Proceedings of Informing Science & IT Education Conference (InSITE) 2015

Cite as: Akinwonmi, F. C, Alese, B. K., Dahunsi, F. M., Osuolale, F. A & Odo, A. E. (2015). Design of circuit system-based cryptography. *Proceedings of Informing Science & IT Education Conference (InSITE) 2015*, 43-58. Retrieved from http://Proceedings.InformingScience.org/InSITE2015/InSITE15p043-058Akinwonmi1552.pdf

Design of Circuit System-Based Cryptography

F. C. Akinwonmi, B. K. Alese, F. M. Dahunsi, F. A. Osuolale
Computer Science Department,
Federal University of Technology, Akure, Nigerial

fcakinwonmi@futa.edu.ng, bkalese@futa.edu.ng,
fmdahunsi@futa.edu.ng, aofestus@futa.edu.ng

A. E. Odo
The Federal University of Oye – Ekiti, Oye – Ekiti, Nigeria

odoalawaye@gmail.com

Abstract

Cryptography is the science of writing in secret codes which can be achieved either by using software encrypter or hardware encrypter. This study presents the development of a pair of circuit system-based (hardware) cryptographic processor. The hardware encryption in this study was achieved using the bitwise logic operations in the registers of the microcontroller and messages are streamed as serial ASCII data from the PC through the USB port to the microcontroller unit. The encryption is performed on each ASCII representation using a pass key embedded in the microcontroller unit. Decryption process is similar but in the reverse order. A comparative analysis of the encryption time for the hardware-based data encryption was made and findings recorded. The processor was implemented using PIC18F4550 microcontroller mounted on a Printed Circuit Board alongside other components to achieve the hardware based circuitry. The software end of the encryption and decryption algorithm was developed based on a library built into C Language Visual Studio version 2010 and the CCS C compiler for communication protocol stack.

Keywords: Cryptology, Cryptography, Encrypter, Communication, Protocol, Microcontroller

Introduction

Communications are electronically processed and information is conveyed along public networks.Therefore, security becomes a tremendously important issue to deal with. There are many aspects to security and many applications ranging from secure commerce and payments to private communications and protecting passwords and one essential aspect for secure communications is that of cryptography (Kessler, 2010).

A documented use of cryptography in writing dated back to 1900 B.C. when an Egyptian scribe used non-standard hieroglyphs in an inscription. Some experts argue that cryptography appeared spontaneously sometime after writing was invented, with applications ranging from diplomatic missives to war-time battle plans. Cryptography could be said to mean the art and science of encrypt-

Editor: Eli Cohen

ing messages in a way that it is unintelligible to whoever is not authorized to have access to it and it is a field that is concerned with linguistic and mathematical techniques for securing information, particularly in communications (Alese, 2000).One can categorize attacks on a cryptosystem as being of a number of different forms: ciphertext-only,known-plaintext, chosen-plaintext and man-in-the-middle attack.

Cryptanalysis on the other hand is the science of recovering the plaintext message without the knowledge of the key which is known as attack as defined by O'Mahony et al. (2001). One can categorize attacks on a cryptosystem as being of a number of different forms:

(a) Ciphertext-only attack: In this attack, the cryptanalyst has the ciphertext of several messages, all of which have been encrypted using the same encryption key. From this, the cryptanalyst attempts to derive either the plaintext or the key.

(b) Known-plaintext attack: The cryptanalyst has access not only to the ciphertext of several messages but also to the corresponding plaintext. From this, the cryptanalyst may be able to derive the key used for encrypting the messages.

(c)Chosen-plaintext attack: The cryptanalyst has access to the ciphertext and associated plaintext for several messages and he or she can gain access to ciphertext corresponding to plaintext that he or she has chosen. These blocks could be chosen to yield more information about the key or to pursue a particular line of attack.
(d) Man-in-the-middle attack: This attack is relevant for cryptographic communication and key exchange protocols. The idea is that when two parties are exchanging keys for secure communications (for example, using Diffie-Hellman), an adversary puts himself between the parties on the communication line. The adversary then performs a separate key exchange with each party. The parties will end up using a different key, each of which is known to the adversary. The adversary will then decrypt any communications with the proper key, and encrypt them with the other key for sending to the other party. The parties will think that they are communicating securely, but in fact the adversary is hearing everything (InfoSysSec, 2013).

Wireless Communications have become a very attractive and interesting sector for the provision of electronic services thereby making mobile networks possible and available almost anywhere, anytime and making the number of the users of wireless hand-held devices at the increase. Before, wireless communication protocols have specified security layers, which support security with high level strength. These wireless protocols security layers use encryption algorithms, which in many cases have been proved unsuitable and outdated for hardware implementations (Koufopavlou et al., 2005).

Furthermore, the evolution of portable devices requires a complete security system which satisfies the demands of fast and secure transactions. Unfortunately, software-based approaches, especially for public-key cryptography, lead to slow implementations that are very inefficient, then the existence of supplementary hardware is essential. In addition, next generation portable devices have enclosed wireless communication protocols and they can operate only with extremely low power conditions. Power management however, is a critical demand to support cryptographic capabilities.

Related Literature

Cryptography can be defined as the science of writing in secret code. According to Kessler (2010), within the context of any application-to-application communication, there are some specific security requirements, including: *authentication, privacy/confidentiality, integrity,* and *non-repudiation*. Cryptography is therefore described as the study of mathematical techniques related

to these aspects of information security requirements (Menezes et al., 1996). Also, cryptography does not only protect data from theft or alteration, it is also used for user authentication.

Beuchat et al. (2007) proposed a compact implementation of the η^T pairing in characteristic three over $\frac{F_3(x)}{x^{97}+x^{\frac{1}{2}}+2}$ using an architecture based on a unified arithmetic operator that yields the smallest available circuit known in literature. The study also showed that the new approach can be generalized to any characteristic 'p' and degree 'm' irreducible polynomial $f(x)$ over F_p.

The study was however unable to resolve the scheduling of all the instructions required for η^T pairing calculations, while it also proposed to develop an ad-hoc compiler based on unified operators.

The original message which is in human readable form is called the plaintextorcleartext, while the disguised message is called the ciphertext and the final sent message is called a cryptogram. The process of transforming plaintext into ciphertext is called encryption and the reverse process of turning the ciphertext into plaintext is called decryption. Plaintext is denoted by P, whereas ciphertext is denoted by C. The encryption function E operates on P to produce C:

$$E(P) = C \quad 2.1$$

In the reverse process, the decryption function D operates on C to produce P:

$$D(C) = P \quad 2.2$$

A restricted cryptosystem requires the encryption and decryption algorithms to be kept secret called security by obscurity and should be used only in very specific cases. All modern encryption algorithms use a key, denoted by K as shown below. The value of this key affects the encryption and decryption functions, so that they can now be written as:

$$E(K, P) = C \quad 2.3$$

$$D(K, C) = P \quad 2.4$$

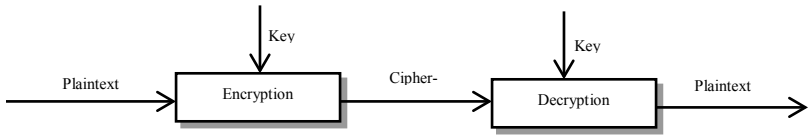

Figure 1: Encryption and decryption using a key Source: (Kessler, 2010)

A message when sent in a number of times using different keys or Initialization Vector looks different each time to a snooper listening to the encrypted traffic on the wire. Secret key cryptography can either be stream ciphers or block ciphers. Stream ciphers operate on a single bit (byte or computer word) at a time and implement some form of feedback mechanism so that the key is constantly changing. A block cipher encrypts one block of data at a time using the same key on each block. In general, the same plaintext block will always encrypt to the same ciphertext when using the same key in a block cipher whereas the same plaintext will encrypt to different ciphertext in a stream cipher. The secret key cryptography algorithms that are in use today are: Data Encryption Standard (DES), Triple DES and Advanced Encryption Standard. For the sake of the research work, DES is going to be used.

Data Encryption Standard (DES)

This is a kind of cryptography that operates on a single chunk of data at a time, encrypting 64 bits (8 bytes) of plaintext to produce 64 bits of ciphertext. The key length is 56 bits, often expressed as an eight character string with the extra bits used as a parity check. DES is a block-cipher employing a 56-bit key that operates on 64-bit blocks which has a complex set of rules and transformations that were designed specifically to yield fast hardware implementations and slow software implementations.

In public-key cryptography, each person gets a pair of keys, called the public and the secret key. The public key is published and widely distributed while the secret key is never revealed. Therefore, when user Alice wishes to send an encrypted message to Bob, she uses Bob's public key (PK_B) to encrypt the message, and sends it off to Bob. Bob then uses his secret key (SK_B) to decrypt the message. Anyone who has access to Bob's public key can send him an encrypted message, but no one else apart from Bob can decrypt it. This method could be also used to prove who sent a message; Alice, for example, could encrypt some plaintext with her private key; when Bob decrypts using Alice's public key, he knows that Alice sent the message and Alice cannot deny having sent the message and this is called non-repudiation. It does not matter which key is applied first, but that both keys are required for the process to work, this approach is also called asymmetric cryptography and is shown figure 2.5.

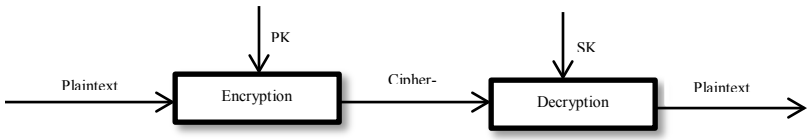

Figure 2: Asymmetric cryptography (Source: O'Mahony, 2001)

Lejlaet al. (2002) presented an overview of hardware implementations for two commonly used types of Public Key Cryptography. That is, RSA and Elliptic Curve Cryptography (ECC). Both are based on modular arithmetic where the mathematical background and the algorithms to implement these cryptosystems were discussed. It was discovered that hardware as seen in the 1980's is still relevant in the sense of its fundamentals such as systolic array and RNS realizations.

Also, Malan et al. (2004) presented a paper on the first known implementation of elliptic curve cryptography over F2*p* for sensor networks based on the 8-bit, 7.3828-MHz MICA2 mote. Through instrumentation of UC Berkeley's TinySec module, it was argued that there remained a need for an efficient and secure mechanism for distribution of secret keys among nodes although secret-key cryptography has been tractable in this domain for some time. Also it was argued that public-key infrastructure was impractical, but through implementation for TinyOS multiplication of points on elliptic curves it was discovered that public-key infrastructure is viable for TinySec keys' distribution, even on the MICA2. It was also shown that it offered equivalent security at lower cost to memory and bandwidth than Diffie-Hellman based on DLP.A public-key infrastructure for key distribution based on elliptic curves is an apt, and viable, choice for TinyOS on the MICA2.

Also, Svetlin (2007) presented a study of the efficiency in applying modern Graphics Processing Units in Symmetric Key Cryptographic solutions. It discussed both traditional style approaches based on the OpenGL graphics API and new ones based on the recent technology trends of major hardware vendors. This explained an efficient implementation of the Advanced Encryption Standard (AES) algorithm in the novel CUDA platform by Nvidia which presented the most efficient currently known approaches in encryption and decryption of messages with AES on pro-

grammable graphics processing units. It suggested that the traditional graphics hardware architectures could now be compared with optimized sequential solutions on the CPU. Beuchat et al. (2007) also proposed a compact implementation of the η^T pairing in characteristic three over

$$\frac{F_3(x)}{x^{97} + x^{\frac{1}{2}} + 2}$$

using an architecture based on a unified arithmetic operator that yields the smallest available circuit known in literature. The study also showed that the new approach can be generalized to any characteristic 'p' and degree 'm' irreducible polynomial $f(x)$ over F_p. The study was however unable to resolve the scheduling of all the instructions required for η^T pairing calculations, while it also proposed to develop an ad-hoc compiler based on unified operators.

System Design

Design of Software for Encryption and Decryption Process

The encryption and decryption of texts were done by using a library called System.Security.Cryptography class embedded into the C Visual Studio Version 2010 software with CCS C-Compiler that uses a real-time operating system (RTOS) and require a complex communication protocol stack such as USB. The System.Security.Cryptography namespace contains support for the most common symmetric (DES, 3DES, RC2, Rijndael), asymmetric (RSA, DSA), and hash (MD5, SHA-1, SHA-256, SHA- 384, SHA-512) cryptography algorithms. In this research work, TripleDES Cryptography Algorithm with function EncrytData(TxSMG) and function DecrptData(TxSMG) alongside with MD5 Algorithm were used. During encryption, the user is mandated to supply a session key and this session key is been encrypted by MD5, then the encrypted session key is added to each character of the plaintext message before function EncrytData(TxSMG) is finally used to encrypt the plaintext message. Decryption is achieved through the same process but in reverse order. Also, there is a function called clear method that overwrites all sensitive data within the object when it is called.

The Data Encryption Algorithm

At each of the iterations, the algorithm takes in two 32-bit inputs to produce two 32-bit outputs. The left output is a replicate of the right input and the right output is an exclusive OR (XOR) of the left input with a function of the right input and the key for the stageK_i. The complexity in the iterations lies in the function f, which does a number of substitutions and permutations using the simple hardware elements called S-boxes (for substitution) and P-boxes (for permutation). Decryption in the DES algorithm uses the same sequence of steps, but the keys used at each of the 16 stages (K1 to K16) are applied in reverse order.

If plaintext = initial 64 bits block of plaintext meant for encryption.
A= the string of text that has not undergone any permutations.
B = the string of text found by performing the permutation IP on A.

Then,

$$B = IP(A) \qquad (3.1)$$

The plaintext message will be divided into two equal half, labeled as L_i and R_i respectively. Then swaps these halves in an interactive method for $i = 1, 2, 3, \ldots, 16$.

$L_0(32\ bitblock) = lefthalves$ of the initial 64 bit block of the plaintext after it has been initially permuted.

$R_0(32\ bitblock) = righthalves$ of the initial 64 bit block of the plaintext after it has been initially permuted.

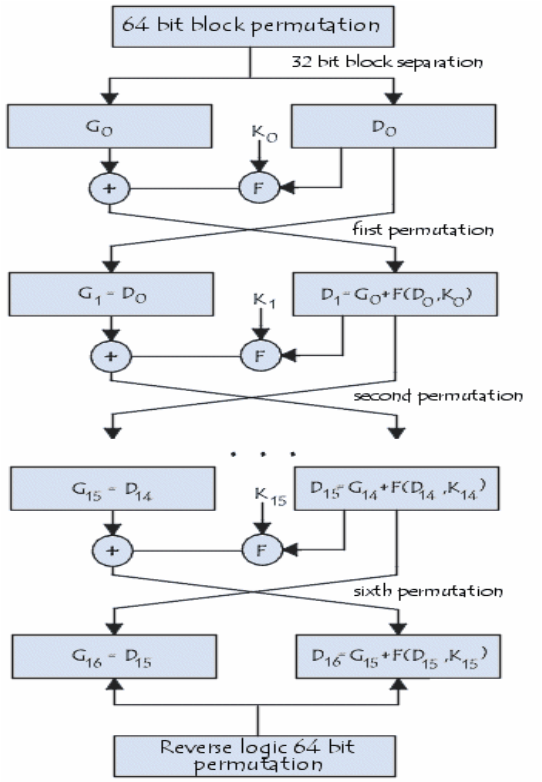

Figure 3: Block Diagram of DES Algorithm. (Source: Kioskea, 2014)

Rounds or Iterations

$i^{th}round = i^{th}iteration$ as described below.

$L_i := R_{i-1}$ for $1 \leq i \leq 16$

$R_i := L_{i-1} \oplus f(R_{i-1}, K_i)$ for $1 \leq i \leq 16$

(Both the function f and the subkeys K_i are described.

Key and the 16 Subkeys Derived From It

The encryption key (K) will be used to generate "subkeys" K_i that will be used in rounds or iterations

$K = Key(64\ bits)$

$= \{56\ bitsdeterminginghowthetrasformationsofthetextistobecarriedout\}$

$\cup\ \{8\ bitsuseforparity\}$

For example, if the message A to be encrypted has its hexadecimal format as $M = \cdots$ "$123456789ABCDEF$" ..., the resulting 64-bit block of text:

$M =$ 0001 0010 0011 0100 0101 0110 0111 1000 1001 1010 1011 1100 1101 1110 1111,

Applying the initial permutation (IP) to the block of text M, gives

$IP =$

1100 1100 0000 0000 1100 1100 1111 1111 1111 0000 1010 1010 1111 0000 1010 1010.

Then, divide the permuted block IP into a left half $L_0$of $32\ bits$, and a right half R_0 of $32\ bits$.

$$L_0 = 1100\ 1100\ 0000\ 0000\ 1100\ 1100\ 1111\ 1111$$
$$R_0 = 1111\ 0000\ 1010\ 1010\ 1111\ 0000\ 1010\ 1010$$

Let K be the hexadecimal $key\ K = 133457799BBCDFF1$. This gives the binary key:

K
$= 00010011\ 00110100\ 01010111\ 01111001\ 10011011\ 10111100\ 11011111\ 11110001$

The 64-bit key is permuted to obtain 56-bit permutation

$K+ = 1111000\ 0110011\ 0010101\ 0101111\ 0101010\ 1011001\ 1001111\ 0001111.$

Next, split this key into left and right halves, C_0andD_0, where each half has 28 bits to get C_0 = 1111000 0110011 0010101 0101111 and D_0 = 0101010 1011001 1001111 0001111.

With $C_0$and D_0 sixteen blocks C_n andD_n, $1 \leq n \leq 16$ were created. Each pair of blocks $C_n$and $D_n$is formed from the previous pair C_{n-1} andD_{n-1}, respectively, to form the keysK_n for

$$1 \leq n \leq 16$$

Iterative Function

$f = f(R_{i-1}, K_i)$: $\{R_i \mid i \in [0,15]\}X\{K_i \mid i \in [1,16]\} \rightarrow \{asubsetoftheinitialplaintext\} = thefunctionusedtodeterminetheiterativesequenseR_i$ (3.2)

Next in the f calculation, XOR the output $E(R_{i-1})$ with the keyK_n:

$$K(n) \oplus E(R_{i-1})$$

To this point R_{i-1} has been expanded from 32 bits to 48 bits, using the selection table and XORed the result with the key K_n to get 48 bits, or eight groups of six bits.Each group of six bits is used as address in tables called "S boxes" to give an address each in a different S box.

Writing the previous result, this is 48 bits, in the form:

$$K(n) \oplus E(R_{i-1}) = B_1B_2B_3B_4B_5B_6B_7B_8 \quad (3.3)$$

Where each $B_i$is a group of six bits. Calculate

$$S_1(B_1)S_2(B_2)S_3(B_3)S_4(B_4)S_5(B_5)S_6(B_6)S_7(B_7)S_8(B_8)$$

Where $S_i(B_i)$refers to the output of theithS box.

Then proceed through 16 iterations, for $1 \leq n \leq 16$, using a function f which operates on two blocks--a data block of 32 bits and a key K_n of 48 bits--to produce a block of 32 bits.

For example, for n = 1, gives

$$K_1 = 000110\ 110000\ 001011\ 101111\ 111111\ 000111\ 000001\ 110010$$

$L_1 = R_0 = 1111\ 0000\ 1010\ 1010\ 1111\ 0000\ 1010\ 1010$

$R_1 = L_0 + f(R_0, K_1)$ = 1110 1111 0100 1010 0110 0101 0100 0100

The final stage in the calculation of f is to do a permutation P of the S-box output to obtain the final value of f:

$$f = P\left(S_1(B_1)S_2(B_2) \ldots S_8(B_8)\right) \quad (3.4)$$

Processing all 16 blocks using the method defined previously, then gives the result below, on the 16th round,

$$L_{16} = 0100\ 0011\ 0100\ 0010\ 0011\ 0010\ 0011\ 0100$$
$$R_{16} = 0000\ 1010\ 0100\ 1100\ 1101\ 1001\ 1001\ 0101$$

Reversing the order of these two blocks and applying the final permutation gives

$R_{16}L_{16}$ = 00001010 01001100 11011001 10010101 01000011 01000010 00110010 00110100

IP^{-1} = 10000101 11101000 00010011 01010100 00001111 00001010 10110100 00000101 which in hexadecimal format is $85E813540F0AB405$.

Therefore the encrypted form of M = 0123456789ABCDEF gives, C = 85E813540F0AB405.

Decryption is simply the inverse of encryption, following the same steps as above, but reversing the order in which the subkeys are applied (Orlin, 2009).

Development of the CCS USB Application Program Interface (API) Software

The CCS C Compiler provides an API for developing USB applications on the PIC. There are several files associated with the USB API that were used in this research work:
pic18_usb.h - Provides hardware layer functions for Microchip PIC18 microcontrollers that have a built-in USB peripheral, such as the PIC18F4550. This gives access for setting up the peripheral, sending and receiving packets, and so on.usb.c - Using an interrupt driven by the hardware, this provides token handler support. The bulk of this file automatically handles and responds to the SETUP tokens used by the Personal Computer to configure the USB device.

usb.h – Prototypes, global definitions and conditional compile statements used in conjunction with usb.c. There are some global definitions made here that can be changed to alter the USB API to an application.usb_desc_*.h - Provides example device, configuration, interface, endpoint and string descriptors needed to describe the USB devices used in the examples. The SETUP handler in usb.c uses these descriptors to answer Get_Descriptor requests. Since USB descriptors are applications dependent, there is a different descriptor for each example program.

Hardware circuit design

Figure below shows the block diagram of the designed circuit. The PIC MCU is connected to Computer 1 through the USB port on the computer and receives power through the same medium at the sending side. The ICSP is connected to the PIC MCU through the ICSP pin on the board then to Computer 1 through the USB port. Interconnection between the two PIC MCUs is done through the two RJ45 connectors and the PIC MCU at the receiving side now connected to Computer 2 through the USB port. In the implementation of the hardware device, these components will be mounted on a Printed Circuit Board (PCB).

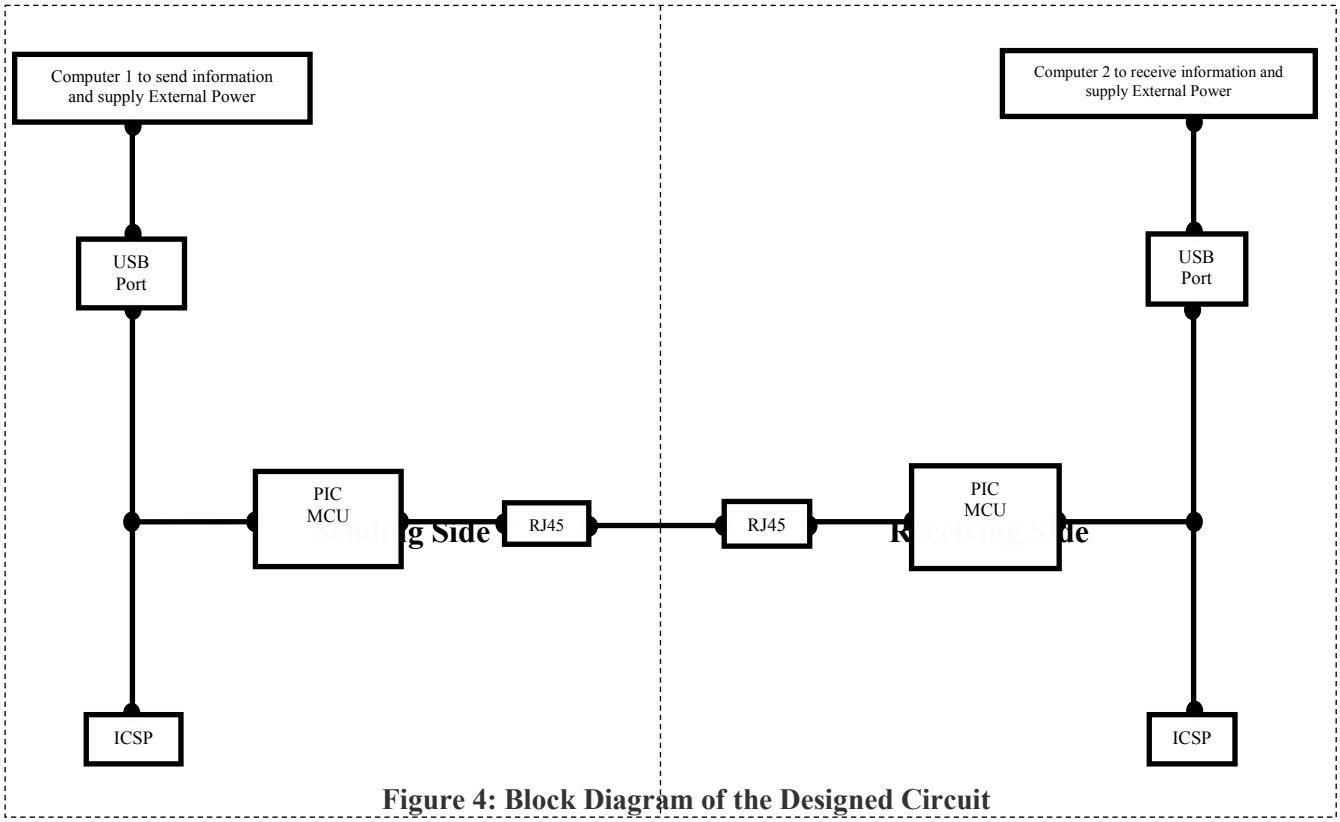

Figure 4: Block Diagram of the Designed Circuit

Implementation of the Designed Circuit

The PIC18F4550 Microcontroller

In this design, the PIC18F4550 microcontroller, which is a high-performance enhanced flash USB microcontroller from Microchip with nano-watt technology was used. This microcontroller offers the advantages of all PIC18 microcontrollers and incorporates a fully featured Universal Serial Bus (USB) communications module that is compliant with the USB Specification Revision 2.0. The module supports both low-speed and full-speed communication for all supported data transfer types. It also incorporates its own on-chip transceiver and 3.3V regulator and supports the use of external transceivers and voltage regulators.

USB Hardware Design for the Encryption System

This work used a pair of PIC18F4550 microcontroller module was used to establish connection between two computers through their USB ports. Figure 4.4shows the block diagram of the hardware section of the encryption system between the two computers. Computer 1 is connected through its USB port to the first PIC18F4550 microcontroller and the first PIC18F4550 microcontroller connected to the second PIC18F4550 microcontroller through its serial port.

The second PIC18F4550 microcontroller module is then connected to another host computer through its USB ports. The transfer and reception of messages are achieved bi-directionally in real time. Figure 4.5 shows the PIC18F4550 hardware circuit design. It consists of the PIC microcontroller as the heart of the module, a USB connector for connection to the host computer, an RJ45 plug for connecting the serial ports of the PIC18F4550 pair, six LED for visual indication of ongoing process and an in-circuit programming pins for downloading the firmware onto the microcontroller module.An important aspect of the design is the oscillator tanks which consist of a 20MHz crystal oscillator and matching capacitors. This ensures that the module work properly with an internal post-scalar used to increase the operating frequency to the 48MHz required speed for USB 2.0 protocol operation. The implementation of the design was achieved on a Printed Circuit Board which was done through a toner transfer process and chemical etching.

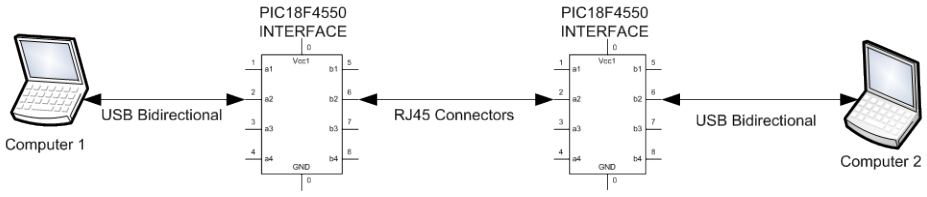

Figure 5: Block diagram of the hardware section of the encryption system between two computers

Hardware encryption procedure

The hardware encryption in this study was achieved using the bitwise logic operations in the registers of the microcontroller described earlier. In this study, messages are streamed as serial ASCII data from the PC through the USB port to the microcontroller unit. The encryption is then performed on each byte representation of the ASCII data using a pass key embedded in the microcontroller unit. The complexity of the encrypted data is a function of the length of the pass key string. First, the ASCII representation of the letter “A” is sent to the microcontroller, the microcontroller then left shift the bits by 1 which is equivalent to multiplying A = 01000001 by 2 to give 10000010 or 130 in decimal. The result is then bitwise **ExclusiveORed (EOR)** with the first character of the pass key “S” resulting in 11010001 or 209 in decimal which has an ASCII repre-

sentation of "Ñ". The result is preceded with the character "S" resulting in S209 or "SÑ" which is sent to the recipient's microcontroller. In principle, the entire encryption process for any given character requires 20 microcontroller clock cycles. Hence for the microcontroller running at $48MHz$ the time required for the encryption of a character is $0.42\mu s$.

Similar operation was performed for the next character "k" from the word **Akin**. This character is however **ExclusiveORed (EOR)** with the next character of the pass key which is "a". The **ExclusiveOR (EOR)** operation is repeated in cycles after every 4 character encryption operation. For increased security of the encrypted message, the pass key string length could be increased and module-up into a non-sensible sentence like "Sade maKeTr JHno EuaFulb".

The decryption process which can be performed either by the second microcontroller or using the PC software interface follows the reverse process of the encryption procedure. First, the encrypted character is striped of the preceding "S", and the result is then **ExclusiveORed (EOR)** with the appropriate character of the pass key. The resulting value is thereafter right shifted by 1 to get back the actual ASCII value of the character.

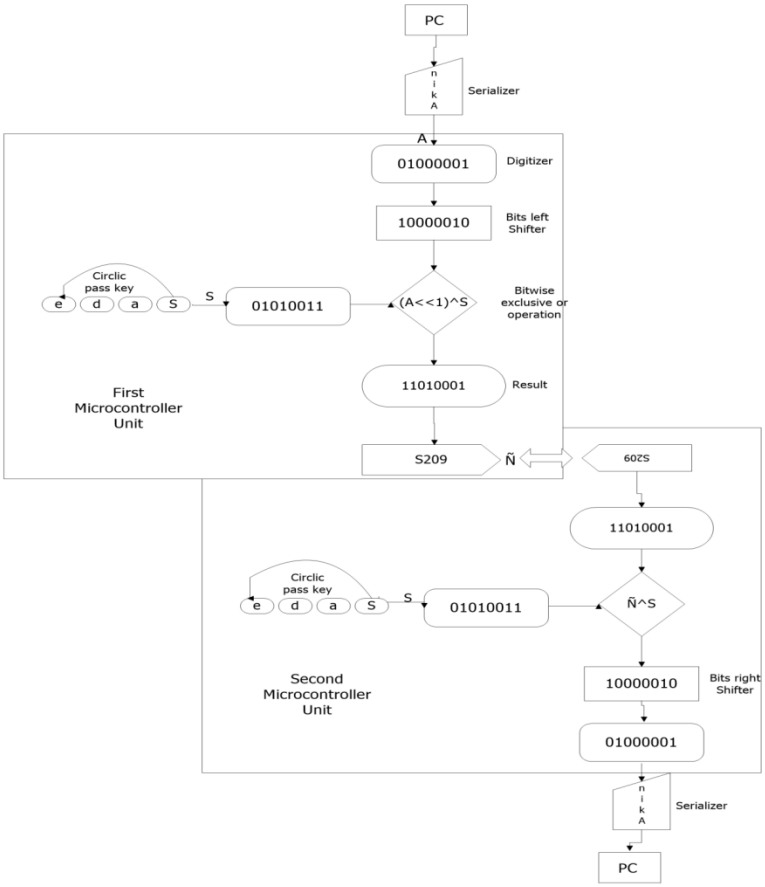

Figure 6: shows the PC software view of an encryption process for the message

The circuit system

The hardware designed for this work consists of Printed Circuit Board that hosts all other components making up the hardware. This Microcontroller is the heart of the hardware because it contains the program that carries out the encryption and decryption processes. It also has USB connector that connects the hardware to the computer. The Crystal Oscillator handles the speed of the hardware by increasing it to the required speed for USB 2.0 protocol operation. The RJ45 plug was used for the connection between the two hardware systems, while the Light Emitting Diodes

(LED) indicates presence of activity between the hardware systems. RED from Hardware 2 indicates that the system is searching for hardware 1. As soon as handshake is established with Hardware 1, the RED indicator will go off and the GREEN indicator will start blinking to confirm successful throughput from Hardware 2.

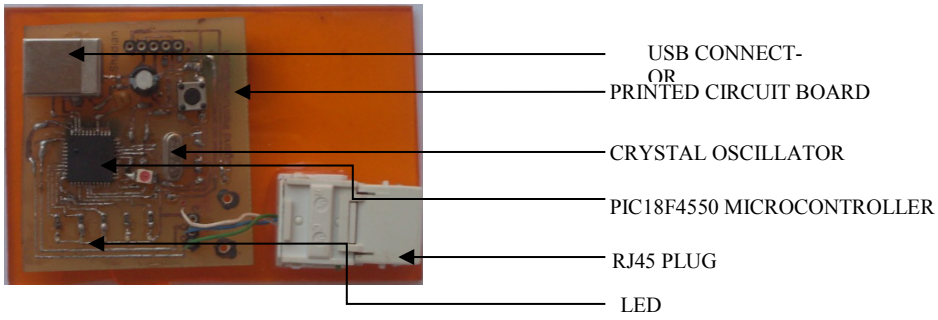

Figure 7: Layout on board of the Circuit System Designed

Figure 8 shows the connection for the complete Circuit System-Based Cryptographic device. Computer 1 displays the Graphic User Interface (GUI) for encryption process. The encryption process is carried out in the PIC18F4450 Microcontroller that communicates with the computer 1 through the USB communication port. After the encryption, the encrypted message is sent via the RJ45 plug from Hardware 1 and received via the RJ45 by Hardware 2 and Computer 2 receives the message via the USB communication port from hardware 2 where decryption is carried out.

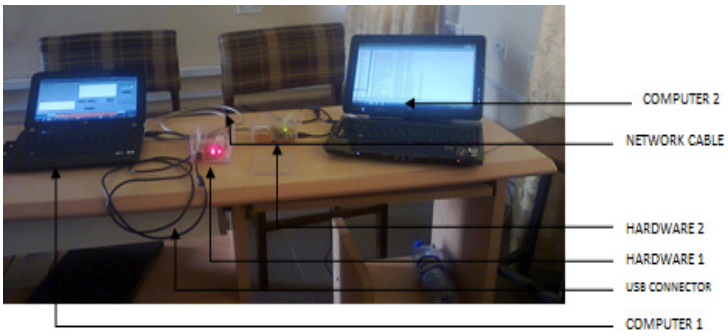

Figure 8: Plate of the Complete System

Results and Discussion

The Development of the Firmware Software

The firmware which is the instruction set that controls the whole functionality of the PIC18F4550, how it communicates with and transfers the encrypted data between the two computers was written using C language with a CCS C compiler.

The pair of module gets their power supplies from the USB port and the basic operation then begins with the connection of the module to the USB port of the host computer. If module 1 is the first to connect to its host computer, it receives power immediately and initializes its USB hardware. It then sends a connection request through its serial port to module 2 using RS232 protocol to confirm if module 2 is connected to a host. Module 2 if connected, will then send back a "connected" acknowledge message to module 1, thus a proper connection is established and all the

transfer and reception of messages from their host computers are handled through both hardware interrupt and a software interrupt service routine embedded in both processors. In the event that one of the modules is not connected to a host, the connected one loops around a wait-and-request connection cycle until connection is established.

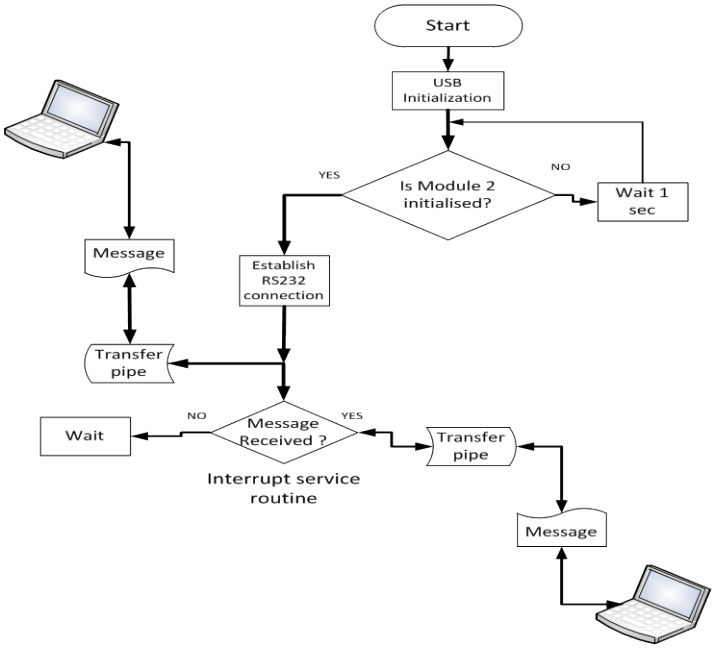

Figure 9: The flow chart describing the basic program design for the implementation of the project

The Development of User Interface Software

The software used for the sending and receiving encrypted messages was implemented on a windows 7 operating system and was written using Visual studio 2010 from Microsoft. Figure 4.5 shows the Graphical User Interface (GUI) which is used to establish connection to the hardware, send and receive encrypted messages.

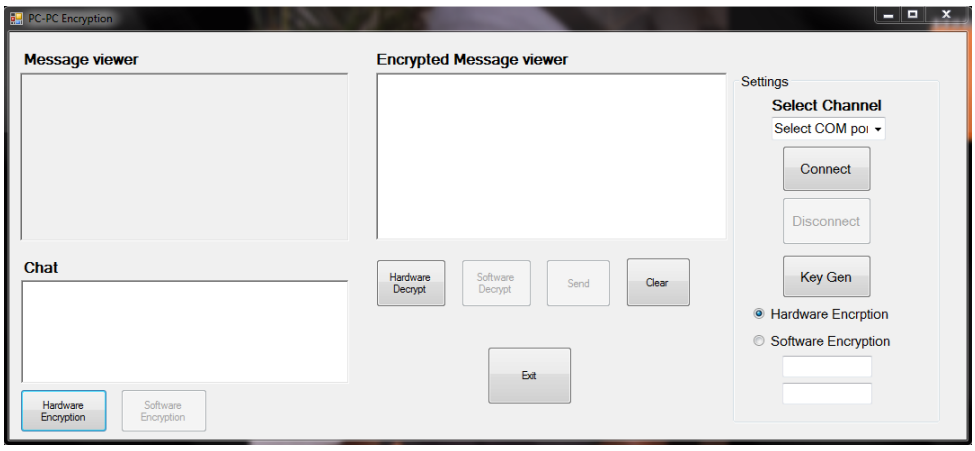

Figure 10: GUI for establishing connection to Hardware

The GUI software has two main interfaces the chart window and the message viewer for sending messages between any two users connected through the designed hardware. In case of sending encrypted messages, the hardware encryption button or software encryption located at the bottom of the Encrypted Message Viewer window is clicked depending on the type of encryption to be carried out, and the encrypted message appears as shown in figure 11 after it has been transferred via the USB port then to the designed hardware and through the designed hardware interface to the other Personal Computer, the decrypted message appears in the message viewer window as shown in figure 12.

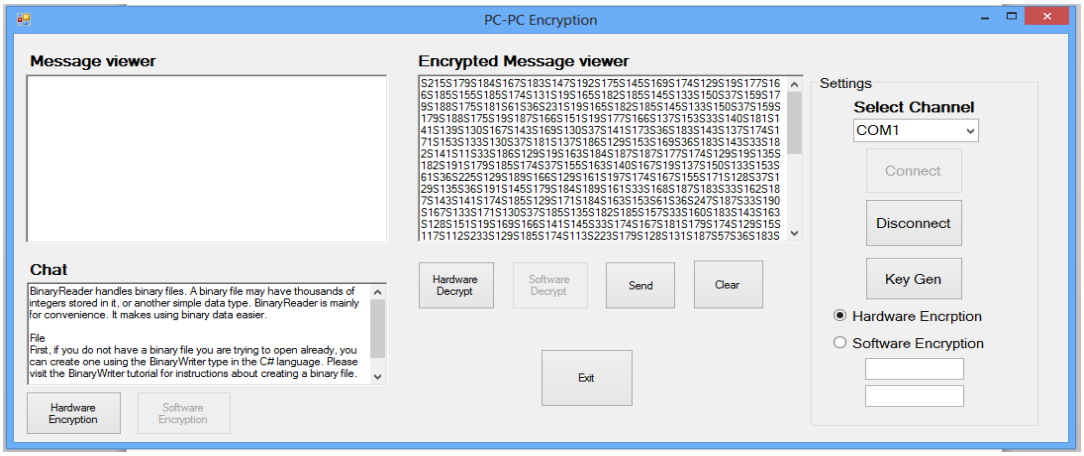

Figure 11: GUI Encrypted Message Viewer

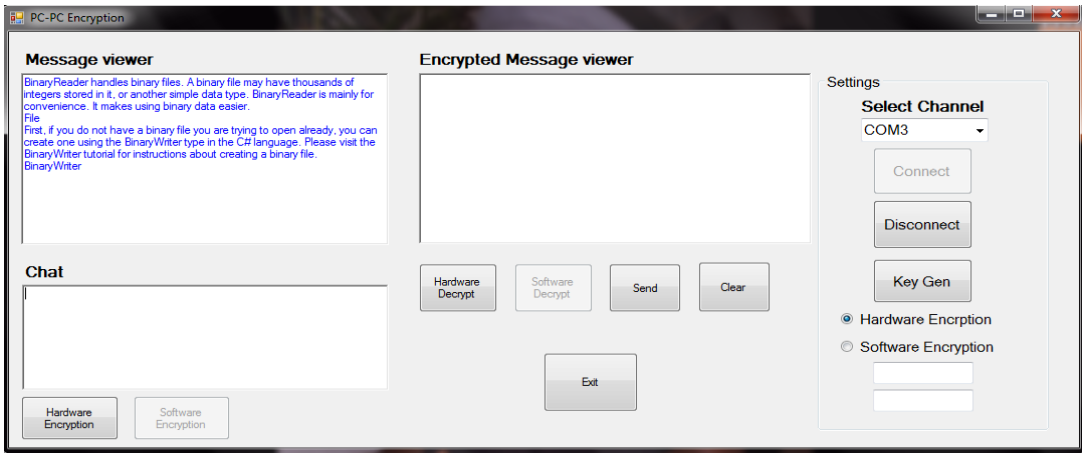

Figure 12: Message Viewer for the GUI

Comparison of Encryption Time for Software and Hardware Implemented in this Study

Figure 13 shows the comparison of the encryption time for the software (represented in blue colour) and the hardware (represented in wine colour). From these results, it could be observed that the encryption time for the software can be approximated by a polynomial function while that of the hardware followed a linear pattern. Also, the encryption time for the software is independent of the size of data thereby making the throughput unstable and so cylic in nature as shown in Figure 13.

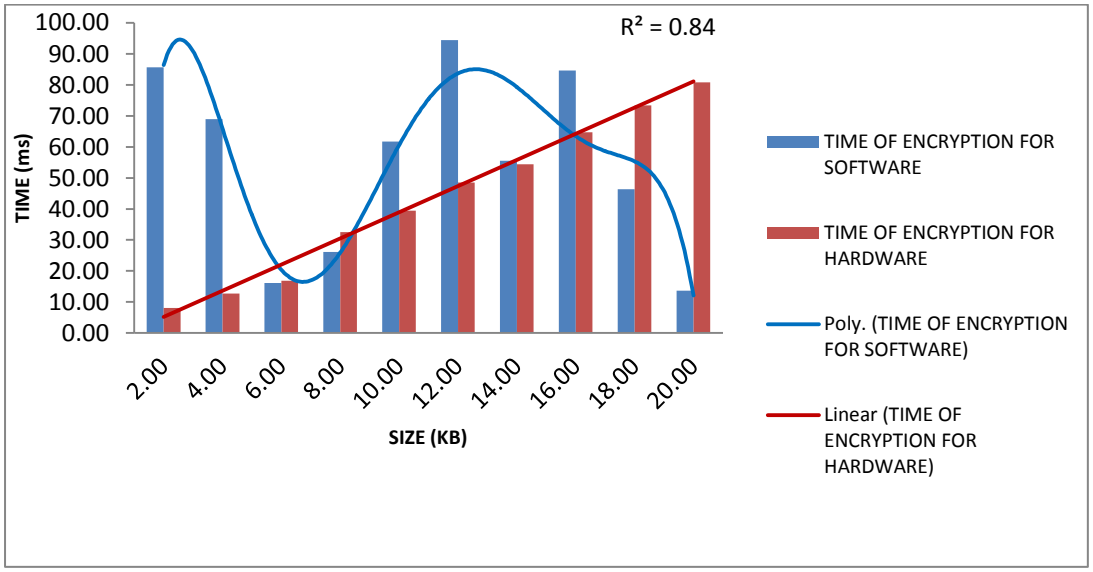

Figure 13: Comparison of encryption time for software and hardware implemented in this study

The result of the encryption for the software is characterized by a triple peak because the software lacks filtering process. By assuming a polynomial fit of order six, the trendline shows a correlation of 84% with the plotted data indicating that the curve is a good representation of the cylic nature of the time to encrypta certain amount of data size. On the other hand, the time to encrypt a defined data size in the hardware increases linearly with size. The reason for this may be due largely to the filtering techniques or procedures used in the design of the PIC18F4550 microcontroller used. Therefore, it is obviuos that the hardware encryption is more reliable and faster than the software encryption process because of the associated specialcharacteristics of the PIC18F4550 microcontroller used which makes it a logical choice for this study.

Conclusion

The study has designed and implemented a pair of hardware based cryptosystems capable of high speed transmission at low power consumption because, the components that make up the built hardware have some characteristics features of lower power operation. Also, it achieved a better security system compare to the software equivalent because it operates in a virtual way to the system that it is connected therefore does not live any trace after use. It also contains full speed and low speed compatible USB Serial Interface Engine which aids fast communication between devices.

References

Alese, B. K. (2000). *Vulnerability analysis of encryption/decryption techniques of computer network security.* M. Tech Thesis, Department of Computer Science, The Federal University of Technology, Akure.

Alese, B. K. (2004). *Design of a new public key cryptosystem using elliptic curve* Ph.D Thesis, Department of Computer Science, The Federal University of Technology, Akure.

Beuchat, J. L., Brisebarre, N., Detrey, J., & Okamoto, E. (2007). Arithmetic operators for pairing-based cryptograph. 9th International Workshop, Cryptographic Hardware and Embedded Systems-CHES 2007, Vienna, Austria. *Lecture Notes in Computer Science, 4727*, 239-255.

InfoSysSec. (2013). *Introduction to cryptography.* Accessed May, 2013 from www.infosyssec.com/infosyssec/cryptintro.htm

Kessler, G. C. (2010). *An overview of cryptography. Handbook on local area networks*. Retrieved October, 2014 from www.cs.princeton.edu/~chazelle/courses/BIB/overview-crypto

Kioskea. (2014). *Introduction to encryption with DES.* Accessed October, 2014 at http://en.kioskea.net/contents/134-introduction-to-encryption-with-des

Menezes, A. J., van Oorschot, P. C., & Vanstone, S. A. (1996). *Handbook of Applied Cryptography*. Published by CRC Press.

Microchip Technology. (2003). *In-Circuit Serial Programming™ (ICSP™) Guide*. Accessed May, 2013 at https://www.elnec.com/sw/30277d.pdf

Microchip Technology. (2007). *MICROCHIP PIC 18F2455/2550/4455/4550 Data Sheet 28/40/44-Pin, High Performance, Enhanced Flash, USB Microcontrollers with nanowatt Technology.* Accessed May, 2013 at http://ww1.microchip.com/downloads/en/DeviceDoc/39632D.pdf

O'Mahony, D, Michael, P. & Hitesh, T. (2001). *Textbook on electronic payment systems for e-commerce* (2nd ed.), 20-24. Artech House Inc.

Orlin, J. G. (2009). *The DES algorithm illustrated*. Accessed October, 2014 at http://page.math.tu-berlin.de/~kant/teaching/hess/krypto-ws2006/des.htm

Biographies

Akinwonmi, Folasade Caroline is a Technologist at the Centre for Space Research and Applications, Federal University of Technology, Akure. She obtained a Post Graduate Diploma and a Master of Technology degree in Computer Science in 2007 and 2014 respectively from the Department of Computer Science of the Federal University of Technology, Akure. Her area of interest is cryptography.

Alese Boniface Kayode is presently an Associate Professor with the Computer Science Department of the Federal University of Technology Akure, Ondo State, Nigeria. He holds a Ph.D. degree in Computer Science from The Federal University of Technology Akure, Ondo State, Nigeria in 2004. He has several awards of excellence. His areas of research include, Computer and Network Security, Quantum Computing and Digital Signal Processing. He is the current holder of the First Bank Professorial Chair in Computer Science of the Federal University of Technology, Akure, Nigeria.

Osuolale, A. Festus received the B.Tech. (with honours) and M.Tech degrees in Computer Science from the Federal University of Technology, Akure, Nigeria in 2002 and 2011 respectively. A former banker who has worked in the Information Technology unit of the United Bank for Africa Plc, Nigeria for several years before joining the teaching profession with the Federal University of Technology, Akure, Nigeria few years ago. He has since been involved in research and teaching of undergraduate courses in his home university with specializations in cloud computing, computer and internet security, networking and information risk and security, and biometrics. He is currently pursuing his PhD in the area of cloud computing with specifics in security risk assessment model in cloud computing. He is currently an Assistant Lecturer with some publications to his credit. He is happily married with children.