Sometimes You Can't Distribute Random-Oracle-Based Proofs ¯_(ツ)_/¯

Jack Doerner

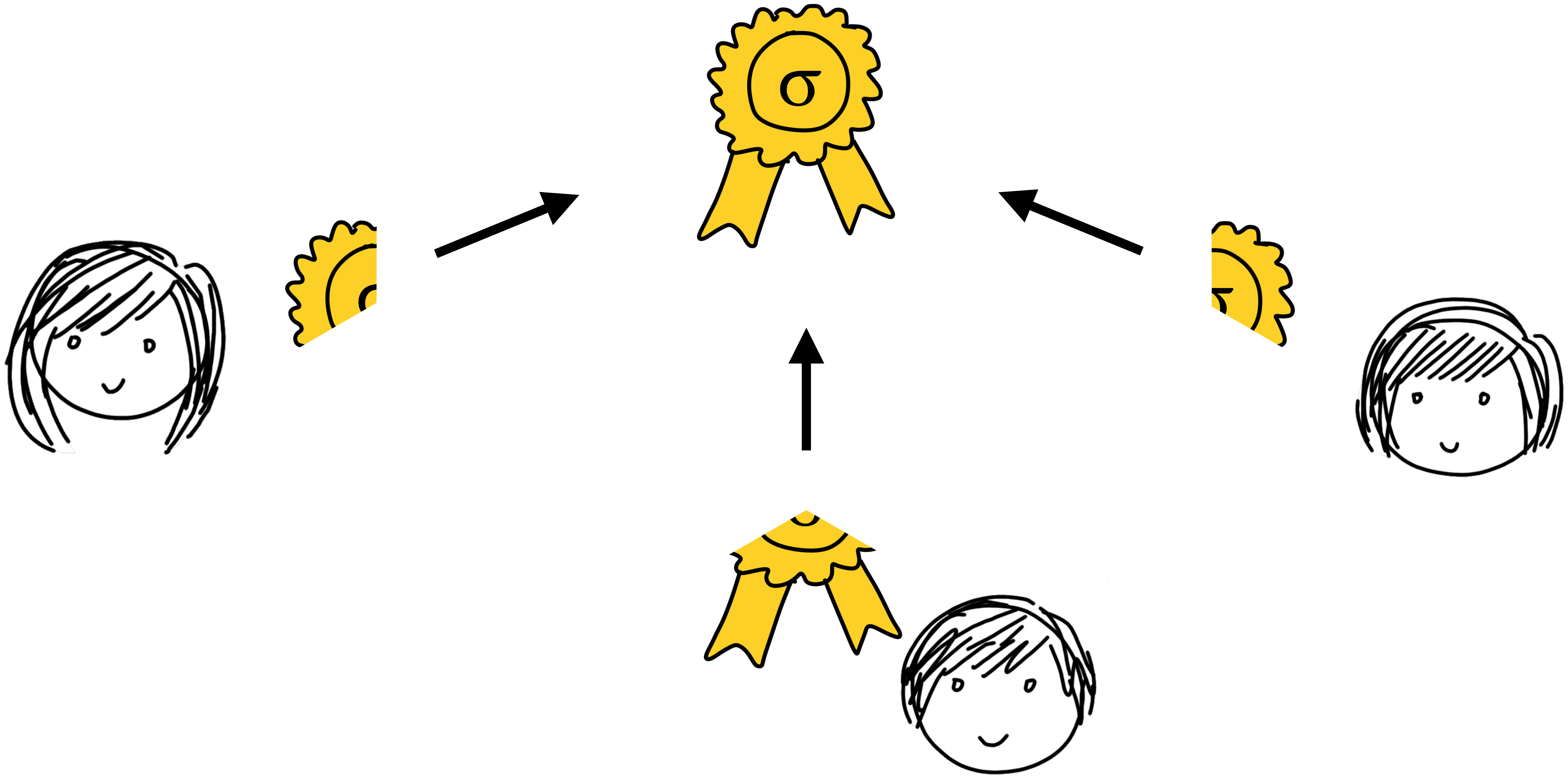
Yashvanth Kondi

Leah Namisa Rosenbloom

eprint.iacr.org/2023/1381

NIST Workshop on Multi-Party Threshold Schemes, September 27 2023

Threshold Signing: What We Want



Threshold Signing: What We Want

- **Compatibility**:
  Verifies w.r.t. original algorithm

Threshold Signing: What We Want

- **Compatibility**:
  Verifies w.r.t. original algorithm
- **Corruption Resilience**:
  Compromising some devices does not leak the signing key

Threshold Signing: What We Want

- **Compatibility**:
  Verifies w.r.t. original algorithm
- **Corruption Resilience**:
  Compromising some devices does not leak the signing key
- **Efficiency**:
  Wall clock time similar to single party signing
  Bandwidth not too high

Achieving "Efficiency"

- Any signing scheme can be distributed via general MPC

- "Practical" efficiency usually requires more fine-grained notions than just feasibility

- One good proxy: practical threshold signing makes **black-box use** of non-linear components of the signing algorithm:
  - Integer arithmetic in $\mathbb{Z}_q$ or $\mathbb{Z}_N^*$
  - Elliptic curve group operations
  - Hash functions

Threshold schemes for RSA, Schnorr/EdDSA, ECDSA, BLS, BBS+ achieve this!

The Magic is in the Hash Function

Specifically: security analysis based on **Straight-Line-Extraction** (SLE) in the **Random Oracle Model** (ROM)

Concurrently-Secure Non-Interactive Zero-Knowledge (NIZK) Techniques

The Magic is in the Hash Function

Concurrently-Secure Non-Interactive
Zero-Knowledge (NIZK) Techniques

This talk: Signatures ⇔ NIZK

 $\Leftrightarrow \pi$

( , ) $\Leftrightarrow (x, w)$

Distributed Signing ⇔ Distributed Proving

The Magic is in the Hash Function

Concurrently-Secure Non-Interactive
Zero-Knowledge (NIZK) Techniques

- MPC-in-the-Head
- PCPs/IOPs
- Sigma Protocol + Fischlin/Unruh

Tight Security   Post Quantum Security

We Prove Limitations

- For some hash based NIZKs[1], there is an inherent barrier[2] to designing practical protocols[3] to distribute their computation.

1. NIZKs that have straight-line extractors in the Random-Oracle Model, and Verifiers that are agnostic to prover count
2. Attack that completely recovers the witness by corrupting all-but-one distributed provers
3. Protocol that is black-box in the same hash function (i.e. Random Oracle) as the NIZK

Implications

For NIZKs/Signatures based on

- MPC-in-the-Head
- PCPs/IOPs
- Sigma Protocol + Fischlin/Unruh

We cannot hope to achieve all three:

- Compatibility
- Corruption Resilience
- Black-box Use of Nonlinear Functions

Table Stakes for RSA, Schnorr/EdDSA, ECDSA, BLS, BBS+, etc.

NIZKPoK

Non-Interactive Zero-Knowledge Proof of Knowledge

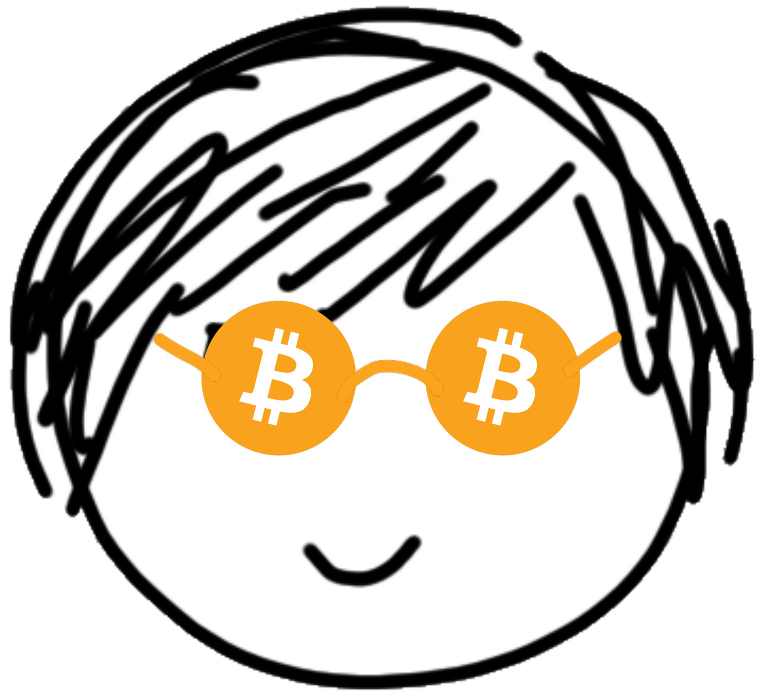
NIZKPoK

Non-Interactive: only one message is sent

NIZKPoK

Zero-Knowledge:
nothing about leaks

Bob



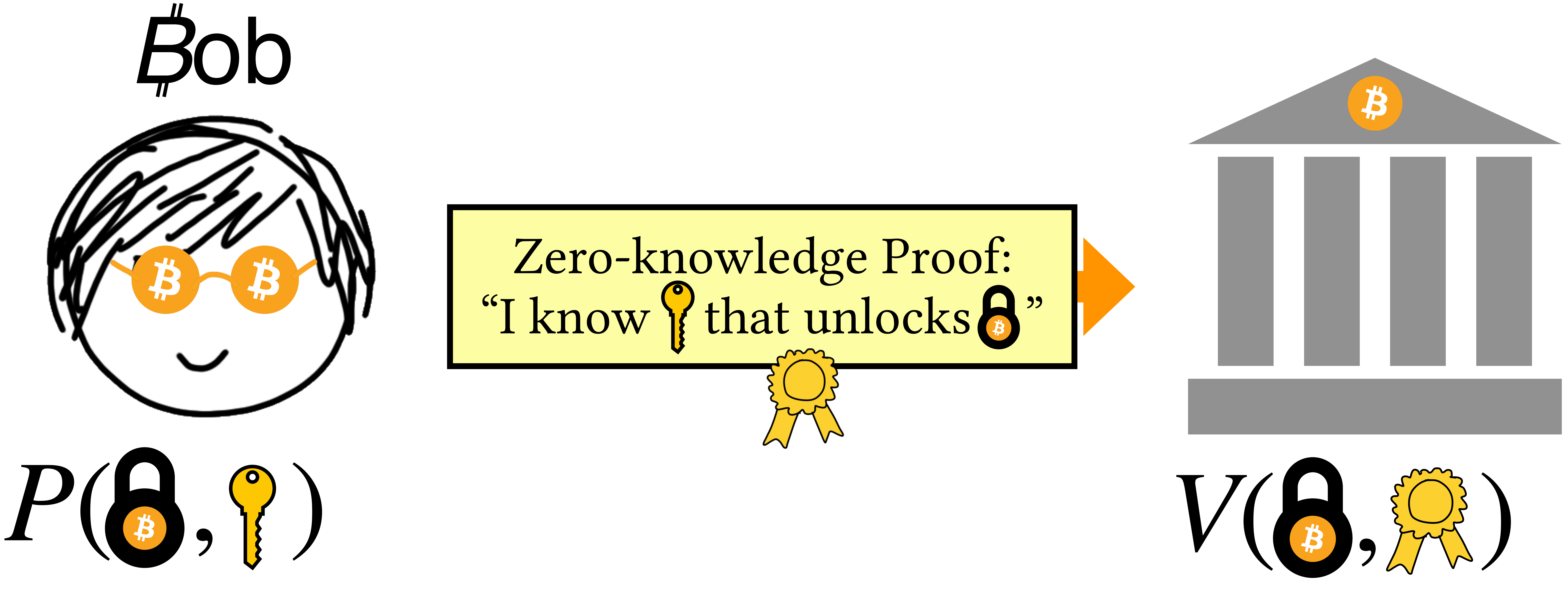
Zero-knowledge Proof:
"I know that unlocks "

$P($, $)$

$V($, $)$

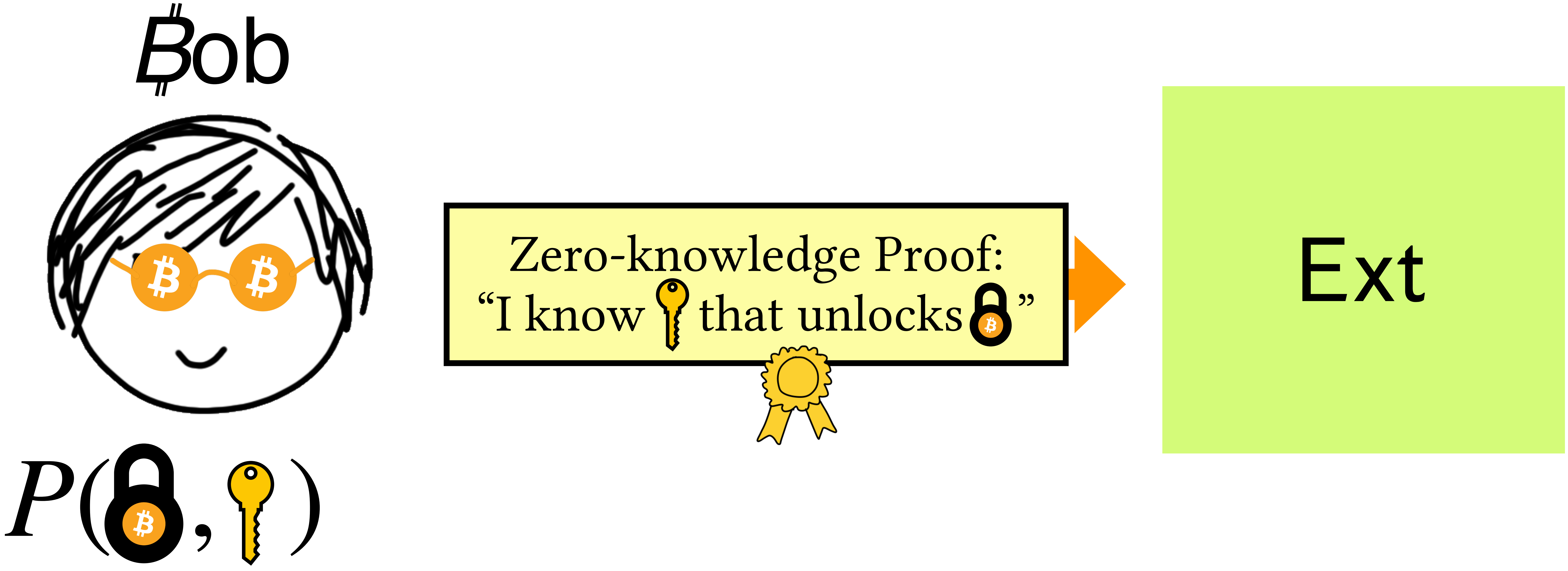
NIZKPoK

But what does it mean to *know* something?



NIZKPoK

Proof of Knowledge
is formalized by *Extraction*



NIZKPoK

Proof of Knowledge
is formalized by *Extraction*

$$\Pr\left[V(\;,\;) = 1\right] \approx \Pr\left[\mathsf{Ext}(\;,\;) = \;\right]$$

Over the coins of $P(\;,\;)$

How is Ext Special?

- Ext cannot be an algorithm that *anybody* can run
- Ext has carefully chosen special privileges:
  - Powerful enough to accomplish extraction
  - Still meaningful as a security claim
- Common special privilege: the ability to *rewind* time for the prover and *fork* the proof protocol

How is Ext Special?

- Ext cannot be an algorithm that *anybody* can run
- Ext has carefu
  - Powerful en
  - Still meanin

Bad news for:

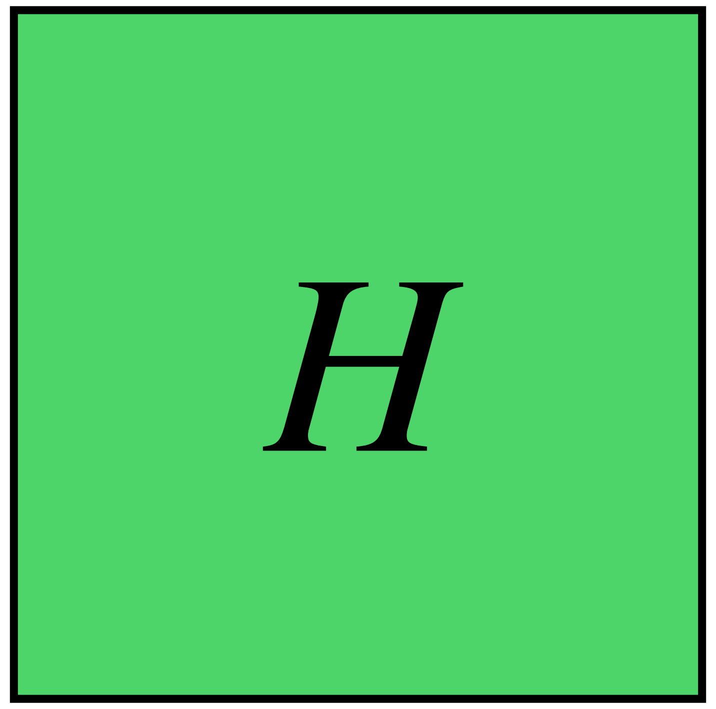
- Composability
- Tightness
- Post Quantum Security

- Common special privilege: the ability to *rewind* time for the prover and *fork* the proof protocol

How is Ext Special?

- Ext cannot be an algorithm that *anybody* can run
- Ext has carefully chosen special privileges:
  - Powerful enough to accomplish extraction
  - Still meaningful as a security claim
- **Straight-line Extraction** (SLE): no rewinding. Instead, use other trapdoor like CRS, RO, etc.

Random Oracle Model

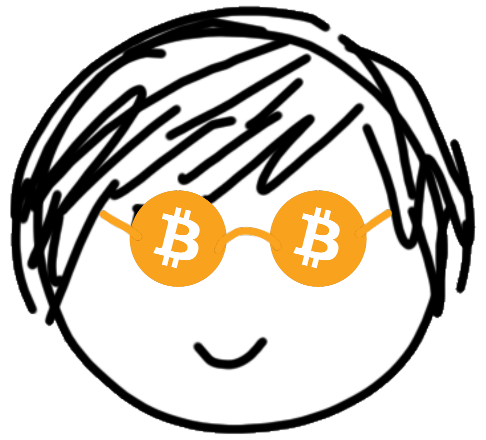


$$H : \{0,1\}^* \mapsto \{0,1\}^\ell$$

Random Oracles as Ext Privilege

$\boxed{H}$

$H : \{0,1\}^* \mapsto \{0,1\}^\ell$

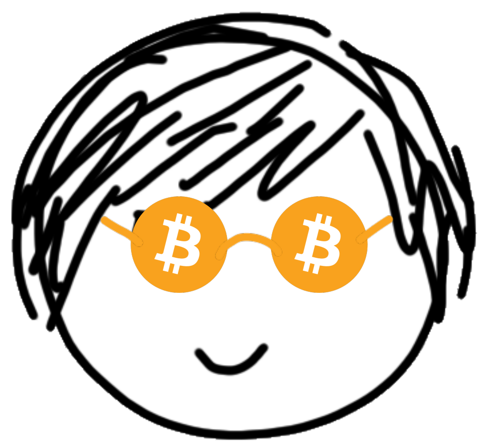


$P($$,$$)$

Random Oracles as Ext Privilege

H

$H : \{0,1\}^* \mapsto \{0,1\}^\ell$

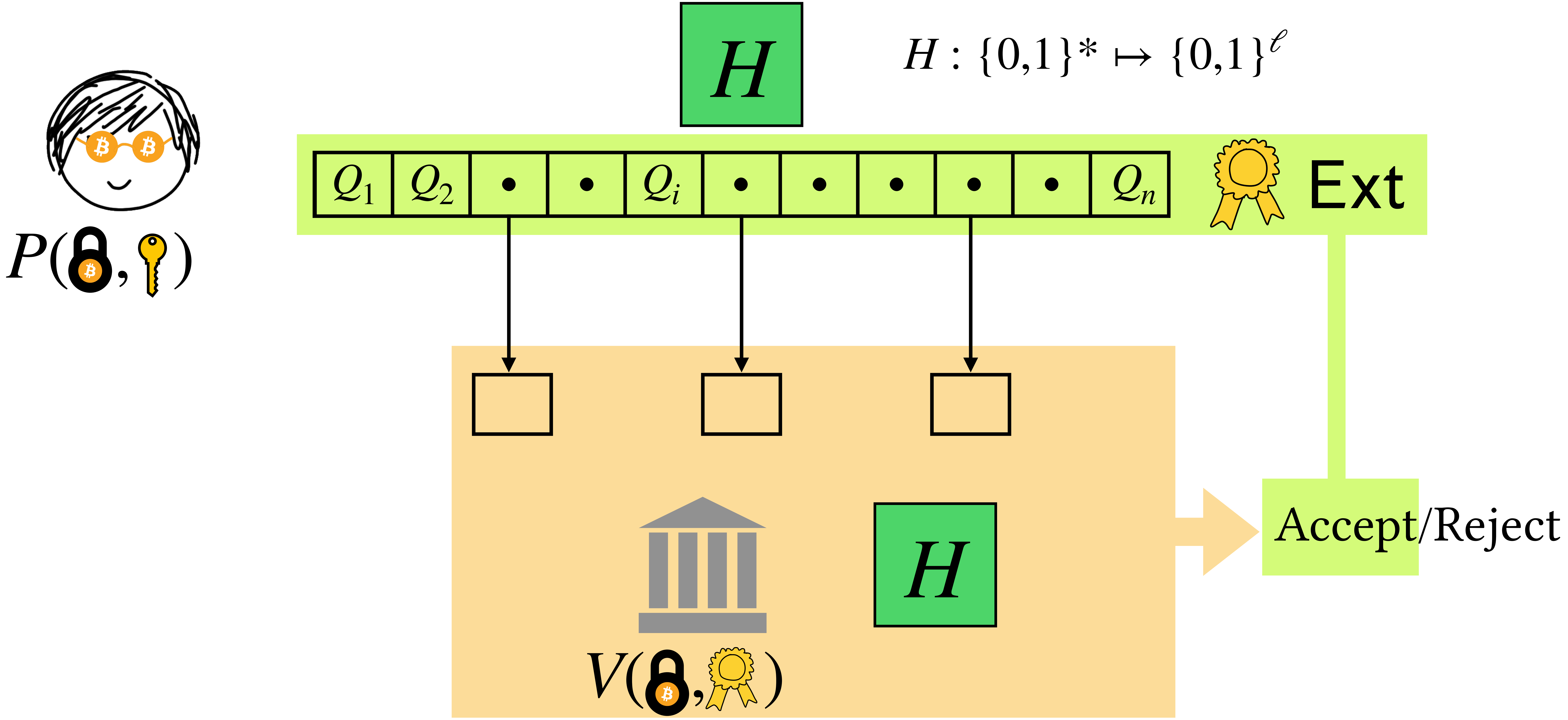


Q_1	Q_2	$\bullet$	$\bullet$	Q_i	$\bullet$	$\bullet$	$\bullet$	$\bullet$	$\bullet$	Q_n

$P($, $)$

$V($, $)$

Random Oracles as Ext Privilege



Random Oracles as Ext Privilege

- Why is it a meaningful trapdoor?
  - Hash functions are complex and highly unstructured
  - Bob must "query" each Q_i to H to obtain $H(Q_i)$
  - Ext gets $\{Q_i\}$ without rewinding
- Practical usage:
  - No "trusted setup", each query is very cheap

Distributing NIZKs in the ROM

- Multiparty protocols to securely compute RO-based NIZKs should ideally make black-box use of H
  - Conceptually: H should not have a circuit description
  - Practically: hash functions have large circuits
- We such protocols “Oracle Respecting Distributed” (ORD) Provers

Trivial Oracle Respecting Distribution

$$\pi \leftarrow P(x, w) \qquad V(x, \pi) = 1$$

Consider languages where (x, w) can be "secret shared":

$x_0 + x_1 + x_2 = x \qquad w_0 + w_1 + w_2 = w \qquad$ (think DLog)

$(x_0, w_0), (x_1, w_1), (x_2, w_2) \in L \Leftrightarrow (x, w) \in L$

Trivial Oracle Respecting Distribution

$$\pi \leftarrow P(x, w) \qquad V(x, \pi) = 1$$

Consider languages where (x, w) can be "secret shared":

$x_0 + x_1 + x_2 = x \qquad w_0 + w_1 + w_2 = w \qquad$ (think DLog)

$(x_0, w_0), (x_1, w_1), (x_2, w_2) \in L \Leftrightarrow (x, w) \in L$

$P^3(x, w)$:

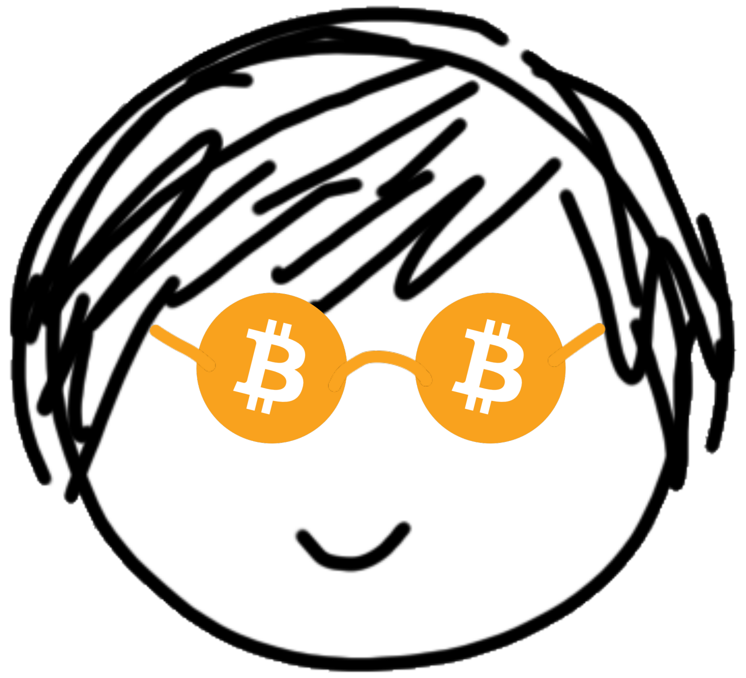
$w_0, w_1, w_2 \leftarrow \mathsf{Share}(w)$

Output $\{\pi_i = P(x_i, w_i)\}_{i \in [3]}$

$V^3(x, \pi_0, \pi_1, \pi_2)$:

$V(x_0, \pi_0) \wedge V(x_1, \pi_1)$
$\wedge V(x_2, \pi_2)$

Trivial Oracle Respecting Distribution

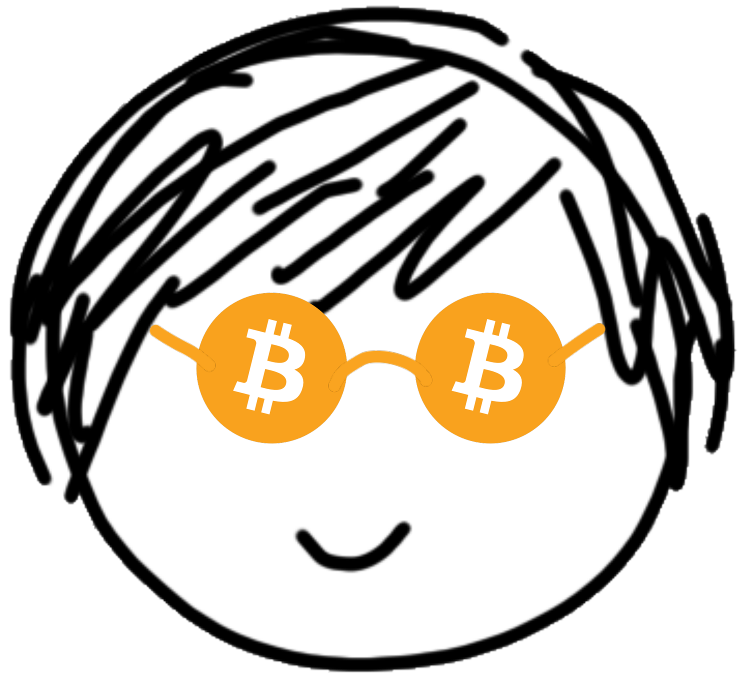


$P^3(x, w)$:

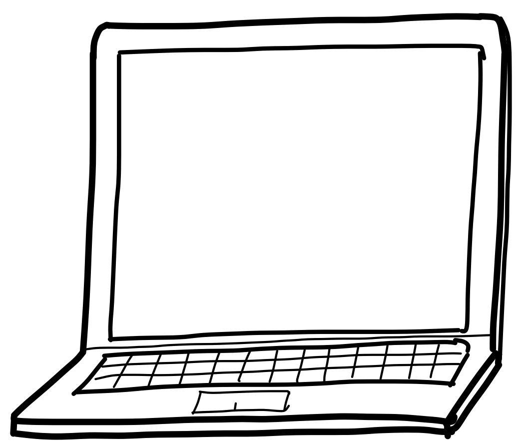
$w_0\,, w_1\,, w_2 \leftarrow \mathsf{Share}(w)$

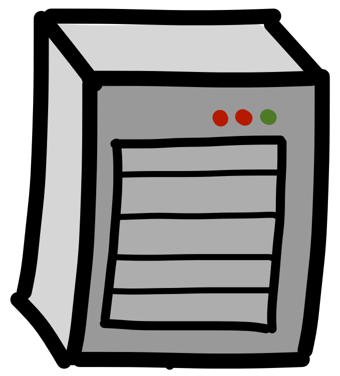
Output $\{\pi_i = P(x_i, w_i)\}_{i \in [3]}$

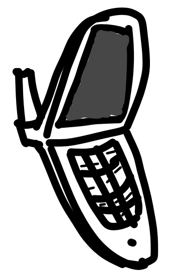
Trivial Oracle Respecting Distribution



$\Pi^3(x, w)$:







H w_0 H w_1 H w_2

$\pi_0 \leftarrow P(w_0)$ $\pi_1 \leftarrow P(w_1)$ $\pi_2 \leftarrow P(w_2)$

Additive secret sharing:
Resilience to two corruptions

Output (π_0, π_1, π_2)

Oracle Respecting Distribution

- Imagine if P^3 had to be distributed among *four* parties instead of three

- <u>In general</u>: P^* that outputs $n \times \pi$ can be distributed amongst n parties, **as long as V^* is aware of n**

- We show that for any NIZK that is SLE in the ROM, this is inherent in the $n-1$ corruption setting

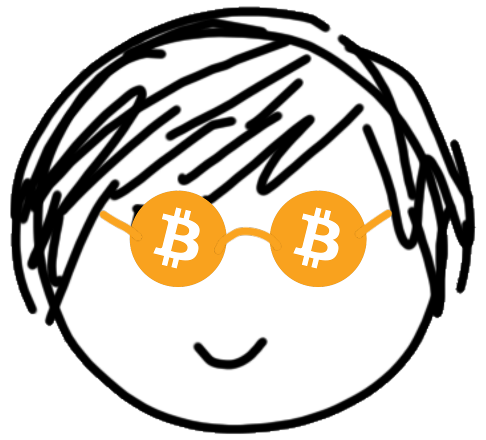
This usually breaks <u>**compatibility**</u>

V is agnostic to n

- Consider a ROM-SLE NIZK (P, V) for some language
- Assumption: $n-1 \in \mathsf{poly}(\kappa)$ is a strict upper bound on queries made by V to the random oracle H
  - Holds for most 'natural' schemes
- We will show: any n-party protocol that ORD-computes P will leak the witness to $n-1$ parties

Trimming Resilience

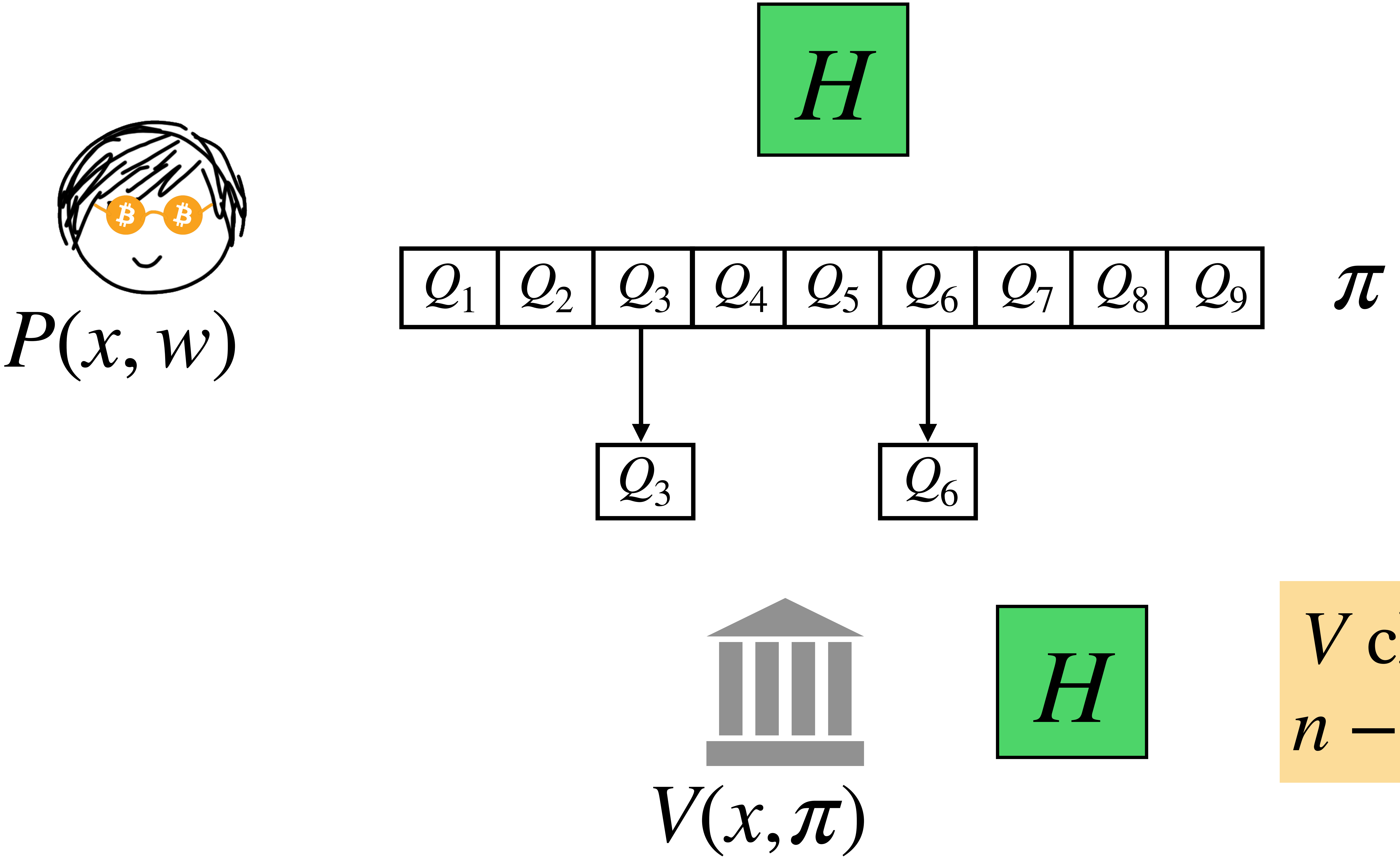
$\boxed{H}$



$P(x, w)$

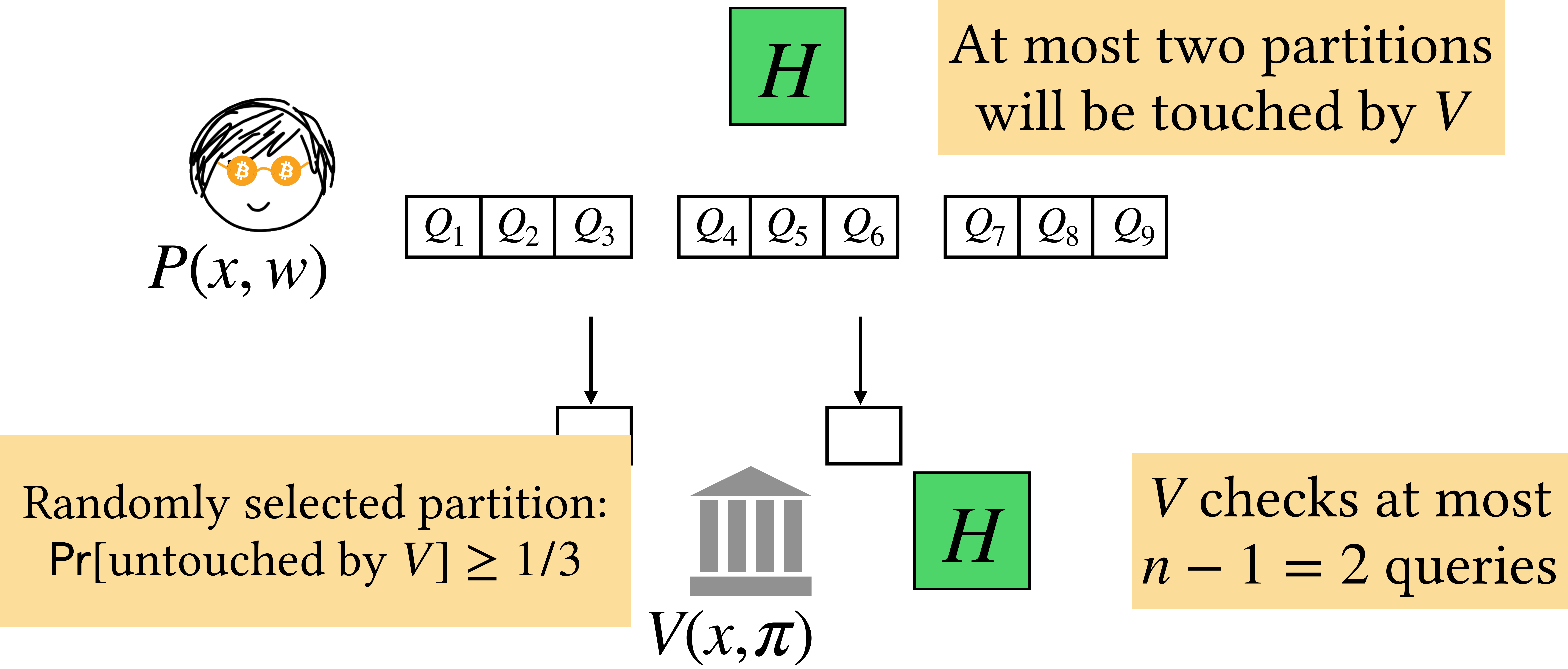
π

Trimming Resilience



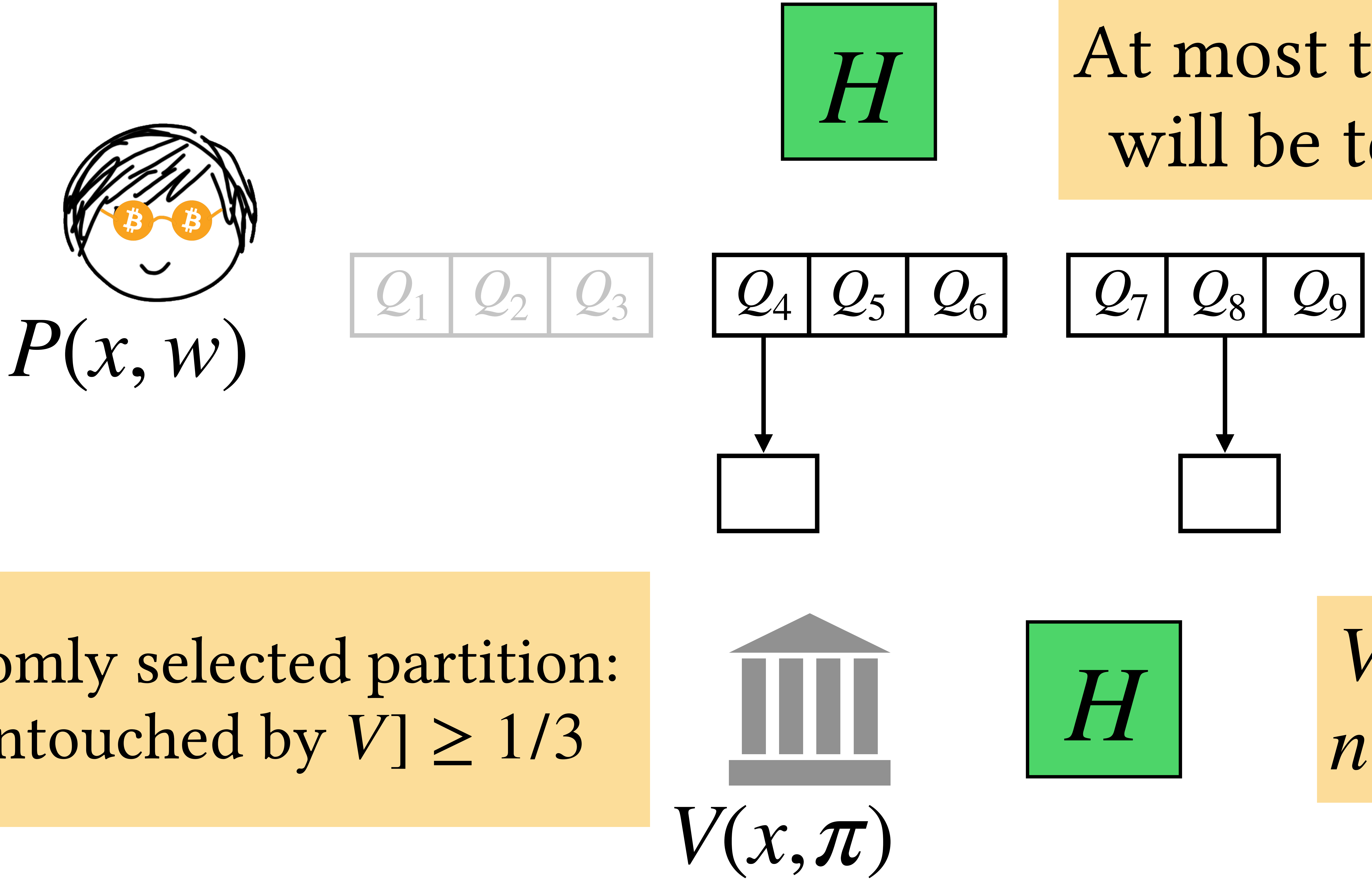
V checks at most $n - 1 = 2$ queries

Trimming Resilience



Trimming Resilience

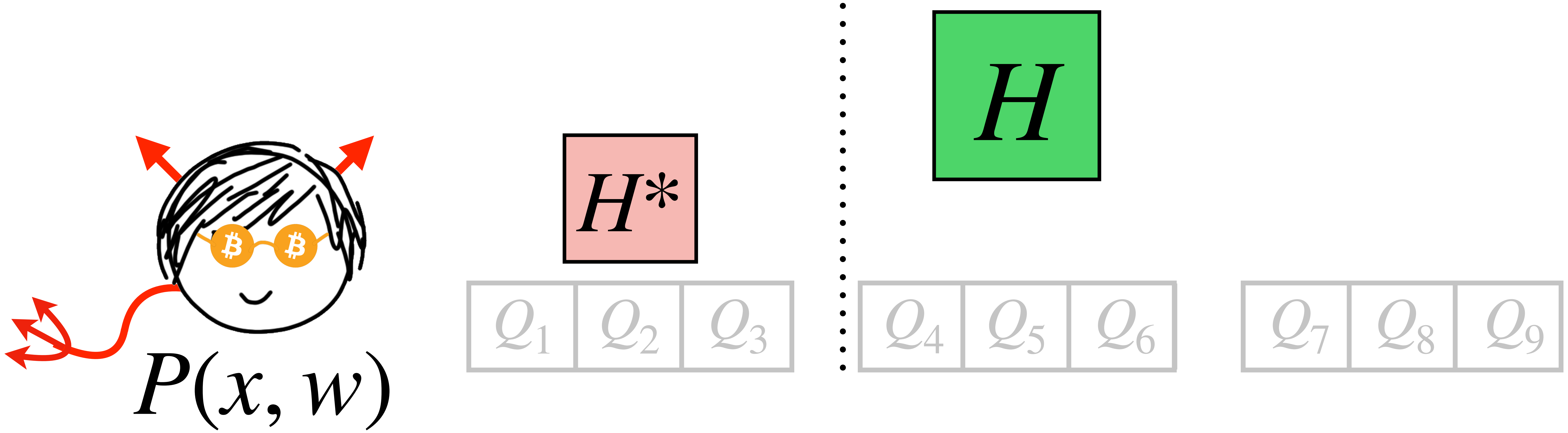
At most two partitions will be touched by V



Randomly selected partition:
Pr[untouched by V] $\geq 1/3$

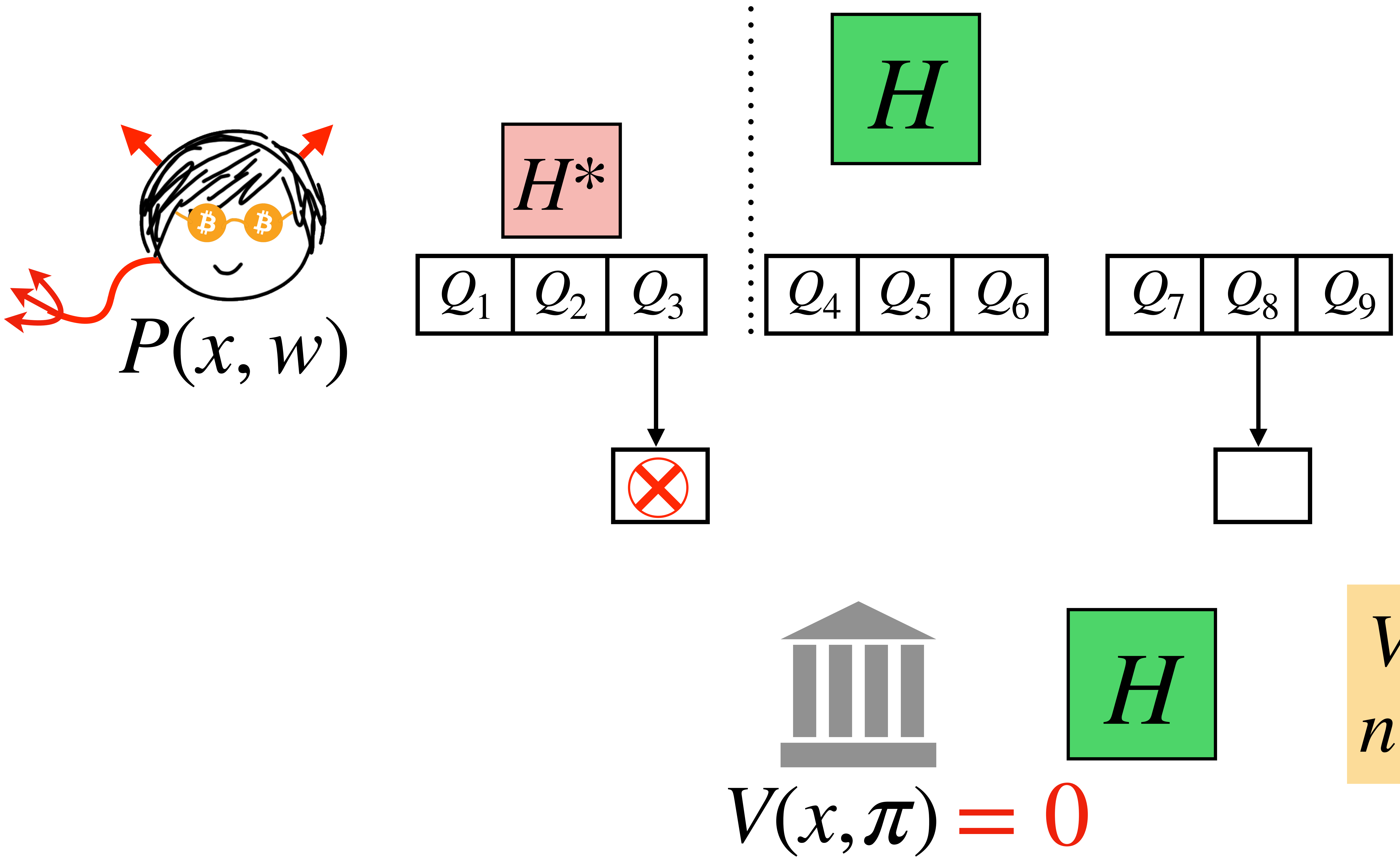
V checks at most
$n - 1 = 2$ queries

Trimming Resilience



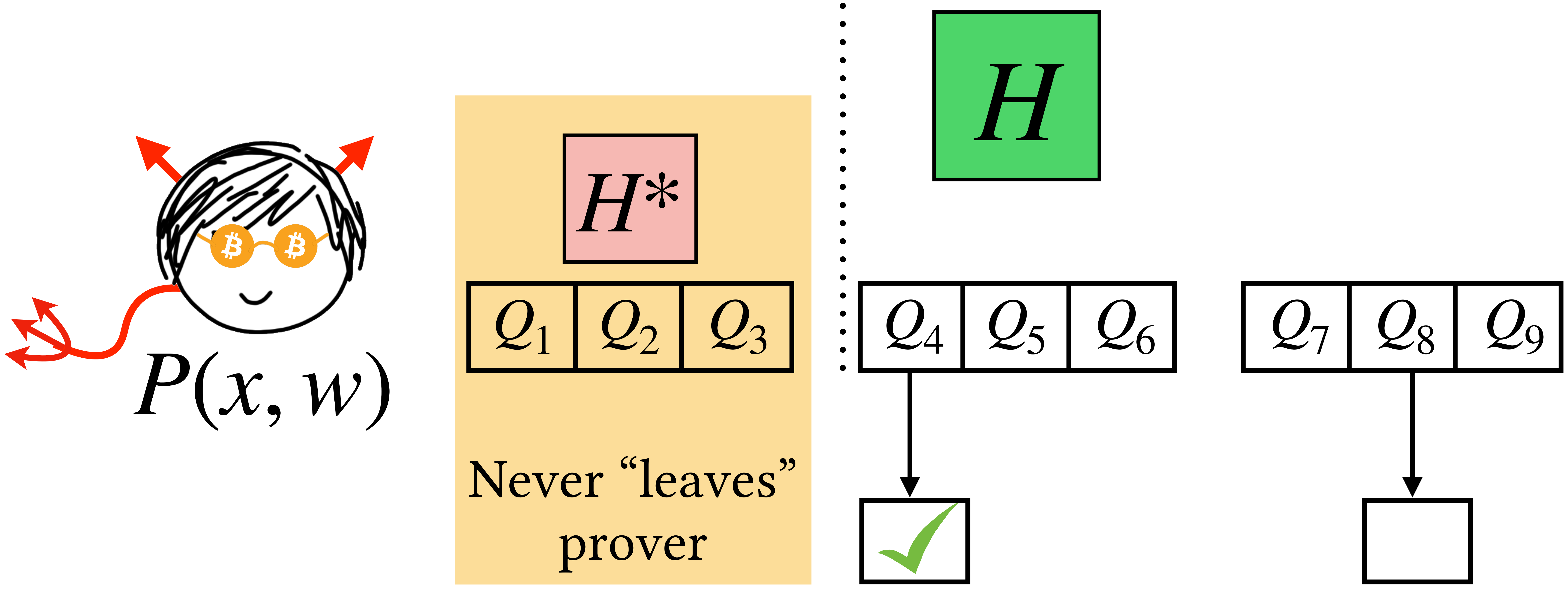
V checks at most $n - 1 = 2$ queries

Trimming Resilience

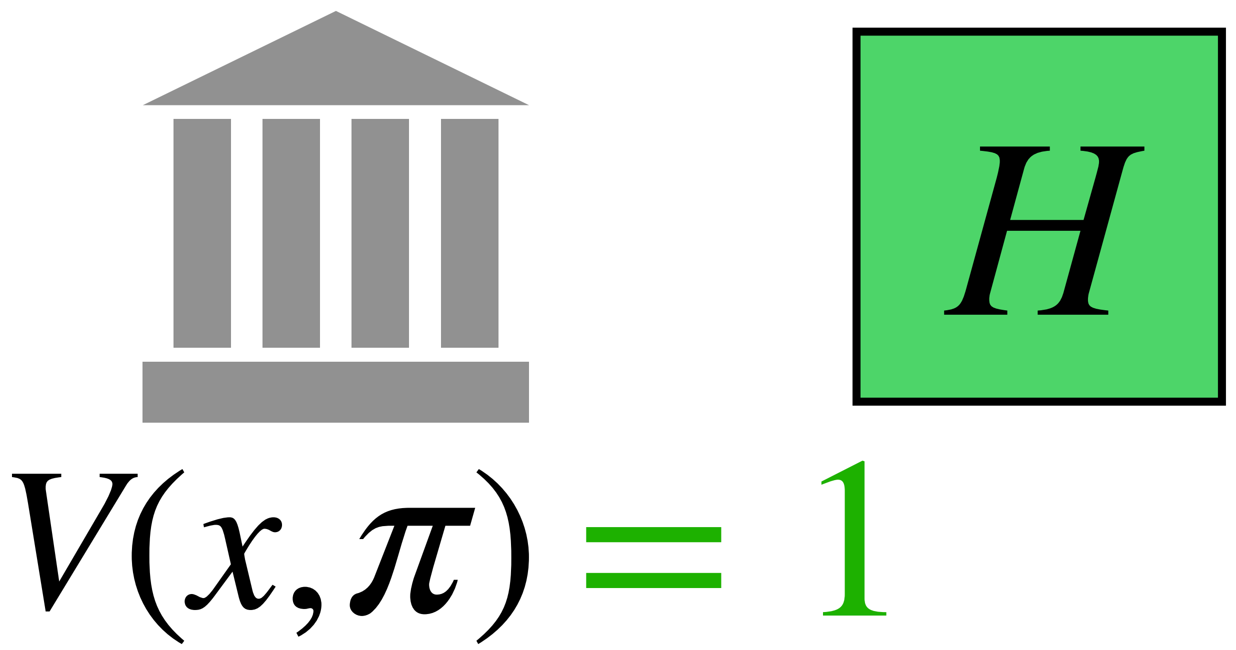


V checks at most $n - 1 = 2$ queries

Trimming Resilience

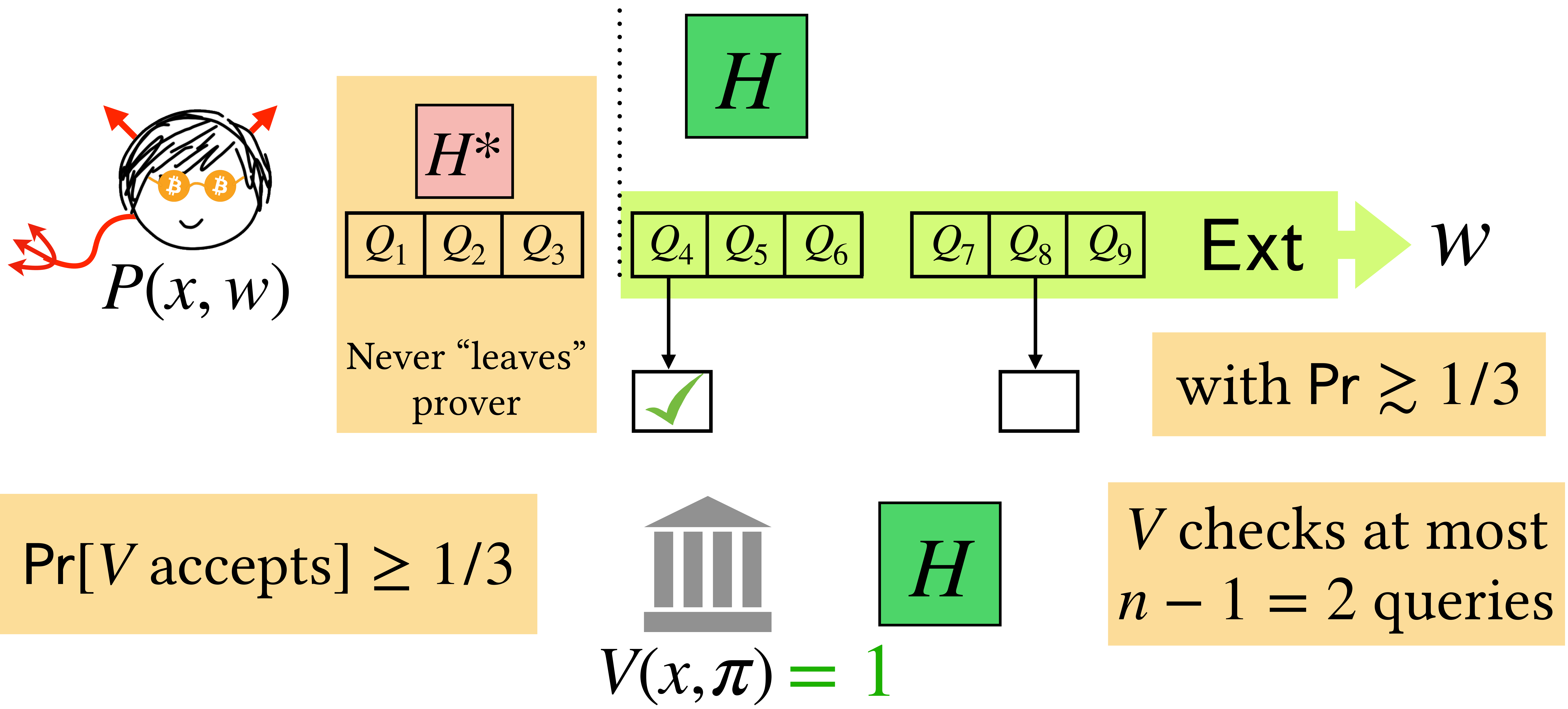


$\Pr[V \text{ accepts}] \geq 1/3$

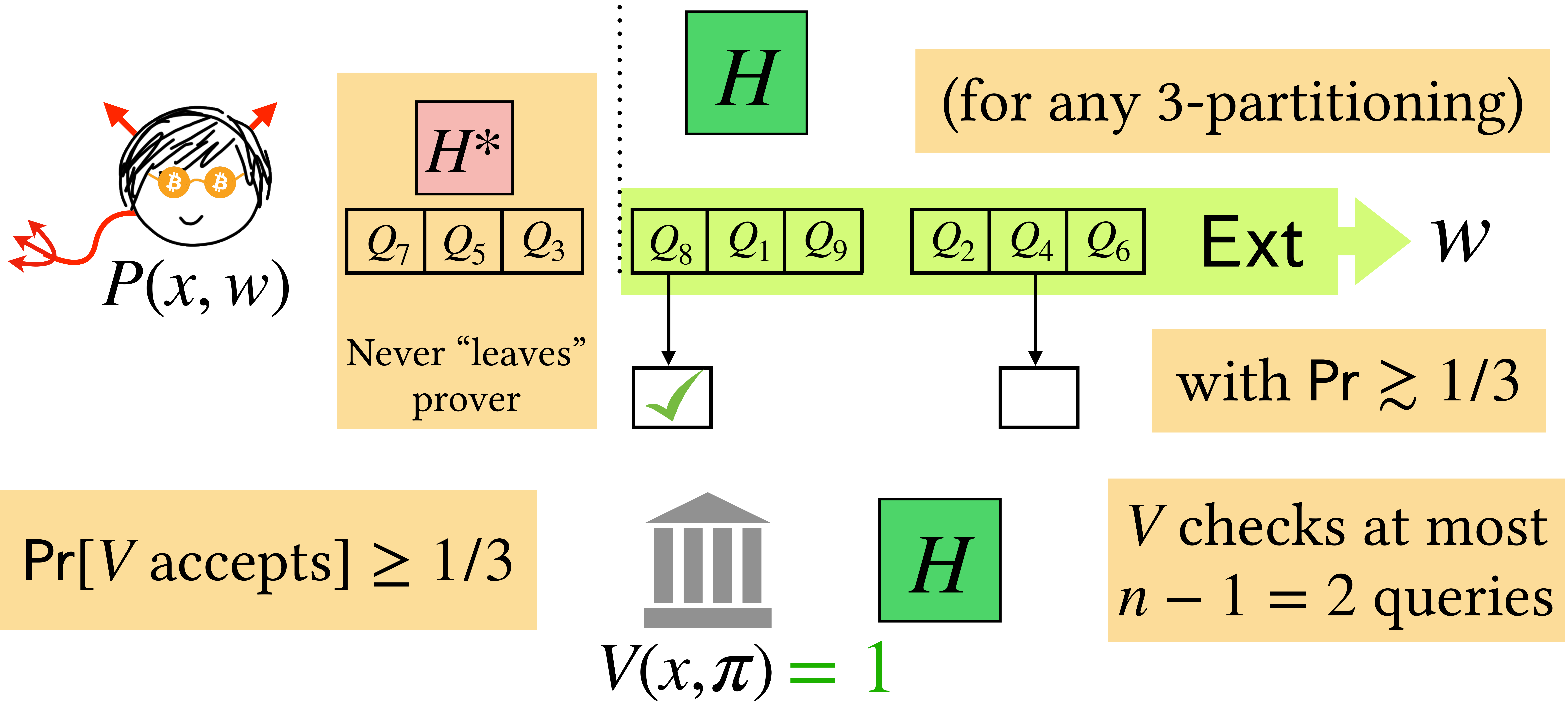


V checks at most $n - 1 = 2$ queries

Trimming Resilience



Trimming Resilience

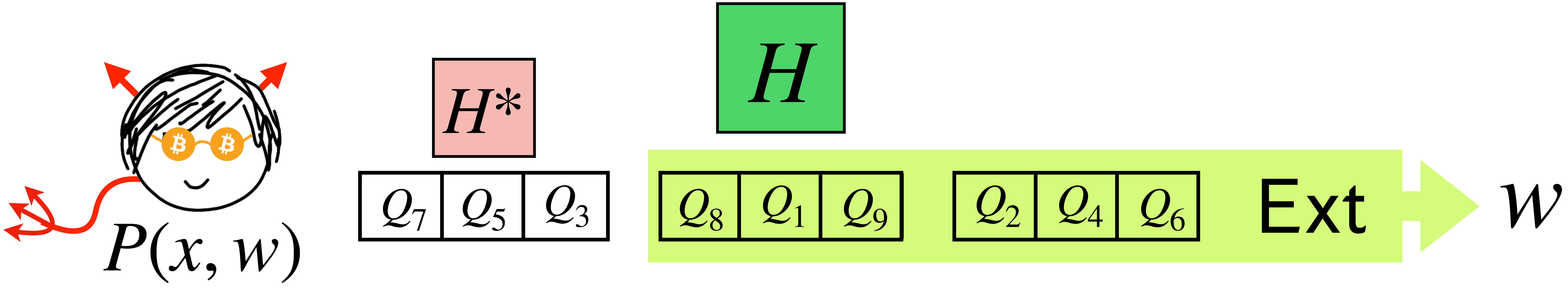


Trimming Resilience

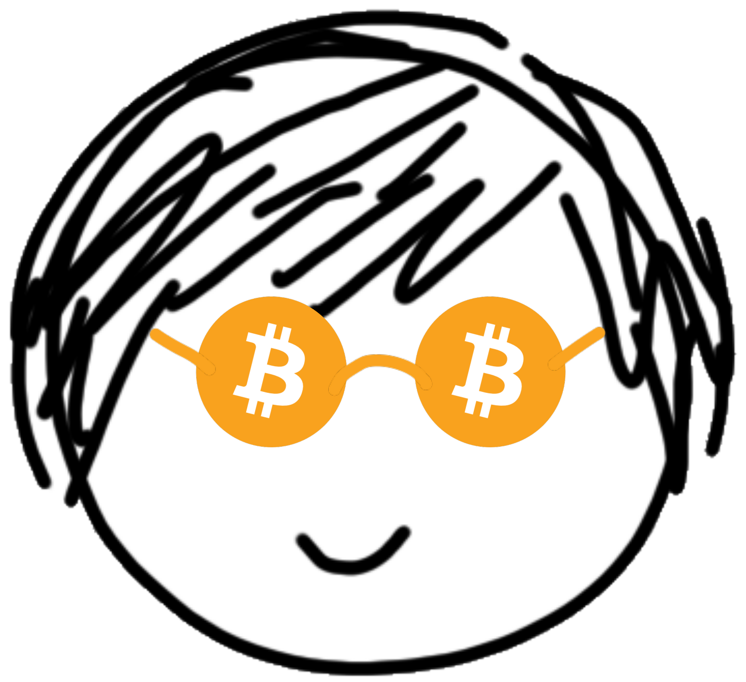
<u>Lemma</u>: For any n-partitioning of RO queries, omitting *one* partition still allows extraction if the verifier checks at most $n - 1$ queries

(random)

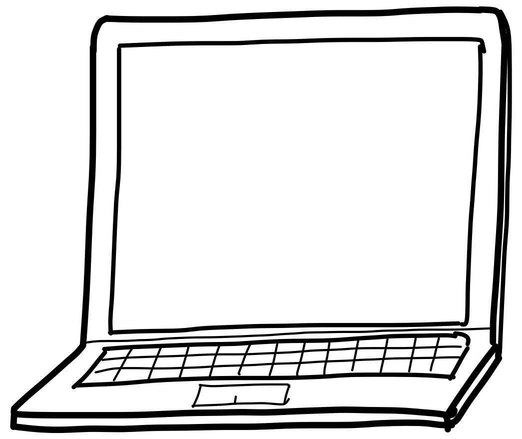
(w. noticeable probability)

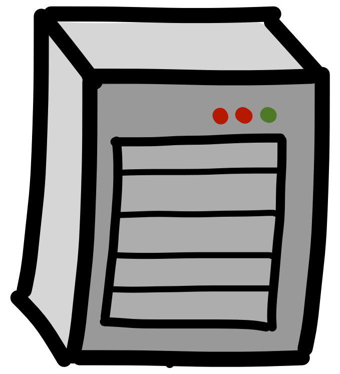


Oracle Respecting Distribution



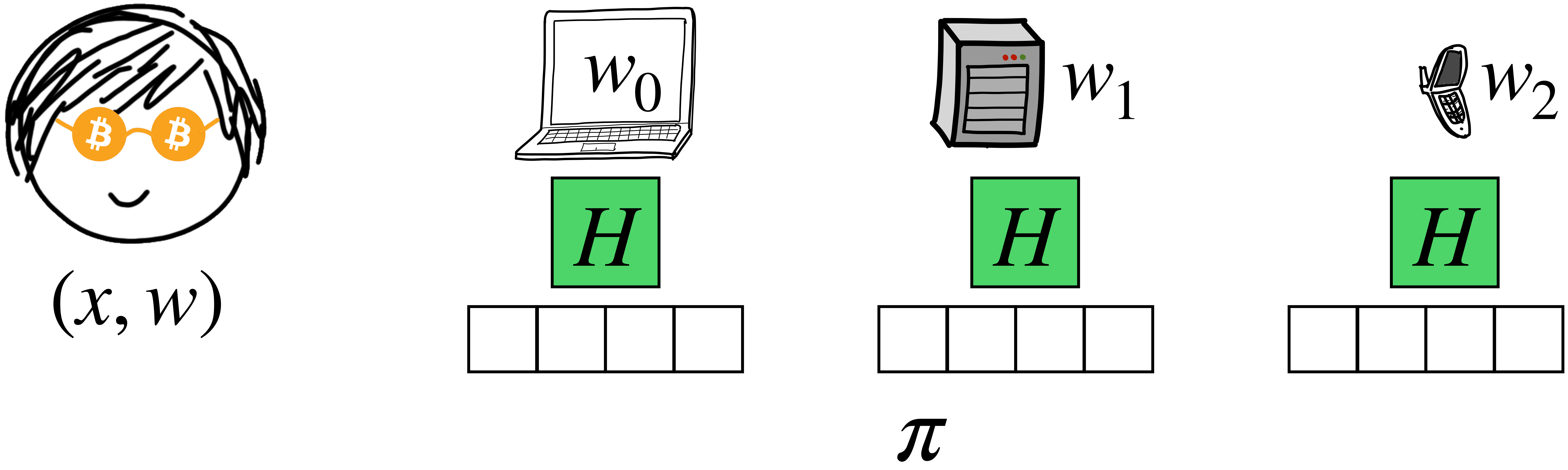
(x, w)



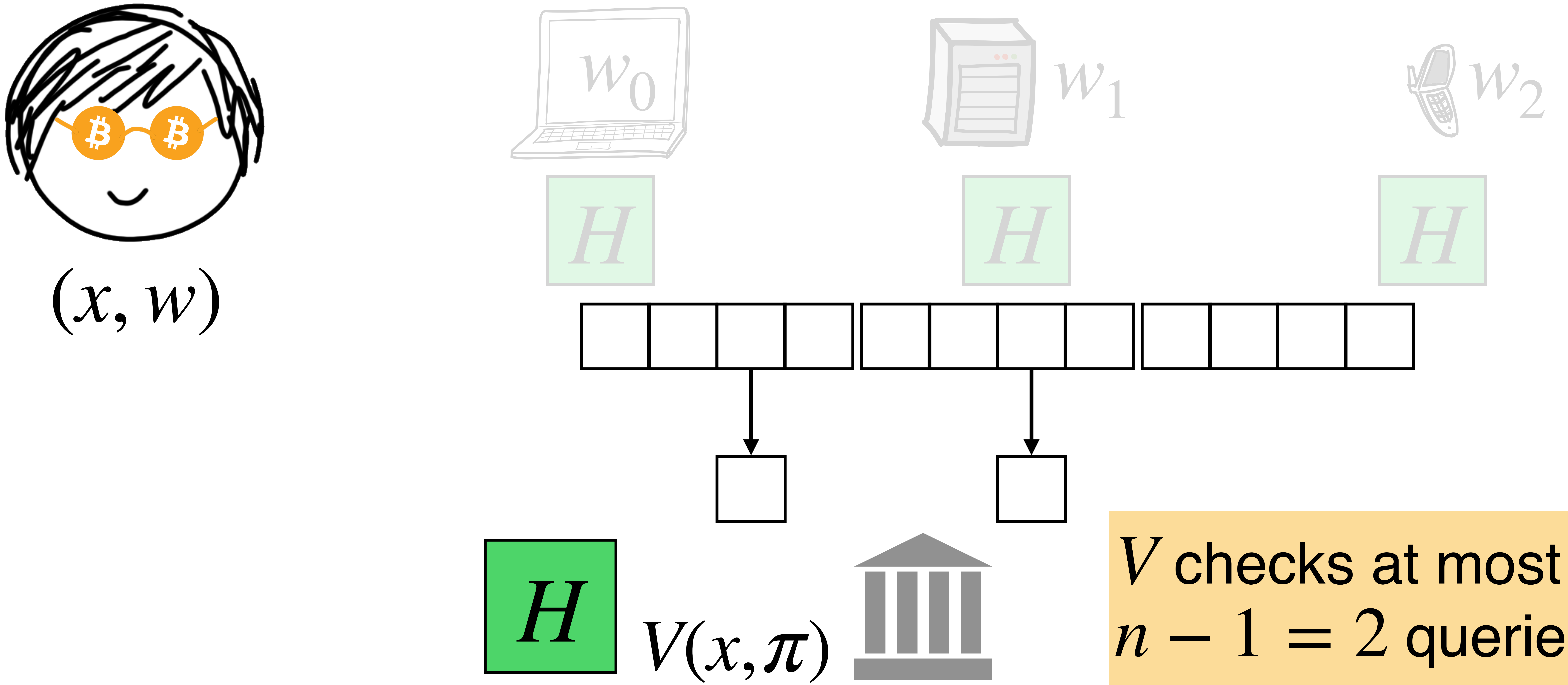


$$w_0\,,\, w_1\,, w_2 \leftarrow \mathsf{Share}(w)$$

Oracle Respecting Distribution

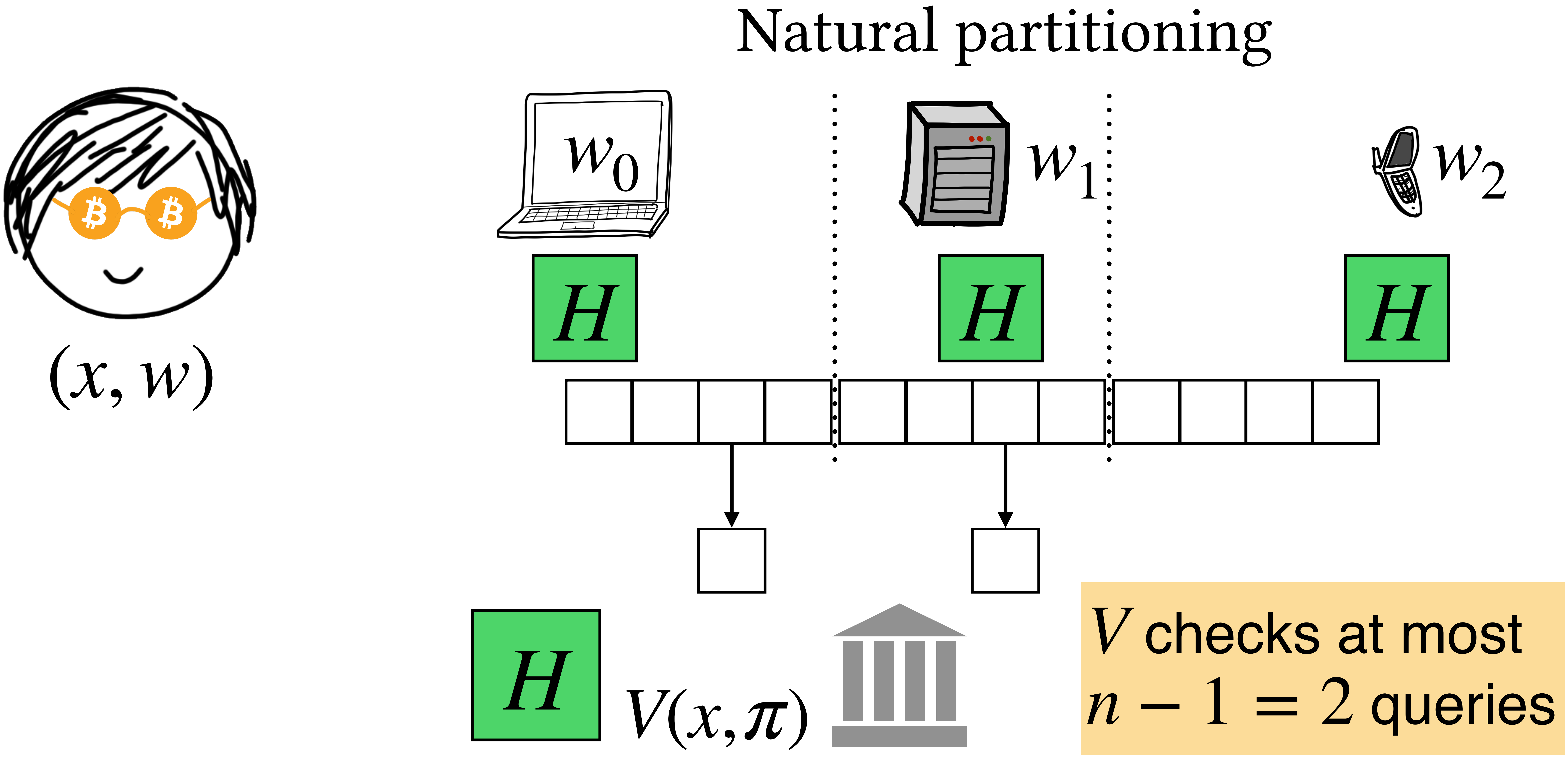


Oracle Respecting Distribution

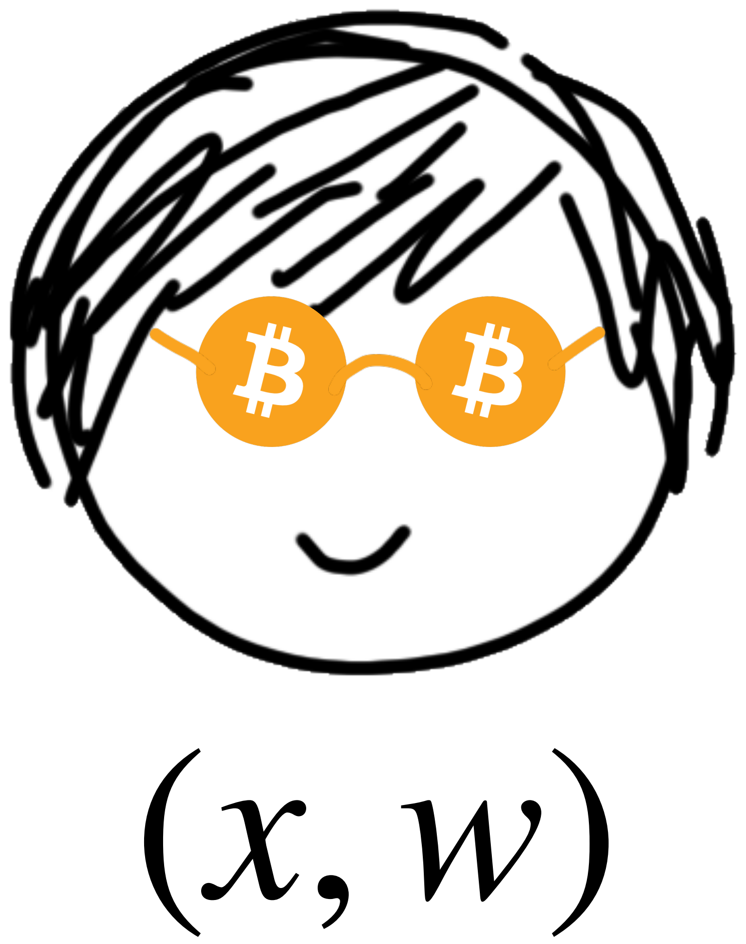


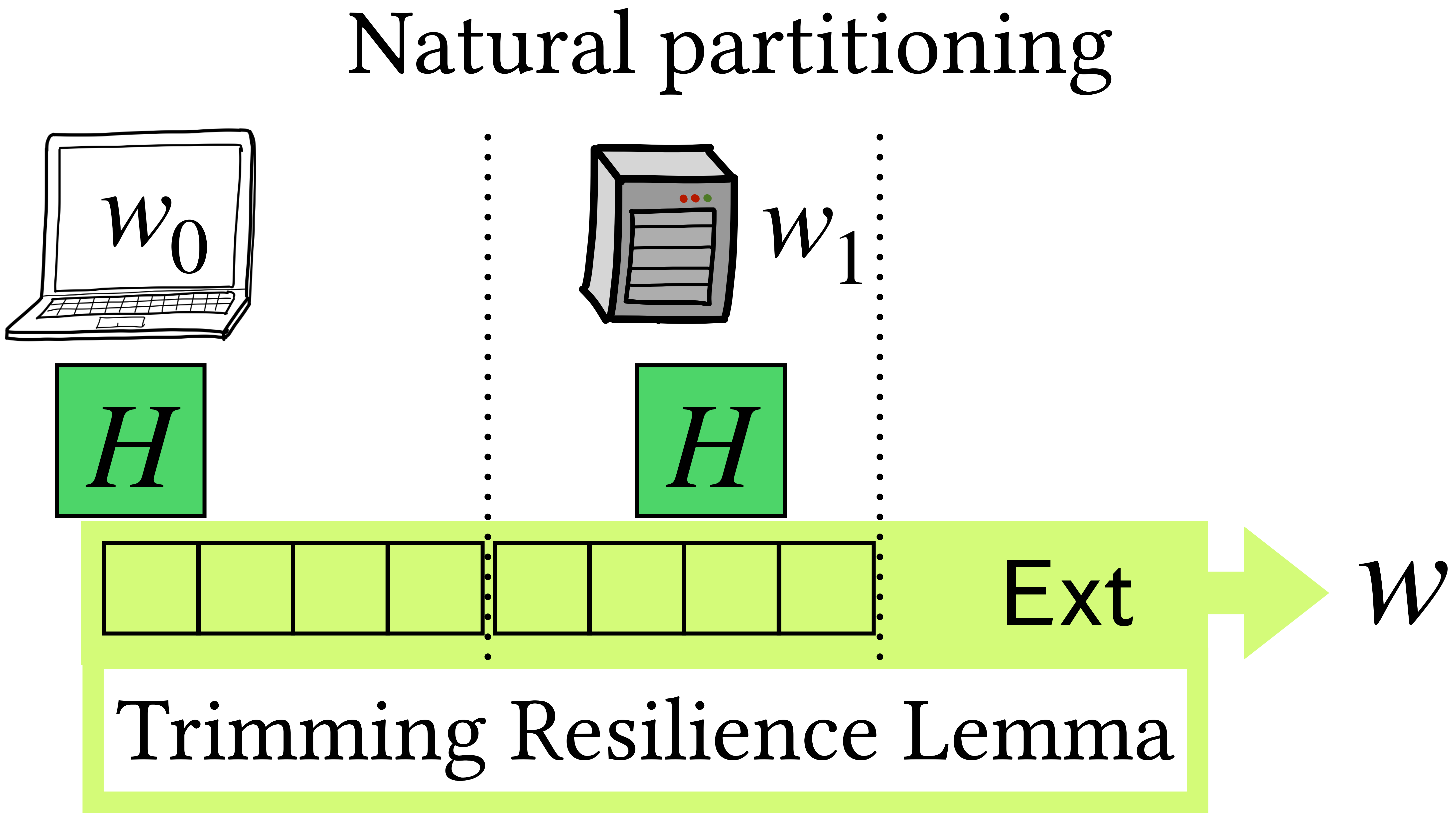
V checks at most $n - 1 = 2$ queries

Oracle Respecting Distribution



Oracle Respecting Distribution





n party ORD protocol can not withstand n-1 passive corruptions

Other Corruption Levels?

- Previous technique cannot be directly extended for fewer than $n - O(1)$ corruptions

  $\exists$ NIZKPoK of DLog π s.t. for any constant c, $\exists$ n-party ORD protocol to securely compute π with tolerance to $c \cdot n$ malicious corruptions

- However, ORD protocols for NIZKs where Ext needs a *single private query* of P seem unlikely **for even one corruption**

A Question

Should threshold signature size grow with signer count?

Sometimes You Can't Distribute Random-Oracle-Based Proofs

¯_(ツ)_/¯

Jack Doerner

Yashvanth Kondi

Leah Namisa Rosenbloom

eprint.iacr.org/2023/1381

Thanks **Eysa Lee** for