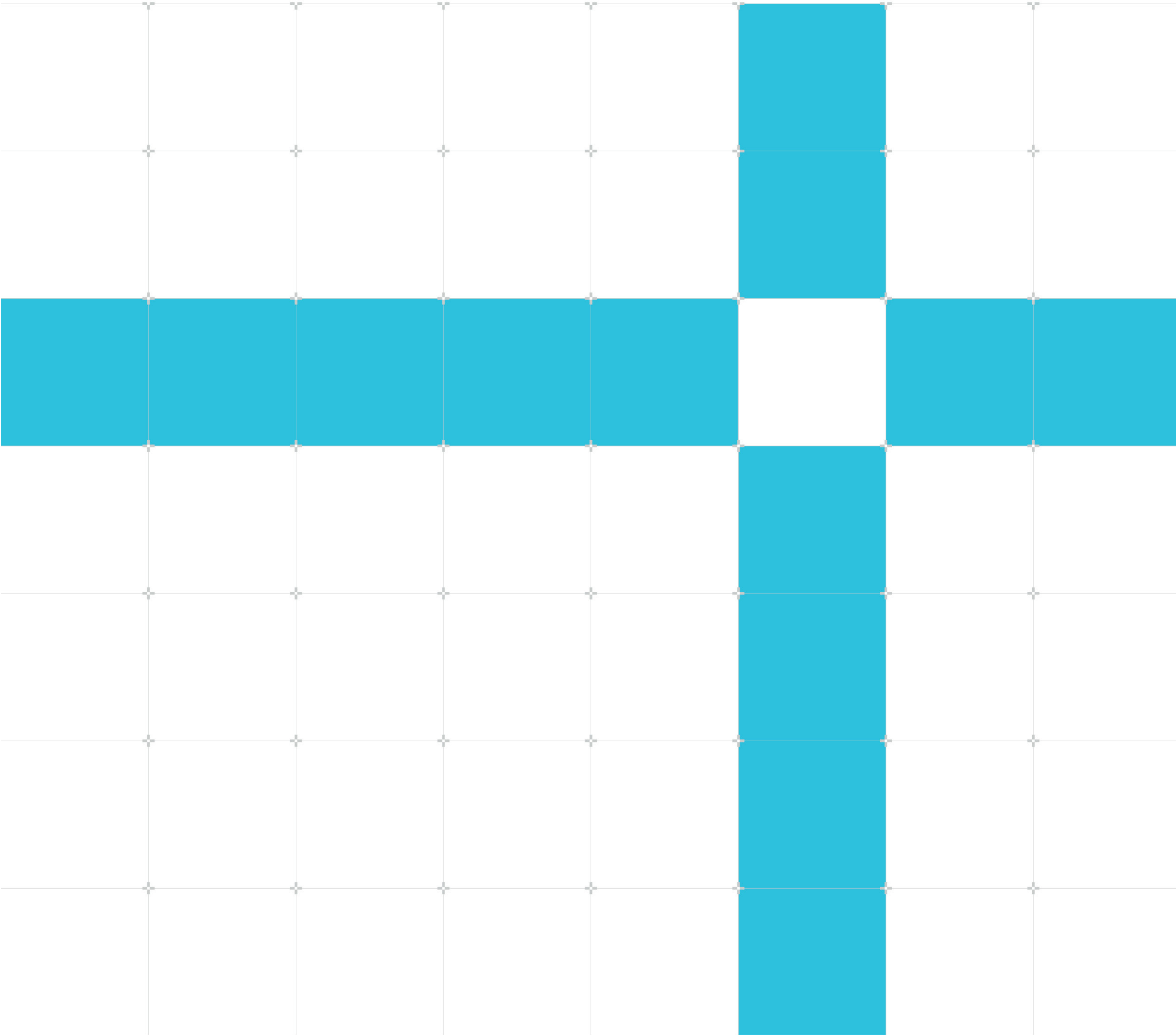
Arm® CryptoCell-312

Product revision: r1p3

OSS RT Release Note

Non-Confidential

PJDOC-1779577084-15998



Arm® CryptoCell-312

OSS RT Release Note

Non-Confidential proprietary notice

110 Fulbourn Road, Cambridge, England CB1 9NJ.

LES-PRE-20349

Confidentiality status

This document is Non-Confidential. The right to use, copy and disclose this document may be subject to license restrictions in accordance with the terms of the agreement entered into by Arm and the party that Arm delivered this document to.

Unrestricted Access is an Arm internal classification.

Product status

The information in this document is Final, that is for a developed product.

Web address

http://www.arm.com

Contents

1 Release overview

1.1 Product description

The Arm® CryptoCell-312 (CryptoCell-312) is an embedded security solution for high-efficiency systems, with emphasis on small footprint and low power-consumption. It offers platform security services, as well as a rich set of cryptographic services, targeting multiple threats.

The services CryptoCell-312 offers are needed across various IoT domains, for example, home automation, factory automation, smart energy, Industrial IoT and any other domain where there is potential usage of a Cortex®-M processor.

1.2 Release status

This is the REL release of r1p3 Arm® CryptoCell-312 runtime software.

All planned verification and validation is complete.

The release is suitable for volume production under the terms of the Agreement.

1.3 Standards compliance

This release is compliant with the following standards:

Table 1-1 Compliant standards

Doc ID	Document title	Compliance	Version
DEN 0007C-4	Arm® Trusted Base System Architecture Client1	Fully	-
DEN 0006C-1	Arm® Trusted Board Boot Requirements CLIENT	Fully	-
ANSI X9.31-1988	Public Key Cryptography Using Reversible Algorithms for the Financial Services Industry (rDSA)	Fully, excluding section C.9.	1998
ANSI X9.42-2003	Public Key Cryptography for the Financial Services Industry: Agreement of Symmetric Keys Using Discrete Logarithm Cryptography	Sections 7.1, 7.2, 7.3, 7.4, 7.5.1, 7.7.1, 7.7.2, 8.1.1, 8.1.2, 8.1.3, 8.1.4 and Annex B.	2003
ANSI X9.62-2005	Public Key Cryptography for the Financial Services Industry, The Elliptic Curve Digital Signature Algorithm (ECDSA)	Sections 7.2, 7.3, and 7.4.1 (prime curves).	2005

Doc ID	Document title	Compliance	Version
ANSI X9.63-2011	Public Key Cryptography for the Financial Services Industry - Key Agreement and Key Transport Using Elliptic Curve Cryptography	Sections 5.2, 5.3, 5.4.1, 5.6.2, 5.6.3, 5.7, 5.9, 6.1, 6.2, 6.3, 6.4, 6.5, 6.6, 6.7 and 6.8 (EC over FP).	2011
BSI AIS-31	Functionality Classes and Evaluation Methodology for True Random Number Generators	Compliant in an implementation using FETRNG driver with PTG.2.	version 3.1, September 2001
-	ChaCha, a variant of Salsa20	Fully	January 2008
Curve25519	New Diffie-Hellman Speed Records	Fully	-
Ed25519	High-Speed High-Security Signatures	Fully	-
FIPS Publication 180-4	Secure Hash Standard (SHS), compliant excluding support for truncated hash operation	Fully	-
FIPS Publication 186-4	Digital Signature Standard (DSS)	Sections 5.1, 6.2, 6.3, 6.4, B.1.2, B.2.2, B.3.6, B.4.2, C.3.1, C.3.3, C.3.5, C.9, and D.1.2.	July 2013
FIPS Publication 197	Advanced Encryption Standard, support only 128-bit and 256-bit keys	Fully	-
FIPS Publication 198-1	The Keyed-Hash Message Authentication Code (HMAC)	Fully	July 2008
IEEE 802.15.4	IEEE Standard for Local and metropolitan area networks—Part 15.4: Low-Rate Wireless Personal Area Networks (LR-WPANs)	Compliant with CCM* (section 7 and Annex B).	5 September 2011
IEEE 1363-2000	IEEE Standard for Standard Specifications for Public-Key Cryptography	Sections 7.2.1, 8 (excluding 8.2.6, 8.2.7, 8.2.8, 8.2.9), 10.3, 11, 12.2, 13 (excluding RIPEMD-160) and 14 (excluding RIPEMD-160).	2000
ISO/IEC 18033-2:2006	Information technology -- Security techniques -- Encryption algorithms -- Part 2: Asymmetric ciphers	Sections 10.2, 10.2.1, 10.2.3 and 10.2.4.	May 2006
ISO/IEC 9797-1	Message Authentication Codes (MACs) -- Part 1: Mechanisms using a block cipher	Compliant with CBC-MAC without padding, output transformation based on sections 6.2, 6.3.1, 6.4, 6.5.1, and 7.1.	-

Doc ID	Document title	Compliance	Version
NIST SP 800-22	A Statistical Test Suite for Random and Pseudorandom Number Generators for Cryptographic Applications	The second phase in the CryptoCell-312 TRNG characterization process is compliant with this.	April 2010
NIST SP 800-38A	Recommendation for Block Cipher Modes of Operation: Methods and Techniques	Sections 6.1, 6.2, 6.4, and 6.5.	-
NIST SP 800-38B	Recommendation for Block Cipher Modes of Operation: the CMAC Mode for Authentication	Fully	-
NIST SP 800-38C	Recommendation for Block Cipher Modes of Operation: the CCM Mode for Authentication and Confidentiality	Fully	July 2007
NIST SP 800-38D	Recommendation for Block Cipher Modes of Operation: Galois/Counter Mode (GCM) and GMAC	Fully	November 2007
NIST SP 800-38F	Recommendation for Block Cipher Modes of Operation: Methods for Key Wrapping,	Section 6.	November 2007
NIST SP 800-56A	Recommendation for Pair-Wise Key Establishment Schemes Using Discrete Logarithm Cryptography	Sections 5.1, 5.2, 5.3, 5.4, 5.5.1.1, 5.6.1, 5.6.2.3, 5.7.1.1, 5.7.1.2 and 5.8.2.	Revision 2, May 2013
NIST SP 800-90A	Recommendation for Random Number Generation Using Deterministic Random Bit Generators – App C.	Section 10.2 - DRBG mechanism based on block ciphers.	January 2012
NIST SP 800-90B	Recommendation for the Entropy Sources Used for Random Bit Generation.	section 4.4 tests in runtime SW.	January 2018
NIST SP 800-90C	Recommendation for Random Bit Generator (RBG) Constructions	Fully	April 2016
NIST SP 800-108	Recommendation for Key Derivation Using Pseudorandom Functions	Section 5.1.	-
NIST SP 800-135	Recommendation for Existing Application-Specific Key Derivation Functions	Fully	Revision 1, December 2011
-	The Poly1305-AES message-authentication code.	Fully	-

Doc ID	Document title	Compliance	Version
Public-Key Cryptography Standards (PKCS) #1:	RSA Encryption Standard	Backwards compatibility required by PKCS#1 Version 2.1.	Version 1.5, November 1993
Public-Key Cryptography Standards (PKCS) #1	RSA Cryptography Specifications	Fully compliant, excluding ASN.1 syntax.	Version 2.1, June 2002
Public-Key Cryptography Standards (PKCS) #3	Diffie Hellman Key Agreement Standard		
Public-Key Cryptography Standards (PKCS) #7	Cryptographic Message Syntax Standard	Section 10.3 – padding scheme.	Version 1.5, November 1993
RFC-2104	HMAC: Keyed-Hashing for Message Authentication	SHA1	February 1997
RFC-3394	Advanced Encryption Standard (AES) Key Wrap Algorithm	Fully	September 2002
RFC-5449	Advanced Encryption Standard (AES) Key Wrap with Padding Algorithm	Fully	August 2009
RFC-3566	The AES-XCBC-MAC-96 Algorithm and Its Use with IPsec	Fully compliant, excluding support for truncation to 96-bits.	-
RFC-5280	Internet X.509 Public Key Infrastructure Certificate and Certificate Revocation List (CRL) Profile	Section 4 – secure boot and secure debug certificates.	May 2008
RFC-5869	HMAC-based Extract-and-Expand Key Derivation Function (HKDF)	Fully	May 2010
RFC-7539	ChaCha20 and Poly1305 for IETF Protocols	Fully	May 2015
SEC 2	Standards for Efficient Cryptography Group (SECG) Recommended Elliptic Curve Domain Parameters	Section 2 160* domains. Smaller domains are not supported.	Version 1.0, September 20, 2000
SEC 2	Standards for Efficient Cryptography Group (SECG) Recommended Elliptic Curve Domain Parameters	Section 2.	Version 2.0, January 27, 2010

Doc ID	Document title	Compliance	Version
SEC1	Elliptic Curve Cryptography	Sections 2.1.1, 2.2.1, 3.1.1, 3.2, 3.3.1, 3.6.1, 4, and 6.1.	2000
SRP	The Secure Remote Password Protocol		1997

1.4 Conventions

The following subsections describe conventions used in Arm documents.

1.4.1 Glossary

The Arm Glossary is a list of terms used in Arm documentation, together with definitions for those terms. The Arm Glossary does not contain terms that are industry standard unless the Arm meaning differs from the generally accepted meaning.

See the Arm® Glossary for more information.

1.4.2 Typographical conventions

Convention	Use
italic	Introduces special terminology, denotes cross-references, and citations.
bold	Highlights interface elements, such as menu names. Denotes signal names. Also used for terms in descriptive lists, where appropriate.
`monospace`	Denotes text that you can enter at the keyboard, such as commands, file and program names, and source code.
`Monospace bold`	Denotes language keywords when used outside example code.
`monospace italic`	Denotes arguments to monospace text where the argument is to be replaced by a specific value.
`monospace underline`	Denotes a permitted abbreviation for a command or option. You can enter the underlined text instead of the full command or option name.
<and>	Encloses replaceable terms for assembler syntax where they appear in code or code fragments. For example: `MRC p15, 0, <Rd>, <CRn>, <CRm>, <Opcode_2>`
SMALL CAPITALS	Used in body text for a few terms that have specific technical meanings, that are defined in the Arm® Glossary. For example, IMPLEMENTATION DEFINED, IMPLEMENTATION SPECIFIC, UNKNOWN, and UNPREDICTABLE.
⚠	Caution
✋	Warning
📝	Note

2 Release contents

The following sub-sections detail:

- The component parts are delivered as part of this release.
- Any changes since the previous release.
- Any known issues and limitations exist at the time of this release.

2.1 Deliverables

Arm® CryptoCell-312 OSS includes the following deliverables:

- CryptoCell-312 runtime software.
- CryptoCell-312 runtime software integration tests.
- CryptoCell-312 runtime tools.
- Runtime API documentation: *Arm® CryptoCell-312 Runtime Software Developers Manual*.

Documentation may change between product releases. For the latest documentation, please check the delivery platform.

2.1.1 Associated products

The following parts are available to licensees only:

- Arm® CryptoCell-312 Boot Services
- Arm® CryptoCell-312 Hardware

2.2 Differences from previous release

This is the first release of CryptoCell-312 runtime software OSS.

2.3 Known limitations

Any issues known at the time of this release are detailed in the following sub-sections.

2.3.1 Missing functionality

- RSA 4K key generation is not supported.
- The PKCS #1 v2.1 standard recommends not using MD5 hash. Therefore, CryptoCell-312 does not support it. Accordingly, the Mbed TLS `MD_NONE` value is not supported.

2.3.2 Open technical issues

The following table details any technical issues that are open at the time of this release.

Table 2-1: Defects in this release

ID	Title	Description	Workaround
RN-001-CC110-R1P3-00REL	Mbed TLS compilation issue	When compiling Mbed TLS while using the flag `MBEDTLS_ECDSA_VERIFY_ALT=1`, a warning appears. This is a known issue in Mbed TLS.	None

The ID is for reference only.

3 Get started

This section details any information to help you get started with accessing, setting up, and using CryptoCell-312 runtime software.

3.1 Licensing information

The Arm® CryptoCell-312 runtime library and integration tests are published under two optional licenses, located at the root of the project tree:

- BSD-3 clause - Full license is disclosed in `BSD-3-Clause.txt`.
- Arm non-OSI - Full license is disclosed in `Arm-proprietary-license.txt`.

3.2 Download the product

Arm delivers the files through github.

You can download the product package in one of the following ways:

- Download a `.zip` file directly from https://github.com/ARM-software/cryptocell-312-runtime
- Use one of the following git clone commands:

  The target directory is only mentioned to align with the compilation commands listed afterwards.

  - `git clone https://github.com/ARM-software/cryptocell-312-runtime.git cryptocell-rt`
  - `git clone git@github.com:ARM-software/cryptocell-312-runtime.git cryptocell-rt`

You can download the product package as a single zip file: `cryptocell-312-runtime-master.zip`.

3.2.1 Unpack the product

If you downloaded a .zip file directly from github, perform the following steps to unpack the product package:

1. Relocate the package file:

   Copy the `.zip` files to the directory where these files are to be installed.

2. Unzip the package.

   This command extracts the package into a directory with the same name as the package name.

3.2.2 Compile the product

The optimization level is O2.

The following steps describe how to compile each constituent part of this product.

1. Compile the runtime library and the utilities:

   This process assumes that the runtime software is downloaded and extracted or cloned to a working directory named `cryptocell-rt`.

```
% export ARM_CPU=<cpu-type>
% export COMPILER_TYPE=<compile-type>
% cd cryptocell-rt
% ./prepare_mbedtls.sh clone
% ./prepare_mbedtls.sh lib
% cd -
% make -C cryptocell-rt/host/src ARM_CPU=$ARM_CPU
```

   Verify that `cryptocell-rt/shared/hw/include/dx_reg_base_host.h` matches the address space of the platform.

   It is assumed that the environment is set correctly with a declared variable for compiling the code. For example, `CROSS_COMPILE`, or `KERNEL_DIR`.

   This product was tested with Cortex®-M3 and Cortex®-M33. You must declare which processor you are using with the following command:

   `export ARM_CPU=<cpu-type>`

   This environment variable must be set to one of the following options:

   - Cortex-M3: `export ARM_CPU=cortex-m3`
   - Cortex-M33: `export ARM_CPU=cortex-m33`

   If you are using Arm compilers, and your KERNEL distributes h files (depends on compiler type), the following must also be declared as a prerequisite step:

   `export COMPILER_TYPE=<compile-type>.`

   This environment variable must be set to one of the following options:

   - Arm compiler 6: `COMPILER_TYPE=armclang`
   - Arm compiler 5: `COMPILER_TYPE=armcc`
   - GCC based compilers: `COMPILER_TYPE=gcc`

2. Compile the runtime integration tests:

```
% make -C cryptocell-rt/host/src/tests/integration_* ARM_CPU=$ARM_CPU
```

3. Compile the CMPU integration tests:

```
% make -C cryptocell-rt/host/src/tests/integration_* ARM_CPU=$ARM_CPU
INTEG_TESTS=cmpu_integration_test
```

4. Compile the DMPU integration tests:

```
% make -C cryptocell-rt/host/src/tests/integration_* ARM_CPU=$ARM_CPU
INTEG_TESTS=dmpu_integration_test
```

The integration tests library and the `cmpu` and `dmpu` test libraries will be located in `cryptocell-rt/host/lib`. Use these libraries to build an appropriate executable for the testing platform.

3.2.3 Directory structure

Figure 3-1 shows the principal directory structure of this release created after unpacking the package:

Figure 3-1 Principal directory structure

```
└── cryptocell-312-runtime-master
    ├── codesafe
    │   └── src
    │       ├── crypto_api
    │       │   ├── cc3x_sym
    │       │   │   ├── api
    │       │   │   └── driver
    │       │   ├── common
    │       │   ├── dh
    │       │   ├── ec_edw
    │       │   ├── ec_mont
    │       │   ├── ec_wrst
    │       │   │   └── ecc_domains
    │       │   ├── ffcdh
    │       │   ├── ffc_domain
    │       │   ├── kdf
    │       │   ├── pki
    │       │   │   ├── common
    │       │   │   ├── ec_edw
    │       │   │   ├── ec_mont
    │       │   │   ├── ec_wrst
    │       │   │   ├── poly
    │       │   │   ├── rsa
    │       │   │   └── srp
    │       │   ├── rnd_dma
    │       │   │   └── local
    │       │   └── rsa
    │       ├── mbedtls_api
    │       └── secure_boot_debug
    │           ├── cc3x_verifier
    │           ├── common
    │           ├── crypto_driver
    │           │   └── reg
    │           ├── platform
    │           │   ├── common
    │           │   │   └── cc3x
    │           │   ├── nvm
    │           │   │   └── cc3x_nvm_rt
    │           │   ├── pal
    │           │   │   └── cc3x
    │           │   └── stage
    │           │       └── rt
    │           │           └── cc3x
    │           ├── secure_boot_gen
    │           ├── secure_debug
    │           │   └── cc3x
    │           ├── util
    │           ├── x509_cert_parser
    │           └── x509_verifier
    ├── doxygen
    │   ├── additional_doc_files_cc312
    │   └── doxywrapper
    ├── host
    │   └── src
    │       ├── cc3x_lib
    │       ├── cc3x_productionlib
    │       │   ├── cmpu
    │       │   ├── common
    │       │   └── dmpu
    │       ├── cc3x_sbromlib
    │       ├── cc_mng
    │       ├── hal
    │       │   └── cc3x
    │       ├── pal
    │       │   ├── freertos
    │       │   ├── linux
    │       │   └── no_os
    │       ├── tests
    │       │   ├── common
    │       │   │   └── linux64
    │       │   ├── integration_cc3x
    │       │   │   ├── cmpu_integration_test
    │       │   │   │   └── pal
    │       │   │   │       └── include
    │       │   │   ├── dmpu_integration_test
    │       │   │   │   └── pal
    │       │   │   │       └── include
    │       │   │   └── runtime_integration_test
    │       │   │       ├── pal
    │       │   │       │   └── include
    │       │   │       └── tests
    │       │   ├── proj
    │       │   │   └── cc3x
    │       │   │       └── cc312_r1
    │       │   └── TestAL
    │       │       ├── configs
    │       │       ├── hal
    │       │       │   ├── include
    │       │       │   ├── Juno
    │       │       │   ├── MPS2+
    │       │       │   └── Zynq
    │       │       └── pal
    │       │           ├── freertos
    │       │           ├── include
    │       │           ├── linux
    │       │           ├── mbedos
    │       │           └── no_os
    │       └── utils
    ├── shared
    │   ├── hw
    │   │   └── include
    │   │       ├── mps2
    │   │       ├── mps2.cm33
    │   │       └── zynq
    │   ├── include
    │   │   ├── cc_mng
    │   │   ├── cc_util
    │   │   ├── crypto_api
    │   │   │   └── cc3x
    │   │   ├── mbedtls
    │   │   ├── pal
    │   │   │   ├── freertos
    │   │   │   ├── linux
    │   │   │   ├── mbedos
    │   │   │   └── no_os
    │   │   ├── proj
    │   │   │   └── cc3x
    │   │   ├── sbrom
    │   │   └── trng
    │   └── src
    │       └── proj
    │           └── cc3x
    └── utils
        └── src
            ├── cc3x_asset_prov_rt
            │   ├── examples
            │   └── lib
            ├── cc3x_boot_cert
            │   ├── cert_lib
            │   ├── cert_utils
            │   ├── common_utils
            │   ├── examples
            │   │   ├── content_cert
            │   │   ├── developer_cert
            │   │   ├── enabler_cert
            │   │   └── key_cert
            │   ├── x509cert_lib
            │   └── x509cert_utils
            ├── cmpu_asset_pkg_util
            │   ├── examples
            │   └── lib
            ├── common
            └── dmpu_asset_pkg_util
                ├── common
                ├── icv_key_response
                │   ├── examples
                │    └── lib
                ├── oem_asset_package
                │    ├── examples
                │    └── lib
                └── oem_key_request
                    ├── examples
                    └── lib
```

3.3 Adapt the product for your system

To run cryptographic operations, you must link to all runtime libraries: `libmbedcrypto.a`, `libmbedtls.a`, and `libcc_312.a`.

To operate the production tools, you must link to the libraries of the ICV factory tools and the OEM factory tools: `libcmpu.a` and `libdmpu.a` respectively.

For more information, see the *CryptoCell-312 Software Integration Manual*.

The *CryptoCell-312 Software Integration Manual* is available only to licensees of *CryptoCell-312*.

4 Support

If you have any issues with the installation, content or use of this release, please raise a ticket on https://support.developer.arm.com.

4.1 Tools

This release has been developed with the following tools:

Table 4-1: Tools used in developing this release

Tool usage	Tool name	Version
PC certificate generation tools	OpenSSL	1.0.1f 6 Jan 2014
	Python	3.4.3
Toolchains	Arm Compiler (as part of arm-ds5)	5.06 update 5 (build 528)
	Arm Compiler	6.12
	arm-none-eabi-gcc GCC	7.3.1 20180622
TLS layer	Arm Mbed™ TLS	2.16.2

4.2 OS

This release has been developed with the following operating systems:

Table 4-2: Operating system used in developing this release

Operating System	Version
Ubuntu	16.04.2 LTS: Linux 4.13.0-32-generic x86-64