

WhatsApp Key Transparency

Sean Lawlor
Software Engineer

Kevin Lewi
Software Engineer



Agenda

Overview

Infrastructure

Auditable Dictionaries

Overview

End-to-End Encrypted Messaging



Phone #s	Public Keys
Alice	pk_Alice
Bob	pk_Bob
...	...

Security Codes

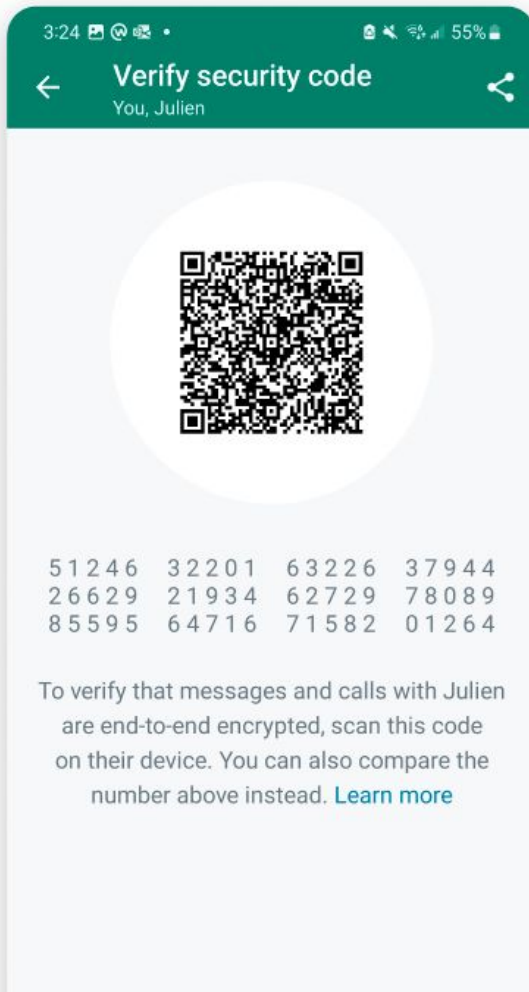


Alice and Bob can verify that they got the correct public key if they have an already-established authenticated channel (e.g. Zoom call, meeting up in person)

Code = Hash(Alice's public keys, Bob's public keys)

Note: this code changes every time Alice or Bob add a new device!

Also: Group chats?



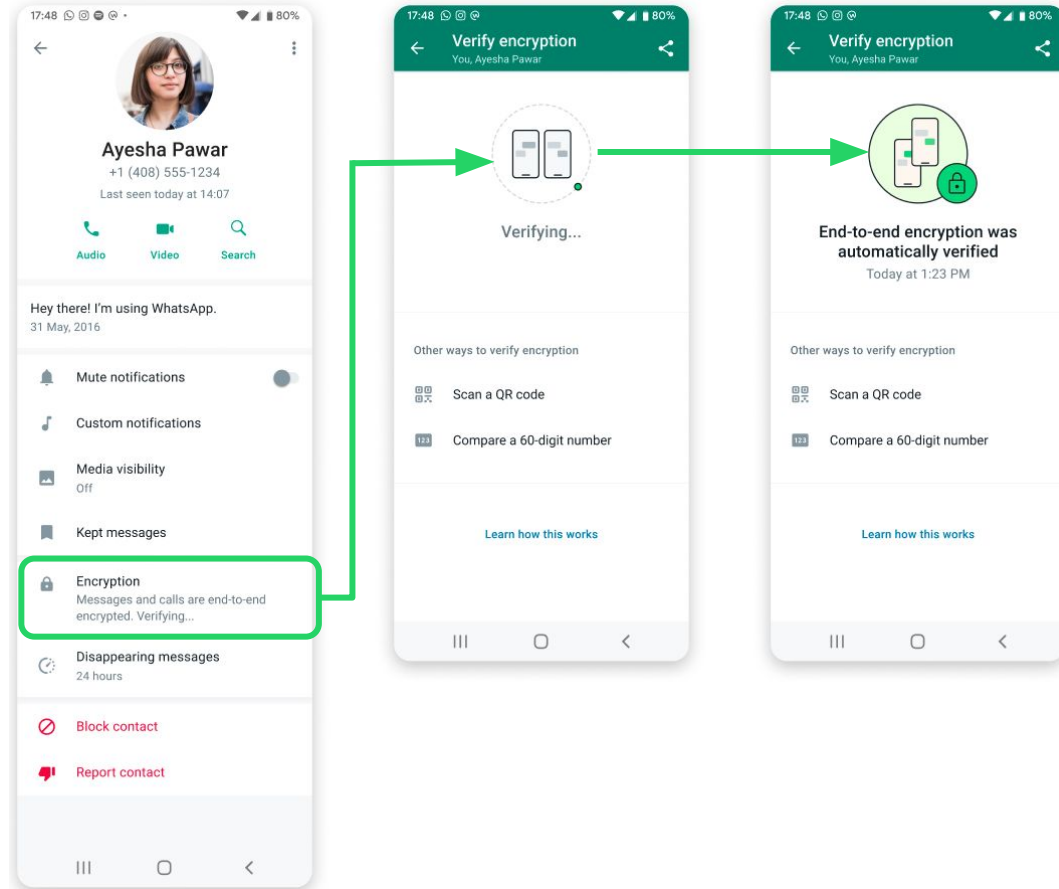
Key Transparency



Automatic validation of public keys

How?

- WhatsApp servers publish a commitment to the database of public keys
- Users check their public keys against this commitment to make they are consistent



Infrastructure

Normal Registration and Lookup

Registration (Write Path):



“Hi, I’m Bob, I want to register a new key:
4c94884df1bc...”



Database

User	Key
Alice	ecb6427d8ae8...
Bob	4c94884df1bc...
Charlie	95f64aee5f4d...

Lookup (Read Path):

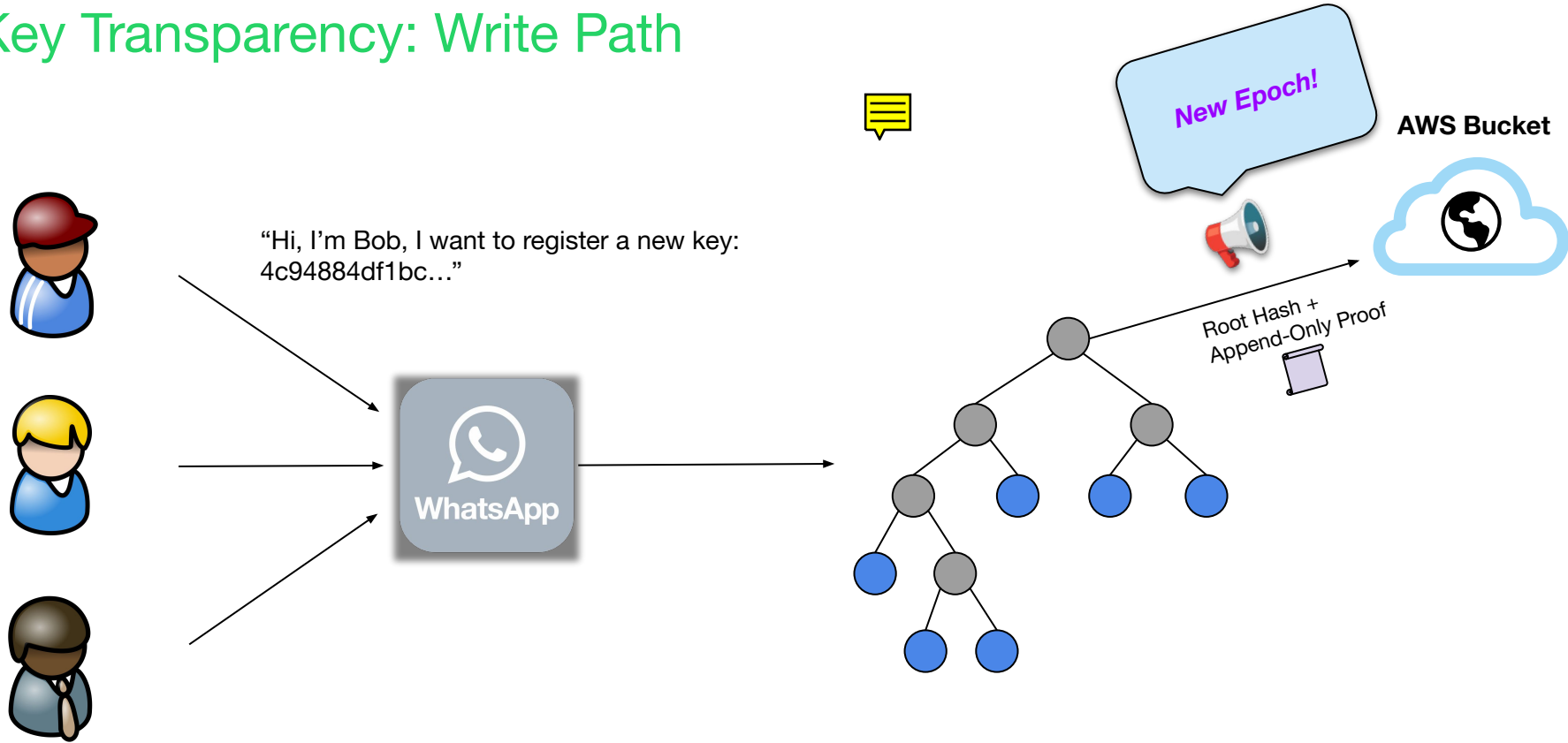


“What is Alice’s latest public key?”

ecb6427d8ae8...



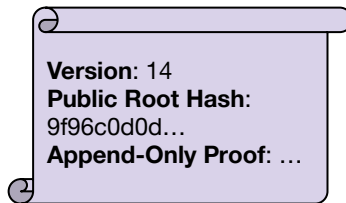
Key Transparency: Write Path



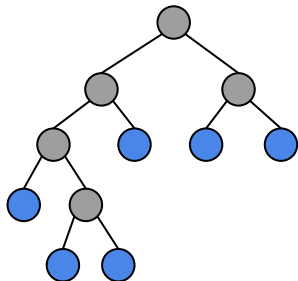
Key Transparency: Publish



Each publish contains:

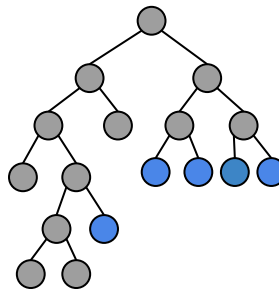


Version: 14
Public Root Hash:
9f96c0d0d583298...



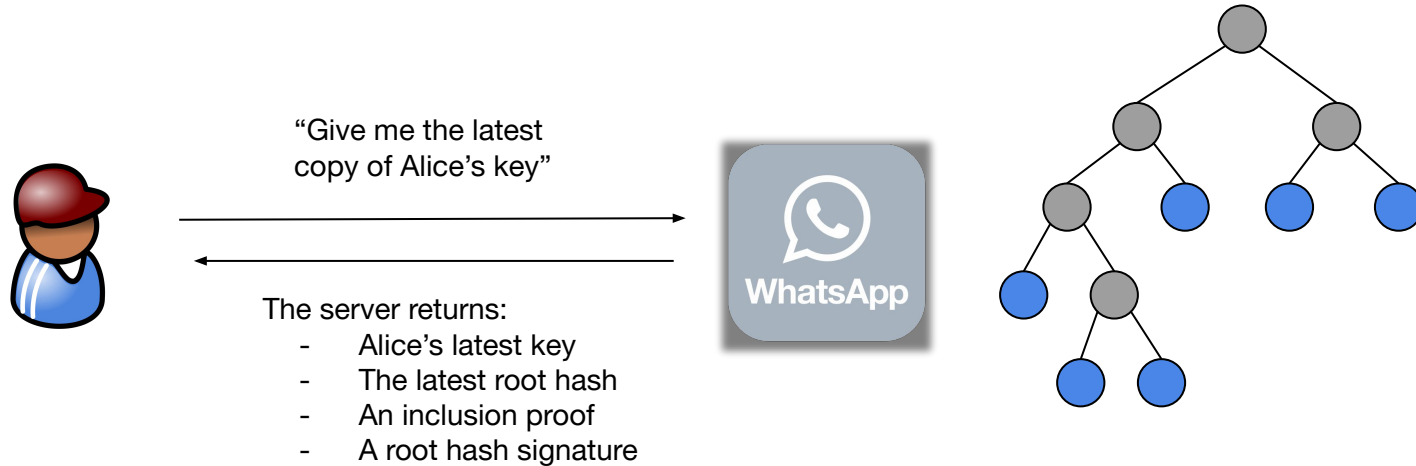
Append-Only
Proof

Version: 15
Public Root Hash:
481109384d45...



Append-only proofs guarantee that we manage the database consistently

Key Transparency: Read Path



Key Transparency: Read Path



AWS Bucket



Root hash signature ties the latest hash in AWS to what the client received

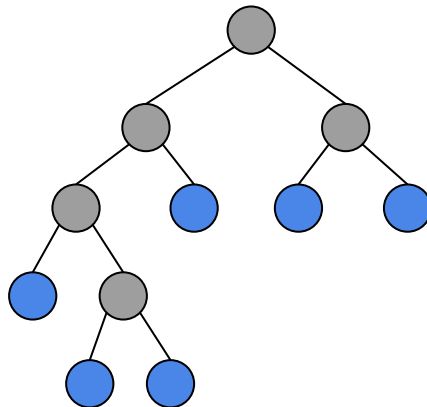


"Give me the latest copy of Alice's key"



The server returns:

- Alice's latest key
- The latest root hash
- An inclusion proof
- A root hash signature



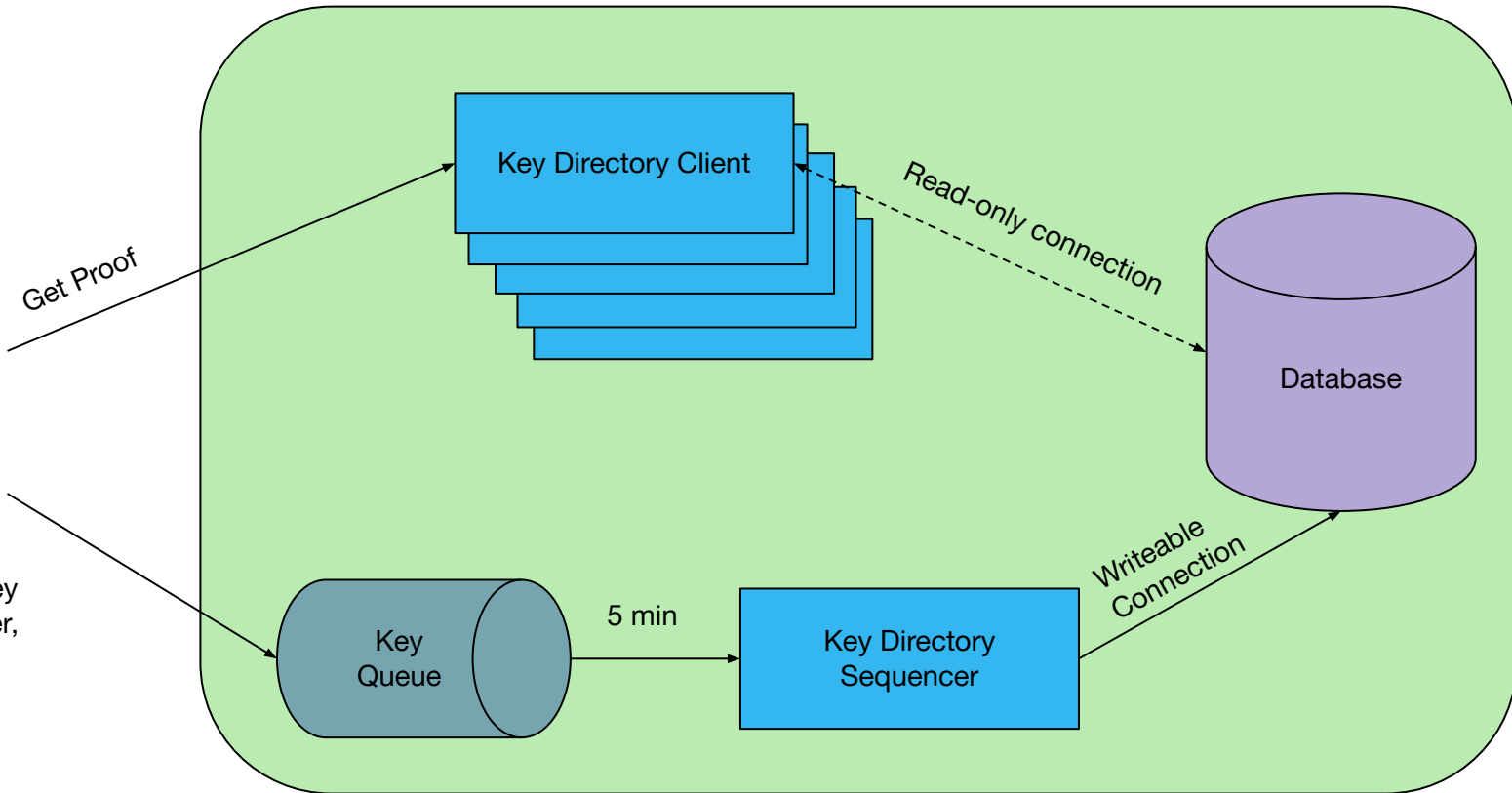
Infrastructure

Single writer, multiple readers

WhatsApp Infrastructure



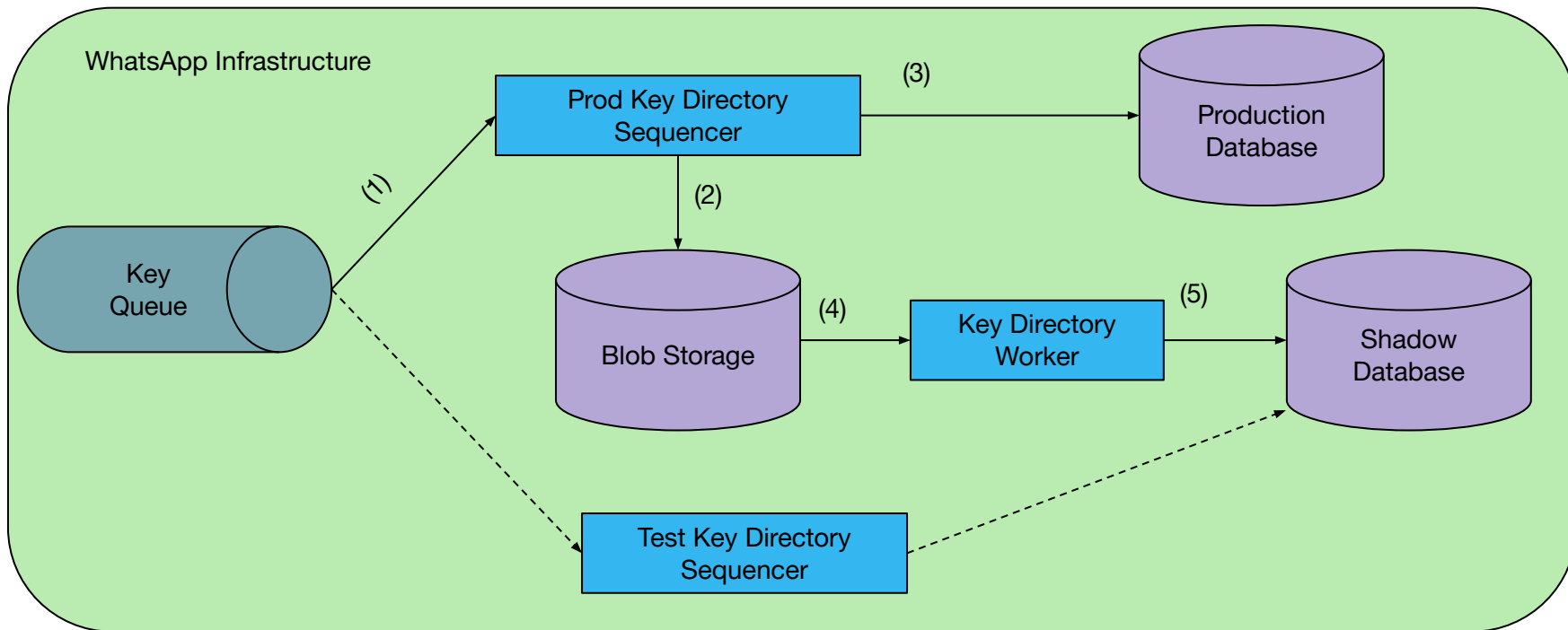
Change identity key
(register, re-register,
delete account)



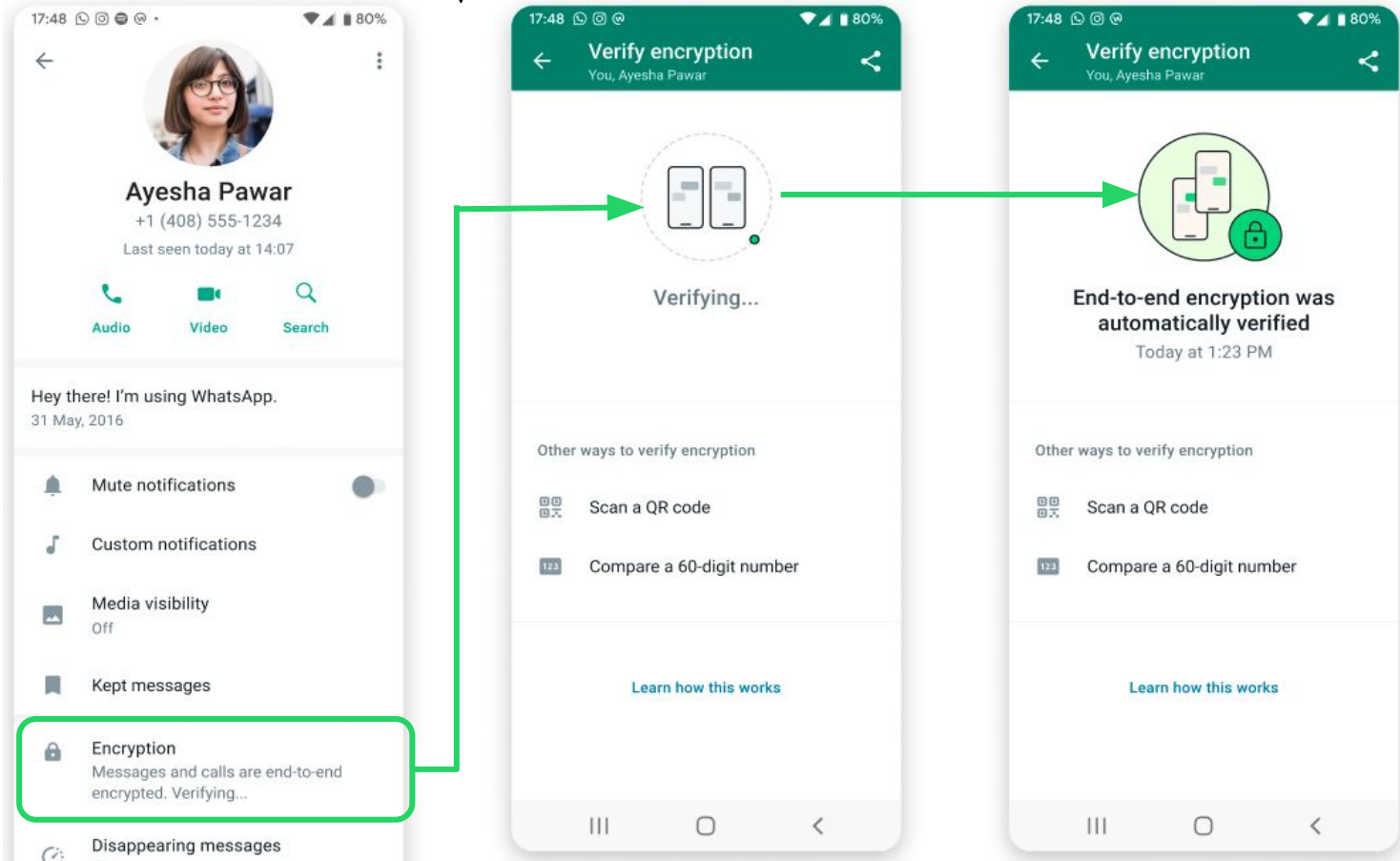
Infrastructure - Some Gotchas

Testing write flows

- Single writer - one binary tree to rule them all
- “Shadow” clone of prod database
- **Pause, resume, replay** all supported from shadow logs



Client Experience



Auditable Dictionaries

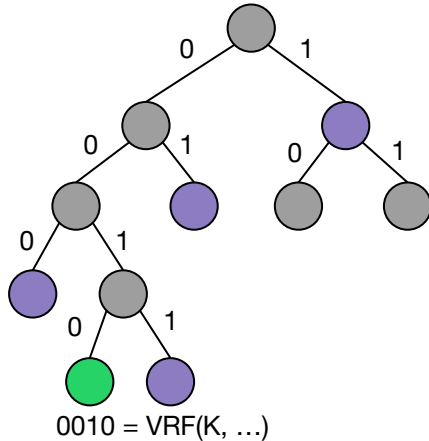
Construction: Sparse Merkle Trees and VRFs [CONIKS 2015]

Sparse Merkle Trees:

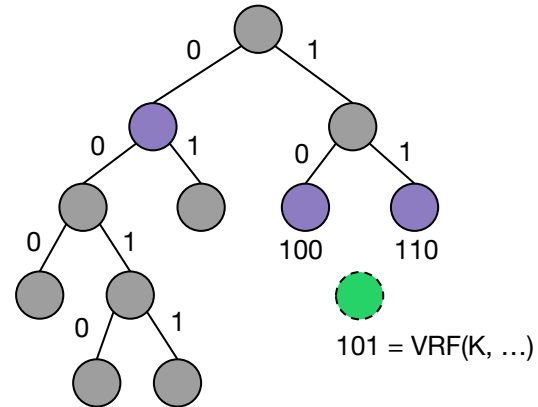
- Unique positions for entries
- Supports inclusion and non-inclusion proofs

We use Verifiable Random Functions (VRFs) to randomize leaf positions in the Merkle tree

Inclusion Proof



Non-Inclusion Proof



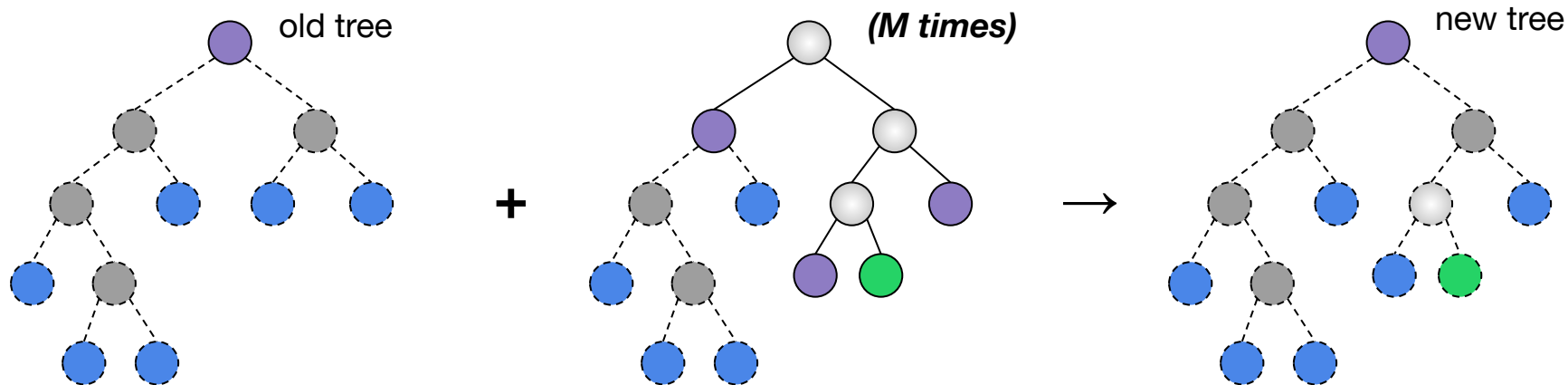
Large Append-Only Proofs

Auditors verify append-only-ness of the tree

Audit proofs only contain leaf values (hashes), not the raw public keys themselves.

However, proofs are $O(M \log N)$ in size, where: M = # of updates, N = total # of leaves in the tree

In practice: they are **~200 MB** each!



<https://github.com/facebook/akd>

- [illegible]

AKD: For Industry

<https://github.com/facebook/akd>

- Composable storage trait for flexibility
- Employs preloading nodes + caching to make operations more efficient
- Audited by NCC Group in Nov 2023



Auditable Key Directory (AKD) Implementation Review

Meta Platforms
Version 1.0 – November 14, 2023

©2023 – NCC Group

Prepared by NCC Group Security Services, Inc. for Meta Platforms, Inc. Portions of this document and the templates used in its production are the property of NCC Group and cannot be copied (in full or in part) without NCC Group's permission.

While precautions have been taken in the preparation of this document, NCC Group the publisher, and the author(s) assume no responsibility for errors, omissions, or for damages resulting from the use of the information contained herein. Use of NCC Group's services does not guarantee the security of a system, or that computer intrusions will not occur.

Prepared By
Elena Bakos Lang
G rald Doussot
Kevin Henry
Thomas Pornin

AKD: For Academics

<https://github.com/facebook/akd>

- We also have benchmarks to test against, future academic works may be able to use this
- Crypto operations are configurable
 - E.g. swapping out VRF, hash function



Related Work

Merkle-tree-based solutions:

- CONIKS [MBBFF'15]
 - SEEMless [CDGM'19]: more efficient history checks + privacy guarantees
 - Parakeet [MKSGOLL'23]: putting SEEMless into practice, handling deletion
- Merkle² [HHKYP'21]: Uses Merkle prefix tree + chronological tree together
- Rotatable Zero Knowledge Sets [CDGGKMM'22]
 - Addresses forward secrecy for VRF private key

Algebraic solutions:

- Transparency Logs via Append-Only Authenticated Dictionaries [TBPPTD'19]
- Verdict [TKPS'21], VeRSA [TFZBT'22]

Other implementations:

- Keybase [2015], Google [2017], Zoom [2020], Apple [2023], Proton [2023]

