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

Jack Doerner — BROWN

Yashvanth Kondi — SILENCE LABORATORIES

Leah Namisa Rosenbloom — Northeastern University

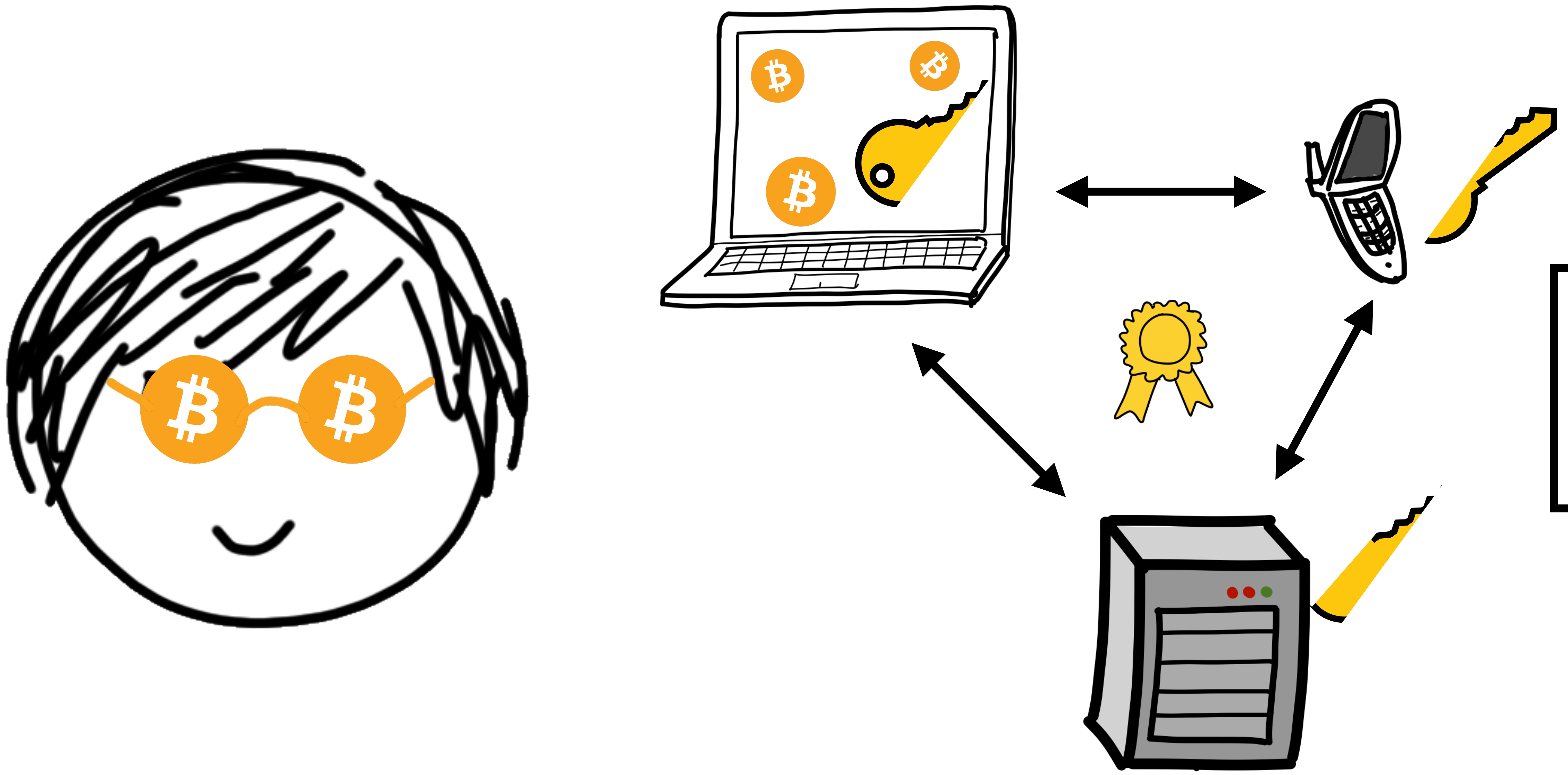
eprint: 2023/1381

v1.2

Threshold / Distributed Signing

- Specialized Multiparty Computation (MPC) protocols to securely compute $\mathsf{Sign}(\mathsf{sk}, m)$ from secret shares of sk

- Commonly applied to decentralize key management



Mitigates risk of key theft

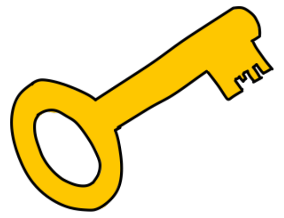
- **Compatibility**:
  Verifies w.r.t. original algorithm

- **Corruption Resilience**:
  Compromising some devices does not leak the signing key

- This talk: Signatures ⇔ Non-interactive Zero-knowledge

 ⇔ π

(

, ) ⇔ (x, w)

Distributed Signing ⇔ Distributed Proving

How to Distribute Signing

- Any signing scheme can be distributed via general MPC
- "Practical" efficiency usually requires more fine-grained notions than just feasibility
- As one proxy, practical distributed signing protocols make **blackbox use** of complex components of the signing algorithm:
  - Integer arithmetic in $\mathbb{Z}_q$ or $\mathbb{Z}_N^*$
  - Elliptic curve group operations
  - Hash functions

How to Distribute Signing

- Any signing scheme can be distributed via general MPC
- "Practical" efficiency usually requires more fine-grained notions than just feasibility
- As one proxy, practical distributed signing protocols make **blackbox use** of complex components of the signing algorithm:
  - Integer arithmetic in $\mathbb{Z}_q$ or $\mathbb{Z}_N^*$
  - Elliptic curve group operations
  - Hash functions

RSA, Schnorr/EdDSA, ECDSA, BLS, BBS+, custom constructions using lattices, isogenies, etc.

What about Purely Hash Based?

- Proof size, verifier time linear in #provers
  [Ozdemir Boneh 22]: distributed version of Fractal
  [Cui Zhang Chen Liu Yu 21]: distributed MPC-in-the-head

- Prove statements about circuit representation of hash function
  [Khaburzaniya Chalkias Lewi Malvai 21]: aggregate Lamport signatures with STARKs

- Hash-based proofs that are designed to be hard to distribute
  [Dziembowski Faust Lizurej 23]: Individual Cryptography
  [Kelkar Babel Daian Austgen Buterin Juels 23]: Complete Knowledge

This Work: Limitations

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

This Work: 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

This Work: 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

2. Attack that completely recovers the witness by corrupting all-but-one distributed provers

This Work: Limitations

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

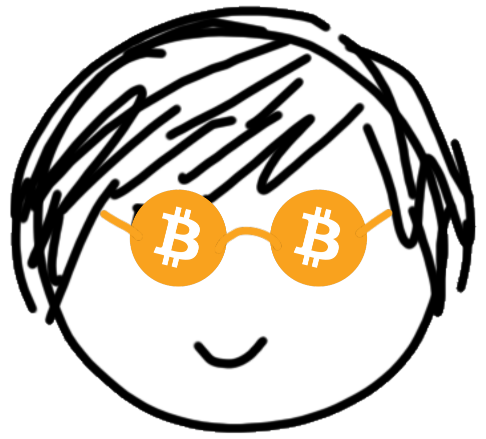
1. NIZKs that have straight-line extractors in the Random-Oracle Model
2. Attack that completely recovers the witness by corrupting all-but-one distributed provers
3. Protocol that is blackbox in the same hash function (i.e. Random Oracle) as the NIZK

Implications for distributing…

- Signing for standard schemes based on MPC-in-the-head
- NIZKs/signatures obtained by compiling Sigma protocols via:
  - Pass' or Fischlin's transformations (tight/concurrent security)
  - Unruh's transformation (post-quantum)
- PCPs/IOPs compiled via hash functions

Proofs of Knowledge

- What does it mean for a proof to certify "knowledge" of a witness?
- "Proof of Knowledge" is formalized by an "extractor" Ext



$P(x, w)$:

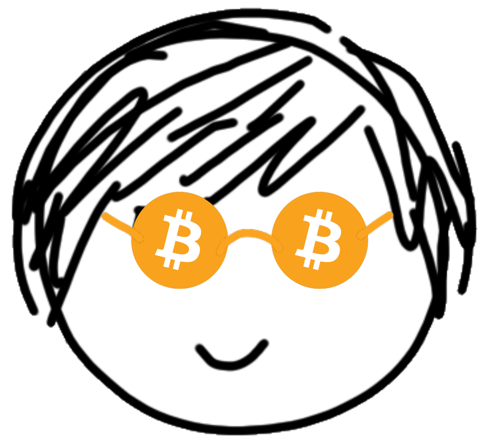
(NI)Zero-knowledge Proof:
"I know w such that $(x, w) \in L$"

$V(x)$

Accept/Reject

Proofs of Knowledge

- What does it mean for a proof to certify "knowledge" of a witness?
- "Proof of Knowledge" is formalized by an "extractor" Ext



$P(x, w)$:

(NI)Zero-knowledge Proof:
"I know w such that $(x, w) \in L$"

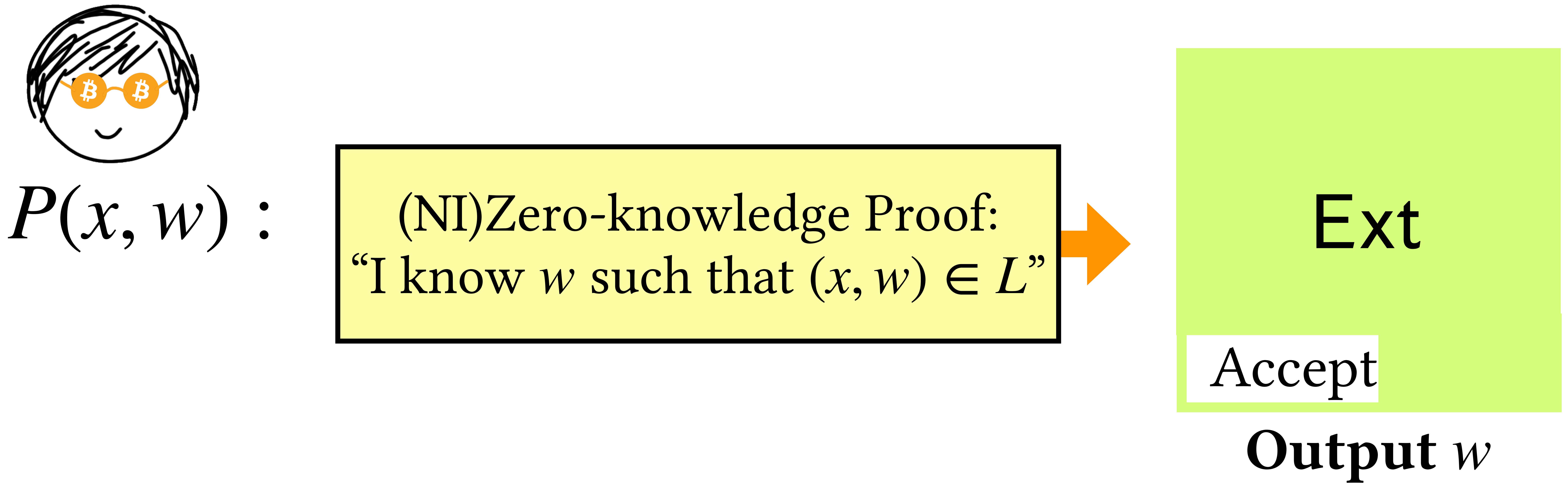
Ext

Accept

Proofs of Knowledge

- What does it mean for a proof to certify "knowledge" of a witness?

- "Proof of Knowledge" is formalized by an "extractor" Ext



Why is Ext special?

- Clearly, Ext must not be an algorithm that just 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.

Why is Ext special?

- Clearly, Ext must not be an algorithm that just anybody can run

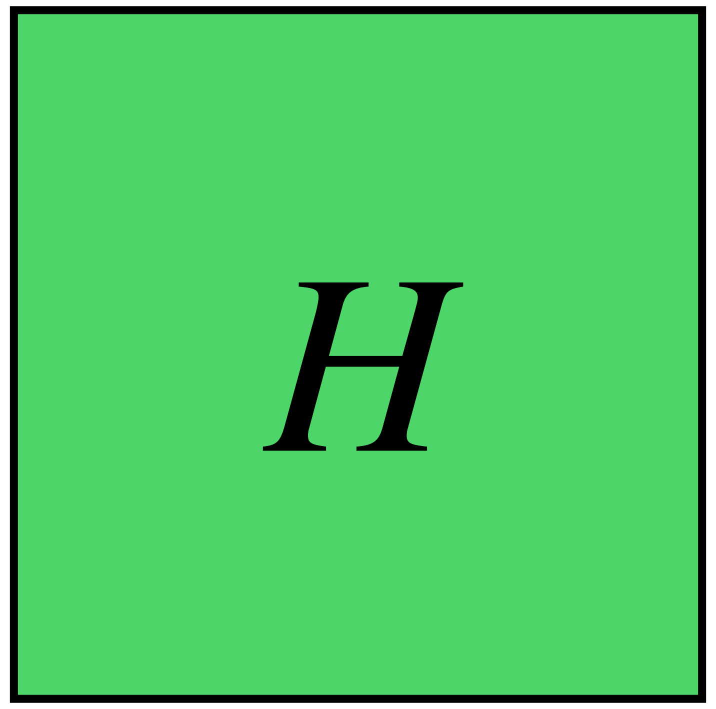
- Ext has carefully chosen special privileges:
  - Powerful enough to accomplish extraction
  - Still meaningful as a security claim

- "Straight-line" Extraction (SLE): no rewinding. Instead, use other trapdoor like CRS, RO, etc.

<u>Bad for</u>:

- Quantum
- Concurrency
- Tightness

Random Oracle Model



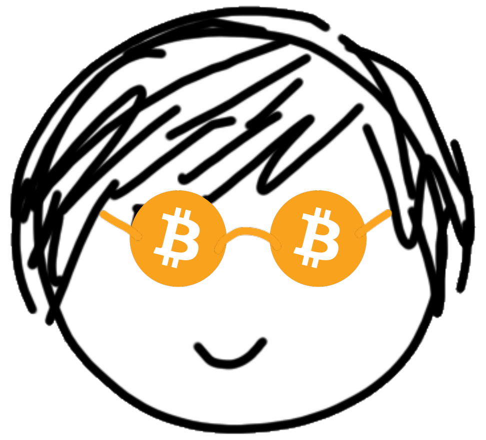
$$H : \{0,1\}^* \mapsto \{0,1\}^\ell$$

Random Oracles as Ext Privilege

[Pass 03]

H

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



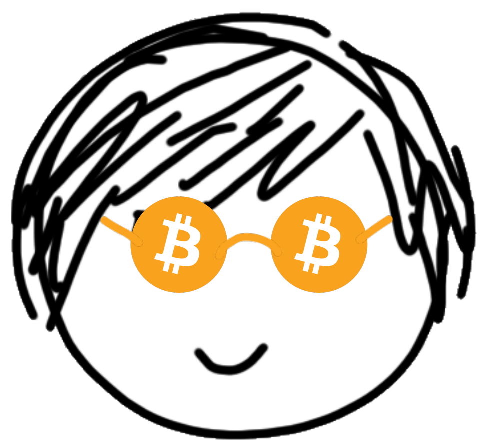
$P(x, w)$

Random Oracles as Ext Privilege

[Pass 03]

H

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



$P(x, w)$

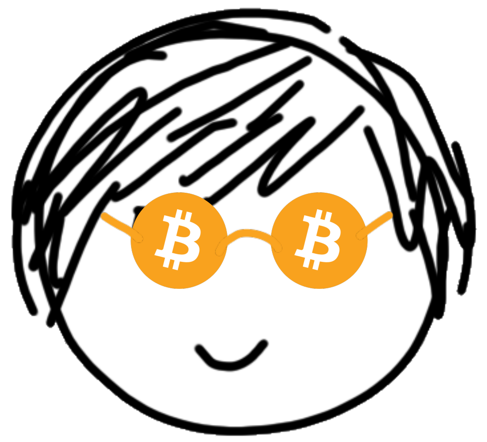
Q_1	Q_2	•	•	Q_i	•	•	•	•	•	Q_n

Random Oracles as Ext Privilege

[Pass 03]

H

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



$P(x, w)$

Q_1	Q_2	•	•	Q_i	•	•	•	•	•	Q_n

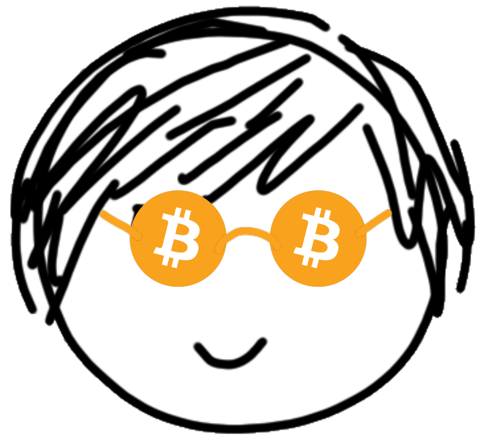
$V(x)$

Random Oracles as Ext Privilege

[Pass 03]

H

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



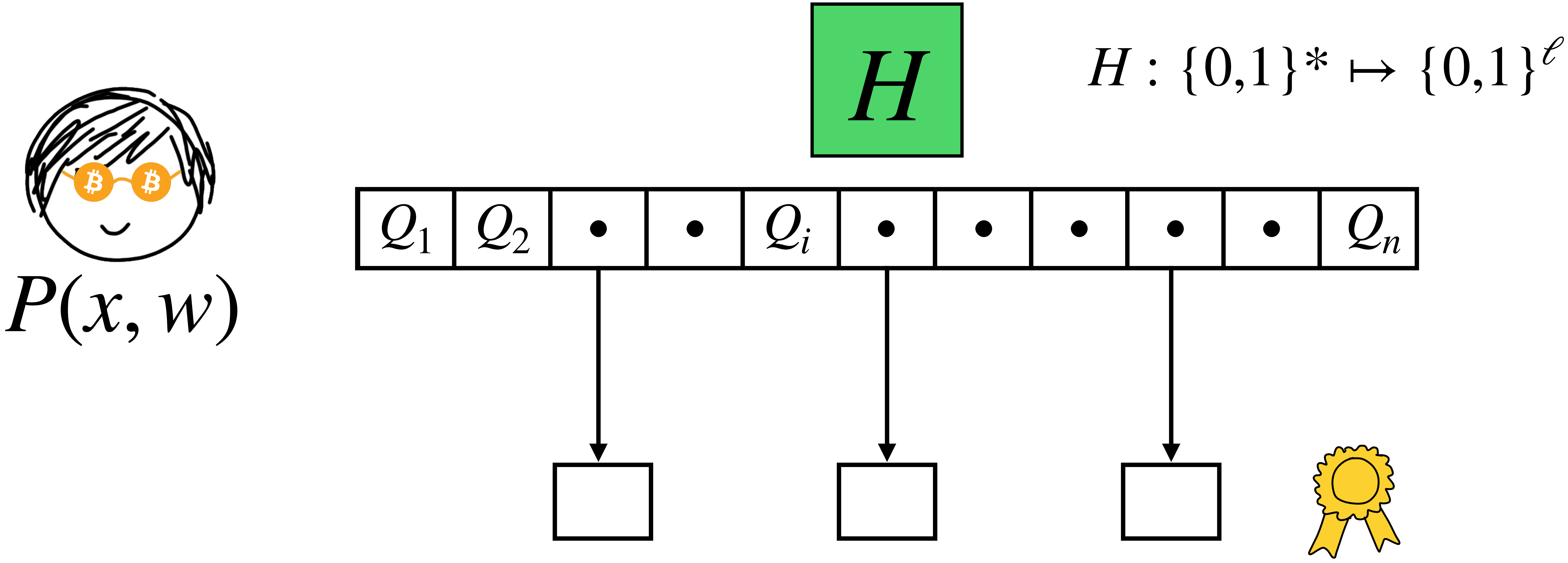
$P(x, w)$

Q_1	Q_2	•	•	Q_i	•	•	•	•	•	Q_n

$V(x)$

Random Oracles as Ext Privilege

[Pass 03]

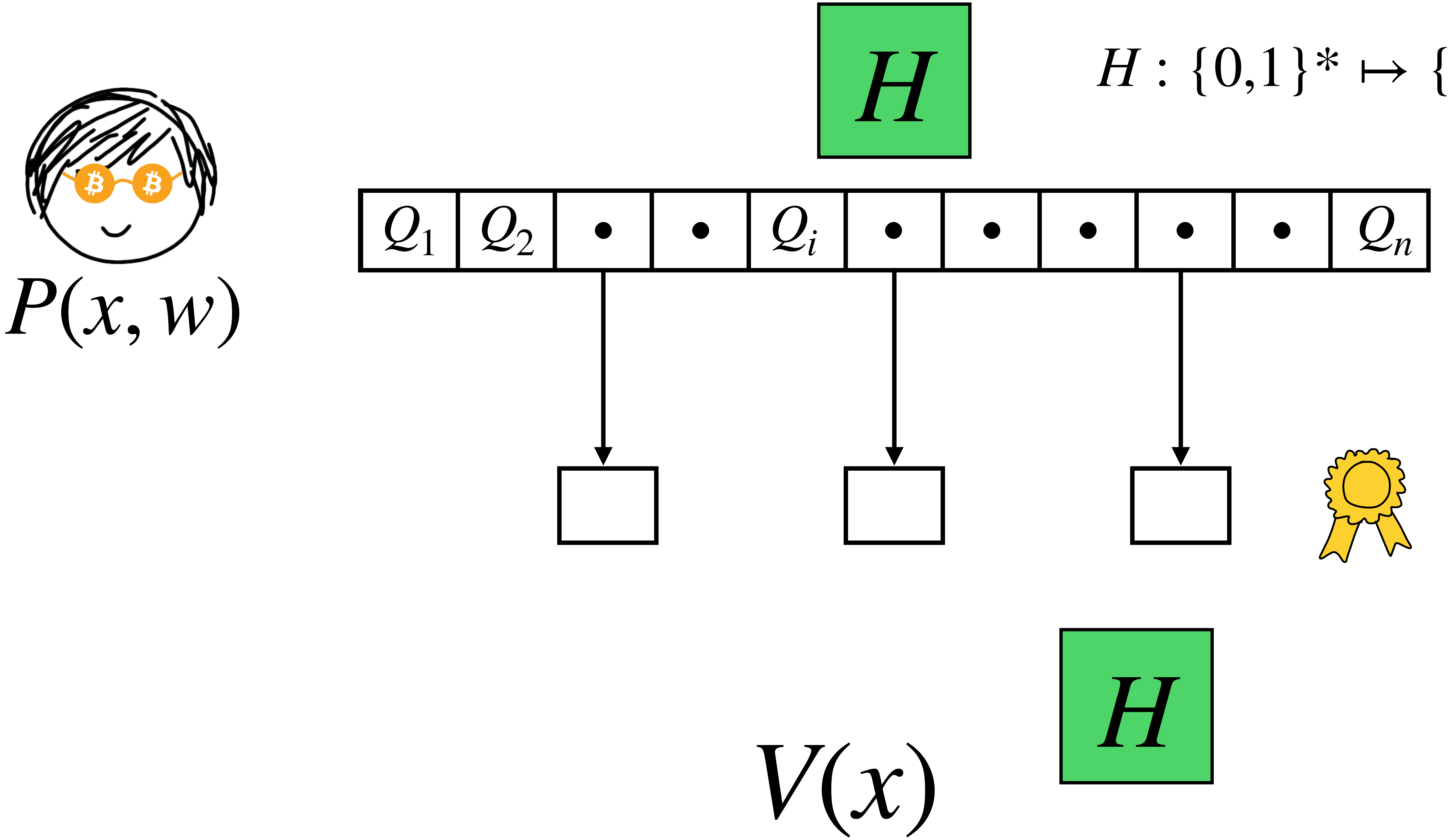


$V(x)$

Random Oracles as Ext Privilege

[Pass 03]

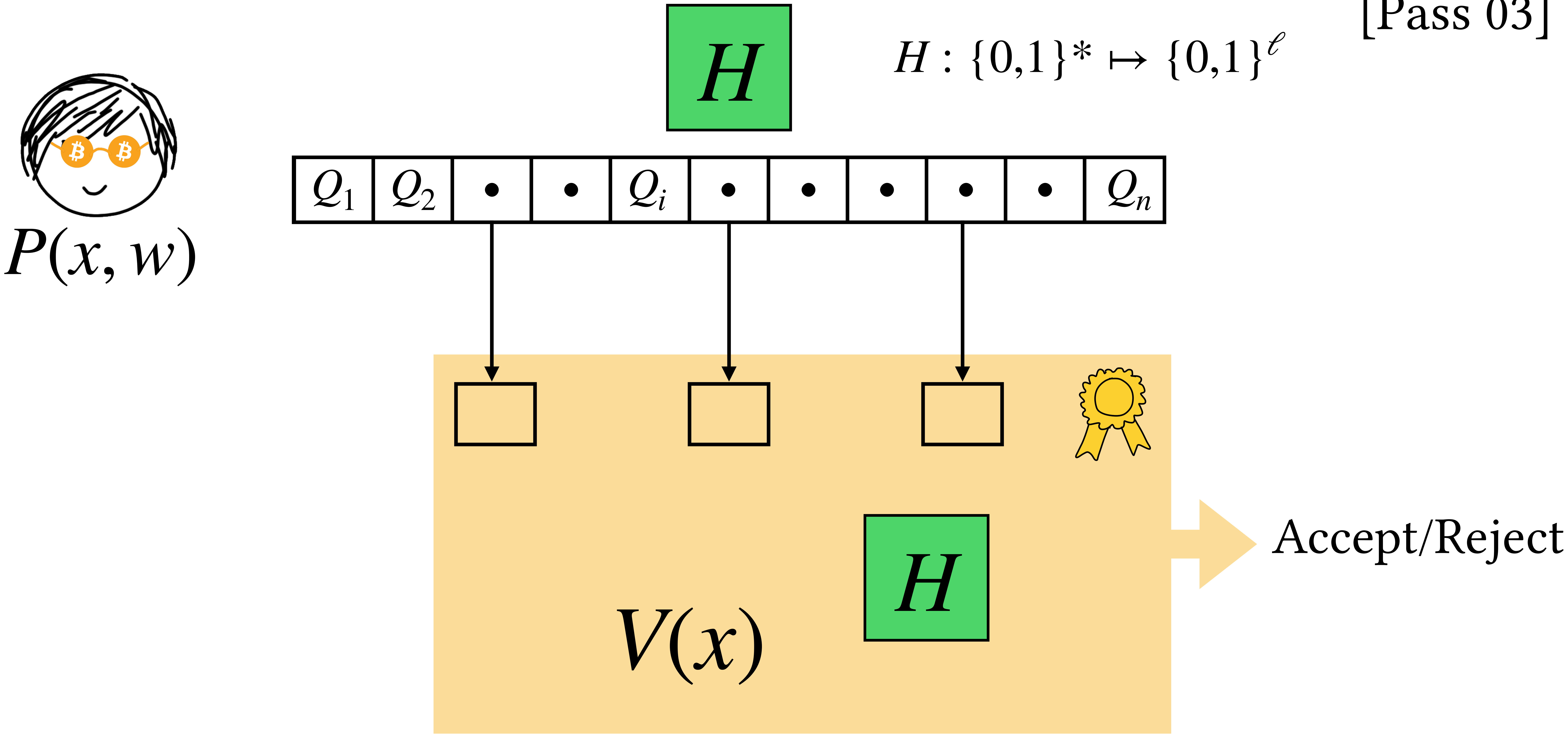
$H : \{0,1\}^* \mapsto \{0,1\}^\ell$



Random Oracles as Ext Privilege

[Pass 03]

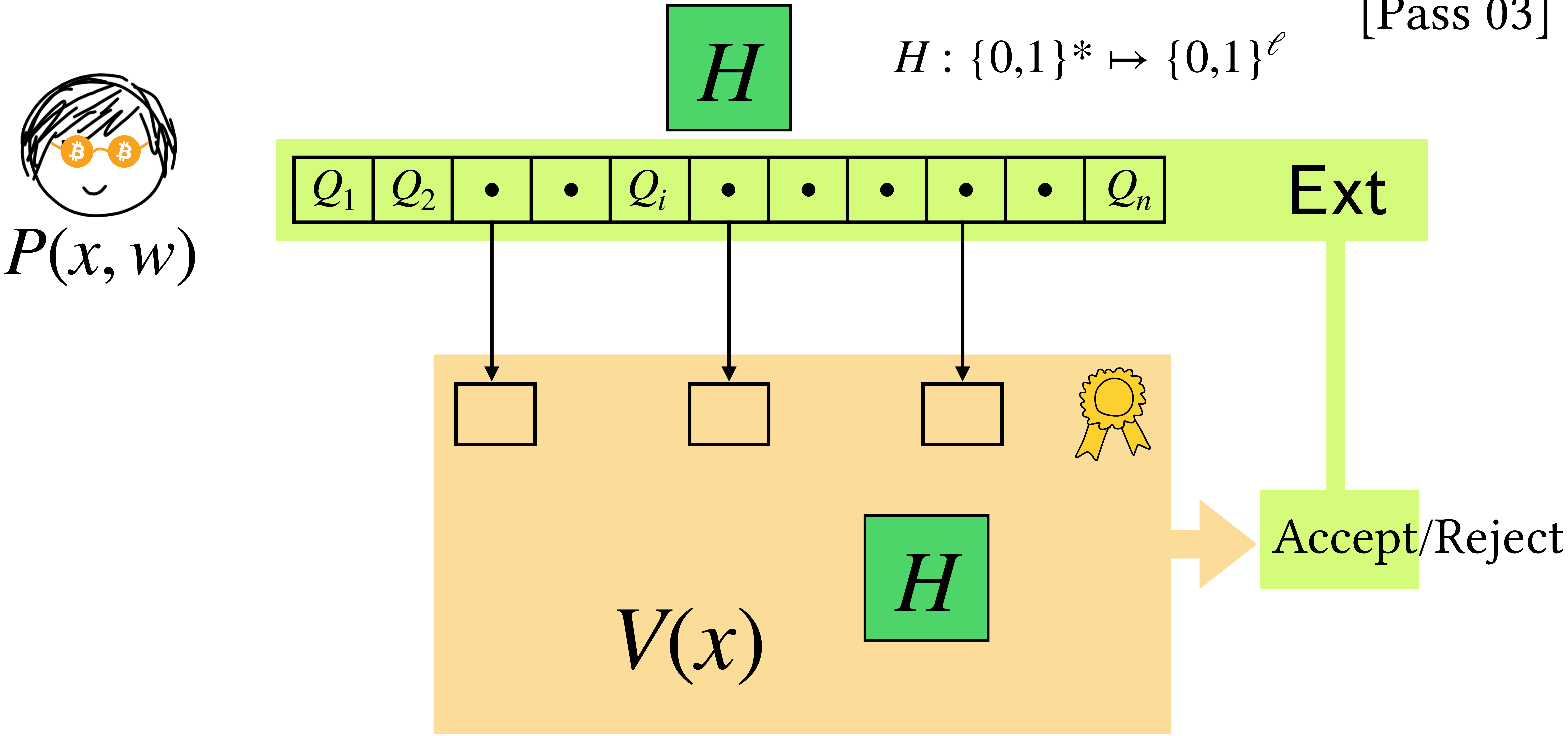
$H : \{0,1\}^* \mapsto \{0,1\}^\ell$



Random Oracles as Ext Privilege

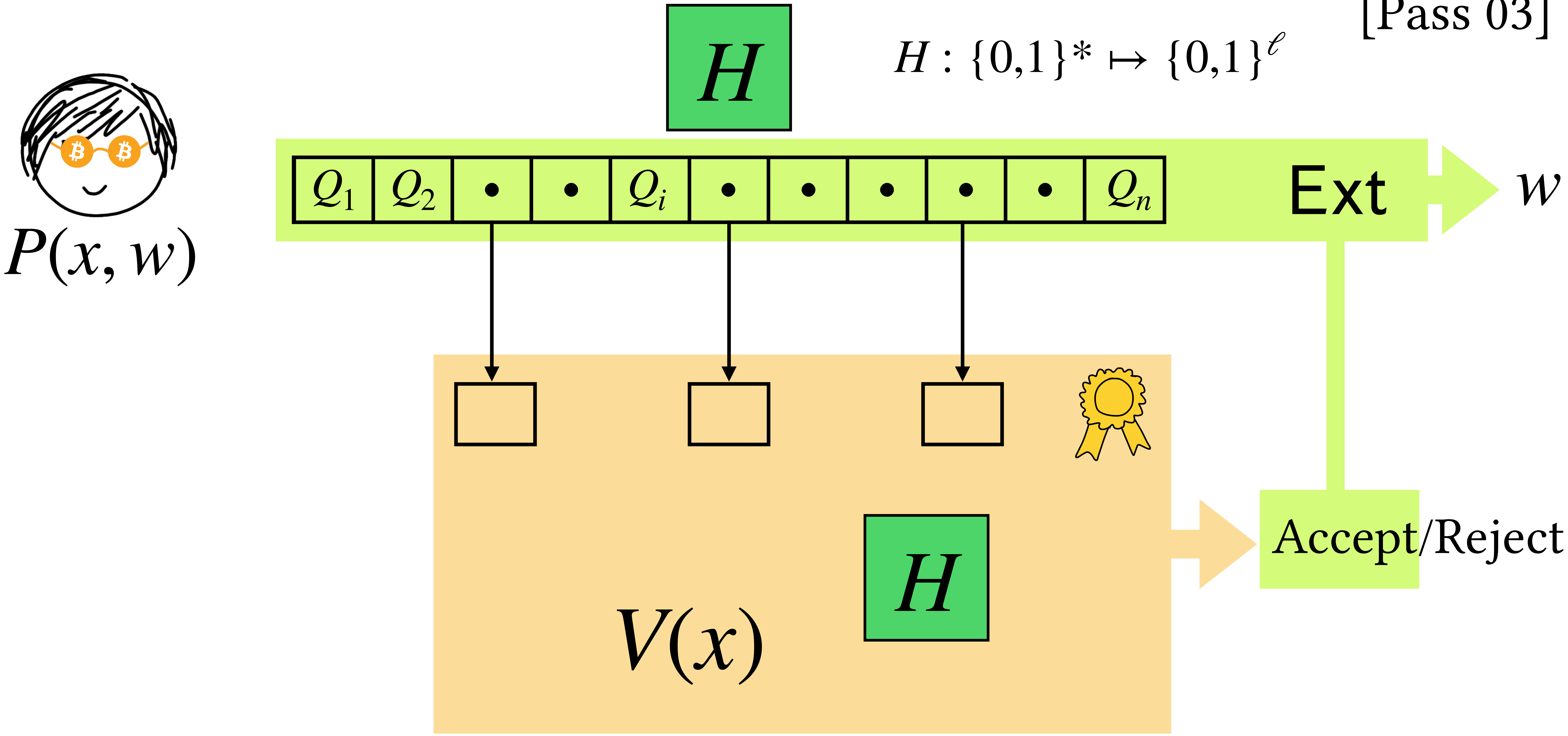
[Pass 03]

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



Random Oracles as Ext Privilege

[Pass 03]



Random Oracles as Ext Privilege

[Pass 03]

- Why is it a meaningful trapdoor?
  - Hash functions are complex and highly unstructured
  - Prover must "query" each Q_i to H to obtain $H(Q_i)$
- Practical usage:
  - No "trusted setup", each query is very cheap
  - Many NIZKs **happen to achieve** SLE in the ROM

Distributing NIZKs in the ROM

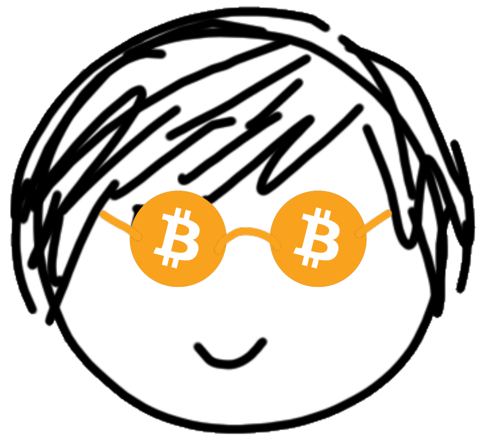
- Multiparty protocols to securely compute RO-based NIZKs should *ideally* make blackbox use of H
  - Conceptually: H should not have a circuit description
  - Practically: hash functions have large circuits
- We call them "Oracle Respecting Distributed" (ORD) protocols

Oracle Respecting Distribution is Leaky

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

Trimming Resilience

H



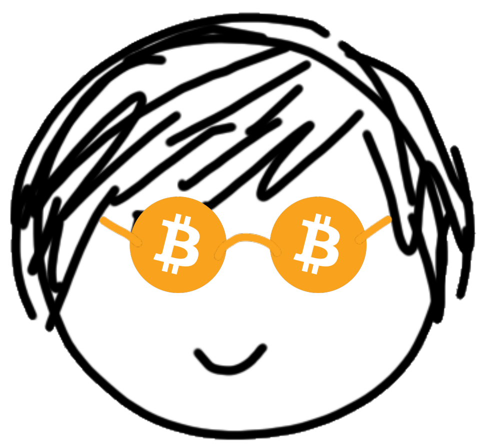
$P(x, w)$

Q_1	Q_2	Q_3	Q_4	Q_5	Q_6	Q_7	Q_8	Q_9

π

Trimming Resilience

H



$P(x, w)$

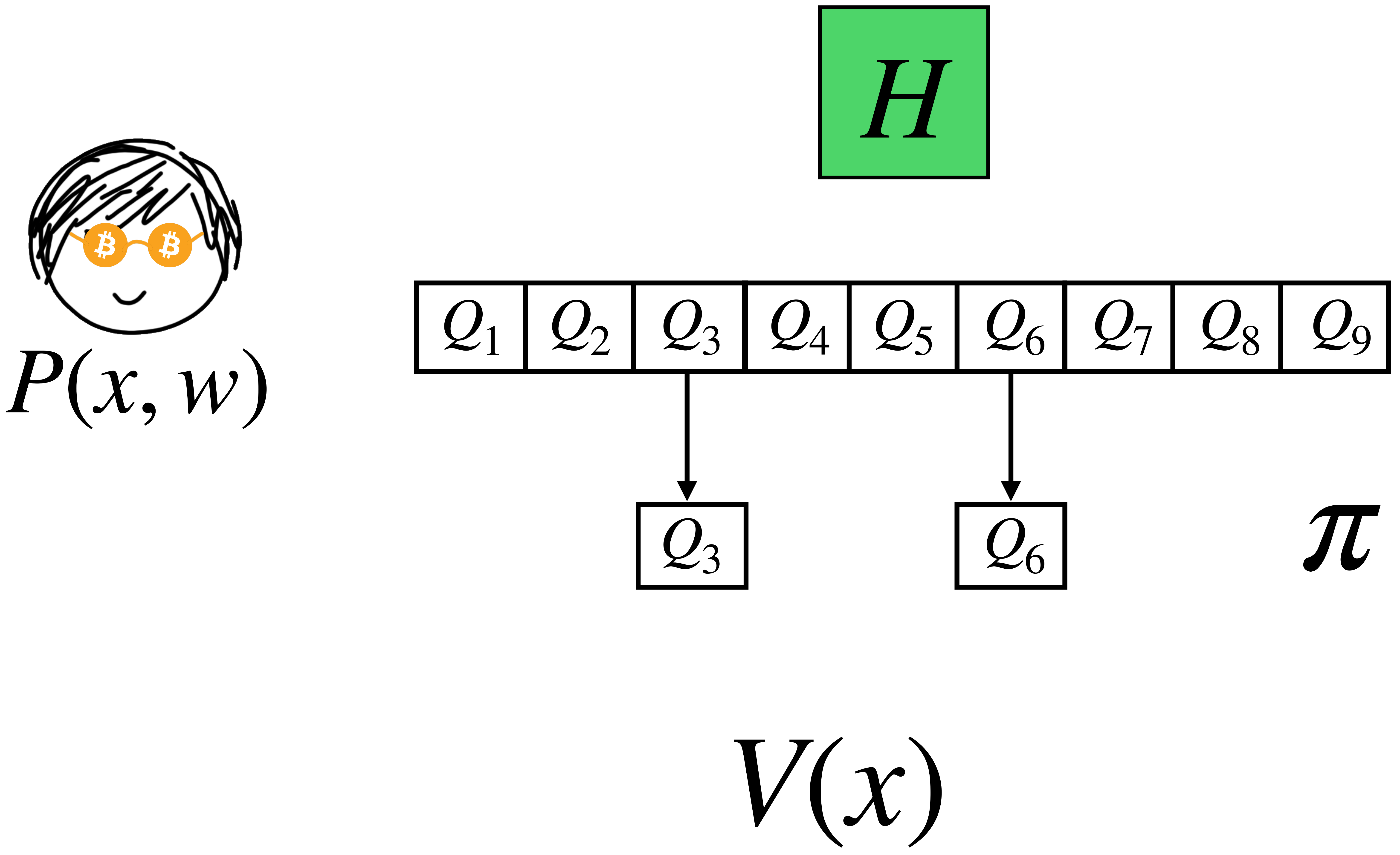
Q_1	Q_2	Q_3	Q_4	Q_5	Q_6	Q_7	Q_8	Q_9

π

$V(x)$

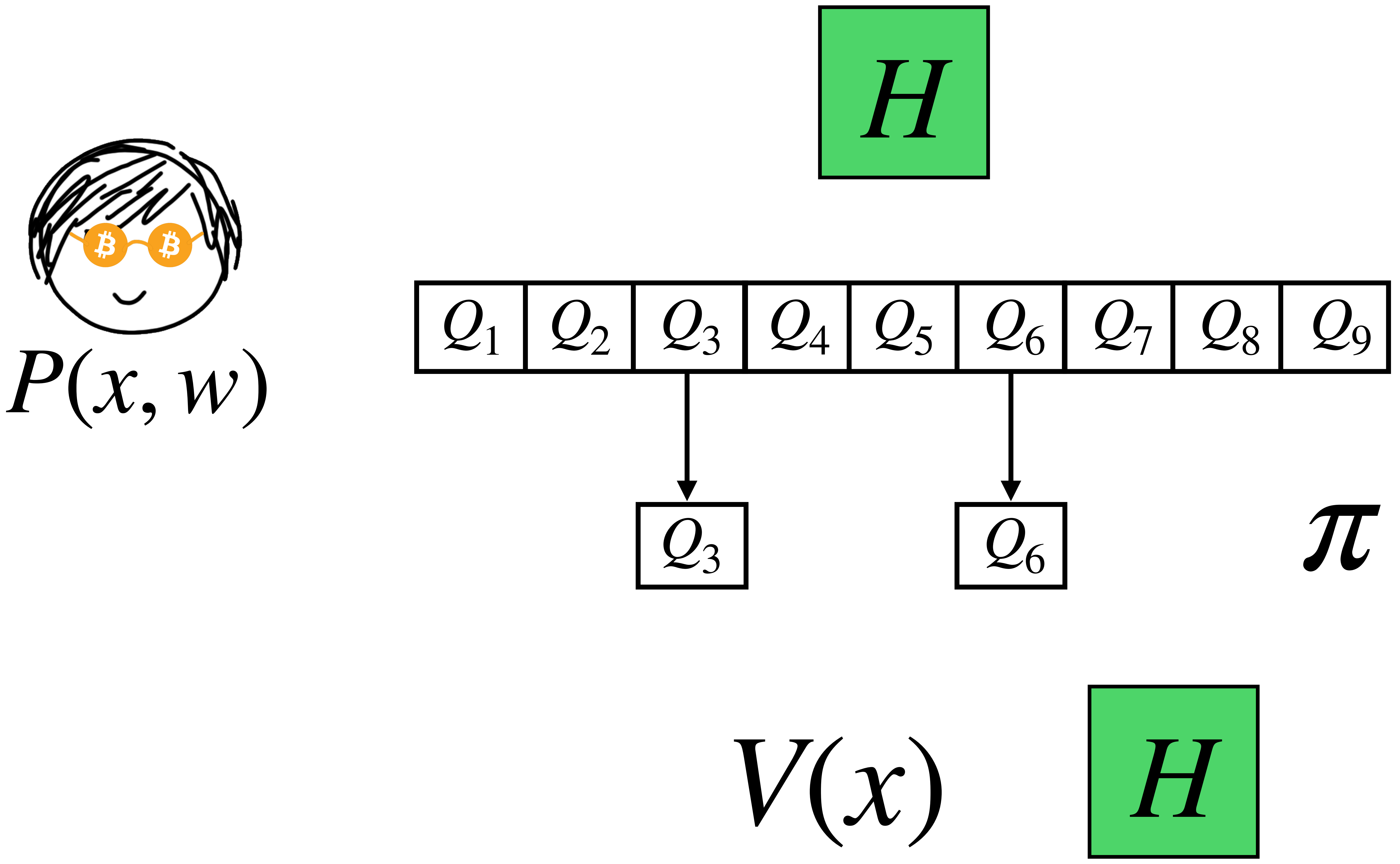
V checks at most $n = 2$ queries

Trimming Resilience



V checks at most $n = 2$ queries

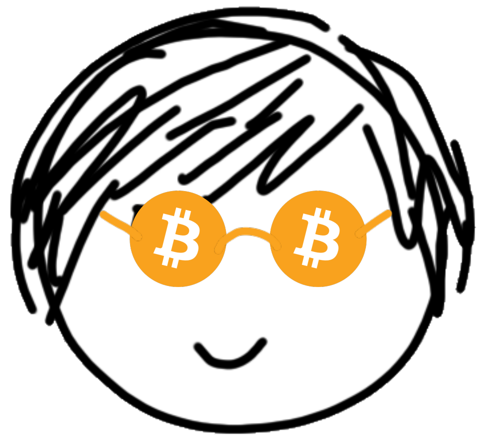
Trimming Resilience



V checks at most $n = 2$ queries

Trimming Resilience

H



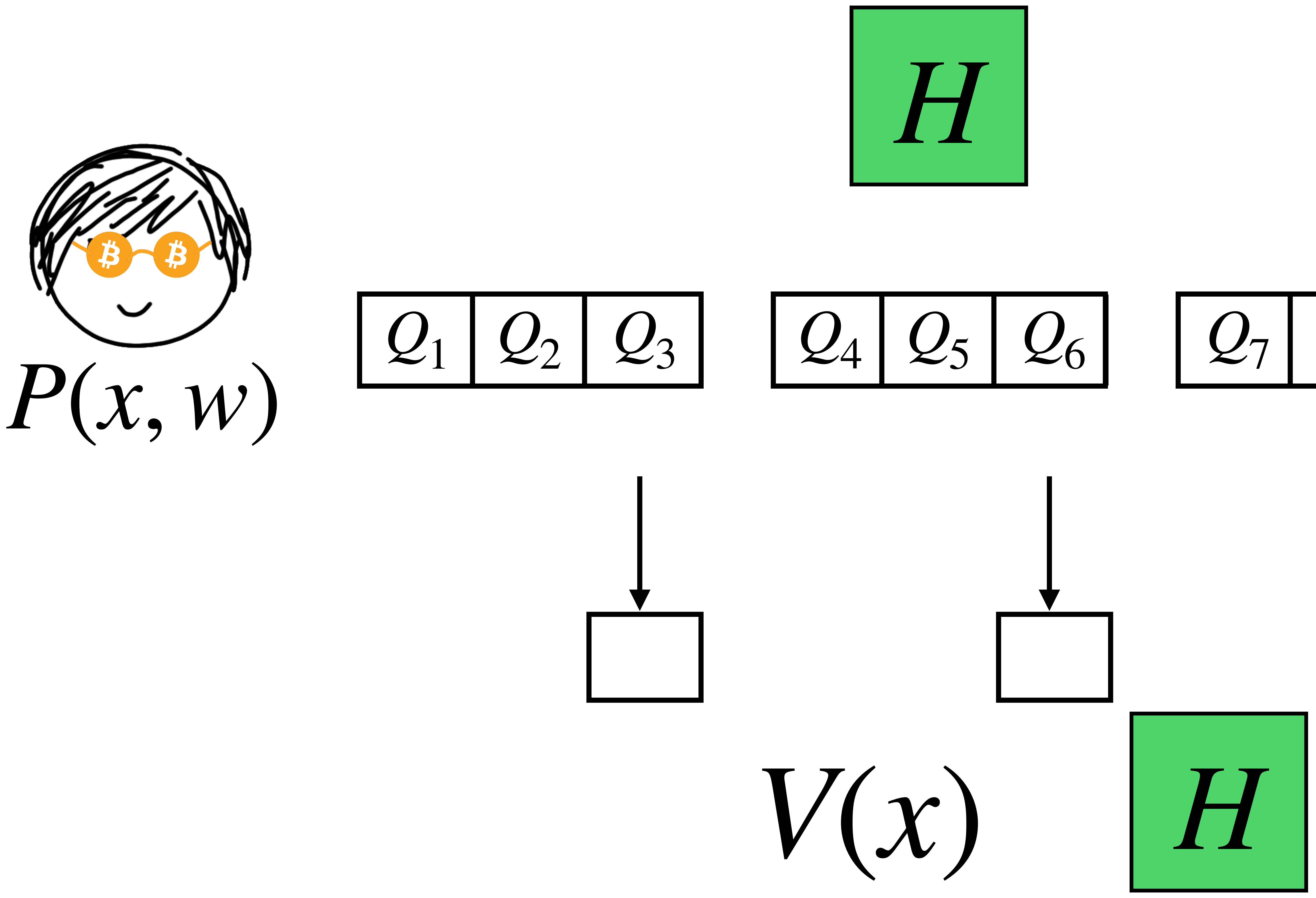
$P(x, w)$

Q_1 Q_2 Q_3 Q_4 Q_5 Q_6 Q_7 Q_8 Q_9

$V(x)$ H

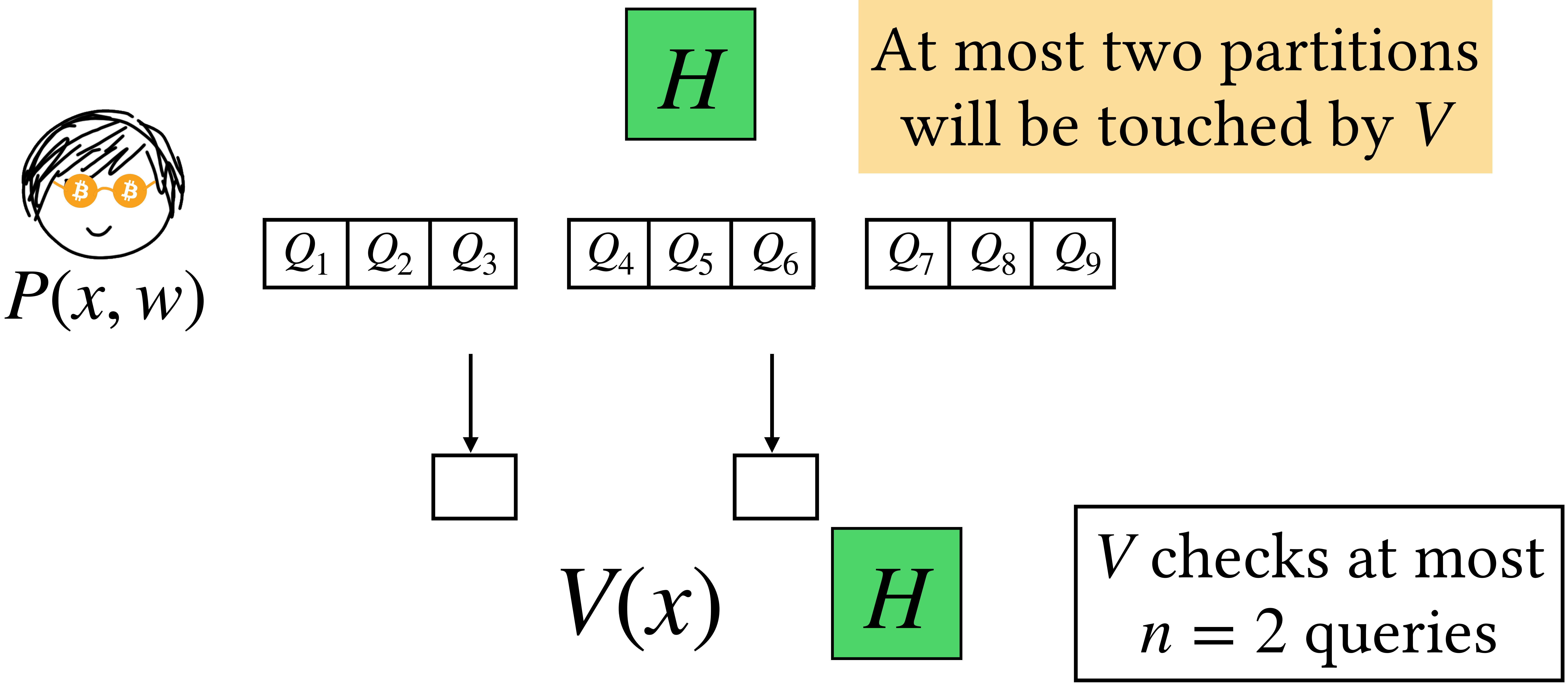
V checks at most $n = 2$ queries

Trimming Resilience

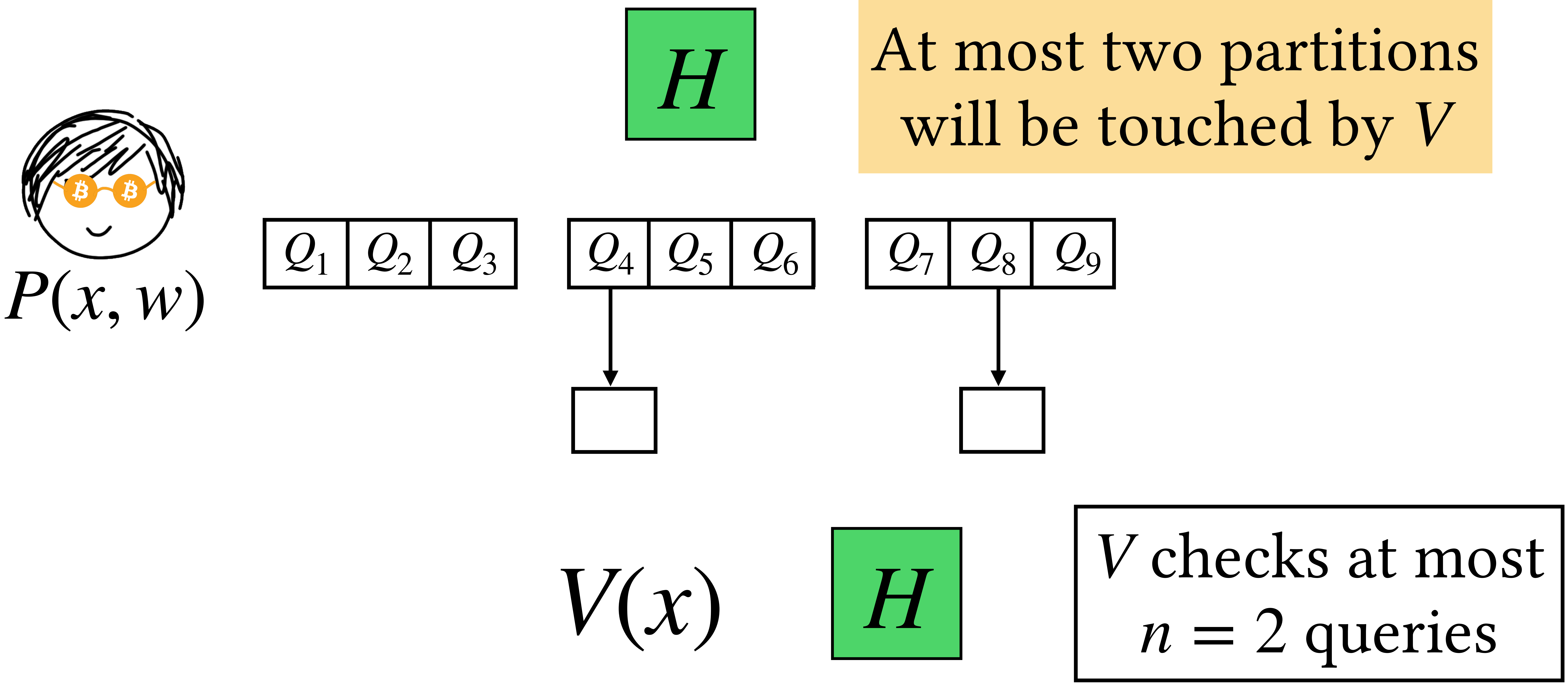


V checks at most $n = 2$ queries

Trimming Resilience



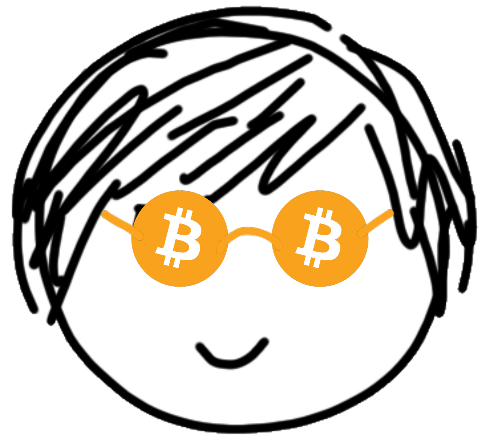
Trimming Resilience



Trimming Resilience

H

At most two partitions will be touched by V



$P(x, w)$

Q_1	Q_2	Q_3

Q_4	Q_5	Q_6

Q_7	Q_8	Q_9

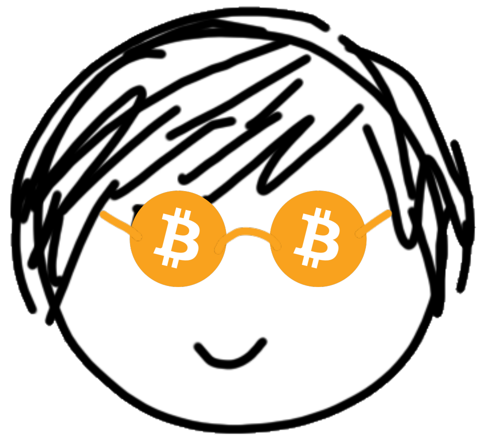
Randomly selected partition:
$\Pr[\text{untouched by } V] \geq 1/3$

$V(x)$

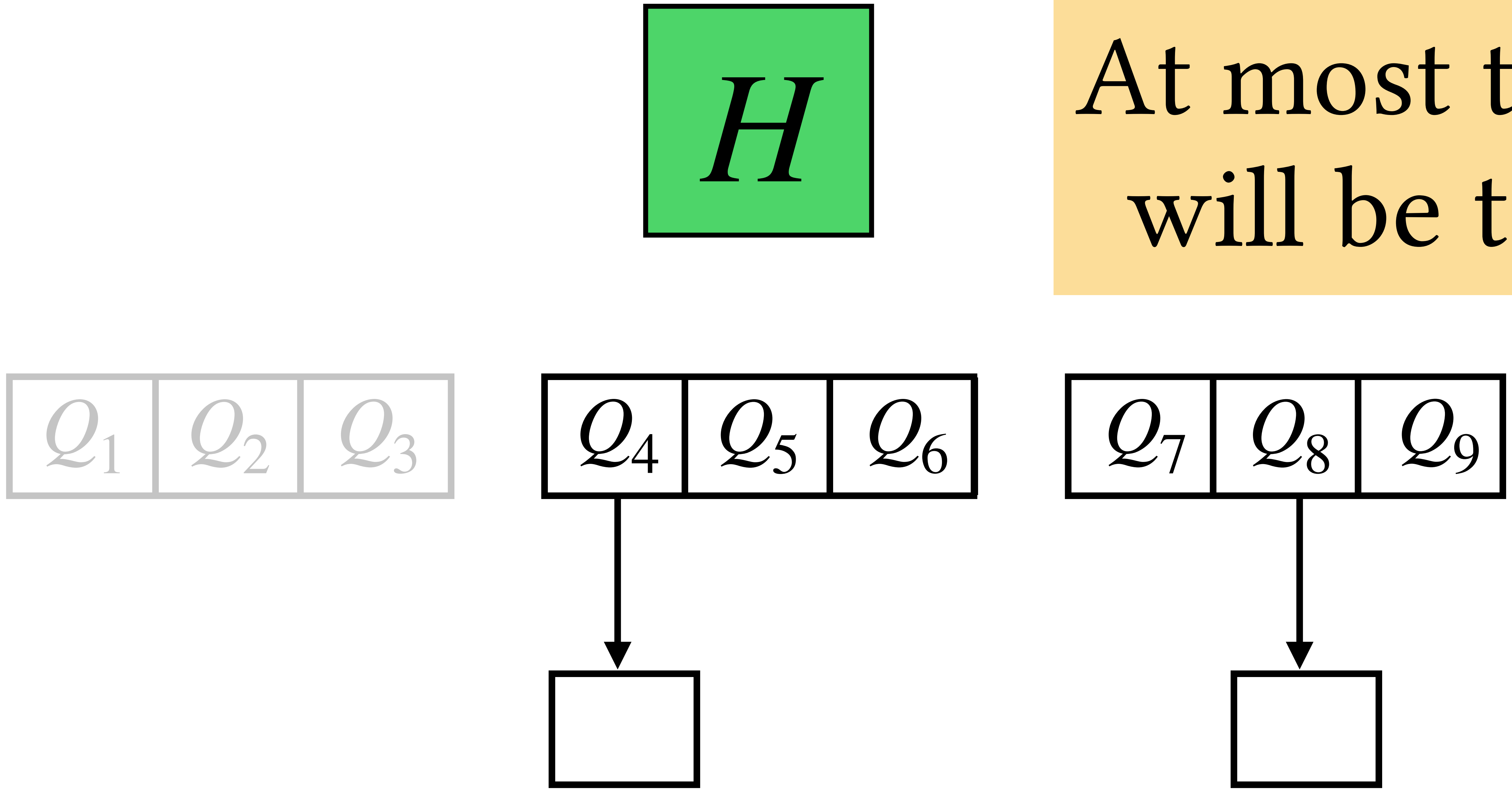
H

V checks at most $n = 2$ queries

Trimming Resilience



$P(x, w)$



At most two partitions will be touched by V

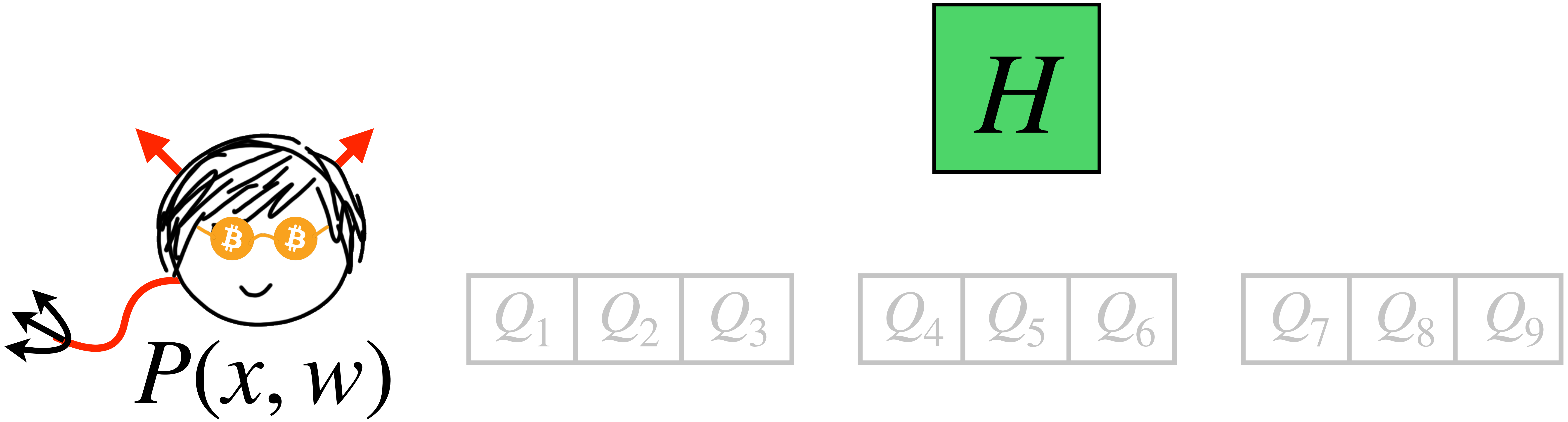
Randomly selected partition:
Pr[untouched by V] $\geq 1/3$

$V(x)$

H

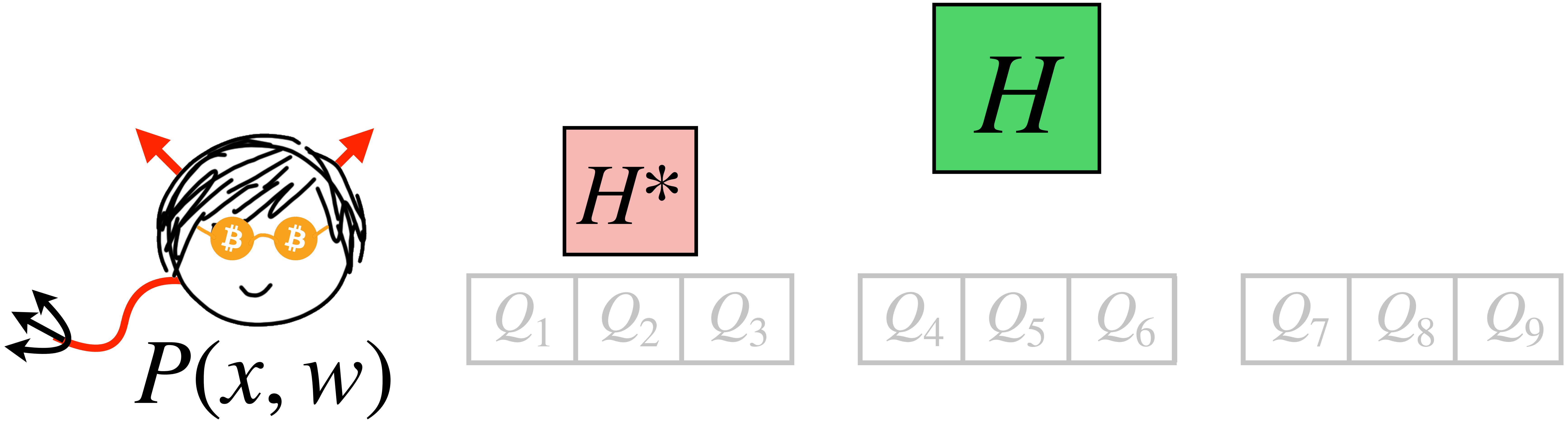
V checks at most $n = 2$ queries

Trimming Resilience



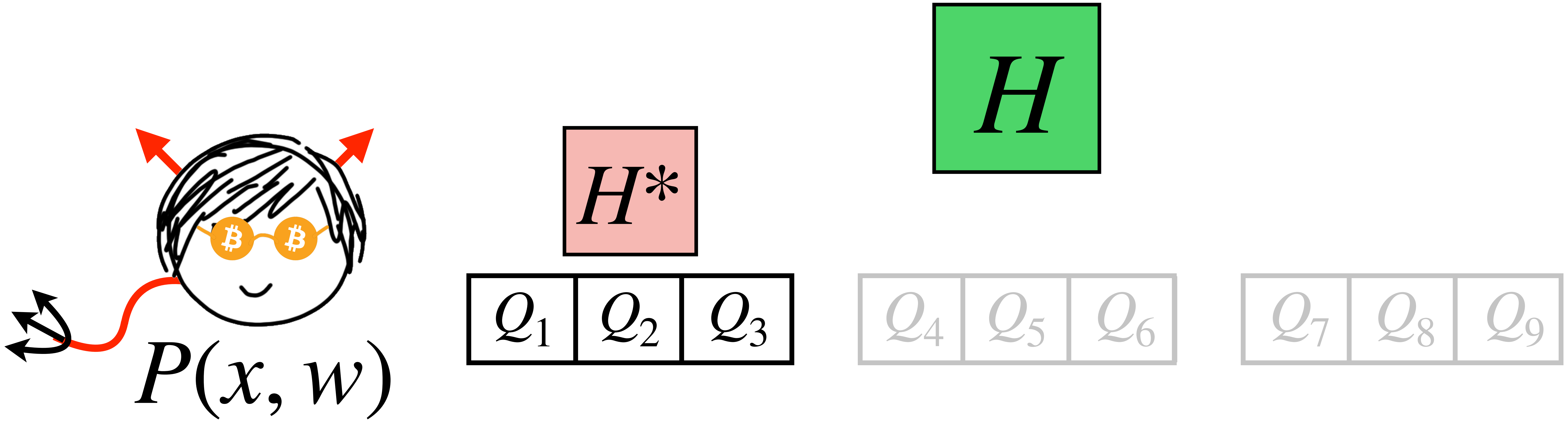
V checks at most $n = 2$ queries

Trimming Resilience



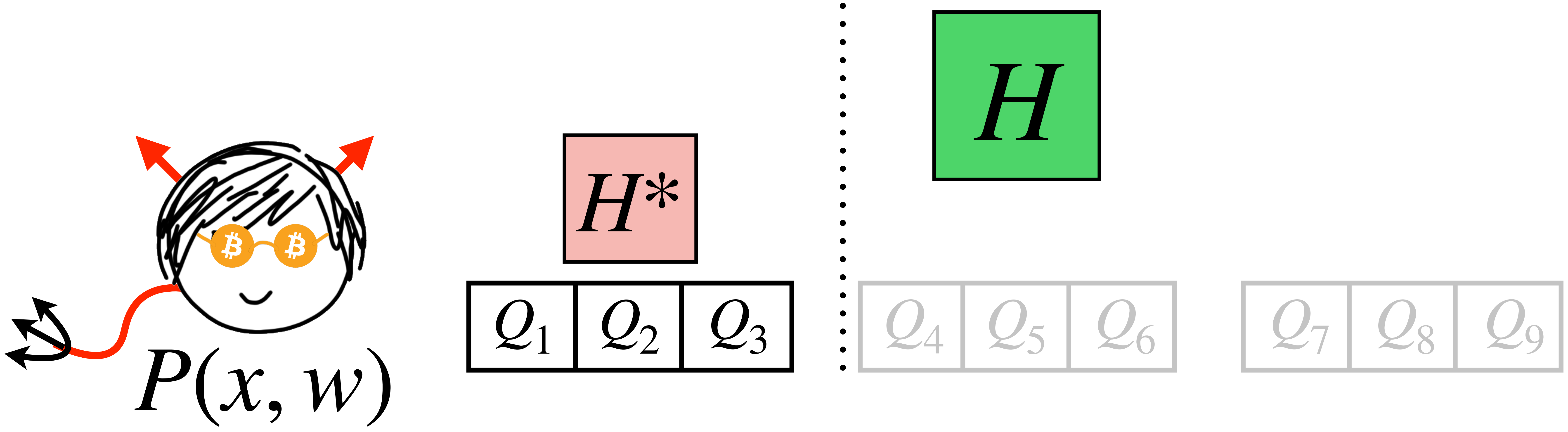
V checks at most $n = 2$ queries

Trimming Resilience



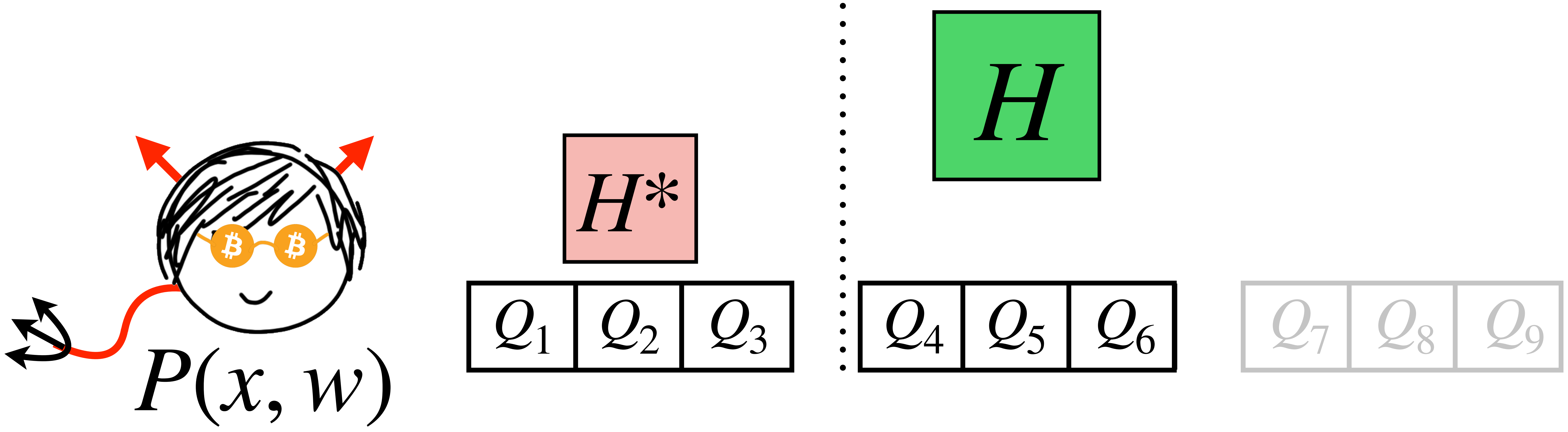
V checks at most
$n = 2$ queries

Trimming Resilience



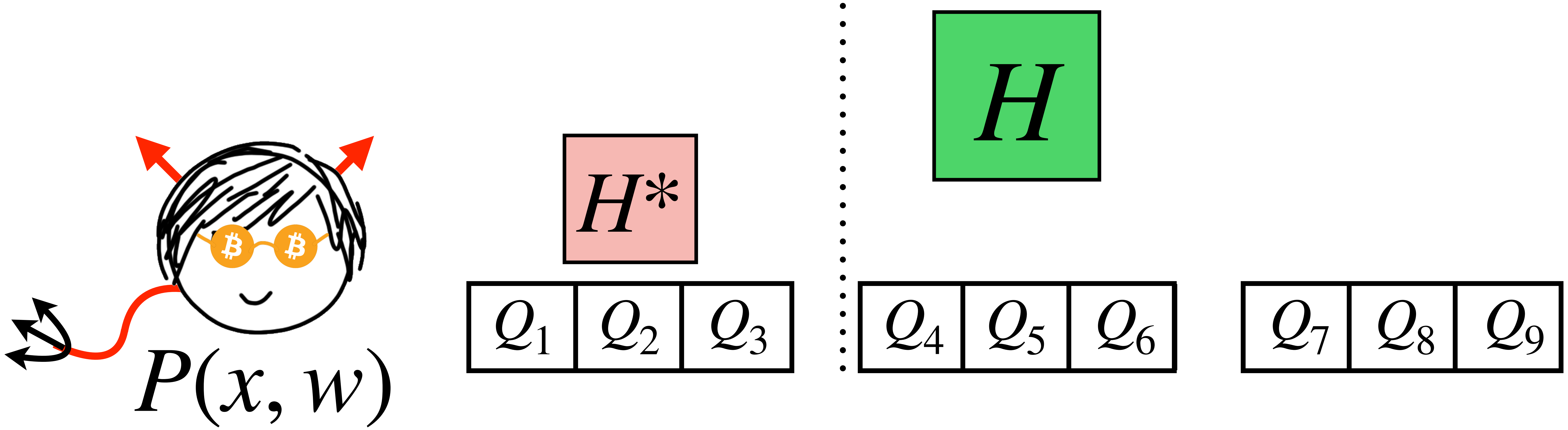
V checks at most $n = 2$ queries

Trimming Resilience



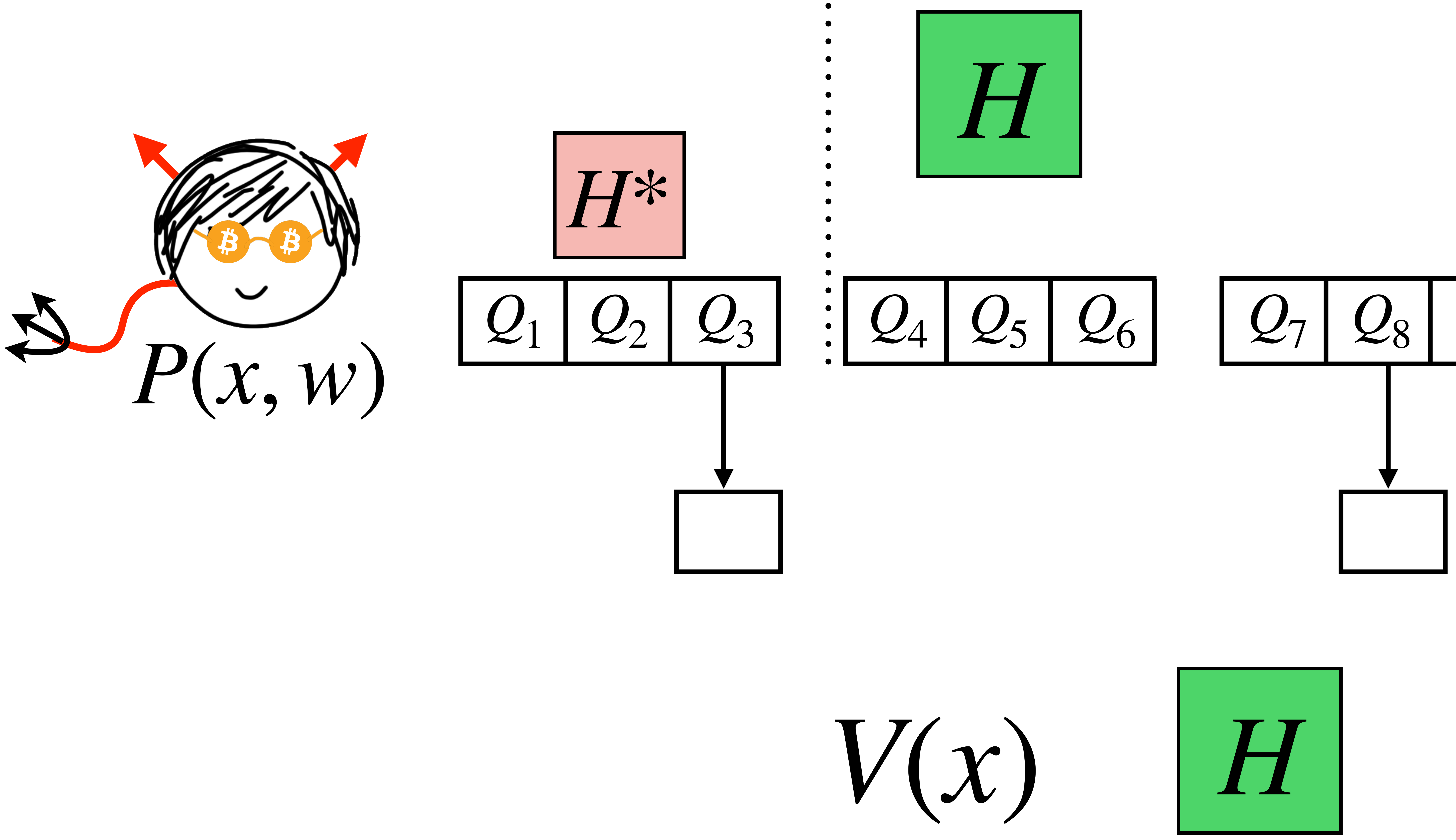
V checks at most $n = 2$ queries

Trimming Resilience



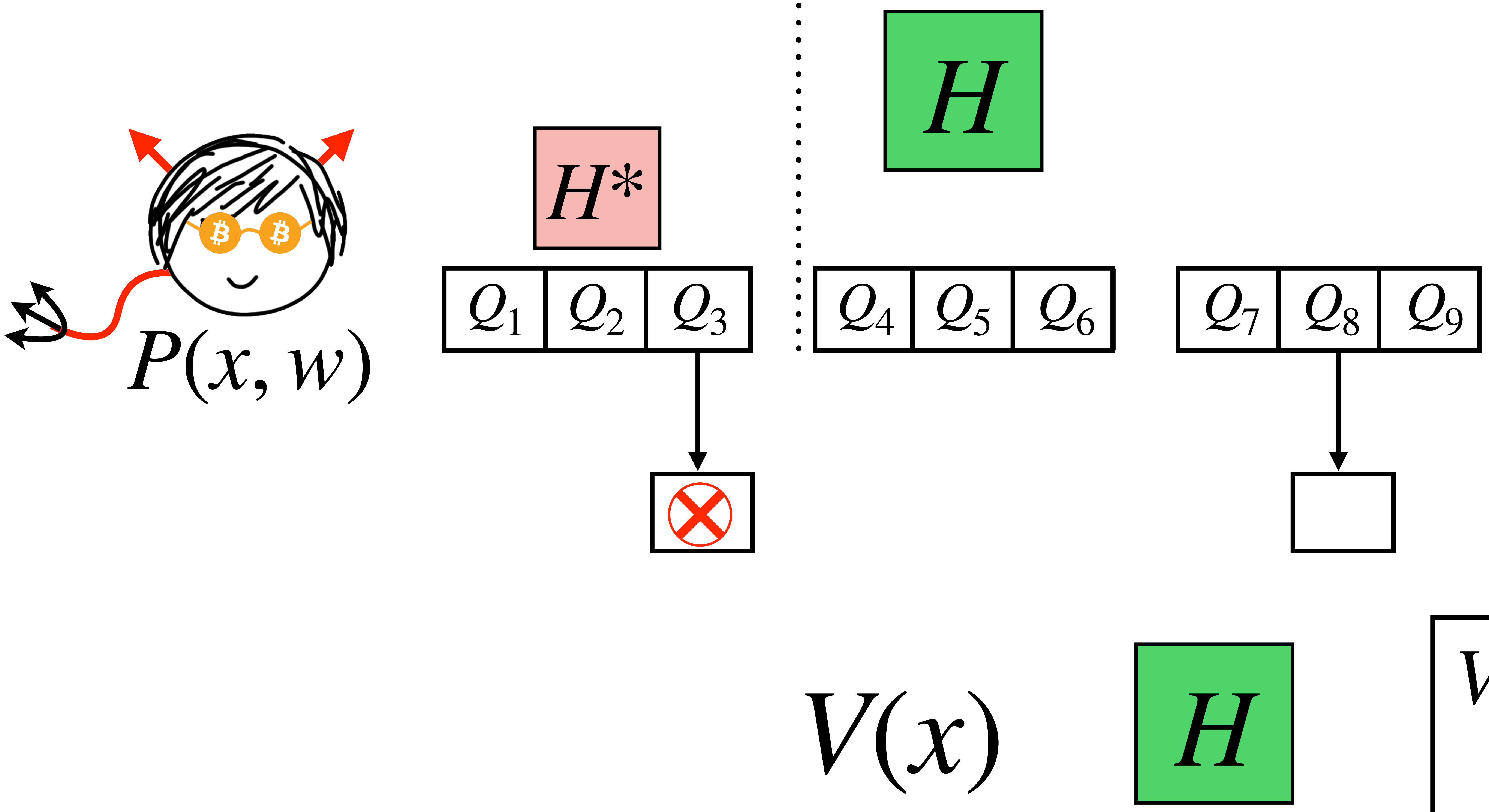
V checks at most
$n = 2$ queries

Trimming Resilience



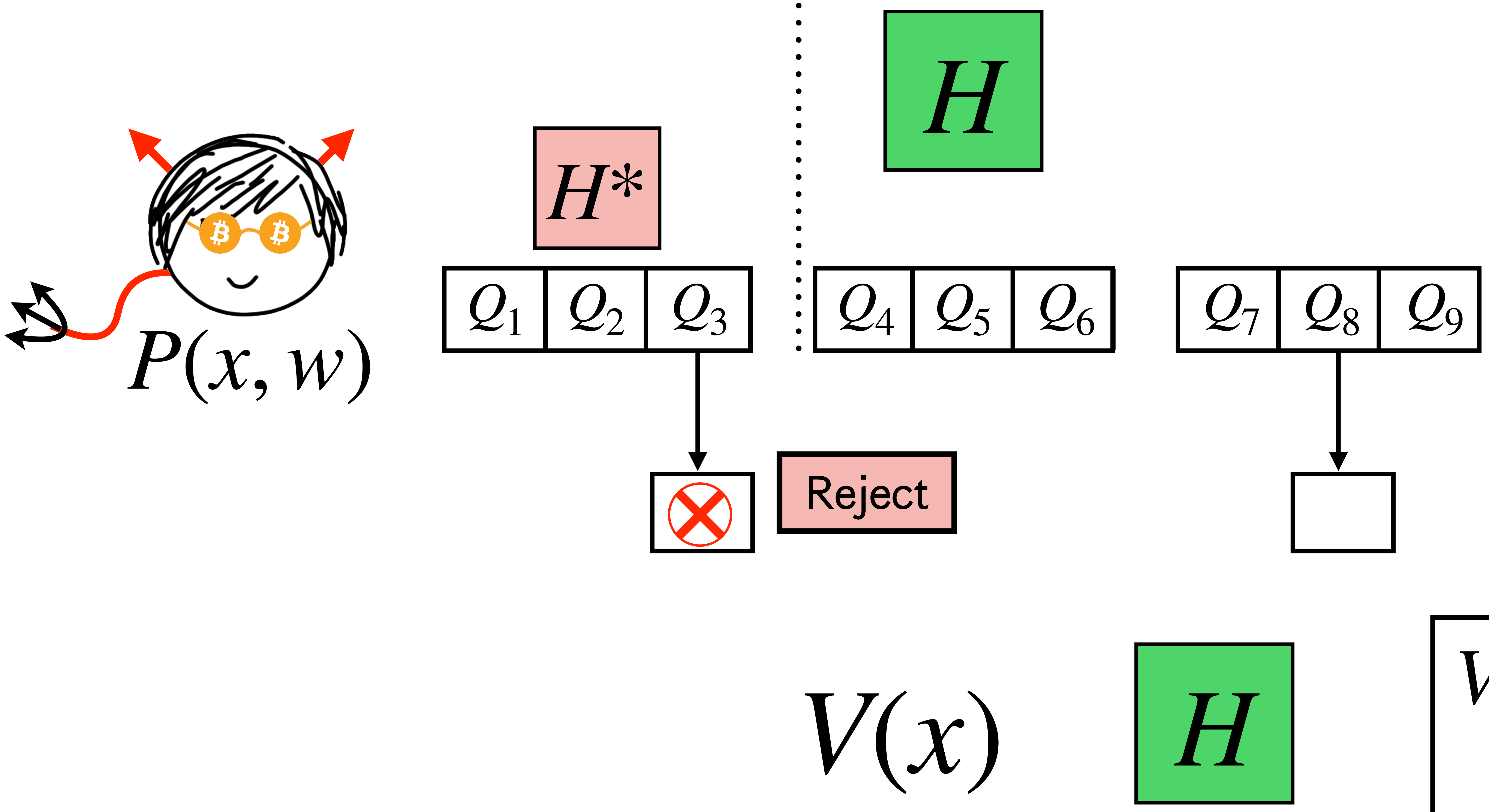
V checks at most $n = 2$ queries

Trimming Resilience



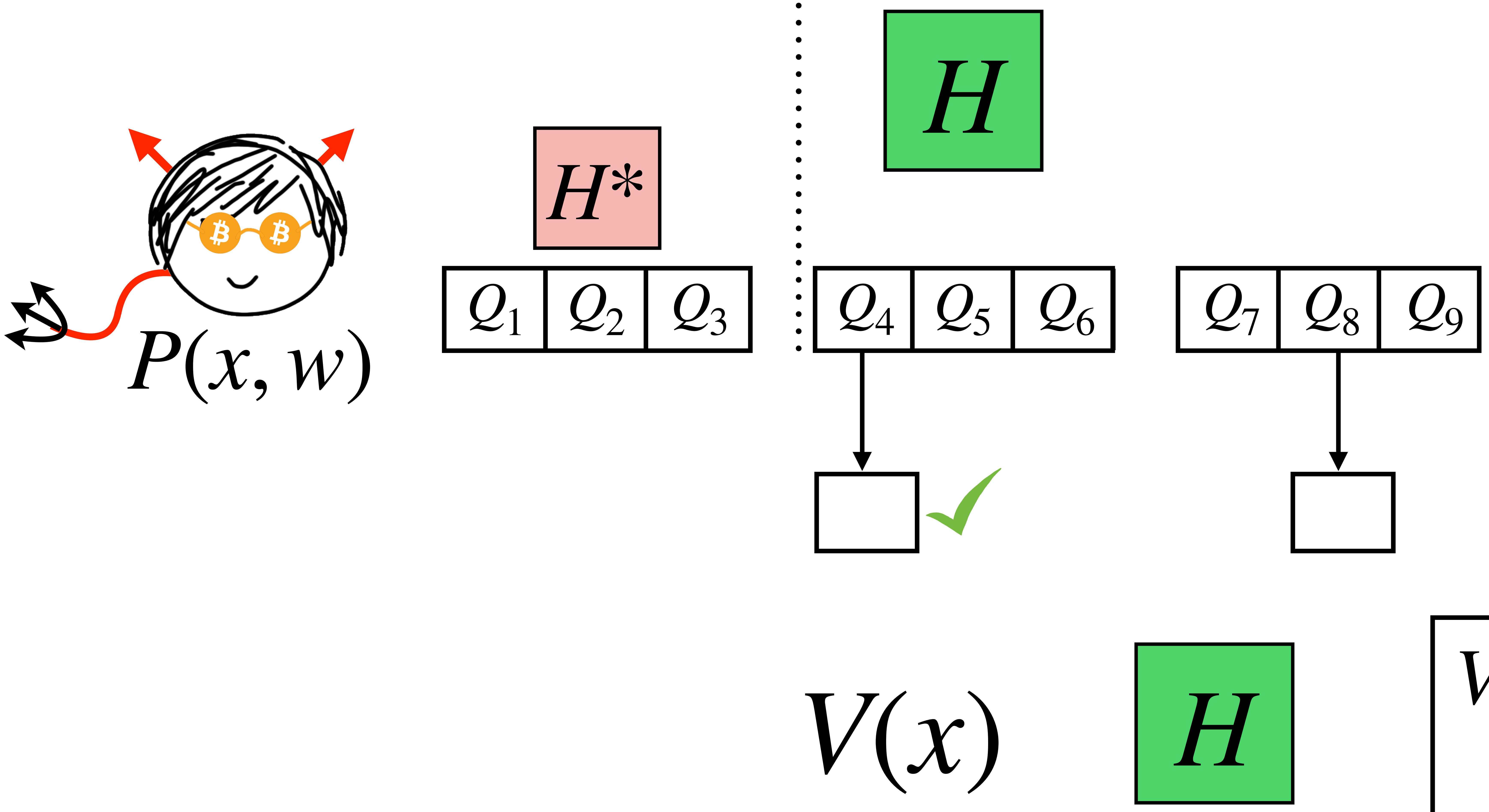
V checks at most $n = 2$ queries

Trimming Resilience



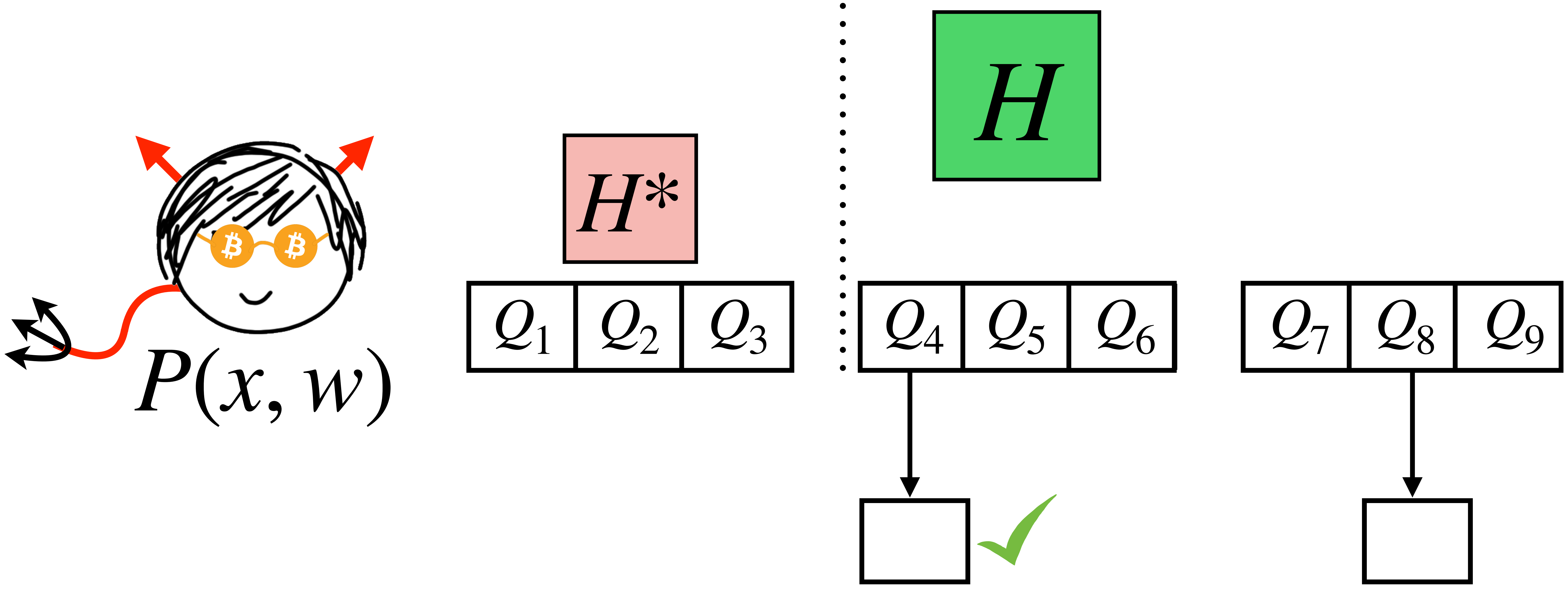
V checks at most $n = 2$ queries

Trimming Resilience



V checks at most $n = 2$ queries

Trimming Resilience



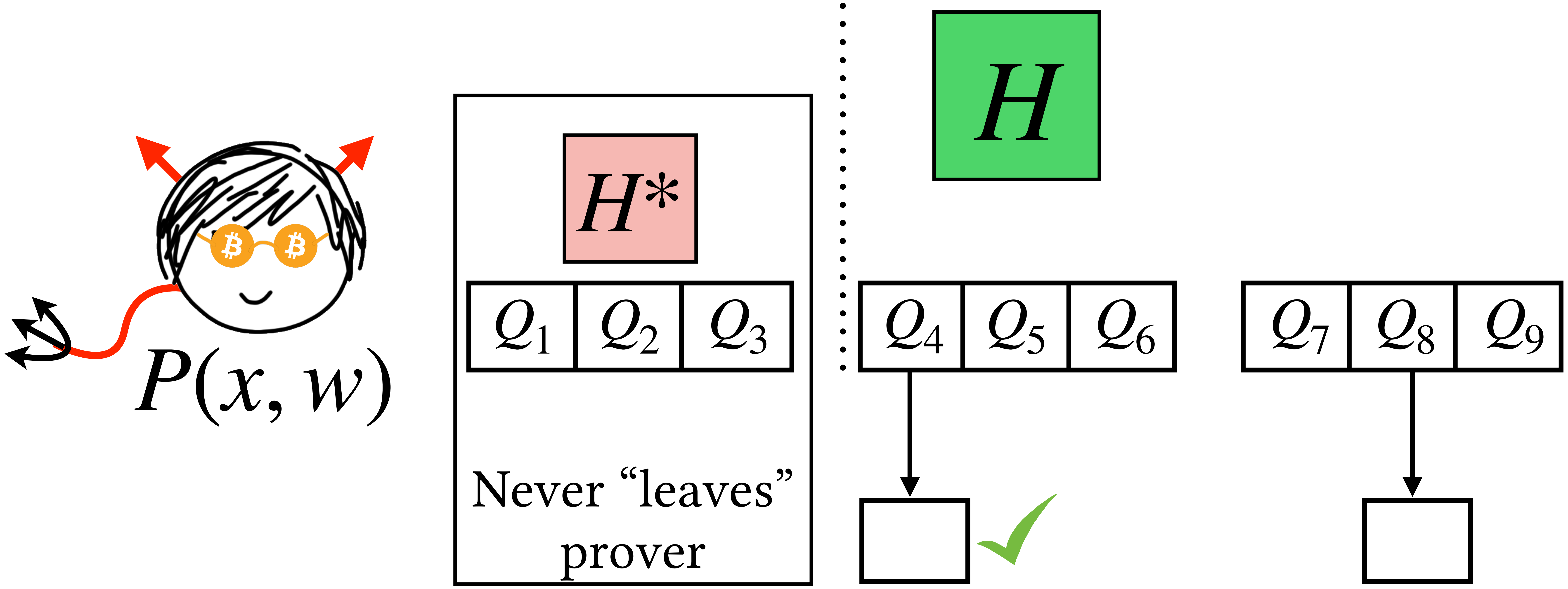
Pr[V accepts] $\geq 1/3$

$V(x)$

H

V checks at most $n = 2$ queries

Trimming Resilience



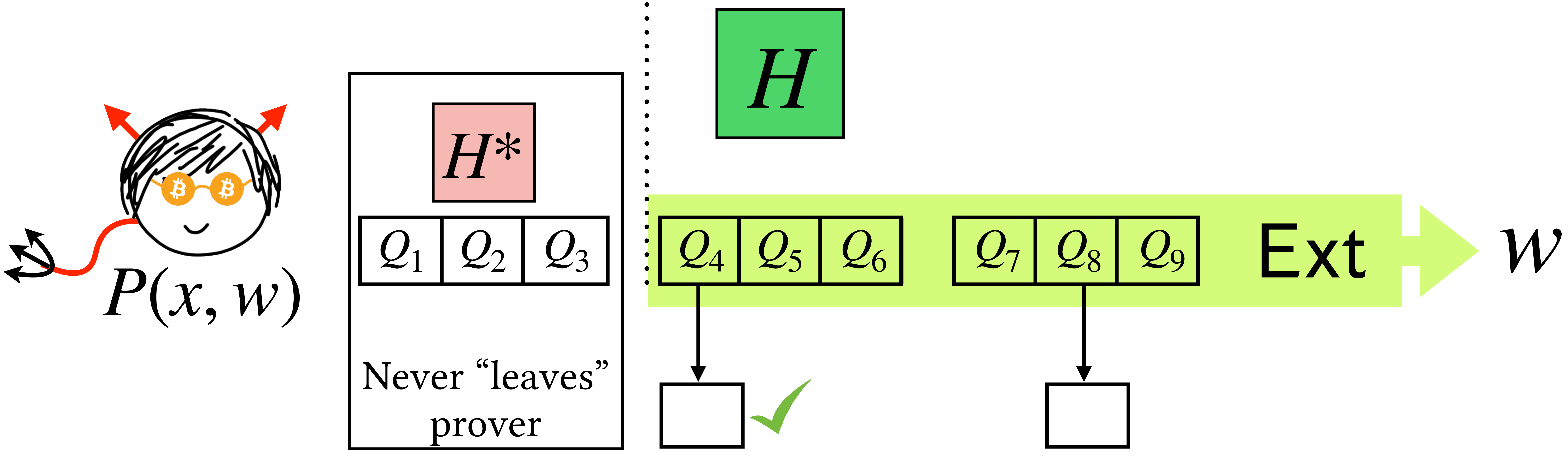
$\Pr[V \text{ accepts}] \geq 1/3$

$V(x)$

H

V checks at most $n = 2$ queries

Trimming Resilience



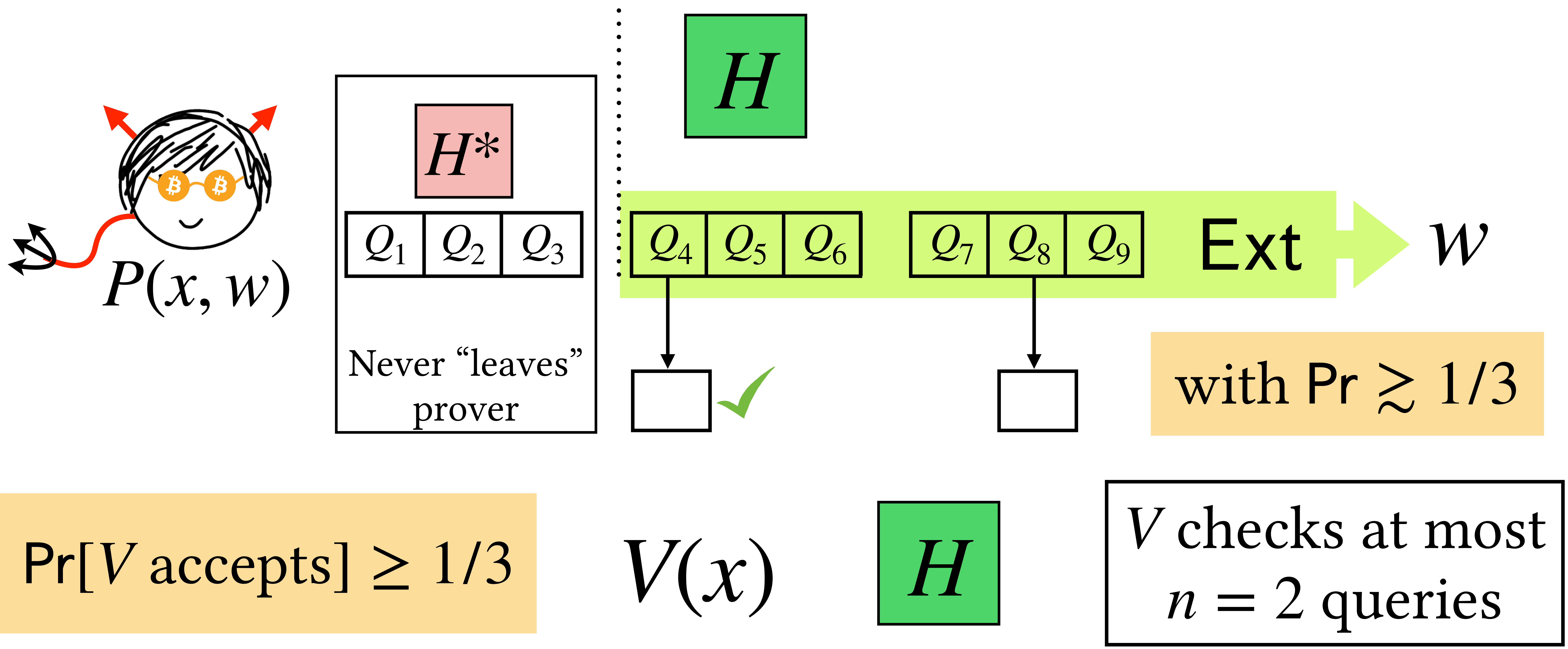
Pr[V accepts] $\geq 1/3$

$V(x)$

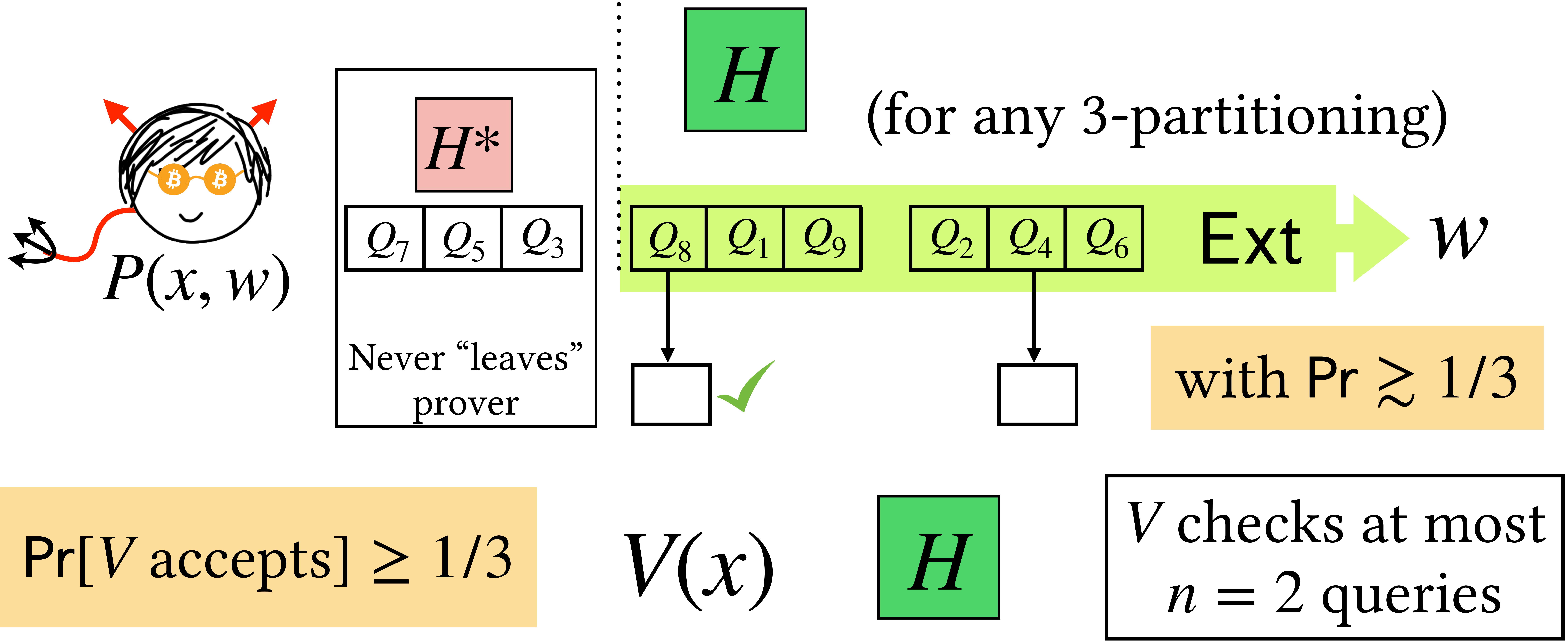
H

V checks at most $n = 2$ queries

Trimming Resilience

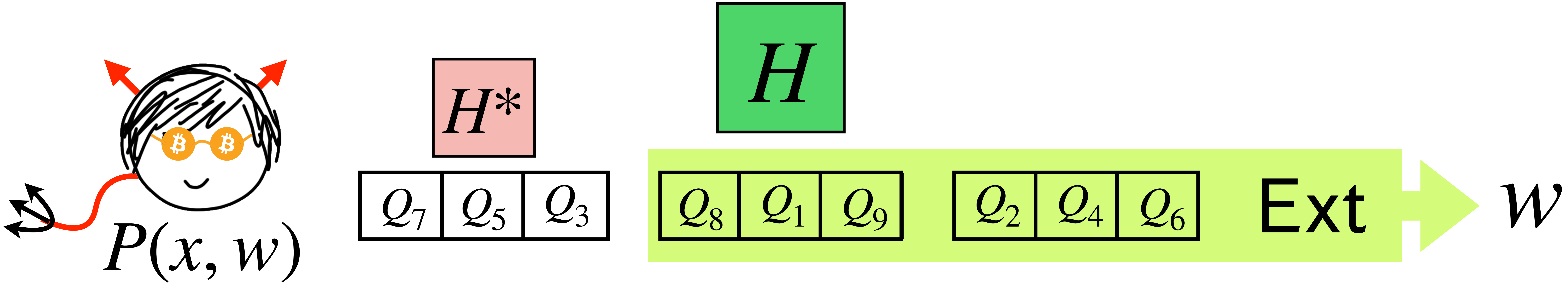


Trimming Resilience



Trimming Resilience

Lemma: For any $n + 1$-partitioning of RO queries, omitting *one* partition still allows extraction

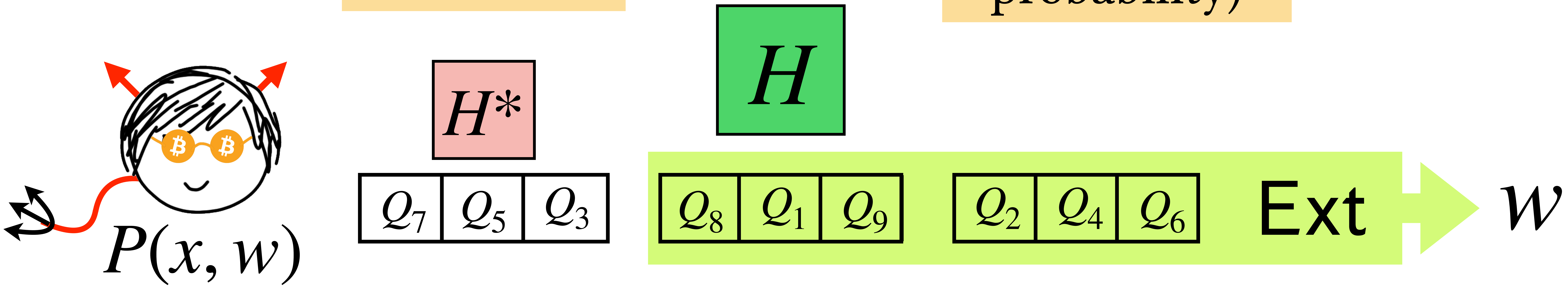


Trimming Resilience

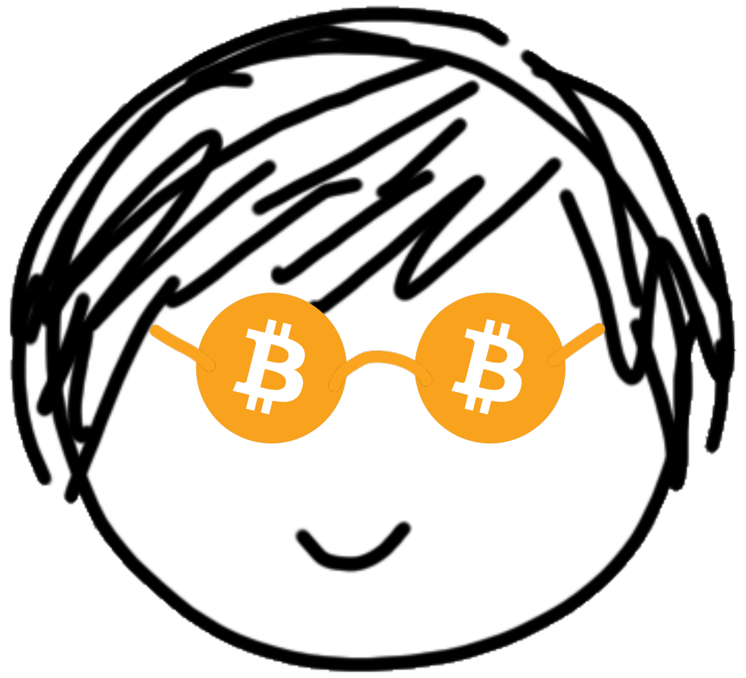
Lemma: For any $n+1$-partitioning of RO queries, omitting *one* partition still allows extraction

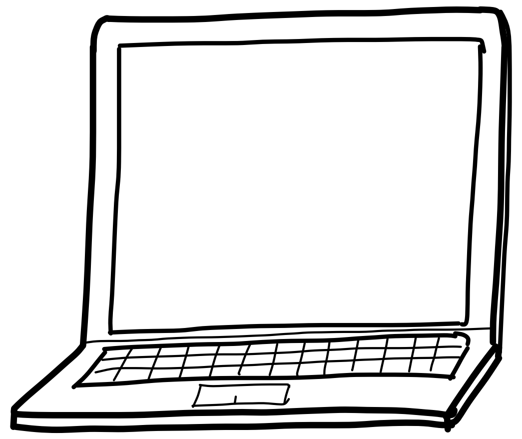
(random)

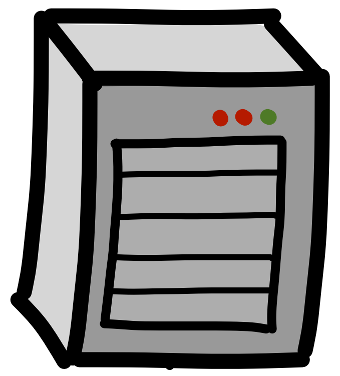
(w. noticeable probability)



Oracle Respecting Distribution



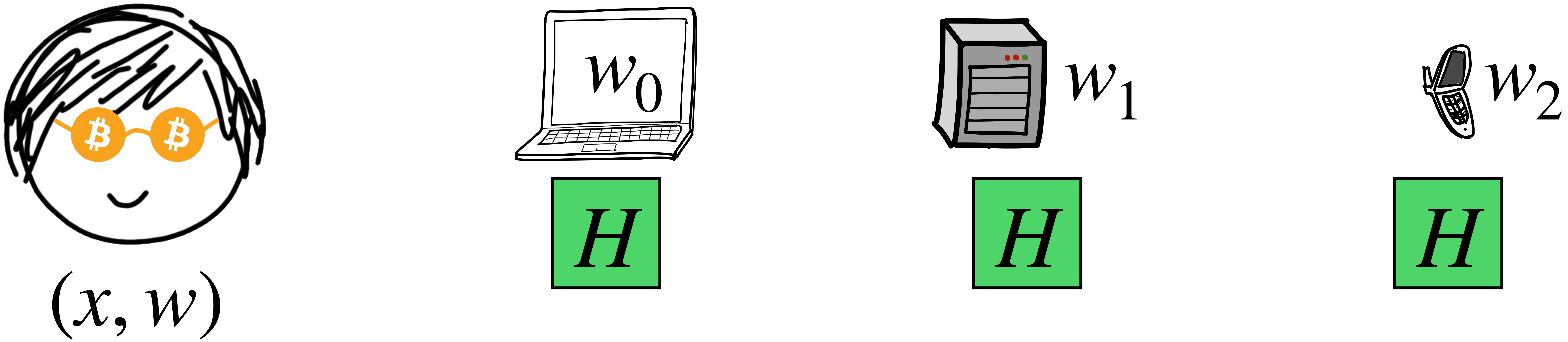




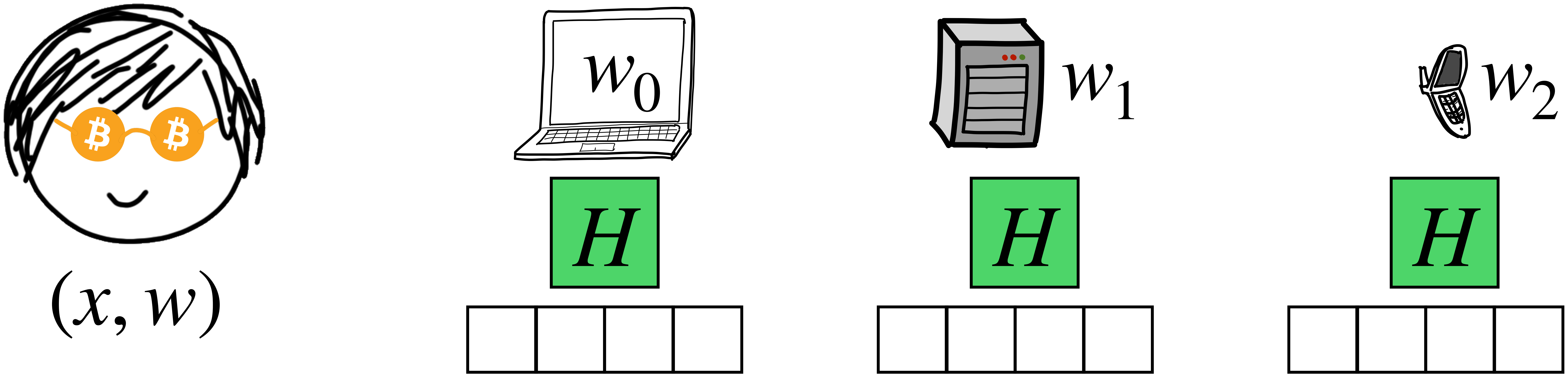
(x, w)

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

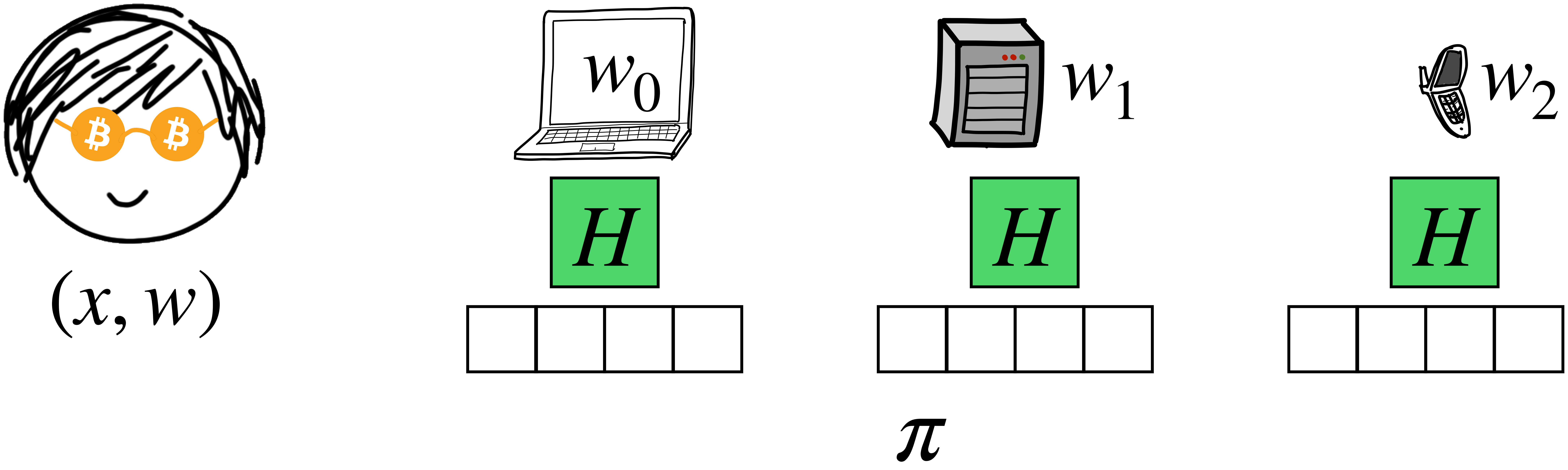
Oracle Respecting Distribution



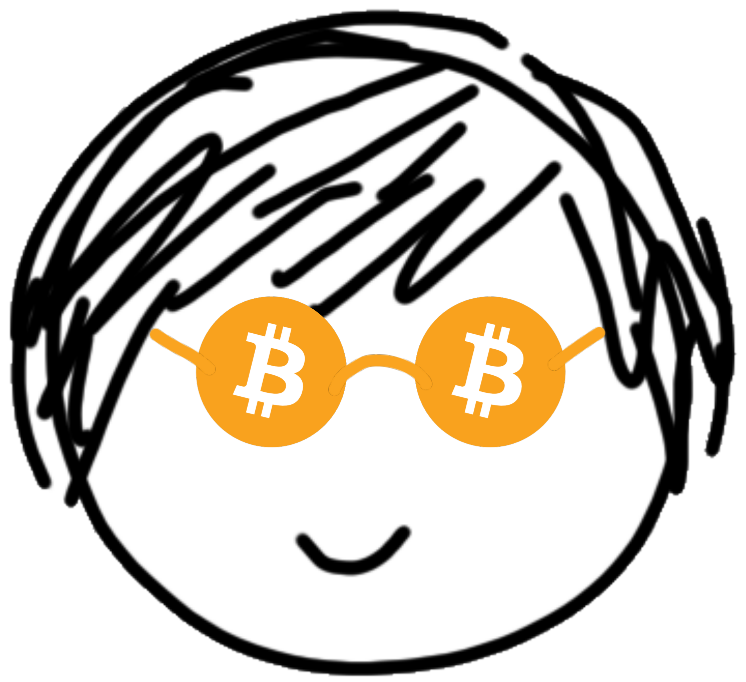
Oracle Respecting Distribution



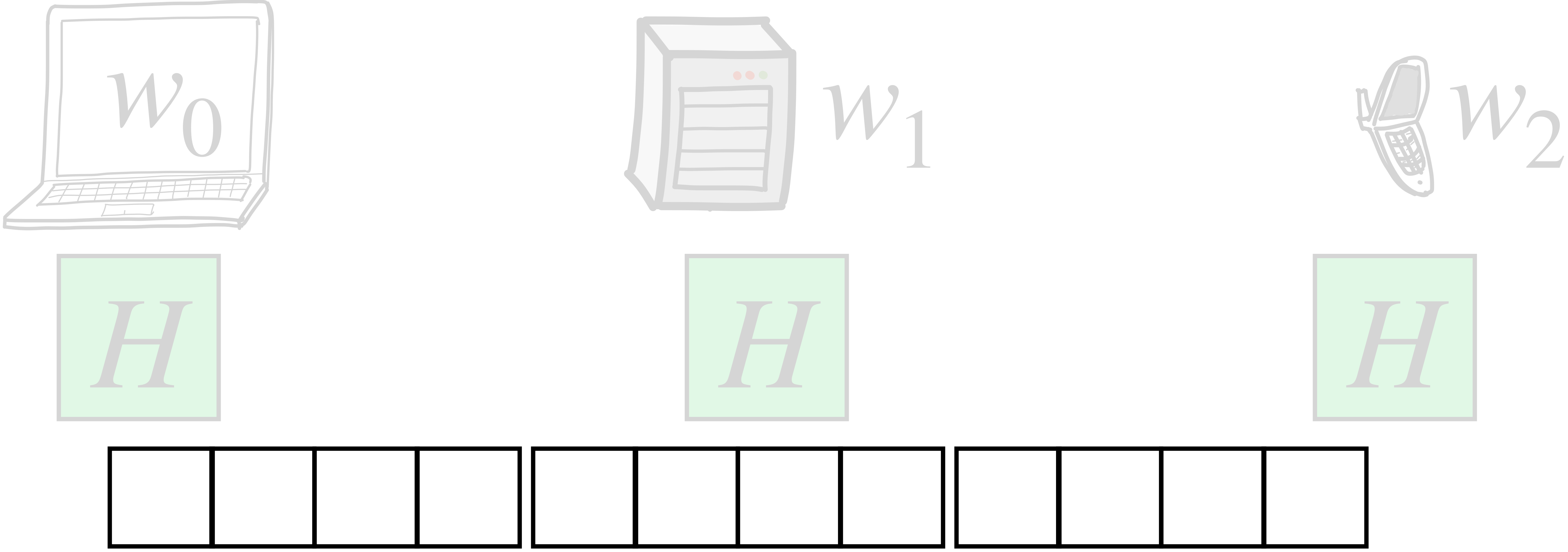
Oracle Respecting Distribution



Oracle Respecting Distribution



(x, w)



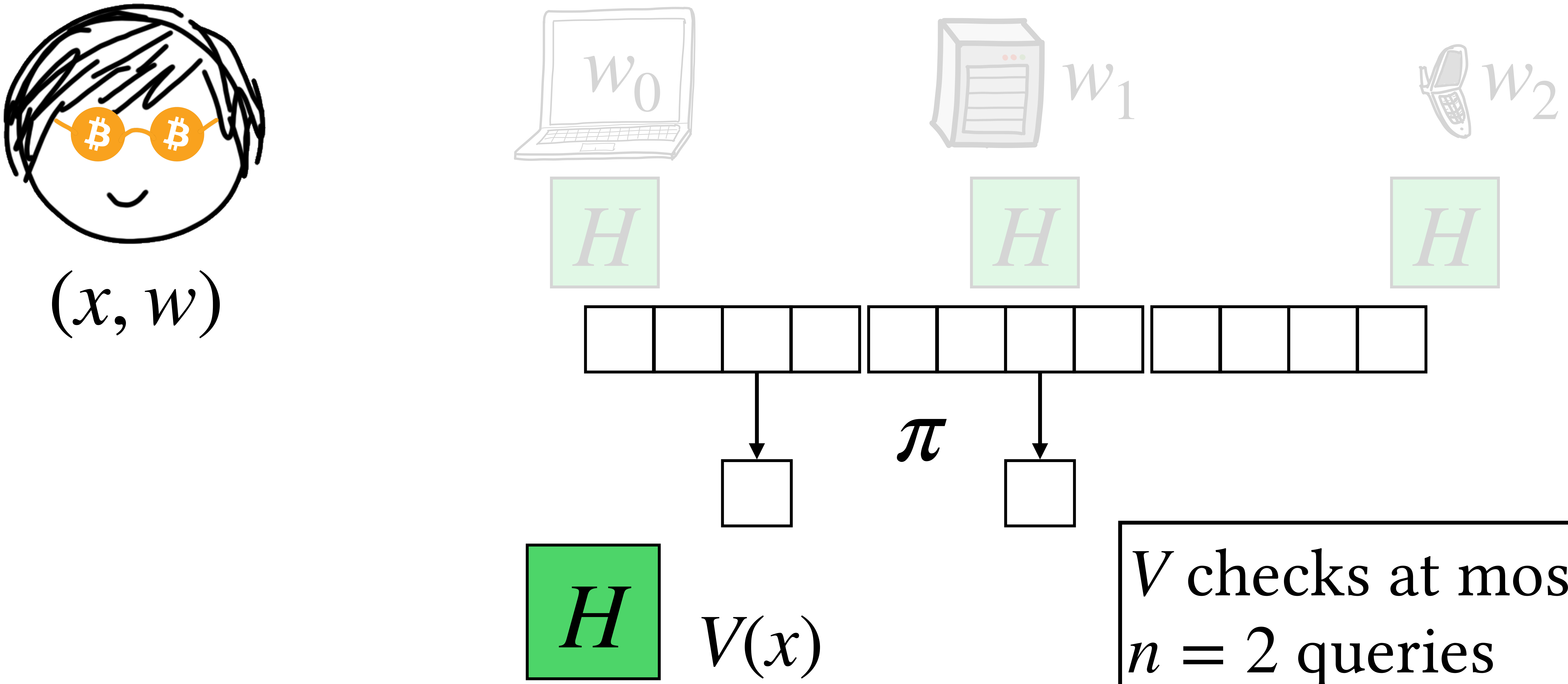
π

H

$V(x)$

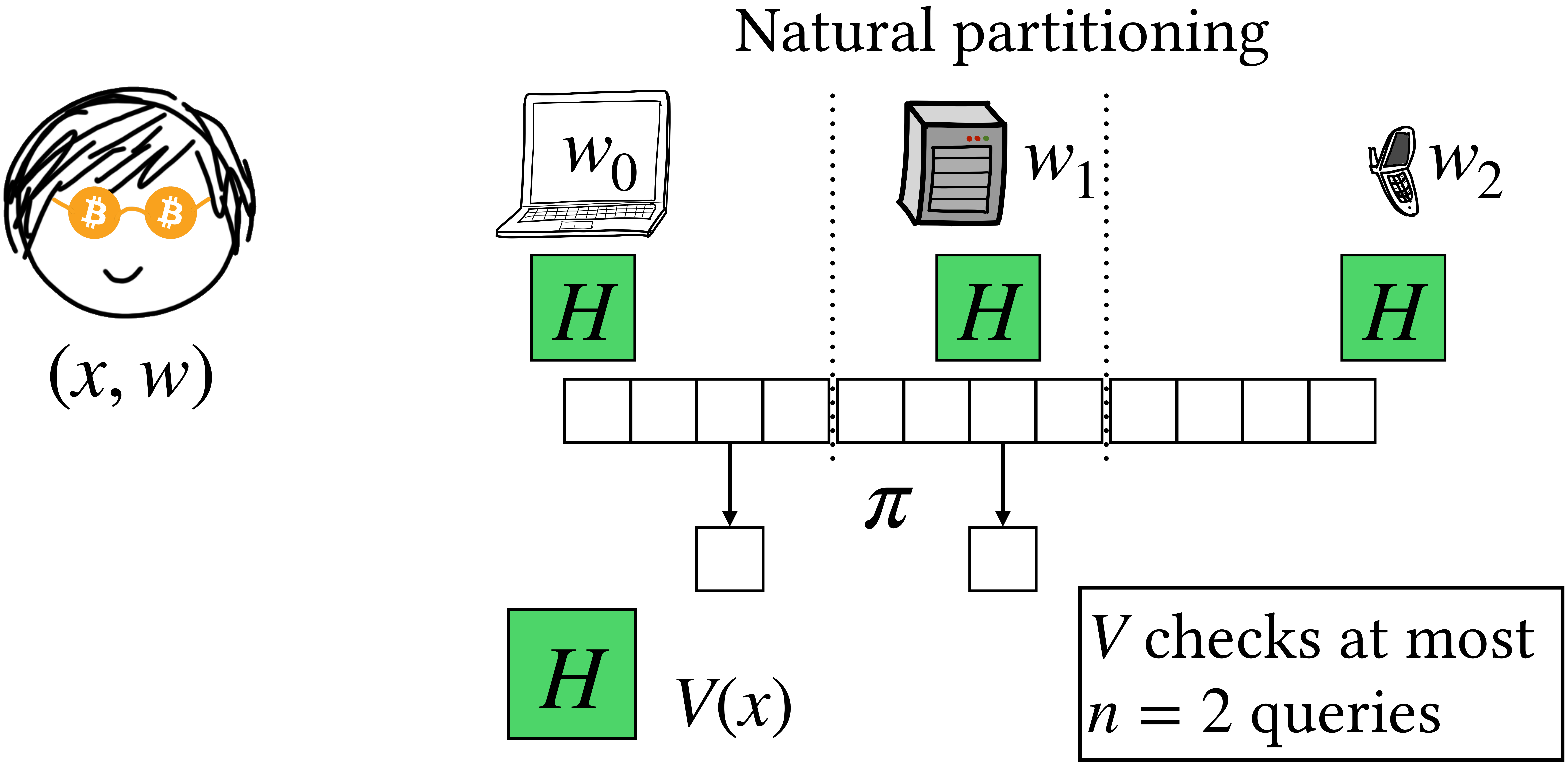
V checks at most $n = 2$ queries

Oracle Respecting Distribution



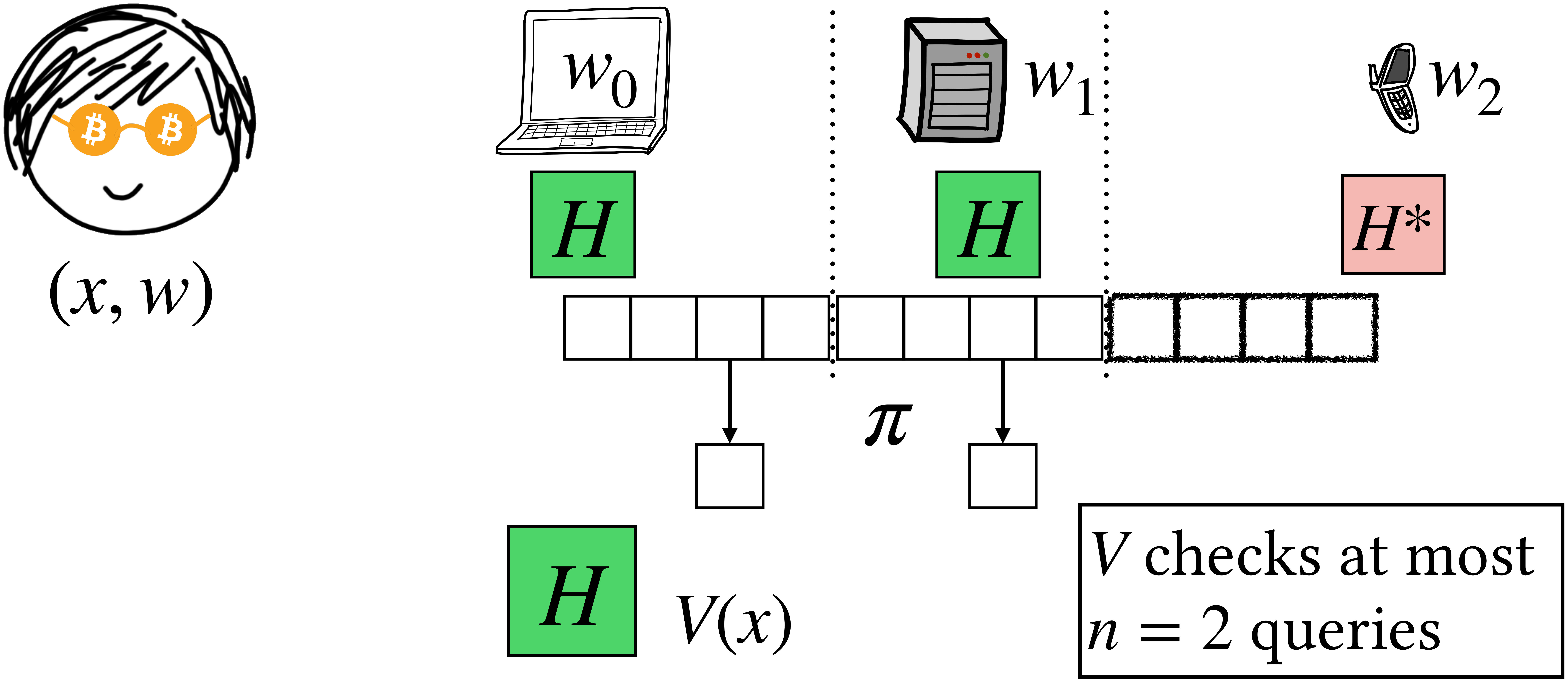
V checks at most $n = 2$ queries

Oracle Respecting Distribution

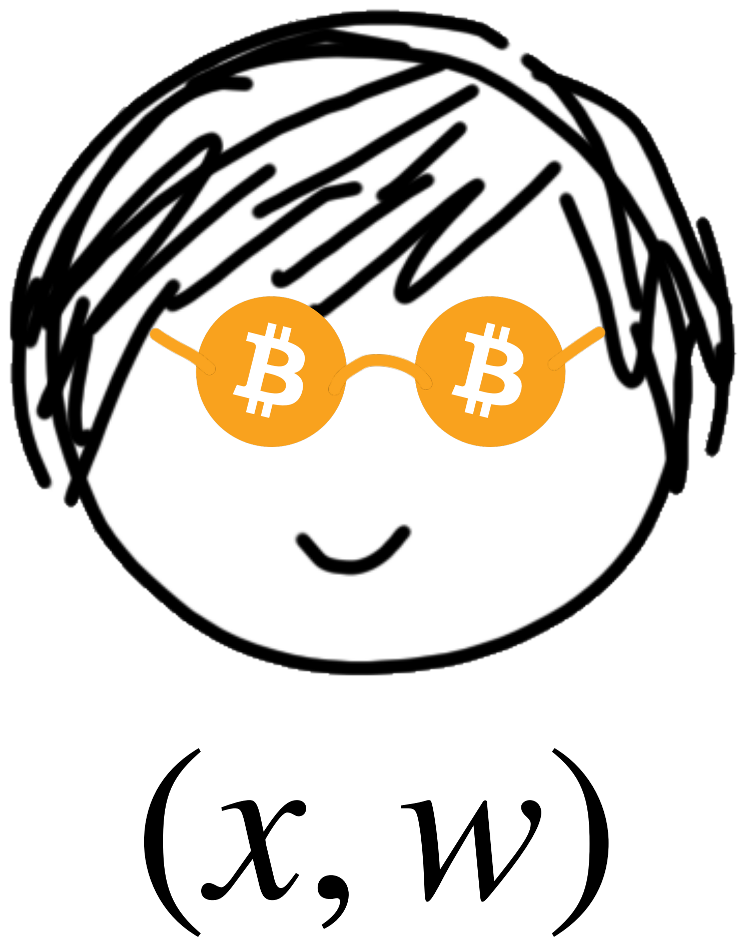


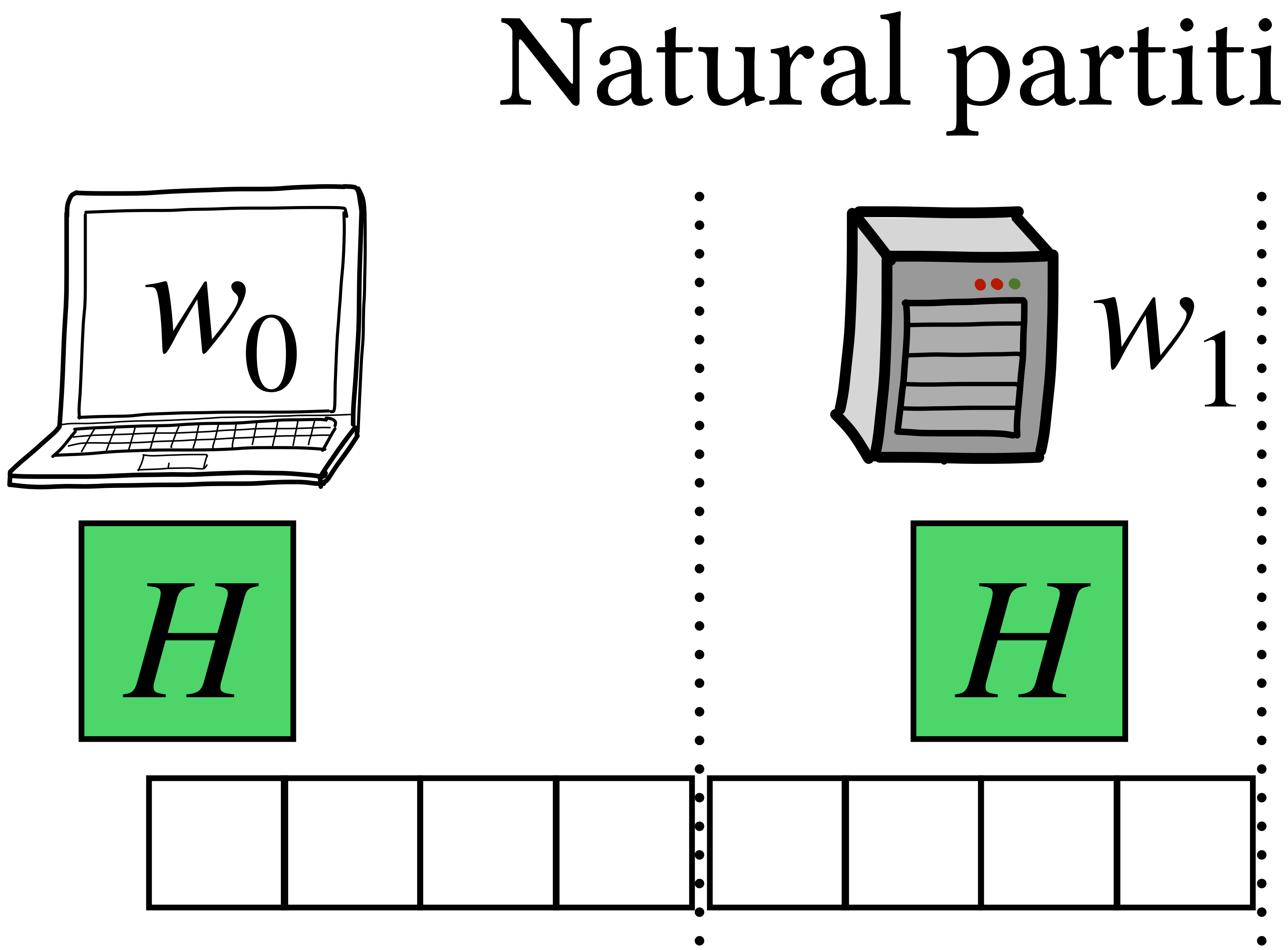
Oracle Respecting Distribution

Natural partitioning

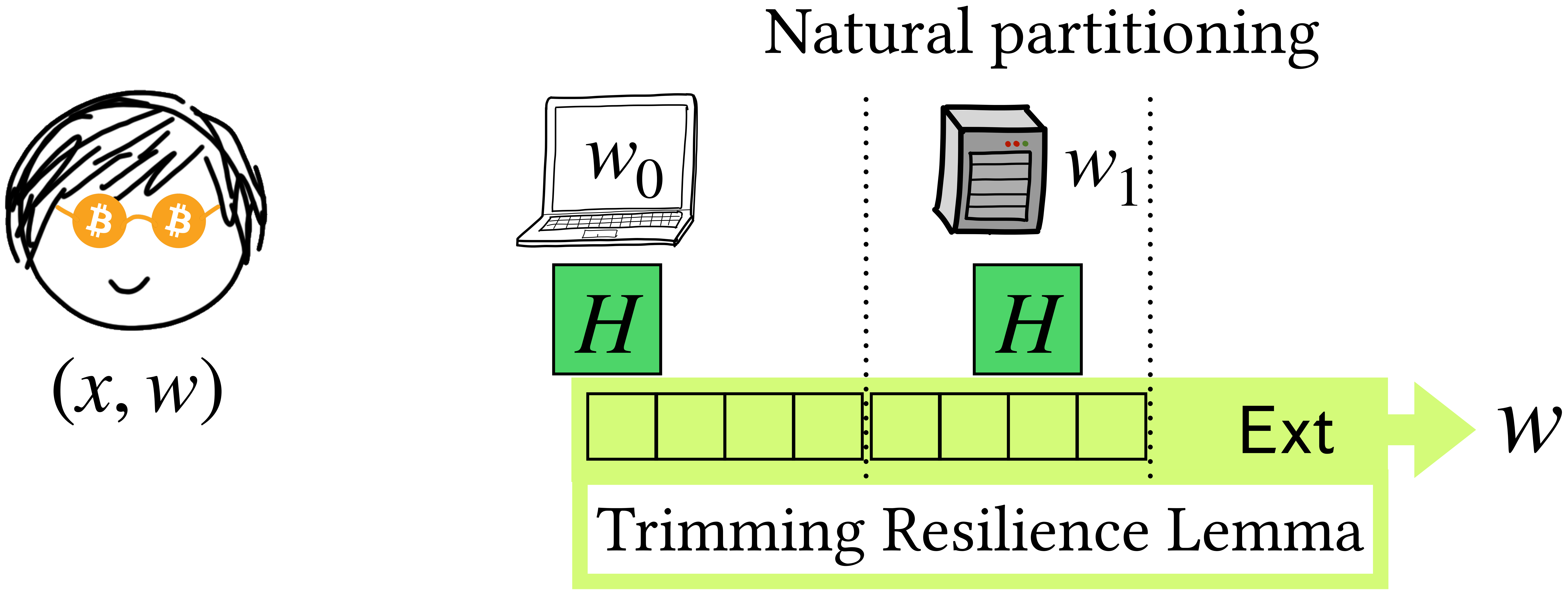


Oracle Respecting Distribution

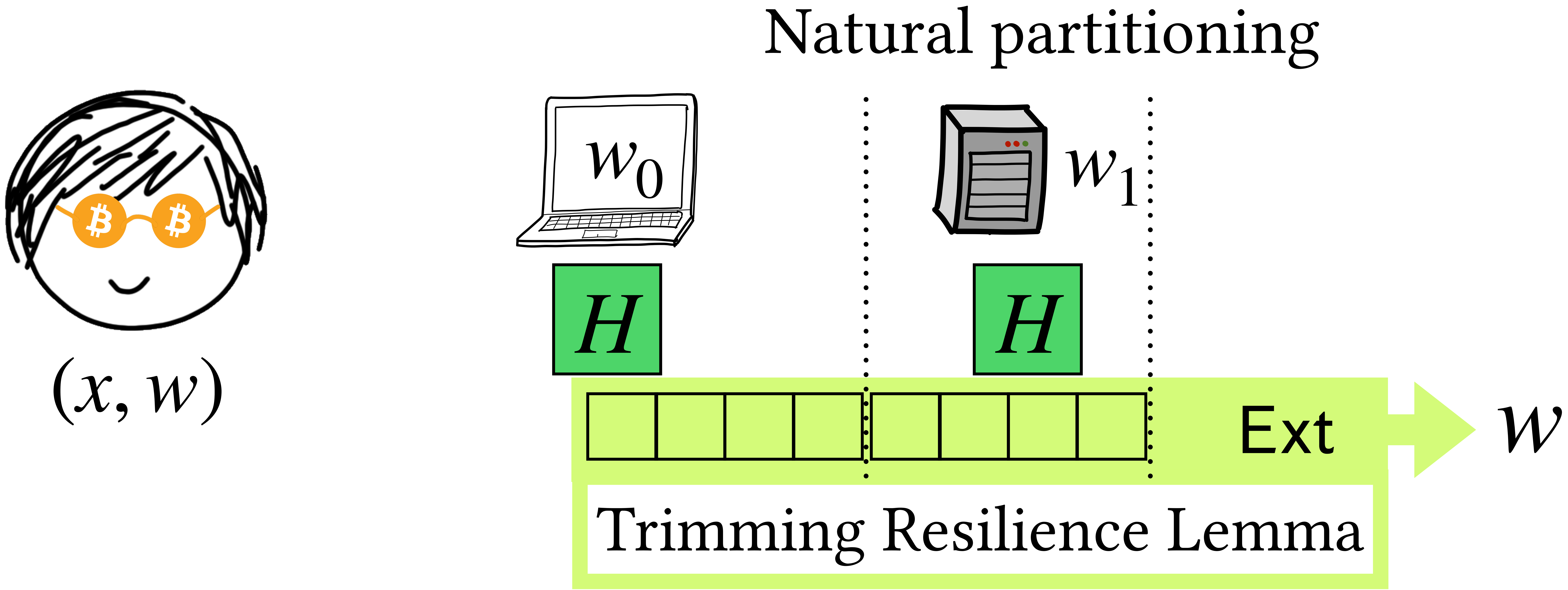




Oracle Respecting Distribution

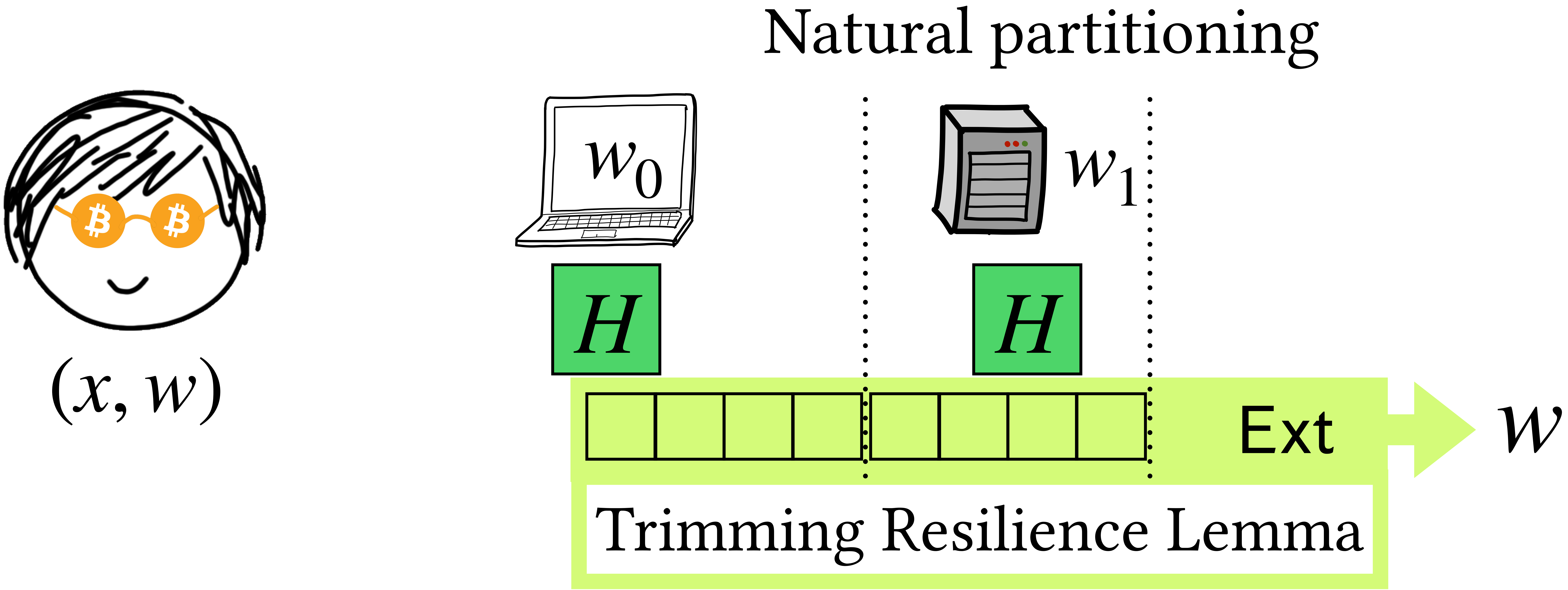


Oracle Respecting Distribution



Two views are sufficient to reconstruct the witness

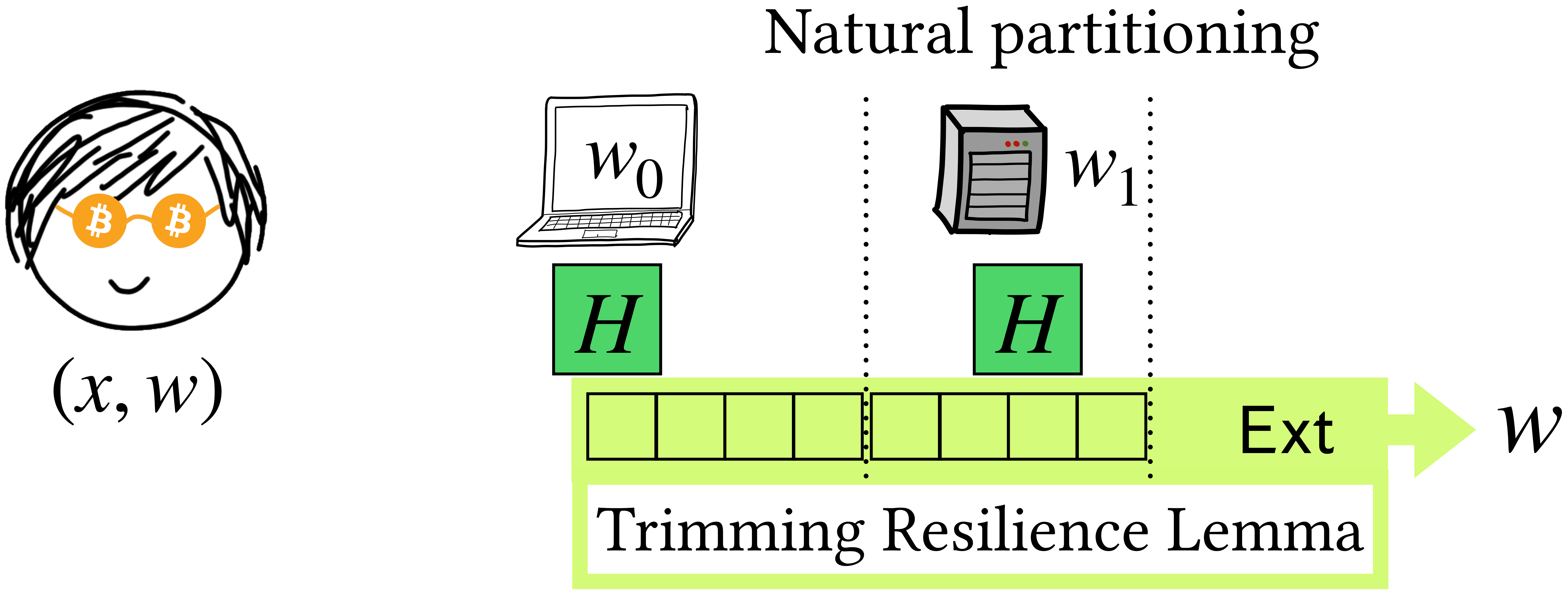
Oracle Respecting Distribution



Two views are sufficient to reconstruct the witness

3 party ORD protocol can not withstand 2 passive corruptions

Oracle Respecting Distribution



Two views are sufficient to reconstruct the witness

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

Caveats

- The n-party protocol must be mapped to a single party algorithm to apply the trimming lemma

- This mapping induces one of two artefacts:
  - Protocol property: Each RO query in the protocol must "traceable" to the party that first made it

    OR

  - NIZK property: $\mathsf{Ext}(\overrightarrow{Q}, \pi)$ does not actually need $H(\overrightarrow{Q})$

Fewer than $n - 1$ Corrupt?

- In the paper:
  - Extend impossibility for $n - O(1)$ corruptions
  - Notes on further barriers for many natural NIZKPoKs

- Impossibility itself does not generalize to $O(1)$ *fraction* of corruptions: $\exists$ NIZK that permits n-party ORD protocol with $\mathsf{const} \cdot n$ corruptions

Conclusion

- We showed that n-party protocols to securely compute certain hash-based signatures/NIZKs can not make blackbox use of the same hash function
  - Includes MPC-in-the-head, Fischlin/Unruh/Pass/Ks22 transform, PCPs/IOPs

- Dist. NIZK Verifier must depend on #parties—could it indicate that thresh. signature must grow with #signers?

Thanks!

eprint: 2023/1381

Thanks **Eysa Lee** for