Journal of Engineering Science and Technology
Vol. 12, No. 4 (2017) 1111 - 1128

BLOSTREAM: A HIGH SPEED STREAM CIPHER

ALI H. KASHMAR[1, 2,*], EDDIE S. ISMAIL[1]

[1]School of Mathematical, Sciences Faculty of Science and Technology,
Universiti Kebangsaan Malaysia, 43600 UKM Bangi, Selangor DE, Malaysia
[2]University of Baghdad, College of Science, Baghdad, Iraq
*Corresponding Author: Kashmar992000@yahoo.dk

Abstract

Although stream ciphers are widely utilized to encrypt sensitive data at fast speeds, security concerns have led to a shift from stream to block ciphers, judging that the current technology in stream cipher is inferior to the technology of block ciphers. This paper presents the design of an improved efficient and secure stream cipher called *Blostream*, which is more secure than conventional stream ciphers that use XOR for mixing. The proposed cipher comprises two major components: the Pseudo Random Number Generator (PRNG) using the Rabbit algorithm and a nonlinear invertible round function (combiner) for encryption and decryption. We evaluate its performance in terms of implementation and security, presenting advantages and disadvantages, comparison of the proposed cipher with similar systems and a statistical test for randomness. The analysis shows that the proposed cipher is more efficient, high speed, and secure than current conventional stream ciphers.

Keywords: Cryptography, Stream cipher, Non-linear invertible round function, Combiner algorithm.

1. Introduction

In designing a cryptographic algorithm, it is necessary to construct good ciphers that can be applied to more difficult processes, balancing a number of properties against each other. Some of the main considerations are typically security level, encryption speed, speed of initialization, hardware needs and software needs. The modern additive cipher commonly relies on generating pseudo-random sequences that are resistant to some security analysis. However, the construction of such a random-number generator (RNG) is not a simple process, since the opponent also

has good analytical knowledge and capabilities that enable him to attack these designed generators [1, 2].

Stream ciphers are broadly used to encrypt sensitive data at fast speeds [3]. Some researchers have done some work for the application of stream ciphers on modern communication [4 - 6].The basic structure of the stream cipher requires generation of a keystream. This keystream is mixed, using 'Exclusive or' (XOR) with a plaintext to generate the ciphertext at the sender end while at the receiver end, the identical keystream is generated and XOR'ed with ciphertext to recover the plaintext. Thus each development of the stream cipher algorithms must satisfy the basic structure to meet the conditions of the required applications. For this reason stream cipher designers offering new algorithms employ the characteristics of the basic structure to get efficient and secure algorithms with advanced security and speed, benefitting from the latest scientific and technological developments.

Stream ciphers have been designed [7, 8] to have their own security and efficiency features. However, their deduced speed was not high enough to resist security attacks [9 - 12], spurring the development of high-speed stream ciphers [13 - 15]. To assess their resistance to security attacks, a new series of stream ciphers was introduced [16, 17], knowing that the stream ciphers of eSTREM, i.e., the ECRYPT stream cipher project, are weak against security attacks [18, 19].

The nonlinear invertible round function (combiner) is an important cryptographic criterion [20, 21] for encryption and decryption. In conventional stream ciphers, the keystream uses XOR operation in encoding and decoding. However, with a Pseudo Random Number Generator (PRNG), using a Rabbit algorithm mixed with a combiner featuring nonlinear invertible round function instead of XOR, it may be hard to compute the keystream or recover the plaintext, rendering it desirable to assess new stream ciphers using the nonlinear invertible round function instead of XOR.

In this paper we propose a high-speed stream cipher called *BloStream*, which has been designed to be more flexible, fast, random and secure than conventional stream ciphers that use XOR for mixing. We show that the proposed algorithm provides additional security with proper performance, producing a new scheme between the block cipher round and stream cipher system. Then a strong, high-performance PRNG is employed to efficiently produce a long keystream. Finally, we produce random-looking sequences that in some sense are indistinguishable from truly random sequence which can be used for cryptographic applications and employing a larger set of alphabetical elements to increase the space of probabilities. The paper is organized as follows; we investigate the structure of the proposed cipher in section 2 and highlight some advantages and disadvantages of the proposed algorithm in section 3. In section 4 we compare the proposed cipher with others and in section 5 we present security analyses with possible attacks. We end in section 6 with a few conclusions.

2. The Structure of the Proposed Stream Cipher

2.1. Algorithm components

Figure 1 presents the three basic parts of the proposed algorithm: PRNG for keystream generation, combiner for encryption/decryption, and cipher feedback.

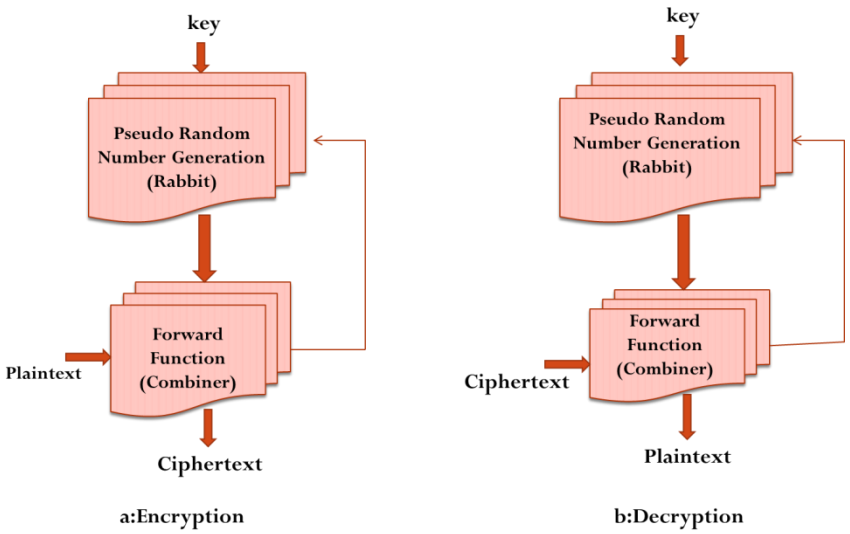


Fig. 1. Graphical illustration of the proposed algorithm.

2.2. Algorithm input

a. The main key (K) is 16-bytes ($k_0, k_1, k_2, \dots, k_{15}$), which are entered to the PRNG to produce the keystreams ($KS1, KS2, KS3, KS4$). Each KS is 16-bytes ($ks_0, ks_1, \dots, ks_{15}$) in each iteration. Using key-dependent S-box in the combiner ensures great difficulties. Not knowing the scope of the S-box increases the number of probabilities faced by the attacker.

b. The plaintext ($P1, P2, P3, P4$) is the plaintext to be encrypted. Each P is 16-bytes ($p_0, p_1, \dots, p_{15}$) entered in the combiner.

c. Text 1 ($P0$) is used only once in the algorithm. It is 16-bytes text ($p_0, p_1, \dots, p_{15}$), utilized as a starting (virtual) plaintext which is input to the encryption/decryption function to produce Text 2, the starting (virtual) ciphertext (which is used in the feedback).

d. Text 2 ($C0$) is the previously generated ciphertext which is input to the PRNG, it contains 16-bytes ($c_0, c_1, \dots, c_{15}$) in each encryption/decryption round.

2.3. Algorithm output

Each ciphertext ($C1, C2, C3, C4$) consists of 16-bytes ($c_0, c_1, \dots, c_{15}$) for each round.

2.4. Algorithm steps

a. *Setup stage*: This stage is performed by both the sender and receiver such that the key is input to the PRNG of the designed algorithm. The internal states of the cipher are initialized by expanding the 128-bit key into the eight 32-bit state variables and eight 32-bit counters. The next state function defined in the Rabbit algorithm [5] is called 3 times only. At the third iteration, 16 bytes (128 bit) are extracted to be assumed as a virtual

keystream $KS0$ which is input with the text 1 ($P0$) to the combiner function to produce the cipher text 2 ($C0$). Each $KS0$,$P0$, 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 key setup is completed. The proposed algorithm now is ready to obtain the first actual keystream ($KS1, KS2, \dots$) to be used in encryption and decryption.

b. *Encryption Stage*: Here, the first 16-byte block of plaintext, $P1 = (P_0, P_1, P_2, \dots . P_{15})$ and 16-byte keystream $KS1 = (k_{S0}, k_{S1}, k_{S2}, \dots k_{S15})$ are input to the combiner (forward round function) and the output is a 16-byte ciphertext $C1 = (C_0, C_1, C_2, \dots . C_{15})$. However, the encryption of the second 16-byte block of plaintext $P2$ requires returning the previous block of ciphertext $C1$ to the PRNG. This previous ciphertext affects the generation of the next 16-byte keystream $KS2$. Then, the second 16-byte keystream $KS2$ and the second 16-byte block of the plaintext $P2$ enter the encryption function to produce the second 16-byte block of ciphertext $C2$. The same mechanism is used to encrypt $P3$ and $P4$ to generate $C3$, $C4$.

c. *Decryption Stage*: When the receiver obtains the ciphertext, $C1$ is input to the combiner with $KS1$ to get $P1$ as a first plaintext block. To get $P2$, $C1$ is input to the PRNG to get $KS2$ which is combined with the $C2$ to produce $P2$. The same process is used to decrypt $C3$, $C4$ into $P3$, $P4$.

2.5. Description of the PRNG

The PRNG used in the proposed cipher is the Rabbit algorithm. Rabbit has been recognized suitable for both software and hardware implementations. The algorithm is initialized by extending the 128-bit key into both the eight 32-bit state variables and the eight 32-bit counters, yielding 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: K0=k[15...0], K1=k[31...16],…K7=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 [5]:

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

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

The coupled system is illustrated in Fig. 2.

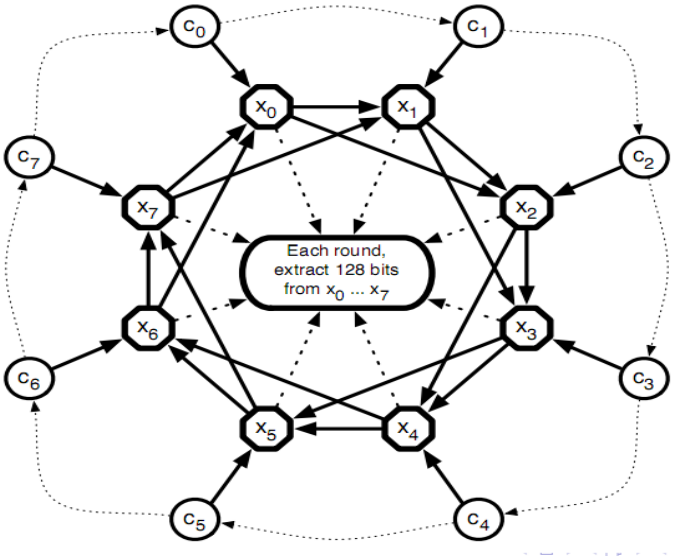


Fig. 2. Graphical illustration of the rabbit algorithm [5].

The chaotic scheme that is used in the PRNG of the proposed cipher can achieve a higher level of complexity than classical binary systems due to their arithmetical properties. The PRNG is a generator with a new type of design. It provides a strong non-linear mixing of the inner state between iterations. As opposed to almost all other designs currently available, it uses neither linear feedback shift registers nor S-boxes. These design decisions have a number of important consequences. The mechanism used in the PRNG of the proposed cipher is depicted in Fig. 3.

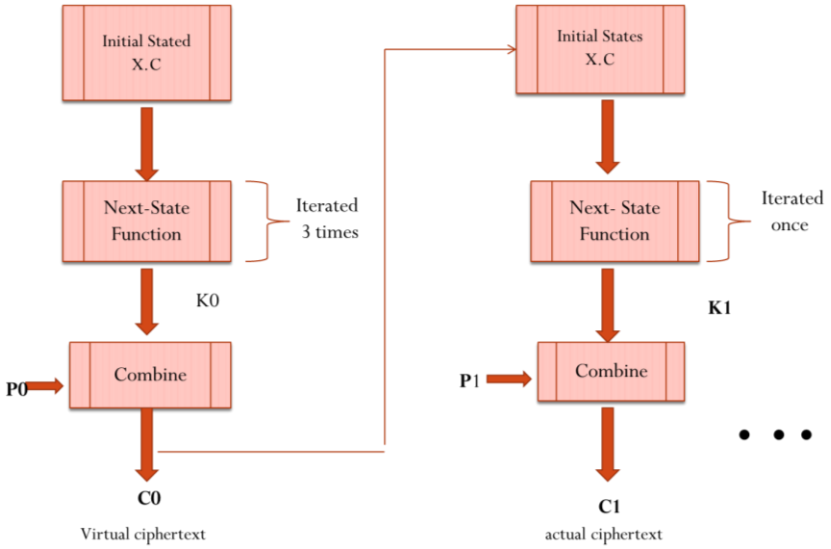


Fig. 3. Mechanism used in PRNG.

A study on a number of stream cipher algorithms found that the Rabbit algorithm was the fastest of the commonly used stream ciphers and was suitable for both hardware and software implementation [21], Table 1.

Table 1. Best encryption speed of some stream ciphers on a Pentium IV [21].

	Algorithm name					
	Trivium	Rabbit	Phelix	Sosemanuk	HC-256	Dargon
Profile	HW	SW/HW	SW	SW	SW	SW
Cycle/byte	8.10	9.46	10.09	10.26	11.12	11.43

2.6. Description of the combiner

The security of the proposed cipher combiner is dependent on having large alphabets in the combining process. The proposed cipher combiner employs a network of simple logic operations. In this new combiner, the substitution-permutation function is designed to replace the dynamic substitution. The new combiner shares more than one character in encryption and decryption. Therefore the combiner gains its cryptographic strength via its round function, in particular using the Rijndael algorithm which forms the basis of the new AES [7, 9]. A modified version of the Rijndael round is at the heart of the combiner function, which operates on 128-bit chunks. The input plaintext block is viewed as 4×4 matrices of bytes called a state, Fig. 4.

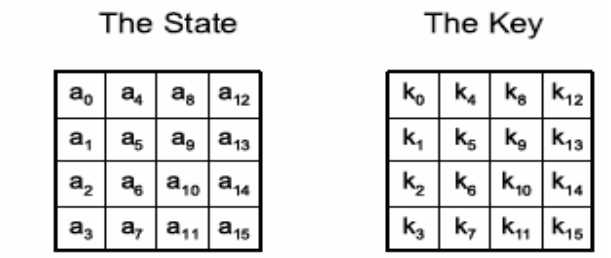


Fig. 4. Structure of the state and the key.

For the new combiner algorithm, the length of the cipher key is 128 bits. Each round consists of four functions: add round keystream, byte substitution, shift rows and dynamic folding transformations.

a. *Add Round Keystream Transformations*: The keystream generated from the PRNG is used byte by byte, from lowest to highest index, so there is no need for keystream array to be in a 2-dimentional form; just use them up and move on. The function Add Round Keystream uses 16 bytes of expanded key every time it is called, Fig. 5. The operation of the inverse Add Round Keystream transformation is simply applied by performing the same forward transformation since Add Round Keystream is its own inverse.

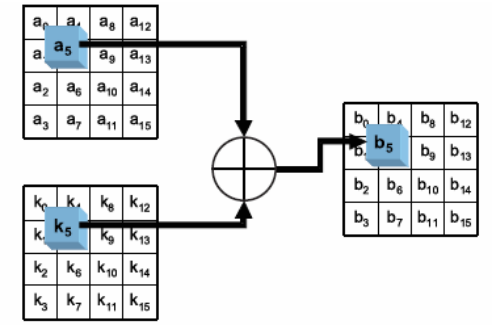


Fig. 5. Add round key stream transformation.

b. *Byte Substitution Transformations*: The Sub Byte transformation is a non-linear byte substitution that acts on every byte of the state in isolation to produce a new byte value, using S-box, Fig. 6.

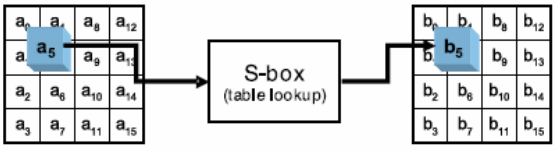


Fig. 6. Subbytes transformation.

The proposed cipher is designed to have restrictions on the amount of ROM available, thus allowing the S-box to use only a small amount of memory with only 256 entries. This S-box is a simple table that contain a permutation of all possible 256 eight-values. The proposed cipher design introduces only one keyed (secret) S-box application as a good shuffler for bytes. S-box [.] is derived from the Rijndale S-box in a key-dependent fashion, Rijndael has been designed to be resistant to known cryptanalytic attacks [9]. Since S-box is private, an attacker cannot know the input to the S-box. A 32-bit key is used in the shuffle process. The secret S-box is initialized as follows:

$$S[i] = SR[\cdots SR[SR[i \oplus k_0] \oplus k_1] \oplus \ldots \oplus k_3], for\ all\ i = 0, \cdots, 255 \qquad (1)$$

where $SR[.]$ is a fixed Rijndael, S-box which is a one-dimensional representation of Table 2. The choice of the S-box provides a much stronger diffusion. Each output bit depends on each input bit of the S-box key. In the decryption process, the inverse Sub Bytes transformation is applied. The same transformation is implemented but with use of the inverse substitution box.

c. *Shift Rows Transformations*: The action of shifting rows is particularly simple; just performing left circular shifts of rows 1, 2 and 3, by amounts of 1, 2, and 3 bytes respectively. Row 0 is not changed, Fig. 7. In the decryption process, the action of inverse shifting rows is particularly simple, just performing right circular shifts of rows 1, 2, and 3, by amounts of 1, 2, and 3 bytes, respectively. Again, Row 0 is not changed.

Table 2. Substitution Table-S-box [*x y*] in hexadecimal.

		y														
		0	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**A**	**B**	**C**	**D**	**E**	**F**
x	**0**	63	7C	77	7B	F2	6B	6F	C5	30	01	67	2B	FE	D7	AB	76
	1	CA	82	C9	7D	FA	59	47	F0	AD	D4	A2	AF	9C	A4	72	C0
	2	B7	FD	93	26	36	3F	F7	CC	34	A5	E5	F1	71	D8	31	15
	3	04	C7	23	C3	18	96	05	9A	07	12	80	E2	EB	27	B2	75
	4	09	83	2C	1A	1B	6E	5A	A0	52	3B	D6	B3	29	E3	2F	84
	5	53	D1	00	ED	20	FC	B1	5B	6A	CB	BE	39	4A	4C	58	CF
	6	D0	EF	AA	FB	43	4D	33	85	45	F9	02	7F	50	3C	9F	A8
	7	51	A3	40	8F	92	9D	38	F5	BC	B6	DA	21	10	FF	F3	D2
	8	CD	0C	13	EC	5F	97	44	17	CA	A7	7E	3D	64	5D	19	73
	9	60	81	4F	DC	22	2A	90	88	46	EE	B8	14	DE	5E	08	D8
	A	E0	32	3A	0A	49	06	24	5C	C2	D3	AC	62	91	95	EA	79
	B	E7	C8	37	6D	8D	D5	4E	A9	6C	56	F4	EA	65	7A	AE	08
	C	BA	78	25	2E	1C	A6	B4	C6	E8	DD	74	1F	4B	BD	8B	8A
	D	70	3E	B5	66	48	03	F6	0E	61	35	57	B9	86	C1	1D	9E
	E	E1	F8	98	11	69	D9	8E	94	9B	1E	87	E9	CE	55	28	DF
	F	8C	A1	89	0D	BF	E6	42	68	41	99	2D	0F	B0	54	BB	16

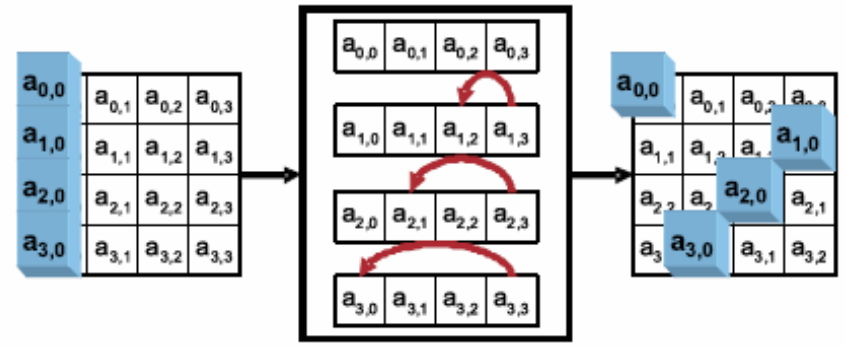


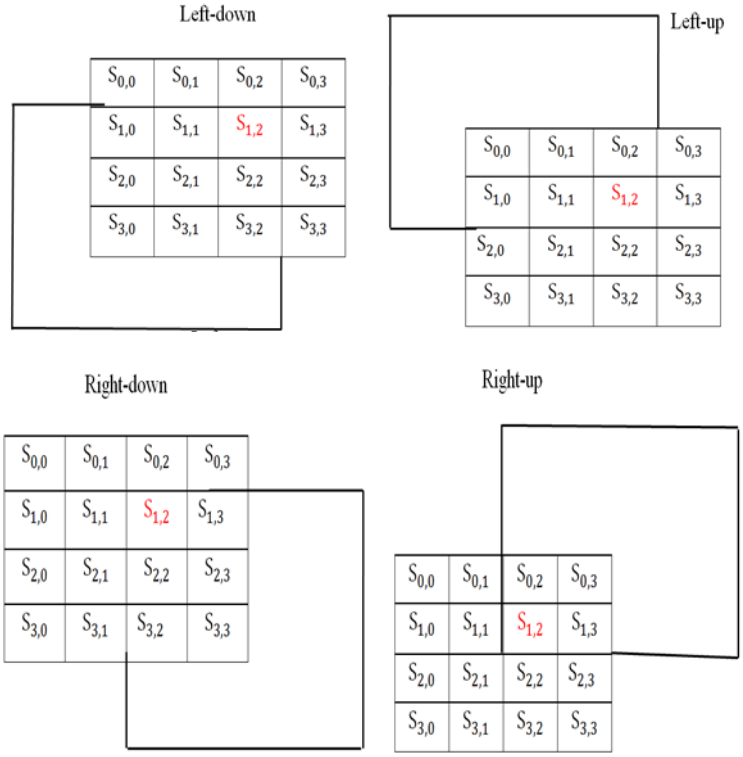
Fig. 7. ShiftRows transformation.

d. *Dynamic Folding Transformations*: In this transformation a complex rotation is applied to the state array by performing a dynamic permutation. In this stage, the elements of the state array are rearranged dynamically to new positions with more probabilities than they have in the normal arrangement. To perform the encryption/decryption process at the combiner, specific values are chosen from the keystream generated by the PRNG. The direction for the new permutation and the position to start are extracted from those

values to implement the dynamic folding. Suppose that the original state prior to dynamic folding is as follows:

$S_{0,0}$	$S_{0,1}$	$S_{0,2}$	$S_{0,3}$
$S_{1,0}$	$S_{1,1}$	$S_{1,2}$	$S_{1,3}$
$S_{2,0}$	$S_{2,1}$	$S_{2,2}$	$S_{2,3}$
$S_{3,0}$	$S_{3,1}$	$S_{3,2}$	$S_{3,3}$

When selecting any position in this (4×4) array, there are 4 directions to rearrange the state array. Therefore, if the row is 1 and the column is 2, then the position to begin folding is (1, 2) with the possibility of using one of the four directions as follows:



According to the input keystream bytes, if the horizontal direction is 0 and the vertical direction is 0 , then the state array will be mutated dynamically to the Right-up direction starting from position (1,2) and the state will be transformed to the following form:

$S_{2,2}$	$S_{2,3}$	$S_{2,0}$	$S_{2,1}$
$S_{3,2}$	$S_{3,3}$	$S_{3,0}$	$S_{3,1}$
$S_{0,2}$	$S_{0,3}$	$S_{0,0}$	$S_{0,1}$
$S_{1,2}$	$S_{1,3}$	$S_{1,0}$	$S_{1,1}$

This transformation is used to further complicate the cryptanalysis. There are 16 locations in the (4×4) state array to start the new permutation according to a specific direction. Instead of a fixed arrangement for the state, there are 64 (4×4) state arrays. Thus, with this increase in difficulty an opponent would not be able to specify the correct order easily. In decryption, inverse dynamic folding is performed using the reverse of the forward dynamic folding, returning the folded state array to the original arrangement. The combiner (round function) is graphically illustrated in Fig. 8. The combiner is implemented efficiently. It consists of four stages: two of permutation and two of substitution. All operations make use of elements over GF(2^8), carried on the byte level. At the start/end of an encryption/decryption, the bytes of the cipher input/output are copied to/from the state array column by column.

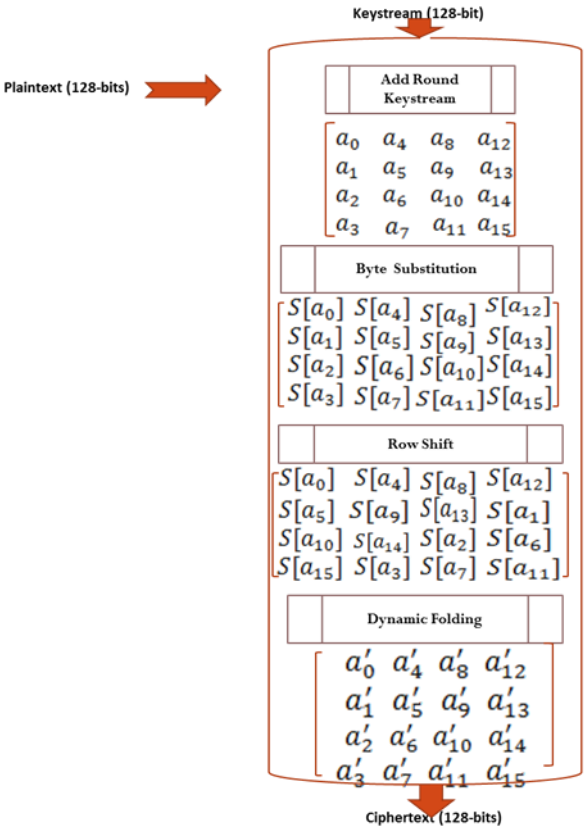


Fig. 8. The graphical illustration of auto-key round combiner.

2.7. The significant combiner properties

The following features were considered in the design of the combiner algorithm:

a. Not only is the first operation XOR, ensuring absolute security to the plaintext, but also the remaining transformations themselves present considerable complexity against the opponent's analysis.

b. Using key-dependent S-box in the combiner ensures great difficulties. Not knowing the scope of the S-box increases the number of probabilities faced by the attacker.
c. Since XOR represents uniquely the first point of meeting between the plaintext with the confusion sequence, the opponent cannot analyse the other parts that transmit the effects of bits among themselves.
d. In the proposed cipher combiner, dynamic folding applies to both rows and columns and involves mutating elements according to specific directions. To enhance effectiveness, it increases the number of probabilities for the cryptanalyst since he is not able to distinguish which 4×4 table to select from among 64 (4×4) arrays which are added to the main body of the combiner; as such, the cryptanalyst is obliged to obtain some key dependent operation before checking the output of each block.

2.8. Cipher feedback

The proposed cipher, producing the ciphertext C_0., makes possible the use of the same key for more encryptions than what the IV offers. This possibility increases randomness in the internal states in addition to lengthening the distance that the plaintext passes through over those of the IV. The data value involved in the ciphertext is feedback into the PRNG internal state by XORing with the four 32-bit counter variables $C_{j,i}$ according to the even/odd condition of the fifth byte of the keystream. In this case, if the result of XORing the 5^{th} byte with 0x01 is 0, then $C_{j,i}[0]$, $C_{j,i}[2]$, $C_{j,i}[4]$,and $C_{j,i}[6]$ are modified with four bytes of the ciphertext. Else $C_{j,i}[1]$, $C_{j,i}[3]$, $C_{j,i}[5]$, and $C_{j,i}[7]$ are modified. The 4^{th}, 8^{th}, 12^{th} and 16^{th} bytes are selected from the previous ciphertext to modify the $C_{j,i}$ counter variable; this will affect the new 16-byte block of the keystream generation in the next iteration to be used in the combiner function. The important advantages of the feedback are to share the round function in the encryption/decryption process and it cannot be isolated. The ciphertext is feedback to the PRNG instead of to the combiner round function because the round function is used merely as an obscurity following XOR. This obscurity reduces the trails for analysing ciphertext. Furthermore, the combined output cannot be used as contributor in the next processes. Figure 9 illustrates the algorithm feedback used in BloStream.

3. Some Advantages and Disadvantages of BloStream Algorithm

3.1. Advantages of the proposed algorithm

The following discussions address some of the advantages and disadvantages of the proposed cipher.

a. The first part of the proposed algorithm gives a considerable strength due to its proper jumping from along the internal state to ensure the required randomness in the generated keystream.
b. The existence of the strong combiner as a second part of the proposed algorithm together with feeding back the ciphertext to the RNG play an effective role in creating a long distance for that jumping (by the effect of

feedback). This is as if it holds a plaintext in its pocket and mixes it well, affecting the next internal states values and rendering it impossible to analyse using the possibilities that one faces when analysing Rabbit RNG alone, since the plaintext has a great effect on the internal state that is used in the generation of keystreams.

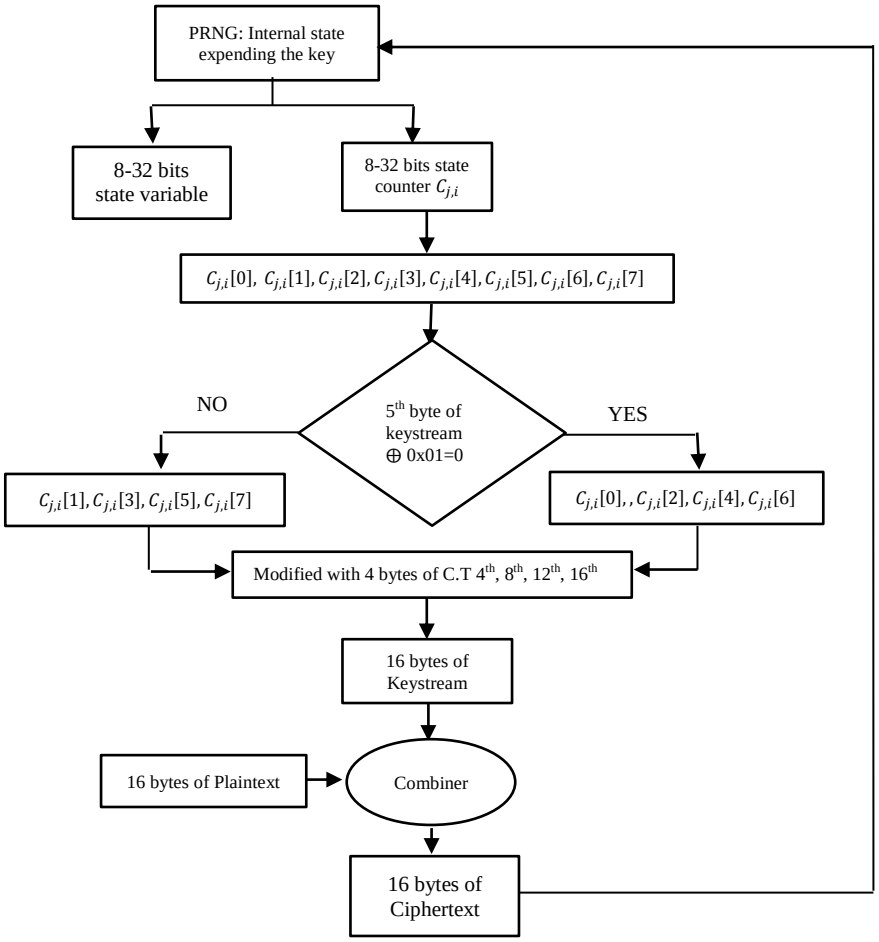


Fig. 9. Cipher feedback algorithm.

c. The jumping and feedback characteristics in the above two points of the proposed algorithm gives it an additional property beyond those of the Rabbit algorithm, i.e., it contains a second part followed by the feedback.

d. Some stream cipher cryptographers use fast but weak RNG as a keystream generator followed by a dynamic substitution combiner such as penknife stream cipher, but the generated keystream cannot be suitable for the most modern developments. Unlike these, the proposed Blostream algorithm uses Rabbit as a high performance RNG, hence producing efficiently random-looking sequences that in some sense are indistinguishable from truly random

sequences. This will remain the basic concept of stream cipher algorithm design in which the RNG is treated as a long way algorithm.

e. In the combiner part, using the round function which by its very nature has special properties, the role of the XOR is not cancelled but is embedded in the round function combiner as the XOR operation is important for the keystream to randomize the plaintext. Also, the XOR is not replaced by a dynamic substitution to ensure a real mixture of the two inputs. However, XOR is supported by a set of transformations with the purpose of complicating the standard XOR combiner. Consequently, the round operation performs a successful combiner function and provides further difficulties in defence against the opponent in addition to sharing its effects in the next encryption/decryption process throughout the use of its resulted cipher as a feedback.

f. In a known and chosen plaintext attack, penetration would seem to be easier if the known-plaintext values were close together. Thus, there is some reason to suspect that an uneven plaintext distribution (which is normal) would have some effect in making the cipher somewhat more vulnerable. But a good value distribution could be enforced by first randomizing the data with a Vernam (XOR) combiner (using a random-number stream) prior to the round function, and the XOR, already contained in the round function, allows the achievement of this randomization in the proposed cipher.

3.2. Disadvantages of the proposed cipher

However, the proposed cipher also has some drawbacks.

a. If an error occurs during a ciphertext transformation, the round function combiner deals with the wrong element, which will affect subsequent parts of the message. This is solved by using a mechanism which smoothed out the frequency differences of alphabetic letters in the message. This is not a very important drawback since most modern communications have their own error-detection and correction capabilities.

b. In the encryption stage of the proposed algorithm, P1 is encrypted to C1 at the time of encryption. Hence, the time required to encrypt C2 is the time needed to generate KS2 plus encryption. In the decryption stage, the time to get P1 is the time of the decryption function, while the time to get P2 is KS2 generation time of decryption function. This means that the required time for encryption and decryption operations is exponentially increased.

c. Using the main secret key, S-box key for shuffling and Text1 in the algorithm may be considered too many inputs to be taken. However, the total number of bytes required in the proposed cipher remains fewer than the number of key bytes used by some stream ciphers that use variable key lengths in their algorithms.

4. Comparison of the Proposed Cipher with Others

This section presents, in three separate tables, the summary of a few computational comparisons between the proposed cipher and a number of more frequently employed ciphers such as RC4, Twofish, and Rijindal with regard to their speed, the number of encryption/decryption operations, and memory

requirements. Table 3 details the properties of the proposed cipher, including speed measurements and memory requirements.

The speed (in Megabyte per second) and memory (in Kilobyte per second) of diverse algorithm ciphers such as Serpent, Mars, Twofish, RC4, Rijndael, and RC6 are included in Table 4. The proposed cipher has the speed of 66 MB/s which is significantly higher than those of other ciphers mentioned in this table.

Table 5 exhibits the number of operations for Key Schedule, Encryption and Decryption processes.

As stated earlier in this paper, the time required for both encryption and decryption in the proposed cipher is exponentially increased; however, the number of operations (i.e., 1798 operations for Key Schedule and 64 for both encryption and decryption) is, as demonstrated in Table 4, still much lower than that for the same operations in RC4 and Chameleon. As such, the proposed cipher is comparatively more efficient than the currently used ciphers in Tables 4 and 5.

Table 3. Speed measurement and memory requirement of the BloStream.

Cipher	***Encryption operation/byte***	***Decryption operation/byte***	***Speed MB/second***	***Memory requirements in Kilobytes***
BloStream	4 operations	4 operations	66 MB/S	2720

Table 4. Comparisons between BloStream and other types of block ciphers.

	Serpent	*Mars*	*Twofish*	*RC6*	*Rijndael*	*BloStream*
Speed MB/S	21.091	27.911	31.411	37.814	61.010	66
Memory KB/S	4438	2737	1076	1139	2902	2720

Table 5. Comparison of RC4, Chameleon and the BloStream.

Algorithm	**Key Schedule without IV**	**Encryption 16-bytes**	**Decryption 16-bytes**
RC4	3328 operations	768 operations	2560 operations
Chameleon	3300 operations	80 operations	112 operations
BloStream	1798 operations	64 operation	64 operation

5. Security Analysis with Possible Attacks

The most important security aspect for the proposed cipher is its resistance against a number of different attacks.

a. *Exhaustive key search*: With a key length of 128 bits, there are 2^{128} possible keys, which are approximately 3.4×10^{38} keys; moreover, Text1 is changed with each communication. Thus, a brute-force attack appears impractical.

b. *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 XOR. Then this data is treated by such transformations of the round function as Sub Bytes, Shift Rows, and dynamic folding. Then the letter frequency statistics will be concealed and the result is a random-like output.

c. *Known Plaintext*: Misuse of any stream cipher, such as reusing keystream, can result in compromising the plaintext without actually revealing any information about the cipher generator itself. When the attacker has some plaintexts and the related ciphertext, then this attack when using XOR 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 breaking the cipher. In the round function, each byte is XOR'ed with the keystream and substituted using a keyed S-box, so that the combination of unknown values of the keystream and the substitutions using S-box deter the attackers from determining the plaintext byte. Also, the dynamic folding according to the selected keystream occurs in each iteration, thus forming a more sophisticated state when deciphering the ciphertext byte.

d. *Statistical attacks*: A keystream generator that exhibits basic statistical biases or detectable characteristics is weak. The nonlinearity in the Rabbit PRNG components and the additional nonlinear transformation in the combiner round function serves to erase the statistical weakness in the keystream. The keystream bit sequences in the proposed algorithm have been extensively tested using statistical tests, which have detected no statistical weaknesses. Table 6 shows the results of these tests.

e. *Time-memory trade-off attacks*: This kind of attack can be applied if the state space of the cipher is too small and does not seem to be applicable to the proposed algorithm.

f. *Guess-and-determine attacks*: The basic idea of this kind of attack 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 4×4 state arrays. Thus, this attack is invalid for the proposed algorithm.

g. *Differential analysis*: The basic concept for this kind of attack is that the attacker exploits the characteristics of a known S-box or transformations. Differential analysis cannot be applied to the proposed cipher because the employed tables are "keyed", i.e., initialized by a particular key. This means that the attacker cannot have prior knowledge of a particular table arrangement.

h. *Distinguishing and* correlation *attacks*: The nonlinear combiner of the proposed cipher has been defined to use an essential amount of input, e.g., virtual plaintext and shuffled (key-dependent) S-box, to perform strong transformations and in combination with the nonlinear equations of the PRNG makes this kind of attack inapplicable, Table 6.

As demonstrated in the last column of Table 6, the keystream with different lengths have passed all statistical attacks.

Table 6. Statistical test of PRNG.

Test	256 bit	385-bit	512-bit	Average Value	Comments
Frequency test	0.390625	0.666667	0.575062	3.841	Pass
Poker test	2.463904	3.283115	1.421505	14.6	Pass
Serial test	1.283905	0.845062	0.021514	5.99	Pass
Auto correlation test	Shift 1 : 0.212873	1.053073	0.012230	1.96	Pass
	Shift 2 : 0.807871	0.392299	0.566666		
	Shift 3 : 0.158306	0.017708	0.024066		
	Shift 4 : 0.015236	0.042035	0.031496		
	Shift 5 : 0.045105	0.141217	0.004437		
	Shift 6 : 0.003388	0.062872	0.017786		
	Shift 7 : 0.021722	0.751341	0.218316		

6. Conclusion

This paper proposed the application of a modified stream cipher called *BloStream* whose combining function covers for the weak classical concept of the simple XOR combiner, transforming it into a stronger version suitable for computer cryptography. A complete description of the algorithm, evaluation of its performance in team of security properties, some advantages and disadvantages and comparison of the proposed cipher with similar systems were presented. Because of its simplicity in implementation and its small processer size, *BloStream* is flexible and the security analysis demonstrated that the proposed cipher is fast and highly secure. The proposed cipher depicts some disadvantages, for instance, in case an error occurs during a ciphertext transformation, the round function combiner may address the wrong element; this operation affects subsequent parts of the message. However, such a disadvantage is rather trivial, for most modern communication systems possess their own error-detection and correction capabilities. As demonstrated in the computational analysis, *Blostream*, in comparison with other currently applied ciphers such as RC4, Chameleon, and Serpent, is distinguished by its significant speed (e.g., 66 MB/Sec against 45 MB/Sec for RC4 and 21 MB/Sec for Serpent) and lower key schedule and memory requirements (e.g., 1798 operations against 3300 operations for Chameleon and 3328 operations for RC4) in encryption/decryption. Finally, *Blostream* revealed considerable resistance against a number of possible attacks, including brute-force attack, statistical attacks, deferential attacks, distinguishing and correlations, which were applied to the proposed cipher.

7. Recommendation for Future Research

In the following, a number of recommendations for future research are presented.

- It is recommended to use more nonlinearity to appear a promising building block for cryptographic systems with certain advantages over linear feedback shift registers (LFSRs).
- It is possible to slightly modify round function (combiner) of the proposed algorithm to gain additional throughput, using the instructions that are available on many 32-bit processors. With the introduction of instruction sets that implement this computation, the cryptographer has a new set of tools to fast ciphers. The plaintext and the keystream that are input to the round

function can be a 32-bit word instead of bytes. Then the Add round Key and Shift Rows can be employed by a single mixed transformation, for example using the PHT (Pseudo Hadamard Transform) which is used in Turing stream cipher for performance purposes.

References

1. Lu, Y. (2006). *Applied stream cipher in mobile communications*. Unpublished Ph.D. Dissertation. Beijing: Polytechnic University.
2. Rose, G.G; and Hawkes, P. (2003). Turing: a fast stream cipher. *International Workshop on Fast Software Encryption*, 290-306.
3. Federal Information Processing Standards Publication 197 (2001). Announcing the advanced encryption standard (AES).
4. Shujun, L.; Xuanqin, M.; and Yuanlong, C. (2001). Pseudo-random bit generator based on couple chaotic systems and its applications in stream-cipher cryptography. *Proceedings of the Second International Conference on Cryptology in India: Progress in Cryptology*, 316-329.
5. Jenog, K.; Lee, Y.; Sung, J.; and Hong, S. (2010). A note on improved fast correlation attacks on stream ciphers. *IACR Cryptology ePrint Archive*, 201021.
6. Tasheva, A.; Tasheva, Z.; and Nakov, O. (2014). Software stream cipher based on pGSSG generator. *International Journal of Cyber-Security and Digital Forensics* (*IJCSDF*), 3(2), 111-121.
7. Dasgupta, A. (2005). Analysis of different types of attacks on stream ciphers and evaluation and security of stream ciphers. Retrieved October 15, 2016, from http://www.securitydocs.com/library/3235.
8. Maximov, A. (2005). *A new stream cipher "Mir-1"*. PhD Dissertation. Department of Information Technology. Lund University. Sweden. eSTREAM report 2005/017.
9. Boesgaard, M.; Vesterager, M.; Pedersen, T.; Christiansen, J.; and Zenner, E. (2004). The stream cipher rabbit. *CRYPTICO A/S, Denmark*.
10. Ali, A. (2011). Efficient implementation of linearization attacks on F-FCSR-16 Type Key-stream generators. 2011 *7th International Conference on Emerging Technologies (ICET)*, Islamabad, 1-6.
11. Gürkaynak, F.K.; Luethi, P.; Bernold, N.; Blattmann, R.; Goode, V.; Marghitola, M.; Kaeslin, H.; Felber, H.; and Fichtner, W. (2006). Hardware evaluation of eSTREAM Candidates: Achterbahn, Grain, MICKEY, MOSQUITO, SFINKS, Trivium, VEST, ZK-Crypt. Retrieved October 25, 2016, from http://www.ecrypt.eu.org/stream/papersdir/2006/015.pdf.
12. McKague, M.E. (2005). *Design and analysis of RC4-like stream ciphers*. M.Sc. Thesis. University of Waterloo, Canada.
13. Johansson, T.; Lano, J.; and Robshaw, M. (2006). eSTREAM: Stream cipher proposals for ongoing analysis. D.STVL.1. *Information Society Technologies*. Katholieke University Leuven (KUL), Retrieved October 25, 2016, from http://www.ecrypt.eu.org/ecrypt1/documents/D.STVL.1-1.0.pdf
14. Qin, Z.G.; Luo, L.; Jiang, S.Q.; Wang, J.; and Zhou, S. (2008). A middleware design for block cipher seamless connected into stream cipher made. *IEEE*

International Conference on Intelligent Information Hiding and Multimedia Signal Processing. Harbin, China, 64-67.

15. Pichler, F. (2007). A highly nonlinear cellular FSM-combiner for stream ciphers. *Proceedings of the 11th International Conference on Computer Aided Systems Theory*, Las Palmas, Spain.
16. Glandman, B. (2003). A specification for Rijndael, the AES algorithm. Retrieved October 5, 2016, from http://techheap.packetizer.com/cryptography/encryption/spec.v36.pdf.
17. Soliman, H.S.; and Omar, M. (2005). Application of synchronous dynamic encryption system in mobile wireless domains. Montreal. *Proceedings of the 1st ACM International Workshop on Quality of Service and Security in Wireless and Mobile Networks*, Montreal, Quebec, Canada, 24-30.
18. Ali, A. (2011). Efficient implementation of linearisation attacks on F-FCSR-16 type key-stream generators. *2011 7th International Conference on Emerging Technologies* (*ICET*), INSPEC Accession Number: 12317105.
19. Mouha, N.; Velichkov, V.; De Cannière, C.; and Preneel, B. (2011). The differential analysis of S-functions. *International Workshop on Selected Areas in Cryptography*, Waterloo, Ontario, Canada, 36-56.
20. Millérioux, G.; and Guillot, P. (2010). Self-synchronizing stream ciphers and dynamical systems: State of the art and open Issues. *International Journal of Bifurcation and Chaos, World Scientific Publishing*, 20(9), 2979-2991.
21. Ekdahl, P. (2003). *On LFSR based stream ciphers - analysis and design*. Ph.D. Dissertation. LUND University, Sweden.
22. Fiskran, A.M.; and Lee, R.B. (2005). Fast parallel table lookups to accelerate symmetric–key cryptography. *International Conference on Information Technology: Coding and Computing* (*ITCC* 2005), INSPEC Accession Number: 8530591.