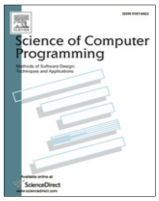
Science of Computer Programming 74 (2008) 23–33

Contents lists available at ScienceDirect

Science of Computer Programming

journal homepage: www.elsevier.com/locate/scico



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", Via del Politecnico, 1 - 00133 Roma, Italy

ARTICLE INFO

Article history:
Received 7 April 2008
Accepted 3 September 2008
Available online 30 September 2008

Keywords:
Privacy
Trust management
Pseudonym system
Blind signature

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 issuing 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 a 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 infrastructure 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 service subscription time, that the user possesses the private key of the still undisclosed pseudonym certificate, thus avoiding transferability of the authorization permission.

1. Introduction

Traditional Authentication and Authorization services take little consideration of 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 [28] or Diameter [26] 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 [5,1,8,17,4], 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 great scientific interest in privacy-preserving approaches, to date only 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), or with reputation systems [15].

A 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 [23,25,2] or some usage of zero-knowledge approaches [18]) 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 with the user. 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 dishonest behavior is detected or when law/security authorities require to do so. 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.

* Corresponding author.
E-mail addresses: giuseppe.bianchi@uniroma2.it (G. Bianchi), marco.bonola@uniroma2.it (M. Bonola), vincenzo.falletta@uniroma2.it (V. Falletta), proto@ing.uniroma2.it (F.S. Proto), simone.teofili@uniroma2.it (S. Teofili).

0167-6423/$ – see front matter © 2008 Elsevier B.V. All rights reserved.
doi:10.1016/j.scico.2008.09.010

Indeed, most of the pseudonym and Identity Escrow systems proposed in the literature [22,21,10,9] base their operation on a single, trusted, third party.

Conversely, the involvement of third parties in an authorization system is strictly necessary only when the authorization procedure relies on user attributes which are, by their nature, external and independent of the service provider domain. For instance, the age of a customer is an objective attribute which is naturally certified by a legal entity different from the service provider. Similarly, the possession (or proof of possession) of a driving licence as condition to access a car hiring service is a credential that must be released by a motor vehicle governmental department.

However, there are several scenarios in which the authorization permission depends in total, or at least in part, on business agreements specifically contracted with the provider, and/or on information that is determined, and managed, only by the provider itself. For instance, the user could get a free of charge access to an added value services because they have adhered to a special promotion involving another service offered by the same provider. Similarly, access to an airport facility could be restricted to users which have acquired a sufficient mileage in a frequent flyer program. In all these cases, the authorization credential is something that is tightly related to the service provider business, and which should be certified only by the provider itself and not by a third party.

We believe that a strategic limitation of most anonymous authorization and credential-based frameworks is the fact that they are based on third parties which issue all the authorization credentials. In fact, despite the specific credential data format (e.g. represented by X.509 certificates or by raw values) and despite the cryptographic signature algorithm employed, a common feature of these approaches [3,7,6] is that the entity which verifies the credentials is different from the one which issues them. This approach has undoubtable significant technical advantages and has lead to the proposal of powerful frameworks [11] that exploit new signature mechanisms such as the one presented in [12] to issue authorization non-transferable multi-show credentials. However, requiring the involvement of third parties for also issuing access permission which is naturally contracted only with the service provider seems unviable. A provider typically wants to have direct control on access permission issued to its own users and on the information based on which such access permissions are released: 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 a price to pay for "just" respecting the user's privacy.

In this paper, we propose an approach which is complementary to credential-based systems. Our approach allows a service provider to directly issue an anonymous authorization permission without requiring the involvement of a third party. Non-transferability of the permission is achieved by blindly binding the authorization credential to a user pseudonym issued by a third-party infrastructure. By doing this, we succeed in coping with the above discussed opposite needs, in terms of third party involvement, of pseudonymization and authorization functions.

In more detail, the infrastructure in charge of assigning and reverting pseudonyms is relaxed from any supplementary authorization function, and it 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 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 their assigned pseudonyms (possibly multiple). While no single entity involved in a pseudonym assignment is capable of tracing the user, the system retains the possibility of determining 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 their 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 service subscription 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 hijacking, i.e. having a 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 service subscription 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

Digital identity related concepts and privacy concerns have been exhaustively discussed in literature [27,20]. However, for the specific purpose of this paper a user identity (also referred to as "real" identity, or identity certificate, whenever possible ambiguity may occur) consists in a standard X.509v3 certificate U, whose private key is owned by the considered user. The scope of such an identity, i.e. whether the certificate U is locally issued by a specific Service Provider SP at an initial "registration" time, or it is valid across a federation of providers, is irrelevant for what follows.

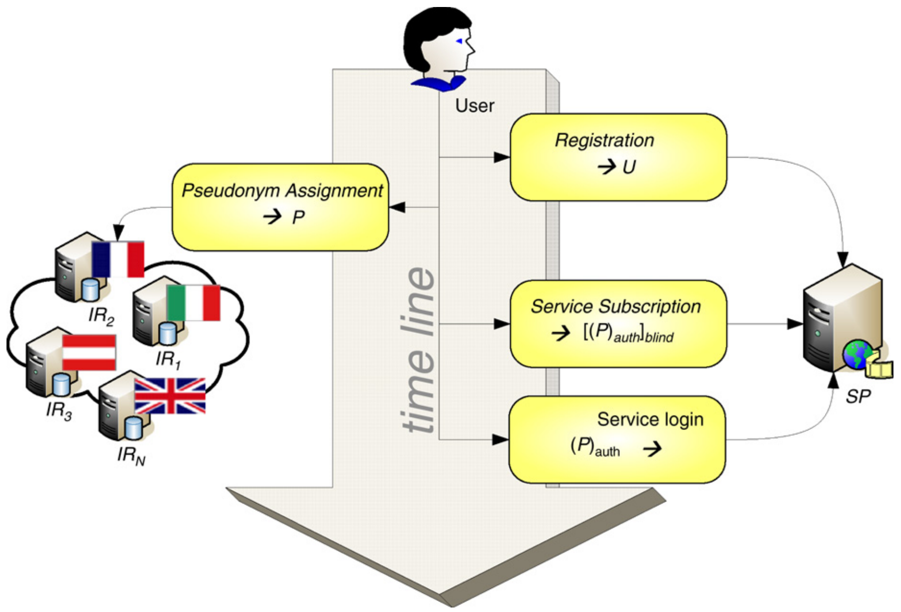


Fig. 1. Summary of the distinct phases: (i) registration — if needed (the user receives an identity certificate U) (ii) pseudonym assignment (the user receives one or more pseudonyms), (iii) service subscription (the pseudonym P is blindly authorized), and (iv) an arbitrary number of accesses to the service/resource (the authorized pseudonym being used as access credential).

The scenario tackled in this paper is that of a user, already in possession of an identity certificate U (e.g., previously registered), initially undergoes a "service subscription" phase. In this phase, the user exhibits their real identity U to the service provider SP and subscribes to a contract for accessing a service/resource S. The type of access is included 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. The service subscription phase may be further based on any supplementary amount of information eventually presented by the user, as well as procedures, such as payment of a flat fare, performed (or documented) at subscription time.

During the service subscription phase, the user will be provided with one or more authorization permissions through which they will be able to access S at later times. An authorization permission consists in having the SP blindly signing, with a signature key specific for each service or resource S, a pseudonym certificate P submitted to the SP (pseudonym authorization). The usage of a blind signature prevents the SP from knowing the pseudonym P used in later access. We anticipate that, for the reasons explained in Section 2.1, an ordinary blind signature cannot be used in this stage, but it is necessary to rely on the signature approach proposed in Section 4.

The pseudonym certificate P blindly signed during the service subscription phase 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). Since the pseudonym certificate P is not exposed to the service provider SP at service subscription time, the verification of its validity (i.e. that it is signed by a trusted IR and satisfies the specific policies imposed by the service S) is delayed until the user will actually access the service. It is worth noting that, since P is blindly signed by the SP through the procedure described in Section 4, and as such not modifiable afterwards, submission of a non valid pseudonym at service subscription time will result in the impossibility of accessing the service later on.

Fig. 1 summarizes the above discussed phases. We remark that the first three phases (registration, pseudonym assignment, and service subscription) are done only once, offline.

It is fundamental to clearly understand the distinction between the validity of the pseudonym and the validity of the authorization permission bound to the pseudonym itself. Pseudonym validity is delegated to the proper operation of the PKI-like third-party infrastructure described in Section 3. A valid pseudonym simply guarantees that the user identity may be traced back from the pseudonym (through the procedure described in Section 3) if regulatory provisions or security reasons require to do so. As such, the above mentioned specific policies which a pseudonym certificate must satisfy are mainly based on legal or contractual obligations which typically depend on the specific type of service considered. The validity of a pseudonym is clearly decoupled from the validity of the authorization permission bound to it. An authorization permission is only locally valid, i.e., in the SP domain, and depends on whether the verification procedure is successful, i.e., the SP can verify the validity of its signature (blindly made during the service subscription phase).

The usage of a pseudonym (an authorized one) for accessing the service makes accesses accountable. If necessary, the user may prevent linkability by asking the SP to authorize more than one pseudonym during the service subscription phase, and then change pseudonyms, among the authorized ones, through different access sessions.

2.1. Why a new blind signature?

If applied in the above described scenario, traditional blind signatures (e.g., that first proposed in [13]) would fail to meet the important requirement of providing non-transferability of an authorized pseudonym. We refer to this problem as “Pseudonym Hijacking”. User U, during the service subscription phase, 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 [24], may be integrated in a comprehensive authorization framework, as the one proposed in [7] which indeed solves these problems, but require deploying 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 [14] 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 service subscription time, an explicit proof of possession of the pseudonym’s private key. This, in fact, would prevent pseudonym hijacking as it would be necessary, for an hijacker, 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 a perfect condition for abusing the pseudonym P': dishonest behavior would be in fact accountable 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 service subscription 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 a user is independent of, and prior to (see Fig. 1), the procedure devised to bind an authorization permission to the pseudonym certificate itself, as well as the actual access to the service. As such, it is in principle executed only once, after the real identity certificate U is issued at registration time. As a consequence, it does not add extra delay and/or computational load to access of the service.

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 registration time.

In addition to the certificate U, still at registration time, the *SP* further releases a “token” certificate T_0. The detailed procedure to release such a token T_0 is not discussed here as it is identical to the handshake between user and Identity Repository discussed next. This certificate is an alias for the real user identity U, and it is generated so that any other entity besides the *SP* 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 a 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.

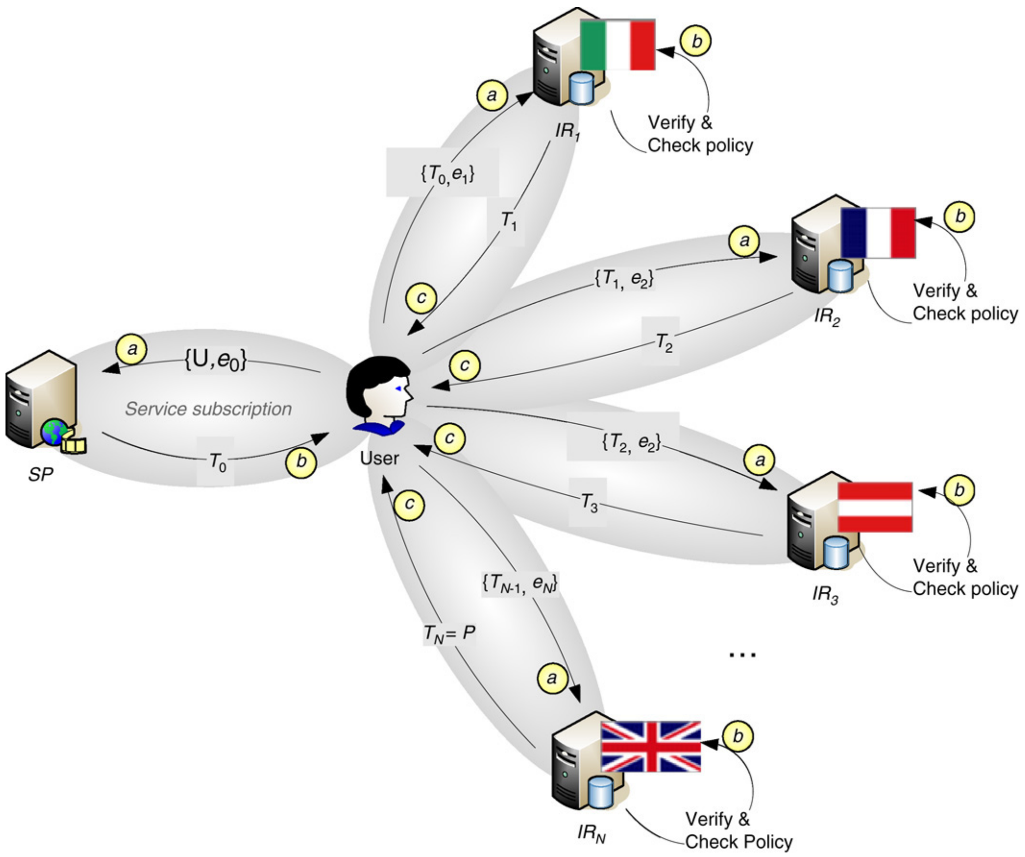


Fig. 2. Pseudonym assignment procedure.

In details, at any generic *i*-th step, the following procedure is adopted:

User $\rightarrow$ IR_i	:	$\{T_{i-1}, e_i\}$	(1)
IR_i	:	verify_signature(T_{i-1})	(2)
IR_i $\leftrightarrow$ User	:	challenge(T_{i-1})	(3)
IR_i	:	policy_check(T_{i-1})	(4)
IR_i $\rightarrow$ User	:	T_i.	(5)

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 decision to generate the pair of public/private keys at the user side is made for both reasons of security (the private key generated is never transmitted) and computational load (the costly computation of the public/private key pair is done by the user and not delegated to the server). 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 — see additional discussion later on in this section) are 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 a final pseudonym *P*, the user simply chooses the last token in this chain (where we stress that such a chain is freely decided by the user). Fig. 2 graphically illustrates the pseudonym assignment procedure in the case of a chain of IR servers.

It is important to remark that the considered handshake is not devised to protect against external observers. Hence, as discussed in Section 5.1, it mandatorily assumes that the communication channel between the user and the IR or SP servers is encrypted and the peers authenticated. This is accomplished by standard lower layer communication security protocols such as IPsec or TLS. We also remark that the handshake employed for obtaining the initial token value T_0 is just a special case. If it is executed at registration time, then there might be no need to actually deliver the identity certificate *U* (already available at the *SP*), but only the public key e_0 to be included in the token certificate T_0.

Thanks to the proposed operation, 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's 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 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, that 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, 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 elaborate problem is the revocation of all the pseudonyms associated with 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 held 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 are not appropriate, as they would permit pseudonym hijacking. To avoid this problem, we introduce a novel blind signature approach. We call "Marked Blind Signature" (MBS), a blind signature mechanism which has the following properties:

- the signature mechanism is devised to include, inside the signed message, a random value R;
- the value R is not known and not forgeable by the signing peer (the server);
- the value R is not known and not forgeable by the peer sending the message to be signed (the user) during the signature protocol, and will be disclosed to the user only at the end of the protocol handshake.

The random value R can be hence considered to be a "mark" of the act of signing (this justifies the name "Marked Blind Signature"). Since it is not forgeable by the user, it can be used in our considered scenario as an ordinary challenge for certificate verification, i.e., over which the user can proof possession of the private key of the pseudonym certificate P. However, since it is not known and forgeable by the server, the server cannot track later on the user from the value R embedded in the signature.

4.1. Marked blind signature

The proposed protocol is developed for RSA blind signatures. 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 $n > n_p$;
- $p = 2p' + 1$ and $q = 2q' + 1$ are safe primes;
- a: DL-strong base [19] for n, generated by the Service Provider.
- $x, s \in \mathbb{Z}_n^*$: random numbers 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 arithmetic modulo n; this can be for instance guaranteed by imposing $x < \sqrt{(}n/2)$, $y < \sqrt{(}n/2)$, and $s < n/2$;
- $B \in \mathbb{Z}_n^*$: 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 a 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.

$$\begin{array}{lll}
\text{SP} \to \text{User}: & a^y & (1) \\
\text{User}: & (a^y)^x a^s = a^R & (2.1) \\
 & \text{sign}(R, P) = H(a^R \| P)^{d_p} \bmod n_p & (2.2) \\
\text{User} \to \text{SP}: & x_1 = B^e x & (3.1) \\
 & x_2 = B^e\,(\text{sign}(R, P) + s) & (3.2) \\
\text{SP}: & x_1 y = B^e xy & (4.1) \\
 & x_2 + x_1 y = B^e\,(\text{sign}(R, P) + s + xy) = B^e\,(\text{sign}(R, P) + R) & (4.2) \\
\text{SP} \to \text{User}: & (x_2 + x_1 y)^d = B\,[\text{sign}(R, P) + R]^d\,. & (5)
\end{array}$$

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 service subscription 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

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

4.3. Discussion

Detailed security analysis of the proposed signature mechanism is outside the goals of the present paper, and it is the subject 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). Obviously, when the user protects her identity through a pseudonym (for example in the service access phase) she authenticates proving the possession of the private key associated with the pseudonym certificate key.

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 granted by the Strong RSA assumption due to the choice of n, and therefore by the anticollision properties of the DL-hashing discussed in [19], 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, s. A drawback of the proposed approach is that the limitations imposed on the values x, y, s yield a non-uniform distribution for the random value R. This property can be, in principle, exploited by an *SP* aiming to trace a specific set of users, by assigning them a specifically forged (e.g. large or small) value y, and then trying to distinguish them from the resulting value R. Assessment of the statistical effectiveness of such an attack is left to future work, as well as a detailed analysis of the trade-off ranges among which the values x, y, s should be optimally chosen (for instance, by increasing the range of choice for s this effect is reduced, but this is traded off with a reduction of the cardinality of x and/or y).

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 a properly chosen hash function.

5. Implementation and deployment issues

In this section, we first discuss supplementary lower-layer security requirements (Section 5.1) and timing requirements (Section 5.2). We then follow up with a description of the actual implementation developed for a web service scenario (Section 5.3).

5.1. Requirements on lower layers

The presentation carried out up to now has focused on design of the pseudonym and authorization system. This occurs at the application level. As such, the security and privacy issues addressed by the proposed approach have been limited to that emerging just because of misbehaving application-layer entities such as an Identity Repository, the Service Provider, or the user itself.

When dealing with actual implementation and deployment throughout an open networking environment, further threats must be mandatorily accounted for, to prevent attacks coming from external entities not involved in the application (such as eavesdroppers or man in the middle attackers). Defense against external attackers requires the usage of network/transport layer security services. We envision the following low-layer security requirements:

(1) *Encryption*. An eavesdropper able to capture all the messages exchanged between the user and the *SP*/*IR*s during the pseudonym assignment procedure described in Section 3 would be able to construct all the chain of sequentially assigned tokens, and hence trivially map the pseudonym ultimately used with the initial user certificate. Therefore encryption of the messages exchanged between the user and the *SP*/*IR*s is a mandatory requirement for the lower-layer support of the application. This can be readily accomplished with the usage of widely deployed network/transport layer communication security protocols, such as IPsec or TLS. The implementation described in Section 5.3 is based on TLS.

(2) *Server/message authentication*. A straightforward consequence of the indeed necessary adoption (for encryption purposes) of network/transport layer communication security protocols is the further provision of server authentication (not explicitly included in the user-*IR* handshake described in Section 3), message authentication, and protection against a variety of attacks including man in the middle attacks, message spoofing, etc. Indeed, note that the user-*IR* handshake described in Section 3 is not specifically devised to provide server authentication inside the handshake, but only requires that the *IR* signature made on the token delivered to the user at the end of the procedure is a valid one (otherwise the token would not be accepted by the next *IR*).

(3) *Protection against linkability attacks based on network addresses*. The anonymization mechanisms provided at the application level would be worthless as long as no adequate mechanisms are also used at the lower layers to protect against linkability attacks. In fact, a same IP address used by the user during both the authorization permission assignment procedure and the actual access to the service would be readily used by the service provider to link the real user's identity (disclosed during the authorization permission assignment) with the pseudonym employed (disclosed during the actual access to the service). Anonymization proxies or Mix networks such as Tor [16] are solutions for protecting against these linkability attacks.

5.2. Timing issues

Any system-level operation which relies on distinct procedures executed at different temporal phases is at stake of linkability attacks which try to extract information from the actual time in which a phase takes place. In our specific case, as illustrated in Fig. 1, there are up to four phases if we assume that the user identity certificate U is provided by the *SP* at an initial registration phase.

In what follows, the only threat considered is the possibility that the *SP* uses the time at which these phases are executed in order to attempt to link the user identity U with the pseudonym P. Attackers other than the *SP* are not considered, since, as discussed in the previous section, they are assumed not capable of eavesdropping the delivered messages.

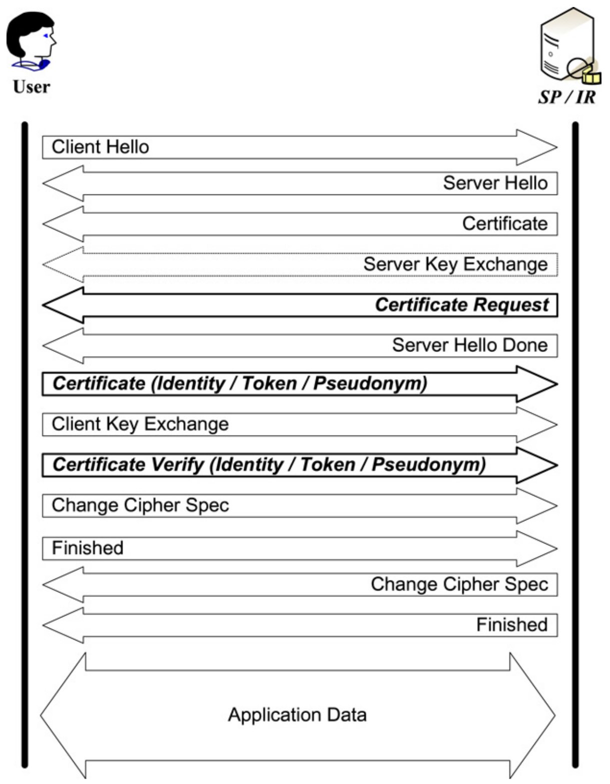


Fig. 3. TLS handshake procedure — the standard TLS handshake messages 5, 7, and 9, in bold, are that specifically used by our implementation to allow user identity, token or pseudonym verification, depending on the specific phase considered.

If the pseudonym *P* is ultimately assigned by an external *IR* (as in the example illustrated in Fig. 2), the registration phase and the pseudonym assignment phase do not raise linkability issues, and hence they can be developed in succession. Note however that, in principle, the *SP* is a certification authority that can be made part of the pseudonym assignment chain: if, in a specific application scenario, it is considered convenient to have the final pseudonym *P* specifically issued by the *SP* instead of generically issued by an *IR*, then timing considerations between these two phases would become important.

Similarly, the service subscription (authorization) phase does not raise any linkability issue, since in this phase the user identity *U* is fully disclosed. As such, it can be executed immediately after the previous two phases.

The only critical case is the time elapsing between the authorization phase and the first service access phase. Having these two phases temporally close each other would allow the *SP* to correlate the user identity disclosed in the authorization phase with the pseudonym disclosed in the service access phase.

5.3. Web service implementation

The SPARTA pseudonym and authorization system described in this paper has been developed in the frame of the IST project Discreet (www.ist-discreet.org). An open software implementation of SPARTA has been released under GPL Licence. it can be downloaded from the Discreet project website together with an application demo. The current implementation is based on X.509v3 standard certificates, with 2048 bit RSA signatures.

Besides the implementation of the specific procedures described in the paper, a further effort has been placed in the integration of the envisioned approach in a web service scenario, using TLS as lower layer communication security protocol. In detail, our implementation takes advantage of the user certificate verification procedure already available in TLS to verify the validity of an user certificate. Fig. 3 reviews the 13 messages exchanged by a TLS handshake. We configured both the *IR* as well as the *SP* server to explicitly request an user certificate (message 5 in the TLS handshake; this message is standard and supported by every TLS implementation, although it is optional and typically not used in ordinary TLS operation which does not require user-side authentication, e.g., protected access to a web server). Depending on the specific considered phase, the user includes in message 7 of the TLS handshake a certificate, and specifically: (i) in the *i*-th communication with an IR, the user browser sends the token certificate T_{i-1}, (ii) in the service subscription phase, the user identifies with the real identity certificate U, and (iii) in the service access phase the user relies on the pseudonym certificate. The user certificate is then verified through the standard certificate-verify message 9 of the TLS handshake.

Note that in both cases of service subscription and service access, the user certificate verification procedure embedded in TLS simply verifies that the actual certificate used to access the server, but both the issuing of the authorization permission (in the service subscription phase) as well as the authorization credential verification (in the service access phase) must be provided at the application layer by exchanging the relevant information inside the TLS session. Therefore, the submission of the authorization credential is prompted, after the establishment of the TLS connection, by a web portal application we have developed. A credential verification tool is then executed: it is invoked as a Unix system call by the web portal application. Using X.509v3 standard for digital certificates, a user is able to store their pseudonym and their private key inside their browser using PKCS#12 (Personal Information Exchange Syntax Standard) and can protect their private key within the embedded browser key-ring.

Rather, with specific reference to the token/pseudonym assignment handshake described in Section 3, the integration of the user token verification inside TLS allows to simplify the application-layer procedure described before. Specifically, since the user token T_{i-1} is now submitted to the server, as well as verified, through the TLS handshake, the following simplifications hold:

- At step (1) the user may submit only the public key e_i to be included in the subsequent token;
- Steps (2) and (3) are no more necessary, as the certificate is verified inside TLS, and hence they may be skipped;

Concerning the actual implementation of the specific application-layer procedures, as well as the primitives for efficient certificates generation and management, the functionalities needed by every component of the system (user, *SP* server, *IR* server), have been organized into a SPARTA Software Library based on the OpenSSL Crypto library [30]. Specific functions have been further implemented for the Marked Blind Signature, by extending the sub-libraries provided by OpenSSL. As a consequence, rather than implementing each different server as a separate software project, we have a single "Multi Purpose Server" (*MPS*) which can be configured either as *SP* or as *IR*. 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 [29], which is well known by the open software community and guarantees good performance 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, authorization and verification procedures. We choose to develop a command-line tool, for ease of integrability in other softwares, with a Graphical User Interface 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.

The SPARTA library further specifies a convenient common message format for all the SPARTA messages. Specifically, we have decided to employ an Attribute-Value pair format similar to the one used in Radius/Diameter. The message is therefore composed of 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 application-layer message types have been specified (client hello, server hello, delivery of public keys, delivery of authorization credentials, key requests, etc) for the authorization and verification procedure.

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 the latter, we have measured the time needed to generate a 2048 bit key pair on an entry-level laptop (e.g. Intel Centrino 1.6 GHz with 512 MB RAM). This time results in about 1 s, 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.

6. Conclusion

This paper addresses the problem of how to allow a service provider to issue anonymous non-transferable authorization permissions without involving third parties in the authorization framework. The proposed solution relies on clear decoupling between pseudonyms and authorization permissions. Pseudonyms are used only to trace back the real identity of a user upon dishonest behavior. As such, they are relieved from any authorization task, and can be managed through a third-party infrastructure. This is a fully distributed user-centric PKI-like infrastructure, which provides the benefit that pseudonym reversion is made possible only through the joint cooperation of multiple, user-chosen, infrastructure entities. Conversely, authorization permissions are bound to an user pseudonym through a distinct procedure based on a novel blind signature approach which, unlike traditional blind signature solutions, further allows one to blindly verify the possession of the pseudonym's private key (and hence accomplish the authorization permission non-transferability requirement). A proof-of-concept implementation has been developed in a web service scenario. The implementation is tightly integrated with the Transport Layer Security protocol, and applies its user certificate verification built-in primitives for user, token, and pseudonym certificate verification purposes.

Our work opens a number of future research directions. A first research challenge is a detailed specification of the policies (and their semantic) used to manage the pseudonym assignment infrastructure, to include regulatory requirements as formal conditions under which a pseudonym should be reverted. A second goal is to assess the security of the proposed marked blind signature, especially against attacks devised to exploit the non-uniform probability distribution function of the random mark embedded in the signature for attempting to trace the user. We believe that the concept of marked blind signature, first presented here, may stimulate several improvements and alternative approaches. A third goal is to integrate our approach with a third-party-based anonymous credential system: as explained in the paper, our approach appears complementary to systems where anonymous credentials are naturally issues by third-party entities. Finally, a further research challenge is to extend the portability and applicability of the current implementation. This can be accomplished, on one side, by integrating the SPARTA framework in an existing AAA protocol such as Diameter, and on the other side by extending the SPARTA framework to support anonymous payments and real-time accounting.

References

[1] S.A. Brands, Rethinking Public Key Infrastructures and Digital Certificates, MIT Press, 2000.
[2] V. Benjumea, J. Lopez, J.M. Troya, Anonymous attribute certificates based on traceable signatures, Internet Res. 16 (2) (2006) 120–139.
[3] J. Camenisch, E. Van Herreweghen, Design and implementation of the idemix anonymous credential system, ACM Computer and Communication Security (2002).
[4] D. Chaum, Security without identification: Transaction systems to make big brother obsolete, Communications of the ACM 28 (10) (1985) 1030–1044.
[5] C. Andersson, J. Camenisch, S. Crane, S. Fischer-Hübner, R. Leenes, S. Pearsson, J.S. Petterson, D. Sommer, Trust in PRIME, in: 5th IEEE Int. Symposium on Signal Processing and Information Technology, Athens, Greece, December 2005.
[6] A. Bender, J. Katz, R. Morselli, Ring signatures: Stronger definitions, and constructions without random oracles, in: Theory of Cryptography, TCC 2006, in: Lecture Notes in Computer Science, vol. 3876, Springer-Verlag, 2006, pp. 60–79.
[7] V. Benjumea, J. Lopez, J.A. Montenegro, J.M. Troya, A first approach to provide anonymity in attribute certificates, in: 2004 International Workshop on Practice and Theory in Public Key Cryptography, in: Lecture Notes in Computer Science, vol. 2947, Springer-Verlag, 2004, pp. 402–415.
[8] S. Brands, F. Légaré, Digital identity management based on digital credentials, GI Jahrestagung, 2002.
[9] L. Bussard, R. Molva, Establishing trust with privacy, in: Security Protocols Workshop 2004, in: Lecture Notes in Computer Science, vol. 3957, Springer-Verlag, 2004, pp. 199–209.
[10] J. Camenisch, A. Lysyanskaya, An identity escrow scheme with appointed verifiers, in: CRYPTO 2001, in: Lecture Notes in Computer Science, vol. 2139, Springer-Verlag, 2001, pp. 388–407.
[11] J. Camenisch, A. Lysyanskaya, Efficient non-transferable anonymous multi-show credential system with optional anonymity revocation, in: Extended Abstract in: Advances in Cryptology — Eurocrypt, 2001.
[12] J. Camenisch, A. Lysyanskaya, A signature scheme for efficient protocols, in: Third Conference on Security in Communication Networks, 2002.
[13] D. Chaum, Blind signatures for untraceable payments, in: Advances in Cryptology — Crypto'82, Springer-Verlag, 1983, pp. 199–203.
[14] D. Chaum, E. van Heyst, Group signatures, in: Advances in Cryptology EUROCRYPT 91, in: Lecture Notes in Computer Science, vol. 547, Springer-Verlag, 1991, pp. 257–265.
[15] D. Cvrcek, V. Matyás Jr., Pseudonymity in the light of evidence-based trust, in: Security Protocols Workshop 2004, in: Lecture Notes in Computer Science, vol. 3957, Springer-Verlag, 2004, pp. 267–274.
[16] R. Dingledine, N. Mathewson, P. Syverson, Tor: The Second-Generation Onion Router, in: Proc. of the 13th USENIX Security Symposium, August 2004.
[17] Y. Dodis, A. Kiayias, A. Nicolosi, V Shoup, Anonymous identification in ad hoc groups, in: EUROCRYPT 2004, in: Lecture Notes in Computer Science, vol. 3027, Springer-Verlag, 2004, pp. 609–626.
[18] U. Feige, A. Fiat, A. Shamir, Zero knowledge proofs of identity, in: Proc. 19th ACM Symp. on Theory of Computing, May 1987, pp. 210–217.
[19] J.K. Gibson, Discrete logarithm hash function that is collision free and one way, IEEE Proceedings 138 (6) (1991) 407–410.
[20] D. Gollmann, Identity and location, in: Security Protocols Workshop 2004, in: Lecture Notes in Computer Science, vol. 3957, Springer-Verlag, 2004, pp. 246–250.
[21] J. Kilian, E. Petrank, Identity escrow, in: CRYPTO 1998, in: Lecture Notes in Computer Science, vol. 1462, Springer-Verlag, 1998, pp. 169–185.
[22] A. Lysyanskaya, R.L. Rivest, A. Sahai, S. Wolf, Pseudonym systems, in: Selected Areas in Cryptography, in: Lecture Notes in Computer Science, vol. 1758, Springer-Verlag, 1999, pp. 184–199.
[23] R. Rivest, A. Shamir, Y. Tauman, How to leak a secret, in: ASIACRYPT 2001, in: Lecture Notes in Computer Science, vol. 2248, Springer-Verlag, 2001, pp. 552–565.
[24] M.A. Stadler, J.M. Piveteau, J.L. Camenisch, Fair blind signatures, in: Advances in Cryptology EUROCRYPT95, in: Lecture Notes in Computer Science, vol. 921, Springer-Verlag, 1995, pp. 209–219.
[25] 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.
[26] P. Calhoun, J. Loughney, E. Guttman, G. Zorn, J. Arkko, Diameter Base Protocol, IETF RFC 3588, September 2003.
[27] T. Nabeth, M. Hildebrandt, D 2.1: Inventory of topics and clusters FIDIS Project Deliverable, September 2005.
[28] C. Rigney, S. Willens, A. Rubens, W. Simpson, Remote Authentication Dial In User Service, RADIUS, IETF RFC 2865, June 2000.
[29] MySQL project homepage. http://www.mysql.org.
[30] OpenSSL project homepage. http://www.openssl.org.