Container Authenticated by Multiple MACs
draft-ietf-krb-wg-cammac-01

Abstract

Abstract: This document proposes a Kerberos Authorization Data container similar to AD-KDC-ISSUED, but that allows for multiple MACs or signatures on the contained Authorization Data elements.

Status of this Memo

This Internet-Draft is submitted in full conformance with the provisions of BCP 78 and BCP 79.

Internet-Drafts are working documents of the Internet Engineering Task Force (IETF). Note that other groups may also distribute working documents as Internet-Drafts. The list of current Internet-Drafts is at http://datatracker.ietf.org/drafts/current/.

Internet-Drafts are draft documents valid for a maximum of six months and may be updated, replaced, or obsoleted by other documents at any time. It is inappropriate to use Internet-Drafts as reference material or to cite them other than as "work in progress."

This Internet-Draft will expire on August 12, 2012.

Copyright Notice

Copyright (c) 2012 IETF Trust and the persons identified as the document authors. All rights reserved.

This document is subject to BCP 78 and the IETF Trust’s Legal Provisions Relating to IETF Documents (http://trustee.ietf.org/license-info) in effect on the date of publication of this document. Please review these documents carefully, as they describe your rights and restrictions with respect to this document. Code Components extracted from this document must include Simplified BSD License text as described in Section 4.e of the Trust Legal Provisions and are provided without warranty as described in the Simplified BSD License.
Table of Contents

1. Introduction ............................................. 3
2. Requirements Language .................................... 3
3. Validation ................................................... 3
4. Encoding .................................................... 4
   4.1. AD-CAMMAC ........................................... 4
5. Assigned numbers .......................................... 6
6. IANA Considerations ....................................... 6
7. Security Considerations .................................... 6
8. Acknowledgements .......................................... 6
9. References .................................................. 6
   9.1. Normative References .................................. 6
   9.2. Informative References ................................ 6
Appendix A. Additional Stuff ................................. 7
Authors’ Addresses ............................................. 7
1. Introduction

This draft proposes a Authorization Data container for Kerberos that identifies a base set of MAC and other elements necessary to authenticate the authorization data being carried in such a way that not only the KDC but also services can independently verify that the data has been authenticated by the KDC and has not been tampered with.

2. Requirements Language

The key words "MUST", "MUST NOT", "REQUIRED", "SHALL", "SHALL NOT", "SHOULD", "SHOULD NOT", "RECOMMENDED", "MAY", and "OPTIONAL" in this document are to be interpreted as described in RFC 2119 [RFC2119].

3. Validation

Authorization data is highly sensitive and must be validated to insure no tampering has occurred.

In order to validate any information the receiving client need to be able to cryptographically verify the data. This is done by introducing a new AuthorizationData element called AD-CAMMAC that contains enough information to bind the contents to a principal in a way that a receiving client can verify autonomously without further contact with the KDC.

The following information is needed:

- The KDC signature.
- The Service Signature.
- Optional Trusted Service Key Signature.
- Optional PUBKEY KDC Signature.

The KDC signature is required to allow the KDC to validate the data without requiring to recompute the contents at every TGS request.

The SVC signature is required so that the Service can verify that the authorization data has been validated by the KDC.

Both the Trusted Service Checksum and the asymmetric KDC Signature are useful to verify the authenticity of the contents on the same host, when the data is received by a less trusted service and passed
to a more trusted service on the same host without the need for additional roundtrips to the KDC.

The ad-type for AD-CAMMAC is (TBD).

4. Encoding

The Kerberos protocol is defined in [RFC4120] using Abstract Syntax Notation One (ASN.1) [X680]. As such, this specification also uses the ASN.1 syntax for specifying both the abstract layout of the AD-CAMMAC attributes, as well as its encoding.

4.1. AD-CAMMAC
AD-CAMMAC ::= SEQUENCE {
  kdc-signature [0] Checksum,
  svc-signature [1] Checksum,
  trusted-svc-signature [2] OPT-Checksum OPTIONAL,
  pubkey-signature [3] TBD OPTIONAL,
}

OPT-Checksum ::= SEQUENCE {
  identifier [0] PrincipalName,
  signature [1] Checksum
}

kdc-signature
A cryptographic checksum computed over the encoding of the elements field, keyed with the krbtgt key.
Checksum type TBD.

svc-signature
A cryptographic checksum computed over the encoding of the elements field, keyed with the service long term key.
Checksum type TBD.

trusted-svc-signature
A principal name and a cryptographic checksum computed over the encoding of the elements field, keyed with the long term key of the principal name specified in the Name field. Unless otherwise explicitly administratively configured, the key SHOULD be found by substituting the service name component of the principal name of the service with ‘host’.
If the service is ‘host’ this checksum is redundant and can be omitted.
If the resulting host/<name>@REALM or the administratively configured service is not found in the KDC database this checksum can be omitted.
Checksum type TBD.

pubkey-signature
A name identifying the asymmetric key-pair used.
A checksum computed over the encoding of the elements field using the Private Key identified in the Name field.
If an asymmetric key is not available this checksum MUST be omitted.
Signature type TBD.

elements
A sequence of authorization data elements issued by the KDC.
5. Assigned numbers

TBD

6. IANA Considerations

TBD.

7. Security Considerations

Although generally authorization data are conveyed within a ticket and are thereby protected using the existing encryption methods on the ticket, some authorization data requires the additional protection provided by the CAMMAC.

8. Acknowledgements

TBD.

9. References

9.1. Normative References


9.2. Informative References


Appendix A.  Additional Stuff

This becomes an Appendix.

Authors’ Addresses

Simo Sorce (editor)
Red Hat

Email: ssorce@redhat.com

Tom Yu (editor)
MIT Kerberos Consortium

Email: tlyu@mit.edu

Thomas Hardjono (editor)
MIT Kerberos Consortium

Email: hardjono@mit.edu
POSIX Authorization Data
draft-ietf-krb-wg-pad-01

Abstract

This document proposes a Kerberos Authorization Data element containing user and group directory information similar to that provided by RFC 2307, typically used by POSIX and POSIX-like systems in the course of login type activities.

Status of this Memo

This Internet-Draft is submitted in full conformance with the provisions of BCP 78 and BCP 79.

Internet-Drafts are working documents of the Internet Engineering Task Force (IETF). Note that other groups may also distribute working documents as Internet-Drafts. The list of current Internet-Drafts is at http://datatracker.ietf.org/drafts/current/.

Internet-Drafts are draft documents valid for a maximum of six months and may be updated, replaced, or obsoleted by other documents at any time. It is inappropriate to use Internet-Drafts as reference material or to cite them other than as "work in progress."

This Internet-Draft will expire on August 12, 2012.

Copyright Notice

Copyright (c) 2012 IETF Trust and the persons identified as the document authors. All rights reserved.

This document is subject to BCP 78 and the IETF Trust’s Legal Provisions Relating to IETF Documents (http://trustee.ietf.org/license-info) in effect on the date of publication of this document. Please review these documents carefully, as they describe your rights and restrictions with respect to this document. Code Components extracted from this document must include Simplified BSD License text as described in Section 4.e of the Trust Legal Provisions and are provided without warranty as
described in the Simplified BSD License.

Table of Contents

1. Introduction ................................................. 3
2. Requirements Language ...................................... 3
3. Use-Case: Cross-Realm Directory Services ..................... 3
4. A POSIX Authorization Structure for Kerberos V5 .............. 4
   4.1. Attributes ........................................... 4
   4.2. PAD-Realm ............................................ 5
   4.3. PAD-DNS-Domain ...................................... 5
   4.4. PAD-Short-Domain .................................... 5
   4.5. PAD-UDID ............................................. 5
   4.6. PAD-Posix-Username ................................ 6
   4.7. PAD-Posix-UID ........................................ 6
   4.8. PAD-Posix-GID ........................................ 6
   4.9. PAD-Posix-Gecos ..................................... 6
   4.10. PAD-Posix-Homedir ................................... 6
   4.11. PAD-Posix-Shell .................................... 6
   4.12. PAD-Fullname ....................................... 7
   4.13. PAD-AlternateNames ................................ 7
   4.14. PAD-Groups ......................................... 7
   4.15. PAD Mapped Attributes ................................ 8
   4.16. RFC2307 references for Directory Services backed KDCs .... 8
      4.16.1. PAD-Posix-Username as 'uid' .................... 8
      4.16.2. PAD-Posix-UID as 'uidNumber' ................ 8
      4.16.3. PAD-Posix-GID as 'gidNumber' ................ 9
      4.16.4. PAD-Posix-Gecos as 'gecos' ................... 9
      4.16.5. PAD-Posix-Homedir as 'homeDirectory' .......... 9
      4.16.6. PAD-Posix-Shell as 'loginShell' ............... 9
5. Validation .................................................... 9
6. Encoding ..................................................... 9
   6.1. PAD Format ............................................ 9
7. Data Structures and Extensions ................................ 11
   7.1. AD-ID-ANCHOR ........................................ 11
   7.2. AD-PAD-DATA ......................................... 12
   7.3. GSS-API Authenticator Extension ......................... 12
8. Assigned numbers ............................................. 13
9. Timeouts Considerations ..................................... 13
10. IANA Considerations ....................................... 13
11. Security Considerations ................................... 14
12. Acknowledgements ......................................... 14
13. References ............................................... 14
   13.1. Normative References ............................... 14
   13.2. Informative References ............................. 15
Appendix A. Additional Stuff .................................. 15
Authors’ Addresses ........................................... 15
1. Introduction

There is an increasing need today for Kerberos to support the delivery and processing of authorization information pertaining to the principals seeking access to the servers. Kerberos today is used extensively for authentication to directory services within the Enterprise. In many cases, a directory service is implemented as a distributed database system organized across multiple realms. As such, when a client in one realm seeks access to a directory service component located within a different realm, information regarding both the identity of the client and the permissions associated with that client must be communicated across the realms. Currently there does not exist a common and standardized structure in Kerberos (V5) for conveying access control or authorization information.

This draft proposes an authorization data element for Kerberos that contains information that is useful for dynamically updating the user and group directory information in a POSIX system, usually in the course of a login type activity.

2. Requirements Language

The key words "MUST", "MUST NOT", "REQUIRED", "SHALL", "SHALL NOT", "SHOULD", "SHOULD NOT", "RECOMMENDED", "MAY", and "OPTIONAL" in this document are to be interpreted as described in RFC 2119 [RFC2119].

3. Use-Case: Cross-Realm Directory Services

In this section we discuss one of the primary use-case scenarios for the POSIX Authorization Data (PAD) structure within Kerberos V5. In this use-case a client principal is seeking to access a service in a different realm. Since the remote service does not have authorization information regarding the client, it needs to obtain it either from querying the directory service in its own realm or the directory service located in the client’s realm. It is here that a common PAD structure becomes necessary and invaluable in order to achieve a high-degree of interoperability between directory services in distinct realms.

In this use-case a client principal C1 in realm R1 is seeking access to services (or servers) located in a different realm R2. In accessing local service S1 in realm R1 the client must first be authenticated by KDC1 in that realm. A directory service (e.g. LDAP) called D1 is used in realm R1 to perform authorization of the client, after the client has been authenticated by KDC1.
When the client principal later seeks to access services or resources S2 in realm R2, following the usual Kerberos flow the client must first obtain a cross-realm TGT from KDC1 (in realm R1) and then present it to KDC2 (in realm R2) in order to obtain a service-ticket for S2. However, one immediate issue is the fact that service S2 does not have authorization information regarding the permissions or privileges of client C1 in realm R1. The service S2 could query its own directory service D2 to obtain authorization information pertaining to client C1. In the absence of such information in D2, the service S2 could then perform a cross-realm query to the directory services D1 operating in realm R1.

However, this cross-realm query from S2 to D1 is not only inefficient, but it also implies knowledge of multiple heterogeneous systems by all actors. Two different realms may rely on completely different infrastructures for user information storage, ranging from different LDAP implementations with different schema conventions to NIS, SQL databases, flat files, and so on. Every service in the realm R2 would have to know what information system is in use in R1, how to reach it, how to read and eventually how to map data from it. Moreover security related aspects on the authentication of S2 by the directory D1, the authorization of S2 to make such a query, the protection of responses from D1 to S2, and so on, would have to be addressed.

This use-case illustrates the need for a common PAD structure to address this cross-realm authorization problem. In particular, the PAD structure for the cross-realm access to remote services needs to be contained or carried within cross-realm TGTs and service-tickets. Such a PAD structure needs to carry enough authorization information such that a decision can be made by service S2 in realm R2 regarding the access request originating from the client principal C1 within realm R1.


4.1. Attributes

The following attributes are defined in this document:

- PAD-Realm
- PAD-DNS-Domain
- PAD-Short-Domain
These are each defined and discussed further below.

4.2. PAD-Realm

The full Realm Name of the Realm the authorization information applies to.

4.3. PAD-DNS-Domain

The DNS Domain name associated to the Realm.

4.4. PAD-Short-Domain

A short domain name that uniquely identifies, within the set of trusted realms, the domain the principal belongs to. The short Domain name is useful for representation purposes in the OS. A KDC MUST verify this name is unique and correctly represent a remote realm within its own realm and is allowed to change or remove this field during validation. This may be done to resolve name conflicts in large trust relationships.

4.5. PAD-UDID

A UDID is a Unique Domain Identifier. Ideally it universally identifies the domain as the one the following local identifiers belongs to. This is used to differentiate between local identifiers belonging to different domains/realms.
The UDID size can be dependent on the specific Domain type and implementation. However it SHOULD be not less than 96 bits long so that chances of conflicts are relatively low. A 96 bit long identifier allows to construct a 128bit account identifier by concatenating the UDID to the local account Identifier (32bit quantity in POSIX).

For the purpose of this document the UDID is a completely opaque number and implementations SHOULD not try to perform any enforcement on the format of this number on receiving it.

4.6. PAD-Posix-Username

This is the user name that correspond to the kerberos principal, this is the name that SHOULD be used by the OS to represent the user. The OS may decide to prefix or suffix this name with the PAD-Domain or PAD-Realm or PAD-Short-Domain names to avoid name conflicts with local accounts.

4.7. PAD-Posix-UID

This is the UID Number associated to the user. This number is local to the domain identified by PAD-UDID.

4.8. PAD-Posix-GID

This is the Primary GID Number associated to the user. This number is local to the domain identified by PAD-UDID.

4.9. PAD-Posix-Gecos

The Gecos field for the User associated to the Principal if available. Can be omitted. If not available PAD-Fullname can be used instead.

4.10. PAD-Posix-Homedir

The home directory path relative to the local system, if available. If not available local defined defaults apply.

4.11. PAD-Posix-Shell

The default shell for the user, defined as the path of the binary relative to the local filesystem, if available. If not available local defined defaults apply.
4.12. PAD-Fullname

The full name of the user if available.

4.13. PAD-AlternateNames

Alternate names can be used by application to identify a user by means that differ from the user principal. Names are in string form and utf8 encoded [UTF-8]. In order to allow applications to recognize the name type without guesswork, alternate names are prefixed with a string followed by the colon ':' character and the name, without any space or other separation character. The following Alternate names are currently recognized: EMAIL, OS, OPENID, OAUTH. It is allowed to include multiple alternate names of the same type. The order in which they are provided represent the priority within the same name type, if applications need to choose between names.

(TODO: need discussion on whether these needs labeled prefixes or explicit attributes for each alternate representation etc...)

4.14. PAD-Groups

This is a structured attribute and defines the groups the principal is member of.

The first value in the structure represents the domain UDID and is optional. If missing the domain UDID is assumed to be the one defined in the PAD-UDID attribute.

Then an array of values that define the groups as follows. Each group value includes 3 subvalues:

- (1) Name: This is the name of the group.
- (2) Type: Optional, type of group
- (3) ID: group ID.

If the type is missing it is assumed that the group is of type "Posix Group" and the following ID is required and represents the gid number. The type is represented through a simplified OID like type where only 2 levels are defined. 0.0 Is reserved for posix groups, and the 0 prefix is reserved to official RFX use. Additional Prefixes can be assigned to organizations that request it for their purposes. Assignment TBD.

Multiple PAD-Groups attributes can be present at the same time. A trusting KDC can augment the original user’s set of groups by adding
a new PAD-Groups structure that contains groups local to the trusting domain. In this case the domain UDID is required. The domain UDID is used for gid number conflict resolution when the PAC is transmitted between services of different realms.

PAD-Groups are optional attributes and the KDC, upon PAC revalidation, may decide to remove the original attributes that do not belong to the KDC security domain in order to save space or to censor information to avoid disclosing data to services.

4.15. PAD Mapped Attributes

In POSIX, users and groups ID are not universally unique, and different Realms (even different machines within an authorization realm actually) may have overlapping and conflicting IDs. If this is the case, a trusting KDC may decide to re-map IDs coming from a foreign Realm to help services with uid/gid mapping and avoid ID conflicts that can lead to serious security issues. The original IDs are generally preserved.

If multiple PAD buffers are received and one of them contains a PAD-UDID that is recognized by the application to be the local security domain identifier, then only the mapped attributes in this buffer SHOULD be used for authorization purposes.

4.16. RFC2307 references for Directory Services backed KDCs

A few attributes contain the keyword 'Posix' in their name. These attributes are usually represented by RFC2307 in Directory Services. If the primary store for these attributes is a Directory the following equivalence with RFC2307 defined attributes can be used.

4.16.1. PAD-Posix-Username as 'uid'

The PAD-Posix-Username is the User ID, and its syntax is equivalent to the attribute named 'uid' in RFC 2307. This attribute is defined in RFC 4519 (2.39). The attribute is defined as multivalued in RFC 4519 but in this context only a single value is allowed. To define aliases refer to the attribute PAD-AlternateNames.

4.16.2. PAD-Posix-UID as 'uidNumber'

The PAD-Posix-UID is the User’s Unique Identifier Number, and its syntax is equivalent to the attribute named ‘uidNumber’ in RFC 2307.
4.16.3. PAD-Posix-GID as 'gidNumber'

The PAD-Posix-GID is the User’s Primary Group Identifier Number, and its syntax is equivalent to the attribute named 'gidNumber' in RFC 2307.

4.16.4. PAD-Posix-Gecos as 'gecos'

The PAD-Posix-Gecos is the User’s Common Name, although, traditionally, this field has been used to convey additional information beyond the user’s full name. Its syntax is equivalent to the attribute named ‘gecos’ in RFC 2307.

4.16.5. PAD-Posix-Homedir as 'homeDirectory'

The PAD-Posix-Homedir is the User’s LOCAL home directory. Its syntax is equivalent to the attribute named 'homeDirectory' in RFC 2307.

4.16.6. PAD-Posix-Shell as 'loginShell'

The PAD-Posix-Shell is the User’s preferred login shell. Its syntax is equivalent to the attribute named 'loginShell' in RFC 2307.

5. Validation

The PAD information is used by a client to perform authorization, therefore this information is highly sensitive and must be validated to insure no tampering has occurred.

Therefore AD-PAD elements MUST always be transmitted contained within an AD-CAMMAC element

6. Encoding

The Kerberos protocol is defined in [RFC4120] using Abstract Syntax Notation One (ASN.1) [X680]. As such, this specification also uses the ASN.1 syntax for specifying both the abstract layout of the PAD attributes, as well as their encodings.

6.1. PAD Format

The information carried in the PAD needs to be augmented by some identifying information in order to tie the PAD data to a specific identity within the Kerberos Realm.

In order to allow additional authorization data to be tied together
and at the same time always verifiable we propose that the PAD is delivered as an AD element within a AD-CAMMAC.

An AuthorizationData element of type AD-ID-ANCHOR is used to bind the PAD to the ticket and the authorization data within the PAD to the specific principal. This element MUST always be present and SHOULD be validated. If this element is not available the PAD data MUST be discarded and considered untrustworthy.

The AD-ID-ANCHOR includes the full principal name, the realm, the expiration time and an optional session ID.

The ad-type for AD-PAD-ANCHOR is (TBD).

The AD-PAD-DATA include the attributes described in paragraph 4.

The ad-type for AD-PAD-DATA is (TBD).

The final structure used to deliver the PAD Data looks loosely like the following diagram.
7. Data Structures and Extensions

7.1. AD-ID-ANCHOR

The AD-ID-ANCHOR is intended to provide a means to bind data, carried in a AD-CAMMAC element, to a specific Identity (Principal), and optionally to a specific Ticket by using the session ID element.
AD-ID-ANCHOR ::=SEQUENCE {
p-realm [0] Realm,
p-name [1] PrincipalName,
expiration [2] KerberosTime,
session-id [3] TBD
}

p-realm, p-name
The realm and name of the principal the authorization data elements apply to.

expiration
The Expiration Date of the Authorization Data. Normally this is the same as the original TGT expiration date.

session-id
A random number that uniquely ties any following ticket this PAD Data is associated to with the original TGT Released to the user.

7.2. AD-PAD-DATA
The AD-PAD-DATA data is intended to provide a means for a Kerberos principal credentials to carry authorization data that the receiving service can use to perform authorization decisions.

AD-PAD-ANCHOR ::=SEQUENCE {
TBD
}

7.3. GSS-API Authenticator Extension
The Authenticator Checksum as defined in RFC 4121 limit the size of delegated credentials in the KRB_CRED message to a size of 64KiB.

In order to be able to transfer larger messages an extension is defined. This extension is used in stead of the Dlght/Deleg fields, and the Dlght and Deleg filed fields MUST not be included when this extensions is appended to the authenticator.
The extension SHALL have the following format which is drafted according to [draft-ietf-krb-wg-gss-cb-hash-agility]:

<table>
<thead>
<tr>
<th>Octet</th>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>0..3</td>
<td>ExtN</td>
<td>A 16bit value identifying the extension. Represented in big-endian order;</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Contains the hex value 0xXXXXXXXXX.</td>
</tr>
<tr>
<td>4..7</td>
<td>Length</td>
<td>The length of the Extended Delegation field. Represented in big-endian order;</td>
</tr>
<tr>
<td>8..N</td>
<td>Data</td>
<td>A KRB_CRED message (N = Length + 8)</td>
</tr>
</tbody>
</table>

A new flag GSS_C_EXT_DELEG_FLAG with Value X is also defined. This flag is used instead of GSS_C_DELEG_FLAG when the delegated credentials are larger than 64KiB and cannot fit in the standard Deleg field.

Implementors SHOULD use this Extensions and this flag only if the KRB_CRED message is larger than 64KiB and use the standard Deleg field otherwise.

8. Assigned numbers

TBD

9. Timeouts Considerations

Current implementations depend on very strict timeouts on obtaining AS Replies. In popular implementations the client will timeout if it doesn't receive a reply within 1 second. Adding authorization data may involve lookups to external (to the KDC) data sources. Implementors should consider whether the current timeout is still reasonable in light of the additional processing KDCs may be required to do.

10. IANA Considerations

TBD.
11. Security Considerations

Although it is anticipated that the PAD structure itself will be carried within a ticket and thereby protected using the existing encryption methods on that ticket, there are a number of issues that have bearings on the security of the entire Kerberos realm as a whole. Some of these issues are as follows:

- **UID and GID Collisions:** There is always the possibility of collision of numbers representing a UID and a GID. This problem can be remedied to a large degree by realms using an appropriate range selection policy and algorithms.

- **When collisions are detected:** The KDC or, alternatively, the receiving Service MUST be able to remap IDs so that they do not conflict with locally defined IDs.

- **Transit-domain issues:** The PAC must be signed by the KDC that is attaching it to a ticket with 2 different signatures. The service signature so that the service can verify its KDC validated the contents. The KDC signature, so that the OS can ask the KDC to confirm the PAD has not been modified by a less trusted service. An optional asymmetric key signature is also allowed if Keys are available in order to avoid additional roundtrips. For cross-realm tickets the "service" signature is made with the cross-realm key. When a KDC receives a PAD it is allowed to modify it in any way. It can filter out information or add information (like group memberships defined locally). A KDC may also decide to change information in different ways depending on what service it is targeted to.

12. Acknowledgements

TBD.

13. References

13.1. Normative References


13.2. Informative References


Appendix A. Additional Stuff

This becomes an Appendix.

Authors’ Addresses

Simo Sorce (editor)
Red Hat

Email: ssorce@redhat.com
Tom Yu (editor)
MIT Kerberos Consortium

Email: tlyu@mit.edu

Thomas Hardjono (editor)
MIT Kerberos Consortium

Email: hardjono@mit.edu
Abstract

Kerberos is widely used for authentication within organisations. It is not, however, commonly used for authentication between domains or realms ("cross-realm operation"). Abfab is a new architecture, based on the AAA framework, that provides a mechanism for federating authentication between realms.

AAA protocols are already widely used for federating authentication for network access scenarios today. It has been proposed that Abfab could be used to provide a mechanism yielding cross-realm functionality for Kerberos. This document discusses two alternative models with the aim of informing and facilitating discussion.

Status of This Memo

This Internet-Draft is submitted in full conformance with the provisions of BCP 78 and BCP 79.

Internet-Drafts are working documents of the Internet Engineering Task Force (IETF). Note that other groups may also distribute working documents as Internet-Drafts. The list of current Internet-Drafts is at http://datatracker.ietf.org/drafts/current/.

Internet-Drafts are draft documents valid for a maximum of six months and may be updated, replaced, or obsoleted by other documents at any time. It is inappropriate to use Internet-Drafts as reference material or to cite them other than as "work in progress."

This Internet-Draft will expire on September 13, 2012.

Copyright Notice

Copyright (c) 2012 IETF Trust and the persons identified as the document authors. All rights reserved.

This document is subject to BCP 78 and the IETF Trust’s Legal Provisions Relating to IETF Documents
Internet-Draft        Abfab-based Kerberos pre-auth           March 2012

(http://trustee.ietf.org/license-info) in effect on the date of
publication of this document. Please review these documents
carefully, as they describe your rights and restrictions with respect
to this document. Code Components extracted from this document must
include Simplified BSD License text as described in Section 4.e of
the Trust Legal Provisions and are provided without warranty as
described in the Simplified BSD License.

Table of Contents

1. Introduction ............................................. 3
2. Use of Abfab ........................................... 3
3. Proposed Models ......................................... 3
   3.1. The Kerberos client is the Abfab initiator .......... 3
   3.2. The Kerberos Client is the Abfab acceptor .......... 4
   3.3. Commonalities ....................................... 5
   3.4. Differences ......................................... 5
4. EAP-based pre-authentication approaches .................. 5
   4.1. EAP pre-authentication .............................. 5
   4.2. GSS-API pre-authentication ........................... 6
5. Security Considerations .................................. 6
6. Informative References .................................... 7
1. Introduction

Kerberos [RFC4120] is a widely deployed system for authentication, being integrated in multiple operating systems and network applications. However, Kerberos is typically only used to manage authentication within the scope of a single realm (typically corresponding to a single organisation). While often supported by implementations, Kerberos cross-realm operation is used relatively infrequently.

The Abfab architecture [I-D.lear-abfab-arch] describes an access management model that enables the application of federated identity within a broad range of use cases. This is achieved by building on proven technologies and widely deployed infrastructures. Some of these use cases are described in [I-D.ietf-abfab-usecases].

This draft considers two alternative models to typical Kerberos cross-realm operation that build on the Abfab architecture. The goal of this document is to describe these approaches in the expectation that this will facilitate and inform further discussion.

2. Use of Abfab

The Extensible Authentication Protocol (EAP) [RFC3748] was originally defined as an authentication framework for data links (such as PPP [RFC1661]), building on the AAA frameworks provided by RADIUS [RFC2865] and Diameter [RFC3588].

Abfab is the application of the EAP and AAA frameworks for authentication of application-level protocols, through the use of a Generic Security Services (GSS) mechanism that acts as an EAP lower-layer: the GSS EAP mechanism [I-D.ietf-abfab-gss-eap].

In the GSS EAP mechanism, the GSS initiator is the EAP peer and the GSS acceptor is the EAP authenticator. In the canonical case the GSS acceptor will act as an EAP pass-through, using an AAA protocol to federate authenticate to a remote AAA/EAP server.

3. Proposed Models

Two models involving the application of Abfab within Kerberos have been proposed.

3.1. The Kerberos client is the Abfab initiator

In this model a Kerberos client, co-located with the User Agent and acting as an EAP peer, obtains a Kerberos ticket by using an EAP credential to authenticate against a foreign realm’s KDC that is
acting as an Abfab relying party. A detailed description of this model can be found in [I-D.perez-abfab-eap-gss-preauth].

The EAP credential is authenticated by the EAP server associated with the user principal wielding the User Agent. If this ticket is a TGT, the Kerberos client can subsequently obtain service tickets from the foreign KDC in the usual way.

In either case, the Kerberos client is able to subsequently request a Kerberos service ticket from the foreign KDC and authenticate to Kerberos services within that realm.

The following figure illustrates this model:

```
User <-> Kerberos <-> EAP
Agent <----------> KDC <----------> Server

(EAP over K5 REQ/REP)
```

**Figure 1**

This model can be understood as the application of Abfab to the Kerberos authentication message exchange.

### 3.2. The Kerberos Client is the Abfab acceptor

In this model a User Agent, acting as an EAP peer, initiates a standard Abfab authentication exchange with an Abfab relying party. A Kerberos client, co-located on the Abfab relying party, obtains a Kerberos ticket (naming the principal wielding the User Agent) by using the EAP tokens to authenticate against its KDC.

As in the previous model the EAP credential is authenticated by the EAP server associated with the user principal wielding the User Agent.

The following figure illustrates this model:

```
User <-> Abfab <-> Kerberos <-> EAP
Agent <----------------> RP <----------------> KDC <----------------> Server

(Abbie) <-> (EAP over K5 REQ/REP)
```

**Figure 2**

Having an additional actor, this model is more complex than the first
model. However, the relying party and KDC can be understood as a single EAP authenticator that is split across two entities. Hence, both entities must share access to the EAP authenticator context (i.e. authenticator name, MSK, RADIUS keys...).

3.3. Commonalities

These model share the following in common:

- EAP is used to authenticate the user principal.
- The KDC is able to act as a point of authorisation for relying parties within its realm.

3.4. Differences

These models are different in the following ways:

- In the first model, the KDC must (in the general case) be exposed to KDC requests from any network location. In the second model, the KDC only needs to be exposed to trusted Kerberos services.
- In the first model, the Kerberos client must be happy to use the Kerberos discovery mechanism.
- In the first model, the User Agent must be happy to always use Kerberos if it is available.
- In the first model, the Kerberos client and KDC must be modified, but the already deployed relying parties remain unmodified. In the second model, the relying parties and the KDC must be modified.

4. EAP-based pre-authentication approaches

The models described above require of a Kerberos pre-authentication based on EAP. However, this pre-authentication can be provided by one of the following two approaches.

4.1. EAP pre-authentication

In this approach, Kerberos itself becomes an EAP lower-layer. The most straightforward way to approach this is to define a new EAP-based FAST factor. This FAST factor transports EAP packets between the EU and the KDC, following the multi round-trip procedure described in [RFC6113] (i.e. returning MORE_PREAUTH_DATA_REQUIRED error code).
This alternative is very simple, as EAP interfaces directly with Kerberos, making the architecture more straightforward. It requires the definition of the FAST factor in such a way that implements [RFC4137], which defines the interface between EAP and the EAP lower-layer.

This approach is applicable to any of the models.

4.2. GSS-API pre-authentication

In this alternative GSS-API is used by the Kerberos client and the KDC to perform pre-authentication. Hence, a pre-authentication mechanism based on the transport of GSS-API tokens is required, such as that proposed by [I-D.perez-krb-wg-gss-preauth]. Such a pre-authentication mechanism provides a generic framework where any GSS-API mechanism can be executed, without further modification to the Kerberos infrastructure.

This alternative introduces an additional layer, the GSS-API, between EAP and Kerberos. The addition of this layer implies a higher complexity of the model, but it also comes with several advantages. The first one is the flexibility it provides. Defining a generic GSS-preauth not only allows performing EAP pre-authentication, but it can be used for any other existing GSS mechanism, and for those to be defined in the future. This implies that using this alternative would serve to integrate Kerberos into any existing federation, not only those based on AAA, just by using a different GSS mechanism instead of GSS-EAP.

Besides, from a design point of view, this alternative takes advantage of the definition and implementation efforts put on the GSS-EAP mechanism of the ABFAB WG and the Moonshot project. That mechanism has already carefully implemented the interfaces defined for an EAP lower-layer.

Finally, as discussed in [I-D.perez-krb-wg-gss-preauth], using this proposal may simplify the generation of the PA-PX-COOKIE, as instead of serializing the whole EAP state machine on each round-trip, it could be possible to exchange GSS-API context handlers.

This approach is applicable to the first model. However, due the specific particularities of the second model, this approach is not easily applicable to it.

5. Security Considerations

To do
6. Informative References


[RFC3588]  Calhoun, P., Loughney, J.,


Authors’ Addresses

Alejandro Perez Mendez
University of Murcia
EMail: alex@um.es

Josh Howlett
Janet
EMail: josh.howlett@ja.net
GSS-API pre-authentication for Kerberos

draft-perez-krb-wg-gss-preauth-01

Abstract

This document describes a pre-authentication mechanism for Kerberos based on the Generic Security Service Application Program Interface (GSS-API), which allows a Key Distribution Center (KDC) to authenticate clients by using any GSS mechanism.

Status of this Memo

This Internet-Draft is submitted in full conformance with the provisions of BCP 78 and BCP 79.

Internet-Drafts are working documents of the Internet Engineering Task Force (IETF). Note that other groups may also distribute working documents as Internet-Drafts. The list of current Internet-Drafts is at http://datatracker.ietf.org/drafts/current/.

Internet-Drafts are draft documents valid for a maximum of six months and may be updated, replaced, or obsoleted by other documents at any time. It is inappropriate to use Internet-Drafts as reference material or to cite them other than as "work in progress."

This Internet-Draft will expire on July 4, 2012.

Copyright Notice

Copyright (c) 2012 IETF Trust and the persons identified as the document authors. All rights reserved.

This document is subject to BCP 78 and the IETF Trust’s Legal Provisions Relating to IETF Documents (http://trustee.ietf.org/license-info) in effect on the date of publication of this document. Please review these documents carefully, as they describe your rights and restrictions with respect to this document. Code Components extracted from this document must include Simplified BSD License text as described in Section 4.e of the Trust Legal Provisions and are provided without warranty as
described in the Simplified BSD License.

Table of Contents

1. Introduction ................................................. 3
   1.1. Requirements Language ................................. 4
2. Motivation .................................................. 4
3. Definition of the Kerberos GSS padata ...................... 4
4. GSS Pre-authentication Operation .......................... 5
   4.1. Generation of GSS preauth requests ................. 5
   4.2. Processing of GSS preauth requests ................. 5
   4.3. Generation of GSS preauth responses ............... 6
   4.4. Processing of GSS preauth responses ............... 6
5. Data in the KDC_ERR_PREAUTH_REQUIRED .................... 7
6. Supported pre-authentication facilities .................... 7
7. KDC state management ....................................... 8
8. Acknowledgements .......................................... 9
9. Security Considerations ................................... 10
10. IANA Considerations ...................................... 10
11. Normative References ..................................... 10
Authors’ Addresses ............................................. 11
1. Introduction

The GSS-API (Generic Security Service Application Programming Interface) [RFC2743] provides a generic toolset of functions that allow applications to establish security contexts in order to protect their communications through security services such as authentication, confidentiality and integrity protection. Thanks to the GSS-API, applications remain independent from the specific underlying mechanism used to establish the context and provide security.

Kerberos [RFC4120] defines a process called pre-authentication. This feature is intended to avoid the security risk of providing tickets encrypted with the user’s long-term key to attackers. The execution of a pre-authentication mechanism may require the exchange of several KRB_AS_REQ/KRB_ERROR messages before the KDC delivers the TGT requested by the client within a KRB_AS_REP. These messages transport authentication information by means of pre-authentication elements.

There exists a variety of pre-authentication mechanisms, like PKINIT [RFC4556] and encrypted time-stamp [RFC4120]. Furthermore, [I-D.ietf-krb-wg-preauth-framework] provides a generic framework for Kerberos pre-authentication, which aims to describe the features that a pre-authentication mechanism may provide (e.g. mutual authentication, replace reply key, etc.). Additionally, in order to simplify the definition of new pre-authentication mechanisms, it defines a mechanism called FAST (Flexible Authentication Secure Tunneling), which provides a generic and secure transport for pre-authentication elements. More specifically, FAST establishes a secure tunnel providing confidentiality and integrity protection between the client and the KDC prior to the exchange of any specific pre-authentication data. Within this tunnel, different pre-authentication methods can be executed. This inner mechanism is called a FAST factor. It is important to note that FAST factors cannot usually be used outside the FAST pre-authentication method since they assume the underlying security layer provided by FAST.

The aim of this draft is to define a new pre-authentication mechanism, following the recommendations of [I-D.ietf-krb-wg-preauth-framework], that relies on the GSS-API security services to pre-authenticate clients. This pre-authentication mechanism will allow the KDC to authenticate clients making use of any current or future GSS mechanism, as long as they satisfy the minimum security requirements described in this specification. The Kerberos client will play the role of the GSS initiator, while the Authentication Server (AS) in the KDC will play the role of the GSS acceptor.
1.1. Requirements Language

The key words "MUST", "MUST NOT", "REQUIRED", "SHALL", "SHALL NOT", "SHOULD", "SHOULD NOT", "RECOMMENDED", "MAY", and "OPTIONAL" in this document are to be interpreted as described in RFC 2119 [RFC2119].

2. Motivation

This work is mainly motivated by the necessity of a way to allow the KDC to make use of the technologies defined in the ABFAB WG to perform the access control of federated users. Specifically, the ABFAB architecture requires relying parties to make use of the GSS-EAP mechanism to perform authentication. The [I-D.perez-abfab-eap-gss-preauth] defines how GSS-EAP is transported on top of the GSS pre-authentication mechanism defined in this document.

3. Definition of the Kerberos GSS padata

To establish the security context, the GSS-API defines the exchange of GSS tokens between the initiator and the acceptor. These tokens, which contain mechanism-specific information, are completely opaque to the application. However, how these tokens are transported between the initiator and the responder depends on the specific application. Since GSS-API is defined as independent of the underlying communications service, its use does not require to implement any specific security feature for the transport. For instance, tokens could just be sent by means of plain UDP datagrams. For this reason, security and ordered delivery of information must be implemented by each specific GSS mechanism (if required).

Therefore, GSS tokens are the atomic piece of information from the application point of view when using GSS-API, which require a proper transport between the initiator (Kerberos client) and the acceptor (AS). In particular, the proposed GSS-based pre-authentication mechanism defines a new pre-authentication element (hereafter padata) called PA-GSS-TOKEN to transport a generic GSS token from the Kerberos client to the AS and vice-versa. The ASN.1 specification for this padata is:

```
PA-GSS-TOKEN          To be defined (TBD)
PA-GSS-TOKEN ::= OCTECT STRING --value of the GSS token
```
4. GSS Pre-authentication Operation

4.1. Generation of GSS preauth requests

The Kerberos client (initiator) starts by calling to the GSS_Init_sec_context function. In the first call to this function, the client provides GSS_C_NO_CTX as the value of the context_handle and NULL as the input_token, given that no context has been initiated yet. When using multi round-trip GSS mechanisms, in subsequent calls to this routine the client will use both, the context_handle value obtained after the first call, and the input_token received from the KDC.

The GSS_Init_sec_context returns a context_handle, an output_token and a status value. If the obtained status is GSS_S_COMPLETE, the generated token contains the necessary information to establish the context and, therefore, no further tokens are expected from the KDC to complete the authentication process. On the contrary, if the obtained status is GSS_S_CONTINUE_NEEDED, the KDC is expected to provide a token in order to continue with the context establishment process. In both cases, the Kerberos client includes the obtained output_token into a new PA-GSS-TOKEN padata and sends it to the KDC through a KRB_AS_REQ message.

4.2. Processing of GSS preauth requests

When the KDC (GSS acceptor) receives a KRB_AS_REQ message containing a PA-GSS-TOKEN, but a PA-FX-COOKIE (see Section 7) is not included, the KDC assumes that this is the first message of a context establishment, and thus GSS_C_NO_CTX is used as context_handle to invoke the GSS_Accept_sec_context routine. Conversely, if a PA-FX-COOKIE is included, the KDC assumes that this message is part an ongoing authentication and the value of the PA-FX-COOKIE is used to recover the state of the authentication (see Section 7). In both cases, after receiving the message, the KDC calls to the GSS_Accept_sec_context function, using the adequate context_handle value and using the received token in the PA-GSS-TOKEN padata as input_token.

Once the execution of the GSS_Accept_sec_context function is completed, the KDC obtains a context_handle, an output_token (optional) that MUST be sent to the initiator in order to continue with the authentication process, and a status value. If the obtained status is GSS_S_COMPLETE, the client is considered authenticated and the value of the output_token may be NULL. If the status is GSS_S_CONTINUE_NEEDED, further information is required to complete the process.
4.3. Generation of GSS preauth responses

Once the KDC has processed the input_token provided by the client (as described in Section 4.2), two main different situations may occur depending on the status value. If the client is successfully authenticated (GSS_S_COMPLETE), the KDC will reply to the client with a KRB_AS_REP message. This message will transport the final output_token (if generated) in a PA-GSS-TOKEN padata type. Additionally, there are three alternatives to encrypt the enc-part field of the KRB_AS_REP message. The first one is to make use of the client’s password as described in the standard Kerberos [RFC4120]. A second option is to strengthen this key by using keying material from the GSS context (more details are provide in Section 6). The final option is to employ a key cryptographically independent from the user’s password which could be generated by using the keying material from the GSS context. Section 6 provides further details regarding these last options.

On the contrary, if further data is required to complete the establishment process (GSS_S_CONTINUE_NEEDED), the KDC will reply to the client with a KDC_ERR_MORE_PREAUTH_DATA_REQUIRED error message [I-D.ietf-krb-wg-preauth-framework]. In the e-data field of the message, the KDC will include two padata types: a PA-FX-COOKIE containing the information that the KDC will need to regenerate the authentication state (see Section 7), and a PA-GSS-TOKEN containing the obtained output_token.

4.4. Processing of GSS preauth responses

When the client receives a KDC_ERR_MORE_PREAUTH_DATA_REQUIRED error, it extracts the token from the PA-GSS-TOKEN element and invokes the GSS_Init_sec_context function, as described in section Section 4.1. The received PA-FX-COOKIE is treated as an opaque element, which is simply copied and included into the following KRB_AS_REQ message without further processing.

On the other hand, when the client receives a KRB_AS_REP, the context establishment has finalized successfully. If the KRB_AS_REP message contains a PA-GSS-TOKEN padata type, the client invokes the GSS_Init_sec_context function using the transported input_token. Note that, to be consistent, this call MUST return GSS_S_COMPLETE and not generate any output_token, since the KDC does not expect further data from the client. Similarly, if the client does not expect any data from the KDC (it obtained a GSS_S_COMPLETE status value on the last call) and the KDC provides an input_token, an unexpected situation happens and the context establishment must be aborted.

If the context establishment is completed correctly, the client MUST
use the same process followed by the KDC (Section 4.3).

5. Data in the KDC_ERR_PREAUTH_REQUIRED

When the KDC sends a KDC_ERR_PREAUTH_REQUIRED error to the client, it includes a sequence of padata, each corresponding to an acceptable pre-authentication method. Optionally, these padata elements contain data valuable for the client to configure the selected mechanism. The data to be included in the padata for this message is described in this section.

TBD. (For example, list of the OIDs of the GSS mechanisms supported by the KDC)

6. Supported pre-authentication facilities

The pre-authentication framework [I-D.ietf-krb-wg-preauth-framework] defines a set of facilities that the pre-authentication mechanisms may provide. Specifically, the GSS pre-authentication mechanism proposed in this draft may provide the following facilities:

- Client-authentication facility. The GSS pre-authentication mechanism authenticates the client based on GSS-API calls. At the end of the GSS context establishment process, the client is authenticated against the KDC by means of the specific GSS mechanism credentials.

- Strengthening-reply-key facility. After a successful authentication, client and KDC may strengthen the reply key (the key used to encrypt the enc-part field of the KRB_AS_REP message) by adding additional keying material to it. This additional keying material can be obtained by means of calls to the GSS_Pseudo_random [RFC4401] function, although the standard GSS_getMIC function could be used if the former is not available for the specific GSS mechanism.

- Replacing-reply-key facility. Similarly to the strengthening facility, client and KDC may decide to completely replace the reply key used to encrypt the KRB_AS_REP by a new one that is cryptographically independent from the client’s password stored in client password on the Kerberos users database. To generate this keying material, the same GSS-API functions used for the previous facility would be used.

- KDC-authentication facility. This facility is also provided, as an optional feature, since the GSS-API allows the initiator of the
security context to request mutual authentication during the establishment process. If the mutual_req_flag is indicated in the GSS_Init_sec_context call, the acceptor (KDC) must be authenticated by the initiator (client) before the context is established.

The selection of the facilities that the GSS pre-authentication mechanism will provide, and how will they be negotiated with the client is still under discussion.

7. KDC state management

The Kerberos standard [RFC4120] defines the KDC as a stateless entity. This means that, if the GSS mechanism requires more than one round-trip, the client MUST provide enough data to the KDC in the following interactions to allow recovering the complete state of the ongoing authentication. This is specially relevant when the client switches from one KDC to different one (within the same realm) during a pre-authentication process. This second KDC must be able to continue with the process in a seamless way. In [I-D.ietf-krb-wg-preauth-framework], the PA-FX-COOKIE pre-authentication element is defined to transport opaque state information from the KDC to the client. This state information is included by the client in the following KRB_AS_REQ message as-is, without further processing. When the KDC receives the PA-FX-COOKIE padata, it tries to recover the state and, if successful, continue with the authentication process.

The GSS-API manages the so-called security contexts. They represent the whole context of an authentication, including all the state and relevant data of the ongoing security context. In order to deal with the statelessness of the KDC, two approaches are being discussed within the Kerberos Working Group, which are summarized in the following.

On the one hand, as the GSS-API is a collection of abstract calls, we assume that it will be the actual implementation of the GSS-API the one in charge of managing of the state associated with each security context and not the KDC, which would remain agnostic to them and, therefore, completely stateless. For example, one may think on a GSS-API implementation which is executed in an independent process to the KDC. In this way, even on an eventual restart of the KDC process (e.g. in case of inetd managed connections), the GSS-API state is not lost. Moreover, the GSS-API implementation could be executed in a distributed way, in such a way that different KDCs within the same realm could make use of the same GSS-API security contexts, even when they were not initiated by them. Note that the GSS acceptor name
will be the TGS name, which is shared amongst all the KDCs in a realm.

When this approach is followed, the contents for the PA-FX-COOKIE padata will consist on the context_handle value returned after the first call to the GSS_Accept_sec_context function. This value will be directly used by the recipient KDC for the subsequent calls to the GSS-API functions, to allow the GSS-API implementation to unequivocally identify the specific security context.

On the other hand, others think that the GSS-API must be considered as a library that is executed within the same thread or process as the calling application (the KDC in this specific case). If such is the assumption, then the KDC will be indeed storing the state associated to the security contexts, making it statefull. In that case, it would not be enough building the PA-FX-COOKIE padata just with the context_handle, as it could not be possible to recover the whole state of the KDC plus GSS-API from that information. On the contrary, the whole GSS-API security context must be serialized and included into the cookie. In order to prevent replay attacks, the PA-FX-COOKIE MUST also contain a time-stamp. Furthermore, since critical information may be contained in the exported security context, the data contained in the PA-FX-COOKIE MUST be confidentiality and integrity protected using a key shared amongst all the KDCs deployed in the realm.

The main problem with this approach is that the current GSS-API specification does not define a function that allows the exportation of a security context which has not been completely established yet. In this regard, the GSS_Export_sec_context() call is intended to create an interprocess token that can be transferred to another process within an end system. However, this call requires a completely established security context. Furthermore, despite allowed, it would require to define the contents of the PA-FX-COOKIE per each mechanism in order to determine the information required to recover the state of the authentication, which may not be trivial.

The authors of this draft defend the use of the first approach, as we consider the GSS-API implementation as something external to the GSS-API caller and, therefore, whether it stores state or not is irrelevant from the point of view of the KDC, which remains stateless.

8. Acknowledgements

This work is supported by the project MULTIGIGABIT EUROPEAN ACADEMIC NETWORK (FP7-INFRASTRUCTURES-2009-1). It is also funded by a Seneca
Foundation grant from the Human Resources Researching Training Program 2007. Authors finally thank the Funding Program for Research Groups of Excellence with code 04552/GERM/06 granted by the Fundacion Seneca.

9. Security Considerations

Protection of Request/Responses with FAST, restriction on GSS mechanism, etc. TBD.

10. IANA Considerations

This document has no actions for IANA.

11. Normative References

[I-D.ietf-krb-wg-preauth-framework]

[I-D.perez-abfab-eap-gss-preauth]


Authors’ Addresses

Alejandro Perez-Mendez (Ed.)
University of Murcia
Campus de Espinardo S/N, Faculty of Computer Science
Murcia, 30100
Spain

Phone: +34 868 88 46 44
Email: alex@um.es

Rafa Marin-Lopez
University of Murcia
Campus de Espinardo S/N, Faculty of Computer Science
Murcia, 30100
Spain

Phone: +34 868 88 85 01
Email: rafa@um.es

Fernando Pereniguez-Garcia
University of Murcia
Campus de Espinardo S/N, Faculty of Computer Science
Murcia, 30100
Spain

Phone: +34 868 88 78 82
Email: pereniguez@um.es

Gabriel Lopez-Millan
University of Murcia
Campus de Espinardo S/N, Faculty of Computer Science
Murcia, 30100
Spain

Phone: +34 868 88 85 04
Email: gabilm@um.es