Aerospace & Defence | Lighting | Power Management | Internet of Things | Transportation

Components – EMEA

Engineering DACH

TrustZone and ATECC508 on SAML11

Quang Hai Nguyen
Revision 1.0, 06.06.2019

Hands-on

Five Years Out

Revision History

Revision, Date	Editor	Subject (major changes)
Revision 0.1, 03.06.2019	Quang Hai Nguyen	Preliminary
Revision 1.0, 06.06.2019	Quang Hai Nguyen	Release

Table of Contents

Revision History 3
Table of Contents 4
List of Abbreviations 5
List of Figures 6
List of Icon Identifiers 7
Overview 8
  Introduction 8
    Arm TrustZone 8
    ATECC508 8
  Description 9
  Assignments 9
  Goal 9
Requirement 10
  Hardware 10
  Software 10
Assignments 10
  Assignment 1: import the project and explore the project configuration 10
  Assignment 2: Adding the code in secure world 14
    TODO 1 – Secure functions header 14
    TODO 2 – secure functions body 16
    TODO 3 – veneer functions header 20
    TODO 4 – veneer functions body 21
    TODO 5 – secure app, declaration 22
    TODO 6 - Initialize non-secure application 22
  Assignment 3: Adding the code in non-secure world 22
    TODO 7 – non-secure app, declaration 23
    TODO 8 – non-secure, main function body 23
  Assignment 4: Testing the application 24

List of Abbreviations

TZ
TrustZone 9

List of Figures

Figure 1: Use case setup....8
Figure 2: Use case Diagram....9
Figure 3: AtmelStudio....10
Figure 4: Project configuration in secure world....12
Figure 5: Project configuration in non-secure world....13
Figure 6: Task list of the hands-on....14
Figure 7: Authenticated and call secure API....26
Figure 8: Calling secure API without a secure element....27

List of Icon Identifiers

Table 1: Icon Identifiers List

ℹ	Extra information about a topic
✎	Task need to be done
⚠	Important information or a warning
✓	Result expected to see

Overview

Introduction

This hands-on will show you how to create a secure application using Secure Element and TrustZone on SAML11 microcontroller.

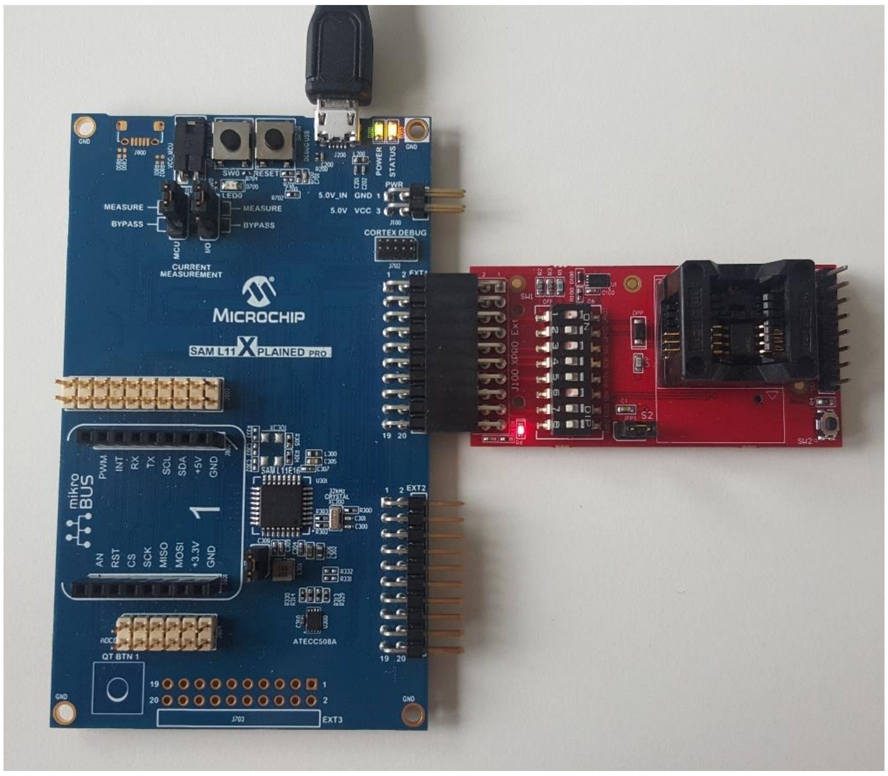


Figure 1: Use case setup

Arm TrustZone

TrustZone provides the flexibility for hardware isolation of memories and peripherals, therefore reinforcing the ability of Intellectual Properties (IP) and Data protection. SAML11 provides up to six regions for the Flash, up to two regions for Data Flash, up to two regions for SRAM and the ability to assign peripherals, I/O pins, interrupts to secure or non-secure application.

ATECC508

The Microchip ATECC508A integrates ECDH (Elliptic Curve Diffie Hellman) security protocol an ultra-secure method to provide key agreement for encryption/decryption, along with ECDSA (Elliptic Curve Digital Signature Algorithm) sign-verify authentication for the Internet of Things (IoT) market including home automation, industrial networking, accessory and consumable authentication, medical, mobile and more.

Description

Inside SAML11, there are two application running, which are the secure and non-secure application. When the non-secure application tries to call the API in TrustZone area, the secure application checks if the non-secure one is already authenticated and allows the API to run or showing error message.

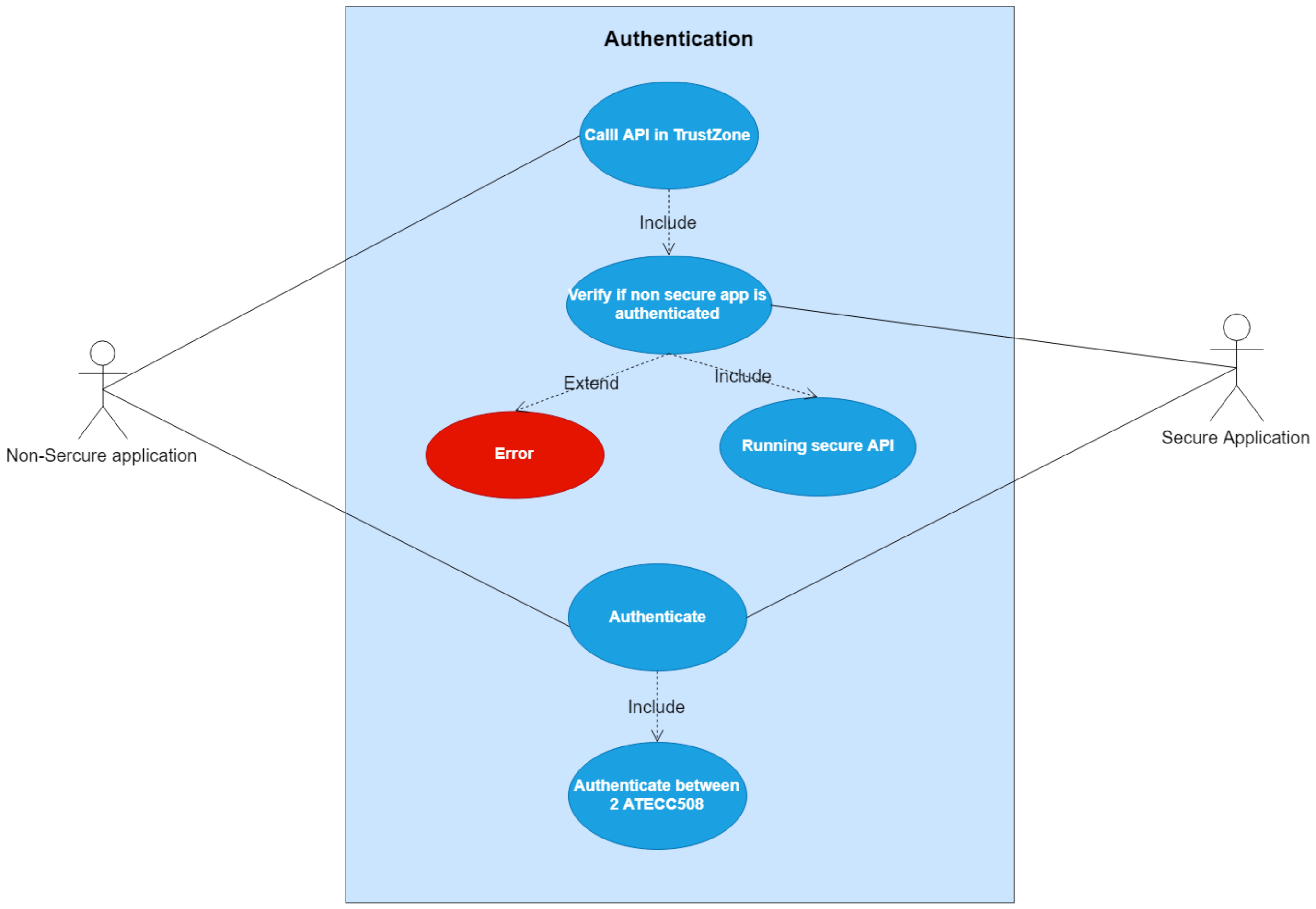


Figure 2: Use case Diagram

The non-secure application is initialized by the secure application. The non-secure application can access to the API by first authenticating to the secure application. The secure application carries out the authentication process and return the status. If the authentication is successful, the non-secure application is allowed to access the API.

Assignments

- Assignment 1: import the project and explore the project configuration
- Assignment 2: Adding the code in secure world
- Assignment 3: Adding the code in non-secure world
- Assignment 4: Testing the application

Goal

After this hands-on, you will know the benefits of using TZ and Secure Element for a secure application. In addition, you will have the confidence to demonstrate the hands-on to customers.

Requirement

Hardware

- SAML11 Xplained
- ATECC508 with adapter
- Type A-to-micro USB cable

Software

- Atmel Studio version 7
- TeraTerm

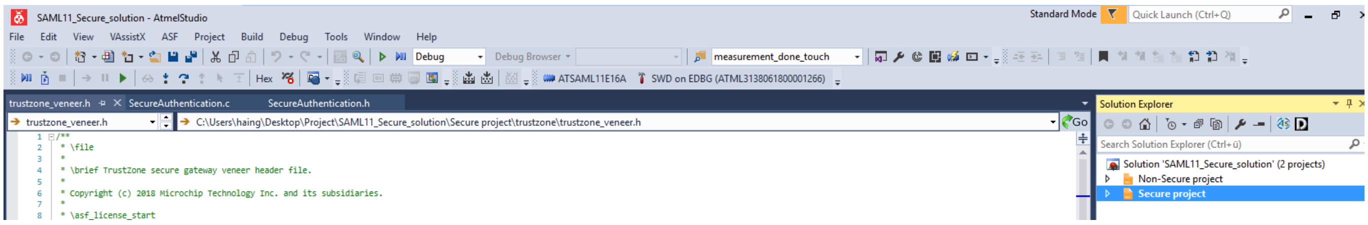
The software is already installed if you are using the virtual machine. Otherwise please refer to the Installation Guide for more information on how to get these programs.

Assignments

Assignment 1: import the project and explore the project configuration

Open the project with AtmelStudio

Navigate to the project folder and double click on the .atsln file. After clicking AtmelStart starts automatically.


SAML11_Secure_solution - AtmelStudio

trustzone_veneer.h — C:\Users\haing\Desktop\Project\SAML11_Secure_solution\Secure project\trustzone\trustzone_veneer.h

```
/**
 * \file
 *
 * \brief TrustZone secure gateway veneer header file.
 *
 * Copyright (c) 2018 Microchip Technology Inc. and its subsidiaries.
 *
 * \asf_license_start
```

Solution 'SAML11_Secure_solution' (2 projects)
- Non-Secure project
- Secure project

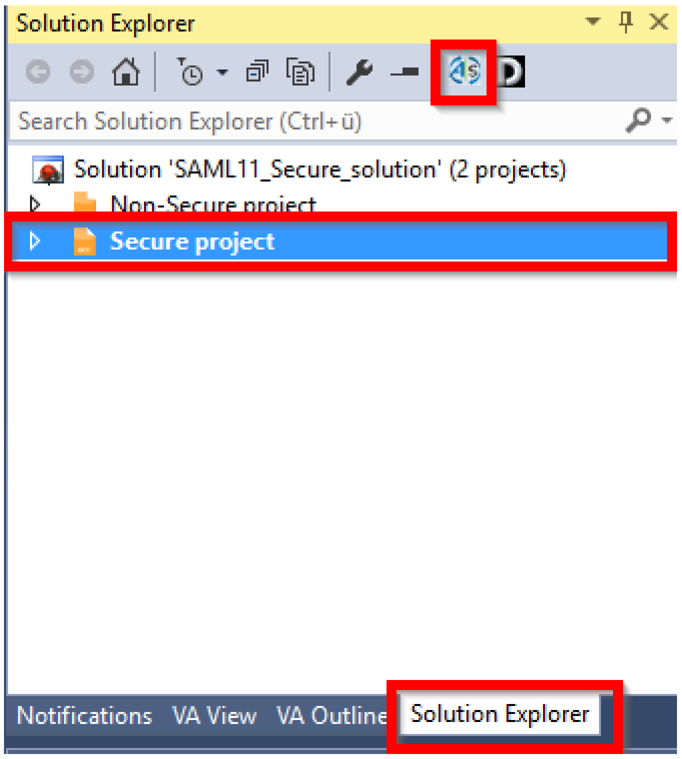
Figure 3: AtmelStudio

Alternatively, you can start AtmelStudio first, then choose File → Open → Project/Solution → point to .atsln file

Because the application runs on both secure and non-secure world, you will see there are two projects in the solution, which are Secure Project and Non-secure Project

In the solution explorer, focus on Secure project and choose AtmelStart icon

AtmelStart is an online tool so please make sure that you have the internet connection



If AtmelStart started correctly, the project configuration should be as following:

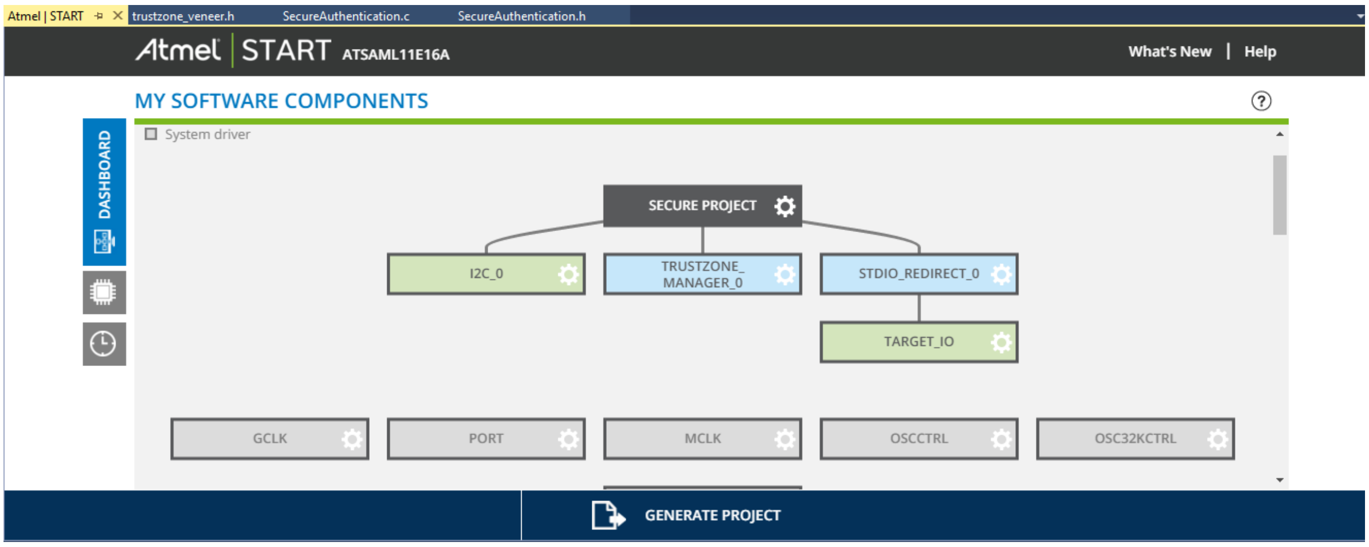
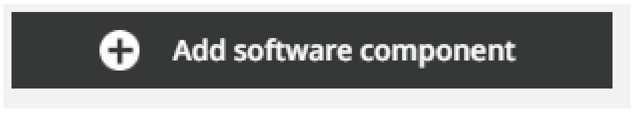


Figure 4: Project configuration in secure world

The project includes three components I2C, TrustZone manager, and stdio redirect. TrustZone manager initializes the memory zone of the microcontroller. Stdio redirect allows routing debug information to the terminal on the PC. I2C block handles the communication between the SAML11 Xplained and the ATECC508 on board.

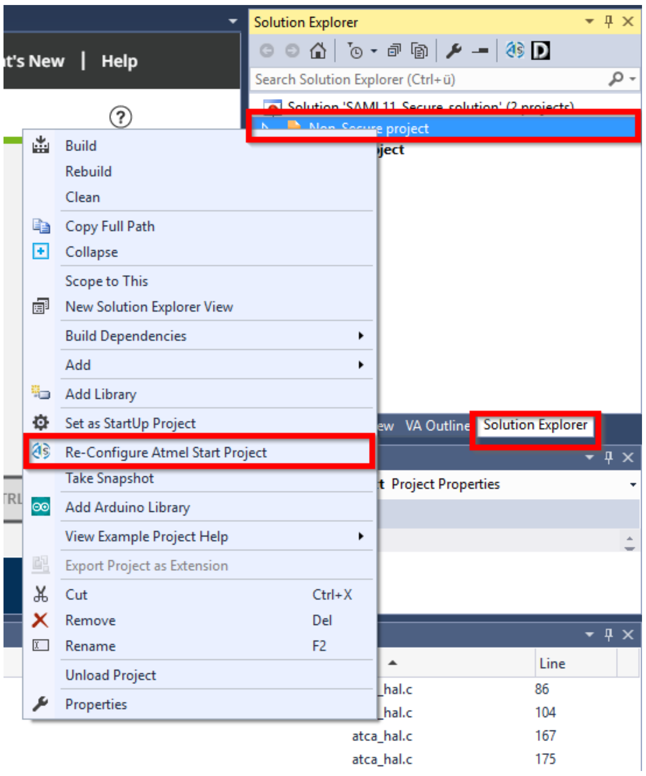
Those components can be added by click at Add software components button



You are more than welcome to explore the detail of each component by clicking on it.

Please do not modify the configuration

The same procedure is applied to the Non-Project. In the solution explorer, right click at Non-Secure project and choose Re-configure Atmel Start project



If AtmelStart started correctly, the project configuration should be as following:

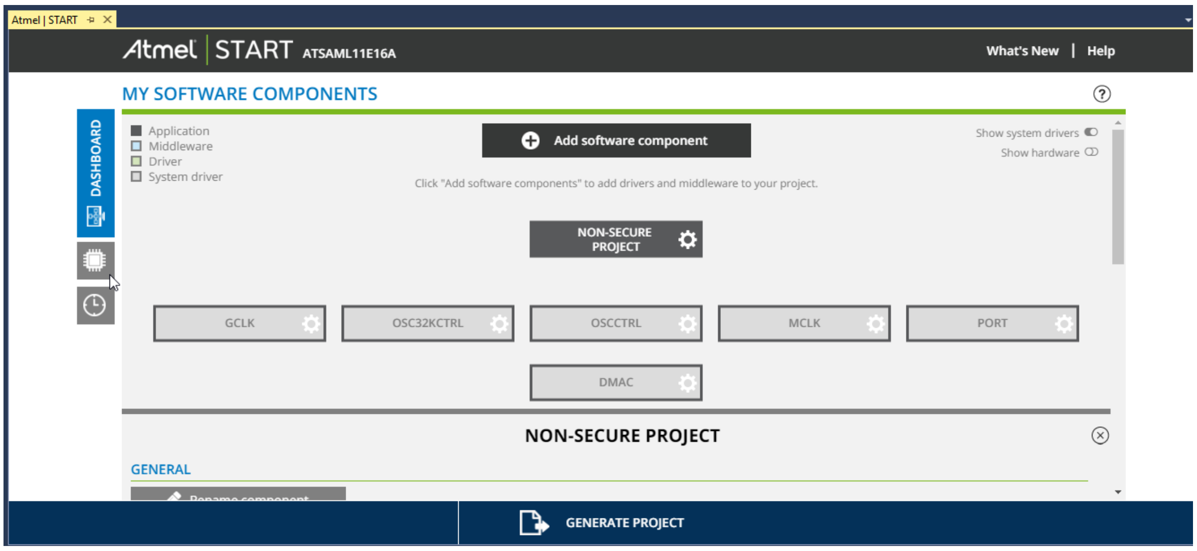


Figure 5: Project configuration in non-secure world

The software components for the non-secure world are empty because they are initialized by the secure world. Non-secure world is only allowed to call those components through defined functions in the veneer table.

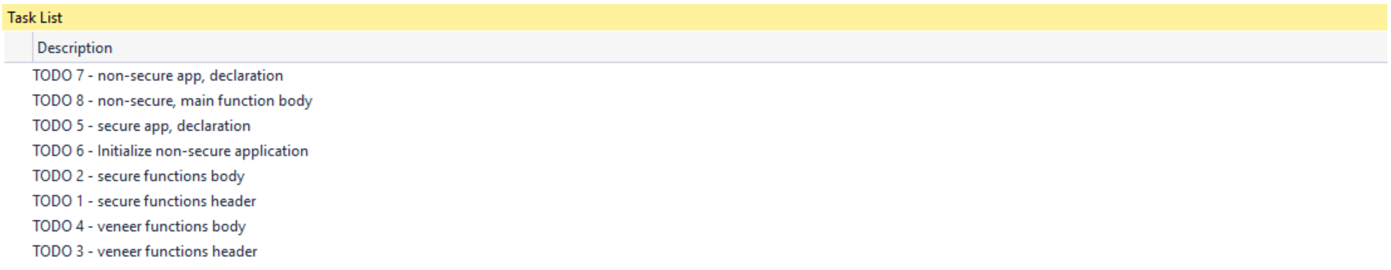
You are more than welcome to explore the detail of each component by clicking on it.

Please do not modify the configuration

Compile the project and see task list

To compile the project, please click Build → Build Solution or simply press F7

After compiling the project, click View → Task List to see the tasks need to be done for this hands-on. Those Tasks will be discussed in detail in the next assignment



```
Task List
Description
TODO 7 - non-secure app, declaration
TODO 8 - non-secure, main function body
TODO 5 - secure app, declaration
TODO 6 - Initialize non-secure application
TODO 2 - secure functions body
TODO 1 - secure functions header
TODO 4 - veneer functions body
TODO 3 - veneer functions header
```

Figure 6: Task list of the hands-on

Assignment 2: Adding the code in secure world

If you get stuck at any point, there is a solution available. But please try it on your own before using the solution. Learning by doing 😉.

Please refer the text file for easily copy paste the code

Modify the code in SecureAuthentication.h

TODO 1 – Secure functions header

Double click on the TODO 1 in the Task list to jump to the code section. This section contains the functions called directly by the secure application or indirectly through a veneer table by the non-secure application.

Right below the TODO line, add the following code:

```
/**
 *  @brief Print text on the terminal
 *
 *          Text will be in green to indicate that it is coming from secure world
 *
 *          @paramstring     string to be displayed
 *
 *          @returnNULL      always return
 *
 *          @date   29.05.2019 - initial
 *
 *  @bug No known bugs.
 */
void secure_console_puts (uint8_t * string);


/**
 *  @brief Print text on the terminal
 *
 *          Text will be in red to indicate that it is coming from non-secure world
 *
 *          @paramstring     string to be displayed
 *
 *          @returnNULL      always return
 *
 *          @date   29.05.2019 - initial
 *
 *  @bug No known bugs.
 */
void non_secure_console_puts (uint8_t * string);

/**
 *  @brief Start authentication process
 *
 *          the underlying process is the challenge and
 *          response between 2 ATECC508
 *          the two ATECC508 are required to be provisioned before using
 *
 *
 *          @returnNULL      always return
 *
 *          @date   29.05.2019 - initial
 *
 *  @bug No known bugs.
 */
void SymmetricAuthentication(void);


/**
 *  @brief Return the authentication status
 *
 *          It allows the application in the non-secure world
 *          to check if it needs to run the authentication process
 *
 *
 *          @returntrue if it is already authenticated, otherwise false
 *
 *          @date   29.05.2019 - initial
 *
 *  @bug No known bugs.
 */
bool IsAuthenticated(void);



/**
 *  @brief Simulate the protected API
 *
 *          it is just a sum function. Before the summation, it verify if
 *          the authentication process is successful
 *
 *          @parama                 first element
 *          @paramb                 second element
 *          @paramsum               return the sum of a and b
 *
 *          @returnfunction status
 *
 *          @date   29.05.2019 - initial
 *
 *  @bug No known bugs.
 */
uint8_t APIInTrustZone(int * a, int * b, int * sum);
```

Please refer the header of each function to know more about each function

Modify the code in SecureAuthentication.c

TODO 2 – secure functions body

Double click on the TODO 2 in the Task list to jump to the code section.

Right below the TODO line, add the following code:

```
#define CHECK_STATUS(s)                                                         \
if(s != ATCA_SUCCESS) {                                                         \
            printf("status code: 0x%x\r\n", s);                                 \
            printf("Error: Line %d in %s\r\n", __LINE__, __FILE__);             \
            return;                                                             \
}                                                                               \



/* Local variable section --------------------------------------------*/
ATCAIfaceCfg cfg_ateccx08a_i2c_host = {
            .iface_type                         = ATCA_I2C_IFACE,
            .devtype                    = ATECC508A,
            .atcai2c.slave_address      = 0xC2,
            .atcai2c.bus                = 1,
            .atcai2c.baud                         = 400000,
            .wake_delay                                       = 800,
            .rx_retries                                       = 20,
            .cfg_data           = &I2C_0
};

ATCAIfaceCfg cfg_ateccx08a_i2c_remote = {
            .iface_type                                       = ATCA_I2C_IFACE,
            .devtype                              = ATECC508A,
            .atcai2c.slave_address      = 0xC0,
            .atcai2c.bus                          = 1,
            .atcai2c.baud                         = 400000,
            .wake_delay                                       = 800,
            .rx_retries                                       = 20,
            .cfg_data           = &I2C_0
};



/* Local function prototype section ----------------------------------*/

static void print_bytes(uint8_t * ptr, uint8_t length);
bool static authen_status = false;


static void print_bytes(uint8_t * ptr, uint8_t length)
{

            uint8_t i = 0;
            uint8_t line_count = 0;
            for(;i < length; i++) {
                        printf("0x%02x, ",ptr[i]);
                        line_count++;
                        if(line_count == 8) {
                                    printf("\r\n");
                                    line_count = 0;
                        }
            }

            printf("\r\n");
}

void secure_console_puts (uint8_t * string)
{
            /* Set display foreground color to green */
            printf("\033[0;32m");
            /* Print string on console */
            printf("%s", string);
}

void non_secure_console_puts (uint8_t * string)
{
            /* Set display foreground color to red */
            printf("\033[0;31m");
```

```
	/* Print string on console */
	printf("%s", string);
}

bool IsAuthenticated(void)
{
	return authen_status;
}


void SymmetricAuthentication(void)
{

	//Step 1.1
	/* Set display foreground color to green */
	printf("\033[0;32m");
	printf("Symmetric Authentication\r\n");

	printf("Authentication in progress\r\n");
	volatile ATCA_STATUS status;
	status = atcab_init( &cfg_ateccx08a_i2c_host ); CHECK_STATUS(status);
	printf("Host init complete\r\n");


	uint8_t serial_number[ATCA_SERIAL_NUM_SIZE];
	status = atcab_read_serial_number((uint8_t*)&serial_number);
	CHECK_STATUS(status);
	printf("Serial Number of host\r\n");
	print_bytes((uint8_t*)&serial_number, 9); printf("\r\n");


	uint8_t nonce[32];
	status = atcab_random((uint8_t*)&nonce);
	CHECK_STATUS(status);
	printf("Random from host\r\n");
	print_bytes((uint8_t*)&nonce, 32);


	status = atcab_init( &cfg_ateccx08a_i2c_remote );
	CHECK_STATUS(status);

	status = atcab_read_serial_number((uint8_t*)&serial_number);
	CHECK_STATUS(status);

	printf("Serial Number of remote\r\n");
	print_bytes((uint8_t*)&serial_number, 9); printf("\r\n");


	uint8_t mac[32];
	uint8_t slot = 0; uint8_t mode = (1<<6); // include serial number
	status = atcab_mac(mode, slot, (const uint8_t*)&nonce, (uint8_t*)&mac);
	CHECK_STATUS(status);
	printf("MAC from remote\r\n");
	print_bytes((uint8_t*)&mac, 32);
	status = atcab_init( &cfg_ateccx08a_i2c_host );
	uint8_t otherdata[CHECKMAC_OTHER_DATA_SIZE];
	memset(otherdata, 0x00, CHECKMAC_OTHER_DATA_SIZE);
	otherdata[0] = 0x08;
	otherdata[1] = 0x40;
	otherdata[7] = serial_number[4];
	otherdata[8] = serial_number[5];
	otherdata[9] = serial_number[6];
	otherdata[10] = serial_number[7];
	otherdata[11] = serial_number[2];
	otherdata[12] = serial_number[3];
	mode = 0;

	status = atcab_checkmac(mode, slot, (const uint8_t*)&nonce, (const uint8_t*)&mac, (const
uint8_t*)&otherdata);

	if(status == ATCA_SUCCESS)
	{
		printf("Authenticated by host\r\n\r\n");
		authen_status = true;
	}
	else
	{
		printf("Failed to authenticate\r\n\r\n");
		authen_status = false;
	}
}

uint8_t APIInTrustZone(int * a, int * b, int * sum)
{

	if(authen_status)
	{
		*sum = *a + *b;
		return 0;
	}
	else
	{
		return 1;
	}

}
```

Modify the code in trustzone_veneer

TODO 3 – veneer functions header

Double click on the TODO 3 in the Task list to jump to the code section. This section contains the functions, which are called by the non-secure application.

Because of TrustZone technology design, non-secure world must and can only access any functions in the secure world through the veneer table.

Right below the TODO line, add the following code:

```
/**
 *  @brief Print text on the terminal
 *
 *          Text will be in red to indicate that it is coming from non-secure world
 *
 *          @paramstring     string to be displayed
 *
 *          @returnNULL      always return
 *
 *          @date    29.05.2019 - initial
 *
 *  @bug No known bugs.
 */
extern void nsc_non_secure_console_puts (uint8_t * string);

/**
 *  @brief Start authentication process
 *
 *          the underlying process is the challenge and
 *          response between 2 ATECC508
 *          the two ATECC508 are required to be provisioned before using
 *
 *
 *          @returnNULL      always return
 *
 *          @date    29.05.2019 - initial
 *
 *  @bug No known bugs.
 */
extern void nsc_SymmetricAuthentication (void);

/**
 *  @brief Return the authentication status
 *
 *          It allows the application in the non-secure world
 *          to check if it need to runs the authentication process
 *
 *
 *          @returntrue if it is already authenticated, otherwise false
 *
 *          @date    29.05.2019 - initial
 *
 *  @bug No known bugs.
 */
extern bool nsc_IsAuthenticated (void);

/**
 *  @brief Simulate the protected API
 *
 *          it is just a sum function. Before the summation, it verify if
 *          the authentication process is successful
 *
 *          @parama                 first element
 *          @paramb                 second element
 *          @paramsum               return the sum of a and b
 *
 *          @return function status
 *
 *          @date   29.05.2019 - initial
 *
 *  @bug No known bugs.
 */
extern uint8_t nsc_APIInTrustZone (int * a, int * b, int * sum);
```

Please refer the header of each function to know more about each function

TODO 4 – veneer functions body

Double click on the TODO 4 in the Task list to jump to the code section.

Right below the TODO line, add the following code:

```
void __attribute__((cmse_nonsecure_entry)) nsc_non_secure_console_puts (uint8_t * string)
{
        non_secure_console_puts(string);
}

void __attribute__((cmse_nonsecure_entry)) nsc_SymmetricAuthentication (void)
{
        SymmetricAuthentication();
}

bool __attribute__((cmse_nonsecure_entry)) nsc_IsAuthenticated (void)
{
        return IsAuthenticated();
}

uint8_t __attribute__((cmse_nonsecure_entry)) nsc_APIInTrustZone (int * a, int * b, int * sum)
{
        return APIInTrustZone(a,b,sum);
}
```

Modify the code in main

TODO 5 – secure app, declaration

Double click on the TODO 5 in the Task list to jump to the code section. It contains the variables and functions declaration for the main.

Right below the TODO line, add the following code:

```
/* TZ_START_NS: Start address of non-secure application */
#define TZ_START_NS 0x00008000

/* typedef for non-secure callback functions */
typedef void (*ns_funcptr_void) (void) __attribute__((cmse_nonsecure_call));
```

TODO 6 - Initialize non-secure application

Double click on the TODO 6 in the Task list to jump to the code section. In this section, we print the "Hello world!" on the terminal, initialize non-secure application and switch to non-secure application.

Right below the TODO line, add the following code:

```
/* Pointer to Non secure reset handler definition*/
ns_funcptr_void NonSecure_ResetHandler;

/* Print Secure Hello world on the terminal window */
secure_console_puts("\x1b[2J");
secure_console_puts ("Secure Hello world !\r\n");

/* - Set non-secure main stack (MSP_NS) */
__TZ_set_MSP_NS(*((uint32_t *)(TZ_START_NS)));

/* - Get non-secure reset handler */
NonSecure_ResetHandler = (ns_funcptr_void)(*((uint32_t *)((TZ_START_NS) + 4U)));

/* - Start Non-secure Application */
NonSecure_ResetHandler();
```

Assignment 3: Adding the code in non-secure world

Modify the code in main

TODO 7 – non-secure app, declaration

Double click on the TODO 7 in the Task list to jump to the code section. It contains the variables and functions declaration for the main.

Right below the TODO line, add the following code:

```
static int a = 10;
static int b = 20;
static int sum = 0;
static char s[100];
```

TODO 8 – non-secure, main function body

Double click on the TODO 8 in the Task list to jump to the code section. In this section, we try to call the secure API before authentication to see the error message. After that we start the authentication process and call the secure API again.

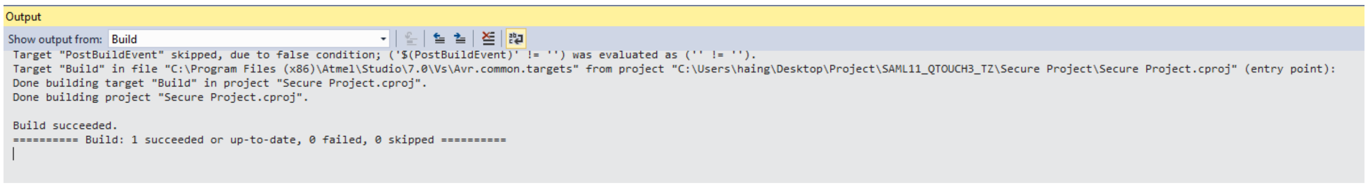
Right below the TODO line, add the following code:

```
/* Verify non-secure application is authenticated */
nsc_non_secure_console_puts((uint8_t *)"Non-Secure Hello World !\r\n");

/*Try to execute secure function call before authentication*/
nsc_non_secure_console_puts((uint8_t *)"Verify non-secure application is authenticated\r\n");
if(nsc_IsAuthenticated())
{
	nsc_non_secure_console_puts((uint8_t *)"Authenticated\r\n");
}
else
{
	nsc_non_secure_console_puts((uint8_t *)"Not authenticated\r\n");
}

/*Execute secure function without authentication*/
nsc_non_secure_console_puts((uint8_t *)"Try to call the API in trustzone without authentication\r\n");
if(0 == nsc_APIInTrustZone(&a, &b, &sum))
{
	sprintf(s, "sum of %d and %d is %d\r\n", a, b, sum);
	nsc_non_secure_console_puts((uint8_t *)s);
}
else
{
	nsc_non_secure_console_puts((uint8_t *)"Function is not executed\r\n");
}

/*Authentication and call the secure function again*/
nsc_non_secure_console_puts((uint8_t *)"Press SW0 to start Authenticate\r\n");
while(gpio_get_pin_level(SW0));

nsc_SymmetricAuthentication();
```

```
nsc_non_secure_console_puts((uint8_t *)"Verify non-secure application is authenticated\r\n");
if(nsc_IsAuthenticated())
{
	nsc_non_secure_console_puts((uint8_t *)"Authenticated\r\n");
}
else
{
	nsc_non_secure_console_puts((uint8_t *)"Not authenticated\r\n");
}

nsc_non_secure_console_puts((uint8_t *)"Try to call the API in trustzone again\r\n");
if(0 == nsc_APIInTrustZone(&a, &b, &sum))
{
	sprintf(s, "sum of %d and %d is %d\r\n", a, b, sum);
	nsc_non_secure_console_puts((uint8_t *)s);
}
else
{
	nsc_non_secure_console_puts((uint8_t *)"Function is not executed\r\n");
}
```

Compile the project again.

If there is no problem, the compilation output should be as following

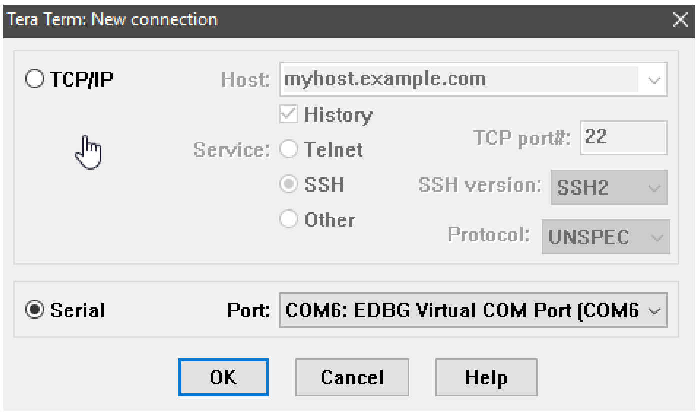


```
Target "PostBuildEvent" skipped, due to false condition; ('$(PostBuildEvent)' != '') was evaluated as ('' != '').
Target "Build" in file "C:\Program Files (x86)\Atmel\Studio\7.0\Vs\Avr.common.targets" from project "C:\Users\haing\Desktop\Project\SAML11_QTOUCH3_TZ\Secure Project\Secure Project.cproj" (entry point):
Done building target "Build" in project "Secure Project.cproj".
Done building project "Secure Project.cproj".

Build succeeded.
========== Build: 1 succeeded or up-to-date, 0 failed, 0 skipped ==========
```

Assignment 4: Testing the application

Setup TeraTerm

Before running the application, we need to setup TeraTerm.

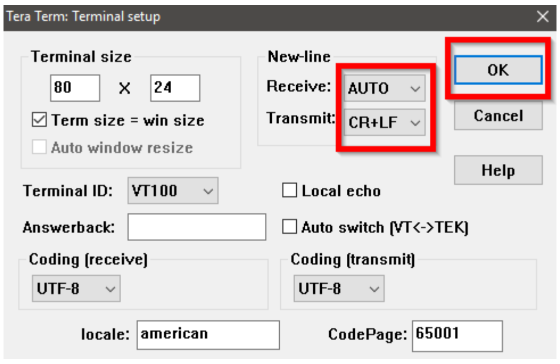

Tera Term: New connection

TCP/IP
Host: myhost.example.com
☑ History
Service: Telnet
[✓ SSH]
Other
TCP port#: 22
SSH version: SSH2
Protocol: UNSPEC

[✓ Serial]
Port: COM6: EDBG Virtual COM Port (COM6

COM port may be different on your machine

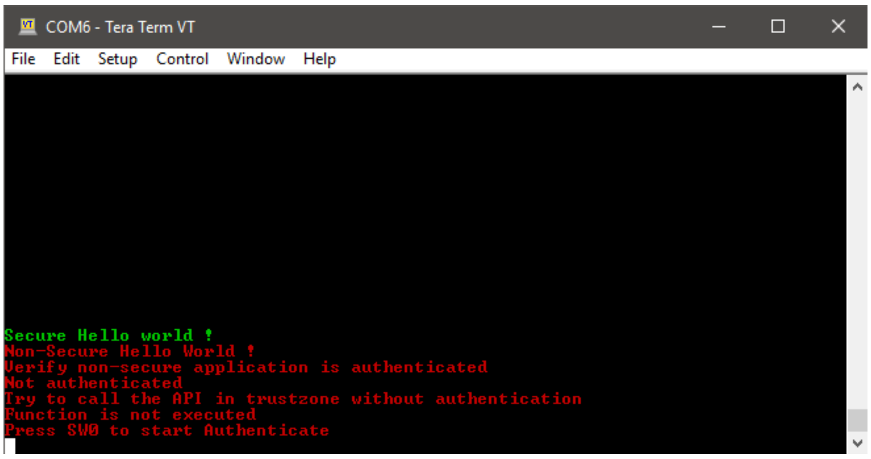
Go to Setup → Terminal and set the following configuration


Tera Term: Terminal setup

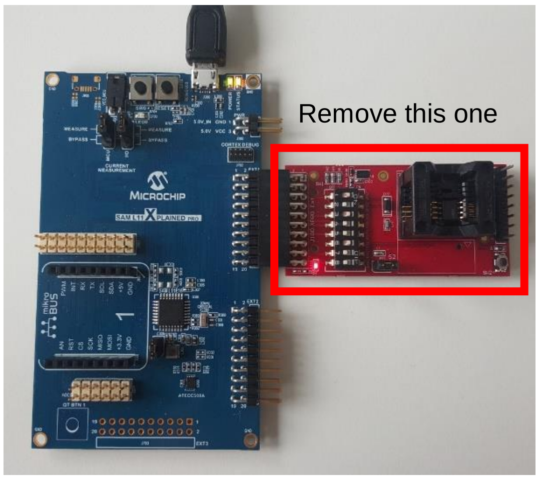
Terminal size: 80 X 24
☑ Term size = win size
☐ Auto window resize
New-line
Receive: AUTO
Transmit: CR+LF
Terminal ID: VT100
Answerback:
☐ Local echo
☐ Auto switch (VT<->TEK)
Coding (receive): UTF-8
Coding (transmit): UTF-8
locale: american
CodePage: 65001

Press reset button on the SAML11 Xplained

If it is setup correctly, you should see the following output on the terminal

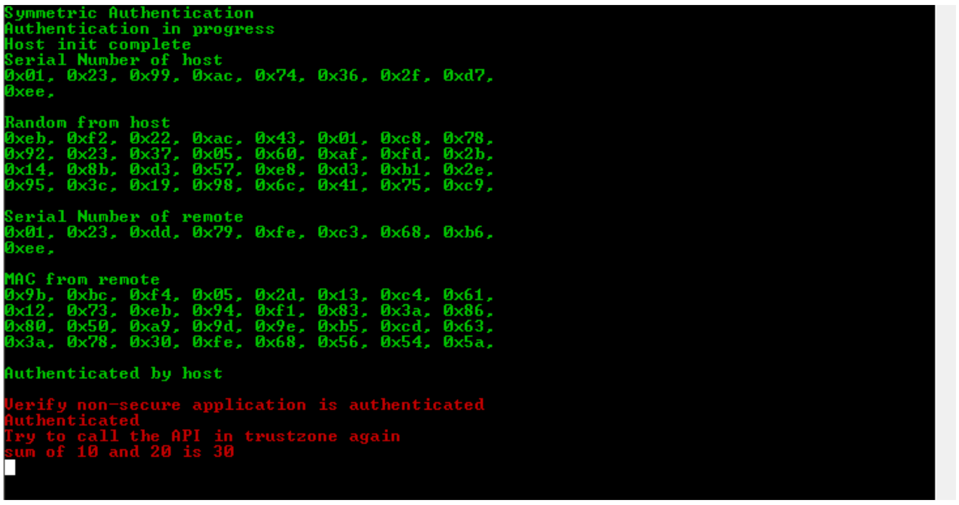


```
Secure Hello world !
Non-Secure Hello World !
Verify non-secure application is authenticated
Not authenticated
Try to call the API in trustzone without authentication
Function is not executed
Press SW0 to start Authenticate
```



At the beginning, the non-secure application tries to call a secure API (the sum function) without authentication. Since un-authorized access, the non-application is not allowed to call the secure API, and the secure API return an error message.

Press SW0 button on the SAML11 to start the authentication process.



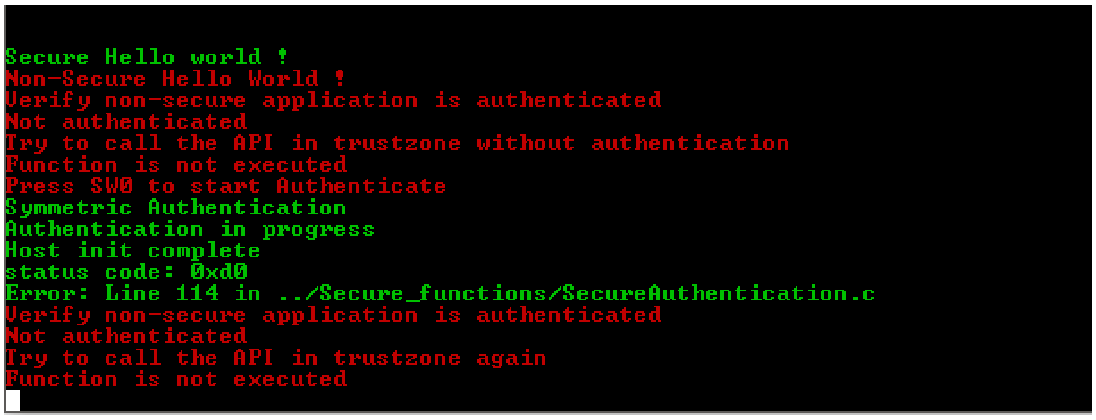
```
Symmetric Authentication
Authentication in progress
Host init complete
Serial Number of host
0x01, 0x23, 0x99, 0xac, 0x74, 0x36, 0x2f, 0xd7,
0xee,

Random from host
0xeb, 0xf2, 0x22, 0xac, 0x43, 0x01, 0xc8, 0x78,
0x92, 0x23, 0x37, 0x05, 0x60, 0xaf, 0xfd, 0x2b,
0x14, 0x8b, 0xd3, 0x57, 0xe8, 0xd3, 0xb1, 0x2e,
0x95, 0x3c, 0x19, 0x98, 0x6c, 0x41, 0x75, 0xc9,

Serial Number of remote
0x01, 0x23, 0xdd, 0x79, 0xfe, 0xc3, 0x68, 0xb6,
0xee,

MAC from remote
0x9b, 0xbc, 0xf4, 0x05, 0x2d, 0x13, 0xc4, 0x61,
0x12, 0x73, 0xeb, 0x94, 0xf1, 0x83, 0x3a, 0x86,
0x80, 0x50, 0xa9, 0x9d, 0x9e, 0xb5, 0xcd, 0x63,
0x3a, 0x78, 0x30, 0xfe, 0x68, 0x56, 0x54, 0x5a,

Authenticated by host

Verify non-secure application is authenticated
Authenticated
Try to call the API in trustzone again
sum of 10 and 20 is 30
```

Figure 7: Authenticated and call secure API

After the successful authentication, the non-secure application calls the secure API again and receives the result of the summation between 10 and 20.

Repeat the step again without plugging the ATECC508 to simulate invalid hardware situation

Because of without the presence of the secure element, the authentication process fails, which is demonstrated by error code on the terminal. Without an authentication, the non-secure world has no way to call the secure API.



```
Secure Hello world !
Non-Secure Hello World !
Verify non-secure application is authenticated
Not authenticated
Try to call the API in trustzone without authentication
Function is not executed
Press SW0 to start Authenticate
Symmetric Authentication
Authentication in progress
Host init complete
status code: 0xd0
Error: Line 114 in ../Secure_functions/SecureAuthentication.c
Verify non-secure application is authenticated
Not authenticated
Try to call the API in trustzone again
Function is not executed
```

Figure 8: Calling secure API without a secure element

Congratulation! You have finished the hands-on!

THE END