Kangaroo A Proposed Stream Cipher Algorithm

Dr. Abdul-Monem S. Rahma*
& Dr. Suhad AbdulZahra Hassan Al-Quraishi**

Received on:18/6/2008
Accepted on:3/9/2009

Abstract

Stream cipher designers offer new algorithms by employing the characteristics of the basic structure to get algorithms that have advanced features from security and speed stand point of view. Taking into consideration the benefit from the scientific and technical development at the publish time of their algorithms versions. This research proposes the design of a new efficient and secure stream cipher, named *(Kangaroo)* which was designed to be more secure than conventional stream cipher systems that use XOR for mixing. The proposed algorithm encompasses two major components. The first part involves the Pseudo Random Number Generator (PRNG) using *Rabbit* algorithm. And the second part involves a nonlinear invertible round function (combiner) to perform the encryption and decryption processes. This will lead to strengthen the weak XOR combiner. The proposed cipher is not a random number generator alone. Instead it is *self-synchronizing* stream cipher such that - the cipher text influences its internal functioning. The proposed algorithm uses 16-bytes secret key to encrypt the plaintext which is multiple of 16-bytes up to (2^{64} bytes) length. Also the performance and memory requirements of the proposed system are evaluated. Such that the setup stage requires 1987 operations and the encryption/decryption processes require 4 operations per byte. And the memory requirement is 344 bytes. The implementation was done on a (Pentium IV, 1.79 GHz, 224 MB of RAM) PC, Windows XP, using (Java Builder 7).

Keywords: Additive stream cipher, AES block cipher round, Combiner.

" الكنغـــــــر" خوارزميـــة مقترحـــة للتشفيـــر الإنسيـــــــابي

الخلاصة

يُقدّم مصمّمي التشفير الانسيابي خوارزميات جديدة بتوظيف خصائص الهيكل الأساسي للحصول على خوارزميات لها صفات متقدمة من وجهة نظر الأمنية والسرعة. مع الأخذ بنظر الاعتبار الاستفادة من التقدم العلمي والتكنولوجي لزمن اصدار خوارزمياتهم.

هذا البحث يقترح تصميم خوارزمية تشفير انسيابي جديدة كفوءة وامينة بإسم **(الكنغـر)** التي صممت لتكون اكثر امنية من انظمة التشفير الانسيابي التقليدي التي تستخدم XOR للخلط. الخوارزمية المقترحة تشمل جزئين رئيسيين. الجزء الأول يتضمن مولد الأرقام العشوائية الكاذبة PRNG باستخدام خوارزمية الأرنب Rabbit. والجزء الثاني يتضمن دالة دورة Round function غيرخطية قابلة للعكس لانجاز عملية التشفير وفك الشفرة. هذا يؤدي لتقوية الدامج الضعيف XOR . تصميم الـــ PRNG يعتمد على تكرار مجموعة من الدوال المزدوجة الغير خطية لتوليد سلسلة الارباك confusion sequence. دالة الدورة تذكّر بتصميم دورة التشفير الكتلي AES. على كل حال , التشفير المقترح ليس مولد ارقام عشوائية فقط. انما هو تشفير إنسيابي ذو تزامن ذاتي self-synchronizing - النص الواضح يؤثر في وظيفته الداخلية.

* **Computer Science Department, University of Technology / Baghdad**
** **Ministry of Electricity/ Baghdad**

https://doi.org/10.30684/etj.28.3.12
2412-0758/University of Technology-Iraq, Baghdad, Iraq

تستخدم الخوارزمية المقترحة مفتاح سري 16-byte لتشفير النص الواضح الذي يكون من مضاعفات الـــ 16-bytes الى طول (2^{64} bytes). كذلك تم تقييم الأداء ومتطلبات الذاكرة للنظام المقترح. حيث تحتاج مرحلة الإعداد الى 1987 عملية وتحتاج عمليات التشفير/ فك الشفرة الى 4 عمليات لكل byte. ومتطلبات الذاكرة هي (344 bytes).
تم التنفيذ على حاسبة شخصية (Pentium IV, 1.79 GHz, 224 MB of RAM)، Windows XP، باستخدام (Java Builder 7)

Introduction

Stream ciphers are an important class of symmetric encryption algorithms. They encrypt individual characters (usually binary digits) of a plaintext message one at a time, using an encryption transformation which varies with time. They are also more appropriate, and in some cases mandatory (e.g., in some telecommunications applications), when buffering is limited or when characters must be individually processed as they are received. Stream ciphers employ- in Shannon's terminology confusion only [1] [2]. Their basic design philosophy is inspired by the Vernam (One-Time Pad) cipher, which encrypts by XOR'ing the plaintext with a random key. The drawback of the Vernam cipher is the requirement that key must be a true random sequence, shared by the sender and the receiver, and can only be used once [3][4]. A combiner is the heart of a stream cipher, which generally uses an "additive" combiner such as exclusive-OR. Unfortunately, additive combiners have absolutely no strength at all: If an opponent somehow comes up with some plaintext and matching cipher text, an additive combiner immediately reveals the confusion sequence. This allows the opponent to begin work on breaking the confusion sequence generator [5] [11].
An alternate approach to the design of a secure stream cipher is to seek combining functions which can resist attack; such functions would act to hide the pseudo-random sequence from analysis. Such cryptographic combining functions could be used to replace the Vernam exclusive-OR combiner (if they have an inverse). An improved combiner is intended to force cryptanalysis to be more difficult, time consuming, and expensive than would be the case with a simple combiner [6].
To complicate the weak exclusive-OR combiner, This Paper Produce a new scheme use a hybrid concept between the block cipher round and stream cipher system.
The proposed cipher scheme aims to use large block sizes in order to overcome attacks such as frequency analysis and ciphertext/plaintext pairs.
The research in the design named **Kangaroo** algorithm and it use key dependent s-boxes (base on the Rijndael like function) [12].

3. Chameleon Stream Cipher

Chameleon is built from a new cryptographic primitive, called a Mutating S-box that shares some basic internal structures with RC4 [7].

4. Rabbit Stream Cipher

Rabbit is characterized by a high performance in software with a measured encryption/decryption speed of 3.7 clock cycles per byte on a Pentium III processor. Rabbit have been specified to be suited for both software and hardware implementations [13].

The algorithm is initialized by expanding the 128-bit key into both the eight state variables and the eight counters such that there is a one-to-one correspondence between the key and the initial state variables $X_{j,0}$and the initial counter $C_{j,0}$,the key $K^{[127..0]}$ is divided into eight sub keys: $k_0=K^{[15..0]}$,$k_1=K^{[31..16]}$,…,$k_7=K^{[127..112]}$. The state and counter variables are initialized from the sub keys. The core of the Rabbit algorithm is the iteration of the system defined by the following equations [8]:

$$x_{0,i+1} = g_{0,i} + (g_{7,i} <<< 16) + (g_{6,i} <<< 16)$$
$$x_{1,i+1} = g_{1,i} + (g_{0,i} <<< 8) + g_{7,i}$$
$$x_{2,i+1} = g_{2,i} + (g_{1,i} <<< 16) + (g_{0,i} <<< 16)$$
$$x_{3,i+1} = g_{3,i} + (g_{2,i} <<< 8) + g_{1,i}$$
$$x_{4,i+1} = g_{3,i} + (g_{3,i} <<< 16) + (g_{2,i} <<< 16)$$
$$x_{5,i+1} = g_{5,i} + (g_{4,i} <<< 8) + (g_{3,i}$$
$$x_{6,i+1} = g_{6,i} + (g_{5,i} <<< 16) + (g_{4,i} <<< 16)$$
$$x_{7,i+1} = g_{7,i} + (g_{6,i} <<< 8) + g_{5,i}$$

$$g_{j,i} = ((x_{j,i} + c_{j,i+1})^2 \oplus ((x_{j,i} + c_{j,i+1})^2 >> 32)) \bmod 2^{32}$$

5. Proposed design Overview

Dynamic substitution is a way to build a cryptographic combiner; it is not a complete cipher. It is used simply as a replacement for the weak exclusive-OR combiner normally used in stream ciphers. The strength of dynamic substitution combiner can support the use of weaker but faster PRN generators. Nowadays, this does not provide the required security. Also this does not represent a real blend (mixing together) between the plaintext and key stream. Instead, it is merely, a hiding process to the plaintext without affecting its actual bits directly. · Therefore, if the confusion RNG is linear, with a small amount of state, the opponent may be possible to try various sets of key stream values until the system is solved. But if the RNG has a large amount of state, selecting a set of correct random values from the larger set of possible key stream values (by many trials) will be very difficult.

5.1 The choice for name

The jumping and feedback characteristics of the proposed algorithm, make it has additional property more than the Rabbit algorithm i.e. it contains the second part (combiner) followed by the feedback. From this situation, the name comes, as a general form; it is similar to Kangaroo, which contains a pocket in front of its body holding its offspring (progeny). In other words the algorithm has a property of holding the ciphertext for the next processes. And accordingly, the algorithm is named Kangaroo that is, its child (is adjacent) in its pocket.

5.2 Design Motivation

The motivation for the choice of the design in the proposed cipher is summarized as follows:

1. PRNG: The PRNG in the proposed stream cipher is implemented using Rabbit algorithm due to the following security properties:

- The cryptanalysis of Rabbit does not reveal an attack better than exhaustive key search.
- Rabbit is characterized by a high performance in software. Rabbit was designed to be faster than commonly used ciphers and to justify a key size of 128 bits for encrypting up to 2^{64} blocks of plaintext.
- The simplicity and small size of Rabbit make it suitable for implementations on processors with limited resources such as 8-bit processors.

2. Combiner: The Combiner in the proposed stream cipher is implemented using AES-round like function due to the following properties:

1. The Rijndael round is used as a nonlinear invertible combiner, since it provides efficient implementation on 8-bit processors, typical for current. Smart cards, and on 32-bit processors, typical for PCs. Thus providing the speed and compactness on wide range of platforms.
2. Design simplicity.
3. Resistance against known attacks.

5.3 Algorithm Inputs

1. The key (k0,kl, k2, ..., kI5): it is the main key of the algorithm. This key is 16-bytes (l28-bits) which entered to the first part (PRNG) to produce the keystreams (Ks1, Ks2, Ks3, Ks4 ...). Each Ks is 16 bytes keystream in each iteration.
2. The plaintext (P1, P2, P3, P4 ...): it is the real message to be encrypted. Each P which is 16-bytes enters the second part (combiner).
3. Text1 (P0): it is used only once in the algorithm. It is 16-bytes text used as a starting (virtual) plaintext which is input to the encryption/decryption function to produce the starting virtual ciphertext C0 (which is used in the feedback).
4. Text2 (C0): it is a virtual ciphertext that is generated before actual encryption/decryption process and it is used in the feedback.

5.4 Algorithm Outputs

The ciphertext (C1, C2, C3, C4 ...): it is the result of combining keystream with plaintext. Each C is 16-bytes for each round.

5.5 Algorithm Stages

Figure (1, 2) illustrates the mechanism used in the algorithm stages as described below:

5.5.1 Setup Stage

This stage is performed by both the sender and receiver. The next state function defined in Rabbit algorithm, is called four times. At the fourth iteration, 16-bytes (128-bit) are extracted to be assumed as a virtual keystream Ks0 which is input with the P0 to the combiner function to produce the cipher C0. Each of Ks0, P0, and C0 is known to both the encipherer and decipherer. Then C0 is input to the PRNG to modify the $C_{j,i}$ counter variables of the internal states. At this point the modified keysetup is completed. The proposed algorithm now is ready to get the actual keystream (Ks1, Ks2 ...) to be used in the encryption and decryption process.

5.5.2 Encryption Stage

Here, the first 16-bytes block of plaintext P1= (p0, p1, p2, p15) and 16-bytes keystream Ks1= (ks0, ks1, ks2, ks15) are input to the combiner (forward round function) and the output is 16-bytes ciphertext C1= (c0, c1, c2, ... c15). But the encryption of the second 16-bytes block of plaintext P2 requires returning the previous block of ciphertext C1 to the RNG.

This pervious ciphertext affects the next 16-bytes keystream Ks2 generation. Then, the second 16-bytes keystream Ks2 and the second 16-bytes block of plaintext P2 enter the encryption function to produce the second 16-bytes block of ciphertext C2. Also the same mechanism is used to encrypt P3, P4....

5.5.3 Decryption Stag

When the receiver obtains the ciphertext Cl. C1 is input to combiner (inverse function) with Ksl to get PI as a first plaintext block. To get P2, C1 is input to the Rabbit PRNG to get Ks2 which is combined with the C2 to produce P2. The same manner is used to decrypt C3, C4...

5.6 Description of the Combiner

The new combiner shares more than one character in the encryption/decryption process. A graphical description of the combiner (round function) is illustrated in Figure (3).

1. The first operation is XOR; this ensures an absolute security to the plaintext. And on the other hand the remaining transformations themselves will be a real complexity against the opponent.
2. Using key dependent S-box in the combiner ensures large difficulties. Since S-box is unknown to some scope, it increases the number of probabilities faced by the attacker.
3. Since XOR represents uniquely the first point of meeting between the plaintext with the confusion sequence. Then the adversary cannot analyze the other parts that transmit the effect of bits among each other. In other words, suppose you use the round function of ***Blowfish*** block cipher algorithm [9][12], as an alternate of Rijndael, and it is assumed to be applied as in Figure(4)

In this Figure, the keystream is input at the top and the plaintext is input at the left-hand side, consequently it easier for the opponent to analyze this function. The left-hand side of keystream encrypts the plaintext using XOR. Then the result enters the function F, and it is encrypted using right-hand side of the keystream. Since the two parts of the keystream are generated. Using the same resource effects as (one key), and then these parts have collective characteristics. Thus the normal abilities of the attacker are sufficient to treat the two parts as one key used to encrypt two plaintexts (one of them is the original plaintext and the other is the output from the F-function). This leads to expose the plaintext in spite of the strength of the keystream. Certainly, choosing Rijndael round function is a better choice for Kangaroo algorithm.

Probability that some items appear with the existence of dynamic folding in the proposed cipher combiner encompasses both rows and columns and involves mutating elements depending on a specific issue. As further effectiveness, it increases probabilities against the cryptanalyst. This means, adding to the main body of the combiner some key dependent operation before outputting each block. There are 16 locations in the (4×4) state array to start the new permutation according to a specific direction. Instead of using a fixed arrangement for the state, there are 64 (4 × 4) state array. Thus the opponent can't specify the correct order easily and this will increase the difficulties.

5.7 Speed Measuring & Memory Requirement

Due to the high portable nature of Java, the implementation of the proposed algorithm in Java could be tested on all platforms for which *NM* is available. Platforms specific tuning and code changes are not needed as may be required in the case of C/C++ implementation.

Table (1) explains the speed measuring and memory requirement of the proposed cipher. Speed

estimation is given in number of operations per byte according to the plain implementation computed. Also the speed of implementing the cipher in Java on Pentium 4 is presented in MB per Second. Memory requirement is 68 bytes for key setup (32 bytes for x states, 32 bytes for c, and 4 bytes for the carry), and for encryption/decryption is 256 bytes plus 16 bytes for state array, in addition to the four bytes to specify the direction and position of dynamic folding i.e. (HorD, VerD, Row, and Col respectively).

5.8 Comparison of the Proposed Cipher Chameleon

In RC4 and Chameleon there is an obvious symmetry between the state and the cycle that produces a clear biasing in the output, but the initialization vector is used to modify the state which is time consuming. In proposed cipher design there is no symmetry between the state and the cycle. Two different algorithms are needed (Rabbit PRNG and AES-like round function) that avoid biases and make the cryptanalysis more difficult. Table (2) summarizes the performance evaluation and memory requirements of Chameleon and the proposed cipher.

Figure (4) shows the comparison between Chameleon and the proposed cipher (in term of key setup, encryption, decryption, and memory bytes). From the presented result, it is shown that the proposed cipher has better performance than Chameleon stream cipher. Also the memory requirements are less in the proposed cipher.

5.9 Security Analysis with Possible Attacks

1. Exhaustive key search: With a key length of 128 bit, there are 2128 possible keys, which are approximately 3.4 x 1038 keys; moreover, Text1 is changed with each communication. Thus, a brute-force attack appears impractical.
2. Ciphertext Only: The opponent accumulates a collection of ciphertext objects and tries to find relationships within the data which expose the system, until the cipher is broken. The plaintext data that is input to the combiner of the proposed algorithm is randomized using the keystream sequence by exclusive-OR. Then this data is treated by many transformations of the round function that include SuBytes, ShiftRows, and dynamic folding. Then the letter frequency statistics will be concealed and the result is random-like output.
3. Known Plaintext: When the attacker has some plaintexts and the related ciphertext, then this attack when using exclusive-OR combining will expose the keystream. The keystream and knowledge of the design of the generator can be used to conclude the initial state, which leads to break the cipher. In the round function, each byte is XOR'ed with the keystream and substituted using keyed S-Box, then the unknown values of the keystream and the shuffled S-Box make the attacker unable to determine the plaintext byte. Also the dynamic folding according to the keystream selected, changes dynamically each iteration. This forms a more sophisticated state when deciphering the ciphertext byte.
4. Statistical Attacks: The nonlinearity in the Rabbit PRNG components and the additional

nonlinear transformation in combiner round function serve to erase the statistical weakness in the keystream. The keystream bit sequences in the proposed algorithm have extensively tested using the statistical tests and have detected no statistical weaknesses.

5. Time-memory trade-of attacks: This kind of attack can be applied if the state space of the cipher is too small. These kinds of attacks do not seem to be applicable to the proposed algorithm.
6. Guess-and-Determine attacks: The basic idea is to guess the value of some unknown variables in the cipher and from the guessed values deduce the value of other unknown variables. In this algorithm, the dynamic folding in the combiner (round function) is the result of nonlinear mapping in the 4 x4 state array. Thus this attack is invalid for the proposed algorithm.
7. Differential Analysis: Differential analysis is invalid to apply to the proposed cipher, because the employed tables are "keyed" i.e. initialized by a particular key. This means that the attacker does not have a prior knowledge of a particular table arrangement.
8. Distinguishing and Correlation Attacks: The nonlinear combiner of the proposed cipher has been defined to use a essential amount of inputs e.g. (virtual plaintext and the shuffled (key dependant) S-box to perform a strong transformations), in addition the nonlinear equations of the PRNG make this kind of attack inapplicable.

5.10 Randomness Tests of the Keystream bits

The randomness property of the PRNG is analyzed by using the statistical tests: Frequency, Serial, Poker, Runs and Autocorrelation tests. Different key sizes are used in these tests. The keystream generated in the proposed cipher successfully passes all the five statistical tests for every run. Table (3) shows the results of applying the statistical equations.

6. Conclusion

The mechanism used in this thesis is a modem stream cipher with combining function which extends the weak classical concept of simple XOR combiner into a stronger form which is suitable for computer cryptography. This research has presented a new stream cipher named Kangaroo. A complete description of the algorithm, an evaluation of its performance, security properties and implementation aspects were given. The analysis shows that the proposed cipher is very secure. The major outlines that are concluded from this study are described as follows:

- Using cipher feedback approach in the proposed cipher has added extra processing time, but overall it is relatively negligible especially for certain applications that require more secure encryption to a relatively large data blocks.
- The nonlinearity of the round, feed backing, and the selection of the taps for the output function, sufficiently disguise any short-distance correlations.
- Finally it is found that the proposed cipher has the following strengths:

Flexible: it has clear flexibility in the processor size and implementation.

Small memory: it requires a small amount of memory.
Speed: it is fast, in particular, compared to RC4-like algorithms.
Secure: it appears to provide more than adequate security normally provided by conventional stream ciphers.

References

[1] Menezes A., Van Oorschot P. and Vanstone S., "*Handbook of Applied Cryptography*", CRC Press, Inc., 1997.

[2] Stamp M., "*Information Security: Principles and Practice*", Published by John Wiley & Sons, Inc., 2006.

[3] Alfred J.M., Paul V. C. and Scott A. V., "*HandBook of Applied Cryptography*", Fifth Addition, CRC Press, 2001.

[4] Robshaw M. J. B., "*Stream Ciphers*", RSA Laboratories Technical Report, a division of RSA Data Security, Inc., 1995.http://www.comms.scitech.susx.ac.uk/fft/crypto/stream_ciphers.pdf

[5] Ritter T., "*Dynamic Substitution in Stream Cipher Cryptography: A Reversible Nonlinear Combiner with Internal State*", 1997.http://www.ciphersbyritter.com/DYNSUB.HTM

[6] Ritter T., "*Substitution Cipher with Pseudo-Random Shuffling: The Dynamic Substitution Combiner*", Cryptologia,1990.http://www.ciphersbyritter.com/ARTS/DYNSUB2.HTM

[7] Matthew E. McKague, "*Design and Analysis of RC4-like Stream Ciphers*", A Thesis presented to the University of Waterloo in fulfillment of the thesis requirement for the degree of Master of Mathematics In Combinatorics and Optimization, Canada, 2005.

[8] Boesgaard M., Pedersen T., Vesterager M., and Zenner E., "*Rabbit Stream Cipher – Design and Security Analysis*", Fast Software Encryption, 2003.

[9] Schneier B., "*Description of a New Variable-Length Key, 64-Bit Block Cipher (Blowfish)*", Fast Software Encryption, Cambridge Security Workshop Proceedings, 1994.

[10] Rose G., Hawkes P., "*Turing: a fast stream cipher*", QUALCOMM Australia, 2002.http://www.theargon.com/achilles/cryptography/turing/Turing.pdf

[11] Ritter T., "*The Penknife Cipher Design*", Cryptologia, 1996.http://www.ciphersbyritter.com/PENDESN.HTM

[12] Stallings W., "*Cryptograhy and Network Security: Principles and Practice*" , Second Edition, Prentice Hall, Inc., 1999.

[13] Frank K. Gurkaynak, Peter L., Nico B., Rene B., Victoria G., Marcel M., Hubert K., Norbert F. and Wolfgang F., "*Hardware Evaluation of eSTREAM Candidates*", Zurich, 2006.

Test vectors

The following test vectors shows(in details) the result of implementing the proposed cipher for encryption/decryption process.

Encrypting

The plaintext

P*: 23 34 88 76 33 56 87 09 40 18 3a 2c ef 4d 16 c7 65 3c 4b 32 a6 53 29 65 42 31 bc ea cd f1 f6 c7*

The main key

K*: 40 31 34 32 23 70 61 77 26 61 77 73 34 38 35 39*

The virtual plaintext

P0*: 64 65 77 33 36 35 67 21 36 35 2a 6c 6a 68 3f 37 35 33 30*

The S-box key

Sboxk*: 61 43 64 46*

The implementation steps are as follows:

Sboxk is used to shuffle the Rijindael S-Box whose new 256 elements are in the following form:

S-Box:

e0 e5 38 de 4b ec 05 70 f7 3c 96 b5 30 4f af 1d 3e 66 ed 44 8c 41 0a 34 fc 6f bd 07 80 f8 db ac 39 a3 75 d7 26 55 b9 32 cc e8 c1 f9 d4 9d 10 58 a9 3a 3d d5 47 5e 14 7d 24 b2 9b 2c 73 2b 3f cb 23 16 11 65 48 9e 4e 6a 78 c6 62 84 f0 8e 6e 51 49 be 2a 1f 2e 67 88 bf dc 22 68 60 7e aa d3 e1 1c b7 7a b1 59 0e a1 c0 fa 79 ea 5a 9c 18 8a 53 57 45 e7 69 89 4d a0 f6 c4 a8 87 46 74 7f 13 29 cf 72 6b 17 ba d6 1b 8f 40 d8 e4 2d d0 20 37 1e 42 9f e6 64 ce ab 50 e9 97 e2 06 3b 43 08 63 b6 77 c9 71 a4 eb 36 b3 6c 28 a5 f5 54 61 33 ca 93 00 a2 d2 c8 da 25 d1 19 fe 31 ff e3 6d fb 09 95 cd 35 a7 0d fd 27 76 ae 8d 7b c2 91 01 bc 15 83 5d 0b f4 f2 99 ee 5b f1 ef d9 df 0f f3 b8 52 0c 5c 56 98 86 9a c5 4c 92 81 dd 8b 4a 7c 5f 90 82 a6 02 2f bb ad 21 85 c7 94 04 c3 12 b0 b4 03 1a

The key setup stage is initiated Then the virtual keystrem ks0 (16-bytes) is extracted. And after 3rd iteration its value is:

Ks0: 34 67 02 83 a2 4f da 3a 8c cb 29 bc 78 ed 1a 8f

Both Ks0 and P0 are input to the combiner to generate the virtual ciphertext C0 according to the following steps

C0: fb 5d ed 38 de fe 49 87 55 00 ce 03 4d 07 ff d6

Now generate the actual keystreams and *encrypt* using the combiner.

At *Iteration 1*

Ks1: e6 ad bc 90 a3 c5 eb 7d 1f 3b 01 8c 2d a2 91 2c

The *first 16-bytes* block of plaintext *p1* are read from a text file. This plaintext block is as follows:

P1*: 23 34 88 76 33 56 87 09 40 18 3a 2c ef 4d 16 c7*

P1 and *ks1* are input to the combiner, to get C1

C1**:** 27 64 2c 4a 42 d7 8f 4c e1 82 47 89 a7 e2 9c 77

At Iteration 2

Ks2: 87 03 dd 75 dd 21 55 6e 51 ca 47 8e 27 d1 ec 34

P2*: 65 3c 4b 32 a6 53 29 65 42 31 bc ea cd f1 f6 c7*

C2: 39 50 b5 44 cb 74 59 8b e7 12 bb 98 12 bd 6a 46

C*: 27 64 2c 4a 42 d7 8f 4c e1 82 47 89 a7 e2 9c 77 39 50 b5 44 cb 74 59 8b e7 12 bb 98 12 bd 6a 46*

The main key

K*: 40 31 34 32 23 70 61 77 26 61 77 73 34 38 35 39*

The virtual plaintext

P0*: 64 65 77 33 36 35 67 21 36 35 2a 6c 6a 68 3f 37 35 33 30*

The S-box key

Sboxk*: 61 43 64 46*

The inverse S-Box is constructed from the forward S-box using *Sboxk*

At *Iteration 1*

Ks1: e6 ad bc 90 a3 c5 eb 7d 1f 3b 01 8c 2d a2 91 2c

c1*: 27 64 2c 4a 42 d7 8f 4c e1 82 47 89 a7 e2 9c 77*

C1 and *ks1* are input to the combiner, to get P1.

P1*: 23 34 88 76 33 56 87 09 40 18 3a 2c ef 4d 16 c7*

At Iteration 2

Ks2: 87 03 dd 75 dd 21 55 6e 51 ca 47 8e 27 d1 ec 34

C2*: 39 50 b5 44 cb 74 59 8b e7 12 bb 98 12 bd 6a 46*

Then *C2* and *ks2* are input to the combiner for P2

P2*: 65 3c 4b 32 a6 53 29 65 42 31 bc ea cd f1 f6 c7*

The total plaintext message which is obtained at the receiver end is as follows:

P*: 23 34 88 76 33 56 87 09 40 18 3a 2c ef 4d 16 c7 65 3c 4b 32 a6 53 29 65 42 31 bc ea cd f1 f6 c7*

Table (1) Speed measuring and the required memory of the proposed cipher

Cipher	*Encryption Operation/byte*	*Decryption Operation/byte*	*Speed MB/S*	*Memory Requirements in Bytes*
Kangaroo	4 *operation*	4 *operation*	*66 MB/S*	*key setup: 68* *Enc. / Dec. : 276* { *256 , 16 , 4*

Table (2) Performance evaluation and memory requirements of Chameleon and the proposed cipher

Cipher	*Key Schedule without IV*	*Encryption 16-byte*	*Decryption 16-byte*	*Memory Requirements*
Chameleon	3328 *operation* { *3 × 256* *10 × 256*	80 *operation*	112 *operation*	*513 Bytes*
Kangaroo	*1987 operation* { *350* *6 × 256* 32 64 5	64 *operation*	64 *operation*	*344 Bytes*

Table (3) Statistical Test of PRNG of the Proposed Algorithm

Tests		*256 bit*	*384-bit*	*512-bit*
Frequency test		0.5625	0.5104167	0.28125
Run test		0.41151961	0.26455128	0.021514893
Poker test		2.463904	4.283115	1.721505
Serial test		0.67282104	0.66711426	1.9320679
Autocorrelation test for 10 shifts	*Shift 1*	0.020383043	1.9317618E-4	0.42174298
	Shift 2	0.029912446	0.18952855	0.24039537
	Shift 3	0.34607568	0.18255141	0.2625798
	Shift 4	0.13385548	0.08507988	1.12325324E-4
	Shift 5	0.15841436	0.32877094	0.80725235
	Shift 6	0.31395411	0.19803871	1.3243107
	Shift 7	0.005365565	0.005590211	0.3403424
	Shift 8	1.230112	0.48235703	0.5545301
	Shift 9	0.0783614	0.1367103	0.39982912
	Shift 10	0.1473804	0.0019449207	0.15548588

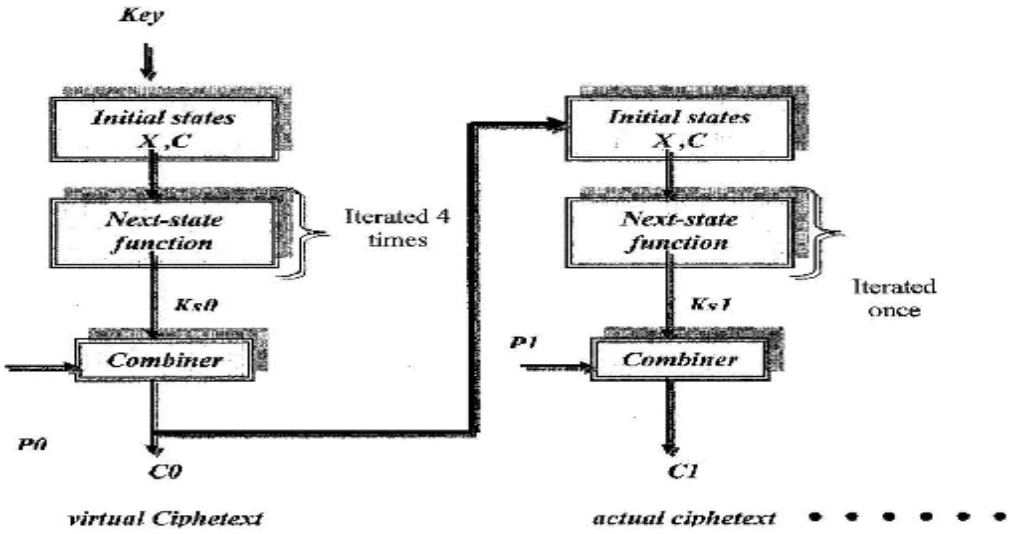


Figure (1) Encryption

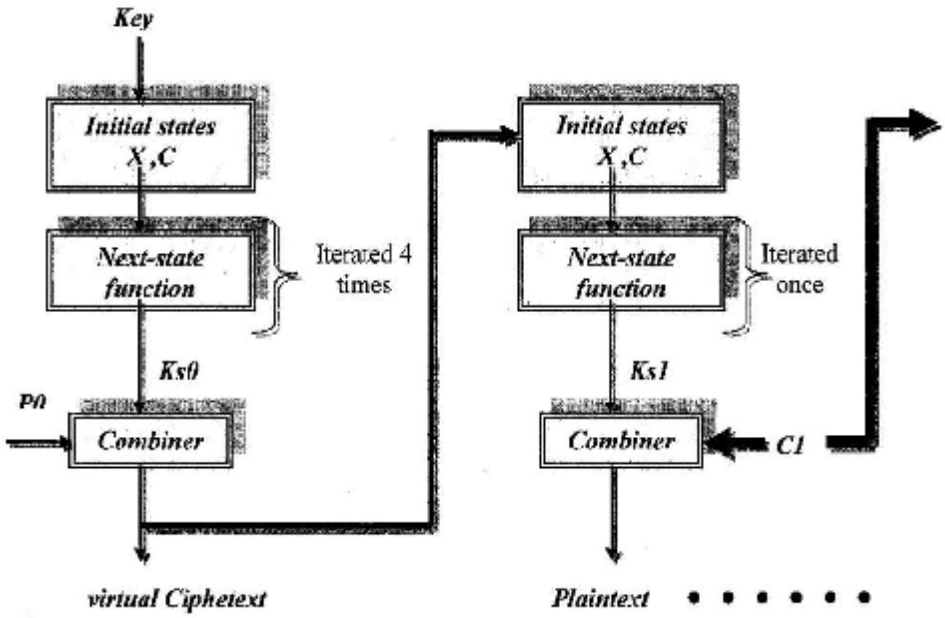
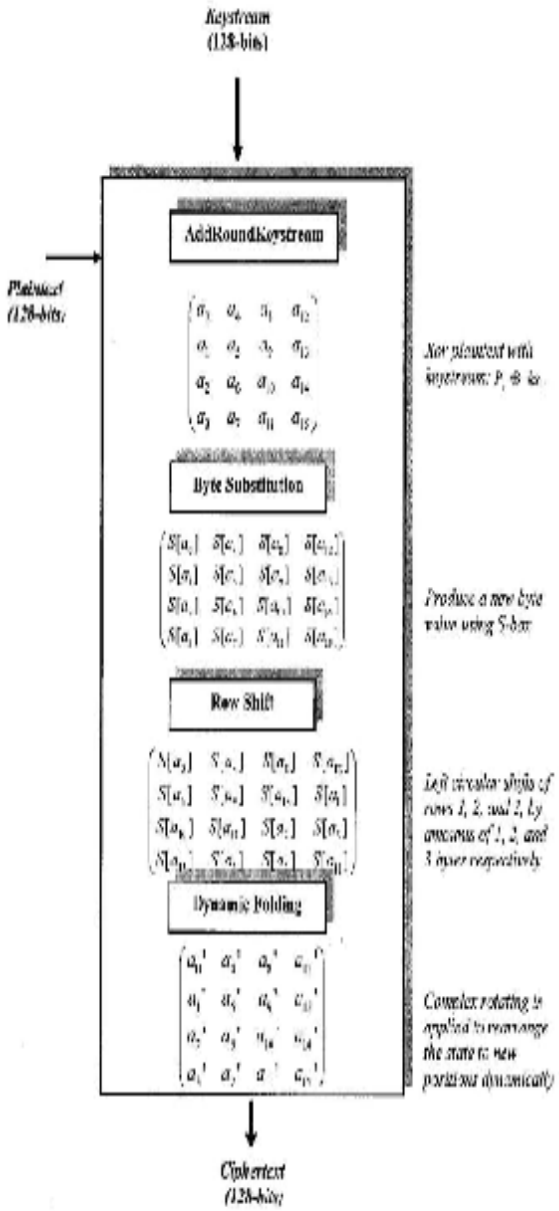


Figure (2) Decryption



Figure(3) The combiner of the proposed cipher

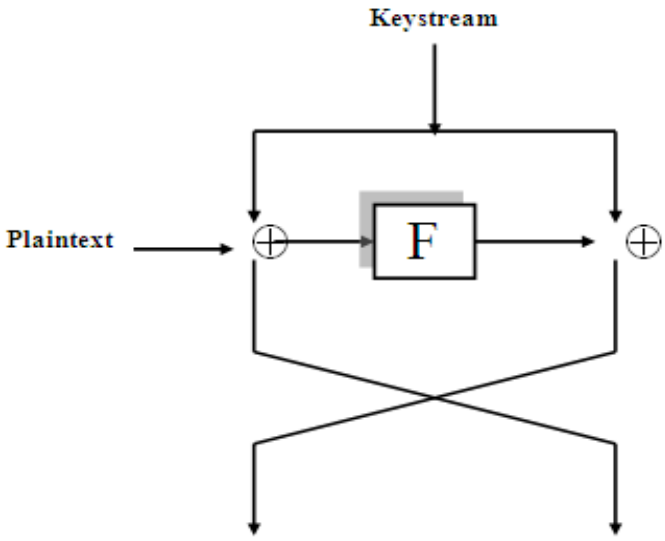


Figure (4) the supposed Round of Blowfish algorithm

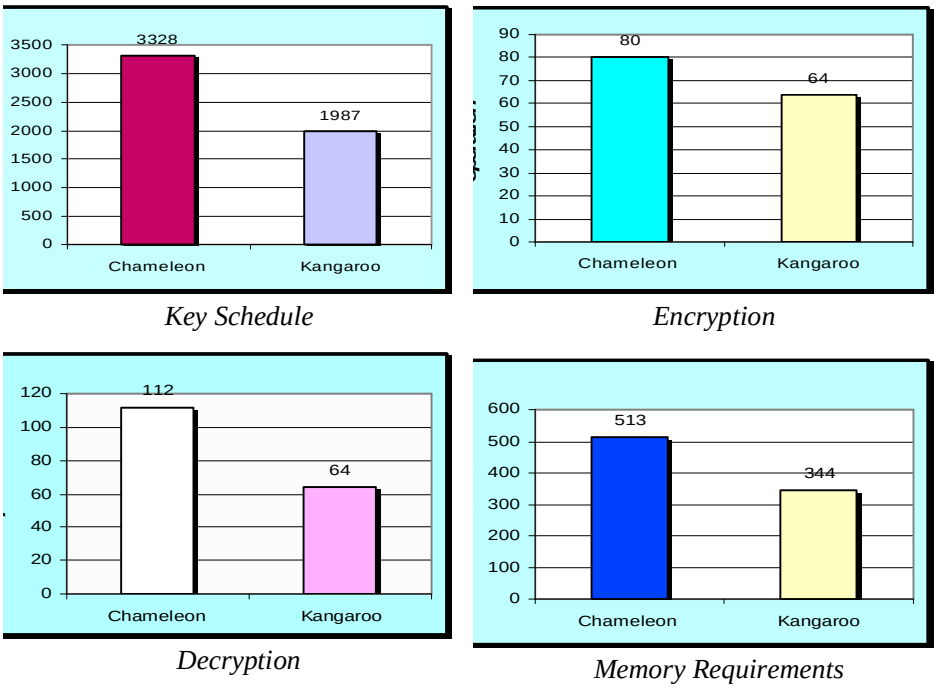


Figure (5) Comparison between Chameleon and Kangaroo