An Agent-Based Secure Internet Payment System for Mobile Computing

Artur Romão and Miguel Mira da Silva

Departamento de Matemática, Universidade de Évora
7000 ÉVORA – PORTUGAL
{artur,mms}@dmat.uevora.pt

Abstract. The SET protocol, in particular its purchasing phase, is intended for users connected to the Internet during an entire transaction. This requirement cannot be easily met in high communication costs and/or low bandwidth settings, typically found in mobile computing environments. In this paper we propose SET/A, a system that works according to the SET rules for purchasing operations, without forcing the user to be connected during the entire transaction. This is achieved by sending an agent to the merchant's server carrying all the necessary data to order and pay for the products he or she wants to buy. The paper shows that this can be achieved safely and efficiently, providing an alternative way for on-line payments using the SET protocol.

1 Introduction

The Internet is now considered as the privileged environment for electronic commerce. Yet, there is still some resistance from the public to buy products and services on-line and pay for them also on the Internet. For example, by browsing a company's Web server, ordering a product and paying for it filling a form with the credit card information. The main difficulty is that almost every Internet user has heard of credit card frauds, performed by hackers eavesdropping connections used to send those data – despite the fact that very few of those attacks have actually succeeded. Even the deployment of secure servers, based on protocols such as SSL or S-HTTP, is not enough, since the credit card information is deposited in the server, where it can easily be read by anyone with access to it (even if not authorized).

The concern for protecting the user's credit card information lead VISA and MasterCard, in association with major software and cryptography companies, into the development of the SET protocol [12]. SET provides important properties like authentication of the participants, non-repudiation, data integrity and confidentiality. Each player knows only what is strictly necessary to play its role, for example, the selling company never knows the buyer's credit card information, and the financial institution authorizing the transaction is not aware of the details of the purchase, like the nature of the products, quantities, etc. Paying for something using a credit card

under the SET protocol is clearly much more secure than doing it, say, in a restaurant, where the card is normally taken away from the customer's eyes.

SET is expected to give buyers and sellers the necessary confidence to launch Internet commerce definitively (despite some technical and non-technical problems that still exist). From the buyer's point of view, SET can be very comfortable, both to use (there will be many SET-compliant software tools to help the users with their credit cards on the Internet) and to trust (if we assume the financial institutions interested in its success are able to explain and convince users of its benefits).

On the other hand, SET is a very complex protocol, and may not be suitable under some technical conditions. In this paper we take a closer look at some computing environments, namely those where the mobility of the users is determinant.

Generally, the devices used in these environments have limited computational capacity and use slow and expensive connections to the Internet. It's our opinion that SET may be too demanding for this kind of equipment and connectivity, preventing on-line transactions for mobile users. We therefore propose a mechanism, called SET/A, guided by the SET rules and based on the *mobile agent paradigm*. With SET/A, the computational burden is taken away from the user's device, so it can be disconnected while the transaction is running.

The rest of the paper is organized as follows. Section 2 describes the SET purchase request transaction and discusses some design issues associated with the usage of SET in a mobile computing environments. In section 3 we introduce the notion of mobile agents and identify the requirements that are important when using agents in on-line payment systems, and the core of the SET/A scheme, i.e., its purchase request transaction is described. In section 4 some of the advantages and application scenarios are identified. Security issues are discussed in section 5. Finally, in section 6 we point out some conclusions and discuss possibilities for future work.

2 Overview of the SET Purchase Request

The SET protocol is composed of several kinds of transactions, ranging from certification of participants, to purchase requests, to payment processing. In this paper we are particularly interested in the *purchase request* phase, which can be outlined as follows (see Figure 1):

Step 1. An user, from now on called a *cardholder* (C), looks at a catalog (printed in paper, supplied in a CD-ROM or available on-line on the Web) of a company, called the *merchant* (M) and, after deciding to purchase something, sends a request to the merchant's server. The request includes the description of the services or the quantities of the goods, the terms of the order, and the brand of the credit card that will be used for payment.

Step 2. The merchant receives the request and sends back its own *signature certificate*[1] C_s(M), and the *key-exchange certificate* C_k(PG) of a *payment gateway*

[1] SET uses two distinct asymmetric key pairs, one for *key-exchange* (whose public key is contained in certificate C_k), which is used for encrypting and decrypting operations, and

(PG). The payment gateway is a device operated by a financial institution with which the merchant established an account for processing payments with the brand used by the cardholder. The merchant also sends a unique identifier, assigned to this transaction.

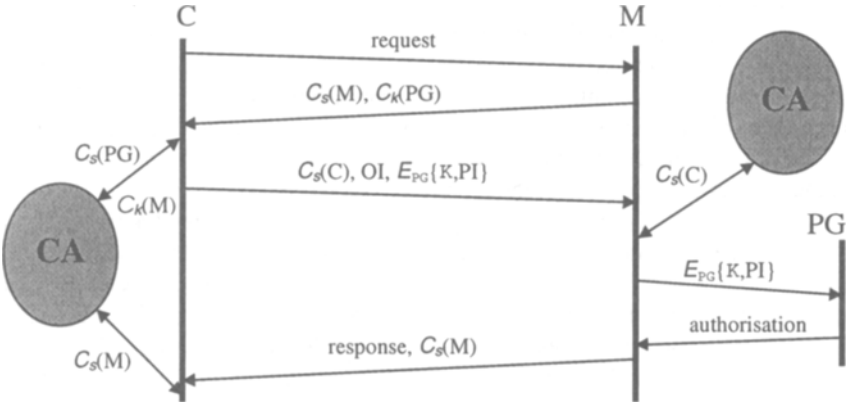

Fig. 1. SET purchase request transaction.

Step 3. The cardholder (i.e., his or her software) verifies the certificates by contacting a *certificate authority* (CA), receives back a confirmation that assures the authenticity and integrity of the data (the merchant had digitally signed it), and creates two pieces of information:

- The *Order Information* (OI), containing control information verified by the merchant to validate the order, card brand and bank identification. The OI also includes *a digest of the order description*, which includes the amount of the transaction and other elements such as quantity, size and price of the items ordered, shipping and billing addresses, etc. This data, *not included in the OI*, will be processed outside the scope of the SET protocol.
- The *Payment Instructions* (PI), containing the amount of the transaction, the card account number and expiration date, instructions for installment payments (if that's the case) and a couple of secret values to prevent guessing and dictionary attacks on the data, among other elements. The PI is encrypted with a randomly generated symmetric key, K.

Both elements will contain the transaction identifier and are dually signed, so they can later be linked together by the payment gateway. Then, the encrypted PI (X_K[PI]), and the key (K) used to encrypt it are encrypted into a *digital envelope*

---

another for *signature* (in certificate C_s), used for creation and verification of digital signatures. Therefore, each SET participant possesses two kinds of certificates, one for each key pair type. A merchant will have a pair of keys for each card brand it accepts.

(E_{PG}),using the payment gateway's public key.[2] Finally, the OI and the digital envelope are sent to the merchant, along with the cardholder's signature certificate C_s(C).

Step 4. The merchant verifies the cardholder certificate and the dual signature on the OI. The request is then processed, which includes forwarding the digital envelope to the payment gateway, for authorization (the details of this operation are outside the scope of this description).

After processing the order, the merchant generates and signs a purchase response, and sends it to the cardholder along with its signature certificate.

If the payment was authorized, the merchant will fulfill the order, by delivering the products bought by the cardholder.

Step 5. The cardholder verifies the merchant signature certificate, checks the digital signature of the response, and takes any appropriate actions based on its contents.

The software responsible for the cardholder's side of the protocol manages a data structure called a *digital wallet*, where sensitive data like certificates, private keys and payment card information are kept, usually in encrypted files.

The merchant will have a more complex system composed of several parts, doing different jobs: managing the dialog with cardholders, signing messages and verifying signatures and certificates with CAs, asking payment gateways for payment authorizations, and so on.

2.1 Dificulties with Mobile Computing

Wireless networks, especially cellular ones, have known explosive growth over the last few years, reflecting a world in which it's important to be active regardless of the location. As the Internet becomes more and more important for business transactions, it is natural that wireless technology be used to connect to this global network. The possibility of having all the resources and benefits offered on the Internet available while being away from home or office is particularly attractive. In *mobile computing* conditions it is desirable to have all the facilities usually found on the Internet, namely *the possibility of acquiring and paying for products and services*.

The kind of mobility we're interested in is based on portable devices with limited computing capacity and/or limited connectivity. For example, computing-capable mobile phones (e.g. Nokia 9000 Communicator [7]), PDAs (e.g. 3Com's PalmPilot [1]), handheld PCs (e.g. Psion Series 5 [9]), up to notebooks, connected to the Internet through a modem attached to a cellular phone (or with an internal GSM modem).

---

[2] We use the notation E_P{K,D} to represent a digital envelope meant to be sent to some participant P. The digital envelope contains a symmetric key K, encrypted with P's public key-exchange key, and data D, encrypted with K, i.e., the contents of the envelope are $X_{k(P)}$[K], where k(P) is P's public key-exchange key (in the case of SET it's really $X_{k(PG)}$[K,account_information], but we'll ignore that for the sake of simplicity), and X_K[D].

Bandwidth and Cost

There are several issues regarding the conditions in which mobile computing takes place. In the context of this paper, we're mainly concerned with the following:

- *Low bandwidth and poor connectivity* – terrestrial wireless network protocols like GSM or satellite-based systems like IRIDIUM [5], typically offer bandwidths in the range of 2,400bps to 9,600bps (although there are claims for much larger bandwidths with broadband satellite systems in the future [6]). On the other hand, the connectivity based on these systems is generally of low quality, with high error rates. These factors make it difficult to handle long, connected sessions, transferring large amounts of data.
- *High cost* – Using a cellular phone or a satellite-based connection is generally more expensive than through a traditional telephone carrier or ISDN. On the other hand, poor connectivity raises costs, since it leads to longer on-line sessions.

In mobile computing, bandwidth capacity is thus proportionally inverse to the cost, and this fact has to be taken into account when designing applications for these environments. In what follows these factors, *together*, will be our main motivation. Although bandwidth is important, it's not the only, and maybe not the most important factor – as stated above, this kind of problem tends to be solved in the near future.

SET in a Mobile Environment

In an error-prone environment like GSM or any other used for mobile communications, a user shopping on the Internet and trying to pay using SET compliant software may experience several connectivity problems during the payment operation. Even with recovery mechanisms, it's easy to imagine how frustrating it can be for the cardholder to deal with a series of connection interruptions, let alone the accumulation of state information both in the wallet and in the merchant's server, in order to let the transaction proceed. Even if it eventually succeeds, *its overall cost has probably been too high*.

What is needed is some kind of asynchronous mode of operation, in which the cardholder can send the request, disconnect, and later re-connect to receive the response from the merchant. This reasoning seems to suggest a typical message-passing (RPC-like) mechanism, but this is not suitable, for two reasons:

1. Three of the five steps of the purchase request transaction described above are executed on the cardholder's side, so there would have to be two messages: one to send the request, and the other to send the OI and the digital envelope, after receiving the first response from the merchant. Since the cardholder may be disconnected from an arbitrarily long period after sending the first request, the whole transaction would have to wait for this intermediate synchronization to happen.
2. On the other hand, if there would be a way to avoid this step and send a single message, this would mean disclosing sensitive information (e.g. account number) and letting it be used by a remote server in an unpredictable way.

This means that it is necessary to reduce the cardholder's role to two steps, the initial request and the final receipt of the response. The request sends all the necessary

information to complete the transaction successfully, but in such a way that the remote system can never take control of it. This clearly demands an *agent-based mechanism.* The cardholder may send an entity (the agent) with enough information for processing the entire transaction and, at the same time, capable of hiding the sensitive data from the outside.

3 SET/A: An Agent-Based Payment System

In this section we introduce the concept of *mobile agent* and propose a secure Internet payment system based on this model and on the SET protocol. We concentrate on the purchase request transaction, as this is the most important phase (from the cardholder's point of view) and the only one that really makes sense for our system. The other SET transactions refer to situations in which there isn't the notion of a buyer paying a seller.

3.1 Mobile Agent Applications

A mobile agent can be defined as a software element (program, procedure, object, thread, etc.), owned by a user or another software element, capable of *migrating* from one computer to another, to execute a set of tasks on behalf of its owner. Mobile agents are said to be *autonomous*, in the sense that they can take their own decisions while away from their host. This implies that a mobile agent (*agent*, for short) is not just a piece of data being transferred between systems, but may also carry some *logic* (i.e., *code*) and *state*, which enables it to perform parts of its tasks in one system, migrate to another and continue its work there.

Agent technology has received growing interest from the research community and has matured significantly in the last few years [10,13], however the number of applications using this technology is still scarce (see, for example, [3]). Electronic commerce is generally seen as one of the most promising application fields for mobile agents [2].

For example, the literature typically refers to a *buying agent*, leaving its host with the mission of querying several vendors about a certain product, determine which one offers the lowest price (or some other kind of preferential condition) and buying from this one, paying for it. Clearly there is the perception that agents are suitable for this kind of activity, and ability to pay is one of the desired properties they should have. The major concern is always how to do it in a secure way, in particular without revealing confidential information to the outside.

In this paper we propose a payment system called SET/A, based on the SET protocol and implemented with an agent travelling from the cardholder's computer, carrying all the relevant information, to the merchant's server. On arrival, the agent performs the cardholder's role and carries on a complete purchase request transaction with the merchant. From the merchant point of view, there will be no distinction between an agent and a real (i.e., human) buyer: a SET purchase request is being

performed with another entity, which represents a cardholder with valid certificate and payment data.

3.2 Agent Requirements

In order to be able to do its job, there are a few requirements that the SET/A agent must fulfill:

- *Small-sized* – since we're assuming low and/or expensive bandwidth, we have to minimize the time it takes to send the agent. On the other hand, transport media with low reliability make it difficult to transmit long streams of data. For example, a reliable transport protocol like TCP would require many re-transmissions [4]. Clearly, the agent should be as small as possible for better performance and smaller costs. This can be a problem if one wants sophisticated agents, as will be discussed below.
- *Survive inside hostile execution environments* – the cardholder wants to be sure that the agent will be able to complete the transaction as expected, without being disturbed by any external factor. This requires a relative degree of tolerance to merchants' server faults, as well as immunity to attacks trying to make the agent take unwanted decisions or perform unwanted actions.
- *Hide confidential information* – one of the main purposes of SET is to offer an appropriate level of security, so that the cardholder can be sure that none of the confidential data (card number, expiry date, etc.) is disclosed to any unauthorized party. Thus, SET/A has to ensure that the confidential data the agent carries is kept private.

3.3 Purchase Request

SET/A is meant to implement the purchase request phase of SET using mobile agents. It should be noted that SET/A is designed to be as compatible with SET as possible, only requiring significant modifications on the cardholder's side. The merchant software could remain unchanged, since its interaction with the agent is mostly the same as it would be with the cardholder. The only exception is that it must be aware that now it's talking to an entity residing in a host other than the cardholder's.

A purchase request under SET/A has a few more steps than in SET, since an agent has to be sent to the merchant's server, and come back to the cardholder's computer when the transaction is done. SET/A purchase request is described below, and Figure 2 illustrates the process.

Step 1. As before, the cardholder C chooses a merchant and builds a request with the same elements as in the original SET request. Then, an agent, A(C), is sent to the merchant's server, carrying the request, the cardholder's signature certificate, the account information and other data needed to compose an OI and a PI (OI_PI_data). Clearly, this data has to be protected somehow, but we'll deal with

that later (see Section 5). For now, we'll just assume it was encrypted with someone's key ($X_?$[OI_PI_data]).

Step 2. After arriving at the merchant's server, the agent sends (now locally) the request to the merchant M (i.e., its order processing software).

Step 3. The merchant returns a signed message with its signature certificate C_s(M), the payment gateway key-exchange certificate C_k(PG), and the transaction identifier to the agent.

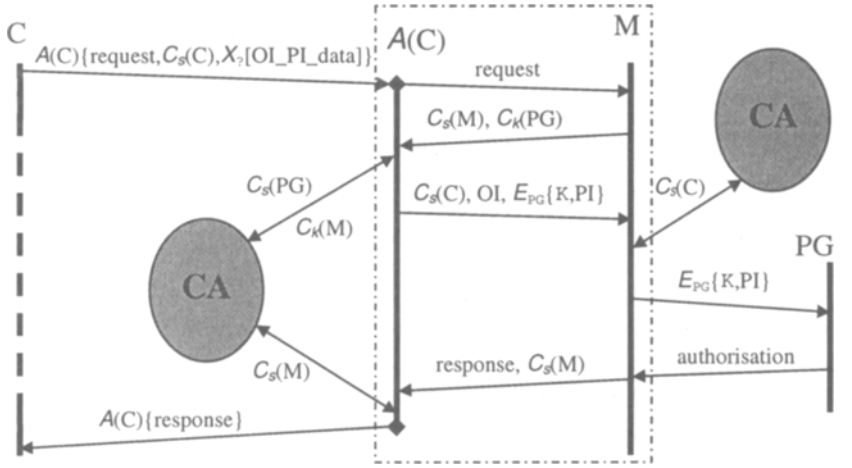

Fig. 2. SET/A purchase request transaction.

Step 4. The agent verifies the certificates and the signature, creates the OI and the PI and generates a dual signature of them. Next, it generates a random symmetric key K and uses it to encrypt PI. Finally, the payment gateway's public key is used to create the digital envelope E_{PG}, containing the encrypted PI (X_K[PI]), and the key K. This digital envelope, the OI and the cardholder's certificate C_s(C) are then sent to the merchant.

Step 5. The merchant verifies the certificate and the dual signature on the OI, and then proceeds as described above in step 4 of the SET purchase request (see section 2), sending a signed response along with its signature certificate.

Step 6. The agent receives the response, verifies the certificate and the signature, and migrates back to the cardholder's computer.

Step 7. The agent arrives at the cardholder's host, carrying the response from the merchant. The cardholder's software then proceeds as in SET's final step.

In step 6 we haven't detailed how the *rendezvous* takes place. One possible approach is to use a mechanism similar the one used by cellular phone operators to deliver SMS messages when the user re-connects.

4 Evaluation and Discussion

We have designed SET/A with mobile computing in mind, especially focused on the two factors we've been pointing at as determinant: low bandwidth and expensive connectivity. Our proposal of adopting an asynchronous computing model is a natural way to overcome those limitations.

Below we discuss the advantages and identify possible scenarios in which the usage of a payment protocol designed for disconnected settings is the best way, or the only one that makes sense economically, to do shopping on the Internet in a mobile computing environment.

4.1 Certificate and Signature Verifications

If one assumes that the agent has a reliable communication channel to the merchant, such as those based on SSL or S-HTTP, then there's no danger of data corruption due to failures or tampering with.[3] In this safe environment, most of the signing operations are not necessary anymore, since they are performed on data exchanged locally, through a previously authenticated connection. Consequently, there's no need for sending or verifying certificates.

Analyzing the SET/A steps outlined above, we conclude that message signing can be avoided in step 3, and signature and merchant certificate verification can be avoided in steps 4 and 6.

Furthermore, the merchant certificate verification by the cardholder can also be avoided in step 7 of SET/A, if the agent signs the final merchant's response with the cardholder's public key, brought within the certificate in step 1. This creates an awkward situation, in which a public key K_{PUB} is used to sign and a private key K_{PRIV} is used to verify the signature. But given that we're using a public-key system, this is only a formal problem, which can be solved by agreeing that K_{PRIV} is the agent's public key (in this case, only known to the cardholder) and K_{PUB} is the private key.

4.2 Processing Capacity

PDAs and mobile phones have limited processing capacity, and can hardly handle processor-demanding activities, such as those involving cryptography (key generation, encryption and signature generation and verification). Therefore, if one wants to use SET on this kind of devices, either the CPU capacity has to be increased, or the load has to be transferred to an external machine.

SET/A solves this problem by doing all the cryptographic work within the merchant's server (or, at least, outside the cardholder's device), which is supposed to be powerful enough to handle many of these activities concurrently. There is still the need to generate cryptographic keys on the cardholder's side, but this belongs to

[3] The establishment of this kind of connection implies the computation of secret values, which may be caught by an intruder. The security considerations regarding these data are similar to those discussed in Section 5, so we'll ignore them for now.

another phase in the SET protocol (the generation of the cardholder's certificates), not covered in this paper. Nevertheless, the keys and certificates could be generated in the cardholder's workstation and securely transferred to the less powerful device.

4.3 Minimizing Costs

One of the problems of electronic payments is the relative high cost of small-value transactions. Using a mobile device to access the Internet and perform a small-value purchase, setting up the connection and maintaining it opened for as long as a SET-based transaction takes place, may have a cost similar (if not higher) to the value of the transaction itself. Having to pay the double of the price of a product, however low it may be, is enough to discourage consumers to use this kind of connectivity for their shopping.

Operations like certificate verification may be unacceptably inefficient when performed using a slow and expensive connection. Eliminating the need for this kind of operations with SET/A contributes significantly for minimizing the consumer costs.

On the merchant's side, having an agent operating inside its server means having to let three (or just one, see section 4.1) additional certificate verifications be originated from it. But it is logical to expect that the merchant has good connectivity to the Internet, good enough so that a few more messages exchanged with some CAs will cause a negligible raise in its costs, if any at all. (But if that would make a substantial difference, the merchant could always charge a little extra over the prices of goods paid for with SET/A.) So, given that SET/A doesn't increase costs, there's no loss in supporting this protocol. On the other hand, it can be quite rewarding, from a marketing point of view.

4.4 Applications

Despite the advantages of SET/A, it can be difficult to imagine today a complex purchase transaction, requiring a lot of interaction with the cardholder during the phase in which the OI and PI are composed, being handle by SET/A. This can be the case when the merchant offers a set of alternatives concerning payment or delivery of the goods, from which the cardholder will have to choose. Letting the agent decide would be a possibility, but that would require more sophistication, i.e., more code, hence a greater size, and still wouldn't guarantee that the agent would be prepared to take the appropriate decision in every situation.

Of course, a moderated degree of sophistication can be acceptable in terms of agent size, and allow it to be used in situations where the possibilities are known in advance. For example, the agent could have enough knowledge to decide what should be the delivery method, depending on the maximum amount of money it is allowed to spend.

We consider SET/A more appropriate when *the contents of the OI and the PI are predictable*, even if the agent is not carrying them with it when migrating to the merchant's host (i.e., it has the ability to compose them at the merchant's). We identify a few areas where the application of SET/A can realize its full potential advantages. For example, when the cardholder is paying:

- The *phone bill* (although this kind of payment may already be done in many different, comfortable ways);
- A *movie in a pay-per-view TV system*, where the outgoing bandwidth is low; or
- A ticket at a theatre in a town where the cardholder is travelling to (and which will be waiting for him or her at the box office).

In all the examples above the elements needed to complete the transaction are well defined and can all be included in the data the agent carries to the merchant's server. This can be generalized to any payment for which the buyer knows in advance what to buy and how much will it cost. Also, those examples fit in the model of mobile computing and/or low bandwidth settings, where long shopping sessions are discouraged.

5 Security Issues

In a network payment system, one of the obvious concerns is to protect the user's critical data, in particular the credit card information. SET's usage of the dual signature mechanism and the encryption of the PI and account information (into a digital envelope with the payment gateway's public key-exchange key), ensure the necessary privacy of the critical data. In particular, the data is protected from a potentially hostile environment, such as the merchant server. Of course, protecting the merchant from the agent is also important, but that concern is clearly outside the scope of SET/A.

5.1 Protecting the Agent

For SET/A to be able to ensure the same level of protection as SET, without modifying the description outlined above, it must be possible for the agent to carry and use classified information without having to disclose it to the wrong entities. Also, the generation of the symmetric key K has to be performed in such a way that no one other than the agent and the payment gateway has knowledge of it.

On the other hand, replay attacks from the merchant, making the agent pay more than once, have to be prevented. This can be done with a protected (see below) internal counter.

Protecting the agent from hostile environments is a major research issue in mobile agent security, and we may consider applying some of the proposals to SET/A. For example, running the agent in a *tamper-proof environment* [14] or a *secure coprocessor* [15] seems a promising possibility. The agent would migrate to a protected (hardware) environment, securely attached to the merchant's server, and all the confidential data would be handled inside this environment. This solution would increase the security level at the cost of an additional investment in hardware from the merchant.

The "software alternative", in which the agent executes inside the merchant server without any hardware protection, requires some kind of wrapping to hide the secret

data. Cryptography is the natural wrapper, and some initial steps to execute *hidden computations* are being taken [11].

5.2 Protecting the Agent's Data

If we relax the requirement of following closely the SET protocol for purchase requests, there may be a better way to achieve the goal of protecting the data. First, recall that the data we want to protect from the merchant is to be encrypted with the payment gateway's key. If the cardholder knows in advance which payment gateway the merchant is using for the card brand, and if the OI and the PI can be built in advance, then the process of generating the dual signature, the random key, and the digital envelope can be performed before the agent leaves the cardholder's computer. When it migrates, all the information, completely protected, can now go with it, and the agent only has to give it to the merchant and wait for the response.

This approach has one major drawback: to obtain the payment gateway's certificate in advance, the cardholder has to perform an initial request to the merchant, wait for the response, and then proceed with the agent. This is similar to the message passing protocol described in section 2.1, and has the same disadvantages. Furthermore, the agent role would be limited to that of a "messenger", delivering the request and bringing back the response, which is, again, no more than simple message passing. (Anyway, the core of the transaction would still benefit from being performed by the agent inside the merchant's node.)

Note that the merchant software could still remain unchanged. In step 4 of SET/A the agent would receive both certificates, but only would have to verify the merchant's. In our opinion, this small difference in the agent's behavior and the cardholder's initial request to obtain the payment gateway's certificate, are two minor changes to the cardholder side of the transaction, and perfectly acceptable given the improved security they offer.

Both approaches need to be further investigated, in order to find a good compromise between them. The first one, based on real mobile and autonomous agents, is clearly more challenging in that it opens a much wider spectrum of possibilities. Electronic commerce is clearly a very promising application field for mobile agents and many research initiatives are undergoing. We expect to benefit from those efforts in order to improve the model and solve the difficulties described above.

6 Conclusions

SET is expected to gain wide acceptance as a secure Internet payment system since it combines the well-known credit card payment method with an elaborated security protocol. It is aimed at providing the necessary security through the authentication of the participants in a commercial transaction, as well as confidentiality of financial information. The fact that SET was developed by the major credit card companies is yet another factor contributing to its acceptance.

However, SET is a very complex and "heavy" protocol, and if from the cardholder's point of view it may be generally simple to understand and use, its complexity may prove it unsuitable for some computational environments. In this paper we discussed the usage of SET in mobile computing settings and argued that it is not practical to use in this kind of environments. The low bandwidth and high connection costs generally associated with mobile computing make it necessary to devise mechanisms that adapt better to those conditions, without loosing any of the benefits SET offers to the cardholders.

Based on these requirements, we proposed SET/A, a payment system based on the SET protocol and the mobile agent model. The purchase transaction, the most important part of SET from the cardholder's point of view, is implemented in SET/A almost as in SET. The cardholder's role is now played by an agent, which is sent to the merchant's server with the relevant information to perform the necessary operations. Since the cardholder doesn't need to keep the connection opened while the transaction is running, this solution contributes to lower cost and higher robustness.

We also discussed the need for small-size agents due to limitations on the bandwidth of the cardholder's device, and the possibility of providing the agent with enough information so that its role on the merchant's server is not limited to hand the data over to the merchant and collect the response. The requirement for mobility makes us pay special attention to the size of the agents we want to produce, in order to find a suitable compromise between the amount of data and code they need to do their work and the limitations imposed by the media through which the agents travel.

However, we are also interested in keeping the agent as intelligent and autonomous as possible, allowing it to take its own decisions (even if very simple) when needed. As part of our future work, we intend to use a mobile agent system to implement SET/A, with agents capable of negotiating with their hosts, based on the knowledge they carry as they migrate from one server to another. This implementation will primarily be developed in the context of electronic commerce, more specifically for *information brokers* and *contract negotiation* (in a project in which the authors are participating [8]). As a side result, we expect that the work with SET/A will produce or improve an agent system that can be used in a variety of environments, especially by travelling business people.

Security is another important issue, as the agent will need to take confidential information with it. The autonomy requirement implies that the agent must be able to have access to this data and act according to its contents, not just being a passive transporter. Even if the information is encrypted, the agent will need to be able to decrypt and use it, yet hiding it from others, which is very difficult when these actions are performed inside extraneous, potentially malicious servers. Thus, we are particularly interested in future advances in this field that can be incorporated into SET/A and, at the same time, willing to offer our own contribution.

References

1. 3COM CORPORATION. *PalmPilot.* http://www.3com.com/palm/.
2. D. CHESS, C. HARRISON, and A. KERSHENBAUM. *Mobile Agents: Are They a Good Idea?* In [13].
3. F. GRIFFEL, T. TU, M. MIRA DA SILVA, and M. MERZ. *Electronic Contract Negotiation as an Application Niche for Mobile Agents.* In Proceedings of the First International Enterprise Distributed Object Computing Workshop, Queensland, Australia, October 1997.
4. L. HURST. *MCK: Mobile Communication Kernel.* Dagstuhl Seminar on Mobile Software Agents, October 1997.
5. IRIDIUM LLC. *The IRIDIUM System.* http://www.iridium.com/systm/systm.html.
6. J. MONTGOMERY. *The Orbiting Internet: Fiber in the Sky.* Byte, Vol. 22, Number 11, November 1997.
7. NOKIA. *Nokia 9000 Communicator.* http://www.nokia.com/com9000/n9000.html.
8. COSMOS CONSORTIUM (PONTON, CEFRIEL, HAMBURG UNIVERSITY, INESC, INTERZONE MUSIC PUBLISHING, ORACLE UK, AND SIA). *COSMOS – Common Open Service Market for SMEs.* ESPRIT Research Project, 1997.
9. PSION. *Psion Series 5 Handheld Computer.* http://www.psion.co.uk/series5/.
10. K. ROTHERMEL, and R. POPESCU-ZELETIN (Eds.). *Mobile Agents.* Lecture Notes in Computer Science 1219, Springer, April 1997.
11. T. SANDER, and C. TSCHUDIN. *Towards Mobile Cryptography.* Technical Report TR-97-049, International Computer Science Institute, November 1997.
12. VISA INTERNATIONAL, and MASTERCARD INTERNATIONAL. *Secure Electronic Transaction (SET) Specification.* Version 1.0, May 1997.
13. J. VITEK, and C. TSCHUDIN (Eds.). *Mobile Object Systems – Towards the Programmable Internet.* Lecture Notes in Computer Science 1222, Springer, July 1996.
14. U. WILHELM, and X. DEFAGO. *Objets Protégés Cryptographiquement.* In Proceedings of RenPar'9, Lausanne, Switzerland, May 1997.
15. B. YEE. A Sanctuary for Mobile Agents. In Proceedings of the DARPA Workshop on Foundations for Secure Mobile Code, Monterey CA, USA, March 1997.