Modular approach with the blend of Argon2 Hashing and Twofish Encryption for strengthening Password Security

MSc Research Project
Cyber Security

Virendra Yadav
Student ID: x21154384

School of Computing
National College of Ireland

Supervisor: Michael Pantridge

National College of Ireland

MSc Project Submission Sheet

School of Computing

Student Name:	Virendra Yadav		
Student ID:	X21154384		
Programme:	Cyber Security	**Year:**	2022
Module:	MSc Research Project		
Supervisor:	Michael Pantridge		
Submission Due Date:	15th December 2022		
Project Title:	Modular approach with the blend of Argon2 Hashing and Twofish Encryption for strengthening Password Security		
Word Count:	7875	**Page Count**	22

Signature:	Virendra Yadav
Date:	15th December 2022

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):	

Modular approach with the blend of Argon2 Hashing and Twofish Encryption for strengthening Password Security

Virendra Yadav
x21154384

Abstract

In this evolving world we must accept the reality that the central databases which are equipped to store our passwords and other sensitive information will eventually experience a data breach in this age of the internet. Cyber risks are growing every day, and we can see that many businesses are experiencing data breaches that compromise the information of unwitting customers. The confidentiality of passwords is still at risk even though numerous cryptographic techniques, including hashing and encryption, have been demonstrated to be secure. The sensitive data of innocent consumers are still susceptible to several advanced-level assaults, including brute force attacks, reverse engineering attacks, GPU attacks, and bespoke hardware attacks. In this research paper, we concentrate on the security of online users using a hybrid combination of accelerated Twofish encryption algorithm and the hard hashing function Argon2, which can be equipped to increase the security of online users' passwords against different assaults.

1 Introduction

1.1 Background and motivation

Providing security at every level of data transmission as well as data storage is one of the many issues that the development of computer networking brings. The online world includes doors leading to the things a user needs, but in order to access these things with their identity, they must first enter their login information and then enter their password as the key to unlock the lock. One of the most delicate issues that must be addressed when one is thinking about security is the disclosure of users' sensitive passwords. Since the passwords are key to security doors, compromising these keys would eventually end up in a serious security threat as the attackers can pose as users and can exploit the users' resources this whole thing leads to a serious security issue to address high-level password security of users' passwords. One of the weak points of security is the storage of sensitive information (such as passwords, and credit card details) in plain text, the transfer of sensitive credentials in plain text, or unprotected encryption. Nowadays, a few protocols utilized for the transmission of data on the internet network are HTTP and HTTPS. Due to the fact that all data communicated through HTTP is sent in plain text only, data encryption is not supported by HTTP. The attacker would be able to read the communications in explicit plain-text format if he managed to hijack the transfer between the browser and the main server. The HTTPS protocols, which include the SSL protocol, were designed to address this kind of problem.

When transmitting data through HTTPS, servers are verified using their corresponding certificates, and data is encrypted as it travels across the network. The HTTPS protocols are not completely secure, though, since an attacker can still carry out Man-in-the-Middle Attacks that can jeopardise the data transfer. In order to overcome this problem and adhere to security standards, systems should conduct application-level high-level encryption processing using safe & highly secure encryption techniques (Singh and Kaur, 2015). Additionally, encrypted data must be kept in databases since doing so will ensure that, even if a database breach occurs, sensitive information about users won't be revealed in plaintext and will remain unreadable to the attackers. (Chandra et al., 2014) says that Symmetric encryption and asymmetric encryption are the two main categories into which data encryption techniques fall. The key type is used as the foundation for this categorization. The suggested Twofish algorithm, which is used in this study, is a new generation of encryption technology that is more sophisticated and secure than DES and roughly similar to AES.

As a symmetric encryption algorithm, Twofish encrypts and decrypts data using the same key. It accepts both the plaintext data and the key as the input. This key then encodes the data into ciphertext, rendering it incomprehensible without proper decoding. The recipient receives the encrypted data and the encryption key either after or concurrently with the ciphertext. The user can decode the encrypted data using this key. The extensive use of pre-computed, key-dependent replacement boxes sets Twofish apart & unique from other encryption algorithms as one of its major properties. The connection between the key and the ciphertext is hidden by the S-box. Additionally, the S-box is already available but requires the encryption key to decode the data. One of the safest encryption techniques is Twofish, which has a 128-bit block size and along with that it has variable-length encryption key which gives it an extra layer of protection.

According to the research done by (Singh and Kinger, 2013), the AES algorithm outperformed the DES, RSA, and 3DES algorithms in terms of execution speed and throughput value. The research also covered the possibility of using hybrid encryption methods to further boost overall security to the next level. In contrast to the AES encryption algorithm, the focus of our proposed research paper is on the Twofish encryption algorithm because, according to the research (Rizvi et al., 2011) it was noted that Twofish encryption is faster than AES encryption when RAM is significantly increased. Additionally, it was discovered that both algorithms were meeting the same safety requirements because there was no discernible difference between these two encryption methods.

In the publication (Biryukov et al., 2016) where the memory hard function of the Argon2 hashing methodology was employed for hashing the password, the researcher used Argon2, which emerged as an award-winning hashing approach. (Wetzels, n.d.) demonstrated a comparison between the SCrypt and BCrypt hashing algorithms with Argon2 hashing. However, it was determined that Argon2 hashing takes less time to complete and that its hardware requirements are comparable to or even lower than those of its rivals. When one equips Argon2 hashing for password hashing, attacks such as TMTO and side-channel attacks can be protected against it. The researcher (Tyagi et al., 2021) offered the idea of combining two methods in order to boost password security to a higher degree. It covered the usage of various hashing methods to combine user-provided passwords with cryptographic hash values, and how the resulting cipher-text would then be fed as input to the AES

Encryption process. When compared to existing models of password security, it was found that the suggested approach had a superior level of security. The researcher combined AES Encryption along with SCrypt Hashing in order to increase password security which can be further used to determine whether the combined algorithm would increase the overall security or not. The researcher came to the conclusion that using AES encryption along with SCrypt hashing would be good enough rather than using AES Encryption along with BCrypt hashing because the combination increases password complexity to a higher level.

Scientist "Horst Feistel" introduced the idea of the avalanche effect and covered it in his journal ("Cryptography and Computer Privacy," n.d.). He proposed the ideas of confusion and dispersion, both of which are useful for fortifying algorithms. Through a series of experiments, he asserted that the corruption of a single-digit position during the encryption stage can tamper with the data to a very significant level. To calculate this level, he coined the term "Avalanche Effect," which states that even small changes to a bit of data should result in significant changes to the cipher-text. The higher the Avalanche value, the more complex the data security which decreases the probability of deciphering secured data. Horst Feistel asserts that the avalanche effect is a desirable property of any particular cryptographic algorithm. However, it is frequently the case that algorithms with a sizable avalanche effect are more secure than those without one.

The combination of Argon2i and AES Encryption was tested by a different independent researcher, and the findings revealed that it was more effective at protecting passwords than the previous SCrypt hashing and AES Encryption methods. Argon2 was made available after improvements in SCrypt hashing and addressing the shortcomings of this method, and it was more secure. The two finest hashing and encryption algorithms have been combined in this research to produce the best results for the password security of online users, following a detailed analysis of the problems and shortcomings of each approach. (Rizvi et al., 2011) concludes that Twofish becomes speedier than AES with more RAM and the security of Twofish is more as compared with the AES encryption technique.

By using the most modern and efficient hashing method, Argon2, together with the Twofish encryption method, which is currently the best encryption algorithm provided the machine has adequate RAM, the suggested research article solves the drawbacks of past systems.

1.2 Motivation

Cybercriminals can get unauthorized access to the system through authentication flaws caused by weak password security. Passwords are still susceptible to several assaults even having strong security methods such as strong encryption algorithms and advanced hashing techniques for password protection. The major goal of this research is to combine two of the most effective techniques to make passwords much more complicated and more challenging for attackers to crack for this we have used Argon2 hashing which is an award-winning as best hashing technique to date and along with that we have equipped Twofish encryption technique which is considered best in terms of security (Rizvi et al., 2011). This robust hybrid strategy can significantly raise the level of password security for online users.

1.3 Research Question

Can we utilize a hybrid mix of Argon2 hashing and the Twofish encryption algorithm to increase the password security of an online user from brute force attacks?

1.4 Structure of Research Paper

This research paper has the following organizational structure: Section 1. The introduction will establish the need for research, various topics of the proposed project, the motivation behind the research and the research question which will be addressed in this paper. Section 2 Related Work elucidates the literature review in a detailed manner along with a detailed analytical overview of numerous methodologies and approaches reviewed in the section. Section 2.1 This section covers the various password-cracking attacks which have happened in previous times, a variety of password cracking, and several strategies & methods which are used by the attackers in order to crack the encrypted passwords. Section 2.2 This section covers the system which equips various hashing algorithms for password protection, and also covers the previous studies on numerous hashing techniques and why Argon2 hashing has been chosen for this research paper. Subsection 2.3 password security using encryption methods; this section covers all the related work which has been done on various encryption techniques over the years and features of numerous encryption methods and also why the Twofish encryption algorithm has been chosen for this project. Section 2.4 Previous hybrid techniques proposed by researchers and the need for hybrid models and why the hybrid model of Twofish encryption and Argon2 hashing would be appropriate to meet the security enhancement of passwords. Next Section 3: The methodology equipped in the proposed model is extensively explained in this section along with a few essential diagrams. It also covers the various stages of hashing and encryption-decryption process. Section 4 Design Specification elaborates on the overall design & architecture of the whole proposed model along with detailed design specifications. Section 5 Implementation section covers the final implementation of the proposed model along with a flow chart for easy understanding. Section 6 Evaluation which has been made over the proposed model and the performance of the proposed system over the various parameters is covered in this section. Section 7 Conclusion of the research and Future works related to it.

2 Related Work

This section of our study proposal illustrates the core theories and helpful notions put out by different researchers and academics employing a variety of unusual methods and procedures. The literature study linked to the Argon2 hashing algorithm, Twofish algorithm, and their hybrid technique that can be equipped to safeguard passwords is provided in the following subsections.

2.1 Assaults on Password Cracking

In the modern online world, attempts that attempt to crack encrypted passwords are becoming increasingly prevalent. Attackers many times get access to the databases that hold the credentials and can download the passwords for their own purposes. The attackers then try to figure out various techniques which can be used to crack the passwords. (Liu et al., 2019) discuss various techniques which are used by hackers for cracking passwords. Researchers then suggested ways for more effectively counting guesses as well as adaptable Markov Models trained on passwords.

In order to guess passwords, the researcher (Melicher et al., n.d.) presented a recurrent neural network made of long short-term memory (LSTM) units in 2016. Similar to Markov models, this method uses a multi-layer special neural network to determine the likelihood of each character in a possible password depending on the characters that came before it. (Liu et al., 2019) with the help of the tools, researcher did the first computationally effective examination of password security against the several kinds of mangled-wordlist assaults that are actually used in the real world. Additionally, they demonstrated how our methods enable four data-driven refinements to enhance the ordering and comprehensiveness of rule lists and wordlists, better matching our models with the covert (but potent) configurations used by skilled attackers.

Password recovery from a cipher-text or hash value by an attacker is known as a password cracking or guessing assault by researchers (Yu and Huang, 2015). Password cracking attempts come in two flavours: online and the other is offline. In online assaults, a hacker can develop a script file or else software with automation that will be performed to attempt each password in the list provided and, when a match is found, will provide the attacker access. An online attacker can create a malicious script file or either a program with automation that, when run, will try each & every password on the given list and, when a match is discovered, will grant access. On the other side, if an attacker has the access to a database of users' hashed password values, then he can try to decrypt each password by contrasting the user's guess with the hash values. Offline assaults, which include brute force, dictionary, and rainbow table attacks, have been split into three categories by researchers (Ertaul and Kaur, n.d.). It is fully detailed how the hardware platforms make password cracking possible with the help of advanced Graphical processing units and numerous Field programmable gate arrays.

2.2 Password protection with the help of hashing methodology

This section describes prior research made on Argon2 and other hashing algorithms which were equipped to bolster password security. (Sriramya and Karthika, 2015) discusses that as compared to encryption, hashing is one of the greatest techniques for password protection since it is a one-way transformation process that prevents anyone from recovering the password's plain text. A hashed password can, however, be cracked using a pre-determined hash value or an active hash dictionary. Due to collision effects, it is conceivable that two separate plain texts could have the precisely same hash value. Therefore, a hacker can use attacks such as dictionary attacks, brute force, and rainbow attacks, as well as lookup tables can also be used against basic hashing techniques like MD5 and SHA256 /SHA1.

The same problem is addressed in research by (Ertaul and Kaur, n.d.), who points out that there is a wide variety of open-source software, including tools like John the Ripper 5 & Hashcat 6, that can defeat conventional hashing methods and retrieve the true password. By using salt, this issue may be avoided. According to (Marton et al., 2010), salt is a string of random bits that may be created safely using the pseudo-randomness technique and adding it to the plain-text password prior to the hashing process. Different hash values will result from the salt if in case the same hashing technique is applied many times to the same plain text. While (Ertaul and Kaur, n.d.) claimed that salt increased the hash's unpredictability and reduced the vulnerability of the password to rainbow table attacks and look table attacks, it did not provide any protection against dictionary and brute force assaults. These assaults can do millions of hashes per second thanks to high-performance specialized hardware, which makes them more potent. Since key stretching is its foundation, more sophisticated hashing methodologies like PBKDF2, BCrypt hashing, and SCrypt are utilized. This method slows down the procedure by adding a calculation to the key creation process to extend the time it takes to produce the hash value. An attacker will find it challenging to quickly crack the hash using this method. In comparison to PBKDF2, BCrypt & SCrypt are slow algorithms. Fast algorithms are easier to defeat than slower ones. In addition to salting, these sophisticated hashing algorithms also include an iteration count as an additional security measure.

Since user-defined passwords have poor entropy and weak randomness, (Turan et al., 2010) advise against using them directly as cryptographic keys for hashing algorithms. The passwords, however, are occasionally the sole choice available for use as secret data by cryptographic methods. As a result, a technique known as the Password-Based Key Derivation Function (PBKDF) can be utilized, in which the key is created from a secret value such as a password/ sensitive data. (Percival and Josefsson, 2016) had demonstrated, meantime, that the majority of key derivation algorithms were solely based on cryptographic security considerations such as the salting process & precise iteration count. These algorithms all have a similar vulnerability to strong attackers. Processors are getting smaller, quicker, and more efficient because of the rapid advancements in semiconductor technology. They can now process a lot more data in parallel for precisely the same price. Because of this excellent parallelism scope, attackers may still swiftly crack passwords via brute-force attacks or dictionary attacks increasing the overall repeat count. As a result, a more potent hashing algorithm such as SCrypt hashing aims to thwart attackers that use specifically designed parallel circuits.

According to (Ertaul and Kaur, n.d.), SCrypt is a hashing technique for password-based keys that produces a large stream of pseudo-random bit strings. It produces a message digest or fixed-length hash value using the input of plain text data. As a result, it offers the maximum level of password security and makes it nearly difficult for an attacker to guess the password using brute force methods. It has significant processing costs and needs a lot of memory for its pseudo-random bits. It is one of the costliest one yet secure hashing algorithms as a result. In addition to being utilized for several cryptocurrency applications and key derivations, SCrypt hashing also served as an inspiration for the design of Argon2. Two Argon2 variations exist Argon2d and Argon2i. Argon2d is faster and relies on data-memory access, making it ideal for cryptocurrency and other programs without side-channel scheduling issues. The main distinction between Argon2i and Argon2d is that Argon2i is required for password hashing as well as for password-based key generation and is totally data-independent. to solve the issues with currently used schemes like SCrypt, which are rigid in their ability to separate memory and time costs and are complicated & difficult to assess.

(Biryukov et al., 2021) suggests Argon2 as a result for applications requiring the maximum level of resilience but with prohibitive time and computational costs when memory reduction is implemented. According to (Biryukov et al., 2021), one reason Argon2 is nowadays majorly used in high-performance applications is that versions 2d and 2i can quickly fill up to 1 GB of RAM in a very less duration, scale effectively for achieving parallel processing of many units, and have an optimized design for simple analysis & implementation.

2.3 Protection of passwords through Encryption techniques

The prior research on various cryptographic methods, such as encryption algorithms, will be the main emphasis of this section. (Nadeem and Javed, 2005) investigated the use of many standard secret key algorithms, such as DES, AES and Blowfish, in the ECB (Electronic Codebook) and Cipher Feedback modes. Encrypting input files with different contents and sizes allowed us to compare how well they all performed overall. They came to the conclusion that the quickest encryption algorithm is Blowfish. Additionally, was

demonstrated that AES outperforms 3DES and DES. Additionally, it demonstrates that DES is three times quicker than 3DES. AES, RC6, 3DES, Blowfish technique, and RC2 were among the symmetric encryption techniques that (Elminaam et al., 2008) tested. The researchers came to the following conclusions: There is no such difference when the results are displayed in the base 64 mode or hexadecimal base encoding; Blowfish and RC6 perform better than other popular encryption techniques; RC2, RC6, and Blowfish become slower when the data type changes, such as when it becomes a picture; and a larger key size significantly increases battery and time usage.
The (Hirani, 2003) investigation demonstrates that AES is quicker and more effective than alternative encryption methods. In his experiment, Hirani demonstrated that cutting the number of rounds can save energy, but that doing so renders the algorithm vulnerable to cryptanalysis and should be avoided. Seven or more rounds can be regarded as quite secure and, in some circumstances, utilized to conserve energy.
(Rizvi et al., 2011) came up with the research where they found a relationship between factors like the speed of encryption for different types of plaintext with variable length, and RAM size, according to research that looked at Twofish and AES for text, image, and audio encryption on different RAM sizes at various operating systems. It was discovered via the investigation that both of them had merely an equivalent safety factor. It was discovered that AES is quicker than Twofish for the Text Encryption process, although Twofish becomes faster than AES with more RAM. The Twofish algorithm performs better at Sound Encryption, and with additional RAM, its performance rises even more. So RAM size affects more the performance of Twofish, and for that, we came up to carry out our research using Twofish encryption as it has more security factors and good performance when enough RAM is provided.

2.4 Protecting passwords by utilizing hybrid models

This section will cover several hybrid approaches in addition to having a strong emphasis on hashing and encryption methods. (Alvarez et al., 2018) has put out a concept that uses SHA3 256 as the secure hashing algorithm to produce the input for AES (128-bit key and Initialization Vector) and AES - 128 in CTR mode to optimize the performance of PBKDFs. The model's security aspects were examined by the researchers, who also evaluated how well it performed in comparison to other reliable hashing techniques like SCrypt hashing and Argon2 hashing. When the same amount of RAM was employed, it was discovered that this hybrid model outperformed Argon2.
(Kumar and Chaudhary, 2018) had suggested using a blend of BCrypt hashing along with AES encryption to protect online accounts from various assaults like brute force. This technique concentrated on maintaining password security even when the database is corrupted. The outcome, which supported this hypothesis, was examined in light of factors like throughput and encryption time. A workable model put out by (Kumar and Chaudhary, 2018) calls for the user in order to access the login portal and enter the user's critical credentials, such as usernames & passwords, which are kept in a list in a database. The password is then hashed using the well-known BCrypt hashing techniques. When executing encryption and decryption procedures, the hash value acquired is utilized as a key for the AES technique, and its performance is measured using matrices such as encryption time, decryption time and throughput time. For the evaluation purpose, the data set was examined for the performance assessment in terms of encryption time and throughput for both Windows and MAC operating systems, as well as performance comparisons with the currently used BCrypt with AES encryption.

(Modugula, n.d.) came up with the advanced combination of argon2 hashing along with AES encryption methodology, which he used for password security. An online portal was made by the researcher and the inputs were taken from the users those sensitive data were first passed through the award-winning Argon2 hashing process and the hashed value was passed to the AES encryption phase following this the decryption phase occurs and subsequently results in hash matching and successful login. (Modugula, n.d.) made the evaluation on execution time and the throughput of the system, along with this the researcher also shows how the length of a password can impact the complexity of passwords. The case studies were handled over various operating systems in order to confirm that the operating system does play a crucial role in any of the evaluation parameters.

Memory-intensive operations are resistant to several types of attacks, including brute force, rainbow, dictionary, and others, but are yet susceptible to GPU or specialized hardware assaults. Due to Twofish, encryption can be used along with Argon2 hashing technique with a hardware acceleration feature, which protects against bespoke hardware assaults, this flaw may be fixed. As a result of the above literature review, the goal of this research is to examine the performance of the suggested method and concentrate on a hybrid Argon2 hashing and Twofish encryption combination. This model's output improves password security for online users by protecting them against various assaults like brute force and rainbow attacks.

3 Research Methodology

The methods utilized in the recommended model to enhance the online protection of social network users' credentials against password-cracking attempts are analyzed in this section. The fundamental concept is to utilize the password as the primary input to create the hash key, carry out the hashing, feed the key and the resulting hash value into the encryption and decryption processes, and then compare the hash value once again. Argon2 hashing utilizing the key derivation function and Twofish encryption process are combined in the suggested model in order to create a hybrid model for boosting the security of passwords against various password-cracking assaults. The passwords are taken from the user and are fetched for Argon2 hashing where the keys are derived and the hashing process executes. In order to add another layer of protection, we have passed the hashed value as input to Twofish Encryption where the encryption of the hash value takes place and we get the resultant ciphertext. For the retrieval, we pass the resultant ciphertext with the decryption phase and after that, the hash value is matched with the previous hash value. In this way, the hashing, encryption, as well as decryption phase, work out for our model.

The Argon2 hashing and Twofish encryption algorithms are clearly the best options for this hybrid architecture, according to the analysis of the literature research, due to their characteristics of memory-hard operating and memory acceleration efficiency. Key Derivation phase for Argon2 hashing, Twofish Encryption, and Twofish Decryption are the three sections that make up this model's structure.

3.1 Argon2 Hashing

The key is created during this key generation step with the use of the Argon2i password-based key generation mechanism. The Argon2 hash function is the progression of the BCrypt as well as the SCrypt scheme as they work in a similar method but with slight modifications along with that, it offers protection against brute force assaults utilizing a variety of parameters. The user's password is requested, as seen in figure (1) and is subsequently feed as input to the Argon2 hash algorithm. The hashing takes place and the generated hash value will be regarded as a 256-bit hash value. This hashed value would be feed as input for encryption and for the decryption operations the hashed value is taken for matching with cipher-text. Due to the user password's unpredictability and higher hashed value, it becomes challenging for an attacker to quickly crack the user's password using a brute force assault.

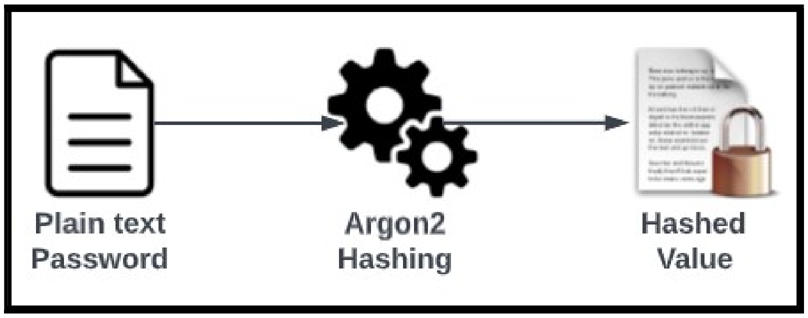


Figure (1): Argon2 Hashing Process

3.2 Twofish Encryption Phase

The input data for Twofish Encryption, a form of block-cipher encryption, is separated into 4 sub-sections, each of which is made up of 32 bits, once the hashed value has been obtained using Argon2 hashing. Four crucial components known as "whitening" will be used to perform bit-XOR input. Due to the Feistel network's use of Twofish encryption, this approach will go through 16 rounds. Through this step the hashed value was feed to encryption, through Twofish encryption we finally get the resultant cipher-text. As the primary password value was passed through the hashing and encryption process, the cipher-text would grow increasingly complicated and more difficult for an attacker to decipher. The below figure (2) is depicting the Twofish Encryption phase.

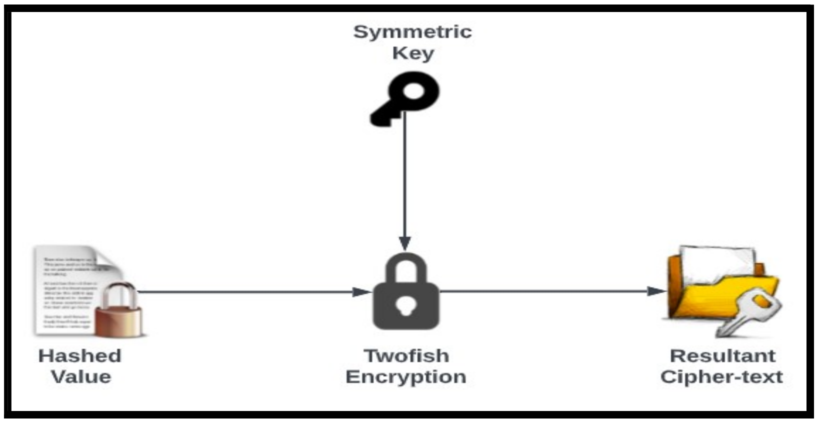


Figure (2): Twofish Encryption Phase

3.3 Twofish Decryption & Hash Matching Phase

The generated cipher-text is initially sent into the Twofish decryption step, where the Twofish algorithm converts the encrypted cipher-text into a hashed value. Secondly, this hashed value is checked against the previously saved hashed value that we have stored at the time of Argon2 hashing, and if the hashed values match, a message indicating a successful hash match is issued. The figure (3) shows the Twofish Decryption and hash matching phase.

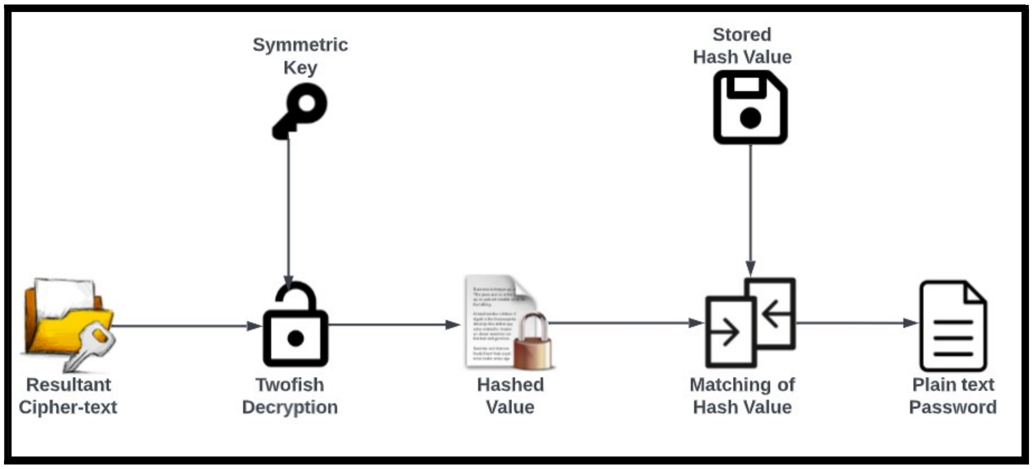


Figure (3): Twofish Decryption & Hash Matching Phase

4 Design Specification

The design specification covers the architecture of the main proposed model and along with it the flow of the model is also demonstrated using the algorithm of the hybrid model. The architecture of this approach, as seen in figure (4), is made up of three major components: Key Derivation (using Argon2 hashing), Twofish Encryption, and Twofish Decryption phase.

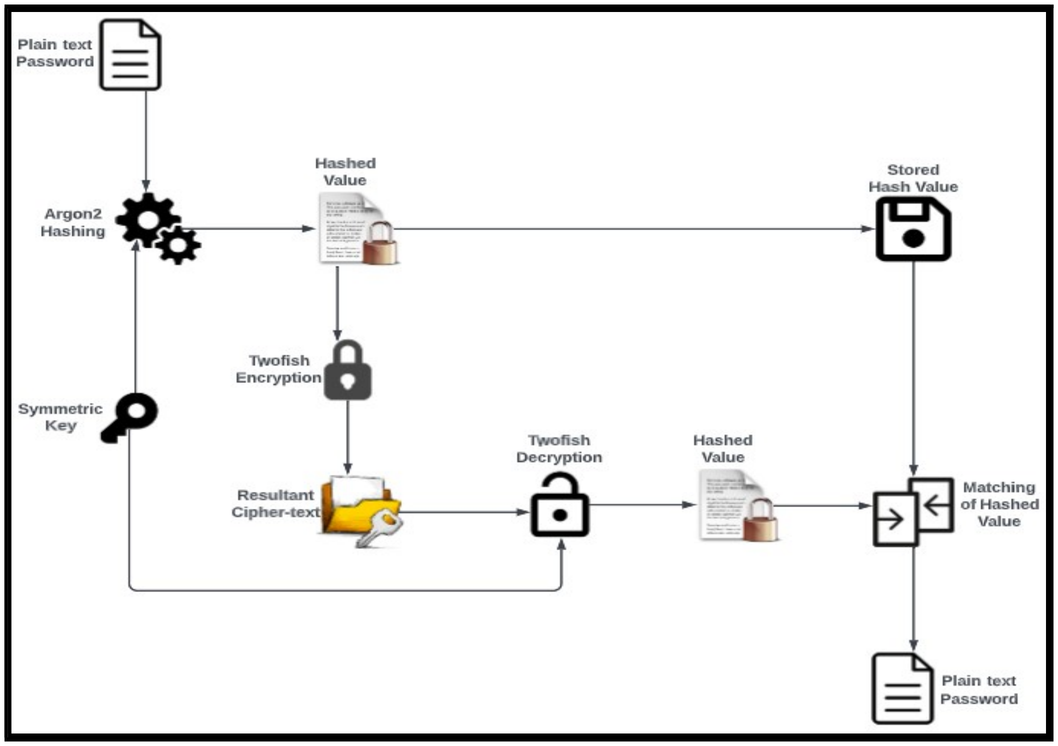


Figure (4): Design Architecture of the Proposed Hybrid Model

4.1 Algorithm for the proposed hybrid Model

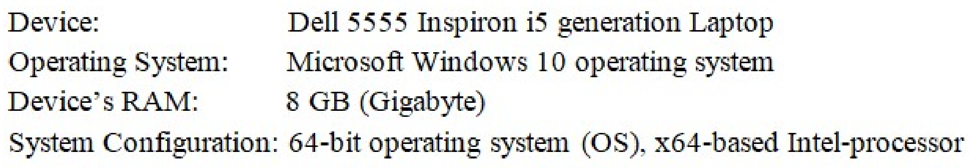
Step 1: Start
Step 2: Input the password taken by the user in plain text format.
Step 3: Plain text data passed for Argon2 hashing

Step 4: Hashing occurs through Argon2 hashing and the hashed value is stored in the database for further check.
Step 5: Hashed value is feed to Twofish encryption.
Step 6: Through Twofish encryption, the key is generated which is further used for encryption.
Step 7: Feed the produced key and hashed value as input to the Twofish encryption scheme.
Step 8: Resultant cipher-text is derived from step 6, which is stored.
Step 9: For the decryption process, the resultant cipher-text is deciphered using the Twofish encryption process.
Step 10: Deciphering takes place using the key and the hashed value is given.
Step 11: Compare the resulting hashed value with the previously stored hash value.
Step 12: Proceed with the hash value matching, "Hash matched" appears after a successful match.
Step 13: Stop

5 Implementation

The proposed hybrid model's implementation is discussed in this section. We have equipped Python 3.0 as our programming language for the implementation portion. Visual Studio Code (VSC) has been extensively used as our primary code editor.

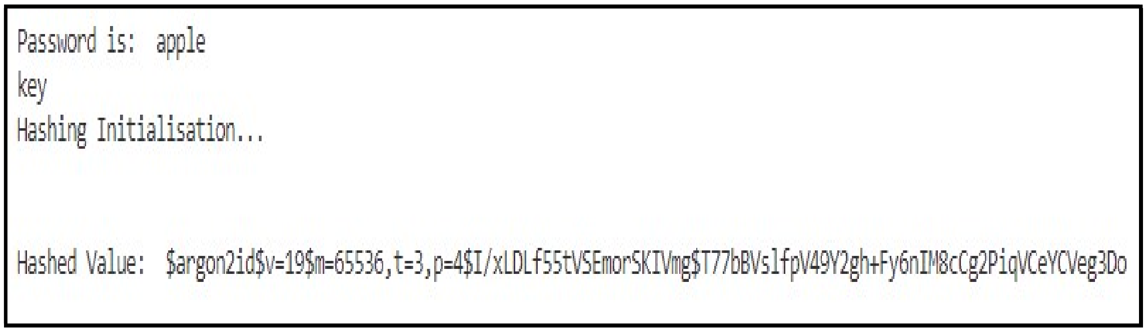
The Device configuration which was used for this project is mentioned below:



Device:	Dell 5555 Inspiron i5 generation Laptop
Operating System:	Microsoft Windows 10 operating system
Device's RAM:	8 GB (Gigabyte)
System Configuration:	64-bit operating system (OS), x64-based Intel-processor

The entire piece of code is broken out into many Python files rather than being packed into one. The main file which has been extensively used for calling all other files is main.py and other files are twoFish.py (Twofish encryption & decryption file), argon_hash.py (Argon2 hashing file), aes.py (The AES encryption file for comparison), testData.py (File for data passing), event.csv (For storing the final data values).
When an input is supplied, the passwords are first sent to the argon hash.py file, where Argon2 hashing is performed, producing a hashed value as output. Next, the hashed value is sent to the twoFish.py file, where encryption occurs and the resulting cipher-text is obtained. The generated cipher-text undergoes Twofish decryption for the decryption process, where a hash value is retrieved and then compared to previously stored hash values. Following this, if the match is successful, a message stating "hash matched" is sent. Event.csv file stores the encryption-decryption time and also the total execution time (which includes hashing time as well).

Argon2 Hashing: After taking the passwords the argon2 hashing is performed which give hashed value as output. The below figure (5) shows the process:

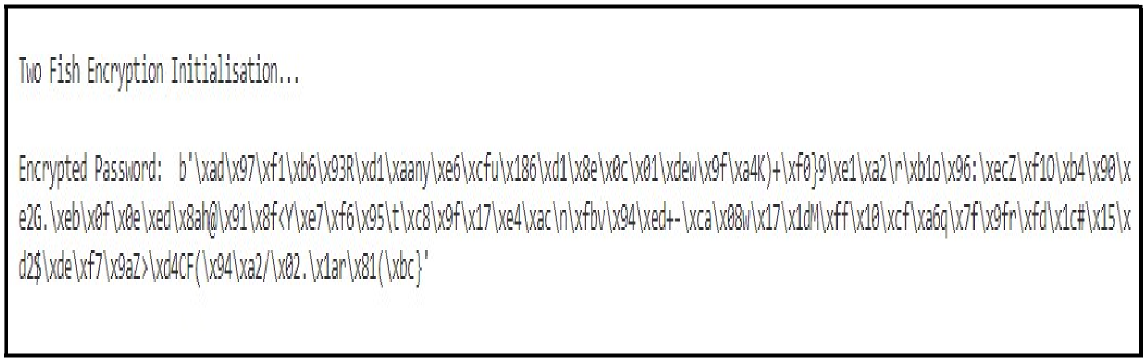


```
Password is:  apple
key
Hashing Initialisation...


Hashed Value:  $argon2id$v=19$m=65536,t=3,p=4$I/xLDLf55tVSEmorSKIVmg$T77bBVslfpV49Y2gh+Fy6nIM8cCg2PiqVCeYCVeg3Do
```

Figure (5): Argon2 Hashing process & output

Twofish Encryption: The hashed value is passed to the twoFish.py python file, upon which the Twofish encryption executes over the hashed value and the resultant cipher-text is produced as output. The figure (6) shows the Twofish encryption phase:

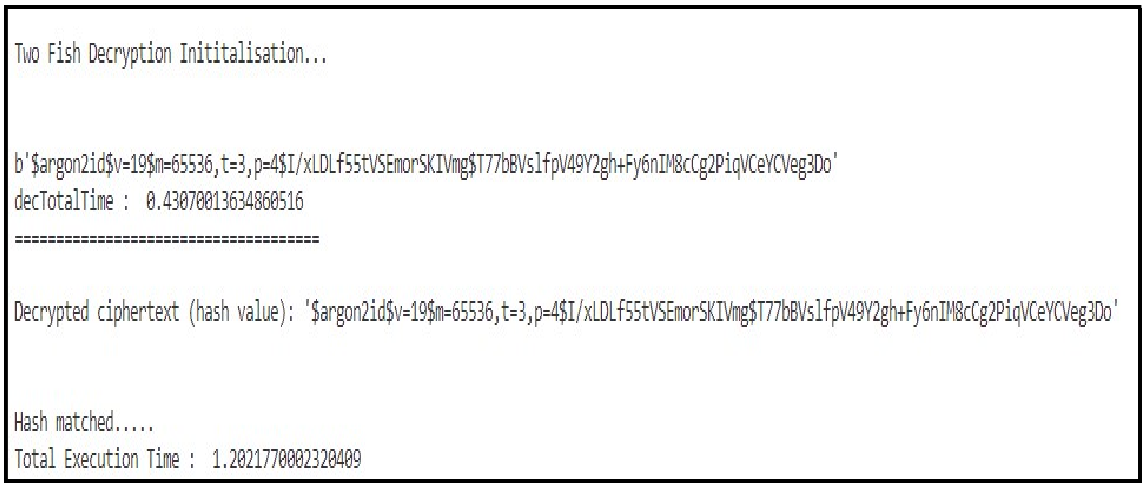


```
Two Fish Encryption Initialisation...


Encrypted Password:  b'\xad\x97\xf1\xb6\x93R\xd1\xaany\xe6\xcfu\x186\xd1\x8e\x0c\x01\xdew\x9f\xa4K)+\xf0]9\xe1\xa2\r\xb1o\x96:\xecZ\xf1O\xb4\x90\x
e2G.\xeb\x0f\x0e\xed\x8ah@\x91\x8f<Y\xe7\xf6\x95\t\xc8\x9f\x17\xe4\xac\n\xfbv\x94\xed+-\xca\x08w\x17\x1dM\xff\x10\xcf\xa6q\x7f\x9fr\xfd\x1c#\x15\x
d2$\xde\xf7\x9aZ>\xd4CF(\x94\xa2/\x02.\x1ar\x81(\xbc}'
```

Figure (6): Twofish Encryption process

Twofish Decryption & hash matching: The resultant cipher-text is taken as input for the Twofish decryption and the deciphering takes place which gives hashed value as output. This hashed value is further matched with the previously stored hash value, upon successful match "hash matched" message is delivered. The below figure (7) shows the decryption phase:



```
Two Fish Decryption Inititalisation...


b'$argon2id$v=19$m=65536,t=3,p=4$I/xLDLf55tVSEmorSKIVmg$T77bBVslfpV49Y2gh+Fy6nIM8cCg2PiqVCeYCVeg3Do'
decTotalTime :  0.43070013634860516
======================================

Decrypted ciphertext (hash value): '$argon2id$v=19$m=65536,t=3,p=4$I/xLDLf55tVSEmorSKIVmg$T77bBVslfpV49Y2gh+Fy6nIM8cCg2PiqVCeYCVeg3Do'


Hash matched.....
Total Execution Time :  1.2021770002320409
```

Figure (7): Twofish Decryption & hash matching

6 Evaluation

This section provides details on a thorough examination of the proposed hybrid model in relation to a number of parameter metrics, including encryption time, avalanche effect, throughput, and decryption time. For analytical purposes, plain text passwords are collected in various lengths, and the length of passwords is determined (in MB). Performance parameters including encryption-decryption time, overall execution time and throughput, and avalanche effect are used to evaluate this paradigm.

The total amount of time required for Argon2 hashing is known as hashing time and the amount of time required for Twofish encryption combined to encrypt the plain text is known as the encryption time, and the total time taken by the whole hybrid system to encrypt and decrypt the plaintext is known as total execution time. According to (Rizvi et al., 2011), the effectiveness of a cryptography technique is inversely related to the execution time. This implies that a method will be more efficient if it takes less time to encrypt & decrypt the plain text.
On the other side, the throughput value can also be used to gauge an algorithm's performance. It can be estimated by dividing the plain text size by the overall execution time. It is directly correlated with performance, meaning that the more throughput, the better the algorithm performance.

The Avalanche effect measures how many bits of the output values are changed if a single bit is flipped in the input value. In high-quality cryptographic algorithms, a minor change in the input plaintext should result in a drastic alteration in the resultant cipher-text value.

$$\boxed{\text{Avalanche Effect} = \frac{\text{Number of flipped bits in ciphered tex}}{\text{Number of bits in ciphered text}}}$$

The security of the algorithm is measured by emulating the avalanche effect, as ("Cryptography and Computer Privacy," n.d.) stated through his research theory that the higher the avalanche effect of an algorithm the higher the security of that algorithm. We executed our suggested hybrid model, and noted the encryption, decryption times, and calculated the throughput value & avalanche impact, in order to assess the model.

6.1 Analysis of encryption, and decryption time with respect to varied data size.

We have taken a varied lengths of passwords in order to figure out whether varying lengths of password data really impact the encryption and decryption time or not. In order to observe it we have plotted the graph which is shown below in figure (8), where we can see that as the size of the password is increasing the time taken for encryption & decryption is also increasing.

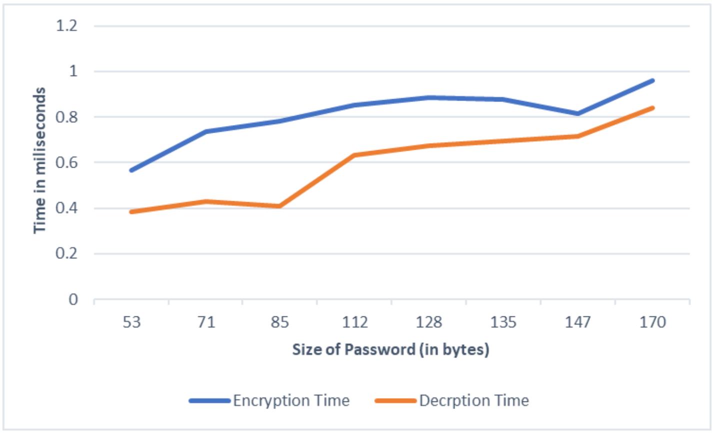


Figure 8: Encryption & decryption time w.r.t. size of password

The overall time taken for hashing, encryption and decryption is the total execution time. Through the below figure (9), we can observe that the total execution time increases with the increase in the size of the password. This depicts that both of them are directly proportional.

Total Execution time = (Hashing time + Encryption time + Decryption time)

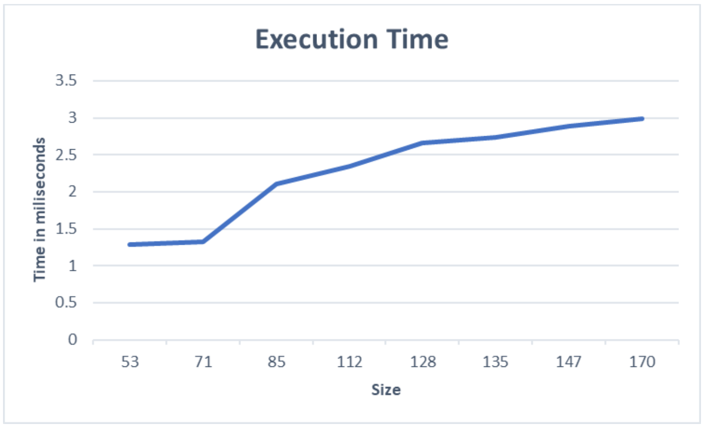


Figure (9): Total execution time w.r.t. size of the password

Analysis of the proposed models' execution time and AES encryptions' execution time with respect to the size of the password is shown in the figure (10).

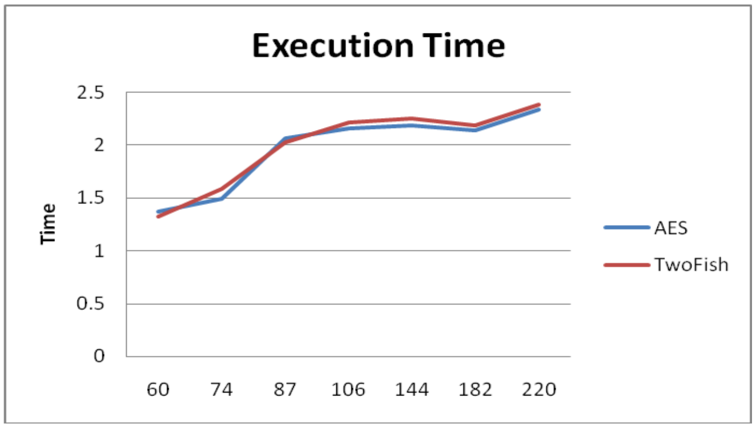


Figure (10): Execution time of algorithms with respect to the size of the password

6.2 Analysis of encryption, and decryption time with respect to different key sizes

In order to evaluate the different key lengths does the time taken by the proposed hybrid model for encryption and decryption change or not. In the below figure (11), we can observe that when the key size has varied the complexity of the password increases and the encryption, and decryption time increases with respect to varying key sizes.

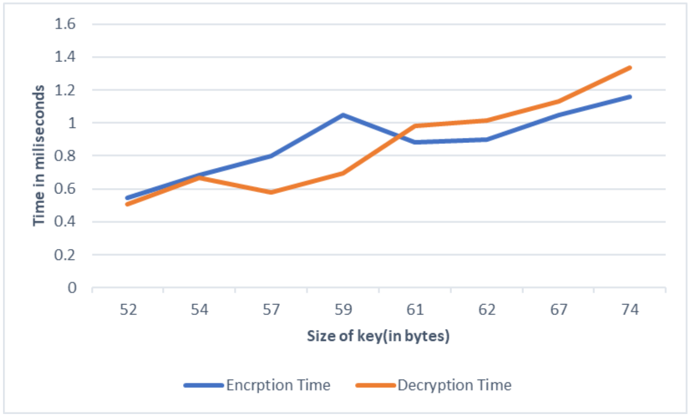


Figure (11): Relationship of the size of key w.r.t. encryption, decryption time

The total execution time varies with the different key sizes as when the key size increases the password complexity increases which makes the processor take a longer time for encryption and decryption. The below figure (12) depicts the observation:

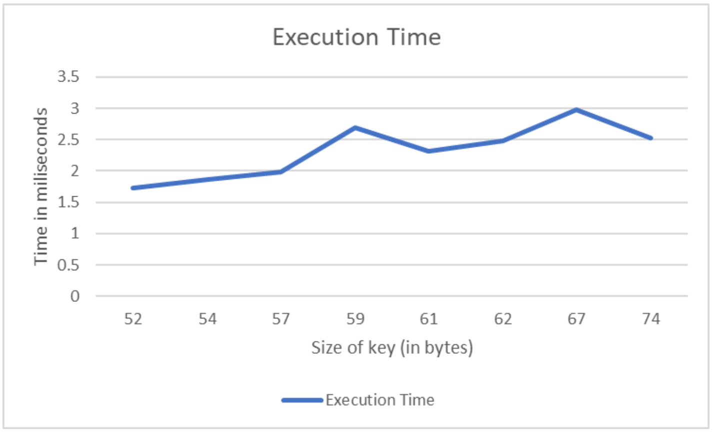


Figure (12): The relationship between total execution time and size of the key

6.3 Analysis of Avalanche Effect

The security of the proposed model can be analyzed by checking the avalanche effect of the model and for this, we have taken a dataset of 10 passwords with variable lengths so that we can find out the relation between the avalanche effect and the size of the password.

For the calculation of the avalanche effect, the plaintext passwords (before change) are converted to bits (B1) and a minor change is made to the plaintext password. After which, this changed plaintext password is converted to bits (B2). Now, these two values (B1 & B2) are compared and the number of bits flipped is observed upon which they are divided by the total number of cipher bits.

From the below figure (13), we can find that the avalanche effect is showing a slight increase in the avalanche percentage with respect to the length of the password.

The dataset of passwords with variable password lengths is shown below table (1):

Plaintext Password (before change)	**Size of Password (in bytes)**	**Plaintext Password (after change)**	**Avalanche Effect**
apple	54	appld	50.68%
appleisgood	60	appleisgooc	50.91%
appleismyfavouritefruit	72	appleismyfavouritefruis	50.86%
appleismyfavouritefruitandilike it	82	appleismyfavouritefruitandilik eis	51.07%
appleisgoodforhealthandeveryo neshouldeatapple	94	appleisgoodforhealthandevery oneshouldeatappld	51.12%
appleisgoodforhealthandeveryo neshouldeatappleandband	101	appleisgoodforhealthandevery oneshouldeatappleandbanc	51.02%
appleisgoodforhealthandeveryo neshouldeatappleasitgivesyouh ealth	112	appleisgoodforhealthandevery oneshouldeatappleasitgivesyo uhealtg	51.13%
appleisgoodforhealthandeveryo neshouldeatappleasitgivesyouh ealthandother	120	appleisgoodforhealthandevery oneshouldeatappleasitgivesyo uhealthandotheq	51.28%
appleisgoodforhealthandeveryo neshouldeatappleasitgivesyouh ealthandotherthanthisitisgoodfo rbrain	144	appleisgoodforhealthandevery oneshouldeatappleasitgivesyo uhealthandotherthanthisitisgo odforbraim	51.21%
appleisgoodforhealthandeveryo neshouldeatappleasitgivesyouh ealthandotherthanthisitisgoodfo rbrainaswellasforstomach	162	appleisgoodforhealthandevery oneshouldeatappleasitgivesyo uhealthandotherthanthisitisgo odforbrainaswellasforstomacg	51.39%

Table (1): Dataset of 10 passwords with their size (in bits) and the respective avalanche effect

The below figure (13) depicts the relationship between the avalanche effect and the length of size:

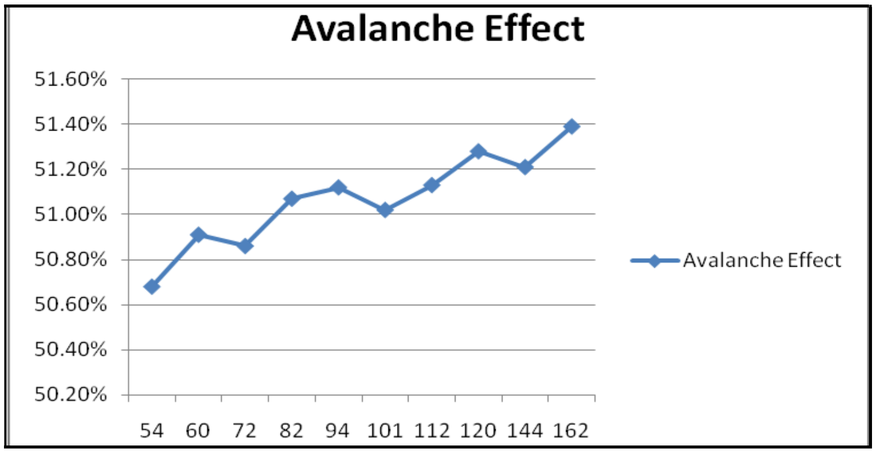


Figure (13): Analysis of Avalanche effect with respect to the size of the password

6.4 Discussion

The proposed hybrid model has been extensively evaluated on various parameters and these parameters include encryption time, decryption time, avalanche effect, throughput etc. From the literature reviews it was clear that Argon2 is the best hashing method which is also an award-winning hashing algorithm, due to this we have deployed this hashing technique in our hybrid model. We have used the Twofish encryption methods in order to add an extra layer to the hybrid model. We have compared the avalanche effect with the length of the password, through this analysis, we get to know that there is a significant increase in the avalanche when the size of the password is subsequently increased. Other than this we compared the dependency of the size of the key with respect to total execution time and it was found that the total execution time increases with the increase in key size. It was also found that the total execution time required for the proposed model and AES encryption method is closer with a negligible difference. Seeking the higher avalanche value of the proposed model we can conclude that the proposed hybrid model is more secure as ("Cryptography and Computer Privacy," n.d.) stated that a higher avalanche value states that the model is more secure & complex to crack. As a result, in this study, the proposed hybrid model of Twofish encryption with Argon2 hashing outperforms the existing models in terms of performance and has been shown to be more secure against brute-force password-cracking attempts. The below given figure (14) shows the throughput of Twofish encryption with compared to AES encryption, and the table (2) depicts the comparison of proposed hybrid model with AES encryption on parameters such as throughput value, total execution time taken.

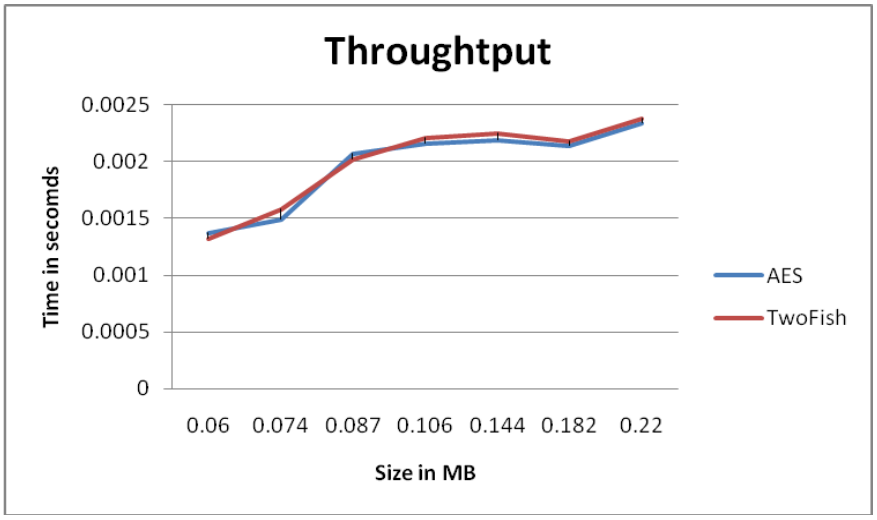


Figure (14): Graph showing the throughput of AES and Twofish encryption

Size of Password (in bytes)	**Size of Password (in MB)**	**AES Total Execution Time (in seconds)**	**Twofish Total Execution Time (in seconds)**	**Throughput Value of AES (in MB/sec)**	**Throughput Value of the proposed model (in MB/sec)**
60	0.000060	0.00137	0.00132	0.04379	0.04545
74	0.000074	0.00149	0.00158	0.04966	0.04683
87	0.000087	0.00207	0.00202	0.04202	0.04306
106	0.000106	0.00216	0.00221	0.04907	0.04796
144	0.000144	0.00219	0.00225	0.06575	0.06401
182	0.000182	0.00214	0.00218	0.08504	0.08348
220	0.000220	0.00234	0.00238	0.09401	0.09243

Table (2): Throughput value comparison of AES encryption and proposed hybrid model

7 Conclusion and Future Work

The major goal of this study was to determine whether we could increase an online user's password security by employing a hybrid mix of Argon2 hashing along with Twofish encryption to thwart brute force and other password-cracking attacks. Twofish is one of the most powerful memory-accelerated encryption schemes, while Argon2 has been demonstrated to be the best memory hard hashing method. By sequentially putting the password through Argon2 hashing first and then to Twofish encryption, we have enhanced its overall complexity. Using the Argon2 Key derivation function, we have generated a hashed value. We supplied this hashed value as input to the Twofish encryption step, from which the cipher-text was produced, to improve the difficulty of the hashed password value. The outcomes of the suggested model demonstrate how effective and performant this hybrid algorithm is when compared to existing models as discussed in the literature review. Due to this, brute force attacks against the password are made more challenging for the attackers. This study comes to the conclusion that using a blend of Argon2 hashing and Twofish encryption successfully boosts a user's password security against brute-force attacks.

The future scope of this research could be to examine if utilizing alternative combinations of numerous hybrid algorithms, either sequentially or concurrently, will considerably boost the security of passwords. In addition, numerous encryption and hashing techniques can be merged to build a strong hybrid model which can further raise the level of password security to a higher level.

References

Alvarez, R., Andrade, A., Zamora, A., 2018. Optimizing a Password Hashing Function with Hardware-Accelerated Symmetric Encryption. Symmetry 10, 705. https://doi.org/10.3390/sym10120705

Biryukov, A., Dinu, D., Khovratovich, D., 2016. Argon2: New Generation of Memory-Hard Functions for Password Hashing and Other Applications, in: 2016 IEEE European Symposium on Security and Privacy (EuroS&P). Presented at the 2016 IEEE European Symposium on Security and Privacy (EuroS&P), pp. 292–302. https://doi.org/10.1109/EuroSP.2016.31

Biryukov, A., Dinu, D., Khovratovich, D., Josefsson, S., 2021. Argon2 Memory-Hard Function for Password Hashing and Proof-of-Work Applications (Request for Comments No. RFC 9106). Internet Engineering Task Force. https://doi.org/10.17487/RFC9106

Chandra, S., Paira, S., Alam, S.S., Sanyal, G., 2014. A comparative survey of Symmetric and Asymmetric Key Cryptography, in: 2014 International Conference on Electronics, Communication and Computational Engineering (ICECCE). Presented at the 2014 International Conference on Electronics, Communication and Computational Engineering (ICECCE), pp. 83–93. https://doi.org/10.1109/ICECCE.2014.7086640

Cryptography and Computer Privacy [WWW Document], n.d. URL https://www.apprendre-en-ligne.net/crypto/bibliotheque/feistel/index.html

Elminaam, D.S.A., Kader, H.M.A., Hadhoud, M.M., 2008. Performance Evaluation of Symmetric Encryption Algorithms.

Ertaul, L., Kaur, M., n.d. Implementation and Performance Analysis of PBKDF2, Bcrypt, Scrypt Algorithms.

Hirani, S.A., 2003. Energy Consumption of Encryption Schemes in Wireless Devices [WWW Document]. URL http://d-scholarship.pitt.edu/7620/

Kumar, N., Chaudhary, P., 2018. Password Security Using Bcrypt with AES Encryption Algorithm. pp. 385–392. https://doi.org/10.1007/978-981-10-5544-7_37

Liu, E., Nakanishi, A., Golla, M., Cash, D., Ur, B., 2019. Reasoning Analytically about Password-Cracking Software, in: 2019 IEEE Symposium on Security and Privacy (SP). Presented at the 2019 IEEE Symposium on Security and Privacy (SP), pp. 380–397. https://doi.org/10.1109/SP.2019.00070

Marton, K., Suciu, A., Ignat, I., 2010. Randomness in Digital Cryptography: A Survey. Romanian J. Inf. Sci. Technol. 13.

Melicher, W., Ur, B., Segreti, S.M., Komanduri, S., Bauer, L., Christin, N., Cranor, L.F., n.d. Fast, Lean, and Accurate: Modeling Password Guessability Using Neural Networks.

Modugula, R.S.R., n.d. A Hybrid approach for Augmenting password security using Argon2i hashing and AES Scheme.

Nadeem, A., Javed, M., 2005. A Performance Comparison of Data Encryption Algorithms, IEEE Information and Communication Technologies. https://doi.org/10.1109/ICICT.2005.1598556

Percival, C., Josefsson, S., 2016. The scrypt Password-Based Key Derivation Function (Request for Comments No. RFC 7914). Internet Engineering Task Force. https://doi.org/10.17487/RFC7914

Rizvi, S.A.M., Hussain, S.Z., Wadhwa, N., 2011. Performance Analysis of AES and TwoFish Encryption Schemes, in: 2011 International Conference on Communication Systems and Network Technologies. Presented at the 2011 International Conference on Communication Systems and Network Technologies, pp. 76–79. https://doi.org/10.1109/CSNT.2011.160

Singh, G., Kinger, S., 2013. A Study of Encryption Algorithms (RSA, DES, 3DES and AES) for Information Security. Int. J. Comput. Appl. 67, 33–38. https://doi.org/10.5120/11507-7224

Singh, P., Kaur, K., 2015. Database security using encryption, in: 2015 International Conference on Futuristic Trends on Computational Analysis and Knowledge Management (ABLAZE). Presented at the 2015 International Conference on Futuristic Trends on Computational Analysis and Knowledge Management (ABLAZE), pp. 353–358. https://doi.org/10.1109/ABLAZE.2015.7155019

Sriramya, P., Karthika, R.A., 2015. Providing password security by salted password hashing using Bcrypt algorithm. ARPN J. Eng. Appl. Sci. 10, 5551–5556.

Turan, M.S., Barker, E.B., Burr, W.E., Chen, L., 2010. Recommendation for password-based key derivation :: part 1: storage applications (No. NIST SP 800-132). National Institute of Standards and Technology, Gaithersburg, MD. https://doi.org/10.6028/NIST.SP.800-132

Tyagi, K., Yadav, S.K., Singh, M., 2021. Novel cryptographic approach to enhance cloud data security. J. Phys. Conf. Ser. 1998, 012022. https://doi.org/10.1088/1742-6596/1998/1/012022

Wetzels, J., n.d. Open Sesame: The Password Hashing Competition and Argon2.

Yu, F., Huang, Y., 2015. An Overview of Study of Passowrd Cracking, in: 2015 International Conference on Computer Science and Mechanical Automation (CSMA). Presented at the 2015 International Conference on Computer Science and Mechanical Automation (CSMA), pp. 25–29. https://doi.org/10.1109/CSMA.2015.12