ISSN (Print) : 0974-6846
ISSN (Online) : 0974-5645
Indian Journal of Science and Technology, *Vol 12(39), DOI: 10.17485/ijst/2019/v12i39/145570, October 2019*

Towards the Proposal of Mobile Security Encryption Algorithm: "RHINO256"

Kaustav Mani Pathak*, Priyanka Jain, Swati Yadav and Veena Tayal

Department of Computer Science & Engineering, MRIIRS, Faridabad, India

Abstract

Background/objectives: Encryption is one of the most important factors of any security structure. Data cannot be protected without an efficient encryption technique. Different encryption algorithms are available nowadays that apply techniques such as encrypting using symmetric or asymmetric keys, using stream or block ciphers, and others such as Feistel networks, substitution, and permutation, etc. **Methods/statistical analysis:** We have analysed the existing algorithms so as to get the problem. **Findings:** The proposed algorithm will solve the problem of maintaining the authenticity of the sender as another third party cannot send a message in the former's name, as was the problem with earlier algorithms such as DES. **Improvements/applications:** The main idea of performing this research was to propose an algorithm that can encrypt a message with the sender's private key and the receiver's public key together and the same can be decrypted using the receiver's private key and sender's public key.

Keywords: S-box, P-box, Feistel Networks, DES

1. Introduction

Mobiles are one of the most used devices of the 21st century and are expected to reach 5 billion by 2019. Every year, millions and millions of mobile devices are sold in the market. The cheaper rates of cell phones and easier programming tools make mobile phones as one of the easiest and most common targets for attackers. Hence with the rise of mobile devices comes the concern for its security. Mobile security can be defined as the protection of electronic devices such as smart phones, tablets, and other handheld portable devices. Mobile security is one of the most important concerns of the hour as almost all of the digital works can be performed through mobiles be it as simple as receiving and making calls to complex and sensitive works such as banking and corporate tasks.

One of the earliest methods of securing and hiding of data is encryption. Encryption hides the data by making it unreadable by using special techniques. It helps to keep the real data secret even if an unauthorized person has access to the data. Encryption is one of the most important parts of any security system. Nowadays, almost all of the modern system encrypts the data whether it is in rest, in motion, or in use. A good encryption system should provide Authenticity, Integrity, and Non repudiation.

In this study, considering the wide popularity of android Smartphones, we have proposed an encryption algorithm that can be used in messaging app to encrypt the data in such a way that a middle man can neither access, nor manipulate nor can he/she sent a fraudulent message in the name of an authentication user. The app shall encrypt messages based on the algorithm designed by us. We will be using a combination of Feistel networks along with a certain number of S-boxes and P-boxes to increase confusion. There will be two keys used in this system to encrypt or decrypt messages.

**Author for correspondence*

S-boxes or substitution boxes are components of a security program that performs substitution and are generally used to hide the relation between the key and the cipher text. Here we will be using S-boxes (with inputs and outputs ranging from 128,64,32,16 and 8 bits).[1]

P-boxes or permutation boxes are components of a security system that permute or transpose the bits given as input in order to increase confusion. Here we will be using a Compression P-box that when fed with a certain length of bits always gives the output as 128 bits.

A Feistel Network also known as a Feistel Cipher is a symmetric structure that is used to construct block ciphers. It is named after the German-born cryptographer Horst Feistel and has been used in many popular algorithms such as DES. The most important part of any Feistel network is the round function F. The main security depends on how strong the F function is made.[2]

The main base for this research came from the study of the DES algorithm. Although our algorithm is very different than that of DES but the Feistel function is quite similar and we have taken the idea of using S-boxes and P-boxes to increase confusion. Again DES uses only one key and can encrypt only 64 bits of data while our algorithm encrypts 256 bits of data using two keys of 128 bits each. The Feistel network of DES had only two parts left and right of 32 bit each where the right side was fed into a round function with the key and the result was than xor with the left side and then the two parts were swapped. The rounds were repeated 16 times with the last round not having the swapping function so as to use the same layout for both encryption and decryption. Whereas our algorithm encryptions data by dividing each part into two child parts after each round and applying feistel on these and after a certain number of splitting and encrypting the parts are then merged and encrypted after each round to get the output as 256 bits. This allows us to use the same layout as many times we want both for encrypting and decrypting just by choosing[3,4] the correct Feistel for encryption or decryption.

DES has the following disadvantages which are solved in our algorithm:

1. DES has smaller block size (64 bit). While our algorithm can encrypt 256 bits.
2. S-box of DES creates same output with two chosen input while the S-box in our algorithm is more specific.
3. In DES, the purpose of initial and final permutation is not clear while RHINO256 has no initial or final permutation.
4. The key length of DES is 56 bits while our algorithm has 128 bits.
5. DES is less secure compared to RHINO256 as DES uses only one key to encrypt and decrypt while our algorithm uses two different keys to encrypt and to decrypt.

2. Proposed Algorithm – (RHINO256)

The main idea of performing this research was to develop an algorithm that can encrypt a message with the sender's private key and the receiver's public key together and the same can be decrypted using the receiver's private key and sender's public key. This will solve the problem of maintaining the authenticity of the sender as another third party cannot send a message in the former's name, as was the problem with earlier algorithms such as DES.

Here we will be using a balanced Feistel network. The round function F is given as: **F(R0, K)** => Ro (n bits) is shown in Figure 1.

As in Figure 2 on the left, the network layout consists of two inputs parts; L0 and R0, of equal length (say n bits). R0 will be fed to the function F along with the key K (n bits). Here, n can be 128, 64, 32, 16, or 8 bits. The output of the F function is then XOR ed with L0 and the values are then swapped to form L1 and R1. Inside the F function, the input R0 is first fed into an S-box and then XOR ed with the key K. The value is then fed into another S-box

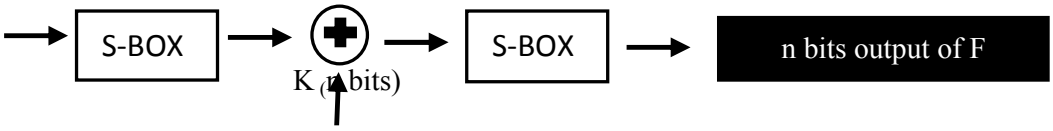

Figure 1. Round function F procedure.

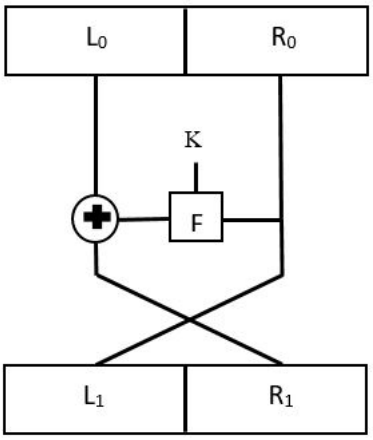

Figure 2. RHINO256 encryption algorithm process.

to finally have the output. The S-boxes help to increase the confusion and the overall strength of the F function.

The main encryption process consists of 10 rounds each with varying no. of Feistel networks. The inputs bits and the key length will also be different in each individual network.

Rounds 1 & 10 will have only one Feistel network each with L0 = R0 = K = 128 bits. The outputs L1 = R1 = 128 bits.

Rounds 2 & 9 will have two Feistel networks each with L0 = R0 = K = 64 bits. The outputs L1 = R1 = 128 bits.

Rounds 3 & 8 will have four Feistel networks each with L0 = R0 = K = 32 bits. The outputs L1 = R1 = 32 bits.

Rounds 4 & 7 will have eight Feistel networks each with L0 = R0 = K = 16 bits. The outputs L1 = R1 = 16 bits.

Rounds 5 & 6 will have sixteen Feistel networks each with L0 = R0 = K = 8 bits. The outputs L1 = R1 = 8 bits.

The keys used in the Feistel networks will always be different from each other. The Round Key Generator will be used to input a different key of the required length to each feistel network. Hence in this algorithm, the plain text is basically divided into two parts then encrypted with a key then the two parts are again divided into two parts each and so on.

Here, the plain text will be converted into 256 bit blocks. The block will then be split into two 128 bit blocks forming one feistel structure and encrypted using a key of 128 bits. The two blocks will then be split into four blocks and the process will be repeated till the block size becomes 8 bits forming sixteen Feistel networks. The 8 bit blocks will be encrypted twice and then joined to form 16 bit blocks which are then encrypted and joined to form 32 bit blocks. The whole process will continue till we have an output block of 256 bits. The whole structure of the algorithm is made symmetric so that the same can be used in reverse for decryption of the cipher text and the same is shown in Figure 3.

3. Encryption and Decryption Mechanism

In our app, we will be encrypting the sender's message in plain text using two keys – the first key will be the private key of the sender and the second key will be the public key of the receiver. So the whole encryption process as described above will be repeated twice. Decryption process will be similar to encryption as the receiver will use their private key first and then the public key of the sender in order to get the plain text. This method helps in preventing fraudulent5 messages sent by an attacker to the receiver in the name of the authorized sender. The encryption/decryption mechanism is shown in Figure 4.

The two keys Public, P1 and Private, P2 will be generated for each user according to a Password, P entered by the user. This password will be different from the app password and will be stored in the app itself.

P (128 bits)

S1 (128 bits) = Salt 1 which is a random value of 128 bits used to increase security against brute force attacks.

S2 (128 bits) = Salt 2

C = Count is the no. of times to which salts are to be added to the password in order to get the public or the private key.

for (i = 0; i < C ; i ++)

{ P1 = P | | S1 }

At the end of the loop, P1 will fed into a compression P-box to get the required public key P1 of 128 bits.

Now, we run the loop again

for (i = 0; i < C ; i ++)

{ P2 = P1 | | S2 }

At the end of this loop, P2 will fed into a compression P-box to get the required private key P2 of 128 bits.

Hence one can generate the private key from the public key or both the keys from the password, if they have access

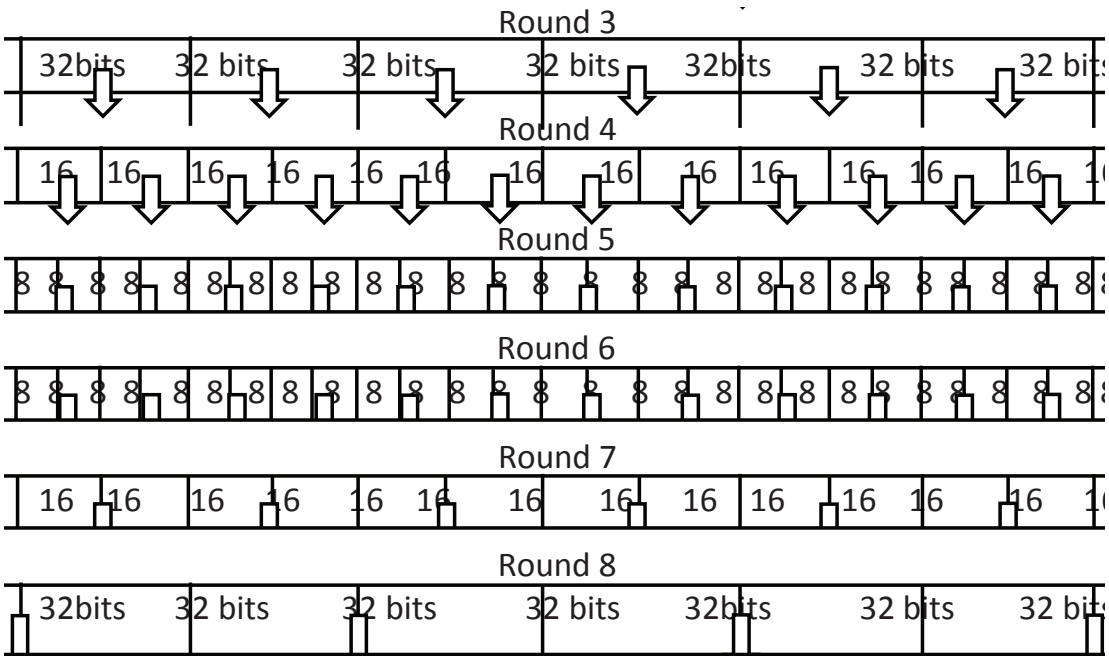

Figure 3. Proposed algorithm.

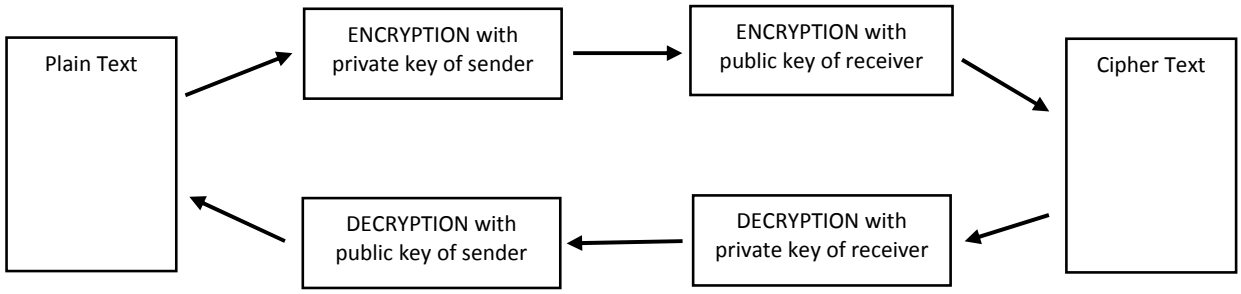

Figure 4. Encryption/decryption mechanism.

to any one of them, provided they know the values of S1, S2, and C which will be kept secret.

4. Round Key Generator

The main usage of this component in our algorithm is to provide the different round keys of variable lengths for the different Feistel structures used in the rounds. The generator will be fed with a main key say the private key P2 of 128 bits which will split into 16 parts of 8 bits each. In the whole encryption structure we will be dealing with 62 Feistel structures hence we will need 62 different keys. The two keys required for the rounds 1 and 10 will be the same as the main key. All the other keys required in the remaining rounds will be provided by combining the 16 parts in different ways according to a secret pattern. The pattern will be chosen in such a way that each bit of the text is encrypted with each bit of the key at least once.

5. Conclusion

RHINO256 is not just a thought but rather an algorithm that can be implemented in bigger projects for safer and more secured digital communications. Our algorithm can be used to replace existing symmetric and asymmetric algorithms for a hybrid system of encryption. Data leakage, sniffing, non-repudiation, hacking, etc. can be

prevented with the use of our algorithm. In the future, we will be increasing the block size and encoding shall be done in Unicode so as to make it encrypt texts written in different languages. Testing for the algorithm will be done in the future scopes of this research.

Acknowledgement

We would like to express our sincere gratitude to Research Co-Ordinators, Accendere CL Educate Ltd. for guiding and helping us complete the paper.

References

1. Zahid AH, Arshad MJ. An innovative design of substitution-boxes using cubic polynomial mapping. Symmetry. 2019;11(3):437.
2. Schneier B, Kelsey J. Unbalanced feistel networks and block cipher design. In: International workshop on fast software encryption. Springer, Berlin, Heidelberg; 1996. P. 121–44.
3. Preneel B, Rijmen V, Bosselaers A. Recent developments in the design of conventional cryptographic algorithms. In: State of the art in applied cryptography. Springer, Berlin, Heidelberg; 1998. P. 105–30.
4. Biryukov A, Leurent G, Perrin L. Cryptanalysis of feistel networks with secret round functions. In: International conference on selected areas in cryptography.Cham: Springer; 2015. P. 102–21.
5. Daemen J, Govaerts R, Vandewalle J. A new approach to block cipher design. In: International workshop on fast software encryption. Springer, Berlin, Heidelberg; 1993. P. 18–32.