Applying the CBB20 algorithm for twitter analyzed data

C Bagath Basha, S Rajaprakash, S Muthuselvan P Saisatishsunder, and SVL Alekhya Rani

Department of Computer Science and Engineering, Aarupadai Veedu Institute of Technology, Vinayaka Mission’s Research Foundation, Chennai, Tamil Nadu, India.

Email: saisatish338@gmail.com

Keywords: Decryption, Encryption, Polarity, Salsa, Security, CBB20.

Abstract. Today’s world has data raised from social media like Facebook and Twitter. In this media used to tweet any topics in the internet world, and to analyze neutral, negative and positive tweets; and to create polarity score. From this score used to predict the opportunity. In this polarity data’s have less security. This data will be hacked easily, and change the data results will make issues, like touching the county financial and product firms. To handle these issues, D.J. Bernstein proposed the family of Salsa that has quicker encryption process; it has ovolo process with better data security. This paper introduced a new methodology is proposed. This methodology has altering the Salsa20/4 to improve the security of the analyzed data. A novel methodology consists of three processes. The 1st process is column operation of the matrix, 2nd process is multiplication of top-secret key, and the 3rd process is shift the diagonal elements. The proposed methodology takes more time and provides inflated data security while compared to Salsa20/4.

1. Introduction

Today’s world has data raised from social media like Facebook and Twitter. These media used to tweet any topics from anywhere in the internet world, and to create polarity scores need safety because the Twitter data is public information in the world. In default, data security is not good in Twitter. D. J. Bernstein introduced Salsa family, which family focuses on encryption speed work. Salsa family introduced first Salsa20/20 that is 20 rounds. Salsa20 has various reduced round versions are Salsa20/12 (8 rounds reduced), Salsa20/8 (12 rounds reduced), Salsa20/7 (13 rounds reduced), Salsa20/6 (14 rounds reduced), Salsa20/5 (15 rounds reduced), and finally Salsa20/4 that is 16 rounds reduced from Salsa20/20. Salsa20/4 aim is reduced the encryption time when comparing to other Salsa20 variants, and faster than Advanced Encryption Standard (AES) [17]. Salsa20 algorithm mainly focuses on encryption speed but not improving the security of data. To overcome this drawback of Salsa20/4, a novel algorithm Chan Bagath Basha 20 (CBB20) are proposed.

2. Related work

Salsa family have good attack model are Salsa20/8 and Salsa20/12, and improved the attack on Salsa20 family [1]. Studied the ChaCha and Salsa reduced rounds [2]. ChaCha6 and Salsa7 is enhanced the cipher attack [3]. ChaCha20 is greater than before reduces the buffer size of the loop.
iterations and overall encryption time [4]. Salsa8 and ChaCha7 improved the attack with proper choice of IVs [5]. ChaCha in 6 rounds and Salsa in 7 rounds considerably improved the time complexities [6]. Salsa20 family designed in 2005 and focuses on encryption speed which is faster than the AES [7].

They discuss about the Volume, Velocity, and Variety [8]. They discuss about operations of transformation and polynomial for the security purpose [9]. To detect the opinion tweets and identify the current information in Twitter through method of emotion [10]. Studied the movie reviews and machine learning methods are used to discuss and compare the performance of between Navie Bayes (NB), Maximum entropy classification, and Support Vector Machine (SVM). The SVM performance is the best while comparing to other algorithms [11]. Studied the machine learning methods for survey purposes. A new method is sentiment analysis method and it is used to calculate the accuracy [12].

The talk data using to apply the Salsa20 cryptography algorithms, and with the help of to analyze the encrypt and decrypt processing time is quietly fast [13]. The “correlation attacks” between Salsa20/9 and ChaCha8 are improved [14]. ChaCha8 proposed by Bernstein D. J. [15], and the design modified from Salsa20/8, to illustrate the differences between Salsa20 and ChaCha [16]. SRB18 is mainly discuss about the protection of the twitter analysed data [19]. SRB21 methodology is proposed by Somasundaram Rajaprakash Bagahbasha21 and they mainly discuss the prime numbers of the secret key [20]. The CBB20 is modifying from the Salsa20/4.

3. Proposed methodology

The particular area data collected from Twitter, and it is used to classifying the tweets by means of RStudio on Twitter. These messages or tweets are used to scrutinize neutral, negative, and positive tweets to construct polarity scores. The output of the analyzed polarity data get from Twitter [18]. These data are converted into a matrix form and applied to the novel algorithm CBB20 with the order of matrix N by N. The proposed algorithm has three types of encryption processes. The 1st process is exchange the column of the matrix, 2nd process is multiplication of the secret key and the 3rd process is shifting to the diagonal element. The proposed encryption methodology CBB20 takes more execution time for decryption the data when compared to Salsa20/4.

3.1. CBB20 Encryption Algorithm

Step 1: To get the analyzed data from Twitter and stored in matrix C.

Step 2: Exchange the column in the matrix.

\[ K = C_{ij} \leftrightarrow \left[ C_{i(j + (N-M))} \right] \]

where K is a matrix. \hspace{1cm} (1)

Step 3: To multiply the top-secret key in the K matrix.

\[ D = SK \]

where D is a first matrix. \hspace{1cm} (2)

Step 4: Diagonal elements move to the 1st row.

if \( i > N \) Then

\[ a_{(k+1)j} = a_{(k+j)i} \]

\[ i = k - N \]

where \( a_{(k+1)j} \) and \( a_{(k+j)i} \) are matrix element,

\( k \) various from 0, to \( N; \)

\( i \) various from 1, to \( N; \)

\( j \) various from 1, to \( N; \) and \( N \) is the order of matrix. \hspace{1cm} (3)

3.2. CBB20 Decryption Algorithm

Step 1: 1st row elements are moved to the diagonal elements using equation 3.
Step 2: Divide the top-secret key in the F matrix.

\[ G = F / S \]

where \( G \) is second decrypted matrix

(4)

Step 3: To get the original data using equation (1).

4. Functioning of CBB20 algorithm

- The CBB20 algorithm is developed and modifying from SRB18 algorithm [20].

\[
C = \begin{pmatrix}
C_{11} & C_{12} & C_{13} & C_{14} & C_{15} & C_{16} \\
C_{21} & C_{22} & C_{23} & C_{24} & C_{25} & C_{26} \\
C_{31} & C_{32} & C_{33} & C_{34} & C_{35} & C_{36} \\
C_{41} & C_{42} & C_{43} & C_{44} & C_{45} & C_{46} \\
C_{51} & C_{52} & C_{53} & C_{54} & C_{55} & C_{56} \\
C_{61} & C_{62} & C_{63} & C_{64} & C_{65} & C_{66}
\end{pmatrix}
\]

Where \( C \) is the analyzed twitter data matrix

Encryption

- All the columns should be exchanged in the matrix \( K \) (1).

\[
K = \begin{pmatrix}
K_{16} & K_{15} & K_{14} & K_{13} & K_{12} & K_{11} \\
K_{26} & K_{25} & K_{24} & K_{23} & K_{22} & K_{21} \\
K_{36} & K_{35} & K_{34} & K_{33} & K_{32} & K_{31} \\
K_{46} & K_{45} & K_{44} & K_{43} & K_{42} & K_{41} \\
K_{56} & K_{55} & K_{54} & K_{53} & K_{52} & K_{51} \\
K_{66} & K_{65} & K_{64} & K_{63} & K_{62} & K_{61}
\end{pmatrix}
\]

- Multiplying \( K \) by top-secret key \( S \) (2).

\[
D = SK
\]

\[
D = \begin{pmatrix}
D_{16} & D_{15} & D_{14} & D_{13} & D_{12} & D_{11} \\
D_{26} & D_{25} & D_{24} & D_{23} & D_{22} & D_{21} \\
D_{36} & D_{35} & D_{34} & D_{33} & D_{32} & D_{31} \\
D_{46} & D_{45} & D_{44} & D_{43} & D_{42} & D_{41} \\
D_{56} & D_{55} & D_{54} & D_{53} & D_{52} & D_{51} \\
D_{66} & D_{65} & D_{64} & D_{63} & D_{62} & D_{61}
\end{pmatrix}
\]

The above matrix, all the diagonal elements of the matrix data are shifted to the 1\(^{st}\) row (3). From this process, the 1\(^{st}\) column of the matrix is no changed. The 2\(^{nd}\) column process is (2,2) to the first row (1,2) as path below: (2,2) \(\Rightarrow\) (1,2), (3,2) \(\Rightarrow\) (2,2), (4,2) \(\Rightarrow\) (3,2), (5,2) \(\Rightarrow\) (4,2), (6,2) \(\Rightarrow\) (5,2), and (1,2) \(\Rightarrow\) (6,2). The 3\(^{rd}\) column process is (3,3) to the 1\(^{st}\) row (1,3) as path below: (3,3) \(\Rightarrow\) (1,3), (4,3) \(\Rightarrow\) (2,3), (5,3) \(\Rightarrow\) (3,3), (6,3) \(\Rightarrow\) (4,3), (1,3) \(\Rightarrow\) (5,3), and (2,3) \(\Rightarrow\) (6,3). The 4\(^{th}\) column process is (4,4) to the 1\(^{st}\) row (1,4) as path below: (4,4) \(\Rightarrow\) (1,4), (5,4) \(\Rightarrow\) (2,4), (6,4) \(\Rightarrow\) (3,4), (1,4) \(\Rightarrow\) (4,4), (2,4) \(\Rightarrow\) (5,4), and (3,4) \(\Rightarrow\) (6,4). The 5\(^{th}\) column process is (5,5) to the 1\(^{st}\) row (1,5) as path below: (5,5) \(\Rightarrow\) (1,5), (6,5) \(\Rightarrow\) (2,5), (1,5) \(\Rightarrow\) (3,5), (2,5) \(\Rightarrow\) (4,5), (3,5) \(\Rightarrow\) (5,5), and (4,5) \(\Rightarrow\) (6,5). The 6\(^{th}\)
column process is \((6,6)\) to the 1\(^{st}\) row \((1,6)\) as path below: \((6,6) \Rightarrow (1,6), (1,6) \Rightarrow (2,6), (2,6) \Rightarrow (3,6), (3,6) \Rightarrow (4,6), (4,6) \Rightarrow (5,6), \) and \((5,6) \Rightarrow (6,6)\).

\[
E = \begin{pmatrix}
E_{16} & E_{25} & E_{34} & E_{43} & E_{52} & E_{61} \\
E_{26} & E_{35} & E_{44} & E_{53} & E_{62} & E_{11} \\
E_{36} & E_{45} & E_{54} & E_{63} & E_{12} & E_{21} \\
E_{46} & E_{55} & E_{64} & E_{13} & E_{22} & E_{31} \\
E_{56} & E_{65} & E_{14} & E_{23} & E_{32} & E_{41} \\
E_{66} & E_{15} & E_{24} & E_{33} & E_{42} & E_{51}
\end{pmatrix}
\]

Decryption

From the above matrix, the 1\(^{st}\) row elements are shifted to the diagonal elements by the columns (3).
In this process, the 1\(^{st}\) column process is no changed. The 2\(^{nd}\) column process is \((6,2)\) to \((1,2)\) as path below: \((6,2) \Rightarrow (1,2), (1,2) \Rightarrow (2,2), (2,2) \Rightarrow (3,2), (3,2) \Rightarrow (4,2), (4,2) \Rightarrow (5,2), \) and \((5,2) \Rightarrow (6,2)\). The 3\(^{rd}\) column process is \((5,3)\) to \((1,3)\) as path below: \((5,3) \Rightarrow (1,3), (6,3) \Rightarrow (2,3), (1,3) \Rightarrow (3,3), (2,3) \Rightarrow (4,3), (3,3) \Rightarrow (5,3), \) and \((4,3) \Rightarrow (6,3)\). The 4\(^{th}\) column process is \((4,4)\) to \((1,4)\) as path below: \((4,4) \Rightarrow (1,4), (5,4) \Rightarrow (2,4), (6,4) \Rightarrow (3,4), (1,4) \Rightarrow (4,4), (2,4) \Rightarrow (5,4), \) and \((3,4) \Rightarrow (6,4)\). The 5\(^{th}\) column process is \((3,5)\) to \((1,5)\) as path below: \((3,5) \Rightarrow (1,5), (4,5) \Rightarrow (2,5), (5,5) \Rightarrow (3,5), (6,5) \Rightarrow (4,5), (1,5) \Rightarrow (5,5), \) and \((2,5) \Rightarrow (6,5)\). The 6\(^{th}\) column process is \((1,6)\) to \((6,6)\) as path below: \((1,6) \Rightarrow (6,6), (2,6) \Rightarrow (1,6), (3,6) \Rightarrow (2,6), (4,6) \Rightarrow (3,6), (5,6) \Rightarrow (4,6), \) and \((6,6) \Rightarrow (5,6)\).

\[
F = \begin{pmatrix}
D_{16} & D_{15} & D_{14} & D_{13} & D_{12} & D_{11} \\
D_{26} & D_{25} & D_{24} & D_{23} & D_{22} & D_{21} \\
D_{36} & D_{35} & D_{34} & D_{33} & D_{32} & D_{31} \\
D_{46} & D_{45} & D_{44} & D_{43} & D_{42} & D_{41} \\
D_{56} & D_{55} & D_{54} & D_{53} & D_{52} & D_{51} \\
D_{66} & D_{65} & D_{64} & D_{63} & D_{62} & D_{61}
\end{pmatrix}
\]

- Dividing \(F\) by top-secret key \(S\) (4)

\[
G = \frac{F}{S}
\]

\[
G = \begin{pmatrix}
K_{16} & K_{15} & K_{14} & K_{13} & K_{12} & K_{11} \\
K_{26} & K_{25} & K_{24} & K_{23} & K_{22} & K_{21} \\
K_{36} & K_{35} & K_{34} & K_{33} & K_{32} & K_{31} \\
K_{46} & K_{45} & K_{44} & K_{43} & K_{42} & K_{41} \\
K_{56} & K_{55} & K_{54} & K_{53} & K_{52} & K_{51} \\
K_{66} & K_{65} & K_{64} & K_{63} & K_{62} & K_{61}
\end{pmatrix}
\]

\[
K = \begin{pmatrix}
C_{11} & C_{12} & C_{13} & C_{14} & C_{15} & C_{16} \\
C_{21} & C_{22} & C_{23} & C_{24} & C_{25} & C_{26} \\
C_{31} & C_{32} & C_{33} & C_{34} & C_{35} & C_{36} \\
C_{41} & C_{42} & C_{43} & C_{44} & C_{45} & C_{46} \\
C_{51} & C_{52} & C_{53} & C_{54} & C_{55} & C_{56} \\
C_{61} & C_{62} & C_{63} & C_{64} & C_{65} & C_{66}
\end{pmatrix}
\]
Example 1: the order of matrix N=4.

\[
C = \begin{pmatrix}
1 & 2 & 3 & 4 \\
5 & 6 & 7 & 8 \\
9 & 10 & 11 & 12 \\
13 & 14 & 15 & 16 \\
\end{pmatrix}
\]

Encryption

- To exchange all columns of the matrix \( K \).

\[
K = \begin{pmatrix}
4 & 3 & 2 & 1 \\
8 & 7 & 6 & 5 \\
12 & 11 & 10 & 9 \\
16 & 15 & 14 & 13 \\
\end{pmatrix}
\]

- To multiplying the secret key of \( K \) by \( S=6 \), we give encrypt matrix

\[
D= S * K
\]

\[
D = \begin{pmatrix}
24 & 18 & 12 & 6 \\
48 & 42 & 36 & 30 \\
72 & 66 & 60 & 54 \\
96 & 90 & 84 & 78 \\
\end{pmatrix}
\]

- Using equation (3), all the diagonal elements are shifted to the 1st row.

\[
E = \begin{pmatrix}
24 & 42 & 60 & 78 \\
48 & 66 & 84 & 6 \\
72 & 90 & 12 & 30 \\
96 & 18 & 36 & 54 \\
\end{pmatrix}
\]

- Finally, the encrypted successfully for analyzed twitter data.

Decryption

\[
E = \begin{pmatrix}
24 & 42 & 60 & 78 \\
48 & 66 & 84 & 6 \\
72 & 90 & 12 & 30 \\
96 & 18 & 36 & 54 \\
\end{pmatrix}
\]

- Using equation (3), the first row should be form the diagonal elements of the particular columns.
\[ F = \begin{pmatrix} 24 & 18 & 12 & 6 \\ 48 & 42 & 36 & 30 \\ 72 & 66 & 60 & 54 \\ 96 & 90 & 84 & 78 \end{pmatrix} \]

- Dividing the elements of \( F \) by the top-secret key \( S \) (4).

\[ G = F / S \]

\[ G = \begin{pmatrix} 4 & 3 & 2 & 1 \\ 8 & 7 & 6 & 5 \\ 12 & 11 & 10 & 9 \\ 16 & 15 & 14 & 13 \end{pmatrix} \]

- All the columns will get back to the original positions of the matrix (1).

\[ K = \begin{pmatrix} 1 & 2 & 3 & 4 \\ 5 & 6 & 7 & 8 \\ 9 & 10 & 11 & 12 \\ 13 & 14 & 15 & 16 \end{pmatrix} \]

- The encrypted matrix of the data should be decrypted.

5. Results
In \((1,1)\) matrix order element size of cell memory is 4 bytes. Similarly, \((2,2)\) matrix order have \((1,1)\) 4 bytes of the size of cell memory, \((1,2)\) 3 bytes of the size of cell memory, \((2,1)\) 3 bytes of the size of cell memory, and \((2,2)\) 2 bytes of the size of cell memory. In \((3,3)\) matrix order have \((1,1)\) 4 bytes of the size of cell memory, \((1,2)\) 3 bytes of the size of cell memory, \((1,3)\) 3 bytes of the size of cell memory, \((2,1)\) 3 bytes of the size of cell memory, \((2,2)\) 2 bytes of the size of cell memory, \((2,3)\) 2 bytes of the size of cell memory, \((3,1)\) 3 bytes of the size of cell memory, \((3,2)\) 2 bytes of the size of cell memory, and \((3,3)\) 2 bytes of the size of cell memory. Thus the total file size of an \((3,3)\) order is 24 bytes. Similarly, the remaining file size of the matrix are show in Table 1.

| S. No. | File Size | Matrix Order | Salsa  | CBB20  |
|-------|-----------|--------------|--------|--------|
| 1     | 24 bytes  | 3x3          | 329.3  | 1751.0 |
| 2     | 76 bytes  | 5x5          | 349.2  | 1693.0 |
| 3     | 111 bytes | 6x6          | 343    | 1712.3 |
| 4     | 312 bytes | 10x10        | 345.3  | 6599.1 |
From Table 1, the CBB20 and Salsa algorithm has comparing the encryption speed as micro seconds (μs). The Salsa encryption time taken as 329.3 μs, 349.2 μs, 343 μs, 345.3 μs, 512.5 μs, 401.6 μs, and 274.2 μs. The CBB20 encryption time taken as 1751 μs, 1693 μs, 1712.3 μs, 6599.1 μs 2895 μs, 2082.7 μs, and 2484.9 μs.

From Figure 1, CBB20 and Salsa algorithm has compared the file size of bytes for encryption speed in micro seconds (μs). The encryption speed 329.3 μs, 349.2 μs, 343 μs, 345.3 μs, 512.5 μs, 401.6 μs, and 274.2 μs for Salsa and 1751 μs, 1693 μs, 1712.3 μs, 6599.1 μs 2895 μs, 2082.7 μs, and 2484.9 μs for CBB20 as the 24 bytes, 76 bytes, 111 bytes, 312 bytes, 822 bytes, 1531 bytes and 6580 bytes. The CBB20 algorithm provide more security of the data and taking more time while compared to Salsa algorithm.

6. Conclusion
In this current work, to the particular area data collected from Twitter, and it is used to classifying the tweets by means of RStudio on Twitter, and used to scrutinize neutral, negative, and positive tweets to construct polarity scores. The analyzed polarity data has been get from Twitter. These polarity data have security issues. This data will be hacked easily, and change the data results will make issues, like touching the county financial and product firms. Salsa20/4 is better security and fast encryption because of ovolo round operations. The novel methodology is proposed by modify and developing from the Salsa20/4 to improve the security of the analyzed polarity data. The proposed algorithm CBB20 has 3 processes. The 1st process is column operations of the matrix, 2nd process is multiply the
top-secret key of the matrix and the $3^{rd}$ process is shift the diagonal keys in N-1 steps. The proposed encryption methodology takes more implementation time but provides elevated data security while compared to Salsa20/4.CBB20 output show the high execution time for encryption algorithm because of data security is high when compared to Salsa20 variant. The main advantage of CBB20 algorithm is decryption algorithm, which has time taken high to be decrypted while compared to Salsa algorithms.

References

[1] Ding L 2019 Improved Related-Cipher Attack on Salsa20 Stream Cipher (IEEE Access) p 30197 – 30202
[2] Dey S and Sarkar S 2017 Improved analysis for reduced round Salsa and ChaCha (Discrete Applied Mathematics, Elsevier).
[3] Deepthi Kakumani and Kunwar 2018 Cryptanalysis of Salsa and ChaCha: Revisited (ICST Institute for Computer Sciences, Social Informatics and Telecommunications Engineering) p 324–338
[4] Parmar, Raj R, Roy S, Bhattacharyya D, Bandyopadhyay S K and Tai-Hoon Kim 2017 Large Scale Encryption in the Hadoop Environment: Challenges and Solutions (IEEE Access) p 7156-7163
[5] Maitra S 2016 Chosen IV cryptanalysis on reduced round ChaCha and Salsa (Discrete Applied Mathematics, Elsevier) p 88-97
[6] Choudhuri A K and Maitra S 2016 Significantly Improved Multi-bit Differentials for Reduced Round Salsa and ChaCha (IACR Transactions on Symmetric Cryptology) p 261-287
[7] Bernstein D J 2008 The Salsa20 family of stream ciphers In New Stream Cipher Designs, Springer (p 84-97).
[8] Laney D 2001 3D data management: Controlling data volume, velocity, and variety (Application Delivery Strategies, META Group).
[9] Daemen J and Rijmen V 1999 The Rijndael block cipher (NIST Computer Security Resource Center) p 1-45
[10] Akcora C G, Bayir M A, Demirbas M and Ferhatoğlu H 2010 Identifying breakpoints in public opinion (1st Workshop on Social Media Analytics) p 62–66
[11] Pang B, Lee L and Vaithyanathan S 2002 Thumbs up?: Sentiment classification using machine learning techniques In Proc. of the Conf. on Empirical Method. Nat. Lang. Proc. (89–96)
[12] Giachanou A and Crestani F 2016 Like It or Not: A Survey of Twitter Sentiment Analysis Methods (ACM Computing Surveys) p 28:1-28:41
[13] Afdhila D, Nasution S M and Azmi F 2016 Implementation of Stream Cipher Salsa20 Algorithm to Secure Voice on Push to Talk Application IEEE Asia Paci. Conf. on Wire. and Mobi. p 137–141
[14] Yadav P, Gupta I and Murthy S K 2016 Study and Analysis of eSTREAM Cipher Salsa and ChaCha In 2nd IEEE Inte. Conf. on Engi. and Tech.
[15] Bernstein D J 2008 ChaCha, a variant of Salsa20 (Workshop Record of SASC) p 1-6
[16] Crowley P 2006 Truncated Differential Cryptanalysis of Five Rounds of Salsa20 (SASC 2006 - Stream Ciphers Revisited; Workshop Record) p 198-202
[17] Fischer S, Meier W, Berbain C, Biassie J F and Robshaw M J B 2006 Non-Randomness in eSTREAM Candidates Salsa20 and TSC-4 (Indocrypt; LNCS, Springer-Verlag) p 2-16
[18] Bagath Basha C and Somasundaram K 2019 A Comparative Study of Twitter Sentiment Analysis Using Machine Learning Algorithms in Big Data (International Journal of Recent Technology and Engineering) p 591 599
[19] Bagath Basha C and Rajapraksh 2020 Enhancing The Security Using SRB18 Method of Embedding Computing (Microprocessors and Microsystems, Elsevier)
[20] Bagath Basha C and Rajapraksh S 2019 Securing Twitter Data Using Phase I Methodology (International Journal Of Scientific & Technology Research) p 1952 1955