A Hybrid approach for Augmenting password security using Argon2i hashing and AES Scheme.

MSC Internship project
CyberSecurity

Raja Sekhar Reddy Modugula
Student ID: x17151911

School of Computing
National College of Ireland

Supervisor: Mr Ross Spelmen

National College of Ireland
Project Submission Sheet
School of Computing

Student Name:	Raja Sekhar Reddy Modugula
Student ID:	x17151911
Programme:	MSc in Cyber Security
Year:	2020
Module:	MSc Internship project
Supervisor:	Mr Ross Spelmen
Submission Due Date:	28/09/2020
Project Title:	A hybrid approach for Augmenting password security using Argon2i hashing and AES Scheme.
Word Count:	5661
Page Count:	18

Signature:	
Date:	28th September 2020

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.	☐

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

A Hybrid Approach for Augmenting password security using Argon2i Hashing and AES Scheme.

Raja Sekhar Reddy Modugula
x17151911

Abstract

In our daily lives, passwords are the primary form of authentication on various web applications and stored in a hashed format in the database. Password security only using hashing algorithms is cracked with password cracking attacks like Brute force attack, Dictionary attack, and Rainbow Tables. In our research project, we have developed a hybrid model, where the password is hashed with Argon2i hashing and encrypted using AES encryption scheme to provide strong security against brute force attacks. To increase the security level of the encryption key, we have derived the crypt-key from PBKDF2 and hashed it again using SHA-256. The performance of our proposed model has outperformed existing hybrid approaches like Bcrypt with AES and Scrypt with AES in terms of Encryption time, Decryption time, and Throughput.

1 Introduction

1.1 Background and Motivation

A Data breach is a security violation where user sensitive and confidential data can be accessed by an unauthorized individual. Data breaches are exposing an individual as well as the company's credentials like Email addresses, passwords, date of birth, credit card numbers, social security numbers and other forms of personally identifiable information. [1]Self key is a blockchain-based identity system, allowing individuals and corporations for controlling and managing their data. [2]Data breaches reported by Self key from 1st quarter of 2020 is a huge list of organisations including One Class, BlueKai, Postbank, Keepnet Labs, AIS, Chartered Professional Accounts of Canada (CPA), True caller, LiveJournal, Virgin Media, Nintendo, GoDaddy, Microsoft, Facebook, OnePlus, T-Mobile and others were listed as the worst data breaches with more than 8 billion exposed records. BlueKai is one of the largest banks for web tracking data owned by Oracle. This company uses website cookies combined with other tracking technology for following user web activity and sells the data to companies and firms. Over a period, on June 19, 2020, the web tracking data is exposed due to lack of passwords including names, addresses, email address and web browsing activity. Another leading online learning platforms of India called Unacademy experienced a massive data breach on May 3, 2020, when an attacker gained database access and sold more than 20 million users account information including Names,

[1]https://selfkey.org/about-us/
[2]https://selfkey.org/data-breaches-in-2019/

emails, passwords. [3]Thycotic's cybersecurity blog have updated with a survey conducted on social network habits of security professionals. The five insights highlighted in the survey are shocking as follows, where 50% have not changed Social networking passwords for a year, 20% have never changed their passwords ever. 30% are still using birthdays, pet names and addresses as their work passwords, 25% claim to change their work password only if the system tells them. In the end, 45% think privileged passwords are the reason behind the cyber-attacks against their companies. Broken Authentication and Session management are one of the OWASP top 10 security attacks (Garver; n.d.), where an attacker tries to access personal information by finding glitches in the authentication process or session management. To overcome broken authentication, developers need to apply hashing and encryption mechanisms for passwords, session ID and other sensitive information. (Ntantogian et al.; 2019) Brute force and Dictionary attacks(Sood et al.; 2009) are most popular attacks in open source web platforms. In cryptography, there are several algorithms for password hashing (Bellovin and Merritt; 1993), (Ntantogian et al.; 2019), and (Ertaul et al.; 2016) namely, PBKDF2, Bcrypt, Scrypt, Argon2, SHA1 etc. for converting the plain text password to hashed format. There are various Encryption schemes in cryptography (Padmavathi and Kumari; 2013), (Sachdeva and Kakkar; 2018), (Singh; 2013), and (Sriramya and Karthika; 2015) namely, AES, DES, 3DES, RSA schemes for converting plain-text into ciphertext with the help of key. In this paper, we concentrate on Argon2i hashing, AES (Advanced Encryption Standard) encryption and combination of them to enhance the password security against offline password cracking attacks like Brute force and Dictionary attacks.

1.2 Motivation: Passwords stored in the form of plain-text are not secure and cracked using broken authentication and session management vulnerabilities. This paper is based on related work by (Bidhuri; 2019), who have combined the Scrypt hashing and AES encryption methods to form a hybrid model to successfully increase the password security of online user from Brute force attacks. The future scope of their research was to replace Scrypt with the award-winning hashing algorithm ARGON2 and combine with AES encryption and check if it can improve the password strength. That gave us purpose and Motivation in forming the hybrid model by using AES with ARGON2i in a PHP based web application for password security against Brute Force attacks. Also, the research works on hybrid models are limited to fewer numbers.

1.3 Research Question: How to enhance password security using a hybrid combination of Argon2i with AES encryption in online social networks?

1.4 Research Variable: To augment the password security in online social networks, we have developed a social networking application to implement the combinational approach of Argon2i hashing and AES encryption scheme for user passwords. We have used the programming languages like HTML, CSS, JavaScript, Bootstrap, PHP, Ajax and Xampp server within the development of a web application.

The format of this paper is as follows; Section 1 Introduction will introduce the topic of the project, research motivation and research question. Secondly, section 2 Related Work explains the literature review of the critical and analytical overview of techniques and methods reviewed until now. Section 2.1 password cracking attacks, types of password cracking, strategies and Methods. Section 2.2 password protection system using hashing algorithms, earlier studies on different hashing techniques and why is Argon2i chosen for our research project. Subsection 2.3 password protection based on encryption techniques; related work is done on different encryption algorithms and features of various encryption

[3]https://thycotic.com/company/blog/2017/

schemes, why we have chosen Advanced Encryption standard scheme. Section 2.4 Earlier studies performed on hybrid cryptographic approaches used, why Argon2 hashing and AES encryption is right for enhancing password security. In section 3: Methodology explains the stages of password hashing, encryption, and decryption. Section 4 Design Specification supplies the design and architecture of the proposed model from the method section. Section 5 gives the Implementation of the proposed model with the help of the flow chart. Section 6 Evaluation is performed and listed with the results of the implementation. Section 7 Conclusion and Future works.

2 Related Work

This section will give the earlier studies on Password cracking attacks, password hashing techniques, password encryption schemes and combination of password hashing techniques with encryption schemes.

2.1 password cracking attacks

(Ntantogian et al.; 2019), (Yu and Huang; 2015) have defined password cracking or guessing attacks as a process of recovering the password from a cipher-text or hash value by an attacker. There are two types of password cracking attacks namely online and offline. In online attacks, an attacker can create a script file or automated program that will be executed to try every password in the list supplied and when matched gives the access to the attacker. On the other hand, an attacker can have a database of user's password hashes in his custody, where he can try to crack every password of the user by comparing the password guess with hash values. Authors (Ntantogian et al.; 2019), (Yu and Huang; 2015) have done their research on offline cracking attacks because offline cracking is comparatively faster than online attacks. (Ertaul et al.; 2016) have categorized the offline attacks into three types which included brute force attack, dictionary attack, and rainbow tables. Hardware platforms which are supporting password cracking like GPU (Graphical processing unit's), FPGA (Field programmable gate arrays) working are explained in detail. They have also explained the existing usage of CMS originated websites like WordPress, Joomla, Drupal, Magento, PrestaShop, TYPO3, and OpenCart. Among the existing CMS, WordPress holds a 31.3 percent market share out of all the websites available on the internet and 59.8 percent market share associated with CMS. (Ntantogian et al.; 2019) have also listed the web application frameworks based on GitHub which uses only open-source projects. A framework proposed by (Ntantogian et al.; 2019) takes parameters of password cracking for both brute force and dictionary attacks. Similarly (Yu and Huang; 2015) have researched Strategies of password cracking, offline attacks and working of offline attacks like Brute force attack, Dictionary attack, Rainbow table cracking, password cracking on HPC (high-performance computer). In the comparison of (Yu and Huang; 2015) research with (Ntantogian et al.; 2019), we can say that (Ntantogian et al.; 2019) is more advanced paper in terms of addressing offline cracking attacks by considering the main parameters of brute force and dictionary attacks, identified default hashing schemes of CMS and web application frameworks, outdated hash functions, comparative security analysis between CMS and web application frameworks. While (Yu and Huang; 2015) was only limited to types and integrating methods to increase the cracking speed of offline password cracking attacks.

2.2 Password protection system using hashing algorithms

This section describes the earlier studies on Argon2, and other hashing algorithms used for enhancing password protection. (Ertaul et al.; 2016) have stated that hashing functions and encryption schemes are two related complementary fields which cannot be used as a replacement for one another. Traditional hashing for storing and securing passwords like SHA256, SHA512 (Sumagita et al.; 2018), WHIRLPOOL, and RipeMD are still susceptible to cracking attacks and recovered with the help of Brute Force, Dictionary attacks, lookup tables, reverse lookup, and Rainbow tables (Ertaul et al.; 2016). Therefore, (Ertaul et al.; 2016) has used PBKDF2, Bcrypt and Scrypt include a security factor called iteration count. Secondly, PBKDF2, Bcrypt and Scrypt are salted hashing functions which makes it extremely difficult to pass the security. Argon and Argon2 by (Biryukov et al.; 2015), designers from the University of Luxembourg have explained the award-winning algorithms at (PHC) password hashing competition. They claim that Argon was originally their submission for PHC, which is a multipurpose hashing function that aims for highest trade-off resilience because any small reductions in memory would lead to serious time and computational penalties. Unlike Argon, Argon2 is streamlined and simple to design. Argon2 is designed to fill the highest memory, effective use of multiple computing units, and supply security against trade-off attacks. Argon2d and Argon2i are two variants of Argon2 in which Argon2d is fast and depends on data memory access, suitable for cryptocurrencies and applications without side-channel timing threats. The major difference between Argon2i vs Argon2d is that Argon2i is data-independent and essential for password hashing and password-based key derivation. To overcome the problems of existing schemes like Scrypt, which is complex, difficult to analyze, nonflexible in separating memory and time costs. Therefore, (Biryukov et al.; 2015) recommends Argon2 for highest resilience trade-off applications which guarantee prohibitive time and computational penalties on implementation of memory reduction. Secondly,(Biryukov et al.; 2015)Argon2 is used in high-performance applications because both versions 2d and 2i are capable of filling 1GB of RAM in few seconds, scales easily for parallel computing units and its optimized design for easy analysis and implementation. Another study by (Biryukov et al.; 2016) also suggest practical settings for both Argon2d and Argon2i as follows: Argon2d practical settings are regular for cryptocurrency and Back-end server authentication. While Argon2i is enhanced for more dangerous settings with key derivation for hard-drive encryption and Front-end server Authentication.

According to (Widiasari; 2012), passwords are the primary aspect of a user account system and the best way to secure passwords is by implementing salted password hashing in online shopping websites. They proposed a Bcrypt hashing function, which is an adaptive function based on Blow fish cipher. Bcrypt can adapt over time, increase iteration count to make it slower and computationally Strong against Brute Force attacks. The developers Provo's and Maiziers have worked on the security standard for password hashing as it is derivative from Blow fish block cipher which uses lookup tables started in the memory to generate the hash. Overall they have discussed four different methods to prevent password security attacks namely, 1. recommended use of the strong passwords to drop the probability of its existence in the dictionary, 2. Salts, 3. Stretching key and Iteration hashing to make slower computations and 4. Use of Chaining method and different initialization vector for every password.

Similarly, (Boonkrong and Somboonpattanakit; 2016), and (Khowfa and Silasai; 2019) have done research on salt generation, placement for secure password storing, and po-

sition of placing salt values. (Boonkrong and Somboonpattanakit; 2016), states that cryptographic hash functions such as MD5 and SHA-1 are not suitable for storing strong passwords and placing a dynamic salt into a password makes it harder for an attacker. Initially, a password is used as input to obtain the hash value using MD5. In the next step, the password and hash value are converted to the binary format where the least significant bit is used in XOR operation in the next step to finding out their salt placement pattern. They have also created a set of 5 salt placement rules to integrate with their proposed algorithm to secure the password. In contrast with (Boonkrong and Somboonpattanakit; 2016) work, (Khowfa and Silasai; 2019) also added salt before performing the hash function and addressed two research goals namely, 1. To enhance the security of weak passwords, 2. To evaluate, if the position of placing salt value has significant to the strength of the password. To evaluate their proposed model, Hashcat-4.0.1 is used as password attacking tool and experimental results from their studies clear that, the position of placing salt in front or after the password is not related to strength in their test. Therefore, we can come to an assumption that securing a password can be possible with adding salt to the password either before or after hashing.
Another research by (Katrandzhiev et al.; 2019), on password protection methods for web-based platforms like PHP and MySQL states that MD5 and SHA1 are not recommended for password security because these algorithms cannot be developed to increase complexity and easily breached using rainbow tables. And for lower PHP versions like 5.5 and above, Bcrypt is recommended as rainbow tables cannot attack it. Unlike rainbow tables, Brute force attacks are used to attack Bcrypt hashing, but the time required for hashing is doubled by every cost parameter increased. In comparison with Bcrypt with cost only as a parameter, Argon2 can lay in increasing computational powers of GPU and ASIC devices with multiple parameters. To determine parameters three steps are recommended as follows, 1. Find the highest amount of threads the system can use for hashing, 2. Finding a large amount of memory for each call for hashing and 3. Determine reasonable execution time and number of passes according to system load and server configuration. Thus,(Katrandzhiev et al.; 2019) conclude by saying that Argon2 is highly recommended for projects using PHP version 7.2 and above with high-end security like hashing and encryption.

2.3 Password protection system using Encryption algorithms

This section will focus on earlier studies of other cryptographic techniques like Encryption schemes. The researcher (Liu et al.; 2019) claims that the primary way to expose any password is by storing it in the form of plain-text or by implementing improper encryption. DES (Data Encryption Standard) is replaced with the advancement in the symmetric block cipher algorithm called AES (Advanced Encryption Standard) scheme. AES works on state-based transformations and encrypts any given data by iterating through the input plain-text and key from round function. There are four major operational steps involved namely Sub Bytes, Shift Rows, Mix Columns, Add Round Key and AES supports a key length of 128bits or 192bits or 256 bits. (Liu et al.; 2019) has proposed a set of password storage and transmission encryption model where the proposed model is implemented by developing the main key and working key. The main key is used to encrypt the key while working key is to encrypt the password and updates automatically at regular intervals. AES is implemented as a transmission encrypted model for password protection and compared with RSA Encryption scheme in terms of key management, operating speed of Encryption and Decryption, security and proved that AES is inferior and can be implemented by software at high speed. During the process of password storage encryption, the key management module gets the key required from the key database and sends it along with the plain-text password to password storage module for AES encryption and stores the generated ciphertext in the database. To evaluate the performance of both AES and RSA algorithms, four sets of passwords were encrypted, decrypted and time has been calculated. Thus the 10-round improved AES algorithm is capable of increasing its encryption speed by 52.9% and decryption speed by 36.2% in comparison with the RSA algorithm. Another research by (Padmavathi and Kumari; 2013) proposed and implemented a combination of cryptographic encryption algorithms and steganography for securing data while transmitting through a network. The data which is transmitted from the sender is encrypted using three encryption techniques namely DES, AES, and RSA cryptographic algorithms. In the next step, the data is hidden in an image with the help of a steganographic algorithm called LSB substitution technique. To evaluate the performance of the proposed model, A comparative analysis of three algorithms based on encryption time, decryption time, and buffer size is conducted. To ensure the security of data, encryption techniques are well studied and analysed for improving performance. Thus (Padmavathi and Kumari; 2013) states that based on the experimental results, the AES algorithm has out-performed DES and RSA algorithms in parameters like encryption time, decryption time, and buffer usage. (Sachdeva and Kakkar; 2018) have proposed a scheme which improves the AES-128 using multiple cipher keys and key-dependent S-boxes for preventing different crypt analytic attacks like linear, differential crypt analysis and side-channel attacks. In their approach, the input data file is divided into data blocks and encrypted using conventional AES approach which uses a static key of 16 bytes. 10 rounds of iterations are made for the primary encryption using AES-128 algorithm and the resultant cipher text is used as plain-text for the second round of encryption. The second encryption is using Random key and a key-dependent S-box, where the random key of 16-byte values is generated by built-in functions of MATLAB and the key is sent to key expansion routine for generating sub keys. Therefore, the AES-128 and their proposed approaches are compared based on performance metrics like execution time, avalanche effect test, and strict avalanche criterion test. The advantages with their proposed approach are showing better Avalanche effect and SAC than conventional approach while, the major drawback seen was an increment in execution time due to the complexity of their proposed model. (Singh; 2013) has surveyed the study of encryption algorithms namely RSA, DES, 3DES, and AES for enhancing information security. Their survey explains the detailed study of requirements to secure the data over a network using most popular encryption methods. Based on the survey and literature review conducted, it is significant that the AES algorithm is efficient in parameters like Speed, Time, Throughput, and Avalanche effect.

2.4 Password protection using hybrid models

(AbdElnapi et al.; 2016) researched data storage in cloud computing using a hybrid hashing security algorithm. Cryptography is one of the most popular practices used for security of data storage and enables security services such as Confidentiality, Integrity, and Non-repudiation. To ensure the security services, they have proposed and implemented a hybrid encryption algorithm (AES and RSA) with a secure hashing algorithm (SHA256). (AbdElnapi et al.; 2016) claims that data storage provided by cloud service providers must ensure the main criteria of security and to achieve security criteria, a combination of hybrid algorithms and hash functions are one of the practical protection solutions. The hybrid algorithm (AES and RSA), and proposed hybrid-SHA256 are executed with different data input sizes like 34kb, 67kb, and 93kb. The best results for both encryption and decryption concerning the hybrid encryption algorithm (AES and RSA) is 32% and 33% respectively. In comparison with the hybrid algorithm, the proposed hybrid-SHA256 model requires more time for encryption and decryption, however, the proposed model is more secured due to hashing and digital signature concepts involved in developing it. Another research on Hybrid approach for data hiding is proposed by (Kaur and Malhotra; 2013) used a combination of Advance hill cipher and DES to ensure the security of data hiding, where Data hiding or text hiding is hiding the data in the form of an image. The proposed model focus on preventing malicious users to increase security using MATLAB software. The steps involved in the proposed model are as follows; Select cover image, hide text, give password up to 24 characters and numbers, Embedded image, give return same password with security, retrieve hidden text. To evaluate the effectiveness of the system, measuring factors MF1 and MF2 are calculated to stand for PSNR and MSE values for quality checking of the images. A novel approach for authentication using multi-level password system is proposed by (Chhetri; 2020) by implementing three tiers of authentication like encryption of text password using AES, colour combination using encryption based on arithmetic operation, and picture password using segmentation and re-sizing algorithm. In the stage of text password encryption, a column matrix of 4x4 size state matrix is used for operating AES. In the second stage of colour combination, there are three buttons (red, green, and blue) with one fixed buffered numeric value for each button, upon clicking the button the buffered value will go to the text box for concatenating with previous value using Masking Algorithm. Before moving to the next authentication, it first retrieves the data from the database and decrypts it with the arithmetic operation for comparing and moves to the next round if it's correct. In picture password the segmentation and re-sizing algorithm, resize the image to a fixed size and then segment that image into 100 invisible clickable buttons, has a fixed buffer value masked from the user. Thus, the proposed system by (Chhetri; 2020) enables user security for windows folder with three levels of passwords for strong authentication and is suitable for personal computers, servers and systems which require authentication before granting access. Authors (Kumar and Chaudhary; 2018) and (Bidhuri; 2019) have researched password protection using Scrypt with AES algorithm respectively. (Kumar and Chaudhary; 2018) have proposed a working model, where the user opens a login form and enters user credentials like usernames and passwords stored in a list, the password is hashed using Bcrypt hashing algorithm. The hash value obtained is used as a key for the AES algorithm in performing encryption and decryption processes and evaluated using performance matrices of encryption time and throughput time. Similarly, (Bidhuri; 2019) researched enhancing password security using Scrypt hashing function and AES encryption algorithm sequentially for preventing password cracking attacks like Brute force attacks. A data set created is tested for evaluation of the proposed model in terms of encryption time and throughput for both windows, MAC operating systems and also compared results with existing Bcrypt with AES encryption for performance.

3 Methodology

In this section an analysis of methods applied in the proposed model for enhancing password protection for Online social network users from password cracking attacks. The main idea of research was using a user supplied password for key derivation using Argon2 key derivation function and then use the key for AES encryption and decryption processes. This idea is obtained from future scope of (Bidhuri; 2019). Based on the related work in the literature review, we can see that Argon2 is the winner of password hashing competition (Wetzels; 2016), and (Kumar and Chaudhary; 2018) AES is the strong encryption scheme which outperformed all existing schemes based on experimental results in (Liu et al.; 2019), (Padmavathi and Kumari; 2013), and (Singh; 2013) works. In the proposed model we are implementing Argon2 hashing based on key derivation function with combination of AES encryption process to form a hybrid model for enhancing password security against cracking attacks. To add another layer of protection, we have hashed the resultant cipher text generated by AES encryption and Hash key generated by ARGON2i using sha256 algorithm in HMAC method. The framework of this model is organised into three phases namely, 1. Key Derivation Phase, 2. AES Encryption Phase, and 3. AES Decryption Phase.

3.1 Key Derivation Phase

In this key derivation phase, the key is generated with the help of password-based key derivation function called Argon2i. (Wetzels; 2016) Argon2i hash function is the evolution of Bcrypt and Scrypt algorithms because it supplies security against Brute force attacks using parameters like predefined memory cost=2048, time cost=4, and threads=3 for security towards GPU attacks. In this stage, the password supplied by the user is given as input to Argon2i hashing function for obtaining the hashed password (hash key) and the hashed password is encrypted using AES encryption.

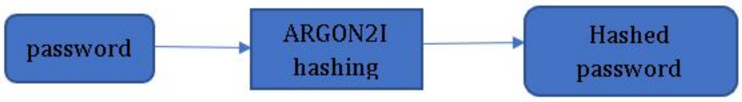
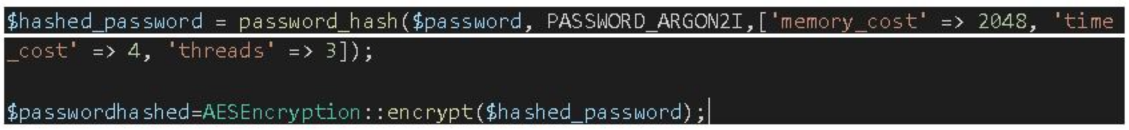


Fig: 1 password hashing using Argon2i



```
$hashed_password = password_hash($password, PASSWORD_ARGON2I,['memory_cost' => 2048, 'time
_cost' => 4, 'threads' => 3]);

$passwordhashed=AESEncryption::encrypt($hashed_password);
```

3.2 AES Encryption Phase

In this AES encryption phase, we have generated Cipher text (cipher data) using openssl encrypt function with parameters like plain text message, AES-256-CBC method, key(salt), iv (initialization vector of 16 bytes) and OPENSSL RAW DATA. The resultant cipher text along with initialization vector iv, hash key (password hashed using ARGON2i) is used for generating keyed hash value using HMAC method and is encoded using base 64 encoding scheme before storing the password in database. The figure below shows the password encryption phase using AES-256-CBC.

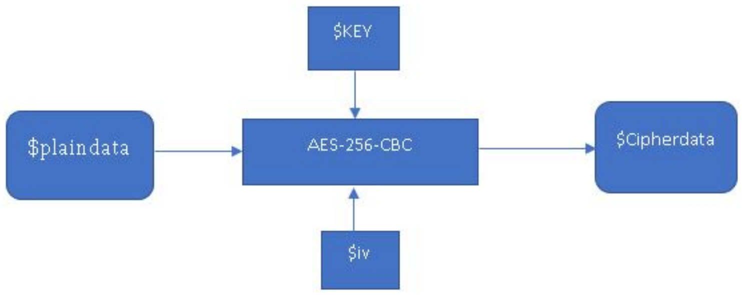
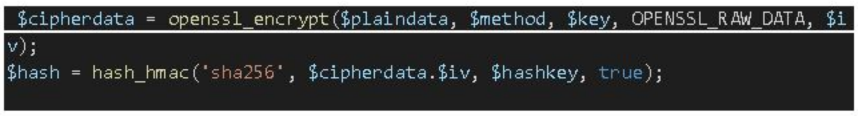


Fig: 2 password encryption using AES-256-CBC



```
 $cipherdata = openssl_encrypt($plaindata, $method, $key, OPENSSL_RAW_DATA, $i
v);
$hash = hash_hmac('sha256', $cipherdata.$iv, $hashkey, true);
```

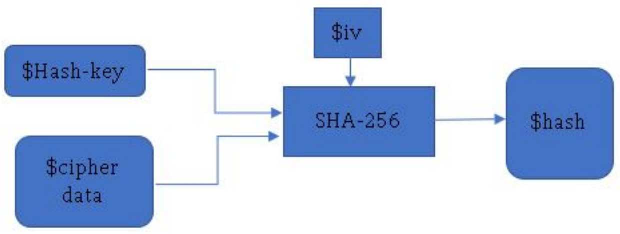


Fig: 3 Hash-key and cipherdata are hashed using SHA-256

3.3 AES Decryption Phase

In this AES decryption phase, firstly we decode the encrypted data using base 64 decode. Secondly, we generate a keyed hash value using HMAC method and check with hash if the hash value is valid or not. In the next step we plain-text (plain data) using openssl decrypt function with parameters like Cipher-text message, AES-256-CBC method, key (salt), iv (initialization vector), and OPENSSL RAW DATA.

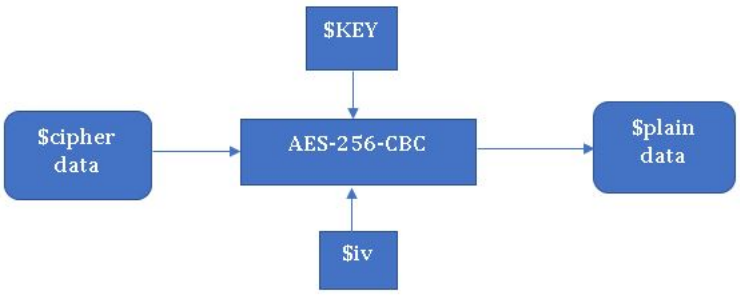
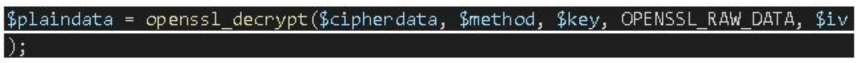


Fig: 4 password decryption using AES-256-CBC



```
$plaindata = openssl_decrypt($cipherdata, $method, $key, OPENSSL_RAW_DATA, $iv
);
```

4 Design Specification

In this section we will discuss in detail about design and architecture of our proposed model discussed in the method section. Figure: IV below shows the architecture of the proposed model and its main sections.

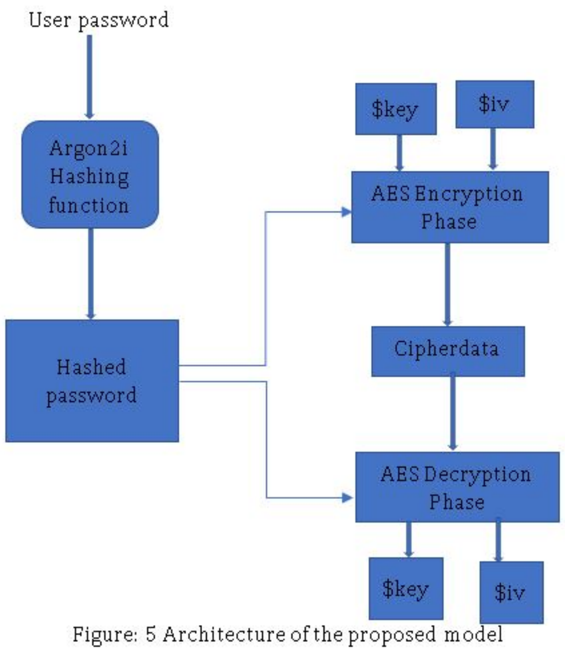


Figure: 5 Architecture of the proposed model

4.1 Steps for the proposed Algorithm

Step 1: Login credentials namely username and password were inserted as input by the user.
Step 2: checks if the user is already registered, if yes then continue to login and proceed to step 8 for verification of password.
Step 3: Input user credentials like first name, last name, email-id, password while registering as a new user.
Step 4: Password is used to generate hash value using Argon2i hashing function and stored it as a $hashed_password.
Step 5: Salt is generated using a Secure random number generator function and stored as $key.
Step 6: plaintext password , $key(salt) along with $iv(16bytes) together is given as input for AES encryption phase.
Step 7: The generated $cipherdata from AES encryption and generated $hashed password are hashed again using HMAC method (sha-256) to generate another hashed key($hash) and base64 encoded to save it in database.
Step 8: decode using base64 and then decrypt AES encryption and check PASSWORD_ARGON2I test valid. If valid continue, else EXIT.
Step 9: Input the $cipherdata, salt($key), and $iv from the database to AES decryption phase for retrieve the $plaindata.
Step 10: Compare the obtained salt value with the salt for this corresponding user.

Step 11: Authenticate the user and allow him to login if the salt value is matched, else EXIT program.

5 Implementation

In this section, we will see the implementation of proposed model. An online social networking web application has been developed using programming languages namely, HTML, CSS, JavaScript, Bootstrap, Ajax, PHP, and Xampp which is the most popularly used for Locally hosting and PHP development environment with built-in PHP and MySQL. MySQL is used as database for handling the user data for this web application. The user is supplied with login page and registration page in the homepage of the web application. In the login page, user is asked for credentials like username, password and logged in using 'Login' button. If the user is new to the application, he can register by supplying first name, last, name, email, confirm email, password, confirm password to successfully complete registration. The below figures V and VI shows the images of Login page and Register page. Firstly, the user entered password is converted into Hash-key by using strong key derivation function in PHP 7.2 called Argon2i. Secondly, the generated hash-key using Argon2i along with Salt ($key) value generated is given as input to AES encryption phase to obtain a Final password which is stored in database after encoded with base64. After implementing hashing and encryption processes, the Cipher Text ($cipherdata), hash-key ($hashed_password), Salt ($key) are also stored in database for authentication purpose. If the user want to login, then he/she must enter the username and password supplied during the time of registration. The password supplied by the user is checked with Argon2i hash value stored in the database. After checking if the results are true, then the resultant Cipher Text and Hash-key are passed through AES decryption phase. After decryption, the obtained Salt value is compared with the salt value stored in the database. If the results of the salt value are matched, user is successfully logged in to the application. This hybrid combination of hashing with encryption is considered to be more complex and difficult for attackers to perform cracking attacks like Brute Force attacks.

Fig: 6 Login Page

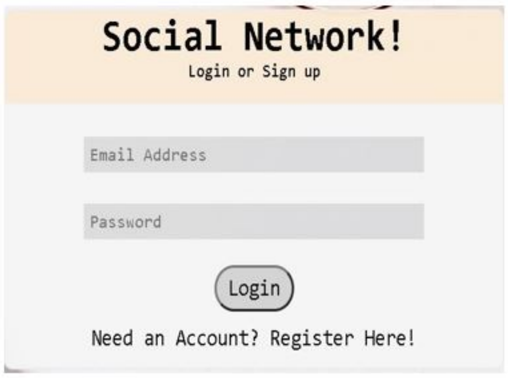
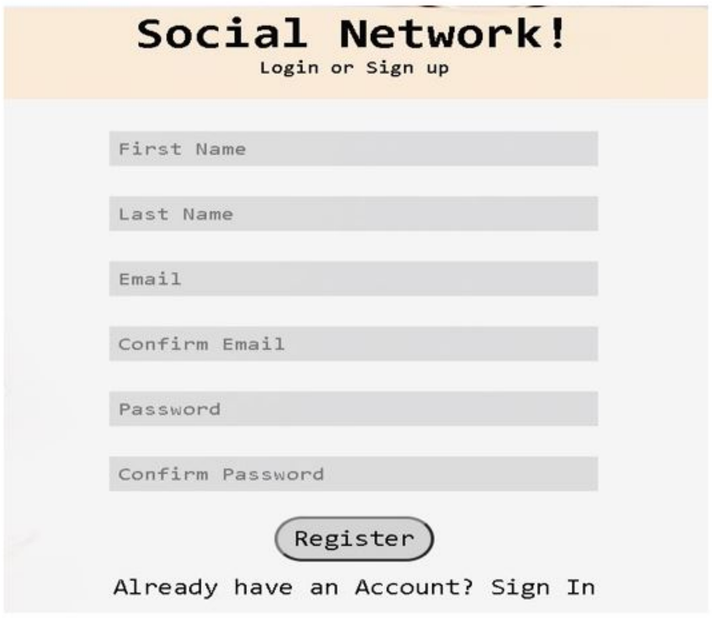


Fig: 7| Registration page



Social Network!
Login or Sign up

In fig: 8 we can see the working flowchart of the proposed model.

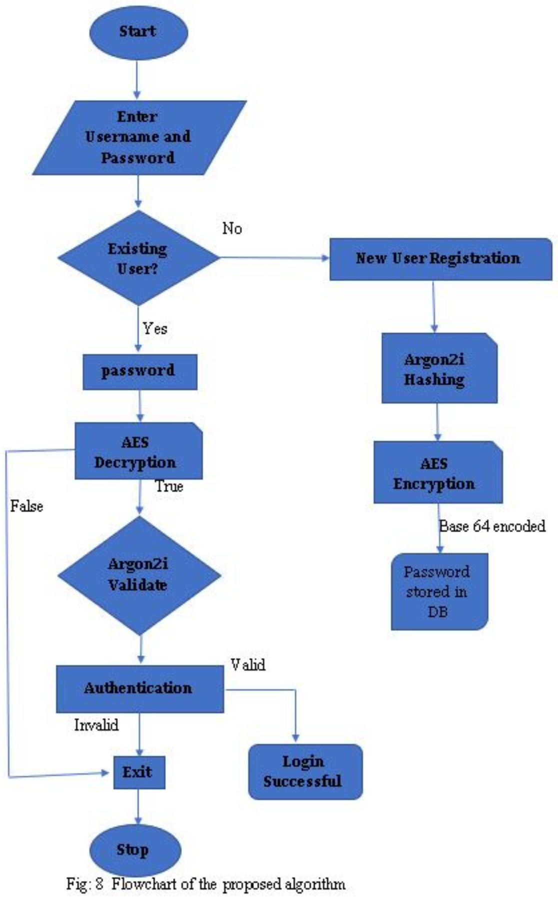
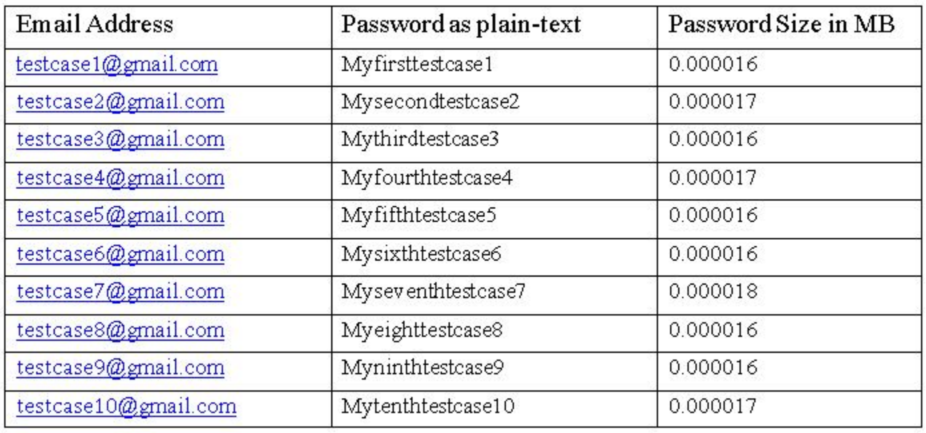


Fig: 8 Flowchart of the proposed algorithm

6 Evaluation

This section supplies the information on the performance analysis of the proposed model. To evaluate the performance of this model, we are considering the performance metrics like Encryption time, Throughput and Decryption time for 10 users registered with our web application. We have listed the usernames, passwords and passwords in MB for the test cases 1 to 10 in Table 1 below.
Firstly, Encryption time is defined as the time taken by the Argon2i hashing and AES encryption scheme together for converting plain-text password into cipher-text (Kumar and Chaudhary; 2018) and throughput is calculated by dividing the size of plain-text password in MB with time taken for encryption in seconds. Similarly, Decryption time is time taken by the AES decryption scheme for converting the cipher-text password into plain-text password.
To evaluate the proposed model, we have implemented it in two different operating systems and recorded the Encryption time, Decryption time, and Throughput for the test data created as shown in the Table1.

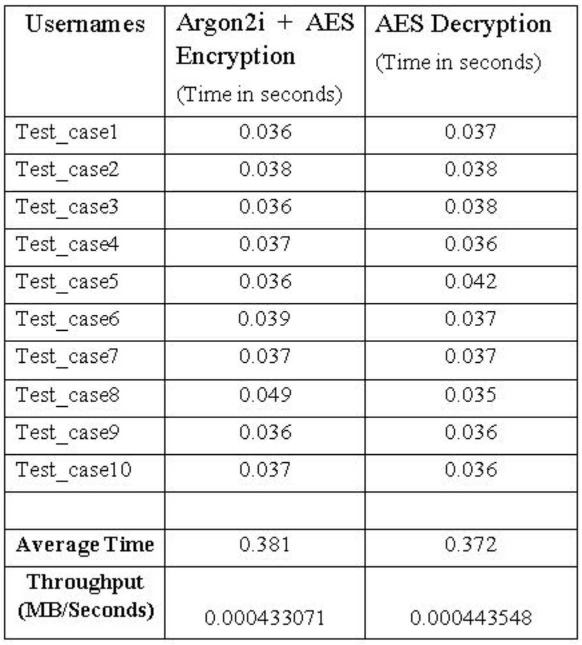


Email Address	Password as plain-text	Password Size in MB
testcase1@gmail.com	Myfirsttestcase1	0.000016
testcase2@gmail.com	Mysecondtestcase2	0.000017
testcase3@gmail.com	Mythirdtestcase3	0.000016
testcase4@gmail.com	Myfourthtestcase4	0.000017
testcase5@gmail.com	Myfifthtestcase5	0.000016
testcase6@gmail.com	Mysixthtestcase6	0.000016
testcase7@gmail.com	Myseventhtestcase7	0.000018
testcase8@gmail.com	Myeighttestcase8	0.000016
testcase9@gmail.com	Myninthtestcase9	0.000016
testcase10@gmail.com	Mytenthtestcase10	0.000017

Table1: Test Data considered for evaluating this model

6.1 Case Study 1

In our case study 1, we have implemented the proposed model on 2.70 GHZ (2cores) Intel i7 processor, with installed memory of 16GB RAM, AMD Radeon TM R7 M445 Graphics with 4GB GDDR5 Graphics Memory, 64-bit Operating system (Windows 10 Home). To evaluate our proposed model, we have created test cases from 1 to 10 as described in Table: 1 with same email addresses and passwords. In Table: 2 we have noted the results of encryption time, decryption time recorded using current time method and can be viewed by error log from logs in Xampp control panel. Throughput was calculated by dividing the password size in MB with time taken in seconds.

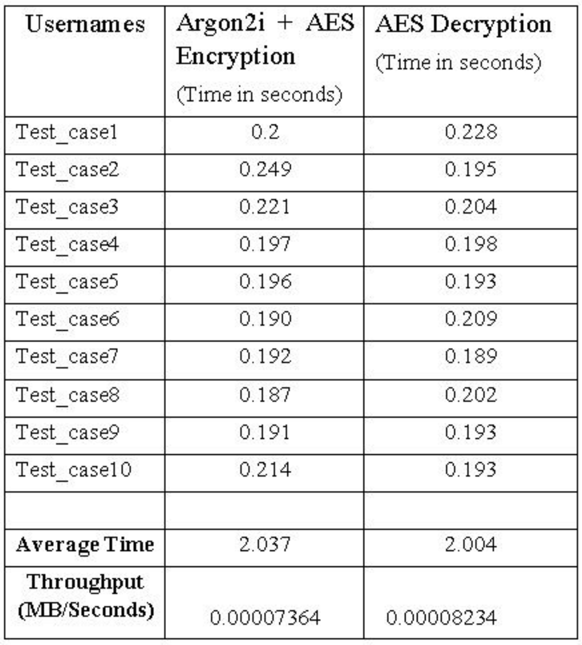


Usernames	Argon2i + AES Encryption (Time in seconds)	AES Decryption (Time in seconds)
Test_case1	0.036	0.037
Test_case2	0.038	0.038
Test_case3	0.036	0.038
Test_case4	0.037	0.036
Test_case5	0.036	0.042
Test_case6	0.039	0.037
Test_case7	0.037	0.037
Test_case8	0.049	0.035
Test_case9	0.036	0.036
Test_case10	0.037	0.036
Average Time	0.381	0.372
Throughput (MB/Seconds)	0.000433071	0.000443548

Table 2: Analysis of Encryption time and decryption time.

6.2 Case Study 2

In our case study 2, we have implemented our proposed model on 1.70GHz Intel core i5-3317U with installed memory of 8.00 GB, windows 10 home operating system. Firstly , we have created test cases from 1 to 10 with the same email addresses and passwords mentioned in the Table:1. By using the Apache error log from logs in xampp control panel, we have recorded the encryption time and decryption for the test cases 1 to 10 and tabulated the results in Table: 3. The encryption and decryption time listed is converted from milli seconds to seconds before noting down in Table:3.

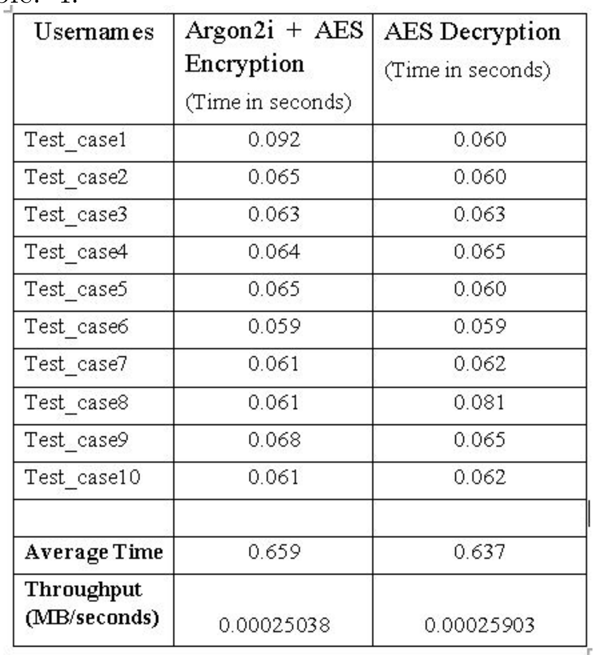


Usernames	Argon2i + AES Encryption (Time in seconds)	AES Decryption (Time in seconds)
Test_case1	0.2	0.228
Test_case2	0.249	0.195
Test_case3	0.221	0.204
Test_case4	0.197	0.198
Test_case5	0.196	0.193
Test_case6	0.190	0.209
Test_case7	0.192	0.189
Test_case8	0.187	0.202
Test_case9	0.191	0.193
Test_case10	0.214	0.193
Average Time	2.037	2.004
Throughput (MB/Seconds)	0.00007364	0.00008234

Table: 3 Analysis of Encryption time and Decryption time

6.3 Case Study 3

In our case study 3, we have implemented our proposed model on 1.80GHz Intel core i5-3337U CPU with installed memory of 4GB, 64-bit operating system. Similar to the above case studies, we have first registered test cases from 1 to 10 with same usernames and passwords as mentioned in Table: 1. To record the encryption time and decryption time, we have used the Apache log from logs in the Xampp control panel. . The encryption and decryption time listed is converted from milli seconds to seconds and later listed down in Table: 4.

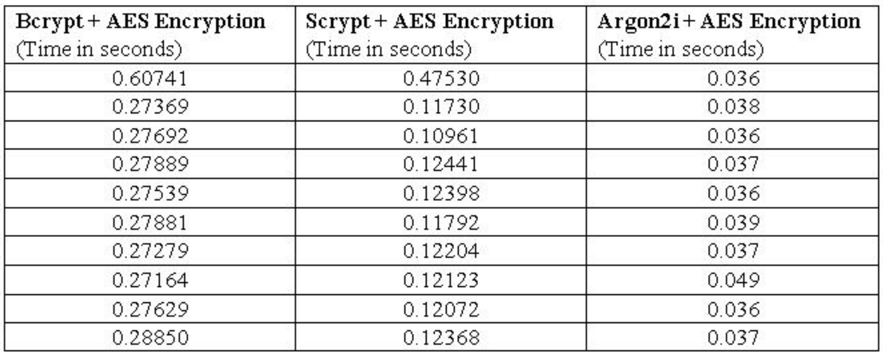


Usernames	Argon2i + AES Encryption (Time in seconds)	AES Decryption (Time in seconds)
Test_case1	0.092	0.060
Test_case2	0.065	0.060
Test_case3	0.063	0.063
Test_case4	0.064	0.065
Test_case5	0.065	0.060
Test_case6	0.059	0.059
Test_case7	0.061	0.062
Test_case8	0.061	0.081
Test_case9	0.068	0.065
Test_case10	0.061	0.062
Average Time	0.659	0.637
Throughput (MB/seconds)	0.00025038	0.00025903

Table: 4 Analysis of Encryption time and Decryption time

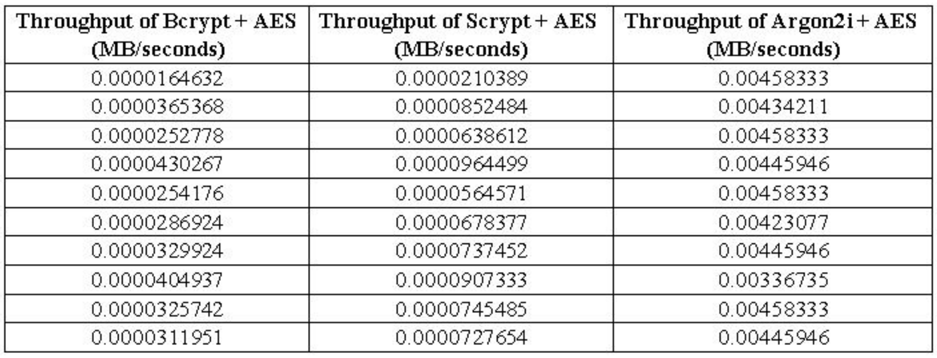
6.4 Discussion

The proposed model is implemented in three machines as shown in above case studies. Bcrypt (Kumar and Chaudhary; 2018) and Scrypt (Bidhuri; 2019) are proven to be the best hashing algorithms for forming hybrid models based on our earlier studies. Hybrid approaches like Bcrypt hashing with AES scheme by (Kumar and Chaudhary; 2018), Scrypt hashing with AES scheme by (Bidhuri; 2019), have made contributions for enhancing password security against brute force attacks. In our proposed model, we have implemented an award-winning hashing algorithm, Argon2i with AES scheme for enhancing password security and results were noted down in case studies 1 to 3. We have compared the results of encryption time and throughput of our proposed model with existing models like Bcrypt with AES (Kumar and Chaudhary; 2018), Scrypt with AES (Bidhuri; 2019) and our proposed model have recorded less encryption time and high throughput as shown in Table: 5 and Table:6. We have added extra protection layer for password security by generating keyed hash value using SHA-256 with inputs namely, password hashed using Argon2i hashing (hash key), cipher data combined with initialization vector (cipher data.iv) and base64 encoded to eliminate special characters before storing it in database. Therefore, our proposed hybrid approach, Argon2i hashing with AES scheme is proven for obtaining high performance and more secure than existing models towards password cracking attacks.



Bcrypt + AES Encryption (Time in seconds)	**Scrypt + AES Encryption** (Time in seconds)	**Argon2i + AES Encryption** (Time in seconds)
0.60741	0.47530	0.036
0.27369	0.11730	0.038
0.27692	0.10961	0.036
0.27889	0.12441	0.037
0.27539	0.12398	0.036
0.27881	0.11792	0.039
0.27279	0.12204	0.037
0.27164	0.12123	0.049
0.27629	0.12072	0.036
0.28850	0.12368	0.037

Table: 5 Analysis of Encryption time in seconds



Throughput of Bcrypt + AES (MB/seconds)	**Throughput of Scrypt + AES (MB/seconds)**	**Throughput of Argon2i + AES (MB/seconds)**
0.0000164632	0.0000210389	0.00458333
0.0000365368	0.0000852484	0.00434211
0.0000252778	0.0000638612	0.00458333
0.0000430267	0.0000964499	0.00445946
0.0000254176	0.0000564571	0.00458333
0.0000286924	0.0000678377	0.00423077
0.0000329924	0.0000737452	0.00445946
0.0000404937	0.0000907333	0.00336735
0.0000325742	0.0000745485	0.00458333
0.0000311951	0.0000727654	0.00445946

Table: 6 Analysis of Throughput in MB/seconds

7 Conclusion and Future Work

The major goal of our research project is to check the performance and password security of an online social networking user by implementing a hybrid combination of Argon2i hashing and AES scheme against password cracking attacks. (Biryukov et al.; 2015)Argon2i uses data-independent memory access and is highly recommended for password hashing and password based key derivation which is used in combination with memory accelerated encryption scheme AES. The intricacy of the password was enhanced by passing it through Argon2i hashing and AES encryption. we have derived two keys namely, cryptokey and hashkey using PBKDF2 Key derivation function. To increase the complexity of cryptokey, we have hashed it again with SHA256 and used as key in AES encryption phase. Also we have generated a keyed hash value inside HMAC method using SHA256 with parameters namely, the cipherdata, Initialization vector, and hashkey. The resultant keyed hash value along with cipher data are base64 encoded to drop special characters before storing the password in database. This research concludes that hybrid combination of Argon2i and AES scheme along with other extra protection layers can successfully increase the password security against password cracking attacks.
In the future works, Firstly, we want to design and implement Internationalization (i18n) in our web application for supporting local languages, cultural settings, and functioning more appropriately based on local norms. Secondly, we can also make major changes to the existing configuration so that web application can be deployed into cloud based on the cloud environment required.

References

AbdElnapi, N. M., Omara, F. A. and Omran, N. F. (2016). A hybrid hashing security algorithm for data storage on cloud computing, *International Journal of Computer Science and Information Security (IJCSIS)* **14**(4).

Bellovin, S. M. and Merritt, M. (1993). Augmented encrypted key exchange: a password-based protocol secure against dictionary attacks and password file compromise, *Proceedings of the 1st ACM Conference on Computer and Communications Security*, pp. 244–250.

Bidhuri, V. (2019). *Enhancing Password Security Using a Hybrid Approach of SCrypt Hashing and AES Encryption*, PhD thesis, Dublin, National College of Ireland.

Biryukov, A., Dinu, D. and Khovratovich, D. (2015). Argon and argon2: password hashing scheme, *Technical report, Tech. Rep.* .

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.

Boonkrong, S. and Somboonpattanakit, C. (2016). Dynamic salt generation and placement for secure password storing, *IAENG International Journal of Computer Science* **43**(1): 27–36.

Chhetri, B. (2020). Novel approach towards authentication using multi level password system, *International Journal of Computer Applications & Information Technology* **12**(1): 292–297.

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.

Garver, M. (n.d.). Owasp top 10 application security threats.

Katrandzhiev, N., Hristozov, D. and Milenkov, B. (2019). A comparison of password protection methods for web-based platforms implemented with php and mysql., *International Journal on Information Technologies & Security* **11**(2).

Kaur, G. and Malhotra, S. (2013). A hybrid approach for data hiding using cryptography schemes, *International Journal of Computer Trends and Technology (IJCTT)* **4**(8).

Khowfa, W. and Silasai, O. (2019). The efficiency of using salt against password attacking, *JOURNAL OF SOUTHERN TECHNOLOGY* **12**(1): 217–227.

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

Liu, Y., Zhang, W., Peng, X., Liu, Y., Zheng, S., Wei, T. and Wang, L. (2019). Design of password encryption model based on aes algorithm, *2019 IEEE 1st International Conference on Civil Aviation Safety and Information Technology (ICCASIT)*, IEEE, pp. 385–389.

Ntantogian, C., Malliaros, S. and Xenakis, C. (2019). Evaluation of password hashing schemes in open source web platforms, *Computers & Security* **84**: 206–224.

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

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).

Sood, S. K., Sarje, A. K. and Singh, K. (2009). Cryptanalysis of password authentication schemes: Current status and key issues, *2009 Proceeding of International Conference on Methods and Models in Computer Science (ICM2CS)*, IEEE, pp. 1–7.

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.

Sumagita, M., Riadi, I., Sh, J. P. D. S. and Warungboto, U. (2018). Analysis of secure hash algorithm (sha) 512 for encryption process on web based application, *International Journal of Cyber-Security and Digital Forensics (IJCSDF)* **7**(4): 373–381.

Wetzels, J. (2016). Open sesame: the password hashing competition and argon2, *arXiv preprint arXiv:1602.03097* .

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).

Yu, F. and Huang, Y. (2015). An overview of study of passowrd cracking, *2015 International Conference on Computer Science and Mechanical Automation (CSMA)*, IEEE, pp. 25–29.