SPEED Protocol: Smartcard-based Payment with Encrypted Electronic Delivery*

Antonio Ruiz[1], Gregorio Martínez[1], Oscar Cánovas[2], Antonio F. Gómez[1]

[1] Department of Information and Communications Engineering
[2] Department of Computer Engineering
University of Murcia, Spain
{arm, gregorio, skarmeta}@dif.um.es, ocanovas@ditec.um.es

Abstract In these times of the dawning of e-commerce, many issues and barriers still remain to be solved before electronic transactions over the Web can be expected to be really successful. One important unresolved problem is the issue of having efficient and secure payment models based on e-purses and including electronic product delivery and price negotiation. In response to this need, the SPEED protocol specification has been proposed. This specification, which is described in this paper, provides a high level of security for all parties involved in e-commerce transactions over the Internet; at the same time, we have combined this aim with the use of highly-recognised standards and all the advantages of using e-purses implemented on multiaplication smart cards. Our work has also been tested in a real environment, providing us an interesting feedback based on technical and user-friendly matters.

1 Introduction

Nowadays, we are under a real revolution concerning the way people as consumers purchase the goods and services they need and desire. The web exceptional growth has created a new way for commercial transactions, called e-commerce, involving some powerful and interesting possibilities such as global market coverage and personalized and direct interaction with customers.

Although the needed technology is already in place, e-commerce has not really grown as expected. In fact, some people were predicting an incredible benefit for year 2000 but it was not so high. The main reason for this reluctant behaviour is that most customers still consider that on-line financial transactions over the Internet are not secure enough. Some other reasons are the lack of efficient distribution systems for delivering individual customer orders, and the existence of few national and no international laws concerning legal issues such as copyright, taxation, and on-line agreements and contracts.

These barriers have been observed by a number of companies and research institutions with interests in secure e-commerce markets. Their answer to this situation was to present a number of secure payment alternatives based on electronic

* Partially supported by TEL-IFD97-1426 EU FEDER project (PISCIS)

money. Although some of these systems, such as PayWord [RS97], MicroMint [RS97], Millicent [G+95], NetBill [CTS95], etc. are secure and flexible enough, they have demonstrated to have many problems to reach real markets. The next list outlines some of the typical difficulties of most of the current proposals.

- The lack of pilot projects with real users
- Some proposals are not so light as intended
- There is no possibility for price negotiation (quite important in brokering markets)
- There is no perception of enough security by customers
- Electronic product delivery is not allowed
- There is no possibility of buying different products at the same time
- User mobility is not fully considered
- Some proposals are not based on recognised standards
- It is quite complicated to handle the change of electronic money
- The lack of proper arbitration (e.g., customer paid and did not receive the product)

Although some of theses problems are already solved by some previous schemas, to provide a common answer to all them in the specific environment we are working, we have defined a new payment system called SPEED which is intended to provide a smart card based stored value payment schema to be used with electronic products delivery.

It is also indicating the scope of this protocol. In fact, e-purses schemas are normally used for frequent and/or fast payments with lower-medium value. So, we have defined and implemented an efficient and secure schema to buy some products such as music, newspapers, keys providing access to pay-per-view shows (like movies and sports) etc.

This basic definition is complemented with some other characteristics of high interest. We have designed this protocol with a negotiation phase (that it can be omitted if not needed) really important in brokering markets, encrypted delivery, and using recognised standards like ASN.1 [ITU95] and PKCS#7 [RSA93].

Furthermore, every player in our system (customer, vendor, and broker) has a private key, normally stored in his or her smart card (it is mandatory for customers and depends on efficiency issues for vendors and brokers), and a X.509 [HFS99] certificate issued by a supporting public key infrastructure. This security information is considered as the trust foundation of the whole payment system.

The reminder of this paper is organised as follows. Section 2 describes the type of smart cards and e-purses we are using in SPEED. Section 3 discusses the SPEED protocol, while Section 4 describes its messages and existing operation modes. Section 5 discusses main security issues concerning this protocol. Section 6 describes the pilot project where SPEED has been defined and its currently used, and outlines some performance analysis. Finally, some conclusions and future work are provided.

2 Smart Card with E-Purse as a Basic Component of SPEED

Memory card based e-purses today seem to be only suitable for closed systems where the e-purse provider is also the service provider or where there is a very low fraud incentive. The reason is that defrauding such systems is relatively easy. By interposing itself between an actual card and a point of payment, a small computer may record the secrets communicated during an initial transaction and can then, as often as required, be used to play the role of a card having the initial balance and to pay as the real user. This is the main reason why we decide to rely our system on smart cards with e-purses.

In our case, the e-purse is implemented as a multicurrency and multiapplication pre-paid IC (Integrated Circuit) card containing digital money which the cardholder can spend in return for goods and services (such as music, books, newspapers, and so on) from retailers. This money is decremented using some specific SAM (Secure Access Module) security modules, which are normally handled by banks. These modules have a daily record of every economic transaction performed on those electronic purses. Periodically, retailers are reimbursed with real money from banks, which already obtained the money in advance from cardholders at the time of storing the corresponding value in their cards (prepayment concept).

These electronic purses are based on the WG10 [CEN96,CEN95] (also called "Inter-sector Electronic Purse") standard proposal defined by CEN (European Committee for Standardisation) which is made up of some of the most important European national standardisation bodies.

On the authentication process with the outside (i.e. the SAM modules) it is relevant to mention that our smart cards are able to run the 3DES symmetric cryptographic algorithm using prestored shared keys, as is described in the standard. A SAM security module uses the keys it shares with the card to authenticate messages during the decrement (i.e. purchase) or the increment transaction. This lets the SAM module convince the smart card that it is genuine. The card convinces the module by using the shared key to authenticate some information, like the amount to manage.

3 SPEED Overview

As commented before, the SPEED protocol is designed for purchasing and delivering electronically low and medium priced goods and services (MP3 songs, electronic documents, keys protecting video streams, etc.). The broker maintains accounts for vendors (and optionally for customers) and it hosts a set of SAM modules to perform decrement operations on the customer smart card e-purse. One SPEED transaction transfers electronic products from one vendor to one consumer, debiting the customer smart card e-purse (or account) and crediting the vendor account for the product price. SPEED is designed as a set of phases to support price negotiation, product delivery and payment.

There are two operation modes in the SPEED protocol: normal mode supports negotiation and it is designed with higher security capabilities (prevention of trivial denial of service attacks and full authentication of participating parties before the product delivery); the aggressive mode is composed by less messages than normal mode, and it is suitable for smaller products or scenarios with lower security requirements.

3.1 Players

The SPEED transaction model involves three main parties: the customer, the vendor, and the broker. Goods and services are delivered over the network, and SPEED links product delivery and payment into a single atomic transaction. The broker is not involved until the payment phase, when the customer submits a transaction request. Previously, the customer and the vendor can negotiate the product price in the negotiation phase.

Only registered users (customers and vendors) can interoperate using SPEED. Firstly, the customer and the vendor agree on the product to be purchased and its price. This can be accomplished after an optional offer and acceptance negotiation phase between the two parties. A customer receives the product he purchases if and only if he pays for the product. When the payment is performed, both customer and vendor obtain a proof of the transaction result (for example, for further complaints). The communication is protected from external entities using symmetric and, in some special cases, asymmetric cryptography.

SPEED assumes asymmetric trust relationships among the three participating entities. Brokers are assumed to be the most trustworthy, then vendors, and, finally, customers. Brokers play the role of serving as accounting intermediaries between customers and vendors. Vendors go into long-term relationships with brokers, in the same way as they would go into an agreement with a bank, credit card or company. Brokers tend to be large, well-known, and distinguished institutions or network service providers. The broker reputation is quite important for pulling toward customers; this reputation would be quickly loosed if customers or vendors have troubles with the broker. Vendor fraud consists of not providing correct products or descriptions. If this happens, customer will complain to the broker, and broker can drop vendors causing frequent complaints.

3.2 Registration

SPEED is designed considering that its participants are certified entities. It is assumed that both customers and vendors have a X.509v3 certificate issued by a trusted certification authority. The broker is also a certified authority, but it is not mandatory that the same certification authority certifies all the parties involved, although every certification authority must be considered trustworthy. Thus, a trusted CA must certify both customers and vendors before they initiate SPEED communications. As we will see below, real application environments show that this is not a problem in order to use SPEED in electronic markets or similar scenarios.

Although the smart card e-purse is the basic payment model (and the preferred one by the authors of this paper) the use of bank accounts is also allowed in the SPEED protocol. In order to make use of this information in the payment process, customer should also provide this data in the registration process.

3.3 SPEED Purchase Overview

Figure 1 shows a global communication scheme conforming the product negotiation, delivery, and payment in the normal operation mode. We can see that messages 1, 2 and 3 constitute the product negotiation phase, and that they are exchanged between the customer and the vendor. Message 4 contains the product ciphered by a symmetric key generated by the vendor, which will be provided to the customer once the payment has been performed (messages 5 and 6). The broker and the customer will exchange additional messages if a smart card based payment has been selected as the payment mode (dotted line).

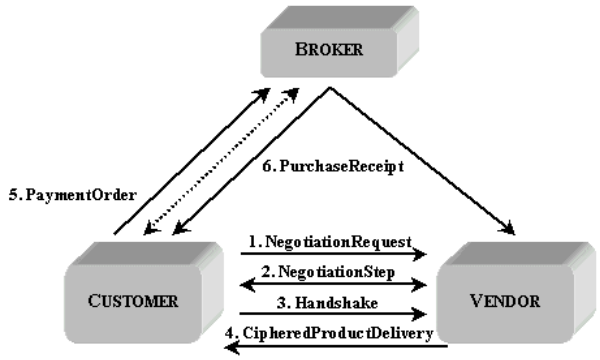

Figure 1. SPEED messages in the normal operation mode

4 SPEED Protocol Specification

4.1 Notation

In this section we will describe the notation used to specify the sequence of messages conforming the SPEED protocol.

$[Data]$	It indicates that this piece of data is optional, and it could not be in the message.
$Hash(Data)$	A message digest of *Data*, obtained using a hash algorithm such as MD5 [Riv92].
$\|Data\|_k$	*Data*, encrypted by a symmetric cipher using the key k.
$\|Data\|^k$	*Data* is authenticated using a HMAC algorithm with a cryptographic key k. This represents a message composed by two elements: Data and its cryptographic checksum.
$\|Data\|_k^{k'}$	This is equivalent to $\|\|Data\|^{k'}\|_k$
$\{Data\}_X$	Data, encrypted for X using public key cryptography (RSA). For computational efficiency, this is implemented using a digital envelop.
$\{Data\}_{X^{-1}}$	Data is signed using the RSA private key of X.
X $\Rightarrow$ Y	It indicates that X sends one message to Y.

In next sections we are going to describe all messages involved in the different phases. This explanation is divided in two blocks related to the normal and the aggressive operation modes.

4.2 Normal Mode

Generally, this operation mode is suitable in the next three scenarios:

- The customer does not know the product price because it is not fixed
- The customer wants to negotiate a lower price
- The product size, in bytes, is high (over 1 MB). As we will see in next sections, it could be very easy to perform a denial of service attack in the aggressive operation mode when the product size is high. For this reason, it is appropriate to use the normal mode if products are songs, videos, etc.

Negotiation Phase This phase is composed of three messages: *NegotiationRequest*, *NegotiationStep* and *Handshake*. First two steps represent a request/response message pair where the customer requests or proposes a price, and the vendor replies to this price. Then, these steps may be repeated as needed until the customer and vendor agree on a price, or any of the two parties decide to break the negotiation. Finally, the *Handshake* message is sent by the customer to notify the vendor that the price is accepted.

The *NegotiationRequest* message is sent the first time a customer and vendor are negotiating a particular product. This message is digitally signed using the customer private key, and asymmetrically encrypted using the vendor public key.

$$1 \quad \text{C} \Rightarrow \text{V} \ \textit{NegotiationRequest} \ \{\{NID, SeqN, ProductID, [Price], VendorID, EnKey, SignKey, Flag\}_{C^{-1}}\}_V$$

- *NID (Negotiation Identifier).* It is a 4 bytes value generated by the customer using a pseudo-random number generator. The NID identifies the negotiation being performed between the customer and the vendor. Although this

identifier is not globally unique, it is used to distinguish between the different negotiations performed by the same vendor
- *SeqN (Sequence Number).* During the negotiation phase, it is possible to exchange several messages. Every message in this sequence must be unique in order to prevent reply attacks. For this reason, a *SeqN* field is present in the negotiation messages and each party must increment the *SeqN* value after receiving this type of messages
- *ProductID (Product Identifier).* It is a string composed by a product code and an application-specific description, which the vendor uses to specify the goods
- *Price.* It is the price that the customer may be willing to pay for the item. It is optional in this message because the customer may not know about the product price. Price is a string specifying a value and the used currency (coded using ISO 4217)
- *VendorID (Vendor Identifier).* This field is the digest of the vendor public key. It identifies the intended vendor of this negotiation request, and binds the negotiation request to a particular vendor. In this way, we try to avoid any possible customer impersonation from malicious vendors (we shall analyse this attack in section 5.4)
- *EnKey (Encryption Key). EnKey* is a symmetric key that will be used to provide confidentiality to the following messages exchanged between the customer and the vendor. It is a 128 bits long value generated by the customer. The default symmetric cipher to employ is IDEA [Lai92]
- *SignKey (Signing Key). SignKey* is a symmetric key used to provide integrity to the subsequent messages exchanged between the customer and the vendor. It is a 128 bits long value generated by the customer. The default cryptographic checksum function to employ is HMAC-MD5 [CG97]
- *Flag.* This field contains a boolean value indicating whether the proposed price is the last customer offer (i.e., the customer do not want to negotiate other price)

Some of these fields will appear later in this document, although we will explain them again only when they have a different meaning.

Both customers and vendors send *NegotiationStep* messages in order to negotiate the final price of the product. It is protected using symmetric cryptography and the keys previously exchanged (the message is authenticated using a HMAC algorithm and then ciphered using a symmetric cipher). In this message, each party makes a bid that could change in following messages (it depends on the negotiation strategy).

2 V $\Rightarrow$ C *NegotiationStep* $|NID, SeqN, Price, Flag|_{EnKey}^{SignKey}$

- *Price* indicates the price proposed by the customer or vendor
- *Flag* codifies a boolean value that represents whether *Price* is the last offer

Both parties send the message in order to obtain the lowest (or the highest) product price. This exchange is repeated until one of the following conditions is reached:

1. The customer accepts the price proposed by the vendor in the last *NegotiationStep* message. In this case, the customer will send a *Handshake* message, as we will see below.
2. The customer or the vendor receive a *NegotiationStep* containing a last offer (flag is activated) that they do not agree with. In this case, the communication is closed.

Once the customer and the vendor have negotiated a price for the product, the *Handshake* message is sent to notify the vendor that the customer accepts his last offer. This message could have been sent after several negotiation steps (it indicates an active negotiation), or it could be the response to the first bid made by the vendor when the customer does not want to negotiate the price. The customer digitally signs the message (in this way the vendor cannot build a *Handshake* message using the shared secrets), and then this message is ciphered using *EnKey*.

3 C $\Rightarrow$ V *Handshake* $|\{NID, Price\}_{C^{-1}}|_{EnKey}$

Product Delivery Phase This phase is composed by one message named *CipheredProductDelivey*. When the vendor receives the *Handshake* message it proceeds to send the ciphered product. Then, the vendor constructs a message containing the product demanded by the customer, but ciphered with a pseudo-randomly generated symmetric key K. On the other hand, a bill is included in this message, which is digitally signed by the vendor, containing all the necessary information to uniquely identify this transaction. After that, this bill is ciphered using *EnKey*, and the message is sent to the customer.

4 V $\Rightarrow$ C *CipheredProductDelivery* $|Product|_K, |\{Bill\}_{V^{-1}}|_{EnKey}$

- $|Product|_K$ is the ciphered product demanded by the customer. Once the payment is done, the symmetric key that protects this product will be sent to the customer in order to decipher the product.
- **Bill**= $\{AccountNumber, K\}_B, Hash(|Product|_K), Hash(ProductID)$, PaymentOrderID, Price, RecKey
- *AccountNumber* is the vendor account number and K is the symmetric key ciphering the product. These fields are encrypted using the broker public key in order to maintain their confidentiality from the customer.
- $Hash(|Product|_K)$ is a message digest computed on the ciphered product, so the customer will detect any discrepancy before proceeding.
- $Hash(ProductID)$ is a message digest of the *ProductID* included in the first message. A hash (and not the complete identifier) is included in order to hide the product description from the broker. If the deciphered product did not match with the specified description, the customer would use this field to reclaim the correct product.

- *PaymentOrderID* is an identifier for the current transaction, and it must satisfy the condition of being unique in the whole system, considering all the transactions made by all the vendors. Two fields compose it: one is the message digest of the vendor public key, and the other is the NID value.
- *RecKey (Receipt Key)* is a symmetric key, 128 bits long, generated by the vendor. As we will see below, the broker will use this key to cipher the receipt.

Payment Phase The number of messages composing this phase depends on the payment mode. There are two main messages, which are present in both operation modes. The *PaymentOrder* message is submitted by the customer to the broker, and marks the point of no return for the customer. Once the payment has been performed, using some of the available methods, the broker returns a receipt (the *PurchaseReceipt* message) including the symmetric key protecting the product, and some information that could be used for further complaints.

5 C $\Rightarrow$ B $PaymentOrder\ \{\{\{Bill\}_{V^{-1}}, PaymentMode\}_{C^{-1}}\}_B$

- *Bill*. The customer includes in this message the bill previously received from the vendor in the *CipheredProductDelivery* message.
- *PaymentMode*. The structure of this field depends on the payment mode selected by the customer. We have the following options:
  - {*AccountNumber*}. If the customer selects the payment mode based on credit card number, this field represents the customer account number.
  - {*BEncKey, BSignKey, S1*}. As has been commented in previous sections, in order to perform a smart card based payment, it is needed to exchange a set of cryptographic messages between the card and a secure access module (hosted by the broker). Those two messages are part of the SPEED protocol, and therefore they must be protected like the other protocol messages. To do that, two symmetric keys are provided with the *PaymentOrder* message to authenticate (*BSignKey*) and to encrypt (*BEncKey*) the subsequent messages taking part of the decrement sequence. The *S1* field is necessary to authenticate the smart card and to do the payment.

When the smart card based payment mode is selected, two additional messages are needed in order to accomplish the payment. The content of these messages has been outlined in a previous section, and any further explanation will be omitted here.

a B $\Rightarrow$ C $|PaymentOrderID, S2|_{BEnKey}^{BSignKey}$
b C $\Rightarrow$ B $|PaymentOrderID, S3|_{BEnKey}^{BSignKey}$

Once the broker decrements the quantity specified in the bill from the smart card or the account number, it proceeds to transmit to the customer and the vendor the receipt containing the product price, the encryption key and the

PaymentOrderID concerning to this transaction. The broker digitally signs this message, and uses the *RecKey* contained in the bill to cipher it.

6 B $\Rightarrow$ C,V *PurchaseReceipt* $|\{PaymentOrderID, Price, K\}_{B^{-1}}|_{RecKey}$

When the customer receives the message, he tries to decipher the product using the symmetric key K. If the key is wrong, the customer will use the receipt and the product description to make a complaint.

4.3 Aggressive Mode

In this operation mode, there is no negotiation phase since the customer accepts the price proposed by the vendor, and receives the bill and the ciphered product immediately. This protocol mode is suitable for scenarios where the negotiation has no sense (the price is fixed) or those situations where the number of transactions per second is the main goal. However, it is not suitable if the product size is large (over 1 MB) since it can be considered as a security flaw to perform denial of service attacks.

This mode reduces the number of messages (four) eliminating the *NegotiationStep* and *Handshake* messages, and modifying the *NegotiationRequest* and the *CipheredProductDelivery* messages. The rest of messages are the same that we have seen in the normal mode section.

Product Request Phase The main differences between the *ProductRequest* message and the *NegotiationRequest* message (normal mode) are those fields related to the negotiation: the negotiation keys, the *flag* and the *SeqN* fields has been deleted since they are not necessaries.

1 C $\Rightarrow$ V *ProductRequest* $\{\{NID, ProductID, Price, VendorID\}_{C^{-1}}\}_V$

Product Delivery Phase The main difference between the *CipheredProductDelivery* of the aggressive mode and the one of the normal mode is that the former bill is encrypted using the customer public key since there are not symmetric keys exchanged in previous phases. However, the message contents are the same.

2 V $\Rightarrow$ C *CipheredProductDelivery* $|Product|_K, \{\{Bill\}_{V^{-1}}\}_C$

5 Security Analysis of the SPEED Protocol

This section provides a technical analysis of the cryptographic strength of the SPEED protocol, and it covers some other aspects related with vendor or customer frauds, complaints, and information visibility.

5.1 Assumptions about Cryptography

In general, our model assumes perfect cryptography. The following list explains what this assumption implies for all the cryptographic functions used in SPEED.

- **Opaque encryption**. Encryption is assumed to be opaque. If a message has the form $\{m\}_X$, only party X can learn m.
- **Unforgeable signatures**. Signatures are assumed to be unforgeable. Messages of the form $\{m\}_{X^{-1}}$ can only be generated by the party X. Anyone who has the verification key of X is able to verify that the message was indeed signed by X.
- **Collision-free hashes**. Hashes are assumed to be collision-free.
- **Trusted certificate authority**. There exists a trusted certificate authority (CA). All parties are assumed to have the verification key of the certification authority, and are able to verify messages signed by it.

5.2 General Objectives: Confidentiality and Authentication

The SPEED protocol encrypts all the information transmitted over the network. SPEED make use of both, asymmetric cryptography and symmetric encryption, in order to obtain a high performance in all transactions.

Symmetric encryption is used during the negotiation and the product delivery phases to protect the product and the bill, as well as in some other messages such as *PurchaseReceipt*. The rest of messages are encrypted using digital envelopes.

Short-term session keys are generated both for encryption and authentication, and they are transmitted using public key cryptography. 128 bits long keys and IDEA algorithm are used, which is currently considered strong enough to protect any private information. However, it is important that SPEED securely protect confidential data against even active attacks. Of course, underlying encryption algorithm should be secure against adaptive chosen plaintext / chosen ciphertext attacks, but this is not enough on its own. One important attack is cut-and-paste attack. SPEED uses the most comprehensive defense against this type of attacks, i.e., it uses strong authentication on all encrypted packets to prevent external modification of the ciphertext. Every message in the protocol is authenticated, and the way this is performed depends on the message, which can be authenticated by a digital signature or by a HMAC checksum.

5.3 Replay Attacks

The use of HMAC or digital signatures does not necessarily stop an adversary from replaying stale packets. There are some scenarios where this type of attack can obtain great benefits for an enemy:

- A malicious vendor could be interested in altering the contents of the negotiation messages exchanged between a particular customer and vendor. For example, the hostile vendor can capture the negotiation messages including

the first price offered by the right vendor, and replay them later to the customer, preventing a possible handshake. This situation is solved inserting a negotiation message sequence number in every message of this phase. This number is incremented by each party before the message is transmitted. A negotiation message with a repeated sequence number is interpreted as an active attack, and it will cause the end of the communication.

- The customer receives more than once the same bill. With this replay attack, the vendor tries to swindle the costumer selling several times the same negotiated product. The customer can avoid this attack keeping track of every paid bill, which is identified by a unique *PaymentOrderID*. The same benefit could be obtained presenting previous bills to the broker in order to force repeated payments. It is mandatory for broker to keep track of every processed transaction.
- A replay attack that could cause a denial of service is the repeated transmission of *Handshake* messages. If the vendor did not maintain a deliveries register, the attacker would force the reiterated transmission of large products, and therefore a performance loss of the attacked vendor. For this reason, vendors should keep track of every received *Handshake*.

5.4 Impersonation

Abadi and Needham postulated some basic engineering practices for cryptographic protocols in [AN96]. One of these principles is related to naming: "*if the identity of a principal is essential to the meaning of a message, it is prudent to mention the principals name explicitly in the message*". Impersonation attack tries to convince some protocol party that the communication is being performed only between entities A and B, although there is a third party participating in that communication (impersonating A or B). We have included the *VendorID* field in the *NegotiationRequest* and *ProductRequest* messages in order to avoid the following situation (M is the malicious vendor):

1	C $\Rightarrow$ M	*NegotiationRequest*	$\{\{NID, SeqN, ProductID, Keys, Flag\}_{C^{-1}}\}_M$
2	M $\Rightarrow$ V	*NegotiationRequest*	$\{\{NID, SeqN, ProductID, Keys, Flag\}_{C^{-1}}\}_V$
3	C $\Leftrightarrow$ M $\Leftrightarrow$ V	*NegotiationStep*	$\|NID, SeqN, Price, Flag\|_{EnKey}^{SignKey}$

In this scenario, the malicious vendor gains access to all the information exchanged during the negotiation phase (can learn the negotiation strategy of both parties, the product price, etc.), the customer ignores that he is not talking with he right vendor, and the honest vendor does not know that there is one man-in-the-middle. Using the *VendorID* field (an identifier of the intended vendor), any forwarded *NegotiationRequest* message can be interpreted as an attack.

5.5 Visibility

In a payment protocol, some of the exchanged data must be readable by only those parties needing this information to accomplish their tasks. For example, vendor account number should be protected from the customer, and vice versa, the product description should be hidden from the broker, etc. The SPEED protocol uses encryption and cryptographic checksums to protect this partially-confidential information from unintended readers. Vendor account number (contained in the bill) is encrypted using the broker public key; only a digest of the product description is inserted in the bill in order to hide it from the broker; and the customer never sends his account number (or some other information related with the smart card) to the vendor.

5.6 Product Delivery Attack

The information contained in the bill transmitted to the customer by the vendor can be different from the negotiated previously. Particularly, the vendor can be interested in changing the delivered product, or in modifying the price in order to get benefit from this change. The customer immediately detects any change in the product description (its hash) or in the product price, since the customer knows the product description (included in the first message) and its negotiated price (it is included in the *Handshake* message). Moreover, as we will see below, if the delivered product did not correspond with the provided description, the customer would possess all the information needed to demonstrate the fraud.

5.7 Customer Complaints

Despite the security mechanisms provided by the protocol, there are frauds that cannot be controlled as, for instance, the delivery of a product that does not match with its description. In this type of situations, the customer will make a complaint to obtain the required product.

- The delivered product does not match with its description. The customer does not receive the key protecting the product until the payment is done. Once the customer recovers this key, he deciphers the product and checks whether it matches with its description. If it does not match, he can present to the broker the product description exchanged in the first message (the broker has only the digest), and the deciphered product. In this way, the customer can demonstrate the vendor fraud.
- The customer does not receive the receipt containing the key. If some intruder had intercepted the *PurchaseReceipt* message, the customer would not be able to decipher the product. It is mandatory for the broker to maintain a register with the status of all the processed transactions in order to know whether it can retransmit the encryption key to the customer.

6 Using SPEED in a Real Environment: the PISCIS Project

The definition of projects and associated pilots on e-commerce marks a qualitative leap in the region or country where it is applied. In fact, we think that these projects are becoming more and more important to demonstrate final customers that related technology is mature enough.

In our case, the SPEED protocol has been designed and implemented as part of the PISCIS project. This project is intended to define an intelligent architecture to securely buy electronic products on the WEB. The communication base is a cable network provided by an important network service provider in our country.

To reach this objective, we have defined three main action lines as described now.

1. The design and implementation of a Java-based public key infrastructure, which provides all needed private keys and certificates to different players in our protocol: customers, vendors and broker (or brokers). The certification authority certificate, which it is distributed in off-line mode, is the trust foundation of the whole payment system
2. The design and implementation of a security module to access from final user applications to the information stored in smart cards. This module has been designed according to the Microsoft Windows security architecture [Cor01], developing a RSA Crypto Service Provider which implements the CryptoAPI 2.0
3. The design and implementation of the SPEED protocol, as described in this paper

The next subsection provides some more details on the implementation of the SPEED protocol in the environment defined by the PISCIS project.

6.1 SPEED Performance Analysis

We carried out the experiments using an Ethernet local area network. Computers running the customer software were Intel Pentium III 800 MHz processors with 128-MB main memory. Broker and server programs were run in Pentium II 450 MHz processors with 128-MB main memory. Windows 2000 professional edition was used as the operating system, and C++ language was used to develop the software. That software was compiled using the Visual C++ Studio 6.0 with the *maximize speed* optimization option activated.

SPEED was tested using product sizes of 16 bytes, 1MB, 3MB and 5MB. For each product size, we used up to five different computers simultaneously acting as customers. These customers buy products using the normal operation mode, and they accept the first price proposed by the vendor (only three messages constitute the negotiation phase). As Figure 2 shows, simultaneous customers are managed well. Purchase time is formed by two components: time concerning

the product delivery, and time related to the rest of messages. The product delivery message originates the differences among those purchases of the same product size, but time used to exchange the rest of messages remains almost independent of the number of concurrent customers (as is shown by the results obtained from the purchase of the 16 bytes long product).

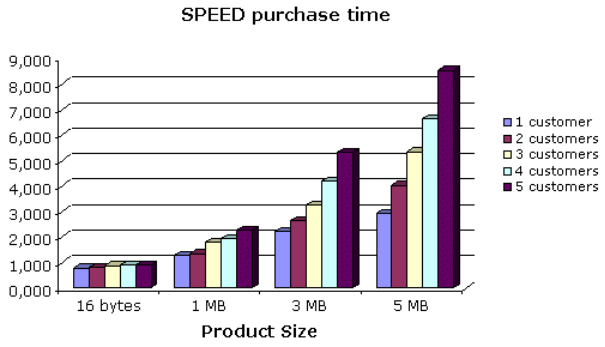

Figure 2. SPEED purchase time

7 Conclusions

This paper has presented one smart-card based payment schema, called SPEED, for making small and medium price purchases over an open communication system. SPEED stands for Smartcard-based Payment with Encrypted Electronic Delivery. This name is providing us some of the main characteristics of this protocol: the use of a smart card electronic purse, the use of encryption as the main technique for protecting products, and the use of an electronic delivery method for sending products from merchants to customers. Some other important characteristics are also coming to our design, such as, security, non-repudiation, user-friendly, price negotiation of one product (or even several ones simultaneously) between the two speakers, web-based implementation, and the use of several standard formats (ASN.1, PKCS#7, X.509, etc).

Our work is really intended to fill the lack of proposals in the line of electronic payment methods based on smart card e-purses, defining a secure schema able to perform a high-speed price negotiation.

We have this payment system running from some time ago in an e-commerce pilot defined together with an important cable network company. As we have shown, results are really hopeful, getting the possibility of completing 4 or 5 transactions per second (with just one medium PC server) when the customer is buying a key to access to one real-time stream of information (music, movies, football matches, etc.)

8 Future Work

The focus of our future research activity is based on payments by mobile phone. In fact, we think that electronic commerce by mobile devices or m-commerce is predicted to be the next step in e-commerce, making use of mobile phones or PDAs with networking capabilities as the easiest, most flexible, and widespread smart card based operating platform. In this line, we are planning the adaptation of the SPEED protocol to m-commerce scenarios making use of some new research lines like SIM Application Toolkit with Java Cards [Sun00], WIN modules [WAP00], and so on.

We are also planning to make use of SPKI [EFL+99] credentials in our protocol to prove group membership in order to ask for special discounts to merchants.

Finally, we are also working in a new e-purse standard called CEPS [CEP99] (Common Electronic Purse Specifications), which it is going to provide interoperability between different standard proposals. It is based on a core set of common technologies and functions, allowing e-purse to operate in a homogeneous and global way.

References

[AN96] M. Abadi and R. Needham. Prudent engineering practice for cryptographic protocols. *IEEE Transactions on Software Engineering*, 1(22):6–15, January 1996.

[CEN95] CEN/TC224/WG10. *Inter-sector Electronic Purse, Part 3: Data Elementents for Interchanges*, December 1995.

[CEN96] CEN/TC224/WG10. *Inter-sector Electronic Purse, Part 2: Security Architecture*, January 1996.

[CEP99] CEPSCO LLC. *Common Electronic Purse Specifications*, March 1999.

[CG97] P. Cheng and R. Glenn. *Tests Cases for HMAC-MD5 and HMAC-SHA-1*, September 1997. Request For Comments (RFC) 2202.

[Cor01] Microsoft Corporation. *CryptoAPI version 2.0.* World Wide Web, http://msdn.microsoft.com/library/psdk/crypto, 2001.

[CTS95] B. Cox, J. D. Tygar, and M. Sirbu. Netbill security and transaction protocol. In *Proceedings of First USENIX Workshop on Electronic Commerce*, 1995.

[EFL+99] C. Ellison, B. Frantz, B. Lampson, R. Rivest, B. Thomas, and T. Ylonen. *SPKI certificate theory*, September 1999. Request For Comments (RFC) 2693.

[G+95] S. Glassman et al. The Millicent protocol for inexpensive electronic commerce. *World Wide Web Journal, Fourth International World Wide Web Conference Proceedings*, pages 603–618, December 1995.

[HFS99] R. Housley, W. Ford, and D. Solo. *Internet Public Key Infrastructure, Part I: X.509 Certificate and CRL Profile*, January 1999. Request for Comments (RFC) 2459.

[ITU95] ITU-T. *ASN.1 encoding rules: Specification of Basic Encoding Rules (BER), Canonical Encoding Rules (CER) and Distinguished Encoding Rules (DER)*, 1995. Recommendation X.690.

[Lai92] X. Lai. *On the design and security of block ciphers*, volume 2. ETH Series in Information Processing, 1992.

[Riv92] R. L. Rivest. *The MD5 Message-Digest Algorithm*, April 1992. Request For Comments (RFC) 1321.

[RS97] R. L. Rivest and A. Shamir. Payword and MicroMint:two simple micropayment schemes. In Mark Lomas, editor, *Proceedings of 1996 International Workshop on Security Protocols*, number 1189 in Lecture Notes in Computer Science, pages 69–87. Springer, 1997.

[RSA93] RSA Laboratories,. *PKCS#7: Cryptographic Message Syntax Standard*, November 1993.

[Sun00] Sun Microsystems. *JavaCard 2.1.1 Specifications*, May 2000.

[WAP00] WAP Forum. *Wireless Application Protocol Identity Module Specification*, February 2000.