Advances and Applications in Mathematical Sciences
Volume 20, Issue 12, 2021, Pages 3157-3167

DNA BASED PATTERN REPRESENTATION FOR MOBILE SECURITY

PADMA BHOGARAJU[1], RAJKUMAR GVS[2], PREETHI GANDRETI[3] and ANITHA PARABATHINA[4]

[1,3,4]Gayatri Vidya Parishad College
for Degree and PG Courses (A)
Visakhapatnam-45, India
E-mail: padma.bhogaraju@gmail.com
preethi@gvpcdpgc.edu.in
anitha501@gvpcdpgc.edu.in

[2]GITAM Deemed to be University
Visakhapatnam-45, India
E-mail: gganapav@gitam.edu

Abstract

Mobile security has turn out to be very important in mobile computing. People are employing their smart phones as messaging tools, and means of regulating and scheduling their private life. The Unlock Pattern is a graphical password scheme widely used for Android to authenticate the user. The SHA-1 hash value of pattern password is kept in a key file, which if compromised; the user can forecast the password using pre-computation attacks such as rainbow table attacks, and dictionary attacks. Android latest versions use TEE (Trusted Execution Environment) but they need additional hardware support. To deal with this problem, a new enhancement to Secure Hash Algorithm (SHA-1) using Deoxyribonucleic acid (DNA) technology is proposed in this paper. This proposed scheme generates a different hash, where it becomes hard to guess password for the cryptanalyst. As the grid is dynamically generated based on the mobile id, Gmail-ID, this scheme is resistant to SHA-1 dictionary and rainbow table attacks on pattern passwords without employing additional hardware.

I. Introduction

Android is the most admired OS for modern mobile computers. These devices support diverse types of screen locks like gesture lock, facial lock, like

2010 Mathematics Subject Classification: 68.
Keywords: Mobile security, SHA-1, smart phone, dictionary attacks, Android, Unlock Pattern.
Received November 20, 2020; Accepted December 20, 2020

swipe lock, PIN lock, pattern lock etc [1]. The Android Unlock Pattern scheme is a graphical password where the user is offered a 3×3 grid and the password of a user is a picture on that grid, which is a chain of lines connecting the dots. During enrolment, a user has to choose a pattern [7] and to authenticate him, he has to draw it on the screen. Now the question remains how secure this pattern based graphical passwords in practice in comparison to other authentication schemes. After user chooses the pattern password, the Android device hashes it with SHA-1 hash algorithm and saves this hash into the password file. When the user tries to access the device again, the system compares the stored SHA-1 message digest with newly generated hash to decide to give admittance to the device or not. The major problem with Android pattern Lock Authentication systems is that it stores pattern lock data as an unsalted hash value. If we pick a pattern 12345, this pattern is kept in the root folder as a 20-byte SHA1 hash. So the SHA-1 hash for 12345 is "8cb2237d0679ca88db6464eac60da96345513964", which is stored in a file called "gesture. key" in "/data/system" folder in the internal memory of the Android device. Statistically, it is not a very big deal to guess the password having all combinations between 1234 and 987654321 using SHA-1 dictionary. The hackers may guess the password using rainbow tables to attack the device passively. A salted hash has a benefit that even the hash is cracked, hackers cannot get the password. But Android lower pattern versions are not salted hashes. Therefore, it is possible to hack SHA-1 hash [13] in no time and gain the original pattern using android "gesture key" file and dictionaries.

In the brute-force attack, existing software is employed to produce a huge number of successive guesses against the data to be broken. Brute force attacks might happen either online or offline. An offline attack needs work from the assailant only with no communication with the system or server under attack. An online attack needs to work with the system which is being attacked including communication. Salted hash [9] has a benefit that when the hash is broken, you cannot get the password. But the newest versions of Android authentication systems used salted hashes generated by Scrypt and HMAC algorithms[10] and are somehow strong against dictionaries and rainbow tables but passwords are executed in a TEE which needs a huge hardware support and passwords are stored in a protected environment

called a 'gatekeeper mechanism' which seems to be very complex. Figure 1. shows how hackers use various forensic tools such as Andriller to gain the password.

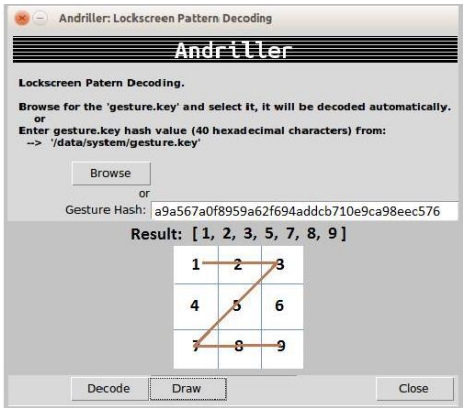

Figure 1. Andriller Tool.

A rainbow table is built by an attacker. By using a salt, an attacker will not be capable of building a rainbow table for a particular algorithm to exercise an attack. Salt is not invented to be undisclosed, you store it beside the password in the database, which is equal to hash (salt+password). In this paper we proposed a new pattern password hashing scheme that dynamically generates the grid based on DNA cryptography and Device-ID and Gmail-ID to avoid dictionary attacks [15] and to improve brute-force security. The most excellent technique to safeguard passwords is to make use of salts in password hashing. If user's passwords are is hashed by means of different salts [4] each time he logs in, the reverse lookup tables and dictionaries will not work either. But salted hashes [5] are susceptible to brute force attacks if they are not protected well. So we need to enhance the SHA-1[3] pattern hash mechanism in a way that it becomes resistant to pre-computations attacks such as Dictionary and brute-force attacks. This paper therefore investigates a solution for pattern authentication using DNA sequence where there is an intermediate hash generated in this proposed methodology which does not need an additional hardware support and becomes resistant to pre-computation attacks.

II. DNA Cryptography: A Literature Review

Nowadays, biological techniques became more prevalent, as they are

applied to many kinds of applications, biochemistry, authentication protocols, and cryptography. Amino acids are very essential to life, and they are functional with regard to in body metabolism. One major function is to serve as the building blocks of the proteins, which can be stated as a linear chain of amino acids. Amino acids can be linked simultaneously in different sequences to figure a great range of proteins. Twenty-two amino acids are apparently incorporated into polypeptides and they are called as typical amino-acids. Transcription phase in cellular process produces messenger RNA (mRNA), which is a molecular copy of one or more genes represented by alphabets A-C-G-U (uracil). The conversion of mRNA template alters genetic information based on nucleotides, to a functional protein. The sequence of proteins contains twenty happening amino acids; so it may be thought that the protein alphabet contains twenty symbolic letters. Each and every amino acid is illustrated by a three-nucleotide sequence named as the triplet codon.

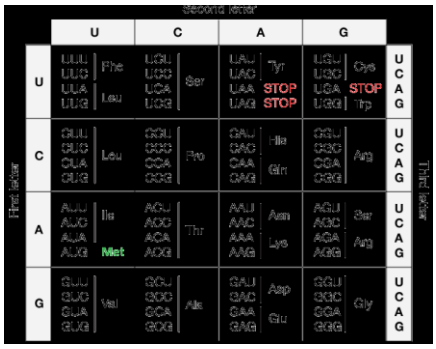

Figure 2. Genetic Code.

There are three-nucleotide code notations, a total of $64\ (4 \times 4 \times 4)$ possible combinations; as a result, a given amino acid is coded by more than one nucleotide triplet which is shown in Figure 2. Encryption of secret data in peptide sequence or amino-acid sequence now turns out to be an important and interesting research topic. This research presents a simple, secure and reversible encryption mechanism that converts the message into an amino-acid protein sequence to offer security of pattern passwords. The amplified use of computer and information technology in cyber bio-security and possibility of manipulating DNA strands has explored elevated risk in purposeful destruction of production of hazardous biological materials and biological equipment's. DNA cryptography is a budding technology. DNA

particles are having the capability in storing, processing and transmitting information, which motivates the initiative of DNA cryptography. It supports the idea of DNA technology and the use of four bases i.e. Adenine -A, Guanine-G, Cytosine-C and Thymine-T for performing computation. Major benefit of DNA technology is the gigantic parallelism of DNA molecules. Many researchers worked on this across the globe for enhancing the active DNA cryptography methodologies and to recommend innovative and new approaches in this domain. In forthcoming years DNA computers will be commercially available, and it will capture modern silicon based technology.

III. Proposed Methodology

In this paper, we are employing a hash based on user identities. The hash based on identity will certainly serves as a one-way hash and the system is secure enough to withstand to the rainbow table and dictionary attacks as the pattern keys are generated dynamically. The user even will not be able to set the desired password if proposed method is exercised because he does not own the dictionary of the passwords generated by this new technique and the reverse hash function generates a different password message, so the system is not vulnerable to rainbow tables. This research employs the DNA cryptography to represent the pattern using Amino Acid sequence. It exploits the concepts of Amino acids which are the building blocks for proteins. The hash generated using amino-acid sequence is generated depending on the user unique identities such as Gmail-ID and Device-ID.

Algorithm (DNA-SHA-1):

1. Concatenate the Gmail account with Device-ID.

2. Represent each character to its equivalent ASCII value.

3. Convert the ASCII values to Binary form.

4. Translate this binary sequence into nucleotides. (A, C, U, G).

5. Translating each nucleotide triplet into an amino acid or a termination signal in a protein.

6. Now select distinct 9 nucleotides to represent the pattern.

7. Now find the SHA-1 hash of this amino-acid pattern sequence in the

device directory.

8. Whenever user logins, the system generates the amino-acid intermediate pattern hash of his selected pattern.

9. Find the SHA-1 hash of this amino-acid pattern.

10. Compare the two hashes, if matched user gets access to the device.

Illustration with an Example:

Concatenate Gmail-ID and Device-ID. We get a string 'myname@gmail.com20013fea6bcc820c'. Represent each character with equivalent ASCII value and then convert to binary code as shown in the table 1. below.

Table1. Characters encoded to binary.

m-109-1101101	y-121-1111001	n-110-1101110	a-97-1100001	m-109-1101101	e-101-1100101	@-64-1000000	g-103-1100111
m-109-1101101	a-97-1100001	i-105-1101001	l-108-1101100	.- 46-101110	c-99-1100011	o-111-1101111	m-109-1101101
2-50-110010	0-48-110000	0-48-110000	1-49-110001	3-51-110011	f-102-1100110	e-101-1100101	a-97-1100001
6-54-110110	b-98-110010	c-99-1100011	c-99-1100011	8-56-111000	2-50-110010	0-48-110000	c-99-1100011

Now concatenate all these binary values, we get a string "1101101110000111010011101100101110110001111011111101101110010110000011000011000111001111001101100101100001110110110010110001111000111100001111100011001011000011001100011". We convert this string to nucleotide sequence using the following of "A-00, U-01, G-10 and C-11". We get a nucleotide sequence now as, "CUG CCG UCU CUG AUC UGC GUU GAA UGU CCU GCG AUC UAC GCA GCG CAU CGC CCU GCG UUG AUG AUG ACG UCG UGC AGC GAU CUG CAG CAU". Now convert the above string to amino-acid sequence by taking triplets and using genetic code in figure 3, we get an amino-acid sequence as:LPSLICVECPAIYAAHRPALMMTSCSDLQH".

Step 3. Now select the first 9 distinct amino-acids from the above string, to represent the pattern. (We can follow any alternate criteria) We select the amino-acids {L, P, S, I, C, V, E, A, Y}. Now out pattern is represented by the above sequence as given in figure 3 below:

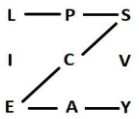

Figure 3. Pattern represented by amino-acid sequence.

If we select the pattern 1-2-3-5-7-8-9, algorithm takes input internally as "LPSCEAY". But this representation is special for different users except few collisions. When we find SHA-1 hash [11] of the above input we get the 64-bit hash, i.e. "**0f0ba14af7b20ebbbeb5f128476563176e931899**". This hash is now stored in the root folder to authenticate the user and cannot be predicted by attackers using dictionaries and rainbow tables.

IV. Brute-Force Security Analysis

In this paper, a new approach is advised for pattern passwords to avoid brute-forcing and dictionary attacks. Here hash is dynamically generated and make the pattern authentication system stronger to withstand to the pre-computation attacks. The proposed scheme never allows user to change the hash for a desired password because he does not know how a hash value is generated for a particular pattern. The representation of the grid is different for each device and it is impossible to discover a universal dictionary and rainbow table to crack the password. In this research, we are presenting a new enhanced SHA-1 pattern password scheme to prevent pre-computation attacks such as dictionaries and brute-forcing. Hackers can pull out the hash value from the device root folder and may try offline brute-force technique to speculate the pattern. But here as the hash is generated using DNA cryptography, the brute-force search space gets increased. Table 3 shows how security is improved with proposed system w.r.to brute-forcing.

Table 2. Brute-force Analysis.

Authentication Scheme used	Brute Search-Space
SHA-1	3,89,112
DNA-SHA-1	$3,\ 89,\ 112 * 22C_9^*9!$

V. Performance Evaluation

The performance of any hashing algorithm can be noticed in terms of the number of collisions [2] and avalanche effect. There are around 4 lacks pattern combinations for a 3×3 grid and to test the collisions is not feasible, we observed the performance of the proposed scheme, which is an enhanced salted SHA-1 hashing mechanism against the existing SHA-1 scheme with respect to the avalanche effect. The Avalanche Effect [12] is fulfilled, if the number of bits in the hash value changes significantly for a single-bit change in the input in the message. We observed the avalanche effect by gathering random input patterns of 4-point, 5-point, 6-point, 7-point and 8-pointpatterns, correspondingly, where the inputs differ in one single bit. We have plotted the graphs for all the above combinations of patterns to examine the Avalanche criterion of the proposed system. Figure 4 and table 2 show the avalanche effect exhibited by SHA-1 and DNA-SHA-1 mechanisms.

Table 3. SACs observed.

SAC	4-dot	5-dot	6-dot	7-dot	8-dot	AVG
SHA-1	79.88	79.233	79.5	79.4	82.6	80.12
DNA-SHA-1	80.36	81.767	79.2	81.8	77	80.03

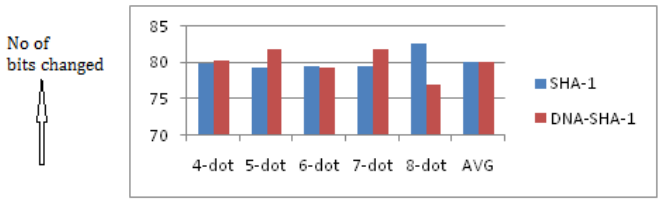

Figure 4. SAC of SHA-1 and DNA-SHA-1.

According to the experimental results, the proposed scheme shows strict avalanche effect. There is a slight variation of SAC between DNA-SHA-1 hashing scheme compared to the existing SHA-1 scheme. Time complexity becomes an important matter when the scale of an application increases. Time complexity analysis helps us comprehend and improve the effectiveness of our code. We always want efficient algorithms. We have derived the time

complexity of the salted hash method and unsalted hash method to observe the trade-offs with respect to the execution time by experimenting with random pattern hashes. These observations are taken using a java application running on Intel core i3 processor with 4GB RAM and 1.6 GHz speed Nowadays, biological techniques became more prevalent, as they are applied to many kinds of applications, biochemistry, authentication protocols [6] and cryptograph respective input pattern sizes. Figure5. And table 3. show the graphical representation of the execution times in milliseconds for generating the hashes using the two unsalted hashing schemes SHA-1 and DNA-SHA-1.

Table 4. Time Complexities.

CPU-Times in millisecs	4-dot	5-dot	6-dot	7-dot	8-dot
SHA-1	9.4	10.3	10.3	10.3	10.3
DNA-SHA-1	17	17	16	16	16.75

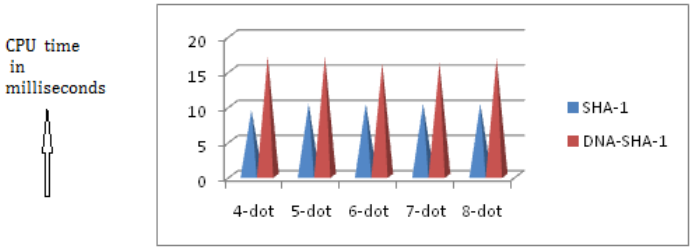

Figure 5. Time complexities of SHA-1 and DNA-SHA-1.

VI. Conclusions

Cell phone plays a crucial role in our professional and private lives. Our sensitive information should be protected from various mobile security threats [8]. The hackers gain access to the sensitive data stored in the device by predicting authentication passwords using certain attacks such as dictionaries. The main objective of this paper was, therefore, to find a more protected way to store passwords. The proposed method generates a hash based on DNA amino-acid sequence. By using our proposed technique, even if an adversary gets grip of the hash value, it is very hard that the password

will be compromised. This is because the way the DNA hash is generated is unknown. The analysis of the proposed method was done in two ways. The first was the security analysis with respect to the Avalanche Effect. It was found that the proposed scheme exhibits SAC. The time complexities of both the schemes were monitored, and it was concluded that the proposed scheme exhibits a small increase in the execution times as it is an extension of the original SHA-1 hash scheme [14]. The proposed scheme offers a secured hash generation scheme using amino-acid protein sequence to produce a SHA-1 message digest value that can withstand pre-computation attacks.

References

[1] Adarsh Singh, et al., Implementation of Color based Android Shuffling Pattern Lock. Int. J. Comp. Sci. Mobile Comput. (IJCSMC) 5(3) (2016), 357-362.

[2] A. V. Adinetz and E. A. Grechnikov, Building a Collision for 75-Round Reduced SHA-1 Using GPU Clusters. Lecture Notes in Computer Science, 7484 (2012), Springer, Berlin, Heidelberg.

[3] Anak Agung Putri Ratna, et al., Analysis and comparison of MD5 and SHA-1 algorithm implementation in Simple-O authentication based security system 99-104. Doi:10.1109/QiR.2013.6632545. 2013.

[4] S. Boonkrong and Chaowalit Somboonpattanakit, Dynamic Salt Generation and Placement for Secure Password Storing, IAENG International Journal of Computer Science 43(1) (2016), 27-36.

[5] E. M. W. R M. Chowdhury, S. Rahman, A. B. M. A. A. Islam and M. S. Rahman, Salty Secret: Let us secretly salt the secret, International Conference on Networking, Systems and Security (NSysS), Dhaka, (2017), 115-123. doi:10.1109/NSysS.

[6] Nikolay Elenkov, Android Explorations, Password Storage in Android M. doi:http://nelenkov.blogspot.in/2015/06/password-storage-in-android-m.html, 2015.

[7] Bh. Padma and G. V. S. Raj Kumar, A Novel Approach to Thwart Security Attacks on Mobile Pattern Authentication Systems, I. J. Computer Network and Information Security, 10(5) (2018), 18-27. Published Online May 2018 in MECS (http://www.mecs-press.org/). DOI: 10.5815/ijcnis.2018.05.03. ISSN: 2074-9104.

[8] Bh. Padma and G. V. S. Raj Kumar, Preventing *c* Attacks on Mobile Pattern Passwords, Journal of theoretical and applied information technology 96(4) (2018). ISSN:1817-3195.

[9] Bh. Padma and G. V. S. Raj Kumar, Dynamic salt generation for mobile data security using elliptic curves against precomputation attacks, International Journal of Image Mining 2(3/4) (2017), 179-194. ISSN: 2055-6047.

[10] Bh. Padma, G. V. S. Raj Kumar, A Review on Android Authentication System Vulnerabilities, International Journal of Modern Trends in engineering Research.

IJMTER 3(8) (2016), 118-123. ISSN: 2349-9745.

[11] Bh. Padma and G. V. S. Raj Kumar, Design and Analysis of an Enhanced SHA-1 Hash Generation Scheme for Android Mobile Computers, International Journal of Applied Engineering Research (IJAER) 11(4) (2016), 2359-2363. ISSN: 0973-4562.

[12] Bh. Padma and G. V. S. Raj Kumar, An Identity based Secure Pattern Authentication System, International Journal of Recent Technology and Engineering (IJRTE), ISSN: 8(1) (2019), 2277-3878.

[13] R. Sobti and G. Geetha, Cryptographic hash functions: a review, International Journal of Computer Science Issues (IJCSI) 9(2) (2012), 461-469.

[14] W. Stallings, Cryptography and Network Security, Principles and Practice. Prentice Hall, New Jersey 2006.

[15] Sukhchain Singh and Amith Gover, Study and Analysis of Dictionary attack and Throughput in WEP for CRC-32 and SHA-1, International Journal of Computer Applications, 96(17) (2014), 15-18. doi:10.5120/16885-6896.