Time-based One Time Password generation with image as secret-key for 2FA

By : Shubh Srivastava, Swati Aarya, Shivang Sharma, Raunak Raj, Vivek Baibhaw

ABSTRACT

Security is the major concern in today's digital life, as the naive users are targeted by hackers many 2FA methods are in use for authenticating users. Also, there are few researches done on image-based authentication techniques following encryption schemes based on conventional cryptography or mathematical background.

Our approach is based on creating 2FA setup key for TOTP by using features of image provided by user. Unlike automatic shared secret key generation, here users themselves provide an image for setting up 2FA (like a token) which can also be used later as a security factor. This concept is chosen to provide user a selection for making shared secret key and to remember the image as a way to identify correct authenticator window, as it will display the user provided image during the TOTP entering phase to avoid fraud directing pages or entering the OTP elsewhere.

In case, the user feels that secret key is no longer a secret or shared to someone by mistake, he/she can change the image anytime to generate new key & have a fresh 2FA setup or can generate another key from the same image. Doing so will invalidate the previous TOTP generating setup.

The image can also be used to generate the backup codes (each for single use & shared to the user only) to skip authentication sometimes when the TOTP generator device is not present or to reset the two-factor authentication (from one of the provided backup codes) when that device is stolen or no more accessible. Our project's secret key generation phase works on the RGB-values of pixels of an image with some randomness. This approach is able to generate many keys with a single input image.

Contents

1. Introduction

The one-time password is a two-way authentication technique and hence secure one-time password generation is very important for 2FA. The current method of one-time password generation is time-consuming and consumes a lot of memory on backend servers. The 4-digit one-time password system with advanced deep learning approaches and faster computing can be possible to break. But now for many 2FAs, HOTP (hash-based one time password) or TOTP (time-based one time password) are in use. It works with a shared secret key between client & server (generally generated by the server & then shared to client), which will get combined with a counter (or UNIX time in case of TOTP) to make a hash then truncated to form the TOTP (keep changing after a specified time interval). The key forms with the URL of the user's profile on the particular application with some alterations in it. Currently various sectors are using different available data security methods. Whenever we sign into any of our online accounts, two-factor authentication methods rely on a user providing a password as the first factor and a second, different factor -- usually either a security token or a biometric factor, such as a fingerprint or facial scan, etc.

1.1 Problem Statement

To make an authentication system which uses a more convenient and user driven key for OTP generation unlike auto generated abstract keys for users. Once the 2FA set up is done, this system must be independent of any mobile or internet services, which means to use a synchronized counter (such as time) based OTP method.

1.2 Research Question

Need to Identify an appropriate method for the 2FA key generation process. Text-based (numeric) OTPs are widely used for authentications. Also many models are proposed for variations and evaluation in the 2FA system. Then, Image based authentication took place, where researchers moved on to the concept of Graphical Passwords (GP) which uses images instead of textual passwords and are partially motivated by the fact that psychological studies say that humans can easily remember images more than text. One-time passwords were mentioned for the first time by Lamport in 1981 to address the security weaknesses of a simple password system. Then, the concept of graphical passwords was originally described and proposed by Greg Blonder in 1996. Our research concept, up to some extent is a mixture of both, numeric & image based authentication methodology. Users will have a 6-digit numeric time-based password generator made from the image-based secret key.

1.3 Objective

The objective of this project is to generate Time-based One Time Passwords in a secure manner by providing an alternate method of making user interactive symmetric key other than automatic symmetric keys. A user can be authenticated even in low/no network conditions. In the last phase of 2FA set up, users should also be provided with a backup method in case the TOTP generator device is lost or damaged or not present at required time. This backup method is secure and can be used to access his/her account through another device and refactor the 2FA setup if needed. A quick response (QR) code for the generated key should be used to share the key to the TOTP generator app effectively. The project is about making Time-Based One Time Passwords (TOTP) which changes after a time interval, with the help of a token provided by the user as an input which will act as a temporary signature to generate a secret key. The key is then used to generate time-based OTPs every 30 seconds which can be used to authenticate user while login (once setup correctly). Some main objectives are listed below:

- To have 2FA system even in low or no network conditions
- Having a changeable key for the 2FA setup
- Replacement for standard SMS based OTP system
- Independent of mobile services
- Key depends on the user, not on profile URL.
- Backup method

1.4 Approach

1. To propose an algorithm which generates a 160-bits (recommended for 2FA) secure shared secret key with the help of some more convenient and user driven token rather than generating key from user's profile URL.
2. And also the key will be converted into a QR code to easily share it with the device that will be the TOTP generator for that user's profile. The generator device can be the same or different, on which our android application will generate TOTP from the key.
3. The key will then be passed to the standard HMAC-SHA1 algorithm to make a hash after combining it with a dynamic counter.
4. Truncate the output of the hash according to the digits required for the TOTP.
5. Once the key gets shared between client and server, step 3 & step 4 for TOTP generation are to be performed on server and client machine independently.
6. Then during the authentication phase the server application matches the user entered TOTP with the user's current TOTP (generated by server itself) to authenticate the user.
7. After first authentication, a proper backup method should be provided by the server application.

2. Literature Review

User authentication has now become a really important topic, many researches have been done to evolve the current system and make it more reliable & efficient. Following table shows the various researches done for two factor authentication, improving security and applying new techniques on existing methods to get better performance.

Author	Title	Study
S Sai Pavan, T Lohit Raja, Harini N	Multi Factor Authentication using IMEI encrypted color QR codes. (2018)	The focus of the paper is to design a multi-factor authentication scheme that uses color QR code that is capable of storing data in multiple layers. The system's security is enhanced by encrypting the secret entities with mobiles IMEI number which will be unique to a given handset and then generating color QR code.
Md Arif Hassan, Zarina Shukur, Mhd. Kamrul Hasan	An Improved Time-Based One Time Password Authentication Framework for Electronic Payments. (2020)	In this paper, they approach the Time-based OTP authentication algorithm with biometric fingerprints to secure an electronic payment. This algorithm uses a secret key exchanged between the client and the server. The shuffle of the TOTP approach better wear by screening the key as being a QR code.
Anmol Geer Bava	Speech based OTP system to prevent shoulder surfing. (2020)	The paper proposes a novel approach of speech recognition to the traditional OTP based system which increases the usability as well as the security of the OTP scheme. Although there have been many speech recognition modules produced, the proposed system comprises of the Google Speech recognition module which perfectly fits the requirements of the proposed authentication scheme.
Madjit Karimov, Malikovich, Zarif Khudoukulov	A method of efficient OTP generation using pseudorandom number generators. (2021)	OTP is generated on both sides at one time based on shared counter, such as, HOTP. In most cases, one-way hash functions are used but because of the weakness of the hash functions, they propose a method of creating one time password based on pseudorandom number generators (PRNG), which they are exist on many systems and programming languages as native.

In a research by Hanan Fahmy & Noha Elkhateeb the difference between text-based and image-based one time password is mentioned. First, the Text Based authentication proposed by several researchers as follows; Mulliner et al. analyze the security architecture of SMS OTP system and conclude that SMS OTP system cannot be considered secured. Donson et al. propose a challenge-response authentication system for web applications using QR code (Quick response code) which contains the challenge and website address. To Log Into the system, the user scans the QR code and software on the mobile phone calculates the response using the user's credentials. The response then submitted through the mobile phone network. However, this technique is vulnerable to online phishing attacks. Lee et al. present an authentication method using mobile OTP with the QR code. The proposed system relies on the certificate authority to verify the OTP. However, the proposed authentication system doesn't protect the user from online phishing attacks. Comparing the random number that appears on the computer screen with the one obtained from scanning the QR code doesn't add any security since the attacker can modify both values.
Then, as mentioned in section 1.2 Image based authentication took place, where researchers moved on to the concept of Graphical Passwords (GP) which uses images instead of textual passwords and are partially motivated by the fact that psychological studies say that human can easily remember images more than text. One-time passwords are mentioned for the first time by Lamport in 1981 to address the security weaknesses of a simple password system. Then, the concept of graphical passwords was originally described and proposed by Greg Blonder in 1996.

Some image-based 2FA schemes proposed by different researchers:

1. Mzohgan Azimpourkivi, Umut Topkara & Bogdan Carbunar introduce Pixie, a novel, camera based two factor authentication solution. User action of snapping a photo is sufficient for Pixie to simultaneously perform a graphical password authentication and a physical token-based authentication, yet it does not require any expensive, uncommon hardware. Pixie establishes trust based on both the knowledge and possession of an arbitrary physical object readily accessible to the user, called trinket. Users choose their trinkets and authenticate by presenting the same trinket to the camera.

2. Kalyanapu Srinivas & Dr.V.Janaki proposed a model for authentication in online transactions. A set of images are displayed from which he/she has to select any image of his/her choice. Then the registered user gets an OTP generated from the selected image to his registered mobile number. After entering the OTP by the user, it is checked on the server for successful authentication. Even if the attacker is able to get the user credentials by a forged website, he/she will not be able to cause any damage as the transaction is not complete without OTP which is only accessible to the valid user.

3. Marc proposed a conventional crypto-graphical scheme where images are used for compression. Initially the selected image is compressed and a part which is pointed out on the compressed image is embedded with the text message and then encrypted at the sender side. This encrypted message is transmitted to recipient through a secure channel. Later the decryption of the cipher text takes place retrieving the message and part of compressed image. Decompression of the retrieved image is done to get the original image which is compared for verification for successful completion of transaction.

4. S.M.Koon paper depicts that images are used to generate message authentication code (MAC) by using hash function. The generated hash is encrypted with the sender secret key and then appended to the image. To transform the data between two individuals in a more secure way using the recipient public key hash can be encrypted. The recipient calculates the hash from the received image. Extracts the appended hash in the received image and decrypts using private key. To perform verification the calculated hash and the extracted hash are compared. If any changes in the hashes, then the images are tampered otherwise authentic.

5. Neha Vishwakarma & Kopal Gangrade proposed an image-based time synchronized OTP generation method using SHA 512 & ECC to counter man in middle attack. In this approach the users register for the first time on the website, they are required to select a set of four images randomly from predefined large set of images. The system generates OTP using one of image selected at the time of registration with sha-512 encryption along with randomly selected text fields given at the time of registration. Then system selects first 8 characters of cipher text and encrypts it with ecc. Then produced OTP will be sent to users' email. When the user enters OTP for validation again it will be decrypted using ecc and matched for authentication. However, this approach has a drawback as the text and images are stored on HDD and this makes them easily cached.

6. Hanan Fahmy & Noha Elkhateeb's proposed RIBPA model framework depicts how OTP is generated by using the features of images. The proposed model framework consists of three main components which are Verification, OTP generation and Result. Firstly, the verification component which consists of registration process and accessing process, where the user will be required to register then he will access the system by logging in with his/her username and password. Then, OTP generation component which consists of images upload, images modifying and features extraction where the user has to upload three images which will be modified and their features will be extracted to get the required OTP. Finally, the result component where the user will get his OTP which will be used to be authenticated

3. Methodology

As described above many models are proposed for variations and evolution in 2FAs, also the difference between text-based and image-based OTP is mentioned above. Our concept up to some extent is a mixture of both, the user will have a 6-digit numeric time-based password generator made from the image-based secret key and that image should be remembered by the user to ensure correct login window rather than being trapped into attacks like phishing.

With auto-generated keys the user remains unknown about the proper authentication and just cares about filling OTPs when required. Attackers with their OTP theft mechanisms may generate similar looking pages for fraud authentication whose motive is just to get the TOTP once to access the user's account.

In our approach, we will use an image provided by the user to generate secret keys for setting up 2 Factor Authentication. This concept is chosen for enabling the users to select a token themselves for making a secret key and to remember the image (token) as an identification of the correct authenticator window, as in our project we will display the user provided image during the OTP entering phase to avoid fraud directing pages or entering the OTP elsewhere.

There are some image-based authenticators which display some random images on screen to choose as a password, or to make OTP from them which is then shared through an SMS. Some calculate the distances between pixel and origin (considering image as a 2D plot) then combine with random numbers to generate the secret key for OTP generation.

We have used RGB values of bitmap image pixels in diagonal (cross) like format. Starting from top left and top right, moving diagonally across pixels calculating the Red-Green-Blue pigments separately till end. Our system will also check (while processing the user provided image) the pixel's color pigment values to ensure a strong enough image for generating the key. This is also explained in further sections.

3.1 Steps to achieve objective

Steps involved in generation of the key:

- Input image from user
- Calculating R value sum, G value sum and B value sum from top-left & top-right corner moving diagonally downwards separately.
- R sum, G sum & B sum of first diagonals is added with R sum, G sum & B sum of second diagonal.
  Rres = R1 + R2, Gres = G1 + G2, Bres = B1 + B2
- Concatenate Rres, Gres, Bres with three-digit random numbers in between, and insert three single digit random numbers in front and at the end of the total string.
- This string is now encoded into base 32 format and then trimmed to 160 bits length.

Steps to generate TOTP from the key:

- The key formed at the server side is then shared with the client device through Quick Response code.
- Combine secret key and current (properly synced) UNIX time into a single 160 bits (20 bytes) hash with the help of HMAC-SHA1.
- The output is then truncated effectively to shorten the hash length.
- Truncated output again trimmed up to the required TOTP length.
- This output is valid for a 30 second time slot, after that a new TOTP is generated.

3.2 Detailed Procedure

Step-1 calculate sum of pixel's R, G & B values separately starting from top-left and top-right

```
For top-left => starting (0,0)
calculate red_sum1, green_sum1, blue_sum1
For top-right => starting (0,imgWidth)
calculate red_sum2, green_sum2, blue_sum2
totalRed = red_sum1 + red_sum2
totalGreen = green_sum1 + green_sum2
totalBlue = blue_sum1 + blue_sum2
```

Step-2 perform two checks for selection of appropriate image for key generation:

1. perform a check for sufficient digits in sum. For this we've taken 5000 as minimum limit. Each totalSum must be at least 5000. (min image size 20x20)
2. Perform a check that the user has chosen a strong image for key generation. An image is considered as strong for key if it has a mix of pixel values. While in a single solid color image (like completely black, white or any other color) pixel value doesn't change much.

We've taken 3 lists of size 50 to store 50 different values of R, G & B while moving on image diagonally. At last (end of diagonal move) following cases arises:

Case1: any one list has less than 50 elements, meaning in the whole diagonal there are not even 50 pixels with different values of that color (either R, G or B). **(image not accepted)** Eg-> only B values list has less than 50 elements, but R and G are mostly changing in every move throughout the diagonal.

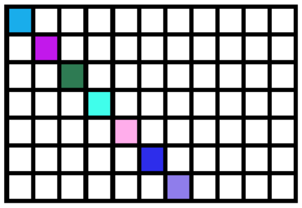
Case2: only one list has completed 50 elements, meaning only that color value is mostly changing throughout the diagonal move. **(image not accepted)** Eg-> only R values list has 50 elements in it (only red is changing mostly)

Case3: all lists have less than 50 elements, meaning RGB values are almost same for all pixels while moving diagonally. **(image not accepted)** Eg-> a solid red, RGB = 250 14 56

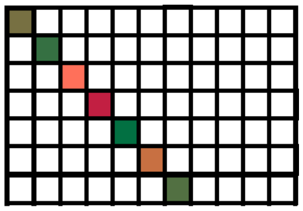
Case4: all lists have 50 elements, meaning pixel values are changing properly.**(accepted)**

Above four cases represented with the following diagram considering a very low size image for example only:

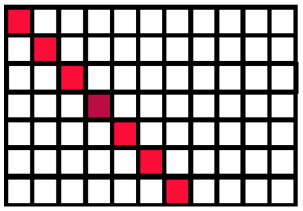
Case1



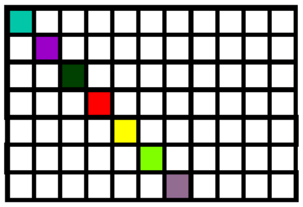
Case2



Case3



Case4



This check should be done for both diagonal traversals. After computation of both diagonal lists, **if ((d1.redList.size()==50 and d1.greenList.size()==50 and d1.blueList.size()==50) and (d1.redList.size()==50 and d1.greenList.size()==50 and d1.blueList.size()==50))** then the image is OK. Here d1 and d2 are two diagonal computation objects. This condition is well suited for normal sized images rather than for our example.

Sample images

Not accepted

Accepted

Step-3 generating some random numbers to put in between calculated toatalRed, totalGreen & totalBlue values. Output will be a long number string of min length = 20 digits
Generate six 1-digit random numbers, three of them will be appended at start and other three will be appended at last.
Generate two 3-digits random numbers, to place in between three calculated sums.

(1d1 1d2 1d3 totalRed 3d1 totalGreen 3d2 totalBlue 1d4 1d5 1d6)

1dn => nth 1-digit random number, 3dn => nth 3-digits random number

Step-4 encode the result of step-3 in base 32 format and trim the length of its output to 32 characters (where each character is of 5 bits). Output length will be 160 bit (20 bytes). This will be the required secret key, which will be then passed to HMAC algo with UNIX time to make TOTP.

Step-5 implementing **HMAC_SHA1** on the secret key and UNIX time, gives 20 bytes hash as output. This output will then be truncated to get the required TOTP.

Step-6 truncation phase, now we need an integer value as TOTP, so there is a need to truncate the 20 byte HMAC output to 4 bytes. Select an offset by getting the value of the last 4 bits from the 160 bits HMAC output.

int **offset** = **hmacHash [19] & 0xf** ;

This offset tells us the index in the hmacHash byte array from where we need to take 4 consecutive bytes. Its value ranges 0 to 15. Suppose if the offset value is 11, then we need to take byte 11, 12, 13 & 14. First bit of these 32 bits should be 0 to prevent getting a -ve value.

int **truncatedHash** = **(hmacHash[offset] & 0x7f)**<<24 | **(hmacHash[offset+1] & 0xff)**<<16 | **(hmacHash[offset+2] & 0xff)**<<8 | **(hmacHash[offset+3] & 0xff)**;

Step-7 now, apply modulo operation on the integer value to get the required number of digits in the TOTP.

int **TOTP = truncatedHash** % 1000000;

3.3 Project functionalities

Random pixels get selected from the input image and their RGB values are calculated, concatenated together, which would form a minimum 3-digit or maximum 9-digit number. To have a secure key (at least 128-bits or recommended 160-bits base 32 encoded key according to RFC 4226) some pseudorandom numbers (minimum 3 digits & maximum 9 digits) are also used with it. Same image does not generate the same key next time, it will be different every time. All this information collected randomly from the image will get encoded in base 32 and transferred to HMAC algorithm (SHA-1) which will make TOTP in every 30 seconds using Unix time. The user sets up the 2-factor authentication to his/her account by providing an image, in the next step a QR gets generated which is scanned by the device to which the TOTP setup needs to be done. After these steps, both (authenticator and generator app) will have the shared secret key of the user formed from the image. Now a check will be performed at last (first authentication) in which the user enters the TOTP generated by the app into the setup interface. If both matches in the timeslot, then the setup is correct and the 2FA for that user is now turned on.

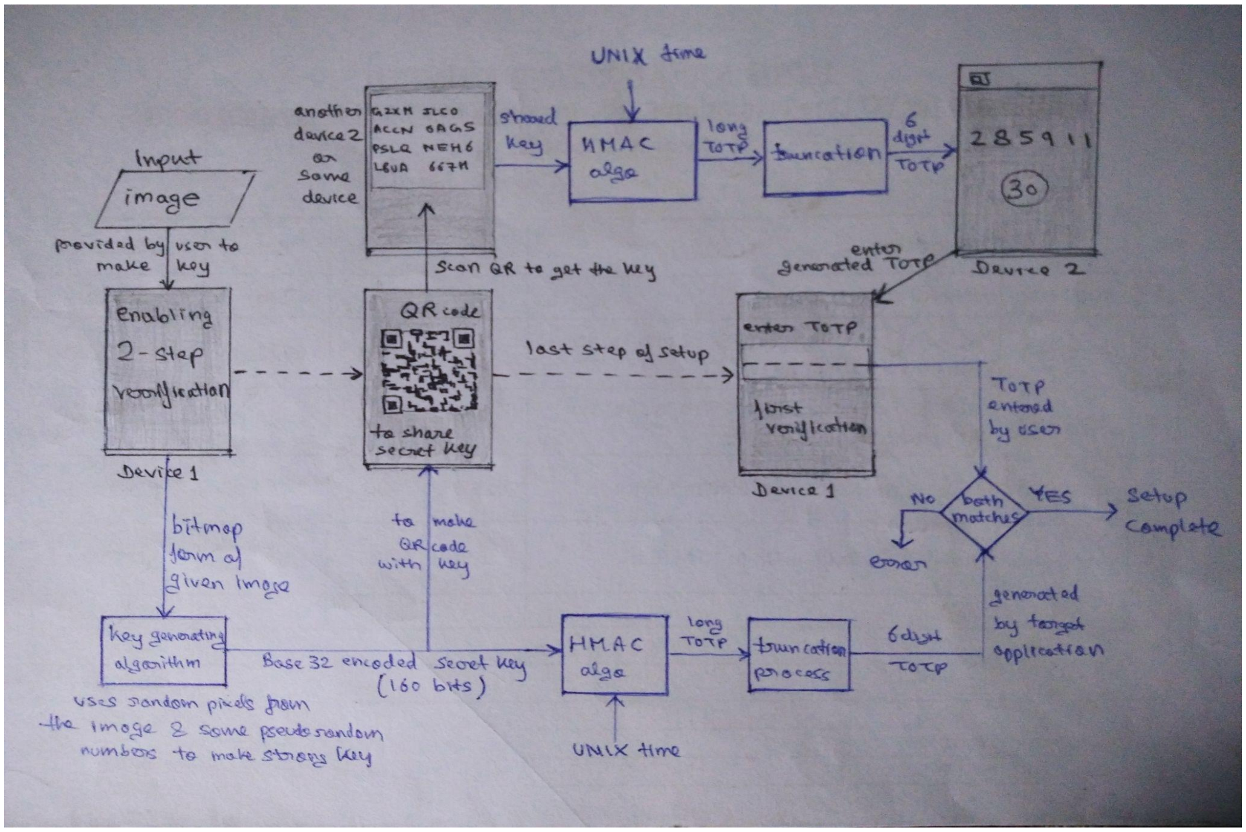
3.4 Additional features

The provided image is also displayed in the background like a watermark effect so that the user can identify he/she is on the correct authentication window, not on some attacker redirected site. This makes the complete process more secure and user interactive.

According to the study the image is found to be more relatable and remembered than the text or numeric code. The images are more appealing and in this case are more secure to be used as a backup method. The image used by the user as a backup can be stored on different locations and even can be shared with other devices without any suspicions or being noticeable by someone else unlike the backup codes.

This whole TOTP generation process works without any need of network connectivity or client-server interaction. A temporary app made with android studio is used to test the methodology and working. Enabling two-FA for users logged into the app, saving data on firebase about user and their 2FA conditions (whether enabled or not, or temporarily disabled). It contains a window to take input image and convert it into secret key represented as both text and QR code on screen. During the setup of 2FA, user is also provided with some backup codes for future use (made from that input image). The final step of the setup is first authentication through TOTP generated by a generator app (gets the key by scanning QR code). After the whole setup, on every login user needs to authenticate himself/herself by providing TOTP generated by the generator app. As server also have the key linked to each user, it will generate TOTP from server side also, if entered OTP matches with server-side OTP, the user gets authenticated.

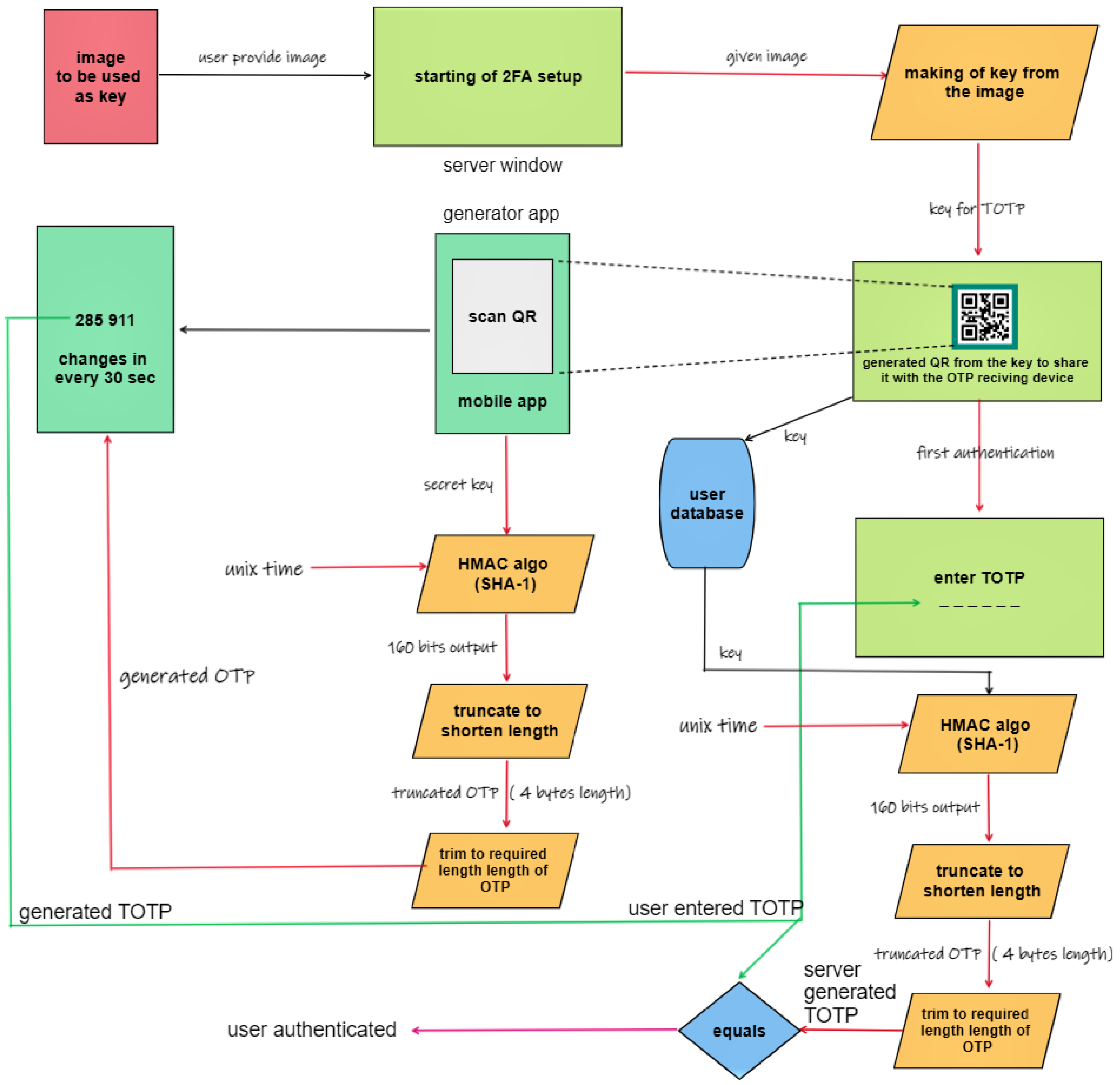
3.5 Project Design



3.6 Working

An application (say tempApp) implementing our methodology for the 2 Factor Authentication of its users, will need to get an input image from the user then follow the steps described in section 3.2. After the key generation, if the user wants to make the same device as TOTP generator, he/she should get any TOTP generator app (like google authenticator) and provide the key generated from the tempApp by copying it. In case user want to make any other device as a generator, a QR code is also provided to share the key to another device. Now when both the tempApp and generator app have the secret key, the process will move towards its first authentication to verify that the user has done the set up correctly as shown in above image. The generator app continuously generates the TOTPs for registered keys, independent of any services. Now if someone or user itself is trying to login his/her account to any other device, he/she after entering login id and password needs the current TOTP generated by the device having generator app with his/her registered key. During this the tempApp also generates the current TOTP with the user registered key, and checks if the user entered TOTP is matching with the server generated TOTP to authenticate the user login. By looking at this example we can see that we have achieved a network independent OTP generation for 2FA with an alternate method of key generation.

3.7 Flowchart



3.8 Project prototype

fig 1

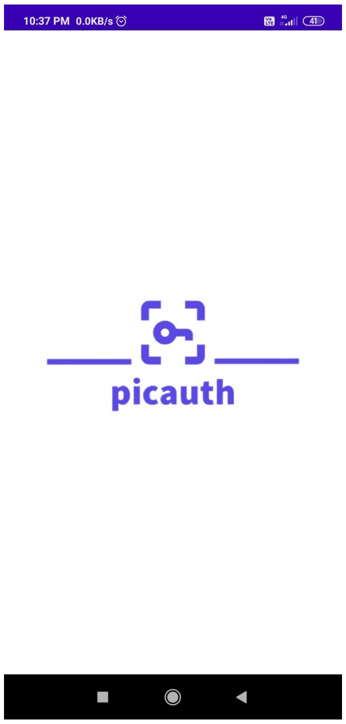


fig 2

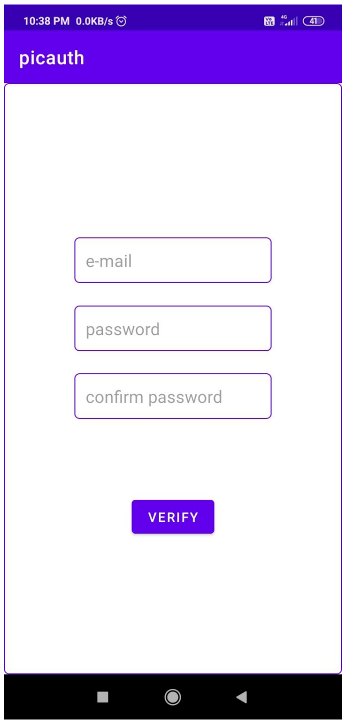
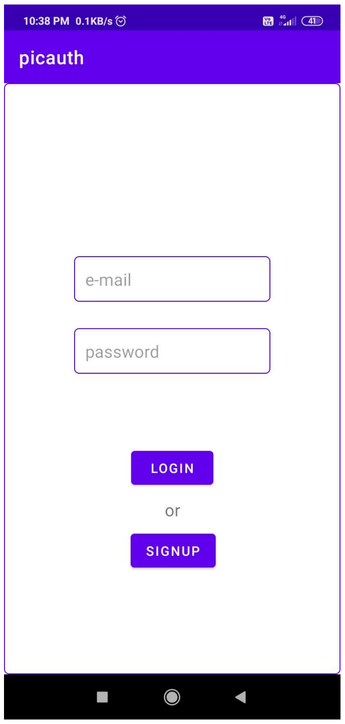
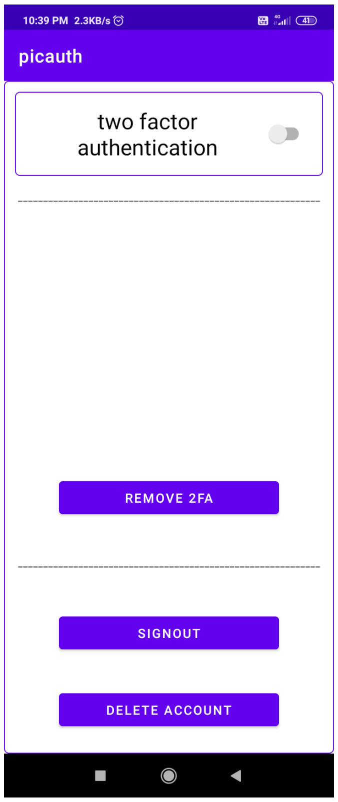
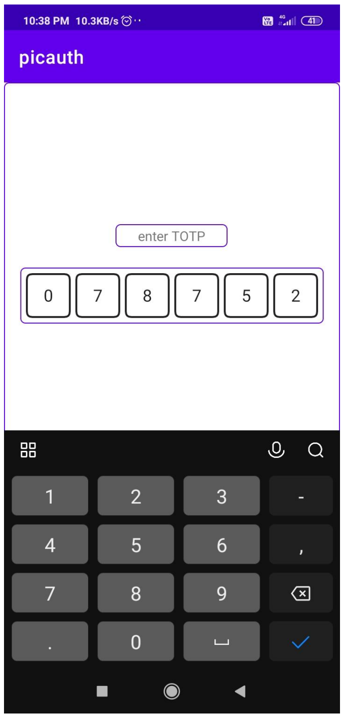


fig 3







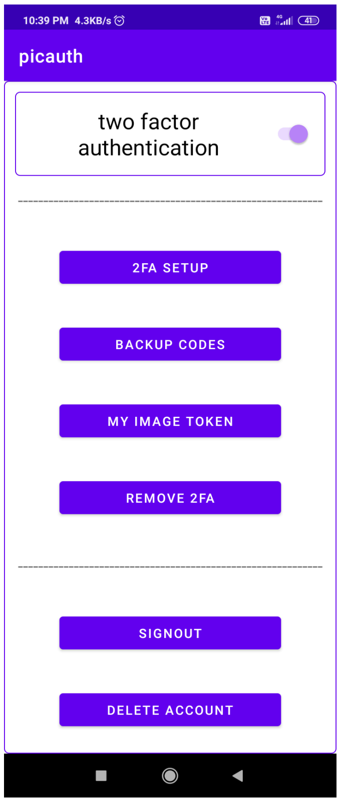
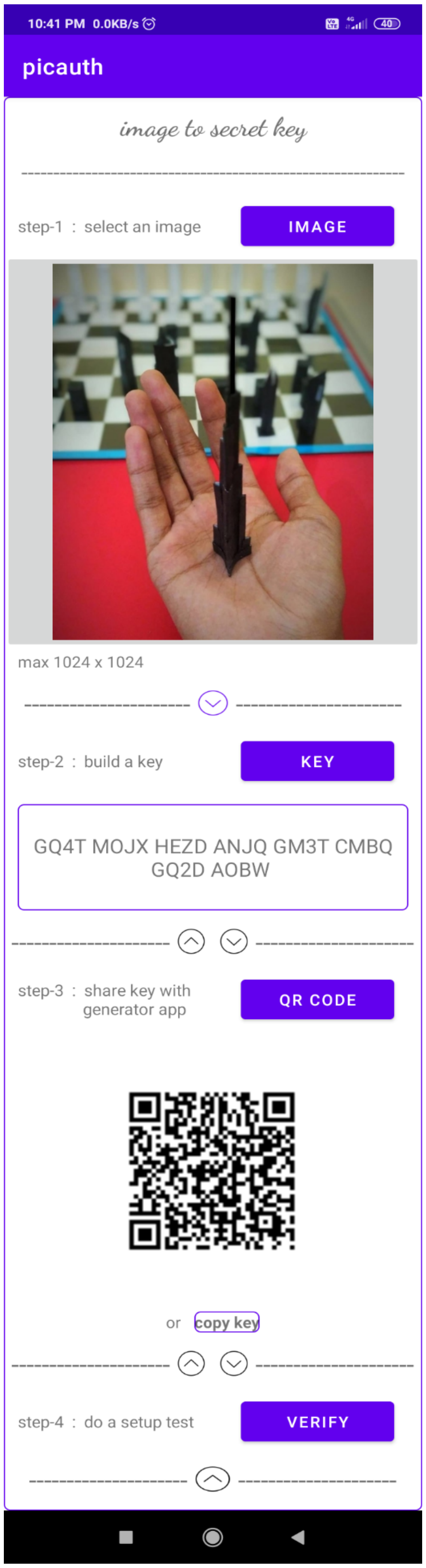


fig 4 fig 5 fig 6

2FA setup phase is divided into steps in the prototype app as shown.


10:41 PM 0.0KB/s

picauth

image to secret key

step-1 : select an image

max 1024 x 1024

step-2 : build a key

GQ4T MOJX HEZD ANJQ GM3T CMBQ GQ2D AOBW

step-3 : share key with generator app

or copy key

step-4 : do a setup test

4. Conclusion and future scope

Conclusion

In the end of this project we are able to generate the unique key with the help of an image provided by the user. The key is shared between client and server side. The complete process discussed above is executed on both sides and then TOTP is verified (TOTP generated by the server = TOTP entered by the user). The TOTP entered is valid for a short time frame only and after that new TOTP will be generated. If TOTP is verified then the 2FA will be successful and in future if the user wants to change the token, then new keys will be formed accordingly and again the whole process will be followed. We are also able to develop the proper and secure backup method that will help the user to access their account in case of any frauds or other activities (theft/lost/broken etc). This backup method works with the image that the user provides which makes it more interactive and sounding.

Future Scope

The current future scope of this project is to make the 2FA more safe and sound by implementing the advanced level of security. Moreover, the project can go to the next level by changing the backup method that is now working on the image itself to a backup method that works on the (biometric scan) eye detection of the user. This will make the process more autonomous, secure and robust. If the images used by the user is not stored somewhere else then this technology can work as a charm. Users can use the eye recognition feature for accessing their account in absence of the device having TOTP generator app.

Also as described in Step-3 of section 3.2, we are using some random numbers to alter the key if the user is using same image for the process. We can replace those programming language based random numbers generating methods with quantum computing. Where the nature of photons decides the random numbers. This can be done to achieve true randomness in the output and is impossible for a hacker to generate the user key if he/she got the user's token (image).

5. References

1. Approach for generation of OTPs using images by Kalyanapu Srinivasa & Dr.V.Janakib in 2016

2. Camera based two factor authentication (pixie) by Mzohgan Azimpourkivi, Umut Topkara & Bogdan Carbunar in 2017.

3. Proposed model for generation of OTP - research by Hanan Fahmy, Noha Elkhateeb in 2018

4. MultiFactor Authentication using IMEI encrypted color QR codes by S Sai Pavan, T Lohit Raja & Harini N in 2018

5. An Improved Time-Based One Time Password Authentication Framework for Electronic Payments – research by Md Arif Hassan, Zarina Shukur, Mhd. Kamrul Hasan3 in 2020

6. Speech based OTP system to prevent shoulder surfing – research by Anmol Geer Bava in 2020

7. A method of efficient OTP generation using pseudorandom number generators – research by Madjit Karimov, Malikovich, Zarif Khudoukulov in 2021

8. Generating MAC by using hash functions, a paper by S.M. Koon

9. Marc proposed a conventional crypto-graphical scheme where images are used for compression.

10. Neha Vishwakarma & Kopal Gangrade proposed an image-based time synchronized OTP generation method using SHA 512 & ECC to counter man in middle attack.

11. Google Authenticator and Authy app work documents & sheets

12. RFC 4226, an HMAC based One-Time Password algorithm