International Journal of Engineering &Technology, 8 (3) (2019) 270-274

International Journal of Engineering & Technology

Website: www.sciencepubco.com/index.php/IJET

Research paper



Design of RC4 stream cipher for secured communication

G. V. R. Sagar *

Ravindra College of Engineering for Women, Kurnool, Andhra Pradesh, India
**Corresponding author E-mail: nusagar@gmail.com*

Abstract

- RC4 protocol is most popular Cipher in Cryptography. In this paper, we present the study of the efficient design of RC4 stream cipher, and proposing the efficient architecture for cipher. The concept of loop unrolling and pipeline combined to produce the 2 RC4 key stream bytes per clock cycle. The design is compared with the previous proposed paper and to check the how much cycles required for complete the individual KSA and PRGA modules and together the RC4 Stream Cipher. The Design is built using the XILINX 13.2 on Vertex ML605 Evaluation FPGA Board.

Keywords: *Cryptography; RC4; Loop Unrolling; Pipeline; Encryption.*

1. Introduction

Cryptography is the study of technique for secure communication. Cryptography is synonym with Encryption. Cryptography can be broadly classified based on the type of key used in the system: symmetric key using a single key at the sender and recipient end , and public key using two different keys, one is the public key known to everyone and the other is private key only know to the recipient end. The symmetric key system is a class of algorithms whose cryptographic keys are same at the encryption and decryption of cipher. This symmetric key system can either used as either Stream Cipher or Block Cipher. Stream Cipher encrypts the bytes of data at once and Block Cipher encrypts only certain bytes of data and repeats the process till end of data.
Stream Cipher is the Symmetrical Key system, where the Cipher is the combination of the data and pseudorandom keystream. Stream Ciphers are in platforms Software and Hardware named as Software Stream Cipher and Hardware Stream Cipher. RC4 protocol is most popular Stream Cipher used in Software used in network Protocols such as WEP, SSL, WPA. Among all other Stream Ciphers, RC4 is more popular due to its simplicity, ease of implementation. In this paper, we implement the architecture of RC4 with loop unrolling and pipeline which helpful for fast generation of keystream.

1.1. RC4 stream cipher

RC4 also know ARCFOUR or AC4 was designed by Ron Rivest for RSA Security in 1987. The RC acronym for Ron's Code and officially termed as
"Rivest Cipher 4".RC4 with help of S-Box generates pseudorandom group of bits (a key stream), which stores each location of 1byte (for N=256). The secret key for permutations key stream is of size l bytes. The array key K of length N holds the main value and repeated as K[y] = k[y and l], for $0 \leq y \leq N-1$ [7].
RC4 Stream Cipher has two main components, namely Key Scheduling Algorithm (KSA) and Pseudo-Random Generation Algorithm (PRGA). The KSA function is to initialize the key into the S-box and repeat the key to N and generating the S-box and the model of the S-box is of Variable length N, and the arbitrary permutations are performed by the second algorithm PRGA with respectively to N and using two 8-bit index pointer i and j helpful for the operations.

Copyright © 2019 G. V. R. Sagar . This is an open access article distributed under the Creative Commons Attribution License, which permits unrestricted use, distribution, and reproduction in any medium, provided the original work is properly cited.

```
Algorithm 1: KSA
1:   KSA (Key K)
2:        Initialization S ←— {0,1…. , N-1} and j=0
3:        for i=0,1,….N-1
4:           Increment: j ←—{ j + S[i] + K[i] } mod N
5:           Swap:  S[i] ←→ S[j]
6:        end for
7:        return S-box S
8:    end KSA

Algorithm 2: PRGA
1:   PRGA(S-box S)
2:  Initialize indices: i ←—0, j  ←—0
3:  while TRUE do
4:   Increment: i ←—{[i+1] mod N},
                j  ←—{[j+S[i]] mod N}
5:   Swap: S[i] ←→S[j]
6:   output Z←—S{[S[i] + S[j] ] mod N}
7:   end while
8:   end PRGA
```

The output Z of the PRGA is the Keystream called as Randomized key output which is XOR-ed with the Data Input to generate the Cipher Data Output (C = D XOR Z) and is Deciphered by repeating the XOR with Same Keystream to the Ciphered Output (D= C XOR Z).

1.2. Our contribution

We present a new design of the RC4 hardware targeted towards the efficiency in terms of bytes per cycle. The RC4 architecture that proposed by Matthews produces 1 byte per cycle [2]. However, our model not only dependent on the pipeline hardware, it's combined with the idea of loop unrolling in our paper. The loop unrolling is independent of pipelining in RC4. We propose a loop unrolling and hardware pipeline simultaneously. The model produces 2 bytes per cycle without any clock performance.First design of RC4 Design Mattews[1] on hardware,and next its extended with " a 3-cycle-per-byte implementation of RC4 on pipelined hardware" proposed by Kitos et al.[1] in 2003.
The Implementation of the design has been done in Verilog VHDL with the help of XILINX ISE 13.2, Mentor Graphics Tool. We use [6] Vertex ML605 Evaluation FPGA Board for the execution.

2. Implementation of RC4

In this paper we design, Encrypting system of N bits (1 to 256) as shown in the fig1. The two blocks KSA and PRGA are together called as RC4. The output of the RC4 is randomized key and that is XOR-ed with the Data input results the Ciphered Data. This is again, Deciphered with the inputs XOR-ed of Cipher Data and the Randomized key. Let's discuss the RC4 Blocks.
The KSA of the design is to initialize the key input and generate the S-Box according to the required input. In the paper, we design a variable S-Box such that the size of it is double the size of the key input K. Thus the Key input is Stored in S-Box, though the size of S-box is larger than the Key, the s-box is filled with repeated key completely. There ends the initialization and generation of the Key and S-box respectively.
The main architecture PRGA of the RC4 deals with the loop unrolling and pipeline, loop unrolling[4] is the process of increasing the no statements in the loop and reducing the time for loop condition checking time by this we can reduce time required. In the Algorithm of PRGA as two index pointers are considered i and j in order to store the S-array, we use a 8-bit bank registers of N (256) in total. In order to access 256 to 1 bank register using Multiplexer we need control lines to address i1, j1, i2 or j2. .Let us study how to increment the i and j values at each level.

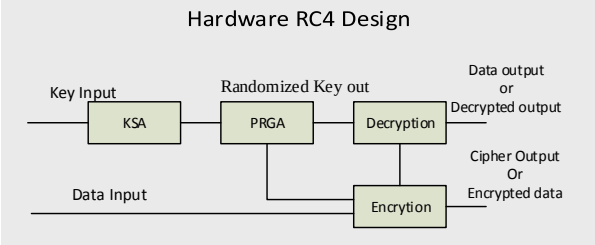


Fig. 1: Hardware RC4 Design.

2.1. Index pointer

In the Algorithm in order to point the particular location of S-Box This are meant for the swapping of the bits in the S-Box.

2.1.1. Calculation of i1 and i2

Using clock signal incrementing i0 by 1 and 2 applied two synchronous 8-bit counters. The counter Values of i1 and i2 is loaded with 00000001 and 00000010 respectively are initialized. Further next the clock pulse is applied to the rest of the fields except for the LSB

positions of the both counters. From circuit 1This results in proper incrimination of i1 to be only odd values 1, 3, 5,7,…., and i2 to be only even values 2,4,6,…, i.e., i1 LSB always 1 and i2 LSB always 0.

Two Consecutive Loops of RC4 Stream Generation

First Loop	Second Loop
$i_1 = i_0 + 1$	$i_2 = i_1 + 1 = i_0 + 2$
$j_1 = j_0 + S_0[i_1]$	$j_2 = j_1 + S_1[i_2] = j_0 + S_0[i_1] + S_1[i_2]$
Swap $S_0[i_1] \leftrightarrow S_0[j_1]$	Swap $S_1[i_2] \leftrightarrow S_1[j_2]$
$Z_1 = S_1[S_0[i_1] + S_0[j_1]]$	$Z_2 = S_2[S_1[i_2] + S_1[j_2]]$

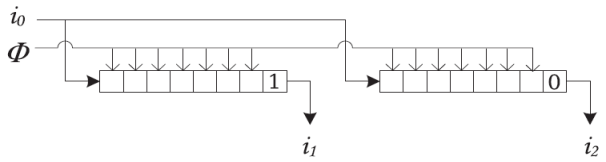


Fig. 2: Computing I1 and I2.

2.1.2. Calculation of j1 and j2

The values are computed using the 2-input parallel adder unit and the every addition is based on modulo addition, in order bring the values in range of N.

$$j_2 = j_0 + S_0[i_1] + S_1[i_2] = \begin{cases} j_0 + S_0[i_1] + S_0[i_2] \text{ if } i_2 \neq j_1, \\ j_0 + S_0[i_1] + S_0[i_1] \text{ if } i_2 = j_1. \end{cases}$$

2.2 Swapping the S values:

The two swap operations in the third row results in one of the following eight possible data transfer requirements among the registers of the S-register
bank, depending on the different possible values of i1; j1; i2, and j2.We have to check if i2 and j2 can be equal to i1 or j1 (we only know that i2 ≠ i1). All the cases in this direction can be listed as in Table 1.

2.3. Calculation of Z1 and Z2

The main idea to get the most out of loop unrolling in RC4 is to completely by pass the generation of S1, and move directly from S0 to S2.

In step 4 of Algorithm 2, we can rewrite the output $Z_1 = S_1[S_1[i_1] + S_1[j_1]] = S_1[S_0[j_1] + S_0[i_1]]$ as

$$Z_1 = \begin{cases} S_2[i_2], & \text{if } S_0[j_1] + S_0[i_1] = j_2, \\ S_2[j_2], & \text{if } S_0[j_1] + S_0[i_1] = i_2, \\ S_2[S_0[j_1] + S_0[i_1]], & \text{otherwise.} \end{cases}$$

Table 1:
Different Cases for the Register-to-Register Transfers in the Swap Operation

#	Condition	Register-to-Register Transfers
1	$i_2 \neq j_1$ & $j_2 \neq i_1$ & $j_2 \neq j_1$	$S_0[i_1] \rightarrow S_0[j_1]$, $S_0[j_1] \rightarrow S_0[i_1]$, $S_0[i_2] \rightarrow S_0[j_2]$, $S_0[j_2] \rightarrow S_0[i_2]$
2	$i_2 \neq j_1$ & $j_2 \neq i_1$ & $j_2 = j_1$	$S_0[i_1] \rightarrow S_0[i_2]$, $S_0[i_2] \rightarrow S_0[j_1] = S_0[j_2]$, $S_0[j_1] \rightarrow S_0[i_1]$
3	$i_2 \neq j_1$ & $j_2 = i_1$ & $j_2 \neq j_1$	$S_0[i_1] \rightarrow S_0[j_1]$, $S_0[i_2] \rightarrow S_0[i_1] = S_0[j_2]$, $S_0[j_1] \rightarrow S_0[i_2]$
4	$i_2 \neq j_1$ & $j_2 = i_1$ & $j_2 = j_1$	$S_0[i_1] \rightarrow S_0[i_2]$, $S_0[i_2] \rightarrow S_0[i_1] = S_0[j_1] = S_0[j_2]$
5	$i_2 = j_1$ & $j_2 \neq i_1$ & $j_2 \neq j_1$	$S_0[i_1] \rightarrow S_0[j_2]$, $S_0[j_2] \rightarrow S_0[j_1] = S_0[i_2]$, $S_0[j_1] \rightarrow S_0[i_1]$
6	$i_2 = j_1$ & $j_2 \neq i_1$ & $j_2 = j_1$	$S_0[i_1] \rightarrow S_0[j_1] = S_0[i_2] = S_0[j_2]$, $S_0[j_1] \rightarrow S_0[i_1]$
7	$i_2 = j_1$ & $j_2 = i_1$ & $j_2 \neq j_1$	Identity permutation, no data transfer.
8	$i_2 = j_1$ & $j_2 = i_1$ & $j_2 = j_1$	Impossible, as it implies $i_1 = i_2 = i_1 + 1$.

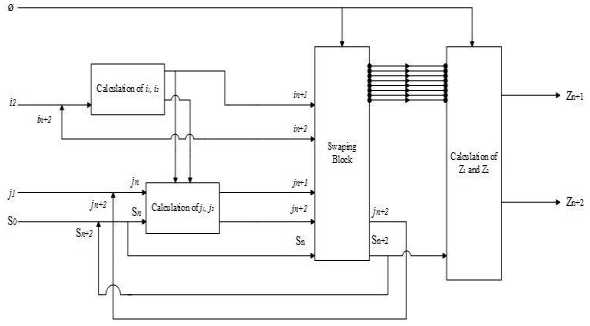


Fig. 3: Calculation of Z1 and Z2.

The value of the Z2 involves addition of S1 [i2], S1 [j2], as formulated below:

$$Z_2 = S_2[S_2[i_2] + S_2[j_2]] = S_2[S_1[j_2] + S_1[i_2]]$$

We consider the values of S1 [i2] and S1 [j2] from S0 states for appropriate conditions:

- $i_2 \neq j_1, j_2 \neq i_1, j_2 \neq j_1 : S_1[i_2] = S_0[i_2], S_1[j_2] = S_0[j_2],$
- $i_2 \neq j_1, j_2 \neq i_1, j_2 = j_1 : S_1[i_2] = S_0[i_2], S_1[j_2] = S_0[i_1],$
- $i_2 \neq j_1, j_2 = i_1, j_2 \neq j_1 : S_1[i_2] = S_0[i_2], S_1[j_2] = S_0[j_1],$

- $i_2 \neq j_1, j_2 = i_1, j_2 = j_1 : S_1[i_2] = S_0[i_2], S_1[j_2] = S_0[j_1],$
- $i_2 = j_1, j_2 \neq i_1, j_2 \neq j_1 : S_1[i_2] = S_0[i_1], S_1[j_2] = S_0[j_2],$
- $i_2 = j_1, j_2 \neq i_1, j_2 = j_1 : S_1[i_2] = S_0[i_1], S_1[j_2] = S_0[i_1],$
- $i_2 = j_1, j_2 = i_1, j_2 \neq j_1 : S_1[i_2] = S_0[i_1], S_1[j_2] = S_0[j_1].$

The Z values are calculated based on the conditions shown above. The circuits are made corresponding with the help of comparators, multiplexers and adders. The final circuit of the Loop Unrolling is shown in the Fig 3.
This Loop Unrolling Architecture performs individual operations and this much number of cycles to complete the Key Stream generation. Lets us combine the Loop Unrolling with the Pipeline.

3. Pipeline data flow

The idea of 2-stage hardware pipeline that of loop unrolling to generate a pipelined RC4 circuit. This is obtained for the case with 2-stage PRGA pipeline and KSA circuit with double iterations per cycle in each case.
The circuit for KSA operates similarly, but has no pipeline feature as the operation happens in a single stage. Here, the increment of indices and swap are done for two consecutive rounds of KSA in a single clock cycle.
In this Pipeline PRGA, it is divided into two modules i,j and swapping all together as one module and output Z as the other module, as shown in fig.

1) Increment of i; j, and Swap in the S-register.
2) Read output byte Z from S-register.

The Output of the RC4 Stream cipher is observed with the help of the waveforms as shown in fig.5. The two inputs are DATA_IN and KEY_IN at the bottom of the waveform. The output is taken with two variables CIPHER_OUT which produces the encrypted key Output and the DATA_OUT which produces decrypted key Output, compared with the DATA_IN value.

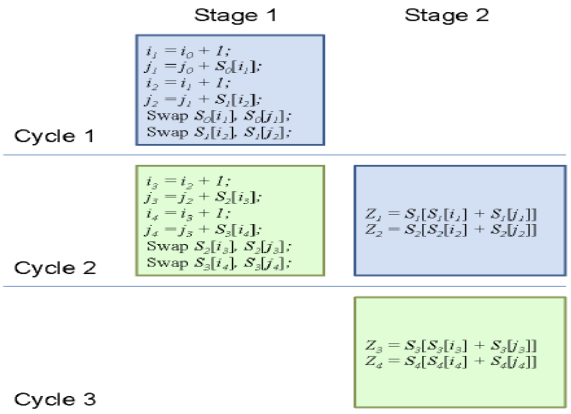


Fig. 4: Pipeline Structure.

4. Results

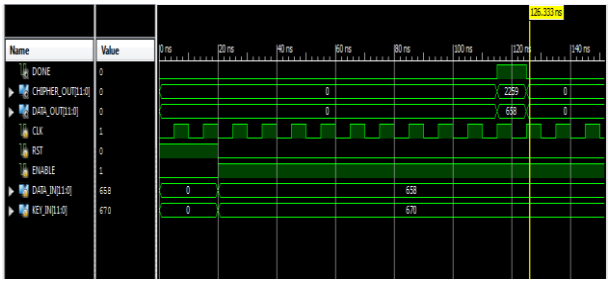


Fig. 5: Simulation Results.

References

[1] Kitos P,G.Kotsoppolus, Sklavos.N, and O. Koufopavlov, "Hardware Implementation of the RC$ Stream Cipher" *Proc. IEEE 46th Midwest Symp. Circuits and Systems.*
[2] D.P. Matthews Jr., "System and Method for a Fast Hardware Implementation of RC4," US Patent Number 6549622, Campbell,CA, 2015.
[3] D.P. Matthews Jr., "Methods and Apparatus for Accelerating ARC4 Processing," US Patent Number 7403615, Morgan Hill, CA, 2016.
[4] "High-Performance Hardware Implementation for RC4 Stream Cipher " Gupta, S.S. Indian Stat. Inst., ASU, Kolkata, India Chattopadhyay, A. ; Sinha, K. ; Maitra, S. ; Sinha, B.P, April 2016.

[5] "Modern Cryptography in Cryptography" in Wiki. Regarding the classifications in cryptography.
[6] Xilinx FPGA Board ML605 Evolution Kit http://www.xilinx.com/products/boards-and-kits/EK-V6-ML605-G.htm
[7] Review on RC4 processing the data http://www.vocal.com/cryptography/rc4-encryption-algoritm/.
[8] Software Performance Results from the eSTREAM Project,eSTREAM, the ECRYPT Stream Cipher Project, http://www.ecrypt.eu.org/stream/perf/results, 2012.
[9] T. Good and M. Benaissa, "Hardware Results for Selected Stream Cipher Candidates," eSTREAM, ECR YPT Stream Cipher Project, SASC, Report 2007/023, 2007.