Introduction into Cyber Security

Chapter 10: Kerberos

WiSe 18/19

Chair of IT Security

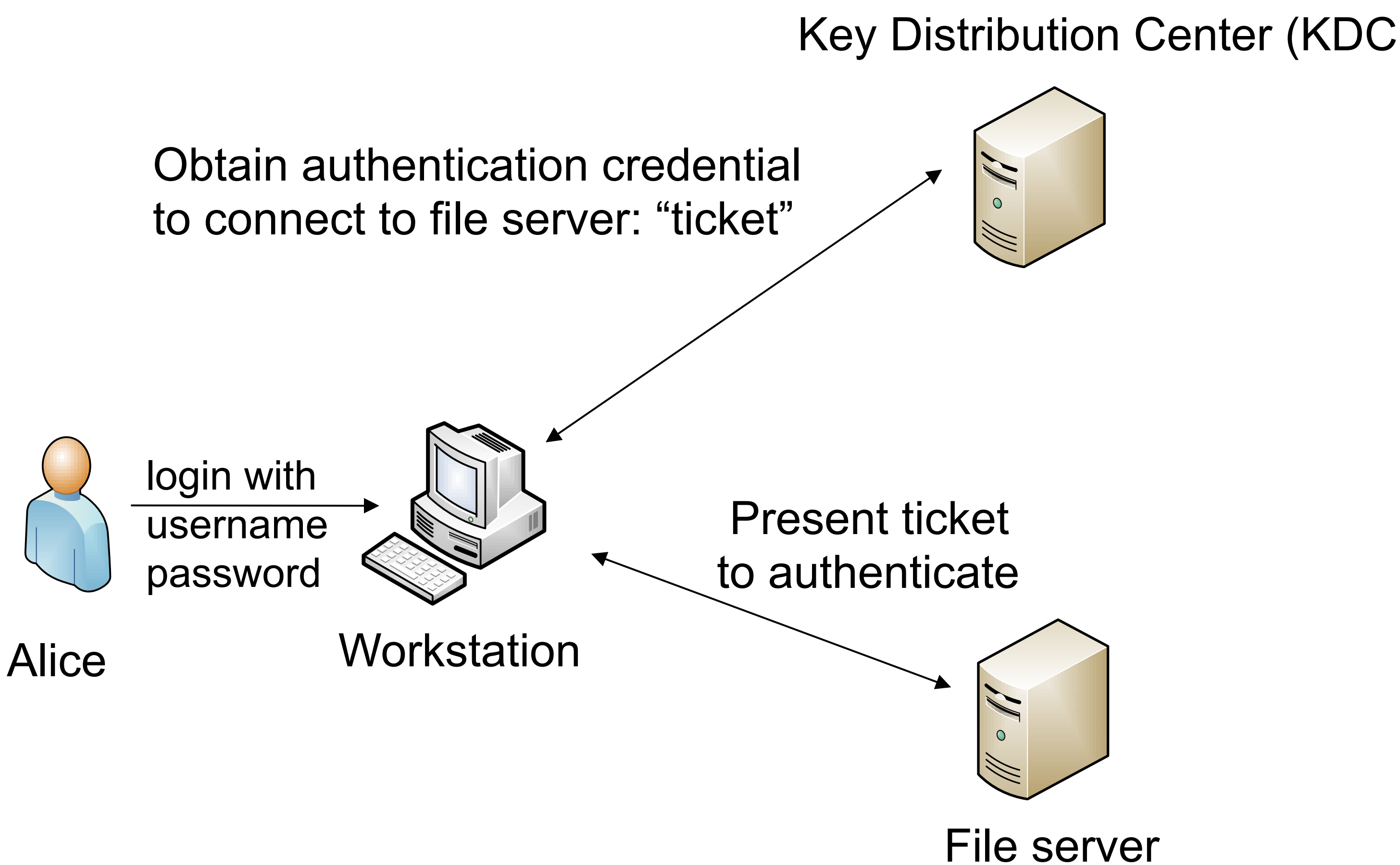
Kerberos – General Information (1/2)

- Computer network authentication protocol
  - Works over non-secure networks
  - Builds on symmetric key cryptography
  - Enables network applications to authenticate their peers (proof of identity in a secure manner)
  - Provides mutual authentication (both the user and the server verify their identity)
  - Allows also for key agreement
- Originally designed by the MIT
- Kerberos is the name of the (three headed) dog that protects the entrance to Hades in Greek mythology
- Currently versions 4 and 5 are in use

Kerberos – General Information (2/2)

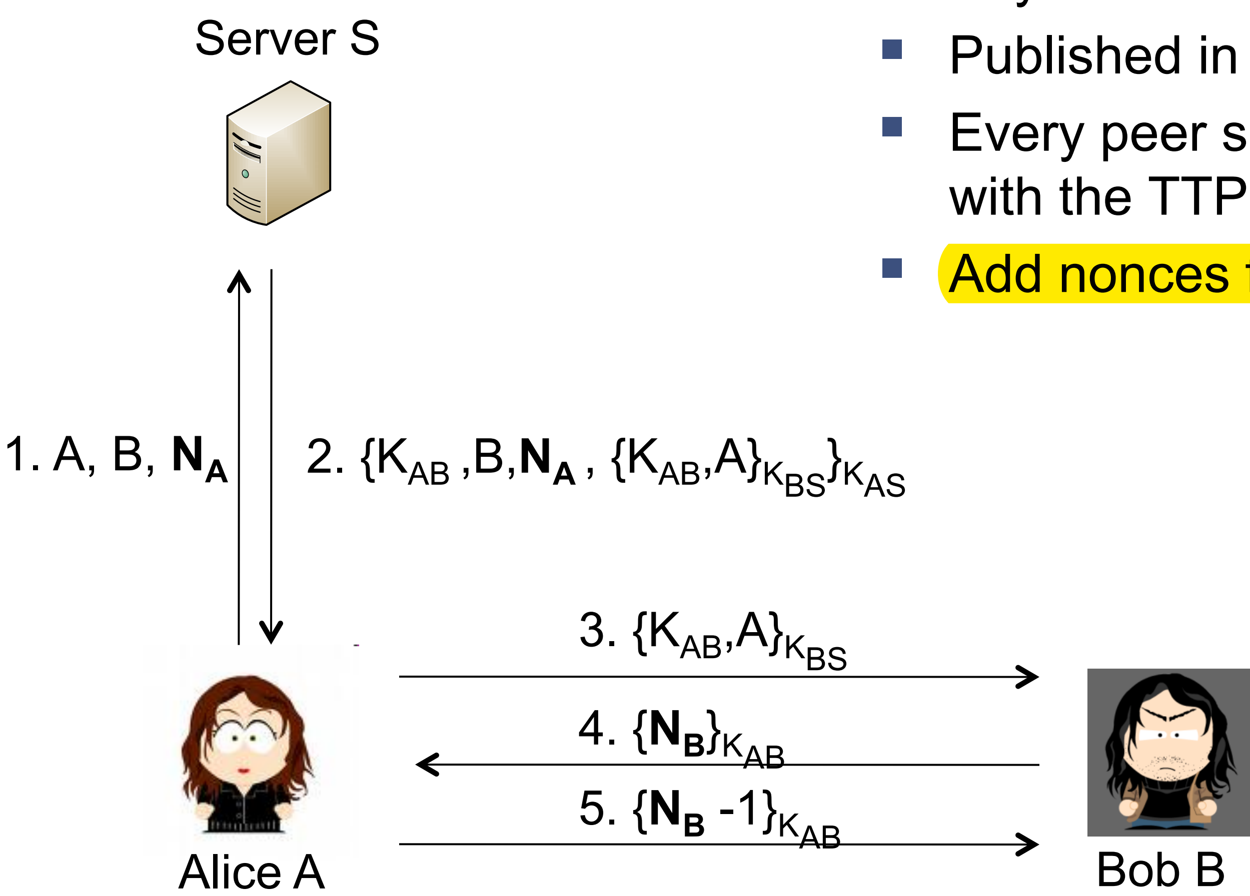
- Based on the authentication protocol by Needham and Schroeder
- Widely used, e.g.,
  - Windows 2000 and later use Kerberos as default authentication method
  - FreeBSD, Apple's Mac OS X in client and server versions
  - Red Hat Enterprise Linux 4 and later in both client and server versions
- Works on the basis of “tickets”
- Requires a trusted third party (TTP)

Kerberos Overview



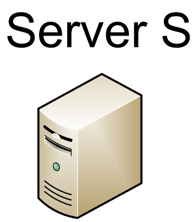
Building Block: Needham-Schroeder Protocol

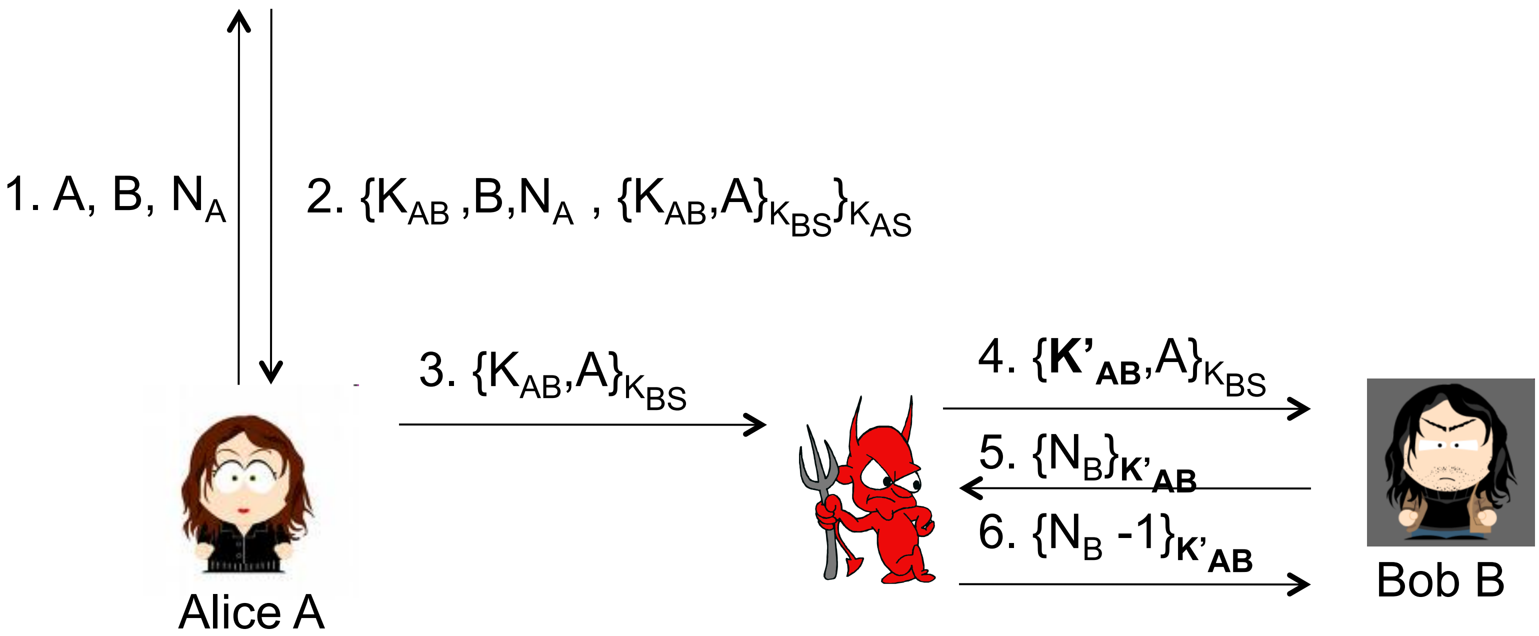
- Key establishment protocol
- Published in 1978
- Every peer shares a symmetric key with the TTP (Server S)
- Add nonces for replay protection



Attack Against the Needham-Schroeder Protocol

- Found by Denning and Sacco in 1981
- Problem: not only A can correctly reply B in the last step if an attacker knows an old session key





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 K_{KDC} known only to KDC
- KDC shares a secret master key K_B with each resource Bob
- KDC shares a secret master key 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

Ticket Granting Tickets (1)

- When Alice logs on, the workstation asks the KDC for a session key for Alice
- KDC
  - Generates a session key S_A for Alice and encrypts it using Alice's master key
  - Generates a "ticket granting ticket"
    - Including S_A, "Alice", and the lifetime of the ticket
  - Encrypts the ticket with its own master key K_{KDC}
  - Sends the ticket granting ticket (TGT) and the encrypted session key to the workstation
- Workstation uses Alice's password to derive Alice's master key and decrypts the session key S_A

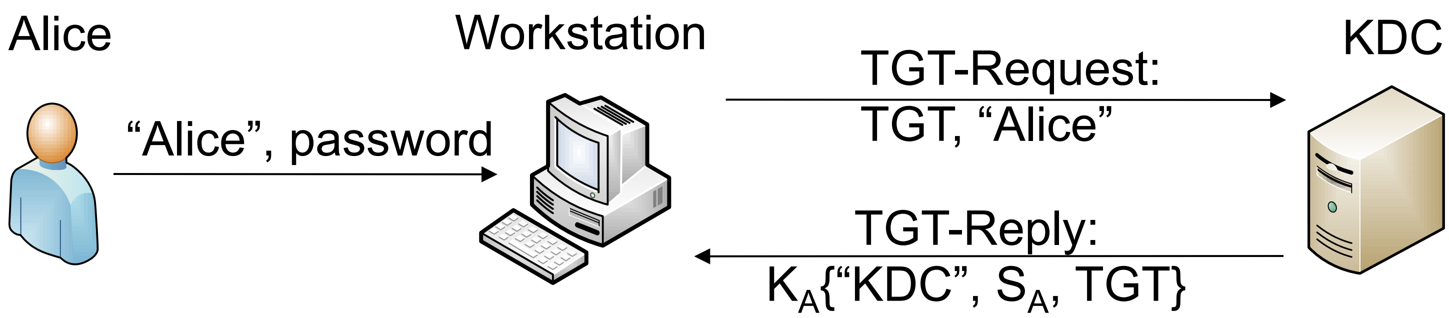
Ticket Granting Tickets (2)

- The ticket granting ticket is used later on to provide S_A to the KDC such that KDC does not have to store any stateful information
- KDC then uses 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

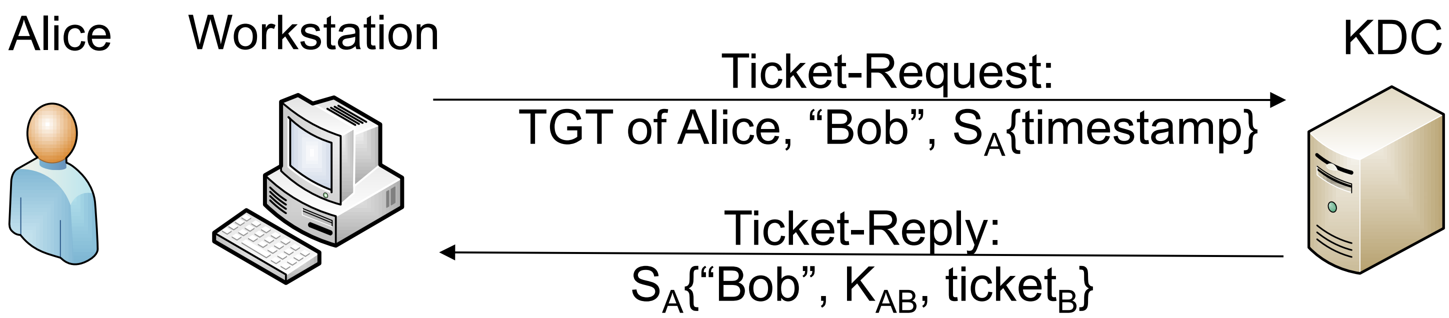
- When Alice wants to access a resource Bob, then she sends a “ticket request” to the KDC
- KDC issues a “ticket” for Bob to Alice by
  - Generating a session key K_{AB}
  - Generating a “ticket” including the session key K_{AB}, “Alice” and the lifetime of the ticket
  - Encrypting the ticket with B’s master key
  - Encrypting K_{AB} and the encrypted ticket with A’s session key and sends this back to Alice
- Alice can now present the encrypted ticket to Bob and Bob will be able to decrypt the ticket with its master key

Obtaining a Ticket Granting Ticket



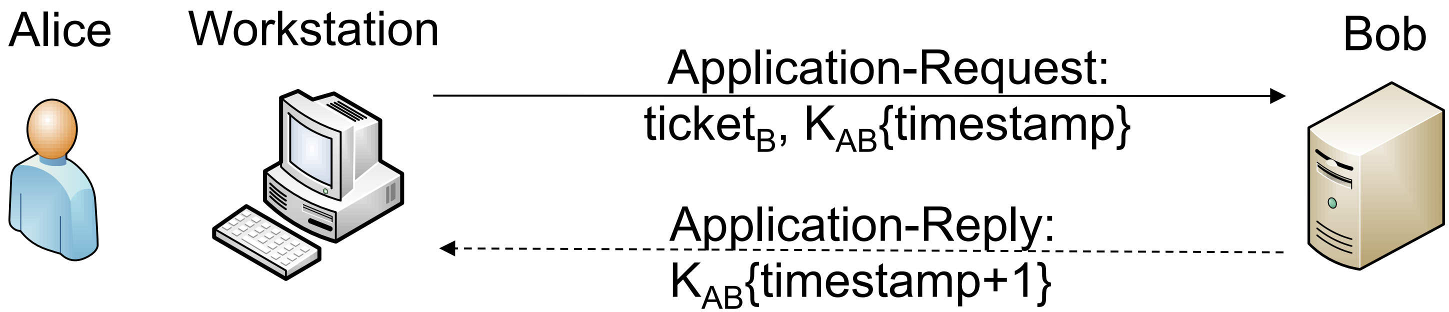
- TGT = Ticket Granting Ticket = K_{KDC}{“Alice”, S_A, lifetime}
- The workstation requests a ticket granting ticket from the KDC on behalf of Alice
- The KDC replies with the TGT and a session key S_A, both encrypted with Alice’s master key K_A
- The workstation requests the password from Alice, uses it to derive the master key and decrypts and stores the session key S_A and the TGT
- Note: in the following K{X} means X is encrypted with some symmetric encryption algorithm using K as the key

Obtaining a Ticket for a Resource



- Alice is already logged on
- Workstation sends a ticket request on behalf of Alice to the KDC requesting a ticket for Bob
- The ticket request includes the TGT of Alice, an identifier for Bob and a timestamp encrypted with Alice’s session key
- KDC decrypts the TGT to obtain S_A, decrypts and checks the timestamp, generates K_{AB} and $ticket_B = K_B$\{“Alice”, K_{AB}, lifetime\}
- KDC encrypts the ticket, the key K_{AB}, and Bob’s identifier with S_A and includes it in the ticket-reply back to the workstation

Login into the Resource Bob



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

Replicated KDCs (1)

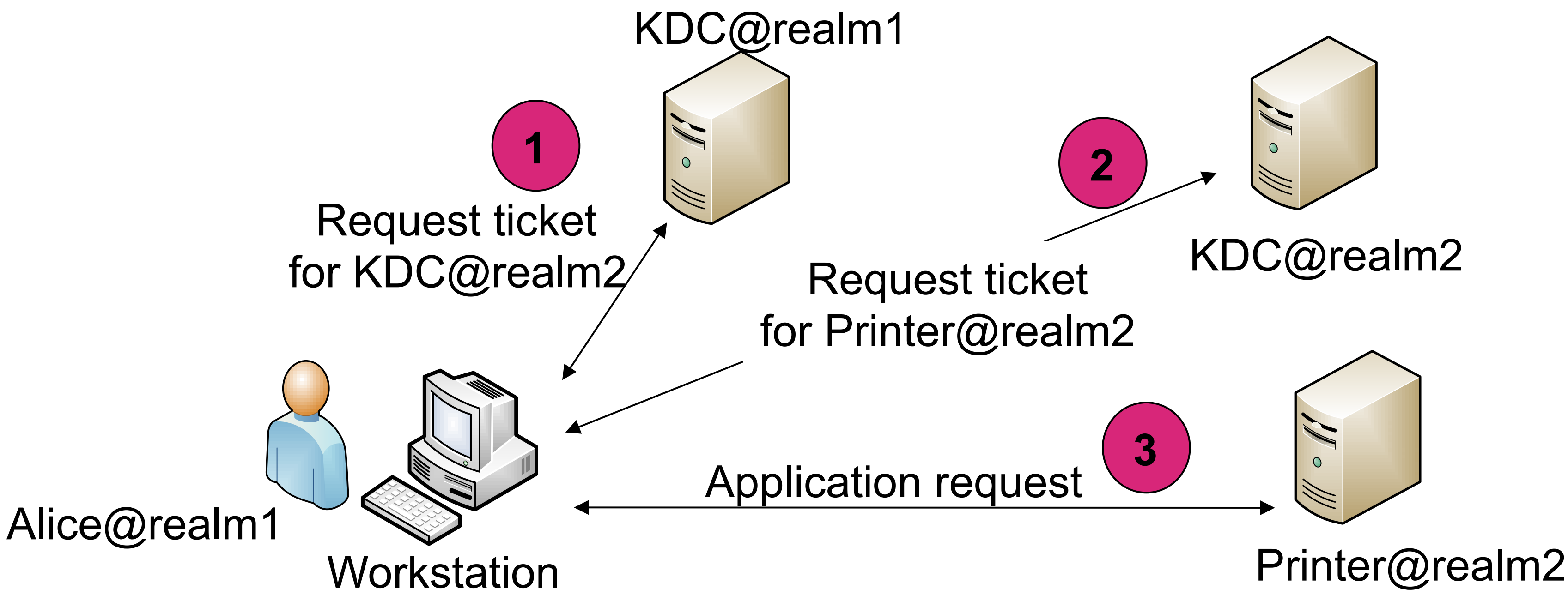
- If a single KDC is used it is a single point of failure and potentially a performance bottleneck
- Therefore it is desirable to have multiple KDCs that store the same master KDC key
- Kerberos supports having a single master copy of the database that stores the principle identifiers and master keys
  - 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 for other operations anymore

Replicated KDCs (2)

- If the KDC with the master copy is down no updates can be made but regular application requests can still be handled by the slave KDCs
- 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



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. Bob's master key is changed while Alice holds a ticket for Bob, 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
- Plaintext-Cipher-Block-Chaining works as follows
  - $c_0 := IV$, $m_o=0$, $c_{n+1} := E_K(m_{n+1} \text{ xor } c_n \text{ xor } m_n)$
  - $m_{n+1} := D_K(c_{n+1}) \text{ xor } c_n \text{ xor } m_n$
- Idea: put some recognizable plaintext at the end of each plaintext, if this decrypts correctly, then no modification took place
- In addition Kerberos provides integrity only:
  - Based on a specifically designed algorithm
    - Uses the session key and the message as input

Version 4 vs. 5

- Limitations of version 4
  - No choice of encryption algorithms (DES only)
  - Relatively small max lifetime of a ticket (ca. 21 hours)
  - No support for ticket delegation
  - Limitations in principle naming and realm names (e.g., length, no dot possible)
- => Version 5
  - Support of different encryption algorithms
  - Sequence numbers as alternative to timestamps
  - Support of hierarchical realms structures

Kerberos V5 Enhancements

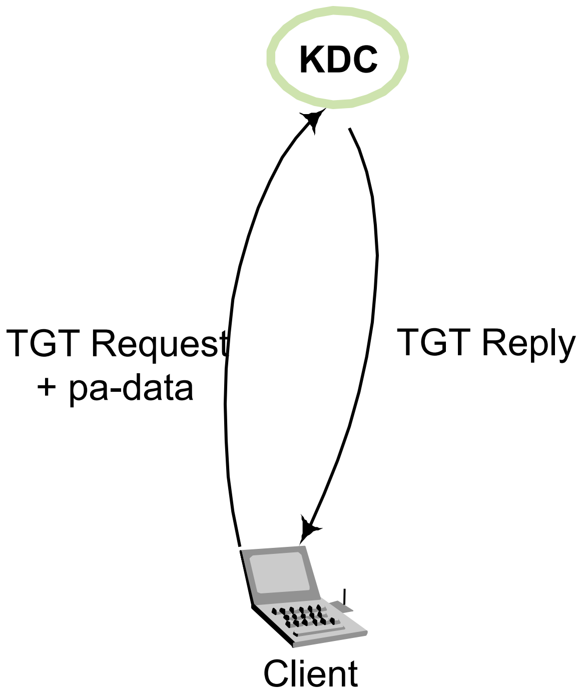
- More flexible naming structures (can work with different kinds of networks)
- Supports delegation of authentication
  - Proxiable vs. forwardable ticket flags
    - Applications can decide whether to accept such tickets or not
- Longer and flexible ticket lifetime (start and end time indication)
- Renewable and postdated (for later use) tickets

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

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., signed hash of the entire ticket request)
  - Simplest form: current timestamp encrypted with the user's key



Use of Public Key Cryptography (PKC) 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!

Limitations

- Every network service must be individually modified for use with Kerberos
- Doesn’t work well in time sharing environment
- Requires a secure Kerberos Server
- Requires a continuously available Kerberos Server
- Stores all passwords encrypted with a single key
- Assumes workstations are secure
- May result in cascading loss of trust
- Scalability

Summary

- Kerberos is a network authentication protocol
  - Allows to access resources from anywhere on the network
  - Authentication is by password
    - It has to be provided/entered only once
    - Workstation does not store the password (except for the brief initial login)
  - User's password is never transmitted
- Client-server model, requires a TTP
- Protection against eavesdropping and replay attacks
- Drawbacks:
  - Single point of failure, requires clock synchronization
  - Unsecure when DES is used

Further Reading

- Kaufman et al. “Network Security” 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