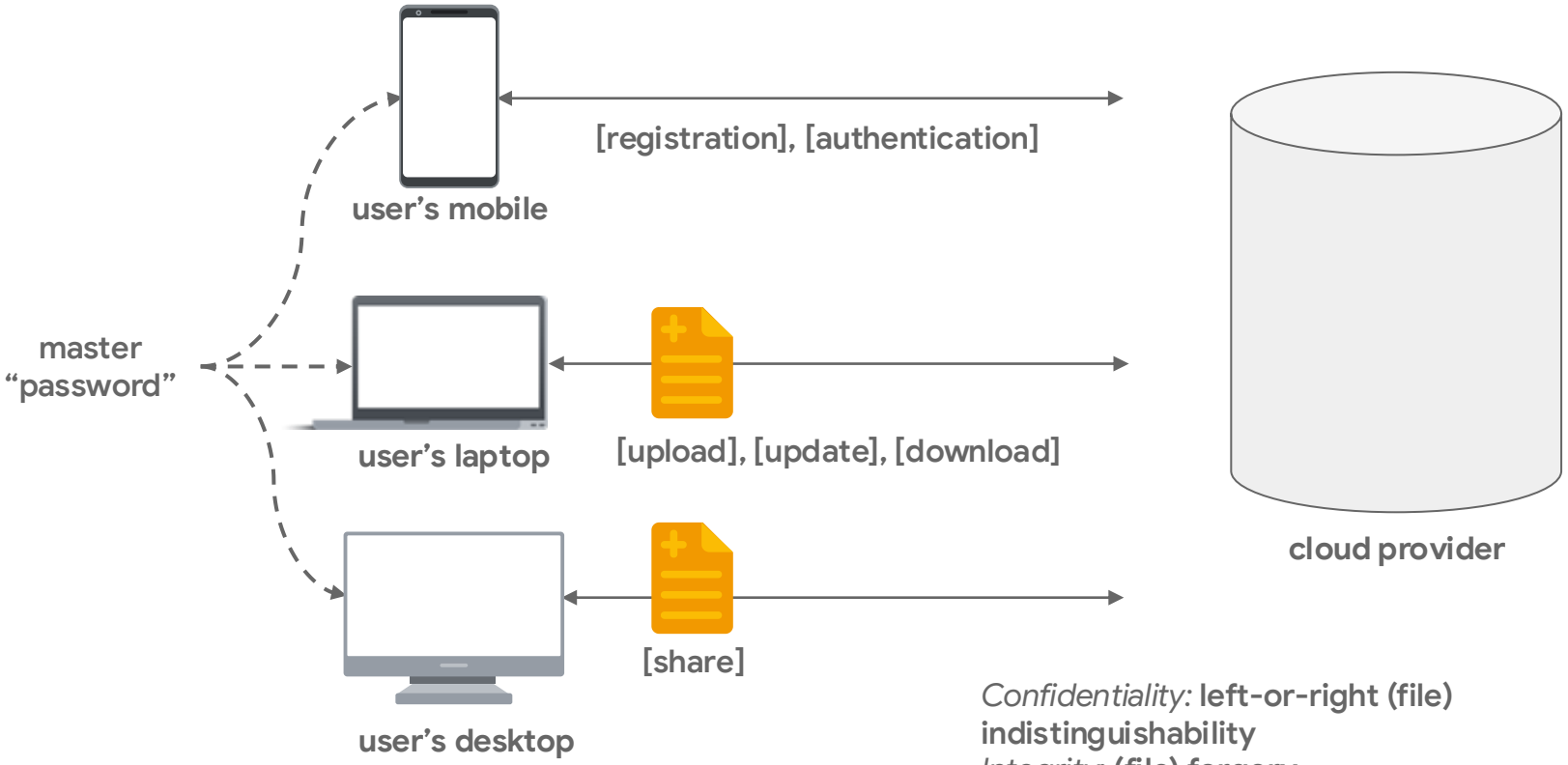
Google

Challenges in Meeting Modern User Expectations While Providing Rigorous E2EE Cloud Storage

Gordon Chu, Fernando Lobato Meeser

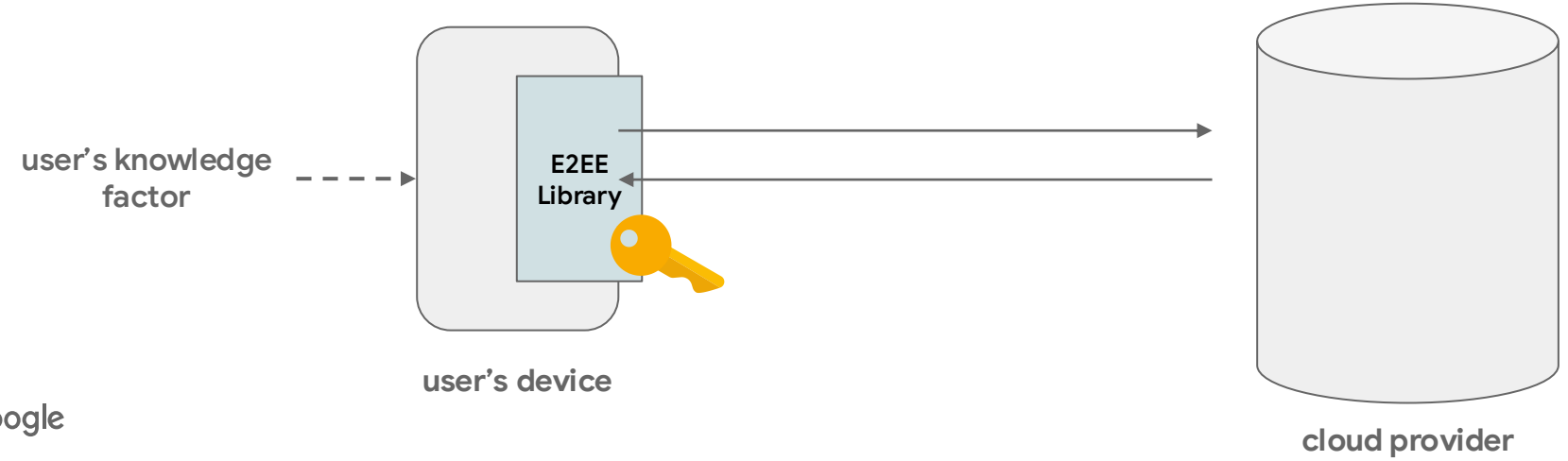
End-to-end encrypted cloud storage [BDGHP24]



Confidentiality: **left-or-right (file) indistinguishability**
Integrity: **(file) forgery**

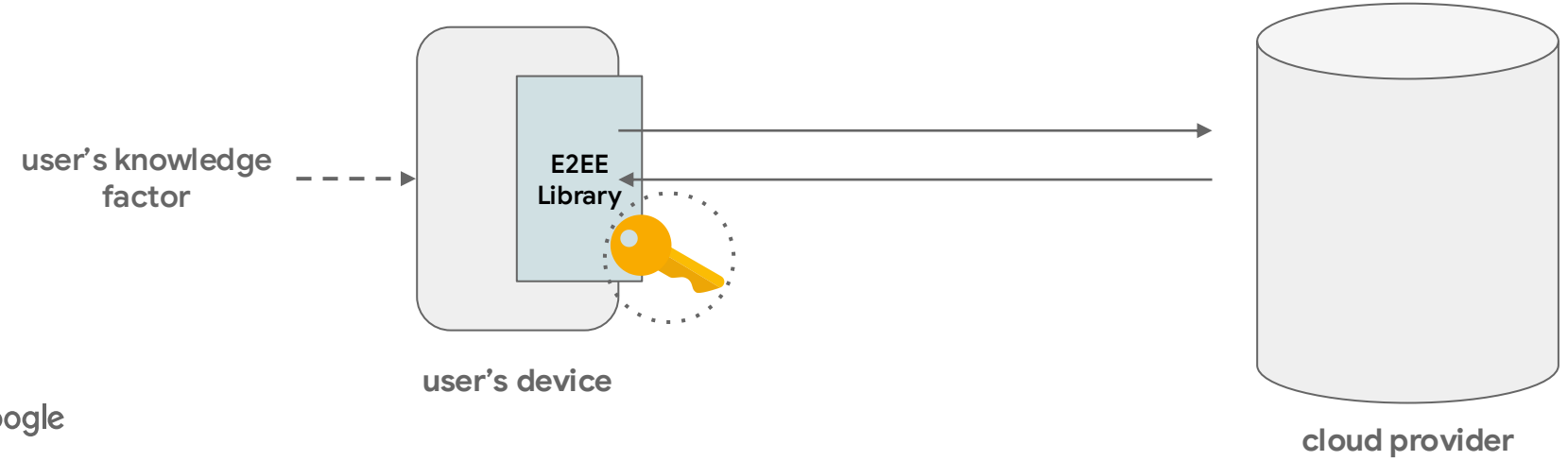
A Google application developer's view of E2EE (1/2)

```
import e2eelib

e2ee_key = e2eelib.generate_e2ee_key()
// enroll E2EE key using user's knowledge factor
e2eelib.enroll_e2ee_key(e2ee_key, user_knwldg_factor)
```

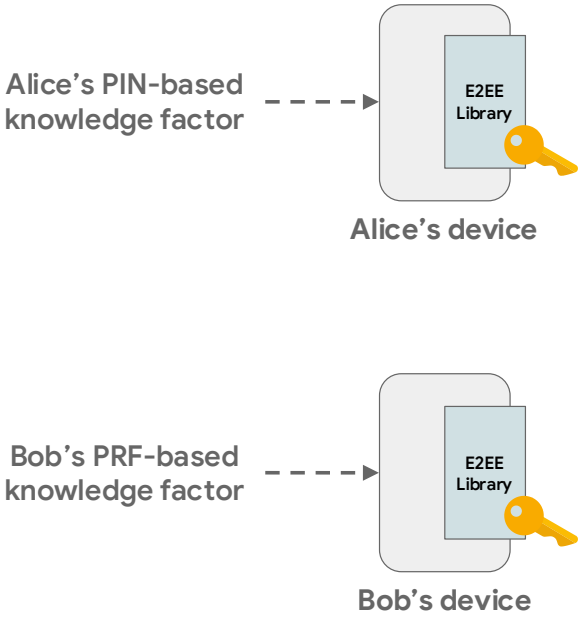


A Google application developer's view of E2EE (2/2)

```
import e2eelib

// assuming user has already registered an E2EE key
e2ee_key = e2eelib.get_e2ee_key(user_knwldg_factor)
// application-specific logic leveraging E2EE key
// e.g. use E2EE key to encrypt data
```



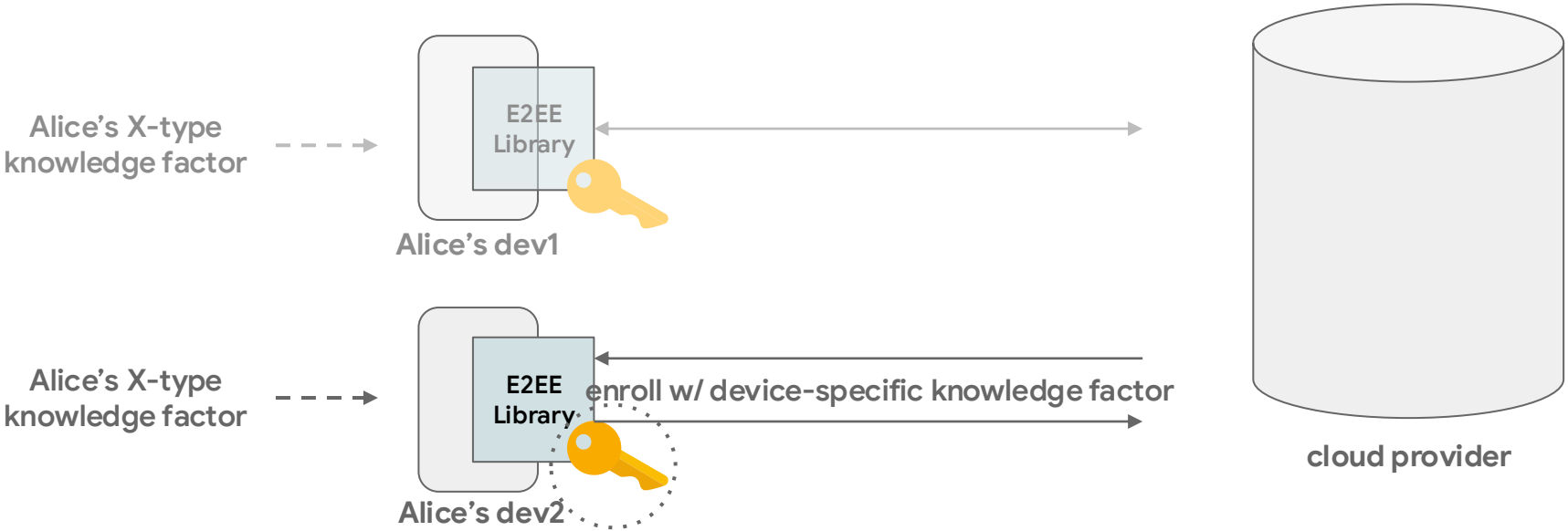
No single “type” of root password, many knowledge factors



- Low-entropy memorable factors that need bruteforce protection (e.g. verifiable rate limiting by secure enclaves); examples: passwords, PINs
- High-entropy factors stored on physical device/medium; examples: PRF on security key

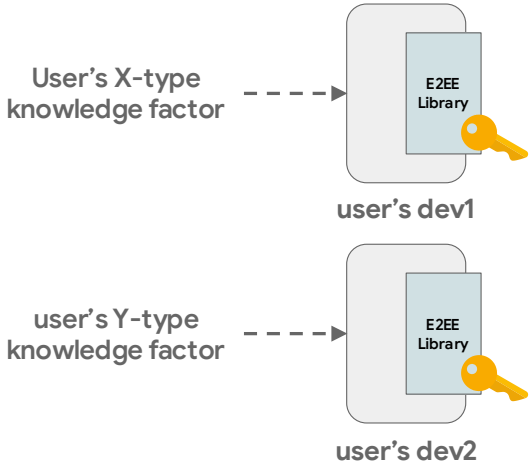
Opportunistic knowledge factor enrollment

Low-entropy *memorable* factors can be used to get E2EE key from one user device to another.



In case user forgets their portable knowledge factor, we opportunistically enroll device-specific knowledge factors when portable factor is used.

Wrinkle #1: Diff knowledge factors can wrap same E2EE key

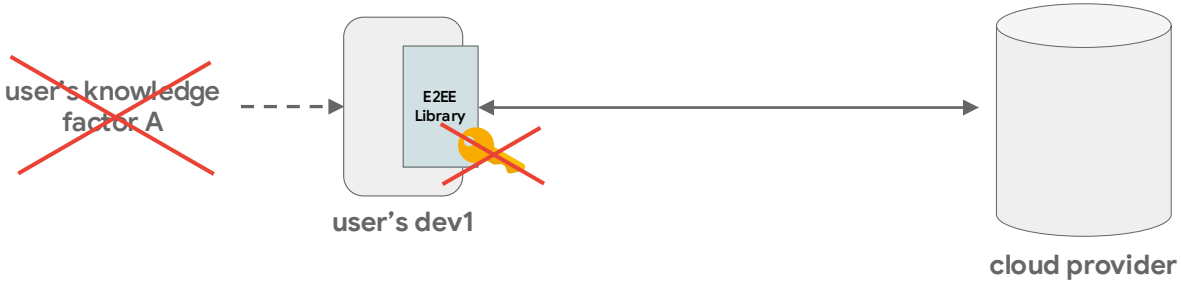


Knowledge Factors
Lock screen
PIN (Google Password Manager)
Password
WebAuthn PRF (Passkeys)

=> need some composition theorem that provides meaningful security guarantees about composition of secure E2EE Cloud Storage schemes.

Wrinkle #2: Disaster recovery

Modern users expect to be able to recover their data even when knowledge factor is forgotten or lost



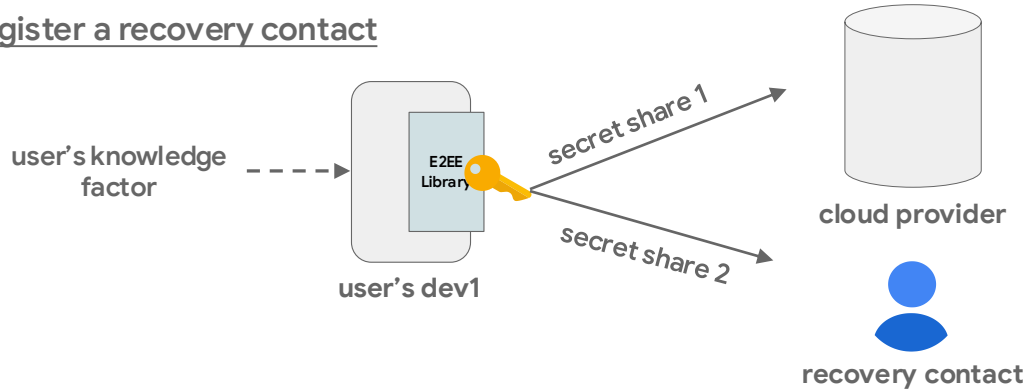
Initially, this requirement seems at odds with meeting rigorous E2EE security..

If user, with help of cooperating provider, can recover their E2EE key without their knowledge factor (or other secrets confidential from the provider), then malicious provider can just run the same protocol with themselves and recover user's E2EE key.

Wrinkle #2: Disaster recovery

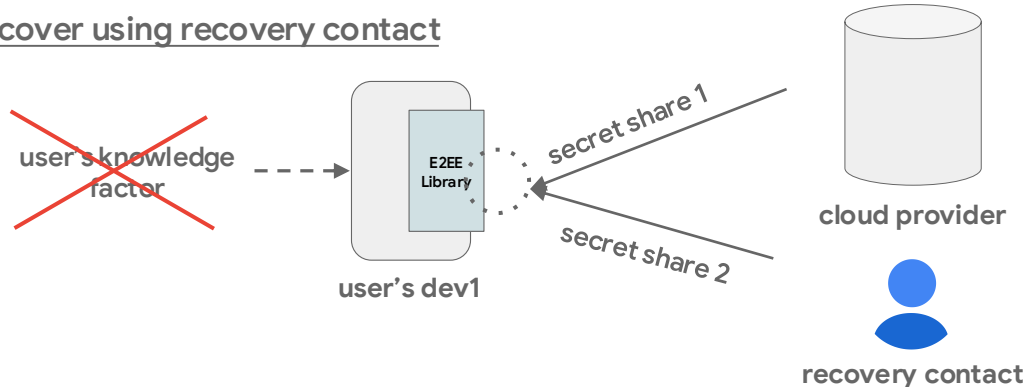
Can circumvent impossibility result by introducing additional trusted “recovery contact” party

Register a recovery contact



What should the threat model be? If plain SS scheme is used, then provider can lie about their share. Can we trust recovery contact to provide verification context if we use a verifiable SS scheme?

Recover using recovery contact



=> Need for rigorous security definitions for E2EE recovery scenarios

Google

Summary

- Industry implementation of E2EE Cloud Storage does not neatly map on to existing E2EE Cloud Storage security models in literature
- In particular,
  - Literature E2EE Cloud storage focuses on single “master password”, users safeguard *same* E2EE key with multiple knowledge factors

    => need for composability of secure E2EE cloud storage schemes

  - Users expect recoverability of E2EE data, ambiguous what E2EE secure recovery looks like

    => need for rigorous security definitions of E2EE recovery
- Hopefully, future research on E2EE Cloud Storage can address these problems