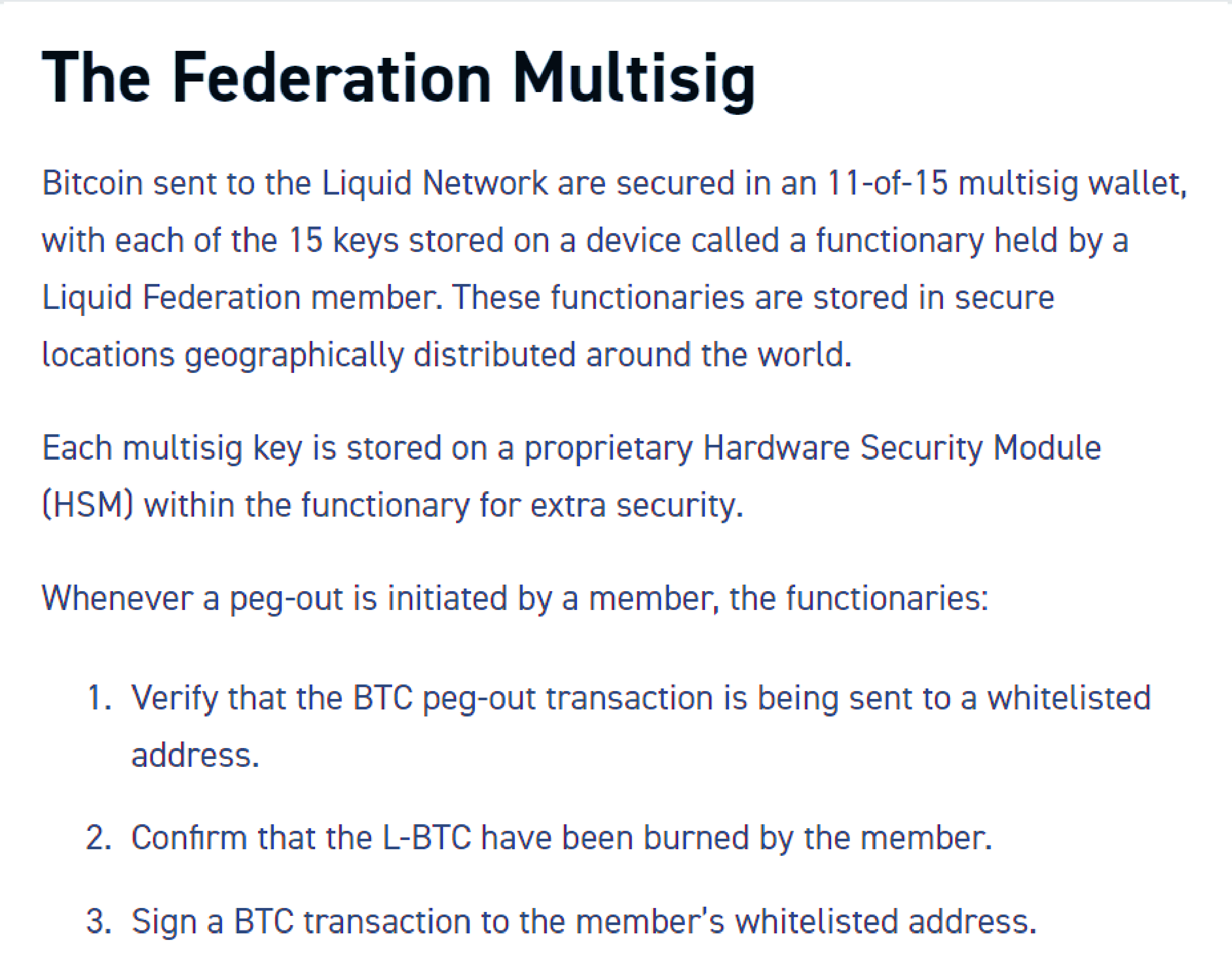
PV204 Security technologies

Hardware Security Modules (HSM), PKCS#11

Petr Švenda *svenda@fi.muni.cz* *@rngsec*
Centre for Research on Cryptography and Security, Masaryk University

Centre for Research on Cryptography and Security



The Federation Multisig

Bitcoin sent to the Liquid Network are secured in an 11-of-15 multisig wallet, with each of the 15 keys stored on a device called a functionary held by a Liquid Federation member. These functionaries are stored in secure locations geographically distributed around the world.

Each multisig key is stored on a proprietary Hardware Security Module (HSM) within the functionary for extra security.

Whenever a peg-out is initiated by a member, the functionaries:

1. Verify that the BTC peg-out transaction is being sent to a whitelisted address.
2. Confirm that the L-BTC have been burned by the member.
3. Sign a BTC transaction to the member's whitelisted address.

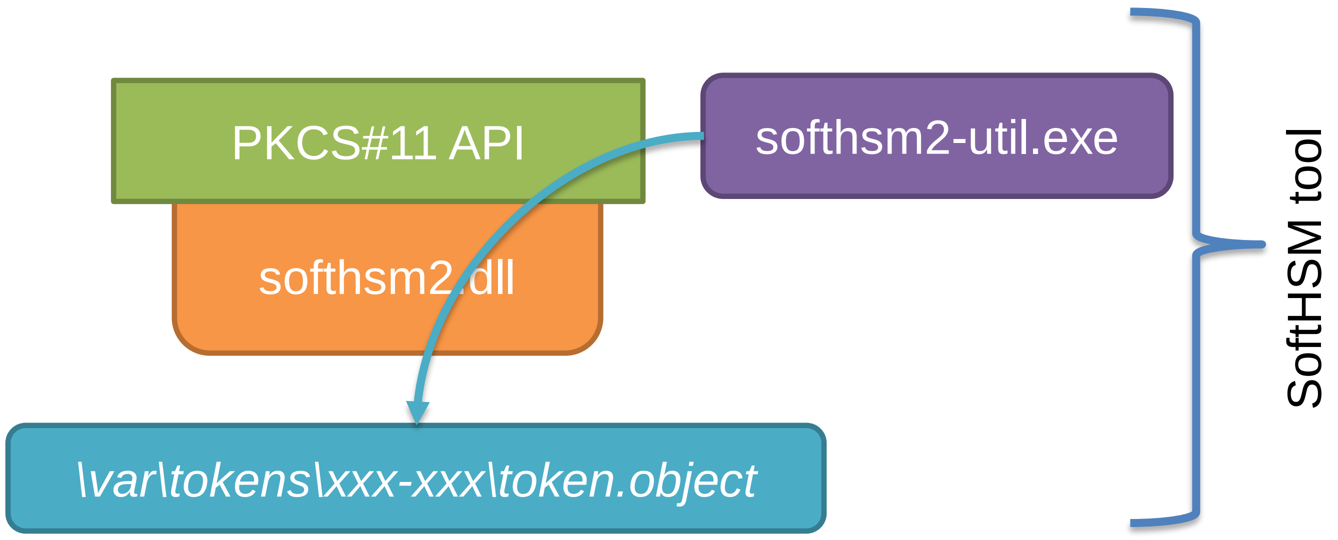
https://help.blockstream.com/hc/en-us/articles/900001408623-How-does-Liquid-Bitcoin-L-BTC-work-

Roadmap

1. Study of certification report
2. Principle of dynamically loaded libraries
3. Why PKCS#11 was introduced
4. Install and create own virtual SoftHSM token
5. Intro into PKCS#11 API (not covered at lecture)
6. Commented debug of PKCS11Example code
7. Comparison between JavaCard API and PKCS#11
8. (VeraCrypt + PKCS#11 token)

Group activity: certification report (10 minutes)

- https://csrc.nist.gov/Projects/cryptographic-module-validation-program/Validated-Modules/Search
- ‘Show all’ option, pick any hardware security module, quick read report
- What FIP140-2 level was achieved?
- What is approved cryptographic functionality?
- How is physical security protected? Side-channels?
- What kind of self-test are executed?
- Is the module also certified within Common Criteria scheme?
  - https://www.commoncriteriaportal.org/products/


PKCS#11 API
softhsm2.dll
softhsm2-util.exe
\var\tokens\xxx-xxx\token.object
SoftHSM tool

SoftHSM

Preparation in IS

- This presentation
- Code from 07_HSM_PKCS11.ZIP

DLL/SO usage

- Windows: **`LoadLibrary(),GetProcAddress(),FreeLibrary()`**
- Unix/Linux: **`dlopen(), dlsym(), dlclose()`**

```
HINSTANCE dllHandle = NULL;
if ((dllHandle = LoadLibrary(our_dll_path)) != NULL) {
    FT_C_Initialize fInitialize = NULL;
    fInitialize = (FT_C_Initialize) GetProcAddress(dllHandle, "C_Initialize");
    if (fInitialize != NULL) {
          (fInitialize)(NULL);
    }
    else status = GetLastError();
}
else status = GetLastError();
```

Prepare SoftHSM (Windows/Linux)

- Download binary for your OS (prefer version from IS)
  - https://github.com/disig/SoftHSM2-for-Windows
  - Libsofthsm http://manpages.ubuntu.com/manpages/utopic/man1/softhsm.1.html

1. Prepare user variables (Control Panel -> Edit environmental variables)
   - set SOFTHSM2_CONF h:\Apps\SoftHSM2\etc\softhsm2.conf (in user variables)
2. Set correct value to directories.tokendir inside softhsm2.conf
   - h:\Apps\SoftHSM2\var\smarthsm2\tokens

- Try to create and initialize new software token (cmd in SoftHSM2\bin\ folder)
  - softhsm2-util.exe --init-token --slot 0 --label “pv204"
- Troubleshooting:
  - Softhsm2-util crash: dll is not available
    - Check PATH, try to put softhsm2.dll into current folder
    - Still crash, check if softhsm2.dll is used (NOT softhsm2-x64.dll)
  - Error: Could not initialize library (check your system variable SOFTHSM2_CONF–name of file should be also included)
    - Check also directories.tokendir inside softhsm2.conf
  - ERROR 30: Could not initialize the token
    - wrong path to software tokens in softhsm2.conf - check

Software token(s)

```
>softhsm2-util.exe --init-token --slot 0 --label "pv204"
*** SO PIN (4-255 characters) ***
Please enter SO PIN: ******
Please reenter SO PIN: ******
*** User PIN (4-255 characters) ***
Please enter user PIN: ****
Please reenter user PIN: ****
The token has been initialized.
```

- New directory (GUID) with software token created in SoftHSM2\var\softhsm2\tokens\ folder
- Multiple tokens can be created
  - Change **--**slot 0 to **--**slot X for additional tokens
  - Otherwise token in slot 0 will be overwritten!

Management of software PKCS#11 token

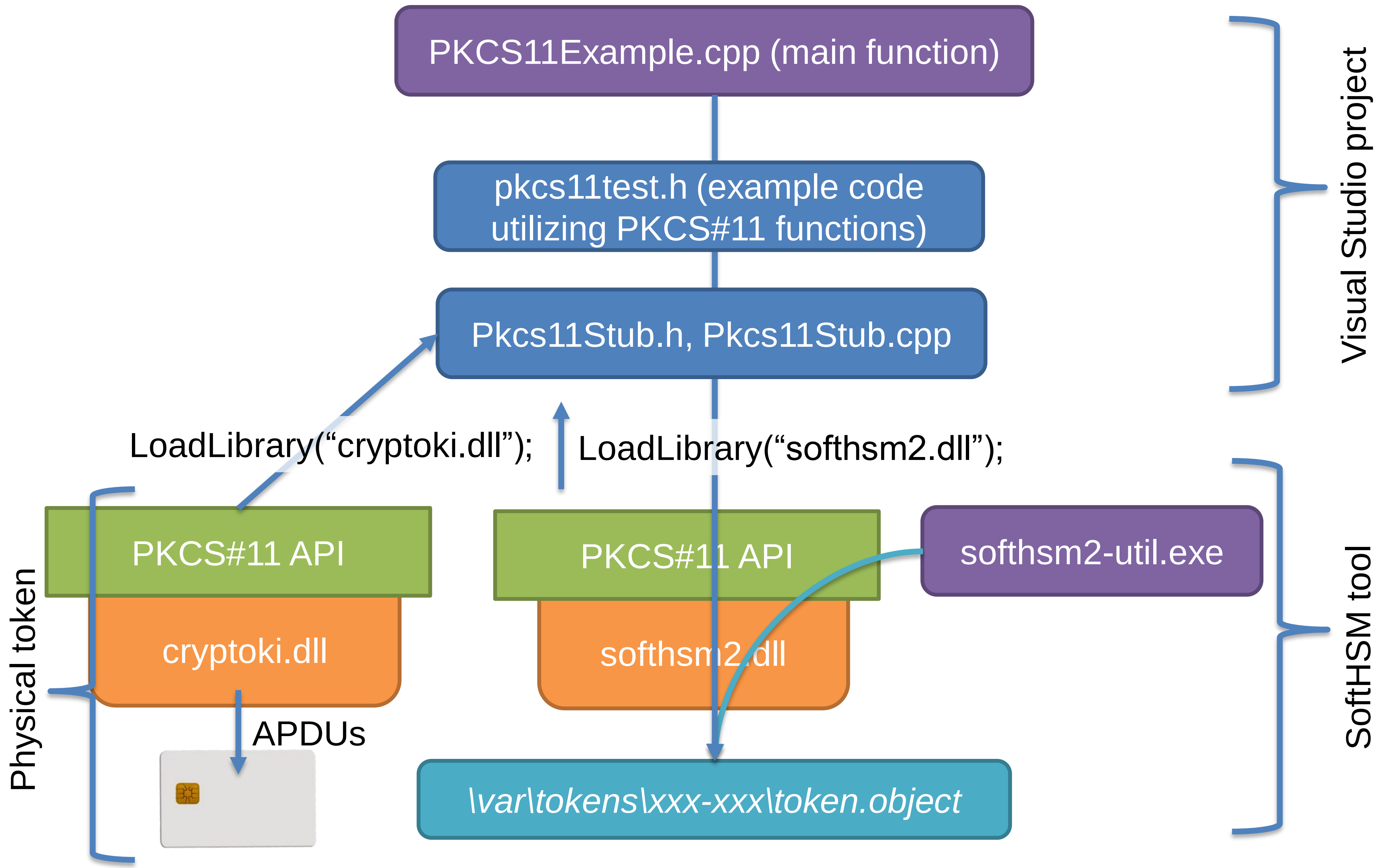
```
>softhsm2-util.exe
Support tool for PKCS#11
Usage: softhsm2-util [ACTION] [OPTIONS]
Action:
  -h              Shows this help screen.
  --help            Shows this help screen.
  --import <path>   Import a key pair from the given path.
                  The file must be in PKCS#8-format.
                  Use with --file-pin, --slot, --label, --id,
                  --no-public-key, and --pin.
  --init-token      Initialize the token at a given slot.
                  Use with --slot or --free, --label, --so-pin, and --pin.
                  WARNING: Any content in token token will be erased.
  --show-slots      Display all the available slots.
  -v              Show version info.
  --version         Show version info.
```

```
Options:
  --file-pin <PIN>  Supply a PIN if the file is encrypted.
  --force           Used to override a warning.
  --free            Initialize the first free token.
  --id <hex>        Defines the ID of the object. Hexadecimal characters.
                  Use with --force if multiple key pairs may share
                  the same ID.
  --label <text>    Defines the label of the object or the token.
  --module <path>   Use another PKCS#11 library than SoftHSM.
  --no-public-key   Do not import the public key.
  --pin <PIN>       The PIN for the normal user.
  --slot <number>   The slot where the token is located.
  --so-pin <PIN>    The PIN for the Security Officer (SO).
```

Before use of PKCS#11 – program API

- Delete all previously created software tokens
  - SoftHSM2\var\softhsm2\tokens\
- Create new token and make sure that
  - Token label is “pv204”
  - SO PIN is “123456”
  - User PIN “1234”

AT THIS MOMENT, WE HAVE AT LEAST ONE INITIALIZED TOKEN (HOPEFULLY ☺)


PKCS11Example.cpp (main function)
pkcs11test.h (example code utilizing PKCS#11 functions)
Pkcs11Stub.h, Pkcs11Stub.cpp
Visual Studio project
LoadLibrary("cryptoki.dll");
LoadLibrary("softhsm2.dll");
PKCS#11 API
cryptoki.dll
PKCS#11 API
softhsm2.dll
softhsm2-util.exe
APDUs
\var\tokens\xxx-xxx\token.object
Physical token
SoftHSM tool

PKCS#11: arguments lists

- Most of the PKCS#11 functions accept parameters as CK_ATTRIBUTE[] array
- Every value is encoded in single CK_ATTRIBUTE
  - CK_ATTRIBUTE_TYPE type
  - CK_VOID_PTR pValue
  - CK_ULONG ulValueLen

```
CK_CHAR label_public[] = {"Test1_public"};      //label of data object
CK_CHAR data_public[] = {"PV204 Public"};
CK_ATTRIBUTE dataTemplate_public[] = {
      {CKA_CLASS, &dataClass, sizeof(dataClass)},
      {CKA_TOKEN, &ptrue, sizeof(ptrue)},
      {CKA_LABEL, label_public, sizeof(label_public)},
      {CKA_VALUE, (CK_VOID_PTR) data_public, sizeof(data_public)},
      {CKA_PRIVATE, &pfalse, sizeof(pfalse)}  // is NOT private object
};
BYTE    numAttributes_public = 5;
C_CreateObject(hSession, dataTemplate_public, numAttributes_public, &hObject);
```

Use of PKCS#11 – program API

- Pre-prepared project for Visual Studio
  - PKCS11Example inside 07_HSM_PKCS11.ZIP
  - Make sure token label is “pv204”!
- Example tests of functionality in PKCS11Test
  - List available tokens (slot, token)
  - List of supported cryptographic mechanisms
  - PIN login/change (user CKU_USER, admin CKU_SO)
  - Create and find objects (public, private)
  - Generate random data on token
- Compile, run and inspect in debug mode
- Try to understand what functions are doing

Own work – during this lab

1. Write own function, which will insert private object with label "VeraCrypt secret1" into token
   - Private object => user must be logged in (C_Login)
2. Write own function, which will list all private objects on token including values
   - C_FindObjectsInit, C_FindObjects, C_FindObjectsFinal
3. Change insert function so that value of objects will be randomly data generated by token itself
   - obtained previously via C_GenerateRandom() function

Use of PKCS#11 – TrueCrypt/VeraCrypt

- Use P#11 token to increase security of VeraCrypt password
- Settings→Security tokens→Select library
  - Point to softhsm2-x64.dll
- Important: at least one private object must exist on token
  - VeraCrypt will search for private objects on token and fail with GENERIC_ERROR if not found
  - Use your private object “VeraCrypt secret1”
- Volumes→Create new volume
  - (Set standard volume info in wizard)
  - Volume Password→Use keyfiles→Keyfiles →Add token files
  - New volume should be created and PIN required on mount

No assignment this week