Electron Commer Res (2012) 12:97–123
DOI 10.1007/s10660-011-9086-0

A lightweight secure mobile Payment protocol for vehicular ad-hoc networks (VANETs)

Jesús Téllez Isaac · Sherali Zeadally · José Sierra Cámara

Published online: 31 December 2011

Abstract In the last few years, many value-added applications (such as Payment services) in *Vehicular Ad hoc NETworks* (*VANETs*) have emerged. Although these applications offer great business opportunities they also introduce new concerns regarding security and privacy. Moreover, the wide range of scenarios (with or without connectivity restrictions) arising from vehicle-to-vehicle and vehicle-to-roadside communications have opened up new security challenges which must be considered by Payment system designers to achieve the same security capabilities independent of the scenario where Payment occurs. We designed and implemented a lightweight (using symmetric-key operations which requires low computational power) secure Payment protocol for those scenarios in VANETs and other mobile environments where the Merchant cannot communicate directly with the Acquirer (the Merchant's financial institution) to process the *Payment Request*. We also present practical performance results that can be achieved with the proposed Payment protocol.

Keywords Performance evaluation · Vehicular ad hoc networks · Payment protocol · Security · Implementation

J.T. Isaac
Computer Science Department (Facyt), Universidad de Carabobo, Av. Universidad, Sector Bárbula, Valencia, Venezuela
e-mail: jtellez@uc.edu.ve

S. Zeadally (✉)
Department of Computer Science and Information Technology, University of the District of Columbia, Washington, DC 20008, USA
e-mail: szeadally@udc.edu

J.S. Cámara
Computer Science Department, Universidad Carlos III de Madrid, Avda. de la Universidad, 30, 28911 Leganés (Madrid), Spain
e-mail: sierra@inf.uc3m.es

1 Introduction

Vehicular Ad hoc NETworks (VANETs), in which vehicles constitute the mobile nodes in the network, aim to provide communications among nearby vehicles (also known as Inter-Vehicle Communication (**IVC**)) and nearby roadside base-stations (also referred to as Vehicle-to-Roadside Communication (**VRC**)). Moreover, VANETs are envisioned to play an important role in the enhancement of road safety and driving experiences by providing numerous promising services (such as collision avoidance, cooperative driving, traffic optimization, lane-changing assistance, Payment services, location-based services, and infotainment).

The application space for vehicle-to-vehicle and vehicle-to-roadside communications is vast and opens up tremendous business opportunities for mobile commerce, the automotive industry, and the research community. VANET applications can be broadly divided into two major categories [34, 36, 45, 51], namely, *safety-related applications*, and *comfort-related applications*. In recent years, the industry and academia have concentrated their research efforts primarily on safety-related applications because of its importance to the automotive domain. However, it is expected that research on comfort applications (that also offer great business opportunities) will continue to attract the attention of researchers and designers to develop non-safety VANET-based applications.

To enable Payments in VANET environments, we need to design Payment systems that satisfy the additional requirements associated with vehicular ad hoc networks. As mentioned previously, both vehicle-to-vehicle and vehicle-to-roadside communications open up new security challenges [32, 38] that must be considered by Payment system designers to achieve the same security capabilities independent of the scenario where Payments occur.

A real-world scenario where Payments can occur in a vehicle-to-roadside communication can be illustrated as follows: a Client is on the road and stops at a Parking facility where the vehicle is left for a while. If the Client wants to pay with a credit or debit-card and the Merchant (the entity that has products or services to offer or sell) is not able to communicate with the merchants financial institution (also known as the Acquirer) to process the Payment due to the absence of the necessary infrastructure, the Client should take an active role in the Payment process and acts as a proxy to allow the communication between the Merchant and the Acquirer. Note that this situation creates a potential security problem because the Merchant cannot send any kind of messages directly to the Acquirer and has to do it through the Client (who should not be able to change the content of the messages but must keep evidence of the Payment).

Symmetric and asymmetric signature methods have been widely used to provide party authentication in electronic Payment systems (including mobile commerce) [16]. However, for those portable devices (such as the ones typically attached to an On-Board Unit (called **OBU**) [9, 39]) available on the market and not based on the Texas Instruments TMS320C55x processor family (which delivers high performance, peripheral options, small packaging, and low power dissipation for the implementation of asymmetric operations in a efficient way) [17], traditional asymmetric signature schemes make the signature computations very expensive and not suitable for them [31]. Therefore, asymmetric authentication schemes are not suitable for scenarios where an engaging party has connectivity restrictions, and consequently, communication with other parties (such a Certification Authority for verifying a certificate) is not possible during such Payment transactions.

The Payment protocol we propose in this work is specific for a scenario where there is no direct communication between the Merchant and the acquirer. We exploit a symmetric-based signature scheme to satisfy the security requirements of the protocol proposed in this work. Symmetric cryptography (which employs a shared key between two parties) provides message confidentiality, message integrity, and provide the authentication of participants in, and provides an alternative in the construction of secure protocols for mobile Payment systems. Moreover, symmetric-key operations do not require high computational power nor do they require additional communication processing steps.

In this work, we designed and implemented a secure Payment protocol that allows the Merchant to send a message to the Acquirer through a Client (who will not be able to decrypt the message) for authentication purposes. The proposed protocol, called the Client Centric Model Payment protocol for VANETs (henceforth referred to as the CCMS-VAN Protocol), supports both credit-card and debit-card transactions, protects the real identity of the Client during the Payment and employs symmetric-key operations for all participants to reduce both, the setup cost for the Payment infrastructure and the transaction cost. Moreover, the CCMS-VAN protocol can be used by a portable device attached to an Application Unit (called **AU**[1]).

A reason to have chosen mobile phone can be found in the following lines: A large percentage of vehicles on the road have to be equipped with a vehicle-to-vehicle or vehicle-to-roadside application before the application can be effective. Each year, only approximately 4–7% of the existing vehicles are replaced with new vehicles in the US. This means that it could take over 10 years to reach the critical mass when all new vehicles start to be equipped with the vehicle-to-roadside or vehicle-to-vehicle applications (a period longer than the time many people own or use a vehicle). Using mobile devices is an effective way to do vehicle-to-vehicle or vehicle-to-roadside since a large percentage of drivers are equipped with such types of devices.

The empirical performance of our proposed CCMS-VAN protocol is evaluated with actual mobile phones as the underlying implementation platform. By using these mobile phones, we demonstrate that our Client-side application can be installed on multiple heterogeneous Java™-enabled memory-constrained portable wireless handheld devices.

The rest of the paper is organized as follows. Section 2 presents related works relevant to this research. We present the contributions of this work in Sect. 3. In Sect. 4, we describe the design of the proposed Payment protocol for vehicle-to-roadside scenarios in VANETs. Section 5 presents the implementation of CCMS-VAN. We present performance evaluation results of our proposed Payment protocol in Sect. 6.

[1]An Application Unit may use the OBU's communication capabilities and can be an integrated part of a vehicle (permanently connected to an OBU) or could be a portable device such as a Personal Digital Assistant (PDA), a mobile phone or a gaming device that can dynamically attach to and detach from an OBU [9].

Finally, we make some concluding remarks in Sect. 7. In Sect. 4, we describe the design of the proposed Payment protocol for vehicle-to-roadside scenarios in VANETs. Section 5 presents the implementation of CCMS-VAN. We present performance evaluation results of our proposed Payment protocol in Sect. 6. Finally, we make some concluding remarks in Sect. 7.

2 Related work

Several studies [8, 13, 14, 29, 30, 48] have been conducted in recent years to improve the security of mobile Payment systems. Many of these efforts have also been dedicated to unify concepts and scenarios into frameworks that would be useful in the design of new electronic Payment systems. Moreover, these studies have considered the following methods to provide authentication in electronic Payment systems (including Mobile commerce (M-commerce): username/password, symmetric, asymmetric and elliptic curve cryptography, smart card, 2d bar code, and biometric methods. There are many authentication mechanisms and protocols based on these methods [3, 4, 11, 12, 15, 25, 26, 33, 35, 40, 47] but some of them do not offer enough security for M-commerce whilst symmetric and asymmetric signatures have been widely used for authentication purposes.

The taxonomy presented in [10] discussed many proposed M-commerce usage scenarios. The following models with communication restrictions have been identified:

- *Kiosk-Centric* (as shown in Fig. 1), where the Merchant acts as a proxy to allow the communication between the Client and the Issuer due to the communications restriction that prevent the direct communication between both entities.
- *Client-Centric* (as shown in Fig. 2), where the Merchant cannot communicate directly with the Acquirer. The Client acts as a proxy to allow the communication among the above entities.

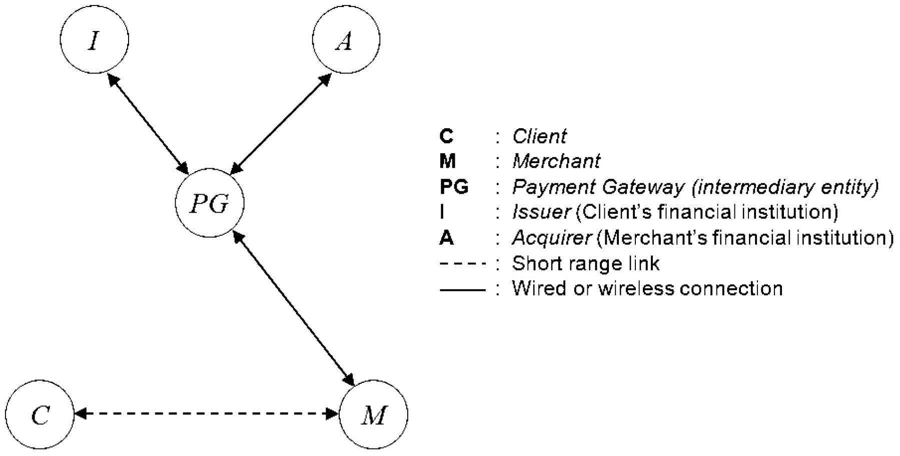

I
A
PG
C
M
C : Client
M : Merchant
PG : Payment Gateway (intermediary entity)
I : Issuer (Client's financial institution)
A : Acquirer (Merchant's financial institution)
- - - - : Short range link
—— : Wired or wireless connection

Fig. 1 Scenario with Merchant acting as a proxy: the Kiosk Centric model

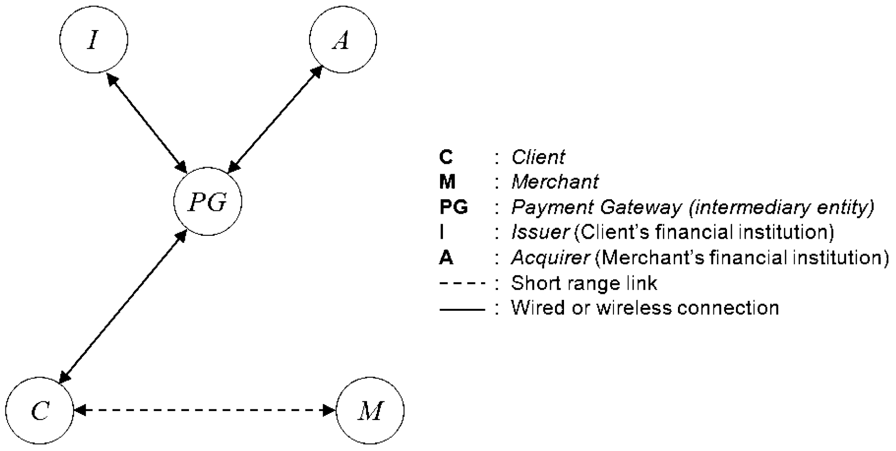

I
A
PG
C
M
C : Client
M : Merchant
PG : Payment Gateway (intermediary entity)
I : Issuer (Client's financial institution)
A : Acquirer (Merchant's financial institution)
- - - - : Short range link
—— : Wired or wireless connection

Fig. 2 Scenario with Client acting as proxy: Client Centric model

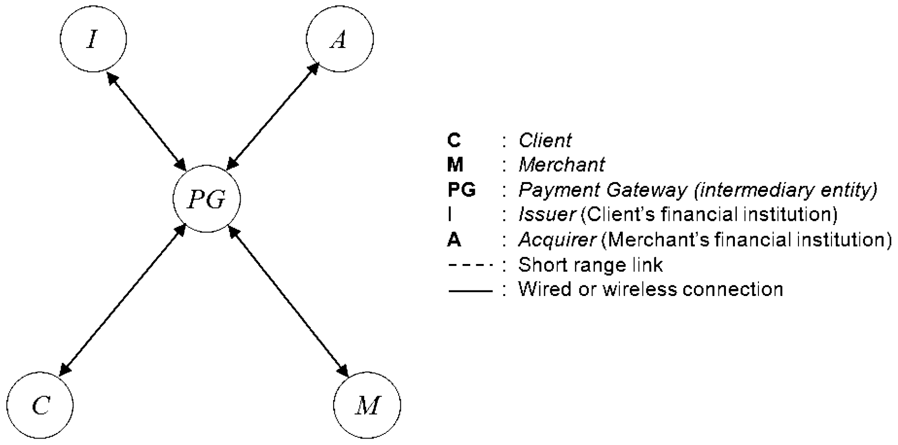

I
A
PG
C
M
C : Client
M : Merchant
PG : Payment Gateway (intermediary entity)
I : Issuer (Client's financial institution)
A : Acquirer (Merchant's financial institution)
- - - - : Short range link
—— : Wired or wireless connection

Fig. 3 Scenario with Payment Gateway as the intermediate between the Client and the Merchant

– *Server-Centric* (as shown in Fig. 3), where the Payment Gateway acts as an intermediary between the Client and the Merchant due to the absence of direct communication between the Client and the Merchant.

These scenarios with communication restrictions create new security and privacy challenges but offer great business opportunities for *Comfort applications* used in VANET.

The *Full Connectivity scenario* (as shown in Fig. 4) (where all the entities are directly connected one to another [10]) has been used for most of the protocols proposed in recent years. Most of them use asymmetric-key operations [5, 13, 14, 29, 48] whereas the remaining scenarios use symmetric-key operations which are more suitable for wireless networks. Unfortunately, many of these past proposed protocols

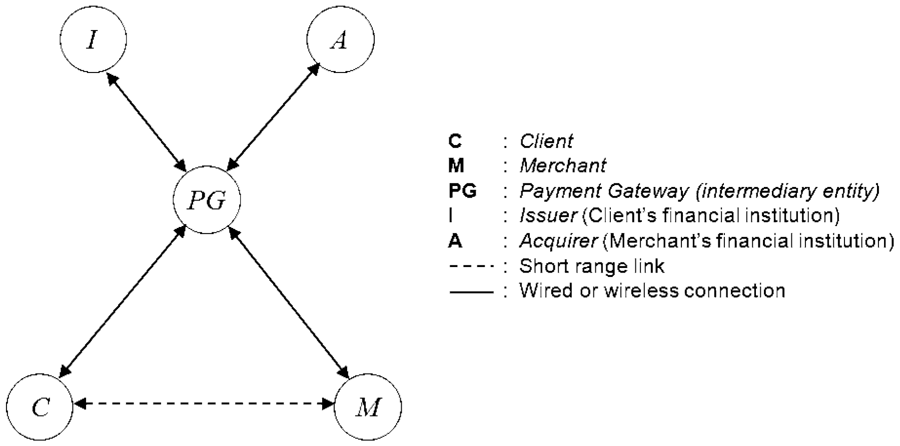

I
A
PG
C
M
C : Client
M : Merchant
PG : Payment Gateway (intermediary entity)
I : Issuer (Client's financial institution)
A : Acquirer (Merchant's financial institution)
---- : Short range link
—— : Wired or wireless connection

Fig. 4 Full connectivity scenario

cannot be used for scenarios with communication restrictions (as in the case for the Client Centric Model [10]) because they assume full connectivity to one another. It is therefore necessary to develop payment systems based on restricted connectivity scenarios along with additional goals such as enforcement of security and high performance levels that can be achieved in the case of full connectivity.

The above scenarios use the following entities:

- *Client*: a user who wants to purchase goods or services from the Merchant.
- *Merchant*: an entity that has products or services to offer or sell. This entity could be a computational one (such as a normal web server, a roadside computing station or an intelligent vending machine) or a physical one (such as a gas station that makes it possible to pay from within an AU) which the user can connect to using a short range link (using wireless technologies such as Wi-fi [46] or Bluetooth).
- *Acquirer*: is the Merchant's financial institution.
- *Issuer*: is the Client's financial institution.
- *Payment Gateway*: an additional entity that acts as in intermediate entity between the Acquirer and the Issuer on the bank's private network side and the Client/Merchant at the Internet side [27].

3 Contributions of this work

Recently, [41–44, 49] have proposed various secure Payment protocols suitable for those scenarios with communication restrictions. However, most of them have been theoretical proposals which did not capture practical performance issues we encounter with the deployment of actual Payment systems. For instance, in [43] we presented a theoretical proposal that makes use of a Digital signature with message recovery using self-certified public keys. One exception to these previous works is the Payment protocol presented in [44] which was actually implemented but for a scenario where direct interaction between the Client and the Issuer is not allowed because of the communication restriction imposed by the model (i.e. the Kiosk Centric model). The proposed protocol uses Symmetric cryptography and it was implemented in Java and using a Nokia™ N95 mobile device devices at the Client side.

In contrast to our previous proposals, in this work we present the design and implementation of our proposed secure Payment protocol (CCMS-VAN) based on Client Centric model [10] (as shown in Fig. 1) for VANET situations where the Merchant cannot communicate directly with the Acquirer. We evaluate the performance of the proposed protocol with an actual experimental testbed consisting of wireless connections and mobile devices. Moreover, since the Client is a mobile device that acts as a proxy, the implementation allows us to determine the additional computation cost when a Client is used as a proxy.

4 Proposed secure Payment protocol for vehicle-to-roadside scenarios in VANETs

4.1 Our proposed CCMS-VAN architecture

Taking into consideration the General Payment Model of Abad-Peiro et al. [1] which can be applied to many different Payment methods, we have designed a CCMS-VAN architecture (based on the Client Centric model) which uses the following entities: Client, Merchant, Acquirer, Issuer, and Payment Gateway (all of which were described earlier).

The parties of the Client Centric model communicate with each other when executing fund transfers using the following 3 primitive Payment transactions:

– In *Payment*, the Client transfers the Payment amount to the Merchant.
– In *Value Subtraction*, the Client requests that the Payment Gateway (on behalf of Issuer) deducts the money from the Client's account.
– In *Value Claim*, the Merchant requests that the Payment Gateway (on behalf of Acquirer) transfers money to the Merchant's account.

The five entities in CCMS-VAN and their interactions are shown in Fig. 5. Note that the Client is a user entity equipped with an **OBU** and/or an **AU**. Moreover, this entity connects directly with the Payment Gateway (an entity which provides the necessary infrastructure to allow a Merchant to accept credit card and other forms of electronic Payment), allowing the Merchant to communicate with the Acquirer using this connection.

It is worth noting that, in our proposed architecture, there is no direct interaction between the Merchant and the Acquirer. Moreover, the connection between the Client and the Payment Gateway is set up through the Internet, using communication technologies (wireless, cellular) offered by a mobile phone operator (such as General Packet Radio Service (**GPRS**), Enhanced Data rates for Global System for Mobile Communications (**GSM**) of Evolution (**EDGE**), Evolution-Data Optimized (**EvDO**) and High Speed Downlink Packet Access (**HSDPA**)). Since the Issuer, the Acquirer, and the Payment Gateway all operate over the private networks of banks, the security of the messages exchanged among them is beyond of the scope of this paper.

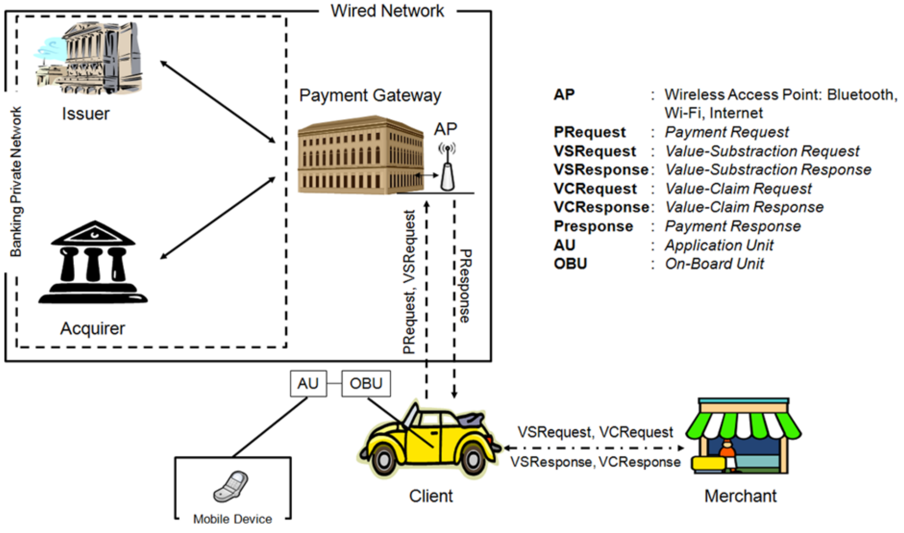

Wired Network
Banking Private Network
Issuer
Acquirer
Payment Gateway
AP
PRequest, VSRequest
PResponse
AU
OBU
Mobile Device
Client
VSRequest, VCRequest
VSResponse, VCResponse
Merchant
AP : Wireless Access Point: Bluetooth, Wi-Fi, Internet
PRequest : Payment Request
VSRequest : Value-Substraction Request
VSResponse : Value-Substraction Response
VCRequest : Value-Claim Request
VCResponse : Value-Claim Response
Presponse : Payment Response
AU : Application Unit
OBU : On-Board Unit

Fig. 5 Proposed CCMS-VAN design architecture

However, before receiving Payment services, the Client must register with an Issuer. During the Client's registration, the following steps are performed:

1. The Client shares his/her credit- and/or debit-card information (CDCI) with the Issuer (who will not reveal it to any Merchant).
2. The Issuer assigns several nicknames to the Client because of the trust relationship between both of them. These nicknames are known only to the Client and the Issuer and are used to prevent the Merchant from knowing the identity of the Client [16].

4.2 Notations

All the entities involved in our protocol are called parties and communicate through either wireless or wired networks or a combination of both. The symbols C, M, PG, I, and A are used to denote the names of the parties Client, Merchant, Payment Gateway, Issuer and Acquirer, respectively. Table 1 shows the symbols used to represent messages used in our proposed protocol.

4.3 Session key generation technique

The CCMS-Van protocol employs two efficient key generation techniques to generate the sets of session keys used in transactions and increase the performance of the protocol by reducing the frequency of key updates.

The key set KS_{C-M_i} (with $i = \{1, \ldots, n\}$), is generated from the secret key KS_{C-M} and stored in both the Client and the Merchant terminal. The set KS_{M-PG_k} (with $k = \{1, \ldots, n\}$), is generated from the secret key KS_{M-PG} and stored both in the Merchant and the Payment Gateway terminals. The set KS_{C-I_z} (with $z = \{1, \ldots, n\}$),

Table 1 Symbols and messages used by our proposed payment protocol

ID_P	The identity of party P that contains the contact information of P
NID_C	Client's nickname, temporary identity
TID	Identity of transaction that includes time and date of the transaction
TST_P	Timestamp generated by P
Stt	The status of transaction ($Stt = \{Accepted, Rejected\}$)
OD	Order description
$Price$	Amount and currency
OI	Order information ($OI = \{TID, OD, h(OD, Price)\}$)
TC	The type of card used in the purchase process ($TC = \{Credit, Debit\}$)
$TIDReq$	The request for TID
$MIDReq$	The request for Merchant Identity (ID_M)
SEC_{A-B}	The master secret shared between parties A and B
$\{M\}_x$	The message M symmetrically encrypted with the shared key x
$h(M)$	The one-way hash function of the message M
$MAC(X, K)$	Message Authentication Code of the message X with the key K
KS_{A-B_t}	The session key shared between parties A and B, generated applying a hash function with t-bit cyclic shifting (either left shift or right shift) of KS_{A-B}
"*PRequest*"	Payment Request
"*PResponse*"	Payment Response
"*VSRequest*"	Value-Substraction Request
"*VSResponse*"	Value-Substraction Response
"*VCResquest*"	Value-Claim Request
"*VCResponse*"	Value-Claim Response

is generated from the secret key secret KS_{CI} and is stored in the Clients device and the terminal of the Issuer. The set KS_{CPG_j} (with $j = \{1, \ldots, n\}$), is generated from the secret key KS_{CPG} and is stored both in the Client and Payment Gateway terminals.

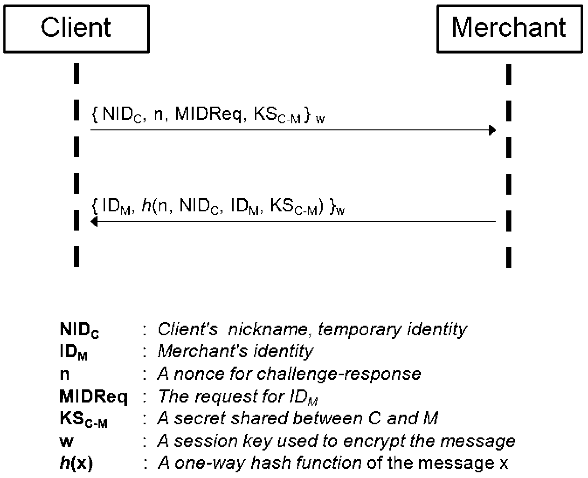
A more in-depth discussion of the two key generation techniques and the generation of the different sets of session keys are given in [44].

4.4 Our proposed Client Centric model Payment Protocol for VANETs (CCMS-VAN)

The CCMS-VAN Protocol is composed by two sub-protocols: the *CCMS-VAN Merchant Registration Protocol* (called *MRP* and it is executed between the Client **C** and the Merchant **M**) and the *CCMS-VAN Payment Protocol* (called *PP* and is executed among the Client **C**, the Merchant **M** and the Payment Gateway **PG**).

For the *CCMS-VAN Merchant Registration Protocol*, the Client has to register with the Merchant to send the master key KS_{CM}. The protocol has to be executed every time the Client wants to perform transactions with a Merchant. The details of the protocol are shown as follows:

Fig. 6 Messages exchanged during the execution of the *Merchant Registration Protocol*


Client
Merchant
{ NIDC, n, MIDReq, KSC-M } w
{ IDM, h(n, NIDC, IDM, KSC-M) }w
NIDC : Client's nickname, temporary identity
IDM : Merchant's identity
n : A nonce for challenge-response
MIDReq : The request for IDM
KSC-M : A secret shared between C and M
w : A session key used to encrypt the message
h(x) : A one-way hash function of the message x

$$\begin{aligned}\mathbf{C} \rightarrow \mathbf{M}&: \{NID_C, n, MIDReq, KS_{C-M}\}_w\\ \mathbf{M} \rightarrow \mathbf{C}&: \{ID_M, h(n, NID_C, ID_M, KS_{C-M})\}_w\end{aligned}$$

The master key KS_{CM} is generated by the Client **C** and shared with the Merchant. To achieve that, **C** sends the master key with her/his nickname NID_C, a nonce n for the challenge-response and *MIDReq* to **M**. After the Merchant **M** receives the message, she/he sends $h(n, NID_C, ID_M, KS_{C-M})$ and the Merchants identity (ID_M). Note that both messages (from the Client to the Merchant and vice-versa) are encrypted with the session key w. The various messages exchanged between the Client and the Merchant during the *CCMS-VAN Merchant Registration Protocol* are shown in Fig. 6.

Once **C** and **M** have exchanged the necessary information, they can generate a new set of KS_{C-M_i} using the same key generation technique. The Client may then start the *CCMS-VAN Payment Protocol.*

For the *CCMS-VAN Payment Protocol*, the Client purchases goods from the Merchant and pays for them using her/his credit-card or debit-card. This protocol is formalized as follows:

$$\begin{aligned}(1)\ \mathbf{C} \rightarrow \mathbf{M}&: NID_C, i, TIDReq\\ \mathbf{M} \rightarrow \mathbf{C}&: \{TID, ID_M\}_{KS_{C-M_i}}\end{aligned}$$

Step 1: The Client **C** and Merchant **M** exchange the information necessary to start the protocol by performing the following sub-steps.

1-1: **C** sends his/her nickname (NID_C), the index i (that will be used to generate the session key between the Client and the Merchant) and the request for the transaction identity (*TIDReq*) to **M**.

1-2: The Merchant receives the request and sends back its identity (ID_M) and *TID* to **C**, encrypted with KS_{C-M_i}.

$$\text{(2) } \mathbf{C} \rightarrow \mathbf{M}\text{:} \quad \{OI, Price, NID_C, ID_I, TST_C, VSRequest\}_{KS_{C-M_i}}, \\ MAC[(OI, Price, NID_C, ID_I, TST_C), KS_{C-M_{i+1}}]$$

$$VSRequest = (MAC[(Price, h(OI), TST_C, TC, ID_M), KS_{C-I_z}], \\ TC, TST_C)$$

Step 2: Client **C** creates a *Payment Request* (referred to in the General Payment Model [1, 27]) using the following sub-steps.

2-1: A *Value-Subtraction Request* (called *VSRequest*) is created and it includes $MAC[(Price, h(OI), TST_C, TC, IDM), KS_{C-I_z}]$, TST_C and TC.

2-2: A new message is created which includes $C's$ nickname, $I's$ identity, *Price*, *OI* (used to inform **M** about the goods and prices requested), *VSRequest* and the timestamp TST_C read from $C's$ clock.

2-3: The message created in the previous sub-step (henceforth referred to as the Payment Request) is encrypted with the session key KS_{C-M_i}.

2-4: The *Payment Request* is sent to the Merchant.

$$\text{(3) } \mathbf{M} \rightarrow \mathbf{C}\text{:} \quad \{VCRequest, ID_M\}_{KS_{C-M_i}}, MAC[(VCRequest, ID_M), KS_{C-M_{i+1}}]$$

$$VCRequest = (\{VSRequest, TST_M, h(OI), TID, Price, NID_C, ID_I\}KS_{M-PG_K}, \\ MAC[(VSRequest, TST_M, h(OI), TID, Price, NID_C), KS_{MPG_{k+1}}])$$

Step 3: The Merchant **M** generates the *Value-Claim Request* (called *VCRequest*) by performing the following sub-steps.

3-1: The message received from C is decrypted to extract *OI*, TST_C and *VSRequest*.

3-2: The timeliness of the *Payment Request* is verified. If the check is successful, the following sub-steps are performed.

3-3: The *VCRequest* is prepared, and contains $C's$ nickname, $h(OI)$, TST_M, the *VSRequest*, order's amount, identity of the transaction, ID_M, $I's$ identity and $MAC[(VSRequest, TST_M, h(OI), TID, ID_M, Precio, NID_C), KS_{M-PG_{k+1}}]$.

3-4: The *VCRequest* and the $M's$ identity are encrypted with KS_{C-M_i}.

3-5: The encrypted message in sub-step 3-4 is then transmitted to the Client **C** with $MAC[(VCRequest, ID_M), KS_{C-M_{i+1}}]$.

$$\text{(4) } \mathbf{C} \rightarrow \mathbf{PG}\text{:} \quad \{VCRequest, ID_M, NID_C, k, z, h(KS_{C-I_z})\}_{KS_{C-PG_J}}, \\ j, MAC[(VCRequest, ID_M, NID_C, k, z, h(KS_{C-I_z})), \\ KS_{C-PG_{j+1}}].$$

Step 4: The Client **C** performs the following sub-steps.

4-1: The received message from **M** is decrypted to retrieve the *VCRequest*.

4-2: A new message is created (that includes ID_M, NID_C, the received *VCRequest*, indices k y z and $h(KS_{C-I_z})$ and is used to prevent the Payment Gateway from modifying the approval result in step 5-5) and is encrypted with KS_{C-PG_j}.

4-3: The last message encrypted in step 4-2 is sent to **PG** with the index j and $MAC[(VCRequest, IDM, NID_C, k, z, h(KS_{C-E_z})), KS_{C-PG_{j+1}}]$.

(5) Using the banking private network,
(5.1) **PG** $\rightarrow$ **I**: $NID_C, ID_M, VSRequest, TID, h(OI), z, Price, h(KS_{C-I_z})$
(5.2) **PG** $\rightarrow$ **A**: $Price, ID_M$
(4.3) **I, A** $\rightarrow$ **PG**: $VSResponse, Stt, h(Stt, h(OI), h(KS_{C-I_z}))$
$VSResponse = \{Stt, h(OI), h(KS_{M-PG_{k+1}})\}_{KS_{C-I_z}}$

Step 5: Using the private network of the banking institution, the Payment Gateway (PG) performs the following sub-steps to verify and approve the Payment.
5-1: The *VCRequest* is decrypted to retrieve *VSRequest* and the others fields, such as ID_M, NID_C.
5-2: The timeliness of *VCRequest* is verified. If the check is successful, the following steps are executed.
5-3: The *VSRequest* and other important, such as: $h(OI)$, TID, ID_M, $Price$, z and $h(KS_{C-I_Z})$ are forwarded to the Issuer (**I**) where it is decided whether to approve or reject the transaction.
5-4: ID_M and the requested price *Price* are sent to confirm to the Acquirer **A** that the Merchant is the party to whom the requested amount *Price* will be transferred to.
5-5: The approved result (*Stt*) and *Value-subtraction Response* (called *VSResponse* and encrypted with KS_{C-I_z}) are received from the Issuer **I**. It is worthwhile noting that the *VSResponse* is prepared by the Issuer after (a) checking the timeliness of *VSRequest* and the validity of the Clients account, and (b) after transferring the total amount of OI to the Merchants account.

(6) **PG** $\rightarrow$ **C**: *PResponse*
$VCResponse = \{Stt, h(Stt, h(OI))\}_{KS_{M-PG_{k+1}}}$
$PResponse = \{VSResponse, VCResponse\}_{KS_{C-PG_{j+1}}}$

Step 6: The Payment Gateway **PG** generates the *Payment Response* (called *PResponse*) in the following sub-steps.
6-1: The *VCResponse* is created (including *Stt* and $h(Stt, h(OI))$) and encrypted with $KS_{M-PG_{k+1}}$.
6-2: The *Payment Response* (called *PResponse* and encrypted with $KS_{C-PG_{j+1}}$) is created and sent to the Client. *PResponse* includes the *VSResponse* and the *VCResponse* (which will be forwarded to the Merchant **M**).

(7) **C** $\rightarrow$ **M**: *VCResponse*

Step 7: The Client **C** performs the following sub-steps.
7-1: The *PResponse* is decrypted to retrieve the *VSResponse* and *VCResponse* primitives.

7-2: The Clients own *OI* is compared with the received $h(OI)$. If they do not match, then the Client performs Sub-step 7-3a, otherwise the Client performs Sub-step 7-3b.

7-3a: A message is sent to the Payment Gateway to notify it of the response failure. The Payment Gateway then starts a recovery procedure or resends the message.

7-3b: The *VCResponse* is sent to M who in turn proceeds to deliver the goods to the Client.

After a transaction is completed, KS_{C-M_i}, KS_{C-I_z}, KS_{M-PG_k} y KS_{C-PG_j} are put in the revocations list of every entity of the system to prevent their replay between the Client and the Merchant. Figure 7 shows the transmitted messages among the parties of the system during the execution of our proposed *CCM-VAN Payment Protocol*.

4.5 Security analysis

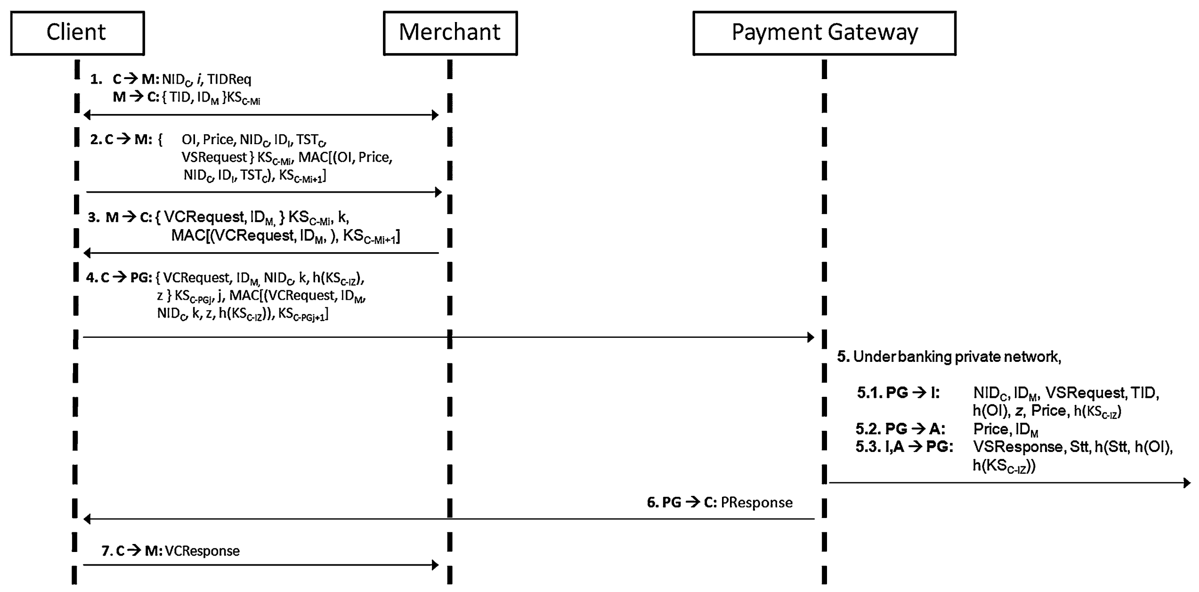
In this section, we performed a detailed security analysis of our proposed secure Payment protocol. The anonymity of the Client in the CCMS-VAN protocol is achieved by using a nickname NID_C (a temporary identity known only to the Client and the Issuer) instead of his/her real identity. As a result, neither the Merchant nor the Payment Gateway can map the nickname to the true identity of the Client. Note that this anonymity protects relevant information from third parties but not unrestrained anonymity because it only implies the protection of relevant information from unintended parties [2].

The *Confidentiality* of messages transmitted in each transaction while in transit in the proposed protocol, is protected by employing symmetric cryptography which uses a secret shared key between the two parties (called sender and receiver) that wish to communicate safely without revealing the content of the message. Moreover, the encryption key also allows the receiver and the sender to authenticate each other. In addition, the proposed protocol uses the Message Authentication Code (MAC) to maintain the *Integrity* of important messages.

Although, generally, in any transaction a party should not trust others unless they can provide a proof of trustworthiness [27], the proposed protocol assumes that the trust relationship between the Client and the Issuer exists because the Client has a credit- and/or debit-card issued by the Issuer who will not reveal it to any other party.

The *Non-repudiation* of a transaction is ensured in the proposed protocol by KS_{C-I_Z}, since it could be generated only by the Client or Issuer but not by the Merchant. Thus, the Merchant can provide a non-repudiable evidence to prove to other parties that the Client has sent a message or requested the Merchant to perform a transaction.

Our proposed protocol is also secure against *replay attacks* because the timestamp included in the transmitted message ensures the freshness of the message and prevents an intruder E from impersonating a legal user by replaying the users transmitting contents. Moreover, it is difficult for any intruder E to get information related to the secret key through an analysis of intercepted data because the authentication key is dynamic (i.e., exploiting, on each transaction, different session keys from one master secret) which makes the proposed protocol secure against *key guessing attacks*.


Client
Merchant
Payment Gateway
1. C → M: NID$_C$, *i*, TIDReq
M → C: { TID, ID$_M$ }KS$_{C-Mi}$
2. C → M: { OI, Price, NID$_C$, ID$_I$, TST$_C$, VSRequest } KS$_{C-Mi}$, MAC[(OI, Price, NID$_C$, ID$_I$, TST$_C$), KS$_{C-Mi+1}$]
3. M → C: { VCRequest, ID$_M$, } KS$_{C-Mi}$, k, MAC[(VCRequest, ID$_M$,), KS$_{C-Mi+1}$]
4. C → PG: { VCRequest, ID$_M$, NID$_C$, k, h(KS$_{C-IZ}$), z } KS$_{C-PGj}$, j, MAC[(VCRequest, ID$_M$, NID$_C$, k, z, h(KS$_{C-IZ}$)), KS$_{C-PGj+1}$]
5. Under banking private network,
5.1. PG → I: NID$_C$, ID$_M$, VSRequest, TID, h(OI), *z*, Price, h(KS$_{C-IZ}$)
5.2. PG → A: Price, ID$_M$
5.3. I,A → PG: VSResponse, Stt, h(Stt, h(OI), h(KS$_{C-IZ}$))
6. PG → C: PResponse
7. C → M: VCResponse

NID$_C$	: *Client's nickname, temporary identity.*	***i, k, z, j***	: *Indexes.*
ID$_M$	: *Merchant's identity.*	**TIDReq**	: *Transaction Identity Request.*
OI	: *Order information.*	**VSRequest**	: *Value-Substraction Request.*
Price	: *Order's amount.*	**VSResponse**	: *Value-Substraction Response.*
TID	: *Identity of transaction.*	**VCRequest**	: *Value-Claim Request.*
TST$_C$	: Client's Timestamp.	**VCResponse**	: *Value-Claim Response*
Stt	: *Transaction's status.*	**PResponse**	: *Payment Response.*
KS$_{A-Bt}$	: *The session shared between **A** and **B***	***h*(x)**	: *A one-way hash function* of the message **x**.
MAC[x,k]	: Message Authentication Code of the message **x** with the key **k**.		

Fig. 7 Message exchange during the execution of the *CCMS-VAN Payment Protocol*

Table 2 Hardware specifications of the systems used in our performance evaluation tests of the proposed protocol

Protocol component	Device	Features
– Client	Nokia™ N95	– 332 MHz Texas Instruments OMAP 2420 (ARM11-based)
		– 160 MB of RAM
		– Symbian OS 9.2, S60 rel. 3.1.
– Merchant	Sony™ Vaio VGN-SZ450N	– Intel Core 2 Duo (2 GHz)
– Payment Gateway		– 2 GB of RAM
– Acquirer		– Windows Vista Business
– Issuer		– Sun's Java Virtual Machine

5 Design and implementation of CCMS-VAN Protocol

5.1 Experimental testbed platform

As mentioned before, the CCMS-VAN implementation is composed of 5 applications (Client, Merchant, Payment Gateway and Issuer) which have been implemented in Java Platform 2, Standard Edition (**J2SE**) [23], except for the Client that was implemented in Java Platform, Micro Edition (**JavaME**) [22] for mobile devices. Since the JaveME Mobile Information Device Profile (MIDP) version 2.0 does not have the necessary security support, we have used security APIs from [7], a light-weight API suitable for mobile computing applications and resource-constrained devices. Thus, all the five applications implemented use the same security APIs.

The wireless communication between the client and merchant is established using TCP/IP over the 802.11b channel with a maximum transfer rate of 11 Mbit/s and Wired Equivalent Privacy (**WEP**) encryption with a 64 bits key.

The hardware configuration of the devices used to execute and evaluate all the components of our proposed payment protocol are shown in Table 2.

5.2 Cryptographic operations

Two important design aspects that should be considered when choosing encryption algorithms and hash functions are their *security* features and their *computational* requirements. Taking into consideration these requirements and the results discussed in [44] (about the comparison of different cryptographic algorithms), Table 3 shows the cryptographic algorithms used by the implementation of CCMS-VAN protocol.

The cryptographic algorithms shown in Table 3 are contained in the Java class named `cCryptography` of the Client application and implemented in the following methods:

– *generateAESKey*(): This method creates an AES key to be used with the cryptography algorithms implemented in the wallet application. We have used the implementation described in [24].

Table 3 Cryptographic operations used by CCMS-VAN protocol

– Symmetric-key algorithm	We use the *Advanced Encryption Standard* (*AES*) [21, 37] algorithm with 128-bit key because it provides higher security, faster operation, and lower energy consumption compared to *Triple Data Encryption Standard algorithm* (*3DES*) [19, 20]
– Hash function	We use the *MD5 Message-Digest Algorithm* (MD5 algorithm) [19] because it requires less computation and consumes less energy than *Secure Hash Algorithm version 1* (*SHA-1*) [19], and produces the same length of output as the AES key (128 bits)
– Keyed-hash function	We use the *Hashed Message Authentication Code with Message Digest 5* algorithm (HMAC-MD5) [18] because the key length of HMAC-MD5 is equivalent to the length of each session key and is considered a secure keyed-hash algorithm available for wireless networks [6]

- *AESEnc*(): Method used to encrypt messages with the *Advanced Encryption Standard* algorithm.
- *AESDec*(): Method used to decrypt messages encrypted with *AESEnc*().
- *HMAC-MD5*(): This method is used to calculate a Message Authentication Code (MAC) involving a cryptographic hash function in combination with a secret key. Thus, the result can be used to simultaneously verify both the data integrity and the authenticity of a message.
- *generateItemofSet*(): Method that allows to create a session key (using the Session Key Generation Technique presented in Sect. 4.3) from a master key.

5.3 The CCMS-VAN software at the Client side

With the CCMS-Van Payment protocol, a software (henceforth referred to as the CCMS-VAN Wallet) is required at the Client's side for purchase transactions. The CCMS-VAN Wallet can be obtained by connecting to the Issuers web site and downloading it or sending a request to the Issuer to receive it by mail. Once the Client has downloaded or received the CCMS-VAN wallet, she/he should install it on her/his mobile device. To prevent unauthorized users from: (a) opening the application, (b) authorizing Payment transactions, or (c) accessing Clients information and key files, the Client is required to set up her/his own password (henceforth referred to as *KWP* password) during the installation process. Figure 8 illustrates snapshots of the main screen, the login screen, and the main menu of CCMS-VAN wallet on a Nokia™ N95 device. The left screen shows the main screen of CCMS-VAN wallet, the middle screen shows the login screen and the right screen shows the main menu.

The CCMS-VAN wallet provides the following functionalities:

- **Personal Details**: To prevent the Client from being prompted for her/his information during both the *Merchant Registration* and *Payment* phases, the CCMS-VAN wallet allows the Client to store in a file (protected by the CCMS Wallet Password (*KWP*)) his/her personal information (such as name, contact information, Issuer ID and credit-and/or debit-card information) for other purposes. To achieve this, the CCMS-VAN wallet prompts the user to enter the above information and store it at

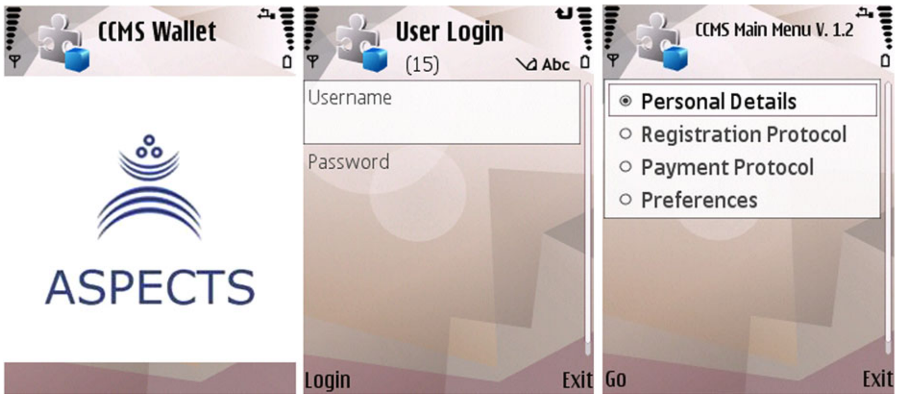
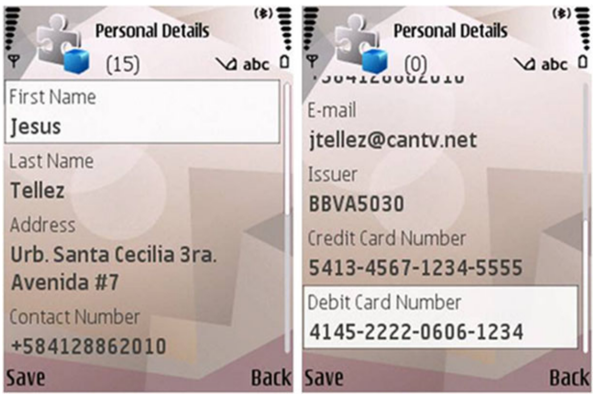


Fig. 8 Snapshots of main screen, the login screen and the main menu of the CCMS-VAN wallet on the Nokia™ N95 device



First Name
Jesus
Last Name
Tellez
Address
Urb. Santa Cecilia 3ra. Avenida #7
Contact Number
+584128862010
E-mail
jtellez@cantv.net
Issuer
BBVA5030
Credit Card Number
5413-4567-1234-5555
Debit Card Number
4145-2222-0606-1234

Fig. 9 Snapshots of CCMS wallet Personal Details form on Nokia™ N95 device

the Clients mobile device. Figure 9 illustrates the CCMS-VAN wallet prompting the Client to enter her/his details on Nokia™ N95 device.

– **Key Generation**: To generate a new session key from the master key, the CCMS-VAN wallet calls *generateItemofSet*() with two values: a master key and a random number j. Upon receipt of the master key, its Big Integer[2] representation is created and depending on the value of j, the required number of zeros is added to the right of the master key. The MD5 function is then applied to the result obtained to produce a new 128-bit session key.

The aforementioned key generation technique can be directly applied to generate the sets of session keys KS_{CI} and KS_{MPG_k} from the keys KS_{CM} and KS_{MPG}, respectively. Moreover, to generate the set of session keys KS_{CI}, the Credit- and/or Debit-Card Information (CDCI) is treated as the Big Integer and is added to the value of KS_{C-I} before doing performing a left shift operation. An example of the proposed session key generation technique is shown in Fig. 10.

[2]The BigInteger class is a library available in JAVA which allows the representation of very large numbers.

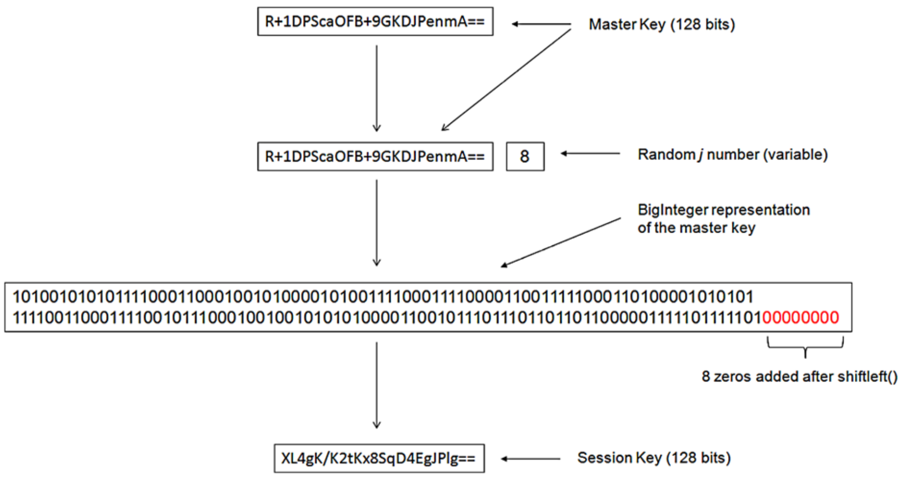


Fig. 10 An example of the key generation technique used by the CCMS wallet

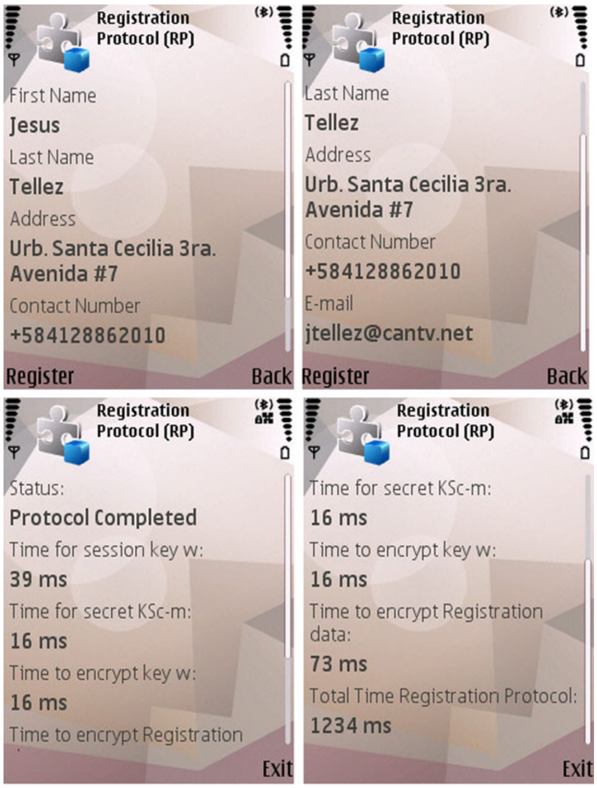
Table 4 Client registration message format

Fields	Size (bits)
Client Data	Variable (maximum 1064)
KS_{C-M}	128
Nonce	128
MIDReq	48

– **Merchant Registration**: The Client should execute the *Merchant Registration Protocol* to register with the Merchant to share the master key KS_{CM} before making Payments to a Merchant. First, the Client is prompted to enter the *KWP* password to retrieve her/his personal information (such as name, address, contact number and email address) stored on the Clients device. Then, the CCMS-VAN wallet generates the secret key KS_{CM} and the session key by using the *generateAESKey*().Once the keys have been generated, the CCMS-VAN wallet encrypts using the session key w by calling the *AESEnc*(), the Client's personal information, the secret KS_{CM}, a nonce, and a request for the Merchants identity before being sent to the Merchant to register the Client. The format of the Client registration message is shown in Table 4.

Upon receipt of the Client registration message, the Merchant decrypts it using *AESDec*() to retrieve the Clients information including the secret key KS_{CM}. The Merchant then encrypts her/his identity and $h(n, NID_C, ID_M, KS_{CM})$ with the session key w using *AESEnc*(). The encrypted message is sent to the Client to confirm the registration.

At the Client side, the Client retrieves the confirmation of her/his registration by decrypting the message using *AESDec*(). Then, the CCMS-VAN wallet stores the secret key KS_{CM} in a key file protected with the *KWP* password. Once the registration is done, the Merchant stores on its device the Client's information together with the secret key KS_{CM}.



Registration Protocol (RP)

First Name
Jesus
Last Name
Tellez
Address
Urb. Santa Cecilia 3ra. Avenida #7
Contact Number
+584128862010

Register Back

Registration Protocol (RP)

Last Name
Tellez
Address
Urb. Santa Cecilia 3ra. Avenida #7
Contact Number
+584128862010
E-mail
jtellez@cantv.net

Register Back

Registration Protocol (RP)

Status:
Protocol Completed
Time for session key w:
39 ms
Time for secret KSc-m:
16 ms
Time to encrypt key w:
16 ms
Time to encrypt Registration

Exit

Registration Protocol (RP)

Time for secret KSc-m:
16 ms
Time to encrypt key w:
16 ms
Time to encrypt Registration data:
73 ms
Total Time Registration Protocol:
1234 ms

Exit

Fig. 11 Snapshots of CCMS-VAN wallet during the execution of the Merchant Registration Protocol phase on Nokia™ N95 device

Figure 11 illustrates the CCMS-VAN wallet during the Merchant Registration phase on the Nokia™ N95 device. The left screen shows the CCMS wallet displaying the Client's details whereas the right screen shows the successful registration.

– **Payment Execution**: To make a payment to the Merchant, the Client is first prompted to enter the following information: Order Description, the Product ID, the Price and the Type of Card to use (*Debit* or *Credit*). After providing the information, the CCMS-VAN wallet application generates the keys KS_{C-M_i} and KS_{C-I_z} based on the random numbers i and z, respectively. The Client then sends his/her nickname, the i, and the Transaction ID Request (**TIDReq**). Upon receipt of the message, the Merchant generates the key KS_{C-M_i} (based on the i value) and sends his/her identity ID_M) and the transaction ID (**TID**) to the Client, encrypted with the session key KS_{CM_i} (using *AESEnc*()). The Client then creates the *Value-Subtraction Request* (*VSRequest*) that is sent to the Issuer. An authenticated hash of this message is computed using the HMAC-MD5 algorithm with the KS_{CI_z} key by using *HMACMD5*(). It is worth pointing out that the Merchant will not able to decrypt or create the request since the Merchant does not have the key KS_{C-I} which is known only the Client and the Issuer. Once the *VSRequest* has been created, the Client prepares the Payment Ordering request (called *PRequest*) and encrypts it with the key KS_{CM_i} using *AESEnc*() to ensures its confidentiality. The *PRequest* is sent to the Merchant and includes the *VSRequest*, the Order Information (**OI**), Price, Client Nickname (NID_C), timestamp TST_C and the Issuer identity (ID_I). Upon receipt of the request from the Client, the Merchant (who has the key KS_{CM}) decrypts the message using *AESDec*(). Then, the Merchant combines the *VSRequest* received from the Client and the amount payable by the Client with the necessary information to create the *VCRequest*. This message is encrypted with the key KS_{CM_i} before being sent to the Client. Note that the Merchant needs to send the message through the Client due to the connectivity restriction that prevents direct communication between the Merchant and the Acquirer.

The message received from the Merchant is decrypted using *AESDec*() to recover the *VCRequest*. The Client then creates a message which includes the received *VCRequest* and other necessary information (such as ID_M, NID_C, k, z and $h(\mathit{KS}_{CI_z})$). This message is encrypted with the key KS_{CPG_j} using *AESDec*() before being sent to the Payment Gateway (PG). PG decrypts the message received from the Client using *AESDec*() and then sends the *VSRequest* to the Issuer. Upon receipt of the approval response, the PG prepares the *PResponse* (which includes *VCResponse* and *VSResponse* (encrypted with $\mathit{KS}_{MPG_{k+1}}$)) and encrypts it with the key $\mathit{KS}_{CPG_{j+1}}$ using *AESEnc*(). PG then sends the Payment Ordering Response (*PResponse*) to the Client which in turn transmits the *VCResponse* (recovered from the message received from PG, using *AESDec*() to decrypt the message). Note that although the flow information between the Payment Gateway, the Issuer, and the Acquirer (step 5 of CCMS-VAN protocol shown in Fig. 7) exists within a private banking network (beyond the scope of CCMS-VAN Protocol), still, we have implemented the Issuer and Acquirer to have a better idea of the performance of the entire Payment system even if it could be assumed that all transactions relevant to Acquirer and Issuer are successful within a limited time as provided by the private banking network.

Once the Merchant receives the message from the Client, it retrieves the *VCResponse* using *AESDec*() to decrypt the message. The format of PRequest, VSRequest, VSResponse, VCRequest and VCResponse primitives are shown in Table 5.

Figure 12 illustrates the CCMS-VAN wallet during the Payment phase on Nokia™ N95 device. The left screenshot shows the CCMS-VAN wallet displaying the Order description whereas the right screen shows the successful Payment transaction.

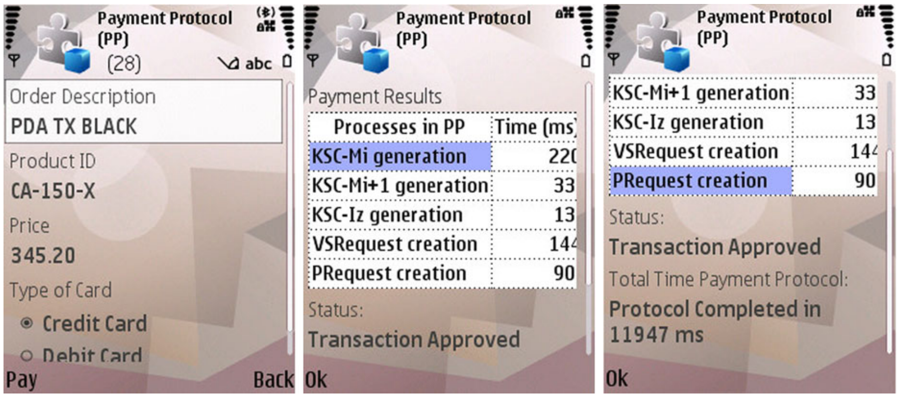
6 Performance evaluation

6.1 Discussions of empirical results

In this section, we present an empirical performance analysis of our proposed CCMS-VAN protocol using performance metrics such as execution time for the various execution phases (such as merchant registration protocol, payment protocol) of the protocol and the size of application code which is an important factor for resource-constrained devices (e.g. those with limited memory).

Table 5 Format of Payment primitive transactions PRequest, VSRequest, VCRequest, VCResponse and VSResponse

Primitive	Fields	Size (bits)
Payment Request (PRequest)	OI	Variable
	Price	
	Client Data	
	Issuer Data	
	Timestamp	
	hmac(VSRequest)	128
Value-Subtraction Request (VSRequest)	Price	Variable
	Merchant Data	
	h(OI)	128
Value-Subtraction Response (VSResponse)	h(OI)	128
	Response (Yes/No)	Variable
Value-Claim Request (VCRequest)	VSRquest	Variable
	Price	
Value-Claim Response (VCResponse)	h(OI)	128
	Response (Yes/No)	Variable



Order Description
PDA TX BLACK
Product ID
CA-150-X
Price
345.20
Type of Card
● Credit Card
○ Debit Card

Payment Results

Processes in PP	Time (ms
KSC-Mi generation	22
KSC-Mi+1 generation	33
KSC-Iz generation	13
VSRequest creation	14
PRequest creation	90

Status:
Transaction Approved

KSC-Mi+1 generation	33
KSC-Iz generation	13
VSRequest creation	14
PRequest creation	90

Status:
Transaction Approved
Total Time Payment Protocol:
Protocol Completed in 11947 ms

Fig. 12 Screenshots of the Payment phase on the Nokia™ N95 Device

6.1.1 Execution time of Merchant Registration Protocol

The empirical results obtained for our implementation of the CCMS-VAN protocol are reported in this section, where we focus on the time taken to perform various parts of the CCMS-VAN protocol and the overall time taken to complete a Payment transaction. The results were collected by performing 10 executions with different sets of data and the time measurements were done using the *getTime* method in the *Date* class of J2SE.

Table 6 Time taken (in milliseconds) at Client to execute the *Merchant Registration Protocol* (**TCRP**)

Processes	Device Nokia™ N95
AESGen_w	39.90
AESGen KS_{c-m}	28.80
AESEnc Data	81.90
AESDec Data	50.10
TCRP	1118.50

Table 7 Time taken (milliseconds) at Merchant to execute the *Merchant Registration Protocol* (**TMRP**)

Processes	Nokia™ N95
AESEnc Data	< 1
AESDec Data	6.40
TMRP	34.70

The average times required by the Client and the Merchant to perform the Merchant Registration Protocol are shown in Tables 6 and 7, respectively. Note that the time taken by the Client to input the data has not been included into the time taken by the application to perform the Merchant Registration Protocol. Moreover, when we tested our implementation by entering the maximum number of characters allowed into each individual field, we found that the maximum times taken by the Client (1196 milliseconds for Nokia™ N95) to perform the Merchant Registration Protocol are not much different from the average times calculated in Table 6 and Table 7.

6.1.2 Execution time of Payment Protocol

The average total time that the Client has spent on performing the first transaction with the Merchant (both the Merchant Registration Protocol and the Payment Protocol) was 9.46 seconds (1.12 of **TRCP** + 8.34 of **TPP** = 9.46 seconds) using the Nokia™ N95 device. For subsequent payment transactions, the total average time to complete each transaction will reduce to only 8.34 seconds on Nokia™ N95 device because the Client does not have to execute the Merchant Registration Protocol. This minimal amount of time to complete a transaction reveals the potential of CCMS-VAN protocol to execute Payment transactions in wireless environments with minimal delays.

Table 8 shows the average time in milliseconds needed to execute the Payment Protocol by the Client, the Merchant, and the Payment Gateway, respectively.

6.1.3 Application size

Due to the limited memory space available on mobile devices, application code size is an important issue which should be considered when developing applications for such devices. As the Client is the only party in our proposed Payment protocol that uses a mobile device to interact with the system, in this section we focus on the Clients side for the performance evaluation measurements.

Table 8 Time taken (in milliseconds) at Client (Nokia™ N95), Merchant, Payment Gateway, and Issuer on performing *Payment Protocol* (**TPP**)

Processes	Nokia™ N95	Merchant	PG	Issuer
KS_{C-M_i} generation	253.90	6.70	–	–
$KS_{C-M_{i+1}}$ generation	29.90	< 1	–	–
KS_{C-I_z} generation	15.00	–	–	< 1
KS_{M-PG_k} generation	–	–	6.80	–
VSRequest creation	145.30	–	–	–
PRequest creation	522.00	–	–	–
PRequest decryption	880.00	–	–	–
VCRequest creation	2.10	–	–	–
VCRequest decryption	–	–	< 1	–
VSResponse creation	–	–	–	4.80
TPP	8342.20	2742.10	322.50	12.80

The CCMS-VAN wallet software has an acceptable file size of 68 Kilobytes for the Nokia™ N95 mobile phone which is about 0.042% of the memory available[3] on the above mobile device.

7 Conclusions and further work

A lightweight protocol for secure on-line Payments in a vehicle-to-roadside Restricted Scenario in VANETs (where the Merchant cannot directly communicate with the Acquirer) was proposed in this research. Our protocol employs symmetric cryptographic techniques which has low computation requirements for all engaging parties (since no public-key operation is required). Moreover, the Client takes an active role in the Payment process and acts as a proxy to allow the communication between the Merchant and the Acquirer.

Our empirical performance evaluation results on the implementation have proven that Payment transactions over a wireless network can be conducted by our proposed CCMS-VAN protocol. The selected lightweight, secure cryptographic exploited by our proposed payment protocol. Deploying such algorithms in CCMS-VAN results in the reduction of messages exchanged and computation costs at the Client's mobile device. As we have seen from the implementation, a Payment transaction by CCMS-VAN can be completed within average of 8.34 second using a Nokia™ N95 device.

The Client acts as a proxy during the Payment process (which requires more computation resources and message exchanges) and despite the limited resources on the mobile device, our performance results demonstrate that we can still complete a Payment transaction within a reasonable amount of time (compared with other payment protocols reported in the literature [28, 44, 50] which have similar end-to-end latency but less message exchanges).

[3]The internal memory available in a Nokia™ N95 mobile phone is 163840 Kilobytes.

The Clients CCMS-VAN wallet software bytecode requires only 68 Kilobytes to be stored on a Nokia™ N95 mobile device. This small code size of the application makes it very suitable for memory-constrained mobile devices.

Acknowledgements We thank the anonymous reviewers for their constructive suggestions and remarks which helped to improve the quality and presentation of this paper.

References

1. Abad Peiro, J. L., Asokan, N., Steiner, M., & Waidner, M. (1997). Designing a generic payment service. *IBM Systems Journal*, *37*(1), 72–88.
2. Asokan, N. (1994). Anonymity in mobile computing environment. In *Workshop on mobile computing systems and applications* (pp. 200–2004).
3. Bakhtiari, S., Baraani, A., & Khayyambashi, M.-R. (2009). MobiCash: a new anonymous mobile payment system implemented by elliptic curve cryptography. In *WRI world congress on computer science and information engineering* (pp. 286–290).
4. Bellare, M., & Rogaway, P. (1993). Entity authentication and key distribution. In *Advances in cryptology (CRYPTO'93)* (pp. 232–249).
5. Bellare, M., Garay, J., Hauser, R., Herzberg, A., Krawczyk, H., Steiner, M., Tsudik, G., Van Herreweghen, Els., & Waidner, M. (2000). Design, implementation and deployment of the *i*KP secure electronic payment system. *IEEE Journal on Selected Areas in Communication*, *18*(4), 611–627.
6. Bellare, M. (2006). New proofs for NMAC and HMAC: security without collision-resistance. In *The 26th annual international cryptology conference (Crypto 2006)* (pp. 602–619).
7. The Legion of the Bouncy Castle (2008). The Legion of the Bouncy Castle Java cryptography APIs version 1.4. http://www.bouncycastle.org/.
8. Buccafurri, F., & Lax, G. (2011). Implementing disposable credit card numbers by mobile phones. *Electronic Commerce Research*, *11*(3), 271–296.
9. Car2Car Communication Consortium (2007). *Overview of the C2C-CC System* (Technical Report version 1.0).
10. Chari, S., Kermani, P., Smith, S., & Tassiulas, L. (2001). Security issues in M-commerce: a usage-based taxonomy. In *E-commerce agents* (pp. 264–282).
11. Ford, W. (1995). Advances in public-key certificate standards. *ACM SIGSAC Review*, *13*(3), 9–15.
12. Gao, J., Kulkarni, V., Ranavat, V., Chang, L., & Mei, H. (2009). A 2D barcode-based mobile payment system. In *Third international conference on multimedia and ubiquitous engineering (MUE 2009)* (pp. 320–329).
13. Hall, J. J., Kilbank, S., Barbeu, M., & Kranakis, E. (2001). WPP: a secure payment protocol for supporting credit- and debit-card transactions over wireless networks. In *International conference on telecommunications (ICT 2001)*.
14. Hassinen, M., Hyppönen, K., & Haatajam, K. (2006). An open, PKI-based mobile payment system. In *International conference on emerging trends in information and communication security (ETRICS'2006)* (pp. 86–100).
15. Housley, R., Ford, W., Polk, W., & Solo, D. (1999). Internet X.509 public key infrastructure certificate and CRL profile, IETF RFC2459.
16. Hu, Z., Liu, Y., Hu, X., & Li, J. (2004). Anonymous micropayments authentication (AMA) in mobile data network. In *23rd annual joint conference of the IEEE computer and communications societies (IEEE INFOCOM)* (pp. 46–53).
17. Hwang, R., Su, F., & Huang, L. (2007). Fast firmware implementation of RSA-like security protocol for mobile devices. *Wireless Personal Communications*, *42*(2), 213–223.
18. Krawczyk, H., Bellare, M., & Canetti, R. (1997). HMAC: Keyed-hashing for message authentication, RFC 2104.
19. Menezes, A. J., Van Oorschot, P. C., & Vanstone, S. A. (1997). *Handbook of applied cryptography*. Boca Raton: CRC Press.
20. NIST (1999). FIPS PUB 46-3 Data Encryption Standard (DES). http://csrc.nist.gov/publications/fips/fips46-3/fips46-3.pdf.
21. NIST (2001). FIPS PUB 197 Advance Encryption Standard (AES). http://csrc.nist.gov/publications/fips/fips197/fips-197.pdf.

22. Sun Microsystem (2008). Java platform, Micro Edition (Java ME), API specification. http://java.sun.com/javame/index.jsp.
23. Sun Microsystem (2008). Java platform, Micro Edition (Java SE) v 1.6.0, API specification. http://java.sun.com/javase/index.jsp.
24. Juntao, M. (2003). *Enterprise J2ME: developing mobile java applications*. New York: Prentice Hall.
25. Xi, K., Ahmad, T., Han, F., & Hu, J. (2010). A fingerprint based bio-cryptographic security protocol designed for client/server authentication in mobile computing environment. *Security and Communication Networks*, *4*(5), 487–499.
26. Kohl, J., & Neuman, B. C. (1993). The Kerberos network authentication service (Version 5), IETF RFC1510.
27. Kungpisdan, S., Srinivasan, B., & Dung Le, P. (2004). A secure account-based mobile payment protocol. In *International conference on information technology: coding and computing (ITCC'04)* (pp. 35–39).
28. Tiong, B., Kungpisdan, S., & Dung Le, P. (2004). KSL protocol: design and implementation. In *IEEE conference on cybernetics and intelligent systems* (pp. 544–549).
29. Lei, Y., Chen, D., & Jiang, Z. (2004). Generating digital signatures on mobile devices. In *18th international conference on advanced information networking and applications (AINA'04)* (pp. 532–535).
30. Misra, S., & Wickamasinghe, N. (2004). Security of a mobile transaction: a trust model. *Electronic Commerce Research*, *4*(4), 359–372.
31. Martinez-Pelaez, R., Rico-Novella, F. J., & Satizaba, C. (2010). Study of mobile payment protocols and its performance evaluation on mobile devices. *International Journal of Information Technology and Management*, *9*(3), 337–356.
32. Mishra, B., Nayak, P., Behera, S., & Jena, D. (2011). Security in vehicular ad hoc networks: a survey. In *Proceedings of the 2011 international conference on communication, computing & security (ICCCS 2011)* (pp. 590–595).
33. Neuman, B. C., & Tso, T. (1994). Kerberos: an authentication service for computer networks. *IEEE Communications*, *32*(9), 33–38.
34. Papadimitratos, P., Kung, A., Hubaux, J.-P., & Kargl, F. (2006). Privacy and identity management for vehicular communication systems: a position paper. In *Workshop on standards for privacy in user-centric identity management*.
35. Ratha, N. K., Connell, J. H., & Bolle, R. M. (2001). Enhancing security and privacy in biometrics-based authentication systems. *IBM Systems Journal*, *40*(3), 614–634.
36. Raya, M., & Hubaux, J.-P. (2005). The security of vehicular ad hoc networks. In *3rd ACM workshop on security of ad hoc and sensor networks (SASN'05)* (pp. 11–21).
37. Sanchez-Avila, C., & Sanchez-Reillol, R. (2001). The Rijndael block cipher (AES proposal): a comparison with DES. In *35th IEEE international Carnahan conference on security technology* (pp. 229–234).
38. Samara, G., Al-Salihy, W., & Sures, R. (2010). Security analysis of vehicular ad hoc networks (VANET). In *Second international conference on network applications, protocols and services* (pp. 55–60).
39. Shin, K., Choi, H., & Jeong, J. (2009). A practical security framework for a VANET-based entertainment service. In *Proceedings of the 4th ACM workshop on performance monitoring and measurement of heterogeneous wireless and wired networks (PM2HW2N 2009)* (pp. 175–182).
40. Shuai, F., You, J., & Li, Z. (2010). Research on symmetric key-based mobile payment protocol security. In *IEEE international conference on information theory and information security (ICITIS 2010)* (pp. 340–344).
41. Téllez, J., Sierra, J., Izquierdo, A., & Torres, J. (2006). Anonymous payment in a kiosk centric model using digital signature scheme with message recovery and low computational power devices. *Journal of Theoretical and Applied Electronic Commerce Research*, *1*(2), 1–11.
42. Téllez, J., & Sierra, J. (2007). A secure payment protocol for restricted connectivity scenarios in m-commerce, EC-Web (pp. 1–10).
43. Téllez, J., Sierra, J., Zeadally, S., & Torres, J. (2008). A secure vehicle-to-roadside communication payment protocol in vehicular ad hoc networks. *Computer Communications*, *31*(10), 2478–2484.
44. Téllez, J., Zeadally, S., & Sierra, J. (2010). Implementation and performance evaluation of a payment protocol for vehicular ad hoc networks. *Electronic Commerce Research*, *10*(2), 209–233.
45. Téllez, J., Zeadally, S., & Sierra, J. (2010). Security attacks and solutions for vehicular ad hoc networks. *IET Communications*, *4*(7), 894–903.

46. Tufail, A., Fraser, M., Hammad, A., Kim Ki, H., & Seung-Wha, Y. (2008). An empirical study to analyze the feasibility of WIFI for VANETs. In *12th international conference on computer supported cooperative work in design (CSCWD 2008)* (pp. 553–558).
47. Vincent, O., Folorunso, O., & Akinde, A. (2010). Improving e-payment security using elliptic curve cryptosystem. *Electronic Commerce Research*, *10*(1), 27–41.
48. Wang, H., & Kranakis, E. (2003). Secure wireless payment protocol. In *International conference on wireless networks* (pp. 576–582).
49. Wang, X., & Cui, N. (2009). Research of security mobile payment protocol in communication restrictions scenarios. In *2009 international conference on computational intelligence and security* (pp. 213–217).
50. Wu, X., Dandash, O., Dung Le, P., & Srinivasan, B. (2006). The design and implementation of a wireless payment system. In *First international conference on communication system software and middleware (Comsware 2006)* (pp. 1–5).
51. Yousefi, S., Mousavi, M., & Fathy, M. (2006). Vehicular ad hoc networks (VANETs): challenges and perspectives. In *6th international conference on ITS telecommunications* (pp. 761–766).

Jesús Téllez Isaac is a System Engineer graduated at the Central Technological University (UNITEC), Venezuela, and he is a doctor candidate at the University Carlos III of Madrid at the Computer Science Department. He is an associate Professor at the Computer Science Department of the University of Carabobo. Also, he has been serving as a Technical Program Committee member in some international conferences. His research interests include Internet Security, performance evaluation of systems, mobile computing, mobile payment systems.

Sherali Zeadally received his B.Sc. in Computer Science from University of Cambridge, England, and the Ph.D. in Computer Science from University of Buckingham, England in 1996. He is an Associate Professor at the University of the District of Columbia. He currently serves on the Editorial Boards of over 15 international journals. He has been serving as a Co-Guest editor for over a dozen special issues of various peer-reviewed scholarly journals. He is a Fellow of the British Computer Society and a Fellow of the Institution of Engineering Technology, UK. His research interests include computer networks including wired and wireless networks, network and system security, mobile computing, ubiquitous computing, RFID, performance evaluation of systems and networks.

José Sierra Cámara is an Associate Professor at the Computer Science Department of the University Carlos III of Madrid. He is Ph.D. in Computer Science and M.Sc. in Business Administration. His research work is centered in the area of the Internet Security and, in this area, at the present time he is working and researching. He has participated in numerous research projects, and had published articles in journals related with the Security in the Technologies of the information.