ISSN (Print) : 0974-6846
Indian Journal of Science and Technology, *Vol 9(3), DOI: 10.17485/ijst/2016/v9i3/80087, January 2016* ISSN (Online) : 0974-5645

Analysis of Quarter Rounds of Salsa and ChaCha Core and Proposal of an Alternative Design to Maximize Diffusion

Rajeev Sobti* **and Geetha Ganesan**

School of Computer Science and Engineering, Lovely Professional University, Phagwara – 144411, Punjab, India;
sobtirajeev@gmail.com, geetha.15484@lpu.co.in

Abstract

Background/Objectives: Salsa and ChaCha are commonly used encryption primitives. Both Salsa and ChaCha core use Quarter round as its core function. The objective of the paper is to analyze the diffusion property of Quarter round of both these algorithms and propose an alternative design named Modified ChaCha Core (MCC). **Methods:** The Quarter round functions of all these three algorithms are compared using the diffusion matrices that reflect change in output words with a small change in input words. For each algorithm we generated more than a million diffusion matrices depending on the possible permutations of rotations constants used in Quarter round. **Findings:** Results of our experiment reflected that for Salsa and ChaCha core, there are high number of alternative rotation constants that generate more diffusion than the rotation constants prescribed by the authors. The comparison of diffusion matrices of all three competing structures also concluded that quarter round of MCC exhibits more diffusion than Quarter round of Salsa and ChaCha and it does so in lesser operations. **Applications:** MCC core; the design proposed in this paper, may be used to generate stream ciphers or may be used to generate collision resistant compression function for a cryptographic hash algorithm.

Keywords: ChaCha, Diffusion, Modified ChaCha, MCC, Salsa, Stream Ciphers

1. Introduction

Stream ciphers are symmetric key encryption methods in which encryption and decryption is done one symbol (character or bit) at a time. Stream ciphers generate pseudorandom key streams and each digit of plaintext is encrypted one at a time with digit of key stream. Stream ciphers are typically smaller and execute faster than block ciphers. A prominent example of stream cipher is A5/1 cipher which has been used in GSM mobile phone standard for voice encryption. RC4 is another example which has been extensively used for encrypting internet traffic. Lightweight security schemes for Wireless Sensor Networks, based on modified RC4 have also been suggested in literature[1]. Traditionally it was assumed that stream ciphers tend to encrypt more efficiently than block ciphers. Efficient for software-optimized stream ciphers means they take fewer processor cycles to encrypt and efficient for hardware-optimized stream ciphers mean they take fewer gates and smaller chip area than a block cipher for encrypting data at the same data rate. However modern block ciphers have been doing really well in software as well as in hardware. Example is AES[2] in software and PRESENT[3] in hardware.

ECRYPT-European Network of Excellence in Cryptology; a European research initiative launched on 1st February 2004, issued a call for Steam Cipher primitives in November 2004 under the Project eSTREAM[4]. The objective was to identify new stream ciphers suitable for widespread adoption. eSTREAM solicited stream cipher proposals in one of the following two profiles: Profile 1: Stream ciphers for software applications with high throughput requirements. Profile 2: Stream ciphers for hardware applications with restricted resources such as limited storage, gate count, or power consumption.

The eSTREAM project was completed in April 2008. Selected Profile 1 algorithms were: HC-128[5], Rabbit[6], Salsa20/12[7] and SOSEMANUK[8]. The selected Profile 2

* *Author for correspondence*

algorithms were: F-FCSR-Hv2[9], Grain v1[10], Mickey v2[11] and Trivium[12]. Out of these, F-FCSR was originally in eSTREAM profile but was later on removed because of its cryptanalysis.

Our paper concentrates on one of the eSTREAM profile cipher Salsa20 (also known as Snuffle 2005) and its improvement ChaCha[13] (also known as Snuffle 2008) stream cipher. Salsa20/12 and Salsa20/8 are reduced round variants of Salsa20 encryption function (Stream Cipher) and make use of 12 and 8 rounds respectively in place of 20 rounds (10 double rounds) of Salsa20. Salsa20 encryption function uses Salsa20 core which is also known Salsa20 hash function (not to be confused with cryptographic hash function which compresses and is collision-resistant). Salsa20 core does not compress. However, using the core, compression function of cryptographic hash can be generated. Example is SHA-3 final round candidate algorithm BLAKE[14]. ChaCha family is an improvement over Salsa family and have variants like ChaCha8, ChaCha12 and ChaCha20 corresponding to Salsa20/8, Salsa20/12 and Salsa20. Both Salsa core and ChaCha core are based on ARX operations (Arithmetic, Rotation with constant and XOR).

Salsa and ChaCha family arranges the data into a matrix of 4x4 elements where each element is a 32-bit word representing nonce/block number, key and constants. The 20 rounds of Salsa20 core (or ChaCha20 core) hash these values and resultant value is used to encrypt 64-byte plaintext by xoring plaintext with this hashed value (output of Salsa20/ChaCha20 core). This concept is expanded to generate encryption for longer plaintexts. The following explanation as given by Bernstein in Salsa 20 stream cipher[7] illustrates the working. ChaCha20 stream cipher works in the similar fashion.

Let key (**k**) is of 32 bytes. Let '**v**' be an 8-byte sequence. Let '**m**' be an *'l'*-byte sequence for some $l \in \{0,1, \ldots, 2^{70}\}$. The Salsa20 encryption of '**m**' with nonce '**v**' under key **k**, denoted $Salsa20_k(v) \oplus m$ is an '***l***'-byte sequence. Normally '**k**' is a secret key; '**v**' is a nonce, i.e., a unique message number; **'m'** is a plaintext message; and $Salsa20_k(v) \oplus m$ is a cipher text message. $Salsa20_k(v)$ is also generated as 2^{70} byte sequence represented as

$Salsa20_k(v, o)$, $Salsa20_k(v, 1)$ $Salsa20_k(v, 2)$, ..., $Salsa20_k(v, 2^{64}-1)$

Here 'i' [the second argument to $Salsa20_k(v, o)$, $Salsa20_k(v, 1)$ etc is the unique 8-byte sequence $(i_{(0.)}\, i_{(1.)}\, i_{(2.)}\, i_{(3.)}\, i_{(4.)}\, i_{(5.)}\, i_{(6.)}\, i_{(7.)})$ such that $i = i_0 + 2^8 i_1 + 2^{16} i_2 + \cdots + 2^{56} i_7$. Combination of **k, v, i** is mapped to matrix of size 4x 4 (of 32 bits each) and Salsa20 core or ChaCha20 core (or reduced round version) is applied and it gives enough values to be XORed with plaintext and obtain encrypted value.

In this paper we analyze the Salsa core and ChaCha core (used to hash 4x4 matrix representing nonce, key and constants) and study how much diffusion it brings. 20 rounds of Salsa20 core or ChaCha20 core may be viewed as 10 double rounds, where each double round consists of 4 column quarter rounds and 4 row (or diagonal) quarter rounds i.e. operations on each single column/row is termed as Quarter Round. The Quarter rounds of Salsa and ChaCha make use of same number of operations but differ considerably. Quarter rounds of both the functions involve 32-bit additions, 32-bit xor and 32-bit constant rotations. For both Salsa and ChaCha core, author has suggested specific round constants. We have conducted an experiment to analyze diffusion property of Quarter rounds of both the functions and also analyzed how the diffusion changes with change in rotation constants. We have also proposed an alternative design of Quarter round involving same number of operations but exhibit better diffusion properties. Based on this newly constructed quarter round we propose new core named Modified ChaCha Core (MCC) which make use of column and row rounds with changed parameter values for better diffusion.

Organization of the paper: Section 2 introduces the Quarter rounds of the Salsa and ChaCha core and also explains the column and row (or diagonal) rounds of both the cores. In Section 3, we introduce our new design MCC. Section 4, explains the experiment used to analyze the diffusion of all three candidate Quarter rounds (Salsa, ChaCha and Modified ChaCha). The results of experiment are discussed in Section 5 followed by Conclusion and future work in Section 6.

2. Salsa and ChaCha Core

To explain both Salsa and ChaCha core, we are following bottom up approach. First we will introduce the basic primitive-Quarter round, followed by Column and Row/Diagonal round and then Double rounds. All the operations are carried out on 4x4 matrixes where each element is 32-bit word. Let's assume the input matrix 'x' is:

$$\begin{bmatrix} x_0 & x_1 & x_2 & x_3 \\ x_4 & x_5 & x_6 & x_7 \\ x_8 & x_9 & x_{10} & x_{11} \\ x_{12} & x_{13} & x_{14} & x_{15} \end{bmatrix}$$

2.1 Salsa Core

The operations of Salsa core[7] is briefly reproduced below:

Quarter round of Salsa core takes four 32-bit words as input and mixes these words and generates four 32-bit words as output. If $x = (x_0, x_1, x_2, x_3)$ is four word input to $Quarter\ RD_{Salsa}(x)$, then output (y_0, y_1, y_2, y_3) is defined as

$$y_1 = x_1 \oplus ((x_0 + x_2) <<< 7)$$
$$y_2 = x_2 \oplus ((y_1 + x_0) <<< 9)$$
$$y_3 = x_3 \oplus ((y_2 + y_1) <<< 13)$$
$$y_0 = x_0 \oplus ((y_2 + y_2) <<< 18)$$

Quarter round makes use of four 32-bit XORs, four 32-bit Additions and four 32-bit Rotations (left rotation i.e. towards higher bits) with constants. The $y = Quarter\ RD_{Salsa}(x)$ is graphically represented in Figure 1.

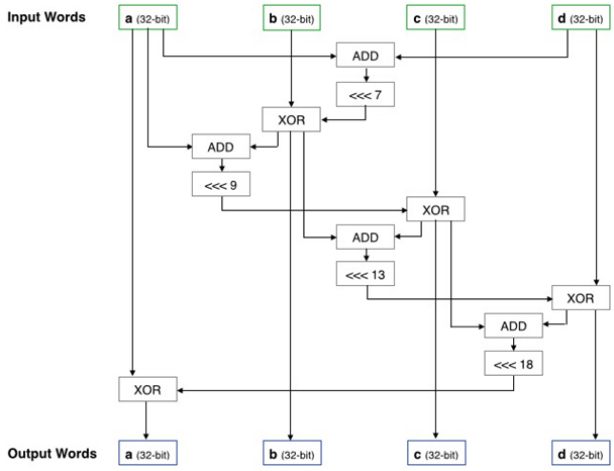

Figure 1 . Quarter round of Salsa Core.

Column round takes 16 words as input and generates 16 words as output. In fact, Column round make use of Quarter rounds. If $x = (x_1, x_2 \ldots x_{12}, x_{14}, x_{15})$ is 16 words input to $ColumnRD_{Salsa}(x)$, then output $y = (y_1, y_2 \ldots y_{12}, y_{14}, y_{15}$ is defined as

$$(y_0, y_4, y_8, y_{12}) = Quarter\ RD_{Salsa}(x_0, x_4, x_8, x_{12})$$
$$(y_5, y_9, y_{12}, y_1) = Quarter\ RD_{Salsa}(x_5, x_9, x_{12}, x_1)$$
$$(y_{10}, y_{14}, y_2, y_6) = Quarter\ RD_{Salsa}(x_{10}, x_{14}, x_2, x_6)$$
$$(y_{15}, y_2, y_7, y_{11}) = Quarter\ RD_{Salsa}(x_{15}, x_2, x_7, x_{11})$$

Row round also takes 16 words as input and generates 16 words as output. Just like Column round, Row round also makes use of four Quarter rounds. If $y = (y_0, y_1, y_2, \ldots y_{13}, y_{14}, y_{15})$ is 16 words input to $ROWRD_{Salsa}(y)$, then output $z = (z_0, z_1, z_2, \ldots z_{13}, z_{14}, z_{15})$ is defined as

$$(z_0, z_1, z_2, z_2) = Quarter\ RD_{Salsa}(y_0, y_1, y_2, y_2)$$
$$(z_5, z_6, z_7, z_4) = Quarter\ RD_{Salsa}(y_5, y_6, y_7, y_4)$$
$$(z_{10}, z_{11}, z_8, z_9) = Quarter\ RD_{Salsa}(y_{10}, y_{11}, y_8, y_9)$$
$$(z_{15}, z_{12}, z_{12}, z_{14}) = Quarter\ RD_{Salsa}(y_{15}, y_{12}, y_{12}, y_{14})$$

In Column (or Row) round, each Column (or Row) of the matrix is used to call Quarter round function but ordering of parameters passed to $Quarter\ RD_{Salsa}$ is different for each column (or row). For example for the first row, parameter passed to $Quarter\ RD_{Salsa}$ is (y_0, y_1, y_2, y_3) but for second row, the parameters are not passed in the same sequence i.e. rather than (y_4, y_5, y_6, y_7), parameters passed are (y_5, y_6, y_7, y_4).

Double round takes 16 words $x = (x_0, x_1, x_2 \ldots x_{12}, x_{14}, x_{15})$ as input and generates 16 words $z = (z_0, z_1, z_2 \ldots z_{13}, z_{14}, z_{15})$ as output. In fact, Double round is a Column round followed by a Row round and is defined below:

$z = Double\ RD_{Salsa}(x) = Row_{RDSalsa}(Column_{RDSalsa}(x))$

Salsa20 core calls 10 Double rounds. Reduced round versions Salsa20/12 or Salsa20/8 calls Double round 6 and 4 times respectively. eSTREAM[4] has Salsa20/12 cipher in its profile that make use of Salsa20/12 core as detailed above.

2.2 ChaCha Core

ChaCha core[13] was proposed by Bernstein as an improvement over Salsa core to increase diffusion using the same number of operations. The details of ChaCha core are given below:

Quarter round of ChaCha, like Salsa's Quarter round, takes four 32-bit words as input and mixes these words and generates four 32-bit words as output. If $x = (x_0, x_1, x_2, x_2)$ is four word input to $QuarterRD_{Salsa}(x)$, then output (y_0, y_1, y_2, y_3) is defined as

$$u_0 = x_0 + x_1 ; u_3 = x_3 \oplus u_0 ; u_3 = u_3 <<<16 ;$$
$$u_2 = x_2 + u_3 ; u_1 = x_1 \oplus u_2 ; u_1 = u_1 <<<12 ;$$
$$y_0 = u_0 + u_1 ; y_3 = u_3 \oplus y_0 ; y_3 = y_3 <<<8 ;$$
$$y_2 = u_2 + y_3 ; y_1 = u_1 \oplus y_2 ; y_1 = y_1 <<<7 ;$$

The $y = QuartRD_{ChaCha}(x)$ is graphically represented in Figure 2.

Close look at Quarter rounds of both Salsa and ChaCha core reflects following major differences:

- ChaCha Quarter round, unlike Salsa Quarter round, gives each input word a chance to affect each output word.
- ChaCha updates each input word twice. For example,

initially x_0 is updated to u_0 and then updated again to y_0. Because of the above two reasons, diffusion by ChaCha Quarter round is much better than Salsa and the same will be evident in the result of our experiment.

- Rotation distances have been changed. Salsa uses 7, 9, 13, 18 as rotation constants while ChaCha uses 16, 12, 8, 7.

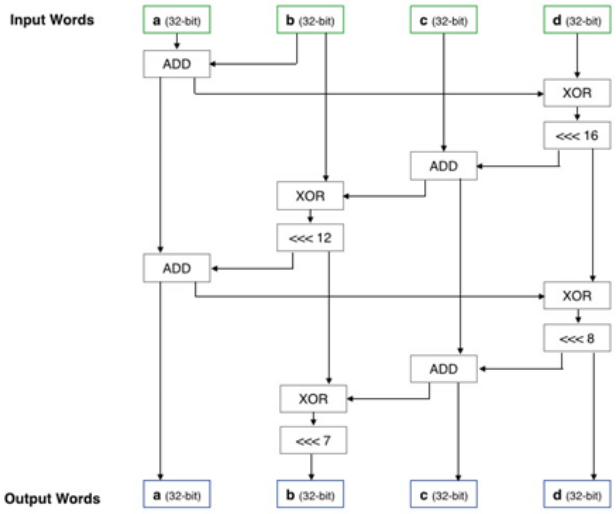

Figure 2 . Quarter round of ChaCha Core.

Column round of ChaCha core is little different from Salsa core in terms of sequence of parameters passed to Quarter rounds. If $x = (x_0, x_1, x_2 \ldots x_{12}, x_{14}, x_{15})$ is 16 words input to $ColumnRD_{ChaCha}(x)$, then output $y = (y_0, y_1, y_2 \ldots y_{12}, y_{14}, y_{15})$ is defined as

$$(y_0, y_4, y_8, y_{12}) = Quarter\ RD_{ChaCha}(x_0, x_4, x_8, x_{12})$$
$$(y_1, y_5, y_9, y_{12}) = Quarter\ RD_{ChaCha}(x_1, x_5, x_9, x_{12})$$
$$(y_2, y_6, y_{10}, y_{14}) = Quarter\ RD_{ChaCha}(x_2, x_6, x_{10}, x_{14})$$
$$(y_2, y_7, y_{11}, y_{15}) = Quarter\ RD_{ChaCha}(x_2, x_7, x_{11}, x_{15})$$

Diagonal round: In place of Row round, ChaCha makes use of diagonal round that calls four quarter rounds; one for each diagonal of input matrix. If $y = (y_0, y_1, y_2 \ldots y_{13}, y_{14}, y_{15})$ is 16 words input to $Diagonal\ RD_{ChaCha}(y)$, then output $z = (z_0, z_1, z_2 \ldots z_{13}, z_{14}, z_{15})$ is defined as

$$(z_0, z_5, z_{10}, z_{15}) = Quarter\ RD_{ChaCha}(y_0, y_5, y_{10}, y_{15})$$
$$(z_1, z_6, z_{11}, z_{12}) = Quarter\ RD_{ChaCha}(y_1, y_6, y_{11}, y_{12})$$
$$(z_2, z_7, z_8, z_{12}) = Quarter\ RD_{ChaCha}(y_2, y_7, y_8, y_{12})$$
$$(z_2, z_4, z_9, z_{14}) = Quarter\ RD_{ChaCha}(y_2, y_4, y_0, y_{14})$$

Double round of ChaCha also takes 16 words $x = (x_0, x_1, x_2 \ldots x_{12}, x_{14}, x_{15})$ as input and generates 16 $z = (z_0, z_1, z_2 \ldots z_{12}, z_{14}, z_{15})$ words as output. Double round of ChaCha is column round followed by a Diagonal round and is defined below:

$z = Double\ RD_{ChaCha}(x) = Diagonal\ RD_{ChaCha}(ColumnRD_{ChaCha}(x))$

ChaCha20 core calls 10 Double rounds. Reduced round versions ChaCha12 or ChaCha8 calls Double round 6 and 4 times respectively. One more difference in ChaCha and Salsa Core that we have not highlighted here is in the mapping of nonce/block number, key and constants to initial matrix 'x'.

3. Modified ChaCha Core (MCC)

A close look at the Quarter round of ChaCha core shown in Figure 2, reflects that the first and third words (word 'a' and 'c') always get updated with 32-bit addition operation, while second and fourth words (word 'b' and 'd') always get updated with an XOR operation followed by Rotation with a constant. In our proposed design (named MCC's quarter round), we break this symmetry and create an alternative design as shown in Figure 3. In our proposed design, all four words are exposed to Addition, XOR and Rotation operation. For example, word 'b' initially gets updated with a 32-bit addition operation and next with a 32-bit XOR and at the last by a 32-bit rotation operation. Similarly, all other words are updated with all three operations. The result of our experiments reflects that this alternative design creates better diffusion than Salsa's and ChaCha's quarter round.

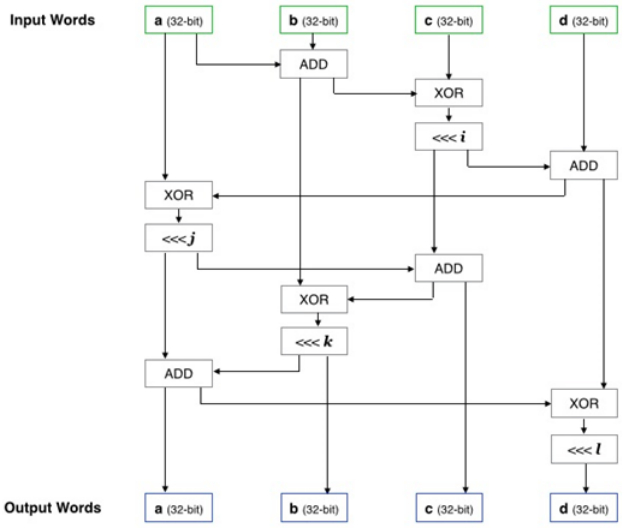

Figure 3 . Quarter round of MCC.

The formal definition of Quarter round as well as Column and Row rounds of MCC is given below:

Quarter round of MCC: If $x = (x_0, x_1, x_2, x_2)$ is four word input to Quarter $RD_{MCO}(x)$, then output (y_0, y_1, y_2, y_3) is defined as

$u_1 = x_1 + x_0 ; u_2 = x_2 \oplus u_1 ; u_2 = u_2 <<< i ;$
$u_3 = x_3 + u_2 ; u_0 = x_0 \oplus u_3 ; u_0 = u_0 <<< j ;$
$y_2 = u_2 + u_0 ; y_1 = u_1 \oplus y_2 ; y_1 = y_1 <<< k ;$
$y_0 = u_0 + y_1 ; y_3 = u_3 \oplus y_0 ; y_3 = y_3 <<< l ;$

In this section we have not specified any particular rotation constants. We will talk about these rotation constants in subsequent section.

Column round of MCC operates in similar fashion like Salsa's Column round. Even the parameters passed are in the same sequence. If $x = (x_0, x_1, x_2 \ldots x_{12}, x_{14}, x_{15})$ is 16 words input to *Column* $RD_{MCC}(x)$, then output $y = (y_0, y_1, y_2 \ldots y_{12}, y_{14}, y_{15})$ is defined as

$(y_0, y_4, y_8, y_{12}) = Quarter\ RD_{MCC}(x_0, x_4, x_8, x_{12})$
$(y_5, y_9, y_{12}, y_1) = Quarter\ RD_{MCC}(x_5, x_9, x_{12}, x_1)$
$(y_{10}, y_{14}, y_2, y_6) = Quarter\ RD_{MCC}(x_{10}, x_{14}, x_2, x_6)$
$(y_{15}, y_2, y_7, y_{11}) = Quarter\ RD_{MCC}(x_{15}, x_2, x_7, x_{11}).$

Row round of MCC is different from both Salsa and ChaCha's Quarter round (ChaCha has diagonal round) in terms of parameters being passed to Quarter rounds. If $y = (y_0, y_1, y_2 \ldots y_{13}, y_{14}, y_{15})$ is 16 words input to $ROW\ RD_{MCC}(y)$, then output $z = (z_0, z_1, z_2 \ldots z_{13}, z_{14}, z_{15})$ is defined as

$(z_1, z_2, z_2, z_0) = Quarter\ RD_{MCC}(y_1, y_2, y_2, y_0)$
$(z_6, z_7, z_4, z_5) = Quarter\ RD_{MCC}(y_6, y_7, y_4, y_5)$
$(z_{11}, z_8, z_9, z_{10}) = Quarter\ RD_{MCC}(y_{11}, y_8, y_9, y_{10})$
$(z_{12}, z_{13}, z_{14}, z_{15}) = Quarter\ RD_{MCC}(y_{11}, y_{13}, y_{14}, y_{15}).$

The reason for passing different sequence of parameters in row and column round is detailed in subsequent section "Results and Discussion".

Double round of MCC is same like Salsa. It takes 16 words $x = (x_0, x_1, x_2 \ldots x_{13}, x_{14}, x_{15})$ input and generates 16 words $z = (z_0, z_1, z_2 \ldots z_{13}, z_{14}, z_{15})$ output. Double round of MCC is Column round followed by a Row round and is defined below:

$z = Double\ RD_{MCC}(x) = ROW\ RD_{MCC}(ColumnRD_{MCC}(x))$

Similar to Salsa and ChaCha, we can define different rounds of MCC also. The number of rounds may be decided based on diffusion factor required i.e. how many full diffusions we need in a cryptographic primitive. MCC is able to generate one full diffusion in two double rounds.

4. Experiment used to Measure the Diffusion Property of Quarter Rounds

Diffusion is considered as one of the main property of the operation of a secure cipher and it refers to the property of a function to quickly spread a small change in the input to maximum possible bits in the output. From the perspective of Quarter round functions, diffusion may be measured in terms of change in output words with a small change in input words. Higher diffusion means a better quarter round function.

Quarter rounds of all three candidate designs have four rotation constants: *[i, j, k, l]*. For Salsa[7], author has defined these rotation constants as [7, 9, 13, 18] and for ChaCha[13], it is defined as [16, 12, 8, 7]. For MCC, we did not specify the exact value of rotation constants in previous section.

To understand the diffusion property of all three candidate quarter rounds (Salsa, ChaCha and MCC), we calculated the matrix as mentioned in Table 1 for all possible values of *i, j, k, and l* where *i, j, k and l* varies from 0 to 31. So we will have 32*32*32*32 = 1,048,576 (more than one million) different matrices for each alternative design.

Each cell like "D_{ab}" of the matrix shown in Table 1 refers average no. of bits modified in word 'b', given a random one-bit difference in input word 'a'. So the first row of the matrix reflects the change (average no. of bits modified) in output words (a to d) with a one-bit difference in input word 'a'. Similarly, second row reflects the change in output words with one-bit difference in input word 'b' and similarly third and fourth row represent changes for input words 'c' and 'd' respectively.

Table 1. Diffusion matrix

OP / IP	a	b	c	d
a	D_{aa}	D_{ab}	D_{ac}	D_{ad}
b	D_{ba}	D_{bb}	D_{bc}	D_{bd}
c	D_{ca}	D_{cb}	D_{cc}	D_{cd}
d	D_{da}	D_{db}	D_{dc}	D_{dd}

For all three alternative designs, we executed the following algorithm to obtain diffusion matrices (as explained above) for all possible rotation constants.

Step 1: Take four words (a, b, c, d) of 32 bits each.

Step 2: Run the following steps (Step 3 to 9) for different values of *i, j, k and l*; each varying from 0 to 31 (so there will be four nested loops and in total there will be 1,048,576 iterations).

Step 3: Generate four 32-bit random values and assign these values to words a, b, c and d respectively.

Step 4: Run the quarter round of concerned design *a′, b′, c′, d′ - Quarter RD* (*a,b,c,d*) with rotation constant as *i, j, k and l.*

[We will call *Quarter* RD_{Salsa} (*a,b,c,d*) or *Quarter* RD_{ChaCha} (*a,b,c,d*) or *Quarter* RD_{MCC} (*a,b,c,d*) depending on the design for which this algorithm is being called].

Step 5: Flip one bit of word 'a' randomly and keep all other words (b, c and d) same and call the quarter round functions of concerned design again.

a″, b″, c″, d″ = QuarterRD ($a_{flipped}$, *b, c, d*) with rotation constant as *i, j, k and l.*

Step 6: Compare *a′, b′, c′, d′* with *a, b, c, d* and find how many bits are different in each word and store it in one dimensional array D_a having four elements $[[D]_{aa}\ D_{ab}\ D_{ac}\ D_{ad}]$ represent change in word 'b' because of one bit flip in word 'a' i.e. difference in *b′* and *b″*. Similarly, D_{ac} represent the change in word 'c' i.e. difference in *c′* and *c″* with one bit flip in word 'a'. Example: If *b′* is 1110011 01110011011100110111001 10 and *b″* is 10111111111100 11010111111101111111 then it means change (no. of bits modified) in word 'b' because of change in one bit of word 'a' is 12. This will be stored in D_{ab}.

Step 7: Repeat step 5 and 6 but this time rather than flipping one bit of 'a', flip one bit of word 'b' and keep all words same. This means we will generate *a″, b″, c″, d″ = Quarter RD* (*a,* $b_{flipped}$, *c, d*) with rotation constant as *i, j, k, l* and will accordingly calculate D_b having four elements $[D_{ba}\ D_{bb}\ D_{bc}\ D_{bd}]$.

Step 8: Similar to step 7, find D_c and D_d by flipping one bit of word 'c' and 'd' respectively. After all this we will have matrix 'D' with first row as $D_a = [D_{aa}\ D_{ab}\ D_{ac}\ D_{ad}]$;
second row as $D_b = [D_{ba}\ D_{bb}\ D_{bc}\ D_{bd}]$; third row as $D_c = [D_{ca}\ D_{cb}\ D_{cc}\ D_{cd}]$ and fourth row as $D_d = [D_{da}\ D_{db}\ D_{dc}\ D_{dd}]$

Step 9: Repeat Step 3 to Step 8, 1000 times and find average of each element of D and after averaging out, we will get diffusion matrix as mentioned in Table 1 for one rotation constants *i, j, k and l.*

Using the above algorithm, the diffusion matrices that we have obtained for Salsa and ChaCha for the rotation constants [7, 9, 13, 18] and [16, 12, 8, 7] respectively 2 and Table 3 clearly reflect that Quarter round of ChaCha performs better diffusion than Salsa. From Salsa diffusion matrix, it is evident that change in word 'b' and 'c' does not bring much change in word 'b' and/or word 'c'.

We executed the above algorithm for all three candidate Quarter round designs (Salsa, ChaCha and MCC). For each design, we obtained 1,048,576 different diffusion matrices, corresponding to all 1,048,576 possible permutations of rotation constants *i, j, k and l* (each varying from 0 to 31). For MCC, experiment was a way to decide the exact rotation distances (constants) to be used. However, for Salsa and ChaCha, the author had already prescribed the rotation constants, but we still calculated diffusion matrices for different rotation distances in case of Salsa and ChaCha also. Objective was to study whether there exist rotation distances that generate better diffusion than prescribed rotation distances of Salsa and ChaCha core.

Table 2. Diffusion matrix of Salsa's quarter round

OP / IP	a	b	c	d
a	12.205	1.955	4.316	6.430
b	7.255	1.0	1.946	4.330
c	4.333	0	1.0	1.958
d	8.784	1.958	2.468	5.649

Mean: 4.0992,

Standard Deviation: 3.1887

OP / IP	a	b	c	d
a	4.047	11.285	9.331	5.989
b	5.987	13.418	10.825	7.794
c	2.47	6.781	4.803	2.47
d	2.399	8.528	6.751	3.399

Mean: 6.6424,

Standard Deviation: 3.2731

5. Results and Discussion

Each diffusion matrix, contains 16 values, all representing diffusions in different words (a, b, c and d). To evaluate all these diffusion matrices (more than 1 million matrices per design) we calculated mean and standard deviation for each matrix and plotted it with mean on 'x' axis and

standard deviation on 'y' axis. For every single diffusion matrix (corresponding to a specific permutation of *i, j, k and l*), one point is plotted on graph. So graph of Salsa as shown in Figure 4, ChaCha shown in Figure 5 and MCC shown in Figure 6, quarter round represent 1,048,576 points each. The graphs for all three design and their comparison is discussed in this section.

5.1 Results for Quarter Round of Salsa Core

The graph for Quarter round of Salsa core is given in Figure 4.

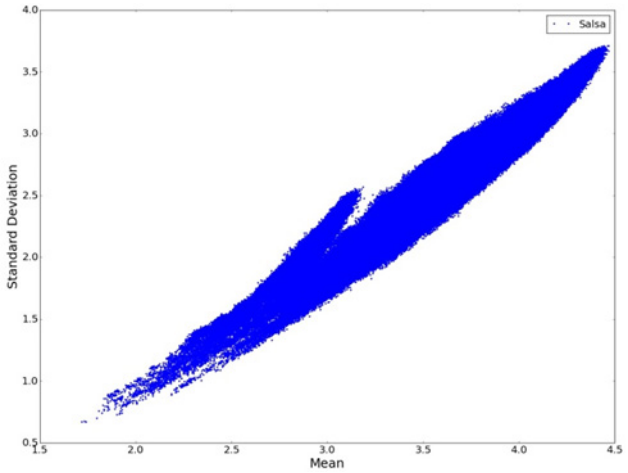

Figure 4 . Mean and standard deviation of diffusion matrices of Salsa's quarter round.

Some interesting observations from the results of Salsa's Quarter round are:

- There are more than 45000 permutations of *i, j, k and l*; that gives slightly better results than prescribed value of rotation constants i.e. mean is higher and standard deviation is lower than the mean and standard deviation obtained with prescribed rotation constants [7, 9, 13, 18].
- The rotation constants that generate the diffusion matrix with the highest mean are [12, 21, 12, 3] i.e. if we use these rotation constants then we will have more diffusion than diffusion obtained with rotation constants prescribed by author. However, the improvement is not huge compared to prescribed rotation constants. The mean and standard deviation of diffusion matrix at [12, 21, 12, 3] are 4.2462, 3.1821 respectively.
- The rotation constants that generate diffusion matrix with smallest standard deviations are [12, 18, 6, 8]. The mean and standard deviation for the diffusion matrix corresponding to these rotation constants are 4.1076, 2.9392 respectively.

5.2 Results for Quarter Round of ChaCha Core

The graph for Quarter round of ChaCha core is given in Figure 5.

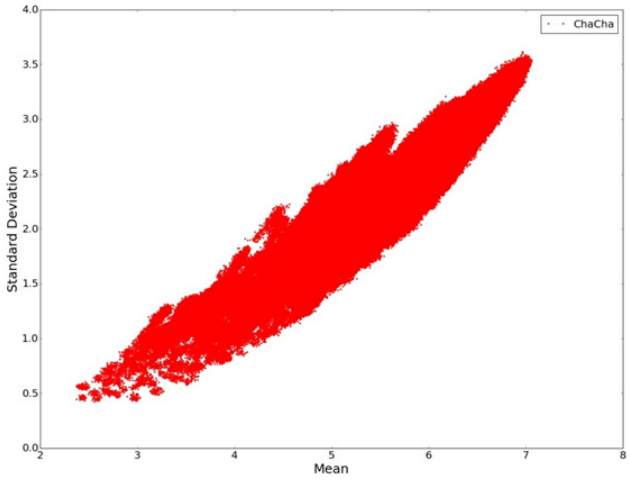

Figure 5 . Mean and standard deviation of diffusion matrices of ChaCha's quarter round.

Some interesting observations from the results of ChaCha Quarter round are:

- For ChaCha quarter round, author changed the rotation constants from [7, 9, 13, 18] to [16, 12, 8, 7]. However, when we examined the diffusion matrix of ChaCha's quarter round on both these rotation constants, we could not find much improvement. For 1000 random values, our experiment reflected that for rotation constants [7, 9, 13, 18]; ChaCha's quarter round created diffusion matrix with mean = 6.8377, standard deviation = 3.2872, and for rotation constants of [16, 12, 8, 7] it generated diffusion matrix with mean = 6.6424, standard deviation = 3.2731.
- There are more than 58000 permutations *of i, j, k and l;* that gives better results than prescribed value of rotation constants i.e. mean is higher and standard deviation is lower than the mean and standard deviation obtained with prescribed rotation constants [16, 12, 8, 7].
- The rotation constants that generate the diffusion matrix with the highest mean is [14, 24, 19, 11] i.e. if we use these rotation constants then we will have maximum diffusion. The mean and standard deviation of diffusion matrix at these rotation constants is 7.0599,

3.5353 respectively.

- The rotation constants that generate diffusion matrix with smallest standard deviation are [5, 13, 6, 23]. The mean and standard deviation for the diffusion matrix corresponding to these rotation constants are 6.6526, 2.8851.

5.3 Results for Quarter Round of MCC

The graph for Quarter round of MCC core is given in Figure 6.

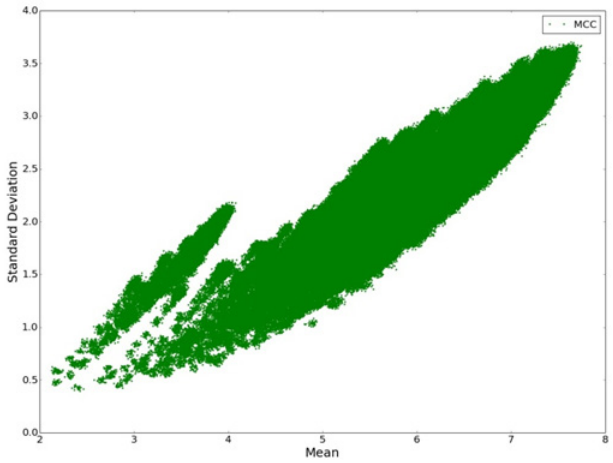

Figure 6 . Mean and standard deviation of diffusion matrices of MCC's quarter round.

Some important observations from the results of MCC Quarter round that led to our decision for rotation constants to be used for MCC quarter round and also about the sequence of parameters to be passed in column and row round are detailed below:

- There are 12 sets of rotation constants that generate diffusion matrices with mean more than 7.70. These 12 top sets are listed in Table 4.
- Diffusion matrices of top 12 sets of rotation constants are quite similar to each other and any one can be picked for the best possible diffusion. However, we picked set no. 5 with rotation constants [4, 17, 8, 0]. This set has a specific property, 4th rotation constant in this set is zero. It means, we need not to have 4th rotation so it reduces our execution cost as we have to do one operation lesser and still we are able to generate the best possible diffusion.
- The diffusion matrix of MCC corresponding to rotation constants [4, 17, 8, 0] is given in Table 5.

Table 5. Diffusion matrix of MCC's quarter round

OP / IP	a	b	c	d
a	13.4	9.03	7.271	15.266
b	10.551	6.909	5.207	12.287
c	7.733	4.319	4.32	9.369
d	5.39	2.77	2.78	6.861

- The diffusion matrix as shown above diffuses output words 'a' and 'd' more than 'b' and 'c'. To be precise, word 'd' is diffused most and word 'c' is diffused least. So in column and row rounds, parameters are passed in a different sequence to make sure that each row and each column have equal opportunities for diffusion. For example, in column round, the sequence of parameters in first call to quarter round is (x_0, x_4, x_8, x_{12}) and this sequence results in highest diffusion in x_4

Table 4. Top 12 sets of rotation constants with Mean 7.70

Sr. No.	1st rotation constant (i)	2nd rotation constant (j)	3rd rotation constant (k)	4th rotation constant (l)	Mean of diffusion matrix	Std. dev. of diffusion matrix
1.	20	17	24	9	7.743	3.669
2.	27	14	22	16	7.730	3.657
3.	4	16	8	19	7.728	3.607
4.	24	16	21	15	7.726	3.527
5.	4	17	8	0	7.716	3.605
6.	21	17	24	15	7.709	3.551
7.	4	17	9	19	7.708	3.652
8.	26	13	23	24	7.706	3.674
9.	4	17	8	16	7.705	3.637
10.	26	14	23	8	7.704	3.644
11.	19	23	15	1	7.704	3.587
12.	26	14	23	22	7.700	3.641

(element of fourth row), followed by x_0 (the element of first row) and then x_8 (element of second row) and least diffusion in x_{12} (element of third row). So when we call quarter round for subsequent columns/rows, our intention is to create more diffusion in those elements which belong to rows that had lesser diffusion in previous calls. With this objective sequence of parameters is decided in column and row round.

5.4 Comparison of Quarter Round of Salsa Core, ChaCha Core and MCC

Graph in Figure 7 gives better picture of relative performance of these three designs. Figure 8 is a zoomed version of this graph showing points having means ≥ 7.0.

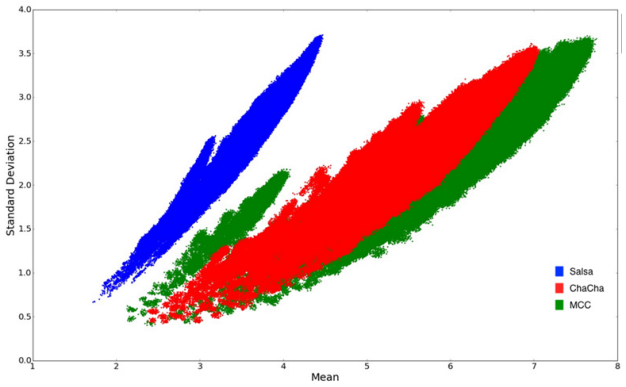

Figure 7 . Comparison of quarter rounds of Salsa core, ChaCha core and MCC.

Important observations from result of all three algorithms are listed below:

- MCC's and ChaCha's Quarter round performs better than Salsa's quarter round: For majority of values of *i, j, k and l;* quarter round of ChaCha and MCC creates more diffusion than Salsa and the same is evident from Figure 7.
- MCC's quarter round outperforms ChaCha's Quarter round: Its evident from Figure 7 and Figure 8 that quarter round of MCC has better diffusion property than that of ChaCha. For considerably high number of permutations of *i, j, k and l*, MCC's quarter round creates more diffusion than ChaCha's quarter round. Our experiment reflected that:
- For at least 200 thousand permutations (about 21%), MCC's quarter round have higher mean than the highest possible mean (7.0599) of ChaCha's quarter round.
- Highest possible mean of ChaCha's diffusion matrix), was not achieved with permutation constants prescribed by author[13]. If we consider prescribed permutation constants, the resultant diffusion matrix has mean of 6.6424, and MCC's core gives at least 450 thousand permutations (about 44%) that have higher mean than this.
- The similar trend is visible if we look at standard deviation of diffusion matrices of both the designs. For more than 650 thousand permutations (about 64%), MCC quarter round generates diffusion matrices, having standard deviation better (lesser) than the best possible diffusion matrix (having lowest standard deviation of 2.8851) of ChaCha's quarter round. As discussed earlier, this diffusion matrix (with least standard deviation) is not observed at author's prescribed rotation constants. Using authors prescribed rotation constants, the diffusion matrix result in standard deviation of 3.2731. For about 90% permutations, MCC's quarter round result in diffusion matrix having lesser standard deviation than this.
- If we compare both mean and standard deviation, more than 340 thousand permutations (about 32%) generate diffusion matrices, that have more mean as well as lesser standard deviation than the diffusion matrix generated by ChaCha's quarter round at [16, 12, 8, 7] (author prescribed rotation constants).
- On average, we found more than 250 thousand permutations (about 22%) in MCC that produce diffusion matrices with mean greater than equal to 7.0 and only about 150 permutations (less than 0.02%) in ChaCha that produce mean greater than or equal 7.0. The same is reflected in Figure 8.

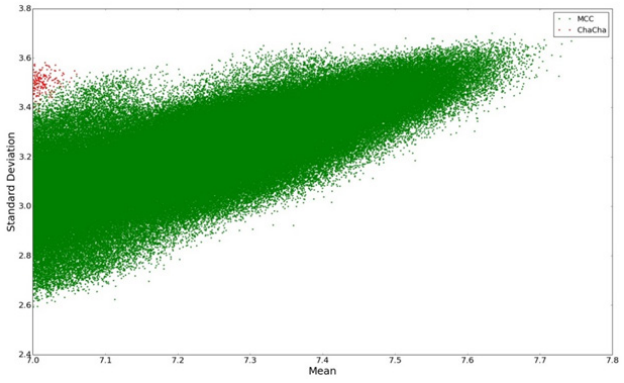

Figure 8. Zoomed version of comparison of three designs for Mean ≥ 7.0.

6. Conclusion and Future Work

Analysis of quarter round of Salsa and ChaCha core for all possible rotation distances reflect that there

are considerably high number of alternative rotation distances that performs better than the rotation constants prescribed by author. Quarter round of MCC creates more diffusion that its counterpart and it does so in lesser operations. The quarter round of MCC as described in Figure 3 is proposed with rotation constants of [4, 17, 8, 0] i.e. without fourth rotation. So our proposed design makes use of four 32-bit additions, four 32-bit XORs and three 32-bit rotations against four rotations in ChaCha as well as Salsa's quarter round. Even after having one lesser operation, MCC has more diffusion than ChaCha and Salsa. MCC's quarter round, on an average generate diffusion matrices with mean of 7.716 against 6.6424 of ChaCha and 4.0992 of Salsa i.e. on an average, gain of 16% from ChaCha and 88% from Salsa Quarter round. The MCC core, based on quarter round as per Figure 3, calls multiple double rounds and may be used to generate stream cipher or may be used to generate collision resistant compression function[15] for a cryptographic hash algorithm. Blake[14], Spritz[16] and Rumba[17] are examples of such an attempt where compression function or hash has been designed from basic structure used by stream cipher. In future, we plan to use MCC core for generating collision resistant compression function for cryptographic hash algorithm.

7. Acknowledgements

We acknowledge the effort and support of Mr. Nikesh Bajaj from School of Electronics and Communication Engineering, Lovely Professional University, Punjab. He has been kind enough to share his time and skills for coding the algorithm in python and his critical inputs has been instrumental in carrying out this research.

8. References

1. Kiruthika B, Ezhilarasie R, Umamakeswari A. Implementation of modified RC4 algorithm for wireless sensor networks on CC2431. Indian Journal of Science and Technology. 2015 May; 8(S9):198–206.
2. Standard, NIST-FIPS. Announcing the Advance Encryption Standard (AES). Federal Information Processing Standards Publication. 2001; 197: p. 1–51.
3. Bogdanov A, Knudsen LR, Leander G, Paar C, Poschmann A, Robsha MJB, et al. PRESENT: An ultra-lightweight block cipher. Proceedings of 9th International Workshop on Cryptographic Hardware and Embedded Systems (CHES); 2007 Sept 10-13; Vienna, Austria. Berlin Heidelberg: Springer; 2007. p. 450–66.
4. ECRYPT Call for Stream Cipher Primitives. 2005. Available from: http://www.ecrypt.eu.org/stream/call/
5. Wu H. The stream cipher HC-128. New Stream Cipher Designs: The eSTREAM Finalists. Springer; 2008. p. 39–47.
6. Boesgaard, Vesterager, Christensen, Zenner. The Rabbit Stream Cipher. New Stream Cipher Designs: The eSTREAM Finalists. Springer; 2008. p. 69–83.
7. Bernstein DJ. The Salsa20 family of stream ciphers. New Stream Cipher Designs: The eSTREAM Finalists. Springer; 2008. p. 84–97.
8. Berbain, Billet, Canteaut, Courtois, Gilbert, Goubin, et al. Sosemanuk, a fast software-oriented stream cipher. New Stream Cipher Designs: The eSTREAM Finalists. Springer; 2008. p. 98–118.
9. Berger, Arnault, Lauradoux. Update on F-FCSR Stream Cipher. The eSTREAM Project - eSTREAM Phase 3. 2008. Available from: http://www.ecrypt.eu.org/stream/p3ciphers/ffcsr/ffcsr_p3.pdf
10. Hell, Johansson, Maximov, Meier. The grain family of stream ciphers. New Stream Cipher Designs: The eSTREAM Finalists. Springer; 2008. p. 179–90.
11. Babbage, Dodd M. The mickey stream ciphers. New Stream Cipher Designs: The eSTREAM Finalists. Springer; 2008. p. 191–209.
12. Canniere CD, Preneel B. Trivium. New Stream Cipher Designs: The eSTREAM Finalists. Springer; 2008. p. 244–56.
13. Bernstein DJ. The ChaCha family of stream ciphers. 2008. Available from: http://cr.yp.to/chacha/chacha-20080128.pdf
14. Aumasson JP, Henzen L, Meier W, Phan RCW. SHA-3 proposal BLAKE - Submission to NIST Hash competition. 2009. Available from: http://csrc.nist.gov/groups/ST/hash/sha-3/Round3/submissions_rnd3.html
15. Nakano Y, Kurihara, Kiyomoto, Tanaka T. Stream cipher-based hash function and its security. e-Business and Telecommunications, 7th International Joint Conference, ICETE; 2010; Athens: Springer. p. 188–202.
16. Rivest RR, Schuldt JCN. Spritz - A spongy RC4-like stream cipher and hash function. CRYPTO 2014 Rump Session; 2014.
17. Bernstein DJ. The Rumba20 compression function. 2007. Available from: http://cr.yp.to/rumba20.html