Implementation of RC4 Stream Cipher Using FPGA

S. C. Wagaj¹, Chetan Bagul², Ramkrushna Chaudhari³

Abstract

In this project work, Implementation of RC4 stream-cipher is proposed. Design of RC4 stream cipher for data Security; RC4 uses a variable length key from 1 to128 bytes to initialize a128-byte array. The array is used for subsequent generation of pseudo-random bytes and then generates a pseudorandom stream, which is XORed with the plaintext/cipher text to give the cipher text/plaintext. The RC4 stream cipher works in two phases. The key setup phase and the pseudorandom key stream generator phase. Both phases must be performed for every new key. The RC4 algorithm will be implemented by FPGA using VHDL software platform.

Keywords

RC4, plaintext, cipher text, cryptography

1. Introduction

Cryptography is a Greek word for "hidden writing". The art and science of transforming (encrypting) information (plaintext) into an intermediate form (cipher text) which secures information in storage or transit. Normally, security occurs as a result of having a vast number of different transformations, as selected by some sort of key. Then, if an opponent acquires some cipher text, a vast number of different plaintext messages presumably could have produced that exact same cipher text, one for each of the possible keys. Message secrecy is one of most important aspect of communication but especially in wireless environment messages are highly insecure and encryption is must in such environment. The various encryption algorithms are available but RC4 encryption algorithm is stream type and can be implemented in hardware and software.


Chetan Bagul. Department of Electronics and Telecommunication Engineering, JSPM’s Rajarshi Shahu College of Engineering, University of Pune, Pune-411033, India.

Ramkrishna Chaudhari, Department of Electronics and Telecommunication Engineering, JSPM’s Rajarshi Shahu College of Engineering, University of Pune, Pune-411033.

RC4 is used for encryption in the wired equivalent privacy (WEP) protocol (part of the IEEE 802.11b wireless LAN security standard), IEEE 802.11 i Lotus Notes, Apple computer's AOCE and Oracle secure SQL[1].

2. RC4 Algorithm

- It uses stream cipher and it can cipher individual units (perhaps bits or bytes) as they occur. It can (but may choose not to) cipher individual data elements immediately, as they arrive. This is a stream cipher signature, and can be identified by analysis of the design. So it takes less time to generate the cipher text.
- RC4 algorithm uses stream cipher that is often used in application where plaintext comes in quantities of unknowable length. Does not need to fill a block, so does not need block padding, and does not need a padding removal structure. [2]
- A particular RC4 algorithm key can be used only once. Encryption is faster than the other algorithms that uses block cipher. The chance of losing the data in wireless transmission is very high, but RC4 algorithm can easily synchronize with the transmission even if the data is lost.
- RC4 algorithm is implemented in software, so the complexity is less and it is cheaper as the software can be easily changed according to the requirements. [3][4]

3. Encryption Using RC4

As it mentioned in general description, the RC4 stream cipher works in two phases. The key setup phase and pseudorandom key stream generator phase. Both phases must be performed for every new key. [5]

Fig.1: Encryption Block Diagram
Fig. 2: Flowchart of RC4

1] Key Setup Phase (For key size 4 bits)
   i. j = (j+ Si + Ki ) mod 4
   ii. Swapping Si with Sj

2] Pseudorandom key stream generator phase 
   (For key size 4 bits)
   1. i = (i + 1) mod 4 , and j = (j + Si ) mod 4
   2. Swapping Si with Sj
   3. t = ( Si + Sj ) mod 4,Random byte St

Simple 4-byte example
S[ ]=[ S0, S1, S2, S3 ] = [0, 1, 2, 3]
K[ ]=[ K0, K1,k2,k3 ] = [1, 7, 1, 7]
Set i = j = 0
First Iteration
   (i = 0, j = 0, S = {0, 1, 2, 3}):
   j = (j + S[ i ] + K[ i ] )mod 4 = (0 + 0 + 1)mod4 = 1
   Swap S[ 0 ] with S[ 1 ]
   S[ ]=[ S0, S1, S2, S3 ] = [1, 0, 2, 3]
Second Iteration
   (i = 1, j = 1, S = {1, 0, 2, 3}):
   j = (j + S[ i ] + K[ i ] )mod4 = (1 + 0 + 7)mod4 = 0
   Swap S[ 1 ] with S[ 0 ]
   S[ ]=[ S0, S1, S2, S3 ] = {0, 1, 2, 3}
Third Iteration
   (i = 2, j = 0, S = {0, 1, 2, 3}):
   j = (j + S[ i ] + K[ i ] ) mod4 = (0 + 2 + 1) mod4= 3
   S[ ]=[ S0, S1, S2, S3 ] = {0, 1, 3, 2}
Fourth Iteration
   (i = 3, j = 3, S = {0, 1, 3, 2}):
   j = (j + S[ i ] + K[ i ] )mod4 = (3 + 2 + 7)mod4 = 0
   Swap S[ 3 ] with S[ 0 ]
   S[ ]=[ S0, S1, S2, S3 ] = [2, 1, 3, 0]

Pseudorandom Key Generation Phase
For this example we use plaintext “HI”

“H”:
i=0, j=0
S[ ]=[ S0, S1, S2, S3 ] = [2, 1, 3, 0]
Because i = i + 1 = 1
   j = (j + Si ) = (1+0)=1, then swap
S1 with s1
New array S[ ]=[ S0, S1, S2, S3 ] = [2, 1, 3, 0]
t = (si+sj)mod4 = ( S1 + S1 )mod4 = 2
   S2 = 3 ( 0000 0011 )
‘H’
0100 1000
XOR 0000 0011
-------------------------------
0100 1011

“J”:
i=1, j=1
S[ ]=[ S0, S1, S2, S3 ] = [2, 1, 3, 0]
Because i = ( i + 1 ) mod4 = 2
j = (j+ S2) = (1+3)mod4=0, then swap
S2 with S0
New array S[ ]=[ S0, S1, S2, S3 ] = [ 3,1,2,0]
t = (si+sj)mod4=( s2+s0)mod4 = 1
   S1= 0 ( 0000 0001 )
‘I’
0100 1001
XOR 0000 0001
-------------------------------
0100 1000

Result Plaintext: 0100 1000 0100 1001
Cipher: 0100 1011 0100 1000

4. Decryption Using RC4

Use the same secret key as during the encryption phase. Generate key stream by running the KSA and PRGA. XOR key stream with the encrypted text to generate the plain text.
Logic is simple : (A xor B) xor B = A
A = Plain Text or Data
B = Key Stream
Using the same secret key used to encrypt generate the RC4 key stream.
Read the encrypted file and XOR every byte of this encrypted stream with the corresponding byte of the
5. Block Diagram

![Block Diagram of project]

**Fig.4:** Block diagram of project

6. Simulation Results

**Encryption of ‘H’**

![Simulation result of Encryption of H]

**Fig.5:** Simulation result of Encryption of H

**Decryption of ‘H’**

![Simulation result of Decryption of H]

**Fig.6:** Simulation result of Decryption of H

7. Hardware Implementation

The PDB file generated is dumped into ProASIC3 A3P250 208FQGA device using FlashPro programming software. ProASIC3, the third-generation family of Micro semi flash FPGAs. Non-volatile flash technology gives ProASIC3 devices the advantage of being a secure, low power, single-chip solution that is live at power-up (LAPU). ProASIC3 is reprogrammable and offers time-to-market benefits at an ASIC-level unit cost. These features enable designers to create high-density systems using existing ASIC or FPGA design flows and tools. [10]

![PROASIC 3 Kit]

**Fig.7:** PROASIC 3 Kit

8. Conclusion

Thus, the proposed system will be implemented for data secrecy which can be useful for variety of applications like defence, satellite TV decoders, business, stock market, internet banking. The system can use variable key length from 1 to 128 bytes providing the flexibility. Hence it will provide the higher security.

**Acknowledgment**

We are heavily indebted to our Principal, Prof.Dr. D.S.Bormane, for his constant inspiration and assistance throughout the paper work. We express our sincere thanks to all the staff of the department and my colleagues who have helped me directly or indirectly in partially completing the paper.
References


Date of birth is 1 Dec 1977 and Birth place Anjangaon solapur Maharashtra. S.C.Wagaj received B.E., M.E degrees in electronics engineering from shivaji University Kolhapur India. From 2004 to till date, he is working as Assistant Professor in JSPM’S Rajasthan Shahu College of Engineering Tathawade Pune Maharashtra .His research interests are network security encoding algorithm optimization and associated VLSI architecture design.

Chetan Bagul received BE degree in Electronics and Telecommunication from University of Pune India. His research interests network security encoding algorithm optimization and associated VLSI architecture design. Ramkrishna Chaudhari received BE degree in Electronics and Telecommunication from University of Pune India. His research interests network security encoding algorithm optimization and associated VLSI architecture design.