Modified RC4 Variants and Their Performance Analysis

Poonam Jindal and Shikhar Makkar

Abstract Two modified RC4 variants have been discussed in this paper. First, the weaknesses in the basic RC4 algorithm have been identified and further, the weaknesses have been removed in the two proposed algorithms. The implementation of basic RC4, RC4+ and the proposed variants RC4-1 and RC4-2 has been carried out in both C++ and VHDL. The performance analysis has been done in terms of encryption time, security, and resource usage. From the obtained numerical values, it is found that the proposed algorithms provide better security without increasing the encryption time of the basic RC4 algorithm.

Keywords Execution time · RC4 · Security · Stream cipher

1 Introduction

Information security becomes a critical issue, especially when one is vulnerable to a plethora of cyberattacks. Cryptography provides the techniques to communicate safely in the presence of third parties. Cryptographic primitives are classified as a public key and private key encryption algorithms. Public key encryption algorithms include block ciphers or stream ciphers [1]. Stream ciphers are bit or byte oriented and encrypt the message bit or byte by byte. Block ciphers operate on a block of fixed size and use padding if required. RC4 is one of the most widely used, high speed, and easily implemented stream ciphers. It is used in the WEP protocol. However, it has certain weaknesses [2]. KSA+ [3] removed most of them but it incorporates an initialization vector (IV), which is generally not used with RC4. In this paper, two RC4 variants/modifications; RC4-1 and RC4-2 have been proposed. The two RC4 variants keep the basic steps of the algorithm same but provide some changes in the

P. Jindal (✉) · S. Makkar
National Institute of Technology, Kurukshetra, Kurukshetra, India
e-mail: poonamjindal81@nitkkr.ac.in

S. Makkar
e-mail: shikharmakkar@gmail.com

G. Panda et al. (eds.), *Microelectronics, Electromagnetics and Telecommunications*, Lecture Notes in Electrical Engineering 521,
https://doi.org/10.1007/978-981-13-1906-8_38

computations to provide a better security margin, i.e., security is achieved without increasing the encryption time.

1.1 Brief Description of RC4

RC4 runs in two phases [4]. In the first phase, key scheduling is performed using Key Scheduling Algorithm (KSA) and in the second phase, a random byte is generated using Pseudo-Random Generator Algorithm (PRGA). KSA initializes the permutation a key of length between 40 and 256 bytes. After KSA, pseudo-random bits are generated by PRGA. These bits are then used for encryption by combining them with plain text bits using bitwise XOR operation. Encryption and decryption are performed in a similar way. Keystream is generated by using an internal state[S]. Algorithm for conventional RC4 is presented in Algorithm 1.

Algorithm 1. Basic RC4

KSA	PRGA
for i = 0 to N – 1 **do**	i = 0
S[i] = i	j = 0
T[i] = k[i mod keylength]	**loop**
end for	i = (i + 1) mod N
j = 0	j = (j + S[i]) mod N
for i = 0 to N–1 **do**	swap S[i], S[j]
j = (j + S[i] + T[i]) mod N	t = (S[i] + S[j]) mod N
swap S[i], S[j]	key = S[t]
end for	**end loop**

Previous Work. In 1995, the first RC4 weakness concerned with KSA was discovered by Roo's [5]. It was observed that perfectly random distribution has not been obtained after RC4 KSA. It was investigated that no swapping was performed when i = j. In [6], it was discovered that the statistical weaknesses in the keystream. For the first few bytes, the output keystream is strongly nonrandom and makes the key vulnerable to security attacks. More correlations between the RC4 keystream and the key were observed in [7]. Further, the authors in [8] observed that the second output byte of the RC4 was biased towards zero with a probability of 1/128, instead of 1/256. A highly secure available RC4 variant is RC4+ [3], having complex three-layer KSA, taking about three times more encryption time and a complex PRGA, taking about 1.7 times more execution time as compared to basic RC4. In RC4+, the first layer of KSA is the same as conventional RC4. In the second layer, more scrambling is carried out using IVs. In the last and final layer, keystream is scrambled in a zigzag manner, where the keystream bytes take values in the order: 0, 255, 1, 254, 2, 253… Though the algorithm is known to very secure but with increased complexity, in turn, increased execution time and is not desirable in RC4. The elaborated discussion along with the pseudo code of RC4+ is presented in [3, 9]. Various correlations of PRGA have also been presented in [10]. A number of RC4 variants are available in the literature [9], but till date, the algorithm is susceptible to a number of security attacks.

Our Contributions. In this paper, we have modified the basic RC4 algorithm and KSA+ to improve the security without increasing the encryption time. This modification is done by modifying the mathematical computations performed in the algorithm and by adding more randomness, achieved through more scrambling in KSA+. It is evident from the modifications that for every value of i and j, a swapping operation is performed and makes the keystream more random and less vulnerable to attacks.

Structure of the Paper. Section 2 briefly elaborates the Roos' biases and the KSA+ algorithm. Section 3 explains the proposed modifications and the design strategy used to implement the design in VHDL. Section 4 provides a comparative timing analysis of the parameters obtained from implementations in C++ and VHDL. The conclusion is drawn in Sect. 5.

2 Known Weaknesses in RC4

A number of weaknesses were observed in RC4, both in KSA and PRGA. In this paper, we have focused on Roo's biases.

2.1 Roos' Biases

Roos [5] investigated that the RC4 KSA did not exactly yield a perfectly random distribution because of the following reasons:

1. In the initial permutations of KSA, it is highly likely that S[i] = i.
2. If an index has been swapped by i, it is highly likely that it will not be swapped again. If we consider 256 bytes, the probability that an index is chosen at random by j is 1/256 and the probability it will not be chosen is therefore 255/256. The probability that it would not be chosen again during all the iterations of KSA is $(255/256)^{256}$. If i = j, no swapping will be performed and this corresponds to approximately 37% [6].

Because of the above weakness, first, several thousand outputs from the PRGA should be discarded to get the state table into a more even distribution.

3 Proposed RC4

In this paper, two RC4 variants have been proposed RC4-1 and RC4-2 to increase the security of the algorithm. The available literature depicts that RC4 is the simplest available stream cipher without incurring any overhead cost. While keeping in mind, the simplicity of RC4, RC4-P1 has been proposed. It provides security without incurring any overheads in terms of encryption time. RC4+ is a highly secure variant of RC4. But, a number of attacks are possible till date. RC4-P2 has been proposed to overcome the weakness of KSA+ by increasing the depth of scrambling.

3.1 RC4-P1

The proposed RC4-1 is presented in Algorithm 2. The algorithm is a modification over the basic RC4. It attempts to remove the first weakness pointed out by Arthur Roo's which, hinged on one of the observations that no swapping takes place when i = j. In the modified algorithm RC4-1, instead of swapping S[i] and S[j], S[i + j + 1] are swapped. It introduces no extra time but improves the security of the algorithm by removing the byte biases.

Algorithm 2. Pseudo code for RC4-1

KSA-P1	PRGA-P1
for i = 0 to N – 1 do	i = 0
S[i] = i	j = 0
T[i] = k[i mod keylength]	loop
end for	i = (i + 1) mod N
j = 0	j = (j + S[i]) mod N
for i = 0 to N – 1 do	**swap S[i], S[i + j +1]**
j = (j + S[i] + T[i]) mod N	t = (S[i] + S[j]) mod N
swap (S[i], S[i + j +1])	Output Z = S[t]
end for	end loop

3.2 RC4-P2

The proposed RC4-2 is presented in Algorithm 3. The proposed algorithm is a modification over existing RC4+. KSA+ has been modified whereas PRGA is similar to PRGA+. In KSA+ the third and final layer has been modified by performing the scrambling operation in more depth as shown in Eq. (1), where, the index i takes values in the order: 0, 255, 254, 1, 253, 252, 2… In general, if y varies from 0 to N − 1 in steps of 1, then

$$i = \begin{cases} \frac{y}{3},\ y = 0 \mod 3 \\ N - \frac{y+2}{3},\ y = 1 \mod 3 \\ N - \frac{y+1}{3},\ y = 2 \mod 3 \end{cases} \tag{1}$$

Algorithm 3. Pseudo code for RC4-2 (KSA)

```
RC4-2 KSA
(Layer 1 and layer 2 are similar to KSA+)
for y = 0 to N – 1 do if y = 0 mod 3
i = y/3
else do if y = 1 mod 3
i = N – (y + 2)/3
j = j + S[i] + K[i]
swap S[i], S[j]
end for
```

The proposed RC4-2 further enhances the keystream randomness. It prevents the formation of recursive equations linking the keybytes and the permutation bytes. Further as the KSA runs only once in RC4, and the added operation does not affect the performance of the cipher. It introduces no extra time as compared to RC4+ but improves the security of the cipher.

4 Performance Analysis of RC4 and Its Proposed Variants

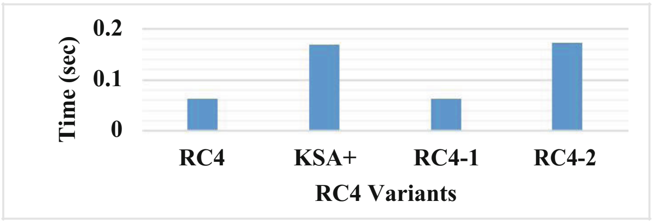
RC4 and its proposed variants RC4-1 and RC4-2 in C++ and VHDL using basic coding techniques have been implemented in this paper. Performance analysis is done in terms of execution time, security analysis, and resource usage by each algorithm.

4.1 Execution Time

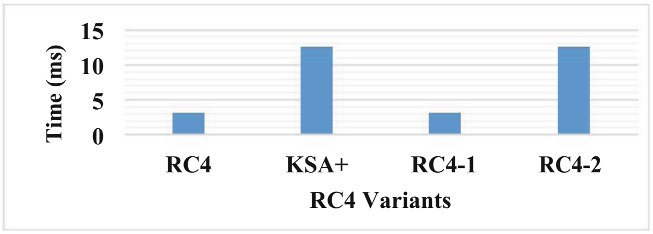
Execution time is the time consumed by the algorithm to convert plaintext into ciphertext. The encryption time for basic RC4, RC4+, RC4-1, and RC4-2 have been analyzed in both C++ and VHDL. For C++ implementation, the time is the total run time for 256 bytes including both KSA and PRGA. For VHDL implementation, the time for PRGA is calculated for 12.5 million bytes. Execution time for all the RC4

Table 1 Timing analysis based on implementation in C++

Algorithm	Time (s) in C++	Time (PRGA (ms), KSA (ns)) in VHDL
RC4	0.062	3.125, 51.25
KSA+	0.168	12.5, 153.65
RC4-1	0.062	3.125, 51.25
RC4-2	0.172	12.5, 153.65



(a)



(b)

Fig. 1 Execution time for different RC4 variants implemented in **a** C++ **b** VHDL

variants is presented in Table 1 and Fig. 1. From the obtained numerical values, ii is found that execution time for RC-1 is similar to that of conventional RC4. Similarly, the time consumed by RC4-2 is similar to that of RC4+. It is worth mentioning that the proposed RC4-1 and RC4-2 are providing better security without incurring any extra time as compared to basic RC4 and RC4+.

4.2 Resource Usage for VHDL Implementations

In VHDL, Algorithmic State Machine (ASM) and Register Transfer Level (RTL) design strategy have been used which, incorporates timing as we move from one state to another in the ASM chart [11, 12]. The control path controls the state transitions and the data path controls various operations that occur in various states. The control

Table 2 Resource usage by RC4 variants

Characteristics	RC4	KSA+	RC4-1	RC4-2
ALMs	14,054	32,465	14,743	32,411
Dedicated logic registers	4344	6808	4344	6808
Combinational ALUTs (number * input size)	(16 * 7) + (10,915 * 6) + (179 * 5) + (515 * 4) + (5552 * 3)	(39 * 7) + (22,963 * 6) + (2199 * 5) + (6220 * 4) + (10,507 * 3)	(16 * 7) + (11,598 * 6) + (430 * 5) + (259 * 4) + (5568 * 3)	(29 * 7) + (20,561 * 6) + (2188 * 5) + (11,541* 4) + (99,113 * 3)

path can be easily implemented as a Finite State Machine (FSM) and the data path can be implemented as a multiplexer (MUX). Since the KSA and PRGA consist of many loops, to store the loop variables and other variables, registers are used to store them. Due to the space constraints, the ASM charts have not been presented in this paper. The hardware resource usage with all the RC4 variants is summarized in Table 2. It is observed that the overall resource utilization with the proposed variants is almost similar to the existing RC4 variants.

4.3 Security Analysis

In this paper, the theoretical security analysis has been performed and presented in Table 3. It is observed that increased layer of operations enhances the security of cipher by removing the identified weaknesses. The obtained results demonstrate that RC4-1 provides comparatively less security as compared to RC4+/RC4-1, but the execution time is very low. Therefore, for the applications like in multimedia, where security is not of much concern but the performance matters, in such scenarios RC4-1 is recommended. Similarly, for the scenarios where, security is of major concern as compared to the network performance such as e-commerce, e-transactions, etc., RC4-2 is recommended to use.

Table 3 Security comparison of RC4 variants

Algorithm	Security
RC4	Security compromised due to several weaknesses
KSA+	Improved security due to additional layers
RC4-1	Increased number of swapping, randomness and removal of one of causes of Roos' biases
RC4-2	Security increased due to increased depth of zigzag scrambling

5 Conclusion

Newer vulnerabilities are being discovered in RC4 every now and then. This raises the need for a better design that is simple, robust and computationally efficient. Our proposed modifications are directed towards the goal of achieving better security without compromising on timing efficiency of the original stream cipher RC4.

In the future, apart from exploring more weaknesses, work can be done by improving the energy efficiency of the stream cipher RC4 when it is implemented in hardware. A trade-off between energy efficiency, complexity, and security can be analyzed.

References

1. Stinson DR (1995) Cryptography: theory and practice, 2005th edn. CRC Press, Boca Raton
2. Mantin I (2005) A practical attack on the fixed RC4 in the WEP mode. In: ASIACRYPT, vol 3788, pp 395–411
3. Maitra S, Paul G (2008) Analysis of RC4 and proposal of additional layers for better security margin. In: INDOCRYPT, vol 5365, pp 27–39
4. Jindal P, Singh B (2015) RC4 encryption-a literature survey. Proc Comput Sci 46:697–705
5. Roos A (1995) A class of weak keys in the RC4 stream cipher
6. Fluhrer S, Mantin I, Shamir A (2001) Weaknesses in the key scheduling algorithm of RC4. In Selected areas in cryptography, vol 2259, pp 1–24
7. Klein A (2008) Attacks on the RC4 stream cipher. Des Codes Crypt 48(3):269–286
8. Mantin I, Shamir A (2001) A practical attack on broadcast RC4. In: International workshop on fast software encryption, pp 152–164. Springer, Berlin, Heidelberg
9. Jindal P, Singh B (2017) Optimization of the security-performance tradeoff in RC4 encryption algorithm. Wirel Pers Commun 92(3):1221–1250
10. Gupta SS, Maitra S, Paul G, Sarkar S (2011) Proof of empirical RC4 biases and new key correlations. In: International workshop on selected areas in cryptography, pp 151–168. Springer, Berlin, Heidelberg
11. Galanis MD, Kitsos P, Kostopoulos G, Sklavos N, Koufopavlou O, Goutis CE (2004) Comparison of the hardware architectures and FPGA implementations of stream ciphers. In: 11th IEEE international conference on electronics, circuits and systems, ICECS 2004, proceedings of the 2004, pp 571–574. IEEE
12. Rane DB, Jyoti UF, Shwetal GR, Chandreshwari PB (2013) Hardware implementation of RC4 stream cipher using VLSI. Int J Electron Commun Soft Comput Sci Eng (IJECSCSE) 70