A CRYPT ANALYSIS OF THE TINY ENCRYPTION ALGORITHM IN KEY GENERATION

Mohammad Shoeb, Vishal Kumar Gupta
Computer Science & Engg. , Institute Of Technology & Management

Received: 07-07-2012, **Revised:** 16-09-2012, **Accepted:** 24-10-2012, **Published online: 13-12-2012**

Abstract— Encryption algorithms are used for security over wireless communications, but securing data also consumes resources. Major important factors to consider when designing a cryptographic system are performance, speed, size, and security. Tiny Encryption Algorithm (TEA), and the Extension of TEA (XTEA) are examples of cryptographic algorithms. There is a requirement to specify cryptographic strength in an objective manner rather than describing it using subjective descriptors such as weak, strong, acceptable etc. This paper proposes the Study on a Tiny Encryption Algorithm. The Tiny Encryption Algorithm (TEA) is a cryptographic algorithm designed to minimize memory footprint and maximize speed. It is a Feistel type cipher that uses operations from mixed (orthogonal) algebraic groups. In this Paper, we propose Tiny Encryption Algorithm (TEA) to some standard for random number generator tests.

KEYWORDS: Data Encryption Standard, Cryptography, TEA (Tiny Encryption Algorithm), XTEA (Extension of Tiny Encryption Algorithm), cryptographic system, RC2, Blowfish, and AES.

1. INTRODUCTION

As computer systems become more pervasive and complex, security is increasingly important. Cryptographic algorithms and protocols constitute the central component of systems that protect network transmissions and store data. The security of such systems greatly depends on the methods used to manage, establish, and distribute the keys employed by the cryptographic techniques. Even if a cryptographic algorithm is ideal in both theory and implementation, the strength of the algorithm will be rendered useless if the relevant keys are poorly managed. Cryptography is the art and science behind the principles, means, and methods for keeping messages secure. Cryptanalysis is a study of how to compromise (defeat) cryptographic mechanism. There are two classes of key-based encryption algorithms: symmetric (or secret-key) and asymmetric (or public-key) algorithms. Symmetric algorithms use the same key for encryption and decryption, whereas asymmetric algorithms use different keys for encryption and decryption.[1] Ideally it is infeasible to compute the decryption key from the encryption key. Feistel ciphers (see Feistel, 1973) are a special class of iterated block ciphers where the cipher text is calculated from the plain text by repeated application of the same transformation or round function. In a Feistel cipher, the text being encrypted is split into Two halves. The round function, F, is applied to one half using a sub key and the output of F is (exclusive-or-ed (XORed)) with the other half. The two halves are then swapped. Each round follows the same pattern except for the last round where there is often no swap. The focus of this project is the TEA Feistel Cipher.[4]

The Tiny Encryption Algorithm (TEA) is a symmetric (private) key encryption algorithm created by David Wheeler and Roger Needham of Cambridge University and published in 1994. It was designed for simplicity and performance, while seeking encryption strength on par with more complicated and resource-intensive algorithms such as DES (Data Encryption Standard). Wheeler and Needham summarize this as follows: “it is hoped that it can easily be translated into most languages in a compatible way… it uses little set up time and does enough rounds to make it secure… it can replace DES in Software, and is short enough to write into almost any program on any computer.” [6]

2. RESEARCH STUDY

The Tiny Encryption Algorithm (TEA) is a cryptographic algorithm designed by Wheeler and Needham (1994). It is designed to minimize memory footprint and maximize speed. This research presents the cryptanalysis of the Tiny Encryption Algorithm based on the differential cryptanalysis proposed by Biham and Shamir (1992) and related-key cryptanalysis proposed by Kelsey, Schneier, and Wagner (1997).

In 1994, the cipher Tiny Encryption Algorithm is a 64-round Feistel cipher that operates on 64-bit blocks and uses a 128-bit key. Designed by Wheeler and Needham, it was presented at FSE 1994. And for simple design, the cipher was subsequently well studied and came under a number of attacks.

In 1996, Kelsey et al. established that the effective key size of TEA was 126 bits [2] and this result led to an attack on Microsoft’s Xbox gaming console where TEA was used as a hash function [12].

In 1997. Kelsey, Schneier and Wagner constructed a related-key attack on TEA with 223 chosen plaintexts and 232 times [4]. Following these results, TEA was redesigned by Needham and Wheeler to yield Block TEA and XTEA (eXtended TEA) [10]. While XTEA has the same block size, key size and number of rounds as TEA, Block TEA caters to variable block sizes for it applies the XTEA round function for several iterations. Both TEA and XTEA are implemented in the Linux kernel.

In 1998. To correct the weaknesses in Block TEA, Needham and Wheeler designed Corrected Block TEA or XXTEA. This cipher uses an unbalanced Feistel network and operates on variable length messages. The number of rounds is determined by the block size, but it is at least six. An attack on the full Block TEA is presented in [11], where some weaknesses in XXTEA are also detailed.

In 2002–2010. A number of cryptanalysis results on the TEA family were reported in this period.[10]

3. High Level Description of TEA

3.1 Background Information

In recent years, many symmetric block ciphers have been presented. The Tiny Encryption Algorithm (TEA) was first published in 1994 by Roger Needham, and David Wheeler from Cambridge University of the United Kingdom. The Tiny Encryption Algorithm (TEA) is a compromise for safety, ease of implementation, lack of specialized tables, and reasonable performance. TEA can replace 1 DES in software, and is short enough to integrate into almost any program on any computer. Some attempts have been made to find weakness of the Tiny Encryption Algorithm. The motivation of this research is to study and implement the proposed attacks on TEA to determine whether such attempts are practically feasible. The Tiny Encryption Algorithm is a Feistel type cipher (Feistel, 1973) that uses operations from mixed (orthogonal) algebraic groups. A dual shift causes all bits of the data and key to be mixed repeatedly. The key schedule algorithm is simple; the 128-bit key K is split into four 32-bit blocks K = (K[0], K[1], K[2], K[3]). TEA seems to be highly resistant to differential cryptanalysis (Biham et al., 1992) and achieves complete diffusion (where a one bit difference in the plaintext will cause approximately 32 bit differences in the cipher text). Time performance on a workstation is very impressive.

TEA was initially designed to be an extremely small algorithm when implemented in terms of the memory foot print required to store the algorithm. This was accomplished by making the basic operations very simple and weak; security is achieved by repeating these simple operations many times. As the basic operations are very simple TEA is also regarded as a very high speed encryption algorithm. These properties have made TEA a choice for both weak hardware or software encryption implementations in the past as TEA can be operated in all modes as specified by DES as outlined in the specification. [12]

There are many notable fallbacks of TEA and it is considered broken. The first issue of note is that TEA uses “equivalent keys” thus weakening the effectiveness of its key length and requires only complexity O(2^32) using a related key attack to break. This is much less than the intended key brute force strength of 2^128. Two revisions of TEA have since been published including XTEA and XXTEA which boast enhanced security and the ability to support arbitrary block sizes making TEA obsolete as a secure cryptographic method. The specification for TEA states a 128-bit key is to be divided into four 32-bit key words and the block size of each encryption is 64 bits, of which is to be divided into two 32-bit words. TEA utilizes a Feistel scheme for its encryption rounds in which 1 round of TEA includes 2 Feistel operations and a number of additions and bitwise XOR operations as shown below in figure 1.

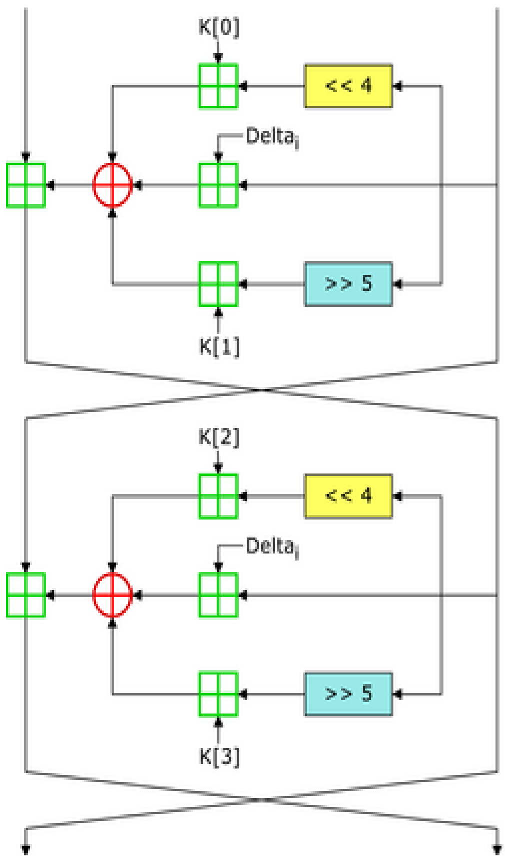


Figure 1 TEA encryption

The specification simply “suggests” that 32 TEA rounds be completed for each 64-bit block encrypted, all online resources appear to follow this suggestion. This means a full encryption of a block is simply 32 TEA rounds which involve 64 Fiestel rounds.

TEA utilizes a value denoted as DELTA in the specification which is defined as 5−1 ∗231 which is “derived from the golden ratio” is used in multiples for

each round to prevent symmetry based exploits on the Feistel operations as shown in figure 1. The key schedule simply exists as exclusive OR'ing the key words with a shifted value of the last state of each of the block words, this operation "causes all bits of the key and data to be mixed repeatedly".
A Decryption simply involves the inverse operations in reserve order; that is 32 rounds of subtract and XOR operations in the opposite order of the encryption. TEA is considered a high speed algorithm as there is virtually no set up or complex key schedule for the encryption and decryption algorithm. A second serious shortcoming of the TEA algorithm is the lack of official test vectors in the specification and proof as to why the operations were chosen and considered secure other than stating a number of other algorithms were tested. The specification also does not explicitly state bit ordering.[15]

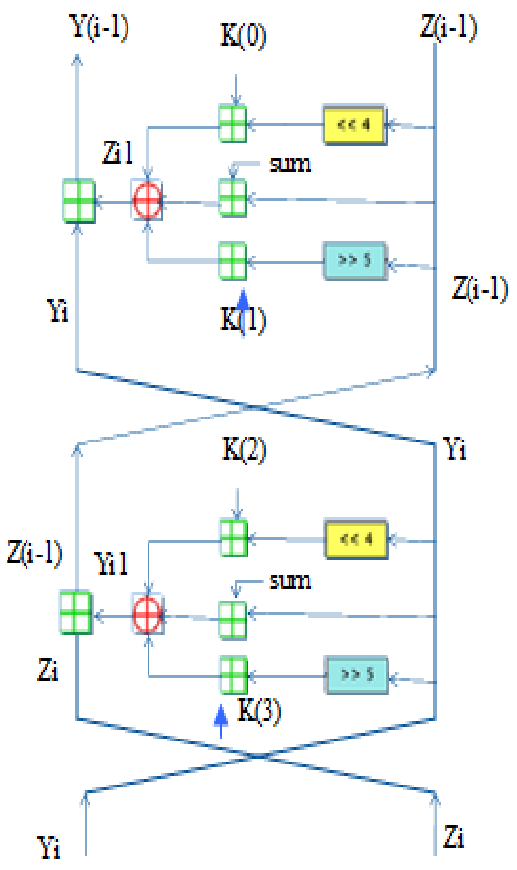


Figure 2. TEA decryption

3.2 NOTATION

The following notation is necessary in TEA..

- **Hexadecimal numbers**: It will be subscripted with "h," e.g., 10 = 16. h
- **Bitwise Shifts:** The logical shift of x by y bits is denoted by x << y. The logical right Shift of x by y bits is denoted by x >> y.
- **Bitwise Rotations:** A left rotation of x by y bits is denoted by x <<< y. A right rotation of x by y bits is denoted by x >>> y.
- **Exclusive-OR:** The operation of addition of n-tuples over the field (also known as 2F Exclusive-or) is denoted by x⊕y.
- **Integer Addition:** The operation of integer addition modulo is denoted by x 2n y. (where x, y ∈). The value of n should be clear from the context. 2nZ
- **Integer Subtraction:** The operation of integer subtraction modulo is denoted by 2n. x y (where x, y ∈).

3.3 Software Implementation

In the original presentation of TEA, David Wheeler and Roger Needham also included some source code for software implementation. The designer's state that the particular algorithm used in the source code for the software implementation of TEA was chosen because it was thought to be a compromise between security and simplicity of design. [6]This algorithm was neither the fastest nor the slowest of those tested prior to the final down selection to one algorithm. As a part of the original presentation of TEA, Wheeler and Needham published the following source code. In the software implementation, the source code separates the 64-bit block into two 32-bit numbers labeled y and z. As previously stated, TEA contains two rounds for one cycle of encryption or decryption. Round one (and subsequent odd rounds) operates on y, and sub keys K[0] and K[1]. Round two (and subsequent even rounds) operates on z, and K [2] and K [3].

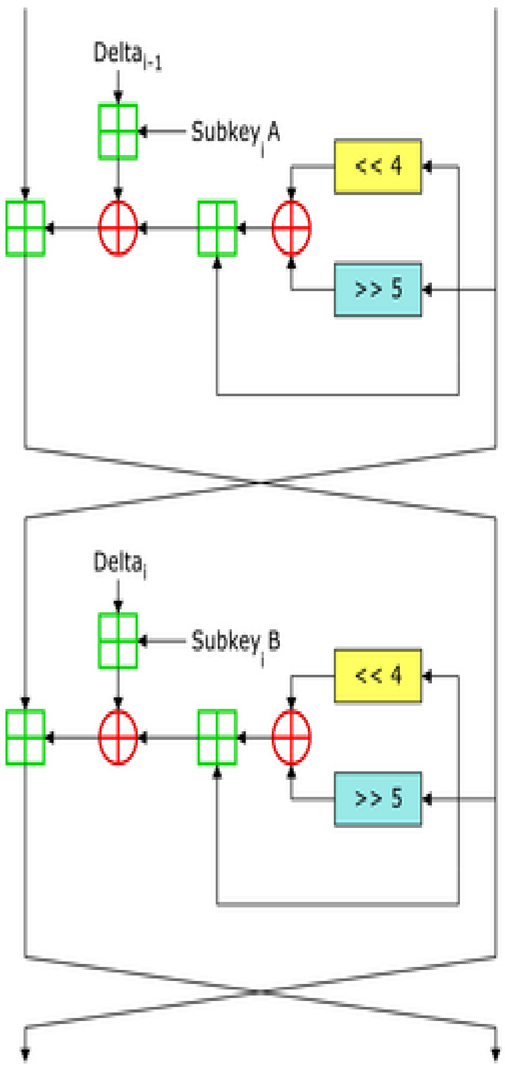
```
void code(long* v, long* k)
{
unsigned long y=v[0],z=v[1], sum=0, /* set up */
delta=0x9e3779b9, /* a key schedule constant */
n=32 ;
while (n-->0)
{ /* basic cycle start */
sum += delta ;
y += ((z<<4)+k[0]) ^ (z+sum) ^ ((z>>5)+k[1]) ;
z += ((y<<4)+k[2]) ^ (y+sum) ^ ((y>>5)+k[3]) ;
} /* end cycle */
v[0]=y ; v[1]=z ;
}
void decode(long* v,long* k)
{
unsigned long n=32, sum, y=v[0],
z=v[1],delta=0x9e3779b9 ;
sum=delta<<5 ;
while (n-->0)
{ /* start cycle */
z-= ((y<<4)+k[2]) ^ (y+sum) ^ ((y>>5)+k[3]) ;
y-= ((z<<4)+k[0]) ^ (z+sum) ^ ((z>>5)+k[1]) ;
sum-=delta ;
} /* end cycle */
v[0]=y ; v[1]=z ;
}
```

4. Extensions of Tiny Encryption Algorithm

4.1 Background Information

After some weaknesses and vulnerabilities of TEA were Discovered and documented, Wheeler and Needham decided to present a new implementation of TEA and called it Extensions of TEA (XTEA). XTEA was first presented in 1997, three years after TEA was first presented. Similar to TEA, XTEA is also a block cipher, which uses Feistel structure. XTEA also uses the same 64-bit block and a 128-bit key as TEA. The same 64 rounds, or 32 cycles, are also recommended for the algorithm. The vulnerabilities of TEA were discovered using differential related-key attacks [4]. Therefore, XTEA attempts to correct the weaknesses by improving some aspects of the algorithm. The first change that was introduced in XTEA was a correction to the key schedule algorithm. In the updated XTEA, the introduction of subkeys is added more slowly. Also, the subkeys are selected by using two bits of the variable 'sum'. In addition, a shift of 11 is also introduced in the key schedule to help create an irregular sequence of the subkeys. Some other changes introduced in XTEA is a rearrangement of the addition, shifts, and XOR operations. The following diagram shows XTEA. Instead of defined placement of the subkeys, now subkeys are introduced as subkey 'A' and subkey 'B'.

Left[i] Right [i]



Left [i+1] Right [i+1]

Figure 3. Feistel Structure of XTEA for 2 rounds

4.2 Software Implementation

Similar to the publication for TEA, when XTEA was published in 1997, Wheeler and Needham also included Source code for XTEA. As previously stated, XTEA contains two rounds for one cycle of encryption or decryption. Round one (and subsequent odd rounds) operates on y. The sub key selection in this round depends on the value of 'sum&3', which is the variable sum logic AND with 3,0x03h, or 0011b. Round two (and subsequent even rounds) operates on z. The subkey selection in this round depends on the value of sum>>11 & 3, which is SUM shifted by 11 and then a logic AND with 3, 0x03h, or 0011b.

```
XTEA (long * v, long * k, long N)
  {
```

```
  unsigned long y = v [0];
  unsigned z =v [1];
  unsigned DELTA = 0x9e3779b9;
  if (N>0)
{
/* coding */
unsigned long limit = DELTA*N;
unsigned long sum = 0;
while (sum!= limit)
y += (z<<4 ^ z>>5) + z ^ sum + k[sum&3],
sum += DELTA,
z += (y<<4 ^ y>>5) + y ^ sum + k [sum>>11 &3];
}
else
{
/* decoding */
Unsigned long sum=DELTA*(-N);
while (sum)
z -= (y<<4 ^ y>>5) + y ^ sum + k[sum>>11 &3],
sum -= DELTA,
y -= (z<<4 ^ z>>5) + z ^ sum + k[sum&3] ;
}
v [0] =y;
v [1]=z ;
return;
}
```

4.3 OPERATION OF XTEA IN ITERATION

The relation between the output (Left [*i*+1], Right [*i*+1]) and the input (Left[*i*], Right[*i*]) for the *i*th cycle of XTEA is defined as follows:

1. Left [*i*+1] = Left[*i*] + F (Right[*i*], K [2*i*-1], delta [*i*-1]),
2. Right [*i*+1] = Right[*i*] + F (Left [*i*+1], K [2*i*], delta[*i*]),
3. Delta[*i*] = (*i*+1)/2 * delta The round function, F, is defined by F (M, K [*], delta [**]) = ((M<<4) xor (M>>5)) + (M xor sum) + K [*].

The addition employed is of modulo 2^{32}.

5. EXPERIMENTAL RESULTS

5.1 Strict Plaintext Avalanche Criterion (SPAC):

SPAC relates to bit changes in ciphertext corresponding to a single bit change in the plaintext. During experiments, a block of plaintext P was taken. For a key K, ciphertext C was obtained by applying the TEA algorithm. Now, plaintext P' was obtained by complementing the first bit of the original plaintext P. This new plaintext block P' is now encrypted using the same key K to obtain the new cipher text C'. Now the number of bits at which C and C' differ is calculated. This number is further divided by 64 (each block is of 64-bit length) to obtain the probability of bit-change. Let's call this probability as bit-change probability. The above experiment is repeated for different P'. There are 64 possible combinations of P', each obtained by flipping one of the 64 bits of the original plaintext block P. The experiment was performed for all the 64 possibilities of P'. The average of all the 64 bit-change probabilities was obtained. The expected value of this probability for a secure algorithm is 0.5. These entire experiments were repeated for 1 to 32 rounds of TEA. The X-axis corresponds to the number of rounds used for encryption and decryption. The Y-axis shows the bit-change-probability deviation from the expected value of 0.5. As expected, the avalanche effect goes closer to the expected value of 0.5 as the number of rounds increases. SPAC seems to be a fair metric though it is difficult to say that it is the best metric. As can be seen from the graph, for rounds 0, 1, 2 and 3, TEA does not produce enough avalanche effect. After four rounds, the avalanche effect approaches the expected value of 0.5 and there is no significant change in the avalanche effect. [11]

6. CONCLUSION

Tiny encryption algorithm is one of the simplest algorithms to implement in hardware. It can be employed, where time is a constraint, i.e. a tradeoff can be made between the levels of security desired and the time to encrypt or decrypt, in terms of number of cycles.
Testing the random numbers is very important in all cryptographic applications. In the absence of generally accepted norms to be used to measure and specify cryptographic strength, it is desirable to carry out a number of tests on different ciphers to get an exposure to their strength and weakness. With this objective, we carried out a variety of randomness and security strength tests which are presented in this paper. The strength of TEA and DES emerges quite significantly. Also as a heuristic technique to be used in cryptanalysis, Tabu search performs better than genetic algorithm and simulated annealing, based on the randomness of the numbers generated from these algorithms .

7. REFERENCES

1. S. Liu, O. Gavrylyako, and P. Bradford, “Implementing the TEA algorithm on Sensors”, Proceedings of the 42nd annual Southeast regional conference, pp. 64-69, 2004.
2. D. Wheeler and R. Needham, “TEA, a tiny encryption algorithm”, Proc. Fast Software Encryption: Second International Workshop, Lecture Notes in Computer Science, vol. 1008, pp. 363-366, December 2004.
3. P. Israsena, “Securing Ubiquitous and Low-Cost RFID Using Tiny Encryption Algorithm”, Wireless Pervasive Computing, 2006 1st International Symposium, pp. 1-4, 2006.
4. J. Kelsey, B. Schneider, and D. Wagner, “Related-Key Cryptanalysis of 3-WAY, Biham-DES, CAST, DES-X, NewDES, RC2, and TEA”, Proceedings of the First International Conference on Information and

Communication Security: Lecture Notes In Computer Science; Vol. 1334, pp. 233-246, 1997.

5. D. Wheeler and R. Needham, "TEA Extensions", unpublished, October 1997.

6. Wheeler D., and R. Needham. TEA, a Tiny Encryption Algorithm, Proceedings of the Second International Workshop on Fast Software Encryption, Springer-Verlag, 1995, pp. 97-110.
7. J. Kelsey, B. Schneier, D. Wagner, "Key-Schedule Cryptoanalysis of IDEA, G-DES, GOST, SAFER, and Triple-DES," CRYPTO 1996 (N. Koblitz, ed.), vol. 1109 of LNCS, pp. 237–251, Springer-Verlag, 1996.
8. William Stallings. *Cryptography and Network Security Principles and Practices*, Third Edition, Pearson Education Inc., 2003.

9. Y. Ko, S. Hong, W. Lee, S. Lee, J.-S. Kang, "Related-Key Differential Attacks on 27 Rounds of XTEA and Full-Round GOST," FSE 2004 (B.K. Roy, W. Meier, eds.), vol. 3017 of LNCS, pp. 299–316, Springer-Verlag, 2004.
10. E. Lee, D. Hong, D. Chang, S. Hong, J. Lim, "A Weak Key Class of XTEA for a Related-Key Rectangle Attack," VIETCRYPT 2006 (P.Q. Nguyen, ed.), vol. 4341 of LNCS, pp. 286–297, Springer-Verlag, 2006.
11. J. Lu, "Related-key rectangle attack on 36 rounds of the XTEA block cipher," International Journal of Information Security, vol. 8(1), pp. 1–11, Springer-Verlag, 2009.
12. M. D. Moon, K. Hwang, W. Lee, S. Lee, J. Lim, "Impossible Differential Cryptanalysis of Reduced Round XTEA and TEA," FSE 2002 (J. Daemen, V. Rijmen, eds.), vol. 2365 of LNCS, pp. 49–60, Springer-Verlag, 2002.
13. R. M. Needham, D.J. Wheeler, "Correction to xtea," technical report, Computer Laboratory, University of Cambridge, October 1998, available at http://www.movable-type.co.uk/scripts/xxtea.pdf.
14. M.-J. Saarinen, "Cryptanalysis of Block TEA," unpublished manuscript, October 1998,
15. M. Steil, "17 Mistakes Microsoft Made in the Xbox Security System," Chaos Communication Congress 2005, available at http://events.ccc.de/congress/2005/fahrplan/events/559.en.html.
16. D.J. Wheeler, R.M. Needham, "TEA, a Tiny Encryption Algorithm," FSE 1994 (B. Preneel, ed.), vol. 1008 of LNCS, pp. 363–366, Springer-Verlag, 1994.
17. J.-P. Kaps, "Chai-Tea, Cryptographic Hardware Implementations of xTEA," INDOCRYPT 2008 (D.R. Chowdhury, V. Rijmen, A. Das, eds.), vol. 5365 of LNCS, pp. 363–375, Springer-Verlag, 2008.
18. Norman D., Jorstad., and Landgrave T. Smith, Jr. Cryptographic Algorithm Metrics, Technical Report, Institute for Defense Analyses, Science and Technology Division, Jan 1997.
19. Bruce Schneier. *Applied Cryptography, 2nd Edition*, John Wiley & Sons, 1996.
20. Rajashekarappa, Sunjiv Soyjaudah, K. M.,"Overview of Differential Cryptanalysis of Hash Functions using SMART Copyback for Data" published at International Journal of Computer Science and Technology(IJCST), Vol. 4, Issue 1, Jan-March 2013, pp 42-45.
21. Behrouz A. Forouzan, (2006)"Cryptography and Network Security", Firstedition, McGraw-Hill.
22. Atul Kahate, " Cryptography and Network Security", TMH, 2003.
23. A.Zugaj, K. Górski, Z. Kotulski, A. Paszkiewicz, J. Szczepański,(2000) "New constructions in linear cryptanalysis of block ciphers", ACS'2000, October.
24. Rajashekarappa and Dr. K M S Soyjaudah "Heuristic Search Procedures for Cryptanalysis and Development of Enhanced Cryptographic Techniques" Published at International Journal of Modern Engineering Research (IJMER), May 2012, Vol.2, Issue.3, pp-949-954.