International Journal of Recent Technology and Engineering (IJRTE)
ISSN: 2277-3878 (Online), Volume-8, Issue-1, May 2019

An Identity based Secure Pattern Authentication System

GVS Raj Kumar, Bh Padma, K Naveen Kumar

Abstract*: Mobile security is critical today as the usage of mobile devices has been increasing and consequently mobile security becomes more crucial. People are frequently using mobile devices for secure storage of their sensitive data like social security numbers, credit card numbers etc. If these devices are not handled securely, anyone can access the devices by hacking authentication passwords. Pattern locking systems are commonly exercised for validating a user for mobile access. But these systems are not safe, and are subjected to pre-computation attacks like dictionaries, rainbow tables and brute-force attacks. Android Kit Kat and Lollipop pattern authentication systems are vulnerable to pre-computations since they use SHA-1 unsalted hashes. The latest versions of Android like Marshmallow utilize SCRYPT hashes and salts for authenticating the users; they need additional hardware support like Trusted Execution Environment (TEE) and Gatekeeper functionality. Therefore this research presents an alternative representation for mobile patterns using elliptic curves, and proposes three algorithms based on this ideology to make the pattern passwords strong against these attacks without using additional hardware. Security analysis regarding SAC (Strict Avalanche Criterion) and brute-force search space is also presented in this paper. Executions times are analyzed after the implementation of the three proposed methods.

Index Terms: brute-forcing, dictionaries, Mobile security, elliptic curves, Pattern Locking, rainbow tables, SAC.

I. INTRODUCTION

Mobile phones hold lots of confidential information, so it is critical to lock our device using any one of available choices. On Android, user can choose from a PIN, pattern, or password. Since a password takes much longer time to type, most people use either a PIN or pattern code. Pattern code is chosen by users by connecting 4-dots among 9 points on a 3x3 grid as shown in Fig. 1. The SHA-1 hash of this pattern is stored in files which is placed system's root folder called gesture.key. Anyone can gain this hash values if the mobile is USB enabled, and rooted. There are 3,89,112 no. of combinations for the user to pick a password.

Revised Manuscript Received on 30 May 2019.
***** Correspondence Author
Dr G.V.S Raj Kumar*, Associate Professor in the Department of Information Technology, GITAM Institute of Technology, GITAM(Deemed to be University) Visakhapatnam.
Smt Bh Padma, Assistant Professor in the Department of Computer Sciences, GVPPG, Visakhapatnam-45.
Dr K Naveen Kumar, Assistant Professor in the department of Information Technology, GIT, GITAM University, Visakhapatnam.

As this number is limited, hackers can experiment freely available SHA-1 dictionaries such as Andriller or SQLite browser [10], which breaks the password [1] for a given Sha-1 hash code. There are many forensic tools available in the market to crack the pattern passwords. So Generating a SHA-1 hash for the pattern is not safer, but it protects the password to some extent.

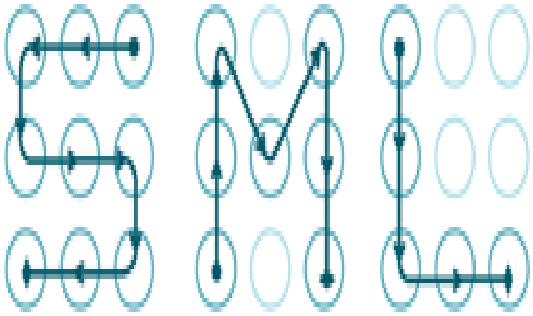


Fig.1: Pattern Authentication

SHA-1 Pattern Authentication System Model

SHA-1 is an irreversible hash function which is applied for authenticating users when they log in through pattern locking systems. It produces a 160-bit checksum which is stored in device's root database for authenticating a user. Whenever a user logs in the SHA-1 hash code is again calculated and stored hash value is tested against the calculated Sha-1 hash value. Fig.1 shows the current system model for authenticating users using pattern locks in older versions of Android. After user selects a pattern, the signature of the pattern password is produced using SHA-1 algorithm by the Android application and is stored as a "gesture.key" file in the device root directory in the folder "/data/system". When the user authenticates again to access the device, the application builds the SHA-1 signature again and the newly generated signature is judged against the signature that is hoarded in 'gesture.key' file. If there is a match, the user can be permitted to access the device else one should make another trail for logging into the device. For Kit Kat and Lollipop patterns SHA-1 is the existing method to produce the hashes. This process is shown in Fig.2.

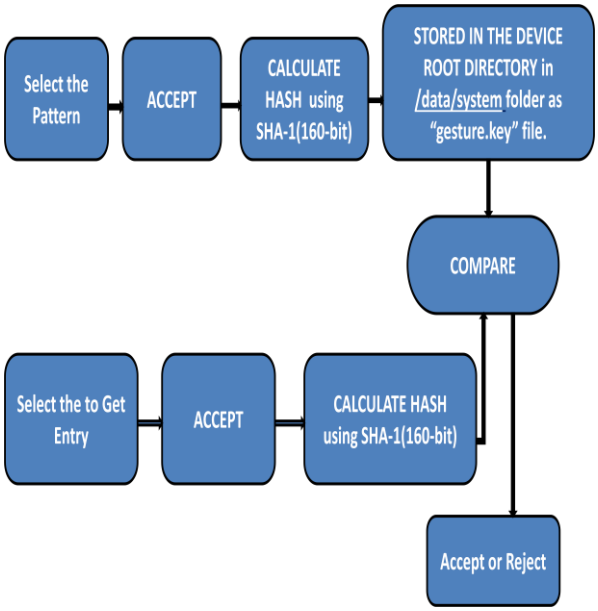


Fig.2: SHA-1 Pattern Authentication

Unfortunately, SHA-1 is prone to dictionaries and rainbow tables [9]. Since SHA-1 is an irreversible encrypting scheme, there will be no reverse function to convert this hash code to original input message. To obtain the hash code, the hacker needs to generate a list of pattern password messages along with their respective hashed strings. Attacker can download this dictionary and without difficulty can discover hash that matches the original pattern sequence. The vulnerabilities of pattern locks happen majorly because the hashes not salted.

The pattern mechanisms of Android particularly in lower versions store SHA-1 checksum of the pattern sequence in /data/system/gesture.key file [5] and this hash is not salted. A big crisis with pattern lock is it stores the hash sequence as an unsalted hash code. After user selects a pattern password 1-4-5-6-9, this pattern password will be stored as a SHA-1 hash of length 160-bits. It is practicable to crack SHA-1 hash in less amount of time and disclose the actual pattern by means of Android "gesture.key" file.

Brute-forcing using Salts

A brute-force attack is typically used as a last-resort approach in a cryptanalysis state, as it very much engages extreme amounts of trial and error and frequently relies on a lot of fortune to find the key. A brute-force attack is dissimilar to a dictionary attack, as it never depends on a dictionary and simply attempts every possible key that could be used. Even for smaller key spaces a brute-force can take a number of days at least depending on accessible computational power and for modern encryption systems the brute-force attack would takes at least hundreds of years**.**

Technologically possible, but no longer is a brute-force attack a realistic way of breaking encryption mechanisms. M. Kammerstetter proposed a high speed brute-force attacks [14] based on FPGA (Field-Programmable Gate Array) and compared it with GPU (Graphics Processing Unit-based systems. 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**.** The former versions of Android like Kit Kat password systems are not using salted hashes. Salted hash has a benefit that when the hash is broken, you cannot get the password. Android Kit Kat pattern locks and Android Lollipop versions are not employing salted hashes. But the newest versions of Android like Marshmallow authentication systems used salted hashes generated by Scrypt and HMAC algorithms 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. 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 is equal to hash (salt+password). This leaves rainbow tables and hash tables useless. Even the pattern lock authentication schemes employ salts, they are susceptible to brute-forcing if the mobile is rooted and USB enabled. Fig. 3 shows how a dictionary can be used even when salts are employed.

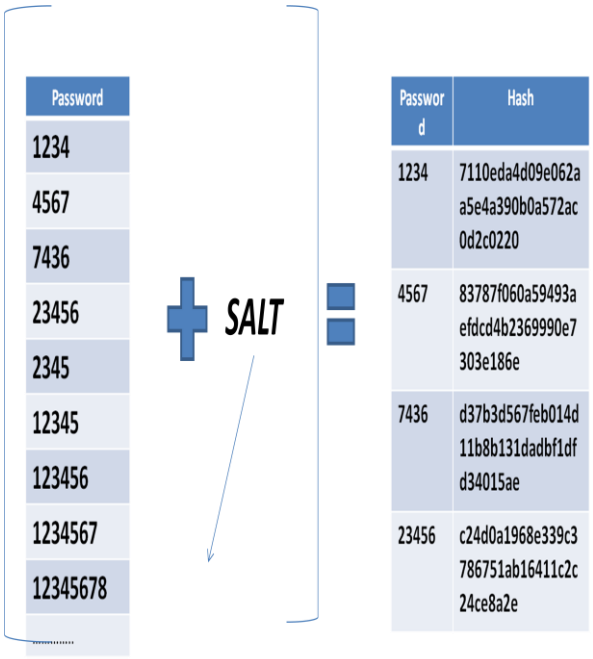


Fig.3: Brute-forcing when salts are used in SHA-1 Authentication

II. PROBLEM STATEMENT

In the Android pattern locking scheme, the system presents a 3×3 pattern or a grid to the user, and the password of a user is a pattern drawn on the grid, by connecting the points on the grid. This locking system is very essential for those who have secret data or personal files in their Android devices. These pattern-based passwords are not protected as compared to other mobile authentication schemes. But many users choose this scheme to authenticate them to the device unfortunately. There exist security breaches on Android Pattern Locking Systems such as rainbow table attacks, dictionary attacks, and brute-forcing particularly while the devices are left USB enabled or rooted.

These attacks are based on certain forensic tools which pre-compute the password hashes by knowing the necessary information about the underlying hashing technique used. Mathematically, it is not impossible to experiment all the combinations between 0123 - 876543210 using dictionaries or rainbow tables. After Kit Kat 4.4, there are a few alterations in the authentication schemes made by Android which employ salt values [9] for hashing especially in the modern versions of Android such as Marshmallow. But even salts are added to the passwords, the hackers still gain the passwords using brute-forcing, as salts are placed in the device folder even though they are generated for each user randomly. If the salts are compromised, then it is easy for the assailant to brute-force the pattern password by utilizing the value of salt. Therefore, this paper mainly focuses on finding the solution to pre-computation attacks on pattern locks. Since pattern locking systems are subjected to dictionary, rainbow tables and brute-forcing attacks, a new secured authentication mechanism is required to construct them hard enough to oppose these attacks.

III. LITERATURE REVIEW

Elliptic Curve Arithmetic

Elliptic curve cryptography [6] was invented by Miller and Koblitz, a very efficient asymmetric key security protocol based on discrete logarithms. When compared with other outstanding cryptographic protocols such as RSA, ECC proposes shorter key sizes but with an equivalent level of security, and demands less system resources like processor and main memory. Obviously ECC is ideal for the electronic gadgets such as PDAs, pagers, cellular phones, smart cards which require less power and computation. The security and efficiency of elliptic curve based protocols such as encryption, signature generation, authentication, secret key exchange etc. depends on how tough it is to resolve the discrete logarithm problem which is stated as 'given P,Q in G and one has to try to solve kP = Q **(**always assume that k exists)'. An elliptic curve defined under a finite field is defined using two parameters a, b (where a, b formulate the relation, $4a^3 + 27b^2 \bmod p \neq 0$), consists of all the points (x, y) that obey a curve represented by the relation $y^2 = x^3 + ax + b \pmod p$.

Consider two points P (x_1,y_1) , Q (x_2,y_2) be elements of the curve. Then

P+Q = R(x_3, y_3) where $x_3 = \lambda^2 - x_1 - x_2$,

$y_3 = \lambda (x_1 - x_3) - y_1$, and

$\lambda = (y_2 - y_1) / (x_2 - x_1)$. The sum of the points P and Q can be thought of as a point of intersection of P and Q is a straight line that goes through both the points.

Let P = (x, y) in $E(F_p)$.

Then the point Q = P + P = 2P = (x_3, y_3) also in E where

$x_3 = \lambda^2 - x_1 - x_2$, $y_3 = \lambda(x_1 - x_3) - y_1$, and

$\lambda = (3x_1^2 + a) / 2y_1$.

The set of points on E(Fp) also comprise point O, which is called as the 'point at infinity' and also an identity element with respect to addition. E(Fp) forms an abelian group [4] under operation 'addition'. If P = (x, y), then the negative of P is denoted by –P = (x, – y) which also lies on the curve. Elliptic Curve Discrete Logarithm Problem can be declared as follows: "Let E be an elliptic curve and p, q are points on the curve E. It is not easy to discover an integer value 'n' such that nP=Q". nP can be calculated as P+P+…..+P(n times). The common problem people constantly face with elliptic curves is before working with text messages, it is required representing them with curve points. In the proposed system also employs Koblitz encoding method for representing the message using elliptic curve points.

Elliptic Curve Cryptography for Mobile Computers

Elliptic curve security systems can be efficiently used on mobile devices, sensors, smart cards, and so on. The key of an elliptic curve based crypto system catches significantly less memory. The ratio amplifies rapidly with the increase of security levels. There was much disbelief about its security. After a decade of severe study and inspection, ECC has yielded highly competent and secure.

Smart Cards are the most accepted devices for security implementations using ECC [8]. Many modern companies are manufacturing smart cards that use elliptic curve based digital signature schemes. These companies include Fujitsu, Phillips, DataKey, MIPS Technologies etc. Smart cards are flexible tools and are utilised in many situations. Smart cards are used as credit or debit cards, electronic tickets, registration cards, PDAs and are believed to be best choice for implementing cryptosystems since they have more and more computing power when compared to other mobile devices, like pagers and cell phones.The size and cost of handheld tools like mobile phones and PDAs is a big issue in implementing strong security solutions as these devices have less amounts of incomplete RAM, ROM and processing speed. Therefore, the problems related to computation and communication overheads and security parameters should be considered seriously regarding these handheld devices.

ECC is being acknowledged as an alternative for well known asymmetric key cryptosystem RSA and ElGamal, as it offers the maximum key strength than other public key cryptosystem identified as of today. The length of ECC keys is comparatively smaller than other public key systems still offers equal level of security. For a specified key size, ECC offers maximum cryptographic security per bit.

Desai S. et al [11] implemented ECC for smart phone Operating Systems. Their experiments have shown that ECC needs less computation times relatively when key length becomes more, which is beneficial on smart phones. Their implementation proved that memory utilization is reduced and cipher text size is also reduced.

According to Ariel Hamlin [12] ECC has shown to be a fast, secure and computationally cheap substitute to other encryption mechanisms on mobile devices. Richard K. Ansah et al [13] stated that elliptic curve supported protocols are suitable for smart cards for banking transactions and these protocols can be used for whatsApp message security in mobile phones.

Wendy Chou [14] in 2013, in his study concluded that ECC is the most suitable in a constrained environment such as mobile computing. Muhammad Yasir Malik [15] in 2010 employed ECC by means of general purpose microcontrollers and Field Programmable Gate Arrays (FPGA). Malik stated that this implementation may be proficiently employed in low-power applications and ECC is a perfect choice for mobile, portable, and low hardware and power applications.

IV. RELATED WORK

SHA-1 and Elliptic Curve based Protocols

As it is seen before that a 'digital signature' is an idea of presenting the genuineness of documents or digital messages. A convincing digital signature presents a receiver reason to judge whether the message was produced by a well-known sender. The sender cannot decline that he just sent the information and that the data was not altered in the transmission as well. Elliptic curves based digital signatures (ECDSA) are used in messaging and security. To build a digital signature, signing algorithm makes a hash code of the data that has to be signed. This hash [7] is consequently encrypted by key of the sender. SHA-1 is believed to be a suitable hashing mechanism for protocols that employ ECDSA (Surbhi Aggarwal, 2014). Many researches proposed SHA-1 as a hash function that goes well with elliptic curve digital signature. This study discusses about a few number of researches done up to now that embraced SHA-1 for ECDSA.

In 2001, Don Johnson [16] stated that the strength per one key bit is significantly more in elliptic curve cryptosystems because of the discrete logarithm systems, and for signature generation they employed SHA-1 scheme. Aqeel Khalique [17] described the implementation of ECDSA and discussed related security issues. On November 28, 2011, O.S. Sury described about the use of SHA-1[19] algorithm with RSA, DSA and for SSHFP finger prints. In 2015, Mr Srinivasan [20] utilized ECDSA for e-payments in secure e-commerce and e-payment systems to give security for their transactions and communications and by shielding passwords. They used SHA-1 checksum for signature implementation using ECDSA. He concluded that encryption techniques use a considerable amount of computational resources such as main memory, CPU time, and battery power. Alfred Menezes discussed about the amount of security offered by ECDSA while using SHA-1 hashes [22]. In October 2006, Erwin of Siemens AG, gave examples of ECGDSA over GF (p) using hash function SHA-1 [23]. V. Gayoso [24] presented a complete introduction to ECIES (Elliptic Curve Integrated Encryption Scheme), where he mentioned SHA-1 as a standard hash function for the implementation of the ECIES. In 2013, Nacy proposed the Vehicular Ad-Hoc Network (VANET) that believes cars as nodes in a cellular network [25]. The digital signature they used in VANET as a standard is ECDSA and SHA-1 is used for signature verification.

Objectives of the Research

This research proposes a method that represents the pattern grid using elliptic curve points and later generates the SHA-1 hash value of the pattern password so that the hash becomes impractical to be hacked using dictionaries. And this method is extended to generate a dynamic salt[21] as well as pepper values which are added to the original password before it is hashed, so that pattern passwords[26] become free from dictionaries. Preliminarily this thesis emphasizes on the following objectives given below, which are found to be implementable and applicable on hashing functions of pattern locks and may consequently make the pre-computations quite impossible.

The existing representation of the pattern using integers should take an alternative representation.

- Make the pattern representation unique to the user internally using points of an elliptic curve so that the input is transformed to an intermediate form and dictionary attacks turn to be unfeasible.
- Incorporate Scalar Multiplication, Discrete Logarithms of elliptic curve as elliptic curve discrete logarithms are difficult to break.
- As dynamic salts are resistant to brute-forcing, extend this idea to create dynamic salts for SHA-1 patterns. As the salt value is unknown, attacker cannot brute-force the password. Attacker cannot acquire the salt from the database as well.
- As salts are susceptible to brute-force, improve security by increasing the brute-force search space by adding a pepper to the password along with salt.

Using this philosophy the current research derives three solutions to make this idea practical. Initially the first proposed method generates a SHA-1 hash of the intermediate input that is gained by representing the grid using elliptic curve points. The second proposed system finally generates a salt which is dynamically created depending on the user's password pattern, using identities of user like Gmail-ID and Device-ID. This dynamic salt need not be placed it in the system's directory securely and therefore prevents the pre-computation assaults such as dictionary attacks, brute-forcing and rainbow tables. This system makes brute-forcing consequently more difficult. The third method extends the second method by adding a dynamic pepper also to the pattern passwords to make the brute-force analysis much difficult for the hackers. The way the salt and pepper values are applied, is idiosyncratic for each password. As a result, a more secured password storing scheme can be accomplished. The subsequent objective of this thesis is to examine the security and performance issues when these two methods are implemented for the pattern passwords.

V. PROPOSED MODEL FOR ANDROID PATTERN LOCKS

The proposed method begins by representing the pattern grid using elliptic curve points as shown the above Figure 1.9, and later generates the SHA-1 hash value of the pattern password. The hash becomes impractical to be hacked using dictionaries. And this method is extended to generate a dynamic salt as well as pepper values which are added to the original password before it is hashed, so that pattern passwords become free from dictionaries. Using this philosophy the current research derives three solutions to make this idea practical. Initially the first proposed method generates a SHA-1 hash of the intermediate input that is gained by representing the grid using elliptic curve points. The second proposed system finally generates a salt which is dynamically created depending on the user's password pattern, using user credentials such as Gmail-ID and Device-ID. This dynamic salt need not be hoarded it in the system's root directory securely and makes brute-force attack consequently more difficult also dictionaries and rainbow tables becomes impossible.

The third method extends the second method by adding a dynamic pepper also to the pattern passwords to make the brute-force analysis much difficult for the hackers. The way the salt and pepper values are applied, is unique to each and every password.

As a result, a better password storage scheme can be accomplished.

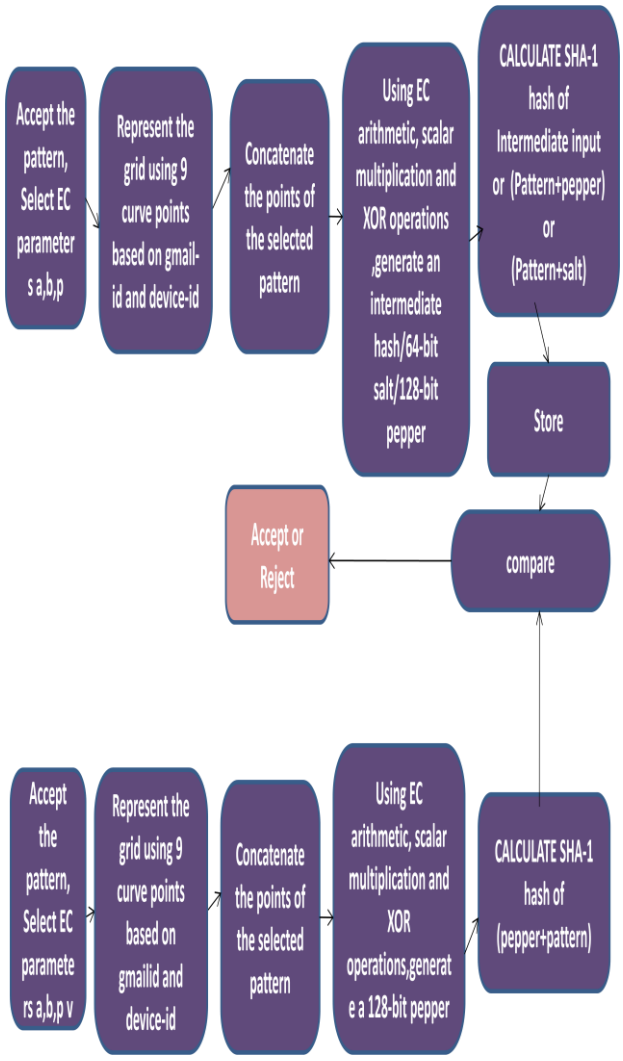


Fig.4: Proposed System Model

Proposed Method-I (Enhanced SHA-1)

The current method represents the pattern grid using points of the elliptic curve and these points are to be originated from the Device-Id and Gmail account of the user. Later the current system is enhanced and the input is transformed using Scalar or Point Multiplication on the selected pattern points of the pattern. This leads the pattern generation using SHA-1 more complex and opposed to Dictionary and Rainbow table attacks[18].

Algorithm:

STEP 1: Attach the Gmail account to Device-Id.
STEP 2: Obtain an integer k with the Device-Id using sequence of Exclusive-OR operations.
STEP 3: Generate points of elliptic curve to represent this string by using Koblitz's encoding method [2].
STEP 4: Select 9 unique points from the generated points to symbolize the grid.
STEP 5: Shape the grid with selected points.
STEP 6: According to the user's chosen pattern multiply each and every point with the integer k which generates a different sequence of points for the user selected pattern.
STEP 7: Now concatenate these points and break the string into two parts and carry out operation 'Exclusive-OR' with each other.
STEP 8: Repeat step (i) two times construct an intermediate message.
STEP 9: The SHA-1 checksum of this file is now stored in the password file.

Proposed Method-II (Salted SHA-1)

This method generates a salt which dynamic and it is to be appended or prepended to the pattern password by representing the pattern using Gmail-Id and Android-Id of the user. Dynamic salts always save passwords from dictionaries and brute-forcing. As this salt need not be placed along with the hash in the database it makes the pattern password difficult to guess using brute-forcing and obviously salts always prevent rainbow tables.

Algorithm:

STEP 1: Perform the steps from (1) through (7) of Proposed Method-1.
STEP 2: To build this message as a 64-bit value, turn round the two equal parts and add them to pad the original string.
STEP 3: Spot this message as a SHA-1 salt value.
STEP 4: Attach this salt to the selected pattern password.
STEP 5: Produce SHA-1 hash of the newly concatenated message.
STEP 6: Store this hash code in the directory(root) of device for authenticating the user.

Proposed Method-III (Peppered SHA-1)

The same ideology of the first proposed method is extended to derive a pepper value to be attached to the pattern passwords. The motivation behind generating this pepper(long salt) is to increase the brute-force security since shorter salts are risky with respect to brute-forcing. Dynamic peppers always save passwords from dictionaries and brute-forcing. As this pepper need not be placed besides the hash in the database it makes the pattern password difficult to guess using brute-forcing.

Algorithm:

STEP 1: Perform the steps (1) through (3) of proposed method - 2.
STEP 2: Mark this slat value as String-1.
Generate a dynamic message by applying cross multiplication and complement.
STEP 3: Mark this message as String-2.
STEP 4: Add String-1 and String-2 and name it as a Pepper.
STEP 5: Concatenate this dynamic pepper to the original pattern password.
STEP 6: Create SHA-1 checksum of the concatenated message.
STEP 7: Store this hash message in the system's root directory of the device for authenticating the user.

Illustration with an Example

Algorithm-1:

Step 1: Pick Gmail-Id and Android-Id and of the mobile user.
Let's say Gmail-Id = accountname@gmail.com,
Device-Id = "67fda43cc1010bca",
Step 2: Now add this Gmail-Id and Device-Id to get a long string STR.
In the present example, the string is
"67fda43cc1010bcaaccountname@gmail.com" .
Step 3: Now discover an integer using Device-Id by carrying out a number of XOR operations after translating each character to its equivalent ASCII code:

$54 \oplus 55 \oplus 102 \oplus 100 \oplus 97 \oplus 52 \oplus 51 \oplus 99 \oplus 99 \oplus 49$

$\oplus$ 48 $\oplus$ 49 $\oplus$ 48 $\oplus$ 98 $\oplus$ 99 $\oplus$ 9 = 5.

Step 4: Choose elliptic curve parameter inputs a, b, p to re-generate the pattern grid.

Let's say a= 9; b=7; p=2011;

The equation of the elliptic curve is

$y^2 \pmod{2011} = x^3 + 9x + 7 \pmod{2011}$.

Step 5: Represent every character of the above string STR, by a point of this curve using Koblitz's method of Encoding which is exposed in table 1.

Step 6: Arbitrarily select 9 points from the above using any criterion. Now sort all these points in the increasing order of y-coordinates and select the first and foremost 9 unique points. Now we will get the points: (1741, 16) (1441,30) (481,91) (1462,123) (421,149) (363,173) (1721,195) (1341,250) (441,445).

Step 7: The pattern is now presented using the above selected points. Let us suppose, the user selects a pattern "123569".

Step 8: Now select the points for the selected input pattern '123569'.

So the points i.e. (1741,16),
(1441,30),(481,91),(421,149),(363,173),
(441,445) signify the pattern "123569".

Step 9: Now execute a scalar multiplication of this selected pattern points by the value obtained using Device-Id, so that different points of elliptic curve will be generated to represent the pattern password. The following table 1 shows the scalar multiplication operations which are performed.

Table 1. Curve points representing the String

6 ->(481,91)	7 -> (502,661)	f ->(1441,30)
d ->(1401,672)	a ->(1341,250)	4 ->(441,445)
3 ->(429,149)	c ->(1382,758)	c ->(1382,758)
1 ->(381,554)	0 ->(363,173)	1 ->(381,554)
0 ->(381,554)	b ->(363,173)	c ->(1382,758)
a ->(1341,250)	a ->(1341,250)	c ->(1382,758)
c ->(1382,758)	o ->(1624,862)	u ->(1741,16)
n ->(1603,905)	t ->(1721,195)	n ->(1603,905)
a ->(1341,250)	m ->(1587,865)	e ->(1421,830)
@->(681,446)	g ->(1462,123)	m ->(1587,865)
a ->(1341,250)	i ->(1502,557)	1->(1561,975)
. ->(321,525)	c ->(1382,758)	o ->(1624,862)
m->(1587,865)		

5*(1741, 16) = (1349, 1265)
5*(1441, 30) = (1792, 1884)
5*(481, 91) = (1217, 1022)
5*(421,149) = (1171, 281)
5*(363,173) = (1671, 1250)
5*(441,445) = (457,931)

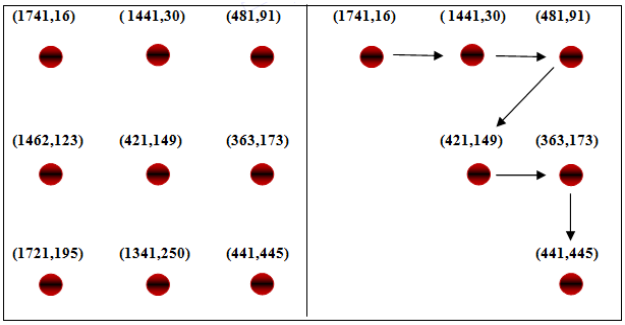


Fig. 5: 1-2-3-5-6-9 new input pattern representation

***Step* 10:** After that convert the points to hexadecimal. The hexadecimal points obtained are: (545,4F1), (700,75C), (4C1, 3FE), (493,119), (687,4E2), (1C9, 3A3). These points are concatenated one after other to a long string: '5454F170075C4C13FE4931196874E21C93A3'.

Step 11: Now divide this string into 2 equal parts and carry out Exclusive-OR operation with each other
and follow this step twice to ultimately get a string.

5454F170075C4C13FE $\oplus$ 4931196874E21C93A3 = 1D65778187370715080776.

The above string is divided again into two halves: (pad with 1's if we get the string length odd).

1D657781873 $\oplus$ 70715080776 = 674142701F05.

This intermediate hash is input for SHA-1 and hash should be found. We need to make this value not exceeding 64 bits by repeating another XOR operation by splitting these values into two halves.

Step 12: Now find SHA-1(674142701F0) = **de09382f281e32b67ca079b388a7968b4627a9c7**, which is placed in gesture.key file and this hash cannot be found in SHA-1 dictionaries.

Algorithm-2:

Step 13: Make the intermediate hash string derived in step 11, as a 64-bit value. For this break the string into two equal halves, reverse them and then the 2 strings obtained are: 241476, 50F107. By concatenating both the strings, the final string is 24147650F107. Concatenate the current derived string, to the string which is derived in Step 11, and use the first 16 bits to produce the dynamic message input. Now choose the first 16-bits of 674142701F0524147650F107, get first string as

String-1 = "674142701F052414".

Step 14: This string will serve as a dynamic salt to be attached to the original password, and obviously salts prevent dictionaries and rainbow tables.

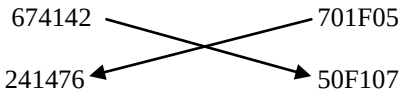
Step 15: Calculate

SHA-1(674142701F052414: 123569) =
'7842042f5cc46a79a4f700629e87a1ab2f81a9b2'
which is stored in gesture.keyfile.

Algorithm-3:

Step 16: Generate another dynamic message string by applying cross concatenation and 1's complement of the string which is derived in Step 13. The string derived in Step 13 is '**674142701F0524147650F107**'. Now divide this string into two equal parts and carry out
cross - concatenation.

Cross Concatenation:
Perform cross- concatenation:



The string derived after cross-concatenation is : 67414250F107701F05241476.

1's Complement:
Perform 1's complement by changing this string into binary code and reverse them and next convert this binary string to hexadecimal the string derived is: '98BEBDAFCEF88FECFADBEB89'.

Now deduce a 16-bit bit pepper value from the above string as "98BEBDAFCEF88FEC". So String-2 = "98BEBDAFCEF88FEC".

Step 17: Concatenate the above two strings to produce a dynamic pepper message of size 128-bits.
Pepper = String-1 + String 2
= "674142701F052414 98BEBDAFCEF88FEC"
SHA-1 ([PEPPER:PASSWORD) produces the final hash.
Now pattern input is: 123569 and the peppered output is 674142701F05241498BEBDAFCEF88FEC: 123569.
Calculate SHA-1 hash of 674142701F05241498BEBDAFCEF88FEC:123569 is **"09611DC681E8CB393E31EE0804246172429C2F".**
This hash will be hoarded in device database. This pepper message always prevents the pattern passwords from precomputations such as brute-forcing, dictionary and rainbow tables.

VI. BRUTE-FORCE SECURITY ANALYSIS

In this research, a modern approach is suggested for pattern password systems to oppose brute-forcing and dictionary attacks. Attackers can obtain the hash code from the device database and exercise offline brute-force attack to speculate the pattern. Here salts and peppers are dynamically generated and make the pattern authentication system strong to survive from these pre-computation attacks[27]. Here the outcome of prepending the salt or a pepper message value is that as they boost the brute-force search space and as a result, increase the endeavour required by the attackers to break the passwords as compared with the existing scheme i.e. SHA-1

Table 2. Brute-force search space assessment by using all the 3 proposed methods

Authentication Scheme used	Brute Search-Space
SHA-1	3,89,112
Enhanced SHA-1	$3,89,112 * 2^{\text{size of the hash}}$
Salted SHA-1	$3,89,112 * 2^{64}$
Peppered SHA-1	$3,89,112 * 2^{64} * 2^{64}$

As it is monitored in the above table 2, the brute-force search space of the peppered scheme is amplified by 2^{64} times, that of the formerly proposed salted SHA-1 system and Enhanced SHA-1 system. It shows that the suggested systems enhance the security with regard to brute-forcing.

VII. PERFORMANCE EVALUATION OF THE PROPOSED SYSTEMS

The performance of the proposed schemes can be obtained using any criterion for hashing algorithms like hashing duration, avalanche effect and collisions. Here it is not feasible to guess the collision resistance of the proposed scheme as there are 3,89,112 no. of combinations of pattern inputs for a 3x3 pattern grid. So, the efficiency and performance and of the proposed systems is tested w.r.to brute-force search space, time complexities and avalanche effect. Avalanche Effect of hash functions in cryptography refers to the most important property of cryptographic hash functions. Avalanche effect can be satisfied if the output alters significantly by means of a little change in message input. Here experimental results as well as comparative analysis of the current system w.r.to the proposed systems are presented.

Strict Avalanche Criterion (SAC)

Strict avalanche Criteria is a special observation in the case of avalanche effect. SAC is an attribute of avalanche effect which fulfills the criteria of the hashing functions [14], "if, one in the input is reversed, every bit in output must change with a possibility of one half." SHA-1 exhibits Strict Avalanche Criterion. Here, as the input message of the pattern is changed or salted or peppered, after modifying the initial input data for hashing, the differences in avalanche effect of the proposed schemes are examined. After that SAC statistics of the existing system SHA-1, and the proposed schemes Enhanced SHA-1, Salted SHA-1 and Peppered SHA-1 are observed using a computer with 4GB RAM and i3 processor with speed 1.90 GHz and 64-bit OS. The SAC values for these schemes are decided by taking hundred input pairs of plaintext pattern messages.

The Avalanche Criterion is assessed by accumulating casual input patterns of 4,5,6,7,8 point pattern which differ by a single bit. The graphs are drawn for all the above combinations of patterns to watch the avalanche effect of the proposed systems and current system SHA-1. The following diagrams 6-10 show the SAC difference between the proposed schemes and current scheme. Consider the number of the pattern on x-axis and Avalanche Effect for large amount of input patterns on y-axis.

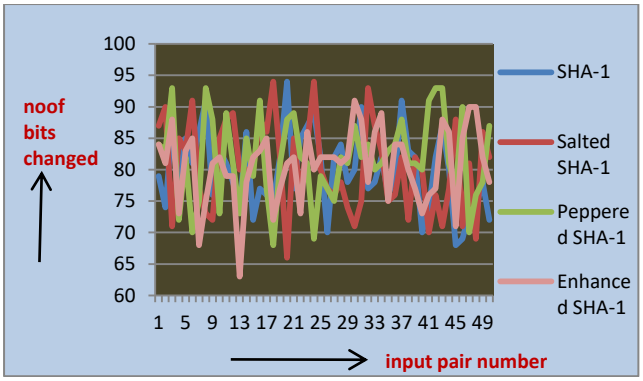


Fig.6: 4-point pattern SAC

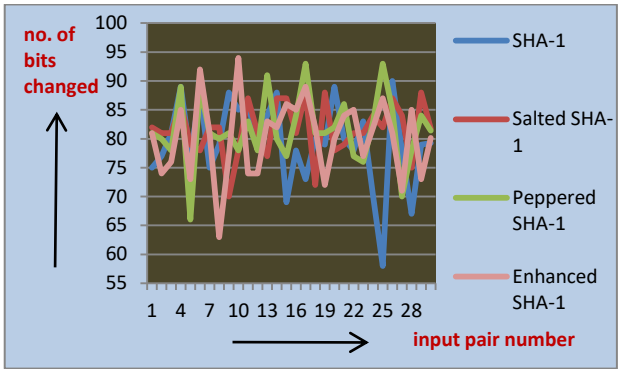


Fig.7: 5-point pattern SAC

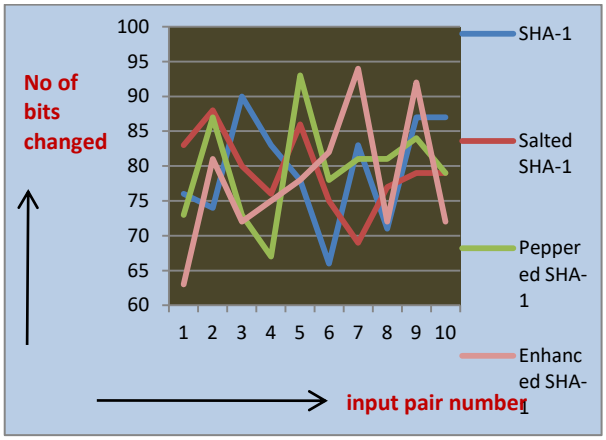


Fig.8: 6-point pattern SAC

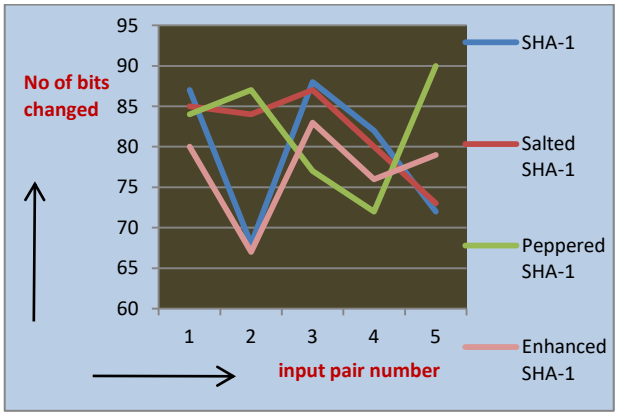


Fig.9: 7-point pattern SAC

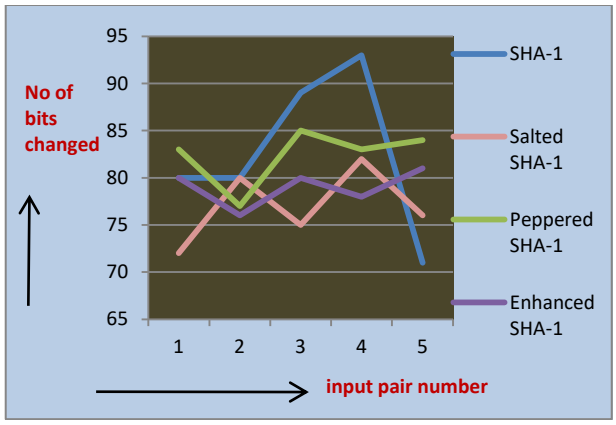


Fig.10: 8-point pattern SAC

Fig. 11 and table 3 present overall Strict Avalanche Criterion of the three proposed methods. According to the results gained, we can assume that the proposed methods exhibit the SAC for any pattern sizes like SHA-1.

Table 3. Overall SAC observed

SAC Observed	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
% of bits changed	49.9	49.51	49.68	49.6	51.6	50.0
Enhanced SHA-1	80.74	80.167	78.1	77	79	79
% of bits changed	50.5	50.10	48.8	48.12	49.3	49.4
Salted SHA-1	80.36	81.767	79.2	81.8	77	80.03
% of bits changed	50.22	51.1	49.5	51.12	48.12	50.01
Peppered SHA-1	81.74	81.433	79.6	82	82.4	81.43
% of bits changed	51.08	50.9	49.75	51.25	51.5	50.9

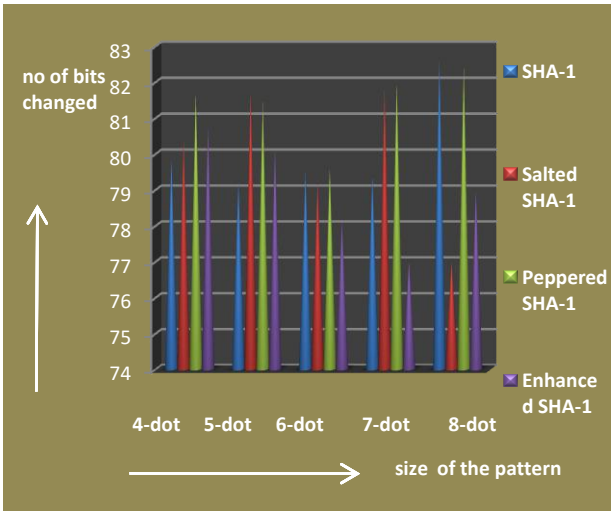


Fig.11 Overall SACs of all the three proposed methods

Time complexities

The time complexity of an algorithm decides the total time required by any algorithm to execute as a function of the number of inputs. Efficient algorithms are always required as CPU time is very costly. Computation is not an advantage if it takes exceptionally large time to resolve a problem. But lazy hash functions will have some benefits if they execute in lesser quantity of time. Brute-forcing needs a lot of time for slow hash functions, this increases security regarding such a threats.

Table 4. Average CPU Execution Times of the 3 methods

CPU Time in millisecs	4-dot	5-dot	6-dot	7-dot	8-dot
SHA-1	9.4	10.3	10.3	10.3	10.3
Enhanced SHA-1	16	16	16	16	16.75
Salted SHA-1	17	17	16	16	16.75
Peppered SHA-1	17.5	18	18.66	18.66	19

To assess the performance, the processor times of the proposed algorithms are calculated. Fig. 12 and table 4 show the average execution times of hashing of the proposed systems for all the pattern sizes after results have been taken after execution. The results confirm that the time complexities have been increased while compared with the existing system SHA-1 as shown in the table 4. It is verified that the time complexity of the scheme peppered SHA-1 hashes is getting increased nearly twice that of the existing system i.e. SHA-1.

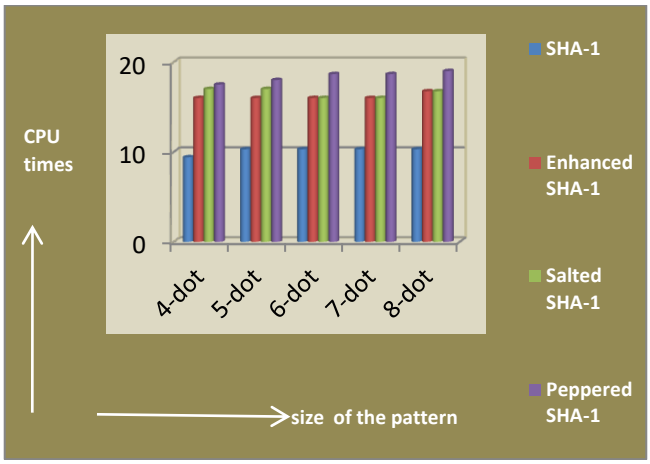


Fig. 12. Average Execution times in milliseconds

But there are no much dissimilarity between the processor time of peppered SHA-1 system and salted SHA-1 system. Scalar multiplication algorithm and additional operations such as 1's complement are taking additional amount for execution. There is always security and time efficiency controversy existing for any cryptography algorithm, but here the proposed systems provides security [4] to the current hashing system at the cost of augmenting the time complexity for hash generation.

VIII. CONCLUSION

Usability and security are treated as the most important issues of system design which increase the user responsiveness of the system. Android mobiles require huge user friendliness. Since there exist many types of attacks on Android, deriving highly secured authentication systems becomes necessary. It is not much difficult to hack or bypass the Android pattern authentication schemes, the only obstruction is that one can make it not possible to get the "/data/system/ folder" and "gesture.key" file except when a mobile is USB enabled or mobile rooted. New security advances to keep away from dangerous taps on mobiles and to present improved authorization schemes compared to the existing one with respect to dictionary attacks are always needed. Further security improvements of these security schemes are very critical to hold out the pre-computation security threats. While hashing passwords, there are two significant considerations, one is security measure and the rest is computational cost.

In the present study, the existing SHA-1 method is improved to enhanced SHA-1 scheme and later generated a dynamic salt and a pepper which has been included to the original passwords to make the system strong in opposition to pre-computation attacks. As the pattern grid is dynamically produced by using elliptic curve points i.e. based on user credentials such as Device-Id and Gmail-Id, consequently these passwords and hashes become unique for each user. The present study commences a dynamic pepper generation algorithm which generates an intermediated hash that is unique to the user, because it is derived based on his identities for Android patterns authentication systems. The current system introduces a modified approach to produce a salt and a pepper of fixed size for SHA-1 pattern inputs based on elliptic curves. This augments the power of the proposed schemes as elliptic curve discrete logarithms elevate the system to be more robust and secured. As the salt and pepper are generated dynamically, it is not prone to brute-forcing using dictionaries. As peppers and salts are added to passwords, the dictionaries and rainbow tables are not possible as well. These algorithms develop the quality, efficiency and effectiveness of the existing technique i.e. SHA-1.

According to experimental results observed, it is proved that the proposed systems satisfies Strict Avalanche Criterion (SAC), the CPU times of the recommended techniques show that they engage more time as compared to original algorithm because these schemes are an expansion of the existing scheme. The proposed systems are safer because it contains elliptic curve arithmetic and discrete logarithm problem as elliptic curve discrete logs are proved to be strong and secured.

REFERENCES

1. Lashkari, A.H., et al., Shoulder Surfing attack in graphical password authentication. International Journal of Computer Science and Information Security, 2009. 6(9).
2. Bh Padma, "Encoding and Decoding of a message in the implementation of Elliptic curve Cryptography using Koblitz' Method", IJCSE,vol-2 Issue: 5, 2010.
3. L.Thulasimani and M.Madheswaran, "Security and Robustness Enhancement of Existing Hash Algorithm", Proc of IEEE International Conference on Signal Processing Systems,15-17 May, 2009.
4. Surbhi Aggarwal, "A review of Comparative Study of MD5 and SHA Security Algorithm", International Journal of Computer Applications (0975 – 8887) Volume 104 – No.14, October 2014.
5. Android Explorations, Password Storage in Android M", http://nelenkov.blogspot.in/2015/06/password-storage-in-android-m.html.
6. Michael Brown, Darrel Hankerson, Julio Lopez, and Alfred Menezes,"Software Implementation of the NIST Elliptic Curves over Prime Fields", D. Naccache,editor, Topics in Cryptology CT-RSA 2001, vol. 2020 of Lecture Notes in Computer Science, pp. 250-265. Springer-Verlag, 2001.

7. Bellare, Mihir, Canetti, Ran, and Krawczyk, Hugo, Keying hash functions for message authentication (online), University of California San Diego, Computer Science and Engineering, 1996, available at "http://cseweb.ucsd.edu//~mihir/papers/-kmd5.pdf", Accessed on 2013-07-03.
8. Randhir Kumar and Akash Anil, "Implementation of elliptical curve cryptography", International Journal of Computer Science Issues (IJCSI), vol. 8, issue 4, no 2, page(s): 544–549, July 2011.
9. Theocharoulis K., Papaefstathiou I., Manifavas C. Implementing Rainbow Tables in High-End FPGAs for Super-Fast Password Cracking. Found in: International Conference on Field Programmable Logic and Applications. Issue Date:September 2010. pp. 145-150.
10. Bh Padma, GVS Raj Kumar (2016), 'A Review on Android Authentication System Vulnerabilities', International Journal of Modern Trends in engineering Research, IJMTER, (ISSN-2349-745),Volume 3, Issue 8, pp:118-123.
11. Desai S., Bedi R.K., Jagdale B.N., Wadhai V.M. (2011) Elliptic Curve Cryptography for Smart Phone OS. In: Abraham A., Lloret Mauri J., Buford J.F., Suzuki J., Thampi S.M. (eds) Advances in Computing and Communications. ACC 2011. Communications in Computer and Information Science, vol 191. Springer, Berlin, Heidelberg.
12. "Overview of Elliptic Curve Cryptography on Mobile Devices" , Ariel Hamlin, Professor Hescott, available at: https://pdfs.semanticscholar.org/e6dc/f250c18d37dfd380efe01f13ee4b24cef702.pdf.
Richard K. Ansah et al, Relevance of Elliptic Curve Cryptography in Modern-Day
13. Technology available at: https://www.researchgate.net/profile/Richard_Ansah/project/Elliptic-Curve-Cryptography-6/attachment/5a0435b4b53d2fed8ad45b70/AS:558739692285953@1510225331980/download/Publi.pdf?context=ProjectUpdatesLog.
14. Wendy Chou, April 2, 2013,Elliptic Curve Cryptography and Its Applications to Mobile Devices. Lecture Notes , available at: http://honors.cs.umd.edu/reports/ECCpaper.pdf 15. Muhammad Yasir Malik, Efficient Implementation of Elliptic Curve
15. Cryptography Using Low-power Digital Signal Processor, Feb. 7-10, 2010 ICACT 2010, https://ieeexplore.ieee.org/document/5440306/?part=1.
16. The Elliptic Curve Digital Signature Algorithm (ECDSA), Johnson, D., Menezes, A. & Vanstone, S. IJIS (2001) 1:36. https://doi.org/10.1007/s102070100002.
17. Aqeel Khalique Kuldip Singh Sandeep Sood, Implementation of Elliptic Curve Digital Signature Algorithm, International Journal of Computer Applications (0975 – 8887) Volume 2 – No.2, May 2010.
18. Padma, Bh., Raj Kumar, G.V.S., 2016. Design And Analysis of an Enhanced SHA-1 Hash Generation Scheme for Android Mobile Computers. International Journal of Applied Engineering Research (IJAER). Volume 11, Number 4 pp 2359-2363. ISSN:0973-4562.
19. O.S. Sury, (November 28, 2011). "Use of SHA-256 Algorithm with RSA, DSA and ECDSA in SSHFP Resource Records", Internet-Draft, available at: https://tools.ietf.org/id/draft-os-ietf-sshfp-ecdsa-sha2-02.html.
20. Data Encryption and Authentication Using Public Key Approach, SRINIVASAN NAGARAJ, Dr.G.S.V.P.RAJU, V.SRINADTH, International Conference on Intelligent Computing, Communication & Convergence, 2014,
Bhubaneswar, Odisha, India, Available online at www.sciencedirect.com.
21. Padma, Bh., Raj Kumar, G.V.S., 2017. Dynamic salt generation for mobile data security using elliptic curves against precomputation attacks, International Journal of Image Mining. Vol. 2, Nos. 3/4, pp 179-194. ISSN:2055-6047.
22. Alfred Menezes, Minghua Qu, Doug Stinson, Yongge Wang ,(2001, January) Evaluation of Security Level of Cryptography: ECDSA Signature Scheme.Available at:https://www.ipa.go.jp/security/enc/CRYPTREC/fy15/doc/1051_ecdsa.pdf
23. Erwin Hess et al, (2006, October). The Digital Signature Scheme ECGDSA, available at:http://www.teletrust.de/fileadmin/files/oid/ecgdsa_final.pdf.
24. V. Gayoso Martínez, L. Hernández Encinas, and C. Sánchez Ávila, A Survey of the Elliptic Curve Integrated Encryption Scheme, JOURNAL OF COMPUTER SCIENCE AND ENGINEERING, VOLUME 2, ISSUE 2, AUGUST 2010.
25. Nacy, Sinan & Oh, Tae & Leone, Jim. (2013). Implementation of SHA-1 and ECDSA for vehicular ad-hoc network using NS-3. RIIT 2013 - Proceedings of the 2nd Annual Conference on Research in Information Technology. 83-88. 10.1145/2512209.2512221.
26. Padma, Bh., Raj Kumar, G.V.S., 2018. A Novel Approach to Thwart Security Attacks on Mobile Pattern Authentication Systems. I. J. Computer Network and Information Security. Vol 10, N0 5, pages: 18-27. Published Online May 2018 in MECS (http://www.mecs-press.org/). DOI: 10.5815/ijcnis.2018.05.03. ISSN:2074-9104.
27. Padma, Bh., and Raj Kumar, G.V.S., 2018. Preventing Security Attacks on Mobile Pattern Passwords. Journal of theoretical and applied information technology. Vol.96. No 4. ISSN:1817-3195.

AUTHOR'S PROFILE

Dr G.V.S Raj Kumar is working as Associate Professor in the Department of Information Technology, GITAM Institute of Technology, GITAM(Deemed to be University) Visakhapatnam. He received his Ph.D in Computer Science & Systems Engineering from Andhra University in 2012. He has published over 31 International and 2 National journals and published one Book entitled "Model based Colour Image Segmentation Techniques and he has guided 3 Ph.D students and guiding 6 Ph.D scholars. He guided several students for getting their M.Tech degrees in Information Technology. His current research interests are information security, image processing. He is a life member of several professional bodies like computer society of India, Institute of Engineers, Indian Science Congress etc.

Smt Bh Padma is working as a Sr Assistant Professsor in the Department of Computer Sciences, GVPPG, Visakhapatnam-45. She has submitted her research from GITAM University under the supervision of Dr GVS Raj Kumar, Dept of IT, GITAM. Her area of specialization is Cryptography and Network Security. Till now she has published 15 research articles in various prominent peer reviewed journals and presented papers in various conferences in this research area.

Dr K Naveen Kumar is presently Assistant Professor in the department of Information Technology, GIT, GITAM University, Visakhapatnam. He presented several research papers in national and international conferences/seminars and journals of good repute. His current research interests include image processing and network security.