Application Report
SWRA509A–February 2017–Revised March 2017

SimpleLink™ CC3120, CC3220 Wi-Fi® Internet-on-a-chip™ Solution Built-In Security Features

ABSTRACT

The CC3120 and CC3220 devices are part of the SimpleLink™ microcontroller (MCU) platform, which consists of Wi-Fi®, *Bluetooth*® low energy, Sub-1 GHz and host MCUs. All share a common, easy-to-use development environment with a single core software development kit (SDK) and rich tool set. A one-time integration of the SimpleLink platform lets you add any combination of devices from the portfolio into your design. The ultimate goal of the SimpleLink platform is to achieve 100 percent reuse when your design requires change. For more information, visit www.ti.com/simplelink.

The SimpleLink™ MCU portfolio offers a single development environment that delivers flexible hardware, software, and tool options for customers developing wired and wireless applications. With an ultimate goal of 100 percent code reuse across host MCUs, Wi-Fi®, Bluetooth® low energy, Sub-1 GHz devices, and more, choose the MCU or connectivity standard that fits your design. A one-time investment with the SimpleLink software development kit (SDK) lets you reuse often, opening the door to create unlimited applications. For more information, visit www.ti.com/simplelink.

Contents

Trademarks

1 Introduction

Internet of Things (IoT) products and systems hold information that may be sensitive and private, thus stressing the importance of securing the data. This data may include passwords, keys, credentials, configurations, personal information, vendor intellectual property (IP), and more. Even data that does not seem sensitive or confidential could potentially reveal information to design an attack on sensitive data.

The SimpleLink Wi-Fi CC3120 and CC3220 Internet-on-a chip family of devices from Texas Instruments offers a wide range of built-in security features to help developers address a variety of security needs, which is achieved without any processing burden on the main MCU. This document describes these security-related features and provides recommendations for leveraging each feature in the context of practical system implementation.

The device is designed such that the network security software runs within the dedicated subsystem on the device. This subsystem is an on-chip network processor equipped with a hardware encryption engine, resulting in a separate execution environment, thus offloading the main MCU of the system. The security features span a wide variety of activities typical throughout the product life cycle, including networking activities, data storage, IP protection, cloning protection, and provisions for security during production. These security features are made available to vendors through an ecosystem that incorporates simple and concise APIs, tools, and documentation.

1.1 Terminology

Table 1 provides brief descriptions of the key terms that are required to understand the security methods and features.

Table 1. Terminology

Term	Description
Asset	Asset is any piece of information (security-relevant elements) that has value to its owner. It therefore must be protected by the measures of the target system (by means of confidentiality, integrity, authenticity). Assets may be proprietary information, personal data, or intellectual property.
Asymmetric keys	Asymmetric key pairs are used in algorithms where one party performs a cryptographic operation with a key that is not the same as the key used to apply the reverse operation. The pairs are defined as public and private keys, and used mostly for digital signing and symmetric key distribution.
Exposure point	An identified entry point to the system that enables the launch of one or more security attacks (an attack vector).
Attack vector	A set of actions used to defeat system security measures, thereby gaining control over one or more assets.
Authenticity	Ensures that assets or entities are genuine and authorized to perform a task or used as intended. The verification process usually involves cryptographic algorithms, which check that the entities are who they claim to be. Some predefined trust mechanism is always part of an authentication scheme.
Certificate authority (CA)	A trusted entity that issues certificates used to verify identities.
Certificate chain, Chain of trust	A certificate chain consists of a hierarchy of certificates that allows anyone to verify the identity of any certificate issuer, down to the root certificate.
Certificates	Certificates are standard-formatted files. They typically contain the public key of the subject, and a CA signature of the header and public key. Anyone provided with the CA public key (or sub-CA in case of certificate chain) can verify the subject's identity.
Cipher suite	Cipher suite is a named combination of algorithms used for authentication, key exchange, data encryption, and message authentication code. This set of algorithms is used in an SSL handshake and session.
Confidentiality	Confidentiality ensures that an asset is not made available or disclosed to unauthorized entities. In most cases, confidentiality translates into encryption, while in other cases, obfuscation techniques are used to maintain confidentiality
Integrity	Attribute describing an object that remains intact, in its entirety, compared to its original version.

Table 1. Terminology (continued)

Term	Description
Keys	Keys are used for data encryption, key establishment, and digital signatures. The key lengths and types depend on the algorithm used, their purpose, and the security level.
Revoked certificate	A certificate that is no longer authorized and valid by its issuer.
Root CA	The topmost certificate provided by a certificate authority, against which the certificate chain is eventually verified. It is always self-signed and publicly available.
Security measures	Measures aiming at providing the intended protection of some assets against some threats.
Symmetric key	A key used in algorithms where both parties perform a cryptographic operation with the same key.

1.2 Internet of Things (IoT) Products and Security

The IoT device is by nature a network-connected device, and therefore may serve as a gateway to malicious access to sensitive data, such as surveillance videos, or control over actuators, such as door locks.

To achieve good security for an internet-enabled product, security assessment must be performed on the specific product and its system-level requirements. This assessment should identify the involved assets, analyze the environment as well as intended and nonintended potential usages of the product, and thereby detect potential vulnerabilities of the product.

This assessment helps the developer define the best protection scheme using the available security capabilities. TI recommends minimizing the number of exposure points by removing or disabling any unused service or functionality.

The environment, the assets, and the processes are different from one product to another, but generally for IoT devices there are some common exposure points:

- Internet (or intranet) network connectivity
- Local network connectivity
- Physical access (with or without the ability to manipulate hardware interfaces)

Figure 1 shows the common exposure points of an IoT-connected product.

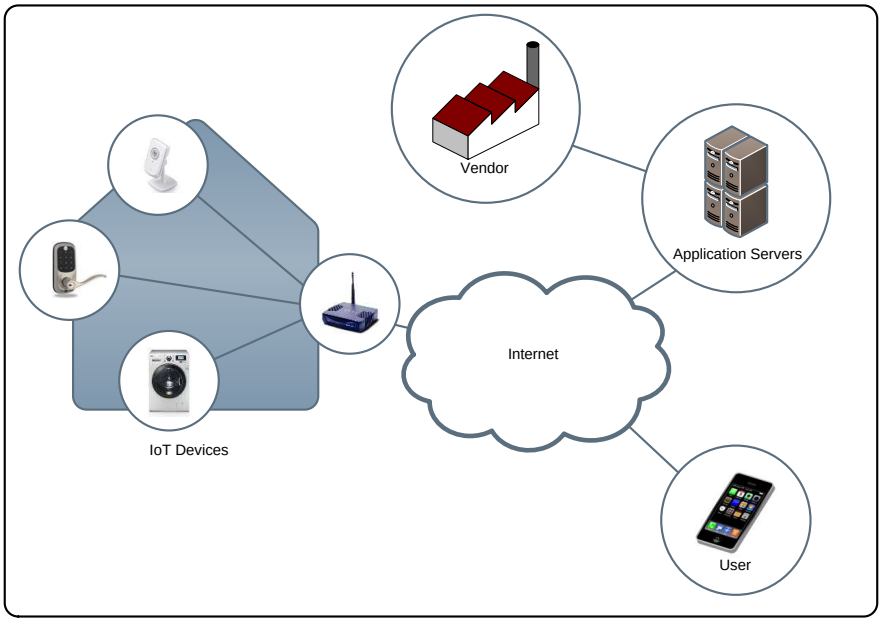

Figure 1. Common Exposure Points of IoT Devices

1.2.1 Internet Network Connectivity

This exposure point relates to all possible attack channels originated from the internet connectivity. The vulnerabilities in this vector could come from all communication channels (such as sockets) and protocols in use. In general, the potential targets of these attacks are all the resources and information that pass through these channels. However, be aware that attackers can try to use these resources to widen their exposure points, and therefore any resource or information on these channels should be examined carefully.

The SimpleLink Wi-Fi device incorporates a variety of features including standard-compliant secure transport layer (SSL/TLS), also referred to as secure sockets, domain name verification, secure content delivery, and device-unique identifiers to provide communication security at the network layer.

These features allow not only data encryption, but also authentication of its origin by means of standard chain of trust verification procedures.

1.2.2 Local Network Connectivity

This exposure point is similar by nature to the internet network connectivity. The general nature of a local network lends itself to a specific set of attack vectors, which are for example the monitoring of the wireless network or injecting malicious or abusive traffic on the WLAN or LAN.

One vector is based on passive monitoring of traffic over the wireless network, without the attacker being connected to it. The monitoring of a wireless network can be done fairly easily, because some of the communication packets headers are not encrypted even in secured wireless networks. These headers might reveal information such as the MAC addresses of the devices on this network and the temporal properties of the traffic they generate.

The Wi-Fi Alliance has regulated security and compliance tests as part of its standards. The SimpleLink Wi-Fi device is a certified Wi-Fi Alliance device, and complies with all these security requirements.

The second vector relates to attacks from a device that is part of the local area network (LAN). This provides additional opportunity for executing an attack vector that involves network access, and the ability to legitimately inject traffic over the network and abuse ports and protocols available on the target device. Access to these ports and protocols is most commonly blocked from the internet, but available on the LAN.

The SimpleLink Wi-Fi device entertains features such as an HTTPS server and RX filters to allow developing additional layers of security for local networks.

1.2.3 Physical Access

In the case of a physical access attack, the exposure is at the product level and there are a couple factors to consider. This occurrence may result in attack vectors such as product-level manipulation and printed circuit board (PCB) manipulation. The product-level manipulation vector refers to attacks in which the attacker can control the way the device is operated by using the external interfaces that the final product offers (such as power line, buttons, and so forth). The second vector relates to the ability of the attacker to have access to the actual board (PCB) to monitor the lines and the hardware interfaces, but in more serious cases even to manipulate the wires, replace devices on the PCB, connect to the main controller, and inject signals to trigger certain actions.

Features such as IP protection, file system security by means of encryption, file integrity checking, cloning protection, and more are provided to help thwart these attack attempts.

1.3 Main Features

The SimpleLink Wi-Fi CC3120 and CC3220 Internet-on-a chip family of devices offer a wide range of built-in security features. These security features can enable and assist designers with addressing a variety of security requirements and reducing the security risk with their intended application. Figure 2 shows a high level diagram of the main security features offered by the SimpleLink Wi-Fi CC3120 and CC3220 Internet-on-a chip family with respect to the access distance of a potential attacker.

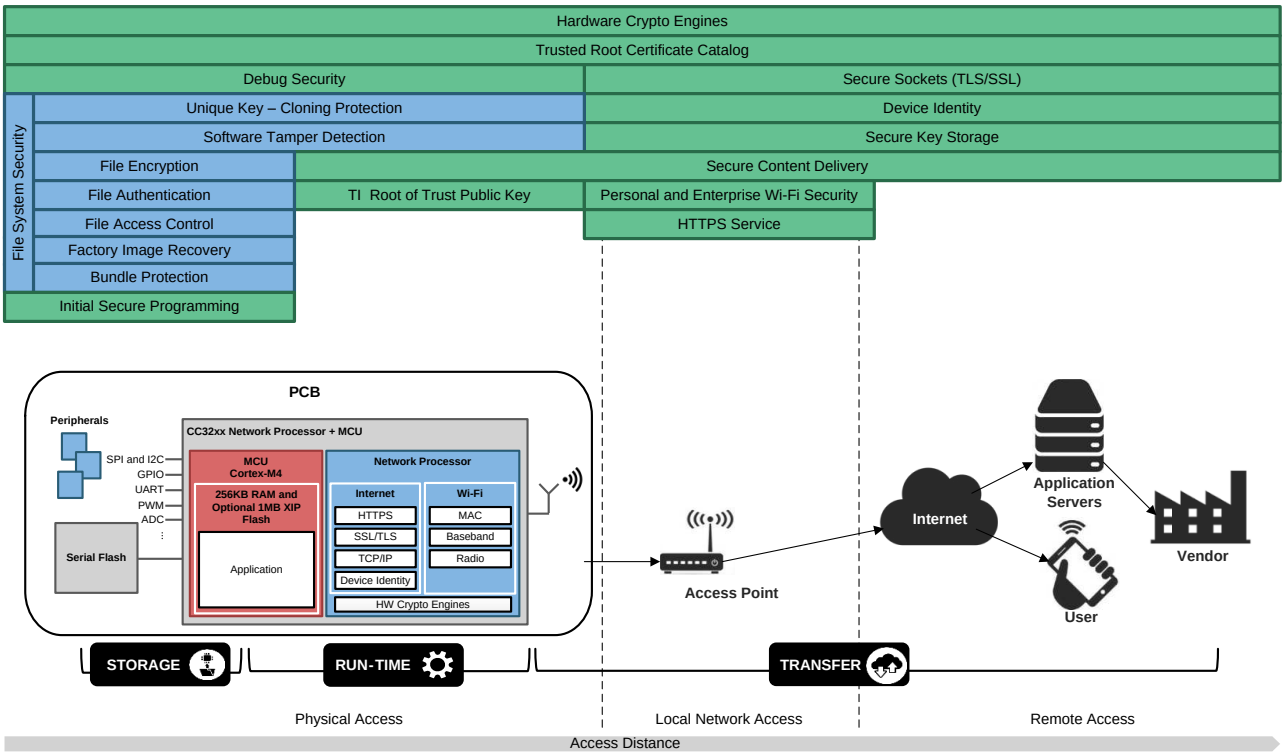

Figure 2. Security Measures

Table 2 lists a high level description of the main security features.

Table 2. Main Security Features

Feature	Description
Personal and Enterprise Wi-Fi security	802.11 standard-compliant security support (WPA/WPA2-PSK/WPA2-EAP)
Secure sockets	Transport layer security comply with SSLv3, TLS1.0/1.1/1.2. Support up to six secure sockets concurrently.
HTTPS server	Internal HTTPS server running on top of a TLS socket, with support for client authentication
Device identity	Unmodifiable unique 128-bit number that TI stores in the device during production. It may serve as a unique device identity (UDID).
Secure key storage	On-chip asymmetric key-pair storage with built-in crypto acceleration and crypto services
Trusted root-certificate catalog	Built-in secure mechanism to ensure a CA is trusted as root of certificate chain for the purpose of TLS and for file signing.
TI root-of-trust public key	The hardware-based mechanism that allows authenticating TI as the genuine origin of a given content (such as software service pack, trusted root-certificate catalog) using asymmetric keys.
File system security	File system security for confidentiality and integrity of data
Secure boot	Validate the integrity and authenticity of the runtime binary during boot (CC3220S and CC3220SF only).
Secure content delivery	Provides an end-to-end ability for delivering confidential information to the system independent of the security of the transport layer.

Table 2. Main Security Features (continued)

Feature	Description
Initial secure programming	Image integrity check and image confidentiality during programming, including system configurations and user files.
Debug security	Blocking the access to debug capabilities such as JTAG interface and file-by-file access from external tools.
Software tamper detection	Detecting and alerting potential unauthorized manipulation of secure file content
Cloning protection	The file system is readable only by the device, which first booted this image.

The CC3220S and CC3220SF have unique architecture, with two physically separate MCU and memory execution environments. Figure 3 shows a block diagram of the security features offered by the CC3220S and CC3220SF devices.

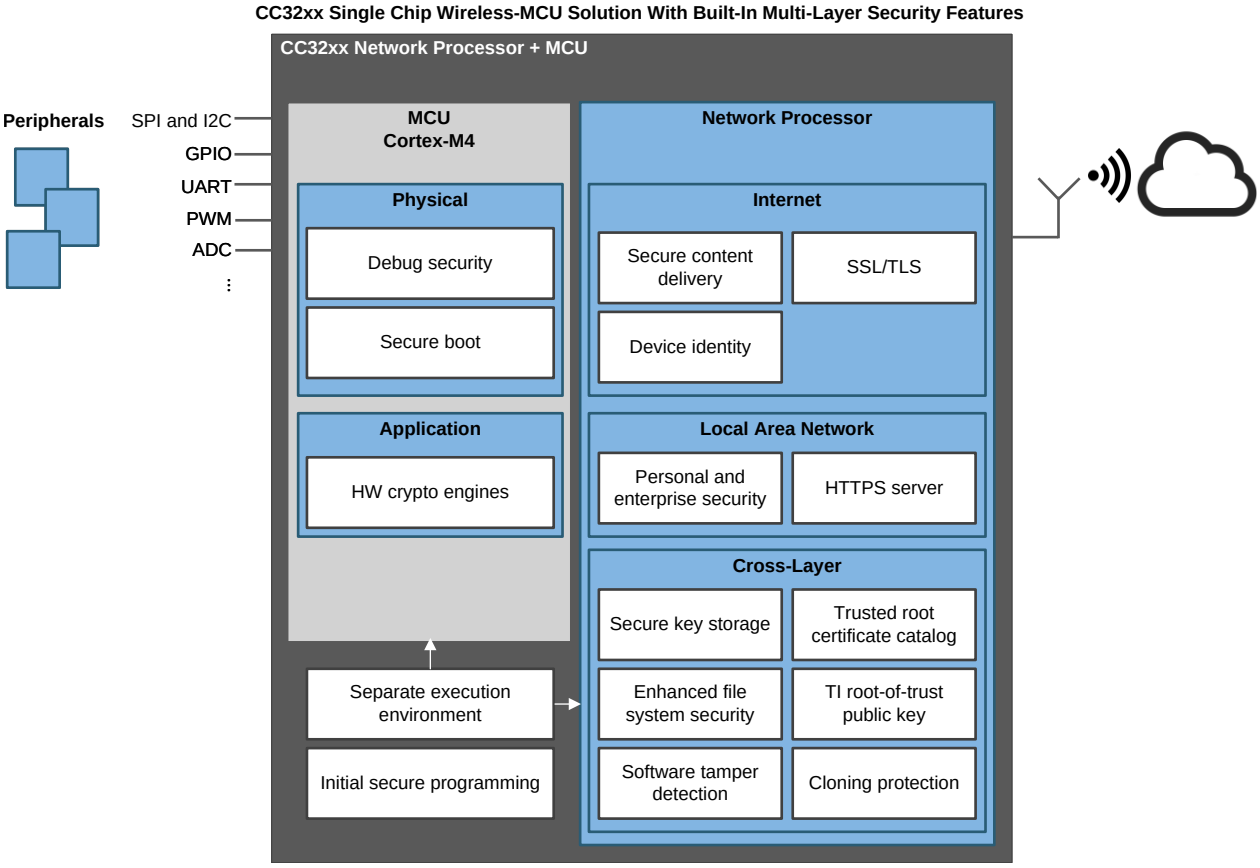

Figure 3. CC3220 Security Features

Figure 4 shows a block diagram of the security features offered by the CC3120 device.

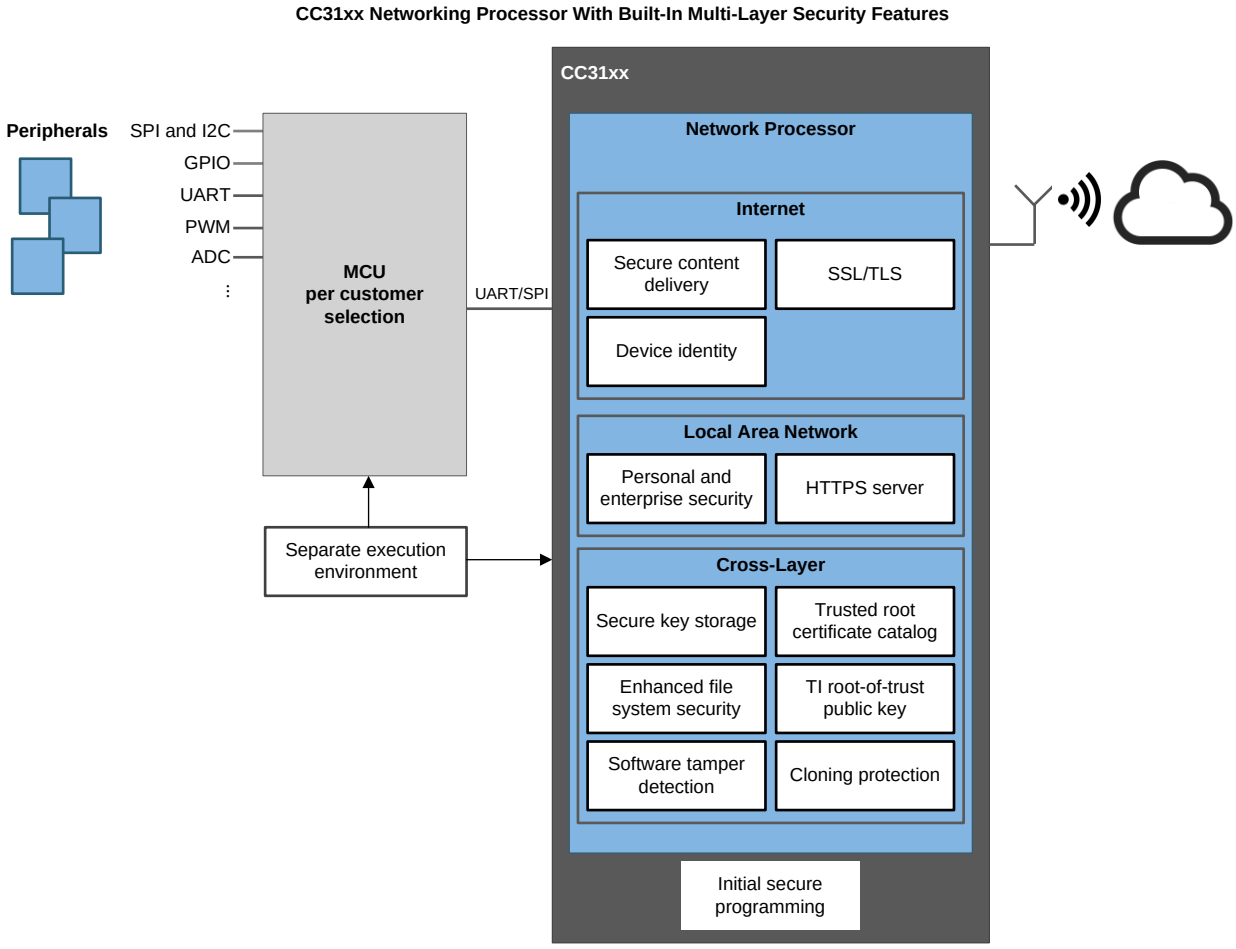

Figure 4. CC3120 Security Features

2 Network Layer Security

The SimpleLink device is a Wi-Fi-based device, which supports 802.11 security protocols for the local segment of the network (between the node and the AP) and TLS/SSL for the transport layer when using TCP/IP for node-to-node communication. TLS/SSL can address confidentiality, data integrity, and authenticity between nodes across the network.

Figure 5 shows common security traffic layers.

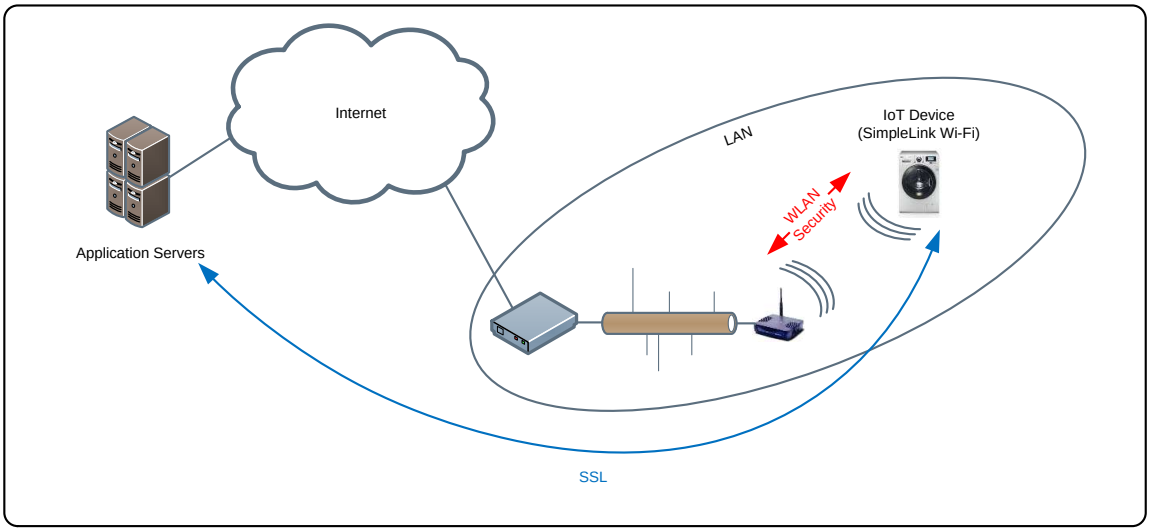

Figure 5. Network Security Topology

2.1 Wi-Fi Security

The Wi-Fi layer of the SimpleLink device complies with 802.11 security to ensure the integrity and confidentiality of the frames (L2 data units) in transactions between AP and STA, or between two peers in the case of Wi-Fi direct mode. The security protocols are described in the IEEE 802.11 specifications and its extensions.

The Wi-Fi subsystem of the SimpleLink device provides support for both personal and enterprise security paradigms, including RADIUS-based authentication (802.1X).

The SimpleLink device is Wi-Fi-certified and complies with Wi-Fi alliance (WFA) security standards and test suits. Table 3 lists the supported Wi-Fi security-related capabilities.

Table 3. Wi-Fi Security

Type	Wi-Fi Security
Personal	AES (WPA2-PSK) TKIP (WPA-PSK) WEP
Enterprise	EAP Fast EAP PEAPv0 MSCHAPv2 EAP PEAPv0 TLS EAP PEAPv1 TLS EAP TLS EAP TTLS TLS EAP TTLS MSCHAPv2

All wireless passwords are stored in an encrypted form on the serial flash. These passwords are solely consumed by the network processor (NWP) and not available to the host through APIs or any debug channel.

NOTE: WEP security is flawed by design and has been compromised. TI strongly recommends not using it. It is supported by the SimpleLink product for legacy compliance reasons only.

2.2 Secure Socket Layer

The SimpleLink device provides a secure transport layer (secure sockets) based on standard-complaint implementation of SSL and TLS protocols, which are network protocols that involve cryptographic paradigms designed to provide communications security over a TCP/IP connection. In most systems, the SSL/TLS is implemented as a layer on top of the transport layer. In the SimpleLink device, the SSL/TLS is embedded into the BSD socket layer, allowing seamless access to secure socket facilities. Extensions to the APIs provide the user some level of control over SSL/TLS behavior and configuration.

The SSL/TLS runs on the network processor subsystem. By design, the architecture of the device is such that the networking processor is a physically separate process subsystem, creating a separate execution environment. Hardware accelerators are used to offload the intense arithmetic calculations involved in the cryptographic algorithms.

The SimpleLink device supports up to six SSL/TLS sockets concurrently.

Table 4 provides a brief description of the supported SSL specification in the SimpleLink device.

Table 4. Supported SSL Specifications

Protocol versions	SSL v3, TLS 1.0, TLS 1.1, TLS 1.2
Ciphers suites	SL_SEC_MASK_SSL_RSA_WITH_RC4_128_SHA SL_SEC_MASK_SSL_RSA_WITH_RC4_128_MD5 SL_SEC_MASK_TLS_RSA_WITH_AES_256_CBC_SHA SL_SEC_MASK_TLS_DHE_RSA_WITH_AES_256_CBC_SHA SL_SEC_MASK_TLS_ECDHE_RSA_WITH_AES_256_CBC_SHA SL_SEC_MASK_TLS_ECDHE_RSA_WITH_RC4_128_SHA SL_SEC_MASK_TLS_RSA_WITH_AES_128_CBC_SHA256 SL_SEC_MASK_TLS_RSA_WITH_AES_256_CBC_SHA256 SL_SEC_MASK_TLS_ECDHE_RSA_WITH_AES_128_CBC_SHA256 SL_SEC_MASK_TLS_ECDHE_ECDSA_WITH_AES_128_CBC_SHA256 SL_SEC_MASK_TLS_ECDHE_ECDSA_WITH_AES_128_CBC_SHA SL_SEC_MASK_TLS_ECDHE_ECDSA_WITH_AES_256_CBC_SHA SL_SEC_MASK_TLS_RSA_WITH_AES_128_GCM_SHA256 SL_SEC_MASK_TLS_RSA_WITH_AES_256_GCM_SHA384 SL_SEC_MASK_TLS_DHE_RSA_WITH_AES_128_GCM_SHA256 SL_SEC_MASK_TLS_DHE_RSA_WITH_AES_256_GCM_SHA384 SL_SEC_MASK_TLS_ECDHE_RSA_WITH_AES_128_GCM_SHA256 SL_SEC_MASK_TLS_ECDHE_RSA_WITH_AES_256_GCM_SHA384 SL_SEC_MASK_TLS_ECDHE_ECDSA_WITH_AES_128_GCM_SHA256 SL_SEC_MASK_TLS_ECDHE_ECDSA_WITH_AES_256_GCM_SHA384 SL_SEC_MASK_TLS_ECDHE_ECDSA_WITH_CHACHA20_POLY1305_SHA256 SL_SEC_MASK_TLS_ECDHE_RSA_WITH_CHACHA20_POLY1305_SHA256 SL_SEC_MASK_TLS_DHE_RSA_WITH_CHACHA20_POLY1305_SHA256
Maximum secure sockets	6
Other	Server authentication Client authentication Domain name verification Separate execution environment

2.2.1 SSL/TLS Certificate Handling and Other Security Files

Certificate files, DH (Diffie-Hellman) parameters, and keys must be stored on the file system of the device to enable some of the SSL/TLS capabilities in accordance with the selected cipher suite (see Table 5).

Certificates are standard-formatted files carrying public keys of an asymmetric key-pair, and are thus not confidential by nature. Private keys, on the other hand, are confidential and should be kept encrypted. Access control restrictions should be applied to these files, limiting read access to the rightful owner (see Section 3.2.4).

The keys and certificates are used internally by the SSL sockets when the handshake occurs. The host must only bind the related files to the SSL/TLS socket using APIs.

Table 5 lists the files supported by the SimpleLink SSL/TLS socket.

Table 5. SSL Files

File	Client	Server
Root CA file: A self-signed certificate that signs the chain of the remote peer	Validates the server chain of trust. ESECSNOVERIFY warning is returned if not installed, but connection is not implicitly terminated. Format: PEM/DER	Enables client verification if installed ESEC_NO_PEER_CERT event is issued by the device if the client failed to provide the proper chain of trust to match the installed CA certificate. Format: PEM/DER
Certificate file: A certificate issued to this entity	This certificate file is used as a client certificate if the server requires client authentication. If the file does not exist and the server requires a client authentication, the server returns an ALERT of peer verify error. A private key must be programmed with this file, or else the connection fails with ESECBADPRIVATEFILE. Format: PEM/DER	This certificate file is used as a server certificate or certificate chain. The certificate should be the first in the list. The file must be configured. If it is not configured, it results in error EBADCERTFILE. Format: PEM
Private KEY file: RSA or ECDSA key	Used as a client private key if the server requires client authentication. ESECBADCERTFILE if a matching public certificate file is not installed with this file. Format: PEM/DER	Used as the private key of the server. Must be programmed to connect to the server using SSL/TLS. EBADCERTFILE is returned if not available. Format: PEM/DER
PEER Certificate, DH Param (client, server)	Programming this file enables the domain verification by full server certificate comparison. This certificate is compared to the one presented by the server certificate during the SSL/TLS handshake process, to protect from MITM attack. It is used to validate that this is the expected server to which connection was attempted. Format: PEM/DER	Used as DH Param file, enables the use of DH-based ciphers. Format: PEM/DER

2.2.2 SSL Handshake Overview

After establishing a TCP connection between two peers, the SSL protocol takes place in negotiating the session security parameters and authenticating the peers.

A short description of this handshake process follows. This handshake is performed by the SimpleLink device, and remains transparent to the user:

1. A client hello message is sent with the following parameters:
   - Protocol version to be used – SSLv3, TLS1.0, TLS1.1, TLS1.2
   - Cipher suites supported by the client
   - Extensions, used for different applications of the SSL
2. The server chooses the protocol version and a cipher suite from the client hello cipher list, and sends a ServerHello message to the client.

3. The server sends the certificate chain (except for the root CA) to the client.
4. The client may verify the chain by checking the signature of each certificate in the chain, starting with the root CA held by the client. The server may request a certificate from the client for a client authentication.
5. The peers determine session (symmetric) keys. The client and server start transferring encrypted data from that point.

Figure 6 shows the SSL handshake process.

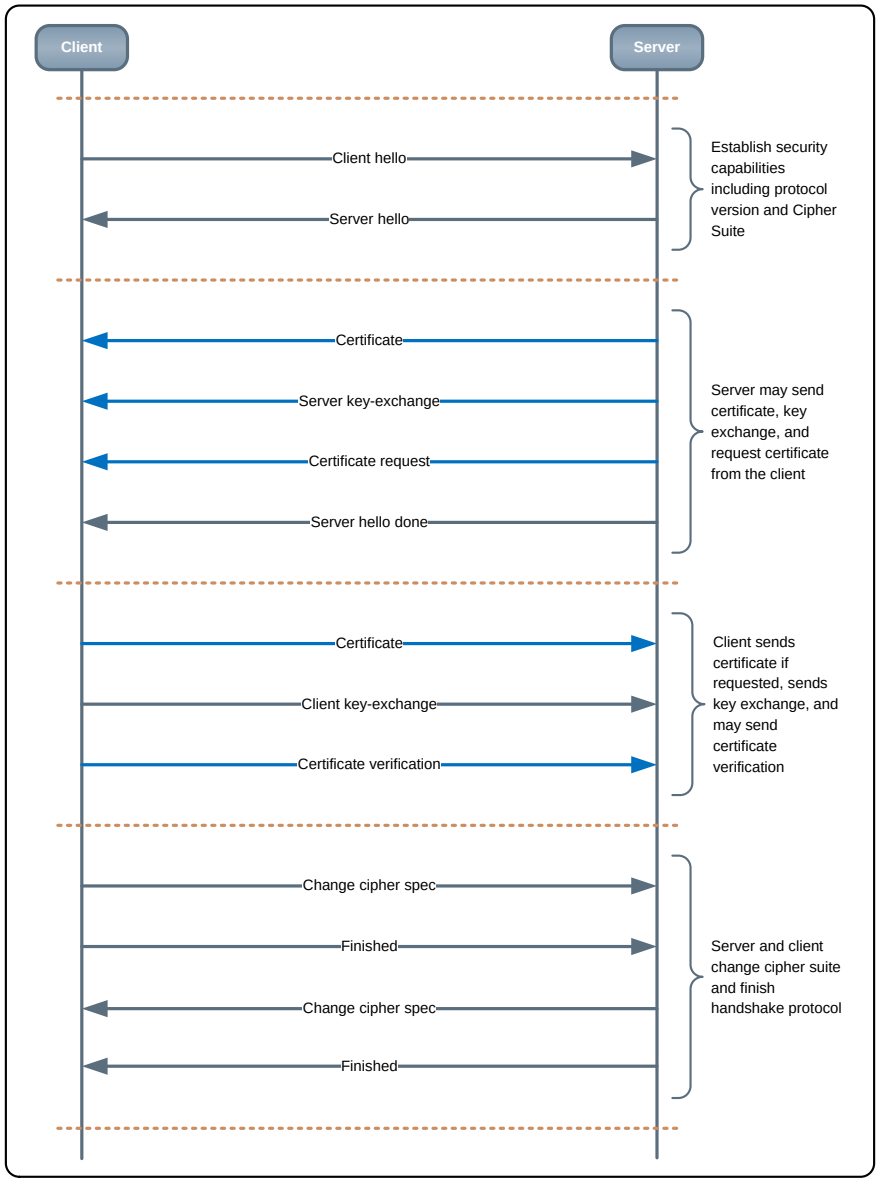

Figure 6. SSL Handshake

2.2.3 SSL/TLS Handshake With the SimpleLink™ Device

The SSL/TLS protocol is an integral part in the network stack of the SimpleLink device, running within the networking subsystem, thus creating a separate execution environment. The only interface to this layer is through API calls from the application processor.

2.2.3.1 SSL/TLS Handshake Supported Capabilities

Table 6 lists the SSL handshake features.

Table 6. SSL/TLS Main Handshake Features

Operation	Client	Server
Authenticating the peer	√	√
STARTTLS	√	√
Server domain verification	√	X
Trusted root-certificate catalog	√	X
Time and date validation	√	X

2.2.3.2 Basic SSL/TLS Connection

On most systems, the SSL/TLS protocol is implemented on top of the TCP layer. In the SimpleLink device, the SSL/TLS socket is embedded into the TCP layer, and retains the look and feel of a standard TCP socket. The entire SSL/TLS handshake is triggered automatically during the connect session (when the connect() or accept() APIs are invoked). This method simplifies the work with secure sockets. Opening a socket with a security parameter in the protocol argument and then calling the connect or accept commands establishes a secure connection to the other peer.

In server mode, a server certificate or certificate chain and a private key must be supplied. This chain is sent in the handshake. Absence of these files raises an error when sl_Accept is called.

In client mode, a client certificate and its private key can be supplied in case the remote server policy requires client-side authentication.

2.2.3.3 Errors and Warnings

The BSD commands return detailed errors on any fault, including the secure socket handshakes.

Some of the error codes reflect security-related warnings, without blocking the process. The developer is urged to consider the following warnings because they indicate a potential security problem:

- SL_ERROR_BSD_ESECSNOVERIFY – Connected without server verification
- SL_ERROR_BSD_ESECDATEERROR – Connected with certificate date validation error
- SL_ERROR_BSD_ESECUNKNOWNROOTCA – Connected but the root CA is untrusted

2.2.3.4 Authenticating the Peer

A basic connection to a server leads to an error received on the sl_Connect command – SL_ERROR_BSD_ESECSNOVERIFY. This means that the data flow is encrypted from now on, but the SimpleLink did not validate that the remote side is an authentic server.

In this case, the data is encrypted and is now confidential from an eavesdropper, but the server identity cannot be verified, and Man-In-The-Middle (MITM) attacks or phishing attacks can be applied.

Setting a root CA that signed the server certificate triggers a signature check on the server chain of trust, also called a *full chain of trust verification*. If the validation fails in this case, the connection is terminated.

If the SimpleLink is in server mode, it can verify clients by setting the root CA that signed the client certificate. If the validation fails, the connection is terminated.

2.2.3.5 Domain Name Verification

Chain of trust evaluation ensures that communication is carried out with a trusted entity. However, it still does not warrant it is the identity with which communication was intended.

This is achieved by domain name verification. This procedure verifies that the identity with which communication was intended is the one with which communication is actually carried out. This is achieved by comparing the content of the certificate presented by the server and the expected prestored certificate.

If the comparison fails, a security warning is triggered and connection is kept. Handling this warning is up to the discretion of the developer.

2.2.3.6 Trusted Root-Certificate Catalog

The trusted root-certificate catalog is a file, provided and signed by TI, that contains a list of known and trusted root CAs (a complete list is available as part of TI SDK and collateral). The catalog holds trusted root certificates from commonly trusted CAs (such as VeriSign®, GoDaddy®, GeoTrust®, and more). The trusted root-certificate catalog also holds a list of revoked certificates known to TI. The trusted root-certificate catalog is used only in client mode. When acting as a server, using a self-signed root CA to authenticate a client is allowed.

When a root CA is unknown to the trusted root-certificate catalog, the sl_Connect command returns the warning SL_ERROR_BSD_ESECUNKNOWNROOTCA, which means that the connection was successful, but the root CA used to verify the server chain of trust is unknown. In this case, the developer should make a choice whether to terminate the connection explicitly or proceed at its own risk.

When a revoked certificate has been received during the SSL/TLS connection (all the certificate chain is checked) or if the root CA that has been set by the user is revoked, the handshake is terminated, and the SL_ERROR_BSD_ESECCERTIFICATEREVOKED is returned by the sl_Connect command.

The trusted root-certificate catalog is stored as a file in the file system, and can be updated to a newer version only.

Figure 7 shows the relationship between the certificate files and the trusted root-certificate catalog.

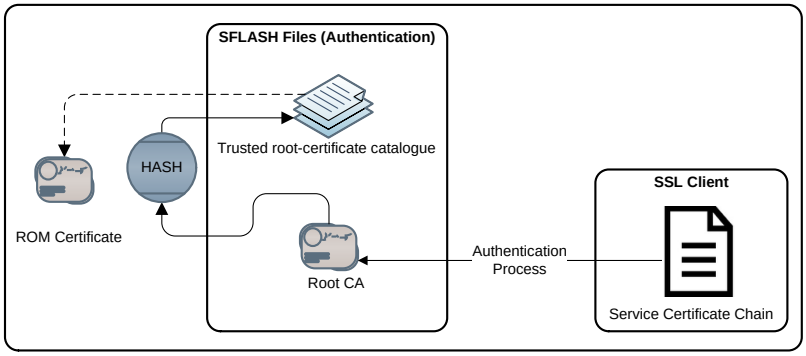

Figure 7. Certificate Files and Relationship With the Trusted Root-Certificate Catalog

Lists of the root CAs that are approved by TI are available as part of the SDK, and can also be found here.

2.2.3.7 Validating the Date of the Server

Every certificate has an expiration date. When connecting to a server in client mode, the server time validation occurs if the root CA was set. If the time validation fails, the sl_connect command returns an error SL_ERROR_BSD_ESECDATEERROR, “connected with time and date validation error.” The user can set the time and date of the device, and this is used in the certificate time and date validation. The date and time are kept in hibernate mode.

Setting a date and time results in a write operation to the serial flash, and should therefore be part of flash endurance considerations.

2.2.4 Maximum Number of SSL/TLS Connections

The SimpleLink device has a maximum of six concurrently connected SSL/TLS client sockets. In case of a server socket, the accept socket (the socket on which the server is listening for new connections) is not counted.

2.2.5 Selecting SSL Parameters

It is possible to set the SSL versions allowed to use a specific cipher list for different implementations. Some cipher algorithms are done by hardware and others by software, making it easy to choose only the hardware-accelerated algorithm.

2.2.6 Security Changes on Socket (STARTTLS)

A regular TCP socket can be upgraded to an SSL/TLS socket. The user must trigger the SSL/TLS handshake explicitly, irrespective of the socket connection (triggered by either sl_connect or sl_accept APIs). This implementation is usually used in applications that use STARTTLS to upgrade a connected session to a secured connection.

An example of this application is the SMTP on port 587. A secured socket can be downgraded to a nonsecured TCP socket only if an SSL/TLS shutdown alert is received from the peer. The SimpleLink user cannot initiate an SSL/TLS shutdown on a secured connection. If a socket was upgraded or downgraded, an asynchronous event is issued to the host indicating the new stage.

3 File System Security

3.1 Overview

The SimpleLink solution has an externally-attached non-volatile memory (NVM) in the form of a serial flash device. Data on the SFLASH is organized in a file system and the device provides APIs to access it. The file system can be used by the host application for creating, writing, and reading files. The file system is also used by the networking subsystem to store configuration and temporary files.

System files are created during the programming process or by the device to hold device state and configurations. These files are inaccessible from the host.

The file system has built-in capabilities designed to keep content protected. Some of these capabilities are applied implicitly, while others leave control for the user.

The following features comprise the device file system security feature set:

- Encryption – Files are stored and encrypted using a standard AES-128 encryption.
- Cloning protection – The entire file system content is only readable by the device instance, which has first booted this image, because the image is encrypted with a device-unique key.
- File system integrity verification – The integrity of the file system structure is verified to check whether the file system was manipulated.
- Software tamper detection – Content integrity can be validated by the file system, and an alert is triggered when a malicious access to a file is detected.
- Access control – File access may be restricted to the rightful owner, per access type.
- Origin authentication – Files can be defined to restrict update eligibility only with a valid certificate provided.
- Recovery mechanism – The file system can be restored to an initial programmed configuration.

The service pack and the trusted root-certificate catalog are system files which must be created as secure files. Those files can only be read by the device. Files which contain confidential data as keys for the SSL/TLS connection should also be created as secure files.

3.2 File System Security Features Description

3.2.1 Encryption

Confidentiality is possible through data encryption. In this case, only the encrypted version of the file is kept on the serial flash. The AES-128-CTR is the selected encryption method. Keys are implicitly generated by the device using a True Random Number Generator (TRNG) hardware engine. Decryption is implicit, and executed on the fly while reading the file.

3.2.2 Cloning Protection

The file system metadata is encrypted with a device key. This key is unique per device, and thus moving a file system that was created on one device to another device is not possible. Doing so results in a security violation detection during the boot sequence, and the device enters a lock state.

3.2.3 Integrity Verification

An integrity check is applied for secure files created with the *secure-and-authenticate* flag. The integrity is conducted by the SimpleLink device. Upon each change of the file, a new hash is executed and the result is a file signature. When the file is opened for read, the file content is verified against this signature.

System files and the file system structure are also protected, and their integrity is tested by the device. The integrity validation of system files is done by using the HMAC-SHA-256 algorithm.

If the device identifies a verification failure, a system security alert is triggered.

If the device identifies a verification failure of the file system structure or a system file, it changes its state to a locked state.

3.2.4 Access Control

Each secure file has several access levels; the access levels define restrictions over the nature of allowed access to the file. Tokens are used to control the access to the secure files.

When a secure file is created, four different tokens (32-bit number) are generated. Each token provides a different access level to the file, as listed in Table 7.

Table 7. Tokens

Type	Description
Master token	Enables full access to the file. A secure file can be deleted only with the master token. The master token is unchanged, as long as the file exists.
Read/write token	Enables read and write access. This token is generated automatically when the file has been changed.
Write only token	Enable write access only. Used for producer-oriented files, such as a private key. This token is generated automatically when the file has changed
Read only token	Enables read access only. Used for consumer-oriented files, such as a public key. This token is generated automatically when the file has changed

The SimpleLink device automatically generates the tokens for secure files. The master token is the output parameter of the file creation function, and is received by the host.

The get info function enables the host to retrieve all the accessible file tokens when setting a valid file token as an input. Only tokens with lower priority than the input one can be retrieved.

The default system behavior is to regenerate all the file tokens, except for the master token, each time the file is opened for a write operation. This default behavior can be changed by the file creation flags.

Table 8 lists the file creation flags that can override the default behavior of the token creation.

Table 8. File Creation Flags – Security Attributes

File Creation Flag	Description
SL_FS_CREATE_STATIC_TOKEN	The tokens for the file are generated when the file is created, and are not changed when the file content is changed. This is useful for files whose access tokens cannot be manipulated dynamically.
SL_FS_CREATE_VENDOR_TOKEN	The master token is set by the host and is the same across all devices carrying the same vendor code (rather than unique per device).
SL_FS_CREATE_PUBLIC_WRITE	The file content can be updated without using a token. Other file operations, such as read and delete, requires a token.
SL_FS_CREATE_PUBLIC_READ	The file can be read without token. Other file operations, such as write or delete, require a token.

When a secure file is opened for write, the open-write function returns the new generated token. The returned token is on the same level as the input token.

For example, for a file opened for write with the write token as input, the new file write token is returned.

3.2.5 Trusted Source Verification

The file system provides another layer of protection by ensuring that the origin of the file signature is trusted by a public root CA. This procedure is applied by default on secure files.

The authentication method used for this process follows PKI with a key size of either 128 or 256. The file signature verifies both the file integrity and the chain of trust used to sign this file.

To create a digital signature for the authentication process, the vendor must have an RSA key-pair. The certificate that holds the public key must be signed by a certificate chained up to a known root-certificate authority (CA). The entire chain of trust for this certificate is verified by the device, and the signature of the root CA is compared to the expected hash using the trusted root-certificate catalog.

NOTE: The certificate expiration date is not validated during this file authentication process, because there is no absolute way to ensure the local date and time validity can work in all use cases.

When creating a new secure file, there is an option to skip the file authentication default behavior. This is done by using a dedicated creation flag: SL_FS_CREATE_NOSIGNATURE.

3.2.5.1 Origin Authentication for the Service-Pack and the Trusted Root-Certificate Catalog

The service pack and the trusted root-certificate catalog are files provided and signed by TI only.

Writing the service pack and the certificate store files requires a signature; the signatures for those files are supplied by TI. Certificate material is preinstalled on the device.

The trusted root-certificate catalog contains a list of revoked certificates. If a certificate from this list is encountered in the course of an SSL/TLS connection, the connection is terminated and the device triggers an event to the host.

Figure 8 shows the file authentication process.

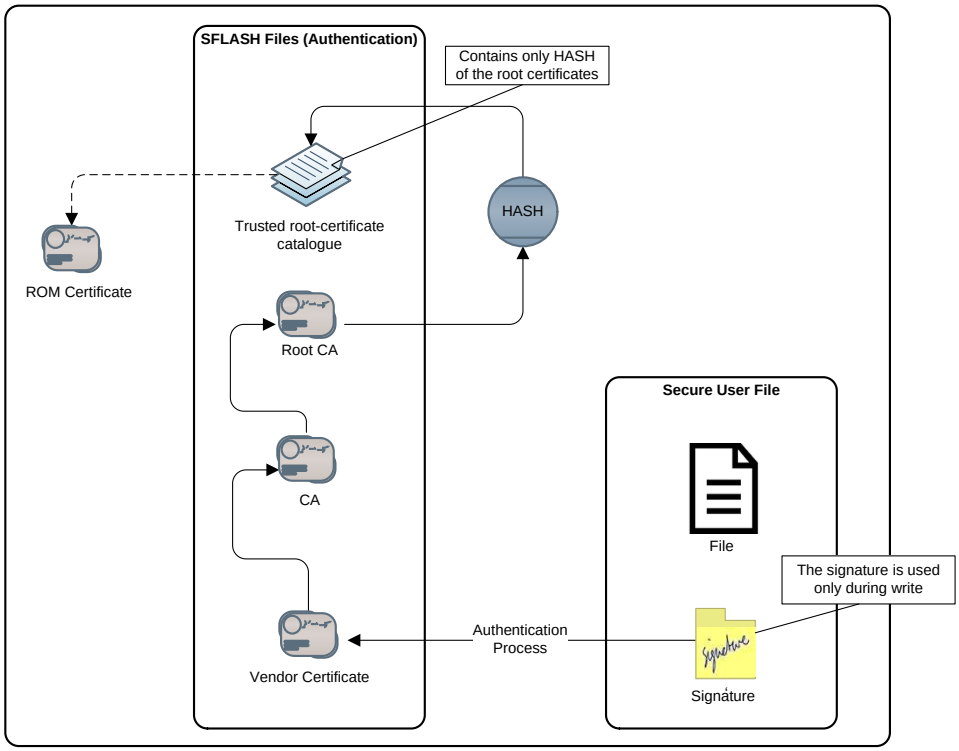

Figure 8. Validate File Signature

3.2.6 Recovery Mechanism

The SimpleLink device embeds an internal recovery mechanism that allows the file system to be rolled back to a predefined image programmed during production.

The recovery image is a system file kept on the serial flash. The file is stored as a secure file. The recovery image file is listed as part of the files list, but is inaccessible from the host.

The following methods can be used to trigger the recovery procedure:

- Setting the SOP
- Function call from the host to sl_DeviceSet

The recovery mechanism supports several modes; the device recovery mode is configured during the creation of the file system image.

The supported recovery modes are:

- None – No recovery settings; in this case, the recovery image file is not kept.
- Enable restore to factory-default.
- Enable restore to factory-image and factory-default.

Restore to factory-image – The factory image file contains the programmed image files, which includes the service pack, host application (in the case of the CC3220), configuration files, and user files. Once invoked, the device is formatted and reprogrammed. The content of all files is replaced with the factory image files. Files that do not exist in the original image are deleted. The restore to factory-image can be invoked by a host command or by setting the SOP.

Restore to factory-default – Restore to factory-default is similar to restore to factory-image. The difference is that the service pack and the host application (in the case of the CC3220) are not restored, so the latest service pack and the latest host application are used. When using this method, any file that the host app might use is restored to the programmed content.

The restore-to-factory process is fail-safe, and it resumes even if a power failure occurs during the operation.

3.2.7 Tamper Alerts

The SimpleLink device provides a data-tampering procedure with a security-alert counter. This procedure detects integrity violations when accessing the file system, for system files and files that are created as *secure and authenticate*.

Access violations (such as attempts to access files with an invalid or improper token) are also monitored and detected as a tamper attempt.

When the system reaches the predefined limit of security alerts, the device is locked and an asynchronous event is triggered.

To recover from a locked state, the device must be reprogrammed or restored to factory image.

The security alerts counter is persistent and its value is set to 0 as part of the programming process, or as part of the restore-to-factory functions only.

The security alert threshold can be configured during image creation (using the image creator tool).

The host can retrieve the current number of security alerts in the system using the sl_deviceGet API.

Table 9 lists the different security alerts.

Table 9. Security Alerts

Type	Description
Explicit alerts (critical)	The device is locked immediately regardless of the alert counter threshold. Explicit alerts are created when detecting the following tamper events: • File system structure integrity error • System file integrity error • Cloned SFLASH
Implicit alerts	The device is locked when the alert counter crosses the alerts threshold. Implicit alerts are created when detecting the following tamper events: • Invalid or improper access token • Integrity violation of a file created as a *secure and authenticate* file when opened for read • Setting an invalid signature or invalid certificate when updating the content of a *secure and authenticate* file

3.2.8 Special System Files

Table 10 lists the files with a special treatment by the SimpleLink device.

Table 10. Special Treatment Files

Type	Description
Service pack	The service pack is created as secure-signed-public write. The signature of this file is provided by TI and validated by the device using the hardware-based root of trust.
Trusted root-certificate catalog	The trusted root-certificate catalog is created as secure-signed-public write. The signature of this file is provided by TI and validated by the device using the hardware-based root of trust. The file is also protected against downgrade; the host cannot write an older version than the one installed.
Runtime binary (host application)	The runtime binary file must be created as secure-signed. The vendor is responsible for creating the host signature and supplying a signed certificate.

3.2.9 Host Interface

The host can create files on the SimpleLink file system, and sets the security level of the created files.

Table 11 lists the possible security levels.

Table 11. File Security Levels

Type	Description
None	The data is stored as plaintext. There is no validation on the data.
Secure file	The file content is stored encrypted. Access to the file requires a token.
Secure-and-Authenticate file	The file content is stored encrypted. Access to the file requires a token. The integrity of the file is tested each time the file is opened for read. The origin of the file authentication and the integrity of the file is validated with each change of the file.

3.3 File Creation Attributes

The file attributes are defined during creation, including security properties of the file. This sets the security-related treatment of this file.

The creation flags are set during the file creation and cannot be changed.

Table 12 is a list of creation flags.

Table 12. Creation Flags

Type	Description
SL_FS_CREATE_FAILSAFE	A file opened with fail-safe has allocated place for two copies, but only one copy is considered to be active at a time. Each time a file is opened for write, the inactive location is erased. If the system is powered off while writing a file with fail-safe, the old instance leaves the active one.
SL_FS_CREATE_SECURE	A file created as secure has its content encrypted on the SFLASH. Access to the file is limited, and requires a file token.
SL_FS_CREATE_NOSIGNATURE	The flag is relevant only for secure files. By default, a secure file requires a signature during write. This signature is used to authenticate the origin of the file. This flag lets the user skip the authentication check during write.
SL_FS_CREATE_STATIC_TOKEN	Relevant only for secure files. This flag changes the default behavior of the file tokens creation: with this flag, the file tokens are not changed each time a file is opened for write.
SL_FS_CREATE_VENDOR_TOKEN	Relevant only for secure files. This flag changes the default behavior of the file tokens creation: with this flag, the file master tokens are set by the host.
SL_FS_CREATE_PUBLIC_WRITE	Relevant only for secure files. This flag changes the default behavior of the file tokens creation: with this flag, the file can be written without a token.
SL_FS_CREATE_PUBLIC_READ	Relevant only for secure files. This flag changes the default behavior of the file tokens creation; with this flag, the file can be read without a token.

The flags listed in Table 13 are not creation flags. These noncreation flags can be set every time a file is opened for write.

Table 13. Noncreation Flags

Type	Description
SL_FS_WRITE_BUNDLE_FILE	Used for the bundle commit feature
SL_FS_WRITE_ENCRYPTED	Used for secure content delivery

4 Programming the Device

The process of programming the device has a major impact on system integrity. To ensure the integrity of the system, the SimpleLink device applies an image-based programming methodology. The device programming procedure uses a single image file, which contains all device configurations, runtime binary, and other complementary data files. The ability to pack all these into a single file, which is then used for programming, ensures that there is no situation in which the device is partially configured.

Image creation is handled by a cross-platform external tool provided by TI. The tool takes all the material that composes the image as an input, and outputs an image binary file that could then be flashed onto the serial flash attached to the device.

The actual burning process is independent of the image structure. The process can therefore be carried out either by the tool, or any other standard flashing tool.

This image serves as the factory image for recovery, if this option is used.

4.1 During Development

Image-based methodology may be less convenient during the development phase. To allow easy development with the SimpleLink device while not compromising the security features, the SimpleLink device enables a separate methodology for development (debug security).

During development mode, all security capabilities are available for use, however to allow for the proper debug environment, all debug interfaces are available, and it is possible to modify the file system content on a file-by-file basis (to avoid the burden of regenerating and programming a new image for each change).

To avoid surrendering the device-unique key material, a separate set of keys is used while in this mode.

The device is put into development mode through the image type, which is determined through the image creation. The development image is device-unique. The image is tightly coupled with the target device onto which it is programmed (according to its hardware-defined MAC address). This inhibits wide use of a development image for production.

The following functionality is exposed in development mode:

- File access is permitted from the image creation tool (file-by-file access).
- File list can be accessed from the image creation tool.
- JTAG interface is enabled (CC3220 only).

4.2 During Production

Production image is appropriate for mass production, and can be programmed to many devices simultaneously. This image is deflated upon first boot of the device and transformed into a device-unique image. JTAG interface is blocked.

4.2.1 Programing Methods

The SimpleLink device offers the following programming methods:

- Image creation tool (using UART transport)
- Host programming (using host APIs)
- External programming (such as "gang" programming)

4.2.1.1 External Programming (Gang Programming) Details

To operate in this method:

1. The serial flash is programmed using a third-party programmer while it is not assembled on the target PCB. The third-party programmer tool is required to support standard file formats (Intel hex or binary raw).
2. The programmed serial flash is assembled on the target PCB.
3. During first power up, the SimpleLink device detects the image, ensures its integrity, and starts deflation.

It is possible to encrypt this image to make it confidential from the third party handling the gang programming. In this case, the key is provided during image creation and the output image is encrypted with this key. In production, upon image detection, the device awaits activation from the OEM by providing the key for decrypting the image over the UART lines. Once the image is authenticated, deflation may commence.

NOTE: The CC32x0 LaunchPad™ Development Kit is equipped with a connector (J25) to allow attaching an external programmer to test this method.

5 In-field Software Updates

The SimpleLink device provides methods for in-field software updates while maintaining system integrity, and validates the origin of the software update using a CA-based chain of trust.

5.1 File Bundle Protection

The bundle protection is a method offered by the SimpleLink device for keeping the system integrity while updating a collection of files referred to as a bundle. A bundle allows applying an update to the entire collection, or rolling back to a previous version. As a result, an intermediate mixture is avoided.

5.1.1 File Bundle Update Workflow

All files belonging to a bundle must be created as FAILSAFE. Only a single bundle can be updated at any certain point.

1. Start the update procedure by writing the new version of each file in the bundle with the bundle flag set.
2. Restart the device.
3. Upon failure, the bundle can be rolled back to its previous version, rendering it rejected. In this case, all the files are recovered to their previous content.

The process of updating the bundle files is fail-safe, and an unintentional reset during the process is considered as an abort of the bundle update procedure. As a result, automatic rollback is triggered and all the bundle files are reverted to their previous copy.

New files can be created in the course of a file bundle update. If the bundle is rolled back, the new files are deleted.

5.2 Secure Content Delivery

The secure content delivery capability provides yet another level of security at the application layer, and enables the exchange of confidential content to the device. The content is decrypted by the device, and no entity along the way carries the knowledge to decipher it. This capability is useful in cases such as over-the-air (OTA) updates. Secure content delivery lets the user program content that has been encrypted by a remote device and decrypted on-the-fly inside the NWP, while the private key used for the process is kept inaccessible, inside the SimpleLink Networking subsystem, with no access from the host. As this capability is applied at the application layer, it allows transferring a file to the system on unsecured tunnel.

5.2.1 Process Description

The process description for the secure content delivery capability follows:

1. Generate a key-pair using the NetUtils APIs.
2. Retrieve a temporary ECC public key.
3. Send the public key to the application remote server.
4. Receive an encrypted file.
5. Open a new file with a special flag indicating secure content delivery is about to be written.
6. Write the encrypted file content sequentially.
7. Close the file.

At the end of this process, the file is saved on the SFLASH, and the file is encrypted as a normal secured file with a different key and method than the one used for the reception process.

Figure 9 shows the secure content delivery process.

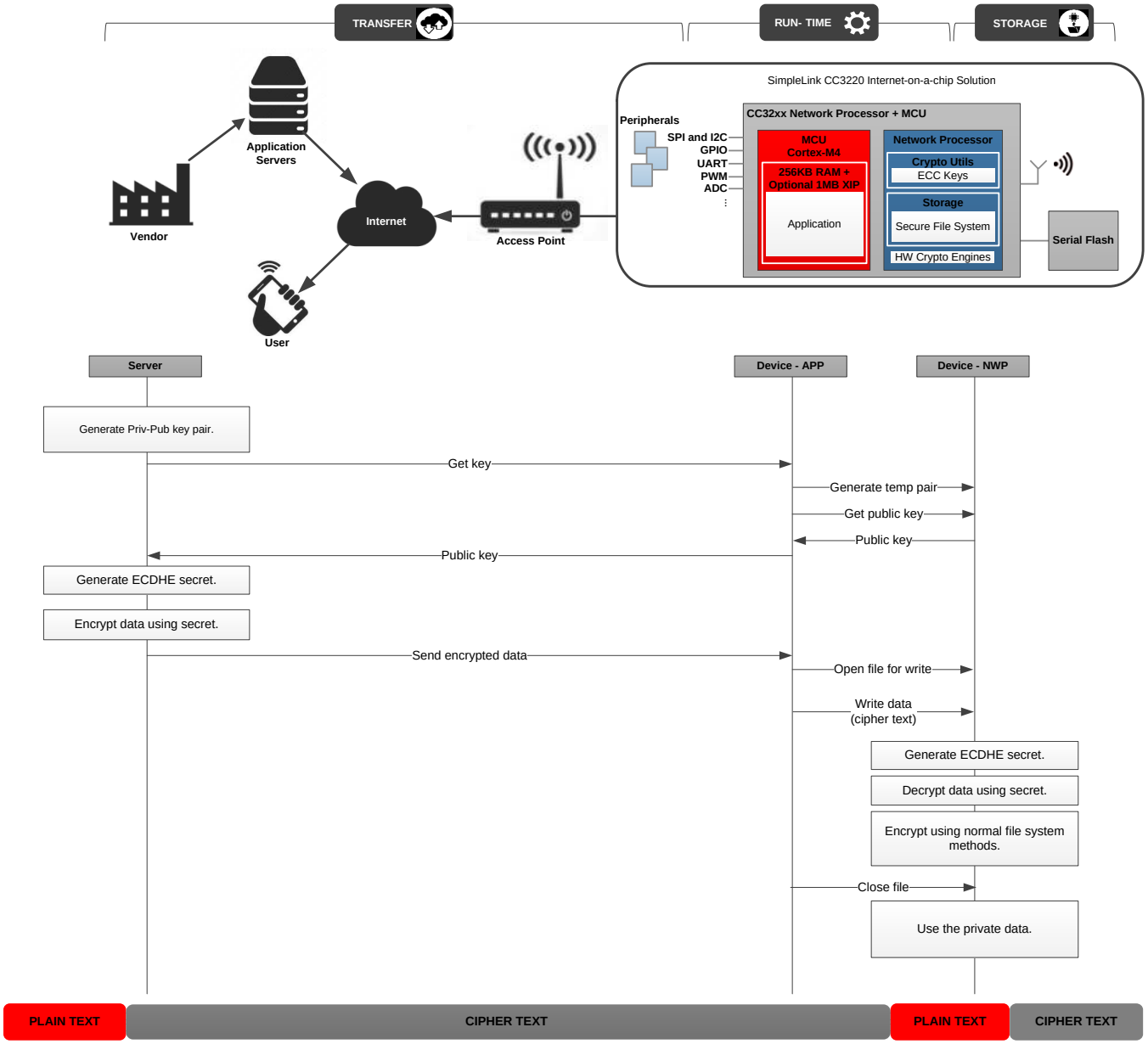

Figure 9. Secure Content Delivery – Process Description

5.2.2 Encrypted File Format

The file delivered with this process should have a special format. Figure 10 shows this format.

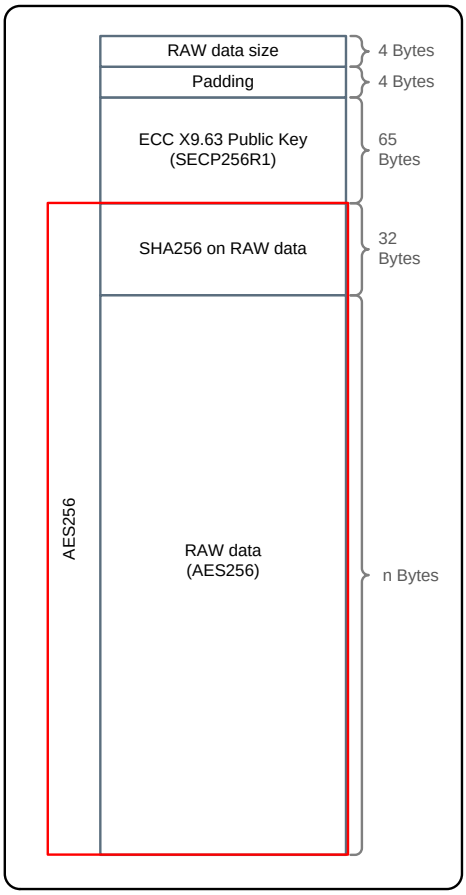

Figure 10. Secure Content Delivery – File Format

5.2.3 Further Details

- The temporary ECC key from the Networking subsystem is nonpersistent. The current temporary key becomes invalid when a new process is started, the SimpleLink device is restarted, or the host requested a new key. Once the key is cleared, there is no way to recover it and content with this key is not be allowed.
- The ECC key follows the SECP256R1 named curve.
- The AES key is generated using the ECDHE algorithm with the NWP public key and the encoder private key.
- The encrypted file write should be in a sequential order with no possibility for offset writing (the offset field in the write API is ignored).

6 Security for Application Layers

6.1 Secure Key Storage

The SimpleLink device supports on-chip asymmetric key-pair storage with built-in crypto acceleration and crypto services to assist in some common cryptographic-related operations.

These crypto services provide a mechanism to manage up to eight ECC key-pairs and use them to sign or verify data buffers. This capability could be used for authenticating the device identity, among other usage options.

There are three types of supported key-pairs:

- Device-unique key-pair: A single 256-bit unique key of the device, embedded in hardware
- Temporary key-pair: Created upon request using the internal TRNG engine
- Installed key-pair: Installed and maintained by the vendor

The system key storage comprises eight indexed key-pairs.

Index 0 is reserved for the device-unique key-pair which is hardware-based. Indexes 1 to 7 are for temporary or installed key-pairs, according to the application needs.

All keys are ECC keys using the SECP256R1 curve.

For all key pairs, the private key is not exposed and can only be applied indirectly when using it for signing and decrypt operations. The public key may be retrieved by the host application.

6.2 Hardware Crypto Engines (CC3220 only)

The SimpleLink CC3220 MCU includes a set of hardware crypto engines to enhance the performance of applications that require custom or application-level security. These engines offload the ARM® Cortex®-M4 MCU from the complex and time-consuming arithmetic involved in cryptographic algorithms.

The device includes the following hardware crypto engines:

- Advanced encryption standard (AES)
- Data encryption standard (DES)
- Secure hash algorithm/message digest algorithm (SHA/MD5)
- Cyclic redundancy check (CRC)

6.2.1 Advanced Encryption Standard (AES)

The AES module provides hardware-accelerated encryption and decryption operations based on a binary key. The AES is a block cipher module that supports 128-, 192-, and 256-bit keys for both encryption and decryption.

The module supports the following AES functionality:

- Basic AES encrypt and decrypt operations
- Key sizes: 128, 192, and 256 bits
- Key scheduling in hardware
- Feedback modes:
  - Electronic code book mode (ECB)
  - Cipher block chaining mode (CBC)
  - Counter mode (CTR)
  - Cipher feed back mode (CFB), 128-bit
  - F8 mode

- AES-XTS
- AES-GCM (using AES-CTR mode and GHASH)
- AES-CCM (using AES-CTR mode and AES-CBC-MAC)
- Authentication-only modes CBC-MAC, f9
- Basic GHASH operation (selecting no encryption)

6.2.2 Data Encryption Standard (DES)

The DES module provides hardware-accelerated data encryption and decryption functions. The DES can run either the single DES algorithm or the triple DES (TDES or 3DES) algorithm.

The DES algorithm generates block ciphers which operate on blocks of plaintext and cipher text. The DES has an 8-byte (64-bit) block size. The DES key consists of 64 binary digits, but only 56 bits are actually used directly by the algorithm. The other 8 bits may be used for error detection.

The triple DES is DES used three times in a row, using three keys, so that key length is effectively 168 bits. Each triple DES encrypt or decrypt operation is a compound of DES encrypt and decrypt operations.

The DES accelerator includes the following main features:

- DES encryption and decryption
- Triple DES (or 3DES) encryption and decryption
- Feedback modes: ECB, CBC, CFB

6.2.3 Secure Hash Algorithm/Message Digest Algorithm (SHA/MD5)

SHA/MD5 is a hardware cryptographic accelerator used for hashing and message authentication.

This module can run the following algorithms:

- MD5 message digest algorithm
- SHA-1 algorithm
- SHA-2 (SHA-224 and SHA-256)
- Hash message authentication code (HMAC) operation using MD5, SHA-1, and SHA-2

The algorithms produce a condensed representation of a message or a data file, called digest or signature, which can then be used to verify the message integrity.

- MD5 hash
- SHA-1, SHA-224, or SHA-256 hash algorithm
- Automatic HMAC key preprocessing for HMAC keys up to 64 bytes
- Host-assisted HMAC key preprocessing for HMAC keys larger than 64 bytes

6.2.4 Cyclic Redundancy Check (CRC)

CRC is an error-detection-code commonly used with data transfers and safety system checks, and also in conjunction with AES and DES.

The CRC module supports the following features:

- Supports major CRC polynomials
  - CRC-16-IBM
  - CRC-16-CCITT
  - CRC-32
  - CRC-32C
  - TCP checksum
- Supports half word and byte swapping
- Allows byte and word feed

7 Runtime Binary Protection

The Simplelink CC3220S and CC3220SF devices provide measures for protecting the host application runtime binary by leveraging the security features of the file system.

7.1 CC3220S

On the SimpleLink CC3220S device, the application runtime binary is stored on an external serial flash. This file is loaded upon boot to internal SRAM and executed from there.

The file is stored in the file system as a secure file with public write permissions and fail-safe attribute set. As such, all properties of a secure file apply to it. This includes encryption and the authenticity check.

The authenticity is achieved based on the trusted root-certificate catalog, and validated during programming. If the signature of an image is invalid, the programming fails.

7.2 CC3220SF

On the SimpleLink CC3220SF device, the application runtime binary can be stored on an on-chip flash and directly executed from it (as with XIP). The device manages the on-chip flash internally, and external direct access is prohibited unless development mode is applied.

During programming, the image is first downloaded onto the serial flash (to a predefined filename with the name /sys/mcuflashimg.bin), where it is kept signed and encrypted. Upon first boot, the bootloader reads the encrypted image from serial flash, then decrypts it and writes it onto the on-chip flash.

To ensure complete image integrity, a hash of the application binary is generated and stored on the file system (as a secure system file). This hash is used during boot time to verify that the image stored on the on-chip flash corresponds with the image that was properly programmed. The hash makes a link between the image on the on-chip flash and the serial flash content. Changes in the serial flash after the image is copied onto the on-chip flash are detected based on this hash and regarded as a tamper attempt. In this case, the on-chip flash is erased.

The device supports a maximum size of 1022KB for the runtime binary. The application binary must be supplied, with a signature, to the image creation tool. The authenticity is achieved based on the certificate store capability, and validated during programming. If the signature of an image is invalid, the programming fails.

Figure 11 captures a simplified view of the image transfer process, from the serial flash to the on-chip flash in a CC3220SF device.

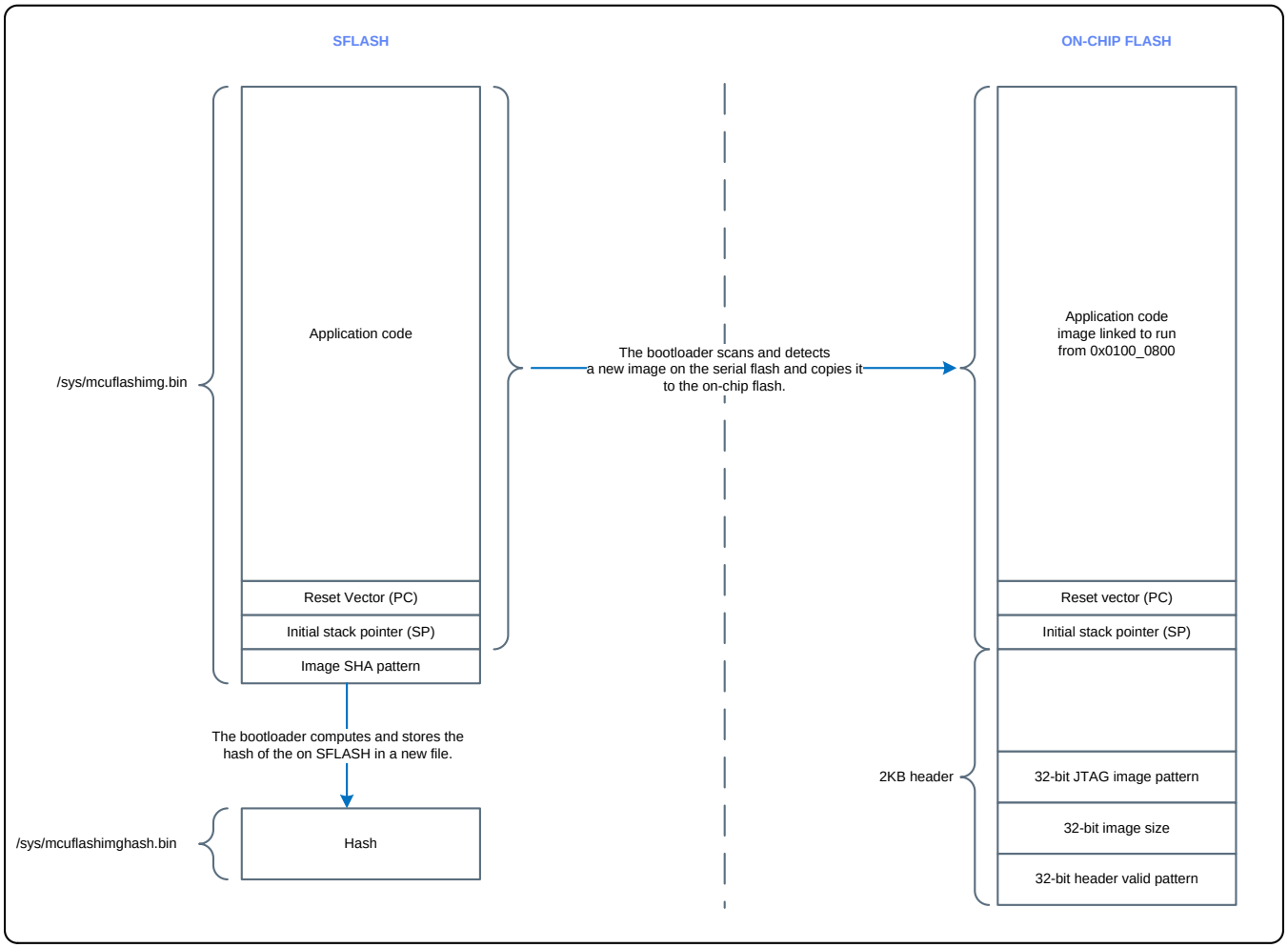

Figure 11. On-Chip Flash Programming and Update

Design for Security

A.1 Closing Remarks

Internet connectivity is often considered protected if the correct network configuration or firewalls are used. This assumption is wrong and far from reality. The problem is that even if the network configuration or firewalls block the access to the devices on the network level, or block some protocols and ports, the internet connectivity can still exploit some application-level vulnerabilities (for example, by sniffing the traffic that passes through the firewalls and configuration pages).

Protection based on firewalls or installation policies alone can lead to security breaches when the instructions are not entirely clear, or when other devices in the network require looser security settings.

In the IoT era, basing the protection on these kinds of policies alone is even worse, because most of the installations would likely be performed by unprofessional end users. This does not mean that this network configuration is not required; however it does mean that the protection cannot rely on this alone, and must also be designed in other layers.

In the last few years, the situation has become even worse, because most of these devices also enable remote software updates (in the common terminology: OTA updates). These updates put at risk not only the specific device that is updated, but also could give malicious users access to the local network. When an attacker succeeds in replacing the software of an internet-enabled device (even a device that seems immaterial or insignificant in terms of privacy and security), the attacker gains access to the local network, and therefore to other devices on the local network. This kind of attack should be considered part of the local network security assessment.

The common perception is that the local network connectivity is safe: only permitted devices or users are connected to this network, and there is a network boundary to the internet. In reality, the major vulnerability of LANs is a poor network security configuration. The availability of Wi-Fi networks among home users and organizations make this kind of poor network security configuration much worse, because they allow everyone access the local network without having a physical connection to an outlet inside the home or office. The only requirement for the malicious user is to have a reasonable distance to this area (tenths of meters).

The LAN provides access to connected devices, ports, and protocol, which are mostly blocked from the WAN or internet and by that provide much wider exposure points. A secured networking application should consider all these vulnerabilities, and mitigate them with other layers of security.

In general, the overall OTA application firmware update process is outside of the scope of this document and must be defined by the OEM. However, TI strongly recommends that the host application verifies the software update origin by supporting the following:

- Verify the domain name of the OTA server.
- Authenticate the OTA server.
- Use secure tunnel (such as SSL/TLS) for content delivery.

To summarize, TI recommends following these designed-for-security best practices:

1. Identify your assets.
2. Identify possible vulnerabilities.
3. Protect what must be protected.
4. Use only well-known security practices.
5. Disable all unused services and functionality (for example, the HTTP server)

Revision History

NOTE: Page numbers for previous revisions may differ from page numbers in the current version.

Changes from Original (February 2017) to A Revision	Page
• Changed Security Measure image	6
• Added CC3220 Security Features image	7
• Added CC3120 Security Features image	8
• Changed Section 3.2.3 text	17
• Changed Section 3.2.5 text	18
• Changed Secure content delivery process image	24

IMPORTANT NOTICE FOR TI DESIGN INFORMATION AND RESOURCES

Texas Instruments Incorporated ('TI") technical, application or other design advice, services or information, including, but not limited to, reference designs and materials relating to evaluation modules, (collectively, "TI Resources") are intended to assist designers who are developing applications that incorporate TI products; by downloading, accessing or using any particular TI Resource in any way, you (individually or, if you are acting on behalf of a company, your company) agree to use it solely for this purpose and subject to the terms of this Notice.

TI's provision of TI Resources does not expand or otherwise alter TI's applicable published warranties or warranty disclaimers for TI products, and no additional obligations or liabilities arise from TI providing such TI Resources. TI reserves the right to make corrections, enhancements, improvements and other changes to its TI Resources.

You understand and agree that you remain responsible for using your independent analysis, evaluation and judgment in designing your applications and that you have full and exclusive responsibility to assure the safety of your applications and compliance of your applications (and of all TI products used in or for your applications) with all applicable regulations, laws and other applicable requirements. You represent that, with respect to your applications, you have all the necessary expertise to create and implement safeguards that (1) anticipate dangerous consequences of failures, (2) monitor failures and their consequences, and (3) lessen the likelihood of failures that might cause harm and take appropriate actions. You agree that prior to using or distributing any applications that include TI products, you will thoroughly test such applications and the functionality of such TI products as used in such applications. TI has not conducted any testing other than that specifically described in the published documentation for a particular TI Resource.

You are authorized to use, copy and modify any individual TI Resource only in connection with the development of applications that include the TI product(s) identified in such TI Resource. NO OTHER LICENSE, EXPRESS OR IMPLIED, BY ESTOPPEL OR OTHERWISE TO ANY OTHER TI INTELLECTUAL PROPERTY RIGHT, AND NO LICENSE TO ANY TECHNOLOGY OR INTELLECTUAL PROPERTY RIGHT OF TI OR ANY THIRD PARTY IS GRANTED HEREIN, including but not limited to any patent right, copyright, mask work right, or other intellectual property right relating to any combination, machine, or process in which TI products or services are used. Information regarding or referencing third-party products or services does not constitute a license to use such products or services, or a warranty or endorsement thereof. Use of TI Resources may require a license from a third party under the patents or other intellectual property of the third party, or a license from TI under the patents or other intellectual property of TI.

TI RESOURCES ARE PROVIDED "AS IS" AND WITH ALL FAULTS. TI DISCLAIMS ALL OTHER WARRANTIES OR REPRESENTATIONS, EXPRESS OR IMPLIED, REGARDING TI RESOURCES OR USE THEREOF, INCLUDING BUT NOT LIMITED TO ACCURACY OR COMPLETENESS, TITLE, ANY EPIDEMIC FAILURE WARRANTY AND ANY IMPLIED WARRANTIES OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE, AND NON-INFRINGEMENT OF ANY THIRD PARTY INTELLECTUAL PROPERTY RIGHTS.

TI SHALL NOT BE LIABLE FOR AND SHALL NOT DEFEND OR INDEMNIFY YOU AGAINST ANY CLAIM, INCLUDING BUT NOT LIMITED TO ANY INFRINGEMENT CLAIM THAT RELATES TO OR IS BASED ON ANY COMBINATION OF PRODUCTS EVEN IF DESCRIBED IN TI RESOURCES OR OTHERWISE. IN NO EVENT SHALL TI BE LIABLE FOR ANY ACTUAL, DIRECT, SPECIAL, COLLATERAL, INDIRECT, PUNITIVE, INCIDENTAL, CONSEQUENTIAL OR EXEMPLARY DAMAGES IN CONNECTION WITH OR ARISING OUT OF TI RESOURCES OR USE THEREOF, AND REGARDLESS OF WHETHER TI HAS BEEN ADVISED OF THE POSSIBILITY OF SUCH DAMAGES.

You agree to fully indemnify TI and its representatives against any damages, costs, losses, and/or liabilities arising out of your non-compliance with the terms and provisions of this Notice.

This Notice applies to TI Resources. Additional terms apply to the use and purchase of certain types of materials, TI products and services. These include; without limitation, TI's standard terms for semiconductor products http://www.ti.com/sc/docs/stdterms.htm), evaluation modules, and samples (http://www.ti.com/sc/docs/sampterms.htm).

Mailing Address: Texas Instruments, Post Office Box 655303, Dallas, Texas 75265