The Master Key: A Private Authentication Approach for Pervasive Computing Environments

Feng Zhu[1] Matt W. Mutka[1] Lionel M. Ni[2]
[1]Dept. of Computer Science and Engr
Michigan State University
East Lansing, Michigan, USA
{zhufeng, mutka}@cse.msu.edu
[2]Dept. of Computer Science
Hong Kong University of Science and Technology
Kowloon, Hong Kong, China
ni@cs.ust.hk

Abstract

We propose a novel entity authentication approach for pervasive computing environments. A person uses a single device, the Master Key, which aggregates all his digital forms of access tokens for entity authentication. The Master Key discovers and selects proper tokens for its owner. With an emphasis on usability, the Master Key secures authentication, protects privacy information from outsiders and insiders, and supports various claimant-verifier relations. We analyze privacy and security properties of our approach and protocols, and we investigate the overhead. Performance measurements show that our protocols are efficient.

1. Introduction

We prove our identities everyday by showing the possession of access tokens. Using a key to open a lock may be the most common form, which has about 4000 years of history since ancient Egypt [1]. As one may access many locks, traditional master keys were designed to enable accessing multiple locks with a single key. Nevertheless, master keys are not widely used. Instead, people carry multiple access tokens for entity authentications, e.g., keys, magnetic stripe cards, smart cards, RFID tags, and other tokens. We propose *the Master Key*, which is a novel approach for digital access tokens to have the advantages of master keys and multiple access tokens.

Traditional master keys are convenient. One does not need to carry many keys and memorize relationships between keys and locks. However, traditional master keys have fatal problems that are not suitable for everyone's daily usage. For example, the delegation of a master key equals delegating access to all locks that one has privilege to access. Revocation of the privilege of accessing one lock from a master key is costly because the lock and the keys of other owners need to be replaced. In addition, if an intruder acquires a master key, then the intruder may access many locks. Moreover, locks that support master keys are vulnerable to the malicious insider who has a normal key [2].

The use of multiple access tokens does not have the fatal delegation and revocation problems as traditional master keys have because one token usually matches one lock. Similarly, if a key-lock pair is compromised, it does not put other locks at risk. Issues concerning delegation and revocation are further addressed by replacing keys with modern access tokens, e.g., a hotel room key in the form of a magnetic stripe card or a smart card. With the encoding of privileges within a digital form, the delegation and revocation of the privileges are simple. Moreover, modern access tokens improve usability in a wide variety of applications, e.g., unlocking a car using a remote control; accessing an enterprise facility using a smart card badge; entering a parking facility using a RFID gate card; opening a hotel room using a magnetic stripe card; or locking and unlocking a computer by wearing a token [3]. Additional token designs are emerging as well as their applications. Nevertheless, the management of access tokens and memorizing the token-lock relationships become inconvenient and difficult as the number of tokens increases.

In pervasive computing environments, entity authentications might be ubiquitously necessary. An intuitive question is how to achieve both the advantages of traditional master keys and multiple access tokens while avoiding their disadvantages. The *Master Key* that we propose aggregates the digital forms of all access tokens that its owner has. The tokens on the Master Key and their respective locks maintain their original relationships (one token matches one lock). Therefore the advantages of using multiple access tokens are retained. The goal of this paper is to design an approach that properly selects access tokens for entity authentications and therefore achieves the convenience of traditional master keys. From now on, we refer to keys as the

* This research was supported in part by the National Science Foundation under grant no. 0334035 and Hong Kong Research Grants Council grants HKUST6183/05E and AoE/E-01/99.

digital access tokens and locks as the target resources.

The Master Key is invoked when the owner pushes a lock or unlock button. At that time, the Master Key discovers the right key for a lock and authenticates with the lock. During the procedure, the Master Key needs to address the following challenging issues.

- **Privacy**. Privacy information includes the key owners' credentials and their presence information. Most security protocols only protect sensitive information from outsiders but not from insiders. However, the concern of privacy may be very important among insiders, namely the concerns among key owners or between a key owner and a lock. For situations when the exposure of presence information is unnecessary, the Master Key protects privacy information from both insiders and outsiders.
- **Security**. As an entity authentication approach, the Master Key should prevent illegitimate keys to open locks. Mutual authentication is employed in the Master Key to properly identify key owners and locks and verify their legitimacy.
- **Scalability and efficiency**. Potentially, one may interact with many locks, and a lock may interact with many key owners. During lock discovery using the Master Key, one may specify hundreds of potential locks and a lock may specify hundreds of potential key owners within one network packet. Our protocols require only three messages to discover locks, identify a key owner, and finish mutual authentication.

We analyze privacy exposure in terms of probabilities and address the relative overhead incurred to preserve privacy. Different exposure strategies and their respective advantages and disadvantages are discussed. Three representative protocols are presented for key-lock relationships, which have different privacy concerns. The protocols are implemented, and measurements on PDAs show that our approach is efficient. A protocol run between the Master Key and a lock is less than a half second.

The rest of the paper is structured as follows. In Section 2, we discuss related work. Section 3 presents our Master Key design and protocols. We demonstrate our claims through analysis and experimentation in Section 4. We discuss some related issues in Section 5. Last, in Section 6, we outline our future work.

2. Related Work

In this section, we first survey several technologies that are widely used and embedded in small devices for entity authentication. Then, we discuss other research work that inspire our work.

Magnetic stripe technology is now widely used as access tokens [4]. For example, hotel guestroom locks and employee badges use the technology, while the most common usage is for bankcards. Most magnetic stripe cards contain three tracks on which data or even PIN numbers are encoded and stored. Since magnetic stripe cards are relatively easy to forge, the annual loss in the UK alone due to counterfeit cards is about £130 million in 2004 [5].

Due to the smart cards' computational and storage capabilities and better security features, they are considered the replacement for many magnetic stripe card applications [6]. Smart cards are also used as prepaid transit cards, ID cards, health cards, or are even embedded within passports [7]. Smart cards may be classified as contact or contactless cards. The former type of cards needs physical contact with readers, while the latter conveniently works over radio frequency links. A smart card may support multiple applications or authentication purposes. Our Master Key approach complements multifunctional smart cards and provides an approach to select the correct credentials privately during authentication.

RFID tags (i.e., passive and active tags) are used as authentication tokens, such as in [8, 9], respectively. However, RFID tags can be tracked by reading tag IDs from their messages and thus they are not appropriate for entity authentication when privacy is a concern. Several solutions are proposed to improve privacy for RFID tags in some situations [10, 11], but the privacy problem is not entirely solved. Moreover, passive RFID tags do not have processing capabilities to perform cryptographic operations. The MIT Card System uses magnetic stripe and passive RFID technologies. The usage of passive RFID tags on the cards introduces several new vulnerabilities [12], which makes the cards even less secure than their old magnetic stripe ID cards.

Remote Keyless Entry systems are commonly installed on new automobiles and garage-doors. On a typical Remote Keyless Entry system [13], when its owner pushes a button, the remote control sends a message (8 or 16 bytes) to the receiver. The message contains a "rolling code" for authentication [14]. The "rolling code" is a pseudo-random number that is generated both at the controller and the receiver by using the same seed. The seed is computationally difficult to find from the pseudo-random numbers.

iButtons are used as keys, e-cash, and asset management devices. For instance, in New York City over 200,000 iButton owners are using iButtons to access over 10,000 buildings [15]. Interestingly, memory and processor chips in the iButtons are encapsulated within stainless steel cans. The cans

serve as the iButton's network interface when iButtons touch readers. Some iButtons support password protected memory data, challenge-response authentication, or even public key encryptions. Moreover, large memories on iButtons allow an owner to store many keys and certificates for various applications. Our approach may complement iButtons when multiple credentials are stored on them and may prudently select and expose sensitive credential information.

If one aggregates all his credentials on one device, he may justify having a more powerful device, which has better processing capability, memory size, power, and user interface, and has the authentication and tamper-resistant features that are used on iButtons or smartcards. On a powerful device, an owner may audit his authentication processes. In addition, having a richer user interface also enables much more information sent to owners. For instance, when unlocking a car, an owner may be notified that the tire pressure is low.

Recent implementation and experimentation show that public key operations, such as elliptic curve cryptography, take only several seconds to run on the 8-bit Berkeley/Crossbow Mica2 mote platform [16]. Energy consumption due to cryptographic operations and wireless communication on those embedded devices are low [17]. These tiny devices are expected to cost as little as a dollar in a few years [18]. Imagine that within pervasive computing environments, such devices may be embedded in locks and other commodities with simplified key delegation and revocation. Together with the Master Keys, locks may further differentiate key owners and provide services accordingly.

Location-based or proximity-based authentication also inspires our work. Improving usability is a major goal of these approaches. For example, the Active Badge, a pioneer location sensing system, may be used as a remote control to start a user's X window display on the nearest computer, and then a user pushes buttons on the badge to control the display [19]. Based on relative location context, content of the applications and displays may be adapted. When a user leaves, applications and the display are automatically closed. With location or even orientation information, the Master Key design could be further improved and unnecessary computation and communication overhead will be reduced.

In another example, Zero-Interaction Authentication (ZIA), an owner wears a token to secure his laptop without actively locking or unlocking his laptop [3]. When the owner leaves, all file systems on the laptop are protected via encryption. When detecting that the owner returns (i.e., the token is nearby), the laptop fetches decryption keys from the token and restores itself to the state before the owner left. Since there is no owner's involvement in an authentication, ZIA achieves optimal usability. XyLoc uses a similar idea for automatically locking and unlocking a PC [20]. However, these approaches may not be suitable for applications that are beyond trustworthy locks and key owners. If an owner wears tokens for all his authentication tasks, a malicious insider (a lock or other key owner) might query a token to track the owner without the owner's knowledge.

Biometric recognition, such as fingerprint, iris, hand geometry, and voice recognition, is used in various authentication applications including keys to open locks [21]. As a proof of "who you are", biometric recognition has several advantages over other authentication approaches such as token-based ("what you have") or password-based ("what you know"). While other approaches may not be appropriate, biometric authentication can be used for negative authentication (prove the person who he denies being) [21]. Nevertheless, biometrics may not be suitable to serve as a master key because not all service providers are trustworthy and will keep the biometric information secure and private. Biometric is difficult or even impossible to revoke and delegate. If it is broken in one application, other applications may be jeopardized. However, biometric recognition seems to be a good approach to secure the Master Key, for instance a fingerprint may be used to activate the Master Key.

In PrudentExposure [22], a similar data structure, namely the Bloom Filter [23], is used during service discovery processes to find legitimate service providers and users. By exchanging precise code words, service providers and users discover each other's existence in one round. The authentication approach discussed in PrudentExposure does not protect privacy information from malicious insiders.

The Bloom filter has wide applications in database and networking [24], and recently it has some applications in security [22, 25]. The Bloom filter is a compression method to express memberships. It has the efficiency advantages of storage space and computational time, while paying the price of false positive cases in membership tests. In many applications, the tradeoff is worthwhile [24]. The Master Key not only utilizes its time and space advantages but also utilizes false positive cases to preserve privacy. Unlike other applications in which each element in a Bloom filter has the same length, elements in the Master Key may have different lengths. Short elements are used to preserve privacy because of their high false positive rates, while long

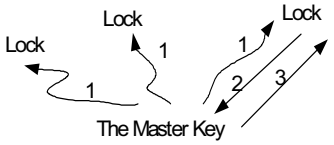
elements are used for their time and space advantages.

Beaufour and Bonnet proposed to use Personal Servers as digital keys [26]. In their design, a lock actively looks for devices to make a connection. Once a connection is established, a lock identifies itself. If the device is a Personal Server with digital keys, it identifies and proves itself to the lock. Otherwise, the lock marks the device as an invalid key device. Sure enough, a lock that initiates an authentication process simplifies the design such that a Personal Server can easily select the key for the lock. Nevertheless, it is irrational for some devices to take the initiative because most processing and communication efforts are wasted. Such waste on a car's Remote Keyless Entry system for example, drains the same battery that the car uses to start the engine. If a car is parked for several weeks, the battery in the car may become completely drained.

Abadi and Fournet proposed two private authentication protocols for ad hoc networks [27]. The protocols enable two principals to authenticate and establish a secure communication channel without explicitly specifying their identities. Assuming that the public keys of the principals are known, authentication messages are encrypted using the other principals' public keys, and thus only the one that holds the private key understands the messages. Unlike protocols in which the target principal is assumed to be known in the vicinity, the target lock in our case needs to be discovered. Moreover, their protocols do not offer protection from malicious insiders. A principal either exposes or refuses to expose, and therefore they are not suitable to discover locks. The Master Key uses different exposure strategies to protect privacy information from insiders, unless it is necessary to expose.

3. The Master Key Design

The Master Key initiates an authentication process by sending a broadcast message, as shown in Figure 1. It queries a set of locks for which it has keys. If a lock finds itself in the set, it replies back. Then the Master Key supplies a key to operate the lock.



1. Broadcast message: any lock in this set nearby?
2. Unicast message: this lock is nearby.
3. Unicast message: this is the key.

Figure 1. An authentication process between the Master Key and a lock.

3.1. Protect Privacy Information from Outsiders – Private Authentication

Unlike standard authentication protocols that send identifications in clear text, our authentication process exchanges only code words. A code word is generated at both the Master Key side and a lock's side. It is calculated from a shared secret between the Master Key and lock. One sends a code word for authentication, while the other verifies. Since only a party who knows the shared secret understands a code word, identifications are protected from being exposed to outsiders. Thus, private authentication is achieved.

Code words in our design have two formats: the hash format and the Bloom filter format. The hash format is used when the key and lock has a one-to-one relation, for example after the Master Key has discovered the target lock. The Bloom filter format is used when keys and locks have one-to-many relationships. For instance, the Master Key queries a set of potential locks, or a lock needs to identify a key owner among many key owners.

The top half of Figure 2 illustrates the generation of a code word in the hash format. A time variant parameter (TVP) and the shared secret are the two inputs to *h(*)*, which is the hash-based message authentication codes proposed in [28]. The Hash function, MD5 or SHA-1, is used in the place of *h()*. The hash result is the hash format code word. The preimage resistance and collision resistance properties of MD5 and SHA-1 ensure that it is computationally difficult to find the shared secret from the hash result [29]. The TVP and hash result are sent to the other party for verification.

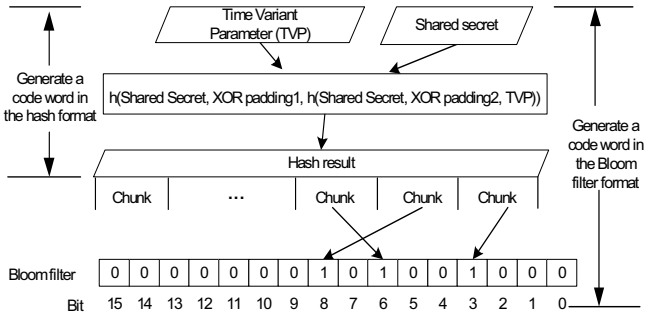


Figure 2. The generation of a code word.

For a code word in the Bloom filter format, a hash result as we discussed above is first generated. Then, the hash result is further separated into chunks as shown in Figure 2. The size of the chunks depends on the length of the Bloom filter, which is an array of 2^X bits. (To be fit in a network packet, a Bloom filter is equal or less than $2^{13} = 8192$ bits.) If

the Bloom filter is 2^{13} bits, then the chunk size is 13 bits. Since the hash result is 128 bits for MD5 and 160 bits for SHA-1, a hash result is separated into at least 10 chunks. All bits in a Bloom filter are initially set to zero. The value of a chunk serves as an index to a Bloom filter and the corresponding bit is set to one. A code word is represented by a combination of several bits that are indexed by the chunks. For example, in Figure 2, bits 8, 6, and 3 represent a code word.

For all potential locks that the Master Key wants to discover, it repeats the above process using the same TVP and sets the code word bits in the same Bloom filter. Then, the Bloom filter and the TVP are broadcasted to query the locks in the vicinity.

If a lock and each of its key owners share a unique secret, the lock may generate code words for all its key owners in a Bloom filter via the same process. Then, the lock sends the Bloom filter and requests the key owner who sends the discovery message to identify himself.

The probability of finding a hash result from a Bloom filter is $1/(\frac{m!}{(m-k)!})$, where k is a code word length and m is the number of bits set in a Bloom filter. The denominator is the number of permutations of k bits from m bits, that is, select k bits from m bits and then make arrangements of the k chunks to guess a hash result. Only part of the hash result might be found if the code word is generated by part of the chunks. Even if the hash result is found, it is still computationally difficult to find the shared secret as in the situation of the hash format code words.

The Bloom filter format of code words is scalable. Hundreds of code words may be expressed in a Bloom filter. For instance, if the Master Key uses a 2^{13}-bit Bloom filter, of which 50% are set, and on average each code word is 5 bits, then at least $2^{13}\times 50\% \div 5 = 819$ code words may be set in a discovery message. The result (819 code words) is a lower bound, which is calculated from the extreme case that no two code words set the same bit.

Code word verification in the Bloom filter format is efficient and independent of the number of code words in a Bloom filter. A party calculates the hash results, as we discussed above, and then verifies whether the bits indexed by the chunks in a Bloom filter are set to one. If any bit is not one, then the code word does not match. A property of Bloom filters ensures that if two hash results match, the Bloom filter format of the code words match [23]. It is possible that a party may find false positive matches, and we will discuss the probability of false positive cases in Section 3.3 and the relative overhead in Section 4.1.

3.2. Secure the Authentication Process

Figure 3 shows the generic protocol for the authentication process. In the first and second messages, both the Master Key and a lock post challenges, TVPs, respectively; while in the second and third messages the other parties respond with code words based on the challenges. Thus, mutual authentication is attained.

The code words are one-time code words. Based on TVPs, code words between the Master Key and a lock differ between protocol runs. In addition, a replay code word can be easily detected by checking the freshness of a TVP. (The clocks on the Master Key and locks are required to be loosely synchronized.)

The total number of bits set in a Bloom filter may be maliciously used as a signature to track or identify a Master Key owner. To counter the attack, code words and some random bits are mixed together to reach a fixed ratio of the number of bits set and total number of bits in a Bloom filter. Thus, all Bloom filters look the same. Moreover, code word lengths in the Bloom filter format are obscured. A lock only indicates the last bit of the matched code word in its reply message, as shown in Figure 3. Thus, every code word looks the same to an attacker.

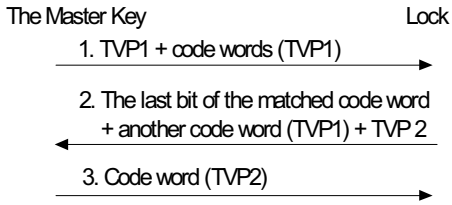


Figure 3. The generic interaction protocol shows the exchange of code words between the Master Key and a lock.

3.3. Protect Privacy Information from Insiders

Since the authentication process includes discovery of the locks and keys, unnecessary code words may be transmitted. Although the Master Key and locks exchange code words, insiders understand the code words. There might be no privacy concerns among insiders. For example, Bob and his wife Alice are not concerned that their Master Keys speak code words for their cars, because only their Master Keys and cars understand the code words. However, in some situations there are privacy concerns among insiders. For example, Bob does not want other gym key owners to use the knowledge of the key for the gym to identity or track him at places other than the gym.

To address the problem, Bob's Master Key may speak a partial code word for the gym door lock in a discovery message. A partial code word causes an insider to be uncertain whether Bob has the gym key. If Bob is not near the gym door, his Master Key will not receive a reply message from the gym door lock, and thus Bob preserves his privacy. A partial code word increases the number of false positive cases, which for example, the gym's door lock may have communication and computation overhead to interact with some illegitimate key owners. (Illegitimate key owners do not gain access because the code words in the last messages can only be generated by legitimate key owners.) When there is no privacy concern, a precise code word should be used to avoid such overhead. For example, Bob's Master Key always speaks a precise code word for his car.

The Bloom filter has the properties that meet our needs. Both precise code words with low false positive rates and partial code words with high false positive rates may be specified in the same Bloom filter. We first examine some mathematical properties of the Bloom filter for our case. Some formal mathematical analysis of the false positive rates and the calculation of the expected false positive rate may be found in [24, 30]. Unlike the analyses, which are based on having each element in a Bloom filter with the same length, the Master Key may use various lengths for different code words. Moreover, the Master Key uses a fixed ratio between the numbers of bits set and the Bloom filter lengths. The false positive rate in our case is a typical sampling with replacement problem. Therefore, the false positive rate at a lock's side is:

$$p(\textit{code word match} \mid \textit{not key owner}) = \left(\frac{m}{n}\right)^k \qquad (1)$$

where n is the Bloom filter length, m is the number of bits set in the Bloom filter, and k is the length of a code word. Thus, given an illegitimate key owner, the false positive rate depends on n, m, and k. (The analysis of the false positive rate at the Master Key's side is similar, and thus we omit it.)

Examining equation 1, we find the property that is useful in our case. When the length of a code word increases and the m/n ratio is fixed, the false positive rate decreases as shown in Figure 4. For example, if the m/n ratio is at 50%, a code word of 1 bit has a false positive rate of 0.5, a code word of 5 bits has a false positive rate about 0.03, and a code word of 10 bits has a false positive rate less than 0.001. The m/n ratio at around 50% provides a good span of false positive rates for the code words from 1 bit to 10 bits. By default, 50% of the m/n ratio is used. In summary, if a key-lock pair wants to achieve a low false positive rate, they may exchange more bits (say 10 bits) in a code word. If there are privacy concerns, they may exchange fewer bits (1, 2 or 3 bits) of a code word.

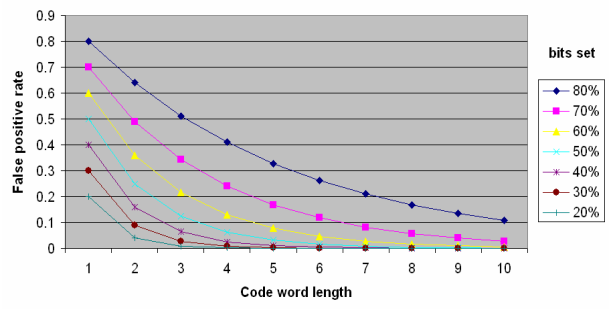


Figure 4. False positive rates decreases as the length of the code word increases.

Long code words reduce unnecessary communication and processing overhead. To further reduce unnecessary overhead, another hash algorithm or another secret may be used to generate a code word of 20 bits or longer. Very long code words are useful for applications that require extremely low false positive cases, such as for Remote Keyless Entry systems.

The use of partial code words changes the order from having the key owner first expose code words precisely to having locks first expose code words precisely. Precise exposure later has the advantage when privacy is a concern. Because when there is a mismatch in the code word, a party that exposes later may avoid the exposure.

3.4. The Master Key Protocols

In this section, we present three protocols to demonstrate that our approach is flexible for different types of key-lock relations. A *unique key* is one that may be owned by one or a few owners to open a lock, for example a car's Remote Keyless Entry system; a *group key* is one that a lock is able to authenticate as a key, but key owners are not differentiable, for example a gate card to entering a parking facility; and an *individual key* is one for which a specific key owner among a group can be identified, for example a badge for entering a factory.

The initialization processes for all relationships are the same. Shared secrets are delivered from locks to the Master Keys via secure channels. In addition, locks indicate the number of bits of the code words in the Bloom filter format.

All protocols have the same first message, which the Master Key uses to discover potential locks. Thus, we discuss the first message only in the protocol for unique keys. In the next two messages,

the Master Key and a lock authenticate each other using their respective authentication protocol.

3.4.1. The Unique Key

The Master Key initiates an authentication process by broadcasting a Bloom filter format of code words with a TVP. The TVP consists of a random number and a timestamp as shown in Figure 5(a). A lock replies only if it finds a match. The lock indicates the last bit of the code word, and uses another hash algorithm to generate a code word in the hash format. In addition, the lock posts its challenge, another TVP. By default, MD5 is used for the code words in the Bloom filter format, while SHA-1 is used for the code words in the hash format. If the Master Key finds the indicated last bit of the code word in the Bloom filter format and the hash format code word are correct and match, it replies back with another hash format code word. It is possible that the Master Key may have several Bloom filter format code words that have the same last bit. But comparison of the code words in the hash format will exclude the false positive cases.

If there are several key owners, they share the same secret with the lock. Usually, this type of key-lock relation is used for owners and locks without privacy concerns, and thus a 10-bit code word in the Bloom filter format is used. However, if key owners have privacy concerns, they may use a code word with fewer bits. The protocol remains the same, while the number of bits of a code word changes. Alternatively, the individual key type may be used to address key owners' privacy concerns.

3.4.2. The Group Key

The special requirement for the group key is that a lock should not be able to differentiate key owners from their keys. This means that all owners should have the same key. The same protocol as the unique key may be used, since the only difference between the group key and the unique may be the number of key owners. But a key owner should only speak a 1-bit or 2-bit code word. A short code word ensures that a lock cannot differentiate among key owners. Because when there are two different code words of length 1 or 2 bits, it is very likely that the lock will find a false positive match and a true match in the first message. The key owner is able to tell that he has a different key than other owners when the lock replies with an incorrect code word in the hash format.

If the overhead caused by the false positive cases is large, a lock and its owners may use more bits for the code word in the Bloom filter format. However, the code word is generated from some plain text, as shown in Figure 5(b). The plain text in the message may be some human readable text such as "the CS department's mail room" or "XYZ Company's parking lot". The use of plain text instead of a secret changes the order of who first expresses knowledge of a secret. Since the lock expresses its knowledge first, the Master Key knows that it shares the same secret as other key owners. If there is more than one secret, the lock may provide an incorrect code word in the second message. Moreover, the plain text is easy for key owners to verify that code words in the Bloom filter format are generated from some meaningful and reasonable text.

Notation:
- L is a lock. M is the Master Key.
- t_X is a timestamp that X attaches.
- R_X is a random number that X generates.
- A t_X and a R_X is a TVP.
- $(\ )_{KX}^{-1}$ is X's signature using its signing private key.
- $BF_P(y, S)$ is a code word in a Bloom filter that P generates from a shared secret, S, and a TVP, y.
- $Hash_P(y, S)$ is a code word in the hash format that P generates from a shared secret, S, and a TVP, y.
- MB_P is the last bit of a code word that a party, P, finds the match.

Msg No.	Sndr/Rcvr	Message
1	M→L:	$BF_M(R_M, t_M, S_{Unique})$, R_M, t_M
2	L→M:	R_M, t_M, MB_L, R_L, t_L, $Hash_L(R_M, t_M, S_{Unique})$
3	M→L:	R_L, t_L, $Hash_M(R_L, t_L, S_{Unique})$

(a). The protocol for the unique key.

Msg No.	Sndr/Rcvr	Message
1	M→L:	$BF_M(R_M, t_M, S_{PlainText})$, R_M, t_M
2	L→M:	R_M, t_M, MB_L, R_L, t_L, $Hash_L(R_M, t_M, S_{Group})$
3	M→L:	R_L, t_L, $Hash_M(R_L, t_L, S_{Group})$

(b). The protocol for the group key.

Msg No.	Sndr/Rcvr	Message
1	M→L:	$BF_M(R_M, t_M, S_{domain})$, R_M, t_M,
2	L→M:	R_M, t_M, MB_L, $Hash_L(R_M, t_M, S_{domain})$, $(Hash_L(R_M, t_M, S_{domain}))_{KL}^{-1}$, $BF_L(R_L, t_L, S_{individual})$, R_L, t_L
3	M→L:	R_L, t_L, MB_M, $Hash_M(R_L, t_L, S_{individual})$

(c). The protocol for the individual key.

Figure 5. The Master Key protocols.

3.4.3. The Individual Key

A key owner of the individual key can be identified among the group of key owners. A secret

$$p(key\ owner \mid match) = \frac{p(match \mid key\ owner) \times p(key\ owner)}{p(match \mid not\ key\ owner) \times p(not\ key\ owner) + p(match \mid key\ owner) \times p(key\ owner)} \quad (2)$$

that the lock shares with a key owner is different from the secrets that it shares with other key owners. Thus, each key-lock pair has an individual secret. However, the individual secret is not used to generate the Bloom filter format of a code word in the first message. Since the Master Key may specify many code words in the first message and a lock may have many key owners, many false positive cases may happen. To address such a situation, another domain secret is shared among all key owners and is used in the first message to identify the lock. Figure 5(c) shows the protocol for the individual key.

In the reply message, the lock proves its knowledge of the domain secret in the hash format. If the key owners are concerned that the reply message may come from another key owner who impersonates the lock, a digital signature may be used in place to counter the attack. As shown in Figure 5(c), the lock signs the hash format code word.

The lock sends another Bloom filter in the second message that encodes code words for every key owner. The code words are generated from the individual secrets. Furthermore, the lock may set some random bits in the Bloom filter to hide the number of key owners that it has. In the third message, the Master Key indicates its identity by specifying the matched code word in the Bloom filter and proves its knowledge of the individual secret.

Revocation methods are different for the three key types. Invalidating the unique secret revokes a unique key. To revoke an individual key from a key owner, a lock invalidates the individual secret, while notification of a new domain secret to other key owners may not be imminent. However, to revoke a group key from a key owner, all other key owners need to update their group keys. This is an open problem in our solution. If an owner updates his key when he finds that the key has expired, the lock system may be able to determine the owner's identity because he has just updated his key. We are designing an approach, which is similar to sending email to a group of recipients, so that a new group key is dispatched to all key owners at the same time.

4. System Analysis and Evaluation

In this section, the effectiveness of privacy protection against insiders and its relative overhead are discussed. Performance measurement of our protocols shows our approach is efficient.

4.1. Analysis of Privacy Protection against Insiders and its Overhead

An insider recognizes a code word in the Bloom filter format, but whether the code word is from a true key owner is a probability, $p(key\ owner \mid match)$ (because there are false positive cases). This probability may be calculated from equation 2, where $p(key\ owner)$ is the percentage of key owners among all people who send discovery messages at a place, and $p(match \mid not\ key\ owner)$ is the false positive rate of a code word in the Bloom filter format. $p(match \mid key\ owner)$ is one because there is no false negative case for code words in the Bloom filter format. The numerator and the denominator on the right side of equation 2 are $p(key\ owner, match)$ and $p(match)$, respectively.

Figure 6 illustrates the relation between the number of bits in a code word and $p(key\ owner \mid match)$ for various $p(key\ owner)$ values. The false positive rates, $p(match \mid not\ key\ owner)$, are based on setting 50% of the bits in the Bloom filters. Note $p(key\ owner \mid match)$ is for one lock at a place. Different locks may have different $p(key\ owner \mid match)$ values at the same place.

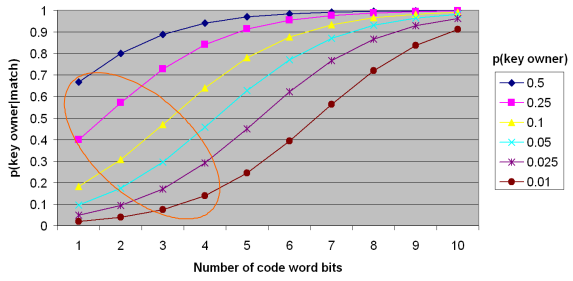


Figure 6. The relation between the number of bits in a Bloom filter format code word and $p(key\ owner \mid match)$.

The eclipse area in Figure 6 shows the number of bits that a key-lock pair uses to protect privacy among insiders. A lock may count successful and unsuccessful protocol runs (unsuccessful runs include false positive matched and unmatched code words in Bloom filters that the lock hears), and then calculate

$p(key\ owner)$ over a period of time at its place. Nevertheless, at other places that no successful run happens, $p(key\ owner)$ is unknown to insiders. An insider is uncertain when he finds a match in the Bloom filter format of a code word because the $p(key\ owner \mid match)$ falls within a wide range. Even if $p(key\ owner)$ is known, $p(key\ owner \mid match)$ is a probability that is not significantly true.

The false positive overhead at the lock's side is $1 - p(key\ owner \mid match)$. Thus, the overhead is high if few bits are used for a code word, while the overhead for a 10-bit code word is low. A lock may calculate $p(key\ owner)$ over a period of time. If the overhead is a concern, it may notify its key owners to adjust the length of a code word. When a lock is not in the vicinity, the overhead at the Master Key side includes the calculation of a code word in the Bloom filter format and possibly the verification of the hash format code word in a false positive case. We will show in the next section that the overhead in terms of processing time is small.

4.2. Performance Measurement

Handheld devices such as cell phones or PDAs are good candidates for Master Keys, since people regularly carry them. Locks may have diverse processing and communication capabilities. Some may have limited processing power, while others may support hundreds of key owners. We chose to measure our implementation on PDAs: a Compaq iPAQ with an ARM SA1110 206 MHz processor, 64MB RAM, and a D-Link DCF-650W wireless card runs as the Master Key and a Dell AXIM X5 with Intel PX250 400 MHz processor, 64MB RAM, and a Dell TrueMobile 1180 wireless card runs as a lock. The PDAs run Microsoft PocketPC 3.0, and the wireless cards are set to 2Mbps in the 802.11 ad hoc mode. One protocol run takes less than a half second in a extreme case, in which a person specifies 820 code words and a lock has 500 key owners. Thus, our design is efficient in most cases.

5. Discussion

The Master Key protocols that we discussed so far are susceptible to the mafia fraud attack, as are many entity authentication protocols. Mafia fraud attacks may not have countermeasures by cryptography alone. Presently, there are several representative solutions to counter the attacks without physically isolating claimants (devices). First, location information may be integrated into an authentication protocol as proposed in [31]. Second, measuring the transmission time between a claimant and a verifier, and then one can determine whether the distance between the two is within the expectation [32, 33]. Recent improvements based on location and time information may be found in [34]. Third, based on the assumption that an eavesdropper is not able to monitor all communication channels, a large number of channels are simultaneously used to obscure some real communication channels [35]. These approaches can be adapted and fit into our protocols. For instance, if the Master Key and a lock know their location information, then the code words can be also based on the location information. Thus, an attack will be easily detected from the location information. Moreover, the Master Key and a lock may measure their upper distance bound. Instead of sending a code word in the hash format in one message, the Master Key and a lock may send a bit at a time over multiple rounds and determine whether their distance is reasonable.

Securing the Master Key is critical. Losing it may be as serious as losing a key chain and/or a wallet. Finger recognition and tamper-resistant features may reduce the problem. Moreover, the Master Key may need multiple interfaces (physical contact, wired communications, and wireless communications) and may communicate over different radio frequencies to interact with various locks. These problems are important, but they are out of the scope of this paper.

6. Conclusion and Future Work

In this paper, we propose the Master Key approach for entity authentication in pervasive computing environments. Our approach improves usability such that a person carries one device for various authentication purposes while it maintains the favorable properties of carrying multiple access tokens. The Master Key exchanges code words with locks securely and privately. Sensitive information, including identities and presence information, is protected from malicious outsiders via encryption and from malicious insiders via a probabilistic approach.

The current design of the Master Key does not support multiple groups of key owners. Thus, a lock will find up to one code word that matches in the Bloom filter format. However, if multiple groups are supported in a lock, the lock may find multiple code words that match several groups due to the false positive cases. The lock could determine the false positive cases by establishing multiple sessions and exchanging messages with the Master Key, but in the group key type this violates the privacy feature and thus the Master Key owner will not be sure that his shared secret for the lock is the same as other owners.

We are designing an approach to support multiple groups, while the group key type still maintains its desirable privacy feature. In the meantime, we are designing an approach to make the revocation of a group key easier.

The design of exchanging a few bits to protect privacy as we discussed in Section 3.3 and 4.1 is conservative. A more aggressive approach could be let a lock and its key owner exchange a code word with more bits when $p(key\ owner)$ is small. However, the challenge is that a selfish lock that only concerns about its overhead will sacrifice its key owners privacy. We are designing an approach to properly balance the overhead and privacy properties.

References

[1] Schlage, "History of Locks," available at http://professional.schlage.com/about_us_historyoflocks.asp.

[2] M. Blaze, "Rights Amplification in Master-Keyed Mechanical Locks," IEEE SECURITY & PRIVACY, vol. 1, pp. 24-32, 2003.

[3] M. Corner and B. Noble, "Zero-Interaction Authentication," presented at Conference on Mobile Computing and Networking (MobiCom), Atlanta, Georgia, USA, 2002.

[4] S. G. Halliday, "Introduction to Magnetic Stripe & Other Card Technologies," presented at SCAN-TECH ASIA 97, Singapore, 1997.

[5] Association for Payment Clearing Services, "UK CARD FRAUD LOSSES REACH £504.8M," 2005, available at http://www.epaynews.com/downloads/APACS%20UK%20Card%20Fraud%20PR%208%20March%202005.pdf.

[6] J. Ferrari, R. Mackinnon, S. Poh, and L. Yatawara, Smart Cards: A Case Study: IBM Corporation, 1998.

[7] Smart Card Alliance, "Smart Card Implementation Profiles," available at http://www.smartcardalliance.org/industry_info/profiles.cfm.

[8] J. Pearson, "Securing the Pharmaceutical Supply Chain with RFID and Public-key infrastructure (PKI) Technologies," Texas Instruments, June 2005, available at http://www.ti.com/rfid/docs/manuals/whtPapers/wp-Securing_Pharma_Supply_Chain_w_RFID_and_PKI_final.pdf.

[9] AXCESS Inc web site, "Personnel Access Control," 2005, available at http://www.axcessinc.com/.

[10] A. Juels, R. Rivest, and M. Szydlo, "Selective Blocking of RFID Tags for Consumer Privacy," presented at 10th ACM Conference on Computer and Communications Security, Washington D.C., 2003.

[11] S. A. Weis, S. Sarma, R. Rivest, and D. Engels, "Security and Privacy Aspects of Low-Cost Radio Frequency Identification Systems," presented at 1st International Conference on Security in Pervasive Computing, Boppard, Germany, 2003.

[12] P. Agrawal, N. Bhargava, C. Chandrasekhar, A. Dahya, and J. D. Zamfirescu, "The MIT ID Card System: Analysis and Recommendations," 2004, available at http://swiss.csail.mit.edu/6.805/student-papers/fall04-papers/mit_id/.

[13] Dallas Semiconductor, "Requirements of Remote Keyless Entry (RKE) Systems," Nov. 11 2004, available at http://www.maxim-ic.com/appnotes.cfm/appnote_number/3395.

[14] G. Goebel, "Codes, Ciphers, & Code breaking," Jun. 1 2004, available at http://www.vectorsite.net/ttcode.html.

[15] iButton home page, 2005, available at http://www.maxim-ic.com/products/ibutton/.

[16] N. Gura, A. Patel, A. Wander, H. Eberle, and S. C. Shantz, "Comparing Elliptic Curve Cryptography and RSA on 8-bit CPUs," presented at Workshop on Cryptographic Hardware and Embedded Systems, Cambridge, MA, 2004.

[17] A. Wander, N. Gura, H. Eberle, V. Gupta, and S. C. Shantz, "Energy Analysis of Public-Key Cryptography for Wireless Sensor Networks," presented at 3rd IEEE Annual Conference on Pervasive Computing and Communications, Kauai island, Hawaii, 2005.

[18] M. Prince, "Tiny Wireless Sensors Are Poised for Market," in The Wall Street Journal, 2003.

[19] A. Harter and A. Hopper, "A Distributed Location System for the Active Office," IEEE Network, vol. 8, 1994.

[20] Ensure Technologies, available at http://www.ensuretech.com/products/technology/technology.html#HowXyLocWorks.

[21] S. Prabhakar, S. Pankanti, and A. Jain, "Biometric Recognition: Security and Privacy Concerns," IEEE SECURITY & PRIVACY, vol. MARCH/APRIL, 2003.

[22] F. Zhu, M. Mutka, and L. Ni, "PrudentExposure: A Private and User-centric Service Discovery Protocol," presented at 2nd IEEE Annual Conference on Pervasive Computing and Communications, Orlando, Florida, 2004.

[23] B. Bloom, "Space/Time Trade-offs in Hash Coding with Allowable Errors," Communications of ACM, pp. 422-426, 1970.

[24] A. Broder and M. Mitzenmacher, "Network Applications of Bloom Filters: A Survey," Internet Mathematics, vol. 1, 2005.

[25] E.-J. Goh, "Secure Indexes," 2004, available at http://crypto.stanford.edu/~eujin/papers/secureindex/index.html.

[26] A. Beaufour and P. Bonnet, "Personal Servers as Digital Keys," presented at 2nd IEEE Annual Conference on Pervasive Computing and Communications, Orlando, Florida, 2004.

[27] M. Abadi and C. Fournet, "Private Authentication," Theoretical Computer Science, vol. September, pp. 427-476, 2004.

[28] M. Bellare, R. Canettiy, and H. Krawczykz, "Keying Hash Functions for Message Authentication," presented at Advances in Cryptology–CRYPTO '96 (LNCS 1109), 1996.

[29] A. Menezes, P. v. Oorschot, and S. A. Vanstone, Handbook of Applied Cryptography: CRC Press, 1996.

[30] L. Fan, P. Cao, J. Almeida, and A. Broder, "Summary Cache: A Scalable Wide-Area Web Cache Sharing Protocol," IEEE/ACM TRANSACTIONS ON NETWORKING, vol. 8, pp. 281-293, 2000.

[31] Y. Desmedt, "Major security problems with the 'unforgeable' (feige)-fiat-shamir proofs of identity and how to overcome them," presented at SecuriCom '88, Paris, France, 1988.

[32] T. Beth and Y. Desmedt, "Identification tokens -- or: Solving the chess grandmaster problem," presented at Advances in Cryptology Crypto '90, 1991.

[33] S. Brands and D. Chaum, "Distance-bounding protocols," presented at Advances in Cryptology – EUROCRYPT '93, Lofthus, Norway, 1993.

[34] Y.-C. Hu, A. Perrig, and D. Johnson, "Packet Leashes: A Defense against Wormhole Attacks in Wireless Networks," presented at 22nd Annual Joint Conference of IEEE Computer and Communication Societies (INFOCOM 2003), San Francisco, CA, 2003.

[35] A. Alkassar, C. Stüble, and A.-R. Sadeghi, "Secure object identification: or: solving the Chess Grandmaster Problem," presented at the 2003 workshop on New security paradigms, Ascona, Switzerland, 2003.