ISSN: 2229-6948(ONLINE)
DOI: 10.21917/ijct.2022.0397

EFFICIENT OTP GENERATION WITH ENCRYPTION AND DECRYPTION FOR SECURE FILE ACCESS IN CLOUD ENVIRONMENT

R. Manjupargavi and M.V. Srinath
Department of Computer Science, Sengamala Thayaar Educational Trust Women's College, India

Abstract

Cloud Computing (CC) has grown increasingly important in recent years, and the security issues related to the cloud model are also on the rise. The most used computer system authentication mechanism is One Time Password (OTP) which is easier and more successful than other techniques while being subject to assaults such as eavesdropping and replay attacks. This study proposes the use of the OTP mechanism to tackle this problem. The user must utilize the OTP provided by the cloud environment, which is produced using the random function, to strengthen the security of data retrieval from the cloud. Documents may be safeguarded against unauthorized users using this OTP (secret key) generation or creation mechanism. Then it offers data encryption, which safeguards data files from a cloud provider. To encrypt the file, the cryptography library uses a symmetric approach based on the same key to encrypt and decode the file. The cryptography packages have been introduced in this work to provide built-in functions for key generation, cipher text encryption, and cipher text decryptions utilizing both encrypt and decrypt methodologies. The fernet module ensures that data encrypted with it cannot be edited or read if the key is not there. This will make it easier to exchange public data with other cloud users who have been verified. The processing time for secure authentication with OTP creation using 4, 6, 8, and 10 digits is reduced to 0.0016, 0.0028, 0.0035, and 0.0042 seconds, respectively, while the elapsed time for Encryption and Decryption algorithms such as Reverse Cipher, Caesar Cipher, and ROT13 is 0.00025, 0.00065, and 0.00207s.

Keywords:

One Time Password (OTP), Encryption, Decryption, Security, Cloud

1. INTRODUCTION

One of the most recent breakthroughs in the IT sector is cloud computing, often known as on-demand computing. CC has made everything more flexible and simpler in recent years. Data security in the cloud is a key worry, and different techniques have been presented. From the perspective of the user, the authors have done a detailed investigation to establish the most pressing security concerns in CC. In the proposed security paradigm, CC services are provided in a tiered medium, therefore some Service Level Agreement (SLA) or service management must be performed across the tiers, ultimately increasing the user's confidence. Based on two stages, this novel approach enhances Cloud security. CC is being utilized to improve the infrastructure and performance of businesses all over the world. However, for intended customers to use these services, the data of cloud users must be secure while transferred to the cloud, and cloud users must be authenticated before they can access the data [1]. To design a security system that will give cloud security and will be highly beneficial for both the user and the cloud data owner, allowing us to execute trusted computing in this paper.

As a result, generating OTPs is an important step in two-factor authentication. OTPs are often supplied to parties in one of two ways. In the first, an OTP is produced on one side and delivered to the other through SMS, email, or other means. In the latter, both parties use a common counter to create OTP at the same time. As a result of malicious attackers amassing passwords, fixed passwords are subject to re-use attacks. OTP, conversely, is a password that is used in a unique way for every login session or transaction. As a result, it is difficult to reuse previously obtained items. It also has the feature of unpredictability, which makes deducing the next password from the present one mathematically impossible. Because of these properties, it is recognized to be extremely safe, besides it might be used in ICT as an authentication service [2] domains like mobile payment [3] [4] in addition to mobile-CC [5] [6]. The passwords are produced by the OTP system by combining information about the time of occurrence of an event with previously disclosed secrets amongst the user and certifying authority. As a result, the attacker is unable to deduce this on his own. The outcome of a succession of unbiased "fair" coins produces the same random number sequence. The value of the bit is 0 or 1 on each side of the coin. The digits 0 and 1 are evenly distributed when thrown in sequence and each trial is run independently. As a result, it's the same as a perfect random number generator's output. Since each component of the random sequence is unrelated to the others, parts that have already been created cannot influence or be anticipated the next time [7]. This paper presents a technique for generating OTP using random bytes generated by any safe randomness.

Cryptography protects users by allowing them to encrypt data and verify the identity of other users. Data encryption is well recognized for its ability to keep information safe from prying eyes. It employs an encryption key to transform data from one format, known as plaintext, to another known as cipher text. At the moment, compression and encryption are carried out independently. Before the current era, cryptography was essentially identical to encryption, or the conversion of data from a readable to what appears to be a nonsense state. Data encryption comprises a random string of bits designed specifically to scramble and unscramble data. Data encryption employs procedures to ensure that each key is unique and random. Cryptography employs both asymmetric and symmetric keys. The most popular are symmetric keys, which employ the same key for both ciphertext's decryption as well as encryption. This is referred to as a secret key. The two most prevalent forms of secret-key ciphers are stream ciphers as well as block ciphers. It encrypts a block of data at the same time with a private key and algorithm, whereas a stream cipher does it one bite at a time. The manual decryption procedure and a technique of decryption utilizing the appropriate codes or keys. Encryption is the process of transforming plain text data into ciphertext. Decryption is the process of converting ciphertext to plaintext. This paper explains how to create a three-level security mechanism that uses a mix of biometrics and a one-time passcode to verify users and devices signing in. As a result, the focus of this research is on three levels

of security techniques that provide the CSP with high authentication for accessing data in the cloud environment.

The organization of the paper is described as follows: Section 2 summarizes the associated efforts in cloud platform based on OTP and symmetric key generation, and Section 3 describes the methodology for OTP method based on random function and key generation process based on a symmetric key algorithm, Section 3 discusses the results and discussion and Section 4 end with the conclusion.

2. LITERATURE REVIEW

In cryptography, pseudo-random bit streams are generated [8]. Nevertheless, it accounts for only a small portion of the OTP used in the bit stream. As a result, using an algorithm to extract parts is required by the OTP technique. The purpose of the study is to demonstrate the security and performance of the different security ideas, notably Advanced Encryption Standard (AES), Blowfish, and Twofish [9]. The safety aspects and security concerns of the blowfish, Twofish, and AES algorithms are discussed in this work. Then, using the three methods, encrypt a range of data types such as text, picture, and audio files to determine encryption speed. A Software Guard Extensions (SGX)- Unified Access Control (UAM), a unique unified access management strategy grounded on Intel software guard extensions, using OTPs (SGX) is presented [10]. The findings show that for a single login request, SGX-UAM takes roughly the same amount of time as OpenID and OAuth2.0 and that it performs consistently while processing several login requests. A technique for generating OTP using Pseudo-Random Number Generators (PRNGs), which are inherent to many operating systems and computer languages [11].

The Image-Based Password System (IBPS) produces an OTP based on the image selected by the user. In this work, random numbers have been generated using image properties and used as an OTP, which serves as a strong authentication factor [12]. A variety of cryptographic systems, including AES, is one of the most powerful. Confidentiality, authenticity, integrity, and nonrepudiation are all features of today's information security system. On the World Wide Web, communication security is a critical concern. It is all about maintaining confidentiality, integrity, and authentication while accessing or altering sensitive internal documents [13]. The picture steganography is achieved using a decoding Rivest–Shamir–Adleman (RSA) message and Hash-LSB encoding or file decryption and encryption, and chaos utilized an image decryption encryption approach [14]. A CC security strategy based on multilayer cryptography is presented [15]. The model takes a hybrid approach to key cryptography, combining symmetric and asymmetric approaches. In this case, the Data Encryption Standard (DES) and RSA are utilized to deliver multi-layer encryption and decryption on both the sender and receiver sides, therefore boosting cloud storage security. In comparison to the present system, this approach raises data security to the highest level possible and takes less time to upload and download text files. A paradigm for reducing cyber-attacks on cloud data secrecy is presented. This method is used to increase the strength of an encryption key by using chaotic random noise, as the strength is decided by the unpredictability of the input noise rather than the length of the key [16].

Visual Cryptography is proposed as an answer to the issue of information storage security [17]. The information is encrypted and stored on data servers. Only after the user's authentication does the service provider grant access to the shared image's keys. To gain access to the secret key, which serves as the key to the information, the user must now overlay the shared picture with this encrypted key. Security concerns are reduced in this study effort using cryptographic techniques. This proposed method enhances cloud storage security by employing several encryption algorithms such as the Feistel Algorithm and the AES algorithm with S-box. The proposed approach provides a solution to various cloud security tasks, like data protection from any infractions and protection against a user with a false authorized identity, both of which hurt cloud security. Multiple concerns and challenges with cloud computing are discussed in this study, many of which compromise data security and privacy. It discusses the dangers and assaults that influence cloud-based data. The findings improve the security fortification approaches for end-users and data owners in the cloud against false users and data owners [18] [19]. The study describes a way of securely storing information in the cloud. The files saved in the cloud are safe thanks to updated hybrid encryption. As a result, the model provides safe and secure file encryption and cloud storage [20].

This section summarizes the various level of security mechanisms that assist the Cloud Service Provider (CSP) with high authentication for accessing the data present in the cloud environment. However, this research aims at building a security technique that combines the use of fingerprint identification with OTP to identify users and applications before using the cloud services.

3. RESEARCH METHODOLOGY

The cloud side of our proposed approach is secure, and the user data is secure as well. This research work presented architecture based on which can guarantee security to both the cloud environment and the user. However, this research aims at building a security technique that combines the use of fingerprint identification with OTP to identify users and applications before using the cloud services. This algorithm will mandate the user/device forward both the finger-print recognition based Modified Artificial Neural Network (MANN) validates the user at the time of log-on to access the cloud services and the one-time passcode passed to the phone as SMS before starting to use the cloud services (invoking any cloud-based API). This is required in addition to the key, the user gets received while signing up with software deployed as a service. The proposed system encrypts both the key and the fingerprint using the OTP code provided and forwards it to the Cloud Service Provider (CSP) and logs in with a password from CSP is allocated using a CSP key for the user in terms of a cloud-based authenticator for validation. Moreover, this system mandates the reset of the CSP key on regular periodic bases for accessing the data or software in the cloud environment.

3.1 GENERATION OF OTP

The proposed OTP algorithm based on the randomization mechanism is described in this section. The Fig.1 depicts a broad overview of the proposed approaches. This system utilizes any secure random function that exists on your operating system or

programming languages, such as CryptGen Random function on Windows OS, /dev/random on Linux OS or Java Random(), and Python Random() classes. OTP is a password that is only valid for one login session or transaction on a computer or digital device. OTPs are now utilized in nearly every service, such as Internet Banking and online commerce. They usually consist of a mix of four or six numeric numbers or a six-digit alphanumeric code.

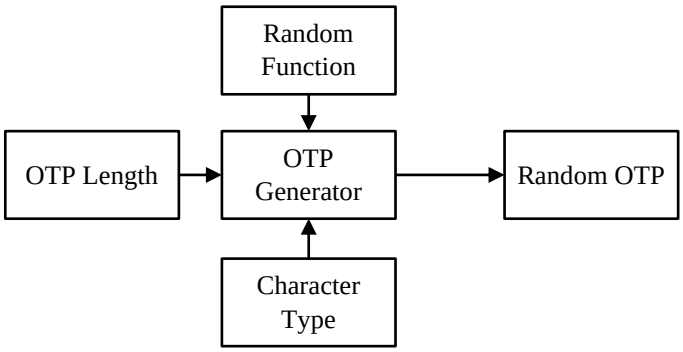

Fig.1. Proposed OTP method based on a random function

The requirements for generated OTP are as follows:

- **Length of OTP:** Required length of OTP must be between 6 and 10. This is a standard requirement for the length of OTP [2, 3].
- **Character type**: Generated OTP may consist numeric, upper alphanumeric and lower-upper alphanumeric characters: [0, … ,9], [0, … ,9, *A*, … , *Z*] and [0, … ,9, *a*, … , *z*, *A*, … , *Z*]. Special characters (for example, @,?,!) are not suitable for OTP because time is limited to input or the process may be unpleasant for users.
- **Used Function**: a) random.random(): Familiar function in Python and other machine learning applications. This method is mainly used to collect the random numbers ranging from 0 to 1. b) math.floor(): The mathematical functions used to convert the floor of the return floating-point number convert into an integer number.

Using the aforesaid code, select a random index from a string array containing all potential possibilities for a certain digit of the OTP.

Algorithm to generate OTP for 4-digit

Step 1: Import the library of random() and math()

Step 2: Generate the OTP function

Step 3: Declare a digit variable

OPT_range = “0123456789”; OTP = ““

Step 4: The password’s length can be altered

Step 5: Changing value based on corresponding range for *i* in range(4) :

Step 6: OTP += OPT_range[math.floor(random.random()*10)]

return OTP

Step 7: Driver the code execution based on main and other functions

Step 8: Start the system clock using the stop watch/ counter functions

Algorithm to generate OTP for 6-digit

Step 1: Import the library of random() and math()

Step 2: Generate the OTP function

Step 3: Declare a string variable

Step 4: String = ’0123456789abcdefghijklmnopqrstuvwxyzABCDEFGHIJKLMNOPQRSTUVWXYZ’

OTP_string = ““

Step 5: Corresponding password’s length can be altered

Step 6: Based on range the corresponding values can be change

Step 7: Driver the code execution based on main and other functions

Step 8: Start the system clock using the stopwatch/counter functions

t1_clockstart = process_time()

print(“OTP of length 6:”, generateOTP())

t1_clockstop = process_time()

print(“Elapsed time during the whole program in seconds:”,t1_clockstop-t1_clockstart)

Algorithm to generate String and Digit OTP

Step 1: Importing the packages for random suction to generate the sequence

Step 2: OTP generation using random string sequence

Step 3: string library function

Step 4: Takes random choices from the initial string OPT

Step 5: OTP considers only ascii_uppercase, ascii_lowercase, and digits

Step 6: generate_OTPpassword

Step 7: Drive the code execution based on main and other functions

if __name__ == ”__main__” :

Step 8: Start the system clock using the stopwatch/counter functions

password = rand_pass(8)

print(“OTP for alpha numeric of length 10:”,password)

t1_clockstop = process_time()

print(“Elapsed time during the whole program in seconds:”,t1_clockstop-t1_clockstart)

3.2 ENCRYPTION AND DECRYPTION OF FILES USING PYTHON

Only on entering this OTP do users get access to the requested file. Then the encrypted key is obtained from the server. Encryption is the process of decoding a communication so that it can only be read by the intended receivers.

To encrypt a file, then utilize the cryptography library employs a symmetric technique to encrypt the file. The same key is used to encrypt and decode the file in symmetric algorithms like Reverse cipher, Caesar Cipher, and ROT13 Algorithm. The cryptography package’s fernet module contains built-in functions for generating the key, using encrypt () and decrypt () methods. This function convert encrypts plain text into cipher text and decodes cipher text into plain text.

All actions will be performed using the nba.csv file.

Installation: Using the command below, you may install the cryptography library:

Generate Key to encrypt the file: In the cryptography library, encrypting the file using a cryptography algorithm is known as fernet.

Once the file is encrypted then open the file containing the key. Create and save the Fernet object in the fernet variable. Examine the original document. Put the file inside an object and encrypt it. Then, in the same file nba.csv, write the encrypted data. Now write the following code for the file encryption:

Import OTP-related packages and required module

Step 1: From cryptography.fernet import Fernet

The function used to generate the key and store:

Step 2: Fernet key generation python function

Key = Fernet.generate_key()

Step 3: Corresponding string key is stored in a file

The above code delivers the following output

J64ZHFpCWFlS9zT7y5zxuQN1Gb09y7cucne_EhuWyDM=

Algorithm for encryption of file using the key generation

Step 1: Open the file that contains the output key

Step 2: Initialize the Fernet object and its corresponding output and store it in the related variable using Fernet(key)

Step 3: Read the corresponding file

#opening the source file which is used to encrypt the corresponding text with open('OutputFile.csv', 'rb') as file:

Step 4: The process encrypts the given file and converts it into an object to save the file with open('OutputFile.csv', 'rb') as file:

Step 5: The given encrypt function writes the encrypted data to store the same file

encrypted file open using the write mode

writing method used to write the encrypted data

The Fig.2 shows the nba.csv file before and after running the aforementioned software. The Fig.3 shows the OutputFile.csv file after running the above program.

	A	B	C	D	E	F	G	H	I
1	Name	Team	Number	Position	Age	Height	Weight	College	Salary
2	Avery Bra	Boston Ce	0	PG	25	06-Feb	180	Texas	7730337
3	Jae Crowd	Boston Ce	99	SF	25	06-Jun	235	Marquette	6796117
4	John Holla	Boston Ce	30	SG	27	06-May	205	Boston University	
5	R.J. Hunte	Boston Ce	28	SG	22	06-May	185	Georgia St	1148640
6	Jonas Jere	Boston Ce	8	PF	29	06-Oct	231		5000000
7	Amir John	Boston Ce	90	PF	29	06-Sep	240		12000000
8	Jordan Mi	Boston Ce	55	PF	21	06-Aug	235	LSU	1170960
9	Kelly Olyn	Boston Ce	41	C	25	7-0	238	Gonzaga	2165160
10	Terry Rozi	Boston Ce	12	PG	22	06-Feb	190	Louisville	1824360
11	Marcus Sn	Boston Ce	36	PG	22	06-Apr	220	Oklahoma	3431040
12	Jared Sulli	Boston Ce	7	C	24	06-Sep	260	Ohio State	2569260
13	Isaiah Tho	Boston Ce	4	PG	27	05-Sep	185	Washingtc	6912869
14	Evan Turn	Boston Ce	11	SG	27	06-Jul	220	Ohio State	3425510
15	James You	Boston Ce	13	SG	20	06-Jun	215	Kentucky	1749840

Fig.2. File for uploading encryption

	A	B	C	D	E	F	G	H
1	gAAAAABf9YzFUxFGmExwMVR9RDlkFmvgJiZjMREB1FroJbr14X9Y6f5pPw1HFxu-HaLnf8ha							
2	GsINm0RDHxgkswX3AltXrJsJ-VVwcuDJ0PGBlAh2cbc3NWR_ZM5ZD8DHjC7YF_24vj4cn3um							

Fig.3. After the encryption process

Algorithm for decryption of file using the key generation

Step 1: Initialize the required objects and variables related to fernet function.

fernet = Fernet(key)

Step 2: Read the encrypted file

#Use Open mode to opening the corresponding encrypted file

with open('OutputFile.csv', 'rb') as enc_file:

Encrypted = encript_file.read()

Step 3: Decrypt the OutputFile and save it an object

decrypting OutputFile

decryptedFile = fernet.decrypt(encrypted)

Step 4: Use write mode and decrypted input data into OutputFile.csv.

Use write mode and writing decrypted information into OutputFile.csv

with open('OutputFile.csv', 'wb') as dec_file:

dec_file.write(decryptedFile)

The Fig.4 shows the OutputFile.csv file before and after running the aforesaid application to retrieve the original data. Fig.5 shows the nba.csv file after decrypting it to acquire the original data. 5

	A	B	C	D	E	F	G	H
1	gAAAAABf9YzFUxFGmExwMVR9RDlkFmvgJiZjMREB1FroJbr14X9Y6f5pPw1HFxu-HaLnf8ha							
2	GsINm0RDHxgkswX3AltXrJsJ-VVwcuDJ0PGBlAh2cbc3NWR_ZM5ZD8DHjC7YF_24vj4cn3um							

Fig.4. Before the decryption process

	A	B	C	D	E	F	G	H	I
1	Name	Team	Number	Position	Age	Height	Weight	College	Salary
2	Avery Bra	Boston Ce	0	PG	25	06-Feb	180	Texas	7730337
3	Jae Crowd	Boston Ce	99	SF	25	06-Jun	235	Marquette	6796117
4	John Holla	Boston Ce	30	SG	27	06-May	205	Boston University	
5	R.J. Hunte	Boston Ce	28	SG	22	06-May	185	Georgia St	1148640
6	Jonas Jere	Boston Ce	8	PF	29	06-Oct	231		5000000
7	Amir John	Boston Ce	90	PF	29	06-Sep	240		12000000
8	Jordan Mi	Boston Ce	55	PF	21	06-Aug	235	LSU	1170960
9	Kelly Olyn	Boston Ce	41	C	25	7-0	238	Gonzaga	2165160
10	Terry Rozi	Boston Ce	12	PG	22	06-Feb	190	Louisville	1824360
11	Marcus Sn	Boston Ce	36	PG	22	06-Apr	220	Oklahoma	3431040
12	Jared Sulli	Boston Ce	7	C	24	06-Sep	260	Ohio State	2569260
13	Isaiah Tho	Boston Ce	4	PG	27	05-Sep	185	Washingtc	6912869
14	Evan Turn	Boston Ce	11	SG	27	06-Jul	220	Ohio State	3425510
15	James You	Boston Ce	13	SG	20	06-Jun	215	Kentucky	1749840

Fig.5. After the decryption to get the original file

The proposed system provides additional security to the data on the cloud. Here proposed research work provides security for accessing the files stored on the cloud by providing OTP to the authenticated person only if the data owner accepts to share his data. The proposed system shows that the user who wants to access the file has to initially request the data owner for file access then if the request is accepted then only OTP is sent on authenticated user number. The user then has to enter the same OTP then only the access is granted or else access is rejected. The program on the cloud server then encrypts the submitted files using the symmetric algorithm and these encrypted files can be accessed by the users only if permission is granted by the data owner. Then Data owner uses the symmetric key to decrypt the ciphertext of data files. Hence in this way the research work by

providing two-way security to the data stored on the cloud by integrating the above two mechanisms.

4. RESULTS AND DISCUSSION

Following the implementation of the proposed methodology, authors have concluded that cloud security can be improved by using a model of secure authentication with OTP based on 4-digit, 6-digit, 8-digit, and 10-digit OTP lengths, along with data encryption and decryption using Reverse cipher, Caesar cipher, and ROT13. The data sent/received by the user is extremely important and must be treated with caution. Using encryption and decryption technologies, we can minimize the processing time for OTP creation and key generation.

Table.1. OTP creation using digits and time

OTP Examples	OTP Digits	Time (s)
1368	4	0.0016
Rxe1hC	6	0.0028
K7RK5Ahm	8	0.0035
2JgRMLycEN	10	0.0042

The Table.1 have shown the time taken for OTP generation for each digit number. The string with an alphanumeric of length 10 takes the time of 0.0042 seconds which is more compared to the string with an alphanumeric of length 8. The time taken for OTP generation of 4-digit number has 0.0016 seconds which is lesser than the string of 6-digit number. The overall time taken for OTP generation of 4-digit numbers consumes lesser than the OTP generation for 6, 8, and 10-digit numbers.

Table.2. Encryption and decryption using processing times in seconds

Algorithm Name	Time (s)
Reverse Cipher	0.00025
Caesar Cipher	0.00065
ROT13	0.00207

The time it takes to decrypt cipher text into plain text and encrypt plain text into cipher text using the decrypt () and encrypt () procedures are shown in Table 2 and Fig.7. The Reverse Cipher algorithm takes the time to encrypt and decrypt the file 0.00025 seconds which is lesser compared to the time taken for encrypting and decrypting the file using Ceaser cipher as 0.00065 seconds and ROT13 as 0.00207 seconds.

5. CONCLUSION

The proposed technique combines the use of biometrics, a one-time passcode, and the usual key to validate the user/device at the time of log-on to access the cloud services. Thus, this research is focused on three-level security mechanisms that assist the CSP with high authentication for accessing the data present in the cloud environment. This system may access the owner task environment by entering the correct owner password and completing the owner task. In this paper two of the most secure encryption, decryption, and OTP algorithms. In comparison to the prior technique, the two security approaches make our framework more secure. The OTP and symmetric key generation are offered as a novel approach for cloud service authentication; this solution is more secure and simple to use. Because the need for cloud computing is growing in today's world, the security of the cloud and its users is a major problem. This paper outlines the security concerns that cloud computing faces in general, as well as the mitigating strategies that have been offered to address the issues. This process successfully developed the above-mentioned system and has concluded that by utilizing this recommended method we may more effectively attain improved security in cloud computing. In the future present various comparisons using our technique and results to demonstrate the efficacy of our proposed framework.

REFERENCES

[1] Ankita Patil, Kiran Zambare, Preeti Yadav, Pankaj Wasulkar and Nisha Kimmatkar, "Integration of Encryption of File and One Time Password for Secure File Access on Cloud", *International Journal of Advances in Computer Science and Cloud Computing*, Vol. 3, No. 1, pp. 1-13, 2015.

[2] V. Mohammadi, A.M. Rahmani, A.M. Darwesh and A. Sahafi, "Trust-Based Recommendation Systems in the Internet of Things: A Systematic Literature Review", *Human-Centric Computing and Information Sciences*, Vol. 9. pp. 1-21, 2019.

[3] Y.S. Jeong and J.H. Park, "Security, Privacy, and Efficiency of Sustainable Computing for Future Smart Cities", *Journal of Information Processing Systems*, Vol. 16, pp. 1-5, 2020.

[4] J.Y. Park and E.N. Huh, "A Cost-Optimization Scheme Using Security Vulnerability Measurement for Efficient Security Enhancement", *Journal of Information Processing Systems*, Vol. 16, pp. 61-82, 2020.

[5] J. Kang, "Mobile Payment in Fintech Environment: Trends, Security Challenges, and Services", *Human-Centric Computing and Information Sciences*, Vol. 8, pp. 1-16, 2018.

[6] H.W. Kim and Y.S. Jeong, "Secure Authentication-Management Human-Centric Scheme for Trusting Personal Resource Information on Mobile Cloud Computing with Blockchain", *Human-Centric Computing and Information Sciences*, Vol. 8, pp. 1-11, 2018.

[7] D.R. Stinson and M. Paterson, "*Cryptography: Theory and Practice*", CRC Press, 2018.

[8] Hyunki Kim, Juhong Han, Chanil Park and Okyeon Yi, "Analysis of Vulnerabilities That Can Occur When Generating One-Time Password", *Applied Sciences*, Vol. 10, pp. 1-12, 2020.

[9] M. Robinson Joel, V. Ebenezer, M. Navaneethakrishnan and N. Karthik, "Encrypting and Decrypting Different Files Over Different Algorithm on Cloud Platform", *International Journal of Emerging Trends in Engineering Research*, Vol. 8, No. 4, pp. 1-5, 2020.

[10] Liangshun Wu, H. J. Cai, and Han Li, "SGX-UAM: A Secure Unified Access Management Scheme with One Time Passwords via Intel SGX", *IEEE Access*, Vol. 9, pp. 38029-38042, 2021.

[11] Karimov Madjit Malikovich, Khudoykulov Zarif Turakulovich and Arzieva Jamila Tileubayevna, "A Method

of Efficient OTP Generation using Pseudorandom Number Generators", *Proceedings of International Conference on Information Science and Communications Technologies*, pp. 1-13, 2021.

[12] Kalyanapu Srinivas and V. Janaki, "A Novel Approach for Generation of OTP'S using Images", *Procedia Computer Science*, Vol. 85, pp. 511-518, 2016.

[13] Sarita Kumari, "A Research Paper on Cryptography Encryption and Compression Techniques", *International Journal of Engineering and Computer Science*, Vol. 6, No. 4, pp. 20915-20919, 2019.

[14] Shweta Joshi and Rekha Mehra, "GUI based Approach for Data Encryption and Decryption on Matlab Platform", *International Journal of Computer Applications*, Vol. 181, No. 16, pp. 1-13, 2018.

[15] Sanjeev Kumar, Garima Karnani, Madhu Sharma Gaur and Anju Mishra, "Cloud Security using Hybrid Cryptography Algorithms", *Proceedings of International Conference on Intelligent Engineering and* Management, pp. 28-30, 2021.

[16] N.N Mosola, M.T Dlamini, J.M Blackledge, J.H.P Eloff and H.S Venter, "Chaos-based Encryption Keys and Neural Key-store for Cloud-hosted Data Confidentiality", *Proceedings of Southern Africa Conference on Telecommunication Networks and Applications*, pp. 1-13, 2017.

[17] K. Brindha and N. Jeyanthi, "Securing cloud Data using Visual Cryptography", *Proceedings of International Conference on Innovation Information in Computing Technologies*, pp. 1-5, 2015.

[18] Pronika and Tyagi, "Secure Data Storage in Cloud using Encryption Algorithm", *Proceedings of International Conference on Intelligent Communication Technologies and Virtual Mobile Networks*, pp. 1-6, 2021.

[19] Amr M. Sauber, Passent M. El-Kafrawy, Amr F. Shawish, Mohamed A. Amin and Ismail M. Hagag, "A New Secure Model for Data Protection over Cloud Computing", *Computational Intelligence and Neuroscience*, Vol. 2021, pp. 1-11, 2021.

[20] Reece B. D'Souza and D. Ruby, "Secure File Storage on Cloud using Enhanced Hybrid Cryptography", *International Research Journal of Engineering and Technology*, Vol. 8, No. 3, pp. 1-13, 2021.