Privacy-Aware Accounting and Billing

Name SURNAME
[1]*Organisation, Address, City, Postcode, Country*
Tel: +countrycode localcode number, Fax: + countrycode localcode number, Email:

Abstract: This paper proposes a smart privacy-preserving payment method devised for a scenario of online services provisioning. Our solution is able to enhance an existing framework holding Authentication and Authorization functionalities (AA) by further adding the Accounting and Billing capabilities, without the need to disclose any sensitive data to the Service Provider e.g. neither the real user identity nor her bank account number. This is achieved through the establishment of a trust network including the Bank, Service Providers and a new category of entities, named *Accounting Server*, acting as a broker between the formers. Services are accounted and billed by the Accounting Server on the basis of anonymous credentials, assigned to the user by her Bank. These credentials ensure that the Bank will pay the Accounting Server - that in turn will pay the Service Provider - on behalf of the undisclosed user. No single entity within the trust network possesses all the information required to profile the user, hence her privacy is preserved unless these entities cooperate against her interests. Starting from a reference framework relying on pseudonym-based Authentication and on Authorization management via anonymous credentials, we succeed in designing an innovative privacy-oriented architecture for complete exploitation of paid services.

Keywords: Privacy, Service Provisioning, AAA, Anonymous Payment.

1. Introduction

Engineering anonymous payment techniques is a topic of paramount interest, still timely and significant, despite it has been a long time since research has started focusing on it, and today several solutions are available in literature [1,2,3,4,5,6,7,8]. Nevertheless all these solutions ensure anonymity of the buyer leaving some open issues. First, ensuring total anonymity and untraceability on transactions is not convenient in situations when abuses are detected and therefore it may be required to track the user who performed the payment, e.g. upon a legal authority mandate. Moreover these solutions do not care about verifying whether a user is allowed to execute a payment operation, that is they do not embed any mechanism to verify the possession of the required credentials; for example, a user could own enough credit to buy an adult movie but she should be also of age. Finally, all these payment systems are best suited for purchasing goods rather than usage-based services, since they do not employ any accounting mechanism i.e. they do not charge users on the basis of resource consumption measures. Typically these measures are accounted by employing standard solutions like RADIUS [9] or Diameter [10]. For example, 3GPP charging management specifications [11,12,13] have adopted the Diameter approach. However these solutions do not devise any kind of protection in order to hide user sensitive data towards SP.

In order to tackle these issues we envision a comprehensive framework for online service provisioning with enhanced privacy-aware Authentication, Authorization and Accounting functionalities (AAA). In particular, we aim to extend a basic privacy-oriented architecture with embedded Authentication and Authorization functionalities (AA) by including also our privacy-aware accounting and billing solutions, which do not require personal data disclosure (e.g. real user identity or bank account number) to the Service

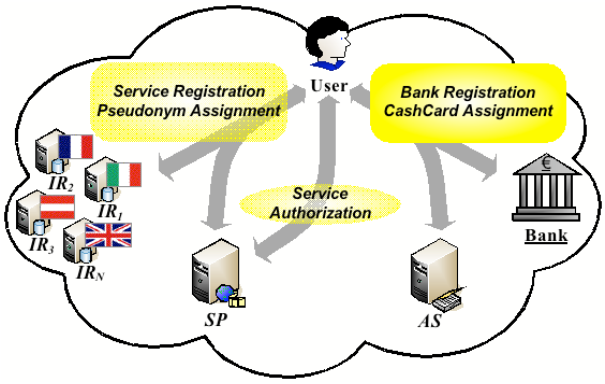

Figure 1: System architecture, offline environment setup phases

Provider. In this paper we show how to accomplish this task by enhancing the reference framework described in [14], deploying the involved entities within a PKI-like trust network.

The rest of the paper is organized as follows. Section 2 will give an overview of the whole architecture, introducing the new components with the respective role. Section 3 will describe the accounting and billing processes in details. Section 4 will discuss security issues, examples of usage and implementation choices. Finally, Section 5 will draw further consideration about potential exploitation opportunities and conclude the paper.

2. System architecture, environment setup

The complete system architecture is shown in Figure 1. Some entities have already been introduced in [14]. The framework is user-centric, meaning that user is free to decide her own way to exploit the architecture functionalities in order to enjoy her favourite services, which are supposed to be provided by the SP server. User will not log in to access service using her real identity, since the architecture is designed to manage pseudonyms. The latter are issued according to an offline setup process thoroughly described in [14] involving auxiliary entities, namely *Identity Repository*, represented by the IR_i servers inside the figure. In addition to the pseudonym, user will have to prove possession of the required credentials (e.g. being of age) needed for service authorization. Once again, the offline setup process for authorization credential issuing and verification, based on a new crypto tool named *Marked Blind Signature* (*MBS*), has been extensively discussed in [14]. Two new entities are now introduced inside the architecture, namely the Bank and the *Accounting Server* AS. The Bank clearly knows the real user identity since it manages her account. AS is a broker entity which allows user to pay services preventing direct communication between the Bank and SP. This is possible under the hypothesis of mutual trust between all the entities of the architecture. In the following we discuss the further offline steps to set the environment ready for accounting and billing.

2.1 Bank registration, CashCard assignment

After user has completed the offline SP registration and pseudonym assignment procedures, she will need her Bank to assign her some anonymous credentials, to be shown and verified at service login in order to start the billing process. For this reason, user negotiates with her bank a *grant* G to be included in a standard X.509v3 certificate. This certificate ensures that the bank will cover any service cost up to the value of G as long as it is valid. Therefore, G will be reasonably lower than the user total available credit, but we assume as hypothesis that it will be much greater than a generic transaction fee. In fact, user will be able to pay for services whose cost is not the same order of magnitude of grant G but is significantly lower. Denoting with c_i the cost of the i^{th} generic service transaction, we will

assume that if N is the expected number of transactions to be executed by user, the following condition must hold:

$$\sum_{i=1}^{N} c_i << G \qquad (1)$$

The value of G must be chosen with care (see Section 4 for discussion). According to this value, which is shown at service login, SP may decide not to start service provisioning, or even set some terms of usage based on a maximum limit for resource consumption. In order to do this SP must be able to verify in real-time the validity of G (which could have been revoked before the expiration time) by asking the Bank about its status. But this would call for a direct communication between SP and Bank, which is harmful for the user privacy; the Bank shall not profile user on the basis of her favourite services. It is then necessary to introduce the AS broker entity inside the architecture. As such, a trust network is required to be established, including the Bank, SP and AS. The task of AS will be to perform real-time checks about the grant status by asking the Bank on behalf of SP, as well as performing accounting and billing functionalities by interacting with SP. This can be accomplished by AS without learning anything about sensitive user data such as her real identity or her total credit amount, since they are never revealed to AS: for its purposes, it is enough to know that the Bank grants money on behalf of the user.

The new credentials are obtained with the following offline setup procedure:

Phase 1: User will start a registration procedure with her bank showing her real identity U and will request the issuance of a certificate $cert_G$. Inside the request the user includes her own suggested value of G (which of course has to be approved by the Bank, being eventually decremented at this time) and a public key e_G, generated on purpose, whose correspondent private key d_G will be kept stored by the user. Both G and e_G will be included inside $cert_G$. User also includes the public key e_{AS} of a self chosen AS, to be bound to $cert_G$ and that will identify the only broker to be paid by the Bank for the services exploited by User, as we will shown on Section 3. User also generates a further key pair, whose public key e_{CC} will be certified by AS later and must be advertised to the Bank. Therefore these steps can be summarised by the following handshake:

User → Bank	:	$\{U, G, e_G, e_{AS}, e_{CC}\}$	(*a*)
Bank → User	:	$\{cert_G\}$	(*b*)

Phase 2: After receiving $cert_G$, User will connect with her favourite AS, which has been selected in the previous phase. AS will include the value of G inside a new certificate, denoted as *CashCard* (*CC*). This new credential will be submitted to SP at service login. It is intended to identify the AS responsible for managing the accounting and billing procedures, and also to advertise the value of G which is covered by AS on behalf of an undisclosed Bank. In order to obtain the CashCard, user performs the following offline steps:

User → AS	:	$\{cert_G, e_{CC}\}$	(*a*)
AS	:	verify_signature($cert_G$)	(*b*)
AS ←→ User	:	challenge($cert_G$)	(*c*)
AS (←→ Bank)	:	policy_check($cert_G$)	(*d*)
AS → User	:	$\{CC\}$	(*e*)

In particular, User will send a request including $cert_G$ and the previously generated public key e_{CC}, which is known by the Bank to belong to User (step (*a*)). AS will first verify the signature on $cert_G$ (step (*b*)) and challenge User to prove she owns the private key d_G (step (*c*)). Step (*d*) further accounts for the possibility to introduce a policy check since, according to the value of G, AS may either immediately release the CashCard, or postpone

the release after the successful outcome of a query to the Bank, or simply reject the request. Finally, if all the policy checks have been passed successfully, AS will issue the requested CashCard *CC* (step (*e*)).

Before accessing a service user must obtain some anonymous authorization credentials to be submitted at service login along with her pseudonym *P*. In order to obtain such credentials, in [14] the MBS scheme is applied using the (private key of the) pseudonym *P* itself. In the extended architecture we prefer to use the (private key of) CashCard *CC*, for the hypothesis of non-transferability becomes stronger[1]. For the sake of simplicity, hereby we summarize the issuing process in three steps, for any technical description please refer to [14].

User → SP	:	$\{U, \underline{CC}\}$	(*a*)
SP ←→ User	:	marked_blind_signature($\underline{CC}$)	(*b*)
SP → User	:	$\{\underline{cred_{CC}}\}$	(*c*)

At step (*a*) User will contact SP showing up with her real identity *U* and a blind version of the CashCard $\underline{CC}$. At step (*b*) SP will perform the MBS on the blind CashCard, where User is requested to use the private key of the submitted certificate. Finally at step (*c*) a blind version of the credential $\underline{cred_{CC}}$ will be issued to User, who then will remove the blinding factor to get the plain credential $cred_{CC}$. Now User will be ready for service login.

3. Accounting and Billing[2]

User logs in by submitting a $\{P, CC, cred_{CC}\}$ triplet to the *SP*. This process is shown in Figure 2.a. In order to gain authorization User must prove to be the legitimate owner of all these credentials, i.e. to possess the corresponding private keys. As regards the pseudonym *P* and the anonymous authorization credential $cred_{CC}$, SP itself will challenge the user. *P* is verified by asking User to sign a random number, whereas to verify $cred_{CC}$ a further expression must be checked by SP, according to the MBS algorithm discussed in [14]. In order to verify the CashCard ownership SP will act as a NAS and forward the *CC* to the AS. This way AS can challenge the user to have proof she is really requesting the service, thus preventing a malicious SP from abusing the *CC*. SP will also send a *GCheck* message to AS, in order to keep the information about the grant *G* up-to-date. AS will forward the message to the Bank and upon a successful *CC* validation will forward the *GCheck_reply* message (boolean: *G* is either valid or revoked) back to SP. If all these checks are successful, User is finally authorized to start service selection.

According to the selected service, the extended architecture supports two different payment methods. The first one is recommended for purchasing goods and it is similar to using a credit card since it requires a single transaction to execute the payment. For example, in order to buy an mp3 online it is required a *one-time billing* before receiving the authorization code for download. The second method relies instead on *real-time accounting* since it is based on continuous updates about resource consumption sent by SP to AS, hence it is recommended for usage-based services provisioning. In the following we discuss both.

3.1 One-time billing

A time diagram for this method is shown in Figure 2.b. After service selection by User, SP performs cost allocation and contacts AS to i) check about grant status, and ii) preparing an

---

[1] Sharing a private key with a friend means transferring to him/her all the credentials associated with that key. Therefore, since a user is not encouraged in sharing a pseudonym private key, which is bound to her identity, we believe she is even more adverse in sharing her CashCard private key, which is bound to her money.

[2] According to [15], accounting is defined as "the collection of resource consumption data for the purposes of capacity and trend analysis, cost allocation, auditing, and billing (…)", while billing is "The act of preparing an invoice".

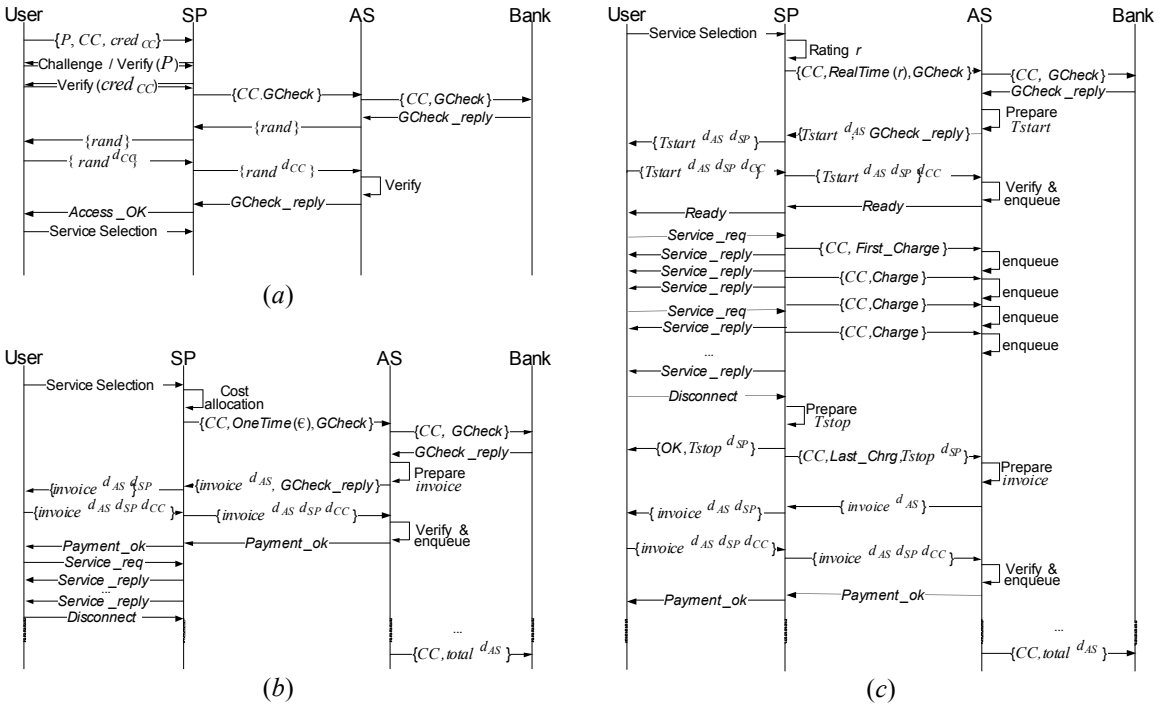

Figure 2: Service Provisioning. (a) authorization phase, (b) one-time billing, (c) real-time accounting

invoice for this payment method. AS will prepare the invoice, containing the operation details (being not necessarily specific information about the item purchased, but at least some essential information like timestamp, SP public key, cost) and will sign and send it to User through SP, that will sign it in turn before forwarding. User then signs the invoice with her CashCard private key and returns it back to SP who forwards it again to AS. AS will verify the signature and send a confirmation to SP that the payment has been successful, so finally SP starts the service provisioning.

As noticed before, this payment scheme resembles the traditional way to pay using a credit card since User simply performs a one-pass procedure to sign the invoice. All the signed invoices are then collected by AS and registered in a queue of pending transactions for the specific CashCard *CC*. Periodically, e.g. every month, this queue is served by AS by calculating a grand total and presenting it to the Bank, who will pay the requested amount to AS by charging the user account. We do not aim to define policies regulating the periodicity when this out-of-band communication between the AS and the Bank is performed, nor policies about the final transaction between AS and SP, for their specification goes out of the scope of this paper. Our hypothesis is that AS sets up some agreement with SP in order to pay it for the accounted services via offline transactions, that will be transparent to the Bank and to the user. For further discussion about business exploitation, see Section 4.

User bank statement will contain an entry regarding the total amount paid to AS. User will be able to verify this sum since she has collected all the signed invoices sent to AS. A more comfortable way to check the details about the entry could be for example logging into the AS website, where the list of all the invoices enqueued by AS will be available.

3.2 Real-time accounting

Figure 2.c shows a time diagram for this payment method, which implements a real-time accounting procedure. When User selects a service whose cost is variable depending on usage consumption, this method must be employed. User will be asked to sign with her CashCard twice, the first time when service starts and the second time when it stops. This

will provide warranties for either the user itself and the SP, as it will be discussed in the following.

After User has selected a usage-based service, SP performs rating and contacts AS, sending the usual grant check message and ordering to start the real-time accounting process, with additional indications about the fare rate to be applied. AS waits for the grant check reply from the Bank and prepares a *Tstart* invoice. This invoice simply contains an indication that a real-time accounting process has been requested, including the service fare and the timestamp. After signing it, AS sends it to SP along with the grant check reply. SP will sign in turn the *Tstart* invoice and forward it to the user, who will be asked to sign it using her CashCard *CC*. The signed invoice is then forwarded back to AS by SP. At this point, AS verifies the signature on the invoice and enqueues it, being now ready to collect the accounting messages. User is then notified that service accounting has started, so the provisioning begins. SP will send accounting messages in real-time, containing the charge that is computed on usage-basis (either time or traffic consumption). AS will keep on cumulating the charges, calculating and updating the service cost as long as it grows. Depending on the duration or on the amount of resource consumption, AS may generate a signed checkpoint invoice, containing the partial cumulated cost, to be signed by both SP and the user (this is not shown in the time diagram). When User will decide to stop service she will send a disconnect message. SP will immediately confirm the choice and send a signed *Tstop* invoice to the user, containing the timestamp and the total resource consumption at the end of the provisioning. Meanwhile, SP sends to AS the last accounting message with the same *Tstop* invoice, in order to let it compute the final service cost and prepare a final *invoice*, having the same format of a one-time bill. Indeed, from now on everything is similar to what we discussed for the one-time billing method. The *invoice* will be sent to the user with the usual forwarding procedure. After AS receives back the signed *invoice*, SP and the user will be notified that the whole payment process has been successful. All the further considerations about invoice enqueueing and the total amount to be paid periodically by the Bank to AS still hold.

The proposed accounting method is fault-tolerant. In the event that User connection goes down during service provisioning, for example if she experiences some failure that physically disconnects her, the Service Provider will immediately prepare the signed *Tstop* invoice upon receiving a connection timeout and perform the disconnection steps as if they were requested by User. AS will prepare the final *invoice* and wait for next User access to have it duly signed, before starting any other billing procedure.

Notice that no one inside the system is encouraged in misbehaving since the invoice collection process is intended for auditing. User cannot misbehave any way because her Bank would pursue her. AS could give proof of the user transactions by showing all the collected signed *invoices*. On the other hand, service cost included in the final *invoice* must be consistent with the usage computed form the *Tstart* and the *Tstop* invoices. Both the invoices are double signed by AS and SP. AS could certify greater usage consumption only by cooperating with SP against the interests of the user.

4. Discussion

Let us discuss about the grant value *G*. We remarked that it must be chosen reasonably lower than the user total credit, and that it is kept constant within the validity period of the $cert_G$ certificate, unless the Bank revokes it before the expiration time. This can happen if the user credit decreases too much and becomes the same order of magnitude of (or even equal to) the grant *G*. For example, if user total credit is $C = 1000€$, the Bank might decide to certify a grant value of $G = 100€$ and keeping it valid as long as *C* stays above 200€ (these values are totally arbitrary), otherwise revoking it. Therefore the user is encouraged

not to let her credit C decrease too much, otherwise $cert_G$ will be revoked and consequently all the credentials issued on the basis of it (e.g. the CashCard) will be revoked as well.

On the other hand, since AS registers all the invoices that have been signed by the user with her CashCard, for each new payment request it can check if the grant value has been exceeded and eventually deny the request. For example, suppose G = 100€ and user has already purchased with her CashCard 4 DVD in two weeks, each one costing 19.99€, hence she spent 79.96€ out of 100€ available in one month (suppose the Bank pays AS every month). If the third week she wants to buy a new DVD at 25€, SP would communicate the price to AS, who would verify G to be exceeded, so SP would be notified that the operation could not be executed. If the requested service required payment with real-time accounting, it would make no difference: service provision would start and go on as long as the cumulative cost calculated by AS does not exceed the grant value G. If this happens, AS would order SP to quit service provisioning immediately.

Notice that a user can request more than a single grant certificate to her Bank, in order to obtain several CashCards from different ASes. For example, suppose user owns two grants G_1 and G_2, associated to two CashCards issued respectively by AS_1 and AS_2. Since the two CashCards can be used in parallel, the revocation threshold r cannot be set lower than the sum of the two grants. If C = 1000€, G_1 = 100€, G_2 = 150€, they both will be valid as long as $C > r$, where $r >$ 250€. If threshold is exceeded, the Bank will decide whether to revoke both grants or just one.

Now we discuss some technical aspects related to security. Our proposed solution is built on top of a network of mutually trusted entities, which is similar to a PKI. Indeed, we assume all the the certificates exchanged inside the framework are conform to the standard X.509v3 format. We further assume that all communication between these entities rely on the functionalities offered by underlying protocols protecting data exchange. In particular, these protocols must provide i) data encryption, ii) peer authentication, iii) message authentication and iv) proxy features. All the necessary mechanisms for data encryption, peer authentication and message authentication are already provided by standard protocols such as TLS and IPSec, preventing an attacker from exploiting MITM and profile a User (e.g. by eavesdropping the input/output traffic generated by an AS, the input/output CashCard mapping would be disclosed). Employing an anonymous proxy or a whole anonymization network (e.g.Tor) would ensure unlinkability of different certificates to the same source address, and hence to the same user. In order to protect the user from being profiled by AS the or by the Bank it is sufficient that the two entities do not cooperate against the user. Moreover, in case user possesses different CashCard certificates from independent AS, in order to draw her complete profile all the selected ASes should cooperate to threaten the user. In order to prevent identity stealing attacks, User generates on her own a new public/private key pair for each new requested certificate, so the private key never leaves the User terminal. Proper storage of the private keys is clearly mandatory and delegated to off-the-shelf software key-rings. A further possible attack to counteract is certificate forgery, including i) access credential forgery/highjacking, and ii) CashCard or $cert_G$ forgery. In the first case an unauthorized attacker nevertheless aims at gaining service access, in the latter case the attacker aims at unaccountability for the exploited services. In order to reduce the probability that a certificate can be forged, we assume to employ 2048 bit RSA keys. Consequently, it would be computationally infeasible to forge a certificate without the possession of either the AS or the Bank private key.

Finally, it is manifest that adding billing functionalities to the framework means adding an intrinsic business model. The role of AS is similar to a credit card company, therefore AS is likely to ask for commissions to be paid for each transaction. For example, SP may be paid immediately by AS after the latter has received and enqueued a signed invoice. This is actually the real-world case for standard credit cards, so AS usually charges users with an

additional fee at the time of being paid by the Bank. Managing AS fares is a well known business oriented profitable activity.

5. Conclusion and further work

This paper proposed a novel privacy-aware accounting and billing scheme for on-demand paid services. We have showed how to achieve a comprehensive privacy-oriented architecture, whose components form a PKI-like trusted network, that allows an user to select, enjoy and pay her favourite services while protecting her personal data from unnecessary disclosure. The anonymous credentials management ensures that no single entity inside the framework is entrusted with all the information needed to profile the user; on the other hand, such a privacy enhancement shall not have a negative impact on service exploitation.

At the time of writing an early implementation of the basic credential management system has been developed in a Linux environment as a proof-of-concept, within the framework of the IST-DISCREET EU Project. We have been merging different technologies, e.g. openssl libraries for RSA signatures and certificate management, MySQL databases and a Java GUI wrapping the C code on the User side. We aim to implement the accounting protocol using attribute-value pair messages in order to develop a natural extension of the Diameter DCCA [16], since most of our communications can be implemented via already existing CCR/CCA messages. For that purpose, we are considering an integration with the OpenBloX project [17].

References

[1] D. Chaum, A. Fiat, M. Naor. Untraceable Electronic Cash. In Proceedings of CRYPTO 1988, p.319-327

[2] G. Medvinsky, B. Clifford Neuman. Netcash: A design for practical electronic currency on the internet. In Proceedings of the First ACM Conference on Computer and Communications Security, Nov. 1993

[3] B. Cox, J.D. Tygar, and M. Sirbu. Netbill Security and Transaction Protocol. In The First USENIX Workshop on Electronic Commerce, pages 77–88, July 1995.

[4] M. Peirce and D. O'Mahony. Scaleable, Secure Cash Payment for WWW Resources with the PayMe Protocol Set. In Fourth International Conference on the World-Wide Web, MIT, Boston, December 1995.

[5] H. Bürk , A.. Pfitzmann, Digital payment systems enabling security and unobservability, Computers and Security, v.8 n.5, p.399-416, August 1989

[6] D. Chaum. Privacy Protected Payments Unconditional Payer and/or Payee untraceability. Smart Card 2000, Proceedings. North Holland, 1989.

[7] S. A. Brands. Untraceable off-line cash in wallets with observers. In Crypto'93 SpringerVerlag, LNCS 773 pp. 302-318.

[8] W3C Micropayment Working Group homepage - http://www.w3.org/TR/Micropayment-Markup

[9] C. Rigney, S. Willens, A. Rubens, W. Simpson. Remote Authentication Dial In User Service (RADIUS), RFC 2865 , June 2000.

[10] P. Calhoun, J. Loughney, E. Guttman, G. Zorn, J. Arkko. Diameter Base Protocol., RFC 3588, Sept. 2003

[11] 3GPP TS 32.225 version 5.11.0, 3rd Generation Partnership Project; Technical Specification Group Services and System Aspects; Telecommunication management; Charging management; Charging data description for the IP Multimedia Subsystem (Release 5).

[12] 3GPP. TS 32.299 version 8.1.0, 3rd Generation Partnership Project; Technical Specification Group Service and System Aspects; Telecommunication management; Charging management; Diameter charging applications (Release 8).

[13] 3GPP. TS 22.115 version 8.2.0, 3rd Generation Partnership Project; Technical Specification Group Service and System Aspects; Service aspects; Charging and billing (Release 8).

[14] G. Bianchi, M. Bonola, V. Falletta, F.S. Proto, S. Teofili (2007). The SPARTA Pseudonym and Authorization System. ESORICS WORKSHOP - 3rd International Workshop on Security and Trust Management (STM 07), September 27 2007, Dresden, Germany.

[15] B. Aboba, J. Arkko, D. Harrington. Introduction to Accounting Management. RFC 2975, October 2000.

[16] H. Hakala, L. Mattila, J-P Koskinen, M. Stura, J. Loughney. Diameter credit-control application. RFC 4006, August 2005.

[17] OpenBloX project homepage - http://sourceforge.net/projects/openblox