Internet Banking Two-Factor Authentication using Smartphones

Costin Andrei SOARE
IT&C Security Master
Department of Economic Informatics and Cybernetics
Bucharest University of Economic Studies, Romania
ROMANIA
soarecostin@gmail.com

Abstract: This paper describes a method of implementing two-factor authentication using smart phones as software tokens. The proposed system will use the mobile phone as a software token and generate unique one time passwords (OTP) that will be used when authenticating to an Internet Banking application. The tokens can also serve as a method of signing online money orders. We will prove in this article the cost efficiency of the proposed architecture for both consumers and companies.

Key-Words: OTP, TOTP, HOTP, token, internet banking, two-factor authentication

1. Introduction

The mobile phones have a huge potential of interacting with services anywhere, anytime since there is more than one phone per citizen in Europe [1]. The application described in this paper uses a mobile phone as a one time password (OTP) generator. The generated password will then be used in the authentication process of a client to an Internet Banking demo application where the client can also sign money orders using a similar challenge-response generator from their mobile phone. We will describe how the application resists on different attacks, in particular phishing attacks. The Internet Banking generally accepted definition, as it allows clients to perform transactions and payments over the internet through a bank's secure website [7], is used in this research.

Security concerns are rising today in all domains, such as banks, government, medical assistance, military organization, etc. Governmental organizations are passing laws that are forcing other organizations and agencies to comply with standards. There are several issues when it comes to security concerns in these industries and one common issue is represented by passwords [2].

Today, most systems are using static passwords for verifying their client's identity, passwords that come with major security concerns. Users tend to use easy to guess passwords, they use the same password for multiple accounts, write the passwords and store them on their machines, etc. Furthermore, hackers can use many techniques to steal passwords, such as shoulder surfing, snooping, sniffing, etc.

Many solutions have been proposed in order to fix this concern. Some of them are hard to implement, others don't meet the security concerns of the companies while others are difficult to be used. Two-factor authentication using hardware devices, such as tokens or ATM cards has also been proposed to solve these problems and proved to be successful and difficult to be hacked. However, this solution has also disadvantages such as the cost of manufacturing and managing tokens or cards. Also, from the consumer's point of view, using more than one token or card has an increased chance of them being lost or stolen [2].

Mobile phones have traditionally been considered tools for making phone calls. But today, given the progress that has been done in the hardware and software, mobile phones use has been expanded
to internet, email, games, music, photos and thousands of other rich applications added by third-party developers.

Installing third-party applications allows mobile phones to provide expanded new services other than communication [2]. The use of mobile phone as a software token will make it easier for the customer to deal with multiple two-factor authentication systems and will also reduce the cost of manufacturing, distributing and maintaining millions of hardware tokens.

2. Describing the problem

In this paper I will introduce a system of two-factor authentication using smart phones, which consists of a server and a smart phone that allows running third-party applications.

By definition, authentication is the use of one or more mechanisms in order to prove that you are who you claim to be. Once your identity is validated, access is granted [2]. Three universally recognized authentication factors exists today: what you know (passwords), what you have (tokens, cards) and what you are (biometrics). Recent work has been done in trying alternative factors, for example somebody you know, a factor that can be applied in social networking.

Two-factor authentication is a mechanism that implements two of the above mentioned factors and is considered stronger and more secure than the traditionally implemented one factor authentication system. For example, withdrawing money from an ATM machine uses two factor authentication: the ATM card (what you have) and the personal identification number (what you know).

Passwords are known to be one of the easiest targets of hackers. Therefore, most companies are searching more ways to protect their customers and employees. Biometrics are known to be very secure [2], but are used only in special organizations (such as military organizations) given the expensive hardware needed and their high maintenance costs. As an alternative, banks and companies are using tokens as a way of two-factor authentication.

A token is a physical device that generates passwords needed in an authentication process. Tokens can either be software or hardware. Hardware tokens are small devices that can be easily carried. Some of these tokens store cryptographic keys or biometric data. Anytime a user wants to authenticate in a service, he uses the one-time password displayed on the token in addition to his normal account password. Software tokens are programs that run on computers and provide a one time password that it is changed after a short amount of time (usually 30 seconds). OTP algorithm's security is very important because no one should be able to guess the next password in sequence. The sequence should be random to the maximum possible extent, unpredictable and irreversible. Factors that can be used in OTP generation include names, time, seeds, etc. Several commercial two-factor authentication systems exist today such as RSA SecurID [2].

SecurID uses a token whose internal clock is synchronized with the main server. Each token has a unique seed which is used to generate a pseudo-random number. This seed is loaded on the server upon purchase of the token and is used to identify the user. An OTP is generated using the token every 60 seconds. The same process occurs on the servers side. A user uses the OTP along with a PIN or a password which only he knows to authenticate on the server side. If the input from the user matches the server generated OTP, the user is authenticated [2].

Even if tokens provide a safer environment for users, they can be costly for companies. For example, a bank with a million customers will have to purchase, install and maintain a million tokens, as well as provide support for non-initiated users on how to use the tokens. The bank also has to be ready to replace any broken or stolen token. Replacing a token is a lot more
expensive than replacing an ATM card or than replacing a password. From the user's perspective, having accounts at multiple banks comes with the need of carrying and maintaining several tokens, which represents a big inconvenience. In many cases, customers are being charged for lost, stolen or broken tokens [2].

The following is the description of an application that is cost-efficient for the companies that will choose to implement it. It will also allow customers to install multiple software tokens on their mobile phones, thus making them worry only about their mobile phone instead of worrying about several other tokens.

3. The solution

3.1 General architecture

The proposed system is secure and consists of:
1. mobile smartphone application compatible with Android OS, written in Java with Android SKD; The software application can be developed for other operating systems as well, using the same architecture as the one described.
2. server application, running from browser, written in PHP using MySQL databases;

The unique OTP is generated by the mobile application offline, without having to connect to the server. The mobile phone will use some unique information in order to generate the password. The server will use the same unique information and validate the OTP.

In order for the system to be secure, the unique OTP must be hard to predict by hackers. The following factors will be used to generate the OTP:

- **IMEI**: *International Mobile Equipment Identity*, unique to each mobile phone and allows each user to be identified by his device. This is accessible on the mobile phone and will be stored in the server's database for each user.
- **PIN**: Needed for verifying the authenticity of the client. If the phone is stolen, a valid OTP can't be generated without knowing the user's PIN. The PIN isn't stored in the phone's memory. It is only being used only to generate the OTP and destroyed immediately after that. In order for the PIN to be hard to guess and resistant to brute-force attacks, a minimum of 8 characters long PIN is required, with a mixture of upper and lower-case characters, digits and symbols.

  - **Timestamp**: Used to generate unique OTP, valid for a short amount of time. The timestamp on the phone must be synchronized with the one from the server.

The algorithm that will generate one time passwords is **TOTP** (Time-based One Time Password), an extension of **HOTP** (HMAC-Based One Time Password) to support a time based moving factor. **TOTP** is an Internet Engineering Task Force standard and a cornerstone of Initiative for Open Authentication (OATH) [3].

As defined in RFC 4226 [4], the HOTP algorithm is based on **HMAC-SHA-1**. The security and strength of this algorithm depends on the properties of the underlying building block HOTP, which is a construction based on HMAC [RFC2104] using SHA-1 as the hash function. The security analysis conclusions described in RFC 4226 are that for all practical purposes.

The conclusion of the security analysis detailed in RFC4226 is that, for all practical purposes, the outputs of the dynamic truncation on distinct inputs are uniformly and independently distributed strings [4].

HOTP was published as an informational IETF RFC 4226 in December 2005, documenting the algorithm along with a Java implementation. Since then, the algorithms were adopted by many companies worldwide and became the world's leading standard for event-based OTP authentication. The HOTP algorithm is a freely available open standard [5].
3.2 HOTP Algorithm

The HOTP algorithm is based on an increasing counter value and a static symmetric key known only to the token and the validation service. In order to create the HOTP value, we will use the HMAC-SHA-1 algorithm, as defined in RFC 2104 [4].

As the output of the HMAC-SHA-1 calculation is 160 bits [4], we must truncate this value to something that can be easily entered by a user.

\[
\text{HOTP}(K, C) = \text{Truncate}(\text{HMAC-SHA-1}(K, C)) \quad (1)
\]

where:
- **Truncate** represents the function that converts an HMAC-SHA-1 value into an HOTP value.
- \( C \) is 8-byte counter value, the moving factor. This counter MUST be synchronized between the HOTP generator (client) and the HOTP validator (server).
- \( K \) is the shared secret between client and server; each HOTP generator has a different and unique secret \( K \).

The HOTP values generated by the HOTP generator are treated as big endian [4].

We can describe the operations in 3 distinct steps [4]:

1. Generate an HMAC-SHA-1 value:

\[
\text{HS} = \text{HMAC-SHA-1}(K, C) \quad (2)
\]

2. Generate a 4-byte string (Dynamic Truncation):

\[
\text{Sbits} = \text{DT}(\text{HS}) \quad (3)
\]

DT returns a 31-bit string

3. Compute an HOTP value:

\[
\text{Snum} = \text{StToNum}(\text{Sbits}) \quad (4)
\]

Convert \( S \) to a number in \( 0 \ldots 2^{31} - 1 \)

\[
\text{Return } D = \text{Snum mod } 10^{\text{Digit}} \quad (5)
\]

\( D \) is a number in the range \( 0 \ldots 10^{\text{Digit}} - 1 \)

\( \text{Digit} \): number of digits in an HOTP value, system parameter.

The Truncate function performs Steps 2 and 3, i.e., the dynamic truncation and then the reduction modulo \( 10^{\text{Digit}} \). The purpose of the dynamic offset truncation technique is to extract a 4-byte dynamic binary code from a 160-bit (20-byte) HMAC-SHA-1 result [4].

We define DT as (6):

\[
\text{DT (String)}
\]

Let \( \text{OffsetBits} \) be the low-order 4 bits of String[19]

\[
\text{Offset} = \text{StToNum}(\text{OffsetBits})
\]

Let \( P = \text{String}[\text{Offset}]...\text{String}[\text{Offset}+3] \)

Return the Last 31 bits of \( P \)

End (DT)

where:

\[
\text{String}=\text{String}[0]...\text{String}[19] \quad (7)
\]

\( 0 \leq \text{Offset} \leq 15 \quad (8) \)

The reason for masking the most significant bit of \( P \) is to avoid confusion about signed vs. unsigned modulo computations. Different processors perform these operations differently, and masking out the signed bit removes all ambiguity. Implementations MUST extract a 6-digit code at a minimum and possibly 7 and 8-digit code [4].

3.3 TOTP Algorithm

Notations:

- \( X \) represents the time step in seconds (default value \( X = 30 \) seconds) and is a system parameter;
- \( T0 \) is the Unix time to start counting time steps (default value is 0, Unix epoch) and is also a system parameter.

Basically, we define TOTP [6] as:

\[
\text{TOTP} = \text{HOTP}(K, T) \quad (9)
\]
where $T$ is an integer and represents the number of time steps between the initial counter time $T_0$ and the current Unix time (i.e. the number of seconds elapsed since midnight UTC of January 1, 1970).

More specifically,

$$T=(\text{Timestamp current} - T_0)/X$$ (10)

The analysis demonstrates that the best possible attack against the HOTP function is the brute force attack [6].

4. The proposed system

4.1 Application's description

In this paper, the shared secret from TOTP algorithm will be computed from the user's ID, phone's IMEI, timestamp and the user's PIN. The solution that I tested requires the completion of three steps: (1) installing the application on the mobile phone (2) user registration and (3) user authentication. The system will have several modules, for user registration, generation of one-time passwords for authentication or for signing money orders. The system will also be able to fix timestamp difference problems that can lead to generating invalid one-time passwords.

![Diagram](image)

Figure 1. Initial user registration on the mobile application

4.1.1 User registration

The server application can create on demand an activation code for a given user ID. In order to complete the initial user registration, an active internet connection is required on the mobile phone. The data transfer will be made through a HTTPS secure connection. After the user installs the application on their mobile phone, he will be prompted for the activation code and the PIN that he wishes to set for the app. This is where the user must enter the activation code generated by an employee of the company in the server application and his chosen PIN. After the user enters the activation code his PIN code, a data transfer between the phone and the server is initiated. The phone will send the activation code, the hash of the PIN, the IMEI number and the timestamp to the server. The server will
identify the user by the activation code. At this point, the PIN can be destroyed from the mobile application.

The server will store the IMEI number, the hash of the user's PIN and the timestamp difference between the two devices. The server will then send a hash of the user's ID (Encrypted ID) to the mobile application, hash that will be later used in generating the unique OTP. I have chosen to send a hash of the user's ID in order to protect user's identity. If the phone is lost or stolen, the owner's user ID will not be stored in the phone's memory, but only it's hash. See fig. 1 for a visual approach of the initial user registration on the mobile application.

4.1.2 Generating OTP passwords

Generating the unique passwords is an offline process once the user has installed, configured and synchronized the application on his mobile, for which an active internet connection is required. When the application starts, the user will be prompted for the PIN code. Since the application does not store the PIN code, it will not be able to verify if the user entered the right one or not. If the PIN is different from the one used at registration, the OTP result will be different from the one computed on the server and therefore the user won't be able to authenticate in the Internet Banking application.

The unique OTP is generated with the TOTP algorithm, using the hash of the user ID (stored on the phone), the phone's IMEI (requested when opening the application) and the user's PIN (also requested every time the user uses the application). These credentials are then signed with the current timestamp, as defined in TOTP. The one-time password will be valid for 30 seconds since it's generation. After 30 seconds, the application will automatically generate a new valid OTP.

4.1.3 Signing money orders

The mobile application can also sign money orders initiated inside the Internet Banking application. In this case, the server will create a challenge code, using credentials like the source account number, the destination account number, the amount of money to be transferred and the current timestamp. The mobile application has a dedicated section for signing money orders, where the user is asked to introduce their PIN code and the challenge code displayed by the server in the Internet Banking application. The OTP algorithm used to generate the response from the token is the same as in the authentication process, with the only difference that the key also concatenates the server challenge code. The validity period of the challenge code is 5 minutes, while the validity of the OTP response is 60 seconds.

4.1.4 Synchronizing timestamps

In order for the system to work, the timestamp must be synchronized between the server and the phone. At the moment of user registration, the timestamp difference between the server and the phone is stored in the database, on the server side. Therefore, every time when the server will generate its own verification OTP code will adjust it's timestamp in order to match the timestamp from the phone.

The system will stop working properly if the phone timestamp will be changed afterward. In this case, the unique code generated by the phone will not match the one computed by the server and the user won't be able to sign into the Internet Banking application.

This problem can be easily fixed by introducing a “Timestamp sync” module in the mobile application that attempts to connect to the server and store the new timestamp difference. In order for this module to work, an active internet connection is needed on the mobile phone.

4.2 System's practical implementation

The described system will consist of three applications, two on the server side and one mobile application:
Internet banking module (front-end), which will be used by clients. They will access the Internet Banking application upon a successful login with their user ID and the unique password generated by the mobile phone application. Once authenticated, they can verify their account statements and initiate money orders.

- Mobile phone application, described above, that will generate unique one time passwords used by the clients in order to authenticate to the Internet Banking application.
- Internet Banking admin application, used by bank's employees, where they can create new clients, new accounts for existing clients or generate activation codes for clients that request access to the Internet Banking application. Employees will have different permissions inside the admin application.

5. Conclusion

Today, the use of one-factor authentication (e.g. password) is not considered secure anymore in the internet banking world. Passwords which are easy to guess, like name, date of birth are sure targets for automated password collecting programs.

Two-factor authentication has been recently introduced to meet the demand of organizations for providing a stronger and safer authentication process for their users. In most cases, a hardware token is given to each client for every Internet Banking application he uses. With the number of users that are requesting for a token viral increasing, the cost for manufacturing, maintaining and replacing them is becoming a burden for both organizations and clients [2]. Since most of the clients carry a smartphone at all times, an alternative is to install all the tokens on the mobile phones, as applications. This approach will help in reducing the costs and the number of devices carried by the client.

This paper focuses on the implementation of two-factor authentication on any smart-phone that allows third-party developers to add and run applications (such as Apple's iOS, Android, BlackBerry OS, Windows Phone, Symbian). The implemented demo application was created for Android, but can be easily written on any other operating system, using the design implementation described.

References