ELEKTRA: Efficient Lightweight multi-dEvice Key TRAnsparency

Julia Len  
Cornell Tech

Melissa Chase  
Microsoft Research

Esha Ghosh  
Microsoft Research

Daniel Jost  
New York University

Balachandar Kesavan  
Zoom Video Communications

Antonio Marcedone  
Zoom Video Communications

CCS 2023
November 29, 2023
End-to-end encryption (E2EE)

- We expect from E2EE that the Service Provider cannot decrypt the communication
• We expect from E2EE that the Service Provider cannot decrypt the communication
• But how do Alice and Bob get each other’s keys?
End-to-end encryption (E2EE)

- Public keys are sent to the Service Provider to be stored and distributed
Meddler-in-the-Middle (MitM) attack

Alice and Bob have no idea that the Service Provider can read their messages!

Service Provider replaces Bob’s key with one it knows

Key Directory

<table>
<thead>
<tr>
<th>Alice</th>
<th>Bob</th>
<th>...</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
Current solutions offload security onto users

- Major messaging apps like Signal and WhatsApp have users compare a 60-digit “safety number” either manually or by scanning a QR code.
- Major video calling apps like Zoom and Microsoft Teams have users read out “security codes.”
Key transparency

Committments to the key directory are posted publicly when there are updates.
Key transparency

Clients can verify the proof w.r.t. the latest commitment.

Commitments to the key directory are posted publicly when there are updates.

Clients can query to get a username’s key and a proof that the reply is consistent.

Clients also periodically monitor the directory for their keys.

Public bulletin board

Auditable Key Directory

<table>
<thead>
<tr>
<th>Service Provider</th>
</tr>
</thead>
<tbody>
<tr>
<td>Alice</td>
</tr>
<tr>
<td>Bob</td>
</tr>
<tr>
<td>…</td>
</tr>
</tbody>
</table>
Key transparency: Privacy

**Bad for privacy:**
- Learning information about other keys in the directory during key lookups!
- External parties learn who updates their keys and when
Key transparency: Privacy

Privacy goals:
- Sensitive information about when users register or update keys should not get leaked to external clients.
- Clients should not learn information about other keys in the directory aside from the one they query.
- No privacy from the server.

Bad for privacy:
- Learning information about other keys in the directory during key lookups!
- External parties learn who updates their keys and when.
Key transparency: Privacy

Privacy goals:
• Sensitive information about when users register or update keys should not get leaked to external clients
• Clients should not learn information about other keys in the directory aside from the one they query
• No privacy from the server

Bad for privacy:
• Learning information about other keys in the directory during key lookups!
• External parties learn who updates their keys and when

Prior work in key transparency guarantees privacy by having the Service Provider choose a secret key and then computing a Verifiable Random Function (VRF) over data.
Key transparency: Privacy

Privacy goals:
- Sensitive information about when users register or update keys should not get leaked to external clients
- Clients should not learn information about other keys in the directory aside from the one they query
- No privacy from the server

Bad for privacy:
- Learning information about other keys in the directory during key lookups!
- External parties learn who updates their keys and when

Prior work in key transparency guarantees privacy by having the Service Provider choose a secret key and then computing a Verifiable Random Function (VRF) over data.

But this key could get compromised!
So we want privacy to be recoverable even after server compromise, a.k.a. post-compromise security
Key transparency: Multiple devices

Prior academic work model the single device setting
Mention extensions of mapping username to list of keys
Key transparency: Multiple devices

Prior academic work model the single device setting
Mention extensions of mapping username to list of keys
And in practice users can lose devices and need to **reset their accounts**!

Many E2EE apps allow for this, but this hasn’t been modeled at all by prior key transparency systems.

Prior academic work model the single device setting

Mention extensions of mapping username to list of keys
Key transparency: Multiple devices

And in practice users can lose devices and need to **reset their accounts**!

Many E2EE apps allow for this, but this hasn’t been modeled at all by prior key transparency systems.

Prior academic work model the single device setting

Mention extensions of mapping username to list of keys

A better approach: existing devices sign key updates

Makes MitM harder since the provider either needs to fake an account reset (suspicious!) or forge a signature.
New design desiderata

Strong multi-device security of Keybase

Post-compromise security of RZKS

Better privacy guarantees of SEEMless

Goal of our system!
ELEKTRA: A new key transparency system

- We model keychains which capture the evolution of a user’s public keys
- User keychains and their updates are stored in a multi-device verifiable key directory (MVKD)
- ELEKTRA is our MVKD construction
ELEKTRA: Challenge #1

**Auditabe Key Directory**

<table>
<thead>
<tr>
<th></th>
<th>Alice</th>
<th>Bob</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>🎄钥</td>
<td>🎄钥</td>
</tr>
<tr>
<td>...</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
ELEKTRA: Challenge #1
ELEKTRA guarantees that the attacker who compromised Alice’s device won’t be able to convince Bob that Alice previously had some other devices before the compromise.
ELEKTRA: Challenge #2

Can I learn about the keys of all the other users?

Auditable Key Directory

<p>| | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>Alice</td>
<td>![Key]</td>
</tr>
<tr>
<td>Bob</td>
<td>![Key]</td>
</tr>
<tr>
<td>...</td>
<td></td>
</tr>
</tbody>
</table>

Service Provider

Alice

Bob
Can I learn about the keys of all the other users?

ELEKTRA preserves privacy for honest users even when clients are compromised.

ELEKTRA: Challenge #2

Auditable Key Directory

<table>
<thead>
<tr>
<th></th>
<th>Alice</th>
<th>Bob</th>
</tr>
</thead>
<tbody>
<tr>
<td>Service Provider</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>...</td>
</tr>
</tbody>
</table>

Alice

Bob
ELEKTRA: Challenge #3

Alice

Service Provider

Bob

Auditable Key Directory

<table>
<thead>
<tr>
<th></th>
<th>Alice</th>
<th>Bob</th>
<th>...</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>🅿️</td>
<td>🅵️</td>
<td></td>
</tr>
</tbody>
</table>
ELEKTRA: Challenge #3

Auditable Key Directory

<table>
<thead>
<tr>
<th></th>
<th>Alice</th>
<th>Bob</th>
</tr>
</thead>
<tbody>
<tr>
<td>Alice</td>
<td>🪪</td>
<td>🪪</td>
</tr>
<tr>
<td>Bob</td>
<td>🪪</td>
<td>🪪</td>
</tr>
<tr>
<td>...</td>
<td>🪪</td>
<td>🪪</td>
</tr>
</tbody>
</table>

Service Provider
ELEKTRA offers post-compromise security, so the attacker doesn’t learn key updates after rotating the Service Provider’s key.
ELEKTRA: Rigorous security proofs

**Completeness**
- Desired functionality for honest parties interacting with an honest server
- Dishonest clients should not be able to affect the protocol for honest clients

**Soundness**
- Security in the presence of an active and fully compromised server
- We define a stronger form of soundness, which is extractable soundness

**Privacy**
- Algorithms don’t leak extra information about the server’s state other than some well-defined leakage function
- More complex than prior definitions: we model corrupted clients and a corrupted server (for PCS guarantees)
Experiments

• Implementation written in Go
• Server run on AWS instance, client run on Google Pixel 6 phone
Experiments: Query

- Implementation written in Go
- Server run on AWS instance, client run on Google Pixel 6 phone
- Simulate joining a small group with 10 unknown users, each with 10 key updates

![Query runtime and proof size graphs]

- **Query runtime**
  - Time (ms) vs. $N$ (links)
  - Client, Latency, Server

- **Query proof size**
  - Bandwidth (KiB) vs. $N$ (links)
  - 1M, 4M, 16M, 64M
Experiments: Update

- Implementation written in Go
- Server run on AWS instance, client run on Google Pixel 6 phone

- In the graph, we measured how long it takes to add 10 random key updates for various directory sizes
- Our experiments also show that ELEKTRA can add 128 keys in about a second to a directory containing 64M keys
- PCSUpdate for a directory of 4M keys takes about 30 minutes
<table>
<thead>
<tr>
<th>System</th>
<th>Strong privacy guarantees</th>
<th>Post-compromise security</th>
<th>Strong multi-device security with account resets</th>
<th>Rigorous security analysis</th>
<th>Efficient</th>
</tr>
</thead>
<tbody>
<tr>
<td>CONIKS [MBBFJ Sec’15]</td>
<td>✗</td>
<td>✗</td>
<td>✗</td>
<td>✗</td>
<td>✓</td>
</tr>
<tr>
<td>SEEMless [CDGM CCS’19]</td>
<td>✓</td>
<td>✗</td>
<td>✗</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td>Merkle(^2) [HHKYP SP’21]</td>
<td>✗</td>
<td>✗</td>
<td>✗</td>
<td>✗</td>
<td>✓</td>
</tr>
<tr>
<td>Parakeet [MKS+ NDSS’23]</td>
<td>✓</td>
<td>✗</td>
<td>✗</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td>ELEKTRA</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
</tr>
</tbody>
</table>
Conclusion

Email: jlen@cs.cornell.edu
https://www.cs.cornell.edu/~jlen/

• Formally model and construct ELEKTRA: the first key transparency system with strong multi-device support

• First key transparency system with post-compromise security for privacy guarantees

• Rigorous security definitions!
  • Completeness, Soundness, Privacy

• Experiments show our protocol is efficient for real-world loads