A Usable and Secure Two-Factor Authentication Scheme

Syed Akram a, Mohammed Misbahuddin b & G. Varaprasad a

a Department of Computer Science and Engineering, B.M.S. College of Engineering, Bangalore, India
b Centre for Development of Advanced Computing, Bangalore, India

Version of record first published: 05 Jun 2012


To link to this article: http://dx.doi.org/10.1080/19393555.2011.629340

PLEASE SCROLL DOWN FOR ARTICLE

Full terms and conditions of use: http://www.tandfonline.com/page/terms-and-conditions

This article may be used for research, teaching, and private study purposes. Any substantial or systematic reproduction, redistribution, reselling, loan, sub-licensing, systematic supply, or distribution in any form to anyone is expressly forbidden.

The publisher does not give any warranty express or implied or make any representation that the contents will be complete or accurate or up to date. The accuracy of any instructions, formulae, and drug doses should be independently verified with primary sources. The publisher shall not be liable for any loss, actions, claims, proceedings, demand, or costs or damages whatsoever or howsoever caused arising directly or indirectly in connection with or arising out of the use of this material.
A Usable and Secure Two-Factor Authentication Scheme

Syed Akram¹, Mohammed Misbahuddin², and G. Varaprasad¹
¹Department of Computer Science and Engineering, B.M.S. College of Engineering, Bangalore, India
²Centre for Development of Advanced Computing, Bangalore, India

ABSTRACT There are many secure authentication schemes that are secure but difficult to use. Most existing network applications authenticate users with a username and password pair. Such systems using the reusable passwords are susceptible to attacks based on the theft of password. Each scheme has its merits and drawbacks (Misbahuddin, Aijaz Ahmed, & Shastri, 2006). To overcome the susceptibility in the existing applications, there is an authentication mechanism known as Two-Factor Authentication. Two-Factor Authentication is a process used to authenticate or verify the identity of a person or other entity requesting access under security constraints. It is a system wherein two different factors are used in conjunction to authenticate. Using two factors as opposed to one factor generally delivers a higher level of authentication assurance. The proposed scheme allows users to freely choose their PassFile (file password) instead of remembering the password, eliminating the problem of entering the reusable password and remembering the password. In this scheme, we proposed an efficient scheme for remote user authentication. It does not maintain verifier table and allows the user to freely choose and change their passwords. The proposed scheme provides best usability for the user in terms of PassFile without changing the existing protocol. This approach uses a smart card and is secure against identity theft, guessing attack, insider attack, stolen verifier attack, replay attack, impersonation attack, and reflection attack. The proposed achieves the mutual authentication essential for many applications.

KEYWORDS authentication, hash functions, network security, privacy-enhancing technologies

The following nomenclature is used in this article: 

- \( U_i \): the user \( i \); 
- \( PW_i \): Password of \( U_i \); 
- \( S \): Remote system; 
- \( h(\cdot) \): one-way hash function using bit wise XOR operation; 
- \( A \Rightarrow B: M \): sends \( M \) to \( B \) through a secure channel; and 
- \( A \leftrightarrow B: M \): \( A \) sends \( M \) to \( B \) through a common channel.

Address correspondence to Dr. G. Varaprasad, Department of Computer Science and Engineering, B.M.S. College of Engineering, Bangalore 560 019, India. 
E-mail: varaprasad555555@yahoo.co.in

1. INTRODUCTION

With the rapid development of the Internet, more and more people are replacing traditional paperwork with computers, leading to more opportunities to access the resources at remote servers. In order to authenticate clients or users, password-based security mechanism have been widely used in many remote login systems because they can be easily implemented (Mishahuddin et al., 2006; Stallings, 2009; Schneier, 2005; National Institute of Standards & Technology, n.d.). The more services, the more username and password pairs

Downloaded by [varaprasad golla] at 03:23 29 June 2012
that a user needs to remember. A most common approach used to authenticate the identity of a user is a password.

Figure 1 shows the introduction of password authentication schemes. According to the characters of a password, it can be either a weak or strong password, as shown in Figure 1a. A weak password means the length of a password is short and easy to remember for users. A strong password means the length of a password is long and the elements are confused. A strong password cannot be found in a dictionary. Note that the server always needs to store a table of password for verifying the identity of the users. If security of the system is depending only on a password, the toughness of security is not enough. From Figure 1b it is evident that authentication can be done with the password table and without password table at the server (Horng, 1981). The medium for authentication can be done using a smart card, or without as in Figure 1c, and the password can be stronger by adding nonce and timestamp to the password, as in Figure 1d.

The smart card has the abilities of storage and computation. In 1981, Lamport first proposed a password authentication scheme by employing a smart card. Many other schemes were proposed following Lamport. However, these schemes have to maintain a verified password table and do not allow passwords to be changed freely. In 1995, Horng first proposed a novel idea that the server does not need to maintain a password table. Whether the schemes use the smart card or not, the following problems exist:

- The attacker invades the system.
- The attacker eavesdrops the communicated messages.
- The legal user accidentally reveals his password.

In a traditional authentication scheme, the server or the remote system must keep a password table to store all the passwords of the registered users in the system (see Microsoft, n.d.; RSA Laboratories, n.d.; Yampolskiy, 2007). Since password tables cannot be revealed and are usually large, it is hard and inefficient to maintain such tables. With password table at the remote system there is a threat of password being revealed to the outside world; to overcome this problem, in many schemes the password table has been replaced by a password verifier table. This table consists of the hashed value of each password instead of passwords in their plaintext form. Yet even the schemes with verifier tables are vulnerable to stolen verifier attack. Therefore, many schemes without verifier table have been proposed, but these are based on static login ID.

There are numerous applications where static ID leaks partial information about the user’s login message to the adversary. The adversary could intercept the login ID and later try to manipulate it to create a forged login message. Therefore, employing a dynamic ID for each login can avoid the risk of ID theft. Recently Das, Saxena, and Gulati (2004) proposed a dynamic ID-based remote user authentication scheme that does not maintain a verifier table and allows the user to freely choose and change passwords. They claimed that their scheme is secure against ID theft, guessing attack, replay attack, forgery attack, insider attack, and stolen verifier attack. However, Lal and Awasthi (2003) have pointed out several weaknesses in Das et al.’s scheme and have shown their scheme to be completely insecure. In addition, Ku and Chang (2005) have shown that Das et al.’s scheme is vulnerable to an impersonation attack in which an adversary can impersonate a valid user and successfully log onto the remote system. In 2005 Liao et al. (Wikipedia, n.d.) pointed out that Das et al.’s scheme is vulnerable to guessing attack and does not achieve mutual authentication to
overcome these weaknesses. Liao et al. proposed a few modifications to the Das et al. scheme; however, the enhanced scheme is still vulnerable to impersonation attack and has the weaknesses as described by Awasthi and Lal. Moreover, the scheme fails to achieve mutual authentication.

In 2006 the Misbah et al. scheme (2008) overcame the vulnerability of the Das et al. scheme. To make protocol more secure, they introduced mutual authentication, but the difficulty in this scheme is that the client and server machines need to capture the timestamp every time during authentication. They also must synchronize the timestamp at the client and the server timeservers, and it is difficult to manage the timeservers. A timeserver is a server computer that reads the actual time from a reference clock and distributes this information to its clients using a computer network (Wikipedia, n.d.). The time server may be a local network time server or an Internet time server. In the Misbah et al. scheme, it is required that the difference between the timestamps be calculated both at the client and server (Misbahuddin & Shoba Bindu, 2008).

In the proposed scheme, instead of taking the timestamp at the client and server, we generate a random number by taking the current timestamp at the client machine, which does not require any timeserver to be maintained. The usability which differentiates the proposed scheme and the Misbah et al. scheme is the introduction of a new concept called PassFile which eliminates the need to remember the password.

Section 2 deals with the two-factor authentication, and section 3 gives a brief overview of the Misbah et al. scheme. Section 4 provides detailed information about the proposed scheme. Section 5 gives a practical proof of concept implementation.

2. TWO FACTOR AUTHENTICATION

Two-factor authentication (T-FA) is a system where two different factors are used to authenticate a user. It adds more security to the system because the user must provide two factors of authentication, that is, password and passcode. The two-factor authentication combines something the user has (e.g., ID card, software token, cell phone, smart card), something the user knows (e.g., password, pass phrase, or personal identification number, file), and something the user is (e.g., fingerprint or retinal pattern, voice recognition, or another biometric identifier), which actually identifies the correct person that he claims to be.

Two-factor authentication means leveraging at least two of the authentication methods mentioned above. Using two factors as opposed to one factor generally delivers a higher level of authentication assurance. In order to gain access to specific resources, an unauthorized user or intruder needs to have access to two factors, that is, the secret codes (password) and the second factor (something the user has). T-FA could tremendously reduce the possibility of online identity theft because the victims’ secret passwords would no longer be sufficient for an intruder to access their information. However, it has been found that T-FA is still vulnerable to Trojan and man-in-the-middle attacks.

2.1. Authentication Types

Some of the common technologies used as the second-factor authentication in concert with user ID and password include:

- Tokens: A security token (or hardware token) is a small device that the user carries to authorize access to the network resources.
- Magnetic cards: Magnetic cards, for example, credit cards, debit cards, ATM cards, and smartcards, provide a possible two-factor authentication.
- Mobile phones: Mobile phones could be used as token devices using SMS messaging or an interactive telephone call.
- Biometrics: Biometrics include face recognition, voice authentication, and fingerprinting.

2.2. PassText: A Usable Authentication Scheme

There are many usable authentication schemes such as PassText, PassDoodle, PassMarks, and PassMaps, but here we will only discuss PassText (Wikipedia, n.d.). PassText is a novel approach to user authentication proposed by Roman et al. (2007), which addresses some of the limitations of current password schemes both graphical and textual. In PassText, instead of having the burden of providing the passphrase rest on the user’s shoulders, it is stored and readily available in the user authentication system itself. At the PassText creation stage, also known as the enrollment stage, users are presented with a large body of text to which they are
• Step 1: Computes operations: The following characters to shift one character to the any simple text string; deleting a character causes all restricted to being plain text. The PassText acts just like operation can be utilized. A resulting PassText is just a unique PassText. A user can delete whole paragraphs, move around sections of the text, replace words with different ones, and replace capitalization of individual characters. Basically any standard word processing authorizations with possible repetitions can be used to produce a unique PassText. A user can delete whole paragraphs, move around sections of the text, replace words with different ones, and replace capitalization of individual characters. Basically any standard word processing atomic modifications include deleting any character from the text or typing any character in any location.

Obviously a combination of the above modifications with possible repetitions can be used to produce a unique PassText. A user can delete whole paragraphs, move around sections of the text, replace words with different ones, and replace capitalization of individual characters. Basically any standard word processing authorizations with possible repetitions can be used to produce a unique PassText. A user can delete whole paragraphs, move around sections of the text, replace words with different ones, and replace capitalization of individual characters. Basically any standard word processing atomic modifications include deleting any character from the text or typing any character in any location.

3. EXISTING WORK

The existing scheme (e.g., Misbahuddin et al., 2006) consists of three phases, the registration phase, the authentication phase, and the password change phase.

3.1. Registration Phase

This phase is invoked when a new user \( U_i \) wants to register with the remote system.

• Step 1: \( U_i \) selects a password \( PW_i \) and submits \( h(PW_i) \) to the remote system through a secure channel.
• Step 2: \( S \) computes a nonce \( N_i = h(PW_i) \oplus h(\text{x} \oplus SID_i) \), here \( x \) is a secret key of the remote system, and \( SID_i \) is smart card’s unique identity.
• Step 3: Personalizes the smart card with the parameters \( h(\text{x}), N_i \), and \( SID_i \).
• Step 4: \( S \rightarrow U_i: \text{Smart card.} \)

3.2. Authentication Phase

Login phase: The user \( U_i \) inserts his smart card into the card reader, and keys his \( ID_i \) and password \( PW_i \). Then, the smart card performs the following operations:

• Step 1: Computes \( B_i = h(PW_i) \oplus N_i \)
• Step 2: Computes \( C_i = h(B_i \oplus Tu) \)
• Step 3: \( U_i - > S: (SID_i, C_i, Tu) \)

Verification phase: Upon receiving the login message \( (SID_i, C_i, Tu) \) at time \( T^* \), the remote system authenticates the user \( U_i \) with the following steps:

• Step 1: Verifies validity of the time interval between \( T_u \& T^* \). If \( (T^*-T_u) \leq \Delta T \), \( S \) accepts \( U_i \) ‘s login request, otherwise rejects the login request; here, \( \Delta T \) denotes valid time interval.
• Step 2: Computes \( B_i = h(x \oplus SID_i) \). Thereafter, the remote system computes \( C_i = h(B_i \oplus Tu) \) and compares it with the received \( C_i \). If it holds, the remote system accepts the login request. Otherwise, rejects the login request.
• Step 3: Now \( S \) computes \( D = h(B_i \oplus h(T_u)) \), where \( T_i \) is the current timestamp of remote system.
• Step 4: \( S - > U_i: (D, T_u) \).
• Step 5: Upon receiving the login-message \( (D, T_u) \) at time \( T^{**} \), if \( (T_u = T_i) U_i \) terminates the session, otherwise checks validity of the time interval between \( T_i \) and \( T^{**} \). If \( (T^{**}-T_i) \leq \Delta T \), then \( U_i \) computes \( D = h(B_i \oplus h(T_u)) \) and compares it with the received \( D \). If it holds the user confirms that the \( U_i \) is communicating with the valid \( S \).

3.3. Password Change Phase

This phase is invoked whenever the user \( U_i \) wants to change his password. He can easily change his password without taking any assistance from the remote system. The phase works as follows:

• Step 1: The user \( U_i \) inserts the smartcard into the smartcard terminal. He submits the password \( PW_i \) and request to change the password.
• Step 2: The user \( U_i \) then chooses a new Password \( PW_i^* \).
• Step 3: The smartcard computes
  \[ N_i^* = N_i \oplus h(PW_i) \oplus h(PW_i^*) \], whichyields
  \[ N_i^* = h(PW_i^*) \oplus h(x) \]
• Step 4: The nonce \( N_i \) will be replaced with \( N_i^* \). The password \( PW_i \) has been changed to \( PW_i^* \).

4. PROPOSED WORK

4.1. Introduction

The proposed scheme is an improvement over the Misbah et al. scheme (2006). This paper is an implementation paper of a proposed scheme. There are two corrections in the proposed scheme compared to Misbah et al.: are PassFile and Random Number.
• PassFile is a new usability feature which we have introduced in the proposed scheme. The PassFile is discussed in detail in section 4.2.
• The only difficulty in the Misbah et al. scheme is that the timeserver must be synchronized and maintained. It is a tedious job, which actually captures the current timestamp of client and the server for which timeserver (Yampolskiy, 2007) is required to synchronize the timestamp. The proposed scheme has come with an idea of generating the random number for each session instead of taking a timestamp. The random number is generated by taking the current timestamp $T_u$ of the client machine and giving timestamp $T_u$ as the seed value to generate the random number $R_i$. Here the timestamp $T_u$ is taken only from the client machine, not at the server machine for which timeserver is not required.

4.2. PassFile

In the proposed scheme there is no need to remember a password. We have developed the idea of PassFile, which is a normal text file or any file with different extensions (e.g., image file) but is limited in size to 200KB. The PassFile will act as our password (even though the need to remember the password is eliminated, the user needs to remember the file). The content of the PassFile can be the user’s own text written by the user himself. If the user forgets the file (PassFile) but remembers the content, then the user can regenerate the PassFile; this is an assumption.

At registration the user selects the PassFile; the client generates the message digest (one-way hash function is applied) of the PassFile which will be our password. If PassFile is stolen or is known to somebody, it is not secure. Our solution to this is that the user adds an extra two uppercase letters (special characters included), and again the one-way hash function is applied to the message digest generated earlier with the extra two characters. Now the generated message digest is our actual password, which is very strong (Microsoft, n.d.; RSA Laboratories, n.d.). Figure 2 shows the PassFile with its content, and Figure 3 shows the generated hashed password from the PassFile bmscse.txt.

4.3. Proposed Secure and Usable Authentication Scheme

There are three phases in the proposed scheme:

4.3.1. Registration Phase

This phase is invoked when a new user $U_i$ wants to register with the remote system as shown in Figure 4. The user $U_i$ requests the registration page with the server and a secure connection is set up. Then digital certificate is sent to the client’s browser. The user inserts a smart card and selects a PassFile $P_{Fi}$ (password $PW_{Fi}$ is generated from PassFile). A message digest $h(PW_{Fi})$ is taken, and an extra two uppercase characters (alphanumeric and special characters) are added to the generated message digest $b(PW_{Fi})$. For the second time, message digest $HPW_{Fi} = h(h(PW_{Fi}) + 2$ uppercase chars) is generated and submits $HPW_{Fi}$ to the remote system (server) through a secure channel with all other personal information.

Now the server computes a nonce $N_i$, which is calculated by applying XOR function and hash function with the parameters such as $x$ (secret key of the server) and $SID_i$ (smart card’s unique identity) and the password $PW_{Fi}$ (in Figure 4). Smart card is personalized with the parameters $b(\cdot), N_i$ and $SID_i$.

4.3.2. Authentication Phase

There are two phases in the authentication phase: login and verification.

Login phase: Similarly as in registration phase, user $U_i$ requests login page with the server and secure (SSL) connection is established, then digital certificate is sent to the client’s browser.
The complete process of the authentication phase is shown in Figure 5. The user $U_i$ inserts his smart card into the card reader; keys his $ID_i$ and selects file (PassFile), generating a password $PW_i$; a message digest is taken as $h(PW_i)$; the generated message digest $h(PW_i)$ is copied into a password box; and the extra two uppercase (or alpha numeric) characters are entered again with hash password in the password box, generating a message digest again as $HPW_i = h(h(PW_i))$. Then, the smart card computed by applying XOR function and hash function with the parameters such as $HPW_i$ and $N_i$ (in Figure 5).

After computing $B_i$, random number $R_i$ is generated at the client machine by capturing the current Timestamp $T_u$ and to generate the random number $R_i$, the timestamp $T_u$ is given as a seed value so that a random number is never repeated and similarly $C_i$ is computed by applying XOR function and hash function with the parameters such as, $B_i$ and $R_i$ (in Figure 5). After computing $C_i$, parameters such as $SID_i$, $C_i$, and $R_i$ are encrypted using public key $K_{pu}$ and sent to the server as $K_{pu}[SID_i, C_i, R_i]$ as shown in Figure 5.

**4.3.3. Verification Phase**

Upon receiving the login message $K_{pu}[SID_i, C_i, R_i]$, the remote system authenticates the user $U_i$. The server decrypts the received message using sever private key $K_{pr}$, such as

$$[SID_i, C_i, R_i] = K_{pr}[K_{pu}[SID_i, C_i, R_i]]$$

After decrypting the message, the server computes $B_i$ by applying the XOR function and the hash function with parameters such as $x$ (secret key of the server) and
SID; only as compared to Bi at the client. Thereafter, remote system computes Ci; similarly, taking parameters such as Bi and Ri. The message digest for Ri is taken separately by adding 1 to Ri such as h(Ri + 1), which ensures that even if Ri is compromised adversary cannot compute Di. The computed Ci; is compared with the received Ci. If it is equal, then the remote system accepts the login request; otherwise, the login request is rejected.

Regarding mutual authentication, the server computes Di by applying the XOR function and hash function with parameters such as Bi and Ri, and the server encrypts Di using server’s private key Kpr as Kpr [Di] and sends it to the user Ui. Upon receiving the login message Kpr[Di], User Ui decrypts the message using servers public key Kpu, such as Di = Kpu [Kpr[Di]] (in Figure 5). After decrypting Di, the user Ui computes Dii as computed by the server. If Di and Dii are equal, then the user confirms that the user is communicating with the valid server S. Kpu and Kpr are the public and private keys. After secure connection public key is shared with the client (user Ui) using digital certificate.

4.3.4. Password Change Phase

This phase is invoked whenever user Ui wants to change his password. He can easily change his password without any assistance from the remote system. User Ui inserts the smartcard into the smartcard terminal and selects the PassFile and generating PWi; the message digest is taken as h(PWi), and two uppercase characters are added to the h(PWi), generating HPWi = h(h(PWi)). The user Ui submits the HPWi, and requests to change the password and, in turn, requests to change the PassFile. The complete process of password change phase is shown in Figure 6.

The user Ui then chooses a new PassFile as PWi*, same procedures of generating message digest and adding extra two upper characters is carried and finally generating a message digest which is our password such as HPWi* = h(h(PWi*)). Then the smartcard computes new nounce Ni*, which is calculated by applying XOR
function only to the parameters such as \( N_i \), \( HPW_i \) and \( HPW_i^* \). The nonce \( N_i \) will be replaced with \( N_i^* \). The password \( HPW_i \) has been changed to new password \( HPW_i^* \) and the new nonce is stored in smart card as shown in Figure 6.

4.4. Security Analysis

The proposed scheme has the same security analysis as the Misbah et al. scheme, though there is much security analysis to be done. The security analysis is:

- **Secure against guessing attack**: It is extremely difficult for an adversary to retrieve the user’s password because we have our own PassFile which generates a password. It is impossible to retrieve using brute force attack, and it is difficult for the adversary to make the extra uppercase characters which makes our password very strong (Microsoft, n.d.; RSA Laboratories, n.d.) and also difficult to know the remote system’s secret key \( x \) from the intercepted parameters \( (SID_i, C_i, R_i) \).

- **Secure against insider attack**: In the registration phase, \( HPW_i = h(h(PW_i)+2 \text{ upper case chars}) \) are submitted instead of submitting the password in plain text form. Therefore, the password will not be revealed to the server. So, the insider attack is not possible.

- **Secure against impersonation attack**: In the proposed scheme, if an adversary intercepts \( K_{pu}[SID_i, C_i, R_i] \), it is difficult to know the private key of the server only the public key distributed through digital certificate. Hence it is very difficult to impersonate and the attack will fail in step 1 of verification phase.

- **Secure against replay attack**: A replay attack cannot work because it will fail at step 1 of the verification phase unless and until the adversary knows private key of the server also if in either case the adversary knows the private key of the server but replay attack is not possible as random number \( R_i \) cannot be used twice, whenever authentication is done we store the newly generated random number \( R_i \) in the database so that the same random number is used twice during authentication.

- **Secure against stolen verifier attack**: Since this scheme does not maintain verifier table, it is secure against stolen verifier attack.

5. PROOF OF CONCEPT IMPLEMENTATION

In this section we provide the implementation of the proposed scheme. We have tried to demonstrate the proposed scheme as a real time system except the use of smart cards, although we consider it as a mere simulation. The proposed research work is implemented as a Web application using Java servlet, JSP, and HTML. In this section we will explain as part of the implementation the research work in detail taking screenshot. As explained in the previous chapter, there are the three phases in the proposed scheme-Registration Phase, Authentication Phase, and Password Change Phase.

5.1. Registration Phase

The user requests a registration page, and the server sets up the SSL connection and sends the digital
The registration page is returned in response with the digital certificate. The digital certificate is generated by keytool command as shown below:

```
keytool -genkey -storepass bmscecse -keyalg RSA -alias 2FA -dname “CN=Syed Akram, OU=CSE, O=BMSCE, L=BENGALURU, ST=KARNATAKA, C=IN, EMAILADDRESS=bmscecse@bmscecse.in,”
```

after generating the digital certificate, the entry of the certificate is stored in server.xml as shown below.

```xml
<Connector port="8443" maxHttpHeaderSize="8192"
    maxThreads="150" minSpareThreads="25"
    maxSpareThreads="75"
    enableLookups="false" disableUpload
    Timeout="true"
    acceptCount="100" scheme="https"
    secure="true"
    clientAuth="false" sslProtocol="TLS"
    keystoreFile="D:\Tomcat 5.5\webapps\ROOT\KeyCA\2FA"
    keystorePass="bmscecse" />
```

We have used SHA-1 algorithm to generate the message digest and the code to generate message digest is shown below:

```java
public String computeDigest(byte[] b) {
    setAlgorithm("SHA-1");
    currentAlgorithm.reset();
    currentAlgorithm.update(b);
    byte[] hash = currentAlgorithm.digest();
    String d = "";
    for (int i = 0; i < hash.length; i++) {
        int v = hash[i] & 0xFF;
        if (v<16)
            d += "0";
        d += Integer.toString(v, 16);
    }
    return d.toString();
}
```

Similarly to calculate Exclusive OR(XOR), the code is shown below:

```java
static String calXOR(String ss,String ss1) {
    StringBuilder contents = new StringBuilder();
    byte b[]=ss.getBytes ();
    byte b1[]= ss1.getBytes ();
    // gets the higher length.
    int len = (b.length > b1.length) ?
        b.length : b1.length ;
    char res[] = new char[len];
    for(int i=0; i<len ; i++)
    {
        if(i < b.length )
        {
            if(i < b1.length){
                res[i]=(char)(b[i]ˆ b1[i]);
                contents.append (res[i]);
            } else {
                res[i]=(char)(b[i]ˆ 0);
                contents.append (res[i]);
            }
        } else {
            res[i]=(char)(0ˆ b1[i]);
            contents.append(res[i]);
        }
    }
    return contents.toString();
}
```

These two code snippets `computeDigest` and `calXOR` are the main methods to compute message digest and to calculate XOR and it is mandatory in our implementation.

In the password-based scheme during the registration phase, the user needs to enter the password two times to confirm whether the passwords are the same; similarly, taking our proposed work we will not encourage the selection of the PassFile two times because we are taking the message digest of a PassFile which always generates the same Hash code whenever and where we select a PassFile (it is an assumption that it is not necessary to select PassFile two times because of the fact explained above). Figure 7 shows the filled registration form, and we can see that the user is prompted to select a PassFile (in Figure 7, the user is selecting a PassFile called bmscse.txt).

Figure 8 shows the successful creation of the user account. Parameters such as $SID_i$ and $N_i$ are generated and personalized (stored) in a smart card (in case the smart card is not used, the parameters are stored in a file and used for authentication). We are storing these
parameters in a database or file. In our proposed work, the user is authenticated either using a database or a file (temporary solution). As soon as the user account is created successfully, the user has an option to change the password by clicking the Change Password link which navigates to password change page (as shown in Figure 11) and logout option as shown in Figure 8.

5.2. Authentication Phase

The authentication phase has two phases: login and verification. In the login phase the user requests the login and the server sets up the secure SSL connection; a digital certificate is sent to the browser as explained above.

The sign-in page is where the user inserts the smart card into the smart card reader (in our implementation we used database and a file) and selects the PassFile (not shown). The message digest is computed and the hashed message digest is copied into the password box, plus the extra two uppercase characters are added to the message digest generated, and the password is hashed again. The parameters such as \( S\!I\!D_i \) and \( N_i \) are read from the smart card and a random number \( R_i \) is generated. The parameters \( B_i \) and \( C_i \) are computed using random variable \( R_i \) and hashed password \( H\!P\!W_i \) and Nonce \( N_i \) as shown in Figure 9. The \( B_i \) is computed as follows:

\[
B_i = H\!P\!W_i \oplus N_i
\]
The $R_i$ is computed by taking the current timestamp $Tu$ at the client system and $Tu$ is given as a seed value to generate $R_i$. Next the parameter $C_i$ is computed as follows:

$$C_i = h(B_i \oplus R_i)$$

(2)

The computed parameters $B_i$, $C_i$ and $R_i$ are encrypted using public key $K_{pu}$ of the server. It is present in the digital certificate and submitted to the server in the form as shown below:

$$K_{pu}[SID_i, C_i, R_i]$$

(3)

In the verification phase, the encrypted parameters $B_i$, $C_i$ and $R_i$ are decrypted using the private key $K_{pr}$ of the server. The parameters $B_i$ and $C_i$ are computed as follow:

$$B_i = h(x^{\oplus SI_D_i})$$

(4)

Thereafter, the remote system computes $C_i$ as follows:

$$C_i = h(B_i^{\oplus} R_i)$$

(5)

Equations (2) and (5) are compared; if they are equal, the server accepts the login request as shown in Figure 10. Now the server computes the parameter $D_i$ for mutual authentication as given below:

$$D_i = h(B_i^{\oplus} h(R_i+1))$$

(6)

Then $D_i$ is sent to the client for mutual authentication. $D_i$ is encrypted using private key $K_{pr}$ as:

$$S => U : K_{pr}[D_i]$$

(7)

If equations (2) and (5) are not equal, then the server rejects the login request as shown in Figure 11. On receiving the login message $K_{pr}[D_i]$ at the client machine the encrypted message is decrypted using servers public key $K_{pu}$ as $D_i = K_{pu}[K_{pr}[D_i]]$. The client computes $D_i$ as:

$$D_i = h(B_i^{\oplus} h(R_i+1))$$

(8)

$D_i$ is compared with the received $D_i$. If equations (6) and (8) are equal, the user confirms that the user is communicating with the valid server as shown in Figure 10; otherwise, the user is not valid (not shown).

### 5.3. Password Change Phase

The password change phase is invoked when a user wishes to change his PassFile. It does not require any support from the server in changing the password;
that is, the need to communicate with the server is eliminated. This phase works as follows:

The user inserts his/her smart card into the smart card reader (in our case we read the parameters either from database or file) and selects an old PassFile. Similar to all other phases, a message digest is taken, plus two characters are added. Again the message digest is taken which is our password $HPWi$ and authenticates the user and request to change the PassFile.

The user chooses a new PassFile, and the same procedure of generating message digest, adding extra two uppercase characters, and finally generating a message digest as follows:

$$HPW^*_i = h(h(PWi*))$$ \hspace{1cm} (9)

After computing both old and new passwords $HPWi$ and $HPW^*_i$, the smart card computes the new nonce $N^*_i$ replacing $N_i$ as given below:

$$N^*_i = N_i \oplus PW^*_i \oplus HPW^*_i$$ \hspace{1cm} (10)

$$N^*_i = HPW^*_i \oplus h(x \oplus SID_i)$$ \hspace{1cm} (11)

Now new nonce is $N^*_i$ and password is changed to $HPW^*_i$ as shown in Figure 12.
6. CONCLUSIONS

In the treacherous Internet environment, each one’s identity and some secret information are easily copied and forged. Today, single factor authentication, for example, passwords, is no longer considered secure in the Internet and banking world. Easy-to-guess passwords, such as names, age, and date of birth, are easily discovered by automated password collecting programs. Two-factor authentications have been introduced recently to meet the demand of organizations for providing stronger authentication options to their users. In most cases, a hardware token is given to each user for each account. The increasing number of carried tokens and the cost the manufacturing and maintaining them are becoming burdensome on both the clients and organizations. Here it is possible to increase both simultaneously through careful design that considers usability and security in combination. We emphasize the need for thorough usability and security evaluations because system design can significantly impact user behavior, sometimes in unanticipated ways, which in turn can significantly impact the security of a system.

In our survey, the idea proposed is a dynamic ID-based remote user authentication scheme which is usable and secure. The proposed scheme is secure against insider attack, impersonation attack, replay attack, ID theft, guessing attack, and stolen verifier attack. Moreover, the scheme achieves mutual authentication. Here in our scheme instead of remembering the pincode (password) the user have to remember only PassFile (file) which in turn used as password with extra two upper case characters. Our scheme is very much secure, and the password created from file makes the brute force attack impossible.

Future work of the proposed scheme is that Passfile can be selected not through the file selector window but by just typing the (PassFile) file name which will be typed in the form of password dots not in a normal text in small CSS window which automatically selects the PassFile where the shoulder surfing attack can be avoided.

REFERENCES

BIOGRAPHIES

Syed Akram received a B.E. in computer science and engineering from M.S. Engineering College under Visvesvaraya Technological University, Belgaum, in 2006. He is pursuing a M. Tech in computer science and engineering from B.M.S. College of Engineering, Bangalore.

Mohammed Misbahuddin received his B. Tech in computer science & engineering from K.B.N. College of Engineering, Gulbarga University, Gulbarga, Karnataka, India, in 2001, and M. Tech in Software Engineering from JNTU, Anantapur, India, in 2006. He is pursuing a Ph.D. in computer science and engineering from JNTU, Hyd, India. His areas of interest include network security, image processing, pattern recognition, and software engineering. He is working as a senior staff scientist in Centerfor Development of Advanced Computing (CDAC), Electronic City, Bangalore, India.

G. Varaprasad obtained a B. Tech in computer science and engineering from the Sri Venkateswara University, Tirupati, in 1999. He received his M. Tech in computer science and engineering from BMSCE, Bangalore, in 2001, and a Ph.D. in computer science and engineering from Anna University, Chennai, in 2004 under the guidance of Dr. R. S. D. Wahidabanu. He worked on the routing algorithms for mesh network and mobile ad hoc networks. In 2004 he joined with Protocol Engineering Technology(PET) Lab, Department of Electrical Communication Engineering, Indian Institute of Science, Bangalore, as a postdoctoral fellow under the guidance of Dr. P. Venkataram. Since 2005, he has been with B. M. S. College of Engineering, Bangalore, in the Department of Computer Science and Engineering, where he is now assistant professor under PG studies (M. Tech course).

S. Akram et al.