www.ijemr.net ISSN (ONLINE): 2250-0758, ISSN (PRINT): 2394-6962

Volume-7, Issue-6, November-December 2017

International Journal of Engineering and Management Research

Page Number: 76-82

A Comparative Study of Some Traditional and Modern Cryptographic Techniques

Oguntunde, B.O.[1], Arekete, S.A.[2], Odim, M.O.[3] and Olakanmi, O.I.[4]
Redeemer's University Ede, NIGERIA

ABSTRACT

In the era of Internet and networks applications with the attendant prevalence of virus attacks, and intrusion of various kinds and intensities, information security has become a major challenge. There is a demand for stronger encryption techniques which are very hard to crack. The role of cryptography in the field of network security has become very pivotal. There are a wide range of cryptographic algorithms that are used for securing networks. There are also continuous research efforts to formulate new cryptographic algorithms aimed at evolving more advanced techniques for more secured communication. This work analyses some cryptographic techniques based on programming approaches and performances using certain criteria such as the block size, key length, and encryption time and security issues. Blowfish technique was found to offer a better performance compared to AES and RSA in terms of average encryption time. However, DES and Blowfish share the smallest block size of 64 bit while DES has the least key length of 56 bits.

Keywords-- Encryption, cryptography, algorithm, block size, key length

I. INTRODUCTION

One important method of data and information security is encryption which initiated a major field of Computer Science called Cryptography. Cryptography deals with the practice and study of techniques for secure communication in the presence of third parties known as adversaries [1],[2]. More generally, cryptography is about constructing and analyzing protocols that prevent third parties or the public from accessing private messages. Various aspects of information security such as data confidentiality, data integrity, authentication, and non-repudiation are fundamental to modern cryptography. Cryptography has been applied to many areas of life, these include but not limited to automated teller machine (ATM) operations, computer passwords, and electronic commerce. Modern cryptography relies on some publicly known mathematical algorithms for coding the information. Secrecy is obtained through a secret key which is used as the seed for the algorithms. The computational difficulty of algorithms, absence of secret key, etc., make it impossible for an attacker to obtain the original information even if he knows the algorithm used for coding.

The earliest form of cryptography was concerned with the confidentiality of messages through encryption which is a specific element of cryptography that hides data or information by transforming it into an undecipherable code [3], [4]. The original message (plain text) is transformed via encryption to produce cipher text, this process renders the message unreadable by interceptors or eavesdroppers who do not have the right key to decrypt the message to get the original message.

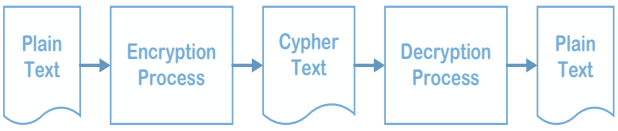

Figure 1 Encryption and Decryption of Plain Text

Encryption enforces secrecy in communications, and are widely used by spies, military leaders, and diplomats. According to [5], the field has recently expanded beyond confidentiality concerns to include techniques for message integrity checking, sender or receiver identity authentication, digital signatures, interactive proofs and secure computation, among others. The main cipher types in the early stage of cryptography include:

i. Transposition ciphers: which rearrange the order of letters in a message, e.g. "happy world" becomes "phayp owrdl" in a trivially simple rearrangement scheme.

ii. Substitution ciphers: which systematically replace letters or groups of letters with other letters or groups of letters E.g. "sleep at night" becomes "tmffq bu ojhiu" by replacing each letter with the one following it in the Latin alphabet.

Simple versions of either have never offered much confidentiality from enterprising opponents.

The rest of the paper is organised as follows: Section II presents a literature review of related research works in the area. In section III, a comparative analysis of selected cryptographic techniques were undertaken

while in section IV the performance evaluation of some cryptographic techniques is presented. Section V presents the conclusion.

II. LITERATURE REVIEW

1. Cryptanalysis

A new field called cryptanalysis has emerged, which involves analyzing information systems in order to study the hidden aspects of the systems. "Cryptanalysis is the science of analyzing and breaking encryption schemes", [6] The goal of cryptanalysis is to find some weakness or insecurity in a cryptographic scheme, thus permitting its subversion or evasion. It means "breaking the code", and it relies on the knowledge of the encryption algorithm (that for civilian applications should be in the public domain) and some knowledge of the possible structure of the plaintext (such as the structure of a typical inter-bank financial transaction) for a partial or full reconstruction of the plaintext from ciphertext. Additionally, the goal is to also infer the key for decryption of future messages [7]. There is a wide range of cryptanalytic attack techniques, and they can be classified in several ways:

Cipher-text only attack where the attacker has access to the cipher text and then tries to analyze the cipher in order to break it. Known-plain text attack in which the attacker has access to a cipher-text and its corresponding plain text. Chosen-plain text attack where the attacker chooses a plain text and learns its corresponding cipher text i.e the attacker will learn a number of plain text and the corresponding cipher text so as to easily recognize the plain text cipher at any given point. Man-in-the-middle attack in which the attacker sits in between the sender and the recipient accesses and modifies the traffic and then forwards it to the recipient.

All forms of cryptanalysis for classical encryption exploit the fact that some aspect of the structure of plaintext may survive in the ciphertext. Cryptanalysis is also used during the design of the new cryptographic techniques to test their security strengths [8]. This work attempts to analyze and compare some cryptography technique algorithms using metrics such as encryption type, block size, key length, and to develop a cross platform cryptography library with functions that strengthens flawed implementation of some cryptography techniques for application development.

2. Cryptography Techniques

This section describes cryptographic techniques under two categories: the legacy and modern techniques.

2.1 Legacy Techniques

The techniques discussed under this category include: Caesar cipher, Vegenere cipher and ROT13.

i. Caesar Cipher: is one of the simplest and most widely known encryption techniques. It is a type of substitution cipher in which each letter in the plaintext is replaced by a letter some fixed number of positions down the alphabet [9]. The method is named after Julius Caesar, who used it in his private correspondence. In a Caesar cipher, each letter of the alphabet is shifted along some number of places; for example, in a Caesar cipher of shift 3, A would become D, B would become E, Y would become B and so on.

ii. Vigenere Cipher: The Vigenere square or Vigenere table, also known as the tabula recta, consists of several Caesar ciphers in sequence with different shift values. To encrypt, a table of alphabets can be used, termed a tabula recta, Vigenere square, or Vigenere table. It consists of the alphabet written out 26 times in different rows, each alphabet shifted cyclically to the left compared to the previous alphabet, corresponding to the 26 possible Caesar ciphers. At different points in the encryption process, the cipher uses a different alphabet from one of the rows. The alphabet used at each point depends on a repeating keyword; it can be used for both encryption and decryption.

iii. ROT13: This is a simple letter substitution cipher that replaces a letter with the 13th letter after it in the alphabet. ROT13 is a special case of the Caesar cipher, developed in ancient Rome. Because there are 26 letters (2×13) in the basic Latin alphabet, ROT13 is its own inverse; that is, to undo ROT13, the same algorithm is applied, so the same action can be used for encoding and decoding. The algorithm provides virtually no cryptographic security, and is often cited as a canonical example of weak encryption (Christopher, 2008). ROT13 is used in online forums as a means of hiding spoilers, punch lines, puzzle solutions, and offensive materials from the casual glance. ROT13 has been described as the "Usenet equivalent of a magazine printing the answer to a quiz upside down". ROT13 has inspired a variety of letter and word games on-line, and is frequently mentioned in newsgroup conversations [10].

3. Modern Techniques

This category is further sub-divided into two: the secret key and public key cryptography.

i. Secret Key: Secret key cryptography techniques that are used in use today include:

a. Data Encryption Standard (DES): DES is a block-cipher employing a 56-bit key that operates on a 64-bit blocks. DES has a complex set of rules and transformations that were designed specifically to yield fast hardware implementations and slow software implementations, although this latter point is becoming less significant today since the speed of computer processors is several orders of magnitude faster today than twenty years ago. IBM also proposed a 112-bit key for DES which was rejected at the time by the government; the use of 2-bit keys was considered in the 1900s, however, conversion was never seriously considered [11].

b. Advanced Encryption Standard (AES): The algorithm can use a variable block length and key length; the latest specification allows any combination of keys length of 128, 192, or 256 bits and blocks of length 128, 192, or 256 bits.(Joan et al, 2002).

c. Blowfish: A symmetric 64-bit block cipher. It is faster than a DES and key lengths can vary from 32 to 448 bits in length [12].

d. Twofish: A 128-bit block cipher using 128-, 192-, or 256-bit keys. It was designed to be highly secure and highly flexible, well-suited for large microprocessors, 8-bit smart card microprocessors, and dedicated hardware [13].

ii. Public-Key Cryptography (PKC): Public-key cryptography (PKC) has been said to be the most significant new development incryptography in the last 300 to 400 years. Modern PKC was first described publicly by Stanford University Professor Martin Hellman and his graduate student, Whitfield Diffie in 1976. Their work described a two-key crypto system in which two parties could engage in a secure communication over a non-secure communications channel without having to share a secret key. Public-key cryptography algorithms that are in use today for key exchange or digital signatures include:

a. RSA: This is the first and commonest PKC implementation, named after the three MIT mathematicians who developed it – Ronald **R**ivest, Adi **S**hamir, and Leonard **A**dleman. RSA can be used for key exchange, digital signatures, or encryption of small blocks of data. RSA uses a variable size encryption block and a variable size key. The key-pair is derived from a very large number, **n**, that is the product of two prime numbers chosen according to special rules; these prime numbers may be 100 or more digits in length each, yielding an **n** with roughly twice as many digits as the prime factors. The public-key information includes **n** and a derivative of one of the factors of **n**; an attacker cannot determine the prime factors of **n** (and, therefore, the private key) from this information alone and that is what makes the RSA algorithm so secured (Pfleeger, 2007). Vollala *et al*. [14] opine that the performance of RSA strongly on the implementation of modular multiplication and modular exponentiation and performance can be improved in three ways, first, reducing the frequency of modular multiplication, secondly reducing the time required for to evaluate modular multiplication and thirdly, by increasing the RSA cores.

b. Diffie Hellman (D-H): After the RSA algorithm was published, Diffie and Hellman came up with another algorithm. D-H is used for authentication or digital signatures.

c. Digital Signature Algorithm (DSA): The algorithm specified in NIST's Digital Signature Standard (DSS), provides digital signature capability for the authentication of messages.

4. Hash Functions

Hash functions, also called message digests and one-way encryption, are algorithms that, in some sense, use no key, instead, a fixed-length hash value is computed based upon the plaintext to be recovered. Hash functions, then, provide a measure of the integrity of a file. Hash algorithms that are in common use today include:

i. Message Digest (MD) algorithms are a series of byte-oriented algorithms that produce a 128-bit hash value from an arbitrary-length message.

ii. MD2**:** Designed for systems with limited memory, such as smart cards (RFC 1319).

iii. MD4**:** Developed by Rivest, is similar to MD2 but designed specifically for fast processing in software (RFC 1320).

iv. MD5**:** Also developed by Rivest after potential weaknesses were reported in MD4; this scheme is similar to MD4 but is slower because more manipulation is made to the original data. MD5 has been implemented in a large number of products although several weaknesses in the algorithm were demonstrated by German cryptographer Hans Dobbertin in 1996 (RFC 1321).

5. Secure Hash Algorithm (SHA)

i. Algorithm for NIST's Secure Hash Standard (SHS). SHA-1 produces a 160-bit hash value and was originally published as FIPS 180-1 and RFC 3174. FIPS 180- 2 (aka SHA-2) describes five algorithms in the SHS: SHA-1 plus SHA-224, SHA-256, SHA-384, and SHA-512 bits in length, respectively. SHA-224, -256, -384, and -512 are also described in RFC 4634.

ii. Whirlpool**:** A relatively new hash function, designed by V. Rijmen and P.S.L.M. Barreto. Whirlpool operates on messages less than 2256 bits in length, and produces a message digest of 512 bits. The design of this function is very different from than that of MD5 and SHA-1, making it immune to the same attacks as on those hashes.

6. Trust Models

Secure use of cryptography requires trust. While secret key cryptography can ensure message confidentiality and hash codes can ensure integrity, none of these works without trust. There are a number of trust models employed by various cryptographic schemes.

i. PGP Web of Trust**:** Pretty Good Privacy is a widely used private email scheme based on public-key methods. A PGP user maintains a local key ring of their known and trusted public keys. The users make their determination about the trustworthiness of a key using what is called a "web of trust." PGP makes no statement and has no protocol about how one user determines whether they trust another user or not.

ii. Kerberos**:** Kerberos is a commonly used authentication scheme on the internet. Developed by MIT's project Athena, Kerberos is named for the three-headed dog who, according to Greek mythology, guards the entrance of Hades (rather than the exit, for some reason). Kerberos employs a client/server architecture and provides user-to-server authentication rather than host-to-host authentication. In this model, security and authentication will

be based on secret key technology where every host on the network has its own secret key. It would clearly be unmanageable if every host had to know the keys of all other hosts so a secure, trusted host somewhere on the network, known as a Key Distribution Centre (KDC), knows the keys for all of the hosts(or at least some of the hosts within a portion of the network, called a realm). In this way, when a new node is brought online, only the KDC and the new node need to be configured with the node's key; keys can be distributed physically or by some other secure means.

iii. Public Key Certificates and Certificate Authorities**:** Certificates and Certificate Authorities (CA)are necessary for widespread use of cryptography for e-commerce applications. How, for example, does one site obtain another party's public key? How does a recipient determine if a public key really belongs to the sender? How does the recipient know that the sender is using their public key for a legitimate purpose for which they are authorized? When does a public key expire? How can a key be reversed in case of compromise or loss? The basic concept of a certificate is one that is familiar. A driver's license, credit card, or SCUBA certification, for example, identify us to others, indicate something that we are authorized to do, have an expiration date, and identify the authority that granted the certificate.

III. COMPARATIVE ANALYSIS OF CRYPTOGRAPHIC TECHNIQUES

This section presents the comparative analysis of selected cryptographic techniques, these include: Caesar cipher, Vigenere cipher, Blowfish Twofish, Message Digest 5 (MD5) and Whirlpool. These algorithms span across the three categories of cryptography techniques discussed earier. Compared algorithms are Blowfish, DES, RSA, AES, MD5, SHA-1 and Diffie Hellman. Comparison metrics are Block Size,Key Length, Encryption Time and Security

1. Caesar's Cipher

The mathematical models of Caesar's cipher is given in Equation 1.

$$Encryption: e(x) = (x + k)(mod\ 26)$$
$$Decryption: e(x) = (x - k)(mod\ 26) \quad \ldots\ldots\ldots\ldots\ldots (1)$$

where *k* is the key applied to each letter.

After applying this function the result is a number which must then be translated back into a letter. A code for Caesar's Cipher is shown in Figure 2. This was implemented into an encrypt function that takes the plaintext and the key n as parameters, then it converts the plaintext to lowercase, and for every letter in the plaintext, it performs a shit of n and returns out a cipher text. The function is also written to manage exception by using the try and except keywords.

```
def encrypt (n, plaintext):
  result = ''
  for n in plaintext.lower():
     try:
          i = (key.index(l) + n; % 26
          result += key[i]
      except ValueError:
          result += l
    return result.lower()
```

Figure 2 Caesar's Cipher

2. Vigenere Cipher

A code for implementing Vigenere Cipher is depicted in Figure 3. This was implemented into an encrypt function that takes the plaintext and the key as parameters. The inbuilt zip and cycle functions in python is used to create pairs among the plaintext and the key. The lambda and reduce functions are used to model a tabula recta data set.

```
def encrypt (key, plaintext):
    pairs = zip(plaintext, cycle(key))
    result = ''
    for pair in pairs:
      total = reduce(lambda x, y: ALPHA, index(x)
      + ALPHA, index(y), pair)
         Result += ALPHA(total % 26)
    Return results.lower()
```

Figure 3 Vigenere Cipher

3. Blowfish

The Blowfish Algorithm is divided into two, the key expansion part and the data encryption part. Blowfish is a variable-length key, 64-bit block cipher. The algorithm consists of two parts: a key-expansion part and a data- encryption part. Key expansion converts a key of at most 448 bits into several sub-key arrays totaling 4168 bytes. Blowfish has 16 rounds. The main algorithm divides the input into two 32 bit halves then in each round it XORs each half and swap the halves. This is done for 16 rounds after which the last swap will be reversed and the two 32 bits halves will be recombined to get the cipher text (Figure 4).

```
def bfexcrypt(x):
    """
    Main encryption algorithm .. depends on subkeys
    having already been generated from key using
    initialize()
    """

    # Divide 64 bit chunk into left and right halves
    xL = x>>32
    xR = x & 0xFFFFFFFFL

    # 16 round Feistel network
    for i in range(16):
       xL = xL ^ parray[i] # XOR with P.box
       xR = F(xL) ^ xR     # apply F to xL, XOR with xR
       xR,xL = xL,xR       #swap xR, xL

    # Reverse most recent swap
    xR,xL = xL,xR

    # XOR using last two P.boxes
    xR = xR ^ parray[16]
    xL = xL ^ parray[17]
    return (xL<<32) | xR
```

Figure 4 Blowfish Algorithm

4. Twofish

Twofish is a symmetric block cipher; a single key is used for encryption and decryption. Twofish has a bloc k size of 128 bits, and accepts a key of any length up to 256 bits. Twofish is fast on both 32-bit and 8-bit CPUs (smart cards, embedded chips, and the like), and in hardware. It's flexible; it can be used in network applications where keys are c hanged frequently and in applications where there is little or no RAM and ROM available (Figure 5). Twofish also has 16 rounds. It divides the 32 bit word from each round into for bytes which are sent through a matrix of 8 bit input and combined back into a 32 bit word.

```
Def encrypt(self, block):
    """Encrypt blocks."""

    if len(block)% 16:
        raise ValueError, "block size must be a multiple
        of 16"
    ciphertext = ''

    while block:
        a, b, c, d = struct.unpack("<4L", block[0:16])
        temp = [a, b, c, d]
        encrypt (self.context, temp)
        ciphertext += struct.pack("<4L", *temp)
        block = block[16:]

    return ciphertext
```

Figure 5 Twofish Algorithm

5. Message Digest 5

MD5 takes an input of arbitrary length and produces a 128 bit "fingerprint" or "message digest" as its output. This shows the padding and the four word buffer function used to compute the message digest (Figure 6). Three functions was written to encode, decode and test respectively.

```
Def encode(input, len):
    k = len >> 2
    res = apply struct pack, ("%iI" % k,)+ tuple(input[:k]))
    return string.join(res, "")

def Decode(input, len):
    k = len >> 2
    res = struct.unpack("%iI" %k, input[:leng])
    return list(res)

def test():
    print 'md5("hallo").digest()'
     from md5 import new
    print 'new("hallo").digest()'
```

Figure 6 MD5 Algorithm

6. Whirlpool

It takes a message of any length less than 256 bytes and returns a digest of 512 bytes. A class was implemented to contain all the four major operations of the whirlpool algorithm (Figure 7). Also a code section is shown where the algorithm checks to see if the message length is less than 256 bytes (Figure 8).

```
DIGESTBYTES = 64
class WhirlpoolStruct:
    def _init_(self):
        self.bitLength = [0]+32
        self.buffer = [0]*64
        self.bufferBits = 0
        self.bufferPos = 0
        self.hash = [0]*8

def WhirlpoolInit(ctx):
    ctx = WhirlpoolStruct()
    return
```

Figure 7 Whirlpool (a)

```
if bufferRem + sourceBits < 8:
    bufferBits += sourceBits
else:
    bufferPos += 1
    bufferBits += 8 - bufferRem
    sourceBits -= 8 - bufferRem
    if bufferBits == 512:
        processBuffer(ctx)
        bufferBits = 0
        bufferPos = 0
    buffr[bufferPos] = b << (8 - bufferrem)
    bufferBits += sourceBits
ctx.bufferBits += bufferBits
ctx.bufferPos = bufferPos
```

Figure 8 Whirlpool (b)

7. Data Encryption Standard (DES)

DES is a block cipher based on the Feistel block cipher. It encrypt data in block of size 64 bits, consisting of a number of rounds where each round contains bit-shuffling, non-linear substitutions (S-boxes) and exclusive OR operations. Most symmetric encryption schemes today are based on this structure.

8. Advanced Encryption Standard (AES)

AES is a block cipher with a block length of 128 bits. It allows for three different key lengths: 128, 192, or 256 bits. A pseudo code summary of AES is shown in figure 4.2 and a full summary of AES encryption and decryption is shown in Figure 9.

```
Cipher(byte int[16], byte out[16], key_array round_key[Nr+1]);
begin
    byte state[16];
    state = in;
    AddRoundKey(state, round_key[0]);
    for i = 1 to Nr-1 stepsize 1 do
        SubBytes(state);
        ShiftRows(state);
        MixColumns(state);
        AddRoundKey(state, round_key[i]);
    end for
    SubBytes(state);
    ShiftRows(state);
    MixColumns(state);
    AddRoundKey(state, round_key[Nr]);
end
```

Figure 9 AES

9. Diffie Hellman (DH)

Diffie Hellman is a way of generating a shared secret between two people in such a way that the secret can't be seen by observing the communication.

10. Secure Hash Algorithm 1 (SHA -1)

SHA-1 is the most widely used of the existing SHA hash functions. It is employed in several widely used applications and protocols including Secure Socket Layer (SSL) security.

IV. PERFORMANCE EVALUATION

In this section, the performance evaluation is presented.

1. Block Size Comparison

The block sizes of the selected algorithms are compare in table 1. It is shown that blowfish and DES require only 64 bits while RSA and MD-5 require 512 bits.

Table 1 Block Size Comparison

Algorithms	BLOCK SIZE (bits)
AES	128
DES	64
BLOWFISH	64
MD-5	512
SHA-1	
RSA	Minimum of 512

2. Key Length Comparison of Algorithms

Table 2 gives the key length comparison of some of the encryption algorithms. DES requires 56 bits while RSA requires 1024 bits. Blowfish requires between 128 to 448 bits.

Table 2 Key Length Comparison

Algorithms	Key Length (Bits)
AES	128 or 192 or 256
DES	56
BLOWFISH	128 to 448
MD5	128
SHA-1	160
RSA	1024

3. Encryption Time

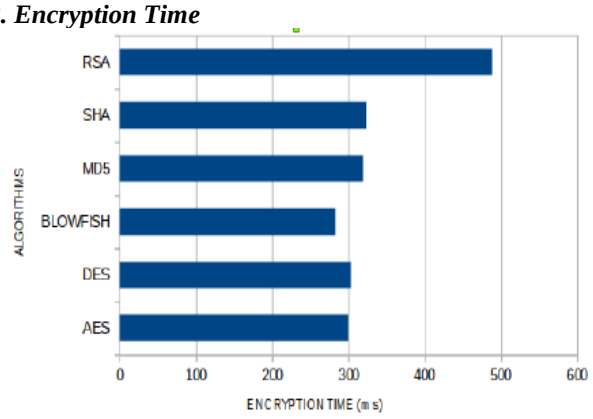

Figure 10 Comparison of Encryption Time

The average encryption time of the selected algorithms are compared in Figure 9. The average time was obtained by running each of the algorithms 50 times, logging the encryption time of each run and calculating the average time of each. Blowfish algorithm is the fast while RSA is the slowest.

Table 3 Attacks on Algorithms

ALGORITHM	KNOWN ATTACKS
AES	Brute Force, XSL Attack, Known-key Distinguishing Attack or Biclique Attack, Side-Channel Attack
DES	Meet-in-the-Middle Attack, the Davies Attack, Differential cryptanalysis, Linear cryptanalysis, differential-linear cryptanalysis, Memory Tradeoff, Exhaustive Search, ne-round attack, Full 16 round Attack
BLOWFISH	Birthday attacks, Known-plaintext attacks, SWEET32 Attack
MD-5	Collision Attack
SHA-1	The SHAppening, SHAttered
RSA	Elementary Attacks e.g Coppersmith's short pad attack, Franklin Reiter related message attack, Hastad's Broadcast attack. Implementation attacks such as Timing attacks and Random Fault.

4. Security

The security of the selected algorithms are compared in terms of the known attacks on encryption types. This shows the number of known attacks to break the selected algorithms.

V. CONCLUSION

In this study we have examined the implementation of some cryptography techniques and compare their performances. The comparison was done using metrics such as block sizes, key lengths, and encryption time and security issues. The implementation was done with an object-oriented programming language, Python. RSA has the longest encryption time with approximately 490ms, and blowfish with the least encryption time of approximately 290ms, RSA and MD-5 have the largest block size of 512 bits while DES and blowfish have the least block size of 56 bits.

REFERENCES

[1] P. L. Rivest, "Cryptography," in *Handbook of Theoretical Computer Science*, 1990.

[2] W. Diffiee and M. Hellman, "New Directions in Cryptography," in *IEEE Transactions on Information Theory*, Ronneby, Sweden, 1976, vol. 22, no. 6, pp. 644-654: IEEE.

[3] E. Thanbiraja, G. Ramesh, and D. R. Umarani, "A survey on Various Most Common Encryption Techniques," *International Journal of Advanced Research in Computer Science and Software*

Engineering 2012, 2(7), July 2012: 226-233., Research Paper vol. 2, no. 7, pp. 226-233, July 2012 2012.
[4] R. Bhanot and R. Hans, "A Review and Comparative Analysis of Various Encryption Algorithms," *International Journal of Security and Its Applications*, vol. 9, no. 4, pp. 289-306, 2015.
[5] K. P. Arindam, L. Devnath, and M. R. Islam, "A Highly Secured Mathematical Model for Data Encryption Using Fingerprint Data," *International Journal on Data Science and Technology*, vol. 2, no. 4, pp. 46-50, 2016.
[6] G. C. Kessler. (2017, June 2017). *An Overview of Cryptography*. Available: http://www.garykessler.net/library/crypto.html#intro
[7] G. V. Bard, *Algebraic Cryptanalysis* ( ). US: Springer US, 2009, p. 300.
[8] S. S. Wagstaff, *Cryptanalysis of Number-Theoretic Ciphers*, First ed. FL, USA CRC Press, Inc. Boca Raton, , 2003, p. 311.
[9] W. Stallings, *Cryptography and Network Security Principles and Practices*, 4th ed. Prentice Hall, 2005.
[10] B. Horrocks. (2003). *Uses of ROT13*.
[11] E. Biham and A. Shamir, "Differential Cryptanalysis of DES-like Cryptosystems," in *Crypto '90 Proceedings of the 10th Annual International Conference on Advances in Cryptology*, 1990, pp. 2-21: Springer Verlag London UK, 1991.
[12] S. Bruce, "Description of a New Variable-Length Key, 64-Bit Block Cipher (Blowfish)," 1993.
[13] M. Shiho and L. Y. Yiqun, "Cryptanalysis of Twofish II," pp. 1-8Available: https://www.schneier.com/twofish-analysis-shiho.pdf
[14] S. Vollala, V. V. Varadhan, K. Geetha, and N. Ramasubramanian, "Design of RSA processor for concurrent cryptograpic transformations," *Microelectronics Journal*, Research Paper vol. 63, no. May 2017, pp. 112-122, May 2017 2017.