A Novel Peer-to-Peer Payment Protocol

Despoina Palaka[1], Petros Daras[1,2], Kosmas Petridis[2], and Michael G. Strintzis[1,2]
(Corresponding author: Michael G. Strintzis)

Information Processing Laboratory, Electrical and Computer Engineer Dept., Aristotle University of Thessaloniki[1]
Thessaloniki 541 24, Greece (Email: {palaka, daras, kosmas}@iti.gr)
Informatics and Telematics Institute[2]
1st km Thermi-Panorama Road, Thermi, Thessaloniki 57001, Greece (Email: strintzi@eng.auth.gr)

(Received July 5, 2005; revised and accepted Sept. 12, 2005)

Abstract

In this paper a novel electronic payment protocol suitable for "peer-to-peer" (P2P) networks is presented. It implements electronic cash-based transactions, between buyers and merchants. It is based on a bank account, though it can be easily extended and can be readily applied to other account payment models like debit cards. The proposed protocol is designed using Millicent's main concept (scrip) and the digital envelope cryptography technique. In this protocol, financial institutions become partners in the e-commerce transaction, conducted by their customers over the Internet. The innovation of the proposed protocol is the reduction of the involvement of the financial institutions to ancillary support services like helping on establishing trust between the parties and at the completion of the peer-to-peer payment transaction. Moreover, the proposed system can be characterized as distributed allocation of provinces to merchants, who are responsible for locally authorizing payments. Finally, it is optimized for repeated payments to the same merchants.

Keywords: P2P networks, payment protocol, and micro-payments

1 Introduction

The worldwide proliferation of the Internet has led to the birth of electronic commerce, a business environment that allows the transfer of electronic payments as well as transactional information via the Internet. Electronic commerce flourishes due to the openness, speed, anonymity, digitization and global accessibility characteristics of the Internet.

At the turn of the century over 70 million computers were connected to the Internet [12]. Successful electronic business sites like Amazon.com [1] or ebay [7] had foreseen the business potential of the huge number of users and offer world-wide services to consumers for buying and selling goods using their web browsers. These business sites provide a centralized trading platform, which offers a certain degree of security to its customers. The advantage of such a centralized architecture is that rules can be enforced easily. However, this turns into a severe problem if we switch the point of view: In any centralized architecture the central entity is a single point of failure and a bottleneck in terms of bandwidth and computing recourses which limits scalability and in turn causes high infrastructure requirements.

Furthermore, this kind of architecture is not suitable for small companies or small merchants that cannot afford a high infrastructure. This is where the peer-to-peer (P2P) [3, 18, 20] architecture comes to give the solution. The P2P computing scheme is increasingly receiving attention as a new distributed computing paradigm for its potential to harness "edge" computers, such as PCs and handheld devices, and make their underutilized resources available to each other. Scalability and fault-tolerance come implicitly with P2P infrastructures, as has been proven by successful P2P systems like Kazaa [13] or Gnutella [11].

The new P2P networking paradigm offers new possibilities for electronic commerce. A major differentiating factor of P2P from traditional electronic commerce models is the reduction of the competence of the financial institutions [21].Evenmore customer peers interchange roles with merchant peers setting this new network economy perfect for example for an electronic market where users sell second hand products, in this example each user can act as both merchant and client using only his/her PC for doing business.

In this paper a new electronic-payment protocol is defined, able to exploit the capabilities offered by P2P networks. The new protocol provides a completely anonymous, secure and practical framework, in which each peer can act both as a merchant and a customer. Further, the proposed peer-to-peer protocol provides a full and secure payment mechanism where personal information and order information cannot be exposed to unauthorized third

parties. This protocol is actually a combination of SET's [15] digital envelope technique and the scrip of Millicent [10].

In SET, message data is encrypted using a randomly generated key that is further encrypted using the recipient's public key. This is referred to as the "digital envelope" of the message and is sent to the recipient along with the encrypted message. The recipient decrypts the digital envelope using his/her private key and unlocks the original message using the symmetric key. The proposed peer-to-peer protocol uses the concept of the "digital envelope" in securing all sensitive information exchanged between all parties of the transaction. The "digital envelope" or "session-key encryption" [16] technique speeds up the encryption [9]; only a small amount of data (the symmetric key) is encrypted using asymmetric encryption (asymmetric encryption is about 1000 times slower than symmetric encryption). Further, the digital envelope technique helps in retaining the public and private key pair resistant to cryptanalysis [9].

On the other hand Millicent offers anonymity, privacy and authenticity [14]. Even more the scrip of Millicent cannot be spent twice because of it's serial number. It's "Certificate" prevents tampering and counterfeiting. It can only spend by it's owner and it has a value only for a specific merchant. And finally it can be produced "on the fly", so there is no need to create it and save it in a big database.

Combining the P2P characteristics with the electronic commerce, many companies are promoting new services via this new infrastructure (Trymedia Systems, Lightshare, PinPost, Center-Span, First peer). All these companies claim to support P2P commerce, by using e-mails or SSL (Secure Socket Layer) [8] for the purchase transaction. SSL is the de facto standard for secure (i.e., encrypted and integrity-protected) communication on the web and it is integrated in almost all web browsers and servers. SSL uses asymmetric encryption but typically only the merchants have public-keys, while the customers are anonymous. Encrypting bank account data with SSL is certainly better than sending them in the clear, but the gain in payment security is very limited:

- Regarding the broker, the use of SSL is completely transparent since no messages are signed, thus the merchant does not gain any security.
- SSL does not hide bank account numbers or any other information from the merchant. Thus, it cannot be used in ID-based authorization.
- Unlike SET or proposed peer-to-peer protocol, SSL does not mandate any specific public-key infrastructure. Thus, there is no guarantee that a customer can verify the merchant's public-key.
- In SSL, merchants and brokers need additional mechanisms (beyond SSL) to transmit bank account data and authorization information.

Additionally, another P2P payment protocol is PPay [24]. PPay is a micro-payments, offline protocol that uses floating, self-managed coins. In this protocol security is sacrificed to reduce the brokers involvement and as a result the brokers load. Though a significant improvement of the systems performance is achieved this protocol is inappropriate for medium and large payments as the proposed peer-to-peer protocol. The secure version of PPay is called WhoPay [23]. WhoPay provides a secure infrastructure for electronic commerce and anonymity between the parties involved in a transaction, though it requires a big database for storing the scripts and does not consider that the P2P environment is an environment of unstable connectivity [20]. PPay's and WhoPay's scalability is based on the domination of the system by the transactions of transfer and renewal of scrips. These transactions require the presence not only of the two parties doing business but also of a third party that "substitutes" the broker. If this party is offline the broker is the one that has to take part to the transaction, so in this case the broker's load is increased. In examples like the one of an electronic market, the scenario of peer customers entering in the system ocassionally for buying goods is most probably and so it makes this protocol unsuitable.

In this paper a new electronic-payment protocol is defined, in this protocol three parties are involved: the customer (who makes the actual payment), the merchant (who receives the payment) and the acquirer gateway (that acts as an intermediary between the electronic payment world and the existing payment infrastructure and authorizes transactions by using the latter). Hereafter, the acquirer gateway will be addressed as simply "the broker". This broker, is used to "bless" the transactions and to enable a trust relationship between the parties, introduces the problem of "single point failure". This problem is typical in any client/server payment system, but the role of the broker is essential for security and financial reasons and it cannot be omitted. In the proposed protocol the broker's participation in the transactions has been minimized in order to minimize the effect of the problem that s/he introduces.

The remainder of the paper is organized as follows. In the following Section a short description of the parties involved in the payment processes along with some basic definitions and notation, are given. A mechanism regarding the users' registration and the exchange of public keys is presented in Section 3. Some security threats and adversaries as well as the security requirements of each party, are described in Sections 4 and 5. The payment process is presented in Section 6. The computational cost of the broker is addressed in Section 7. Finally, conclusions are drawn in Section 8.

2 Definitions

2.1 Parties

The proposed peer-to-peer payment protocol deals with the payment transaction and involves only three parties: C-Customer, M-Merchant, and B-Broker (gateway). Recall that B is not the acquirer/issuer in the financial sense, but a gateway to the existing bank network. In other words, the function of B is to serve as a front-end to the current infrastructure that remains unchanged. The payment system is operated by a payment system provider that maintains a fixed business relationship with a number of banks. Banks act as issuers to customers, and/or as acquirers of payment records from merchants. It is assumed that each customer/merchant (vendor) is somehow assigned (or selects) a PIN (PANSecret). A customer/merchant (user) can obtain her/his PANSecret by physical attending in the financial institution.

2.2 Protocol Definitions

The following terms are used for the description of the protocol:
PAN: is the bank account number.
PANSecret: is the combination of two secrets: The secret of the broker and the secret of the Peer Customer/Peer Merchant. Both the broker and the customer/merchant have this combination.
ID: is a unique identifier for the peer customer/merchant and it can certify his/her identity. It is the digest: Hash(PANSecret|Hash(PANSecret)|PAN)
UserID: is a unique identifier for each peer (user) and it does not provide any information about the identity of the user.
BrokerScrip: is electronic cash produced by the broker(bank).
VendorScrip: is electronic cash produced by a merchant (vendor) and it can be spent only to him/her.
ScripBody: consists of the following fields (Figure 1):

- ProducerID: is a unique identifier for the broker/merchant.
- Value: is the amount of the scrip.
- ScripID: is an identifier of the Scrip. Part of it is used to specify the MasterScripSecret (see definition below).
- CustID: is an identifier of the customer. Part of it is used to specify the MasterCustomerSecret (see definition below).
- Expires: is the expiration date

MasterScripSecret: is the look-up value of the ScripID. It is used to produce the certificate (see definition below).
Certificate: is the signature of the scrip (Figure 2) (The term "Certificate" is used with respect to the Millicents one.) It is used to verify that the scrip is valid. It is produced by hashing the concatenation of the ScripBody and the MasterScripSecret: Hash (ScripBody|MasterScripSecret).

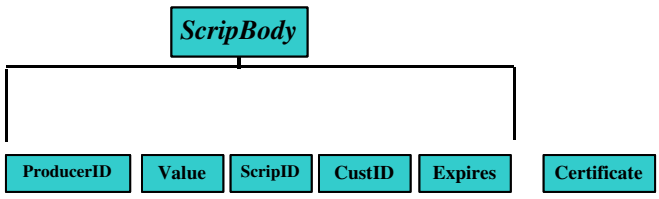


Figure 1: Scrip

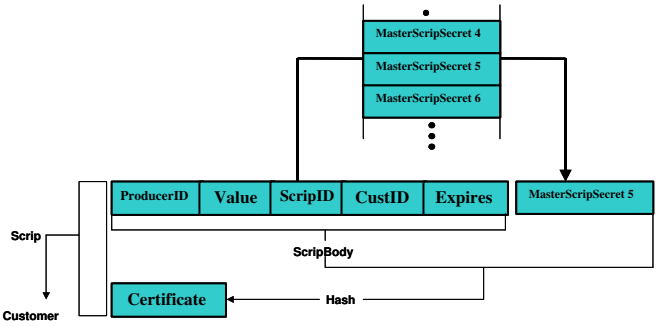


Figure 2: Certificate creation and Scrip's structure completion

MasterCustomerSecret: is the look-up value of the CustID. It is used to produce the CustomerSecret.
CustomerSecret: is used to prove ownership of the scrip. It is produced by hashing the concatenation of the CustID and the MasterCustomerSecret: Hash(CustID|MasterCustomerSecret) (Figure 3).

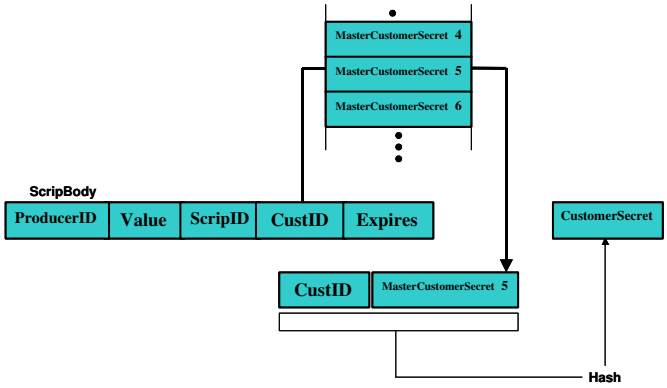


Figure 3: CustomerSecret creation

Table 1: Cryptographic primitives

K_A	is a 192-bits long, symmetric key
K_{Pr}	is a 1024-bits long, private (asymmetric) key
K_{Pu}	is a 1024-bits long, public (asymmetric) key
$Enc_{K_A}(.)$	Symmetric encryption using the AES (Rijndael) algorithm
$Sign_{K_{Pr}}(.)$	Digital signature that users the SHA1 algorithm for hashing and the RSA algorithm for encrypting
$SignOnly_{K_{Pr}}(.)$	Asymmetric encryption (using the RSA algorithm) of a message digest produced by the SHA1 algorithm
$Enc_{K_A}(SignOnly_{K_{Pr}}(.))$	Symmetric encryption (using the Rijndael algorithm) of the cipher-text produced by the $SignOnly_{K_{Pr}}(.)$ function
$PKEnc_{K_{Pu}}(.)$	Asymmetric encryption using the RSA algorithm
X, Y	X is concatenated with Y

2.3 Notation

In (Table 1) the notation of cryptographic primitives used in the protocol is presented, while in (Table 2) the notation of the basic message elements used in the payment protocol is shown.

3 Public Keys

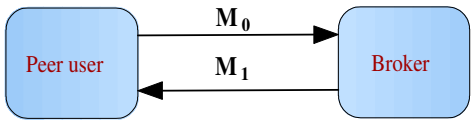
The proposed peer-to-peer payment protocol is based on public key cryptography, thus a mechanism is needed so as to authenticate the public keys. For this reason a certification authority (CA) is assumed that has a private key and the other parties involved hold its public counterpart. For the sake of simplicity, it is assumed in the rest of the paper that there is a single certification authority and that is the broker.

In the proposed peer-to-peer payment protocol, the broker B has a private key, which enables signing and encryption. Its public counterpart, that enables signature verification and encryption, is held by each accredited customer/merchant. As in current operation, the broker that stores (in a database) the customers'/merchants' PANSecrets and receives their IDs, is trusted to all parties involved, keeping these secrets confidential.

3.1 Peer Registration

When a peer user (customer/merchant) requests to open a bank account, his personal information (bank account number and PANSecret) is stored to the broker's database. Further, prior to the "peer-to-peer" protocol's initiation, a pair of keys is generated (public and private key)and stored locally in the user's file system. Moreover, the broker requires the user ID and the public key of the peer user, in order to complete his/her registration to the database. This specific information is sent to the broker through the "Peer registration" transaction step (Figure 4).

In M_0, the ID of the user (which is known only to him/her) and the signature, prove to the broker that the user authorized the transaction. Moreover, session key encryption ensures the customer about the confidentiality of the transmitted information. Further, the broker that receives this message retrieves the user's information and checks if the user is already registered (this is done to detect any replay attacks). If the user is not registered and the data in the received message (M_0) is valid, the broker stores the user's information in the database and then sends M_1 to the user to inform him/her that the transaction was successfully completed.



C_0	*Registration request*
X_0	PAN
M_0	**C_0, UID_P, $Enc_{K0}(Sign_{KP}(X_0))$, $Enc_{K0}(SignOnly_{KP}(ID_C), K_P)$, $PKEnc_{KB}(K_0)$**
C_1	*Registration response*
X_1	C_1, UID_B, I
M_1	**X_1**

Figure 4: Peer registration

In M_1 the broker's signature ensures the user that the broker authorized the transaction.

In Figure 4 the peer user forms the following message elements:

C_0 = Registration request

UID_P = the peer's user ID

PAN = the peer's bank account number

ID_C = the peer's ID

Table 2: Notation of some basic message elements

C_i	Lable of the message
UID_i	Unique identifier of the peer user
W_t	Value of the BrokerScrip, VendorScrip or product
N	Random generated nonce
ID_i	Unique identifier of the customer's or merchant's bank account
B_j	BrokerScrip
V_j	VendorScrip
CS_t	BrokerScrip's or VendorScrip's corresponding CustomerSecret
R	Authorization message, R="OK" or "NOK"
OI	Order information consisting of the product's name, price, quantity and a unique identifier
I	Information message

K_{Pu} = the peer's public key

K_{Pr} = the peer's private key

K_B = the broker's public key

K_0 = a random generated symmetric key

X_0 = PAN

creates the following message and sends it to the broker:

$$M_0 = C_0, UID_P, Enc_{K_0}(Sign_{K_{Pr}}(X_0)), Enc_{K_0}(SignOnly_{K_{Pr}}(ID_C)), Enc_{K_0}(K_{Pu}), PKEnc_{K_B}(K_0).$$

The broker receives the message, retrieves the user's information and checks if the user is already registered (this is done to detect any replay attacks). If the user is not registered the broker decrypts the message and verifies the message's data. If the received data is valid, the broker forms:

C_1 = Registration Response

UID_B = the broker's user ID

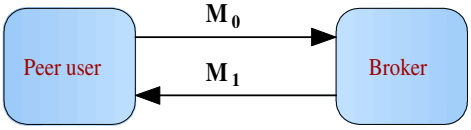
I = Information message

$X_1 = C_1, UID_B$, I

creates the message: $M_1 = X_1$ and sends it to the customer.

3.2 Public Key Request

All messages exchanged in the "peer-to-peer" protocol are asymmetrically encrypted, thus each peer user (customer/merchant) requires, besides the broker's public key, the public key of the third party (merchant/customer) involved in the payment transaction. The public keys of the legitimate (registered) peer users are stored in the broker's database, thus in order to enable payments with another



C_0	*Public key request*
X_0	UID_P
M_0	**C_0, UID_R, Sign $_{KR}(X_0)$**
C_1	*Public key response*
X_1	UID_P, Kp
M_1	**C_1, UID_B, Enc $_{K1}$(Sign $_{KB}(X_1)$), PKEnc $_{KR}(K_1)$**

Figure 5: Public key request

peer, each peer user must request/obtain from the broker the peer's public key (Figure5).

In M_0, the user's signature provides proof to the broker that the user authorized the transaction. Further, if the signature of the message is valid, the broker queries the database and retrieves the requested public key. Then, s/he sends this key to the user who requested it in a new message M_1.

In M_1, the broker's digital signature ensures the user that the received public key is not folly. Further, the message's encryption ensures confidentiality of the information sent.

4 Adversaries and Threats

Three different adversaries are considered:

1) Eavesdropper: listens to messages and tries to learn secrets (e.g., bank account numbers, PANSecrets, IDs).

2) Active Attacker: introduces forged messages in an attempt to cause the system to misbehave.

3) Insider: is either a legitimate party or one who learns the party's secrets. (One example is a dishonest merchant who tries to get paid without the customer's authorization).

Internet is a heterogeneous network, without single ownership of the network resources and functions. In particular, one cannot exclude the possibility that messages between the legitimate parties would pass through a maliciously controlled computer. Furthermore, the routing mechanisms in the Internet are not designed to protect against malicious attacks. Therefore, neither confidentiality nor authentication for messages sent over the Internet can be assumed, unless proper cryptographic mechanisms are employed.

Additionally, one must be concerned about the trustworthiness of the merchants providing Internet service. The kind of business is expected in the Internet, includes the so-called cottage industry-small merchants. It is very easy for an adversary to set up a shop and put up a fake electronic storefront in order to get customers' secrets ([22]). This implies that the IDs of the customers' should travel from customer to broker without being revealed to the merchant (who needs only an authorization message from the broker in order to complete the transaction).

Finally, three possible attacks by customers or adversaries are also considered, namely double-spending, faulty scrip attack and scrip forgery. Double spending involves spending scrip more than once, faulty scrip attack involves creation of scrip without the correct structure and scrip forgery attack involves forging the scrip's data.

1) Double Spending: as already mentioned, scrip is concatenated with two secrets the MasterScripSecret and the MasterCustomerSecret. These secrets are known only to the producer of the scrip. Each time a scrip is used, its secrets are deleted from the producer's look up tables, ensuring that the scrip cannot be reused in another transaction.

2) Faulty Scrip: each user of the payment protocol can act both as merchant and customer and s/he is able to produce scrip, but this scrip can only be used to authorize payments with its producer (the scrip carries the Producer's ID (Figure 1)).

3) Scrip Forgery: scrip consists of the scrip body, which contains the information of the scrip and a certificate, which is the signature of the scrip. Any alteration of the information contained in the scrip body can be detected by verifying the scrip's certificate.

5 Security Requirements

5.1 Issuer/Acquirer Requirements

The issuer and the acquirer are assumed to enjoy some degree of mutual trust. Moreover, an infrastructure enabling secure communication between these parties is already in place. Therefore, their roles and their respective requirements are unified.

- *Proof of transaction Authorization by the Customer*: When the broker records a debit from a certain bank account by a certain amount (the actual debit will happen later), the broker must be in possession of an unforgeable proof that the owner of the bank account has authorized this payment. This proof must not be "replayable", or usable as proof for another transaction. Note also, that in this context it is considered that the merchant may be an adversary; such a seller must not be able to generate a fake transaction.

- *Proof of Transaction Authorization by specific Merchant*: When the broker authorizes a payment to a certain merchant, the broker must be in possession of an unforgeable proof that the customer has asked to start a payment transaction with this merchant and also that this merchant is legitimate.

5.2 Merchant Requirements

- *Proof of transaction Authorization by Broker*: The merchant needs an unforgeable proof that the broker has authorized the transaction.

- *Proof of transaction Authorization by Customer*: Before the merchant receives the transaction authorization from the broker, the merchant needs an unforgeable proof that the customer has authenticated it. Furthermore, before the merchant sends the information message to the broker about a payment, s/he must be certain that this specific customer requested it.

5.3 Customer Requirements

- *Anonymity*: Customers desire anonymity from eavesdroppers and from merchants (merchants are aware only of the customers user identification number and cannot link it with his/her personal information, only the broker can). This feature is desirable in all payment systems that try to imitate cash, like the proposed peer-to-peer protocol.

- *Privacy*: The proposed peer-to-peer protocol respects the customers' privacy of order and payment information. For example, an investor purchasing information on certain stocks may not want competitors to be aware of the stocks s/he is interested in. The encryption of this information ensures the customers' privacy. Note that the proposed protocol does not provide unlinkability of customers and merchants with respect to the broker.

- *Impossibility of Unauthorized Payment*: It must be impossible to charge a customer's bank account without possession of the bank account number, PANSecret and private key. Thus, neither Internet rogues

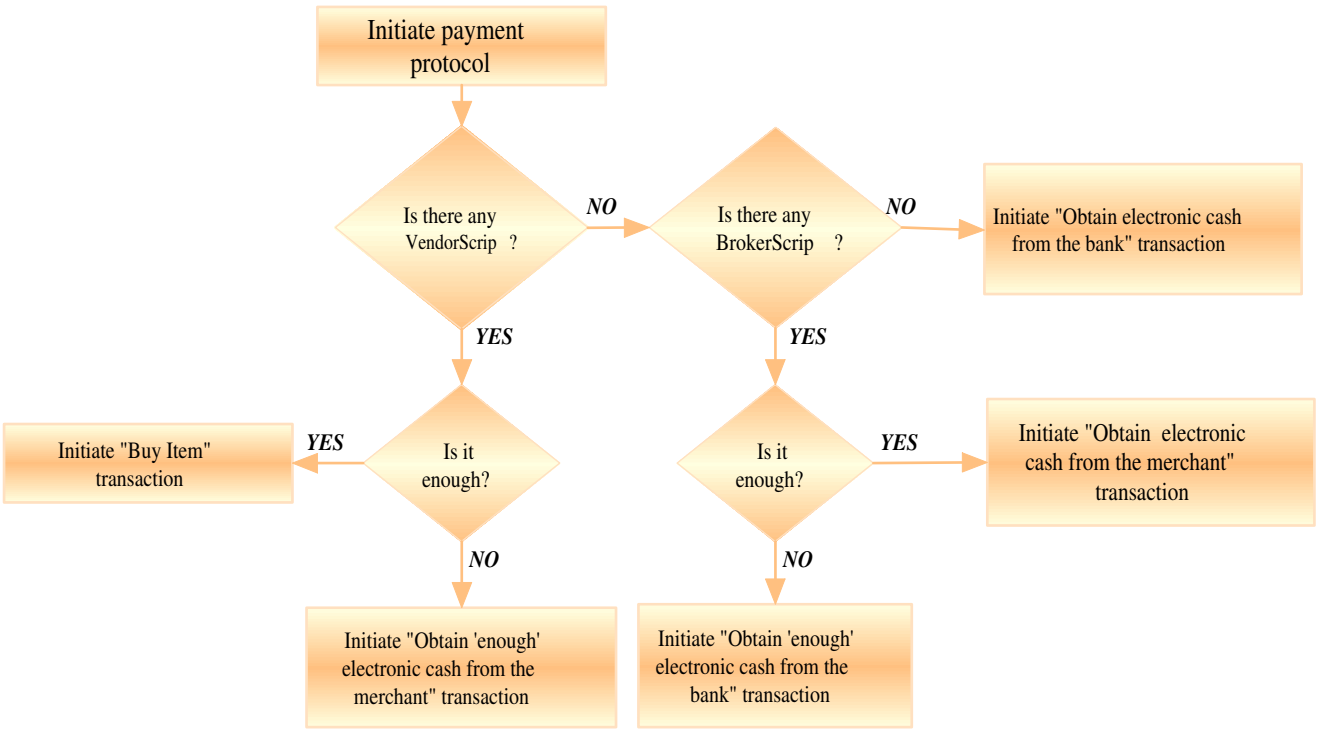


Figure 6: Flow chart of the payment protocol

nor malicious merchants must be able to generate spurious transactions, which end up approved by the broker. This case must remain even if the customer has engaged in many prior legitimate transactions. In other words, information sent in one (legitimate) transaction must not enable a later spurious transaction. So, in particular, the ID of the customer must not be sent in the clear, and not even be the subject to guessing attacks.

- *Proof of Transaction Authorization by Broker*: Customer might need to have a proof that the broker authorized the transaction.
- *Authentication of Merchant*: Customer may need proof that the merchant is a legitimate user of the payment system.
- *Receipt of the purchase*: The broker keeps records of all transactions that took place, thus a receipt is not necessary.

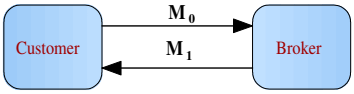
6 Payment Processing

In the two following paragraphs the preprocessing steps of the proposed "peer-to-peer" payment protocol are described using an example of an imaginary electronic market of second hand sold products. These steps are needed so that a trusted relationship between the merchant and the customer is established. Through the "Obtain electronic cash from the bank" transaction step the customer purchases from the bank electronic cash using a single macro-payment. Then, through the "Obtain electronic cash from the merchant" transaction step the customer, using once more a macro-payment, exchanges an amount of his/her electronic cash from the bank, with electronic cash from the merchant. In Figure 6 a flowchart of the payment process is given.

6.1 Obtain Electronic Cash From The Bank (BrokerScrip)

A customer peer that desires to buy products sold in the electronic market, needs to acquire BrokerScrip in order to exchange it afterwards for VendorScrip and finally being able to buy products from him/her. This is achieved through this transaction step (Figure 7), s/he establishes a connection to the broker and buys, using real-money, the desirable BrokerScrip. Having received the payment the broker delivers the BrokerScrip to the customer. The customer possesses only one BrokerScrip and s/he can obtain a new one only if s/he has spent it all. The BrokerScrip is used so as to obtain electronic cash from a merchant.

In M_0, the combination of the customer's digital signature and ID provide strong proof to the broker that the customer authorized the transaction. Further, the use of nonce ensures that the message is not replayable. Moreover, the use of encryption eliminates the exposure of the customer's ID and ensures the broker that the message was not altered.



C_0	*BrokerScrip request*
X_0	W_{Br}, N
M_0	**C_0, UID_C, $Enc_{K0}(SignOnly_{KC}(ID_C), Sign_{KC}(X_0))$, $PKEnc_{KB}(K_0)$**
C_1	*BrokerScrip response*
X_1	B_0, CS_{B0}, N
M_1	**C_1, UID_B, $Enc_{K1}(Sign_{KB}(X_1))$, $PKEnc_{KC}(K_1)$**

Figure 7: Obtain electronic cash from the bank

If the information received in this message and processed is valid, the broker creates the requested BrokerScrip. Furthermore, s/he records the information of the transaction. The recorded information can be used in case of a dispute. The broker sends the requested BrokerScrip in a new message (M_1). In this message, the digital signature of the broker ensures the customer that the broker authorized the transaction and the received BrokerScrip is legitimate. Further, the received nonce offers him/her proof that the message does not come from a replay attack. Finally, encryption ensures confidentiality of the information sent.

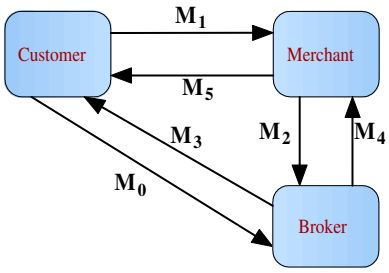
When the customer receives M_1, s/he decrypts it, verifies its signature and checks if the value of the received BrokerScrip is the requested one. If the processed data is valid, s/he stores the $PKEnc_{K_C}(B_0)$ and the $PKEnc_{K_C}(CS_{B_0})$, locally.

6.2 Obtain Electronic Cash From The Merchant

Each merchant accepts VendorScrip issued by him/her, so the customer that wants to purchase an item from the merchant and already owns BrokerScrip but no VendorScrip needs to apply for it. If the value of the owned BrokerScrip is higher than or equal to the one of the desirable VendorScrip, this transaction step is initiated (Figure 8).

In M_0, the digital signature and the customer's ID along with the BrokerScrip's corresponding CustomerSecret are used by the broker as a proof that the customer authorized the transaction. So, if this information is valid the broker records the information in a log file.

Further, the customer sends another message to the merchant (M_1). In this message, the digital signature and the customer's ID along with the BrokerScrip's corresponding CustomerSecret are used by the broker as a proof that the customer authorized the transaction. The broker is ensured that the message is not the product of a replay attack, because the BrokerScrip is valid only if it has not been used before. Finally, the use of encryption ensures confidentiality to the customer and proof to the



C_0	*Initiate VendorScrip request*
X_0	UID_M
M_0	**C_0, UID_C, $Enc_{K0}(SignOnly_{KC}(ID_C))$, $Enc_{K0}(Sign_{KC}(X_0))$, $PKEnc_{KB}(K_0)$**
C_1	*VendorScrip request*
X_1	B_0, CS_{B0}, W_{V0}
X_2	W_{V0}
M_1	**C_1, UID_C, $Enc_{K1}(SignOnly_{KC}(ID_C))$, $Enc_{K1}(Sign_{KC}(X_1))$, $PKEnc_{KB}(K_1)$, $Enc_{K2}(Sign_{KC}(X_2))$, $PKEnc_{KM}(K_2)$**
C_2	*Authorization request*
X_3	W_{V0}
M_2	**C_2, UID_M, $Enc_{K3}(SignOnly_{KM}(ID_M)$, $Sign_{KM}(X_3))$, $PKEnc_{KB}(K_3)$, $Enc_{K1}(SignOnly_{KC}(ID_C), Sign_{KC}(X_1))$, $PKEnc_{KB}(K_1)$**
C_3	*Change BrokerScrip*
X_4	B_1, CS_{B1}
M_3	**C_3, UID_B, $Enc_{K4}(Sign_{KB}(X_4))$, $PKEnc_{KC}(K_4)$**
C_4	*Authorization response*
X_5	R
M_4	**C_4, UID_B, $Sign_{KB}(X_5)$**
C_5	*VendorScrip response*
X_6	V_0, CS_{V0}
M_5	**C_5, UID_M, $Enc_{K5}(Sign_{KM}(X_6))$, $PKEnc_{KC}(K_5)$**

Figure 8: Obtain electronic cash from the merchant

broker that the message was not altered. Note that, for confidentiality reasons, the broker can only decrypt the part of the message that contains the ID of the customer.

The merchant receiving this message is able to process only her/his part (C_1, UID_C, $Enc_{K_2}(Sign_{K_C}(X_2))$, $PKEnc_{K_M}(K_2)$). S/he decrypts the message elements and verifies the signature of the message. The signature of the message ensures the merchant that the customer authorized the transaction. So, if the received information is valid, the merchant forms M_2 and sends it to the broker. In this message, the use of the merchant's ID along with the digital signature proves that the merchant authorized the transaction. Further, the use of encryption ensures confidentiality of the message elements and especially of the merchant's ID and moreover that the message can only be read by the broker.

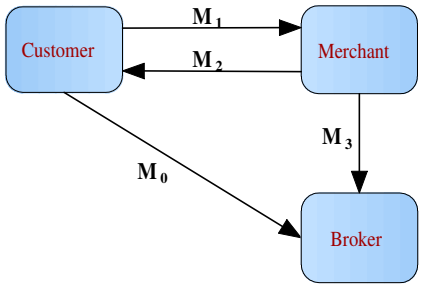
The broker receiving M_2, decrypts it and verifies the its signatures. Further, s/he verifies the IDs and the BrokerScrip. Finally, s/he checks if the values sent by both

customer and merchant are equal and if the value of the received BrokerScrip is greater/equal to the value of the requested VendorScrip. If the processed data is valid, the broker forms two messages, one for the customer (M_3) and one for the merchant (M_4) and updates the corresponding log file of the transaction.

In M_3 the use of the broker's digital signature, provides to the customer proof that the producer of the message is the broker. Further, the use of encryption ensures confidentiality of the information, sent.

In message M_4 the use of the broker's signature provides proof to the merchant that the broker authorized the transaction. So the merchant that receives the broker's message, verifies its signature. If the signature is valid and if the broker's authorization message is positive, the merchant forms M_5 and sends it to the customer otherwise the transaction stops. Regarding M_5's security requirements, the use of the merchant's digital signature provides strong proof to the customer that the merchant authorized the transaction. Further, the use of the session key encryption provides confidentiality.

The customer receives both messages, M_3 sent by the broker and M_5 sent by the merchant, decrypts them and checks their signatures. Further, s/he checks if the amounts are correct and then stores $PKEnc_{K_C}(V_0)$, $PKEnc_{K_C}(CS_{V_0})$, $PKEnc_{K_C}(B_1)$ and $PKEnc_{K_C}(CS_{B_1})$ locally.



C_0	*Initiate Purchase request*
X_0	UID_M, W_P
M_0	**$C_0, UID_C, Sign_{KC}(X_0)$**
C_1	*Purchase request*
X_1	V_0, CS_{V0}, OI
M_1	**$C_1, UID_C, Enc_{K0}(Sign_{KC}(X_1)), PKEnc_{KM}(K_0)$**
C_2	*Purchase response*
X_2	V_1, CS_{V1}, OI
M_2	**$C_2, UID_M, Enc_{K1}(Sign_{KC}(X_2)), PKEnc_{KC}(K_1)$**
C_3	*Purchase request initiated*
X_3	UID_C, W_P
M_3	**$C_3, UID_M, Sign_{KM}(X_3)$**

Figure 9: Buy Item

6.3 Buy Item

The customer that owns appropriate Vendorscrip for purchasing a desired item from the merchant should send it to him/her (Figure 9). The merchant checks and validates the scrip, s/he reduces the value of the scrip and sends a new scrip (the change) to the customer. This interaction means that the customer has paid the merchant.

In M_0, the digital signature provides proof to the broker that the customer authorized the transaction and that the message was not altered. The broker receiving this message verifies its signature and records the transaction's information to a log file.

In M_1, the CustomerSecret along with the customer's digital signature ensure the merchant that the customer authorized the transaction. Further, the use of encryption ensures the customer concerning the confidentiality of the transmitted data and allows the merchant to detect any modifications of the message. The merchant receiving the message decrypts its elements and verifies the message's signature. Further, s/he verifies the VendorScrip and sends the change VendorScrip to the customer in anew message (M_2). Finally, s/he sends an information message to the broker (M_3).

In M_2, the use of the merchant's signature ensures the customer that the merchant authorized the transaction. Further, encryption provides confidentiality. The customer who receives the message, decrypts its elements and verifies its signature. Then s/he checks if the value of the change VendorScrip is correct and stores the $PKEnc_{K_C}(V_1)$ and the $PKEnc_{K_C}(CS_{V_1})$ locally.

The broker is ensured that the M_3 message was not altered and that the merchant authorized the transaction, by verifying the digital signature of the message. Further, if the signature is valid s/he retrieves the corresponding log file and updates it.

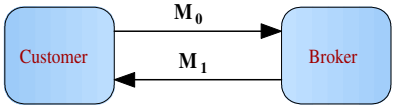
6.4 Obtain "enough" Electronic Cash From The Bank

The customer, always, holds only one BrokerScrip, which is used in many transaction from obtaining VendorScrips. If the desirable VendorScrip's value exceeds the BrokerScrip's value this transaction is initiated (Figure 10).

In M_0, the customer's ID, the scrip's CustomerSecret and the customer's digital signature ensure the broker that the customer authorized the transaction. Further, encryption guarantees confidentiality of the transmitted data. Finally, since the scrip is not reusable, the broker is ensured that the message is not the product of a replay attack. So, if all data received in this message and processed is valid, the broker forms a new message (M_1), sends it to the customer and also records the information of the transaction in a log file.

In M_1, the broker's signature provides strong proof to the customer that the broker authorized the transaction. Moreover, data encryption ensures confidentiality of the transmitted data.

The customer receives the message, decrypts its ele-



C_0	*"Enough" BrokerScrip request*
X_0	W_{B1}, $_{*0}$, CS_{B0}
M_0	**C_0, UID_C, Enc_{K0}(SignOnly $_{KC}$(ID_C)),Sign(X_0)), $PKEnc_{KB}$(K_0)**
C_1	*"Enough" BrokerScrip response*
X_1	B_1, CS_{B1}
M_1	**C_1, UID_B, Enc_{K1}(Sign $_{KB}$(X_1)), $PKEnc_{KC}$(K_1)**

Figure 10: Obtain "enough" electronic cash from the bank

ments, verifies the signature of the message and finally, checks if the amount of the received BrokerScrip is equal to the requested one and then s/he stores locally the $PKEnc_{K_C}(B_1)$ and the $PKEnc_{K_C}(CS_{B_1})$.

6.5 Obtain "enough" Electronic Cash From The Merchant

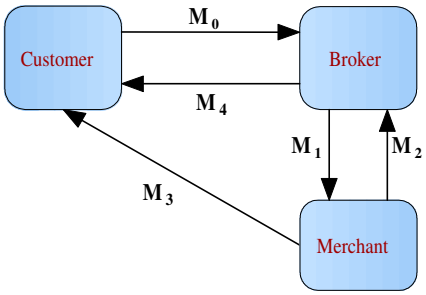
When a customer has already purchased a product from a merchant and wants to continue doing business with this merchant and furthermore holds VendorScrip, from this specific merchant but the item's value exceeds the VendorScrip's value, this transaction is initiated (Figure 11).

The customer forms M_0 and sends it to the broker. The scrip's (BrokerScrip/VendorScrip) corresponding CustomerSecret and the digital signature of the message, provide proof to the broker/merchant that the customer authorized the transaction. Further, confidentiality and data integrity are ensured by the use of encryption.

The broker receives the message and processes the part intended for her/him. S/he decrypts the message elements, verifies the signature of the message and the received BrokerScrip. Further, s/he checks if the value of the received BrokerScrip exceeds the value of the needed $(W_1 - W_2)$ VendorScrip. If the processed information is valid, s/he forms M_1, sends it to the merchant and stores, locally, the received BrokerScrip along with the change BrokerScrip if there is any. Further, s/he records the transaction information to a log file.

In M_1, the signature of the broker is the proof for the merchant that the broker authorized the transaction. Further, confidentiality is ensured by the use of encryption.

The merchant receives M_1 which is actually a concatenation of the two messages; one formed by the broker and one forwarded by the broker (originally sent by the customer). First, s/he processes the broker's part; s/he decrypts its elements and verifies the signature. If the processed message elements are valid, s/he processes the customer's part; decrypts the message's elements, verifies the signature and the received VendorScrip. If the processed data is valid, s/he checks if W_1 and W_3 are equal, if they are, s/he forms two messages one for the customer (M_3) and one for the broker (M_2).



C_0	*"Enough" VendorScrip request*
X_0	W_{V1}, W_V, UID_M, B_0, CS_{B0}
X_1	W_{V1}, V_0, CS_{V0}
M_0	**C_0, UID_C, Enc_{K0}(Sign $_{KC}$(X_0)), $PKEnc_{KB}$(K_0), Enc_{K1}(Sign $_{KC}$(X_1)), $PKEnc_{KM}$(K_1)**
C_1	*Authorization request*
X_2	W_{V1}, UID_C
M_1	**C_1, UID_B, Enc_{K2}(Sign $_{KB}$(X_2)), $PKEnc_{KM}$(K_2), Enc_{K1}(Sign $_{KC}$(X_1)), $PKEnc_{KM}$(K_1)**
C_2	*Authorization response*
X_3	A , UID_C
M_2	**C_2, UID_M, Sign $_{KM}$(X_3)**
C_3	*"Enough" VendorScrip response*
X_4	V_1, CS_{V1}
M_3	**C_3, UID_M, Enc_{K3}(Sign $_{KM}$(X_4)), $PKEnc_{KC}$(K_3)**
C_4	*Change BrokerScrip*
X_5	B_1, CS_{B1}
M_4	**C_4, UID_B, Enc_{K4}(Sign $_{KB}$(X_5)), $PKEnc_{KC}$(K_4)**

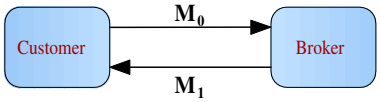
Figure 11: Obtain "enough" electronic cash from the merchant

In M_3, the digital signature of the message ensures the customer that the merchant authorized the transaction. Moreover, the use of encryption ensures confidentiality of the transmitted information.

In M_2, the merchant's digital signature ensures the broker that the merchant authorized the transaction. So, the broker who receives this message verifies its signature and if it's valid s/he deletes the temporary stored BrokerScrip. Furthermore, if there is any change BrokerScrip, s/he forms the M_4 message sends it to the customer and deletes the temporarily stored change BrokerScrip. Finally, s/he updates the corresponding log file with the information of the transaction.

In M_4, the customer is ensured that the broker authorized the transaction through the digital signature of the broker. Further, data integrity and confidentiality are ensured by the appliance of encryption.

The customer receives the merchant's message (M_3),



C_0	*BrokerScrip withdraw request*
X_0	B_0, CS_{B0}
M_0	**C_0, UID_C, Enc_{K0}(SignOnly $_{KC}$(ID_C), Enc_{K0}(Sign $_{KC}$(X_0)), $PKEnc_{KB}$(K_0)**
C_1	*BrokerScrip withdraw response*
X_1	I
M_1	**C_1, UID_B, Sign $_{KB}$(X_1)**

Figure 12: BrokerScrip withdraw

decrypts its elements and verifies its signature. Further, s/he checks if the value of the received VendorScrip is equal to the requested one. Finally, s/he stores the $PKEnc_{K_C}(V_2)$ and the $PKEnc_{K_C}(CS_{V_2})$, locally. The customer also receives the broker's message (M_4), decrypts its elements and verifies its signature. Further, s/he checks if the value of the received BrokerScrip is correct. Finally, s/he stores the $PKEnc_{K_C}(B_2)$ and the $PKEnc_{K_C}(CS_{B_2})$, locally.

6.6 BrokerScrip Withdraw

When the customer does not desire anymore buying things from the electronic market can withdraw his/her BrokerScrip and to deposit its value back to his/her bank account (Figure 12).

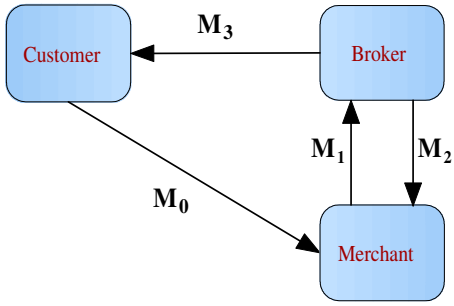
In M_0, the scrip's CustomerSecret along with the customer's ID and digital signature ensure that the customer authorized the transaction. Further, confidentiality and data integrity are ensured by the use of encryption. Finally, based on the non-reusable nature of the scrip, any replay attacks can be detected.

The broker receives the above message, decrypts its elements and verifies its signature. Further, verifies the customer's ID and the received BrokerScrip. If the processed data is valid the transaction is recorded, the corresponding MasterScripSecret and MasterCustomerSecret of the message are deleted and an information message is sent to the customer M_1. The digital signature of this message ensures the customer that the broker authorized the transaction.

Finally, the customer receives the message and verifies its signature. Then, s/he deletes from her/his local file system the withdrawn BrokerScrip and its corresponding CustomerSecret.

6.7 VendorScrip Withdraw

The customer has also the ability to withdraw his/her VendorScrip and to deposit its value back to his/her bank account (Figure 13).



C_0	*VendorScrip withdraw request*
X_0	V_0, CS_{V0}
X_1	W_{V0}
M_0	**C_0, UID_C, Enc_{K0}(Sign $_{KC}$(X_0)), $PKEnc_{KM}$(K_0), Enc_{K1}(SignOnly $_{KC}$(ID_C), Sign $_{KC}$(X_1)), $PKEnc_{KB}$(K_1)**
C_1	*Authorization request*
X_2	W_{V0}, UID_C
M_1	**C_1, UID_M, Enc_{K2}(Sign $_{KM}$(X_2)), $PKEnc_{KB}$(K_2), Enc_{K1}(SignOnly $_{KC}$(ID_C),Sign $_{KC}$(X_1)), $PKEnc_{KB}$(K_1)**
C_2	*Authorization response*
X_3	R, UID_C
M_2	**C_2, UID_M, Enc_{K3}(Sign $_{KB}$(X_3)), $PKEnc_{KM}$(K_3)**
C_3	*VendorScrip withdraw response*
X_4	UID_M, I
M_3	**C_3, UID_B, Enc_{K4}(Sign $_{KM}$(X_4)), $PKEnc_{KC}$(K_3)**

Figure 13: VendorScrip withdraw

In M_0, the customer's ID and the digital signature provide proof to the broker that the customer authorized the transaction. Further, the use of the scrip's CustomerSecret and the customer's digital signature ensure the merchant that the customer authorized the transaction. Since the scrip is not reusable, the merchant is able to detect any replay attacks. Finally, confidentiality and data integrity are ensured by the use of encryption.

The merchant receives M_0, decrypts the part of it intended for her/him, verifies its signature and the received VendorScrip. If the received information is valid, s/he forms M_1, sends it to the broker and stores temporarily the received VendorScrip.

In M_1, the merchant's signature provides proof to the broker that the merchant authorized the transaction. Further, encryption ensures data integrity and confidentiality of the transmitted information. The broker receives this message and processes its two parts; decrypts their message elements and verifies their signatures. Further, s/he verifies the customer's ID. If the processed information is valid, s/he compares the W_1 and W_2 and if they are equal forms two messages, one for the merchant (M_2) and one for the customer (M_3), sends them to their recipients and records the transaction's information in a log file.

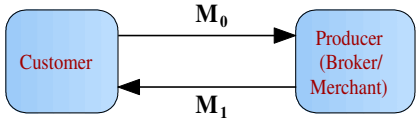
In M_3, the broker's signature ensures the customer that the broker authorized the transaction. Confidentiality and data integrity are ensured by encrypting the transmitted information. The customer receives this message, decrypts its elements and verifies its signature. Finally, s/he deletes the VendorScrip and its corresponding CustomerSecret from his/her local file repository.

In M_2, confidentiality and data integrity are ensured through the use of encryption. Further, the merchant has strong proof that the broker authorized the transaction, because of the digital signature of the message. The merchant that receives the message verifies its signature, if it is valid s/he retrieves and deletes the temporarily stored VendorScrip and its corresponding MasterScripSecret and MasterCustomerSecret.

6.8 Expired Scrip

The scrip (BrokerScrip/VendorScrip) has an expiration date. After this date the scrip is not valid. When the scrip is not valid this transaction is initiated (Figure 14) and the scrip is sent to its producer in order to be renewed. Through this process the MasterScripSecret and MasterCustomerSecret which correspond to the scrip and are stored in the producer's look-up tables, are renewed. The same procedure takes place regarding the corresponding CustomerSecret, which is stored in the customer's local file repository.

In M_0, the digital signature and the scrip's corresponding CustomerSecret provide strong proof to the producer that the customer authorized the transaction. Moreover, data integrity and confidentiality of the transmitted information are ensured through the use of encryption. Finally, the non-reusable nature of the scrip ensures the



C_0	*Expired scrip request*
X_0	E_0, CS_{E0}
M_0	**C_0, UID_C, $Enc_{K0}(Sign_{KC}(X_0))$, $PKEnc_{KP}(K_0)$**
C_1	*Expired scrip response*
X_1	E_1, CS_{E1}
M_1	**C_1, UID_P, $Enc_{K0}(Sign_{KP}(X_1))$, $PKEnc_{KC}(K_1)$**

Figure 14: Expired scrip

producer that the message is not the product of a replay attack.

The producer of the scrip receives this message, decrypts its elements and verifies the signature of the message. Further, s/he verifies the received scrip and creates a new one. The new scrip contains a new expiration date, new ScripID and CustID, a new certificate and new CustomerSecret. The only common part between the old and the new scrip is their values. Then s/he forms M_1, sends it to the customer and deletes the old scrip's corresponding MasterScripSecret and MasterCustomerSecret.

In the message sent, the use of the producer's digital signature provides proof to the customer that the producer authorized the transaction. Further, encryption ensures data integrity and confidentiality of the transmitted information.

The customer receives M_1, decrypts its elements, verifies the signature of the message and checks if the value of the updated scrip equals to the one of the expired scrip. If the processed data is valid, s/he stores the $PKEnc_{K_C}(E_1)$ and the $PKEnc_{K_C}(CS_{E_1})$, locally.

7 Brokers Computational Cost

In the proposed protocol the involvement of the broker and so for his/her operational and computational cost has been reduced. As mentioned previously, the broker represents the financial institutions so his/her role in the payment process is essential. In the transaction steps of the payment protocol the broker acts as both the payment authorization entity and as an observer/recorder of the transactions/transactions details.

Regarding the three main transaction steps of the payment process of the protocol : 1."Obtain electronic cash from the bank", 2."Obtain electronic cash from the merchant" and 3."Buy item" (the rest transaction steps can be considered as supplements of these steps), in the two first ones the broker acts both as the payment authorization entity and as the observer so his/her computational cost is high (s/he has to process a lot of cryptographical operations). But in the third transaction step s/he acts as

the observer/recorder and his/her computational cost is reduced to two signature verifications and to the insertion and update of a log file.

In an optimized use of the proposed protocol the two first steps, where a trustworthy buyer/seller relationship is established between the peers, should occur less times than the third step. This observation implies that with the peer-to-peer protocol its achieved a reduction of the brokers computational load.

8 Conclusion

In this paper a novel payment protocol is presented. This protocol can be used in any kind of network architecture but its main purpose is to be used in a P2P network. The first two transaction steps of the payment process, "Obtain electronic cash from the bank" and "Obtain electronic cash from the merchant", are considered to be the necessary steps so as to establish a trustworthy business relationship between the customer and the merchant. In the third transaction step, which actually enables the purchase, the broker's role is minimized to one of an observer that records the information of the transaction; the actual transaction takes place between the customer and the merchant. Further, the new protocol is compliant to all parties' requirements involved in a transaction and offers confidentiality and full anonymity to the customers. Finally, it establishes a framework for enabling secure payment transactions.

References

[1] *Amazon.com, Inc*, http://www.amazon.com/.

[2] P. C. Cheng, J. Garay, A. Herzberg, and H. Krawczyk, "A security architecture for the internet protocol", *IBM Systems Journal - Special Issue Internet*, vol. 37, no. 1, pp. 42-60, 1998.

[3] I. Clarke, O. Sandberg, B. Wiley, and T. Hong. Freenet, "A distributed anonymous information storage and retrieval system: Designing privacy enhancing technologies", *Proceedings of International Workshop on Design Issues in Anonymity and Unobservability*, LNCS 2009, pp. 46-66, 2001.

[4] B. Cox, J. D. Tygar, and M. Sirbu, "Netbill: security and transaction protocol", in *The first USENIX Workshop on Electronic Commerce*, New York, pp. 77-88, 1995.

[5] J. Daemen and V. Rijmen, *"The Rijndael block cipher"*, http://csrc.nist.gov/encryption/aes/round2/AESlgs/Rijndael/Rijndael.pdf, Sept. 1991, AES Proposal.

[6] T. Dierks and C. Allen, *"The TLS protocol version 1.0"*, Internet RFC 2246, Jan. 1999

[7] *ebay, Inc*, http://www.ebay.com/.

[8] A. O. Freier, P. Kariton, and P. C. Kocher, *"The SSL protocol: Version 3.0"*, 1996.

[9] J. Garms and D. Somerfield, *"Professional Java Security"*, Wrox, 2001.

[10] S. Glassman, M. Manasse, M. Abadi, P. Gauthier, and P. Sobalvarro, "The Millicent protocol for inexpensive electronic commerce", in *Proceeding of the 4th International World Wide Conference*, pp. 603-618, Dec. 1995.

[11] *Gnutella.com, Inc*, http://gnutella.wego.com/, http://www.gnutella.co.uk/.

[12] *IAIK Java Group. IAIK Java Crypto Software*, http://jce.iaik.tu-graz.ac.at/.

[13] *Kazaa.com*, http://www.kazaa.com/.

[14] Z. Y. Lee, H. C. Yu, and P. J. Kuo, "An analysis and comparison of different types of electronic payment systems", *Management of Engineering and Technology, PICMET '01*, vol. 2, pp. 38-45, 2001.

[15] Mastercard, Visa, *"SET 1.0 - Secure Electronic Transaction Specification"*, http://www.mastercard.com/set.html, 1997.

[16] Mastercard, Visa, *"SET Secure Electronic Transactions Protocol"*, version 1.0 ed. May 1997, Book One: *Business Specifications*, Book Two: *Technical Specification*, Book Three: *Formal Protocol Definition.*

[17] A. Menezes, P. van Oorschot, and S. Vanstone, "Handbook of Applied Cryptography", *CRC Press*, 1996.

[18] D. S. Milojicic, V. Kalogeraki, R. Lukose, K. Nagaraja, J. Pruyne, B. Richard, S. Rollins, and Z. Xu, *"Peer-to-Peer Computing"*, HP Laboratories Palo Alto. Technical Report HPL-2002-57, 2002.

[19] R. L. Rivest, A. Shamir, and L. M. Adleman, "A method for obtaining digital signatures and public key cryptosystems", *Communications of the ACM*, vol. 21, no. 2, pp. 120-126, Feb. 1978.

[20] C. Shirky, *"What is P2P... And What Isn't"*, http://www.openp2p.com/pub/a/p2p/2000/11/24/shirky1-whatisp2p.html

[21] C. Wade, *"New Models for Providing Payment Services over the Internet"*, http://commerce.net.

[22] P. Wallish, "Cyber view: How to steal millions in champ change", *Science American*, pp. 32-33, Aug. 1999.

[23] K. Wei, Y. F. R. Chen, A. J. Smith, B. Vo, "WhoPay: A scalable and anonymous payment system for peer-to-peer environments", *Network Computer Sience Technical Reference Library*, http://uther.dlib.vt.edu/

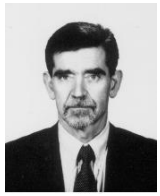
[24] B. Yang and H. Garcia-Molina, "PPay : Micropayments for peer-to-peer systems", *Proceedings of the 10th ACM conference on Computer and communications security , CCS'03*, pp. 300-310, Oct. 2003.

Despoina Palaka was born in Larissa, Greece in 1978 and she is an Asscociate Researcher at the Informatics and Telematics Institute. She received the Diploma degree in Electrical and Computer Engineering from the Aristotle University of Thessaloniki, Greece, in 2002. Her main research interests include peer-to-peer technologies, cryptography, Network Security and watermarking of 3D objects.

Petros Daras was born in Athens, Greece in 1974 and he is an Associate Researcher at the Informatics and Telematics Institute. He received the Diploma degree in Electrical and Computer Engineering, the MSc degree in Medical Informatics and the Ph.D. degree in Electrical and Computer Engineering from the Aristotle University of Thessaloniki, Greece, in 1999, 2002 and 2005 respectively. His main research interests include Computer Vision, search and retrieval of 3D objects, the MPEG-4 standard, peer-to-peer technologies and medical informatics. He has been involved in more than 10 European and National research projects. Dr. Daras is a member of the Technical Chamber of Greece.

Kosmas Petridis was born in Katerini, Greece, in 1974. He received the Bachelor degree in Computer Science from the University of Crete, Heraklion in 1996. He is currently pursuing his Masters degree in "Advanced Computing & Communication Systems" at the Electrical & Computer Engineering department, Aristotle University of Thessaloniki.

Since 2000 he has been working as a research associate with the Informatics and Telematics Institute, Thessaloniki, Greece. He has participated in several research projects funded by the European Union. His research interests include Software Engineering, Distributed Engineering, Database Development & Administration, Semantic Web and Human-Computer Interaction.



Michael G. Strintzis received the Diploma degree in electrical engineering from the National Technical University of Athens, Athens, Greece, in 1967, and the M.A. and Ph.D. degrees in electrical engineering from Princeton University, Princeton, NJ, in 1969 and 1970, respectively. He then joined the Electrical Engineering Department at the University of Pittsburgh, Pittsburgh, PA., where he served as Assistant Professor (1970-1976) and Associate Professor (1976-1980). Since 1980, he has been Professor of electrical and computer engineering at the University of Thessaloniki, Thessaloniki, Greece, and, since 1999, Director of the Informatics and Telematics Research Institute, Thessaloniki. His current research interests include 2-D and 3-D image coding, image processing, biomedical signal and image processing, and DVD and Internet data authentication and copy protection. Dr. Strintzis has served as Associate Editor for the IEEE Transactions on Circuits and Systems for Video Technology since 1999. In 1984, he was awarded one of the Centennial Medals of the IEEE.