Application Note

Integration of MbedTLS on Sitara MCU Devices

Shaunak Deshpande

ABSTRACT

TI Sitara™ MCU devices have a strong networking stack and hardware IP support that is already being utilized widely in the industry. Connectivity with external world is a risky proposition if there is no promised security in the network. While the networking capabilities and hardware IPs are gracefully evolving, becoming more efficient and optimized, the security aspect cannot be overlooked. Lack of security can lead to improper functioning of the system or even make the environment prone to Cyber-attacks, for example, MITM, eavesdropping, tampering, or message forgery. This gap can be bridged by having Transport Layer Security (TLS). TLS is a cryptographic protocol that provides secure communication over the internet majorly through encrypting the data that is communicated. TLS ensures that a secure communication channel can be established between two or more entities in the network.

TLS protocol can be divided into two parts:

- Handshake Layer: This layer is responsible for performing a TLS handshake, changing the cipher spec after the entities have been verified and handshake has been completed.
- Record Layer: This layer is responsible for fragmentation, compression, Authentication, Encryption of data that is to be transferred.

This document introduces integration of MbedTLS applicable to Sitara MCU devices over the existing LwIP TCP/IP networking stack, adding security at the Transport layer (L4 of the OSI model). Transport layer provides a secure end-to-end communication channel. As a result, all the data passing through the network after Transport Layer (layer 4) can be securely transferred. The MbedTLS project was ported over as an independent library to TI architecture, measure the performance, and ways of optimizing the cryptographic operations via hardware acceleration. The use of MbedTLS in network security examples is also discussed.

The code and examples discussed in this document can be found in TI MCU_PLUS_SDK v09.00 or later for AM243x, AM263x, AM273x, AM64x devices.

Table of Contents

List of Figures

List of Tables

Trademarks

Sitara™ is a trademark of Texas Instruments.
All trademarks are the property of their respective owners.

1 Introduction

Industry 4.0 is data intense and relies on real time decision making and applications need a mechanism to securely transmit the data across the network. This demonstration is an advancement in the networking and security domains for the TI Sitara MCU devices.

This application note addresses the following:

- Educate users about MbedTLS
- Demonstrate how MbedTLS is coupled with LwIP
- Exemplify network security built over MbedTLS

With high performance multi-core processing power, the TI Sitara MCU device is designed for real time processing and connectivity. The integration of MbedTLS over LwIP aims to further strengthen the networking and connectivity by adding a layer of security. LwIP is a lightweight TCP/ IP stack commonly used in embedded systems. The devices have CPSW and ICSS IPs for networking. The devices with its raw processing power and ability to control signals real time, with support of multi-protocol Ethernet standards, networks can operate at speeds as high as 1 Gbps. They also possess a strong crypto accelerator to offload cryptography from software to hardware, further optimizing the performance of the overall application.

1.1 Acronyms Used in This Document

1. **AES**: Advanced Encryption Standard
2. **CA**: Certificate Authority
3. **CPSW**: Common Platform Ethernet Switch
4. **DER**: Distinguished Encoding Rules
5. **DES**: Data Encryption Standard
6. **ECDSA**: Elliptic Curve Digital Signature Algorithm
7. **FOTA**: Firmware Over the Air
8. **HTTP**: Hyper Text Transfer Protocol
9. **ICSS**: Industrial Communication Subsytem
10. **MITM:** Man In The Middle attack
11. **MQTT**: Message Queuing Telemetry Transport
12. **OSI:** Open System Interconnection
13. **PBUF**: Packet Buffer
14. **PCB**: Protocol Contol Block
15. **PEM**: Privacy Enhanced Mail
16. **RSA**: Rivest-Shamir-Adleman
17. **SHA**: Secure Hash Algorithm
18. **SSL**: Secure Socket Layer
19. **TCP:** Transmission Control Protocol
20. **TLS:** Transport Layer Security

2 MbedTLS

2.1 What is MbedTLS?

MbedTLS is an implementation of the SSL and TLS protocols, along with the necessary support code and corresponding cryptographic algorithms. MbedTLS is distributed under the Apache License, version 2.0. MbedTLS provides an abstraction layer for secure communication over the TCP/IP stack as shown in Figure 2-1. The Software stack needed for MbedTLS and LwIP network security examples.

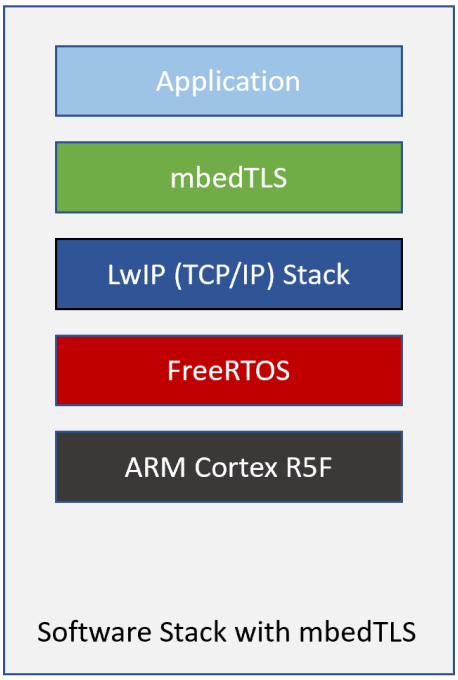

Figure 2-1. MbedTLS Secure Communication Over TCP/IP Stack

2.2 Why MbedTLS?

Alternatives to MbedTLS include OpenSSL, GnuTLS, LibreSSL, but TI prefers the usage of MbedTLS as MbedTLS has direct support in LwIP. MbedTLS can be ported over LwIP with minimal changes in existing LwIP configurations.

MbedTLS builds as an independent library which has hooks in LwIP for application layer APIs, which is discussed in detail in the next section. With detailed documentation and easy to use API, MbedTLS is a quick and efficient way to include security in your applications. MbedTLS APIs not only can be used in adding security to network communications but can be used across applications which need cryptographic implementations.

Moreover, MbedTLS provides a well-defined approach to offload the cryptographic operations from software to hardware. To note down a few, AES, SHA, RSA, ECDSA, ENTROPY, Timing functions, DES, ARIA, and so forth. can be offloaded to hardware. MbedTLS also provides an option to offload a subpart of cryptography, for example, offloading encryption to hardware and performing decryption on software, increasing the overall flexibility of optimization. Depending on factors such as clock speed of the CPU, RAM execution, Cache usage and so forth, the throughput can be optimized. On testing the same for TI Sitara™ AM2434, there was a 8-10x increase in throughput on hardware cryptography accelerator for AES encryption and decryption as compared to performing software cryptography (Note: This is not the benchmark number for performance of the hardware offloaded cryptography, the performance numbers are implementation specific and can be further improved). MbedTLS configurations are controlled by a header file which can be used to enable or disable the desired Cryptographic modules or even the cipher suites. This can be used to reduce the code size of the MbedTLS library.

2.3 Application of MbedTLS

With MbedTLS, we add security over open world networking on TI devices. MbedTLS can be used wherever a TLS gap needs to be fulfilled as a part of networking. MbedTLS can be used to operate the device as TLS entities (Client or Server). On a broader note, from application point of view, MbedTLS functionality can be utilized in Firmware Over The Air (FOTA) updates. Other potential applications can include Remote Diagnostics, Vehicle health monitoring, Secure Shell and Secure Logging, Secure Information and Event Management, and so forth. MbedTLS is a potential use case in integration of OpenVPN.

3 MbedTLS Over Lwip

The lightweight TCP/IP (LwIP) stack is an open-source TCP IP stack used in embedded systems. The LwIP stack is designed to reduce memory usage and code size. LwIP follows a layered approach. The relaxed scheme for communication between layers is established by using a shared memory, which means the application process and the networking code can use the same memory or internal buffers. Internally packets are represented as Pbufs (Packet Buffers) in LwIP. These Pbufs contain the payload of the packet to be transmitted in the network. In a scenario where MbedTLS is not present, the Pbufs contain the actual application data that is to be transmitted. But in case of secure communication, this data needs to be secured by one of the many possible approaches such as encryption, hashing, and so forth.

The LwIP project already has inbuilt support for MbedTLS. With MbedTLS being enabled, the data which passes through the default TCP layer now takes an alternate route and is processed by an Alternate TCP/IP layer (also referred to as ALTCP layer). These layers have callback functions set. The callback functions determine the behavior of application on establishing connection, sending data, receiving data, closing connection, handling errors, and so forth. MbedTLS can be used with LwIP socket APIs, BSD Socket APIs, Netconn APIs. This broad usage is achieved by populating a general altcp_tls_config struct, which internally uses MbedTLS functions for cryptographic/SSL operations.

The struct altcp_tls_config holds state that is needed to create new TLS client or server connections. This TLS config is then directly passed to LwIP functions or other LwIP internal structures. For example, the data that would generally pass through a callback tcp_recv (in case of MbedTLS not being enabled) now passes through altcp_recv (in case of MbedTLS being enabled). These alternate implementations have additional code (by default inside lwip-stack). All the alternate layer implementations are defined in the files at the following path: *MCU_PLUS_SDK/source/networking/lwip/lwip-stack/src/apps/altcp_tls.*

The usage of MbedTLS with LwIP is controlled at the top level by a macro defined in lwip-config/lwipopts.h. Enabling MbedTLS here, enables ALTCP layer. When the LwIP library is then rebuilt (MbedTLS enabled) the ALTCP layer handles the data. The data passes through an Alternate TCP layer (altcp) when MbedTLS is enabled as demonstrated in Figure 3-1. The data passes through an Alternate TCP layer (altcp) when MbedTLS is enabled.

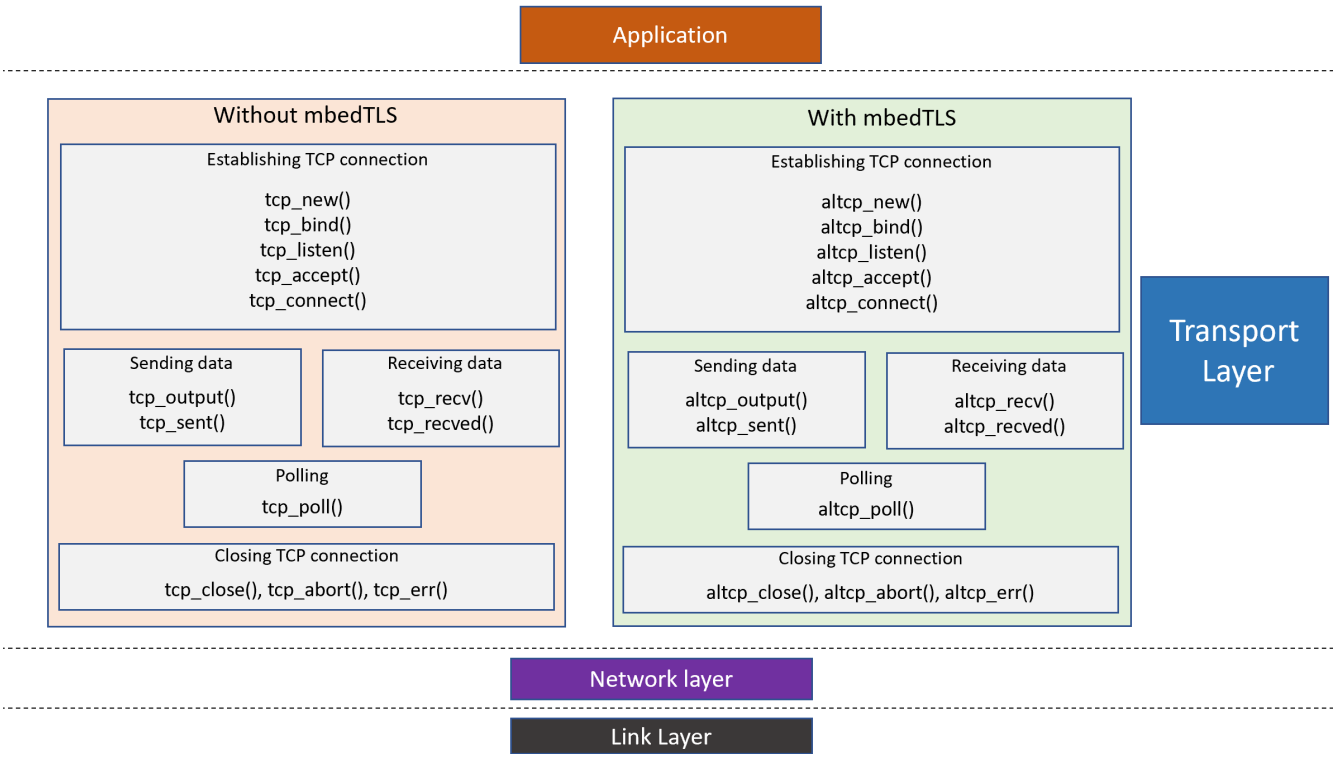

Figure 3-1. MbedTLS Functionality With LwIP

When the tcp_new() function is called, a new Protocol Control Block (TCP PCB) is created. This PCB is a new TCP connection identifier that can be either set to listen for new connection or explicitly connected to another host. In case of MbedTLS, an alternate PCB is used (altcp_pcb instead of tcp_pcb). The altcp_pcb structure contains altcp_functions, a reference to the next PCB, pointers to the arguments, state of the PCB, and application callbacks.

LwIP offers sample applications such as HTTP client and server, MQTT client, and so forth. The vanilla LwIP applications are not secured and the communication in the network is insecure, which in field would results in the device being an easy target for a cyberattack.

Having MbedTLS over LwIP ensures the data passing through a secured end to end transmission channel. MbedTLS performs a 3-way TLS handshake before the actual data is transmitted. TLS handshake ensures that the entities willing to communicate are authorized and trustable. TLS ensures encryption and integrity of data, and authentication of entities in the network. Furthermore, MbedTLS also provides 2-way SSL authentication as well if additional security is desired.

The change in memory footprint of LwIP after MbedTLS enablement is almost negligible. The memory footprint of MbedTLS itself in applications was observed to be approximately 176KB (which can further be reduced). Table 3-1 represents the memory footprint.

Table 3-1. MbedTLS Library Memory Footprint

Code	RO Data	RW Data	Total Size (bytes)
127988	39289	9159	176436

3.1 TLS Server Example (HTTPS Server)

This example demonstrates the AM2x device in the form of an HTTPS server which accepts client connections and sends a fixed response back. Here, MbedTLS was used with LwIP to attain the complete functionality.

A top-level working of HTTPS server is described below:

Role of MbedTLS:

- A TLS config using the server certificate, private key, password for accessing the private key (optional) is created. A standard allocator function creates an altcp PCB for TLS over TCP.
- The certificate and private key is loaded. Then based on the format of the x509 certificate (DER or PEM), the certificate and keys are parsed.
- The modulus of the private key is compared with the modulus of the public key present in the certificate. The certificate is signed with private key of trusted CA.
- The public key is used to verify the above signature. If the verification is complete, then next steps are executed.
- Pass the TLS config to LwIP application APIs, which internally use the same TLS config in PCB (Process Control Block of the corresponding TCP connection).
- Using mbedTLS cryptographic functions, certificates and keys are verified. If any parsing error or data inconsistency, certificate and keys invalidation occur, the ongoing network connection is dropped.

Role of LwIP:

- A LwIP PCB is set up and the PCB is then bind to the defined port (8080 in case of HTTPS). Then the PCB binds the connection to the local port number and IP address, which in this case is attained by a DHCP server.
- Then the state of TCP connection is set to LISTEN (mode to accept new connections).
- An altcp_accept callback is set, which handles new incoming connections, allocates memory to manage the connection state, sets callbacks for sending, receiving, error handling and polling.
- The http_recv callback on getting data, informs TCP PCB that data has been received, then parses the data.
- The http_poll callback polls the other side of the connection every 2 seconds, if data is not received for 8 seconds, the connection is closed.
- The http_err callback closes the connection and frees the resources when error emerges.
- The http_sent callback function is responsible for sending data and getting acknowledgment from the remote host.

Figure 3-2 demonstrates the same with SA2UL cryptography accelerator being used for cryptography (optional). The other alternative is to use MbedTLS software cryptography instead of offloading it to hardware.

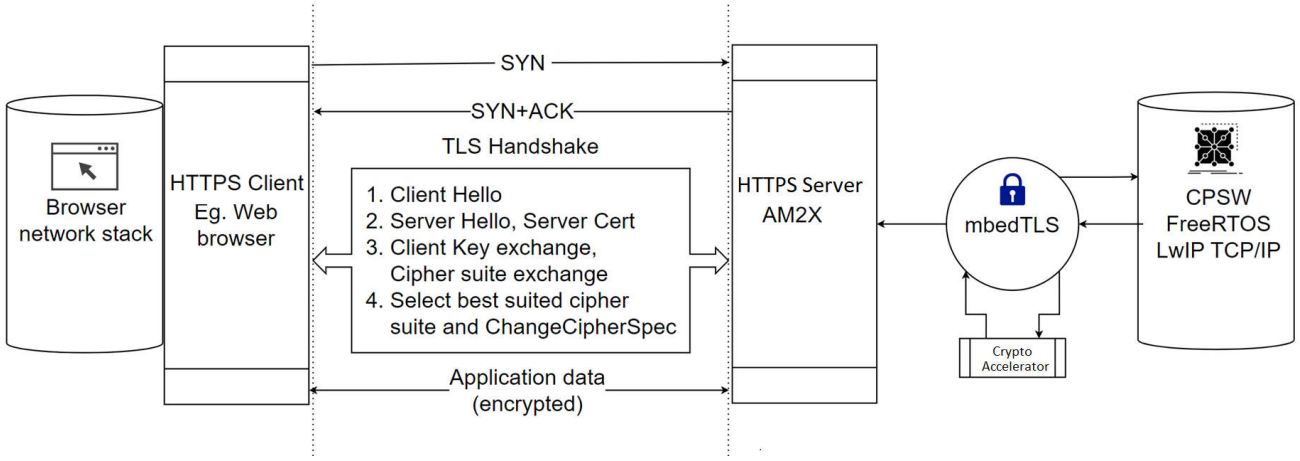

Figure 3-2. Overview of How HTTPS Server Example Works End to End

For further details, see: AM243x MCU+ SDK: CPSW Lwip HTTPS Server Example

3.2 TLS Client Example (MQTT Client)

This example demonstrates the AM2x device in the form of an MQTT client, subscribing to an MQTT broker and intercepting data when published by another client. The MQTT broker used in this case is an open-source software, Mosquito MQTT. The IP address assigned to the Mosquito broker and the MQTT client is static.

In this example, a 2-way mutual authentication is performed, which involves the client verifying the server and the server verifying the client as well. Since locally generated and self-signed certificates are used, CA certificates and CA keys are used as server certificates and keys.

A top-level working of MQTT + TLS client is described below:

Role of MbedTLS:

- A TLS config using the CA certificate, CA key, client certificate, private key, password for accessing the private key (optional) is created. A standard allocator function creates an altcp PCB for TLS over TCP.
- The client certificate and private key is loaded. Then based on the format of x509 certificate (DER or PEM), the certificate and keys are parsed.
- The private key is passed and compared with the public key present in the certificate. If they match, the certificate and keys/ keypair are verified.
- Pass the TLS config to LwIP application APIs, which internally use the same TLS config in PCB (Process Control Block of the corresponding TCP connection).
- Using mbedTLS cryptographic functions, certificates and keys are verified. If any parsing error or data inconsistency, certificate and keys invalidation occur, the ongoing network connection is dropped.

Role of LwIP:

- The TLS config is populated inside the LwIP's mqtt_context structure.
- An altcp PCB is created for the defined MQTT port, IP address, and connection callbacks are set. The client information containing the TLS configuration is passed.
- An altcp_accept callback is set, which handles new incoming connections, allocates memory to manage the connection state, sets callbacks for sending, receiving, error handling and polling.
- The mqtt connection callback is responsible for subscribing/unsubscribing to a topic, sending the required data about the same to the MQTT broker.
- Input callback functions are set which determine the behavior of client when data is received.
- Based on the state of the connection, the data is processed.

Figure 3-3 demonstrates the same with SA2UL cryptography accelerator being used for cryptography (optional).

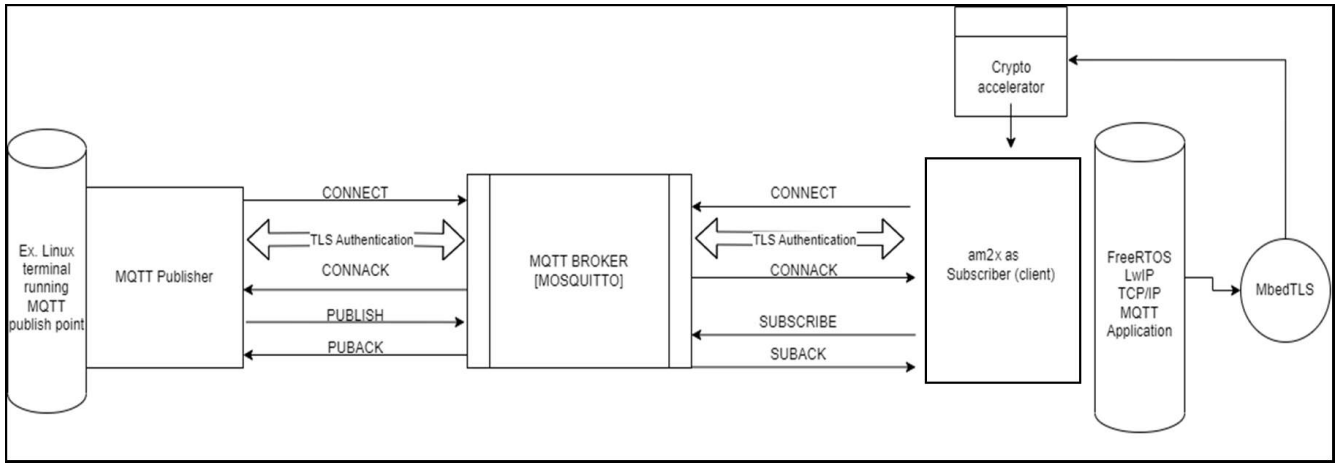

Figure 3-3. Overview of Working of MQTT + TLS Client

IMPORTANT NOTICE AND DISCLAIMER

TI PROVIDES TECHNICAL AND RELIABILITY DATA (INCLUDING DATA SHEETS), DESIGN RESOURCES (INCLUDING REFERENCE DESIGNS), APPLICATION OR OTHER DESIGN ADVICE, WEB TOOLS, SAFETY INFORMATION, AND OTHER RESOURCES “AS IS” AND WITH ALL FAULTS, AND DISCLAIMS ALL WARRANTIES, EXPRESS AND IMPLIED, INCLUDING WITHOUT LIMITATION ANY IMPLIED WARRANTIES OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE OR NON-INFRINGEMENT OF THIRD PARTY INTELLECTUAL PROPERTY RIGHTS.

These resources are intended for skilled developers designing with TI products. You are solely responsible for (1) selecting the appropriate TI products for your application, (2) designing, validating and testing your application, and (3) ensuring your application meets applicable standards, and any other safety, security, regulatory or other requirements.

These resources are subject to change without notice. TI grants you permission to use these resources only for development of an application that uses the TI products described in the resource. Other reproduction and display of these resources is prohibited. No license is granted to any other TI intellectual property right or to any third party intellectual property right. TI disclaims responsibility for, and you will fully indemnify TI and its representatives against, any claims, damages, costs, losses, and liabilities arising out of your use of these resources.

TI’s products are provided subject to TI’s Terms of Sale or other applicable terms available either on ti.com or provided in conjunction with such TI products. TI’s provision of these resources does not expand or otherwise alter TI’s applicable warranties or warranty disclaimers for TI products.

TI objects to and rejects any additional or different terms you may have proposed.

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