Project: Autonomous Parking

Using TRENTOS, CARLA, and TPM module

Selim Mert Kaştan Alisa Parashchenko Mirena Flores Valdez

13.02.2024

Overall concept: Autonomous driving

- CARLA (autonomous driving simulator) running on PC
- Python client receives environment data from CARLA and sends it to Raspberry Pi
- `TestApp` component decides how to drive, and sends it back to Python client
- Communication must happen **securely**. . .

Overall concept: Secure communication

- (Symmetrically) **encrypted** sockets, provided by our `SecureCommunication` component
- **Key exchange** using asymmetric encryption
  - `SecureCommunication` component implements algorithm
  - `WolfTPM` component implements hardware-assisted asymmetric decryption
- TPM is **authenticated** by comparing the generated key with a stored hash

Concept

- WolfTPM handles the specifics of communicating with the TPM
- But it doesn't know how to do input/output on TRENTOS
  - ⇒ Need to add custom I/O function to the HAL ("Hardware Abstraction Layer")
  - Custom I/O wrapper actually just a thin wrapper around the BCM2837 library
- It also doesn't have a CAmkES interface that would let it talk to other TRENTOS components
  - ⇒ That's what our custom interfaces are for

Custom Interfaces

- Larger data, e.g. keys or encrypted text, get passed over the dataport
- Functions only take the parameters needed to process the data
- \+ No additional buffer needed, dataport pointer can immediately be passed to WolfTPM library
- \+ Avoids unnecessary copying of data
- − Dataport doesn't show up in function signature

Custom Interfaces

`if_KeyStore`: Functions for getting or storing keys

```
void     getCEK_RSA2048(uint32_t *exp);
void     getCSRK_RSA1024(uint32_t *exp);
uint32_t storeKey(uint32_t keyLen);
int      loadKey(uint32_t hdl, uint32_t *keyLen);
```

Note:

- Nothing requires the data that's stored to actually be a key
- "Handle" actually just an offset, no validity checks

Custom Interfaces

`if_Crypto`: Functions for encryption and decryption

```
int decrypt_RSA_OAEP(int key, int *len);
int encrypt_RSA_OAEP(int key, int *len);
```

Note that this signature allows to call multiple decryption operations after each other!

```
crypto.decrypt_RSA_OAEP(IF_CRYPTO_KEY_CEK, &len);
crypto.decrypt_RSA_OAEP(IF_CRYPTO_KEY_CSRK, &len);
```

Custom Interfaces

`if_TPMctrl`: Functions to control the TPM itself

```
void shutdown(void)
```

Why is this needed?

- TPM goes into Dictionary Attack Lockdown mode if improperly shut down too often
- CAmkES appears not to have a `pre_shutdown()` equivalent of `pre_init()`

⇒ Shutdown command must be sent by `TestApp`

Initialization

- Upon component startup, `pre_init()` gets called
- Also possible to define component-specific init functions and `post_init()`, but not needed for this implementation

Our `pre_init()` function:

1. Initializes BCM2837 SPI with correct parameters (MSB first, CS1, 31.25 MHz)
2. Pulls $\overline{\text{RESET}}$ pin high
3. Generates real SRK (needed for generation of other keys and for NV storage)
4. Loads cEK into TPM…
5. Generates cSRK

Loading cEK

Problem:

- First run of the code must save cEK into TPM
- Python client also must receive the cEK, for verification
- Subsequent runs must load this cEK, not generate a new one!

Loading cEK

How to determine whether cEK already exists?

- pre_init() tries to open NV storage

If it succeeds:

- Read the cEK from NV storage into a WOLFTPM2_KEYBLOB struct
- Load the cEK into TPM main memory

Loading cEK

If it fails:

- Generate new cEK (as a `WOLFTPM2_KEYBLOB` struct)
- Store it in the NVRAM of the TPM
- Print a hexdump of the new key and exit
- Python client can import cEK using `importEK.py`

How to clear TPM and re-create cEK on purpose?

- Define `CLEAR_TPM` and re-compile
- That will clear the TPM and re-create the key

Overview

TPM Driver

Secure Communication

Simulator

Events and Notifications

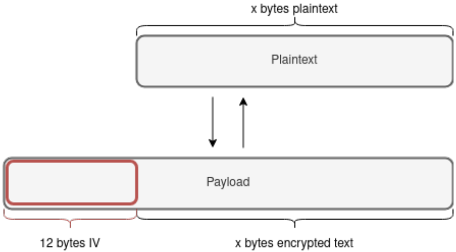
- Keeping it simple: `SecureCommunication` is a passive component
- `NetworkStack_PicoTcp` component uses `run()` routine to notify clients about new events
- **Problem:** No `run()` loop = No notifications = Calls like `OS_Socket_wait()` don't work anymore
- **Solution:** Merge functionality of `OS_Socket_wait()` with `OS_Socket_connect()`
  - In our usecase, `OS_Socket_wait()` was only called to make sure we received an event confirming the successful establishment of a connection

Key Exchange

- Key exchange must happen before sending/receiving any other data
- **Problem:** How do we synchronize `SecureCommunication` with the client component?
- **Solution:** A socket must always be created before you can read/write data on it.
  - Static variable in `SecureCommunication` keeps track of whether the key has been exchanged already.
  - Inside the code called by `OS_Socket_create()`, check the value of that variable, exchange key if necessary before creating socket and returning.

Encryption and Decryption Protocol

- Previously in the homeworks: static IV
- **Problem:** Using a repeated IV is unsafe!
- **Solution:** Append a fresh IV to the beginning of every message



Sensor Data Exchange 1

- Data exchange in real-time using TCP

- **Problem:** Nagle's algorithm
  It aims to gain efficiency by reducing the number of packets that need to be sent over the network.
- **Solution:**
  - Set socket option `TCP_NODELAY` to disable Nagle's algorithm

- **Problem:** Small sliding window
  Window size is the amount of unacknowledged data that can be in transit at any given time.
- **Solution:**
  - Buffer the sensor data before sending

Sensor Data Exchange 2

- **Problem:** Sensor data synchronisation
  The application doesn't know which radar detection was made at which location.
- **Solution:**
  - Send GNSS and Radar detections combined as pairs

- **Problem:** Split data pairs
  TCP is a byte stream protocol that doesn't preserve boundaries. `OS_Socket_read` could receive only half of a pair.
- **Solution:**
  - Add size before every pair

Park Spot Detection & Parking

- **Assumptions:**
  - Same street, Same vehicles, Parking space between two vehicles
- Compare each detection with last detection to get distance difference
- If distance is big enough, parking space is found
- **Parking:** It is a fixed procedure and consists of 4 steps
  - Stop Vehicle if parking space is found.
  - Do a right lane change
  - Drive backwards to the vehicle above parking space
  - Park vehicle