International Journal of Computational Intelligence Research
ISSN 0973-1873 Volume 13, Number 6 (2017), pp. 1431-1440

http://www.ripublication.com

An Improved Version of SAFER+ Algorithm

Dr. Manish Shrivastava
[1]*Department of Computer Science & engineering, Institute of Technology, GuruGhasidas University, Bilaspur, India.*

Ranbir Sinha
[2]*Department of Computer Science & engineering, Institute of Technology, GuruGhasidas University, Bilaspur, India.*

Abstract

In this paper the algorithm and the practical implementations of the modifications or the upgrades made on the existing SAFER+ Algorithm is presented & demonstrated. The SAFER+ Algorithm runs as a cipher for protecting information transferred through bluetooth and other wireless technologies, and exhibits several problems such as pseudo key collisions, differential cryptanalysis, etc. After analyzing the key schedule and the structure of the SAFER+ Algorithm, the authors have made significant changes that has eliminated all the discussed problems. Moreover the proposed algorithm is demonstrated to be better than the existing SAFER+ Algorithm and provides good amount of security. The proposed algorithm takes 16–bytes of plaintext and 16–bytes of key as input, and is a symmetric cipher that is the same key is used for decryption also. The algorithm then converts them to 20–bytes by producing the 4–bytes from pseudo random number generator (PRNG). The explanation of how the algorithm satisfies the Shannon's Principles of Security and Privacy is also quoted.

Keywords: Cryptanalysis · Collisions · SAFER+

1. INTRODUCTION

James Massey proposed non–proprietary, secret–key block–enciphering algorithm, designated as SAFER K (abbreviation for Secure And Fast Encryption Routine with a Key of length 64 bits), at 1993 Cambridge Security Work-

shop on Fast Software Encryption [3]. The SAFER K Algorithm operated on blocks of 64–bit length under the control of 64–bit key. The algorithm is ”Byte–oriented” and ”iterated” cipher where ciphertext is obtained by itera- tively applying the same function to plaintext for some considerable number of rounds. The number of rounds in the algorithm is a minimum of 6 and a maximum of 10 [3],[4]. Massey presented a paper named ”SAFER K-64: One Year Later” depicting the one–year research work on SAFER K-64 and de- fined SAFER K-128 (SAFER with a key length of 128 bits) at the 1994 Leuven Workshop on Cryptographic Algorithms [4].

Walking into the history we see that SAFER K-64 was sponsored by Cylink Corporation of Sunnyvale, USA. SAFER K-64 was kept unpatented since its in- ception and was free for anyone to use. With the widespread use of SAFER K-64 algorithm, the designers of the algorithm at Cylink Corporation started receiving requests about the various demerits and loopholes of the 64–bit ver- sion and this laid the foundation for the design of SAFER K-128 algorithm. The Ministry of Home Affairs, Singapore took the initiative to design a new key scheduling algorithm to be used with the basic SAFER Algorithm [4].

An important factor which lead to the success of SAFER K Algorithm is its byte orientation. With every operation performed being ”Byte–Oriented”, this algorithm suits the implementation of smart cards with 8-bit internal pro- cessors. The smart–card implementation of SAFER K is 2.5 times as faster as fully optimized smart–card implementation of DES Algorithm [4]. In this paper, we investigate certain algebraic properties of the SAFER K Algorithm and demonstrate how these properties are a potential source of cryptographic weaknesses.

This paper is organized as follows. Firstly, a short introduction of SAFER K Algorithm is delivered. Following the introduction, we present the structure of our proposed algorithm. Afterwards we demonstrate the strength of our proposed algorithm over the weaknesses of the existing SAFER+ Algorithm.

2. DESCRIPTION OF THE PROPOSED ALGORITHM

The proposed algorithm is a symmetric block cipher that operates on block size and key size both of 16–bytes and with byte–by–byte operations involved. Firstly, the algorithm takes 16–bytes of plaintext and 16–bytes of key as in- put. The working process of the algorithm is based on 20–bytes of plaintext and key size respectively. The 16–bytes
plaintext block and key block are expanded to 20–bytes by adding 4–bytes of pseudo random numbers generated using Pseudo Random Number Generator (PRNG). Overall structure of the algorithm consists of round transformations

which are iterated r times and the last stage being the output transformation. The authors took the default value of r to be 10. The key–scheduling algorithm converts the inputted key into (2r+1) 20–byte subkeys K_1 , K_2 , ..., K_{2r+1} .

The overall structure of the proposed algorithm is shown in figure 1.

Each round i utilizes two subkeys K_{2i-1} & K_{2i}, and the subkey K_{2r+1} is used in the output transformation. The i^{th} round contains seven basic operations as given in figure 2.

(1) Mixed XOR/Addition Layer

Bytes 1, 4, 5, 8, 11, 14, 17, 20 of the round input are XORed with bytes 1, 4, 5, 8, 11, 14, 17, 20 of subkey K_{2i-1}. Bytes 2, 3, 6, 7, 9, 10, 12, 13, 15, 16, 18, 19 of the round input are added byte–wise (modulo 256) with bytes 2, 3, 6, 7, 9, 10, 12, 13, 15, 16, 18, 19 of subkey K_{2i-1}.

(2) Non Linear Layer

For a byte x, $45^{(x)}$ is defined to be 45^x modulo 257, where x is regarded as a number 0 <= x <= 255, with the convention that $45^{(128)} = 0$. As 257 is prime and 45 is a primitive element modulo 257, this is an invertible function of a byte, and $\log_{45}(.)$ is defined to be its inverse. The $45^{(.)}$ transformation is applied to bytes 1, 4, 5, 8, 11, 14, 17, 20 of the output of the Mixed XOR/Addition layer and the $\log_{45}(.)$ transformation to bytes 2, 3, 6, 7, 9, 10, 12, 13, 15, 16, 18, 19.

(3) Shuffling Bytes

The shuffling bytes (Table 1) defines the permutation of the bytes obtained as the output of the non–linear layer. The shuffling sequence is the arme- nian shuffle extended to 20–bytes by learning the patterns existing in the original armenian shuffle.

The pattern is : 9 12 13 16 20 3 2 7 6 11 17 10 15 14 1 18 8 5 4 19.

This means that the 9^{th} block now becomes the first block, 12^{th} block now becomes the second block and so on.

(4) Non Linear Layer

For a byte x, $45^{(x)}$ is defined to be 45^x modulo 257, where x is regarded as a number 0 <= x <= 255, with the convention that $45^{(128)} = 0$. As 257 is prime and 45 is a primitive element modulo 257, this is an invertible function of a byte, and $\log_{45}(.)$ is defined to be its inverse. The $45^{(.)}$ transformation is applied to bytes 1, 4, 5, 8, 11, 14, 17, 20 of the output of the Mixed XOR/Addition layer and the $\log_{45}(.)$ transformation to bytes 2, 3, 6, 7, 9, 10, 12, 13, 15, 16, 18, 19.

(5) Shuffling Bytes

The shuffling bytes (Table 1) defines the permutation of the bytes obtained as the output of the non–linear layer. The shuffling sequence is the arme- nian

shuffle extended to 20–bytes by learning the patterns existing in the original armenian shuffle.

The pattern is : 9 12 13 16 20 3 2 7 6 11 17 10 15 14 1 18 8 5 4 19.

This means that the 9^{th} block now becomes the first block, 12^{th} block now becomes the second block and so on.

(6) Mixed Addition/XOR Layer

Bytes 1, 4, 5, 8, 11, 14, 17, 20 of the round input are XORed with bytes 1, 4, 5, 8, 11, 14, 17, 20 of subkey K_{2i-1}. Bytes 2, 3, 6, 7, 9, 10, 12, 13, 15, 16, 18, 19 of the round input are added byte–wise (modulo 256) with bytes 2, 3, 6, 7, 9, 10, 12, 13, 15, 16, 18, 19 of subkey K_{2i-1}.

(7) Pseudo Hadamard Transform (PHT) Layer

The proposed algorithm contains 4–point Pseudo Hadamard Transform as the linear transform. Called as 4–PHT, it can be implemented with 6 mod- ular additions. The 4–PHT can be represented as : -

a[n+3]= ((a[n]+a[n+1]+a[n+2]+a[n+3]) Mod 256)

a[n]= ((a[n]+a[n+3]) Mod 256) a[n+1]= ((a[n+1]+a[n+3]) Mod 256) a[n+2]= ((a[n+2]+a[n+3]) Mod 256)

The ultimate objective of this 4–PHT layer is produce the appropriate dif- fusion.

(8) Permutation Box

The permutation box defines the permutation of the bits as given in the Table 2. The table 2 can be read as : the 151^{th} bit now becomes the first bit, 141^{th} bit now becomes the second bit and so on. This box as well as the shuffling bytes box is defined by the authors and is revised many times through Java program until we got the final permutation.

The output of the Permutation Box Layer forms the output of the round function. After r = 10 rounds, the final output is obtained by the appli- cation of Mixed XOR/Addition Layer to the output of r^{th} round with the subkey K_{2r+1}.

Decryption is done by the application of the round functions in reverse.

The decryption structure of the algorithm is shown in figure 6.

3. KEY SCHEDULE OF THE PROPOSED ALGORITHM

Again, the key schedule of SAFER K is Byte–Oriented. A 16–byte Key K is inputted in the proposed algorithm. The algorithm then expands the 16–byte key to 20–byte key by generating 4–bytes by using pseudo random number

Table 1. Shuffle Pattern of Bytes

Byte Position	Byte # Assigned
1	9
2	12
3	13
4	16
5	20
6	3
7	2
8	7
9	6
10	11
11	17
12	10
13	15
14	14
15	1
16	18
17	8
18	5
19	4
20	19

Table 2. Permutation Box

	Col1	Col2	Col3	Col4	Col5	Col6	Col7	Col8
row1	151	141	131	121	111	101	91	81
row2	153	143	133	123	113	103	93	83
row3	155	145	135	125	115	105	95	85
row4	157	147	137	127	117	107	97	87
row5	159	149	139	129	119	109	99	89
row6	152	142	132	122	112	102	92	82
row7	154	144	134	124	114	104	94	84
row8	156	146	136	126	116	106	96	86
row9	158	148	138	128	118	108	98	88
row10	160	150	140	130	120	110	100	90
row11	79	69	59	49	39	29	19	9
row12	77	67	57	47	37	27	17	7
row13	75	65	55	45	35	25	15	5
row14	73	63	53	43	33	23	13	3

row15	71	61	51	41	31	21	11	1
row16	80	70	60	50	40	30	20	10
row17	78	68	58	48	38	28	18	8
row18	76	66	56	46	36	26	16	6
row19	74	64	54	44	34	24	14	4
row20	72	62	52	42	32	22	12	2

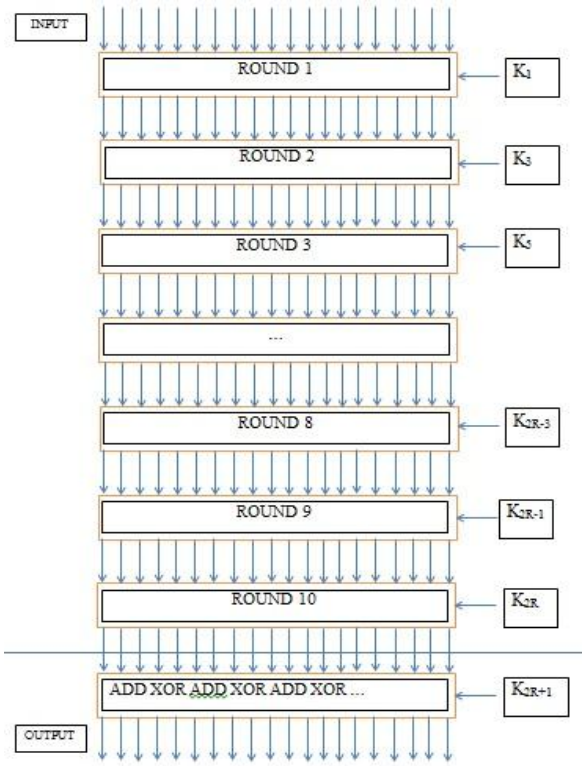


Fig. 1. Overall Structure of Proposed Algorithm

generator. Let the 20–byte key K be K = ($k_{1,1}$, $k_{1,2}$, ..., $k_{1,20}$). The notation for the round key byte j in round i is $K_{i,j}$. The individual round key bytes are derived as follows:

$K_{1,j} = k_{1,j}$ for j = 1, 2, ..., 20.
$k_{i,j} = k_{i-1,j} << 3$.
$K_{i,j} = k_{i,j} + bias[i, j] \bmod 256$
for i = 2, 3, ..., 2r + 1 and j = 1, 2, ..., 20.
<< 3 denotes bitwise rotation by three bits to the left and bias[i, j] = X (X (9i+ j)), where X denotes exponentiation permutation.

4 EXPERIMENTATION & RESULTS

4.1 Pseudo Key Collision

One of the biggest security flaw in SAFER+ Algorithm is the existence of Pseudo Key Collision.

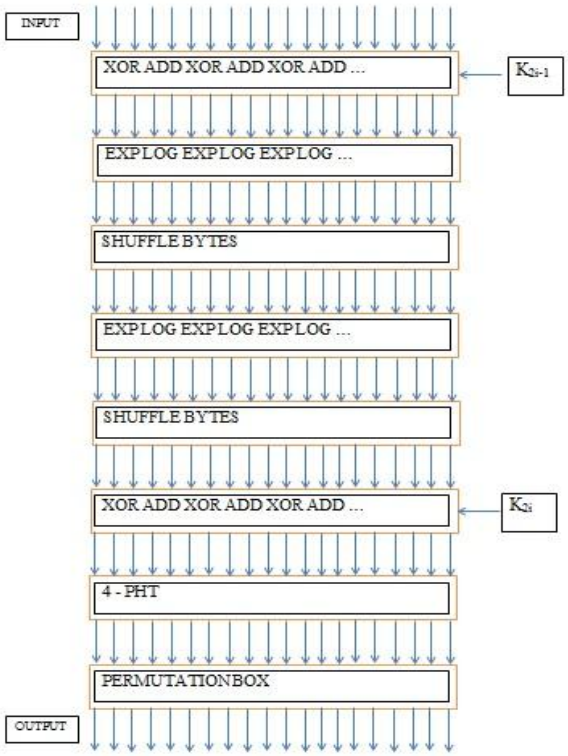


Fig. 2. Round Structure of Proposed Algorithm

The Pseudo Key Collision can be stated as :

Let us assume a situation where a plaintext P is to be encrypted using two keys that differ only in the last byte. Then, the resulting ciphertext from the application of both the keys over the same plaintext also differ in same position."

Paper [1] mentioned the figure 4 as the example of the existence of pseudo key collision in SAFER+ Algorithm.

The above discussed problem is totally removed in our proposed algorithm due to the presence of random bytes which produce the appropriate confusion and diffusion. Figure 5 confirms our claim. Even when the last byte of the two keys differ keeping the plaintext same, the ciphertext is entirely different and nowhere near to each other. Thus the proposed algorithm eliminates the security flaws as demonstrated in the SAFER+ Algorithm.

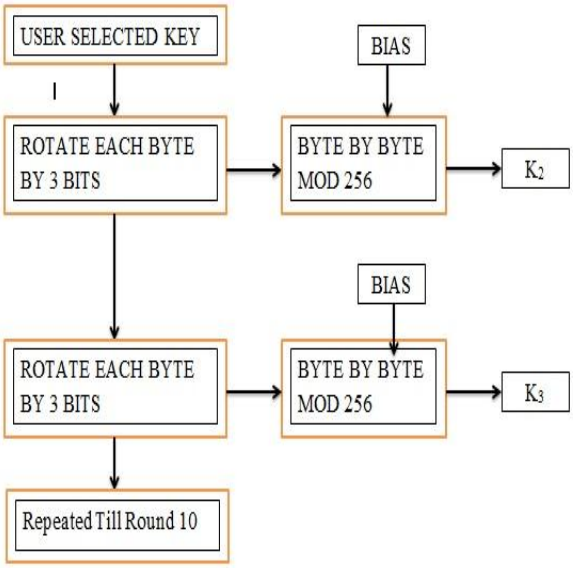


Fig. 3 Key Schedule of Proposed Algorithm

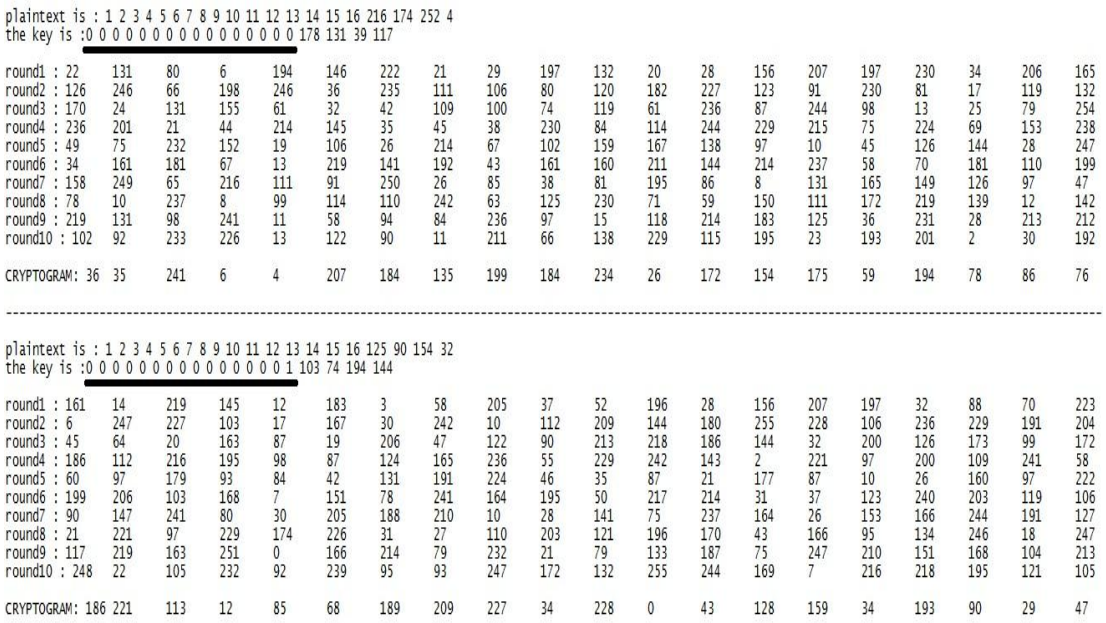
Plaintext	Keys	Ciphertexts
8a 2c 62 a2 a2 81 c1 8c	e0 81 01 85 eb 3b 48 76	ca dd fc f6 30 ac 71 38
8a 2c 62 a2 a2 81 c1 8c	e0 81 01 85 eb 3b 48 bc	ca dd fc f6 30 ac 71 5c
50 1c 7a 44 39 63 f7 8c	e0 81 01 85 eb 3b 48 76	6a 7d db 51 44 89 5a f7
50 1c 7a 44 39 63 f7 8c	e0 81 01 85 eb 3b 48 bc	6a 7d db 51 44 89 5a 93

Fig. 4 Pseudo Key Collision in SAFER+

4.2 Shannon Theory

The proposed algorithm is designed in accordance with the Shannon's principle of confusion and diffusion for enhancing the security architecture. When a sub– key is combined with the Mixed XOR/Byte Addition Layer, then the whole effect is like a non–linear combination with respect to the transformations in the non–linear and linear layer. This produces the desired confusion enough to make the statistics of the ciphertext depend in a weird way on the statistics of the plaintext.

Not only this, the presence of 4–PHT begets the appropriate diffusion in the algorithm structure. Thus the proposed algorithm satisfies the constraints of Shannon's Principle.



plaintext is : 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 216 174 252 4
the key is :0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 178 131 39 117

round1 :	22	131	80	6	194	146	222	21	29	197	132	20	28	156	207	197	230	34	206	165
round2 :	126	246	66	198	246	36	235	111	106	80	120	182	227	123	91	230	81	17	119	132
round3 :	170	24	131	155	61	32	42	109	100	74	119	61	236	87	244	98	13	25	79	254
round4 :	236	201	21	44	214	145	35	45	38	230	84	114	244	229	215	75	224	69	153	238
round5 :	49	75	232	152	19	106	26	214	67	102	159	167	138	97	10	45	126	144	28	247
round6 :	34	161	181	67	13	219	141	192	43	161	160	211	144	214	237	58	70	181	110	199
round7 :	158	249	65	216	111	91	250	26	85	38	81	195	86	8	131	165	149	126	97	47
round8 :	78	10	237	8	99	114	110	242	63	125	230	71	59	150	111	172	219	139	12	142
round9 :	219	131	98	241	11	58	94	84	236	97	15	118	214	183	125	36	231	28	213	212
round10 :	102	92	233	226	13	122	90	11	211	66	138	229	115	195	23	193	201	2	30	192
CRYPTOGRAM:	36	35	241	6	4	207	184	135	199	184	234	26	172	154	175	59	194	78	86	76

plaintext is : 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 125 90 154 32
the key is :0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 103 74 194 144

round1 :	161	14	219	145	12	183	3	58	205	37	52	196	28	156	207	197	32	88	70	223
round2 :	6	247	227	103	17	167	30	242	10	112	209	144	180	255	228	106	236	229	191	204
round3 :	45	64	20	163	87	19	206	47	122	90	213	218	186	144	32	200	126	173	99	172
round4 :	186	112	216	195	98	87	124	165	236	55	229	242	143	2	221	97	200	109	241	58
round5 :	60	97	179	93	84	42	131	191	224	46	35	87	21	177	87	10	26	160	97	222
round6 :	199	206	103	168	7	151	78	241	164	195	50	217	214	31	37	123	240	203	119	106
round7 :	90	147	241	80	30	205	188	210	10	28	141	75	237	164	26	153	166	244	191	127
round8 :	21	221	97	229	174	226	31	27	110	203	121	196	170	43	166	95	134	246	18	247
round9 :	117	219	163	251	0	166	214	79	232	21	79	133	187	75	247	210	151	168	104	213
round10 :	248	22	105	232	92	239	95	93	247	172	132	255	244	169	7	216	218	195	121	105
CRYPTOGRAM:	186	221	113	12	85	68	189	209	227	34	228	0	43	128	159	34	193	90	29	47

Fig. 5 Pseudo Key Collision does not exist in proposed algorithm

5. CONCLUSION

The proposed algorithm verifies both the confusion and diffusion properties of Shannon Theory, which are the essential contributor of the security features. In today's world one of the most prevalent attacks is the differential cryptanalysis. It can be easily seen that there is appropriate difference between any pair of ciphertext blocks. Thus our proposed algorithm can be considered as a Markov Cipher. Since, any Markov Cipher is resistant to differential cryptanalysis, therefore our proposed algorithm is also resistant to differential cryptanalysis attack. The patterns of bytes in the ciphertext makes it immune to various other attacks which were initially possible on SAFER+ Algorithm.

REFERENCES

[1] Lars R. Knudsen, Advances in Cryptology – CRYPTO 1995. A Key–schedule weakness in SAFER K-64 (LNCS 963, pages 274-286). Springer–Verlag, Berlin.

[2] Lars R. Knudsen and Thomas A. Berson, Fast Software Encryption,Third International Workshop, Cambridge. February 1996. Truncated differentials of SAFER (LNCS 1039 pages 15-26). Springer–Verlag, Berlin.

[3] James L. Massey, Fast Software Encryption – Proc. Cambridge Security Workshop, Cambridge. 1994. SAFER K-64: A byte-oriented block-ciphering algorithm (LNCS 809, pages 1-17). Springer–Verlag, Berlin.

[4] James L. Massey, Fast Software Encryption – Second International Workshop, Leuven. 1995. SAFER K-64: One year later (LNCS 1008, pages 212-241). Springer–Verlag, Berlin.