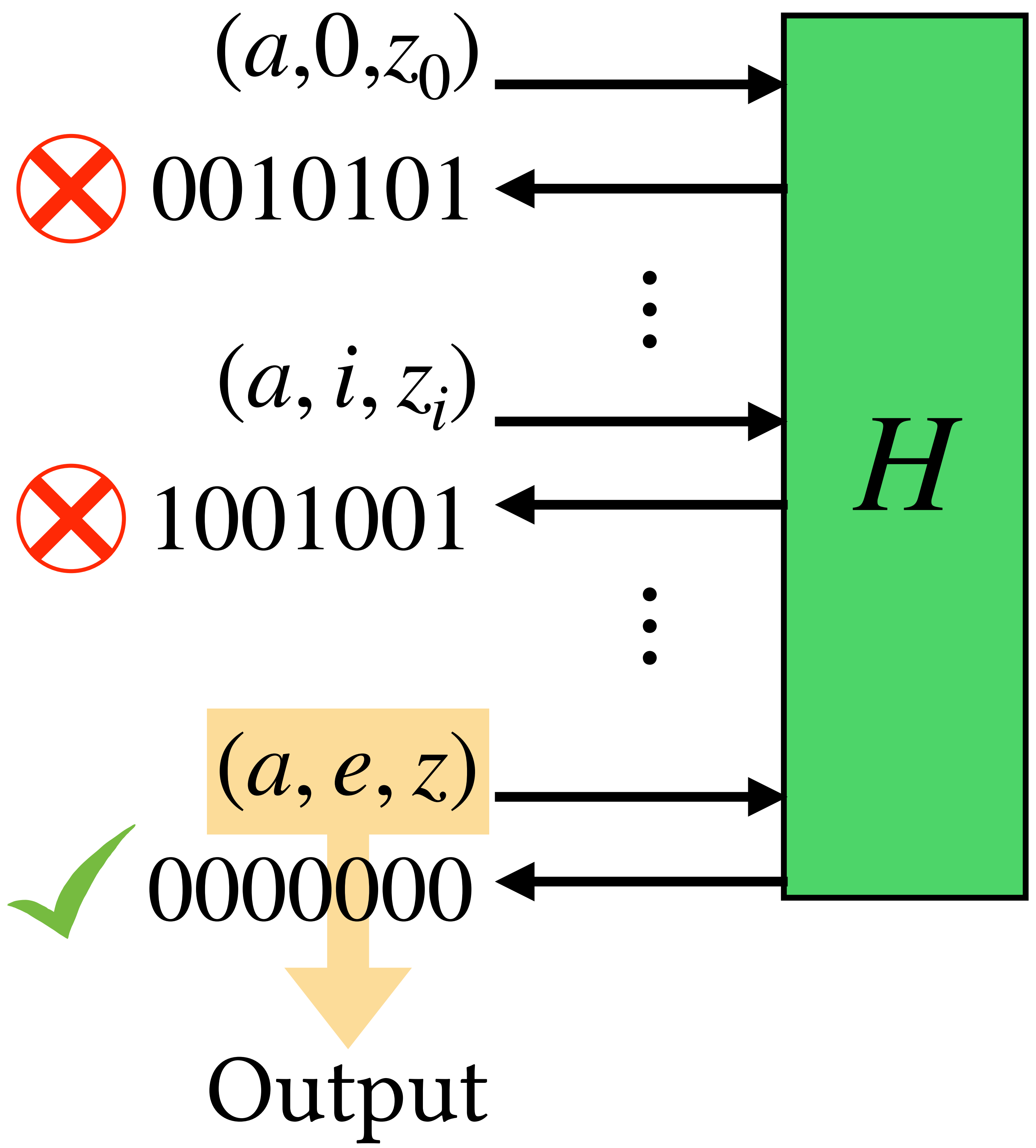
Soundness: Except with Pr=$2^{-\ell}$, P is forced to query more than one accepting transcript to $\boxed{H}$

Completeness: P terminates in poly time when ℓ is small, i.e. $O(\log \kappa)$

Fischlin's Transformation

- Let $\boxed{H}: \{0,1\}^* \mapsto \{0,1\}^\ell$ be a random oracle

$P(X, w)$: Sample Σ-protocol first message 'a'



$(a,0,z_0)$

☒ 0010101

⋮

(a, i, z_i)

☒ 1001001

⋮

(a, e, z)

✓ 0000000

Output

Soundness: Except with Pr=$2^{-\ell}$, P is forced to query more than one accepting transcript to $\boxed{H}$

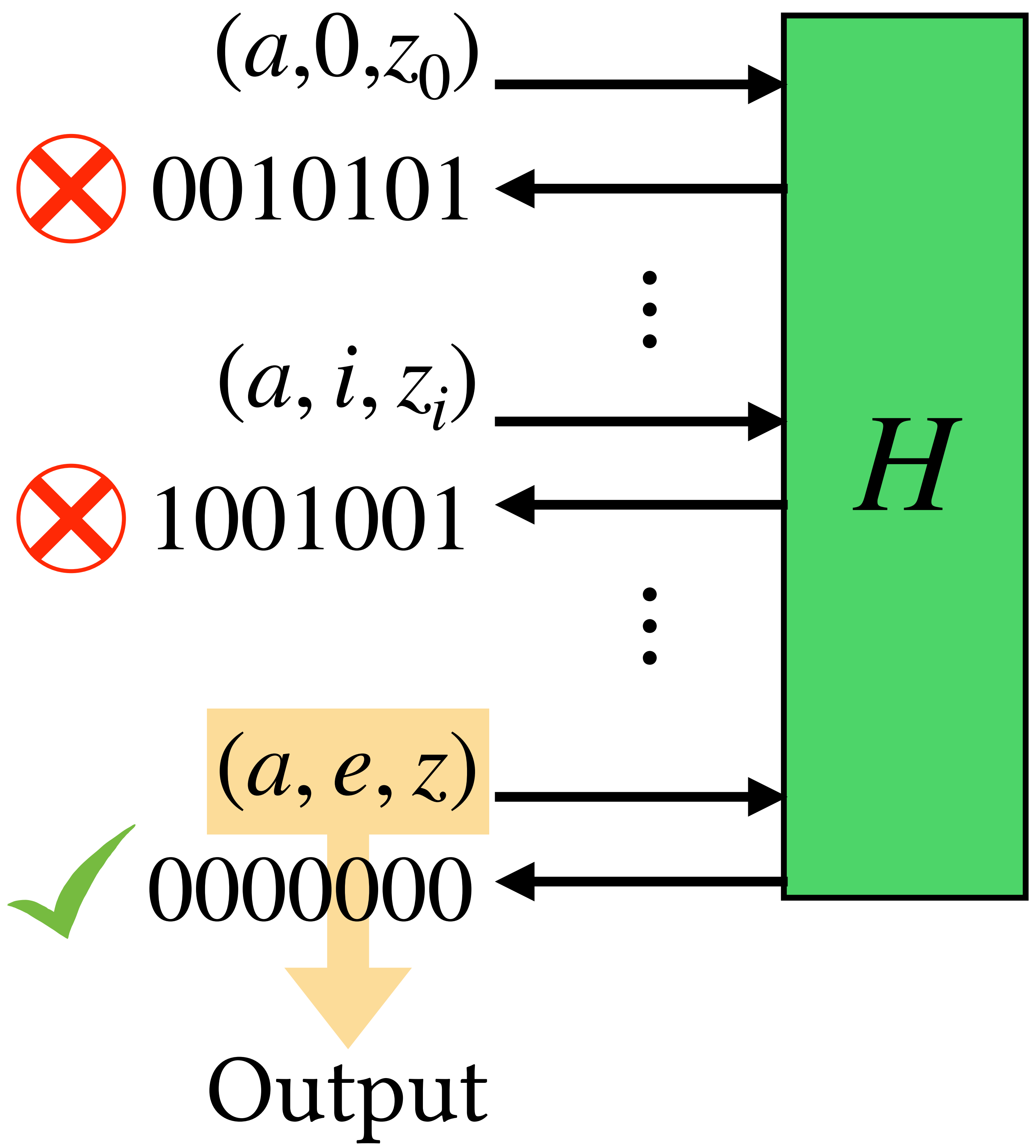
Completeness: P terminates in poly time when ℓ is small, i.e. $O(\log \kappa)$

Problem!

Fischlin's Transformation

- Let $\boxed{H}: \{0,1\}^* \mapsto \{0,1\}^\ell$ be a random oracle

$P(X, w)$: Sample Σ-protocol first message 'a'



$(a,0,z_0)$

✗ 0010101

⋮

(a,i,z_i)

✗ 1001001

⋮

(a,e,z)

✓ 0000000

Output

H

Soundness: Except with $\Pr = 2^{-\ell}$, P is forced to query more than one accepting transcript to $\boxed{H}$

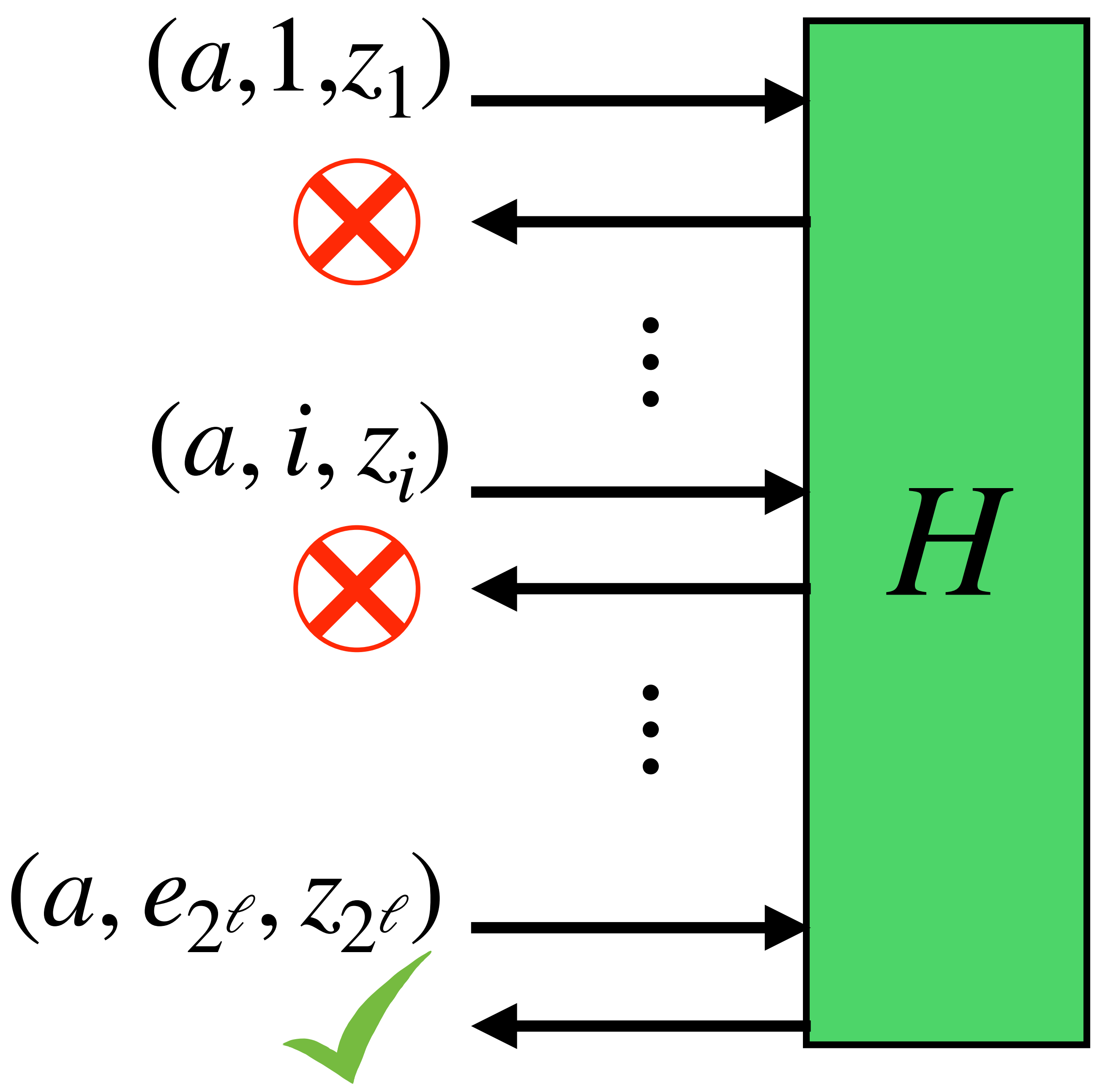
Completeness: P terminates in poly time when ℓ is small, i.e. $O(\log \kappa)$

Problem!

Full Soundness: Repeat r times

Fischlin05 vs Pass03

$P(X, w)$: Sample Σ-protocol first message ‘a’



$(a, 1, z_1)$ ✗

⋮

(a, i, z_i) ✗

⋮

$(a, e_{2^\ell}, z_{2^\ell})$ ✓

H

Commit $\begin{Bmatrix} (a, 1, z_1) \\ \vdots \\ (a, i, z_i) \\ \vdots \\ (a, e_{2^\ell}, z_{2^\ell}) \end{Bmatrix}$ Open(a, e_i, z_i)

Soundness: $2^{-\ell}$

Output: $O(\kappa)$ bits

Soundness: $2^{-\ell}$

Output: $O(\kappa) + O(\kappa) + O(\kappa \cdot \log \kappa)$

Fischlin05 vs Pass03: Qualitative

- Pass’ compiler works for *any* Sigma protocol
- Fischlin’s compiler works for a restricted class of Sigma protocols with ‘quasi-unique responses’
- Supported by many standard Sigma protocols (eg. DLog), but many *may* not—especially if a statement can have multiple witnesses (eg. Pedersen Commitment opening, 1-of-2 witnesses, etc.)

Quasi-unique Responses [Fischlin 05]

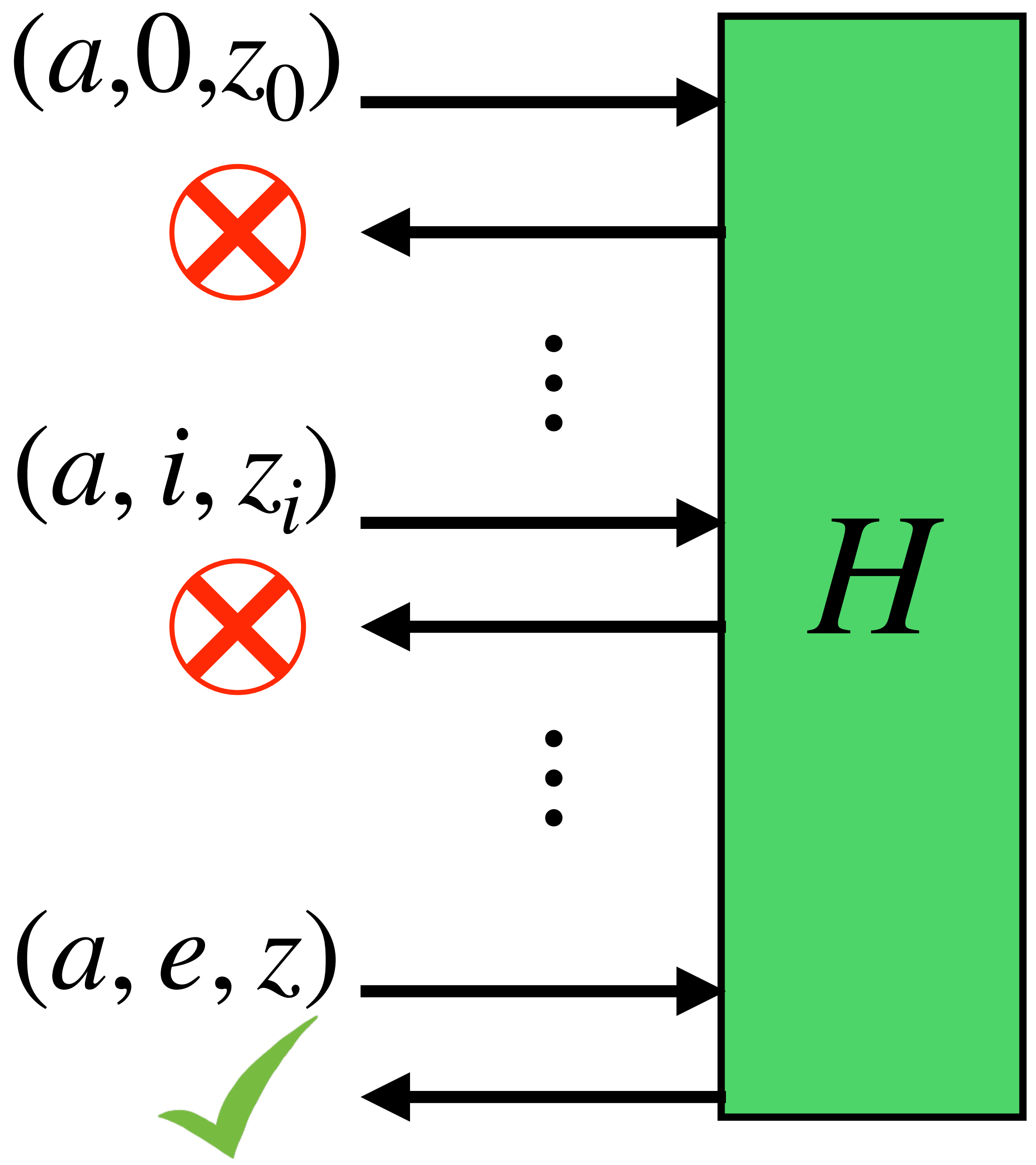
Hard: $(a, e, z, z') \leftarrow \mathscr{A}(\mathsf{pp})$ such that
$V(a, e, z) = V(a, e, z') = 1$

Fixing (a, e) fixes z

Quasi-unique Responses [Fischlin 05]

Hard: $(a, e, z, z') \leftarrow \mathscr{A}(\mathsf{pp})$ such that $V(a, e, z) = V(a, e, z') = 1$

Fixing (a, e) fixes z

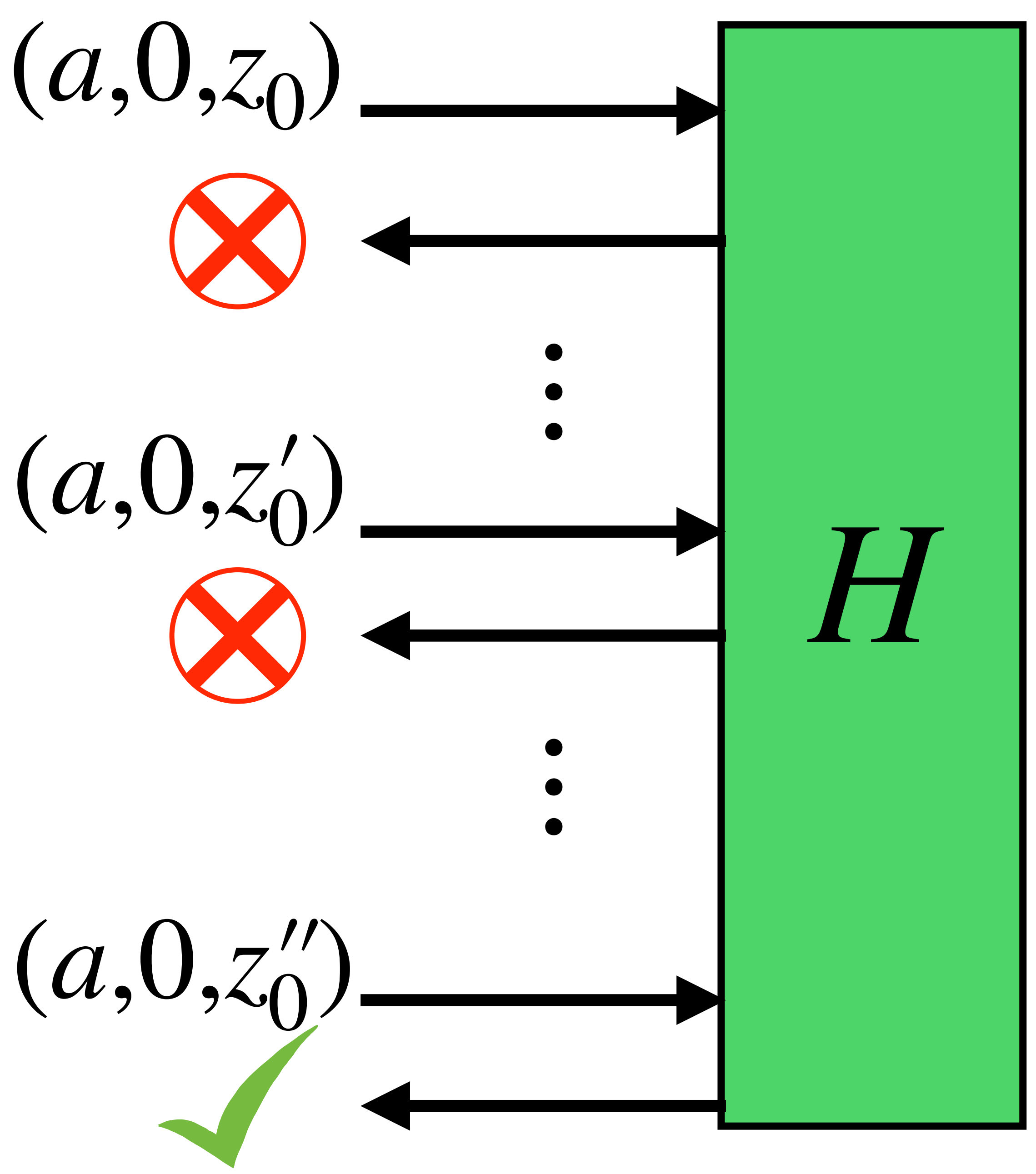


Quasi-unique Responses [Fischlin 05]

Hard: $(a, e, z, z') \leftarrow \mathscr{A}(\mathsf{pp})$ such that
$V(a, e, z) = V(a, e, z') = 1$

Fixing (a, e) fixes z

Easy to see how this ties into soundness of Fischlin's compiler

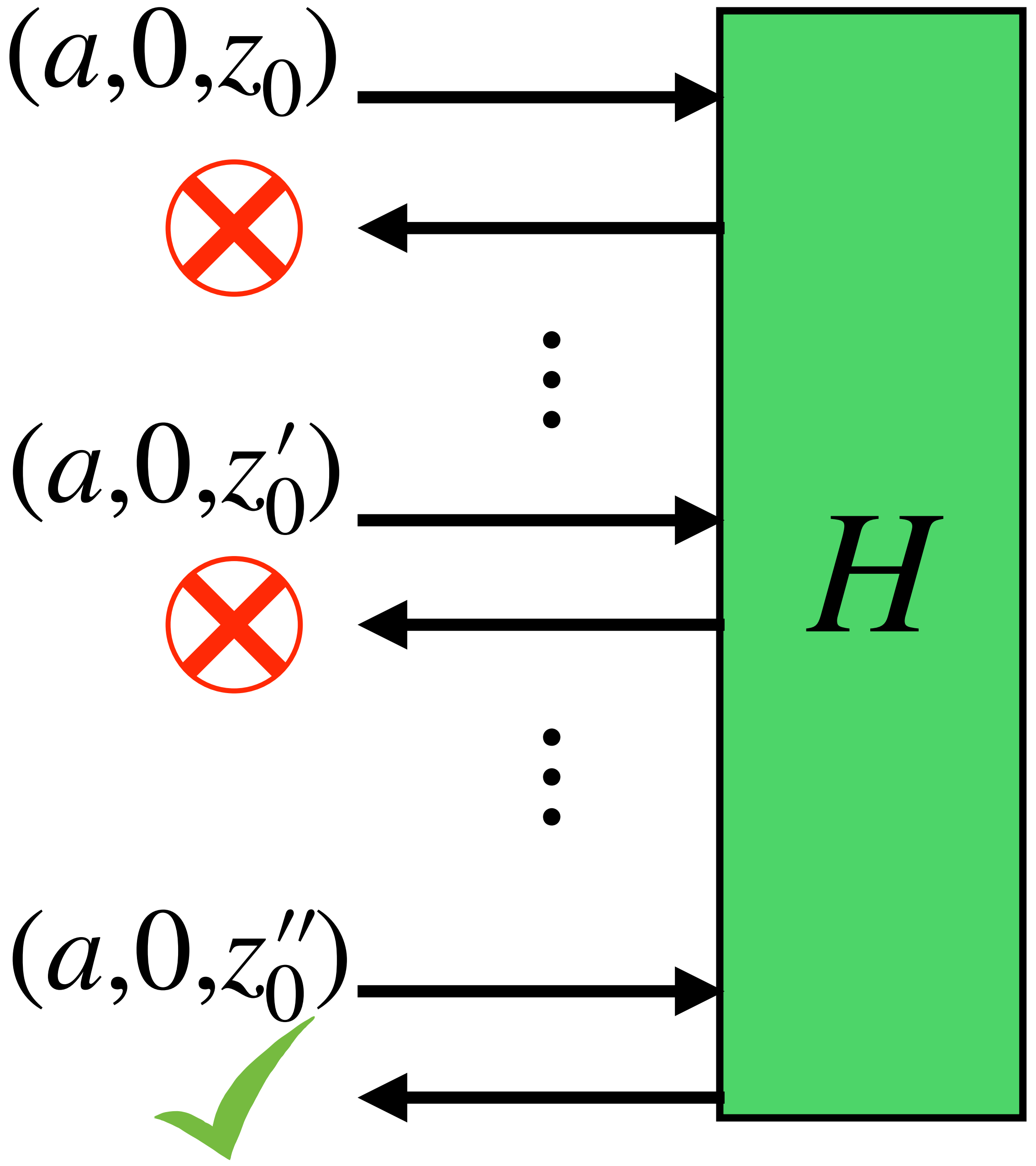


Quasi-unique Responses [Fischlin 05]

Hard: $(a, e, z, z') \leftarrow \mathscr{A}(\mathsf{pp})$ such that
$V(a, e, z) = V(a, e, z') = 1$

Fixing (a, e) fixes z

Easy to see how this ties into soundness of Fischlin's compiler



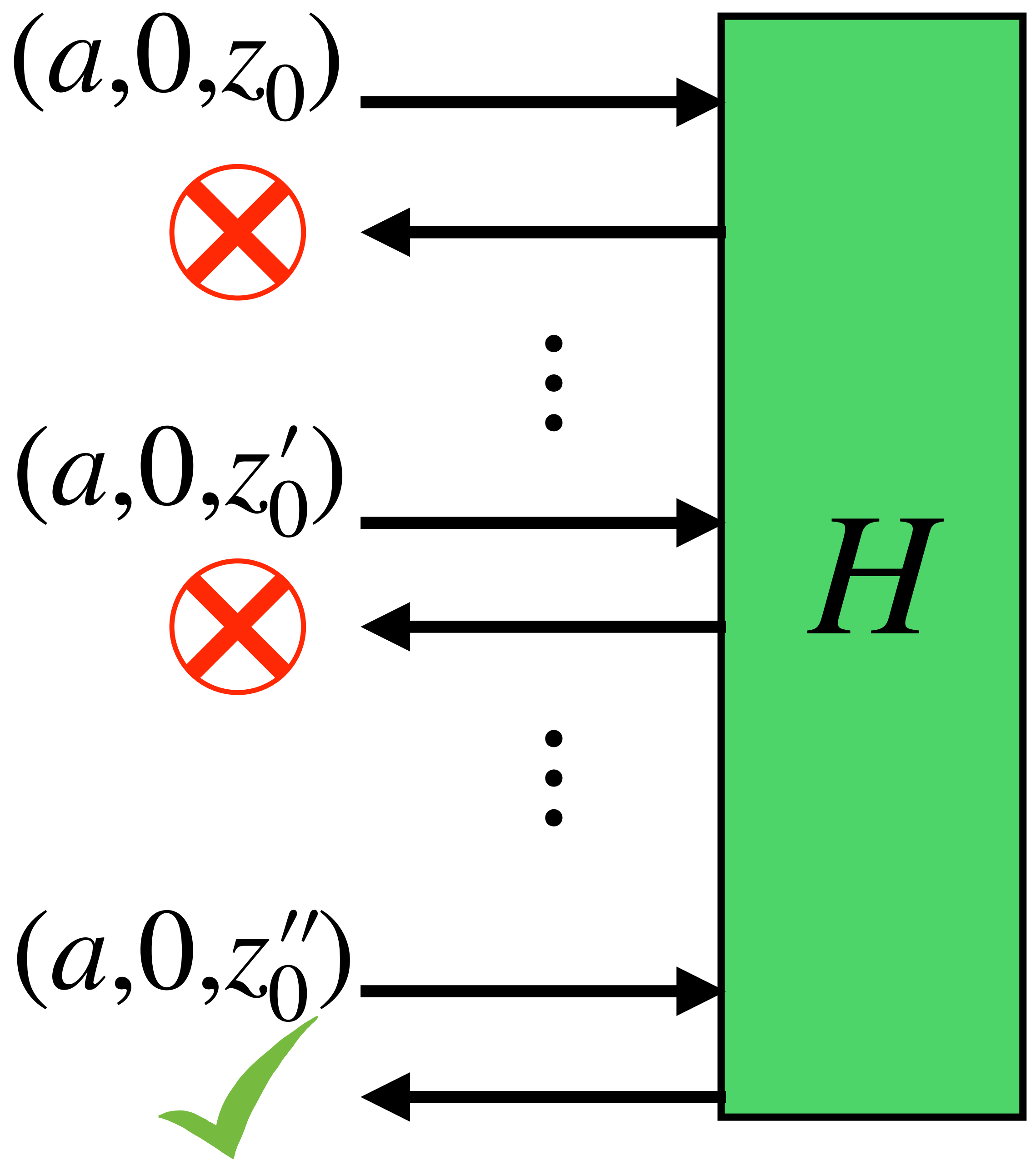
Prover can produce a proof without ever having to try more than one challenge

Quasi-unique Responses [Fischlin 05]

Hard: $(a, e, z, z') \leftarrow \mathscr{A}(\mathsf{pp})$ such that
$V(a, e, z) = V(a, e, z') = 1$

Fixing (a, e) fixes z

Easy to see how this ties into soundness of Fischlin's compiler



Prover can produce a proof without ever having to try more than one challenge

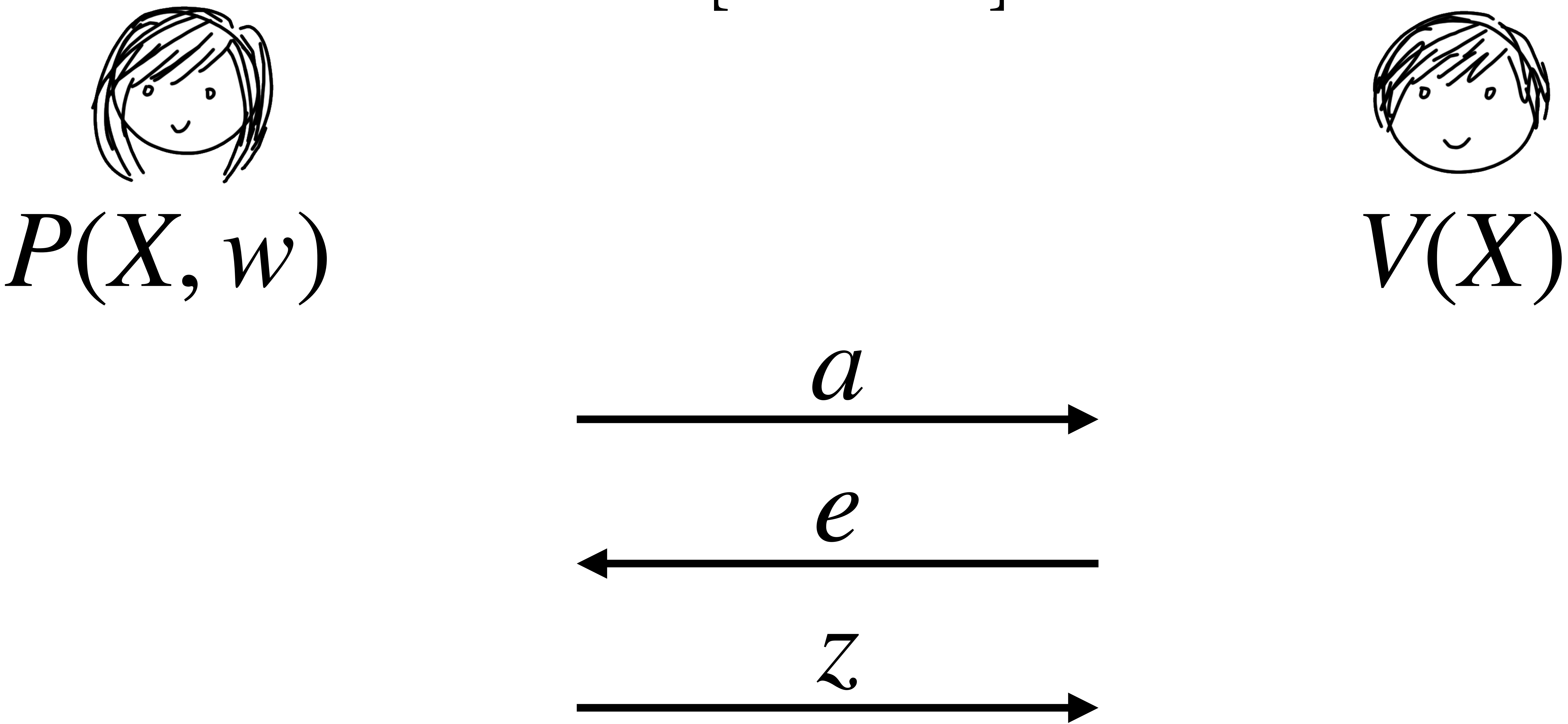
Recall:
Extractor needs transcripts with different challenges

Is it *really* necessary, though?

- Folklore: breaking Sigma protocol abstraction, and simply ‘adjusting syntax’ of the extractor is usually sufficient to preserve Proof of Knowledge
- This is demonstrated by the Sigma protocol to prove knowledge of one-out-of-two witnesses [Cramer Damgård Schoenmakers 94]
- Intuition: (a, e, z, z') allow for the extraction of a witness

Tightening Conditions for Extraction

[This work]



$P(X, w)$

$V(X)$

a

e

z

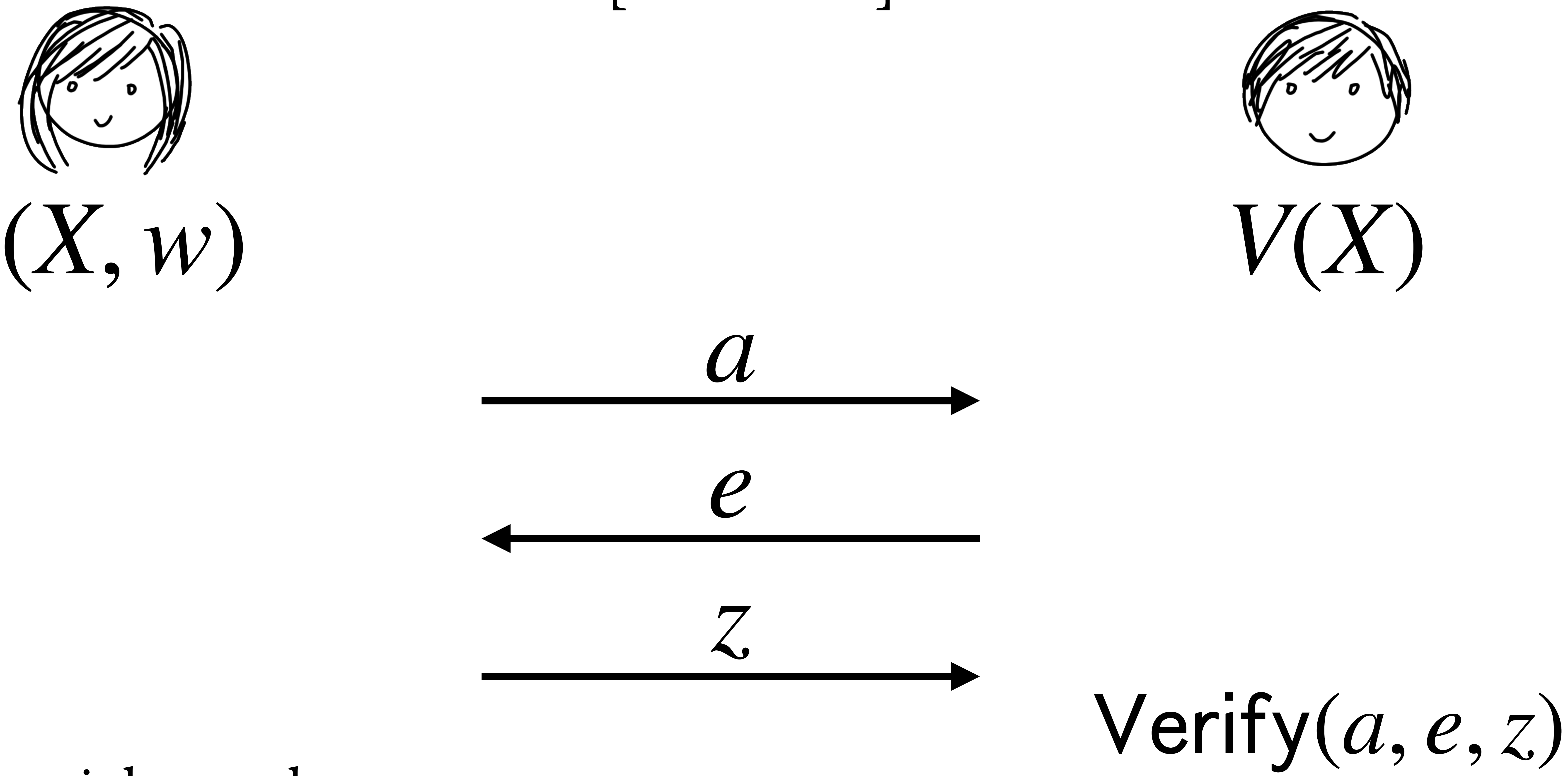
$\mathsf{Verify}(a, e, z)$

2-special soundness:

$$w \leftarrow \mathsf{Ext}(X, a, (e_1, z_1), (e_2, z_2)) \text{ such that } R(X, w) = 1$$

Tightening Conditions for Extraction

[This work]



$P(X, w)$

$V(X)$

a

e

z

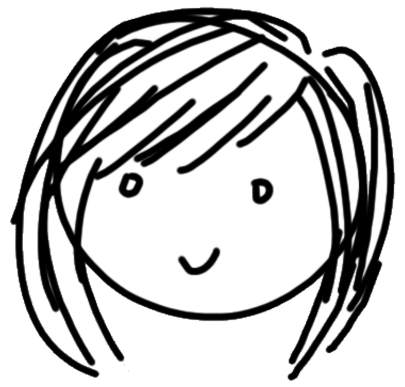
$\mathsf{Verify}(a, e, z)$

Strong 2-special soundness:

$$w \leftarrow \mathsf{Ext}(X, a, (e_1, z_1), (e_2, z_2)) \text{ such that } R(X, w) = 1$$

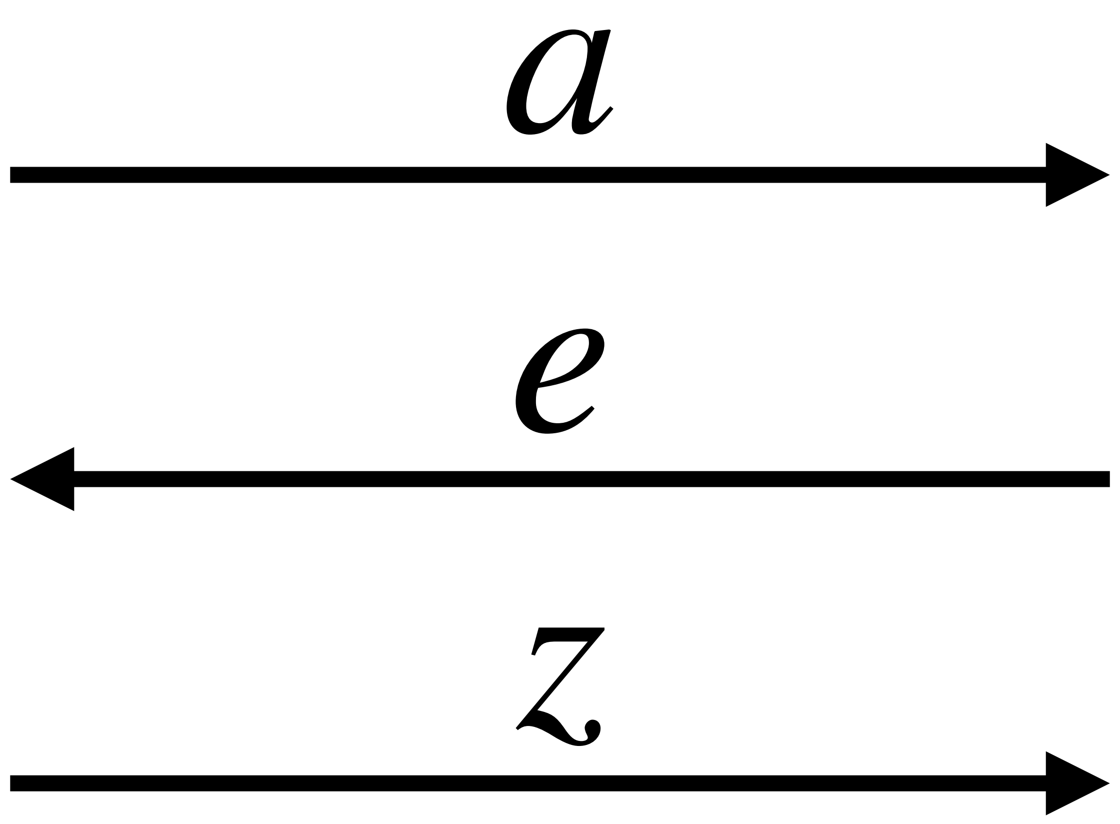
Tightening Conditions for Extraction

[This work]



$P(X, w)$

$V(X)$



$\mathsf{Verify}(a, e, z)$

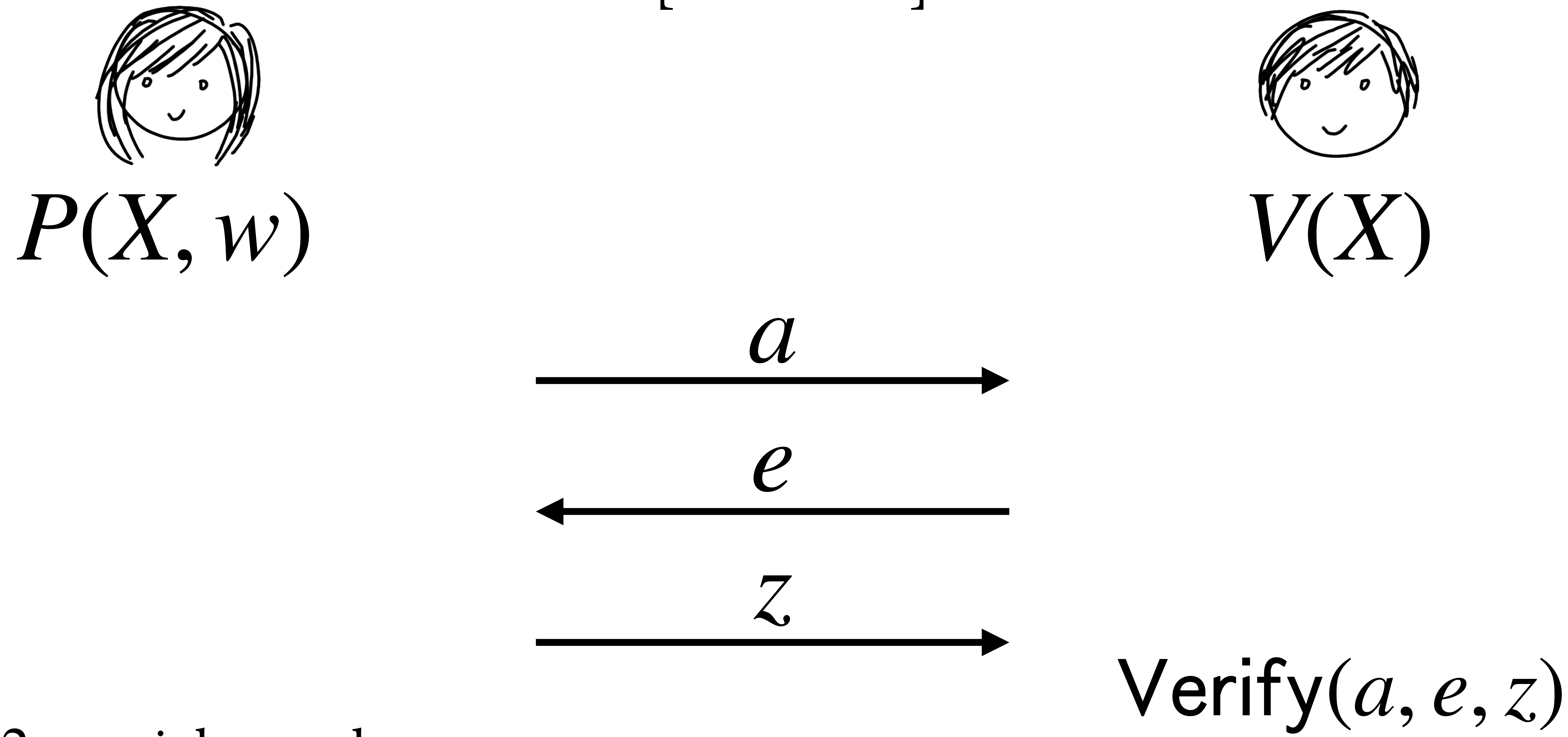
Strong 2-special soundness:

$$w \leftarrow \mathsf{Ext}(X, a, (e_1, z_1), (e_2, z_2)) \text{ such that } R(X, w) = 1$$

$$e_1 \neq e_2 \ \text{ OR } \ z_1 \neq z_2$$

Tightening Conditions for Extraction

[This work]



$P(X, w)$

$V(X)$

$$a \longrightarrow$$

$$e \longleftarrow$$

$$z \longrightarrow$$

$\mathsf{Verify}(a, e, z)$

Strong 2-special soundness:

$$w \leftarrow \mathsf{Ext}(X, a, (e_1, z_1), (e_2, z_2)) \text{ such that } R(X, w) = 1$$

$$e_1 \neq e_2 \;\text{ OR }\; z_1 \neq z_2$$

...are we done?

What about Zero-knowledge?

- Interestingly, Fischlin's proof of Zero-knowledge also depends on quasi-unique responses

- Unlike extraction, it is not intuitive as to why (or whether it's even necessary)

- [This work]: In the absence of unique responses, an explicit attack on *Witness Indistinguishability*

The Attack

- **Fact 1**: In some Sigma protocols, the prover's internal state is exposed to an adversary who has the witness.
  eg. Schnorr: $z = xe + r$; given x can solve for r

- **Fact 2**: Once a is fixed, Fischlin's compiler is deterministic

The Attack

- **Fact 1**: In some Sigma protocols, the prover's internal state is exposed to an adversary who has the witness.
  eg. Schnorr: $z = xe + r$; given x can solve for r

- **Fact 2**: Once a is fixed, Fischlin's compiler is deterministic

Attacker $\mathscr{A}$: Given Fischlin-compiled proof π, retrieve prover's internal state, and retrace its steps, i.e. attempt to recompute the proof

The Attack

- **Fact 1**: In some Sigma protocols, the prover's internal state is exposed to an adversary who has the witness.
  eg. Schnorr: $z = xe + r$; given x can solve for r

- **Fact 2**: Once a is fixed, Fischlin's compiler is deterministic

Attacker $\mathscr{A}$: Given Fischlin-compiled proof π, retrieve prover's internal state, and retrace its steps, i.e. attempt to recompute the proof

- **Fact 3**: In some Sigma protocols, for the same (a, e), the response z will depend on which witness is used. e.g. PoK of w_0 OR w_1

The Attack

- **Fact 1**: In some Sigma protocols, the prover's internal state is exposed to an adversary who has the witness.
  eg. Schnorr: $z = xe + r$; given x can solve for r

- **Fact 2**: Once a is fixed, Fischlin's compiler is deterministic

Attacker $\mathscr{A}$: Given Fischlin-compiled proof π, retrieve prover's internal state, and retrace its steps, i.e. attempt to recompute the proof

- **Fact 3**: In some Sigma protocols, for the same (a, e), the response z will depend on which witness is used. e.g. PoK of w_0 OR w_1

If the "wrong" witness is used, w.h.p. $\mathscr{A}$ will output a *different* proof $\pi' \neq \pi$

How to Fix it?

- Can’t do anything about **Fact 1** and **Fact 3**, i.e. properties of many natural Sigma protocols
- We can fix **Fact 2**—Fischlin’s compiler can be randomized
- Instead of incrementally stepping through challenges, the Prover can try *random* challenges until an accepting transcript is found
- Retrieving Sigma protocol randomness (via Fact 1) is now insufficient to retrace the Prover’s steps

This Work

- We explore two dimensions of Fischlin's NIZKPoK compiler:

✓ **Applicability**:
Only proven for Sigma protocols with 'quasi-unique responses'
(doesn't include logical OR, Pedersen commitment PoK, etc.)
Folklore: "works anyway"

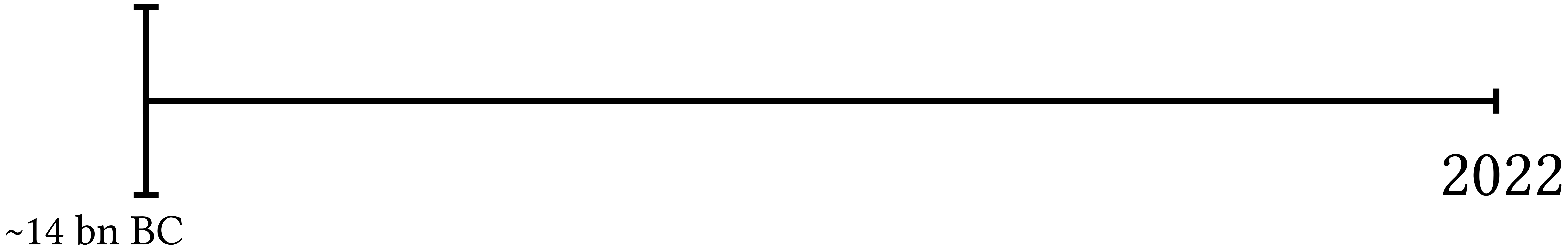
1a) Contrary to folklore: attack on Witness Indistinguishability
1b) Simple randomization fixes the problem

- **Computation cost**:
Usually the bottleneck — can we improve on it?

2) Lower bound: Fischlin05 is optimal up to a small constant
3) Application-specific optimization: 200× for EdDSA aggregation

Straight-line Extractable NIZK in the ROM

For simple algebraic statements,
eg. Schnorr’s PoK of DLog

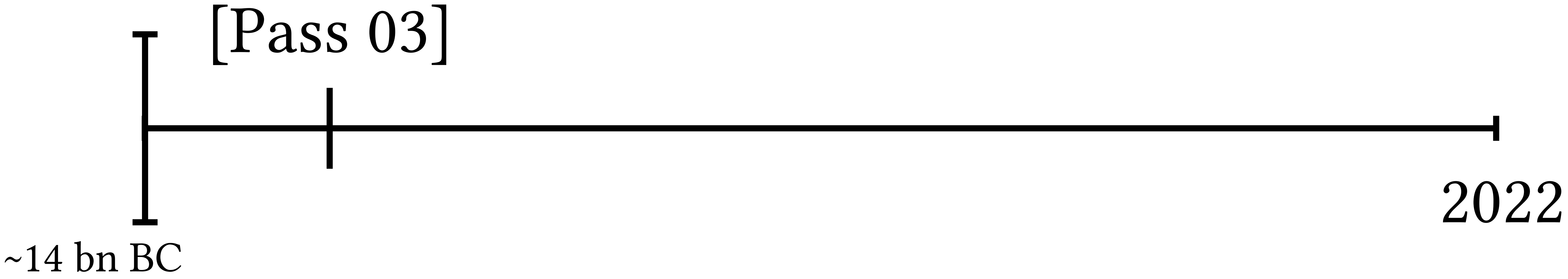


2022

~14 bn BC

Straight-line Extractable NIZK in the ROM

For simple algebraic statements,
eg. Schnorr’s PoK of DLog



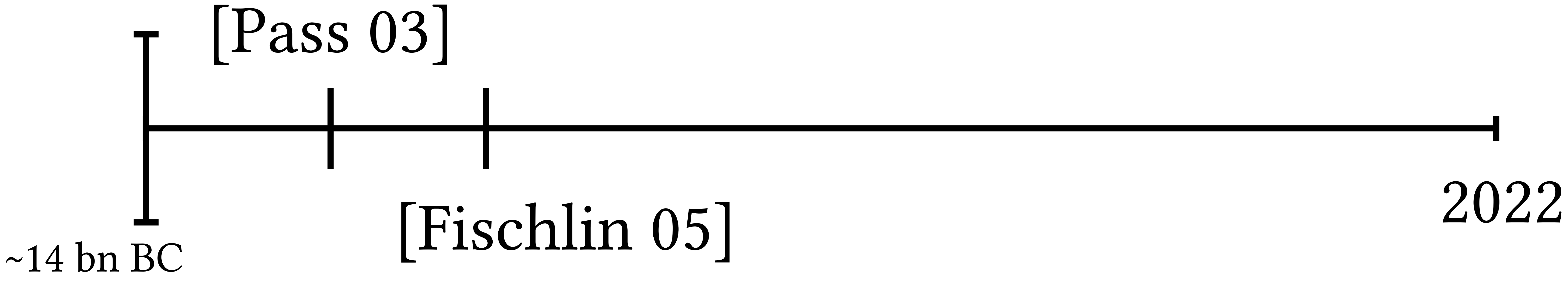
[Pass 03]

~14 bn BC

2022

Straight-line Extractable NIZK in the ROM

For simple algebraic statements,
eg. Schnorr's PoK of DLog



[Pass 03]

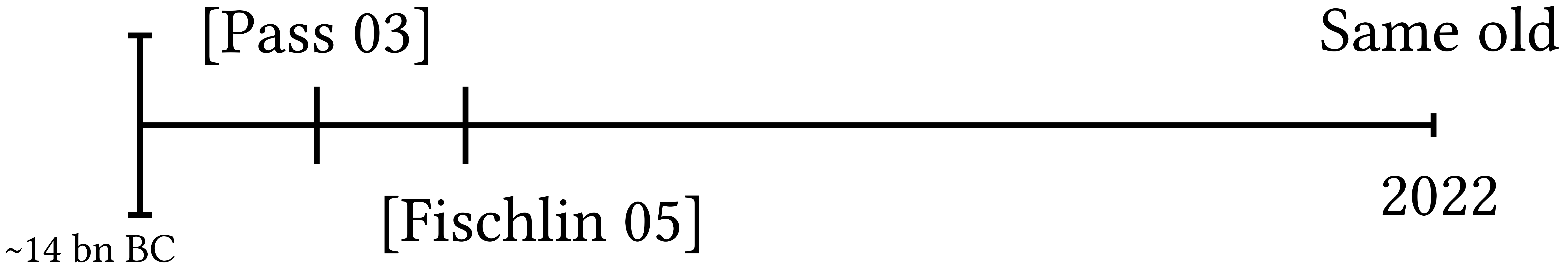
~14 bn BC

[Fischlin 05]

2022

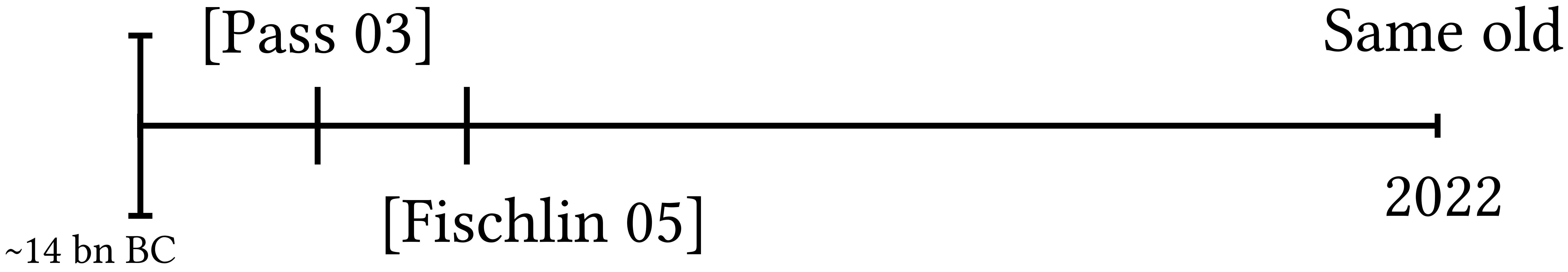
Straight-line Extractable NIZK in the ROM

For simple algebraic statements,
eg. Schnorr's PoK of DLog



Straight-line Extractable NIZK in the ROM

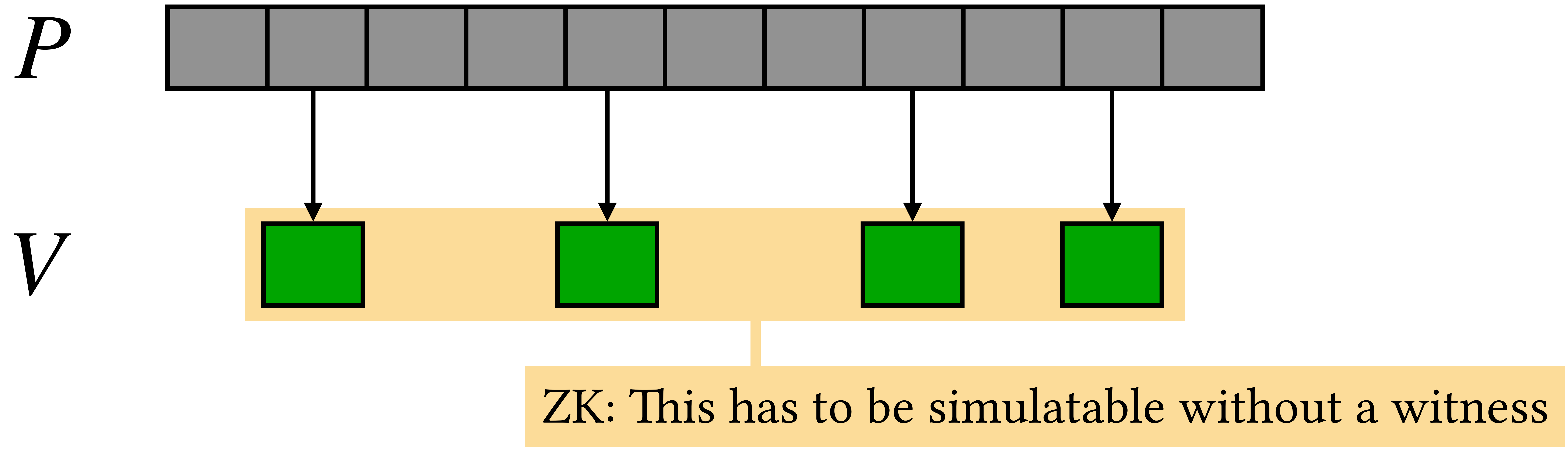
For simple algebraic statements,
eg. Schnorr’s PoK of DLog


[Pass 03]
Same old
~14 bn BC
[Fischlin 05]
2022

But why?

RO Query Complexity for NIZK

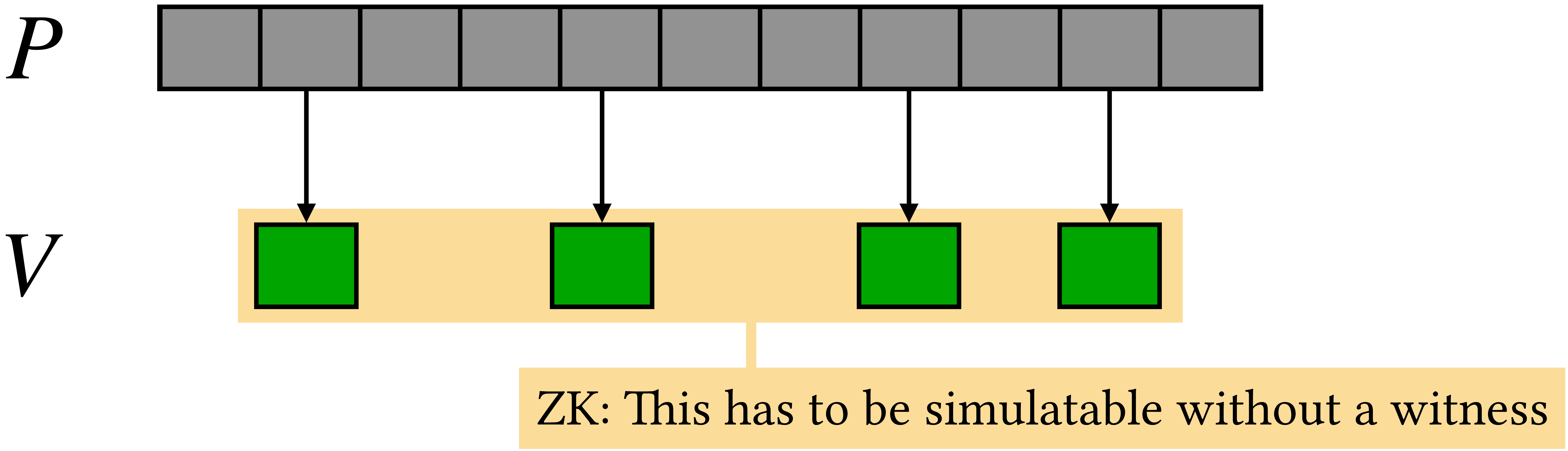
- If ZK is desired, we can prove that Fischlin's technique is nearly optimal (within factor of $e \approx 2.7$) for a *non-programming* straight-line extractor
- Our proof is a tightening of an asymptotic bound in [Fischlin 05]
- Lower bound states that if verifier makes V queries and prover P, then $\binom{P}{V} > 2^{\kappa}$



ZK: This has to be simulatable without a witness

RO Query Complexity for NIZK

- If ZK is desired, we can prove that Fischlin's technique is nearly optimal (within factor of $e \approx 2.7$) for a *non-programming* straight-line extractor
- Our proof is a tightening of an asymptotic bound in [Fischlin 05]
- Lower bound states that if verifier makes V queries and prover P, then $\binom{P}{V} > 2^{\kappa}$



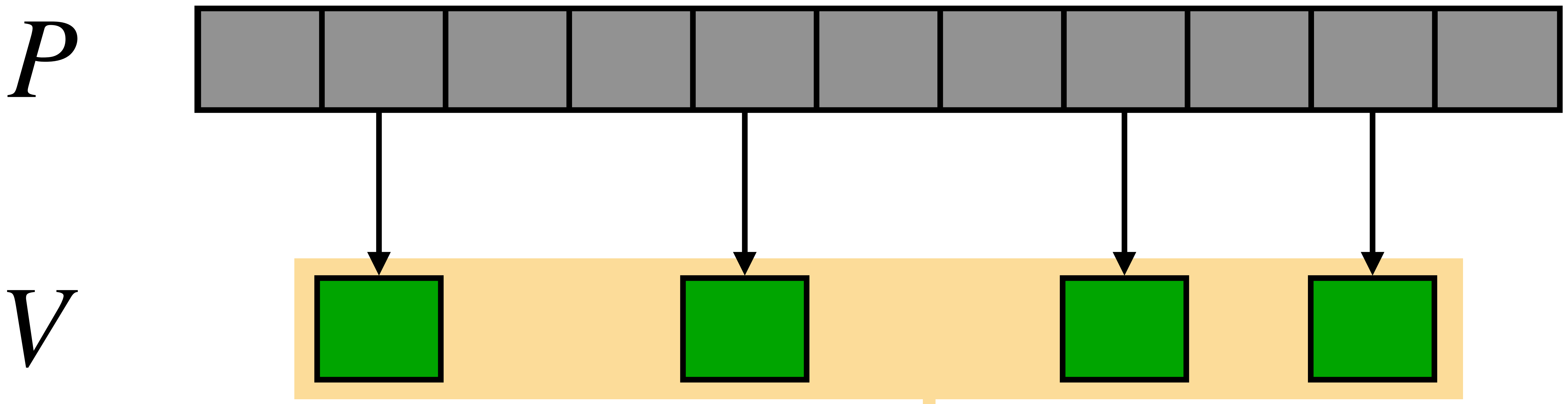
ZK: This has to be simulatable without a witness

RO Query Complexity for NIZK

- If ZK is desired, we can prove that Fischlin's technique is nearly optimal (within factor of $e \approx 2.7$) for a *non-programming* straight-li

Loose bound? Or room for improvement?

- Our proof is a tightening of an asymptotic bound in [Fischlin 05]
- Lower bound states that if verifier makes V queries and prover P, then $\binom{P}{V} > 2^{\kappa}$



ZK: This has to be simulatable without a witness

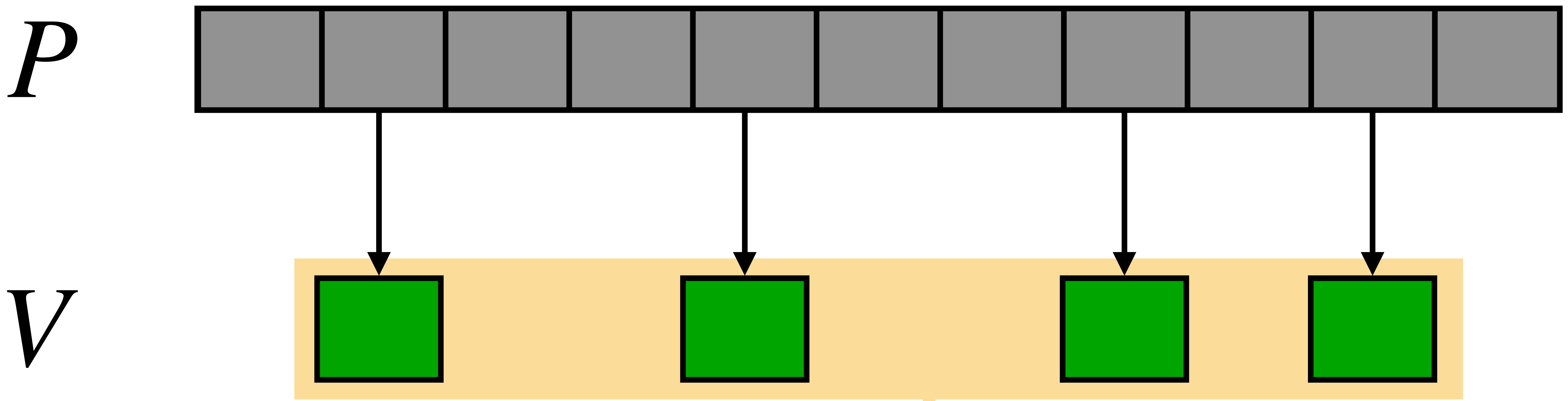
RO Query Complexity for NIZK

- If ZK is desired, we can prove that Fischlin's technique is nearly optimal (within factor of $e \approx 2.7$) for a *non-programming* straight-li
- Our proof is a tightening of an asymptotic boun
- Lower bound states that if verifier makes V quer

Loose bound?

Or room for improvement?

Having the Prover find *collisions* rather than inversions of H gives a bit of a speedup



ZK: This has to be simulatable without a witness

RO Query Complexity for NIZK

- If ZK is desired, we can prove that Fischlin's technique is nearly optimal (within factor of $e \approx 2.7$) for a *non-programming* straight-lir

Loose bound?

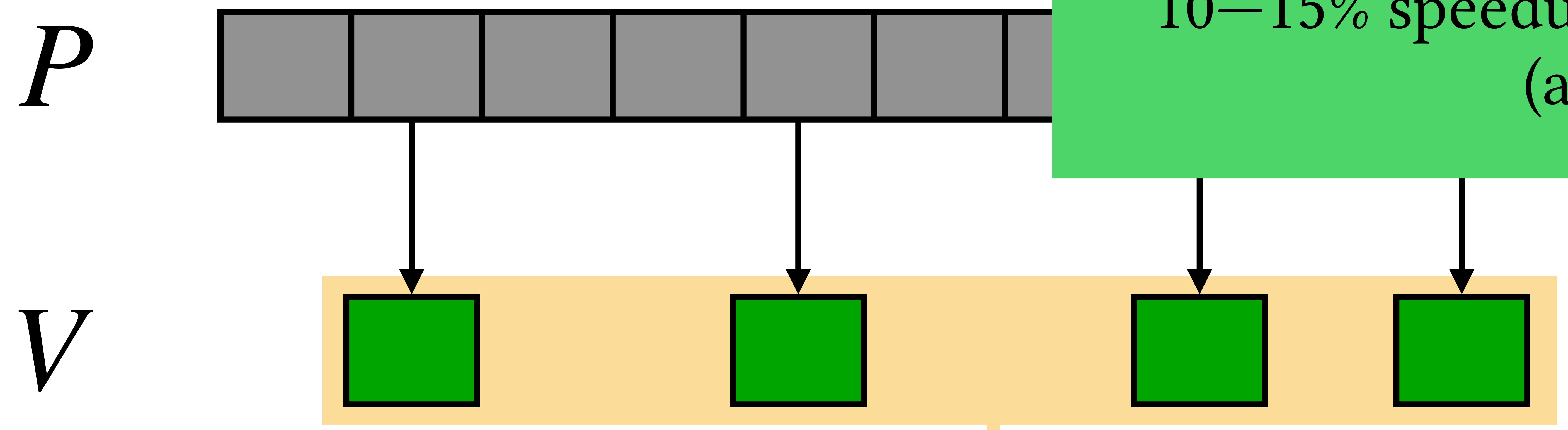
Or room for improvement?

- Our proof is a tightening of an asymptotic boun

Having the Prover find *collisions* rather than inversions of H gives a bit of a speedup

10—15% speedup in the general case for (almost) free

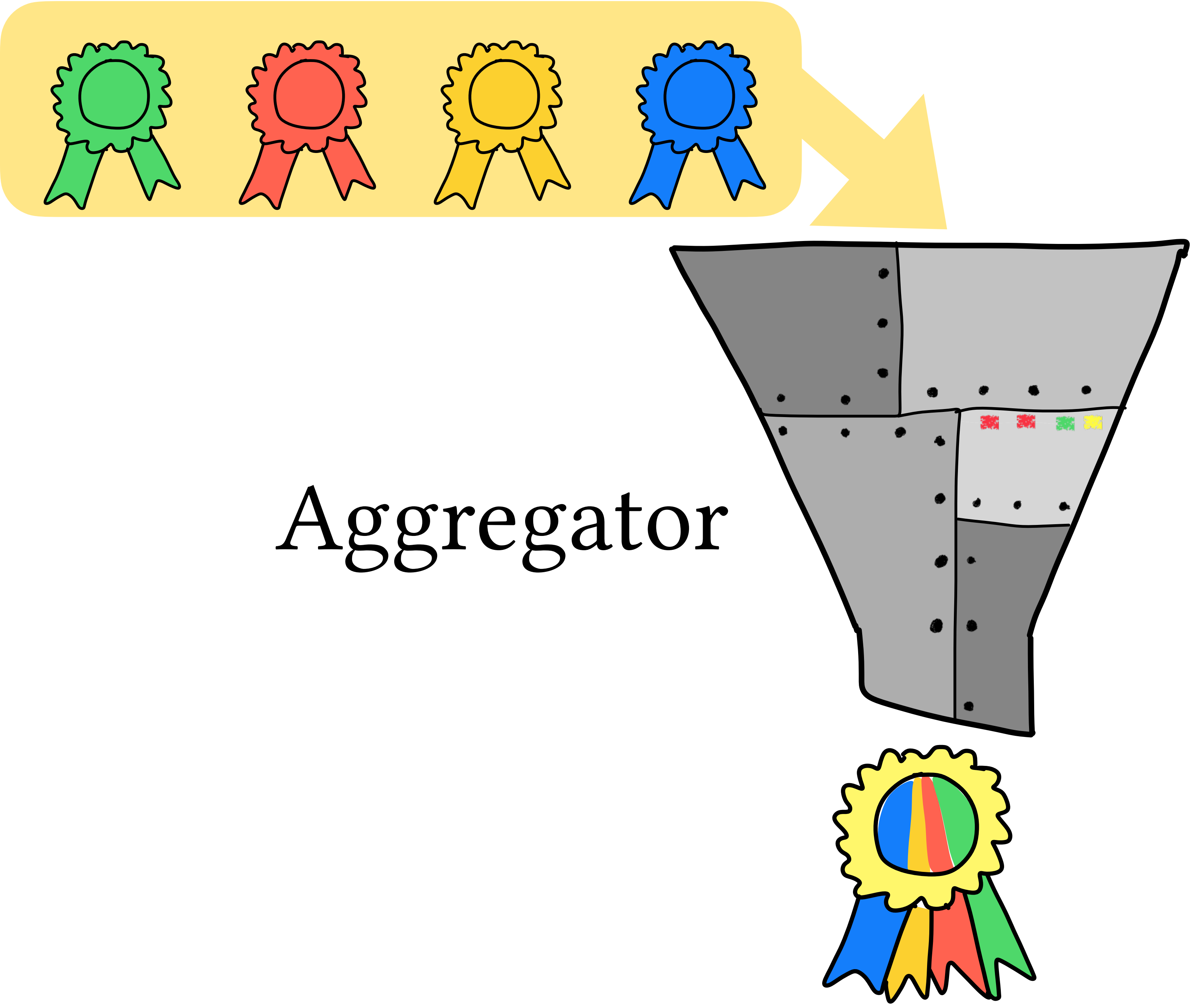
- Lower bound states that if verifier makes V quer

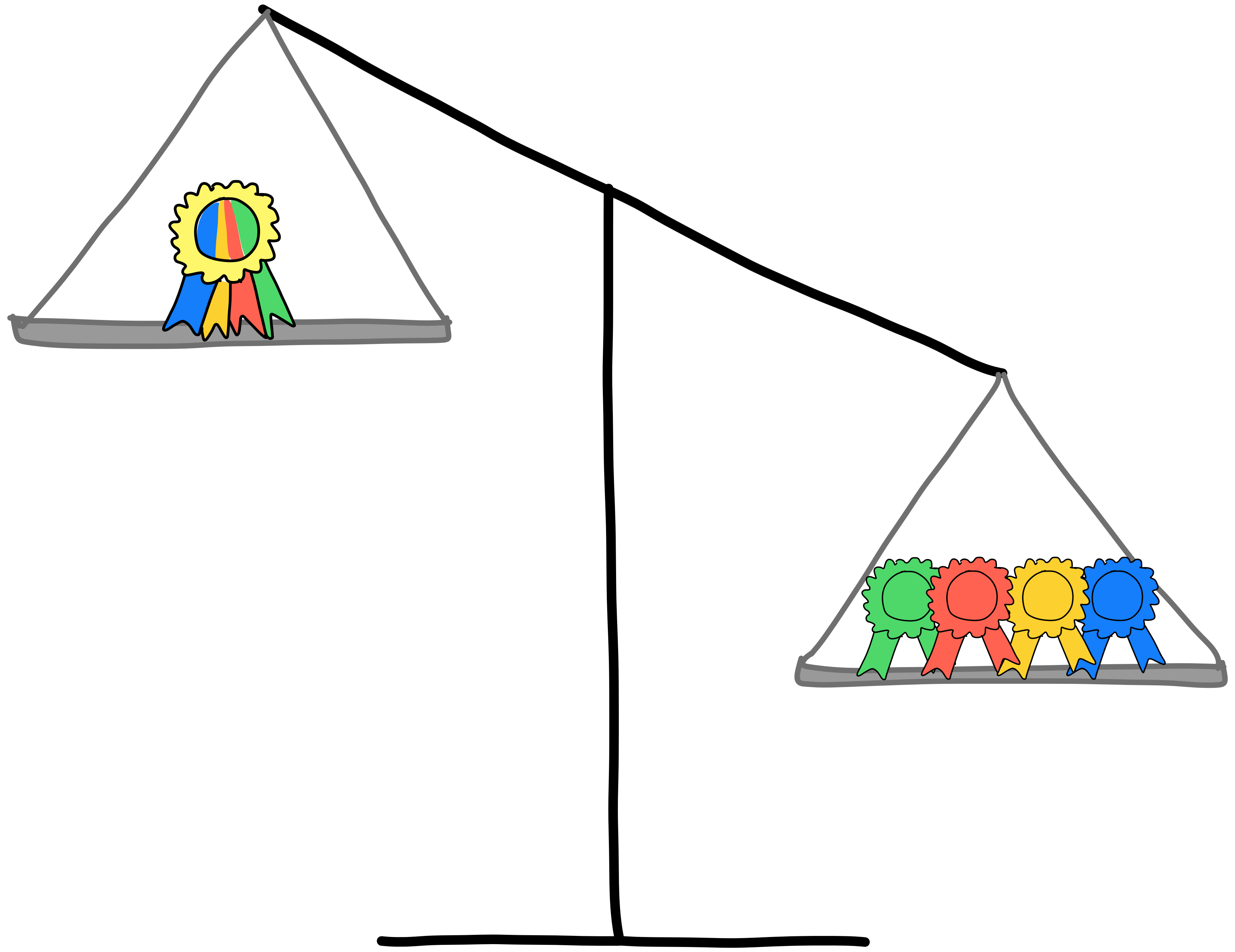


ZK: This has to be simulatable without a witness

Application-Specific Optimization

- We show that it is possible to optimize computation cost of Fischlin's technique in specific applications
- We consider Schnorr/EdDSA signature aggregation [CGKN21]: **200× improvement**

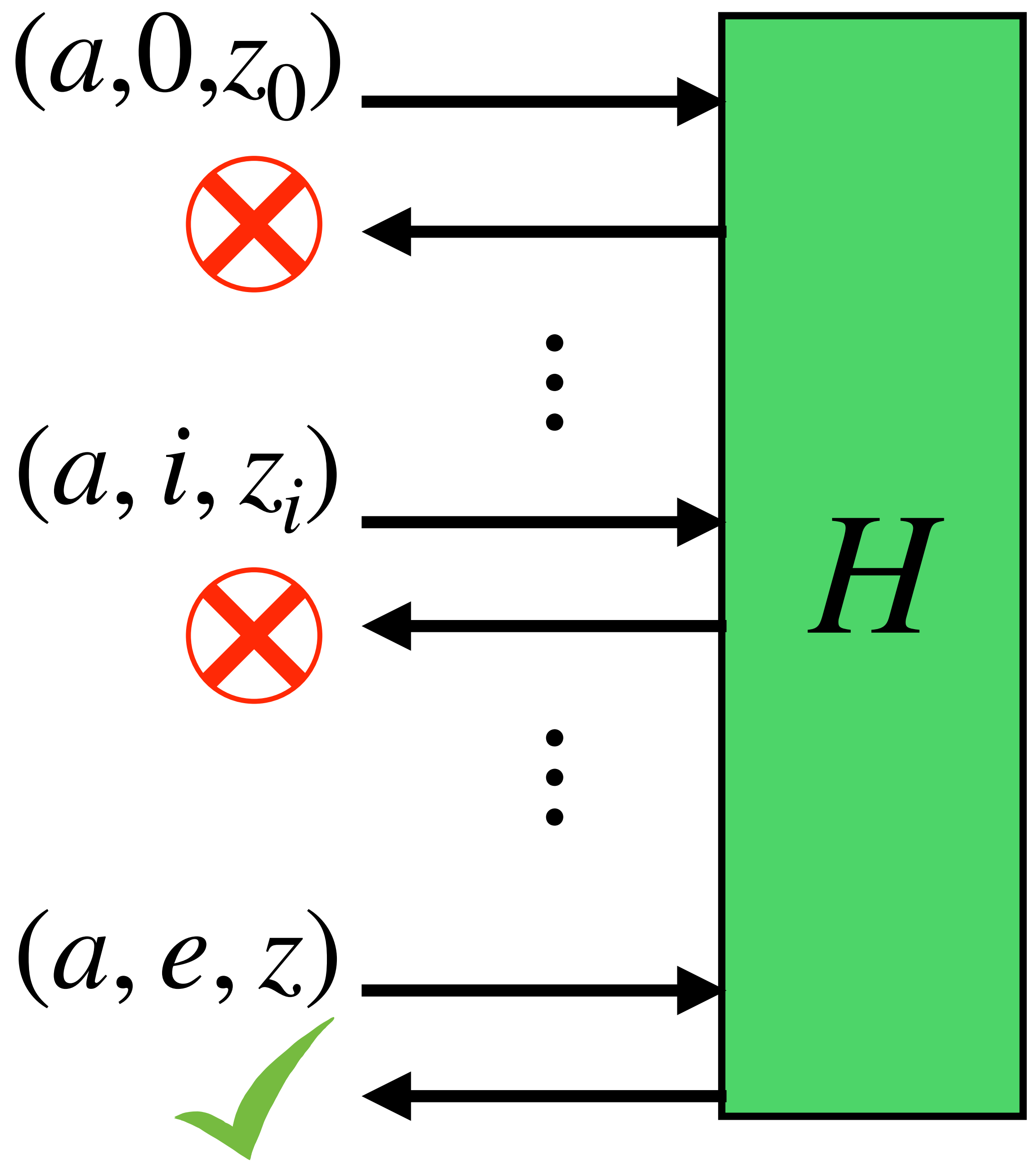




Understanding Computation Cost

- Let $\boxed{H}: \{0,1\}^* \mapsto \{0,1\}^\ell$ be a random oracle

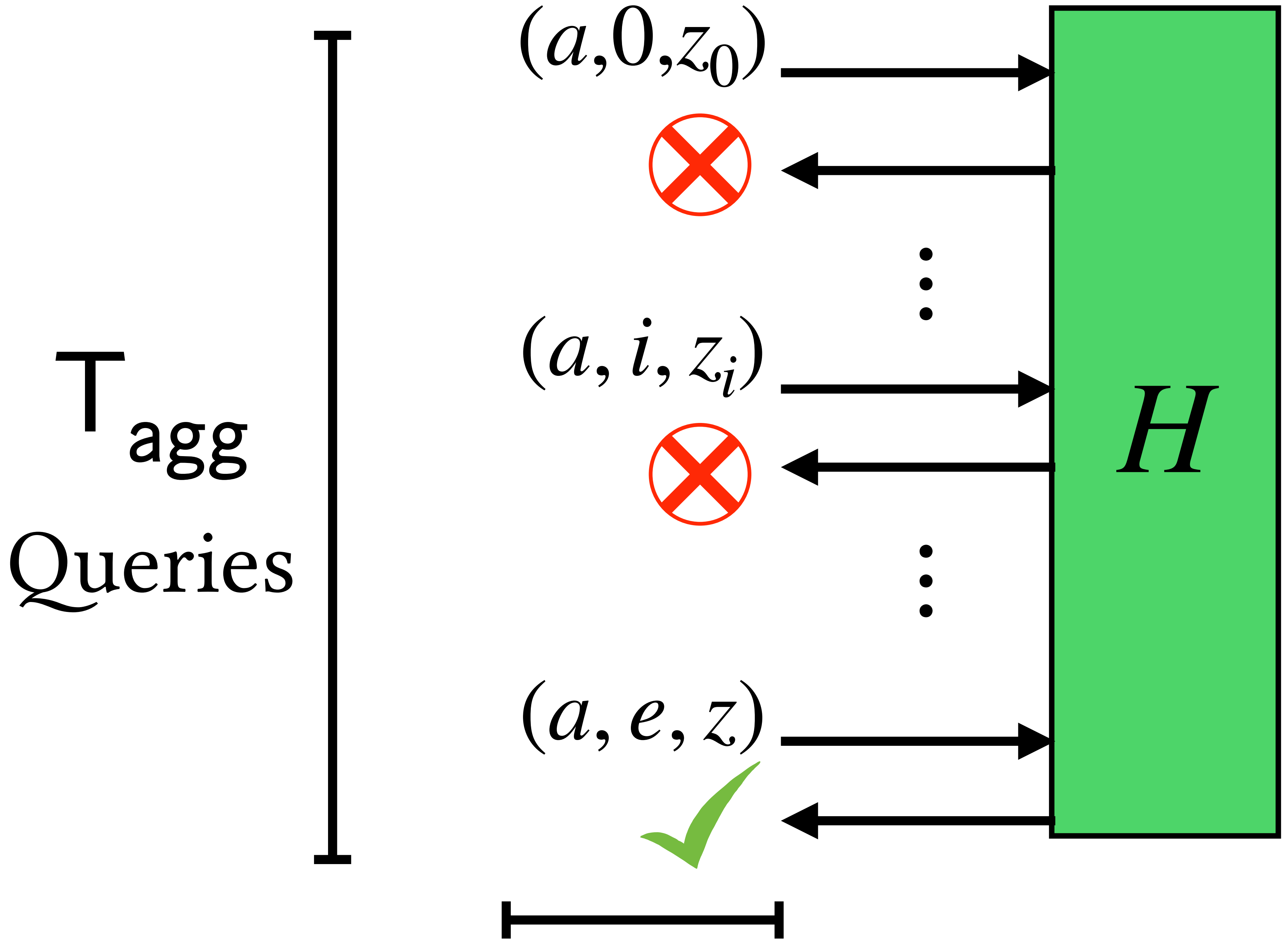
$P(X, w)$: Sample Σ-protocol first message ‘a’



Understanding Computation Cost

- Let $\boxed{H}: \{0,1\}^* \mapsto \{0,1\}^{\ell}$ be a random oracle

$P(X, w)$: Sample Σ-protocol first message ‘a’



$(a, 0, z_0)$

(a, i, z_i)

(a, e, z)

H

$\mathsf{T}_{\mathsf{agg}}$ Queries

Costs $\mathsf{C}_{\mathsf{qry}}$ per query

<u>Total cost</u>: $\mathsf{T}_{\mathsf{agg}} \cdot \mathsf{C}_{\mathsf{qry}}$

We improve both dimensions

Improving $\mathsf{T}_{\mathsf{agg}}$

- The query complexity $\mathsf{T}_{\mathsf{agg}}$ corresponds to the (expected) running time of finding r inversions of an ℓ-bit hash function

- Insight: finding r **collision** of ℓ'-bit hash is 1.5—2× **faster than inversion**

  via birthday attack combinatorial analyses [von Mises 39, Preneel 93]
  (ℓ' adjusted to respect the security constraint for the same κ)

- This translates to the Zero-Knowledge (NIZK) setting as well

Improving $\mathsf{T}_{\mathsf{agg}}$

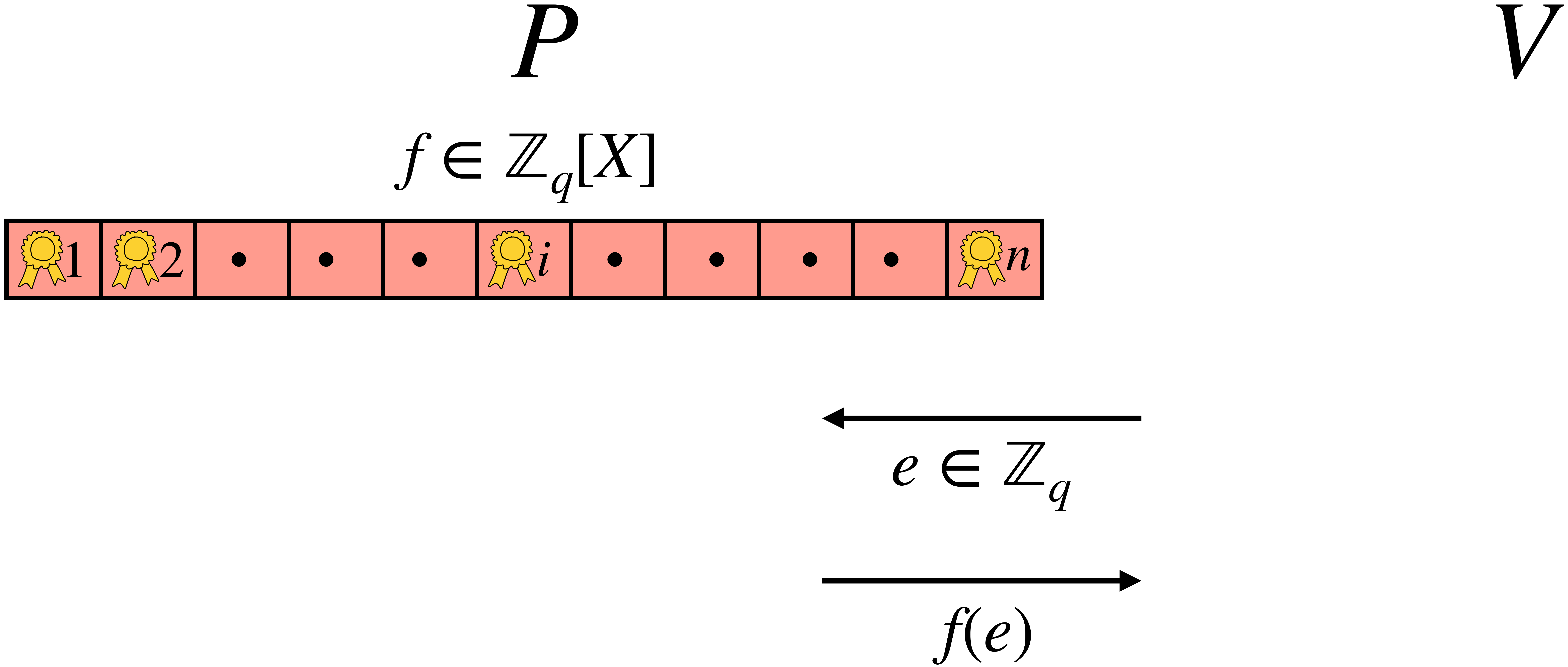
- The query complexity $\mathsf{T}_{\mathsf{agg}}$ corresponds to the (expected) running time of finding r inversions of an ℓ-bit hash function

- Insight: finding r **collision** of ℓ'-bit hash is 1.5—2× **faster than inversion**

  via birthday attack combinatorial analyses [von Mises 39, Preneel 93]
  (ℓ' adjusted to respect the security constraint for the same κ)

- This translates to the Zero-Knowledge (NIZK) setting as well

Improving C_{qry}



P V

$f \in \mathbb{Z}_q[X]$

$\xleftarrow{\hspace{3em}}$

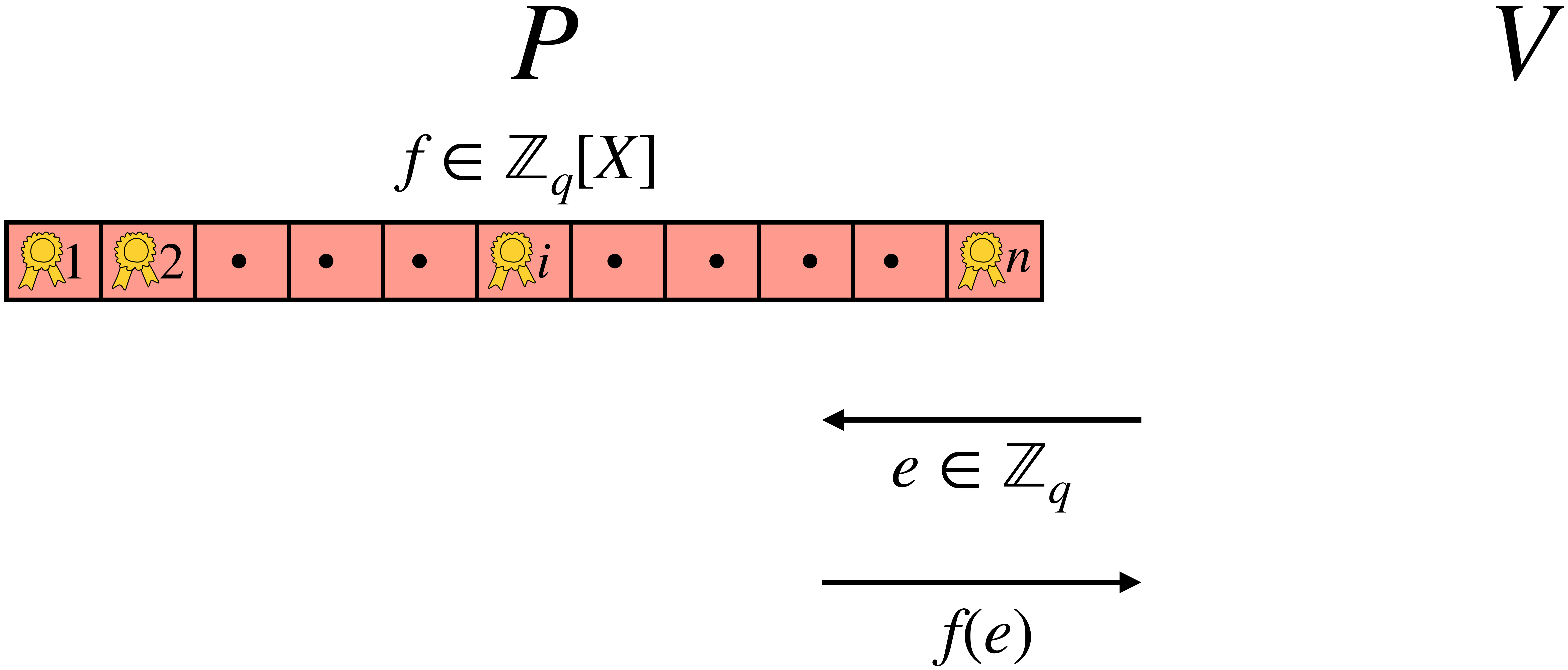
$e \in \mathbb{Z}_q$

$\xrightarrow{\hspace{3em}}$

$f(e)$

Improving $\mathsf{C}_{\mathsf{qry}}$

$\mathsf{C}_{\mathsf{qry}}$ in Schnorr aggregation Sigma protocol:



P

V

$f \in \mathbb{Z}_q[X]$

$\xleftarrow{\qquad}$

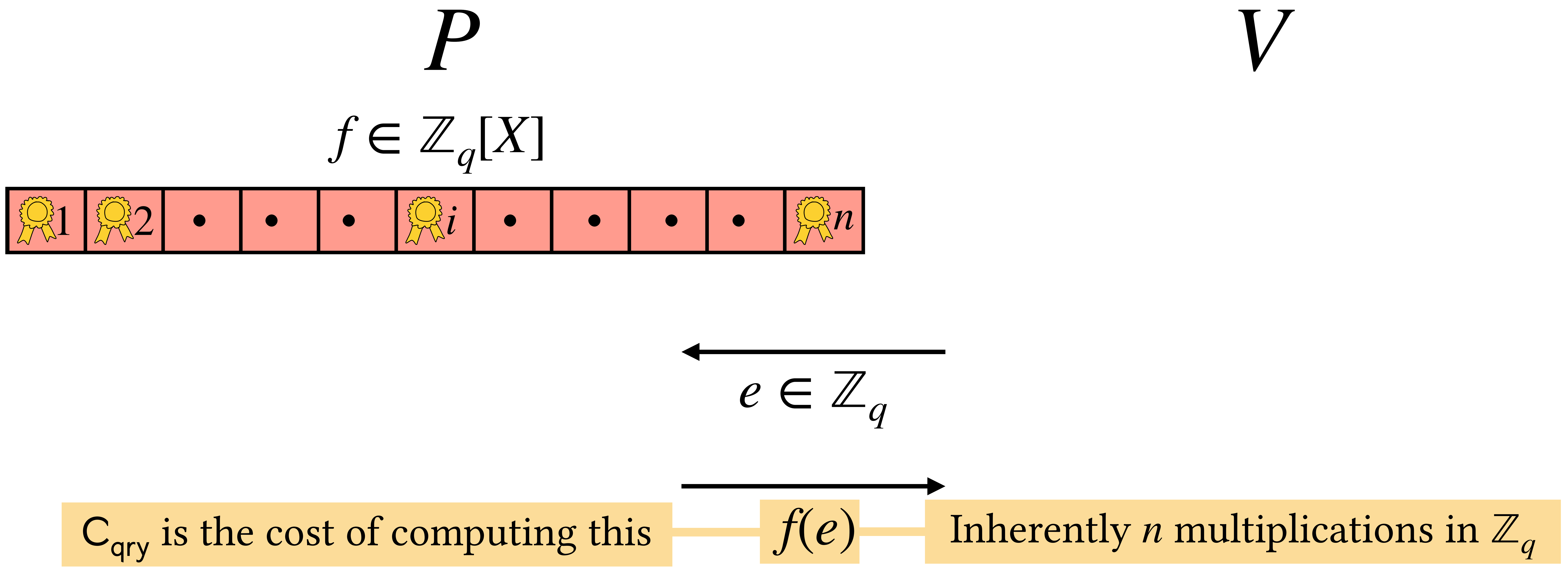
$e \in \mathbb{Z}_q$

$\xrightarrow{\qquad}$

$f(e)$

Improving $\mathsf{C}_{\mathsf{qry}}$

$\mathsf{C}_{\mathsf{qry}}$ in Schnorr aggregation Sigma protocol:



P

V

$f \in \mathbb{Z}_q[X]$

$e \in \mathbb{Z}_q$

$f(e)$

$\mathsf{C}_{\mathsf{qry}}$ is the cost of computing this

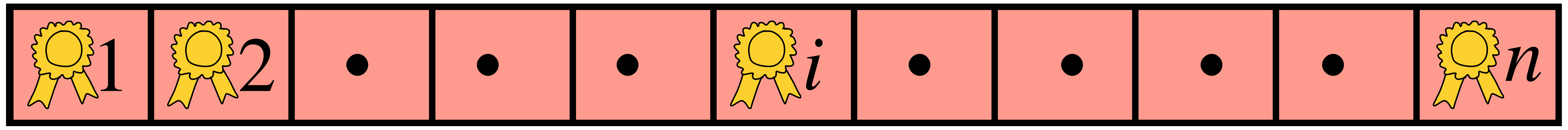
Inherently n multiplications in $\mathbb{Z}_q$

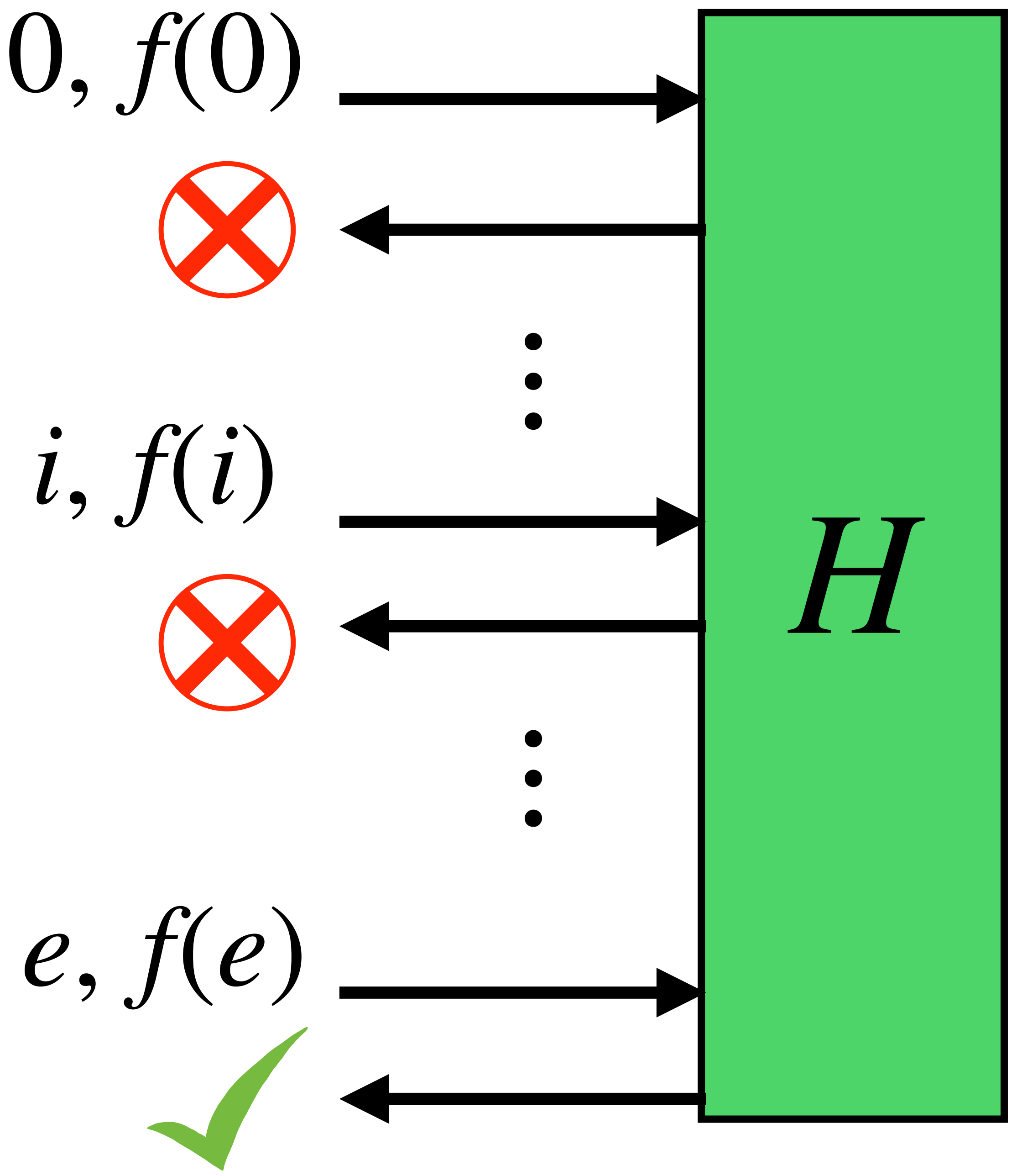
Improving $\mathsf{C}_{\mathsf{qry}}$

$\mathsf{C}_{\mathsf{qry}}$ in Schnorr aggregation **Fischlin proof**

$$P$$

$$f \in \mathbb{Z}_q[X]$$



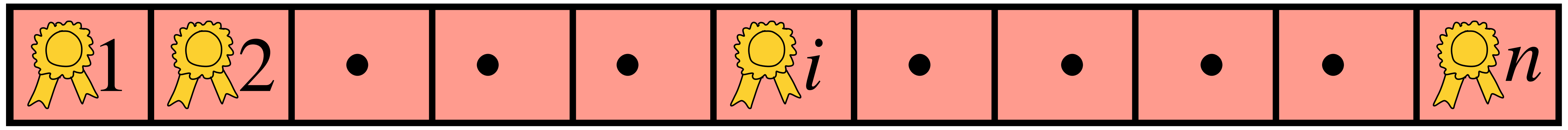


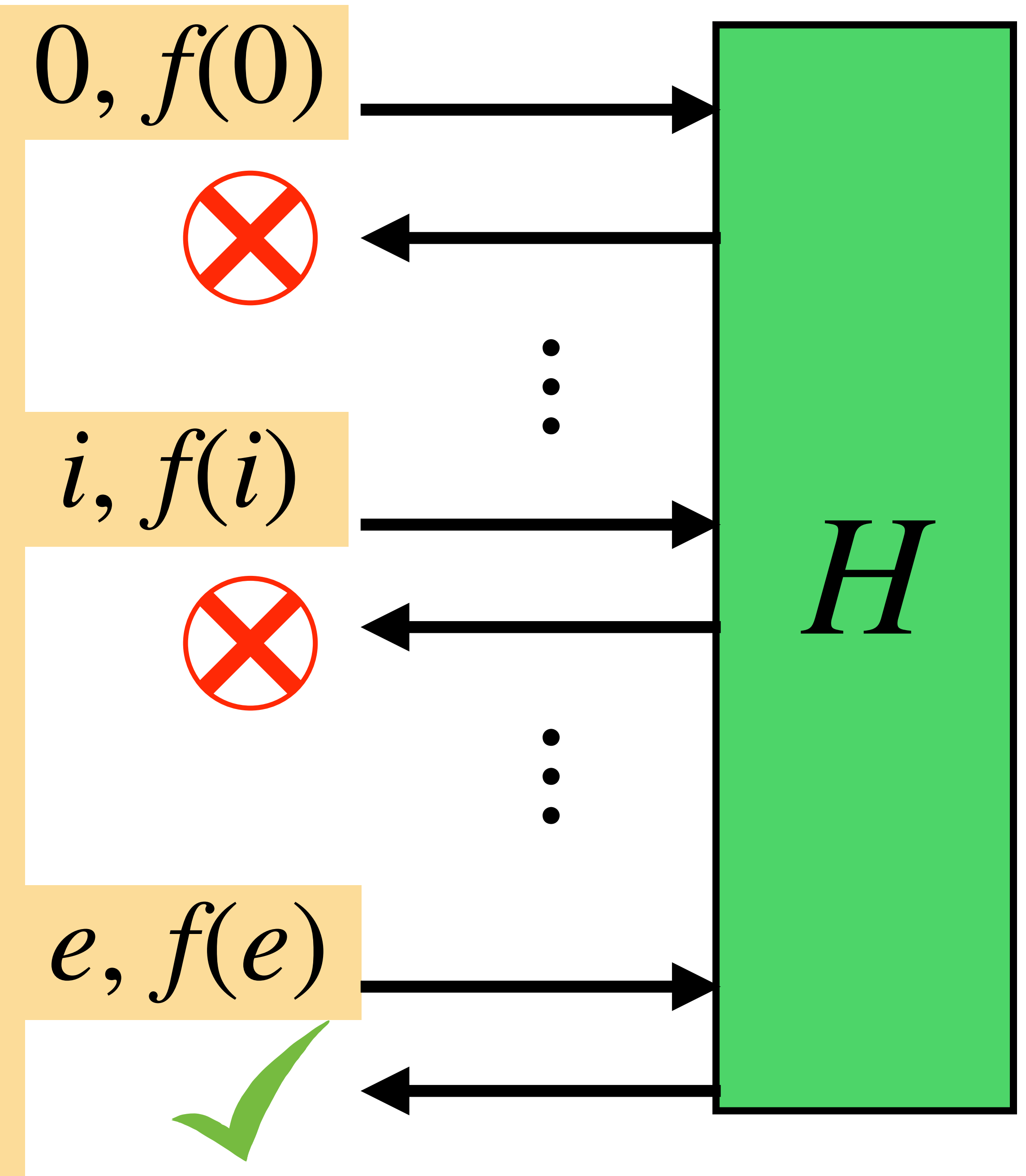
Improving $\mathsf{C}_{\mathsf{qry}}$

$\mathsf{C}_{\mathsf{qry}}$ in Schnorr aggregation Fischlin proof

$$P$$

$$f \in \mathbb{Z}_q[X]$$

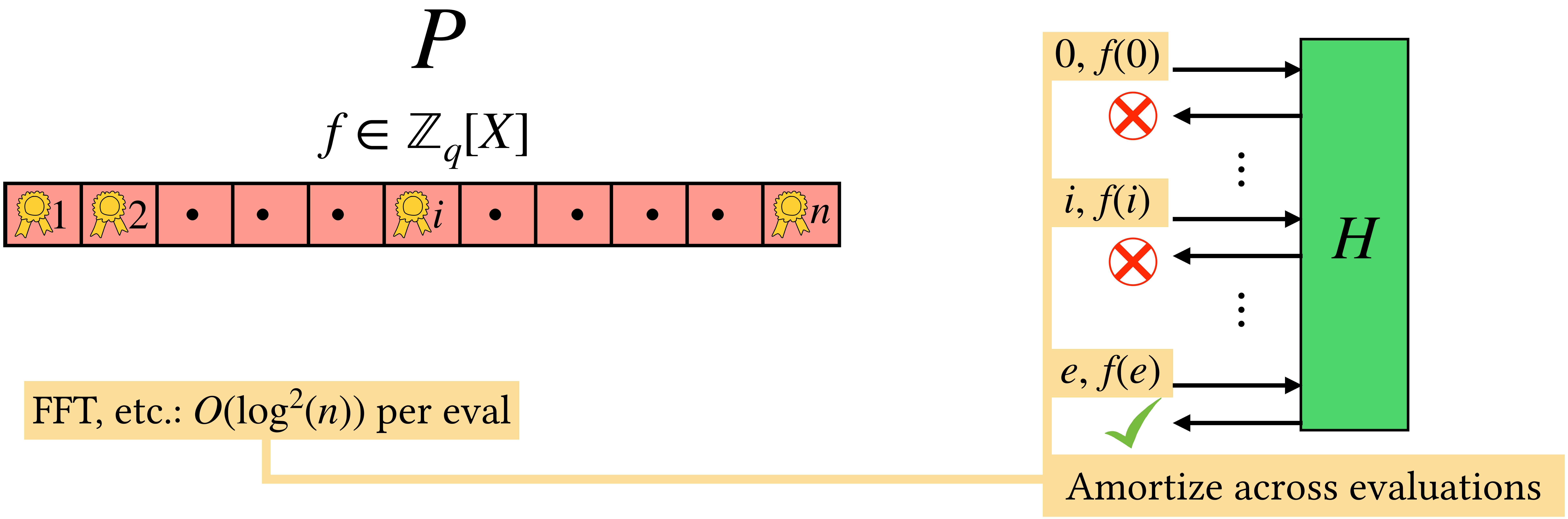




Amortize across evaluations

Improving C_{qry}

C_{qry} in Schnorr aggregation **Fischlin proof**



P

$f \in \mathbb{Z}_q[X]$

FFT, etc.: $O(\log^2(n))$ per eval

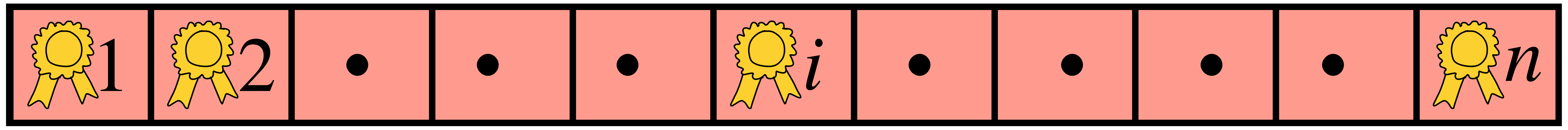
Amortize across evaluations

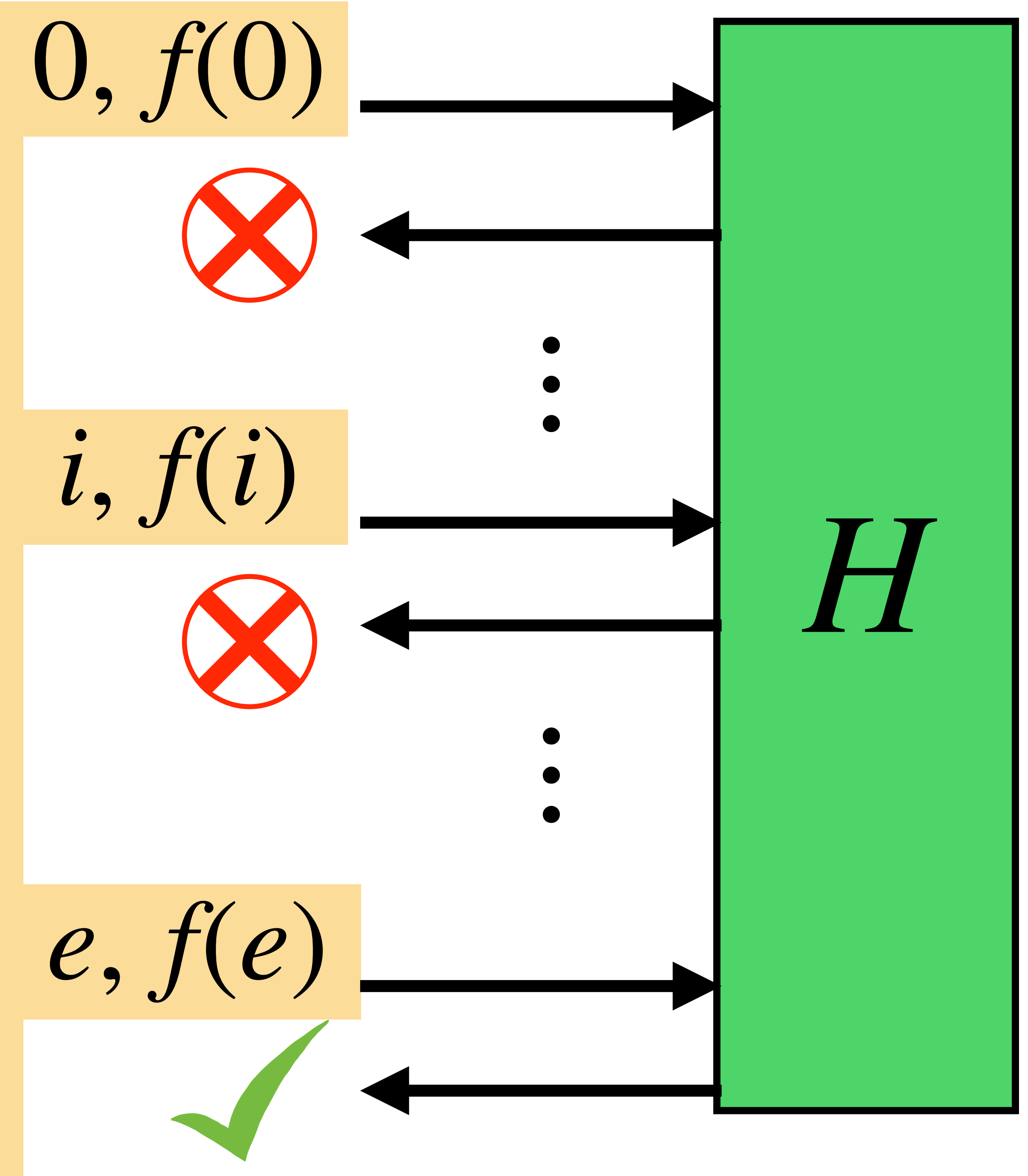
Improving $\mathsf{C}_{\mathsf{qry}}$

$\mathsf{C}_{\mathsf{qry}}$ in Schnorr aggregation Fischlin proof

$$P$$

$$f \in \mathbb{Z}_q[X]$$





FFT, etc.: $O(\log^2(n))$ per eval

Most signing curves incompatible with FFT

Amortize across evaluations

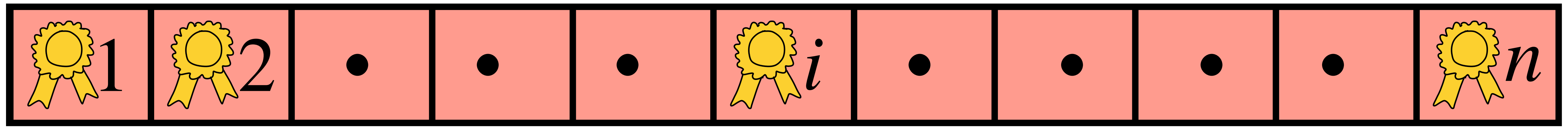
Asymptotically efficient general multipoint evaluation is unsatisfying for $n < 1000$

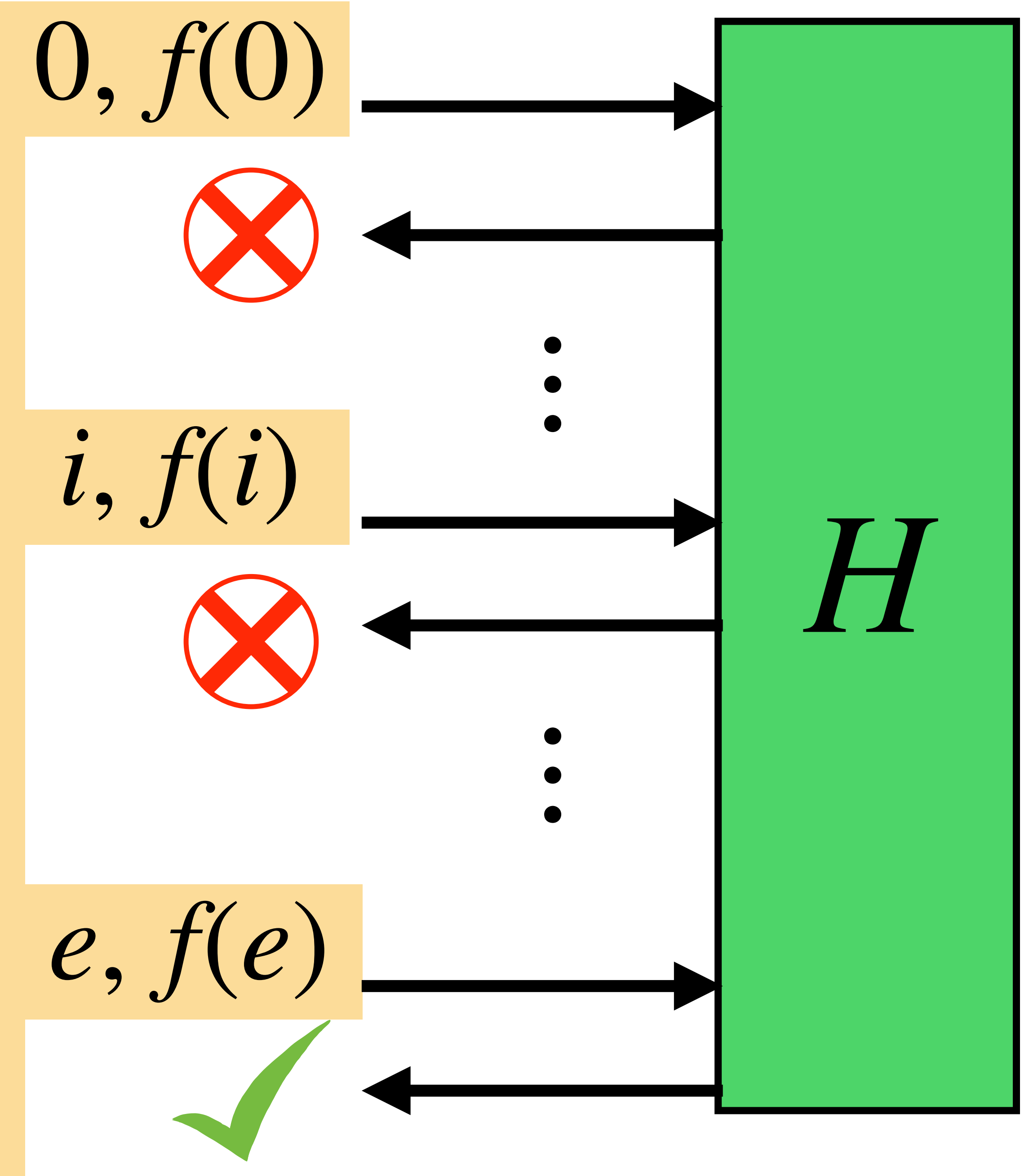
Improving C_{qry}

C_{qry} in Schnorr aggregation Fischlin proof

P

$f \in \mathbb{Z}_q[X]$





FFT, etc.: $O(\log^2(n))$ per eval

Most signing curves incompatible with FFT

Amortize across evaluations

Asymptotically efficient general multipoint evaluation is unsatisfying for $n < 1000$

This work: $2\sqrt{n}$ per eval

In Summary

- Fischlin's transform does not preserve Witness Indistinguishability in general — we show how randomization can fix this
- Lower bound explaining lack of progress in SLE in the ROM
    - We show that application-specific optimization is possible
    - Modest general improvement via hash collisions

Thanks!

eprint.iacr.org/2022/393

The Attack

- **Fact 3**: In some Sigma protocols, for the same (a, e), the response z will depend on which witness is used. e.g. PoK of w_0 OR w_1

Consider a given (a, e, z)

$\boxed{\text{Common } a}$

$P_{\mathsf{OR}}(w_0)$:

$P_{\mathsf{OR}}(w_1)$:

(e, z)

If $P_{\mathsf{OR}}(w_0)$ and $P_{\mathsf{OR}}(w_1)$ "agree" at e, then they "disagree" at any $e' \neq e$

The Attack

- **<u>Fact 3</u>**: In some Sigma protocols, for the same (a, e), the response z will depend on which witness is used. e.g. PoK of w_0 OR w_1

Consider a given (a, e, z)

$P_{\mathsf{OR}}(w_0):$

Common a

If $P_{\mathsf{OR}}(w_0)$ and $P_{\mathsf{OR}}(w_1)$ "agree" at e, then they "disagree" at any $e' \neq e$

$P_{\mathsf{OR}}(w_1):$

$(e\,,z)$

The Attack

- **Fact 3**: In some Sigma protocols, for the same (a, e), the response z will depend on which witness is used. e.g. PoK of w_0 OR w_1

Consider a given (a, e, z)

$P_{\mathsf{OR}}(w_0)$:

$\boxed{\text{Common } a}$

$(0,\ z_0)$

$(1,\ z_1)$

$\vdots$

$P_{\mathsf{OR}}(w_1)$:

(e, z)

If $P_{\mathsf{OR}}(w_0)$ and $P_{\mathsf{OR}}(w_1)$ “agree” at e, then they “disagree” at any $e' \neq e$

The Attack

- **Fact 3**: In some Sigma protocols, for the same (a, e), the response z will depend on which witness is used. e.g. PoK of w_0 OR w_1

Consider a given (a, e, z)

$P_{\mathsf{OR}}(w_0)$:

$(0, z_0)$

$(1, z_1)$

$\vdots$

$\boxed{\text{Common } a}$

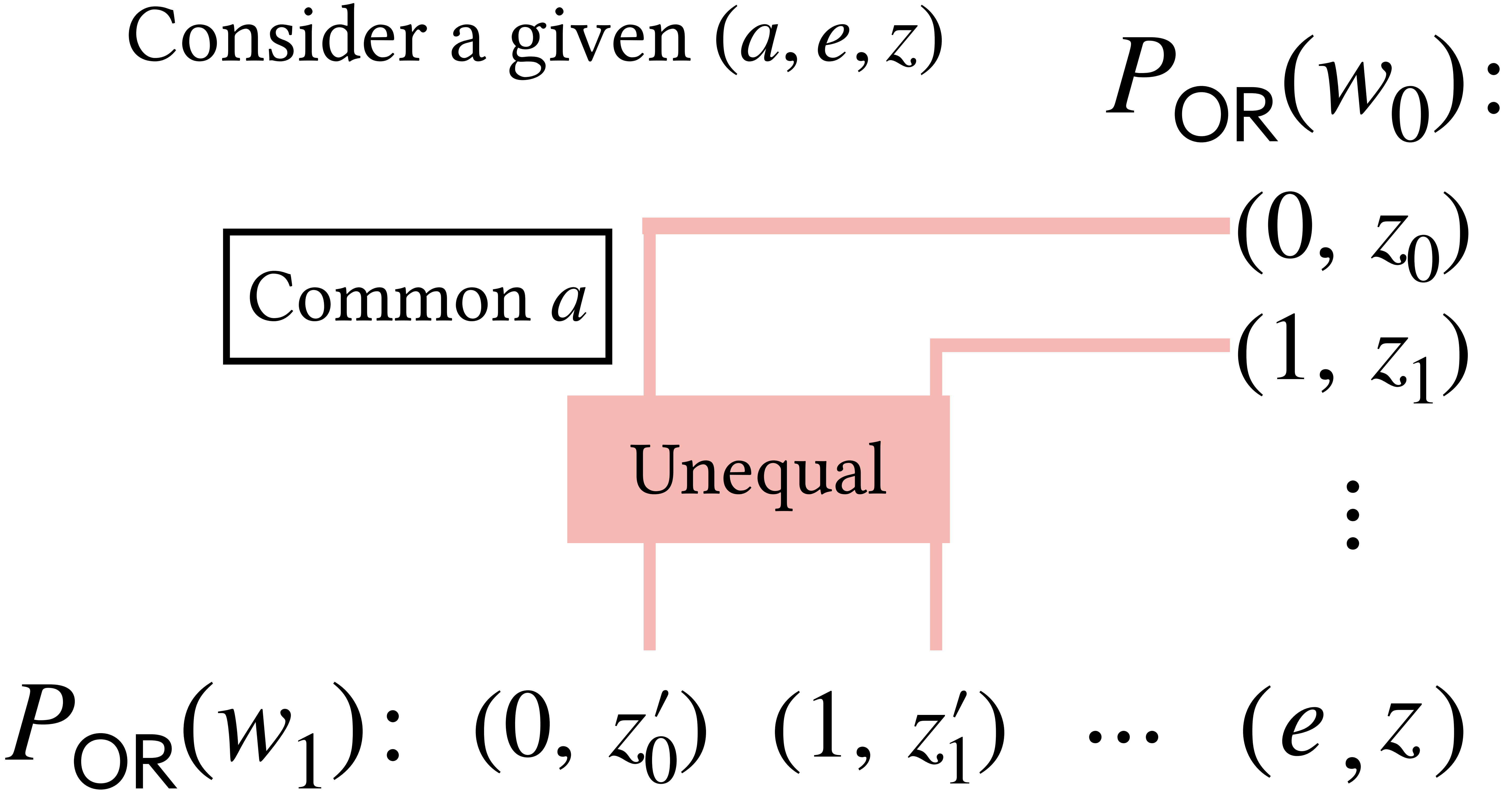
If $P_{\mathsf{OR}}(w_0)$ and $P_{\mathsf{OR}}(w_1)$ "agree" at e, then they "disagree" at any $e' \neq e$

$P_{\mathsf{OR}}(w_1)$: $(0, z_0')$ $(1, z_1')$ $\cdots$ (e, z)

The Attack

- <u>**Fact 3**</u>: In some Sigma protocols, for the same (a, e), the response z will depend on which witness is used. e.g. PoK of w_0 OR w_1

Consider a given (a, e, z)



$P_{\mathsf{OR}}(w_0):$

$(0,\ z_0)$

$(1,\ z_1)$

$\vdots$

Common a

Unequal

$P_{\mathsf{OR}}(w_1):\ (0,\ z_0')\ \ (1,\ z_1')\ \ \cdots\ \ (e\,,z\,)$

If $P_{\mathsf{OR}}(w_0)$ and $P_{\mathsf{OR}}(w_1)$ "agree" at e, then they "disagree" at any $e' \neq e$

The Attack

- **Fact 3**: In some Sigma protocols, for the same (a, e), the response z will depend on which witness is used. e.g. PoK of w_0 OR w_1

Consider a given (a, e, z)

Given (a, e, z) produced by Fischlin's compiler, we can test which path is "plausible"

$P_{\mathsf{OR}}(w_0)$:

$(0,\ z_0)$

$(1,\ z_1)$

$\vdots$

$P_{\mathsf{OR}}(w_1)$: $(0,\ z_0')$ $(1,\ z_1')$ $\cdots$ $(e\ ,z\ )$

The Attack

- **Fact 3**: In some Sigma protocols, for the same (a, e), the response z will depend on which witness is used. e.g. PoK of w_0 OR w_1

Consider a given (a, e, z)

Given (a, e, z) produced by Fischlin's compiler, we can test which path is "plausible"

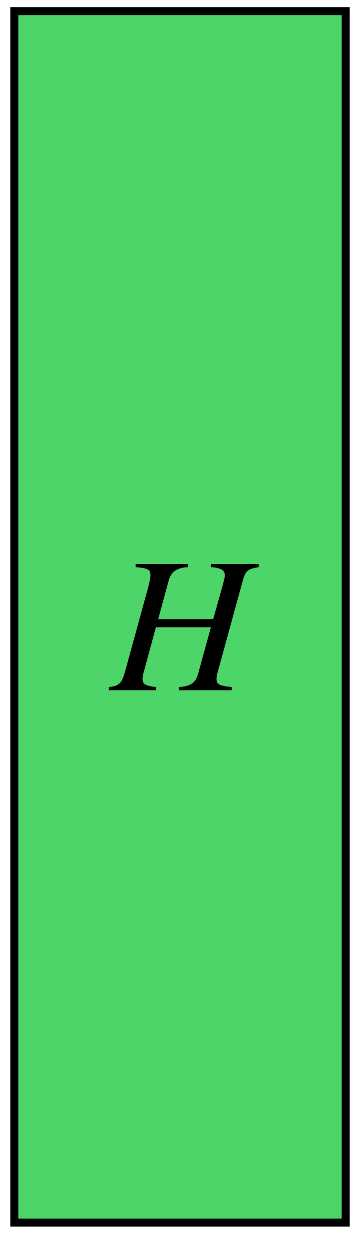
$P_{\mathsf{OR}}(w_0):$

$(0,\ z_0)$

$(1,\ z_1)$

$\vdots$

(e, z)



$P_{\mathsf{OR}}(w_1):$ $(0,\ z_0')$ $(1,\ z_1')$ $\cdots$

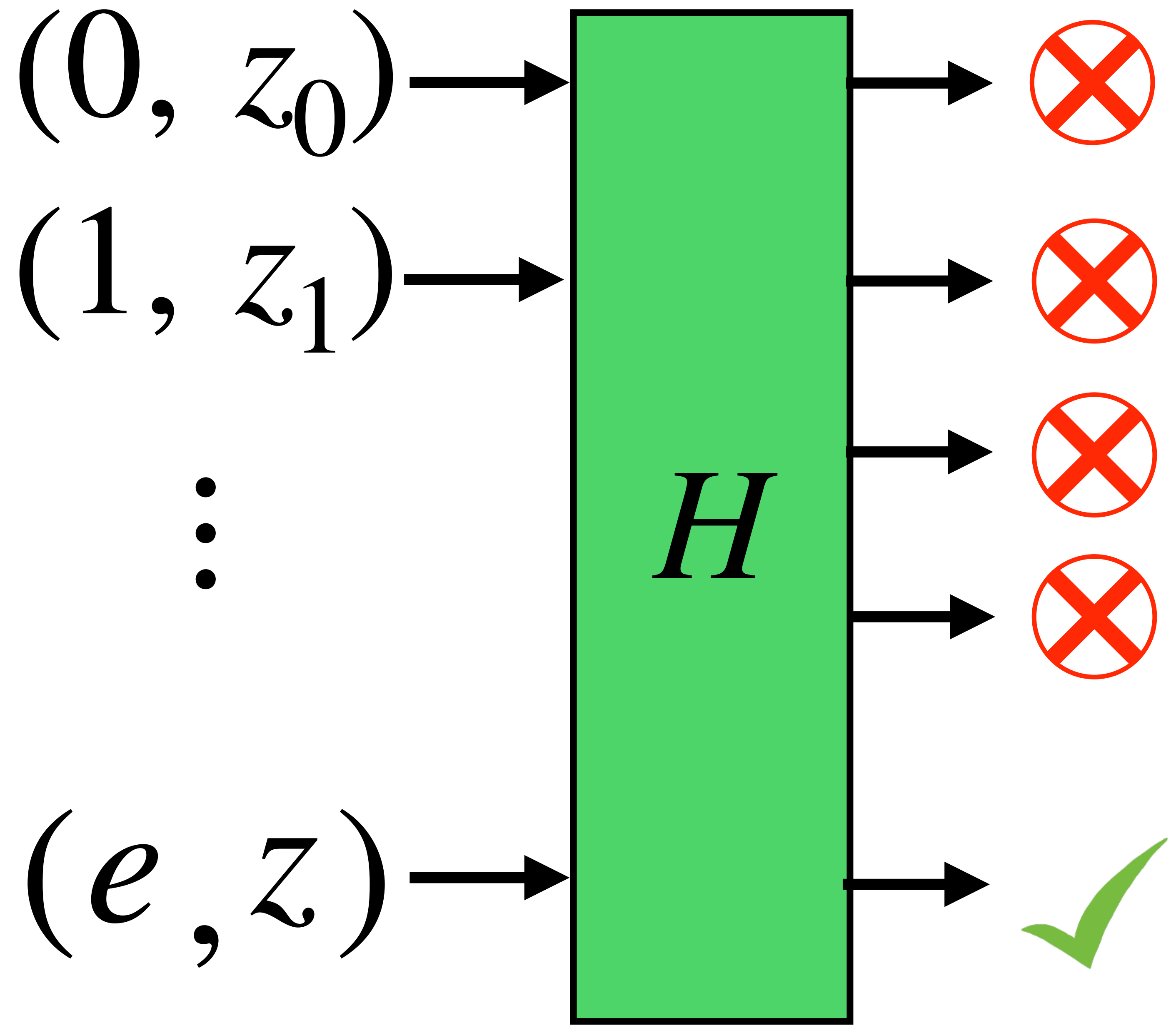
The Attack

- **Fact 3**: In some Sigma protocols, for the same (a, e), the response z will depend on which witness is used. e.g. PoK of w_0 OR w_1

Consider a given (a, e, z)

Given (a, e, z) produced by Fischlin's compiler, we can test which path is "plausible"

$P_{\text{OR}}(w_0):$



$(0,\ z_0) \rightarrow H \rightarrow$ ⊗

$(1,\ z_1) \rightarrow H \rightarrow$ ⊗

$\vdots$

$(e, z) \rightarrow H \rightarrow$ ✓

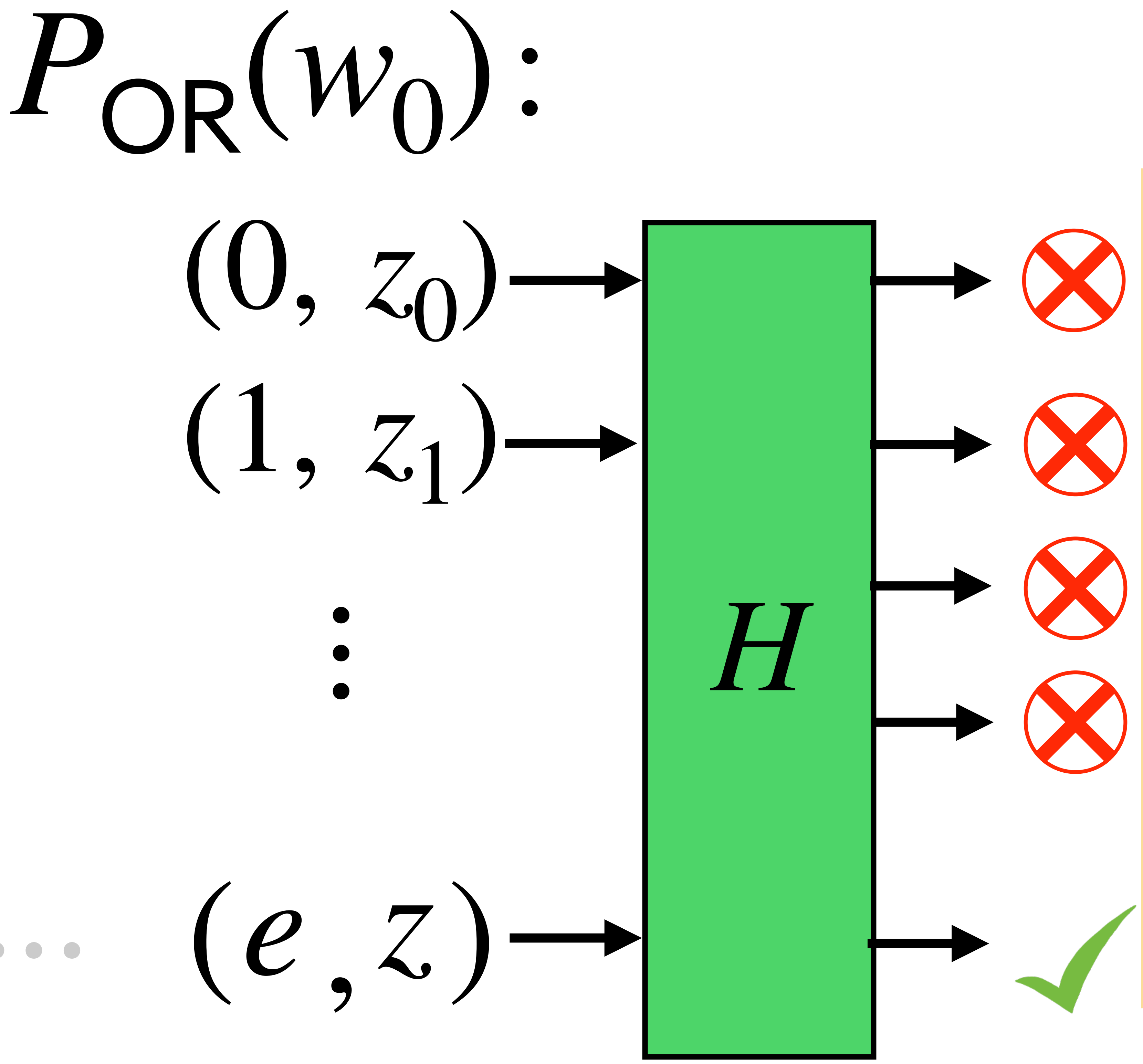
$P_{\text{OR}}(w_1):\ (0,\ z_0')\ \ (1,\ z_1')\ \cdots$

The Attack

- **Fact 3**: In some Sigma protocols, for the same (a, e), the response z will depend on which witness is used. e.g. PoK of w_0 OR w_1

Consider a given (a, e, z)

Given (a, e, z) produced by Fischlin's compiler, we can test which path is "plausible"



$P_{\mathsf{OR}}(w_1)$: $(0,\ z_0')$ $(1,\ z_1')$ $\cdots$

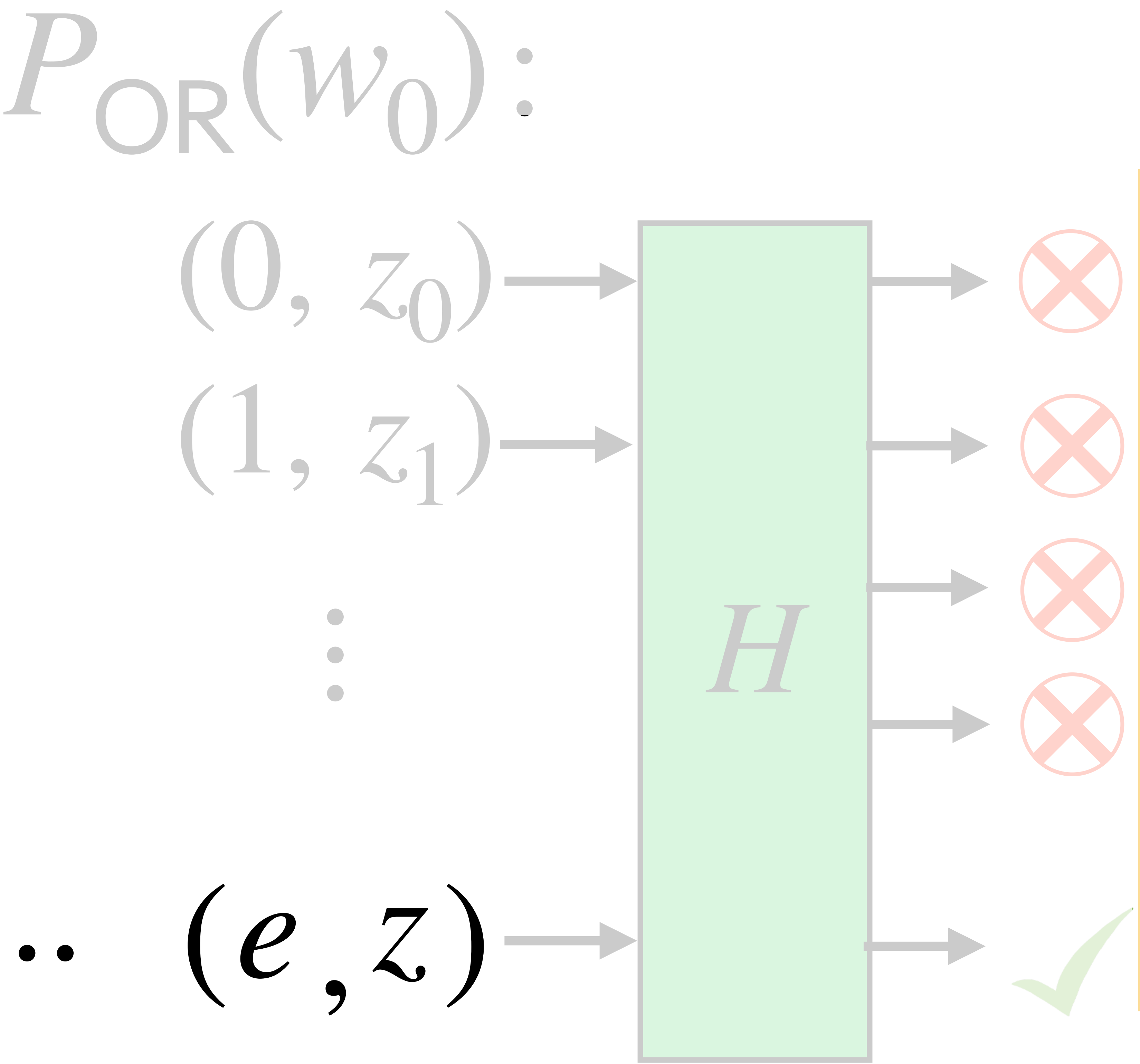
W.h.p., only one path—induced by one of $P_{\mathsf{OR}}(w_0)$ or $P_{\mathsf{OR}}(w_1)$—is plausible

The Attack

- **<u>Fact 3</u>**: In some Sigma protocols, for the same (a, e), the response z will depend on which witness is used. e.g. PoK of w_0 OR w_1

Consider a given (a, e, z)

Given (a, e, z) produced by Fischlin's compiler, we can test which path is "plausible"



$$P_{\mathsf{OR}}(w_1): \ (0, z_0') \ (1, z_1') \ \cdots \ (e, z)$$

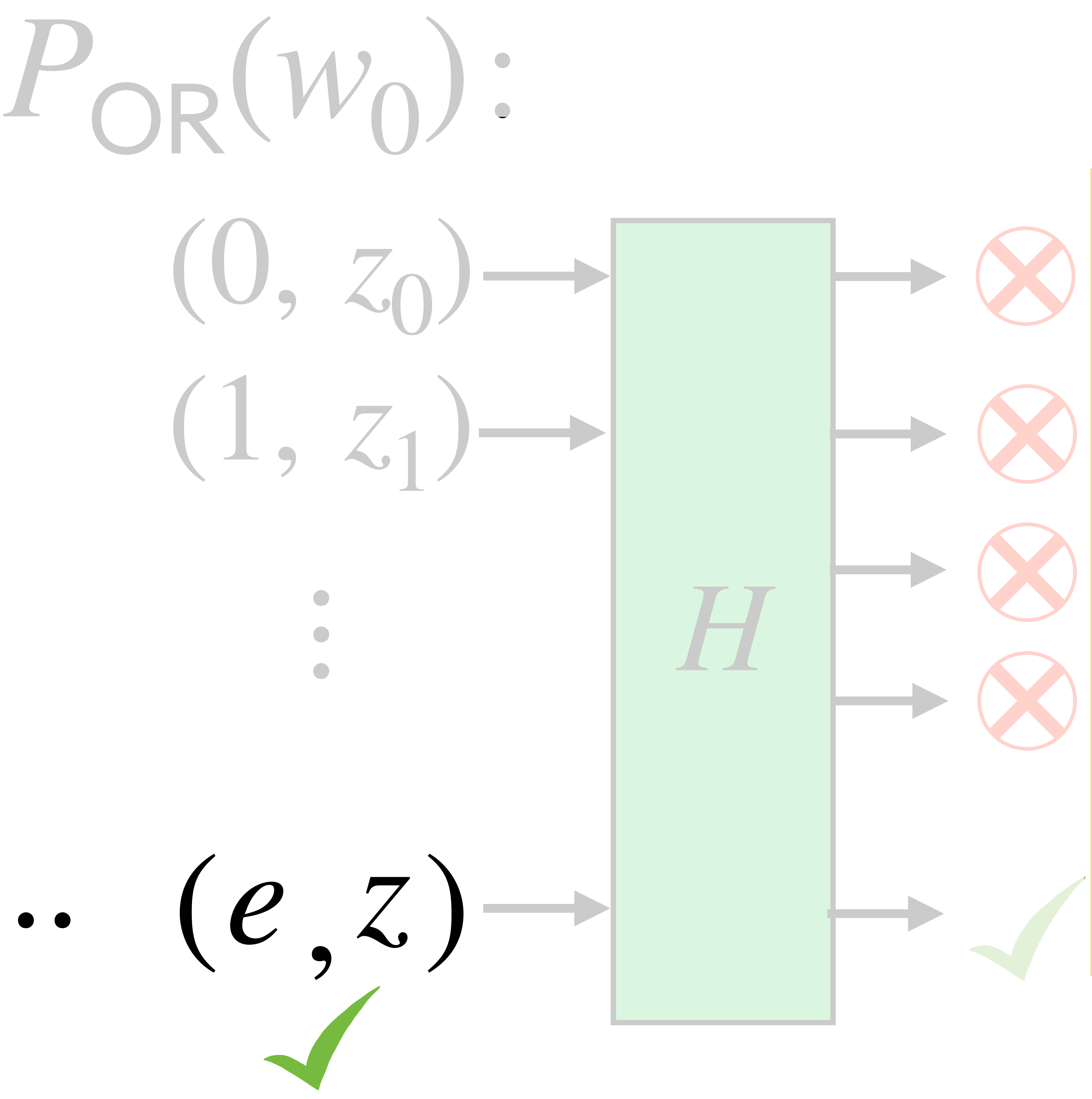
W.h.p., only one path—induced by one of $P_{\mathsf{OR}}(w_0)$ or $P_{\mathsf{OR}}(w_1)$—is plausible

The Attack

- **<u>Fact 3</u>**: In some Sigma protocols, for the same (a, e), the response z will depend on which witness is used. e.g. PoK of w_0 OR w_1

Consider a given (a, e, z)

Given (a, e, z) produced by Fischlin's compiler, we can test which path is "plausible"



$P_{\mathsf{OR}}(w_0)$:

$(0,\ z_0)$ → H → ⊗

$(1,\ z_1)$ → H → ⊗

$\vdots$

⊗

⊗

$P_{\mathsf{OR}}(w_1)$: $(0,\ z_0')$ $(1,\ z_1')$ $\cdots$ $(e\ ,z)$ → H → ✓

✓

W.h.p., only one path—induced by one of $P_{\mathsf{OR}}(w_0)$ or $P_{\mathsf{OR}}(w_1)$—is plausible

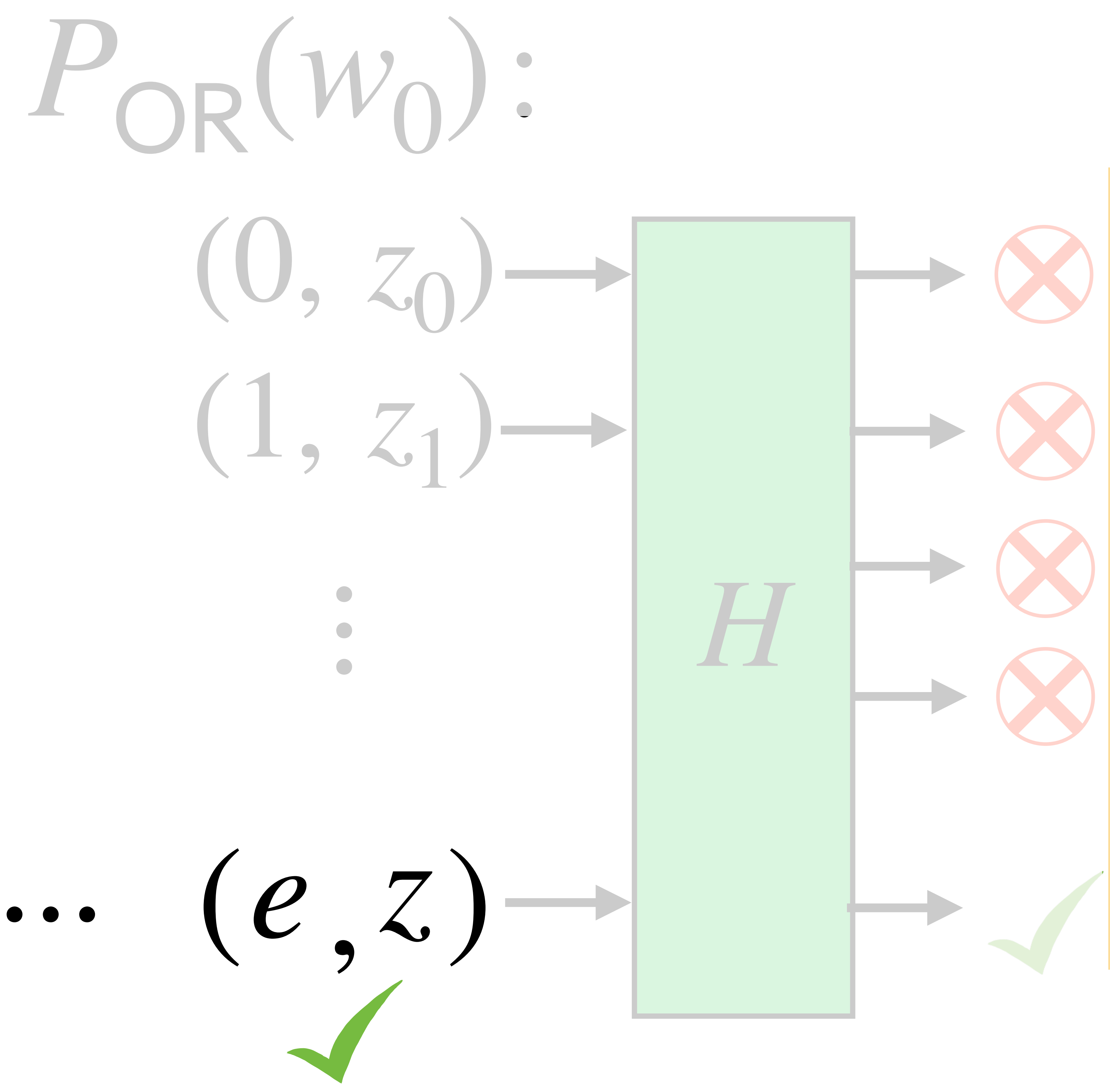
The Attack

- **Fact 3**: In some Sigma protocols, for the same (a, e), the response z will depend on which witness is used. e.g. PoK of w_0 OR w_1

Consider a given (a, e, z)

Given (a, e, z) produced by Fischlin's compiler, we can test which path is "plausible"

This path induces **fresh** queries to H



$P_{\mathsf{OR}}(w_0)$:

$(0,\ z_0)$

$(1,\ z_1)$

$\vdots$

$P_{\mathsf{OR}}(w_1)$: $(0,\ z_0')$ $(1,\ z_1')$ $\cdots$ $(e\ ,z)$ ✓

W.h.p., only one path—induced by one of $P_{\mathsf{OR}}(w_0)$ or $P_{\mathsf{OR}}(w_1)$—is plausible

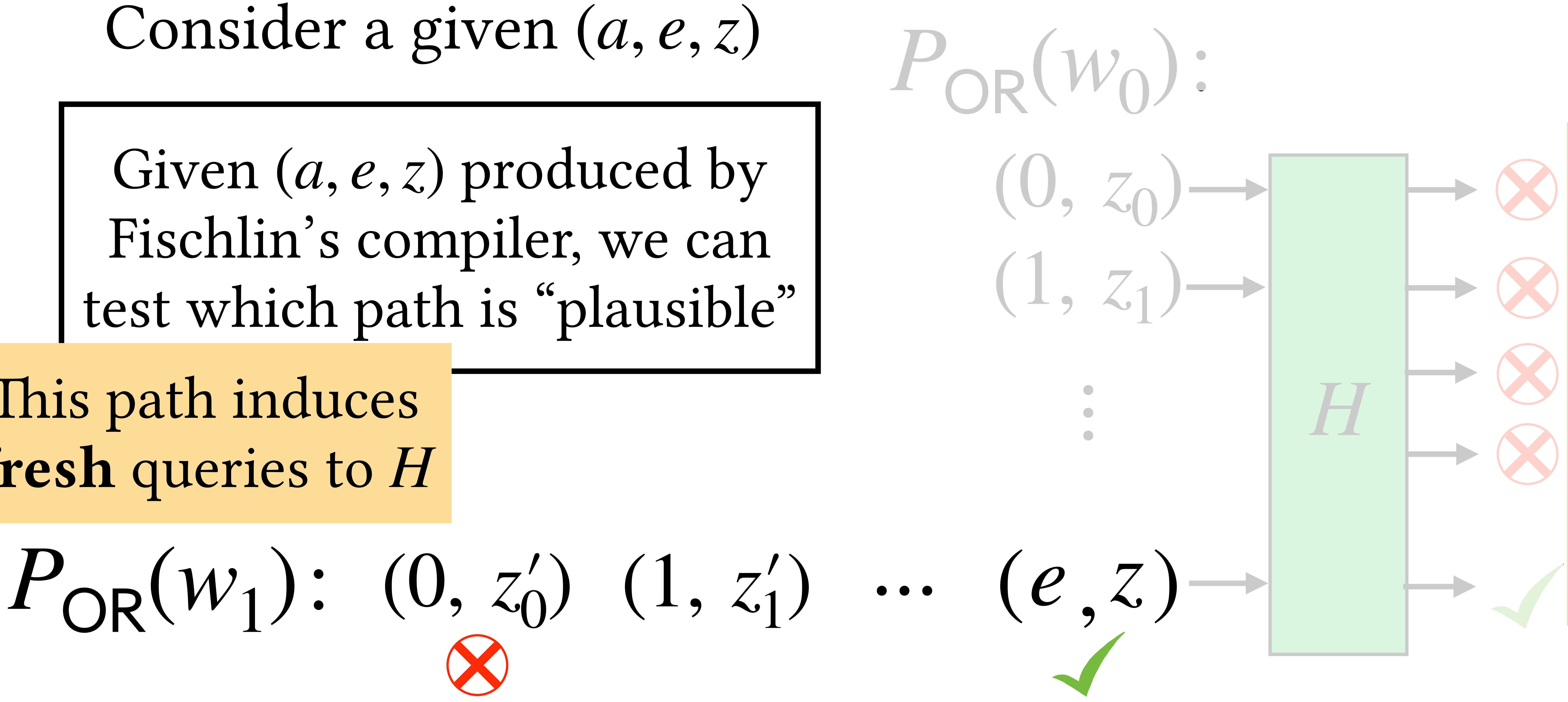
The Attack

- **<u>Fact 3</u>**: In some Sigma protocols, for the same (a, e), the response z will depend on which witness is used. e.g. PoK of w_0 OR w_1

Consider a given (a, e, z)

Given (a, e, z) produced by Fischlin's compiler, we can test which path is "plausible"

This path induces **fresh** queries to H



$P_{\mathsf{OR}}(w_0):$

$(0,\ z_0)$

$(1,\ z_1)$

$\vdots$

H

$P_{\mathsf{OR}}(w_1):\quad (0,\ z_0') \quad (1,\ z_1') \quad \cdots \quad (e\ ,z)$

W.h.p., only one path—induced by one of $P_{\mathsf{OR}}(w_0)$ or $P_{\mathsf{OR}}(w_1)$—is plausible

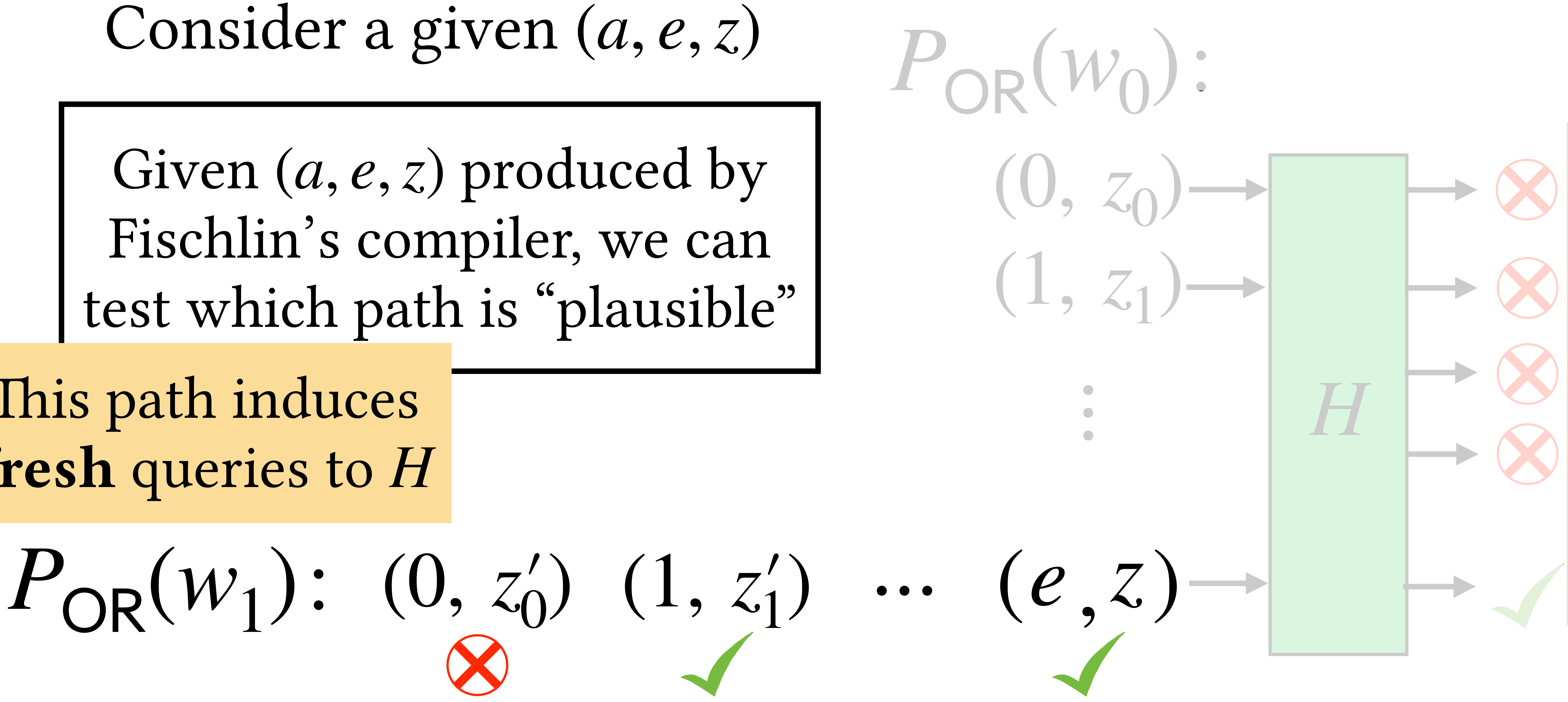
The Attack

- **<u>Fact 3</u>**: In some Sigma protocols, for the same (a, e), the response z will depend on which witness is used. e.g. PoK of w_0 OR w_1

Consider a given (a, e, z)

Given (a, e, z) produced by Fischlin's compiler, we can test which path is "plausible"

This path induces **fresh** queries to H



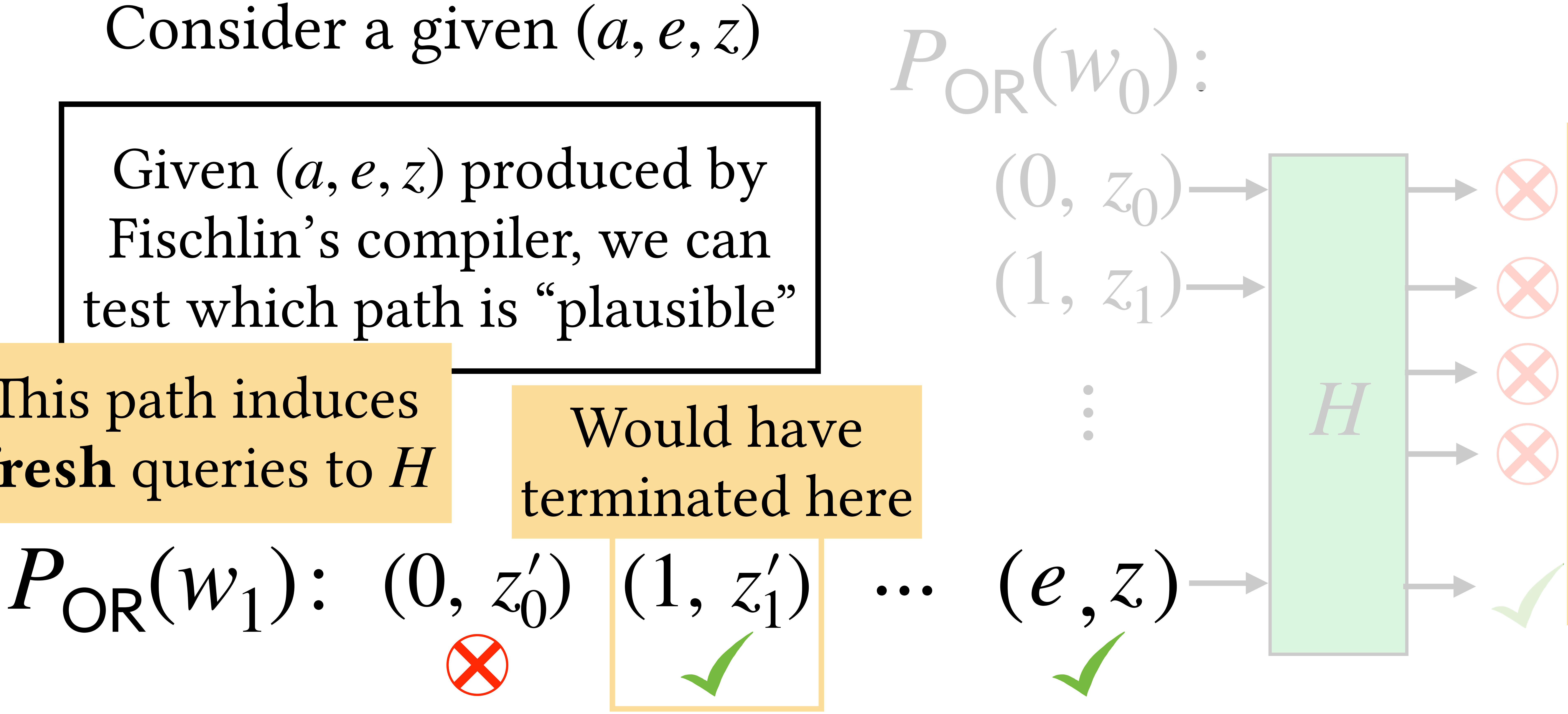
W.h.p., only one path—induced by one of $P_{\mathsf{OR}}(w_0)$ or $P_{\mathsf{OR}}(w_1)$—is plausible

The Attack

- **<u>Fact 3</u>**: In some Sigma protocols, for the same (a, e), the response z will depend on which witness is used. e.g. PoK of w_0 OR w_1

Consider a given (a, e, z)

Given (a, e, z) produced by Fischlin's compiler, we can test which path is "plausible"



$P_{\mathsf{OR}}(w_0)$:

$(0,\ z_0)$ ⊗

$(1,\ z_1)$ ⊗

$\vdots$

H

This path induces **fresh** queries to H

Would have terminated here

$P_{\mathsf{OR}}(w_1)$: $(0,\ z_0')$ ⊗ $(1,\ z_1')$ ✓ $\cdots$ (e, z) ✓

W.h.p., only one path—induced by one of $P_{\mathsf{OR}}(w_0)$ or $P_{\mathsf{OR}}(w_1)$—is plausible