available at www.sciencedirect.com

ScienceDirect

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

Computers & Security

PrivaKERB: A user privacy framework for Kerberos

F. Pereniguez [a,*], R. Marin-Lopez [a], G. Kambourakis [b], S. Gritzalis [b], A.F. Gomez [a]

[a] *Dept. Information and Communications Engineering (DIIC), University of Murcia, Facultad de Informatica, Campus de Espinardo S/N, 30100 Murcia, Spain*

[b] *Laboratory of Information and Communication Systems Security, Department of Information and Communication Systems Engineering, University of the Aegean, Samos GR-83200, Greece*

ARTICLE INFO

Article history:
Received 28 October 2010
Received in revised form
15 April 2011
Accepted 22 April 2011

Keywords:
Identity
Kerberos
Network Federation
Privacy
Security

ABSTRACT

Kerberos is one of the most well-respected and widely used authentication protocols in open and insecure networks. It is envisaged that its impact will increase as it comprises a reliable and scalable solution to support authentication and secure service acquisition in the *Next Generation Networks* (NGN) era. This means however that security and privacy issues related to the protocol itself must be carefully considered. This paper proposes a novel two-level privacy framework, namely PrivaKERB, to address user privacy in Kerberos. Our solution offers two privacy levels to cope with user anonymity and service access untraceability. We detail how these modes operate in preserving user privacy in both single-realm and cross-realm scenarios. By using the extensibility mechanisms already available in Kerberos, PrivaKERB does not change the semantics of messages and enables future implementations to maintain interoperability. We also evaluate our solution in terms of service time and resource utilization. The results show that PrivaKERB is a lightweight solution imposing negligible overhead in both the participating entities and network.

1. Introduction

Although Kerberos (Neuman et al., 2005a) was designed several years ago, its evolution has continued unabated in recent years with the development of new features, which increase the security of the protocol or address new needs (Hartman and Zhu, 2010; Zhu et al., 2010; Josefsson, 2010; MIT Kerberos Distribution). Nowadays, organizations like the IETF (Table 1)[1] (*Internet Engineering Task Force*) Kerberos Working Group (Kerberos WG) or the Kerberos Consortium (MIT Kerberos Consortium) are leading the improvement of the protocol in order to constitute Kerberos as the universal solution to the distributed authentication for access services. Nevertheless, researchers agree that Kerberos must be able to face different security threats that arise in the next generation of mobile communications, generally called *Next Generation Networks* (NGNs) (Kerberos WG).

Within these networks, privacy is a key aspect (Apostolopoulos et al., 2010; Pfitzmann and Hansen, 2010) and represents a complex problem that can affect different network layers (Chen et al., 2008; Bowen and Martin, 2007; Bagnulo et al., 2007; Christin et al., 2010; Cardoso et al., 2007; Karopoulos et al., 2010). So, for example, a user can be traced by the use of the same IP or link-layer address when accessing a service (e.g., (Chen et al., 2008; Bowen and Martin, 2007; Bagnulo et al., 2007)). The same happens at application level (e.g., (Christin et al., 2010; Karopoulos et al., 2010)), where many network applications need to take care in preserving

* *Corresponding author*. Tel.: +34 868 88 78 82; fax: +34 868 88 41 51.
E-mail addresses: pereniguez@um.es (F. Pereniguez), rafa@um.es (R. Marin-Lopez), gkamb@aegean.gr (G. Kambourakis), sgritz@aegean.gr (S. Gritzalis), skarmeta@um.es (A.F. Gomez).

[1] A list of acronyms used in this paper is summarized in Table 1.

0167-4048/$ – see front matter © 2011 Elsevier Ltd. All rights reserved.
doi:10.1016/j.cose.2011.04.001

Table 1 – List of acronyms.

Acronym	Meaning
KDC	Key Distribution Center
AS	Authentication Server
TGS	Ticket Granting Server
TGT	Ticket Granting Ticket
ST	Service Ticket
KRB_AS_REQ/REP exchange	Kerberos Authentication Server Request/Response exchange. In short, AS exchange.
KRB_TGS_REQ/REP exchange	Kerberos Ticket Granting Server Request/Response exchange. In short, *TGS exchange*.
KRB_AP_REQ/REP exchange	Kerberos Application Request/ Response exchange. In short, *AP exchange*.
padata	pre-authentication data
PKI	Public Key Infrastructure
PKINIT	Public Key Cryptography for Initial Authentication in Kerberos
TLS	Transport Layer Security
IETF	Internet Engineering Task Force
RFC	Request For Comments
API	Application Programming Interface

user privacy through application-specific solutions. In this regard, Kerberos, as application level protocol, must pay attention to user privacy in order to achieve a successful deployment in NGNs. Nevertheless, the standard Kerberos (Neuman et al., 2005a; Kim, April 2009; Brennen, 2004) suffers from the lack of a mechanism to preserve the user privacy. Indeed, Kerberos identifies the different participant entities through the so-called Kerberos *principal identifiers*, which have the form of "principal_name@realm_name". The *principal_name* part unequivocally identifies an individual user in the administrative realm specified in the *realm_name* part. However, these principal identifiers are transmitted in clear-text in Kerberos.

Exposure of this sensitive information to unauthorized third parties allows for several malicious acts that clearly violate the user's private sphere (Ohm, 2009; Tene, 2010; Sweeney, 2000; Golle, 2006). The most obvious is that an eavesdropper is able to obtain access to the user's real identity and observe which services are being accessed, so violating the principle of *user anonymity* (Pfitzmann and Hansen, 2010). In the long term, when this kind of information is systematically collected, the user can be profiled and sensitive information (e.g., preferred services) may be inferred (Ohm, 2009; Tene, 2010). Secondly, even when the user's identity remains anonymous, an eavesdropper may obtain some general information about behavioral patterns of service access of specific anonymous users in the network (*service access untraceability*) (King and Jessen, 2010; Tene, 2010).

In this paper, we study in depth and solve these two issues (*user anonymity* and *service access untraceability*) in Kerberos. In particular, we develop a novel two-level privacy architecture named *PrivaKERB* that preserves the privacy of the user during activity with Kerberos. Our architecture provides a flexible solution, which represents a trade-off between easy deployment and level of privacy achieved. This is possible thanks to the design of two different levels of privacy: *level 1* which provides *user anonymity* through pseudonyms, so that an eavesdropper cannot determine the real identity of the user; and *level 2* where, apart from user anonymity, service access untraceability is assured, so that an eavesdropper cannot collect information about per-user behavioral access patterns to different services (Ohm, 2009; Tene, 2010; Sweeney, 2000; Golle, 2006; King and Jessen, 2010; Narayanan and Shmatikov, 2008). Moreover, this flexible design allows a privacy level to be selected that satisfies better the needs in terms of deployment cost and type of privacy provided in the network. As we will show, the solution is even applicable in cross-realm scenarios where the user requests access to services located in foreign realms. Furthermore, since we have employed the extensibility mechanisms available in Kerberos (Neuman et al., 2005a), no modifications to the protocol itself are required. Unlike other solutions (Hartman and Zhu, 2010; Zhu et al., 2010; Josefsson, 2010; Medvinsky et al., 1998; Zhu and Tung, 2006) for preserving Kerberos privacy (which we will discuss thoroughly in Section 3), the use of pseudonyms allows, in certain controlled situations, trusted parties to perform essential operations (e.g., to charge the user for service access) which may demand some association with the specific user without revealing its real identity (Benjumea et al., 2008). Finally, through an implemented prototype in different scenarios, we demonstrate that the overhead imposed by the privacy extensions is almost negligible in comparison with the standard Kerberos.

The remainder of the paper is organized as follows: the next section describes details of Kerberos operation and the specific privacy problems that remain unsolved. Section 3 shows relevant related work. Section 4 introduces and details our privacy framework, namely PrivaKERB, describing the different operation modes which correspond to different privacy levels. In Section 5, we use a real implementation testbed to measure the additional overhead required by our privacy extensions in comparison with standard Kerberos. Finally, Section 6 concludes the paper and gives some directions for future work.

2. Background: Kerberos protocol privacy issues

2.1. *Protocol overview*

Kerberos is a secure three-party protocol for authentication and key management based on shared secret key cryptography (Boyd and Mathuria, 2003). The base protocol is fully specified and described in the IETF RFC 4120 (Neuman et al., 2005a), which we overview in the following.

The protocol provides a single sign-on platform through the so-called *tickets*. A ticket is a piece of encrypted and integrity protected information that allows a user to be authenticated without requiring the user to re-enter its password. Kerberos messages are exchanged between three types of entities (called *principals*): a *client* that represents a user[2] willing to access a specific service, an *application server*

[2] In this paper we use the terms *user* and *client* indistinctly.

(hereafter *service*) providing a specific service, and a *Key Distribution Center* (KDC) in charge of authenticating users and distributing tickets within a specific *realm*. At the same time, the KDC is integrated by two servers: the *Authentication Server* (AS) and the *Ticket Granting Server* (TGS). Kerberos assumes that both the client and service have a pre-established trust relationship with the AS and TGS, respectively. Traditionally, the trust relationship between AS and client is defined by a shared *secret key* which is derived from the client's password.

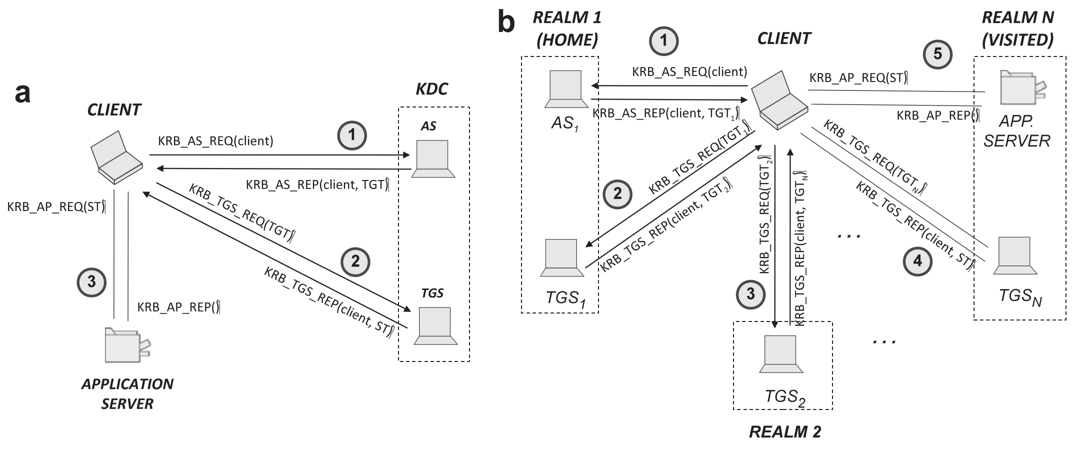
Kerberos communication consists of three different exchanges (see Fig. 1(a)). Initially, by means of the *AS exchange* (KRB_AS_REQ/REP) *(1)*, the client contacts the AS module in the KDC (*AS/KDC*) to request a *Ticket Granting Ticket* (TGT). The TGT is a special ticket, protected with a secret key shared between the AS and TGS, and is used by the client to request *Service Tickets* (ST). When the KDC receives the KRB_AS_REQ message, it generates a *TGS session key* (shared between client and TGS), which is included in the TGT. Together with other ticket-related information (e.g., validity period), this key is also sent in the KRB_AS_REP message encrypted using the client's secret key. Both messages contain in cleartext which specific user (client's identifier) is requesting a TGT valid for the TGS/KDC.

Once the client acquires a TGT, it is ready to solicit STs to access different services through the *TGS exchange* (KRB_TGS_REQ/REP) *(2)*. To do so, the client sends a KRB_TGS_REQ message to the TGS module in the KDC (*TGS/KDC*). In addition to the service's identity (in cleartext), this message contains the TGT and an authenticator generated by using the TGS session key included in the TGT. This authenticator allows the TGS to verify the client's identity. After successful verification of both the TGT and the authenticator, the TGS generates an ST (protected with the secret key shared between TGS and service) containing the *service session key* (shared between client and service). Again, this key is also delivered to the client by means of the KRB_TGS_REP, which also transports information (protected with the TGS session key) required by the client to use the ST. Like the KRB_AS_REP, this message contains both the client's and service identifier in cleartext. Finally, through the *AP exchange* (KRB_AP_REQ/REP) *(3)*, the client authenticates itself against the service (Neuman et al., 2005a). In the KRB_AP_REQ, the client sends the previously obtained ST, together with an authenticator computed by using the service session key that enables the service to verify that the client owns such a key. Optionally, when mutual authentication is required, the service responds with a KRB_AP_REP message.

When the client requests access to a service in a visited realm (different from its home realm), Kerberos defines a special operation mode called *cross-realm* (see Fig. 1(b)). Cross-realm authentication (Neuman et al., 2005a) allows a client from one organization to be authenticated in another, thanks to the definition of trust relationships between TGS/KDCs from different realms. The process starts when the client engages in a KRB_AS_REQ/REP exchange *(1)* with its home AS/KDC (where the client is really registered) in order to obtain a *single-realm* TGT. This TGT is used in a posterior KRB_TGS_REQ/REP exchange with the home TGS/KDC where the client requests a special ticket, called *cross-realm* TGT. This type of ticket can be used to communicate with a TGS located in a different realm thanks to the multi-realm KDC architecture deployed *(2)*. By repeating this process *(3)*, the client follows the authentication path from the home to the *visited* realm, where the service is deployed. Finally, the TGS placed in the visited realm delivers an ST *(4)* used by the client to authenticate itself to the service *(5)*.

2.2. Privacy aspects

As already pointed out, Kerberos assigns an identifier (or *identity*) to the different entities that participate in the protocol operation. According to the format specified in (Neuman et al., 2005a), these identities are in the form "principal_name@realm_name". The first part is a multi-component string (each component is separated by the "/" symbol) that unequivocally identifies an entity in the realm specified by the second part. For example, assuming the realm is UM.ES, *peter@UM.ES* and *printer/server.um.es@UM.ES* are valid to identify a user and a service, respectively. Nevertheless, principal identifiers are transmitted in cleartext during Kerberos execution. On the one hand, tickets contain in cleartext the service identity for which the ticket has been generated. On the other, Kerberos messages also define fields in cleartext to convey the identities of the principals (Neuman et al., 2005a). More specifically, the KRB_AS_REQ message defines a client name (*cname*) field



Kerberos standard signalling Kerberos cross-realm operation

Fig. 1 – Kerberos signaling.

that contains in cleartext the identity of the client requesting a TGT for the TGS service specified in the service named (*sname)* field. The latter is also included in the KRB_TGS_REQ to specify the identity of the service for which an ST is being solicited. Similarly, both the KRB_AS_REP and KRB_TGS_REP messages include the identity of the client (*cname* field) for whom a ticket has been issued. Therefore, an eavesdropper can easily obtain client's real identity and observe which services are being accessed, thus violating the principle of *user anonymity* (Pfitzmann and Hansen, 2010).

It turns out that, even in the hypothetical case that the client's real identity may remain unknown, an eavesdropper may obtain some general information about behavioral patterns of service access of specific anonymous users in the network. The reason is that, according to Kerberos specification (Neuman et al., 2005a), a client typically uses the same TGT to access multiple services (by requesting the corresponding STs). As a consequence, an eavesdropper can determine that the same (anonymous) client is accessing these services just by tracing the use of the same TGT, therefore not accomplishing *service access untraceability* (Tene, 2010; King and Jessen, 2010).

Thus, our goal is to design a lightweight solution to handle these two specific problems in Kerberos: *user anonymity* and *service access untraceability*. In the following sections, we provide some important related work and the details of our contribution.

3. Related work

The provision of client privacy in Kerberos is an aspect that has not been completely ignored by researchers. One of the earliest works in this area is found in (Medvinsky et al., 1998), where the concept of *anonymous ticket* is introduced. An anonymous ticket is a regular Kerberos ticket which does not contain any information about the client's real identity. Thus, when a client uses an anonymous ticket to access a service, the client's real identity is not revealed either to the service or to eavesdroppers. However, the deficiencies of this solution stem from the methods proposed to deliver anonymous tickets to the clients. The first method assumes that the client does not share any secret with the KDC and proposes the use of *Public Key Cryptography for Initial Authentication in Kerberos* (PKINIT) (Zhu and Tung, 2006) in order to securely deliver the session key associated to the anonymous ticket. Nevertheless, this method requires the existence of a *Public Key Infrastructure* (PKI) that, unfortunately, may not always be available. The second method only allows registered users within a Kerberos realm to obtain an anonymous ticket. More precisely, the client obtains an anonymous ticket by performing a standard AS exchange with its real identity. Thus, the real client's identity and the distributed anonymous ticket are visible for an eavesdropper during the AS exchange, which allows the eavesdropper to establish a relationship between them. Consequently, an eavesdropper can easily infer the client's real identity by just tracing the use of the specific anonymous ticket.

On the basis of this initial work, the IETF Kerberos Working Group defines a solution (Zhu et al., 2010), also based on the use of anonymous tickets, to completely hide the client's identity from KDCs and observers. This work mainly focuses on the definition of the Kerberos protocol extensions required to implement the anonymity solution presented in (Medvinsky et al., 1998). For example, the working group defines the well-known anonymous principal name as an identifier having a special meaning other than identifying a particular user. Similarly, new flags are proposed to distinguish anonymous tickets from standard ones and to allow users to solicit an anonymous ticket. Nevertheless, since the technical solution defined by the working group remains the same as proposed in (Medvinsky et al., 1998), the same deficiencies previously mentioned for each anonymous ticket acquisition method apply here.

Another way to offer client privacy in Kerberos is to transmit messages through a protected TLS tunnel (Josefsson, 2010). Once the TLS tunnel is established between two entities (e.g., client and KDC), Kerberos exchanges can be performed within the tunnel. Thus, sensitive information such as the client's real identity is not revealed to eavesdroppers. However, this solution requires the client to successfully establish a TLS tunnel with any entity (KDCs and services) with which it desires to exchange Kerberos messages. Excluding the overhead that a full TLS handshake may impose (Apostolopoulos et al., 1999), this requirement is especially problematic during cross-realm operation where the client has to establish TLS tunnels with intermediary KDCs. Since we cannot assume a pre-established trust relationship between the client and intermediary realms, a multi-domain PKI infrastructure (Shimaoka et al., 2008) is required to assist a typical certificate-based TLS authentication, increasing the deployment cost of the solution. Regardless of this inconvenience, it is important to note that this solution is just shifting the privacy problem to the TLS protocol where, during the authentication phase prior to the TLS tunnel establishment, it is necessary to define a mechanism to preserve the client's privacy.

The *Generalized Framework for Kerberos Pre-authentication* (Hartman and Zhu, 2010) proposes an architecture to assist the design of authentication mechanisms which allow the client to be authenticated before granting a ticket. An additional objective of this solution is to enhance the security of the protocol by protecting information that Kerberos sends in cleartext. In particular, the client's identity is confidentiality protected in the messages sent from the client to the KDC (KRB_AS_REQ and KRB_TGS_REQ). Using this improvement in conjunction with the extensions defined in Zhu et al. (2010), the solution allows a client to solicit an anonymous ticket without revealing its real identity. Nevertheless, the pre-authentication framework solution presents some drawbacks related with the so-called *armor TGT*, which clients must obtain before starting to use the pre-authentication extensions with a specific KDC. In particular, the three methods proposed to obtain such special TGT present some deficiencies. First, a client can perform a standard AS exchange using its real identity to request an armor TGT. Nevertheless, it allows eavesdroppers to relate the acquired armor TGT with the client's identity. Thus, when the client re-uses the armor TGT to request an anonymous ticket, an observer can infer the real identity associated to the anonymous ticket. Secondly, assuming the KDC owns a valid certificate, the user can use anonymous PKINIT (Zhu and Tung, 2006) to obtain an armor TGT. When a PKI infrastructure is not available, a final method proposes to acquire the armor TGT using anonymous PKINIT without KDC authentication. Nevertheless, as recognized by authors of the work, this option third is vulnerable to man-in-the-middle attacks.

Thus, to the best of authors' knowledge, none of these proposals define a solution which at the same time allows: a) the client to remain anonymous and untraceable from eavesdroppers; b) to identify the client in the specific cases that important processes such as accounting and charging operations require it; and c) to eliminate the need of other infrastructures different than the Kerberos ones, such as a PKI. As seen, proposals such as Hartman and Zhu (2010), Zhu et al. (2010) and Medvinsky et al. (1998) aim for the complete anonymity of the client, which makes it unfeasible to perform some vital operations like the accounting process performed by trusted entities. Conversely, the work in Josefsson (2010) requires the support of a PKI infrastructure to operate that makes its deployment harder. In this way, the presented privacy framework for Kerberos is a novel approach which gives flexibility between the level of privacy provided to the user and reduces the cost required to deploy the solution. Furthermore, introducing an almost negligible penalization to the protocol, our proposal achieves an effective privacy protection of the client without affecting other vital processes where the client needs to be identified such as accounting and charging operations. Next, we provide details about our proposal.

4. Proposed privacy framework

4.1. *General overview*

Motivated by the privacy issues described in Section 2.2, in this work we focus on hiding client's real identity from unauthorized parties and reduce the information that an eavesdropper is able to collect from looking at Kerberos exchanges about service access behavior of specific (anonymous) users. Thus, our solution must fulfill two important requirements:

1. *User anonymity*. During its activity with Kerberos, a client must remain anonymous not only to eavesdroppers but also to any entity in the visited realm. Only the KDC in the home realm will have access to the client's real identity.
2. *Service access untraceability*. Eavesdroppers must be unable to trace the different services accessed by a specific *anonymous* user. This information will be only known by the specific and trusted KDC controlling the set of services accessed by the client.

As we observe, these two requirements are interrelated since service access untraceability assumes that user anonymity is ensured (Pfitzmann and Hansen, 2010). Instead of proposing a single solution supporting the highest level of privacy (*service access untraceability*), our objective is to design a flexible framework which allows to select which specific requirement must be accomplished. The reason of providing this flexibility is that, as we will analyze, it is expected that a higher level of privacy shall require more extensions and implementation efforts. Therefore, if only user anonymity is required/implemented within a realm there is no need to implement any extension related with service access traceability. Moreover, the selection of a specific level of privacy can also consider factors such as deployment cost, since a higher level of privacy will imply more changes in existing Kerberos infrastructures.

To achieve this flexibility our solution therefore works in two modes of operation. Each one, which is identified with a specific privacy level, satisfies one of the previously described requirement. More specifically, we distinguish:

- *Level 1*. Lowest privacy level providing only user anonymity (*req.1*).
- *Level 2*. Higher privacy level providing user anonymity and service access untraceability (*req.2*).

These privacy levels have been designed considering the following guidelines. Firstly, we favor the interoperability of the solution with current implementations of Kerberos (e.g., Massachusetts Institute of Technology – *MIT* – Kerberos Implementation (MIT Kerberos Distribution)). In other words, the proposed solution respects the standard Kerberos specification (Neuman et al., 2005a), and modifications to the protocol (e.g., definition of new message fields) are not required. Indeed, we employ the extensibility mechanisms available in the standard Kerberos (Neuman et al., 2005a): a) definition of new flags in the messages; b) design of new pre-authentication data (hereafter *padata*); and c) new authorization elements (Neuman et al., 2005a) which can carry any valuable information for new applications, such as our privacy extensions. Using these mechanisms, our solution integrates smoothly with current Kerberos implementations without privacy support. Secondly, for the solution we demand the support of the cross-realm operation. This aspect becomes important in the context of NGNs, where users frequently change their point of attachment to the network and solicit access to services located in foreign realms.

In the following sections, we elaborate each of the designed levels and detail how to accomplish the two requirements with our proposal. For the sake of clarity, we present first the notation we employ to describe our solution:

- *name$_i$@realm*: ith principal name employed by a user in the specified Kerberos realm.
- $\mathrm{TGT}^i_X[Y]$: ith TGT for KDC X that contains the information Y confidentiality and integrity protected.
- $\mathrm{ST}_X[Y]$: service ticket for service X that contains the information Y protected providing confidentiality and integrity.
- N_i: ith random number generated by the client.
- PA–NAME(Y): a padata type named NAME containing the information Y.
- AD(X,Y,...): pieces of information X,Y,... are transported in the authorization-data (AD) field of a ticket by using different authorization data elements.
- FlagX: a flag named X.

4.2. *Level 1: user anonymity*

4.2.1. *Preliminaries*

Level 1 is the lowest level of privacy that we have designed to accomplish requirement 1. That is, level 1 keeps the user anonymous during its activity with Kerberos. One possibility is to use an anonymous identity which will be shared by all users desiring to be anonymous, such as implemented in (Zhu et al., 2010). Nevertheless, this kind of solution does not allow to perform some operations that require some association to the specific user, as for example, to charge her for the received services. In our opinion, a more convenient solution would be to allow the user to act using pseudonyms so that its real identity is not revealed. Moreover, the association between these pseudonyms and the real user identity is only known by a trusted party of the realm where the user has a subscription (*home KDC*).

One approximation may be based on assigning to the user a unique and permanent pseudonym, which would be used as her identity. Nevertheless, if the association *(real identity, pseudonym)* is revealed to unauthorized third parties (e.g., some security leak), the whole user's activity can be traced. To avoid this problem, we opt by dynamically assigning pseudonyms, which are only valid for a specific period of time. In particular, we follow a similar approach to that introduced in Pereniguez et al. (2010). With this approach, the user owns a unique and temporary pseudonym which is used as identity for a period. Specifically, we bind this time to the TGT lifetime, so that the pseudonym for which a TGT is distributed must be renewed before the TGT lifetime expires. This binding is possible since in our solution we propose a *KDC-controlled pseudonym generation* mechanism where a KDC generates and distributes the pseudonyms to the client, which the client will use as client name (*cname*) in subsequent accesses with Kerberos.

It is worth noting that the pseudonyms generated by the KDC are unique, so different users do not employ the same pseudonym. In this manner, trusted entities like KDCs can easily identify which activity has been performed by a certain user through a set of pseudonyms. By associating these pseudonyms with the real user's identity, the home KDC may generate, for example, an invoice specifying the resources consumed by a client. Taking into account these general aspects, we proceed to describe the details of the operation of level 1.

4.2.2. *Detailed operation*

Fig. 2 illustrates the operation and the extensions to Kerberos required to implement privacy level 1 in a single-realm scenario, (i.e. the client accesses services in the home realm). As we observe, the process starts when the client sends a KRB_AS_REQ message to the home KDC using the pseudonym CP_1 as the name (the realm associated to this name is *hrealm* in this example) *(1)*. When the KDC receives the message and retrieves from its user database the profile associated to the user identified as CP_1, it realizes that this identity is a pseudonym and that privacy level 1 is enabled for this user.

From this point, the KDC follows the standard Kerberos operation, assuming CP_1 as client's name (i.e. CP_1 is set in the field *cname* of Kerberos messages). In this manner, the KDC answers the client with the KRB_AS_REP *(2)* containing a TGT for the KDC (TGT_H^1). Nevertheless, this message is extended to deliver the privacy level assigned by the KDC to the client as well; and a new pseudonym (i.e. CP_2), which the client must employ in the subsequent AS exchange. To convey these values, we propose the use of the *padata* field defined for those messages exchanged with the KDC. According to Kerberos standard specification (Neuman et al., 2005a), this field can be used to extend the protocol exchanges between client and KDC. Specifically, we define two new padata types named *PA-LEVEL* and *PA-PSEUD* (PrivaKERB). The former contains a numeric value informing the client about the privacy level assigned by the KDC (denoted as *L1*). The latter contains the new pseudonym (i.e. CP_2) for the subsequent AS exchange. It is worth noting that the distribution of these pieces of information must be protected in such a manner that third parties cannot observe or modify their values. To provide a secure transmission between KDC and client of these values we therefore define a new padata type named *PA-PRIV* (PrivaKERB). *PA-PRIV* contains a sequence of padatas that are confidentiality and integrity protected under the same key employed to compute the so-called encrypted part (*enc-part*) field defined in the KRB_AS_REP message, as specified in (Neuman et al., 2005a). In addition to the encrypted sequence of padatas, the PA-PRIV padata includes the nonce N_1 (already defined in the Kerberos specification (Neuman et al., 2005a)) sent by the client in the KRB_AS_REQ to provide freshness to the padata content.

Once this AS exchange has been successfully performed, the client follows the standard protocol operation to access services, employing CP_1 as the client's identity (that is, by performing TGS exchange *(3)* with the TGT_H^1, as explained in section 2.1). When TGT_H^1 expires and the client solicits a new TGT, the client performs another AS exchange *(4)* using now pseudonym CP_2 as the new identity. The home KDC operates as previously described, that is, the client receives the assigned privacy level and a new pseudo-random pseudonym CP_3 is distributed for the next AS exchange, and so on. As we can observe, two different pseudonyms are associated to a specific user: (a) the current pseudonym (*cpseud*), which is currently being used as client's identity (e.g., CP_1), and (b) the new pseudonym (*npseud*), which will be used as client's identity in the next AS exchange to solicit a new TGT (e.g., in this example CP_2). The 2-tuple of pseudonyms (*cpseud*, *npseud*) are maintained by both client and home KDC. Whereas the *cpseud* can be maintained in volatile memory in the client, *npseud* is stored in permanent memory (i.e. a smart card) so that it is available when client's device is reinitialized. On the other side, the KDC stores the 2-tuple in its database as identities associated to a particular user, though only the home KDC knows the association between these pseudonyms and the client's real identity. As we may observe, it is important to note that the KDC cannot confirm that client really received and stored the distributed CP_2. The only way to confirm this is when the client really uses CP_2 as identity in a subsequent AS exchange. If the KDC receives the CP_1 instead of CP_2, it can determine that client did not receive the CP_2. In this case, we propose the KDC to redistribute the same CP_2 securely instead of generating a new pseudonym. Subsequently, the client will be identified as CP_2 while the acquired TGT_H^2 is valid.

Given that an eavesdropper cannot relate pseudonyms CP_1 and CP_2 since they are distributed encrypted to the client, user traceability is broken each time a new TGT is acquired. That is, as depicted in Fig. 2, an eavesdropper is unable to infer from

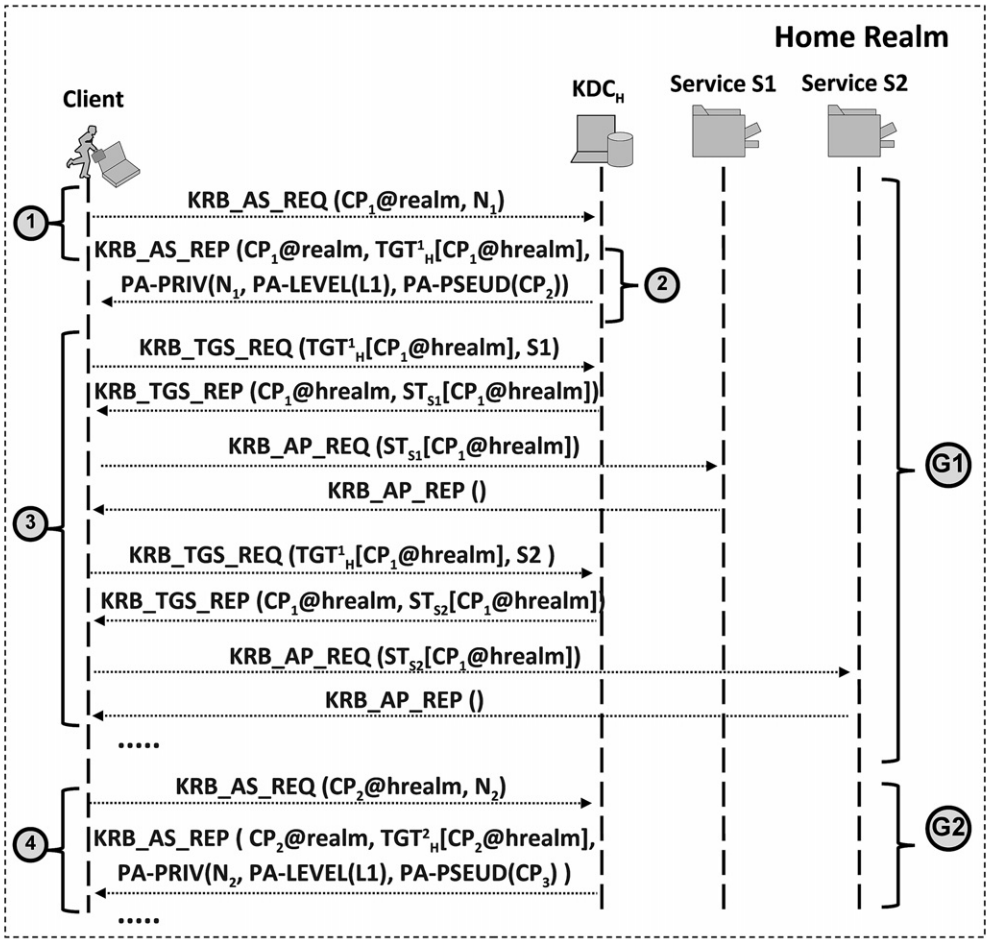

Home Realm
Client
KDC$_H$
Service S1
Service S2
KRB_AS_REQ (CP$_1$@realm, N$_1$)
KRB_AS_REP (CP$_1$@realm, TGT^1_H[CP$_1$@hrealm],
PA-PRIV(N$_1$, PA-LEVEL(L1), PA-PSEUD(CP$_2$))
KRB_TGS_REQ (TGT^1_H[CP$_1$@hrealm], S1)
KRB_TGS_REP (CP$_1$@hrealm, ST$_{S1}$[CP$_1$@hrealm])
KRB_AP_REQ (ST$_{S1}$[CP$_1$@hrealm])
KRB_AP_REP ()
KRB_TGS_REQ (TGT^1_H[CP$_1$@hrealm], S2)
KRB_TGS_REP (CP$_1$@hrealm, ST$_{S2}$[CP$_1$@hrealm])
KRB_AP_REQ (ST$_{S2}$[CP$_1$@hrealm])
KRB_AP_REP ()
.....
KRB_AS_REQ (CP$_2$@hrealm, N$_2$)
KRB_AS_REP (CP$_2$@realm, TGT^2_H[CP$_2$@hrealm],
PA-PRIV(N$_2$, PA-LEVEL(L1), PA-PSEUD(CP$_3$))
.....
1
2
3
4
G1
G2

Fig. 2 – Privacy level 1: single-realm scenario.

looking at Kerberos exchanges that groups of messages *G1* and *G2* are actually exchanged by the same user.

It is worth highlighting a special case that occurs when the client starts an AS exchange for the very first time. In this case, we assume that client is provided with an initial pseudonym by the home network through some out-of-band mechanism (e.g., pre-installed in the device). The details of such mechanisms are outside the scope of this work.

4.3. *Level 2: service access untraceability*

4.3.1. *Preliminaries*

Although the use of pseudonyms prevents the client's real identity being revealed, an eavesdropper can easily determine that the same client with a specific pseudonym (e.g., CP_1) is accessing a set of services. This is possible by simply observing the *cname* field in each KRB_TGS_REP. Thus, although the real client's identity is not accessible, eavesdroppers are able to infer (Tene, 2010; King and Jessen, 2010) which services have been requested by a specific anonymous user (*pseudonym-based service access traceability*).

We may hide the client's pseudonym from eavesdroppers to avoid this problem. However, this is a necessary but not sufficient condition to solve it. Indeed, a client typically re-uses the same TGT several times to request access to multiple services. An eavesdropper can use this TGT as a reference to discover the service access pattern of the specific anonymous user. Therefore, although the real client's identity is not accessible, eavesdroppers can infer which services have been requested by a specific anonymous user (while the same TGT is employed) and obtain anonymous profiles (Tene, 2010; King and Jessen, 2010) during the TGT lifetime (*TGT-based service access traceability*). In other words, the simple use of pseudonyms enables user anonymity but does not prevent eavesdroppers from tracing the different services accesses performed by a concrete anonymous user during the period of time that such a user employs the same TGT. When this information is systematically collected, other valuable information like service access behavioral patterns of the specific anonymous clients can be inferred (King and Jessen, 2010). As a consequence our requirement of *service access untraceability* is not met with mere pseudonyms. In fact, to achieve this requirement, we need to accomplish two goals: 1) to hide the pseudonym from the eavesdropper; and 2) to avoid the use of the same TGT each time an anonymous client requests an ST.

For the first goal, we enhance the pseudonym-based approach followed in level 1 by employing the so-called *anonymous ticket* defined in (Zhu et al., 2010). The anonymous tickets have a flag designating ($Flag_A$) that the ticket is anonymous, and are always associated to the anonymous client *anon@anon*, which is the client's identity that can be observed by an eavesdropper in the KRB_TGS_REP messages. In particular, we extend the concept of anonymous ticket by including both the privacy level assigned to the client and its pseudonym in the *authorization-data* field defined in (Neuman et al., 2005a). With this new type of ticket, we protect the client's pseudonym so that it is only revealed to authorized parties like KDCs or services, which may require it to perform, for example, charging operations.

For the second goal, we introduce the concept of *self-renewed TGT*, which has been specifically designed for our solution. While traditional TGTs in Kerberos are generated by the AS and processed by the TGS (Neuman et al., 2005a), self-renewed TGTs are generated and processed by only the TGS. In this sense, we define a new secret key named TSRK (TGT self-renewal key) generated and only known by the TGS destined to create self-renewed TGTs. The TSRK may be pre-established or randomly created when the KDC is initialized for the first time. Thus, the same security properties achieved with ordinary TGTs are also fulfilled with self-renewed TGTs. Each time a client uses a self-renewed TGT to request an ST for a service, the TGS will not only distribute the ST but also a new self-renewed TGT to the client, which the client will use to request the new ST for another service. In other words, the self-renewed TGTs are used only once (one-use ticket) for ST solicitation. Moreover, the distribution of self-renewed tickets is performed confidentially, so that an eavesdropper cannot know that the new self-renewed TGT is related in any way with the recently used self-renewed TGT. This is key to achieving service access untraceability, since the eavesdropper will observe different (one-use tickets) and unrelated TGTs each time the client accesses a service. Thus, the eavesdropper will be now unable to use of the same TGT as a pointer to obtain any service access pattern of a specific anonymous user. In the following section, we describe in detail how these tickets are used to achieve service access untraceability.

4.3.2. *Detailed operation*

Fig. 3 shows an example about how privacy level 2 works assuming that a user is accessing services in its home realm (typical single-realm scenario). The process starts when the client is authenticated through an AS exchange *(1)* following a process similar to privacy level 1. The client selects pseudonym CP_1 as principal name and sends a KRB_AS_REQ message to the AS/KDC. Upon reception, the AS/KDC realizes that CP_1 is a pseudonym by consulting its client database and determines that privacy level 2 must be enabled for the client (e.g., according to its profile). Since privacy level 2 is based on privacy level 1, the same extensions proposed for the latter are also applied here. For this reason, the AS/KDC generates a new random pseudonym (CP_2), which is sent to the client together with the assigned privacy level 2 (denoted as *L*2). Additionally, the AS/KDC generates an anonymous TGT ($\mathrm{TGT}^1_\mathrm{H}$[FlagA…]) where both privacy level and client's pseudonym CP_1 are encrypted as part of the authorization data elements in the ticket.

Next, when the client needs an ST for accessing service S1, it builds a KRB_TGS_REQ *(2)* that contains $\mathrm{TGT}^1_\mathrm{H}$. Upon reception, the TGS/KDC examines the TGT presented and realizes that privacy is enabled since the anonymous ticket flag (*FlagA*) (Zhu et al., 2010) is set. Although, the anonymous TGT is associated to the anonymous user (*anon@anon*) the TGS/KDC becomes aware that the client with pseudonym CP_1 is employing privacy level 2 extensions by looking at the authorization data field in the ticket. When the request is successfully validated, the TGS/KDC generates two tickets and sends them to the client in the KRB_TGS_REP *(3)*: a) an anonymous ST ST_{S1} (whose format is defined in Zhu et al. (2010)) associated to client *anon@anon* containing the same authorization data in the anonymous TGT; and b) an anonymous self-renewed TGT ($\mathrm{srTGT}^1_\mathrm{H}$) that contains the same values of the anonymous TGT, but an updated *starttime* field. As we can observe, $\mathrm{srTGT}^1_\mathrm{H}$ is transported through a new padata type which we define called PA-SR-TGT that, in turn, is contained in a PA-PRIV padata for achieving a secure distribution process. For this reason, the self-renewed TGT is neither visible to eavesdroppers nor can it be modified by an attacker. As a consequence, an eavesdropper cannot know what ticket the client will employ for the next ST solicitation since the self-renewed TGT is not related in any way with the TGT $\mathrm{TGT}^1_\mathrm{H}$ presented. Finally, the ST_{S1} is delivered to the service through a standard KRB_AP_REQ/KRB_AP_REP exchange *(4)*. In this case, if the service recognizes our extensions and needs to know the client's pseudonym (e.g., to charge the client for the offered service), it can obtain it from to the authorization data contained in the ST. Conversely, if the service does not support them, it will be unable to interpret the authorization data containing the client's pseudonym. The service could be provided to the anonymous user (*anon@anon*), but no charging operations will be carried out.

The same process we have explained for accessing service S1 is also applied when the client requests access to another service S2, but now the client sends $\mathrm{srTGT}^1_\mathrm{H}$ in the KRB_TGS_REQ to the TGS/KDC *(5)*. An eavesdropper tracing the communication is unable to deduce that this TGT belongs to the same client which sent the previous TGT $\mathrm{TGT}^1_\mathrm{H}$. The reason is twofold: a) the resulting self-renewed TGT is completely different from the previous one $\mathrm{TGT}^1_\mathrm{H}$; and b) the eavesdropper is unable to determine that the distributed self-renewed TGT $\mathrm{srTGT}^1_\mathrm{H}$ is related to $\mathrm{TGT}^1_\mathrm{H}$, since the former is confidentiality protected by using the PA-PRIV padata during the distribution process. After successful validation, the client receives an anonymous ST ST_{S2}[FlagA…] and a new self-renewed TGT $\mathrm{srTGT}^2_\mathrm{H}$ for the next access to the TGS/KDC *(6)*. While the former is used to access service S2 *(7)*, the latter replaces the previous $\mathrm{srTGT}^1_\mathrm{H}$. Given that self-renewed TGTs are neither re-used (they are considered as one-use tickets) nor sent in cleartext, an eavesdropper cannot relate the different ST solicitations performed by the same user. More precisely, according to Fig. 3, accesses to services S1 and S2 (represented by group of messages *G1* and *G2*, respectively) are performed by non-related users from the eavesdropper's point of view.

When the self-renewed TGT expires, the client must be authenticated through a new AS exchange *(8)*. Then the client starts using CP_2 as its identity, and obtains a new pseudonym for the next AS exchange. As already discussed in Section 4.2, the change of pseudonym prevents an eavesdropper from

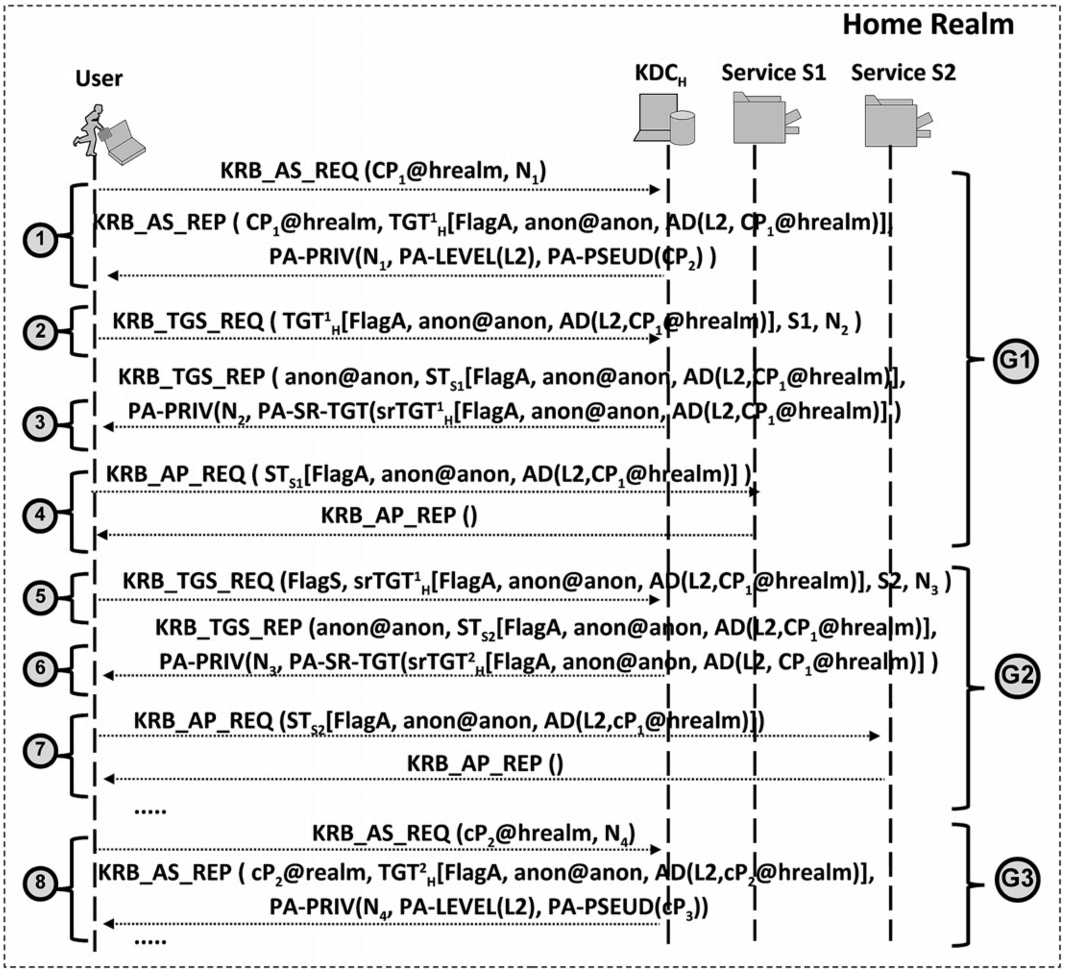

Home Realm
User
KDC_H
Service S1
Service S2
KRB_AS_REQ (CP1@hrealm, N1)
KRB_AS_REP (CP1@hrealm, TGT1H[FlagA, anon@anon, AD(L2, CP1@hrealm)],
PA-PRIV(N1, PA-LEVEL(L2), PA-PSEUD(CP2))
KRB_TGS_REQ (TGT1H[FlagA, anon@anon, AD(L2,CP1@hrealm)], S1, N2)
KRB_TGS_REP (anon@anon, STS1[FlagA, anon@anon, AD(L2,CP1@hrealm)],
PA-PRIV(N2, PA-SR-TGT(srTGT1H[FlagA, anon@anon, AD(L2,CP1@hrealm)])
KRB_AP_REQ (STS1[FlagA, anon@anon, AD(L2,CP1@hrealm)])
KRB_AP_REP ()
KRB_TGS_REQ (FlagS, srTGT1H[FlagA, anon@anon, AD(L2,CP1@hrealm)], S2, N3)
KRB_TGS_REP (anon@anon, STS2[FlagA, anon@anon, AD(L2,CP1@hrealm)],
PA-PRIV(N3, PA-SR-TGT(srTGT2H[FlagA, anon@anon, AD(L2, CP1@hrealm)])
KRB_AP_REQ (STS2[FlagA, anon@anon, AD(L2,cP1@hrealm)])
KRB_AP_REP ()
.....
KRB_AS_REQ (cP2@hrealm, N4)
KRB_AS_REP (cP2@realm, TGT2H[FlagA, anon@anon, AD(L2,cP2@hrealm)],
PA-PRIV(N4, PA-LEVEL(L2), PA-PSEUD(cP3))
.....
1
2
3
4
5
6
7
8
G1
G2
G3

Fig. 3 – Privacy level 2: single-realm scenario.

relating messages exchanged under a new pseudonym (*G3*) with previous ones (*G1* and *G2*).

4.4. *Operation in cross-realm scenarios*

In a cross-realm authentication, our objective is to minimize the deployment cost of the solution and favor its adoption. Indeed, we have designed a privacy-enhanced cross-realm operation where only KDCs where the client has a subscription (*home KDCs*) and the KDCs that the client visits in another realm (*visited KDCs*) must be updated to support our privacy extensions. Thus, intermediary KDCs (which are placed to intermediate between home and visited KDCs in a Kerberos cross-realm infrastructure (Neuman et al., 2005a), as depicted in Fig. 1(b)) can use existing implementations based on the Kerberos specification (Neuman et al., 2005a) without support of our privacy extensions. Next, we describe how both levels of privacy operate in a cross-realm scenario.

4.4.1. *Level 1*

Level 1 can be adapted straightforwardly to the cross-realm operation as it only introduces some extensions to the initial AS exchange performed with the AS/KDC server. So basically, intermediate TGS/KDCs are not affected by our extensions.

Fig. 4 details how privacy level 1 works over a cross-realm scenario. In our example, we describe the process when a user accesses services S1 and S2 controlled by the visited KDC (KDC_V), other than its home KDC (KDC_H). To simplify the analysis, we assume that there only exists an intermediary KDC KDC_I. After AS exchange *(1)*, the client is informed about both the privacy level assigned and the new pseudonym to be employed in the next AS exchange. Next, a typical cross-realm process based on several KRB_TGS_REQ/REP exchanges is performed following the standard Kerberos *(2)*. Once the client acquires a valid ST for service *S1*, it authenticates itself against the service *(3)*.

By re-using the acquired cross-realm TGT (TGT^1_V), the client is able to solicit another ST for accessing service *S2 (4)*. As observed, the use of privacy level 1 is transparent to the KDC in the visited realm KDC_V and the intermediary KDC KDC_I. However, although user anonymity is achieved, the client employs the same pseudonym CP_1 inside the visited realm, so an eavesdropper can determine that all Kerberos messages (denoted as group of messages *G1*) for the cross-realm operation and service accesses are performed by the same user. To

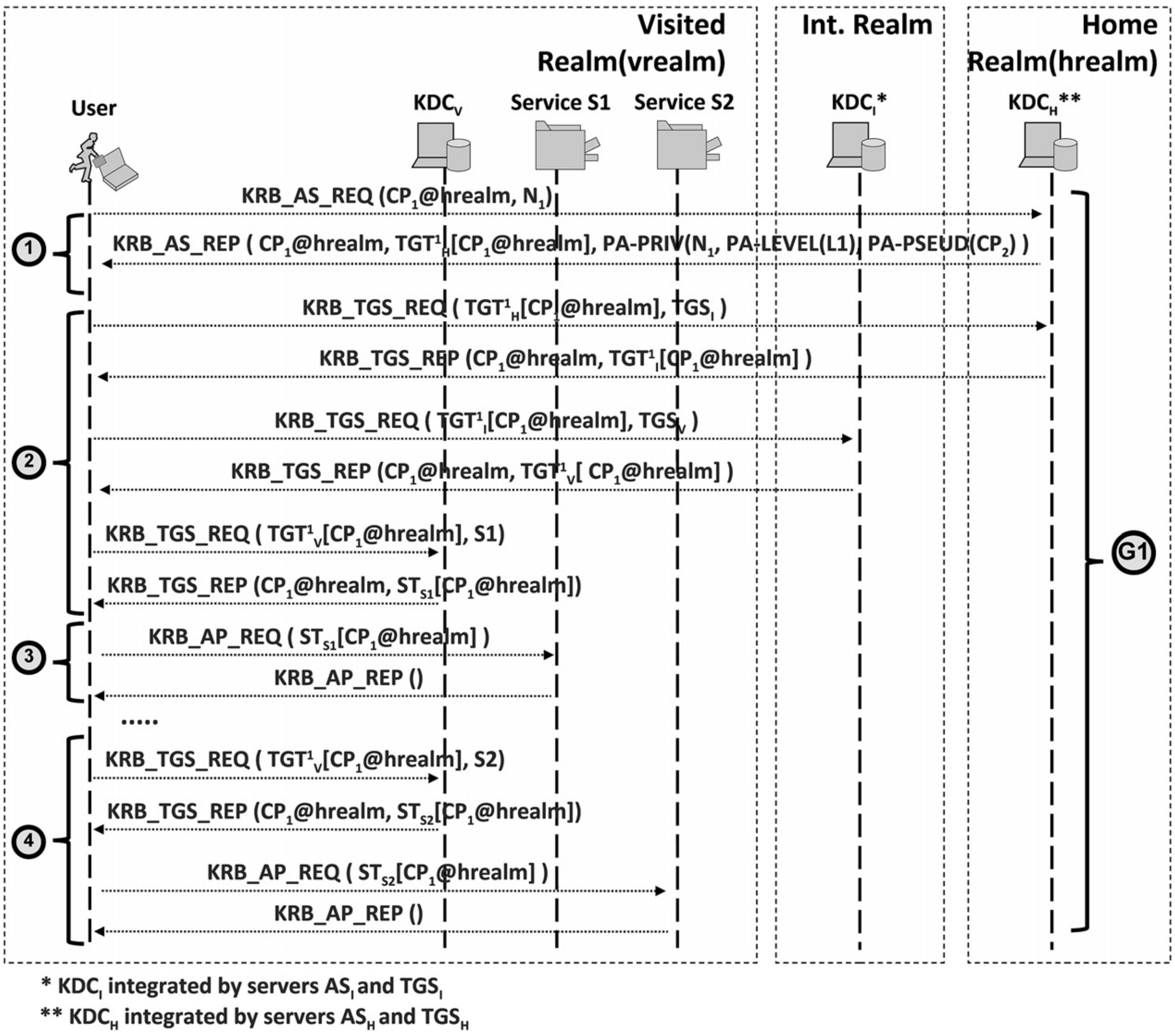

Visited Realm(vrealm)
Int. Realm
Home Realm(hrealm)
User
KDC_V
Service S1
Service S2
KDC_I*
KDC_H**
KRB_AS_REQ (CP_1@hrealm, N_1)
KRB_AS_REP (CP_1@hrealm, TGT¹_H[CP_1@hrealm], PA-PRIV(N_1, PA-LEVEL(L1)), PA-PSEUD(CP_2))
KRB_TGS_REQ (TGT¹_H[CP_1@hrealm], TGS_I)
KRB_TGS_REP (CP_1@hrealm, TGT¹_I[CP_1@hrealm])
KRB_TGS_REQ (TGT¹_I[CP_1@hrealm], TGS_V)
KRB_TGS_REP (CP_1@hrealm, TGT¹_V[CP_1@hrealm])
KRB_TGS_REQ (TGT¹_V[CP_1@hrealm], S1)
KRB_TGS_REP (CP_1@hrealm, ST_S1[CP_1@hrealm])
KRB_AP_REQ (ST_S1[CP_1@hrealm])
KRB_AP_REP ()
.....
KRB_TGS_REQ (TGT¹_V[CP_1@hrealm], S2)
KRB_TGS_REP (CP_1@hrealm, ST_S2[CP_1@hrealm])
KRB_AP_REQ (ST_S2[CP_1@hrealm])
KRB_AP_REP ()
1
2
3
4
G1
* KDC_I integrated by servers AS_I and TGS_I
** KDC_H integrated by servers AS_H and TGS_H

Fig. 4 – Privacy level 1: cross-realm scenario.

solve this issue, which violates service access untraceability, privacy level 2 must be used instead.

4.4.2. *Level 2*

Fig. 5 shows how privacy level 2 works in a cross-realm operation. Initially, if the client does not possess a valid TGT for the home KDC, it performs an AS exchange to obtain one following the process described in section 4.3 by using CP_1 as name *(1)*. In this step, the KDC generates an anonymous TGT (TGT_H^1) and a new pseudonym CP_2 for the next AS exchange.

After that, the client starts a cross-realm operation involving (three) TGS exchanges to obtain a service ticket for S1, i.e. one with every KDC (KDC_H, KDC_I and KDC_V). In the first TGS exchange with KDC_H *(2)*, the client solicits a cross-realm TGT for KDC_I. According to the privacy level 2 operation, KDC_H issues an anonymous TGT (TGT_I^1) for the KDC_I and a self-renewed TGT ($srTGT_H^1$). While the client replaces TGT_H^1 (acquired in the initial AS exchange) by $srTGT_H^1$, the former is sent to KDC_I *(3)* in order to obtain a cross-realm TGT for KDC_V (TGT_V^1). Since it is assumed that KDC_I does not support privacy, it will process the request following the base Kerberos specification (Neuman et al., 2005a), that is: (a) since KDCs ignore unknown flags, the anonymous flag (*FlagA*) will not be set in the TGT_V^1 issued by the intermediary KDC (KDC_I); and (b) since authorization data elements defined in our solution and contained in TGT_I^1 are not recognized by intermediate KDCs (which may not support our privacy extensions), they propagate those data types to derivative tickets (TGT_V^1 in this case). In summary, only authorization data is propagated through cross-realm TGTs.

Once the client acquires TGT_V^1, it is ready to perform the final TGS exchange with KDC_V *(4)*. On the reception of the KRB_TGS_REQ, KDC_V starts analyzing TGT_V^1. At first sight, it observes that is a cross-realm TGT issued to the well-known anonymous client (*anon@anon*). Assuming that KDC_V is privacy-enabled, more information can be extracted from the authorization data ($AD(L2, CP_1)$) contained in the ticket. With this information, KDC_V deduces that a client named CP_1 coming from realm *hrealm* is demanding privacy level 2 support. Therefore, following the privacy level 2 operation (see Section 4.3), KDC_V distributes an anonymous ST ($ST_{S1}[FlagA\ldots$)

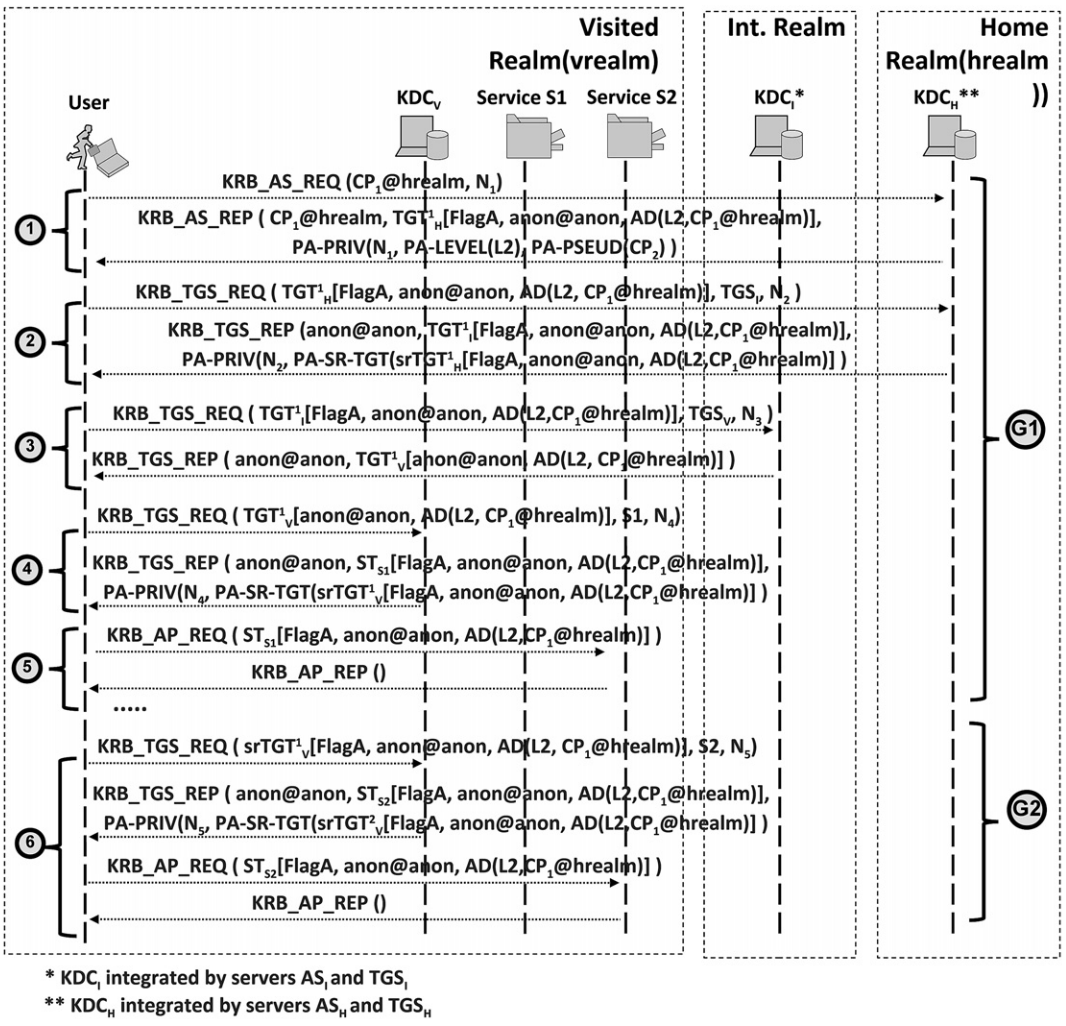

Visited Realm(vrealm)
Int. Realm
Home Realm(hrealm))
User
KDCV
Service S1
Service S2
KDCI*
KDCH**
KRB_AS_REQ (CP1@hrealm, N1)
KRB_AS_REP (CP1@hrealm, TGT1H[FlagA, anon@anon, AD(L2,CP1@hrealm)], PA-PRIV(N1, PA-LEVEL(L2), PA-PSEUD(CP2))
KRB_TGS_REQ (TGT1H[FlagA, anon@anon, AD(L2, CP1@hrealm)], TGSI, N2)
KRB_TGS_REP (anon@anon, TGT1I[FlagA, anon@anon, AD(L2,CP1@hrealm)], PA-PRIV(N2, PA-SR-TGT(srTGT1H[FlagA, anon@anon, AD(L2,CP1@hrealm)])
KRB_TGS_REQ (TGT1I[FlagA, anon@anon, AD(L2,CP1@hrealm)], TGSV, N3)
KRB_TGS_REP (anon@anon, TGT1V[anon@anon, AD(L2, CP1@hrealm)])
KRB_TGS_REQ (TGT1V[anon@anon, AD(L2, CP1@hrealm)], S1, N4)
KRB_TGS_REP (anon@anon, STS1[FlagA, anon@anon, AD(L2,CP1@hrealm)], PA-PRIV(N4, PA-SR-TGT(srTGT1V[FlagA, anon@anon, AD(L2,CP1@hrealm)])
KRB_AP_REQ (STS1[FlagA, anon@anon, AD(L2,CP1@hrealm)])
KRB_AP_REP ()
.....
KRB_TGS_REQ (srTGT1V[FlagA, anon@anon, AD(L2, CP1@hrealm)], S2, N5)
KRB_TGS_REP (anon@anon, STS2[FlagA, anon@anon, AD(L2,CP1@hrealm)], PA-PRIV(N5, PA-SR-TGT(srTGT2V[FlagA, anon@anon, AD(L2,CP1@hrealm)])
KRB_AP_REQ (STS2[FlagA, anon@anon, AD(L2,CP1@hrealm)])
KRB_AP_REP ()
1
2
3
4
5
6
G1
G2
* KDCI integrated by servers ASI and TGSI
** KDCH integrated by servers ASH and TGSH

Fig. 5 – Privacy level 2: cross-realm scenario.

and self-renewed TGT ($srTGT^1_V$). Note that KDC_V re-establishes the use of the anonymous flag in both tickets. If the KDC_V does not support our extensions it will ignore unknown flags and authorization data types contained in the ticket presented and answer with a KRB_TGS_REP, which does not include any of the our privacy extensions (following the standard Kerberos (Neuman et al., 2005a)). The client can note this situation just by verifying the KRB_TGS_REP and can decide to abort the process since privacy level in which it is enrolled is not provided. Finally, the client can use the anonymous ST ST_{S1} to access the service *(5)*. If the service needs the client's pseudonym for charging purposes, it can obtain it from the authorization data in the ST_{S1}.

As we can observe, even if intermediary KDCs do not support privacy, the service access untraceability is respected. In particular, when the service access implies a cross-realm operation, an eavesdropper will be able to determine that all exchanges grouped by label G1 belongs to the same client. Nevertheless, once the client communicates with the visited KDC, the latter provides a self-renewed TGT that allows traceability with subsequent access to a service (G2) located in the visited realm to be broken, by accomplishing *requirement* 2. Thus, when the client uses the self-renewed TGT to solicit an ST *(6)* to access another service (e.g., S2) in the visited realm, an eavesdropper is unable to deduce that the access is performed by the same anonymous client.

In conclusion, our solution provides both levels of privacy not only in a single-realm case but also when a cross-realm Kerberos infrastructure is deployed. Table 2 summarizes both privacy levels. As we may observe, level 1 is a trade-off between privacy provided to the user and deployment cost. In this level, the user remains anonymous by using periodically renewed pseudonyms instead of the real identity. This simple but effective solution requires the Kerberos protocol to be extended by defining three new padata types, one of which is especially destined to protect with confidentiality and integrity the information sent from the KDC to the client. Furthermore, the main advantage of level 1 is that these extensions only affect the AS exchange, so only the AS module of the home KDC needs to be updated. This feature becomes more important in cross-realm scenarios since neither intermediary nor visited KDCs need to be modified. For this reason, level 1 is suitable for those scenarios where it is desirable to minimize the deployment impact.

Table 2 – Privacy levels comparison and related issues.

	LEVEL 1	LEVEL 2
Type of privacy	Anonymity	Anonymity + Service Access Untraceability
Eavesdropper knowledge	The behavior of an anonymous user can be profiled while using the same TGT	The set of messages exchanged to access a service can be traced
Features	Pseudonyms	Pseudonyms, anonymous tickets and self-renewed TGTs
Kerberos extensions	New padata types (*PA-LEVEL*, *PA-PSEUD*, *PA-PRIV*)	Level 1 extensions + well-known anonymous identity (*anon@anon*) + anonymous flag (*FlagA*) + new padata type (*PA-SR-TGT*) + new authorization data types (level and client's identity)
Deployment	Easier deployment. Only the home AS/KDC must be updated	Home and visited KDCs must be updated. Implementation of anonymous tickets and self-renewed TGTs.

On the other hand, although users remain anonymous, just using pseudonyms enables eavesdroppers to trace the activity of a certain user during the period of time that it employs the same pseudonym and to extract anonymous profiles and behavioral patterns. This situation does not pose a risk for a particular client since profiles cannot be linked to its real identity. Nevertheless, it must be carefully considered by operators and service providers since observers are able to identify common behavioral patterns of their clients (King and Jessen, 2010). If this circumstance represents a serious concern, the use of privacy level 2 is recommended. As a consequence, level 2 is a qualitative improvement on level 1, since it offers not only user anonymity but also service access untraceability. Nevertheless, a higher privacy protection is achieved at the expense of increasing the deployment cost. In our opinion, this is assumable though, taking into account that intermediary KDCs are not required to be privacy-enabled servers in cross-realm operations.

5. Performance evaluation

It is expected that the proposed extensions will introduce some additional latency with respect to the base Kerberos protocol, so increasing the service access time. Thus, the key question is whether the additional cost imposed by our privacy extensions is affordable for next generation networks. To perform this evaluation, we have developed an implementation prototype that is used to evaluate the performance of our privacy framework.

5.1. *Deployed testbed and implementation details*

To implement privacy levels 1 and 2, we have selected the open-source MIT Kerberos implementation v.1.6.3 (MIT Kerberos Distribution). The main implementation work has focused on extending the message construction and processing message engine since most changes affect the information transported by messages. Additionally, special effort has been devoted to the encode and decode routines in order to support new authorization data and padata types (described in (PrivaKERB)).

To conduct real experiments we have developed different programs that simulate a generic client and service employing the Kerberos *Application Programming Interface* (API). Both programs interact with a KDC (already implemented by the MIT Kerberos distribution) in charge of distributing tickets to the client. The client implements all the Kerberos exchanges in order to be successfully authenticated by the service performing the next processes: (1) TGT acquisition through an AS exchange; (2) ST acquisition through one or several TGS exchanges, (3) service access through an AP exchange. As regards the service, we have implemented the privacy extensions that must implement real services in order to understand our new authorization data types (if present in the ST) containing privacy information of the user.

We have built a basic and generic scenario that allows us to represent different situations that may occur in a real mobile environment. The experimental testbed comprises the elements shown in Table 3 and two scenarios as depicted in Fig. 6. Scenario I is to experiment with single-realm cases and the Scenario II for cross-realm, where the three KDCs are geographically distributed at different places (two of them in Spain and the third in Greece) by simulating three different administrative realms (average roundtrip times between realms are specified in Section 5.2.2).

Finally, for the sake of completeness, we would like to highlight three important implementation aspects. First, we use the *KerberosString* format to generate pseudonyms (which act as principal names) following the instructions in Neuman et al. (2005a). This format is a *GeneralString* ASN.1 data type that is constrained to contain characters in *IA5String*. Since control characters should not be used in principal names, only the 95 printable characters of the IA5String (Dubuisson, 2001) alphabet are available. To avoid collisions, we randomly produce 8 character length principal names which means

Table 3 – Testbed machines.

Role	CPU type	Freq (MHz)	RAM (MB)
Kerberos client	VIA Nehemiah	1200	488
Kerberos service	VIA Nehemiah	1200	488
Wireless-G Broadband Router	Linksys WRT54GL	200	16
KDC	Pentium 4	3200	1024
KDC	Pentium 4	3000	512
KDC	Pentium 4	3000	512

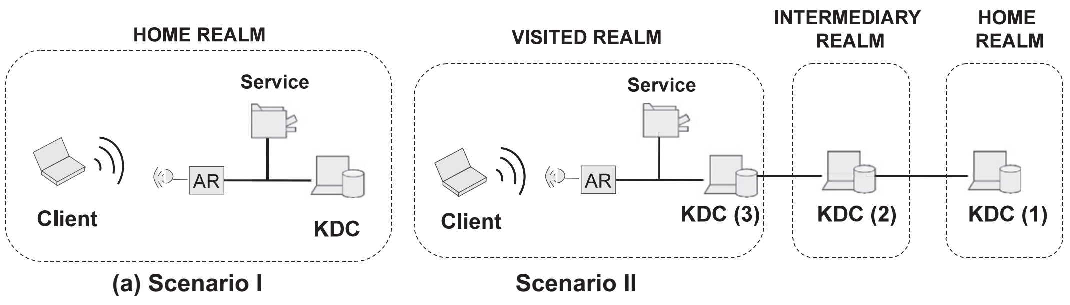


Fig. 6 – Deployed Testbed.

more than 6 quadrillion different pseudonyms to be generated. Second, in addition to the shared secret keys defined in Neuman et al. (2005a), the KDC is also configured with a TSRK that is stored in the KDC database. The TSRK is a secret key only known by the TGS and independent of the user. Therefore, this key is not disclosed to any third parties and can only be recovered by the TGS module when it issues or receives self-renewed TGTs. And third, with regard to the cryptographic operations, we have used the well-known *Advanced Encryption Standard* (AES) algorithm with a block cipher size of 128 bits and cipher-block chaining (CBC) operation mode (Neuman et al., 2005b; Raeburn, 2005). When integrity protection is required, we employ the HMAC-SHA1-96 function (Raeburn, 2005; Madson and Glenn, 1998).

5.2. *Performance analysis*

Using the aforementioned testbed we evaluate the penalty introduced by our privacy framework. We use the term *standard Kerberos* to refer the execution of the original version of the Kerberos protocol such as defined in Neuman et al. (2005a) (that is, without our extensions); and the term *PrivaKERB* to denote our privacy-enhanced Kerberos implementation with privacy support. We carry out this comparison in two different scenarios depending on whether the client solicits access to a service controlled by its home KDC (single-realm scenario) or by a visited KDC (cross-realm scenario). Since we assume that the client needs to be authenticated against the KDC, the client performs the three Kerberos exchanges: AS, TGS and AP. Every service access is executed around 500 times with both the standard and the privacy-enhanced Kerberos. When testing the standard Kerberos configuration we collect the following information:

- *Message length*. This is the length of a specific message transmitted over the network including IP, data link and physical headers.
- *Network time*. This metric represents (as a 95% confidence interval) the time devoted to transmitting messages over the network.
- *Message processing time*. This measures the time devoted by a certain entity (represented as a 95% confidence interval) to process a message, since it is received on the network interface until a response is sent. Therefore, for example, IP routing time is also included.
- *Exchange time*. This collects a 95% confidence interval that contains the total time required by a client to complete a specific Kerberos exchange. This metric comprises both network and message processing times.

For the privacy-enhanced solution we employ the same metrics. Nevertheless, we specifically measure the *privacy processing time* to perform an accurate measurement of the additional latency. That is, the additional time required by each entity to perform the additional privacy-related tasks.

5.2.1. *Scenario I: single-realm case*

We assume a client attempting to access a service for the first time with any valid TGT. Thus, the client will need to perform the three different Kerberos exchanges in order to access the service. To test this situation we develop the network architecture shown in Fig. 6(a), where we simulate a client employing a wireless connection through a wireless access router (AR).

Table 4 collects results obtained for the standard Kerberos protocol (Neuman et al., 2005a). These values are used as base reference to contrast with the overhead introduced by our privacy extensions. As observed, we provide the specific values measured for the different metrics indicating confidence intervals when measuring times. The results obtained for PrivaKERB are summarized in Fig. 7 to perform a graphical comparison between the different privacy levels. As observed, only mean values are graphically represented for the measured values. Details about the specific values (indicating confidence intervals where appropriate) measured for PrivaKERB operation can be found in PrivaKERB.

Fig. 7(a) shows the mean time required by the client to complete each Kerberos exchange using both standard Kerberos and our privacy extensions. First, regarding the standard Kerberos, we notice that the AS exchange requires far more time (about 9 times) than the other exchanges. Through information obtained from the source code, we detected that most of this time is consumed to derive the client's secret key from the password. Regarding the overhead introduced by PrivaKERB, in general we can observe that the use of self-renewed TGTs produces a small increment in the TGS exchange, remaining with both the AS and AP exchanges remaining below similar values. Indeed, analyzing times related to level 1, we observe that a small latency of ~0.2 ms is introduced in the AS exchange. As expected, since this level does not introduce any extension to the TGS and AP exchanges, these times remain below values similar to the

Table 4 – Results for standard Kerberos in scenario I.

	Exchange time (ms)	Message processing time (ms)		Message length (bytes)		Network time (ms)
AS Exchange	59.479 ± 0.041	Client	57.791 ± 0.103	KR_AS_REQ	204	1.075 ± 0.023
		KDC	0.503 ± 0.031	KR_AS_REP	584	
TGS Exchange	6.545 ± 0.023	Client	3.618 ± 0.052	KR_TGS_REQ	624	1.295 ± 0.028
		KDC	1.740 ± 0.055	KR_TGS_REP	611	
AP Exchange	6.987 ± 0.041	Client	2.583 ± 0.048	KR_AP_REQ	451	0.915 ± 0.029
		Service	3.491 ± 0.042	KR_AP_REP	127	

standard Kerberos. Conversely, level 2 requires some additional time in every exchange. More precisely, the AS exchange is increased by ∼0.35 ms. This time is higher in the TGS exchange, where ∼1.4 ms are required to complete the privacy extensions. Instead, the latency introduced in the AP exchange is ∼0.02 ms, smaller than in the aforementioned exchanges.

The impact of the privacy-enhanced solution over each entity in terms of computing time is depicted in Fig. 7(b). More specifically, we collect the latency spent in executing the privacy extensions. These values must be compared with the processing time devoted by each entity to complete every exchange (see Table 4) in the standard Kerberos case. Level 1 only produces a time penalty to the client of ∼0.15 ms and to KDC of ∼0.07 ms in the AS exchange. These values are slightly higher for privacy level 2, where the client and KDC require about 0.91 ms and 0.18 ms to handle anonymous tickets and self-renewed TGTs. This time devoted to complete the privacy level 2 tasks increases in the TGS exchange where the client and KDC need ∼0.9 ms and ∼0.2 ms, respectively. Regarding the AP exchange, the service only needs some extra time (∼0.2 ms) to check the new authorization data types contained in the ST presented by the client.

Finally, since the privacy-enhanced solution requires additional information to be exchanged between entities, another aspect we have measured is the message size and network times that may affect the bandwidth consumption. While Fig. 7(c) provides the size of those messages involved in the different Kerberos exchanges, Fig. 7(d) specifies the mean network time devoted to transmission and propagation over the network. In comparison with standard Kerberos, level 1 introduces the lowest network overload since only the KRB_AS_REP message increases in ∼110 bytes. Therefore, while we can appreciate an increment close to 0.04 ms in the AS exchange, network times in both TGS and AP exchanges remain under similar values. In contrast, level 2 not only affects the size of the AS exchange messages but also the TGS and AP exchanges. An appreciable increment is observed in the KRB_AS_REP (∼180 bytes) and KRB_TGS_REP

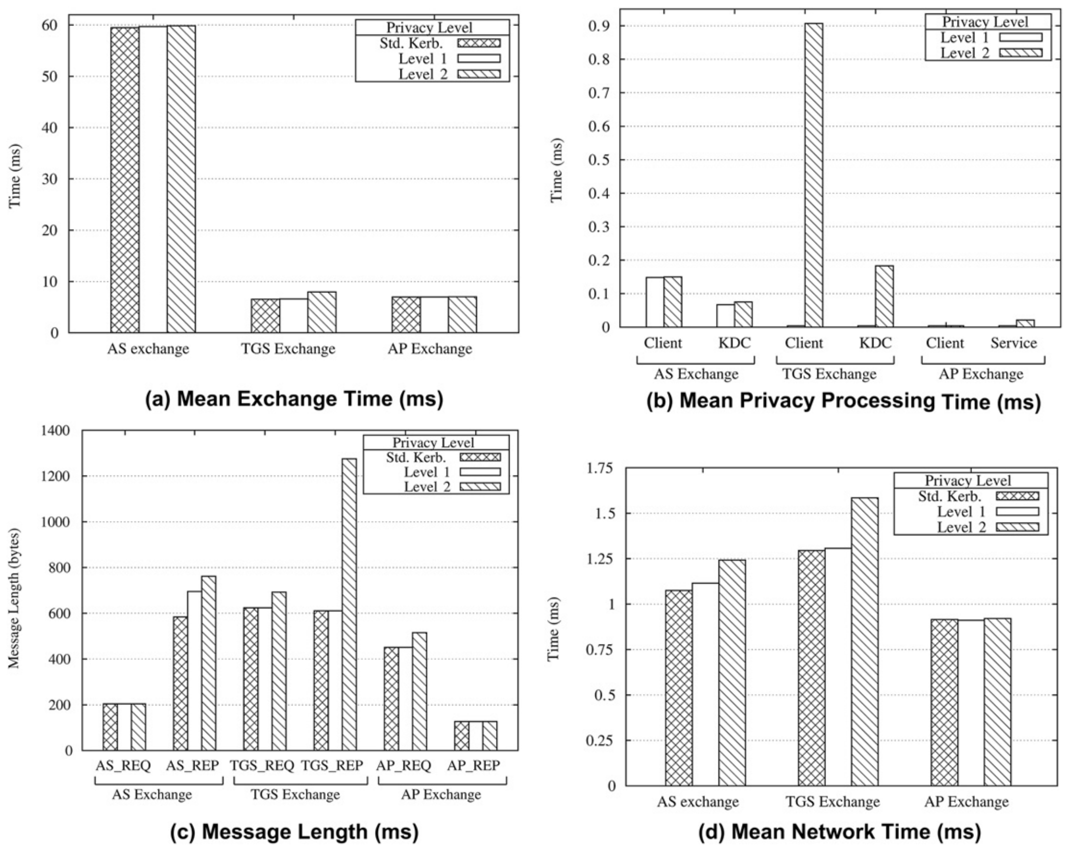


Fig. 7 – Results for PrivaKERB in scenario I.

Table 5 – Results for standard Kerberos in scenario II.

	Exchange time (ms)	Message processing time (ms)		Message length (bytes)		Network time (ms)
AS Exch.	162.717 ± 0.061	Client	57.860 ± 0.087	KR_AS_REQ	204	104.250 ± 0.067
		KDC	0.608 ± 0.037	KR_AS_REP	584	
TGS Exch. 1	111.238 ± 0.097	Client	3.671 ± 0.111	KR_TGS_REQ	602	104.723 ± 0.061
		KDC	2.844 ± 0.027	KR_TGS_REP	564	
TGS Exch. 2	124.583 ± 0.128	Client	3.615 ± 0.104	KR_TGS_REQ	606	117.653 ± 0.052
		KDC	2.810 ± 0.056	KR_TGS_REP	609	
TGS Exch. 3	6.756 ± 0.110	Client	3.770 ± 0.034	KR_TGS_REQ	652	1.291 ± 0.041
		KDC	1.712 ± 0.026	KR_TGS_REP	622	
AP Exch.	6.964 ± 0.045	Client	2.535 ± 0.051	KR_AP_REQ	460	0.917 ± 0.039
		Service	3.486 ± 0.043	KR_AP_REP	127	

(~660 bytes) due to the presence of self-renewed TGTs. Additionally, we note that the use of anonymous tickets increases the message length ~60 bytes (e.g., see KRB_AP_REQ). However, we observe that these sizes correspond to an insignificant increment of ~0.30 ms (TGS exchange) in network time.

5.2.2. *Scenario II: cross-realm case*

In this second scenario we analyze the behavior of our privacy-enhanced solution during a cross-realm scenario. As in Section 5.2.1, we suppose a user that initializes its device and access to a kerberized service for the first time. In the network architecture for this scenario (depicted in Fig. 6(b)) three separated KDCs are deployed to simulate a cross-realm operation. The average roundtrip time between the home and visited network is ~104 ms and ~117 ms between the visited and intermediate networks, although these values may only be considered as an indication. The client will follow the authentication path from the home to the visited KDC through an intermediary one. It is important to mention that in all tests performed in this scenario, even in the privacy-enhanced configuration, the intermediary KDC deploys the standard MIT Kerberos implementation without our privacy extensions.

Table 5 shows the measurements obtained for the standard Kerberos protocol. It collects means values and confidence intervals of the exchange time, message processing time, message sizes and network times. Please note that, TGS Exchange 1, 2, 3 represents TGS exchanges for Home KDC, Intermediate KDC and Visited KDC respectively. For the PrivaKERB solution, results are summarized in Fig. 8 by graphs (detailed measured values are provided in PrivaKERB).

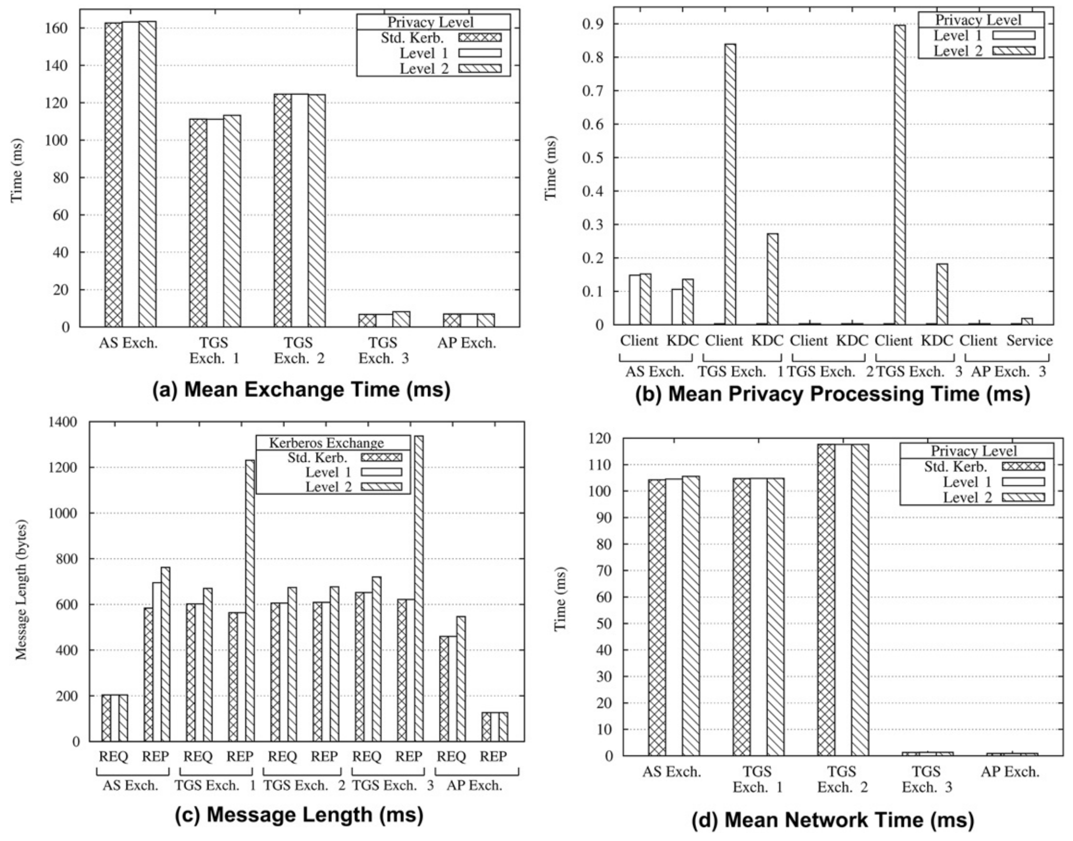


Fig. 8 – Results for PrivaKERB in scenario II.

Starting with the mean exchange times depicted in Fig. 8(a), we observe that level 1 introduces a small latency of ~0.5 ms in the AS exchange. Since neither TGS nor AP exchange is extended at this level, times obtained for the other exchanges are similar to the standard Kerberos case. On the other hand, when analyzing measurements from level 2, we observe that, except "TGS exchange 2" performed with the intermediary KDC, all times include an additional latency compared with standard Kerberos. Although these penalizations fluctuate around values between 1 and 2 ms, the impact in the final time is insignificant, especially in those messages exchanged with the home KDC. For example, the additional time to perform "TGS exchange 1" enabling level 2 extensions is ~2.06 ms, which increases the total exchange time by only ~1.9%. Focusing on the processing time results (see Table 5 and Fig. 8(b)), we observe that the intermediary KDC does not perform any privacy task (columns of "TGS exchange 2" are set to 0). Recall that our privacy framework does not require intermediary KDCs to be privacy-enabled. Comparing both privacy levels, we can conclude that level 1 is the most lightweight since it only overloads the AS exchange with 0.15 ms (client) and 0.11 ms (KDC) to execute the privacy extensions. On the contrary, level 2 introduces some extra computing time in every exchange with the home and visited KDC. The highest times are found for the client that devotes ~0.9 ms to complete the "TGS Exchange 3" privacy tasks. On the other hand, KDCs attend to a privacy-enhanced request in less than 0.3 ms.

Regarding the message sizes (Fig. 8(c)) and network times (Fig. 8(d)), the same conclusions as extracted for the single-realm scenario can be drawn here. On examining the message sizes, we see that level 1 only requires the additional transmission of ~110 bytes in the KRB_AS_REP message. Consequently, network times for level 1 are similar to that of standard Kerberos, detecting only an increment of ~0.25 ms in the AS exchange. Conversely, the use of anonymous tickets in level 2 increases the size of every message transporting a ticket, even those exchanged with the intermediary KDC. This increment is even higher when the home or visited KDC sends the client a self-renewed TGT, where the KRB_TGS_REP is up to 1200–1300 bytes. However, that extra time to transmit this information over the network is ~1.1 ms in the worst case (TGS exchange 1).

6. Conclusions and future work

In the onset of the 4G era, user privacy is becoming extremely important in wired and wireless communication infrastructures. In this context, this paper proposes a framework named PrivaKERB, which preserves user privacy in Kerberos. PrivaKERB comprises a dualmode solution which provides anonymity and service access untraceability to Kerberos clients. This can be attained even in cross-realm transactions where the client roams away from his home network. Special care has been taken not to inflict any modifications on the standard Kerberos protocol and hence be able to interact with existing implementations without privacy support. After analyzing the internal components of our framework, we demonstrate that it is also lightweight, imposing almost insignificant overheads in terms of service times, resource and network utilization. To reach this conclusion we compare the behavior of PrivaKERB against standard Kerberos over a properly designed testbed. We argue that PrivaKERB provides a flexible privacy preserving mechanism for Kerberos so far as it guarantees anonymity, does not rely on any form of PKI, and does not obstruct important network operations like accounting/charging service access.

Several research lines arise as future work from the contribution presented in this paper. The proposal of a secure and reliable mechanism that will allow the home network to provide the user with an initial pseudonym (as noted in Section 4.2.2) is an interesting issue that calls for investigation. Also, despite our privacy extensions require a minimal extra computing time on the KDC, we will analyze the behavior of PrivaKERB under stress, i.e. having a large number of clients requesting tickets from the KDC. Additionally, we will study the application of concepts related with PrivaKERB (that is, use of pseudonyms, anonymous tickets enriched with protected pseudonym and self-renewed tickets) to provide privacy to others ticket-based protocols (e.g., (Syverson, 1993; Lootah et al., 2007; Lee et al., 2010)). Finally, regarding the improvement of the PrivaKERB framework itself, two interesting topics can be identified. On the one hand, we will concentrate on incrementing the untraceability in Kerberos by, for example, applying the idea of self-renewed tickets also when a client accesses the same service with the same ST. The reason is that, potentially, a user can re-use an ST to access a specific service during the period of time the ticket is valid. Since the same ST is presented in every access, an eavesdropper can easily track all the accesses performed by the same anonymous user to the service. Thus, we foresee a solution based on applying the idea of self-renewed ticket to the STs. On the other hand, we also consider the service identity protection, so that the identity of an accessed service is not revealed to unauthorized parties. This feature is interesting for service providers that, for example, want to hide the most popular services.

Acknowledgments

This work is partially supported by a Seneca Foundation Human Resources Researching Training Program 2010 and Funding Program for Research Groups of Excellence (04552/GERM/06), and the Spanish Ministry of Science and Education (TIN2008-06441-C02-02).

REFERENCES

Apostolopoulos G, Peris V, Saha D. Transport Layer Security: how much does it really cost?. In: Proc. of IEEE INFOCOM 1999, vol. 2. New York, NY, USA: IEEE Computer Society; March 1999. p. 717–25.

Apostolopoulos G, Peris V, Saha D. A critical review of 10 years of privacy technology. In: Proc. of proceedings of surveillance cultures: a global surveillance society? UK, April 2010.

Bagnulo M, Garcia-Martines A, Azcorra A. An architecture for network layer privacy. In; ICCC 2007: International Conference on Communications 2007, pages 1509–1514, Washington, DC, USA, 2007.

Benjumea V, Lopez J, Troya JM. Anonymity analysis in credentials-based systems: a formal framework. Computer Standards & Interfaces 2008;30(4):253–61.

Bowen CL, Martin TL. A survey of location privacy and an approach for solitary users. In: Proc. of the 40th Annual Hawaii International Conference on System Sciences, page 163c, Washington, DC, USA, 2007.

Boyd C, Mathuria A. Protocols for authentication and key establishment. Springer; Sept. 2003.

Brennen VA. Kerberos infrastructure HOWTO, http://cryptnet.net/fdp/admin/kerby-infra/en/kerby-infra.html; Sept. 2004.

Cardoso RS, Speicys R, Valerie I. Architecting pervasive computing systems for privacy: a survey. In: WICSA 2007: Proceedings of the Sixth working IEEE/IFIP conference on Software architecture. Washington, DC, USA: IEEE Computer Society; 2007. p. 26.

Chen H, Xiao Y, Hong X, Hu F, Xie J. A survey of anonymity in wireless communication systems. Security and Communication Networks 2008;2(5):427–44.

Christin D, Hollick M, Manulis M. Security and privacy objectives for sensing applications in wireless community networks. In: ICCCN 2010: Proceedings of 19th international conference on computer communications and networks. Washington, DC, USA: IEEE Computer Society; 2010. p. 1095–2055.

Dubuisson O. ASN.1-communication between heterogeneous networks. 1st ed. Morgan Kaufmann Publishers; Aug. 2001.

Golle P. Revisiting the uniqueness of simple demographics in the US population. In: Proc. of 5th ACM workshop on Privacy in electronic society, Alexandria, VA, USA, Oct. 2006.

Hartman S, Zhu L. A Generalized framework for Kerberos pre-authentication. IETF internet draft, draft-ietf-krb-wg-preauth-framework-17; June 2010.

Josefsson S. Using Kerberos V5 over the transport layer security (TLS) protocol. IETF internet draft, IETF draft-josefsson-kerberos5-starttls-09.txt; August 2010.

Karopoulos G, Kambourakis G, Gritzalis S, Konstantinou E. A framework for identity privacy in SIP. Journal of Network and Computer Applications Jan. 2010;33(1):16–28.

Kerberos WG. http://www.ietf.org/html.charters/krb-wg-charter.html.

Kim Minkyu. A survey of Kerberos V and public-key Kerberos security, http://www1.cse.wustl.edu/jain/cse571%E2%80%9309/ftp/kerb5/index.html; April 2009.

King NJ, Jessen PW. Profiling the mobile customer? Privacy concerns when behavioural advertisers target mobile phones. Computer Law & Security Review 2010;26(5):455–78.

Lee J, You I, Kim B. caTBUA: context-aware ticket-based binding update authentication protocol for trust-enabled mobile networks. Wiley International Journal of Communication Systems 2010;23:1382–404.

Lootah W, Enck W, McDaniel P. TARP: ticket-based address resolution protocol. Elsevier Computer Networks 2007;51(4):4322–37.

Madson C, Glenn R. The use of HMAC-SHA-1-96 within ESP and AH. IETF RFC 2404; Nov. 1998.

Medvinsky A, Cargille J, Hur M. Anonymous Credentials in Kerberos. IETF Internet Draft, IETF draft-ietf-cat-kerberos-anoncred-00.txt; March 1998.

The MIT Kerberos Consortium. http://www.kerberos.org.

MIT Kerberos Distribution. http://web.mit.edu/Kerberos/.

Narayanan A, Shmatikov V. Robust de-anonymization of large sparse datasets. In: Proc. of the 29th IEEE Symposium on Security and Privacy 2008, pages 111–125, Oakland, CA, USA, May 2008.

Neuman C, Yu T, Hartman S, Raeburn K. The Kerberos network authentication service (V5). IETF RFC 4120, July 2005a.

Neuman C, Yu T, Hartman S, Raeburn K. Encryption and Checksum specifications for Kerberos 5. IETF RFC 3961, February 2005b.

Ohm P. Broken promises of privacy: responding to the surprising failure of anonymization. Available at SSRN: http://ssrn.com/abstract=1450006; Aug. 2009. University of Colorado Law Legal Studies Research Paper No. 09-12.

Pereniguez F, Kambourakis G, Marin-Lopez R, Gritzalis S, Gomez AF. Privacy-enhanced fast re-authentication for EAP-based next generation network. Computer Communications 2010;33(14):1682–94.

Pfitzmann A, Hansen M. A terminology for talking about privacy by data minimization: anonymity, unlinkability, undetectability, unobservability, pseudonymity, and identity management, http://dud.inf.tu-dresden.de/literatur/AnonTerminologyv0.34.pdf; Aug. 2010. v0.34.

PrivaKERB: A user privacy framework for Kerberos. Online material. http://quantum.inf.um.es/privakerb/online-material.pdf.

Raeburn K. Advanced Encryption standard (AES) encryption for Kerberos 5. IETF RFC 3962; Feb. 2005.

Shimaoka M, Hastings N, Nielsen R. Memorandum for multi-domain public key infrastructure interoperability. IETF RFC July 2008;5217.

Sweeney L. Uniqueness of simple demographics in the U.S. Population. Laboratory for international data privacy working paper; 2000.

Syverson P. On key distribution protocols for repeated authentication. ACM SIGOPS Operating Systems Review 1993; 27(4).

Tene O. Privacy: the new generations. Oxford Journal International Data Privacy Law; Nov. 2010:1–13.

Zhu L, Tung B. Public key cryptography for initial authentication in Kerberos (PKINIT). IETF RFC 4556; June 2006.

Zhu L, Leach P, Hartman S. Anonymity Support for Kerberos. IETF Internet Draft, IETF draft-ietf-krb-wg-anon-12.txt; August 2010.

Fernando Pereñiguez García is a Ph.D. student granted by a Séneca Foundation within the Human Resources Researching Training Program 2007 at the Department Information and Communications Engineering, University of Murcia. His research interests are focused in the definition of fast and secure mechanisms which enable seamless handoff between heterogeneous wireless networks.

Dr. Rafael Marin Lopez is a full time assistant lecturer in the Department Information and Communications Engineering at the University of Murcia (Spain). He received B.E., M.E. and Ph.D. degrees in Computer Science from University of Murcia in 1998, 2000 and 2008, respectively. Additionally he is collaborating actively in IETF above all PANA and HOKEY Working Groups. His main research interests include network access authentication, key distribution and security in mobile networks.

Dr. Georgios Kambourakis was born in Samos, Greece, in 1970. He received the Diploma in Applied Informatics from the Athens University of Economics and Business (AUEB) in 1993 and the Ph.D. in information and communication systems engineering from the department of Information and Communications Systems Engineering of the University of Aegean (UoA). He also holds an M.Ed. from the Hellenic Open University. Currently Dr. Kambourakis is a Lecturer at the Department of Information and Communication Systems Engineering of the University of the Aegean, Greece. His research interests are in the fields of Mobile and ad-hoc networks security, VoIP security, security protocols, Public Key Infrastructure and mLearning and he has more than 65 publications in the above areas. He has been involved in several national and EU funded R&D projects in the areas of Information and Communication Systems Security. He is a reviewer of several IEEE and other international journals and has served as a technical program committee member in numerous conferences. Dr. Kambourakis is a member of the Greek Computer Society.

Dr. Stefanos Gritzalis holds a B.Sc. in Physics, an M.Sc. in Electronic Automation, and a Ph.D. in Information and Communications Security from the Dept. of Informatics and Telecommunications, University of Athens, Greece. Currently he is the Deputy Head of the Department of Information and Communication Systems Engineering, University of the Aegean, Greece and the Director of the Laboratory of Information and Communication Systems Security (Info-Sec-Lab). He has been involved in several national and EU funded R&D projects. His published scientific work includes 30 books or book chapters and more than 200 journals and international refereed conferences and workshop papers. The focus of these publications is on Information and Communications Security and Privacy. His most highly cited papers have more than 850 citations (h-index = 16). He has acted as Guest Editor in 20 journal special issues, and has leaded more than 30 international conferences and workshops as General Chair or Program Committee Chair. He has served on more than 200 Program Committees of international conferences and workshops. He is an Editor-in-Chief or Editor or Editorial Board member for 15 journals and a Reviewer for more than 40 journals. He has supervised 10 Ph.D. dissertations. He was an elected Member of the Board (Secretary General, Treasurer) of the Greek Computer Society. His professional experience includes senior consulting and researcher positions in a number of private and public institutions. He is a Member of the ACM, the IEEE, and the IEEE Communications Society "Communications and Information Security Technical Committee".

Antonio F. Gomez Skarmeta is an associate professor at the University of Murcia, Spain. His research interests include advanced networking services and applications over IP networks, network security and mobility. He received an M.Sc. in computer science from the University of Granada and a Ph.D. in computer science from the University of Murcia.