

ISA 666

Internet Security Protocols

Authentication

Kerberos

Authentication

- Any process through which one proves or verifies certain information.
- Example: user authentication
 - a user proves that s(he) has the claimed characteristics
 - i.e. processes a claimed set of attributes.

Identification

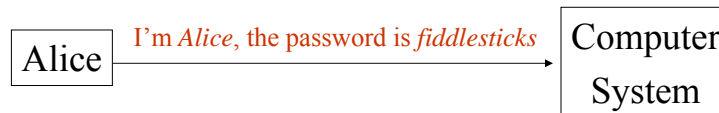
- **Identification** is a process through which one ascertains the identity of another person or entity.
- Identification requires establishing the **uniqueness** of an entity
- Authentication **does not always require uniqueness**.
- Authentication and identification are different.
 - Identification requires that the verifier check the information presented against all the entities it knows about
 - Authentication requires that the information be checked for a single, previously identified, entity.

Authentication mechanisms

- **Knowledge:**
 - What the claimant knows
 - Example: passwords, mothers maiden name
- **Physical Possesions**
 - What the user possesses
 - Example: a physical key, a ticket, a passport, a token, a smart card
- **Physical attributes**
 - What the user is biometrically
 - fingerprints, voiceprint, signature dynamics

Password Based Authentication

- Most commonly used method.

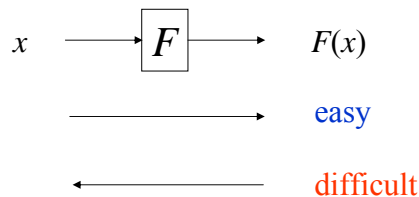


Storing User Passwords

- Directly Store the Passwords?
 - Not a good idea!
 - High risk
 - Have to trust the system administrator?
 - Anyone who captures the password database could impersonate all the users.
 - The password database would be very attractive to hackers.

One-Way Hash Function

- One-way hash function F
 - $F(x)$ is easy to compute
 - From $F(x)$, x is difficult to compute
 - Example:
 - $F(x) = g^x \bmod p$, where p is a large prime number and g is a primitive root of p .



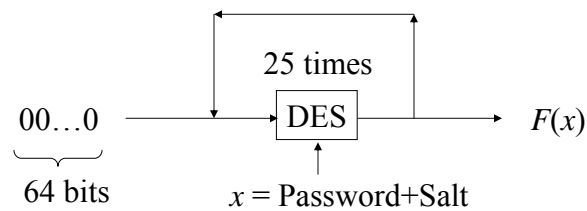
Storing Passwords

- For each user, system stores
(user name, $F(\text{password})$)
in a password file, where F is a one-way hash function
- When a user enters the password, system computes $F(\text{password})$; a match provides proof of identity
- Implications
 - There could be $\text{password}' \neq \text{password}$ such that $F(\text{password}) = F(\text{password}')$

What is F ?

- **crypt Algorithm (Unix)**

- Designed by Bob Morris and Ken Thompson
- Use Data Encryption Standard (DES) encryption algorithm
- User password and salt is used as the encryption key to encrypt a 64-bit block of zeros
- This process is repeated 25 times



Choice of Passwords

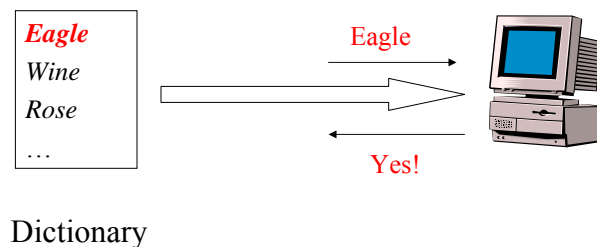
- Suppose passwords can be from 1 to 9 characters in length
- Possible choices for passwords = $26^1 + 26^2 + \dots + 26^9$
 $= 5 * 10^{12}$
- At the rate of 1 password per millisecond, it will take on the order of 150 years to test all passwords

Choice of Passwords (Cont'd)

- However, we don't need to try all possible passwords, only the probable passwords
- In a Bell Labs study (Morris & Thompson 1979), 3,289 passwords were examined
 - 15 single ASCII characters, 72 two ASCII characters, 464 three ASCII characters, 477 four alphanumeric character, 706 five letters(all lower or all upper case), 605 six letters all lower case, 492 weak passwords (dictionary words spelled backwards, first names, surnames, etc.)
 - **Summary: 2,831 passwords (86% of the sample) were weak,**
i.e., they were either too easily predictable or too short

Dictionary Attacks

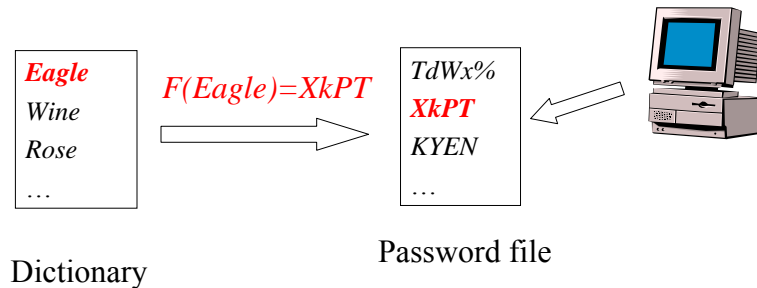
- **Attack 1:**
 - Create a dictionary of common words and names and their simple transformations
 - Use these to guess the password



Dictionary Attacks (Cont'd)

- **Attack 2:**

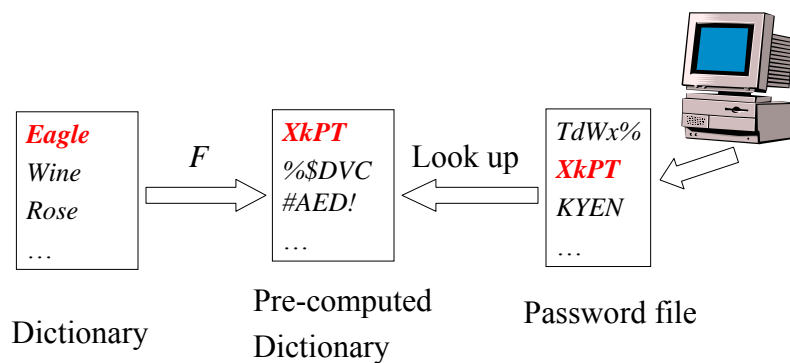
- Usually F is public and so is the password file
 - In Unix, F is encrypted, and the password file is `/etc/passwd`.
- Compute $F(\text{word})$ for each word in the dictionary
- A match gives the password



Dictionary Attacks (Cont'd)

- **Attack 3:**

- To speed up search, pre-compute $F(\text{dictionary})$
- A simple look up gives the password

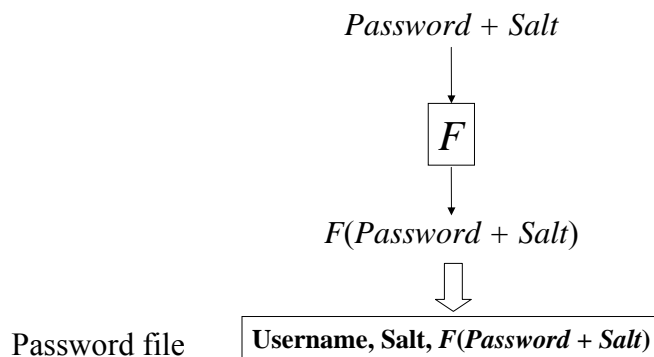


Password Salt

- To make the dictionary attack a bit more difficult
- Salt is a 12-bit number between 0 and 4095
- Derived from the system clock and the process identifier
- It does NOT make it any harder to guess any user's password!
- It does make it impossible to check a group of hashed passwords with single round of hashing a dictionary.

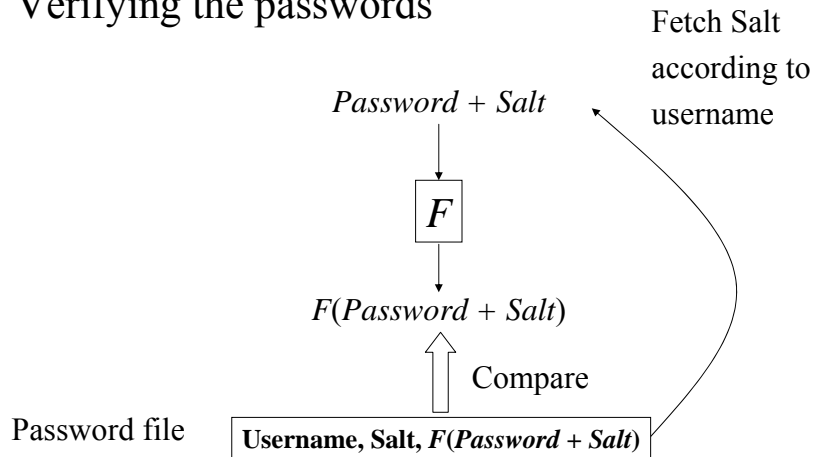
Password Salt (Cont'd)

- Storing the passwords



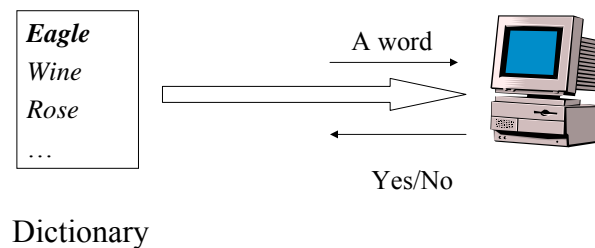
Password Salt (Cont'd)

- Verifying the passwords



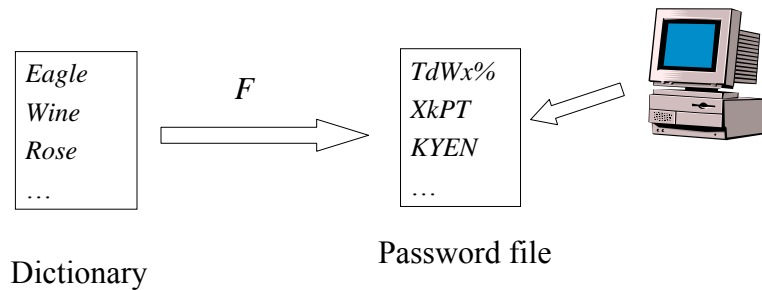
Does the Password Salt Help?

- Attack 1?
 - Without Salt
 - With Salt



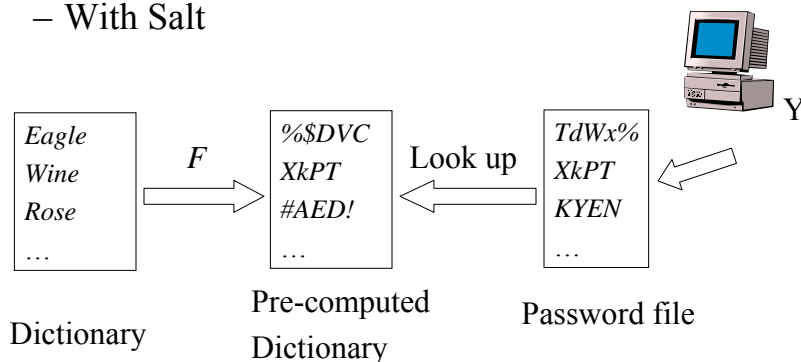
Does Password Salt Help?

- Attack 2?
 - Without Salt
 - With Salt



Does Password Salt Help?

- Attack 3?
 - Without Salt
 - With Salt



Password Management Policy and Procedure

- Educate users to make better choices
 - Does not work if the user population is large or novice
- Define rules for good password selection and ask users to follow them
 - Rules serve as guideline for attackers
- Ask or force users to change their passwords periodically
- Force users to use machine generated passwords
 - Random passwords are difficult to memorize; also password generator may become known to the attacker through analysis
- Actively attempt to break users' passwords; force users to change those that are broken
 - Attacker may have better dictionary
- Screen password choices; if a choice is weak, force users to make a different choice

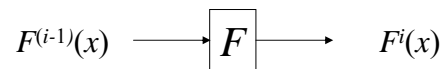
One-time Passwords

- Randomly generate different passwords, and
Use the password exactly once!

S/Key (Cont'd)

- Authentication

- The first time, the user supplies $F^{(n-1)}(x)$.
- The system checks if $F(F^{(n-1)}(x))=F^n(x)$. If yes, the user is authenticated and the system replaces $F^n(x)$ with $F^{(n-1)}(x)$.
- The second time, the user supplies $F^{(n-2)}(x)$.
- The third time, ...



S/Key (Cont'd)

- Enhancement

- Add salt $F^n(x|\text{salt})$
- Each server will have different salt

- Limitations

- No mutual authentication
 - Only authenticate client to the server
 - No authentication of server
 - Subject to server spoofing – **man in the middle attack!**

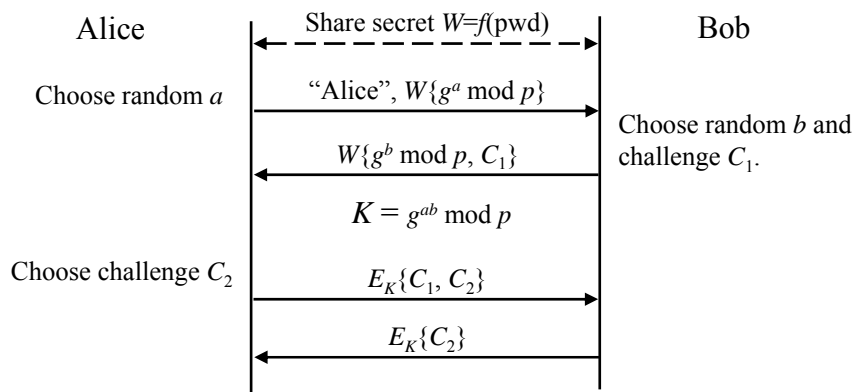
- Small n attack

- Trudy pretends to be Bob, and responds with small n '
- Alice will send Trudy $F^n(x|\text{salt})$

Strong Password Protocols

- Goals – prevent offline dictionary attack!
 - The following attacks will NOT gain useful information for an off-line dictionary attack
 - Eavesdropping of exchanged messages
 - Impersonating either end

Encrypted Key Exchange (EKE)

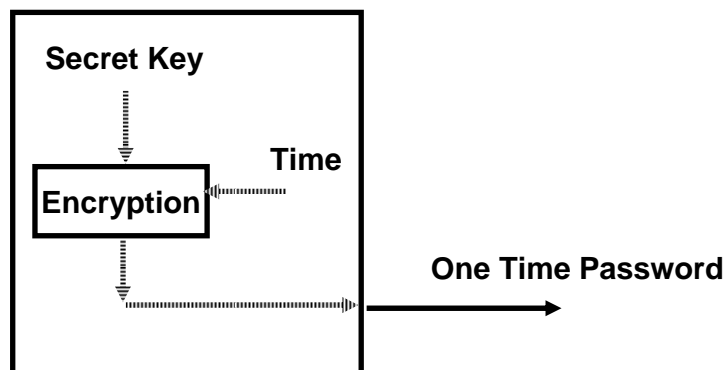


$W = \text{weak secret} = \text{hash of Alice's password}$

Time Synchronized

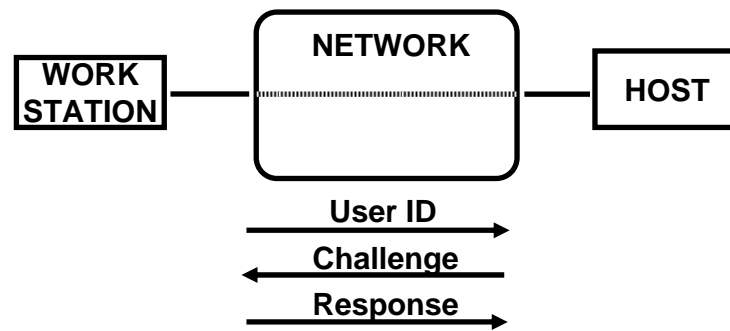
- There is a hand-held authenticator
 - It contains an internal clock, a secret key, and a display
 - Display outputs a function of the current time and the key
 - It changes about once per minute
- User supplies the user id and the display value
- Host uses the secret key, the function, and its clock to calculate the expected input
- The login is valid if the values match
- In practice, the clock skew is a problem

Time Synchronized (Cont'd)

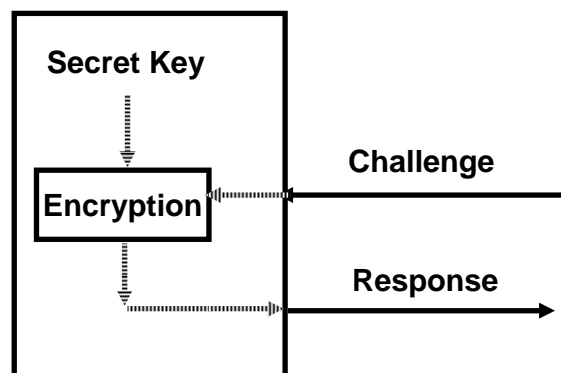


Challenge Response

- A non-repeating challenge from the host instead of a clock is used
- Note that the device requires a keypad.



Challenge Response (Cont'd)



Challenge Response (Cont'd)

- Problems with challenge/response schemes
 - Key database is extremely sensitive
 - This can be avoided if public key algorithms are used;
 - However, the outputs would be too long for users to input conveniently

Biometrics

- Fingerprint
- Retina scan
- Iris scan
- Handprint
- Voice pattern
- Signature
- Keystroke timing

Biometrics (Cont'd)

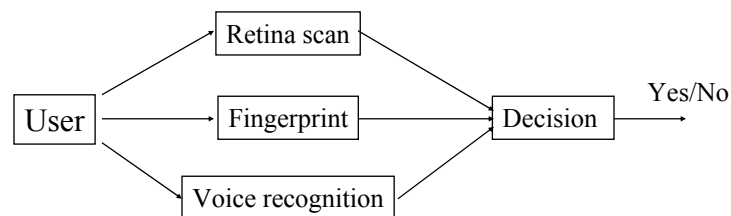
Technique	Description	Min. Cost	False Reading
Retina	Eyes scanned 1 to 2 inches from screening device	\$2,400	1/10,000,000+
Iris	Camera image of eye takes from 14 inches	\$3,500	1/131,000
Handprint	Hand scanned on plate by three video cameras at different angles	\$2,150	1/500
Fingerprint	Finger scanned on glass plate	\$1,995	1/500
Signature	Written with special pen on digitizer tablet	\$1,000	1/50
Voice	Predefined phrase spoken into telephone or microphone	\$1,500	1/50

Effectiveness of Biometrics

- Two types of errors for authentication
 - False acceptance (FA)
 - Let imposters in
 - **FAR: the probability that an imposter is authenticated.**
 - False rejection (FR)
 - Keep authorized users out
 - **FRR: the probability that an authorized user is rejected.**
- Another type of error for identification
 - False match (FM)
 - One user is mistaken for another (legitimate user)
 - **FMR: the probability that a user is incorrectly matched to a different user's profile.**
- No technique is perfect!

Multimodal Biometrics

- Use multiple Biometrics together.
 - AND: Accept only when all are passed
 - Why do we need this?
 - OR: Accept as long as at least one is passed
 - Why do we need this?
 - Others



Summary

- Password authentication
 - Storing passwords
 - Dictionary attacks
 - Password Salt
- One-time passwords
 - S/Key
 - Time synchronized
 - Challenge response
- Biometrics

Distributed Systems Security

Authentication Problem In Distributed Systems

- Assume an open distributed environment in which users at workstations wish to access services on servers distributed throughout the network.
- Restrict access to authorized users and to be able to authenticate requests for service.
- Prevent server spoofing

Other Technical Issues

- Usually uses a client-server paradigm for resource usage
- The clients need to know how to ask for services
 - Service discovery (protocols)
 - Universal Description, Discovery, and Integration UDDI:
 - A Web service standard to discover and access services in a secure manner
- Mutual authentication based on trust
 - Trust negotiation: some challenge response schema
- Name and identity translation issues

Threats to Authentication Algorithms in a Distributed Environment

- Some threats:
 - Subject Impersonation:** A mal actor may gain access to a particular workstation and pretend to be another user operating from that workstation.
 - Address Impersonation:** A mal actor may alter the network address of a workstation so that the requests sent from the altered workstation appear to come from the impersonated workstation.
 - Password stealing:**
 - Using replay attacks:** A mal actor may eavesdrop on exchanges and use a replay attack to gain entrance to a server or to disrupt operations.
 - Installing Trojans on the server:** Login Trojan horse to steal password
 - Impersonating Servers:** A bogus server may trick client into believing it is the right server
- **One solution:** Kerberos



Kerberose

Chapters 13 and 14

ISA 666

Edited by Duminda Wijesekera.
original: Dr. Xinyuan Wang

43

What Kerberos Provides

- A centralized authentication service
- Authenticate users to services
- Authenticate services to users
- Servers are relieved of the burden of maintaining authentication information
- Supports inter-server authentication

ISE at George Mason University

ISA 666 Edited by Duminda Wijesekera.
original: Dr. Xinyuan Wang

44

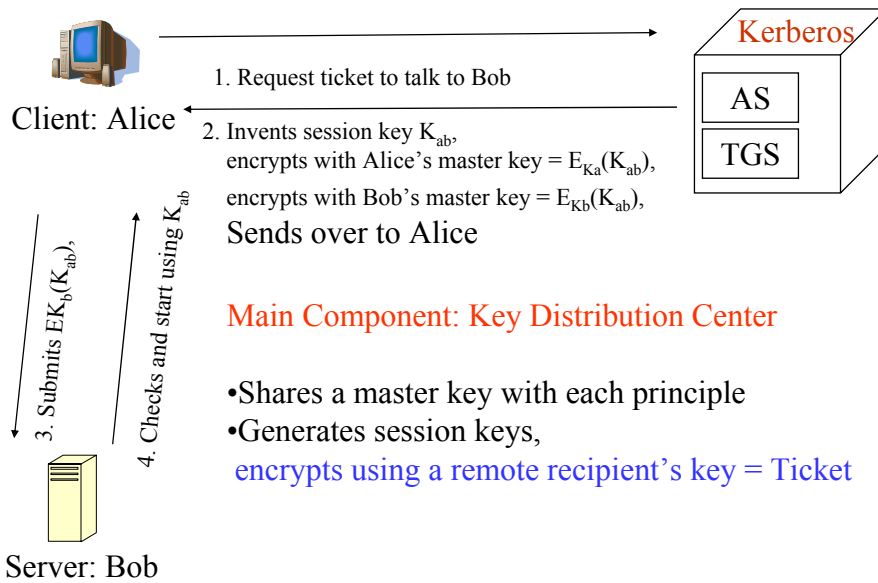
Properties of Kerberos

- Rely exclusively on conventional encryption.
 - Public key based Kerberos has been considered.
- **Stateless**: Kerberos server doesn't need to maintain the state information about any entities being authenticated.

Kerberos Design Requirements

- **Secure**
 - A network eavesdropper should not be able to obtain the necessary to impersonate a user.
- **Reliable**
 - Kerberos should be highly available and should employ a distributed server architecture.
- **Transparent**
 - The user shouldn't be aware that authentication is taking place.
- **Scalable**
 - The system should be capable of supporting large numbers of clients and servers.

Main Components and Interaction Mode



The Kerberos Protocol

- Outline of the introduction to the Kerberos protocol
 - A simple authentication protocol
 - A more secure authentication protocol
 - Kerberos Version 4 authentication protocol

A Simple Authentication Protocol

- Use an authentication server (AS)
- Basic idea: use a **ticket** to authenticate a user to a server.
- Protocol
 1. $C \rightarrow AS$: $ID_C \parallel PW_C \parallel ID_V$
 2. $AS \rightarrow C$: $Ticket_V = E_{K_V}[ID_C \parallel ID_V]$
 3. $C \rightarrow V$: $ID_C \parallel Ticket_V$

A Simple Authentication Protocol (Cont'd)

- Advantages
 - A centralized authentication service
- Weaknesses
 - A user needs to enter a password for every different service.
 - Password is transmitted in plaintext.
 - Subject to replay attack

A More Secure Authentication Protocol

- A new server: **ticket-granting server (TGS)**

- The protocol

- Once per user logon session

1. $C \rightarrow AS: ID_C \parallel ID_{tgs}$

TS=time stamp

2. $AS \rightarrow C: E_{K_C}[Ticket_{tgs}]$

- Once per type of service

3. $C \rightarrow TGS: ID_C \parallel ID_V \parallel Ticket_{tgs}$

4. $TGS \rightarrow C: Ticket_V$

- Once per service session

5. $C \rightarrow V: ID_C \parallel Ticket_V$

- $Ticket_{tgs} = E_{K_{tgs}}[ID_C \parallel ID_{tgs} \parallel TS_1 \parallel lifetime_1]$

- $Ticket_V = E_{K_V}[ID_C \parallel ID_V \parallel TS_2 \parallel lifetime_2]$

A More Secure Authentication Protocol (Cont'd)

- Ticket-granting ticket (TGT): $Ticket_{tgs}$.

- Service-granting ticket: $Ticket_V$.

- Weaknesses

- Replay attack: No authentication of the valid ownership of the tickets.
- No authentication of the servers.

- What are the components in the tickets?

- Why do we have them?

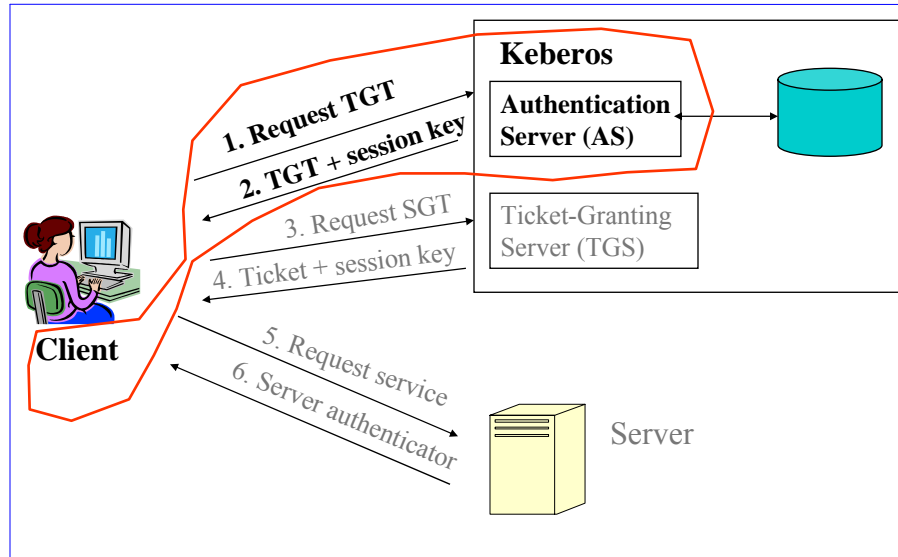
Kerberos Version 4 Protocol

- Basic idea to address the previous weaknesses
 - Session key
 - Authentication of the valid ownership of the tickets
 - Provide authentication of servers.
- Assume authentication server knows
 - Each client's master secret key, which is derived from each client's password

Kerberos Version 4 Protocol (Cont'd)

- Client \leftrightarrow authentication service exchange:
to obtain ticket-granting ticket (TGT)
 1. $C \rightarrow AS$: $ID_C \parallel ID_{tgs} \parallel TS_1$
 2. $AS \rightarrow C$: $E_{K_C}[K_{C,tgs} \parallel ID_{tgs} \parallel TS_2 \parallel Lifetime_2 \parallel Ticket_{tgs}]$
 $Ticket_{tgs} = E_{K_{tgs}}[K_{C,tgs} \parallel ID_C \parallel ID_{tgs} \parallel TS_2 \parallel Lifetime_2]$
 $K_{C,tgs}$ is the session key between C and TGS!

The Client \leftrightarrow AS Interaction



ISE at George Mason University

ISA 666

Edited by Duminda Wijesekera.
original: Dr. Xinyuan Wang

55

Kerberos Version 4 Protocol (Cont'd)

- Client \leftrightarrow ticket granting service exchange:
to obtain service-granting ticket (SGT)

3. $C \rightarrow TGS$: $ID_V \parallel Ticket_{tgs} \parallel Authenticator_c$

4. $TGS \rightarrow C$: $E_{K_{C,tgs}}[K_{C,V} \parallel ID_V \parallel TS_4 \parallel Lifetime_4 \parallel Ticket_V]$

$Ticket_{tgs} = E_{K_{tgs}}[K_{C,tgs} \parallel ID_C \parallel ID_{tgs} \parallel TS_2 \parallel Lifetime_2]$

$Ticket_V = E_{K_V}[K_{C,V} \parallel ID_C \parallel ID_V \parallel TS_4 \parallel Lifetime_4]$

$Authenticator_c = E_{K_{C,tgs}}[ID_C \parallel TS_3]$

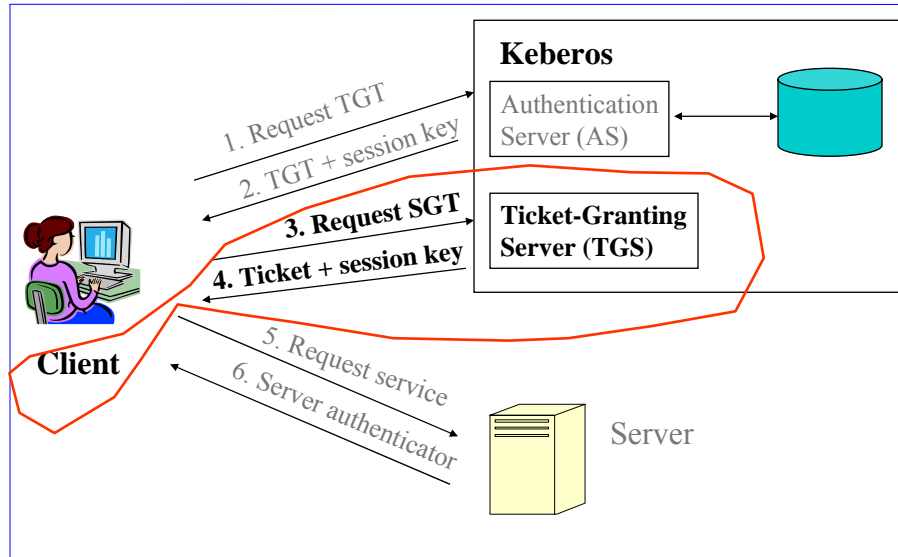
ISE at George Mason University

ISA 666

Edited by Duminda Wijesekera.
original: Dr. Xinyuan Wang

56

Client \leftrightarrow TGS exchange



ISE at George Mason University

ISA 666

Edited by Duminda Wijesekera.
original: Dr. Xinyuan Wang

57

Kerberos Version 4 Protocol (Cont'd)

- Client \leftrightarrow server authentication exchange:
to obtain service

5. C \rightarrow V: Ticket_v || Authenticator_c

6. V \rightarrow C: $E_{K_{C,V}}[TS_5 + 1]$

Where

Ticket_v = $E_{K_{C,V}}[K_{C,V} || ID_C || AD_C || ID_V || TS_4 || Lifetime_4]$

Authenticator_c = $E_{K_{C,V}}[ID_C || AD_C || TS_5]$

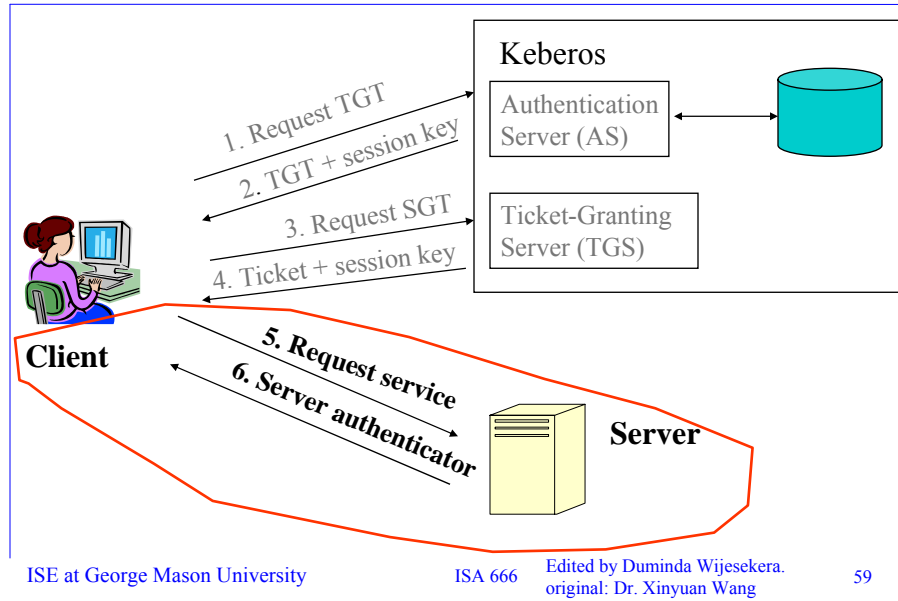
ISE at George Mason University

ISA 666

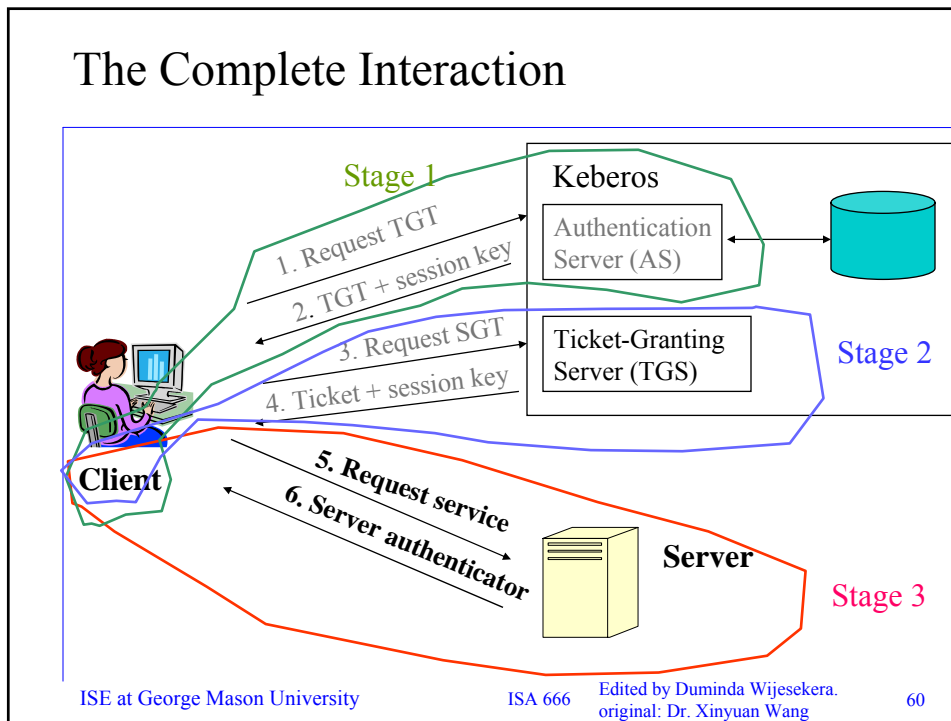
Edited by Duminda Wijesekera.
original: Dr. Xinyuan Wang

58

The Client \leftrightarrow Server Interaction



The Complete Interaction



Kerberos Deployment

- The Kerberos server must have the user ID and hashed password of all participating users in its database.
- The Kerberos server must share a secret key with each server.
- Kerberis are *physically* secured
- Kerberos libraries are distributed on all nodes with users, applications, and other Kerberos-controlled resources

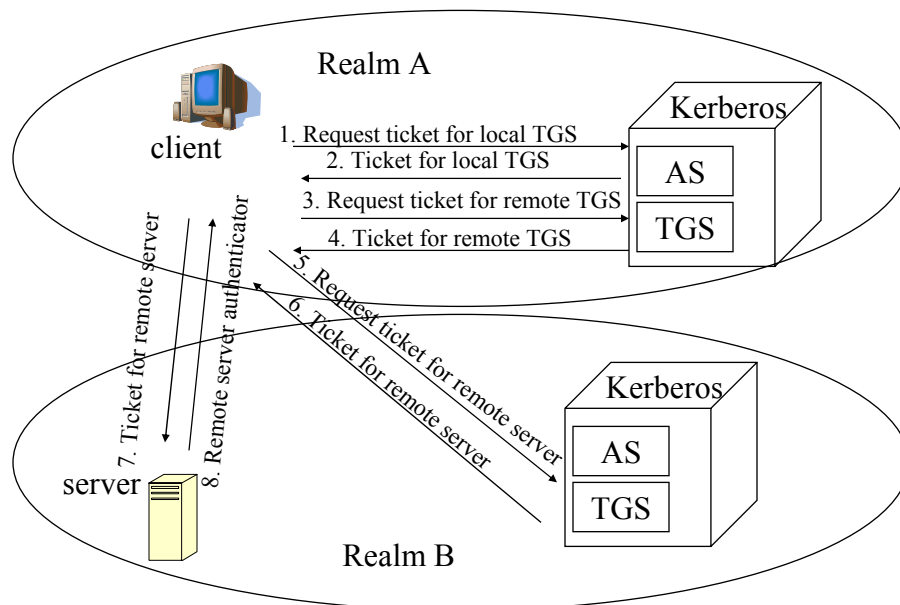
Replicated Kerberos

- Multiple replica of Kerberos:
 - to enhance availability and performance
- Keeping Kerberos databases consistent
 - Single master Kerberos as the point of direct update to principals' database entries
 - Updated database is downloaded from the master to all replica Kerberos
 - Periodic download or on-demand

Kerberos Realms and Multiple Kerberis

- Kerberos realm
 - A full-service Kerberos environment consisting of a Kerberos server, a number of clients, and a number of application servers
- Inter-realm authentication
 - The Kerberos server in each interoperating realm shares a secret key with the server in the other realm. The two Kerberos servers are registered with each other.

Inter-realm Authentication



Race condition in multiple Kerberis

- What happens if some key is updated?
 - An existing ticket may become obsolete since it was encrypted with old keys!
 - Alice's TGT – TGS has updated its master key K_{TGS} !
 - Alice's ticket to Bob – Bob has updated his DES key K_{Bob} !
 - TGS sends Alice her TGT – Alice has updated her password K_{Alice} !
- In Kerberos, TGS and all servers maintain several versions of keys
- What about Alice? Does she maintain several versions of her keys?
 - Small time window that Alice could not login by using her recently updated password

Kerberos V5

Read Chapter 14

Notes on the mid-term examination: 03/06

- Covers all chapters up to, but **excludes Kerberos**
- About 5 to 6 questions:
 - Some computations, some descriptions: **Bring calculators!**
- Chapters 1 and 2: Introduction to basic security
 - MAC and DAC
- Chapters 3 and 4: Secret key Cryptography:
 - AES, DES, 3-DES, Block chaining, hashing, message digests
- Chapter 6: Number theory
 - Modular arithmetic, Primes and Euclid's algorithm
 - Chinese remainder theorem, Euler's totient function

Notes on the mid-term examination – continued..

- Chapter 5: Hash functions
 - MD4, MD5, SHA and HMAC
- Chapter 6: Public key cryptography
 - RSA
 - Finding and factoring big primes
 - Diffie-Hellman, DSS
- Chapters 9, 10, 11, 12: Authentication
 - Using timestamps and nonces, Lamport's algorithm
 - Handshakes and their pitfalls.
 - Needham-Schroder