National College of Ireland

Improvising Jumbling Salting algorithm using even or odd technique

MSc Internship

Cybersecurity

Thamarai Kannan Sabapathy Venkatachalapathy

Student ID: X18105114

School of Computing

National College of Ireland

Supervisor: Imran Khan

National College of Ireland

MSc Project Submission Sheet

School of Computing

Student Name:	Thamarai Kannan Sabapathy Venkatachalapathy		
Student ID:	X18105114		
Programme:	MSc. Cyber Security **Year:** 2019-2020		
Module:	Academic Internship		
Supervisor:	Imran Khan		
Submission Due Date:	12th December 2019		
Project Title:	Improvising Jumbling Salting algorithm using even or odd technique.		
Word Count:	**5268** **Page Count: 22**		

I hereby certify that the information contained in this (my submission) is information pertaining to research I conducted for this project. All information other than my own contribution will be fully referenced and listed in the relevant bibliography section at the rear of the project.
ALL internet material must be referenced in the bibliography section. Students are required to use the Referencing Standard specified in the report template. To use other author's written or electronic work is illegal (plagiarism) and may result in disciplinary action.

Signature:	....
Date:	....

PLEASE READ THE FOLLOWING INSTRUCTIONS AND CHECKLIST

Attach a completed copy of this sheet to each project (including multiple copies)	□
Attach a Moodle submission receipt of the online project submission, to each project (including multiple copies).	□
You must ensure that you retain a HARD COPY of the project, both for your own reference and in case a project is lost or mislaid. It is not sufficient to keep a copy on computer.	□

Assignments that are submitted to the Programme Coordinator Office must be placed into the assignment box located outside the office.

Office Use Only	
Signature:	
Date:	
Penalty Applied (if applicable):	

Improvising Jumbling Salting Algorithm using even or odd technique

Thamarai Kannan Sabapathy Venkatachalapathy
X18105114

Abstract

This research paper presents a secure method to encrypt a plaintext password into server's database. The main process in password security is encryption. Password security faces lot of challenges from many attacks. A new methodology for making passwords more secure is the Jumbling Salting algorithm. In this algorithm the jumbling block consists of shuffled plaintext and random values generated from a predefined set with modulus function. Jumbling block is ordered according to the length of the initialized password, whether it is even or odd. Salt values are initialized by the value of the timestamp at which a user registers and is added to a final jumbled block. At last AES is executed to get final block with hash values and stores them in database. This encryption technique is more unique and random in storing values, making it more secured from many attacks and more improvised with encryption decryption throughput.

Keywords: Plaintext, Encryption, Decryption, Salt, AES, Cryptography

1 Introduction

"There are cyber threats out there, this is a dangerous world, and we have to be safe, we have to be secure no matter the cost" by Edward states that there are many cyber threats happening all around the world, and he emphasizes on safeguarding our data from hackers.
Using a password is the first stage in securing personal data where a string of characters is used for identifying themselves in secured websites with username for integrity, storing information for confidentiality and secrecy for authorization. Passwords are mostly used in accessing bank accounts, online purchase with money and private information. Cryptography is used for securing passwords by providing secrecy and integrity to data, authentication and anonymity to communication by encryption and decryption process of the given password i.e. plaintext entered will be changed into a cipher text with mathematical algorithms.

Nowadays it has become easy for attackers with many new techniques to crack passwords and get confidential information from the victims. Numerous methods are used by the attackers to crack passwords like Brute-Force, Dictionary Attack, Trojan Horse Attack, Phishing, Virus and worms etc. In Brute-force the password will be guessed with trial and error method at multiple intervals of time by automated tools; Dictionary attack, the passwords are all guessed from the previously stored list; Trojan Horse attack, steals the passwords that are stored in the computer; whereas in Rainbow table attack, the passwords are guessed from rainbow tables which will have large database of hashes. (Gautam and Jain, 2019)

Research Question

Can the Jumbling Sorting Algorithm be more improvised as compared to the previous algorithm?

This research paper proposes a new method to secure passwords using Jumbling-Salting algorithm in the area of Cryptography. The previous method was on encrypting passwords using JS algorithm where the length of cipher-text after adding sub-block is a prime number, the cipher text will be reversed before salting and applying Advance Encryption standard algorithm.

This research paper is on encrypting the passwords using Jumbling-Salting algorithm with some changes from the previous method, where in Case I, when the length of password is odd, the plaintext will be shuffled in shuffled password block and added in sequence with jumbled block, salt block and stored in the database, whereas in Case II, if the length of plaintext is even, first the jumbled block is added with shuffled password block followed by salt block and stores the final ciphertext in database. The principle thought of this proposed algorithm is utilized to alleviate the issues looked in the past work. Existing algorithm and methods secure the fundamental issues of encoding passwords with specific imperfections in it. Utilization of Jumbling Salting algorithm in this proposed methodology can defeat the issues by evolving plain-message progressively complex by expanding size and expanding the security of secret word from being hacked from attacker. This algorithm is new, exceedingly effective to encrypt and keeps the password secret.

This research paper is organized as follows. Section I Introduction, this section covers research question, motivation to do this research and a summarized brief explanation about work. Section II Related Work contains literature review of previous works findings, state of the art, comparison and justification for this thesis question. Section III Research Methodology explains about how this model is designed through research procedure and evaluation methodology with detailed explanation of each practice and resources used in this section. Section IV explains the implementation of this project through diagrams like UML, State and use case diagrams. Section V Implementation explains about the execution of proposed idea step by step containing code written, developed models and the language used in this section. Section VI Evaluation provides a complete study of the results and findings from academic and practitioner view in an inside, out and thorough examination of the outcomes through graphs, charts and plots. Section VII Conclusion and future work states how effective the research has accomplished its goals by rehashing key discoveries pursued by a potential future work. The last section contains all the references used as the source for doing this project from research papers and websites.

2 Related Work

Passwords must be secure as they are used by the servers to identify users and authorize them. In encrypting passwords, cryptography plays a major role as there are many cryptographic algorithms previously used for encrypting passwords. This section will provide a critical audit of previous research work relating to algorithms used by briefing advantages and disadvantages faced in present state from the findings.

2.1 Data Encryption Standard Algorithm:

DES algorithm is the first encryption algorithm to be standardized by National Institute of Standard and Technology. It is a symmetric encryption algorithm with two inputs plaintext and the secret key works by using 128 bits for encryption, dividing them into 64 bits to initialize input block 54, and 8 bits for checking odd parity. 16 rounds of process where the block encrypts and decrypts in direction of keys.
(Saikumar, 2017) in her research journal explained the advantages of using DES algorithm.

- DES algorithm encrypts input message of 64 bits with a secret key where the encrypted key will be the cipher key used for further operations on a large scale.
- They use many numbers of rounds making them more complicated to crack.
- DES algorithm is fast in encryption and decryption when compared with RSA Encryption algorithm. (Saikumar, 2017)

(Zodpe, Wani and Mehta, 2012) in their research paper designed an algorithm for DES Cryptanalysis based on known plaintext attack using brute force. For known plain-text attack the attacker will have access to one plaintext where the ciphertext block is decoded with all possible keys and the resultant plaintext is contrasted with known plaintext. The key for which the resultant plaintext matches with the known plaintext is viewed as the right key.

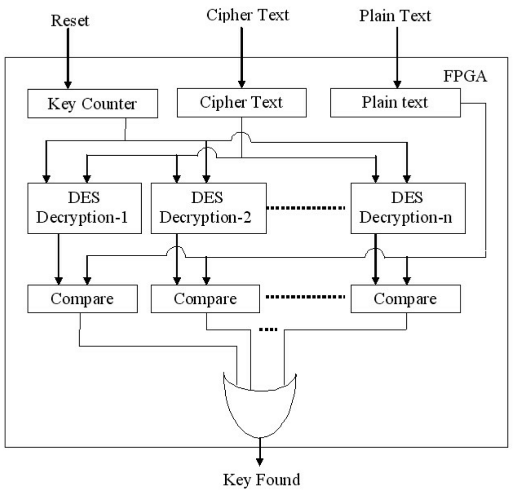


Figure 1 : DES cryptanalysis attack. (Zodpe, Wani and Mehta, 2012)

From the above diagram of the research paper every DES Decryption block decrypts the ciphertext with various set of keys. Along these lines with 'n' nos. of DES Decryption blocks 'n' diverse keys can be looked in one clock cycle in this way decreasing the time required for key hunt by a factor of 'n'. The no. of DES Decryption blocks 'n' relies upon the accessible rationale assets in a FPGA and the rationale use of one DES Decryption block.
DES encryption uses small size of secret making them insecure and easy to crack by brute force search with all possible set of keys within a day.

2.2 Triple Data Encryption Standard Algorithm:

Triple data encryption algorithm practices DES algorithm thrice where the cipher encrypts its information three times. Due to the weakness in DES algorithm, 3DES was implemented and became a general encryption algorithm. In 3DES instead of utilizing single key as in DES, they run the DES algorithm thrice with 56-bit keys.

- Key 1 used to encrypt plaintext
- Key 2 used to decrypt the encrypted plaintext with key1
- Key 3 used to encrypt the plaintext that was decrypted with key2. (Lake, 2019)

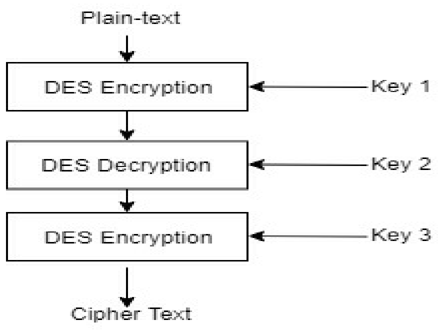


Figure 2: 3DES algorithm

(Potlapally et al., 2007) in their research paper performed cryptanalysis attack on 3DES stated that 48 rounds maps 64 bits of plaintext into 64 bits of ciphertext parameterized on 168-bit secret key. To conduct cryptanalysis attack and crack 3DES a minimum of four plaintext and ciphertext pairs are enough making 3DES also vulnerable to hackers.

2.3 Blowfish Algorithm:

The Blowfish algorithm is a 64-bit block cipher utilized as a swap for DES calculation by taking length key variable of 32 bits to 448 bits with 16 rounds, each round including change of key conditions and key or information subordinate substitution. Each activity done by the XORs and 32-bit expressions of increments with four ordered exhibit information queries per round quoted by Dr. Dobb in his journal. (Scheneir, 1995)

Presenting to (Wang and Que, 2009) paper's examination done on blowfish calculation. It is anything but difficult to learn and execute with all subkeys that are made through Blowfish_Encrypt() which makes all the keys and information mixed together to make it confused to analyze the key.

In (Poston and Dhandania, 2019) research paper, it demonstrated that the blowfish algorithm requires more memory designation contrasted with DES calculation to introduce sub-keys and S-boxes. Blowfish algorithm additionally sets aside a great deal of effort to encode information making them more secure from brute-force attacks while plaintexts with little block size of 64 bits are defenseless against birthday attack.

2.4 Bcrypt:

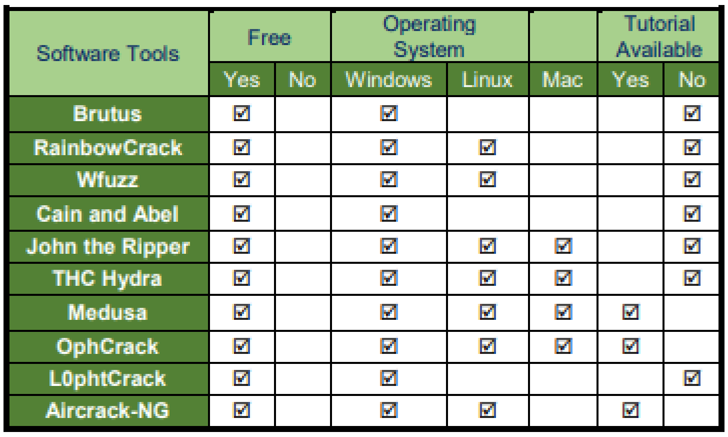
In 1999, Provos and Mazieres developed the password hash technique - Bcrypt. The origin of this algorithm is from the Blowfish algorithm that has an expensive key setup, expand key and user input.
In Expensive key setup, initialization of the values are done by populating the P-array and S-box whereas in Expand Key, the initialized values are added with salt and passwords. In User input the mathematical algorithm is initialize with 18 or 32-bit subkeys.
Due to its slow hashing process, Bcrypt algorithm has been adopted/deployed by Yahoo, Dropox and AshleyMadison, to prevent offline hackers from cracking passwords.
The paper 'Economics of Offline Password Cracking', published by Jermiah, Ben and Samson, describes about the security breaches faced by Yahoo, Dropbox, Ashley Madison and Lastpass.
70 million usersof Yahoo faced a security breach due its hash iteration τ.
Dropbox also faced breach of 68.7 million dropbox users. Dropbox used Bcrypt with 256 hash iterations to hash passwords. Zipf's law parameters for datasets along with RockYOu allows them to predict the passwords used.
AshleyMadison's 40 million password hashes were stolen in 2015. AshleyMadision used Bcrypt at level 12 to hash passwords. Like Dropbox these passwords were predicted using Zipf's law parameters.

2.5 Advanced Encryption Standard:

AES is a symmetric block cipher implemented by NIST to overcome the problems faced by the DES algorithm. AES is a round-based well-defined 128 bit block size for key lengths of 128, 192 and 256 bits with SubBytes, ShiftRows, MixColumns and AddRoundkey. Encryption, Decryption and Round key Generator are the three main modules used in the AES algorithm. (Biglari, Qasemi and Pourmohseni, 2013)
Advantages of AES Algorithm:

- AES provides more security as they are implemented in hardware and software.
- Key sizes used in AES have a higher length like 128, 192 and 256 bits of encryption
- To crack 128-bit encrypted text $2^{128 \text{ tries}}$ are needed making it harder. (Rfwireless-world.com, 2019)

(Kumar and Farik, 2017) in their journal reviewed the software that can crack AES easily. From their journal there are several tools used to crack the AES algorithm which is available free and used mainly as mentioned in the below table 1.



Software Tools	Free		Operating System			Tutorial Available	
	Yes	No	Windows	Linux	Mac	Yes	No
Brutus	☑		☑				☑
RainbowCrack	☑		☑	☑			☑
Wfuzz	☑		☑	☑			☑
Cain and Abel	☑		☑				☑
John the Ripper	☑		☑	☑	☑		☑
THC Hydra	☑		☑	☑	☑		☑
Medusa	☑		☑	☑	☑	☑	
OphCrack	☑		☑	☑	☑	☑	
L0phtCrack	☑		☑				☑
Aircrack-NG	☑		☑	☑		☑	

Table 1

From this journal there is a clear evidence that there are many free tools to crack the AES algorithm making them less secured from hackers. Hence a better encryption technique for securing passwords is to be implemented.

2.6 Jumbling Salting Algorithm:

Encryption technique used for passwords in previous algorithm does provide security by preventing them from many attacks, but still are defenseless to attacks like brute-force or dictionary attack. To avoid this problem (Churi, Ghate and Ghag, 2015) proposed a new improvised encryption technique called Jumbling-Salting Algorithm.

Initially, in Jumbling process addition, selection and reverse processes take place.

1. **Additional Sub-block**: Original plaintext is added with an additional sub-block consisting of random values in this block.
2. **Selection Sub-block:** This block will select the random values to be added in additional block with a pre-defined set of character, digits and special symbols as shown in the below figure 3.

Alphabets	A, B, …….., Y, Z
	a, b, ………, y, z
Digits	0, 1, 2, 3, 4, 5, 6, 7, 8, 9
Special symbols	! ` @ # $ % ^ & * ( ) _ - + = [ ] : ; " ' < >, . ? /

Figure 3 Predefined Set for Jumbling Salting Algorithm. (Churi, Ghate and Ghag, 2015)

3. **Reverse Sub-block:** This block reverses the entire block with the mathematical condition stating if the length of the sub-block even making the text more complex.

Salting block: After Jumbling process salting block will be added to the final jumbled block. Where the standards of salt are user's timestamp value. Format of salt array will YYYY MM DD HH mm ss.
Where, Y = Year, M = Month, D = Date, H = Hours in 24 hours, m = Minute, s= Seconds

In (Churi, Ghate and Ghag, 2015) research paper they conducted an analysis of the plain text size and cipher text size where the size of encrypted cipher text using JS algorithm is three times more than the size of plain text, with no straight relationship between cipher and plaintext. Study of research conducted by (Prasad et al., 2018) tested the plaintext of length size 16 for 11 number of times to encrypt into cipher text using Jumbling Salting algorithm, the outcome of cipher text size varied from 1732 to 946 lengths with an average of 1444. This research proves that the JS algorithm randomly generates cipher text with no replication for providing more security from guessing the length or size of the cipher text.

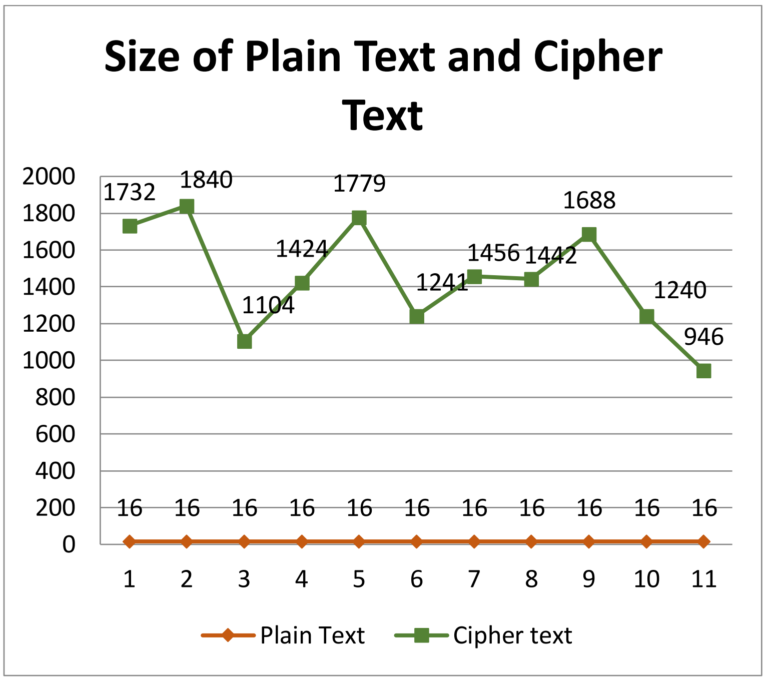


Figure 4 Plaintext vs Ciphertext. (Prasad et al., 2018)

Encryption Time is the time taken to encrypt plain text into cipher text.
JS Encryption Time = Encryption Time for Jumbling Process + Encryption Time taken for Salting + Encryption Time for AES algorithm. With the examination of AES and DES calculation the encryption time of Jumbling Salting calculation is increasing because of numerous means occurring in it. Quickest in encryption is DES calculation pursued by AES then Jumbling Salting where the time taken to encode in Jumbling Salting is thrice the time taken to scramble in DES calculation roughly as mentioned in (Churi, Ghate and Ghag, 2015). Conferring to (Prasad et al., 2018) paper the plain text with the length of 16 for multiple times set aside divergent effort to scramble beginning from 43.74 milliseconds to 64.9 milliseconds with a normal of 54.23 milliseconds.

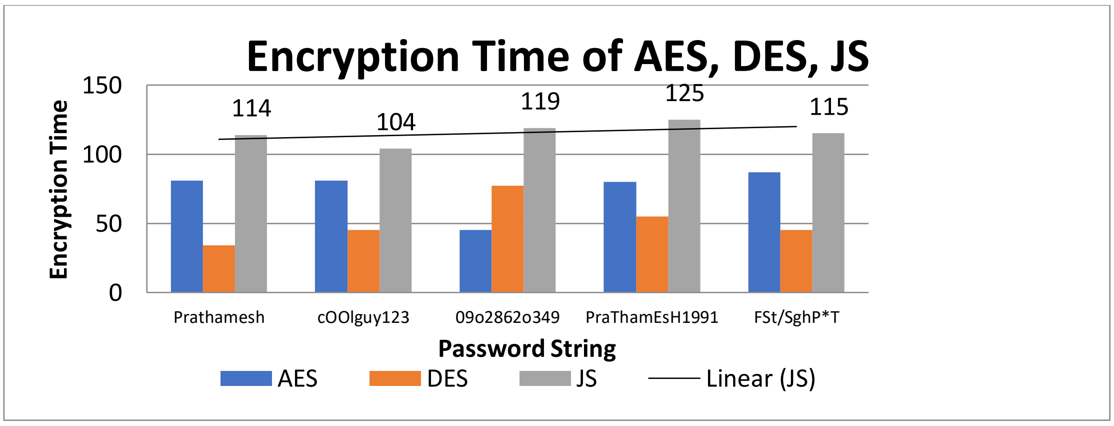


Figure 5: Encryption time of JS, AES, DES Algorithm. (Churi, Ghate and Ghag, 2015)

Decryption Time is the time taken to decrypt cipher text into original plaintext.
JS Decryption Time = Decryption Time of Jumbling Process + Decryption time taken for Salt and AES algorithm. Contrasted with AES and DES calculation the decoding time of disordering salting calculation is high with less millisecond's distinction where the unscrambling time taken by

muddling salting calculation is less contrasted with the time taken or encryption. As in (Prasad et al., 2018) paper 11 cipher texts are being decrypted to 16 plaintexts where each took decoding from 14.57 to 26.52 milliseconds with a normal of 22.292 milliseconds from this decoding time is less contrasted with the time taken to encode.

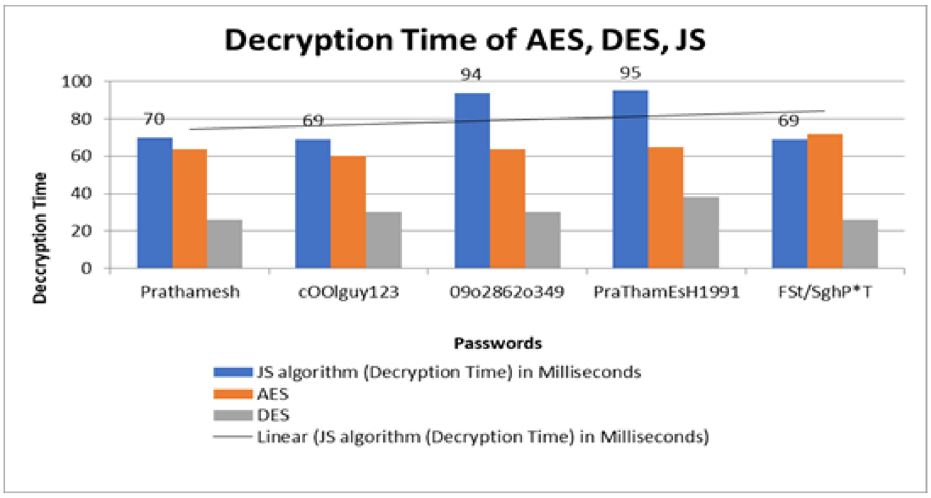


Figure 6: Decryption time of AES, DES, JS. (Churi, Ghate and Ghag, 2015)

Throughput Time is the total time calculated by encryption and decryption time. Various plaintext sizes are utilized in (Saikumar, 2017) paper to inquire about in AES, DES and Jumbling Salting calculations. From (Churi, Ghate and Ghag, 2015) paper, we come to realize that littler the plain-content size decryption byte and time taken to scramble and decode are less contrasted with greater plain-text size. A length of 10 plain-text changes to 176 bytes after encryption with 112 milliseconds to encode with JS calculation while 6808 length of plaint content size changes to 18568 bytes after encryption and takes 375 milliseconds to encrypt. In (Prasad et al., 2018) paper throughput of encryption time is determined by isolating size of Plaintext by Encryption time and throughput of unscrambling time by partitioning Size of figure message by Decryption time. (Bali, Udgata and Churi, 2018) in their research paper tried to encrypt files and send using Jumbling Salting Algorithm resulted in linear throughput where all the files are split and sent as chunk.

2.7 Proposed Methodology

Previous methods have certain flaws in it which allows the attacker to crack password easily. To provide more security, a new methodology of encrypting the password has been implemented with a new technique in Jumbling Salting Algorithm according to the pre-defined conditions to overcome all the problems faced in previous technique. The time period required is talked about in the assessment area with the adequacy and the likelihood of password encryption.

3 Research Methodology

Until now we have examined the requirement for password phrase security. To show the proposed idea, a user's password being stored is secured into the database. The research idea used for this paper has been taken from the paper proposed by Prathamesh and Vaishali and Kranti in [12]. Methodology used in [12] had only one condition where if the length of plaintext is even the final process after encryption is reversed and stored in a database, whereas in proposed idea there will be two different operations taking place if the length of plaintext is even or odd.

3.1 Tools and Software Used:

This research is implemented with the help of two languages and one software tool.

Software tool: Microsoft Visual Studio

Languages: .NET for front end, C sharp for back end

3.2 Enhanced Jumbling Salting Algorithm for password encryption:

After registration of user the plaintext will undergo processes in different blocks for encryption.

A) Shuffled Password Block is the block where the password entered is rearranged in random order to move them left, right, back and forth to change them more trickily. For example, if the plain text is ABC in password block. Then the Shuffled password block will be BAC.

B) Jumbled Block is the block where the random values are generated from a pre-defined set of character, digits and special symbols as shown in the below figure 7.

Alphabets	A, B, ……., Y, Z
	a, b, ………, y, z
Digits	0, 1, 2, 3, 4, 5, 6, 7, 8, 9
Special symbols	! ` @ # $ % ^ & * ( ) _ - + = [ ] : ; “ ‘ < > , . ? /

Figure 7: Random Values

C) Salt Block is the block where the salt to be added to for ciphering password will be stored. The standard of the salt to be added into this research follows the timestamp’s table where the format of the salt array is YYYY MM DD HH mm ss.

D) Advanced Encryption Block After salt block AES rijndael algorithm will be implemented for both encryption and decryption of plaintext with sha-256 hashing.

The password undergoes two different phases to encrypt according to the conditions followed in case1 and case2.

Case 1 (odd)
In case1 if the length of password is odd, then the Cipher block to be stored in database will be Shuffled Password Block + Jumbled Block + Salt Block.
For example, if we take password as Abc
Length: 3

Shuffled password block: bac
Jumbled block: 1#k
Salt block: 26073019162045
Jumbling Salting Block: **bac1#k26073019162045**.
Final ciphertext to be stored in database after AES encryption
edwqnlfih214235otr94rlnenfsko4284o123

Case 2 (even)
In case2 if the length of password is even, then the Cipher block to be stored in database will be Jumbled Block + Shuffled Password Block + Salt Block.
For example, if we take password as Abcd
Length: 4
Shuffled password block: dbca
Jumbled block: 1#k%
Saltblock: 26073019162045
Jumbling Salting Block: **1#k%dbca26073019162045**.
Finalciphertext to be stored in database after AES encryption **oiwqfeno1oi3h8493oeifnfu139**

4 Design Specification:

In this section execution and practice of the plan are explained by the architecture of the effective working of the proposed idea. The activity diagram for this research is displayed below. The plan consists of plaintext, shuffled password block, jumbled block, salt block and two cases of operation happening according to the conditions and final cipher text block. This area outlines diverse UML graphs in connection to the proposed model. For instance, activity and sequence diagram.

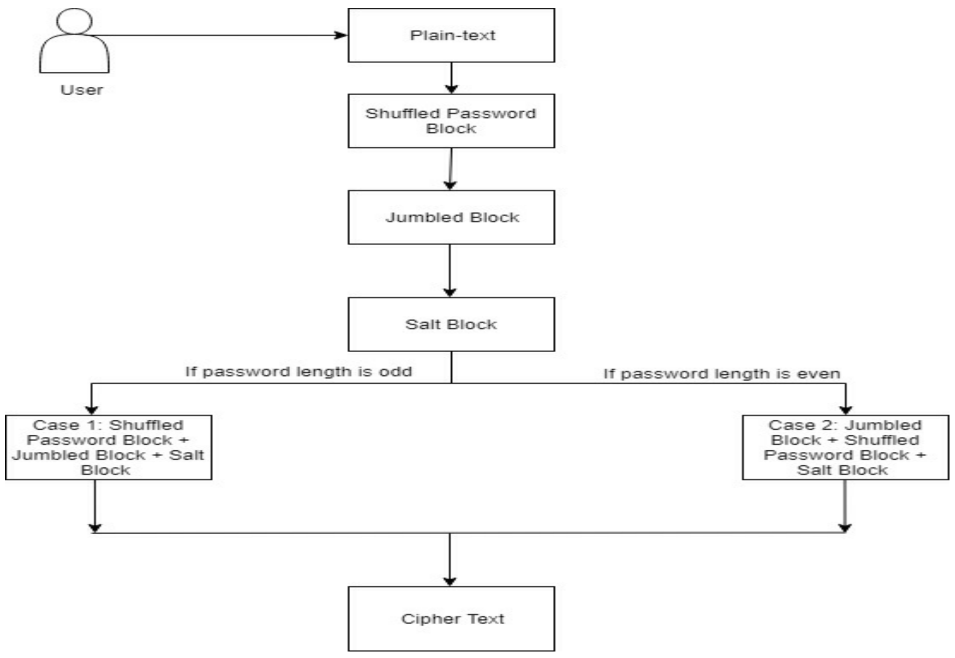

User
Plain-text
Shuffled Password Block
Jumbled Block
Salt Block
If password length is odd
If password length is even
Case 1: Shuffled Password Block + Jumbled Block + Salt Block
Case 2: Jumbled Block + Shuffled Password Block + Salt Block
Cipher Text

Figure 8: Activity Diagram

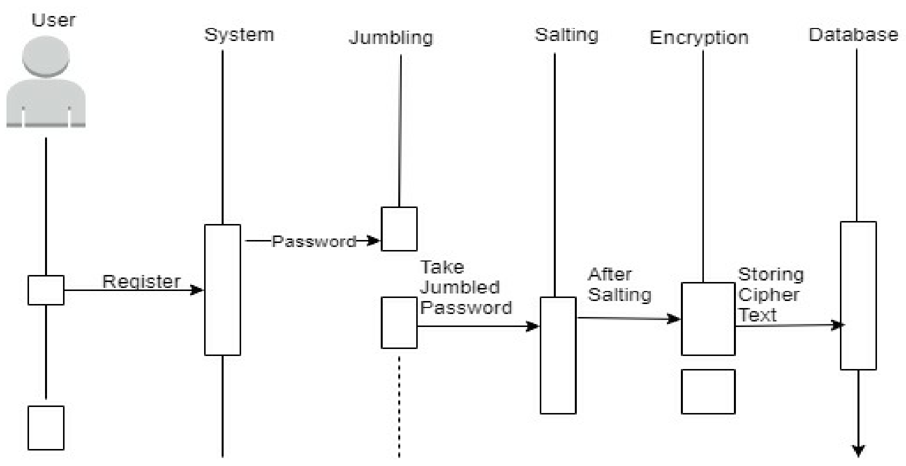
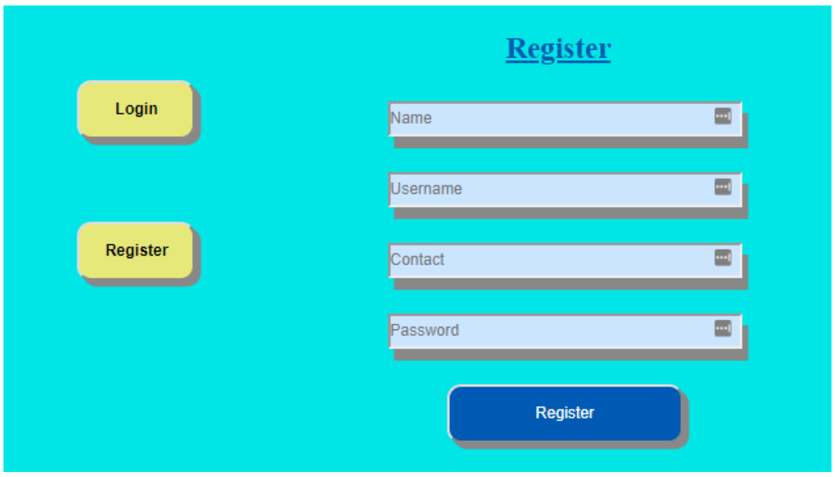


Figure 9: Sequence Diagram

5 Implementation

5.1 User Registration:

The procedure begins by the user creating a new account starting by filling the registration form by providing personal information. The data taken from the user are saved into database to check information when they login again. Screenshot of registration page is as displayed in below figure 10.



. Figure 10: User Registration Page

5.2 Password Encryption and Decryption Implementation:

Input: Plaintext

Output: Ciphertext

1. Storing a password into database by creating a connection through SQL.

2. Passing the password variable into the JumblingSalting function.

3. Defining the password length to zero.

4. Using Random function to generate random values.

5. Defining Salting algorithm using time stamp value to fetch datatime value as salt.

6. Implementing modValus to shuffle and swap the characters with jumbledblock.

7. Creating a condition where the password length is an even number, add the shuffled block first followed by jumbled block.

8. Values obtained by the random function are added to jumbling block followed by shuffled password.

9. If the password length is an odd number, random values are first added to jumbling block followed by shuffled password.

10. Salt values generated through timestamp value are added to final block.

11. RijndaelManaged class is initialized for AES encryption with 256 bytes key size and 128 bytes of block size.

12. SHA256 is used to add hashing to the encrypted jumbled block.

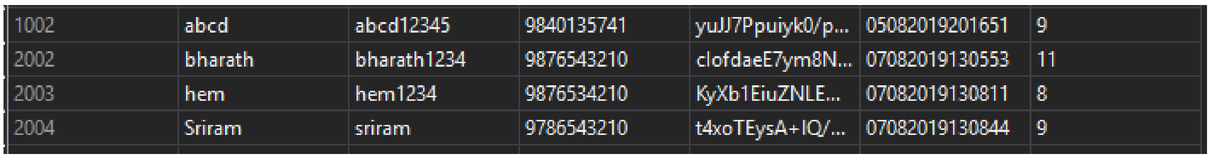
13.For encryption and decryption process AES algorithm using Rijndael method is used.

14. The encrypted password is finally stored in the variable name encryptedBytes.

15. To calculate Encryptiontime in milliseconds the Datetime.Now function is used.

5.3 Database:

All the information given by the user while registering, length of password, Encrypted password and salt are all stored in sql database server using VisualStudio as shown in below figure 11.



1002	abcd	abcd12345	9840135741	yuJJ7Ppuiyk0/p...	05082019201651	9
2002	bharath	bharath1234	9876543210	clofdaeE7ym8N...	07082019130553	11
2003	hem	hem1234	9876534210	KyXb1EiuZNLE...	07082019130811	8
2004	Sriram	sriram	9786543210	t4xoTEysA+IQ/...	07082019130844	9

Figure 11: Database table

6 Evaluation:

6.1 Case Study 1

Password Length, Salt length and Cipher text length

To check the length of plaintext vs cipher text I have taken 5 usernames with password length 11, 10 and 14 with salt length of 12 and found that the resultant ciphertext length as shown in below table 2 and figure 12.

Password length	Salt Length	Ciphertext length
11	12	24
10	12	24
11	12	24
14	12	24
10	12	24

Table 2: Password length, Salt length and Ciphertext length

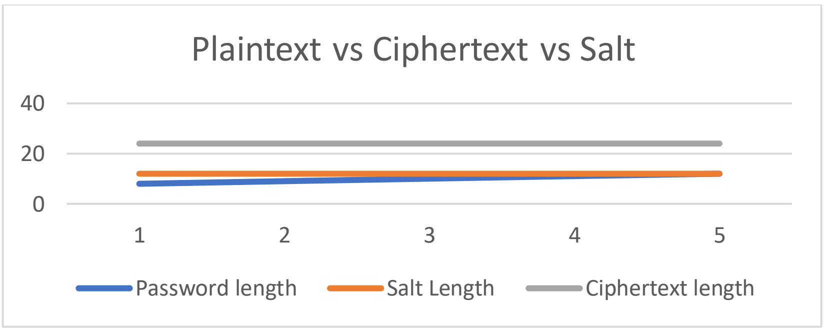


Figure 12: Plaintext vs Ciphertext vs Salt

According to the above graph the when the password entered was non-linear the ciphertext length was showed to be linear irrespective of the password length.

6.2 Case Study 2

Encryption Time

Encryption Time is calculated by the time taken to encrypt the plaintext given by the user from the starting time and time taken for the plaintext to be encrypted and stored in the database. Encryption time calculated in milli seconds as given in below table 3 and graph.

Password String	JS Improvised (ms)	JS (ms)	AES (ms)	DES (ms)
Pr@thamesh1	30.9513	114	81	34
C00lguy@23	36.899	104	81	40
09o2862@34B	38.9282	119	77	45
Pr@ThamEsh199 1	26.9261	125	80	55
FSt/5ghP*t	27.9228	115	87	45

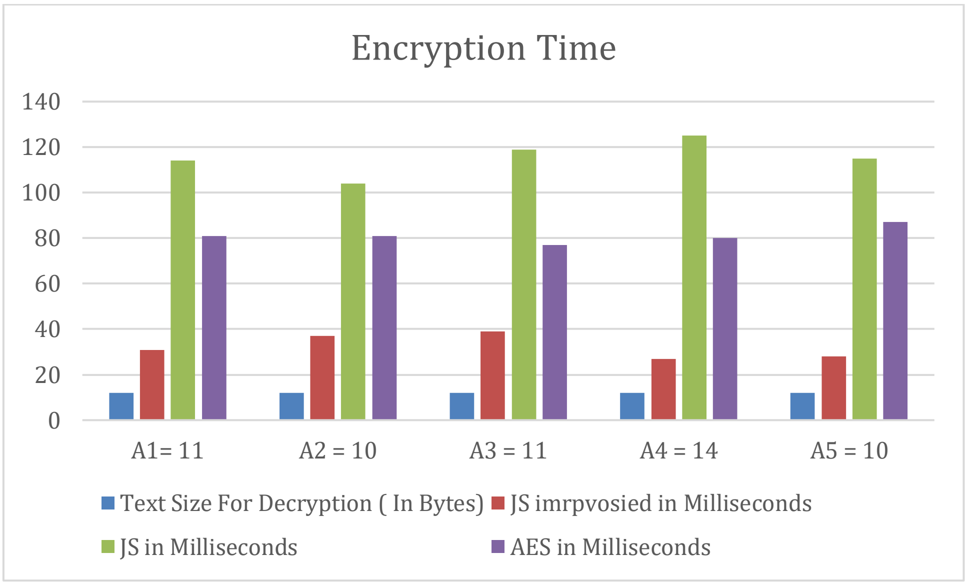


Figure 13 Encryption Time

The time taken by the previous research for encrypting the password using JS, AES and DES algorithms was much more as compared to the improvised algorithm.

6.3 Case Study 3

Decryption Time is calculated from the time taken to login the page after the user logs in with password. This time will be based on the time taken to decrypt the ciphertext stored in the database.

Password String	JS Improvised in Milliseconds	JS in Milliseconds	AES in Milliseconds	DES in Milliseconds
Pr@thamesh1	27.9275	70	64	26
C00lguy@23	31.9476	69	60	30
09o2862@34B	31.9151	94	64	30
Pr@ThamEsh1 991	25.8948	95	65	38
FSt/5ghP*t	25.9311	69	72	26

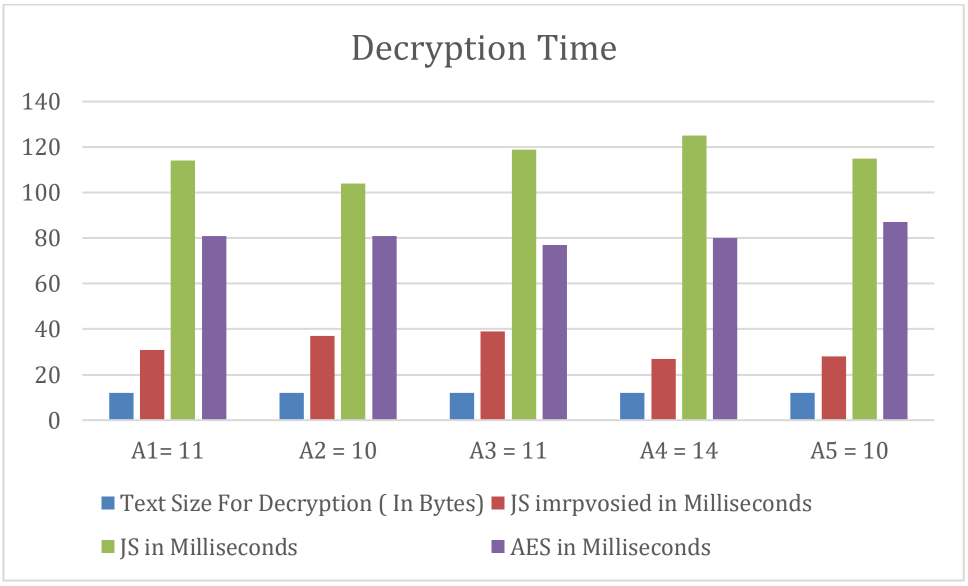


Figure 14. Decryption Time

The time taken by the previous research for decrypting the password using JS, AES and DES algorithms was much more as compared to the improvised algorithm

6.4 Case Study 4

Encryption Time of Improvised Js, Js, AES, DES algorithm for throughput calculation:

Text Size for Encryption (In Bytes)	Text Size For Decryption (In Bytes)	JS imrpvosied (ms)	JS (ms)	AES (ms)	DES (ms)
11	12	30.9513	112	99	71
10	12	36.899	145	128	111
11	12	38.9282	185	163	129
14	12	26.9261	240	203	167
10	12	27.9228	375	311	269
TP 56	60	161.6274	1077	904	747

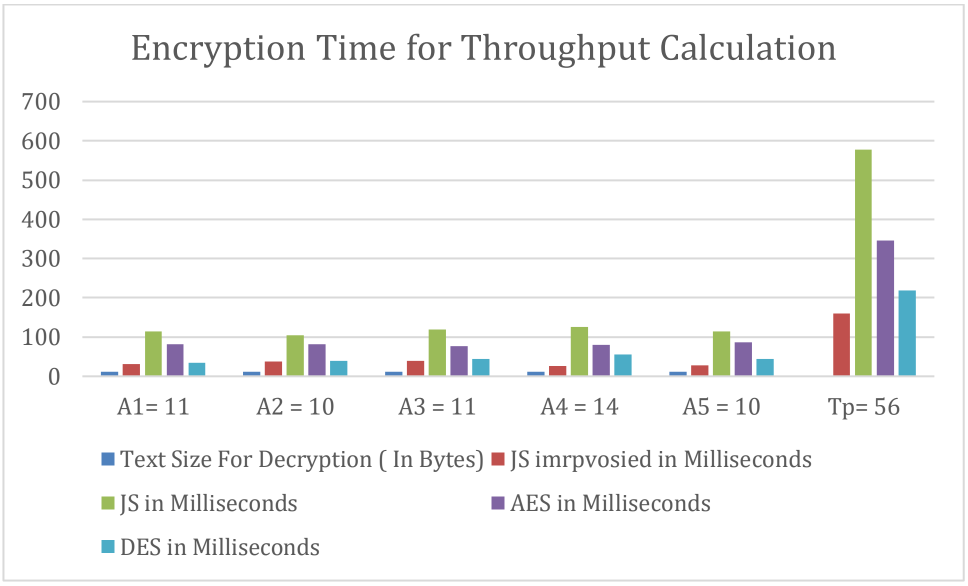


Figure 15. Encryption Time for Throughput Calculation

6.5 Case Study 5

Decryption Time of Improvised Js, Js, AES, DES algorithms for throughput calculation:

Text Size for Encryption (In Bytes)	Text Size For Decryption (In Bytes)	JS imrpvosied in Milliseconds	JS in Milliseconds	AES in Milliseconds	DES in Milliseconds
A1= 11	12	30.9513	114	81	34
A2 = 10	12	36.899	104	81	40
A3 = 11	12	38.9282	119	77	45
A4 = 14	12	26.9261	125	80	55
A5 = 10	12	27.9228	115	87	45
Tp= 56	Tp = 60	160.4274	577	346	219

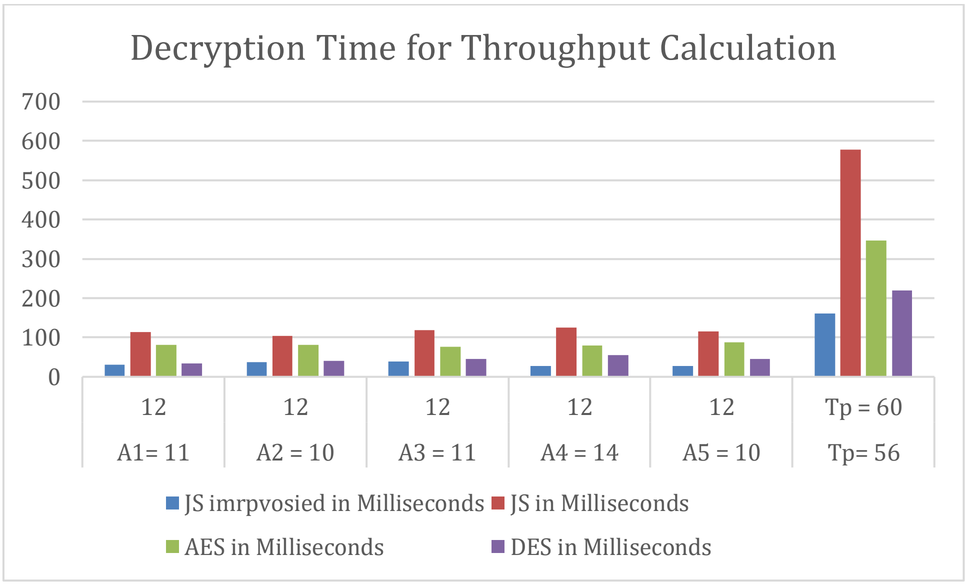


Figure 16. Decryption Time for Throughput Calculation

6.6 Case Study 6

Encryption and Decryption Throughput:

Text	Encryption Throughput in m/s	Decryption Throughput in m/s
A1= 11	2.813	2.53
A2 =10	3.6899	3.19
A3 =11	3.5389	2.9
A4 =14	1.9232	1.84
A5 =10	2.79	2.593

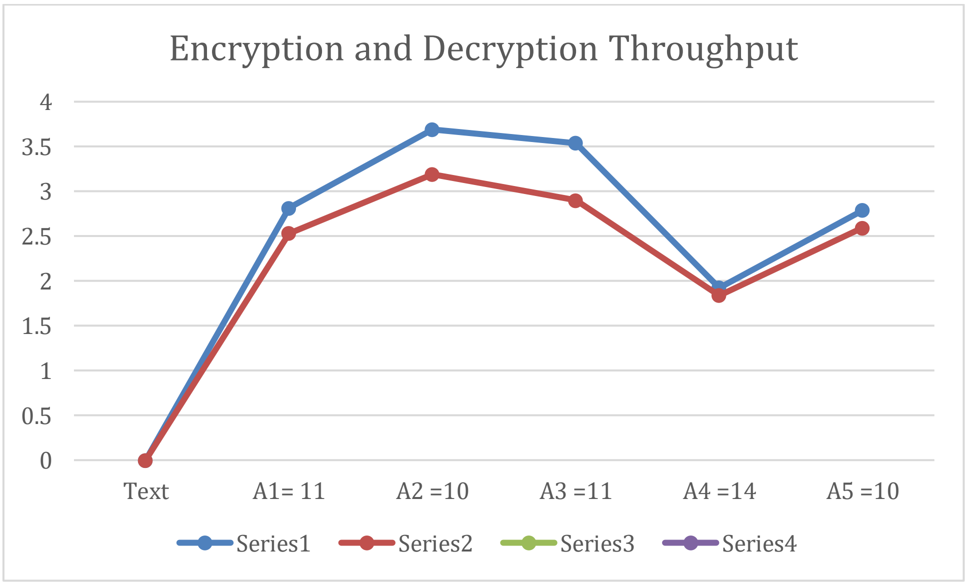


Figure 17. Encryption and Decryption throughput

Encryption Throughput is the total password bytes divided by the average encryption time. The result of this process in the last research was high as compared to the new research algorithm defined in this project.

6.7 Discussion

The size of cipher text in each process is same from Case Study 1. For example, if we take a password length of 11, 10 and 14 the encrypted cipher text length stored in the database is 24, making them more vulnerable to linear cryptanalysis.

From Case Study 2 the encryption time taken by Improvised Jumbling Salting algorithm with the same plaintext is less than the time taken by Jumbling Salting, AES and DES algorithms. For example, the time taken for encryption by the Improved Jumbling Salting Algorithm is 30ms whereas the time taken by the above-mentioned algorithms is 115ms, 87ms and 45ms respectively.

The Case Study 3 shows the decryption times taken by various algorithms including the algorithm under research. The Improvised Jumbling Salting algorithm proved to be more time efficient when compared to other algorithms namely the Jumbling Salting algorithm, AES and DES algorithms.

The Case Study 6 reports encryption and decryption throughput which shows that the improvised jumbling salting algorithm has a good result over the previously implemented algorithms with lot of complexity in the algorithm.

7 Conclusion and Future Work

Jumbling Salting Algorithm used in this research produces an encrypted form of plaintext. There is no real way to stop cyber-attacks. They can be made less compelling by using Jumbling Salting algorithm due to randomization of value.

An improvement of more involvement in this calculation can bring about validation of password from the first stage. Jumbled passwords give uniqueness and increases complexity. AES algorithm is additionally utilized in this calculation to give greater security to encryption and decryption process.

Being a password encryption algorithm, the time complexity also plays an important role along with security. By running this algorithm on a website, I found the result in a neutral format. The average encryption and decryption time calculated at 32.284 ms/p and 31.80 ms/p which is a decent result by adding shuffled password, random values, salt and hash compared to the previous algorithm’s time which took a lot of computation power.

In future, this encryption technique can be implemented by randomizing the length of ciphertext value to prevent linear cryptanalysis attack and random values generated from a set of predefined set of characters can also be made more complex to provide a more secured randomization process.

8. References

Gautam, T. and Jain, A. (2019). Analysis of brute force attack using TG — Dataset.

Saikumar, I. (2017). DES- Data Encryption Standard. International Research Journal of Engineering and Technology (IRJET), (2395 -0056).

Zodpe, H., Wani, P. and Mehta, R. (2012). Design and implementation of algorithm for DES cryptanalysis. 2012 12th International Conference on Hybrid Intelligent Systems (HIS).

Lake, J. (2019). What is 3DES encryption and how does DES work? | Comparitech. [online] Comparitech. Available at: https://www.comparitech.com/blog/information-security/3des-encryption/ [Accessed 9 Aug. 2019].

Potlapally, N., Raghunathan, A., Ravi, S., Jha, N. and Lee, R. (2007). Aiding Side-Channel Attacks on Cryptographic Software With Satisfiability-Based Analysis. IEEE Transactions on Very Large Scale Integration (VLSI) Systems, 15(4), pp.465-470.

Scheneir, B. (1995). The Blowfish Encryption Algorithm. Dr. Dobb's Journal.

Wang, M. and Que, Y. (2009). The Design and Implementation of Passwords Management System Based on Blowfish Cryptographic Algorithm. 2009 International Forum on Computer Science-Technology and Applications.

Poston, H. and Dhandania, K. (2019). Blowfish: The first well-known encryption algorithm in public domain | CommonLounge. [online] Commonlounge.com. Available at: https://www.commonlounge.com/discussion/d95616beecc148daaa23f35178691c35 [Accessed 9 Aug. 2019].

Blocki, J., Harsha, B. and Zhou, S. (2018). On the Economics of Offline Password Cracking. *2018 IEEE Symposium on Security and Privacy (SP)*.

Biglari, M., Qasemi, E. and Pourmohseni, B. (2013). Maestro: A high performance AES encryption/decryption system. The 17th CSI International Symposium on Computer Architecture & Digital Systems (CADS 2013).

Rfwireless-world.com. (2019). Advantages of AES | disadvantages of AES. [online] Available at: https://www.rfwireless-world.com/Terminology/Advantages-and-disadvantages-of-AES.html [Accessed 9 Aug. 2019].

Kumar, J. and Farik, M. (2017). Cracking Advanced Encryption Standard-A Review. International Journal of Scientific Technology Research.

Churi, P., Ghate, V. and Ghag, K. (2015). Jumbling-Salting: An improvised approach for password encryption. 2015 International Conference on Science and Technology (TICST).

Prasad, M., Oruganti, M., Shah, M., Pavri, M. and Churi, P. (2018). Improvised E-commerce Transaction Security using JSSecure Algorithm. 2018 IEEE International Conference on System, Computation, Automation and Networking (ICSCA).

Bali, M., Udgata, M. and Churi, M. (2018). Symmetric Jumbling-Salting Encryption Algorithm for Files. 2018 Fifth HCT Information Technology Trends (ITT).