PHILIPPINE ENGINEERING JOURNAL
PEJ 2012; Vol. 33, No. 1: 51-68

SOUL SYSTEM: SECURE ONLINE USB LOGIN SYSTEM

Kevin Charles Atienza, Rod Xavier Bondoc, Joshua Arvin Lat and Susan B. Pancho-Festin
College of Engineering, University of the Philippines Diliman

ABSTRACT

The SOUL System is a secure online authentication system involving a two-factor authentication scheme that uses a password and an ordinary hardware device as security token.

The three main parts of the system include the website, the ordinary hardware device, and a trusted third party. The website must first be integrated with the web API provided and then registered to the trusted third party website to allow two-factor authentication. The security token is any ordinary hardware digital container that holds files such as BMP and PNG where the user's data are hidden. Examples of possible containers include a USB flash drive, a laptop, a cellular phone, and even a dropbox folder. It must be registered with the trusted third party for it to access the SOUL-System-integrated websites. The trusted third party stores and provides the public keys of both the two-factor-login-enabled websites and the registered security tokens.

The SOUL System ensures a more secure website authentication by adding another requirement to the login and registration processes. Instead of having only a password to log in, the user now requires both a password and the security token to access the website. If any hacker manages to obtain the user's password but not the contents of the security token, he would still be unable to access the accounts. If the hacker manages to steal the security token, the accounts are still inaccessible without the password.

General Terms: Security

Keywords: Two-factor Authentication, Trusted Third Party, USB Token, Web Framework

1. INTRODUCTION

Security is one of the most important factors people consider when browsing the Internet. With the rapid advancement of Internet technologies, security risks and flaws are uncovered over time. Exploitation of these risks affects millions of online users every day.

Sadly, many people who use the Internet are unaware of existing online security threats. In the past years, several government websites have been attacked by hackers. Countless online accounts have been hacked as well and a lot of private information and identities stolen. In addition, these attacks can go as far as stealing money from users' online bank accounts.

Correspondence to: Department of Computer Science, College of Engineering, University of the Philippines, Diliman

Received: April 17, 2012
Accepted: May 31, 2012

One of the most widespread forms of Internet security attacks is password cracking. This involves penetrating a network or system to unlock a resource secured by a password through transmitted data or stored information in the system. The most common forms of password cracking attacks include the dictionary attack, the hybrid attack and the brute-force attack (Shimonski, 2002). These attacks may or may not involve direct access to the user's computer and can be done even without the use of viruses or malwares. Other common attacks include man-in-the-middle attacks, password-sniffing attacks, and keystroke logging attacks. These usually involve the presence and use of viruses and malwares in an unsuspecting computer.

Most websites today implement a one-factor authentication structure using a username and password login scheme. This scheme, however, is prone to password-cracking and keystroke logging attacks. Some websites can be accessed via HTTPS to secure the transfer of data between the user's computer and the web server. The problem with HTTPS is the high cost of SSL certificates, making it impractical for startup businesses and websites. In addition to that, HTTPS does not protect the user from keystroke logging attacks since keyloggers work locally in the user's computer.

The addition of another factor to the authentication scheme increases the security of websites dramatically. The additional factor is something a user owns which he can use for authentication in addition to the password. Many research papers have shown that biometrics can be used as a second-level security factor to identify users from unauthorized people. Unfortunately, biometricsare not 100% foolproof, and specialized hardware is necessary for such system to work.

Little research has been done on the possibility of using secure tokens as additional security measures for websites. A security token, also called cryptographic token is a hardware device used to provide access to authorized users. It may be used in addition to or in place of passwords to electronically prove one's identity. Some of these tokens are already available in the market but at high costs, unaffordable to most users. These tokens also use specialized hardware, making it inaccessible to majority of users.

This study aims to transform any ordinary hardware device container such as a USB flash drive into a security token which can be used for authentication of websites integrated with the two-factor login system. It aims to design and implement a two-factor authentication system. This involves a Software Development Kit for the websites, a USB security token, and a trusted third party website. This research also seeks to replace the existing username-password scheme with a more secure system that is low-cost, reliable, practical, portable, and flexible.

1.1 Problem Statement

The study seeks to design and implement a secure online authentication structure that involves Software Development Kits for several available web frameworks and languages, a secure USB token, and a trusted third party website for information storage. There are currently few studies about online authentication with secure tokens. Some solutions are expensive because of the use of specialized and commercial secure tokens. This study specifically seeks to provide a secure and low-cost alternative that converts any hardware storage device into a security token that can be used for registering and logging into websites. It also seeks to replace the existing unsecure username-password login scheme.

Specific objectives for the design process include the implementation of a protocol that allows secure transfer of unbounded data between two entities (e.g. Local program and trusted third party), and the secrecy of the data in the security token. The proposed system must also be portable and flexible for it to work in most computers and operating systems. It should be practical enough to handle situations where security tokens get lost, corrupted, or stolen.

1.2 Scope and Delimitation

The SOUL System supports any operating system with Java Runtime Environment installed (JRE 1.6+). In addition, the SOUL System SDK supports Java, Python, and PHP languages, providing several functions and shortcuts for most frameworks. The USB flash drive container can be any ordinary flash drive that can store any type and number of files. The USB flash drive container must have at least one image file which is either in 24-bit BMP or PNG format where the user's data is encrypted and hidden.

1.3 Significance of the Study

This research seeks to provide a low-cost and more secure online authentication system that will replace the existing username-password scheme. It is also designed to ease and automate the integration of the login system with existing sites and web frameworks. Furthermore, the proposed system is intended to speed up the registration and login processes to provide better security for millions of websites and online accounts. The success of this research can make the online world a safer place.

With the implementation, release, and public use of the proposed system, the researchers expect less accounts being hacked and stolen. In addition, more businesses that have online websites will thrive since the system is cost-effective and practical to use. Finally, the more expensive HTTPS may be replaced by the proposed secure authentication system.

2. RELATED LITERATURE

2.1 Cryptography

Cryptography is the science and study of techniques concerned with information security. It involves several components of information security including confidentiality, data integrity, authentication, and non-repudiation (Menezes et al., 1997).

2.2 Two-factor Authentication

Two-factor Authentication is an authentication scheme involving two kinds of evidences or factors to verify user identity. This scheme aims to improve security by requiring something the user knows and owns. This often involves password and other entities such as bookmarks, biometrics, or other security tokens (Adida, 2007; Jin et al., 2004).

2.3 Cryptographic Algorithms

Symmetric Key Algorithm. Symmetric Key Algorithm involves the encryption and decryption of plaintext using the same key (Denning 1984). AES or the Advanced Encryption Standard is one popular example of this algorithm (Grembowski et al., 2002).

Asymmetric Key Algorithm. Asymmetric Key Algorithm involves the encryption and decryption of plaintext using two keys: one public and one private. The RSA scheme is an asymmetric key algorithm developed by Rivest, Shamir, and Adleman. Its cryptographic strength relies on the difficulty of factoring very large numbers (Denning,1982).

The symmetric and asymmetric key algorithms both have advantages and disadvantages. Symmetric key algorithms can encrypt and decrypt large amounts of data very quickly compared to asymmetric key algorithms. Their keys are also much shorter than that of public-key algorithms. The problem with symmetric-key ciphers is that the shared key must remain secret to communicating sides. The public key in asymmetric-key algorithms can be exposed and used to transfer data securely without the danger of exposing the value of the corresponding private key. Many public-key schemes have more efficient digital signature processes compared with shared-key schemes. To make good use of these advantages and disadvantages, hybrid cryptosystems involve the use of public-key schemes to transfer the shared key which can be used for symmetric encryption and decryption. (Menezes et al., 1997).

Digital Signatures. Digital Signatures allow a person to attach his identity to transferred data. It converts data into a signature which can be verified by the receiving entity. Digital signatures are possible through public-key encryption schemes (Menezes et al., 1997). The sender signs the data using a private key; then, the receiver verifies the data using the sender's public key. As long as the sender is the only holder of the private key, the attacker cannot fake the digital signature.

Hash Functions. Hash functions are one-way algorithms used to convert readable information into unreadable data. SHA-512 offers better security compared to the other hash functions from the same family (Grembowski et al., 2002).

Key Establishment. Key Establishment refers to the process of ensuring the availability of a shared key to communicating parties for symmetric cryptography. Key management, on the other hand, refers to the set of processes for the maintenance and establishment of keys and key relationships between communicating parties.

2.4 File Systems

The file system is that part of the operating system that deals with files. It is used to efficiently organize data. It also ensures the reliability of data used by a device or system. The existence of a file system is due to three essential requirements for long-term information storage. These include the following: (i) it must be possible to store a very large amount of data/information, (ii) the data must survive the termination of the process using it and (iii) multiple processes must be able to access the information concurrently (Tanenbaum, 2007).

Moreover, there are special types of file systems called Cryptographic File Systems. These encrypt the data before it is stored in the device. Some cryptographic file systems store encrypted files directly while others serve as device drivers that contain a traditional file system on top of it. Each of these approaches has its pros and cons. The types of cryptographic file systems include the following (Dubrawsky, 2010):

1. Volume encryptors - This type of file system uses the device driver layer to encrypt and decrypt information to and from a physical disk. Moreover, this type encrypts the whole drive and is convenient to use for its transparency to the end user. On the other hand, this type does not grant fine-grained access control to individual files or directories.

2. File Encryptors - This type of file system works on the presentation layer and provides an end-to-end file encryption. However, it does not scale well with large storage systems.

3. File System Encryptors - This type of file system allows a per-file or per-directory encryption using a single encryption key.

2.4.1 Existing Implementation of Cryptographic File Systems

Cryptographic File System for Unix. The Cryptographic File System(CFS) is a type of secured file system developed by Matt Blaze of AT&T Bell Laboratories. CFS provides user a system level secure storage through UNIX's standard file system interface. CFS allows users to secure an encrypted file under a normal directory by using a cryptographic key which temporarily decrypts the file into plaintext while the user works with it (Blaze, 1993).

Encrypting File System for Windows. The Encrypting File System(EFS) is a new feature introduced in NTFS version 3.0. It provides the user a file system level encryption using a public-key-based scheme. A fast symmetric algorithm encrypts a file using a randomly generated encryption key. Moreover, this key is encrypted using one or more public keys from a user's version 3.0 certificate (Dubrawsky, 2010).

NCryptfs. The NCryptfs was developed to ensure the confidentiality of data while it aims to balance security, performance and convenience. This serves as a security wrapper that is bound to a directory which stores encrypted data. Moreover, the encrypted data directory may be on any file system. A plaintext version of each file may be seen using the standard Unix file access API (Wright et al., 2003).

Transparent Cryptographic File System. The Transparent Cryptographic File System(TCFS) is a type of secured file system solely based on Sun's Network File System(NFS). This is the most widespread protocol for file system sharing during that time. The TCFS is completely transparent to applications because it works under the VFS (Virtual File System) layer. In addition, it uses the Date Encryption Standard (DES) algorithm to ensure security. The keys are saved in a special database that also stores the user's login credentials (Maurellio, 1997).

StegFS. StegFS is a file system that performs both steganography and encryption. It was developed by Andrew McDonald and Markus Kuhn and is based on a modified EXT2 file system kernel. The main advantage of StegFS is its ability to hide the content of hidden data from attackers. That is, even if they know that there are files, they will not be able to look into their content. (Pang et al., 2003).

2.5 Trusted Third Party

A trusted third party (TTP) is an additional reliable entity which supports protocol completion in secure systems (Pagnia and GA˜ d'rtner, 1999). A TTP can either be unconditionally or functionally trusted. It is unconditionally trusted if it is relied upon on all major matters such as storing secret and private keys. It is functionally trusted if it does not have access to private keys but is still considered fair and trusted (Menezes et al., 1997).

The SOUL System's TTP is functionally trusted. It primarily serves as a storage of public keys of both the website and USB accounts. It also verifies the integrity of hashed passwords and accounts but never storing the actual passwords and private keys of the said entities.

2.6 Web Application Frameworks

A Web Framework is a collection or set of software tools and components that makes the development and deployment of web-based user interface (UI) easier and faster. The user interfaces are delivered to a remote user through the user's web browser. The software on the server is then executed either on the server or the client's browser.

Web frameworks may be classified as server-centric or browser-centric. In server-centric frameworks, the control is embedded in the software residing in the server. On the other hand, browser-centric frameworks rely more on the client-side code for the computations of the details displayed in the user interface (Vosloo and Kourie, 2008).

Popular examples of server-centric Web Application Frameworks include Django, Flask, Web2Py, CodeIgniter, CakePHP, Zend Framework, Symphony, Stripes, spring, and Struts2. On the other hand, popular examples of browser-centric frameworks include JavaScript frameworks such as jQuery, javascriptmvc, Prototype JS, Knockout JS, and Batman JS.

2.7 Steganography

Steganography is a branch of information privacy that seeks to hide the existence of data inside an entity such as images (Menezes et al., 1997). Several researches have allowed steganography to be used with image files such as PNG files and 24-bit BMP files. Steganographic techniques include LSB insertion which hides the information along the pixel data of the images (Dautrich, 2009).

2.8 Cryptographic Attacks

Cryptographic attacks on encryption and protocols can be classified as either passive or active. A passive attack involves invading the transmission of data and threatening its confidentiality. On the other hand, an active attack threatens the confidentiality, integrity, and authentication of the data through the alteration of the transmission on the channel.

There are several attacks on encryption schemes. These include the ciphertext-only attack where an attempt is done to obtain the decryption key or plaintext from the ciphertext. The known-plaintext attack involves the deduction of the decryption key given a list of plaintext and corresponding list of ciphertext. The chosen-plaintext attack involves the deduction of the decryption key given the chosen plaintext and the corresponding ciphertext. The chosen- ciphertext attack involves the deduction of the decryption key where the attacker chooses the ciphertext and obtains the corresponding plaintext. As an example, the chosen-ciphertext attack can be done even if the attacker only has access to the decryption tool in the absence of the decryption key.

There are also several known attacks on protocols. These include the known-key attack where old keys are used to obtain an algorithm to determine new keys. The replay attack involves the recording and replaying of an entire session to hack a system. Impersonation is done when an attacker assumes an identity known by the system. A dictionary attack on passwords involves using a list of probable passwords to obtain matches with their hashed password counterparts in the system (Menezes et al., 1997).

3. METHODOLOGY

3.1 Solution

3.1.1 System Design

The SOUL System involves three major entities: the website using the SOUL **System SDK,** the security token containing the images where the encrypted user data is hidden, and the Trusted Third Party where the public keys are stored. The SOUL System Software Development Kit is available for Java, Python, and PHP. The website need not be accessible via HTTPS to support the system. The security token is any ordinary hardware storage device that can keep the image file with an encrypted hidden data.

The security token and the user's password are involved in the two-factor authentication scheme. Public keys for all the registered websites and the security tokens are stored in the TTP.

The trusted third party hosts the Signed Java Applet. This extracts the hidden data inside the image file, and allows the secure authentication of websites as well as the decryption of data using the user's password. A trusted third party is necessary to prevent certain attacks such as the man-in-the-middle attack which exploits the vulnerabilities in the exchange of public keys. The Signed Java Applet already has the public key of the trusted third party ensuring the secure transfer of data to the trusted third party at the initial steps.

3.1.2 Secure Transfer Data

The data transferred among the three major entities are secured using a hybrid cryptosystem involving a symmetric-key and an asymmetric-key cryptosystem. The asymmetric-key cryptosystem is used initially to transfer a newly generated key. On the other hand, the symmetric-key cryptosystem uses the newly generated key to encrypt and decrypt data of any length for transfer. Private and public keys are used to sign and verify the encrypted data respectively. POST requests are used to send and receive data from websites.

3.1.3 Registration of USB token to Trusted Third Party

In order to obtain a registered security token, the user opens the trusted third party (TTP) page with the Signed Java Applet (SJA). The user selects the image file inside the security token, types in the password, and inputs other personal data to be stored inside the image file. The data to be transferred to the TTP during registration does not contain the user's personal data. After a registration, the TTP has the UUID, username, hashed password, and public key of the security token. The hashed password is stored for future deactivation of accounts.

3.1.4 Registration of Website to Trusted Third Party

The Signed Java Applet (SJA) generates a Public and private key pair to register a website to the trusted third party (TTP). The website will have its own username and password. These data along with the public key are stored inside the TTP. The encrypted website data is stored inside a file that the website server loads during runtime to allow secure two-factor login. After registration, the TTP has the UUID, the username, hashed hashed password, public key of the website account, and other important information such as the website URL.

3.1.5 Registration and Login to Selected Website

To register or log in to a website integrated with the SOUL system, the user needs to visit the website with the SOUL System plugin and mount the security token on the computer. The user, then, selects the image where the encrypted data is hidden and the password is typed for authentication. USB flash drives are usually used as security tokens since they are portable and widely used.

During registration, the user data stored inside the USB token is transferred to the website. If the user has changed the information stored inside the USB token, his data is updated during login. The image file hash where the encrypted data is hidden is checked first by the TTP before allowing any of the processes.

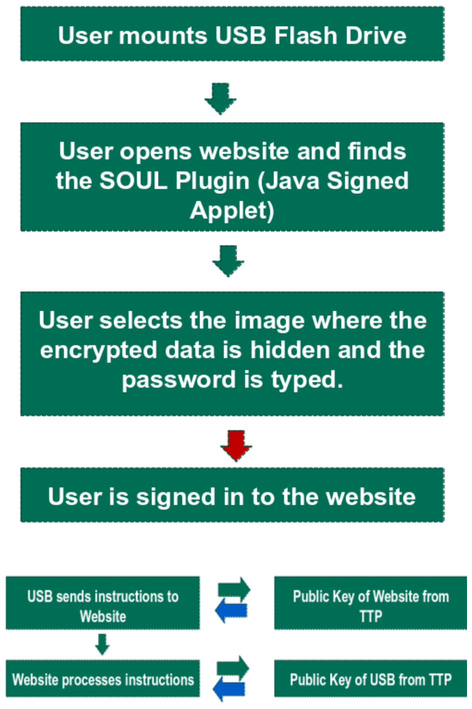
User mounts USB Flash Drive
User opens website and finds the SOUL Plugin (Java Signed Applet)
User selects the image where the encrypted data is hidden and the password is typed.
User is signed in to the website
USB sends instructions to Website
Public Key of Website from TTP
Website processes instructions
Public Key of USB from TTP

Figure 1: Basic System Flow

3.1.6 Account Deactivation

If the security token is lost, gets corrupted, or gets stolen by an attacker, the user can no longer access all accounts connected to the security token. He has the option of deactivating all of the accounts associated with the security token. Deactivation can be done using the TTP's deactivation service. This requires verification with an email using the user's email address. Once the accounts are Deactivated, they can no longer be reactivated.

3.1.7 Backup Key System

The SOUL System supports a maximum of two security tokens per account. This was included as back up in case the keys get lost, stolen, or corrupted. When one of the keys is lost, the other key and the email address can be used to deactivate the lost key while creating a new one as replacement. A verification message is sent to the email address of the user to complete the deactivation and generation process.

3.2 Implementation

3.2.1 Support and Setup

The USB flash drive is assumed as the security token in this study. The SOUL System supports any operating system with JRE. No other Java library needs to be installed in the client computer for website login or registration. Any ordinary hardware storage device can be used as a security token for website authentication. The security token needs to have at least one image file either in PNG or 24-bit BMP format.

The system is designed to support any web application using the SOUL System SDK available for Java, Python, and PHP. The SOUL System SDK contains the cross-language cryptographic library (XLCrypt) that allows cryptographic algorithms to interact smoothly. This includes key generation, encryption, and decryption for the symmetric key algorithms (AES). This also includes key generation, encryption, decryption, signing, and verification for the asymmetric key algorithm (RSA). The following are also included: base-64 encoding and decoding of strings, serialization and deserialization of strings and string arrays/lists, UUID (universally unique ID) generation, and secure hashing of strings (SHA-512).

No other software is required to hide the data inside the security token. The Signed Java Applet can hide encrypted data inside PNG and 24-bit BMP files stored in the security token. The Signed Java Applet will request for the user's password to decrypt the data inside the image file. The Signed Java Applet securely connects with the TTP to verify the integrity of the image file containing the encrypted data.

3.2.2 XLCrypt Library

The XLCrypt Library is the cross-language cryptographic library developed for this project. It currently supports Python, Java, and PHP. It provides language-independent libraries for AES, RSA, SHA-512, UUID generation, base-64 encoding and decoding, and serialization and deserialization of strings and string arrays and lists. For example, a plaintext encrypted with Python code can be decrypted properly with Java code and vice versa. The AES libraries support 256-bit keys while the RSA libraries support both 1024-bit and 2048-bit keys. 2048-bit keys for RSA are used at runtime. The XLCrypt Library for Python is a wrapper library for available cryptographic class's including M2Crypto, UUID, hashlib, and json. The library for Java wraps the available cryptographic classes and packages including Bouncy Castle API, gson, java security classes, and Base64Coder library. The library for PHP wraps the available cryptographic package Phpseclib which is a pure PHP implementation of the needed security algorithms.

The three main classes in the XLCrypt Library are XLCrypt AES, XLCrypt RSA, and XLCrypt Utils. XLCrypt AES provides 3 static functions like encrypt, decrypt, and generate key. XLCrypt RSA provides 5 static functions including generate key pair, encrypt, decrypt, sign, and verify. The generate key pair function returns an array of strings containing the generated public and private keys. All ciphertexts are base-64 encoded strings. The XLCrypt Utils provides static functions such as encode 64, decode 64, serialize, deserialize, hash sha512, and generate UUID. The serialize function converts an array (PHP, Java) or list (Python) of strings into one base-64 encoded string. The deserialize function RST decodes the base-64 encoded string and returns it to its original form. The hash function hash sha512 returns the same hash value for the same string in all supported languages.

3.2.3 SOUL System SDK

The SOUL System SDK includes the libraries available for the supported languages Java, Python and PHP. The available SDKs can be downloaded from the trusted third party sites which can then be easily integrated to websites. The SOUL System SDK contains the XLCrypt library and other tools needed for fast website integration. Shortcut functions are also contained inside the SDK.

3.2.4 Key Management

The Signed Java Applet contains the public key of the TTP along with the code. This ensures the integrity of the public key since altering the contents of the Signed Java Applet while maintaining the signature is impossible. The encrypted data of the user hidden inside the image file contains the private key and public key of the security token. The web server of the site has an encrypted file containing its private and public keys, as well as the public key of the TTP. The encrypted file can be decrypted using the web server's own account password during runtime.

The TTP primarily acts as the storage for the public keys of the registered websites and the security tokens. Registration of websites and security tokens allow storage of public keys, hashed passwords, and hashes of image files which can be later used for verification of the integrity of entities.

3.2.5 Key Establishment

The SOUL System involves an optimized version of a hybrid cryptosystem that uses the public key scheme to transfer the shared key for symmetric cryptography. A POST request with parameter values for the public key encrypted and shared key encrypted messages is initially sent from the source (e.g. Signed Java Applet/Web Server of Website) to the receiver (e.g. TTP). The public key encrypted value contains the shared key used to decrypt the shared key encrypted value. The public key encrypted value is encrypted with the available public key. Only the TTP can decrypt the data which includes the shared key.

The shared key encrypted value contains a hashed unique symmetric label. It is also associated with a shared key that can decrypt subsequent POST requests. Subsequent POST requests rely on the value of the unique symmetric label by sending its hash as well as the message encrypted with the shared key associated with the said hash. The unique symmetric label is replaced and updated during every POST request.

During the transmission of data between the Signed Java Applet and the web server of the website, both parties would obtain a copy of each other's public key from the TTP. These public keys are used to verify the signature of messages from the sender. The Signed Java Applet uses the private key of the security token to sign the encrypted data for the web server. The web server uses the public key of the security token from the TTP to verify the signature received along with the encrypted message. The same goes with the message and signature sent back to the Signed Java Applet for processing.

3.2.6 Registration and Login of token Website

The user opens the website with the SOUL System plug-in, then mounts the security token. The SOUL System plug-in (SSP) requests for the user's password and the image file containing the hidden encrypted data. The SOUL System plug-in securely communicates with the TTP with the hybrid cryptosystem to verify the integrity of the image file using SHA-512. After integrity verification, the TTP returns the public key of the website to the SSP. It is used to encrypt the data to be transferred to the website using RSA.

A shared key, along with other necessary information is sent from the SSP to the website to allow transfer of data of any length between the two entities. The data transferred is encrypted with the public key of the website inside the SSP and then decrypted with its private key stored in the server. Along with the public key encrypted data, the shared key encrypted data is passed in the same request to reduce the number of POST requests to the server. The integrity of the shared key encrypted data source is verified using the public key of the security token obtained from the TTP.

The final GET request to open the web browser and allow access to the user involves a one-time password independent of old one-time passwords. This is done to prevent replay attacks and known-key attacks.

3.2.7 Conversion Data During Transmission

The SOUL System uses dictionaries to store and handle any type and amount of data easily. The dictionary is first serialized into a single string and then encrypted with the appropriate encryption scheme. The encrypted data is then signed with the sender's private key and then it is transmitted together with the digital signature. The receiver verifies the source of the data using the sender's available public key. The encrypted data is decrypted after verification. This is then deserialized back into dictionary form. Since Java does not have built-in dictionary support, an XLDict class for Java has been created for this purpose.

3.2.8 Encryption and Steganography Inside Images

PNG files and 24-bit BMP files are used to store the user's data. The user's data is first encrypted with AES using the user's hashed password and then hidden inside the selected image file. Since the data is hidden with steganography, the unsuspecting user will not be able to notice any difference between the original file and the old file. In addition, comparing the old and the new image files cannot be done because the original file is overwritten.

During the development of the SOUL System, other types of files that support steganography include MP3 and JPG files. Unfortunately, they cannot store the minimum amount of data needed so they were excluded from the system.

3.2.9 Minimum Image File Dimensions

The SOUL System requires a minimum of 10,000 characters to be stored inside an image file. After several tests and calculations to analyze the relationship between the area of the image file and the amount of data that can be stored, several results have been obtained. The 24-bit BMP file needs to have an area of at least 80,036 square pixels (e.g. 283px by 283px) to store 10,000 characters of unencrypted string. The PNG file needs to have an area of at least 26,712 square pixels (e.g. 164px by 164px).

$y = 8x + 36$ is the obtained function relating the area of the image file and the number of characters that can be stored for 24-bit BMP files. The variable "y" represents the area of the image in pixels while the variable "x" represents the number of characters that can be stored. On the other hand, $y = 2.67x + 12$ is the function for PNG files.

3.2.10 Signed Java Applet

The Signed Java Applet acts like an ordinary Java Applet except that it has local access to the user's files in the computer. It is embedded in a webpage and its signature is automatically verified by the user's browser. The Signed Java Applet acts as an interface where the user can log in and register to the TTP and SOUL-enabled websites. It contains the XLCrypt library for Java, the XLClass library, and other utilities needed for secure transmission and storage of data.

3.2.11 Account Deactivation

During account deactivation, the user simply requests for a deactivation email from the TTP to be sent to the user's registered email address. The corresponding user accounts can no longer be used after deactivation. The deactivation code is a one-time password inside a link which when visited will automatically deactivate the user's accounts.

3.3 Evolution of the SOUL System

The architecture of the SOUL System has changed a lot from its initial version that did not include a trusted third party and that involved a cryptographic file system. It now currently includes a trusted third party, a signed java applet, and steganographic techniques inside image files. Some of the latest features included are the backup key system and the optimized hybrid cryptosystem.

3.3.1 Version 1.0

Design Summary: The system involved only the USB Flash Drive and the website supporting the SOUL System. The USB Flash Drive involved an exposed Cryptographic File System Program and an exposed Cryptographic File System File. There is an initial exchange of public keys which is then followed by the transmission of the shared key using the public key.

Design Problem: The system is prone to man-in-the- middle attacks during the initial exchange of public keys. There is no guarantee regarding the integrity of the received public key from the other party. The cryptographic file system programs inside the USB flash drive are exposed and can be seen by any user. Anyone who happens to obtain the flash drive can easily identify the device as a secure token.

3.3.2 Version 2.0

Design Summary: The system now involves a trusted third party that serves as a storage for the public keys of the registered websites and USB accounts. The cryptographic file system programs initially contain the public key of the TTP to ensure their integrity.

Design Problem: The cryptographic file system programs and the cryptographic file system are still exposed inside the USB flash drive.

3.3.3 Version 3.0

Design Summary: The cryptographic file system program is now an executable jar file that encrypts and hides data inside a PNG file using steganographic algorithms. A signed java applet is available online to check the integrity of the jar file residing inside the security tokens.

Design Problem: The cryptographic java program which is the executable jar file is still visible to users.

3.3.4 Version 4.0

Design Summary: The functions of the executable jar file are transferred to the signed java applet. The encrypted data can now be stored inside BMP files.

Design Problem: No major design problems and security issues

3.3.5 Version 5.0

Design Summary: The system now supports the backup key system. The hybrid cryptosystem has been optimized to reduce the number of required POST requests by 1. That is, requests that originally require 2 requests to complete now involves only 1 POST request. Most requests involve only a single POST request.

Design Problem: No major design problems and security issues.

4. RESULTS

4.1 Security

The system has been secured with hybrid cryptosystem and other security features such as UUIDs, message UUIDs, RSA Signing and Verification, and double password hashing. Attacks that include man-in-the-middle attacks, dictionary attacks, and key logging attacks would not work against the current system. The attacker needs to attack and get both the user's password and security token to have access to the user's accounts.

4.2 Integrity

The integrity of the source and targets during the transfer of data are checked and verified in every step of the processes. The Hash of the image file containing the encrypted user data is stored inside the TTP for verification and prevention of image file cloning.

4.3 Portability

The SOUL System works in all operating systems with Java installed. Image files can be stored inside any ordinary hardware storage devices which includes new cellular phones, laptops, and USB flash drives that can keep photos and other files. No new software is needed to log in to the user's accounts.

4.4 Flexibility

The SOUL System SDK, which is available for Java, Python, and PHP, makes the integration of the SOUL System to any website very easy, fast, and flexible. Developers can easily have their own way of managing sessions and user accounts.

4.5 Visibility

The user's encrypted data is hidden inside 24-bit BMP and PNG files using steganography. This means that no SOUL program or SOUL data is visible to unsuspecting people.

5. DISCUSSION

5.1 Vulnerability Analysis

5.1.1 Security of Private Key/Public Key Pairs

It is maintained throughout the system that private key/public key pairs are generated where they are needed. This means that security tokens generate their own key pairs independent of the TTP's servers and vice versa. The key pairs for the security tokens and websites are generated using the Signed Java Applet which works as a local program in the client computers.

Comment [U1]:

5.1.2 Integrity of Public Keys

The Signed Java Applet already has the public key of the TTP so the integrity of the target entity and the transferred data is ensured. Only the holder of the private key can decrypt the data transferred using the Signed Java Applet. With this initial setup of keys, man-in-the-middle attacks are impossible.

5.1.3 Cloning of Image Files

The contents of the image files are updated every time they are used, and the corresponding file hashes are stored in the TTP. If the image file is cloned and the original file is used to log in to the website, the original file's updated file hash will be accepted while the cloned file's hash will be rejected. Only one image file can be used to log in to websites.

5.1.4 Key Size Strength

The system uses 2048-bit keys for RSA, 256-bit keys for AES, and SHA-512 for encryption, decryption, signing, verification, and hashing. These keys are considered secured compared to proven less-secure counterparts such as those from SHA-1 families and lower length keys.

5.2 Fighting Against Known Cryptographic Attacks

The **brute-force attack** is easily prevented with password salting and limited password attempt count.

Collision attacks especially concerning hash collisions are limited with the use of relatively stronger hash functions (e.g. SHA-512). SHA-512 is considered safe against hash collisions compared to SHA-1 functions.

Dictionary attacks are limited with password salting. The hashes of similar passwords are different with the use of salts. The salt is appended to the password and the resulting string is hashed.

Keylogger attacks can only steal the password of the user. The attacker cannot access the account with only the password. The system requires both the password and the image file inside the security tokens to log in.

The known-plaintext attack is impossible because both the symmetric label and the shared key are updated every POST request, so no list of plaintext and ciphertext can be obtained.

Man-in-the-middle attack is prevented with the use of message UUIDs along with the hybrid cryptosystem. The shared key cannot be intercepted unless the attacker manages to hack the TTP and obtain the TTP's private key.

Replay attacks have been prevented with the use of one- time passwords which are independent of old one-time pass- words. One-time passwords are generated using hashed values of UUIDS.

5.3 Conclusion

The design and implementation of the proposed two-factor online authentication system was successful in providing a low cost and secure design alternative to both the username-password scheme and the commercially available secure tokens. It can support any website done in PHP, Java, or Python. It is secured with the use of a trusted third party, a hybrid cryptosystem, one-time password login scheme, and cross-language cryptographic libraries. The developed two- factor login system is not vulnerable to known attacks that can hack the username-password scheme.

5.4 Recommendations

Increasing the number of steganographically supported files would definitely improve the security of the system. Currently available steganographic techniques that hide data inside JPG and MP3 files cannot be used because the image files cannot store all of the user's data.

Supporting more languages such as Ruby is another great addition for the system. With the availability of the SOUL System SDK for Ruby, several Ruby web frameworks such as Ruby on Rails can be supported as well.

6. ACKNOWLEDGMENT

Special thanks to our beloved professors and advisers Susan Pancho-Festin, Philip Zuniga, and Ronald Sarcaoga for guiding us during the development of this research project.

REFERENCES

1. Ben Adida. Beamauth: two-factor web authentication with a bookmark. In Proceedings of the 14th ACM conference on Computer and communications security, CCS '07, pages 48–57, New York, NY, USA, 2007. ACM. ISBN 978-1-59593-703-2. doi: http://doi.acm.org/10.1145/1315245.1315253. URL http://doi.acm.org/10.1145/ 1315245.1315253.
2. Ayushi. A symmetric key cryptographic algorithm. International Journal of Computer Applications, 1(14):1–4, February 2010. Published By Foundation of Computer Science.
3. Matt Blaze. A cryptographic file system for unix. In Proceedings of the 1st ACM conference on Computer and communications security, CCS '93, pages 9–16, New York, NY, USA, 1993. ACM. ISBN 0-89791-629-8. doi: http://doi.acm.org/10.1145/168588.168590. URL http://doi.acm.org/10.1145/168588.168590.
4. Christian Cachin and Nishanth Chandran. A secure cryptographic token interface. In Proceedings of the 2009 22nd IEEE Computer Security Foundations Symposium, pages 141-153, Washington, DC, USA, 2009. IEEE Computer Society. ISBN 978-0-7695-3712-2. Doi: 10.1109/CSF.2009.7. URL http://dl.acm.org/citation.cfm?id=1602936.1603625.
5. Jonathan Dautrich. Multi-class steganalysis, 2009.
6. Dorothy Denning. Cryptography and Data Security. Addison-Wesley Publishing Company, 1982.
7. Ido Dubrawsky. Cryptographic filesystems, part one: Design and implementation. http://www.symantec.com/connect/articles/cryptographic-filesystems-part-one-design-and-implementation, November 2010.
8. Martin Fielder: Real steganography with true-crypt. http://keyj.emphy.de/real-steganography-with-truecrypt/, February 2011.
9. Tim Grembowski, Roar Lien, Kris Gaj, Nghi Nguyen, Peter Bellows, Jaroslav Flidr, Tom Lehman, and Brian Schott. Comparative analysis of the hardware implementations of hash functions sha-1 and sha-512. In Proceedings of the 5th International Conference on Information Security, ISC '02, pages 75–89, London, UK, UK, 2002. Springer-Verlag. ISBN 3-540-44270-7. URL http://dl.acm.org/citation.cfm?id=648026.744521.
10. M. Halcrow. Review of affective computing. Linux Journal Magazine, pages 54–64, 2007.
11. Hanjae Jeong, Younsung Choi, Woongryel Jeon, Fei Yang, Yunho Lee, Seungjoo Kim, and Dongho Won. Vulnerability analysis of secure usb flash drives. In Proceedings of the 2007 IEEE International Workshop on Memory Technology, Design and Testing, pages 61–64, Washington, DC, USA, 2007. IEEE Computer Society. ISBN 978-14244-1656-1. doi: 10.1109/MTDT.2007.4547620. URL http://dl.acm.org/citation.cfm?id=1548882.1549152.

12. Y. Jeong. Vulnerability analysis of secure usb flash drives. EEE International Workshop on Memory Technology, De- sign and Testing, pages 61–64, 2007.
13. Andrew Teoh Beng Jin, David Ngo Chek Ling, and Alwyn Goh. Biohashing: two factor authentication featuring fingerprint data and tokenised random number. Pattern Recognition, 37(11):2245-2255, 2004.
14. M. Lin. Secured storage device with two-stage symmetric- key algorithm, 2008.
15. Ermelindo Maurellio. Tcfs: Transparent cryptographic file system. Linux Journal, August 1997.
16. Alfred Meneses, Paul Van Oorschot, and Scott Vanstone, Handbook of Applied Cryptography. CRC Press, 1997.
17. Henning Pagnia and Felix C. GA˜ d'rtner. On the impossibility of fair exchange without a trusted third party. Technical report, Darmstadt University of Technology, 1999.
18. R. K. Pal. Design and implementation of secure file system, 2008.
19. Hweehwa Pang, Kian Lee Tan, and Xuan Zhou. Stegfs: A steganographic file system. In Proceedings of the 19th International Conference on Data Engineering, pages 657-668, 2003.
20. Ronald L. Rivest. Cryptology, 1990.
21. Robert Shimonski. Hacking techniques: Introduction to password cracking. http://www.ibm.com/developerworks/library/s-crack/, July 2002.
22. Andrew S. Tanenbaum. Modern Operating Systems. Prentice Hall, 2007.
23. Iwan Vosloo and Derrick G. Kourie. Server-centric web frameworks: An overview. ACM Comput. Surv., 40:4:1–4:33, May 2008. ISSN 0360-0300. doi: http://doi.acm.org/10.1145/1348246.1348247. URL http://doi.acm.org/10.1145/1348246.1348247
24. C.P. Wright, M. Martino, and E. Zadok. Proceedings of the Annual USENIX Technical Conference, pages 197-210, San Antonio, TX, June 2003. USENIX Association.