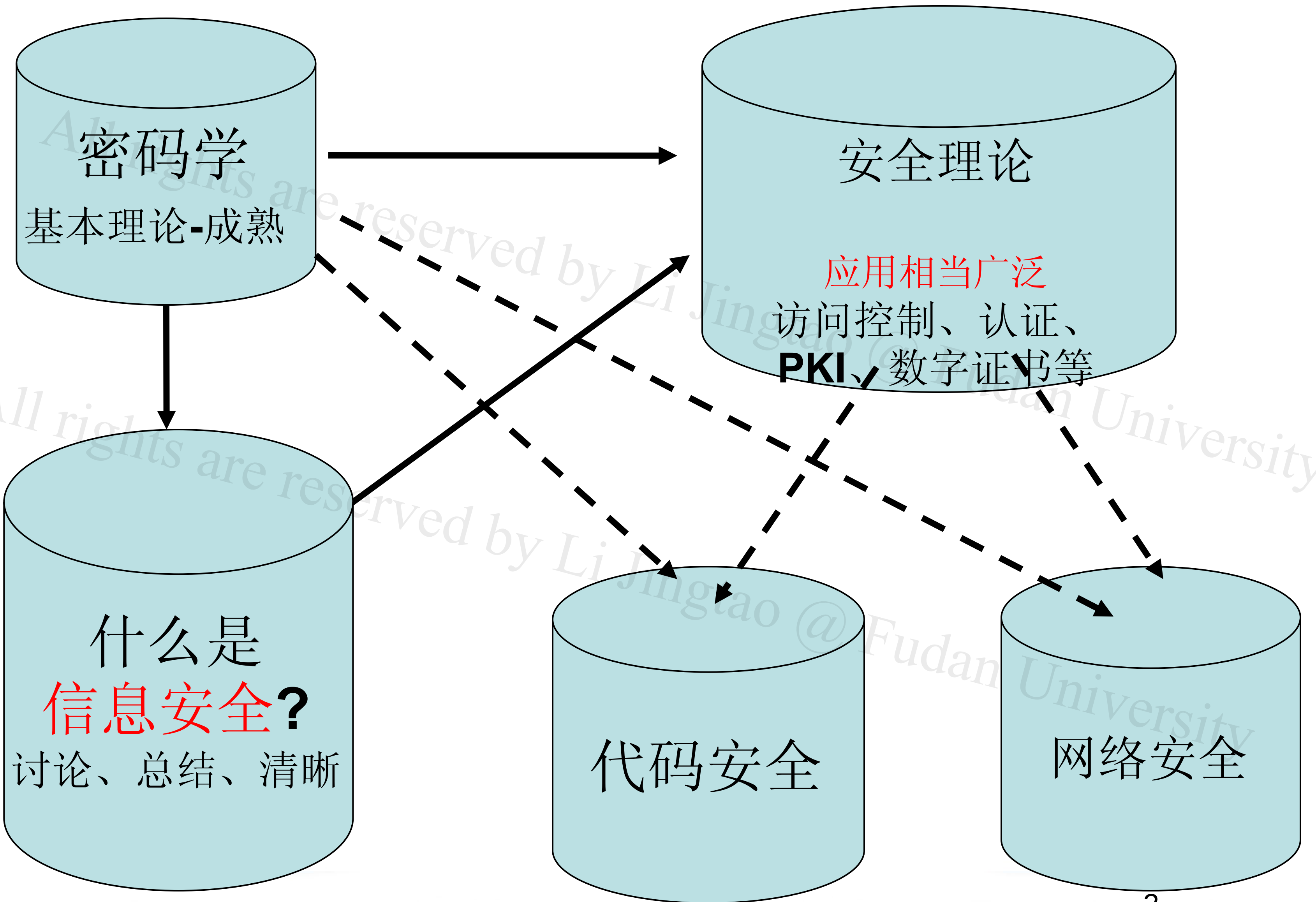
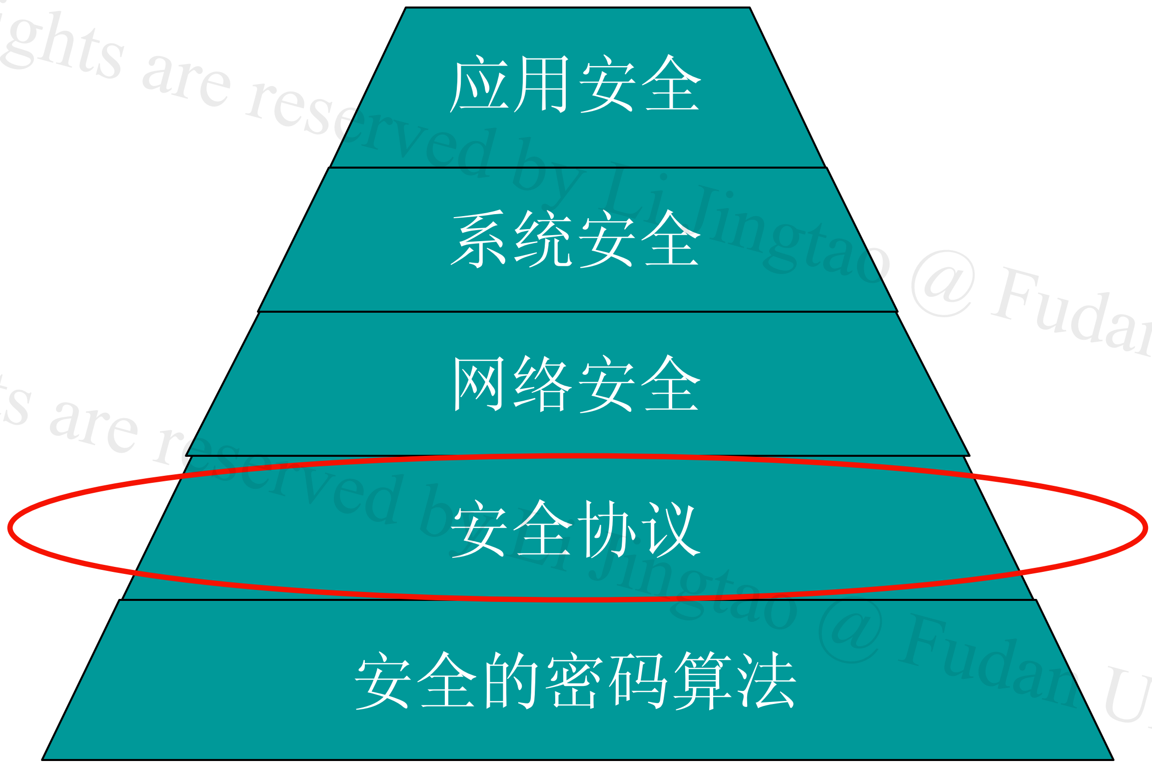
Information Security 09

Authentication
- Basic protocol constructions
- Kerberos

内容间的联系


密码学
基本理论-成熟
安全理论
应用相当广泛
访问控制、认证、
PKI、数字证书等
什么是
信息安全?
讨论、总结、清晰
代码安全
网络安全

Review: 安全层次



Outline of Talk

- Definitions
- Passwords
  - Unix Passwords
  - One time passwords
- Challenge-response techniques
  - Basic protocol constructions
  - Also “one-time”
- Authentication Involving TTP
  - Needham-Schroeder
  - Kerberos

Definitions

Authentication:

- A *claimant* tries to show a *verifier* that the claimant is as declared
  - Identification
  - **Entity Authentication**

Basis of Authentication

- Something *known* - passwords, PINs, keys…
- Something *possessed* - cards, handhelds…
- Something *inherent* - biometrics

Definitions

- Claimant (A): The party that claims a certain identity [and provides evidence of possessing the identity]
  - e.g. through possessing a specific secret

- Verifier (B): The party that verifies the identity of the claimant (accepts or rejects)
  - e.g. through verifying the possession of the secret by claimant

Definitions

- 单向 Unilateral authentication
- 双向 Mutual authentication

Definitions

- Message authentication
  - Data-Origin Authentication
  - Data Integrity
- Entity Authentication

Definitions

- Data-Origin Authentication/Data Integrity
- Entity Authentication
  - Often, a claimed identity in a protocol is a message in its own right. So, confidence about a claimed identity and about the liveness of the claimant can be established by applying data-origin authentication mechanisms.

Authentication scheme

- Weak authentication
  - Passwords, PIN, etc
  - One-time passwords(semi-strong authentication)
- Strong (cryptographic) authentication
  - Challenge – Response Mechanisms
- Zero-knowledge authentication
  - Allow Claimant to demonstrate knowledge of a secret without revealing any information whatsoever of the secret.

Outline of Talk

- Definitions
- Passwords
  - Unix Passwords
  - One time passwords
- Challenge-response techniques

One time passwords

- Avoids *replay attacks*
- ***Shared lists*** - pre-distribute list
- ***Sequentially updated*** - create next password while entering current password
- ***Based on one way functions*** - Lamport's scheme…

Lamport's One Time Passwords

- 1981, by Lamport
- Initialization
  - User has a secret w
  - Using a OWF h, create the password sequence:

$$w, h(w), h(h(w)), \ldots, h^t(w)$$

  - Bob knows only $h^t(w)$
- Authentication：
  - Password for i^{th} identification is:

$$w_i = h^{t-i}(w)$$

S/KEY One-Time Password System

- Based on Lamport's OTP
- Initialization
  - User has a secret: *w*, *seed* (non-secret)
  - Using a OWF *h*, create the password sequence:

$$w,\ h(w, seed),\ h(h(w),\ seed), \ldots, h^t = h(h^{t-1},\ seed)$$

  - Bob server knows: *seed*, Sequence#, h^t

- Authentication:
  - Password for i^{th} identification is:

$$w_i = h^{t-i} = h(w_{i-1},\ seed)$$

使用seed, Sequence#

- 多个server，Password 可重用(使用不同seed即可)
- Server 可发起Challenge:
  – [seed, sequence#]

Attacks on OTPs..

- ***Pre-play attack*** - Eve intercepts an unused password and uses it later
- Make sure you're giving password to the right party
- Bob server must be *authenticated*

Shortcomings of OTPs..

- 使用500-1000次需要Reinitialization
  - 开销不小
- 不支持双向认证
- 保密性没考虑

Outline of Talk

- Definitions
- Passwords
  - Unix Passwords
  - One time passwords
- Challenge-response techniques
  - Basic protocol constructions
  - Also "one-time"
- Authentication Involving TTP

All rights are reserved by Li Jingtao @ Fudan University

Challenge-response authentication

- numerous protocol-based techniques for realizing authentication
- the **basic protocol constructions**, such as *C-R techniques*, in particular those which should be regarded as **good** ones, and the simple technical ideas behind the good constructions, are not so diverse.

- freshness or liveness are the most basic goals

Challenge-response authentication

- Alice is identified by a *secret* she possesses
- *Bob* needs to know that Alice does indeed possess this secret
- *Alice* provides ***response*** to a **time-variant *challenge*** (Nonce, **N**umber used **ONCE**)
- Response depends on ***both*** secret and challenge
  - Transformation
    - Encryption, A knows the key;
    - Simply plus 1, A knows the NOCE
- To defense sniffer attack

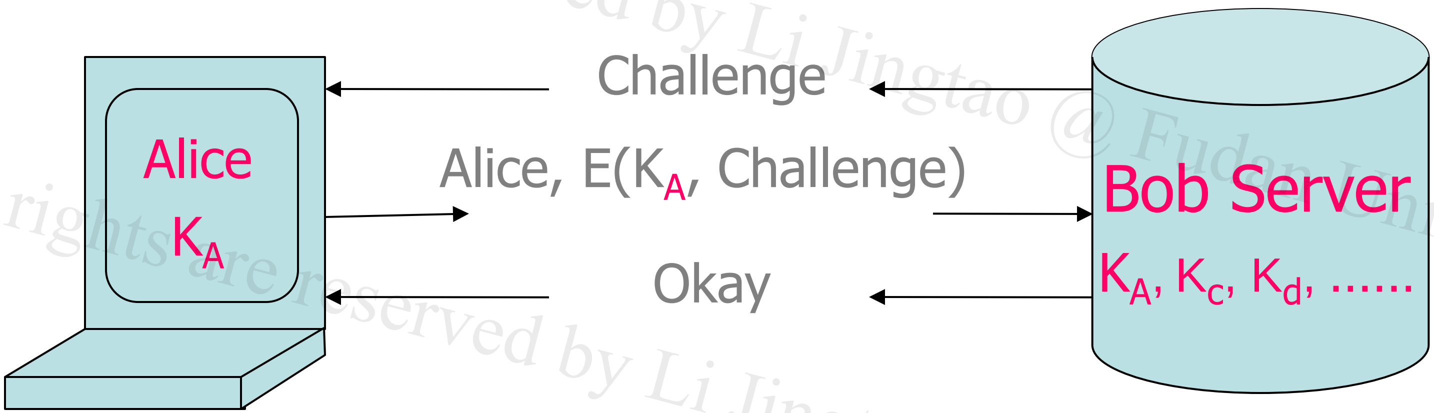
Challenge-response authentication

Using

- Symmetric encryption
- One way functions
- Public key encryption
- Digital signatures

using Symmetric Key Encryption

- Alice and Bob share a key K_A



Unilateral: Using random numbers

- Bob → Alice: r_b
- Alice → Bob: $E_K(r_b, B)$
- Bob checks to see if r_b is the one it sent out
  – Also checks "B" - prevents reflection attack
- r_b must be ***non-repeating, random***
  – prevents replay attack

Reflection attack

- **A reflection attack is a method of attacking a challenge-response authentication system that uses the same protocol in both directions. That is, the same challenge-response protocol is used by each side to authenticate the other side.**

- **Challenge-response reflection attack**
  Where N is a challenge

- B → I(A): N
  I(A) → B: N
  B → I(A): $E_K\{N\}$
  I(A) → B: $E_K\{N\}$

A variation for mechanism

- Bob → Alice: $E_K(r_b, B)$
- Alice → Bob: r_b

- Bob
  - accepts, if returned r_b is correct
  - rejects, otherwise

Unilateral: Using timestamps

- Time-Based Implicit Challenge

- Alice → Bob: $E_K(t_A, B)$
- Bob decrypts and verified that timestamp is OK
- Parameter *B* prevents reflection of same message in B → A direction

认证协议中的常用技术-时间戳(Time-stamp)

- 时间戳
  - A收到一个消息，根据消息中的时间戳信息，判断消息的有效性
    - 如果消息的时间戳与A所知道的当前时间足够接近
  - 这种方法要求不同参与者之间的时钟需要同步
    - 在网络环境中，特别是在分布式网络环境中，时钟同步并不容易做到
    - 一旦时钟同步失败
      - 要么协议不能正常服务，影响可用性(availability)，造成拒绝服务(DOS)
      - 要么放大时钟窗口，造成攻击的机会
  - 时间窗大小的选择应根据消息的时效性来确定

mutual: using random numbers

- Bob → Alice: r_b
- Alice → Bob: $E_K(r_a, r_b, B)$
  - Alice Challenge Bob
- Bob → Alice: $E_K(r_a, r_b)$
- Alice checks that r_a, r_b are the ones used earlier

Shortcomings..

- multiple server, should share different keys
  - – Key Distribution ?
  - – Key management ?

Shortcomings..

- Claimant and verifier required to share a symmetric key
  - A priori key distribution for small, closed systems
  - In larger systems, centralized (on-line) key server required
- Often combined with key agreement (e.g.
  - Needham-Schroeder, Kerberos)
- Assume:
  - prior existence of a shared secret key

Attacks on Authentication Protocols

- An attack consists of an attacker or a coalition of them (Malice) achieving an unentitled gain.
  - a serious one such as Malice obtaining a secret message or key,
  - or a less serious one such as Malice successfully deceiving a principal to establish a wrong belief about a claimed property.
- Authentication protocols are insecure **not because** the underlying cryptographic algorithm they use are weak, **but because** of protocol design flaws.
- usually assume that the underlying cryptographic algorithms are "perfect" without considering their possible weakness.

Conventions

- An honest principal in a protocol does not understand the semantical meanings of any message before a protocol terminates successfully. may make wrong interpretations on protocol messages.
- An honest principal in a protocol cannot recognize a random-looking number (a nonce, a sequence number or a cryptographic key), unless the random-looking number has been created by the principal in the current run of the protocol
- Stateless, does not maintain any state information after a protocol run terminates successfully
- Malice knows the "stupidities" (weaknesses) of honest principals, and will always try to exploit them.

Challenge-response authentication

Using

- Symmetric encryption
- One way functions
- Public key encryption
- Digital signatures

based on keyed OWFs

- Instead of encryption, used keyed MAC h_K
- Check: compute MAC, and check with message
- SKID2 (unilateral), and SKID3(mutual)

Mutual: using keyed MAC – SKID3

- Bob → Alice: r_b
- Alice → Bob: r_a, $h_K(r_a, r_b, B)$
- Bob → Alice: $h_K(r_a, r_b, A)$

Unilateral: using keyed MAC – SKID2

- Bob → Alice: r_b
- Alice → Bob: r_a, $h_K(r_a, r_b, B)$

- Same as SKID3 without last exchange

Challenge-response authentication

Using

- Symmetric encryption
- One way functions
- Public key encryption
- Digital signatures

Authentication based on public key decryption

Witness to chosen random *r*

Challenge to Alice – encrypted with her public key

- Bob → Alice: $h(r)$, B, $PU_A(r, B)$
- Alice → Bob: r

Alice decrypts challenge to get *r*. Checks with *h(r)*. Sends *r* back for Bob to check.

Challenge-response authentication

Using

- Symmetric encryption
- One way functions
- Public key encryption
- Digital signatures

Unilateral Authentication using Signatures

Bob → Alice: r_B

Alice → Bob: $cert_A, r_A, B, PR_A(r_A, r_B, B)$

Bob checks:

- Identifier "B" is its own
- Signature is valid (after getting public key of Alice using certificate)
- Signed r_A prevents chosen-text attacks

Mutual Authentication using Signatures

Bob → Alice: r_B

Alice → Bob: $cert_A$, r_A, B, $PR_A(r_A, r_B, B)$

Bob → Alice: $cert_B$, A, $PR_B(r_A, r_B, A)$

Unilateral Authentication using Signatures

Time-Based Implicit Challenge

Alice → Bob: $cert_A$, t_A, B, $PR_A(t_A, B)$

Bob checks:

- Timestamp OK
- Identifier "B" is its own
- Signature is valid (after getting public key of Alice using certificate)

Standardization of the Challenge-response

- The ISO and the IEC (the International Electrotechnical Commission) have standardized **the three** challenge-response mechanisms as the basic constructions for **unilateral entity authentication** mechanisms.

- "ISO Two-Pass Unilateral Authentication Protocol":

  $B \rightarrow A : R_B \,||\, \text{Text11}$

  $A \rightarrow B : \text{Token}_{AB}$

  - $\text{Token}_{AB} = \text{Text3} \,||\, K_{AB}(R_B \,||\, B \,||\, \text{Text2})$.

Outline of Talk

- Definitions
- Passwords
  - Unix Passwords
  - One time passwords
- Challenge-response techniques
- Authentication Involving TTP
  - Needham-Schroeder
  - Kerberos

Authentication Involving TTP

- Authenticated key establishment protocols usually use a **trusted third party** (TTP), we usually name him Trent

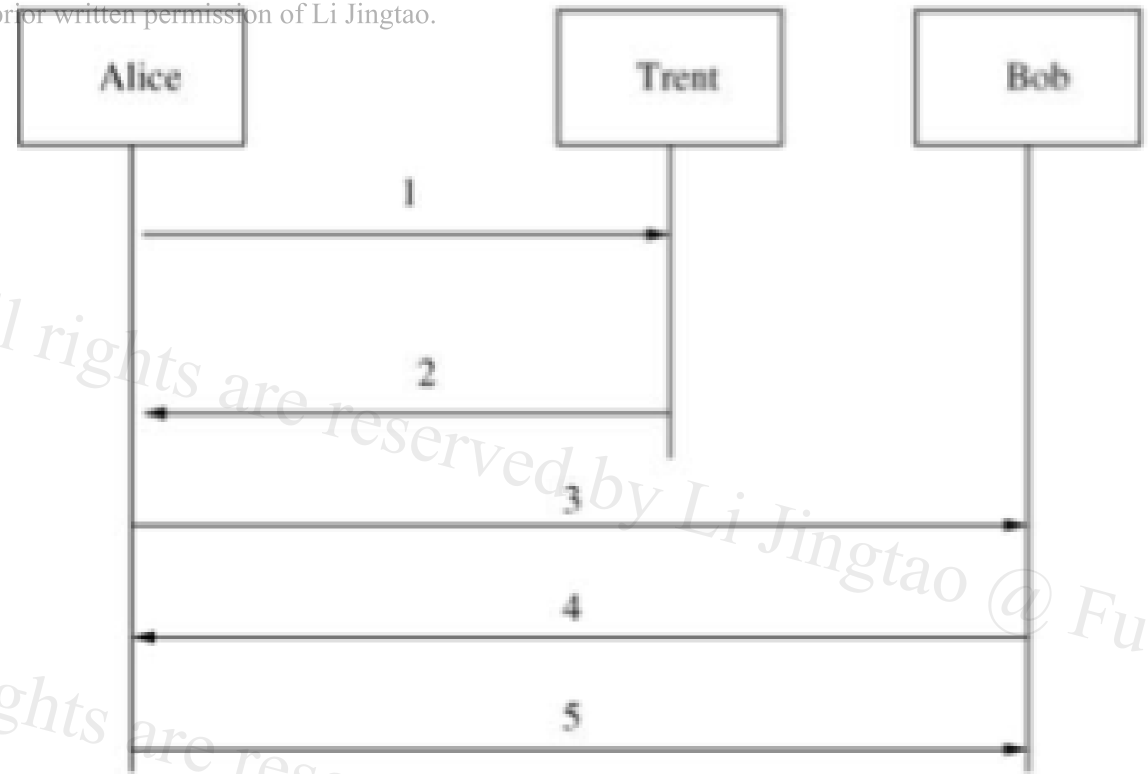
- The usual role of Trent is key distribution center (KDC)
  - Trent serves a large population of end users, he shares a long-term key with each of these users, e.g., K_{AT} , K_{BT}
  - Trent generates random session keys for end users, e.g., K_{AB}

- Using Trent's service, secure communication between any two end users can be achieved without having them to meet physically; they can run an authentication protocol to establish a shared session key

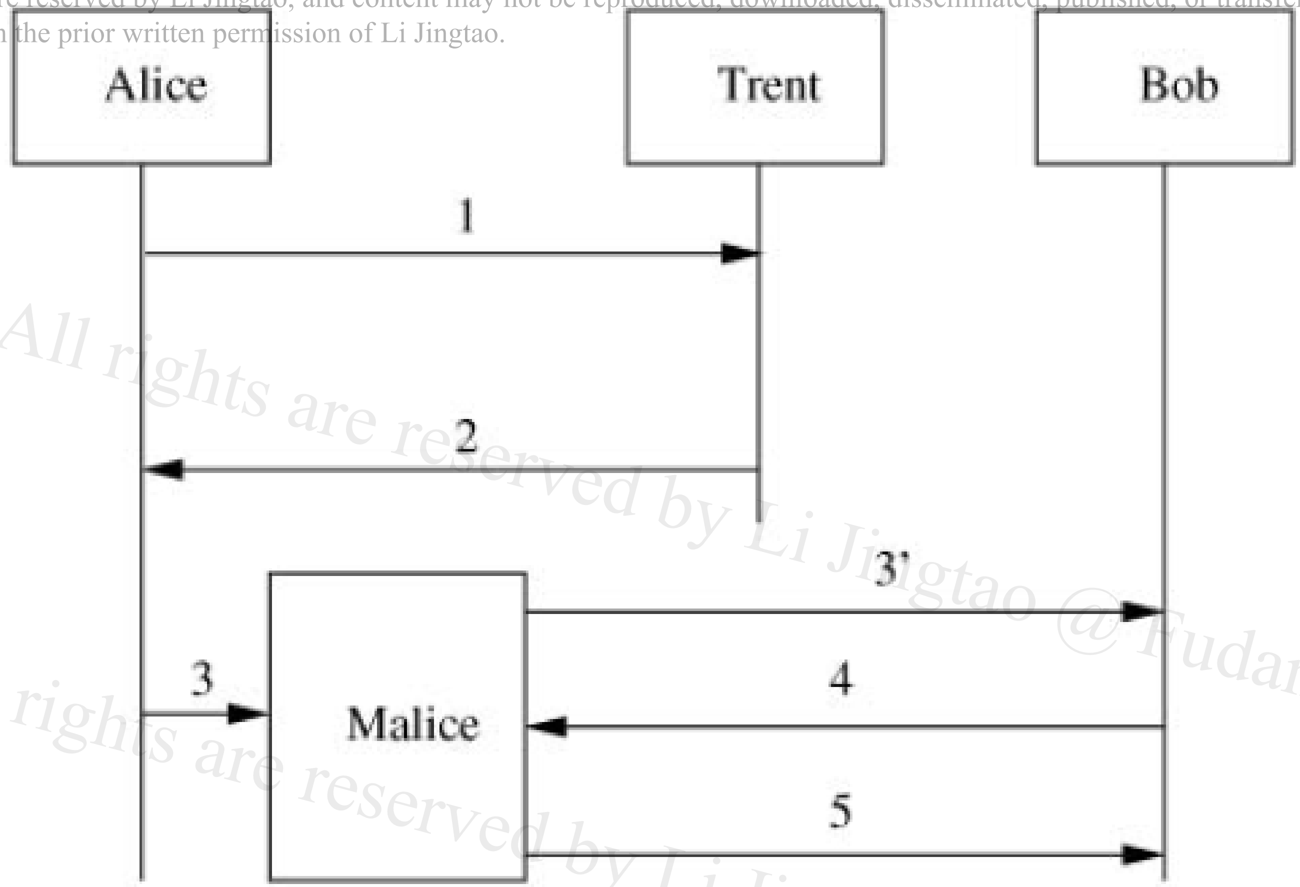
- After a session finishes, end users can forget each other

Needham-Schroeder Protocol

- Probably the most well-known authentication protocol
- Published in 1978, found flawed in 1981 by Denning and Sacco
- Corrected version becomes the basis for Kerberos
- PREMISE: Alice and Trent share key K_{AT}; Bob and Trent share key K_{BT}.
- GOAL: Alice and Bob want to establish a new and shared secret key K.



1. Alice creates N_A at random and sends to Trent: $Alice, Bob, N_A$;
2. Trent generates K at random and sends to Alice: $\{N_A, K, Bob, \{K, Alice\}_{KBT}\}_{KAT}$;
3. Alice decrypts, checks her nonce N_A, checks Bob's ID and sends to Bob: $Trent, \{K, Alice\}_{KBT}$;
4. Bob decrypts, checks Alice's ID, creates random N_B and sends to Alice: $\{I'm\ Bob!N_B\}_K$;
5. Alice sends to Bob: $\{I'm\ Alice!N_B - 1\}_K$.



1 and 2. (same as in a normal run)

3. Alice sends to Malice("Bob"): ...

3'. Malice("Alice") sends to Bob: $\{K', Alice\}_{K_{BT}}$;

4. Bob decrypts, checks Alice's ID and sends to Malice("Alice"): $\{I'm\ Bob!N_B\}_{K'}$;

5. Malice("Alice") sends to Bob: $\{I'm\ Alice!N_B - 1\}_{K'}$.

An Attack: Message Replay Attack

- **RESULT OF ATTACK**
  - Bob thinks he is sharing a new session key with Alice while actually the key is an old one and may be known to Malice.

- **Fix: Using Timestamp**

2. Trent sends to Alice: $\{Bob, K, T, \{Alice, K, T\}_{K_{BT}}\}_{K_{AT}}$;

3. Alice sends to Bob: $\{Alice, K, T\}_{K_{BT}}$;

1,4,5 Same as in the Needham-Schroeder.

- A,B checking

$$|Clock - T| < \Delta t_1 + \Delta t_2$$

KERBEROS

希腊神话里看护地狱之门的三头狗

Kerberos认证协议的历史

- Kerberos是一个经过长期考验的认证协议
  - 80年代中期
  - 是MIT的Athena工程的产物
  - 版本
    - 前三个版本仅用于内部
    - 第四版得到了广泛的应用
    - 第五版于1989年开始设计
      - RFC 1510, 1993年确定
      - 标准Kerberos
- 解决的问题
  - 认证、数据完整性、保密性

KERBEROS

- 解决的问题是：在一个分布式环境中，用户希望获取服务器上提供的服务。服务器能限制授权用户的访问，并能对服务请求进行认证
- 处理三种威胁：
  - 用户伪装成另一个用户访问服务器
  - 用户更改工作站的网络地址
  - 用户窃听报文交换过程，利用重放攻击进入服务器

KERBEROS

- 基于一个集中的认证服务器(可信第三方)，实现服务器（**Bob Server**）与用户**(Alice)**间的双向认证
  - AS, Authentication Server
  - KDC
- 基于对称加密实现，没有采用公开密钥体制
- 版本**4**使用**DES**算法

Kerberos V4

- 术语：

1. C＝客户
2. AS＝认证服务器（存放着所有用户及用户口令信息）
3. V＝服务器
4. IDc ＝在C上的用户标识符
5. IDv ＝V的标识符
6. Pc＝在C上的用户口令
7. ADc＝C的网络地址
8. Kv＝AS和V共享的加密密钥

一个简单的基于可信第三方的认证对话

（1） C → AS: IDc || Pc || IDv

（2） AS → C: Ticket

（3） C → V: IDc || Ticket

Ticket = E_{Kv}[IDc || ADc || IDv]

存在的问题

- 要求用户频繁地输入口令
- 申请不同的服务，用户需要新的票据
- 口令是明文传送的，敌对方可能窃听到口令
- 敌对方窃听到**Ticket**，摹仿**C**进行重放攻击

简单协议的改进

增加一个票据许可服务器TGS

用户登录时获取票据许可票：

（1）C → AS: IDc || IDtgs

（2）AS → C: E_{Kc} [$Ticket_{tgs}$]

E_{Kc} (user's secret key) is computed by a one-way function from the user's password

请求某种服务类型时获取服务许可票：

(3) C → TGS: ID_c ||ID_v||$Ticket_{tgs}$

(4) TGS → C: $Ticket_v$

获取服务：

(5) C → V: ID_c||$Ticket_v$

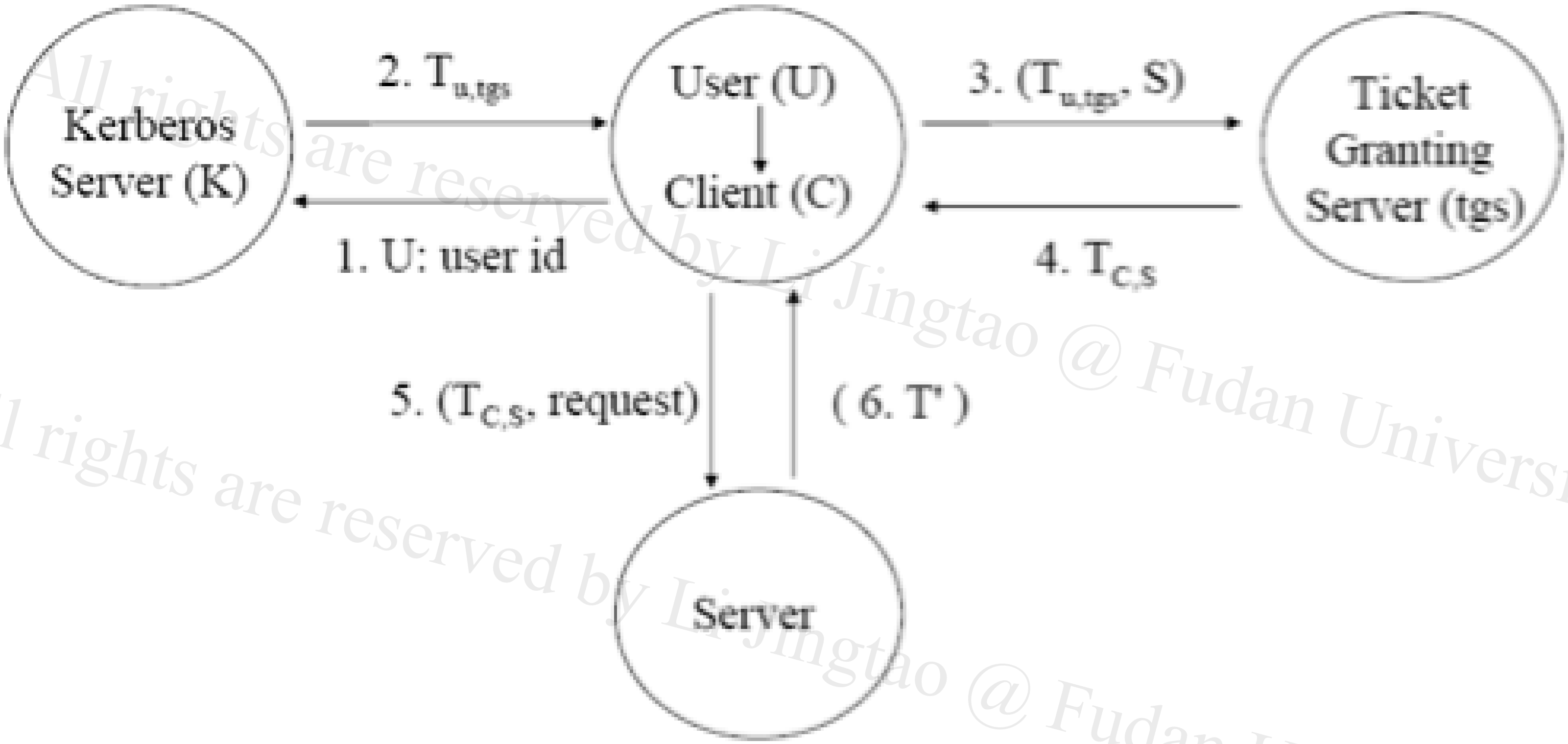
$Ticket_{tgs} = E_{Ktgs}[ID_c||AD_c||ID_{tgs}||TS_1||Lifetime_1]$

$Ticket_v = E_{kv}[ID_c||AD_c||ID_v||TS_2||Lifetime_2]$

存在的问题

- 每一张**ticket**的有效期限设置
  1. 如果太短，要求用户频繁地输入口令
  2. 如果太长，更多的机会遭受到重放攻击
- 敌对方可能偷窃**ticket**，在它过期之前进行使用
- 服务器如何向用户认证自己

Protocol steps



Ticket Structure:

$E_{K(S)}$ {C, S, $K_{C,S}$, timestamp, lifetime}

Conventions

Requirements:

- each user has a private password known only to the user
- a user's secret key can be computed by a one-way function from the user's password
- the AS knows the secret key of each user and the TGS
- each server has a secret key know by itself and TGS

Kerberos V4 对话

用户登录时获取票据许可票：

（1）C → AS: $ID_c \,||\, ID_{tgs} \,||TS_1$

（2）AS → C: $E_{Kc}\,[K_{c,tgs}||ID_{tgs}||TS_2||Lifetime_2||Ticket_{tgs}]$

请求某种服务类型时获取服务许可票：

(3) C → TGS: $ID_v||Ticket_{tgs}\,||Authenticator_c$

(4) TGS → C: $E_{Kc,tgs}[K_{c,v}||ID_v||TS_4||Ticket_v\,]$

$Authenticator_c = E_{Kc,tgs}[ID_c\,||AD_c||TS_3]$

获取服务：

(5) C → V: $Ticket_v\,||Authenticator_c$

(6) V → C: $E_{Kc,v}[TS_5+1]$

$Ticket_{tgs} = E_{Ktgs}[\,K_{c,tgs}||\,ID_c\,||AD_c||ID_{tgs}||TS_2||Lifetime_2]$

$Ticket_v = E_{kv}[K_{c,v}||ID_c||AD_c||ID_v||TS_4||Lifetime_4]$

$Authenticator_c = E_{Kc,v}[ID_c\,||AD_c||TS_5]$

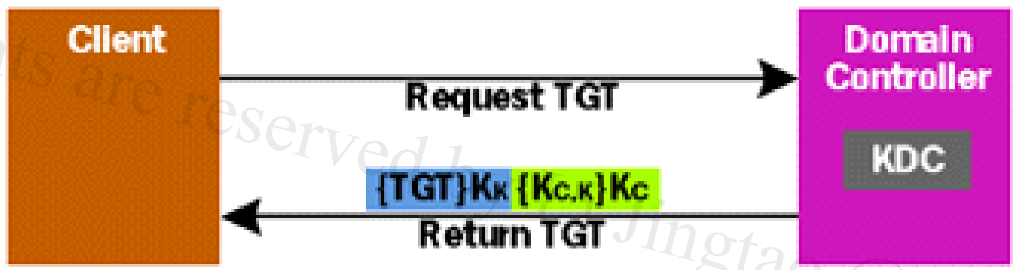
Kerberos协议的实现——MS版本

- Kerberos代替Windows NT的NT LM认证协议，是Win2000的默认认证协议，也是Windows 2000分布式安全服务的一部分
- 与Windows 2000的目录服务集成在一起
  - Kerberos是AD的一部分
- 与系统的授权数据信息结合在一起
- 对MIT Kerberos作了扩展，也不完全兼容

Win2k Kerberos的Ticket结构

Ticket交换

- 登录



- 访问服务