CHAPTER 4

A GRAPHICAL PASSWORD BASED AUTHENTICATION SCHEME

This chapter presents a proposed graphical password based Two-Factor Authentication scheme for web-based services. The scheme does not require a verifier table and therefore provides security against stolen verifier attack. The graphical password used in the proposed scheme is similar to other recognition-based methods where in the user need not remember a textual password, instead, he recognizes and selects previously chosen secret image to get authenticated.

The proposed scheme is novel as it provides strong authentication by way of addressing usability and security issues related to client, server and transmission channel security. It addresses the usability issues by employing graphical passwords. The security issues at client are addressed by employing smart cards and at server using the proposed scheme which resists the stolen verifier attack. Moreover, the transmission channel security is ensured by sending the encrypted message digest of the message, encrypted using server's public key.

The security analysis of the proposed scheme against the attacks discussed in section 2.6 is presented. Lastly, the formal analysis of the proposed scheme using Scyther Tool is presented along with the verification code.
4.1 PROPOSED SCHEME

4.1.1 Background

This section discusses how the proposed Two-Factor Authentication Scheme works. The idea of the proposed scheme is to provide three 3X3 image grids and allow the user to choose one image as secret from each image grid. At Login, the user has to correctly identify the pre-selected graphical password images and enter the corresponding number written with the image to get authenticated. It is also proposed that the users be allowed to upload their personal photograph set consisting of distinct images (say 27 in a set) at registration time. This will allow them to easily recognize their own personal photographs at login. Here the user should carefully select those pictures for upload that cannot be guessed for password entry by their close aides. It is also suggested that only low resolution images be used to ensure fast access. The advantage of using user’s personal photographs for their account is that it provides a very large password space. Figure 4.1 below shows a sample image grid consisting of users personal photographs.

The proposed scheme works as follows: If a user wants to register with an online service; he sends a registration request by clicking on the register link and entering an ID. Upon validating the received request, server sends ‘n’ images to the client which are displayed on three 3x3 grids one after the other (Figure 4.1 - Example Login Interface). The idea behind displaying images on 3 grids one after the other is to increase the
guessability and thus avoid the possibility of guessing attack.

Figure 4.1 Example Image Grid for Login

To register, the user has to choose at least one image from each grid by entering the corresponding image number. In the repetitive process the user must choose at least three images in order to complete the password selection process. At the end of password selection, the user would have selected 3 images as his graphical password for that account. The client now computes the user’s password as $PW_i = h(h(I_1)) \ XOR \ h(I_2) \ XOR \ h(I_3))$ and then sends $PW_i$ to the server for registration.

During login, after user’s login request is validated from server, the user is presented with a challenge response mechanism using images where in the user is challenged with at least three 3x3 grids of images. The user then recognizes his Image categories & enters the
The proposed method maintains an image database and user profile database at server which contains registered user’s profiles. The user profile table stores the unique user identity $h(\text{ID}_i)$ which is used to check the validity of a registered user and to display the user’s portfolio images accordingly. The user’s portfolio images consist of miscellaneous images besides those which the user had chosen as password. When a registered user tries to login, the $h(\text{ID}_i)$ entry will be verified and the portfolio images will be presented to him as challenge.

### 4.1.2 The Proposed Scheme

This section presents the proposed scheme which has three phases namely Registration phase, Login phase and Mutual Authentication & Session key agreement phase. The scheme uses simple hash functions for efficient computations, and random nonce to resist replay attack.

**Registration Phase**

It is recommended that the registration phase be executed in secure environment such as HTTPS. Whenever a new user wants to register with the system, he chooses an ID $i$ and proceeds as follows:

*Step R1:* $U_i$ submits registration request to $S$ by entering an ID.

*Step R2:* After checking the availability of received ID, $S$ sends images to the client. The client displays three 3x3 image grids one after the other.
Step R3: Ui chooses a password by selecting one image from each grid totaling to 3 images as his graphical password from the 27 displayed images. The client submits Pw_i to S along with the user’s portfolio images. The portfolio image set contains 27 images which includes the user’s secret images + other randomly chosen images by client from the displayed grid.

Step R4: Upon receipt of h(Pw_i), S computes the following:

\[ Q_i = h(ID_i | y_i); \]
\[ V_i = h(Pw_i | ID_i) \oplus Q_i; \]
\[ G_i = h(x \oplus h(ID_i)); \] and \[ H_i = h(Q_i) \]

Here \( y_i \) = concatenation of IP-address & current time

Step R5: S first creates a user profile which stores the user’s h(ID_i), h(y_i) and the portfolio images. The server then personalizes the smart card with \( \{G_i, V_i, H_i, y_i\} \) and delivers it to the user through secure channel.

Login Phase

Whenever a user wants to login to access his account, he inserts the smart card into the card reader and proceeds as follows:

Step L1: Ui requests for login.

Step L2: Upon receipt of the login request, the server sends the login page along with server’s Digital Certificate containing its public key.

Step L3: User enters his ID_i, the client computes \( H_i^* = h(h(ID_i | y_i)) \); checks whether \( H_i^* \) equals \( H_i \) (stored in smart card); If valid it generates a random secret ‘P_i’.

Step L4: It then computes \( R_i = h(ID_i) \oplus h(P_i) \); Encrypts \( R_i, P_i \) and \( h(y_i) \)
using server’s public key as $S_i = E_{KUs}(R_i, P_i, h(y_i))$ and sends $S_i$ to server.

**Step L5:** Upon receipt of $S_i$, server decrypts it using its private key as $D_{KRs}(R_i, P_i, h(y_i))$.

**Step L6:** The server then computes $h(ID_i) = R_i \oplus h(P_i)$. Checks the validity of $h(ID_i)$ and $h(y_i)$ and if it is valid, it computes $h(P_i+1)$;

**Step L7:** ‘$S$’ generates a random secret ‘$P_j$’ and retrieves the user’s portfolio image set.

**Step L9:** It computes $J_i = E_{h(P_i+1)}(\text{Images}, P_j)$, and sends $J_i$ to user.

**Step L10:** Upon receipt of $J_i$, the client Decrypts the Images as $D_{h(P_i+1)}(\text{Images}, P_j)$; and displays received images for password entry, This step is very crucial in resisting the Phishing attack as the attacker cannot reproduce the $h(P_i+1)$ computed using $P_i$ which is never transmitted in plain text form across the channel.

**Step L11:** $U_i$ enters his password $PW_i$.

**Step L12:** The client computes $V_i^* = V_i \oplus h(PW_i | ID_i)$; Checks if $h(V_i^*)$ equals $H_i$, if valid, it computes $C_i = h(G_i \oplus h(P_j+1))$;

**Step L15:** $U_i$ sends $C_i$, Server

**Mutual Authentication & Key Agreement Phase**

Upon receipt of $C_i$, the server computes the following:

**Step VI:** $G_i' = h(x \oplus h(ID_i))$; $C_i = h(G_i' \oplus h(P_j+1))$ and checks whether $C_i$ equals $C_i'$, If it does not hold, $S$ rejects the login request, else, both client and server proceeds to compute the session key as follows:

$$Sk = h(h(P_i+1) \oplus h(P_j+1) \oplus G_i)$$
From here on the communications between client and server are encrypted using this session key.

**Password Change Phase**

A registered user can change his password by sending password change request to the server; but for this, he has to first login to the server. If the login is successful, then the password change phase runs in secure environment meaning that all the communication between client and server is encrypted with the newly generated authenticated session key. Therefore, when the user submits a request for change password, the client performs the following steps:

**Step C1:** Client generates a random number \( P_k \) and encrypts it with session key as \( E_{Sk}(P_k) \).

**Step C2:** Upon receipt of \( E_{Sk}(P_k) \), server decrypts it as \( D_{Sk}(P_k) \) and checks the freshness of the nonce. If valid it creates \( P_k+1 \) and picks a random set of images for change password. It then encrypts these with \( Sk \)

**Step C3:** Server sends \( E_{Sk}(\text{Image Set}, P_k+1) \)

**Step C4:** Upon receipt of \( E_{Sk}(\text{Image Set}, P_k+1) \), the client decrypts it as \( D_{Sk}(\text{Image Set}, P_k+1) \) and checks the freshness of nonce. If valid, it displays the images

**Step C5:** User chooses a new password \( P_{w_i}^* \) as discussed in 4.2.1 and submits to client

**Step C6:** Client computes \( V_i' = h(P_{w_i}^* \mid ID_i) \oplus Q_i \); Replaces \( V_i \) (stored in smart card) with \( V_i' \)
Step C7: Client updates the contents of smart card as \{ G_i, V_i', H_i y_i \};

Step C8: It sends encrypted Portfolio Images and P_{k+2} to server

Step C9: Upon receipt, the server checks for the freshness of random number and then updates the user profile as discussed in 4.2.1

The important feature of proposed password change phase is that the client does all the required computations and updates the smart card. Thus, the computation and communication cost at the server is drastically reduced. Moreover, the user secret parameter V_i', which is important in password verification at client, is never transmitted over the channel, thus providing the security against interception.

The Registration, Mutual Authentication & Key Agreement and Password Change Phases are shown in figures 4.2, 4.3 and 4.4 respectively.

**Figure 4.2** Registration Phase
Client

User enters IDi, client computes Hi = h(IDi, yi) and checks if Hi == H; if it holds, it computes Ri = h(IDi) ⊕ h(Pi), Encrypts Ri, Pi and h(yi) using server's public key as Si = EKUs (Ri, Pi, h(yi))

Decrypts the Images using DhP+1(Images, Pj); displays received images.

User enters password, client computes V* = h(Pwi, IDi) ⊕ VI; Checks if h(V*) == H. If valid it computes Ci = h(Gi ⊕ h(Pj+1))

Computes session key SK = h(h(Pi+1) ⊕ h(Pj+1) ⊕ Gi)

Common channel

Server

Computes D_KRs (Ri, Pi, h(yi)); h(IDi) = Ri ⊕ h(Pi) ; Checks the validity of h(IDi) and h(yi)

If it is valid, it computes h(Pi+1). Generates random no. Pj; It then retrieves images and computes Ji = EhP+1(Images, Pj)

Gi’ = h(x ⊕ h(IDi)); Ci’ = h(Gi’ ⊕ h(Pj+1)); Checks if Ci’ == Ci, If it holds proceeds else terminates the session

Computes session key Sk = h( h(Pi+1) ⊕ h(Pj+1) ⊕ Gi)

Figure 4.3 Mutual Authentication & Key Agreement Phase

Client

User Submits Password change request by generating random no. Pk. Encrypts it with Sk,

Decrypts as E_Sk(Images, Pk+1) and verifies the freshness of nonce. Client displays images.

User Chooses password and submits. Client computes V’ = h(Pwi, IDi) ⊕ Qi; Replaces V with V’

Updates Smart card with {Gi, V’, H, yi}

Server

Decrypts it and Checks the freshness of Pk, Picks a random Image Set; Sends Images and Pk+1 in encrypted

Checks the freshness of Pk+2, Updates user profile

Session key encrypted Channel

Figure 4.4 Password Change Phase
4.2 SECURITY ANALYSIS

This section presents the security analysis of the proposed scheme against common attacks performed on smart card based schemes and graphical password based schemes.

Security against Reconnaissance Attack

In the registration phase of the proposed scheme, as the user’s registration request is sent to the server, the server in response sends images to the client for choosing the user’s password. Here, an adversary may try to gain information about the set of all images stored in the image database which are randomly picked and displayed at each registration request. In order to do so, he sends repeated requests with different ID’s to the server. But such an act will not help the adversary in gaining knowledge about the images because the server checks whether the requests are repeatedly coming from the same IP address and if so, it sends the same set of images at every request.

In the worst case, the adversary may send repeated registration requests with different ID’s from different systems to get the complete set of images in the database. Later on, he may try to use this knowledge to perform a guessing attack on a valid user account which is highly difficult for him as proposed scheme does not allow such attack even if the adversary is in possession of a smart card of a registered user because of local verification at smart card level.
Security against Replay Attack

Suppose that if an adversary intercepts the messages $S_i$, $J_i$ and $C_i$ during transmission and try to replay it at later time, the attack will fail because the intercepted parameters are computed using the random nonce values which are checked for freshness by the receiver (client / server). Moreover, since the nonce values are never transmitted in plain text form in steps L4, L9 and L15. Hence without the knowledge of $P_i$ & $P_j$, he cannot perform replay attack.

Security against Insider Attack

As all the user related sensitive information such as user ID, secret questions and other registration related information is stored as message digest, the scheme ensures the resistance against Insider Attack.

Security against Stolen Verifier Attack

As the scheme doesn’t maintain verifier table, it is secure against stolen verifier attack.

Security against Shoulder Surfing Attack

The proposed scheme requires the password image number to be entered in a password field (which shows xxxxxx pattern for the entered text). This method of password entry even captured on camera will not reveal the user secret. Hence, it is inferred that the proposed scheme is secure against the shoulder surfing attack. Moreover, the images are displayed randomly at each login; therefore the user might be entering
different values for the same image every time. In the proposed scheme, the sequence of entering the password string is not mandatory.

**Provision of User Anonymity**

During each login, the messages $S_i$, $J_i$ and $C_i$ exchanged in steps L4, L9 & L15 respectively are the message digests. Suppose if the attacker intercepts these messages to get the knowledge of the user ID$_i$, he will not be successful in doing so as the MD’s are irreversible. Moreover, if the attacker tries to perform offline guessing on $S_i$ and $C_i$ to derive ID$_i$, he will not be successful, as the computation of $S_i$ and $J_i$ requires the knowledge of server’s private key and $P_j$ generated at server respectively. Hence, the scheme provides user anonymity.

**Security against Server Spoofing Attack**

In step L9, the server communicates with user by sending $J_i$. So, to perform spoofing attack successfully on the scheme, the attacker has to create $J_i$ using $h(P_i+1)$. But since $P_i / P_i+1$ were never transmitted, the attacker will not have knowledge of $h(P_i+1)$. If the attacker creates $h(P_i+1)'$ and computes & sends $J_i'$ to the client, this will be easily detected at the client system during computations of Step L10. Hence the scheme protects the user from masqueraded server.

**Security against Guessing Attack**

In order to perform guessing attack, there are two ways. One way is to create a dictionary of all images available in the server and use it for
offline guessing. Creating a dictionary requires access to all images stored in the image database which is highly difficult task. Alternatively, if the user is asked to use his personal photographs for registration and login then it will be infeasible for the attacker to collect all users’ photographs to create a dictionary.

Another way of guessing is to intercept all images and messages transmitted during registration and login session of a user and then perform guessing attack to get the password. This is highly difficult to do because during registration, if the attacker intercepts the images and the login messages such as $S_i$, $J_i$ & $C_i$ transmitted in steps L4, L9 and L15, he has to now guess the correct image and compare it with the intercepted message. This attack will fail because $S_i$, $J_i$ are encrypted using server’s public key and random generated key $(h(p_i+1))$ respectively. So, the attacker, without decrypting the intercepted parameters, cannot compare the guessed results with these. Moreover, In order to decrypt it the attacker must have the knowledge of server’s private key and the random number which are highly difficult and impractical to get.

Security against Phishing Attack

In the scheme, before displaying the received images from the server, the client and server between steps L1 and L4 ensures that both are communicating with valid participants by checking the freshness of random numbers which were never transmitted in plain text over the
channel. Hence, it is implied that phishing attack does not work successfully.

*Explicit Key Authentication*

An authenticated Key Agreement scheme is said to provide Explicit Key Authentication if it provides both the (Implicit) Key Authentication and Key Confirmation.

*Implicit Key Authentication*

In the proposed scheme, neither the key nor the secret values used to compute the key are transmitted in plaintext over the channel. Therefore, there is no way that a third party could learn the key value being agreed on.

*Key Confirmation*

In Steps L4 to L15 of verification phase, the client and server mutually authenticate each other and also shares secret values $P_i$ and $P_j$. This shows that only client and server are in possession of the secret values which are later used in computing the session key. Hence, Implicit Key Authentication is achieved. Moreover, in step V1 both client and server assures that both possess the secret session key, this shows that Key confirmation is also achieved.
Security of Session Key

Known-Key security

The proposed scheme provides known-key security as; firstly, the key is never transmitted over the channel during agreement phase, thus providing resistance from interception. Secondly, the session key SK is a message digest which is computed as discussed in step V1 using $G_i$, $h(P_i+1)$ and $h(P_j+1)$. These parameters are also never transmitted over the channel. Suppose, in the worst case scenario, if the past session key was disclosed to the attacker, he still cannot derive it because the key is a message digest which cannot be reversed. This implies that, a new key cannot be generated using the past intercepted key.

Forward Secrecy

Suppose that the attacker has the knowledge of ‘x’ and he intercepted the earlier transmitted session as well. In order to compromise this session, the attacker must decrypt it using the session key used in that session. Since the session key in every session is a message digest computed using $G_i$, $h(P_i+1)$ (a nonce) and $h(P_j+1)$ (another nonce) and never transmitted over channel, the attacker cannot create the session key with the knowledge of only ‘x’. Hence, it is implied that the scheme achieves forward secrecy.
4.3 **EFFICIENCY ANALYSIS**

This section presents the efficiency analysis of the scheme in terms of computational cost, communication cost and the memory required for each user. The scheme uses Hash functions, XOR operations and Encryptions are used for computations at Client and Server.

**Table 4.1 Efficiency Analysis of Proposed Scheme**

<table>
<thead>
<tr>
<th></th>
<th>E1 (T_h)</th>
<th>E2 (T_r)</th>
<th>E3 (T_E)</th>
<th>E4 (T_x)</th>
<th>E5 (Bits)</th>
<th>E6 (Bits)</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Client</strong></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Registration Phase</td>
<td>4</td>
<td>0</td>
<td>-NA-</td>
<td>2</td>
<td>512</td>
<td>256</td>
</tr>
<tr>
<td>Login, Auth. &amp; Key Agreement phase</td>
<td>14*</td>
<td>1</td>
<td>2</td>
<td>9*</td>
<td>-NA-</td>
<td>512</td>
</tr>
<tr>
<td>Password Change Phase</td>
<td>5*</td>
<td>1</td>
<td>2</td>
<td>2</td>
<td>Nil, as the contents will be updated</td>
<td>540Kb for images and 256 bits for message</td>
</tr>
<tr>
<td><strong>Server</strong></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Registration Phase</td>
<td>5</td>
<td>0</td>
<td>-NA-</td>
<td>4</td>
<td>256</td>
<td>540 Kbits</td>
</tr>
<tr>
<td>Login, Auth. &amp; Key Agreement phase</td>
<td>6</td>
<td>1</td>
<td>2</td>
<td>5</td>
<td>-NA-</td>
<td>540Kb for images and 128 bits for message.</td>
</tr>
<tr>
<td>Password Change Phase</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>Nil, as the contents will be updated</td>
<td>540Kb for images and 128 bits for message</td>
</tr>
</tbody>
</table>

* It includes hash and XOR operations performed to get PW_i refer 4.2.1

E1: Number of hash computations (T_h)
E2: Number of Random no. generations (T_r)
E3: Number of Encryption / Decryption (T_E)
E4: Number of XOR and concatenation operations (T_x)
E5: Memory needed to store user credentials;
E6: Communication cost
It is assumed that the Pwi, nonce, output of hash functions and other computations etc. are all 128-bit long and the images are of 20 Kb each.

In registration phase, the client and server computes 4 and 5 hash functions with 2 and 4 XOR, concatenation operations respectively.

In Login, Authentication & key agreement phase, the client performs fourteen hash computations (including PW_i hash computations), two Encryptions (one asymmetric encryption and one symmetric decryption), one random number generation and nine XOR operations, whereas the server performs six hash, two encryption (one asymmetric decryption and one symmetric encryption), one random number generation and five XOR operations.

In password change phase, the client computes five hash functions and two XOR / concatenation operations, one random number generation and two encryptions, whereas the server computes only one encryption.

The memory needed to store G_i, V_i, H_i, y_i in smart card is 512 bits (4x128) whereas the memory needed at the server to store h(ID_i), h(y_i) is 256 bits (2 x 128). This value is excluding the size of the images stored in the image database. After the password change phase, there is no need for additional space as the updated V_i' after change password computation will be replaced by the current V_i stored in the smart card.
The communication cost of authentication includes the capacity of message transmission involved in the authentication scheme. From client to server the transmission cost is 256 bits (2 x 128) for ID_i & Pw_i in registration phase. During login, authentication & key agreement phase, the cost is 512 bits. From server to client, the capacity is 540 K (27x 20 Kb images) in registration phase and in login phase; it is 540 K (27x 20 Kb) for images and 128 bits (1x 128) for P_j. In password change phase, the cost involved in transmission of all messages including images and encrypted P_k from client to server is 540Kb for images and 256 bits for message, whereas the cost involved from server to client is 540Kb for images and 128 bits for message.

Thus, the above analysis shows that the scheme is efficient as most of the computations are based on hash functions which are considered to be efficient functions; hence, the scheme can be used with Java cards or any other second factor device for web based applications.

4.4 FORMAL VERIFICATION

This section presents the formal analysis including the source code of the verification in .spdl, and the screen shot of result. As per the discussion of section 2.3.3, the Agent model here consists of two agents i.e. ‘C’ – Client and ‘S’ – Server. Each agent performs the roles, therefore two roles i.e. ‘role C’ and ‘role S’ are defined. Role Event is role H which is defined to handle the intermediate computations. The analysis also modeled the adversary model considering that the adversary has
complete control of the network. Since Scyther checks for the freshness and synchronization by default, these attributes are also claimed in analysis. Also note that, this protocol requires public key encryption which is declared here as Ekus function as per the specification of the language.

```plaintext
usertype SessionKey;
secret k: Function;
const succ: Function;
const Fresh: Function;
const Compromised: Function;
const hash: Function;
const XOR: Function;
const plus: Function;
const compare: Function;
const Ekus: Function;
const Dkrs: Function;
const EncryptbyhashofPiplus1: Function;

protocol StarCompromise(H)
{
  // Read the names of 3 agents and disclose a session between them including corresponding session key to simulate key compromise
  role H {
    const ID,Pi,1,Yi,Pj,Images,Skey: Nonce;
    var S,C: Agent;
    read_!H1(H,H, S,C);
    send_!H2(H,H, (S,
      hash(XOR(hash(XOR(Skey,hash(ID))),hash(plus(Pj,1))), Ekus(Pi,hash(Yi),hash(ID)), EncryptbyhashofPiplus1(Images,Pj)) ) ) ;
    # claim_H3(C,Empty, (Compromised));
  }
}

protocol sam08(S,C)
{
  role S {
    const ID,Pi,1,Yi,Pj,Images,Skey: Nonce;
    read_1(C,S, Ekus(Pi,hash(Yi),hash(ID)));
    send_2(S,C, EncryptbyhashofPiplus1(Images,Pj));
    read_3(C,S, hash(XOR( hash(XOR(Skey,hash(ID)))
      ,hash(plus(Pj,1)) )))
    claim_S1(S,Nisynch);
    claim_S2(S,Niagree);
    claim_S3(S,Secret,Ekus(Pi,hash(Yi),hash(ID)) ) ;
    claim_S4(S,Secret,EncryptbyhashofPiplus1(Images,Pj) ) ;
  }
}
```
claim_S5(S,Secret,hash(XOR(hash(XOR(Skey,hash(ID))),hash(plus(Pj,1)))));

role C{

    const ID,Pi,1,Yi,Pj,Images,Skey: Nonce;
    send_1(C,S, Ekus(Pi,hash(Yi),hash(ID)));
    read_2(S,C, EncryptbyhashofPiplus1(Images,Pj));
    send_3(C,S, hash(XOR(hash(XOR(Skey,hash(ID))),hash(plus(Pj,1)))));
    claim_C1(C,Nisynch);
    claim_C2(C,Niagree);
    claim_C3(C,Secret,Ekus(Pi,hash(Yi),hash(ID)));
    claim_C4(C,Secret,EncryptbyhashofPiplus1(Images,Pj));
    claim_C5(C,Secret,hash(XOR(hash(XOR(Skey,hash(ID))),hash(plus(Pj,1)))));
}

const Alice,Bob,Eve: Agent;
untrusted Eve;
const ne: Nonce;
const kee: SessionKey;
compromised k(Eve,Eve);
compromised k(Eve,Alice);
compromised k(Eve,Bob);
compromised k(Alice,Eve);
compromised k(Bob,Eve);

Figure 4.5 Screen Shot of Verification Result