National College of Ireland

Enhancing Password Security Using a Hybrid Approach of SCrypt Hashing and AES Encryption

MSc Internship
Cyber Security

Vicky Bidhuri
Student ID: x18103120

School of Computing
National College of Ireland

Supervisor: Niall Heffernan

National College of Ireland
Project Submission Sheet
School of Computing

National College of Ireland

Student Name:	Vicky Bidhuri
Student ID:	x18103120
Programme:	Cyber Security
Year:	2018
Module:	MSc Internship
Supervisor:	Niall Heffernan
Submission Due Date:	12/08/2019
Project Title:	Enhancing Password Security Using a Hybrid Approach of SCrypt Hashing and AES Encryption
Word Count:	5391
Page Count:	19

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:	7th August 2019

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):	

Enhancing Password Security Using a Hybrid Approach of SCrypt Hashing and AES Encryption

Vicky Bidhuri
x18103120

Abstract

To be a part of this modern society, we have to register for online accounts and have to swallow the fact that a centralized database storing our passwords will sooner or later suffer a data breach. Cryptography techniques like hashing (SHA256, BCrypt, SCrypt and Argon2) and encryption (AES, 3DES and ECC) are proven secure; however, the confidentiality of password is still compromised. Our sensitive and valuable information is still vulnerable to multiple attacks like brute force, reverse engineering, Graphical Processing Units (GPU) and custom hardware attack. This research paper focuses on the hybrid combination of a strong memory hard hashing function SCrypt and a memory accelerated Advanced Encryption Standard (AES) algorithm, which can be used to enhance the security of password of online users from brute force attacks.

1 Introduction

Cybercrimes are becoming inevitable these days because of the tremendous advancement of technology. It became challenging for an organization to retaliate against attackers who are responsible for the leakage of sensitive information. Data breaches happen almost daily, which exposes billions of user data like passwords, credit card number, bank account number and other highly valuable and sensitive data[1]. Recently, a leading gaming website "Emuparadise" face a data breach, which leads to over 1.1 million account details leaked. The reason behind that was the vulnerable cryptographic algorithm (MD5) used for the protection of passwords[2]. "Facebook" has also admitted that they had failed to protect password of over 600 million users which were stored in the form of plain text[3]. Passwords have always been the main target for attackers to get into the system. Broken authentication is a vulnerability caused by insecure passwords, and it comes under the top 10 list of the Open Web Application Security Project (OWASP)[4]. There are multiple attacks such as brute force, dictionary, rainbow table, reverse lookup table attacks and more, which are the major threat to passwords (Ertaul, Kaur and Gudise; 2016). Brute force is a kind of attack where multiple combinations of username and password are tried again and again until the correct one is found (Turan, Barker, Burr and Chen; 2010). Cryptography is the process of converting the plain text into an encoded text with the

[1] https://selfkey.org/data-breaches-in-2019/
[2] https://www.zdnet.com/article/emuparadise-gaming-rom-repository-suffers-data-breach/
[3] https://selfkey.org/data-breaches-in-2019/
[4] https://www.owasp.org/index.php/Category:OWASP_Top_Ten_Project

help of mathematical functions. Different cryptography approaches like hashing and encryption can be used for data protection (Padmavathi and Kumari; 2013). Hashing is a single way transformation of the plain text into a fixed-length value called message digest or hash; for example, MD5, SHA1/2/3, PBKDF2, BCrypt, SCrypt and Argon2 (Sriramya and Karthika; 2015). Encryption is a process of converting the plain text into cipher text and vice-versa with the help of a key; for example, RC5, ECC, DES, 3DES and AES (Singh; 2013). In this paper, we are focusing on SCrypt hashing, AES encryption and their combination to be used against brute force attack.

Motivation: Poor password protection leads to broken authentication vulnerabilities, that allow cybercriminals to get unauthorized access into the system. Although having a robust security mechanism like cryptographic hashing and encryption used for password protection, passwords are still vulnerable to multiple attacks. The main motive behind this research is to blend two or more different cryptography approaches, which increases the complexity of passwords and make it difficult for attackers to break them.

In this paper, we seek to address the following research question: **Can we enhance the password security of an online user from brute force attack by using a hybrid combination of SCrypt hashing and the Advanced Encryption Standard (AES) algorithm?**

The structure of this paper is organized as follows; *Section 1: Introduction* will introduce the topic and motivation behind this research. After that, *Section 2: Related Work* will provide the literature review of this research with *Subsection 2.1 Password Protection using Hashing Algorithms*, which outlines the related work done on different hashing techniques and why SCrypt is chosen for this research. Then, *Subsection 2.2 Password Protection using Encryption Algorithms*, provides a comparative study of different encryption algorithms and the features of the AES encryption scheme. The *Subsection 2.3 Previous work on Hybrid Algorithms* discusses the research already done on hybrid cryptography approaches and why SCrypt and AES combination is right for enhancing the security of passwords. In the next *Section 3: Methodology* of this model is explained briefly with the help of diagrams. The subsections *3.1 Key Derivation*, *3.2 AES Encryption* and *3.3 AES Decryption* provides more detail about these phases. Next *Section 4: Design Specification* illustrates the architecture and a word-based pseudo algorithm of this model. Then, *Section 5: Implementation* provides a full working of this model with a flowchart. After that, *Section 6: Evaluation* discusses the results of this research with tables and graphs. The final *Section 7: Conclusion and Future Work* summarizes the research and provide a future scope.

2 Related Work

The following subsections will provide the literature review related to SCrypt hashing algorithm, Advanced Encryption Standard (AES) algorithm and their hybrid approach.

2.1 Password Protection using Hashing Algorithms

This section mainly focuses on SCrypt and other hashing algorithms that are used for the protection of the password. (Sriramya and Karthika; 2015) state that hashing is one of the best methods used to protect passwords compared to encryption because it is a one-way transformation process, and no one can retrieve the plain text from its hash value. However, a hashed password has a possibility of a break down by using the pre-calculated

hash value or active hash dictionary. It could be possible that two different plain texts have the same hash value, because of collision effects (Ertaul et al.; 2016). Therefore, simple hashing algorithms like MD5, SHA1 and SHA256 are prone to the dictionary, brute force, rainbow and lookup table attacks. Research done by (Ertaul et al.; 2016) also addresses the same issue and express that there are many open-source software like *John the Ripper*[5] and *Hashcat*[6] available in the market, which can crack the traditional hashing algorithms and recover the actual password. In order to avoid this problem, salt can be used. As stated by (Marton, Suciu and Ignat; 2010), salt is a random string of bits which can be generated securely with pseudo-randomness and added to the plain text password before hashing. If the same hashing algorithm applies multiple times on a single plain text, different hash values will generate because of the salt. While, (Ertaul et al.; 2016) stated that salt increases the randomness of hash and makes the password less susceptible to rainbow table and look table attacks, but still does not protect from the dictionary and brute force attacks. These kinds of attacks are more powerful because of high computational custom hardware, which is capable of performing millions of hashing per second. Therefore, more advanced hashing algorithms like PBKDF2, BCrypt and SCrypt are used as they are based on key stretching technique. In this technique, a computation is added in the key generation process to stretch the time for generating the hash value and thus slow down the algorithm. This technique makes it difficult for an attacker to crack the hash easily. BCrypt and SCrypt are slow algorithms as compared to PBKDF2. Slow algorithms are difficult to break than fast algorithms. Apart from salting, an iteration count is another security factor added in these advanced hashing algorithms.

According to (Turan et al.; 2010) a user-defined password should not be used directly as cryptographic key for the hashing algorithm as they have weak randomness and low entropy (the amount of uncertainty in an unknown value). However, there are a few situations where the password is the only option that can be used as secret information for cryptographic algorithms. Therefore, a Password-Based Key Derivation Function (PBKDF) algorithm can be used, where the key is generated from a secret value like a password. Meanwhile, (Percival and Josefsson; 2016) had shown that most of the key derivation functions like PKCS5 PBKD2, SHA2CRYPT, NTLM Hash, FreeBSD MD5 Crypt and BCrypt were based on cryptographic security factors like salting and iteration count. All these algorithms share a common weakness against powerful attackers. With the drastic development in semiconductor technology, processors become very small and much faster, and can perform a large amount of parallel processing at the same cost. Because of this strong parallelism power, attackers can crack the password faster by using a brute force attack even after increasing the iteration count. Therefore, a stronger hashing algorithm like SCrypt aims to bring down the attackers that are taking advantage of custom-designed parallel circuits.

As described by (Ertaul et al.; 2016), SCrypt is a password-based key derivation hashing algorithm that generates a large vector of pseudo-random bit strings. It takes the plain text as the input and generates a fixed-length hash value or message digest. It is based on memory-hard functions, therefore provides the highest level of security to passwords and makes it almost impossible for an attacker to crack the password by brute force attacks. It requires a large amount of memory for its pseudo-random bits and has high processing costs. Therefore, it is one of the most expensive but secure

[5] `https://www.openwall.com/john/`

[6] `https://hashcat.net/hashcat/`

hashing algorithms. SCrypt is used in the proof of works, key derivations and for many cryptocurrency applications; and also inspired the design of Argon2 (winner of password hashing competition) (Hatzivasilis, Papaefstathiou and Manifavas; 2015).

A study conducted by (Percival; 2009) suggests increasing the circuit size used for key derivation to stand against parallel processing attackers. If the circuit size is double than only half copies can be placed on a given area of silicon. Still operating within the given resources available for software implementation, including a strong CPU and large RAM. In order to increase the cost of parallel attacks, the operation itself will not be the only parameter considered but the memory usage as well. Therefore, authors introduce Memory Hard Functions (MHF), which is an algorithm that asymptotically uses as much memory as its operations. It was also stated by (Alwen, Chen, Pietrzak, Reyzin and Tessaro; 2017) that MHFs are the hashing algorithms where evaluation cost is controlled by memory cost. Therefore, it is challenging to evaluate these algorithms on dedicated hardware at a significantly lower cost. This research also proves that SCrypt being a sequential MHF is optimally memory-hard. Different types of hashing algorithms were compared by (Percival; 2009), as shown in Table 1, which provides the estimated cost (in dollars) of the hardware required to break the hashed password in a year. It is clear from the analysis that SCrypt is much more expensive for an attacker to crack the password as compared to other hashing algorithms like DES CRYPT, MD5, MD5 CRYPT, PBKDF2 and BCrypt.

KDF	6 letters	8 letters	8 chars	10 chars	40 chars text	80 chars text
DES CRYPT	<$1	<$1	<$1	<$1	<$1	<$1
MD5	<$1	<$1	<$1	$1.1k	$1	$1.5T
MD5 CRYPT	<$1	<$1	$130	$1.1M	$1.4k	$1.5 x 10^{15}
PBKDF2 (100ms)	<$1	<$1	$18k	$160M	$200k	$2.2 x 10^{17}
Bcrypt (95ms)	<$1	$4	$130k	$1.2B	$1.5M	$48B
Scrypt (64ms)	<$1	$150	$4.8M	$43B	$52M	$6 x 10^{19}
PBKDF2 (5.0s)	<$1	$29	$920k	$8.3B	$10M	$11 x 10^{18}
Bcrypt (3.0s)	<$1	$130	$4.3M	$39B	$47M	$1.5T
Scrypt (3.8s)	$900	$610k	$19B	$175T	$210B	$2.3 x 10^{23}

Table 1: Estimated hardware cost need to crack the password in one year (Percival; 2009)

It was stated by (Hatzivasilis et al.; 2015) that the evolution of parallel computing and dedicated hardware devices on GPU, FPGA and ASIC also gave advantage to the cyber-criminals to perform more powerful and effective attacks. This research shows that even modern hashing algorithm like PBKDF2 and BCrypt are susceptible to these attacks. Whereas, a memory-hard function like SCrypt as a solution to this problem can be much expensive, but secure. However, SCrypt is also vulnerable to another kind of attacks like Cache-Timing and Garbage-Collector attacks.

By summarizing the above section, we can state that simple hashing algorithms like MD5, SHA256/512, PBKDF2 and BCrypt with salting and key stretching are not sufficient to make the passwords secure from brute force, dictionary, rainbow table, reverse engineering table and lookup table attacks. More advanced and secure hashing algorithm based on memory-hard function like SCrypt can be used but with some additional high cost. However, they are still not fully prone to advanced level parallel processing attacks. Therefore, other cryptography technique like a robust encryption algorithm may be considered for the protection of user password.

2.2 Password Protection using Encryption Algorithms

The previous section provides an overview of strong hashing techniques that can be used for password protection; however, they have their own weakness towards powerful attacks. This section will focus on other cryptography technique like encryption, especially AES, which may fulfill the desired security level for passwords. It was researched by (Álvarez-Sánchez, Andrade-Bazurto, Santos-González and Zamora-Gómez; 2017) that password management can be a crucial task when hashing is involved in their protection. As attackers have access to GPU and custom hardware, they can successfully break the password by performing a brute force attack. Therefore, there is a need for an advance and unique hashing algorithm or other cryptography function where the time and space parameters could be adjusted in order to slow down brute force attacks. (Musliyana, Arif and Munadi; 2015) had mentioned about encryption algorithms like AES, DES, IDEA, RC4, RSA, DSA, DH and ECC, as the alternative of hashing that could be used for better protection of passwords.

(Widiasari; 2012) stated that the National Institute of Standard and Technology (NIST) had chosen the Rijndael algorithm to be used as Advanced Encryption Standard (AES) because of its high security and efficiency. DES was replaced by AES to be used as the data encryption standard because of its small key size (56-bit). It was not enough to provide a sufficient level of data security. Although 3DES came up with a solution as a more efficient and secure version of DES, but it was very slow. AES is a block cipher algorithm that works on substitution (S-box) and permutation (P-box) system. AES has three different versions based on key length, which are AES-128, AES-192 and AES-256 with the key size of 128 bits, 192 bits and 256 bits respectively. AES is one of a best encryption algorithm which is currently used as the data encryption standard.

The research done by (Musliyana et al.; 2015) also describes that AES is a secure and strong encryption algorithm that provides better protection to passwords as compared to few hashing algorithms like MD5 and SHA1. AES used with time-based key generation provides relatively fast encryption and decryption speed, which leads to stronger password security. A comparison between symmetric encryption algorithms (AES, DES and 3DES) and asymmetric algorithm like RSA was conducted by (Singh; 2013). The authors observed that AES was more efficient in terms of encryption/decryption speed, time, throughput and avalanche effect. Similarly, (Padmavathi and Kumari; 2013) had conducted a comparative analysis of algorithms like AES, DES and RSA with a steganographic algorithm like LSB substitution technique. This analysis was done in order to check the performance of these algorithms while ensuring security. The result of this analysis concluded that AES algorithm was not only faster and more efficient than RSA and DES in terms of encryption/decryption time but also in terms of buffer usage and software/hardware implementation. In addition to that, AES was also found to be highly secure and consume less power than others. Different symmetric encryption algorithms are compared and analyzed by (Kant and Sharma; 2013), as shown in Table 2, based on factors like speed, key length, block size, cryptoanalysis resistance and security. Researchers concluded that AES was much faster, efficient (in terms of encryption/decryption time and throughput), secure and resistant towards many attacks like differential linear, interpolation and square attacks. (Álvarez-Sánchez et al.; 2017) stated that AES is natively slower than a few stream ciphers but still has a significant advantage of hardware acceleration in modern processors. AES can be used to protect data against any custom hardware or GPU attacks since it is already implemented as hardware in the system,

Characteristics	AES	DES	3DES	IDES	Blowfish	RC5
Key Length	128, 192, 256	56	112, 168	128	32 - 448	MAX 2040
Block Size	128, 192, 256	64	64	64	64	32, 64, 128
Speed	Very Fast	Very Slow	Slow	Slow	Fast	Slow
Security	Considered Secure	Proven Inadequate	Considered Secure	Proven Inadequate	Considered Secure	Considered Secure
Cryptanalysis Resistance	Very strong against differential, truncated, differential, linear, interpolation and square attack.	Vulnerable to differential and linear cryptanalysis, weak substitution table.	Strong against differential, truncated differential, linear, interpolation and square attack.	Vulnerable to differential and linear cryptanalysis.	Strong against the standard differential and linear cryptanalysis.	Vulnerable to differential, truncated differential, linear, interpolation and square attack.

Table 2: Comparative study of multiple encryption algorithms (Kant and Sharma; 2013)

balancing the field between attackers and those protecting the data. The other main advantage of AES against these attacks is its memory usage that prevents parallelism. Unlike hashing algorithms which perform multiple orders of magnitude faster on these platforms and allowing brute force attacks to succeed. These features of AES made it very fast with speed of over 1GB/s on many machines and qualified itself as the building block for those cryptosystems that aim for speed and security. The researchers also compared the performance of AES in CTR mode based on time and memory factors with strong hashing technique like SCrypt; and concluded that AES under particular condition was much more efficient than SCrypt. Although, AES being immune to many cryptanalytic attacks, various improvements can be suggested in order to enhance its security feature. The researchers (Sachdeva and Kakkar; 2018) proposed a technique to enhance the security of AES by using three cipher keys. Increased number of keys will lead to increased encryption/decryption time that makes it very difficult and time-consuming for an attacker to crack the password by brute force attack.

(Chhabra and Lata; 2018) contrasted that AES being one of the most popular and successful algorithms used for making secure cryptosystem is still not entirely immune to some attacks like advance brute force, key-based, biclique and side-channel attacks at software or hardware level. Therefore, multiple techniques like white-box cryptography have been deployed for protecting software level attacks on AES. At the hardware level, multiple protection techniques can be used like hardware obfuscation, fingerprinting, digital watermarking and more. However, all these solutions are still not able to stand against more powerful attacks like reverse engineering attack.

The above section summarizes that AES is proven to be faster, efficient and highly secure algorithm as compared to other encryption algorithms like DES, 3DES, RSA, IDES, RC5, blowfish and some hashing algorithms like MD5, SHA1 and SCrypt under certain conditions. AES is stronger against multiple attacks like GPU, custom hardware, parallel processing, differential, linear and algebraic attack because of its memory acceleration feature. However, there are few stronger and powerful attacks like key-based, biclique, side-channel and reverse engineering attacks that can break AES. Therefore, a hybrid ap-

proach with a different combination of hashing and encryption algorithms under certain conditions may be considered to increase the strength of password protection.

2.3 Previous Work on Hybrid Algorithms

The above two sections have focused on strong cryptography techniques like hashing (SCrypt) and encryption (AES); both are individually proven durable against multiple attacks. However, they both possess some weakness against high computation attacks like custom hardware or reverse engineering attacks. This section will discuss about the multiple hybrid approaches along with the primary focus on SCrypt and AES combination. (Álvarez et al.; 2018) had proposed a model to optimise the performance of PBKDFs by employing AES - 128 in CTR mode (acts as a pseudo-random generator) and SHA3 256 as the secure hashing algorithm that generates the input for AES (128-bit key and Initialisation Vector). The researchers analyzed the model based on its security features and compare its performance with robust hashing algorithms like SCrypt and Argon2. This hybrid model was found to be faster than Argon2 for an equal amount of memory used. (Kumar and Chaudhary; 2018) had proposed a hybrid combination of BCrypt and AES in order to secure online accounts from different attacks like brute force. This model focused on the security of passwords, even if the database is compromised. The result was analyzed on the basis of parameters like encryption time and throughput, which was found to be in favour of this model. Another hybrid approach of modified Blowfish with SHA was proposed by (Gore, Meena and Purohit; 2016) for data security in the cloud. This model improves the security problems related to the transfer of files or data in cloud computing. A similar approach was utilized by (Kaur and Sharma; 2018), as they proposed a hybrid model called "HESSIS". This model used the combination of SHA3, ECC and AES to enhance the strength of data security in transit. ECC was used for generating keys to be seed as input for AES and SHA3 was used to enhance the capabilities of ECC and secure the key generation process. This hybrid model was found to be faster, highly accurate and efficient than others, that could be used to provide more accuracy and security while sharing images over the cloud. (Almorabea and Aslam; 2015) proposed a hybrid algorithm that utilized AES in Galois/Counter Mode (GCM), BLAKE2 and SCrypt. The research was based on strong key derivation functions (BLAKE2 and SCrypt) which ensure the strength of keys, to be used for AES encryption. This model was found to be more efficient and secure for data protection in both desktop and mobile applications.

With the help of above literature review, we can conclude that SCrypt is found to be the strongest hashing algorithms based on memory hard functions that can be used against multiple attacks like brute force, rainbow, dictionary and more, but still somehow vulnerable to GPU or specialized hardware attacks. This weakness can be cured by AES encryption as of its hardware acceleration feature that resists custom hardware attacks. However, it is still sensitive to some high computation reverse engineering attacks. As suggested by (Almorabea and Aslam; 2015; Álvarez et al.; 2018; Gore et al.; 2016; Kaur and Sharma; 2018; Kumar and Chaudhary; 2018), different hybrid combination of algorithms must be implemented to analyze the security of passwords. Therefore, this research focuses on a hybrid combination of SCrypt with AES, and to analyze the performance of the proposed algorithm. The result of this model leads to the advancement of password security of an online user from multiple attacks like brute force.

3 Methodology

The following section provides an overview of the methodology applied in the proposed model for the password protection of an online user from brute force attack. The main idea is to make use of the password as a primary element to generate the key and then feed that key into encryption and decryption process (Almorabea and Aslam; 2015). As per the analysis done in the literature review, it is clear that SCrypt hashing and AES encryption algorithms are proven as the strongest candidates for this hybrid model because of their features like memory-hard functioning and memory acceleration. The structure of this model is divided into three phases, which are key Derivation, AES Encryption and AES Decryption.

3.1 Key Derivation

Firstly, the key is generated with the help of a strong password based key derivation algorithm SCrypt. As shown in Figure 1, the alphanumeric password with a minimum length of 8 characters is taken from the user and then seeded as input to SCrypt hash function. The resulting hash value will be considered as the key (named S-key) of size 256 bits. This key is used as the private key for the next encryption and decryption processes. The randomness of a user password and larger key space make it difficult for an attacker to crack the password easily through brute force attack. Multiple parameters are assumed while using SCrypt such as CPU or memory cost parameter (N = 214), block size parameter (r = 8), password from user (p = 1).

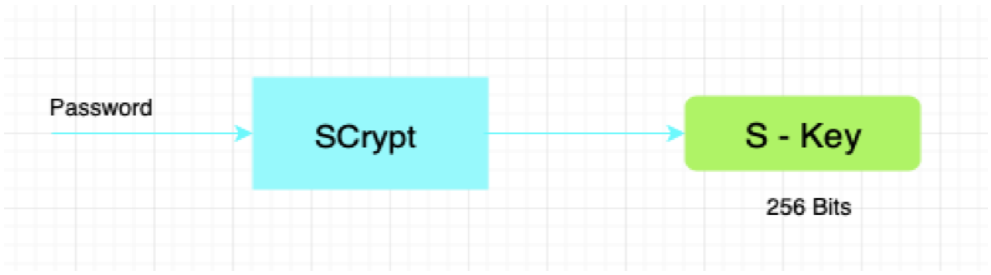

Figure 1: Key Derivation Process

3.2 AES Encryption

In the second phase, AES-256 in ECB (Electronic Code Book) mode is used for encryption in this model. The private key (S-key) generated in key derivation phase and a nonce (32 bytes) is used for AES encryption (Figure 2). A nonce is an arbitrary secure pseudo-random number that can be used only once. AES encryption takes S-key and nonce as its input and generates a cipher text. This way the user can register into the existing model. The value of nonce is unique for every password, and it is not possible to have two passwords having the same nonce value. The reason for adding a nonce to AES encryption is to ensure the confidentiality and integrity of the password. Finally, the resulting encrypted text is stored in the database.

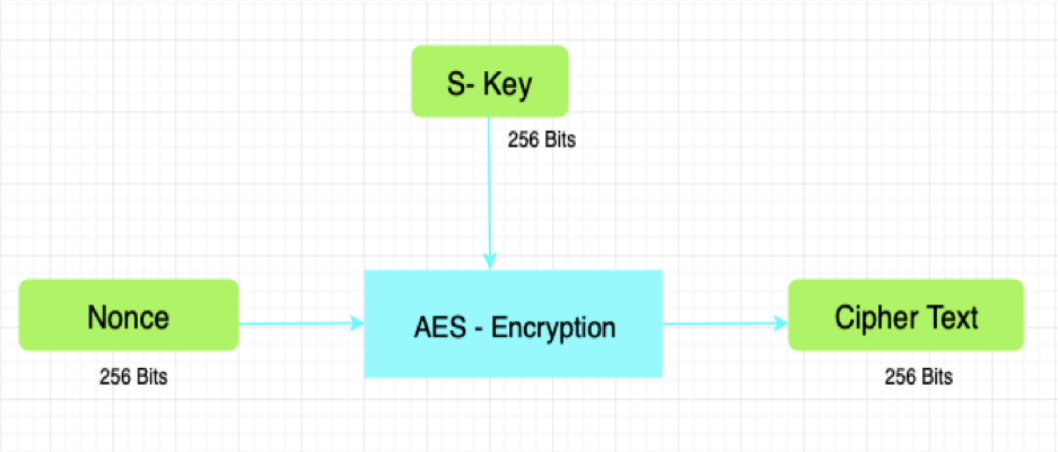

Figure 2: AES Encryption Process

3.3 AES Decryption

In this last phase, the cipher text (final complex encrypted password) extracted in the previous phase along with the same S-key is used for AES-256 decryption in ECB mode (Figure 3). If the value of resulting decrypted text is same as the value of nonce used in previous encryption phase, then the user can successfully login into the system. Somehow, if the intruder gets the encrypted cipher text generated in the last step and tampers it to get access into the system. Then at the time of login, validation never results in success. As decryption will never happen and results never match to the same nonce value. Therefore, the attacker will never get access to the system.

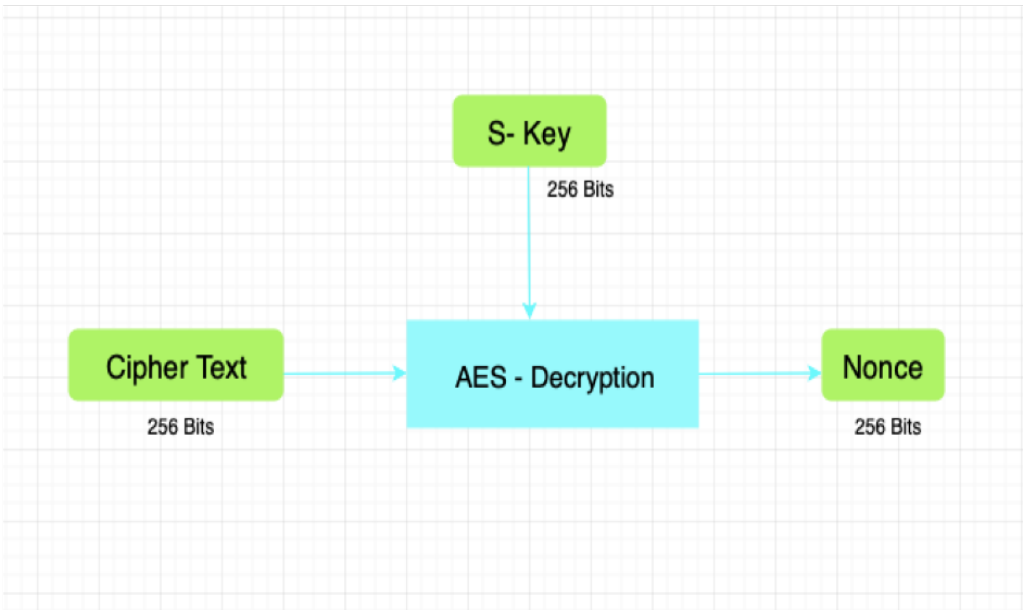

Figure 3: AES Decryption Process

4 Design Specification

This section provides detail about the design and architecture of the proposed model discussed in the methodology section. As shown in Figure 4, the architecture of this model comprises of three main parts, that are Key Derivation (using SCrypt), AES Encryption and AES Decryption.

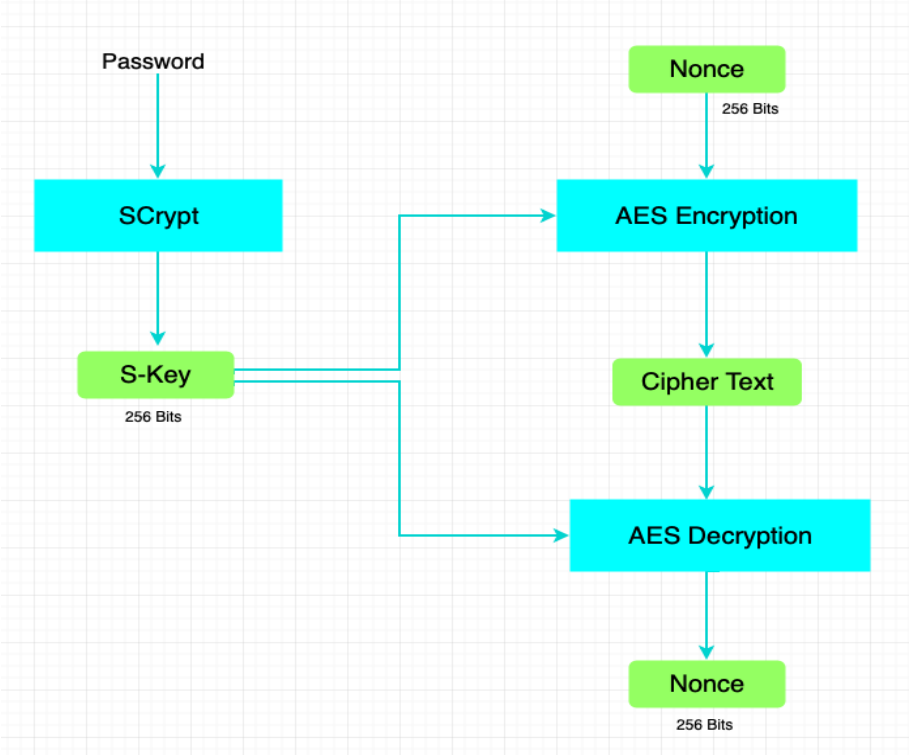

Figure 4: Architecture of the propsed model (Almorabea and Aslam; 2015)

4.1 Algorithm for the Proposed Model

Step 1: Input plain text user-name and password entered by the user.
Step 2: Check if the user already exists, if yes then proceed to Step 8 otherwise continue.
Step 3: Input user credentials like user name, password, email and contact number.
Step 4: Take password and generate a hash using SCrypt function. Set this as the S-key.
Step 5: Generate nonce using a secure random number generator function.
Step 6: Pass the key produced in step 4 and nonce as input to AES encryption function.
Step 7: Store the cipher text as the encrypted password with other details like user name, email, contact number, hash value and nonce.
Step 8: Take the plain text password and validate it with the hash value stored in the database for the corresponding user by using *SCrypt.util.check* function. If it results false then EXIT program, otherwise continue.
Step 9: Take the hash value (S-key) and encrypted password from the database for this user and pass them to AES decryption function.
Step 10: Compare the resulting value with nonce for this user.
Step 11: If the value matches then authenticate the user and allow to login into the system, otherwise EXIT program.

5 Implementation

This section describes the implementation of the proposed model. We have developed a web application in JAVA programming language by using Eclipse IDE[7]. The application is hosted locally over Apache Tomcat server[8]. MySQL Workbench[9] is used for handling the database of this web application. The work flow of the web application is illustrated in Figure 6. The homepage of the web application has two input boxes prompting for user-name and password along with two submit buttons for "Sign In" and "Register" (Figure 5). Initially, the user is required to register into the web application. A user enters the password in order to generate the private key. The password is passed through SCrypt hashing algorithm (key derivation phase). For example, a user password is "*n1ZeAJ6Bo8ZV*", then the hash value will be:

"*s0e0801nH6NZrJi2jxHcT8 + pKBIVw ==CCa2i/77ELjMmOXEQALDEMGz6f DruAE+NPKlrBZawac=*"

As explained earlier in the previous sections, if the same password is used multiple times, then web application generates different hash value every time because of random salt. This hash value will be treated as the private key (S- key) for next steps. Now, the

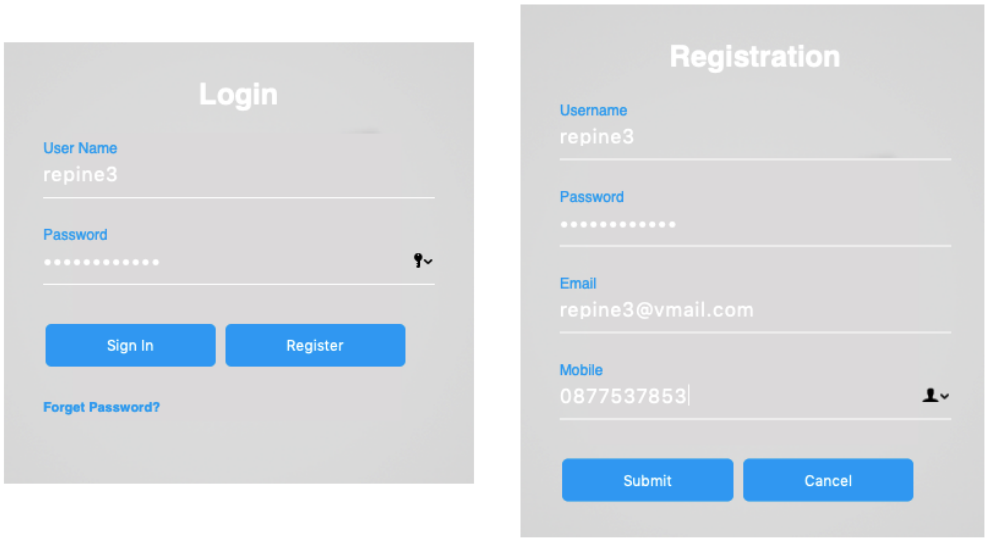

Figure 5: Login and registration page of web application

S-key along with nonce is passed through AES Encryption algorithm to encrypt the hash value and generates a complex and secure password. For example, using the above hash value and a random nonce like *"ayqWYFBjugn60mACqBcn9A=="*, the final password after encryption will be:

"rqa7qCj8LmJkn2ffpsqX2+w46xvuSa3k6TAfhOvTUw="

[7] https://www.eclipse.org/
[8] https://tomcat.apache.org/
[9] https://www.mysql.com/products/workbench/

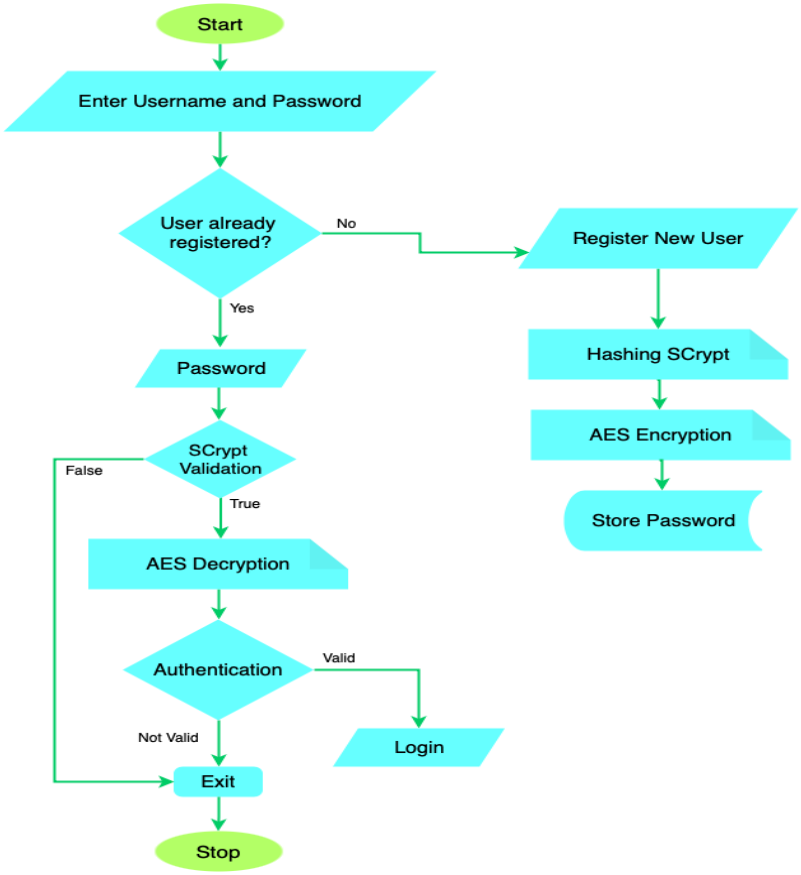

Figure 6: Flowchart of the proposed algorithm

After applying hashing and encryption, this final password is stored in the database. Apart from this password, hash value and random nonce generated for a particular user are also stored in the database to be used for the authentication process. This hybrid password is considered as more secure and difficult to crack by attackers even through brute force attack.

At the time of login (Figure 5) user needs to enter the same credentials used during registration. The password entered by the user is then passed through the authentication phase, where the password is validated with the SCrypt hash value stored in the database for the corresponding user. If the result is true, then the corresponding encrypted password and S-key are passed through the AES decryption algorithm. Finally, the resulting value is compared with the nonce value for this user. If the results match, the user successfully login into the system (Figure 7). Somehow, if the password is captured or malformed by an attacker, then decryption will never happen, and an unauthorized user cannot log in into the system. Hence, brute forcing this complex password would be more difficult for an attacker.

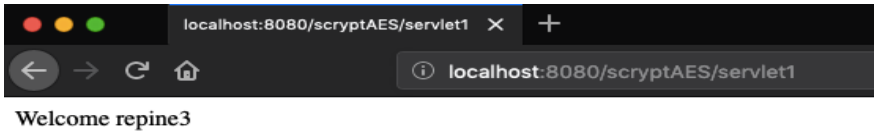

Figure 7: User successfully logged in

6 Evaluation

This section provides information about the performance analysis of the proposed model. We have ethically generated a dataset using *mackaroo*[10], comprises of 10 users with random user-name and password (shown in Table 3). The size of plain text password entered by users is calculated (in MB) in Table 3. The evaluation of this model is based on performance metrics like encryption time and throughput.

Encryption time is the total time taken to encrypt the plain text, i.e. total amount of time taken for SCrypt hashing and AES encryption together. The efficiency of a cryptography algorithm is inversely proportional to encryption time (Arora, Sharma and Engles; 2017). That means less encryption time taken by an algorithm to convert the plain text will have more efficiency.

On the other hand, the performance of an algorithm can be determined by its throughput. It can be calculated by dividing the size of plain text by encryption time (Arora et al.; 2017). It is directly proportional to performance, i.e. more the throughput, higher the performance of an algorithm.

In order to evaluate the model, we have implemented our proposed hybrid algorithm and record the encryption time and throughput for all users in the given dataset. Because of SCrypt and AES being memory hard techniques, we have studied the output in two different operating systems of different configurations (as per the limited resourced available).

Data	User Name	Password (in Plain Text)	Password Size (in MB)
Data 1	fbriffett0	AK85ISs0uj	0.00001
Data 2	vgaltone1	QQWbwI0bjH	0.00001
Data 3	pcarruth2	By3Lcab	0.000007
Data 4	repine3	n1ZeAJ6Bo8ZV	0.000012
Data 5	sbelliard4	ROFK2jw	0.000007
Data 6	ckiossel5	R3bekjnc	0.000008
Data 7	ckrolle6	FXQerp2Hd	0.000009
Data 8	lharnor7	6XeHrSZ9lUE	0.000011
Data 9	cgwin8	6qpMnzuO9	0.000009
Data 10	gsyers9	KumUXEH80	0.000009

Table 3: Dataset generated for this model

[10] `https://mockaroo.com`

6.1 Case Study 1: Mac OS

The hybrid model is implemented on *1.6 GHz (2 CPUs) Intel Core i5 processor, 4 GB 1600 MHz DDR3 RAM, Intel HD Graphics 6000 1536 MB GPU and 64-bit macOS Mojave (v10.14.5)* operating system. Results of encryption time taken by BCrypt with AES and SCrypt with AES for the same users are compared in Table 4 and Figure 8. It has been observed that SCrypt with AES takes less time as compared to BCrypt with AES. Hence, the proposed hybrid algorithm of SCrypt with AES is more efficient than others.

Data	BCrypt + AES (in Seconds)	SCrypt + AES (in Seconds)
Data 1	0.487353239	0.406827051
Data 2	0.408241016	0.206647516
Data 3	0.50283432	0.190456361
Data 4	0.40723707	0.201213828
Data 5	0.394571483	0.188463434
Data 6	0.41776396	0.199798072
Data 7	0.40431096	0.193992917
Data 8	0.408630993	0.182048292
Data 9	0.405210823	0.181761045
Data 10	0.392928871	0.196285763
Average Time	0.4229082735	0.214749428
Throughput (MB/s)	0.00002175412629	0.0000428406

Table 4: Analysis of encryption time (macOS)

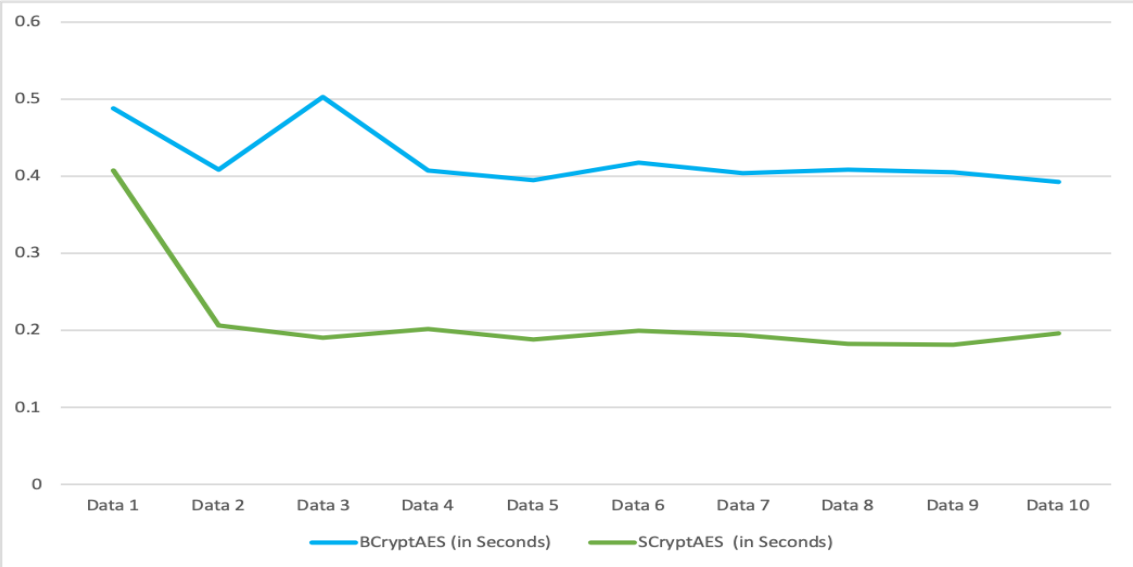

Figure 8: Encryption time (macOS)

.

It can also be observed from Table 5 and Figure 9, that throughput of SCrypt with AES is much higher than the throughput of BCrypt with AES. Hence, our proposed algorithm has high performance as compared to others.

Data	BCrypt + AES (MB/sec)	SCrypt + AES (MB/sec)
Data 1	0.0000205190	2.45805E-05
Data 2	0.0000244953	4.83916E-05
Data 3	0.0000139211	3.67538E-05
Data 4	0.0000294669	5.9638E-05
Data 5	0.0000177408	3.71425E-05
Data 6	0.0000191496	4.00404E-05
Data 7	0.0000222601	4.63934E-05
Data 8	0.0000269192	6.04235E-05
Data 9	0.0000222107	4.95156E-05
Data 10	0.0000229049	4.58515E-05

Table 5: Analysis of throughput (macOS)

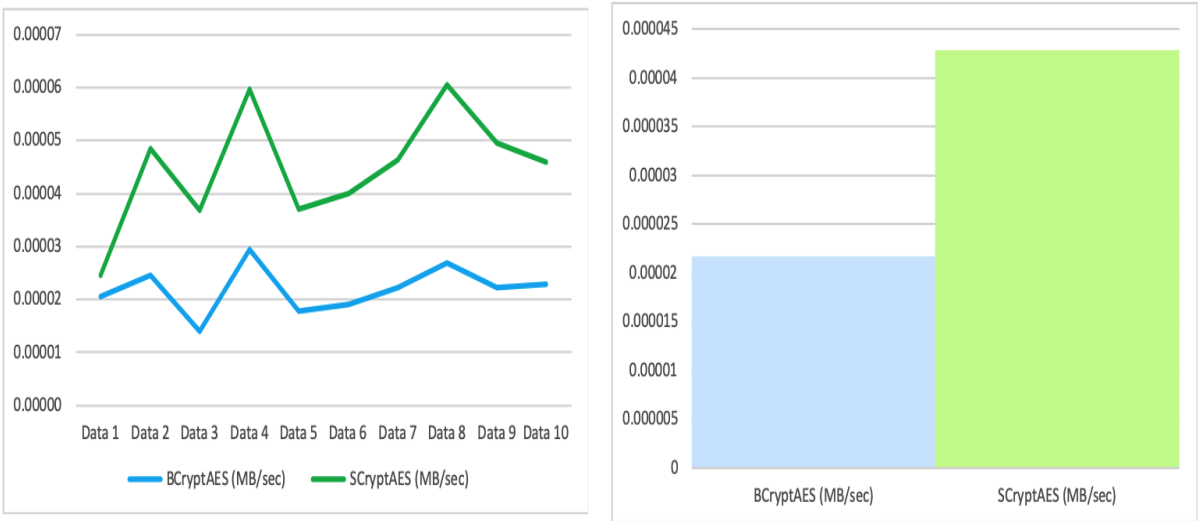

Figure 9: Throughput (macOS)

6.2 Case Study 2: Windows OS

The hybrid model is implemented on a different machine with *2.8 GHz (8 CPUs) Intel Core i7 7700HQ processor, 16 GB 2667 MHz DDR4 RAM, NVIDIA GeForce GTX 1060 6GB GPU and 64-bit Win10* Windows operating system. As shown in Table 6 and Figure 10, SCrypt with AES takes much less time as compared to BCrypt with AES. Therefore, the proposed hybrid algorithm is more efficient than others in this machine as well.

Data	BCrypt + AES (in Seconds)	SCrypt + AES (in Seconds)
Data 1	0.607414401	0.4753091
Data 2	0.2736964	0.1173043
Data 3	0.2769226	0.1096127
Data 4	0.2788966	0.1244169
Data 5	0.275399301	0.123988
Data 6	0.278819199	0.1179286
Data 7	0.2727903	0.1220418
Data 8	0.2716474	0.1212344
Data 9	0.276292301	0.1207267
Data 10	0.288506801	0.1236852
Average Time	0.31003853	0.15562477
Throughput (MB/s)	0.0000296737312	0.00005911655323

Table 6: Analysis of encryption time (Windows)

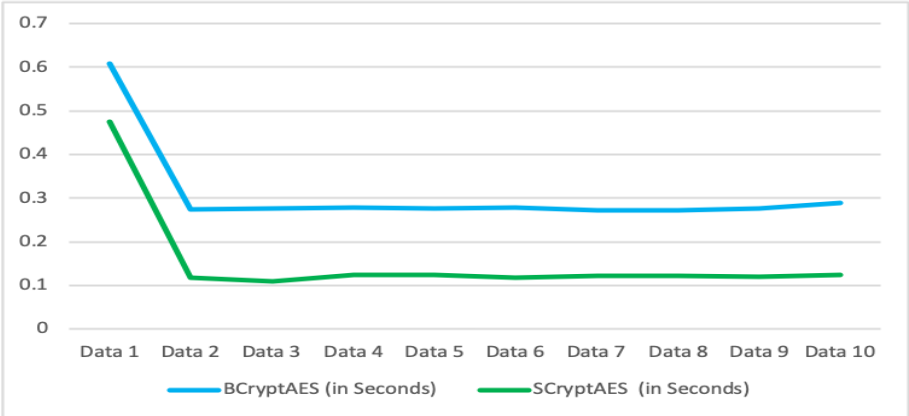

Figure 10: Encryption time (Windows)

It can also be observed from Table 7 and Figure 11, that throughput of SCrypt with AES is much higher than that of BCrypt with AES. Hence, our model has a high performance than others.

6.3 Discussion

The proposed model have been implemented on two different machines. As discussed in the literature review, BCrypt was the strongest hashing algorithm before SCrypt and Argon2. The combination of BCrypt and AES has already been researched and examined by (Kumar and Chaudhary; 2018), which showed positive results for increased security of password against brute force attack. Now, the model for this research has been compared with the previous model of BCrypt with AES. The results from the above two case studies shows that SCrypt with AES has very low encryption time and high throughput as compared to others. The more advanced the configuration of the machine, the higher the efficiency and performance of SCrypt with AES algorithm. Therefore, for this research, the proposed hybrid model of SCrypt with AES is proved to be more secure than others against brute force attack.

Data	BCrypt + AES (MB/sec)	SCrypt + AES (MB/sec)
Data 1	0.0000164632	0.0000210389
Data 2	0.0000365368	0.0000852484
Data 3	0.0000252778	0.0000638612
Data 4	0.0000430267	0.0000964499
Data 5	0.0000254176	0.0000564571
Data 6	0.0000286924	0.0000678377
Data 7	0.0000329924	0.0000737452
Data 8	0.0000404937	0.0000907333
Data 9	0.0000325742	0.0000745485
Data 10	0.0000311951	0.0000727654

Table 7: Analysis of throughput (Windows)

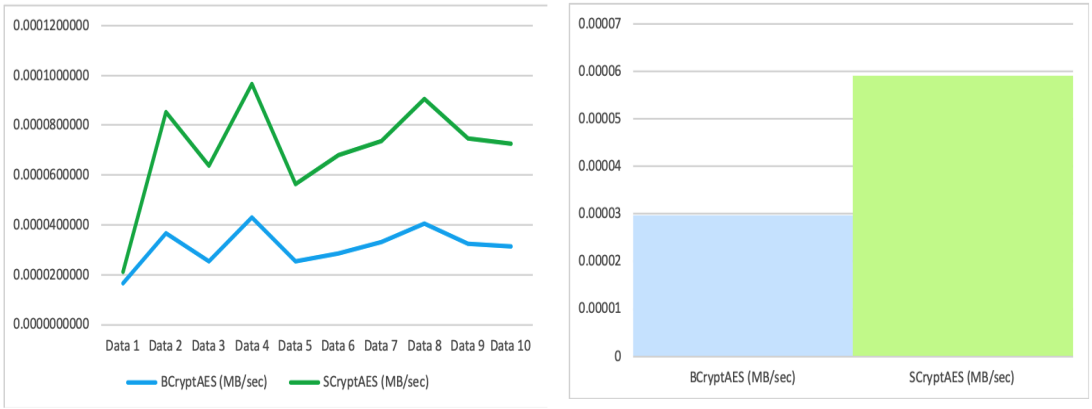

Figure 11: Throughput (Windows)

7 Conclusion and Future Work

The main objective of this research was to check whether we could enhance the password security of an online user by using a hybrid combination of SCrypt hashing and AES encryption against brute force attack. SCrypt is proven as an optimal memory hard hashing algorithm and AES is the strongest memory accelerated encryption scheme. We have increased the complexity of a password by passing it through SCrypt and then to AES in sequential order. The Results of the proposed model proves that this hybrid algorithm is highly efficient and have high performance as compared to others. This makes it more difficult for an attacker to crack the password even with brute force attack. This research concludes that the hybrid combination of SCrypt hashing and AES encryption successfully increases the password security of an online user from brute force attack.

Argon2 has been recently proven as the best hashing algorithm among all others, which may give better results with AES than SCrypt (Biryukov et al.; 2016; Hatzivasilis et al.; 2015). The future scope of this research could be to check whether the security of passwords will significantly increase by using AES with Argon2, either sequentially or parallelly. A different combination of multiple hybrid algorithms can also be considered with this model.

References

Almorabea, A. M. and Aslam, M. A. (2015). Symmetric key encryption using aes-gcm and external key derivation for smart phones, pp. 264–270.

Álvarez, R., Andrade, A. and Zamora, A. (2018). Optimizing a password hashing function with hardware-accelerated symmetric encryption, *Symmetry* **10**(12): 705.

Álvarez-Sánchez, R., Andrade-Bazurto, A., Santos-González, I. and Zamora-Gómez, A. (2017). Aes-ctr as a password-hashing function, *International Joint Conference SOCO17-CISIS17-ICEUTE17 León, Spain, September 6–8, 2017, Proceeding*, Springer, pp. 610–617.

Alwen, J., Chen, B., Pietrzak, K., Reyzin, L. and Tessaro, S. (2017). Scrypt is maximally memory-hard, *Annual International Conference on the Theory and Applications of Cryptographic Techniques*, Springer, pp. 33–62.

Arora, M., Sharma, S. and Engles, D. (2017). Parametric comparison of emds algorithm with some symmetric cryptosystems, *Egyptian informatics journal* **18**(2): 141–149.

Biryukov, A., Dinu, D. and Khovratovich, D. (2016). Argon2: new generation of memory-hard functions for password hashing and other applications, *2016 IEEE European Symposium on Security and Privacy (EuroS&P)*, IEEE, pp. 292–302.

Chhabra, S. and Lata, K. (2018). Enhancing data security using obfuscated 128-bit aes algorithm-an active hardware obfuscation approach at rtl level, *2018 International Conference on Advances in Computing, Communications and Informatics (ICACCI)*, IEEE, pp. 401–406.

Ertaul, L., Kaur, M. and Gudise, V. A. K. R. (2016). Implementation and performance analysis of pbkdf2, bcrypt, scrypt algorithms, *Proceedings of the International Conference on Wireless Networks (ICWN)*, The Steering Committee of The World Congress in Computer Science, Computer , p. 66.

Gore, A., Meena, S. and Purohit, P. (2016). Hybrid cryptosystem using modified blowfish algorithm and sha algorithm on public cloud, *International Journal of Computer Applications* **155**(3): 6–10.

Hatzivasilis, G., Papaefstathiou, I. and Manifavas, C. (2015). Password hashing competition-survey and benchmark., *IACR Cryptology ePrint Archive* **2015**: 265.

Kant, D. C. and Sharma, Y. (2013). Enhanced security architecture for cloud data security, *International journal of advanced research in computer science and software engineering* **3**(5).

Kaur, J. and Sharma, S. (2018). Hessis: Hybrid encryption scheme for secure image sharing in a cloud environment, *International Conference on Advanced Informatics for Computing Research*, Springer, pp. 204–216.

Kumar, N. and Chaudhary, P. (2018). Password security using bcrypt with aes encryption algorithm, *Smart Computing and Informatics*, Springer, pp. 385–392.

Marton, K., Suciu, A. and Ignat, I. (2010). Randomness in digital cryptography: A survey, *Romanian journal of information science and technology* **13**(3): 219–240.

Musliyana, Z., Arif, T. Y. and Munadi, R. (2015). Security enhancement of advanced encryption standard (aes) using time-based dynamic key generation, *ARPN Journal of Engineering and Applied Sciences* **10**(18).

Padmavathi, B. and Kumari, S. R. (2013). A survey on performance analysis of des, aes and rsa algorithm along with lsb substitution, *IJSR, India* .

Percival, C. (2009). Stronger key derivation via sequential memory-hard functions.

Percival, C. and Josefsson, S. (2016). The scrypt password-based key derivation function.

Sachdeva, S. and Kakkar, A. (2018). Implementation of aes-128 using multiple cipher keys, *International Conference on Futuristic Trends in Network and Communication Technologies*, Springer, pp. 3–16.

Singh, G. (2013). A study of encryption algorithms (rsa, des, 3des and aes) for information security, *International Journal of Computer Applications* **67**(19).

Sriramya, P. and Karthika, R. (2015). Providing password security by salted password hashing using bcrypt algorithm, *ARPN Journal of Engineering and Applied Sciences* **10**(13): 5551–5556.

Turan, M. S., Barker, E., Burr, W. and Chen, L. (2010). Recommendation for password-based key derivation, *NIST special publication* **800**: 132.

Widiasari, I. R. (2012). Combining advanced encryption standard (aes) and one time pad (otp) encryption for data security, *International Journal of Computer Applications* **57**(20).