Public-Key Infrastructure

Haipeng Dai

haipengdai@nju.edu.cn
313 CS Building
Department of Computer Science and Technology
Nanjing University

Public Key Infrastructure

- In public key cryptography, we have assumed that everyone knows everyone else's public key. However, this is not easy to achieve.
  - How does Alice know that the public key she received is really Bob's public key?

- A public key infrastructure (PKI) consists of all necessary components to securely distribute public keys.

- Basic idea: use certificate to distribute public keys. We need
  - Entities for creating certificates
  - Methods for evaluating certificates
  - Methods for revoking certificates

Distribution of Public Keys

- Original paper on public-key cryptography proposed the use of a Public File: Public-key white pages

- Public-key certificate
  - Signed statement specifying the key and identity
    - "Bob", PK_B, $sig_{authority}(MD("Bob", PK_B))$

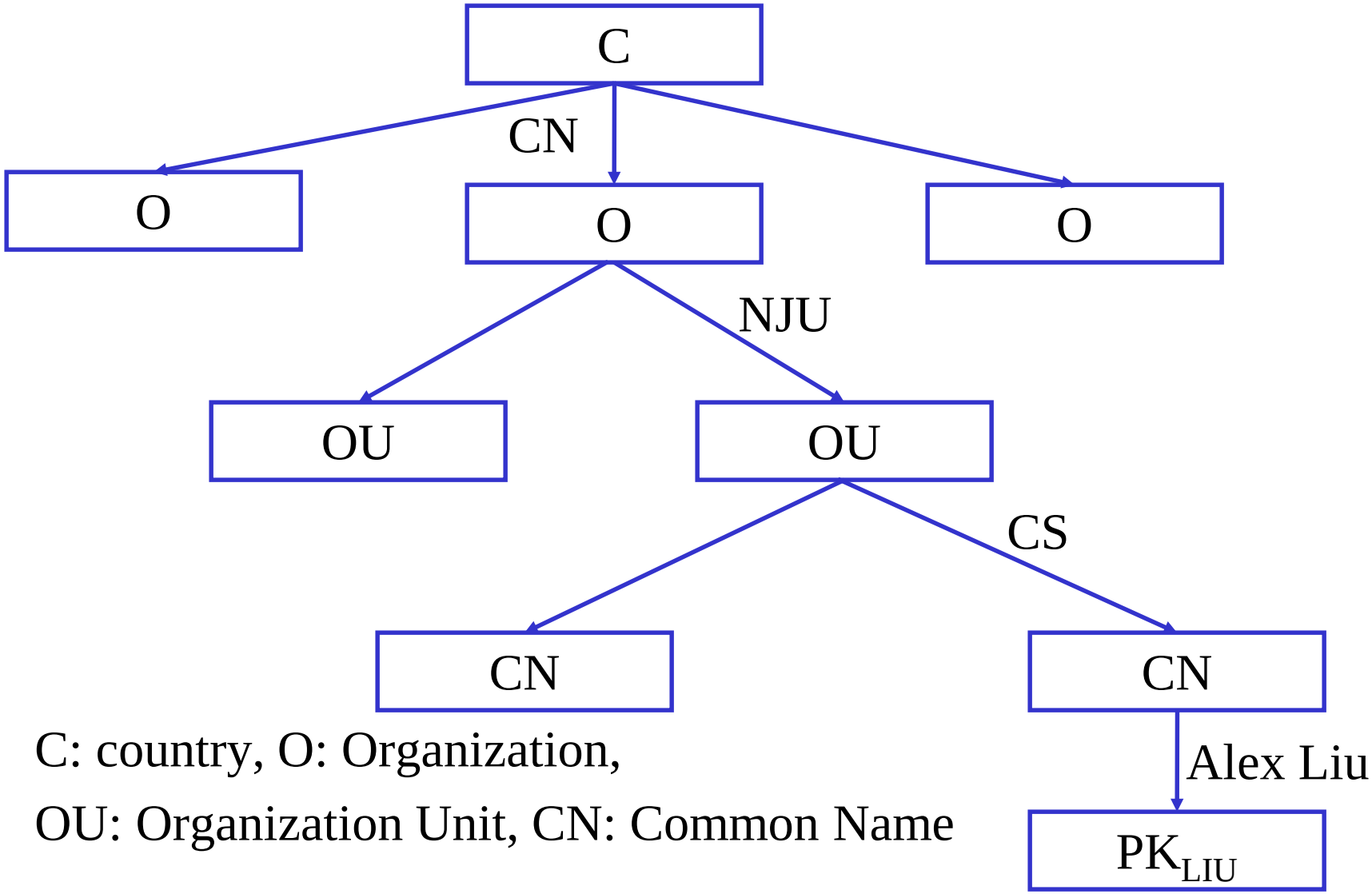
- Basic idea: Authenticity of many public keys is reduced to the authenticity of one key (the public key of the authority).

- Who is the authority that you trust?

PKI Trust Models 1: Monopoly Model

- The world has only one CA trusted by everyone in the world.
  - The public key of this CA is embedded in all software and hardware as the PKI trust anchor.
- Loren Kohnfelder BS MIT thesis in 1978
  - Offline CA signs (name + key) to bind the two in a certificate
  - Online directory distributes certificates
- OSI proposed in X.500 a global directory run by monopoly telecommunication companies
  - Hierarchical database (or data organization, or both)
  - Path through the directory/database to keys is defined by a series of Relative Distinguished Names (RDNs)
  - Collection of RDNs form a Distinguished Name (DN)
  - Data being looked up is found at the end of the RDN path

Hierarchical Directory Structure



- C: country, O: Organization,
- OU: Organization Unit, CN: Common Name

Problems of the Monopoly Model

- There is no such a universally trusted organization.
- Infeasible to change that CA's public key, if it is compromised.
  - All software and hardware are preconfigured with the CA's public key
- Difficult to certify.
  - How does the CA can certify your identity? Just because you paid?
- Single point of failure.
  - What if that CA has a corrupt employee?
- The world CA will charge monopoly price for certification.
- Concerns about the use of world directory
  - Companies don't like making their internal structure public
    - Directory for corporate headhunters
  - Privacy concerns
    - Directory of single women
    - Directory of teenage children

PKI Trust Model 2: Monopoly + Registration Authorities (RAs)

- Same as the monopoly model except that the world CA uses some other organizations to certify identities of users.
- Getting certified becomes easier.
- All other drawbacks of the monopoly model still applies.

PKI Trust Model 3: Monopoly + Delegated CAs

- Similar to model 2 except that it uses CAs instead of RAs
- Has a hierarchical structure: root certificate authority signs certificates for lower-level authorities, lower-level authorities sign certificates for individual users, and so on.
- User sees a chain of certificates, instead of one certificate
  - "NJU", PK_{NJU}, $sig_{Verisign}$("NJU", PK_{NJU}),
    "Alex Liu", PK_{LIU}, sig_{NJU}("Alex Liu", PK_{LIU})

PKI Trust Model 4: Oligarchy – multiple CAs for the world

- The world trust multiple CAs. This is the model used today for web browsers.
- Web browsers today come shipped with the public keys of about 80 CAs.
- Problems:
  - Any of the trust anchor organizations getting comprised will put the security of the world into risk.
  - The trust anchor organizations are trusted by the vendor, not by the user.
  - It is easy to trick a naïve user to add a bogus trust anchor into the set:
    - Warning: This was signed by an unknown CA. Would you like to accept the certificate anyway? (OK)
    - Would you like to always accept this certificate without being asked in the future? (OK)
    - Would you like to always accept certificates from the CA that issued the certificate? (OK)
    - Would you like to always accept certificates from any CA? (OK)
    - Since you are willing to trust anyone for anything, would you like me to make random edits to the files on your hard drive without bothering you with a pop-up box? (OK)
  - Use of public machine. What if the previous user added a malicious anchor?

Getting your CA key into Browsers

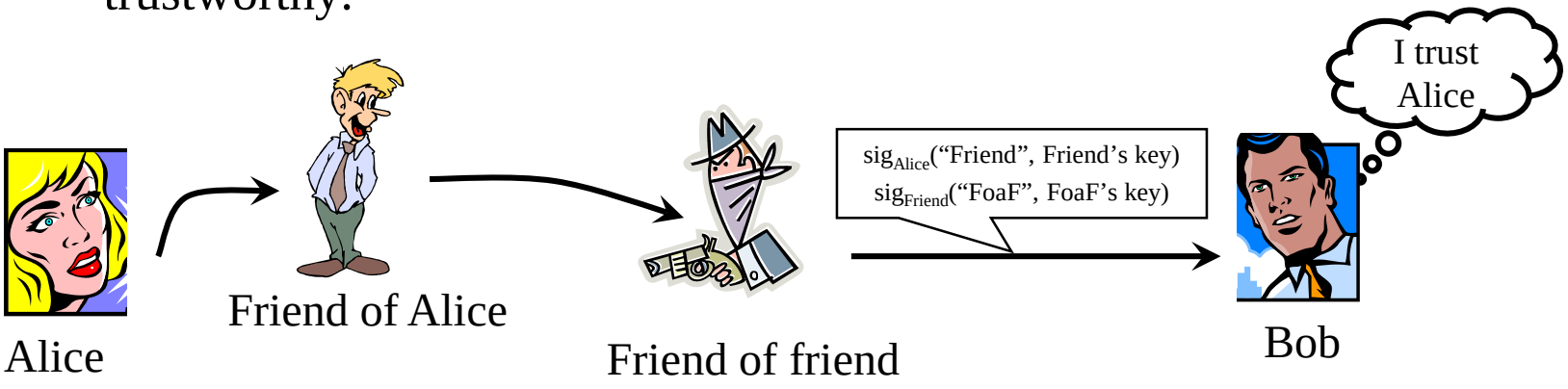
- Total cost: half a million USD per browser
- NetScape: hand me the cash and a floppy
- MSIE: No special charge, but you must pass an SAS70 electronic data security audit
  - US CPA Statement on Auditing Standards 70
  - Lengthy (up to 6 months), expensive, and painful
  - Infrastructure, policy, staff, and auditing costs run to half a million
- CA keys are bought and sold on the secondary market
  - Equifax's certificates are actually owned by Geotrust

PKI Trust Model 5: Anarchy Model, no CAs

- This is the model used in PGP.
- Each user is responsible for configuring their trust anchors.
- Anyone can sign the certificate of anyone else.
- There is a central database storing certificates that people put in.
- To verify a certificate, you can search the DB to see whether you can find a path from one of your trust anchors to that certificate.

Problems of the Anachy Model

- The database can get unworkably large if it were deployed on Internet scale.
  - Every person donates 10 certificates.
  - There are 5-6 billion people.
  - 50-60 billion certificates in the database.
  - Even worse, you need to construct a path!
- Trust the root, but may not trust the chain.
  - What if Alice's friend's friend is a bad guy?
  - Therefore, this model works in a small community where the users are trustworthy.



PKI Trust Model 6: Name Constraints

- Why trust is a binary value: either fully trust or fully distrust?

- A CA is trusted for a domain of users.
  - NJU is trusted for signing anyname.nju.edu.cn, but not something.sjtu.edu.cn

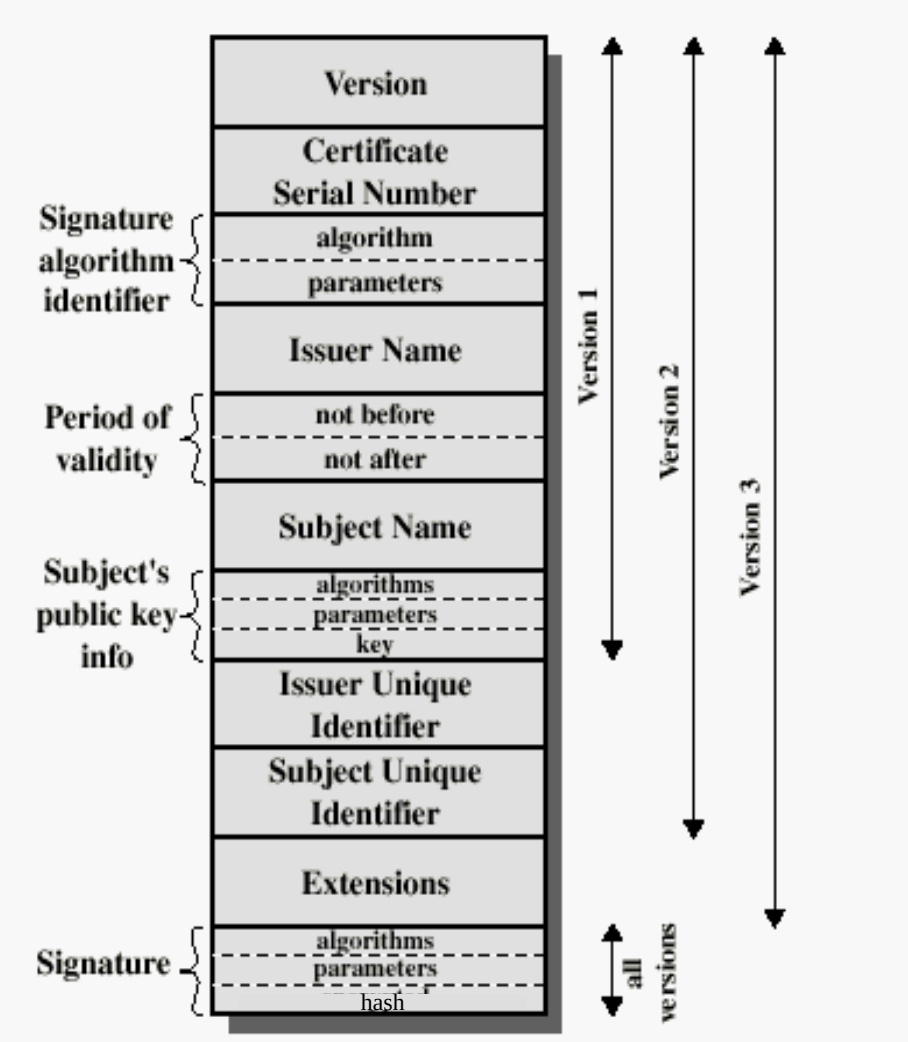
PKI Trust Model 7: Hierarchical Name Constraints

- Similar to the model of Monopoly + Delegated CAs, except that each delegated CA can only issue certificate for their portion of the name space.

X.509 Authentication Service

- Internet standard (1988-2000)
- Specifies certificate format
  - X.509 certificates are used in IPsec and SSL/TLS
- Specifies certificate directory service
  - For retrieving other users' CA-certified public keys
- Specifies a set of authentication protocols
  - For proving identity using public-key signatures
- Does not specify crypto algorithms
  - Can use it with any digital signature scheme and hash function, but hashing is required before signing

X.509 Certificate



Important Things in X.509 Certificate

- Country: C=CN
- Organization: O=Nanjing University
- Organization Unit: OU=Department of Computer Science and Technology
- Common Name: CN=Alex Liu

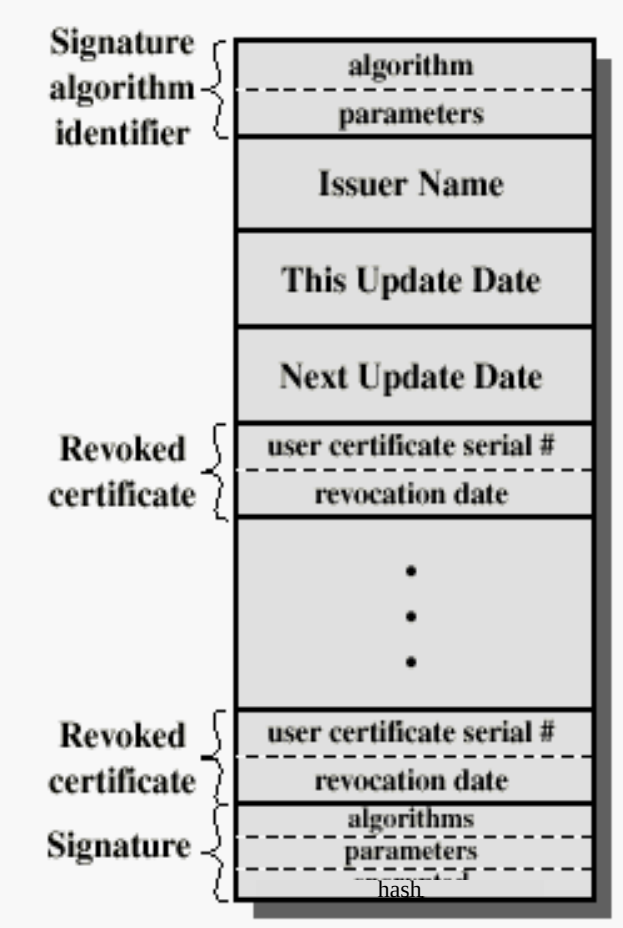
Certificate Revocation

- Revocation is very important
- Many valid reasons to revoke a certificate
  - Private key corresponding to the certified public key has been compromised
  - User stopped paying his certification fee to this CA and CA no longer wishes to certify him
  - CA's certificate has been compromised!
- Expiration is a form of revocation, too
  - Many deployed systems don't bother with revocation
  - Re-issuance of certificates is a big revenue source for certificate authorities

Certificate Revocation Mechanisms

- Online revocation service
  - When a certificate is presented, recipient goes to a special online service to verify whether it is still valid
    - Like a merchant dialing up the credit card processor
- Certificate revocation list (CRL)
  - CA periodically issues a signed list of revoked certificates
    - In 1970s, credit card companies used to issue thick books of canceled credit card numbers
  - Can issue a “delta CRL” containing only updates

X.509 Certificate Revocation List



Because certificate serial numbers must be unique within each CA, this is enough to identify the certificate

CRL Problems

- CRLs don't work in reality
  - Blacklist approach was abandoned by credit card vendors 20 years ago because it didn't work properly
- CRLs mirror credit card blacklist problems
  - Not issued frequently enough to be effective against an attacker
  - Expensive to distribute
  - Vulnerable to simple DOS attacks
    - Attacker can prevent revocation by blocking CRL delivery
- CRLs add further problems of their own
  - Can contain retroactive invalidity dates
  - CRL issued right now can indicate that a cert was invalid last week
    - Checking that something was valid at time *t isn't sufficient* to establish validity
    - Back-dated CRL can appear at any point in the future
  - Destroys the entire concept of nonrepudiation

CRL Problems (continued)

- Revoking self-signed certificates is hairy
  - Cert revokes itself
  - Applications may
    - Accept the CRL as valid and revoke the certificate
    - Reject the CRL as invalid since it was signed with a revoked certificate
    - Crash
- CRL Distribution Problems
  - CRLs have a fixed validity period
    - Valid from *issue date to expiry date*
  - At *expiry date, all relying parties connect to the CA to fetch the* new CRL
    - Massive peak loads when a CRL expires (DDOS attack)
  - Issuing CRLs to provide timely revocation exacerbates the problem
  - 10M clients download a 1MB CRL issued once a minute =~150GB/s traffic
  - Even per-minute CRLs aren't timely enough for high-value transactions with interest calculated by the minute

CRL Alternative: Online Certification Status Protocol

- OCSP is an Internet protocol used for obtaining the revocation status of an X.509 digital certificate.
- Described in RFC 2560.
- The "request/response" nature of these messages leads to OCSP servers being termed OCSP responders.
- An OCSP responder may return a signed response signifying that the certificate specified in the request is 'good', 'revoked' or 'unknown'. If it cannot process the request, it may return an error code.
- The OCSP request format supports additional extensions. This enables extensive customization to a particular PKI scheme.
- How does OCSP prevents replay attacks?
  - OCSP can be resistant to replay attacks, where a signed, 'good' response is captured by a malicious intermediary and replayed to the client at a later date after the subject certificate may have been revoked. OCSP overcomes this by allowing a nonce to be included in the request that must be included in the corresponding response.

OCSP Example

- 1. Alice and Bob have public key certificates issued by Ivan, the Certificate Authority (CA).
- 2. Alice wishes to perform a transaction with Bob and sends him her public key certificate.
- 3. Bob, concerned that Alice's private key may have been compromised, creates an 'OCSP request' that contains Alice's certificate serial number and sends it to Ivan.
- 4. Ivan's OCSP responder looks up the revocation status of Alice's certificate (using the certificate serial number Bob informed) in his own CA database. If Alice's private key had been compromised, this is the only trusted location at which the fact would be recorded.
- 5. Ivan's OCSP responder confirms that Alice's certificate is still OK, and returns a signed, successful 'OCSP response' to Bob.
- 6. Bob cryptographically verifies the signed response (He has Ivan's public key on-hand – Ivan is a trusted responder) and ensures that it was produced recently.
- 7. Bob completes the transaction with Alice.

Revocation/Status Checking in the Real World

- In practice, revocation checking is turned off in user software
  - Because it slows everything down a lot
- CRLs are useful in special-case situations where there exists a statutory or contractual obligation to use them
  - Relying party needs to be able to claim CRL use for due diligence purposes or to avoid liability
- CA key compromise
  - Sun handled revocation of their CA key via posts to mailing lists and newsgroups