A secure protocol for a payment system based on a kiosk centric case mobile scenario

Jesús A. Téllez [(1)], José M. Sierra [(2)], Antonio Izquierdo [(2)], Mildrey Carbonell [(2)]

[(1)] Computer Science Department-FACYT, University of Carabobo, Venezuela. Email:jtellez@uc.edu.ve
[(2)] Computer Science Department, University Carlos III of Madrid, Leganés (Madrid), Spain.
Email: sierra@inf.uc3m.es, aizquier@inf.uc3.es, mildreycc@yahoo.es.

Abstract

The Full connectivity scenario has been widely used in most of the mobile payment systems proposed up until now because it allows to protocol's designers to simplify the design and development of payment protocols without losing security guarantees. However, the aforementioned scenario does not consider those situations in which the customer cannot communicate with the issuer due to absence of Internet access in his/her infrastructure. In order to overcome this restriction, in this paper we present a secure protocol for a mobile payment system based on a Kiosk Centric Case mobile Scenario that employs symmetric-key operations which require low computational power and can be processed much faster than asymmetric ones. Our protocol protects the real identity of the clients during the purchase and illustrates how a portable device equipped with a short range link (such Bluetooth, Infrared, etc.) and low computational power should be enough to buy goods in a secure way.

Keywords: Mobile payment system, electronic commerce protocol, security.

Un protocolo seguro para un sistema de pago móvil basado en el modelo céntrico del quiosco

Resumen

El escenario de conectividad completa ha sido usado en la mayor parte de los sistemas de pago móvil propuestos hasta ahora ya que permite a los diseñadores de protocolos, simplificar el diseño y desarrollo de protocolos de pago sin perder las garantías de seguridad. Sin embargo, el mencionado escenario no considera aquellas situaciones en las cuales el cliente no puede comunicarse con el emisor debido a la ausencia de acceso a Internet en su infraestructura. Para superar esta restricción, en este artículo presentamos un protocolo seguro para un sistema de pago móvil basado en el modelo céntrico del quiosco que utiliza operaciones de clave simétrica que requiere bajo poder computacional y son procesadas con mayor rapidez que las asimétricas. Nuestro protocolo protege la identidad verdadera de los clientes durante la compra e ilustra como un dispositivo portable equipado con un enlace de corto enlace (como Bluetooth, Infrarrojo, etc.) y bajo poder computacional debería ser suficiente para comprar bienes de manera segura.

Palabras Clave: Sistema de pago móvil, protocolo comercio electrónico, seguridad.

1. INTRODUCTION

The popularity of m-commerce has increased in the last years thanks to advances in the portable devices and the rapid development of the mobile communication technologies that have allowed people to use mobile telephones or Personal Digital Assistant (PDA) to access

the Internet (to read email, browse web pages or purchase information or goods) anywhere and anytime.

Different mobile payment systems have been proposed in the last years, but the one developed by [2] (called 3-D Secure) has become a standard due to its benefits regarding security and flexibility in the authentication methods. This schema allows the authentication of the payer (customer) when she makes an on-line payment using a debit or credit card. The transaction flow for this scheme is shown in figure 1 where all the main communications links are protected using SSL/TLS and the communication between the issuer/consumer is mandatory.

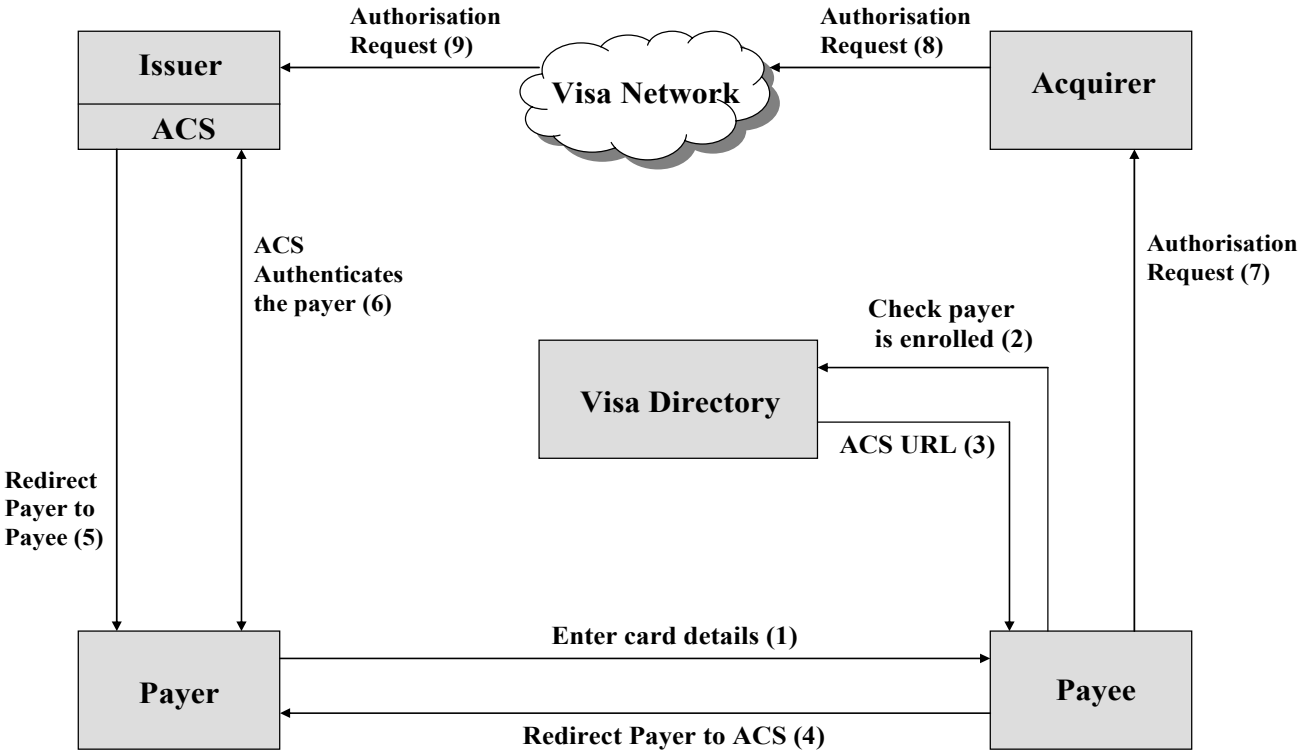

Figure 1. 3-D Secure transaction ([1]).

Despite of the flexibility that 3-D Secure gives to the issuer to choose the authentication methods, relationship between payer and issuer is quite strict (although required for Visa's 3D-Secure scheme) and does not allow the use of schemes in which the communication among these parties is not possible due to: 1) the impossibility of the client to connect to Internet from the mobile device and 2) the high costs of the infrastructure necessary to implement other mechanisms of communication between the client and the issuer.

Most of the mobile payment systems proposed up until now assume the consumer has Internet connectivity through her mobile device, so the restrictions mentioned previously do not represent an important issue. However, it is quite common that the client meets situations in which it is not possible to connect to Internet so it becomes necessary to develop mobile payment systems where the user could use her mobile device as a shopping means, even thought she may not have Internet access.

On the other hand, in spite of the wide range of mobile devices available, they all have common limitations [3]: 1) poor computational capabilities, 2) limited storage space and 3) short battery life. These limitations prevent that these devices execute, in an efficient way, computations that require a lot of resources, like those of asymmetric cryptography.

Symmetric cryptography (which employs a shared key between two parties) provides, like asymmetric cryptography, message confidentiality, message integrity

and party authentication, and represents an alternative in the construction of secure protocols for mobile payment systems, because symmetric-key operations do not require of a high computational power nor additional communications steps (as happens in protocols based on public-key infrastructure where the public-key certificates have to be verified by a Certificate Authority).

In this paper, we present a protocol (that supports both credit-card and debit-card transactions) for a mobile payment system based on a Kiosk Centric Case mobile scenario (proposed by [4]) which overcomes the limitations mentioned before. Our proposal represents an alternative to the restrictions of mobile payment systems (including Visa´s 3-D Secure) as for the connection between the client and issuer. Moreover, it uses symmetric-key operations in all engaging parties to reduce both, the setup cost for payment infrastructure and the transaction cost. Another benefit derived of the using of our proposal is a reduction of all parties´ computation and communications steps (in comparison with protocols based on public-key infrastructure) that make it suitable for mobiles devices with low computational power.

The rest of this paper is organized as follows: In next section, we survey related work. Section 3 presents the proposed system. In section 4, we analyze the scheme proposed. We end with our conclusions in Section 5.

2. RELATED WORK

In recent years, several studies have been conducted to improve the security of mobile payment systems. Meanwhile, efforts have also been dedicated to unify concepts and scenarios into frameworks that will be useful to develop new electronic payment systems. Research conducted by [4] is an example of a study that unifies many proposed m--commerce usages into a single framework. This research intended to revise the possible range of mobility scenarios, identifying the security issues for each connectivity scenario. As a result, five scenarios were identified and analyzed: Disconnected Interaction, Server Centric Case, Client Centric Case, Full Connectivity and Kiosk Centric Case. The latest has been considered as the starting point in the design of our proposal.

In [5], payment methods are classified according to several standards and analyzed to point out their advantages and drawbacks. Besides, the research also provides a payment process for mobile devices based on pre-payment and accounts. This proposed solution´s requirements are low (both on cost and technical capabilities) and it also has high scalability and security properties. However, their methods and processes are not suitable for our proposal, as our goal is to suggest a scheme based on post-payment and symmetric cryptography.

A secure and efficient one-way mobile payment system was proposed by [3]. In their solution the security of the system is based on the intractability of the discrete logarithm problem and the one-wayness of keyed hash function. As opposed to their goal (designing a mobile payment system with minimal complexity using two public key pairs), our solution aims for devising a scheme that relies on symmetric-key operations instead.

The closest work to ours is [6]. Their work proposed a secure account-based payment protocol suitable for wireless networks that employs symmetric-key operations which require lower computation at all engaging parties than existing payment protocols. Also, they use an Authenticated-Key Exchange protocol (called AKE) that does not use public-key cryptography (see [7]), instead of ikp ([8]) and SET protocols). While this proposal satisfies the majority of our requirements, we have to reformulate their protocol (from now on, SAMPP) to satisfy the requirements of the scheme that we suggest in this work, where the customer never establishes any connection with the bank (by any way) during the payment transaction.

As the payment software (also called wallet software) must be sent to the customer by the issuer through the vendor, it becomes necessary the use of techniques to assure that the program received by the client was created and sent by the issuer, and has not been tampered. In order to obtain the protection of the payment software in the aspects mentioned before, two different proposals related to the aforementioned techniques will be detailed in the following paragraphs.

The first work (proposed by [9]) introduced a new approach to watermarking, called path based watermarking, that embeds the watermark, with relatively low cost, in the dynamic branch structure of the program, and shows how error-correcting and tamper proofing techniques can be used to make path based watermarks resilient against a wide variety of attacks. The other work, proposed by [10], describes three techniques for obfuscation of program design: 1) The class coalescing obfuscation, 2) Class splitting obfuscation, and 3) Type hiding obfuscation. The experimental results (applying theses obfuscations to a medium-size java program) shows that the run-time overhead, in the worst of the case (class splitting obfuscation), is less than 10% of the total running time of the program.

3. SCHEME PROPOSED

3.1 Notations

All the entities involved in our protocol are called parties and communicate through wireless and wired network.

The symbols C, V, P, I, A are used to denote the names of the parties Customer, Vendor, Payment Gateway, Issuer and Acquirer respectively. The following symbols are used to represent other messages and protocols:

- ID_P : the identity of party *P* that contains the contact information of *P*.
- NIDC : Client's nickname, temporary identity.
- TID: Identity of transaction that includes time and date of the transaction.
- OI: Order information (OI = {TID, *h*(OI, Price)}) where OI and Price are order descriptions and its amount.
- TC: The type of card used in the purchase process (TC=Credit, Debit).
- Stt: The status of the transaction (Stt = {Accepted, Rejected}).
- TIDReq : The request fot TID.
- VIDReq : The resquest for ID_V .
- $\{M\}_X$: the message *M* symmetrically encrypted with the shared key *X*.
- MAC(X, K): Message Authentication Code of the message *X* with the key *K*.
- *h*(X): the one-way has function of the message *X*.

3.2. Operational Model

Generally, operational models for m-commerce found in literature involve transaction between two or more entities. Our operational model is composed of four entities: 1) *Customer:* a user who wants to buy information or goods from the vendor and has a mobile device with low computational power and equipped with a built-in display, keyboard (not necessarily with a QWERTY layout), short range link (such Infrared, Wi-Fi or Bluetooth) and capability to execute a java program, 2) *Vendor:* a computational entity (a normal web or an intelligent vending machine) that wants to sell information or goods and with which the user participates in a transaction, 3) *Acquirer:* the vendor's financial institution, 4) *Issuer:* the customer's financial institution, and 5) *Payment Gateway:* additional entity that acts as a medium between acquirer/issuer at banking private network side and customer/vendor at the Internet side for clearing purpose.

In figure 2, we specify the links among the four entities of our scheme. Note that there is no direct connection involving the customer and the issuer. Moreover, the connection between the customer and the vendor (denoted as the dotted arrow) is set up through a wireless channel.

On the other hand, interaction among the vendor and the payment gateway (depicted as solid arrow in the scheme) should be reliable and secure against passive and active attacks. Therefore, the connection is supposed to be established through a secure wired channel by using the well-know security protocol like SSL/TLS [3]. Note that the issuer, acquirer and payment gateway operates under the banking private network so we do not concern about connection´s security among these entities.

The protocol based in symmetric cryptography proposed by [6] is a starting point of our protocol. We reformulated this protocol to satisfy the requirements of that, as stated before, pretends to allow the client to make purchases from its mobile device without connecting itself to Internet.

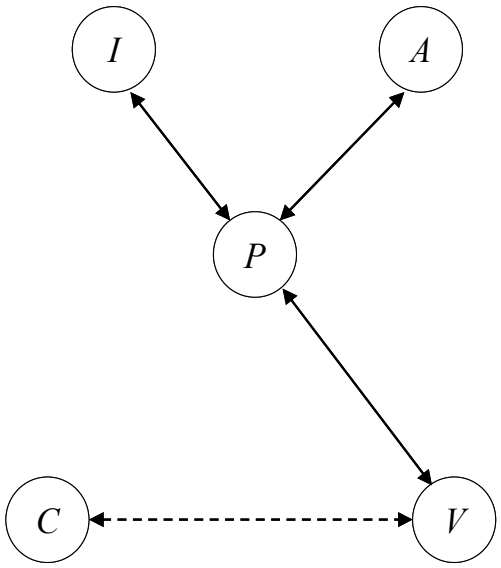

Figure 2. Operational Model.

3.2 Key Generation Technique

Our scheme handles three different sets of shared keys used for encrypt a message symmetrically. Each one is generated off-line in the entity that will store them.

The first set $VPSec_j$, is generated from the secret VPSec and stored in the vendor and Payment gateway

terminals respectively. The other set $CISec_i$ (stored in the customer's device and issuer's terminal, respectively), is generated from the secret CISec. The last set $CVSec_k$ is generated from the secret CVSec and are stored in the customers device and the vendors terminal respectively.

In order to generate the sets of shared keys, we apply a Hash algorithm with one-bit cyclic chain function of a master secret each time a session key is generated [6]. The details are shown as follows:

Generating $VPSec_j$ and $CVSec_k$

$VPSec_1 = h$(1-bit-shift-of-VPSec),
$VPSec_2 = h$(2-bit-shift-of-VPSec),…,
$VPSec_n = h$(n-bit-shift-of-VPSec)
$CVSec_1 = h$(1-bit-shift-of-CVSec),
$CVSec_2 = h$(2-bit-shift-of-CVSec),…,
$CVSec_n = h$(n-bit-shift-of-CVSec))

Generating $CISec_i$

$CISec_1 = h$(1-bit-shift-of-(CDCI,CISec)),
$CISec_2 = h$(2-bit-shift-of-(CDCI,CISec)),…,
$CISec_n = h$(n-bit-shift-of-(CDCI,CISec))

3.4 Detailed Protocols

Our protocol consists of four sub–protocols: Registration, Purchase, Withdrawal and Deposit. Each sub–protocol has the following main functions:

Registration ($C \leftrightarrow V, C \leftrightarrow I$): This sub–protocol involves the customer, the vendor and the issuer. The process starts with the assignment of several nicknames to the client in order to protect her real identity when she communicates with the merchant. These nicknames are known only by the client and the issuer.

On the other hand, the customer shares her credit and/or debit-card information (CDCI) with her issuer. CDCI contains the longterm secret CISec known only by the customer and her issuer and will be used as an authentication method by the customer in future withdrawals. In addition, the secret SSWSec is shared between the customer/issuer and will be used as watermark value for the watermarking process at the issuer's side and as software input at customer's side to detect its authenticity.

When the first purchase takes place, *V* will detect if the wallet software is available in the mobile device. If not, *V* sends a software request to *P*, which will forward the request to *I*. The issuer intends to protect the software against various types of attacks carried away at any moment, following these steps: 1) First, choose one of the obfuscation methods proposed by [10] and apply it to the java code, and 2) Then, apply a watermarking process (proposed by [9]) to the software (using SSWSec as a watermark value and embedded into the software).

Once the software has been prepared, *I* will forward it to the *P*, which will send it to *V* , who will finally send it to *C*. After *C* receives the software, she will install it and check its authenticity using the secret SSWSec. If a problem occurs, *C* could abort the registration sub–protocol or start the process again.

When the software is successfully installed and working, *C* generates CVSec and send it to *V* with ID*C* and a nonce *n* encrypted with the session key K, generated by running AKE protocol with *V* . Then *V* sends $h(n, \text{CVSec})$ to *C* as a confirmation of customer's registration. After the sub–protocol has been completed, *C* and *V* can generate a new set of CVSec*i* by using the same key generation technique. On the other hand, the vendor registers herself to the Payment Gateway and share the secret VPSec.

1) C → V: $\{NID_C, \text{CVSec}, n\}_K$
2) V → C: $h(n, \text{CVSec})$

Purchase ($C \leftrightarrow V$): This sub–protocol is carried out between C and V over the wireless channel. The process starts when *C* sends to *V* the information necessary to set up the sub–protocol (step 3). After this information exchange ends, *C* builds up the Payment-script Request with *OI* and *TC*. Then, *C* encrypts it and sends to *V* where the message is decrypted to retrieve *OI*.

3) C → V: NID_C, i, TIDReq, VIDReq
4) V → C: $\{TID, ID_V\}_{CVSeci}$
5) C → V: $\{OI, Price, MAC[(Price, TC, h(OI), ID_V), CISeci]\}_{CVSeci}$, $MAC[(OI, Price, NID_C, ID_I), CVSec_{i+1}]$

Note that, although V can decrypt the message using *CVSeci*, she cannot generate this message since she does not have the necessary $CISec_i$ to construct MAC[(Price, TC, h(OI), ID_V), CISec*i*]. Thus, any entity of the mobile payment system can ensure that the message is truly sent from *C*.

Withdrawal (V ↔ P): Withdrawal sub–protocol occurs between *V* and *P* through a secure wired channel. *V* decrypts the message received from *C* (to retrieve *OI*), prepares the Withdrawal-script Request (including NID_C, ID_I, and the index *i* used to identify the current session

key in the set of CISec*i*) encrypted with VPSec*j* and then sends it to *P*. After the script was received by *P*, she forwards it to *I*, adding some information such her identity (ID_P). Here, this script is called Withdrawal–script Request and will be processed by *I* to approve or reject the transaction.

Once the issuer has processed the request and prepared the Withdrawal–script Response (including *Stt*), she must send it to *P* who in turn proceeds to forward to *V* . The Deposit sub–protocol is activated by *P* only when the Withdrawal is approved. Otherwise, *P* assigns the value Discarded to *Std*. After the Withdrawal and Deposit sub–protocols are completed, *P* sends the Withdrawal-script Response to *V* (including the Deposit-script Response). Then *V* prepares the Payment-script Response and sends it to *C*.

6) V → P: { MAC[(Price, TC, *h*(OI), ID_V), CISeci], j, ID_V , *h*(OI), i, TID, Price, NID_C, ID_I}$_{V\ PSecj}$, MAC[(*h*(OI), i, TID, NID_C, ID_I), VPSecj+1]

7) P → I: MAC[(Price, TC, *h*(OI), ID*V*), *CISeci*], i, *h*(OI), TID, Price, NID_C, ID_V, *h*(VPSecj+1)

8) I → P: Stt, *h*(Stt, *h*(OI), *h*(CISec*i*)), {*h*(OI), Stt, $h(VPSec_{j+1})\}_{CISeci}$

11) P → V: {Stt, {*h*(OI), $h(VPSec_{j+1})\}_{CISeci}$, *h*(Stt, *h*(OI), $h(CISec_i)$), Std, *h*(Std, *h*(OI))}$_{VPSecj+1}$

12) V → C: {{*h*(OI), Stt, $h(VPSec_{j+1})\}_{CISeci}$ }$_{CVSeci+1}$

Deposit (*P* ↔ *A*): This sub–protocol occurs between the *P* and *A* trough a secure wired channel when no problems have found at the Withdrawal sub–protocol. Here, the Deposit-script Request is prepared by *P* who sends it to *A* who checks the Price received with the negotiated during the purchase process. If they are matched, the value *Accepted* is assigned to *Std* and the total amount of the *OI* is transferred to the vendor's account. Otherwise, the deposit is refused (the value *Discarded* is assigned to *Std*) and it not represents an excuse for *V* to not deliver the good to *C* because the Withdrawal sub–protocol has been complete successfully. Then, a dispute occurs between *V* , *P* and *A*.

The Deposit-script Response is prepared by *A* and then sent to *P* in order to complete the deposit sub–protocol.

9) P → A: ID_P, Price, TID, Stt, *h*(OI), ID_V, $h(VPSec_{j+1})$

10) A → P: ID_A, Std, *h*(Std, *h*(OI))

After a transaction is completed, each entity of the payment system put in her revocations list, CVSec*i* and CISec*i* to prevent their replay from customer and vendor. In the following purchases, the registration sub–protocol will not occur until the customer is notified to update the secret CVSec. Thus, when become necessary to renew the secret, the customer runs the Registration sub–protocol to get a new CVSec. While the secret is not updated, the customer can use other values in the set of CVSeci to perform transactions. To update the VPSEC, the Payment Gateway sends the new secret to the vendor by using an AKE protocol. Finally, to update the CISec, the issuer has to add a message with the new secret to the Withdrawal-script Response which will be modified as following:

$\{h(OI), Stt, h(VPSec_{j+1}), NewSecret, h(NewSecret)\}_{CISeci}$

4. ANALYSIS

4.1 Comparison with SAMPP

In this section, we present a comparison between SAMPP and ours in order to establish the differences between both protocols.

The major difference between both protocols relies on the operational environment in which they are used. In SAMPP, the mobile device has access to the Internet which allows the client to communicate with the issuer when needed whereas our protocol is based on the idea of the consumer not being able to connect directly to the issuer, in consequence, any information or program that the issuer wants to send to the client, will have to do it through the vendor.

Another difference is the distribution method used with the payment software. While in SAMPP the customer must either download the software from the issuer or receive it by e-mail, in our proposal the wallet software must be sent from the issuer to the consumer through the vendor. This has lead us to the inclusion of security mechanisms (such as code obfuscation and watermarking) that assure the software against several types of attacks.

The third difference worth mentioning can be found in the number of sub–protocols that compose the protocol. SAMPP is composed of two sub–protocols whereas ours it is made up of four sub–protocols. In our protocol, each sub–protocol of the payment process is activated when it is needed (like the deposit sub–protocol that is activated when the issuer approves the withdrawal) and

unnecessary steps are avoided (as happens in SAMPP where the Payment Gateway must send the information to the issuer and the acquirer at the same time even though the withdrawal has not been approved).

The fourth difference can be found in the payment modes allowed by both protocols. In SAMPP, at the moment of the purchase, the client can use only his credit card whereas in ours, credit- or debit-card transactions are supported.

The last difference is the exchange of the secret shared between the client and the issuer (CISec). In the case of SAMPP, at the time of updating the CISec secret, a protocol AKE is used (among client/issuer) whereas in ours, the new secret must be sent inserted in the Withdrawal-script Response.

4.2 Security

Transaction Security: Our protocol satisfies the following transaction securities:

- *Entity authentication:* ensured by symmetric encryption and the secret CISec (which guarantees that the message is originated by the client).
- *Transaction Privacy:* ensured by the symmetric encryption.
- *Transaction Integrity:* ensured by MAC.

Anonymity: In order to prevent a merchant from knowing the identity of her clients, usage of client's nickname (NID_C) instead of her real identity is required during a communication from *C* to *V*. Since the C's nickname is known only by the client and the issuer, merchant cannot map the nickname and C's true identity. Thus, client's privacy is protected and untraceable.

Trust Relationships: Generally, in any transaction, a party should not trust others unless they can provide a proof of trustworthiness [6]. However, as in our protocol the issuer issues a credit- and/or debit-card to the client and she will not reveal it to any part, we state the trust relationship between the client and the issuer.

4.3 Performance

As SAMPP was reformulated to fit our needs, in this section we perform a comparison of both protocols in terms of performance, focusing on the number of cryptographic operations performed by each one (results of this comparison are shown in table 1). We can see that although operational models are different and our proposal is an evolution of SAMPP, the performance of our protocol is the same that of SAMPP.

Cryptographic Operations		SAMPP	Ours
1. Symmetric-key encryptions / decryptions	**C**	4	4
	V	5	5
	P	2	2
2.- Hash Function	**C**	2	2
	V	-	-
	P	-	-
3. Keyed-hash functions	**C**	2	2
	V	2	2
	P	1	1
4.- Key generations	**C**	2	2
	V	1	1
	P	1	1

Table 1. The number of cryptographic operations of SAMPP, and our protocol, respectively.

5. CONCLUSIONS

We have proposed a secure protocol which uses symmetric cryptographic techniques. It is applicable to mobile payment systems where direct communication between the client and the issuer does not exist. Thus, the client takes advantage of the infrastructure of the vendor and payment gateway to communicate with the issuer and purchase securely from her mobile device. Our proposal represents an alternative to all mobile payment systems where the connection between the client and issuer is mandatory, including Visa's 3-D Secure scheme. Moreover, our scheme illustrates how a portable device equipped with a short range link (such Bluetooth, Infrared or Wi-Fi) and low computational power is enough to interact with a vendor machine in order to buy goods in a secure way

The symmetric cryptographic technique used in our protocol has lower computation requirements at both parties (since no public-key operation is required) and offers the capability of dealing with protocol failures and disputes among parties. Moreover, we have shown that our protocol's performance is about the same than that of SAMPP, although this protocol is used in different operational models. As a result, we state that our proposed protocol allows mobile users to have efficient

and secure payment systems even if the communication with the issuer is not possible.

6. REFERENCES

[1] Al-Meaither, M. (2004): "Secure electronic payments for Islamic finance". PhD thesis, University of London.

[2] Visa International} (2002): "3-d secure mobile authentication scenarios version 1.0. [Online], Available: http://partnernetwork.visa.com/pf/3dsec/specifications.jsp.

[3] Ham, W., Choi, H., Xie, Y., Lee, M., and Kim, K. (2002): "A secure one-way mobile payment system keeping low computation in mobile devices". In The 3rd International Workshop on Information Security Applications (WISA), pp. 287--301.

[4] Chari, S., Chari, S., Kermani, P., Smith, S., and Tassiulas, L. (2001): "Security issues in m-commerce: A usage-based taxonomy". In: E-Commerce Agents, Marketplace Solutions, Security Issues, and Supply and Demand. Volume 2033 of Lecture Notes in Computer Science. Part II Security Issues, pp. 264-282.

[5] Zheng, X. and Chen, D. (2003): "Study of mobile payments system". In IEEE International Conference on Electronic Commerce (CEC), pp. 24-.

[6] Kungpisdan, S. (2004): "A secure account-based modbile payment system protocol". In International Conference on Information Technology: Coding and Computing (ITCC), pp.35-39.

[7] Bellare, M., Garay, J., Hauser, R., Herzberg, A., Krawczyk, H., Steiner, M., Tsudik, G., Herreweghen, E., Waidner, M.(2000): "Design, implementation and deployment of the iKP secure electronic payment system", IEEE Journal on Selected Areas in Communications, Vol. 18, No. 4, pp. 611–627.

[8] Bae S., Kang M., Lee S. (2003): "Authenticated Key Exchange Protocol Secure against Offline Dictionary Attack and Server Compromise, In Grid and Cooperative Computing, Second International Workshop (GCC)", pp. 924-931.

[9] Collberg, C., Carter, E., Debray, S., Huntwork, A., Kececioglu, J., Linn, C., Stepp, M. (2004): "Dynamic path-based software watermarking". In ACM SIGPLAN Conference on Programming Language Design and Implementation, pp. 107--118.

[10] Sosonkin, M., Naumovich, G., and Memon, N. (2003): "Obfuscation of design intent in object-oriented applications". In ACM workshop on Digital rights management (DRM), pp. 142-153.