International Journal on Recent and Innovation Trends in Computing and Communication ISSN: 2321-8169
Volume: 4 Issue: 6 98 - 102

Resilient security against hackers using enchanced encryption techniques: Blowfish and Honey Encryption

Prof. Rohini S. More
Department of Computer Science and Engineering
A.G. Patil Institute Of Technology
Solapur, India
e-mail: rohinimore0808@gmail.com

Prof. Smita S. Konda
Department of Computer Science and Engineering
A.G. Patil Institute Of Technology
Solapur, India
e-mail: smitakonda@gmail.com

***Abstract*—** Data security refers to protective digital privacy measures that are applied to prevent unauthorized access to computers, databases and websites. Data security also protects data from corruption. Data security is the main priority for organizations of every size and genre. Data security is also known as information security (IS) or computer security. So in this paper to provide security to our data we use encryption. The primary purpose of encryption is to protect the confidentiality of digital data stored on computer systems or transmitted via the Internet or other computer networks. To achieve encryption we are using two advanced encryption techniques: blowfish and honey encryption.

Keywords*-Blowfish encryption, Honey encryption, DTE, Cryptography, Key expansion*

*****

I. INTRODUCTION

An encryption algorithm plays a vital role to protect the information during storage or transfer. The encryption algorithms are of two types: Symmetric (secret) and Asymmetric (public) keys encryption. In Symmetric key encryption or secret key encryption, only 1 key is used for encryption as well as for decryption of information ,Eg: Blowfish Encryption algorithm Triple DES, Advanced Encryption Standard(AES) and Data encryption standard(DES). Asymmetric key encryption or public key encryption uses two keys, one for encryption and other for decryption Eg: RSA.

In this paper for providing high security to data we are using two encryption algorithms: 1) Blowfish Encryption Algorithm. 2) Honey Encryption Algorithm. Blowfish is a symmetric key block cipher. It takes a variable-length keys for encryption as well as decryption, ranges from 32 bits to 448 bits, to make it ideal for both domestic and exportable use. Blowfish was invented in 1993 by Bruce Schneier as a free, fast substitute to existing encryption algorithms. Since then it has been analyzed significantly, and it is slowly gaining recognition as a well-built encryption algorithm. Blowfish is unpatented and license-free, and is available free for all uses. Blowfish provides a high-quality encryption rate in software and no efficient cryptanalysis of it has been found to date. Schneier considered Blowfish as a general-purpose algorithm, planned as an substitute to the aging DES. Blowfish is known for both its tremendous speed and overall effectiveness as many claim that it has never been defeated. Blowfish can be found in software categories ranging from e-commerce platforms for securing payments to password management tools, where it used to protect passwords. It's definitely one of the more flexible encryption methods available.

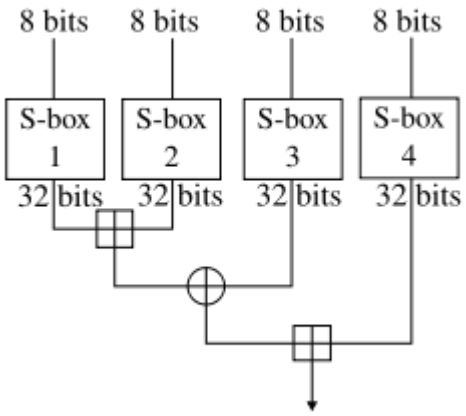

Figure No.1 The round function (Feistel function) of Blowfish

Honey encryption (HE) is a simple approach to encrypting messages using low min-entropy keys such as passwords. HE is designed to produce a ciphertext which, when decrypted with any of a number of incorrect keys, yields plausible-looking but bogus plaintexts called honey messages[8]. There are Honey encryption schemes for password-based encryption of RSA secret keys and credit card numbers. Honey Encryption turns every wrong password guess made by a hacker into a confusing dead-end. When an application or user enters and sends a password key to access an encrypted database or file, as long as the password is correct, the data is decrypted and accessible in its original, and readable, format. If the password key is incorrect the data will continue to be unreadable and encrypted. Hackers who steal databases of user logins and passwords only have to guess a single correct password in order to get access to the data. The way they know they have the correct password is when the database or file becomes readable. To speed up the process, hackers have access to sophisticated software that can send thousands of passwords each minute to applications in an attempt to decrypt the data. Using higher speed, multi-core processors also shortens the time it can take to break encryption. With Honey

Encryption, decrypting with an incorrect password results attempt. For example, if a hacker made 100 password attempts, they would receive 100 plain text results. Even if one of the passwords were correct, the real data would be indistinguishable from the fake data.

II. RELATED WORK

Nirvan Tyagi, Jessica Wang, Kevin Wen, Daniel Zuo[2]., described in this paper about implementing message spaces with the proper API. Various required functions can be made for each specific message space.

- Cumulative distr(message) - takes in a message and outputs cumulative probability representing where in ordered message space the message lies
- Probability distr(message) - takes in a message and outputs probability of that single message
- Next message(message) - takes in a message and outputs the next message in ordered message space
- Get inverse cumul distr samples() - returns list of pre-calculated sampling of cumulative distribution values to messages

Vinayak P P, Nahala M A[1]., proposed new encryption technique in which an attacker tries to different keys different decrypted data will be got, thus the brute forcer will be confused totally about the actual content. Also new message encoding scheme called a distribution transforming encoder (DTE). Honey encryption represents the space of original message using DTE.

Ari Juels, Thomas Ristenpart[4]., suggested that use of Low-entropy keys such as passwords that helps resist brute-force attacks that try every key; Low-entropy secrets such as passwords are likely to persist in computer systems for many years. Their use in encryption leaves resources vulnerable to offline attack Gadkar Prathamesh S., Gawali Sanket D., Khalkar Yogeshwar D.[5], Narode Aniket K. described cryptographic system to ensure the protection to the a variety of applications . They will propose a system to strongly transmit provenance for sensor data. They will introduce efficient technique for provenance data verification. They will design the new system for secure encryption & decryption. For that purpose privacy, validation, reliability & accessibility are the basic security requirements of wireless sensor network (WSN). The security threats in WSN are broadly classified as 1. Inside attacks versus outside attacks 2. Active attacks versus passive attacks sensor class or laptop class attacks. To achieve this, they have proposed various techniques: 1. Data Encryption Standard (DES) 2. Triple Data Encryption Standard (Triple DES) 3. Honey Encryption

Pratap Chnadra Mandal [6] has done a fair comparison between four most common and used symmetric key algorithms: DES, 3DES, AES and Blowfish. The comparison has been done on the basis of various parameters like rounds block size, key size, and encryption / decryption time, CPU process time in the form of throughput and power utilization. So he analyzed that blowfish is better than other algorithms. To achieve his goal in fake, but realistic looking data for every incorrect password he used Cryptography scheme. The main purpose of cryptography is privacy of data, non alteration of data. There are several goals of cryptography: 1. Confidentiality: Data in computer which is transmitted and should be accessed only by the authorized person and not by anyone else. 2. Authentication: The data received by any system has to verify the identity of the sender that whether the information is arriving from an authorized person or not. 3. Integrity: Only the authorized person is permitted to alter the transmitted information. **4.** Non Repudiation: Ensures that neither the sender, nor the receiver of message should be able to deny the transmission. 5. Access Control: Only the authorized person is able to access the given information. Compared algorithms are RC2, DES, 3DES, AES, RC6 & blowfish. In this paper finally he concluded that blowfish is best of all. In future he can perform same experiments on image, audio & video and developing a stronger encryption algorithm with high speed and least energy utilization.

Ankita Deshpande, P.S.Choudhary [7] described blowfish is not only safe, but also quick and suitable for different platforms, therefore, it has high value of application in the field of information security. One should find it important that the maximum key size was used, and the key was chosen at random from the full key space of size 2448, since maximum key length is 448 bits. Blowfish has 3 parts 1. Encryption algorithm 2. Key-expansion 3. Decryption algorithm. In future a variety of encryption algorithms will be used to accommodate the wireless network applications & can be optimized further in future. In addition, a stronger encryption algorithm with high speed and least amount of energy utilization can be developed to get enhanced security and the performance evaluation parameters can be optimised.

Ms NehaKhatri – Valmik, Prof. V. K Kshirsagar [8]., discussed Blowfish algorithm and concluded that it is faster than DES. Blowfish is provides stronger security and also placed in public domains.

Jawahar Thakur, Nagesh Kumar [9]., discussed comparison between three most common symmetric key cryptography algorithms: DES, AES, and Blowfish. The comparison is made on the basis of these parameters: speed, block size, and key size.

III. BLOWFISH ALGORITHM

The algorithm uses variable key size encryption algorithm. As it is symmetric encryption only one key is used for encryption as well as decryption. One advantage is that it can use different key size upto the length of 448 bits. Blowfish is a symmetric block encryption algorithm designed with,

Fast: 32-bit microprocessors and rate of 26 clock cycles per byte.
Compact: runs in less than 5K of memory.

Simple: for addition, XOR, lookup table with 32-bit operands.

Secure: variable key length, it can be in the range of 32~448 bits: default 128 bits key length.

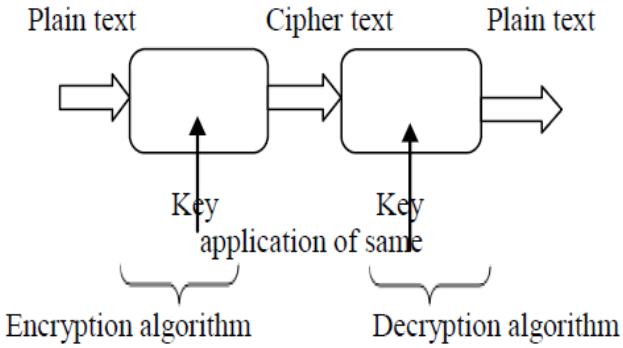

Figure No.2 Symmetric encryption/decryption process for Blowfish Algorithm [7]

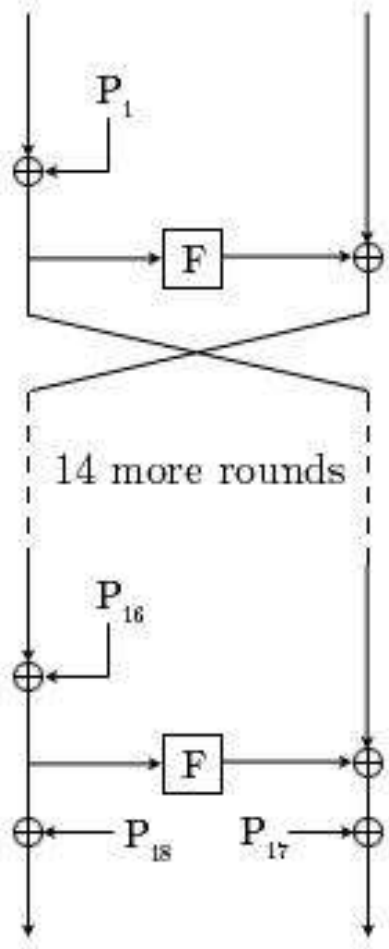

Figure No.3 The Fiestel structure of Blowfish. [3]

Divide x into two 32-bit halves: xL, xR
For i = 1 to 16:
xL = XL XOR Pi
xR = F(XL) XOR xR
Swap XL and xR
Swap XL and xR (Undo the last swap.)
xR = xR XOR P17
xL = xL XOR P18
Recombine xL and xR

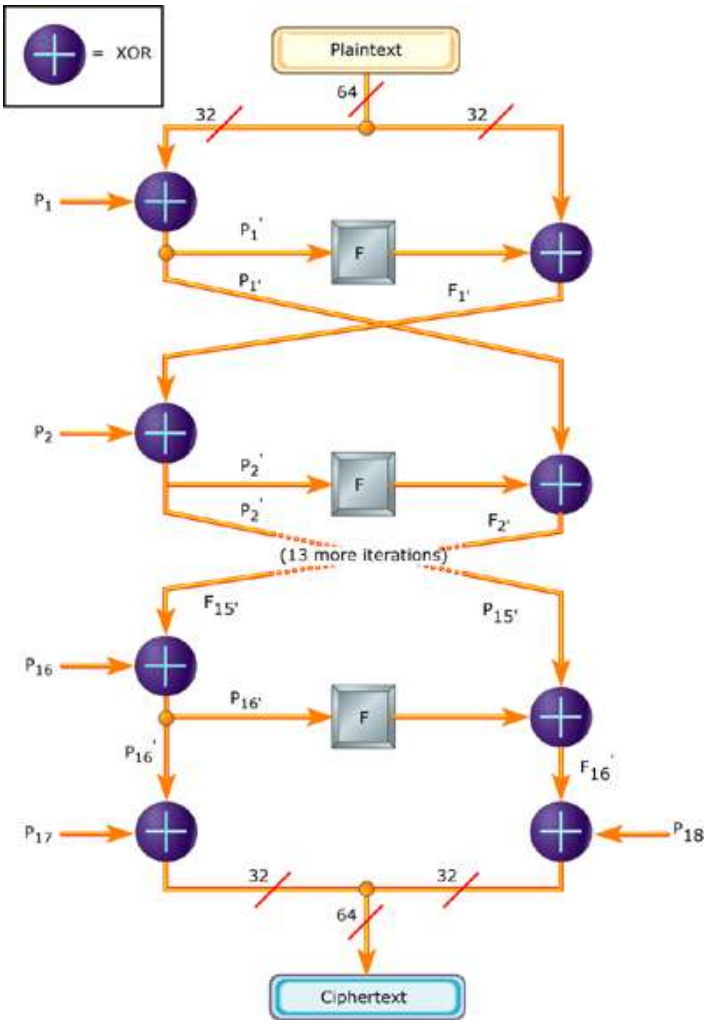

Figure No.4 Blowfish Encryption [8].

The structure of Blowfish is divided into two parts: data-expansion and key encryption. In the key expansion phase, the input key is converted into several sub key arrays total 4168 bytes. There are P array and S-boxes in the design of Blowfish. There are S-boxes, which are four 32-bit arrays with 256 entries each and P array is eighteen 32-bit boxes. First the string initialization process is done by the first 32 bits of the key are XORed with P1. For decryption, the same process is repeated, except that the sub-keys Pi must be supplied in reverse order. The Fiestel network ensures that every half is swapped for the next round [14].

There are three parts of Blowfish: Encryption algorithm, Key-expansion and Decryption algorithm.

In Encryption algorithm, encryption is done by taking the first 32 bits of the key and XOR with the first 32-bit box in the P-array. Then the next 32 bits of the key are XORed with next 32-bit box in the P-array., and so on, until all 448, or fewer, key bits have been XOR. This cycle is completed by returning to the beginning of the key, until the entire P-array has been XOR with the key.

In Key-expansion, a simple cipher, where same key is used. Exclusive-OR the key with the plaintext. Then it is reversed by another exclusive-OR of the same key with the cipher text. In the case of the Blowfish, there are a number of rounds, needing the key, so the actual key size is 64 bytes.
In Decryption algorithm, the same procedure is repeated as encryption only the thing that the P-arrays are used in reverse order.

IV. HONEY ENCRYPTION

Ari Juels and Thomas Ristenpart of the University of Wisconsin, the developers of the encryption system, presented a paper on Honey Encryption at the 2014 Euro crypt cryptography conference. Honey encryption provides a high security against brute force attacks for certain messages. The honey encryption provides a cipher text C that decrypts under any key or password that is provided by the attacker and it will be reasonable or believable looking message .The forged key or password will get a fake message when trying to decrypt the data that looks like legal to an attacker as it is the actual message. The internal structure of the honey encryption includes a specialized encoding schema and a normal encryption schema[2]. Honey Encryption is a type of data encryption that produces a ciphertext, which, when decrypted with an incorrect key as guessed by the attacker, presents a plausible-looking yet incorrect plaintext password or encryption key.

A. Method of protection

A brute-force attack involves frequent decryption with random keys. This is corresponding to selecting random plaintexts from the all possible plaintexts with a uniform distribution. This is efficient because even though the attacker is evenly probable to see any given plaintext, most plaintexts are very implausible to be genuine i.e. the distribution of valid plaintexts is non-uniform. Honey Encryption defeats such attacks by first transforming the plaintext into a space such that the distribution of legitimate plaintexts is uniform. Thus an attacker guessing keys will see legitimate-looking plaintexts regularly and random-looking plaintexts rarely. This makes it hard to determine when the right key has been guessed. In effect, Honey Encryption provides false data in response to every false guess of the password or encryption key. Honey Encryption can defend against these attacks by first mapping credit card numbers to a larger space where they match their probability of authenticity. Numbers with invalid IINs and checksums are not mapped at all (i.e. have probability 0 of authenticity). Numbers from large brands like MasterCard and Visa map to big regions of this space, while less popular brands map to minor regions, etc. An attacker brute-forcing such an encryption scheme would only see legitimate-looking credit card numbers when they brute-force, and the numbers would come out with the frequency the attacker would expect from the real world.

B. Honey Encryption Scheme Set-up

We now describe the original honey encryption scheme proposed by Juels and Ristenpart. In this construction, we have a message space M which consists of all possible messages. We map these messages to a seed space S through the use of a distribution-transforming encoder (DTE). The seed space is simply the space of all n-bit binary strings for some predetermined n. Each message in m Є M is mapped to a seed range in S. The size of the seed range of m is directly proportional to how probable m is in the message space M. We need some knowledge about the message space M in order for the DTE to map messages to seed ranges; particularly the DTE requires the cumulative distribution function (CDF) of M and some information on the ordering of messages. Furthermore, the seed space has to be large enough so that even the message with least likelihood in the message space is assigned at least one seed. With this information, we can get the cumulative likelihood range corresponding the message m and map it to the same percentile seed range in S. We illustrate the encryption process below with a basic example.

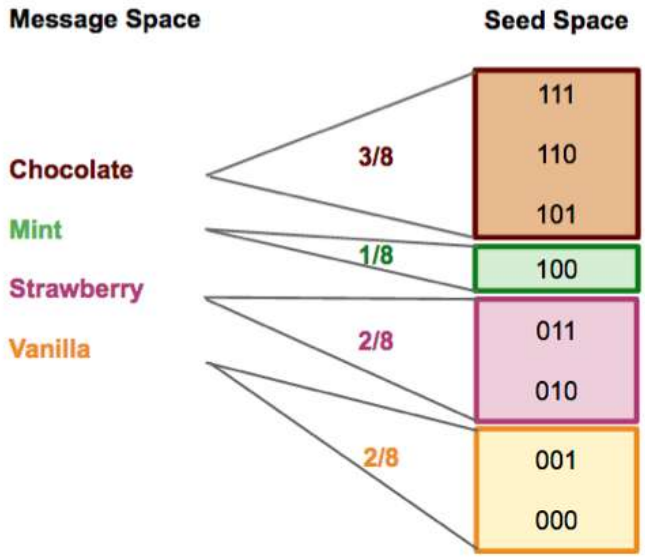

Figure No.5 Encryption process [2].

Consider the simple example of encoding ice cream flavors in figure 4. Our message space M consists of various flavors, M = {chocolate, mint, strawberry, vanilla}. Through facts of some population's first choice of ice cream flavors, probabilities are assigned to each flavor. Consider a seed space S of 3-bit strings. With these probabilities, we can then map each flavor to a seed range. In this case, the flavors are ordered alphabetically. Note that this is an arbitrary ordering and a different ordering would lead to a different mapping of seeds. Now consider the process of encrypting a message, say chocolate. Using the DTE, we randomly select a seed in the corresponding seed range. This seed is XORed with a shared secret key to generate the cipher text. Decryption is slightly difficult. The cipher text is XORed with the secret key is return the seed. We know that the seed falls into a seed range that corresponds to a message's CDF value. Here we run into a problem. For most message spaces, the CDF is one-way and we cannot go back to a message given a value in its CDF range. Instead, we use an inverse sampling table. Using a precalculated table of sampled CDF values to messages from the message space,

we can run binary search on the inverse sampling table and then linear scan the message space from there [2].

V. ADVANTAGES OF TWO TECHNIQUES

Blowfish is the most stronger and fast encryption algorithm. It helps in providing high-speed data encryption that can be applied to various I/O devices. It is most secure algorithm. The design of S-boxes and P-boxes make the structure of algorithm more powerful.

Honey encryption provides huge security against low entropy to resist brute force attacks. Honey encryption is also more powerful algorithm that works on keys.

VI. DISADVANTAGES OF TWO TECHNIQUES

Blowfish algorithm uses same key for two parties. This is unique pair of keys given to each pair of users. So key management is major problem. This algorithm doesn't provide non-repudiation and authentication services. This algorithm also leads to time consumption.

Honey encryption fails to provide security when the opponent has some side information about the message.

VII. CONCLUSION

Honey Encryption and Blowfish algorithms are a new, innovative approach to defending against data theft and brute forcing passwords. However it is not easy to generate fake data for all possible real cases. But that challenge can be overcome by using real data that are publicized across the Internet during several data breaches.

VIII. FUTURE WORK

In future we will try to work on the drawbacks of both the algorithms so that a problem like key management is carried on smoothly as well as provide all security services. Implementation can also be done by combining both the algorithms to generate stronger algorithm that overcomes these drawbacks.

REFERENCES

[1] Vinayak P P, Nahala M A, "Avoiding Brute Force attack in MANET using Honey Encryption," ISSN (ONLINE): 2319-7064

[2] Nirvan Tyagi [ntyagi], Jessica Wang [jzwang], Kevin Wen [kevinwen], Daniel Zuo [dzuo], "Honey Encryption Applications," 6.857 Computer & Network Security, 13 May 2015.

[3] http://iitd.vlab.co.in/?sub=66&brch=184&sim=1147&cnt=1

[4] Ari Juels, Thomas Ristenpart, "Honey Encryption: Security Beyond the Brute-Force Bound," January 29, 2014

[5] Gadkar Prathamesh S., Gawali Sanket D., Khalkar Yogeshwar D., Narode Aniket K., "Secure Data Transmission in WSN using 3DES with Honey Encryption," Vol-1 Issue-4 2015, IJARIIE-ISSN(O)-2395-4396.

[6] Pratap Chnadra Mandal, "Superiority of Blowfish Algorithm," Volume 2, Issue 9, September 2012, ISSN :2277 128X.

[7] Ankita Deshpande, P.S.Choudhary, "FPGA Implementation of Blowfish Cryptographic Algorithm," Volume 4, Issue 4, April 2014, ISSN:2277 128X

[8] Ms.NehaKhatri–Valmik, Prof. V. K Kshirsagar, "Blowfish Algorithm," Volume 16, Issue 2, Ver. X (Mar-Apr. 2014), e-ISSN:2278-0661,p-ISSN:2278-8727