

# Authentication

## ***Abstract***

*This section introduces general concepts about user authentication. Security aspects of authentication by password are presented. Schemes of authentication by Challenge & Response and One-Time Password are reviewed, including the Lamport's hash chain procedure. Schemes based on security tokens and biometrics are mentioned. The Kerberos system for key distribution and indirect authentication is presented.*

# Outline

---

- **General concepts of authentication**
  - Username and password
  - Challenge and response
  - One-Time Password
  - Biometrics
  - Kerberos

# Authentication and Authorization

---

- *Authentication*: guaranteeing the identity of someone or something
  - ◆ of a user
  - ◆ of a machine
  - ◆ of the source of data
  - ◆ IETF RFC 4949 definition: the process of verifying an identity claimed by or for a system entity
  - ◆ NIST definition: the process of establishing confidence in user identities that are presented electronically to an information system
- *Authorization*: determining what resources or services an authentic entity can access
  - ◆ authentication does not imply authorization
    - Alice is authorized to access a service, but Bob is not
  - ◆ authorization depends on authentication
    - only the true Alice is authorized

# Steps of Authentication

---

- Registration

- ◆ the initial requirement of user authentication is that the user has been registered with the system

- Identification

- ◆ the *claimant* presents an identifier to the *verifier*, i.e. the authenticating system

- Verification

- ◆ authentication information is presented or generated, which proves the binding between the entity and the identifier

# Types of Authentication

---

- Local authentication
  - ◆ the user accesses a local system (e.g., laptop)
- Direct (remote) authentication
  - ◆ the user accesses directly a remote system, which provides the service (e.g., FTP server)
- Indirect authentication
  - ◆ authentication is delegated to a third party (e.g., Kerberos)
- Off-line authentication
  - ◆ services decides about authentication autonomously, without contacting the user or authentication server (e.g., PKI certificates)

# Means of Authentication

---

- *What you know*
  - ◆ password, PIN, ...
- *What you have*
  - ◆ cryptographic keys, keycard, smart card, USB dongle, ... (a.k.a.: token)
- *What you are*
  - ◆ static biometrics: fingerprint, retina, face
- *What you do*
  - ◆ dynamic biometrics: voice pattern, handwriting
- Combined authentication
  - ◆ two factors: USB dongle + fingerprint
  - ◆ three factors: smartcard + its PIN + fingerprint

# Security of Authentication

---

- Security of credentials *transmission*
  - ◆ on authentication, credentials should be not transmitted as plain text, or not even transmitted
  - ◆ otherwise, Oscar can eavesdrop credentials and pretend to be Alice with Bob
- Security of credentials *storage*
  - ◆ credentials must not be stored as plaintext in a data base
  - ◆ otherwise, if the system is compromised, an attacker can impersonate anyone whose credentials have been stolen

# Outline

---

- General concepts of authentication
- **Username and password**
- Challenge and response
- One-Time Password
- Biometrics
- Kerberos

# Username and Password

---



- The first and simplest authentication method
  - ◆ the username is public
  - ◆ security is based on *secrecy of password* (Kerckhoffs' Principle)
- Username and password should be not transmitted as plain text...
  - ◆ ...but often are (with direct or indirect authentication via TCP)
  - ◆ eavesdropping is a problem
- Username and password must not be stored as plaintext in the DB
  - ◆ a **digest** (*hash*) of passwords is stored
  - ◆ storing just the hash of password is not enough
    - password guessing by dictionary attack
    - open source tools can be very efficient (Hashcat on GPU)

# Password Strength: Entropy

- **Entropy**: a measure of *unpredictability* of the state, or equivalently of the *information content* of a message, or of the *average information content* of a source (by Claude E. Shannon, A Mathematical Theory of Communication)

- Let  $X$  be a memoryless source of messages  $x$  over a finite alphabet

$$X \in \{X_1, X_2, \dots, X_N\}$$

with probability distribution

$$p(x_i) = P(x=x_i)$$

- The **Information** [bit] in a single message  $x = x_i$  is defined as

$$I(x) = -\log_2 ( p(x_i) ) \geq 0$$

- The *source Entropy* is its *average Information* (per character) over the alphabet

$$H(X) = -\sum_{i=1..N} p(x_i) \cdot \log_2 ( p(x_i) ) \geq 0$$

# Entropy of a Password

- Example of simple alphabetical password  $X$ 
  - ◆ a combination of 7-bit ASCII characters
  - ◆ excluding 32 control characters
  - ◆ 95 printable characters, supposed with same probability
- $H(X) = \log_2 95 \approx 6.57$  bit/character
- A 10-character random password has 65.7 bit of entropy (information)
- Using a real language word (writable with Latin alphabet)
  - ◆ 27 letters and space  $\Rightarrow H(X) = \log_2 27 \approx 4.75$  bit/character
  - ◆ but characters have not same probability...
  - ◆ considering a sample Italian language frequency distribution of letters ("I Promessi Sposi", Alessandro Manzoni)  $\Rightarrow H_1(X) = \approx 3.97$  bit/character
- But real language text is not memoryless...
  - ◆ characters in words and phrases are correlated

# Entropy of a Passphrase

- Considering the correlation in a sequence of characters of a real language phrase, the entropy of a phrase is much lower
- Based on distributions of groups of  $n$  letters (digrams, trigrams, ...) in a large sample text ("I Promessi Sposi", A. Manzoni) and on the definition of *conditional entropy*, the average entropy can be evaluated
  - ◆  $H_1(X) = \approx 3.97$  bit/character       $H_2(X) = \approx 3.15$  bit/character
  - ◆  $H_3(X) = \approx 2.67$  bit/character       $H_4(X) = \approx 2.22$  bit/character
  - ◆  $H_5(X) = \approx 1.87$  bit/character      ....
- Shannon estimated the entropy of an English language text as in range  $0.6 < H < 1.3$  bit/character
  - ◆ today the entropy of a password or passphrase is estimated on the order of about 2 bit/character

# Security of a Password

---

- Various ways envisaged to identify a password
  - ◆ traffic sniffing
    - when transmitted as plain text in direct or indirect authentication
  - ◆ online guessing
    - the attacker interacts directly with the service making attempts (POP3, FTP,...)
    - countermeasures: locking accounts, slowing down next attempts
  - ◆ offline guessing
    - brute force and dictionary attacks to the hashed password file
    - countermeasures: salting, encrypt the pw file, make it accessible only by admin
  - ◆ more creative approaches
    - social engineering
    - shoulder surfing
    - phishing
    - keylogger
    - Van Eck sniffing (pick up monitor radiations)
    - mouse pad survey
    - keyboard acoustic emanation

# Example: Password in UNIX

---

- UNIX systems store account and password information in two files
  - ◆ */etc/passwd*: account generic information, readable by everyone
  - ◆ */etc/shadow* (Linux) or */etc/master.passwd* (BSD): password hashes, readable only by root
- The password digest is made by a *slow* hash function with *salt*
  - ◆ crypt: 25 modified DES encryptions
  - ◆ crypt-MD5: modified crypt based on MD5 (1000 iterations)
  - ◆ crypt-blowfish
  - ◆ crypt-SHA1

# Cracking Weak Passwords

---

- Multi-purpose tools are available to crack weak passwords by various means, including
  - ◆ offline cracking of hashed password files (Windows, Unix, etc.)
  - ◆ brute force and dictionary attacks to the hashed password file
  - ◆ traffic sniffing
  - ◆ WLAN key recovery
  - ◆ GPU to take advantage of parallel computing
- Some of the most popular multi-purpose tools
  - ◆ John the Ripper
  - ◆ Cain & Abel (for Microsoft)
  - ◆ Hashcat

# Outline

---

- General concepts of authentication
- Username and password
- **Challenge and response**
- One-Time Password
- Biometrics
- Kerberos

# Challenge and Response



- It solves the problem of eavesdropping, as *the credentials are not transmitted*
  - ◆ the client must be able to *use a secret information, proving he knows it*
- Challenge
  - ◆ the server S sends a pseudo-random sequence
- Response by the client C may be
  - ◆ by way of *symmetric key encryption*:  
C encrypts the challenge using a shared secret symmetric key  $K$ 
    - $E_K\{\text{challenge}\}$
  - ◆ by way of a *hash function*: C concatenates the challenge with a shared secret symmetric key  $K$  and then hashes it
    - $h(\text{challenge} || K)$
  - ◆ by way of *public key encryption*:  
C encrypts the challenge with its private key  $K_{Cp}$ 
    - $E_{K_{Cp}}\{\text{challenge}\}$

# Vulnerability of Challenge-and-Response Authentication



- Challenge-and-Response authentication is secure on transmission, but **vulnerable on storage of credentials** (in case of symmetric encryption)
  - the server must know the secret
  - brute force and dictionary attacks to the file of hashed secrets
  - public-key encryption solves this vulnerability



# Outline

---

- General concepts of authentication
- Username and password
- Challenge and response
- **One-Time Password**
- Biometrics
- Kerberos

# One-Time Password (OTP)



- Another way to avoid the problem of eavesdropping credentials
- One-Time Password (OTP): a password that is valid only for one login session or transaction
  - ◆ not vulnerable to replay attacks
- The user is given an ordered list of passwords, to be used one after the other, each for a single time
- OTPs are transmitted from the user to the server as plain text
- OTPs can be generated in advance and transmitted as bulk to the user, or generated by a specific algorithm on software or hardware device
  - ◆ token + PIN: *what you have + what you know*

# One-Time Password

## Lamport's Hash Chain Scheme



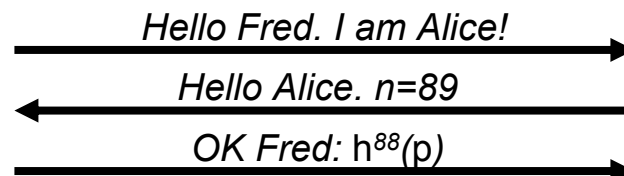
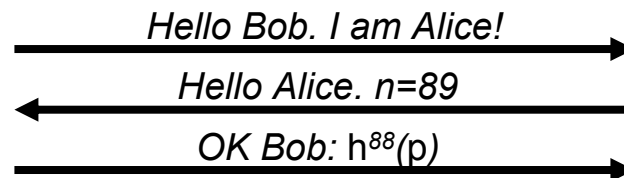
- A Challenge-And-Response scheme to generate OTPs (L. Lamport, 1981), guaranteeing security of both transmission and storage
- Based on repeated computation of a secure *one-way* hash function
- The user Alice knows a password  $p$
- The server Bob, for each user, knows
  - ◆ the user's name and an integer seed  $n$  (e.g.,  $n=1000$ )
  - ◆ the result of  $h^n(p) = h(h(h(\dots h(p)\dots)))$  (computed  $n$  times)
- **Authentication process** of Alice with Bob:
  - ◆ Alice sends her user's name (plain text)
  - ◆ Bob replies the integer  $n$  (plain text)
  - ◆ Alice sends  $h^{n-1}(p)$
  - ◆ Bob computes  $h(h^{n-1}(p))$  and compares it to the stored value  $h^n(p)$
  - ◆ Bob deletes  $h^n(p)$  and replaces it with the new value  $h^{n-1}(p)$
  - ◆ next time, Bob will send the integer  $n-1$

# Security Threats of the Lamport's Scheme

## Two Authentication Servers?



- Alice wants to be authenticated by two servers Bob and Fred
  - Fred sends  $n+1$  and Bob sends  $n$
  - Alice sends  $h^n(p)$  to Fred and  $h^{n-1}(p)$  to Bob
- Oscar may intercept  $h^{n-1}(p)$  and, after Fred has sent  $n+1$  and Alice has replied  $h^n(p)$ , use it to get authentication from Fred as Alice



- A **different public salt** (seed)  $s$  is assigned to each server, stored on both server and user
  - the server stores  $h^n(p||s)$
  - Alice sends  $h^{n-1}(p||s)$



*Hello Bob. I am Alice!*

*Hello Alice.  $n=89$*

*OK Bob:  $h^{88}(p)$*



*Hello Fred. I am Alice!*

*Hello Alice.  $n=89$*

*?????*



# Security Threats of the Lamport's Scheme

## Small- $n$ Attack

---



- After  $n$  authentications, the system has to be re-initialized
- Small- $n$  attack: Oscar impersonates Bob
  - ◆ Oscar waits for Alice to request authentication
  - ◆ Oscar sends Alice a small  $n$  (e.g., 50)
  - ◆ Alice sends  $h^{49}(p)$
- If the real Bob is storing a large  $n > 50$ , Oscar can now get authentication as Alice  $n-50$  times

# What You Have

## Security Token

---



- Hardware device to generate OTPs
  - ◆ *time-based OTP token (synchronous token)*: the OTP is computed based on the current time and is valid in the current time window (need of time synchronization between tokens and server)
  - ◆ *counter token*: the OTP is computed based on a counter, incremented each time (need to keep tokens aligned on the server)

# Two-Factor Security Token

---

- Only the legitimate owner can use the token
  - ◆ *what you have* (token device) + *what you know* (PIN)
- Different alternative approaches to incorporate a PIN in the OTP
  - ◆ Alice gives both the PIN and the OTP
    - the server verifies both the PIN and the OTP
  - ◆ the PIN is incorporated in the OTP computation
    - the OTP is not valid, if the correct PIN has not been given to the token
  - ◆ the token device requires a PIN to run the algorithm and generate an OTP

# Outline

---

- General concepts of authentication
- Username and password
- Challenge and response
- One-Time Password
- **Biometrics**
- Kerberos

# What You Are / What You Do

## Biometric Patterns

---



- A way to obviate the weakness of users
  - ◆ passwords can be disclosed or guessed easily, tokens can be lost
- Various possible applications
  - ◆ *authentication* of a user (search one-to-one)
    - is the user really who declares to be?
  - ◆ *identification* of a user (search one-to-many)
    - given a sample, can we associate it to a single person or to a small group?
    - the database is assumed *complete*
  - ◆ *uniqueness* (is the user present in the current database?)
    - given a sample, can we determine whether the owner is already registered?
    - the database is not assumed *complete*
- *What you are or what you do?*
  - ◆ static biometrics: physical characteristics
  - ◆ dynamic biometrics: behavior patterns
- Search in the database by *closest match*

- Fingerprint
  - ◆ unique pattern, different also for homozygous twins
  - ◆ advances: 3D acquisition, electrical measurement, thermal fingerprint
- Hand geometry
  - ◆ 3D acquisition and measurement of shape, finger length,...
- Retina or iris
  - ◆ image acquisition with a specific camera
  - ◆ unique pattern, different also for homozygous twins
- Face
  - ◆ image acquisition and geometric pattern identification
  - ◆ homozygous twins are not discriminated
- Others
  - ◆ thermogram, ear image, odor analysis, DNA,...

- Speech recognition
  - ◆ identification of a vocal "fingerprint"
  - ◆ can be deceived using high-quality recordings
- Signature dynamics (on a tablet)
  - ◆ analysis of the image and of the variations of pen pressure and speed
- Typing speed pattern
  - ◆ analysis of the time series of keyboard hits

# Accuracy of Biometric Authentication

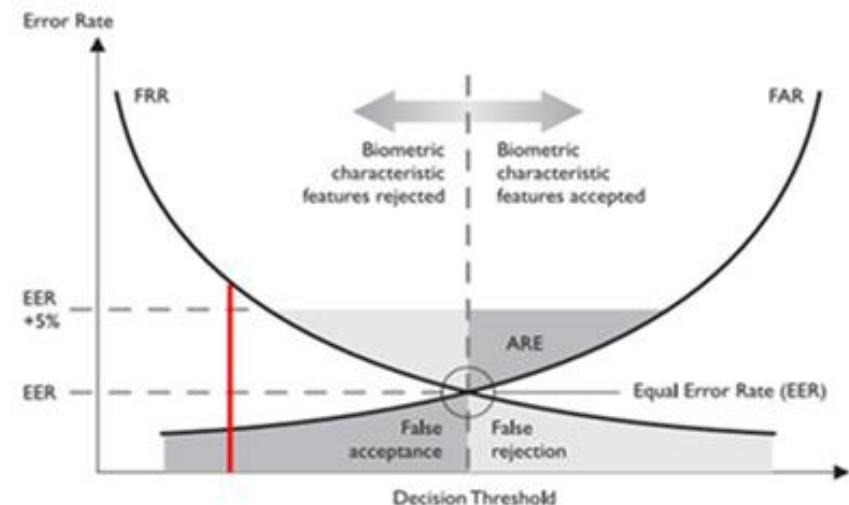
## ■ *False Acceptance Rate (FAR)*

- ◆ measure of the probability that the system will **incorrectly accept** an unauthorized user (impostor)
- ◆  $(\text{number of unauthorized users that are incorrectly accepted}) / (\text{total number of attempts by unauthorized users})$

## ■ *False Rejection Rate (FRR)*

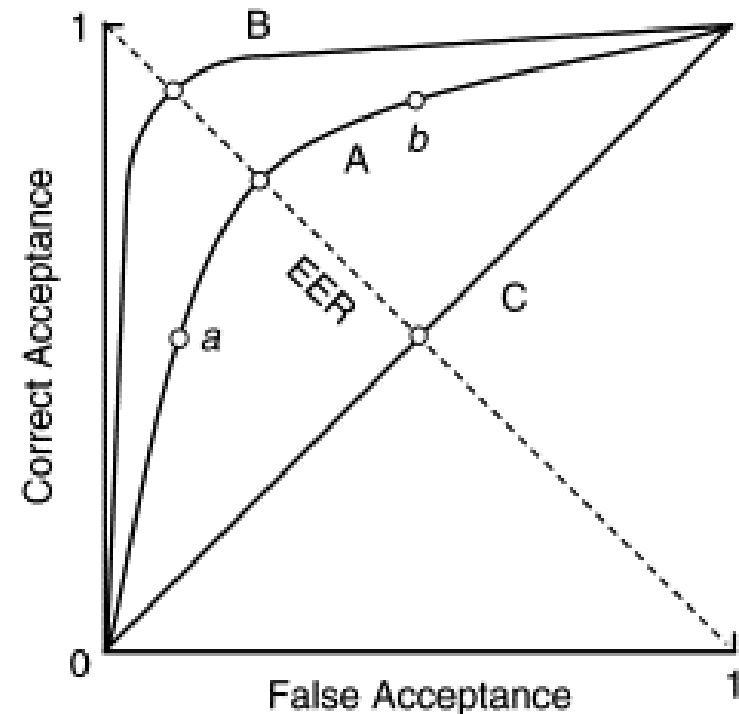
- ◆ measure of the probability that the system will **incorrectly reject** an authorized (legitimate) user
- ◆  $(\text{number of authorized users that are incorrectly rejected}) / (\text{total number of attempts by authorized users})$

## ■ *Equal Error Rate (EER):* $\text{FAR} = \text{FRR}$



# Receiver Operating Characteristic (ROC)

- First defined for speech recognition
  - FAR vs. FRR over a range of decision threshold values
- 
- B performs better than A
  - C: purely random decision
- 
- a: strict decision criterion
  - b: lax decision criterion



# Performance of Biometric Authentication

---

- FAR and FFR performance achieved by current biometric systems is still not adequate to allow its adoption as single authentication factor
- Weak points of *single-factor biometric systems*
  - ◆ noise and impairments in acquired data (e.g., dirty sensor or finger)
  - ◆ intra-class variation (e.g., variable positioning vs. the camera)
  - ◆ inter-class similitude (e.g., persons with very similar faces)
  - ◆ non-universality (not all individuals present adequate quality of the unique features, e.g. damaged fingers)
  - ◆ spoofing (e.g., fingerprint mould)
- *Multiple-factor biometric systems* are under study

# Outline

---

- General concepts of authentication
- Username and password
- Challenge and response
- One-Time Password
- Biometrics
- **Kerberos**

# Kerberos

---

- Developed at MIT (Athena Project, 1989)
- Provides a centralized service for *key distribution* and *indirect authentication* in an open, untrusted and distributed environment
  - ◆ users at workstations access services on servers throughout the network
  - ◆ servers restrict access to authorized users and authenticate requests for specific services
    - malicious users can gain access on legitimate workstations, spoof IP address
    - attackers can read, modify, add packets in the protocol traffic
  - ◆ based on the Needham-Schroeder protocol with trusted third-party *Authentication Server*
  - ◆ based exclusively on *symmetric cryptography* (no public key)
  - ◆ requires clock *time synchronization* (e.g., by NTP)
- Two versions are in use
  - ◆ Kerberos v4 (Athena Project, 1989)
  - ◆ Kerberos v5 (Internet Standard RFC 4120)

# A Simple Authentication Dialogue

- Open untrusted environment
  - ◆ anyone can apply to any server for any service
  - ◆ to counteract *impersonation*, an Authentication Server (AS) stores passwords of all users in a central database and knows a symmetric key with each server

- The user C requests access to server S

**C→AS:**  $ID_C || P_C || ID_S$  (plain text!)

**AS→C:** Ticket (AS has checked identity of C and if C is allowed on S)

**C→S:**  $ID_C || Ticket$

$Ticket = \{ ID_C || AD_C || ID_S \}_{K_{S-AS}}$

$ID_C, P_C, ID_S$ : identifier and password of user on client C or of server S

$AD_C$ : network address of user on client C

$K_{S-AS}$ : symmetric key shared by S and AS

# Improving the Simple Authentication Dialogue



- Unsolved problems in this simple authentication dialogue
  - ◆ password is supplied as plain text
  - ◆ each time a user requests a ticket must supply the password
    - tickets should be reusable throughout logon sessions
    - however a user needs a ticket for every different service
- The simple scheme has to be improved
  - ◆ password not transmitted as plain text
  - ◆ a Ticket-Granting Server (TGS) is introduced to distribute tickets to access each service
  - ◆ tickets can be used repeatedly
  - ◆ two types of tickets
    - Ticket Granting Ticket (TGT), which allows requesting a regular Ticket
    - regular Service Ticket (ST), which allows to access a certain service

# A More Secure Authentication Dialogue

- Once per user logon session

**C**→**AS**:  $ID_C \parallel ID_{TGS}$

**AS**→**C**:  $\{Ticket_{TGS}\}_{K_{C-AS}}$

- Once per type of service

**C**→**TGS**:  $ID_C \parallel ID_S \parallel Ticket_{TGS}$

**TGS**→**C**:  $Ticket_S$

- Once per service session

**C**→**S**:  $ID_C \parallel Ticket_S$

$Ticket_{TGS} = \{ID_C \parallel AD_C \parallel ID_{TGS} \parallel TS_1 \parallel Lifetime_1\}_{K_{TGS-AS}}$

$Ticket_S = \{ID_C \parallel AD_C \parallel ID_S \parallel TS_2 \parallel Lifetime_2\}_{K_{S-TGS}}$

$ID_{TGS}$ , identifier of the Ticket-Granting Server

$K_{C-AS}$ : symmetric key of C and AS, derived from the password of C

$K_{TGS-AS}$ : symmetric key of TGS and AS

$K_{S-AS}$ : symmetric key of S and AS       $TS$ : time stamp

# Yet Improving the Authentication Dialogue

---

- If Lifetimes are short → the user must supply password often
- If Lifetimes are long → replay attacks
  - ◆  $Ticket_{TGS}$  can be eavesdropped and replayed to request  $Ticket_S$  impersonating the legitimate user
  - ◆  $Ticket_S$  can be eavesdropped and replayed to request service impersonating the legitimate user
  - ◆ simple IP spoofing is enough to impersonate the user
- Mutual authentication is needed
  - ◆ to authenticate users to the server → no replay attack by false user
  - ◆ to authenticate the server to users → no false server

# Summary of Kerberos v4 Dialogue



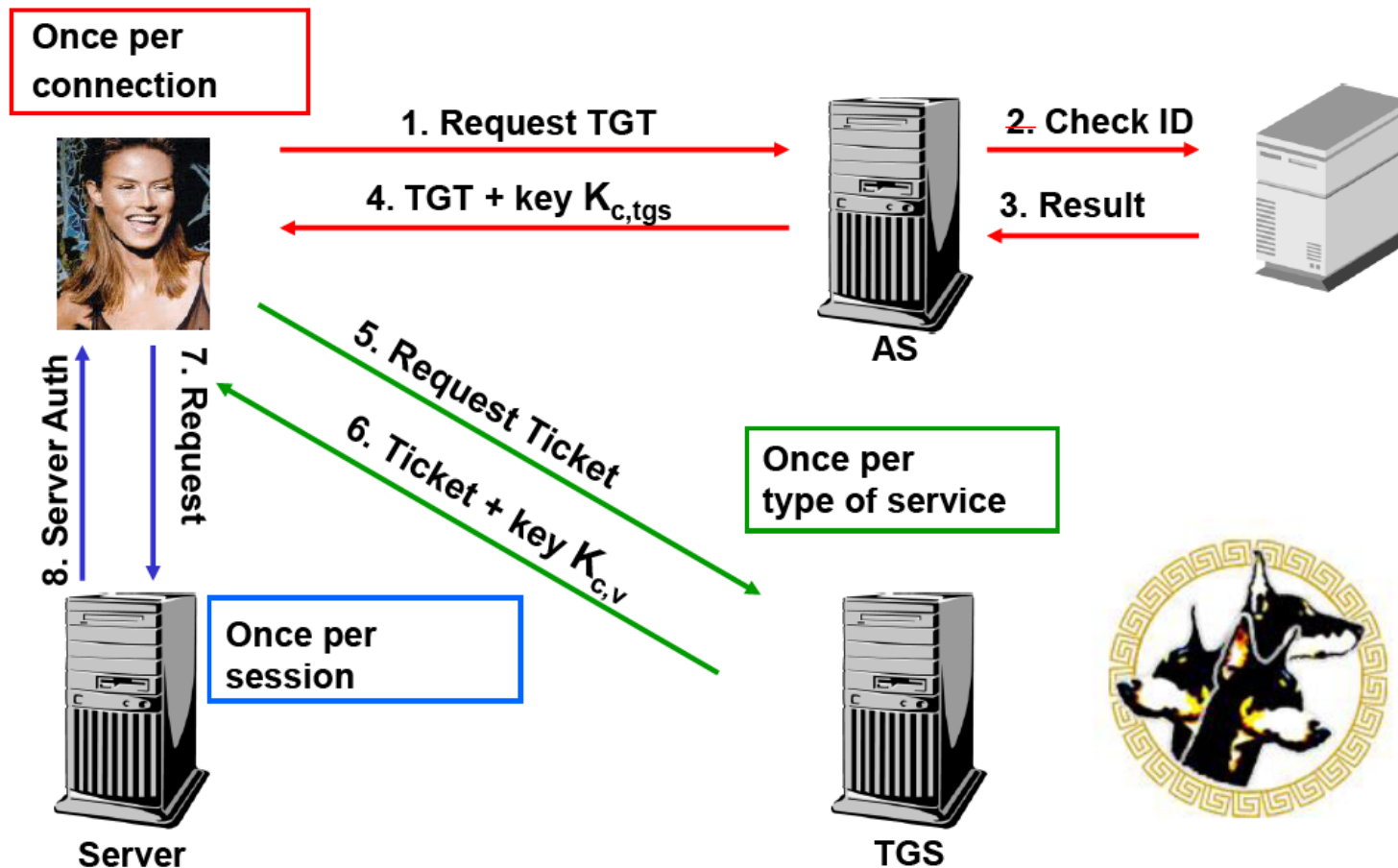
- Authentication Service Exchange to get the Ticket-Granting Ticket
  - 1) **C**→**AS**:  $ID_C \parallel ID_{TGS} \parallel TS_1$
  - 2) **AS**→**C**:  $\{ K_{C-TGS} \parallel ID_{TGS} \parallel TS_2 \parallel Lifetime_2 \parallel Ticket_{TGS} \}_{K_{C-AS}}$
- TG Service Exchange to get the Service Ticket
  - 3) **C**→**TGS**:  $ID_S \parallel Ticket_{TGS} \parallel Authenticator_{C1}$
  - 4) **TGS**→**C**:  $\{ K_{C-S} \parallel ID_S \parallel TS_4 \parallel Ticket_S \}_{K_{C-TGS}}$
- Client/Server Authentication Exchange to get service
  - 5) **C**→**S**:  $Ticket_S \parallel Authenticator_{C2}$
  - 6) **S**→**C**:  $\{ TS_{5+1} \}_{K_{C-S}}$

$$Ticket_{TGS} = \{ K_{C-TGS} \parallel ID_C \parallel AD_C \parallel ID_{TGS} \parallel TS_2 \parallel Lifetime_2 \}_{K_{TGS-AS}}$$

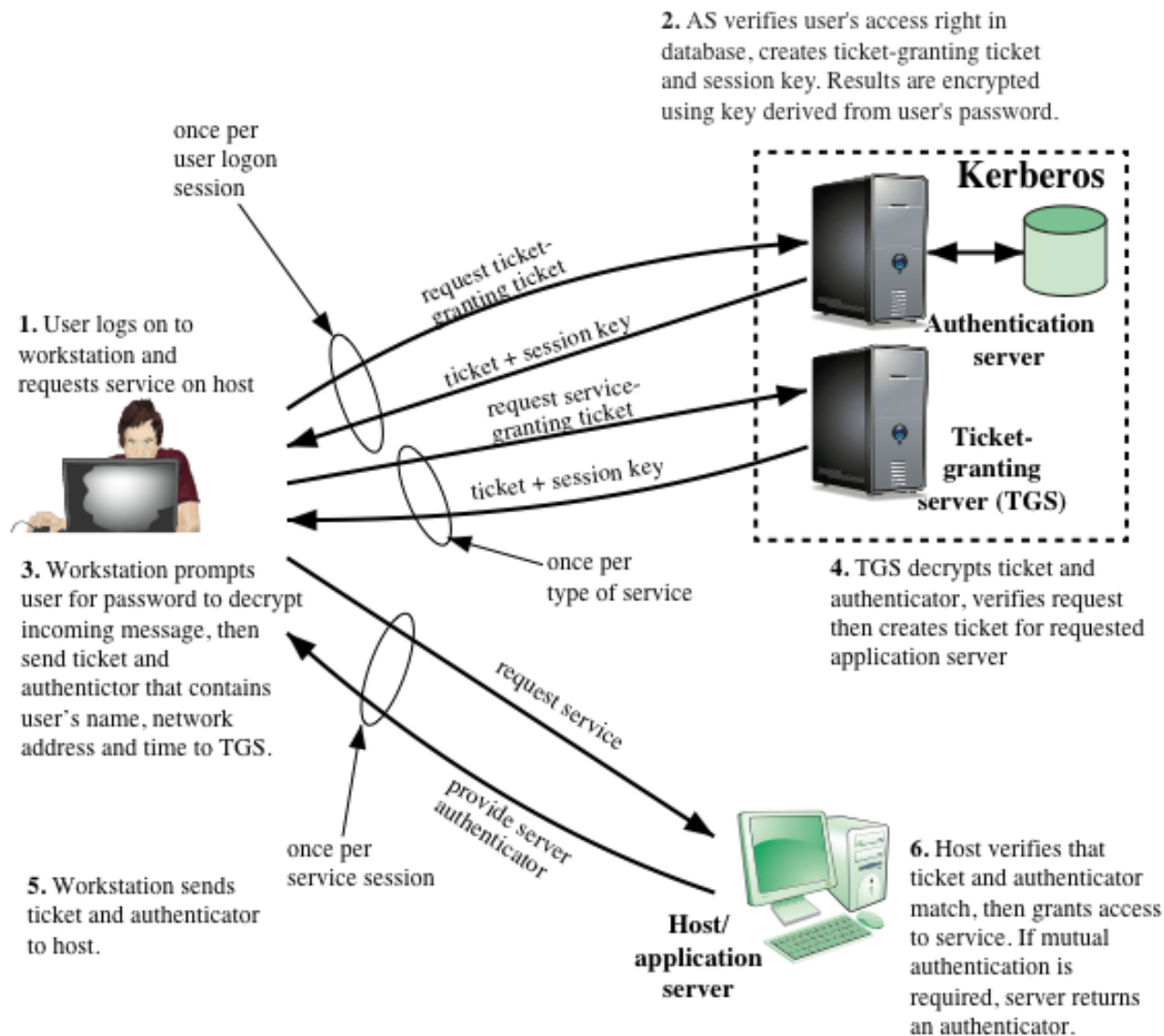
$$Ticket_S = \{ K_{C-S} \parallel ID_C \parallel AD_C \parallel ID_S \parallel TS_4 \parallel Lifetime_4 \}_{K_{S-TGS}}$$

$$Authenticator_{C1} = \{ ID_C \parallel AD_C \parallel TS_3 \}_{K_{C-TGS}}$$

$$Authenticator_{C2} = \{ ID_C \parallel AD_C \parallel TS_5 \}_{K_{C-S}}$$



## Overview of Kerberos v4 Dialogue



- A *Kerberos Realm* consists of one Kerberos server, a few clients, a few application servers
  - ◆ i.e., a set of managed nodes that share the same Kerberos database, which resides on the Kerberos master computer (physically secured)
- The Kerberos server
  - ◆ knows the user ID and hashed passwords of all (registered) users
  - ◆ shares a secret key with each (registered) application server
- Networks under different administrations (different policies) typically belong to distinct realms
- Users in one realm may need access to servers in other realms
  - ◆ procedure for inter-realm authentication

# Limitations and Improvements (1)

---

- Dependence on encryption system
  - ◆ v4 requires DES encryption → v5 may use any encryption system
- Dependence on IP
  - ◆ v4 requires IP addresses → v5 may use any network addressing system
- Byte ordering of messages
  - ◆ v4 uses a variable byte ordering scheme → v5 uses a single standard (Abstract Syntax Notation 1 and Basic Encoding Rules)
- Ticket lifetime
  - ◆ v4 encodes lifetime as 8 bit in 5' units ( $\max 256 \times 5' = 21h$ )  
→ v5 specifies explicit start and end times
- Authentication forwarding
  - ◆ v5 allows a client to access a server, and this server to access another server on behalf of the client (e.g., a print server accesses a file server)
- Inter-Real authentication improved

# Limitations and Improvements (2)

---



- Double encryption in messages 2 and 4 is avoided
- Non-standard PCBC DES mode is replaced by standard CBC DES
  - ◆ Propagating Cipher Block Chaining was adopted to include integrity check besides encryption, but it was proved to be vulnerable
- Sub-session keys are introduced in v5 to avoid repeated use of the same session key for many connections
- Both v4 and v5 are somehow vulnerable to offline password attacks (brute force or dictionary attacks)