INFORMATION SECURITY

Authentication and Kerberos

Information Security

School of Data & Computer Science
Sun Yat-sen University

Lecture Notes:
courseware_is_sysu@163.com
Instructor: Guoyang Cai
email: isscgy@mail.sysu.edu.cn

中山大學
SUN YAT-SEN UNIVERSITY

Outline

- **Chap 6. Authentication and Kerberos**
  - **Introduction**
    - Authentication Technologies
    - The Weak/Strong Authentication Scheme
    - Zero-knowledge Authentication & *Fiat-Shamir* Algorithm
    - The Application of Authentication Technologies
    - Attacks to Authentication
    - The Security Guidelines to Protect Authentication Schemes
  - **Public Key Infrastructure**
    - Introduction to PKI
    - PKIX
    - Public Key Certificate
    - Trust Hierarchy Model

Outline

- **Chap 6. Authentication and Kerberos**
  - **Kerberos**
    - History & Development
    - *Needham-Schroeder* Symmetric Key Protocol
    - Description of Kerberos
    - Kerberos Process in Detail
    - Drawbacks & Limitations
    - Introduction
  - **X.509**
    - X.509 Certificate
    - Security problems
    - Application

Authentication Technologies

- Prover and Verifier
  - Prover and verifier are two parties involved in authentication
  - Prover
    - A claimant 申请者/声称者/出证人
    - A prover presents its identity and a proof of that identity.
  - Verifier
    - A recipient 接受者/验证方/审议人
    - A verifier checks the prover’s proof to ensure the identity of the prover.

Authentication Technologies

- Entity Authentication and Message Authentication
  - Entity Authentication 实体认证
    - *Entity Authentication* can be defined as the process through which the identity of an entity (such as an individual, a computer/an operating process, an application, or a network) is demonstrated.
    - Entity authentication involves no meaningful message except the claim of being an entity.
  - Message Authentication 消息认证
    - *Message Authentication* provides the assurance that a message has not been modified during its transmission.
    - Message authentication does not provide time-related guarantees with respect to when the message has been created, signed, sent, or delivered to destination.

Goals of an Authentication

- An authentication protocol is a process to achieve the following goals:
  - The probability that a third party different from the prover, using the authentication protocol and impersonating the prover, can cause the verifier to authenticate the third party as the prover, is neglectable.
    - 第三方假冒出证人，使用认证协议导致审议人将第三方身份认证为出证人的概率可以忽略不计。
  - The verifier should not be able to reutilize the information provided by the prover to impersonate him/her (the prover) to a third party.
    - 审议人不能够重复使用出证人提供的信息向第三方假冒出证人身份。
  - A signed message cannot be recovered from the signature code during signature verification.
    - 签名验证过程中不能够从签名代码恢复签名保护的消息。

Three Types of Entity Authentication

- Know-object-based authentication 基于原始信息的
  - E.g., password, PIN, ...
- Passed-object-based authentication 基于持有证明的
  - E.g., credit card, smart card, ...
- Biometric-object-based authentication 基于生物学特征的
  - E.g., fingerprints, retinal pattern, voice, hand geometry, keystroke dynamics, ...

The Weak Authentication Scheme

- Base on the password
  - Password authentication is perhaps the most common way of authenticating a user to an electronic system.
  - The entity provides a (login, password) pair; Differ somewhat by the technique used to perform the verification and store the information providing password verification.
  - Non-time-varying password schemes present various security weakness.
- Base on PIN
  - PIN schemas can be classified as special time-invariant passwords.
  - This scheme represents a vulnerability that should be covered by additional constraints.
  - Online/Off-line verification.

The Strong Authentication Scheme

- Challenge-Response with password
  - Secret key cryptography 对称密码学方法
    - r: a random number generated by *Bob*, the verifier
    - r': a random number generated by *Alice*, the prover
    - k: the secret key shared by the prover and verifier
    - E_k: a secret-key encryption function with key k
    - D_k: a secret-key decryption function with key k
    - m: optional message used to prevent replay attacks
    - The authentication of *Alice*: (*Alice* - prover, *Bob* - verifier)
      - *Bob*: $r \Rightarrow$ *Alice* (*Bob* sent a random number r to *Alice*)
      - *Alice*: $E_k(<r, m>) \Rightarrow$ *Bob*
      - *Bob*: $D_k(E_k(<r, m>))$, check r and m
    - Mutual authentication:
      - *Bob*: $r \Rightarrow$ *Alice*
      - *Alice*: $E_k(<r, r', m>) \Rightarrow$ *Bob*
      - *Bob*: $D_k(E_k(<r, m>))$, check r and m
      - *Bob*: $E_k(<r', m'>) \Rightarrow$ *Alice*
      - *Alice*: $D_k(E_k(<r', m'>))$, check r' and m'

The Strong Authentication Scheme

- Challenge-Response with password
  - Public key cryptography 非对称密码学方法
    - r: a random number generated by *Bob*
    - K_{PUa}: the public key of *Alice*
    - K_{PRa}: the private key of *Alice*
    - $C(k, -)$: a public-key encryption function with parameter k
    - $D(k, -)$: a public-key decryption function with parameter k
    - m : optional message used to prevent replay attacks
    - The authentication of *Alice*: (*Alice* - prover, *Bob* - verifier)
      - *Bob*: $C(K_{PUa}, r) \Rightarrow$ *Alice*
      - *Alice*: $C(K_{PRa}, <r, m>) \Rightarrow$ *Bob*
      - *Bob*: $D(K_{PUa}, C(K_{PUa}, <r, m>))$, check r and m
    - Mutual authentication
      - *Think about it.*

The Zero-knowledge Proof

- The goal of zero-knowledge proof (ZKP) systems is to prove a statement without leaking extra information. A zero-knowledge proof must satisfy three properties:
  - Completeness: if the statement is true, the honest verifier will be convinced of this fact by an honest prover. (完备性)
  - Soundness: if the statement is false, no cheating prover can convince the honest verifier that it is true, except with some small probability. (有效性)
  - Zero-knowledge: if the statement is true, no cheating verifier learns anything other than the fact that the statement is true. That is, just knowing the statement (not the secret) is sufficient to show that the prover knows the secret.
- The first two properties are of more general interactive proof systems. The third is what makes the proof zero-knowledge.

Zero-knowledge Authentication

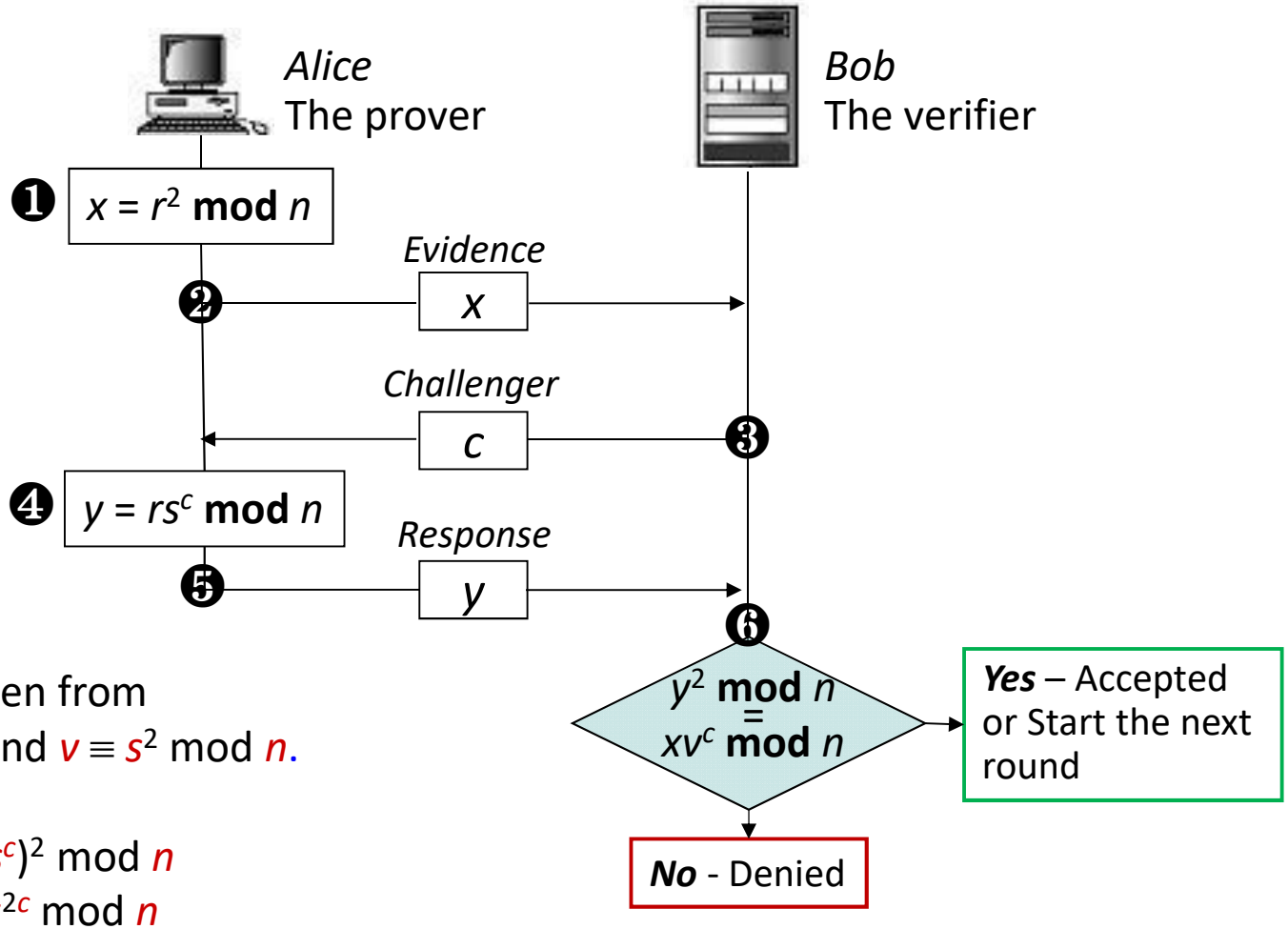
- Zero-knowledge proofs are *probabilistic* "proofs" rather than deterministic proofs. There is some small probability, the *soundness error* (失效), that a cheating prover will be able to convince the verifier of a false statement.
- The zero-knowledge authentication (零知识认证) is a form of interactive proof, during which the prover and the verifier exchange various messages and random numbers to achieve authentication.
- *Example*.
  - The *Ali Baba* cave
    - *Alice* claims that she knows a secrete key *s* without showing it to *Bob.*
  - *Fiat-Shamir* algorithm (*Amos Fiat* and *Adi Shamir*, 1986)
  - *Feige-Fiat-Shamir* identification scheme (*Uriel Feige*, *Amos Fiat,* and *Adi Shamir*, 1988)

Zero-knowledge Authentication

- *Fiat-Shamir* algorithm.
  - n: $n = pq$, p and q are big primes. n is generated by a trusted third party and released in public, p, q kept in secret (or destroyed).
  - s: *Alice*'s secrete key, $0 < s < n$, $\gcd(s, n) = 1$. As the prover *Alice* must proof that she knows s, without showing s to anyone.
  - v: *Alice*'s "public key" satisfying $v \equiv s^2 \bmod n$.
    - *Alice* computes and shows v to *Bob*, the verifier.
    - It is hard to know s from v and n (as hard as the factoring large numbers problem when $n = pq$, as selected).
  - r: *Alice* selected a random number r, $0 < r < n - 1$, and start the process.
  - x: $x = r^2 \bmod n$ as an evidence, computed by *Alice* and sent to *Bob*.
  - c: The challenger randomly selected in {1, 0} from *Bob*.
  - y: $y = rs^c \bmod n$ as the response, computed by *Alice* and sent to *Bob*.

Zero-knowledge Authentication

- *Fiat-Shamir* algorithm.



If *Alice* knows s, then from

$x = r^2 \bmod n$ and $v \equiv s^2 \bmod n$.

we have

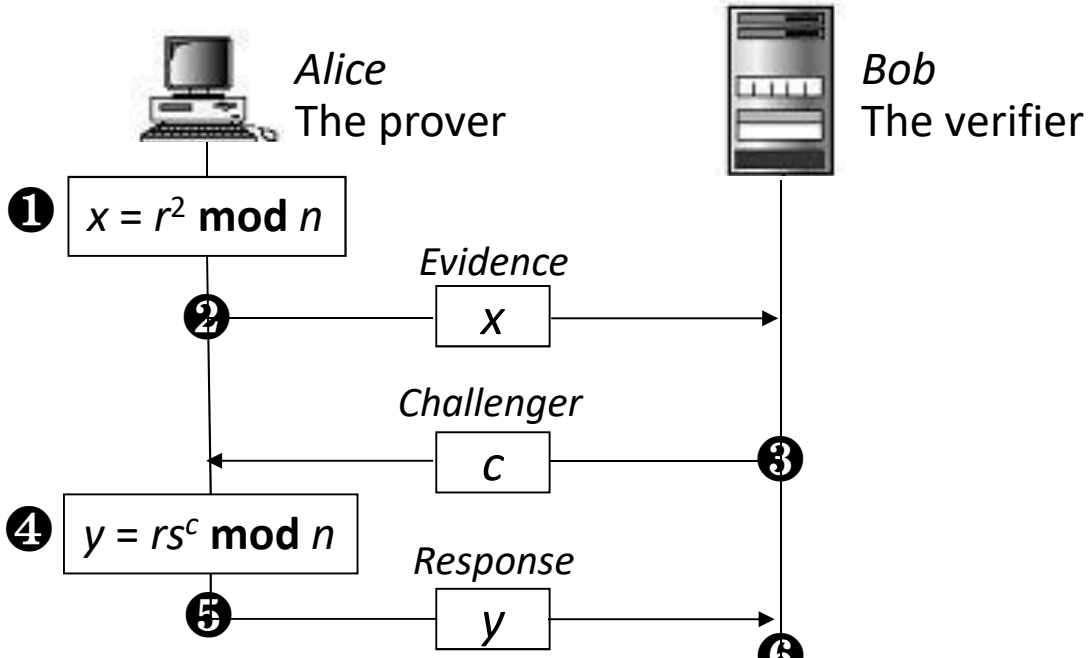
$y^2 \bmod n = (rs^c)^2 \bmod n$

$= r^2 s^{2c} \bmod n$

$= xv^c \bmod n.$

Zero-knowledge Authentication

- *Fiat-Shamir* algorithm.



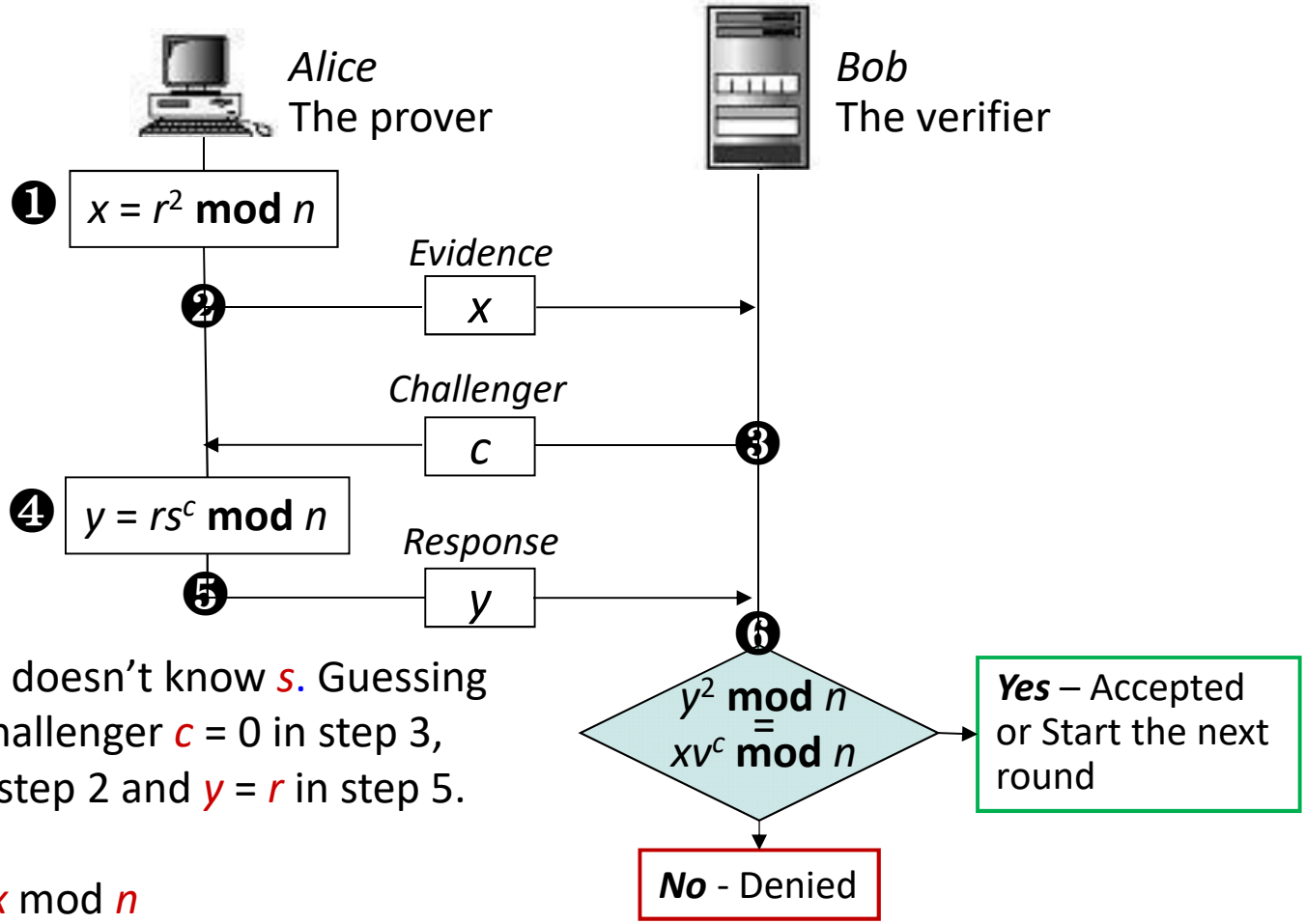
Suppose that *Alice* doesn't know s. Guessing *Bob* sending the challenger $c = 1$ in step 3, *Alice* sent $x = r^2/v$ in step 2 and $y = r$ in step 5. Then

$$xv^c \bmod n = (r^2/v)v \bmod n$$
$$= r^2 \bmod n = y^2 \bmod n.$$

and passed step 6.

Zero-knowledge Authentication

- *Fiat-Shamir* algorithm.



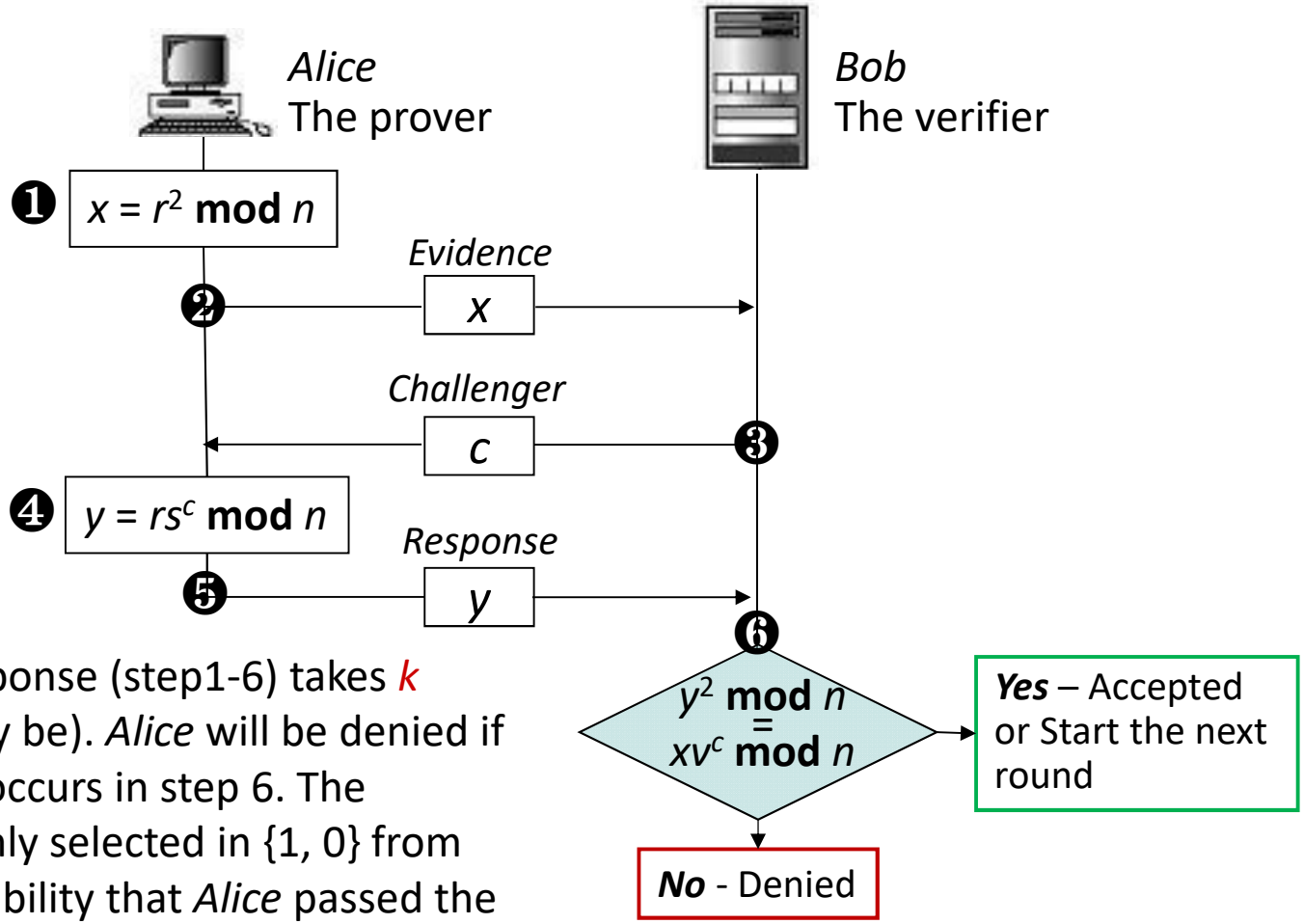
Suppose that *Alice* doesn't know s. Guessing *Bob* sending the challenger $c = 0$ in step 3, *Alice* sent $x = r^2$ in step 2 and $y = r$ in step 5. Then

$$xv^c \bmod n = x \bmod n$$
$$= r^2 \bmod n = y^2 \bmod n.$$

and also passed step 6.

Zero-knowledge Authentication

- *Fiat-Shamir* algorithm.



The challenge-response (step1-6) takes k rounds (k = 20 may be). *Alice* will be denied if a dismatch firstly occurs in step 6. The challenger randomly selected in {1, 0} from *Bob* and the probability that *Alice* passed the verification illegally is $1/2^k$, 1ppm.

Zero-knowledge Authentication

- Base on Device
  - Using device to help user authentication in a hostile environment (敌对环境).
  - The device's key is randomly selected out of the key space of the embedded cryptographic algorithm.
  - Since the key is strong, the probability of success of a brute force attack is almost null.

The Application of Authentication Technologies

- X.509
  - The ITU-T recommendation X.509 defines a directory service that maintains a database of information about users for the provision of authentication services.
- Kerberos
  - Kerberos is an authentication service developed at MIT which allows a distributed system to be able to authenticate requests for service generated from workstations.

Attacks to Authentication

- Impersonation attacks
  - 假冒攻击
- Replay attacks
  - 重放攻击
- Forced delay attacks
  - 强迫延时攻击
- Interleaving attacks
  - 交错攻击
- Oracle session attack
  - Oracle 会话攻击
- Parallel session attack
  - 并行会话攻击

The Security Guidelines to Protect Authentication Schemes

- Risk Mitigation Strategy (风险降低策略)
  - When a weak authentication based on passwords is used, a pass-policy should be stated and should include rules describe how to manage a user's accounts.
- Implementation of a Risk Mitigation Strategy
  - The authentication process and the integrity service should use different keying materials (密钥资料) if necessary.
  - The working station's procedures should be protected and tightly synchronized, if they are involved in the timestamps' computation and verification.
  - Anonymous remote attempts of authentication should be unauthorized and limit the number of try.
  - Security parameters should take the reduction of the probability of successful attacks into consideration.

The Security Guidelines to Protect Authentication Schemes

- Implementation of a Risk Mitigation Strategy (cont.)
  - In the case where a trust relationship is defined between servers, authentication-related configurations should be reviewed carefully.
  - Assessing the cryptographic and digital signature.
  - In assessing an authentication scheme, the potential impact of compromise of keying material should be studied.
    - 在作认证方案评估时，对危及密钥资料的潜在威胁须加以研究。

Public Key Infrastructure

- Public Key Infrastructure (PKI, 公钥基础设施) provides well-conceived infrastructures (精心设计的基础设施) to deliver security services in an efficient and unified style. PKI is a long-term solution that can be used to provide a large spectrum of security protection.
- PKI is used to
  - generate digital certificates.
  - manage the certificates, certificate statuses, and the business element.
  - involve symmetric key cryptography for different purposes
  - other security purposes.

Public Key Infrastructure

- PKI 的概念
  - PKI 是一组服务和策略，提供了一个将公钥和用户身份唯一绑定的机制，以及如何实施并维护这个绑定相关信息的框架；
  - PKI 是一个通过使用公开密钥技术和数字证书来确保系统信息安全，并负责验证数字证书持有者身份的体系。
- PKI 的主要功能
  - 签发数字证书以绑定证书持有者的身份和相关的公开密钥；
  - 为用户获取证书、访问证书和吊销证书提供途径；
  - 利用数字证书及相关的各种服务 (证书发布、黑名单发布等) 实现通信过程中各实体的身份认证，保证通信数据的完整性和不可否认性。

Public Key Infrastructure

- PKI 已经获得广泛应用，典型应用如：
  - 虚拟专用网络 VPN
    - VPN 是一种构建在公用通信基础设施上的专用数据通信网络，利用网络层安全协议 (如 Ipsec) 和建立在 PKI 上的加密与数字签名技术来获得机密性保护。
  - 安全电子邮件
    - 可以利用 PKI 实现电子邮件的安全要求，包括机密、完整、认证和不可否认性。目前发展很快的安全电子邮件协议 S/MIME，是一个允许发送加密和有签名邮件的协议。该协议采用了 PKI 数字签名技术并支持消息和附件的加密，无须收发双方共享相同密钥。

Public Key Infrastructure

- PKI 已经获得广泛应用，典型应用如：
  - Web 服务安全
    - 为了解决 Web 服务的安全问题，在两个实体进行通信之前，先建立 SSL 连接，以此实现对应用层透明的安全通信。利用 PKI 技术，SSL 协议在协商时完成了对服务器和客户端基于证书的身份认证 (其中对客户端的认证是可选的)。

Public Key Infrastructure

- PKI 场景：一个 B/S 架构下安全浏览网页的例子

(1) Web 服务器 *W* 生成一对私钥/公钥 (PR_W, PU_W) 并向认证机构 ***CA*** 申请一个数字证书 ***X***，证书中包含了 PU_W；***CA*** 保证 PU_W 是 *W* 的公钥；证书 ***X*** 用 ***CA*** 的私钥 $\boldsymbol{PR_{CA}}$ 加密作为数字签名；***CA*** 的公钥 $\boldsymbol{PU_{CA}}$ 是公开声明的；

(2) *W* 向客户端浏览器 *B* 发送数字证书 ***X***；

(3) *B* 用 ***CA*** 的公钥 $\boldsymbol{PU_{CA}}$ 认证数字证书 ***X*** 确实是 ***CA*** 发布的；

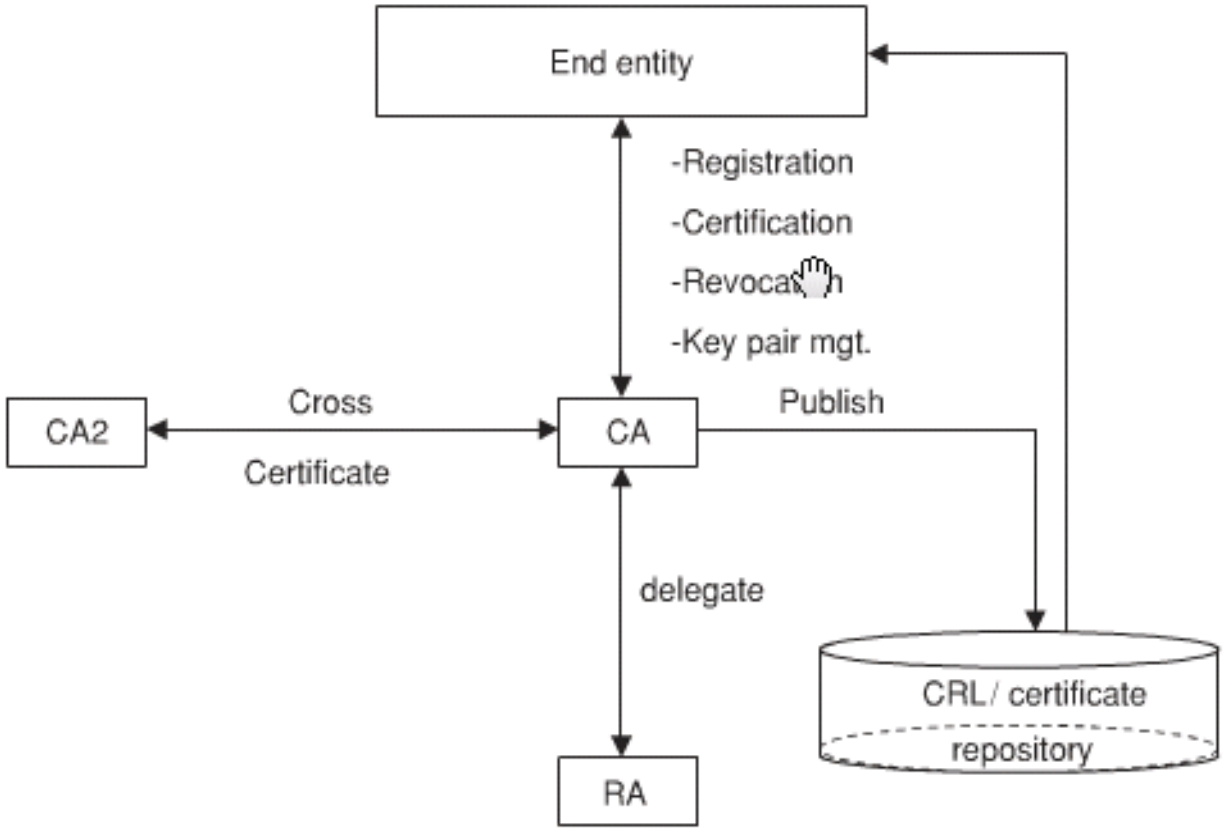
(4) *B* 产生一对私钥/公钥 (PR_B, PU_B)，利用数字证书 ***X*** 中包含的 PU_W 去加密公钥 PU_B 得到公钥密文 M_{PUB} 并发给 *W*；

(5) *W* 使用自己的私钥 PR_W 解密 M_{PUB}，得到 PU_B，然后 *W* 使用 PU_B 加密网页 *G* 得到密文 G_{PUB} 并传送给 *B*；

(6) *B* 使用自己的私钥 PR_B 解密 G_{PUB}，恢复明文网页 *G*。

PKIX

- The PKIX (Public key infrastructure X.509) model defines the elements that comprise a PKI including components, documents, and policy instruments.
- The Component of PKIX.



PKIX

- The Component of PKIX
  - The End-entity (终端实体)
    - the user/consumers of the PKI-related services, such as subscribers, network devices, processes, or any other entity that has applied for and received a digital certificate for use in supporting the security and trust in transactions to be undertaken.
  - Certification Authority (CA, 证书授权机构/认证机构)
    - the issuer of PKC and certificate revocation lists (CRL).
    - PKC are digitally signed by the CA, which effectively (and legally) binds the subject name to subject public key and the CA's public key.
    - a CA also involved in a number of administrative and technical tasks.

PKIX

- The Component of PKIX
  - Registration Authority (RA, 注册机构/注册中心)
    - a registration authority is an administrative component to which a CA delegates certain management functions. However, the RAs are not allowed to issue certificates or CRLs. (RA 是受CA委托实施某些管理功能的管理组件)
  - Certification Repository (CR, 证书仓库)
    - a certificate repository is a generic term used to specify any method for storing and retrieving certificate-related information such as the public key certificates issued for end-entities and the CRLs which report on revoked certificates.

PKIX

- The Component of PKIX
  - Public Key Certificate (PKC, 公开密钥证书)
    - PKC is a digital document that is associated with and end-entity. It provides a means of identifying end-entities of their identities to public keys.
  - Certificate Revocation List (CRL, 证书撤回清单/证书吊销列表)
    - a signed document containing reference to certificates, which are decide to be no longer valid.
  - CRL issuer (CRL, 签发者)
    - CRLI may be an optional entity to which a CA delegates the verification of information related to revocation, issuance and the publication of CRLs.

PKIX

- PKI Documents
  - a PKI must be operated in accordance with well-defined policies that define the rules to perform the PKI activities appropriately. Four important documents are:
    - Certificate policy (CP, 证书策略)
    - Certificate practice statement (CPS, 证书操作规范)
    - Subscriber agreements (用户协议)
    - Relying party agreements (第三方信任协议)

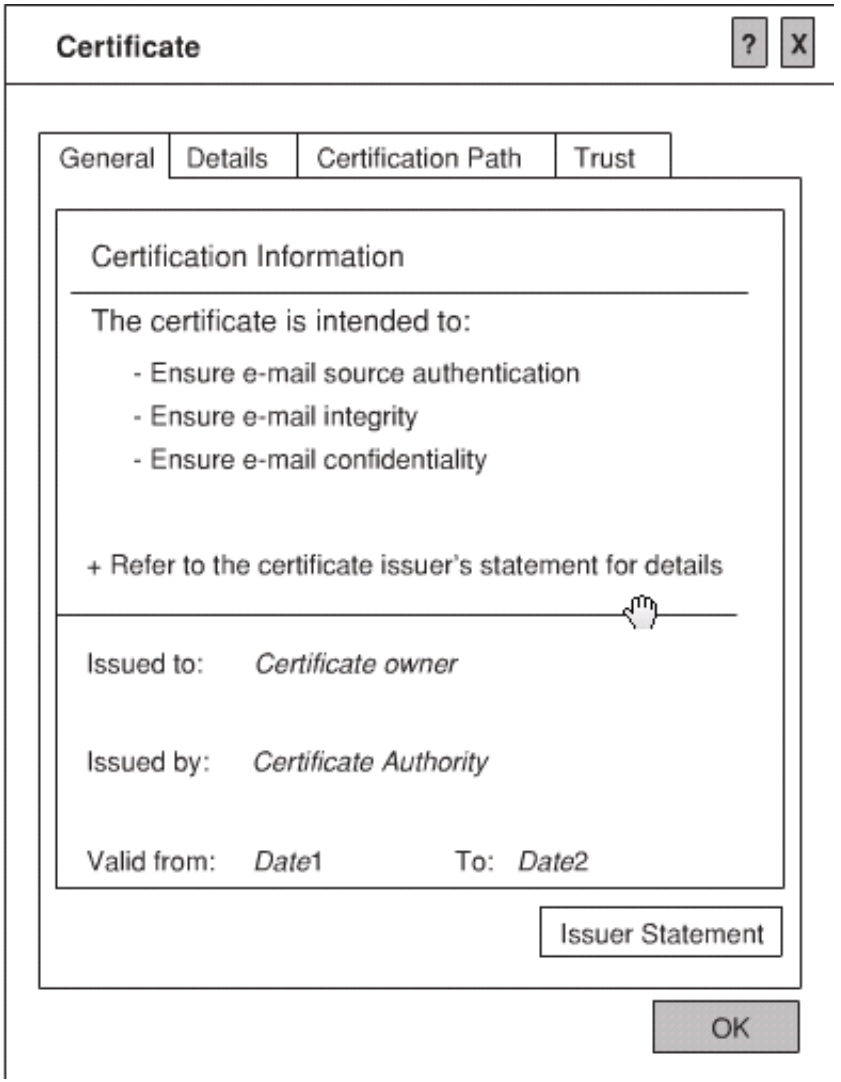
- **PKIX**
  - PKI Documents
    - Certificate policy (CP)
      - a certificate policy sets forth general requirements that PKI participants must meet in order to operate within a PKI. A CP is also a named set of rules that indicate the applicability of a certificate to a given application.
    - Certificate practice statement (CPS)
      - a certificate practice statement defines a comprehensive statement of practices and procedures followed by a single CA or a related set of CAs set out in a CP.

- **PKIX**
  - PKI Documents
    - Subscriber agreements
      - a document representing an agreement between the subscriber applying and receiving a certificate and the issuing authority of the certificate. It focuses on the subscriber's responsibilities, rights, and obligations in using the certificate.
    - Relying party agreements
      - this is typically an agreement between a party that wishes to rely on a certificate and the information contained in it.

- **PKIX**
  - The Management of PKIX
    - Registration
    - Initialization
    - Certificate generation
    - Certificate update
    - Revocation
    - Key pair management
    - Cross-certification
    - Additional management functions

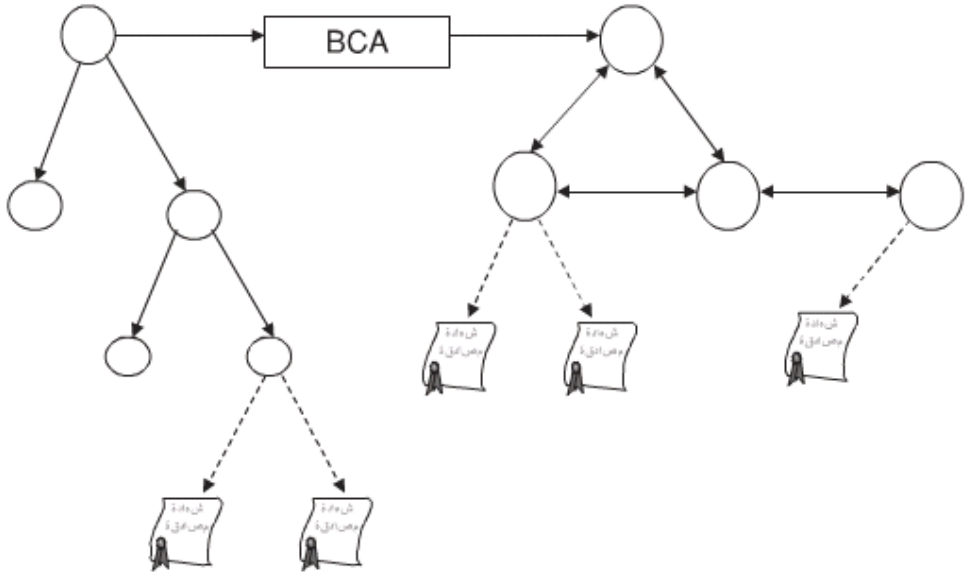
Public Key Certificate

- The Form of certificate
  - (1) Certificate version
  - (2) Serial number
  - (3) Signature algorithm
  - (4) Issuer
  - (5) Validity
  - (6) Subject
  - (7) Subject public key info



Trust Hierarchy Model

- Hierarchy Model 严格分层模型
- Mesh PKI 对等模型
- Bridge CA 桥接模型



Introduction

- *Kerberos* (ITU-T) is a computer network authentication protocol which works on the basis of “tickets” (票据) to allow nodes communicating over a non-secure network to prove their identity to one another in a secure manner.
  - Its designers aimed primarily at a client–server model, and it provides mutual authentication - both the user and the server verify each other’s identity.
  - Kerberos protocol messages are protected against eavesdropping and replay attacks.
  - It is a free Software and used widely in different platforms.
- Version 1-3: occurred internally at MIT
- Version 4: designed by *Steve Miller* and *Clifford Neuman,* published in the late 1980s, also targeted at Project Athena.
- Version 5: published by *Clifford Neuman* and *John Kohl* in 1993, appeared as RFC 1510, and obsoleted by RFC 4120 in 2005.

Introduction

- Kerberos is based on *Needham* & *Schroeder*'s protocol, and as part of MIT's *Athena* project.
- Kerberos builds on symmetric key cryptography and requires a trusted third party, and optionally may use public-key cryptography by utilizing asymmetric key cryptography during certain phases of authentication.
- Kerberos was named by a *Greek* mythological character, the watching dog of the gate of Hell with three heads representing the three A's functions:
  - Authentication (complete)
  - Accounting (not complete)
  - Audit (not complete)

Needham-Schroeder Symmetric Key Protocol

- The *Needham–Schroeder* Symmetric Key Protocol
  - Invented by *Roger Needham* and *Michael Schroeder*, 1978.
  - It aims to establish a session key between two parties on a network, typically to protect further communication. It is based on a symmetric encryption algorithm and forms the basis for the Kerberos protocol. A trusted third party *TA* (KDC, Key Distributing Centre) is required.
- Scenario: *Alice* wishes to communicate with *Bob*.
  - *TA* is a trusted third party that provides session keys (e.g., a shared key K_{AB} between *Alice* and *Bob*).
  - $E(K, -)$ is a symmetric cryptographic algorithm (e.g., DES).
  - *TA* has a key K_{AT} in common with *Alice* and a key K_{BT} with *Bob*.
  - *Alice* (as a verifier) authenticates *TA* (as a prover) using a nonce r_A and obtains a session key K_{AB} from *TA*.
  - *Alice* (as a prover) authenticates to *Bob* (as a verifier) and transports the session key securely.

Needham-Schroeder Symmetric Key Protocol

- Scenario: *Alice* wishes to communicate with *Bob*.
  - (1) *Alice* $\Rightarrow$ *TA*: $E(K_{AT}, <Alice, Bob, r_A>)$ or client-based logon
    - *TA* decrypts the message with K_{AT}, authenticates *Alice*'s logon.
  - (2) *TA* $\Rightarrow$ *Alice*: $E(K_{AT}, <K_{AB}, r_A, Bob, E(K_{BT}, <K_{AB}, Alice>)>)$
    - *Alice* decrypts the message with K_{AT}, checks r_A and "*Bob*", holds K_{AB} for future correspondence with *Bob.*
  - (3) *Alice* $\Rightarrow$ *Bob*: $E(K_{BT}, <K_{AB}, Alice>)$
    - *Bob* decrypts the message with K_{BT}, holds K_{AB} for future correspondence with *Alice.*
  - (4) *Bob* $\Rightarrow$ *Alice*: $E(K_{AB}, r_B)$
    - *Alice* decrypts the message with K_{AB}.
  - (5) *Alice* $\Rightarrow$ *Bob*: $E(K_{AB}, r_B - 1)$
    - *Bob* checks $r_B - 1$.
    - Now *Alice* and *Bob* share a secret key K_{AB}.

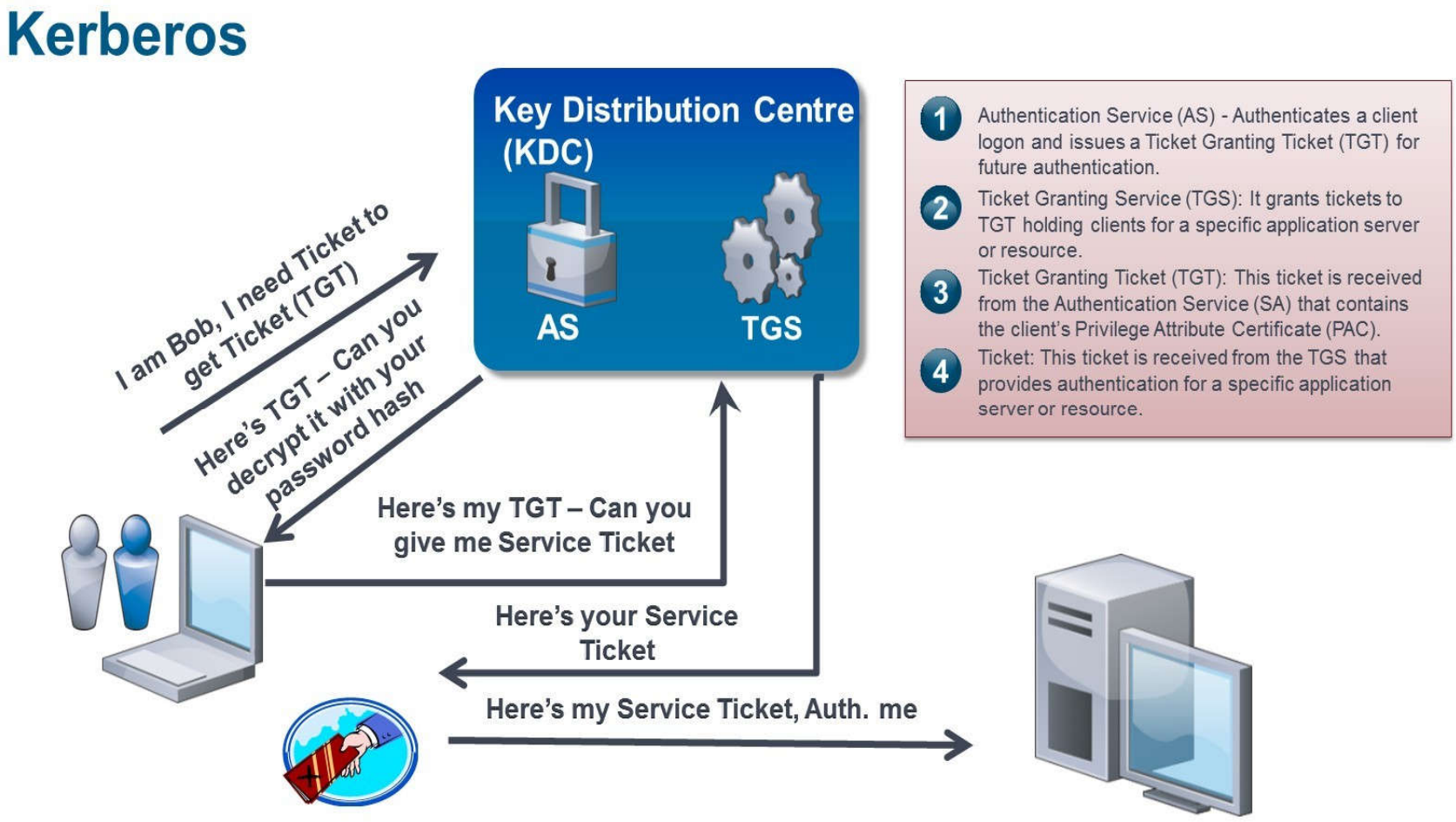
Needham-Schroeder Symmetric Key Protocol

- Scenario: *Alice* wishes to communicate with *Bob*.
  - (1) *Alice* $\Rightarrow$ *TA*: $E(K_{AT}, <Alice, Bob, r_A>)$ or client-based logon
    - *TA* decrypts the message with K_{AT}, authenticates *Alice*'s logon.
  - (2) *TA* $\Rightarrow$ *Alice*: $E(K_{AT}, <K_{AB}, r_A, Bob, E(K_{BT}, <K_{AB}, Alice>)>)$
    - *Alice* decrypts the message with K_{AT}, checks r_A and "*Bob*", holds K_{AB} for future correspondence with *Bob*.
  - (3) *Alice* $\Rightarrow$ *Bob*: $E(K_{BT}, <K_{AB}, Alice>)$
    - *Bob* decrypts the message with K_{BT}, holds K_{AB} for future correspondence with *Alice*.
  - (4) *Bob* $\Rightarrow$ *Alice*: $E(K_{AB}, r_B)$
    - *Alice* decrypts the message with K_{AB}.
  - (5) *Alice* $\Rightarrow$ *Bob*: $E(K_{AB}, r_B - 1)$
    - *Bob* checks $r_B - 1$.
    - Now *Alice* and *Bob* share a secret key K_{AB}.
- Attack on the *Needham–Schroeder* protocol
  - Suppose that *Darth* stole an old K_{AB} and replay step (3), pretending to be *Alice*. *Bob* will accept it, being unable to tell that the key is not fresh.

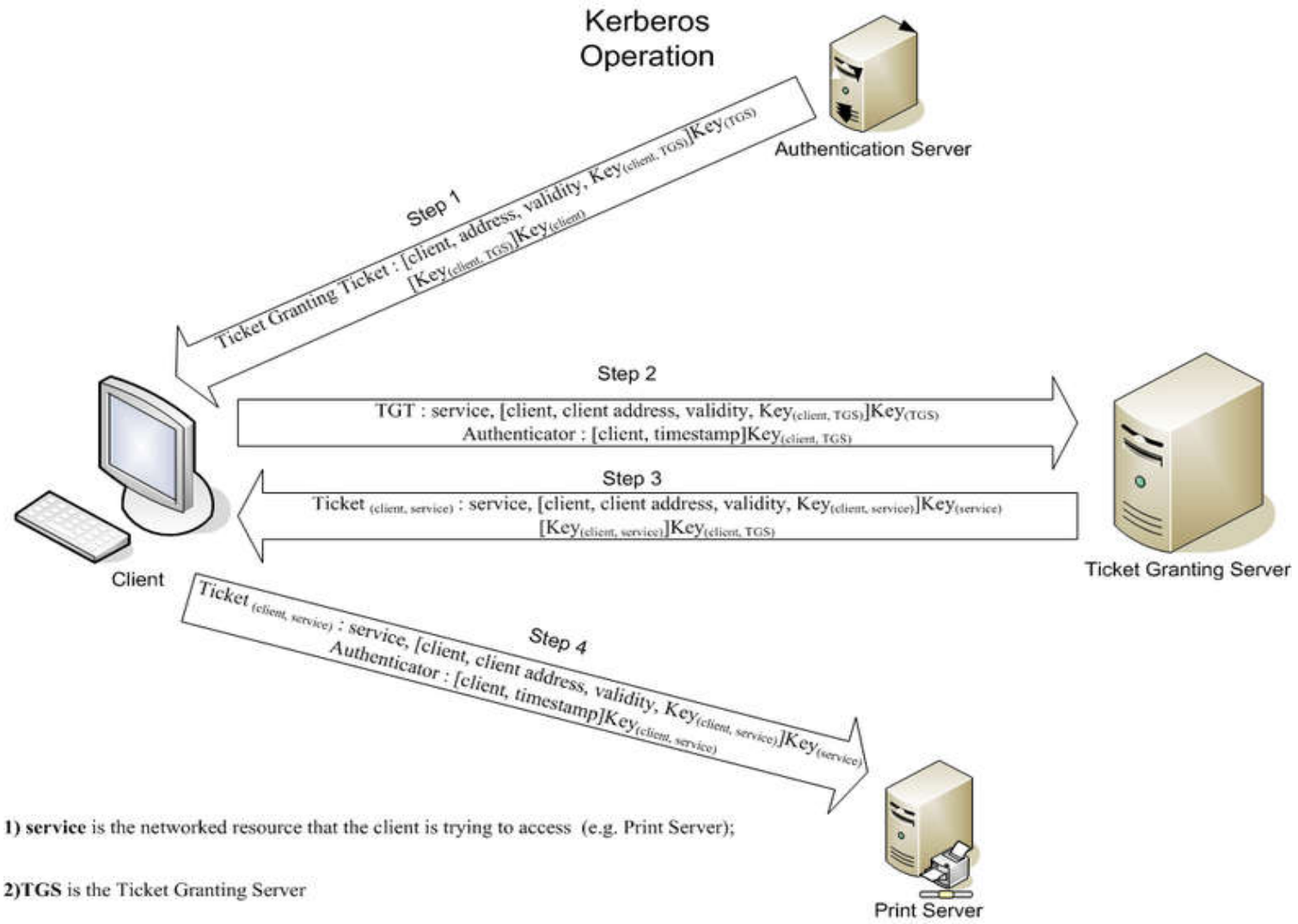
Needham-Schroeder Symmetric Key Protocol

- An improved *Needham-Schroeder* symmetric key protocol
  - (1) *Alice* $\Rightarrow$ *Bob*: *Alice*
  - (2) *Bob* $\Rightarrow$ *Alice*: $E(K_{BT}, <Alice, r_B{}^0>)$
  - (3) *Alice* $\Rightarrow$ *TA*: $E(K_{AT}, <Alice, Bob, r_A , E(K_{BT} , <Alice, r_B{}^0>)>)$
  - (4) *TA* $\Rightarrow$ *Alice*: $E(K_{AT}, <K_{AB}, r_A, Bob, E(K_{BT}, <K_{AB} , Alice, r_B{}^0>)>)$
  - (5) *Alice* $\Rightarrow$ *Bob*: $E(K_{BT}, <K_{AB} , Alice, r_B{}^0>)$
    - *Bob* decrypts the message with K_{BT} and checks $r_B{}^0$.
  - (6) *Bob* $\Rightarrow$ *Alice*: $E(K_{AB} , r_B)$
    - *Alice* decrypts the message with K_{AB} .
  - (7) *Alice* $\Rightarrow$ *Bob*: $E(K_{AB} , r_B -1)$
    - *Bob* checks $r_B -1$.
- The improved process fixed the attack on the protocol: the session key K_{AB} is bound with $r_B{}^0$ ($r_B{}^0$ can be a nonce or a timestamp). *Bob* can recognize that the key is fresh or not in step (5).

Description of Kerberos



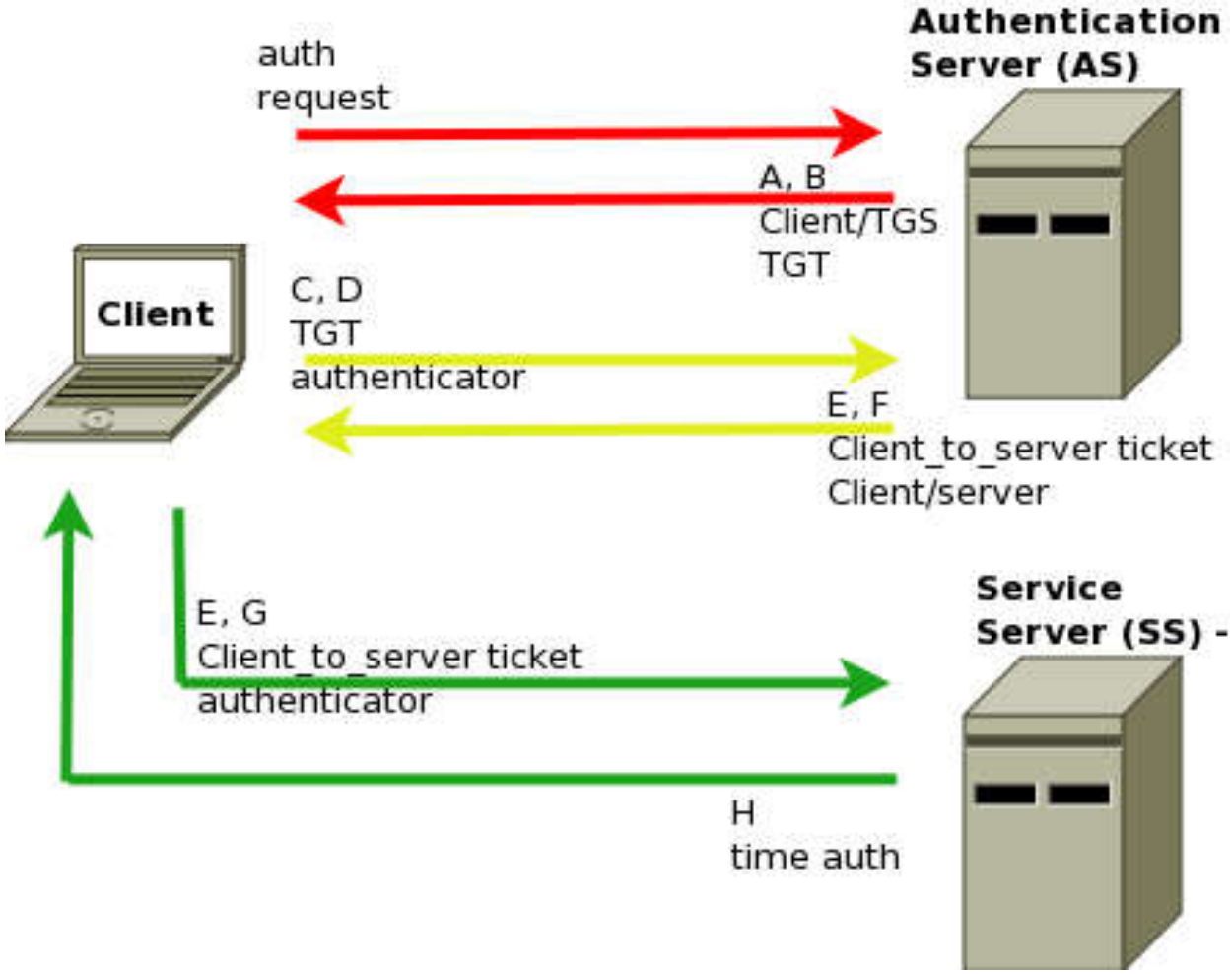
Description of Kerberos



1) service is the networked resource that the client is trying to access (e.g. Print Server);

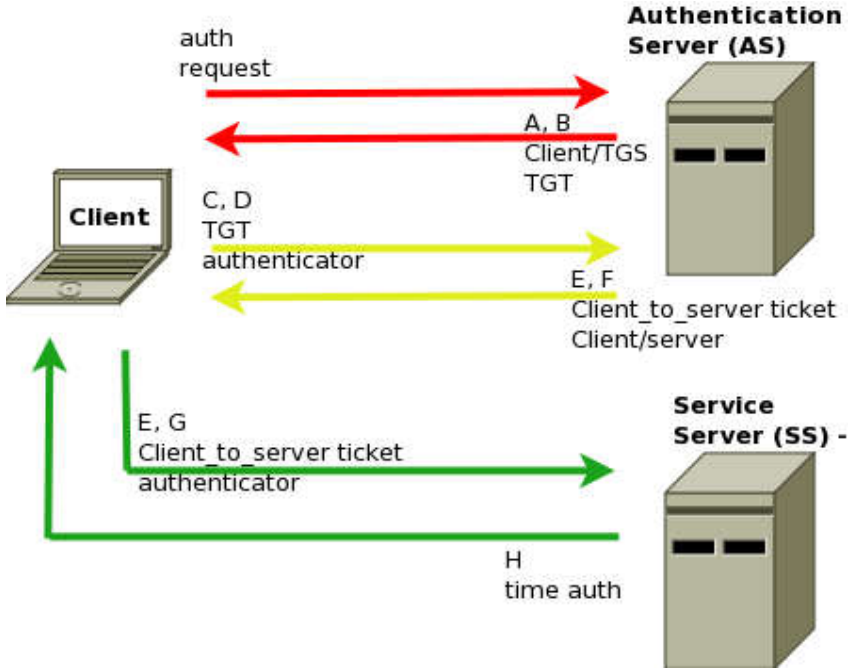
2)TGS is the Ticket Granting Server

Description of Kerberos



Description of Kerberos

- The client (machine) authenticates to the *AS* once using a long-term shared secret (e.g. a password) and receives a *TGT* from the *AS*.
- Later, when the client wants to contact some *SS*, it can (re)use this *TGT* to get additional tickets *ST*s from *TGS* for *SS*, without resorting to using the shared secret. These *ST*s can be used to prove authentication to *SS*.
  - *AS* = Authentication Server
  - *TGT* = Ticket-Granting Ticket
  - *TGS* = Ticket-Granting Server
  - *ST* = Service Ticket
  - *SS* = Service Server



Process of Kerberos

- The process includes
  - User Client-based Logon
  - Client Authentication
  - Client Service Authorization
  - Client Service Request
- KEYs include
  - K_{Client} the Master Key prenegotiated by *Client* and *AS*
  - K_{TGS} the symmetric key prenegotiated by *TGS* and *AS*
  - K_{SS} the symmetric key prenegotiated by *SS* and *TGS*
  - $K_{Client\text{-}TGS}$ the session key between *Client* and *TGS*
  - $K_{Client\text{-}SS}$ the session key between *Client* and *SS*
- Encryption and Decryption
  - They depend on the cipher-suite used
    - $E(K, -)$: encrypts message with key K;
    - $D(K, -)$: decrypts message with key K.

Process of Kerberos

- 用户 (*User*) 基于客户机程序 (*Client*) 登录的步骤：
  - 用户 *Bob* 输入用户名和口令密码到客户机 *Client*。上述的用户名和口令密码构成长期密钥 (Long-term Key)。长期密钥在本地保管，其加密的数据不应该在网络上传输以规避暴力攻击。
  - 客户机 *Client* 将 *Bob* 的口令密码通过 Hash 函数或其它内嵌规则转换成用于对称密码体系的 *Client/User* 主密钥 K_{Client}，这是客户机用户 *Bob* 和 *AS* 预先协商好的主密钥 (*Master Key*)。由于 Hash 函数的单向性，对主密钥的破解不会威胁到长期密钥的安全。
  - 同一台客户机 *Client* 的不同用户对应于不同的主密钥。主密钥由 *Client* 保管。
  - 后面涉及的 *Client-TGS* 会话密钥 $K_{Client\text{-}TGS}$ 以及 *Client-SS* 会话密钥 $K_{Client\text{-}SS}$ 是短期密钥或会话密钥 (Short-term Key / Session Key)，它们通过上述主密钥实现交换或发布。

Process of Kerberos

- 客户机 *Client* 身份认证步骤：
  - *Client* 向 *AS* 发送一个明文消息，代表用户请求服务。
    - 例如 “用户 *Bob* 请求服务”。
  - *AS* 检查数据库是否存在该用户 ID 的记录。确定的话 *AS* 返回以下两条消息给 *Client*：
    - 消息 *A*：$E(K_{Client}, K_{Client\text{-}TGS})$
    - 消息 *B*：$E(K_{TGS}, <client\ ID, client\ address, validity, K_{Client\text{-}TGS}>)$
      - 消息 *B* 是用 *TGS* 密钥加密的票据授权票据 *TGT*，包括客户 ID、客户网络地址、票据有效期、 *Client-TGS* 会话密钥。
    - 注意到 *Client* 无法解密消息 *B*。
  - *Client* 收到消息 *A* 和 *B* 后，应用 $D(K_{Client}, A)$ 得到 $K_{Client\text{-}TGS}$ 用于后续与 *TGS* 的通信。*Client* 将凭借消息 *B* 携带的有效的 *TGT* 向 *TGS* 证明其身份。

Process of Kerberos

- 客户机 *Client* 服务认证步骤：
  - 申请服务时，*Client* 向 *TGS* 发送以下两条消息：
    - 消息 *C*：*service ID*, *B*
    - 消息 *D*：$E(K_{Client\text{-}TGS}, <client\ ID, timestamp>)$
      - 消息 *D* 是用 $K_{Client\text{-}TGS}$ 会话密钥加密的认证。
  - *TGS* 从消息 *C* 中获得消息 *B*，应用 $D(K_{TGS}, B)$ 得到 $K_{Client\text{-}TGS}$，再应用 $D(K_{Client\text{-}TGS}, D)$ 得到认证内容，并返回给 *Client* 两条消息：
    - 消息 *E*：*service ID*, *ST*
      - $ST = E(K_{SS}, <client\ ID, client\ net\ address, validity, K_{Client\text{-}SS}>)$ 称为服务票据，包括客户 ID、客户网络地址、票据有效期限、 *Client-SS* 会话密钥。
    - 消息 *F*：$E(K_{Client\text{-}TGS}, K_{Client\text{-}SS})$
    - 注意到 *Client* 无法解密服务票据 *ST*。

Process of Kerberos

- 客户机 *Client* 服务申请步骤：
  - *Client* 应用 $D(K_{Client\text{-}TGS}, F)$ 得到 $K_{Client\text{-}SS}$，然后向 *SS* 发出以下两条消息：
    - 消息 *E*：由先前步骤得到的 *service ID*, *ST*
    - 消息 *G*：$E(K_{Client\text{-}SS}, <client\ ID, timestamp>)$
      - 消息 *G* 是用 $K_{Client\text{-}SS}$ 会话密钥加密的一个新的认证。
  - *SS* 应用 $D(K_{SS}, ST)$ 解密得到 $K_{Client\text{-}SS}$，再应用 $D(K_{Client\text{-}SS}, G)$ 解密得到认证 *G*，然后从中提取时间戳 *TS* = *timestamp*，返回 *Client* 一条消息 *H* 作为确认函以确认客户的身份真实，同意向该客户提供服务：
    - 消息 *H*：$E(K_{Client\text{-}SS}, <client\ ID, TS+1>)$
  - *Client* 应用 $D(K_{Client\text{-}SS}, H)$ 解密消息 *H*。如果其中的时间戳被正确更新，则 *SS* 可以信赖，*Client* 可以向 *SS* 发送服务请求。
  - 认证过程至此结束，*SS* 向 *Client* 客户机提供其所请求的服务。

Drawbacks & Limitations

- Single point failure (continuously available server)
- Demand clock synchronization
  - Network time protocol for synchronization
- The administration protocol is not standardized
- No general implementation standard
- All authentication in KDC (Key Distributing Center)

Drawbacks & Limitations

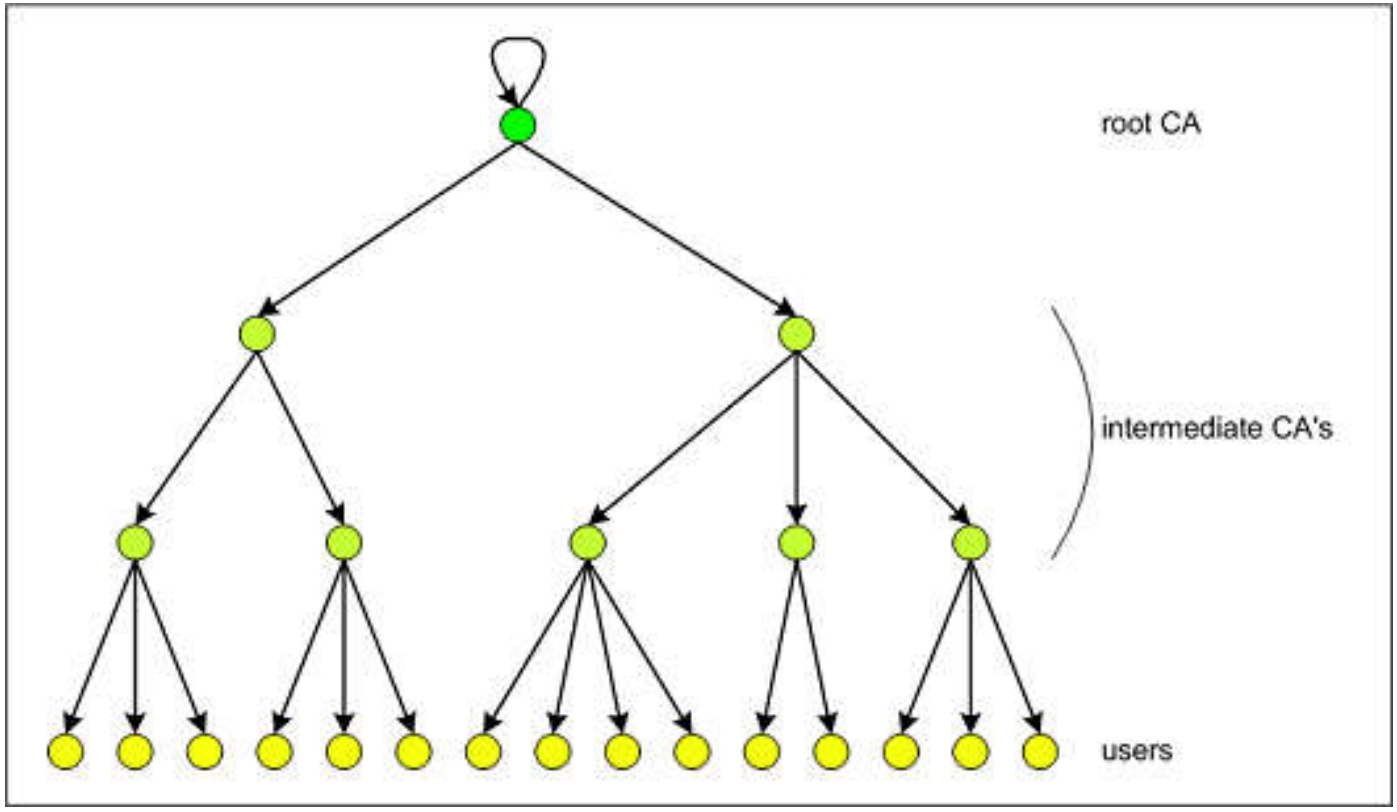
- 单点失败
  - 认证过程需要认证中心服务器 AS 的持续响应。这个缺陷可以通过使用复合 Kerberos 服务器和缺陷认证机制弥补。
  - 所有用户使用的主密钥都存储于 KDC 中，危及 KDC 安全的行为将危及所有用户的密钥。
  - 一个危险的客户机将可能危及用户密码安全。
- 时钟同步
  - Kerberos 要求参与通信的主机的时钟同步。票据具有一定有效期，如果主机的时钟与 Kerberos 服务器的时钟不同步将导致认证失败。默认设置要求时钟的时间差不超过10分钟，通常用网络时间协议后台程序保持时钟同步。
- 管理协议未标准化 (RFC 3244 描述了一些更改)。

X.509

- X.509 is an ITU-T standard for a public key infrastructure (PKI) for single sign-on (SSO) and Privilege Management Infrastructure (PMI).
  - SSO, 单点登录
  - PMI, 特权管理基础架构
- X.509 specifies, amongst other things, standard formats for public key certificates, certificate revocation lists, attribute certificates, and a certification path validation algorithm.
- Key words
  - Certificate
  - CA (Certificate Authority)
  - Hierarchy
- History and Versions
  - Version 1 (1988): issued and associated with the X.500 Standard
  - Version 2 (not used widely): include Subject and Issuer identifier
  - Version 3 (1996): Extensions added
    - PKIX
  - RFC 5280

X.509

- Hierarchy



X.509 Certificate

- In the X.509 system, a Certification Authority (CA) issues a certificate binding a public key to a particular distinguished name in the X.500 tradition, or to an alternative name such as an e-mail address or a DNS-entry.
- An organization‘s trusted root certificates (根证书) can be distributed to all employees so that they can use the company PKI system.
- Browsers such as IE, Netscape/Mozilla, Opera, Safari and Chrome come with root certificates pre-installed, so SSL certificates from larger vendors will work instantly; in effect the browsers' developers determine which CAs are trusted third parties for the browsers' users.

X.509 Certificate

- Structure of Certificate
  - Certificate
    - Version
    - Serial Number
    - Algorithm ID
    - Issuer (CA's name)
    - Validity
      - Not Before
      - Not After
    - Subject
    - Subject Public Key Info
      - Public Key Algorithm
      - Subject Public Key
    - Issuer Unique Identifier (Optional)
    - Subject Unique Identifier (Optional)
    - Extensions (Optional)
    - Certificate Signature Algorithm
    - Certificate Signature

X.509 Certificate

- Example.
  - 证书的明文格式

```
Certificate:
    Data:
        Version: 1 (0x0)
    Serial Number: 7829 (0x1e95)
    Signature Algorithm: md5WithRSAEncryption
    Issuer: C=ZA, ST=Western Cape, L=Cape Town, O=Thawte
Consulting cc, OU=Certification Services Division, CN=Thawte
Server, CA/emailAddress=server-certs@thawte.com
    Validity:
        Not Before: Jul  9 16:04:02 1998 GMT
        Not After : Jul  9 16:04:02 1999 GMT
    Subject: C=US, ST=Maryland, L=Pasadena, O=Brent Baccala,
OU=FreeSoft, CN=www.freesoft.org/emailAddress=baccala@freesoft.org
```

X.509 Certificate

- Example.
  - 证书的明文格式

```
Certificate:
    Subject Public Key Info:
    Public Key Algorithm: rsaEncryption
    RSA Public Key: (1024 bit)
        Modulus (1024 bit):
            00:b4:31:98:0a:c4:bc:62:c1:88:aa:dc:b0:c8:bb:33:
            35:19:d5:0c:64:b9:3d:41:b2:96:fc:f3:31:e1:66:36:
            d0:8e:56:12:44:ba:75:eb:e8:1c:9c:5b:66:70:33:52:
            14:c9:ec:4f:91:51:70:39:de:53:85:17:16:94:6e:ee:
            f4:d5:6f:d5:ca:b3:47:5e:1b:0c:7b:c5:cc:2b:6b:c1:
            90:c3:16:31:0d:bf:7a:c7:47:77:8f:a0:21:c7:4c:d0:
            16:65:00:c1:0f:d7:b8:80:e3:d2:75:6b:c1:ea:9e:5c:
            5c:ea:7d:c1:a1:10:bc:b8:e8:35:1c:9e:27:52:7e:41:8f
        Exponent: 65537 (0x10001)
```

X.509 Certificate

- Example.
  - 证书的明文格式

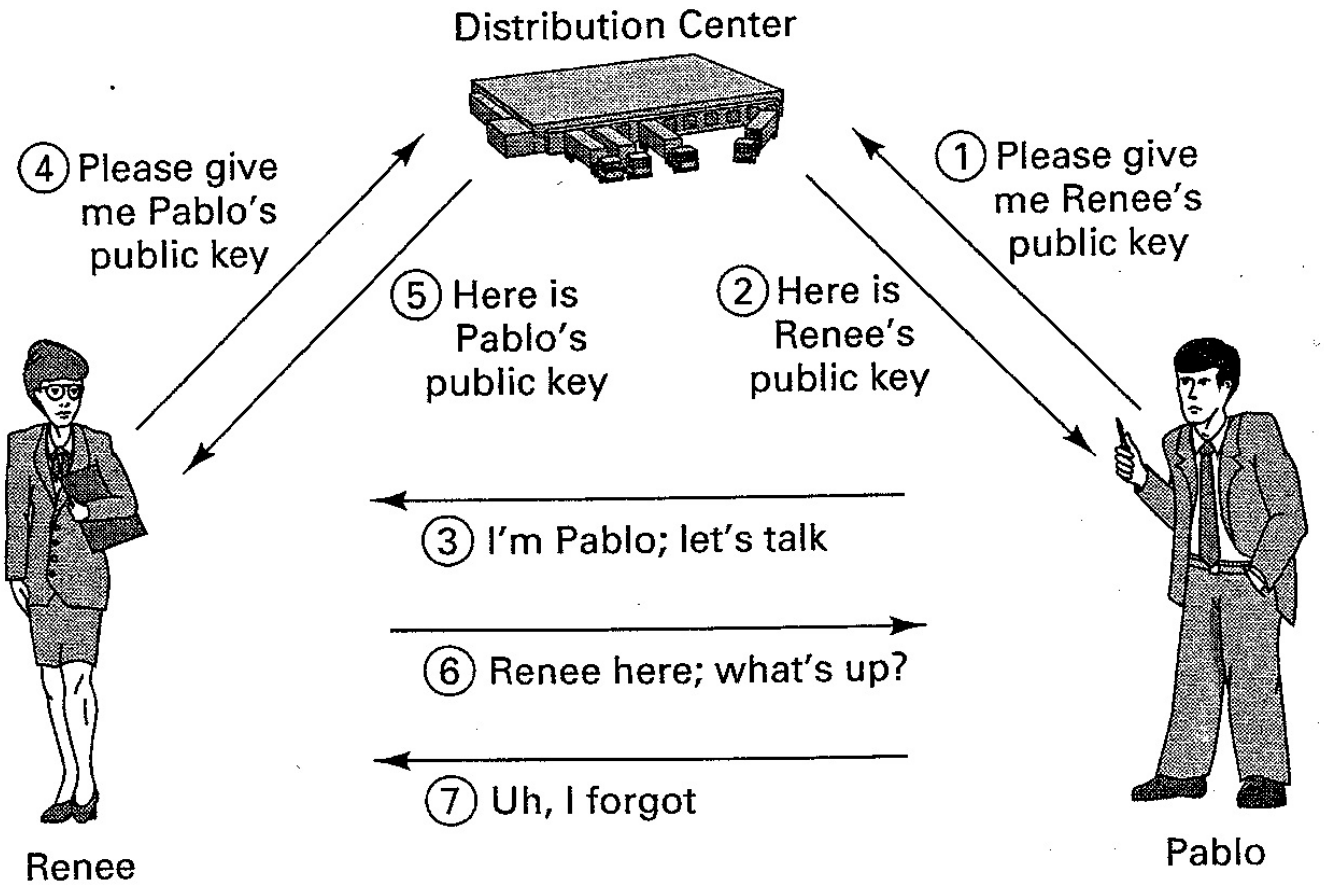
```
Certificate:
    Signature Algorithm: md5WithRSAEncryption
93:5f:8f:5f:c5:af:bf:0a:ab:a5:6d:fb:24:5f:b6:59:5d:9d:
92:2e:4a:1b:8b:ac:7d:99:17:5d:cd:19:f6:ad:ef:63:2f:92:
ab:2f:4b:cf:0a:13:90:ee:2c:0e:43:03:be:f6:ea:8e:9c:67:
d0:a2:40:03:f7:ef:6a:15:09:79:a9:46:ed:b7:16:1b:41:72:
0d:19:aa:ad:dd:9a:df:ab:97:50:65:f5:5e:85:a6:ef:19:d1:
5a:de:9d:ea:63:cd:cb:cc:6d:5d:01:85:b5:6d:c8:f3:d9:f7:
8f:0e:fc:ba:1f:34:e9:96:6e:6c:cf:f2:ef:9b:bf:de:b5:22:
68:9f
```

- **X.509 Certificate**
  - Certificates Issuing
    - CA distributing
    - Certificate transported by link
      - Feature
        - Everyone with CA's public key can get the public key in the CA's certificate.
        - Only CA can modify the certificate.
      - CA can exchange their PK safely.
  - Certificates Revoking
    - Why revoking
      - Every certificate has a validity.
      - User don't think the key is secure.
      - User don't trust CA.
      - CA think the certificate is unsecure.
    - CRL (Certificate Revocation List)
      - Certificates revoked by not out of valid time

- **Security Problems with X.509**
  - Specification
    - Complexity and lack of quality
  - Architectural flaw
  - Commercial certificate authorities
  - Implementation

Security Problems with X.509

- Specification Complexity and Lack of Quality.



Applications with X.509

- S/MIME (Multipurpose Internet Mail Extensions)
- SSL (Secure Socket Layer)
- TLS (Transport Layer Security)
- SET (Secure Electronic Trade)
- PKI (Public Key Infrastructure)
- ……