Configuring and Optimizing mbed TLS

ARM

Simon Butcher
Principal Security Engineer
mbed TLS Tech Lead

Wyboston Lakes
28th March 2017

Confidential © ARM 2017

Agenda

- Configuration for application usage
- Options for the application

Confidential © ARM 2017

Configure for the Application Usage

Confidential © ARM 2017

Typical Code Size Figures for mbed TLS 2.4.2

- Estimated code size (flash/ROM) figures for current development version of mbed TLS 2.4.2

Configuration Profile*	Code Size (approx)
mbed TLS Default	248kb
mbed OS 5.4.1 default	170kb
Thread	58kb
NIST Suite B	80kb
Pre-shared Key	28kb

* *arm-none-eabi-gcc 5.3.1 used to obtain figures with flag –Os.*

Confidential © ARM 2017

Possible Configuration modes

- Full TLS Support
  - Includes full X.509 Support
  - Includes large subset of possible ciphersuites for interoperability
  - Requires networking and entropy source
- Pre-shared Keys
  - X.509 excluded
  - Only ciphersuites required for application need to be used
  - Requires networking and entropy source
- Cryptographic primitives only
  - Only use what you need
  - Entropy source optional dependent on use cases

Confidential © ARM 2017

Things to Consider in Optimization

- Future proof - ciphersuites may need to be replaced in the future
  - Need to consider possible future RAM and ROM requirements
- End to end control – if you can control the server, you can restrict the client
- Key Management and Deployment
  - Pre-shared keys require provisioning or special manufacturing considerations

Confidential © ARM 2017

Optimize for the application

Confidential © ARM 2017

Remove RSA if you don't need it

- You can save approximately 20KB of ROM

```
#undef MBEDTLS_RSA_C
#undef MBEDTLS_KEY_EXCHANGE_ECDHE_RSA_ENABLED
```


Confidential © ARM 2017

Remove debug functionality in production

- Remove
  - Error messages
  - Debug code
  - Version features list
- Save 40KB ROM

```
#undef MBEDTLS_ERROR_STRERROR_DUMMY
#undef MBEDTLS_VERSION_FEATURES
#undef MBEDTLS_DEBUG_C
```


Confidential © ARM 2017

Reduce I/O buffer size

- Save ~20 KB of RAM
- Caveat
  - Reduces maximum fragment length
  - Make sure all handshake messages fit into the reduced fragment (Certificates!)

```
#define MBEDTLS_SSL_MAX_CONTENT_LEN 2048
```


Confidential © ARM 2017

Other tips

- Move AES tables to ROM
- Saves 8KB RAM at the expense of RAM

```
#undef MBEDTLS_AES_ROM_TABLES
```

- Constrain cipher-suite list
- Might save a small amount of ROM and RAM

```
#define MBEDTLS_SSL_CIPHERSUITES
```


Confidential © ARM 2017

And don’t forget!

- Always configure your compiler for size, if size is the problem!

- Check that your configuration of options is optimal. Remove features you don’t need
- `MBEDTLS_DEBUG_C` macros are for application configuration

Confidential © ARM 2017

Questions?

Confidential © ARM 2017