Available online at www.sciencedirect.com
ScienceDirect
Electronic Notes in Theoretical Computer Science 197 (2008) 57–71
Electronic Notes in Theoretical Computer Science
www.elsevier.com/locate/entcs

The SPARTA Pseudonym and Authorization System

G. Bianchi, M. Bonola, V. Falletta, F. S. Proto, S. Teofili

Dipartimento di Ingegneria Elettronica
Università di Roma "Tor Vergata"
Roma, Italy

Abstract

This paper deals with privacy-preserving (pseudonymized) access to a service resource. In such a scenario, two opposite needs seem to emerge. On one side, the service provider may want to control in first place the user accessing its resources, i.e., without being forced to delegate the management of access permissions to third parties to meet privacy requirements. On the other side, it should be technically possible to trace back the real identity of an user upon dishonest behavior, and of course this must be necessary accomplished by an external authority distinct from the provider itself. The framework described in this paper aims at coping with these two opposite needs. This is accomplished through i) a distributed third-party-based instrastructure devised to assign and manage pseudonym certificates, decoupled from ii) a two-party procedure, devised to bind an authorization permission to a pseudonym certificate with no third-party involvement. The latter procedure is based on a novel blind signature approach which allows the provider to blindly verify, at registration time, that the user possesses the private key of the still undisclosed pseudonym certificate, thus avoiding transferability of the authorization permission.

Keywords: Privacy, Trust management, Pseudonym system, Blind Signature

1 Introduction

Traditional Authentication and Authorization services take into little consideration the protection of the user's privacy. For instance, most of the currently deployed AAA (Authentication, Authorization and Accounting) functions are managed through a (logically) single AAA server such as Radius [1] or Diameter [2] which univocally refers to the real user identity. However, disclosure of the user identity is, in general, not strictly necessary for the service provision. As widely discussed in literature work [3,4,5,6], service authorization may in fact be conveniently based on the proof that the user possesses some "rights" (e.g. credentials, certificates, money availability, etc) which guarantee her permission to access the service meanwhile retaining anonymity. Despite the great scientific interest in privacy-preserving approaches, to date only a limited effort has been spent to adapt such approaches to operate with existing and widely deployed standards (see e.g. [7] for a standard-based approach relying on X.509 Attribute Certificates).

doi:10.1016/j.entcs.2007.12.017

Real world application of privacy-preserving techniques has to further face the important fact that fully anonymous access (as provided by techniques such as ring signatures [8,9,10] or some usage of zero-knowledge approaches [11]) is not a viable solution. For accounting or service control/enforcement/revocation purposes, it is convenient to have technical ways to link the authorization credentials to a single - although undisclosed - user, e.g., by having an explicit label (namely, a pseudonym) associated to the user herself. Even more important, social security reasons, regulatory provisions, or even simple business convenience, mandate for the technical possibility to trace back the real user identity, e.g when a dishonest behavior is detected or when law/security authorities require so [1]. Clearly, the ability to revoke anonymity must be delegated to third party entities, to guarantee the users that the service provider is not able to violate their privacy in ordinary conditions. Indeed, most of the pseudonym and Identity Escrow systems proposed in the literature [12,13,14] base their operation on a single, trusted, third party.

Conversely, the involvement of third parties in an authorization system not only is not functionally necessary, but may also be considered as counter-productive. In fact, a provider typically wants to have direct control on the access permissions issued to its own users: delegation of such a business-critical feature to an external party, not directly involved in the service provider business, may be in practice considered a too high price to pay for "just" respecting the user's privacy. However, designing an anonymous authorization system which does not make use of third parties appears a hard task, especially when non-transferability of the access credentials is required. And in fact, to the best of our knowledge, basically all the most known anonymous authorization and credential-based frameworks [15,16,17,18,7] rely on a complex infrastructure involving trusted third parties.

The system described in this paper is part of a currently under development framework, called SPARTA (Secure Pseudonym Architecture with Real Time Accounting [2]). The SPARTA framework aims at coping with the opposite needs, in terms of third party involvement, of the pseudonymization and authorization functions.

The infrastructure in charge to assign and revert pseudonyms is reduced to a simple distributed infrastructure whose only goal is to provide valid pseudonym certificates. This allows us to rely on widely accepted and standard-based certificate formats (X.509), and on the efficient and mature means to handle them in a PKI (Public Key Infrastructure). The proposed approach is distributed and user-centric, meaning that the user has the freedom to decide which, and how many, entities composing the infrastructure will be involved in her assigned pseudonyms (possibly multiple). While no single entity involved in a pseudonym assignment is capable

[1] Note that restricting the triggering of a pseudonym reversion operation to the occurrence of one among a clearly specified set of technical events might be overly restrictive in terms of real-world applicability, as law provisions might not be readily expressed into precise technical statements (e.g. through a formal policy language).

[2] Our SPARTA framework is being developed in the frame of the EU funded IST Project DISCREET, contract number 027679. As the acronym implies, we indeed aim at further extending the system hereafter described with privacy-preserving accounting and billing functionalities. The specification of these supplementary functionalities is, to date, preliminary and object of ongoing research work (and as such outside the goals of this present paper).

to trace the user, the system retains the possibility to determine the real identity behind a pseudonym through explicit interoperation (e.g., triggered by an authority) among all the entities involved in its assignment. In other words, rather than being forced to trust a specific and single third party, the user needs only to trust that two or more entities selected by the user herself will not collude against her privacy rights.

The authorization function consists in binding a specific service provider signature to the pseudonym that later on will be used to access a resource. Different privileges are provided by having distinct signatures for each different resource and access permission level. The service provider signature is performed at registration time through a blind approach (to prevent disclosure of the pseudonym used later on) involving only the service provider and the end user. To avoid transferability of the pseudonym signature (and in particular to avoid pseudonym hijiacking, i.e. having an user submitting for service provider signature another user's pseudonym), an innovative "marked" blind signature approach is introduced, to include verification of possession of the pseudonym certificate private key inside the signature itself, i.e., at registration time.

The rest of this paper is organized as follows. Section 2 introduces the basic ideas behind the proposed system. Section 3 details the distributed pseudonym assignment infrastructure and its design as a PKI. Section 4 describes the cryptographic details of the novel "marked" blind signature solution proposed to issue authorization permissions. Section 5 discusses implementation issues and deployment assumptions concerning the lower layer security primitives. Conclusions are drawn in section 6.

2 Scenario and Basic concepts

The reference scenario considered in this paper is the following. We assume that an user is provided with an identity certificate U, for instance released at an initial off-line "subscription" time by a service provider SP in the form of a standard X.509v3 certificate[3]. We further assume that there is a subsequent off-line "registration" time where the user presents herself with her real identity U to the service provider SP, and agrees to access a service/resource S. We include the type of access in the definition of S, meaning that if a same physical resource or service may be accessed with different privilege levels, we treat these different cases as different services S. This agreement may be further based on supplementary information eventually presented by the user at registration time, as well as procedures, such as payment of a flat fare, performed (or documented) at registration time. No anonymity or privacy protection is provided, in our framework, for data presented during this phase.

[3] We remark that the extension to federated identity management systems, e.g. Liberty Alliance [19] is conceptually straightforward - the difference being that in such case a further indirection exists between the provider that offers the service, which we referred to as SP, and the central entity that manages the user identity. In the provided description, for ease of presentation, we non restrictively assume coincidence between these two entities.

During such registration phase, the user will receive one or more authorization permissions through which she will be able to access S at later times. An authorization permission consists in having the SP signing, with a signature key specific for each service or resource S, a pseudonym certificate blindly submitted to the SP (pseudonym authorization). As such, an authorization permission is accountable, i.e. reuse of a same access permission implies reuse of the same authorized pseudonym. Clearly, the user may prevent linkability by asking the SP to authorize more than one pseudonym during registration phase, and then change pseudonyms, among the authorized ones, through different access sessions.

To be valid, a pseudonym certificate must be released by a third party certification authority (referred to as Identity Repository in section 3), trusted by the SP. The SP may also require the users to access S with a pseudonym certificate satisfying specific policies (e.g., expiration date, state in which it is released, etc). It is up to the user to submit, at registration time, a valid pseudonym, as its validity cannot be checked at registration time (being the pseudonym blindly submitted), but will be at access time and a non valid pseudonym will not be granted access.

An important point characterizing our proposal is that the pseudonym validity, which is delegated to the proper operation of the PKI-like infrastructure described in section 3, is clearly decoupled from the authorization permission, which is independently issued by the SP and binded to the pseudonym. A valid pseudonym simply guarantees that the user identity may be traced back from the pseudonym (through a procedure involving the PKI) if regulatory provisions or security reasons require to do so. But it is important to remark that the SP remains independently capable to revoke the access permission when a previously authorized pseudonym is used in a dishonest way (this being trivial as the subsequent accesses performed with a specific pseudonym are accountable).

2.1 Why a new blind signature?

If applied in the above described scenario, traditional blind signatures (e.g., that first proposed in [20]) would fail to meet the important requirement of providing non-transferability of an authorized pseudonym. The problem, which we refer to as “Pseudonym Hijiacking”, is that the user U, during the registration time, may deliver the SP a pseudonym certificate P' of another user U', and have it blindly signed for authorization. Note that the other user would only need to give U the pseudonym certificate P' for its blind signature, and not the corresponding private key, thus remaining the only one able to actually use the certificate P'. More advanced blind signatures, such as the Fair Blind Signatures first proposed in [21], may be integrated in a comprehensive authorization framework, as the one proposed in [7] which indeed solves these problems, but require to deploy an elaborated operation involving third party entities (such as [7]'s Attribute Authorities/Sub-Authorities) to support the SP for authorization tasks. Similarly, group signatures [22] would again guarantee non-transferability, but would require a third party verifier, which is something that we are trying to avoid.

Conversely, we remark that it would be possible to trivially solve the pseudonym non-transferability issue, meanwhile retaining the above described two-party authorization framework, by devising a blind signature which integrates, in its operation performed at registration time, an explicit proof of possession of the pseudonym's private key. This in fact would prevent pseudonym hijiacking as it would be necessary, for an hijiacker, not only to provide the user with the pseudonym certificate P', but also its private key (i.e., giving away the pseudonym). Note that the user U would now be in the perfect condition for abusing of the the pseudonym P': dishonest behavior would be in fact accounted to P', and hence linked to the user U'!

As thoroughly described in section 4, we have provided pseudonym non-transferability by designing a novel blind signature handshake which generates a random value R, unforgeable by both the user and the service provider, and which remains unknown to the SP (while it will be ultimately revealed to the user at the end of the handshake, as this value needs to be submitted later on at verification time). R can be hence used as random challenge, to execute what we descriptively refer to as *Delayed Pseudonym Certificate Verification.* The idea is to ask the user, at registration time, to prove possession of the private key of the pseudonym certificate P through a signature taken over a message $f(R, P)$, and wrap this signature inside the blind message which will be signed by the SP. Verification of the pseudonym signature will occur later on when the access permission will be exposed (hence the "delayed" verification feature) and in conjunction with the access permission verification.

To the best of our knowledge, ours is the first blind signature approach which integrates inside the signed message an unknown and unforgeable random value, which may be thought as a "mark" of the act of blind signing. Hence the name "marked blind signature".

3 Pseudonym Assignment

The procedure to assign a pseudonym certificate P to an user is done offline, i.e. before the actual access to the service, and as such does not add extra time and/or and/or computational burden to the service provision. Since a pseudonym P shall be submitted by the user at registration time (see section 2), the pseudonym assignment procedure is performed after the real identity certificate U is issued and before the registration time.

Consistently with the scenario described in section 2, we assume that the identity certificate U, representing the real identity of the user, is issued by the SP at subscription time. In addition to the certificate U, still at subscription time, the SP further releases[4] a "token" certificate T_0. This certificate is an alias for the real user identity U, and it is generated[5] so that any other entity besides the SP

[4] For security reasons and performance/implementation convenience - see further considerations in section 5 -, the private/public key pairs for T_0, as well as for all the subsequent certificates, are generated by the end user. The certificate issuing procedure for T_0 is therefore a simplified version of that illustrated later on for the more general IR case.

[5] While the rule to compute the token certificate T_0 is in most generality left to the SP, and in principle

should not be able to determine U from T_0. Instead, the SP will keep locally track of the mapping between U and T_0.

The $U \rightarrow T_0$ mapping provides a first level of indirection for the real user identity U. Now, the idea is to proceed with such an indirection and derive an user pseudonym by simply involving supplementary entities. Each intermediate entity acts as a Certification Authority (CA), devised to i) receive, as input, a valid token certificate, ii) return, as output, another valid certificate, and iii) keep track of the input-output certificate mapping. This indirection mechanism is provided through completely standard PKI primitives and their off-the-shelf crypto mechanisms.

To this purpose, after having received the token certificate T_0, the user chooses one of such entities, hereafter referred to as Identity Repositories (IR), and submits T_0. Note that the IR is not able to determine the identity of the user from T_0, but can only verify that T_0 is a valid certificate, and specifically that it is issued by a valid CA, in this first case the SP itself. In return, the user receives a new token certificate T_1 signed by the chosen IR. This process can be either i) iterated through a chain of IRs, or ii) parallelized, by having the user submitting the initial T_0 more than once and receive in response multiple tokens.

In details, at any generic i-th step, the following same procedure is adopted (we assume that the communication channel is secured and the peers authenticated through a lower layer mechanism - e.g., IPsec or TLS, see discussion in section 5 -, to offload the procedure from supplementary security mechanisms not strictly functional for the envisioned operation, and added only to protect from e.g., eavesdropping and MITM attacks):

$$\begin{array}{lll} \text{User} \rightarrow \text{IR}_\text{i} : & \{T_{i-1}, e_i\} & (1) \\ \text{IR}_\text{i} \qquad\quad : & \text{verify\_signature}(T_{i-1}) & (2) \\ \text{IR}_\text{i} \leftrightarrow \text{User} : & \text{challenge}(T_{i-1}) & (3) \\ \text{IR}_\text{i} \qquad\quad : & \text{policy\_check}(T_{i-1}) & (4) \\ \text{IR}_\text{i} \rightarrow \text{User} : & T_i & (5) \end{array}$$

In this straightforward handshake, at step (1) the user generates a pair of public/private keys, and sends the IR the certificate (token) currently owned (namely T_{i-1} to point out that this is the token achieved at step $i-1$), plus the public key e_i to be included in the next token T_i. The IR duly verifies (step 2) that the certificate was issued by a valid certification authority (IR or SP), and verifies that the user possesses the certificate private key through signature of a random challenge (step 3). Further policy controls on the certificate (state, associated permissions, expiration time, etc - some additional remarks are provided at the end of this section) are

a random generation of T_0 would be appropriate, for practical application it may be convenient to provide a fixed rule which limits the SP to the possibility of generating a unique T_0 value for each user. This may be for instance accomplished having $T_0 = U^*$ where U^* is a certificate containing an encrypted version of the user identity. A possible reason for such a practical limitation may be the fact that, in a scenario involving several small SPs, they may not be considered equivalent, in terms of trust level, to the other IR certification authorities deployed in the system. As such, a rule that prevents the possibility for an SP to generate multiple tokens per each user may be appropriate.

then carried out. Finally, if all checks are successful, the IR embeds the provided public key e_i into a new token certificate T_i. As final pseudonym P, the user simply choses the last token in this chain (where we stress that such chain is freely decided by the user).

Neglecting for the time being security attacks, the identity of a user can be reconstructed if and only if the initial SP and all the subsequent IRs chosen by the user explicitly interact to revert back the input-output certificate mapping. Through this proposed operation we thus avoid that a single entity alone (e.g. one of the IR or the SP) shall be capable of reverting the user pseudonym and linking it back to the real user identity. Meanwhile we guarantee the technical possibility to revert the assigned pseudonym through explicit interaction between the IRs and the SP.

Despite its extreme simplicity, this approach is indeed effective and may be extended to give raise to a full-fledged Identity Management PKI driven by the user decisions. In fact, it is up to the user to decide which IRs to use, and whether to use a single IR or a multiplicity (for improved robustness of the reversion of this process). This makes all the framework strongly user centric.

In parallel, the set of deployed IRs form a PKI infrastructure. This means that the IR must maintain a list of trusted CAs (both IRs and SPs), and accept certificates issued by other CAs depending on deployed policies (regulatory, etc). For instance, this allows the user to derive tokens (pseudonyms) from a chain of IRs involving different administrative domains or even states, which they may be later on accepted as valid by the SP depending on the specifically issued service (in other words, for some services it is possible to impose that the pseudonym must be issued by a subset of IRs - e.g. from a same state). We point out that the choice of obtaining a pseudonym through a given chain of SP/IRs clearly affects the regulatory conditions under which the pseudonym may be reverted. For a trivial example, the fact that a pseudonym has been obtained by chaining two IRs from two different states means that the authority capable of reverting it must be a trans-national one.

As shown in the next section, the revocation of an authorization permission for a single misbehaving pseudonym is locally managed by the SP itself. In fact we will show that an authorization permission is a credential issued by the SP only, with no involvement of the described pseudonym PKI. A more elaborated problem is the revocation of all the pseudonyms associated to a same real user identity. This can be accomplished in a distributed way by the PKI components through the usual revocation approaches (management of Certificate Revocation Lists). Particularly, each IR server shall periodically check that its issued certificates are not included in the CRL. If an issued certificate is found to be revoked, we can take advantage of the mapping internally hold by the IR, and accelerate the pseudonym revocation procedure by selectively informing the parent IR in the chain.

4 Authorization Permission Assignment

As discussed in section 2, traditional blind signatures applied to our scenario suffer of the pseudonym hijiacking problem. To avoid this problem, we have devised a novel blind signature which allows to include, at the time of signature, a proof of possession of the private key of the pseudonym P to be used later on.

4.1 Marked Blind Signature

The proposed protocol is delevoped for RSA blind signatures. It operates by explicitly including a "mark", i.e., an unknown and unforgeable random value R, inside the signed message. This random mark is then exploited as challenge to verify possession of the pseudonym certificate private key. The following notation is hereafter used:

- P: pseudonym certificate, with RSA public key $= e_p$, private key $= d_p$ and modulo n_p;
- S: service authorization certificate, with RSA public key $= e$, private key $= d$ and modulo n with the assumption $n > n_p$;
- a: DL-strong base [23] for n, generated by the Service Provider.
- $x, s \in \mathbb{Z}_n^*$: random number generated by the user, with the further limitations discussed in the next definition of R;
- $y \in \mathbb{Z}_n^*$: random number generated by the Service Provider, with the further limitations discussed in the next definition of R;
- $R \triangleq xy + s$, with the condition that $xy + s < n$, or in other words that the computation of R using non modular arithmetic results equal to that using arithmentic modulo n; this can be for instance guaranteed by imposing $x < \sqrt{(}n/2)$, $y < \sqrt{(}n/2)$, and $z < n/2$;
- $B \in \mathbb{Z}_n^*$: random blind factor generated by the user, with B^{-1} being its inverse modulo n;
- H: a one way hash function, such as $\mathrm{Im}(H) \subseteq \mathbb{Z}_{n_p}$

Unless otherwise specified, all the following operations are modulo n. The following handshake relies on the double homomorphic property of the Discrete Logarithm hashing. For any two values X_1 and X_2 it is:

$$\left(a^{X_1}\right)^{X_2} = a^{X_1 \cdot X_2}$$

$$a^{X_1} \cdot a^{X_2} = a^{X_1 + X_2}$$

These properties allow the user to perform an homomorphic computation of R, hence without getting to know the actual value R. In fact, given three values x, y and s so that $R = xy + s$, it is $(a^y)^x \cdot a^s = a^{xy+s} = a^R$. Moreover, the supplementary conditions on the random values x, y, s imply that the ordinary algebraic computation of $xy + s$, as occurring as exponent of a, coincides with that performed in modulo n. This is inserted in the signature handshake as follows.

$$\text{SP} \rightarrow \text{User}: \quad a^y \tag{1}$$

$$\text{User}: \quad (a^y)^x a^s = a^R \tag{2.1}$$

$$\text{sign}(R, P) = H(a^R \| P)^{d_p} \bmod n_p \tag{2.2}$$

$$\text{User} \rightarrow \text{SP}: \quad x_1 = B^e x \tag{3.1}$$

$$x_2 = B^e \left(\text{sign}(R, P) + s\right) \tag{3.2}$$

$$\text{SP}: \quad x_1 y = B^e x y \tag{4.1}$$

$$x_2 + x_1 y = B^e \left(\text{sign}(R, P) + s + xy\right) = B^e \left(\text{sign}(R, P) + R\right) \tag{4.2}$$

$$\text{SP} \rightarrow \text{User}: \quad (x_2 + x_1 y)^d = B \left[\text{sign}(R, P) + R\right]^d \tag{5}$$

As a result, after removing the blinding factor (through modular multiplication with B^{-1}) in the message received at step (5) the user obtains the signed authorization credential

$$\text{cred} = \left[\left(H(a^R \| P)^{d_p} \bmod n_p\right) + R\right]^d$$

The user can now compute R as

$$R = \text{cred}^e - \left(H(a^R \| P)^{d_p} \bmod n_p\right)$$

where the second term was earlier computed at step (2.2).

4.2 Authorization credential verification

The above authorization credential, constructed at registration time, will be verified later on at access time. Verification is straightforward and consists in the following steps:

- the user presents the pseudonym certificate P, which is verified through an usual challenge-response handshake;
- the user then presents the pair (cred, R), and specifies which service S is the authorization credential valid for;
- the SP computes $H(a^R \| P)$ and verifies, using the RSA public key e associated to S, and the RSA public key e_p associated to P, that

$$(\text{cred}^e - R)^{e_p} = H(a^R \| P)$$

4.3 Discussion

The detailed security analysis of the proposed signature mechanism is outside the goals of the present paper, and it is object of work in progress. Some preliminary considerations follow, with the double goal of i) understanding the rationale behind the proposed approach, and ii) describing how the proposed approach is devised to defend against some simple forgeability and traceability attacks. In the following discussion we assume that the communication channel is secure and the communicating peers authenticated (i.e. no MITM attacks).

The transmission of the server side random value y occurs at step (1). Due to the discrete logarithm hashing, it is computationally hard for the user to obtain y. Note that this random value must remain unknown to the user during the handshake as, otherwise, it would be trivial for the user to forge a value R' and vanish the desired properties of this signature mechanism. This would be obtained by sending $x_1 = B^e y^{-1} x'$ and $x_2 = B^e \left[\left(H(a^{x'+s'} \| P)^{d_p} \bmod n_p \right) + s' \right]$ thus embedding in the credential an arbitrarily chosen value $R' = x' + s'$.

Step (2.1) consists in the homomorphic computation of R on the user side. Unforgeability of R is provided by the usage of modulo n in the Discrete Logarithm, and therefore by the anticollision properties of the DL-hashing discussed in [23], whose security is proven to be equivalent to the factorization of n. The term a^R so computed is prepended to the pseudonym certificate P, and the result is hashed and signed with the pseudonym certificate private key (step 2.2).

Step (3) consists in the blind transmission of both the user random value x as well as the previously signed hash. Note that a second random number s is here added to the result, to prevent that the elimination of the blinding factor B^e, e.g. through $(x_2 + x_1 y)/(x_1 y)$, results in a term including only R which could hence be used to trace the subsequent access.

We remark that the security of the system requires the SP to explicitly include, inside the signature, its own computed version of the random value R, to prevent the user from forging R at will. This computation is blindly carried out in step (4.2). Since this computation occurs with modulo n arithmetics, to guarantee that the user computation of R (occurring with ordinary arithmetic) coincides with the service provider computation we restrict the value R to result lower than n through appropriate restrictions on the randomly generated values x, y, z.

Furthermore, we remark that step (4) is specifically designed to include R as a modular addendum, and for this reason two blinded messages are sent. It can be argued that a multiplicative insertion of R in the signature could have been trivially achieved with just a single blinded message, but this would have in fact lead to an universally forgeable signature.

Finally, forgeability of the access credential is prevented by the one-way properties of the chosen hash function. Specifically, to include in a previously signed credential c a new value $\bar{R}$ chosen by the user, it is necessary to find a value $\bar{R}$ which satisfies the following condition:

$$\left(c^e - \bar{R}\right)^{e_p} = H(a^{\bar{R}} | P) \bmod n_p$$

which is prevented by the anticollision properties of the properly chosen hash function.

5 Implementation issues and choices

5.1 Assumptions on lower layers

In the previous description, we have assumed that security services are provided by the layers below the proposed application to guarantee a secure communication channel and authenticated peers to prevent Man In The Middle attacks. We now provide a more precise list of assumptions concerning the underlying communication.

(i) *Encryption*: All data exchanges within our system are protected by datagram encryption. Employing TLS and IPSec and their support for off-the-shelf advanced key management and encryption means ensure robustness against crypto analysis attacks.

(ii) *Server/Message Authentication*: Relying on standard protocols (such as EAP-TLS, EAP-TTLS, TLS including the pseudonym certificate for user authentication - see also the following discussion concerning our TLS implementation) guarantees a strong protection against common attacks such as MITM and Rougue Servers, preventing message modification and tagging.

(iii) *IP Natting / Anonymous networking*: Employing an Anomymous Proxy (located inside a domain not managed by the user ISP) having IP Natting functionalities, or using Mix networks such as Tor [24] prevents users from being linked by any Server which could simply bind different pseudonyms/credentials to the same IP address.

(iv) *Safe Key Storage*: All the users' and servers' private keys must be safely stored.

5.2 Implementation issues

Special care must be placed to the timing of the various procedures. For instance, it would be preferable to use the SP as a final element of a CA chain since it would release the final pseudonym P to be used later on for accessing the offered service. However, this would lead to possible correlation attacks devised to link the time at which a pseudonym is released, and the subsequent time in which the user registers and exposes its real identity.

5.3 Current implementation choices

A complete implementation of the current version of the SPARTA framework is available online [6] . The current implementation is based on RSA, with 2048 bit keys, and on the X.509v3 standard for digital certificates. The SPARTA Software Library

[6] All the software developed is released under GPL License and is available for downloading at http://www.ist-discreet.org together with an online demo.

is based on the OpenSSL Crypto library [25] for efficient certificates generation and management. Specific functions have been further implemented for the Marked Blind Signature, by extending the sub-libraries provided by OpenSSL.

An integrated implementation approach has been followed. Rather than implementing each different server as a separate entity, a general Multi Purpose Server (MPS) integrates the server side logic for the pseudonym assignment handshake (section 3) as well as the authorization credential issuing handshake (section 4). The MPS is implemented as a multithreaded server thus allowing the management of several clients in parallel without significant performance impairments. The server stores the transactions log in a back-end database. We used the MySQL database [26], which is well known by the open software community and guarantees good performances while processing logs of many concurrent clients. On the user side, we developed a Pseudonym Manager tool to assist the user through the various token/pseudonym assignment, registration and verification procedures. We choose to develop a command-line tool, for ease of integrability in other softwares, with a GUI commander for standalone user friendly operation. As regards the credential verification at the time of service provisioning we also developed a standalone command-line verification tool, with the intent of being easily integrable in the logic of the service application.

Concerning exchanged messages, we have defined a common unique format. Specifically, we have decided to employ an Attribute-Value pair format similar to the one used in Radius/Diameter. The message is therefore composed by three fields: i) a *type* field which specifies the type of message; ii) a *length* field with defines its 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) for the registration and verification procedure. Further specific message types are introduced to support the web service implementation version presented in section 5.4.

All the messages are conveyed over TCP sockets. As discussed in section 5.1, we assume the presence of an underlying security protocol to secure all the communications, and our implementation choice was TLS, mostly for ease of implementation and the possibility to deploy the proposed solution in a web service scenario (section 5.4).

In the current implementation, all the public/private key pairs are generated at the user terminal side. Such a decision is due to both security and performance reasons. For what concerns security, the fact that a private key for a token or pseudonym certificate is never released by the end user prevents the possibility to use rogue IR servers in order to steal the token/pseudonym certificate private key, i.e. ultimately steal the user identity. Clearly proper storage of the private keys is mandatory and delegated to off-the-shelf software key-rings. Regarding performance, we have measured the time needed to generate a 2048 bit key pair on an entry-level laptop (e.g. Intel Centrino 1.6GHz with 512MB RAM). This time results in about 1 second, which is tolerable on the user side, but which might become a performance bottleneck if implemented on a server side, especially when scalability is aimed at.

5.4 Example service scenario: web portal

An example service scenario has been developed as a web portal. An user is able to login by submitting a valid pseudonym certificate (instead of a username), and a valid authorization credential (instead of a password). It is worth to remark that the pseudonym certificate submission is being integrated as client certificate inside the TLS handshake. As such, pseudonym certificate verification is directly provided inside the TLS operation. To date we have not find a way to include the authorization credential verification inside TLS (we believe that this requires modification to the TLS standard, through addition of a specific supplementary message). Therefore, the submission of the authorization credential is prompted by the web portal, after the establishment of the TLS connection and the visualization of the portal home page. A credential verification tool is then executed: it is invoked as a Unix system call by a PHP script which implements the web portal application. Using X.509v3 standard for digital certificates, an user is able to store her pseudonym and her private key inside her browser using PKCS#12 (Personal Information Exchange Syntax Standard) and can protect her private key within the embedded browser key-ring.

5.5 Ongoing implementation work

The implementation of policy-based decisions on the token/pseudonym certificates will follow as a next step. As anticipated in section 3, the token/pseudonym certificate verification is not limited to signature verification, but explicitly includes a policy check, which is missing in the current implementation. Indeed, this is a key feature for a viable practical deployment of our proposed approach to allow the integration of services characterized by different policies and regulatory provisions.

We are also working to extend the web service scenario. Indeed, integrating SPARTA components into web-based applications brings about a series of advantages. First, employing user's pseudonym certificates inside the TLS handshake for connection security and certificate verification will speed up software performance. Second, web applications are intrinsically user friendly, cross platform, and accessible to a wider scope of users. To this purpose, current work consists in the integration of the user-side Pseudonym Manager as a plug-in of a widely used public domain web browser (for instance, as done by the Higgins project [27] Firefox plugin), and improve the integration of the current server-side functionalities into web-based applications and in the TLS protocol.

Finally, SPARTA is being extended to support anonymous payments and real-time accounting. The idea is to generalize the use of the marked blind signature as authorization tool non only for pseudonym certificates, but also for anonymized payment-related certificates.

6 Conclusion

In this paper we proposed a framework for user pseudonymization with service authorization. The main novelties this paper introduces are i) a fully distributed PKI-like user centric infrastructure for pseudonym assignment, ii) a system that works with the absence of a trusted third party and finally iii) a novel blind signature approach to provide a valid credential for service authorization to a user pseudonym.

References

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

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

[3] C. Andersson, J. Camenisch, S. Crane, S. Fischer-Hübner, R. Leenes, S. Pearsson, J. S. Petterson, D. Sommer, "Trust in PRIME", 5th IEEE Int. Symposium on Signal Processing and Information Technology, Athens, Greece, December 2005

[4] S. Brands, F. Légaré, "Digital Identity Management based on Digital Credentials", GI Jahrestagung 2002

[5] Y. Dodis, A. Kiayias, A. Nicolosi, V Shoup, "Anonymous Identification in Ad Hoc Groups", EUROCRYPT 2004, Lecture Notes in Computer Science, vol. 3027, pp. 609-626. Springer-Verlag, 2004

[6] D. Chaum, "Security without identification: Transaction systems to make big brother obsolete", Communications of the ACM, 28(10), pp. 1030-1044, Oct. 1985

[7] V. Benjumea, J. Lopez, J.A. Montenegro, J.M. Troya, "A first approach to provide anonymity in attribute certificates", 2004 International Workshop on Practice and Theory in Public Key Cryptography, Lecture Notes in Computer Science, vol. 2947, pp. 402-415, Springer-Verlag, 2004

[8] R. Rivest, A. Shamir, Y. Tauman, "How to leak a secret", ASIACRYPT 2001, Lecture Notes in Computer Science, vol. 2248, pp. 552-565. Springer-Verlag, 2001

[9] M. Au, J. K. Liu, P. P. Tsang, D. S. Wong, "A suite of id-based threshold ring signature schemes with different levels of anonymity", Cryptology ePrint Archive, Report 2005

[10] A. Bender, J. Katz, R. Morselli, "Ring Signatures: Stronger Definitions, and Constructions Without Random Oracles", Theory of Cryptography (TCC 2006), Lecture Notes in Computer Science, vol. 3876, pp. 60-79. Springer-Verlag, 2006

[11] U. Feige, A. Fiat, A. Shamir, "Zero Knowledge Proofs of Identity", Proc. 19th ACM Symp. on Theory of Computing, May 1987, pp. 210-217

[12] A. Lysyanskaya, R. L. Rivest, A. Sahai, S. Wolf, "Pseudonym Systems", Selected Areas in Cryptography, pages 184-199. Springer-Verlag, 1999. LNCS no. 1758

[13] J. Kilian, E. Petrank, "Identity Escrow", CRYPTO 1998, Lecture Notes in Computer Science, vol. 1462, pp. 169-185, Springer-Verlag, 1998

[14] J. Camenisch, A. Lysyanskaya, "An Identity Escrow Scheme with Appointed Verifiers", CRYPTO 2001, Lecture Notes in Computer Science, vol. 2139, pp. 388-407, Springer-Verlag, 2001

[15] S. A. Brands, "Rethinking public key infrastructures and digital certificates", MIT Press, 2000.

[16] J. Camenisch, E. Van Herreweghen, "Design and Implementation of the Idemix Anonymous Credential System", ACM Computer and Communication Security, 2002.

[17] J. Camenisch, A. Lysyanskaya, "Efficient Non-transferable Anonymous Multi-show Credential System with Optional Anonymity Revocation", Extended abstract in: Advances in Cryptology - Eurocrypt 2001.

[18] J. Camenisch, A. Lysyanskaya, "A Signature Scheme for Efficient Protocols", In Third Conference on Security in Communication Networks, 2002.

[19] Liberty Alliance homepage, http://www.projectliberty.org

[20] D. Chaum, "Blind signatures for untraceable payments", Advances in Cryptology - Crypto '82, Springer-Verlag (1983), pp. 199-203.

[21] M. A. Stadler, J. M. Piveteau, and J. L. Camenisch, "Fair blind signatures", Advances in Cryptology EUROCRYPT95, Lecture Notes in Computer Science, vol. 921, pp. 209219, Springer-Verlag, 1995.

[22] D. Chaum, E. van Heyst, "Group signatures", Advances in Cryptology EUROCRYPT 91, Lecture Notes in Computer Science, vol. 547, pp. 257-265, 1991.

[23] J.K. Gibson, "Discrete logarithm hash function that is collision free and one way", IEEE Proceedings, Vol. 138, No. 6, November 1991, pp. 407-410.

[24] R. Dingledine, N. Mathewson, P. Syverson, "Tor: The Second-Generation Onion Router", Proc. of the 13th USENIX Security Symposium, August 2004.

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

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

[27] Higgins project homepage - http://www.eclipse.org/higgins/