网络安全技术

刘振

上海交通大学 计算机科学与工程系

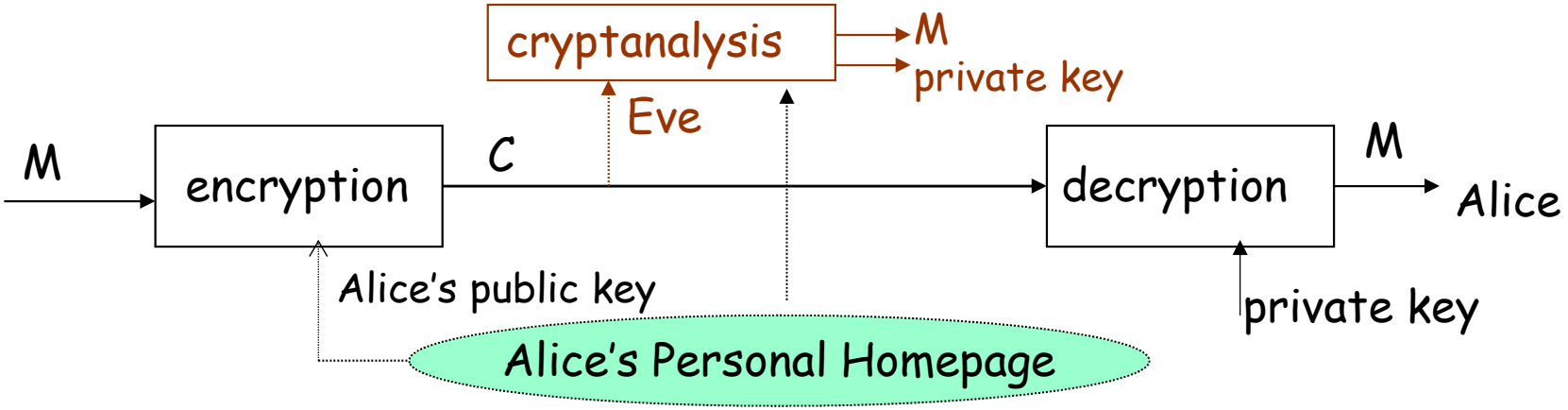
电信群楼3-509

liuzhen@sjtu.edu.cn

Public Key Infrastructure

Digital Certificates

- Public key encryption: encrypt using receiver's public key
  - sender **has to be sure** that the public key used for encryption is indeed the receiver's public key
- Digital signature: verify a signature
  - Verifier **has to be sure** that the public key used for signature verification is indeed the signer's public key
- **How can the encryptor / verifier be sure that the public key is authentic?**



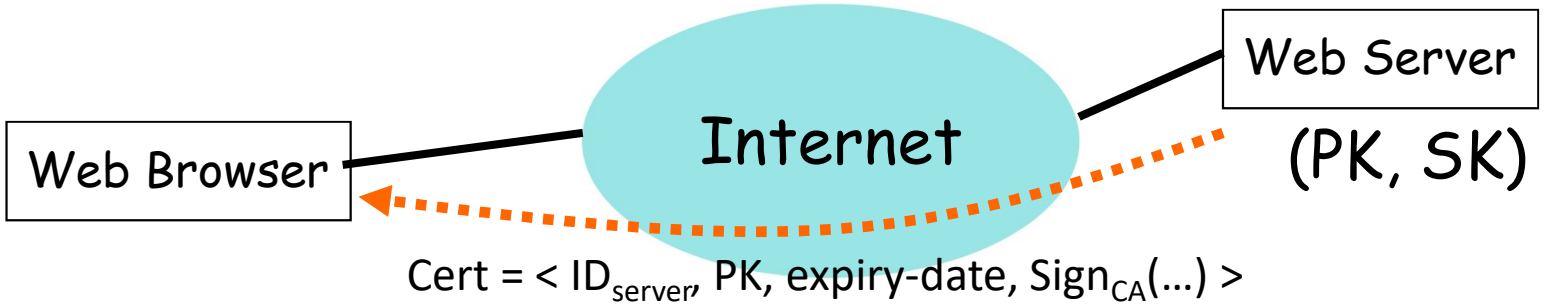
- How about posting the public key at a personal homepage?
- How about sending the public key to the encryptor / verifier using email?

Digital Certificates

- How it works:
  - There is an entity called Certification Authority (CA) in the system
  - CA has a public key which is **ASSUMED** to be well known
    - e.g. built-in, preinstalled into all the web browsers/operating systems
  - CA issues a certificate to each public key owner
  - The certificate bears (1) the public key owner's identity, (2) the public key, (3) a validity period of the certificate and (4) the CA's signature
  - By using the certificate, the CA vouches that the public key in the certificate is owned by the public key owner.
  - The CA publishes a Certification Practice Statement (CPS) that specifies the policies (including liabilities) governing the use of the certificates issued.
- Only the CA can create a legitimate certificate
  - Only the CA can generate the signature in the certificate which requires the knowledge of CA's private key
- Anyone can verify the authenticity of the certificate using CA's public key

$Cert_A = (ID_A, PK_A, \text{expiry-date}, Sign_{CA}(ID_A, PK_A, \text{expiry-date}))$

How to Use a Certificate – An example of Secure Web Browsing



- The web browser has the CA's public key built in.
  - **The legitimacy of the web browser software becomes crucial for ensuring the security of digital certificates**
  - A certificate is **NO** more secure than the security of the web browser
- In practice, each browser trusts multiple CAs rather than just one
- Exercise: find out the number of CAs that your IE and Firefox trust

What's Inside a Certificate (X.509)

User Name
Certificate Version
Validity Period
Serial No
User's Public Key
Other user attributes
CA's name
CA's signature (of all the above)

e.g.

User Name (Common Name): http://www.hangseng.com/

Validity Period: 2018/11/27 – 2019/11/28

User's Public Key: RSA (2048 bits)

Modulus (2048 bits): 30 82 01 0a 02 82 01…

Exponent (24 bits): 01 00 01

CA's name (Issuer): VeriSign Class 3 Extended Validation SSL SGC CA

CA's signature (Certificate Signature Value): **Size: 256 Bytes / 2048 Bits**

There are many other attributes: Certificate serial no., certificate version number, HSBC public key algorithm, CA's signing algorithm, etc.

$Cert_A = (ID_A, PK_A, \text{expiry-date}, \ldots, Sign_{CA}(ID_A, PK_A, \text{expiry-date}, \ldots))$

Some Remarks on Digital Certificates

- Certificate authority (CA) is *considered as* a Trusted Third Party (TTP) that issues and signs certificates
  - Verifying CA's signature in a certificate only verifies the **binding validity** between the public key and the identity in the certificate vouched by the CA
  - Verifying CA's signature does **not** verify the identity of the source that the certificate comes from!
    - E.g. Alice may receive Carol's certificate from Bob
  - Certificates are public!
  - Common format for certificates is ITU-T X.509.

Certificate Revocation

- There are cases when a certificate has to be made invalid before its expiry date
  - For example, when an employee leaves an organization, or when a participant's private key has been compromised.
- Certificate Revocation List (CRL)
  - The CA should periodically, or on demand basis, distribute CRL (which is signed by the CA) listing the serial numbers of the certificates that have been revoked.
  - A participant using a certificate should check the latest CRL from the CA, to determine if the certificate is still valid.

PKI

- Public Key Infrastructure (PKI) consists of all pieces needed to securely use public key cryptography
  - Key generation and management
  - Certification authorities, digital certificates
  - Certificate revocation lists (CRLs)
- No general standard for PKI
- We consider a few models of PKI

PKI Trust Models

Monopoly model

- One universally trusted organization is the CA for the known universe
- Big problems if CA is ever compromised
- Big problem if you don't trust the CA!
  - Should Chinese trust VeriSign (US)?

PKI Trust Models

- Anarchy model
  - Everyone is a CA!
  - Users must decide which “CAs” to trust
  - Used in PGP (Pretty Good Privacy)
    - www.pgpi.org
  - Why do they call it “anarchy”?
    - Suppose cert. is signed by Frank and I don’t know Frank, but I do trust Bob and Bob vouches for Frank. Should I trust Frank?
    - Suppose cert. is signed by Frank and I don’t know Frank, but I do trust Bob and Bob says Alice is trustworthy and Alice vouches for Frank. Should I trust Frank?

PGP – Anarchy Model

- **Unstructured**
  - Suppose a public key is received and claimed to be Alice's.
  - The public key and Alice's identity are signed by some others (CAs). Each signature is considered as a certificate:

$$\text{Cert}_{\text{Bob}}(\text{Alice}),\ \text{Cert}_{\text{Carol}}(\text{Alice}),\ \text{Cert}_{\text{Dave}}(\text{Alice}),\ \text{Cert}_{\text{Eve}}(\text{Alice})$$

Example: if my trust in certificates issued by Bob, Carol, and Dave (whose public keys I already have valid copies) are 1/2, 1/3, 1/3, respectively (and I don't have Eve's public key), then the above public key for Alice is considered as trust-worthy as $1/2 + 1/3 + 1/3 \geq 1$

PKI Trust Models

- Oligarchy
  - Multiple trusted CAs
  - Used today
  - Browser may have tens of root CAs' public keys build-in
  - User can decide which CAs to trust (by default, you trust what the browser said so)

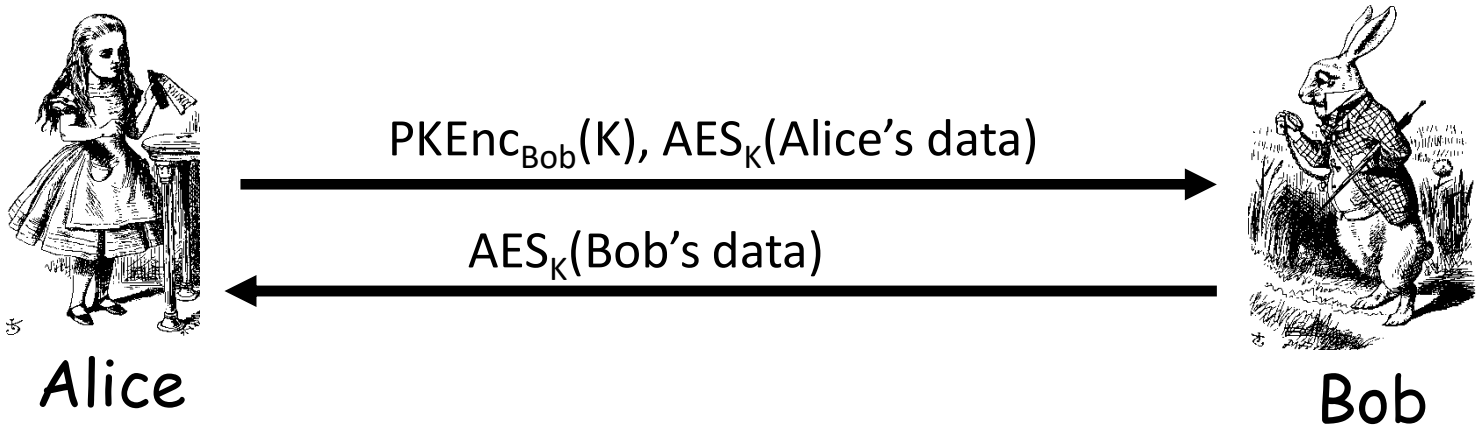
Hybrid Encryption

Hybrid Encryption

- Secret key encryption scheme
  - Significantly more efficient than public key encryption.
  - Has major problem in key distribution.
- Public key encryption scheme
  - Slow.
  - No key distribution problem.
- Hybrid encryption scheme uses
  - Public key encryption to avoid key distribution problem.
  - Secret key encryption to do bulk encrypting for efficiency.
- Most crypto packages in use today are hybrid encryption schemes.
  - E.g. PGP, SSL, IPSec, S/MIME

Hybrid Encryption

- Public key crypto to establish a key
- Symmetric key crypto to encrypt data



- The symmetric key, K, is usually much shorter than the plaintext (i.e. Alice's data and Bob's data). Hence the speed advantage obtaining by using AES_K(plaintext) is not nullified by the public key encryption of K.

- Can Bob be sure he's talking to Alice?
- Can Alice be sure that she's talking to Bob?

Summary

- **Digital Certificate**
- **PKI**
- **Hybird Encryption**