*Universal Journal of Computer Science and Engineering Technology*
*1 (2), 133-140, Nov. 2010.*


# Eliminating Vulnerable Attacks Using One-Time Password and PassText – Analytical Study of Blended Schema

M. Viju Prakash[1], P. Alwin Infant[2] and S. Jeya Shobana[3]
1. Senior Lecturer, Department of Computer Science & Engineering, Sardar Raja College of Engineering,
2. Lecturer, Department of Computer Science & Engineering, Sardar Raja College of Engineering,
3. Lecturer, Department of Computer Science and engineering, Francis Xavier engineering College.
E-mail:- vijuprakash@ymail.com, alwininfant@hotmail.com, sofismiles@rediffmail.com

***Abstract*** **- A secure network partially depends on user authentication and unfortunately authentication schemes used at present are not utterly secure. Some passwords are not computationally dominant, where brute force attacks on this unprecedented scale became potential. Here we have designed a combined schema of One Time Password (OTP) algorithm concatenated with PassText which makes uncomplicated to commit to memory and is computationally powerful. It can be fairly and rapidly provided to the system, while at the same time remaining impractical to break the brute force attack. OTP algorithm powered with user's unique identifications like International Mobile Equipment Identification and Subscriber Identification Module; makes a finite alphanumeric token valid for a session and for a single use. PassText is an easy way of system authentication schema which enables the user not obligatory to memorize any difficult passwords or character combinations. Concatenation of these two schemas gives maximum security for authentications and almost impossible to break. We have also proposed a novel measure of security levels of many popular authentication schemas against the one we proposed.**

**Keywords: Brute force, OTP, PassText, Token, Security.**

## I. INTRODUCTION

Most of the systems today rely on static passwords to verify the user's identity. However, such passwords come with major management security concerns. Users tend to use easy-to-guess passwords, use the same password in multiple accounts, write the passwords or store them on their machines etc. Furthermore, hackers have the option of using many techniques to steal passwords such as shoulder surfing, snooping, sniffing, guessing etc.

This blended measure is also having disadvantages which include the cost of purchasing handheld devices like mobile, issuing and managing tokens or cards. From the customer's point of view, using more than one authentication system requires carrying multiple tokens/cards which are likely to get vanished or stolen. These are some limitations in this approach.

## II. RELATED WORKS

Researches on various types of attacks for secure networks have been published and here we have described the relevant one in terms of techniques they apply and the source of data they analyze.

Table I sums up the features of correlated works. Each one is classified according to the type of network and data attacks. Few network attacks like Eavesdropping [1], DoS and DNS poisoning are not modifying the original data in network communications. These attacks focus on monitoring data movement, routing and listening data paths. Data modification, IP address spoofing, Man-in-the-Middle [2] and Sniffer attacks are best known examples for confidentiality problems, where as they try to modify the source data sent by a station or transmitter.

Table I
Features of Related Works

| Attack types | Modification in data | Associated problems |
|---|---|---|
| Eaves dropping | No | Sniffing, Snooping |
| Data Modification | Yes | Original data cannot be transmitted |
| IP address Spoofing | Yes | Reroute, Modify the data |
| Denial of Service (DoS) | No | Network Floods |
| Man-in-the-Middle | Yes | Privacy, Legacy |
| Sniffer Attack | Yes | Privacy, Network Floods |



*Corresponding Author: M. Viju Prakash, Department of Computer Science & Engineering, Sardar Raja College of Engineering.*

A brute force attack [3] consists of trying every possible code, combination, or password until an intruder finds the right one. Brute force attack reduces the system performance, floods the network and makes vulnerable executions in the network middleware's. Here we have addressed these types of problems and to overcome the shortcomings of problems associated with brute force attack.

## III. PRINCIPLIES BEHIND BRUTE FORCE

In cryptography, key size or key length is the size measured in bits of the key used in a cryptographic algorithm. An algorithm's key length is distinct from its cryptographic security, which is a logarithmic measure of the fastest known computational attack on the algorithm, also measured in bits. The security of an algorithm cannot exceed its key length but it can be smaller. For example, Triple DES [4] has a key size of 168 bits but provides at most 112 bits of security, since an attack of complexity $2^{112}$ is known. This property of Triple DES is not a weakness, provided 112 bits of security is sufficient for an application.

The actual degree of security achieved over time varies, as more computational power and more powerful mathematical analytic methods become available. Even if a symmetric cipher is currently unbreakable by exploiting structural weaknesses in its algorithm, it is possible to run through the entire space of keys in which is known as brute force attack. Since longer symmetric keys require exponentially more work to brute force search, a sufficiently long symmetric key will prevent this line of attack. With a key of length n bits, there are $2^n$ possible keys. This number grows very rapidly as *n* increases. Moore's law suggests that computing power doubles roughly every 18 to 24 months, but even this doubling effect leaves the larger symmetric key lengths currently considered acceptably well out of reach. The large number of operations ($2^{128}$) required to try all possible 128-bit keys is widely considered to be out of reach for conventional digital computing techniques for the foreseeable future. However alternative forms of computing technology are anticipated which may have superior processing power than classical computers. So here it's a need of to design a password schema which must be computationally powerful than brute force attack.

## IV. EXISTING OTP ALGORITHM

Certain types of encryption in the networks, by their mathematical properties cannot be easily overcome by brute force. An example of this is the one-time password algorithm (OTP) [5], where every clear text bit has a corresponding key bit. One-time passwords rely on the ability to generate a truly random sequence of key bits. A brute force attack would eventually reveal the correct decoding, but also every other possible combination of bits, and would have no way of distinguishing one from the other. A small, 100-byte, one-time-password encoded string subjected to a brute force attack would eventually reveal every 100-byte string possible, including the correct answer, but possibly low chance.

Here we have analyzed one-time password algorithm for a secure network [6] available today based on mobile authentication and we have listed the possible attacks [7] to the one-time password algorithm.

### *One-Time Password Algorithm*

In existing one-time password algorithm, Java MIDlet is a client application and we assume that this runs in client mobile phones which can be able to receive one time passwords. A MIDlet is an application that uses the Mobile Information Device Profile (MIDP) of the Connected Limited Device Configuration (CLDC) for the Java ME environment. Typical applications include games running on mobile devices and cell phones which have small graphical displays, simple numeric keypad interfaces and limited network access over HTTP.

This whole design describes the two main protocols used by Java MIDlet system. Initially, the user downloads the client (Java MIDlet) to his mobile phone. Then the client executes a protocol to register with both server and a service provider utilizing server system for user authentication. After the successful execution of the activation protocol the user can run the authentication protocol an unlimited number of times.

### *Activation Protocol*

After the user has downloaded the client software from a service onto his mobile phone, he must activate the phone as an OTP receiver before it can be used for authentication to a web-based secure service [8]. The activation protocol takes place between five parties: the user, the user's mobile phone, the user's PC, the Server (SE), and the service provider (SP). The main steps of the protocol are summarized below.

1. The user authenticates himself to SP using credentials already known to SP.
2. When the user asks to activate his mobile phone as an OTP receiver, SP redirects the user's browser to SE with a URL that contains an activation request and a Secure Object [9].
3. SE verifies that the Secure Object comes from SP, and gets the user's phone number and other unique identification number like IMEI.

4. SE sends an activation code to the user's PC and an SMS message to the user's phone asking it to start the client software.
5. The mobile phone asks the user to enter the activation code, available on his PC, and transmits the code to SE.
6. SE verifies that the activation code is the same as the one sent to the PC, and sends a challenge to the mobile phone together with an encryption key $K_0$ (The role of $K_0$ is explained separately).
7. The user chooses a personal identification number (PIN) and enters it on the mobile phone, which generates a security code and a response. The response is the encryption of the challenge using the security code as key. The security code and response are sent to SE, and SE stores the security code.
8. SE verifies that the response and the security code correspond to the challenge, and if so, the user has activated the mobile phone as an OTP receiver for use with SP.

These steps should ensure that the PC and the mobile phone are in the same location, or at least that there exists a communication link between the person using the PC and the holder of the phone. Since the person using the PC is authenticated and has transferred the activation code to the phone, we can assume that this person really wants to activate the mobile phone as an OTP generator.

*Authentication Protocol*

The OTP-based authentication protocol takes place between five parties: the user, the user's mobile phone, the user's PC, the SE, and SP. The main steps of the protocol are described below.

1. The user enters the identity he shares with SP on its login page.
2. SP asks the user for an OTP, and sends a request to ES to generate an OTP for the user.
3. SE first sends an SMS to the user's mobile phone to start the client software. It then sends a challenge to the phone together with two encryption keys $K_i$ and $K_{i+1}$ (The role of $K_i$ and $K_{i+1}$ is explained separately).
4. The user enters his PIN on the phone, and the phone computes the same security code generated at the time of activation. The phone then encrypts the challenge with the security code as key and sends the cipher text as a response to SE.
5. ES verifies that the response from the mobile phone corresponds to the challenge, and sends an OTP to the phone.
6. The user enters the OTP on the SP's login page, and SP contacts ES to verify that the OTP is indeed the correct one for this user.

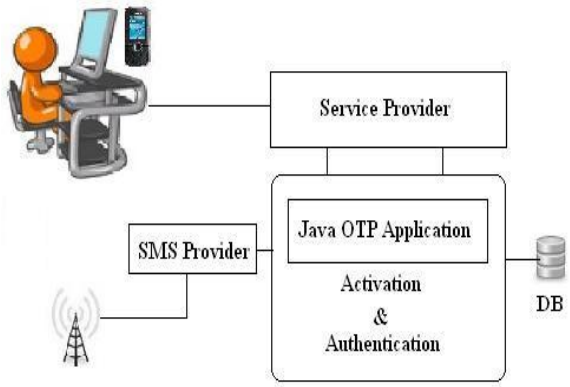


Fig1. Existing OTP Generation Mechanism

Fig1 shows architecture of existing OTP mechanism. The authentication protocol's main goal is to ensure that only the legitimate user can obtain an OTP from SE. The goal is not achieved fully because the phone's response to SE challenge is the encryption of the challenge using the key (security code) made during activation. The answer for this challenge may be known to non-legitimate users also. The correct generation of this key requires the correct PIN and other unique information, which possibly other person who are not legitimate also is supposed to have. This person was in turn authenticated at the time of activation; hence we cannot be confident that he is the legitimate user.

## V. ATTACKS ON MOBILE ACTIVATION

Existing mobile based one-time password algorithms are having the following possible attacks on the mobile activation process.

### *A. Malware-Based Replay Attack*

The goal of the following malware-based replay attack [10] is to collect a victim's username and password, and to generate the victim's OTPs on a mobile phone of the attacker's choice.

First, the malware captures the victim's username and password when he logs on to his secure network. Second, when the victim starts the implemented activation protocol, the malware captures the URL containing the Secure Object. The activation procedure is not secured against replay attacks occurring inside a time window of a few minutes, nor is it tied to one particular IP address or SSL session [11]. Consequently, the malware need not disrupt the user's activation process, but can just wait until the user has completed the activation and then transmit the URL to

the attacker's PC. The attacker enters the URL, containing the Secure Object, into a browser. Finally, the attacker submits his own phone number to download, install, and activate the Java MIDlet on his mobile phone. This type of attack is possible when an attacker captures the Secure Object.

*B. Phishing Attack*

To initiate a phishing attack [12], an attacker can generate phishing e-mails asking users to log on to a proxy masquerading [13] as the users secure network. There is sample evidence showing that many individuals receiving phishing e-mails enter their login credentials at fake web sites.

Once a user has connected to the proxy, it forwards messages in both directions between the customer's PC and the secure network's central infrastructure. The proxy can read all messages, change their contents, and create fake messages; in particular, the proxy records the username and password transmitted by the tricked user. The proxy can then generate a request to activate a mobile phone as an OTP generator and use the returned URL to activate the attacker's phone as an OTP generator for the tricked user's login. An attack will normally give the attacker access to an account only once. This attack is different because it lets an attacker generate as many OTPs as he wants on his own phone. He can therefore access an account whenever he desires until the account is closed by the secure network.

*C. Malware-based Impersonation Attack*

The malware-based replay attack can be modified to obtain a malware-based impersonation attack targeting a user in a secure network system. Here all users have the option to activate their mobile phones as OTP generators. Here malware just waits for a user to log on the secure system. The malware then sends a request to activate a mobile phone as an OTP generator without the user realizing what is going on. When the URL with the secure object is returned, it is forwarded to the attacker's PC instead of redirecting the user's browser to SE. The attacker utilizes the URL to activate his own mobile phone as an OTP generator for the user's secure account. This attack is deemed practical since there already exists malware that steals information and manipulate client – server communication.

## VI. PROPOSAL

Here we designed a PassText based authentication scheme in order to produce a security code instead of using a challenge. Our proposed idea explores the usage of PassText [14] which is impossible to break with brute force attack and stay remains as unpredictable. Proposed works are listed below.

*Proposed Activation Protocol*

1. The user authenticates himself to SP using credentials already known to SP.
2. When the user asks to activate his mobile phone as an OTP receiver, SP redirects the user's browser to SE with a URL that contains an activation request and a Secure Object.
3. SE verifies that the Secure Object comes from SP, and gets the user's phone number and other unique identification number like IMEI. Here users have to specify PIN.
4. SE sends an activation code to the user's PC and an SMS message to the user's phone asking it to start the client software.
5. The mobile phone asks the user to enter the activation code, available on his/her PC, and transmits the code to SE.
6. SE verifies that the activation code is the same as the one sent to the PC, and sends a Passphrase to the mobile phone together with an encryption key $K_0$.
7. The user chooses a personal identification number (PIN) and enters it on the mobile phone, which generates a security code and a PassText response. The response is the encryption of the PassText using the security code as key. PassText is known only to the legitimate user. The security code and response are sent to SE, and SE stores the security code.
8. SE verifies that the response and the security code correspond to the PassPhrase send, and if so, the user has activated the mobile phone as an OTP receiver for use with SP.

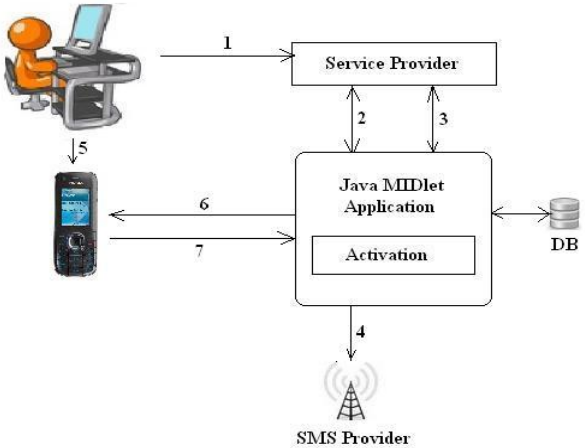


Fig2. Proposed Layout of Activation Protocol

In the above proposed protocol, Passphrase is a

simple passage given by either a user or SP and it sent from SP to user while activation process. User converts this Passphrase into PassText by some remember able changes. These changes are known to and done by only legitimate users. So this leads to maximum security and so security level for this scheme becomes unpredictable and proposed security level of this measure described later.

*Proposed Authentication Protocol*

The proposed protocol steps are as follows. This ensures the necessary authentication steps for recognizing legitimate user.

1. User requesting SP for an OTP at the first step, rather than user enters the secure login page.
2. SP verifies the mobile request with existing database and if it's approved request, SP request SE to generate an OTP for the user.
3. SE sends a PassPhrase to the phone together with two encryption keys $K_i$ and $K_{i+1}$.
4. The user enters his PIN and PassText from Passphrase on the phone, and so the mobile computes security code. The mobile then encrypts the PassText with the security code as key and sends the cipher text as a response to SE.
5. ES verifies that the response from the mobile phone corresponds to the passphrase, and sends a sms to the users mobile to login using the identity shares with SP.
6. Now session cookie is activated by SE and it will be send to user's PC after completing login session by the user.
7. If session cookie has arrived to user's PC, secure login page will be redirected automatically by session cookie after a successful logon process finished by a user. If user logins already without mobile authentication, secure page will not be redirected as there is no session cookie. This method completely avoids malicious user to login with the secure system.
8. Redirected URL will request OTP for a secure authentication. Now SE will send an OTP to user mobile through SP.
9. User can authenticate them by the OTP received through mobile. Successful users only can receive OTP from SP after completing a strong mobile authentication.

This proposed protocol ensures that only legitimate users are accessing the secure system and it also avoids malware based attacks. It's proven that PassText is a string which is known neither only to the legitimate users nor to unauthorized. So brute force attack turns to be zero and its measures are discussed later.

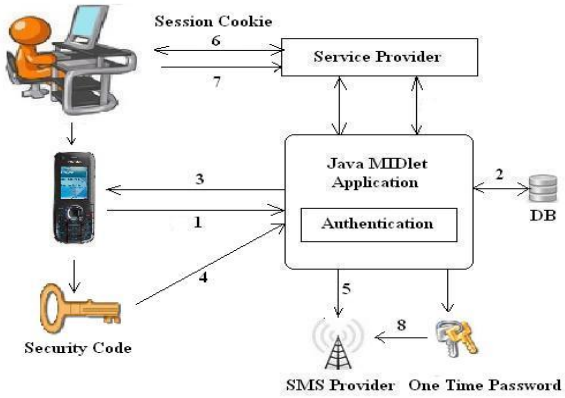


Fig3. Proposed Layout of Authentication Protocol

*C. Generation of Security code and Responses*

The hash function SHA-1 and the encryption algorithm AES with a 16-byte key are used to generate the security code and the responses. Hashing is denoted by $H$ () and encryption with key $K$ is denoted by $E_K$ ().

The security code, *SC*, is computed by the following hash, truncated to 16 bytes,

$$SC=H(PIN||IMEI||CR||SPID)_{16} \quad (1)$$

*where* || denotes concatenation. PIN is a secret number with at least four digits entered by the user; IMEI is a 14 digit code uniquely identifying the mobile phone where the Java MIDlet client is installed; CR or client reference is a 40-byte random string generated on the mobile phone during the activation protocol; and SPID is a public value identifying the SP to whom the user wishes to authenticate.

The client reference needs to be stored on the mobile phone for later use. It is only stored in encrypted form. During the activation protocol it is encrypted using AES with the key $K_0$ which is sent by ES together with the PassPhrase.

When the client reference is needed in the authentication protocol it is first decrypted using Ki, and when it goes back into storage it is encrypted using $K_{i+1}$. The keys $K_i$ and $K_{i+1}$ are sent from SE together with the challenge in the authentication protocol.

The generation of a 16-byte response, R, to a Passphrase PP, is defined by the expression

$$R = ESC (PP) \quad (2)$$

where SC is the 16-byte security code defined by (1) and PP is a 16-byte PassPhrase received from ES.

*D. Encryption of Client Reference*

The expression (1) for the security code contains a random 40-byte client reference (CR). The purpose of the client reference seems to be to increase the entropy of the input to the hash function in (1). The client reference needs to be stored on the mobile phone for future use. It is specified that the client reference should only be stored in encrypted form with keys to encrypt and decrypt supplied by ES.

At first there seems to be some added protection from this encryption: An attacker who gets hold of a user's mobile phone can determine the IMEI number of the phone and read the memory where the client reference is stored. The SPID is publicly known. If the client reference was stored in clear text, then the attacker could record these values, and exhaustively try all different PINs to generate the set of possible security codes. Determining the correct security code would then be no harder than guessing the user's PIN. However, this approach is not available to the attacker since the client reference is encrypted before it is stored. Thus, the attacker needs the decryption key before being able to generate the (relatively small) set of possible security codes.

Unfortunately, the SE supplies the needed decryption key before any authentication takes place. If an attacker gets hold of a user's mobile phone and is able to read the encrypted client reference, all he needs to do is to follow the authentication protocol to get the decryption key from SE. Hence, the encryption of the client reference does not add to the security of the scheme. But even though it seems to be easy, PassText gives maximum security against brute force and so other users cannot authenticate with the system.

Another reason for introducing pairs of keys $K_i$,$K_{i+1}$ to repeatedly decrypt and re-encrypt the client reference is to ensure that no two phones can obtain the same sequence of OTPs from the server. If an attacker can copy the memory of a legitimate user's phone to his own phone, both cannot be used to generate the correct SC because of PassText.

*E. Making PassText from Passphrase*

Here user should provide a combination of 40 – byte passphrase as is humanly possible to remember, making them impossible to guess by brute force or other means. At the same time, the users should not easily forget their passphrase or parts thereof as time from the initial enrollment step passes. Instead of having the burden of providing the passphrase all the time in order to produce SC, it should be instead stored and readily available in the SP itself. The following diagram shows an example of PassText from Passphrase which has small changes. A possible list of atomic modifications includes: deleting any character from the text or typing any character in any location. Making a PassText from Passphrase is very easy process and resultant PassText and Passphrase will be saved in the same server after the users credentials has been successfully verified.

| | |
|---|---|
| There is lots of honking with the cars, but in some places, it's still serene and there are absolutely the most breath taking views… | There is lots of honking with the bikes, but in some places, it's still serene ad there are absolutely the most breath taking views… |

Fig 4.Left: Passphrase; right: PassText

## VII. SIMULATION SETUP

It has been analyzed that in order to thwart the possible attacks in a secure system, three countermeasures have to be considered. To protect against malware-based replay attack, the proposed activation protocol needs to be secured against the replay of old requests. SE must ensure that each secure object is only used once. Also it should not be allowed to activate another mobile phone as OTP generator for an account, if there already exists a mobile phone activated for the specific account. In order to protect against malware-based impersonation attack and phishing attack there is a need for a tighter control over the transition from old OTP generator to a new mobile phone.

Simulation was done between a system and a mobile phone which is having the capacity of receiving and storing the secure object (Fig 5). Mobile phone can be replaced with any device which should have an enough capacity to manipulate a secure object. While analyzing the existing one-time password algorithm, obtained results shows that brute force attack is possible in most of the secure networks. Even though an OTP giving more reliability, according to the cryptographic policy, brute force attack will try all possible combinations of passwords until it finds the correct one. It is very difficult to defend against brute force.

First when a user requests SP to provide an OTP, SP verifies that whether the mobile request has been registered already. If not, the request will be discarded immediately. If the requested is approved, SE sends a Passphrase which can be considered as a challenge to the user along with the keys. By the help of personal identification number and a passtext, mobile phone will generate a response to the challenge.

Here we have designed a schema which will ensure that same challenge and response should not be given to different users. In this way a secure system can be protected safely from a number of user's. If the

response is approved to ES, a message will be sent to user's mobile.

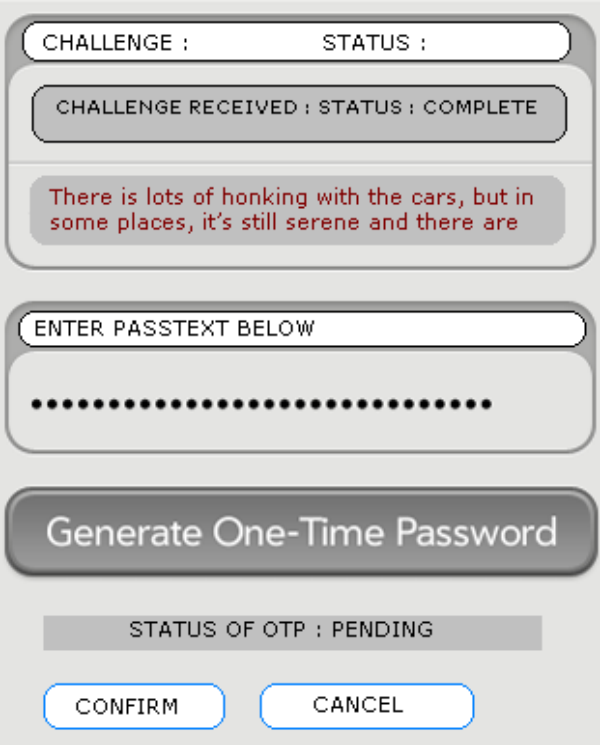


Fig5. Simulation setup (Making of OTP)

After getting the confirmation message from ES, user will logon by having his/her own credentials. This mechanism completely avoids malware based impersonation attack since user will not advised to use URL at the first step itself.

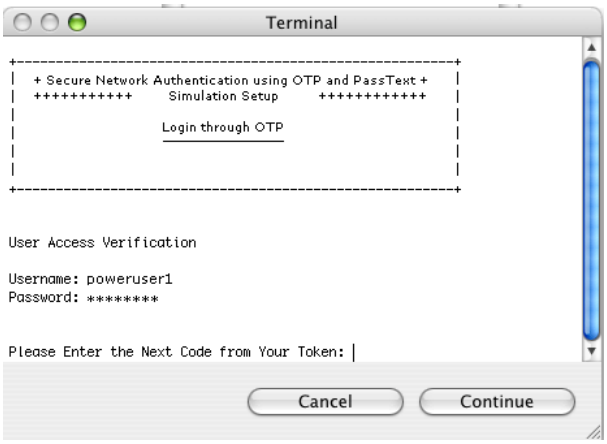


Fig6. User logon process

When the authenticated user logon into the system, SE will send a session cookie to the user's PC and this will redirect to the secure system. If the user have an OTP token, SE will never send session cookie and this will be stop the malicious users to access the secure system.

## VIII. SIMULATION RESULTS

It seems unfair to say that any set of alphanumeric characters are equally easy to commit to memory. For example "A7Jo0" and word "commit" are not both equal to six units of memory. We have compared password space with different password schemas we can identify the most secure approaches with respect to brute force attack. Table2 demonstrates comparison of password space and password length for popular user authentication schemas. Table2 shows that the approach presented by us is both more secured and the easiest to remember. At the same time, it is relatively fast to produce during an authentication procedure.

Table II
Password space comparison

| Authentication System | Alphabet | Password Length | Password Space Size |
|---|---|---|---|
| Password | 64 | 8 Char | $2.8 \times 10^{14}$ |
| Pin Number | 10 | 4 Numbers | $1 \times 10^{4}$ |
| Text with Graphical Assistant | 10 spaces | 8 Char | $2 \times 10^{6}$ |
| PassText | 40 | 3 Changes | $55 \times 10^{1250}$ |
| OTP with PassText | 40 | 3 Changes | $55 \times 10^{125000}$ |

Table II shows the higher strengthen measure of OTP when it added with proper PassText. This highest measure cannot be predicted and not to be traced by malwares. At the same time, it is relatively fastest authentication scheme for secure networks. Fig7 illustrates the strength of the proposed method. Brute force at this level is impractical.

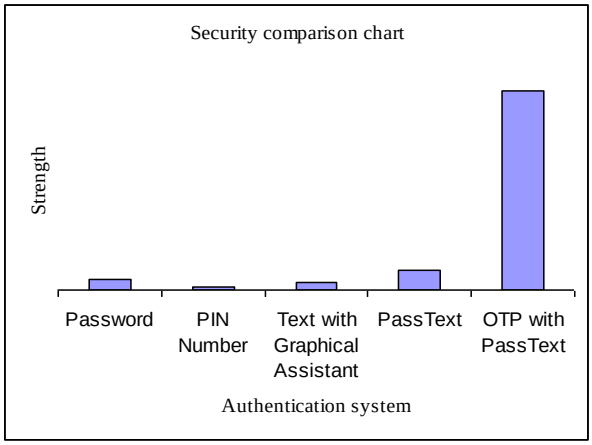


Fig7. Comparison of security levels

## IX. CONCLUSION

In this paper, we have analyzed the security attacks in a secure network and we have eliminated most of vulnerable attacks including brute force. Results are shown us that this proposed system can be applied in all the networks which requires higher authentication.

## X. REFERENCES

[1] Yi-Bing Lin Meng-Hsun Tsai Nat. Chiao Tung Univ., Hsinchu "*Eavesdropping Through Mobile phone*" IEEE Transactions on Vehicular Technology Volume:56 Issue:6 on pages: 3596 – 3600.

[2] Callegati, F. Cerroni, W. Ramilli, M. Univ. of Bologna, Bologna "*Man-in-the-Middle Attack to the HTTPS Protocol*" IEEE Transactions on Security & Privacy Volume: 7 Issue: 1 on pages: 78 – 81.

[3] Kong, A.W.K.; Zhang, D.; Kamel, M "*Analysis of Brute-Force Break-Ins of a Palmprint Authentication System*" IEEE Transactions on Systems, Man, and Cybernetics, Part B: Cybernetics Volume: 36 Issue: 5 on pages: 1201 – 1205.

[4] Coppersmith, D.; Johnson, D. B.; Matyas, S. M "*A proposed mode for triple-DES encryption*" IEEE Transactions on IBM Jounal of Research and Development Volume: 40 Issue: 2 on pages: 253 – 262.

[5] Sung-Ming Yen "*Security of a One-Time Password Signature*" IEEE Transactions on Electronics Letters Volume: 33 Issue: 8 on pages: 677 – 679 published in 1997.

[6] Hiltgen, A.; Kramp, T.; Weigold, T "*Secure Internet Banking Authentication*" IEEE Transactions on Security & Privacy Volume: 4 Issue: 2 on pages: 21 – 29 published in 2006.

[7] Okamoto, E.; Tanaka, K. "*Identity-based information security management system for personal computer networks*" IEEE Transactions on Selected Areas in Communications Volume: 7 Issue: 2 on pages: 290 – 294.

[8] Jinpeng Wei, Singaravelu, L.; Pu, C. "*A Secure Information Flow Architecture for Web Service Platforms*" IEEE Transactions on Service Computing Volume: 1 Issue: 2 on pages: 75 – 87.

[9] Lu Ma; Tsai, J.J.P. "*Formal Modeling and Analysis of a Secure Mobile Agent System*" IEEE Transactions on Systems, Man and Cybernetics Volume: 38 Issue: 1 on pages: 180 – 196.

[10] O'Donnell, A.J "*When Malware Attacks (Anything but Windows)*" IEEE Transactions on Security and Privacy Volume: 6 Issue: 3 on pages: 68 – 70.

[11] Zhao Huawei; Liu Ruixia "*A Scheme to Improve the Security of SSL Layer*" IEEE Conference on Circuits, Communications and Systems PACCS 2009 on pages: 401 – 404.

[12] Urien, P. "*Introducing TLS-PSK authentication for EMV devices*" IEEE Conference on Collaborative Technologies and Systems CTS 2010 on pages: 371 – 377.

[13] Maxion, R.A.; Townsend, T.N. "*Masquerade Detection Augmented with Error Analysis*" IEEE Transactions on Reliability Volume: 53 Issue: 1 on pages: 124 – 147.

[14] Yampolskiy, Roman V. "*Secure Network Authentication Using PassText*" IEEE Conference on Information Technology ITNG 2007 on pages: 831 – 837.

## ABOUT AUTHORS

M. Viju Prakash has accomplished Bachelor's and Master's degree in Computer Science and Engineering from Anna University, Chennai. His areas of interest are Network Security, Wireless Sensor Networks and Mobile Ad hoc Networks. He is presently working as a senior lecturer in Sardar Raja College of Engineering.

P. Alwin Infant has accomplished Bachelor's degree in Computer Science and Engineering from M.S University and Master's degree from Anna University, Chennai. His areas of interest are Network Security, Mobile Ad hoc Computing and Data Structures. He is presently working as a lecturer in Sardar Raja College of Engineering.

S. Jeya Shobana has accomplished Bachelor's degree in Computer Science and Engineering from MS University and Master's degree from Anna University, Chennai. Her areas of interest are Network Security, Wireless Sensor Network and Operating Systems. She is now working as a lecturer in Francis Xavier Engineering College.