A Survey of Kerberos V and Public-Key Kerberos Security

Minkyu Kim, mk13@cec.wustl.edu (A project report written under the guidance of Prof. Raj Jain)

Download 

Abstract

Kerberos was initially developed at MIT as a part of Project Athena and in these days it is widely deployed single sign-on protocol that is developed to authenticate clients to multiple networked services. Furthermore, Cross-realm authentication is a useful and interesting component of Kerberos aimed at enabling secure access to services astride organizational boundaries. Also, Kerberos has continued to evolve as new functionalities are added to the basic protocol and one of well-known these protocols is PKINIT. First, I review and analyze the structure of Kerberos recently proposed and the cross-realm authentication model of Kerberos. Also, I discuss PKINT, an extension version of Kerberos, which modifies the basic protocol to allow public-key authentication. Although Kerberos has been proven its strengths so far, it also has a number of limitations and some flaws. I dedicate my efforts to an analysis of PKINIT and mainly focus on a number of vulnerability, flaws and attacks lately discovered on Kerberos as well as PKINIT in this paper. Lastly, I introduce several possible solutions to enhance Kerberos.

Keywords

Kerberos, Attack on Kerberos, PKINT, Kerberos 5, Kerberos security, Reply attack, Password attack, Guessing attack. Cross-Realm Authentication

Table of Contents

---

1 Introduction

Kerberos was initially designed at MIT as a part of Project Athena [Neuman06] . It has been successfully deployed as a single sign-on protocol that is designed to authenticate clients to multiple different network services. There have been two different versions of the protocol in widely used, known as Kerberos 4 and 5. Kerberos 5 is the most recently proposed and is a trusted third-party authentication mechanism designed for TCP/IP networks. It uses strong symmetric cryptography to enable secure authentication in an insecure network. Currently it is available for all major operating systems, e.g., Linux, Microsoft Windows as well as Apple's OS X. Furthermore, Kerberos 5 has been improved as new functionalities are added to the basic protocol and one of these results is known as PKINIT [Zhu05] (Public-Key Cryptography for Initial Authentication) which modifies the basic protocol to allow public-key authentication and it causes considerable complexity to the protocol.

Regarding the security issues of Kerberos, it has been discussed in several papers which represents possible weak points including replay attacks, password attack against Ticket-Granting tickets or pre-authentication data, attacks against network time protocols (Kerberos requires time synchronization) and malicious client software. Furthermore, a guessing attack and particularly man-in-the-middle attack in PKINIT have been discovered. Before discussing flaws and weakness of Kerberos, in Section 2-4, an analysis of the structure of Kerberos 5, intra- and cross-realm authentication as well as a detailed description of PKINIT will be reviewed.

In Section 5-7, I discuss the flaws and attacks on Kerberos. In Section 5, I focus on the attacks on the basic protocol, Kerberos 5 without PKINIT, such as the password attack, reply attack and guessing attack. Firstly, regarding the reply attack, I reason that it is feasible by presenting attacks on both SMB and LDAPv3. An attacker will be able to access file shares and modify directory entries with the victim's credentials. Some server implementations have actual weaknesses, while others have default configurations that make the attack possible. Secondly, I show that a password attack is feasible, thus allowing the attacker to discover weak user passwords. Pre-authentication data are used for this attack. A replay attack is presented with the SMB protocol. This allows an attacker to access file shares with the victim's credentials without actually knowing the password. Lastly, in many computer systems, users are authenticated via passwords which they choose. Unfortunately, people tend to choose easy-to-remember passwords, which are vulnerable to guessing attacks. A malicious attacker can guess such passwords using the words in a machine-readable dictionary. I show that Kerberos is one of many existing authentication protocols which are vulnerable to so-called off-line guessing attacks, and In Section 8, I will discuss some useful guidelines to be secure against guessing attack as well as other attacks. Based on these guidelines, I will discuss a possible solution to enhance Kerberos protocol so that it can resist the each of attacks.

In Section 6, I discuss the attack on PKINIT, particularly man-in-the-middle attack, which allows an attacker to impersonate Kerberos administrative principals Key Distribution Center(KDC) and end-servers to a client, therefore breaching the authentication guarantees of Kerberos. It also gives the attacker the keys that the KDC would normally generate to encrypt the service requests of this client, hence defeating confidentiality as well, In Section 7, I will discuss about the possible enhancement for scalability and reliability issues in Kerberos cross-realm operation, followed by in Section 9, I provide some concluding remarks.

2 Kerveros V Basic

Networked computer systems provide a great number of shared resources at a user's fingertips; without leaving one's desk, remote hosts, file servers, printers, and many other networked services are readily at hand. Authentication and other security mechanisms are needed so that this convenience is not abused, especially where one's personal computer or organization network is at the risk of dangerous backdoors when connected to the Internet. A simple solution to this problem, requiring users to authenticate to each service they use (for example using a password) is not only inconvenient, but also insecure in practice as people are poor at dealing with a large number of different passwords.

The Kerberos protocol was designed to provide transparent access to all the networked resources a legitimate user needed for a typical day once he/she logs on his/her terminal [Neuman06] . For example, each time the user needs to retrieve a file from a remote server, the required authentication will be handled by Kerberos securely behind the scene, with no user's intervention needed.

This section will review how the latest verion of this protocol, Kerberos 5 [Neuman06] , achieves secure authentication based on a single logon, and for the time being on situations where all the authentications take place within the same administrative domain (or realm) without PKINIT.

2.1. Principals

The informal example above has described three of principals, that form a typical Kerberos exchange: the human user at his/her terminal, the client process that recognizes the user's password and transparently handles the authentication of each request on the user's behalf, and the requested services, or servers in Kerberos terminology. Kerberos relies on two additional administrative principals together, namely the KDC: the Kerberos Authentication Server (KAS) which authenticates the user and provides the corresponding client with credentials to use the network for a typical day, and the Ticket Granting Server (TGS) which authenticates the client to each requested server based on those credentials. The high-level picture is given in Figure 1.

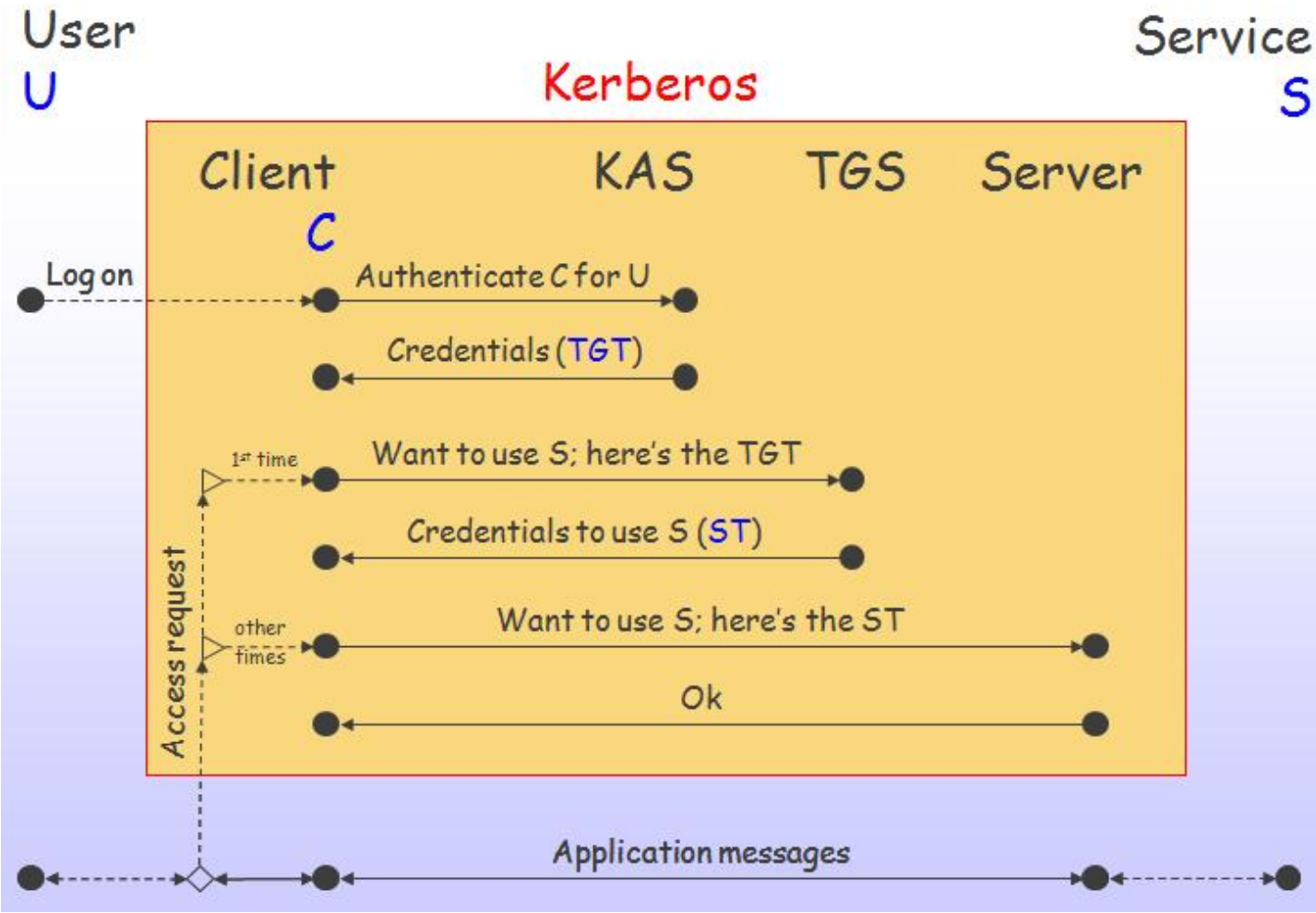

Figure 1: An Overview of Kerberos V Operation

The top of the figure represents the daily authentication process to Kerberos: as the user (U) logs on, the KAS authenticates the client process representing the user and provides credentials to use the system for that day. These credentials from the KAS are called the Ticket Granting Ticket (TGT). Whenever the user wants to use a networked service, the client on his/her behalf will seek authentication to the process S managing this service. This is done in two steps: the first time U attempts to access S, C presents the TGT from the KAS to the ticket granting server (TGS) who will in turn provide credentials for S. These credentials are called the Service Ticket (ST). Every subsequent time U wants to access this particular service, C forwards ST to S, without involving the TGS. The line at the bottom of the figure represents the actual use of the desired service: this is all the user sees as the client process handles the authentication overhead.

The above mode of interaction represents a typical single organization, or realm in Kerberos terminology. Each realm is regulated by a single KDC, although there may be synchronized replicas for performance and fault tolerance reasons. Within a realm, there will be generally multiple clients and multiple servers. Intra-realm authentication, as this modality is known, is widely deployed and has been extensively studied. Kerberos also supports cross-realm authentication [Bella07] [Butler01] [Mitcell08] , a scheme by which a client in a realm R1 can access a service in a different realm Rn. The rest of this paper will explore how Kerberos achieves cross-realm authentication. Firstly, let's recall how the basic intra-realm protocol works.

2.2. Message Exchange within the same administrative realm

This section focuses on the messages exchanged during a typical intra-realm authentication session between a

client C and a server S, as shown in the box of Figure 1 [Cervesato09] . Sufficient detail is provided to support their formal specification in the next section. However, it is important to notice that Kerberos is far more complex than the abstract view given here. The simplified version of the Kerberos 5 exchanges is given in Figure 2: the top part relies on the traditional "Alice-and-Bob" notation, with the standard name [Figure 2] for each message given on the left. I will now explain each of the three roundtrips between a client (C) and the KAS (K), the TGS (T), and a server (S), respectively.

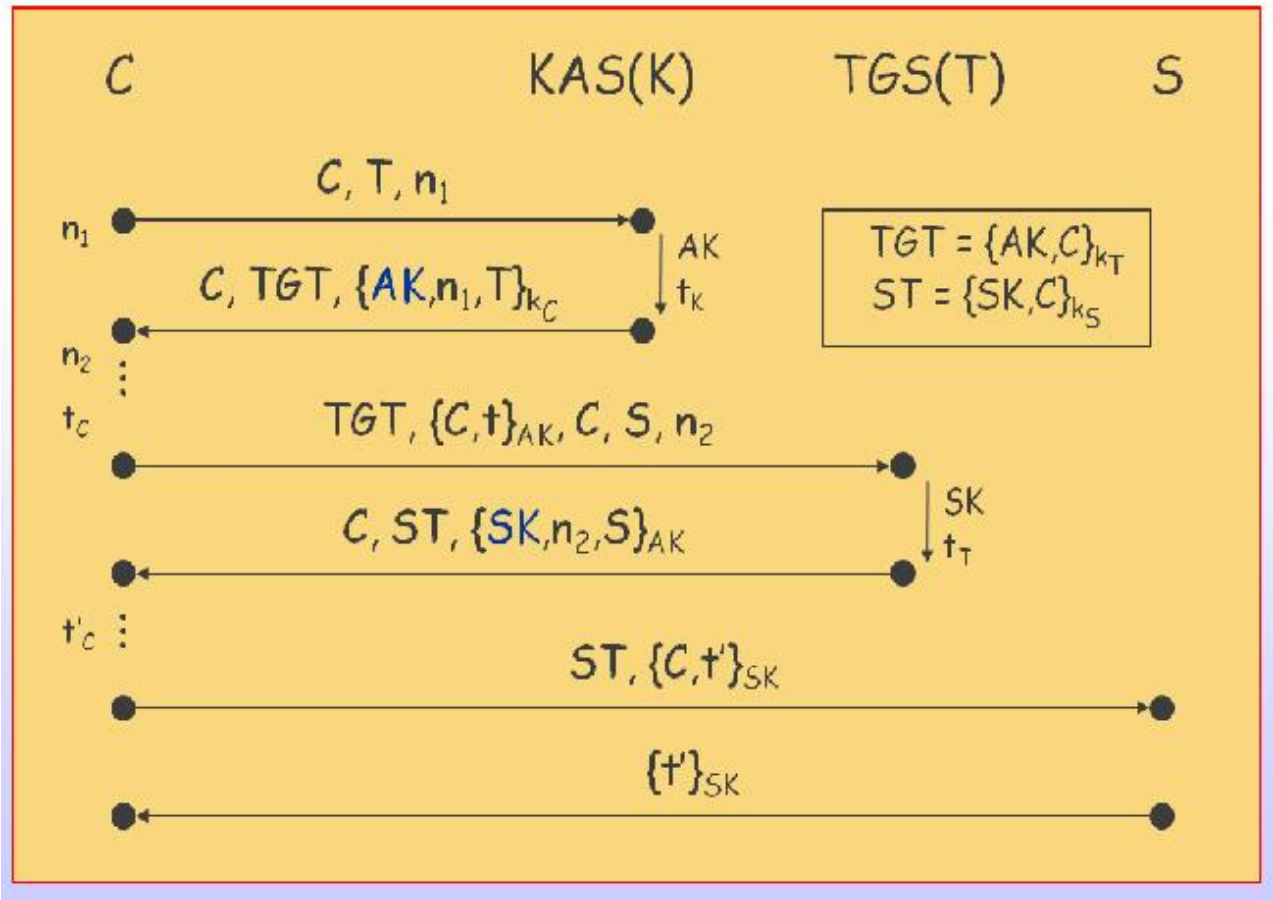

Figure 2: Message Exchange within the same administrative realm

Authentication Service Exchange (C <—@gt; K):

- This exchange takes place as the user first logs on to a Kerberized network. The client process C generates a nonce n1 and sends it to the KAS together with its own name, C, which represents the user, and the name of the TGS (officially "krbtgt", here abbreviated as T).
- After recognizing C, the KAS replies with a message containing two encrypted components: the ticket granting ticket (TGT) {AK,C, tK}kT that is cached by C and will be used to obtain service tickets for the rest of the day, and {AK, n1, tK, T}kC with which the KAS informs C of the parameters of the ticket. AK is the autentication key and the TGT is meant for the TGS and is encrypted with the long-term key kT that the KAS shares with the TGS. It contains a newly generated authentication key AK and a timestamp tK in addition to C's name. The key kC used to encrypt the second component is a longterm secret between C and the KAS derived from the user's password. AK will be used in every subsequent communication with the TGS, sparing the more vulnerable kC. The timestamp tK will assure the TGS and C that this ticket was issued recently, as all Kerberos agents have loosely synchronized clocks. The nonce n1 in the second component binds this response to C's original request.

Ticket Granting Exchange (C <—@gt; T):

- This exchange takes place the first time U attempts to access a service S. In the outgoing message, C transmits the cached TGT and S's name together with a newly generated nonce n2, and the authenticator {C, t}AK, where t is a timestamp. The authenticator proves to T that C actually knows the authentication key AK.
- After authenticating C and verifying that it is allowed to use S, the TGS sends a response with the same structure as the second message above except the service ticket {SK,C, tT }kS is now encrypted with the long-term key shared between the KDC and S, and it contains a freshly generated service key SK, C's name, and a timestamp tT . The other encrypted component is as in the second message above, but now encrypted with the authentication key AK. C caches the service ticket.

Client/Server Exchange (C <—@gt; S):

- This exchange takes place each time the client initiates a new session with the server S. With a service ticket in hand, C simply contacts S with this ticket and an authenticator similar to the one described above.
- The response from S is optional as the subsequent application exchanges may subsume it. When present, it provides assurance to C that S is alive, for example by returning the timestamp t0C that C included in its request, encrypted with the service key.

2.3. Security Consideration

One weakness of the standard Kerberos protocol lies in that the key kC used to encrypt the client's credentials is derived from a password, and passwords are undoubtedly vulnerable to dictionary attacks [Newman01]. In addition, since the initial request is completely plaintext, an active attacker can repeatedly make requests for an honest client's credentials and accrue a large number of plaintext-ciphertext pairs, the latter component being encrypted with the client's long-term key kC (which is derived from a password). While the attacker is unable to use these credentials to authenticate to the system, he is given considerable opportunity to perform an active dictionary attack against the key.

Kerberos can optionally use pre-authentication, a feature designed to prevent an attacker from actively requesting and obtaining credentials for an honest user. Pre-authentication functions by requiring the client to include a timestamp encrypted with his/her long-term key in the initial request. The authentication server will only return credentials if the decrypted timestamp is recent enough. This method successfully prevents an attacker from actively obtaining ciphertext encrypted with the long-term key; however, it does not prevent passive dictionary attacks, i.e., a passive attacker could eavesdrop on network communications, record credentials as the honest client requests them, and attempt off-line dictionary decryption. Hence, pre-authentication makes it slower for an attacker to perform cryptanalysis against the user's long-term key, but it does not prevent the attack. PKINIT, along with a number of other methods, aims at eliminating this dictionary attack vulnerability. In Section 4, I will introduce PKINIT and concentrate on the PKINT attack in Section 6.

3. Kerberos Cross-Realm Authentication

Kerberos supports authentication across organizational boundaries by permitting clients and servers to reside on different realms. A realm consists of a group of clients, a KDC, and application servers. For example, the Network Security group in the CSE Department of Washington University in Saint Louis may create an independent realm RNS with its own users, services and administrators. Similarly, the CSE department may organize a Kerberos realm RCSE to allow CSE members to access shared resources, and the University may as well have a realm RW to operate university-wide resources such as printers and scanners in computer labs. Cross-realm authentication enables a student at her workstation in the Network Security group to transparently access a file on the common CSE server, and even to smoothly print it on a printer in any computer lab. Without cross-realm authentication this student would need a separate account in each realm, log onto each of them, and

transfer files from account to account in order to achieve the same goals. This is inconvenient, not scalable, and less secure as several passwords would be needed, one for each realm.

In the simplest case, the cross-realm authentication of a client C in realm R1 to a server S in Rn is accomplished by registering the KDC of Rn as a deginated server in R1 and using a variant of the intra-realm protocol to first acquire a TGT for C in R1. And then, a ST for Rn's KDC seen as a local service in R1 [Cervesato09] . This ST has the same format as a TGT for C in Rn, and as such it is submitted to the KDC of Rn to obtain a service ticket for accessing S. The key used by R1's KDC to encrypt the ticket for the special service corresponding to Rn's KDC is called a cross-realm key. This is all Kerberos 4 allows. In Kerberos 5, C's access to S may require traversing intermediate realms R2, ... , Rn-1 if there is no cross-realm key between R1 and Rn, but R1 has such a partnership with R2, R2 with R3, etc. up to Rn. C then needs to obtain a TGT for each of these realms in succession before accessing S. The list of traversed KDC's [R1, ... ,Rn] is called the authentication path of C's access to S. This highlevel description [Cervesato09] is schematically shown in Figure 3.

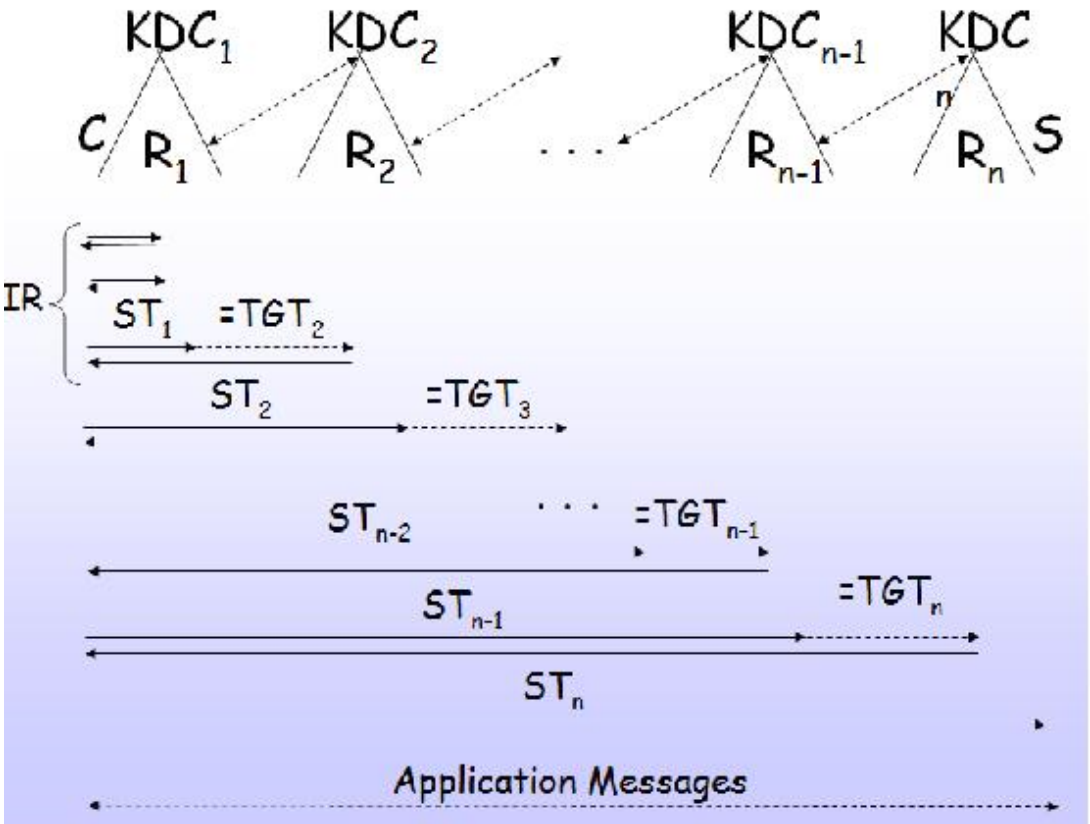

Figure 3: Schematic Cross-Realm Authentication

3.1. Issues in Kerberos Cross-Realm Operation

In the following sections, I will introduce several issues related to cross-realm operations followed by a discussion on the possible ways to enhance it. The cross-realm operations in Kerberos allows users to access services offered by foreign realms either in roaming scenarios where the user is physically located in a visited realm or in remote access scenarios where the user needs to access the remote application service from his/her home realm.

Inter-realm trust management:

- In Kerberos, the cross-realm operations assume that realms have direct or indirect trust relationship. A

direct trust relationship means that the realms involved in the cross-realm authentication share keys and their respective TGS principals are registered in each other's KDC. When direct trust relationships are present, the KDC of each realm must maintain keys with all foreign realms. This can become a cumbersome task as the number of realms increases. Therefore, cross-realm authentication based on indirect trust relationships offers better scalability.

- On the other hand, indirect trust relationship means that there is a 'chain of trust' linking two realms. By having a common trusted realm or a chain of intermediary trusted realms can realize this. If the realms belong to the same institution, a chain of trust can be determined by the client or the KDC by following the DNS domain hierarchy and supposing that the parent domains share keys with all its child sub-domains [Neuman 01]. However, if the inter-realm trust model does not follow the hierarchical approach, the trust paths must be specified manually. In such case, the management of inter-realm trust may become a quite troublesome task.

Reliability and Forward Secrecy:

- When intermediary realms are involved, the success of cross-realm authentication completely depends on the realms that are part of the authentication path. If any of the realms in the authentication path are not available, the principals of the end-realms will not be able perform cross-realm operations. This constitutes a reliability issue that can fail Kerberos as a promising authentication system for mission-critical deployments such as large factory automation and military applications.
- Furthermore, any KDC in the authentication path can learn the session key that will be used between the client and the desired service, this means that any intermediary realm is able to misrepresent the identity of the service and the client as well as to eavesdrop on the communication between the client and the server [Mitcell08] . If an intermediary KDC is corrupted, all authentication operations using the corrupted KDC will be corrupted. The forward secrecy issue in cross-realm operations is a serious problem, which makes the whole web of trust as vulnerable as the weakest KDC.

Client centralized exchanges:

- During cross-realm operations, Kerberos clients have to perform TGS exchanges with all the KDCs in the trust path, including the home KDC and the target KDC. In the case where the client has limited computational capabilities, the overhead of these cross-realm exchanges may grow into unacceptable delays. Moreover, if the number of intermediary realms increases, the delay caused by the cross-realm messages can result in unacceptable delays independently from the hardware characteristics of the user's device.

4. Public-Key Kerberos: PKINIT

PKINIT is known as an extension to Kerberos 5, which uses public key cryptography to avoid shared secrets between a client and KAS [Zhu05] ; it modifies the AS exchange. However, other parts of the basic Kerberos 5 protocol are the same. The long-term shared key (kC) in the traditional AS exchange is typically derived from a password, which limits the strength of the authentication to the user's ability to design and memorize good passwords; PKINIT does not use kC and thus solves this issue. Also, PKINIT allows network administrators to use an existing public key infrastructure (PKI) rather than expend additional effort on managing users' long-term keys needed for traditional Kerberos. This protocol extension adds complexity to Kerberos as it retains symmetric encryption in the later rounds but relies on asymmetric encryption, digital signatures, and corresponding certificates in the first round [Tsay03] .

In PKINIT, the client C and the KAS has independent public/secret key pairs, e.g.,(pkC, skC) and (pkK, skK). Certificate sets CertC CertK issued by a PKI independent from Kerberos are used to testify the binding between each principal and its purported public key [Tsay03] . This simplifies administration as authentication decisions can now be reached based on the trust the KDC holds in just a few known certification authorities within the

PKI, rather than keys individually shared with each client. Dictionary attacks are defeated as user-chosen passwords are replaced with automatically generated asymmetric keys. The login process changes as very few users would be able to remember a random public/secret key pair. In Microsoft Windows, keys and certificate chains are stored in a smartcard that the user swipes in a reader at login time. A passphrase is generally required as an additional security measure [Clercq10] . Other possibilities including keeping these credentials on the user's hard drive, are again protected by a passphrase.

In RFC 4556 [Zhu05] , as the PKINIT extension to Kerberos has recently been defined after a sequence of Internet Draft found in [IETFSeq04] , Cervesato et al. use "PKINIT-n" to refer to the protocol as specified in the nth draft revision and "PKINIT" for the protocol more generally and these drafts and the RFC can be found at [IETFSeq04] .

There are two operation modes in PKINIT. First, in public-key encryption mode, the key pairs,e.g.,(pkC, skC) and (pkK, skK), are used for both signature and encryption. The latter is designed to protect the confidentiality of AK, while the former ensures its integrity. Another mode is known as Diffie-Hellman (DH) mode, the key pairs are used to provide digital signature support for an authenticated Diffie-Hellman key agreement which is used to protect the fresh key AK shared between the client and KAS. A variant of this mode allows the reuse of previously generated shared secrets. In the following section, I will take a look more detail about these two modes.

4.1 Public-key encryption mode

In PKINIT-26, the AS exchange is illustrated in Figure 2. In discussing this and other descriptions of the protocol, Cerversato et al. write [m]sk for the digital signature of message m with secret key sk. (PKINIT realizes digital signatures by concatenating the message and a keyed hash for it, occasionally with other data in between). Cerversato et al. make the standard assumption that digital signatures are unforgeable [Goldwasser11] . Denote that the encryption of m with public key pk is {{m}}pk because, as earlier, I indicated that {m}k is for the encryption of m with symmetric key k.

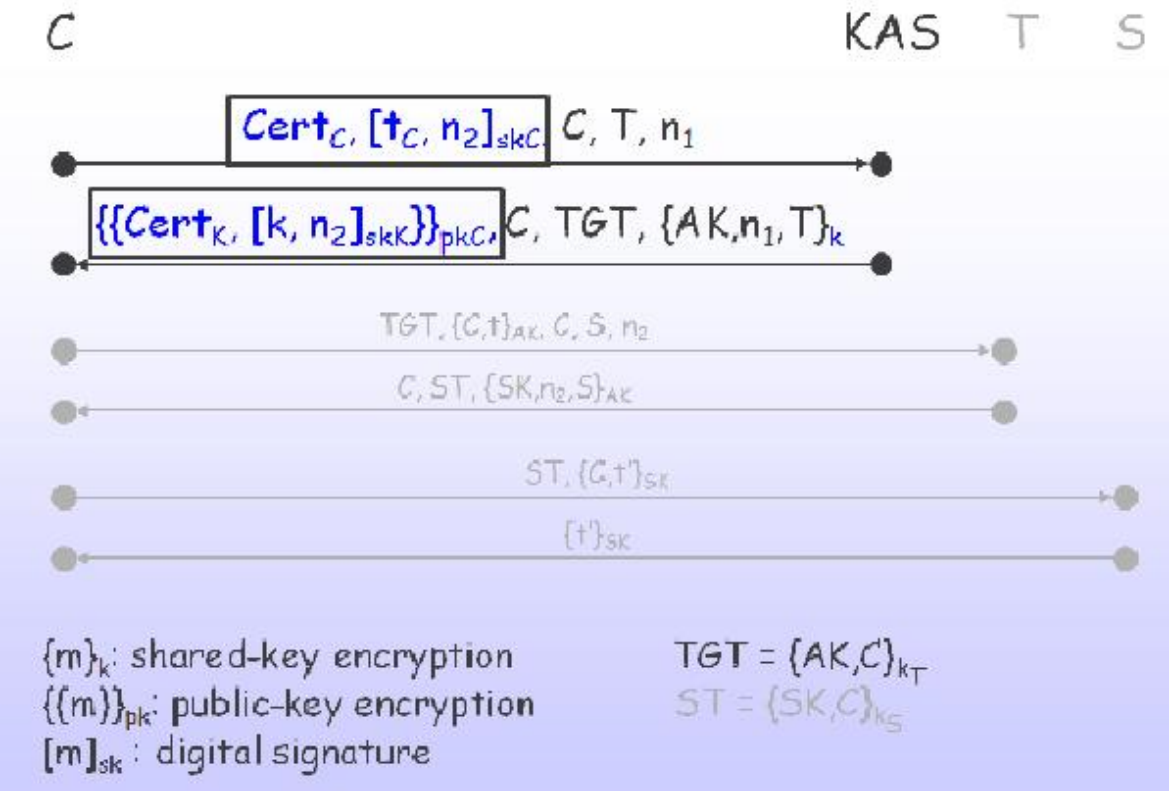

Figure 4: Public-key encryption mode

First line in Public-key Encryption mode:

- This shows the relevant parts of the request that a client C sends to a KAS K using PKINIT-26. The last part of the message "C, T, n1" is exactly the same as in basic Kerberos 5, containing the client's name, the name of the TGS for which he/she wants a TGT, and a nonce. The "boxed" parts are added by PKINIT and contain the client's certificates CertC and his/her signature (with the secret key skC) over a timestamp tC and another nonce n2. The nonces and timestamp to the left of this line indicate that these are generated by C particularly for this request, with the box indicating data not included in the abstract formalization of basic Kerberos 5 [Tsay03] [Butler01] . This effectively implements a form of pre-authentication.

Second line in Public-key Encryption mode:

- This shows the formalization of K's response, which is more complex than that of basic Kerberos. The last part of the message "C, TGT,{AK, n1, tK, T}k"is very similar to K's reply in basic Kerberos; the difference boxed is that the symmetric key k protecting AK is now freshly generated by K and not a long-term shared key. The TGT and the message encrypted under k are as in traditional Kerberos. Because k is freshly generated for the reply, it must be informed to C before C can learn AK. PKINIT does this by adding the boxed message {{CertK, [k, n2]skK}}pkC. This contains K's certificates and its signature, using its secret key skK, over k and the nonce n2 from C's request; all of this is encrypted under C's public key pkC.

4.2 Diffie-Hellman mode

I will briefly describe the Diffie-Hellman (DH) mode of PKINIT in this section, although Cervesato's preliminary investigation did not reveal any flaw in this mode [Butler02] . It should be noted that this mode appears not to have been included in any of the major operating systems. The only support can be found is within the PacketCable system [CTL12] , developed by CableLabs, a cable television research consortium.

In the DH moed, except that k is generated by a Diffie-Hellman key exchange instead of the KDC using some key generation algorithm, it is very simliar to the public-key encryption mode. Figure 5 shows the messages involved. The first message is the same except the signature contains, in addition to a timestamp and nonce, Diffie-Hellman parameters and C's public Diffie-Hellman value.

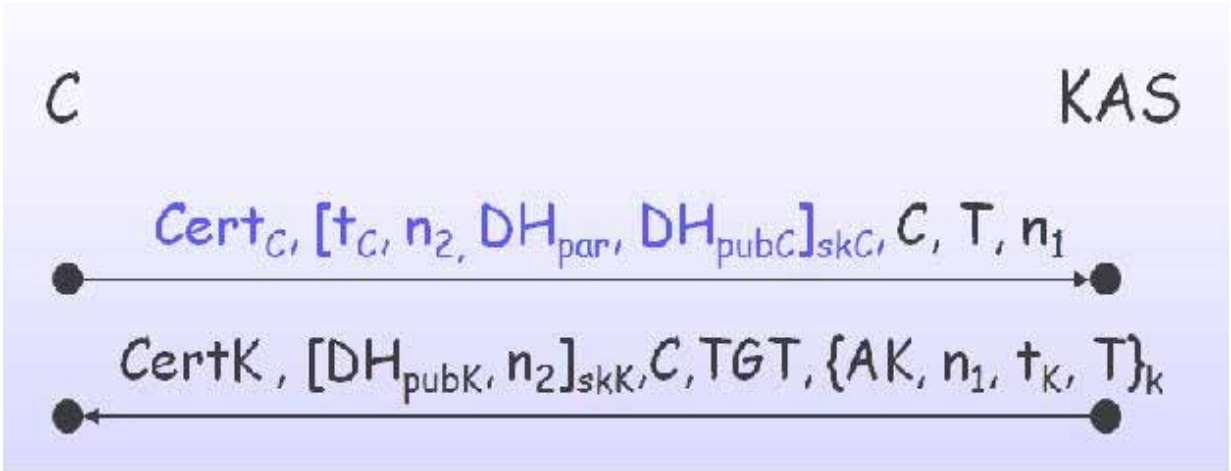

Figure 5: Diffie-Hellman mode

This will perform the same verification actions as in the public-key mode and K will also check that the parameters for the Diffie-Hellman key generation are acceptable when K receives this message. For instance, the system administrator may configure a minimum key length. If everything is valid then K will send a reply as in the second message of Figure 5. K's reply is again similar to the public-key encryption method, except that now it is unnecessary to encrypt the information used to obtain k. This is because k is derived from a Diffie-Hellman key exchange and, under the common assumption that discrete logarithms cannot be computed in polynomial time, learning the public values gives an attacker no information about the actual derived key. K sends n2 and its public Diffie-Hellman value DHpubK signed to C. The signature is necessary for ensuring that DHpubK was created and sent by K.

C receives K's reply and validates it by checking K's certificate and the signature across DHpubK, n2. If everything is valid then C computes the Diffie-Hellman shared secret using DHpubK and its own private Diffie-Hellman value, and then can use the shared secret to derive k. C will then decrypt the authenticator and the protocol proceeds as per the standard Kerberos protocol. Since Diffie-Hellman key generation is expensive, it is desirable to reduce the load on the KDC by reusing Diffie-Hellman shared secrets.

For this reason a variant of the DH mode exists which reuses previously generated Diffie-Hellman shared secrets to derive new keys. In this mode the first message is identical to that of the DH mode except C also includes an extra nonce nC in the signed data. This nonce will be used to compute the new key from the existing Diffie-Hellman secret.

5. Attacks on Kerberos V

Kerberos V implicitly relies on the servers being secure and software being nonmalicious. However, the most interesting assumptions are the ones about password guessing and replay attacks. Both attacks are non-trivial but could be carried out over the local network. Password guessing attacks can be based on any text encrypted with the key derived from the victim's password, and will result in exposure of the plaintext password. Replay attacks will usually result in the attacker assuming the victim's identity without actually recovering the password. We will discuss both attacks in the next chapter.

In the following sections, I discuss how an attacker might hijack a network connection allowing active monitoring and modification of the victim's network traffic.

5.1 Hijacking a Network Connection on a Switched Network

According to Kasslin and Tikkanen [Kasslin14] , to hijack a network connection of the target machine we have to be able to direct the flow of network traffic from the target machine to our machine. The rest is accomplished by redirecting the packets in the kernel level. This problem can be solved by the weaknesses of the ARP (address resolution protocol). The ARP is a stateless protocol so it is completely legal by the protocol to send ARP reply packets to the target machine even if it has not send any ARP requests yet.

This makes it possible for the attacker to send forged ARP reply packets continuously to the victim where the MAC address is forged to correspond to the one of the attacker's machines. Usually when you want to sniff the traffic originating from a machine, you need to spoof the gateway of the network.

5.1.1 Analysis of this Attack

The tool required for this attack are already implemented. We only need one tool from this package: arpspoof. Iptables is available on most Linux distributions by default.

The ARP spoof is carried out as follows [Kasslin14] :

- First make sure that the attacking machine has ip-packet forwarding enabled
- On RedHat Linux 8.0 this can be accomplished by executing the command:
  - echo 1 > /proc/sys/net/ipv4/ip_forward
- From the attacking machine run the following command:
  - arpspoof -t [ip address of the victim] [ip-address of the gateway]
- Packet redirection is done with iptables with the following commands:
  - iptables -t nat -A PREROUTING -i eth0 -p tcp -s [victim ip] -d [server ip] --dport [dest_port] / -j REDIRECT --to-port [dest_port_on_attacker_machine]
- Now traffic to the port [dest_port] from the victim to the server is redirected to the attacker's IP address with destination port [dest_port_on_attacker_machine].

The hijacking attack allows the network traffic from the victim to be easily monitored and controlled by the attacker on a switched network. As a result of the ARP spoofing attack all the traffic from the victim can be routed to the client through the attacking machine. This situation allowed us to launch the replay attack on Kerberos 5 and SMB

.

5.1.2 Protecting your environment against this attack

Network connection hijacking can be done in many ways. Here I take the solutions against ARP spoofing for discussion [Kasslin14] . There are two well known ways to detect ARP spoofing attempts monitoring the local ARP cache and monitoring the network traffic on the wire. ARP cache monitoring on a local machine can be accomplished with the arpcommand. This can be done automatically with a tool called arpwatch. Network traffic monitoring can be implemented with certain Intrusion Detection Systems. The Open Source IDS called Snort is able to do this in real time.

One of the best ways to protect machines against ARP spoofing attacks is to enforce static ARP entries on the local machines, especially the entry for the local gateway should be static.

5.2 Password Attack

The Microsoft Windows implementation of Kerberos 5 protocol requires the use of the pre-authentication data in the KRB_AS_REQ message by default, which makes it harder to implement offline password attacks [Kasslin15] . If pre-authentication is not used, anyone can make a request for a TGT from the KDC (Key Distribution Center) and launch an offline password attack against it. The default implementation of pre-authentication data in Windows consists of an encrypted Kerberos timestamp created with a key derived from the user's password and a cryptographic checksum.

If an attacker is able to monitor the network traffic between the victim and the KDC server, a password attack becomes possible. This is based on the fact that before encryption the Kerberos timestamp is an ASCII-encoded string with the syntax "YYYYMMDDHHMMSSZ". This information makes it possible to find a valid password by running a dictionary or brute force attack against the encrypted timestamp. The correctness of the result can be verified by calculating the checksum. The detailed descriptions of the cryptographic operations are provided in [Song16] .

5.2.1 Analysis of this Attack

This attack shows that the pre-authentication scheme based on the symmetrically encrypted timestamp is very vulnerable to the dictionary and brute force attacks [Kasslin15] . It was trivial to gather pre-authentication data between the victim and the KDC server by passively monitoring the network traffic. Dictionary attacks were successfully launched against weak passwords. It can be concluded that the feasibility of this attack depends

mainly on the strength of the used passwords.

To make it easier to perform this attack Kasslin and Tikkanen created two new tools. The first tool [Kasslin17] is a network sniffer which monitors the network traffic in promiscuous mode and collects pre-authentication data from the KRB_AS_REQ messages. The second tool [Kasslin18] performs a dictionary attack against the data collected by the first tool.

5.2.2 Protecting your environment against this attack

This attack is accomplished by passively listening to the network traffic between the victim and the Kerberos KDC server. The only way to detect this is by monitoring the network for symptoms which might show a hint that someone is running a sniffer on the network.

This attack will become infeasible if a strong password policy is implemented. The Windows implementation of Kerberos 5 also supports another pre-authentication method in addition to the password-based. As discussed, PKINIT does not suffer from the weakness described here. Another effective way to prevent this attack is to encrypt the network traffic, for example by using IPSEC.

5.3 Reply Attack

Replay attacks against Kerberos 5 are targeted on the final message transferred from the client to the server, called the KRB_AP_REQ message [Kasslin19] . An attacker will attempt to capture this message and reuse its data to authenticate himself as the victim. If successful, the attacker will have full access to the same service the victim accessed. However, He will not be able to recover the victim's actual password. This attack requires that traffic from the victim to the server is subverted to the attacker's network address. This can be achieved with a hijacking attack described in Section 5.1. In SMB case, there are two questions we need to answer regarding this issue.

- The first question is about the handling of the authenticators by the Windows Server. If a server does not cache used authenticators, replay attacks become much easier, as the attacker only has to passively monitor the network traffic from the victim. If a cache is used, the attacker has to actively prevent the server from seeing the security blob sent by the victim.
- The other question is on the network addresses in tickets. The address included in a service ticket is optional according to [Kasslin20] , but is still highly recommended. This field would limit the use of the ticket to pre-defined hosts, and make replay attacks more difficult, as the attacker would be forced to use the victims IP address when replaying the credentials.

5.3.1 Analysis of this Attack

We can conclude from the results of Kasslin and Tikkanen's research that replay attacks against SMB and Kerberos 5 on a Windows domain are feasible. An attacker will be able to use the victim's credentials to access file shares. Kasslin and Tikkanen's research shows that the Windows Server SP3 does actually cache used authenticators. the attempt to replaying used authenticators failed, because the server refused to accept them. This indicates that an attacker must use an active man-in-the-middle attack to listen on the SMB session setup and prevent the server from seeing the credentials the victim sends. As such, when the attacker replays the security blob, and the server has not seen the authenticator, the attack succeeds. Kasslin and Tikkanen's research shows that the Windows Server SP3 acting as a file server either does not verify the address field or the Windows KDC does not include it in the tickets it issues. This means that an attacker, once he has captured the victim's security blob, may reuse it from his own network address. This makes replay attacks easier.

A tool to perform such an attack, [Kasslin21] is a proxy that listens to connections on the attacker's machine, forwards session negotiations between the real server and the victim and captures the security blob inside the

Session Setup AndX message.

5.3.2 Protecting your environment against this attack

To detect a replay attack, one option would be to attempt detecting ARP spoofing altogether. This is described in more detail in section 6.1. If this is successful, the attack becomes infeasible. The victim can also detect a possible attack if attempted connections seem to fail. When an attack is under way, the victim will see an error message stating that the service is not available. This is because the attacker will stop proxying traffic to the server after capturing the security blob. However, this is not an efficient solution, since such errors are also possible in normal circumstances [Kasslin19] . Also, counting on users in such problems is probably not the best choice. The detection of this attack is very difficult. More effort should be made on preventing it from happening. This is possible in a number of ways, among which, the most efficient is to use some form of encryption on the IP layer. The use of IPSEC would be a sufficient protective action. However, using it to encrypt all client-to-server traffic is very difficult. SMB signing, which is available on some implementations, can be used to prevent replay attacks.

In brief, when signing is enabled, packets will include a cryptographic MD5 checksum created with a session key to ensure their integrity. There is a significant pitfall. Servers usually support SMB signing, but do not require that clients always use it. If the victim is using SMB signing, the connection can still be attacked. The security blob is easily extracted, since no encryption is used. If the attacker is then allowed to connect to the server with the stolen credentials without signing, the attack will succeed.

The server must require SMB signing for all connections for the attack to fail. In this case, the attacker will not know the key to create the checksums, and therefore cannot create a connection. If SMB connections have to be made in an unsafe network, other authentication methods such as NTLMv2 are highly possibly safer than Kerberos [Kasslin19] [Kasslin20] . Replay attacks on such challenge-response mechanisms are not possible, but dictionary attacks on weak passwords surely are.

6. Attacks on Public-Key Kerberos

In this section, I discuss a dangerous attack against PKINIT in public-key encryption mode [Cervesato13] . I start with a detailed description of the attacker's actions in the AS exchange, the key to the attack, followed by an explanation of the conditions required for the attack. Then I close this section with a discussion on how the attacker may propagate the effects of his AS exchange actions throughout the rest of a protocol run.

6.1 How to break Public-Key Kerberos

Figure illustrate "man-in-middle-attack". As a consequence of this attack, C believed to be talking to KAS, is talking to I instead and this causes a failure of authentication problem. Also, regarding a failure of confidentiality, I knows AK and k, then C believes that KAS produced AK and k just for her. Note for this attack is as the following:

1. This is a deterministic attack
2. The attacker(I) must be a legal user, otherwise KAS would not talk to him.
3. C is authenticated to S as I (not as C). The attacker (I) does not trick S to believe he is C. The attacker (I) can observe all communications between C and S and I can also pretend to be S to C
4. Diff-Helman mode appears to avoid this attack; however, it still needs to formally prove security for DH

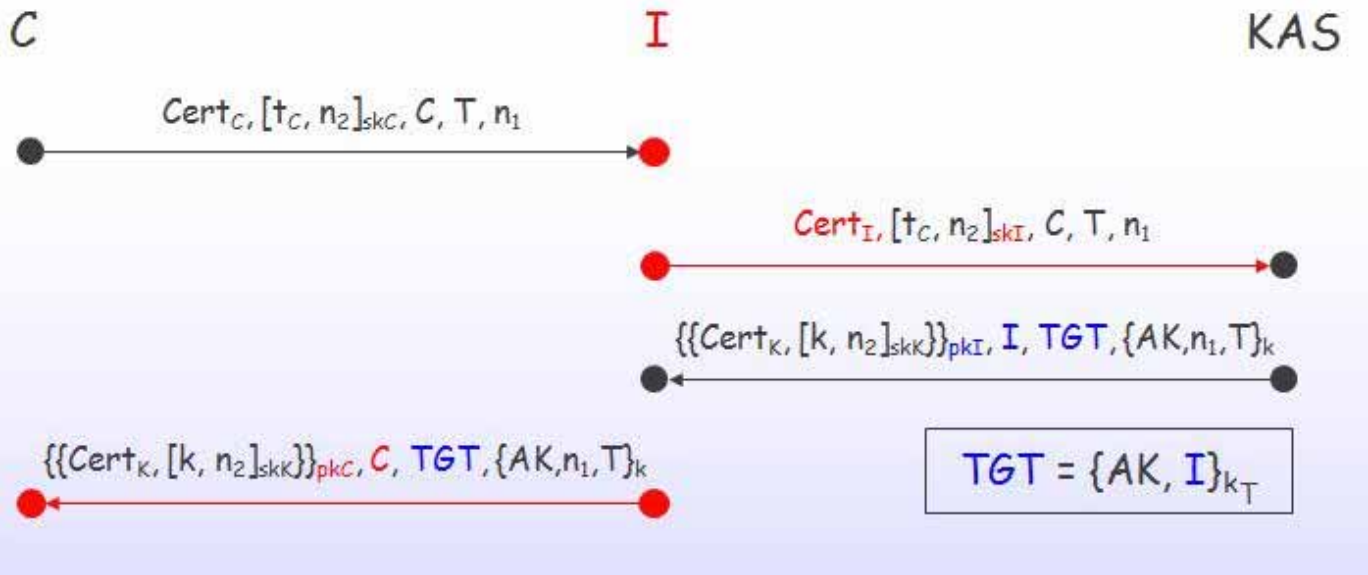

Figure 6: Message flow in the man-in-the-middle attack on PKINIT

6.2 Effects of this attack

Attacker observes traffic and learns keys in replies:

- If the attacker learns AK in the AS exchange, he may either mediate C's interactions with the various servers (essentially logging in as I while leaking data to C so she believes she has logged in) while observing this traffic or simply impersonate the servers in the later exchanges. In the first, which is shown in Figure 6, once C has AK and a TGT, she would normally contact the TGS to get a service ticket for some application server S. This request contains an authenticator of the form {C, tC,T }AK (i.e., C's name and a timestamp, encrypted with AK). Because I knows AK, he may intercept the request and replace the authenticator with one that refers to himself: {I , tI ,T }AK. The reply from the TGS contains a freshly generated key SK; this is encrypted under AK, for C to read and thus accessible to I, and also included in a service ticket ST that is opaque to all but the TGS [Cervesato13]

Attacker impersonates servers:

- It is possible that the attacker can intercept C's requests in the TG and CS exchanges and impersonate the involved servers instead of forwarding altered messages to them; the message flow for this version of the attack is shown in Figure 6. In the TG exchange, I will ignore the TGT and only decrypt the portion of the request encrypted under AK (which he learned during the initial exchange). The attacker will then generate a bogus service ticket XST , which the client expects to be opaque, and a fresh key SK encrypted (along with other data n3, tT , S) under AK, and send these to C in what appears to be a properly formatted reply from the TGS. In the CS exchange the attacker may again intercept the client's request; in this case, no new keys need to be generated, and the attacker only needs to return the client's timestamp encrypted under SK which I himself generated in the previous exchange for C to believe that she has completed this exchange with the application server S. Note that the attacker may take the first approach mediating the exchange between C and a TGS in the TG exchange and then the second impersonating the application server in the CS exchange. The reverse is not possible because I cannot forge a valid ST for S when impersonating T.
- Despite which approach the attacker uses to propagate the attack throughout the protocol run, C finishes the CS exchange believing that she has interacted with server S and that T has generated a fresh key SK

known only to C and S. But in fact, I knows SK in addition to, or instead of, S (depending on how I propagated the attack). Thus I may learn any data that C attempts to send to S; depending on the type of server involved, such data could be very sensitive. Note that this attack does not enable I to impersonate C to a TGS or an application server because all involved tickets name I; Section 6.2 discusses a related authentication property. This also means that if C is in communication with an actual server (T or S), that server will view the client as I , not C.

6.3 Detecting and preventing this attack

What's wrong with PKINIT-26:

- As shown in Figure 7, there is misbinding of request and reply. Hence, the attacker can both tamper with signiture in request and with encryption in reply [Cervesato13] . Having traced the origin of the discovered attack to the fact that the client cannot verify that the received credentials (the TGT and the key AK) were generated for her, the problem can be fixed by having the KAS include the client's name, C, in the reply, in such a way that it cannot be modified enroute and that the client can check it.

Figure 7: Message in the man-in-the-middle attack on PKINIT-26

Solution adopted in PKINIT version 27:

- Currently, the solution adopted in PKINIT-27 and current candidates for H include hmac-sha1-96-aes128. New strong keyed checksums can be used for ck as they are developed. The checksum-based approach was later included in PKINIT-27 [Cervesato13] . The message flow of this version of PKINIT is displayed in Figure 8. Here, ck is a checksum of the client's request keyed with the key k, that is ck has the form Hk(CertC, [tC, n2]skC ,C, T, n1) where H is a preimage-resistant MAC function. This means that it is infeasible for the attacker to find a message whose checksum matches that of a given message.

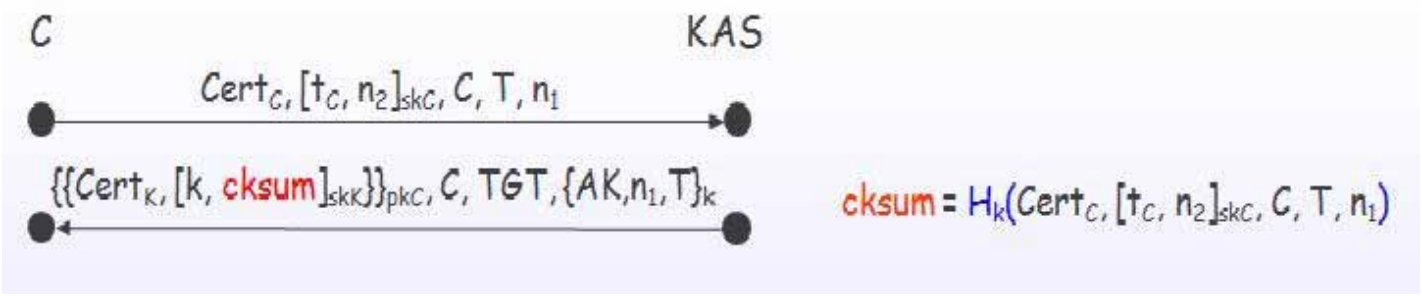

Figure 8: Fix Adopted by PKINIT-27

7. Improving Kerberos for Cross-Realm Collaborative Interactions

As discussed in Section 3, the Kerberos has some issues related to the cross-realm operations. These issues are related to the inter-realm trust management and client centricity of the proto col exchanges. The inter-realm trust in the current specification is managed through shared secret keys. This method is not scalable when the number of realms increases. The client centricity of the Kerberos protocol exchanges on the other hand puts all the load of cross-realm operation on the client side. For devices such as PDAs and sensors, this processing could result in unacceptable delays. In order to solve these issues, recently, the Extensible Key Distribution Center Protocol(XKDCP) is new proposed for the cross-realm operations in Kerberos [Zrelli23]

7.1 XKDCP

Once again, there are some frameworks of public-key based trust relationship (e.g., RADIUS or EAP) is well established and widely adopted nowadays. It is viewed as the most scalable and safe way of conveying autgebtucation. However, public-key based authentication system is considerably expensive and is not suitable for devices with low computational power. As one of ths feasible solutions, the XKDCP protocol consists of two sub-protocols; The Inter Authentication Service Protocol(XASP) [Zrelli22] and the Inter Tiket Granting Servive Protocol (XTGSP) [Zrelli22] . Either of the protocols has its own use that will be explained in the following sections. Briefly, The XTGSP protocol can be used In remote access scenarios to allow the local KDC to deliver credentials for services located in remote realms. On the other hand, the XASP protocol can be used in case of cross-realm roaming scenarios to allow the visited KDC to deliver credentials for roaming users.

7.1.1 XASP

Acquiring TGT in visited realm:

- XASP is designed by S Zrelli et al. As an extension of the Kerberos protocol, which enables two Kerberos KDCs to collaborate in order to authenticate a roaming user and to deliver a TGT that can be used in the visited realm to obtain service tickets for accessing application services.
- Based on the assumption that the home KDC and the KDC of the visited realm both support the XASP extension, the client acts as if the visited KDC is capable of delivering a TGT, even though he/she is aware that he/she is not registered as principal in the visited realm. When the visited KDC supports the XASP extention, it is capable of processing AS-REQ (Authentication Server Request Message) requests from users belonging to any realm. The processing of these requests is specified by the XASP extension.
- The roaming client starts with contacting the KDC of the visited realm by issuing an AS-REQ message requesting a TGT for use in the visited realm. The AS-REQ message contains information of the user and her home realm. The KDC of the visited realm locates the KDC of the roaming user and issues an XASP-REQ message. The XASP-REQ message is built upon the users's AS-REQ message; in addition, a signature that verifies the identity of the visited KDC is added. The home KDC, which received the XASP-REQ message, firstly authenticates the message by verfying the signature. Then the home KDC creates a TGS session key that it will encrpy using the user's secret key. A copy of the same TGS session key is encrpyted using the public key of the visited KDC. The home KDC sends these two encrypted components to the visited KDC in an XASP-REP message. The XASP-REP message means a signature that authenticates the home KDC to the visited KDC. When the visited KDC receives the XASP-REP message, it validates it by verifying the attached signature. If the authentication phase succeeds, the visited KDC decrpyts the TGS session key using its own private key. The TGS session key is used to build a TGT for the roaming users. The TGT and the TGS session key encrypted using the user's secret key are sent to the user in an AS-REP message.
- As soon as the user receives the AS-REP message, he/she can decpyt the TGS session key and use the TGT to request service tickets from the visited KDC through TGS exchanges.

7.1.2 XTGSP

Acquiring tickets for remote services:

- XTGSP [Zrelli22] allows users to retrieve ST from a KDC although the service is not registered in that KDC. The typical use of XTGSP is in remote access scenarios where a user has a TGT for a local KDC and wants to access services deployed in a remote realm. The advantage of using XTGSP protocol lies in that the client does not need to have a cross-realm TGT for the target realm deploying the service.
- Furthermore, the client does not need to contact the remote KDC since the local KDC will deliver the service ticket that can be used directly to authenticate with the remote service. In fact, from the client's point of view, the local KDC delivers the service ticket as if the remote service was registered in the local realm. The cross-realm operations are managed by the local KDC and made transparent to the client. As a result of the XTGSP exchange, the local KDC acquires enough materials to be able to deliver the requested service ticket to the client.
- Based on the XTGSP protocol, for delivering a service ticket to a client. After obtaining a TGT for the local realm:
    1. Firstly, the client issues a TGS-REQ message asking for credentials (Ticket and the associated session key) to access a certain remote service.
    2. After validating the request, the local KDC locates the remote realm and the associated remote KDC.
    3. Then, it issues an XTGSP-REQ message. The XTGSP-REQ is generated upon the user's TGS-REQ message, and additionally, includes a signed payload that authenticates the local KDC to the remote KDC.
    4. The remote KDC authenticates the XTGSP-REQ message by verifying the public-key signature, then issues a ticket and an associated session key. The session key is encrypted using the public key of the local KDC.
    5. The encrypted session key and the Ticket are then sent to the local KDC in an XTGSP-REP message. The XTGSP-REP message is signed using the public key of the remote KDC.
    6. When the local KDC receives the XTGSP-REP message, it authenticates the message by verifying the signature. It then decrypts the session key.
    7. Finally, the local KDC encrypts the session key using the TGS session key shared with the client and send the result along with the Ticket to the client in a TGS-REP message. Once the client has received the TGS-REP message, she can authenticate with the remote service through an AP exchange.

8 Summary

In summary, I reviewed and analyzed the structure of Kerberos recently proposed and the cross-realm authentication model of Kerberos as well as an extension version of Kerberos, PKINIT, modifies the basic protocol to allow public-key authentication. As discussed, even though Kerberos has been proven its strengths so far, we could spot several security weakness on Kerberos V and PKINIT. Overall, we look at the recent discovery of attacks against Kerveros V and PKINIT. Regarding attacks on Kerveros V, we discuss about Hijacking a Network Connection on a Switched Network, password attack and reply attack. Particularly for PKINIT, we concentrate on man-in-middle attack and currently, the solution adopted in PKINIT-27 and current candidates for H include hmac-sha1-96-aes128. Lastly, based on the recent published papers, I introduce several feasible solution to prevnet other possible attacks and way to protect your envionment.

9 References

[Butler01] F. Butler, I. Cervesato, A. D. Jaggard, A. Scedrov, C. Walstad, Formal Analysis of Kerberos 5, Theoretical Computer Science 367 (1-2), 2006.

http://eprints.kfupm.edu.sa/17429/

[Butler02] M. Backes, I. Cervesato, A. D. Jaggard, A. Scedrov, J. K. Tsay, Cryptographically Sound Security Proofs for Basic and Public-key Kerberos, ESORICS��06, 2006.
http://www.springerlink.com/index/432558h2046mh722.pdf

[Tsay03] JK Tsay, Formal analysis of the Kerberos authentication protocol, Univ. of Pennsylvania Electronic Dissertations, 2008.
http://repository.upenn.edu/dissertations/AAI3328667

[IETFSeq04] JIETF, Public Key Cryptography for Initial Authentication in Kerberos, RFC 4556. Preliminary versions available as a sequence of Internet Drafts, 1996-2006
http://tools.ietf.org/wg/krb-wg/draft-ietf-cat-kerberos-pk-init/

[Zhu05] L. Zhu, B. Tung, IETF RFC 4556, Public Key Cryptography for Initial Authentication in Kerberos, June, 2006,
http://www.ietf.org/rfc/rfc4556.txt

[Neuman06] C. Neuman, T. Yu, S. Hartman, K. Raeburn, IETF RFC4120, The Kerberos Network Authentication Service (V5), July 2005,
http://www.ietf.org/rfc/rfc4120.txt

[Bella07] G. Bella, Inductive Verification of Cryptographic Protocols, Ph.D. thesis, University of Cambridge, March 2000,
http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.14.2215&rep=rep1&type=pdf

[Mitcell08] J. C. Mitchell, M. Mitchell, and U. Stern, Automated Analysis of Cryptographic Protocols Using Mur, Proc. of the IEEE Symposium on Security and Privacy, IEEE Computer Society Press, 1997, pp. 141-153,
http://ieeexplore.ieee.org/xpls/abs_all.jsp?arnumber=601329

[Cervesato09] I. Cervesato, A. D. Jaggard, A. Scedrov, and C. Walstad, Specifying kerberos 5 cross realm authentication, pp. 12-26, 10-11 January 2005.
http://portal.acm.org/citation.cfm?id=1045405.1045408

[Clercq10] J. De Clercq, M. Balladelli, Windows 2000 authentication, digital Press, 2001
http://www.windowsitlibrary.com/Content/617/06/6

[Goldwasser11] S. Goldwasser, S. Micali, R. L. Rivest, A Digital Signature Scheme Secure Against Adaptive Chosen Message Attacks, SIAM J. Computing 17 (1988) 281-308.
http://eprints.kfupm.edu.sa/17429/

[CTL12] Cable Television Laboratories, Inc., PacketCable Security Specification, technical document PKT-SP-SEC-I11-040730 (2004).
http://www.cablelabs.com/

[Cervesato13] Iliano Cervesato, "Breaking and Fixing Public Key Kerberos", 2007.
http://linkinghub.elsevier.com/retrieve/pii/S089054010700123X/

[Kasslin14] K. Kasslin, A. Tikkanen. Hijacking a Network Connection on a Switched Network, March 2003.
http://users.tkk.fi/autikkan/kerberos/docs/phase1/pdf/LATEST_hijacking_attack.pdf

[Kasslin15] K. Kasslin, A. Tikkanen. Password Attack on Kerberos V and Windows 2000, March 2003.
http://users.tkk.fi/autikkan/kerberos/docs/phase1/pdf/LATEST_password_attack.pdf

[Song16] D. Song. Toolset dsniff.
http://naughty.monkey.org/~dugsong/dsniff/

[Kasslin17] K. Kasslin, A. Tikkanen. sniff_krb5_asreq_packet.c

[Kasslin18] K. Kasslin, A. Tikkanen. crack_krb5_preauth_data.c.

[Kasslin19] K. Kasslin, A. Tikkanen. Replay Attack on Kerberos V and SMB, March 2003.
http://users.tkk.fi/autikkan/kerberos/docs/phase1/pdf/LATEST_replay_attack.pdf

[Kasslin20] K. Kasslin, A. Tikkanen. Replay Attack against Kerberos V and LDAPv3, April 2003.
http://users.tkk.fi/autikkan/kerberos/docs/phase1/pdf/LATEST_ldap_replay.pdf

[Kasslin21] K. Kasslin, A. Tikkanen. smb_catchblob.c

[Zrelli22] S. Zrelli, Y Shinoda, S. Sakane, K. Kamada, and M. Ishiyama, Xkdcp, the inter kdc protocol for cross realm operations in kerberos, IETF, Internet Draft, July 2006
http://tools.ietf.org/html/?draft=draft-zrelli-krb-xkdcp-00

[Zrelli23] S. Zrelli, Tunc Medeni and Yoichi Shinoda, Improving Kerberos Security System for Cross Realm Collaborative Interactions:An Innovative Example of Knowledge Technology for Evolving & Verifiable E Society, 2007
http://ieeexplore.ieee.org/xpls/abs_all.jsp?arnumber=4223076

List of Acronyms

AS	Authentication Server
CS	Client-Server
CSE	Computer Science and Engineering
EAP	Extensible Authentication Protocol
IETF	Internet Engineering Task Force
KAS	Kerberos Authentication Server
KDC	Key Distribution Center
LDAPv3	Lightweight Directory Access Protocol
MIT	Massachusetts Institute of Technology
NTLMv2	NT LAN Manager Version 2
PKI	Public Key Infrastructure
PKINIT	Public Key Cryptography for Initial Authentication in Kerberos,
RADIUS	Remote Authentication Dial In User Service
RFC	Request For Comments
SMB	Server Message Block
ST	Service Ticket
TGT	Ticket granting ticket
TGS	Ticket granting Server
XASP	Inter Authentication Service Protocol
XKDCP	Extensible Key Distribution Center Protocol
XTGST	Inter Ticket Granting Service Protocol

Last modified: April 20, 2009
This and other papers on latest advances in network security are available on line at http://www.cse.wustl.edu/~jain/cse571-09/index.html


Back to Raj Jain's Home Page