Private Entity Authentication for Pervasive Computing Environments

Feng Zhu[1], Matt W. Mutka[2], and Lionel M. Ni[3]
(Corresponding author: Feng Zhu)

Department of Computer Science, University of Alabama in Huntsville, Huntsville, Alabama 35899, USA[1]
Department of Computer Science and Engr, Michigan State University, East Lansing, Michigan 48824, USA[2]
Department of Computer Science, Hong Kong University of Science and Technology, Kowloon, Hong Kong, China[3]
(Email: fzhu@cs.uah.edu, mutka@cse.msu.edu, ni@cs.ust.hk)

(Received June 21, 2010; revised and accepted Aug. 29, 2010)

Abstract

Entity authentication becomes ubiquitously necessary in pervasive computing environments. We provide taxonomy of entity authentication between keys and locks. Based on the discussion, we propose a novel authentication approach for pervasive computing environments. A person uses a single device, the Master Key, which aggregates all his digital keys for entity authentication. The Master Key initiates authentication and selects proper keys by exchanging code words with locks. With an emphasis on usability, the Master Key secures authentication, keeps authentication private, and supports various key-lock interactions. We analyze privacy and security properties, verify the protocols, and measured performance.

Keywords: Authentication, pervasive computing, privacy, probabilistic, security

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. 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, for example, 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 both master keys and multiple access tokens. In this paper, we use the term, key, for an access token or a digital access token and the term, lock, for a digital lock or a computing resource.

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 use. The delegation of a master key equals delegating access to all locks that one has privilege to access. Revocation of a master key to access a lock is costly because the lock and the keys of other owners need to be replaced. If an intruder acquires a master key, then the intruder may open many locks. In addition, locks that support master keys are vulnerable to the malicious insider who has a normal key [9].

The use of multiple keys does not have the fatal delegation and revocation problems as traditional master keys have because one key usually matches one lock. If a key-lock pair is compromised, it does not put other locks at risk.

Issues of delegation and revocation are better addressed by replacing keys with digital keys, for instance, 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 done on the computers at the front desks. Moreover, digital keys improve usability in a wide variety of applications, for example, 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 digital key [17]. Additional digital key designs are emerging as well as their applications. Nevertheless, the management of keys and key-lock relationships become tedious as the number of keys increases.

In pervasive computing environments, entity authentications might be ubiquitously necessary. To the best of our knowledge, our Master Key is the first design for digital master keys. It achieves the good usability as traditional master keys because it automatically and properly selects digital keys for entity authentications. The Master Key also achieves security and privacy as multiple digital keys. While it aggregates one's digital keys on a single device, it maintains the original key-lock relationships (one key for one lock). The key contributions of this paper are:

Good Usability.
The Master Key frees users from memorizing key-lock relationships. Users operate the Master Key by pushing lock and unlock buttons. The Master Key and locks discover whether an authentication process is needed by speaking code words - encrypted messages that are without keys' or locks' identifications and only the proper keys and locks understand.

Key Initiated Authentication.
The Master Key does not require users to select keys, instead it automatically select the proper keys for authentications. The design becomes challenging when the Master Key initiates an authentication process. Via three messages, the Master Key discovers and selects the proper key among hundreds of keys and finishes a mutual authentication.

Private Authentication.
The threat in pervasive computing environments is that unauthorized parties can eavesdrop and learn the identities sent in clear text and associate with the time, location, and other context information. When authentication is ubiquitous necessary, unauthorized parties may infer much sensitive information from authentication sessions. We mitigate the threat by speaking code words only. And therefore, only those who understand code words understand who is authenticating with whom.

Flexible Encoding Scheme.
Our novel encoding scheme enables the Master Key and locks to express authentication information from precisely to vaguely in terms of probability. In addition, each key-lock pair may choose the order of exposure. Therefore, the Master Key and locks can protect sensitive information by control the amount of authentication information they expose.

We formally define the Master Key design problem. The mathematical properties of our encoding scheme are proved. Different exposure strategies and their respective advantages and disadvantages are discussed. We present three representative protocols for different key-lock relationships. The protocols are formally verified using BAN logic and our extension to ensure the protocols' freshness, binding, and privacy properties. Our implementation on PDAs shows that the approach is efficient. An authentication process between the Master Key and a lock is less than half a second.

The rest of the paper is structured as follows. In Section 2, we discuss related work. Then in Section 3, we illustrate taxonomy of the key-lock interaction design and the requirements for the Master Key design. Section 4 presents the design and analysis of the automatic key selection process using code words. In Section 5, we discuss the support for various exposure orders and amount of exposure between keys and locks. In Section 6, we demonstrate three protocols for key-lock interactions with different requirements. We show the implementation and performance measurements of our protocols in Section 7. We discuss some related issues in Section 8. Last, in Section 9, we outline our future work and conclude our contribution.

2 Related Work

Several technologies are used and embedded in small devices for entity authentication. Magnetic stripe cards are widely adopted as hotel guestroom keys, bankcards, and employee badges [23]. Most magnetic stripe cards contain three tracks on which data or even PIN numbers are encoded and stored. Nevertheless, counterfeit cards cause huge financial loss every year [3].

Smart cards are utilized as prepaid transit cards, ID cards, health cards, or even embedded within passports [20, 33]. Many smart cards contain not only memory but also microprocessors. The chips on the smart cards have tamper-resistant features. There are contact or contactless smart cards. The former type needs physical contact with readers, while the latter works over wireless links.

RFID tags are mainly used to provide unique identification numbers as the name suggests. Some RFID tags (i.e., passive, semi-passive, and active tags) are used as authentication tokens, such as in [4, 29], respectively. Passive RFID tags usually do not have processing capabilities to perform cryptographic operations. Semi-passive RFID tags may be capable of running hash functions, whereas active tags can execute public key encryption operations. The main privacy concerns for RFID tags are clandestine tracking and inventorying. Counter attack approaches have been designed, but there are still serious challenges [25].

Remote Keyless Entry systems are commonly installed on new automobiles and garage-doors. On a typical Remote Keyless Entry system [34], 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 [21]. The "rolling code" is a pseudo-random number that is generated at both 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 [24]. Interestingly, memory and processor chips in 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. In Table 1,

Table 1: Comparison of entity authentication technologies

	Built-in processing power	Memory size	Power source	Authentication	Tamper resistant	Multiple-key support	User interface
Magnetic stripe	None	Hundreds of bytes	N/A	One-way	No	No	No
Smart card	Yes (most)	Up to few MB	From readers	Mutual	Yes	Yes	No
RFID (passive)	Limited capability	4KB	No	One-way	No	No	No
RFID (semi-passive/active)	Yes	4KB	Battery	Unknown	No	No	No
Remote Keyless Entry	Limited capability	4KB	Battery	One-way or mutual	No	No	Buttons
iButton	Yes (some iButtons)	Up to 134KB	From readers or battery	Mutual	Yes	Yes	No

we compare the devices that we have discussed.

Using a single digital key as a master key might be a possible design choice. It would achieve good usability as the traditional master keys. Nevertheless, a single digital key sacrifices users' privacy and reduces security. A recent unsuccessful attempt, Microsoft Passport, uses a single digital identity for websites across administrative domains. Using a single identity unnecessarily exposes the same identity information to all websites that a user accesses [13]. Additionally, users' preferences and behaviors can be connected and learned.

There have been attempts to improve the usability of entity authentication using digital tokens. Beaufour and Bonnet proposed to use Personal Servers as digital keys [6]. 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 locks to send discovery messages continuously because most processing and communication efforts are wasted. Moreover, without mutual authentication and privacy protection, malicious attackers may send a fake message to query one's identity and track him.

In Zero-Interaction Authentication (ZIA), an owner wears a token to secure files on his laptop without actively typing the password [15]. 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 to automatically lock and unlock a PC [17]. 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.

To address the usability issues of multiple identities for different websites of a user, cross domain single sign on systems have been developed and are evolving rapidly. Security Assertion Markup Language (SAML), an Oasis and ITU standard, offers a framework for different service providers and identity providers to exchange authentication and authorization information [31]. Behind the scenes, service providers and identity providers link a user's accounts of different websites and share related attributes with each other. Therefore, a user only needs to login once for accessing protected websites across administrative domains. Similarly, OpenID eliminates the multiple user names and passwords for different websites and applications [19]. The OpenID authentication protocol enables websites to redirect authentication requests to OpenID providers to verify users. Unlike SAML and OpenID, the Master Key allows users to keep different identities for different applications and automatically selects correct identities for authentications.

Windows CardSpace stores all digital access tokens of a user as information cards [14]. CardSpace provides a consistent user experience for Internet accesses with the capability to support any type of digital tokens including user names, x.509 certificates, and Kerberos tickets. When a user accesses a protected website, the website sends a policy to the web browser and then to CardSpace. CardSpace finds information cards that match the policy and let the user select a digital access token. The Master Key addresses a more difficult challenge. A user does not need to explicitly specify a lock and memorize the key-lock relationships.

Biometric recognition, such as fingerprint, iris, hand geometry, and voice recognition, is used in various authentication applications including being used as keys to open locks [30] and secure data [26]. Nevertheless, biometrics may not be suitable to serve as a master key because biometrics is difficult or even impossible to revoke and delegate. However, biometric recognition can be used to secure the Master Key, for instance using fingerprint recognition to activate the Master Key.

In our PrudentExposure paper [35], a similar data structure, namely the Bloom Filter [10], is used during service discovery processes to find legitimate service providers and users. The PrudentExposure uses a different encoding scheme, which is limited to the full exposure.

In our progressive service discovery protocol [36], service providers and users exchange partial information in multiple rounds to discover available services. In case of mismatches during the service discovery processes, both service providers and users stop communicating. Thus, service discovery and authentication information is protected because only partial information is exposed and a malicious party acquires the sensitive information with uncertainty. We may adopt the approach in the Master Key design when both a key owner and a lock have privacy concerns.

The Bloom filter has wide applications in database and networking [11], and it has applications in security [22, 35]. 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 most applications, the tradeoff is worthwhile [11]. The Master Key utilizes not only its time and space efficiency but also false positive cases to preserve privacy.

3 Key-Lock Interaction

Since we have not seen any systematic analysis and comparison of digital key-lock interaction designs, we present taxonomy of existing designs. There are many choices when designing the Master Key. We discuss the potential choices for the Master Key to achieve good usability, private authentication, and security. Then, we formally define the Master Key design problem.

3.1 Taxonomy of the Key-lock Interaction

Figure 1 illustrates the taxonomy based on three design characteristics, namely communication among keys and locks, a key's initial status, and the exposure methods.

During communication, some keys may physically contact locks in order to operate locks. For example, a guest inserts a card into a hotel room's lock. Other keys may operate locks via wireless links, and thus are contactless. Contactless interaction may work over a span from short distance to long distance. Contactless smart cards interact with locks within a few centimeters, whereas a key in the Remote Keyless Entry system works over dozens of meters. Contact keys are the least convenient and slowest to use because they need to physically contact locks. Long distance contactless keys do not have the limitation that locks need to be within the reach of a key owner.

During authentication, contact keys implicitly provide strong context that the key is present. Although one might assume a similar context that contactless keys are within the vicinity, such assumption may be invalid. For instance, a RFID tag usually is accessible within a short distance, but a malicious attacker may build a powerful reader to read tags [1].

A key's initial status may be passive, reactive, or active. If a key is passive, it requires the owner to insert the key into the lock, such as a magnetic stripe card or an iButton. Passive keys communicate with locks via physical contact. While being reactive, a key is ready to respond upon receiving an authentication message from a lock. For example, when using a passive/semi-passive RFID tag or a contactless smart card as a key, the key is reactive. After receiving a message and power from a lock (reader), the key proves itself to the lock. Active means that a key initiates the authentication process and transmits a message. For instance, in a Remote Keyless Entry system for a car, the key (remote control) becomes active when an owner pushes a button, while the system in the car wakes up frequently (every 20ms) to receive commands from the key [34]. In general, reactive keys should be avoided for entity authentication, since they are more susceptible to attacks. Unlike the passive and active keys, a reactive key does not require its owner to initiate an authentication process. Thus, a hacker may build powerful readers to track people and maliciously acquire authentication messages without a key owner's knowledge. For example, recent attack on Mifare Classic card may potentially affect billions of users who use the RFID cards as reactive digital keys [28]. Passive keys are the safest but the least convenient because key owners need physically touch the locks using the keys. Like passive keys, active keys require key owners to initiate authentication sessions. Locks do not need to send authentication messages periodically and thus are very suitable for the locks using batteries.

If only a lock verifies a key, it is one-way authentication. If they both verify the other parties, it is mutual authentication. *Mutual* authentication provides better security and increases users' trust.

Keys and locks may expose different degrees of their authenticity to the other parties. In one-way authentication, a key exposes its information to the lock in *full*, while a lock exposes *no* information to the key. In mutual authentication, both a lock and a key expose their information in full. Potentially, a key and a lock may expose *partial* information in a round and verify the other party's authenticity in multiple rounds. Although such exposure approach has not been used in digital keys and locks, we proved elsewhere that it can effectively protect both parties' privacy [36]. If a party detects that authentication is unnecessary, it can immediately stop the process without fully exposing its authentication information.

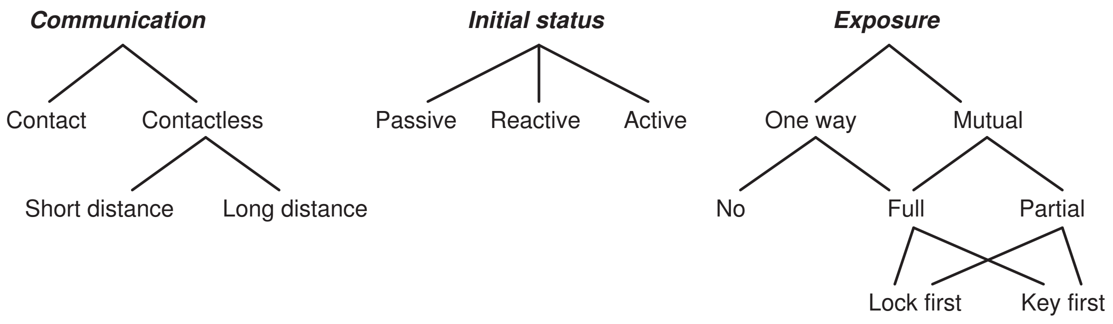

Figure 1: AES in counter mode acts as a keystream generator for an additive stream cipher

Keys and locks may predetermine their orders of the exposure. Either a key sends the authentication message first (key first), or a lock sends its authentication message first (lock first). Based on the authentication message received, the party that exposed later can determine proper reactions. For example, a Remote Keyless Entry system ignores authentication messages if they are not for the car. When the authentication is one-way, the order of exposure becomes a degenerate case, in which only the key exposes. The design choices for digital keys are shown in Table 2.

3.2 The Master Key and Lock Interaction

To meet different key-lock interaction requirements and constraints, the Master Key supports all key-lock interactions except reactive and one-way authentication. Keys with reactive initial status are susceptible to attacks as discussed in Section 3.1. For contactless communication, the Master Key requires users to actively start an authentication process. For contact communication, the Master Key will be passive. We assume that the Master Key has appropriate physical and wireless channels to support passive and active keys. We require the Master Key and locks to mutually authenticate each other (no one-way authentication). Thus, the Master Key can ensure that a key is exposed to the intended lock. Combining all characteristics together, Figure 2 shows the interaction approaches that the Master Key supports.

Besides supporting various key-lock interactions, the Master Key needs to address a critical requirement - users do not need to remember key-lock relations. From users' perspective, the ultimate goal is to ensure access to protected physical places and digital resources instead of managing keys and locks. Ideally, the Master Key automatically selects the correct key for an intended lock, and thus it frees users from identifying keys and memorizing the relationship between keys and locks. We use a private and secure discovery approach to find the proper key for the intended lock.

We define the Master Key and lock interaction as follows:

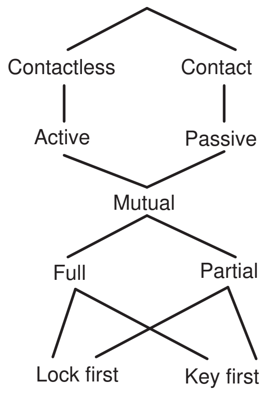

Figure 2: Supported interactions of the Master Key

The Master Key contains a set of keys, $\{K_i\}_i \in N$. A lock, L, may be operated by one or more digital keys $\{K^L_M\}_m \in N$.

The Master Key and lock achieve strong authentication via mutual authentication such that a lock believes $K_i \in \{L^L_M\}$ and the Master Key believes the lock is the intended lock, $L_j = L_{intended}$.

Based on their requirements between a lock and the Master Key, they may choose partial or full exposure in a message. Therefore, the probability $p(K_i \in \{K^L_M\}) = a, a \in (0, 1)$ and the probability $p(L_j = L_{intended} = b, b \in (0, 1))$.

The Master Key and a lock may predetermine who exposes authentication information first.

When a user operates a lock, $L_{intended}$, the Master Key automatically selects the key, $K^{L_{intended}}$, $K^{L_{intended}} \in \{K_i\}_i \in N$.

Table 2: Comparison of key-lock interaction design

	Initial status	Communication	Exposure
Magnetic stripe	Passive	Contact	Full, one-way
Smart card (contact)	Passive	Contact	Most mutual, some one-way, full
Smart card (contactless)	Reactive	Contactless, 10 cm or less	Full, mutual
RFID (passive/semi-passive)	Reactive	Contactless, a few feet	Full, one-way
RFID (active)	Active	Contactless, hundreds of feet	Full, one-way
Remote Keyless Entry	Active	Contactless, dozens of feet	Full, most one-way, some mutual
iButton	Passive	Contact	Full, mutual
ZIA	Reactive	Contactless	Full, mutual
Personal Servers	Reactive	Contactless	Full, one-way

4 Discover the Proper Key to Operate a Lock

There are two situations. In the simpler situation, a lock actively notifies the Master Key its information. The Master Key can search all the key-lock pairs based on the lock's information. Thus, the Master Key can select the proper key, $K^{L_{intended}}, K^{L_{intended}} \in \{K_i\}_i \in N$, for authentication. This method can be applied to locks that require digital keys to physically contact them (passive initial status). The event that a lock and a key physically touch each other triggers the lock to notify the Master Key its information.

For a key and a lock that communicate over wireless channels (contactless), a lock might continuously broadcast authentication messages. Nevertheless, locks waste almost all their power and communication because they are not being operated most of the time. For locks running on batteries such as Remote Keyless Entry on cars, the method may not be acceptable. When some locks do not continuously broadcast authentication messages, the Master Key has to actively discover locks. In the rest of the paper, we focus on this more challenging situation.

The Master Key initiates a contactless entity authentication (active initial status). Locks are in the listening mode. (A battery powered lock may switch between sleep mode and listening mode periodically to save power.) When a user pushes a button to lock and unlock, the Master Key broadcasts a discovery message to query whether there is any lock in the vicinity that it can operate. The message is in the code word form. If a lock understands and recognizes the Master Key, it replies back. Then, the Master Key searches all the key-lock pairs and verifies the lock. If a proper key is found, the Master Key submits the proper key, $K^{L_{intended}}, K^{L_{intended}} \in \{K_i\}_i \in N$, for authentication.

4.1 Private Discovery by Speaking Code Words

In pervasive computing environments, a user may authenticate with different parties in unprotected network environments. If identities are exchanged in clear text, malicious attackers may infer sensitive information and associate it with the identities. To mitigate the threat, the Master Key and locks exchange only code words. A code word is generated at both the Master Key side and a lock's side. It is calculated from a secret shared between the Master Key and the lock. The shared secrets are unique. (We will discuss more details about shared secrets in Section 6.)

The Master Key operates many locks at different places. Without knowing the existence of a lock, it broadcasts all potential code words to locks in the vicinity. The code words are encoded in an array as shown in Figure 3. A code word is represented by the combination of some bits in the array. For example, Figure 3 shows that two bits (bit 5 and 11) is a code word. The Master Key sets all code words in the array.

A lock verifies the code word bits in the array. If all the bits that represent a code word are set to 1, the lock believes that there is a code word match. For example, when the lock sees the bits 5 and 11 are set in the array, it believes that the Master Key knows the code word. Then, the lock notifies the Master Key its information and requests the Master Key to supply the key. (Note that the code words are just for the Master Key and locks to identify each other. They are not used as keys to open the locks.)

In addition, the Master Key will randomly set bits in the array to reach a fixed percentage of bits set in the array. By default, the Master Key uses an 8192-bit array with 50% of the bits set in the array. Eavesdroppers who listen on the channel but do not understand code words find arrays with the same length and the same number of bits set. They do not know which Master Key is sending the discovery message.

Our encoding scheme is scalable. Hundreds of code

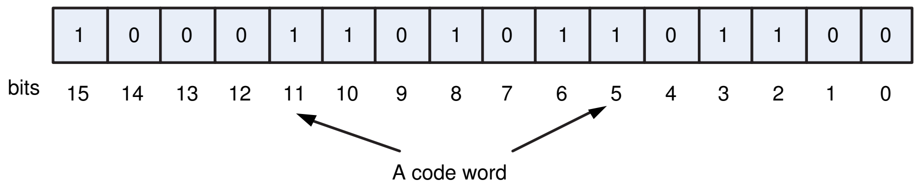

Figure 3: Code words encoded in an array

words may be expressed in one network packet. For instance, if the Master Key uses an 8192-bit array, of which 50% are set, and on average each code word is 5 bits, then at least $8192 \div 50\%5 = 819$ 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, in which no two code words set the same bits.

This is an application of the Bloom filter [10]. Our Bloom filter generation (discussed in the following Section 4.2) has different constraints than those used in database and networking [11]. Unlike other approaches, our approach allows different number of bits to represent a member (code word). In addition, random bits are set in the Bloom filter. The mathematical properties are different and we discuss and prove them in Section 4.3.

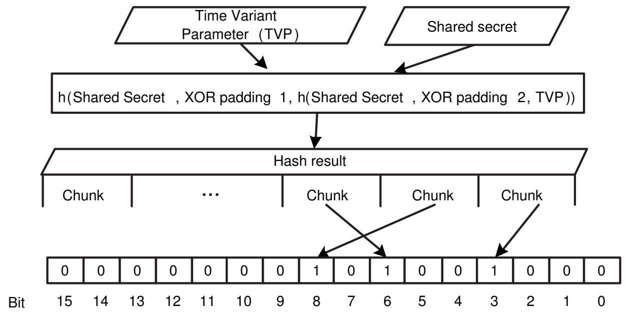
Time Variant Parameter (TVP)
Shared secret
h(Shared Secret , XOR padding 1, h(Shared Secret , XOR padding 2, TVP))
Hash result
Chunk
...
Chunk
Chunk
Chunk
0 0 0 0 0 0 0 1 0 1 0 0 1 0 0 0
Bit 15 14 13 12 11 10 9 8 7 6 5 4 3 2 1 0

Figure 4: The generation of a code word

4.2 Generating and Verifying One-time Code Words

Figure 4 illustrates the generation of a code word in an array. All bits in the array are initially set to zero. A time variant parameter (TVP) and the shared secret are the two inputs to $h(*)$. A TVP consists of a timestamp and a random number. We assume that the Master Key and the locks have loosely synchronized clocks, and thus they do not need to maintain large caches to verify whether or not a TVP is fresh. Function $h(*)$ is the hash-based message authentication codes (HMAC) proposed in [8]. MD5 or SHA-1 [27] is used in the place of $h()$.

The hash result is further separated into chunks. The size of the chunks, x, depends on the length of the array, which is 2^X bits. For example, if the array size is $8192 = 2^{13}$ bits, the chunk size is 13 bits. The value of a chunk serves as an index to the array. The Master Key uses it to set the corresponding bit in the array. A code word is represented by a combination of several bits. The Master Key uses several chunks to set the bits. For example, three chunks are used to set bits 8, 6, and 3 to represent a code word as shown in Figure 4. The hash results are 128 bits and 160 bits when MD5 and SHA-1 are used, respectively. Thus, they can be separated into 9 and 12 chunks.

For all potential locks that the Master Key wants to discover, it repeats the above process using the same TVP and the shared secrets. Then, it sets the code word bits in the same array. Next, the Master Key generates random numbers in the range between 0 and $2^X - 1$. It uses the random numbers as indices to set bits in the array until the number of bits that are set reach a fixed ratio. Last, the array and the TVP are broadcasted to query the locks in the vicinity.

Code word verification is efficient and independent of the number of code words in an array. First, a lock checks whether 50% of the bits are set in a Bloom filter. Then, it calculates the hash results using the TVP and its copy of the shared secret. Next, it verifies whether the bits indexed by the chunks of the code word in the array are set to one. If all the bits are one, there is a code word match. If any bit is not one, then the code word does not match.

The code words are one-time code words. Each authentication session, the Master Key and a lock use a different TVP. Thus, their code word bits set in the array are different in sessions.

4.3 Mathematical Properties of the Code Words

Unlike other applications [11, 18], which are based on having each element in a Bloom filter with the same length, the Master Key may use various code word lengths and set random bits in the array. Specifically, the mathematical property of the false positive rate is different from other applications.

Proposition 1. *The code word encoding scheme has:*

1) No false negative case, that is

$$p(codewordmatch|keyowner) = 1$$

2) The false positive rate is:

$$p(codewordmatch|notkeyowner) = (\frac{m}{n})^k. \qquad (1)$$

Proof.

1) The first part means if the Master Key has a key to operate a lock, the lock will always find the code word match in the array. No false negative case is a property of the Bloom filter. In our case, the Master Key and the lock use their shared secret and the same TVP as inputs. The hash results are the same. The Master Key uses the hash results to set the bits in the array and the lock uses the bits to verify. So, they will always find matches.

2) False positive match is possible because the bits that are set to 1 in the array are not exclusive. When a lock finds a code word match, the same bits may be set by other code words or the random bits that the Master Key generates. If the code word bits follow uniform distribution, the false positive rate in our case is a typical sampling with replacement problem. By our design, the number of bits set and the array length, m/n, is a fixed ratio. Given that the array is not generated by a key owner, the probability to find a code word match is $= (\frac{m}{n})^k$.

□

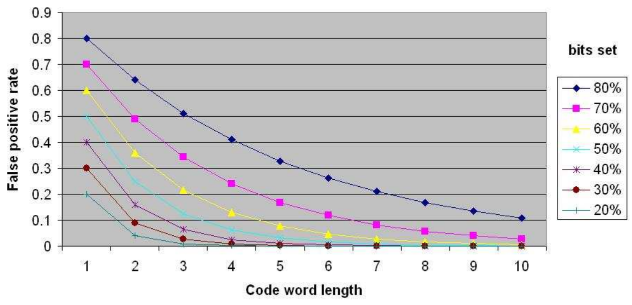

Figure 5: False positive rates decrease as the length of the code word increases

Examining Equation 1, we find that a key-lock pair can control their false positive rate by selecting the code word length. When the length of a code word increases, the false positive rate decreases as shown in Figure 5. The m/n ratio at around 50% provides a good span of false positive rates. 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 of 0.03, and a code word of 10 bits has a false positive rate less than 0.001. To further reduce unnecessary overhead, another hash algorithm or another shared secret may be used to generate a code word of 20 bits or longer.

It is computationally difficult for an eavesdropper who does not know the shared secret to find a hash result from the array of the code words. For example, if the array is 8192 bits, 50% bits are set, and a code word is 5 bit, the probability to find the partial hash result is less than 10-18. We prove the following proposition.

Proposition 2. *The probability to find a hash result from an array of code words is* $\frac{1}{\sum_{k=n_1}^{n_2}(\frac{m!}{(m-k)!})}$, *where k is a code word length and m is the number of bits set in the array.*

Proof. If an eavesdropper does not know the code word length, he needs to try different lengths from $k = n_1$ to $k = n_2$. For each k, an eavesdropper needs to select k bits from m bits and then permutate the k chunks to guess a hash result. Assume the code word bits follow uniform distribution, there are $(\frac{m!}{(m-k!)})$ permutations. Therefore, the probability to find the hash result is $\frac{1}{\sum_{k=n_1}^{n_2}(\frac{m!}{(m-k)!})}$. Note that the hash result found may be only part of the hash result because code words may not use all the chunks to set the bits in the array. □

4.4 Code Words Follow Uniform Distribution Over an Integer Set

When we analyze the mathematical properties of the code words, we assume that the values of the chunks of the hash results follow the uniform distribution over an integer set [32]. That is any bit in the Bloom filter is equally likely to be set. If the bits are not equally likely to be set, we may not acquire desired results as we discussed in last sub section.

We use the chi-square goodness-of-fit tests to determine if this assumption (null hypothesis) holds. Since MD5 and SHA-1 are the underlying functions used in HMAC, we test results by using both functions. For MD5, our algorithm may use up to 9 chunks. For SHA-1, up to 12 chunks may be used. Every individual chunk is tested as shown in Table 3. We use a randomly generated number as the shared secret, a 2-byte timestamp, 14 1-byte random numbers as a TVP, and an array of 213-bit. For each chunk, 6,400,000 code words are generated and the number of occurrences for each outcome is counted.

We calculate the chi-square test statistics and select 5 percent as the significance level. Table 3(a) and 3(b) shows the test results. If the value is greater than 8402.5, then it is significant, which means we reject the hypothesis. There is only one test, chunk 2 for the hash result using SHA-1, is insignificant. However, it may be given a false result. Because, given that the significance level is 5 percent, it seems reasonable that 1 out of 21 tests is false. Thus, we do 20 runs to test the chunk again. Table 3(c) shows that only 1 of the 20 tests (run number 15) is significant. Therefore, we believe that the values of

Table 3: Pearson's chi-square tests for code word chunks

(a) x^2 tests on chunks generated using MD5.

Chunk no.	X^2	p-value
1	7986.62	0.9458
2	8181.34	0.5281
3	8084.54	0.7969
4	7978.28	0.9527
5	8334.58	0.1314
6	8303.95	0.1883
7	8150.03	0.6239
8	8155.17	0.6086
9	8084.77	0.8801

(b) x^2 tests on chunks generated using SHA-1

Chunk no.	X^2	p-value
1	8302.02	0.1925
2	8498.40	**0.0087**
3	8106.66	0.7442
4	8138.58	0.6276
5	8019.14	0.9111
6	8082.59	0.8011
7	8286.50	0.2271
8	8192.96	0.4917
9	8264.93	0.2807
10	8032.32	0.8930
11	8036.60	0.8866
12	8121.27	0.8057

(c) 20 runs of tests on chunk no. 2

Chunk no.	X^2	Run	X^2
1	8115.41	11	8217.66
2	8272.11	12	8308.70
3	8228.07	13	8273.77
4	8081.83	14	8300.54
5	8046.90	15	**8441.22**
6	8186.90	16	8381.66
7	8201.50	17	8166.25
8	8215.75	18	7876.70
9	8332.42	19	8145.32
10	8142.11	20	8231.54

the chunks follow the uniform distribution over an integer set.

5 Choose the Amount of Exposure and Exposure Orders

Our code word encoding scheme is flexible that the Master Key and a lock can determine their amount of exposure based on their requirements. They may choose from partial exposure to full exposure. Meanwhile, their amount of exposure is independent of other code words in the array because an array always has the fixed ratio of bits set. To control the amount of exposure, the Master Key and a lock can simply choose the number of bits used in the code word. The more bits used the more precisely the Master Key exposed. If the code word is very precise (very low false positive rate), it is considered as full exposure.

If a lock has privacy concerns or power constraints and only wants to expose to its key owners, the Master Key may specify a precise code word. If the Master Key wants to make sure a certain lock is in the vicinity before its precise exposure, the Master Key may specify a partial code word with few bits. The use of partial code word changes the order of precise exposure from a key to send authentication information first to a lock to send authentication information first. Precise exposure at a later time has the advantage to protect privacy. If there is any mismatch and unnecessary exposure, a party that exposes later can avoid exposure. If both the Master Key and a lock have concerns and want the other party to expose first, they may progressively expose partial code words and verify each other. We described this approach and analysis in [36]. In short, our design enables a lock and the Master Key to choose their order of exposure and partial or full exposure in a message, such that the probability $p(K_i \in \{K_M^L\}) = a, a \subset (0, 1)$ and the probability $p(L_j = L_{intended}) = b, b \subset (0, 1)$.

From a lock's perspective, the preciseness that the Master Key exposes is determined by the probability, $p(keyowner|codewordmatch)$. That is the probability that the code word is generated by a true key owner, if the lock finds a code word match. Based on the definition of conditional probability, it equals to $p(keyowner|codewordmatch)/p(codeword, atch)$. After applying multiplication law and law of total probability to the numerator and denominator respectively, we get the following equation. (It is similar from the Master

Key's perspective.)

$$p(keyowner|match) = p(match|keyowner) \times p(keyowner)/p(match|notkeyowner) \times p(notkeyowner) + p(match|keyowner) \times p(keyowner) \quad (2)$$

where $p(keyowner)$ is the percentage of key owners among all people who send discovery messages at a place; $p(codewordmatch|notkeyowner)$ is the false positive rate; and is as we discussed in Proposition 1. Note that Equation (2) is only valid at the lock's place. At other places, $p(keyowner|codewordmatch)$ is not defined because $p(keyowner)$ is unknown.

Figure 6 illustrates the relation between the number of bits in a code word and $p(keyowner|match)$ for various $p(keyowner)$ values. The false positive rates, $p(match|notkeyowner)$, are based on setting 50% of the bits in the Bloom filters. Note $p(keyowner|match)$ is for one lock at a place. Different locks may have different $p(keyowner|match)$ values at the same places.

If the Master Key speaks code words vaguely, it may use few bits for a code word. The eclipse area in Figure 6 suggests the number of bits to use for different values. Nevertheless, overhead occurs at the lock's side. The lock may unnecessarily involve in the interactions with non-key owners. The probability of the overhead is $1 - p(keyowner|codewordmatch)$. We will show in Section 7 that the overhead in terms of processing time is small.

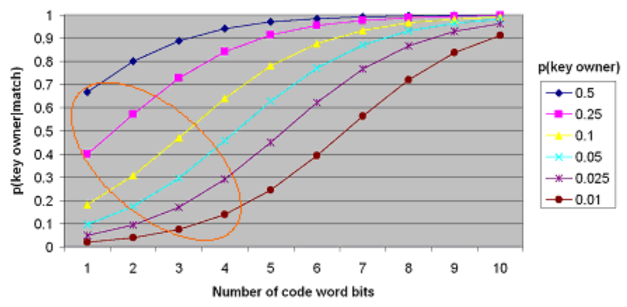

Figure 6: The relationship between the number of bits in a Bloom filter format code word and $p(keyowner|codewordmatch)$

6 Mutual Authentication Protocols for Different Key-Lock Types

In this section, we present protocols for three types of key-lock relations. A unique key is owned by one or a few owners to open a lock. A lock and the Master Key share a unique secret. An individual key is that its key owner can be identified among a group of owners. Beside the unique shared secret for the group as the unique key, a lock and the Master Key share an individual secret with each owner. A *group key* is also owned by a group of people. The lock is able to verify a key, but key owners are not differentiable. They may share some plain text to discover each other.

The initialization processes for all types of keys are the same. We assume that shared secrets are delivered from locks to the Master Keys via secure channels. Locks indicate the number of bits for code words.

6.1 The Unique Key

The protocol for the unique key is shown in Figure 7(a). In the first message, the Master Key sends code words to locks in the vicinity. After a lock finds a code word match, it indicates the last bit of the matched code word and proves its knowledge of the shared secret by replying back a hash result. Last, the Master Key supplies another hash as the key to operate the lock. In the latter two messages, the Master Key and the lock may use any authentication approach that they prefer, for example certificates. Without loss of generality, we use the HMAC as discussed in Section 4.2. (The Master Key and a lock exchange hash results directly). In the first and second messages, both the Master Key and a lock post challenges, TVP1 and TVP2, respectively; while in the second and third messages the other parties respond based on the challenges.

Usually, a lock and its few key owners use this type of keys to exchange precise code words. For example, one uses the unique key to lock and unlock a car. This key type minimizes unnecessary authentication between a lock and the Master Key.

6.2 The Individual Key

Some locks need to identity individual key owners among many key owners. The challenge for the Master Key design is that the Master Key holds many keys and a lock has many key owners. If we directly use the unique key approach, locks find that many false positive cases happen. To address the problem, we use domain secrets and individual secrets. A domain secret is shared and used by all key owners to discover the lock, whereas an individual secret is used by the lock to identity an individual key owner. The individual secret is only shared between a key owner and the lock and it is unique.

Figure 7(b) shows the protocol for the individual key. The first message uses the domain secret. In the reply message, the lock proves its knowledge of the domain secret by generating the hash. 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 7(b), the lock signs the hash result. The lock sends another Bloom filter in the second message that encodes code words for every key owner. The code words

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 TVP consists of a t_X and a R_X.
$(\ )_{KX}^{-1}$ 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 hash result 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:	$BF_M(R_M, t_M, S_{domain}), R_M, t_M,$
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:	$R_L, t_L, MB_M, Hash_M(R_L, t_L, S_{individual})$

(b). The protocol for the individual 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:	$R_L, t_L, Hash_M(R_L, t_L, S_{Group})$

(c). The protocol for the group key.

Figure 7: The Master Key protocols

are generated from the individual secrets using the same approach that the Master Key generates the code words. Furthermore, the lock sets random bits in the array to reach a fixed ratio of bits set. In the third message, the Master Key indicates its identity by specifying the matched code word and supplies the hash result as the key.

6.3 The Group Key

Unlike the individual key, the group key has the requirement that a lock cannot differentiate key owners from their keys. This means that all key owners should have the same key. Nevertheless, the Master Key initiates the authentication process. A malicious lock may provide different key owners with different keys. Based on the keys, the lock differentiates key owners and responds accordingly. We suggest the following two approaches.

The Master Key and a lock may use the unique key protocol. But a key owner only speaks few bits of a 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 code word array. If the lock replies with an incorrect hash, the key owner knows that he has a different key than other owners.

If the overhead caused by the false positive cases is large, a lock and its owners may use more bits for the code word. However, the code word is generated from some plain text, as shown in Figure 7 (c). 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 usage 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.

6.4 Revocation

To revoke a unique key, a lock invalidates the shared secret. If there is more than one key owner, a new shared secret needs to be delivered to the other key owners. 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.

To revoke a group key from a key owner, all other key owners need to update their group keys. Not all key owners are online all the time, and thus timely delivery of the new key may be a problem. If an owner updates his key when he finds that the key has expired, the lock 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.

6.5 Formal Verification of the Protocols

We use BAN logic [12] to verify our protocols to assure that the bindings are correct and messages are fresh. Our extension is added to the logic for the verification of code words and hashes. We add three logic constructs:

$P\infty^{y}Q$: P shares a secret Y with Q. Q may be an individual or a group.

$(M \subset G)$: M is a member of group G.

$P[CW]Y$: P finds a match in a code word, CW, which uses Y as a shared secret. If the code word is in the array, there is a possibility that the party that generates the array knows the shared secret, Y. When a hash is used, the party that generates it knows the secret.

The message-meaning and nonce-verification rules are extended as follows:

$\frac{P_{\infty}^{Y}, P[CW]Y}{P|\equiv Q|\ CW}$: When P finds a matched code word, P knows that Q once said it. If Q is a member of a group, P only knows that one member (including P) once said the code word. If a hash is received, P is sure that Q said it. If the code word is received, P knows there is a probability that Q said it.

$\frac{P|\equiv Q|\ CW, \sharp(CW)}{P|\equiv Q|\equiv CW}$: Based on the freshness of the code word, P believes that the code word or hash is fresh.

Since the verifications of the protocols are similar, we only discuss the protocol for the individual key type. Following the BAN logic's procedure, we convert the protocol to an idealized protocol as shown in Table 4(a). Then we explicitly write our assumptions in Table 4(b). Next, we deduct step-by-step to reach conclusions and check whether the conclusions are consistent with our expectation. Due to space limitations, we omit the lengthy deduction of the protocol and only show the stepwise results in Table 4(a). We reach the conclusions: a lock believes that the Master Key has the key to operate it, $K_i \in \{K_m^L\}$, and the Master Key believes that the lock is the intended lock, $L_j = L_{intended}$.

7 Performance Measurement

We implemented our protocols and measured performance on a set of PDAs. Handheld devices such as cell phones or PDAs are good candidates for the Master Key, since people regularly carry them. Locks may have diverse processing and communication capabilities. Some may have limited processing power, while others may be powerful to support hundreds of key owners. We used a Compaq iPAQ as the Master Key (an ARM SA1110 206 MHz processor, 64MB RAM, and a D-Link DCF-650W wireless card) and a Dell AXIM X5 as a lock (Intel PX250 400 MHz processor, 64MB RAM, and a Dell TrueMobile 1180 wireless card). The PDAs run Microsoft PocketPC 3.0, and the wireless cards are set to 2Mbps in the 802.11 ad hoc mode.

Table 5 shows processing and communication times of the major components of the Master Key and lock. The processing time is an average measurement of 100 runs. It takes about 231 ms for the Master Key to generate and send the first message with 820 code words. It takes the lock about 4 ms to generate and send the second message, and the Master Key about 3 ms to generate and send the third message. In case of the individual key, it takes the lock another 169 ms to generate a Bloom filter with 500 key owners. One protocol run takes less than half second in this extreme case, in which the Master Key specifies 820 code words and a lock has 500 key owners. Therefore, our design is efficient in most cases.

8 Discussion

During our design of the Master Key, an interesting question was raised: when there are multiple locks at a place and the Master Key owner has the privileges to access them, to which lock should the Master Key send the operation code? For example, suppose Bob walks to his office and pushes a button on the Master Key to unlock the door. However, the mailroom is also close by. Which door should the Master Key unlock? If the antenna on the Master Key is directional, then the door at which the Master Key points is unlocked. Nevertheless, if Bob also uses the Master Key to unlock his computer in the office, then the office door and the computer may in the same direction. Should both the computer and the door be unlocked? It may be convenient that both the door and the computer are unlocked. Or perhaps the Master Key may store a rule such that unless the office door is unlocked, the computer will not be unlocked. Alternatively, the Master Key may utilize location or proximity information if such information is available. Only if the Master Key is within a certain distance, the computer accepts the unlock command. For a greater challenge, if Bob has two cars parked side by side outside his house, which car should be unlocked when Bob pushes the unlock button on his Master Key? The Master Key may list the locks that reply back in the order of the frequency of the locks'

Table 4: Formal verification of the protocol for the individual key

(a) The idealize protocol of the individual key and the stepwise results of our verification

Msg no.	Idealize protocol	Stepwise results
1	$BF_M\{R_M, t_M\}$	$L\| \equiv (\subset G)\| \equiv BF_M$
2	$Hash_L\{R_M, t_M\}, Hash_L\{R_L, t_L\}\}^{-1}_{KL}, BF_L\{R_L, t_L\}$	$M\| \equiv L\| \equiv Hash_{Sdomain}, M\| \equiv L\| \equiv BF_L$
3	$Hash_M\{R_L, t_L\},$	$L\| \equiv M\| \equiv Hash_{Sindividual}$

(b) Assumptions

$M\| \equiv M^{Sindividud} \leftrightarrow L, L\| \equiv M^{Sindividud} \leftrightarrow LL, M\| \equiv L^{Sdomain} \leftrightarrow LL, L\| \equiv M^{Sdomain} \leftrightarrow LL$
$L\| \equiv \sharp(t_L), M\| \equiv \sharp(t_L), L\| \equiv \sharp(t_M), M\| \equiv \sharp(t_M), L\| \equiv \sharp(R_L), M\| \equiv \sharp(R_L), L\| \equiv \sharp(R_M), M\| \equiv \sharp(R_M), M\| \equiv\xrightarrow{K} L$

Table 5: Performance measurement of the major protocol components

Party	Operation
The Master Key	Generate an 8192-bit array with 820 code words. (Average code word length is 5 bits.)
The Master Key	Generate and set random bits in the array to reach 50% of bits set.
The Master Key	Send the first message.
Lock	Generate and check the code word in the array in the first message.
Lock	Generate a hash.
Lock	Send the second message.
The Master Key	Waiting time for the reply message.
The Master Key	Verify the hash from the lock and generate another hash.
The Master Key	Send the third message.
Lock	Generate an 8192-bit array with 500 key owners. Each key owner is identified by 2 bits.
The Master Key	Generate a code word from an individual secret and check it in the array.

usage, and then let Bob select one. Note that Bob selects a lock and not a key. The fundamental difference between multiple keys and the Master Key is that a user does not need to remember the key for the lock.

The Master Key protocols that we discussed so far are susceptible to the Mafia fraud attack, as are other entity authentication protocols. In our case, for example, an adversary may put a device near Bob's office and a device near Bob's house. When Bob pushes a button on the Master Key to unlock his office door, the adversary's devices relays messages - the first discovery message to Bob's house, the house's reply message back to the Master Key, and the Master Key's unlock command to Bob's house door to gain access to Bob's house. If the Master Key notifies Bob that two doors are ready to unlock and lets Bob select a door to unlock before it sends the third message, Bob may notice that something abnormal is occurring.

Mafia fraud attacks may not have countermeasures by cryptography alone. Presently, there are some representative solutions: using location information [16], measuring the transmission time [5, 7], and using multiple communication channels simultaneously [2]. These approaches can be adapted and fit into our protocols. For instance, the Master Key and a lock use location information to generate code words. An attack can 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 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. The problem is important, but it is out of the scope of this paper.

9 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 and supports various key-lock interactions. Users do not need to remember the relations between keys and locks.

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. 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 with 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 Master Key needs contact and contactless interfaces for various locks. The current design requires users to identify the correct interface to access a lock. Different interfaces may reduce the usability or might even cause authentication errors if a user infrequently accesses a lock. Proper user interfaces or visual hints may help users to choose the proper interfaces. This is an interesting issue and it is out of the scope of this paper.

The Master Key will be further extended to function more than just a set of keys. It may possibly represent a person's real life roles. For instance, the Master Key may be used as a remote control for a TV. Therefore, the TV channels that are available to Bob may be different from what are available to his son. Nevertheless, the more functions that the Master Key supports, the more challenging it is for automatic authentication.

Acknowledgements

This paper is an extension of a paper that appeared in the Proceedings of the Forth IEEE International Conference on Pervasive Computing and Communications, March, 2006.

References

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

[2] A. Alkassar, C. Stüble, and A. R. Sadeghi, "Secure object identification or: Solving the chess grandmaster problem," *Workshop on New security paradigms*, pp. 77-85, Ascona, Switzerland, 2003.

[3] Association for Payment Clearing Services, *UK Card Fraud Losses Reach l504.8M*, 2005. (http://www.epaynews.com/downloads/APACS%20 UK%20Card%20Fraud%20PR%208%20March%2020 05.pdf)

[4] AXCESS Inc web site, *Personnel Access Control,* 2005. (http://www.axcessinc.com/)

[5] S. Brands, and D. Chaum, "Distance-bounding protocols," *Advances in Cryptology - EUROCRYPT'93*, pp. 344-359, Lofthus, Norway, 1993.

[6] A. Beaufour, and P. Bonnet, "Personal Servers as Digital Keys," *2nd IEEE Annual Conference on Pervasive Computing, and Communications*, pp. 319-328, Orlando, Florida, 2004.

[7] T. Beth, and Y. Desmedt, "Identification tokens - or: Solving the chess grandmaster problem," *Advances in Cryptology Crypto'90*, pp. 169-176, 1991.

[8] M. Bellare, R. Canettiy, and H. Krawczykz, "Keying hash functions for message authentication," *Advances in Cryptology-CRYPTO'96*, *LNCS 1109*, pp. 1-15, 1996.

[9] M. Blaze, "Rights amplification in master-keyed mechanical locks," *IEEE Security & Privacy*, vol. 1, pp. 24-32, 2003.

[10] B. Bloom, "Space/time trade-offs in hash coding with allowable errors," *Communications of ACM*, vol. 13, no. 7, pp. 422-426, 1970.

[11] A. Broder, and M. Mitzenmacher, "Network applications of bloom filters: A survey," *Internet Mathematics*, vol. 1, pp. 485-509, 2005.

[12] M. Burrows, M. Abadi, and R. Needham, "A logic of authentication," *ACM Transactions on Computer Systems*, vol. 8, no. 1, pp. 18-36, Feb. 1990.

[13] K. Cameron, *The Laws of Identity*, Microsoft Corporation, May 2005. (http://msdn.microsoft.com/en-us/library/ms996456.aspx)

[14] D. Chappell, *Introducing Windows CardSpace*, Chappell & Associates, Apr. 2006. (http://msdn. microsoft. com/en-us/ library/ aa480189.aspx)

[15] M. Corner, and B. Noble, "Zero-interaction authentication," *Conference on Mobile Computing, and Networking (MobiCom)*, pp. 1-11, Atlanta, Georgia, USA, 2002.

[16] Y. Desmedt, "Major security problems with the 'unforgeable' (feige)-fiat-shamir proofs of identity, and how to overcome them," *SecuriCom'88*, pp. 15-17, Paris, France, 1988.

[17] Ensure Technologies, (http://www.ensuretech.com/ products/technology/technology.html♯HowXyLoc Works)

[18] L. Fan, *et al.*, "Summary cache: A scalable wide-area web cache sharing protocol," *IEEE/ACM Transactions on Networking*, vol. 8, pp. 281-293, June 2000.

[19] B. Ferg, *et al.*, "OpenID authentication 2.0," Dec. 5 2007. (http://openid.net/specs/openid-authentication-2_0.html)

[20] J. Ferrari, *et al.*, "Smart cards: a case study: IBM corporation," *IBM Press*, 1998.

[21] G. Goebel, *Codes, Ciphers, & Code breaking,* Jun. 1 2004 (http://www.vectorsite.net/ttcode.html)

[22] E. J. Goh, "Secure Indexes," 2004. (http://crypto. stanford.edu/ eujin/papers/secureindex/index.html)

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

[24] iButton home page, 2005. (http://www.maxim-ic.com/products/ibutton/)

[25] A. Juels, "RFID security, and privacy: A research survey," *IEEE Journal on Selected Areas in Communications,* vol. 24, pp. 381-394, 2006.

[26] M. Mana, M. Feham, and B. A. Bensaber, "Trust key management scheme for wireless body area networks," *International Journal of Network Security*, vol. 12, pp. 75-83, 2011.

[27] A. Menezes, P. v. Oorschot, and S. A. Vanstone, "Handbook of applied cryptography," *CRC Press*, 1996.

[28] K. Nohl, *et al.*, "Reverse-engineering a cryptographic RFID tag," *17th USENIX Security Symposium*, pp. 185-193, San Jose, CA, 2008.

[29] J. Pearson, *Securing the Pharmaceutical Supply Chain with RFID and Public-Key Infrastructure (PKI) Technologies*, Texas Instruments, June 2005. (http://www.ti.com/rfid/docs/manuals/whtPapers/wp-Securing_Phar-ma_Supply_Chain_w_RFID_and_PKI_final.pdf)

[30] S. Prabhakar, S. Pankanti, and A. Jain, "Biometric recognition: security, and privacy concerns," *IEEE SECURITY & PRIVACY*, vol. Mar./Apr., pp. 33-42, 2003.

[31] N. Ragouzis, *et al.*, "Security assertion markup language (SAML) V2.0 technical overview," *OASIS Open*, 9 Oct. 2006. (http://www.oasis-open.org/committees/documents.php?wg_abbrev=security)

[32] J. Rice, *Mathematical Statistics, and Data Analysis*, 2nd Duxbury Press, Thomson Learning, 1995.

[33] Smart Card Alliance, *Smart Card Implementation Profiles*. (http://www.smartcardalliance.org/industry_info/profiles.cfm)

[34] D. Semiconductor, "Requirements of remote keyless entry (RKE) Systems," Nov. 11, 2004. (http://www.maxim-ic.com/appnotes.cfm/appnote_number/3395)

[35] F. Zhu, M. Mutka, and L. Ni, "A private, secure, and user-centric information exposure model for service discovery protocols," *IEEE Transactions on Mobile Computing*, vol. 5, pp. 418-429, 2006.

[36] F. Zhu, *et al.*, "Private, and secure service discovery via progressive, and probabilistic exposure," *IEEE Transactions on Parallel, and Distributed Systems*, vol. 18, pp. 1565-1577, 2007.

Feng Zhu received the B.S. degree in computer science from East China Normal University, the M.S. degree in computer science and engineering from Michigan State University, the M.S. degree in statistics from Michigan State University, and the Ph.D. degree from Michigan State University. He is an assistant professor at The University of Alabama in Huntsville. He was a program manager at Microsoft and a software engineer at Intel. His current research interests include pervasive computing, security for pervasive computing, computer networks, and distributed systems.

Matt Mutka received the B.S. degree in electrical engineering from the University of Missouri-Rolla, the M.S. degree in electrical engineering from Stanford University, and the Ph.D. degree in Computer Science from the University of Wisconsin-Madison. He is on the faculty of the Department of Computer Science and Engineering, Michigan State University, East Lansing, Michigan, where he is currently professor and department chairperson. He has been a visiting scholar at the University of Helsinki, Helsinki Finland and a member of technical staff at Bell Laboratories in Denver, Colorado. His current research interests include mobile computing, wireless networking, and multimedia networking.

Lionel M. Ni earned the B.S. degree in electrical engineering from National Taiwan University in 1973, the M.S. degree in electrical and computer engineering from Wayne State University, Detroit, MI, in 1977, and the Ph.D. degree in electrical and computer engineering from Purdue University, West Lafayette, IN, in 1980. He is Chair Professor in the Computer Science and Engineering Department at HKUST. He served as the Department Head from 2002 to 2008. He also serves as Director of HKUST China Ministry of Education/Microsoft Research Asia IT Key Lab, Director of HKUST Fok Ying Tung Graduate School Digital Life Research Center, and Chief Scientist of the National Basic Research Program of China (973 Program) on Wireless Sensor Networks. Before joining HKUST in July 2002, he was a professor in Computer Science and Engineering Department at Michigan State University, where he started his academic career in 1981.