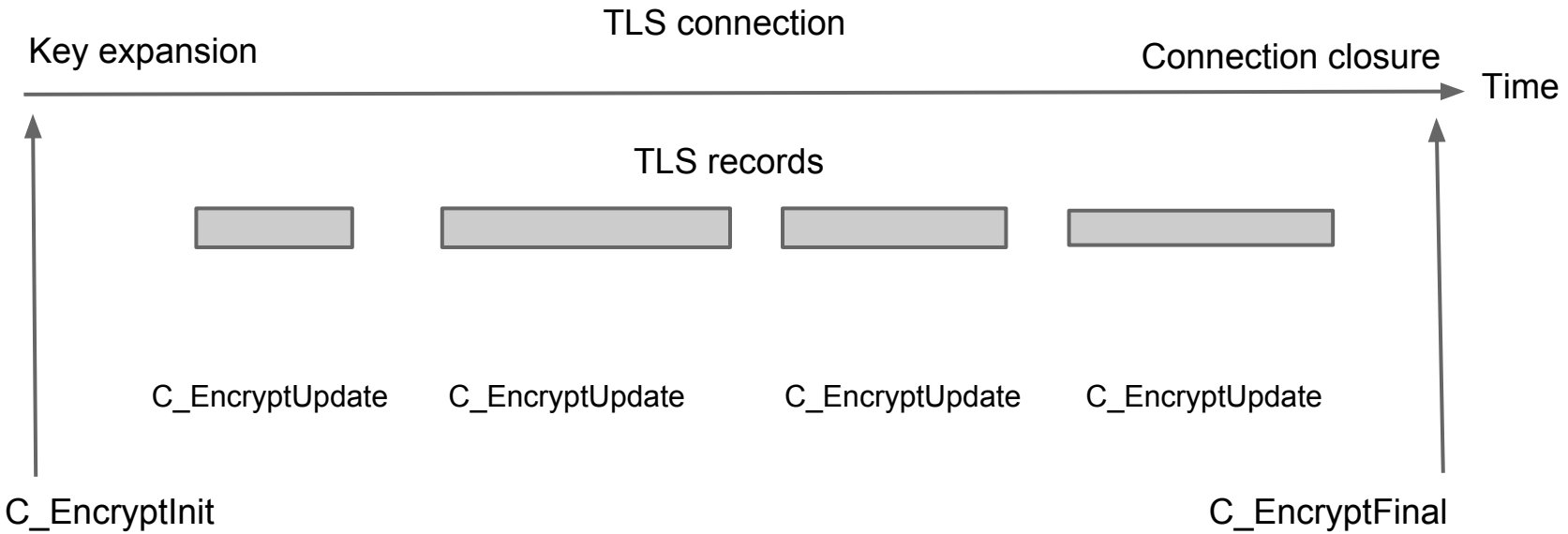
PKCS #11 AEAD functions

Wan-Teh Chang <wtc@google.com>

Outline

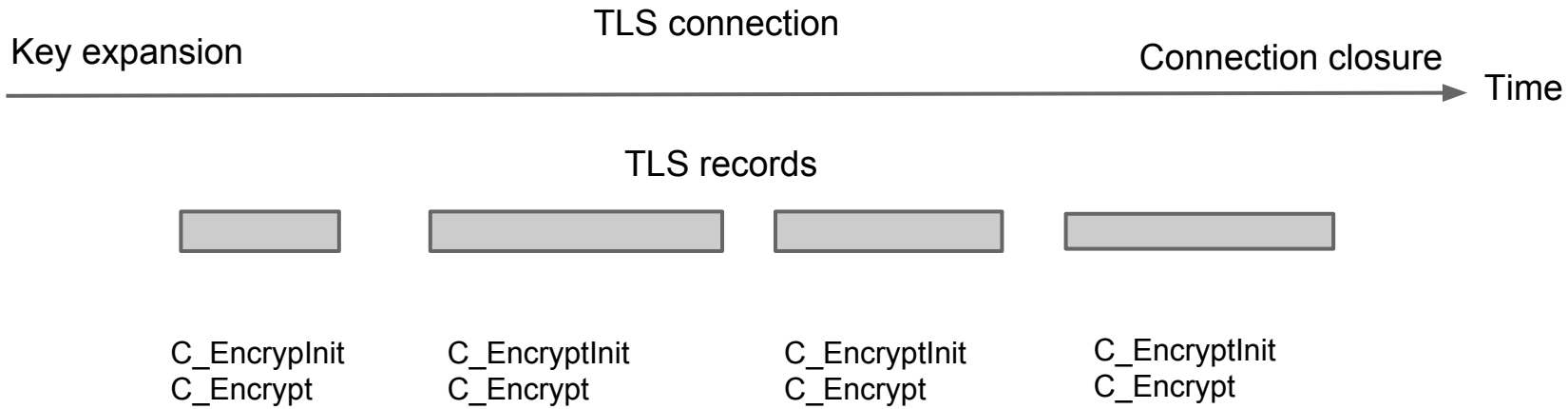
- Use case
- Problems
- Proposed solutions
- Review comments needed

Use case: encrypting TLS records



RC4 stream cipher and CBC block ciphers,
even with per-record explicit IV

Use case: encrypting TLS records



Need a **C_EncrypInit** call for each record to specify the **IV** and **AAD**, even though the **key stays the same**.

AEAD ciphers such as AES-GCM

Problem 1: C_EncryptInit overhead

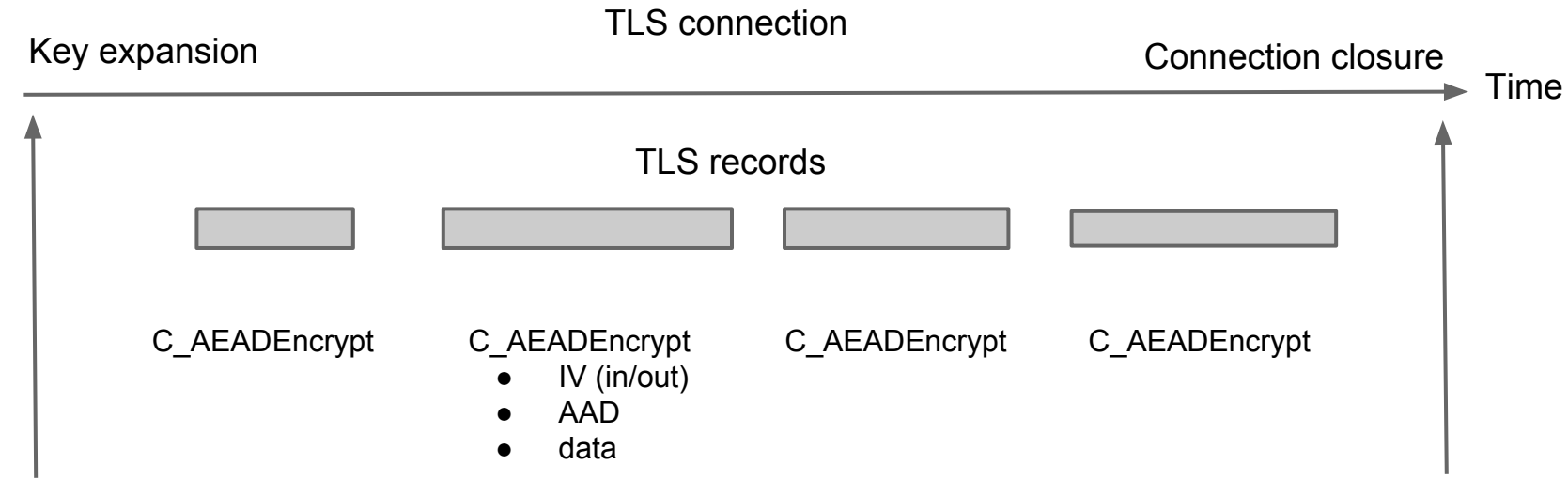
- Minimize number of PKCS #11 calls to encrypt each record

- C_EncryptInit performs IV-independent initialization repeatedly
  - AES-GCM derives the GHASH key H from the AES key K: H = AES(K, 0^128)

Problem 2: IV generation

- For CTR and GCM, IV must not be repeated
  - IV provided by the caller: hard to verify uniqueness
  - IV generated by a crypto module: IV generation can be validated
- Applies to both AEAD and non-AEAD
- Reference: draft-mcgrew-iv-gen-03

Use case: encrypting TLS records



- IV (in/out)
- AAD
- data

- AEAD mech.
- key
- IV generator mech.

C_AEADEncrypt operates in **single-part mode**.

AES GCM and nonce generator

```
#define CKM_AEAD_AES_GCM 0x00000700

typedef struct CK_AEAD_GCM_PARAMS {
  CK_ULONG ulNonceLen;
  CK_ULONG ulTagLen;
} CK_AEAD_GCM_PARAMS;


#define CKM_GCM_NONCE_DETERMINISTIC   0x00000750
#define CKM_GCM_NONCE_RBG_BASED       0x00000751

typedef struct CK_GCM_NONCE_DETERMINISTIC_PARAMS {
  CK_BYTE_PTR pFixed;  /* the fixed field */
  CK_ULONG    ulFixedLen;
} CK_GCM_NONCE_DETERMINISTIC_PARAMS;

typedef struct CK_GCM_NONCE_RBG_BASED_PARAMS {
  CK_BYTE_PTR pFree;  /* the free field */
  CK_ULONG    ulFreeLen;
} CK_GCM_NONCE_RBG_BASED_PARAMS;
```

Review

- GCM and CTR as target algorithms. Anything else?
- Other use cases?
- Are repeated calls to C_EncryptInit expensive?