IT-Security 1

Chapter 8: Kerberos

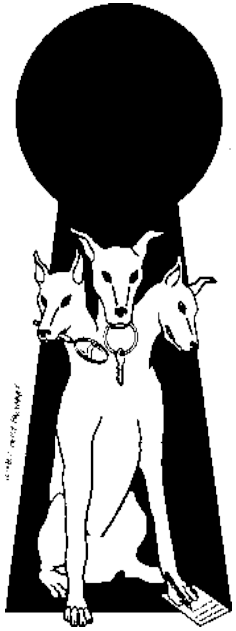
Prof. Dr.-Ing. Ulrike Meyer

WS 15/16

Kerberos

- Protocol family that provide authentication and key agreement in a network
  - Enables network applications to authenticate their peers
  - Based on shared secret keys
- Originally designed by the MIT
- Based on the Needham and Schroeder protocol
- Used e.g. in
  - Windows 2000 and later use Kerberos as default authentication method
  - Apple's Mac OS X in client and server versions
  - Red Hat Enterprise Linux 4 and later in both client and server versions
- Currently Versions 4 and 5 in use
  - We will first study Kerberos v4 and then Kerberos v5

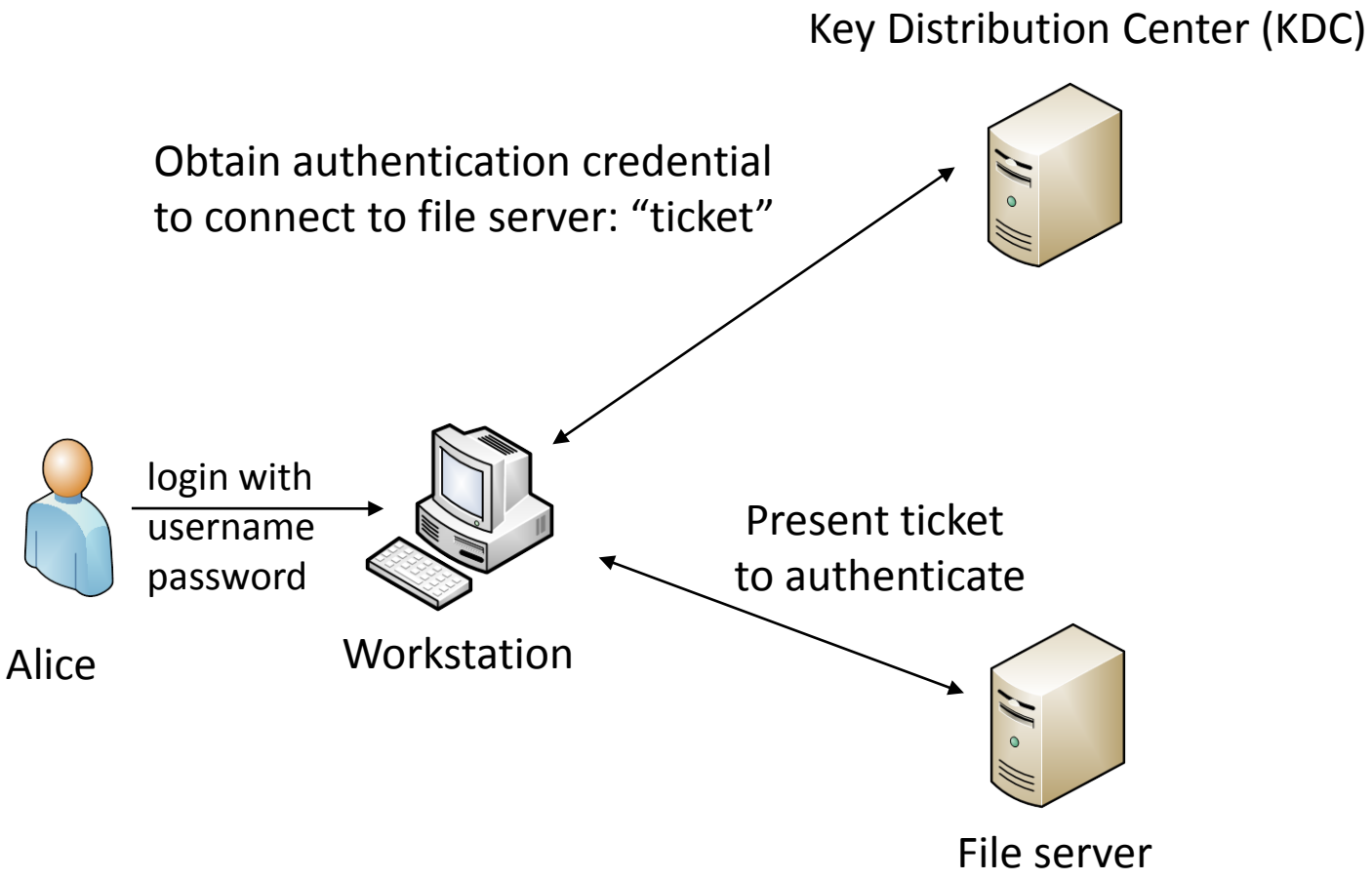
The Name



- Kerberos is the name of the (three headed) dog that protects the entrance to Hades in Greek mythology
- There is no agreement about the number of heads though ;)

Kerberos v4

Kerberos Overview

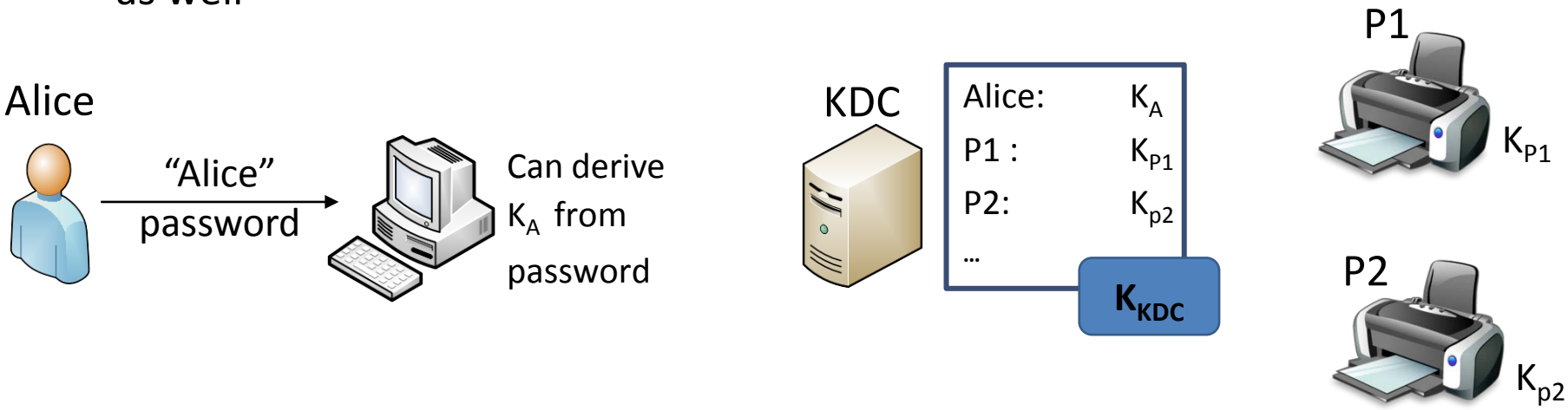


Kerberos Entities

- Kerberos Key Distribution Center (KDC) consists of
  - Kerberos Authentication Server (AS)
  - Kerberos Ticket Granting Server (TGS)
- KDC supplies tickets and session keys
- Realm
  - Kerberos Administrative Domain, represents a group of principals
  - A single KDC may be responsible for one or more realms
- Principal
  - User or service installed on a particular host
  - Principal Identifier: Unique identifier for a principal
    - {service}{user}:host@realm
    - “realm” is typically chosen to be the DNS name

Long-term Keys

- KDC has a secret key $\mathbf{K_{KDC}}$ known only to KDC
- KDC shares a secret master key $\mathbf{K_R}$ with each resource R
- KDC shares a secret master key $\mathbf{K_A}$ with each user Alice
  - The master secrets of users are derived from a user's password
- KDC keeps these keys in a database encrypted with the KDC's master key
- Kerberos 4 uses DES as encryption algorithm Kerberos 5 supports e.g. AES as well

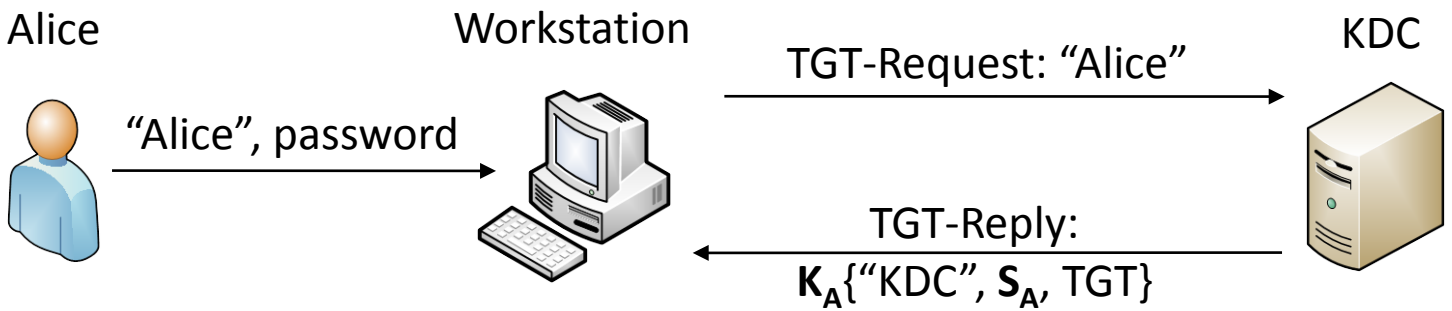


Notation

- Note: in the following K{X} means X is encrypted and integrity protected with some symmetric encryption algorithm using K as the key

Ticket Granting Tickets (1)

- Alice logs on, the workstation asks the KDC for a session key for Alice
- KDC
  - Generates a session key $\mathbf{S_A}$ for Alice
  - Generates a "ticket granting ticket" TGT = $\mathbf{K_{KDC}}${"Alice", $\mathbf{S_A}$, lifetime}
  - Encrypts the TGT with its own master key $\mathbf{K_{KDC}}$
  - Encrypts its identifier, $\mathbf{S_A}$ and the TGT with Alice's master key $\mathbf{K_A}$ and sends it to the workstation
- Workstation uses Alice's password to derive Alice's master key and decrypts the session key $\mathbf{S_A}$

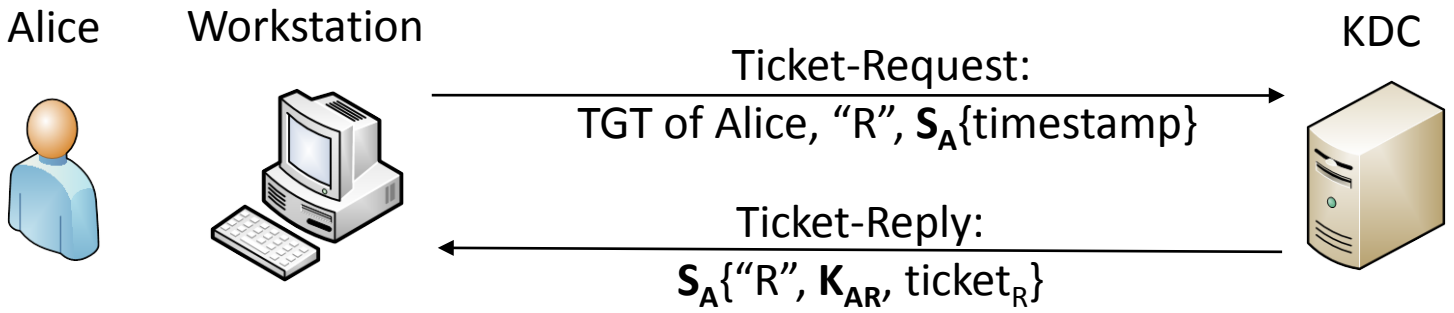


Ticket Granting Tickets (2)

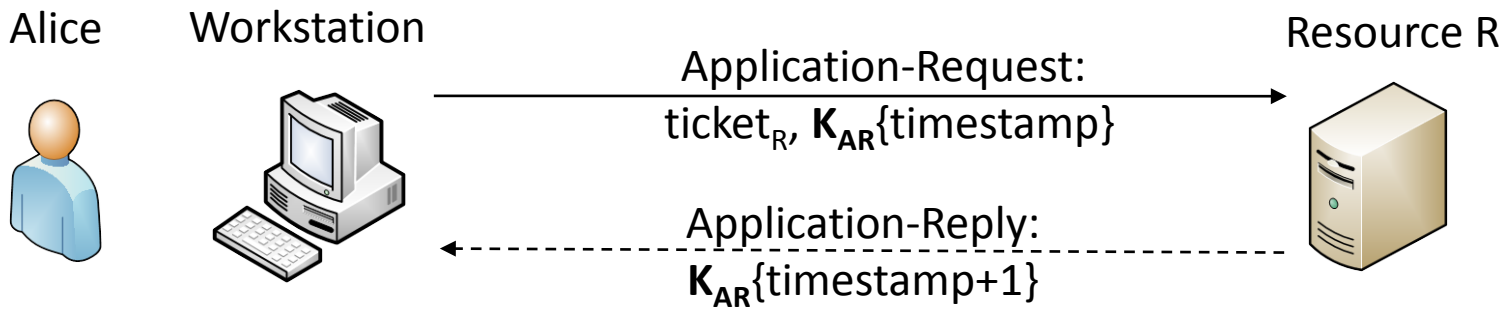
- The ticket granting ticket is used later on to provide $\mathbf{S_A}$ to the KDC such that KDC does not have to store any stateful information
- KDC then uses $\mathbf{S_A}$ to encrypt keying material when issuing tickets to the workstation for other applications like file servers, print servers, …
- The use of a session key instead of the master key minimizes the use of a key that is derived from the user's password
  - Makes password attacks more difficult

Tickets

- When Alice wants to access a resource R, then she sends a "ticket request" to the KDC
- KDC decrypts the TGT to obtain $\mathbf{S_A}$, decrypts and checks the timestamp
- KDC then issues a "ticket" for R to Alice by
  - Generating a session key $\mathbf{K_{AR}}$
  - Generating a "ticket" $ticket_R = \mathbf{K_R}\{\text{"Alice"}, \mathbf{K_{AR}}, lifetime\}$ encrypted with $\mathbf{K_R}$
- Sends the "ticket reply" back to Alice's workstation
  - Containing R's identifier, $\mathbf{K_{AR}}$ and $ticket_R$, all encrypted with $\mathbf{S_A}$
- Alice can now present the encrypted ticket to R and R will be able to decrypt the ticket with its master key



Login into the Resource R

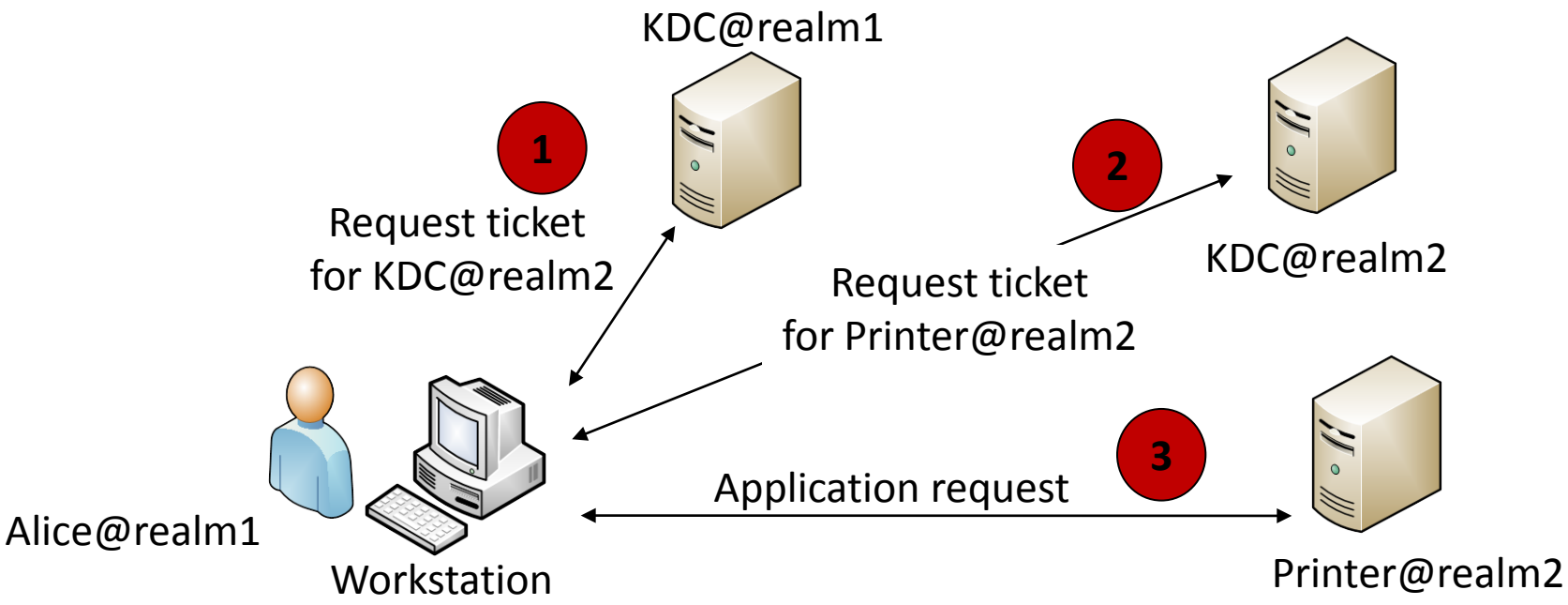


- To authenticate Alice to R, the workstation sends an application request including the ticket and a timestamp encrypted with the key K_{AR} from the ticket
- R decrypts the ticket to obtain K_{AR} and uses it to check the timestamp to authenticate Alice
- R sends back the application-reply including the timestamp incremented by one and encrypted with K_{AR} to proof that he was able to decrypt the ticket, i.e. he is indeed R

Replicated KDCs

- Single KDC is a single point of failure and a performance bottleneck
- Multiple KDCs that store the same master KDC key are desirable
- Kerberos supports having
  - A single master copy of the database that stores the principle identifiers and master keys
  - Updates (adding, deletion, modification of entries) are made to that master copy only
  - Other KDC sites download the master copy periodically
- Still single point of failure for updates but not e.g. for application requests
  - These can be handled by slave KDCs even if master KDC is down
- When copying the data base to the slave, transfer should be integrity and replay protected to protect against modification and replay

Realms



- Kerberos supports access to resources in realm2 by users in another realm1
  - The KDC of realm2 then acts as a resource in realm1

Identifiers

- Identifiers used in Kerberos 4 have the form
  - Name, instance, realm
- The realm component specifies the administrative domain
- The instance component specifies the type entity e.g. “printer”
- The name is the actual name of a particular instance in a particular realm

Key Version Numbers

- Passwords and master keys have to be exchanged from time to time
  - Due to forgotten passwords, key compromises or preventive measures
- If e.g. R's master key is changed while Alice holds a ticket for resource R, then this may cause problems
- Kerberos therefore uses key version numbers to indicate the key used in requests and replies
- Each resource keeps previous keys for a pre-defined time before completely deleting them

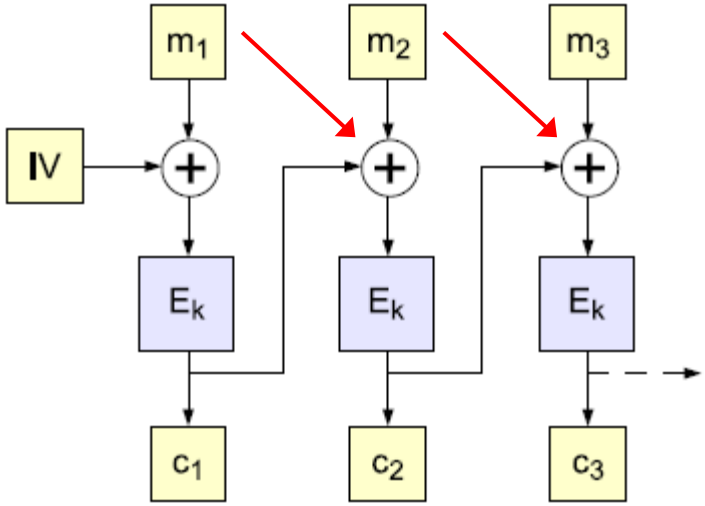
Combined Encryption and Integrity in Kerberos v4

- Kerberos 4 uses DES in PCBC mode to provide encryption and integrity protection after authentication

- Plaintext-Cipher-Block-Chaining works as follows
  - $c_0 := IV$, $m_0 = 0$, $c_{n+1} := E_K(m_{n+1} \oplus c_n \oplus m_n)$
  - $m_{n+1} = D_K(c_{n+1}) \oplus c_n \oplus m_n = D_K(c_{n+1}) \oplus c_n \oplus D_K(c_n) \oplus c_{n-1} \oplus m_{n-1}$
    $= D_K(c_{n+1}) \oplus c_n \oplus D_K(c_n) \oplus c_{n-1} \oplus \cdots \oplus D_K(c_1) \oplus c_0$

- Idea: put some recognizable plaintext at the end of each plaintext, if this decrypts correctly, then no modification took place
- In addition, Kerberos 4 supports no protection and integrity only:
  - Integrity-only is based on a specifically designed algorithm
    - Uses the session key and the message as input

A word on Plaintext-Cipher-Block-Chaining



Additional operations done in PCBC compared to CBC

- Difference to CBC:
  - CBC: modification of c_i will garble only m_i and m_{i+1}
  - PCBC:modification of c_i will garble all following: m_i, m_{i+1}, ..., m_n
- Idea: in PCBC m_n, decrypts properly only if the message was not changed
  - m_n can be used to carry a checksum over the message
- Unfortunately PCBC does not really provide integrity: e.g. if attacker swaps c_i and c_j for $i \neq j$, $i,j < n$, then m_n still decrypts correctly

Network Layer Addresses in the Tickets

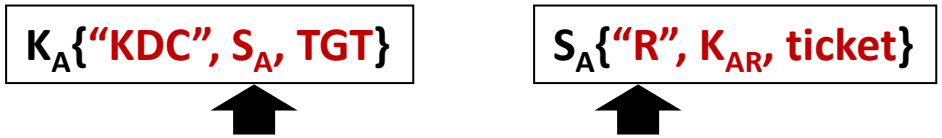
- In addition to the identifiers, tickets include the network layer address (IP address) of the requester
- When issuing tickets KDC checks if Alice's IP address in the TGT equals the one the request comes from
- If Alice uses a ticket to authenticate to some resource R, R checks if the application request originates from the IP address in the ticket
- The RFC argues that this makes it a little harder to replay tickets to resource R or TGTs to KDC as an attacker has to spoof the IP address

Message Formats

- We will now describe the message formats for
  - TGT-request
  - TGT-reply
  - Ticket-request
  - Ticket-reply
  - Application-request
  - The optional application-reply
- Note that the formats for the messages TGT-reply and ticket-reply are the same
- Note that the formats for ticket-requests and application-requests are the same
- I.e. Kerberos in fact only defines 4 message formats

Common Fields in Message Formats

- Some fields are used in several different messages
- These are
  - Credential



    - Encrypted part of **TGT-replies** and **ticket-replies**
    - If used in a TGT-reply: encrypted with Alice's master key $\mathbf{K_A}$
    - If used in a ticket-reply: encrypted with Alice's session key $\mathbf{S_A}$
  - Ticket
    - Included in **TGT-replies** and **ticket-replies**
    - Includes the resource's name and instance in case of a ticket-reply and KDC's name and instance in case of a TGT-reply
  - Authenticators
    - Included in **ticket-requests** and **application-requests** to carry the time stamp

Credentials

# octets	
8	Session key S_A or K_{AB}
40	KDC's or R's name
40	KDC's or R's instance
40	KDC's or R's realm
1	Ticket lifetime
1	KDC's or R's key version number used for ticket
1	Length of ticket
variable	ticket
4	Timestamp in seconds when ticket was created

Explanation

- A credential is included in each **TGT-reply** or **ticket-reply** message from the KDC and carries the TGT or ticket
- A credential carrying a TGT is encrypted with Alice's master key $\mathbf{K_A}$
- A credential carrying another ticket is encrypted with Alice's session key $\mathbf{S_A}$

Ticket-Granting Tickets and Tickets

# octets	
40	Alice's name
40	Alice's instance
40	Alice's realm
4	Alice's network layer address
8	Session key S_A or K_{AB}
1	Ticket lifetime in 5 minute units
4	KDC's timestamp when ticket made
40	KDC's or the resource's name
40	KDC's or the resource's instance

Explanation

- In case of a ticket granting ticket, the session key $\mathbf{S_A}$ is included, in case of a ticket for resource R, $\mathbf{K_{AR}}$ is included
- The lifetime of the ticket is the minimum of the requested lifetime and the lifetime the KDC is willing to permit for a given request
- In case of a ticket granting ticket, KDC's name and instance is included in the ticket, otherwise the resource's name and instance are included

Authenticators

- **Encrypted with S_A or K_{AB}**

# octets	
40	Alice's name
40	Alice's instance
40	Alice's realm
4	Checksum to detect errors
1	Rest of Alice's timestamp in 5 ms units
4	Alices's timestamp in seconds

Explanation

- Is encrypted with $\mathbf{S_A}$ if used in a **ticket-request**
- Is encrypted with $\mathbf{K_{AR}}$ if used in an **application-request**
- Proofs that Alice is in possession of the respective key by the inclusion of Alice's time stamp
- Guarantees that **ticket-requests**, **application-requests**, **application-replies** cannot be replayed

TGT-Request

# octets	
1	Version of Kerberos (4)
1	Message type (1)
40	Alice’s name
40	Alice’s instance
40	Alice’s realm
4	Alice’s workstation’s timestamp in seconds
1	Desired TGT lifetime in 5 minutes units
40	KDC’s name
40	KDC’s instance

TGT-Reply

# octets	
1	Version of Kerberos (4)
1	Message type (2)
40	Alice’s name
40	Alice’s instance
40	Alice’s realm
4	Alice’s workstation’s timestamp (from TGT request)
1	Number of tickets (1)
1	Alice’s key version number
2	Credential length
variable	Credential encrypted with Alice’s master key

Ticket Request

# octets	
1	Version of Kerberos (4)
1	Message type (3)
1	KDC's key version number
40	KDC's realm
1	Length of TGT
1	Length of authenticator
variable	TGT
variable	authenticator
4	Alice's timestamp
1	Desired Ticket Lifetime
40	resource's name
40	resource's instance

Ticket Reply

- Same format as TGT-Reply but credential encrypted with Alice's session key
- Credential includes a ticket for resource "R" instead of the TGT

Application Request

# octets	
1	Version of Kerberos (4)
1	Message type (8)
1	resource's key version number
40	resource's realm
1	Length of ticket
1	Length of authenticator
variable	ticket
variable	authenticator
variable	application data*

*e.g. in case of a ticket-request the application data consists of the desired lifetime of the ticket, and resource's name and instance (see above)

Application Reply

- Not mandatory (only used if resource is to authenticate to Alice as well)
- “Encrypted stuff” includes the timestamp+1 from the request

# octets	
1	Version of Kerberos (4)
1	Message type (6)
1	Length of encrypted stuff
variable	Encrypted stuff

Protection of Application Data

- In addition to the authentication messages, Kerberos defines two message types to protect application data
- The safe-message type integrity protects the application data carried within the message
- The private-message type encrypts and integrity protects the application data carried within the message with DES in PCBC mode
- Both message types use the session key $\mathbf{K_{AR}}$ for integrity and encryption respectively

Conceptual Limitations of Version 4

- No choice of encryption and integrity algorithms
  - Only supports the use of DES in PCBC mode
- Requires the use of IP addresses and is therefore usable only on top of IP
- Message byte ordering indicated by sender instead of being fixed
- Maximum lifetime of a ticket in Kerberos 4 is 21:20 h
  - Not enough for some applications
- No support for delegation
  - i.e. tickets cannot be delegated to other IP addresses

Conceptual Limitations of Version 4

- Problems with principle naming
  - Name, instance, realm can each be 39 characters with a terminating 0, use of "." excluded in the name part
- Inter-realm authentication may not scale as each KDC in a realm has to share a secret key with each KDC in an interoperating realm
- Double encryption: tickets are encrypted twice when they are provided to a client
  - With the client's (Alice's) key
  - With the application server's (resource's) key

Technical Limitations of Version 4

- PCBC mode: non-standard DES mode that does not provide integrity protection
  - Intruders can modify messages
- Authenticators and replay detection: timestamps require a window of acceptable timestamps. Timestamps already accepted from that window have to be recorded.
- Password attacks: the ticket granting ticket in the TGT-Reply is encrypted with the master secret of the user and includes verifiable plaintext ("KDC"). It therefore allows for password guessing attacks

TGT-Reply: $\mathbf{K}_A\{\text{"KDC"}, \mathbf{S}_A, \text{TGT}\}$

- Integrity protection used in the safe-message format is of questionable quality

Kerberos v5

Kerberos 5

- Original specification: RFC 1510
  - Referred to as specification 1 in the following
- Latest specification: RFC 4120 (2005)
  - Referred to as specification 2 in the following
- The following concepts are reused in Kerberos 5
  - Each principal shares a master key with the KDC
  - Ticket-granting tickets, TGT-requests, TGT-replies
  - Tickets, Ticket-requests and replies
  - Application-requests and replies
  - Basic message contents is the same as in Kerberos 4

Improvements in Kerberos 5 (1)

- Support of different encryption algorithms in authentication messages and private-message type
  - Kerberos implementations can specify their own encryption algorithms as well
  - Specified as part of Kerberos 5 specification 1:
    - DES-CBC-CRC
    - DES-CBC-MD4
    - DES-CBC-MD5
  - All three aim to provide integrity and encryption
  - Basic idea of all:
    - Compute a hash with CRC, MD4 or MD5
    - Append it to the message to protect
    - Encrypt the message with DES in CBC mode

Improvements in Version 5 (2)

- Supports different integrity algorithms in the safe-message type, namely (specification 1)
  - RSA-MD5-DES: Encrypts MD5 hash of message with DES in CBC mode
  - DES-MAC: DES CBC MAC construction
  - DES-MAC-k
    - Same as DES-MAC but using same key twice
    - Not recommended any more
  - RSA-MD4-DES
    - same as RSA-MD5-DES but with MD4
  - RSA-MD4-DES-k
    - Same as RSA-MD5-DES but with MD4 and same key used twice
    - Not recommended any more
  - NOTE: has nothing to do with RSA encryption, but MD4 and MD5 are algorithms designed by the RSA company

New in Specification 2

- Since specification 2 of Kerberos v5 the following encryption mechanisms are
  - Mandatory: AES256-CTS-HMAC-SHA1-96 (AES in CBC mode with ciphertext stealing)
  - Optional: AES128-CTS-HMAC-SHA1-96, DES-CBC-MD5, DES3-CBC-SHA1-KD
- Since specification 2 of Kerberos 5 the following integrity protection mechanism are
  - Mandatory: HMAC-SHA1-96-AES256
  - Optional: DES-MD5, HMAC-SHA1-DES3-KD, HMAC-SHA1-96-AES128
- In the update RFC 6649 the use of the DES-based ciphers and integrity protection mechanisms were deprecated

Improvements in Version 5 (3)

- Support of several address types, not only IP
  - Network addresses in Kerberos 5 messages are tagged with a type and a length field such that the use of other addresses than IP addresses is enabled
- Principal naming: name and realm part only
  - No restrictions on length of name and characters to use in names
  - Realm names can be DNS names or X.500 names
  - Support of hierarchical realm names

Improvements in Kerberos 5 (4)

- Encoding problems (byte ordering) solved by the use of standard encodings (ANS.1 syntax with the Basic Encoding Rules)
- Changes to tickets and TGTs:
  - KDC's name no longer encrypted in TGT-Reply
  - No double encryption of TGTs with KDC's and Alice's key anymore
  - No double encryption of tickets with resource's and Alice's key anymore

Kerberos 4	Kerberos 5
TGT-Reply: $\mathbf{K_A}${"KDC", $\mathbf{S_A}$, TGT}	TGT-Reply: $\mathbf{K_A}${$\mathbf{S_A}$}, TGT
Ticket-Reply: $\mathbf{S_A}${"R", $\mathbf{K_{AR}}$, $ticket_R$}	Ticket-Reply: $\mathbf{S_A}${"R", $\mathbf{K_{AR}}$}, $ticket_R$}

New Protocol Features

- Support of alternate or additional authentication mechanisms with the help of the pre-authentication data field in TGT-requests
  - E.g. based on one-time passwords generated by tokens
- Sequence number support: Kerberos 5 supports the use of sequence numbers as alternative to timestamps

Delegation of Rights

- Support for delegation of rights (optional):
  - Allows Alice to request a TGT or ticket bound to the network address of Peter and provide that TGT or ticket to Peter
  - Peter can subsequently use the TGT to obtain tickets on behalf of Alice or use an application specified in the ticket on behalf of Alice
  - Alice can restrict the rights of Peter by requesting the inclusion of authorization data in the TGT or ticket
  - Flags in the TGT indicate if Alice has the right to request a TGT or a ticket for Peter

Kerberos V5 Ticket Flags for Delegation

- Proxiable Ticket Flag
  - Set in a TGT that Alice can use to request tickets for another network address than her own
  - A ticket requested with a proxiable TGT then has the proxy flag set
- Forwardable Ticket Flag
  - Set in a TGT that allows Alice to request a TGT for another network address than her own
  - A TGT requested with a forwardable TGT then has the forwarded flag set
  - A ticket requested with a forwarded TGT has the forwarded and the proxy flag set
- The purpose of the flagging is to allow applications to decide whether or not to accept forwarded and/or proxy tickets

Kerberos V5 Ticket Lifetimes

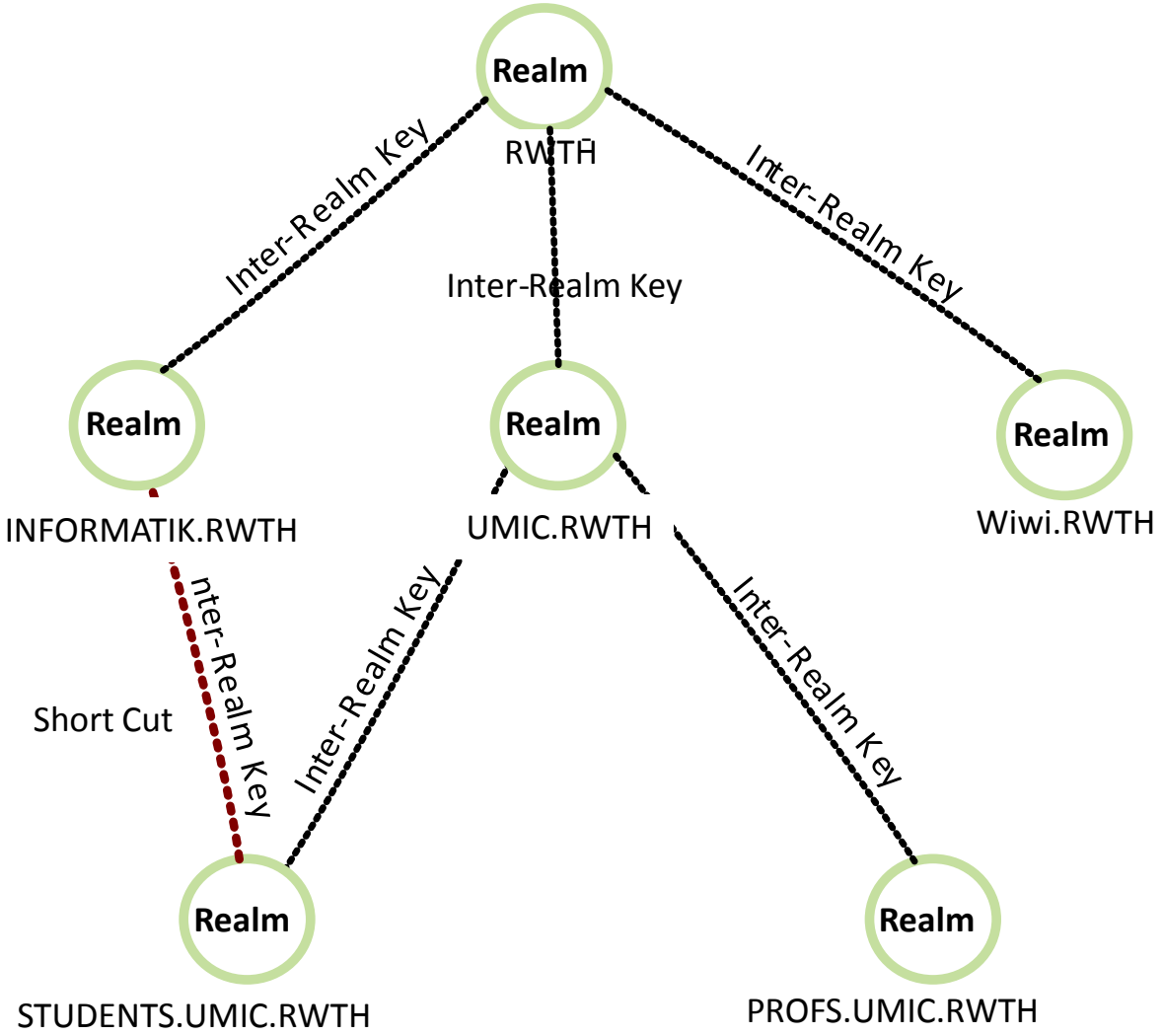
- In Kerberos v4 maximum lifetime of ticket restricted to ~ 21 h as one octet used for lifetime in 5 minute units
- Kerberos v5 allows for start time and end time indication
  - Lifetime can be set as long as wanted
- Renewable Ticket Flag
  - Ticket has second expiration time: renew-till
  - Ticket must be sent to the KDC prior to the renew-till time to renew it
- Postdated Ticket Flag
  - Purpose: Request a ticket for later use I.e. batch jobs
  - Invalid until the start time has been reached (invalid flag)
  - Ticket must be sent to the KDC to convert it to a valid one before use

Cross-Realm Authentication

- Support of hierarchical realms structures
  - Reduces number of keys required between realms
  - Each realm shares a master key with its parent and master keys with all its children
    - i.e. a child acts as application in the parent realm and the parent realm acts as application in the child realm!
  - In addition short cuts between individual realms are supported
  - Ticket requests are proxied to the appropriate KDC in the foreign realm and include a list of all realms contacted on the way

Kerberos V5 Cross-Realm Hierarchy

- Realms can be Hierarchical Structured
- Both client and KDC run the same algorithm to determine the authentication path
- Short-cuts limit the number of requests

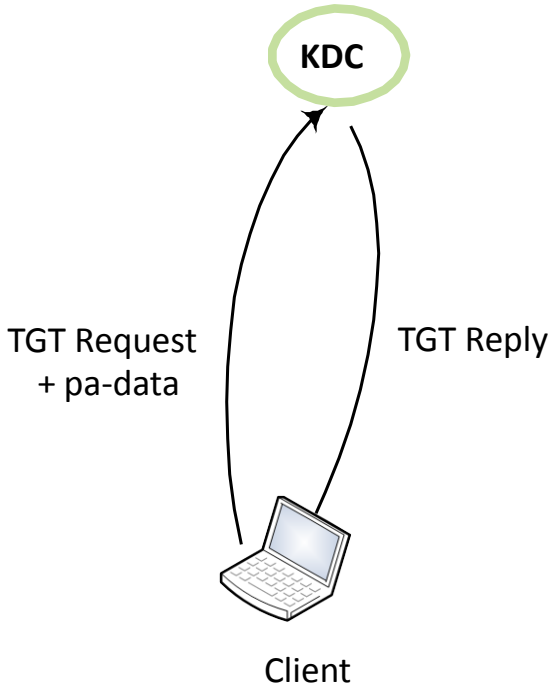


Kerberos V5 Cross-Realm Authentication

- The sequence of realms used is referred to as authentication path
- Client determines authentication path by using a realm-naming convention similar to the DNS naming convention
- Server runs the same algorithm but he may return a TGT that is closer to the final realm if available
- Example:
  - Client located at STUDENTS.UMIC.RWTH
  - Application-Server located at WIWI.RWTH
  - Client asks STUDENTS.UMIC.RWTH for a TGT for UMIC.RWTH
    - TGT will be encrypted with the inter-realm-key shared between the two realms
  - Client then uses the TGT to request a TGT for RWTH from UMIC.RWTH
  - And finally asks RWTH for a TGT for WiWi.RWTH
  - Now the client can use this TGT to request a ticket for the application server

Pre-Authentication

- What is pre-authentication?
  - TGT-request to the KDC is unauthenticated
  - Reply contains data encrypted with user's master secret
  - Pre-authentication is used to authenticate TGT-request to KDC already
- Why use pre-authentication?
  - Limit Denial of Service Attacks
  - Limit Dictionary Attacks
- How to enable it?
  - TGT-request to the KDC contains pre-authentication field that can be used to carry a MAC
  - E.g. the MAC can be computed with the user's password and a one-time-password displayed on the user's token



Use of Public Key Cryptography in Kerberos 5

- The pre-authentication data field used in the TGT-request message enables PKC use
- Pre-authentication data contains a digitally signed hash of the entire ticket-request
- KDC stores the client certificate instead of the user password
- The KDC has two options:
  - Key Transport: The KDC generates a random key and transmits this key encrypted with the client's public key.
  - Key Agreement: The KDC creates a Diffie-Hellman private/public value pair and both parties compute the DH session key.
- Further Kerberos protocol steps remain unmodified!

Examples for Applications using Kerberos

- Resource Reservation Protocol (RSVP)
- Radius/Diameter
- Windows 2000 and later for all authentication procedures
- Apple's Mac OS X in client and server versions
- Red Hat Enterprise Linux 4 and later in both client and server versions
- DHCP
- Telnet
- FTP
- SNMP
- ….

Implementation Issues

Vulnerability in Kerberos Could Allow Elevation of Privilege (3011780)

Published: November 18, 2014

Version: 1.0

Executive Summary

This security update resolves a privately reported vulnerability in Microsoft Windows Kerberos KDC that could allow an attacker to elevate unprivileged domain user account privileges to those of the domain administrator account. An attacker could use these elevated privileges to compromise any computer in the domain, including domain controllers. An attacker must have valid domain credentials to exploit this vulnerability. The affected component is available remotely to users who have standard user accounts with domain credentials; this is not the case for users with local account credentials only. When this security bulletin was issued, Microsoft was aware of limited, targeted attacks that attempt to exploit this vulnerability.

This security update is rated Critical for all supported editions of Windows Server 2003, Windows Server 2008, Windows Server 2008 R2, Windows Server 2012, and Windows Server 2012 R2. The update is also being provided on a defense-in-depth basis for all supported editions of Windows Vista, Windows 7, Windows 8, and Windows 8.1. For more information, see the **Affected Software** section.

The security update addresses the vulnerability by correcting signature verification behavior in Windows implementations of Kerberos. For more information about the vulnerability, see the **Frequently Asked Questions (FAQ)** subsection for the specific vulnerability.

Reading

- Kaufman Chapter 13 and 14
- J. Kohl, B. Neuman: “The Evolution of the *Kerberos* Authentication Service”, 1994
- RFC 4120: “The Kerberos Network Authentication Service (V5)“, 2005
- RFC 4556: “Public Key Cryptography for Initial Authentication in Kerberos (PKINIT)”, 2006