Ruiz-Martínez and Marín-López *EURASIP Journal on Wireless Communications and Networking* 2012, **2012**:161
http:?/jwcn.eurasipjournals.com/content/2012/1/161

EURASIP Journal on Wireless Communications and Networking
a SpringerOpen Journal

RESEARCH Open Access

A lightweight payment scheme for real-time services based on SIP

Antonio Ruiz-Martínez* and C Inmaculada Marín-López

Abstract

In the session initiation protocol (SIP), payments have been proposed as a way for vendors to obtain profit from the services they provide. Payments in SIP have also been proposed for microbilling and even as a solution to SPAM in VoIP systems. Although several proposals exist for making payments in SIP, they present some limitations when we want to pay for access to real-time services: either they are not suitable for micropayments or they do not consider security in the payment information exchanged. As a response to these limitations, we propose a new SIP payment protocol, *LP-SIP*, that supports the payment according to different models like pay-per-time, session-based, etc. It also performs payments in SIP efficiently and takes into account the secure exchange of payment information, unlike other existing proposals. Thus, we provide a lightweight payment protocol that can be used for the payment of real-time services.

1 Introduction

Many multimedia services rely on session initiation protocol (SIP) [1], which is a cornerstone of the next generation networking and IP multimedia subsystem. Some of the most important real-time multimedia services based on SIP are telephone calls, video conferencing, Internet television, and instant messaging. Additionally, some other new, innovative services such as using the voice to compose e-mails, event notification, paying for parking spaces, etc., are becoming steadily available under this protocol.

As is usual on the Internet, multimedia service providers want to obtain profits from the services they provide by SIP. There are two kind of solutions for charging these services: those based on authentication, authorization and accounting (AAA) infrastructures such as those based on RADIUS [2], Diameter [3] or SIPA/SIPA+ [4] and those based on SIP-based payment mechanisms such as Fischl and Tschofenig's proposal [5], SIMPA [6], SIP-Coin [7], Fan's et al. protocol [8], or Zhang's et al. protocol [9]. In AAA infrastructures the charging process is focused on the interaction between the vendor and payment provider (a client-server architecture) [5,10]. On the hand, in SIP-based payment mechanisms the client is involved directly in the transaction and exchanges payment information with the vendor (a peer-to-peer architecture).

As stated in [5,10], the integration of payments in SIP can suppose several advantages over AAA-based mechanisms. First, the use of simpler pricing models that avoid the complexity of AAA protocols [5,10], which are less efficient for real-time and high available services [10]. Second, the support of different payment models. Third, scalability because there is no need to establish a direct trust-relationship between the client and the vendor. Furthermore, as there is no direct relationship, the risk can be an acceptable to the client, since the amount of money at risk is limited [5]. Finally, it overcomes the high costs of traditional payment mechanisms that may be suitable for a single payment during a transaction but not when low value payments have to be made or when the payment has to be made with a negligible delay.

The integration of payment information in SIP has two additional advantages. First, the client can ascertain the price of accessing the service during the session initiation, which facilitates price discrimination and avoids the client having to use other means to discover this information. This also gives vendors the support of different business models, which makes some vendors more attractive than others [11]. Second, the client does not need additional flows with the vendor or a payment system. The separation of the flows of payment process and session access

* Correspondence: arm@um.es
Department of Information and Communications Engineering, Faculty of Computer Science, Regional Campus of International Excellence "Campus Mare Nostrum", University of Murcia, 30100 Murcia, Spain

supposes that the payment process has to generate some payment information (token or receipt of payment) for subsequent inclusion in the SIP session to confirm the payment and for the SIP session to allow access [5]. With the inclusion of all payment information in SIP the payment is made at the same time as the access and no additional flows are needed, which can also improve efficiency.

Some interesting solutions have been proposed with the aim of supporting payments for multimedia services based on SIP [5-9,12]. The main drawbacks of these proposals are that they are either not suitable for micropayments or they do not consider the secure and efficient exchange of payment information.

Another approach that we could have followed was the integration of some of the existing protocols. However, this approach has two drawbacks: these micropayment protocols do not consider the confidentiality of the payment information exchanged and they use asymmetric cryptography in the payment phase [13-16], which is not suitable for real-time transactions due to the computational costs of these operations.

A payment protocol that supports making the payment of real-time services based on SIP should satisfy two main requirements. First, the protocol should be a micropayment protocol since these kind of protocols allow payments of (very) low value amounts [13,17,18] and are characterized because they try to reduce the number of asymmetric cryptographic operations, the number of exchanges and the participation of third parties [13,17-20]. Thus, the protocol can be used to make payment of real-time services. Second, the incorporation of payment in SIP should be based on the extensibility mechanisms defined by this protocol [1,12,21] in order to avoid high costs of development and SIP overheads, that is, it should be based only on the introduction of new headers, contents for the body of the messages and tags [21].

In this article, we propose a new SIP extension for making micropayments that satisfies all the requirements mentioned. We have also taken into account other interesting features. First, apart from supporting the payment for the whole session, we also support additional payments while the session is in progress. Thus, it is possible to make an initial payment for an initial period and then additional payments so as to continue with the session without any interruption. For this purpose, the payment protocol should be efficient. Finally, our proposal guarantees that the exchange of payment information is made in a secure way and offers protection against forgery or double-spending.

The article is organized as follows. Section 2 examines background related to SIP and payment scenarios. These scenarios are useful to derive the requirements (defined in Section 3) to cover in SIP payment protocol. In Sections 4 and 5, we describe in detail our payment protocol for SIP, named **LP-SIP**, which is analyzed in depth in Section 6. Section 7 analyzes related study. Section 8 provides an in-depth comparison between SIP-Coin and our proposal. Finally, Section 9 concludes the article and introduces future study.

2 Background

This section describes some proposals in which LP-SIP is based, namely, SIP [1], the model offer/answer [22] and the provisional response acknowledgement (PRACK) [23] method. Furthermore, in order to establish the requirements our proposal should satisfy, we describe some possible scenarios for payments in SIP.

2.1 SIP

The SIP [1] is a protocol designed to help in the creation and management of multimedia sessions such as a voice over IP call, a distributed conference, an instant message conversation or, in general, every type of data communication between peers, also called user agents.

2.1.1 Session set up

SIP uses a three-way protocol to carry out the session set up, unlike other SIP functions that are based only on the request-response pattern. The third message is used as a reliability mechanism.

The three-way protocol works as depicted in Figure 1. It starts when a user agent sends an *INVITE* request, which contains a description of the type of session the user wants to take part in, the media and ports to be used and the codecs supported. This information is described by means of the session description protocol (SDP) [24]. As a response to the *INVITE* request, the callee answers with a provisional response (*100 Trying*). Next, the second provisional response (*180 Ringing*) is used so that the caller can know that the user agent contacted is running and trying to notify the callee. When the user answers the call, the *200 (OK)* response code is sent to inform the acceptance of the call including codecs and parts. When this response is received, the caller ends the three-way protocol by an *ACK* message. Finally, media transmission starts using the codecs exchanged during the three-way protocol. Usually, this transmission is made by using UDP. However, TCP or TLS could be used when a higher level of security is required.

2.1.2 Other functions

In SIP a session remains established until one of the peers starts its termination by the *BYE* message. This request can be sent at any moment during the session.

The interchange of messages is possible between the peers throughout the session thanks to the instant messaging facilities that SIP includes by *MESSAGE* and *200 (OK)* messages.

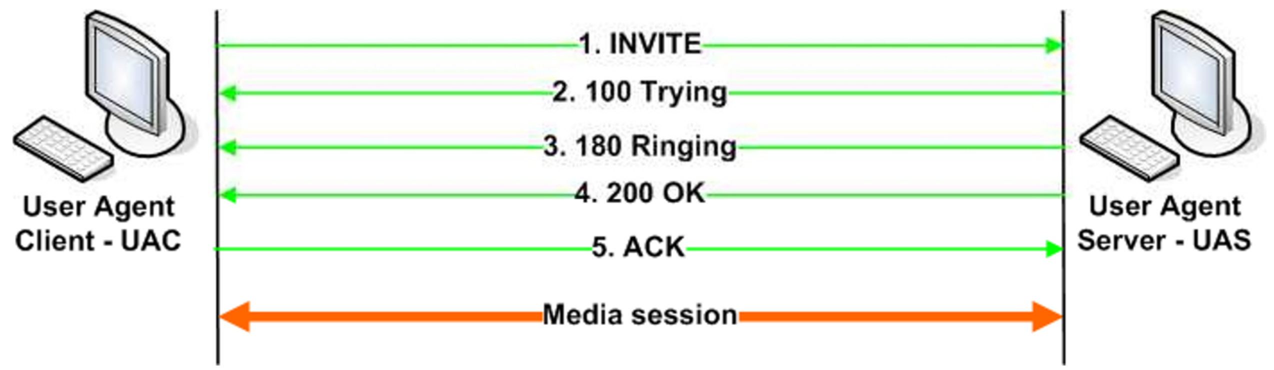

Figure 1 SIP session set up. This figure shows the messages exchanged during the session set up process in SIP.

2.1.3 Extensibility

SIP offers different extensibility mechanisms - it is possible to define new headers, new address schemes to support communication with new protocols and new option tags to establish new parameters for headers.

The *Require* header defined in SIP can be used for User Agent Clients (UACs) to indicate user agent servers the options expected.

2.2 Session negotiation based on the offer/answer Model

In this section, we introduce the different mechanisms we can use to achieve an agreement on the attributes of a session. First, we explain the basic offer/answer model and, next, the *PRACK* method.

2.2.1 Offer/answer model

The offer/answer model [22] defines a mechanism that allows two entities to agree on the attributes to use in a multimedia session. This process uses SDP to reach the agreement. According to this specification, a higher layer protocol like SIP is needed for the offer/answer exchange. In SIP the exchange of SDP descriptions is carried out during session establishment.

2.2.2 PRACK

The PRACK method [23] is an extension of SIP that allows peers to refine the offer/answer model by offering additional possibilities for these exchanges.

This extension includes three messages: a *183 (Session progress)* provisional response, a *PRACK* and its corresponding *200 (OK)* response to the PRACK. The first message is issued by the server side to indicate that a *PRACK-200 (OK)* cycle is about to start. Then, the *PRACK* and *200 (OK)* messages let clients and servers exchange offers and answers about the kind of media to be used in the session. When the negotiation is finished, the server issues the *200 (OK)* response code to the *INVITE* message.

2.3 Use case scenarios

This section presents different types of scenarios that our proposal tries to cover. These scenarios have been used and mentioned in the literature as scenarios where the use of payment in SIP is suitable. In fact, our use case scenarios are a more detailed description of the scenarios mentioned in [5-7,9,12,25]. All of them use SIP to support the payment of real-time multimedia services and try to show the different possibilities that should be supported. It is important to point out that other different scenarios could have chosen. Our goal with these use case scenarios is twofold. On the one hand, to facilitate the understanding of the proposal and show when some of the features provided by the protocol could be used. Although the scenarios presented are different, the features related to payment are similar and, therefore, the scenarios could represent a wide range of SIP-based real-time multimedia services that require a payment.

2.3.1 Scenario 1. Preventing SPAM in VoIP Systems

In the same way that spam has proliferated in e-mail systems, spam over the Internet telephony and instant messaging is expected to become even more important.

The solution proposed in [5,12] to this problem is that the first time a user receives an unsolicited call, the recipient, before accepting it, requests a payment whose amount is (very) small (micropayments), e.g., four cents. When the payment is made, the call is answered and the caller is included in a whitelist for future communications. Furthermore, the payment made by the caller could be refunded, with the caller and the recipient exchanging their roles. In the refund the recipient makes the payment and the caller receives it.

2.3.2 Scenario 2. Real-time communications services in P2P networks

Let us suppose that a company wants to offer language-training services over a P2P network created for the provision of multimedia services. These services could be provided by means of video, audio and/or instant messages [5]. A user can access these services by making a payment. The services could be charged following a pay-per-time or flat free model. There are also other models that could be followed, such as pay-per-volume, session-based or pay-as-you-watch [26].

If we suppose a pay-per-time model, the user would make an initial payment to start the session and one additional payment for each period of time (usually, this period of time is short, between 10 s and 1 min) the user is using the service. Thus, the user would pay only for the time she uses the service. We could suppose the initial payment could be one Dollar/Euro and each additional minute five cents. The user would be notified some time before the time/data finishes and the payment should be made quickly to avoid session interruption. Thus, if the user spends 40 min in a session training, 40 payments are made.

2.3.3 Scenario 3. VoIP (Voice messages and calls)

Let us suppose that a VoIP company provides a service in which national calls are paid by a flat rate and other additional services such as international calls and listening to voice messages by making a payment.

In the access to voice messages, the company charges an initial amount for the establishment of the call, e.g., 29 cents (Euros/Dollars) and a payment for each voice message, e.g., 13 cents. Thus, if a user were to listen to, e.g., three voice messages, she would have to make four payments.

The establishment of international calls could work in a similar way. The service provider could request an initial amount for the establishment, e.g., 30 cents (Euros/Dollars) and, after the establishment and before the time/data paid is finished, 25 cents for each minute or fraction. Thus, if a user established a call that lasted 4 min and 30 s, she would have to make six payments. In this scenario, it is also important that payments are made very quickly to avoid the interruption of the session, once started.

3 Requirements for supporting payments in SIP

From the aforementioned use case scenarios and the requirements established in previous proposals that make payments in SIP [5-7,9,12,25], we can derive the different requirements that should satisfy a lightweight payment protocol for SIP. These requirements are (if a requirement maps directly with a use case scenario we identify it by the number of scenario between parenthesis. Otherwise, the requirement is a generic one derived from previous proposals):

- *Payment protocol based on extensibility mechanisms*. The support of a payment protocol in SIP should be based on the extensibility mechanisms of SIP in order to facilitate the incorporation of payments to the current developments of SIP. This also facilitates the acceptance of the solution.
- *Different payment models*. The vendor should be able to support different payment models: time (scenarios 2 and 3), volume (scenarios 2 and 3) and session-based charging (scenario 1) [5]. It could even support the pay-as-you-watch model [26] in which the session is initiated without any payment. Later, once some content has been sent, a payment is requested. The models supported should be indicated together with payment information. Furthermore, we should provide the messages needed to request the payment when necessary (scenarios 2 and 3).
- *Additional payments*. The vendor should be able to receive incremental or additional payments [7]. That is, the client could make an initial payment for a part of the session instead of paying for the whole session. Later, before the session finishes, if the user is interested in it, she could make an additional payment. It is important to point out that this additional payment should be received without stopping the session in progress (scenarios 2 and 3).
- *Time/Data notification*. The merchant should be able to notify the customer that the time/data she paid for is finishing. Thus, session interruption can be avoided (scenarios 2 and 3) [7].
- *Lightweight and efficient protocol*. The payment protocol should allow the payment to be efficient by using lightweight cryptography (mainly based on symmetric cryptography instead of using asymmetric cryptography) and avoiding additional connections to make the payment, as well as the minimization of the participation of trusted third party (TTP) entities [5,7]. Thus, the initiation of the session and the payment would be performed quickly (all the three scenarios) and the protocol would also make it more difficult for an interruption to occur when an additional payment is requested (scenarios 2 and 3).
- *Security in the payment process*. The payment information should be exchanged in such a way that the confidentiality and integrity is guaranteed against possible attackers. Furthermore, the protocol should guarantee that any entity (clients, vendors, or external attackers) cannot perform double spending without being detected, nor generate bogus payments, etc.

4 The LP-SIP protocol foundations

This section describes the foundations of the Lightweight Payment SIP protocol that we propose, hereinafter, *LP-SIP* protocol. First, we describe the participating entities. Then, we outline how the system works, the different processes involved in LP-SIP and the extensions we have made to SIP to support it.

4.1 Roles

Three kinds of entities participate in the LP-SIP protocol: clients, vendors, and brokers.

Clients are entities that want to access real-time services and are willing to pay for them. Vendors (we could also name them service providers) are the entities that provide those real-time services based on SIP and charge clients that want to access them. These services can be offered by following different payment models such as pay-per-time, pay-per-data, etc. Payments are made by using e-coins that are minted by a broker.

A broker acts as a TTP and plays the role of financial service provider between customers and vendors. This role could be played by different kind of entities such as a bank, a payment service provider, network service provider, a (mobile) network operator, an Internet Service Provider, a multimedia application provider, etc. This entity is responsible for registering clients and vendors, minting e-coins for making payments, charging the user for the e-money they withdraw, paying vendors from the e-coins they deposit, and key management that allows secure communications between clients and vendors, so guaranteeing confidentiality, integrity and authentication.

For the client and the vendor to perform a payment transaction both entities have to be registered at the same broker since each broker can only verify its own e-coins. But this does not mean that they have only to work with one broker since both client and vendor could be registered at different brokers. Thus, they can purchase/sell real-time services to a broad number of vendors/clients, respectively. We could compare it with a client owning several credit cards from different companies and a vendor supporting the payment with different credit card brands.

4.2 Overview

LP-SIP is based on the use of e-coins that are obtained prior to accessing the SIP-based real-time service. These e-coins are charged to the client's account when she withdraws them. That is, our protocol is based on token and on prepayment. The client obtains e-coins from a broker and they can be used with any vendor registered with that broker. Thus, any entity has to carry out a registration process with the broker to take part in the system (see step 0 in Figure 2, hereinafter all the steps are referred to this figure).

In the client's registration process (step 0), the client establishes an account that the broker uses to charge her for the different e-coins that she will obtain (in the withdrawal process) from the broker to make the payments. In this process she also obtains two symmetric cryptographic keys (for encryption and authentication), which are used to communicate with the broker in a secure, efficient way.

In the vendor's registration process, the vendor establishes some cryptographic keys (master keys) with the broker. The broker is responsible for increasing the balance of the vendor's account from the e-coins the vendor deposits.

Once the client has completed the registration process, she has to obtain one or more e-coins from the broker in order to be able to make purchases from the vendors (step 1). Each e-coin that the client obtains could be used with several vendors as long as the value of the purchases does not exceed the value of the e-coin. This is because during a purchase the portion of the e-coin to be paid can be specified. The control of the amount spent is performed between the broker and the vendor.

In the withdrawal request, the client can also request keys to communicate with the vendors she wants to carry out transactions with. In this process, the broker charges the amount of e-coins only at this moment or the charge could be made, at the end of some period established by the client and the broker (at the end of the day, the week, etc). At any subsequent moment, the client may need to request keys for communicating with new vendors, which is done by the client launching a vendor's key request (see step 2).

The payment process starts at the same time the SIP session is initiated (step 2). Thus, the vendor requests a payment when the client initiates a SIP session to access a payment-based service. The payment requested could be for the whole session or for only some amount of time or data. Then, the client sends an e-coin to the vendor with the amount of the e-coin she wants to pay. For example, the e-coin value could be 1 Euro (€)/Dollar ($) and the client could want to pay only 0.2 Euros/Dollars.

All the information exchanged in the payment phase between the client and the vendor is protected by using authentication and ciphering so that the information related to the purchase cannot be modified without being detected, and is confidential between them.

It is important to point out that a specific e-coin issued by the broker can be used with several vendors in different transactions. Thus, we avoid two problems of e-coins that are issued for a specific vendor [27]. First, refunding processes when an e-coin is not completely spent and we are not going to perform any new transactions with a vendor. Second, handling many different e-coins, if we work with many vendors this, additionally, could suppose tidying up an important of e-money that cannot be used with other vendors. Thanks to the possibility of defining the portion of the e-coin to be used, we also avoid the problem of the e-coins that can be used with any vendor that is the establishment of a connection with the broker for each payment [27].

When the vendor (Vendor$_X$) receives an e-coin (let us name it e$_1$), he checks whether he knows the amount

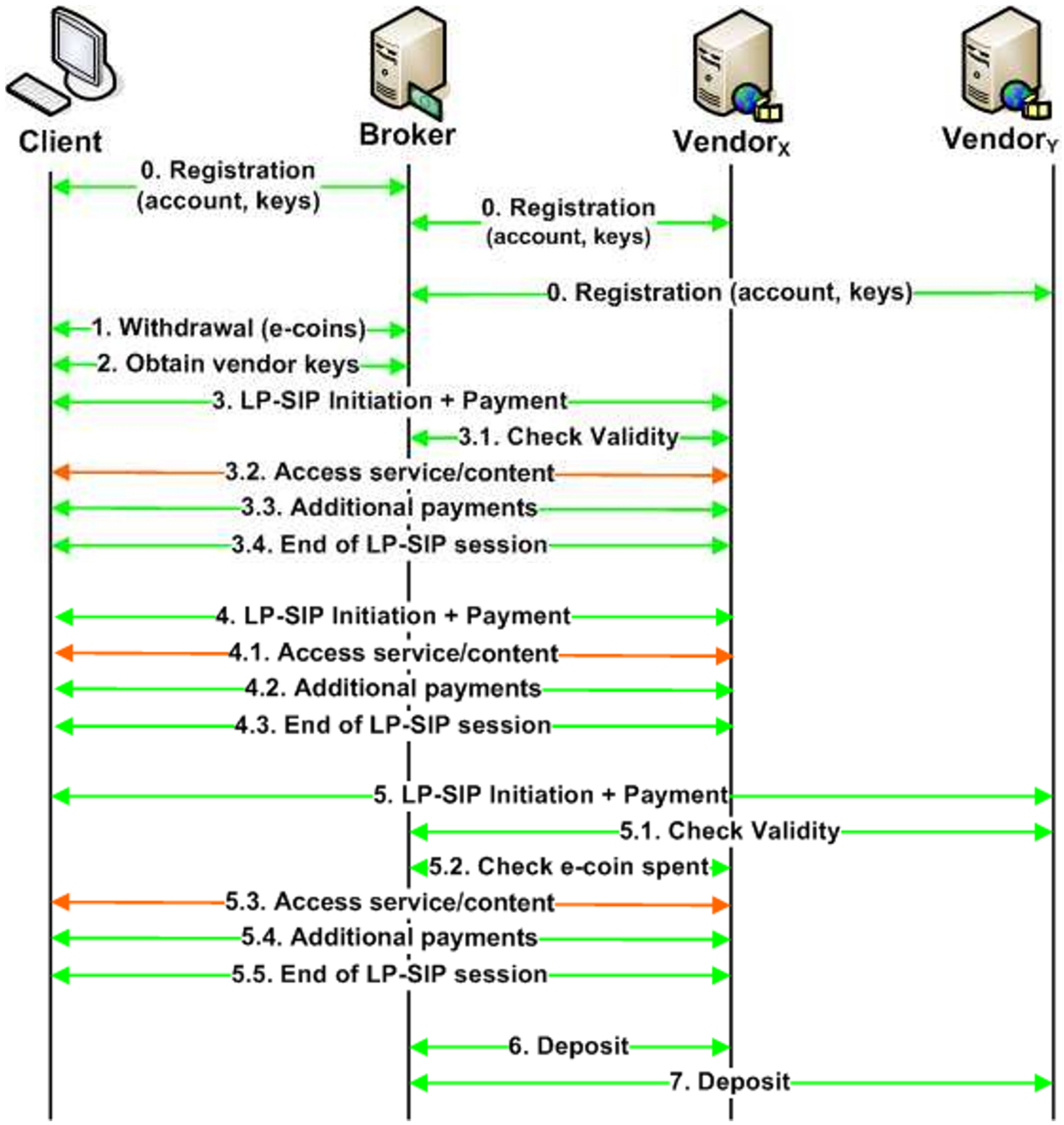
Client
Broker
Vendor$_X$
Vendor$_Y$
0. Registration (account, keys)
0. Registration (account, keys)
0. Registration (account, keys)
1. Withdrawal (e-coins)
2. Obtain vendor keys
3. LP-SIP Initiation + Payment
3.1. Check Validity
3.2. Access service/content
3.3. Additional payments
3.4. End of LP-SIP session
4. LP-SIP Initiation + Payment
4.1. Access service/content
4.2. Additional payments
4.3. End of LP-SIP session
5. LP-SIP Initiation + Payment
5.1. Check Validity
5.2. Check e-coin spent
5.3. Access service/content
5.4. Additional payments
5.5. End of LP-SIP session
6. Deposit
7. Deposit

Figure 2 Overview of the LP-SIP processes. This figure depicts an overview of the messages that are exchanged in the different processes that LP-SIP defines.

spent with that e-coin and if he does not, he queries the broker (step 3.1). With this information he can know whether the amount paid is valid or not. If the payment is valid then the vendor provides the client the access to the service (step 3.2).

If the payment is for the whole session, the vendor provides the access to the service until the session finishes (when it has reached its end or when the client decides to finish it). Then, step 3.4 is executed.

In contrast, if the access is only for a limited amount of time or data, when the limit is about to be reached the vendor either requests a new payment to continue with the session or the session is finished. If the client supports receiving payment requests in an established session a payment request with information about the data/time remaining is sent. Thus, after step 3.2, step 3.3 is executed. This optional feature is called *additional payments*. For this payment it is not required to make any check with the broker (see step 3). Otherwise, when the payment is not made, the session finishes (step 3.4).

Subsequent payments of new services can be made with e-coin e_1 to this same vendor during a period of time pre-established by the broker (one day, one week, etc, usually, in micropayments, the period is established at one day [17,18], and this is the period that we consider) with no requirement to check the amount with the broker because the vendor can control it (see the access to a payment-based service in step 4). Thus, we avoid the vendor having to make many queries to the broker, which reduces overload for the system (both for the vendor and the broker).

On the other hand, if the e-coin e_1 is used with a new vendor ($V\,endor_Y$) (step 5), this new vendor will check the e-coin with the broker. The broker, as he knows that is being used by another vendor (Vendor_X), will request that he indicates the last amount that the vendor received as payment. Then, the broker informs the new vendor (Vendor_Y) and the rest of the process is performed as we have just explained for the payment with a vendor for the first time.

In this description, we suppose that the payments are made sequentially in time. Thus, the same e-coin (e_1) can be used. However, if we had to manage *n* different concurrent payments, *n* different e-coins would have to be used. In spite of this, the payment process would be the same as described. The only difference is that the e-coins used are different for each transaction (*n* transactions, *n* coins). For this reason we allow the user to request several e-coins (in the withdrawal process). Subsequently, the e-coins used in these concurrent payment transactions could also be used with the same or different vendors in the same way we have described above.

At the end of the period established by the broker, each vendor sends the broker the amounts of the different e-coins he has received so that the broker can pay the amounts into the account of each vendor.

4.3 SIP extensions to support LP-SIP protocol

This section describes the extensions we have defined in SIP to support the LP-SIP protocol. It is important to point out that our extensions are based on the extensibility mechanisms defined in SIP. Namely, our proposal is based on three kinds of elements. Firstly, on the definition of new options-tags for different headers of SIP messages: *Required*, *Supported*, and *Accept*. Second, the definition of a new header named *LP-SIP* for conveying information related to a payment transaction. Finally, we have defined a new type of content for the body of the SIP messages. This new content conveys the messages that we have defined for the different exchanges of LP-SIP. Next, we describe each of these elements in more detail.

4.3.1 New option-tags

Headers are used in SIP to convey different kinds of information during the session. In general, they specify the supported or required features by means of a list of option-tags. For this purpose, SIP uses the *Required* header in order to indicate the required features; the *Supported* header to specify some additional features that are supported; and, finally, the *Accept* header, which indicates the kind of information that is supported in the exchanges. For these headers we have defined the following option-tags:

- *lpayment*. This indicates that an entity supports LP-SIP. It has been defined for the *Supported* header and its use is mandatory when a payment is required. In this situation, if a client sends an *INVITE* and it does not contain it, the vendor answers with a *402 (payment required)* response code.
- *additionalpayment*. Its use is optional and is included in the *Supported* header so that the client can indicate she supports the reception of payment requests once the session is established. Hence, if the vendor also supports this feature, the client will receive a notification indicating that the amount of time or data she paid is about to finish and that a new payment is due.
- *application/lp-sip*. In the *Accept* header it indicates that an entity supports the exchange of the LP-SIP messages in the body of SIP messages. This option-tag also defines a new content-type that indicates that the content of a SIP message is a payment message according to our protocol.

4.3.2 LP-SIP header

We have also defined a new header for SIP messages named *LP-SIP*. This new header is used in the different messages to convey some basic information related to different processes that are associated to our proposal, such as entity registration, withdrawal of e-coins, payments, etc., (see overview in Section 4.2).

The definition of the *LP-SIP* header is according to the Backus-Naur Form (BNF) notation [28] (see below). The starting symbol is represented by *LP-SIP-HEADER* variable. In a SIP message our header is represented with the name *LP-SIP*. Basically, the header contains the kind of exchange that is being performed. As presented above, there are ten kinds of exchanges: *Vendor Registration*, *Client Registration*, *Vendor Key*, *Withdrawal*, *Payment Request*, *Payment*, *Check Validity*, *Ecoin Notification*, *Additional Payment*, and *Deposit*. De-pending on the exchange, the header could also include some additional information, e.g., in the payment request exchange (*PaymentRequest*), we can include the different prices of the service according to different payment models (data, time, etc).

```
 <LP-SIP-HEADER>   ::=   "LP-SIP:   "
<operation>
 <operation> ::= "VendorRegistration"

   | "ClientRegistration"
   | "VendorKey"
   | "Withdrawal"
   | "KeyRequest"
   | "PaymentRequest;" "PID=" <identifier>
   *(<modelInfo>) "Expiry=" <date>
   | "Payment;" "DateTime=' " <dateTime> "
   ' " [ <encryptedkey> ]
   | "CheckValidity"
   | "EcoinNotification"
   |     "AdditionalPayment;     "PID="
   <identifier>
        "TimeDataToStop=" <quantity> 1*
        (<AdditionalModelInfo>)
   | "Deposit"

 <encryptedKey>   ::=   \EncryptedKey="
<encryptedKeyValue>
 <modelInfo>  ::=  <modelId>  <brokers>
<model> "unit=" <quantity> <initialPay-
ment>

      [ <additionalPayment> ]

 <AdditionalmodelInfo>  ::=  <modelId>
<model> "unit=" <quantity> <amounts>
 <modelId> ::= "modID=" <identifier>
 <model> ::= "model=" <modelType>
 <brokers> ::= "brokers='" <identifier> *
("," <identifier>) " ' "
 <modelType> ::= "pay-per-time" | "pay-
per-data" | "pay-as-you-watch" | "session"
 <quantity> ::= <numeric-value><unit-
tag>
 <unit-tag> ::= "seg" | "min" | "h" | "Mb" |
"Kb" | "b" | "session"
 <initialPayment> ::= "initialPayment;"
<amounts>
 <amounts> ::= "amount='" <amount> *(","
<amount>) " ' "
 <additionalPayment> ::= "additionalPay-
ment;" 1*(<modelId> <model>

        "unit=" <quantity> <amounts>)
```

For the sake of simplicity we have not specified some elements in the BNF notation such as *identifier*, which is an alphanumerical string; *numeric-value*, which is a real number; *date-time*, which is a string representing a date and the time according to the extended form of International Organization for Standardization (ISO) 8601 [29]; and, *amount*, which is coded according to ISO 4217 [30], which allows us to express an amount where the quantity and the currency are coded as an alphanumerical string.

A simple example of the *LP-SIP* header that would be included in a SIP message for the payment exchange would be the following:

```
LP-SIP: Payment; DateTime= '2010-04-
05T14:30'
```

This header indicates with *Payment* operation that the payment is taking place and with the value of *DateTime* it indicates the time when the payment is being carried out.

4.3.3 Application/lp-sip content

The *application/lp-sip* content-type has been defined to include a new type of content in the body of the SIP messages. This content is a message associated to some of the LP-SIP processes. These messages are defined in Abstract Syntax Notation One (ASN.1) and we make use of some existing structures, defined as X.509 certificates or the PKCS#7/CMS format, to convey cryptographic information. The messages should be encoded according to distinguished encoding rules and then converted to its Base64 format in order to be conveyed in the body of the SIP message.

Since we have defined that the messages are included as a content in the body, we can combine the sending of payment information with other kinds of data such as text messages, images, etc.

The messages and responses codes of SIP that we use to convey some of the messages defined in our system are: *INVITE*, *183 Session in progress*, *PRACK*, *200 OK* and *MESSAGE*.

5 The LP-SIP protocol specification

The notation that we have decided to use in order to specify the sequence of messages in our protocol specification is shown in Table 1. This notation is based on the notation that appears in [31].

5.1 Vendor registration process

The vendor makes a registration process with the broker in order to register the account where he wants to receive the amounts of the e-coins he receives from clients. In this registration process the vendor also indicates to the broker a pair of master keys ($K_{EnV\ BMasterKey}$, $K_{SigV\ BMasterKey}$) that will be used to provide

Table 1 Cryptographic notation

Notation	Meaning
Data	This indicates that this piece of *data* is optional, and may not be in the message.
H(*Data*)	A message digest of *Data*, obtained using a secure hash algorithm such as SHA2 [35].
E_K (*Data*)	*Data*, encrypted by a symmetric cipher such as Advanced Encryption Standard (AES) [54] using the symmetric key *K*.
H_K (*Data*)	*Data* is authenticated using a HMAC algorithm using a secure hash function such as SHA-2 with a cryptographic key *K*.
$E_{K,K'}$ (*Data*)	E_K (Data, $H_{K'}$ (Data)). *Data* that is authenticated using a HMAC function with key *K'* and then encrypted using a symmetric cipher using the key *K*.
S_X (*Data*)	*Data* is signed using the private key of entity *X*. As a signature algorithm we could use RSA, DSA, ECDSA, etc.
E_X (*Data*)	*Data*, encrypted for entity *X* using public key cryptography (RSA) or elliptic curve cryptography–ECC–(ECDH, ECMQH, . . .). For computational efficiency, this is implemented using either a digital envelope (in RSA) or the ephemeral-static Elliptic Curve Diffie-Hellman (ECDH) key agreement algorithm (in ECC) as specified in [55,56].
$X \rightarrow Y$	This indicates that *X* sends a message to *Y*.

the clients with keys for establishing secure communication with him.

The different messages exchanged in this process are:

1. $V \rightarrow B$: MESSAGE (VendorRegistration-Request)

This message denotes the vendor sends the *MESSAGE* request defined in SIP. In brackets we indicate the kind of message that we include in the body of the message by means of the *application/lp-sip* content-type. Furthermore, in the *LP-SIP* header we indicate the kind of process that is being performed. We follow this convention to represent the rest of the messages that will appear throughout the article.

In this exchange, in the *MESSAGE* request, the vendor includes a request for registration with the broker (*VendorRegistrationRequest*). This registration process is marked *LP-SIP* header as below:

```
LP-SIP: VendorRegistration
```

Moreover, the message contains in the body (indicated with the *application/lp-sip* content-type):

$E_B(S_V$ (V, B, VRID, $K_{EnV\ BMasterKey}$, $K_{SigV\ BMasterKey}$, VendorAccount))

where:

- *V*, *B* are the identifiers of the *Vendor* and *Broker*, respectively. These party identifiers are the digest of the parties' public key and are used to avoid impersonation attacks [32].
- *VRID (Vendor Registration Identifier)*. VRID identifies the registration process that is being performed between the vendor and broker. This identifier or label could be a randomly generated number. However, due to the importance of this process and in order to provide a high level of security we have decided to follow the principles proposed in [33]. These design principles recommend that the label has the following properties: verifiability, uniqueness and secrecy. Therefore, we have generated this identifier as H(V, B, H($K_{EnV\ BMasterKey}$), H($K_{SigV\ BMasterKey}$), VendorAccount).
- $K_{EnV\ BMasterKey}$ *(Master Encryption Key)*. This is a symmetric master key generated by the vendor that will be used by the broker to provide the clients with encryption keys to communicate with the vendor confidentially. The default symmetric cipher to employ is AES.
- $K_{SigV\ BMasterKey}$ *(Master Signature Key)*. This is a symmetric master key generated by the vendor that will be used by the broker to provide the clients with symmetric signature keys to guarantee the integrity of the communication with the vendor. The default cryptographic checksum function to employ is hash-based message authentication code (HMAC) [34] with SHA2 [35].
- *VendorAccount*. The bank account number that the broker will use to pay the vendor the amounts of e-coins he deposits. Each vendor has a unique account associated, which could be changed when needed.

This message allows the vendor to register his account with the broker as well as the master keys that the broker will use to provide symmetric keys to the clients in order to communicate with the vendor in a way that guarantees authentication, integrity and confidentiality.

2. $B \rightarrow V$: MESSAGE (VendorRegistration-Response)

In this message, the *LP-SIP* header has the same content as the previous message. In the body of the message, the content of the *VendorRegistrationResponse* is:

$E_{K_{VB}}$ (S_B(B, V, VRID, Confirmation)) where:

- K_{VB}. This symmetric key is generated from $K_{EnV\ BMasterKey}$ and broker identifier (B) in the following way: $K_{VB} = H_{K_{EnV\ BMasterKey}}(\mathrm{B})$.
- *Confirmation*. This is a flag that indicates whether the process has been successful. This message is to confirm whether the registration process was performed successfully.

5.2 Client registration process

For a client to be able to make purchases in the system, she has to obtain e-coins from a broker. But, prior to this she needs to perform a registration with a broker. In this process the user establishes the account that will be used to charge her for the e-coins she requests.

As a result of this registration process, the user obtains some symmetric keys to communicate with the vendor during the purchase process. The different messages exchanged in this process are:

```
1. C → B: MESSAGE (ClientRegistration-
Request)
```

For this message, the content of the *LP-SIP* header is:

```
LP-SIP: ClientRegistration
```

In the body of this message, the *ClientRegistration-Request* contains:

$\mathrm{E}_B(\mathrm{S}_C\ (\mathrm{C, B, CRID,}\ \mathrm{K}_{Enkey}, \mathrm{ClientAccount}))$

where:

- *C*, *B* are the identifiers of the *Client* and *Broker*, respectively.
- *CRID (Client Registration Identifier)*. *CRID* identifies the registration process that is being performed between client and broker. For the generation of this identifier or label we have followed the same approach as explained in the vendor registration process, i.e., $\mathrm{CIRD{=}H(C, B, H(K_{Enkey}), ClientAccount)}$.
- K_{Enkey}. This is a symmetric key that is generated by the client. The key is only used to preserve the confidentiality of the response message.
- *ClientAccount*. The client's bank account number.

This message allows the client to send her bank account information. In this message the client also sends a symmetric key (K_{Enkey}) that the broker uses to send the response to the client.

```
2. B → C: MESSAGE (ClientRegistration-
Response)
```

The header of *LP-SIP* is the same as the previous message and in the body the content of *ClientRegistration-Response* is:

$\mathrm{E}_{K_{Enkey}}\ (\mathrm{S}_B(\mathrm{B,\ C,\ CRID,}\ \mathrm{K}_{CB},\ \mathrm{K}'_{CB}, \mathrm{Confirmation}))$

where:

- K_{CB}. This encryption symmetric key is generated from $K_{EnC\ BMasterKey}$ and client identifier (C) in the following way: $K_{CB} = H_{K_{EnC\ BMasterKey}}(\mathrm{C})$. $\mathrm{K}_{EnC\ BMasterKey}$ is a master secret key that the broker uses to derive the different keys he will use to maintain the confidentiality of the following messages between broker and client.
- K'_{CB}. This signature symmetric key is generated from $K_{SignCBMasterKey}$ and client identifier (C) in the following way: $K'_{CB} = H_{K_{SignC\ BMasterKey}}(\mathrm{C})$. $\mathrm{K}_{SignCBMasterKey}$ is a master secret key that the broker uses to derive the different keys he will use to guarantee the integrity and authenticity of the following messages between broker and client.

As a response to the client's request, the broker sends a confirmation with two symmetric keys (K_{CB}, K'_{CB}) that will be used by the client in future communications with the broker, e.g., for requesting an e-coin.

5.3 Withdrawal of e-coins

The client performs the withdrawal process of e-coins in order to obtain electronic money to make the payment of the purchases of the services she wants.

The e-coin(s) the client obtains can be used in different transactions and with different vendors as long as the amount of the purchases is lower than the e-coin value. The amount of the e-coins withdrawn are charged to the client's associated account that was provided during the registration process. This charge can be made at this very moment or at the end of a period established between the broker and the client.

Additionally, at the same time the user requests the e-coins, she could request the keys that she needs to communicate in a secure way with the vendors.

When a client wants to obtain some e-coins, the different messages exchanged in this process are:

1. `C` $\rightarrow$ `B: MESSAGE (WithdrawalRequest)`

The *LP-SIP* header contains:

```
LP-SIP: Withdrawal
```

The *WithdrawalRequest* included in the body contains:

`C,` $E_{K_{CB}}$ `(`$H_{K'_{CB}}$ `(C, B, WID, Amount`$_1$ `[, . . . , Amount`$_n$`], [VendorKeysReq]))`

where:

- *withdrawal identifier (WID)*. WID identifies the withdrawal process that is being performed between client and broker. It is generated as WID=H(C, B, vCall-ID, Amount$_1$[, . . . , Amount$_n$]) using the value of the *Call-ID* header (denoted as *vCall-ID*) included in the message.
- Amount$_1$ [, . . . , Amount$_n$]. These indicate the amounts of the different e-coins the client wants to withdraw. Thus, for each amount indicated, the broker has to generate an e-coin of the quantity requested. The amount is expressed according to the ISO 4217 that indicates the quantity and the currency. The client, at least, will request an e-coin. As an example, the user could want 3 e-coins with different values: one of 1 €, 1 of 1 $ and one of 3 €.
- *vendor keys request (VendorKeysReq)*. This a list containing the identifiers of the vendors the user wants to communicate with. `VendorKeysReq = V`$_1$`, V`$_2$`, . . . , V`$_n$.

When the broker receives the request, from the client identifier, the broker obtains the keys the client used to cipher and symmetrically sign the message. For this purpose, the broker uses the same master keys that were used during the client registration. If the message is correctly deciphered and the signature is valid, the broker mints the different e-coins requested.

For each amount indicated in the request an e-coin is generated as we show next.

`e-coin`$_X$ `= ID`$_X$ `, B, Amount, C, Expiry,` $H_{K_{BCoin}}$ `(ID`$_X$ `, B, Amount, C, Expiry)`

where $x = 1, \ldots, n$ (the number of e-coins)

In the e-coin, the ID_X is its identifier, B is the broker's identifier that has minted it, the *Amount* indicates its value, C indicates the client the e-coin is generated for, and, *Expiry* indicates the validity of the e-coin.

This e-coin is specifically generated for a client and she can use it with different vendors and in different transactions as long as the value of the transactions does not exceed its value. Moreover, the amount spent in each transaction may be different, so we can consider the e-coin is divisible. This e-coin can only be validated by the broker. Later, we will explain when the broker has to participate in a transaction.

As a response to the client request, the broker sends the different e-coins in the following message:

2. `B` $\rightarrow$ `C: MESSAGE (WithdrawalResponse)`

In this message, the broker includes the same *LP-SIP* header as the previous message and, in its body, the *WithdrawalResponse* contains:

$E_{K_{CB},K'_{CB}}$ `(B, C, WID, e-coin`$_1$ `[. . . , e-coin`$_n$`], VendorKeys)`

where:

- *VendorKeys*. This represents the different keys the client can use to communicate in a secure way with the different vendors. Its value is: V_1, K_{CV_1}, $K'_{CV_1}, \ldots,$ V_n, K_{CV_n}, K'_{CV_n}.

Apart from the e-coins, the broker provides a set of keys to communicate with the different vendors which the client requested in the previous step. Later, if the client needs keys for a new vendor, by using the keys provided in the registration phase, she can communicate with the broker to obtain them as we explain now.

5.4 Obtaining vendors keys

When a client wants to obtain vendor keys to communicate with them, the different messages exchanged in this process are:

1. `C` $\rightarrow$ `B: MESSAGE (KeyRequest)`

The *LP-SIP* header contains:

```
LP-SIP: KeyRequest
```

The *KeyRequest* included in the body contains:

`C,` $E_{K_{CB},K'_{CB}}$ `(C, B, KRID, VendorKeysReq)`

where:

- *key request identifier (KRID)*. It identifies the key request process that is being performed between client and broker and is generated as KRID=H(C, B, vCall-ID, VendorKeysReq)

As a response to the client request, the broker sends the different keys in the following message:

2. B $\rightarrow$ C: MESSAGE (KeyRequestResponse)

The body of this message contains:

$E_{K_{CB},K'_{CB}}$ (B, C, WID, VendorKeys)

This process can be used at any moment the user needs to establish a new relationship with a vendor that works with that broker. However, this process will be used seldom since the number of vendors a user, usually, works with is small and keys can also be requested at the same time the user withdraws e-coins.

5.5 Purchase protocol

Of the different processes described so far, the purchase process is the process that will be used more frequently. This process will be executed at the same time as the initiation of the SIP session, except for the pay-as-you-watch model, in which the payment is made once the session is initiated and some data is received.

When the user wants to access a payment-based service, the messages exchanged are shown in Figure 3, and are explained below:

1. C $\rightarrow$ V: INVITE

The client sends this message to start the set up of the session. The message contains an SDP description of the session in the body. She also includes the *lpayment* option-tag for the *Supported* header to indicate that LP-SIP is supported. The client could also specify the *additionalpayment* option-tag if it supports additional payments during the session. Also, the client should include

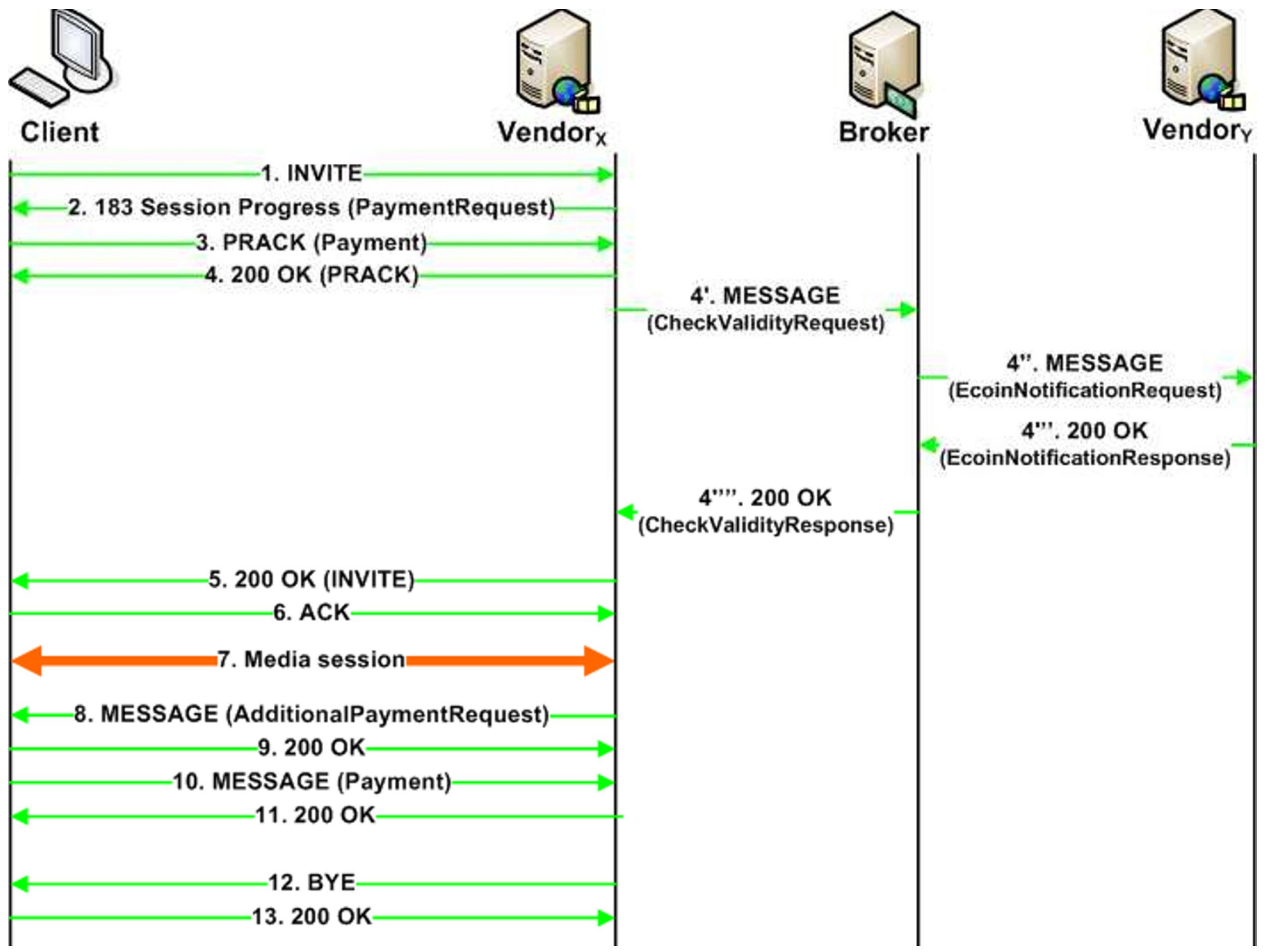
Client
VendorX
Broker
VendorY
1. INVITE
2. 183 Session Progress (PaymentRequest)
3. PRACK (Payment)
4. 200 OK (PRACK)
4'. MESSAGE (CheckValidityRequest)
4''. MESSAGE (EcoinNotificationRequest)
4'''. 200 OK (EcoinNotificationResponse)
4''''. 200 OK (CheckValidityResponse)
5. 200 OK (INVITE)
6. ACK
7. Media session
8. MESSAGE (AdditionalPaymentRequest)
9. 200 OK
10. MESSAGE (Payment)
11. 200 OK
12. BYE
13. 200 OK

Figure 3 LP-SIP Purchase protocol. This figure depicts in detail the messages that are exchanged in LP-SIP during the purchase process.

the *application/lp-sip* in the *Accept* header to signal the SIP messages that can support the exchange of the messages of the LP-SIP protocol.

```
2. V → C: 183 Session in Progress
(PaymentRequest)
```

When the vendor receives an *INVITE* message, if the service requires a payment and the *lpayment* option-tag is not included in the *Supported* header, the vendor returns a *402 (payment required)* message.

Otherwise, the vendor returns this message with the *LP-SIP* header containing a *PaymentRequest* indicating the prices of accessing the service in the different currencies and the models supported. The header indicates the different models supported and for each model the initial amount required to initiate the session and, if supported, the different prices for the additional payments. Thus, the *LP-SIP* header would be, simplified, as follows (the whole specification is in Section 4.3.2):

LP-SIP: PaymentRequest; PID=pidv modelID=mID$_1$ brokers= 'B' model=modeltype unit=quantity$_1$ initialPayment; amounts= 'amount$_1$, amount$_2$, . . . ' additionalpayment; modelID=mID$_2$, brokers='B' model=modeltype unit=quantity$_2$ amounts= 'amount$_1$, amount$_2$, . . . Expiry=datetime

In this header *payment request identifier (PID)* is a identifier of the transaction generated by the vendor and *modelID* identifies the model.

In a payment request there could be different payment models to choose (pay-per-view, pay-per-data, etc). These models could be based on the amount of time spent in the session (pay-per-time), on the amount of data received (pay-per-data), etc. For each model (the type is indicated in *model* and identified by *modelID*) the vendor indicates the price (attribute *amounts*) based on different units (attribute *unit*), e.g., in a pay-per-time model the prices could be expressed in units of 5 minutes that cost 0.5 € each unit. With the *brokers* attribute he indicates from which brokers he accepts e-coins for the payment. This header also indicates until when this offer is valid, by means of the *Expiry* attribute. Similarly, the vendor also indicates the price of additional payments with the information included from *additional-payment* attribute.

```
3. C → V: PRACK (Payment)
```

If the client agrees to the payment, she sends the *PRACK* message with the *LP-SIP* header indicating the exchange is a payment. The message also contains the payment message in its body with content-type *application/lp-sip*. The client indicates the following information in the *LP-SIP* header:

```
SIPPayment: Payment; DateTime=date-
time-value
```

In the content of type *application/payment* the following message is included:

C, $E_{K_{CV},K'_{CV}}$ (C, V, B, vCall-ID$_1$, pidv, date-time-value, fKey, [flowKey,] modelID$_Y$, e-coin$_X$, L_i, U_i, InfoToB)

where:

InfoToB = $H_{K'_{CB}}$ (C, V, B, ID$_X$, date-time-value, L_i, U_i)

This message contains *vCall-ID*$_1$, which indicates that the value of *Call-ID* header is from message 1 (the *INVITE* message) and that it is the same for all the messages of the initiation process. The *pidv* indicates the *PID* value generated by the vendor for this transaction. The *modelID*$_Y$ is chosen from the *PaymentRequest* message with the e-coin. L_i and U_i indicate the portion of the e-coin that is going to be spent. The *fKey* is a flag that the client can use to indicate to the vendor that, if possible, she wants to receive the flows of the service in a ciphered way with the key *flowKey*. Besides the e-coins, the message also contains some information to the vendor to confirm the payment with the broker (*InfoToB*).

In a payment, L_i indicates the amount of the e-coin that has been spent so far and U_i indicates the limit to spend in this payment. Thus, the amount to pay in the transaction is U_i-L_i. For example, with an e-coin of 1 €, the first time a payment is made, L_i is 0 and if the amount to pay is 0.2 €, then U_i is 0.2. In the next payment, L_i will be 0.2. For the sake of simplicity, we have included the information of one e-coin, though different e-coins could be included if necessary. In this case, the client would include each e-coin and its respective parts of the e-coin to be used.

```
4. V → C: 200 OK (PRACK)
```

With this message the vendor confirms the reception of the e-coin. When the vendor receives the *PRACK* message (message 3) with the e-coin there are two possible situations:

1. During the day the transaction is taking place, either the vendor has received this e-coin for the first

time or he has had it but another vendor obtained it later. In both cases, the vendor has to contact the broker to check whether the e-coin is valid, was minted by the broker and what portion has been spent. As a response, the broker indicates if the e-coin is valid and the amount that has been spent. Thus, the vendor can calculate whether the client can use the e-coin to make the payment.

2. The vendor has previously received an e-coin during the day and he has not received a notification that the e-coin is being spent with another vendor. In this case, the vendor does not need to check with the broker the amount spent because he knows it, since he was the last vendor to receive a payment with that e-coin.

In the first situation, the vendor ($Vendor_X$), before sending a response in the message 5, has to exchange the messages we describe next (from 4' to 4""). In the second situation, the vendor continues the process with message 5, explained below.

4′. V $\rightarrow$ B: MESSAGE (CheckValidity-Request)

The *LP-SIP* header for this message is:

```
LP-SIP: CheckValidity
```

The *CheckValidityRequest* contains:

V, $E_{K_{VB},K'_{VB}}$ (V, B, vCall-$ID_{4'}$, C, ID_X , L_i, U_i, e-$coinID_X$, date-time-value, InfoToB)

Basically, in this message the vendor sends the e-coin to the broker with the information the broker needs to check it. If the e-coin has not already been used during this day, the broker checks the amount of the e-coin spent so far and whether the amount spent is consistent with that information, that is, the value spent has to coincide with the valued indicated in L_i. In this case, the broker registers that the vendor is receiving a payment with this e-coin and sends a confirmation to the vendor with message (4"") as shown below. Thus, the exchange would be messages 4' and 4"" .

However, if the e-coin has already been used, the broker has to contact the previous vendor ($V\ endor_Y$) to obtain the amount spent, as we see next:

4″. B $\rightarrow$ V′: MESSAGE (EcoinNotification-Request)

In this message, the *LP-SIP* header is:

```
LP-SIP: EcoinNotification
```

The *EcoinNotificationRequest* contains:

B, $E_{K_{V'B},K'_{V'B}}$ (B, V', vCall-$ID_{4''}$, ID_X , C)

With this message, the broker requests the $Vendor_Y$ to indicate to him the last portion of the e-coin that the client spent with him. This information is sent in the following message.

4"′. V′ $\rightarrow$ B: 200 OK (EcoinNotification-Response)

In this message, the *LP-SIP* header is the same as the previous message. In the body, the *EcoinNotificationResponse* contains:

$E_{K_{V'B},K'_{V'B}}$ (V′, B, vCall-$ID_{4''}$, ID_X, C, U_i)

In this message the $Vendor_Y$ sends the last portion of the amount that the client spent with him (U'_i). The U'_i value has to coincide with the initial portion of the e-coin (L_i) that the new vendor is requesting.

Both if the broker has to check with the previous vendor the amount spent and if this step is not necessary, because the broker already knows it, as a response to the vendor request on the e-coin, the broker sends:

4"″. B$\rightarrow$ V: 200 OK (CheckValidityResponse)

In this message, the *LP-SIP* header has the same value as message 4′ . As a response, in the *CheckValidityResponse*, the vendor receives:

$E_{K_{VB},K'_{VB}}$ (B, V, vCall-$ID_{4'}$, C, L_i)

With this message the vendor receives the confirmation on the amount of the e-coin that has been spent (L_i). If $Amount_X$ - L_i ≥ *Price* of the product/service, the vendor accepts the payment and provides the access to the product or service requested by confirming the establishment of the session by sending message 5. Otherwise, the vendor would send an error indicating the payment is not valid.

Once the vendor has received this information he can control the amount of e-money the client can use in subsequent payments that take place in the same day. Therefore, in the next payment, messages 4" and 4"' are not exchanged. This is the second situation we mentioned.

5. V $\rightarrow$ C: 200 OK (INVITE)

With this message, in the header section the vendor includes the following header:

LP-SIP: Payment; DateTime=date-time-value, EncryptedKey=keyvalue

Thus, the vendor confirms the payment has been received correctly. Furthermore, with this header, the client could receive (if requested during the payment) a key (*flowkey*) that will be used to cipher the session flows that will be exchanged when the session is established. This key is received in a ciphered way in the *EncryptedKey* attribute, whose value is: keyvalue= $E_{K_{CV},K'_{CV}}$ (date-time-value, flowkey). Next, the client acknowledges the reception and the session starts. Thus, the client receives the access to the service.

6. C $\rightarrow$ V: ACK

This message finishes the establishment of the SIP session and, then, the different multimedia flows are exchanged. This message does not need any additional header or content.

7. C $\longleftrightarrow$ V: Multimedia information

This flow depends on the protocol and media agreed on during this session's establishment. The flow is relayed until either the session finishes (messages 12 and 13) or the time/data paid for by the user is up. When the time/data the user paid is about to finish, if the user supports the additional payment mechanism, messages 8-11 are exchanged. Otherwise, the session finishes (messages 12 and 13).

8. V $\rightarrow$ C: MESSAGE (AdditionalPayment-Request)

This message is used by the vendor to request an additional payment (if this mechanism is supported) to the client when the time/data the client paid is about to finish.

In the *LP-SIP* header, the vendor indicates the time or data that rests before the session finishes with the *Time-DataStop* structure. It also contains information about the prices to pay in a similar way as expressed in the *PaymentRequest* message.

SIPPayment: AdditionalPayment; PID=-pidv TimeDataToStop=value modelID=maID brokers= 'B' model=modeltype unit=quantity amount=amount_1 Expiry=datetime

Thus, the prices could be different according to the model (for the whole specification see Section 4.3.2).

The client sends a *200 OK* as a response to this request to confirm the reception of this message.

9. C $\rightarrow$ V: 200 OK

This response does not contain any additional header or content. If the client decides to continue with the session and makes a new payment, she sends a *MESSAGE* request with the e-coin previously used with this vendor and with a new payment portion, as long as the remaining value of the e-coin is greater than the amount to pay. Otherwise, the client would have to include an additional e-coin.

10. C $\rightarrow$ V: MESSAGE (Payment)

The content of this message is similar to message 3, previously explained. As a response to this message, the vendor sends the following message:

11. V $\rightarrow$ C: 200 OK

If the payment is correctly made, the session continues during the time or the data the client has paid for. Otherwise, the vendor sends an error message and the session finishes (with messages 12 and 13).

This session finishes at the end of the time/data that the session lasts or when the client is not willing to make more payments. In both cases, the session finishes with the following messages:

12. C $\rightarrow$ V: BYE

This indicates the client wants to finish the session.

13. V $\rightarrow$ C: 200 OK

The vendor confirms the reception of the message and the session finishes.

5.6 Deposit protocol

At the end of the day, the vendor sends the broker the amounts received from the different e-coins. Thus, the broker can perform the deposit of value received by the vendor. The messages exchanged are:

1. V $\rightarrow$ B: MESSAGE (DepositRequest)

In this message, the *LP-SIP* header is:

```
LP-SIP: Deposit
```

The *DepositRequest* contains:

V, $E_{K_{VB},K'_{VB}}$ (V, B, vCall-ID$_1$, ID_{X_1}, L$_1$, U$_1$ [, . . . , ID_{X_n}, L$_n$, U$_n$, . . .])

With this message, the vendor indicates, for each e-coin, the portion of the e-coin that he has received during all the day (L_i indicates the starting position and U_i the last value received). As a response, the broker indicates with the $flag_X$ value if the payment deposit has been correctly processed for each e-coin.

2. B $\rightarrow$ V: 200 OK (DepositResponse)
$E_{K_{VB},K'_{VB}}$ (V, B, vCall-ID$_1$, ID_{X_1}, flag$_1$ [, . . . , ID_{X_n}, flag$_n$, . . .])

6 Analysis of the LP-SIP protocol

In this section, we make an analysis of the LP-SIP protocol with regard to the requirements defined in Section 3 in order to show how this protocol satisfies them. We also describe how the protocol allows us to support the use of the case scenarios introduced in Section 2.3.

6.1 Payment protocol based on extensibility mechanisms

LP-SIP is based on the extensibility mechanisms defined in SIP [1]. We have defined new option-tags (*lpayment*, *additionalpayment*, *application/lp-sip*, Section 4.3.1) and a new header, *LP-SIP* (Section 4.3.2). Thus, we facilitate LP-SIP being supported in current implementations of SIP and, therefore, its acceptability.

6.2 Payment models

The protocol can support different payment models: pay-per-time, pay-per-data, session-based charging and pay-as-you-watch. In general, these models are supported because the protocol allows us to make a payment at the beginning of the session (in the three-way protocol combined with the *PRACK* method, see messages from 1 to 6 in Section 5.5) as well as at any moment once the session is initiated (with the additional payment mechanism, see messages 8 to 11 in Section 5.5).

In the session-based charging model the payment is made for the whole session. Thus, there is only one payment, made at the beginning of the session.

In the pay-per-time and the pay-per-data models, apart from a payment at the beginning of the session (messages from 1 to 6 in Section 5.5), it is required that a payment is made from time to time (e.g., every thirty seconds, every minute, etc). Thus, the vendor has to control the time/the data sent and requires a payment once the unit of consumption (seconds or bytes) defined is spent. These payments are requested and made by using the additional payment feature (messages from 8 to 11 in Section 5.5).

The support of the pay-as-you-watch model is different from the models we have just presented. In this model, the session is initiated without a payment. Thus, the *INVITE* message contains the headers indicating the support of LP-SIP. However, the *183 Session in progress* does not contain a *PaymentRequest* and the *PRACK* method is not necessary, with the session being initiated on SIP as usual.

Once the session has been established and the user has received some data (during some time) a payment request is sent. Thus, in *MESSAGE* request, instead of sending an *AdditionalPaymentRequest*, a *PaymentRequest* would be sent. The payment would be received in the *200 OK*. Once the payment is made, the session would send some additional data during some time and a new payment request would be sent. All payments after the initial payment are requested using the *AdditionalPaymentRequest* message.

Thus, thanks to the possibility of making payments both at the beginning of the session and once it is initiated, our proposal supports different payment models.

6.3 Additional payments

As shown throughout the article, LP-SIP supports making additional payments once the session is initiated. For this purpose and based on the SIP extensibility mechanisms, LP-SIP extends the *MESSAGE* request and the *200 OK* response in order to support the request and send additional payments without interrupting the session. Thus, we can support different models, as mentioned in the previous section.

6.4 Time/data notification

In order for the user to be aware of the remaining time paid associated to the session, the LP-SIP protocol notifies users via the additional payments feature. Thus, at the same time the time is notified, a new payment is requested (if required). Therefore, we satisfy the time/data notification requirement.

6.5 Lightweight and efficient protocol

To support real-time payments, the protocol has to be lightweight and efficient. For this purpose, the purchase process has been designed using symmetric cryptography (encryption based on a symmetric algorithm, such as AES, and signature based on HMAC with a secure hash function, such as SHA-2), which is much more lightweight than asymmetric cryptography. We have also avoided the broker having to participate in the transaction for every payment. Furthermore, our protocol is more lightweight than previous studies, as presented in Section 8. Therefore, LP-SIP can be considered lightweight and efficient.

6.6 Security analysis

In this section we provide an analysis of the different properties that the protocol satisfies in terms of security. The main objective of the payment system is to provide security and efficiency during the highly used payment phase. As mentioned in the previous section, security is provided by means of symmetric cryptography. In other processes where there is no previous contact between entities, such as client and vendor registration processes, the asymmetric encryption and digital signatures are used. In this system there are two possible kind of attackers: internal and external.

On the one hand, external attackers of the transaction such as other entities that compete in the system as other brokers, vendors or clients (whose aim is for an entity to have a bad reputation and thus, the number of its users would be reduced in benefit of another entity in the system, e.g., a vendor could change the information exchanged between a client and another vendor so that the client has to pay more quantity for the product and the vendor could be seen to be cheating the client) or any other entity that wanted to obtain benefits from the payment transaction.

On the other hand, internal attackers of the transaction, i.e., the client, and the vendor (the broker is considered a trusted party) could aim to obtain more benefits fraudulently from the transactions, e.g., the client trying to re-use an e-coin already spent to obtain new content, etc. In our justification of security properties, instead of referring to them as internal attackers we will mention them by name in order to distinguish between the attacks that the client can try from those of the vendor.

In the following sections, we analyze in more detail the security properties and how we can avoid the different attacks that clients, vendors and external attackers (or simply attackers) can try to carry out.

6.6.1 Integrity and authentication

The integrity of all the messages is guaranteed by different mechanisms: in the vendor and client registration processes by means of digital signatures, and in the withdrawal and deposit of e-coins and in the purchase process by means of HMAC checksums. Thus, any change made by an external attacker to the messages of the transaction will be detected by the entities participating in the system.

In the transactions where a digital signature is used (registration processes), the recipient can be sure that nobody except the sender has signed the message. Thus, this digital signature guarantees non-repudiation.

In the messages where symmetric signature based on HMAC is used (withdrawal, purchase and deposit processes), although the signing key is shared between the entities, the recipient can be sure that the message comes from the other party (if we suppose that the sender has not revealed his/her key to another party, since this is against his/her own interests). With this symmetric signature, non-repudiation is not guaranteed but it is not required, since the purpose is to guarantee that the information cannot be produced by an external attacker. Non-repudiation is analyzed in more detail below.

6.6.2 Replay attacks

An external attacker could copy a message exchanged between the client and the vendor and he could re-send it later in order to try to obtain the access to the service or that the client is charged more than once. However, if that attacker re-sends an old message the vendor will detect that the current identifier contained in the message for the transaction (*PID* value) does not match with the current transaction and that the e-coin and the amount indicated in the message has been already spent.

Furthermore, if the vendor has previously received the e-coin, he knows the portion of the e-coin spent and he can check that this portion has already been spent. If the vendor does not know this information, he will query the broker. Therefore, the vendor can detect if the payment was made and in this case, the vendor will not charge the client and will not provide access to the service.

6.6.3 Confidentiality

In the LP-SIP protocol, we have used both symmetric and asymmetric cipher algorithms to ensure the confidentiality of the information exchanged. In the different messages, we have, as far as possible, used symmetric cryptography for the sake of efficiency. This is especially interesting in the purchase process since this will be used most frequently. In this process the real-time requirements are fundamental so that access to real-time services is provided seamlessly. It is also used in other processes such as the withdrawal and deposit of e-coins.

Furthermore, in the event the user considers that the information contained in the flow should not be eavesdropped by external attackers, she can request the vendor to cipher the communication with a symmetric key as well. This key is received during the purchase process in a secure way. Therefore, supposing perfect cryptography, the content of the flow cannot be accessed by an external attacker unless the key is revealed by the vendor or client.

As symmetric cipher we propose AES because since it was introduced no significant security problems have been revealed. In those cases where there had been no previous contact between the entities (client and vendor registration processes), it is necessary to use asymmetric cryptography.

Therefore, as all the communications are ciphered and if we suppose perfect cryptography, no external attacker can eavesdrop the information exchanged between the different parties.

6.6.4 Key distribution

The broker is the entity responsible for distributing to the clients, the keys that they will use to communicate with vendors. This distribution can be made in two different processes: when the user withdraws e-coins (see Section 5.3) or at any moment after registration with the key request process (see Section 5.4).

In both processes the communication between the user and the broker is ciphered with the keys that the broker generated and distributed to the client (K_{CB}, K'_{CB}). These keys are only known by them and unless one of them reveals them, no external attacker can eavesdrop the vendors' keys the client obtains. Thus, if we suppose perfect cryptography for the keys used in AES algorithm for ciphering and HMAC-SHA-2 for integrity, the information exchanged cannot be accessed by an external attacker.

Another possible attack that the client could try is to obtain the vendor's master key in order to eavesdrop other communications between the vendor and other clients. There are two possible ways to obtain it. First, by eavesdropping the vendor's registration process, since in this process the vendor sends master keys to the broker. However, the communication is ciphered. Second, from a client key, calculating the master key by brute force or finding collisions in the hash function (SHA-2) used in the HMAC. But these attacks are not possible if we suppose perfect cryptography.

6.6.5 Non-repudiation

Non-repudiation aims to provide the evidences needed in the event that a party (client or vendor) denies his/her participation in a transaction or having received the content. This feature is fundamental when the amounts involved in the transactions are large. However, as mentioned in [36,37], in purchase transactions where it is expected that the amounts involved will be small or very small (micropayments) as in our payment system, non-repudiation could be considered unnecessary in order to save some computations and perform the transactions more efficiently.

As in most micropayment schemes, the model is based on the fact that the amount involved in the transactions is (very) small. Therefore, the profit the vendor can obtain by cheating the user (not providing the access to the service or content) is very low and the consequence would be that he would obtain a bad reputation, which could suppose that no new customers decide to make purchases with him [5], as well as the broker possibly deciding to drop the vendor from the system. At the same time, the broker can check the amounts spent by each customer and control if a user is making frequent complaints to avoid the payment, in this case the user will be dropped from the system.

In the event of non-repudiation being needed, the purchase protocol could be extended by substituting the "signature" based on HMAC by an electronic signature based on public-key cryptography or using S/MIME headers to sign the SIP messages [38,39].

In registration and vendor keys, non-repudiation is guaranteed so that the broker can prove to the entity that manages client and vendor accounts (if it is not itself) that the client and/or vendor really provided him her/his account.

6.6.6 Bogus e-coin

The e-coins are only generated and verified by the broker. Therefore, unless an (internal or external) attacker knows the keys used to mint the e-coins, it is not possible to use bogus e-coins.

Each e-coin is protected by using a symmetric key that depends on the client identifier. If we suppose perfect cryptography, an (internal or external) attacker cannot obtain the key associated to generate e-coins for a particular client. As an additional control measure, the broker could control the serial numbers generated for the different clients in order to detect a possible attack on the key used to generate e-coins for a client. In this case, the identifier will be drawn.

Even if an (internal or external) attacker has managed to obtain the key for a client (not possible under the conditions we have just explained), if he wanted to generate e-coins for a different client, he would have to obtain the broker's master key. But this attack is not possible if we suppose perfect cryptography and it can also be detected in the same way as we have just mentioned for a particular client master key.

6.6.7 Double spending/overspending

Double spending/overspending is an attack that the client can try during the payment process. The control of whether an e-coin is double spent or not is performed between the broker and the vendors.

The first time a vendor receives the e-coin from the client he has to contact the broker in order to ascertain if the e-coin is valid and the total amount of the e-coin that has been spent. The broker checks, from its serial number, if the e-coin is being used with another vendor and in that case checks the amount spent with that vendor. He also notifies that vendor that the e-coin is going to be used with another vendor.

As a result of this process, the broker provides the new vendor the amount of e-coin that has been spent. Thus, the vendor can check if the amount to pay has already been used or not. Subsequent payments in a short period of time (established by the broker, usually a day) will be controlled by the vendor until he receives a notification from the broker indicating that the e-coin is going to be used with another vendor.

In order to limit the validity of a serial number of an e-coin, each e-coin incorporates an expiry date. Thus, the broker can also check if an e-coin that was used a long time ago is valid or not.

6.6.8 Double deposit/over deposit

In a similar way as the client can try to double spend an e-coin, the vendor could try to deposit the same e-coin with the broker several times. Thus, the vendor would receive more money in his account. However, the broker can control this situation thanks to the identifier contained in the e-coin as well as the information about the portion of the e-coin spent in previous transactions.

Each time the broker receives a deposit, he obtains the identifiers of each e-coin as well as the portion of the e-coin deposited (L_i - U_i). Then, for each e-coin, he checks that the e-coin has not expired and that portion has not been spent yet (for each e-coin the broker stores the last portion spent–U_i–until the e-coin expires). If the verification is successful, the broker updates his information and pays in the amount to the vendor's account. Otherwise, the attack is detected. The broker only stores information on e-coins that have not expired. Thus, we limit the information that the broker has to store.

The vendor could also try to charge more money than he actually received, e.g., a vendor (let's say Vendor$_X$) that should charge 0.3 € from a 1 €e-coin could request from the broker a payment of 0.9 €. This attack may be performed in the deposit process by changing the limits the vendor sends to the broker, e.g., Vendor$_X$ could send 0 as L$_i$ and 0.9 as U$_i$. If the e-coin was used with more vendors during the day, the broker will detect the over deposit because he will see that intervals received by the different vendors for the same e-coin are overlapping. The broker can also know, previous to the deposit process, the amount to pay to each vendor if each time there is a change in the vendor that is receiving the e-coin he registers the amount spent with each vendor. Thus, the broker can make a double check. Continuing with the example we have just mentioned, if the client uses the same e-coin with Vendor$_Y$ to make a payment of 0.4 €, he would send to the vendor 0.3 as L$_i$ and 0.7 as U$_i$. Then, at the end of the day when the broker receives the two deposits he will see that the e-coin interval of Vendor$_X$ (0-0.9) overlaps with the e-coin interval of Vendor$_Y$ (0.3-0.7). Therefore, one of them is trying to make an over deposit.

In order to determine which vendor is cheating, the broker would request the $InfoToB$ token (see message 3 in Section 5.5) that the client sends to the vendor when he makes a payment. This token confirms the amount to be paid to the vendor in the transaction and this information cannot be modified by the vendor since it is signed using a symmetric key shared only between the client and the broker. Thus, from the $InfoToB$ token received from each vendor, the broker can detect the vendor that is cheating.

6.6.9 Visibility

In a payment protocol, some of the data exchanged must be readable by only those parties needing this information to accomplish their tasks, e.g., vendor account number should be protected from the client, and likewise, the broker does not need to know the services the user is accessing, etc. Hence, this feature is to guarantee confidentiality (see Section 6.6.3) and not include more information than necessary in the messages.

As mentioned, the LP-SIP protocol uses asymmetric encryption and electronic signature to protect this partially confidential information from unintended readers and symmetric encryption based on AES and cryptographic checksums based on HMAC to exchange e-coins. Thus, the information exchanged in the payment system during the client registration phase and the withdrawal e-coin phase can be seen only by the client and broker. In a similar way, in the processes of vendor registration, check validity and deposit, the information is only seen between vendor and broker.

In the payment phase, the information exchanged can be seen only by the client, vendor and broker. No other party can see it. Furthermore, the protocol for the vendor, in the queries or in the deposit to the broker does not include information on the services the user is accessing. The broker knows only the different relationships established between the entities. The vendor cannot know the other vendors the e-coin has been used with since the e-coin does not contain this information and the interval does not provide this information. He cannot know the client's account either. This information is known only by the broker. Therefore, the vendor cannot obtain it if we suppose that broker behaves as a TTP.

In the event that client and vendor want to reduce the visibility of the information even more so that the broker cannot see it, they could use the messages of a payment either to transport a key transport protocol (the client could send a symmetric key encrypted with the public key of the vendor) or perform a key agreement exchange, negotiating other symmetric keys to perform the payment exchange or use an SSL/TLS channel or IPsec [38,39].

The problem of using network/transport level mechanisms (TLS, IPsec, etc) is that the protection end-to-end is not guaranteed when there are intermediaries in the SIP communication [40]. So, when a intermediary entity does not use it, the client and vendor identities are exposed and could be profiled. Several solutions to preserve privacy in SIP have been proposed [1,40-43].

From the solutions to preserve privacy in SIP the right choice is to use PrivaSIP [40,42,43] when both the protection of the user identity and authentication are required [40]. Otherwise, the use of Anonymous URI [1] is recommended [40]. Apart from using these solutions in LP-SIP we should replace parties' identifiers with pseudonyms.

In LP-SIP, the mechanism of Anonymous URI involves the use of an anonymous URI like "sip:anonymous@anonymous.invalid" in the SIP <From >header replacing client's identity. When authentication is required and we want to preserve both client and vendor identity with PrivaSIP we have to encrypt the client's identity and replace the display name by the string "anonymous" in the <From >header, replace the <Contact >header with the IP address of the client and encrypt the vendor's identity.

6.6.10 Formal analysis

Apart from the analysis of the different security properties of the protocol we have made in the previous section, we decided to check them by means of a formal analysis tool. We carried out a formal validation of the different protocols and sub-protocols proposed using the Automated Validation of Internet Security Protocols and Applications (AVISPA) tool [44,45]. It is important to point out that this tool allows us to check most of previous properties that are related to the secure exchange of information (the properties that are checked with this tool are detailed below. As far as we know, the checking of the properties related to privacy or payment such as double spending, bogus e-coins, etc is not supported). Thus, thanks to this tool, we can assert that the analysis made in previous sections is accurate.

In AVISPA, the validation process is the following. First, we specify our protocol in the High Level Protocol Specification Language (HLPSL) [46]. The notation is similar to the one we have used in the specification of the protocol. The main difference is the way in which the different cryptographic operations are denoted. Furthermore, we specify the different elements that are used in the validation of authentication and secrecy goals. Then, the AVISPA tool translates it into the Intermediate Format (IF) specification [44,45]. Finally, this IF specification is analyzed by invoking state-of-the-art back-ends that this tool provides, which are currently: On-the-Fly Model Checker (OFMC) [45,47], Constraint-Logic-based Attack Searcher (CL-AtSe) [45,48] SAT-based Model Checker (SATMC) [49], and Tree Automata-based Protocol Analyzer (TA4SP) [50]. These back-ends allow us to check a set of automatic analysis techniques, such as protocol falsification or abstraction-based verification.

Specifically, AVISPA allows us to check, from the specification of LP-SIP in HLPSL, if the machine of the LP-SIP protocol is correctly designed (non-deterministic protocols), and the following properties: replay attacks, confidentiality, impersonation, secrecy and authentication. Furthermore, the back-ends of this tool follow the standard Dolev-Yao model, in which the intruder is assumed to have control over the network.

This specification has been tested with the different back-ends mentioned above. As a result, these back-ends return attacks (if any). In our case, we have verified the properties previously mentioned successfully. The specification of the purchase process according to AVISPA is available in [51].

6.7 Support of the use case scenarios

In this section we analyze how the use case scenarios introduced in Section 2.3 are supported.

For scenario 1, both caller and callee have to play the role of client and vendor. In this scenario, the first time a caller calls a callee, she makes a payment according to the flow of messages from 1 to 6 presented in Section 5.5 (hereinafter all the messages refer to that section). Thus, the caller behaves as a client and the callee as a vendor. Once the payment is made, the callee includes the caller in a whitelist and the call takes place. If the call is not spam, prior to finalizing the session, the callee, behaving as a vendor, sends a payment request according to the flow of messages from 8 to 11 depicted in Section 5.5. Then, the session finishes with messages 12 and 13.

In scenario 2, a company can provide real-time services according to different business models such as pay-per-time, pay-per-data, session-based or pay-as-you-watch. When the client wants to access these services by initiating an LP-SIP session, the vendor informs about the different models supported and the prices associated by means of the *PaymentRequest* sent in the *183 Session in progress* answer (message 2). From this message, the flow is different, depending on the model.

In the session-charged model, the client makes a payment with message 3 and messages 4-6 are exchanged to finish the establishment of the session. Once the session is established the client accesses the real-time service (messages represented in step 7) until the session finishes with messages 12 and 13. In this model, the only payment made is at the beginning of the session.

In the pay-per-time or pay-per-data model, the session is initiated as in the previous model, then, the client accesses the real-time service for some time (e.g., 30 s) or some amount of data (e.g., 300 Kb). The control of the time and data is controlled by the vendor. When the time/data paid is about to finish, the vendor sends the request for an additional payment so that the client can access the service for some additional time or amount of data. Thus, messages 8 and 9 are exchanged. If the client agrees to continue with the session, she sends a new payment with messages 10 and 11. The flow of messages from 8 to 11 will be repeated each time the amount of time/data paid is about to finish and the user is willing to continue with the session. Otherwise, the session finishes with messages 12 and 13.

In the pay-as-you-watch model the approach is different to the model we have just described. In this model, instead of paying before receiving some data, the payment is made once the user has received some amount of data. Thus, in this model, the session is initiated without requesting any payment using the SIP three-way protocol. The only information included is the option-tags related to the LP-SIP protocol and defined for the *Required* and *Supported* headers, that is, *lpayment* and *additionalpayments* (see Section 4.3.1).

Once the session is initiated, after some data have been sent to the client, the vendor requests a payment. This payment is requested by using messages 8 and 9, but instead of an *AdditionalPaymentRequest*, a *PaymentRequest* is sent. If the client agrees to the payment, then, in messages 10 and 11, the payment is sent. Subsequent payments in the session are requested using messages 8 and 9 with *AdditionalPaymentRequest* and making payment with messages 10 and 11. Thus, the session continues while the client is making payments. Otherwise, the session finishes with messages 12 and 13.

Scenario 3 is similar to scenario 2 when the pay-per-time is followed. Thus, there is an initial payment during the initiation of the session for the establishment of the call. Then, once the session is initiated, an additional payment is requested each minute in order to continue with the call.

As we have explained in this section, thanks to the different mechanisms and the information exchanged in LP-SIP, the LP-SIP supports the use of different payment models.

7 Related study

There are two kind of mechanisms for charging SIP services: those that are based on accounting and those based on SIP-based payment mechanisms such as Fischl and Tschofenig's proposal [5], SIMPA [6], SIPCoin [7], Fan's et al. protocol [8], or Zhang's et al. protocol [9].

In general, accounting mechanism has been used for charging these services since it is based on infrastructures that were already deployed by network operators and service providers. Particularly, these accounting systems are based on AAA infrastructures. In these infrastructures the most used protocols are RADIUS and Diameter, although Diameter seems to be the choice in the establishment of new AAA infrastructures since it fixes RADIUS deficiencies [3]. However, Diameter has some vulnerabilities that mean that the accounting information might be not valid or accurate. As a response to these vulnerabilities SIPA Diameter application was proposed to offer a fully fledged accounting solution for SIP, which provides proper billing for services.

These solutions that are based on AAA can provide in an efficient way subscription-based service access type, but not pay-per-use service access type, since they require the credit-server to participate in each payment and the service fee calculation requires contacting another entity, such as the credit-control client or the credit-control server, unlike a (micro)payment solution where the service provider can calculate the fee without contacting another entity [10]. These drawbacks hinder the use of accounting systems for paying in an efficient and scalable way for real-time payment services in SIP. This is proved by the comparison between Diameter SIP application and Diameter credit-control application with a micropayment protocol as SIPCoin [10], which in turn, we improve.

Micropayments are the ideal mechanism for paying for (very) small amounts [13,17,18] since they are designed to be lightweight and can be used to pay for real-time based services [7,10]. In general, micropayments can be classified into token-based or account-based [20]. In token-based systems, a token or e-coin is used to make payments. Later, when the broker receives the token, he pays the vendor. In an account-based system, the customers authorize the transfer of e-money from the client's account to the vendor's account.

In order to support (micro-)payments in SIP we could consider integrating some of the existing payment protocols in SIP (either token-based or account-based). However, if we review the latest micropayment protocols defined [14-16], these have not considered the secure exchange of payment information or if they have defined it, they use asymmetric cryptography. Furthermore, although many have reduced the number of asymmetric operations in the purchase process, they still use some asymmetric operations in this process (asymmetric cryptography is about 100 times slower than the symmetric one [52]). Furthermore, some of them require the management of different tokens at the same time [15], which increases transaction costs [13].

Apart from this possible integration, there are several proposals that consider payment in SIP. These proposals, which are designed to make the payment in SIP are [6-9,12]. From these proposals we can point out that [5,6,8,9,11] are not suitable for making micropayments since a third party participates in each payment and asymmetric cryptography is used, which also involves certificates, verifications [53] and increased transactions costs [13]. Therefore, they are not suitable for making real-time micropayments.

On the other hand, Hao et al.'s [7] proposal, named *SIP-Coin*, is token-based, namely in hash-chains in order to be able to make payments efficiently. Their goal was to define a micropayment protocol based on the ideas of previous micropayment proposals but avoiding asymmetric cryptography. Thus, the micropayment could be made very quickly and real-time payment could be supported.

In SIPCoin, in each transaction (but not in each payment) a third party (a payment provider) participates to generate the hash chain to be used and the hash chain used is different; therefore, unspent hash chains are not reused. We can also mention that they do not define how to exchange payment information in a secure way that guarantees authentication, confidentiality and integrity, although they mention that mechanisms such as S/MIME, TLS and SIPS URI could be used for this purpose. In fact, at least one of these mechanisms should be used since, if not, the vendor could eavesdrop on the communication and obtain the root used to generate the hash chain and thus receive a payment of higher value than that specified by the vendor.

LP-SIP, as SIPCoin, is based on a token and is suitable for making micropayments since the payment phase is based on symmetric cryptography and a third party does not participate in all the payments, which is desirable to reduce attacks and costs [13]. We have also considered the secure exchange of payment information by using symmetric cryptography in order to guarantee authentication, confidentiality and integrity in a more efficient way than S/MIME or TLS, which require asymmetric cryptography, which is computationally less efficient than symmetric cryptography. Furthermore, we have reduced the participation of a third party in the payment. In the following section we compare SIPCoin and LP-SIP in more detail.

8 Comparison with previous work

We compare in depth LP-SIP with SIPCoin proposal since from previous study this is the only proposal that could be considered for making real-time micropayments for access to multimedia services based on SIP.

We analyze both proposals for the number of messages exchanged between the different parties (*C* - Client, *V* - Vendor, *B* - Broker or Payment provider) during the payment phase and for the different cryptographic operations involved to make the payment. This information is shown in Table 2.

For comparison we have considered two cases. First, when a client makes a payment to access the whole session. Thus, the client makes the payment, the session is established, the service is provided and once the service is finished, the session ends. This case is named *basic payment* and appears in Table 2 as row *A*. Second, a client makes an initial payment for a part of the session; the session starts and after some time/amount of data an additional payment is requested to continue with the session. If the user decides to continue with the session, the client makes a new payment and the session continues for some additional time/amount of data. After this, a new payment could be requested or the session could finish. This case is named *session with additional payments* and appears in row *B*. For simplicity, in the comparison we suppose that only one additional payment is requested before finishing the session.

As shown in Table 2 there are two columns for LP-SIP. LP-SIP$_1$ is the case when the client contacts a vendor for the first time in a day (this case is also applicable if the client has also previously used the e-coin in the same day with another vendor). LP-SIP$_2$ is the case when the client has already made a payment with that vendor using the same e-coin and without having used it with another vendor.

In Table 2 we can also see that for SIPCoin the number of hash operations is expressed according to *N*, *X*, and *I*. This is because the payment in SIPCoin is made from a hash chain. When the client starts a transaction, the broker generates a hash chain of SIPCoins. Let us suppose the hash chain has a value of 1 €and that each hash value (SIPCoin) in the chain has a value of 0.1 €. In this

Table 2 Comparison based on the number of messages and cryptographic operations

		SIPCoin			LP-SIP$_1$			LP-SIP$_2$		
		C	**V**	**B**	**C**	**V**	**B**	**C**	**V**	**B**
	Messages sent	5	4	4	4	6	2	4	4	
	Message received	5	4	4	4	5	2	4	3	
A	Symmetric encryption				1	3	4	1	1	
	HMAC				2	3	5	2	1	
	Hash	X+N	X	N						
	Messages sent	7	6		6	8		6	6	
	Message received	7	6		6	7	6	5		
B	Symmetric encryption				2	4		2	2	
	HMAC				4	4		4	2	
	Hash	N+X+(I-X)	X+(I-X)	N						

situation, the length of the chain is 10 (N) and this is the number of initial hashes the user has to make to verify the chain received. Later, for the initial payment, let us suppose that it is 0.3 €, the user would calculate the hash from the root value that he received for the chain. In this case the number of hashes would be 3 (X). Later, let us suppose an additional payment of 0.2 €is required, then the total number of hashes would be 5 (I where $I \leq N$) and for this additional payment the number of hashes performed are 2 (I - X).

From this comparison we can draw several conclusions. First, LP-SIP reduces the participation of a third party like a broker or payment provider. It is also important to point out that in SIPCoin the broker has to participate in all transactions (only for the first payment) unlike LP-SIP where the participation of the broker is only required the first time a user makes a payment with the broker during the day or when the user uses the same e-coin with different vendors. If the e-coin is used several times with the same vendor, broker participation is only required the first time. Thus, payments can be made more efficiently and the broker does not become a bottleneck on account of multiple transactions from many vendors and clients.

Second, the number of symmetric operations needed to guarantee authentication, confidentiality and integrity in LP-SIP is reduced. SIPCoin proposes them as an additional feature by means S/MIME, which supposes the use of asymmetric cryptography, or SSL/TLS that requires the use of asymmetric and symmetric cryptography with the exchange of additional messages (at least five messages to establish the channel and, if client authentication is established, two asymmetric signatures and one asymmetric encryption are required). But if these mechanisms are not used, a vendor could eavesdrop on the communication, obtain the root value to generate the hash chain and, in this way could calculate additional e-coins that he will be paid. As this feature is not an integral part of SIP these operations have not been included in the table but we should take into account their cost (in messages and cryptographic operations as mentioned above) in the comparison. Thus, we can assert that we support authentication, confidentiality, and integrity efficiently.

Third, the number of operations in LP-SIP is independent of the amount to pay, unlike SIPCoin, which depends on the value of the e-coins that are part of the hash chain. In SIPCoin, all the e-coins of a hash chain have the same value. Therefore, the divisibility is more difficult to achieve (as a solution to this problem the value of the e-coins could be very low, e.g., 0.01 but this would suppose many hash operations to make a payment of 1 €) unlike our protocol, where we can specify the exact amount to pay.

Fourth, particularly for vendors, the support of our protocol is efficient since the number of symmetric operations is reduced (in the worst case, the number of symmetric operations is 8), unlike SIPCoin, as we have just mentioned, which depends on the value of e-coins of the hash chain and does not offer additional features that we support like authentication, integrity and confidentiality of payment information, and its provision with S/MIME or SSL/TLS would be more costly. Thus, in LP-SIP, vendors could also support many different transactions with many different clients at the same time.

Finally, from the analysis of the number of messages and the operations to perform we can conclude LP-SIP offers divisibility, better security properties (including them efficiently) and reduced participation of the broker compared to SIPCoin, which makes it suitable for making real-time payments.

9 Conclusions and future work

SIP is one of the most important protocols to establish (multimedia) sessions and its use has even spread to mobile communications. Due to this, the extension of SIP to support payments is considered as a very interesting idea to charge users for the real-time services that a services provider offers. In fact, several proposals have appeared to support payments on SIP, such as SIMPA or SIPCoin. However, some of these proposals are not suitable for making payment for access to low-value services that have to be paid in real-time.

From these proposals, SIPCoin could be considered suitable for the above purpose. However, this proposal does not take into account as an integral part of the protocol properties such as authentication, confidentiality and integrity of payment information. For this purpose, SIP security mechanisms such as S/MIME, SSL/TLS or SIPS URI could be used. However, this increases the use of asymmetric cryptography during the payment phase, which is not suitable for very efficient payments.

As a response to the need for a payment protocol for the access to real-time services in SIP that considers the authentication, confidentiality and integrity of payment information, we have proposed LP-SIP. This new protocol is based on SIP extensibility mechanisms, which can facilitate its adoption. Furthermore, our proposal also reduces the participation of a third party such as a broker or a payment provider, which makes it better for making micropayments for access to real-time services.

Future work will focus on studying whether it is possible to reduce the participation of the broker even more or whether we can design more lightweight primitives for the operations performed in the protocol.

Acknowledgements
We would like to thank the anonymous reviewers for their comments and suggestions. They have significantly contributed to improve the quality of this article. We also want to thank Rafael Marín for his valuable comments on some issues of this article. This study has been partially funded by the following projects: "Seneca Foundation for Excellent Group in the Region 04552/GERM/06", TSI-020400-2008-101, CICYT TIN2008-06441-C02-02 and "Secure Management of Information across multiple Stakeholders (SEMIRAMIS)" CIP-ICT PSP-2009-3 250453, within the EC Seven Framework Programme (FP7).

Competing interests
The authors declare that they have no competing interests.

Received: 20 August 2011 Accepted: 4 May 2012 Published: 4 May 2012

References

1. J Rosenberg, H Schulzrinne, G Camarillo, A Johnston, J Peterson, R Sparks, M Handley, E Schooler, SIP: Session initiation protocol. *RFC 3261*. Internet Engineering Task Force (2002)
2. C Rigney, S Willens, A Rubens, W Simpson, Remote Authentication Dial In User Service (RADIUS). *RFC 2865 (Draft Standard)*. Internet Engineering Task Force (2000)
3. P Calhoun, J Loughney, E Guttman, G Zorn, J Arkko, Diameter base protocol. *RFC 3588 (Proposed Standard)*. Internet Engineering Task Force (2003)
4. A Tsakountakis, G Kambourakis, S Gritzalis, SIPA: generic and secure accounting for SIP. Security and Communication Networks (2010), http://onlinelibrary.wiley.com/doi/10.1002/sec.387/abstract
5. J Fischl, H Tschofenig, Making SIP make cents. Queue. **5**(2), 42–49 (2007). doi:10.1145/1229899.1229911
6. G Zhang, F Cheng, C Meinel, SIMPA: a SIP-based mobile payment architecture, in *Proceedings of the Seventh IEEE/ACIS International Conference on Computer and Information Science (ICIS 2008)*, Portland, Oregon, USA: IEEE, **1**, 287–292 (2008)
7. J Hao, J Zou, Y Dai, A real-time payment scheme for SIP service based on hash chain, in *IEEE International Conference on E-Business Engineering*, Xi'An, China: IEEE, **1**, 279–286 (2008)
8. S Fan, Z He, Y Zhang, H Zhang, R Su, An m-business model based on session initiation protocol, in *2008 International Symposium on Electronic Commerce and Security*, Guangzhou, China, 250–253 (2008)
9. G Zhang, F Cheng, C Meinel, Towards secure mobile payment based on SIP, in *Proceedings of the 15th Annual IEEE International Conference and Workshop on the Engineering of Computer Based Systems (ECBS 2008)*, Belfast, UK: IEEE, 96–104 (2008)
10. A Ahmed, S Khattab, K Mostafa, S El-Gamal, Comparison of online charging mechanisms for SIP services. Int J Electric Comput Sci IJECS-IJENS. **10**(2), 60–67 (2010)
11. KJ Lin, E-commerce technology: back to a prominent future. IEEE Internet Comput. **12**, 60–65 (2008)
12. C Jennings, J Fischl, H Tschofenig, G Jun, Payment for services in session initiation protocol (SIP). Internet-Draft draft-jennings-sipping-pay-06, IETF Secretariat (2007)
13. I Papaefstathiou, C Manifavas, Evaluation of micropayment transaction costs. J Electron Commerce Res. **5**, 99–113 (2004)
14. S Kardan, M Shajari, A lightweight buyer's trust model for micropayment systems. WSEAS Trans Inf Sci Appl. **5**(7), 1170–1179 (2008)
15. H Wang, J Ma, J Sun, Micro-payment protocol based on multiple hash chains, in *International Symposium on Electronic Commerce and Security*, Nanchang, China: IEEE, **1**, 71–74 (2009)
16. MS Javan, M Shajari, Flash payment: payment using flash disks, in *Proceedings of the 11th International Conference on Electronic Commerce*, ICEC'09, Taipei, Taiwan: ACM, 346–351 (2009)
17. D O'Mahony, MA Peirce, H Tewari, O Donal, *Electronic Payment Systems for E-Commerce*, 2nd edn. (Artech House Publishers, Norwood, MA, 2001)
18. MH Sherif, *Protocols for Secure Electronic Commerce*, 2nd edn. (CRC, Boca Raton, London New York Washington, D.C, 2003)
19. M Lesk, Micropayments: an idea whose time has passed twice?. IEEE Secur Privacy. **2**, 61–63 (2004)
20. R Párhonyi, L Nieuwenhuis, A Pras, Second generation micropayment systems: lessons learned, in *Challenges of Expanding Internet: E-Commerce, E-Business, and E-Government, IFIP International Federation for Information Processing*, vol. 189. (Springer Boston, 2005), pp. 345–359. doi:10.1007/0-387-29773-1_23
21. H Sinnreich, AB Johnston, *Internet Communications Using SIP: Delivering VoIP and Multimedia Services with Session Initiation Protocol*, 2nd edn. (Wiley, Indianapolis, Indiana, 2006)
22. J Rosenberg, H Schulzrinne, An offer/answer model with the session description protocol (SDP). RFC 3264, RFC Editor (2002)
23. J Rosenberg, H Schulzrinne, Reliability of provisional responses in the session initiation protocol (SIP). RFC 3262, RFC Editor (2002)
24. M Handley, V Jacobson, C Perkins, SDP: Session Description Protocol. RFC 4566, RFC Editor (1998)
25. A Ruiz-Martínez, JA Sánchez-Laguna, AF Gómez-Skarmeta, *SIP extensions to support (micro) payments*, Niagara Falls, Ontario, Canada, IEEE, **1**, 289-296 (2007)
26. J Domingo-Ferrer, A Martínez-Ballesté, STREAMOBILE: pay-per-view video streaming to mobile devices over the Internet, in *Proceedings of the 13th International Workshop on Database and Expert Systems Applications*, Aix-en-Provence, France, IEEE, **1**, 418–422 (2002)
27. A Ruiz-Martínez, Ó Cánovas, AF Gómez-Skarmeta, smartcard-based e-coin for electronic payments on the (mobile) Internet, in *Proceedings of The Third International Conference on Signal-image Technology & Internet-based Systems (SITIS'2007)*, Shangai, China, IEEE, 361–368 (2007)
28. P Overell, D Crocker, Augmented BNF for Syntax Specifications: ABNF. RFC 5234, RFC Editor (2008)
29. ISO: ISO 8601:1988. Data elements and interchange formats–Information interchange–Representation of dates and times. International Organization for Standardization (1988)
30. ISO: ISO 4217:1995: Codes for the representation of currencies and funds. International Organization for Standardization (1995)
31. B Schneier, Applied cryptography (2nd ed.): protocols, algorithms, and source code in C, (John Wiley & Sons, Inc., New York, 1995)
32. M Abadi, R Needham, Prudent engineering practice for cryptographic protocols. IEEE Trans Softw Eng. **22**, 6–15 (1996). doi:10.1109/32.481513
33. S Gürgens, C Rudolph, H Vogt, On the security of fair non-repudiation protocols. Int J Inf Secur. **4**(4), 253–262 (2005). doi:10.1007/s10207-004-0063-7
34. H Krawczyk, R Canetti, M Bellare, HMAC: keyed-hashing for message authentication. RFC 2104, RFC Editor (1997)
35. National Institute of Standards and Technology, *Secure Hash Standard*, (National Institute of Standards and Technology, Washington, 2002)
36. A Herzberg, Micropayments, in *Payment technologies for E-commerce*, 1st edn. (Springer, Heildelberg, 2003), p. 344
37. J Kytöjoki, V Kärpijoki, Micropayments–Requirements and Solutions, http://users.tkk.fi/~vkarpijo/netsec99/
38. SA Ahson, M Ilyas, *SIP Handbook: Services, Technologies, and Security of Session Initiation Protocol*, (CRC Press, Boca Raton, London New York Washington, D.C., 2008)
39. D Sisalem, J Floroiu, J Kuthan, U Abend, PH Schulzrinne, *SIP Security*, (Wiley, West Sussex, 2009)
40. G Karopoulos, G Kambourakis, S Gritzalis, PrivaSIP: ad-hoc identity privacy in SIP. Comput Stand Interf. **33**(3), 301–314 (2011). doi:10.1016/j.csi.2010.07.002
41. J Peterson, A Privacy Mechanism for the Session Initiation Protocol (SIP), *RFC 3323 (Proposed Standard)*, Internet Engineering Task Force, (2002)
42. G Karopoulos, G Kambourakis, S Gritzalis, Caller identity privacy in SIP heterogeneous realms: a practical solution, Marakkech, Morocco, IEEE. 37–43 (2008)
43. G Karopoulos, G Kambourakis, S Gritzalis, E Konstantinou, A framework for identity privacy in SIP. J Netw Comput Appl. **33**, 16–28 (2010). doi:10.1016/j.jnca.2009.07.004
44. Avispa project: Automated validation of internet security protocols and applications (AVISPA), http://www.avispa-project.org
45. A Armando, D Basin, Y Boichut, Y Chevalier, L Compagna, J Cuellar, P Hankes Drielsma, PC Heám, J Mantovani, S Mödersheim, D von Oheimb, M Rusinowitch, J Santiago, M Turuani, L Viganò, L Vigneron, The AVISPA tool for the automated validation of Internet security protocols and applications, in *Proceedings of the 17th International Conference on Computer Aided Verification (CAV'05), LNCS*, vol. 3576, ed. by K Etessami, SK Rajamani (Scotland, UK, Springer, 2005), pp. 281–285
46. Y Chevalier, L Compagna, J Cuellar, PH Drielsma, J Mantovani, S Mödersheim, L Vigneron, A high level protocol specification language for industrial security-sensitive protocols, in *Proceedings of Workshop on*

Specification and Automated Processing of Security Requirements (SAPS), 193–205 (2004)
47. D Basin, S Mödersheim, L Viganò, An on-the-fly model-checker for security protocol analysis, in *Computer Security–ESORICS 2003*, vol. 2808, ed. by E Snekkenes, D Gollmann (Springer, Berlin Heidelberg, 2003), pp. 253–270. doi:10.1007/978-3-540-39650-5_15
48. M Turuani, Sécurité des Protocoles Cryptographiques: Décidabilité et Complexité, *PhD thesis*, (Université Henri Poincaré, Nancy, 2003)
49. A Armando, L Compagna, SATMC: a SAT-based model checker for security protocols, in *Logics in Artificial Intelligence*, vol. 3229, ed. by JJ Alferes, J Leite (Springer, Berlin Heidelberg, 2004), pp. 730–733. doi:10.1007/978-3-540-30227-8_68
50. Y Boichut, P Heam, D Basin, Improvements on the Genet and Klay technique to automatically verify security protocols, in *Automated Verification of Infinite States Systems AVIS'04 (WS ETAPS'04)* (2004)
51. A Ruiz-Martínez, CI Marín-López, LP-SIP. HSPL Specification. (2012), http://ants.inf.um.es/~arm/lp-sip-avispa.hspl
52. SM Shedid, M El-Hennawy, M Kouta, Modified SET protocol for mobile payment: an empirical analysis. IJCSNS Int J Comput Sci Netw Secur. **10**(7), 289–295 (2010)
53. A Ruiz-Martínez, D Sánchez-Martínez, CI Marín-López, M Gil-Pérez, AF Gómez-Skarmeta, An advanced certificate validation service and architecture based on XKMS. Softw Pract Exper. **41**(3), 209–236 (2011). doi:10.1002/spe.996
54. National Institute of Standards and Technology, Federal Information Processing Standard Publication 197, Advanced Encryption Standard, (National Institute of Standards and Technology, 2001)
55. R Housley, Cryptographic Message Syntax (CMS). RFC 3852, RFC Editor (2009)
56. S Turner, DRL Brown, Use of elliptic curve cryptography (ECC) algorithms in cryptographic message syntax (CMS). RFC 5753, RFC Editor (2010)

doi:10.1186/1687-1499-2012-161
Cite this article as: Ruiz-Martínez and Marín-López: **A lightweight payment scheme for real-time services based on SIP.** *EURASIP Journal on Wireless Communications and Networking* 2012 **2012**:161.