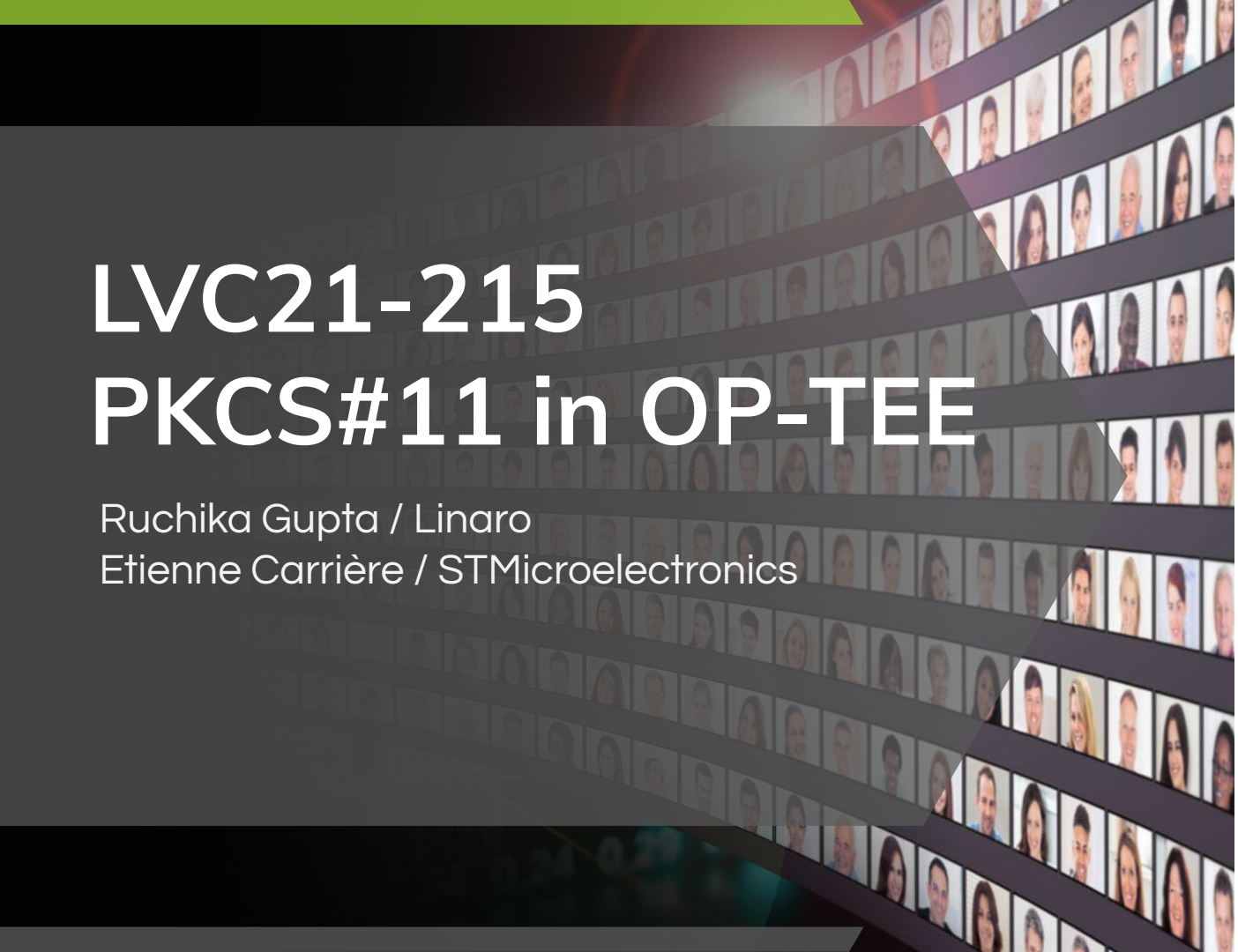


LVC21-215
PKCS#11 in OP-TEE

Ruchika Gupta / Linaro
Etienne Carrière / STMicroelectronics

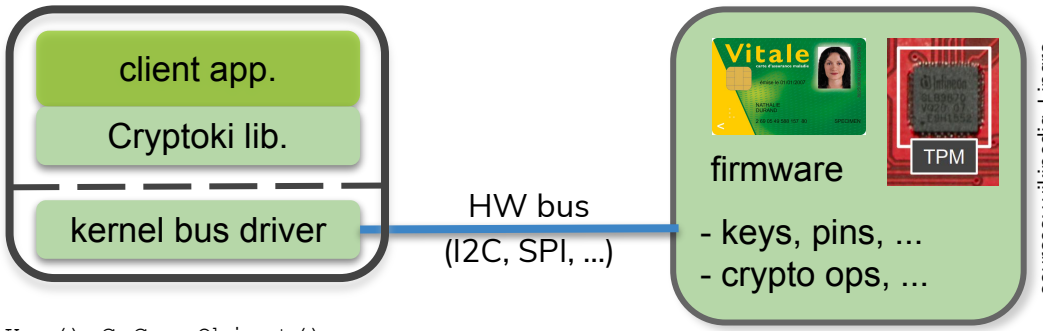
Linaro Connect
VIRTUAL • SPRING 2021

Agenda

- PKCS#11 specifications
- OP-TEE and GPD TEE specifications
- Status in 3.12.0
- libckteec
- pkcs11 TA
- Next steps

PKCS#11 specifications

- Specifications
  - Considering the 2 latest version: v2.40-errata-01 & v3.0
  - 5 documents (base, profiles, 2 mechanisms docs, user guide) + Cryptoki C/C++ header files
- Interface to manage remote cryptographic operations & objects
  - Remote objects (raw data, formated keys, certificates)
  - Remote operations using the remote objects (ciphering, authentication, key derivation, ...)
- User authentication
  - 1 SO + 1 user per token
  - PIN based authentication
  - Alternate authentication




```
C_Initialize() C_OpenSession() C_Login() …
C_CreateObject() C_GenerateKeyPair() C_DeriveKey() C_CopyObject() …
C_FindObjects() C_GetAttributes() C_SetAttributes() …
C_Encrypt() C_Decrypt() C_Sign() C_Verify() …
```

PKCS#11 as a standard

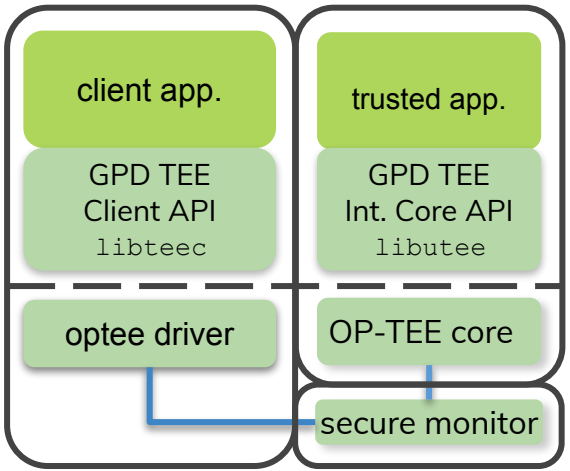
Several standard packages support interfacing PKCS#11 tokens

OpenSSH	github.com/OpenSC/OpenSC/wiki/OpenSSH-and-smart-cards-PKCS%2311 fossies.org/linux/openssh/ssh-pkcs11-helper.8
gnuTLS	www.gnutls.org/reference/gnutls-pkcs11.html
OpenSSL	No PKCS#11 engine in native OpenSSL. OpenSC proposes one (RedHat, Ubuntu, ...)
python	pypi.org/project/python-pkcs11/
AWS	Use PKCS#11 for the cryptographic operations control and alternate user authentication
LUKS	0pointer.net/blog/unlocking-luks2-volumes-with-tpm2-fido2-pkcs11-security-hardware-on-systemd-248.html
OP-TEE	:-)

Warning: specifications may have weaknesses if not flaws and should be carefully handled

OP-TEE & GPD TEE specifications

- OP-TEE is an enclave that:
  - Manages isolated trusted applications (PKCS#11 token is not the sole secure service)
  - Manages secure objects as keys for secure operations
  - Leverages platform capabilities (HW accel., ...)
  - Supports Arm7/v8 Cortex-A CPUs, RiscV in progress
  - Mostly 2 clause BSD license terms
  - https://github.com/OP-TEE
    Started in 2014, 3.x.0 series since 2018, latest is 3.12.0

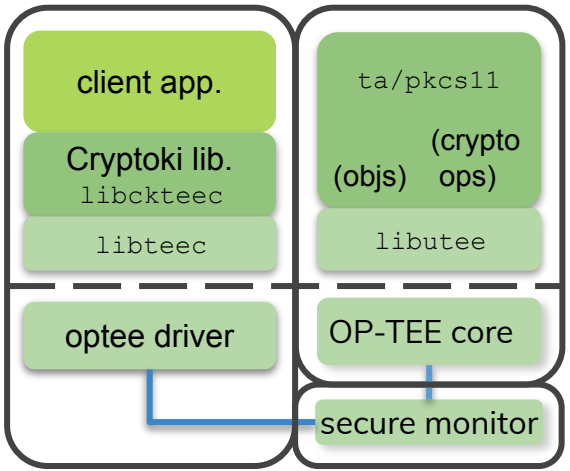


- OP-TEE is based on GPD TEE APIs
  - Does not expose object/crypto API functions to client
  - Client opens/closes sessions toward trusted app. and invokes commands (4 params)
  - In 2017 Linaro investigated on a pkcs11 TA, presented at HKG18-402

PKCS#11 Token in an OP-TEE TA

- Goal: deliver a PKCS#11 solution, reliable and maintained by the OP-TEE community

- Expose Cryptoki API to Linux user space application
  - ckteec library, 2 clause BSD license,
    hosted in optee_client.git
  - Converts Cryptoki API function call in a GPD TEE TA
    invocation command/arguments message



- PKCS#11 token is implemented in an OP-TEE TA
  - Based on GPD TEE APIs for secure storage & crypto
  - Implements PKCS#11 specification,
  - 2 clause BSD license, hosted in optee_os.git

- Regression tests: pkcs11 testsuite in OP-TEE `xtest`, hosted in optee_test.git

Achievements

- Slot and token discovery
- User session management
- User authentication (PIN & Linux ACL)
- Object (session and permanent) creation and generation (AES keys and generic secrets)
- Key derivation (by AES encryption)
- Ciphering (a bit of AES: CBC & ECB)
- MAC computation (SHA*_HMAC)
- Digest (SHA*)
- Random number generation
- Self-made pkcs11 regression tests running in OP-TEE CI

Give a try!

Download OP-TEE manifest for Qemu Arm (**default.xml**[(1)]), build and run Qemu with `host> make run CFG_PKCS11_TA=y`

From embedded shell: `sh> pkcs11-tool --list-token-slots`

(1): qemu_v8.xml: `make run CFG_PKCS11_TA=y CFG_USER_TA_TARGET_pkcs11=ta_arm64`

libckteec

- Cryptoki API in Linux userland
  - 1 Cryptoki API function for 1 pkcs11 TA command ID
  - Serialize client arguments to sent to TA (attributes lists, various structures passed)
  - Deserialize data sent back from TA (object attributes retrieved)
  - No complex processing expected : a thin API wrapper
- As of pre-3.13.0, almost all the main API functions are defined in pkcs11 TA API or will soon be.

```
C_Initialize() C_Finalize() C_GetInfo() C_GetFunctionList() C_GetSlotList() C_GetSlotInfo() C_GetTokenInfo()
C_GetMechanismList() C_GetMechanismInfo() C_InitToken() C_InitPIN() C_SetPIN() C_OpenSession() C_CloseSession()
C_CloseAllSessions() C_GetSessionInfo() CK_C_Login() C_Login() C_Logout() C_CreateObject() C_CopyObject()
C_DestroyObject() C_GetObjectSize() C_GetAttributeValue() C_SetAttributeValue() C_FindObjectsInit() C_FindObjects()
C_FindObjectsFinal() C_EncryptInit() C_Encrypt() C_EncryptUpdate() C_EncryptFinal() C_DecryptInit() C_Decrypt()
C_DecryptUpdate() C_DecryptFinal() C_DigestInit() C_Digest() C_DigestUpdate() C_DigestKey() C_DigestFinal() C_SignInit()
C_Sign() C_SignUpdate() C_SignFinal() C_VerifyInit() C_Verify() C_VerifyUpdate() C_VerifyFinal() C_GenerateKey()
C_GenerateKeyPair() C_WrapKey() C_UnwrapKey() C_DeriveKey() C_SeedRandom() C_GenerateRandom()
```

- Few are still missing, contributions are welcome:

```
C_GetOperationState() C_SetOperationState() C_SignRecoverInit() C_SignRecover() C_VerifyRecoverInit() C_VerifyRecover()
C_DigestEncryptUpdate() C_DecryptDigestUpdate() C_SignEncryptUpdate() C_DecryptVerifyUpdate()
C_GetFunctionStatus() C_CancelFunction() C_WaitForSlotEvent()
```

pkcs11 TA

- Client sessions
  - TA can implements several PKCS#11 tokens
  - Authentication based on a hash of client PIN/credentials
  - Session and object references are registered in lists in the TA space
- Objects
  - An object is a list of attributes built as a serialized byte stream
  - When object is created, attributes are added in a PKCS#11 consistent way
  - When object is used, TA checks object is visible/usable to the client for the operation
- Crypto operations
  - A token can executes a single operation at a time : the active processing.
    It must be initialized and finalized, as specified in PKCS#11.
  - Object used or generated during a operation is verified against PKCS#11 rules:
    Token state, object constraints, mechanism constraints, operation constraints, ...
  - TA relies on GPD TEE API for crypto and object storage

User authentication

- Standard default PIN based authentication (SO and user)

- Linux ACL based authentication: no PIN, only caller user and/or group IDs
  - Contributions from Vesa and Eero from Vaisala Oyj
  - Use of ACL TEE client identity: github.com/OP-TEE/optee_os/pull/4222 and related
  - Proposed configuration tool: github.com/OP-TEE/optee_client/pull/259

- Other alternate authentication can be considered
  - Coupled ACL + PIN
  - Dedicated platform means (I.e. HW under OP-TEE control)
  - ...

Testing

- A new pkcs11 testsuite is added in OP-TEE's xtest
  - For each new feature in pkcs11 TA, a xtest is implemented
  - Tests legitimate and invalid manipulations of objects/operations through Cryptoki API
  - Also tests the crypto algorithm minimal compliance (xtest test vectors and means)
  - Integrated in OP-TEE CI

    Build with `CFG_PKCS11_TA=y`

    From embedded shell: `sh> xtest -t pkcs11`
- pkcs11test
  - pkcs11test is PKCS#11 tester tool which uses Google Test.
    - Build: `sh> make CXX=/path/to/aarch64-linux-gnu-g++ AR=/path/to/aarch64-linux-gnu-ar`
    - From embedded shell: `sh> pkcs11test -m libckteec.so -s 0 --gtest_filter=<test name>`

Note: This testing is WIP. We are using these tests as we develop features for basic sanity.
You may see failures when testing.

Testing

- SoftHSM
  - Is a software based implementation of HSM
  - Has a rich unitary test library which we are building standalone and using with libckteec.so
  - Steps to compile and use it can be found at https://github.com/ruchi393/softhsm-ut-arm

- pkcs11-tool
  - Integrated in Buildroot environment
  - Get tokens/slots info

    `pkcs11-tool --show-info` `pkcs11-tool --list-token-slots` `pkcs11-tool --list-mechanisms`
  - Initialize token and user PIN

    ```
    pkcs11-tool --init-token --label test-token --so-pin 1234567890
    pkcs11-tool --label test-token --login --so-pin 1234567890 --init-pin --pin ABCDEFGHIJ
    ```
  - Generate AES Key

    ```
    pkcs11-tool --token-label foo --pin 123 --keygen --key-type AES:16 --id 1 --label
    my-key
    ```
  - List Objects

    `pkcs11-tool --token-label test-token --list-objects`

Next steps

- Wrap/unwrap keys
- RSA ciphering and authentication <--- next steps before real world application
- ECDSA & ECDH <---------- next steps before real world application
- DSA, DH
- More symmetric ciphers, MACs and KDFs
- Improve data storage
- Garbage collection of secure storage content <--- needed for long term stability
- Documentation in optee docs.
- Enhance tokens isolation and parallelization
- Test results improvements - pkcs11test and SoftHSM
- ...

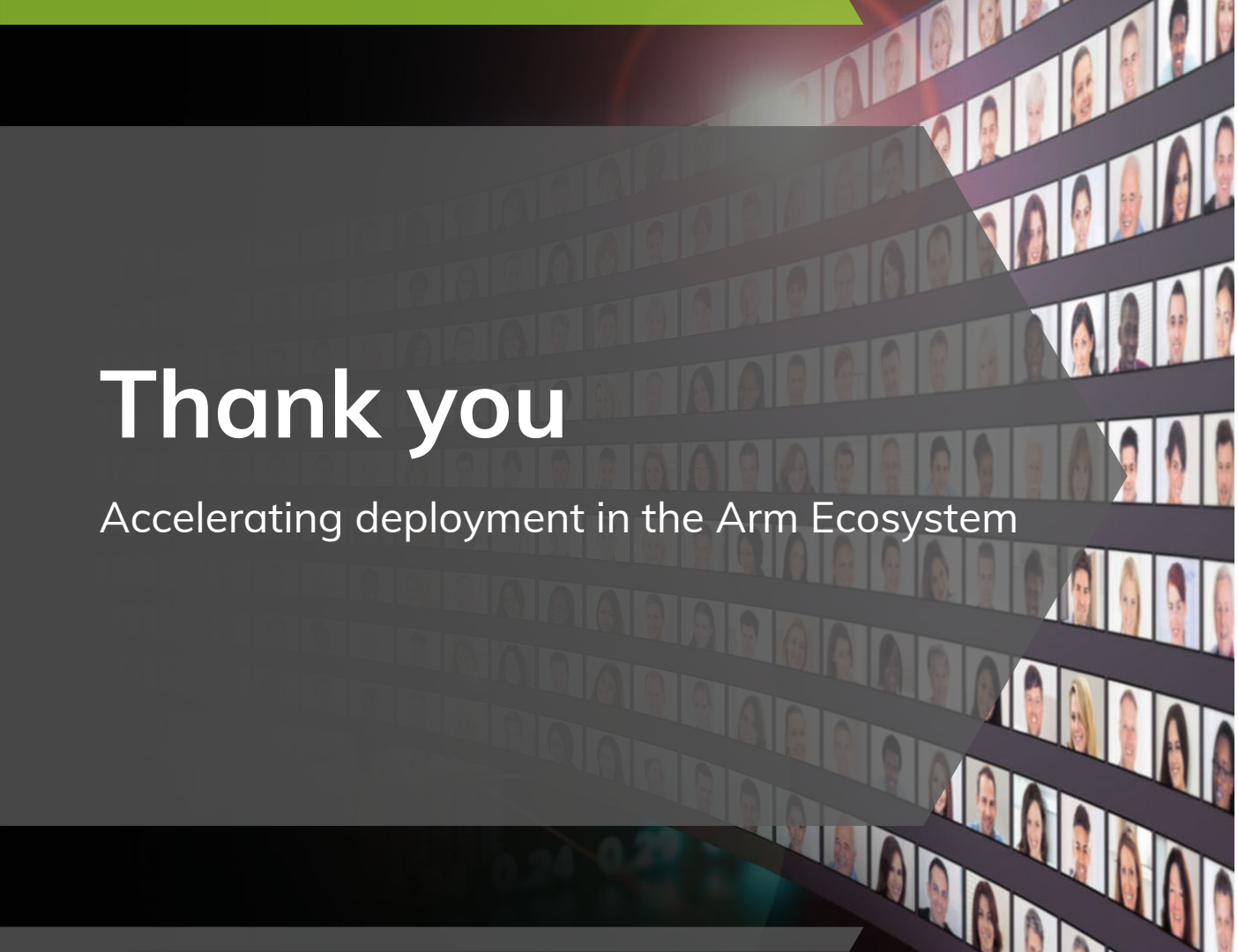
Contributions are welcome!

`ta/pkcs11`

`libckteec`

Many thanks to contributions, being code, review comments or issue reports,

Vesa Jääskeläinen, Ruchika Gupta, Rouven Czerwinski, Robin van der Gracht, Ricardo Salveti, Markus S. Wamser, Joakim Bech, Jérôme Forissier, Jens Wiklander, Gábor Székely, Etienne Carrière, Eero Aaltonen, ...



Thank you

Accelerating deployment in the Arm Ecosystem

Linaro Connect
VIRTUAL • SPRING 2021