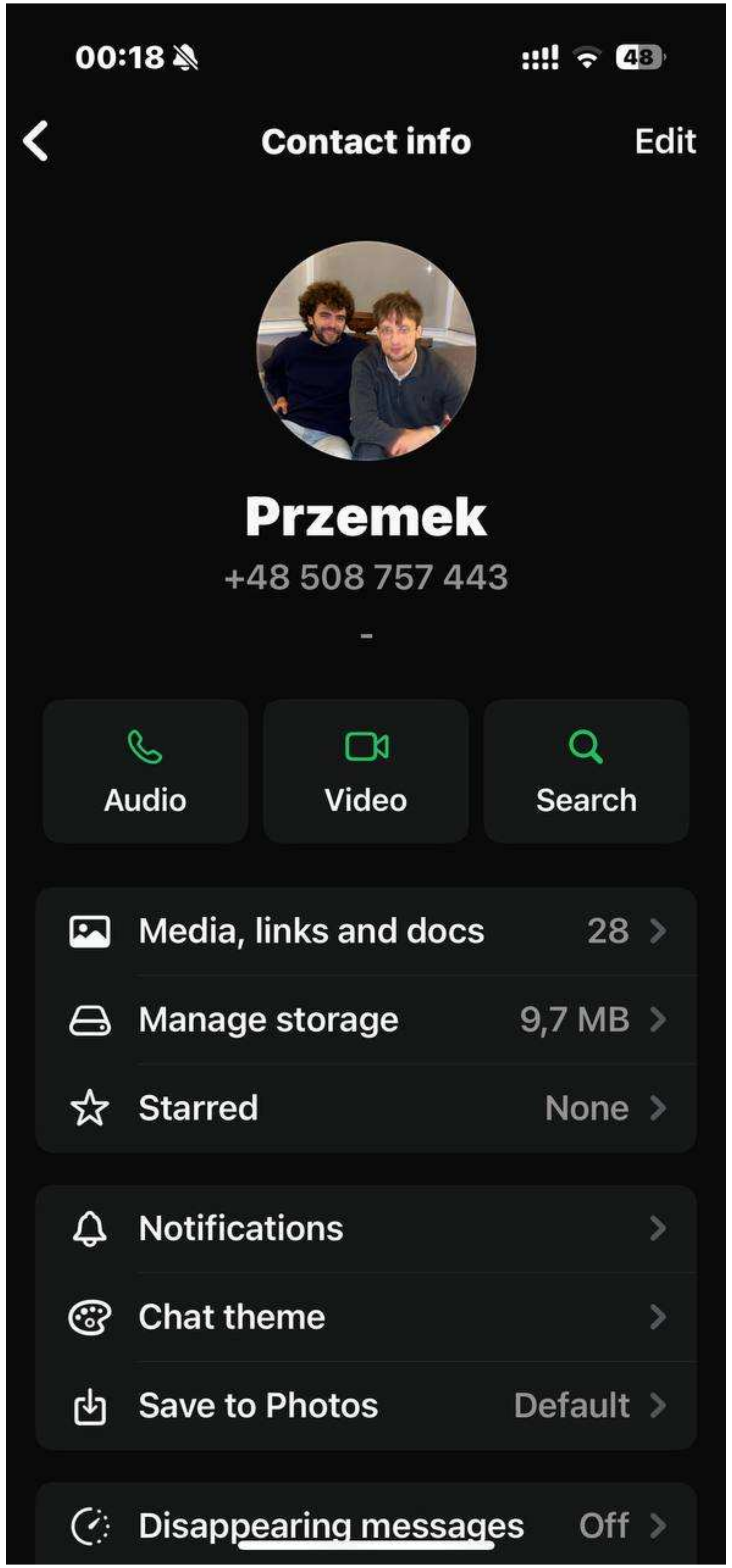
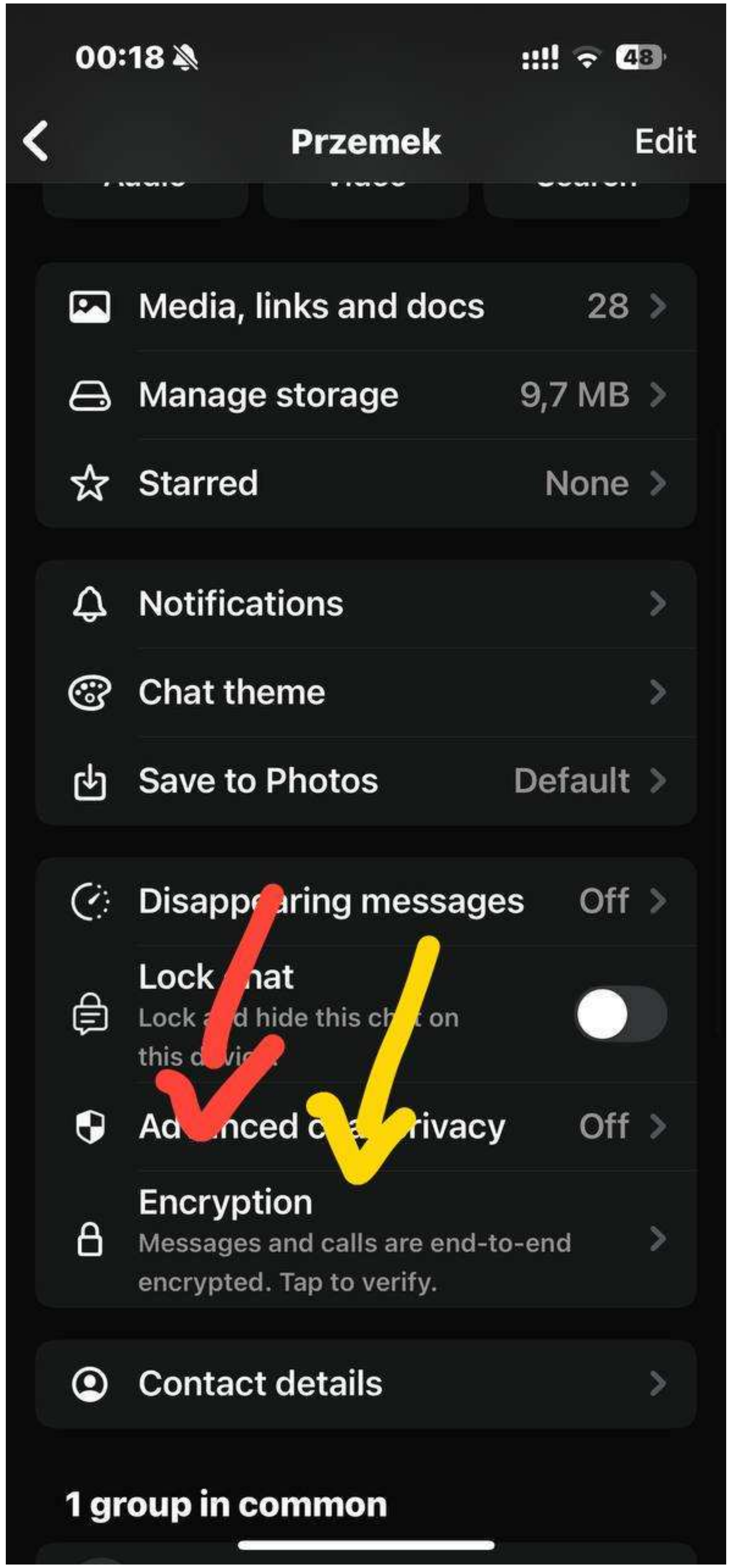
Self-Auditable Key Transparency

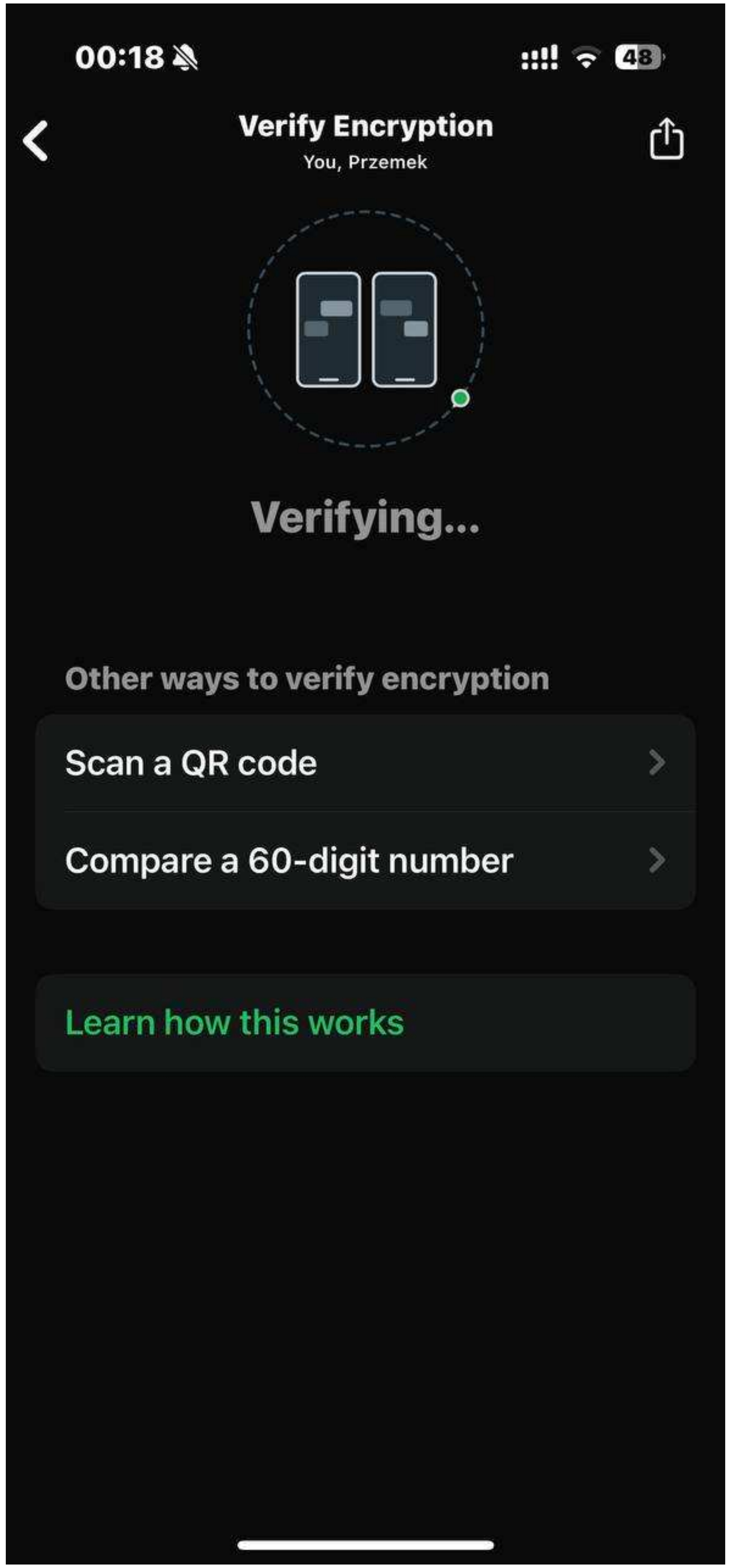
Hossein Hafezi

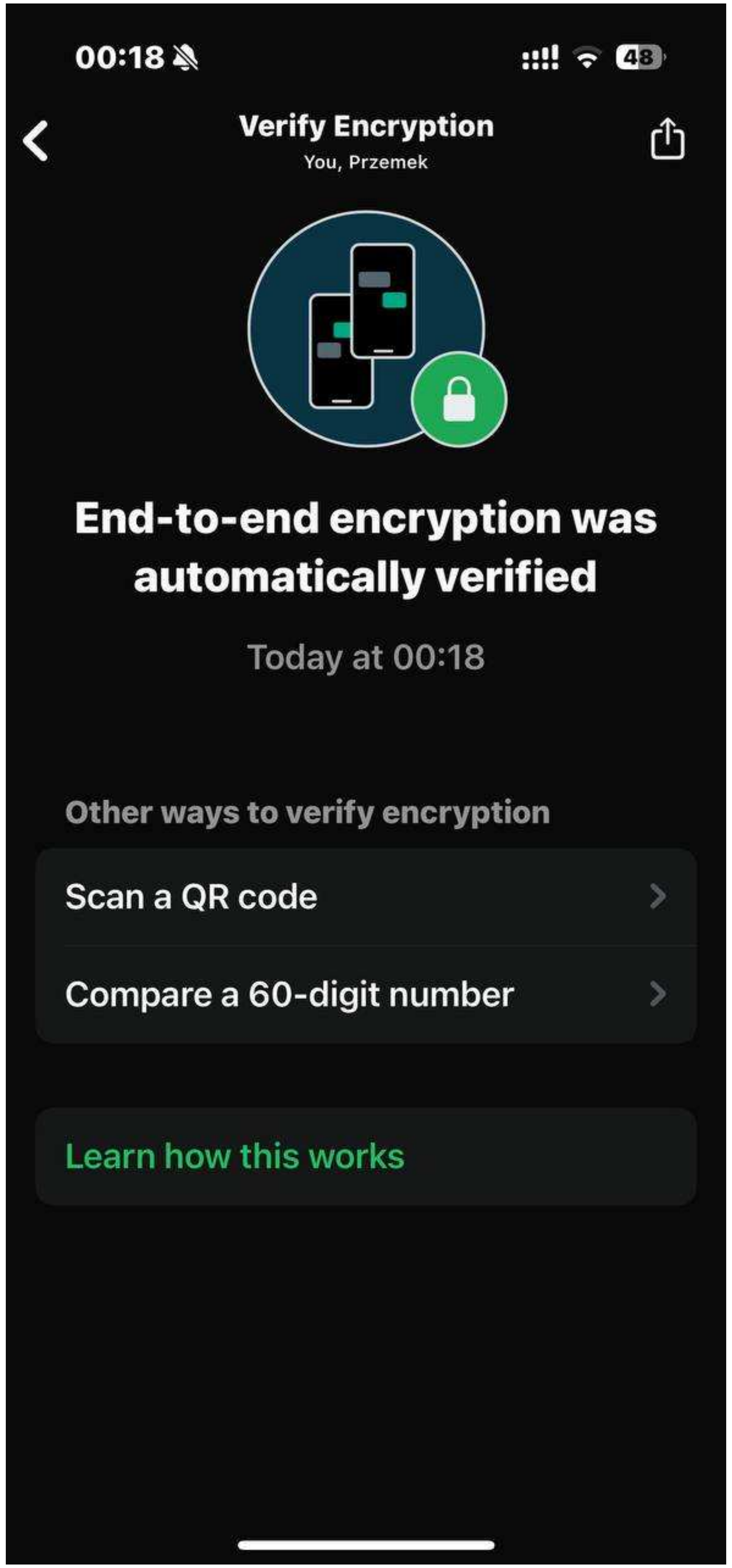
UNIVERSITY OF CAMBRIDGE

NYU


00:18
Contact info
Edit
Przemek
+48 508 757 443
-
Audio
Video
Search
Media, links and docs 28
Manage storage 9,7 MB
Starred None
Notifications
Chat theme
Save to Photos Default
Disappearing messages Off


00:18
Przemek
Edit
Media, links and docs 28
Manage storage 9,7 MB
Starred None
Notifications
Chat theme
Save to Photos Default
Disappearing messages Off
Lock chat
Lock and hide this chat on this device
Advanced chat privacy Off
Encryption
Messages and calls are end-to-end encrypted. Tap to verify.
Contact details
1 group in common


00:18
Verify Encryption
You, Przemek
Verifying...
Other ways to verify encryption
Scan a QR code
Compare a 60-digit number
Learn how this works


00:18
Verify Encryption
You, Przemek
End-to-end encryption was automatically verified
Today at 00:18
Other ways to verify encryption
Scan a QR code
Compare a 60-digit number
Learn how this works

Motivation Behind Key Transparency

End-to-End Encrypted Messaging

$(pk_{\text{Jon}}, pk_{\text{Ygritte}})$

$(pk_{\text{Jon}}, pk_{\text{Ygritte}})$

WhatsApp

End-to-End Encrypted Messaging

$(pk_{\text{Jon}}, pk_{\text{Ygritte}})$

$(pk_{\text{Jon}}, pk_{\text{Ygritte}})$

The service provider only transmits ciphertexts and cannot decrypt them

End-to-End Encrypted Messaging

$(pk_{\text{Jon}}, pk_{\text{Ygritte}})$

$(pk_{\text{Jon}}, pk_{\text{Ygritte}})$

WhatsApp

The service provider only transmits ciphertexts and cannot decrypt them

How do users get each other public keys for example from a phone number?

End-to-End Encrypted Messaging

$(pk_{\text{Jon}}, pk_{\text{NK}})$

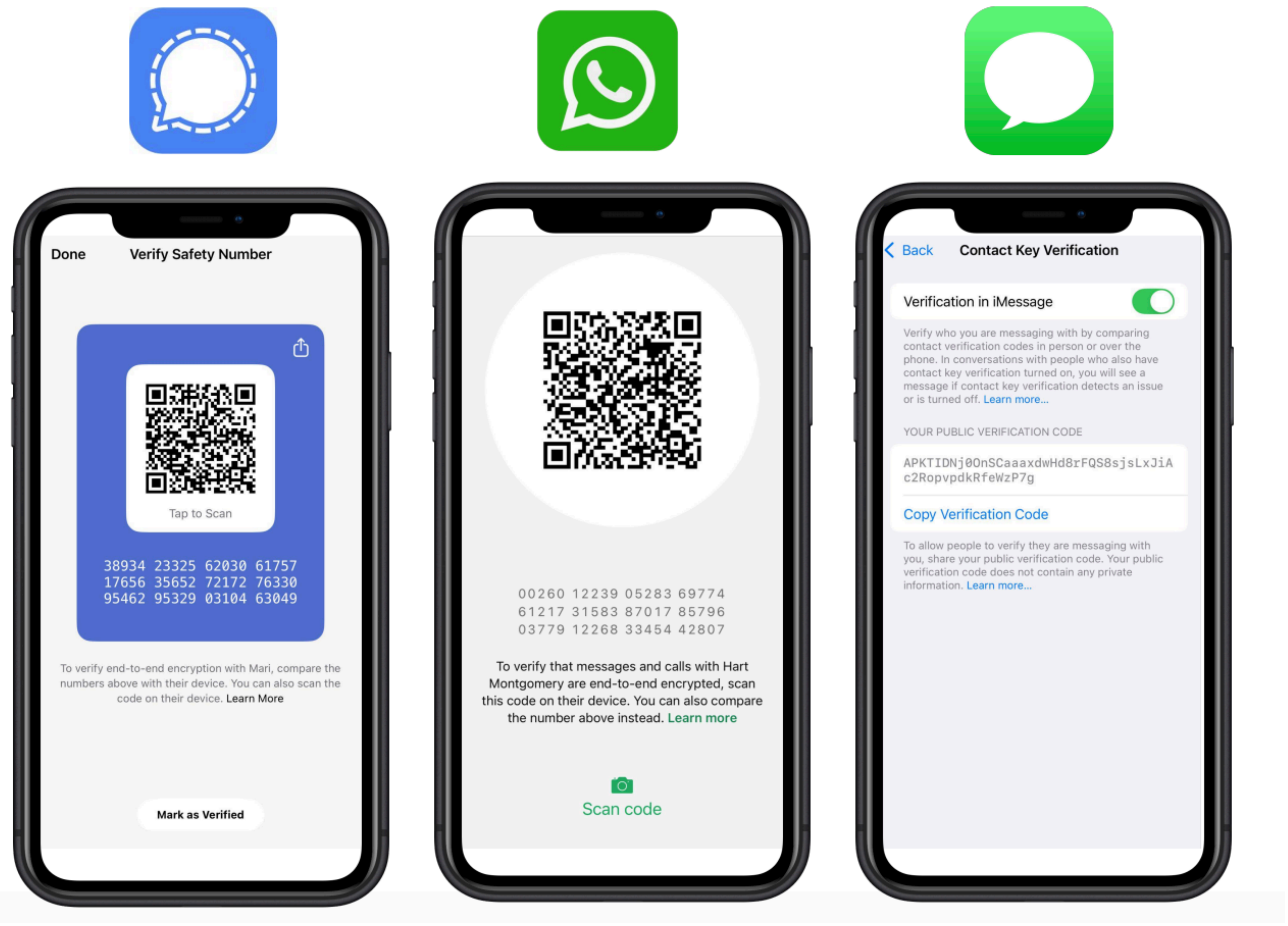
$(pk_{\text{Jon}}, pk_{\text{NK}})$

WhatsApp

Give Jon a malicious public key


"You know nothing, Jon Snow."

Traditional Solution: Out-of-Band Verification



Traditional Solution: Out-of-Band Verification

Key Transparency: CONIKS [Mel+15]

Key Transparency: CONIKS [Mel+15]

Periodically post

Signed commitments

to directory

WhatsApp

Key Transparency: CONIKS [Mel+15]

Periodically post

Signed commitments

to directory

pk_{Ygritte}, π

Key Transparency: CONIKS [Mel+15]

Verify pk is consistent with the commitment to the bulletin board

Periodically post

Signed commitments

to directory

pk_{Ygritte}, π

WhatsApp

Key Transparency: CONIKS [Mel+15]

Verify pk is consistent with the commitment to the bulletin board

Periodically post

Signed commitments

to directory

pk_{Ygritte}, π

pk_{Ygritte}, π

Key Transparency: CONIKS [Mel+15]

Verify pk is consistent with the commitment to the bulletin board

Verify my key is included correctly in the latest epoch

Periodically post

Signed commitments

to directory

pk_{Ygritte}, π

pk_{Ygritte}, π

Key Transparency: CONIKS [Mel+15]

Verify pk is consistent with the commitment to the bulletin board

Verify my key is included correctly in the latest epoch

Periodically post

Signed commitments

to directory

pk_{Ygritte}, π

pk_{Ygritte}, π

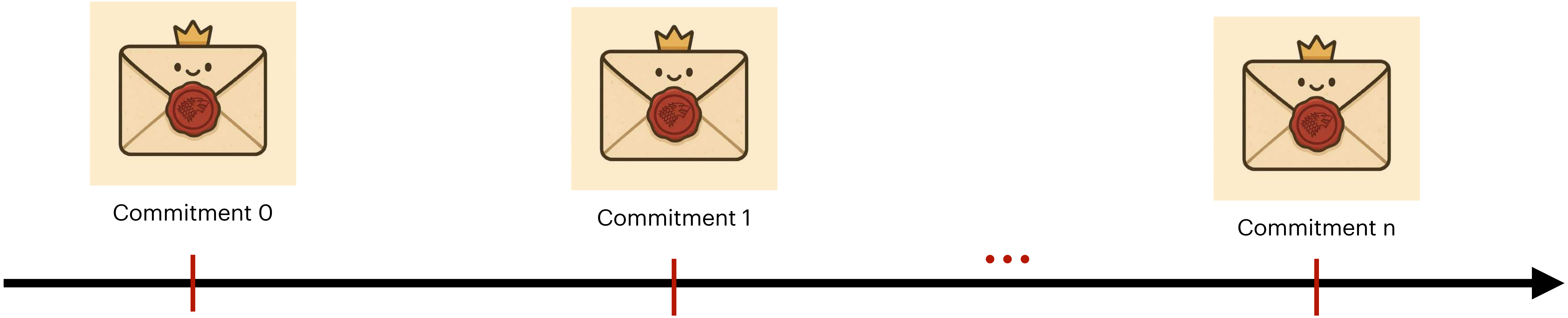
WhatsApp

AUTOMATED SOLUTION: no manual effort for Jon and Ygritte

Key Transparency Model

Key Transparency Scheme

- Key transparency scheme consists of a server that at every epoch publishes the new commitment to the directory along with auxiliary data (invariance proof)



Key Transparency Scheme

Jon

(I) What is Ygritte's latest public key?

Lookup

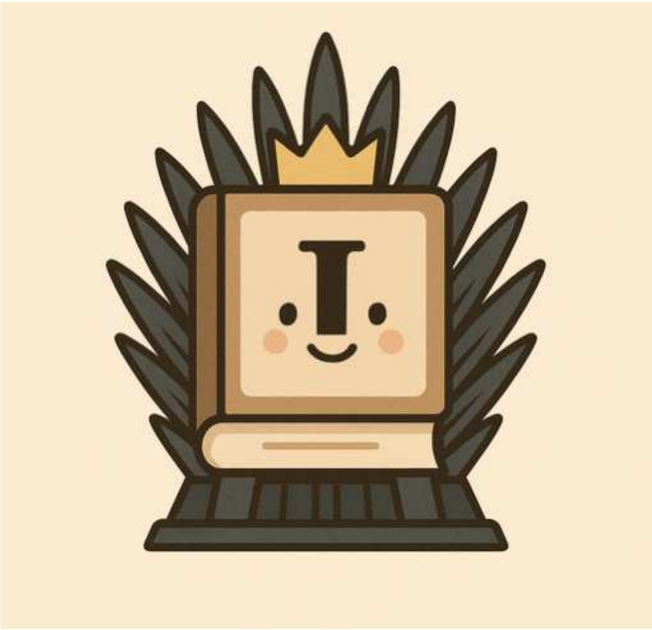
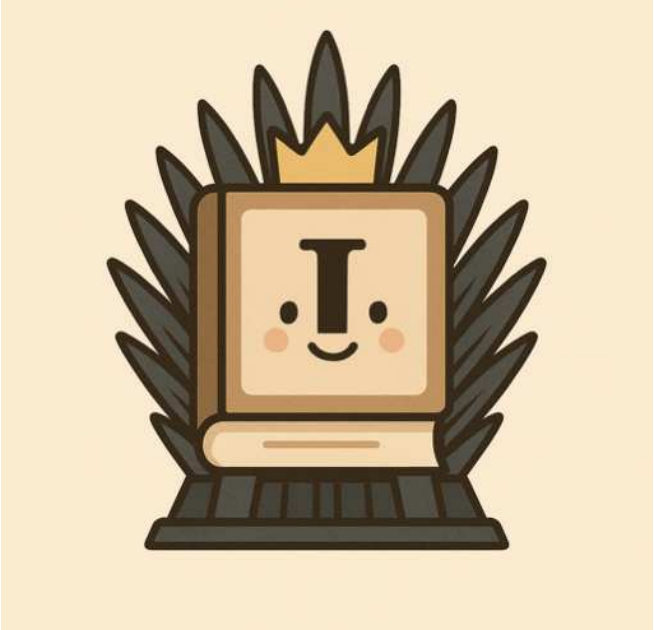
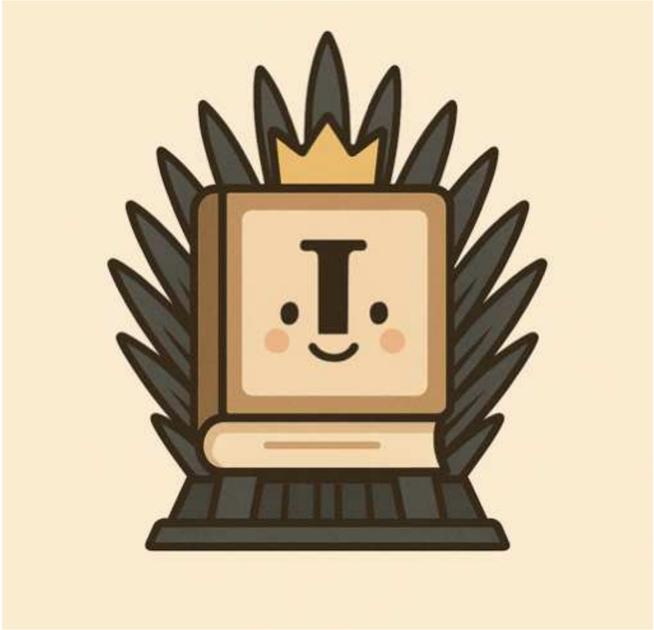
(II) My key hasn't changed since last year?

Proof of consistency

WhatsApp

Necessity of Proof of Consistency

Assume Jon is offline from state i+1 to state j-1


→

→ … →


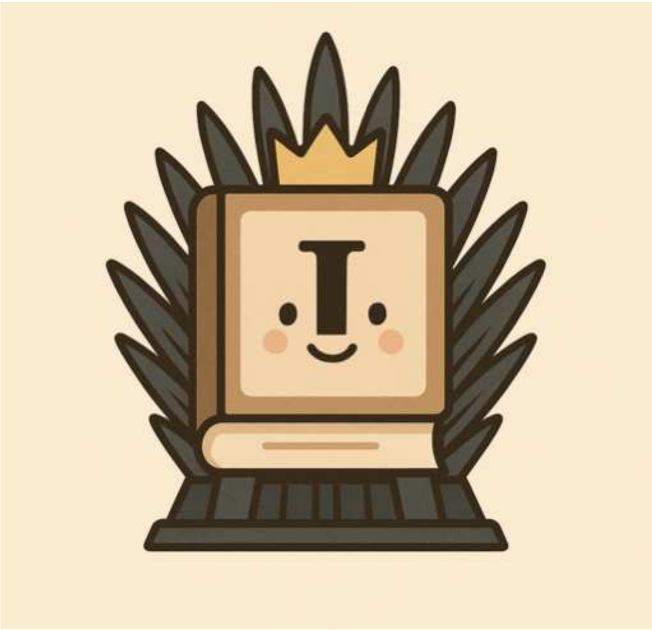
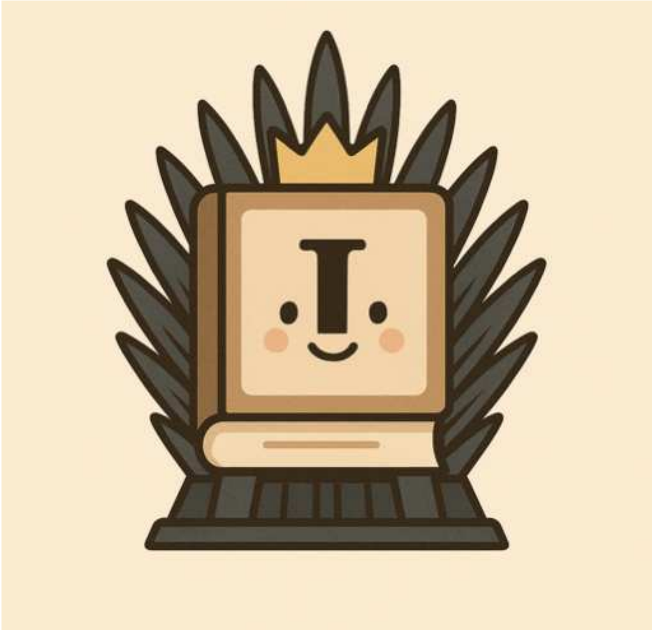
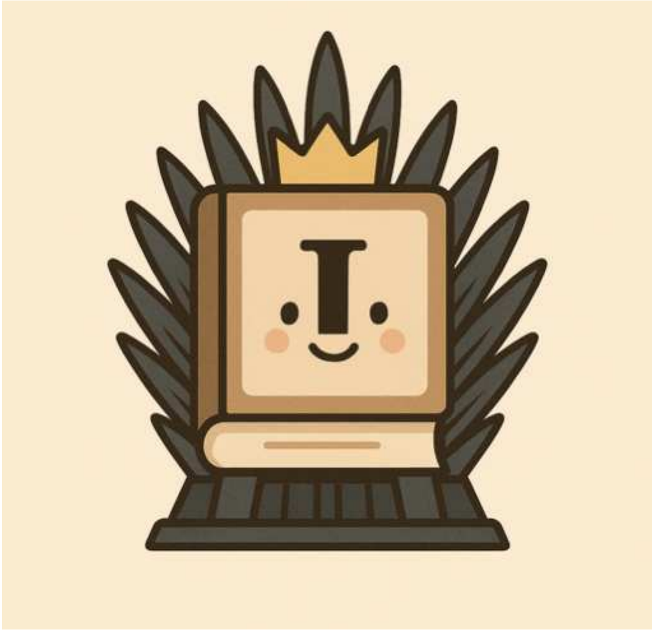
State i

State i+1

State j

Necessity of Proof of Consistency

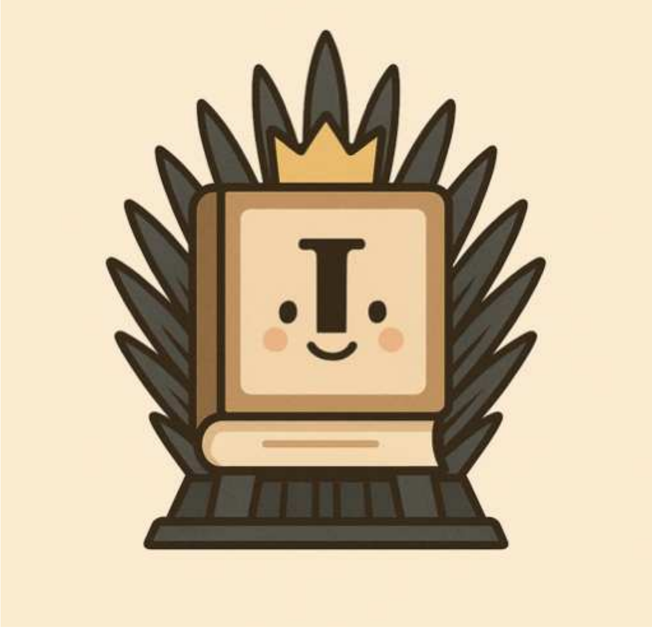
Assume Jon is offline from state i+1 to state j-1

 →  → … → 

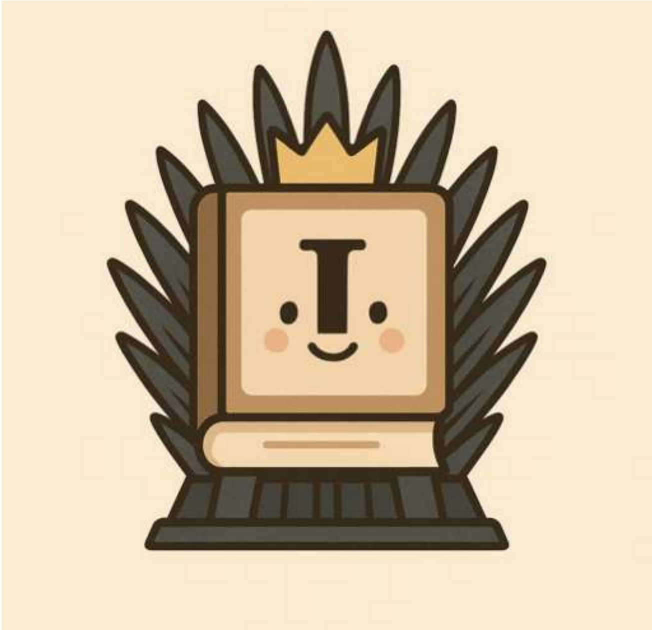
State i | State i+1 | State j

Necessity of Proof of Consistency

Assume Jon is offline from state i+1 to state j-1

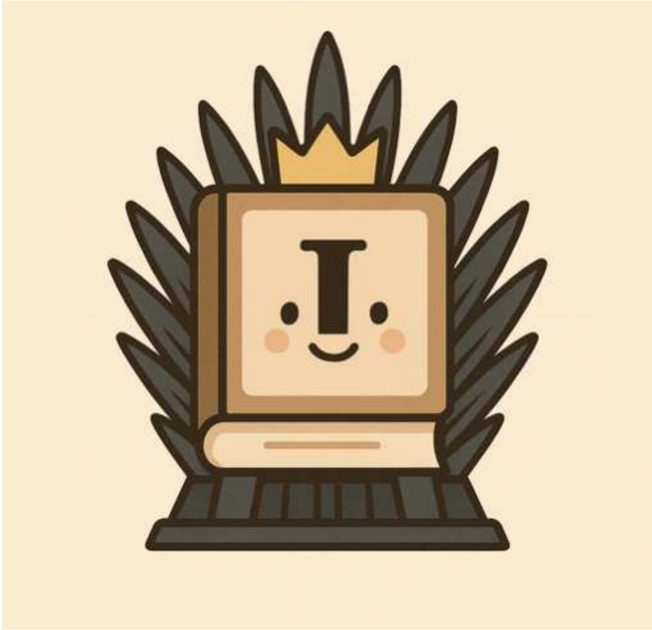


State i



State i+1

...



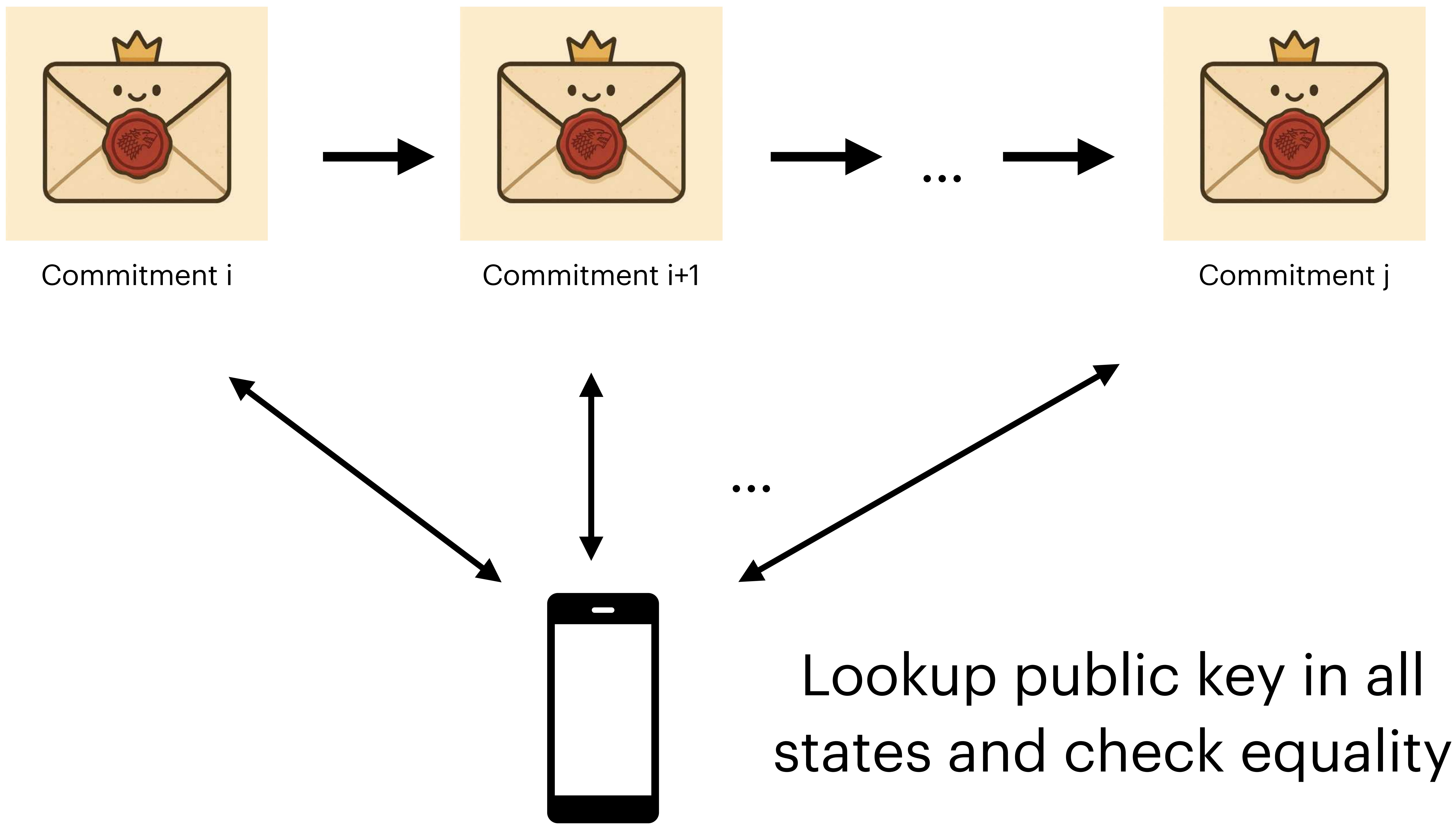
State j

Request a proof of consistency from epoch i to epoch j

Different Solutions to Proof of Consistency

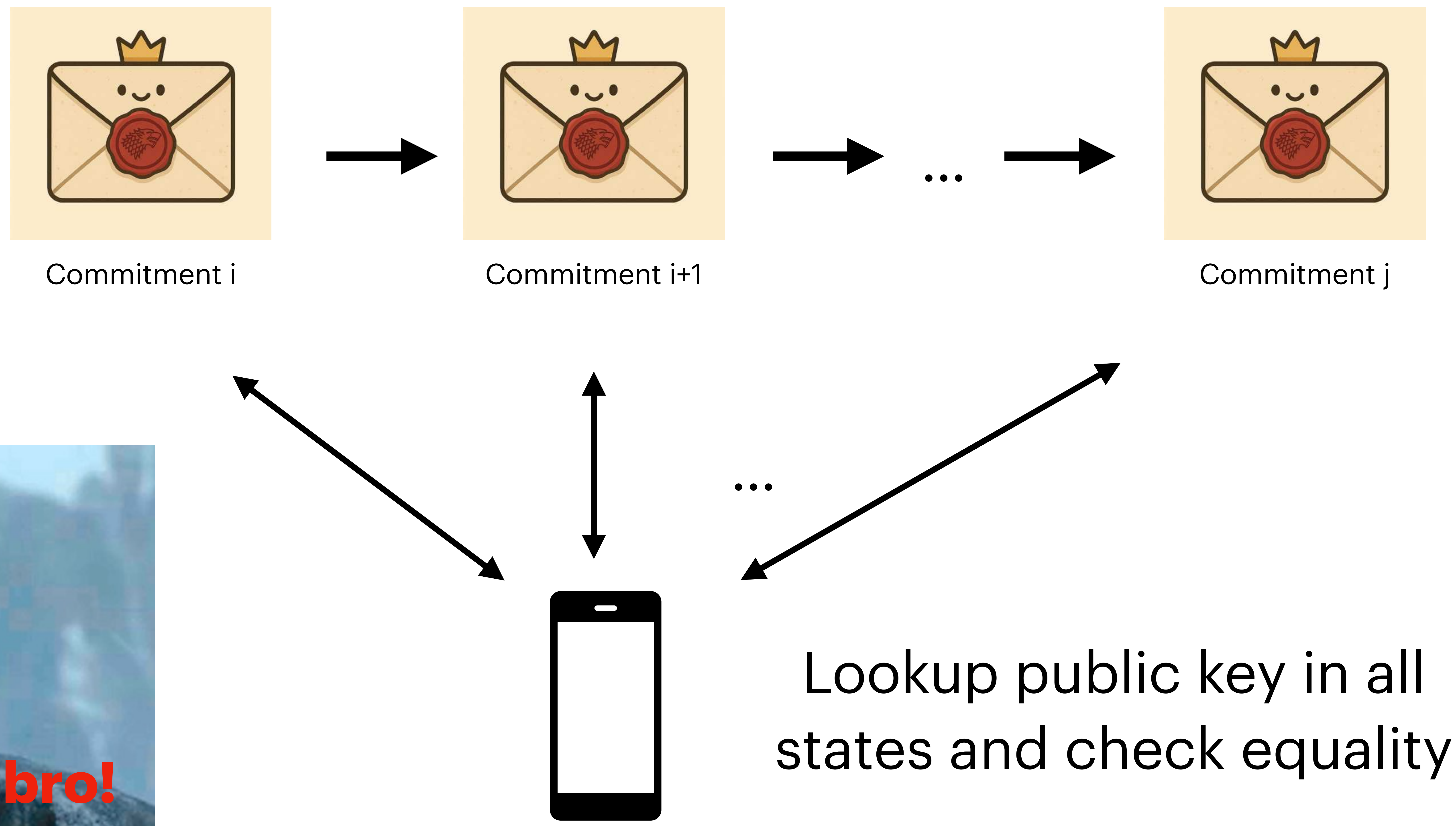
- Naively performing lookup for every epoch, e.g. CONIKS [Mel+15]
- Version Invariant, e.g. SEEMless [Cha+19], Parakeet [Mal+23], Optiks [Len+24]
- Random Version Invariant, e.g. IronDict [Haf+26] and Aegon [Haf+26]

Proof of Consistency: Naive Solution [Mel+15]



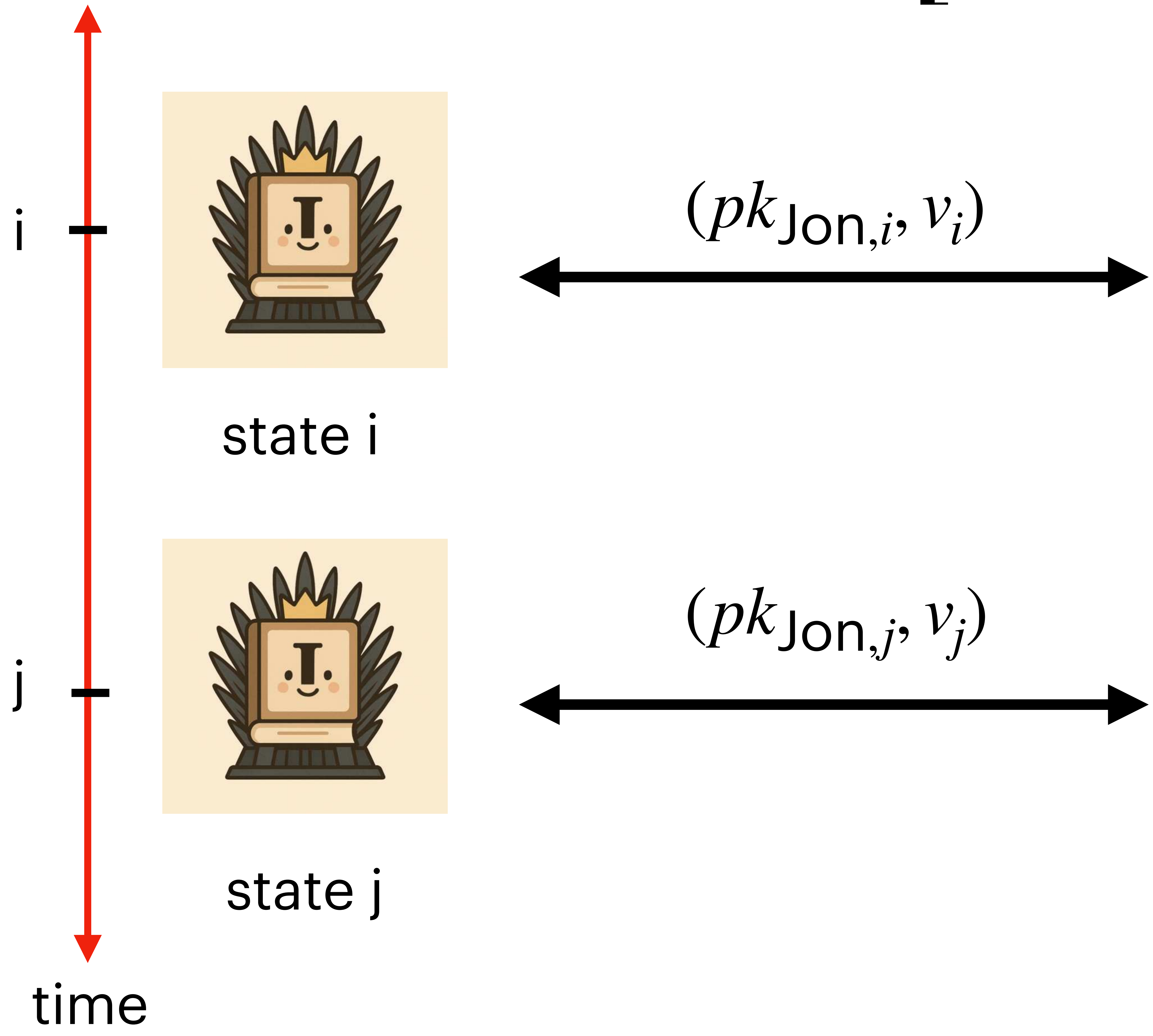
Lookup public key in all states and check equality

Proof of Consistency: Naive Solution [Mel+15]



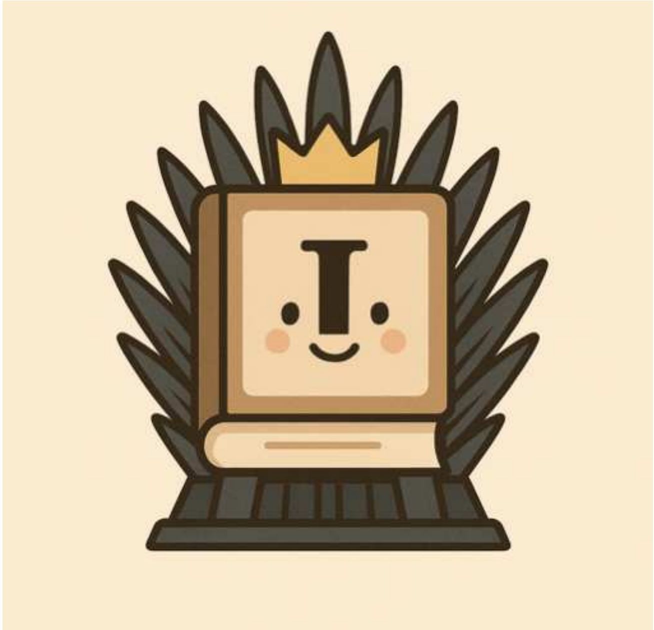
Lookup public key in all states and check equality

Version Invariant [Cha+19, Mal+23, Len+24]



Version Invariant [Cha+19, Mal+23, Len+24]

i



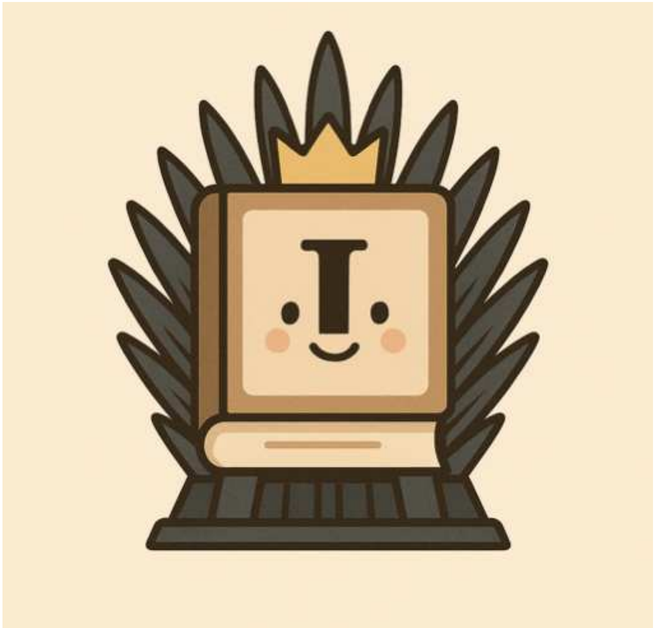
state i

$(pk_{\mathsf{Jon},i}, v_i)$

proof of consistency from epoch i to j

$\iff v_i = v_j$

j



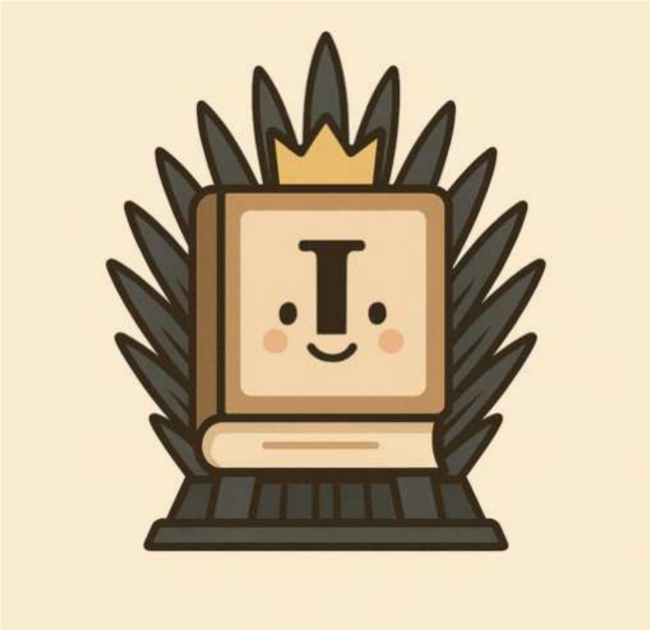
state j

$(pk_{\mathsf{Jon},j}, v_j)$

time

Version Invariant [Cha+19, Mal+23, Len+24]

i



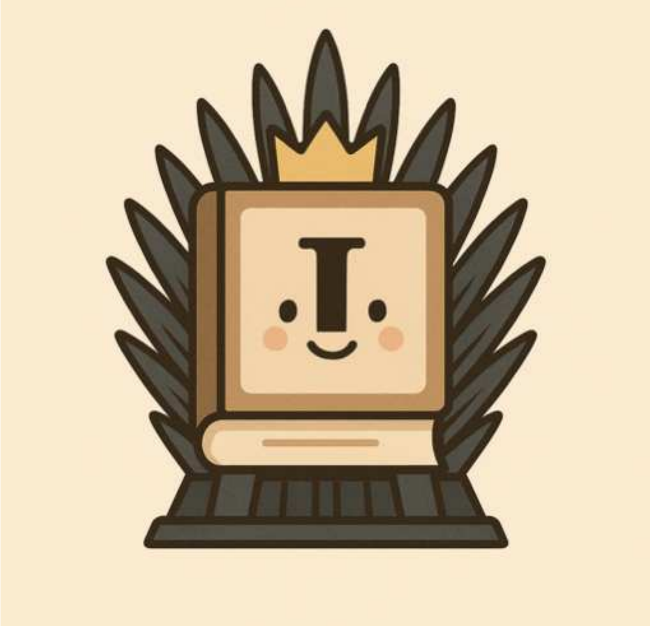
state i

$(pk_{\text{Jon},i}, v_i)$

proof of consistency from epoch i to j

$\iff v_i = v_j$

j



state j

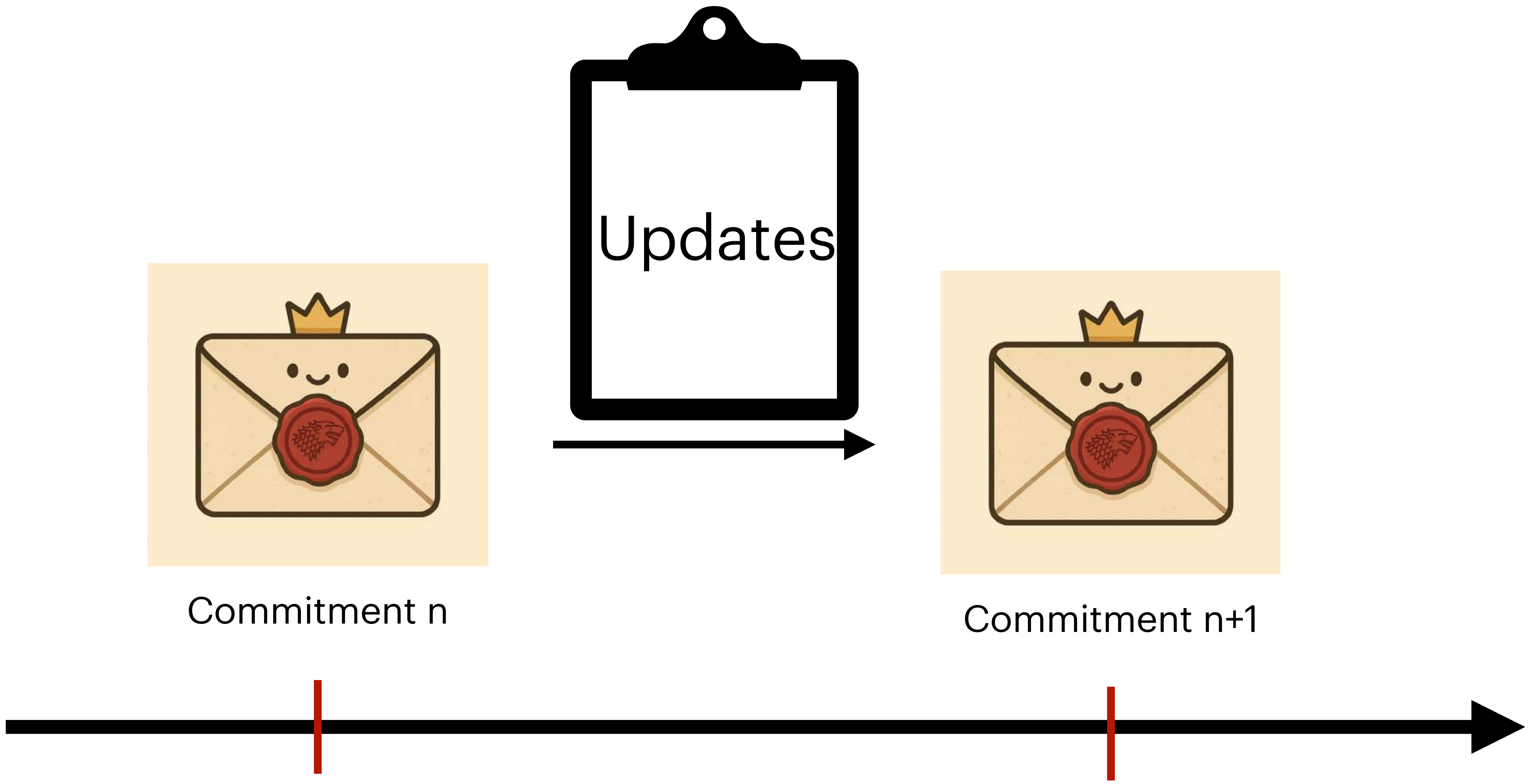
$(pk_{\text{Jon},j}, v_j)$

How to audit the version invariant?

time

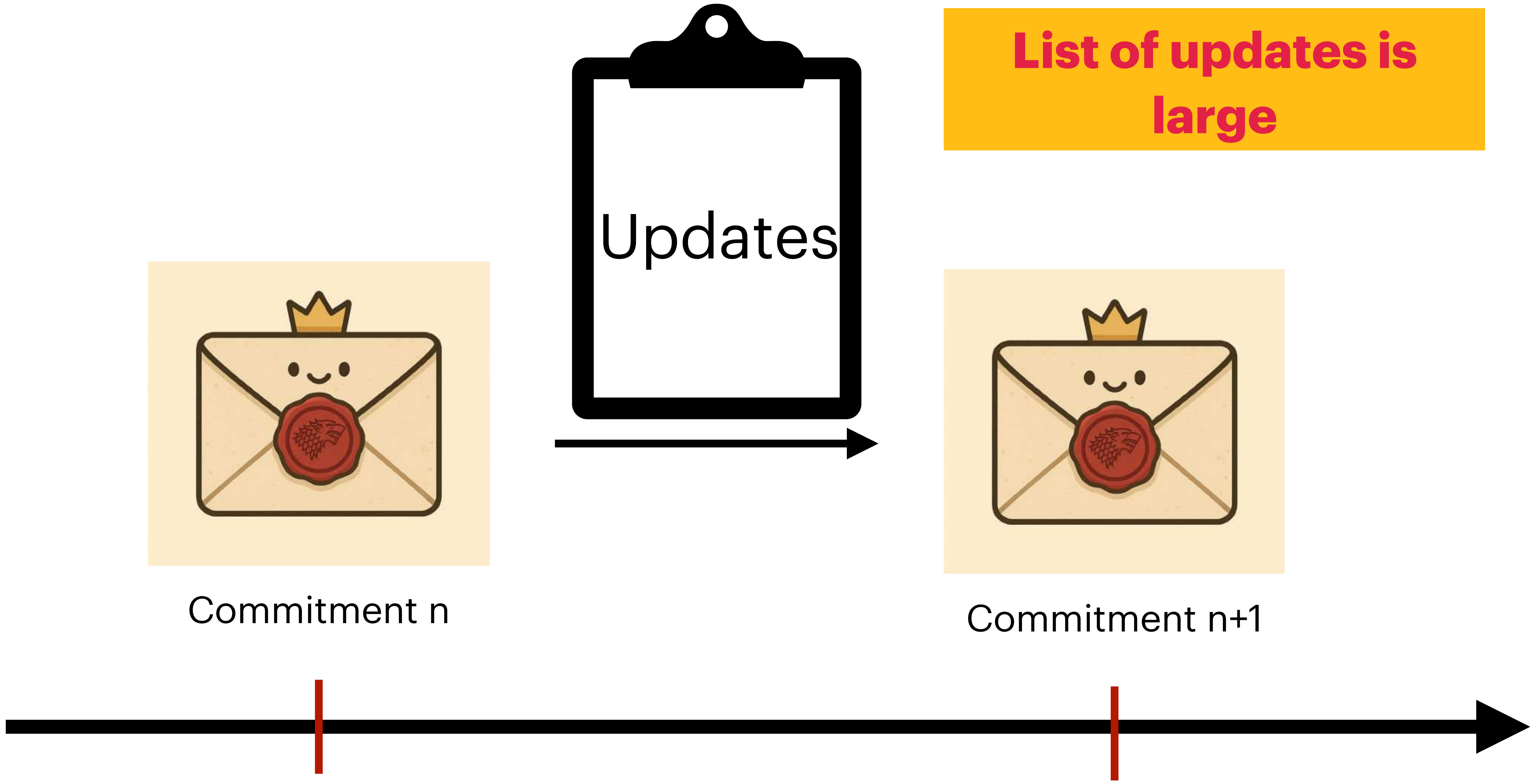
Auditing Version Invariant [Cha+19, Mal+23, Len+24]

- Server publishes invariance proof == the list of updates.
- ***Auditors*** verify the updates have been appended correctly.



Auditing Version Invariant [Cha+19, Mal+23, Len+24]

- Server publishes invariance proof == the list of updates.
- ***Auditors*** verify the updates have been appended correctly.



Drawbacks of Existing Deployed Schemes

Main Challenges of Existing Deployed Schemes?
(SEEMless [Cha+19], Parakeet [Mal+23], Optiks [Len+24])

- Self-Auditability
- Perfect Privacy
- History Pruning

Challenge I: Self-Auditability

- Proof of invariance is large ==> linear in the number of updates in the epoch.

Challenge I: Self-Auditability

- Proof of invariance is large ==> linear in the number of updates in the epoch.
- = hashing 80MB per minute for WhatsApp KT.

Challenge I: Self-Auditability

- Proof of invariance is large ==> linear in the number of updates in the epoch.
- = hashing 80MB per minutes for WhatsApp KT.
- Auditing delegated to Cloudflare

Challenge I: Self-Auditability

- Proof of invariance is large ==> linear in the number of updates in the epoch.
- = hashing 80MB per minutes for WhatsApp KT.
- Auditing delegated to Cloudflare

Attempts to make the proof succinct using SNARKs (e.g., Verdict [Tzi+22] and Hekaton [Ros+24]) ==> high hash-function costs in circuits limit scalability.

Challenge II: Perfect Privacy

- Privacy is crucial in the context of key transparency:
    - Lookups shouldn’t reveal pattern of changing keys of users.
    - Otherwise an adversary observing a user doesn’t change their key regularly, makes it a potential target for them.

Challenge II: Perfect Privacy

- Privacy is crucial in the context of key transparency:
    - Lookups shouldn't reveal pattern of changing keys of users.
    - Otherwise an adversary observing a user doesn't change their key regularly, makes it a potential target for them.
- Tree-based solutions do not offer perfect privacy.
    - SEEMless reveals the last time a key was updated during a lookup.

Challenge III: History Pruning

- The tree-based models (except for CONIKS) are based on append-only trees.

Challenge III: History Pruning

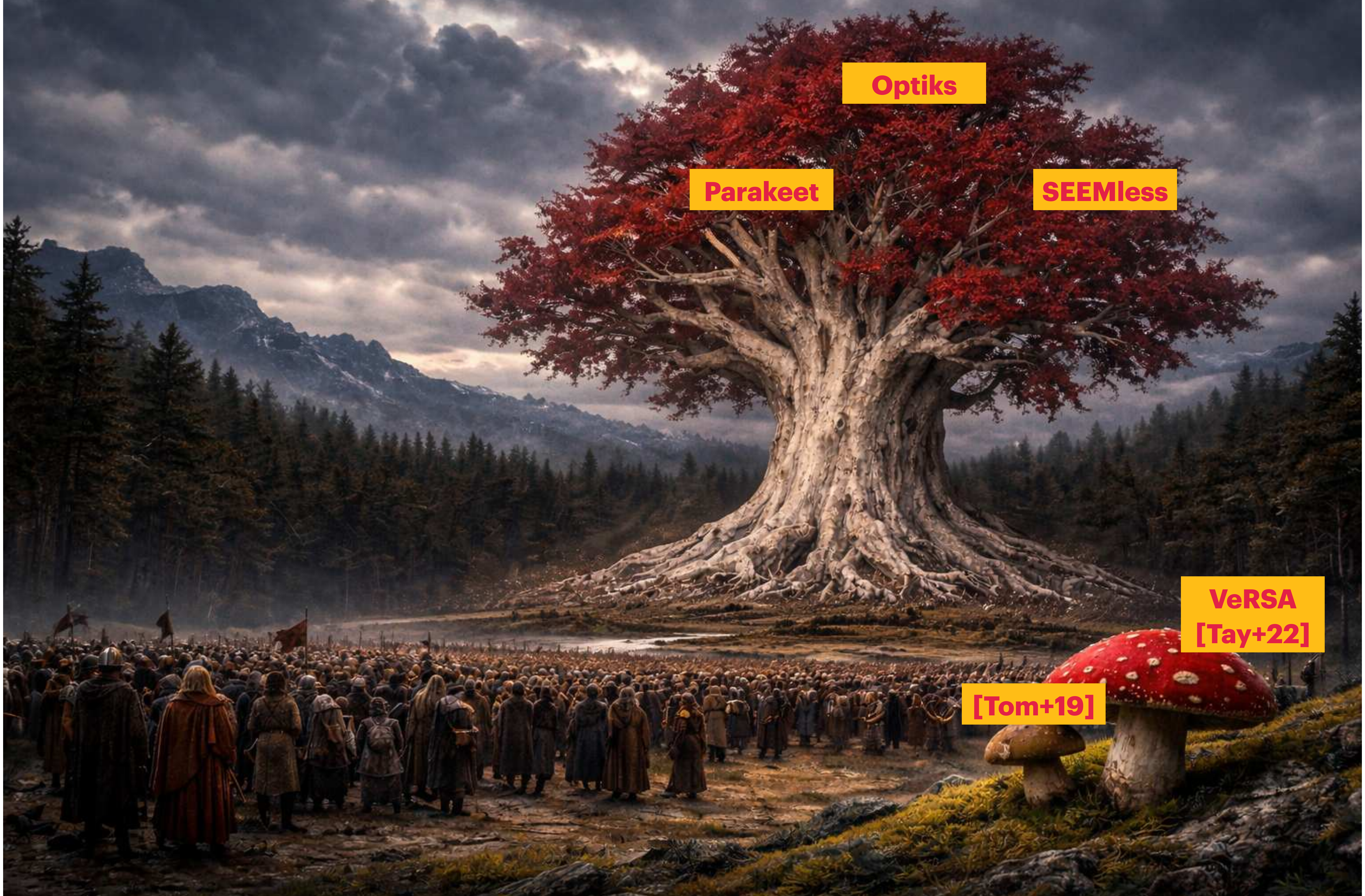
- The tree-based models (except for CONIKS) are based on append-only trees.
- All unnecessary historical data are kept in the tree and its size grows indefinitely.

Challenge III: History Pruning

- The tree-based models (except for CONIKS) are based on append-only trees.
- All unnecessary historical data are kept in the tree and its size grows indefinitely.
- WhatsApp has a user base of 3-4 billion, but its tree size is more than 100 billion.

Our Proposed Schemes: IronDict and Aegon


Optiks
Parakeet
SEEMless


Optiks
Parakeet
SEEMless
VeRSA
[Tay+22]
[Tom+19]


Optiks
Parakeet
SEEMless
VeRSA
[Tay+22]
[Tom+19]


IronDict / Aegon
based on polynomials
Optiks
Parakeet
SEEMless
VeRSA
[Tay+22]
[Tom+19]

IronDict: Transparent Dictionaries from Polynomial Commitments

Hossein Hafezi†[1], Alireza Shirzad†[2], Benedikt Bünz[3], and Joseph Bonneau[4]

[1,3,4]New York University
[2]University of Pennsylvania

USENIX Security 2026

IronDict: Transparent Dictionaries from Polynomial Commitments

Hossein Hafezi[†1], Alireza Shirzad[†2], Benedikt Bünz[3], and Joseph Bonneau[4]

[1,3,4]New York University
[2]University of Pennsylvania

USENIX Security 2026

Aegon: Self-Auditable Key Transparency

Hossein Hafezi*
University of Cambridge

Alireza Shirzad*
University of Pennsylvania

Benedikt Bünz
NYU

Kevin Lewi
Meta

Dillon George
Meta

Joseph Bonneau
NYU

IronDict

- **Self-auditability:** constant proof size, 8 kB for dictionary of size 1 billion.

IronDict

- **Self-auditability:** constant proof size, 8 kB for dictionary of size 1 billion.
    - **Fast-forwarding:** Check a single audit proof and verify the whole history
    - Self-auditability without fast-forwarding is pointless.

IronDict

- **Self-auditability:** constant proof size, 8 kB for dictionary of size 1 billion.
    - **Fast-forwarding:** Check a single audit proof and verify the whole history
    - Self-auditablity without fast-forwarding is pointless.
- **Perfect privacy**


AEGON
IRONDICT
Requirement of a Linear-Sized Setup
High Epoch Latency
Not Horizontally Scalable
Storage Inefficiency

Random Version Invariant of IronDict and Aegon

$$\mathrm{Dict}_i = (pk_1^{(i)}, pk_2^{(i)}, \ldots, pk_N^{(i)}),$$

Random Version Invariant of IronDict and Aegon

$$\mathrm{Dict}_i = (pk_1^{(i)}, pk_2^{(i)}, \ldots, pk_N^{(i)}), \quad \Delta_i = \mathrm{Dict}_{i+1} - \mathrm{Dict}_i$$

Random Version Invariant of IronDict and Aegon

$$\text{Dict}_i = (pk_1^{(i)}, pk_2^{(i)}, \ldots, pk_N^{(i)}), \quad \Delta_i = \text{Dict}_{i+1} - \text{Dict}_i$$

Proof of consistency for index t: $\text{Dict}_i[t] = \text{Dict}_i[t] = \ldots = \text{Dict}_j[t]$

Random Version Invariant of IronDict and Aegon

$$\mathrm{Dict}_i = (pk_1^{(i)}, pk_2^{(i)}, \ldots, pk_N^{(i)}), \quad \Delta_i = \mathrm{Dict}_{i+1} - \mathrm{Dict}_i$$

Proof of consistency for index t: $\mathrm{Dict}_i[t] = \mathrm{Dict}_i[t] = \ldots = \mathrm{Dict}_j[t]$

$$\iff \Delta_{i+1}[t] = \Delta_{i+2}[t] = \ldots = \Delta_j[t] = 0$$

Random Version Invariant of IronDict and Aegon

$$\mathrm{Dict}_i = (pk_1^{(i)}, pk_2^{(i)}, \ldots, pk_N^{(i)}), \quad \Delta_i = \mathrm{Dict}_{i+1} - \mathrm{Dict}_i$$

Proof of consistency for index t: $\mathrm{Dict}_i[t] = \mathrm{Dict}_i[t] = \ldots = \mathrm{Dict}_j[t]$

$$\iff \Delta_{i+1}[t] = \Delta_{i+2}[t] = \ldots = \Delta_j[t] = 0$$

Ghost key attack : $\sum_{i<k\leq j} \Delta_k[t] = 0$ and $\exists\, i < i_0 \leq j : \Delta_{i_0}[t] \neq 0$

Random Version Invariant of IronDict and Aegon

$$\mathrm{Dict}_i = (pk_1^{(i)}, pk_2^{(i)}, \ldots, pk_N^{(i)}), \quad \Delta_i = \mathrm{Dict}_{i+1} - \mathrm{Dict}_i$$

Proof of consistency for index t: $\mathrm{Dict}_i[t] = \mathrm{Dict}_i[t] = \ldots = \mathrm{Dict}_j[t]$

$$\iff \Delta_{i+1}[t] = \Delta_{i+2}[t] = \ldots = \Delta_j[t] = 0$$

Ghost key attack : $\sum_{i<k\leq j} \Delta_k[t] = 0$ and $\exists\, i < i_0 \leq j : \Delta_{i_0}[t] \neq 0$

Use randomness so the differences do not cancel each other out

Random Version Invariant of IronDict and Aegon

$$\text{Rand vector} : \text{Rand}_n = \text{Dict}_0 + \sum_{i \le n} r_i \cdot \Delta_i \equiv \text{Rand}_{n+1} = \text{Rand}_n + r_{n+1} \cdot \Delta_{n+1}$$

Random Version Invariant of IronDict and Aegon

$$\text{Rand vector} : \text{Rand}_n = \text{Dict}_0 + \sum_{i \leq n} r_i \cdot \Delta_i \equiv \text{Rand}_{n+1} = \text{Rand}_n + r_{n+1} \cdot \Delta_{n+1}$$

$$\text{proof of consistency from epoch i to epoch j} \iff \text{Rand}_i[t] = \text{Rand}_j[t]$$

Random Version Invariant of IronDict and Aegon

$$\text{Rand vector} : \text{Rand}_n = \text{Dict}_0 + \sum_{i \leq n} r_i \cdot \Delta_i \equiv \text{Rand}_{n+1} = \text{Rand}_n + r_{n+1} \cdot \Delta_{n+1}$$

$$\text{proof of consistency from epoch i to epoch j} \iff \text{Rand}_i[t] = \text{Rand}_j[t]$$

$$\text{auditor's task} \implies \text{ensure Rand}_n \text{ is computed correctly}$$

See Numbers!

	Aegon	SEEMLESS + Parakeet
Epoch Latency	60 s	60 s
Throughput	~1000 update/s	~1000 update/s
Lookup proof / verification	20KB / 29ms	2-4 KB / 0.1 ms
Lookup latency (server)	33ms (more computation)	0.5 s < (no computation)
Audit proof / verification	256B / 2ms	80 MB / 0.7s

Concrete Instantiation?

- Aegon = generic over any polynomial commitment schemes. We instantiate it with **KZH** [Kad+25] that requires a powers-of-tau setup ==> MPC ceremony setup

KZH-Fold:
Accountable Voting from Sublinear Accumulation

George Kadianakis[1], Arantxa Zapico[2], Hossein Hafezi[3], and Benedikt Bünz[4]

[1,2]Ethereum Foundation
[3,4]New York University

- Future Work: Design alternative polynomial commitments without a trusted setup.

Details of IronDict and Aegon?

- Check out the eprint
  - IronDict: https://ia.cr/2025/1580
  - Aegon: Would be online in less than a month!
- Talk on Details of IronDict and Aegon:
  - Check out talk we gave at Microsoft Research (thanks to Melissa Chase!)

Thanks!

Disclaimer: The content of this presentation, including all statements, interpretations, and opinions, reflects solely my personal views. Nothing presented here should be construed as representing or implying the views, opinions, policies, or positions of Meta or any of its affiliates.