Adopted!
Feedback from IETF 118

- How would Sealed Sender work?
- Do we say anything about federation?
- Lifecycle management
- Compliance with privacy laws
- Disagreement about privacy properties
Changes since IETF 118

- Support for anonymous end-to-end encryption
- Discussion of federation
- Recommendations for lifecycle management
- Compliance with privacy laws
- New privacy properties
Anonymity
How does anonymous e2ee work?

1. Sender gets a receiver’s credential (potentially in an expensive way)
How does anonymous e2ee work?

2. Sender sends their own credential, encrypted, over a much cheaper anonymous channel.

Encrypt(credential, credential || "Hi!")
How does anonymous e2ee work?

3. Receiver downloads and decrypts (no anonymous channel needed)
How does anonymous e2ee work?

4. Receiver responds (optional)
Adjusting KT to fit this pattern

- **Problem:** KT is an interactive protocol.
- **Solution:** Make users an anonymizing proxy for looking up their own public key
Federation
Rules for Federation?

- Generally just one: ensure lookup keys are properly namespace\(d\) (@domain.com)

- Secondarily: If you want to protect the privacy of your users, act as an anonymizing proxy (à la OHTTP) rather than mirror other federation members’ logs
  - Minimizes privacy concerns
Lifecycle Management
- Pre-populate new log
- Store final root hash of old log in new log
- Allow clients to finish their monitoring of the old log
Pruning

User data that is not needed by the application can be deleted freely.
Pruning

KT-specific data that is no longer needed can be garbage collected
Privacy Law
User DATA

- User data can be pruned!
- We will MAYBE need to retain a commitment
- Without the commitment’s opening, leaks nothing
User NAMES

- Lookup keys (usernames, phone numbers) will likely be stored in some kind of prefix tree
- This can be pruned as well, but leaves behind a common prefix
- Lookup keys are randomized with a VRF: the common prefix will likely be too short to uniquely identify anyone
Privacy
Before:

- No padding / storage overhead
- Outside observers see: Users < $2^{256}$
- Third-party Auditor CAN tell whether an update creates a *new* lookup key or updates an *existing* lookup key

After:

- Requires padding initial userbase with fakes (maybe 100GB overhead)
- Outside observers see: Users < $2^{32}$ (Maybe $2^{31}$?)
- Third-party Auditor CAN’T tell whether an update creates or changes a lookup key

Better for service operator

Better for individual users
The End