CRYPTOGRAPHY FOR COMPUTER SECURITY APPLICATION

Zirra P.B.[1], and G.M. Wajiga[2]

[1]*Department of Mathematical Sciences, Adamawa State University, Mubi*

[2]*Department of Mathematical Sciences, Federal University of Technology, Yola*

Abstract

This paper presents a cryptography application that is able to work with any type of file; for example: image files, data files, documentation files…etc. The method of encryption is simple enough yet powerful enough to fit the needs of students and staff in a small institution. The application uses simple key generation method of random number generation and combination. The final encryption is a binary one performed through rotation of bits and XOR operation applied on each block of data in any file using a symmetric decimal key. The key generation and encryption are all done by the system itself after clicking the encryption button with transparency to the user. The same encryption key is also used to decrypt the encrypted binary file. Experimental results were given to demonstrate the effectiveness of the application.

Keywords: encryption, decryption, key, rotation, xor.

Introduction

Cryptography helps protect data from being viewed or modified and helps provide a secure means of communication over otherwise insecure channels (Freeman, Neely, & Megalo, 1998; Diaa, Hatem, & Mohiy, 2010). For example, data can be encrypted using a cryptographic algorithm, transmitted in an encrypted state, and later decrypted by the intended party (Wikipedia, 2006). If a third party intercepts the encrypted data, it will be difficult to decipher.

The unencrypted data is referred to as the plaintext (Kessler, 2010) and the encrypted data as the ciphertext (Ritter, 2007), which is representation of the original data in a difference form (Freeman, Neely, & Megalo, 1998).

Key-based algorithms use an encryption key to encrypt the message (Kahate, 2008; Ibrahim, 2009). There are two general categories for key-based Encryption: Symmetric encryption which uses a single key to encrypt and decrypt the message and asymmetric encryption which uses two different keys – a public key to encrypt the message, and a private key to decrypt it (Agnew, Mullin, Onyszchuk & Vqanstone, 1995; Kak, 2009). Currently, there are several types of key based encryption algorithms such as: DES, RSA, PGP, Elliptic curve, and others but all of these algorithms depend on high mathematical manipulations (Beth & Gollmann, 1989; IBM, 1994).

One simple and good way to encrypt data is through rotation of bits or sometimes called bit shifting. But, rotation of bits is more advanced than simple bit shifting. In rotation of bits operation, the bits are moved, or shifted, to the left or to the right. The different kinds of shifts typically differ in what they do with the bits (Wikipedia, 2006). Another way is to perform logical operation on the bits of the file such as XOR operation. The idea behind XOR Encryption is that it is impossible to reverse the operation without knowing the initial value of one of the two arguments (Andy, 1998).

Preliminaries

There are several kinds of Encryption software in the market categorized by their functions and target groups. For example, some are single Encryption applications for files and database security; some are for messenger security or email Encryption applications that hide the actual text in the medium between the sender and the receiver (Baraka, El-Manawy and Attiya, 1998). One of the first types of Encryption was made by

Julius Caesar. Microsoft (2000) in his system, Caesar wrote B instead of A and C instead of B – so to a sentence "ABC" will be written in "BCD" (Wikipedia, 2006).

dsCrypt is AES/Rijndael file Encryption software with simple, multi-file, drag-and-drop operations. It features optimal implementation, performance and safety measures. dsCrypt uses an advanced Encryption algorithm and offers unique options for enhanced security (Dariusz, 2006).

NeoCrypt is a free, open-source File Protection Utility for Windows. It helps to protect sensitive information easily by encrypting it with password or key. It yields fast, reliable and unbreakable Encryption and supports many popular Encryption algorithms. All types of files can be encrypted like Audio, Video, Documents and Executables programs.

Neek protect is a software in the market right now with the ability to make Encryption on any files in the window platform, a key is set when one try to encrypt a files and the key will be used again when someone else trying to open the files been decrypted through decryption on the certain files (Vivek, 2006). Neek Protect is a good software operated under Microsoft windows because of the flexibility of this program's advanced features integration such as double click, file icons, .npt file extension etc.

This paper reports on a similar encryption technique that uses binary rotation of bits with XOR logical operation using a custom made encryption key that operates on any type of a file.

Methodology

The research included six (6) professional files. These were analyzed and then evaluated for their efficiency.

Key Generation

A symmetric encryption key is used for this application, which means the same key is shared for both encryption and decryption. A copy of the generated key is saved in a file named .ekf during the encryption process and the same key is used as the decryption key to retrieve the encrypted file. The technique of generating the key uses two methods: random number generation and combination as follows.

a) A long number with only digit values called *x* is generated.
b) Another long number with character values called *y* is generated. The size of *y* is twice the size of *x*.
c) An insertion operation is performed such that each digit of *x* is inserted after two characters of *y*. The result of the insertion is called *z*.
d) Another only digit number called *d* is randomly generated. *z* is combined with *q* by placing alternately one character or digit from *z* after a character or a digit from *q*. The result of the combination is a relatively strong key.
e) An odd and even partitioning is performed on the key. The position of each character in the key decides it to be an even or an odd character. For example, the character at position 0 is an even one while the character at position 1 is an odd character. Similarly, position 2 is an even position while position 3 is odd one. The even part of the key is combined together and the odd part of the key is combined together.
f) The two parts of the key are joined as an even part followed by an odd part to produce one final encryption key. Since the final key is a key that consists of all characters, another key with only ASCII values of each character is obtained. The result is a very long decimal key. Figure 1 shows the complete key generation process.

Encryption Rules

For the encryption method, a single digit in the decimal ASCII representation will decide which encryption method is to be applied to the single a binary block in a file.

i. *0* in the key means a rotation of bit to the left is performed and the next integer to *0* in the ASCII code decides how many bits the block will be moved to the left.

ii. *1* in the key means a rotation of bit to the right is performed and the next integer to *1* in the ASCII code decides how many bits the block will be moved to the right.

iii. *2* means the block will be passed to an *XOR* encryption to be performed with a binary block from the file.

Else, all other numbers in the key like 3,4,5,6,7,8,9 are ignored.

File Types

There are no limitations of the type of files accepted for encryption in this application, which means any type of a file such as data files, audio files, video files or image files can be encrypted by the application. This is because all the files are encrypted at the binary level. There is also no limitation of the size of the file that can be encrypted using this application, which provides flexibility to the user. The encrypted file can only be opened and viewed after it has been decrypted to its original file using the symmetric encryption key. The above rules are summarized in the encryption chart shown in figure 2.

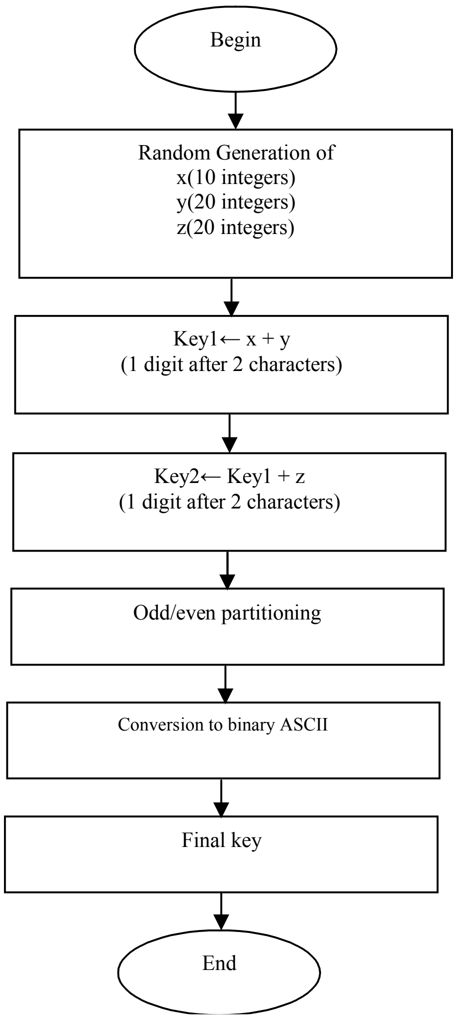


FIGURE 1: Key Generation Flowchart

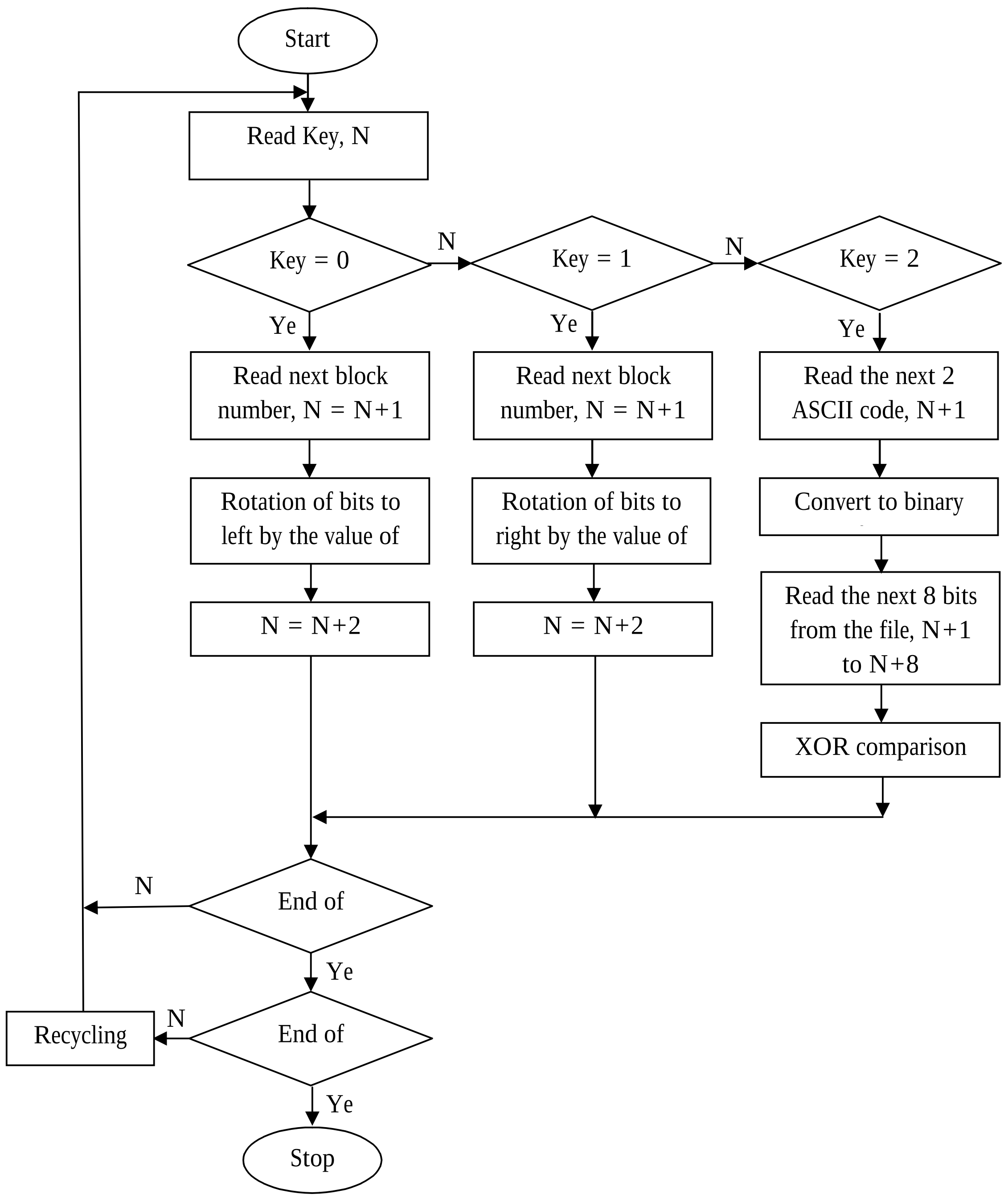
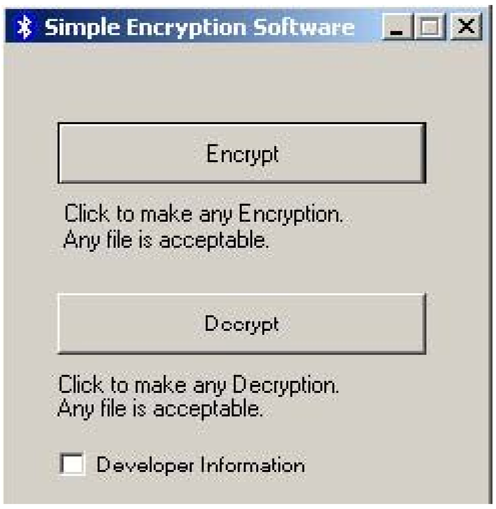

Start
Read Key, N
Key = 0
N
Key = 1
N
Key = 2
Ye
Ye
Ye
Read next block number, N = N+1
Read next block number, N = N+1
Read the next 2 ASCII code, N+1
Rotation of bits to left by the value of
Rotation of bits to right by the value of
Convert to binary
N = N+2
N = N+2
Read the next 8 bits from the file, N+1 to N+8
XOR comparison
N
End of
Ye
Recycling
N
End of
Ye
Stop

Figure 2: Encryption Flowchart

Results

The interface of the application is simple enough to be used by any user. Figure 3 shows the interface with the encryption and a decryption buttons. The encryption is performed simply by choosing any file while decryption is executed by choosing an encrypted file with an appropriate key. Figure 4 and 5 shows a successful encryption and decryption, respectively.

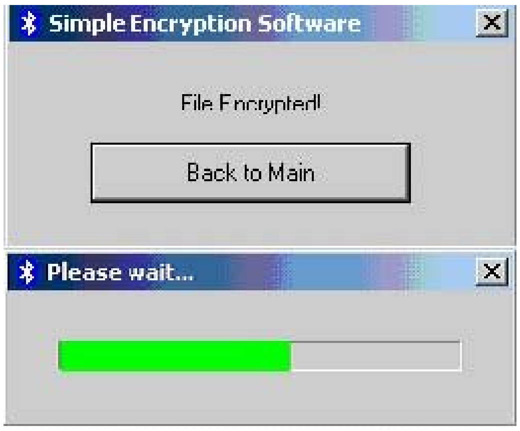
Figure 3: Shows the Encryption/Decryption interface.



Click to make any Encryption.
Any file is acceptable.

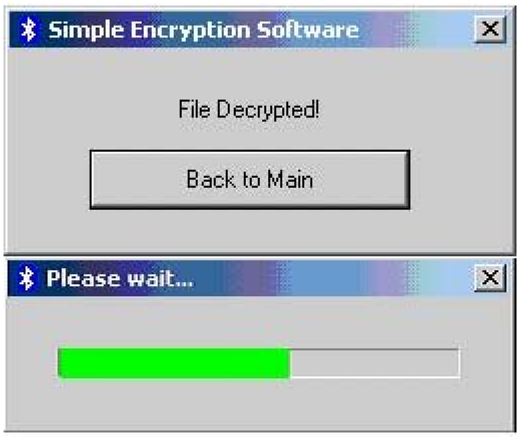
Click to make any Decryption.
Any file is acceptable.

Figure 4: Shows a successful encryption and the encryption progress bar.



File Encrypted!

Figure 5: Successful decryption and the decryption progress bar.



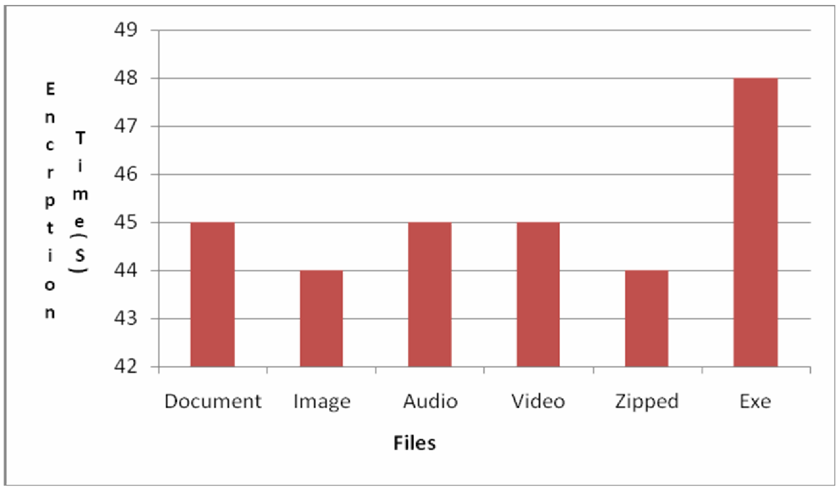
Application Testing & Performance Checking

Application testing is applied to the entire application with multiple application features to make sure the application can encrypt all types and all sizes of files. Successful testing means the application is user-friendly and comfortable to be used by all range of target users. For performance checking, the application was tested with different types and sizes of files and the performance of the application was rated by computing the time required for encryption of the files. Also, the reliability of the application was examined by the success rate of encryption. A successful execution means an encrypted file is not visible by others; also successful execution means a decrypted file was obtained using a key and an encrypted file. Table 1 shows the testing of different types and sizes of files. Figure 6 shows the encryption time for six different types of files that are of the same size of 5 Mb. It can be seen that the encryption time is similar for all the files especially when the file size is small. The small size of files is a typical example of the use of this application as it is mainly targeted for small university campus. Most of the documents used in this environment are of text type with some figures inside the text; therefore, the sizes of the files may not go over few mega bytes.

Table 1: Testing of the Application with Different Types and Sized Files.

File Types	File Size(Mb)	Encryption Time (S)	Success Rate
Document	1/ 3/ 5	9/27/45	100%
Image	1/ 3/ 5	10/26/44	100%
Audio/ Video	1/ 3/ 5	18/28/45	100%
Zipped	1/ 3/ 5	10/25/44	100%
Exe.	1/ 3/ 5	12/27/48	100%

Figure 6: Graphical view of the encryption time for different types of files.



Conclusion and Future Work

A new simple tool has been created, which is targeted for use inside a small institution such as a small university for lecturers' daily use of sending exam files and sensitive material such that the material can be encrypted and the file is sent in one e-mail while the encryption key is sent in another e-mail or via any secure communication channel. The encryption application developed and described in this paper might not be comparable to well-known encryption algorithms but its simplicity and availability proves that tools can be developed that fit the needs of an institution without resorting to purchasing expensive software from the market. For future enhancement to this application public key encryption can be applied where two keys can be generated: one to encrypt a file using the public key and another private key to decrypt it. Also, other more advanced encryption operations can be included to enhance the security of the application so that it can be used to encrypt more sensitive administrative material in an institution.

References

Agnew, G. B., Mullin, R. C., Onyszchuk, I. M., and Vqanstone, S. A. (1995). An implementation for a fast public-key cryptosystems. *Journal of Cryptology*, 3(2), 63-79.

Andy, W. (1998). Tips and Tricks: XOR Encryption Available at: http:// www. andyw. com /director /xor. Asp.

Baraka, H., El-Manawy, H. A., and Attiya, A. (1998). An Integrated Model for Internet Security Using Prevention and Detection Techniques. *IEEE Journal of Computer and Communication*, 99, 25-33.

Beth, T. and Gollmann, D. (1989). Algorithm Engineering for Public Key Algorithms. *IEEE Journal on Selected Areas in Communications*; 7(4), 458-466.

Dariusz, S. (2006). Free Software copyright 1997 -2006. Available at: Error! Hyperlink.

Diaa, S. A. M., Hatem, M. A. K., & Mohiy, M. H. (2010). Evaluating the performance of symmetric encryption algorithms. *International journal of network security*, *10*(3), *213-219.*

Freeman, J., Neely, R., and Megalo, L. (1998). Developing secure systems: issues and solutions. *IEEE Journal of Computer and Communication*, 89, 36-45.

Ibrahim, M. H. (2009). Receiver-deniable public-key encryption, *International journal of network security*, *8*(2),159-165.

IBM (1994). The Data Encryption Standard (DES) and its strength against attacks. *IBM Journal of Research and Development*, 38, 243-250.

Kahate, A. (2008). *Cryptography and network security (2nd ed*.). New Delhi: Tata McGraw Hill.

Kak, A. (2009). Classical encryption techniques. Lecture notes on "computer and network security" Purdue University.

Kessler, G.C. (2010). Handbook on local area networks: An overview of cryptography. United Kingdom: Auerbach. Retrieved January 3, 2010 from http://www.garykessler.net/library/crypto.html.

Ritter, T. (2007). Crypto glossary and dictionary of technical cryptography. Retrieved August 17, 2009 from *http://www.ciphersbyritter.com/GLOSSARY.HTM*

Vivek, T. (2006). NeekProtect. Available at: http://neekprotect.sourceforge.net.

Wikipedia (2006). Encryption. Available at: http://en.wikipedia.org/wiki/Encryption, on 13 December 2006.