More Privacy in Context-aware Platforms: User Controlled Access Right Delegation using Kerberos

PETER LANGENDOERFER and KRZYSZTOF PIOTROWSKI
IHP
Im Technologiepark 25, D-15236 Frankfurt (Oder)
GERMANY
http://www.ihp-microelectronics.com/~langend

Abstract: - In this paper we propose a distributed Kerberos architecture in which each mobile client runs her own Kerberos ticket granting server. Each of these individual TGS may provide tickets only for data that is owned by the mobile (user) on behalf of which it is executed. In addition the initial authentication phase can be done by the standard Kerberos approach as well as based on PKI using certificate chains. So our architecture gives the user back control over her personal data and it provides better scalability to the context aware platform. It also opens up the Kerberos approach for environments in which the mobile client discovers new services, which are not registered at its platform, i.e. at the Kerberos server. Our measurements indicate that running a ticket granting server on the mobile device does not inhibit a real burden. Compiling a ticket is done in about 100ms at 238 MHz and the client application size of our Java implementation is less than 50kByte.

Key-Words: - Privacy, Context awareness, Location based services, Kerberos, Middleware, JAAS, Mobile Devices

1 Introduction

Privacy has been identified already several years ago as one of the major concerns of Internet users [2, 7]. The risk for privacy will increase when contest-aware or ambient intelligence systems are in place [6, 8]. The lack of privacy may become the major pitfall of these context-sensitive systems. In order to tackle this problem a lot of work has been done. Almost each location-aware platform provides some privacy protecting functionality [4, 9, 13, 14]. But most of these solutions are focusing on protecting the current position, which makes those approaches inflexible, i.e. they cannot be applied for other context. In addition, the granularity of the protection means is quite coarse, i.e. people can be visible or invisible, which also restricts the applicability.

In this paper we present an architecture, which enables each user of a certain platform to delegate fine grained access rights to her data. The authentication and authorization mechanisms are derived from Kerberos [12], which is well analysed and known to be secure. The core concept of our approach is that each client device runs its own Kerberos ticket granting server. So each client can delegate access rights for her own data to any other client or service the client trusts. This trust may be set up by relying on the Kerberos infrastructure or by verifying PKI certificates. The major benefits of this approach are:

1. The user is back in control, since only the user can provide tickets for her data.
2. The distribution of the ticket granting server provides the system with better scalability and robustness.
3. Newly discovered services can get access to user data w/o first establishing a trust relationship between those services and the platform.

The latter may be of high importance especially in Web service based architectures.

A skeleton implementation of the architecture is already finished and the measurements done clearly indicate that running a ticket granting server on a state of the art mobile device is feasible. Compiling a ticket took about 100 ms at 238 MHz.

Deleted:

The rest of this paper is structured as follows. Section 2 provides a short state of the art, including Kerberos and other work that intends to use Kerberos in wireless networks. Then we discuss our architecture. The protocols applied are investigated in section 4. The measurement results are presented in section 5. The paper concludes with a short summary and an outlook on further research steps.

2 Related work

Kerberos was developed at MIT in the 80s to provide authentication and authorisation in campus computing network. Since then it has been revised and improved. The current version, Kerberos 5, is used by many applications and operating systems.

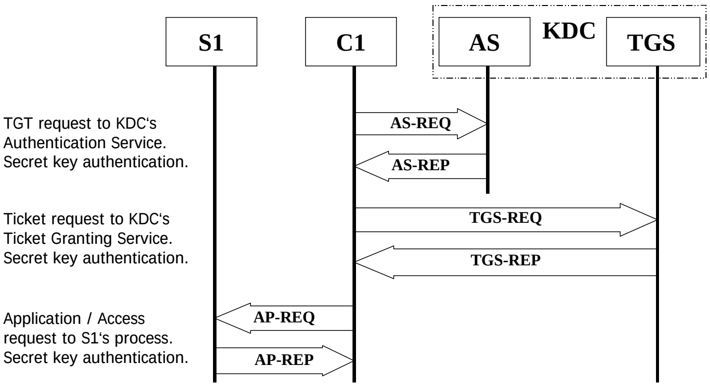
S1
C1
AS
KDC
TGS
TGT request to KDC's Authentication Service. Secret key authentication.
AS-REQ
AS-REP
Ticket request to KDC's Ticket Granting Service. Secret key authentication.
TGS-REQ
TGS-REP
Application / Access request to S1's process. Secret key authentication.
AP-REQ
AP-REP

Fig. 1: Standard Kerberos protocol

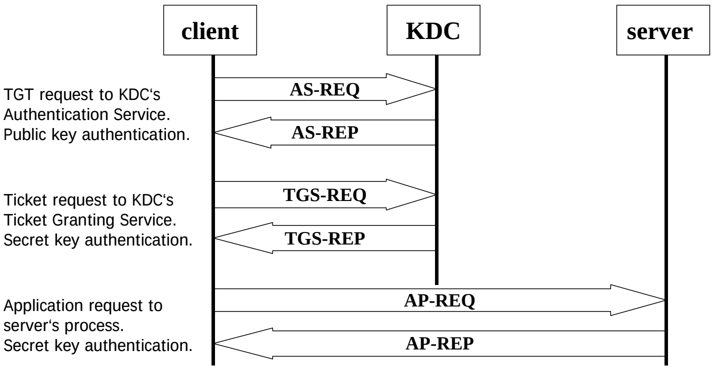
client
KDC
server
TGT request to KDC's Authentication Service. Public key authentication.
AS-REQ
AS-REP
Ticket request to KDC's Ticket Granting Service. Secret key authentication.
TGS-REQ
TGS-REP
Application request to server's process. Secret key authentication.
AP-REQ
AP-REP

Fig. 2: The PKINIT approach to use PKC in Kerberos

The term Kerberos refers to the protocol and the software implementation, but in this paper we use it to refer to the protocol only (see Fig.1).

For all the years of its employment Kerberos has been proved to be secure and reliable. The only weak point in our opinion is the need for a secure storage of secret keys that are a priori selected or agreed. This initial problem of key agreement or transfer and the need for one secret for each participant registered by the Key Distribution Center (KDC) cause the protocol to be less scalable and flexible. To solve these problems several approaches have been proposed. To reduce the burden of secret key storage public key cryptography has been applied to the initial authentication in the PKINIT approach (see Fig.2). However, this solution requires an employment of external Public Key Infrastructure (PKI). Furthermore, the public key operations are said to be more computationally expensive. This causes the need for investigating the balance between calculation cost and secret key management burden in real applications.

Here we focus on single Realm scenario, but in multi Realm application there is additional need to define the authentication mechanisms used between KDCs. But this issue is out of scope of this paper.

We also focus on mobile device applications. There are several approaches that try to optimise the Kerberos for the use on mobile devices. They focus either on the limited resources of the device (Charon) or on the ability to build ad-hoc networks (Kaman).

Charon [3] reduces the computations on the mobile device by applying a proxy between client and KDC. This optimises the operation speed, but causes several disadvantages like simply the existence of another trusted party and possible latency delays in the authentication mechanisms. Similar issues are in M-PKINIT [5], which combines Charon and PKINIT.

On the other hand, the idea of Kaman [12] is to use Kerberos authentication and authorisation protocol in ad-hoc networks. In such networks there is a problem with node persistence, thus the KDC is distributed between mobile nodes that act as authentication servers. Additionally, Kaman uses a modified Kerberos protocol known as four-pass Kerberos [1] so there is no need for the TGS (see Fig.3), i.e., the tickets are prepared by the Authentication Service (AS). But this approach still requires a priori registration at the distributed KDC and storage of information about each user at each mobile device that acts as AS.

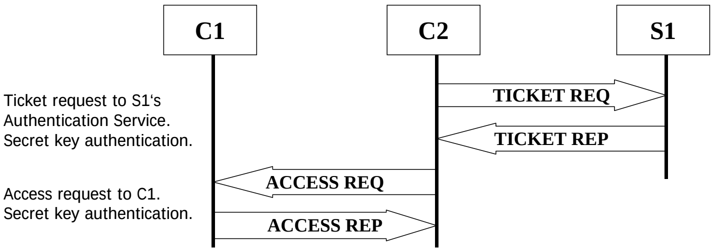
C1
C2
S1
Ticket request to S1's Authentication Service. Secret key authentication.
TICKET REQ
TICKET REP
Access request to C1. Secret key authentication.
ACCESS REQ
ACCESS REP

Fig. 3: The protocol of the Kaman approach

3 Architecture

In this section we discuss our distributed authorization delegation architecture. We first give a short overview of how the systems works before we describe in detail the architecture on the infrastructure and client side.

3.1 Overview - The basic idea

From our point of view the major challenge when protecting user privacy is that the user looses control over her data as soon as it is exposed to a certain service or other users. We want to tackle this issue by ensuring that only the user has the right and the capabilities to grant access to her data. There are two basic concepts to provide access rights: namely access right lists and capabilities. In a very dynamic and highly distributed environment, where the user is not the owner of the infrastructure in which her data is stored, managing access right list by the user is not feasible. So, the choice is that the user authorizes services etc. by providing capabilities. Kerberos is a system which implements the delegation of access rights by providing capabilities, i.e. tickets in the Kerberos terminology. Kerberos is known to be secure, so relying on this approach helps to reduce security flaws in the design to a minimum. But, there are two issues in the Kerberos approach that have to be adapted:

1. Kerberos uses a centralized service, i.e. its Ticket Granting Server (TGS) to distribute tickets among its clients.
2. Kerberos uses a centralized approach for initial authentication and trust set-up.

In the application environment we have in mind, there may be some services that are developed on top of the context-aware platform and are therefore registered at the Kerberos KDC of the platform. But since we talk about mobile devices, it seems to be reasonable to assume that a user will discover service that needs access to the users data but is not already registered at the Kerberos KDC. In order to allow the client to use such a service the authentication and trust set up has to be decentralized.

The centralized TGS contradicts the idea that the user is the only part of the system, which may grant access to his/her data. So, each mobile device has to run its own TGS. In addition no TGS may be run on the infrastructure side.

3.2 Infrastructure Architecture

In the infrastructure we distinguish three functional parts (see Fig. 4):

1. The intrinsic platform functionality such as position and profile handling, which is needed to provide useful services but irrelevant for our further discussion.
2. The Kerberos KDC, which is used for mutual authentication between the platform and its subscribers (mobile clients and services).
3. The privacy enforcement part, which ensures that data can be accessed only if a valid ticket is provided.

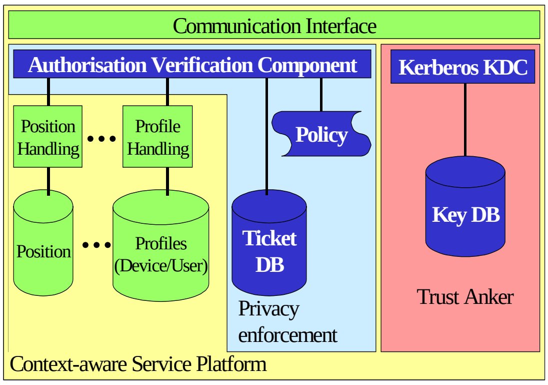

Fig. 4: Architecture of the infrastructure side

The Kerberos KDC provides the usual functionality, but only in those cases when both parties i.e. the client and the service are registered with the context-aware platform. So, it is the initial point to set-up a certain level of trust between these parties. Due to the fact that there exists also a chance to set-up trust via certificates using public key infrastructure the role of Kerberos is less crucial and the system avoids a single point of failure, which improves its robustness.

The essential part is the privacy enforcement part. It is responsible to check each incoming ticket. In addition to all checks that are done in a standard Kerberos system such as checking whether or not it is still valid etc. the privacy enforcement part has to check whether the issuing TGS may grant access to the data requested. This is needed to ensure that malicious clients do not issue tickets for other clients' data. The needed information is stored in the policy files.

3.3 Client Architecture

Two services are executed on the client in order to set-up trust between the client and a service and to delegate the access rights (see Fig. 5). The authentication service provides means to verify a TGT presented by the service in case both parties are subscribers of the same platform. It is also capable to verify a certificate chain delivered by the service in case the service is not registered at the same platform or if the Kerberos part is down for some reason. The data needed to verify the trustworthiness of a certain service is stored in the key store and trusted certificate authorities file for standard and PKI cases, respectively.

The authorization service compiles a new ticket for a certain service if the authentication phase was successful. In addition it checks the privacy preferences stored in the according policy file to determine details of the ticket under preparation such as the kind of the access right (read/write), validity time etc. This ticket is then send to the service, which presents it to the authorization verification component of the platform in order to get access to the requested data.

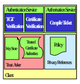

Fig. 5: Architecture of the client side

4 The protocol

To explain the protocol flow of our approach we will provide a simple scenario. Suppose that there is a mobile client *C1* that is registered in the architecture that senses current location of this client. This already implies a relationship between C1 and infrastructure. The sensed data is then stored by the infrastructure and should be available only to parties that are authorised by C1 to access this data. To simplify the further description we refer to the infrastructure components that manage the location information and access rights as the *info server*. Imagine that the client C1 performs a service discovery operation and finds out that there is a guiding service *C2* that can show her the way to a place she wants to go to. However, this guiding service requires the knowledge about the current location of C1. Thus, C1 sends a service request to C2 with the information that her location data is available at the info server and that C2 needs to authenticate himself to C1 in order to be authorized by C1 to access this data. The authentication process can be performed in two ways. C2 can prove his identity directly to C1 using the certificate chain (see Fig.6) or to the Authentication Service (AS) in the infrastructure and then request authorisation from C1 to access her data at the info server with the Ticket Granting Ticket (TGT) from AS (see Fig.7). The main difference between these two protocols is that the direct authentication and authorisation does not require C2 to be registered at the infrastructure. After the authentication process, C1, depending on her local policy, can provide C2 with a ticket that grants the access to C1's data at the info server. This ticket is additionally signed by C1 to assure the info server about the source it comes from.

Due to space limitations we do not show the structure of the packets. But all messages used are equivalent to the original Kerberos packets, i.e. they have the same structure and content. This ensures that our approach is as secure as the original Kerberos approach is, and its backwards compatability with original approach. The major difference is that we use a public key based mechanism, i.e. DSA to sign the messages. His is necessary to allow the authorization component to verify that the ticket was really issued by the client whose data is required, in cases in which the KDC of the info server is completely omitted.

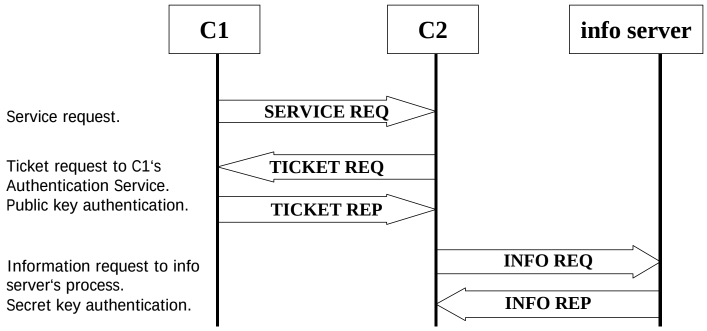
C1
C2
info server
Service request.
SERVICE REQ
Ticket request to C1's Authentication Service. Public key authentication.
TICKET REQ
TICKET REP
Information request to info server's process. Secret key authentication.
INFO REQ
INFO REP

Fig. 6: The protocol with direct authentication (service C2 to client C1)

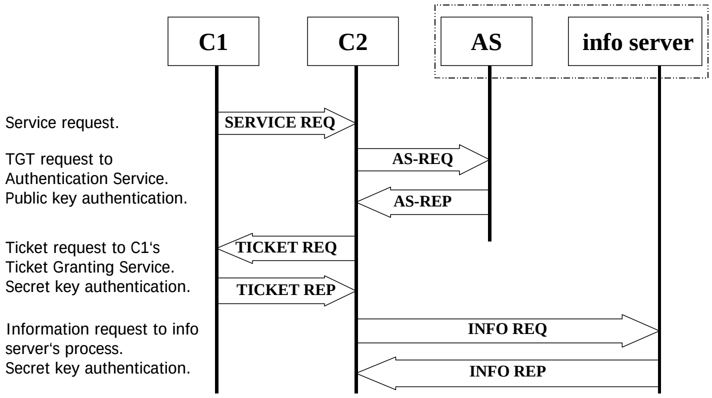
C1
C2
AS
info server
Service request.
SERVICE REQ
TGT request to Authentication Service. Public key authentication.
AS-REQ
AS-REP
Ticket request to C1's Ticket Granting Service. Secret key authentication.
TICKET REQ
TICKET REP
Information request to info server's process. Secret key authentication.
INFO REQ
INFO REP

Fig. 7: The protocol with indirect authentication (service C2 to infrastructure AS)

5 Measurements

We implemented our prototype of distributed Kerberos application using SUN Java SDK, version 1.4.2. To process the tickets we used the Java Authentication and Authorisation Service (JAAS). As the context aware platform we used our in-house approach PLASMA [10]. For the part of Kerberos that was not modified we used the MIT Kerberos 5 Release 1.2.7 that is distributed with RedHat Linux 9.

To check the applicability of our approach, we measured the time needed by a mobile device to compile a ticket. We simulated a mobile device by a Pentium-M laptop with CPU clock rate reduced to 238 MHz for a first measurement setup and to 595 MHz for the second. These two setups are approximately of the performance of a typical PDA (250-600MHz). For each CPU speed we performed 10 passes. For each pass the client was creating the ticket including signing it with the DSA signature. We also additionally measured how much time the client needs to pack the ticket into a byte stream that can be send over the network. The results of our measurements are provided in Table 1.

	Ticket compilation [ms]		Ticket packing [ms]	
	238 MHz	595 MHz	238 MHz	595 MHz
1	90	80	3,01	1,1
2	70	50	2,8	1,1
3	80	90	3,21	1,2
4	70	81	2,9	1,1
5	140	60	2,7	1,17
6	120	80	2,7	1,1
7	120	100	2,81	1,2
8	70	70	2,8	1,11
9	180	61	2,7	1,1
10	70	90	3,01	1,15
average	101	76,2	2,864	1,133

Table 1: Time measurements

The size of implemented Java Class files is less than 50kBytes.

As shown in Table 1, the time needed to compile a ticket and pack it to a byte stream is approximately 100 ms. Thus, our measurements show that it feasible to run the Kerberos Ticket Granting Service on a mobile device without exhausting its resources.

6 Conclusions

In this paper we proposed a distributed Kerberos architecture in which each mobile client runs own Kerberos ticket granting server. Each of these individual TGS may provide tickets only for data that is owned by the mobile (user) on behalf of which it is executed. In addition the initial authentication phase can be done by the standard Kerberos approach as well as based on PKI using certificate chains. The major benefits of our approach are:

1. The user is back in control, since only the user can provide tickets for her/his data.
2. The distribution of the ticket granting server provides the system with better scalability and robustness.
3. Newly discovered services can get access to user data w/o first establishing a trust relationship between those services and the platform.

Our measurements indicate that running a ticket granting server on the mobile device does not inhibit a real burden. Compiling a ticket is done in about 100ms at 238 MHz and the client application size is less than 50kByte.

In our next research steps we will focus on measurements of the whole protocol, and set up experiments to verify the benefit of our approach with respect to scalability. We also intend to migrate the implementation on a PDA as soon as modular Kerberos support is provided. Then we are going to use our own AES and Elliptic Curve Cryptography (ECC) implementations to replace DES and RSA (or DSA) respectively. We expect a significant gain with respect to processing time, if these cipher mechanisms are applied.

Acknowledgement

This work was partially funded by the German government under grant 01AK044C and grant 01AK047F.

References:

[1] D. W. Carman, P. S. Kruus, B. J. Matt, Constraints and approaches for distributed sensor network security, Technical Report 00-010, NAI Labs.

[2] L.F. Cranor, Beyond Concern: Under-standing Net Users' Attitudes About Online Privacy, In: Ingo Vogelsang and Benjamin M. Compaine, eds. *The Internet Upheaval: Raising Questions, Seeking Answers in Communications Policy*, Cambridge, Massachusetts: The MIT Press, p. 47-70, 2000.

[3] A. Fox, S. D. Gribble, Security on the move: Indirect Authentication using Kerberos, *Proc. of the Second Annual International Conference on Mobile Computing and Networking*, 155-164.

[4] M. Gruteser, D. Grunwald: Anonymous Usage of Location-Based Services Through Spatial and Temporal Cloaking, ACM/USENIX International Conference on Mobile Systems, Applications, and Services (MobiSys), 2003

[5] A. Harbitter, D. A. Menasce, The performance of public key enabled Kerberos authentication in mobile computing applications, *Proc. of the 8th ACM conference on Computer and Communications Security*, 78-85.

[6] IST Advisory Group (ISTAG) chap. 4 *Ambient Intelligence*: from Vision to Reality, In G. Riva, F. Vatalaro, F. Davide, M. Alcañiz (Eds.) IOS Press, 2005, http://www.ambientintelligence.org

Field Code Changed

[7] Juniper Research, Consumers worried about online privacy, 2002, available at: http://www.nua.com/surveys/index.cgi?f=VS&art_i
905358019&rel=true; last visited May 2005

Field Code Changed

[8] P. Langendoerfer: m-commerce why it does not fly (yet?), *Proceedings of the International Conference on Advances in Infrastructure for e-business, e-Education, e-Science and e-Medicine on the Internet*, 2002.

[9] P. Langendörfer, R. Kraemer: Towards User Defined Privacy in location-aware Platforms, *Proceeding of the 3rd international Conference on Internet computing*, USA. CSREA Press, 2002.

[10] P. Langendoerfer, O. Maye, Z. Dyka, R. Sorge, R.Winkler, R. Kraemer. Middleware for location-based services: Design and implementation issues. In Q. Mahmoud (Ed.): Middleware for Communication. Wiley, 2004.

[11] A. Pirzada, C. McDonald, Kerberos assisted authentication in mobile ad-hoc networks, *Proceedings of the 27th Australasian Computer Science Conference (ACSC) 26(1)*, 41-46.

[12] J. Steiner, C. Neuman, J. Schiller, Kerberos: an authentication service for open network systems, *Proceedings Usenix Winter Conference*, Berkeley 1988.

[13] K. Synnes, J. Nord, P. Parnes: Location Privacy in the Alipes platform. *Proceedings of the Hawai'i International Conference on System Sciences (HICSS-36)*, Big Island, Hawai´i, USA, January 2003.

[14] W. Wagealla, S. Terzis, C. English: Trust-based Model for Privacy Control in Context-aware Systems, 2nd Workshop on Security in Ubiquitous Computing, Ubicomp, 2003