PSA Crypto API

Hannes Tschofenig

Agenda

- Motivation
- Introduction to the PSA Crypto API
- Example code
  - How to build the examples
  - Configuration options
- Symmetric encryption example
- Key export and import

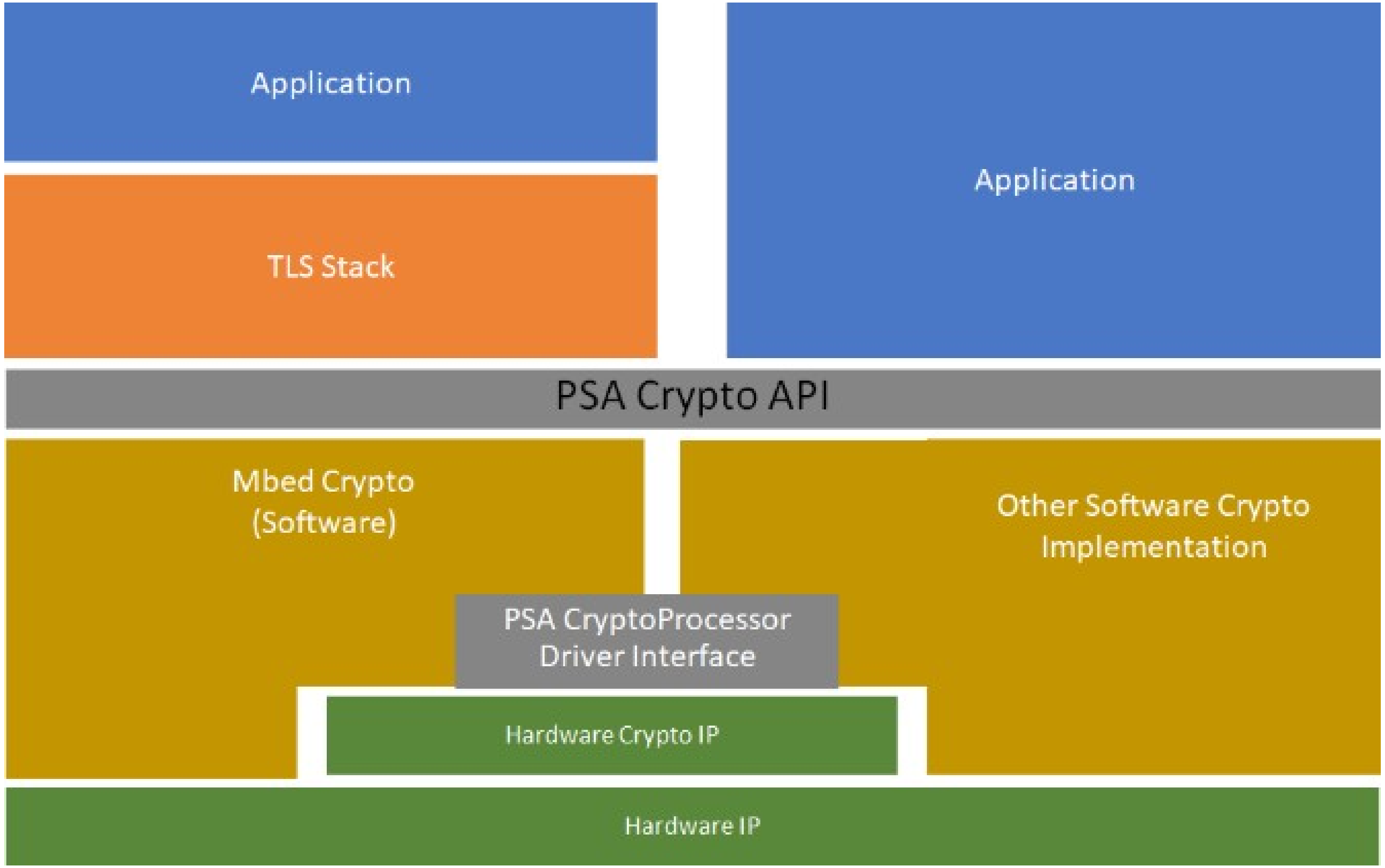
Motivation

- Hardware security mechanisms offer well-known advantages for IoT devices.
- Unfortunately, there are several challenges:
  - The list of features differs across MCUs.
  - Readily available libraries are not available.
  - Hardware security features are often poorly documented.
  - Performance improvements and reduction of power consumption is often hard to quantify.
  - Cryptographic features are often integrated at lower layers in the stack. Developers need to understand the complete stack for proper integration.

PSA Crypto API

- An attempt to define a cryptography API suitable for embedded devices.
- Covers common cryptographic primitives, use of hardware supported key storage, and deployment scenarios found in today's MCUs.
- The PSA Crypto API specification can be found at https://armmbed.github.io/mbed-crypto/html/
- A reference implementation of it can be found in Mbed TLS at https://github.com/ARMmbed/mbedtls

Software Architecture



Selected Features

Hash functions

Random number generator

Key management functions

Message authentication codes

Symmetric key encryption

Digital signatures

Key agreement

Examples

Available at https://github.com/ARMmbed/mbedtls/pull/5064

Steps to build the examples

```
git clone https://github.com/hannestschofenig/mbedtls.git
cd mbedtls/
git checkout crypto_api_examples
make generated_files
mkdir build
cd build
cmake ..
make
```

Binaries are found in programs/psa

Compile-Time Configurations

- Mbed TLS uses C-preprocessor directives to influence what functionality is included during the build progress.
- The custom configuration file is found in `include/mbedtls/mbedtls_config.h`
- PSA-related configuration is found in `include/mbedtls/config_psa.h`

Example: Symmetric Encryption

```
const uint8_t key_bytes[32] = "aaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaa";
psa_status_t status;
psa_key_attributes_t attributes = PSA_KEY_ATTRIBUTES_INIT;
psa_key_handle_t key_handle = 0;

psa_crypto_init( );

psa_set_key_usage_flags( &attributes, PSA_KEY_USAGE_ENCRYPT | PSA_KEY_USAGE_DECRYPT );
psa_set_key_algorithm( &attributes, PSA_ALG_CCM );
psa_set_key_type( &attributes, PSA_KEY_TYPE_AES );
psa_set_key_bits( &attributes, 256 );

psa_import_key( &attributes, key_bytes, sizeof( key_bytes ), &key_handle );

psa_aead_encrypt( key_handle,                       // key
                  PSA_ALG_CCM,                      // algorithm
                  nonce, nonce_length,              // nonce
                  NULL, 0,                          // additional data
                  plaintext, sizeof( plaintext ),   // plaintext
                  encrypt, sizeof( encrypt ),       // ciphertext
                  &ciphertext_length );             // length of output
 psa_destroy_key( key_handle );
```

Example:
Key Agreement (1)

Client-Side Key Generation

```
psa_key_attributes_t client_attributes = PSA_KEY_ATTRIBUTES_INIT;
psa_key_attributes_t server_attributes = PSA_KEY_ATTRIBUTES_INIT;
psa_key_handle_t client_key_handle = 0;
psa_key_handle_t server_key_handle = 0;

psa_set_key_usage_flags( &client_attributes, PSA_KEY_USAGE_DERIVE );
psa_set_key_algorithm( &client_attributes, PSA_ALG_ECDH );
psa_set_key_type( &client_attributes, PSA_KEY_TYPE_ECC_KEY_PAIR(PSA_ECC_FAMILY_SECP_R1) );
psa_set_key_bits( &client_attributes, 256 );

psa_generate_key( &client_attributes, &client_key_handle );
```

Example:
Key Agreement (2)

Server-Side Public Key Processing

```
psa_set_key_usage_flags( &server_attributes, PSA_KEY_USAGE_DERIVE | PSA_KEY_USAGE_EXPORT );
psa_set_key_algorithm( &server_attributes, PSA_ALG_ECDSA_ANY );
psa_set_key_type( &server_attributes, PSA_KEY_TYPE_ECC_PUBLIC_KEY(PSA_ECC_FAMILY_SECP_R1) );

/* ------- RECEIVING SERVER ECDHE PUBLIC KEY ------- */

status = psa_import_key( &server_attributes, server_pk, sizeof( server_pk ),
&server_key_handle );
```

Example: Key Agreement (3)

ECDHE Key Generation

```
/* Produce ECDHE derived key */
psa_raw_key_agreement( PSA_ALG_ECDH,                       // algorithm
                       client_key_handle,                  // client secret key
                       server_pk, sizeof( server_pk ),     // server public key
                       derived_key, sizeof( derived_key ), // buffer to store derived key
                       &derived_key_len );
```

Importing and Exporting Keys

- The PSA Crypto API exports and imports ECC keys in the format described in Section 2.3.3 of [*SEC 1]*.
- To make it easier for developers to utilize already existing keys in PEM or DER format an application has been written to convert keys.
- The *key_writer* app uses a public or private key as input and then converts the provided key into a given output format.
- Source code can be found at https://github.com/hannestschofenig/key_writer

[SEC1] Standards for Efficient Cryptography, SEC 1: Elliptic Curve Cryptography, May 2009. https://www.secg.org/sec1-v2.pdf

Provide Feedback

- You may have questions or feedback on the spec.

- Two options:
  - Send a private e-mail to arm.psa-feedback@arm.com
  - Post to the public mailing list: https://lists.trustedfirmware.org/pipermail/psa-crypto/