Securing One Time Password (OTP) for Multi-Factor Out-of-Band Authentication through a 128-bit Blowfish Algorithm

Ariel Roy L. Reyes, Enrique D. Festijo and Ruji P. Medina

Graduate Programs, Technological Institute of the Philippines, Quezon City, Philippines

***Abstract*:** Authentication and cryptography have been used to address security issues on various online services. However, studies have shown that even the most commonly-used multi-factor out-of-band authentication mechanism is vulnerable to attacks while traditional crypto-algorithms exhibit drawbacks. In the present study, an innovative modification of the Blowfish cryptographic algorithm is introduced that capitalizes on the algorithm's strengths but supports 128-bits input block size using dynamic selection encryption method and reduction of cipher function execution through randomly determined rounds. Experimentation results on 128-bit input text revealed significant performance improvements with utmost 5.91 % in terms of avalanche effect, 38.97 % for integrity, and 41.02 % in terms of execution time. The modification provided an additional layer of security, thus, displaying higher complexity and stronger diffusion at faster execution time making it more resilient to attacks by unauthorized parties and desirable to be used for applications with multiple users respectively. This is a good contribution to the continuous developments in the field of information security particularly in cryptography and towards providing a secure OTP for multifactor out-of-band authentication.

***Keywords*:** Authentication, Cryptography, Crypto-algorithm, Blowfish, Dynamic Selection Method, OTP, SMS-based OTP**.**

1. Introduction

Internet users rely on carrying out various activities through online services that offer accessibility and convenience [1]. However, security issues on personal information have emerged and are considered as some of the most important concerns in today's connected world [2][3]. Therefore, a more secure online environment is necessary to obtain the trust and confidence of users [1].

To better enforce security in online services, authentication has been the major means of defense [1][4][5]. It is a mechanism to verify the authenticity of a legitimate personality to access data on protected systems [5]. It is enacted by multiple means among which single-factor authentication, or the use of user ID and password, has served as the traditional security mechanism. However, significant downsides of single-factor authentication have led to the increasing adoption of multi-factor authentication which is considered to offer stronger security mechanism through additional user requirement such as something the user knows and have which provides a second layer of authentication [1][6][7]. To this end, Short Message Service (SMS)-based One Time Password (OTP) remains one of the most commonly used multi-factor authentication and authorization mechanism. This method has found wide use such as in online banking, email services, social networks, transactions with financial institutions and online marketplaces, and online academic information applications [6][8][9]. But even this enhanced method has not remained attack-proof. An experimental social engineering attack against Google's SMS-based authentication was demonstrated to have a 50% success rate. The same attack showed 25% success rate using an out-of-band authentication modality. Moreover, an increase in the number of attacks using this method has been reported, with twenty-two (22) instances of such attack in China in a so-called Verification Code Forwarding Attack or VCFA [8][10].

To further strengthen security against untrusted environment, cryptography has been used to make information indecipherable [11][12][13][14] and promote a safer virtual world [15][16]. Several cryptographic algorithms or crypto-algorithms have been developed, categorized as either symmetric or asymmetric [2][14][17][18][19], of which symmetric algorithms show good performance with respect to speed [20][16] and security through strong key size [16].

Blowfish is a symmetric algorithm considered to be best in terms of security, avalanche effect, throughput, memory usage, execution time, and power consumption giving it distinct advantage over other symmetric algorithms [21]. It has found use in many applications requiring secure transmission of data such as bulk encryption, packet encryption, and Internet-based security [21][16]. It is also very suitable for multiple user applications such as multiple-factor out-of-band authentication implementations. However, since symmetric algorithms like Blowfish requires the same key for encryption and decryption, secrecy and size of the secret key are the only means of defense [2][20][22].

This paper will secure OTP for multi-factor out-of-band authentication modality through an alternative approach of OTP delivery and encryption using a 128-bit Blowfish algorithm. The primary focus of this paper is to expand the input text of Blowfish algorithm to 128-bits block size to improve its execution time for wider block size input, increase complexity by using dynamic selection encryption method, and reduce its execution of cipher function in randomly determined rounds to further improve complexity and the algorithm's overall execution time, all without compromising the security feature of the original Blowfish algorithm.. The results of this study will be a good contribution to the continuous developments in the field of information security particularly in cryptography and towards providing a secure OTP for out-of-band authentication.

2. Literature Review

SMS-based OTP is the most widely used multi-factor authentication and authorization scheme for many different applications because all it requires is a mobile phone as

additional device [6]. In an SMS-based OTP system (Figure 1), users are required to provide something they know (username, and password) and something they have (OTP verification code) before they can access secured systems, thus, effectively reducing risks against illegitimate access since both factors have to be broken or to be compromised [6].

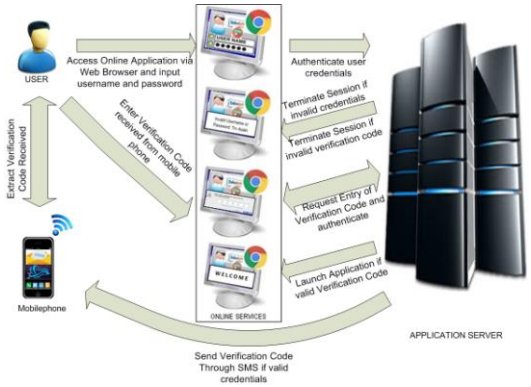

Figure 1. SMS-based OTP Operation

As shown in Figure 1, SMS-based OTP starts with a generated verification code sent via SMS to a registered mobile phone of a user if the supplied username and password are authentic. The sent verification code is normally presented as plain text to be readable for users to encode on online service provider's web application [6][9].

The vulnerabilities of SMS lies in the messaging service and the available functionalities of mobile networks that become an attractive area for attackers [23][24]. Phishing has been identified and recorded to be on the rise in many real-world instances [8]. SMS Phishing or SMiShing, is a technique comparable to Internet phishing where users are fooled by a non-genuine message that looks interesting to steal OTP issued by online service providers. This technique is normally accomplished with the help of social engineering practices and a possible malware installed on user's mobile phone [8][24]. Instances of phishing attacks have been recorded. Citizen Lab recorded events where users were deceived to use their credentials and verification codes on a fake login page [8]. Symantec also revealed cases where users were lured to forward verification codes to an attacker. A new form of phishing, called Verification Code Forwarding Attack (VCFA), was also discovered with success rate of 25% [8].

Another common threat is to eavesdrop a verification code [8]. Eavesdropping or to secretly acquire relevant information can be accomplished using Key Logger, Screen Capturing, and Shoulder surfing. Keylogger attack captures all user keystrokes and sends the logs periodically to the attacker. Often, a combination with Keylogger, Screen capturing captures both the keystrokes and visual items. Screen capturing attack can also take the screenshots of the whole screen to retrieve confidential information. Shoulder surfing, on the other hand, is a technique to disclose sensitive information by merely looking at the keyboard or the screen while users perform online transactions [24]. The foregoing threats necessitate the need to hide sensitive information such as OTP verification codes through cryptographic methods.

Crypto-algorithms such as AES, T-DES, DES, RC2, RC6, RSA, CAST-128, and Blowfish have been used depending on its suitability for specific applications and based on their strengths and weaknesses. Comparative analysis based on the level of security, average encryption time, throughput, memory usage and battery consumption revealed that Blowfish provided more security at great encryption speed [21]. Blowfish algorithm was also said to be the best-suited algorithm for applications were time and memory usage is the primary consideration as compared to DES, T-DES, AES or RSA algorithms [15]. When compared with DES and AES, Blowfish was observed to perform twice as fast as DES and four times faster than AES, in addition to consuming a smaller amount of memory and while providing great security through a strong key size [16]. Blowfish also exhibited stronger security against CAST-128 [19]. It showed good non-linear relation between the plain text and the ciphertext [25]. In general, the Blowfish algorithm can be considered superior among the crypto-algorithms [21].

Blowfish is referred to as a robust general-purpose keyed symmetric block cipher algorithm that can be used as an informal replacement for DES or IDEA. Blowfish supports 64-bit block size and a variable key length from 32 bits up to 448 bits. It is a 16-round Feistel cipher that uses large key-dependent S-boxes and requires 521 iterations to generate all essential subkeys. A single initial key in this algorithm derives 18 sub-keys [15][16][19]. Aside from not being patented, Blowfish has a free license that allows free use [15]. The Blowfish algorithm flowchart is shown in Figure 2.

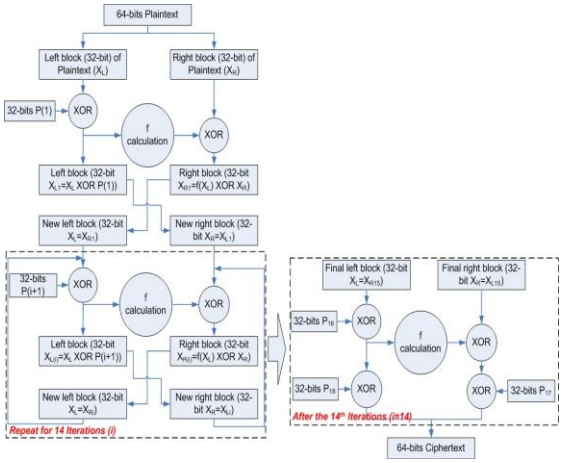

Figure 2. Blowfish Algorithm Flowchart

As shown in Figure 2, Blowfish algorithm starts by splitting the 64-bits plain text into two equal blocks. The first 32-bits block (L) and the first 32-bits P array are subjected to the bitwise XOR operator and the result is considered as input for the computation of the cipher function or function f. The said function is used to permute the data into a 32-bits block segment which is then XOR'ed with the second 32-bits block (R). After having the result of the XOR operation, the two 32-bits segments (L and R) are exchanged and the process repeats for 15 iterations. After the 15th iteration, XOR operation with the remaining P array and cipher function computation is performed to produce the ciphertext. The computation of cipher function is considered as the most complex part of this algorithm where S-boxes is utilized [25]. Besides the strengths of Blowfish algorithm in terms of performance, not being patented and licensed has made it freely available for use and modification [15]. To date, various enhancements to Blowfish algorithm have been

carried out to further improve performance and make it suitable for different intended applications.

Elliptic Curve Cryptography and Blowfish algorithm were integrated to provide authentication and confidentiality mechanism for mobile data in the cloud. A bitwise XOR operation with the generated random number and the plaintext served as the starting point in this structure in order to increase computational complexities and strengthen security. To improve Blowfish performance, the number of rounds was randomized. The design was executed and verified on different platforms such as personal computer, smartphone, emulator, and tablet [26].

Embedding Sensitive Information Transferring Technique and Blowfish Algorithm for Key Generation was used to improve image-based passwords. The objective was to provide a solution to combat different attacks on internet applications such as online guessing, shoulder surfing, phishing, and brute force. In this configuration, the watermarked image spawned from the embedded algorithm process was divided into image blocks to be encrypted using the Blowfish algorithm. Results showed a better entropy and correlation rate [27].

A modified Feistel network and a cipher function for the Blowfish algorithm was developed using the concepts of genetic algorithm and mutation. The new cipher function was called G-function derived from the main contributor of the design, the concept of genetic algorithm. The design presented new methods for the algorithm's key generation and transmission of data. Results showed improvements in the ciphertext obtained in terms of complexity and security [28].

A new method for reducing time complexity in S-boxes and P-arrays generation for the Blowfish algorithm was introduced. It was realized by replacing the 521 encryptions required in the Blowfish algorithm to generate the key-dependent P-arrays and S-boxes with the linear feedback shift registers (LFSR). The modification was designed for speech encryption process and the result showed the same level of security with the original Blowfish algorithm but with less computational overhead for key generation [29].

3. Proposed Modified Blowfish Algorithm Framework

The framework for the proposed algorithm, shown in Figure 3, will be the first version of the Blowfish algorithm designed to support 128-bits block size that implements dynamic selection encryption method and reduction of the execution of cipher function in a randomly determined round to improve complexity and the algorithm's execution time.

As shown in Figure 3, the framework of the proposed modified blowfish algorithm has five components: Block Selector, Random Number Generator, Crypto-algorithm Processor, Inverted Crypto-algorithm Processor, and Blocks Merger. The Block Selector divides the 128-bits verification code into two equal parts of 64-bits block size. A random selection as to which block goes to the Crypto-algorithm or the Inverted Crypto-algorithm Processors will be determined by the Random Number Generator that generates 8 numbers from 1 to 16 and its sum will define which among the processors the blocks has to go through. If the sum is odd, the 64-bits left side block undergoes encryption with the Crypto-algorithm Processor while the other 64-bits block will be encrypted using the Inverted Crypto-algorithm Processor, thus making the encryption selective to improve complexity and increase security. Both the Crypto-algorithm and the Inverted Crypto-algorithm processors will behave the same as the traditional Blowfish algorithm except for the execution of the cipher function. Shown in Figure 4 is the detailed flowchart of the proposed modified Blowfish algorithm and how the two crypto-algorithm processors operate.

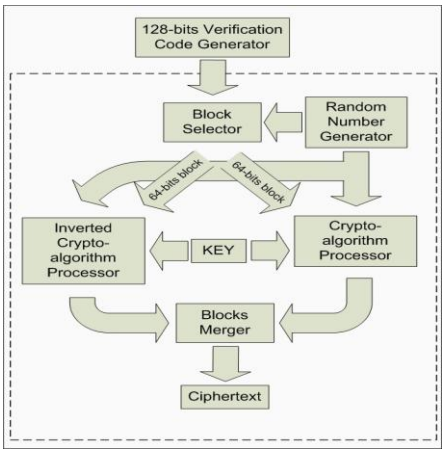

Figure 3. Framework for the Proposed Modified Blowfish Algorithm

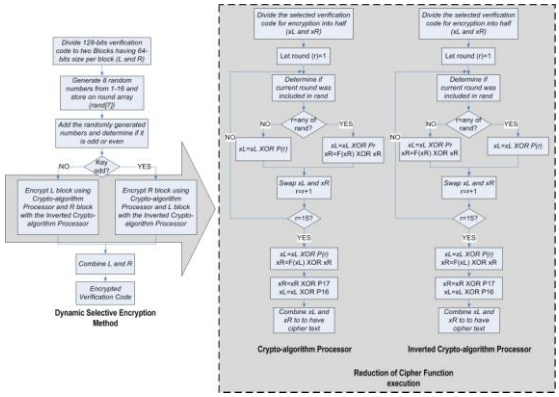

Figure 4. Proposed Modified Blowfish Algorithm Flowchart

Both processors will execute the cipher function only on selected rounds demarcated by the random number generator (Figure 4). The 8 numbers generated by the random number generator specify the rounds on which the cipher function should and should not be executed, thus, delimits the execution of the said function to only 8 rounds. For the Crypto-algorithm Processor, cipher function executes on rounds included in the randomly generated numbers, whereas, in the Inverted Crypto-algorithm, cipher function executes on rounds not present in the array of randomly generated numbers. This method will improve the execution time and the security of the algorithm through reduction of the execution of cipher function into half and through complexity and confusion respectively. To produce the ciphertext, the two outputs of both processors will be combined by Blocks Merger. The output of the Blocks Merger is the encrypted 128-bits verification code.

4. Results and Discussions

Both the original Blowfish algorithm and the proposed modified Blowfish algorithm were implemented and tested in a Pentium, Dual Core-powered CPU with 4GB of memory

and running on Windows 10 64-bit operating system. XAMPP v3.2.1 was utilized to provide a web server solution stack for experimentation purposes. The original Blowfish algorithm written by Matt Harris in PHP [30] anchored directly from Schneier's C code example on his website was used as the starting point to implement the original Blowfish algorithm [31]. An additional class, however, was created to process actual inputs and provides evaluation results for analysis. To verify the validity of the implementation, it was tested using the test vectors provided by Eric Young also posted on Schneier's website [31].

For consistency of results in running the tests and achieve a fair basis for comparison, the same sets of input data were used all throughout the experimentation. Ten (10) sets of 128-bit text and a 64-bit key were used as inputs for encryption. For the modified implementation, two sets of random numbers were tested with first set including 0, 1, 3, 4, 6, 7, 12, 15 as random numbers and the other set having 1, 2, 3, 5, 6, 9, 12, 15 whose sum are even and odd numbers respectively. Both implementations were tested using the same key, the same input text and under ECB mode. The avalanche effect, integrity check, and execution time were taken and recorded in every input sets for performance evaluation and analysis.

4.1 Avalanche Effect Experimentation Results

Avalanche effect is an important property of a cryptographic algorithm that depicts its strength in terms of a property called diffusion [15]. It is also called as the diffusion test [32] that refers to the substantial change in output (ciphertext) when a slight change in input key is applied [15][19][25][32]. The avalanche effect is computed according to Equation 1 [15].

$$\text{Avalanche Effect} = (\text{Hamming Distance} \div \text{Total Bits Length}) \quad (1)$$

where Hamming distance is determined by taking the dissimilarity between the ciphertext produced before and after the slight change in the key was applied [15]. In this study, the key's 9th bit was flipped and the difference between the two ciphertexts was recorded using PHP's array_diff_assoc instruction [33]. Shown in Table 1 is the summary of the experimentation results on the avalanche effect using ECB mode.

Table 1. Experimentation Results for Avalanche Effect

		Average Changed in Ciphertext	
		Number of Bits Changed (Hamming Distance)	Changed in Percentage (Avalanche Effect)
Original Blowfish		60.90	47.58 %
Modified Blowfish	Even Random Number	64.49	50.39 %
	Odd Random Number	61.33	47.92 %

The modified version caused an increase in the number of bits changed as the 9th bit of the key was flipped (Table 1) leading to increased avalanche effect (Figure 5). The least improvement observed was when an odd random number was used while a higher improvement for the even random number.

As shown in Figure 5, a 0.34 % difference as compared to the original Blowfish algorithm for the odd random number while 2.81 % difference was recorded for the even random number. Results show that the security of the original Blowfish algorithm in terms of avalanche effect was not compromised, but instead, were improved to 0.71% for the odd random number and 5.91% for an even random number when the proposed modifications were implemented. This signifies that the proposed modification offers stronger diffusion property compared to the original Blowfish algorithm making it more difficult to perform analysis on ciphertext when mounting an attack.

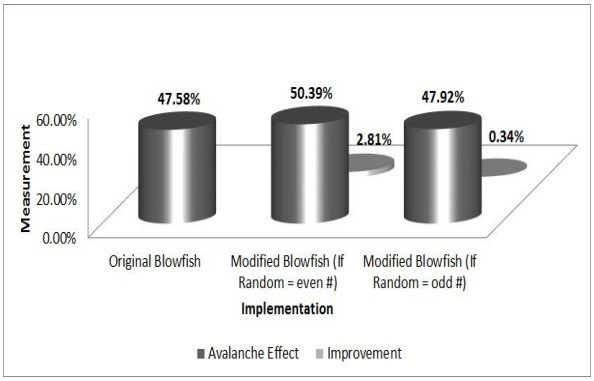

Figure 5. Experimentation Result in Avalanche Effect

4.2 Integrity Check Experimentation Results

Integrity check is the magnitude of changes in the plain text whenever there is a single bit change in the ciphertext [19]. In this study, integrity was determined in the same manner as avalanche effect, except that instead of computing for dissimilarity in the ciphertext, dissimilarity in the plain text was determined whenever the 9th bit of the ciphertext was flipped. The summary of the experimentations results for integrity check is shown in Table 2 and Figure 6.

Table 2. Experimentation Result in Integrity Check

		Average Changed in Plain text	
		Number of Bits Changed (Hamming Distance)	Changed in Percentage (Integrity Check)
Original Blowfish		32.56	25.43 %
Modified Blowfish	Even Random Number	35.64	27.84 %
	Odd Random Number	45.23	35.34 %

The number of bits changed in the plain text increased in both cases, either when the random number used was odd or even number as compared to the original Blowfish algorithm (Table 2). As presented in Figure 6, improvements were observed in the odd random number at 9.90 % performance difference and 2.41 % in even random number. This also validates that the security advantage in terms of the integrity of the original Blowfish algorithm was not compromised but was even improved to 38.97 % for the odd random number and 9.48 % for an odd random number when the proposed modifications were implemented, thus, making the diffusion stronger.

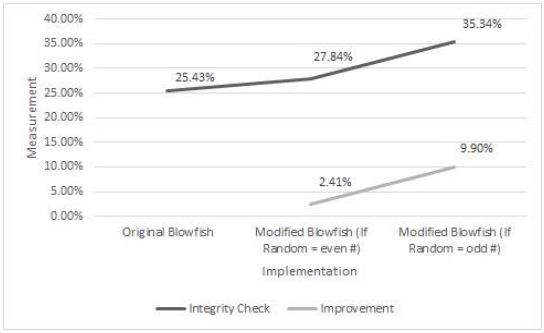

Figure 6. Experimentation Result in Integrity Check

4.3 Execution Time Experimentation Results

Execution time refers to the total time expended in converting the plain text to the ciphertext (encryption time) and the time necessary to recover the plaintext from the ciphertext (decryption time). This impacts the performance of the system and determines how fast and responsive it is [15]. In this study, the same sets of inputs were executed repeatedly to ensure fair results and the execution time was determined using Equation 2.

$$\text{Execution Time} = \text{Average Encryption Time} + \text{Average Decryption Time} \quad (2)$$

Presented in Table 3 are the average encryption and decryption values in milliseconds.

Table 3. Experimentation Result in Execution Time

	Original Blowfish		Modified Blowfish			
			Encryption		Decryption	
	Encrypt	Decrypt	Even Rand	Odd Rand	Even Rand	Odd Rand
Time	0.24	0.15	0.12	0.14	0.11	0.13
Execution Time	**Original Blowfish**		0.39			
	Modified Blowfish		Even Rand		0.23	
			Odd Rand		0.26	

It can be noted from Table 3 that the modified Blowfish algorithm consistently required less time to encrypt a plain text and recover it from a ciphertext as compared to unmodified version regardless whether the random number was odd or even. The execution time differed by 0.13 milliseconds for the random odd number and 0.16 milliseconds for a random even number as compared to the original Blowfish algorithm (Figure 7). Thus, significant improvements in execution time with the modified version was observed at 33.33 % improvement for the odd random number and 41.02 % for an even random number as compared to the original Blowfish algorithm.

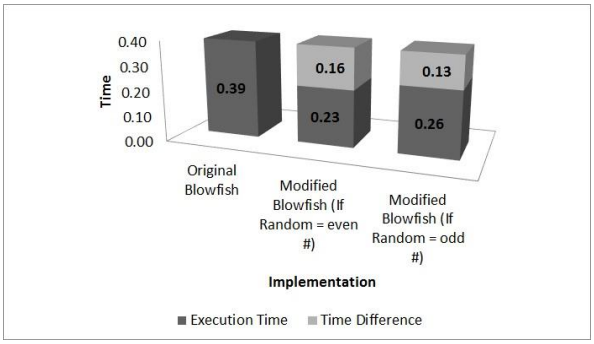

Figure 7. Experimentation Result in Execution Time

5. Conclusions

In this study, the Blowfish algorithm was implemented by extending its supported block size from 64-bits to 128-bits. Along with this was to use a dynamic selection encryption method and introduce randomly determined rounds for cipher function execution. The results and analysis conducted revealed that the implementation of dynamic selection encryption and dynamic determination of rounds for the execution of cipher function introduced additional complexity and confusion for an adversary. Relevant information that includes the key used for encryption and the details on which particular block of the input text needs to be subjected for a modified Blowfish encryption process, as well as the specific rounds where the cipher function should and should not be executed, would be necessary before gaining access on the information. This had added an extra layer of security and makes it more complicated and difficult to acquire the information even if the encryption key is compromised. Results also revealed that the proposed modifications had not compromised the security feature of the original Blowfish algorithm, instead, had increased the security in terms of avalanche effect and integrity as well as reduced the execution time. These results lead to a conclusion that the proposed modification for the Blowfish algorithm had improved the degree of complexity and diffusion, thus making it more difficult and complex for an unauthorized individual to decipher the information, and caused significant improvement in the execution time making it more suitable for applications with multiple users respectively.

References

[1] D. Dasgupta, A. Roy, and A. Nag, "Toward the design of adaptive selection strategies for multi-factor authentication," Elsevier Journal of Computers & Security, vol. 63, pp. 85–116, Nov. 2016.

[2] Swathi S V, Lahari P M, and Bindu A Thomas, "Encryption Algorithms: A Survey," International Journal of Advanced Research in Computer Science & Technology, vol. 4, no. 2, 2016.

[3] A. Joshi, M. Wazid, and R. H. Goudar, "An Efficient Cryptographic Scheme for Text Message Protection Against Brute Force and Cryptanalytic Attacks," Procedia Computer Science, International Conference on Computer, Communication and Convergence (ICCC 2015), vol. 48, no. Supplement C, pp. 360–366, Jan. 2015.

[4] A. K. Nag, A. Roy, and D. Dasgupta, "An Adaptive Approach Towards the Selection of Multi-Factor Authentication," 2015 IEEE Symposium Series on Computational Intelligence, pp. 463–472, 2015.

[5] R. Thandeeswaran and M. A. S. Durai, "DPCA: Dual Phase Cloud Infrastructure Authentication," International Journal of Communication Networks and Information Security (IJCNIS), vol. 8, no. 3, Dec. 2016.

[6] Mohsen Gerami and Satar Ghiasvand, "One-Time Passwords via SMS," Bulletin de la Société Royale des Sciences de Liège, vol. 85, pp. 106–113, 2016.

[7] Ms. E.Kalaikavitha M.C.A., M.Phil., Mrs. Juliana gnanaselvi M.Sc., and M.Phil., Ph.D., "Secure Login Using Encrypted One Time Password (OTP) and Mobile Based Login Methodology," International Journal Of Engineering And Science, vol. 2, no. 10, pp. 14–17, 2013.

[8] H. Siadati, T. Nguyen, P. Gupta, M. Jakobsson, and N. Memon, "Mind your SMSes: Mitigating social engineering in

second-factor authentication," Elsevier Journal of Computers & Security, vol. 65, pp. 14–28, Mar. 2017.

[9] E. Sediyono, K. I. Santoso, and Suhartono, "Secure login by using One-time Password authentication based on MD5 Hash encrypted SMS," 2013 International Conference on Advances in Computing, Communications and Informatics (ICACCI), pp. 1604–1608, 2013.

[10] M. Balduzzi, P. Gupta, L. Gu, D. Gao, and M. Ahamad, "MobiPot: Understanding Mobile Telephony Threats with Honey cards," 11th ACM on Asia Conference on Computer and Communications Security, New York, NY, USA, pp. 723–734, 2016.

[11] S. K. Pujari, G. Bhattacharjee, and S. Bhoi, "A Hybridized Model for Image Encryption through Genetic Algorithm and DNA Sequence," Procedia Computer Science, The 6th International Conference on Smart Computing and Communications, vol. 125, pp. 165–171, Jan. 2018.

[12] S. Deng, D. Yue, A. Zhou, X. Fu, L. Yang, and Y. Xue, "Distributed content filtering algorithm based on data label and policy expression in active distribution networks," Elsevier Journal of Neurocomputing, vol. 270, pp. 159–169, Dec. 2017.

[13] K. Dhiman and S. S. Kasana, "Extended visual cryptography techniques for true color images," Elsevier Journal of Computers & Electrical Engineering, Oct. 2017.

[14] M. I. ElSharkawy, "Integrating and Securing Video, Audio and Text Using Quaternion Fourier Transform," International Journal of Communication Networks and Information Security (IJCNIS), vol. 9, no. 3, Dec. 2017.

[15] P. Patil, P. Narayankar, Narayan D.G., and Meena S.M., "A Comprehensive Evaluation of Cryptographic Algorithms: DES, 3DES, AES, RSA and Blowfish," Procedia Computer Science, 1st International Conference on Information Security & Privacy, Nagpur, India, vol. 78, pp. 617–624, Jan. 2016.

[16] A. Ramesh and A. Suruliandi, "Performance analysis of encryption algorithms for Information Security," International Conference on Circuits, Power and Computing Technologies (ICCPCT), pp. 840–844, 2013.

[17] Ü. Çavuşoğlu, A. Akgül, A. Zengin, and I. Pehlivan, "The design and implementation of hybrid RSA algorithm using a novel chaos-based RNG," Elsevier Journal of Chaos, Solitons & Fractals, vol. 104, pp. 655–667, Nov. 2017.

[18] M. Kumar, S. Kumar, R. Budhiraja, M. K. Das, and S. Singh, "A cryptographic model based on the logistic map and a 3-D matrix," Journal of Information Security and Applications, vol. 32, pp. 47–58, Feb. 2017.

[19] Youssouf Mahamat koukou, Siti Hajar Othman, Maheyzah MD Siraj, and Herve Nkiama, "Comparative Study Of AES, Blowfish, CAST-128 And DES Encryption Algorithm," IOSR Journal of Engineering (IOSRJEN), vol. 6, no. 6, pp. 1–7, 2016.

[20] A. Bhardwaj, G. V. B. Subrahmanyam, V. Avasthi, and H. Sastry, "Security Algorithms for Cloud Computing," Procedia Computer Science, International Conference on Computational Modelling and Security (CMS 2016), vol. 85, pp. 535–542, Jan. 2016.

[21] M. Suresh and M. Neema, "Hardware Implementation of Blowfish Algorithm for the Secure Data Transmission in Internet of Things," Procedia Technology, 1st Global Colloquium on Recent Advancements and Effectual Researches in Engineering, Science and Technology - RAEREST, vol. 25, no. Supplement C, pp. 248–255, Jan. 2016.

[22] S. Chandra, B. Mandal, S. S. Alam, and S. Bhattacharyya, "Content-Based Double Encryption Algorithm Using Symmetric Key Cryptography," Procedia Computer Science, 3rd International Conference on Recent Trends in Computing (ICRTC-2015), vol. 57, no. Supplement C, pp. 1228–1234, Jan. 2015.

[23] M. Thomas and V. Panchami, "An encryption protocol for end-to-end secure transmission of SMS," International Conference on Circuits, Power and Computing Technologies [ICCPCT-2015], pp. 1–6, 2015.

[24] A.S. Chaudhari, "Security Analysis of SMS and Related Technologies," Research Master Thesis, Dept. of Mathematics and Computer Science, Eindhoven University of Technology, 2015.

[25] A. Alabaichi, F. Ahmad, and R. Mahmod, "Security analysis of blowfish algorithm," Second International Conference on Informatics Applications (ICIA), pp. 12–18, 2013.

[26] P. Patel, R. Patel, and N. Patel, "Integrated ECC and Blowfish for Smartphone Security," Procedia Computer Science, 1st International Conference on Information Security & Privacy, vol. 78, pp. 210–216, Jan. 2016.

[27] K. L. Prasad, P. Anusha, G. Jyothi, and K. Dileepkumar, "Design and Analysis of Secure and Efficient Image with Embedded Sensitive Information Transferring Technique using Blowfish Algorithm," i-Manager's Journal on Information Technology; Nagercoil, vol. 5, no. 3, pp. 1–8, Aug. 2016.

[28] S. G. Saravana Kumar and Dr.A.Shanmugam, "Modified F – Function for Feistel Network in Blowfish Algorithm," International Journal of Engineering and Innovative Technology (IJEIT), vol. 4, no. 4, pp. 229–232, 2014.

[29] A. A. A. El-Sadek, T. A. El-Garf, and M. M. Fouad, "Speech encryption applying a modified Blowfish algorithm," International Conference on Engineering and Technology (ICET), pp. 1–6, 2014.

[30] Harris, M. (2018). themattharris (Matt Harris). [online] GitHub. Available at: https://github.com/themattharris [Accessed 2 Feb. 2018].

[31] Schneier.com. (2018). Schneier on Security: The Blowfish Encryption Algorithm. [online] Available at: https://www.schneier.com/academic/blowfish/ [Accessed 2 Feb. 2018].

[32] A. Amiruddin, A. A. P. Ratna, and R. F. Sari, "New Key Generation and Encryption Algorithms for Privacy Preservation in Mobile Ad Hoc Networks," International Journal of Communication Networks and Information Security (IJCNIS), vol. 9, no. 3, Dec. 2017.

[33] Php.net. (2018). PHP: array_diff_assoc - Manual. [online] Available at: http://php.net/manual/en/function.array-diff-assoc.php [Accessed 2 Feb. 2018].