
Copyright 2010 IEEE
Abstract

The rapid growth in the number of online services leads to an increasing number of different digital identities each user needs to manage. As a result, many people feel overloaded with credentials, which in turn negatively impacts their ability to manage them securely. Passwords are perhaps the most common type of credential used today. To avoid the tedious task of remembering difficult passwords, users often behave less securely by using low entropy and weak passwords. Weak passwords and bad password habits represent security threats to online services. Some solutions have been developed to eliminate the need for users to create and manage passwords. A typical solution is based on giving the user a hardware token that generates one-time-passwords, i.e. passwords for single session or transaction usage. Unfortunately, most of these solutions do not satisfy scalability and/or usability requirements, or they are simply insecure. In this paper, we propose a scalable OTP solution using mobile phones and based on trusted computing technology that combines enhanced usability with strong security.

1 Introduction

Identity management is normally interpreted as the management of users’ identities and credentials for accessing resources. Identity management systems can thus be seen as consisting of multiple functions related to authentication and access control. For example, identities need to be registered and deregistered. Credentials need to be securely distributed to principals. Access authorizations need to be defined and implemented on the system. During operation, principals need to be authenticated, and the access control function must grant or deny access to applications and resources residing on a system or in a network, as a function of the previously defined authorizations. The authentication and access control components are often tightly integrated.

Authentication is the process of verifying the correctness of a claimed identity or origin. With online services there are two types of authentications; user authentication and data origin authentication. User authentication is the process of validating the digital identity of an entity, e.g. when logging in and starting a new session. It is a way of ensuring that users are who they claim to be when they access systems. Data origin authentication is to validate that the source of data is as claimed. It is the verification that data has not been tampered with in transit (data integrity) and that it originated from the expected sender (authenticity).

In some online systems, data origin authentication is important. Although the user has logged on from a specific client terminal and has been authenticated at the start of a session, this in itself does not guarantee that every data packet originating from the client terminal is the intentional result of user actions. For example, a Trojan application could initiate online bank transactions from the client terminal without the user’s consent or knowledge. The Clampi virus is one such example which enables attackers to remotely control victims’ computers to perform financial transactions\(^1\). Data origin authentication can theoretically eliminate this threat by authenticating the transaction request itself.

The increasing number of digital identities that each user holds will negatively impact the user’s ability to manage them securely. This situation can be described by saying that users suffer from identity overload and password fatigue\(^1\).

The traditional requirements which dictate that passwords shall be difficult to guess and be different for different services put a considerable mental burden on users. Various studies\(^2\) show that people use heuristic strategies to reduce the mental load. Unfortunately, these strategies also make passwords vulnerable to attack. A typical strategy consists of reusing a small number of passwords for all the services a user accesses. This means that the number of passwords is constant while the number of services increases. To protect the service with the highest risk, users often reserve a single password for that service. Users tend to reuse the same password, or variations of the same password for all low risk services. This practice reflects that users will bypass or ignore good security practice when faced with frustrating tasks\(^3\). This represents a serious threat to the security of user authentication, making systems vulnerable to all variants of password cracking attacks.

As a response to the growing threats to online services security, special password management methods have been developed. In addition, these methods enhance data origin authentication by allowing the authentication process even at the transaction level. User authentication alone is insufficient given the vulnerability of the standard client terminal and the relatively high risk of some online services such as bank transactions.

A typical method for data origin authentication is to use an OTP (One-Time-Password). An OTP is 6 to 8 alphanumeric password that is used only once for user authentication or for the user to authorize a transaction. This method effectively eliminates the need for users to create and manage passwords and consequently reduces password cracking attacks. Banks can implement this by issuing special hardware tokens that can generate OTPs. The next section describes OTP generators in some detail.

\(^1\)A Trojan is a malicious software application that is not controlled by the owner of the computer.

2 One Time Password (OTP) Tokens

There are two types of OTP generator tokens, hardware tokens and software tokens. An OTP hardware token is a dedicated password generator device with an LCD screen which displays a pseudo-random number consisting of 6 or more alphanumeric characters (Studies showed that capacity of short term human memory load is normally 7 +/- 2 items [6]). Software tokens are OTP's generating functions that are stored on a general-purpose device such as a desktop PC, laptop, PDA, or mobile phone.

Also, tokens can either be time based or event based. With the time-based tokens, OTPs change at a specific time interval such as every 30 or 60 seconds and requires some sort of synchronization between the user's token and the authentication server. On the other hand, event based tokens work on the principle that when one OTP has been used, the next OTP in sequence is generated. The pseudo-random number changes when clicking a button on the token. The token device is synchronized with a peer OTP generator on the service provider (referred to henceforth as 'SP') side and both tokens generate the same sequence of numbers. The main weakness with time-based tokens is that they may become unsynchronized. This requires the user to resynchronize the token with the server by entering, for example, a number of consecutive pass codes.

The simplest and most common tokens do not need any connection to a computer and the OTP must be copied manually from the token to the client terminal. In fact, this separation gives this type of tokens its strength.

Tokens come in other forms as well. They can be smart card or USB stick tokens that can be physically connected to the client computer. Other types of tokens connect to the client computer using wireless techniques, such as Bluetooth tokens. Connected tokens have the advantage of transmitting OTP automatically to the client computer. This feature can enhance usability by removing the need for the user to manually copy and submit the OTP.

Another type of physical tokens is the contactless token. This category does not require a physical connection making them more convenient than both connected and disconnected tokens. Contactless tokens use RFID (Radio-frequency identification) to transmit authentication information.

In general, software tokens are considered to be weaker than hardware tokens because of the unstable nature of software compared to the security solid state framework of hardware tokens, thereby making hardware tokens less vulnerable to attacks; for example, with time-based software tokens, valid future passwords can be generated by simply forwarding the clock on the platform where the software token is installed.

The main disadvantage of hardware tokens is their usability problem. With the increasing number of hardware tokens needed by one user to be authenticated to different SPs, the more inconvenient it becomes to manage all the different tokens. On the other hand, since software tokens are installed in the devices where authentication is needed, this will eliminate the need for carrying separate physical tokens for each SP no matter how many tokens are required. This will make software tokens more usable and will also make them cheaper than hardware tokens.

Complex mathematical algorithm such as hash functions are usually used to generate the series of OTPs. Each password is unpredictable, even when previous passwords are known. With software tokens, OTPs are typically generated using a shared secret key. In this manner, an administrator will normally create and then submit a shared secret key to each user. The shared secret key will then be seeded to the software token to generate the series of OTPs.

In our opinion, the shared secret key should be under the SP control and beyond the user control. Rather than other entities such as applications, the user or owner of the token should not even have access to the shared secret key. This is because users often are weakest link in the security chain which is reflected by the relatively high success rate of social engineering attacks to compromise security systems, whereby the attacker manipulates the victim user to divulge confidential information that can be used to defeat the system's security. Implementing this feature (i.e. SP controlling the shared secret key) will enhance the security of shared secret key tokens. Unless the shared secret key is known to the attacker, such scheme will make a stolen software token useless.

Mobile phone can be configured to work as security tokens. Mobile phone tokens can reduce costs and eliminate the need for separate token device for each SP. In the next section we will introduce Trusted Computing Technology then, in the following sections, we will describe how to adopt this new technology to develop a more secure shared secret key tokens using mobile phones.

3 Trusted Computing (TC)

Trusted Computing (TC) is a general term used to describe security systems that are based on secure hardware. The Trusted Platform Module (TPM) is standardized chip that can provide support for TC. The purpose of the TPM standard, developed and promoted by the Trusted Computing Group (TCG)\(^2\), is to support strong security functions in general as well as Digital Rights Management (DRM) in particular. With TC, certain aspects of a computer's behavior can be enforced by hardware combined with software. TC functions of the TPM are based on verifying that software hash values (so-called measurements) are equal to predefined values, which in turn can be used to provide assurance of the authenticity and integrity of software modules. The TPM is currently integrated into and shipped with many hardware platforms such as PCs and servers. This development is mainly driven by business models for DRM. While the TPM is rarely used in current systems, its relatively large penetration provides a basis for implementing identity management solutions with strong security. Fig. 1 shows an example of a TPM chip.

Specific type's behavior can be enforced using TPM functionality. The TPM offers facilities for secure generation of cryptographic keys, and for controlling their use. The usage of some cryptographic keys of the TPM is restricted. For example, the owner of a computer is denied from knowing or accessing these keys in general. The usage of the keys inside the TPM is controlled according to policies specified in the TPM standard, and enforced by the TPM chip. The TPM security requirements are defined and evaluated according to the so called Common Criteria. The TPM provides the kernel components of a subsystem used to assure integrity, confidentiality and authenticity within a Trusted Computing Platform. The TPM provides the following functionality [7, 8]:

- Random number generator.
- Asymmetric key generation.

\(^2\)TCG is an initiative led by IBM, Intel, Microsoft, AMD and others which is used to be known as Trusted Computing Platform Alliance (TCPA). See: http://www.trustedcomputinggroup.org
Encryption/decryption.

Generation/verification of digital signature.

Hash algorithms.

Identification and authentication mechanisms.

Secure storage.

Every TPM comes with a unique key called the Endorsement Key (EK) which is embedded in the TPM at the manufacturing time [7]. The EK is an RSA pair key consisting of a private key \( (E_{pr}) \) and a public key \( (E_{pu}) \) that are 2048-bit RSA key each. The private key of the EK key pair \( (E_{pr}) \) is stored within the TPM’s secure hardware boundaries, is only used inside the TPM and cannot be accessed outside the TPM. The uniqueness of the EK pair represents the individuality of every TPM and the trusted platform based on it and can be used to facilitate the encryption of sensitive information in such a way that only the TPM can decrypt it.

Upon activating the TPM a special 2048-bit RSA key is generated and is guaranteed to always be present in the TPM. This key is called the Storage Root Key (SRK) and is in fact the root for all keys generated by the TPM [7].

Keys generated by the TPM can be either migratable or non-migratable [7, 8]. Migratable keys can move from one platform to another to provide the capability of more than one system to use a key while non-migratable keys are used when the private portion of the key needs to be guaranteed to only exist in the TPM. Also, keys generated by the TPM can be classified according to their usage [7, 8]. Storage keys are used to store other keys or data. Binding keys can be used to store symmetric keys and bind them to the TPM so they can not be accessed outside the TPM. Identity keys or Attestation Identity Keys (AIK) are used to sign other TPM keys. Identity keys are also used to sign PCR values (representing the software state of a platform). The AIK is an RSA 2048-bit private/public key pair \( (AIK_{pr}, AIK_{pu}) \) used to identify the TPM to a local or remote entity. An AIK pair can only be used in strictly controlled ways, and only ever for generating and verifying signatures. The main intended use for an AIK is for attestation [7]. That is, the AIK private key \( (AIK_{pr}) \) can be used by a TPM to sign PCR values and the public key \( (AIK_{pu}) \) can hence be used for verifying signatures on PCR values. More generally, an \( AIK_{pu} \) can only be used for signing values generated by the TPM itself and the \( AIK_{pu} \) is then used to verify these digital signatures.

The TPM uses a number of certificates. The certificates are classified in a tree or hierarchical chain, which ensures that every certificate can be validated by its predecessor [7, 9]. The validation process continues up to the root certificate which is assumed to be known and trusted. This arrangement allows external entities to authenticate the origins of a given certificate and track its ancestor certificates in the hierarchy. The root or the starting point of this certificate tree is the TPM’s EK Certificate. The EK is loaded to the TPM with a certificate from the TPM manufacturer and/or the platform’s vendor (where it is embedded in) and can prove to a secondary party that a key generated in the TPM was generated in a genuine TPM [7].

The TPM manufacturer keeps an RSA key pair \( (MK_{pr}, MK_{pu}) \) for that purpose. The private key part \( (MK_{pr}) \) is used to issue the Endorsement Certificate of the EK public key \( (E_{pu}) \) of each TPM. Assuming that the manufacturer and its public key are known, the existence of this certificate proves that the TPM is genuine and manufactured by a trusted manufacturer according to the TCG specifications. A TPM with an Endorsement Certificates that does not chain to a known and trusted root should not be trusted.

The private key \( (MK_{pu}) \) of the TPM manufacturer CA key pair is stored in a highly secure environment at the manufacturer’s site. It is only used to sign EK public key \( (E_{pu}) \) after production while the public key \( (MK_{pu}) \) of the manufacturer CA key pair is made publicly available to be used for the verification of signed EK. As an example, the manufacturing certificate of Infineon (a market leader in TPM chips and solutions) is published in the Infineon web page for the purpose of verification of the Infineon TPM [10].

To add another level of trust and make the TPM internal certificates more recognizable and trusted, the TPM root certificate (i.e. TPM’s EK certificate) can be digitally signed by an external Trusted Third Party. For example, the Infineon TPM EK Root certificate is chained to the Trusted Platform Module Root Certificate Authority of VeriSign (the acknowledged market leader in highly secure certificate technology) [10]. This extended signature chain enhances security and raises the level of trust of the entire TPM certificate chain by relying on the manufacturer root verification process to the worldwide trusted infrastructure of VeriSign’s PKI, certificates and signatures in addition to the Infineon website. The VeriSign certificate will be safely stored in the Infineon TPM in addition to the already existed certificates [10]. This could allow the owner of a TPM with an Infineon EK certificate to perform path-validation for any given EK certificate from Infineon.

Many industry players are planning to include TC in the coming generation of devices such as mobile phones and PDAs where TPM’s functions could be employed. The TPM specification provides security services common to all platforms and allows them to be implemented for specific platforms. The TCG provided a security specification for a trusted module similar to the TPM but is intended for mobile devices. The Mobile Trusted Module (MTM) specification [11] builds on the TPM security specification and trust model. The MTM specification identifies multiple owners of a mobile phone and defines two types of MTMs [11, 12]: the Mobile Local-Owner Trusted Module (MLTM) and the Mobile Remote-Owner Trusted Module (MRTM). The MTM is used to refer to both MLTM and MRTM. The MTM and TPM are compatible except for minor differences. The MLTM is defined in terms of capabilities and commands that

A PCR is a Platform Configuration Register in the TPM that stores a cumulatively updated hash of the platform configuration values commonly known as measurements.

\[5\] See the TCG Mobile Phone Work Group at: http://www.trustedcomputinggroup.org/developers/mobile
are defined in the TCG TPM specifications [11]. The MRTM can provide similar services (but not necessarily all) as the MLTM with extra functions to enable remote owners (e.g. the phone vendor and the cellular network provider) to control services such as access to the IMEI\(^6\) and the cellular network [11, 12]. More information about TC can be found in [7, 13, 14].

Encryption is an important feature of trusted computing technology. Data can be securely encrypted using \textit{binding} or \textit{sealing} functionality provided by the TPM whereby data is encrypted in such a way that it may be decrypted only under TPM control. "Binding" encrypts data using a TPM generated non-migratable binding key while "Sealing" encrypts data similar to binding, but specifies a state in which the TPM should be in for the data to be unsealed (i.e. decrypted).

An RSA key pair can be generated by the MLTM to protect sensitive information. Information can be encrypted using the public key so only the MLTM installed in a mobile phone (where the private key is stored) can decrypt it. Using this method, the clear text data can be read only to a particular mobile phone application where the software hash value is equal to a predefined hash value that is stored in the MLTM. Attempts to read the data by any application other than the predefined one will be refused by the MLTM. This feature can be used to utilize mobile phones to work as security tokens. An application can be installed on the mobile phone to perform similar functions normally provided by a dedicated tokens. In the next sections we will describe an enhanced solution that uses trusted computing technology to configure the mobile phone as a multi OTP generator device.

4 OTP Token Using Trusted Computing

The basic idea of the OTP authentication scheme is that a user device (such as an OTP token) uses a mechanism to authenticate the user to the SP by sending the user’s credentials and then connect to the SP’s server. The SP grants access to its services by verifying the OTP credential. The user device can be configured to facilitate the generation and management of the OTP credentials. The user side and the SP side can be configured to use the same function to generate the exact series of OTPs. In this manner, as the two functions at both sides are fed with the same inputs, both sides will generate the same output. Upon receiving the expected OTP from the user, the SP can verify the authenticity of the user and may then grant him access to its services. This is done by comparing the OTP generated by the verification function on the SP side with the OTP received from the user. For practical reasons, the comparison is often done against a window of expected (and unused) OTP values.

The security of the OTP authentication scheme is based on the properties that 1) each OTP has high entropy, and 2) each OTP can only be used once. This means that OTPs can not be guessed in advance, and an intercepted OTP is not released to the user. Also, the MLTM cryptographic functionality can provide physical protection of the sensitive parameters needed to generate the OTPs. The parameters (e.g. shared secret key) can be securely stored outside the boundary of the MLTM in the external memory in such a way that it is only accessible under the control of the MLTM. In other words, the OTP related parameters can be bound to the MLTM where only the MLTM can unbind them. Giving this, when the MLTM needs to generate an OTP, it will load the required bound parameter data from memory, unbind it and then seeds it to the OTP generation function (i.e. hash function) within its secure boundary. The MLTM only outputs the data which will be converted to OTP and does not release any OTP related parameters. This proposed solution will strengthen the OTP scheme by taking the security level down to the hardware and combining it with software to make the process of generating OTPs highly secure.

In order to securely generate the same sequence of OTPs, the MLTM and the SP should generate and maintain a \textit{shared secret key} to be fed to the OTP algorithms at both sides. The OTP algorithm supported by the MLTM is actually an HMAC (e.g., HMAC-SHA-1) function [7, 8]. In our proposed solution, SHA-1 function uses the shared secret key and a counter (for transaction-based method) or time value (for time-based method) to generate the OTP. In the transaction-based method, the MLTM and the SP increment the counter every time they generate a new OTP while in the time-based method the clocks in both sides should be synchronized.

Finally, before the device can display the generated OTP to a user, it needs to be truncated and converted to human readable form.

4.1 Architecture Description

Assume that User-A has a business relationship with SP-i. User-A needs to use his/her MLTM equipped mobile phone as an OTP token. To implement this, a special middleware application (referred to henceforth as ‘OTP Generator’) can be installed on the user mobile phone. The OTP Generator consists of two parts: the OTP token configuration and OTP generation. The first part is needed to initialize and configure the OTP token by managing the secure exchange of the shared secret key between the user’s mobile phone and SP-i in addition to the secure storage of the shared secret key on the mobile phone. The OTP generation part handles the process of generating the OTPs. Both parts are explained in more details below.

The key management part is processed only once at the beginning when configuring the OTP token while the OTP generation is executed every time a new OTP is needed.

A user can configure his/her mobile phone as a multi OTP generator token by requesting the service from the SP. For example, the user can call the SP or login to his/her account with the SP to request the service (similar to the way of ordering physical OTP tokens currently used my some bank customers), the SP may then send a CD containing the OTP...
Generator software to the user through the normal mail or the user may download the application from the SP’s website. The user then installs the OTP Generator on his/her mobile phone.

4.1.1 OTP Token Configuration

The security of the proposed scheme rests on the trust established between SP- and the MLTM equipped mobile phone. The trust will be assured if the shared secret key - which will be seeded to the OTP function - is created, encrypted and submitted to the intended mobile by SP- in a way that guarantees that the key can not be tampered with, and will only be used by the mobile’s MLTM of the intended user.

To establish a new shared secret key $SK_{i,A}$, the steps (indicated by numbered circles) illustrated in Fig. 2 are required. The user first requests the shared secret key from SP- (step1), then uses the OTP Generator to instruct the MLTM to generate a non-migratable binding key pair ($K_{pu-A}$, $K_{pr-A}$) (e.g. using the command TPM_CreateWrapKey) and obtains the resulting public key (e.g. using the command TPM_GetPubKey) (step4). Steps 2 and 3 will be discussed later.

The public key $K_{pu-A}$ will be then sent to SP- to use it to encrypt $SK_{i,A}$ so only the MLTM, where $K_{pr-A}$ exists, can decrypt it. However, if $K_{pu-A}$ alone is sent to SP- there is no way for SP- to validate that it has indeed come from the intended genuine MLTM. The mobile phone, where the shared secret key is intended to be migrated to, must be first identified to SP-.

The identification process involves two parts:

I. validating that the MLTM which generated $K_{pu-A}$ is genuine.

II. verifying that the mobile phone, where the MLTM is installed, belongs to the intended user (i.e. user-A).

The first identification part is the validation that the MLTM which generated $K_{pu-A}$ is genuine. To achieve this, $K_{pu-A}$ is sent to SP- with the required certificate to validate it. This is where $K_{pu-A}$ is signed. The user could be required to obtain a hash of $K_{pu-A}$ and then sign it (e.g. using the command TPM_CertifyKey) with an AIK. The signed public key ($K_{pu-A}$) with its certificate can then be sent electronically to SP- (part of step 5) which can verify that the key has been generated by a genuine MLTM.

SP- performs the validation, using the same certificate path-validation process described in Sec.3, before sending the shared secret key to the user. SP- has access to the certificate associated with the MLTM’s identity key though a third party certification authority. Consequently, SP- can be assured that $K_{pu-A}$ belongs to a trusted MLTM and can then send the shared secret key to it.

The second part is the authentication of the user in which SP- needs to know the identity of User-A who owns the mobile phone where the specific MLTM is installed. That will be achieved by the following tasks:

a. When receiving a request for a shared secret key, the SP-’s server will generate a nonce $N_a$ (step 2) and SP- will send it to the user’s mobile phone as an SMS through the cellular network (an out-of-bound channel (step 3).

b. The OTP Generator on the user mobile phone will then generate a hash value of both the public key ($K_{pu-A}$) and the received $N_a$ (i.e. public key and $N_a$ being the input to the hash function) and then send the hash value together with $K_{pu-A}$ certificate to SP- through the Internet channel. (step 5)

c. Upon receiving the data, SP- will authenticate the user by generating a hash of $N_a$ and the received public key ($K_{pu-A}$) and verifying that the generated hash value and the received hash value are equal.

The cryptographic association between the common secret $N_a$ (i.e. known only to the SP- and User-A) and $K_{pu-A}$ enables SP- to know the identity of the customer who sends $K_{pu-A}$.

The Man-In-The-Middle attack will be avoided because of the assumption that it is difficult for the attacker to obtain $N_a$. The Cellular network is used to exchange the value $N_a$ and the hash function is irreversible which means that the attacker can not get $N_a$ out of the hash value. The attack is avoided because the attacker can not regenerate another hash value of $N_a$ and his public key. This allows the user to prove to SP- his knowledge of the common secret (i.e. $N_a$) without revealing it.

Sending the hash value and the certificate (including public key) to SP- could be done, for example, by the user login normally to his/her account with SP- and then uploads the data.

At SP- side, $K_{pu-A}$ will be subsequently used to associate the shared secret key to the user mobile phone. Practically, this process is implemented by binding the shared secret key to the MLTM of the user mobile phone by the non-migratable binding public key $K_{pu-A}$. When $SK_{i,A}$ is
bound by the public key \((K_{pu-A})\) to the MLTM of the user mobile phone, only the private key \((K_{pr-A})\) which is protected by the MLTM of the mobile phone, can unbind it. The encrypted shared secret key \(E_{K_{pr-A}}(SK_{i-A})\) together with an initial counter \((Count_{i-A})\) value will be then sent to the user’s mobile phone (step 7). Again, to avoid the Man-In-The-Middle attack, a hash value of \(E_{K_{pr-A}}(SK_{i-A})\), \(Count_{i-A}\) and \(N_a\) will be sent to the user.

At the user side, \(E_{K_{pr-A}}(SK_{i-A})\) and \(Count_{i-A}\) can be stored in the mobile phone external memory. When OTP generation is needed, the OTP Generator requests the MLTM to unbind the stored value \(E_{K_{pr-A}}(SK_{i-A})\) to get the shared secret key. The private key \(K_{pr-A}\) corresponding to the public key \(K_{pu-A}\) that was used to encrypt the shared secret key must first be loaded into the mobile phone’s MLTM (e.g. using the command TPM_LoadKey) and then decrypt the value \(E_{K_{pr-A}}(SK_{i-A})\) (e.g. using the command TPM_Unbind). The decryption will be executed entirely on board the MLTM and within its secure boundaries. The OTP Generator is now ready for operation to generate OTPs.

The steps of Fig.2 are summarized in sequential order below:

1. User-A, known by SP-i requests a shared secret key \(SK_{i-A}\).
2. A nonce \(N_a\) is generated by the SP-i’s server.
3. SP-i sends \(N_a\) as an SMS to User-A.
4. The OTP Generator uses the MLTM to generate a non-migratable binding RSA key pair \((K_{pu-A}, K_{pr-A})\).
5. The public key \(K_{pu-A}\) is certified by the MLTM under an AIK. The OTP Generator sends the public key certificate (including \(K_{pu-A}\)) to the SP-i’s server. Also, the OTP Generator generates a hash of \(K_{pu-A}\) and \(N_a\) and sends the hash value to the SP-i’s server.
6. The SP-i’s server:
   a. generates a hash value of \(N_a\) and the received \(K_{pu-A}\),
   b. verifies that the received hash and the generated hash are equal in order to authenticate the user,
   c. validates that the MLTM is genuine using the retrieved certificate and
   d. generates the shared secret key \(SK_{i-A}\).
7. The server sends \(SK_{i-A}\) encrypted with \(K_{pu-A}\), \(Count_{i-A}\) and a hash of both values and \(N_a\) to the user mobile phone.
8. The OTP Generator stores \(Count_{i-A}\) and the value \(E_{K_{pr-A}}(SK_{i-A})\) in the mobile phone securely and uses the MLTM to decrypt it when needed.

This procedure enhances the system security by creating a root of trust where the SP (e.g. a bank) can ensure that only the mobile phone of the intended user can access the sensitive shared secret key.

The procedure described above can be repeated for an arbitrary number of SPs. This turns the mobile phone into a general purpose OTP generator token. The user will no longer need to carry separate hardware tokens for different SP. The OTP tokens can be virtualized and loaded into the same hardware device.

### 4.1.2 OTP Generation

As discussed earlier, the OTP function we propose to use to generate the OTPs is the SHA-1 function implemented by the MLTM. Also, we introduced two main classes of OTP generation methods. The first is the transaction-based where a new password is generated every time the user sends an OTP to SP-i. This method is based on a counter that requires synchronization between User-A and SP-i. The second class is the time-based where a new OTP is generated at specific time or time intervals. This method also requires synchronization of timing in the user side and SP-i side. The OTP generation is based on multiple inputs. We suggest to use the transaction-based method which uses a counter and the shared secret key, as illustrated in Fig.3. This will provide strong security against attempts to imitate the OTP function by attackers.

![Figure 3: Input parameters to the OTP generation function](image)

For this to work, the key needed to unbind the shared secret key (i.e. \(K_{pr-A}\)) must be first loaded to the MLTM. The MLTM will then use the loaded key to unbind the shared secret key to obtain \(SK_{i-A}\) . \(SK_{i-A}\) and other parameters will be fed to the SHA-1 function (e.g. using the commands TPM_SHA1Start, TPM_SHA1Update and TPM_SHA1Complete). The MLTM will output a 160-bit digest. All these operations will be executes within the secure boundaries of the MLTM. Finally, the OTP Generator application will convert the resulting digest to a human readable value (a.k.a. the OTP).

If the OTPs run out of order, the OTP Generator can request a new \(Count_{i-A}\) value from SP-i to resynchronize the OTP generation.

Access to generating an OTP is secured by a PIN as usual which could be the PIN used to access the mobile phone. This will provide two-factor authentication, i.e. the claimant user must control the device and know the PIN at the same time.

When the user wants to replace the mobile phone, the OTP Generator application must be installed in the new telephone and then the procedure of Fig.2 must be repeated for each SP.

### 5 Security Analysis

The proposed scheme combines the advantages of both the physical and the software based OTP tokens. More specifically, the scheme’s security is rooted in hardware, and it allows OTP generation for multiple SPs while still only requiring one single hardware device.

The scheme is aimed at enhancing the system security by creating a root of trust where the SP (e.g. a bank) can ensure that only the intended user’s hardware device, e.g. mobile phone with MLTM, can access the sensitive shared secret key. To gain SP trust, the scheme needs to satisfy two conditions:

1. The shared secret key must be distributed securely.
2. Every SP needs assurance that the shared secret key will not be compromised and will only be used by the genuine MLTM embedded in the intended user’s mobile phone.

To achieve that, every SP should be able to validate the MLTM’s source. The SP must be able to verify that the MLTM is manufactured by a known and trusted manufacturer according to the TCG specifications, and to verify that the validated MLTM indeed belongs to the intended user.

Standard key management and secret exchanging protocols like Diffie-Hellman and Needham-Schroeder are by themselves unable to accomplish the above requirements. The scheme needs not only to allow the SP and the user to securely create and exchange the shared secret key, but the scheme has to link the shared secret key physically to the user’s mobile phone and establish an association between the user identity and his/her mobile phone’s MLTM.

Binding the shared secret key to the private key which is protected by the MLTM of the user’s mobile phone, and performing the OTP calculation within the secure boundaries of the MLTM in addition to securely distributing the shared secret key between the involved entities makes it difficult for attackers to compromise the shared secret key. If the encrypted value of the shared secret key is intercepted and read by an attacker (e.g. Man-In-The-Middle), it will be of no value to the attacker unless he can obtain the private key to decrypt it. The scheme also reduces the attack opportunities to a negotiable risk level by distributing the shared secret key securely using two separate channels. A brief security analysis is outlined below.

The cost of compromising the shared secret key involves the cost of compromising the private key or the cost of simultaneously controlling the two separate channels (i.e. Internet and cellular network) to the same user. Once inside the mobile phone, the shared secret key is protected by the MLTM, and it is assumed that the MLTM is secure. It is also assumed that it is difficult for an attacker to take over the control of the user’s personal mobile phone and obtain the PIN to activate it.

Controlling the two separate channels simultaneously would be costly since it is assumed that it is difficult for attackers to control the cellular network or the SMS messages sent from the SP to the users while in transit through the mobile network. Even if interception and cryptanalysis of SMS messages sent over the air were possible, it requires that the attacker is physically present in the same base station coverage area, and this excludes attacks from anywhere in the world. This added constraint of using the cellular network which is separate from the Internet, where the public key is to be sent, will increase the cost of any attempt to attack the distribution of the shared secret key.

As an example, spoofing the user by the Man-in-the-Middle attack will not only require the interception of the Internet connection and replacing the public key of the user by the public key of the attacker, but it would also require the attacker to intercept the SMS sent from the SP to the user to get the value \( N_u \) and then generate a hash value of attacker’s public key and \( N_a \).

As mentioned, the mobile phone should be protected by a PIN and possibly also biometrics to prevent unauthorized entities from accessing its content in cases where the mobile phone is lost or stolen. In case a user fears that a stolen phone could be compromised, the user should prevent unauthorized entities from using it to generate valid OTPs by remotely locking it. Remote locking is a service provided by some mobile phone manufacturers and/or carriers to prevent unauthorized access of lost mobile phones. The user must also contact the SP to deactivate the OTP token. Deactivating the token will invalidate the OTPs generated by the lost mobile phone and guarantee that it cannot be used for authentication with that SP even if the locking service is not available.

6 Limitations and Future Work

We assume that \( N_a \) is a common secret between User-A and SP-1 at the time of OTP configuration because of the assumptions that:

- The SP knows the mobile phone number of User-A.
- User-A controls his mobile at the time of requesting the shared secret key and receiving \( N_a \).
- There is no other mobile phone with the same phone number.
- It is difficult for an attacker to control both the Internet and the mobile network at the time of OTP configuration.

However, the user can not be certain that \( N_a \) is coming from the SP. An attacker can intercept the request for shared secret key from the user (Step 1 in Fig.2) and send a false \( N_u \) masquerading as the SP. It is possible for anyone to use the SMS services provided on some Internet sited to send a message to any mobile phone number and spoof any originating phone number. The attack could establish an OTP scheme between the user and the attacker, but this would not allow the attacker to masquerade as the user and gain access to the SP’s services. The attack would normally be discovered by the user when he is unable to generate valid OTPs and consequently becomes unable to execute transaction requests.

According to the TCG standardization, the TPM (as well as the MLM) only provides support for hashing data using the SHA-1 algorithm. Although SHA-1 still can provide secure hashes, it is considered marginal. Among the features under consideration for the next generation of TPM, TCG has indicated that SHA-1 is being phased out [15], which is in practice means that it may be replaced by one of the SHA-2 family hash algorithms, or by the future SHA-3. A stronger Multi OTP solution can be deployed using one of the SHA-2 family functions which can have hash blocks sizes of 224, 256, 384 or 512 bits if they are supported by the future TPM.

The separation of the mobile phone from the user client terminal gives the OTP authentication scheme added security strength. However, with the trend of malware attacks migrating from the Internet to mobile networks it would also be possible for attackers to control the mobile phone and gain access to the OTP Generator. In fact, a perfectly secure system will never exist and there will always be weaknesses. For example, attackers can get access to the mobile phone if it is connected to the Internet or if the phone’s Bluetooth is enabled i.e. making it available for a connection. The relatively new attack known as snarfing, for example, allows intruders to gain access to Bluetooth enabled phones by exploiting a security flaw in the wireless protocol [16].

Also, this separation limits the scheme’s usability, which requires the user to manually copy the OTP from the mobile phone to the client terminal. To enhance the scheme’s usability, the process of copying the OTP can be automated by...
securely connecting the mobile phone to the client terminal without compromising the security strength. However, care must be taken with this type of implementation, making sure that the OTP Generator application and the shared secret key on the mobile phone are protected from unauthorized access.

To date there are no mobile phones with MTM, so this is not a solution that can be deployed quickly, however the increasing level of risk related to identity theft when using Internet services requires us to be well prepared.

7 Conclusion

The traditional requirements which dictate that passwords shall be difficult to guess and be different for different services put a considerable mental burden on users. Various studies [2] show that people use heuristic strategies to reduce the mental load. Unfortunately, these strategies also make passwords vulnerable to attack. This represents a serious threat to the security of user authentication, making systems vulnerable to all variants of password cracking attacks.

As a response to the growing threats to online services security, special password management methods that use an OTP (One-Time-Password) have been developed. This is implemented by issuing special tokens that can generate OTPs. OTP tokens can be hardware tokens or software tokens.

A fundamental problem with introducing software modules for identity management such as software tokens on the client platform is that its security relies on the inherent security of the platform. By storing shared secrets on the platform, the risk caused by a compromised platform is amplified dramatically.

Although hardware tokens are considered to be stronger than software tokens, however they have a usability problem. With the increasing number of hardware tokens needed by one user to be authenticated to different SPs, the more inconvenient it becomes to manage all the different tokens. On the other hand software tokens can eliminate the need for carrying separate physical tokens for each SP no matter how many tokens are required.

The solution proposed in this paper involves the integration of hardware and software in order to assist users with identity management. In particular, we have described a solution where it is possible to install multiple virtual one-time-password generators in a mobile phone that is equipped with the hardware Mobile Trusted Module.

A situation where users are forced to have multiple physical OTP Generation devices results in usability problems and does not scale. In general, since poor usability leads to poor security, our approach which allows multiple OTP generators on a single device will improve security by improving the usability and solving the scalability problem from the user perspective.

References