ISSN 2347 - 3983
Volume 8. No. 10, October 2020
International Journal of Emerging Trends in Engineering Research
Available Online at http://www.warse.org/IJETER/static/pdf/file/ijeter1478102020.pdf
https://doi.org/10.30534/ijeter/2020/1478102020

An Efficiency Enhanced Scheme for WG Stream Cipher

Mahesh V [1], Mohankumar N [2], Girish Shankar Mishra [3], Arun Kumar M [4], Rakesh Raveendran [5]

1. Asst Professor, EECE, GITAM School of Technology, Bengaluru, India, mvarier@gitam.edu
2. Professor, EECE, GITAM School of Technology, Bengaluru, India, mnagaraj2@gitam.edu
3. Asst Professor, EECE, GITAM School of Technology, Bengaluru, India, gmishra@gitam.edu
4. Asst Professor, EECE, GITAM School of Technology, Bengaluru, India, amanohar@gitam.edu
5. Oracle India Pvt Ltd, Bangalore, rak007.008@gmail.com

ABSTRACT

Presented is a hardware implementation of WG (Welch-Gong) stream Cipher, a crypto algorithm for communication and wireless systems. The Montgomery multiplication algorithm architecture is analyzed based on two approaches: folded and pipelined. Montgomery approach reduces the slice LUT's by 90% power by 10%. Results are evaluated on the basis of utilization factors and performance parameters. Along with key-stream generation, randomness properties are also guaranteed in the finite field.

Keywords WG stream cipher, Montgomery multiplication algorithm, Keystream, Finite field

1. INTRODUCTION

WG cipher is a stream cipher, synchronous in nature with a bit oriented production of key-streams. Binary sequences are produced from the key-stream generator, which consist of Linear feedback shift register (LFSR) and combiner functions. In order to decrypt, an identical key stream is generated and will be provided between transmitter and receiver. This key-stream is XORed with cipher text and it recovers the plain text. Bit by bit encryption and decryption is carried out in this stream cipher.

Stream cipher has applications in wireless and communication systems like 3GPP, LTE, Bluetooth [1], RFID[2] etc. It is resistant to many cryptanalytic attacks like data trade off, time and memory. Concept of the secure stream ciphers is materialized in the eSTREAM project [3] and is implemented in the hardware profile as Word oriented and Bit oriented. Word oriented stream ciphers like ZUC [4] , SNOW 3G [5] have limitations in providing certain key stream properties like linear complexity, auto correlation etc. which are critical in certain communication systems. While bit oriented stream ciphers like A5[6] lack in features including statistical properties, period and linear complexity etc.

2. METHODOLOGY

The WG (Welch-Gong) (29, 11) is implemented based on transformations [7]. A Transformation block along with a pseudo sequence random generator, which is an LFSR, constitutes the main processing block. It consists of shifting, inversion and multiplication operations. Normal basis multiplication [8] is carried out in the transformation block to generate the key stream.

Different combinations of key and initial vector (IV) are the main inputs for this algorithm which ensures high security.

Notations used to define WG cipher operation are:

F_2: finite field with 0 and 1 in Galois field GF(2)

F_2^{29}:field with 2^{29} elements in Galois field and each vector of length 29 bit.

The WG cipher can be used with different length of keys such as 80, 96, 112 and 128. The IV can be of length 32 and 64. Using the same number of bit length for key and IV gives better security which is selected as 128 bit in the current work.

Main blocks of this algorithm are an 11 stage LFSR and a Transformation Block. WG (29,11) shows that each stage of its LFSR consists of 29 bit vector and a total of 11 stages. An FSM and a 4 × 2 multiplexer are used for control operation. Working of this cipher is divided into 3 stages:

a. Loading Phase
b. Initialization Phase
c. Running Phase

The 128 bit key and IV are loaded into the 11 stage LFSR.

We use the notation h(j) for the stages of LFSR where $1 \le j \le 11$, and the key is denoted as $B_{1....b,}$ $1 \le b \le 128$ and IV is denoted as $IV_{1....n,}$ $1 \le m \le 128$.

$h_{1....16}(1)=B_{1....16}$ $h_{17....24}(1)=IV_{1....8}$ $h_{1....8}(2)=B_{17.....24}$
$h_{9....24}(2)=IV_{9...24}$ $h_{1....16}(3)=B_{25...40}$ $h_{17...24}(3)=IV_{25.....32}$
$h_{1....8}(4)=B_{41...48}$ $h_{9....24}(4)=IV_{33...48}$ $h_{1....16}(5)=B_{49.....64}$

$h_{17\ldots24}(5)=IV_{49..56}$ $h_{1\ldots..8}(6)=B_{65\ldots.72}$ $h_{9\ldots.24}(6)=IV_{57.....72}$
$h_{1\ldots.16}(7)=B_{73\ldots.88}$ $h_{17\ldots24}(7)=IV_{73\ldots.80}$ $h_{1\ldots.8}(8)=B_{89\ldots..96}$
$h_{9\ldots.24}(8)=IV_{81\ldots96}$ $h_{1\ldots.16}(9)=B_{97\ldots.112}$ $h_{17\ldots24}(9)=IV_{97\ldots104}$
$h_{1\ldots.8}(10)=B_{113\ldots120}$ $h_{9\ldots.24}(10)=IV_{105\ldots120}$ $h_{1\ldots8}(11)=B_{121\ldots128}$
$h_{17\ldots24}(11)=IV_{121\ldots128}$

In LFSR, zeroes are padded to the remaining bits and loading phase will take 11 clock cycles. The LFSR's feedback polynomial is:

$$p(y) = y^{11} + y^{10} + y^9 + y^6 + y^3 + y + \gamma \quad (1)$$

and the primitive polynomial *q(y)* will generate the F_2^{29}

$$q(y) = y^{29} + y^{28} + y^{24} + y^{21} + y^{20} + y^{19} + y^{18} + y^{17} + y^{14} + y^{12} + y^{11} + y^{10} + y^7 + y^6 + y^4 + y + 1 \quad (2)$$

where β is the root of q*(y)* and so $\gamma = \beta^{464730077}$

h(1),h(2),h(3)……..h(11)F_2^{29} will be the LFSR's internal state.

In initialization phase, key and bit IV are loaded into LFSR, for 22 clock cycles and the initial feedback is carried out . The feedback is XORed with linear feedback and it is fed to the 11^{th} stage of the LFSR so that the LFSR will get updated. A 4 × 2 multiplexer is used to select the key initialization, loading of initial vector and also for linear feedback. Select lines are controlled by the FSM as in Figure 1.

The LFSR is clocked with the key initialization process and the contents of LFSR will get updated. A total of 2^{319-1} combinations will be there, following which the running phase will start and bit by bit key-streams are generated. This is XORed with the plain text to provide the cipher text completing the crypto operation.

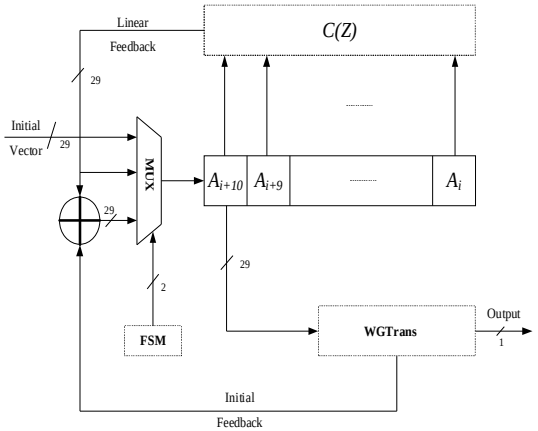

Figure 1: .Block Diagram

In the transformation block $F_2^{29} \rightarrow F_2$ will take place and the computations are done in normal basis [8] with exponentiation by the right cyclic shift.

If y belongs to F_2^{29}, y^{2^i} is obtained by shifting the bits of y cyclically towards right by j times, and in normal basis inverts the bits of the field element for addition with 1.The transformation output is:

$$output = \underline{\oplus}(\triangleright (q_1 \oplus (q_2 \oplus (q_3 \oplus (q_4 \oplus Z))))) \quad (3)$$

where

$$q_1 = (Z \gg 9) \otimes ((Z \gg 19) \otimes Z)$$
$$q_2 = (Z^{-1} \gg 9) \otimes ((Z \gg 19) \otimes Z)$$
$$q_3 = (Z^{-1} \otimes (I \gg 19)) \otimes (Z \gg 10) \quad (4)$$
$$q_4 = (Z \gg 10) \otimes Z$$
$$Z = \triangleright (input)$$

where: $\otimes$ is the normal basis multiplication,$\oplus$ is XOR (bitwise addition), >> right cyclic shift,$\triangleright$ complements all 29 bits, and $\underline{\oplus}$ is addition of 29 bits of *x* over F_2 (XOR) so that a total of 11 clock cycles are taken in loading phase and 22 clock cycles in key initialization phase. It is followed by running phase where each key-stream is generated in single bits.Finite State Machine (FSM) has a connected 2-bit binary and 11-bit one-hot counter.

3. PROPOSED WORK

Inversion and multiplication in the WG transformation block are the most expensive operations leading to large area occupancy and speed. In this contest analysis using the Montgomery Multiplication algorithm [9] inside the transformation block is carried out and the result is encouraging. The finite field arithmetic provides the solutions in the field of cryptography, coding theory etc, where multiplication is the most demanding operation. Main advantage of the finite field arithmetic is the faster results with no loss in accuracy. They are well used in hardware realization, efficiently using VLSI gates. For multiplication, two approaches were proposed [9]: Folded and pipelined one of Montgomery multiplication algorithms. The standard basis is more advantageous compared to the normal basis, because all the polynomials do not have normal basis so that we can't generalize normal basis for every polynomial.

Modular arithmetic is carried out in $GF(2^K)$ fields with the main obstacle that division need a remainder, which is time consuming. The Montgomery multiplication algorithm is performed by bypassing this division so that multiplication in $GF(2^k)$ is a modular operation and it is,

$$d(y) = p(y)o(y) mod\ g(y) \quad (5)$$

so that in this algorithm, instead of *o(x)o(x)mod g(x)*, $p(y)o(y)r^{-1}(y) mod\ g(y)$ and r(y) is pre-computed. The algorithm is as follows:

Input **p**(*y*),o(*y*), g (*y*)
Output $g(y) = p(y)o(y)y^{-k} \bmod d(y)$
1. d(*y*) = 0
2. For j=0 to k-1 **do begin**
3. $d(y) = d(y) + p_j\, o(y)$
4. $d(y) = d(y) + d_0\, g(y)$
5. $c(x) = c(x)/x$
End
6. Return *c(x)*

Therefore multiplier element of this architecture is given as in Figure 2:

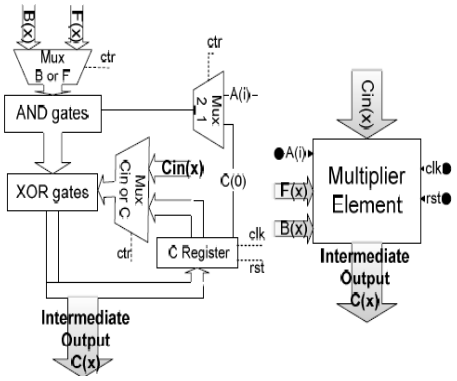

Figure 2: Multiplier Element

In the folded architecture, one input and one output registers are used to store C(x) which is an intermediate result. After completing k-loop cycles, output register stores the result. By using the folded architecture we could achieve very small chip covering area in VLSI implementation. Pipelined approach of this multiplier increases throughput compared to the folded architecture with an increase in the chip covering area. Arrangements of these architectures are shown in the Figures 3(a) & 3(b).

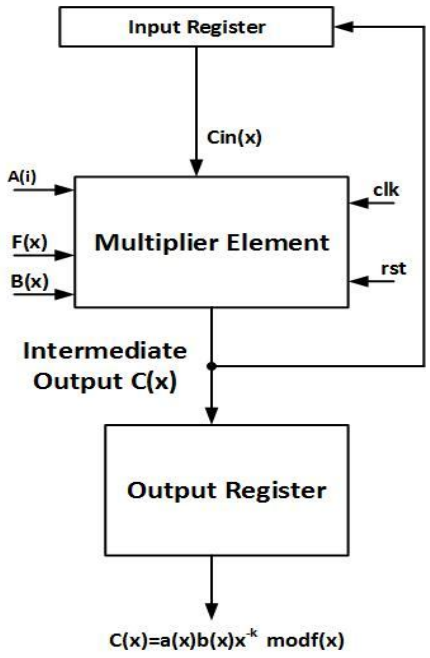

Figure 3(a): Folded approach[9]

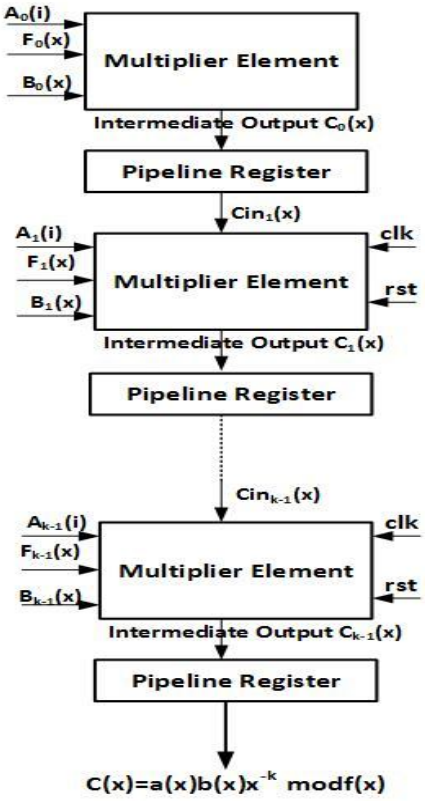

Figure 3(b): Pipelined Approach[9]

4. ANALYSIS AND COMPARISON OF RESULTS

Virtex 6 –XC6VLX130T device with the package FF1156 was used for the implementation of the current architecture. A detailed comparison is made on the basis of device utilization and power consumption as summarized in table 1.

Table 1: Comparison Table

Architecture	Slice LUT'S	Slice registers	LUT-FF pairs	Power (W)	Throughput (Mbps)
WG Basic implementation	4169	446	370	2.985	200
Folded architecture	417	255	254	2.705	227
Pipelined architecture	422	350	350	2.719	267

Slice LUT's in the basic implementation is higher and can be reduced in folded and pipelined approaches. Accordingly power is reduced from 2.985W to 2.705W which is around 10%. Throughput is increased from 200Mbps to 267Mbps, which is 33%.

5. CONCLUSION

The normal basis multiplication in the WG Stream cipher is replaced by Montgomery multiplication algorithm. Three different architectures are compared on the basis of device utilization and power. The results shows that the Montgomery approach is able to reduce the slice LUT's by 90% as compared to the basic implementation and also the power is reduced by 10% in the proposed architecture by preserving the cryptographic properties.

REFERENCES

[1] Bluetooth Special Interest Group. (2010, Jun.). **Adopted Bluetooth Core Specifications, Core Version 4.0**, Kirkland, WA, USA [Online]. Available: https://www.bluetooth.org/

[2] Y. Luo, Q. Chai, G. Gong, and X. Lai, "**A lightweight stream cipher WG-7 for RFID encryption and authentication,"** in *Proc. IEEE Global Telecommunication. Conf.*, Dec. 2010, pp. 1–6

[3] *eSTREAM—The ECRYPT Stream Cipher Project* [Online].Available: http://www.ecrypt.eu.org/stream/

[4] P. Kitsos, N. Sklavos, and A. Skodras, **" An FPGA Implementation of the ZUC Stream Cipher,"** *in Digital System Design (DSD), 2011,14th Euromicro Conference* on, pp. 814–817,IEEE, 2011

[5] Specification of the 3GPP Confidentiality and Integrity Algorithms, UAEA2 & UIA2. Document 2: SNOW 3G Specification, ETSI/SAGE Specification, Version: 1.1Date: 6th September 2006

[6] M. Briceno, I. Goldberg, and D. Wagner, A Pedagogical Implementation of A5, http://www.scard.org, May 1999.

[7] Hayssam El-Razouk, Arash Reyhani-Masoleh, "**New Implementations of WG Stream Cipher**" *IEEE Trans on VLSI* Systems,VOL. 22, No.9, SEPTEMBER 2014, London

[8] H. Fan, D. Liu and Y. Dai, **Two Software Normal Basis Multiplication Algorithms for GF(2n),** Tsinghua Science and Technology, vol. 11, No.3

[9] A.P.Fournaris and O.Koupofavlov "**GF(2k) Multipliers based on Montgomery Multiplication Algorithm**" *International symposium on Circuits & Systems, May 2004.*