

ASSIGNMENT TOP SHEET

Faculty of Creative Arts, Technologies & Science

Department of Computer Science & Technology

Student Ref. No 1030483	Unit Code **CIS000-6**
Unit Name **MSC Project**	Deadline for Submission(s) **25th May 2012, by 4 pm**
Student's Surname Zubair	Student's Forename Zaland
Unit Leader's Name **Dr Fiaz Hussain**	Supervisor's Name Dr. Petros Karadimas

Assignment Details:

MSc Project – Final Thesis Report

Instructions to Student:

Please note: Work presented in an assessment must be the student's own. Plagiarism is where a student copies work from another source, published or unpublished (including the work of a fellow student) and fails to acknowledge the influence of another's work or to attribute quotes to the author. Plagiarism is an academic offence.

Work presented in an assessment must be your own. Plagiarism is where a student copies work from another source, published or unpublished (including the work of another student) and fails to acknowledge the influence of another's work or to attribute quotes to the author. Plagiarism is an academic offence and the penalty can be serious. The University's policies relating to Plagiarism can be found in the regulations at http://www.luton.ac.uk/livingandstudying/qa/documents. To detect possible plagiarism we may submit your work to the national plagiarism detection facility. This searches the Internet and an extensive database of reference material including other students' work to identify. Once your work has been submitted to the detection service it will be stored electronically in a database and compared against work submitted from this and other universities. It will therefore be necessary to take electronic copies of your materials for transmission, storage and comparison purposes and for the operational back-up process. This material will be stored in this manner indefinitely.

I have read the above information and I confirm that this work is my own and that it may be processed and stored in the manner described.

Signature: ..... **Date: .....**

Extension deadline

CAAS agrees that the assignment may be submitted ____ days after the deadline and should be marked without penalty.

CAAS confirmation.....

Please leave sufficient time to meet the given deadline and do not leave the handing-in of assignments to the last minute. You need to allow time for any system problems or other issues.

MATLOG: A Feistel Based Poly-Alphabetic Encryption Algorithm

A thesis submitted at University of Bedfordshire
In partial fulfilment for the degree of
Masters of Science
in

Mobile Computing

Zubair Zaland

1030483

Supervisor: Dr. Petros Karadimas
Department of Computer Science and Technology



University of Bedfordshire

May 2012

Acknowledgement

First of all I would like to acknowledge Dr. Petros Karadimas, my supervisor for his support and guidance at every step of the project. I owe my gratitude to Dr. Gregory Epiphaniou, Dr. Song Yan for conveying the knowledge of the subject and finally my friend Mr. Akbar Aarif, a MSC Computer Networking student at university of Bedfordshire for his guidance in mathematics related issues during the project.

Declaration

I hereby declare that this project is my own unaided work and it is being submitted to the university of Bedfordshire for the degree of MSC Mobile Computing. This work has not been submitted anywhere else for any other degree. All sources of information have been acknowledged in the thesis.

Name: Zubair Zaland Signature:

Date:

Abstract

One might question, "What makes MATLOG different from other feistel based ciphers?", the answer is the encryption methods that are both Feistel and non Feistel. First of all the idea of poly-alphabetic encryption is unique to feistel approach. The capabilities like generating round keys from a single key like Advance Encryption Standard (The Rijndeal Algorithm) but that of variable lengths and the AES itself not being feistel based. The idea of security through obscurity, although not implemented with its true meaning but MATLOG does not use the user key directly for encryption and the computer generated keys do not directly encrypt the data not even letting the user himself know how it is done. The artefact is currently set to 10 rounds of encryption whereas within each round it performs 2 - 6 rounds of substitution and one round of transposition using two methods of transposition joined as one. In this document I am going to discuss the various standards of encryption and various modes of operation they work on. Then discuss the survey I conducted online in which the university students and other people on social networks having the knowledge of the subject took part, Followed by the design of the algorithm Itself which involves splitting of the plaintext first of all; Splitting the block here means creating two separate blocks out of random numbers and the user data, topic is discussed in further detail under the heading of design. after splitting the blocks we apply transposition to one block and substitution to the other and vice versa to the result of the first round this process goes on for ten rounds and finally we have our cipher text when the result of the two encrypted blocks is merged into one. The decryption process is the same only the keys decrypt the opposite way i.e. the last key decrypts first. The process of key management is very crucial, even though the user key is hardly used in encryption the application has to maintain the user key and the automatically generated keys intact until decryption successfully takes place. MATLOG is a highly flexible and dynamic algorithm it can be set to transposition in several ways, the keys are always different and they process the poly-alphabetically generated data, the keys are also of different size and value for every round within substitution function making the substitution half highly flexible too and best of all it can be commercially implemented both on hardware and the application layer. It can also be set to perform cryptography on storage devices and not only communications. The design of MATLOG is based on independent blocks eliminating the block dependencies like that of cipher block chaining (CBC) , this allows the algorithm to support parallel processing making it as efficient as it would be as any stream cipher algorithm. I have eliminated the Initialization vector in this algorithm and replaced it with the One-time pads making it stronger and leaving no patterns of encryption which was one of the flaws on initialization vector, it needed frequent management one of the reasons that caused the WEP 1 protocol to fail was the short size and mismanagement of the initialization vector, instead my idea of spreading blocks is highly random and the transposition and substitution functions reinforce the safety to the my idea of IV's replacement.

Table of Contents

Acknowledgement 4
Declaration 5
Abstract 6
Introduction 9
Aims and Objectives 9
Literature Review 9
I. BLOCK CIPHER MODES OF OPERATIONS 9
II. PADDING 11
III. Mono-alphabetic and Poly-alphabetic Ciphers: 11
IV. FEISTEL CIPHER: 11
V. One-time Pads: 11
VI. AES 12
Survey Analysis: 12
The Sample 12
Questions 12
Design 16
Splitting blocks 16
Transposition 1 17
Folding Matrix 17
Inverse Whirl 18
Key Management 19
The Encryption Function (F) 19
The Overall Design of MATLOG 20
Development & Implementation 21
Programming language 21
Splitting blocks 22
Transposition 1 23
Folding Matrix 23
Inverse Whirl 23
Folding Inverse Whirl 24
Key Management 25

The Encryption Function (F) 25
Testing 26
Test plan: 26
Items to be tested: 26
Test approach: 27
Test cases: 27
Conclusion 30
References: 31
Appendix 32
The survey questionnaire: 32
The source code for modules 32
Project Poster 37
Market Survey Returns 38

Introduction

Cryptography, the art and science of hiding information, transforming it into a form that only the person to whom it was originally meant to can understand it and the aim is always to keep it safe from others eavesdropping into it. The art started from simply writing the information down as there were not many literate people at the time and got more complex over the time. Many known people contributed to the subject that includes the Caesar cipher from the Romans another one from Charles Magne's cipher, Mary Queen of Scots. and many more.

MATLOG is an algorithm I have designed based on the Horst Feistel's encryption method, it is a magnificent design which provided the bases for many well known encryption algorithms known to man today. MATLOG as the name shows is the combination of both mathematical and logical operations. There are many algorithms present that make use of only one of those operations such as the elliptic curve cryptography, which uses the mathematical equations for encoding data. The number of reversible Logic gates is limited therefore they provide vast but countable number of permutations to decode an encryption making it easy for the ever increasing efficient computers of the day to crack; therefore, by applying the mathematical equations onto the resultant of the logical operation exponentially increases the permutation of possible combination for the encrypted data.
In this report I am going to discuss the design I have developed based on feistel cipher and using strengths of other known methodologies and blending them into something better.

Aims and Objectives

The basic objective of this project is to devise an algorithm that has the competitive features of the known standard algorithms and is secure and does not contain the flaw of predictability.
In order to achieve that task I would have to complete the following task:

1. Study the known technologies and document the essential details related to the completion of the project.
2. Based on the study devise a poly-alphabetic encryption algorithm.
3. Implement the algorithm in order to simulate the various steps of encryption.

Literature Review

If we divide the study of cryptography in terms of processing we have two major types of encryptions: the stream cipher and the block cipher. The stream cipher is normally faster to execute than most of the block cipher modes of operation because it can be performed in parallel while in Block cipher modes like ECB each block of plaintext is dependent on the output of the operation performed on the previous block, Hence, it cannot be executed in parallel. On the other hand block ciphers are stronger against direct attacks and requires more resources in order to perform an efficient attack (Baldwin, 2008) as compared to stream ciphers.

I. BLOCK CIPHER MODES OF OPERATIONS

Block ciphering is mechanism works on the division of data into blocks of equal size and then encrypting them using a key or a set of keys. It is a private key mode of cryptography also called Symmetric Key Cryptography that encrypts 128-bit plain text data with a 128-bit, 192-bit, or 256-bit cipher key (Qin, 2005). This involves a single key or a single set of keys generated by the person performing encryption and known only to the person authorized to see the content. The operating mode of an encryption algorithm allows block ciphers to output messages of arbitrary length or turns block ciphers into self-synchronizing stream ciphers, which generate a continuous key stream

to produce cipher texts of arbitrary length (Diesburg, 2008). Block ciphering initially came up with serious limitations such as similar blocks encrypted with same key returned similar blocks of cipher text. These issues are catered using different modes of operations given below.

A. ECB

The simplest of the block ciphering mechanisms (Electronic CodeBook) encrypts each block of data separately using the same key. This method is fast but produces similar blocks of cipher text as mentioned above exposing the pattern of the data (Ehrsam,1976).

B. CBC

Cipher Block Chaining performs XOR operation to each block before it is encrypted with the key. The initial block is XORed with a user defined initial vector which comes with better security as the initial works as a separate key for the initial block. Due to the well-known birthday attack and the small input length, the CBC mode becomes less secure as the number of data blocks to be encrypted increases. This leads to a challenging task, namely to design schemes for storage device block or sector level data encryption that are efficient and do not have the disadvantages mentioned above (zheng, 2005). CBC is the most commonly used method but its main issue is its efficiency as due to its sequential mechanism of ciphering it would take same amount of time on a single processor as on multiple processor reason being that it cannot be parallelized. CBC encryption becomes insecure once 2n/2 blocks have been encrypted, in the sense that at this point partial information about the message begins to leak, due to the well-known birthday attacks where n is the cipher-block length (zheng, 2005).

C. PCBC

Propagating Cipher Block Chaining commonly known as Plaintext CBC uses the same method of encryption using initial vector with the first block but for the next block it performs the XOR of the Plaintext and the cipher text of the previous block this method is also sequential and has same issues as CBC [1] .PCBC isused in advance algorithms like Kerberos although critisized alot on selection of this mode of operation when faster and more secure options were available (Rogaway, 2003).

D. CFB

Cipher Feedback operation works quite the opposite of CBC although it is almost same during decryption, here the initial vector is encrypted using the key and then XORed with the first block of the data. While the following blocks of cipher text are produced by encrypting the cipher text of the previous block and then XORing it with the plaintext block. the main drawback of just XORing the encrpyted key with the plain text is that it can be cracked in a few minutes of time just by XORing it back again with randomly generated 128 bit key with the cost of O(n exp n).

E. OFB

Output Feedback is almost same as CFB it does not encrypt the plaintext at any stage just performs XOR with the encrypted output of initial vector and the key. ”Both the support for random access and vulnerability characteristics of the OFB mode is the similar to the counter mode in the storage context, since the key stream is based on only the original key and Initialization Vector” (Ehrsam, 1976). The same output is forwarded to be encrypted again with the key of the following blocks and XORing with the respective plaintext blocks. Just like the CTR mode,” the OFB mode leaks information about the plaintext from the first changed block to potentially the last block in the file or extent” (Ehrsam, 1976 , Diesburg, 2008).

F. CTR

Counter or Segmented Integer Counter (SIC) generates a set of values through a function for each block and encrypting those values with a key and XORing the output with the plaintext. This method

is fast as it is parallel and independent of other blocks output. CTR is the most secure and suitable for most cases provided that the security of the counter function is ensured. ”Unlike CFB, the counter mode supports random access, where an in-place update does not require the remaining file to be re-encrypted. However, since the key stream for the entire file can be reused, different versions of the file can be XORed to reveal changed information” (Diesburg, 2008).

II. PADDING

The Last data blocks that are usually short of data bits are simply completed by adding null bits at the end of data stream until the block of 128 bits is complete. This is one of the simplest methods of the technique called padding. The Problem with this padding method was keeping account of the padded bits, because when the data is decrypted the data stream might also have null bits so one has to differentiate between null bits of data and padded bits.

III. Mono-alphabetic and Poly-alphabetic Ciphers:

The mono-alphabetic ciphers are ones in which a certain plaintext and a key would always produce a similar cipher text provided that the encryption function remains the same. This type of encryption gave birth to many concepts such as cipher/plain text, Symmetric cryptography etc (GARC´IA-PASQUEL, 2006). The classic Caesar cipher is an example of mono- alphabetic ciphers and the free masons' cipher where every alphabet is replaced by a symbol where as the symbol remained the same in every case.

The poly-alphabetic ciphers generally considered more tricky than its counterpart gives a different output for the same key and plaintext every time. This type of encryption became famous when most of the early 20th century mono-alphabetic encryption methods were cracked by frequency analysis. The Alberti cipher was considered one of the most secure poly-alphabetic cipher until the predictability of the substitution was calculated for each character.

IV. FEISTEL CIPHER:

Feistel cipher is a symmetric key block ciphering algorithm named after the cryptographer Horst Feistel. It is a methodology used by many well known cryptographic algorithms such as DES, Blowfish, 3DES, Two fish, RC5 , Lucifer etc. The mechanism basically divides the data block into two sub blocks L, R, Such that: g(**L**^i-1, **R**^i-1, **K**^i) = (**L**^i, **R**^i). (Stinson, 2006).

Different Algorithms implementing feistel use different block and key sizes most commonly 128 bits. The encryption function is called iteratively using the method mentioned above therefore feistel is called as an iterative cipher and the iterative function is called round function.

V. One-time Pads:

This type of encryption is performed using a set of random letters equal to the size of the data. Each random letter on the pad is used to encrypt only one letter of plaintext. Once the data is encrypted the set of random letters responsible for encryption is destroyed on encryption end. The legitimate receiver of the encrypted data is supposed to have an identical pad which is used to decrypt the data. The letters used to decrypt are also destroyed at the end of the process (Kahn, 1967).

The beauty of this algorithm is that a random set of data is used as a key and every set is discarded once used so the key is ever-changing making it impossible to perform the cryptanalysis of the data, in other words the algorithm is unbreakable provided the pad is secure. But on the other hand this algorithm also has its limitations, it cannot be commercially implemented on devices unlike AES and other algorithms that are built-in on devices like WIFI-modems etc. because the size of random data required to be generated and destroyed on multiple Mbps internet connections is economical and insecure (Schneier).

Another problem is if you want this algorithm to work commercially that means the encryption pads should be available to everyone using the algorithm (including the hacker) in that case there is no point in encryption its only matter of intercepting the data and running through person's own pad.

VI. AES

Advance Encryption Standard, originally Rijndael algorithm (Daemen, 1998) is considered to be the most secure mono-Alphabetic ciphers. AES is not a feistel based cipher but is iterative and block cipher. This algorithm was declared AES in 2001 by National Institute of Standards and Technology (NIST). Its Strengths over other candidates of AES include efficiency, performing both Substitution and Transposition, multiple rounds of encryption based on the block size of 128 and key sizes of 128,192,256, and 512 (Mohd., 2011) bits and multiple key generation based on single key for each iteration of encryption.
The transposition part of the cipher works by shifting rows, mixing columns and XORing round key considering the data to be in a matrix format where each column takes 1 byte of space.

Survey Analysis:

A short and relevant survey was designed using the online survey provider ***www.surveymonkey.com*** this survey consisted of only five questions that would be discussed later under this topic with each question followed by its relevance and the survey response. The aim of the survey was to gain people's view about the types of cryptography and their priorities when it comes to choosing between them. The survey link was sent to the people via university email and posted on the social networks. During the period of 3 weeks we only got 35 responses the results of which are discussed below:

The Sample

The survey was designed only for the people with some knowledge of the subject therefore I sent the email only to students of department of computer science with instructions to fill only if they have the knowledge of cryptography.

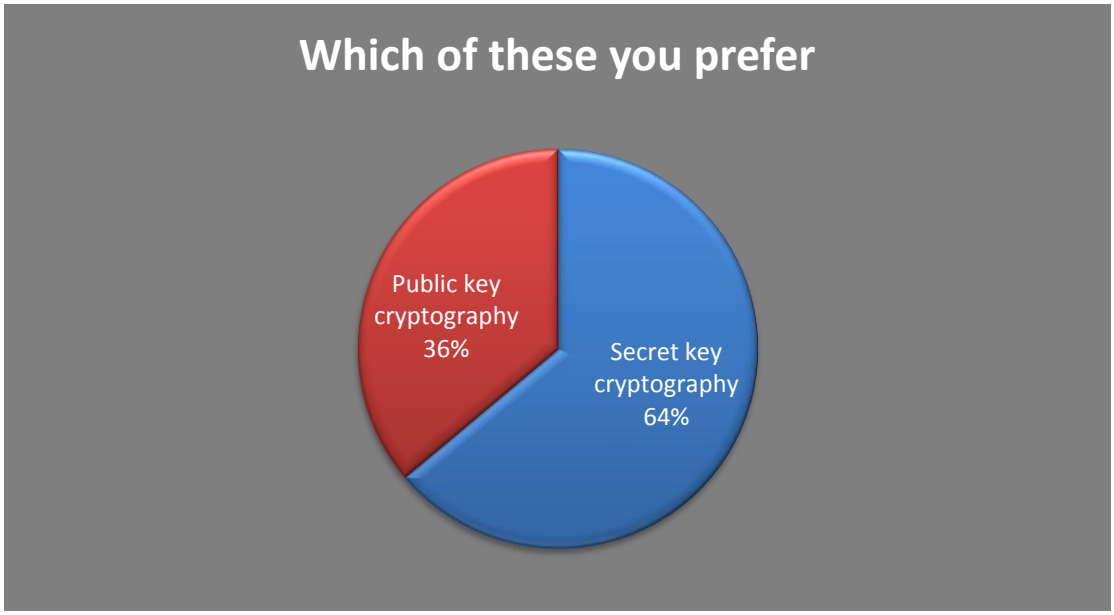
Questions

1. What type of encryption do you prefer?

- Mono-Alphabetic
- Poly-Alphabetic

Response:

This question determined which of the types of algorithm has more of user trust and it turned out to be poly-alphabetic cipher not too closely beating its counterpart. I believe it is due to the frequency analysis method that broke many mono-alphabetic ciphers so when you ask about the ciphers as a whole the people tend towards poly-alphabetic ideas.

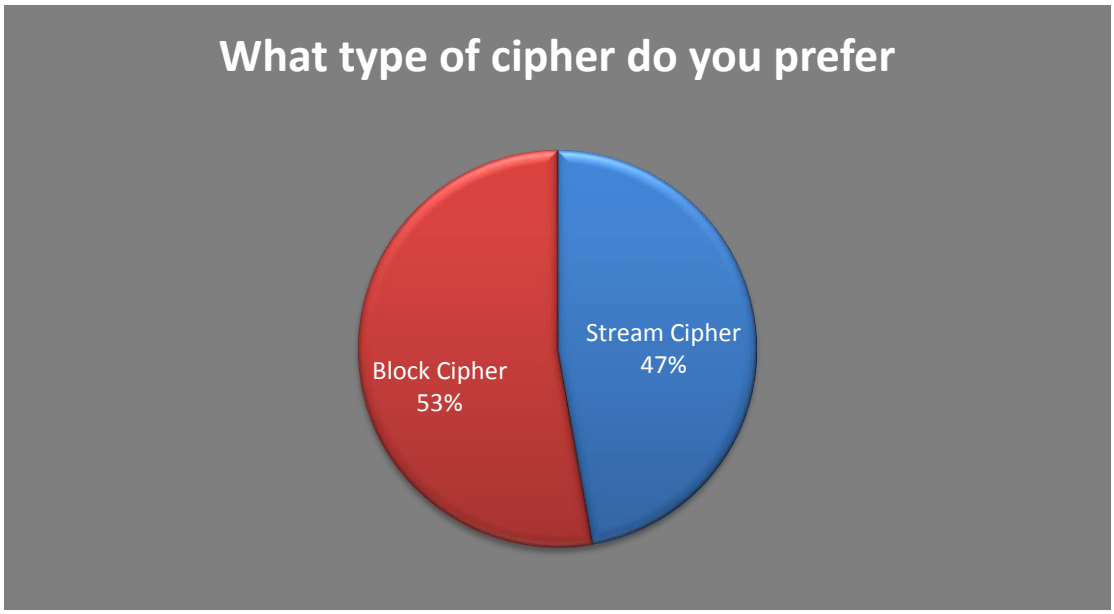


2. What type of Cipher do you prefer?

- Stream cipher
- Block cipher

Response

In order to implement a feistel based encryption algorithm I had to use the block cipher mode but also had to know the response of people about its counterpart and the results were too close with a difference of only 2 responses the block cipher stood people's favourite.



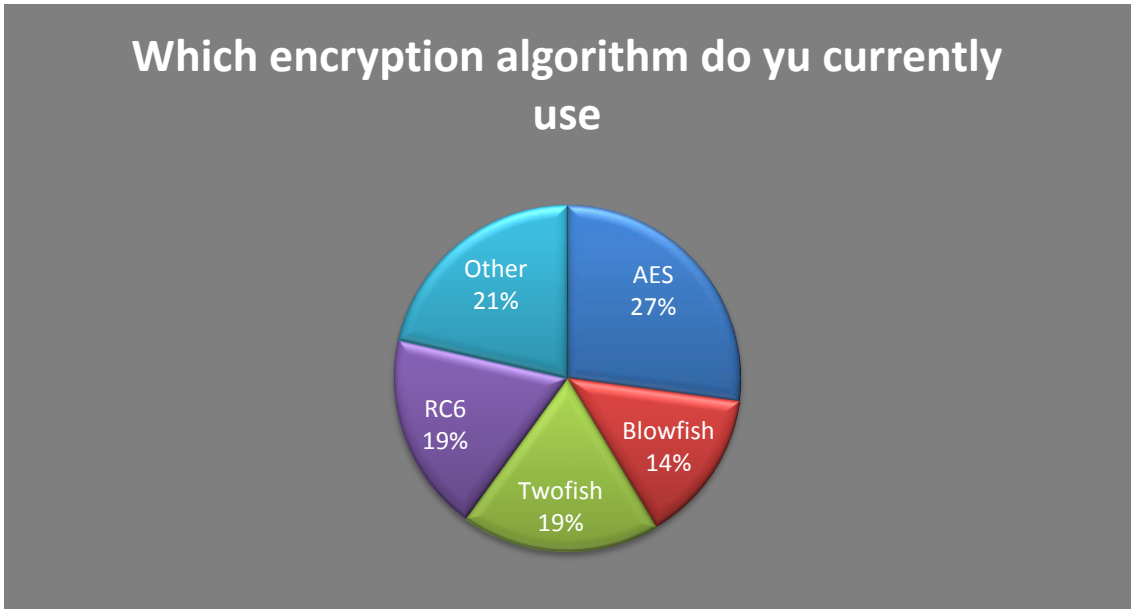
3. which encryption algorithm do you currently use?

- ☐ AES
- ☐ Blowfish
- ☐ Twofish
- ☐ RC6
- ☐ Other

Response:

While designing this question I had in mind that people having knowledge of cryptography would have known dozens of crypto algorithms but how many do they use at their line of study or work. I was expecting AES to have an overwhelming majority because it is built-in in all major modems we normally use at our homes but the response was evenly distributed though majority still chose AES. This also answers why people chose block cipher over stream cipher even though stream cipher is more efficient in some cases, the reason is that the AES does not use stream cipher it works in different sizes of blocks i.e. 128, 256,192.

The list I chose for this multiple answer question comprised of AES a standard yet non feistel followed by other feistel based encryptions Blowfish and Twofish and the RC6 and finally all others that people may be using.



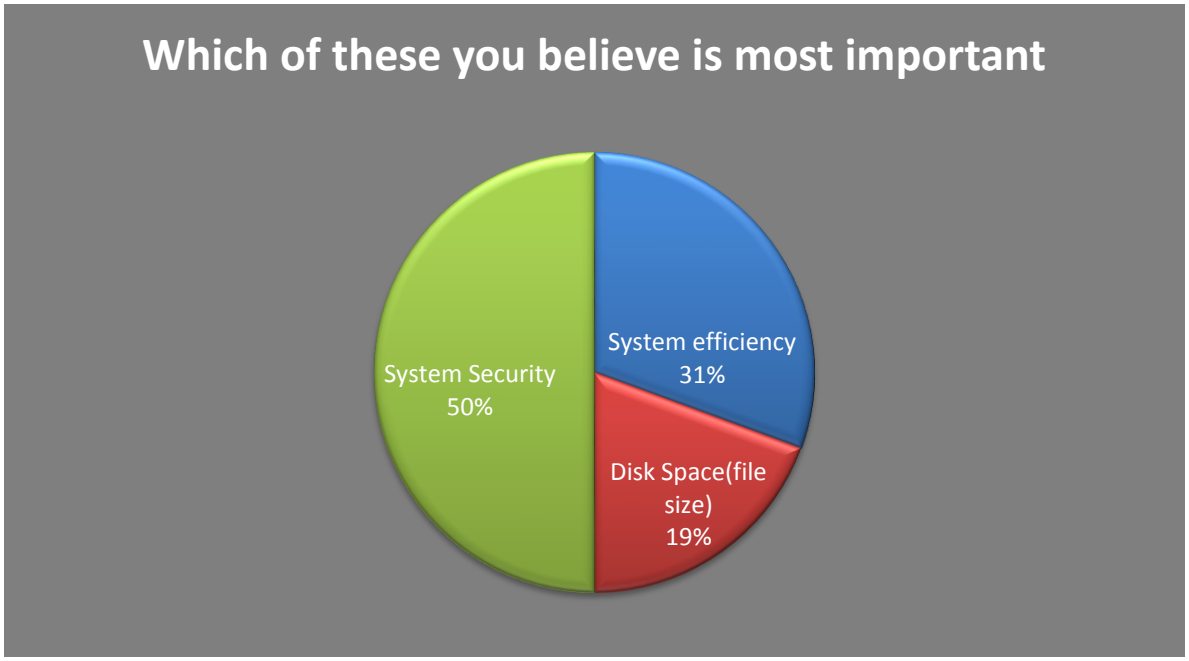
4. Which of these you believe is most Important?

- ◯ System Efficiency

- Disk Space(file size)
- System Security

Response

The response to this question eased my reservations towards the MATLOG using the encryption twice the size of the original text and people chose security to be most important and file size to be the least probably due to easy availability of terabytes of storage space and ever increasing bandwidth provided by the service providers yet we should not ignore the importance of efficiency of the algorithm as well which people still give high response even though less than security of the system.

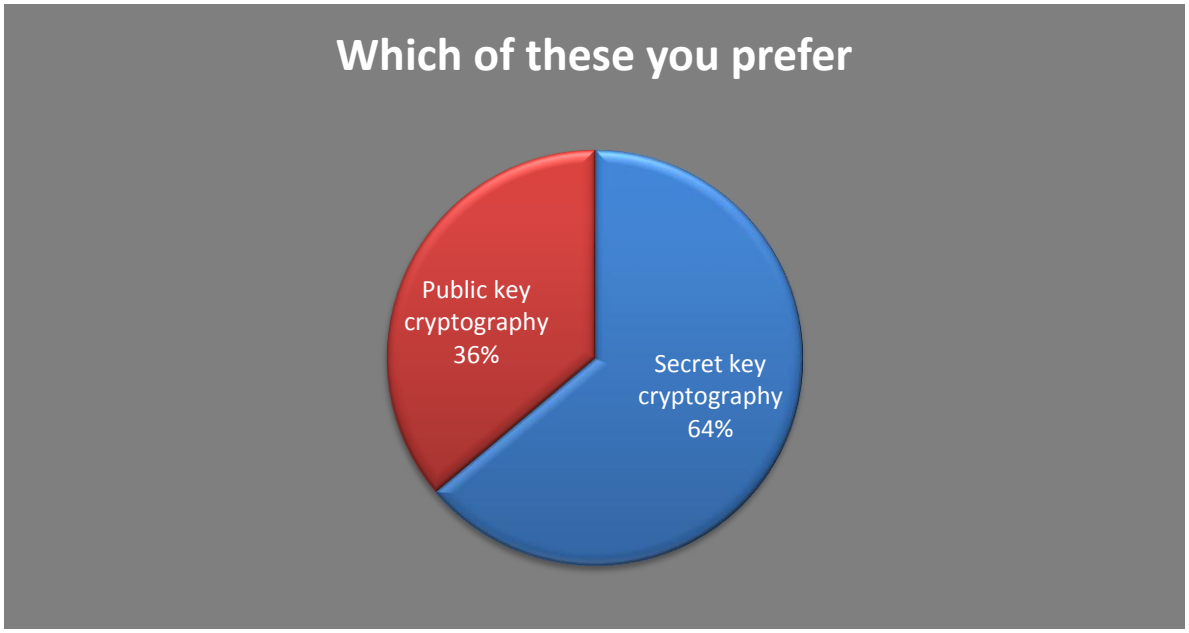


5. which of these do you prefer?

- Secret Key Cryptography
- Public Key Cryptography

Response

The last question I consider flawed, the correct approach would have been the same question followed by the reason for their choice. But on the other hand it would be safe to presume once again that AES is the popular choice therefore people went for the symmetric key cryptography in this case.



Design

The design of the algorithm encapsulates the ideas from different technologies and methodologies mentioned in the literature review. I am going to discuss the process step by step how I came up with the final algorithm.

Splitting blocks

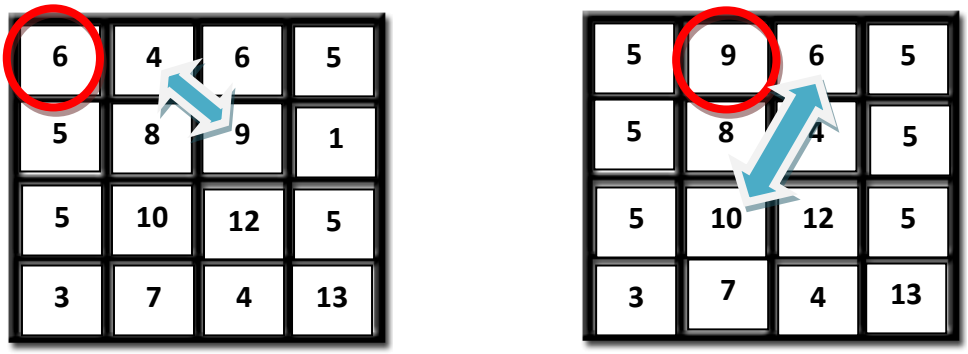
The idea of splitting blocks came from the Counter mode of operation and the One-time pads, the difference here is that unlike One-time pads or the Counter the MATLOG only produces random digits at the sender's end and are destroyed by the receiver of the encrypted data. The numbers are totally random and pads cannot be maintained for this method to determine which random numbers have been used before. This method solves the problem with One-time pads that could not be used commercially due to wide scale highly economical maintenance of the pads. The process of block splitting is as follows:

1. j = Random (0- ASCII(data[a])) // create a random number j between 0 and ASCII value of current data item
2. x[b] = ASCII (data[a]) - j //the current index of split block will store the difference between the data value and the random number "j".
3. x[b+1] = j //the following index stores the random number j
4. a = a+1
5. b=b+2
6. loop

The above process results into two complete blocks of code that contain the data of the first split into two, and the random numbers stored along with them. It must be clear by now that we have eliminated the use of initialization vector in this algorithm and instead of Feistel's L and R sub blocks we have two full blocks e1 and e2 of 128 bits or 16 bytes each.

Transposition 1

After we have the blocks e1 and e2, the problem is that they are placed next to each other and anyone could add them up and find out the actual data. The solution is to transposition the data in such a way that it is not traceable, my idea is to let the data itself decide where it is to be kept and I call the process as transposition 1; it is based on column swapping inspired by the row shifting of the AES. The figure below shows how this transposition works:

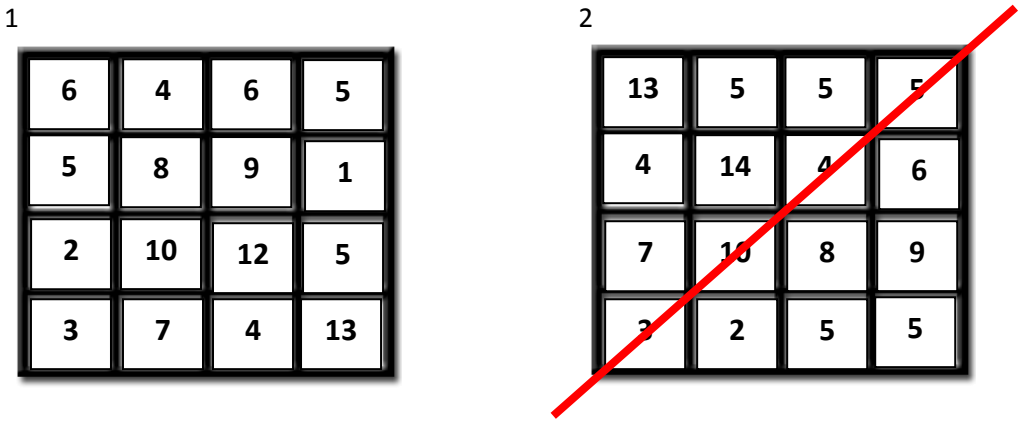


As you can see in the diagram above the data of the current index marked by a red circle determines which index from 0 to 16 should the following index be swapped with. In the above example it index 0 suggested the following index should swap with 9 and the index 1 suggests the following index should swap with index 9.

This method is highly dynamic in a sense that every time the data is different and the random e1 and e2 generated are also different so the pattern of data transposition will be ever-changing.

Folding Matrix

The transposition 1 method is followed by another method I call folding matrix where I am going to fold the 16 byte matrix along its diagonals leaving the values along the diagonals intact and swapping the rest of the numbers opposite to one another. The following diagram explains the process:



3

13	5	5	5
4	14	4	6
7	10	8	9
3	2	5	5

4

13	4	7	3
5	14	10	2
5	4	8	5
5	6	9	5

Inverse Whirl

As the name shows this method is going to transform the block in the pattern of a whirlpool going outwards. It is very similar to the clockwise routes (Wrixon, 2005) only that the inverse whirl works on smaller blocks and different pattern. The system can be explained better through the figure below:

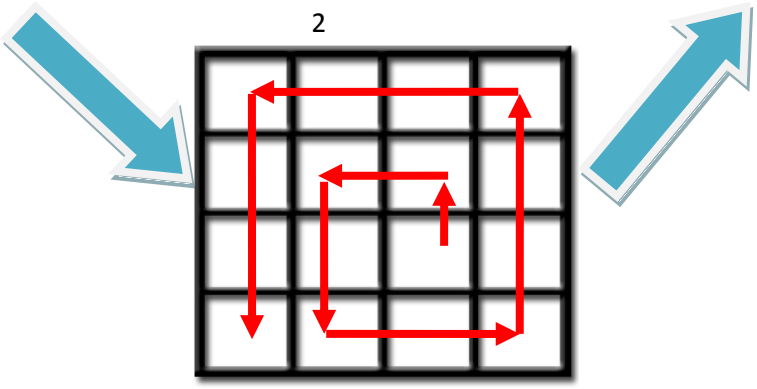
1

6	4	6	5
5	8	9	1
2	10	12	5
3	7	4	13

3

12	9	8	10
7	4	13	5
1	5	6	4
6	5	2	3

2



This method can be implemented using several other possible combination, it provides quite a riddle to those who do not know the process through which it has been transformed.

Key Management

MATLOG also like AES generates multiple keys, but the keys in this algorithm are polynomials the first one is entered by the user but the rest are generated through differentiation. So if the first key is in the form:

f(x) = ax ^ n + bx ^ (n - 1) + cx ^ (n - 2) + k

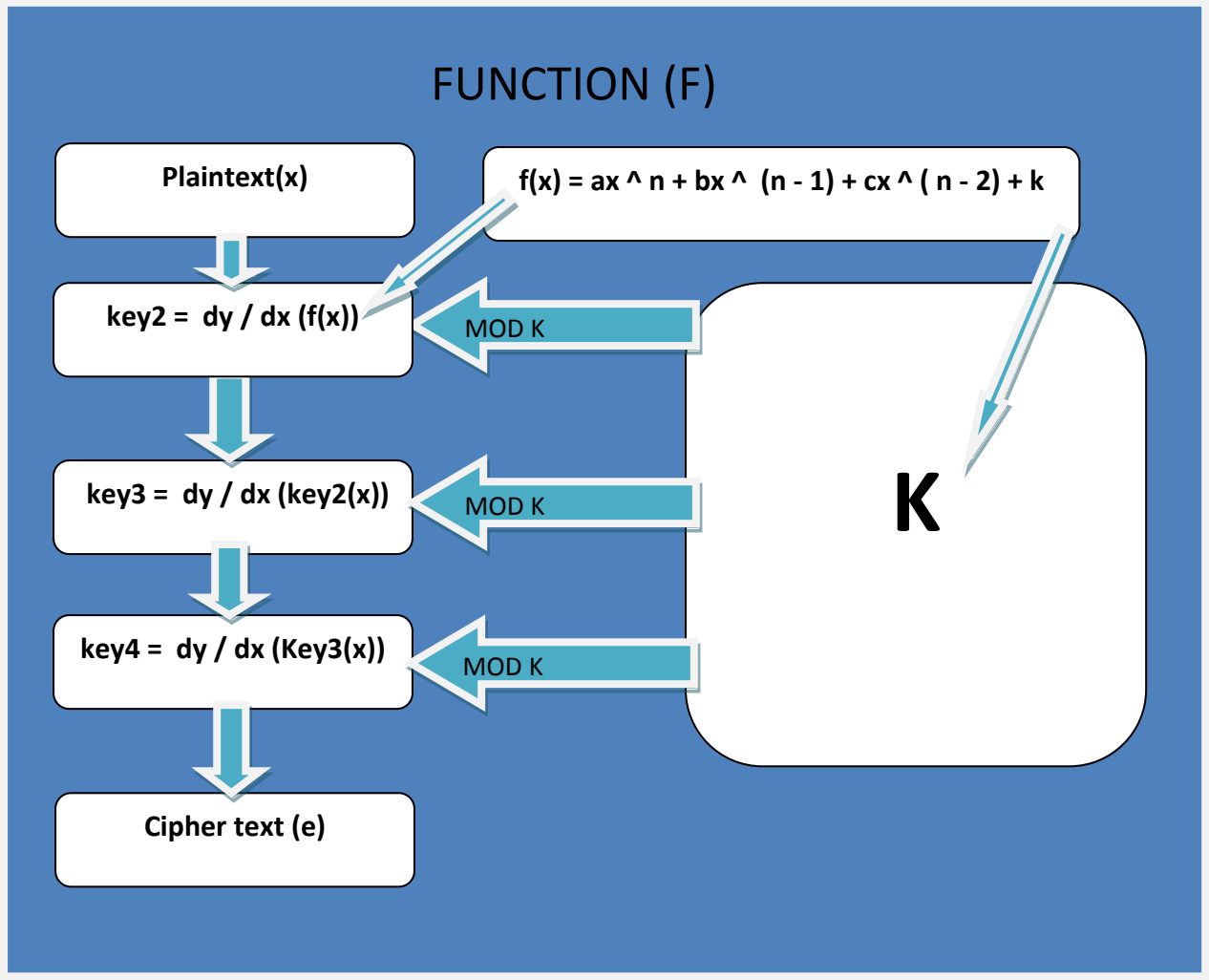
--------> Where x is the data value that has to be calculated for the encryption function followed by the next round key:

key2 = dy / dx (f(x))

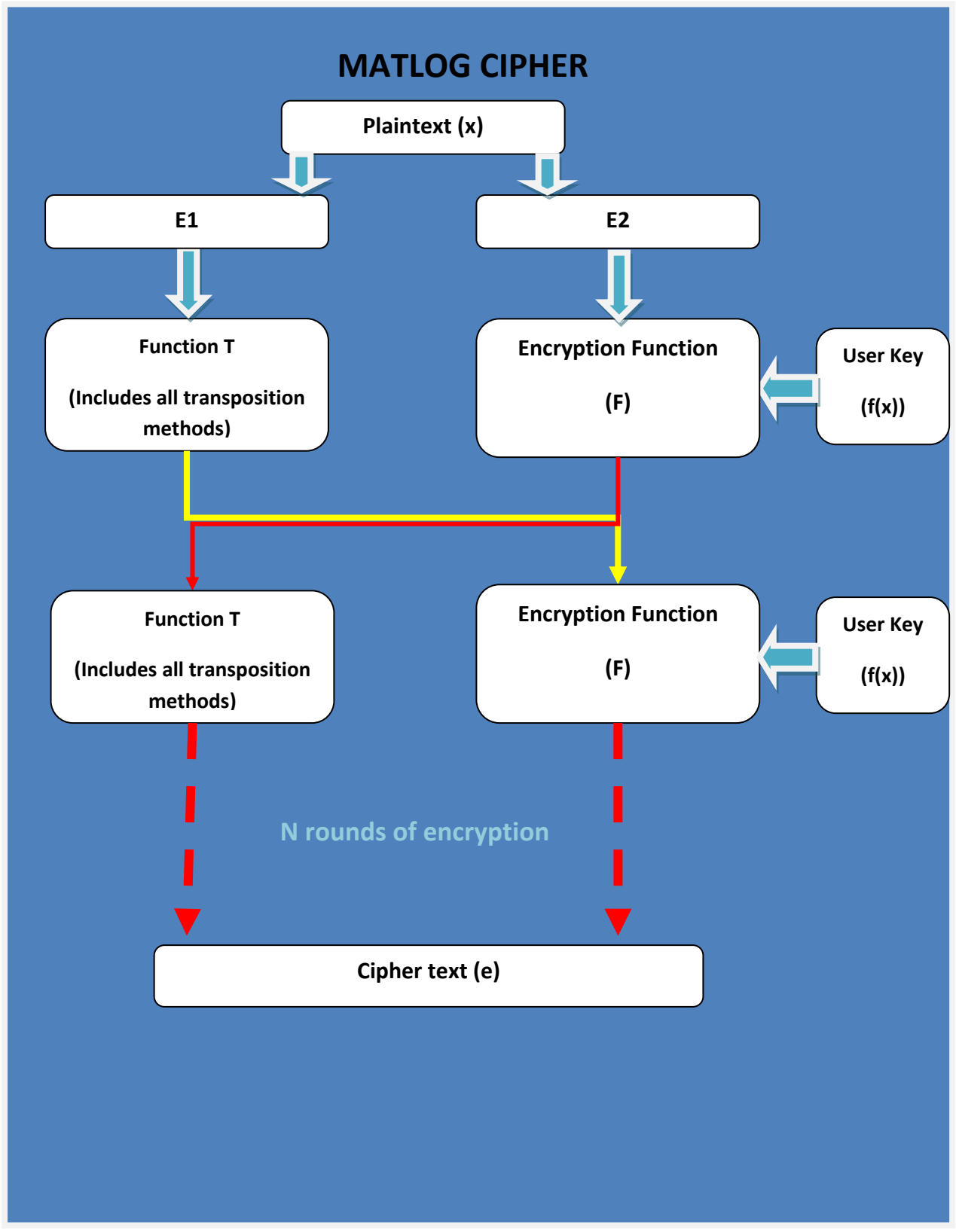
key3 = dy / dx (key2(x))

The Encryption Function (F)

The AES is both a substitution and transposition cipher, to be any close to the Advance Encryption Standard MATLOG must also incorporate as many qualities of the standard algorithm as possible. The task of function F is to perform substitution using the polynomial keys generated earlier in a manner that the data is encrypted and the encryption does not cause overflow in the space provided by the data type. The Encryption function is as follows:

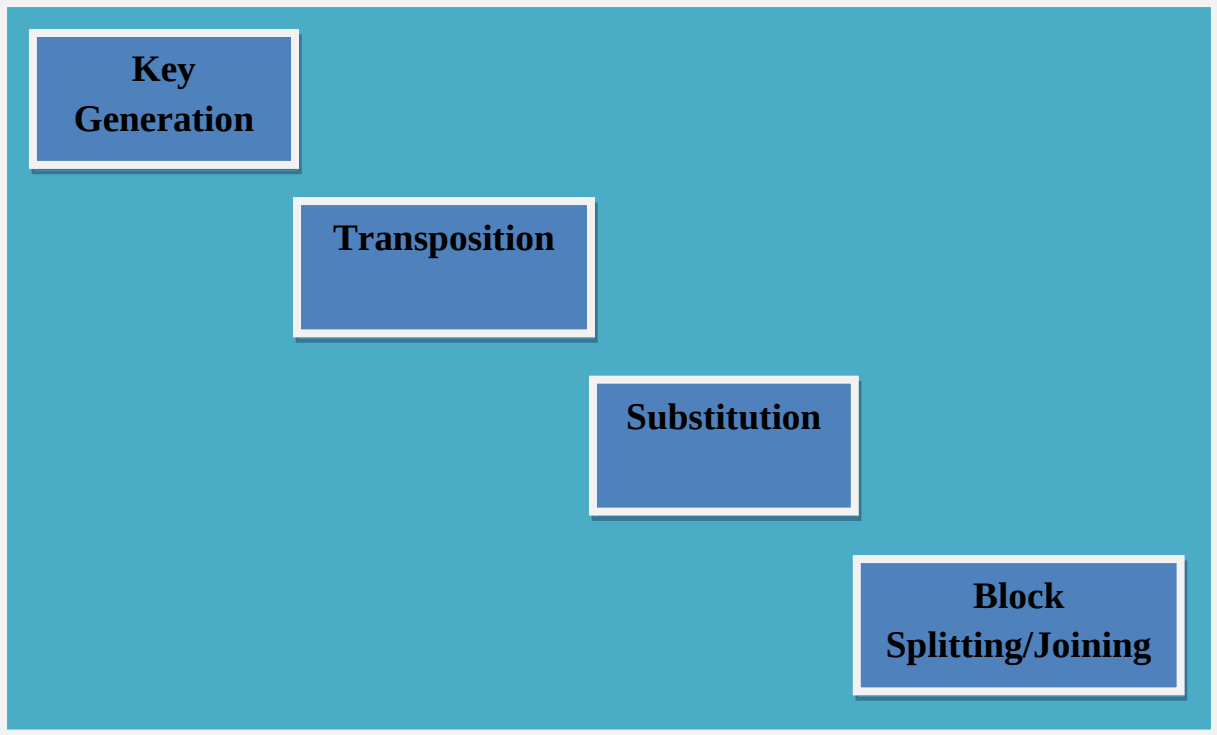


The Overall Design of MATLOG



Development & Implementation

In order to correctly and precisely study the process of encryption I have decided to implement the algorithm in modular form by dividing the algorithm into separate modules in terms of tasks they are designed to perform, the modules were implemented and tested separately before forming them into the final MATLOG Algorithm these simulations are added to the artefact. The following figure shows the modules that are implemented in the artefact:



Programming language

Microsoft Visual Basic is a purely event driven programming language and is ideal for the implementation of the modular architecture mentioned above. The idea is to program different functions or procedures for each module and call them at different events and monitor the values as they occur. I have provided the compiled MATLOG.exe file along with the project deliverables but better understanding of the functions will come only from running the source code using the visual basic IDE. The language is both compiler and interpreter based therefore one can run the application using flag points and monitor crucial variables at every point of execution.

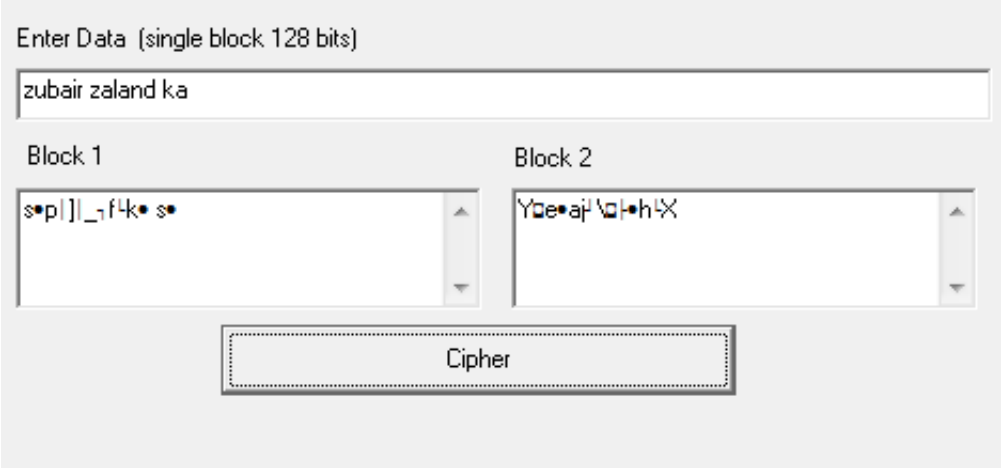
```
Public Sub functionF(data, key, counter)

x = 0
For j = 1 To counter - 1
a = counter - 1
res = 0
    While a <> -1
        res = res + key(j, a) * (data(x) ^ a) ''''''' function f(x)= ax^n+bx^n-1+cx^n-3....
        a = a - 1                 data(x) = 115
    Wend
    res = (res Xor data(x) ^ j)      ''''''''x XOR (f(x) mod x)

    data(x) = res

    Next
```

The above figure shows the data flag points in red and the yellow area shows the instruction that interpreter is going to execute next. The value "*data(x) = 115*" indicates the value of the index of the array at that point. This method is very helpful in development, error removal and testing.

I have not used any method for padding in the artefact because the Visual basic reads the left over indexes as "Empty" and does not need to store padding data the empty indices would indicate themselves to be the end of file.

Splitting blocks

As far as splitting blocks is concerned it was easily done, the visual basic generates random numbers with floating points the main task was to make sure the random number generated is refined enough to be converted to the integer form in order to subtract the ASCII code of the original data.



The Block 1 and block 2 show the initial state of sub blocks e1 and e2. The process was the same as in design only the syntax had to be transformed to that of visual basic script.

```
Public Function spreadAscii(txt As TextBox, i, j, data)
For a = 0 To 15              ''' Make sure the matrix is initially empty
    data(a) = Empty
Next
For a = 0 To Len(txt.Text) - 1              ''''Fetch every character of user's plaintext
   data(a) = Asc(Mid(txt.Text, a + 1, 1))
Next
 x = 0              y = 0
For a = 0 To Len(txt.Text) - 1
   p = Int(Rnd(data(a)) * 10)              ''''generate random number
```

```
If a <= 7 Then
    i(x) = data(a) - p              ''''fill the first block e1
    i(x + 1) = p
    x = x + 2
Else
    j(y) = data(a) - p              ''''if characters remain fill the second block e2
    j(y + 1) = p
    y = y + 2
End If
Next
```

Transposition 1

The transposition 1, as complex as it was, caused even more complications during development. The idea had 3 major flaws that i came across during simulations.

1. The idea worked perfectly in theory but practically I omitted one minor detail which was that all common ASCII codes start above 90 so it is impossible to perform transformation on the indices of 0 to 15 with the data mostly above that figure. I solved that problem by introducing a wider array of the size of ASCII spectrum the idea worked initially but that was not the only problem.
2. The second problem was that after few rounds of encryption the data tend to become greater than the ASCII spectrum and even the newer array was not enough for the method to work.
3. The final and the greatest problem was that during a single round of transposition if any index was swapped more than once it became impossible to trace and decrypt the data which is why after discussing the matter with my supervisor I discarded the method from the main MATLOG algorithm. The simulation is still present separately in the artefact.

Folding Matrix

The folding matrix was very easy to implement it works similar to the XOR or NOT gate; if you run it the same way twice it returns the original value. But as simple the implementation the solution was simple to crack as well. After a complete transformation it generated the result perfectly inverted easily understood by anyone. But I have not discarded this method because with it I developed another transformation that I have discussed later in this document.

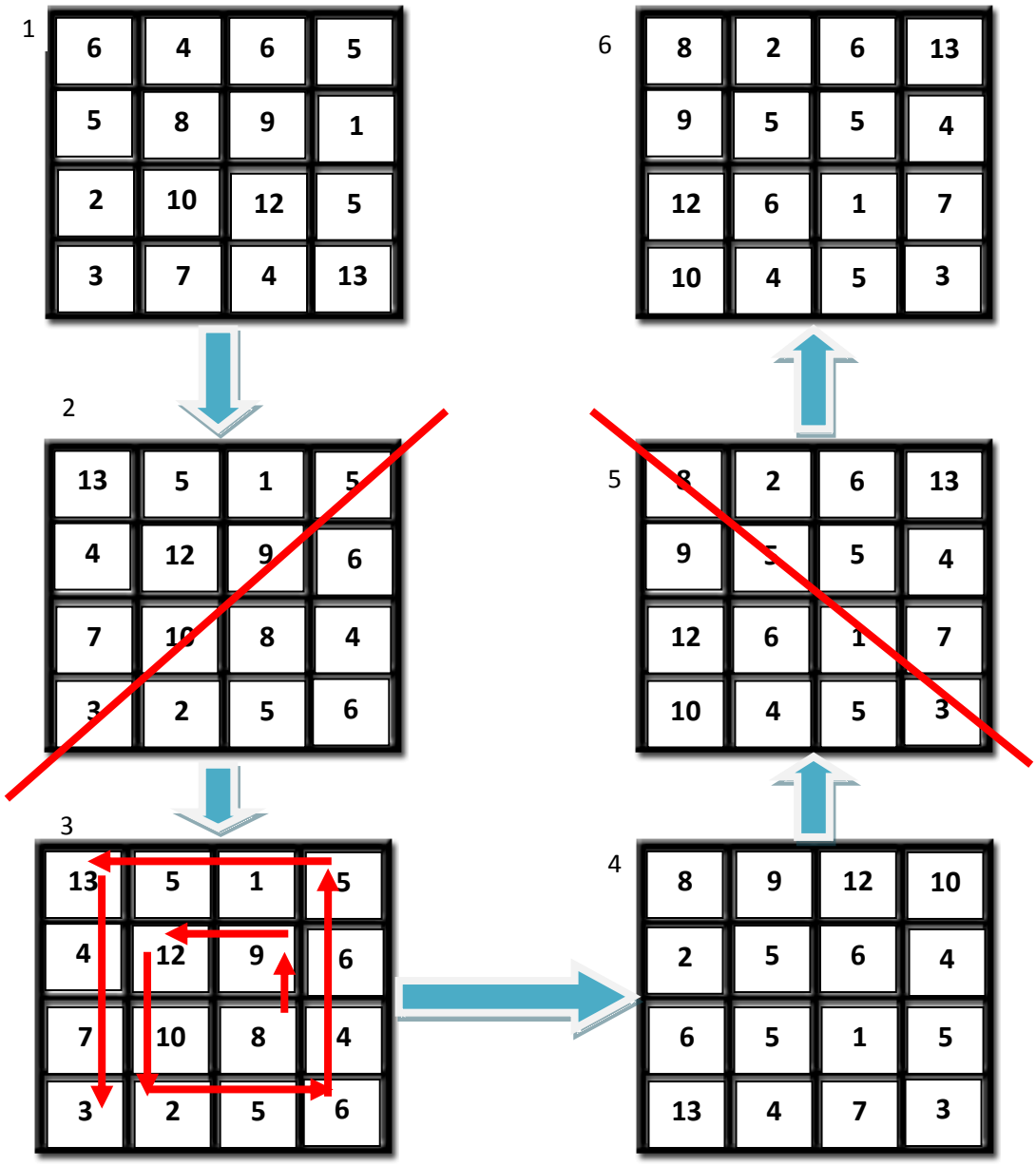
Inverse Whirl

The inverse whirl worked perfectly it was implemented using a dummy array "w[]" which stored the pattern that I have designed for the whirl. The array worked like a transformation key it worked in both transposition and decryption of the transformed state. The whirl was not as complex as Transformation 1 and not as simple as folding matrix but still nothing compared to any transposition

methods of AES on its own; therefore I devised a fourth algorithm pretty much solves the complexity issue.

Folding Inverse Whirl

This method is the combination of both folding matrix and inverse whirl, the idea is to call the inverse whirl procedure just as the folding matrix is half way through. This produces a result more complex than both of them executed separately. The figure below shows the working of the method in detail.



The figure above shows even with repeated data in the block the method has successfully shuffled it in a way that it cannot be easily understood if you are unaware of the encryption process. This method will take place of the transformation block explained in the MATLOG design.

Key Management

The key management was divided into small phases as it is the most essential part of the encryption process and key mismanagement would mean loss of encrypted data. I have implemented the key management task in following phases:

1. Declare a multi-dimensional array and store user key into it in such a way that the array stores it as a polynomial.
2. Find and separate the constant K from the user key for the encryption function F.
3. Derive all the derivatives of the user key.
4. Keep track of all the counters involved in the encryption process as other processes are dependent on counters of the key generation process.

The following table shows how the array have been used to act as a polynomial suppose the user key is = 2x^5 + 5x^3 - 4x^2 +6x + 9 . The first index is designed to hold the user key:

Key	0	1	2	3	4	5	6	7	8
0	9	6	-4	5	0	2	<--user key		
1	6	-8	15	0	10		<-- First derivative		
2	-8	30	0	30			<-- 2nd Derivative		
3									
4									
5		K = (0,0) = 9							

The key generation algorithm is as follows:

```
Public Function keygen(k)

For x = 1 To counter - 1

   counter2 = counter - 1

   Do While counter2 > 0

      key(x, counter2 - 1) = k(x - 1, counter2) * counter2

      Text3.Text = Text3.Text & key(x, counter2 - 1) & "x^" & counter2 - 1 & "+"

      counter2 = counter2 - 1

   Loop

   Display the derivative of the previous key

Next

End Function
```

The Encryption Function (F)

The encryption function uses the keys, data and the counter of the previous process as I mentioned earlier therefore when calling the function we send three arguments to the function:

```
Call enc.functionF (e1, key, counter)
```

Whereas the function itself is as follows:

```
Public Sub functionF(data, key, counter)

x = 0

For j = 1 To counter - 1

        a = counter - 1

        res = 0

        While a <> -1

                res = res + key(j, a) * (data(x) ^ a) ''''''' function f(x)= ax^n + bx^n-1 + cx^n-3....

                a = a - 1

        Wend

        res = (res Mod key(0, 0))    ''''''' (f(x) mod x)

        data(x) = res

Next

End Sub
```

Testing

Test plan:

The purpose of this test is to validate the acceptability of the algorithm and the overall project. It will confirm that the modules and features are working as they are supposed to. This plan identifies each module separately and test them in different scenarios .

Following are the objectives for MATLOG cipher test plan:

- Identification of all items and features that should be objected by the tests.
- Summarize/outline the testing approach that will be use for testing.
- Identifying the required resources for testing.
- Providing the estimations for the test efforts.
- List all the deliverables of the test activities.
- Delivers the scheduling for testing.

Items to be tested:

1. The Main MDI Form
2. Key generation and management module
3. Folding matrix

4. Inverse whirl
5. Folding Inverse whirl
6. Splitting and joining blocks
7. Crypto Function F
8. Compilation of the artefact

Test approach:

We will run the implanted code to test its results according to the requirements. In case of failure, code implementation will be revised.

Test cases:

Serial Number:	1
Plan For:	Key generation and management
Objective:	To confirm that the form itself loads and exits along with other simulation forms within the MDI.
Technique:	Using Visual basic MDI form and Menu editor.
Requirements:	None
Success Criteria:	All forms within the project are displayed
Test Data	None
Steps to be executed	Run the MATLOG application Open the key generation module from the top menu in the MDI form Open and close all the other forms available on that main form.
Expected Result	All simulation forms should open using the top menu of the main form.
Actual Result	All simulations opened and closed successfully
Pass / Fail	Pass

Serial Number:	2
Plan For:	Key generation and management
Objective:	To confirm that the keys are properly generated and kept in proper order within the array.
Technique:	Separate simulation has been designed to perform the task
Requirements:	None
Success Criteria:	All possible derivatives of the key are displayed
Test Data	2, 4, 2, 5, 5, 2
Steps to be executed	Run the MATLOG application Open the key generation module from the top menu in the MDI form Enter the coefficients of x as the test data.
Expected Result	At least five derivates of the equation displayed
Actual Result	The derivatives were successfully shown
Pass / Fail	Pass

Serial Number	3
Plan For	Folding matrix procedure
Objective	To make sure the method produces desired results
Technique	Separate simulation has been designed to perform the task
Requirements	none
Success Criteria	The data should be shuffled and rearranged.
Test Data	Zubair zaland123
Steps to be executed	Run the MATLOG Application Open the folding matrix module present on the MDI form. Enter the test data in the text field labelled "Enter Data" .
Expected Result	The data should be displayed in reverse order and back to normal
Actual Result	The test data was shown in inverted form.
Pass / Fail	Pass

Serial Number	4
Plan For	Inverse whirl procedure
Objective	To make sure the method produces desired results
Technique	Separate simulation has been designed to perform the task
Requirements	None
Success Criteria	The data should be shuffled and rearranged and back to its original form
Test Data	Zubair zaland @1
Steps to be executed	Run the MATLOG Application Open the folding matrix module present on the top menu of the MDI form. Enter the test data in the text field labelled "Enter Data" . Click Cipher
Expected Result	The data should be displayed in random order
Actual Result	The test data was shown in mixed up/ random order and successfully arranged back to normal.
Pass / Fail	Pass

Serial Number	5
Plan For	Inverse whirl procedure
Objective	To make sure the method produces desired results
Technique	Separate simulation has been designed to perform the task
Requirements	None
Success Criteria	The data should be shuffled and rearranged in even more complex manner and back to its original form
Test Data	Zubair zaland @3
Steps to be executed	Run the MATLOG Application Open the folding matrix module present on the top menu of the MDI form. Enter the test data in the text field labelled "Enter Data" .

Expected Result	The data should be displayed in random order
Actual Result	The test data was shown in mixed up/ random order and rearranged to its original format.
Pass / Fail	Pass

Serial Number	6
Plan For	Splitting and joining blocks procedure
Objective	To make sure the method produces desired results
Technique	Separate simulation Produces random number the difference of which from the data is stored along with the random number and encrypted.
Requirements	Data should not be null
Success Criteria	The data should be converted into ASCII form and replaced by some other random ASCII characters.
Test Data	Zubair zaland @3
Steps to be executed	Run the MATLOG Application Open the MATLOG form module present on the top menu of the MDI form. Enter the test data in the text field labelled "Enter Data" and click cipher.
Expected Result	The data should be replaced by two blocks of fully random ascii digits.
Actual Result	Two blocks were generated with random data twice the data size.
Pass / Fail	Pass

Serial Number	7
Plan For	Crypto function F procedure
Objective	To make sure the method produces desired results
Technique	Separate simulation Produces random number the difference of which from the data is stored along with the random number and encrypted.
Requirements	Data should not be null, key must be entered
Success Criteria	The data should be converted into ASCII form and replaced by some other random ASCII characters and then further substituted once the function is executed.
Test Data	12, 3, 3, 4, 2, Zubair zaland @3
Steps to be executed	Run the MATLOG Application Open the MATLOG form module present on the top menu of the MDI form. Enter the coefficients of x as the test data. Enter the test data in the text field labelled "Enter Data" and click cipher.
Expected Result	The data should be replaced by two blocks of fully random ASCII digits. The random numbers should then be further substituted by

	other ASCII Characters.
Actual Result	Two blocks were generated with random data twice the data size. Those blocks were finally brought together in form of one whole cipher text.
Pass / Fail	Pass

Serial Number	8
Plan For	Compilation of the entire artefact
Objective	To make sure the method produces desired results
Technique	Using visual basic compiler
Requirements	No error or exception to be thrown during the process
Success Criteria	An Executable file should be created without errors or exceptions and is capable of opening separately from Visual basic IDE
Test Data	File name (MATLOG.exe)
Steps to be executed	Open the project in visual basic Test all modules once again to make sure no errors occur Go to file menu and select "Make project exe" Enter the name of executable file in the required field.
Expected Result	The Executable file should be created in the desired location
Actual Result	An executable file was created within the project folder.
Pass / Fail	Pass

Conclusion

This project was not completed with all the desired modules that were designed in the beginning, but it is safe to say I finally reached my target and the MATLOG cipher finally worked ; with many alterations inspired by the existing technologies and some unique qualities of its own. The MATLOG incorporated many qualities of the Advance Encryption Standard into its own in a different way; as discussed earlier the MATLOG uses the multiple round keys same as the AES (Daemen, 1998) but in a different manner. Just like the AES it uses both substitution and transposition but both using its own techniques and also just like AES I used multiple transposition techniques to shuffle the data blocks. The key generation was the most tricky part and other processes were dependent on it so I had to keep them in form of coefficients in a multi-dimensional array to act as polynomials and also keep track of the other auto generated keys. Until I had the idea of mixing up the two transpositions (folding matrix, Inverse whirl) I thought the project had failed because both these transpositions separately are nothing compared to the AES transposition.

The ideas of all the methods and modules all came to me in the sequence of the design part of this document some were the result of my studies at the university, some came from reading the relevant material and others were purely ideas out of nowhere and I looked them up and found similar work done before which would automatically be helpful in perfecting the flaws in my ideas I would acknowledge Sarah et al (Diesburg, 2008), their work has been highly inspirational and helpful. One can notice the MATLOG also works on the same principal of All Or Nothing

transformation (Boyko, 1999, Rivest, 1997) as they mentioned in their work i.e. if we lose a single byte out of the equation we cannot get neither the transposed nor the substituted data back.

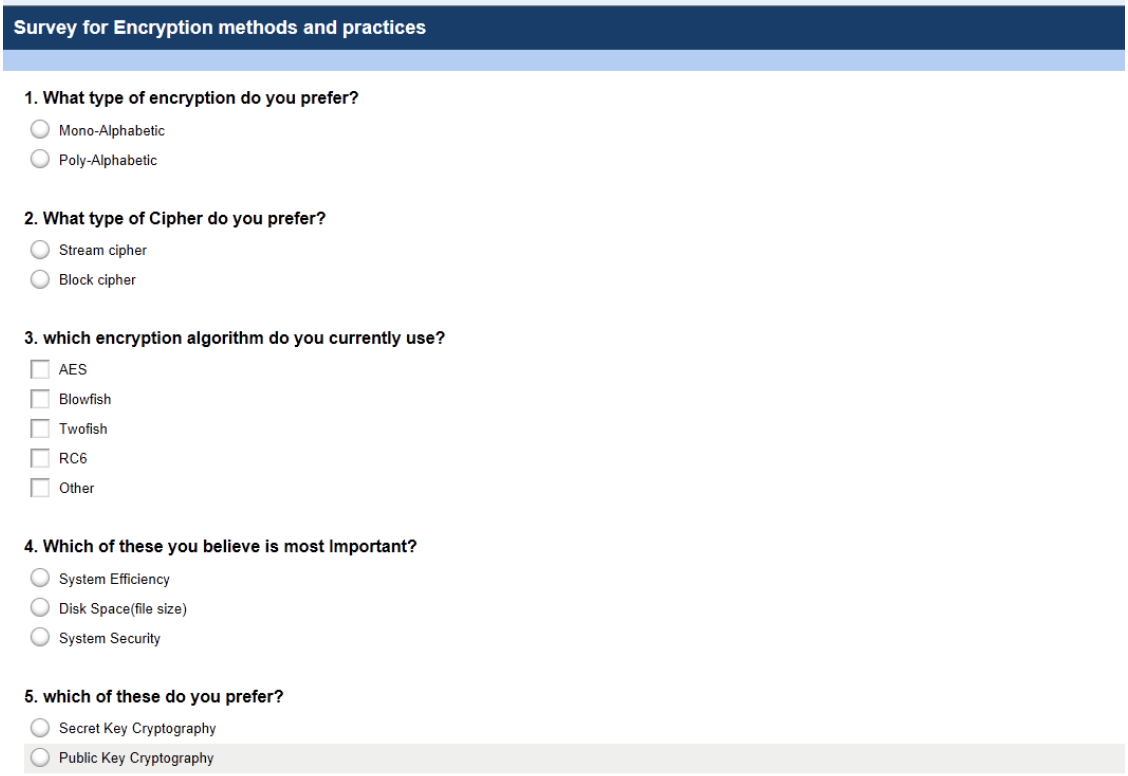
I regarded the transposition 1 to be the most complex and the jewel of this encryption because of the idea of data itself performing its transposition. Unfortunately for me, I could not make it work in time there were a lot of flaws in the basic idea of the procedure and given more time and study I would come up with a perfect solution for this kind of transposition. Another regret for me is that along with other qualities of AES embedded into this project I could not find a way to design my own bit level transformation like that of the AES which makes it also part of my future to do list and I am confident that it can be done with proper study. The good thing about this project is that apart from many failures that occurred during the course of this project we finally have the blue prints of the world's first Feistel based poly-alphabetic cipher and I am looking forward to making more improvements to the idea and make the upcoming MATLOG versions safer, Efficient and closer to the well known encryption algorithms.

References:

- Sarah M. Diesburg, Christopher R. Meyers, David M. Lary, An-I Andy Wang, (2008), *When Cryptography Meets Storage*, ACM.
- Brian Baldwin, Emanuel M. Popovici, Michael Tunstall, and William P. Marnane, (2008), *Fault Injection Platform for Block Ciphers*, ISSC.
- William F. Ehrsam, Carl H. W. Meyer, John L. Smith, Walter L. Tuchman. (1976), *Message verification and transmission error detection by block chaining*. US Patent 4074066.
- Yuliang Zheng, Yongge Wang, (2005), *Efficient and Provably Secure Ciphers for Storage Device Block Level Encryption*. SIS Department, UNC Charlotte, ACM.
- Hui QIN, Tsutomu SASAO, Yukihiro IGUCHI. (2005), *An FPGA Design of AES Encryption Circuit with 128-bit Keys.* Dept.of Computer Science and Electronics, Kyushu Institute of Technology, Iizuka, 8208502, Japan, ACM.
- PHILLIP ROGAWAY, MIHIR BELLARE, JOHN J. BLACK. (2003), *OCB: A Block-Cipher Mode of Operation for Efficient Authenticated Encryption*, ACM.
- JES ´ US ADOLFO GARC´IA-PASQUEL and JOS´E GALAVIZ, (2006), Ganz´ua: A Cryptanalysis Tool for Monoalphabetic and Polyalphabetic Ciphers, ACM.
- Douglas R. Stinson,(2006), *Cryptography: Theory and Practice*, 3rd Ed, Chapman & Hall/CRC.
- Abidalrahman.Mohd, Yaser Jararweh,(2011) *AES-512: 512-Bit Advanced Encryption Standard Algorithm Design and Evaluation*,IEEE.
- j. Daemen and V. Rijmen, (1998), The Rijndael Block Cipher:AES Proposal, Proc. 1st AES Candidate Conference.
- Boyko V., (1999), On the security properties of OAEP as an all-ornothing transform. In *Advances in Cryptology - Crypto'99 Proceedings,* Springer-Verlag, pp. 503–518.
- Bruce Schneier, Applied Cryptography, 2nd Ed, John Wiley & Sons Inc.
- D. Kahn, (1967), The Codebreakers: The Story of Secret writing, New York; Macmillan Publishing Co.
- Rivest RL. All-or-nothing encryption and the package transform. *Proceedings of the Fast Software Encryption Conference*, 1997.
- Fred B. Wrixon,(2005), CODES, Ciphers, Secrets and Cryptic Communication, 1st Ed., Black Dog and Leventhal Publishers Inc.

Appendix

The survey questionnaire:



Survey for Encryption methods and practices

1. What type of encryption do you prefer?

- Mono-Alphabetic
- Poly-Alphabetic

2. What type of Cipher do you prefer?

- Stream cipher
- Block cipher

3. which encryption algorithm do you currently use?

- AES
- Blowfish
- Twofish
- RC6
- Other

4. Which of these you believe is most important?

- System Efficiency
- Disk Space(file size)
- System Security

5. which of these do you prefer?

- Secret Key Cryptography
- Public Key Cryptography

The following link leads to the online survey:

http://www.surveymonkey.com/s/6GR577Q

The source code for modules

```
Dim w(0 To 15)

Public Sub functionF(data, key, counter)

For x = 0 To 15

For j = 1 To counter - 1

a = counter - 1

res = 0

  While a <> -1

    res = res + key(j, a) * (data(x) ^ a) ''''''' function f(x)= ax^n+bx^n-1+cx^n-3....

    a = a - 1

  Wend

  res = (res Mod key(0, 0))   ''''''' (f(x) mod x)
```

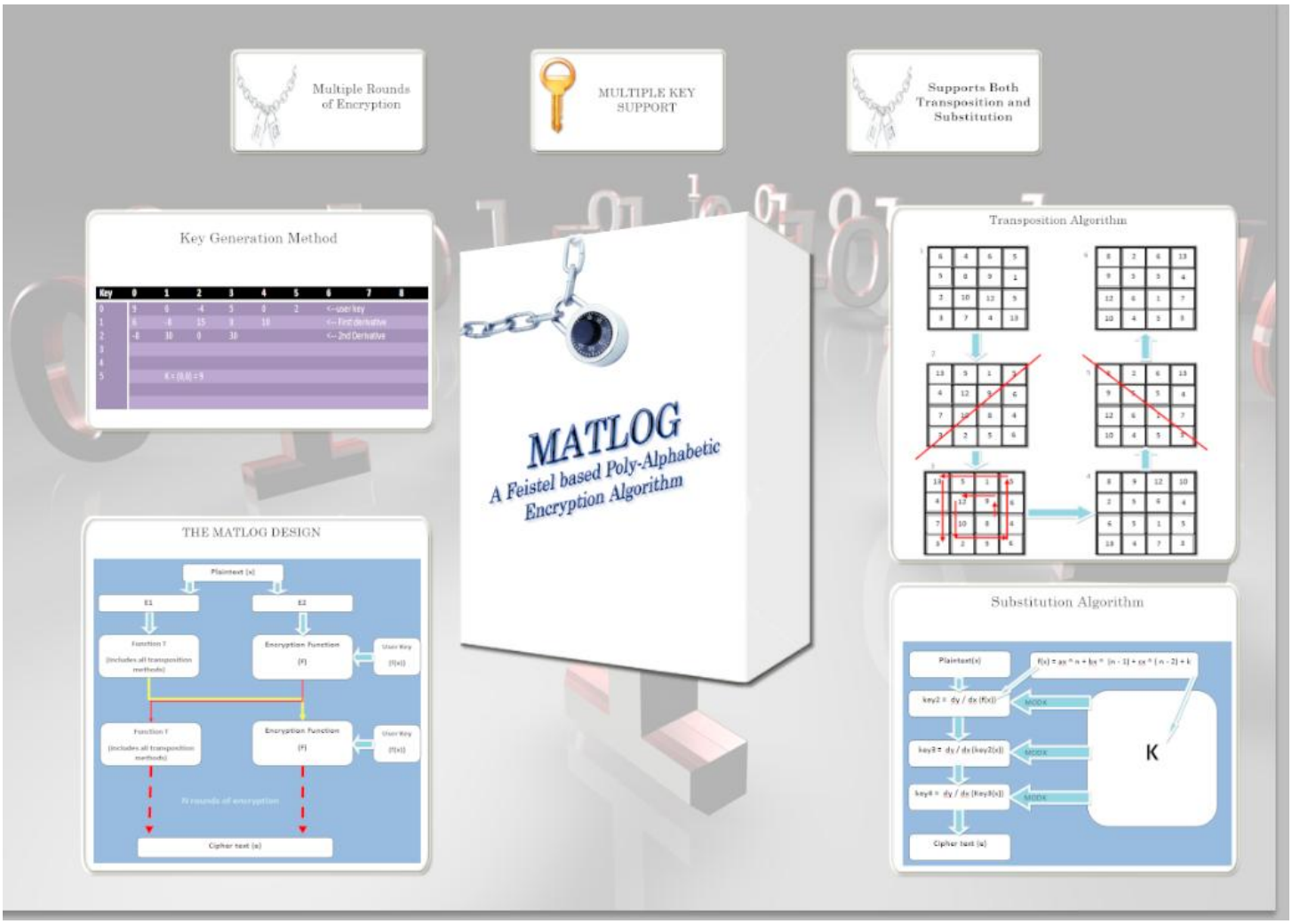
```
   data(x) = res
      Next
Next
End Sub
Public Sub swap(a, b)
Dim c
   c = a
   a = b
   b = c
End Sub
Public Function keygen(k)
Dim x
For x = 1 To counter - 1
   counter2 = counter - 1
   Do While counter2 > 0
      key(x, counter2 - 1) = k(x - 1, counter2) * counter2
      Text3.Text = Text3.Text & key(x, counter2 - 1) & "x^" & counter2 - 1 & "+"
      counter2 = counter2 - 1
   Loop
   Text3.Text = Text3.Text & " ..... "
Next
End Function
Public Function spreadAscii(txt As TextBox, i, j, data)
Dim p
For a = 0 To 15
    data(a) = Empty
```

```
Next
For a = 0 To Len(txt.Text) - 1
   data(a) = Asc(Mid(txt.Text, a + 1, 1))
Next
 x = 0
 y = 0
For a = 0 To Len(txt.Text) - 1
   p = Int(Rnd(data(a)) * 10)
   If a <= 7 Then
      i(x) = data(a) - p
      i(x + 1) = p
      x = x + 2
   Else
      j(y) = data(a) - p
      j(y + 1) = p
      y = y + 2
   End If
Next
End Function
Public Function contractAscii(i, j, k)
x = 0
y = 0
While a <> 15
   If a <= 7 Then
      k(a) = i(x) + i(x + 1)
      x = x + 2
   Else
```

```
    k(a) = j(y) + j(y + 1)
    y = y + 2
  End If
  a = a + 1
Wend
End Function
Public Function whirl(d, e)
Dim temp(0 To 15)
For a = 0 To 15
temp(a) = e(a)
Next
For a = 0 To 15
  d(w(a)) = temp(a)
Next
End Function
Public Function foldWhirl(data)
For a = 0 To 15
  If (a = 7) Then
  Call whirl(data, data)
  End If
    Call swap(data(a), data(15 - a))
Next
End Function
Public Function foldinvWhirl(data)
For a = 0 To 15
  If (a = 7) Then
     Call Invwhirl(data, data)
```

```
    End If

        Call swap(data(a), data(15 - a))

Next

 foldinvWhirl = data

End Function

Public Function Invwhirl(e, data)

Dim temp(0 To 15)

w(0) = 10       w(1) = 9        w(2) = 5        w(3) = 6        w(4) = 7        w(5) = 11

w(6) = 15       w(7) = 14       w(8) = 13       w(9) = 12       w(10) = 8       w(11) = 4

w(12) = 0       w(13) = 1       w(14) = 2       w(15) = 3

For a = 0 To 15

temp(a) = data(a)

Next

For a = 0 To 15

    e(a) = temp(w(a))

Next

End Function
```

Project Poster



Market Survey Returns

PAGE: 1

1. What type of encryption do you prefer?	Create Chart	Download
	Response Percent	Response Count
Mono-Alphabetic	41.7%	15
Poly-Alphabetic	**58.3%**	**21**
	answered question	**36**
	skipped question	**0**

2. What type of Cipher do you prefer?	Create Chart	Download
	Response Percent	Response Count
Stream cipher	47.2%	17
Block cipher	**52.8%**	**19**
	answered question	**36**
	skipped question	**0**

3. which encryption algorithm do you currently use?	Create Chart	Download
	Response Percent	Response Count
AES	**52.8%**	**19**
Blowfish	27.8%	10
Twofish	36.1%	13
RC6	36.1%	13
Other	41.7%	15
	answered question	**36**
	skipped question	**0**

4. Which of these you believe is most Important?	Create Chart	Download
	Response Percent	Response Count
System Efficiency	30.6%	11
Disk Space(file size)	19.4%	7
System Security	**50.0%**	**18**
	answered question	**36**
	skipped question	**0**

5. which of these do you prefer?	Create Chart	Download
	Response Percent	Response Count
Secret Key Cryptography	**63.9%**	**23**
Public Key Cryptography	36.1%	13
	answered question	**36**
	skipped question	**0**