|  |  |
| --- | --- |
|  | j0319484.wmf  **ADVANCED MICROPROCESSORS**  **Project Εξαμήνου**  **UNIVERSITY OF PATRAS**  **ENGINEERING SCHOOL**  **ELECTRICAL COMPUTER ENGINEERING DEPARTMENT** |

|  |  |
| --- | --- |
|  | Secure communication between 2 ARM boards implementing AES-128 in  ARM THUMB assembly for low power. |

|  |  |
| --- | --- |
|  | **Name: ID: Dept: Year:**  Nikolaos Tsafas 8219 ECE 5th  Nikolaos Skamnelos 1041878 CEID 3rd |

|  |  |
| --- | --- |
|  | Fall Semester 2016-2017 |

|  |  |
| --- | --- |
|  |  |

**Table of Contents**

ABSTRACT

1. INTRODUCTION…………………………………………………………………p.3.
2. DESIGN OVERVIEW…………………………………………………………….p.3.
3. ARM CORTEX-M……………………………………………………………..p.3.
4. AES……………………………………………………………………………….p.4.
5. ALGORITHM…………………………………………………………p.4.
6. MODE OF OPERATION…………………………………………..p.8.
7. PADDING……………………………………………………………..p.9.
8. USB COMMUNICATION…………………………………………………..p.10.
9. UART COMMUNICATION………………………………………………..p.11.
10. IMPLEMENTATION………………………………………………………………p.13.
11. DESCRIPTION……………………………………………………………….p.13.
12. OPERATION………………………………………………………………….p.14.
13. LESSONS LEARNED ……………………………………………………………p.16.
14. EXPANDING THE IDEA………………………………………………………..p.16.
15. REFERENCES………………………………………………………………………p.17.

ABOUT THE AUTHORS…………………………………………………………….p.17.

ABSTRACT

This project describes a secure system of communication between two computers and two arm boards implementing AES 128-bit CBC cryptography for the Cortex-M4 embedded microprocessors exclusively in Assembly ARM THUMB-2 code.

1. INTRODUCTION

Security is one of the most important factors in our life. We apply password to our PC’s, laptops for protecting our private data. Information is an asset and an asset needs to be secured from attacks. To be secured, information needs to be hidden from unauthorized access, protected from unauthorized change and availability to an authorized entity when needed.

Data security is very important especially from secured transmission of data point of view. Cryptography is a one of the various techniques for transmitting data securely.

For these reasons we present a secure communication system for the most popular modern microprocessor for embedded devices, the ARM Cortex-M4.

Sometimes an embedded device contains a coprocessor that can perform AES encryption in hardware, but such a coprocessor is not always available especially in low power implementations such as Cortex-M0 or even versions of Cortex-M4. Simply compiling an existing implementation written in, for example, the C programming language, is unlikely to produce optimal performance. For this reason, we wrote our implementation exclusively in Assembly ARM THUMB-2 code with optimization regarding loop unfolding and usage of exclusively thumb 16-bit instructions for code density and speed resulting in a suitable for low area and low resources scenario.

1. DESIGN OVERVIEW
2. ARM CORTEX-M

The Cortex-M is a family of 32-bit processors by ARM meant for use in embedded

microcontrollers. They are designed to be cheap and to be energy efficient, while

still being powerful enough to offer adequate performance in applications such

as automotive systems, medical instruments, the Internet of Things, or other

consumer products.

We used the STM32F407VG development board which has a 168MHz Cortex-M4 core, 1024KB of flash memory and 192KB of RAM.

The STM32F407 embeds four universal synchronous/asynchronous receiver transmitters and two universal asynchronous receiver transmitters, as well as, a USB OTG full-speed device/host/OTG peripheral with integrated transceivers. The USB OTG FS peripheral is compliant with the USB 2.0 specification and with the OTG 1.0 specification. It has software-configurable endpoint setting and supports suspend/resume. The USB OTG full-speed controller requires a dedicated 48 MHz clock that is generated by a PLL connected to the HSE oscillator.

On the STM32F407VG, there are many regions to store data: SRAM, FLASH and CCM. In our case we used the SRAM to increase performance.

1. AES

The Advanced Encryption Standard (AES), also known as Rijndael, is a specification for the encryption of electronic data established by the U.S. National Institute of Standards and Technology (NIST) in 2001. AES has been adopted by the U.S. government and is now used worldwide. It supersedes the Data Encryption Standard (DES), which was published in 1977.

The algorithm described by AES is a symmetric-key algorithm, meaning the same key is used for both encrypting and decrypting the data.

1. Algorithm

AES is based on a design principle known as a substitution-permutation network, combination of both substitution and permutation, and is fast in both software and hardware.

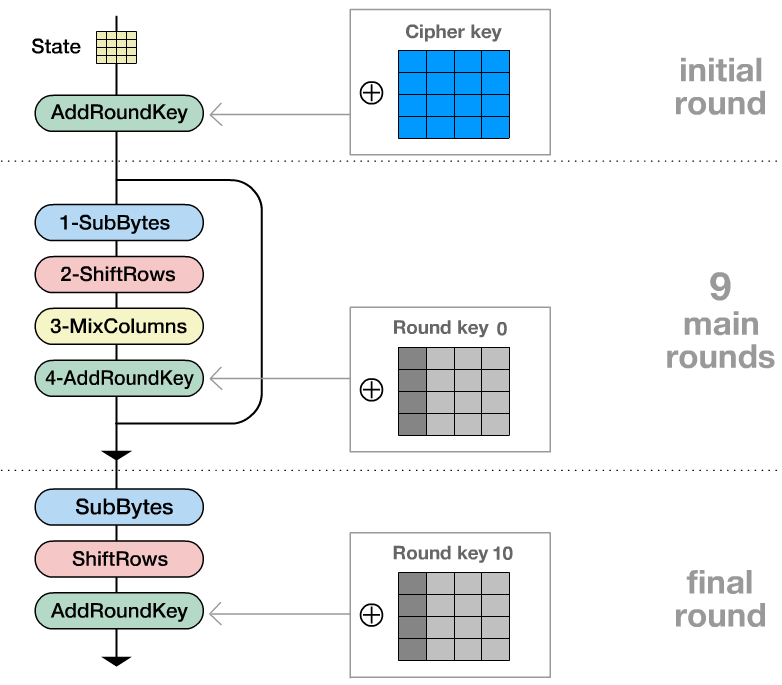
The cipher has a fixed block size of 128 bits, and operates on a 4 × 4 column-major order matrix of bytes, termed “**the state**”.

The key size used for an AES cipher specifies the number of repetitions of transformation rounds that convert the input, called the plaintext, into the final output, called the ciphertext. The number of cycles of repetition are as follows:

* 10 cycles of repetition for 128-bit keys.
* 12 cycles of repetition for 192-bit keys.
* 14 cycles of repetition for 256-bit keys.

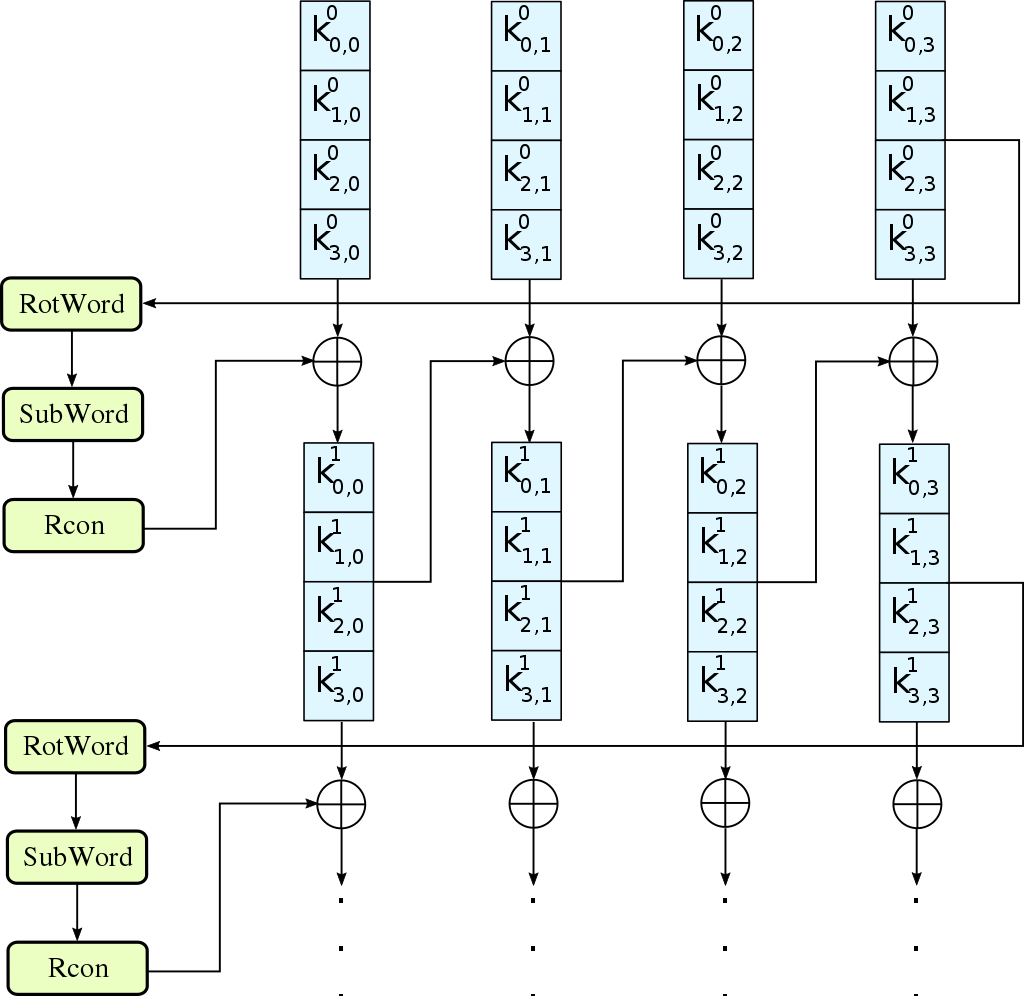
One that depends on the encryption key itself and a set of rounds each containing four similar but different stages

Each round consists of several processing steps, each containing four similar but different stages



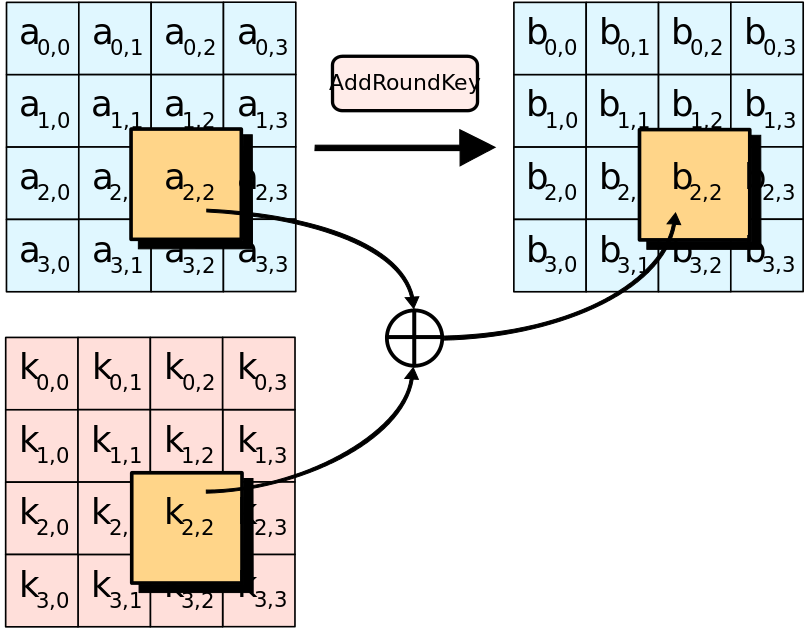
1. Key Expansion:

Round keys are derived from the cipher key using Rijndael's key schedule. AES requires a separate 128-bit round key block for each round plus one more.



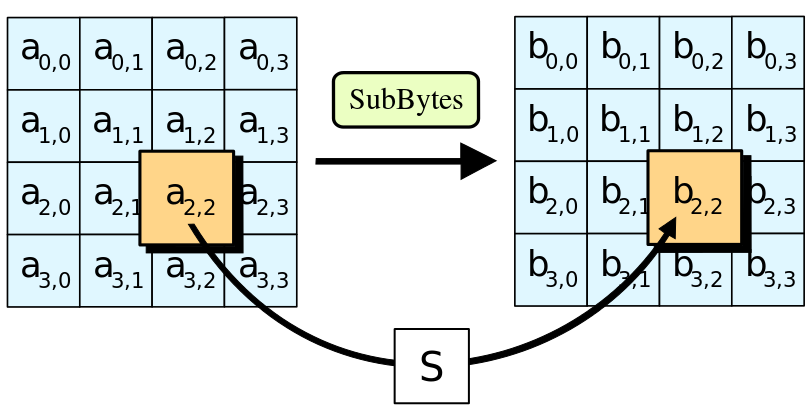
1. Add Round Key:

Each byte of the state is combined with a block of the round key using bitwise xor.



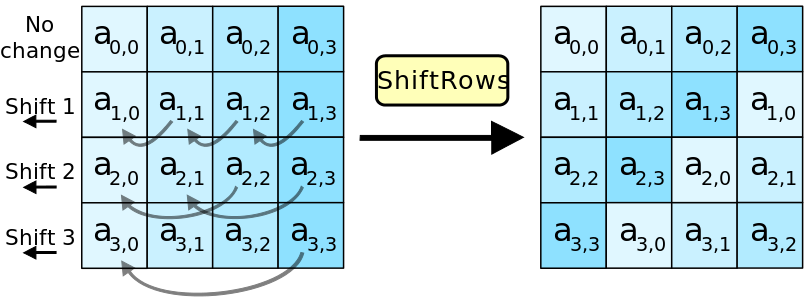
1. Sub Bytes:

In the SubBytes step, each byte in the state matrix is replaced with a SubByte using an 8-bit substitution box, the Rijndael S-box. This operation provides the non-linearity in the cipher.



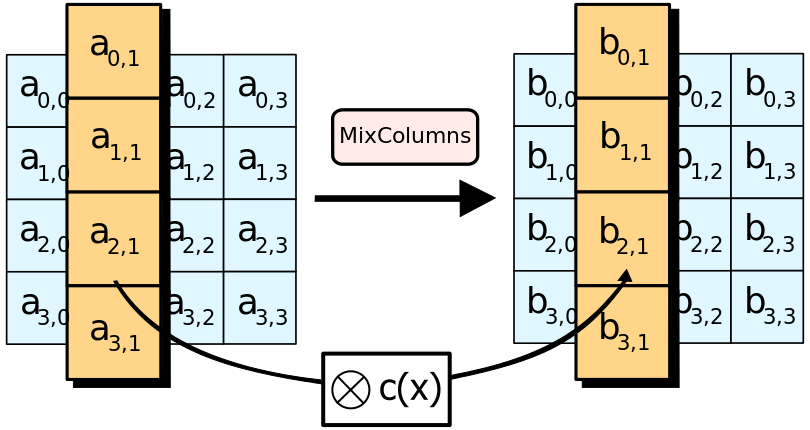
1. Shift Row:

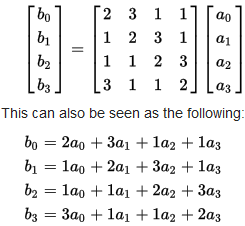
A transposition step where the last three rows of the state are shifted cyclically a certain number of steps.



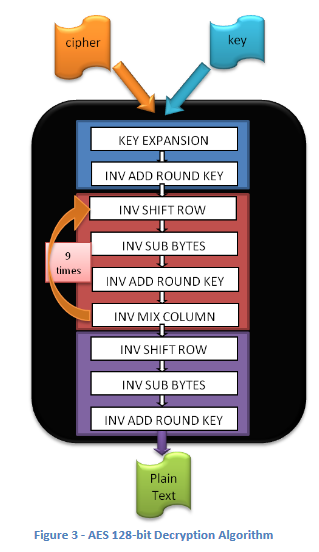
1. Mix Column:

In the Mix Columns step, each column of the state is multiplied with a fixed polynomial c ( x ) .





Respectively the decryption process is described by the following schematic, in which the processes within each round are executed in reverse order and each function has been changed by an inverted one:

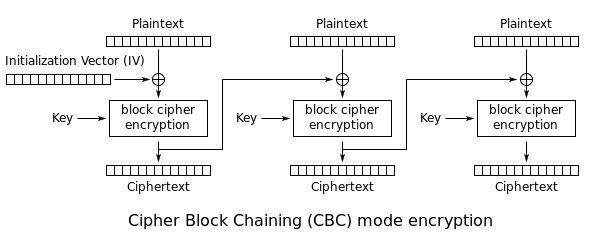


1. Mode of operation

A mode of operation describes how to repeatedly apply a cipher's single-block operation to securely transform amounts of data larger than a block.

Cipher Block Chaining (CBC):

In CBC mode, each block of plaintext is XORed with the previous ciphertext block before being encrypted. This way, each ciphertext block depends on all plaintext blocks processed up to that point. To make each message unique, an initialization vector must be used in the first block.





1. Padding

Block ciphers have one or more block size(s), but during transformation the block size is always fixed. Block cipher modes operate on whole blocks and require that the last part of the data be padded to a full block if it is smaller than the current block size.

There are at least five common conventions:

1. Pad with bytes all of the same value as the number of padding bytes
2. Pad with 0x80 followed by zero bytes
3. Pad with zeroes except make the last byte equal to the number of padding bytes
4. Pad with zero (null) characters
5. Pad with space characters

Method one is the most popular and is usually referred to as "PKCS5 padding". It is generally recommended in the absence of any other considerations.

1. USB COMMUNICATION

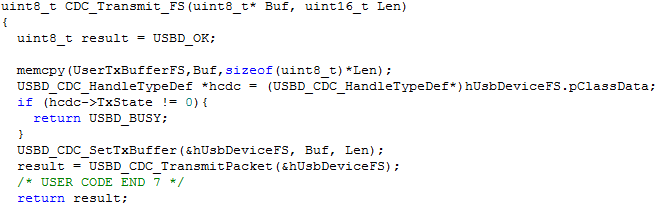
USB, short for Universal Serial Bus, is an industry standard initially developed in the mid-1990s that defines the cables, connectors and communications protocols used in a bus for connection, communication, and power supply between computers and electronic devices.

The functionality of USB devices is defined by class codes. The communications device class is used for computer networking devices. Devices of this class are also implemented in embedded systems. The data interfaces are generally used to perform bulk data transfer.

In order to implement this module in our project we used the HAL libraries provided by ST.

Below our usb receive and transmit function implementation is shown.



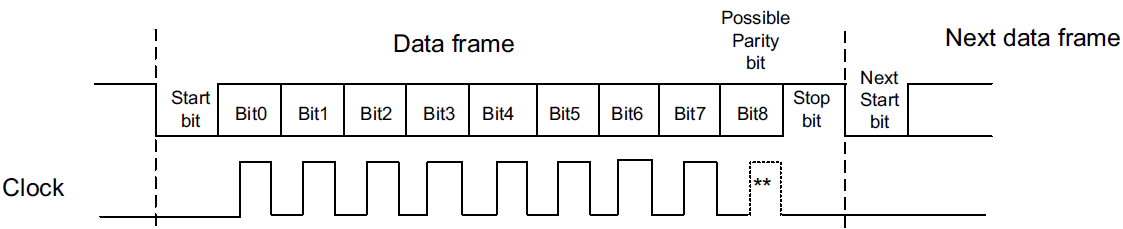


1. UART COMMUNICATION
2. Description

The universal synchronous asynchronous receiver transmitter (USART) offers a flexible means of full-duplex data exchange with external equipment using an asynchronous serial data format.

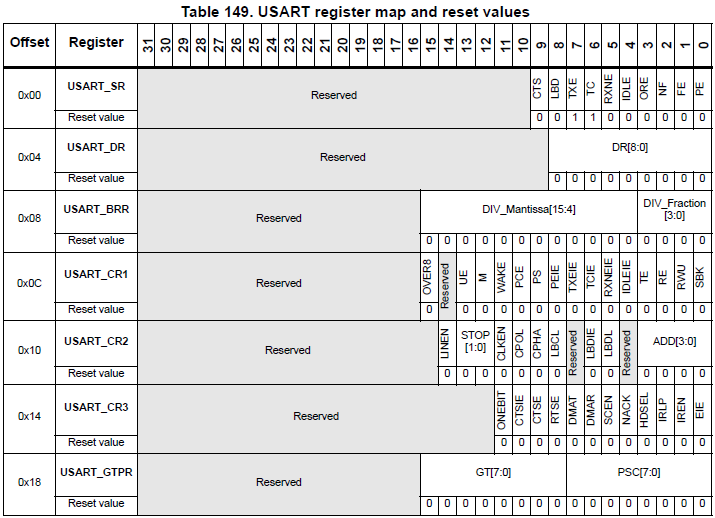
USART bidirectional communication requires a minimum of two pins: Receive Data In (RX) and Transmit Data Out (TX).

The transmitter can send data words of either 8 or 9 bits. Every character is preceded by a start bit which is a logic level low for one-bit period. The character is terminated by a configurable number of stop bits. The following stop bits are supported by USART: 0.5, 1, 1.5 and 2 stop bits.



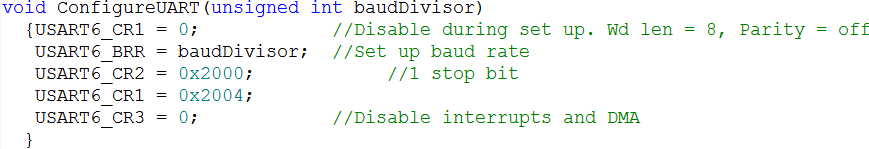
1. Configuration

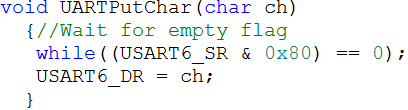
This project utilizes USART6 of the board (USART6\_RX : pin PC7 USART6\_RX : pin PC6) located in address “0x40011400”. The registers of the USART are shown below.

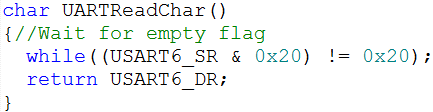


Procedure:

1. Enable the USART by writing the UE bit in USART\_CR1 register to 1.
2. Program the M bit in USART\_CR1 to define the word length.
3. Program the number of stop bits in USART\_CR2.
4. Select DMA enable (DMAT) in USART\_CR3 if Multi buffer Communication is to take place. Configure the DMA register as explained in multibuffer communication.
5. Select the desired baud rate using the USART\_BRR register.
6. Set the TE bit in USART\_CR1 to send an idle frame as first transmission.
7. Write the data to send in the USART\_DR register (this clears the TXE bit). Repeat this for each data to be transmitted in case of single buffer.
8. After writing the last data into the USART\_DR register, wait until TC=1. This indicates that the transmission of the last frame is complete. This is required for instance when the USART is disabled or enters the Halt mode to avoid corrupting the last transmission.

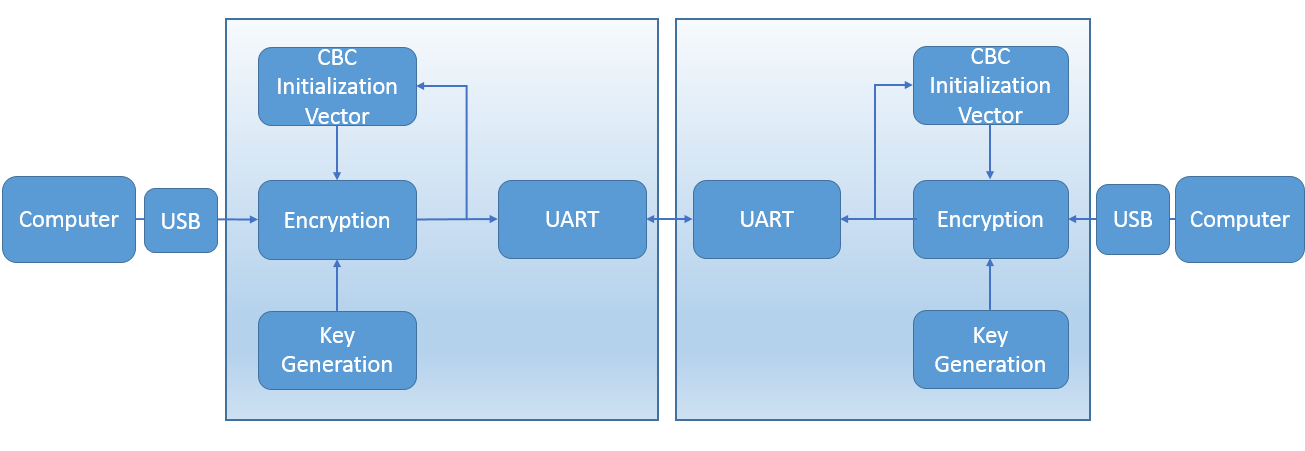


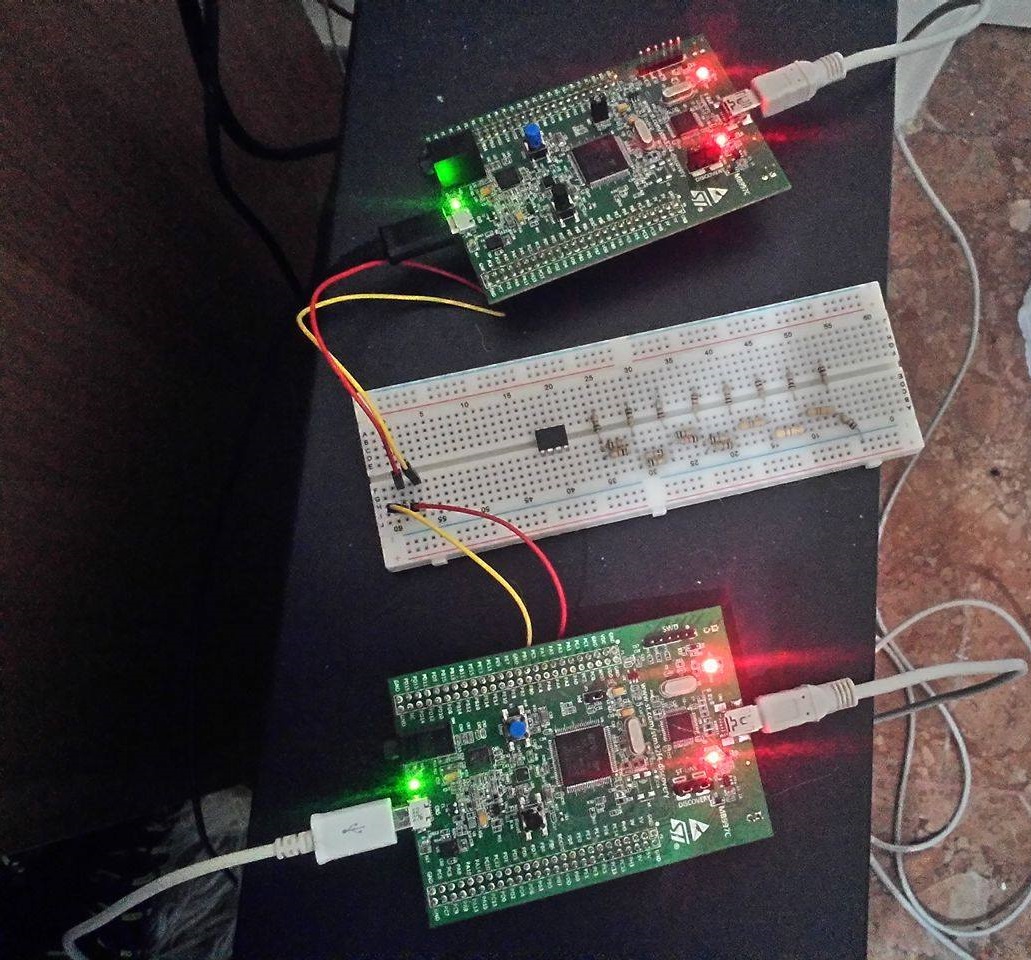




1. IMPLEMENTATION

Our implementation is presented in the following schematic.





1. Description

A user sends a message through usb to the application board.

The encryption process begins with the shared key and a standard Initialization Vector and using CBC mode it sends each block through UART.

The second board performs the decryption and sends the decrypted message to the second user.

This scheme can work in either direction as both boards are equipped with encryption- decryption and transmit-receive functions.

1. Operation

We send the message “we have been compromised”

In the left window we see the ciphertext in asci while in the right window we see the decrypted message in the receiver end.

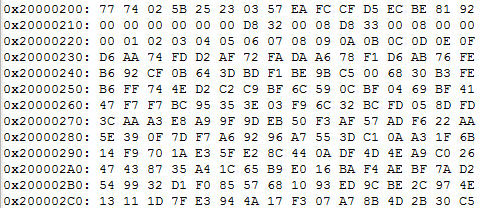
In this example the cbc mode of operation is checked if working properly as can be also seen in debugging mode of the processor in the next image.



In the following images the encryption process is presented in the memory of the arm board.

We put the first block in 0x20000200

While in 0x20000220 we have the expanded key

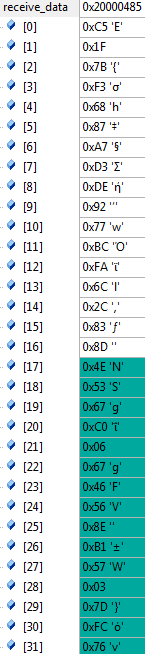
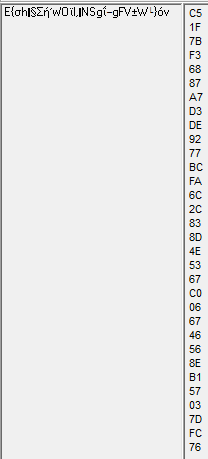
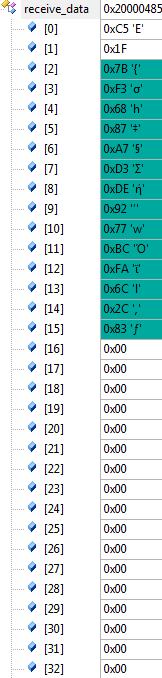


Next we see the two encrypted blocks after the 2 iterations as the message length is 2 blocks.





Also we see how the variables are updated after each block operation and the final ciphertext transmitted through UART.



In the next image we see an example of two way communication.

We send the message ”project2017” and then the 2nd user replies “key expansion”

In both windows we see both the messages received and the ciphertext sent.



1. LESSONS LEARNED

The completition of this projects was based on partitioning of tasks and modularity enabling working simultaneously on different parts of the design, as well as, the development of skills on specification definition.

Another important aspect of our work was time management and organization with each module being implemented and tested on a finite timeline in order to produce improved working versions of our system as time progressed.

Finally, thorough study of the concepts involved in the project made us aware of the vast importance of secure communication as well as the capabilities of embedded hardware in the IoT environment.

1. EXPANDING THE IDEA

Some ideas the great team has come up with were:

* Implementation of AES 192-bit/256-bit decryption-encryption process and various other block modes of operation (Electronic Codebook (ECB), Propagating Cipher Block Chaining (PCBC), Cipher Feedback (CFB), Output Feedback (OFB), Counter (CTR)).
* Replacement of the UART communication with wireless connectivity such as,

Wi-Fi or Bluetooth, for exchanging data between not only two boards but a group of users in range.

* Creation of a key distribution system that will allow the exchange of different keys between the available users.
* Using the capabilities of the ARM Microprocessor we could provide our system with a keyboard and display resulting in an autonomous secure communication device without the need of a host computer.

S

1. REFERENCES
2. https://en.wikipedia.org/wiki/Advanced\_Encryption\_Standard
3. http://csrc.nist.gov/publications/fips/fips197/fips-197.pdf
4. https://en.wikipedia.org/wiki/Block\_cipher\_mode\_of\_operation#CBC
5. https://sites.google.com/site/terminalbpp/

ABOUT THE AUTHORS



Skamnelos Nikolaos is an undergraduate student at Patras Department of Computer Engineering & Informatics. His interests lay with computer architecture and embedded software.

Tsafas Nikolaos is currently pursuing his Diploma degree in the ECE dept., University of Patras. His interests are VLSI circuit design, digital signal processing and embedded systems.