Distributed and Parallel Databases, 8, 85–117 (2000)

A Secure Agent-based Framework for Internet Trading in Mobile Computing Environments

XUN YI exyi@ntu.edu.sg
CHEE KHEONG SIEW
XIAO FENG WANG
EIJI OKAMOTO
ICIS, School of EEE, Nanyang Technological University, Nanyang Avenue, Singapore 639798

Recommended by: Vijay Atluri and Pierangela Samarati

Abstract. Most of the current Internet trading frameworks, in particular their negotiation and payment phases, are intended for customers frequently connected to the Internet during an entire transaction. This requirement cannot be easily met in the high communication cost and/or low bandwidth settings, typically found in mobile computing environments. Based on the software agent paradigm, a new secure agent-based framework for Internet trading in mobile computing environments is proposed in this paper. The framework is composed of two new protocols. One is the agent-based auction-like negotiation protocol, another is the agent-based payment protocol. Both of them are dedicated to solve the trade problems of Internet trading in mobile computing environments and ensured to be safe by cryptographic technologies. The combination of the two secure protocols constitutes an integrative solution for Internet trading in mobile computing environments.

Keywords: electronic commerce, mobile agent, auction-like negotiation, secure electronic transaction (SET), signcryption

1. Introduction

In recent years, the Internet are becoming an increasingly important channel for retail commerce as well as business to business transactions. Large mainstream companies are setting up shops on the Internet not merely to have a presence, but to actually make sales online. Some specialty operations exist that do all of their business on the Internet such as Amazon.com (http://www.amazon.com), and NECX Direct (http://www.necx.com), FirstAuction (http://www.firstauction.com) and etc. Various recent studies by analysts from Nielsen, Forrestor and IDC have shown that the number of web buyers, sellers and transactions are growing at rapid pace. The number of people buying on the Web is expected to increase from 18 million in December 1997 to 128 million in 2002, representing more than USD400 billion worth of commerce transactions.

Wireless networks have been known explosive growth over the last few years, reflecting a world in which it is important to be active regardless of location. As the Internet becomes more and more important for business transactions, it is natural to expect that wireless technology will 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 the

office is particularly attractive. In mobile computing environments it is desirable to have all the facilities usually found on the Internet, including the possibility of acquiring and paying for products and services.

The kind of mobility is usually based on portable devices with limited computing capacity and/or limited connectivity. For example, computing capable mobile phones (e.g. the Nokia 9000 Communicator [19]), PDAs (e.g. 3Com's PalmPilot [7]), Handheld PCs (e.g. Psion Series 5 [20]), up to notebooks, connected to the Internet through a modem attached to a cellular phone (or with an internal GSM modem). In view of the conditions under which mobile computing takes place, including low bandwidth, poor connectivity and the high cost of connect time, it becomes difficult and expensive to handle long, connected sessions, which require the transfer of large amounts of data.

Most of the current Internet trading framework, in particular their negotiation and payment phases, are intended for users frequently connected to the Internet. This requirement cannot be easily satisfied in mobile computing environments. For example, in an error-prone environment such as GSM or any other used for mobile communications, a user shopping on the Internet and trying to pay using SET compliant software [28] may experience several connectivity problems during the payment operation. Even with recovery mechanisms, it is easy to imagine how frustrating it can be for the customer 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 does succeed, its overall cost would probably have been too high.

Furthermore, the potential of the Internet for truly transforming commerce is largely unrealized to date. Electronic purchases are still largely non-automated. While information about different products and vendors is more easily accessible and orders and payments can be dealt with electronically, a human buyer is still responsible for collecting and interpreting information on merchants and products, making decisions on merchants and products and finally entering purchase and payment information.

Software agent technologies offer a new paradigm for trading on the Internet. It can be used to automate several of the most time consuming stages of the buying process. Unlike "traditional" software, software agents are personalized, continuously running and semi-autonomous [16]. As a mobile, flexible and autonomous small program unit, mobile agents can roam a network, collect and analyze the information from servers on the Internet, negotiate with servers, make decisions where to buy and even make automated payments on behalf of customers. These qualities are conducive for optimizing the whole buying experience and revolutionizing commerce as we know it today [18]. However, customers are wary about employing agents to trade on behalf of them, largely because of concerns about unknown risks they may face. The key to alleviating many of these concerns—to mitigating the risks—is security issue of agents.

In order to run, a mobile agent has to expose its code and data to the host environment which supplies the means for it to run. Therefore, the host is easy to decompose agent code, scan his data, tamper and even kill the agent. The confidential data in a mobile agent, such as negotiation strategies and credit card information, cannot keep secret to the host.

So far, one of the best solutions to agent-based negotiation for goods sale is through auction. The most attracting character of auction is its open and simple framework. It is

unnecessary for a negotiation agent to keep its negotiation strategies secret to a merchant host. The research on an agent-based auction-like negotiation protocol can be found in reference [36]. This protocol allows negotiation agents dynamically to decide their route across the Internet in merchant hosts. Therefore, a merchant host may alter, to his advantage, the next stop on the agent traveling agenda. In this paper, by ordaining the list of merchants in negotiation agents, we propose another new secure agent-based auction-like negotiation protocol for Internet trading in mobile environments, which has the following particular features: (1) negotiation for agent-based trading is performed through a novel pattern of electronic auction. (2) negotiation results in merchant servers are ensured to be valid with their signatures. (3) malicious behaviors can be detected and the breeder can be dug out by the help of sociological factors.

SET/A [22], guided by the SET rules and based on the mobile agent paradigm, has recently been developed to meet the requirements of Internet payment in mobile computing environments. In order to protect agent's confidential data (i.e., credit card information) against the potentially malicious merchants, SET/A has to rely on a secure agent execution environment, such as in a tamper-proof environment [31] or a secure coprocessor [33], located at merchant servers. In our opinion, this solution is high cost for merchants and the required security is not easy to ensure. In this paper, we propose another secure agent-based payment protocol, namely SET/A+, which can remove the limitation of the security of the agent's execution environment at the merchant server by adding a trust verification center in the payment system for mobile computing. SET/A+ is able to ensure the same level of security as SET, providing an alternative means of online payment using the SET protocol.

In addition, considering the characteristics of payment agents, an effective signcryption scheme binding encryption to digital signature is proposed as the underlying signature and encryption schemes of the SET/A+ protocol. By this signcryption, the private signature key is not only used to sign, but also to specify a symmetric key by which a message is encrypted.

By combining the above protocols, we finally propose an integrative solution for agent-based Internet trading in mobile computing environments.

The remainder of this paper is organized as follows. Section 2 introduces background knowledge necessary to describe a new secure agent-based framework. Section 3 presents a new agent-based auction-like negotiation protocol. Section 4 proposes a new agent-based payment protocol. Section 5 combines the agent-based auction-like negotiation protocol and the agent-based payment protocol to constitute an integrative agent-based framework for Internet trading in mobile computing environments. Security issues and performance of the proposed framework are analyzed in Section 6 and Section 7 respectively; Conclusions are drawn in the last section.

2. Background knowledge

In this section, we introduce the background knowledge necessary to describe a secure agent-based framework for Internet trading in mobile computing environments.

2.1. Difficulties of internet trading in mobile computing environments

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

1. Low bandwidth and poor connectivity—terrestrial wireless network protocols like GSM or satellite-based systems like IRIDIUM [12], typically offer bandwidths in the range of 2,400 bps to 9,600 bps (although there are claims for much larger bandwidths with broadband satellite systems in the future [17]). 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.
2. 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 online 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. What is needed is some kind of asynchronous mode of operation, in which the customer can send the purchase 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 in the SET protocol 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. negotiation strategy and account number) and letting it be used by a remote server in an unpredictable way.

This means that it is necessary to reduce the customer's role to two steps, the initial purchase 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 customer 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.

2.2. Mobile agents

Software agents are probably one of the fastest growing areas of information technology. They are being used, and touted for applications as diverse as personalized information management, electronic commerce, computer games and etc. An agent can be thought of

as a computer program that simulates a human relationship by doing something that another person could do for you [26]. An agent is a self-contained program capable of controlling its own decision making and acting, based on its perception of its environment, in pursuit of one or more objectives [14].

More than one type of agent is possible. In its simplest form it is a software object that sifts through large amounts of data and present a subset of this data as useful information to another agent, user or system. An example of this is an agent that read and analyze all incoming e-mail, and route it to an appropriate department or another agent for reply [32]. These types of agents are called static agents.

Mobile agents, owned by a user or another software element, are capable of migrating from one computer to another to execute a set of tasks on behalf of its owner. The agents would typically gather and analyze data from a multitude of nodes on the network, and present a subset of this data as information to a user, agent or system. Mobile agents are said to be autonomous, in the sense that they can make 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 some part 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 [23, 29]. However, the number of applications using this technology is still small. Electronic commerce is generally seen as one of the most promising application fields for mobile agents. For example, a buying agent leaves its host with the mission of querying several vendors about a certain product, determines which one offers the lowest price (or some other kind of preferred feature), buys the product from that one and pays for it. Clearly there is a perception that agents are suitable for this kind of activity and that the ability to pay is one of the desired properties they should have. A major concern is always how to do this in a secure way, in particular without revealing confidential information to the outside world.

The mobile agent security can be split into two broad areas [5, 6]. The first involves the protection of host nodes from destructive mobile agents while the second involves the protection of mobile agents from destructive hosts. Telescript [30] has been developed to allow the safe interaction between mobile agents and the host, to control access to system supplied resources and to provide authentication facilities, communication privacy and system level authorization. Both Java [9] and Safe-Tcl [3] also provide similar security features. These approaches can effectively protect host nodes against destructive mobile agents, but appear forceless in preventing mobile agents from malicious hosts because in order to run, a mobile agent has to expose its data and code to the host environment which supplies the means for him to run. Therefore, the host is easy to decompose agent code, scan his data, tamper and even kill the agent.

In order for a mobile agent to trade on behalf of human, there are a few requirements that the agent must fulfill:

1. Small-sized—since we're assuming low and/or expensive bandwidth, we have to mini mize 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 [11]. Clearly, the agent should be as small as possible for better performance and smaller costs.
2. Survive inside hostile execution environments—the customer 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.
3. Hide confidential information—one of the main purposes of SET is to offer an appropriate level of security, so that the customer can be sure that none of the confidential data (card number, expiry date, etc.) is disclosed to any unauthorized party.

In this paper, we will examine mobile agents that have the ability to buy goods on behalf of its owner.

2.3. Agent-based auction-like negotiation

Negotiation is an important part of Internet trading. In their survey paper on automatic negotiation, Beam and Segev [1] define automatic negotiation as the process by which two or more parties multilaterally bargain resources for mutual intended gain, using the tools and techniques of electronic commerce.

So far, one of the best solutions to automatic negotiation for goods sale is through auction. Auctions are usually used in the cases when the auctioneer wants to sell an item and get the highest possible payment for it while the bidders want to acquire the items at the lowest possible price. The most attracting character of auction is its open and simple framework. The research on electronic negotiation through Internet-based auction can be found in reference [2]. Up to now, some electronic auction houses are already established on the web, such as, Onsale (http://www.onsale.com), FirstAuction (http://www.firstauction.com), ZAuction (http://www.zauction.com), Dealdeal (http://www.dealdeal.com) and Ubid (http://www.ubid.com). They post on-sale goods on the web page every day, customers can bid for them via the web browsers freely.

Although there is an "auction fever" which tends to take auction as a panacea for shopping and selling, a closer look at its characteristics, however, reveals its hostility towards retail commerce. Guttman and Maes [10] pointed out problems with current online auctions such as "winner's curse" and low performances.

Nowadays, consumers are much more in the driver's seat in the online market than in the physical-world market. This is largely due to the dramatic reduction of search costs. This increases the competition among retailers and forces them to positively differentiate themselves in value dimensions other than price. However, instead of merchants competing for consumer patronage, traditional online retail auctions force consumers to compete with one another for a specific merchant offering. This brings in the "winner's curse", which pushes up the winning bid above the product's market valuation. On the other hand, retailers often care less about profit on any given transaction and care more about long-term profitability. Customers' loss through "winner's curse" actually destroys

the relationship between retailers and customers, thus damaging the retailers' benefits in the long run.

Furthermore, the traditional online auctions have long delay between starting auctions and purchasing the product. This retards a large number of impatient or time-constrained consumers. In fact, the English and Yankee auction protocols are usually implemented over the Internet for several days. Since only the highest bidder of an auction can purchase the auctioned goods, the rest of the bidders (the majority) have to endure the long fruitless delays. Unlike the vendue of works of art, this is quite annoying in retail commerce

The development of mobile agents may provide some opportunities to improve auction performance. Chavez and Maes suggested dispatching agents to a centralized salesroom to conduct the auction locally [4]. This causes security concerns. In order for an agent to run, it must expose its data and code to the host resources. Therefore, if the auctioneer conspires with the owner of the salesroom, the auctioneer can manipulate the auction to his advantage. If the mobile agent's negotiation strategy is known to the host, it may be at a significant disadvantage. Suppose the merchant's host knows that the buyer's negotiation strategy is to accept all offers under a certain (unknown) threshold value. The merchant can begin at a price greater than the threshold value and repeatedly offer the buyer a penny less each time, until the buyer's threshold value is reached, at which point the (worst possible, for the buyer) deal is made. This results in the bidding agents to suffer losses.

Considering the difficulty for a negotiation agent to hide negotiation strategies and the open and simple framework of auction, an agent-based auction-like negotiation protocol for Internet trading has been proposed in [36]. Different from traditional auction models, a negotiation agent of this protocol acts as a mobile auctioneer while online retailers bid for low price. It is strategically analogous to English auction. This in fact eliminates the "winner curse". In order to combine information gathering and negotiation process together, this protocol allows the negotiation agent dynamically to decide their route across the Internet in merchant hosts. Therefore, a merchant host may alter, to his advantage, the next stop on the agent traveling agenda or a few merchant hosts collude so that the negotiation agent only negotiate among them.

In this paper, by restricting the negotiation agent always to roam in a specified list of merchant servers, we will propose another new secure agent-based auction-like negotiation protocol for Internet trading in mobile environments.

2.4. Agent-based payment protocol

In this section, we firstly describe the SET protocol and then an agent-based payment protocol for mobile computing environments.

2.4.1. The SET protocol. Almost every Internet user has heard of credit card fraud, performed by hackers eavesdropping on connections used to send the 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 on the server, where it can easily be read by anyone with access to it (even if not authorized).

The concern about protecting the user's credit card information lead VISA and MasterCard, in association with major software and cryptography companies, to the development of the SET protocol [28]. The SET protocol is composed of several kinds of transactions, ranging from cardholder registration, merchant registration, to purchase requests, to payment authorization and payment capture. The participants of these phases are as follows:

1. Cardholder—a customer using a payment card that has been issued by a certificate authority.
2. Issuer—a financial institution that establishes an account for a cardholder and issues the payment card. The Issuer guarantees payment for authorized transactions using the payment card in accordance with the payment card brand regulations and local legislation.
3. Merchant—a merchant offering goods for sale or providing services in exchange for payment. A merchant that accepts payment cards must have a relationship with an Acquirer.
4. Acquirer—the financial institution that establishes an account with a merchant and processes payment card authorization and payments.
5. Payment gateway—a device operated by an Acquirer or a designated third party that processes merchant payment messages, including payment instruction from cardholders.

On the assumption that cardholders and merchants have registered and obtained their certificates from the Issuer. The purchase request phase can be depicted in figure 1.

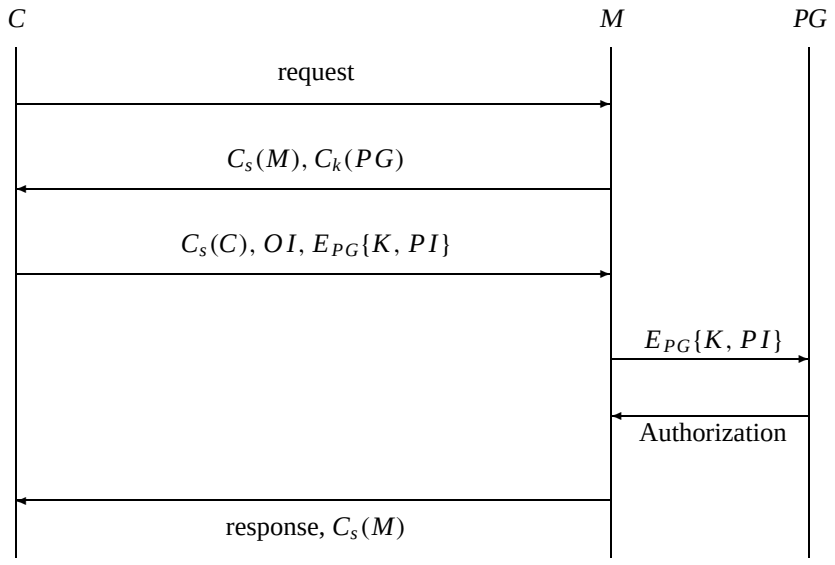

Figure 1. SET purchase request transaction.

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 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.

The procedure of purchase request transaction (shown in figure 1) is described as follows:

Step 1. A cardholder (C), looks at a catalog (printed in paper, supplied in a CD-ROM or available online on the Web) provided by a 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 $C_s(M)$, and the key-exchange certificate $C_k(PG)$ of a payment gateway (PG). The merchant also sends a unique identifier, assigned to this transaction.

Step 3. The cardholder (i.e., his or her software) verifies the certificates by traversing the trust chain to the root key (the public signature key of a certificate authority (CA)) so as to 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 (i.e., $X_K[PI]$), and the key (K) used to encrypt it are encrypted into a digital envelope, denoted as $E_{PG}(K, PI)$ ($=X_{PG}[K] \parallel X_K[PI]$, where "$\parallel$" represents the concatenation of two data blocks), using the payment gateway's public key. 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 CA_s, asking payment gateways for payment authorizations, and so on.

SET provides important properties like authentication of the participants, non-repudiation, data integrity and confidentiality. Each player knows only what is strictly necessary for his or her role.

2.4.2. The SET/A purchase request. SET/A [22], guided by the SET rules and based on the mobile agent paradigm, has recently been developed to meet the requirements of Internet payment in mobile computing environments. The cardholder registration, merchant registration, payment authorization and payment capture of SET/A are the same as SET. The only change is in the purchase request transaction which can be depicted in figure 2.

The process of SET/A purchase request transaction is described as follows:

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

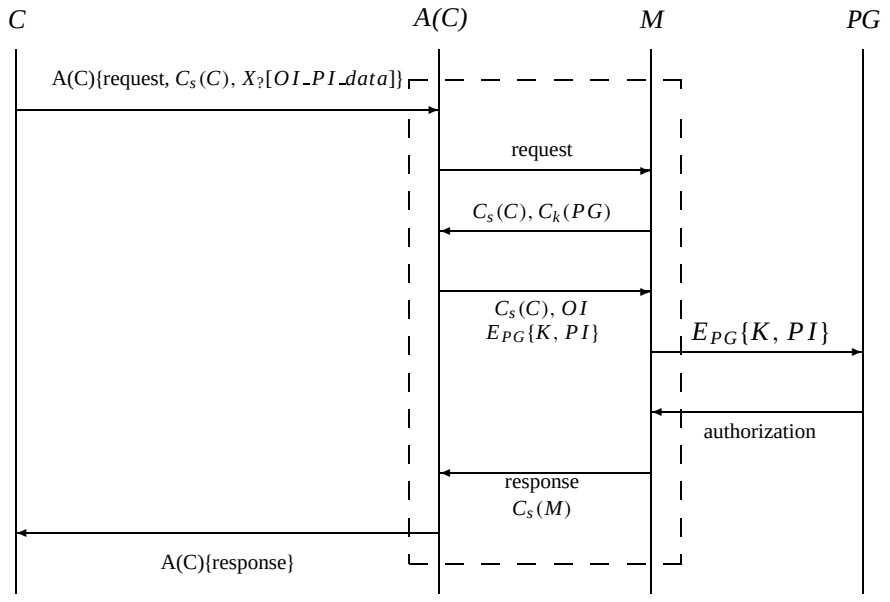

Figure 2. SET/A purchase request transaction.

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 SET/A have not given a detailed proposal to deal with it and just assume it was encrypted with someone's key ($X_?[OI\_PI\_data]$). To our understanding, $X_?$ should be the public key of the secure agent execution environment located at a merchant server.

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.

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 (i.e., $X_K[PI]$), 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 2.4.1), 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.

Protecting confidential information, such as credit card information, in an agent from hostile environments is a major research issue in mobile agent security. The SET/A protocol suggests running the agent in a tamper-proof environment [31] or a secure coprocessor [33], to protect the agent against malicious merchants. That means that 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.

Another approach suitable for SET/A to protect the agent—"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, for example, taking some initial steps to execute hidden computations [24, 25].

However, for SET/A, whether running the agent in a tamper-proof environment or a secure coprocessor or taking some initial steps to execute hidden computations needs to know the public key of the secure agent execution environment of a merchant in advance so that the agent can safely bring the confidential information, such as credit card information, into it. If a cardholder gets the public key by sending a request to the merchant and receiving it from a reply, SET/A almost has not reduced the SET computational burden on the cardholder's side.

2.5. Signcryption

Signcryption is a new paradigm in public key cryptography. A remarkable property of a signcryption scheme is that it fulfills both the functions of public key encryption and digital signature, with a cost significantly smaller than that required by signature-then-encryption. The explanation how to achieve the cost (signature & encryption) much less than the cost(signature) + cost(encryption) can be found in reference [37].

Clearly, mobile agents should be as small as possible for better performance and smaller costs. In view of it, a new signcryption protocol is proposed here for a participant of our framework to sign a message and distribute a symmetric key with his private signature key. The procedure of signing a message can be described as follows:

1. As the Digital Signature Standard (DSS) [27], the signcryption protocol chooses three parameters (p, q, g), where p is a large prime, q is a large prime factor of $p-1$, $g = h^{(p-1)/q}(mod\ p)$ with h being an integer satisfying $1 < h < p-1$ and $h^{(p-1)/q}(mod\ p) > 1$.
2. Each participant in our framework is required to generate a pair of signature public-secret keys. The pair of signature public-secret keys of an entity A is (y_A, x_A), where $y_A = g^{x_A}$ (mod p) and x_A is a secret key chosen randomly from $GF(q)^*$.
3. The message (M) required to sign is hashed with an one-way 2n-bit hash function (H) which is based on a n-bit block cipher with 2n-bit key (such as IDEA [15] and the encryption algorithm in [34]) and shown in figure 3. In figure 3, M_i, G_i and H_i are n-bit integers; E denotes a n-bit block cipher using 2n-bit key; $\oplus$ presents bitwise exclusive-or of n-bit blocks while $\boxplus$ indicates addition modulo 2^n of n-bit integers. The detail hash process can be found in literature [35].
4. The digital signature of an entity A on a message (M) is composed of two numbers s and t which are defined as

$$s = (g^r(mod\ p))(mod\ q) \tag{1}$$

$$t = -s \cdot x_A - H(M) \cdot r(mod\ q) \tag{2}$$

where r is a random number chosen from $GF(q)^*$.

Given (M^*, s^*, t^*), one can verify whether (s^*, t^*) is indeed a genuine signature of the

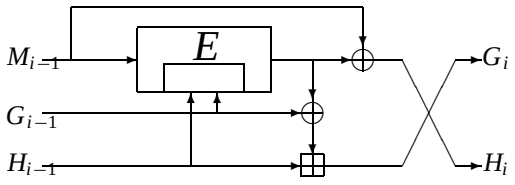

Figure 3. Computational graph for the hash round function h.

entity A on M^* only by checking the following equation:

$$g^{t^*} \cdot y_A^{s^*} \cdot s^{*H(M^*)} (mod\ p) = 1 \tag{3}$$

One can accept the digital signature of the entity A on the message M^* if the above equation holds. This conclusion is based on the following theorem.

Theorem 1. *If s and t are computed as formulae* (1) *and* (2) *respectively, then Eq.* (3) *holds.*

Proof: From Eq. (2), we can deduce

$$\begin{aligned} g^t &= g^{(-s \cdot x_A - H(M) \cdot r)(mod\ q)} (mod\ p) \\ &= y_A^{-s} \cdot s^{-H(M)} (mod\ p) \end{aligned}$$

After moving the right term to the left, we can obtain the Eq. (3). □

The above signature scheme is used for the certificate authority (CA) to issue a signature certificate for the entity A in our framework as follows:

The certificate authority (CA) adopts the certificate format of X.509 [13] to construct the signature certificate message M_A which may contain such information as certificate serial number, validity period, the public key (y_A) of the entity A, the public key (y_{CA}) of CA, etc., and then sign M_A in accordance with the above signature scheme.

When the entity A sends its signature on a message (M), i.e., (s, t), to the entity B, he can not provide the plaintext of M, but encrypt the message M with a block cipher (E) (which is the same as the underlying block cipher in the hash function shown in figure 3) and provide the entity B with the ciphertext of M, i.e., $E_k(M)$, where k is randomly generated and encapsulated in the following form:

$$z = y_B^{(-r \cdot s - x_A \cdot t)(mod\ q)} \cdot k (mod\ p) \tag{4}$$

where (r, s, t) is the same as that in Eqs. (1) and (2).

After the entity B receives (s, t, z) from the entity A, it can retrieve k from z on basis of the following theorem.

Theorem 2. *If z is computed as formula* (4)*, then the following equation holds*:

$$s^{x_B \cdot s} \cdot y_A^{x_B \cdot t} \cdot z = k (mod\ p) \tag{5}$$

Proof:

$$\begin{aligned} & s^{x_B \cdot s} \cdot y_A^{x_B \cdot t} \cdot z \\ &= g^{r \cdot x_B \cdot s} \cdot g^{x_A \cdot x_B \cdot t} \cdot z \\ &= y_B^{(r \cdot s + x_A \cdot t)} \cdot z \\ &= k (mod\ p) \end{aligned}$$

The Eq. (5) means that the entity B can retrieve k if (s, t, z) and the entity A's certificate are given. With the secret-key k of the block cipher E, the entity B can decrypt the encrypted message, then hash the message and verify the signature of entity A on the message M according to the Eq. (3). □

3. Description of the proposed agent-based auction-like negotiation protocol

In this section, we will firstly deal with the structure of mobile negotiation agents and the process of auction-like negotiation.

3.1. Structure of mobile negotiation agent

In our framework, a mobile negotiation agent is defined as a software element (program, procedure, object, etc.), owned by a buyer who gets access to Internet by mobile devices, capable of migrating from one computer to another to execute negotiation task on behalf of its owner. The basic structure of a mobile negotiation agent can be divided into seven distinct sections as follows:

1. Descriptor—the illustration of the agent's structure and format, the transaction identifier, encryption algorithm identifier, hash function identifier, signature algorithm identifier and etc.
2. Certificate—the certificate of the agent's owner.
3. Code—the program executed in an agent execution environment to fulfill negotiation mission.
4. Data—the data including a list of merchants which the agent will visit, some possible error actions (the actions the agent should take should an error occur while the agent runs on the environment supplied by a server. Some possible error actions include discarding the agent, delivering an error notification to a specified address, routing the agent to a server) and etc.
5. Time stamp—the time when the agent launches from the buyer.
6. Signature—the signature of the buyer on the previous five sections. The buyer signs the agent in accordance with Eqs. (1) and (2) in Section 2.5. The signature can be verified by Eq. (3).
7. State—the section for merchants to post their bids and signatures.

The structure of a mobile negotiation agent can be illustrated in figure 4.

Note that the representation of the code and data is outside of the scope of this paper. Here we focus on the negotiation process.

3.2. Generation of mobile negotiation agent

When a buyer wants to purchase something, he firstly looks at a catalog (printed in paper, supplied in a CD-ROM or available online on the Web) provided by merchants and chooses

Descriptor	Certificate	Code	Data	Time Stamp	Signature	State

Figure 4. The structure of a mobile negotiation agent.

a list of merchants with whom the buyer wants to negotiate to make a good deal. Then he creates a mobile negotiation agent in accordance with the structure as shown in figure 4. Finally the agent is motivated and migrates to the nearest merchant server on Internet according to the route specified in the data section of the agent.

3.3. Verification of mobile negotiation agent

When the mobile negotiation agent enters a merchant server first time, the server will automatically perform the following common verification procedure:

1. Verifying the agent certificate with the signature public key of the certificate authority (i.e., y_{CA}), supposed to be known by each participate of our framework.
2. Checking the validity of Time Stamp in the agent and verifying the signature of the buyer on the agent with the signature public key of the buyer (i.e., y_B) which is included in the certificate of the agent.
3. Verifying the previous merchant's certificate and signature on the state of the agent in the above same way.

If without problem, the merchant server supplies an agent execution environment for the agent to run its code. Otherwise, it must ask the previous server repeatedly to send the agent until it receives the correct agent or deliver a error notification to an emergency server.

When the mobile negotiation agent re-enters a merchant server, the server do not need to perform the verification procedure (1) and (2), but only (3).

In addition, merchant servers always need to keep the states of the passing agent and the previous merchant's signatures as non-repudiation evidences for a short period. Of course, merchant servers can automatically clean the data after the period.

3.4. Agent-based auction-like negotiation process

Our agent-based auction-like negotiation protocol for Internet trading in mobile computing environments is similar to the English Auction and the common values model (bidder's valuation is influenced by other's). In this protocol, the flow of the negotiation agent among the list of merchant servers $S^1, S^2, \ldots, S^n$ can be illustrated in figure 5.

In figure 5, "A" denotes the negotiation agent. In addition, we use symbol "Bid^k_{Sj}" to represent the bid offered by S^j during the k round bid of auction.

The negotiation process is described as follows:

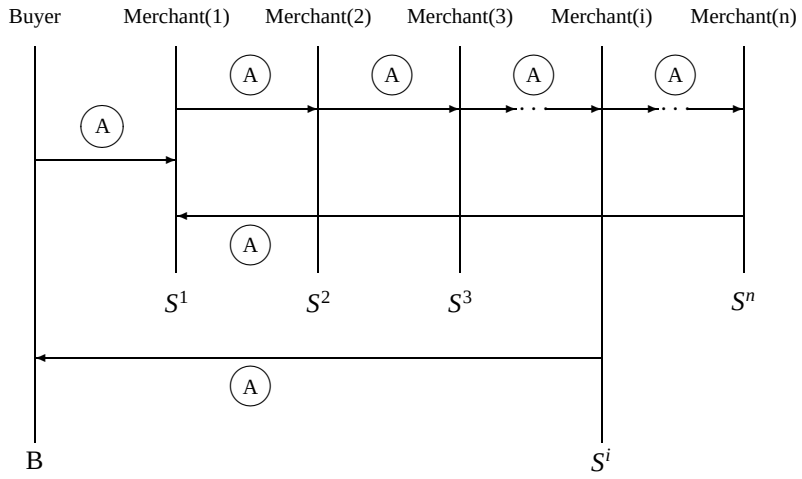

Figure 5. Agent-based auction-like negotiation protocol.

Generation of Initial Minimum Bid and Minimum Bid Decrement: The initial minimum bid (MB_0) and minimum bid decrement (*MBD*) are two important parameters for auction-like negotiation. They can be assigned by two ways: one is by the buyer, another is by the first bidder who wants to sell the goods. According to conventions, our negotiation protocol adopts the first way to assign *MBD* and MB_0 (put into the data of the agent).

First Round Bid of Auction: In general, we suppose the negotiation agent A having fulfilled its task in S^{i-1} and roaming into S^i.

The merchant server S^i firstly performs the verification procedure as Section 3.3. If without problem, it supplies an agent execution environment for the agent to run.

When the agent run its code, it automatically tells S^i (1) what goods it wants to buy and (2) the current lowest bid Bid_L. Then it asks S^i to bid, i.e., at what price the merchant can sell the goods. If S^1 offers its bid $Bid^1_{S^i}$, the negotiation agent will create a negotiation result $NR^1_{S^i}$ including:

- The transaction identifier found in the descriptor of the negotiation agent;
- The round number of auction which is 1 in this case;
- $Bid^1_{S^i}$ which should be less than Bid_L (at least one minimum bid decrement);
- The current time which acts as the time stamp.

If S^i wants to waive bidding, the place of the bid in the above contents will be a waiving declaration, indicating at which price it gives up.

Afterward, the agent requires the merchant S^i to sign $NR^1_{S^i}$ (the signature is denoted as $Sign(NR^1_{S^i})$) and then updates its state by attaching (1) S^i's certificate, (2) $NR^1_{S^i}$ and (3) $Sign(NR^1_{S^i})$ to the previous state.

Previous State	S^i's Certificate $NR^1_{S^i}$ $Sign(NR^1_{S^i})$	S^i's Signature on State

Figure 6. The state of agent in the merchant server S^i.

Descriptor	S^1's Certificate $NR^1_{S^1}$ $Sign(NR^1_{S^1})$	S^2's Certificate $NR^1_{S^2}$ $Sign(NR^1_{S^2})$	$\cdots$	S^n's Certificate $NR^1_{S^n}$ $Sign(NR^1_{S^n})$	S^n's Signature on State

Figure 7. The first round negotiation results.

Furthermore, the agent demands S^i to sign the new state of the agent and attaches the signature behind the state (shown in figure 6).

Finally, the agent duplicates itself, keep one copy in the server, and migrates another to some merchant server in the list of merchant servers which have not been visited in the first round bid of auction.

In this way, the negotiation agent roams one by one through the whole list of merchant servers. The last merchant server S^n in the first round bid of auction sends a series of negotiation results of this round bid (shown in figure 7) back to the first merchant server S^1. At this time, the first round bid of auction terminates.

The kth Round Bid of Auction ($k = 2, 3, \ldots$): The k-th round bid of auction is almost same as the first round. The changes lie in:

1. The negotiation agent never re-visit the merchant servers which have waived bidding.
2. The message transferred among merchant servers is not the whole body of the negotiation agent, but its state with its descriptor identifying the agent (as shown in figure 7).
3. Instead of attaching in the state of the negotiation agent like the first round, the negotiation result $NR^k_{S^i}$ in the merchant server S^i only replace the last round negotiation result $NR^{k-1}_{S^i}$.

On basis of the above discussion, we can obtain the following conclusion:

Theorem 3. *Assume $Bid^k_{S^{m_1}}, Bid^k_{S^{m_2}}, Bid^{k_1}_{S^m}, Bid^{k_2}_{S^m}$ be significant, then the following inequalities hold.*

$$Bid^k_{S^{m_2}} < Bid^k_{S^{m_1}} \quad \text{if } m_2 > m_1 \tag{6}$$

$$Bid^{k_2}_{S^m} < Bid^{k_1}_{S^m} \quad \text{if } k_2 > k_1 \tag{7}$$

With the increase of the round number of auction, the lowest bid (Bid_L) will become less and less. Because the initial minimum bid (MB_0) is limited and the minimum bid

decrement (*MBD*) is greater than 0, this auction cannot repeat forever. There is always a bidder winning the auction after a few rounds, i.e., except from one bidder, all the others declare to waive bidding. In figure 5, we suppose the winner is the merchant server S^i. Therefore, the negotiation agent residing in S^i will send the final negotiation results like figure 7 to the buyer.

3.5. Negotiation result check

When the buyer gets access to Internet again, he can receive the negotiation results by using a mechanism similar the one used by cellular phone operators to deliver SMS messages when the user re-connects.

From the negotiation results collected by the mobile negotiation agent, the buyer B can firstly verify each merchant's certificate and then its signature on the relevant negotiation result. If except from one, all the other merchants declare to waive bidding in some round auction, the auction-like negotiation results can be regarded as reliable. The agent-based auction-like negotiation is successfully fulfilled in the time.

In the following procedure, the buyer may create a payment agent and migrate it to the winning merchant to perform the payment task.

4. Description of the SET/A+ protocol

In Section 2.4.2, we introduced an agent-based payment system—SET/A. We also point out that SET/A depends on a secure execution environment or hidden computation on the merchant's server to protect an agent's confidential data (i.e., credit card information) against a malicious merchant. In our opinion, the solution is high cost for merchants and the required security is not easy to ensure.

In this section, we propose another agent-based payment system for mobile computing on Internet, namely SET/A+, which removes the limitation of the security of the agent's execution environment on merchant's server by adding a trust verification center in the payment system for mobile computing.

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.

4.1. Structure of payment agent in the SET/A+ protocol

In the SET/A+ protocol, the structure of a payment agent is almost same as that of a negotiation agent shown in figure 4. The differences between them lie in the code, data and state. The code in the payment agent dedicates to fulfill certain payment mission. The data in the payment agent will include the order information, the payment information described in Section 2.4.1 and etc. The state of figure 4 is put together with code in SET/A+.

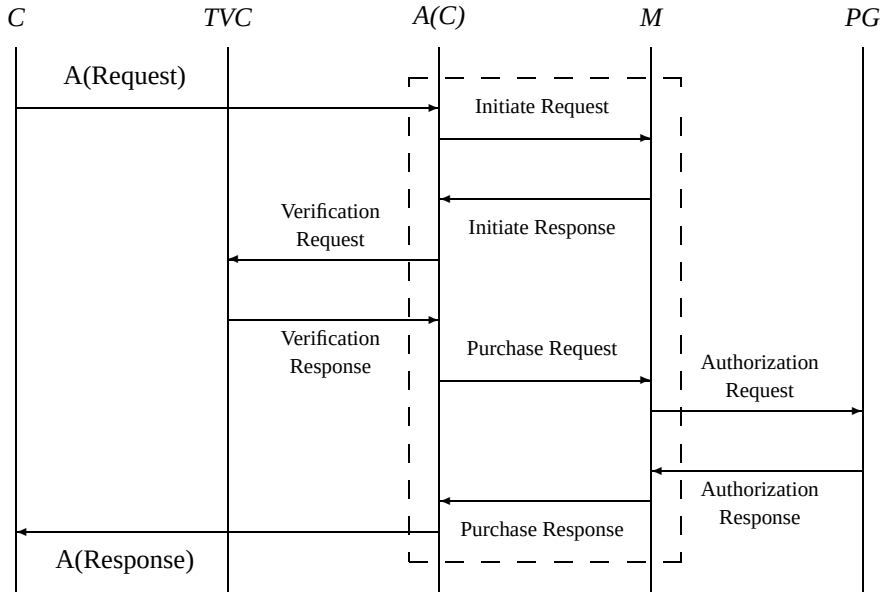

Figure 8. SET/A+ purchase request transaction.

4.2. Purchase request of the SET/A+ protocol

A purchase request under SET/A+ has a few more steps than in SET, since a payment agent has to be generated in a cardholder and to be sent to the merchant's server and return to the cardholder when the transaction is done. The process of a SET/A+ purchase request can be depicted in figure 8.

Different from SET/A, we introduce a trust verification center (TVC) in our proposal. The TVC will charge cardholders or merchants for providing verification service. Therefore, it is reasonable for the TVC to be added in the payment system. In addition, we assume that public signature key of the TVC, i.e., $y_{TVC}(= g^{x_{TVC}})$ is known to each cardholder.

Under the above assumptions, the procedure of the SET/A+ protocol can be described as follows:

Generation of Payment Agent: As SET/A, a cardholder (C), inspects a catalog (printed in paper, supplied on a CD-ROM or available online on the Web) of a merchant (M), after deciding to purchase something, creates a payment agent in accordance with figure 4, i.e.,

1. The cardholder's signature certificate ($C_s(C)$) is put into the certificate portion of the payment agent.
2. The code and state portion are filled by program. The description of this program is out of the range of this paper.

3. The order information (OI), the encrypted payment instructions (PI), the digest of PI (i.e., H(PI)), the information for the TVC, the dual signature of the cardholder on OI and PI, and etc., are put into the data portion respectively.

   - The PI is encrypted with a randomly generated symmetric key k, i.e. $E_k(PI)$, where the bit length of k must be shorter than that of the prime p.
   - Both OI and PI contain the unique transaction identifier I_C assigned by the cardholder C.
   - The information for the trust verification center contains:

$$s = (g^r (mod\ p))(mod\ q) \tag{8}$$

$$t = -s \cdot x_C - H(M_{TVC}) \cdot r (mod\ q) \tag{9}$$

$$z = y_{TVC}^{(-r \cdot s - x_C \cdot t)(mod\ q)} \cdot (k + I_C \cdot T + R)(mod\ p) \tag{10}$$

$$w = E_{(k+I_C \cdot T+R)(mod 2^l)}(M_{TVC}) \tag{11}$$

   where r is a random number chosen from $GF(q)^*$, x_C is the private signature key of the cardholder, y_{TVC} is the public signature key of the trust verification center, M_{TVC} is the message for the TVC, R is a random number chosen from $GF(p)^*$, l is the key bit length of the block cipher (E) and T is the current time.

   - It should be noted that R is also put into the date portion of the payment agent. M_{TVC} contains the transaction identifier I_C, the name of the merchant host from which the cardholder wants to order goods and the time stamp T.
   - The dual signature of the cardholder on OI and PI is the signature of the cardholder on the message $H[H(OI) \parallel H(PI)]$. The dual signature is denoted as $Sign_C[H(H(OI) \parallel H(PI)]$.

4. The current time T is put into the time stamp portion.
5. The signature of the cardholder on the message from descriptor to time stamp in the payment agent is put into the signature portion.

Finally, a payment agent is sent to a merchant's server.

Authenticating Payment Agent: After arriving at the merchant's server, the agent hands in its signature certificate and signature on the agent so that the merchant's server can verify the certificate and the signature with Eq. (3) by using:

- The public signature key of the Issuer (y_{CA}) which is known to each participant in the payment system;
- The public signature key of the cardholder (y_C) which is known from the cardholder's signature certificate.

If no problem, the merchant's server supplies an agent execution environment for the agent to run.

Initial Request: The agent resides in the environment and sends (now locally) an initiate request to the merchant M. The message indicates which payment card brand will be used for

the transaction and which trust verification center will be used for certificate and signature verifications (i.e., $C_s(TVC)$) and requests a copy of the gateway's certificate.

Initial Response: When the merchant receives the request, it assigns a unique transaction identifier I_M to the message. It then provides the agent with an initial response, which ranges over the merchant signature certificate ($C_s(M)$) and payment gateway key-exchange certificate ($C_k(PG)$) that correspond to the payment card brand indicated by the agent and the transaction identifier I_M. The initial response takes the signcryption form as formulae (1), (2), (4) and $E_k(*)$.

Verification Request: The agent contacts the trust verification center by transmitting a verification request, which is composed of $C_s(C)$, $C_s(M)$, the information for the TVC, i.e., (s, t, z, w) shown in (8)–(11), and the initial response from the merchant.

After verifying $C_s(C)$, the TVC retrieves the symmetric key in the same way as Theorem 2, i.e.,

$$\begin{aligned} & s^{x_{TVC}\cdot s} \cdot y_C^{x_{TVC}\cdot t} \cdot z \\ & \quad = g^{r\cdot x_{TVC}\cdot s} \cdot g^{x_C\cdot x_{TVC}\cdot t} \cdot z \\ & \quad = y_{TVC}^{(r\cdot s + x_C\cdot t)} \cdot z \\ & \quad = (k + I_C \cdot T + R)(mod\ p) \end{aligned}$$

Then the TVC decrypts w with the secret key $(k + I_C \cdot t + R)(mod\ 2^l)$ to obtain M_{TVC}, verifies the signature of the cardholder on M_{TVC} and keeps (M_{TVC}, s, t) as a charge evidence.

In the way, the TVC retrieves $C_k(PG)$ and I_M from the initial response of the merchant after verifying $C_s(M)$ and the signature of the merchant on the initial response.

Finally, after the TVC verifies $C_k(PG)$, he replies the agent with a confirmation which includes:

$$PG_p((k + I_C \cdot T + R) + I_M \cdot T)$$

where PG_p is the payment gateway's public key-exchange key obtained from the key-exchange certificate $C_k(PG)$.

Purchase Request: After receiving the above confirmation, the agent creates the digital envelope E_{PG} for the payment gateway, containing the following items:

$$E_{PG} = \{PG_p(k + I_C \cdot T + R + I_M \cdot T),\ I_C,\ I_M,\ T,\ R,\ E_k(PI)\} \tag{12}$$

The purchase request including

$$C_s(C),\ OI,\ H(PI),\ Sign_C[H(H(OI) \parallel H(PI))],\ E_{PG}.$$

is then sent to the merchant.

Authorization Request: After checking the certificate $C_s(C)$, in order to ensure that the order has not been tampered with in transit and that it was signed using the cardholder private signature key, the merchant verifies the dual signature of the cardholder in the following way:

- Because OI is known to the merchant, he can compute the digest of OI, i.e., H(OI);
- Because $H(PI)$ is known to the merchant, he can calculate the digest of $H(OI) \parallel H(PI)$, i.e., $H[H(OI) \parallel H(PI)]$;
- The merchant can finally check the signature of the cardholder on $H[H(OI) \parallel H(PI)]$ with Eq. (3).

The purchase request is then processed, which includes forwarding the information

$$C_s(C),\ E_{PG},\ H(OI),\ Sign_C[H(H(OI) \parallel H(PI))])$$

to the payment gateway, for authorization.

Authorization Response: If I_C, I_M, R and T are compatible, the payment gateway should be able to obtain correct k and then PI from E_{PG}. If OI and PI match, the dual signature can be verified with H(OI), PI and $Sign_C[H(H(OI) \parallel H(PI))]$ in the similar way as in the former step. If no problem, it ensures that the PI has not been tampered with in transit and that it was signed using the cardholder private signature key.

Next, the PG formats and sends an authorization request to the Issuer via a payment system. Upon receiving an authorization response, the PG generates and digitally signs an authorization request message, which includes the Issuer's response and a copy of the PG signature certificate. Then the PG sends the authorization response to the merchant.

Purchase Response: This step is the same as that of the SET protocol. After the OI has been processed, the merchant software generates and digital signs a purchase response message, which indicates that the cardholder's order has been received by the merchant. The response is then transmitted to the agent.

If the payment is authorized, the merchant will fulfill the order by shipping the goods or performing the services indicated in the order.

Agent Return: After obtaining the purchase response from the merchant server, the agent puts the signature certificate of the merchant, purchase response and the digital signature of the merchant on the purchase response into its body and then migrates back to the cardholder's device. The cardholder's software then proceeds as in SET's final step.

4.3. Extended application of the SET/A+ protocol

SET/A+ is not only suitable for the payment system in which single merchant is involved, but also eligible for the payment system with multi-merchants. For example, a cardholder wants to order a few goods from different merchants respectively at one time, or to try different merchants until the goods is ordered. To fulfill this mission, it is necessary for the cardholder to specify a list of merchants in the information (M_{TVC}) for the trust verification center and the corresponding roam route through the Internet and some transaction identifiers for the agent. After traversing the scheduled merchant servers, the agent brings a series of purchase responses back to the cardholder device.

Next, we give a simple example to explain the extended application of the SET/A+ protocol. Let us imagine that a cardholder wants to order a salable goods and he knows that a series of merchants sell the salable goods by inspecting a catalog (printed in paper,

supplied on a CD-ROM or available online on the Web) of merchants, but he does not affirm which merchant has sold out the salable goods. Of course, he is reluctant to see his payment agent back without a useful purchase response from a visited merchant. In this case, the cardholder can specify a list of merchants visited by the agent in the data portion of the agent. When the agent is told that the salable goods is out of stock by a merchant, it will automatically roam to next merchant on the list. Until the agent finds a merchant who holds the salable goods for sale, true SET/A+ protocol begins.

The main reason why SET/A+ is most suitable for multi-merchant payment system is that no fixed public keys of merchants or secure agent execution environment are required when generating a payment agent.

5. Description of the proposed agent-based framework for internet trading in mobile computing environments

In this section, we consider combining the proposed agent-based auction-like negotiation protocol and payment protocol together to obtain an integrative solution for Internet trading in mobile computing environments. The integrative solution can be depicted in figure 9.

The agent-based framework for Internet trading in mobile computing environments shown in figure 9 can be described as follows:

Generation of Payment Agent (PA): The buyer creates a payment agent on basis of the SET/A+ protocol. Different from SET/A+, the buyer in the proposed framework will directly send a payment agent to the trusted verification center (TVC). Therefore, there are differences between the current payment agent and the previous payment agent. The differences lie in:

1. R can not been put into the date section of the current payment agent.

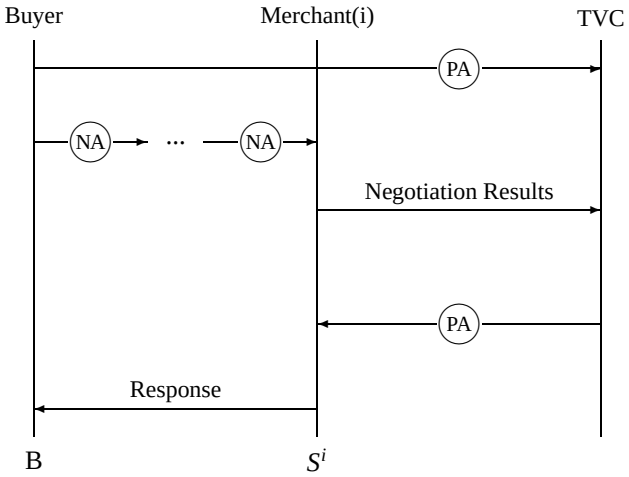

Figure 9. Agent-based framework for Internet trading.

2. The current M_{TVC} contains the transaction identifier I_C and the time stamp T, but instead of the name of the exact merchant host from which the cardholder will order goods, a list of merchant hosts from which the cardholder probably order goods is included in M_{TVC}. In addition, the current M_{TVC} should include the highest price (HP) that the cardholder can accept to buy the goods.

Generation of the Negotiation Agent (NA): The negotiation agent in the proposed framework is almost the same as that in the previous negotiation protocol. A little modification is adding R and the order information (OI) in the data section of the current negotiation agent.

Flow of Mobile Agents in the Proposed Framework: The flow of mobile agents in the proposed framework can be described in the following steps:

1. On basis of figure 9, the payment agent is sent to the trusted verification center. After arriving at the TVC, the agent hands in its signature certificate and signature on the agent to the TVC for verification.

   If no problem, the TVC retrieves $(k + I_C \cdot T + R)$ and decrypts the M_{TVC} in the way of verification request procedure of SET/A+, then supplies an agent execution environment for the payment agent to run, i.e., waiting for the negotiation results from the negotiation agent.
2. The negotiation agent from the buyer roams the list of merchants to negotiate the price of the goods by using the proposed agent-based auction-like negotiation protocol. After the negotiation process is completed, the negotiation agent in the winning merchant server (supposed to be S^i) submits the negotiation results along with a verification request which contains a unique transaction identifier I_M specified by the merchant to this trade, the merchant signature certificate ($C_s(M)$) and payment gateway key-exchange certificate ($C_k(PG)$) that correspond to the payment card brand specified by the cardholder in the data section of the negotiation agent.
3. The payment agent checks the negotiation results just like the buyer to do in Section 3 on behalf of the buyer. If the lowest bid (Bid_L) of the negotiation results is lower than that of the highest price (HP) which the buyer can accept to buy this goods, the payment agent verifies $C_k(PG)$ and creates a message which includes:

   $$E_k(PI), PG_p((k + I_C \cdot T + R) + I_M \cdot T), H(PI), Sign_C[H(H(OI) \parallel H(PI))]$$

   and sends it to the negotiation agent residing in the merchant server S^i.
4. After receiving the above confirmation, the negotiation agent in S^i turns to the payment agent. It creates the digital envelope E_{PG} for the payment gateway in the way of (12). Then the purchase request including

   $$C_s(C), OI, H(PI), Sign_C[H(H(OI) \parallel H(PI))], E_{PG}$$

   is sent to the merchant.
5. The other steps, such as authorization request, authorization response, and purchase response, are the same as the SET/A+. The purchase response is directly sent to the buyer or the cardholder.

6. Security issues

In this section, we will firstly analyze the security of the proposed signcryption scheme and then discuss how the agent-based auction-like negotiation protocol protects negotiation results against potentially malicious merchants and how the SET/A+ protocol protects the cardholder payment information against potentially hostile merchants. Finally, we will simply explain the security of the proposed agent-based framework for Internet trading in mobile computing environments.

6.1. Security of the signcryption scheme

We now consider that a signer signs a message by adopting the signature scheme of the new signcryption scheme described in Section 2.5. For example, the entity A signs M by generating a pair (s, t) according to Eqs. (1) and (2). Because of the participation of the entity A's secret key in the formation of the digital signature (s, t) of the entity A on M, an attacker can not directly use the Eqs. (1) and (2) to forge a valid signature pair (s, t).

A direct method by which an attacker may try is choosing a random number s (or t) and then solving out t (or s) from the Eq. (3) to forge a valid pair (s, t). However, the difficulty is at least equal to that of computing discrete logarithm over finite fields.

In view of the above analysis, we conclude that:

Theorem 4. *The difficulty of breaking the signature scheme of the new signcryption scheme is at least equivalent to the difficulty of computing discrete logarithm over finite fields.*

We now take the key distribution between an entity A and an entity B as an example. Except the entity A and the entity B, the others do not know how to compute $y_B^{(-r\cdot s - x_A\cdot t)(mod\ q)}$ $(mod\ p)$ (or $s^{x_B\cdot s}\cdot y_A^{x_B\cdot t}(mod\ p)$) because at least one of the entity A's or the entity B's private signature keys is definitely required in the computation. Therefore, the others can not separate the shared secret key k between entity A and entity B from z like the process in Theorem 2. The idea is similar to the scheme proposed by El Gamal [8], in which message M is sent to an entity B in the form $(g^r, M\cdot y_B^r)$, where r is a random number, where $y_B = g^{x_B}(mod\ p)$. It suffices for the sender to know y_B, whereas B can recover M by computing first $g^{-r\cdot x_B}$. An eavesdropper faces the problem of computing discrete logarithms.

On basis of the above analysis, we conclude that:

Theorem 5. *The difficulty of breaking the key distribution scheme of the new signcryption scheme is equivalent to the difficulty of breaking the El Gamal scheme.*

6.2. Protection of negotiation results against malicious environments

In order for a negotiation agent to run, it must expose its code and data to the host environment which supplies the means for that agent to run. Therefore, the host can always scan the

agent for information, alter the agent state and code, even kill the agent. Thus, the trade agent is unprotected from the host.

Current consensus is that it is computationally impossible to protect mobile agents from malicious hosts. Instead of tackling the problem from a computational (difficult) point of view, current research [21] is looking at sociological means of enforcing good host behavior. In order to enforce good merchant behavior, we need firstly to establish some rules for the merchants to run negotiation agents.

Rules for Merchants to Run Negotiation Agent: These rules can be outlined as follows:

1. Whether merchants bid or waive, they must sign their bids or waiving declarations and put their decisions into the state of the negotiation agent.
2. Sending the negotiation agent to the next server on the list of merchant servers which have not been visited in this round bid of auction until the next server declares correctly to receive the agent is the duty of merchants. If the next server rejects to receive the agent or no reply returns, the current server should send an error notification to an emergency server.
3. Each merchant must keep the state of a passing negotiation agent from the previous server and the signature of the previous server on the state in each stage as no-repudiation evidence for a period.
4. Merchants are prohibited to tamper other merchants' negotiation results.

As the sociological laws and regulations, the above rules can not ensure that it is impossible for merchants to break a law. However, we are always able to detect the kind of irregularities and finally dig out the breeder. If a merchant is accused as not honest, trading agents will never visit the merchant server again. It will result in decrease of the merchant's retail profit.

Protection of Negotiation Results to Be Tampered: If the payment agent residing in the trusted verification center finds any problem when it checks the negotiation results or it receives an error notification from the negotiation agent, it will notice the TVC and the TVC will performs the following procedure to dig out the breeder:

1. TVC informs all the merchants on the list of merchants in the negotiation agent that an error occurs and it is probably caused by malicious behaviors and asks them to commit their non-repudiation evidences.
2. Those merchants who observe the negotiation rules certainly prefer to provide their true non-repudiation evidences to the TVC so as to prove them guiltless. The merchant who breaks the negotiation rules fears to do like this. But he have to do, otherwise he will be doubted as a malicious merchant in the beginning.
3. After the TVC obtains all the states of the negotiation agent and the signatures of merchants on the states, it can recover the agent in each stage.
4. Suppose the state of the agent in S^j is $State_j^k$ (where k represents the bid round number of auction), then $State_j^k$ and $State_{j-1}^k$ have the following relation:

$$State_j^k = State_{j-1}^k \leftarrow \left(NR_{S^j}^k \parallel Sign\left(NR_{S^j}^k\right)\right) \tag{13}$$

where "$\leftarrow$" means using the latter term to replace the relevant portion in the former term.

If the TVC have not found any problem before S^j, it can judge the merchant S^j by checking whether the above equation holds. In Eq. (13), $State_j^k$ (including $NR_{S^j}^k \parallel Sign(NR_{S^j}^k)$) and $State_{j-1}^k$ are provided by S^{j+1} and S^j respectively, both of them are provided by S^j and S^{j-1} and ensured to be true with their signatures.

In addition, the final winner of the auction may take a strategy against our negotiation protocol. At first, he offers his bid in very low level so that the other bidders have to waive and sign a waiving declaration. At last, he alters his bid into a higher level and signs it. This attack cannot succeed in our negotiation protocol because each waiving bidder has included the current lowest price in his waiving declaration. The bid of the final winner must be the lowest of all bids.

On basis of the above discussion, we can conclude the following theorem:

Theorem 6. *If the negotiation rules* (1)–(4) *can be set up in the agent-based auction-like negotiation protocol, any malicious behavior can be detected and the breeder can be dug out.*

6.3. Protection of payment information against hostile environment

In the Internet payment system, one obvious concern is to protect the user's critical data, in particular the credit card information. SET's use 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 critical data.

Slightly Different from SET, both SET/A and SET/A+ dedicate to apply agent technology to overcome the difficulties faced by SET in mobile environment. Besides the consideration of protection of user's credit card information against the malicious host, it is required to think of security issues of mobile agents in the SET/A+ protocol.

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

In accordance with the above considerations, in SET/A+ protocol, the cardholder is asked randomly to choose his own symmetric key k and personally encrypts the payment information with the key before the agent leaves the cardholder. Besides the cardholder and the payment gateway, no other participants can know the symmetric key k if we suppose that the trust verification center and the merchant never collude. The two reasons are as follows:

- Although the TVC can retrieve $(k + I_C \cdot T + R)$ with its private signature key, it has no knowledge of k in view of the existence of random number R. R is kept in the

agent execution environment located at merchant server and is never sent to the TVC. Furthermore, the TVC has no access to the order information.

- Although the merchant can obtain the payment information encrypted with k, he can not retrieve either $(k + I_C \cdot T + R)$ from z or $(k + I_C \cdot T + R + I_M \cdot T)$ from the envelope E_{PG}. Therefore, the agent do not disclose the payment information to the merchant in SET/A+ protocol.

In view of the above reasons, we conclude:

Theorem 7. *Under the assumption that the trust verification center and the merchant never collude, the SET/A+ protocol ensures the same level of protection as the SET protocol.*

6.4. Protection of merchant servers against malicious mobile agents

A mobile agent is unique in that its code is executed by a server. Thus an executing agent has automatic access to some of a server resources. With this level of access agents can mount attacks by propagating viruses, worms and Trojan horses, impersonating other users and mounting denial of service attack. The standard approach to this problem is to reject all unknown code from entry into servers. It is not a viable solution in a mobile agent environment. Both Telescript [30] and Safe-Tcl [3] offer approaches to solve the problem.

In the proposed agent-based framework for Internet trading, before a merchant server supplies an agent execution environment for a visiting agent to run its code, the server will verify the signature. Once any problem occurs when the server runs the code on an agent execution environment, the owner of the agent is probably malicious and will be accused.

6.5. Security of the proposed agent-based framework for Internet trading

The proposed agent-based framework for Internet trading in mobile environments is based on the agent-based auction-like negotiation protocol and the SET/A+ payment protocol. Therefore, its security is determined by the security of the two protocols.

The negotiation phase of the proposed framework is completely the same as the agent-based auction-like negotiation protocol. Therefore, the security of this phase is also the same as the protocol.

Although the payment phase of the proposed framework is slight different from the SET/A+ protocol, their design criterion is same, i.e., do not disclose classified information to the wrong entities. For example, although the TVC can retrieve $(k + I_C \cdot T + R)$ with its private signature key, it has no knowledge of k in view of the existence of random number R. R is kept in the agent execution environment located at a merchant server and is never sent to the TVC. Therefore, the security of this phase is the same as that of the SET/A+ protocol.

7. Performance analysis of the agent-based framework for Internet trading

We have designed the agent-based framework with mobile computing in mind, especially focused on the two factors we have 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.

In this section, we discuss the advantages and identify possible scenarios in which the usage of the proposed agent-based framework 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 environments.

7.1. 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 online shopping software on this kind of devices, either the CPU capacity has to be increased, or the load has to be transferred to an external machine.

The proposed agent-based framework solves this problem by (1) moving complete negotiation process to merchant servers and (2) adding a trust verification center on the Internet and doing part of cryptographic work at the TVC (or, at least, outside the cardholder's device). Therefore, this proposal is able to remove the computational burden from the user's device, which can be disconnected while the transaction is occurring.

There is still the need to generate a pair of signature keys on the cardholder's side, but this belongs to another phase in the framework (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.

7.2. Minimizing costs

One of the problems of Internet trading 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 customers 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 the payment protocol in the proposed agent-based framework contributes significantly for minimizing the customer costs. In addition, the payment protocol omits the key-exchange operation in the cardholder side and adopts the same block cipher in hash function and encryption operation. This purpose is to save storage in the mobile device.

On the merchant's side, having an agent operating inside its server means having to let one 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 the trust verification center and other merchant servers 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 the

proposed agent-based framework. So, given that the proposal does not increase costs, there is no loss in supporting this framework. On the other hand, it can be quite rewarding, from a marketing point of view.

7.3. Improvement to the traditional online retail auctions

The proposed agent-based auction-like negotiation protocol improves the traditional auction protocols such as Onsale (http://www.onsale.com), FirstAuction (http://www.firstauction.com) and etc. from the following sides:

1. Traditional online retail auctions force consumers to compete with one another for a specific merchant offering. This brings in "winner curse" (winning bid is greater than the product market valuation) which in turn hurts the benefit of retailer himself. Instead of consumers competing with one another for a specific merchant offering, the proposed agent-based auction-like negotiation protocol forces merchants to compete for consumer patronage. In this way, our proposal completely overcomes "winner curse" which benefits the retailer in the long run. This makes the auction appears more forgiving than the traditional counterparts.
2. In traditional online retail auctions, the long delay between starting auctions and purchasing the product retards a large number of impatient or time—constrained consumers. In the proposed agent-based auction-like negotiation protocol, a negotiation agent, working on behalf of consumer, acts as a mobile auctioneer, while online retailers bid for low price. Because the agent actively finds potential bidders in a list of merchants and does not wait for a critical mass of bidders to complete auction, the auction efficiency is certainly much higher than that of traditional online retail auction. This also means the time consuming during the auction in our proposal is much shorter than that in traditional online retail auction. By limiting the negotiation agent to roam in a small or large range of merchant servers, the hurry buyer may let agent return early with somewhat higher purchase price, while patient one may keep agent searching until locating a satisfying bargain. The auction process therefore can cater to the different requirements from customers.
3. Traditional online retail auctions have brought communication intensifying: during the online auction, the salesroom server will become the center of the information exchange. This increases the communication load of certain network segment and the process load of the server. By contrast, the proposed agent-based auction-like negotiation decreases communications during the auction. Since most communications happen among different servers, the loads are also balanced to the different network segments. This improves the network performance as a whole. In addition, the communication load in the customer side is decreased to minimum, i.e., one time for delivering the negotiation agent and another for receiving the purchase response. Therefore, it is most suitable for the customer in mobile computing environments.

7.4. Comparison of the SET/A+ with SET

By comparing with SET, the merchant software in SET/A+ remains unchanged except providing an agent execution environment for a payment agent to run. The payment gateway

software in SET/A+ only needs to make slight modification when retrieving the symmetric key k. The main difference between SET/A+ and SET lies in the cardholder software.

SET/A+ is not only suitable for payment in mobile computing environment, but also eligible for online payment on the Internet. Let us imagine the situation in which a cardholder tries to order a few goods from different merchants on the Internet or order salable goods until a purchase is fulfilled. To satisfy this requirement, the cardholder in SET/A+ only need to send an agent to the Internet and receive the purchase responses brought back by the agent. It totals to two communications for the cardholder. However, if using SET, it will need much more communications for the cardholder than using SET/A+.

As a consequence, the applications of SET/A+ will decrease the communication load for a cardholder to minimum. This is a notable characteristic of SET/A+.

7.5. Comparison of the SET/A+ with the SET/A

Although both SET/A+ and SET/A are agent-based payment systems, their required supports are distinct. In SET/A, each merchant on the Internet need to provide a secure agent execution environment for agents to run. This solution would increase the security level at the cost of an additional investment in hardware from the merchant. In addition, unless the secure agent environments owned by distinct merchants have same public keys, SET/A needs to know them firstly before sending an agent out.

SET/A+ removes the limitation of secure agent execution environment in SET/A by adding a trust verification center in the payment system. It only needs to know the public signature key of the trust verification center.

8. Conclusion

The recent burgeoning of new communications technologies and, in particular, the Internet explosion has brought electronic commerce to the brink of widespread deployment. However, businesses are wary about treading beyond that brink, largely because of concerns about unknown risks they may face. The key to alleviating many of these concerns—to mitigating the risks—is security.

Although the traditional auction protocols such as Onsale (http://www.onsale.com) and FirstAuction (http://www.firstauction.com) can easily deal with security issues, they suffer from some other problems such as “winner’s curse” and low performances. It makes online auction less forgiving than would be expected in retail shopping.

In this paper, we have proposed a new secure agent-based auction-like negotiation protocol for Internet trading in mobile environments, whose features lie in: (1) negotiation for agent-based trading is performed through a novel pattern of electronic auction. (2) negotiation results in merchant servers are ensured to be valid with their signatures. (3) malicious behaviors can be detected and the breeder can be dug out by the help of sociological factors.

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 view of this, SET/A, based on the SET protocol and the mobile agent model, has recently been proposed. In order to protect an agent from a potentially malicious environment, SET/A depends on a secure execution environment in the merchant's server for an agent to run. However, the required security is not easy to ensure.

In order to remove the limitation of the security of the agent execution environment, a novel secure agent-based payment system for mobile computing on the Internet (namely SET/A+) has been proposed in this paper. By adding a trust verification center into the payment system, SET/A+ is able to ensure the same level of security as SET, providing an alternative means of online payment using the SET protocol.

Finally, by combining the proposed agent-based auction-like negotiation protocol and the SET/A+, we obtain an integrative solution for Internet trading—A secure agent-based framework for Internet trading in mobile computing environments.

We are also interested in keeping the agent as intelligent and autonomous as possible, allowing it to make its own decisions (even if very simple) when needed. As part of our future work, we intend to use Aglet language to implement the agent-based framework for Internet trading, with agents capable of negotiating with their hosts on basis of the knowledge they carry as they migrate from one server to another and pay for goods on behalf of their owner.

Acknowledgment

The author would like to express appreciation to the anonymous reviewers for their valuable comments.

References

1. C. Beam and A. Segev, "Electronic catalogs and negotiations," CITM Working Paper 96-WP-1016, available at http://haas.berkeley.edu/citm/wp-1016-summary.html.
2. C. Beam, A. Segev, and J.G. Shanthikumar, "Electronic negotiation through Internet-based auction," CITM working paper 96-WP-1016, Haas School, Berkeley, 1996.
3. N. Borenstein, "Email with a mind of its own: The Safe-Tcl language for enabled mail," IFIP WG 65 Conference, Barcelona, May, 1994, North Holland, Amsterdam, 1994.
4. A. Chavez and P. Maes, "Kasbah: An agent marketplace for buying and selling goods," in Proceedings of the First International Conference on the Practical Application of Intelligent Agents and Multi-Agent Technology, London, UK, April 1996.
5. D. Chess, B. Grosof, C. Harrison, D. Levine, C. Parris, and G. Tsudik, "Itinerant agents for mobile computing," Technical Report, IBM T.J. Watson Research Center, NY, October 1995.
6. D. Chess, C. Harrison, and A. Kershenbaum, "Mobile agents: are they a good idea," Technical Report, IBM T.J. Watson Research Center, NY, March 1995.
7. 3Com Corporation, Palmpilot, http://www.3com.com/palm.
8. T. ElGamal, "A public key cryptosystem and a signature scheme based on discrete logarithm," IEEE Trans. Info. Theory, vol. IT-31, no. 4, pp. 468–472, July 1985.

9. J. Gosling and H. McGilton, “The Java language environment,” Sun Microsystems white paper, 1995.
10. R.H. Guttman and P. Maes, “Agent-mediated integrative negotiation for retail electronic commerce,” Proceedings of Workshop on Agent Mediated Electronic Trading, Minneapolis, Minnesota, USA, May 1998.
11. L. Hurst, “MCK: mobile communication kernel,” Dagstuhl Seminar on Mobile Software Agents, October 1997.
12. IRIDIUM LLC, The IRIDIUM system, http://www.iridium.com/system/system.html.
13. ISO/IEC 8696-8 Information Technology—Open System Interconnection—The Directory: Authentication framework, 1993.
14. N. Jennings and M. Wooldridge, “Software agents,” IEEE Review, January 1996.
15. X.J. Lai and J.L. Massey, “A proposal for a new block encryption standard,” Advances in Cryptology, Proc. of EUROCRYPT’90, Lecture Notes in Computer Science, vol. 473, pp. 389–404, 1991.
16. P. Maes, “Agents that reduce work and information overload,” Communications of the ACM, vol. 37, no. 7, pp. 31–40, 146, ACM Press, July 1994.
17. J. Montgomery, “The orbiting Internet: Fiber in the sky,” Byte, vol. 22, no. 11, November 1997.
18. A. Moukas, R. Guttman, and P. Maes, “Agent-mediated electronic commerce: an MIT media laboratory perspective,” to appear in Proceedings of the International Conference on Electronic Commerce.
19. Nokia, Nokia 9000 communicator, http://www.nokia.com/com9000/n9000.html.
20. Psion, Psion series 5 handheld computer, http://www.psion.co.uk/series5.
21. L. Rasmusson and S. Janson, “Simulated social control for secure Internet commerce,” in New Security Paradigms’96, ACM Press, September 1996.
22. Artur Romao and Miguel Mira da Silva, “An agent-based secure Internet payment system for mobile computing,” TrEC’98, Hamburg, Germany, 3–5 June 1998, LNCS, vol. 1402, Springer.
23. K. Rothermel and R. Popescu-Zeletin (Eds.), “Mobile agents,” Lecture Notes in Computer Science, vol. 1219, Springer, April 1997.
24. T. Sander and C. Tschudin, “Towards mobile cryptography,” Technical Report TR-97-049, International Computer Science Institute, November 1997.
25. T. Sander and C.F. Tschudin, “Protecting mobile agent against malicious hosts,” Mobile Agents and Security, LNCS 1419, Springer-Verlag, 1998.
26. T. Selker, “A teaching agent that learns,” Communications of the ACM, vol. 37, no. 7, 1994.
27. “The digital signature standard,” Communications of the ACM, vol. 35, no. 7, pp. 36–40, 1992.
28. Visa International and MasterCard International, Secure electronic transaction (SET) specification, Version 1.0, May 1997.
29. J. Vitek and C. Tschudin (Eds.), “Mobile object systems—towards the programmable Internet,” Lecture Notes on Computer Science, vol. 1222, Springer, July 1996.
30. J. White, “Telescript technology: the foundation of the electronic market,” General Magic white paper, 1995.
31. U. Wilhelm and X. Defago, “Objects proteges cryptographiquement,” in Proceedings of RenPar’97, Lausanne, Switzerland, May 1997.
32. L. Wirthman, “Gradient DCE has sign-on feature,” PC Week, March 1996.
33. B. Yee, “A sanctuary for mobile agents,” in Proceedings of the DARPA Workshop on Foundations for Secure Mobile Code, Monterey, CA, USA, March 1997.
34. X. Yi, “On design and analysis of a new block cipher,” Proceedings of 1996 Asian Computing Science Conference, Asian’96, Singapore, LNCS, vol. 1179, Spring-Verlag, December 1996.
35. X. Yi and K.Y. Lam, “Hash function based on block cipher,” IEE Electronics Letters, vol. 33, no. 23, 1997.
36. X. Yi, X.F. Wang, K.Y. Lam, E. Okamoto, and D. Frank Hus, “A secure auction-like negotiation protocol for agent-based Internet trading,” in Proceedings of 17th IEEE Symposium on Reliable Distributed Systems, Purdue University, IN, USA, 20–23 October 1998.
37. Y. Zheng, “Digital signcryption or how to achieve cost(signature & encryption) $\ll$ cost(signature) + cost(encryption),” Advances in Cryptology—Crypto’97, Lecture Notes in Computer Science, vol. 1294, pp. 165–179, Springer-Verlag, 1997.