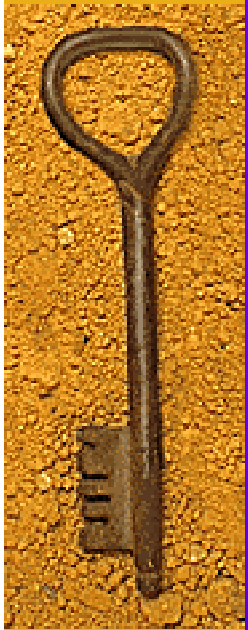
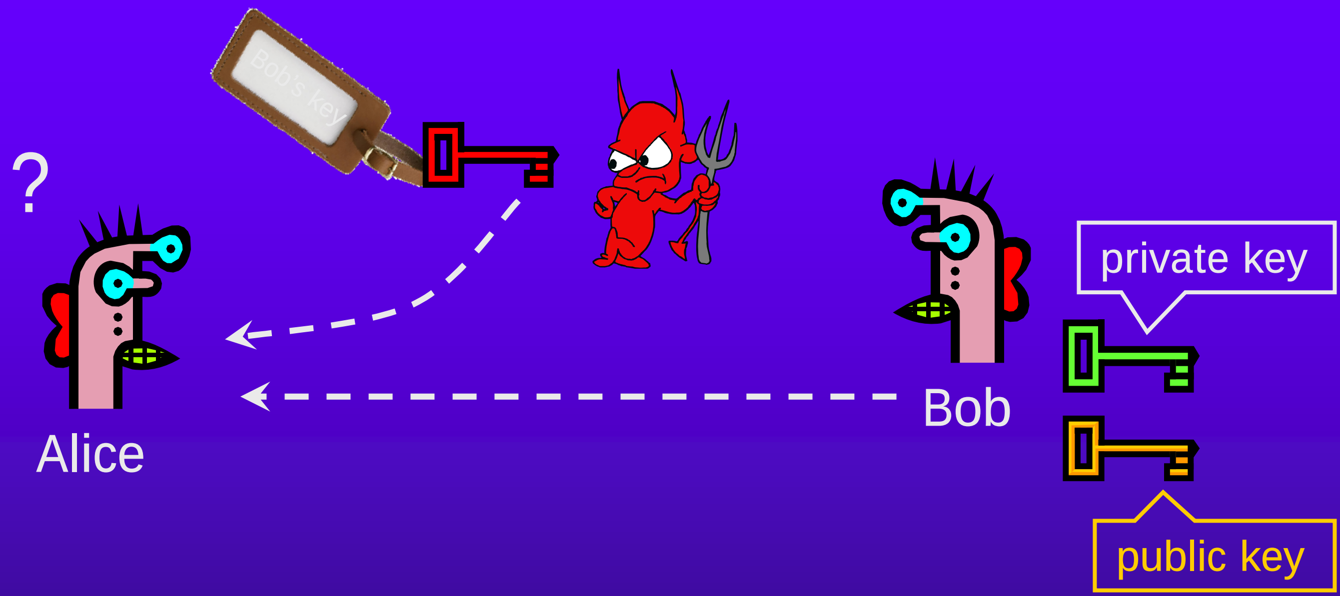
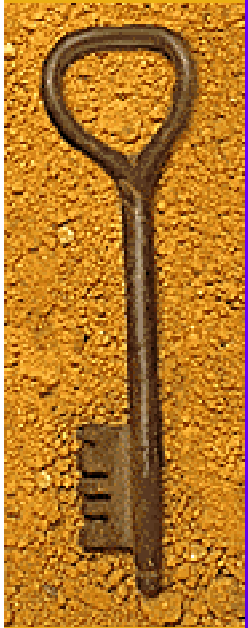
Public-Key Infrastructure (PKI)



Authenticity of Public Keys



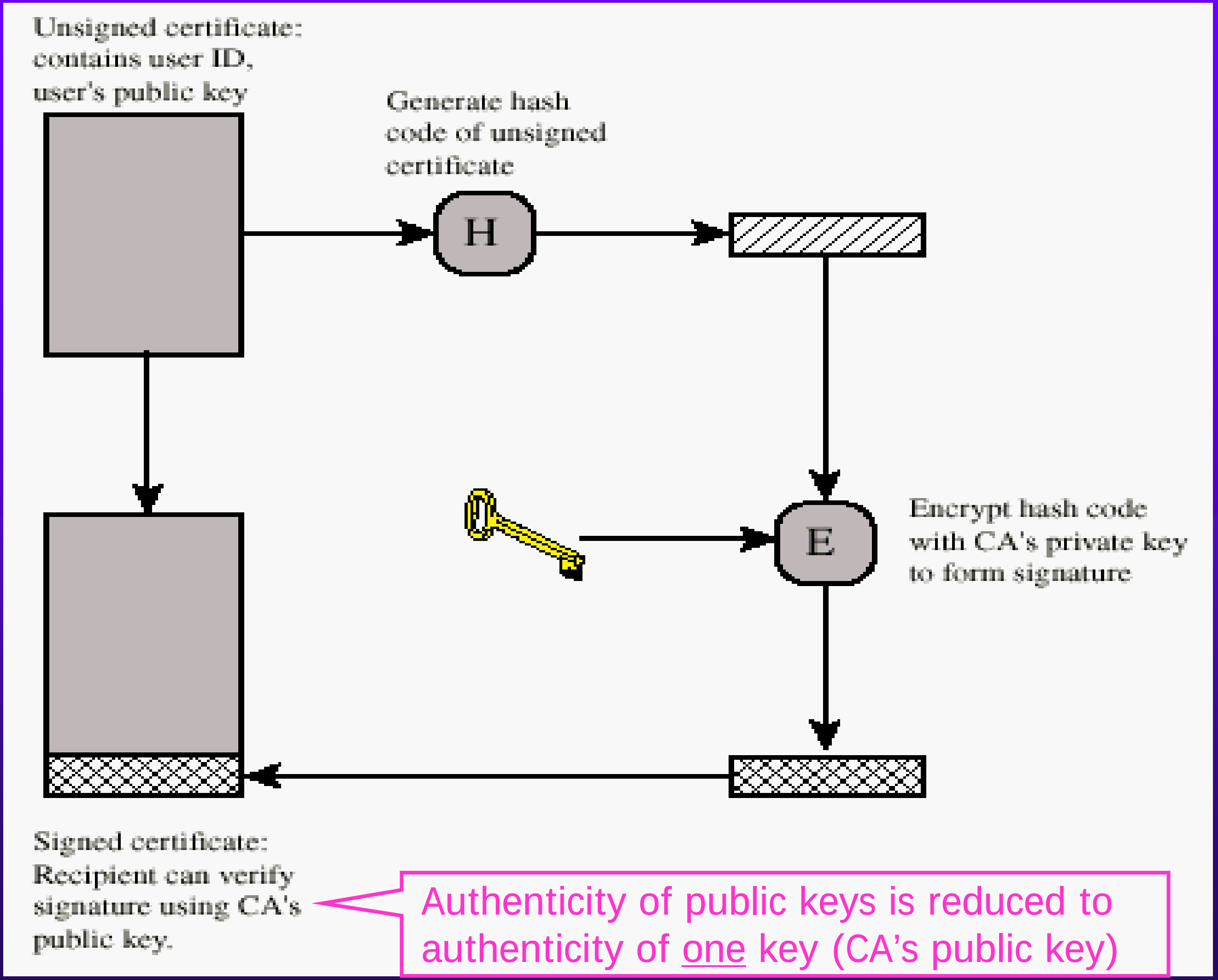
Problem: How does Alice know that the public key she received is really Bob's public key?



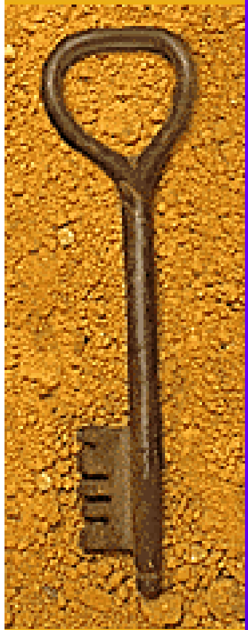
Distribution of Public Keys

- Public announcement or public directory
  - Risks: forgery and tampering
- Public-key certificate
  - Signed statement specifying the key and identity
    - sig_{Alice}(“Bob”, PK_B)
- Common approach: certificate authority (CA)
  - Single agency responsible for certifying public keys
  - After generating a private/public key pair, user proves his identity and knowledge of the private key to obtain CA’s certificate for the public key (offline)
  - Every computer is pre-configured with CA’s public key

Using Public-Key Certificates

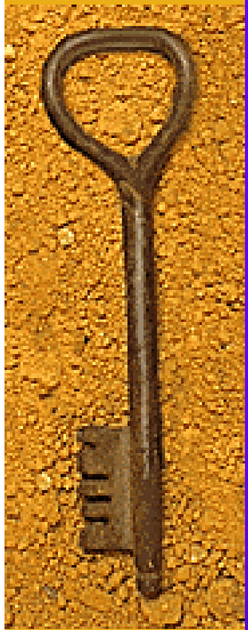


Authenticity of public keys is reduced to authenticity of <u>one</u> key (CA's public key)



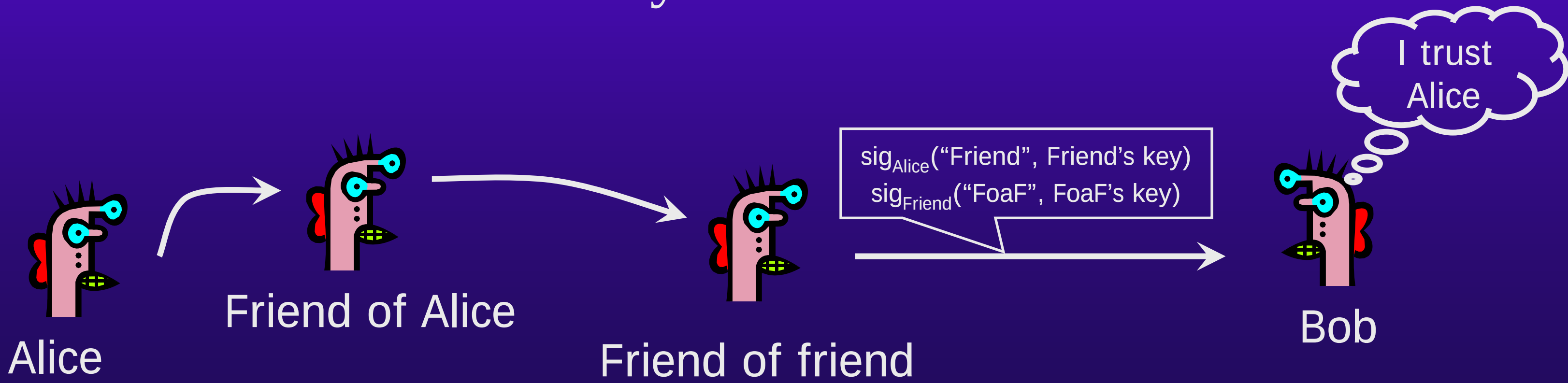
Hierarchical Approach

- Single CA certifying every public key is impractical
- Instead, use a trusted root authority
  - For example, Verisign
  - Everybody must know the public key for verifying root authority’s signatures
- Root authority signs certificates for lower-level authorities, lower-level authorities sign certificates for individual networks, and so on
  - Instead of a single certificate, use a certificate chain
    - $sig_{Verisign}$(“UT Austin”, PK_{UT}), sig_{UT}(“Vitaly S.”, PK_V)
  - What happens if root authority is ever compromised?

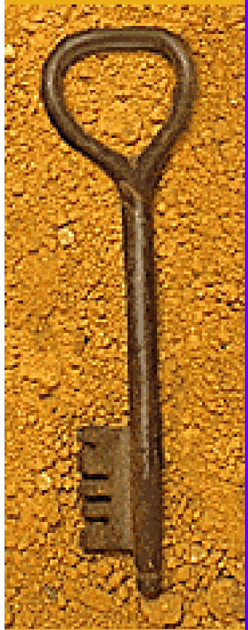


Alternative: “Web of Trust”

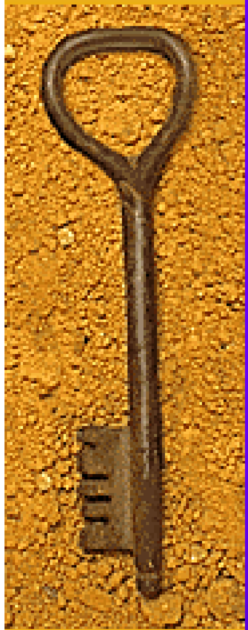
- Used in PGP (Pretty Good Privacy)
- Instead of a single root certificate authority, each person has a set of keys they “trust”
  - If public-key certificate is signed by one of the “trusted” keys, the public key contained in it will be deemed valid
- Trust can be transitive
  - Can use certified keys for further certification



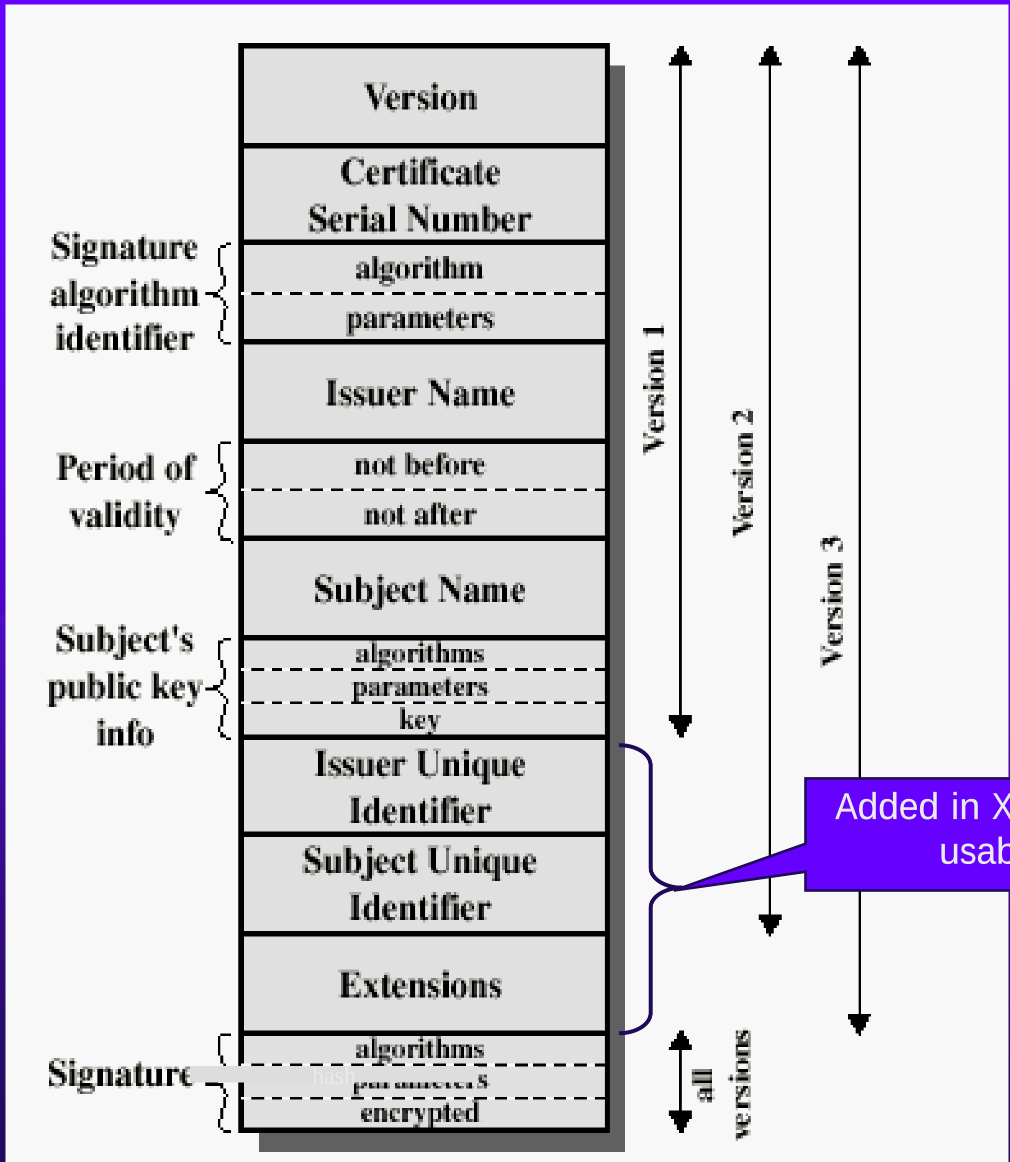
X.509 Authentication Service



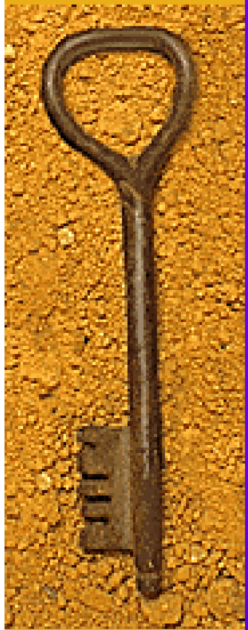
- Internet standard (1988-2000)
- Specifies certificate format
  - X.509 certificates are used in IPSec and SSL/TLS
- Specifies certificate directory service
  - For retrieving other users' CA-certified public keys
- Specifies a set of authentication protocols
  - For proving identity using public-key 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

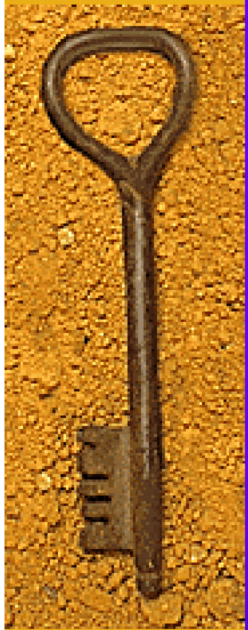


Added in X.509 versions 2 and 3 to address usability and security problems



Certificate Revocation

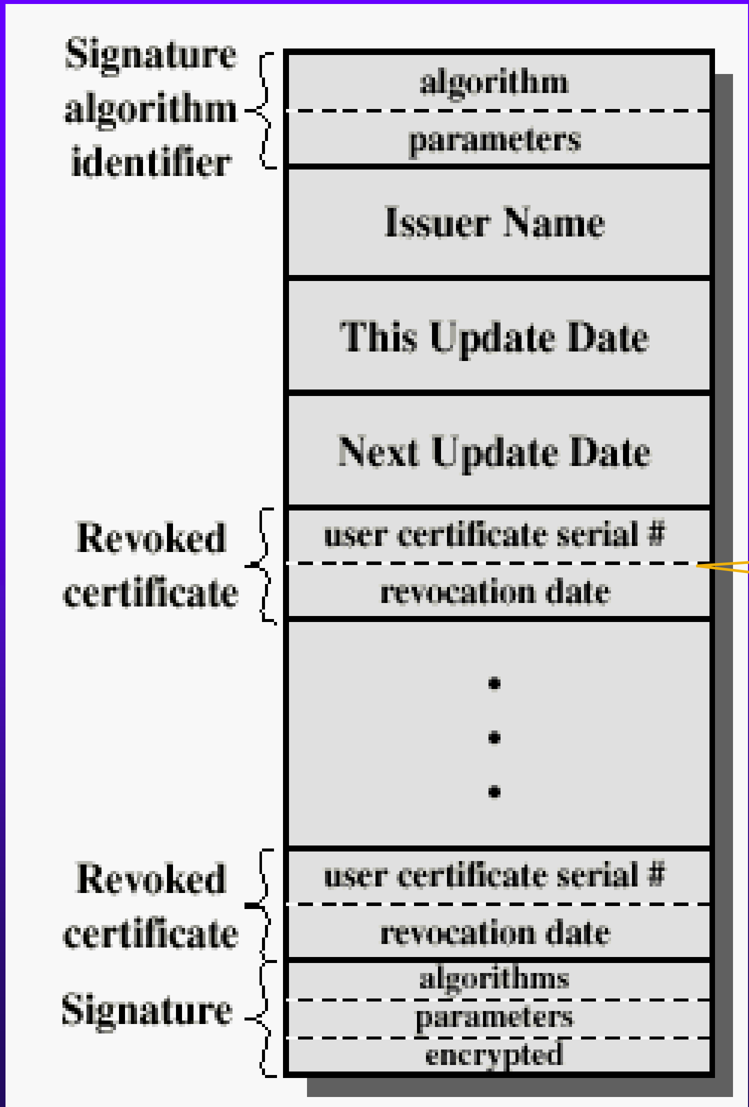
- Revocation is <u>very</u> important
- Many valid reasons to revoke a certificate
  - Private key corresponding to the certified public key has been compromised
  - User stopped paying his certification fee to this CA and CA no longer wishes to certify him
  - CA's certificate has been compromised!
- 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



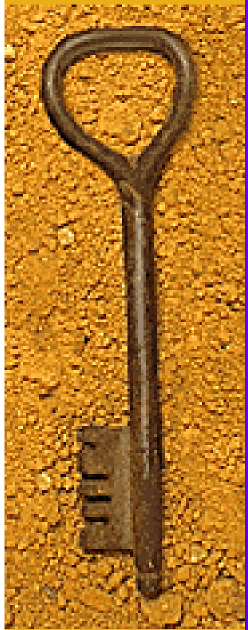
Certificate Revocation Mechanisms

- 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
    - Credit card companies used to issue thick books of canceled credit card numbers
  - Can issue a "delta CRL" containing only updates
- Question: does revocation protect against forged certificates?

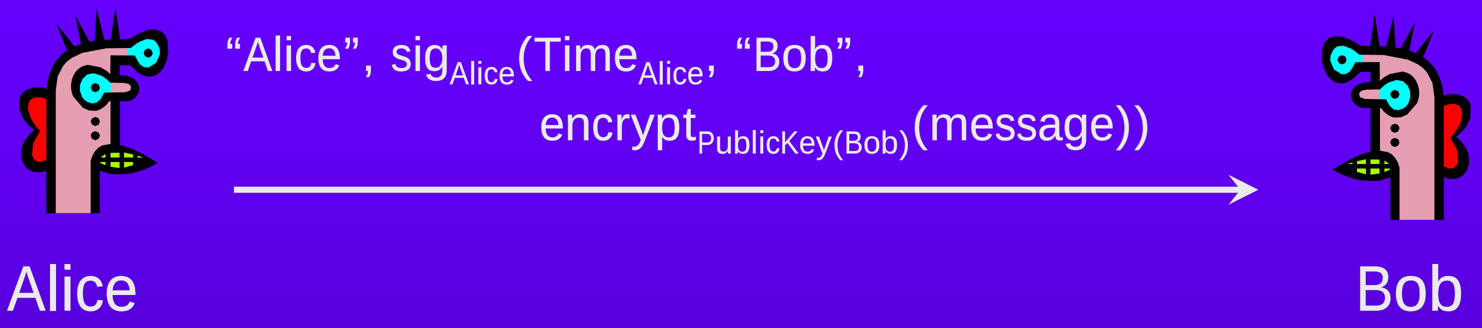
X.509 Certificate Revocation List



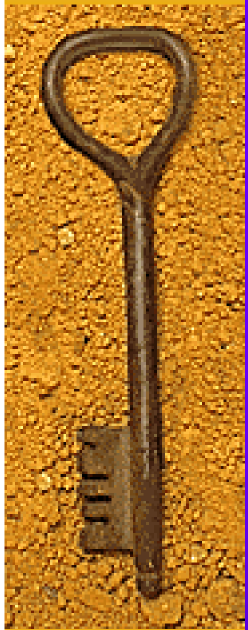
Because certificate serial numbers must be unique within each CA, this is enough to identify the certificate



X.509 Version 1



- Encrypt, then sign for authenticated encryption
  - Goal: achieve both confidentiality and authentication
  - E.g., encrypted, signed password for access control
- Does this work?



Attack on X.509 Version 1

- Receiving encrypted password under signature does <u>not</u> mean that the sender actually knows the password!
- Proper usage: sign, then encrypt

Authentication with Public Keys

1. Only Alice can create a valid signature
2. Signature is on a fresh, unpredictable challenge

Potential problem: Alice will sign <u>anything</u>

Mafia-in-the-Middle Attack [from Anderson's book]

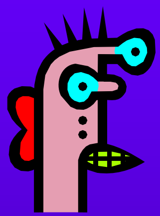
Early Version of SSL (Simplified)

fresh session key

$\text{encrypt}_{\text{PublicKey(Bob)}}(\text{“Alice”}, K_{AB})$ →

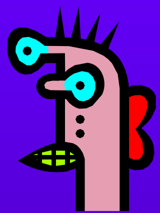
fresh random number

← $\text{encrypt}_{K_{AB}}(N_B)$



Alice

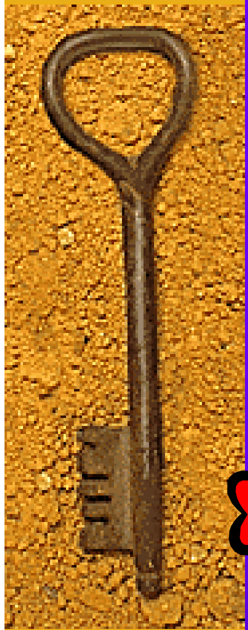
$\text{encrypt}_{K_{AB}}(\text{“Alice”}, \text{sig}_{\text{Alice}}(N_B))$ →

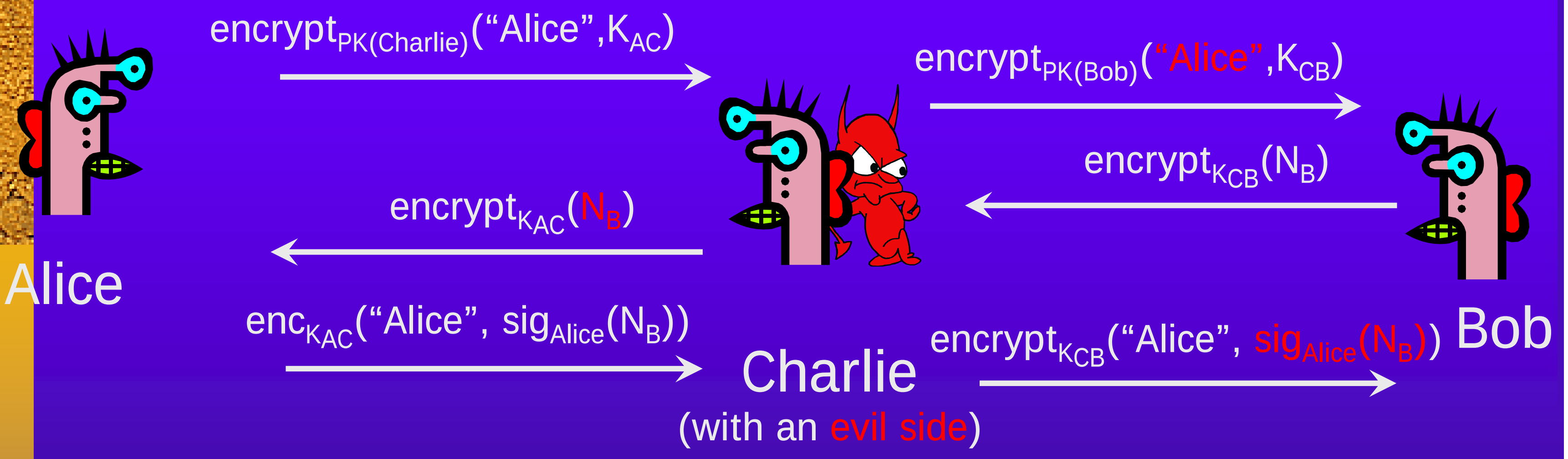


Bob

- Bob's reasoning: I must be talking to Alice because…
  - Whoever signed N_B knows Alice's private key… Only Alice knows her private key… Alice must have signed N_B… N_B is fresh and random and I sent it encrypted under K_{AB}… Alice could have learned N_B only if she knows K_{AB}… She must be the person who sent me K_{AB} in the first message…

Breaking Early SSL





- Charlie uses his legitimate conversation with Alice to impersonate Alice to Bob
  - Information signed by Alice is not sufficiently explicit