Using Agents to Improve Security and Convenience in Mobile E-Commerce

Jian Tang* and Jari Veijalainen♣

Department of Computer Science and Information Systems, University of Jyväskylä
jian@cs.mun.ca, veijalai@cs.jyu.fi

Abstract

The main driving force for the rapid acceptance rate of small sized mobile devices to do e-commerce is its increased convenience and efficiency in performing simple transactions compared with the stationary machines. Two of the weaknesses of this type of e-commerce are: (1) due to its small size, a mobile device can easily get lost and get stolen; (2) it is neither very convenient nor cheap for a mobile user to perform transactions that require more than just a few messages, scanning of catalogues, or negotiations. Personal identification numbers (PIN) are currently used both for protection of the devices and for authentication of the user to solve (1). Sophisticated thefts might still uncover the PIN inside the mobile devices. (2) has not deserved much attention by the research community so far. In this paper, we propose to use agent technology to cope with both problems. The related protocols are presented and the improvements on the security over the existing schemes are studied.

Keywords: mobile e-commerce, security, authentication, agent architectures

1. Introduction

Toward the end of the last century, the integration of the e-commerce and wireless communication started to emerge. As e-commerce technologies advance into the twenty first century and more technical hurdles are eliminated in mobile computing, there is no doubt that the popularity of mobile e-commerce will further increase. Many mobile e-commerce transactions are carried out by small sized, hand-held mobile devices, such as smart phones with embedded SIM card to enhance processing power and storage capabilities. (In the following, unless otherwise mentioned, phrase ‘mobile device’ refers to the aforementioned hand-held mobile device, and phrase ‘small mobile e-commerce’ refers to e-commerce using small sized, hand-held, or palmtop, mobile devices.) The early reports have indicated that this type of mobile e-commerce is rapidly gaining the acceptance from the market [12,13].

The main driving force for the rapid acceptance rate of mobile e-commerce is its increased convenience and efficiency in performing simple transactions compared with the stationary machines. For example, some transactions involve only a single request and reply between a customer and a merchant, and possibly another round trip message exchange between the customer and the bank. Using a mobile device, people can perform this kind of transactions anytime and anywhere, and use very simple interface that fits only the small sized screen provided by the mobile device. It is secure provided the mobile device (e.g. using the embedded ‘smart card’) has the capability of performing cryptographic algorithms and the access to these algorithms is further authenticated by the secret personal identification number (PIN) of the user. A PIN is a short sequence of digits or mix of digits and letters. In this paper, we assume that it has a length of four, because the SIM cards currently used in GSM terminals commonly use this length.

Mobile e-commerce has, however, also some weaknesses compared with the other types of e-commerce. Among other things, due to its small size, a mobile device can easily get lost. (According to a recent statistics in one of the major cities in Finland, the most commonly found things at a local lost-and-found service center are cellular phones.) Loosing the device always means a possibility of unauthorized usage of it. It can also easily fall into the prey of a theft, either because first lost, or because other people deliberately steal the device.

Another problem is that it is not very convenient for a mobile user who is constantly travelling or on the move to perform transactions that require more than just a few round trip message exchanges, require scanning of catalogues or fancy graphics, not to speak about bargains requiring extensive negotiations with the merchant.

For the first problem area, the most widely used solution is a PIN authentication scheme that stores the PIN itself (or its variations, like a hash-value) inside the internal memory (or the embedded SIM card) of the mobile device [12]. A user must enter the PIN first, which

---

*The work was performed while the author was on leave from Memorial University of Newfoundland, St. John’s, NF, Canada.

♣ On leave; current affiliation GMD-FIT, Schloß Birlinghoven, Sankt Augustin, Germany.
is then verified by the device, before he/she is allowed to start the radio transmitter and further any transactions. In communicator type devices (see [14] for an example) the integrated PDA can be used without PIN and transactions started from that side. The phone cannot be used, however, without giving the PIN if the radio part of the device is not on already.

The PIN scheme provides some protection. However, since a mobile device (as well as the embedded smart card) is nothing more than a small computer, a sophisticated thief may still find a way to access its internal structure to uncover the PIN. If the device is on when stolen, a thief can in the worst case directly use it to initiate transactions successfully. If each transaction requires the user to input a separate PIN, the thief needs to find out or guess one PIN. And so on.

Considering the relatively high volumes of the devices which may fall into the victims of thefts, the damages so caused may not be negligible. Also, this kind of risks may deter the users from using mobile e-commerce facilities.

For the second problem, we have not found any literature that has paid attention to it in the context of mobile e-commerce, except [16] where initial considerations are laid down concerning the requirements for mobile e-commerce.

In this paper, we propose to use agents to solve both problems. The basic idea is that instead of storing the entire PIN digits in the mobile device, we store part of the PIN in a remote machine where an agent is run. The PIN verification then involves both the mobile device and the agent, which must verify their respective parts of the PIN.

To cope with the second problem area, we simply let an agent do the negotiation and therefore free the mobile user from the constant attention he/she has to pay while he/she is on the move. In addition to the common benefit provided by the agent technology for mobile computing environment, such as reduced bandwidth requirements, increased efficiency, enhanced functionality, etc., using agents in mobile e-commerce as above is beneficial in a unique way. First, by distributing the PIN digits geographically and anonymously, it is more difficult for unauthorized users to uncover the PIN in its entirety, even if he or she had the device. Second, mobile users who are on the move are most likely to initiate simple transactions only which involve, non-sophisticated negotiations, if any. In these negotiations the search space is small and is normally a priori fixed, and both sides of the negotiation rarely have incentive to misrepresent their preferences. (Examples are flight seat reservations, car rentals, purchasing an item with a set of preferred delivery dates and costs). In these kind of negotiations agents can be implemented in a more straightforward manner than those in the general cases where a negotiation may involve game-playing strategies [2,4]. Also, in most negotiations of this type, agents can do the task without interventions from the mobile users. Thus the communication latency is not a main concern. As a result, the agents do not need to move during the entire negotiation session even though the mobile users move. If the mobile users need to have another negotiation, another agent process can be instantiated, possibly at a different site. In general, process instantiation is much easier than process migration, as the former does not require complicated mechanisms to preserve the states of the process being migrated.

The rest of the paper is organized as follows. In Section 2, we present an architecture for our scheme, explaining the functionalities of the components and the required security capabilities. In Section 3, we describe in detail how the two functions stated above are realized, including some important protocols involved. In Section 4, we outline the implementation. In Section 5, we present a risk analysis for the commonly used PIN verification schemes and the agent assisted PIN verification schemes proposed in this paper. We conclude the paper by discussing some related work and summarizing the main results.

2. An architecture and capabilities of components

2.1. Components and functionalities

We use an architecture that is similar to that for mobile telephony service. Shown in Figure 1 is an overall view of this architecture.

The wireless network is divided into service areas, which cover multiple cells, actually an entire network operated by one operator. Each service area contains an agent broker, and a number of agent servers. An agent broker assigns mobile devices to agent servers. Each agent server manages a number of agents. This includes, among other things, creating and destroying agents, as well as keeping track of the mappings between agents and the corresponding mobile devices.

There are two kinds of agents, PIN-verifying agents (PVA) and negotiation agents (NA). A PVA verifies the last (two) digits of a PIN for a mobile device, and always exists, at least conceptually. In a real implementation the a process instance running the PVA code can invoked when needed. An NA performs negotiation in an e-transaction on behalf of a mobile user, and is created on demand by the mobile user.

How do the mobile device and the infrastructure communicate? We assume that the mobile device and the infrastructure rely on the same PKI and that the keys are distributed by secure means. For instance, the private key of the mobile device is delivered on the SIM card or by
the manufacturer of the device. The public keys of the entities are distributed in as reliable a manner as the private keys. The private keys recorded in the devices are allocated for private persons, namely for that one who is the owner of the device. That is, the Certification Authority has established a connection with the person and the public and private key. It also records the device identity where the keys are stored. We assume for simplicity that the device identity is an IP number (i.e. we assume all-IP infrastructure). The reader is urged to consult e.g.

The agent broker corresponds about the Home Location Register in the GSM system. A mobile device is permanently registered with one of the agent brokers. That is, the IP-number of the device is known by it, and further information about the owner of the device and public key allocated to the user. The device knows the IP-number of the agent broker.

At the time of registration, the agent broker assigns the mobile device to one of the agent server in its service area (e.g. optimal one from the network traffic pont of view; this is not further discussed here). The agent broker also sends the IP-number of the device and the public key of the owner attached to the device to the agent server. The agent server then allocates a PVA for the device, sending the public key and IP-number of the device.

The PVA then takes contact with the mobile device using the IP-number as device identity. It sends its agent id (an IP-number) to the mobile device, along the seed rs for random number generation. This information is encrypted using the public key of the device, i.e. the message is of form \( E_{\text{pva}}(\text{ag IP,rs}) \). The device decrypts the message with its private key and stores agent IP and the seed rs. As response to the message the mobile device sends its IP, a random number generated using the seed rs, and the last two digits of the PIN, encrypted by its private key, i.e. \( E_{\text{mb}}(\text{Mbid,r1,P(-,2)}) \) to the PVA. The PVA decrypts the message, generates a random number \( r1' \) from the initial seed, checks that it corresponds to \( r1 \) delivered by the device and stores the digits of the PIN in a secure database, along the new seed and Mbid. The secure database is accessed only by the PVAs and under no circumstances can the access right be granted to any other process.

When a mobile device wishes to authenticate a mobile user, it requests the assistance from the PVA allocated to it by establishing a transport connection using its agent id, ag_IP. In general, at the time of registering, a mobile device and the agent broker are necessarily in the same service area. Once allocated, the PVA

---

**Figure 1. An architecture for the agent systems**

will stay on the same machine and associated with the same mobile device. In other words, no migration will take place for PVA. This does not mean that the device could not roam, only that it will take contact with the same PVA having the identity ag_IP.
A mobile device can also be temporarily registered with one or more agent brokers, possibly at different services areas. During the process of temporary registering, the agent broker also assigns an agent server to the mobile device. The agent server then instantiates an NA for the mobile device. In other words, such an NA did not exist before the registration. This is different from permanent registration where the allocated PVA always exists. At the time of temporary registering, the mobile device and the agent broker, and therefore the NA instantiated at that time, are usually in the same service area. As the mobile device moves and changes service areas, it may request new NAs to be instantiated to engage in different negotiations. Physically, both kinds of agents are ordinary processes with the capability of communicating with other processes. Their main communication partners are the client software in the mobile device and the entities with whom the mobile user wants to negotiate.

2.2. Security capabilities during operation

Agents are involved in activities requiring security measures. Thus they need to have the capability of signing a document. On the other hand, since an agent server must be sure that it is managing the agents for the right mobile users (i.e., the owners, not someone who pretend to be the owners, of the mobile devices.), it must be able to verify the requests it receives. A mobile device is the entry point for a user to start an e-transaction. Thus it must preserve the capabilities of enforcing necessary security measures, for example, signing a message, performing limited certification, and authenticating the mobile user. The last job is important in light that a mobile device can easily fall into the prey of a thief. A good authentication scheme can minimize the damage that can possibly occur in this case.

To fulfill the above requirements, we use the public key cryptography. We assume an agent server possesses a private key from some reliable external source when it is created. When an agent server creates a PVA or NA, it at the same time creates a public key pair, passes the private key half on to the agent. The mobile device also receives its private key from some external source. At the time of registration when it is assigned to an agent server, the public key of the mobile device is delivered to the agent server by the certificate issued by a Certificate Authority (CA). Similarly, it receives the public key of the agent server through the certificate from a CA. The agent server then passes public key of the mobile device to a PVA. Since both the PVA and the mobile device now have each other’s public keys, they are able to establish secure communication subsequently. One thing the mobile device needs to do at this time is to send the last two digits of the PIN to the PVA. It does this by encrypting them using the public key of the PVA and then encrypting it again using its own private key, and sending the result to the PVA.

When the agent server creates an NA upon request by a mobile device, it also passes the public key of the mobile device to the NA. In addition to its own private key, a mobile device may store the public keys of some other business partners, such as those with whom the mobile user may later negotiate.

A mobile user has a unique identity. This may be represented by different things on different occasions for different entities (for example, a combination of the name and birth date, password, private key, etc.). As far as the mobile device is concerned, however, it is the PIN or PINs which uniquely identify a user who has the necessary access rights to perform certain operations. Thus, if the user is able to present the right PINs for the device while requested, the conclusion is drawn: “This is the user for which I have the private key stored”; that is, the device is now representing a certain individual towards the e-commerce infrastructure.

3. The Methods

3.1. Using agents to assist in PIN verification

A mobile device requires the user to deliver a PIN when a user requests to (1) create an NA or (2) start a new transaction or (3) continue an on-going transaction but the idle time exceeds a predefined threshold. The third case includes a user inquiring its NA about the progress so far in the negotiation or continuing a transaction by his own.

As mentioned before, to enhance security we split the PIN into two halves of two digits each. The first two digits are stored inside the mobile device, and the last two digits are controlled by the PVA allocated to the mobile device upon registration (cf. above). When the user enters the PIN, the software client of the mobile device verifies the first two digits. If they are correct, it sends the second two digits in the encrypted form to the PVA for verification. It then accepts or rejects the user’s request depending upon the results from the PVA.

Specifically, let p(4-k,-) and p(-,k) be the first 4-k and the last k digits of the PIN, respectively. H( ) be a one way hash function. For any message X, we use $E_{y, pk}(X)$ to denote the encryption of X using the public key of entity y (‘as’ for agent server, ‘pva’ for PVA, ‘mb’ for mobile device). After the user enters the PIN, the client software verifies $p(4-k,-)$. Whether or not this is valid, it sends a message containing MBid and $E_{pva, pk}(\text{date}, \text{time}, r1, H(\text{MBid}, p(-,k)))$ to the PVA. The MBid uniquely identifies the mobile device. We assume for simplicity that it is the IP-number of the device.
In order to detect man-in-the-middle attack, a random number \( r_1 \) is included. The date and time are the current date and time, which are included to prevent replay attack. When the PVA receives this message, it decrypts the encryption part of the message. It then checks the freshness of the message by examining the date and time (allowing certain time interval for the error in the clock synchronization and traffic delay). If the message is not fresh, it stops, otherwise it verifies \( H(MBid, p(\cdot, k)) \) using the MBid contained in the incoming message and \( p(\cdot, k) \) maintained in the secure database. Variable \( k \) varies between 1 and 3 (see below). It then sends \( E_{mb, pk}(MBid, date, time, r_1, yes/no) \) to the mobile terminal. The protocol is specified in Figure 2.

\[
\begin{align*}
MB & \leftrightarrow PVA: \\
MB & \rightarrow PVA: MBid, E_{pva, pk}(date, time, r_1, H(MBid, p(\cdot, k))) \\
PVA: & \text{decode } E_{pva, pk}(date, time, r_1, H(MBid, p(\cdot, k))) \text{ with private key;}
\end{align*}
\]

- if date and time are too old stop.
- Evaluate \( H(MBid, p(\cdot, k)) \); if ok \( \text{PinValid} \leftarrow Yes \)
  - else \( \text{PinValid} \leftarrow No; \)

\[
\begin{align*}
PVA & \rightarrow MB: E_{mb, pk}(date, time, r_1, \text{PinValid}) \\
MB: & \text{if date and time are not current or } r_1
\end{align*}
\]

- is incorrect then
  - discard the message
- else if \( \text{PinValid} = Yes \) and
  - \( p(4-k, \cdot) \) valid then
  - Accept the mobile user’s request
  - else reject the mobile user’s request

Figure 2. Protocol for PIN Verification.

3.2. Using agents for negotiation

3.2.1. Creating a negotiation agent

A mobile user can request to the mobile software client for creating a negotiation agent. After verifying his PIN, the mobile device then sends a message to the agent server. The message contains a command, NaCreate, a transaction id \( Tid, MBid, date, time, \) and a random number \( r_2 \). Command NaCreate tells the agent server that this request is for creating an NA. \( Tid \) is used to identify other messages that may be sent subsequently for the same transaction. Date and time are included to prevent replay attack. Similar to \( r_1 \), \( r_2 \) is used to detect man-in-the-middle attack. When the agent server receives this message, it creates an NA and passes necessary parameters to it. After creating the NA, the agent server sends back a message to acknowledge the mobile device for the successful creation of the NA. The protocol is described in Figure 3.

\[
\begin{align*}
MB & \leftrightarrow AS: \\
MB & \rightarrow AS: \text{NaCreate, Tid, MBid, } E_{as, pk}(r_2, date, time, \\
& H(\text{NaCreate, Tid, MBid})); \\
AS: & \text{decrypt the encrypted part and verify the value of function } H; \\
& \text{check the date and time for the freshness of the message; if the above checks are ok, create an NA and a public key pair, passes the private key part, the public key of the mobile device, Tid, MBid to the NA;}
\end{align*}
\]

\[
\begin{align*}
AS & \rightarrow MB: E_{mb, pk}(r_2, date, time, \text{NaCreated}); \\
MS: & \text{decrypt;}
\end{align*}
\]

- if message is not fresh or the value of \( r_2 \) is incorrect
  - discard the message

Figure 3. Protocol for NA creation.

3.2.2. Communicating with the negotiation agent

Once the NA has been created, the mobile user may need to communicate with it from time to time. The need for such a communication arises mainly in two occasions: (1) immediately after the NA is created, the mobile user needs to instruct the NA on how to proceed with the negotiation and (2) the mobile user wants to know the progress of the negotiation at some particular point in time.

We consider the first case first. Let \( P \) be the party the mobile user wants the NA to negotiate with. Before the negotiation starts, \( NA \) and \( P \) must ‘know about’ each other. More precisely, they must know each other’s public keys. Although this can be done using normal approach, i.e., through CA, this would require the NA to register with the CA. Since an NA is just an ad-hoc process in nature, and it exists only for the duration of a negotiation, frequent registering and deregistering is a hassle. On the other hand, the mobile user knows the public keys for both \( NA \) and \( P \). Thus we can use a more efficient method similar to token-mediated certification in [6], i.e., the mobile software client serves as a temporary CA for \( P \) and \( NA \). More precisely, we have:

\[
\begin{align*}
MB & \rightarrow P: MBid, S_{mb, pk}(\text{PUKY}, \text{NAid}); \\
MB & \rightarrow NA: \text{StartNeg, MBid, Instr, } S_{mb, pk}(\text{PUKY}, \\
& Pid, H(\text{Instr})); \\
S_{mb}(m) & \text{is a message digitally signed by the mobile device. Command startNeg sent to the NA signifies that this message is for it to start a negotiation. Instr is the instruction that tells NA how to proceed with the negotiation.}
\end{align*}
\]
There are proposals for data structures that can be used by agents negotiations on some specific topics. In general, for different negotiation topics, the data structures may be different. For example, in [3], the authors propose DAG for negotiations for buying manufactured goods. To maximize the success rate, the instruction should specify more than one alternative and order them according to preferences for flexible attributes such as prices, models, manufactures, delivery dates, etc. In the following, we use a simpler example to illustrate the idea. Suppose the mobile user wants to make a booking for a flight from Helsinki to Hong Kong from a travel agent (TA). Since it takes time for the TA to work out an acceptable travel schedule, the mobile user leaves the negotiation process to NA. We assume that all the authentication procedures described previously have been done successfully. Now the user enters the instructions into the interface provided by the mobile device, which is implemented as a table shown in Table 1. (For clarity, we use a pseudo language for the specification.) The negotiation itself is just a sequence of message exchanges. The NA sends the messages containing the information about the booking it wants. The travel agent then responds with regard to the availability of the NA’s requests. In general, multiple rounds of message exchanges may be needed to reach a compromise. Each time when the NA receives the response from the travel agent, it checks if every item contained in the response matches the instruction. If not, it will send a new request to indicate the problem. In Figure 4, we list a partial sequence of message exchanges.

<table>
<thead>
<tr>
<th>NA</th>
<th>TA</th>
<th>Figure 4. A partial list of message exchanges in the negotiation.</th>
</tr>
</thead>
<tbody>
<tr>
<td>depart city</td>
<td>Helsinki</td>
<td>FinnAir, Hel to HK, June 30</td>
</tr>
<tr>
<td>arrival city</td>
<td>Hong Kong</td>
<td>Flight: Fin 128, depart Hel: 10:00, June 30, avl Paris: 14:00, June 30,</td>
</tr>
<tr>
<td>airline</td>
<td>1. FinnAir, 2. AirFrance, 3. Lufthansa</td>
<td>&lt; depart Paris: 21:00, June 30, avl HK: 15:00, July 1</td>
</tr>
<tr>
<td>depart date</td>
<td>date: June 30 ≤ date ≤ July 5</td>
<td>not acceptable: depart Paris – avl Paris &gt; 6</td>
</tr>
<tr>
<td>depart time</td>
<td>time: 10:00 ≤ time ≤ 22:00</td>
<td>&lt; cannot accommodate</td>
</tr>
<tr>
<td>max. waiting hours</td>
<td>6</td>
<td>AirFrance, Hel to HK, June 30</td>
</tr>
<tr>
<td></td>
<td>for intermediate connection</td>
<td></td>
</tr>
</tbody>
</table>

Table 1. Instruction for booking an airline ticket

*: required items in NA’s request

4. About Implementation

Both kinds of agents can be implemented as user level processes. For PVA, since its only function is to verify the 1-3 digits of the PIN, its logic is straightforward. It first performs a SQL-like query to retrieve the public key and the two digits of the PIN from the database. It then calculates the hash values of the PIN digits, checks if it matches the one in the incoming message, and sends the results back to the mobile user (with proper encryption as described before). We mentioned before that the PIN digits are stored in a secure database which is accessed exclusively by the PVA. Logically, the public keys of mobile devices are stored in a separated public key database whose security measure can be less stringent. However, since the only purpose of storing public keys is

---

1 In reality the mobile user should be allowed to revise the old instructions based on the current circumstance in the negotiation. Since this does not add any technical sophistication, we omit it here.
for the access by the PVA, there is no reason why it should not be stored together with the PIN digits. Thus we view them as the two attributes of the same entities. In terms of relational schema, for example, the structure is

<table>
<thead>
<tr>
<th>MBid</th>
<th>PUBLIC KEY</th>
<th>PIN DIGITS</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

MOBILE DEVICE

where MBid is the primary key (and also the only search key) of the relation MOBILE DEVICE. A simplified implementation in the form of object-oriented design is described in Figure 5.

Class PinVerifyingAgent {
  Public
    PinVerifyingAgent ( ); // constructor
    void GetData(ID *Mbid, VALUE *EncryptVal); // receive message
    SearchPkyPin(KEY Mbid, PKY *Mbpky, DIGIT *i, DIGIT *j); // search for public key and PIN digits for Mbid
    boolean ValidPin(ID Mbid, DIGIT i, DIGIT j); // verify the PIN digits sent by Mbid
  .......
  Private
    KEY PrivateKey;
    VALUE H(ID Mbid, DIGIT i, DIGIT j); // one way hash function
    .......
}

Figure 5. PVA object definition.

For NA, since its existence is on per mobile device basis, there is no need to create a public key database for it, rather, it maintains the public key for the mobile device internally. The main data structure it uses is for storing the instructions. Note that logically an NA is devoted only to a single kind of negotiation. Thus its data structures need to accommodate only a single kind of instructions. Physically, however, if we know in advance the kinds of negotiations in which the mobile user may possibly get involved, provisions can be made to accommodate all these kinds of instructions. The NA can then select the kind of instructions in accordance with the mobile user's specifications at run time. An implementation for an NA to negotiate for a flight connection is sketched in Figure 6.

Class NegotiatingAgent {
  Public
    void NegotiatingAgent( );
    void GetDataFromMobile( ... INSTRUCTION *instr, ...);
    void Negotiating( ...);
  .......
  Private
    KEY PrivateKey;
    KEY PublicKeyOfMobileDevice;
    .......
}

struct INSTRUCTION {
  struct AirLine* {
    char* name;
    int preference;
    struct AirLine* next;
  } al;
  struct DepartDate* {
    char* date;
    int preference;
    struct DepartDate* next;
  } dd;
  struct DepartTime* {
    char* earlier; // left and right ends of preferred time interval
    char* later;
    int preference;
    struct DepartTime* next;
  } dt;
  char* DepartCity;
  char* ArrivalCity;
  int InterMedWaitHours;
  .........
}

Figure 6. NA object definition.

5. A Risk Evaluation

We now give an evaluation of the risk of losing the PIN in the commonly used PIN verification scheme and that presented in this paper. First note that due to the stringent cryptographic schemes used for the messages, the adversarial attacks such as replay, man-in-the-middle, etc., are not possible in either scheme. We assume that it would be possible for a thief to uncover the PIN from the system with probability Px, uncovering the private key and using it correctly during the "window of opportunity" (see below) would have probability Py << Px (e.g. Py < 0.01*Px). Thus we concentrate on the case where the mobile unit is stolen and possibly the PIN is
guessed or uncovered subsequently, but the private key remains uncovered or at least unused.

Second, usually there is a time interval, "window of opportunity", from the point of time when the device is stolen to the point when the user discovers his/her losing the device. After he/she discovers this, the user is likely to take necessary actions to block the use of the device to prevent it from launching any possible transactions. In other words, the thief may cause damage only in this time interval. Since our purpose is to compare the two schemes, not to calculate the exact value for the risk, we omit this detail in both schemes to simplify the calculation. That is, we assume that the thief uses a strategy (to be described below) to maximize the success rate of uncovering the PIN. He/she is either successful and causes the damage, or unsuccessful and gives up or the window of opportunity closes.

After stealing a mobile unit, the thief may guess the PIN, or try to access the internal memory of the device or the network server to find out the PIN. More generally, he may try both. Note that the current mobile devices and the SIM cards are manufactured in such a way that they allows only a few trials for the PIN. If all the trials fail then the device and/or the SIM card will be locked. We assume three is the maximum number of trials the mobile device allows and that the window of opportunity is closed after that. We further assume that opening the locked device or card is not possible for a thief.

Also, without loss of generality, we assume that the thief will try both to access the internal memory of the mobile device or the network and to guess the PIN. More precisely, in the commonly used PIN verification schemes, the thief first tries to uncover the PIN. If he/she is successful, transactions are launched as soon as possible. If he/she is unable to uncover the PIN from the device/network, he/she begins guessing. This is done then as long as the either the guess is right or the device is locked and this ends the trials.

What is now the risk of finding PIN? We perform an analysis based on the probability of this to happen. The strategy of the thief is that he/she first tries to uncover the PIN both at the device and at the network, where ever it or part of it is stored. We thus assume that the thief knows where to search for the PIN. If uncovering does not work out at all or successfull only partially, he/she performs three guesses that we assume the device would allow. The latter can be performed in about ten seconds, the time for former depends on the window of opportunity the thief has.

Thus, for any scheme, the probability of finding out the right PIN using the above strategy is

\[ P(\text{uncover entire PIN or uncover part}+\text{guess the rest of PIN or fail in uncovering}+\text{guess the whole PIN}) = Pf \]

Let \( Px \) describe the probability of uncovering the whole PIN or part of it directly. \( Ppg \) is the probability of a pure guess for the whole PIN. For 4-digits PIN \( Ppg \sim 3 \times 10^{-4} \). \( Px \) depends on the mechanism used. We analyse \( Pf \) below in different cases.

Let us assume that the whole PIN is stored on the device and denote the probability of uncovering the whole PIN by \( Pd \) in that case. The probability \( Pd \) is larger than zero, but how much, is unclear. But it does not actually matter in this context, as we see below. Because the whole PIN is stored at the device (and the thief knows it), \( Pf = Pd + (1-Pd) \cdot Ppg \) in this case.

Let us then assume that the whole PIN is stored at PVA and denote the \( Px \) by \( Pn \) in that case. Now \( Pf = Pn + (1-Pn) \cdot Ppg \). In both of the latter cases we see that \( Pf > Ppg \) assuming \( Pn > 0 \) and \( Pd > 0 \).

Let then the PIN be stored partially at the device, partially at the network. We can assume that it is as easy to uncover a part of the PIN as in the case where the whole PIN with \( n \) digits is stored.

\( Pf \) becomes in this case

\[ P(\text{uncover both parts of the PIN}) + P(\text{uncover n-k digits at network, fail to uncover k digits at device, guess that k digits}) + P(\text{uncover k digits at the device, fail to uncover the n-k digits at network, guess that n-k digits}) + P(\text{fail to uncover n-k digits at network, fail to uncover the k digits at device, guess the whole PIN}) \]

Now, we assume that uncovering the digits at device and at the PVA server are independent of each other, i.e. \( Pd \) and \( Pn \) are independent in this case. Further, uncovering and guessing can be kept independent (i.e. uncovering one part of the digits does not increase probability of guessing the rest, and guessing one part does not increase the probability of uncovering the rest), we have three cases above, which are all exclusive. In addition, the pairs of clauses within the terms are also independent. Therefore, we can use the production and addition rule of probability theory. Also, notice that the thief can guess three times the remaining \( k \) or \( n-k \) digits after uncovering the other part and the guesses are independent of each other. Of course the same \( k \) digits that failed are not tried twice, but this does not make much difference here. Thus, we can sum up the probabilities above to reach an approximation

\[ Pf = Pd \cdot Pn + (1-Pd) \cdot Pn \cdot 3 \cdot (1/10)^k + (1-Pn) \cdot Pd \cdot 3 \cdot (1/10)^{(n-k)} + (1-Pd) \cdot (1-Pn) \cdot Ppg \]

Knowing that \( Pd \) and \( Pn \) must be rather small (at most \( 10^{-3} \)), we can ignore the terms \((1-Pn)\) and \((1-Pd)\). Thus we get for \( Pf \) the approximation

\[ (*) Pf \sim Pd \cdot Pn \cdot 3 \cdot (1/10)^k + Pd \cdot 3 \cdot (1/10)^{(n-k)} + Ppg \]
Coming now to Pf, how does it behave in different cases? First, it always holds: Pf > Ppg. That is, the possibility of uncovering the PIN increases indeed with the non-zero probability of a thief being successful in uncovering the PIN. How much? Assume that Pd = Ppg. Then in the non-mixed cases Pf ~ 2*Ppg. If Pd = 10*Ppg then Pf ~ 10*Ppg = Px, and if Pd = 0.1*Ppg, then Pf ~ 1.1*Ppg. The same holds for Pn. Thus, if the non-mixed uncovering probability Pd or Pn is at least as big as Ppg, then it makes sense to try to use mixed scheme.

In this vein, where does (*) reach its minimum in the mixed case? Evidently, when \( P_x = Pd*Pn + Pn*3*(1/10)^j + Pd*3*(1/10)^{n-k} \) reaches it. Assume, Pd and Pn are of the same magnitude, i.e. of form s*10^{j-1} and e*10^{j} for some j >1 and 0 < s, e < 10. In this case, clearly, Pf reaches its minimum if k = n-k, i.e. for n = 4, when k = 2. If the probabilities Pd and Pn are of different magnitude, then the minimum shifts. Assume e.g. that Pd = 100*Pn. Then, the minimum is reached by k = 1. This is natural, as many bits as possible should be kept there where their uncovering is most difficult (i.e. in this case at PVA).

As a summary, (*) reaches its minimum when the right side reaches its minimum, because in all cases the basic raw guessing probability Ppg is the same and the differences come from the uncovering part. Looking this in mind again at Pf in order to decide where to store the digits, evidently the mixed case is the most promising, because there we have probabilities that are certainly less than, say 0.01, multiplied and divided by 10, 100 or 1000. Thus the sum in (*) above remains smaller than any of the individual probabilities Pn or Pd. This is at least the case if the probabilities are about of the same magnitude. If they are not, the digits are stored unevenly (3-1, or 1-3) corresponding the differences in the values Pd and Pn. The rule is: store more digits there where the smaller of Pd and Pn holds, otherwise store them evenly.

Notice, that if the digits are stored opposite to the probabilities, then this scheme performs worse than one of the all-nothing schemes, and is not optimal. To see this, assume again that Pd = 100*Pn and that three digits are stored now at the device. Now the definition of Px above yields the following approximation for Px:

\[
Px \approx 100*P_n^2 + 3*P_n/1000 + 3*P_n*100/10 \sim 30*P_n.
\]

That is, it would be about 30 times easier to uncover the PIN from the device than if it was stored as a whole at the network server. But still, it is about three times more difficult to uncover the PIN in this distributed setting than if it was stored completely at the device. On the other hand, if only one digit is stored at the device, three at the server, \( P_{un} \approx 0.6*P_n < P_n \) holds. This is optimal. For 2-2 scheme we get the approximation \( P_x \sim 3*P_n, \) i.e. it is clearly worse than the optimal scheme, but performs much better than 4-0 (all digits at the device) or 3-1 (three at the device) schemes.

As a first conclusion, dividing the storage of the PIN at the network and at the device reduces the risk of uncovering it, if done properly. Second, the absolute values of Pn and Pd are not important for deciding the placement. If the magnitude of Pn/Pd is known then the placement of the digits can be decided in an optimal way. In this case the division of the digits at the device and at the PVA server yield the best protection, i.e. minimizes the possibility of uncovering the PIN. Third, 2-2 storage scheme performs better than 4-0 scheme in any case. Therefore, if nothing is known of the Pn and Pd then it can be used.

When makes it in practice sense to apply the division scheme? This causes, anyhow, quite much additional hassle. If the probability Pd above is of magnitude of the probability of pure guess, i.e. ca. \( 3*10^{-4} \) then it might already make sense. This would namely mean that the probability of getting the PIN by thief would grow with about 100 % if nothing is done. On the other hand, the division scheme could keep it essentially at the level of pure guessing; Assuming that Pn ~ Pd and 2-2 storage scheme, (*) yields Pf ~ 0.03*Pn + 0.03*Pd + Ppg ~ 1.06*Ppg.

The biggest threat that seems currently plausible is that the people begin to store diverse PINs in the hand-held devices in clear form or just barely encrypted. This raises the probability of uncovering the PIN by thieves drastically. And what is worse, the division scheme above does not help in this case.

6. Related Work and Concluding Remarks

Although e-commerce and mobile computing as two different disciplines have been studied extensively since 90’s in the last century, the research on their integration only starts to emerge. Nowadays, most of the transactions for mobile e-commerce are restricted only to simple ones. The protocols they employ are also straightforward. It seems that the best technological progress that has been made in the area of mobile e-commerce in the last several years is the use of smart cards [5,11]. Some products using smart card also start to appear on the horizon [5,12].

The agent technology has been studied under different disciplines, such as artificial intelligence, coordination theory, CSCW, e-commerce, mobile computing [1,2,3,4,7,8,9,10]. Although agents in different context are aimed at different goals, and the degrees of the complexities of their structures and functionalities also vary, their basic capabilities are the same, that is, communication, negotiation and coordination. The work that handles agents in negotiations, such as those in [2,4,8,9], deals with a more general setting and is not
appropriate for mobile e-commerce due to their complexity. Nether did these schemes take security as a concern. The agents in [15,18] have also negotiation aspects included, as well as many other aspects. Security is not addressed, as far as we could find out.

In this paper, we apply the agent technology to mobile e-commerce to attain two goals, enhancing security and increased adaptability of e-commerce transactions to the mobile environment. We describe an architecture for the agent systems, and present protocols for the interactions of the components that can cooperatively attain our goal. We give a risk analysis to show that our PIN evaluation scheme indeed improves the security over the existing schemes.

Themes for further study are the real probabilities $P_n$ and $P_d$ above and the applicability of the scheme. Refining the ideas about negotiating agents in mobile environments is also necessary.

References


Acknowledgements

The authors wish to thank the anonymous referees for their valuable comments and Prof. Vagan Terzian for his guidance in searching for the newest developments in the mobile e-commerce agent field and checking some of the probability calculations.