P-DIBS: Pseudonymised DIstributed Billing System for Improved Privacy Protection

Vincenzo Falletta, Simone Teofili, Saverio Proto, Giuseppe Bianchi
CNIT / Università di Roma "Tor Vergata", Italy
{vincenzo.falletta,simone.teofili,giuseppe.bianchi}@uniroma2.it; proto@ing.uniroma2.it

***Abstract*—The deployment of payment systems protective of the customer privacy is an hard challenge. Accountability and payment seem to require a direct link to the customer credentials (e.g. his credit card number or bank account), this exposes the user to be profiled on his habits. Static and uniquely identified mappings to user credentials, hold by a trusted third party, may vanish all the parallel anonymization / pseudonymisation efforts done to avoid disclosure of the user identity to the provider of the service. This paper proposes P-DIBS (*Pseudonymised Distributed BIlling System*), a billing framework devised to protect user privacy. P-DIBS is developed as an extension of a previously proposed pseudonymization mechanism. It relies on an intermediate brokerage entity, referred to as "Accounting Server", operating between the Bank and the Service Provider on behalf of the end user, yet having no knowledge neither about his real identity nor about his real account number. A fundamental novelty of the proposed approach is the possibility, through a distributed procedure involving mutual interaction across the various system components, to guarantee linkability upon improper user behavior (e.g. misuses) without requiring a single Trusted Third Party in the system to possess all the knowledge necessary to disclose the user.**

I. Introduction

Protection of the user privacy is a challenging issue. Anonymity and uncontrolled access, although technically feasible is not a proper approach, as it would be immediate, for intruders and malicious users, to abuse of such strong data protection measures for inappropriate (or even criminal) purposes. Indeed, regulatory provisions (e.g. Article 13 of the EU Directive 95/46/EC just to mention a well known one) explicitly set forth limits within which public interest prevails, including national and public security, prevention, investigation, detection and prosecution of criminal offences, protection of the data subject or of the rights and freedoms of others, etc.

On the other side, it is not reasonable to delegate the control to operators and service providers over the user data, as this can become a threat when the operators themselves (or their employers) make an improper usage of the personal data gathered while providing services. A possibile countermeasure would consist in involving third intermediary parties that provide uniquely assigned labels, namely *pseudonyms*, to mask the real user identity from the service provider. Advanced approaches do indeed exist (e.g. based on group signatures or Identity Escrow approaches [1], [2] although in all cases they ultimately rely on a single Trusted Third Party, such as the group/escrow verifier). Furthermore, identity pseudonyms are not sufficient. In fact, when payment services are involved, there is the need to rely on static/persistent data, such as credit card numbers or bank accounts, which uniquely identify the user exactly as his identity, and hence may become the labels upon which tracking and profiling activities carried out by the various parties involved in the service provision. True, also these data may be object of suitable pseudonymization. However, in general this approach does not solve the problem, but only shifts it: from the need to trust the ultimate service provider to the need to trust the intermediary broker.

Goal of this paper is to describe a comprehensive framework devised to provide both identity pseudonymisation and pseudonymised billing, without the need to rely on a single intermediary entity, but preserving the possibility to revert the various involved transactions and link them back to the real underlying user upon need (e.g. misuses). A key advantage of our framework, with respect of previous literature, is the simultaneous management of identity pseudonyms and payment. In fact, high quality previous works in the area of anonymous payments already exist. For example, the well known E-Cash mechanism by Chaum et al. [3] and the NetCash infrastructure [4] are examples of secure payment frameworks with anonymity features. Other proposals such as NetBill [5], PayMe [6], the approach presented in [7], and a number of privacy-oriented payment schemes under developement in the Internet community (e.g. Micropayment [8]) have significant strengths, but in some cases they circumvent the main problem, i.e. they do not solve the problem of maintaining anonymity towards a vendor whom we are already registered to. In other words, whether they theoretically work with an online shop scenario, still they are hardly exploitable in case the vendor is the service provider since pseudonymisation is not implemented any way.

Our approach is devised to challenge the situation in which a customer, duly registered to a private Service Provider (SP), needs to access a service directly offered by the SP itself, and which is subject to payment. In order to manage the user identity, we rely on the preliminary concepts earlier proposed in our IST-DISCREET project [9] for pseudonym assignment [10], and we significantly extend them to integrate accounting functionalities. The proposed mechanism, P-DIBS (*Pseudonymised DIstributed Billing System*), ensures an improved privacy management relying on the simple concept of having information distributed over multiple entities. That is, no single entity within the scenario knows everything about

the user, but many have to collude in order to disclose sensitive user data, thus linkability is also guaranteed. As an extended framework, in addition to the existing actors playing in [10] P-DIBS introduces a new entity, the *Accounting Server* (AS), to accomplish all the anonymous billing operations. The AS acts in real time as a broker between the SP and the Bank on behalf of end users, therefore they have the possibility to choose several pre-paid or post-paid services without actually disclosing to the Bank any information about what they are purchasing. Moreover, employing pseudonymisation users are protected from disclosing their real identity or their real account number even to AS, hence the system is conceived to be fully anonymous.

The rest of the paper is organised as follows [1] : Section II offers a brief survey on the pseudonym assignment procedure described in [10]. The anonymous accounting system is then shown in Section III, whereas Section IV gives technical detail about the implementation. Finally in Section V we draw conclusions.

We will use the following notation to denote cryptographic operations:

- (pk_y, sk_y): public and private key pair for entity y.
- $id(y)$: unique identifier associated to a public key pk_y.
- $cert_X(id(y), pk_y)$: certificate issued by entity X and containing both the identifier $id(y)$ and the related public key pk_y.
- $CHA(A; X)$: challenge handshake function performed by entity X towards the owner of public key specified in certificate A.
- $VER(A, X)$: verifying function performed by entity X to validate the signature of certificate A.
- $ENC(D, X, k)$: encryption function performed by entity X using key k on data D.

II. Pseudonym Assignment

This paper leverages on the pseudonym assigment approach proposed in [10], and significantly extends it in order to provide support for billing. For a complete comprehension of the P-DIBS operation, it is hence necessary to briefly review the basic pseudonym assignment approach.

The key ideas in [10] are i) to completely decouple the verification of the authorization credentials that the user presents to a service provider to receive a pseudonym, from the user identity, and ii) delegate the task of managing the linkability to the real user identity to administratively distinct entities. As a consequence, it is possible to rely on completely anonymous authorization mechanisms such as Ring Signatures [13] (which, unlike group signatures, do not provide any way to revoke the anonymity of an individual signature), and delegate the task of reverting/revoking an assigned pseudonym to a fully distributed operation involving two or more distinct and independent entities (and not to a single centralized entity such as a group or Identity Escrow verifier).

The framework proposed in [10] consists of separate entities, namely the Service Provider SP to whom the user is subscribed, and one or more independent entities denoted as *Identity Repositories*, IR. Neglecting some technical details, the sketch of the pseudonym assignment operation is the following:

0) When a user U initially subscribes to a Service Provider SP, he is provided with a certificate, hereafter referred as `user_cert`, signed by SP, and containing the user identity encrypted with a secret key belonging to SP, along with a relevant public key pk_U. As such, SP is the only entity capable of understanding whom a certificate belongs to.
1) Prior of accessing the service domain, the user generates a new key pair (pk_{TnIR}, sk_{TnIR}) and sends the new public key together with the user certificate to the IR Server. After having verified SP signature on the user certificate, IR validates that the user possesses the certificate secret key a through a challenge-response handshake. Finally, IR generates a new certificate (namely `TnIR_cert`) containing a unique identifier randomly generated by IR and associated to the pk_{TnIR} earlier submitted by the user. In summary:

   1.1) $U \rightarrow IR : \{\texttt{user\_cert}; pk_{TnIR}\}$
   1.2) $VER(\texttt{user\_cert}, IR)$
   1.3) $CHA(\texttt{user\_cert}, IR)$
   1.4) $U \leftarrow IR : \texttt{TnIR\_cert}$

   where
   $\texttt{TnIR\_cert} = cert_{IR}(id(TnIR), pk_{TnIR})$,
   $id(TnIR)$: unique identifier randomly generated by IR, associated to pk_{TnIR}.

   Note that, at the end of these operations, IR will keep track of the mapping between encrypted user identity and assigned token. Moreover, we remark that the public key included in `TnIR_cert` has been generated by the user himself who is not forced to reveal the corresponding private key. This mechanism ensures that IR is not able to forge a token certificate with the same public/private key pair of the user.
2) An almost identical procedure is performed between U and SP: after generating a new key pair (pk_{Ps}, sk_{Ps}), U transmits the token certificate `TnIR_cert` to SP together with the key pk_{Ps}, and receives back the final pseudonym certificate `pseud_cert`. The key difference is that, in addition to these operations, the user is further required to submit an additional anonymous credential: our current approach is to use a Ring Signature. Thus the verification of the user's credentials is guaranteed through Ring Signature validation. Note that is essential to guarantee that SP should be able to

[1] Due to space constraints, we decided not to include a dedicated section with a thorough security analysis of our system. In a nutshell, consider what follows: since we exploit the encryption and authentication functionalities offered by underlying security protocols such as IPsec [11] or TLS [12], our system is protected against well known MITM and/or rogue server attacks. On the other hand, for all the issues related to the management of keys and certificates, our system provides the same protection level (and consequently has the same vulnerabilities) of any ordinary PKI.

verify in first person the validity of its subscribed users: delegation of this process to external entities (e.g. what would occur if only verification of the IR certificate were performed) would not be acceptable for obvious reasons. Summarizing:

2.1) $U \rightarrow SP : \{\texttt{TnIR\_cert}; pk_{Ps}\}$
2.2) $VER(\texttt{TnIR\_cert}, SP)$
2.3) $CHA(\texttt{TnIR\_cert}, SP)$
2.4) $RnSn$ verification
2.5) $U \leftarrow SP : \texttt{pseud\_cert}$

where
$RnSn =$ anonymous credential (Ring Signature),
$\texttt{pseud\_cert} = cert_{SP}(id(Ps), pk_{Ps})$,
$id(Ps)$: unique identifier randomly generated by SP, associated to pk_{Ps}.

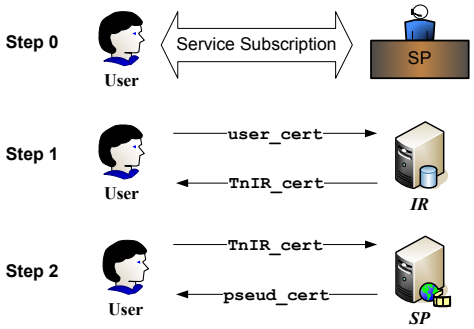

Fig. 1. Environment setup - Pseudonym assignment.

All these steps are summarized in Fig. 1. It is easy to see that our proposal is completely user driven. Indeed to obtain a pseudonym, the user is not forced to trust any specific Trusted Third Party, on the contrary he is free to choose among different IR servers whose public key are known to SP. Furthermore, neither user has to reveal to IR nor to SP his private keys that will be employed to prove that he is the legitimate owner of both the IR certificate and the pseudonym certificate. This increases the difficulty for an external or an insider attack (i.e. a sys admin managing one of the server) to create a fake IR or pseudonym certificate.

Once again we remark that the previous steps are part of the environment setup process, hence are performed offline and do not supply any additional delay to the service provisioning, which at this time has not started yet. Nevertheless, user is free to execute what described in step 3 immediately before requesting a service.

III. Accounting

A. P-DIBS accounting scheme

A new entity, denoted as the Accounting Server AS, is included in the system operations. This entity has the goal to mutually hide the SP from the user-associated financial provider, hereafter for convenience denoted as the Bank B. As shown in what follows, AS is not just an intermediary entity which possesses the knowledge of both SP and B information, but its operation is more subtle, as it is intended to minimize the information available to AS. In order to exploit AS as an intermediary, the user needs to perform the following three step set-up procedure:

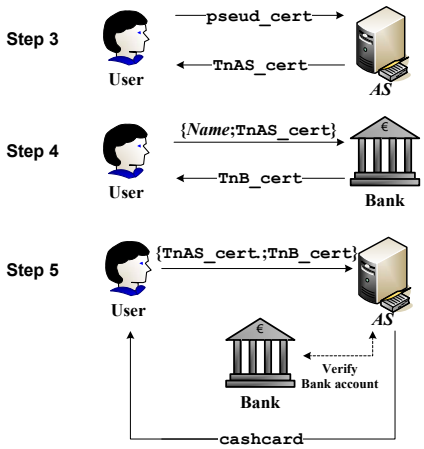

Fig. 2. Environment setup - Accounting.

3) User registers to AS via one of his pseudonyms certified by $\texttt{pseud\_cert}$, forging as usual a new key pair (pk_{TnAS}, sk_{TnAS}) and sending the public key together with the certificate to AS. Upon the success of the usual validation procedures, AS will sign and send an new certificate $\texttt{TnAS\_cert}$ having an analogue structure of the previously examined ones:

3.1) $U \rightarrow AS : \{\texttt{pseud\_cert}; pk_{TnAS}\}$
3.2) $VER(\texttt{pseud\_cert}, AS)$
3.3) $CHA(\texttt{pseud\_cert}, AS)$
3.4) $U \leftarrow AS : \texttt{TnAS\_cert}$

where
$\texttt{TnAS\_cert} = cert_{AS}(id(TnAS), pk_{TnAS})$,
$id(TnAS)$: unique identifier randomly generated by AS, associated to pk_{TnAS}.

4) Next step user announces to his Bank B that he has been registered to AS and asks B to generate an encrypted identifier of his own account number in order to let AS withdraw money from his personal account (or simply to check the balance) without disclosing the real number. This time no challenge is needed since user has to provide his name to the bank.

4.1) $U \rightarrow B : \{Name, \texttt{TnAS\_cert}, pk_{TnB}\}$
4.2) $U \leftarrow B : \texttt{TnB\_cert}$

where
$\texttt{TnB\_cert} = cert_B(id(TnB), pk_{TnB})$,
$id(TnB) = ENC(AccountNo., B, K_B)$: encrypted user account number, associated to pk_{TnB},
K_B : secret key belonging to B.

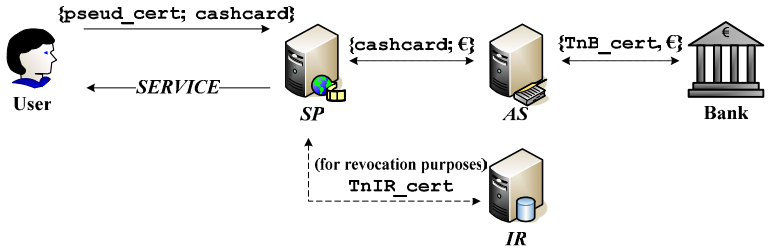

Fig. 3. Online service provisioning with pseudonymised billing.

5) Now user returns to AS with `TnAS_cert` and `TnB_cert` asking for a `cashcard` i.e. a cash certificate needed to prove he is able to purchase services. After the usual verification/challenge steps performed for both certificates, AS then contacts B to check the balance of the account addressed by $id(TnB)$, and upon a positive response from B finally releases the `cashcard`:

   5.1) $U \rightarrow AS$: {`TnAS_cert`, `TnB_cert`}
   5.2) VER(`TnAS_cert`, AS)
   5.3) CHA(`TnAS_cert`, AS)
   5.3) VER(`TnB_cert`, AS)
   5.4) CHA(`TnB_cert`, AS)
   5.5) $AS \leftrightarrow B$: verify balance of $id(TnB)$
   5.6) $U \leftarrow AS$: `cashcard`

   where
   `cashcard` $= cert_{AS}$(`TnAS_cert`,info),
   "info" includes several information depending on the type of payment (e.g. pre/post-paid method) agreed between user and AS: expiration date, maximum payment limit, notes or recommendations by AS are example of possible fields.

Once the `cashcard` has been received (we remark that this operation can be performed offline, and thus it does not affect the real-time performance), access to services takes place as shown in Figure 3. The user simply needs to access SP via notification of the previously assigned `pseud_cert` and the `cashcard`. Additional authorization credentials may be further considered to restrict the access to the service (a thorough discussion on how to manage this in an anonymous manner, e.g. through attribute certificates, is out of the goals of this work). While the payment solution depends on the specific payment method (pre/post paid, flat rate, etc), it is here important to note that AS has no information about the user, as it only handles the cash-card which is associated to the pseudonym. Conversely, the Bank has clearly knowledge of the specific user, but payments are submitted through the `TnB_cert`, which does not provide information on neither SP that requires the payment, nor the pseudonym under which the payment is accounted.

B. Mapping on existing solutions

Internet Services Providers (ISP) exploits their AAA architecture for gathering data regarding their network resources consumption and also for generating accounting records [14]. The IETF protocols Diameter [15] or RADIUS [16] are used in the AAA architecture for the transmission of charging information [17]. An ISP may apply a flat rate pricing model or a usage-based one, in that case the ISP incorporates a billing system to calculate the charges. While the pseudonymisation of the user's identity could be easily achieved by extending the Radius and Diameter protocols to allow the usage of pseudonyms instead of real user identities inside X.509 certificates[18], the pseudonomysed billing mechanism would require some changes, actually not significant, in the accounting mechanisms implemented in the AAA servers. In particular, servers providing accounting records would still keep on issuing them, yet these records would be addressed to a $cashcard$ instead of a disclosed user identity.

For what concerns specific payment models, we can discuss two separated cases:

1) flat-rate services: For flat-rate services payment can be issued once, at the moment the service is subscribed. When user requests a flat-rate service, SP will contact AS asking for the needed coverage, which will be paid by AS itself in case of a pre-paid $cashcard$ (of course, given that enough credit is available for the requested amount), or will be redirected by AS towards B if a post-paid method is specified.

2) Usage-based services: In case of usage-based services, if the $cashcard$ is pre-paid the mechanism is quite straightforward since user has already paid AS at the time card was released, therefore AS itself will directly manage the remaining credit that can be spent. When user requests a usage-based service, SP will contact AS asking for the maximum amount user can afford end eventually decide whether to start or refuse the provisioning. Otherwise, if the $cashcard$ is post-paid AS will forward the query issued by SP to B and backwards, thus acting as a broker besides keeping the account status up-to-date in an internal cache (this can be done periodically[2]). In both cases, when service provisioning is over SP will finally ask AS for the bill so we return to the same situation discussed above in Section III-B1

IV. Implementation Issues

We are implementing the above described distributed framework in a Linux environment. The core components are the

[2]Notice that finding optimized ways to allow synchronization between AS and B (e.g. let AS cache the account status) is something which is not within the scope of this paper.

User Client, the Identity Repository, the Accounting Server, the Bank and the Service Provider. To date, the User Client and the IR server are already at advanced stage of implementation, while we are still finalizing the implementation of the SP, AS and B servers. Nevertheless, some preliminary lessons have been learned even from our partial implementation, and are hereafter discussed.

The IR server has been implemented as a multithreaded server, to allow management of several clients in parallel without significant performance impairments. Certificates have been generated and managed through the OpenSSL Crypto library [19]. The whole framework is based on the RSA algorithm [20] and on the X.509 standard for digital certificates. Using off-the-shelf crypto functions reduces the implementation time. The server store the transactions log in a back-end database. We used MySQL database [21] in the implementation since it is widely used in Linux systems and guaranties good performances when processing logs of many concurrent clients.

We remark that the decision to generate at the user side the various public/private key pairs involved in the described operation is not only due to security reasons (which in fact would not be compelling in the presence of an underlying security protocol such as TLS[12] or IPsec[11]), but is strongly motivated by performance issues. Actually, we measured that a 2048 bit key pair generation is done in the order of 1 second on a standard PC. As such, if done at the server side, this would represent a dramatic limiting factor for what concerns scalability.

A format for the exchange of messages have been further defined. Specifically, we have decided to employ an Attribute-Value-Pair format similar to that used in RADIUS / Diameter. The message is therefore composed by three fields: i) a type field which specifies the message; ii) a length field with defines the size, and iii) a value field that contains the delivered information. A variety of message types have been specified (client hello, server hello, delivery of the various certificates previously introduced, key requests, etc).

All the messages are conveyed over TCP sockets, and protected by a lower layer security protocol (IPsec has been tested to date; TLS is another viable option). Such a lower layer protection is essential to avoid eavesdropping of the transmitted certificates, and to avoid the need to implement further mechanisms within the proposed handshake to protect the challenge-response from MITM attacks, as well as to protect from other traditional attacks (such as rogue servers, etc).

We consider very important, for exploitation purposes, to integrate our proposed approaches (i.e. pseudonyms assignment and anonymous payment) on top of an highly deployed AAA standard, such as RADIUS or Diameter. Given the higher flexibility of Diameter as both base platform and in terms of AVP capacity and handling, a next step of the work will consist in integrating our code as an extension of opendiameter [22]. Once the whole standalone framework will be ready we intend to develop a plugin version to allow integration with the legacy infrastructures at the service providers.

V. Conclusions

This paper has proposed an integrated framework for distributed pseudonymisation and billing devised to protect the user privacy. The key advantage of our framework is the lack of a single trusted entity. Information is in fact distributed across distinct entities so that its reconstruction is made possible only through their explicit concertation. Ongoing work includes i) a detailed testing of the anonymous authorization credential used in the pseudonym assignment handshake (although perfect for our purposes, Ring Signature may raise scalability problems); ii) finalization of the implementation to include also the accounting part; iii) integration of accounting in a widely deployed AAA platform, with Diameter being the technology of choice.

References

[1] J. Kilian, E. Petrank, Identity Escrow, CRYPTO 1998, LNCS 1462, pp. 169-185, Springer-Verlag, 1998.

[2] J. Camenisch, A. Lysyanskaya, An Identity Escrow Scheme with Appointed Verifiers, CRYPTO 2001, LNCS 2139, pp. 388-407, Springer-Verlag, 2001.

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

[4] G. Medvinsky and 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, November 1993.

[5] 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.

[6] 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.

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

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

[9] DISCREET project homepage - http://www.ist-discreet.org

[10] G. Bianchi and S. Teofili. Multiple-entity-based Multiple Pseudonym Assignment for Improved Privacy Protection. International Conference On Late Advances in Networks (ICLAN'2006), Paris, December 06-08, 2006.

[11] S. Kent, R. Atkinson. Security Architecture for the Internet Protocol, RFC 2041, November 1998.

[12] T. Dierks, C. Allen. The TLS Protocol Version 1.0, RFC 2246, January 1999.

[13] R. Rivest, A. Shamir, and Y. Tauman, How to Leak a Secret, Proc. Asiacrypt '01, pp. 552-565, 2001.

[14] G. Carle, S. Zander and T. Zseby. Policy-based Accounting, RFC 3334, October 2002.

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

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

[17] D. Mitton et al., Authentication, Authorization, and Accounting: Protocol Evaluation, RFC 3127, June 2001.

[18] R. Housley, W. Polk, W. Ford, D. Solo. Internet X.509 Public Key Infrastructure Certificate and Certificate Revocation List (CRL) Profile, RFC 3280, April 2002

[19] OpenSSL project homepage - http://www.openssl.org

[20] R. L. Rivest, A. Shamir, L.A. Adleman. A method for obtaining digital signatures and public-key cryptosystems. Communications of the ACM, Vol.21, Nr.2, 1978, S.120-126

[21] MySQL project homepage - http://www.mysql.org

[22] Open Diameter project homepage - http://www.opendiameter.org