

KAMU: providing advanced user privacy in Kerberos multi-domain scenarios

F. Pereñíguez-García · R. Marín-López ·
G. Kambourakis · A. Ruiz-Martínez · S. Gritzalis ·
A. F. Skarmeta-Gómez

© Springer-Verlag Berlin Heidelberg 2013

Abstract In *Next Generation Networks*, Kerberos is becoming a key component to support authentication and key distribution for Internet application services. However, for this purpose, Kerberos needs to rectify certain deficiencies, especially in the area of privacy, which allow an eavesdropper to obtain information of the services users are accessing. This paper presents a comprehensive privacy framework that guarantees user anonymity, service access unlinkability and message exchange unlinkability in Kerberos both in single-domain and multi-domain scenarios. This proposal is based on different extensibility mechanisms already defined for Kerberos, which facilitate its adoption in already deployed systems. Apart from evaluating our proposal in terms of performance to prove its lightweight nature, we demonstrate its

capability to work in perfect harmony with a widely used anonymous communication system like Tor.

Keywords Identity · Kerberos · Privacy · Security · Inter-domain

1 Introduction

Nowadays, privacy is becoming a sine qua non in network communications to preserve the user's right of concealing personal information. In this context, Internet protocols constitute a serious risk for user privacy since they can transport (either explicitly or implicitly) personal data about individuals. In fact, in the literature, there exists numerous works that analyze the privacy vulnerabilities that appear at different network layers [1–8] when an unauthorized entity can monitor the information sent and/or received (traffic analysis) by network users. This problem is even more important in the so-called *Next Generation Networks* (NGNs) since wireless networks are prone to eavesdropping, so that malicious entities can easily trace the network activity of any user within a certain coverage area.

The collection of personal information about network users enables several malicious acts that clearly violate the user's private sphere [9, 10]. The most obvious is that an eavesdropper is able to obtain access to the user's real identity and track the network activity carried out by a specific user. Even more, in the long term, after systematically collecting this information, the user can be profiled and sensitive information (e.g., user's preferences) may be inferred [11, 12]. For this reason, an essential requirement when dealing with user privacy relies on the provision of *user anonymity* [13] in order to allow a user to remain unidentified.

F. Pereñíguez-García (✉) · R. Marín-López · A. Ruiz-Martínez ·
A. F. Skarmeta-Gómez
Department of Information and Communications Engineering
(DIIC), Faculty of Computer Science, University of Murcia,
30100 Espinardo, Murcia, Spain
e-mail: pereniguez@um.es

R. Marín-López
e-mail: rafa@um.es

A. Ruiz-Martínez
e-mail: arm@um.es

A. F. Skarmeta-Gómez
e-mail: skarmeta@um.es

G. Kambourakis · S. Gritzalis
Laboratory of Information and Communication Systems Security,
Department of Information and Communication Systems
Engineering, University of Aegean, Samos 83200, Greece
e-mail: gkamb@aegean.gr

S. Gritzalis
e-mail: sgritz@aegean.gr

However, in most cases, anonymity is a necessary but not sufficient feature to fight against network communications observers. Even when the user remains anonymous, protocol messages typically contain linkable information like, for example, the typical sequence number. This gives eavesdroppers the ability to correlate the different messages that integrate a network transaction. By exploiting this vulnerability, attackers can profile the user's activity and obtain some general information about behavioral patterns of specific anonymous users [12, 14]. This problem is avoided by ensuring message *unlinkability* [13] in such a manner that attackers are unable to relate protocol messages and, ultimately, to distinguish the presence of specific (anonymous) users.

This paper deals with the provision of user privacy in Kerberos, one of the most well-known and respected Internet protocols for controlling the access to network services. Kerberos provides a *single sign-on* (SSO) operation for accessing services in both single- and multi-domain scenarios. However, it presents some privacy-related deficiencies that may hinder its wide deployment in future wireless communications. In fact, Kerberos explicitly indicates in its messages the identity of the entities participating in the different protocol transactions. More precisely, Kerberos identifies the different participant entities through the so-called *principal identifiers* which are transmitted in cleartext during protocol operation. Therefore, eavesdroppers are able to access the user's real identity and track which services are being accessed, so directly violating the principle of user anonymity. Even worse, Kerberos is unable to satisfy the message unlinkability requirement since it is relatively easy for an eavesdropper to relate all the protocol exchanges followed by a specific user to access a service. This circumstance can be exploited by eavesdroppers, say, to determine the origin realm of foreign users when they first access to a service in the visited realm.

Capitalizing on our previous work in [15], we develop a novel privacy architecture for Kerberos named *KAMU*¹ to achieve not only user anonymity but also complete message exchange unlinkability, which prevents from relating different Kerberos messages sent by the same anonymous user. This is accomplished by obscuring any linkable information conveyed by standard Kerberos messages in such a way that the interoperability of the protocol is mostly preserved. Unlike existing works, KAMU conserves user anonymity and message unlinkability in both single-domain and multi-domain scenarios, with an almost negligible overhead and a low deployment impact compared to the standard Kerberos protocol.

It is important to note that the goal of this work is to solely solve the aforementioned privacy problems at Kerberos level.

Therefore, privacy risks derived from, say, traffic analysis in other network layers are considered out-of-scope. Actually, for this purpose, different solutions [6, 16] and tools [8] are already in place to prevent an important number of problems related to traffic analysis based on packet contexts, sizes and timing. However, for the sake of completeness, we explain in this paper how these kind of privacy attacks can be easily solved by combining KAMU with an anonymous communication system. Furthermore, we demonstrate how this combination can be performed in a transparent manner as happens with other application-layer solutions like HTTP, BitTorrent and SSL [17, 18].

The remainder of the paper is organized as follows. Section 2 describes the standard Kerberos operation, necessary to understand the description of the proposed KAMU privacy framework provided in Sect. 3. Section 4 demonstrates the potential of KAMU to achieve a robust cross-layer privacy preserving system by cooperating with Tor [19], which is one of the most important anonymous communication systems nowadays. Next, Sect. 5 evaluates the additional overhead induced by our privacy extensions in comparison with standard Kerberos. Finally, Sect. 6 discusses related work before extracting relevant conclusions and providing some future work directions in Sect. 7.

2 Background: Kerberos protocol operation

Kerberos [20] is a well-known three-party key distribution protocol widely used for controlling the access to network resources. Kerberos provides a SSO platform thanks to the so-called *tickets*, which allow a user to be authenticated without requiring her to continuously provide her credentials.

Each organization deploying Kerberos must establish its own *realm*. Within this realm, Kerberos messages are exchanged between three types of entities (called *principals*): A *client* that represents a user willing to access a specific service, an *application server* (also referred simply as *service*) providing a specific functionality, and a *Key Distribution Center* (KDC) in charge of authenticating users and distributing tickets.² At the same time, the KDC is integrated by two servers: the *Authentication Server* (AS) and the *Ticket Granting Server* (TGS). While the former is responsible for authenticating the client, the latter is in charge of issuing tickets to access services.

The standard Kerberos protocol operation requires the existence of some trust relationships based on a shared secret key. Table 1 details the different keys used in Kerberos, indicating whether they are statically pre-established or dynam-

¹ Kerberos User Anonymity and Message Exchange Unlinkability.

² In this paper, we use the terms *realm/domain* and *user/client* indistinctly.

Table 1 Trust relationships in Kerberos

Key	Type	Entities	Usage
KDC secret key	Static	AS \leftrightarrow TGS	Protect TGTs
Service secret key	Static	Service \leftrightarrow TGS	Protect STs
Reply key	Static	Client \leftrightarrow AS	Protect AS exchange
TGS session key	Dynamic	Client \leftrightarrow TGS	Protect TGS exchange
Service session key	Dynamic	Client \leftrightarrow Service	Protect AP exchange

ically generated during the protocol execution, the entities knowing the shared secret key and the usage within the protocol.

The basic Kerberos communication consists of three different exchanges (see Fig. 1a). Initially, by means of the *AS exchange* (KRB_AS_REQ/REP messages) (1), the user contacts the AS to request a TGT. When the KDC receives the KRB_AS_REQ message, it generates the *TGS session key* which is included in the TGT and also sent to the client within the KRB_AS_REP message. It is stressed that both messages contain in cleartext the client's identifier which is being authenticated.

Once the client acquires a TGT, it can solicit STs for accessing different services through the *TGS exchange* (2). In addition to the service's identity, the KRB_TGS_REQ message contains the TGT and an authentication tag generated with the *TGS session key*. After successful verification of both TGT and the client credentials, the TGS generates an ST containing the *service session key*. Again, this key is also

distributed to the client by means of the KRB_TGS_REP message. Similar to the KRB_AS_REP, the KRB_TGS_REP contains in cleartext the client's identifier to which an ST has been issued.

Finally, through the *AP exchange* (3), the client authenticates itself against the service. In the KRB_AP_REQ, the client sends the previously obtained ST, together with an authentication tag computed by using the *service session key*. After successful verification of the ST, the client and the service are able to use the *service session key* for protecting their application protocol.

Through a special operation mode called *cross-realm* (see Fig. 1b), Kerberos offers a flexible support to multi-domain scenarios where a client requests access to a service not controlled by the KDC where the client is registered (*home KDC*). Thanks to the definition of trust relationships between TGS/KDCs of different realms, the client follows the path from the home to the visited organization (1 to 3) by obtaining the so-called *cross-realm TGT*. The process finalizes when

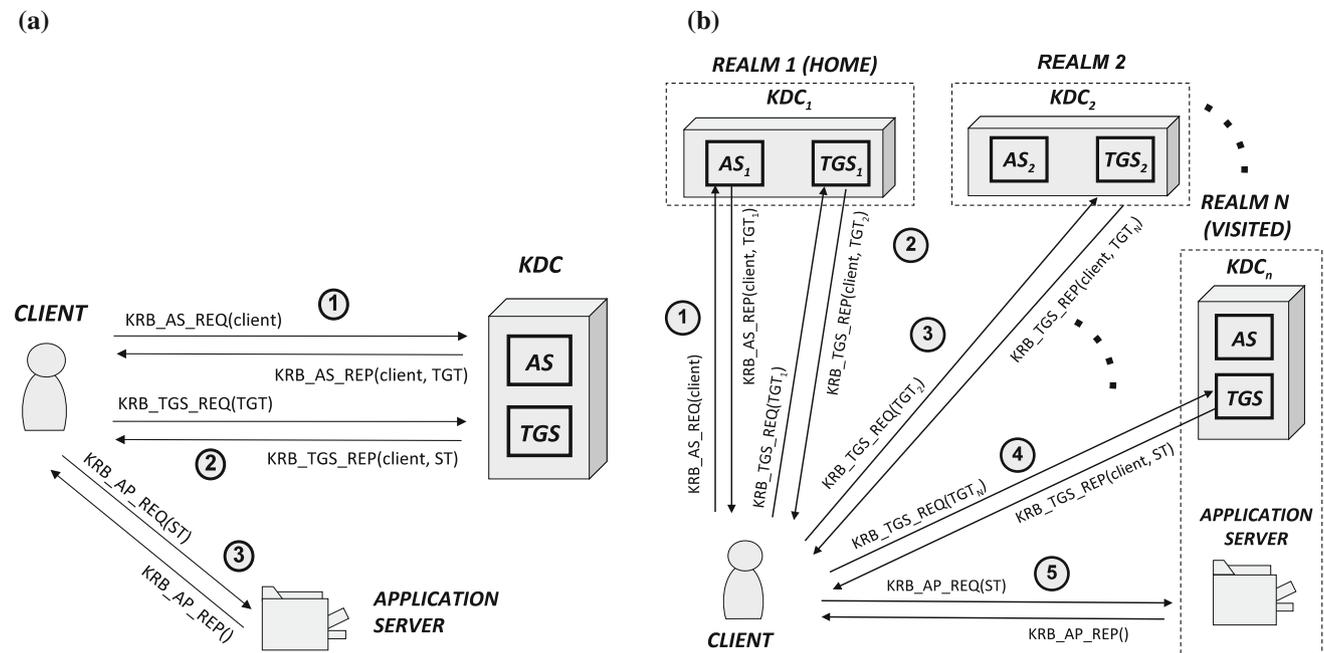


Fig. 1 Kerberos signaling. **a** Kerberos standard signaling, **b** Kerberos cross-realm operation

Table 2 Comparison of existing privacy solutions for Kerberos

	Privacy protection	Mechanism	Deficiencies
Kerberos over TLS [49]	Full obfuscation of the protocol (anonymity and unlinkability)	Kerberos messages transmitted within a TLS tunnel	High deployment cost in multi-domain scenarios
Anonymous tickets [23,24]	User anonymity	New type of tickets associated with the anonymous user (<i>anon@anon</i>)	Requires the use of PKI infrastructures; Solution not compatible with processes like accounting
Generalized framework for Kerberos Pre-authentication [50]	User anonymity	Confidentiality protection of information transmitted in cleartext	Requires the use of PKI infrastructures; Solution not compatible with processes like accounting
PrivaKERB [15]	User anonymity and service access unlinkability	New types of tickets: extended anonymous ticket and TGT	Unable to protect against service access linkability; Deficient privacy protection in multi-domain scenarios

the client contacts the visited KDC (4) and obtains a valid ST for the target service (5).

Kerberos is a protocol under constant evolution. In fact, organizations like the Internet Engineering Task Force (IETF) Kerberos Working Group [21] or the Kerberos Consortium [22] are leading the improvement of the protocol aiming to promote Kerberos as the universal solution for service access authentication. This activity is facilitated by the Kerberos extensible nature to cope with new functionalities. Some outstanding features are the possibility of defining new flags, new authorization elements or pre-authentication data (hereafter *pdata*) that allow the transportation of valuable information for new applications at either message or ticket level.

3 KAMU: advanced privacy architecture for Kerberos

This section presents KAMU, our advanced privacy architecture for Kerberos. For this purpose, first we present the different privacy risks that users are exposed to during the Kerberos protocol execution. Then, we describe how these vulnerabilities are mitigated by KAMU. Apart from detailing the privacy extensions, we elaborate on the operation of the solution in both single-domain and multi-domain scenarios in order to demonstrate its ability to preserve the privacy of the user.

3.1 Problem statement

As previously mentioned, Kerberos identifies the different participant entities through the so-called Kerberos *principal*

identifiers. Principals are globally unique names which have the form:

```
component[/component]...@realm_name
```

As we can observe, the principal is integrated by three types of components, namely: *principal name*, *instance*, and *realm name*. The *principal name* can be either a username or service name. This component is followed by an optional *instance* which is useful, for example, to distinguish instances of the same service on different machines (by including the hostname of the machine as instance component). The principal identifier is completed by attaching (after the “@” symbol) the specific *realm name* where the entity is registered. For example, assuming the realm UM.ES, *peter@UM.ES* and *printer/server.um.es@UM.ES* are valid identifiers of a user and a service, respectively.

As described in Sect. 2, Kerberos transmits in cleartext principal identifiers associated with clients and services. For example, despite most of the information contained within an ST is encrypted with the service secret key shared between the service and the TGS/KDC, some information is not protected and thus accessible to eavesdroppers. In particular, the service’s identifier for which the ticket has been issued is transmitted in cleartext. Similarly, Kerberos messages also transport identity information which is transmitted in cleartext. Namely, the KRB_AS_REQ and KRB_AS_REP messages convey the client’s identifier which is being authenticated by the AS module of the KDC. The same situation happens in the KRB_TGS_REQ/REP exchange where an

observer can also learn the identity of the service the client is willing to access.

This situation leads to the apparition of important vulnerabilities that result in a violation of the client's private sphere. For example, the most obvious one is that an eavesdropper can easily deduce which services are accessed by a specific user. In the long term, the systematic collection of this information can reveal valuable information such as client's preferences and habits. By preserving the *anonymity* of the client (e.g., allow clients to employ an anonymous identity), some privacy problems that are present in Kerberos can be mitigated. Nevertheless, even when users remain anonymous, the intrinsic operation of Kerberos still raises some privacy issues. For example, the basic Kerberos specification [20] proposes that a specific TGT can be re-used each time a user contacts the KDC for requesting an ST. As a consequence, an observer is able to infer the set of services accessed by a specific (even anonymous) client, thus violating the principle of *unlinkability* [13].

This privacy property is also violated by the basic service access model defined in Kerberos. As explained in Sect. 2, Kerberos defines an atomic operation where the client engages in a message exchange with the KDC to solicit a ticket that is presented to the service in a subsequent exchange. As already pointed out, this operation is applied twice in the protocol. Firstly, in the AS exchange which is destined to provide the client a TGT to be used in the TGS exchange. Secondly, in the TGS exchange where the client obtains an ST that is sent to the service through the AP exchange. Since certain pieces of information contained in a ticket are conveyed in cleartext, this particularity allows an eavesdropper to relate the different messages sent or received by a client to access a service.

Taking into account these privacy inefficiencies, our objective is to develop a complete privacy solution for Kerberos, namely KAMU, satisfying two important requirements. First, *user anonymity* must be assured in such a manner that eavesdroppers tracing a Kerberos communication are unable to learn the real identifier (e.g., the username) of the specific user involved in the transaction. Second, the solution must prevent aggressors from relating the different messages sent and/or received by a specific user, thus providing *message exchange unlinkability*. It is important to note that this unlinkability property can be considered as a more sophisticated kind of anonymity since eavesdroppers are not only unable to infer the identity of a user, but also to determine any relationship between the exchanged messages and the user (i.e., correlating the messages into transaction records for each user).

To provide the desired level of privacy, the proposed architecture must define some mechanisms to achieve the following goals:

- (G1) *Conceal the user's identifier*. Since Kerberos messages transport in cleartext the identifier of the involved parties, an eavesdropper can easily identify the user participating in the communication. For this reason, the solution is required to hide any identification-related data accessible to eavesdroppers.
- (G2) *Obscure any linkable information*. Kerberos messages contain different pieces of information that, by linking them, eavesdroppers can relate the different messages exchanged by a certain (even anonymous) user. As explained, the analysis of the traced messages can reveal valuable information not only regarding the anonymous user itself (e.g., preferences or habits), but also about the user's Kerberos realm (e.g., most attractive services for roaming users).

3.2 Description

Some features of our proposal are adopted from existing solutions dealing with privacy protection in Kerberos, which are analyzed in Sect. 6. For example, to hide the user's identifier transmitted in Kerberos messages (goal G1), we use the *anonymous ticket* concept presented in our previous work [15] which, in turn, is extended from [23,24].

Anonymous tickets are regular tickets which have the anonymous flag activated (*Flag_A*) and are associated with the anonymous client *anon@anon*. But this is only from an eavesdropper's perspective because the user's identifier is actually included in the *authorization-data* field defined by the Kerberos specification [20] for the tickets. Since the content of this field is only accessible to the entity to which the ticket has been generated, the client's identifier is only revealed to authorized parties like KDCs or services, which may require it to perform, for example, charging operations. Additionally, anonymous tickets rely on a pseudonym-based user identification scheme. Pseudonyms, which are valid during a specific period of time, are bound to the home TGT lifetime: Each time a new home TGT is acquired, and the user renews the pseudonym used as identification. Furthermore, since the home KDC controls the generation and distribution of pseudonyms to the client, the relationship between the pseudonym and real user identity is only known to the home KDC. This is plausible since only the home domain of the user can be considered trusted.

Regarding the concealment of linkable information present in Kerberos messages (goal G2), in Sect. 3.1 we identified two critical pieces of information. First, the SSO model proposed by Kerberos is based on the reuse of the TGT acquired by the user after the initial authentication during the KRB_AS_REQ/REP exchange. In other words, each time a user requests access to a service, and the TGT is used to

contact the KDC and solicit an ST, thus being quite easy for an attacker to learn the different accessed services. Second, the service access model is also susceptible to linking attacks. Kerberos defines an operation where a client willing to access a service must obtain from the KDC an ST (KRB_TGS_REQ/REP) that is subsequently presented to the service (KRB_AP_REQ/REP). In this case, the ST is a clear hint for an attacker to monitor users accessing a service.

To avoid TGT-based linking attacks, we use the concept of *self-renewed TGT (srTGT)* presented in [15]. Self-renewed TGTs are regular TGTs used by clients to request STs that are produced and consumed by the TGS. Nevertheless, an srTGT can be only used to request access to one service. For this reason, during the KRB_TGS_REQ/REP exchange, the TGS not only distributes the ST, but also a fresh srTGT to the client that instantly replaces the old one. The srTGT is sent by the TGS to the client in the KRB_TGS_REP message in a confidential manner so that eavesdroppers are unable to relate the different srTGTs used by a certain anonymous user.

However, the use of srTGTs is a necessary but not sufficient condition to achieve a complete message exchange unlinkability. This is because the ticket generated by the KDC is distributed to the client in cleartext during a first exchange and used in a subsequent one(s). For example, this vulnerability can be easily appreciated in Fig. 1a, which describes the basic Kerberos operation. Initially, during the KRB_AS_REQ/REP exchange, the client receives a TGT generated by the KDC. This ticket is distributed in cleartext and eavesdroppers can observe it. When the client desires to access a specific service within the Kerberos realm, an ST is requested through a KRB_TGS_REQ/REP exchange. Eavesdroppers tracing the communication can easily link this transaction (TGS exchange) with the previous one (AS exchange) since the TGT previously distributed by the KDC is now sent by the client in the KRB_TGS_REQ message. The same situation happens with the ST since this ticket is distributed to the client in the KRB_TGS_REP message and used by the client in the KRB_AP_REQ when accessing to the service. In short, both TGT and ST are linkable information that can be used by attackers to infer the different Kerberos transactions in which a client is involved.

To remediate this vulnerability, which results in a violation of the unlinkability privacy principle, it is necessary the definition of some mechanism to hide the ticket distributed by the KDC (i.e., the TGT in the KRB_AS_REQ/REP message and the ST in the KRB_TGS_REQ/REP message), in order to prevent eavesdroppers from learning the different tickets obtained by a client. In particular, KAMU addresses this problem by proposing a novel procedure to carry out ticket distribution in Kerberos. Roughly speaking, the idea consists in hiding through encryption the ticket sent by the KDC to the client, so that eavesdroppers cannot observe it and only the client

can recover it. The proposed ticket distribution process distinguishes two types of tickets which are used in tandem:

- *Authentic ticket.* This ticket represents the standard ticket generated by the KDC and requested by the client for accessing a service. The authentic ticket can be a TGT or an ST distributed by the KDC to the client within a KRB_AS_REP or KRB_TGS_REP message, respectively. Nevertheless, compared to the standard Kerberos specification, the distribution of the authentic ticket differs in two aspects. First, instead of being distributed in cleartext, the authentic ticket is confidentiality and integrity protected so that attackers can neither access nor modify it. Second, the authentic ticket is located in the *padata* field [20] of the message (either KRB_AS_REP or KRB_TGS_REP), which is an extensible field defined by Kerberos to transport additional data to cover new functionalities.
- *Fake ticket.* A fake ticket is a new type of ticket where, except the *flags* field, all the fields belonging to the protected part (named *EncTicketPart* [20]), contain invalid information. For example, these fields can be null or randomly initialized. Fake tickets are distinguished from authentic tickets thanks to the presence of the *fake flag*. A fake ticket can be sent by the KDC to the client within a KRB_AS_REP or KRB_TGS_REP message. Following the standard ticket distribution process, a fake ticket is located in the standard *ticket* field [20].

The ticket distribution process defined in KAMU combines both authentic and fake tickets. Let us assume a KDC willing to distribute a TGT to a client within a KRB_AS_REP message. The real TGT generated by the KDC following the standard Kerberos operation represents the authentic ticket. As previously explained, this ticket will be confidentiality and integrity protected and included within the *padata* field of the KRB_AS_REP message. Apart from the real TGT, the KDC generates a fake ticket containing invalid data and the fake flag activated. This fake ticket is placed in the *ticket* field, which is the usual field defined by the Kerberos specification to be used by the KDC to send the TGT to the client in KRB_AS_REP message. Thanks to this strategy, an eavesdropper capturing the communication will be unable to access the real TGT being sent to the client. On the one hand, if the eavesdropper is only familiar with standard Kerberos protocol, the field of the KRB_AS_REP message where an eavesdropper expects to find the TGT contains the *fake ticket*. On the other, in case the eavesdropper understands KAMU extensions and examines the content of the *padata* field of the message, it will be also unable to access the real TGT since it is confidentiality and integrity protected.

Note that the same procedure is applied during the ST distribution within KRB_TGS_REP message: The real ST is treated as the authentic ticket and a new fake ticket is generated and distributed following the conventional process. In summary, as we can observe, the new ticket distribution process based on authentic and fake tickets solves the ticket-based linkability problem present in standard Kerberos protocol.

3.3 Operation

In the following, we describe the operation of our privacy architecture in both single-domain and multi-domain scenarios. We would like to clarify that we mainly elaborate on the privacy extensions necessary to implement the fake ticket feature, which is a novel mechanism to implement unlinkability. For details about features which are present in our proposal but inherited from previous research works such as anonymous tickets or self-renewed TGTs, we encourage the reader to refer to our previous work in [15]. For the sake of clarity, we present first the notation we employ to describe the information flow:

- $name_i@realm$: The i th principal name employed by a user in a given Kerberos realm.
- $TGT_X^i[Y]$: The i th TGT for KDC X that contains the information X protected to preserve confidentiality and integrity.
- $ST_X[Y]$: The service ticket for service X that contains the information X confidentiality and integrity protected.
- N_i : The i th random number generated by the client.
- $PA - NAME(Y)$: A Kerberos pre-authentication data type named NAME containing the information X .
- $AD(X, Y, \dots)$: The pieces of information X, Y, \dots which are transported in the authorization-data (AD) field of a ticket by using different authorization data elements.
- $FlagX$: A flag named X .

3.3.1 Single-domain scenario

Figure 2 shows the operation of the KAMU architecture when users solicit access to services located in their home domain. Initially, the user is authenticated through the KRB_AS_REQ/REP exchange ($G1$). In the KRB_AS_REQ, the user employs the pseudonym CP_1 as identification instead of the real identity. On the reception, the KDC determines that privacy protection must be enabled since the user has employed a pseudonym to initiate the exchange. By means of a policy configured on the KDC (out of scope), the type of privacy to be applied is determined. Supposing that other privacy schemes (e.g., PrivaKERB [15]) may be supported by the communicating entities, we need a way to notify the client about the privacy mode they need to activate

(KAMU in our case). This is achieved by using a new padata type called PA-MODE which contains a numeric value identifying the KAMU framework. Additionally, the KDC generates a new random pseudonym (CP_2) to be used by the client in the subsequent KRB_AS_REQ/REP exchange. This pseudonym is distributed to the client within the PA-PSEUD padata type. The distribution of these pieces of information must be protected in such a manner that eavesdroppers cannot observe or modify their values. For this purpose, both PA-MODE and PA-PSEUD are sent within the PA-PRIV padata type, which is able to provide confidentiality, integrity and freshness to the set of padata that contains.

Moreover, the KDC sends to the client via the KRB_AS_REP message a TGT to be used in the future for requesting STs. This TGT follows the anonymous ticket format where the ticket is assigned to the anonymous user ($anon@anon$) and the real identity is included as authorization data (AD). Additionally, the ticket contains as authorization data the privacy operation mode ($KAMU$) assigned to the user. As we can observe, the anonymous TGT ($TGT_H^1[FlagA, \dots]$) being used is not transmitted in clear. Rather the enhanced ticket distribution mechanism based on authentic and fake tickets is employed. According to this, the anonymous TGT is treated as the authentic ticket, thus being transported through a new padata type named $PA-TICKET$. Together with TGT_H^1 , this padata contains information (omitted for simplicity) that will normally appear in the $enc-part$ field of the KRB_AS_REP and necessary for the client (e.g., TGT lifetime). As we can see, the $PA-TICKET$ belongs to the sequence of padata protected by the $PA-PRIV$ padata. In other words, the $PA-TICKET$ padata containing the authentic TGT is confidentiality and integrity protected, so that the ticket is unobservable for eavesdroppers tracing the communication. In fact, eavesdroppers can only access to the fake TGT $FakeTGT[FlagF \dots]$ (with the fake flag enabled— $FlagF$), which is located in the standard field (visible for everyone) reserved for transporting the TGT in the KRB_AS_REP message. On the reception, the client ignores the fake ticket (thanks to the $FlagF$) and realizes that the authentic TGT is placed in the $PA-TICKET$ padata type.

Next, when the client needs an ST for accessing service $S1$, it issues a KRB_TGS_REQ ($G2$) that contains the previously acquired TGT and the identity of service $S1$. When the KDC processes the presented TGT, it becomes aware that KAMU privacy protection is enabled for client CP_1 . When the message is successfully validated, the KDC generates two anonymous tickets: a) an anonymous self-renewed TGT $srTGT_H^1$; b) an anonymous ST $ST_{S1}[FlagA \dots]$. On the one hand, the self-renewed TGT is sent to the client by using the $PA-PRIV$ for achieving confidentiality and avoiding attackers to link TGT_H^1 with $srTGT_H^1$. On the other hand, the ST distribution follows the enhanced ticket transportation procedure based on authentic and fake tickets: The authentic ST

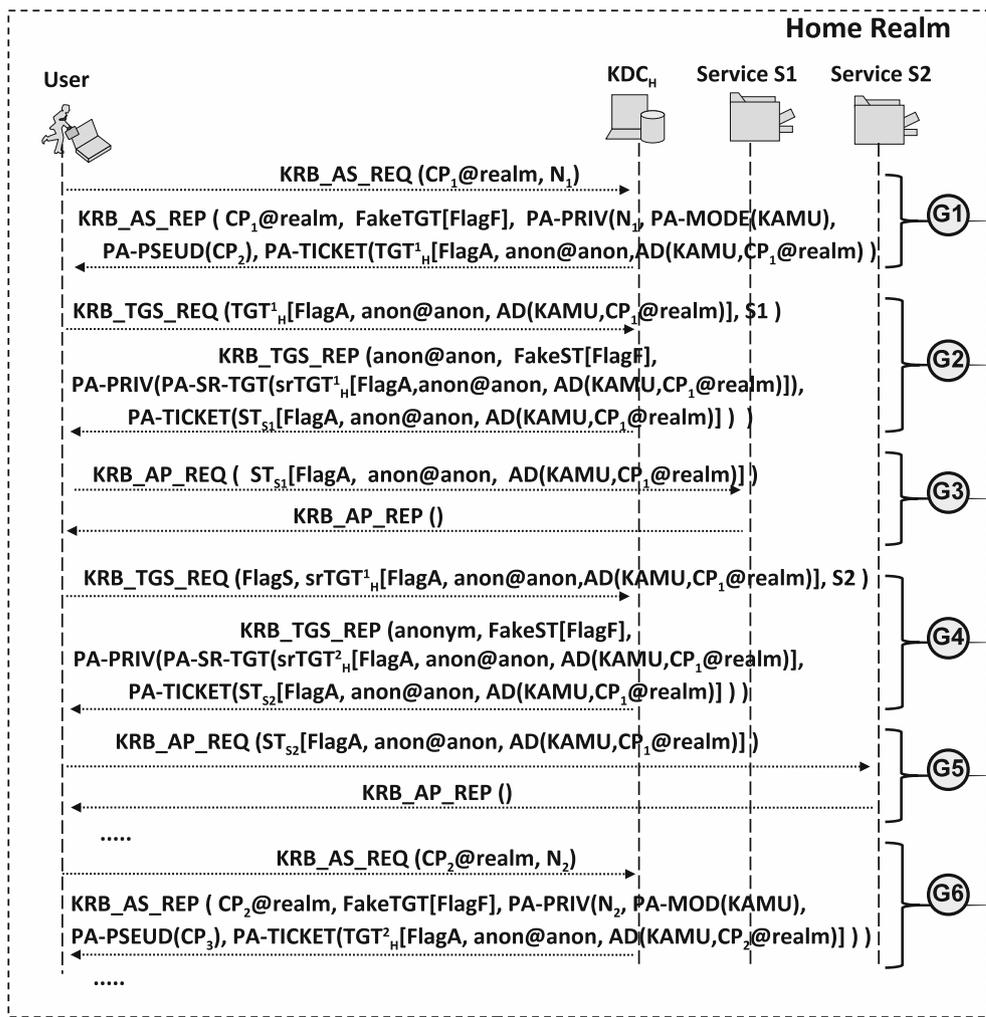


Fig. 2 KAMU operation in single-realm scenario

is conveyed within the PA-TICKET padata (confidentiality protected through the PA-PRIV padata) while a fake ST is placed in the standard field of the KRB_TGS_REP message reserved to transport the ST.

As a result, thanks to the enhanced ticket distribution procedure employed in the KRB_TGS_REP message, an eavesdropper can only observe the fake ST. The real one remains hidden by means of the PA-PRIV padata and can be only recovered by the legitimate client. This is done using the same key (typically the secret key shared between the client and the TGS) employed to process the *enc-part* field of the response. Therefore, for the time being, it is infeasible for an eavesdropper to know the ST delivered to the client. For this reason, when the client presents the received ST to the service through a KRB_AP_REQ/REP (G3), an eavesdropper cannot deduce that this operation is performed by the same anonymous user involved in the previous KRB_TGS_REQ/REP because it has no means to infer that ST was previously acquired by the same client.

The same process is applied when the user requests, for example, access to another service S2. Thanks to the KAMU privacy framework, we allow users to remain completely anonymous during their Kerberized activity. Additionally, message exchange unlinkability is ensured in such a manner that eavesdroppers are unable to relate the different Kerberos exchanges in the protocol. Figure 2 summarizes the above analysis. That is, an eavesdropper tracing the communication cannot deduce that the different Kerberos exchanges (from G1 to G6) are performed by the same anonymous user.

3.3.2 Multi-domain scenario

In a cross-realm authentication, our objective is to minimize the deployment cost of the solution and thus favor its adoption. For this reason, the proposed privacy architecture is able to operate in multi-domain scenarios where only the home and visited KDCs must be updated to support our privacy extensions. Thus, intermediary KDCs, placed in-between the

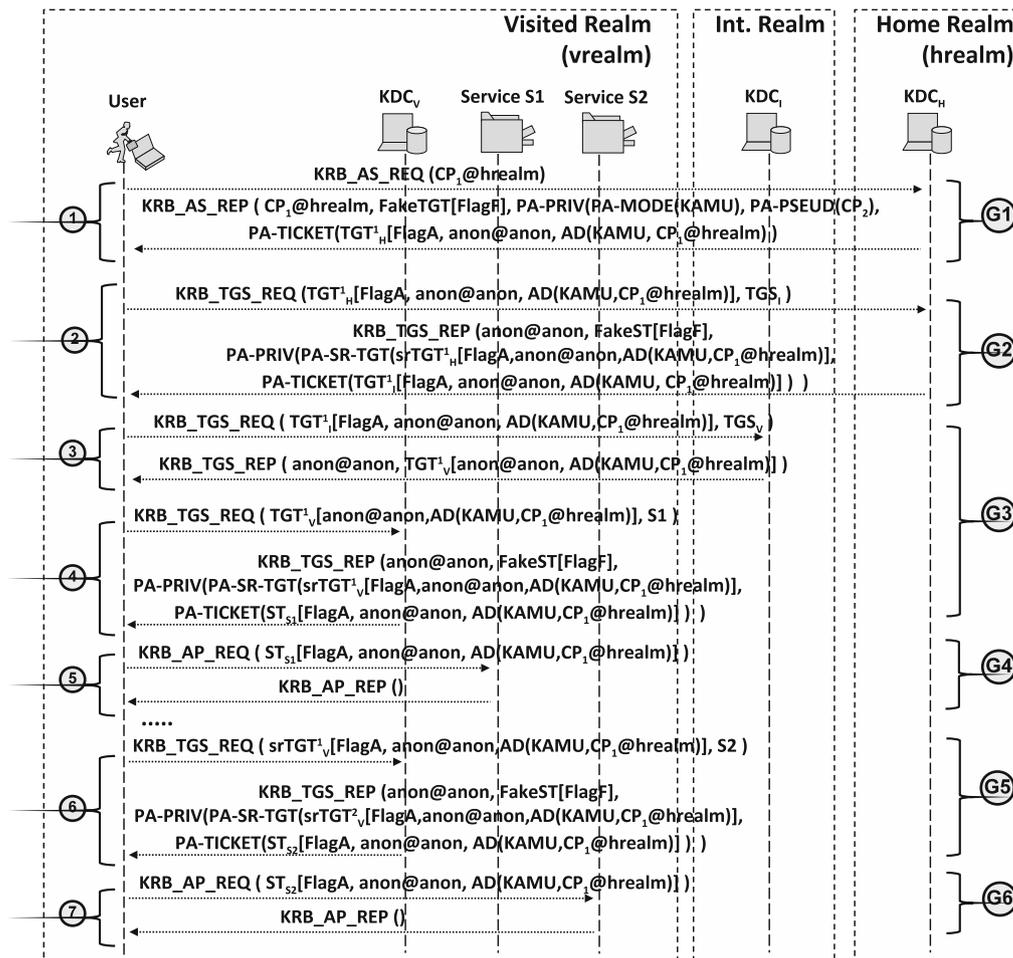


Fig. 3 KAMU operation in a typical cross-realm scenario

home and visited KDCs in a Kerberos cross-realm infrastructure, can use existing implementations based on the Kerberos specification [20] without support of our privacy extensions.

Figure 3 depicts the KAMU operation in a scenario where a user requests access to service S1 located in a foreign domain. Initially, the AS and TGS exchanges with the home KDC (KDC_H) are equal to those explained for the single-domain case (see Sect. 3.3.1). As observed, an eavesdropper cannot infer that the anonymous user requesting a TGT for KDC_I (G2) is the same person as the one previously authenticated (G1). Furthermore, this first TGS exchange (G2) cannot be related to the second TGS exchange (G3) where, following the standard Kerberos protocol, the client contacts KDC_I to request a cross-realm TGT for KDC_V . Since KDC_I does not support our privacy extensions, TGT_V^1 is distributed in the KRB_TGS_REP without using the enhanced ticket distribution procedure based on authentic and fake tickets, thus being visible to anyone. As a consequence, it allows eavesdroppers to relate the second (G3) with the third TGS exchange (G4), where the client solicits an ST for service S1. Nevertheless, only these two exchanges are affected because

KDC_V restores back the use of KAMU privacy protection and the KRB_TGS_REP message communicates ST_{S1} under the protective umbrella of the enhanced ticket distribution procedure. For this reason, from the eavesdropper’s point of view, access to service S1 (G5) cannot be associated with previous exchanges. Finally, subsequent exchanges to access services in the visited realm (G6 and G7) cannot be related to the same (anonymous) user thanks to KAMU. As a result, because of the fake ticket feature which provides exchange unlinkability, an eavesdropper tracing the communication cannot deduce that the different Kerberos exchanges (from G1 to G6) are performed by the same anonymous user.

The application of the KAMU framework is particularly beneficial in multi-domain scenarios. Due to the message exchange unlinkability property, eavesdroppers cannot trace the whole process followed by the user to obtain a service ticket in the foreign realm. Hence, attackers cannot obtain information regarding the roaming of the users like, for example, which are the preferred services solicited by foreign users in a realm. Furthermore, since we do not require intermediary KDCs to support our privacy extensions, our solution can be

easily integrated in existing deployments. Despite this advantage allows eavesdroppers to trace those pair of messages exchanged from the first intermediary KDC to the visited KDC, it is important to note that eavesdroppers neither learn the identity of the user nor the realm to which the user belongs to. Furthermore, since this situation only affects to this specific set of messages in the cross-realm infrastructure but not when the user is contacting the service, an eavesdropper cannot obtain valuable information to construct anonymous service access profiles. Under this prism, the value of such an attack is relatively minimal. Therefore, the decision of not requiring intermediary KDCs to support the KAMU privacy extensions is a trade-off between deployment cost and the level of unlinkability provided to the user.

4 Achieving a cross-layer privacy preserving communication system

By extending a previous work presented by authors in [15], we develop a fully fledged privacy framework for the Kerberos application protocol. The novelty of this solution relies on its ability to achieve not only anonymity, but also complete message exchange unlinkability, thus achieving a full obfuscation of the protocol from an eavesdropper viewpoint. As we have demonstrated, these privacy principles are satisfied in both single-domain and multi-domain scenarios.

KAMU can be considered as an amendment to the standard Kerberos protocol to solve the privacy flaws that compromise the user's right of not disclosing personal data to unauthorized third parties. That is, KAMU aims at assuring that the information transported within Kerberos messages and visible for observers will neither reveal the real user's identity nor allow eavesdroppers to trace Kerberized user's activity by linking exchanged messages between client–KDC and client–service.

Nevertheless, as mentioned in [8,25], privacy can be compromised through the various pieces of information gathered at different network layers. With KAMU, we have solved the problems at Kerberos level, that is the application layer. However, if a comprehensive solution is meant to be provided, we have to complement KAMU with privacy solutions for the TCP/IP layer since this layer leaks valuable information that directly put user privacy at stake. Namely, for this purpose, the information that could be used at this layer is as follows: IP address, ports, domain name, geolocalization information, packet contexts, sizes and timing [8,26]. Even by means of the round-trip time of user's connection, one could identify a sender from others. For the interested reader, more details on possible attacks based on traffic analysis at this level can be found in [26–29].

Fortunately, in the literature, one can find numerous research works addressing these issues. From these works,

we can pick out Chaum's mix networks [30], Tor [19], Crowds [31], Tarzan [32] and Rebolledo-Monedero et al.'s proposal [33]. These are solutions that aim to specify anonymous communications systems that are defined independently of the protocols of upper layers so that they can be used in conjunction with them. A deeper analysis of these kind of solutions can be found in [6,16,34].

To prevent privacy attacks beyond those based on Kerberos message content, KAMU must be combined with some of these solutions solving the privacy problems stemming from the TCP/IP layer. The combination can be performed in a straightforward way since TCP/IP layer solutions are conceived to operate with any application-level protocol (and in particular with Kerberos). In the following, we describe as an example how a cross-layer full-fledged privacy solution can be achieved by integrating KAMU with Tor, which is one of the most used anonymous communications systems [35].

4.1 Brief overview of Tor

Tor [19] is a low-latency anonymous communication system based on the onion routing mechanism [36]. In Tor, the communication between the sender and the receiver happens through the so-called *circuit*, which is a chain of intermediary nodes that receive and forward information in an encrypted way. These intermediary nodes only know their predecessor and successor in the chain, but no other nodes in the circuit. In particular, the onion routing mechanism applied by Tor distinguishes three different entities:

- *Onion Proxy* (OP). It is a proxy that is executed locally by a user willing to establish an anonymous communication with another entity by using the Tor system. Among other tasks, OPs are responsible for managing connections requested by user applications and establishing circuits.
- *Onion Router* (OR). This entity plays the role of an intermediary node of the circuit in charge of relaying information received from other ORs and OPs. ORs share a *session key* with the OP that is used to protect the exchanged information.
- *Directory Server* (DS). This server assists the Tor operation by providing OPs with an updated list of ORs available that can be used to establish circuits as well as the *onion keys* that must be used with each router to negotiate the establishment of circuits.

The Tor operation begins when a user (sender) solicits the transmission of certain information to another node in the network (receiver). This request is received by the OP located in the user local machine, which will establish a circuit toward the receiver by following two different steps. First, the OP

obtains from the DS the list of available ORs in the Tor network. From this list, the OP randomly chooses the set of ORs that will integrate the path (circuit) from the sender to the destination node. After that, the OP will establish a session key with each OR of the circuit. This session key is negotiated through a *Diffie–Hellman* handshake protected with the use of public key cryptography (each OR is set up with public/private key pairs named *onion keys*).

Once the circuit is established, the transmission of information from the sender to the destination node (receiver) can be performed in an anonymous manner. The Tor system makes the transmission of information through the circuit to be performed in fixed-size *cells* of 512 bytes. The cells sent from the OP to the receiver are iteratively encrypted with the session keys shared with ORs integrating the circuit. During the transmission of the cell along the circuit, each OR decrypts the incoming cell (i.e. removes one encryption layer) using the session key shared with the sender, validates its content and forwards the resulting cell to the next OR. This process, which is similar to peeling an onion layer by layer, ensures that each node in the path does not know about the sender and the receiver. When this process is performed by the last OR of the circuit, the resulting decrypted cell will contain the original application data that is forwarded in clear using standard TCP/IP routing to the receiver.

When the receiver replies to the sender, the generated cells follow the reverse path in a similar manner. In this case, each OR receives the cell from the previous hop (OR) in the circuit, encrypts its content using the session key shared with the sender and forwards the cell back to the next hop in the circuit. When this cell arrives to the sender, it will have to remove the different layers of encryption in order to access to the original unencrypted data generated by the receiver.

As we have previously mentioned, Tor is a famous low-latency anonymous communication system which is currently used for HTTP, BitTorrent and SSL [17, 18]. Nevertheless, it has been conceived to be seamlessly integrated with other TCP-based application protocols requiring a low-latency communication. In our case, we deal with the case where Kerberos is used as an application protocol. In the following, we demonstrate how KAMU and Tor can work in perfect harmony, resulting in a robust cross-layer privacy system.

4.2 Integrating KAMU with Tor

Figure 4 depicts the operation of KAMU when using Tor as underlying low-latency anonymous communication system. The integration between KAMU and Tor requires the KAMU traffic to be sent to the OP. Since this proxy can only handle TCP packets, KAMU must be configured to use TCP as transport layer protocol. For sending KAMU traffic to the OP, we can use, for example, TCP portmapping with *3proxy* [37, 38].

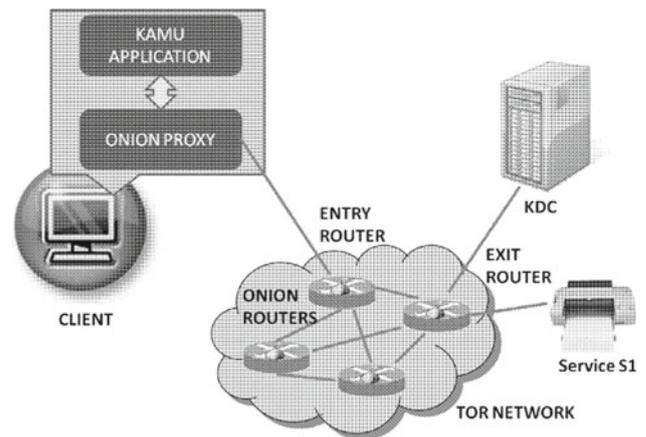


Fig. 4 Integrated architecture combining KAMU and Tor

Alternatively, in the event that we prefer sending Kerberos by using UDP packets, we should establish an anonymous tunnel with the receiver supporting UDP and use the tunnel to run KAMU. This could be made by using the tool *Whonix* [39]. This combination makes that the traffic sent by Tor is protected at TCP/IP layers. Furthermore, with the use of KAMU, Kerberos data are protected at boundaries of Tor (between the last onion router and the KDC or the serviced).

As an example, with the integration of KAMU and Tor, the steps shown in Fig. 2 now are the following. The user generates the *KRB_AS_REQ* and sends it to the OP. This establishes a circuit with a number of ORs (this circuit can handle many TCP streams and the OP builds a new circuit periodically). Then, OP sends the message to the Entry OR, which re-sends it to the next OR in the path. When the message arrives at Exit OR, this OR sends it to the KDC. The KDC processes the message, generates the *KRB_AS_REP* and sends it to the Exit OR. Then, the message follows the reverse path to arrive at the Entry OR. The Entry OR sends the message to the OP and, finally, the user receives the message. The process is the same for the exchange of the *KRB_TGS_REQ* and *KRB_TGS_REP*. Once, the exchange with the KDC has been made, the process to exchange messages with the Service S1 and Service S2 (messages *KRB_AP_REQ* and *KRB_AP_REP*) is similar to the process we have just described for the exchange of messages with the KDC.

4.3 Privacy analysis

The combination of KAMU with Tor results in a cross-layer privacy preserving low-latency communication system protecting against a considerable number of privacy attacks. More precisely, thanks to the use of Tor, the following privacy attacks are avoided [28, 40]:

- *Message coding*: By using different layers of encryption, Tor provokes messages to change their coding during the transmission through the Tor network.
- *Message size*: This attack is prevented in Tor by forcing the use of fixed-length cells (512 bytes). Therefore, all the information circulating through the Tor network is undistinguishable for an attacker inspecting the message length.
- *TCP/IP packet contexts*: Tor is a privacy solution for the TCP/IP layer and, consequently, is able to prevent attackers from developing privacy attacks based on packets contexts (e.g. IP address of the sender, sequence number or application port included in the TCP header, etc.). This is obviously achieved thanks to the transmission of information in an encrypted manner through random circuits of ORs.
- *Timing*: Tor greatly alleviates privacy attacks based on message timing. Despite under the global attacker model we can find a timing attack in Tor [41], this attack is not plausible since it supposes the adversary is able to observe all nodes in the network. Similarly, under the weaker model threat attack (without global adversary), there are some possible attacks as noted by authors in [42, 43]. However, these attacks can be easily mitigated by applying complementary mechanisms such as adaptive padding, the prevention that all nodes gather a list of all nodes or the insertion of cover traffic so that the network cannot find the stream's signature [42–44].
- *Collusion*: Tor defeats against this kind of attack thanks to its completely distributed design. Information is routed through circuits integrated by a set of ORs that only are aware of the predecessor and successor in the chain.

Additionally, it is worth noting that Tor does not provide protection at the boundaries of the system. On the one hand, the communication between the sender and the OP does not suppose a vulnerability for the system since OPs are expected to be co-located in the sender's machine, so the communication happens locally between processes in the same machine. Nevertheless, the communication between the last OR and the receiver is not protected and attackers could exploit this situation to conduct some privacy attacks based on the packet content. For this reason, Tor requires applications to cooperate in the preservation of user privacy in such a manner that they do not transport in clear privacy sensitive information. In our case, when using Kerberos as application protocol, we solve this problem by applying KAMU, our contribution to solve any privacy attack derived from analyzing Kerberos messages contents. In particular, KAMU ensures that both user anonymity and message exchange unlinkability are preserved.

In summary, we can point out that with the combination of KAMU with Tor, we provide a comprehensive solution that

guarantees cross-layer privacy protection of Kerberos-based applications. Despite we have selected Tor as a representative low-latency anonymous communication system widely analyzed and used nowadays, it is worth clarifying that our intention is to demonstrate the feasibility of complementing our solution (KAMU) with other mechanisms to achieve cross-layer privacy protection. Following a process similar to that described in Sect. 4.2, KAMU could be straightforwardly combined with other privacy solutions for TCP/IP level such as Tarzan [32] or MorphMix [45] (which is resistant to some of the timing attacks previously mentioned [43]).

5 Performance evaluation

In this research work, we propose some extensions to the Kerberos protocol that allow to preserve the privacy of the user. It is obvious that these extensions will introduce some additional latency compared with the base protocol, thus expecting access and/or service time to increase. Therefore, the key question is whether the additional cost imposed by our privacy extensions is affordable for the different scenarios where Kerberos can be used to enforce a controlled access to network resources. To perform this evaluation, we have developed an implementation prototype that is used to assess the performance of our privacy framework.

It is important to stress that the assessment we have conducted exclusively analyzes the additional overhead imposed by the KAMU privacy extensions, and it is not considered the use of an anonymous communication system like the one explained in Sect. 4 (Tor). In fact, the additional latency imposed by the anonymous communication system has already been evaluated by other works [46, 47] and significantly depends on the specific system being used.

5.1 Deployed testbed and implementation details

The KAMU privacy framework prototype has been implemented using the open-source MIT Kerberos distribution v.1.6.3 [48]. The implementation process has concentrated on two main aspects. On the one hand, the KDC implementation has been extended to support the proposed privacy extensions. Special effort has been devoted on establishing the fake ticket distribution process. Also, since the KAMU framework adheres to the standard Kerberos protocol and uses the extensibility mechanisms offered by the protocol per se (definition of new flags, new pre-authentication data types and new authorization data), it is important to mention that the implementation process does not present excessive complexity. On the other hand, the Kerberos *Application Program Interface* (API) used by client and services has been also adapted to support the KAMU operation. In particular,

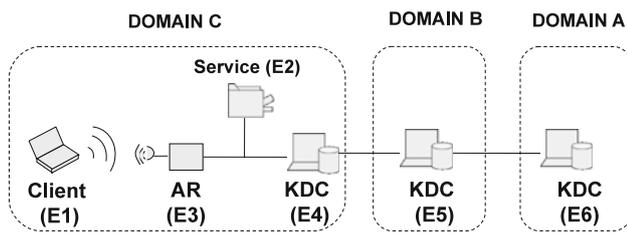


Fig. 5 Deployed testbed

routines for message construction and processing have been extended to support the new privacy extensions.

In order to provide a proper testbed to analyze the behavior and to measure the performance of the proposed privacy framework, we build a basic and generic scenario that allows us to test both single-domain and multi-domain transactions. In this context, the experimental testbed comprises the following elements:

- (E1) A low-end laptop incorporating a VIA Nehemiah 1200 MHz processor along with 488 MB of RAM. This machine integrates an 802.11 wireless interface and is used as Kerberos client.
- (E2) One VIA Nehemiah 1200 MHz machine with 488 MB of RAM. This low-end device affords Ethernet connectivity and is used as Kerberos service.
- (E3) A Wireless-G Broadband Router model Linksys WRT-54GL incorporating a MIPS 200 MHz processor with 16 MB of RAM. This device integrates both wireless and wired interfaces and is used to offer wireless connectivity to the client.
- (E4) One Intel Pentium 4 3.2 GHz PC that incorporates 1024 MB of RAM. This machine has Ethernet connectivity and is used as a high capacity KDC server.
- (E5) One high-end PC incorporating an Intel Pentium 4 3 GHz processor with 512 MB of RAM. This device has one Ethernet interface and acts as KDC server.
- (E6) One desktop machine with an Intel Pentium 4 processor clocked at 3 GHz and 512 MB of RAM. This device, used as an high capacity KDC server, connects to Internet through an Ethernet interface.

To experiment with realistic roaming scenarios, these devices are geographically distributed in order to simulate different administrative domains. As depicted in Fig. 5, while machines E5 and E6 constitute administrative domains A and B, respectively, the remaining elements (from E1 to E4) are placed within domain A. The average roundtrip time between the domains C and A is ≈ 104 ms and ≈ 117 ms between the C and B, although these values should only be considered as an indication.

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 to the execution of the original version of the Kerberos protocol such as defined in [20] (that is, without our extensions); and the term *KAMU* to denote our privacy-enhanced Kerberos implementation with privacy support.

As discussed in Sect. 6, none existing solution is able to achieve the level of privacy offered by KAMU. For this reason, we do not compare the performance of KAMU against alternative methods that could be used for providing some sort of privacy in Kerberos. Nevertheless, for the sake of completeness, we do compare against PrivaKERB [15] since it is the only alternative solution offering some level of unlinkability when operating in the so-called *Level 2*.

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-domain scenario) or by a visited KDC (multi-domain 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 Kerberos and privacy-enhanced schemes namely PrivaKERB and KAMU. 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, represented as a 95 % confidence interval, represents the time devoted to transmitting messages over the network.
- *Message processing time*. This metric, represented as a 95 % confidence interval, measures the time devoted by a certain entity 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 metric collects a 95 % confidence interval that contains the total time required by the client to complete a specific Kerberos exchange. It comprises both network and message processing times.

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

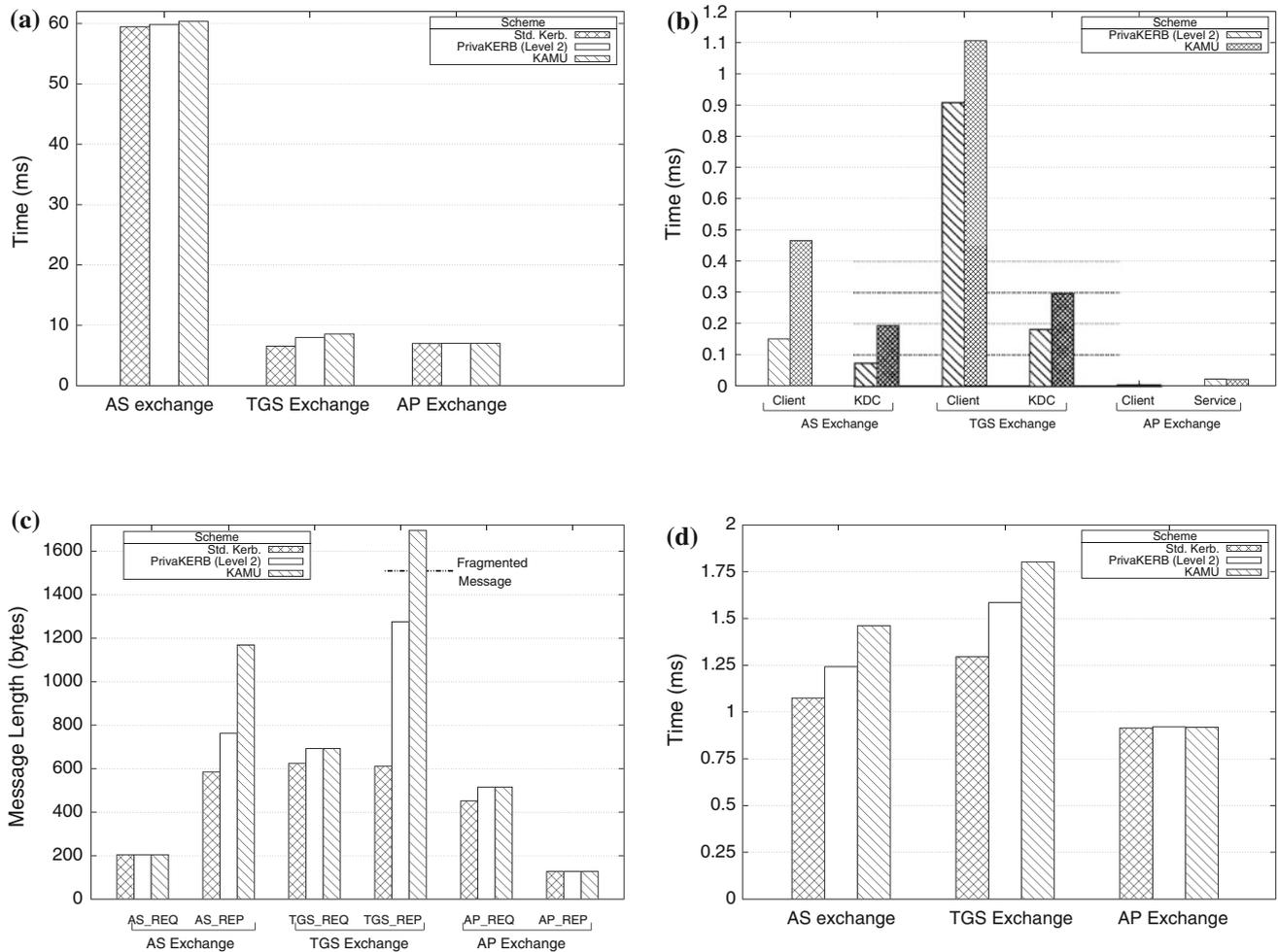


Fig. 6 Performance results in single-domain scenario. **a** Mean exchange time (ms), **b** mean privacy processing time (ms), **c** message length (bytes), **d** mean network time (ms)

5.2.1 Single-domain operation

The first scenario is intended to evaluate the privacy penalty when a client, without any valid TGT, attempts to access a service controlled by the home KDC. For this, in the deployed network architecture (see Fig. 5), we assume that the client (E1) is registered with the KDC (E4) located in domain C and requests access to a service (E2) controlled by this KDC. As observed, we simulate a client employing a wireless connection through a wireless access router (E2).

Figure 6 provides the different measurements obtained for standard Kerberos and the privacy-enhanced schemes (PrivaKERB and KAMU). To facilitate the analysis of the proposed solution and easily extract conclusions, results are displayed through graphs. As observed, only mean values are graphically represented for those metrics measuring times. Nevertheless, the reader may refer to “Appendix” where we explicate the specific values (indicating confidence intervals where appropriate) measured for the different schemes.

Figure 6a shows the mean time required by the client to complete each Kerberos exchange using standard Kerberos, PrivaKERB and KAMU privacy extensions. Regarding the standard Kerberos, we notice that the AS exchange requires far more time (about 9 times) than the other exchanges. After analyzing the different steps, we concluded that this time is required by the client to derive the *reply key* from the user’s password. In fact, this extra time is not perceived in other exchanges since the shared secret keys between client-KDC and client-service are directly recovered from the client’s credential cache.

Regarding the overhead introduced, it can be argued that in general, both PrivaKERB and KAMU produce a small increment in the AS and TGS exchanges, while the remaining AP exchanges are below similar values. As observed, in the AS and TGS exchanges, the PrivaKERB scheme increases the exchange time by ≈ 0.35 ms and ≈ 1.4 ms, respectively. These values are higher for KAMU since additional time is devoted to the enhanced ticket distribution. In particular, the

superior privacy protection achieved in KAMU introduces an insignificant latency of ≈ 0.89 ms (AS exchange) and ≈ 2 ms (TGS exchange).

The impact over each entity in terms of computing time is depicted in Fig. 6b. More specifically, we solely 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 3) in the standard Kerberos case. The results reveal that both PrivaKERB and KAMU introduce a rather tiny latency in every exchange. For example, in the TGS exchange based on PrivaKERB, the client and the KDC need ≈ 0.9 ms and ≈ 0.2 ms, respectively. Compared with PrivaKERB, these values are slightly higher when using KAMU due to the procedures related to the operation associated with the enhanced ticket distribution. For example, in the worst case (TGS exchange), this increment is ≈ 1.1 ms for the client and ≈ 0.3 ms for the KDC. Nevertheless, compared to the message processing time for standard Kerberos (see Table 3), these extra times are insignificant.

Finally, since our privacy-enhanced solution requires additional information to be exchanged between entities, another aspect of interest is the message size and network times that may affect the bandwidth consumption. While Fig. 6c provides the size of those messages involved in the different Kerberos exchanges, Fig. 6d specifies the mean network time devoted to the transmission and propagation of messages over the network. As expected, we observe that KAMU introduces an appreciable increment of those messages where the KDC delivers a ticket to the client using the fake ticket feature. While the KRB_AS_REP is up to 1,168 bytes, the KRB_TGS_REP is sent as two fragmented UDP packets of 1,514 and 181 bytes, respectively. This is because the reply message contains a fake ST, the real ST and the self-renewed TGT. Nevertheless, these increments do not provoke an appreciable penalization in the time required to transmit the messages over the network. In fact, compared with standard Kerberos, the network time required to complete the AS and TGS exchanges under KAMU only increases by ≈ 0.39 ms and ≈ 0.5 ms, respectively. Thus, considering the privacy protection achieved, we can conclude that KAMU is not a demanding solution and its privacy extensions do not affect the lightweight nature of the Kerberos protocol.

5.2.2 Multi-domain operation

In this second scenario, our intention is to analyze the behavior of our privacy-enhanced solution during a cross-realm scenario. That is, when the client requests access to a service controlled by a foreign KDC. As in previous Sect. 5.2.1, we suppose a user that initializes its device and access to a kerberized service for the first time. For this, in the deployed network architecture (see Fig. 5), we assume that the client

(E1) is registered with the KDC (E6) located in domain A and requests access to a service (E2) controlled by KDC (E4) of domain C. Thus, the client will follow the authentication path from domain A to domain C through an intermediary one (domain B). It is important to mention that all tests performed over this scenario, even in the privacy-enhanced configurations, the intermediary KDC deploys the original MIT Kerberos implementation without the need to deploy any of our privacy extensions (see Sect. 3.3.2).

Similar to the single-domain analysis developed in Sect. 5.2.1, results obtained for the different schemes are shown through graphs (see Fig. 7). Nevertheless, the reader can examine “Appendix” which provides the detailed results (indicating confidence intervals where appropriate) obtained for the tests conducted in the multi-domain scenario.

Regarding the mean exchange times depicted in Fig. 7a, it is obvious that, except “TGS exchange 2” performed with the intermediary KDC, all KAMU times include an additional latency compared with standard Kerberos. Although these penalizations fluctuate around values between 1 and 2 ms, the impact in the overall time is insignificant, especially for those messages exchanged with the home KDC. For example, the additional time to perform “TGS exchange 1” after enabling KAMU extensions is ≈ 2.9 ms, which increases the total exchange time by only $\approx 2.5\%$.

Focusing on the processing time results (see Table 6, Fig. 7b), we observe that the intermediary KDC does not perform any privacy task (columns of “TGS exchange 2” are set to zero) neither in PrivaKERB nor in KAMU since both privacy solutions do not require intermediary KDCs to be privacy-aware. Again, we realize that KAMU requires longer times to cope with all privacy extensions. Nevertheless, as in the single-domain scenario, all values are negligible. For example, in the worst case, the privacy process time for the client and KDC is ≈ 1.2 ms and ≈ 0.46 ms, respectively.

Regarding the message sizes (Fig. 7c) and network times (Fig. 7d), the same conclusions extracted for the single-domain scenario can be drawn here. While the KRB_AS_REP message reaches 1,168 bytes, the KRB_TGS_REP messages generated by home and visited KDC exceed the size of 1,500 bytes, being transmitted as two fragmented UDP packets. Nevertheless, this situation has a minimum impact over the network time. For example, in the worst case, “TGS exchange 1” requires an additional time of ≈ 1.54 ms compared with that of standard Kerberos.

6 Related work

The literature is rife with works studying the provision of privacy in network communications. Some surveys can be found in [6, 8, 34]. These contributions prove that privacy can

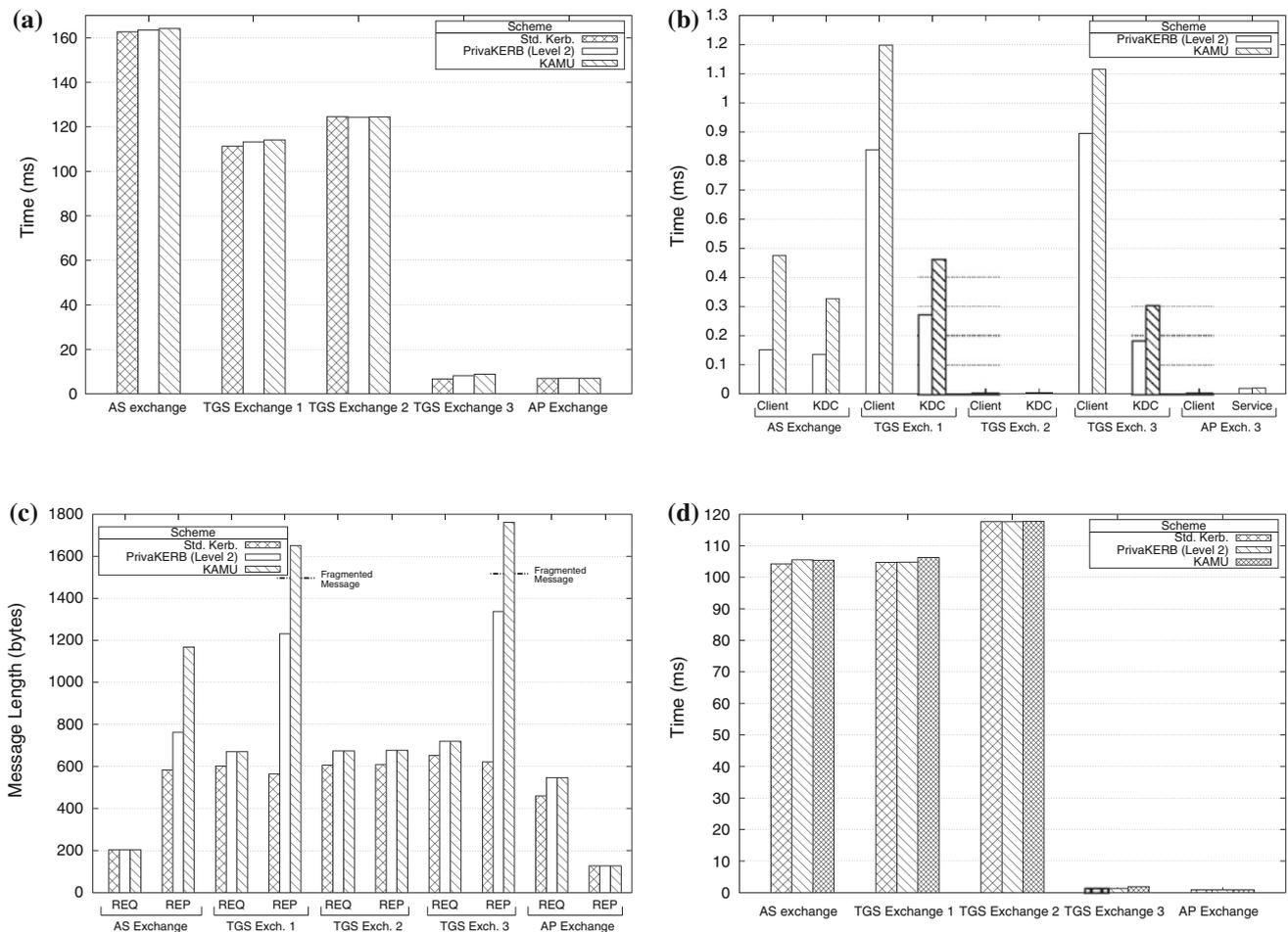


Fig. 7 Performance results in multi-domain scenario. **a** Mean exchange time (ms), **b** mean privacy processing time (ms), **c** message length (bytes), **d** mean network time (ms)

be compromised by analyzing the information provided at the different network layers (link layer, network layer and application layer). Most of the works are centered on preventing user information to be gathered from the network layer. In this regard, as already pointed out, some relevant solutions are the Chaum's mix networks [30], Tor [19], Crowds [31], Tarzan [32] or the solution proposed by Rebollo-Monedero et al. in [33]. A deeper analysis of these kind of solutions for anonymous communications can be found in [6, 16]. As for the application level, most of the works have focused on providing privacy in the Web. However, little effort has been devoted by the research community to the provision of user privacy in the Kerberos protocol. In the following, we examine the most relevant proposals, which are summarized in Table 2.

Considering that Kerberos is an application-level protocol, one approach could be based on the use of a transport layer mechanism providing confidentiality to Kerberos messages in their entirety [49]. For example, by using *Transport Layer Security* (TLS) as transport for Kerberos communi-

tion, sensitive information like the client's identifier cannot be observed by eavesdroppers. Nevertheless, this solution is not adequate since it requires the establishment of a TLS session between every pair of communicating entities (e.g., client and KDC). This requirement is particularly problematic in multi-domain scenarios since clients are required to establish a TLS tunnel with every intermediary KDC in the path from the home to the visited realm in a hop-by-hop fashion. As mentioned in [49], to avoid a wide range of active attacks (e.g. man-in-the-middle), it is recommended the use of a certificate-based authentication, thus requiring the existence of a multi-domain PKI infrastructure [51] that increases by far the deployment cost of the solution. Also, one has to affirm that every network hop does afford (enables) a TLS tunnel. Of course, the same disadvantages apply to lower level tunnels established through other mechanisms as in the case of *IP security* (IPSec) protocol [52] (IPSec).

One of the most relevant contributions [23, 24] tries to achieve user anonymity by enhancing the Kerberos protocol itself. This is done by introducing the *anonymous ticket*

concept. An anonymous ticket is a regular Kerberos ticket which, instead of being assigned to a specific user registered in a realm, it is associated with the anonymous user (*anon@anon*). Thus, when a client uses an anonymous ticket to access a service, the client remains anonymous since its real identity is not revealed either to the service or to eavesdroppers. However, this solution presents several disadvantages. On the one hand, to successfully achieve user anonymity, the solution requires the use of anonymous *Public Key Cryptography for Initial Authentication in Kerberos* (PKINIT) [53] for the anonymous TGT acquisition. By using a secure channel established with only certificate-based KDC authentication, the KDC securely delivers to the client the session key associated with the anonymous ticket. Nevertheless, the existence of a *Public Key Infrastructure* (PKI) may not always be available, specially in multi-domain scenarios. On the other hand, according to this solution, the user is anonymous even for trusted parties such as KDCs and services. This is particularly problematic in multi-domain scenarios where the organization controlling the visited domain typically needs to charge the visiting user for the accessed services.

Another solution to provide client privacy is found in the *Generalized Framework for Kerberos Pre-authentication* [50]. Among other objectives, this contribution proposes an architecture to enhance the security of Kerberos by protecting information that Kerberos transmits in cleartext. In particular, this framework protects the confidentiality of the client's identity in the messages sent from the client to the KDC (KRB_AS_REQ and KRB_TGS_REQ). By integrating this security extensions with the anonymous ticket concept, clients can acquire anonymous tickets without requiring the use of anonymous PKINIT. Nevertheless, the use of PKINIT is still required by the security framework to obtain a special ticket called *armor TGT*. This armor TGT contains the key that constitutes a shared secret between the client and the KDC and used to build the protected tunnel necessary to enhance the security in Kerberos exchanges. As a consequence, due to the need of anonymous PKINIT operation, the same deficiencies previously identified for the anonymous ticket-based proposals are also applicable to this solution.

The proposals previously analyzed only focus on the preservation of user anonymity and do not pay attention to the unlinkability property. As explained in previous Sect. 3.1, the intrinsic Kerberos operation allows eavesdroppers to infer valuable information when profiling Kerberos communications (e.g., preferred services) even when the user remains anonymous to the attacker. For this reason, in [15], the authors propose a privacy framework for Kerberos, namely *PrivaKERB*, which provides the following features: (1) User anonymity compatible with the user identification required by important processes such as accounting and charging; (2)

service access unlinkability deters eavesdroppers to trace the different services accessed by a specific anonymous user; (3) the use of other infrastructures (e.g., PKI) different than the Kerberos ones are not required. *PrivaKERB* achieves this privacy protection by introducing innovative mechanisms. First, user anonymity is ensured by providing to the client pseudonyms valid for a specific period of time. More specifically, the pseudonym generation is controlled by the KDC which distributes to the client a fresh pseudonym every time a new home TGT is issued. Second, service access unlinkability is achieved through extended anonymous tickets which contain the client's pseudonym in such a manner that is only accessible by trusted parties such as KDCs or services. Additionally, unlinkability is attained thanks to a new kind of one-use TGT (called *self-renewed TGT*) which breaks the TGT-based linkability that occurs when a client reuses the same TGT several times to request access to multiple services.

Despite *PrivaKERB* represents a significant improvement compared with other privacy solutions for Kerberos, it still presents some privacy weaknesses. More precisely, this solution is capable of achieving a basic level of unlinkability that prevents eavesdroppers from linking the different service accesses performed by a specific anonymous user. That is, by examining the content of Kerberos messages, attackers have not enough information to determine whether different service accesses are performed by the same or different Kerberos clients. Nevertheless, *PrivaKERB* fails in preventing eavesdroppers from linking the sequence of messages exchanged by a specific anonymous user to access a service: TGT acquisition from the KDC (AS exchange), ST acquisition from the KDC (TGS exchange) and ST presentation to the service (AP exchange). This linking vulnerability is particularly problematic in multi-domain scenarios where an eavesdropper can easily determine the service visited by a client in a remote realm. In the long term, this can be exploited to infer valuable information such as the origin realm of clients visiting a specific Kerberos realm or deduce the most attractive services for visiting users in a realm.

7 Conclusions and future work

In the last decade, the protection of users' privacy has become an essential issue in both wired and wireless communication networks. Particularly, privacy has special relevance in NGNs as the transmission of information over the air is prone to eavesdropping. In this context, Internet protocols constitute a serious risk for the user privacy protection since they may transport sensitive data that must not be disclosed to unauthorized third parties. So, one question is whether standard and well-established protocols commonly used nowadays in Internet communications are privacy ready or need

to be rectified in order to embed privacy preserving facilities for the end-user. In this context, this paper analyzes the privacy deficiencies that, despite the up to now research efforts, the standard Kerberos protocol still presents and needs to be solved.

As a response to these shortages, we develop a solution named KAMU to provide privacy protection to Kerberos. Specifically, our extensions provide user anonymity and complete message exchange unlinkability, thus achieving a full obfuscation of the protocol for eavesdroppers not offered by any existing solution. The KAMU privacy protection enables the user to remain unidentifiable but also solves the ticket-based linkability problem derived from the ticket usage and distribution procedure defined in the standard Kerberos protocol. Furthermore, KAMU supports both single- and multi-domain Kerberos operations and can be smoothly and straightforwardly integrated into existing implementations due to KAMU is based on existing Kerberos's extensibility mechanisms. Finally, the ability of KAMU to operate with an anonymous communication system has been analyzed to demonstrate the potential of our solution to achieve a cross-layer privacy preserving communication system able to protect against privacy attacks derived from both application and other TCP/IP levels.

The proposed solution has been implemented, and several tests have been performed in order to evaluate the overhead introduced with the new extensions. Two distinct scenarios (single-domain and multi-domain) have been extensively evaluated by considering different parameters: exchange time, privacy processing time, message length and network time. The results obtained show that the developed solution does not introduce significant overheads in the mentioned parameters when compared to the standard version of the Kerberos protocol. Even in the worst case, penalizations fluctuate around values between 1 and 2 ms, which have an almost inappreciable impact in the overall time. In fact, this reduced overhead is fundamental if the solution is to be adopted in NGNs where the access time and service continuity are essential.

From this work, we can derive a number of future research directions. In the evaluation presented, although the protocol does not introduce a significant overhead, we have demonstrated that when fake tickets are used, fragmentation may occur. This is due to the amount of data to be sent. Compelled

by this fact, a future research work involves the improvement of the protocol so as to avoid such fragmentation. Another aspect to be taken into account is the choice of the privacy level to be applied in a scenario. At the moment, the privacy level is established by KDC. However, users could support different levels of privacy and, depending on the scenario, they could be interested in self-deciding the level to be applied. Thus, the user could estimate the importance of the data being sent and how this affects to their privacy and the performance of the transaction.

Finally, we can also mention two aspects worth exploring in the future. First, KAMU assumes a user owing an initial pseudonym that must be established previously with the KDC. To overcome this requirement, we will study ways to perform the registration of a pseudonym at the same time we start a transaction with the protocol. Second, the KAMU basic operation consists in providing the user with pseudonyms that are used during a certain period of time even across different visited realms. In this sense, an important effort is envisaged to explore alternatives of blocking colluding visiting realms from sharing information on a user employing the same pseudonym.

Acknowledgments This work has been partially supported by the Ministerio de Ciencia e Innovación, Spain, under Grant TIN2011-27543-C03 by the European Seventh Framework Program through the INTER-TRUST project (contract 317731) and by the “Seneca Foundation for Excellent Group in the Region 04552/GERM/06”. Also, we would like to thank the anonymous reviewers for their valuable comments and suggestions, which have significantly contributed to improve the quality of this paper.

8 Appendix: Performance analysis detailed results

To simplify the performance analysis conducted in Sect. 5.2, numerical results are displayed through different plots. Nevertheless, for the sake of completeness, in the following, we provide the detailed measurements taken for the different metrics used as reference: message length, network time, message processing time and exchange time. Tables 3, 4 and 5 contain results obtained in the single-domain scenario for standard Kerberos, PrivaKERB (level 2) and KAMU, respectively. Similarly, Tables 6, 7 and 8 show values collected in the multi-domain scenario for these schemes.

Table 3 Results for standard Kerberos in single-domain scenario

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

Table 4 Results for PrivaKERB (level 2) in single-domain scenario

	Exchange time (ms)	Privacy processing time (ms)		Message length (bytes)		Network time (ms)
AS exchange	59.832 ± 0.046	<i>Client</i>	0.150 ± 0.002	<i>KR_AS_REQ</i>	204	1.242 ± 0.014
		<i>KDC</i>	0.075 ± 0.001	<i>KR_AS_REP</i>	762	
TGS exchange	7.698 ± 0.039	<i>Client</i>	0.907 ± 0.001	<i>KR_TGS_REQ</i>	692	1.585 ± 0.044
		<i>KDC</i>	0.183 ± 0.001	<i>KR_TGS_REP</i>	1,275	
AP exchange	7.010 ± 0.045	<i>Client</i>	0	<i>KR_AP_REQ</i>	515	0.921 ± 0.028
		<i>Service</i>	0.021 ± 0.001	<i>KR_AP_REP</i>	127	

Table 5 Results for KAMU in single-domain scenario

	Exchange time (ms)	Privacy processing time (ms)		Message length (bytes)		Network time (ms)
AS exchange	60.362 ± 0.041	<i>Client</i>	0.465 ± 0.002	<i>KR_AS_REQ</i>	204	1.461 ± 0.017
		<i>KDC</i>	0.193 ± 0.001	<i>KR_AS_REP</i>	1168	
TGS exchange	8.563 ± 0.023	<i>Client</i>	1.106 ± 0.001	<i>KR_TGS_REQ</i>	692	1.802 ± 0.039
		<i>KDC</i>	0.298 ± 0.001	<i>KR_TGS_REP</i>	1514 + 181*	
AP exchange	7.008 ± 0.039	<i>Client</i>	0	<i>KR_AP_REQ</i>	515	0.918 ± 0.019
		<i>Service</i>	0.020 ± 0.001	<i>KR_AP_REP</i>	127	

* Message sent as two fragmented UDP messages

Table 6 Results for standard Kerberos in multi-domain scenario

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

Table 7 Results for PrivaKERB (level 2) in multi-domain scenario

	Exchange time (ms)	Privacy processing time (ms)	Message length (bytes)		Network time (ms)
AS exchange	163.470 ± 0.115	<i>Client</i>	0.152 ± 0.001	<i>KR_AS_REQ</i>	204
		<i>KDC</i>	0.136 ± 0.003	<i>KR_AS_REP</i>	762
TGS exchange 1	113.228 ± 0.073	<i>Client</i>	0.839 ± 0.001	<i>KR_TGS_REQ</i>	670
		<i>KDC</i>	0.272 ± 0.004	<i>KR_TGS_REP</i>	1231
TGS exchange 2	124.301 ± 0.123	<i>Client</i>	0	<i>KR_TGS_REQ</i>	674
		<i>KDC</i>	0	<i>KR_TGS_REP</i>	677
TGS exchange 3	8.201 ± 0.063	<i>Client</i>	0.895 ± 0.002	<i>KR_TGS_REQ</i>	720
		<i>KDC</i>	0.182 ± 0.001	<i>KR_TGS_REP</i>	1337
AP exchange	6.986 ± 0.024	<i>Client</i>	0	<i>KR_AP_REQ</i>	547
		<i>Service</i>	0.019 ± 0.001	<i>KR_AP_REP</i>	127

Table 8 Results for KAMU in Multi-Domain Scenario

	Exchange time (ms)	Privacy processing time (ms)	Message length (bytes)		Network time (ms)
AS exchange	164.132 ± 0.084	<i>Client</i>	0.475 ± 0.002	<i>KR_AS_REQ</i>	204
		<i>KDC</i>	0.327 ± 0.004	<i>KR_AS_REP</i>	1168
TGS exchange 1	114.120 ± 0.084	<i>Client</i>	1.198 ± 0.002	<i>KR_TGS_REQ</i>	670
		<i>KDC</i>	0.460 ± 0.004	<i>KR_TGS_REP</i>	1506 + 145 *
TGS exchange 2	124.408 ± 0.142	<i>Client</i>	0	<i>KR_TGS_REQ</i>	674
		<i>KDC</i>	0	<i>KR_TGS_REP</i>	677
TGS exchange 3	8.791 ± 0.078	<i>Client</i>	1.115 ± 0.002	<i>KR_TGS_REQ</i>	720
		<i>KDC</i>	0.301 ± 0.001	<i>KR_TGS_REP</i>	1514 + 247 *
AP exchange	6.983 ± 0.021	<i>Client</i>	0	<i>KR_AP_REQ</i>	547
		<i>Service</i>	0.020 ± 0.001	<i>KR_AP_REP</i>	127

* Message sent as two fragmented UDP messages

References

- Chen, H., Xiao, Y., Hong, X., Hu, F., Xie, J.: A survey of anonymity in wireless communication systems. *Secur. Commun. Netw.* **2**(5), 427–444 (2008)
- Bowen, C.L., Martin, T.L.: A survey of location privacy and an approach for solitary users. In: *Proceedings of the 40th Annual Hawaii International Conference on System Sciences*, p. 163, Washington, DC, USA (2007)
- Bagnulo, M., Garcia-Martines, A., Azcorra, A.: An architecture for network layer privacy. In: *ICCC 2007: International Conference on Communications*, pp. 1509–1514, Washington, DC, USA (2007)
- 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*, pp. 1095–2055. IEEE Computer Society, Washington, DC (2010)
- Cardoso, R.S., 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*, p. 26. IEEE Computer Society, Washington, DC (2007)
- Yener, B., Edman, M.: On anonymity in an electronic society: a survey of anonymous communication systems. *ACM Comput. Surv.* **42**(1), 1–35 (2009)
- Karopoulos, G., Kambourakis, G., Gritzalis, S., Konstantinou, E.: A framework for identity privacy in SIP. *J. Netw. Comput. Appl.* **33**(1), 16–28 (2010)
- Ruiz-Martínez, A.: A survey on solutions and main free tools for privacy enhancing web communications. *J. Netw. Comput. Appl.* **35**(5), 1473–1492 (2012)
- Sweeney, L.: Uniqueness of simple demographics in the U.S. population. *Laboratory for International Data Privacy working paper* (2000)
- Golle, P.: Revisiting the uniqueness of simple demographics in the US population. In: *Proceedings of 5th ACM Workshop on Privacy in Electronic Society*, Alexandria, VA, USA, October 2006
- Ohm, P.: Broken promises of privacy: responding to the surprising failure of anonymization. Available at SSRN: <http://ssrn>.

- [com/abstract=1450006](#). University of Colorado Law Legal Studies research paper no. 09-12, August 2009
12. Tene, O.: Privacy: the new generations. *Oxford Journal, International Data Privacy Law*, pp. 1–13, November 2010
 13. Hansen, M., Tschofenig, H., Smith, R.: Privacy terminology. IETF Internet Draft, draft-hansen-privacy-terminology-03, October 2011
 14. King, N.J., Jessen, P.W.: Profiling the mobile customer—privacy concerns when behavioural advertisers target mobile phones. *Comput. Law Secur. Rev.* **26**(5), 455–478 (2010)
 15. Pereniguez, F., Marin-Lopez, R., Kambourakis, G., Gritzalis, S., Gomez, A.F.: PrivaKERB: a user privacy framework for Kerberos. *Comput. Secur.* **30**(6–7), 446–463 (2011)
 16. Ren, J., Wu, J.: Survey on anonymous communications in computer networks. *Comput. Commun.* **33**, 420–431 (2010)
 17. Mccoy, D., Bauer, K., Grunwald, D., Kohno, T., Sicker, D.: Shining light in dark places: understanding the Tor network. In: *Proceedings of the 8th International Symposium on Privacy Enhancing Technologies, PETS '08*, pp. 63–76. Springer, Berlin (2008)
 18. Chaabane, A., Manils, P., Ali Kaafar, M.: Digging into anonymous traffic: a deep analysis of the Tor anonymizing network. In: *Proceedings of the 2010 Fourth International Conference on Network and System Security, NSS '10*, pp. 167–174. IEEE Computer Society, Washington, DC (2010)
 19. Dingleline, R., Mathewson, N., Syverson, P.: Tor: the second-generation onion router. In: *Proceedings of the 13th Conference on USENIX Security Symposium*, vol. 13, pp. 21–21. USENIX Association (2004)
 20. Neuman, C., Yu, T., Hartman, S., Raeburn, K.: The Kerberos network authentication service (V5). IETF RFC 4120, July 2005
 21. Kerberos WG. <http://datatracker.ietf.org/wg/krb-wg/>
 22. The MIT Kerberos Consortium. <http://www.kerberos.org>
 23. Medvinsky, A., Cargille, J., Hur, M.: Anonymous credentials in Kerberos. IETF Internet Draft, IETF draft-ietf-cat-kerberos-anoncred-00.txt, March 1998
 24. Zhu, L., Leach, P., Hartman, S.: Anonymity support for Kerberos. IETF Internet Draft, IETF draft-ietf-krb-wg-anon-12.txt, August 2010
 25. Gulyás, G., Schulcz, R., Imre, S.: Comprehensive analysis of web privacy and anonymous web browsers: Are next generation services based on collaborative filtering? In: *Proceedings of the Joint SPACE and TIME Workshops*, pp. 17–32 (2008)
 26. Zalewski, M.: *Silence on the wire: a field guide to passive reconnaissance and indirect attacks*, 1st edn. No Starch Press, San Francisco, CA (2005)
 27. Back, A., Möller, U., Stiglic, A.: Traffic analysis attacks and trade-offs in anonymity providing systems. In: *Proceedings of the 4th International Workshop on Information Hiding*, pp. 245–257. Springer (2001)
 28. Hopper, N., Vasserman, E.Y., Chan-Tin, E.: How much anonymity does network latency leak? In: *Proceedings of the 14th ACM Conference on Computer and Communications Security*, pp. 82–91. ACM (2007)
 29. Schlegel, R., Wong, D.S.: Low latency high bandwidth anonymous overlay network with anonymous routing. Published: Cryptology ePrint Archive. Report 2009/294 (2009). <http://eprint.iacr.org/>
 30. Chaum, D.L.: Untraceable electronic mail, return addresses, and digital pseudonyms. *Commun. ACM* **24**, 84–90 (1981)
 31. Reiter, M.K., Rubin, A.D.: Crowds: anonymity for web transactions. *ACM Trans. Inf. Syst. Secur.* **1**(1), 66–92 (1998)
 32. Freedman, M.J., Morris, R.: Tarzan: a peer-to-peer anonymizing network layer. In: *Proceedings of the 9th ACM Conference on Computer and Communications Security*, pp. 193–206. ACM (2002)
 33. Rebollo-Monedero, D., Forné, J., Solanas, A., Martínez-Ballesté, A.: Private location-based information retrieval through user collaboration. *Comput. Commun.* **33**(6), 762–774 (2010)
 34. Danezis, G., Diaz, C., Syverson, P.: *Systems for Anonymous Communication*. CRC Cryptography and Network Security Series, pp. 341–389. Chapman & Hall/CRC, London (2009)
 35. Li, B., Erdin, E., Güneş, M.H., Bebis, G., Shipley, T.: An analysis of anonymity technology usage. In: *Proceedings of the Third International Conference on Traffic Monitoring and Analysis, TMA'11*, pp. 108–121. Springer, Berlin (2011)
 36. Syverson, P., Tsudik, G., Reed, M., Landwehr, C.: Towards an analysis of onion routing security. In: *International Workshop on Designing Privacy Enhancing Technologies: Design Issues in Anonymity and Unobservability*, pp. 96–114. Springer, New York, NY (2001)
 37. 3proxy tiny free proxy server. <http://www.3proxy.ru/>
 38. TorifyHOWTO. <https://trac.torproject.org/projects/tor/wiki/doc/TorifyHOWTO>
 39. Whonix. <http://sourceforge.net/p/whonix/>
 40. Berthold, O., Federrath, H., Köhntopp, M.: Project anonymity and unobservability in the Internet. In: *Proceedings of the Tenth Conference on Computers, Freedom and Privacy: Challenging the Assumptions, CFP '00*, pp. 57–65. ACM, New York, NY (2000)
 41. Danezis, G.: The traffic analysis of continuous-time mixes. In: *PET'04*, pp. 35–50. Springer (2005)
 42. Shmatikov, V., Wang, M.-H.: Timing analysis in low-latency mix networks: attacks and defenses. In: *ESORICS'06*. Springer (2006)
 43. Wiangsripanawan, R., Susilo, W., Safavi-Naini, R.: Design Principles for Low Latency Anonymous Network Systems Secure against Timing Attacks. In: *Proceedings of the fifth Australasian Symposium on ACSW Frontiers*, **68**, 183–191 (2007)
 44. Johnson, A., Feigenbaum, J., Syverson, P.: Preventing active timing attacks in low-latency anonymous. *Communication*, July 2010
 45. Rennhard, M., Plattner, B.: Practical anonymity for the masses with MorphMix, vol. 3110 of *Lecture Notes in Computer Science*, pp. 233–250. Springer, February 2004
 46. Wendolsky, R., Herrmann, D., Federrath, H.: Performance Comparison of Low-Latency Anonymisation Services from a User Perspective, pp. 233–253. Springer, Berlin (2007)
 47. Fabian, B., Goertz, F., Kunz, S., Müller, S., Nitzsche, M.: *Privately Waiting—A Usability Analysis of the Tor Anonymity Network*, vol. 58, pp. 63–75. Springer, Berlin (2010)
 48. MIT Kerberos Distribution. <http://web.mit.edu/Kerberos/>
 49. Josefsson, S.: Using Kerberos V5 over the transport layer security (TLS) protocol. IETF RFC 6251, May 2011
 50. Hartman, S., Zhu, L.: A generalized framework for Kerberos pre-authentication. IETF Internet Draft, draft-ietf-krb-wg-preauth-framework-17, June 2010
 51. Shimaoka, M., Hastings, N., Nielsen, R.: Memorandum for multi-domain public key infrastructure interoperability. IETF RFC 5217, July 2008
 52. Kent, S., Seo, K.: Security architecture for the Internet protocol. IETF RFC 4301, December 2005
 53. Zhu, L., Tung, B.: Public key cryptography for initial authentication in Kerberos (PKINIT). IETF RFC 4556, June 2006