A Comparative Study Between Two Cybersecurity Attacks: Brute Force and Dictionary Attacks

Nabaa Alaa Hussian
Computer Science Department
College of Science for Women
Babylon University
Babylon, Iraq
nabaa.alzeade.gsci148@student.uobabylon.edu.iq
Orcid.org/0009-0008-0124-2091

Farah Al-Shareefi
Computer Science Department
College of Science for Women
Babylon University
Babylon, Iraq
wsci.farah.mohammed@uobabylon.edu.iq

DOI: http://dx.doi.org/10.31642/JoKMC/2018/110216
Received Apr. 5, 2024. Accepted for publication May. 4, 2024

Abstract**—** *In today's world, information drives not only business but nearly every aspect of human life. Therefore, safeguarding valuable information from malicious activities like attacks has become essential. Brute force search and dictionary attacks are prevalent cybersecurity threats, in which an attacker systematically attempts all possible passwords and passphrases to gain access to a user's account. These types of attacks are common due to the widespread reuse of simple password variations. The goal of this study is to discover as many passwords as possible and demonstrate their predictability and susceptibility. Our focus was on comparing two cracking methods, such as brute force search attacks and dictionary attacks, assessing their effectiveness and time requirements.*

Keywords— Dictionary Attack, Brute Force Attack, Password Based Encryption, Secure Hash Algorithm

I. INTRODUCTION

For the last forty years, the struggle against user tendencies and inertia has continued unabated. With technological advancements, even longer passwords are becoming increasingly susceptible, dictionaries are expanding daily, and all measures cannot prevent users from making poor choices [1]. Moreover, password policies may inadvertently aid attackers by exposing the minimum search space. Furthermore, since users often seek to comply with rules with minimal effort and follow recognizable patterns, enemies might give these passwords priority in their assaults while ignoring less secure options. Changing passwords on a regular basis is another important password security measure. However, most users still neglect to update their password once they've set it, even after efforts to educate them about this problem[2].

The primary goal of an attacker is to compromise a cryptosystem and uncover the plaintext from the ciphertext. Finding the secret decryption key is the sole requirement for obtaining the plaintext, since the algorithm is already publicly available. As a result, the attacker dedicates maximum effort to discovering the secret key used in the cryptosystem. Once the key is determined, the system is considered compromised [3]. Various attacks can be employed against cryptosystems, including brute-force and dictionary attacks.

Brute-force and dictionary attacks are both types of cyber security breaches wherein the assailant tries to gain access to an account by testing various passwords until the correct one is found. These widely-used attacks can be successful because many users rely on common variations of a few passwords. Brute-force and dictionary attacks utilize distinctive systematic methods, rather than merely inputting random combinations of characters [4].

The reason why these attacks are so popular is that once attackers obtain the right password for an account, they may access it and Log in the same way as any other user without the need for additional exploitation or bypassing security controls. A brute-force attack or an efficient dictionary is especially rewarding for the attacker when the compromised account is that of a system administrator, which typically has higher privileges than a regular user account [5]. Encryption is utilized as an effective method to safeguard critical information from hacking. Password Based Encryption (PBE) is one of these approaches. This cryptographic approach is symmetric, i.e., utilizing a similar password for both encryption and decryption processes, ensuring that the output matches the original plaintext input. Encrypted plaintext data will result in cipher text that is

unreadable to unauthorized individuals. Chipher text is the information that will be transmitted to a second party with guaranteed confidentiality. The chipper text data will be altered based on the supplied password. PBE cryptography relies on the hashing method. Essentially, in PBE the password and salt (random generated number) will be merged to create random data using the applied hashing function. This data will then undergo processing based on the iteration count to produce encrypted data (more secured password) once the mixing procedure is completed [6]. This research aims at studying these two types of attacks and discussing the main differences between them.

II. RELATED WORK

As our work examines the brute force and dictionary attacks against the password-based encryption, this section reviews the literatures that have been conducted in the similar direction.

The researchers in [7] developed a cloud computing system based on brute force by salting the user's password using hash functions to help users creating a strong password. After testing the brute force system, the tested results indicated that the brute force was unsuccessful 100% .

In the study of [8], the researchers carried out a large-scale attack using a number of proven hacking methods, including dictionary, hybrid, and brute force assaults, on passwords that Slovenian University students needed to access an online grading system. They wanted to demonstrate how easy it was to break passwords created by users by employing straightforward patterns. They tested the strength of both cracked and non-broken passwords in order to identify the differences between them. The findings shown that, in a matter of days, a single contemporary low- to mid-range GPU could break through over 95% of passwords, but a more specialized system could break through all but the toughest 0.5% of passwords.

In [9], a study on a cloud computing brute-force prevention solution was conducted. Using a cryptographic hashing function, it generates a new encryption key by salting the user's password with an arbitrary value, therefore implementing the symmetric key encryption process as a safeguard against brute-force attacks. The purpose of a password policy was to help users create strong passwords. Utilizing a one-time password increased system security. The purpose of this brute-force avoidance technology is to help users create strong passwords. The results of testing the system to prevent brute-force attacks revealed that successful brute-force attacks were at 0% and unsuccessful brute-force attacks were at 100%. Furthermore, a strong password can aid in thwarting brute-force assaults. If a brute-force assault was successful, though, it was probably because users didn't adequately secure their login information or disclosed it. According to the study's findings, the password regulations that are in place give the brute-force protection system a high level of resistance against brute-force attacks.

In [10] an analysis of an IoT network File Transfer Protocol (FTP) server brute force attack (BFA) was conducted by the researchers in reference [10]. To find its settings, they used a time-sensitive statistical relationship approach and created a visual representation of the attack patterns. Assuming that a firewall is in place for the IoT network, the investigation's focus is on attacks that originate from within the network. Experiments were conducted on an IoT network testbed that simulates an internal attack scenario with three main goals in mind: (i) topologically describing the attack mechanism of an insider; (ii) extracting attack patterns from raw sniffed data; and (iii) using attack pattern identification as a parameter to visualize attacks in real time. Their experimental results verified the investigation. Researchers carefully examined the SHA family of secure hashing algorithms in terms of launching brute-force assaults[20]. Due to its larger hashes, which make brute-force attacks more difficult, they discovered that SHA-512 is noticeably more secure than SHA-256, SHA-0, and SHA-1.

III. THEORETICAL BACKGROUND

The following sub sections explain the theoretical principles of the used methods in our system.

1. ENCRYPTION METHODS

Typically, two primary categories of encryption techniques exist[9, 15]: Symmetric Encryption (private-key encryption) and Asymmetric Encryption (public-key encryption).

Both of these techniques have comparable cryptographic strength, but asymmetric encryption needs more computer power and more mathematics than symmetric encryption. As a result, the focus of this research is on symmetric encryption and a similar technique known as the password-based encryption method, which is covered in depth below.

SYMMETRIC ENCRYPTION (PRIVATE-KEY ENCRYPTION)

The most basic kind of encryption uses a single secret key to both cipher and decode data. Symmetric encryption is the most established and well-known technique. It uses a secret key, which might be an arbitrary phrase, integer, or string of characters. It is mixed in with the plain text to change a message's content in a certain way. Each communication needs a secret key, which the sender and the recipient should know in order to encrypt and decode it. Along with password-based encryption, symmetric encryption uses Blowfish, DES, AES, RC4, DES, RC5, and RC6. DES, AES-128, AES-192, and AES-256 are the most often used symmetric algorithms [10]. Symmetric key encryption's principal flaw is that decoding the data

necessitates sharing the encryption key amongst all parties involved[6,9].

PASSWORD-BASED ENCRYPTION (PBE)

Password-based encryption is a popular method for creating safe cryptographic keys. The strength of the secret key determines the cipher's power. Because passwords are frequently composed of subsets of ASCII or UTF-8 characters that are easy to remember, a strong secret key should have unpredictable characters. Therefore, it is not possible to simply create the secret key from the user's password.

Password-based encryption allows the production of strong secret keys based on passwords supplied by the user. Creating key bytes that are as surprising and unpredictable as possible is the aim. In addition to the user's password, the PBE algorithm also takes into account two other input parameters: salt and iteration count. Using a secure hash function as the foundation for the mixing function, PBE applies a number of times (specified by an iteration count). After mixing, the resulting bytes are utilized to construct the cipher's key, along with the initialization vector if needed[9]. Remember that the length of the random integer that makes up the salt is identical to the output size of the hash function. We recommend running the PBE algorithm at least 1000 times to produce a sufficiently complicated key. As a result, the process for generating the secret key would be significantly lengthier[10].

HASHING ALGORITHM

An encryption hash function is a hashing algorithm. It is a mathematical technique that converts arbitrary-sized data into a fixed-sized hash. The purpose of a hash function algorithm is to be a one-way function that is impossible to reverse, unintelligible, unscrambled, and decoded by another person. Since hashing secures data while it's at rest, the files on your server stay unreadable even if someone manages to get access to it. Additionally, hashing can assist you in demonstrating that data is not changed or updated once the author has completed it [11]. Although there are several hashing algorithm kinds, SHA-1, or Secure Hash Algorithm 1, is the one used in this study[12]. It receives an input and outputs a hash value that is 160 bits (20 bytes). Since 2005, SHA-1 is now regarded as insecure.

CYBERSECURITY ATTACKS

Cyberattacks are defined as hostile acts directed on computer networks, computer information systems, personal computers, or infrastructures. An attacker is a process or someone who tries to get data and carry out actions in other systematically restricted areas without permission or maybe with malevolent intent[4]. Although there are many other types of security assaults, the ones that were compared in this study are shown below.

BRUTE FORCE SEARCH ATTACK

A brute force attack is a kind of hacking method that uses trial and error to crack passwords, login credentials, and encryption keys. It is a simple yet efficient method for gaining unauthorized access to networks, systems, and user accounts within businesses. The hacker tries a range of usernames and passwords until they find the correct login information, usually attempting a wide range of combinations on a workstation[5]. Using a lot of force to try to get access to user accounts is known as "brute force" assault techniques. Although brute force attacks are an older type of cyberattack, hackers still employ it as a tried-and-true method[4] .

DICTIONARY ATTACK

In a basic brute force assault, the attacker selects a target and then compares prospective passwords to the username of the target. This technique is known as a dictionary attack. While the attack method isn't really a brute force attack, it may play a big role in a malicious actor's attempt to break into a password. "Dictionary attack" describes the technique wherein hackers use dictionary scanning to manipulate words containing special characters and integers [14]. In general, this type of attack is less successful and requires a lot of time compared to more modern and effective attack methods.

IV. THE DESIGNED METHOD

The purpose of this study is to implement a program that implements one of the selected attacks: brute force or dictionary attack, see Figure 3.1 that illustrates this method schematically. The time taken by a conducted attack is measured. When the program conducts the brute force attack then it will take passwords of varying length as user input. This password is then to be used to perform encryption of a predefined plaintext. The program will then again ask the user for its password, and would decrypt the ciphertext back to the plaintext using this password. The Password-Based Encryption (PBE) technique handles both encryption and decryption. The encryption/decryption process carried out in the first section of the program is then subjected to a brute force search assault by the software. In a brute force search attack, the attacker tries each password once in an attempt to figure out what the user's password is. Although it is believed that the attacker is unaware of the password, it is aware of the following parameters:

- The predefined plaintext
- The ciphertext produced by the encryption process
- The salt
- The iteration count

Therefore, the brute force search attack class should allow the user to enter their password and encrypt the plaintext, and then attempt every password in sequence, starting from the lowest, up to the maximum value that the password could be, encrypt the plaintext again, and compare the resultant ciphertext with that generated by the user at the start, If the ciphertexts matched, then the passwords used is the same, so the attack will find the user's password.

In case that the dictionary attack is chosen, then it is assumed that the attacker is given:

- passwords.txt file that contains a list of passwords recovered by using the attack.
- englishDic is the dictionary used during the attack to recover passwords

SHA-1 is employed to hash the passwords. When a salt is used, the passwords and salt are simply concatenated as salt + password. The attack consists of reading the dictionary file line by line and calculating ten distinct hashed passwords for each word. Each password in the password.txt file is compared to these ten potential hashes to check whether there is a match. We have recovered a password if there is a match. If not, we just read the dictionary word by word again. The 6 possible hashes computed for each word from the dictionary file are:

- SHA1(password)
- SHA1(adoprssw) (ascending order)
- SHA1(wssrpoda) (descending order)
- SHA1(drowssap) (reversed word)
- SHA1(psswrd) (word without vowels)
- SHA1(salt + password) (salted word)
- SHA1(salt + drowssap) (salted reversed word)
- SHA1(salt + adoprssw) (salted ascending order)
- SHA1(salt + wssrpoda) (salted descending order)
- SHA1(salt + psswrd) (salted word without vowels)

V. RESULTS AND DISCUSSION

This chapter will show the result obtained from running the brute force attack program on Password Based Encryption algorithm, and will show the result obtained from running the dictionary attack program.

ANALYSING BRUTE FORCE SEARCH ATTACK

Once our method is created and tested as working, 2 runs of performance analysis tests were performed. Firstly, the effect of password length on search time was examined. This was done by performing multiple runs of the brute force search attack on different password lengths, with a fixed iteration count value. Furthermore, the effect of the iteration count value on the attack performance was also investigated. In this test, 3 fixed 5-character passwords were tested against different iteration count values, and the results obtained were averaged.

For both tests, the time for the password to be found was measured in nanoseconds, seconds, and minutes using the system time at the start and end of the search. All tests were performed on a laptop equipped with an Intel Core i7-4700MQ processor running at 2.4GHz, with 12GB of available RAM. The plaintext used throughout was the string "Hello, you found me brute!".

A) Password Length

As you can see from the graph below, the average time to find the correct password for the predefined plaintext/ciphertext, with fixed value of the iteration account, depends on the length of the password. The longer password means the more time that is taken to work out the correct password. This is increasing on a scale of around 10 every time.

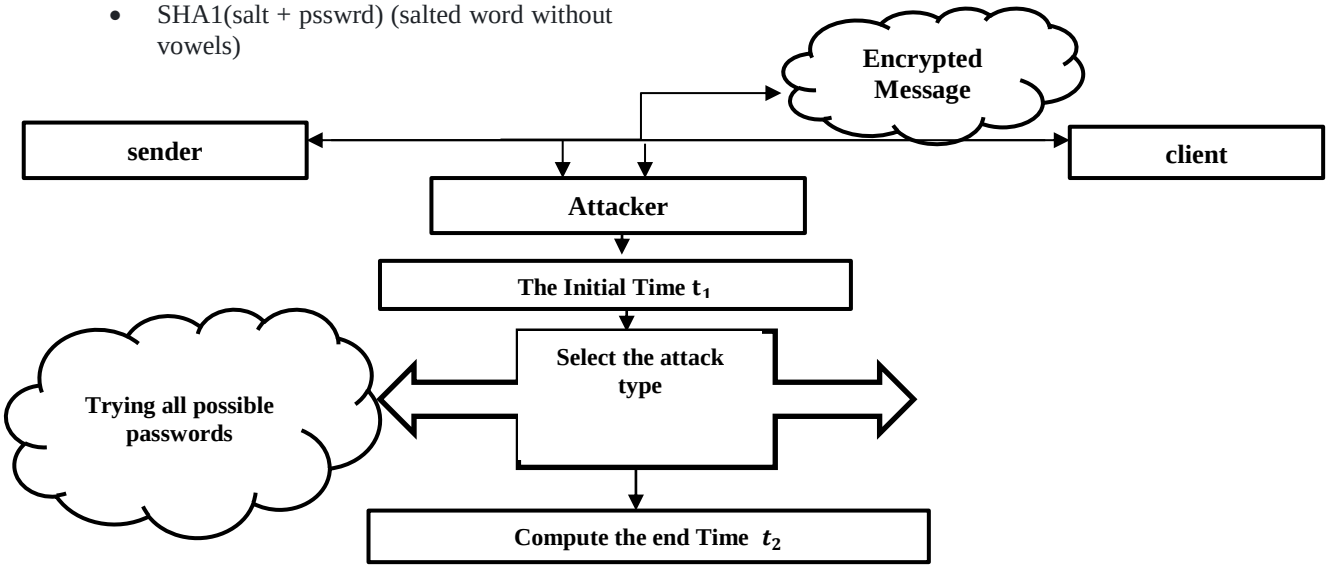
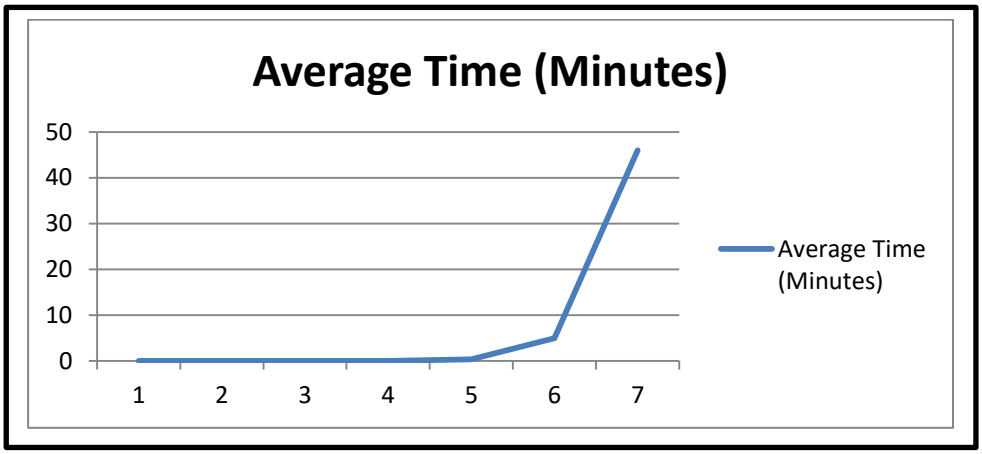


Figure 1. Block Diagram of the Designed Method

Length of password	Attempt 1 (NanoSeconds)	Attempt 2 (NanoSeconds)	Attempt 3 (NanoSeconds)	Average Time (NanoSeconds)	Average Time (Seconds)	Average Time (Minutes)
1 Digit	13345490	12507280	12211498	12688089.33	0.012688089	0.000211468
2 Digits	87587833	91443598	81105641	86712357.33	0.086712357	0.001445206
3 Digits	444308810	472888789	470330975	462509524.7	0.462509525	0.007708492
4 Digits	3178089973	4655326308	3154056677	3662490986	3.662490986	0.061041516
5 Digits	28752893126	27411712184	29175585461	28446730257	28.44673026	0.4744112171
6 Digits	2.73E+11	2.83E+11	2.9285E+11	2.82865E+11	282.865	4.714416667
7 Digits	2.82E+12	2.66E+12	2.91002E+12	2.79695E+12	2796.95	46.61583333

Table 1 Time Taken by Brute Force Attack Depending on Different Password Lengths



gure 2 The Effects of Password Length on Time Taken to Complete Brute Force Attack

B) Iteration count

Regarding the iteration count affects the time of, it does, by lowering the iteration count from 1024, to 512, you can see that the time it takes to compute these gets quicker and quicker. This is shown in the below graph , which has the average times taken for each password length of each iteration. This shows the times getting smaller when the smaller iteration count is considered.

As for an attack in which the attacker knows everything, except the iteration count, we can assume that the time taken to calculate these will be longer than the previous attempts. This is because you are doing for every pass for every iteration. Meaning an exponential amount of combinations that need to be tried.

Password Length	Iteration Count		
	256	512	1024
1	6801408.667	8416875.667	12688089.33
2	098157.33262252301.3	68320868.67	86712357.33
3	262252301.3	325875388	462509524.7
4	1322953332	2036023008	3662490986
5	8626930709	18565476768	28446730257

Table 2 Time Taken by Brute Force Attack Depending on Different Iteration Count Values

ANALYZING DICTIONARY ATTACK

Once our method is created and tested as working, 1 run of performance analysis test is performed. Typically, the effect of 10 possible hashing passwords on search time was examined.

The time for the password to be found was measured in milliseconds using the system time at the start and end of the search. All tests were performed on a laptop equipped with an Intel Core i7-4700MQ processor running at 2.4GHz, with 12GB of available RAM.

As you can see from Table 4.1, the time to find the correct password for the predefined plaintext/ciphertext, depends on the taken hashing possibility of the password. The more complicated hashes password means the more time that is taken to work out the correct password. Basically, both of the salted reversed password and salted password without vowels take the longest time which is equal to 2329 ms.

Hashing Password Possibility	Time in ms
word (December)	140
reversed word (tfosorciM)	156
word (Monday)	156
reversed word (brosba)	218
reversed word (yllacitebahpla)	406
word without vowels (ntrstwrthnss)	2282
salted word (uplifting)	2282
Salted word in ascending order (aery)	2290
Salted word in descending order (yrea)	2290
word without vowels (vsblts)	2298
salted word without vowels (wrrsm)	2329
salted reversed word (kcitsdray)	2329

Table 3 Time Taken by Dictionary Attack Depending on 6 Possible Hashing Passwords

According to the above results we can conclude the main differences between both attacks in the table below:

Comparison Parameter	Brute Force	Dictionary Attack
Definition	Every plausible password combination is attempted by the attacker.	A pre-collected list of popular passwords is checked by the attacker.
Effectiveness	If the password is concise, brute force is performs better.	If the password is repeatedly used, Dictionary Attack performs better.
Duration of an attack	The length of the password and iteration count affect length of time at attack takes.	The length of the dictionary affects time taken by an attack.

Table 4 Brute Force vs. Dictionary Attack

VI. CONCLUSIONS

In conclusion, even though dictionary attacks and brute force attacks are common ways to compromise cyber security, but they are not same successfulness. Brute Force is a good attack for breaching short random passwords, on the other hand, Dictionary is a better tool for attacking long password based on real words or whatever words from a specific dictionary or a defined reference source. Attacks using brute force target a single character at a time, while Dictionary attacks completes their task by processing one password at a time. Dictionary attacks are a little harder to set up than brute force attacks, but they both are easy to implement.

REFERENCES

[1]Sharma, Anand, et al. "Password Based Authentication: Philosophical Survey." 2010 IEEE International Conference on Intelligent Computing and Intelligent Systems, IEEE, 2010.

[2]Martinez-Diaz, M., et al. "A Comparative Evaluation of Finger-Drawn Graphical Password Verification Methods." 2010 12th International Conference on Frontiers in Handwriting Recognition, IEEE, 2010.

[3]Klein, D. "Foiling the Cracker: A Survey of, and Improvements to, Password Security." Proceedings of the 2nd USENIX Security Workshop, 1990, pp. 5–14.

[4]Gautam, Tanvi, and Anurag Jain. "Analysis of Brute Force Attack Using TG-Dataset." SAI Intelligent Systems Conference (IntelliSys), IEEE, 2015.

[5] Dave, Konark Truptiben. "Brute-force attack 'seeking but distressing'." Int. J. Innov. Eng Technol Brute-force 2.3 (2013): 75-78.

[6] A. J. Menezes, P. C. van Oorschot, and S. A. Vanstone. Handbook of Applied Cryptography.CRCPress,October1996.

[7]Ayankoya, Folasade, and Blaise Ohwo. "Brute-Force Attack Prevention in Cloud Computing Using One-Time Password and Cryptographic Hash Function." International Journal of Computer Science and Information Security, vol. 17, 2019, pp. 7–19.

[8] L. Bosnjak, J. Sres and B. Brumen, "Brute-force and dictionary attack on hashed real-world passwords", Proc. 41st Int. Conv. Inf. Commun. Technol. Electron. Microelectron. (MIPRO), pp. 1161-1166, Feb. 2018.

[9] Ayankoya, Folasade, and Blaise Ohwo. "Brute-Force Attack Prevention in Cloud Computing Using One-Time Password and Cryptographic Hash Function." International Journal of Computer Science and Information Security, vol. 17, 2019, pp. 7–19.

[10] Stiawan, Deris, et al. "Investigating Brute Force Attack Patterns in IoT Network." Journal of Electrical and Computer Engineering, vol. 2019, 2019, pp. 1–13, doi:10.1155/2019/4568368.

[11]Verma, Rajat, et al. "Enhancing Security with In-Depth Analysis of Brute-Force Attack on Secure Hashing Algorithms." Proceedings of Trends in Electronics and Health Informatics, Springer Nature Singapore, 2022, pp. 513–522.

[12] A. Juels and J. Brainard. Client puzzles: A cryptographic defense against connection depletion attacks. In Network and Distributed System Security Symposium, 1999.

[13]Johnson, Leighton. "Security Component Fundamentals for Assessment." Security Controls Evaluation, Testing, and Assessment Handbook, Elsevier, 2016, pp. 531–627.

[14] Jallouli, O. "Chaos-Based Security under Real-Time and Energy To Cite This Version: Thèse de Doctorat Ons J ALLOULI." Univ. Nantes, 2017.

[15] M. Marras, "Dictionary attacks on speaker verification," IEEE Transactions on Information Forensics and Security, vol. 18, pp. 773–788, 2022.

[16] Easttom, W. Modern cryptography: applied mathematics for encryption and information security. Springer Nature, 2022.