Computer and Information Science; Vol. 8, No. 3; 2015
ISSN 1913-8989 E-ISSN 1913-8997
Published by Canadian Center of Science and Education

Modified USB Security Token for User Authentication

Wala'a M. AlOmari[1] & Hesham Abusaimeh[2]

[1] Computer Engineer, Jordan Academy for Maritime Studies, Amman, Jordan

[2] Associate Prof., Department of Computer Science, Faculty of Information Technology, Applied Science University, Amman, Jordan

Correspondence: Hesham Abusaimeh, Department of Computer Science, Faculty of Information Technology, Applied Science University, Amman, Jordan. E-mail: h_saimeh@asu.edu.jo/w.alomary@yahoo.com

Received: March 9, 2015 Accepted: April 23, 2015 Online Published: July 31, 2015

doi:10.5539/cis.v8n3p51 URL: http://dx.doi.org/10.5539/cis.v8n3p51

Abstract

Computer security has been a significant importance in today's world. Many researches have been done in order to improve the security services with encryption and decryption of sensitive. In addition, Security protocols have been developed to protect accessing the data from the authorized users. One of these protocols is the One-Time Password (OTP) authentication in the USB security tokens. A well-known USB security Token is the Yubikey security tokens. However this token has protocol overhead and time consuming in addition to the speed and memory capacity limitations.

In this paper, we have proposed a modification to the Yubikey security protocols in order to enhance the overhead, speed and size limitations in the user authentication process, in addition to increase the security factors that depend on a random number generated from the server and sent to the user via e-mail and SMS to his mobile. Experimental results have been conducted using C# programming language for the user and the server side. All the results show the efficiency improvement of our proposed protocol over the Yubikey security token in the terms of authentication factors, speed and memory size.

Keywords: usb tokens, yubikey tokens, computer security, user authentication, advanced encryption standard (aes), one time password (otp)

1. Introduction

Computer security is critical issue and it has been developed in recent years, it is defined as the protection mechanisms and services of the computing systems and the data stored or accessed. The term security in general has three main classifications. The Computer Security, which is a generic name for the collection of tools designed to protect data and to thwart hackers inside the computing systems. The Network Security that concentrates on protecting the data during their transmission over the computer net-works. And finally the Internet Security that provides the mechanisms of protecting data during their transmission over a collection of interconnected networks. Many researches have been conducted in general to protect users and to enhance their security level. However few of these researches concentrate on both hardware and software to protect these data (Stamp, 2006; Stallings, 2011).

The rapid growth of online identity theft has exposed the vulnerabilities of logging with a username and password. Any information that is static, or stored on a device connected to the Internet, can be copied and misused by malware. Security experts agree that the best way to secure an online identity is to move out the user credentials to a separate device that is not connected to the Internet, and which generates a new and encrypted pass code every time it is used. Most uses of security tokens face many problems related to complexity, difficulty and security attacks (Stamp, 2006; Stallings, 2011; Paar & Pelzl, 2010).

A well-known USB security Token is the Yubikey security tokens. However this token has protocol overhead and time consuming in addition to the speed and memory capacity limitations. New threats, risks, and vulnerabilities as well as evolving business requirements underscore to the need for a strong authentication approach based on simple service delivery.

In this paper, we have proposed a modification to the Yubikey security protocol. This work aims to maintain high security level; by the utilization of three authentication factors instead of one; Knowledge, Possession and

Biometric. On the hand the modified protocol of the user authentication will complex, easy to use, and users and vendors will be satisfied.

Our modified protocol should be able to have some special characteristics that give the work the justifications to be implemented, which are:

1- Strong multi-factor authentication: It combines the hardware authentication token with a PIN and a biometric authentication.

2- One Time Password: The modified security token works by emitting a One Time Password (OTP) by emulating a USB keyboard. USB keyboards are supported by al-most all modern computers; the token requires no additional client software.

3- Affordable, Secure and Durable: The modified security token is practically indestructible; it does not contain any moving parts or batteries. The token strengthens the relationship between usability, security and cost; offering great security with minimal cost and hassle.

4- Reliable: This security protocol can be used within companies, institutions, universities and institutes, because of itsapplicability and simplicity.

This paper is organized as follows: Section II is about computer security, In section III some related work is explained, and in section IV the research methodology is shown, while sections V and VI is a full description about the modified protocol and how it works, Section VII mentioned the found results, and finally, In Section VIII the paper is concluded.

1.1 Computer Security

The synopsis "CIAAN" briefly summarizes the whole computer security services, Confidentiality, which provides protection of data from unauthorized disclosure, Integrity which is the assurance that data received is exactly as sent by an authorized entity, Authentication which is the assurance that the communicating entity is the one who claimed, Access Control which is the prevention of the unauthorized use of a resource, and finally, Non-Repudiation which is the protection against denial of service attack (Stamp, 2006; Stallings, 2011; Paar & Pelzl, 2010).

In this paper we will concentrate on the authentication security service. Authentication is one of the most important services of computer security that allows the authorized users to access their data. User authentication insures the allowed users by one of the authentication methods that are most often evaluated in terms of effectiveness (i.e. ability to reject unauthorized and accept authorized users), cost and user acceptance (i.e. to what degree the users find the method acceptable). The main issue related to user authentication is balancing security with ease of use (Paar & Pelzl, 2010; Duncan, 2001).

Research indicates that 97% of organizations use passwords as a mean of user authentication. Furthermore, numerous researchers have concluded that users do indeed trust these systems. Failings in user authentication cause large financial loss; in 2007 US$3.2 billion was lost to hacking (Paar & Pelzl, 2010; Blanchet, 2008). The consequences of its failing involve losing of privacy and money. Furthermore, it has become an essential basis for trust in business relationships. Authentication establishes trust by proving the identity of a participant in anycommunication, or any transaction. Simply, authentication solutions within the enterprise are designed to ensure that a person is the one who he/she claims to be.

Many researches have studied the authentication as one of the main computer security services. Generally Authentication is the process of ascertaining that somebody really is the one who claims to be. User authentication has two steps, Identification step which specifies identifier (as granting users unique IDs) and the Verification step which binds entity (User) and identifier (as passwords) (Paar & Pelzl, 2010; Duncan, 2001).

The main idea here is how the user can be authenticated. The three independent authentication factors are:

• Knowledge factor, which is something the user knows, like PIN and passwords.

• Possession factor, which is something the user own, like keys and tokens.

• Biometric factor, which is something the user is, like fingerprints and handwriting characteristics, this factor usually used with smart phones biometric authentication.

Network authentication is the process of verifying the identity of the user or computer attempting to connect to the network, the server must receive proof of identity from the user or computer in the form of credentials. Depending on the authentication modules used on the server, there are some basic used authentication modes, Such as Password-Based Authentication, One-Time Passwords-Based Authentication or a combination between

passwords' mode and OTPs' mode which creates a high level of security and generates strong authentication techniques. Other one is the Token-Based Authentication which are cards and USB tokens.

A biometric authentication technique attempts to authenticate the user based on unique physical characteristics. These include static characteristics, such as fingerprints, hand geometry, facial characteristics, and retinal and iris patterns; and dynamic characteristics, such as voiceprint and signature.

1.2 Related Work

The literature study in this section investigates the user acceptance of the three factors of authentication; Knowledge, Possession and Biometric, when they are used for user authentication and not for user identification; which means to verify an identity claimed by a user, not to identify a user. Determining if a user is a valid or invalid user of a system is the art of user authentication. User authentication is a two-step process: First the system needs to identify the user; second, that user's identity needs to be verified. The user is authenticated using something the user knows (such as a password), something the user has (such as a key or o tokens) and something the user is (such as fingerprint and biometric characteristics). The central issue related to user authentication is balancing security with ease of use. One wishes to maintain high security while not making the process of authentication too cumbersome and complex (Stallings, 2011; Paar & Pelzl, 2010).

A user possesses for the purpose of user authentication generally is called USB tokens; now examine two types of tokens that are widely used, which are cards and USB to-kens.

Authentication cards include the following four major types: The oldest is Embossed/Raised characters only (e.g. Old credit card), others are Magnetic stripes with Magnetic bar on back (e.g. Bank card, bank clerk and cashier), others are Memory ones that have electronic memory inside them which stores but not process data (e.g. Prepaid phone card), finally the recently used ones which are the Smartcards that have electronic memory and processor inside them (e.g. Biometric ID card) (Ivarsson, 2012).

The term "USB Security Token" refers to physical hardware device or refers to software token, that authorized users, the USB security token is used to prove people's identity electronically, the token is used in addition to provide the password in order to prove that the user is the one who claims to be. Hardware tokens are typically small enough to be carried in a pocket. The USB token is a good example of Multi-factors authentication.

USB token is a good alternative to the smart card because it is small, inexpensive and it has the same functionality as a smart card, but connects to the existing USB port on a computer; hence it does not need a specific card reader (Steel, 2013; Ivarsson, 2012), The main famous example of this type nowadays is the Yubikey USB security token (Robert & Graham, 2013).

This is a USB contains a smart card chip inside it; this chip provides the functionality of both USB tokens and smart cards. They enable a wide range of security solutions and provide the abilities and security of a traditional smart card with-out requiring a unique input device, the gain of using such tokens is that USB has become somewhat of a standard interface; hence no additional hardware is required. Furthermore, USB is faster in comparison to conventional smartcards readers (Hoon& Ronnie, 2011; Attridge, 2002).

The concept of Two-Factor authentication also applies to Challenge-Response authentication approach. Such approaches enable a user to be authenticated without transmitting the actual base secret. This is achieved by having the server issue a challenge to the user, e.g. enter a number sent by the server in the token. The token generates a One-Time-Password using the issued numbers and the base secret (the latter encoded in the token). The same computation is done in the server-side. If the results on both sides match the user is granted access (Rosvall, 2005).

The goal of using One-Time-Password is to eliminate the re-using of previously used passwords; as such passwords are only valid for a limited time period, which is the most important shortcoming; related to static passwords, that is addressed by OTPs. They are becoming increasingly common in e-commerce and e-banking (Blanchet, 2008).

An encryption algorithm and key are needed to encrypt and convert the data on the USB tokens, those algorithms are divided into the following types: Symmetric Algorithms; If the same key is used for both encryption and decryption, that key is called a secret key and the algorithm is called a symmetric algorithm (Nigel, 2013; Mohan & AndRaji, 2001; List, 1999), such as: DES, Triple-DES and AES (Mohan & AndRaji, 2001).

If different keys are used for encryption and decryption, the algorithm is called an asymmetric algorithm. Asymmetric algorithms involve extremely complex mathematics typically involving the factoring of large prime numbers. Asymmetric algorithms are typically stronger than a short key length symmetric algorithm. But

because of their complexity they are used in signing a message or a certificate. They are not ordinarily used for data transmission encryption (Stallings, 2011; Paar & Pelzl, 2010; Nigel, 2013).

The most well-known asymmetric algorithm used in security tokens is RSA.

Effective security system planning takes into account the need for authorized users to access data easily, while considering the many threats that this access presents to the security services of the information. The security tokens mentioned before, affect many threatened problems.

All data and passwords stored on a card can be erased or modified, others are Differential Power Analysis (DPA). In some types of security tokens, they found passwords or secret words in plaintext on hard drives. And both hardware and software tokens are vulnerable to man-in-the-middle attacks and simple phishing attacks.

Many popular security and encryption devices were broken; In 2011 RSA Inc was hacked and secret information about RSA's SecurID token was stolen which allows to crack them.

On the other side, in 2010 it was revealed that AES-256 encrypted and FIPS 140-2 Level 2 certified USB storage devices of the following vendors could be easily accessed by using a default password: Kingston, SanDisk, Verbatim, MXI, and PICO.

2. Method

The modified authentication protocol involves three actors: the user, the service and the verification server. The user can have access to the service if he provides its own valid OTP generated by the security USB token; the validity is verified by the verification server. The protocol bases on using an OTP authentication mode in addition to a token based mode. The famous structure of the USB token with OTP is Yubikey security protocol; therefore we will explain it in this section in details before discussing our proposed protocol.

The Yubico Yubikey is an example of authentication device capable of generating OTPs. The Yubikey is connected to a USB port and can identify itself as a standard USB HID (Hardware Interface Device) keyboard, which allows it to be used in most computer environments using the system's native drivers.

The Yubikey comprises an integrated touch-button that triggers the generation of the OTP. The generated OTPs are sent as emulated keystrokes via the keyboard input path, thereby allowing the OTPs to be received by any text input field or command prompt.

The Yubikey has been widely adopted by companies and universities. The most important companies are: Microsoft which uses it to increase security for cloud platform, Windows, Facebook, Symantec, Fedora and Google, which offers to clients of Google Apps for Business the possibility to login securely using this device.

The basic OTP generation scheme in Yubikey can be generally described as:

1.When the Yubikey token is inserted into the USB port. The computer detects it as an external USB device.

2. Then the user should press the Yubikey's OTP generation button.

3. Internally, a byte string is formed by concatenation of various internally stored and calculated fields, including a non-volatile counter, timer and random number.

4. The byte string is then encrypted with a 128-bit AES key.

5. The encrypted string is converted to a series of characters that are resulted as keystrokes via the keyboard port.

The generated string of keystrokes is then typically sent out via an input dialog or to a server or host application for verification. The basic steps for verification can be generally described as:

1. The received string is converted back to a byte string.

2. The byte string is decrypted using the same (symmetric) 128-bit AES key.

3. The string's checksum is verified. If not valid, the OTP is rejected.

4. Additional fields are verified. If not valid, the OTP is rejected.

5. The non-volatile counter is compared with the previously received value. If lower than or equal to the stored value, the received OTP is rejected as a replay status.

6. If greater than the stored value, the received value is stored and the OTP is accepted as valid.

The authentication protocol of the Yubikey involves three actors: the user, the service and the verification server. The user can have access to the service, if he can provide its own valid OTP generated by the Yubikey; the validity is verified by the verification server.

The authentication protocol consists of two messages exchanged between X and Y. The first message sent from the service to the verification server with its identification number, a nonce and the OTP received from a user. The second message is the response from the server after the verification of the password is made and consists of the password, the validity status; the nonce sent by the client and a HMAC over these fields using a shared key between the client and server.

In this section, an explanation of the two phases, the generation of the password and the verification process will be discussed.

The OTP is a 32 character passwords that can be verified only once inside a server in order to gain the permission to access specific service. A request for a new authentication token is triggered by touching a button that is on the Yubikey device. As a result, some counters that are stored on the device are incremented and some random values are generated in order to create a fresh 16-byte plaintext having the following six concatenated fields, as shown in Figure 1.

1. Unique Device ID: 6 bytes.

2. Session Counter: 2 byte. It is the first counter and it stores the number of authenticated tokens generated during a session.

3. Timestamp: 3 bytes. At each plug in it is set to a random value and afterwards it is incremented by a clock in order to track the period of time the device was connected during a session use.

4. Token Counter: 1 byte. It is a second counter and it is initialized to 1 at first time and incremented at each plug in of the device.

5. Pseudo-random: 2 bytes. Field used to store a random number generated using as seed the touch button sensor USB activity.

6. CRC-16-value: 2 bytes. Stores the checksum computed over the entire token excluding the CRC field.

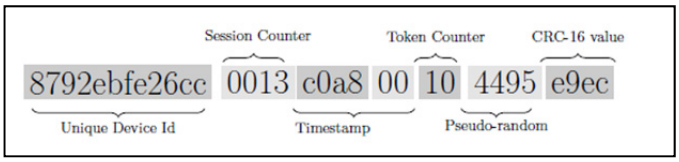

Figure 1. Structure of the Yubikey OTP.

The use of the session and token counters, timer and the pseudo-random field, ensures that each plain text is unique. After the generation, the plaintext is encrypted using the AES algorithm with a unique secret key that is stored on the device and also known by the server. The result is encoded in a modified hexadecimal coding, called Modhex. This coding uses only the characters that have the same position in all keyboard layouts in order to make the device independent of the language settings of the operating system. The result is a 32 character long password that can be sent to be verified against the server.

The verification server has two databases in which it stores information about the Yubikey users and the services that are registered for verification. For each Yubikey de-vice, there is a record in the Yubikeys table with the shared secret key, the values of the fields of the last received OTP and the value of its own time when it is received the last password. The database of the services is used to store a unique id of the service and a secret shared key that is used for signing the challenged password concatenated with its validity status. The key shared between the service and the verification server is obviously different from the secret shared key of the Yubikey device and the server. The service has no information about the Yubikey's configuration. The user sends the OTP generated by a Yubikey device to the service which forwards it to the verification server.

Each dynamically generated OTP, as above, can be concatenated with an optional static public id of a maximum length of 128 bits which can be used to extract the appropriate AES key from the server's database to decrypt the password.

The verification of an OTP is performed by comparing the received values with the ones stored in the database, after the decryption was made. The following cases are possible:

• Valid OTP, database is updated with the new values if:

- The session counter has a bigger value that the last one received.

- The session counter has the same value, but the session use counter is incremented.

• Invalid OTP, if:

- The unique device id is incorrect.
- The session counter is smaller than the value stored.
- The session counter equals the value stored, but the session use counter is smaller.

In order to prevent a phishing attack of the form in which the intruder asks the user for an OTP, that is used later to gain access to the service. The time elapsed between two OTPs generations can be verified. This is possible due to the fact that the server re-members its own clock value at the moment he received a new valid OTP from the user and the database contains the timestamp value of the last valid OTP. Comparing the two differences of time it can be prevented an attack of this form. In our protocol we will ignore this additional verification step and we will consider the value of the counters.

After the verification is performed, the server sends to the service the OTP, the result, the nonce received together with a HMAC of these fields generated using the secret shared key with the client. If the OTP is valid, then the client will grant access to the user.

2.1 The Modified Protocol of Usb Token

Firstly, using Smart-Card USB based token rather than USB token (because of its less power consumption, thinner and still no need for card reader), as motioned previously in chapter two.

Secondly, the modified security protocol covers three authentications factors (Multi-factor authentication) rather than the two authentication factors that are originally used in Yubikey. The modified work's sequence goes through the following steps:

1. The token should be connected to the computer via the USB port. (Level one authentication factor: Possession; Something you own).

2. The user should insert his PIN code. (Level two authentication factor: Knowledge; Something you know). Note: This is not used in The Yubikey protocol, which allows the token to start generating the OTP without verifying who plugged it. Moreover; there is no need to store any unique device ID, This allows the token to be used by any user PIN.

3. The server will check the user inserted PIN, if it is accepted; a random value will be sent to the user phone number, the random value will be randomly derivative from a presently time value. Otherwise the server will block the token after three wrong trials of entering the PIN.

4. The user then should insert the received random value. Note: this will apply the biometric authentication used with the smart phone. (Level three authentication factor: Biometric; something you are).

5. The server finally verifies the user by comparing his inserted PIN with the sent random related to his smart phone number which is stored in the data base, In the modified algorithm, the OTP will use the user’s PIN rather than the device ID; which is 6 byte, while the user PIN will be 2 byte; 4 characters (216 = 65536 unique user). Note: When using this structure the OTP will be shorter by 4 byte. The server will block the token after three wrong inserting tries of the random.

The basic OTP generation scheme in this protocol can be generally described as:

1. The token is inserted into the USB port. The computer detects it as an external USB.

2. The user inserts his PIN.

3. A time-based random value will be sent to the user smart phone.

4. The user inserts the random.

5. Internally, a byte string is formed by concatenation of various stored and calculated fields, including as a non-volatile counter, a PIN and a random number.

6. The byte string is encrypted with a 128-bit AES key.

7. The encrypted string is converted to a series of characters that are outputted as keystrokes via the keyboard port. As shown in Figure.2.

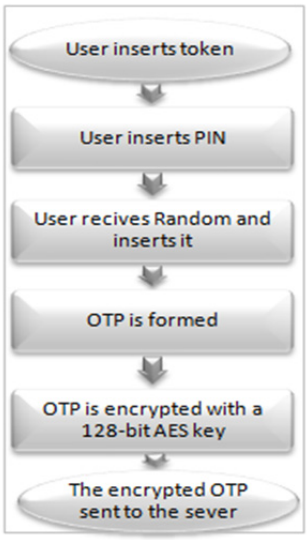

Figure 2. The User Side

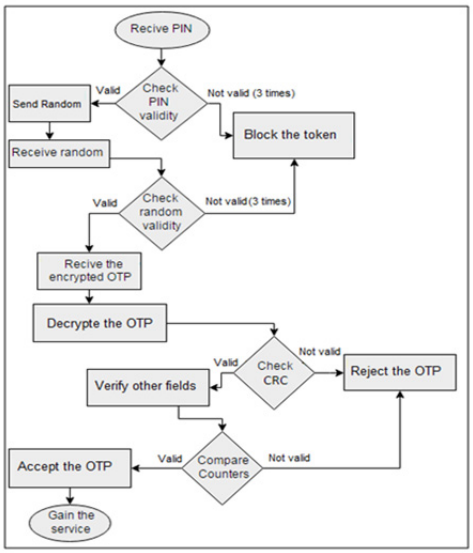

Figure 3. The Server Side

The generated string of keystrokes is then typically sent via an input dialog to a server or host application for verification. The basic steps for the OTP verification can be generally described as:

1. The received string is converted back to a byte string.

2. The byte string is decrypted using the same (symmetric) 128-bit AES key.

3. The string's checksum is verified. If not valid, the OTP is rejected.

4. Additional fields are verified. If not valid, the OTP is rejected.

5. The PIN and random were previously compared while insertion, If the sent random not equal the stored one for this phone number, the received OTP is rejected as a replay status.

6. The non-volatile counter is compared with the previously received value. If lower than or equal to the stored value, the received OTP is rejected as a replay status. If greater than the stored value, the received value is stored and the OTP is accepted as valid. As shown in Figure 3.

2.2 Prototype Implementation

The generated OTP is a 9 byte string; 18 characters, (While Yubikey is 32 characters). Request for a new authentication token is triggered by plugging the Smart-card-USB based device on a computer USB port.

When plugging the token, the user is asked to enter two required fields:

1. PIN: Which the user knows, it is a 4 character (2 byte).

2. Random: 4 characters (2 byte) are sent to the user's smart phone.

As a result, some counters that are stored on the data base are incremented in order to create a fresh 9-byte plaintext having the following concatenated five fields, as shown in Figure.4, knowing that this clear OTP is valid just for 30 seconds in order to keep the high level of security.

1. Unique PIN: 2 bytes.

2. Session Use: 2 byte. It is the first counter and it stores the number of authenticated tokens generated during a session.

3. Token Counter: 1 byte. It is a second counter and it is initialized to 1 at first time and incremented at each plug in of the device.

4. Pseudo-random: 2 bytes. Field used to store a random number which was sent to the user's smart phone when asked for it.

5. CRC-16-value: 2 bytes. Stores the checksum computed over the entire token excluding the CRC field.

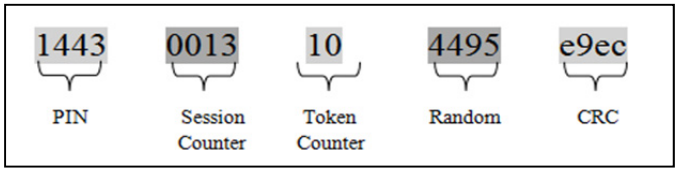

Figure 4. The modified structure of the OTP

2.3 Security Measurements

To measure the security level for any hardware security token, we must firstly detect the strength of the used security protocol; which depends mainly on [19, 21, 22]:

1. Key: As well as it's resistance to cryptanalytic attacks. We say an algorithm is broken if there exists an attack better than brute force. For security tokens, the design concept for key affects the performance very much. In our modified protocol we will use the AES with 128-bit key size.

2. Randomness: This is the used data randomness degree. In the modified protocol we will use 2 byte time-based pseudo-random field in order to generate the OTP.

3. Time to Be Broken: Algorithm strength will be proportional to the time cryptanalyst has taken to break your encryption. The "strength" is defined as "the amount of work needed to “break the algorithms”.

4. Algorithm Complexity: Strength in terms of used operations is a much more concrete measure and the mathematical proof for standardization of security provided by that algorithm. In the modified protocol we use only one round of encryption because we focus on other sides of security.

5. Algorithm Knowing: The algorithm should be known to public; but the key needs to be confidential, knowing the algorithm should not allow the cipher-text to be broken. Knowledge of the algorithm should not reduce the strength of the cipher, but will substantially reduce the resources. In the modified protocol we use AES algorithm which is one of the most famous security algorithms.

6. Algorithm Speed: Response time for Encryption or Decryption (depends on the used system and the number of transferring packets).In the modified protocol we use only one round of encryption and decryption by 128-bit key length (longer keys are slower), and minimum packets' transferring.

7. Especially for using OTP generating:

• Must be allowed to be used only once.

• Must only be usable for the user it was created for.

• Must only be sent via HTTPS.

• Should have an expiry limited time.

8. Response Time: Use of this type of measure usually requires a baseline assessment of existing average response times. This average should be compared with a benchmark or desired standard.

9. Program Efficiency (Program Goal): This outcome measure is intended to capture the cumulative effect of individual process efficiency initiatives (outputs). A typical long- term goal might be to limit overall security program cost increases to a variable percentage per year. The results of individual efficiencies must be tracked, recorded, and summed.

10. Resource Requirements: These measures track the resources required to accomplish the security program mission.

11. Efficiency: NIST; National institute of Standards and Technology, considers AES to be efficient on smart cards. Smart cards with 8-bit CPUs tend to be used in applications that are not speed critical. It is usually the I/O, or other system components, that is the limiting factor in performance and not the cryptography. Then, the speed of almost all the candidates on common 8-bit CPUs seems to be acceptably fast. There are some applications that require fast encryption operations on an 8-bit CPU.

12. Power consumption: All vendors tend to have less power dissipation within any electronic device, and especially for any attached devices to not have an independent and separated power source.In the modified protocol, the device receives power through the USB connection, and does not need any external power supply or a built-in battery.

Our modified protocol is based on AES cryptography algorithm, For security measurement, the strong algebraic structure of the AES cipher has led some researchers to suggest that it might be susceptible to algebraic attacks. However, such attacks have not been shown to be effective. Related key attacks can clearly be avoided by always selecting cryptographic keys independently at random. A bi-clique technique can be applied to the cipher to reduce the complexity of key search. For example, attacker can break AES-128 with 2126.2 encryption operations and 288 chosen plaintexts.

High speed and low RAM requirements were criteria of the AES selection process. Thus AES performs well on a wide variety of hardware, from 8-bit smart cards to high performance computers. On a Pentium Pro, AES encryption requires 18 clock cycles per byte, equivalent to a throughput of about 11 MB/s for a 200 MHz processor. On a 1.7 GHz Pentium M throughput is about 60 MB/s.

2.4 Work Description

The work was fully implemented using C# programming language as a software frame-work in (.NET framework 4 client profiles) with (Windows Application) output type in the both sides; user side and server side.

The C sharp (C#) programming language is a multi-paradigm programming language developed by Microsoft within its .NET initiative and later approved as a standard by ECMA (ECMA-334) and ISO (ISO/IEC 23270:2006), It is defied in their website as:

“C# (pronounced "C sharp") is a programming language that is designed for building a variety of applications that run on the .NET Framework. C# is simple, powerful, type-safe, and object-oriented. Visual C# is an implementation of the C# language by Microsoft and it is supported by Microsoft Visual Studio”.

The system’s used variables appearing in the code almost with their real means, such as; PIN, OTP, session, phone-num, token and ID. Some of them were inserted by the project’s administrator and the user, while others are generated during run time.

As a matter of use, the system’s users divided into two major types; the system’s administrator and the user. The administrator’s main responsibility is the users’ and the tokens’ identification.

The hypothesis was: “The modified security token improves the relationship between security, usability and cost by offering high level of security with minimum cost”.

1- Security Tests:

The tests focus firstly on security level, the strong multi-factor authentication; that com-bines the hardware authentication token with a PIN and a biometric authentication used in the smart phone. This combination covers all the three authentications’ factors; Knowledge factor which is something the user knows (PIN value), Possession factor which is something the user has (Security token) and Biometric factor which is some-thing the user is (used in smart-phones).

Moreover; using the user’s PIN rather than the device ID will allow the token to be used by any user, because it doesn’t store any unique device ID and gives the verifying as PIN when plugged it.

The protocol was depending on AES cryptography algorithm, for security measurement, the strong algebraic structure of the AES cipher has led some researchers to consider AES as the strongest security algorithm.

2- Usability Test:

On the other hand, the test focuses secondly on usability, the modified security token works by emitting an OTP by emulating a USB keyboard. As USB keyboards are sup-ported by almost all modern computers then the token doesn’t require any additional software.

Moreover, the modified security token system was durable and affordable; it is practically indestructible; it does not contain any moving parts or batteries. Use of an existing mobile device avoids the need to obtain and carry an additional OTP generator.

3- Cost Test:

The test for cost especially focused on time and size. For required time, a stop-watch function was used to one complete user authentication process, The PIN and the USB was pre-identified; because this test is a measurement of the needed method’s time.

The time needed to authorized one user (without alerting the user and the server of this acceptance; just open the connection) was around (10 seconds).

As a result of this hypothesis testing, although of those security, usability and cost tests, the modified security

protocol is still keeping aware of the central issue related to user authentication which is balancing security with ease of use; by keeping the high security level while not making the process of authentication too cumbersome and complex.

3. Results

This chapter includes the results found in using the modified USB security token com-pared to the existent Yubikey token, the comparison based on some important factors, which are: security, time and size.

1- Security:

As mentioned previously, the security of the modified security token shows higher level and better performance in almost all aspects of security measurements, this was shown minutely in the next table, as shown in Table 1.

Table 1. Security results

Security Aspects	Yubikey Token	Modified Token
Number of authentication factors	Two	Three
User Identification(Token's plugger)	Un-known	Known
Attempts to access	Un-limited	Limited to three
Authentication mode	One-Time Password	One-Time Password + Password-Based Authentication (PIN)
Building OTP	Any token owner	Just pre-identified user
Security Algorithm	AES	AES
Security Algorithm's Complexity	One round of encryption/decryption	One round of encryption/decryption
Key Size	128-Bit	128-Bit
Randomness	Regardless the time	Regarding the time
Stored data in the token	Token ID	Nothing
Plaintext Transferring	Nothing	Nothing
OTP Validation	Un-Limited	Valid to 30 seconds
Token Validation	Un-Limited	Limited to 100 times
Code Source	Open	Preserved

2- Time:

As mentioned previously, knowing that the time needed to accept one user in the modified token is almost (10 seconds), as shown in Table.2 and Figure.5, this time seemed to be trivial in comparison with the time needed to accept one user in Yubikey, which was around (26 sec).

Table 2. Time results

Number of user \ Time	Yubikey	Modified Token
One user	26.34 second	10.1 second
Time-saving ratio	61.65% (16.24 saved second)	
100 user	2833.1 second	1198.33 second
Time-saving ratio	57.7% (1634.77 saved second =27 minutes)	

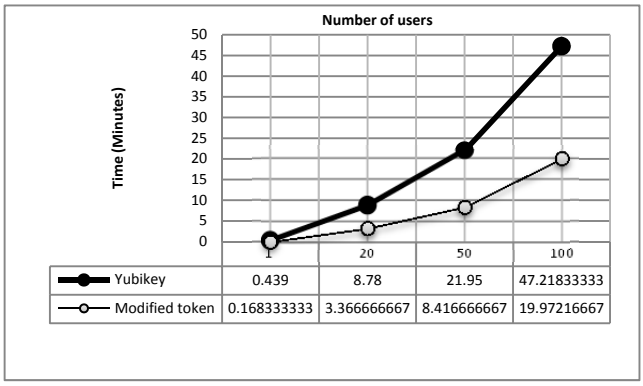

Figure 5. Time results

Moreover, the size for the plain text of the OTP before encryption will directly affect the required time, the size of OTP in Yubikey system is 16 byte, while the size of OTP in the modified system is 9 byte, as shown in Table 3.

Table 3. OTP size results

OTP size	Yubikey	Modified Token
	16 byte	9 byte
Size-saving ratio	43.75 % (Smaller with 7 byte)	

3- Code Size:

As mentioned previously, the size of the whole project in the modified system was nearly (3.7 MB), while it is (2.9 MB) in the Yubikey project, as shown in Table.4 and Figure.6.

Table 4. Size results

Size	Yubikey	Modified Token
Source Code	102 KB	156 KB
Server-side project	196 KB	209 KB
User-side project	2.8 MB	3.5 MB
Whole project	3 MB	3.7 MB
Size-consuming ratio	23.33 % (Larger with 0.7 MB)	

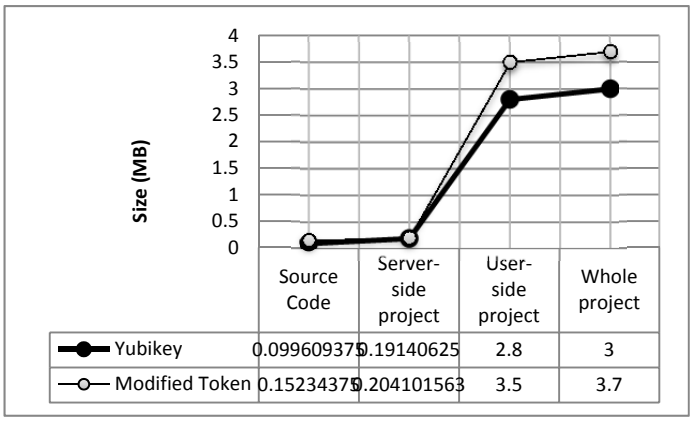

Figure 6. Size results

4. Discussion

The results show that the modified security token offers better performance in almost all comparison factors, except the size factor.

The manifest difference in security level due to a strong security approach using in the modified system.

The obvious difference in time relates to reading data from the token in Yubikey sys-tem; the Yubikey's approach depends on extracting the Device-ID directly from the to-ken while it is a different approach in the modified system and surly this valuable improvement leads to less RAM requirements. Moreover, the smaller OTP length leads to less packet transferring through the network.

The inconsiderable difference in code size relates to identification's code of user-PIN and token before using.

As mentioned previously, the modified system uses Smart-Card USB based to-ken rather than USB token because of its less power consumption, thinner and still no need for card reader. In addition, the modified token doesn't have a special button; just plugging the Smart-card-USB based token on the computer leads to start the authentication process, while requesting for a new authentication token is triggered by touching a button that is on the device in Yubikey system.

In general, the modified USB security token behaves better than Yubikey tokens.

4.1 Conclusion

In this paper, we have analyzed the user authentication security service using one time passwords authentication mode and we provided an efficient hardware implementation for the proposed security protocol on Visual C#.

Our proposed design is based on three authentication factors, which are: Knowledge factor; by using a PIN to identify user, Possession factor; by using a USB token and biometric factor; by using a smart phone applications to receive a random value.

Our proposed security protocol is a modified security protocol of another one using in Yubikey USB security tokens. Their protocol based on generating a fresh 16 byte OTP by touching a button on the token, the OTP then will be encrypted and sent to the verification server to decrypt it and authorize the user. On the other hand, our proposed protocol starts by verifying the user and the USB token before sending the OTP to the verification server.

The Yubikey security protocol faced some critical weaknesses and important threats; in our proposed protocol we worked to solve those weaknesses. The following are some of those security issues and how we solve it to get higher performance in almost all comparisons aspects:

- Multi authentication factors: by applying the biometric factor within the smart phone.

- User Identification: by using the user PIN rather than the device ID.

- Access attempts: by limiting the wrong trials of PIN insertion and random insertion to three trials.

- Validation: by limit the validation period of the OTP to 30 seconds and the validation times of using the token to 100 times.

More than that, the proposed protocol provides better performance in the size of the used OTP with seven byte reduction in size. Besides that, the USB token will be a smart-card based USB rather than a traditional USB, as a result of this; the proposed protocol's speed will be affected, so; it shows better performance in time too.

In conclusion, the experimental results that we have provided illustrate the efficiency of our proposed design in terms of security, speed, throughput and size.

Therefore our modified USB token has achieved better performance in all aspects of security and time compared with the Yubikey token. In our implementation we have used the email to send the random four digits to the user, while in the real life will send the random using SMS protocol which will be faster also.

As future work, we seek to study the used validation limitations and choosing the perfect ones in order to enhance the results and improve the protocol's performance. In addition of that we are working on using Arabic language within the system to add a spread feature to it.

References

Advanced Encryption Standard (AES) Report, National Technical Information Ser-vice (NTIS), 5285 Port Royal Road, Springfield, VA 22161.

Attridge, J. (2002). An Overview of Hardware Security Modules,Version 1.2f of GSECPracti-cal Assignment or

GIAC Certification.

Blanchet, B. (2008). Automatic Verification of Correspondences for Security Proto-cols,IOS Press Amsterdam, The Netherlands, ISSN: 0926-227X.

Duncan, R. (2001). An Overview of Different Authentication Methods and Proto-cols,SANS Institute.

Google apps business login using Yubikey information. Retrieved from http://wiki.yubico.com/wiki/index.php/Applications:Google_Apps. 64.

Hoon & Ronnie (2011). A Review of Smartcard Security Issues, Journal of Security Engineering Research 11,Chine.

Identifiers and Authentication – Smart Credential Choices to Protect Digital Identi-ty. Retrieved from http://www.smartcardalliance.org.

Ivarsson, J. (2012). A Review of Hardware Security Modules.

List. (2010). Attacking and Fixing PKCS#11 Security Tokens, ACM 978-1-4503-0244-9/10/10 ,Chicago, Illinois, USA.

List. (1999),Performance Comparison of the AES Submissions,University of Cali-fornia Berkeley, Soda Hall, Berkeley, CA 94720, USA.

List. (2008),Efficient Padding Oracle Attacks on Cryptographic Hardware, CRYP-TO'12 ,CNRS & ENS-Cachan, France.

Mohan, H. S., & AndRaji, A. (2001). Performance Analysis of AES and MARS Encryp-tionAlgorithms,IJCSI International Journal of Computer Science Issues, 8(4), No 1, July 2011, ISSN (Online): 1694-0814.

Nigel, P. S. (2013). Algorithms, Key Sizes and Parameters, version 1.0, ENISA under contract P/18/12/TCD Lot K.U.Leuven (BE), University of Bristol, UK.

Paar, C., & Pelzl, J. (2010). Understanding Cryptography, springer,Germany.

Robert & Graham (2013). YubisecureFormal Security Analysis Results for the Yu-bikey and yubihsm, INRIA Project ProSecCo, Paris, France.

Rosvall, K., & Niklasson, K. (2005). User Authentication: Passive USB tokens, an alternative to Passwords, Bachelor of Applied Information Technology Thesis, Re-port No. 2009:064, ISSN: 1651-4769, University of Gothenburg, Sweden.

Schneier, B. (2008). A Self-study course in block-cipher cryptanalysis, Volume 24 Issue 1, Taylor & Francis, Inc. Bristol, PA, USA.

Stallings, W. (2011). Cryptography and Network Security, (Fifth Edition), Prentice Hall, USA.

Stamp, M. (2006). Information Security Principles and Practice, (2nd Edition), John Wily &Sons.Inc, ISBN: 978-0-470-62639-9.

Steel, G. (2013). Analyzing Cryptographic Hardware Interfaces with Tookan, tookan.gforge.inria.fr, France.

Volunteers, Security Tokens, On-Line, AVAILABLE.

Yubico Company. Yubikey official page. Retrieved from http://www.yubico.com/yubikey.

Yubikey Security Evaluation (2009). Retrieved from http://www.yubico.com/references.

Yubikey's customers list. Retrieved from http://www.yubico.com/references.