A Modularized Electronic Payment System for Agent-based E-commerce

Sheng-Uei Guan, Sin Lip Tan and Feng Hua

Department of Electrical & Computer Engineering
National University of Singapore
10 Kent Ridge Crescent, Singapore 119260

With the explosive growth of the Internet, electronic-commerce (e-commerce) is an increasingly important segment of commercial activities on the web. The Secure Agent Fabrication, Evolution and Roaming (SAFER) architecture was proposed to further facilitate e-commerce using agent technology. In this paper, the electronic payment aspect of SAFER will be explored. The Secure Electronic Transaction (SET) protocol and E-cash were selected as the bases for the electronic payment system implementation. The various modules of the payment system and how they interface with each other are shown. An extensible implementation using JavaTM will also be elaborated. This application incorporates agent roaming functionality and the ability to conduct e-commerce transactions and carry out intelligent e-payment procedures.

ACS Classification: H.4 (Information Systems Applications), H.4.3 (Communications Applications)

1. INTRODUCTION

Commercial activities on the Internet have increased in tandem with the fast growth of the Internet itself. With electronic commerce (e-commerce), business transactions have been made easier and faster via the Internet. However, there are still uncertainties and lack of standardized e-commerce procedures. This has slowed down the acceptance of e-commerce activities online. It would thus be beneficial if there was some way to streamline and standardize e-commerce.

Agent technology was introduced to e-commerce to provide automation in conducting business transactions. Agents can perform tasks autonomously on behalf of its user. Hence, an agent framework and administration infrastructure called SAFER (Secure Agent Fabrication, Evolution and Roaming) has been proposed (Zu *et al*, 2000; Guan and Yang, 2002; Wang *et al*, 2002; Guan and Zhu, 2002; Ng *et al*, 2002; Sim and Guan, 2002; Yeo *et al*, 2002; Wang and Guan, 2000). The goal of SAFER is to construct open, dynamic, and evolutionary agent architecture for e-commerce. This solution makes use of software agents to carry out product search and differentiation on behalf of human owners. It has the potential to allow e-commerce transactions and payment to be carried out with good security and reliability.

Copyright© 2004, Australian Computer Society Inc. General permission to republish, but not for profit, all or part of this material is granted, provided that the JRPIT copyright notice is given and that reference is made to the publication, to its date of issue, and to the fact that reprinting privileges were granted by permission of the Australian Computer Society Inc.

Manuscript received: 19 September, 2002
Communicating Editor: Joan Cooper

This paper will elaborate on the design of a modularized payment system for SAFER. It will give an idea of the various technologies used in the implementation process of the payment system for SAFER. The background of the research will first be introduced in Section 2 including agent technology, and current payment schemes. An overview of the SAFER payment system is presented in Section 3. The modular design of the implemented Java application is then given in Section 4. In Section 5, a discussion of the implementation is included. The advantages of the design are discussed and possible technical considerations are explained. Comparison to related work is covered in Section 6. The SET (Loeb, 1998) (Secure Electronic Transaction) protocol is explained in Appendix I.

2. AGENTS AND E-PAYMENT SYSTEMS

Agents are bits of software that help computer users by performing routine tasks, typically in the background on behalf of its user. Information gathering, filtering and presentation are some well-defined tasks prescribed to agents. Traditional software such as word processors and spreadsheets only respond to human input in a fixed and predictable manner. Intelligent agents are capable of "thinking" and producing intelligent feedback.

2.1 Overview of the SAFER Architecture

SAFER (Zhu *et al*, 2000) is an infrastructure designed to serve agents in e-commerce and to establish necessary mechanisms to manipulate them. It focuses on three fundamental activities of agents, namely, fabrication, evolution and roaming.

SAFER agents are envisaged to act on behalf of customers to carry e-commerce activities in a simple and independent manner. Common tasks that could be entrusted to agents include product search, negotiation over interested items, payment, etc.

The agent community is the basic unit in SAFER (Figure1). A network client can join a SAFER community by applying to a local Community Administration Centre (CAC). CAC will issue a digital certificate to the applicant if it accepts the application. This certificate can be used to identify clients' agents by trusted remote hosts that the agents roam to. Under these organized communities, agents are fabricated by the Agent Factory per member's request. After customization, they can be controlled by individual owners through a coordinating entity called Agent Butler.

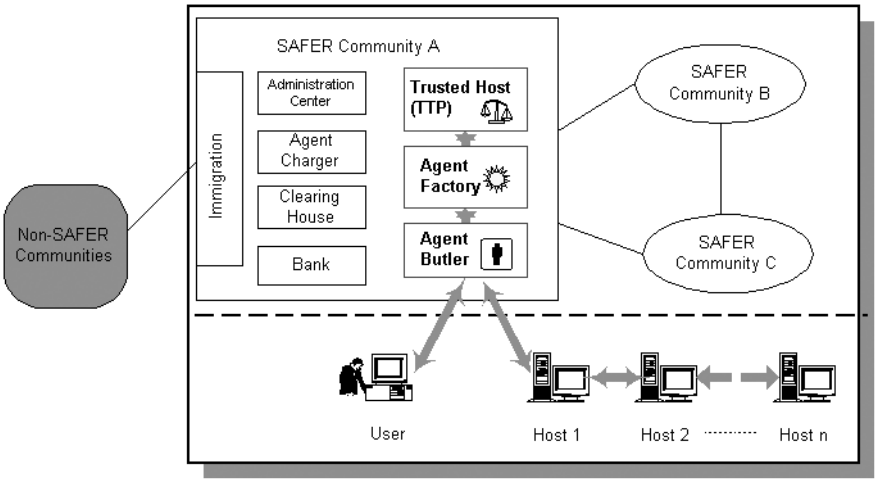

Figure 1: SAFER Agent Communities

2.2 Electronic Payment Schemes

A key element in any e-commerce system is the method of payment. However, existing monetary and fund-transfer arrangements are difficult to be transplanted directly into the e-commerce marketplace. Currently, a common e-payment method involves the client transmitting to the merchant details of a payment card such as a VISA credit card. The merchant receives the information and proceeds to carry out a payment request with the card issuer via traditional payment card procedures. This system is simple and does not require the development of a new commercial infrastructure. However, the system is susceptible to frauds from either transacting parties. The card information transmitted over the Internet could also be stolen by malicious parties. Many electronic payment schemes have been proposed but not all of them offer solutions to these problems.

One research project called BABSy proposed by Rockinger and Baumeister (2000) is based on the consumer buying the behaviour model listed in the next section. It is claimed to be an accounting system that helps automated payment in an agent based e-commerce environment. In BABSy, there are only three types of agents which represent the three parties involved in an e-commerce transaction: merchant, bank and user. They are service agent, accounting agent and user agent.

Research work of an Agent-based Bill Payment Service (ABPS) (Wong and Lau, 2000), also conducted at Queensland University, is hosted on a website which is certified digitally. Consumers must first register with ABPS by providing their personal information. To acquire services, customers authorize an ABPS payment agent to pay the related parties. In their system, the payment agent is responsible for obtaining settlement instructions and settling bills via appropriate financial institutes or external payment services.

Project Eleanor is an Identrus (2003) initiative to introduce secure, direct business-to-business payments on the Internet. Project Eleanor aims to provide Web-based specifications to initiate B2B payments on traditional bank systems. Project Eleanor includes six B2B e-payment options, including payment orders, conditional payment orders, etc. Trading partners will have pre-established instructions with their banks for payment authorization, routing and settlement.

As a different example, the IBM Multi-payment Framework (2003) (MPF) offers a suite of software products enabling merchants to use multiple types of payment in Internet commerce. The kernel of the framework is implemented in the IBM WebSphere Payment Manager which is an electronic cash register for merchants. This is an offering for service providers to host payment for multiple remote merchants. It allows merchants to receive payments from consumers on the Internet and to process those payments with banks and financial institutions. Their system enabled merchants to provide or utilize as many payment mechanisms as the customers may need.

In this paper, the *Secure Electronic Transaction* (Loeb, 1998) (SET) protocol was chosen as the payment scheme because it satisfies the three criteria of security, scalability and compatibility.

The SET protocol is an evolution of the existing credit-card based payment system. It provides enhanced security for information transfer as well as authentication of transaction participant identities by registration and certification. SET is supported by major corporations such as VISA Inc. and MasterCard. SET is also an international standard with published protocol specifications.

Digital cash uses electronic tokens (mostly a unique coded string) to represent monetary value. The issuing bank of tokens has a record of all the tokens. The acquiring bank of the merchants that receive the tokens will transfer them to a clearing house to process them. When the tokens are verified by the issuing bank, the real transaction of funds will take place and the tokens cannot be used again. The usage of digital cash enables full anonymity that cannot be found in other payment systems. Some published works on digital cash include E-cash (Brands, 1995) NetCash, and CAFÉ (Mjolsnes and Michelsen, 1997) etc.

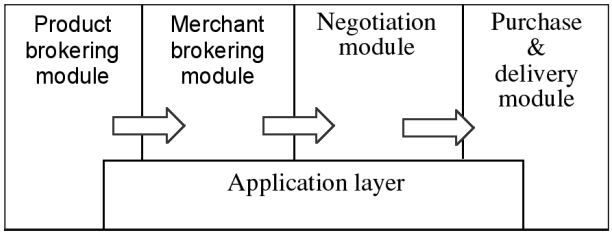

Figure 2: Buying Behaviour Model Structure

2.3 Agent Based E-Payment Systems for E-Commerce

To categorize existing agent-mediated e-commerce systems, a Consumer Buying Model was presented by Maes' in the MIT Media Lab (Guttman and Maes, 1998). According to the nomenclature, the model (Figure 2) is separated into six stages, namely:

1. Need Identification
2. Product Brokering
3. Merchant Brokering
4. Negotiation
5. Payment and Delivery
6. Product Service and Evaluation

These stages may overlap and migrate from one step to another in a non-linear and iterative way. The model helps provide a solution to identify the role of agents as mediator in e-commerce. However, there is no automated system today with all these stages. Some pilot research projects assist various stages of the buying process.

For example, an agent market place system called Kasbah (Chavz and Maes, 1996) was implemented by the MIT Media Lab using multiple agents that are intended to bring about changes in the way buying and selling is conducted and doing much of the work on the user's behalf. Buyers who need to procure particular goods would create an agent, give it basic strategic direction, and send it off into the electronic marketplace. The Kasbah agents would then pro-actively seek for potential sellers and negotiate with them on the buyer's behalf, based on a set of constraints specified by the buyer, including a highest acceptable price and a transaction completion date. However, it is clear that it only covers some aspects of the buying process, i.e. from stage two to four as listed above. It does not support the payment stage in their systems.

Here, we propose a modularized electronic payment system for agent-based e-commerce, especially for the SAFER architecture. It combines the agent technology with current payment schemes described in the previous section. The SAFER payment system does not limit itself to a fixed method for electronic payment. The payment functionality of agents or the Agent Butler is extensible and will be able to handle different forms of payment such as payment card or digital cash, etc. For the current system implementation, SET and E-cash were chosen as the payment schemes.

3. SAFER ELECTRONIC PAYMENT SYSTEM

In Section 2.1, we presented an overview of the SAFER community. Here, we discuss in detail the entities involved in the SAFER e-payment system under an agent-based SET protocol (Figure 3).

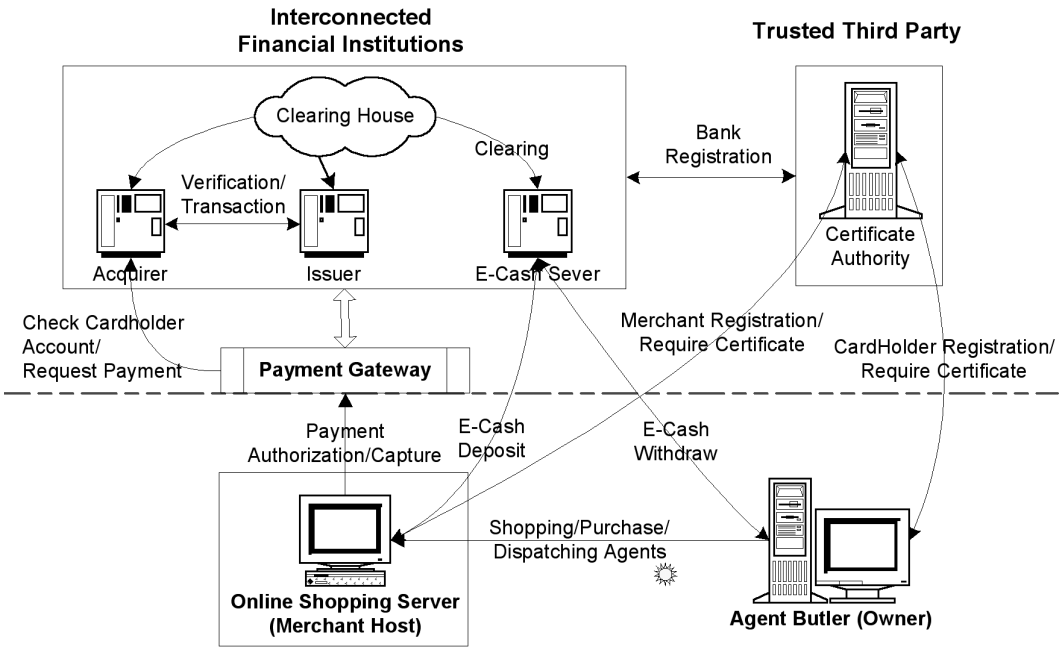

Figure 3: Entities Involved in the SAFER E-payment System

The Agent Butler represents the Cardholder who makes payment using a payment card through the SET (or E-cash) protocol. The Agent Butler resides in the user's PC as a static user agent and has a number of functions (Figure 3) pertaining to agent management and e-commerce. Firstly, the user interacts with the Agent Butler through the Agent Butler *User Interface*. Also, the Agent Butler can dispatch *Mobile Agents* to remote e-commerce hosts using the *Agent Transport* module. It receives messages and shopping information from dispatched agents through its *Agent Receptionist*. Finally, it carries out e-commerce transactions and payments through its *Financing Agency*.

Financial Institutions consist of bank servers and clearing houses. As depicted in Figure 4, the Issuer refers to the bank that establishes an account for the owner and issues the payment card or

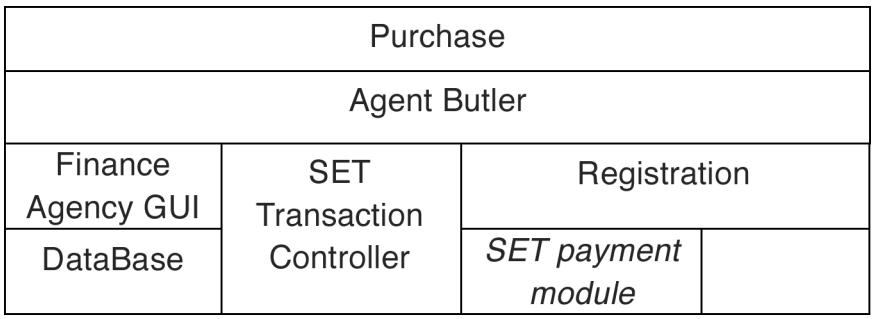

Figure 4: Function Modules in the Agent Butler

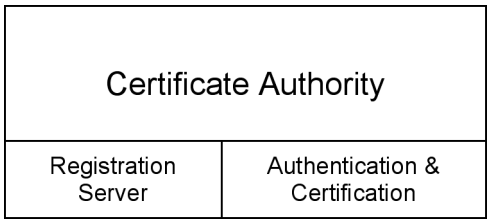

Figure 5: Modules in CA

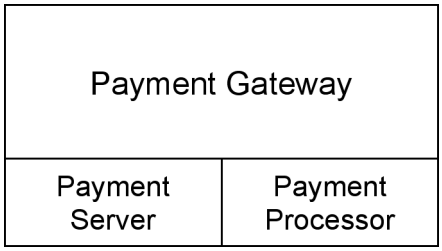

Figure 6: Modules in Payment Gateway

electronic checks to the account. It guarantees payment for authorized transactions using the payment card in accordance with payment card regulations. The Acquirer is the bank that establishes an account with the Merchant Host and processes payment cards or validates authorizations and transactions. Payment is implemented by a payer paying the payee via the Issuer and Acquirer (Chavz and Maes, 1996). E-cash server refers to the bank sever that handles issuing and verification of electronic currency.

Certificate Authority (Figures 3 and 5) is one of the indispensable entities under SAFER. The Certificate Authority (CA) is the provider of trusted digital certificates (digital certificates are described in Section 2.1). It runs a *Registration Server* to handle SET registration requests from both the Cardholder (Agent Butler) and Merchant Host. The processing of such requests is handled by the *Authentication and Certification* module.

The *Payment Gateway* (Figures 3 and 6) is similar to CA. It runs a *Payment Server* waiting to handle SET payment authorization or capture requests from the Merchant. When such requests arrive, the *Payment Processor* module processes the request.

Note that before the user or Agent Butler can proceed with any activities, CA, Payment Gateway, and E-cash Server should be permanently running and waiting for SET or E-cash transaction requests.

The *Merchant Host* (Figures 3 and 7) is an online e-commerce retailer that is willing to receive and run agents through the *Shopping Server*. It possesses product information in a locally accessible database for the agent to access and extract data. Each host runs in an autonomous fashion. It can carry out SET/E-cash transactions with the Agent Butler using the *Purchase Server*.

The Merchant Host will carry out merchant registration with CA as soon as it is set-up and running. Only after it has completed registration and obtained SET certification will it be capable of SET purchase transactions.

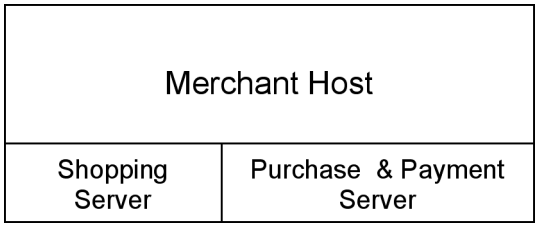

Figure 7: Modules that Comprise the Merchant Host

4. DESIGN AND IMPLEMENTATION OF A MODULARIZED ELECTRONIC PAYMENT SYSTEM FOR THE SAFER ARCHITECTURE

4.1 Agent Butler

Agent Butler plays a significant role in the whole architecture, especially in the payment transaction process. It has a number of functions that can be categorized into two major roles namely: 1) roles with the owner and 2) roles with the user. The first major role is its task with the owner. In the absence of its owner, the Agent Butler will, depending on the authorization given, make decisions on behalf of the agent owner. The stationary Agent Butler provides a Graphical User Interface (GUI) for accepting data input and displaying the results of a specific task to the owner instantaneously. In the second major role, the Agent Butler can dispatch mobile agents to Web hosts to carry out e-commerce activities. When mobile agents are sent out roaming in the network, the butler has the responsibility of keeping track of agent activities and locations by sending and receiving messages with agents. This is especially important when an agent's task is important. Agent Butler is necessary in all agent transactions, because in SAFER, mobile agents won't be given the authorization to carry big amounts of credits/E-cash.

The modular structure of the Agent Butler is shown in Figure 8. It comprises the information storage (database and archive), financing agency, agent receptionist, and agent dispatch module.

Information Storage

A Database object is owned by the Agent Butler and it stores for the Agent Butler information like the IP address of the host, the network port number to connect to, etc. Similar information is provided about the possible CAs that the Agent Butler can register with. It is assumed that as part of the SAFER community, agents would be fabricated in a remote Agent Factory before being sent to the owner. Information about the agents that the Agent Butler currently owns would be stored in the Database too.

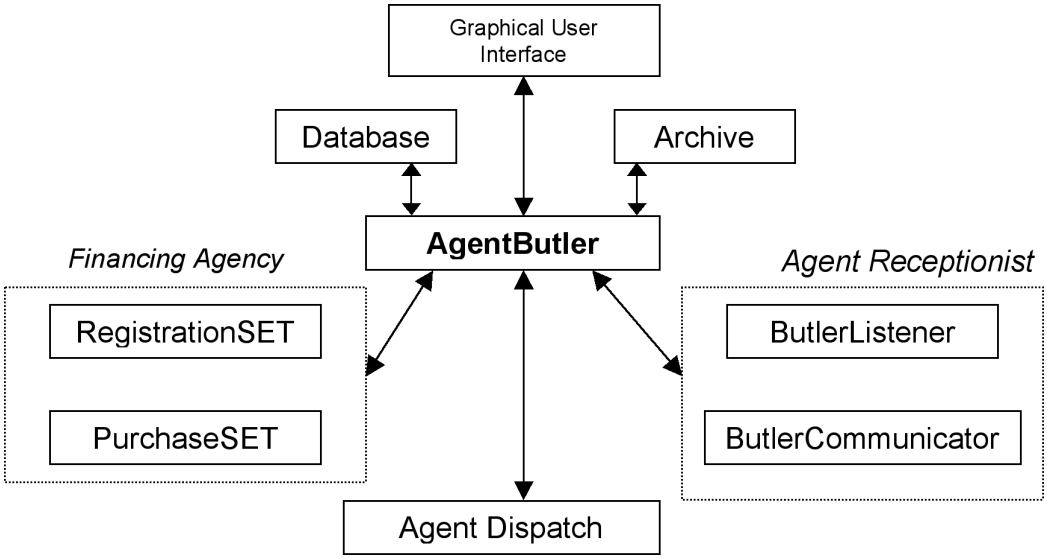

Figure 8: Agent Butler Modular Structure

Another object – the Archive also belongs to the Agent Butler, which is responsible for recording the past transaction information from the various e-commerce transactions and SET/E-cash payments.

Financing Agency
In this payment architecture, a subsystem called agency is in place. An agency can be considered as a multi-layered agent group or a federation of agents with specific goals and functional roles in the architecture. The Agent Butler is in charge of these subsystems, enabling each with some particular expertise. When a purchase decision is made, the Agent Butler will activate the Financing Agency to conduct transactions with the merchant host via certain payment schemes or protocols.

Agent Receptionist
A pair of communication objects handles external socket communications with dispatched agents. ButlerListener waits for messages or return information from agents in remote hosts while ButlerCommunicator is capable of sending messages to these agents.

Authentication and Agent Dispatch
The Authentication Module (see Appendix II for figure and more details) performs the authentication of Agent Butler before an agent can be transmitted to the host. After which, the Agent Butler dispatches or sends the agent out to the host.

4.2 Mobile Agent

4.2.1 Basic Agent Structure

The mobile agent's task is to assist in the initiation of electronic payment for the purpose of e-commerce between the Agent Butler representing the owner, and the host site.

Figure 9 shows the class modules that comprise the Mobile Agent object. In the centre is the Mobile Agent class, which controls the rest of its components. It can be regarded as a coordination centre without external functions. It contains details about each agent such as ID, date of creation,

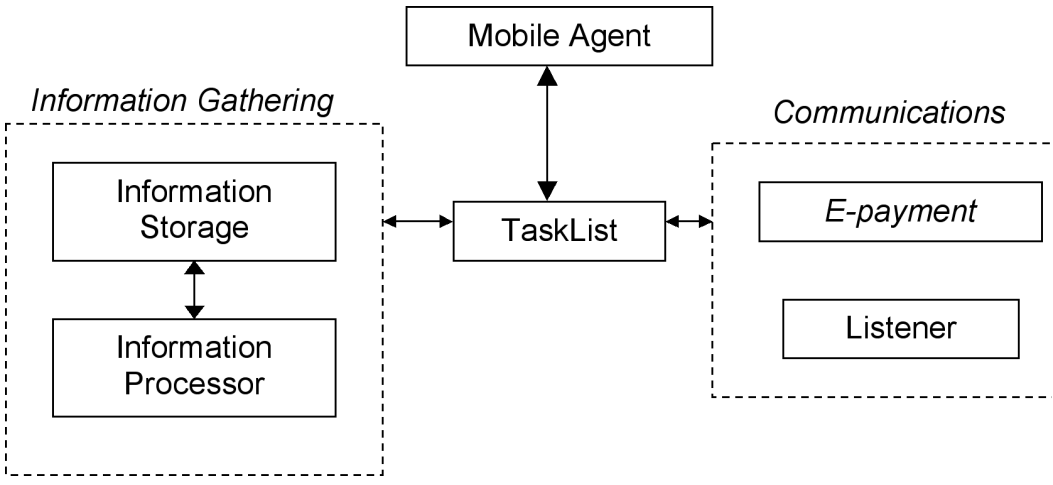

Figure 9: Modular (Class) Structure of a Mobile Agent

information about the originating host, etc. All these identity details arise from the need to be compatible with the SAFER architecture. This means that the agents are not just anonymous byte-code flowing around, but possess specific capabilities and unique identification to be residents of the SAFER communities. See Appendix III for details in the implementation of mobile agent.

4.2.2 TaskList

The TaskList is the foreman of the Agent entity, helping an agent to carry out its activities in sequence. It has a list of various objectives that has been given to the agent before dispatching it to a host. When the agent comes alive at the host, it will consult the TaskList to carry out the tasks sequentially in terms of the priority assigned. Such tasks could be as simple as giving an acknowledgement signal across the network back to its owner or as complicated as accessing the host database. If a task fails or cannot be carried out, perhaps because the network was down, the task can be delegated to a later stage and re-invoked when other tasks have been attempted.

The idea of having TaskList is to simulate certain limited 'intelligence' in the agent. Further enhancements could be carried out in possible ways like fine-tuning of the prescribed TaskList through parameters. This would require the parameters given to be quantified so that the TaskList can be adjusted based on the values given. A combination of varying parameters may then be used to achieve a greater degree of change.

4.2.3 Agent Communications Module

The agent has a communications module that allows the agent to communicate with the Agent Butler or other SAFER community entities. The Agent Communicator object establishes a link with an external communication entity and then transmits the information as a Message object. The Agent Listener listens at a specific port for an external connection request to initiate mutual messaging.

4.2.4 Agent Activity

When the agent reaches the host, it will be activated by the host. The activated agent can carry out a variety of pre-defined tasks while it is alive. It can communicate with the Agent Butler, access local product databases, process obtained information or return to the Agent Butler according to its TaskList. Every task would have an assigned priority and the required task parameters. For example, if the next task is to contact another agent, the ID and address of the other agent would be available. The TaskList (Figure 9) object is created and input by the Agent Butler to the agent before dispatch. The agent attempts to fulfil every task in a sequential manner.

The flexibility of such an approach is that different types of tasks can be created by the Agent Butler. The parameters of a type of task can be varied according to need. Each task is specified based on some pre-specified sub-tasks in a finer-grain level.

4.2.5 Agent Shopping

One of the primary possible tasks of an agent within the SAFER framework would be to roam to remote hosting sites with e-commerce retailers. This would allow the agent to carry out product information gathering. For example, an agent that has roamed to a shopping website would be able to access the local product database using its Information Processor module. After the information from the local database has been successfully retrieved, the agent can carry out further processing on the data. For example, the agent can attempt to match the retrieved product information with a shopping list that has been set at by the Agent Butler. When product matches are obtained, the prices of the matched products could then be analysed to find out if the price range is acceptable to the owner.

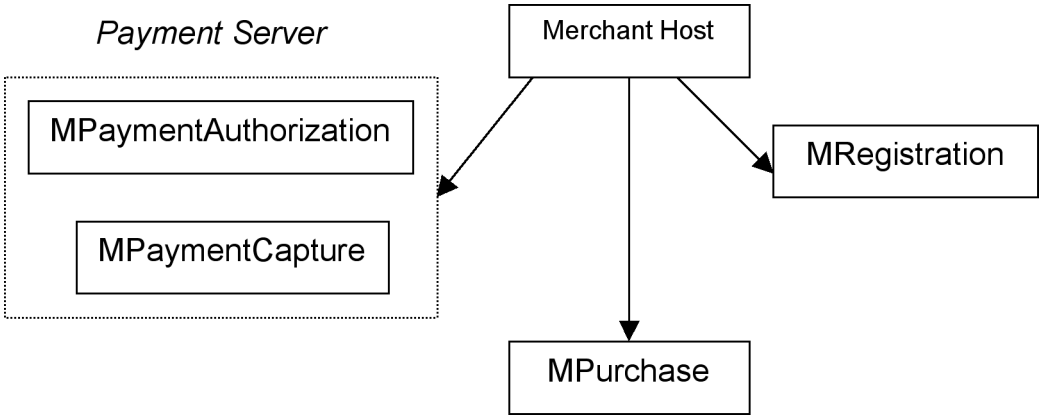

Figure10: Merchant Host Modules

4.3 Merchant Host

Merchant Host (Figure 10) represents any e-commerce retailer or website that allows online shopping. Like the Agent Butler, the host also has a set of payment objects that allow it to carry out transactions. For example, the MRegistration object carries out Merchant registration with a CA. The MPurchase object processes a purchase request from the Agent Butler. The MPaymentAuthorization and the MPaymentCapture objects allow the host to do payment authorization and payment capture respectively with the Payment Gateway.

4.4 Certificate Authority (CA) and Payment Gateway (PG)

Both CA (Figure 11) and the Payment Gateway are Java programs running on different machines separated from the Merchant Host or the Agent Butler. They each have server modules containing two server threads listening to different network ports. For CA, one thread would wait to service a

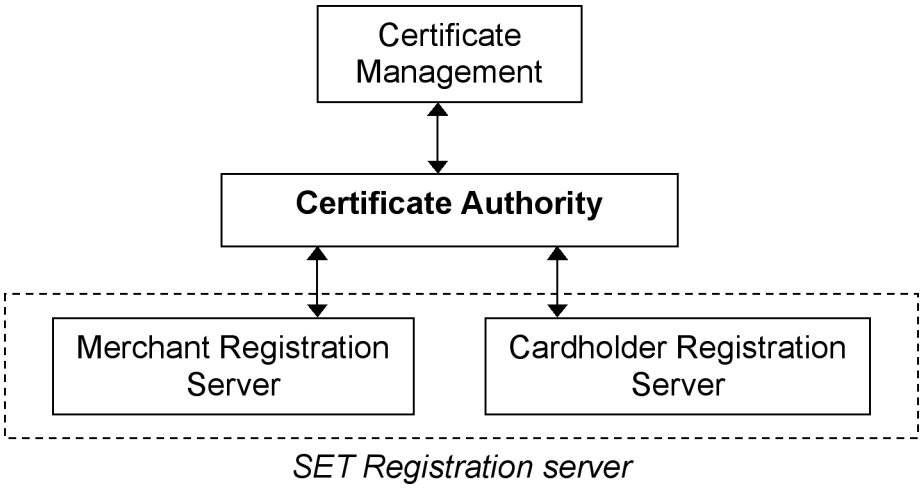

Figure11: Structure of CA

Cardholder registration request and the other would wait for a merchant requesting to register. The Payment Gateway waits for either a payment authorization or a payment capture request from a host. The separate threads allow simultaneous requests to be serviced at the same time and add to the robustness of these two servers.

Certificate Management under CA
The Cardholder and Merchant certificates as well as the CA key-exchange certificate and Payment Gateway certificate are generated by using Keytool with the RSA public-key algorithm. They were then stored into a KeyStore file. CA upon program execution will load these certificates from a file into a Java KeyStore object. This allows the certificates to be extracted quickly and transmitted during registration. The key-exchange private key will be extracted from the KeyStore object and used for public key cryptography during SET registrations with the Agent Butler or host. This functionality is included in the *Certificate Management* module of CA.

4.5 Registration and Purchase

When the registration commences, the registration module is activated by the Agent Butler's Financing Agency (Figures 8 and 20). The payment card account details are used to allow CA to authenticate the Cardholder. On successful registration, the Cardholder certificates are created by CA and sent to the Agent Butler. These certificates are stored for later use. After successful registration, purchase transactions can be carried out by the Purchase Module (Figures 8 and 20). The product information of e-commerce host received by the dispatched agents is used to select products. The selected items from different hosts are entered into a combined shopping list.

When a purchase is confirmed, the Agent Butler analyses the shopping list. It then simultaneously carries out purchase requests with all the hosts for items that have been ordered. Using SET-based payment as an example, the host(s) involved in the purchase transactions will invoke Payment Authorization request to the Payment Gateway when payment information is received from the Agent Butler. It stores the returned payment capture token. The host then proceeds to conclude the purchase request with the Agent Butler. At a suitable time, the host will then carry out the actual Payment Capture with the Payment Gateway using the previously stored capture token.

4.6 E-cash Payment

As shown in Figure 12, E-cash is realized by an E-cash object. *SerialNumber* is simulated as a randomly generated 50-digit numeric string. *Value* denotes the value that this E-cash object represents. *Signed* denotes whether E-cash has been certified by the bank server. *Expiration Date* denotes the expiration date of E-cash.

An electronic wallet class is implemented in a local environment, which could be used to manage, generate, and store E-cash. When the owner/Agent Butler needs some cash, the electronic wallet will be activated.

```
public class ECash extends Object
{   private String serialNumber;
    private double value;
    private boolean signed;
    private Date expirationDate;
 }
```

Figure 12: Sample Code of E-cash

```
NumberOfRules=n

Rule(n)Priority =
Rule(n)Factor =
Rule(n)Condition =
Rule(n)PaymentMethodName =
```

Figure 13: Rule Base Template

4.7 Automated Payment

To automate the payment process, we have incorporated a rule-based decision capability to automate the decision process of choosing a payment agent. A simple scheme is suggested in our architecture. A set of rules is defined in a rule-base for choosing a specific payment method under certain conditions. The template of a rule base is shown in Figure 13.

NumberOfRules specifies how many rules are defined in the rule base. In the template, each rule owns a unique ID, which is marked as " (n) " in the above figure. Additionally, each rule has four attributes, namely Priority, Factor, Condition, and PaymentMethodName. The meaning of each will be clear after we go through the following example.

We have incorporated a rule-based decision facility to automate the decision process of choosing a payment agent. A simple scheme is included in our architecture. A set of rules is defined in a rule base for choosing a specific payment method under certain conditions. Each rule has one factor that specifies the selection condition with certain priority denotation. Rules are validated in priority order. Once a rule is found valid, the corresponding payment method is chosen. A sample rule base is shown in Figure 14.

The rule base sample defines some rules of selecting a payment method. The first rule has the highest Priority 1. The decision factor is transaction amount. This rule is valid provided that the transaction amount is less than $50. The second rule has a lower Priority 2. The decision factor is transaction amount and trusted_merchant. This rule is valid provided that the transaction amount is less than $100 and the merchant is a trusted merchant. Agent Butler evaluates all the rules defined in the rule base in priority order. Agent Butler checks whether the condition of the first rule is met. If met, Agent Butler selects SET as the payment method. Otherwise, Agent Butler continues to evaluate the next rule. If all rules are invalid, Agent Butler can report to the owner and wait for his payment decision.

```
Rule1Priority=1
Rule1Factor= transact-amount
Rule1Condition= transact-amount < 50
Rule1PaymentMethodName= SET

Rule2Priority=2
Rule2Factor= transact-amount & trusted_merchant
Rule2Condition= (transact-amount  < 100)  &  (trusted_merchant  is
TRUE)
Rule2PaymentMethodName= ECash
… … …
```

Figure 14: Rule Base Sample

5. SYSTEM TESTING AND PERFORMANCE ANALYSIS

We have tested our implementation intensively. In this section, we give some samples of the system testing done. In addition, we also collect the performance data for payment transactions and include the performance analysis in the section as well.

5.1 System Testing

We planned our system testing based on the features we designed for our system. The objective of the system testing is to exercise each feature in our system under different conditions to allow the system to work properly. The following is the feature list against which we planned our system testing:

- Support SET protocol based credit-card payment transaction
- Support E-cash payment transaction
- Different payment methods are used based on the defined rule
- E-cash is generated and signed by the E-cash bank server provided there is not enough E-cash to complete the payment transaction

Based on the features listed above, we have come up with a list of test cases to cover different test scenarios. The following test cases are summarized below:

Test Case 1: To test if the Merchant or the Owner is able to register with Certificate Authority successfully during system startup.

Test Case 2: When two rules are both satisfied, the rule with a higher priority will be applied first.

Test Case 3: When more than one rule is satisfied and these rules are with the same priority, the rule with the smallest rule ID is applied first.

Test Case 4: When neither condition of two rules is satisfied, the default rule will be applied.

Test Case 5: When the transaction amount is beyond the authority given to Payment Manager, Payment transaction is pending for the Owner's approval.

Test Case 6: When the condition of E-cash payment method is valid, E-cash needs to be generated locally and signed by the E-cash bank server.

Each system test when carried out, performed just the way it was expected. The system achieved the objectives it was designed to do.

5.2 Performance Analysis

In addition to system testing, we did performance testing. The objectives of the test were: to measure how long it takes to complete a payment transaction and to analyze the system performance. We benchmarked the two types of payment methods implemented in our system: SET-based credit-card payment and E-cash payment. Each measurement of payment transaction started from a payment transaction initiation sent by payment agent to the Merchant and ended with the payment confirmation received from the Merchant by the payment agent.

Based on 10 trials for each payment scheme, the time costs were noted for both the SET-based credit-card payment and E-cash payment as shown in Table 1 below:

	Average Time (milliseconds)
Credit-card Based Payment	1307
E-cash payment	633

Table 1: Payment Schemes Performance Results

We noticed that the SET protocol based credit card payment method takes longer processing time than the E-cash payment method. However, this difference is reasonable and also expectable in our design. Most of the time costs were spent on message exchanges among different entities as well as the encryption/decryption processing.

The SET protocol aims to provide a more secure guarantee for electronic payment by specifically separating the communication only to related parties in certain stages of the payment process and encrypting all the messages exchanged among different entities. In the payment confirmation stage, the Owner, Merchant Host, Payment Gateway and Certificate Authority are all involved in message exchanges. In addition, the Payment Gateway and Certificate Authority are requested to validate the Owner's payment information (the Owner's account related information) before the Merchant can send out the payment confirmation to the SET payment agent. The whole process is time consuming.

In comparison, the E-cash payment method has a more simplified process. When the Merchant receives the E-cash notes, it only needs to contact the E-cash bank server to deposit the E-cash. The bank server will do the validation process. If all the E-cash notes are valid, the bank will send the Merchant a deposit confirmation, so that the Merchant can send the payment confirmation to the E-cash payment agent to complete the payment. The E-cash payment method is more efficient than the SET-based credit card payment method, which instead is more secure. However, since the E-cash bank server needs to validate the E-cash notes one by one, when there are a lot of E-cash notes used, the processing time for the E-cash payment method will increase to some extent.

Based on these facts and analysis, therefore, we highly recommend using the E-cash payment method for small-amount transactions in our design for efficiency and cost saving concern, and using the SET-based credit card payment method for large-amount transactions.

6. DISCUSSION AND EVALUATION

The payment module, for example SET payment, has a clearly defined interface with the Agent Butler through the interface module as seen in Figure 15. This enables the payment module to be independent of the Agent Butler. A new payment module can be easily plugged in. Such a need may arise when the parties involved in a transaction opt for a different payment scheme. Major modifications or disruptions to the system can thus be avoided. In addition, different payment schemes can easily be added into the framework as shown in Figure 16. This increases the versatility of the system.

The implemented agent does not have functionality or authority to carry out electronic payment transactions on its own. At this stage, it is still difficult to safeguard agents dispatched to external entities. Vital payment information is thus retained in the Agent Butler where it can be easily secured. All transactions are tightly controlled by the Agent Butler. If in the future, a certain level of integrity and secrecy can be achieved for mobile agents, payment functions would then be incorporated into the mobile agents.

In a mature e-commerce environment, users may not have the time to negotiate with individual retailers or surf the web to locate individuals' products. Agents dispatched by the Agent Butler will return information about the products available online. This information will be consolidated and presented to the user who then can interact with a single interface for all his possible shopping needs. There may be a problem with the possible compatibility and integration of these e-commerce websites when using agents. It is suggested that the use of an agent based virtual marketplace could be a stopgap solution before certain protocol standards could be imposed. The virtual marketplace, also under research, considered to be an integral part of SAFER application would also bring with it enhanced security features.

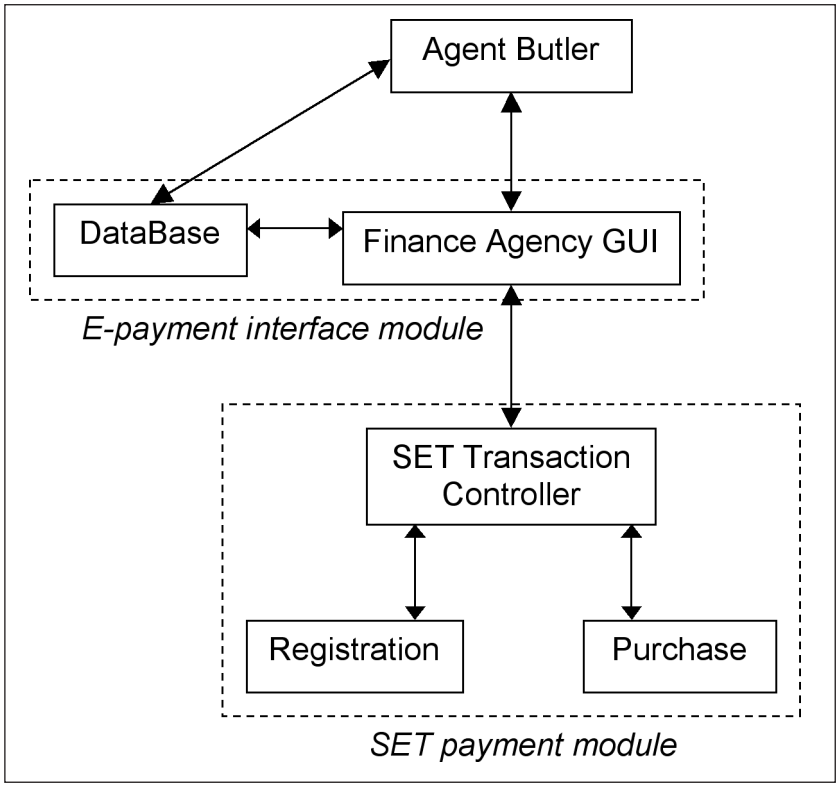

Figure 15: Payment Modules in the Agent Butler

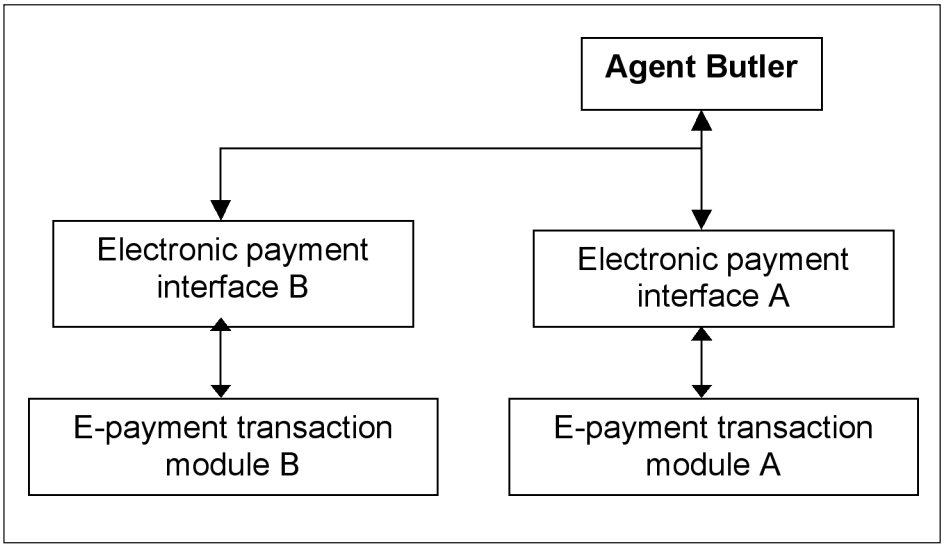

Figure 16: Addition of another E-payment Module

Table 2 is presented for the purpose of comparing the SAFER payment system with some related works.

	feature	SAFER feature
BABSy	not flexible to accommodate different payment schemes	flexible to accommodate different payment schemes
ABPS	scalability problem due to centralized architecture	avoids centralized architecture
ELEANOR	handles bank-to-bank transactions	handles business-to-consumer payment solutions
MPF	multiple payment capability on merchant side	multiple payment capability on consumer side

Table 2: Comparison of SAFER and Related Works

BABSy does not provide a flexible framework that allows more payment mechanisms to be added in future, since adding a new payment method requires modifying the whole user agent. In addition, this approach does not facilitate reusability, since all functionalities are encapsulated inside a single agent of each party.

ABPS is also centralized to some extent. Except for the payment agent in ABPS, software agents are not explicitly used by participants in their systems. The heavy burden of managing an ever-increasing knowledge base and the growing load for the single payment agent server would be a problem. Our payment scheme avoids a centralized architecture. Instead, we make use of cooperative multi-agents. Different types of agents are clearly defined and are embedded with certain functional modules as well as decision-making logic according to their roles in the system.

The focus of Eleanor is corporate users and financial institutions. It is more like a clearing-house, or a third party that handles bank-to-bank transactions. Our payment architecture is to provide business-to-consumer payment solutions.

The objective of MPF is to provide the capabilities to support multiple payment options for merchants. Therefore merchants in their system are able to deal with consumers who pay in a way that may be different from each other. Our payment architecture is to allow consumers to be able to use different payment methods to pay when they deal with different merchants. MPF and our payment architecture both address the issue of bridging different payment methods between merchants and consumers, but from different perspectives. MPF addresses the problem from the merchant's perspective by providing multiple payment capability on the merchant side. Our system addresses the problem from the consumer's perspective by providing multiple payment capabilities on the consumer side. These two systems should complement each other to provide the greatest flexibilities to all entities involved in e-commerce.

In terms of design, MPF has a similar approach as our agent based payment architecture. It has a modular design and some common interfaces. So different payment methods can be added easily. But their framework does not provide intelligence to choose the best payment option from the merchant's view. In contrast, our system has the capabilities to automatically choose the best payment option for the consumers by using agents based on defined rules.

7. SUMMARY AND CONCLUSION

This paper presents an extensible SAFER-based e-payment system suited to the requirements of agent-based e-commerce. The Secure Electronic Transaction (SET) and E-cash protocols were chosen as the payment schemes implemented. The prototype built illustrates a high degree of functionality. For instance, orders are made in a single interface window for products from different merchants. The clearly defined interfaces also facilitate the addition of new features in a single module without compromising reliability in other modules.

Additional developments of the system in the future could include the incorporation of agent security measures. Research has already been carried out in this area by other concurrent projects and the results could be used to enhance the current system. In addition, other electronic payment schemes can be implemented as additional payment modules to add to the flexibility of our framework for the convenience of users.

REFERENCES

AN OVERVIEW of Public-Key Encryption and Digital Signatures: http://www.hack.gr/users/dij/crypto/overview/publickey.html

BIGUS, J. P. (1998): Constructing intelligent agents with Java, John Wiley, 182–342.

BRANDS, S. (1995): Electronic cash on the internet. Network and distributed system security, *Proc. of the Symposium*, 64–84.

CHAVZ, A. and MAES, P. (1996): MIT Media Lab, Kashbah: An agent marketplace for buying and selling goods, *Proc. of 1st International Conference on the Practical Application of Intelligent Agents and Multi-Agent Technology*, London, UK, 1996.

GUAN, S.U. and YANG, Y. (2002): SAFE: Secure-roaming agents for e-commerce, *Computers & Industrial Engineering Journal*, Elsevier Science, 42: 481–493.

GUAN, S.U. and ZHU, F. (2002): Agent fabrication and its implementation for agent-based electronic commerce, *International Journal of Information Technology and Decision Making (IJITDM)*, ISSN: 0219-6220, 7(6): June.

GUTTMAN R.H. and MAES, P. (1998): Agent-mediated negotiation for retail electronic commerce, agent mediated electronic commerce, *Proc. of 1st International Workshop on Agent Mediated Electronic Trading*.

HO, D., CHIENG, I. (2000): A mobile agent brokering environment for the future open network marketplace", *Proc. of the 7th International Conference on Intelligence in Services and Networks, IS&N*, 3–15.

HUA, F. and GUAN, S.U. (2000): Agents and payment systems in e-commerce. In RAHMAN, S.M. and BIGNALL, R.J. (Eds.), Internet commerce and software agents: Cases, technologies and opportunities, IDEA Group Publishing, 317–330.

IBM WebSphere Payment Manager (1998): ftp://ftp.software.ibm.com/software/websphere/commerce/paymentsw/paymgr313install.pdf

IDENTRUS and its Project Eleanor (2003): http://www.identrus.com/services/eleanor.html

JCA/JCE Application Programming Interface Overview: http://www.openjce.org/docs/jce_api_overview.html

LOEB, L. (1998): *Secure electronic transactions: Introduction by technical reference*, Boston, Artech House.

McGRAW, G. and FELTON, E. (1997): Java Security, John Wiley.

MJOLSNES, S.F. and MICHELSEN, R. (1997): CAFÉ. Open transactional system for digital currency payment. *Proc. of the 13th Hawaii International Conference on System Sciences*, 5: 198–207.

NELSON, J. (1999): Programming mobile objects with Java, John Wiley, 466–469.

NG, C.H., GUAN, S.U. and ZHU, F. (2002): Virtual marketplace for agent-based electronic commerce, Book Chapter: Architectural issues of web-enabled electronic business, edited by NANSI, S., Idea Group Publishing.

NWANA, H.S. (1996): Software agents: An overview, *Knowledge Engineering Review* 2(3): 1–40.

POH, T.K. and GUAN, S.U. (2000): Internet-enabled smart card agent environment and applications, in the book: Internet commerce and software agents: Cases, technologies and opportunities, Idea Group Publishing.

ROCKINGER, R. and BAUMEISTER, H. (2000): BABSy: Basic agent framework billing system, *Proc.of the International ICSC Symposia on Multi-Agents and Mobile Agents in Virtual Organizations and E-Commerce (MAMA'2000)*, Wollongong, December.

RUSTY, H. E. (1997): Java network programming, O'Reilly, 242–261.

SIM, L.W. and GUAN, S.U. (2002): An agent-based architecture for product selection and evaluation under e-commerce, Book Chapter: Architectural issues of web-enabled electronic business, edited by NANSI, S., Idea Group Publishing.

The SET Standard Book 1 Business Description, http://www.setco.org/download.html/#spec

WANG, X.F. (1999): Secure agent-mediated auctionlike negotiation protocol for internet retail commerce, *Proc. of the 3rd International WorkShop*. CIA'99, 291–302.

WANG, T.H. and GUAN, S.U. (2000): An agent based auction services for electronic commerce, *Proc. of International ICSC Congress on Intelligent System & Applications*, CD #1524-045.

WANG, T., GUAN, S.U. and CHAN, T.K. (2002): Integrity protection for code-on-demand mobile agents in e-commerce, *Journal of Systems and Software*, 60(3): 211–221.
WONG, O. and LAU, R. (2000): Possibilistic reasoning for intelligent payment agents, *Proc. of the Second Workshop on AI in Electronic Commerce (AIEC)*, 1–13.
YANG, Y. and GUAN, S.U. (1999): Intelligent mobile agents for e-commerce: Security issues and agent transport, In RAHMAN, S.M. and RAISINGHANI, M. (Eds.), Electronic commerce: opportunities and challenges. Idea Group publishing, 321–336.
YEO, W.C., GUAN, S.U. and ZHU, F. (2002): An architecture for authentication and authorization of mobile agents in e-commerce, Book Chapter: Architectural issues of web-enabled electronic eusiness, edited by NANSI, S., Idea Group Publishing.
YOULL, J. (2001): Agent-based electronic commerce: Opportunities and challenges, *Proc. of the 5th International Symposium on Autonomous Decentralized System*, 146–148.
ZHU, F.M., GUAN, S.U. and YANG, Y. (2000): SAFER E-commerce: A new architecture for agent-based electronic commerce, in the book: Internet commerce and software agents: Cases, technologies and opportunities, Idea Group Publishing.

APPENDIX I: INTRODUCTION OF THE SET PAYMENT PROCEDURE

SET (Loeb, 1998) is an open specification system that enhances the existing payment card based schemes. The features of SET include *Cryptography*, *Verification* and *Authentication*.

The protocol involves three major stages, shown in Figure 17. In the first stage, both the Merchant and the Cardholder has to register separately with a trusted CA to obtain Merchant and Cardholder certificates respectively. Information such as unique identity code and acquirer/financial institute account information has to be provided for CA to verify. The possession of these certificates effectively authenticates their identities to any other parties who enter into a SET transaction with them. The sequence for Cardholder registration is shown in Figure 18.

SET Payment Procedure

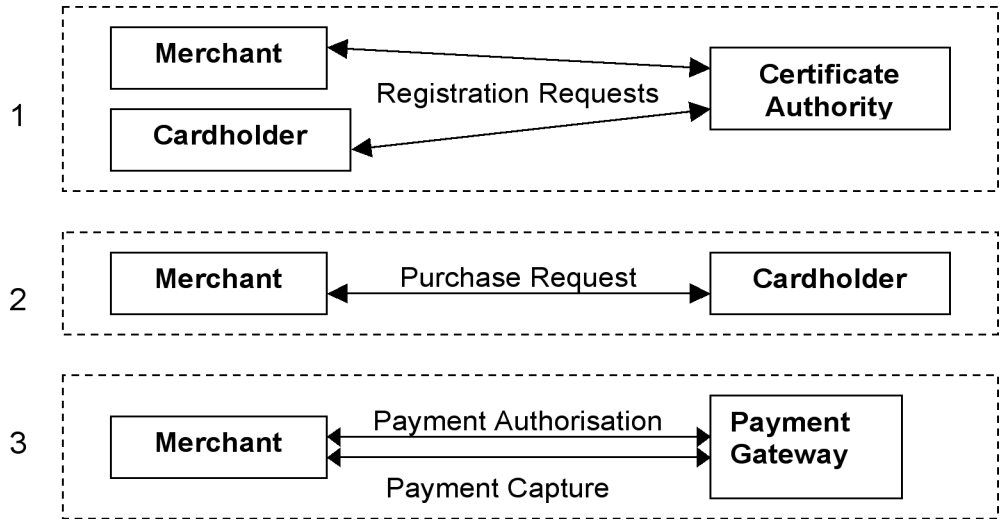

Figure 17: Steps Involved in SET

After the Cardholder shops at the Merchant's website, it initiates a purchase request to the Merchant (Figure 19). The two parties authenticate each other's identity by exchanging their SET certificates and the Cardholder transmits the encrypted order and payment information to the Merchant.

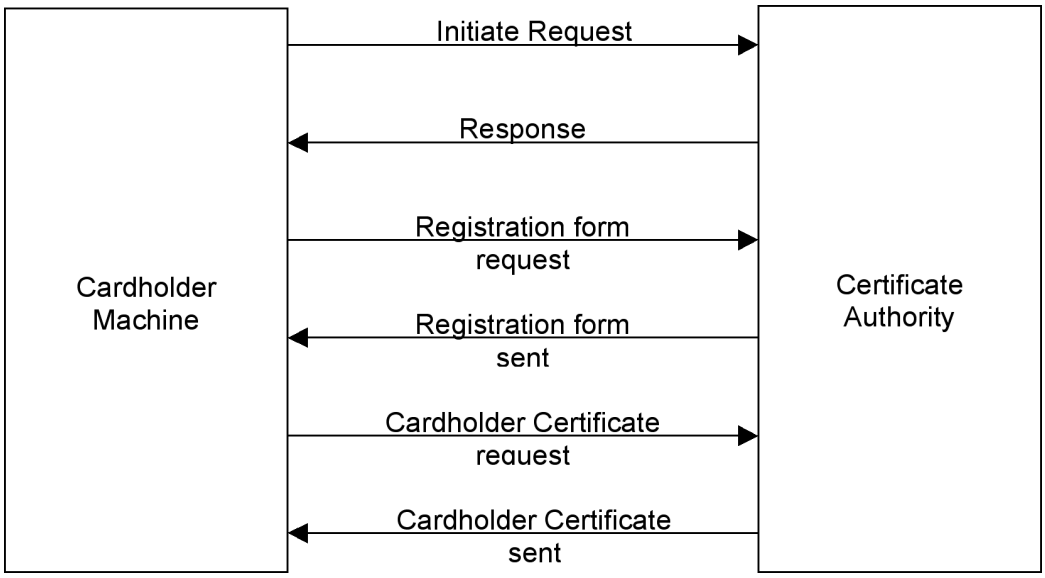

Figure 18: Cardholder Registration Process

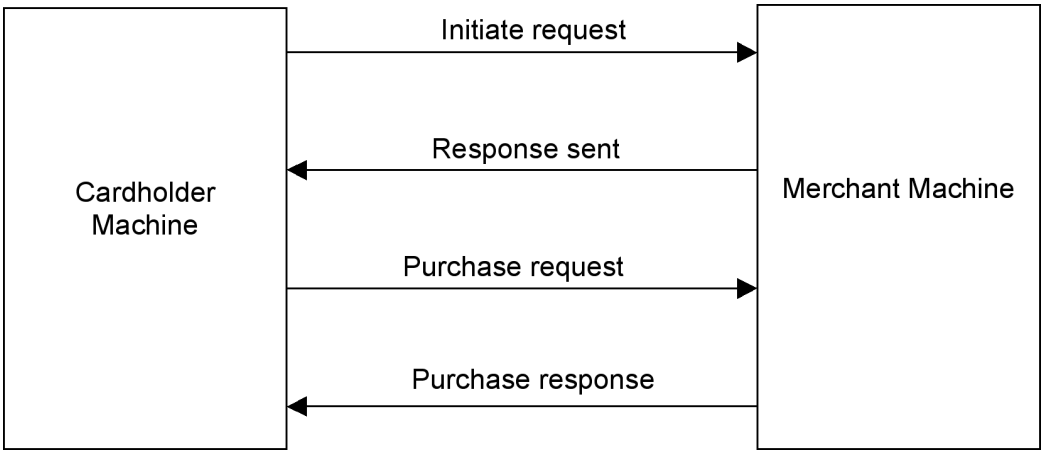

Figure 19: Purchase Request

The Merchant uses this payment information to make a payment authorization request to a Payment Gateway. If these payment instructions are approved, a *capture token* is sent to the Merchant. After completing the processing of an order, the Merchant can request the actual payment. The payment sum would usually be directly credited into the Merchant's bank account from the card Issuer. There would normally be a significant time lapse between Payment Authorization and Payment Capture in accordance to normal financial transaction procedures.

To initiate the Payment Capture process, a capture request would first have to be generated by the Merchant. This includes information such as the total payment amount, the transaction

identifier, etc. The request together with the *capture token* from the earlier Payment Authorization process are encrypted using the Payment Gateway's public key. When the Payment Gateway receives the capture request, it decrypts the request message and capture token. It verifies if both have consistent payment information, and then uses the information to format a clearing request that is sent to the card Issuer to carry out the actual credit transfer through financial networks.

APPENDIX II: THE AUTHENTICATION MODULE

The Authentication Module starts the authentication process when the user (or Agent Butler) selects an agent and a host (Merchant Host) destination. Before the agent can be transmitted to the host, there is a prior handshaking process by which the Agent Butler has to identify itself to the host. The certificate from the Community Administration Centre is sent along with the request message to ascertain that it is part of a trusted SAFER community. The host verifies the certificate and the digital signature on the message for authenticity and then evaluates the request. If approval is given, the Agent Butler proceeds to dispatch the agent to the remote host. The GUI of the Authentication Module is shown in Figure 21.

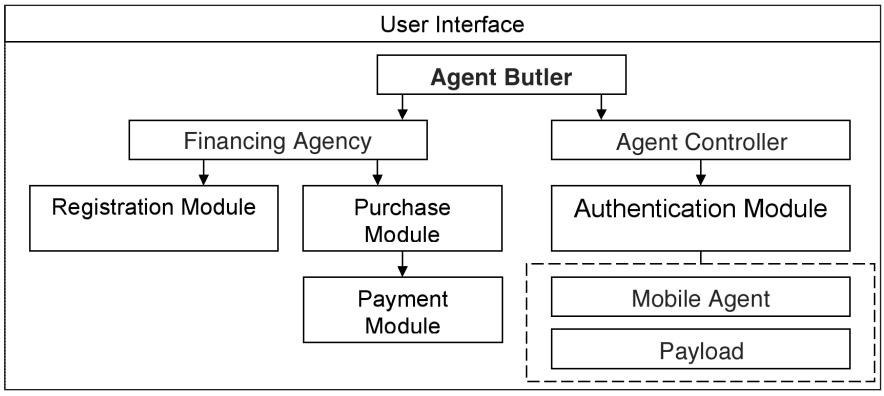

Figure 20: Authentication Modular Structure

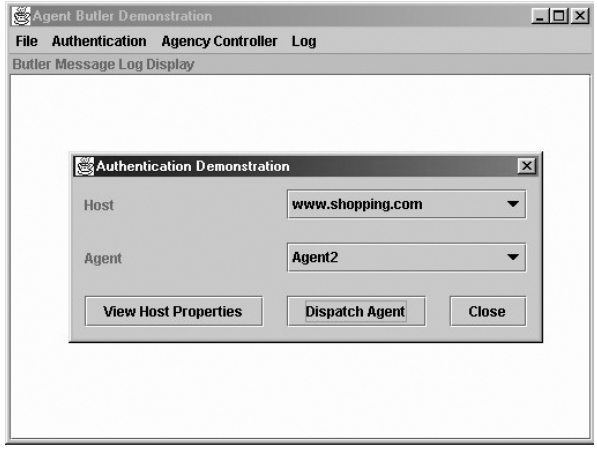

Figure 21: Agent Dispatch Option Dialog

APPENDIX III: MOBILE AGENT IMPLEMENTATION

The mobile agent implemented is in the form of a JAVA object that extends the Thread class and the Serializable interface. It is thus capable of being serialized in a byte-stream. This would enable the agent to be transported over existing TCP/IP networks to other machines that are capable of receiving the agent. A host machine can simply activate the agent by running it as a separate process thread.

BIOGRAPHICAL NOTES

Steven Guan

Steven Guan received his MSc and PhD from the University of North Carolina at Chapel Hill. He is currently with the Electrical and Computer Engineering Department at the National University of Singapore. Professor Guan worked in a prestigious R&D organization for several years, serving as a design engineer, project leader and manager. He has also served as a member on the ROC Information and Communication National Standard Draft Committee. After leaving industry, he joined Yuan-Ze University in Taiwan for three and half years. He served as deputy director for the Computing Center, and also as the chairman for the Department of Information and Communication Technology. Later he joined La Trobe University in Australia with the Department of Computer Science and Computer Engineering where he helped to create a new Multimedia Systems stream.

Feng Hua

Feng Hua received her BSc degree from Beijing Polytechnic University, China in 1999. In 2002, she received her MSc of Engineering from the National University of Singapore. Her research interests include electronic commerce, software agents, and secure electronic payment systems. She is currently with Hewlett Packard Pte Ltd, Singapore as an IT engineer.

Sin Lip Tan received his BSc degree from the National University of Singapore in 2002. His research interests include electronic commence and software agents. He is currently working in the IT industry.