Improvised Blowfish under Bouncy Castle Framework

Rakhee Single, S. G. Vaidya, M. B. Ansari

Abstract*: Greater demand for Internet applications require data to be transmitted securely with improved facilities for networking. But the transmission of data in the public communication system is not secure because of the interception and improper handling by indiscreet. It is necessary to secure the information we want to convey. This need to secure information introduces the concept "Cryptography", which is the art and science of writing hidden. Cryptography before the modern age was effectively synonymous with encryption, data conversion means readable in an apparent nonsense. The proposed system uses Bouncy Castle APIs for cryptography, which is founded in 2000 year. Basically bouncy castle is a collection of APIs used in cryptography. Using Bouncy Castle APIs for cryptography provides security information such as data confidentiality, data integrity, authentication and non-repudiation.

***Index Terms*: Bouncy castle, cryptography, Blowfish algorithm, Homomorphic Encryption**

I. INTRODUCTION

The cryptography word comes from the Greek words. Oddly enough; Cryptography is the art of secret writing. More generally, people think of cryptography as the art of mutilating apparent unintelligibility information in a manner allowing a secret method of transformation. The basic service provided by cryptography is the ability to send information between the participants in a manner that prevents others from reading it. This kind of cryptography can provide other services, such as:

• Integrity Check-reassure the recipient of a message that the message has not been altered since it was generated by a legitimate source

• Authentication—verifying someone's (or something's) identity But back to the traditional use of cryptography.

A message in its original form is known as text or plain text. The information processed is known as cipher text. The process for producing a plaintext cipher text is known as encryption. The encryption backhand is called decryption; which production process plaintext from the cipher text. An algorithm for transforming an intelligible message one that is unintelligible by methods of transposition and / or substitution is known as cipher name. Some critical information used by the encryption, known only to the sender and the receiver is known as a key.

Revised Version Manuscript Received on November 04, 2016.

Rakhee Single, Department of Computer Science of Engineering, Shreeyash College of Engineering and Technology, Aurangabad (Maharashtra). India.

Prof. S. G. Vaidya, Department of Computer Science of Engineering, Shreeyash College of Engineering and Technology, Aurangabad (Maharashtra). India.

Prof. M. B. Ansari, Department of Computer Science of Engineering, Shreeyash College of Engineering and Technology, Aurangabad (Maharashtra). India.

The text conversion process clear to encrypt text using an encryption algorithm and a key s called encoding. [1]. Reverse encoding is known as decoding; which process cryptogram again conversion in plaintext using an encryption algorithm and a key. Cryptanalysis is the study of the principles and methods of transformation of an unintelligible new posts in an intelligible message without knowledge of the key. Cryptanalysis is also called code breaking. Cryptography helps many security goals to avoid a security problem. Cryptography is widely used today, due to the safety advantages such as privacy, authentication, integrity, and access control [2]

Bouncy castle has origin in Australia and founded in 2000. Bouncy castle is nothing but the collection of APIs used in cryptography. Firstly, it has included API for the Java programming language. In 2006, it included C ++ API programming language enemy. It provided support for J2ME, a JCE provider / JCA base and generation of X.509 certificates. Bouncy castle strongly focus focus on compliance and adaptability. It offers public support services that include emission monitoring system, the dev mailing list and a wiki all available on the website. provided commercial support under resources for the relevant APIs listed on the site Bouncy Castle. [11]

The Bouncy Castle architecture consists of two main elements that support the basic cryptographic capabilities. These are known as "light" API, and the JCE provider. There are other components that rely on the JCE provider that supports additional features such as support for PGP, S / MIME. The low-level, or 'light-weight', API is a set of APIs that implement all the underlying cryptographic algorithms. The APIs were designed to be simple enough to use if needed, but provided the basic building blocks for the JCE provider. The intent is to use the low-level API in memory constrained devices (Java ME) or when easy access to the JCE libraries is not possible (such as distribution in an applet). As the light-weight API is just Java code, the JVM does not impose any restrictions on the operation of the code, and at early times of the Bouncy Castle history it was the only way to develop strong cryptography that was not crippled by the Jurisdiction Policy files which prevented any JCE providers from performing "strong" encryption.

The JCE-compatible provider is built upon the low-level APIs. As such, the source code for the JCE provider is an example of how to implement many of the "common" crypto problems using the low-level API. Many projects have been built using the JCE provider, including an Open Source Certificate Authority EJBCA [11].

II. LITERATURE SURVEY

Types of Cryptography Encryption algorithms are classified in two broad categories- Symmetric key Cryptography and Asymmetric Key Cryptography [2].

1) Symmetric Key Cryptography

The key used for encryption is similar to the key used in decryption in symmetric key cryptography. So, key distribution for this cryptography has to be made to the transmission of information [7]. As security depends on the nature of the key length, key plays an important role in this cryptography. It includes some symmetric key algorithms such as DES &AES.

2) Asymmetric Key Cryptography

Two different keys are used for encryption and decryption in asymmetric key cryptography. Those keys are public and private. The public key is one which is available to everyone in the network. Those who want to encrypt the plaintext should know the Public Key of receiver. Only the authorized person can decrypt the cipher text through his/her own private key. Private Key is kept secret from the others for the purpose of security so that we can prevent secrete data from eavesdropper [6]. Symmetric Encryption Algorithm is faster as compared to Asymmetric key algorithms. The memory requirement of Symmetric algorithm is less than asymmetric.

3) Data Encryption Standard (DES)

DES (Data Encryption Standard) is the first encryption standard recommended by NIST (National Institute of Standards and Technology). It was developed in 1947 by IBM and accepted as a national standard in 1997. DES is a 64-bit block cipher which uses 56-bit key [9]. This algorithm consists of a permutation of sixteen rounds block cipher and a final permutation. DES applications are very famous in the commercial, military, and other domains. DES standard is public & the design criteria used are classified.

4) Advanced Encryption Standard (AES)

In the year 2000 AES was discovered by scientists Joan and Vincent Rijmen. AES uses Rijndael block cipher and Rijndael key. Length of a Block in AES can be of 128, 192 or 256-bits. If both the key-length and block length are 128-bit, Rijndael performs 9 processing rounds. If the block/key is 192-bit, it will per-form 11 processing rounds & for 256-bits, Rijndael performs 13 processing rounds [10].

Each processing round involves the following four steps:

1. Substitute bytes – It uses an S-box to perform a byte by byte substitution of the block,
2. Shift rows – It is simple permutation,
3. Mix column – It is a substitution method where data in each column from the shift row step is multiplied by the algorithm's matrix and
4. Add round key – It is the key for the processing round is XORed with the data [1].

III. PROPOSED SYSTEM

This Software tool involves Cryptographic enciphering and deciphering along with File Splitting and Merging mechanisms. In this approach a file which has secret data is sliced into desired number of pieces upon user's specification and then the cryptographic encryption phase is carried out. In order to achieve more security we can adopt more than one cryptographic scheme which definitely ensures nil suspicion and more security. In this paper, we differentiate the cryptographic scheme by providing different key for each encryption of sliced files; provided the key should be given correctly at the time of decryption to avoid erroneous results. We are using modified Blowfish algorithm for Encryption and Decryption of data which serves as a better solution both in terms of performance and as well as security. This enhancement in security and performance is sustainably justified in our previous work [2].

In the file joining phase, En-Ciphered files thus obtained from the different En- Ciphering techniques are merged and hence transmitted to reception side as a single file which makes the file infeasible to breach and suspicion less to get to know that varying crypto schemes are adopted. And hence the data security is maximized. At the receiver's end, We once again splits the files and decrypts it using the same algorithm and then joins all split files together to retrieve the original message. When coming to cryptographic perspective, in order to enhance the performance of the Blowfish Algorithm it is proposed to modify the F-Function by adopting the concept of multithreading.

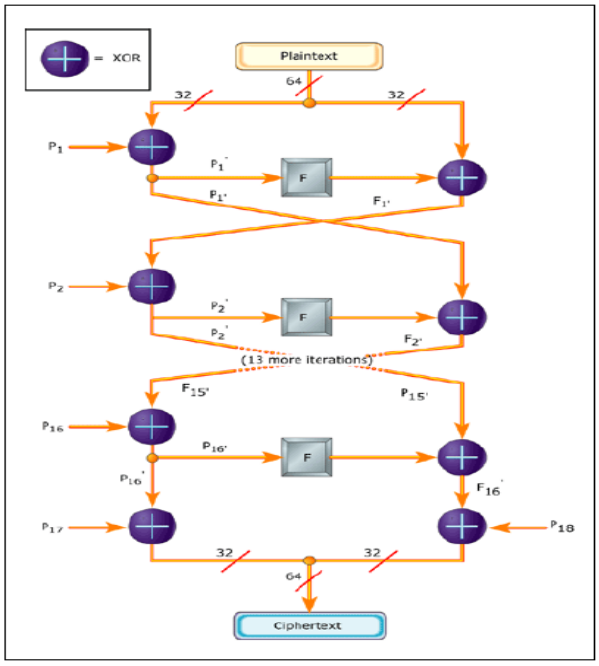

Fig 1: Working of proposed method

A. Modified F-Function:

Function F plays an important role in the algorithm, and we decided to modify function F. Original function F is defined as follows. [3]

F(X) = ((S1 + S2 mod 232) XOR S3) + S4 mod 232

Instead, we modified the F-Function by replacing 2 addition operations as XOR Operations . Thus the modified F-

Function is written as,

F(X) = ((S1 XOR S2 mod 232) + (S3 XOR S4mod 232))

This modification leads to the simultaneous execution of two XOR operations. In the case of original F-function which executes in sequential order and it requires 32 Addition operations and 16 XOR operations. But in the case of our modified F-function it requires the same 48 gate operations (32-XOR,16-addition) but time taken to execute these 48 operations will be reduced because of multithreading [2].

ALGORITHM STEPS

Divide X into two 64 -bit halves XL and XR

For i=1 to 32:

XL = XL⊕ Pi

XR = F (XL)⊕ XR

Swap XL and XR

End for

Swap XL and XR (Undo the last swap.)

XR = XR⊕ P17

XL = XL⊕ P1

Recombine XL and

Output X (64-bit data block: cipher text)

For decryption, the same process is applied, ex-cept that the sub-keys Pi must be supplied in re-verse order. The nature of the Feistel network ensures that every half is swapped for the next round (except, here, for the last two sub-keys P17 and P18).

B. Simulation and Results:

It is having a function to iterate 16 times of network. Each round consists of key-dependent permutation and a key and data-dependent substitution. All operations are XORs and additions on 32-bit words. The only additional operations are four indexed array data lookup tables for each round.

Algorithm	Time (5 bytes)	Time (410 bytes)
Std Blowfish Algorithm	Encrypt: 0.572 ms Decrypt: 0.666 ms	Encrypt: 2.44 ms Decrypt: 6.35 ms
Custom Blowfish Algorithm	Encrypt: 0.302 ms Decrypt: 0.168 ms	Encrypt: 2.26 ms Decrypt: 3.20 ms
Difference	Encrypt: 0.270 ms Decrypt: 0.498 ms	Encrypt : 0.18 ms Decrypt: 3.15 ms

Figure 1 Response time comparison in milliseconds

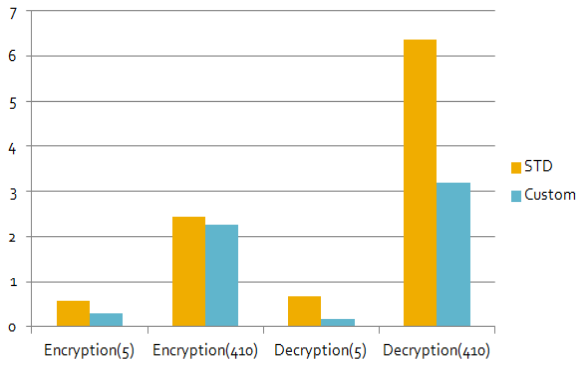

Figure 2 Comparison between standard and customized blowfish algorithm

For decryption purpose, the same process is applied. The only change is that, sub keys Pi must be supplied in reverse order. The nature of the Feiestelnet work [1] assures that every half cycle is exchanged for the upcoming next round (last two sub-keysP17 and P18 are the exceptions for this case).

IV. SIGNIFICAN FEATURES

This proposal has several merits to be appreciated.

Encryption Function F(x) recombining and re iterating modules:

1.Its simpler design & easy in implementing it.

2.The Design and working is fashioned in such a way that, it is infeasible to breach.

3.Next, On considering the Encryption & Decryption Phase,

a. The execution time of Blowfish algorithm is approximately reduced up to 13.5% on comparing with the original Blowfish Algorithm.

b. Although, we used 2-XOR gates and 1- ADDER but the original F-function uses 2-ADDERs and 1-XOR gate and there is no abrupt change in the execution time or clock cycles required for execution.

This is because all fundamental logical operations like AND,OR,XOR takes more or less equal time when running under any programming languages since those languages are logically driven.

It's quite hard for the eavesdroppers to realize that the F-function is modified and hence probability of attack is less on comparing with the original Blowfish algorithm.

V. CONCLUSION

This paper will satisfy our foremost aim of providing a system which is "infeasible to get breached". It also provides a high end data security when transmitting over any insecure medium. Intruders will not have any idea about our modification both in terms of algorithm as well as in our design, so breaching this system is highly impossible. We are sure that this software tool is unique of its kind and it can also be tuned in terms of higher performance and security in near future by adding or replacing cryptographic part because of its modularity in design. That is, it has a good performance without compromising the security and the modified F-function also enhances the performance by reducing the clock cycles upto 33% and reduces the execution time upto 14%

Proposed system can achieve efficient data encryption up to 4 bits per clock. In this design, we avoid I/O limited constraint by modifying the I/O from 64 bits to 16 bits. Cryptography using Blowfish algorithm requires less memory and provides high speed data encryption. Respectively, it can be applied to various devices. This proposed system uses variable-length key block cipher. It is suitable for applications where the key do not change often, like a communications link. It is faster than DES. Blowfish is a 16 pass block encryption algorithm that is never broken.

Blowfish algorithm is repeatedly tested on devices respectively and it is found to be secure. It provides high speed data encryption because it takes advantage of built-in instructions on the current microprocessors for basic bit shuffling operations.

ACKNOWLEDGMENT

First and foremost, I would like to thank my guide, Prof. M. B. Ansari, for his guidance and support. I will forever remain grateful for the constant support and guidance extended by guide, in making this paper. Through our many discussions, he helped me to form and solidify ideas. The invaluable discussions I had with him, the penetrating questions he has put to me and the constant motivation, has all led to the development of this project.

REFERENCES

1. Bruce Schneier, “Applied Cryptography”, John Wiley & Sons, Inc. 1996
2. T. Morkel, J.H.P. Eloff, M.S. Olivier, “An overview of Cryptography”, (ICSA), 2004.
3. The homepage of description of a new variable-length key, 64-bit block cipher http://www.counterpane.com/bfsverlag.html.
4. Ms Neha Khatri – Valmik, Prof. V. K Kshirsagar, “Blowfish Algorithm”, IOSR Journal of Computer Engineering (IOSR-JCE), Volume 16, Issue 2, Ver. X (Mar-Apr. 2014), PP 80-83.
5. Patterson and Hennessy, “Computer Organization & Design: The Hardware/ Software Interface”, Morgan Kaufmann, Inc. 1994.
6. P. Karthigai Kumar and K. Baskaran. 2010. An ASIC implementation of low power and high throughput blowfish crypto algorithm Microelectron. J. 41, 6 (June 2010), 347-355.
7. B. Schneier, "Description of a New Variable-Length Key, 64-bit Block Cipher (Blowfish)," Fast Software Encryption: Second International Workshop, Leuven, Belgium, December 1994, Proceedings, Springer-Verlag, 1994, pp.191-204.
8. TingyuanNie; Chuanwang Song; XulongZhi, "Performance Evaluation of DES and Blowfish Algorithms," Biomedical Engi-neering and Computer Science (ICBECS), 2010 International Conference on, vol., no., pp.1-4, 23- 25 April 2010.
9. S. Vaudenay, "On the Weak Keys in Blowsh," Fast Software Encryption, Third International Workshop Proceedings, Springer-Verlag, 1996, pp. 27-32.
10. B. Schneier, Applied Cryptography: Protocols, Algorithms, and Source Code in C, 2nd ed., John Wiley & Sons, 1995J. Jones. (1991, May 10). Networks (2nd ed.) [Online]. Available: http://www.atm.com
11. Manikandan Ganesan, Krishnan Ganesan, “A Novel Approach to the Performance and Security Enhancement Using Blowfish Algorithm”, International journal of Advanced Research in Computer Science, 2011
12. Kishnamurthy G.N, Dr. V. Ramaswamy and Mrs. Leela.G.H ,“Performance Enhancement of Blowfish algorithm by modifying its function” Proceedings of International Conference on Computers, Information, System Sciences and Engineering 2006, University of Bridgeport, Bridgeport, CT, USA. pp. 240-244.
13. William Stallings, Cryptography and Network Security, 3rd Ed, Wiley, 1995.
14. B. Schneier, “Description of a New Variable-Length Key, 64-Bit Block Cipher (Blowfish)”, Fast Software Encryption, Cambridge Security Workshop proceedings (December 1993), Springer-Verlag, 1994, pp. 191-204.
15. Dr.V. Ramaswamy, Kishnamurthy. G.N, Mrs. Leela. G.H, Ashalatha M.E, “Performance enhancement of
16. CAST –128 Algorithm by modifying its function” Proceedings of International Conference on Computers, Information, System Sciences and Engineering 2007, University of Bridgeport, Bridgeport, CT, USA.

AUTHORS PROFILE

Rakhee Single, I have completed BE in Computer Science and Engineering in 2011 from Jawaharlal Neharu Engineering College, Aurangabad. Two years I worked as a Assistant Professor in Jawaharlal Neharu Engineering College, Aurangabad (2011-2013) I am doing ME in Computer Science and Engineering (2013-15) from Shreeyash College of Engineering and Technology, Aurangabad. I also worked for one year as a Software Engineer in BNT SOFT PRIVATE LTD