ISSN 1512-1232

SOPHISTICATED SUBKEY GENERATION FOR SYMMETRIC ENCRYPTION

Krishna C Kommanapalli

Computer Science and Engineering Department, Vignan's inst. Of inf. Tech, duvvada, visakhapatnam, india
kkrishna.ai@gmail.com

Abstract

This paper presents a new way of Subkey generation for block symmetric encryption which operates on numbers rather than bits. Logarithmic approach is followed to generate Subkeys. Feistel network is used for encryption and operates on 128-bit data. It supports a key length up to 256-bits. To evaluate this approach I have developed a simulation. Test results indicate that this is a promising approach for symmetric encryption.

***Keywords**: Symmetric encryption, feistel network, subkey, ciphertext, cryptanalysis.*

I. INTRODUCTION

As the computational capabilities of the processors are burgeoning, conventional algorithms need to be updated or replaced with sophisticated algorithms to meet the needs of contemporary world of Information. By this we can infer that the antediluvian symmetric encryption algorithms should be replaced while preserving their essence. So there is need to improve the computational complexity of the algorithms like DES [1], which usually operates on 64-bit blocks and uses relatively small keys like 56-bit, 128-bit etc. A good cryptosystem is one in which all the security is inherent in knowledge of the key and none is inherent in the knowledge of algorithm. This is why key management is so important in cryptography.

Small keys are vulnerable to brutal force attacks. Large keys are difficult to remember, they need to be stored some where thus compromising security. So by taking both these things into consideration, a 256-bit key may optimize our needs. Similarly larger the size of the block greater the security is, but very large block size, takes more time for encryption and decryption. No block cipher is ideally suitable for all applications, even one offering a high level of security. This is a result of inevitable tradeoffs required in practical applications, including those arise from, for example, speed requirements and memory limitations (e.g., code size, data size, cache memory), constraints imposed by implementation platforms (e.g., hardware, software, chip cards), and different tolerances and of applications to properties of various modes of operations. In addition, efficiency must typically be traded off against security. Thus it is beneficial to have a number of candidate ciphers from which one to draw. In the last two decades cryptographers developed various key generation algorithms but are confined to operations like XOR, addition, shift etc. As Intel has launched its new core processor which exhibits high performance, there is need to replace these operations with some high level operations. So adding the strength of logarithms to the key generation makes it difficult for cryptanalysis. Logarithm gives unique values for all the 256 decimal values starting from one, when used e as its base. Using feistel network (described by Horst Feistel of IBM in 1973 [2] for symmetric block encryption) in encryption and decryption gives us an advantage that there is no attack recorded till now except exhaustive key search, also it is very easy to implement because we need not implement two different algorithms, one for encryption and one for decryption

ISSN 1512-1232

The algorithm discussed in this paper is a block symmetric encryption algorithm which uses the feistel network and extends the capabilities of the DES by using a 256-bit key and operating on 128-bit blocks. The main idea is to improve the computational complexity of the algorithm, at the same time making it easy for any programmer or user to understand and implement it in any language, because if the algorithm can be concisely and clearly explained, it is easier to analyse that algorithm for cryptanalytic vulnerabilities and therefore develop a higher level of assurance as to its strength. To make encryption and decryption fast, I used only X0R operations on the key and the message. For evaluating this approach, I have simulated it in c language, and the results are promising. It is suitable for applications where the key does not change often, like a communication link or an automatic file encryptor. This can be compatible with processors like 80486 to new Intel's core processors.

Overview: The remainder of this paper is structured as follows. In section II, I elaborate the procedure for subkey generation. Section III explains about implementing the encryption algorithm using feistel network. Section IV discusses about various potential attacks (theoretically) to this algorithm. Section V gives an example, section VI shows related work in this field and finally in section VII conclusions are drawn.

II. SUBKEY GENERATION

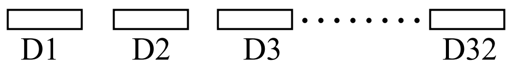
In this section, I explain the generation of Subkey from the given main key. The security of the algorithm rests in the key. I first explain the permutation of bits. Then I introduce the logarithmic approach of generation of Subkeys. We take variable length key as input. Where, $k_v \leq$ 256 bits.

If the key size is less than 256 bits, the bits are being to make it absolute. $k=k_v \| p$. Where p is usually the no. of 0's to make it a 256 bit key. Where $\|$ represents concatenation. So k represents the concatenation of k_v and p. Thus we get fixed key size for variable key.

Permutation

As we are padding bits, permutation should be done to make the key obscure for cryptanalysis.

We divide the key into blocks of 8bits. So we get,



Each bit in the block is represented by the index.ie., D1, x represents xth bit in D1 block from left to right.

1. Starting from the first block, select blocks with common difference between each block as 4.
2. So we take first bit from left to right from each block
   K1= D1, D5, D9, D13, D17, D21, D25, D29.
   K2= D2, D6, D10, D14, D18, D22, D26, D30.
   K3= D3, D7, D11, D15, D19, D23, D27, D31.
   K4= D4, D8, D12, D16, D20, D24, D28, D32.
3. Repeat for remaining 7 bits in the block.
4. So we get now the array of blocks from K1 to K32 each consisting of 8bits.

We can change the order based on the time factor and the number of encryptions to make cryptanalysis more difficult.

ISSN 1512-1232

Applying logarithm

As we got the permuted blocks, we now get the numerical value of each block as, N_i=decimal value of Ki, i=1to32. If N_i is 0 or 1, it is incremented with the no. of times it appears. The advantage of incrementing is, as we are padding bits it may be easy for cryptanalyst to analyse the key, to avoid that incrementing provides randomness for variable length key.

$H_i=\log_e(N_i)$.

M_i=mantissa (H_i)

Consider only the 32-bit hex value from mantissa.

As we got 32 32-bit hex values, concatenate two 32-bit hex values to a single subkey.ie.,

```
For each i in 1 to 16 do
Skey_i = (M_j||M_{j+1})
j=j+2.
```

Now we got 16 64-bit hex values, i.e., 1024 bits have been generated from 256-bit key. These 16 values are used as the Subkeys.

III. IMPLIMENTING FIESTAL NETWORK

This is a feistel network consisting of 16 rounds. The input is a 128-bit data element, x. Each round consists of a key dependent substitution. There is no complexity in implementing this algorithm as i used only X0R operation on 64-bit words. The keys which have been generated by subkey generator are used as inputs for each round. The message should be a multiple of 128; to accommodate this we pad the extra bits to make it absolute.

$Msg=Msg_v||P$.

Msg_v	P	X

X= no. of bits padded.

P= no. of zeros to make it absolute.

Msg_v= actual message to be encrypted.

Msg =message after padding bits.

```
Divide x into two 64-bit halves: xL, xR
For i=1 to16 do:
    xtemp = xL XOR (xR XOR Skey_i)
    xL = xR
    xR = xtemp
Next i: swap xL and xR
Combine xL and xR.
```

Decryption is exactly same as encryption except the keys are used in the reverse order; this is the advantage of using feistel network. In the function, f (xR, Skey) I used XOR to improve the speed of encryption

The network is shown in fig1.

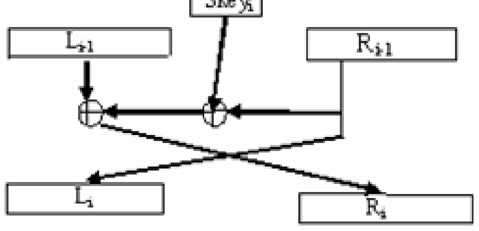


Fig1. Feistel Network.

A. *Updating keys*

Imagine an encrypted data link where you want to change keys daily. Sometimes it's a pain to distribute a new key ever day. An easier solution is to generate a new key from old key; this is sometimes called key updating. To do this we take only 8-bit hex digits from the mantissa for every block of the key and make it as new key.

IV. RESISTING VARIOUS ATTACKS

In this section I discuss various attacks which are familiar and resistance of this encryption algorithm to these attacks.

A. *Brute force attacks:*

A brute-force approach involves trying every possible key until an intelligible translation of the cipher text into plaintext is obtained. DES algorithm has received its setback in 1999 when a team has introduced DES cracker which works on a 56-bit key. On an average, half of all possible keys must be tried to achieve success. As 256-bit key is used, assuming that it takes 1µs to perform a 10^6 decryption [3], no. of alternative keys are 2^{256}=1.15*10^{77}. It takes 5.7*10^{70} years to decrypt. It's virtually infeasible. So this algorithm is resilient to exhaustive key search. This is only of theoretical interest. The way we use keys for the real time applications are confined to only alphabets and numbers as they are easy to remember, thus the first bit of every block is zero and it makes the job of brute force attack lighter.

B. *Linear cryptanalysis:*

There are 2 parts to linear cryptanalysis. The first is constructing linear equations relating plaintext, cipher text and key bits that have a high bias, that is, whose probabilities of holding are as close as possible to 0 or 1. The second is to use these linear equations in conjunction with known plaintext-cipher text pairs to derive key bits.

For example, the following eqn, from a hypothetical cipher, states the XOR sum of the first and third plaintext bits (as in a block cipher's block) and the first cipher text bit equal to the second bit of the key.

$P_1 \oplus P_3 \oplus C_1 = K_2$

If we can get the second bit of the key we can directly apply the methods discussed by Matsui in [4] and [5]

C. *Avalanche effect:*

In cryptography avalanche effect is a property which is evident if, when an input is changed slightly, the output changes significantly. It is described by horst feistel in [] Also an algorithm when each of its output bit changes with a probability of 0.5 by complementing a single bit in the input tend to exhibit Strict Avalanche Criterion[7], . The algorithm presented in this paper exhibits avalanche effect.

There are several other kinds of attacks like differential cryptanalysis in [8] and [9], integral cryptanalysis [11], boomerang attack by David Wagner in [12], can be used as a measure of validation for this algorithm. I'm not saying that this algorithm is resistant to all possible attacks, but an algorithm should be built keeping these basic attacks in consideration.

V. SIMULATION : AN EXAMPLE

In this section I describe the working model of this encryption algorithm as an example, which describes the keys and encrypted message. The simulation is done in c language; the code for this algorithm is not shown here as it is unnecessary and consumes lot of space, I show the results through example.

Key=”hello”, the hex decimal notation of Subkeys generated is as follows:

```
2950962C3AC70A86
E63FD77B245348D8
997AACF832C8F337
32C8F33732C8F337
997AACF832C8F337
32C8F33732C8F337
2F314A6D38617187
997AACF8B45AD231
32C8F33732C8F337
5EB3A27132C8F337
1CE7169921AC716D
5EB3A27132C8F337
26173DB12A33FD19.
```

These 16 digit hexadecimal keys are used in each round to encrypt the given data.

Message=”welcome to this planet, it’s just awesome.”

Encrypted message is as follows:

```
23 7C AD E7 76 E1 42 3C 38 94 9C 56 AC 42 8D A9 39 6E A7 ED 34 FB ED 6F 25 8F 9B
51 E4 41 8B FA 52 7D 7F 3B 32 E0 33 30 00 33 37 33 44 39 41 33.
```

The encrypted message is shown in hexadecimal notation to avoid ambiguity. The length of encrypted message is a multiple of 16. At the decryption side, pad is removed after decryption.

VI. RELATED WORK

Information security is an extensively study field of research. Various encryption algorithms have been proposed for the last three decades [14]. Cellular and cordless telephones transmit conversations via radio waves that can be easily intercepted. Electronic mail messages pass openly from one computer to another across a network accessible to innumerable people. So there is never ending demand to develop new algorithms every year and the previously proposed algorithms are revisited by many cryptographers around the world. Cryptographers are tailoring the algorithms such that they reach the needs of this contemporary world of information security. Brand new publicized algorithms shouldn’t be used, they need to be first exposed to many cryptanalysts and should prove flawless, so even when there are a lot of encryption algorithms are published every year so some stand as standard encryption algorithms. There is lot of research work going on in universities and various private and public organizations in there enthusiasm to build sophisticated algorithms.

VII. CONCLUSIONS

In this paper I presented Subkey generation for block symmetric encryption and also the algorithm for encryption using feistel network. There is never ending research going on to develop new and new encryption algorithms to fit the changing world, every year several algorithms are proposed for standards, we can’t conclude an algorithm as the best because it depends on the real world environment and how situation demands it. The results shown in the simulation imitate real time encryption. The keys and message are displayed in hex decimal to reduce ambiguity. As we know that ASCII representation contains characters like NULL, BS etc which creates some kind of ambiguity when decrypting, that is, the characters like back space may delete the previous characters and while decryption we may not get the desired text.

ACKNOWLEDGMENT

I would like to thank Bruce Schiener for his book "Applied Cryptography" which helped me a lot to write this paper and my academic book "Network Security Essentials" by William Stallings which drew my interest to write this paper.

REFERENCES

1. *Walter Tuchman."A brief history of the data encryption standard". Internet besieged: countering cyberspace scofflaws. ACM Press/Addison-Wesley Publishing Co. New York, NY, USA pp.275-280. 1997.*
2. Horst Feistel," Cryptography and Computer Privacy." Scientific American, vol. 228, no.5, 1973.
3. William Stallings, Network Security Essentials: Applications and Standards.3nd ed. Pearson Education.
4. Matsui, M and Yamagishi, A" A new method for known plaintext attack of FEAL cipher". Advances in Cryptology-EUROCRYPT 1992.
5. Matsui, M."The first experimental cryptanalysis of the data encryption standard." Advances in Cryptology CRYPTO 1994.
6. Matsui, M."Linear cryptanalysis method for DES cipher" Advances in cryptology-EUROCRYPT 1993.
7. A.F. Webster, Stafford E. Tavaraes: On the design of S-Boxes. CRYPTO 1985:523-534.
8. Eli Biham, Adi Shamir, Differential Cryptanalysis of the Data Encryption Standard, Springer Verlag, 1993.
9. Eli Biham, Adi Shamir, "Differential Cryptanalysis of the Full 16-Round DES," CS 708, Proceedings of CRYPTO '92, Volume 740 of Lecture Notes in Computer Science, December 1991.
10. Joan Daemen, Lars Knudsen, Vincent Rijmen (January 1997). "The Block Cipher Square" . *4th International Workshop on Fast Software.*
    Encryption (FSE '97), Volume 1267 of Lecture Notes in Computer Science. Haifa: Springer-Verlag. pp. 149–165
11. Lars Knudsen, David Wagner. "Integral Cryptanalysis". *9th International Workshop on Fast Software Encryption (FSE '02)*. Leuven: Springer-Verlag. pp. 112–127. 2001
12. David Wagner. "The Boomerang Attack". *6th International Workshop on Fast Software Encryption (FSE '99)*. 1999.
13. Bruce Schiener, applied cryptography: protocols algorithms and source code in c, second edition, John Wiley and sons.
14. Limor Elbaz, Hagai Bar-El, Strength Assessment of Encryption Algorithms. Discretix technologies ltd. 2000

Article received: 2010-07-01