Google | Privacy, Safety & Security

Friendly KEMs

OCSW 2024
Fernando Lobato Meeser

Google

Agenda

Hybrid

Article Talk

From Wikipedia, the free encyclopedia

Hybrid may refer to:

- Hybrid Encryption ~ HPKE/ECIES
- Combined/Hybrid KEM ~ X-WING / DHKEM + P256 + KYBER

Community Feedback

(FIPS 203) ML-KEM

SP 800-227

01
Key Encapsulation Mechanism

Key Encapsulation Mechanism

- Generate() -> (**sk**, **pk**)
  - Randomized algorithm to generate private & public key pair.

- Encapsulate(**pk**) -> (**kem**, **ss**)
  - Randomized algorithm to encapsulate a shared secret **ss**.

- Decapsulate(**sk**, **kem**) -> **ss**
  - Deterministic algorithm to decapsulate the shared secret **ss**.

03

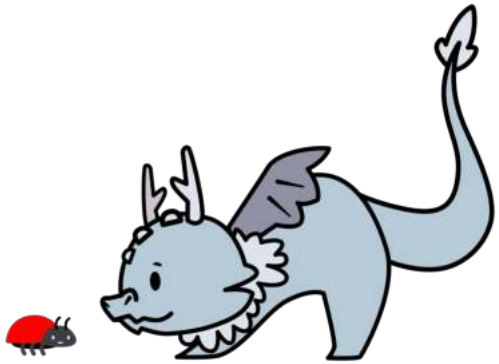
Do we want a public KEM primitive in Tink?

Case Against a KEM

- Contradiction to Tink's philosophy
  - Tink Principle: *Restrict/Limit access to secret key material.*
  - KEM: *Returns secret key material.*
- A KEM is a building block for a use case, could we provide a solution instead?

Case for a KEM

- NIST Standard = New tool (hammer)
- Hard to create comprehensive solution for every use case.
- We've seen developers use Hybrid Encryption as a KEM.
- Key visibility/auditability restrictions.
- Tink Key Derivation.



02
Tink

Multi language, multi-platform, and open source

A cryptographic library with secure APIs that are easy to use correctly, and harder(er) to misuse.

Carefully design for developers, and engineers.

Tink

- **Design goals**:
  - Secure and easy to use APIs
  - Support for key management
  - Smooth key/algorithmic rotation
  - Extensible
- **Additional Resources**:
  - RWC OSCW 2023 - Tink Mechanics
  - RWC 2019 - Introducing Tink
  - Repos: github.com/tink-crypto
  - Documentation: developers.google.com/tink

Primitive

- Abstract **cryptographic functionality**
- Defines the functionality at a high-level and its security properties
- Similar to interfaces in software development

Primitives

```
type HybridEncrypt interface {
   Encrypt(plaintext []byte, contextInfo []byte) ([]byte, error)
}
```

```
type HybridDecrypt interface {
   Decrypt(ciphertext []byte, contextInfo []byte) ([]byte, error)
}
```

Tink (Structured) Keys

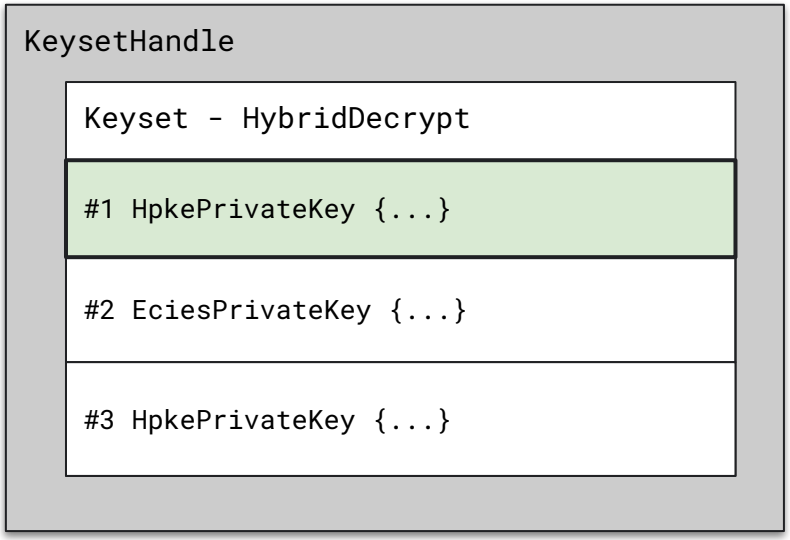
	HPKE Private	KEM: DHKEM_X25519_HKDF_SHA256 KDF: HKDF_SHA256 AEAD: CHACHA20_POLY1305	x:0x04f0… s:0x0a66…

Tink Keysets

Keyset - Hybrid Decrypt				
0x13af		HPKE Private	KEM: DHKEM_P256_HKDF_SHA256 KDF: HKDF_SHA256 AEAD: AES_GCM_256	x:0x5c24... y:0x17cb... s:0x6ba1...
0xa5c8		ECIES Private	Elliptic Curve: P-256 HKDF-HASH: SHA-256 Point Format: uncompressed AEAD Template: {AES-GCM-256}	x:0x9c34... y:0x87ab... s:0x5b41...
0x6d55		HPKE Private	KEM: DHKEM_X25519_HKDF_SHA256 KDF: HKDF_SHA256 AEAD: CHACHA20_POLY1305	x:0x04f0... s:0x0a66...

Tink Keyset Handle

- Raw bytes access require **restricted** API.
- Can only be exported encrypted unrestricted APIs.
- Can also provide Auditability
- Additional Protection (Sanitization, core dump protection)

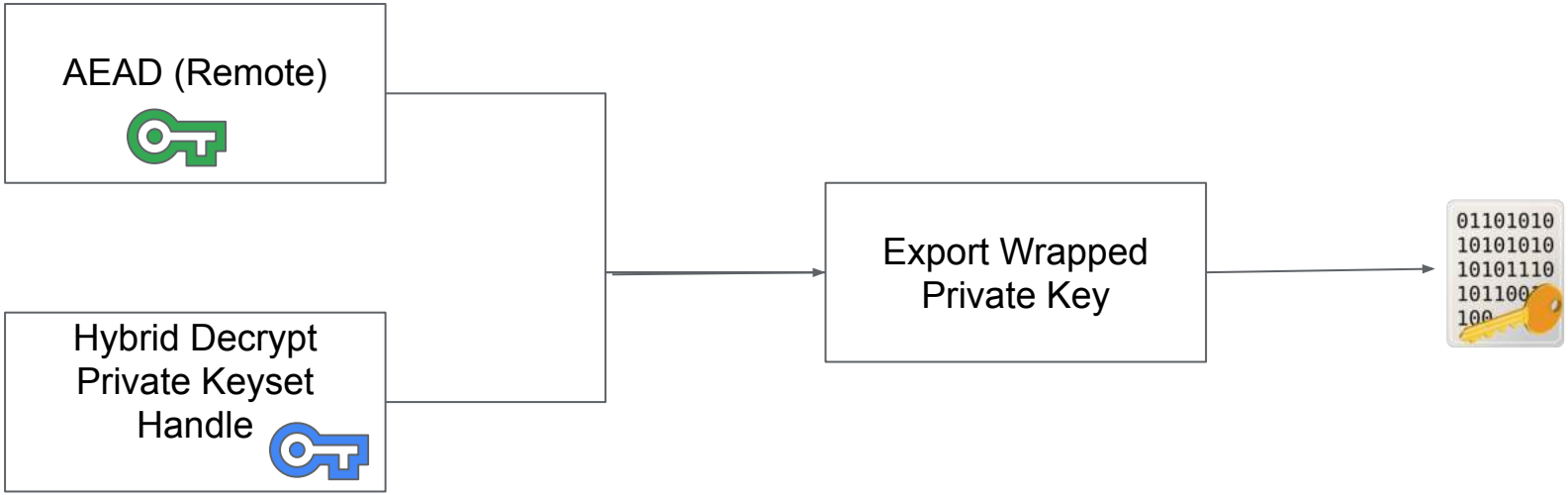


Tink Keyset Handle Raw Export

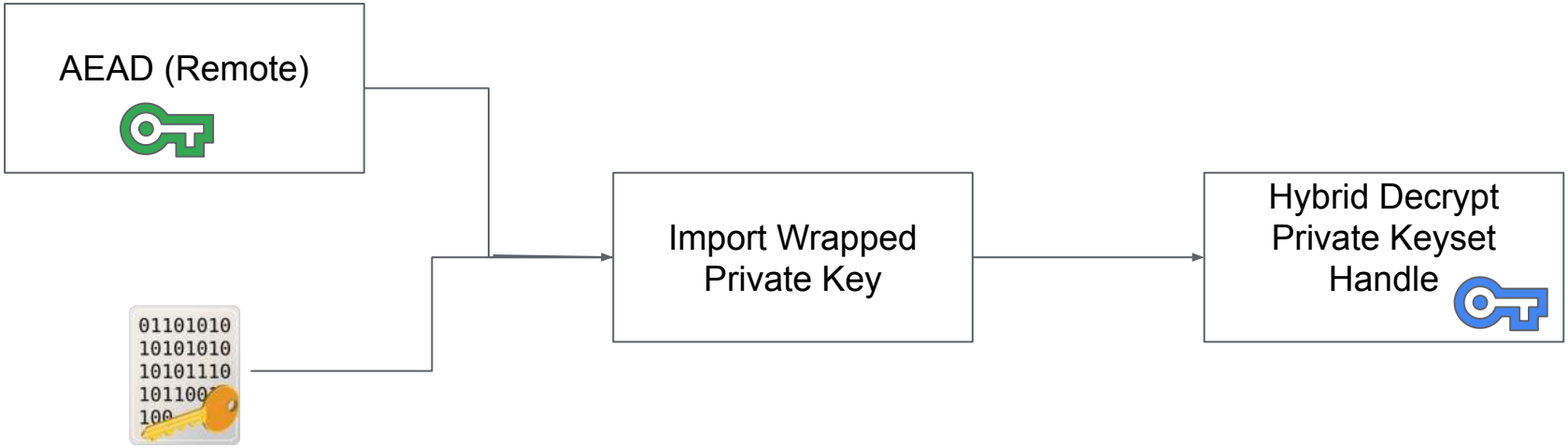
Tink Keyset Handle Raw Export

Tink Keyset Handle Raw Export - Auditing

Tink Keyset Handle Export

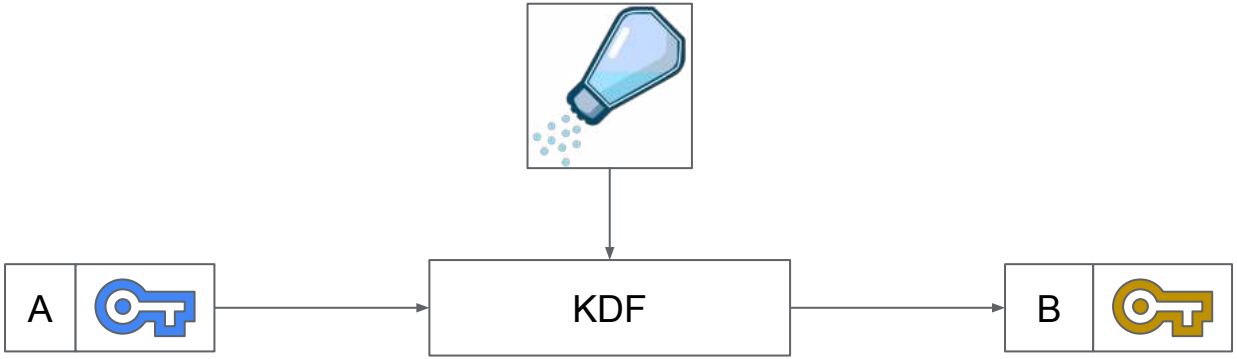


Tink Keyset Handle Import



04
Design Considerations

Keyset Deriver




```
type KeysetDeriver interface {
    DeriveKeyset(salt []byte) (*keyset.Handle, error)
}
```

Keyset Deriver

KeysetHandle

Keyset - PRF Deriver

#1 PRF:HKDF-HMAC-SHA256:{AEAD:AES-GCM}

#2
PRF:HKDF-HMAC-SHA256:{AEAD:AES-CTR-HMAC}

#3 PRF:HKDF-HMAC-SHA256:{AEAD:AES-GCM}

Derive Key

KeysetHandle

Keyset - AEAD

#1:AES-GCM { … }

#2 AES-CTR-HMAC { … }

#3 AES-GCM { … }

KEM differences/subtleties

- Randomized
- Implicit Rejection
- KEM Hybridization (algorithmic explosion)
- Security Properties

05
Tink KEM API

KEM Structured Keys

0x23fc		DHKEM -Kyber PublicKey	KEM: DHKEM_X25519_HKDF_SHA_256 + KYBER_768 Secret Usage: { StreamingAead: { AES-CTR-HMAC-HKDF } }

0x13af		X-Wing PublicKey	KEM: X-WING Secret Usage: { StreamingAead: AES-GCM-HKDF }

0xd3ca		DHKEM PublicKey	KEM: DHKEM_X25519_HKDF_SHA_256 Secret Usage: { StreamingAead: AES-GCM-HKDF }

Encapsulate Primitive Definition

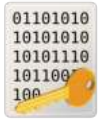
```
type KemEncapsulation struct {
  Ciphertext    []byte
  KeysetHandle  *keyset.handle
}

type KemEncapsulate interface {
   Encapsulate() (KemEncapsulation, error)
}
```

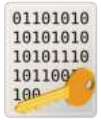
Primitive Definition

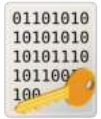
```
type KemDecapsulatation interface {
   Decapsulate(ciphertext[]byte)  (*keyset.handle,error)
}
```

Existing Bottlenecks

Hybrid Encrypt
Public Key

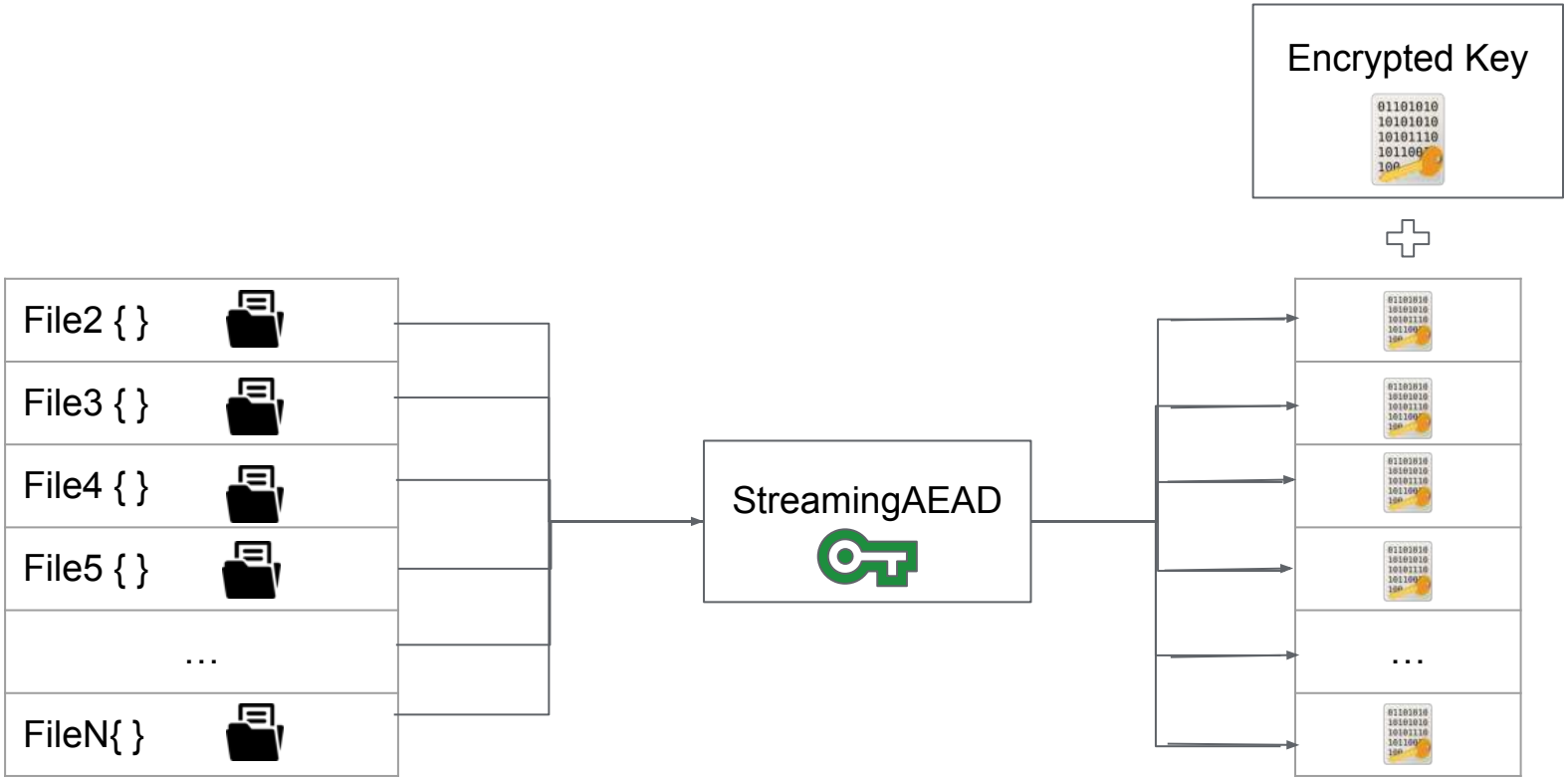




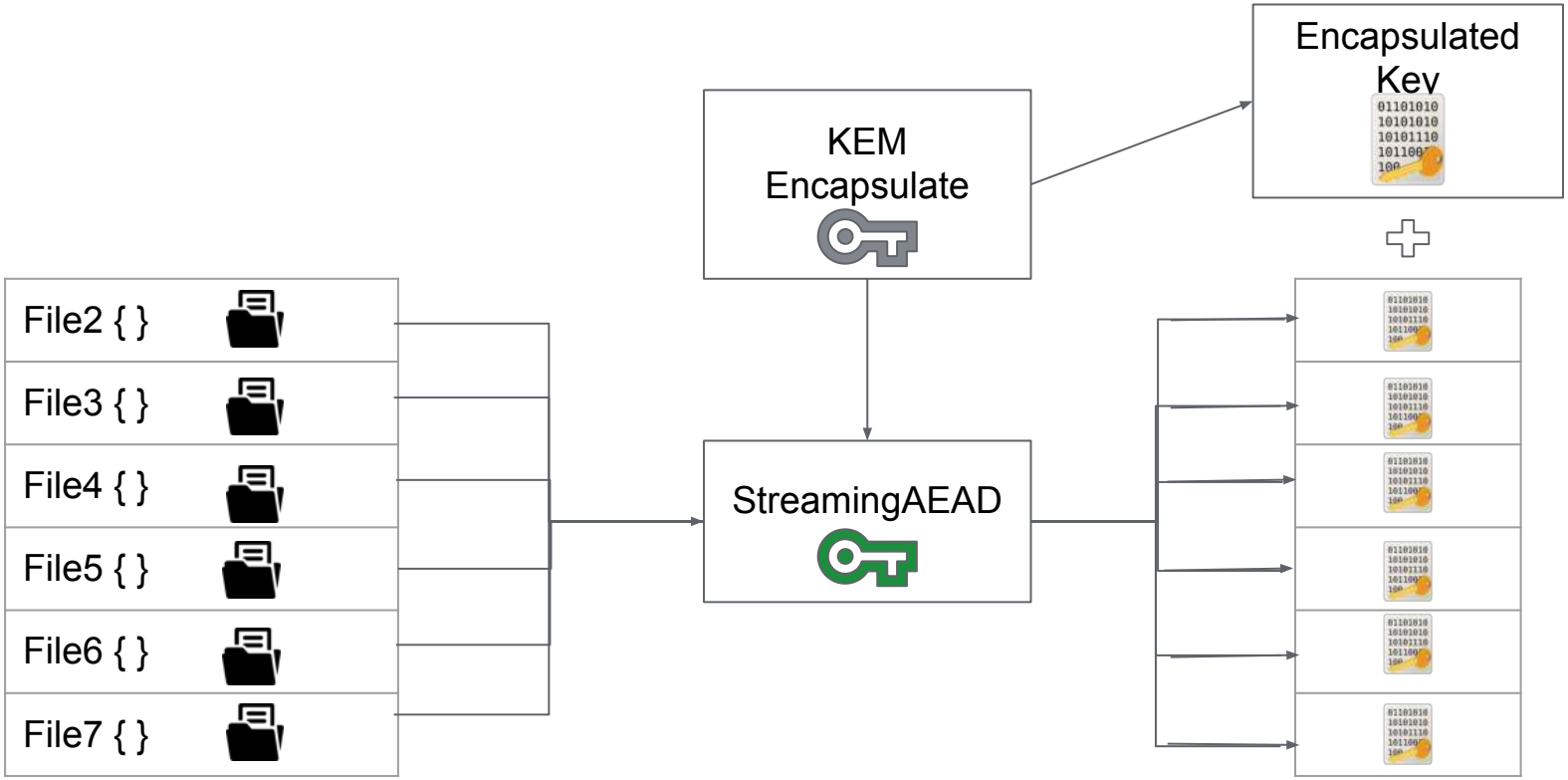


Existing Solution

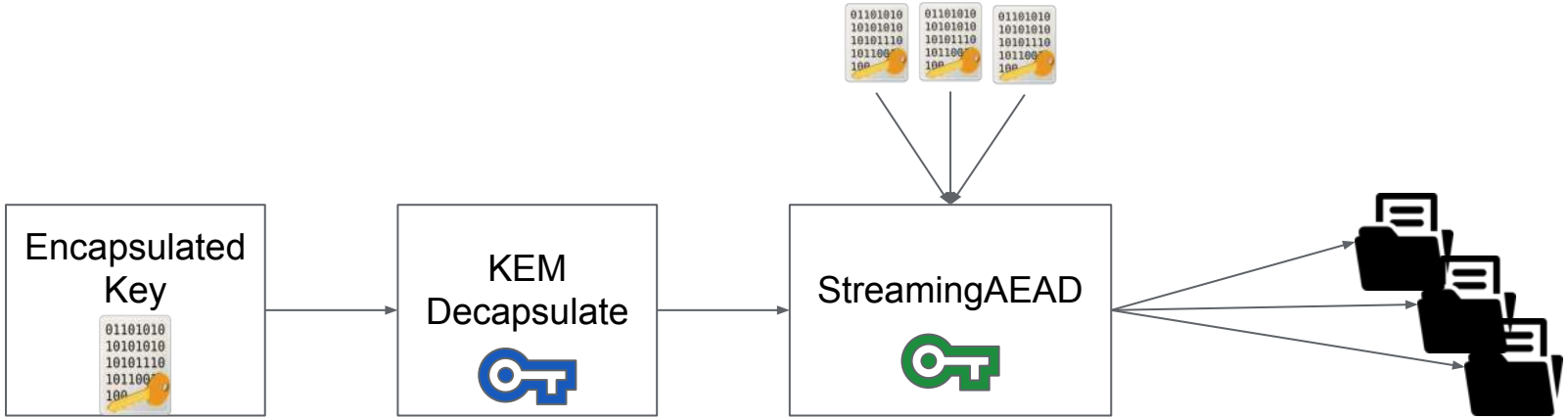
Asymmetrically Encrypt Files



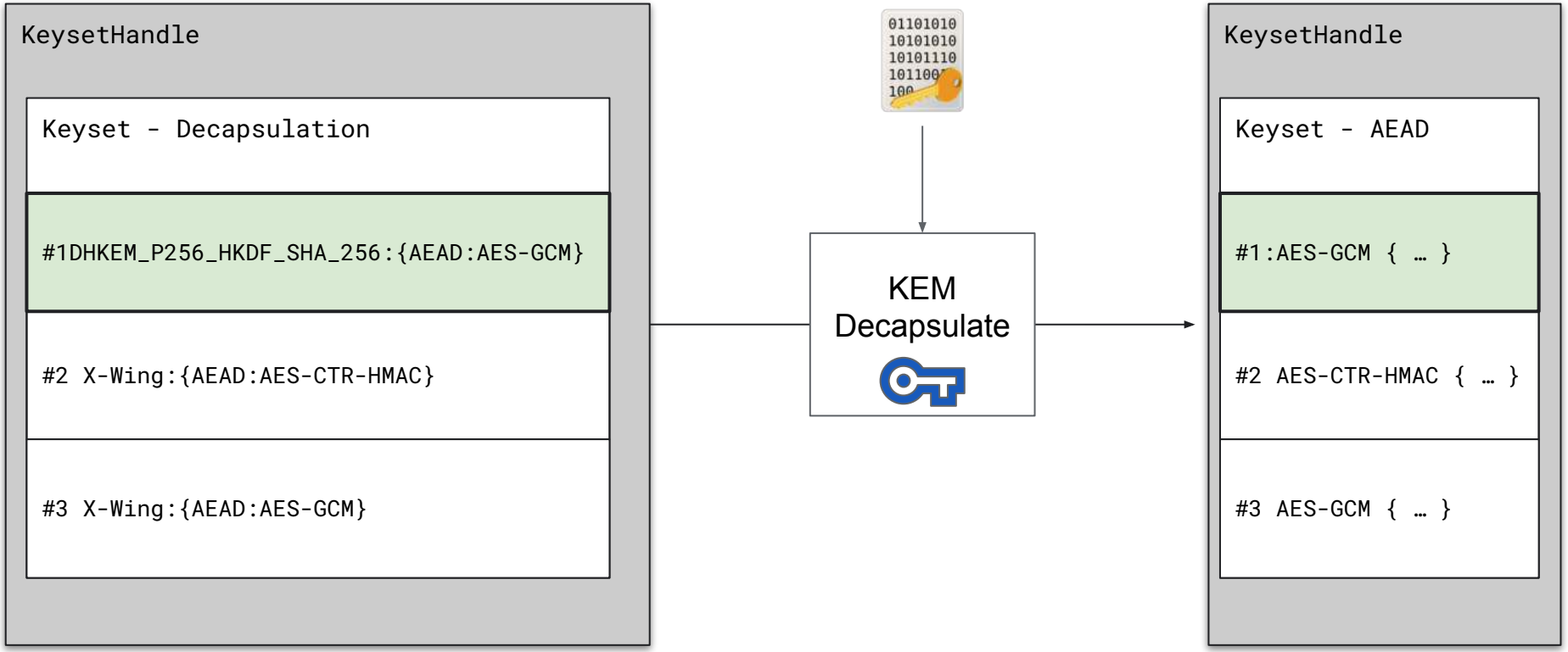
A More Composable Primitive



Decapsulation



Decapsulating Keysets



Implicit Rejection

Tink Key IDs

0xa25f		X-Wing PublicKey	KEM: X-WING Secret Usage: { StreamingAead: AES-GCM-HKDF }

KEM Encapsulate

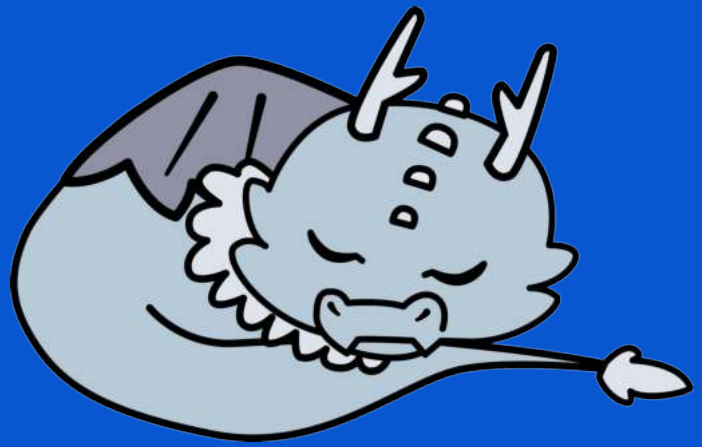
Ciphertext: 01a25f9da0eb...

Limitations

- Difference from HybridEncrypt, no stateful algorithms (we thought of an ephemeral KEM)
- Implicit rejection means we require the use of Tink IDs to avoid inconsistencies.
- You get a keyset with some useless keys in keysets.

06
Q&A

Thank you for your time!



Fernando Lobato Meeser
felobato@google.com