Challenges With Building End-to-End Encrypted Applications – Learnings From Etesync

stosb.com/talks

Tom Hacohen
FOSDEM 2019

tom@stosb.com
@TomHacohen

Who Am I?

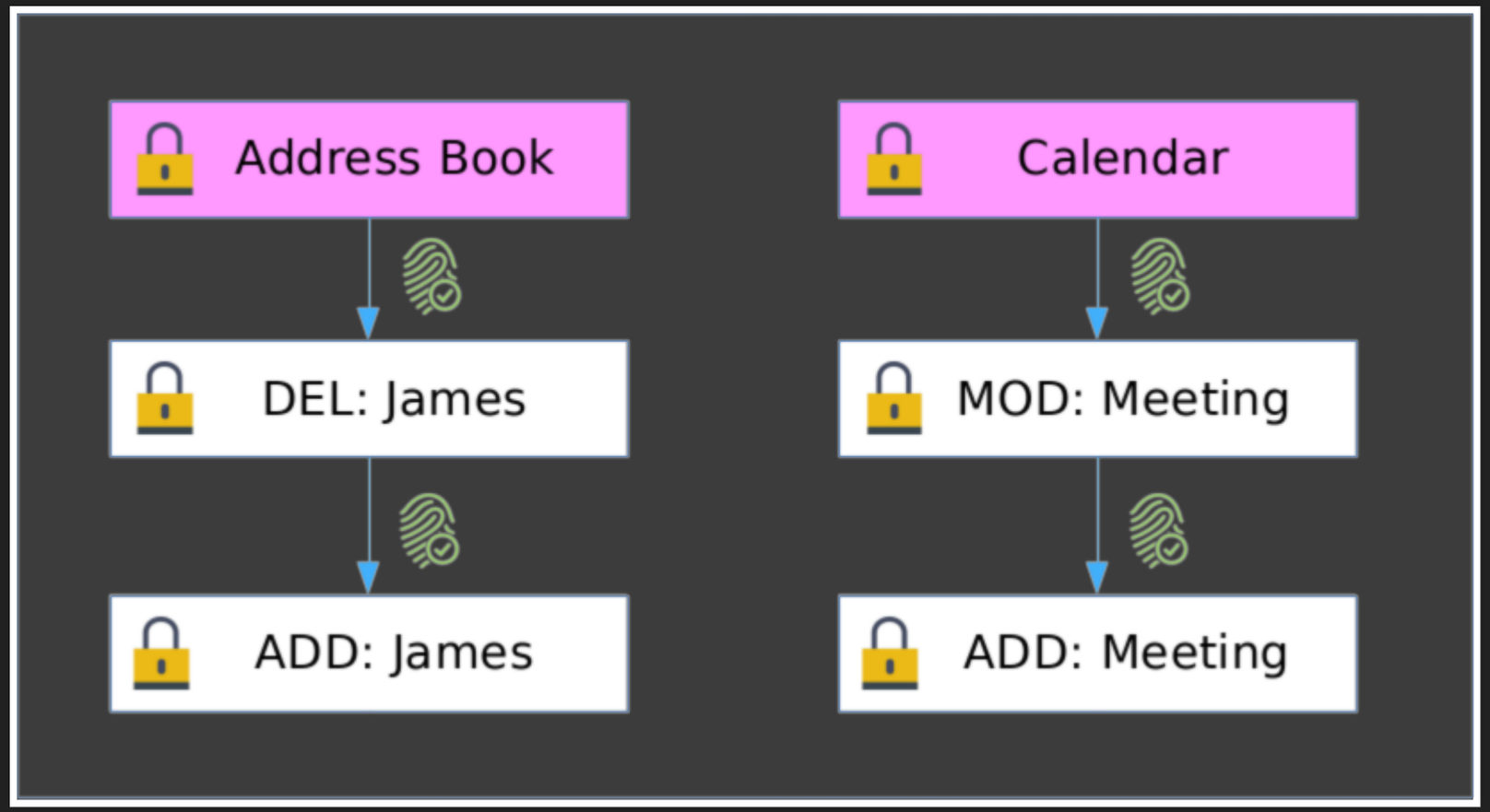
- Long time Open Source developer
- Privacy and digital security enthusiast
- Maintainer and creator of EteSync
- Building a security startup with Entrepreneur First

EteSync

Secure, end-to-end encrypted and fully versioned personal information sync for Android, the desktop and the web. Currently supports contacts, calendars and tasks, with more on the way.

EteSync Overview

Encrypted and Tamper-Proof Journal



EteSync Overview

How Are the Encryption Keys Derived

```
# The key from all of the others are derived
master_key = Scrypt(user_email, password)

# An encryption key for each journal and its contents
journal_key = HMAC_SHA256(journal_uid, master_key)
journal_key_enc = HMAC_SHA256("aes", journal_key)
journal_key_mac = HMAC_SHA256("hmac", journal_key)
```

EteSync Overview

How Is the Data Encrypted

```
# The journal itself (meta information)
journal_info_enc = iv + AES_CBC_PKCS7(iv, journal_key_enc, journal_info_clear)
journal_info_mac = HMAC_SHA256(journal_info_enc + version, journal_key_mac)

journal_info = journal_info_mac + journal_info_enc
journal_uid = RANDOM_SHA256()  # A random sha256 like blob


# The journal entries:
prev_uid = PREVIOUS_UID  # The uid of the previous entry

entry_info = iv + AES_CBC_PKCS7(iv, journal_key_enc, entry_info_clear)
entry_uid = HMAC_SHA256(prev_uid + entry_info + version, journal_key_mac)
```

So Let's Talk About the Challenges…

Platform Portability

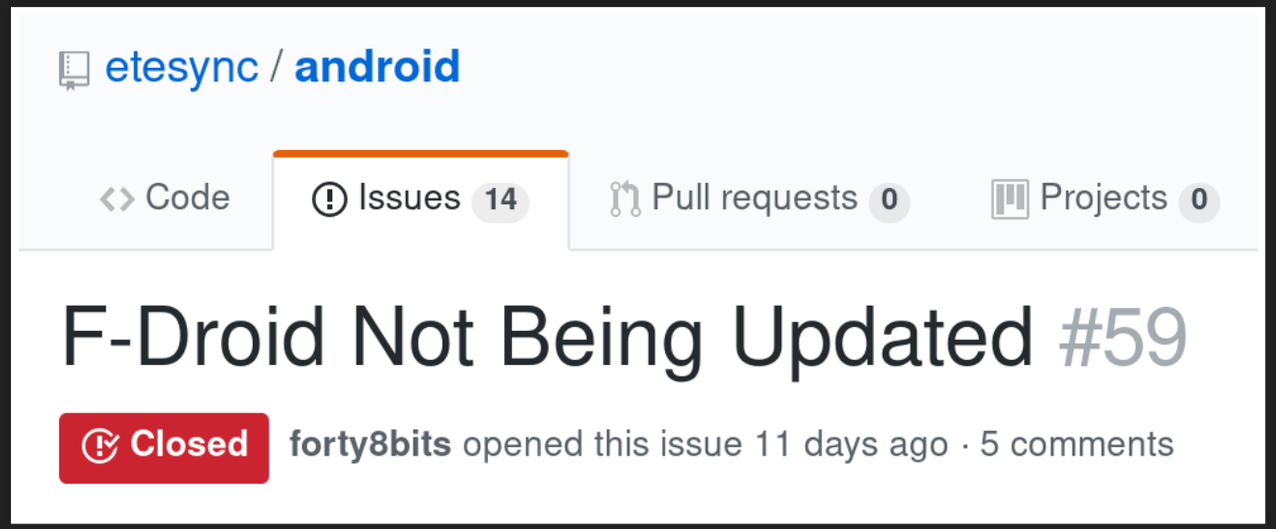
- Everything is implemented on the client, so…
- *All* clients need libraries for *all* crypto primitives
- Want library support on all platforms (e.g. iCal support)
- Need to write the same code for all platforms

Account Init & Protocol Upgrade

- Everything is implemented on the client, so…
- On every client:
  - Account init code - set initial state
  - Account upgrade code - changes in data format
  - Support for past and current protocol versions
- Partial *"solution"*: only implement in *master* clients

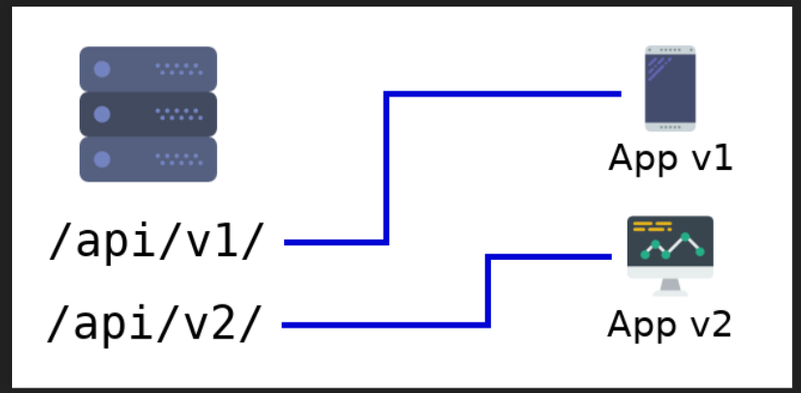
Protocol Upgrade

- Every client needs to support the new version, so either…
  - Update all apps simultaneously (hard with F-Droid)
  - First deploy support, and then deploy upgrade logic



Protocol Upgrade (Part 2)

- You can't transform the data on the server, so…
  - You can't support multiple API versions at once
  - Gracefully handle future unsupported versions



What's Considered a Protocol Upgrade?

- Everything.
- Changing cryptography methods (e.g. elliptic curves)
- Changing cryptography parameters (e.g. for Scrypt)
- Changing the structure of the data
- Every other thing you can think of

Development Speed

Did I mention everything needs to be implemented on every client?

Debugging

- You can't ask for data, and when you do, you often won't get it
- No access to data make it hard to investigate issues
- Can't test changes and fixes on existing data
- Can't look in the data for affected users
- Have to rely on users to test and reproduce on their own devices

3rd Party Applications

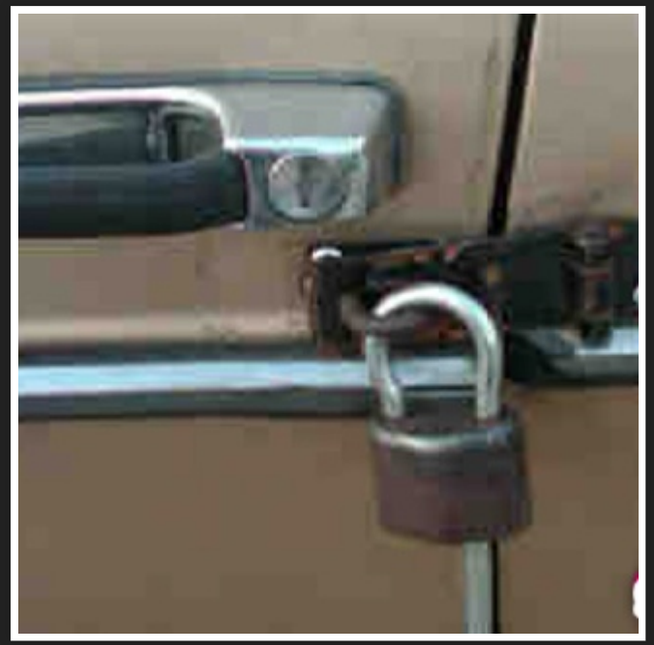
- We can't trust 3rd parties with encryption passwords, so…
  - We can't easily add integrations with 3rd party apps
- Never let users enter credentials in 3rd party apps

Data Immutability

- Because the journal is immutable:
  - You can't fix saved malformed data
  - Can't update the saved format
  - Always need to support old formats and malformed data
  - You guessed it. On *all* the clients!

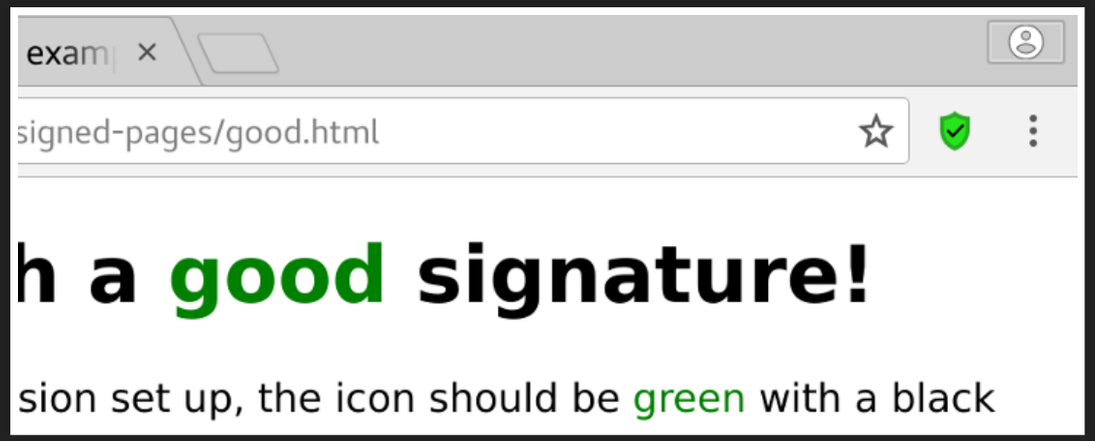
Usability Issues

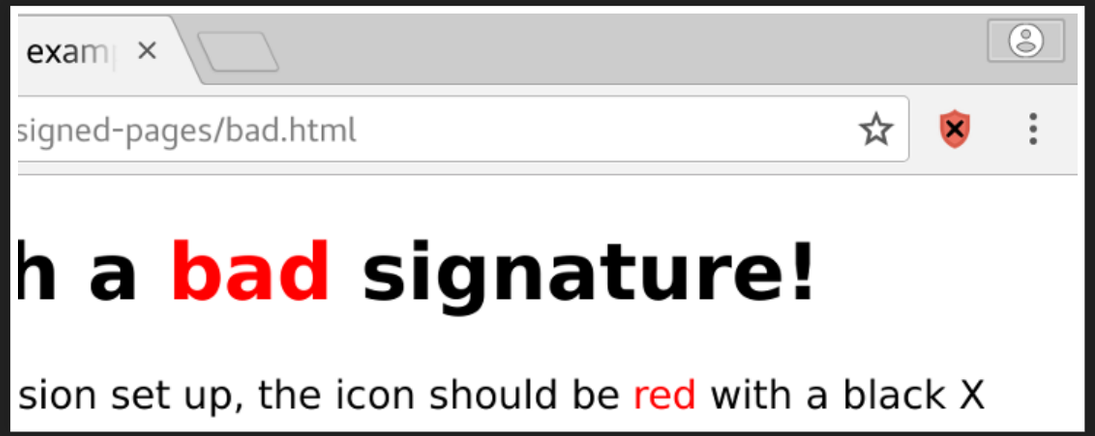
- Having both an encryption and a login password
- Encryption password recovery is not straightforward



You Are Held to a Higher Standard

- Signed Pages - secure web app delivery
- As mentioned before: can't ask for data
- Watch out with what you put in logs and debug info





A Few More Things to Watch out For…

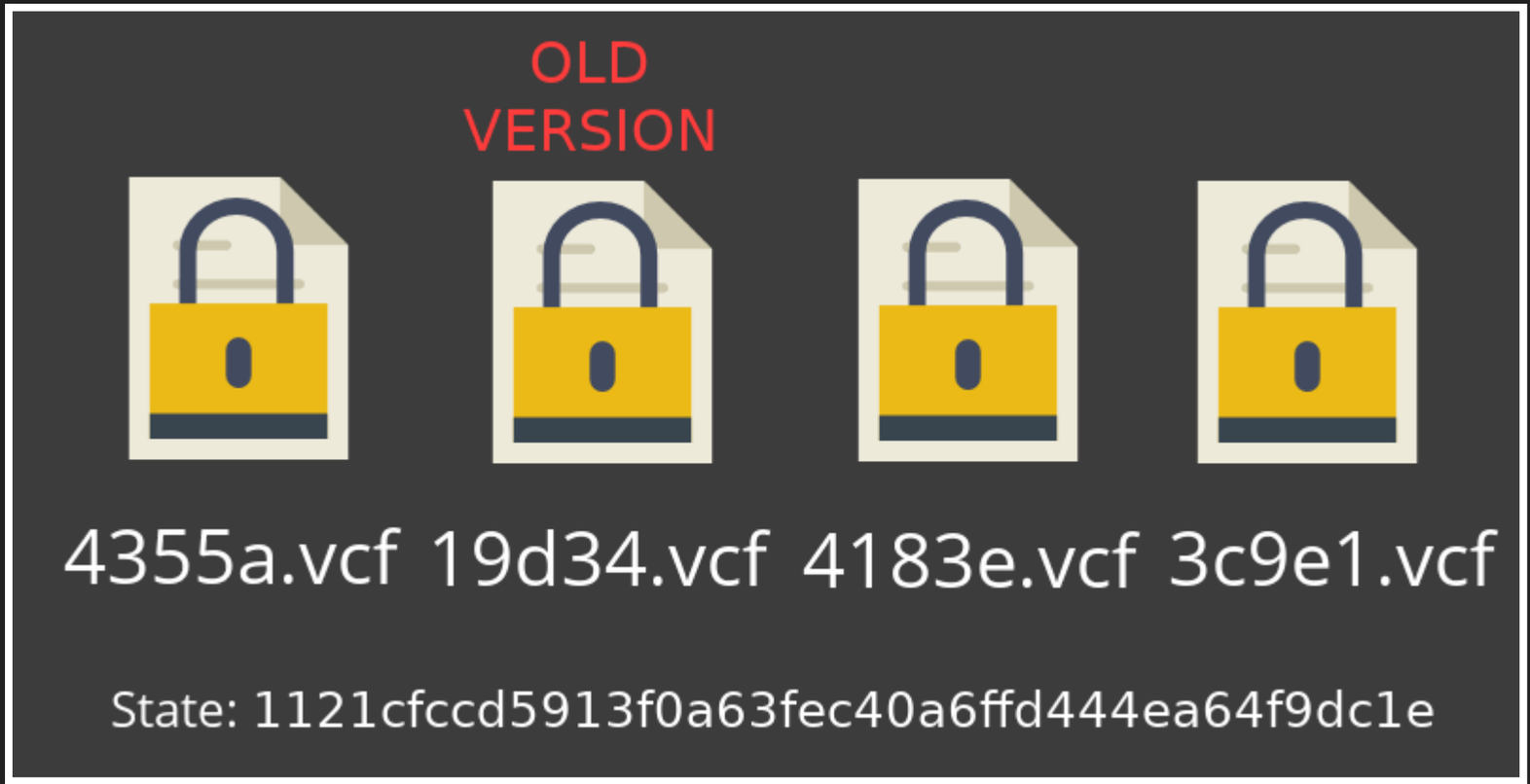
Performance Considerations

- No server-side search or processing
- Have to download all the data, or at least an index
- However, most operations are fast because they are local

A False Sense of Security

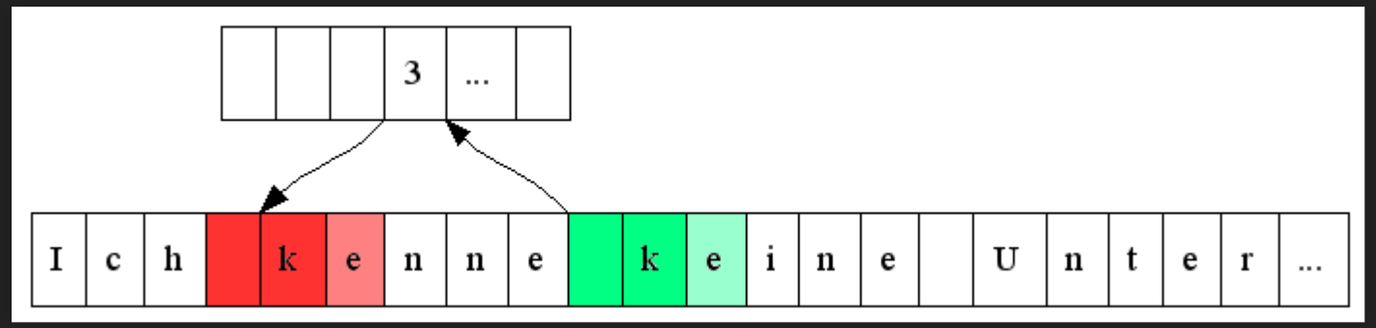
- Revoking or changing encryption passwords:
  - Encrypt the old key with the new key (potentially insecure)
  - Re-encrypt the whole data (problematic)
  - Old key for past data, new key for new data (complex)
- Offer alternatives? How do you educate users about the trade-offs?

Replay and Downgrade Attacks



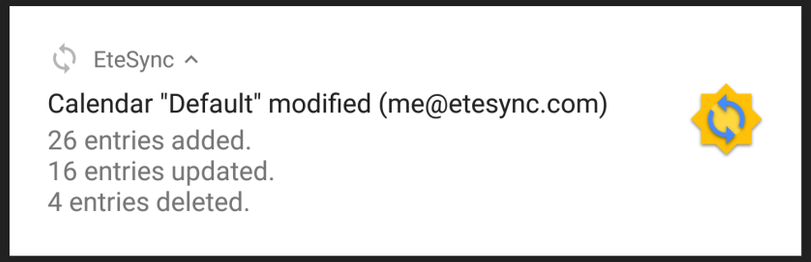
Leaking User Data

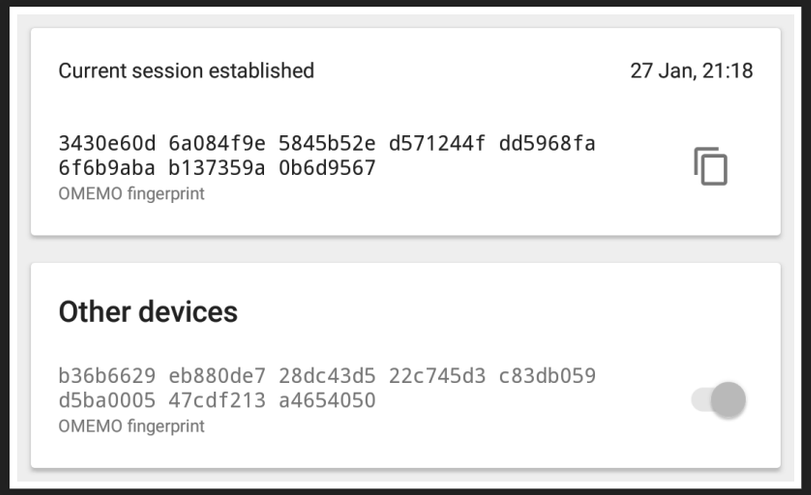
- Sensitive information in logs and debug info
- Mixing together user-controlled and non-user controlled data
- Optimisations often leak data, for example:
  - Compressing data can often lead to leaks
  - De-duplicating files using clear-text SHA256 sum
  - Variable bitrate audio and video



The UI Can Make All the Difference

- Informing users when data is changed
- Showing users how many devices are active
- There are other potential flaws and safeguards





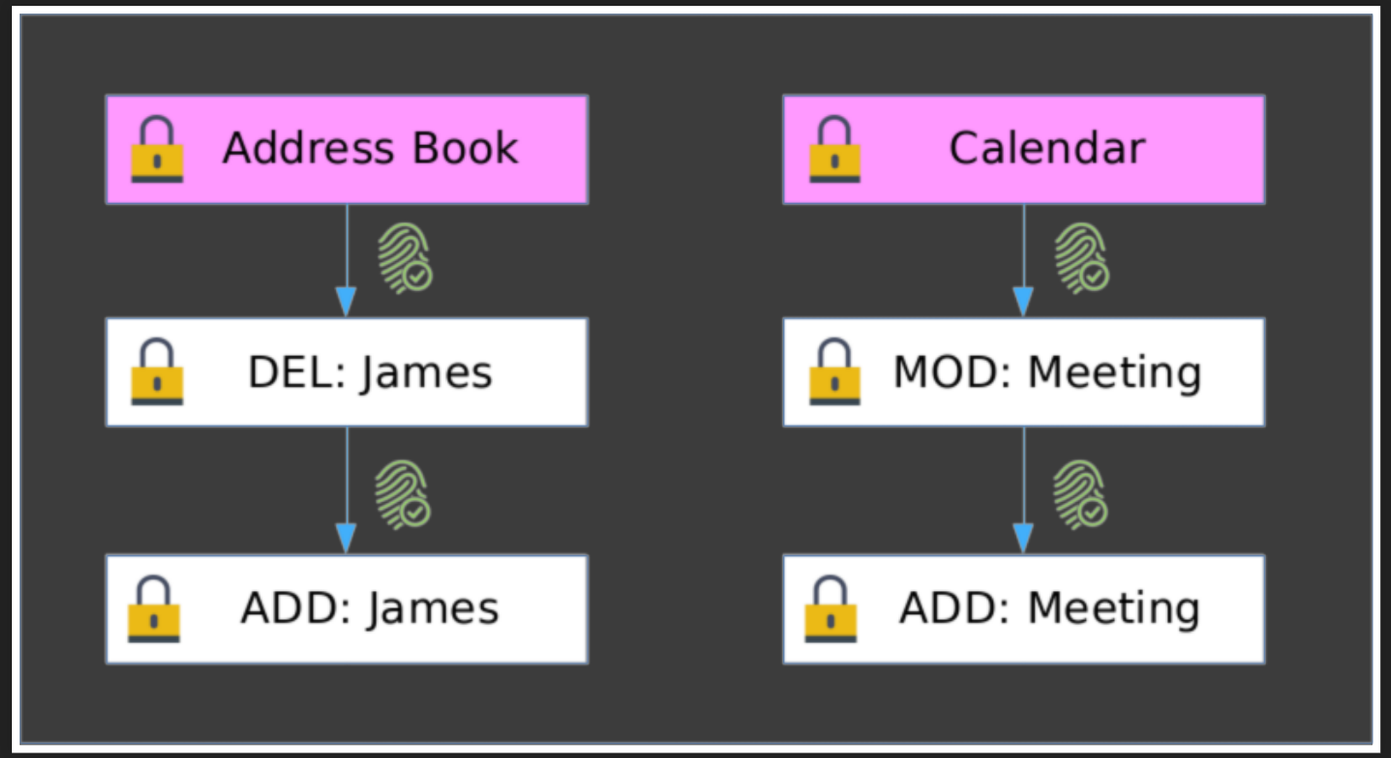
Improving the EteSync Protocol

Untying the Username and Encryption Key

- Deriving the key from the username proved problematic
- Was a useful shortcut but a big pain
- Can't easily change the username
- It accidentally made the user inconsistently case sensitive

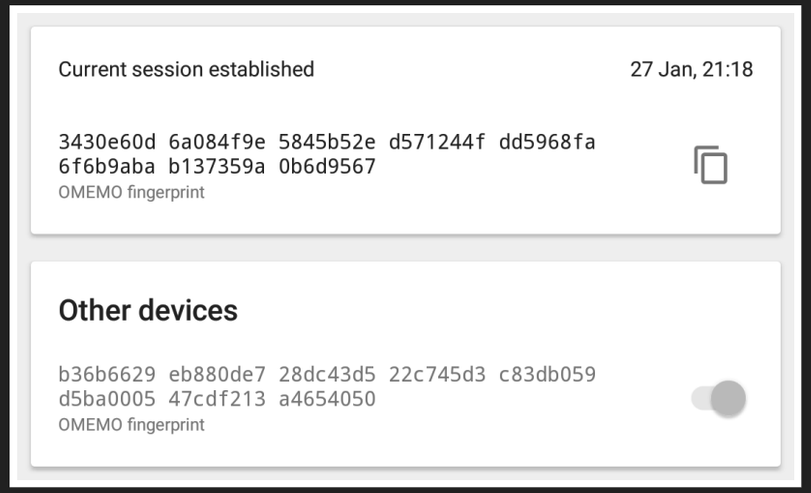
Improve Integrity Assurances

- Sign journal items (rather than just HMAC)
- HMAC the global state + have a counter



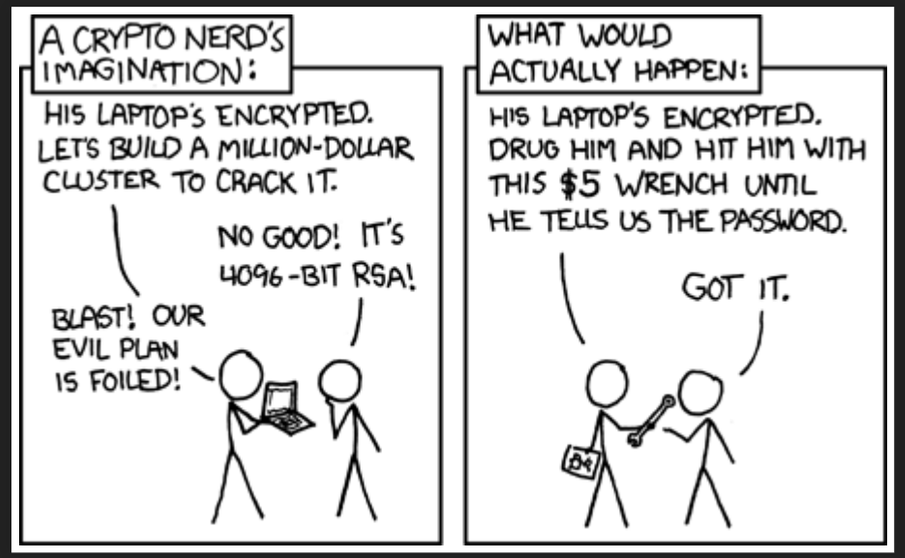
Move to Per-Device Keys

- Can better use hardware tokens
  - Sometimes built-in hardware!
- Can better handle lost devices
- Useful infrastructure for 3rd party apps
  - Can have a key per app, not just device…



Finishing Notes

- End-to-end encryption is the only way forward!
- Privacy is a sacred right, don't give it up!
- You're the weakest link:



Useful Links

- My blog: https://stosb.com
- EteSync's website: https://www.etesync.com
- EteSync's sources: https://github.com/etesync
- Signed Pages: https://github.com/tasn/webext-signed-pages

Questions?

stosb.com/talks

Tom Hacohen
FOSDEM 2019

tom@stosb.com
@TomHacohen

Attribution

- Icon by Freepik from flaticon.com is licensed under CC 3.0 BY
- Icon by Smashicons from flaticon.com is licensed under CC 3.0 BY
- Compression example by the unicode.org
- Security by Randall Munroe (XKCD)