IJCRT.ORG ISSN : 2320-2882

INTERNATIONAL JOURNAL OF CREATIVE RESEARCH THOUGHTS (IJCRT)

An International Open Access, Peer-reviewed, Refereed Journal

SMART PRIVACY HANDLER

Pinal Hansora[1], Rajshree Bhavsar[2], Shweta Chougule[3], Tejal Chaudhary[4]

[1]*Assistant Professor Laxmi Institute Of Technology, Sarigam*

[2]*Laxmi Institute Of Technology* , Sarigam

[3]*Laxmi Institute Of Technology , Sarigam*

[4]*Laxmi Institute Of Technology , Sarigam*

***Abstract*— This paper presents an android based security application. The idea behind this project is to develop a lightweight Android app that enables users to apply a system lock or application lock, preventing access to the locked apps without password so that no other person can access the phone. This application provides three mode of operation, first mode is for accessing the locked app using a pin code, second mode is by using a tile lock and the last one is by using a gesture lock. These modes operate on choice method (i.e. the user can open his locked app using his pin code lock or tile lock or by gesture lock). Apart from securing the applications, Email alert feature is present for recovery of the password and intruder alert.**

***Keywords*—gesture, tile, passcode, lock, security, shared preference, intruder, mail.**

I. INTRODUCTION

Android is currently the most popular and widely used mobile operating system that runs on the Linux kernel which is specially designed for mobiles provides a flexible environment for Android Mobile Application Development. With the increased stride of life and shrinking time, nowadays people need everything in the palm of their hand, everything happening with a touch. Android supports enormous number of applications in Smart Phones. These applications make life of human beings more comfortable and advanced, so securing such Smartphone which contains number of applications is one of the major tasks and to make system secure we need to set a password to lock the apps. A password is a secret code that is used for authentication and also the most common method for identifying user which is supposed to be known only to the user. For example, if you have an applock on friendbook, one of your friend borrow your phone to play game or to make a call then he/she won't get into your friendbook app without a password you have set for the locked app. So, to make system more secure we have proposed a special mechanism for setting password. There are three ways for setting a password: first is by setting a four-digit numeric pin code, second is by setting tile lock and the last is by setting a gesture pattern as lock. It also has the option to hide the pattern draw or enable shuffled keyboard and can also enable as System lock. The proposed app also contains additional feature of intruder selfie and Mail alert in which the built-in camera takes a picture of anyone who tries to unlock your apps or for password recovery by clicking on forget password it sends to your email address.

II. SHARED PREFERENCE IN ANDROID

Shared preferences is one of the most Interesting Data Storage option Android provides its users is Shared Preferences. Shared Preferences is the way in which one can store and retrieve small amounts of primitive data as key/value pairs to a file on the device storage such as String, int, float, Boolean that make up your preferences in an XML file inside the app on the device storage. Shared Preferences can be thought of as a dictionary or a key/value pair. For example, you might have a key being "username" and for the value, you might store the user's username. And then you could retrieve that by its key (here username). You can have simple shared preferences API that you can use to store preferences and pull them back as and when needed. Shared Preferences class provides APIs for reading, writing and managing this data.

Shared Preferences is suitable in different situations. For example, when the user's settings need to be saved or to store data that can be used in different activities within the app. As you know, onPause() will always be called before your activity is placed in the background or destroyed, so for the data to be saved persistently we prefer saving it in onPause(), which could be restored in onCreate() of the activity. The data stored using shared preferences are kept private within the scope of the application. However, shared preferences are different from that activity's instance state[2].

How to Write Data in Shared Preferences

```
SharedPreferences sharedPreferences
   = getSharedPreferences("MySharedPref",
                          MODE_PRIVATE);

SharedPreferences.Editor myEdit
   = sharedPreferences.edit();

myEdit.putString(
   "name",
   name.getText().toString());
myEdit.putInt(
   "age",
   Integer.parseInt(
      age.getText().toString()));

myEdit.commit();
```

How to Read Data in Shared Preferences

```
SharedPreferences sh
   = getSharedPreferences("MySharedPref",
                          MODE_APPEND);

String s1 = sh.getString("name", "");
int a = sh.getInt("age", 0);

name.setText(s1);
age.setText(String.valueOf(a));
```

In our application we have used the shared preferences to store the data of all the three modes of locks (gesture, pass code, tile)

III. PASSCODE

You can set up a screen lock to secure your Android phone. Each time when user turn on his/her device or wake up the screen, he/she be asked to unlock the device with any of three modes PIN, TILE, or GESTURE. PASSCODE module is a default module in our application in which there will be 0-9 digits using that digit the user have to set a 4 digit pin code. Each time when user turns on his/her device to unlock, the digits will be randomly shuffled. As this module is set as default one if user forgot the passcode then he/she can click on the forgot and the passcode will be send to the given Email Id.
Algorithms to implement the pass code module we have use the SHA1- for storing the passwords.

SHA-1 (Secure Hash Algorithm 1) is a cryptographic hash function which takes an input and produces a 160-bit (20-byte) hash value known as a message digest – typically rendered as a hexadecimal number, 40 digits long. It was designed by the United States National Security Agency, and is a U.S. Federal Information Processing Standard[5].

ALGORITHM:

Initialize variables:

```
h0 = 0x67452301
h1 = 0xEFCDAB89
h2 = 0x98BADCFE
h3 = 0x10325476
h4 = 0xC3D2E1F0
```

ml = message length in bits (always a multiple of the number of bits in a character).

Pre-processing:
append the bit '1' to the message e.g. by adding 0x80 if message length is a multiple of 8 bits.
append $0 \le k < 512$ bits '0', such that the resulting message length in bits is congruent to $-64 \equiv 448 \pmod{512}$
append ml, the original message length, as a 64-bit big-endian integer. Thus, the total length is a multiple of 512 bits.

Process the message in successive 512-bit chunks:
break message into 512-bit chunks
for each chunk
break chunk into sixteen 32-bit big-endian words w[i], $0 \le i \le 15$

Message schedule: extend the sixteen 32-bit words into eighty 32-bit words:
for i **from** 16 to 79
Note 3: SHA-0 differs by not having this leftrotate.
w[i] = (w[i-3] **xor** w[i-8] **xor** w[i-14] **xor** w[i-16]) **leftrotate** 1

Initialize hash value for this chunk:
a = h0
b = h1
c = h2
d = h3
e = h4

Main loop:[3][56]
for i **from** 0 **to** 79
if $0 \le i \le 19$ **then**
f = (b **and** c) **or** ((**not** b) **and** d)
k = 0x5A827999
else if $20 \le i \le 39$
f = b **xor** c **xor** d
k = 0x6ED9EBA1
else if $40 \le i \le 59$
f = (b **and** c) **or** (b **and** d) **or** (c **and** d)
k = 0x8F1BBCDC
else if $60 \le i \le 79$
f = b **xor** c **xor** d
k = 0xCA62C1D6

temp = (a **leftrotate** 5) + f + e + k + w[i]
e = d
d = c
c = b **leftrotate** 30
b = a
a = temp

Add this chunk's hash to result so far:
h0 = h0 + a
h1 = h1 + b
h2 = h2 + c
h3 = h3 + d
h4 = h4 + e

Produce the final hash value (big-endian) as a 160-bit number:
hh = (h0 **leftshift** 128) **or** (h1 **leftshift** 96) **or** (h2 **leftshift** 64) **or** (h3 **leftshift** 32) **or** h4.

PASSCODE

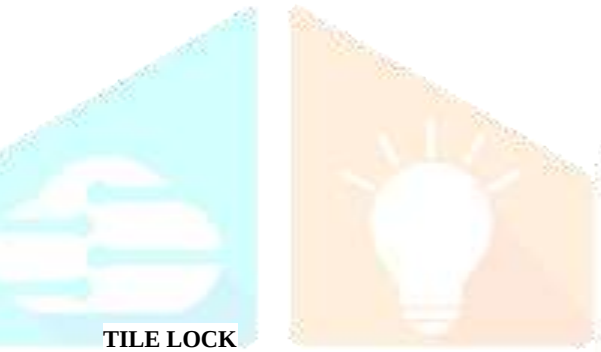
IV. TILE LOCK

In this mode of lock the user is provided with a unique UI, user can Lock/Unlock the device by setting the selected sequence of four colours in the order from the 3*3 tile panel which is filled by the nine colours. Each time when user turns on his/her device to unlock, the tiles will be randomly shuffled. While user using this mode to lock the device and forgets the sequence of the tiles the user can unlock the device by clicking on the forgot and the sequence of the tiles which was set by the user will be send to the Email id .

To store the combinations of tiles we have used a dynamic array and Shared Preferences concept. To shuffle tiles randomly we have used the shuffled tile algorithm.

```
Function shuffletileindex ()
{
For(i=Tile_index_length-1;i>0;i--)

        Index=random. Next (i+1)
        If(index not equal to i)
        then
                Tile index [index] ^=Tile index[i];
                Tile_index[i] ^=Tile_index[index]
                Tile_index[index]^=Tile_index[i]
        End if
End for
    }
```



TILE LOCK

V. GESTURE LOCK

Gestures are the new clicks which is any physical movement used by the user to activate a specific control within the design.

In this mode of lock the user can set any wallpaper from his/her gallery of the device or the default wallpaper or can have an instance click of photo. Once the user has selected the wallpaper, he\she have to set a specify motion on the screen then by drawing that particularly motion the user can unlock the device Or user can draw a special symbol, letters, shapes etc and add as gesture pattern on-screen. If user using this mode to lock the device and forgets the sequence of the tiles then user can unlock the device by clicking on forgot and automatically gesture will be sent to the email id which is set in the settings.

To store the combinations of images symbols shapes letters etc. We have used a Shared Preferences concept.

GESTURE LOCK

VI. INTRUDER SELFIE

In this feature the picture is taken automatically when someone enters an incorrect password/pin , tiles sequence or draws a wrong gesture on your phone after three trials then it automatically open the front camera to take a picture of the person who tries to unlock your Android mobile. After the picture is clicked then automatically the notification is send directly to the given Email Id with date and time and the record of picture is also saved in the gallery of the application to verify that someone has unlock your phone.

A) Notify on the email

To develop the feature, we have used the concept of PHPMailer.

The code library PHPMailer is used to send emails safely and easily by PHP code from a web server. To send emails directly using PHP code require a thorough knowledge of SMTP standard protocol and also the related issues and vulnerabilities about Email injection for spamming. It simplifies the process of sending emails and makes user very easy to use.

VII. CONCLUSION

In this study, we proposed an android application which contains three mode of operation: pin code lock, tile lock and gesture lock respectively. User can either apply pin code lock or tile lock or gesture lock as per his/her wish to apply system lock or application lock. There is also an E-Mail alert feature in which the built-in camera takes a picture of anyone who tries to unlock your phone and by clicking on forget password user can recover password. It sends the old password to the user's email address. This system will provide stronger security against the unauthorized user over user's personalized apps. This satisfies user's requirements where authentication is primary concern.

VIII. FUTURE WORK

In future work, the algorithm used in present application will be modified as the release of any new android version comes up. To add feature like hiding the app icon from the app list and set any fake icon, hiding photos and videos, preventing uninstallation of app without authenticating, hiding notification preview, etc.

IX. REFERENCES

[1] Sumaiya Patel, Darshana Thakur, Sujit Sherkar, Priyanka Dhamane, "Lockme-Android Security Application". International Journal Of Computational Engineering Research Vol. 3 Issue 3, 2013.

[2] Shared Preference: https://abhiandroid.com/programming/shared-preference.

[3] Shrutika S. Yande, Renuka C. Walimbe, "Image Password Based Authentication in an Android System", International Journal of

Computer Science and Mobile Computing, Vol.5 Issue.9, September- 2016, pg. 51-56

[4] ApplicationReference: https://play.google.com/store/apps/details?id=com.domobile.applockwatcher

[5] SHA-1 Preference: https://en.wikipedia.org/wiki/SHA-1

[6] Machiry, R. Tahiliani, and M. Naik, "Dynodroid: An input generation system for Android apps," in Proc. of FSE, 2013, pp. 224–234.

[7] J. Fierrez and J. Ortega-Garcia, "On-line signature verification," in Hand- book of Biometrics. A. K. Jain and A. Ross, and P. Flynn, Eds. New York, NY, USA: Springer, 2008, pp. 189–209.

[8] Martinez-Diaz, J. Fierrez, and J. Galbally, "The DooDB graphical password database: Data analysis and benchmark results," IEEE Access, vol. 1, pp. 596–605, 2013.