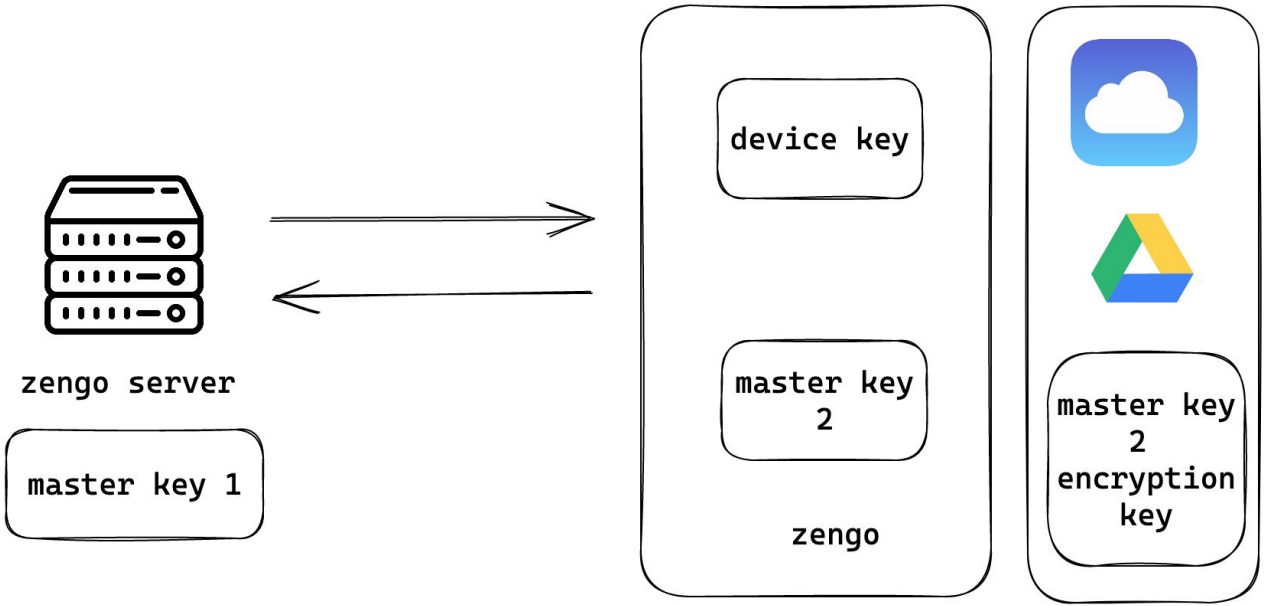
A security review of Zengo Wallet

From a privileged attacker’s point of view

Executive Summary

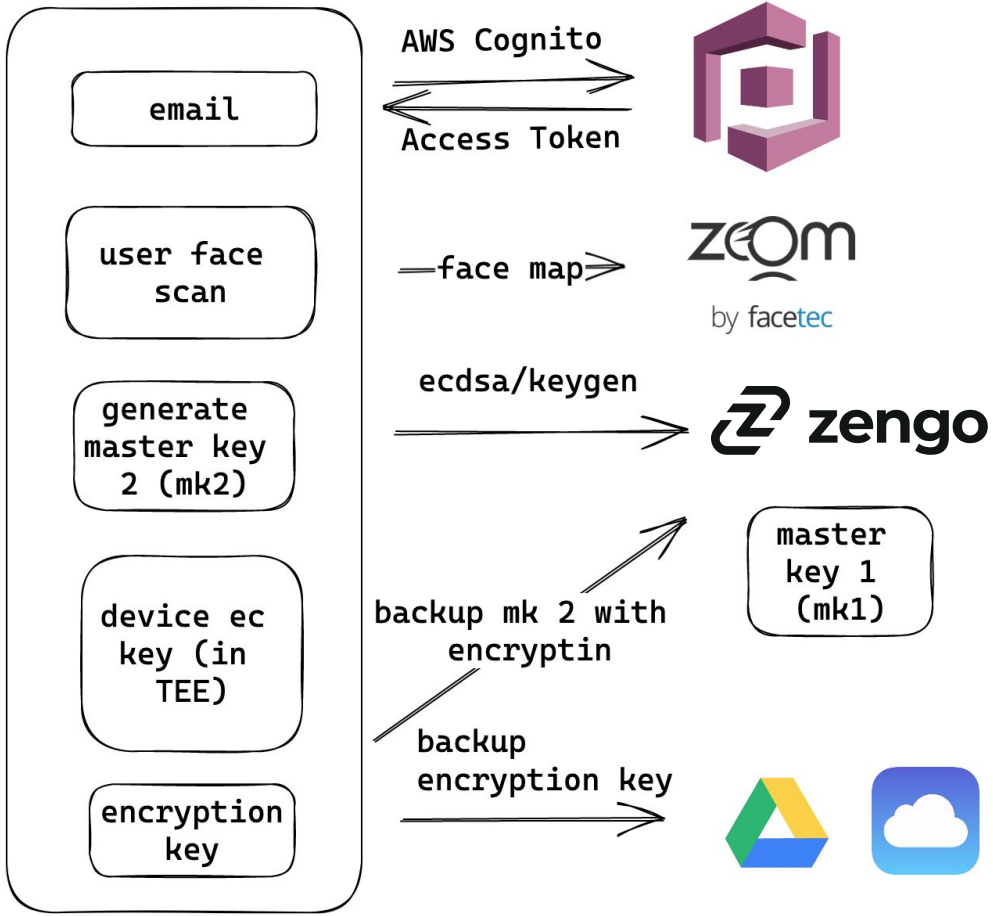
- Zengo's MPC solution provides stronger security defenses than regular mobile wallets.
- In particular, Zengo's wallet can defend against direct attacks from privileged attackers, such as those who leverage zero-day vulnerabilities or advanced malware to gain root access on user devices, especially for high profile wallet users.
- The use of FaceTec's 3D FaceLock technique makes it nearly impossible to retrieve the backup of master key2, and the appropriate usage of TEE for signing message with timestamp between device and server makes even privileged user difficult to attack.
- CertiK identified & helped Zengo fix a unique issue that could allow a privileged user to grab the JWT token and enroll a new device key, which would allow an attacker to construct transaction calls with a new device key and master key 2.

Security Architecture Overview



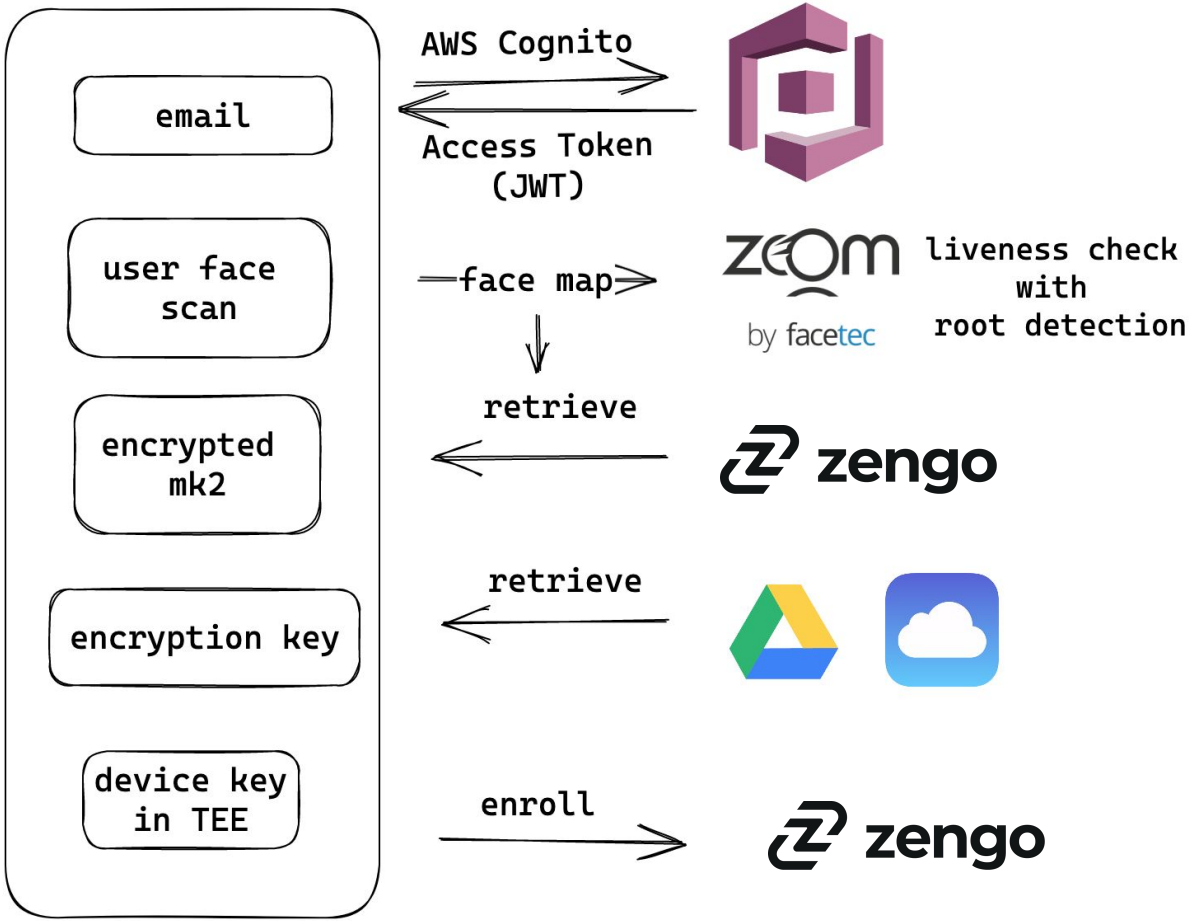
User Sign Up

- FaceTec's 3D FaceLock required
- Two master keys generated (one at local one at Zengo server)
- One device key generated in TEE for following message signature
- Master key 2 encrypted with local ec key and then sent to server for backup
- EC key is backed up in either google drive or iCloud



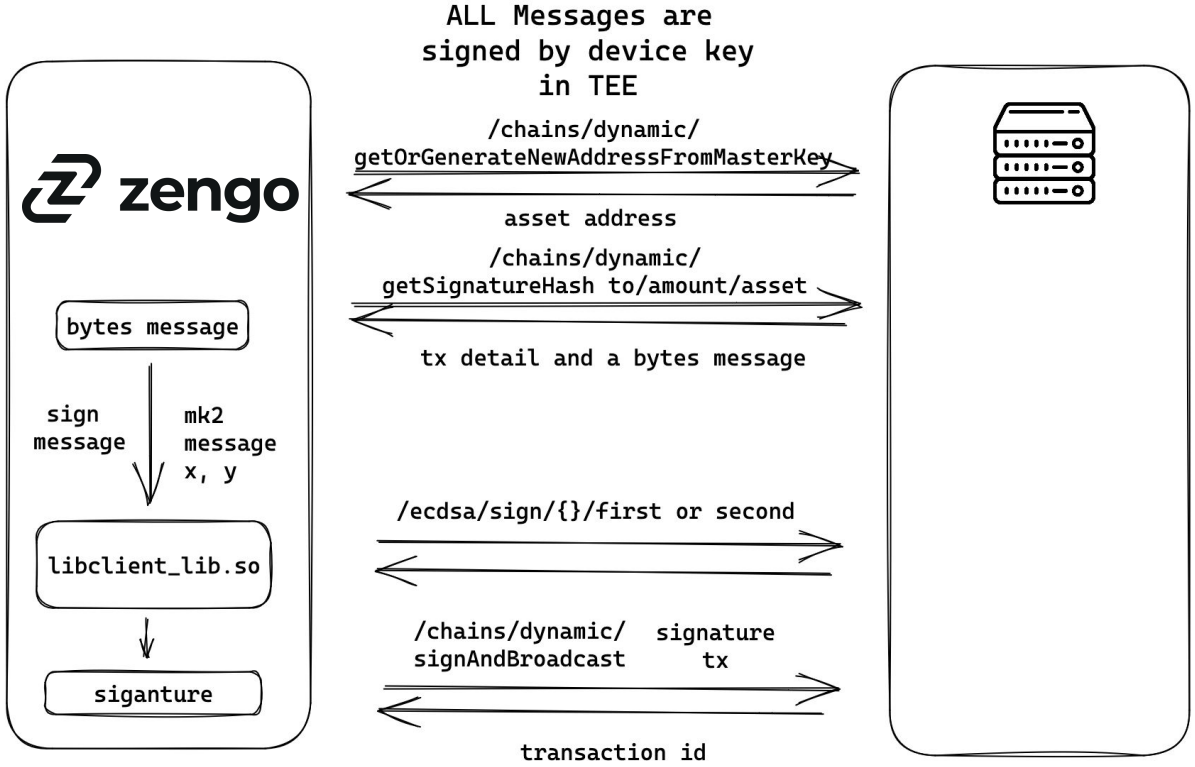
User Recovery

- Encrypted master key 2 retrieved after 3D FaceLock working
- New device key generated and enrolled for later message signature generation



Transactions

- Device get transaction details from server
- Transaction details include a message for master key 2 to sign and blockchain tx
- Two party ECDSA so library called to interact with server using masterkey 2 in ***plaintext***
- Client sent back server the signature and then server broadcast the tx



Where are the keys stored?

- Master key 1
  - Zengo server, user can never get access to it, only the master key 1 id
- Master key 2
  - Encrypted local storage (react-native-secure-storage) if not used
  - Encryption key generated in TEE, decrypted in TEE
  - Used as plaintext format in memory
- Device key
  - Generated and stored in TEE, cannot be exported
- JWT Token
  - Cleartext format stored in local sqlite3 database

For Privileged Attackers

- Cannot get the encrypted master key 2 unless passing the 3D FaceLock
  - The 3D FaceLock session token only valid for a short period of time
- Can intercept the master key 2 as it eventually needs to be used in a so library written in rust
  - https://github.com/ZenGo-X/gotham-city
- Can read JWT token from local storage to construct requests to server
- Still need a valid device key to interact with the server
  - All message are signed and validated on server side
- Device key cannot be extracted as it’s generated and stored in TEE

Resolved Issue

- The issue: Enrolling a new device key does not need 3D FaceLock, only master key recover API requires extra verification
  - Attack steps:
    - Extract master key 2 from memory
    - Read JWT token from local sqlite3 database
    - Generate a new ECDSA key pair with SHA256 signature hash scheme
    - Enroll this new key pair by calling “device/setDevicePK”
    - Perform every single API call to Zengo server afterwards
  - Two party ecdsa sign library available on Github and is the same as the one used in Zengo APP
- The fix:
  - restricting device key enrollment API with FaceTec authentication
  - Fix deployment verified

Conclusions

- We believe Zengo can prevent even privileged users from accessing user funds. Defending against privileged attackers is a difficult task, and not many mobile wallets can handle it.
- Zengo's security practices demonstrate a comprehensive approach to protecting users, surpassing those of many regular wallets on the market today.