Chair for Network Architectures and Services
Institute of Informatics
TU München – Prof. Carle

Network Security

Chapter 3

Kerberos

Acknowledgments

This course is based to a significant extend on slides provided by Günter Schäfer, author of the **book "Netzsicherheit - Algorithmische Grundlagen und Protokolle"**, available in German from **dpunkt Verlag**. The English version of the book is entitled “Security in Fixed and Wireless Networks: An Introduction to Securing Data Communications” and is published by Wiley is also available. We gratefully acknowledge his support.

The slides by Günter Schäfer have been partially reworked by Heiko Niedermayer, Ali Fessi, Ralph Holz and Georg Carle.

Kerberos (1)

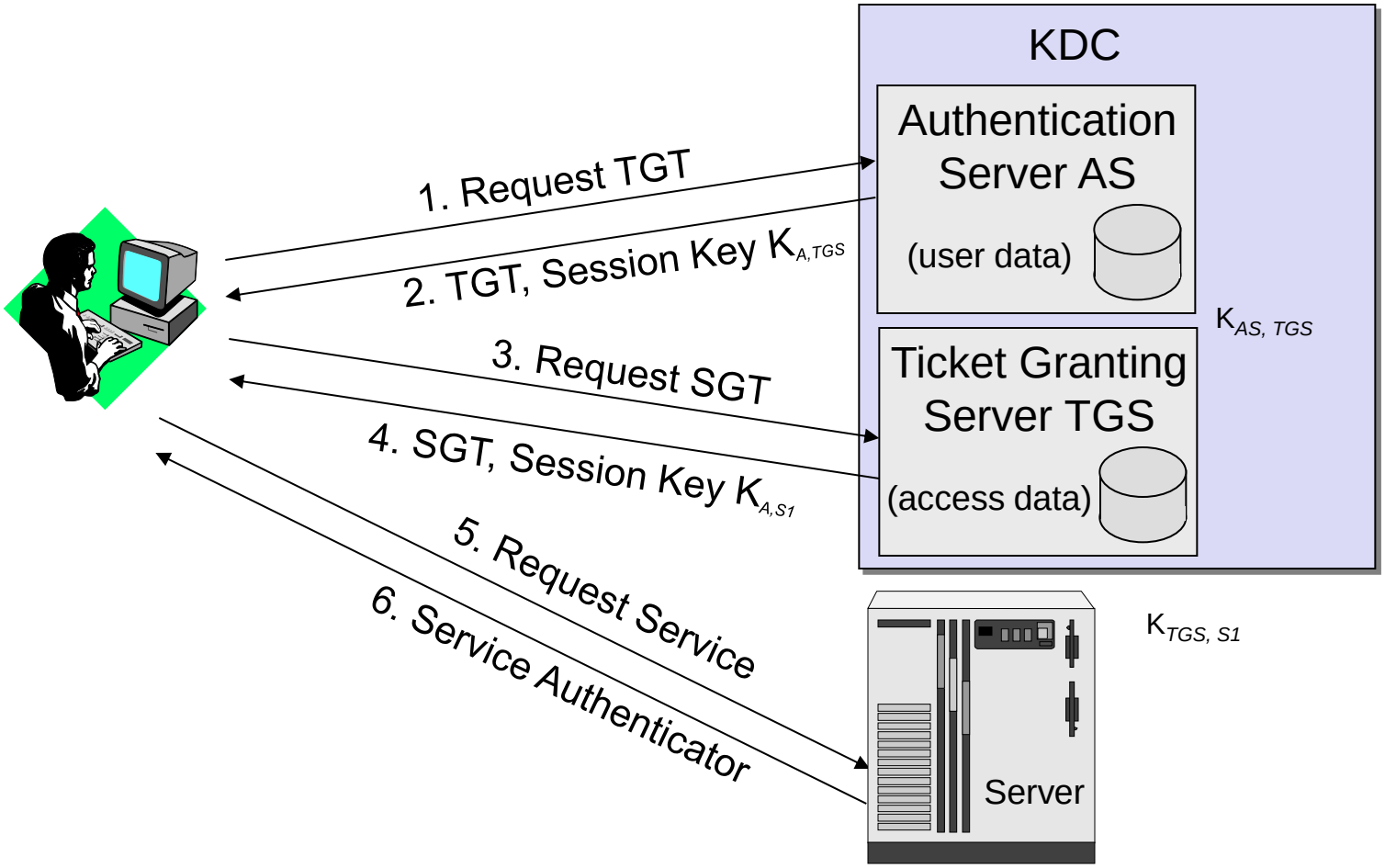
- Kerberos is an authentication and access control service for work-station clusters that was designed at the MIT during the late 1980s
- Design goals:
  - *Security:* eavesdroppers or active attackers should not be able to obtain the necessary information to impersonate a user when accessing a service
  - *Reliability:* as every use of a service requires prior authentication, Kerberos should be highly reliable and available
  - *Transparency:* the authentication process should be transparent to the user beyond the requirement to enter a password
  - *Scalability:* the system should be able to support a large number of clients and servers
- The underlying cryptographic primitive of Kerberos is symmetric encryption (Kerberos V. 4 uses DES, V. 5 allows other algorithms)
- A good tutorial on the reasoning beyond the Kerberos design is given in [Bry88a], that develops the protocol in a series of fictive dialogues

Kerberos (2)

- The basic usage scenario of Kerberos is a user, *Alice,* who wants to access one or more different services (e.g. AFS, printing server, email server), that are provided by different servers S1, S2, ... connected over an insecure network

- Kerberos deals with the following security aspects of this scenario:
  - *Authentication:* Alice will authenticate to an *authentication server (AS)* who will provide a *temporal permit* to demand access for services. This permit is called *ticket-granting ticket (TGT, also called* $Ticket_{TGS}$*)* and is comparable to a temporal passport.
  - *Access control:* by presenting her *ticket-granting ticket (*$Ticket_{TGS}$*)* Alice can demand a *ticket granting server (TGS)* to obtain *access to a service* provided by a specific server *S1*. The TGS decides if the access will be permitted and answers with a *service granting ticket* (SGS, also called $Ticket_{S1}$) for server *S1*.
  - *Key exchange:* the authentication server provides a *session key* $K_{A,TGS}$ for communication between Alice and TGS and the TGS provides a *session key* $K_{A,S1}$ for communication between Alice and *S1*. The use of these session keys also serves for authentication purposes.

Kerberos (3)



Accessing a Service with Kerberos Version 4 - Protocol Overview

Kerberos (4)

- At the beginning, the user *A* logs on at his workstation and requests to access a service:
  - The workstation represents him in the Kerberos protocol and sends the first message to the authentication server *AS,* containing his *name A*, a *timestamp* t_A , the name of an appropriate *ticket granting server TGS* and the requested ticket lifetime:

    1.) $A \rightarrow AS: (A, t_A, TGS, RequestedTicketLifetime_{TGS})$

- *AS* looks up *A* and his password in the user's database, generates the master key $K_{A,AS}$ out of A's password ($K_{A,AS}$ = MD5(Password$_A$)), extracts the workstation *IP address* $Addr_A$, creates a *ticket granting ticket* $Ticket_{TGS}$ and a *session key* $K_{A,TGS}$, and sends the following message to A:

  2.) $AS \rightarrow A: \{K_{A,TGS}, TGS, t_{AS}, LifetimeTicket_{TGS}, Ticket_{TGS}\}_{K_{A,AS}}$

  with $Ticket_{TGS} = \{K_{A,TGS}, A, Addr_A, TGS, t_{AS}, LifetimeTicket_{TGS}\}_{K_{AS,TGS}}$

- Upon receipt of this message, the workstation asks user A to type in her password, computes the key $K_{A,AS}$ from it, and uses this key to decrypt the message. If Alice does not provide her "authentic" password, message (2) can not be decrypted correctly and the protocol run will fail.

Kerberos (5)

- Alice creates a so-called *authenticator* and sends it together with the ticket-granting ticket and the server name *S1* to TGS:

  3.) $A \rightarrow TGS: (S1, Ticket_{TGS}, Authenticator_{A,TGS})$

  with $Authenticator_{A,TGS} = \{A, Addr_A, t'_A\}_{K_{A,TGS}}$

  - With the Authenticator, *A* can prove to TGS that she knows the secret $K_{A,TGS}$
  - In order to counter reply attacks, a fresh timestamp t'_A is included in the *Authenticator.*
  - An authenticator must be used only once.
- Upon receipt, *TGS* decrypts $Ticket_{TGS}$, extracts the session key $K_{A,TGS}$ and uses this key to decrypt $Authenticator_{A,TGS}$.
- If the name and address in the authenticator and in the ticket are matching and the timestamp t'_A is still fresh (not older than 5 minutes), it checks if *A* may access the service S1 based on the access policies database and creates the following message:

  4.) $TGS \rightarrow A: \{K_{A,S1}, S1, t_{TGS}, Ticket_{S1}\}_{K_{A,TGS}}$

  with $Ticket_{S1} = \{K_{A,S1}, A, Addr_A, S1, t_{TGS}, LifetimeTicket_{S1}\}_{K_{TGS,S1}}$

Kerberos (6)

- Alice decrypts the message and does now hold a session key for secure communication with *S1*. She now sends a message to *S1* to show him her ticket and a new authenticator:

  5.) $A \rightarrow S1: (Ticket_{S1}, Authenticator_{A,S1})$

  with $Authenticator_{A,S1} = \{A, Addr_A, t''_A\}_{K_{A,S1}}$

- Here, the *Authenticator* is used to counter *replay attacks.*
- Upon receipt, *S1* decrypts the ticket with the key $K_{TGS,S1}$ he shares with *TGS* and obtains the session key $K_{A,S1}$ for secure communication with *A*. Using this key he checks the authenticator and responds to *A*:

  6.) $S1 \rightarrow A: \{t''_A + 1\}_{K_{A,S1}}$

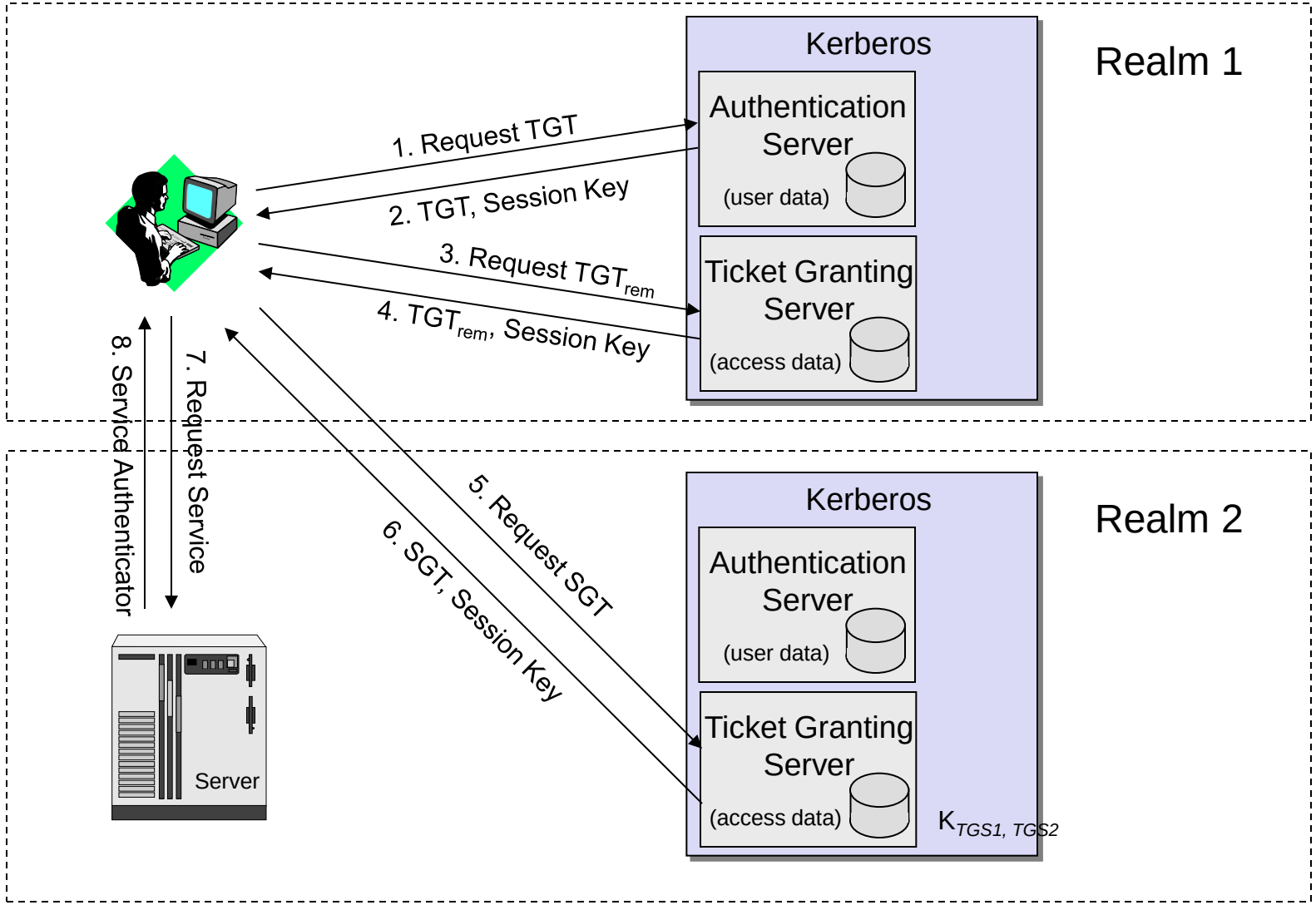
- By decrypting this message and checking the contained value, Alice can verify that she is really communicating with *S1*, as only he (besides *TGS*) knows the key $K_{TGS,S1}$ required to decrypt $Ticket_{S1}$ which contains the session key $K_{A,S1}$, and so only he is able to decrypt $Authenticator_{A,S1}$ and to answer with $t''_A + 1$ encrypted with $K_{A,S1}$

Multiple Domain Kerberos (1)

- Consider an organization with workstation clusters on two different sites, and imagine that user *A* of site 1 wants to use a server of site 2:
  - If both sites do use their own Kerberos servers and user databases (containing passwords) then there are in fact two different *domains*, also called *realms* in Kerberos terminology.
  - In order to avoid that user A has to be registered in both realms, Kerberos allows to perform an inter-realm authentication.

- Inter-realm authentication requires, that the ticket granting servers of both domains share a secret key $K_{TGS1,TGS2}$
  - The basic idea is that the *TGS of another realm* is viewed as a *normal server* for which the TGS of the local realm can hand out a ticket.
  - After obtaining the ticket for the remote realm, Alice requests a service granting ticket from the remote TGS
  - However, this implies that remote realm has to trust the Kerberos authentication service of the home domain of a “visiting” user!
  - Scalability problem: *n* realms require $n \times (n-1) / 2$ secret keys!

Multiple Domain Kerberos (2)



Multiple Domain Kerberos (3)

- Messages exchanged during a multiple domain protocol run:

1.) $A \rightarrow AS1: (A, t_A, TGS1, RequestedTicketLifetime_{TGS})$

2.) $AS1 \rightarrow A: \{K_{A,TGS1}, TGS1, t_{AS}, LifetimeTicket_{TGS1}, Ticket_{TGS1}\}_{K_{A,AS1}}$

with $Ticket_{TGS1} = \{K_{A,TGS1}, A, Addr_A, TGS1, t_{AS}, LifetimeTicket_{TGS1}\}_{K_{AS1,TGS1}}$

3.) $A \rightarrow TGS1: (TGS2, Ticket_{TGS1}, Authenticator_{A,TGS1})$

with $Authenticator_{A,TGS1} = \{A, Addr_A, t'_A\}_{K_{A,TGS1}}$

4.) $TGS1 \rightarrow A: \{K_{A,TGS2}, TGS2, t_{TGS1}, Ticket_{TGS2}\}_{K_{A,TGS1}}$

with $Ticket_{TGS2} = \{K_{A,TGS2}, A, Addr_A, TGS2, t_{TGS1}, LifetimeTicket_{TGS2}\}_{K_{TGS1,TGS2}}$

5.) $A \rightarrow TGS2: (S2, Ticket_{TGS2}, Authenticator_{A,TGS2})$

with $Authenticator_{A,TGS2} = \{A, Addr_A, t''_A\}_{K_{A,TGS2}}$

6.) $TGS2 \rightarrow A: \{K_{A,S2}, S2, t_{TGS2}, Ticket_{S2}\}_{K_{A,TGS2}}$

with $Ticket_{S2} = \{K_{A,S2}, A, Addr_A, S2, t_{TGS2}, LifetimeTicket_{S2}\}_{K_{TGS2,S1}}$

7.) $A \rightarrow S2: (Ticket_{S2}, Authenticator_{A,S2})$

with $Authenticator_{A,S2} = \{A, Addr_A, t'''_A\}_{K_{A,S2}}$

8.) $S2 \rightarrow A: \{t'''_A + 1\}_{K_{A,S2}}$

Kerberos V.5 (1)

- Encoding
  - Kerberos V.5 uses ASN.1 syntax which is more flexible than binary hard-coded type length
  - e.g.

```
HostAddress ::=  SEQUENCE {
                         addr-type[0]      INTEGER,
                         address[1]        OCTET STRING
}
```

- Ticket lifetimes
  - Kerberos V.5 allows for much longer ticket lifetimes, since time encoding in ASN.1 allows for times until Dec 31, 9999.
  - Since it would useful to invalidate Kerberos tickets that have a long lifetime, an additional management of the tickets is required.
    - E.g. in case employee X has left the company and had root access.
  - Kerberos V5 offers the option that tickets can be re-validated by the KDC with a fresh timestamp before they can be re-used.
  - The new ticket lifetime features render the management of master key versions at the KDC more complicated than in V.4

Kerberos V.5 (2)

- Delegation of rights
  - In contrast to Kerberos V.4, in V.5 Alice can request that multiple network addresses should be included in the ticket or no address at all, which means that it can be used from any user's address
  - This is useful, e.g. if the user wants to execute some batch scripts even if he is not logged in, e.g. for periodic backups
  - It is a policy decision if the KDC issues such tickets, and if a service accepts tickets from specific addresses or not.
  - The KDC can log all delegation events and can provide an audit trail in case of a security compromise of a service.

➔ This access control model provides for a lot of flexibility but is also inherently dangerous if not configured correctly.

Kerberos V.5 (3)

- Some improvements of cryptographic primitives
  - The master key is a hash function of Alice's password and the realm name.
  - Kerberos V.5 also allows DES and permits new modes which should provide confidentiality and integrity protection.

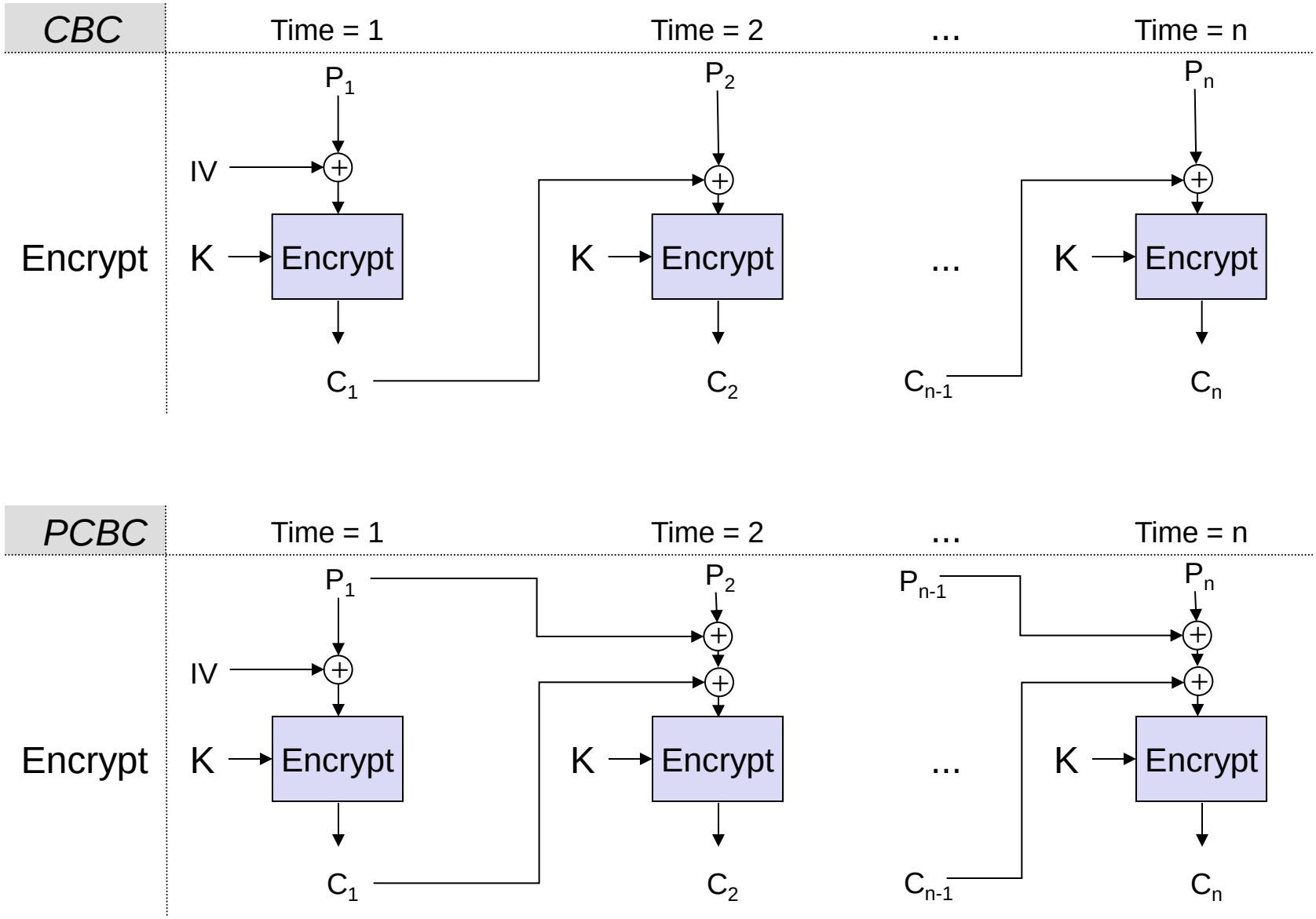
Kerberos - Evading password guessing attacks

- In Kerberos v4, any user, or attacker Mallory, can request a ticket for Alice.
- Mallory can not immediately decrypt the message (2) received from *AS*, since she does not know Alice's master key $K_{A,AS}$
- However, since the algorithm how $K_{A,AS,}$ is derived from Alice's password is known (a hash of the Alice's password), Mallory can perform a password guessing attack (also called dictionary attack) using message (2)
- Kerberos v5 has the *pre-authentication* option to prevent this attack
  - Alice needs to include a fresh timestamp encrypted with her master key $\{t_A\}_{K_{A,AS}}$ when sending the authentication request, i.e. message (1)
- This measure is effective against active attackers.
- However, passive attacks are still possible
  - Mallory can record authentication exchanges of other users and perform a password guessing attack.
- Therefore, it is important to choose good passwords for the security of Kerberos.

Kerberos V4: Misuse of encryption for message authentication (1)

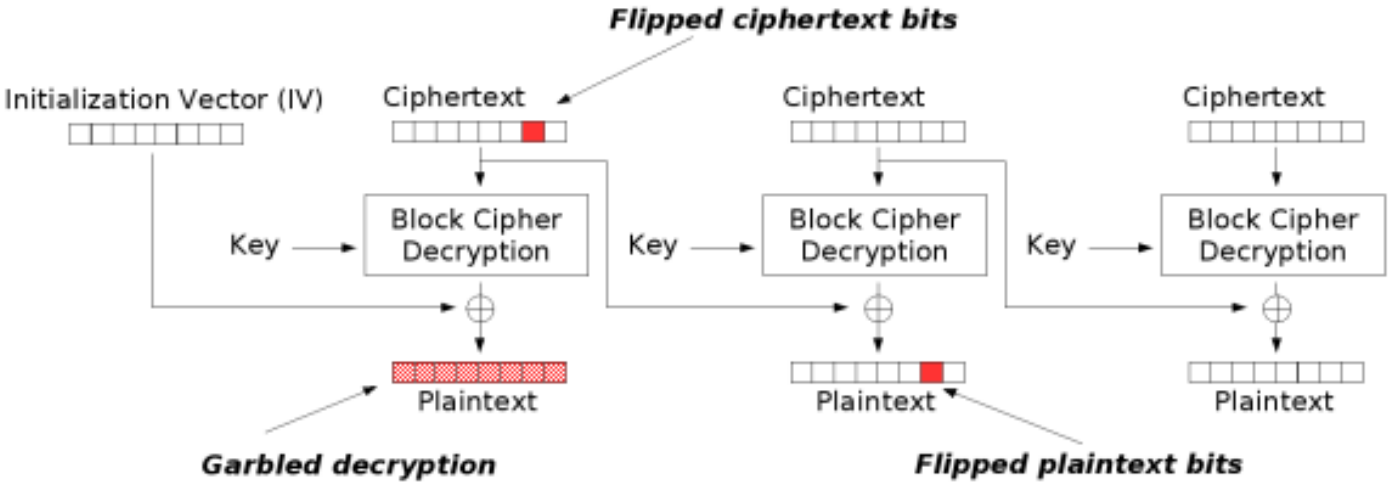
- Kerberos V4 uses DES in a special mode called *Propagating Cipher Block Chaining* (*PCBC*).
- A modification in a cipher text encrypted in CBC mode results into a damage in the next two blocks in the decrypted plain text.
- A modification in a cipher text encrypted in PCBC mode results into a damage in all the remaining blocks in the decrypted plain text.

CBC vs. PCBC


CBC
Time = 1
Time = 2
...
Time = n
Encrypt
P1
P2
Pn
IV
K
Encrypt
C1
C2
Cn-1
Cn
PCBC
Time = 1
Time = 2
...
Time = n
Encrypt
Pn-1

Reminder: CBC Error Propagation

- A distorted cipher text block results in two distorted plaintext blocks, as p_i' is computed using c_{i-1} and c_i



Modification attack or transmission error for CBC

Source: http://www.wikipedia.org/

Kerberos V4: Misuse of encryption for message authentication (2)

- In Kerberos V4, the PCBC mode was supposed to allow for providing message encryption and integrity by processing the message only once:
  - If a block in the cipher text is manipulated by an attacker, all the remaining blocks in the decrypted plain text at the receiver‘s side will be damaged.
  - If order to be able to verify the latter case (i.e. whether the message is damaged) Kerberos V4 uses a checksum computed with the key ($K_{AS,A}$,$K_{A,TGS}$ or $K_{A,S1}$) and the message.
  - The encrypted message is transmitted together with the checksum.
  - The algorithm for the checksum is not documented, only implemented.
- Although not broken. Not believed to be strong.
- Potential problems:
  - The checksum is not a cryptographic hash function.
  - Therefore, it might reveal some information about the key (the one-way property of cryptographic hash functions is not necessarily satisfied)
  - It might be also easier to find two messages with the same check sum (the 2nd pre-image resistance of cryptographic hash function is not necessarily satisfied)
- Not used in V5.

Kerberos V5: are things getting better? (1)

- Algorithms for encryption and message integrity for Kerberos V5 are specified in [RFC3961]
- [RFC3961] allows for *unkeyed* checksums for verifying the data integrity,
  - e.g. CRC, MD5, SHA-1
  - The checksum is encrypted together with the message to be transmitted.

  - *"An unkeyed checksum mechanism can be used with any encryption type, as the key is ignored (a key is not needed for the computing of the checksum), but its use must be limited to cases where the checksum itself is protected, to avoid trivial attacks."* [RFC3961] (Section 4)

  - *"These (unkeyed) checksum types use no encryption keys and thus can be used in combination with any encryption type, but they may only be used with caution, in limited circumstances where the lack of a key does not provide a window for an attack, preferably as part of an encrypted message. Keyed checksum algorithms are recommended."* [RFC3961] (Section 6.1)

Kerberos V5: are things getting better? (2)

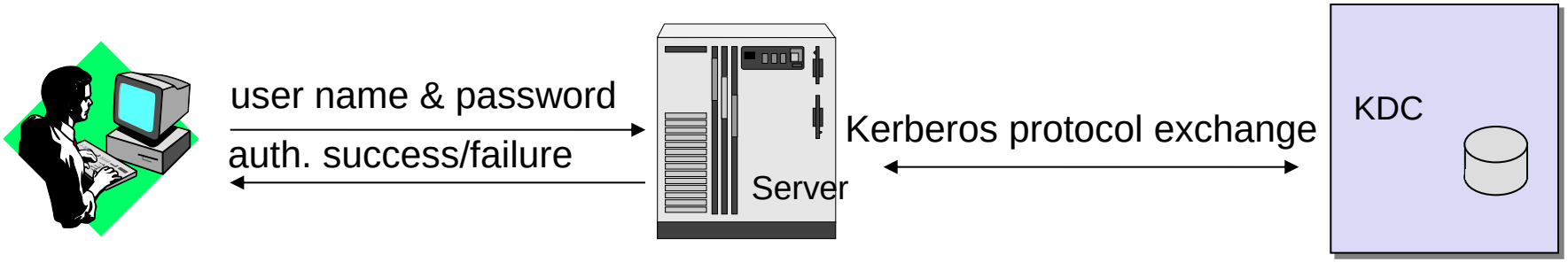
- Fortunately, Kerberos V5 allows also for keyed checksums
  - e.g. HMAC with MD5 or SHA-1
- In this case, different keys can be used for encryption and for data integrity.
  - *"Due to advances in cryptography, some cryptographers consider using the same key for multiple purposes unwise. Since keys are used in performing a number of different functions in Kerberos, it is desirable to use different keys for each of these purposes, even though we start with a single long-term or session key."*

    [RFC3961] (Section 2)
- More information on the algorithms used in Kerberos for encryption and data integrity can be found in
  - [RFC3961] Encryption and Checksum algorithms for Kerberos V5
  - [RFC3962] Adds AES cipher suites for Kerberos V5
  - [RFC4757] Microsoft implementation of Kerberos (with RC4 and HMAC)

- If you want to use Kerberos, you should use the latest version of it.
- It has been around for a while and many competent people have looked at it.

Kerberos – Reality check (1)

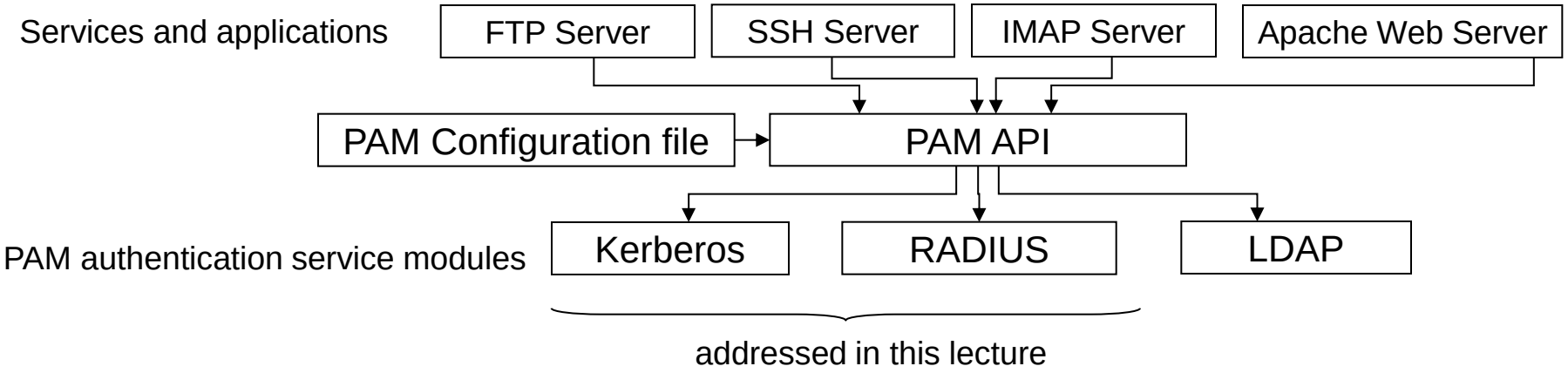
- In many environements, the application of the user is not „kerberized“.



- In this case, the application servers needs to perform the Kerberos exchange on behalf on the user, and get a ticket for itself, if authentication is successful.
- Since the application server requires the user name and password in order to perform the Kerberos authentication exchange successfully, it is very important that the user name and password are protected by another mean, e.g. a TLS tunnel between the user’s application and the application server.
- Note: the user‘s password will be revealed to the application server.

Kerberos – Reality check (2)

- Even worse: most of the application servers do not support Kerberos by themselves.
- However, they can be enabled to use Kerberos with so-called *Pluggable Authentication Modules (PAM)*



- PAM can authenticate users based on different sources of authentication databases and potentially with different protocols.
- The native Kerberos protocol as described by MIT is rarely used today.

Kerberos - Discussion

- General properties
  - Kerberos provides authentication of users and authorization to access services over an insecure network.
  - The KDC is logically separated into an *Authentication Server* and a *Ticket Granting Server*
  - The user needs to enter her password only once (*Single-Sign-On*). Authenticated access to services with the service tickets occurs transparently to the user.
- Reliability
  - The KDC is involved in each authentication process.

    ➔ The KDC must be highly reliable
  - The design of the Kerberos protocol does not foresee a solution for reliability
  - Reliability is implemented with backup KDCs.

Kerberos - Discussion

- Security and complexity
  - The requirement that all hosts need to have synchronized clocks could be fixed if random numbers are used for the *Authenticators* instead of timestamps. (will be treated at assignments)
  - Kerberos V4 tries to protect the password by using it only once in an authentication protocol run.
  - However, dictionary attacks are very easy to perform
  - An „attacker“ Mallory can request a ticket for any user „Alice“ and performs a dictionary attacks based on the received response from the KDC.
  - Kerberos V5 tries to fix this problem with the pre-authentication
  - However, dictionary attacks are still possible.
  - Mallory needs to wait until Alice requests an ticket.
  - Mallory can also listen to Kerberos authentication protocol runs in the network and collects the messages required for a dictionary attack.
  - Dictionary attacks on Kerberos could be avoided with an additional DH exchange (will be treated at assignments)
  - However, such a fix is currently not foreseen.

References

[Bell95] M. Bellare and P. Rogaway, Provably Secure Session Key Distribution - The Three Party Case, Proc. 27th STOC, 1995, pp 57--64

[Boyd03] Colin Boyd, Anish Mathuria, “Protocols for Authentication and Key Establishment”, Springer, 2003

[Bry88a] R. Bryant. *Designing an Authentication System: A Dialogue in Four Scenes.* Project Athena, Massachusetts Institute of Technology, Cambridge, USA, 1988.

[Diff92] W. Diffie, P. C. van Oorschot, and M. J. Wiener. Authentication and authenticated key exchanges. Designs, Codes, and Cryptography, 1992

[Dol81a] D. Dolev, A.C. Yao. *On the security of public key protocols.* Proceedings of IEEE 22nd Annual Symposium on Foundations of Computer Science, pp. 350-357, 1981.

[Fer00] Niels Ferguson, Bruce Schneier, “A Cryptographic Evaluation of IPsec”. http://www.counterpane.com/ipsec.pdf 2000

[Fer03] Niels Ferguson, Bruce Schneier, „Practical Cryptography“, John Wiley & Sons, 2003

[Gar03] Jason Garman, “Kerberos. The Definitive Guide”, O'Reilly Media, 1st Edition, 2003

References

[Kau02a] C. Kaufman, R. Perlman, M. Speciner. *Network Security*. Prentice Hall, 2nd edition, 2002.

[Koh94a] J. Kohl, C. Neuman, T. T'so, *The Evolution of the Kerberos Authentication System.* In Distributed Open Systems, pages 78-94. IEEE Computer Society Press, 1994.

[Mao04a] W. Mao. *Modern Cryptography: Theory & Practice*. Hewlett-Packard Books, 2004.

[Nee78] R. Needham, M. Schroeder. *Using Encryption for Authentication in Large Networks of Computers.* Communications of the ACM, Vol. 21, No. 12, 1978.

[Woo92a] T.Y.C Woo, S.S. Lam. *Authentication for distributed systems.* Computer, 25(1):39-52, 1992.

[Lowe95] G. Lowe, „An Attack on the Needham-Schroeder Public-Key Authentication Protocol”, *Information Processing Letters*, volume 56, number 3, pages 131-133, 1995.

Additional references from the IETF

[RFC2560] M. Myers, et al., “X.509 Internet Public Key Infrastructure Online Certificate Status Protocol – OCSP”, June 1999

[RFC3961] K. Raeburn, “Encryption and Checksum Specifications for Kerberos 5”, February 2005

[RFC3962] K. Raeburn, “Advanced Encryption Standard (AES) Encryption for Kerberos 5”, February 2005

[RFC4757] K. Jaganathan, et al., “The RC4-HMAC Kerberos Encryption Types Used by Microsoft Windows ”, December 2006

[RFC4120] C. Neuman, et al., “The Kerberos Network Authentication Service (V5)”, July 2005

[RFC4537] L. Zhu, et al, “Kerberos Cryptosystem Negotiation Extension”, June 2006

[RFC5055] T. Freeman, et al, “Server-Based Certificate Validation Protocol (SCVP)”, December 2007