

CS 578 – Cryptography

Prof. Michael Backes

Certificates and Authentication Methods

June 23, 2006

Administrative Announcements

- Handouts:
 - New lecture notes

Recall: Signature Schemes

- Setup: Generates secret key sk , public (verification) key pk
- $\text{Sign}(sk, m)$ outputs sig
- $\text{Verify}(pk, m, sig)$: outputs “yes” or “no”
- Security: Existential unforgeability under chosen-message attack (CMA)

Recall: Signature Schemes

- Built signature scheme from one-way functions
- From generic one-way functions
 - One-time signatures, can be extended
 - Fast, but signatures are long
- $F^{\text{DLog}}(x) = g^x \bmod p$
 - Can built signatures out of this
 - ElGamal signatures, Schnorr signatures, DSA

Recall: Full-Domain Hash

- Works for every trapdoor permutation.

$S(\text{sk}, m)$:

- $D := \text{Hash}(m)$ for $\text{Hash}: \{0,1\}^* \rightarrow M_{\text{pk}}$
- $\text{sig} := F^{-1}(\text{sk}, D)$

(Hash: collision-resistance + maps into the full set M_{pk} (thus the name “full domain”))

- $V(\text{pk}, m, \text{sig})$:
 - Test if $F(\text{pk}, \text{sig}) = \text{Hash}(m)$

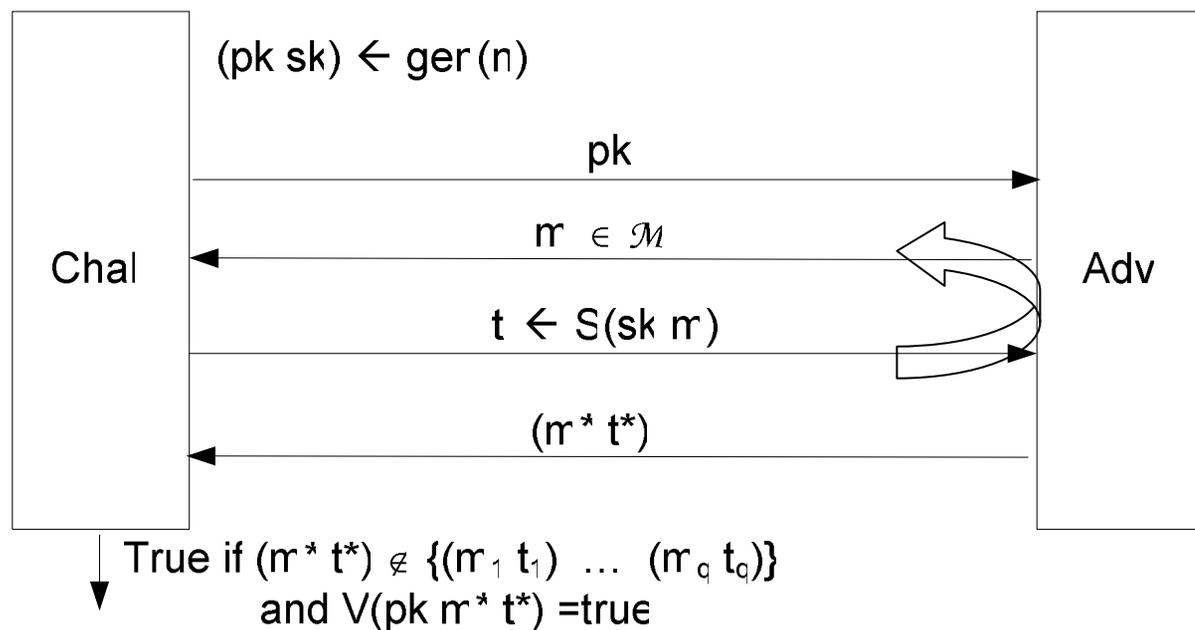
Full-Domain Hash

- Theorem (Bellare, Rogaway'94): If hash is a “random oracle” then FDH is existentially unforgeably under chosen-message attack assuming that F is one-way.

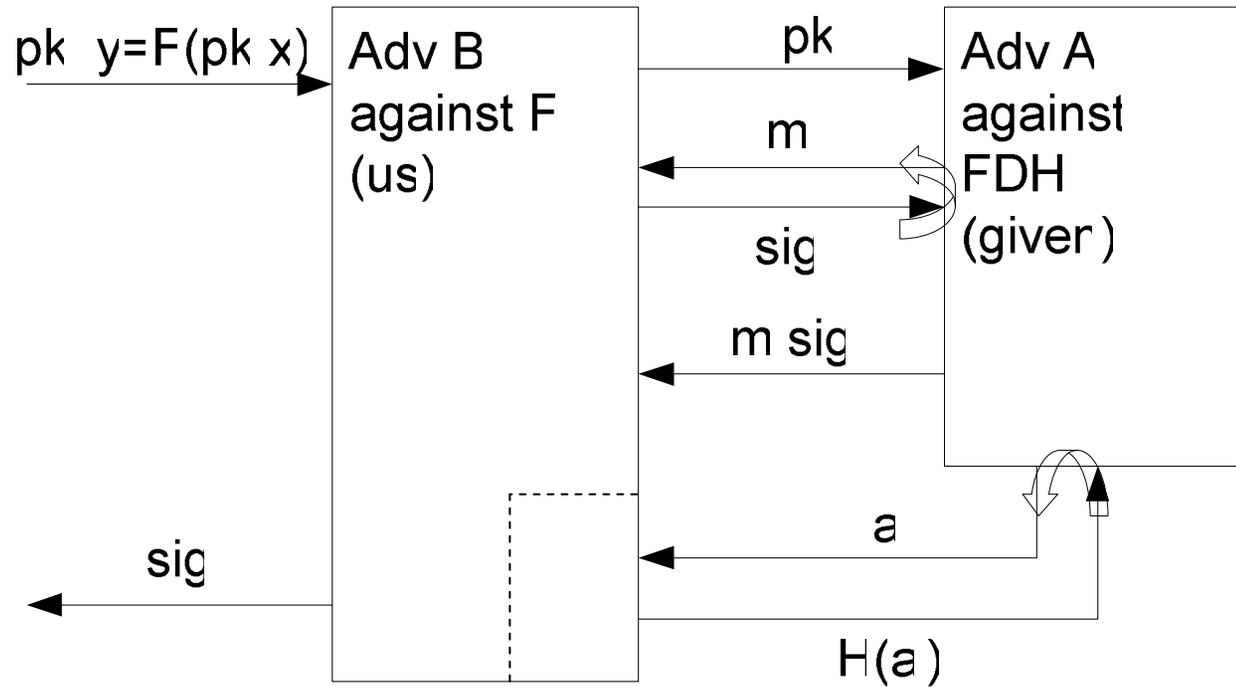
[proof on the board]

Recall: CMA-Security of Signatures

- For a signature scheme $\text{Sig} = (\text{Gen}, \text{S}, \text{V})$, define the following CMA game:



Full-Domain Hash – Proof Sketch



Summary of Digital Signatures

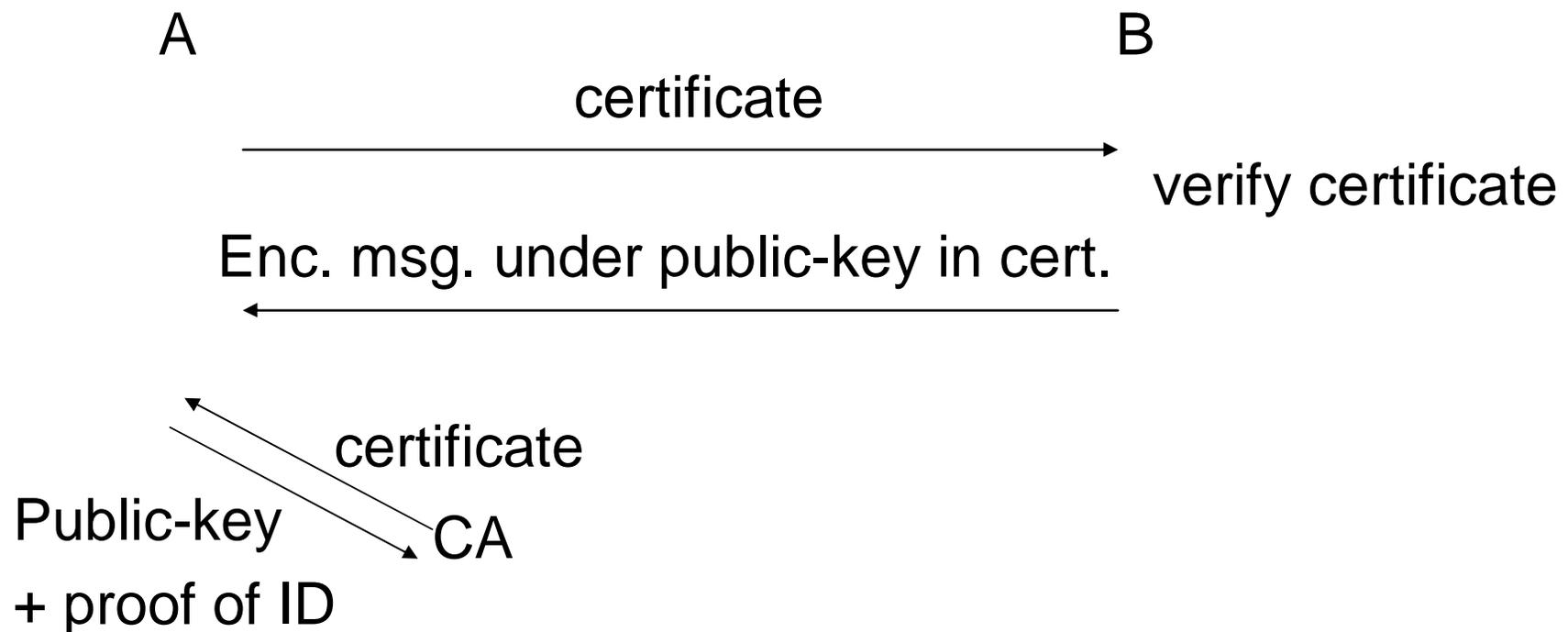
- **CMA-secure signature schemes exist if OWF exist**
- First **one-time signatures**: Lamport, Merkle, etc.
- **ElGamal signatures**: Security based on hardness of discrete logarithms, security not proven, relies on ability to solve equations in the exponents, common variants: Schnorr, DSA
- **RSA-based signatures**: Naïve use insecure, Full-domain Hash (FDH) based on any trapdoor permutation. CMA-secure in the random oracle model
- **Hash-then-Sign**: Often used in practice, however not mandatory (!), lots of books do as if this had to be done
- Today: Key management, certificates, trust

Public-key Management

- How does Alice authentically obtain Bob's public key?
- Main classes of possibilities:
 - Personally on disk
 - From public pk-dictionaries
 - Over the web with a certificate of a certification authority (CA)
 - Over a chain of people she knows

Public-key Management

- How does Alice obtain Bob's public key?
 1. Single Domain Certification Authority (CA)



Public-key Certification

- Public-key certification by X509.V3
- Cert = ([X500 Subject ID (home,org.,address)
X500 Signer ID/issuer,
the public key pk that is signed,
validity dates,
serial number,
other extensions, ...]
|| Sign(sk_{CA}, [....]

Public-key Certification

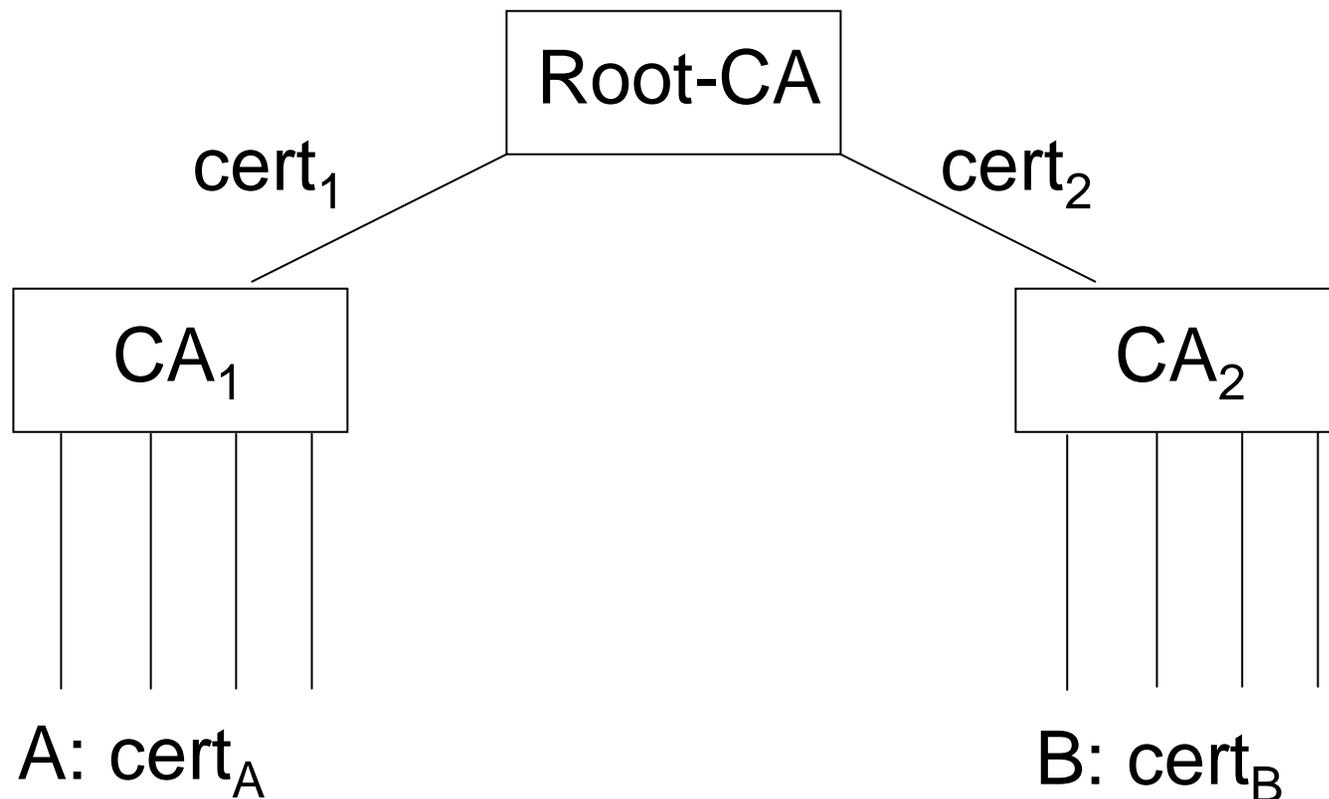
- [Certificates in Browser]

Public-key Certification

- Remarks:
- Alice obtains certificates offline, and CA is an offline entity
- Verify Alice's certificates requires Bob to have the CA's public-key
- CA is a trusted party
- CA's signing key has to be highly guarded

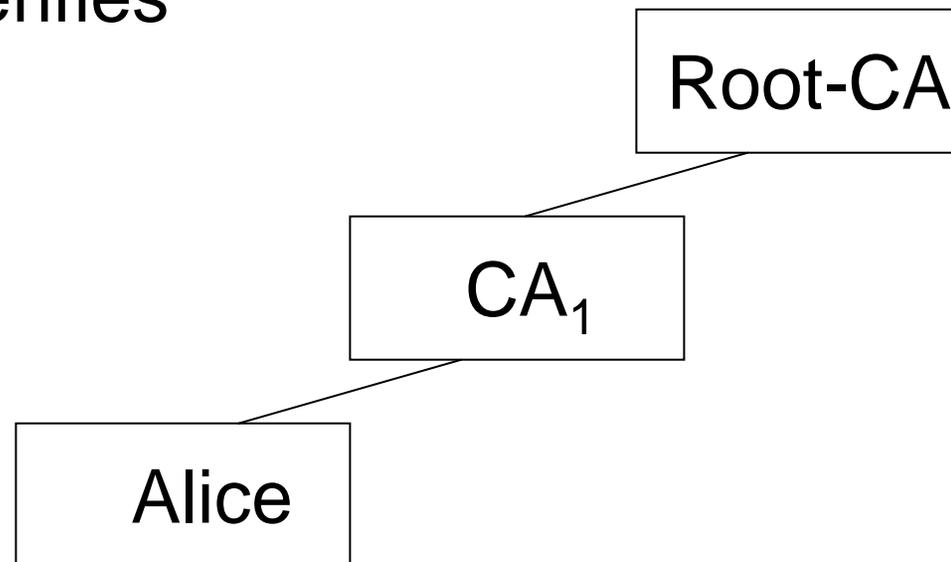
Cross-domain Certification

2. Cross Domain Certification

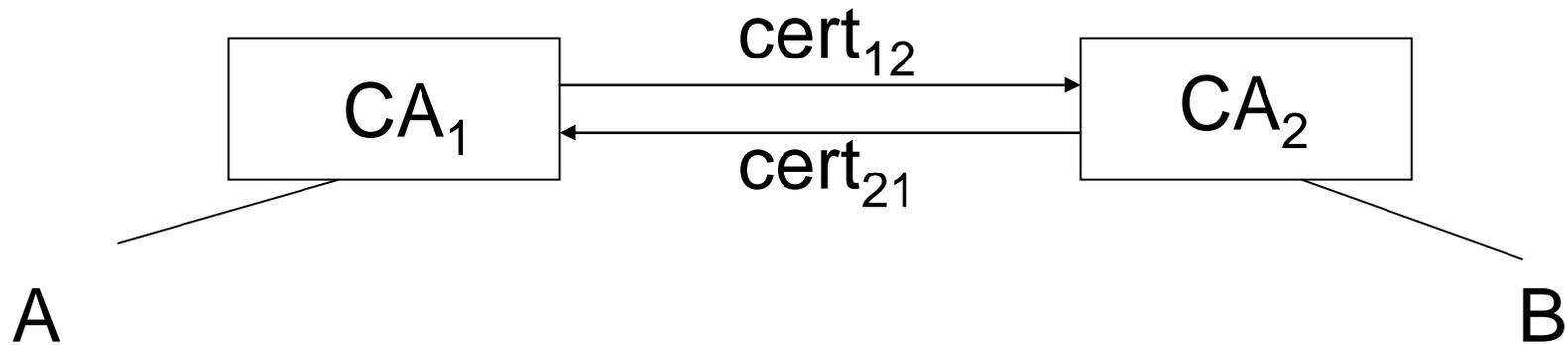


Cross-domain Certification (cont'd)

- Everyone has to have the root-CA public-key
- Alice's certificate: [Alice's info, CA₁'s sig] || [CA₁'s info, root-CA's sig]
- Bob verifies

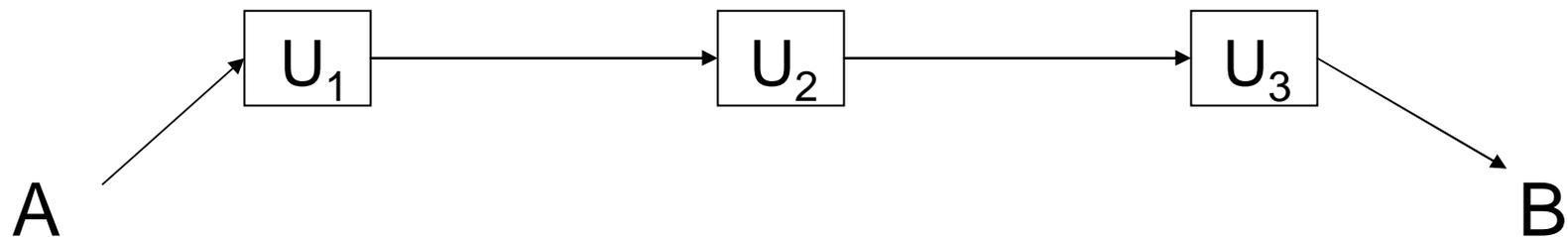


Cross-domain Certification (cont'd)



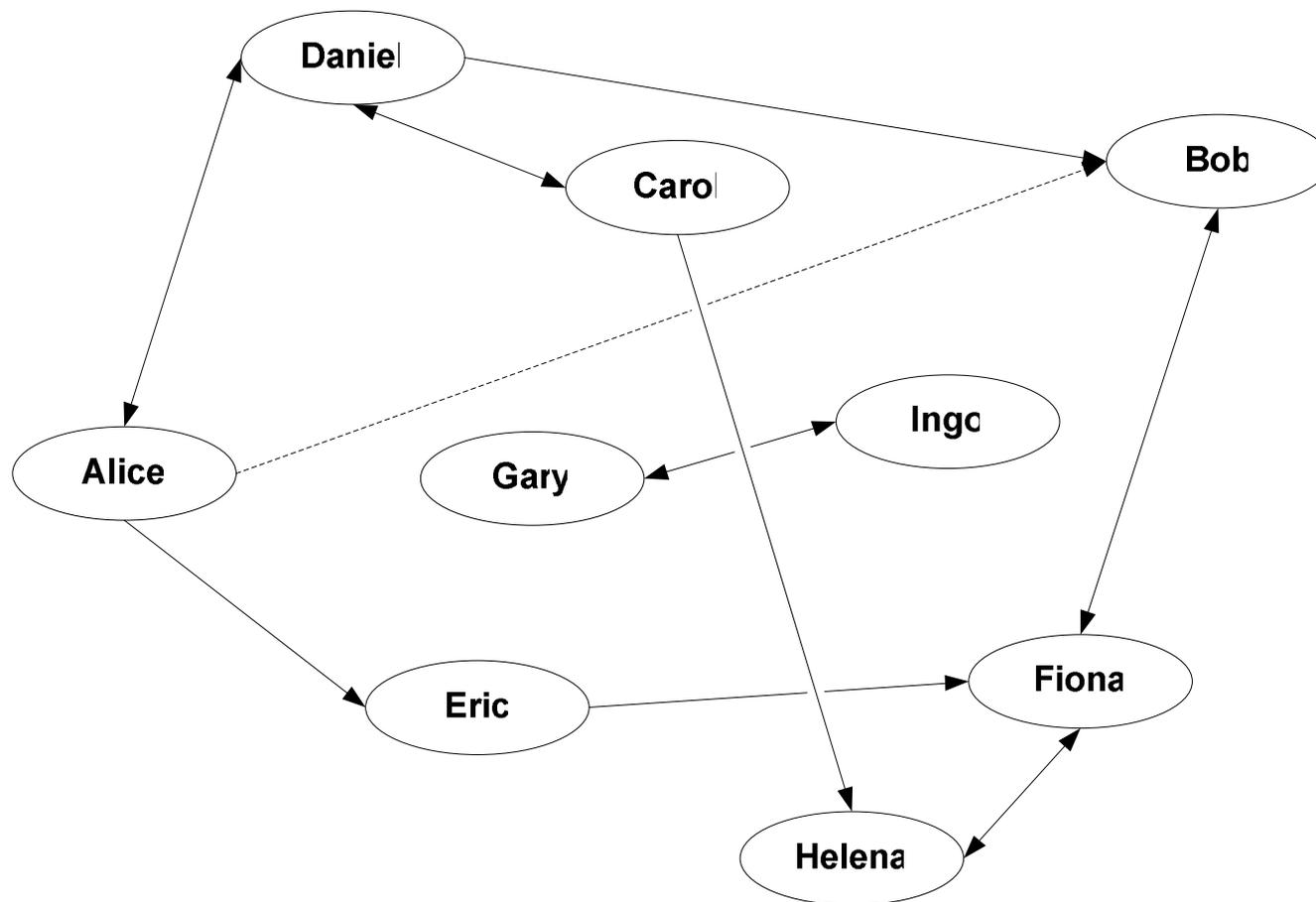
- A → B: cert₁₂ || cert₂₁ + signed/encrypted message

Web of Trust (PGP)

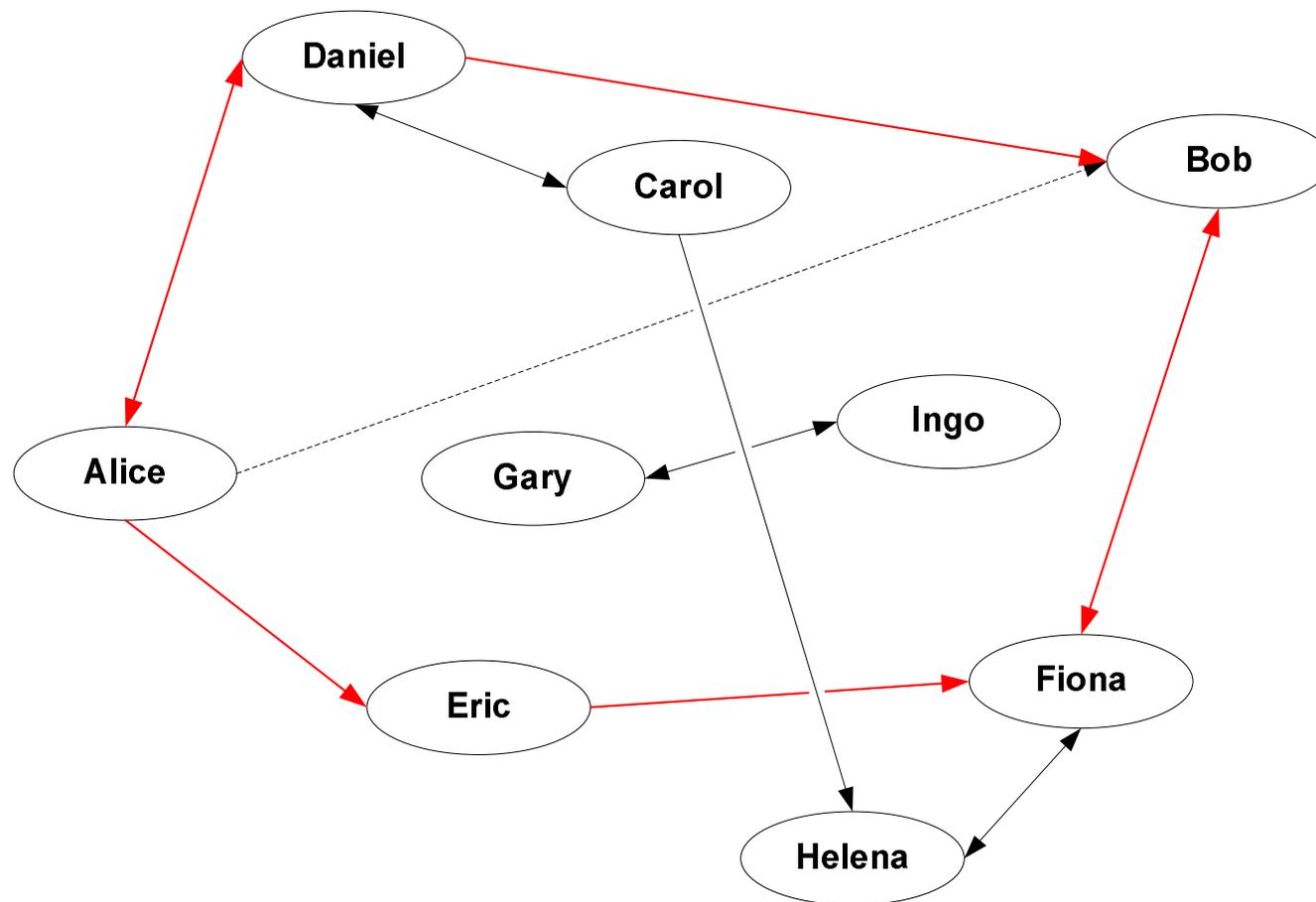


- $\text{Cert}_A = [\text{cert}(U_1 \rightarrow A), \text{cert}(U_2 \rightarrow U_1), \text{cert}(U_3 \rightarrow U_2)]$

Web of Trust Graph



Web of Trust Graph



Trust Evaluation due to Maurer

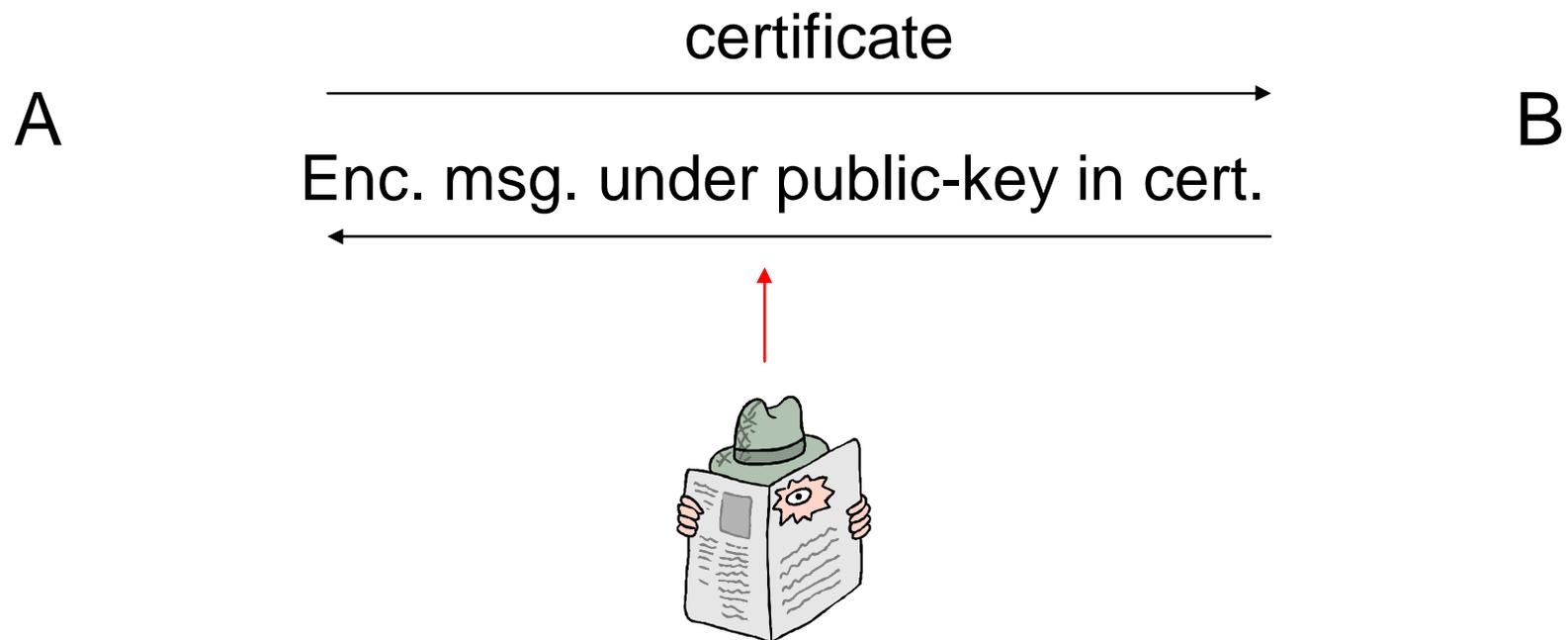
- Every user B maintains predicates:
 - $\text{Aut}(X)$: B considers public key of X authentic
 - $\text{Cert}(X,Y)$: B maintains certificate issued by X on pk of Y
 - $\text{Trust}(X,1)$: B trusts X that X correctly checks identities before issuing a certificate
 - $\text{Rec}(X,Y,1)$ “recommendation of depth 1”: B has (attribute) certificate of X, stating that X trusts Y to correctly issue certificates.
- Inductive extension:
 - $\text{Trust}(X,i)$: B trusts X that X correctly issues recommendations of order $i-1$
 - $\text{Rec}(X,Y,i)$: corresponding certificate

Trust Evaluation due to Maurer

- Evaluation rules: Meaning for Cert:
$$\text{Aut}(X) \wedge \text{Trust}(X,1) \wedge \text{Cert}(X,Y) \rightarrow \text{Aut}(Y)$$
- Evaluation rules: Meaning for Rec:
$$\text{Aut}(X) \wedge \text{Trust}(X,i+1) \wedge \text{Rec}(X,Y,i) \rightarrow \text{Trust}(Y,i)$$
- Allows for deducing trust. Starting point:
 - All $\text{Aut}(P)$ for which B has identified person P himself
 - All $\text{Trust}(P,i)$ that B considers true himself
 - All certificates B possesses, encoded as cert or rec
- Remark:
 - $\text{Trust}(X,1)$, $\text{Rec}(X,Y,1)$: Trust in humans to carefully certify
 - $\text{Trust}(X,2)$, $\text{Rec}(X,Y,2)$: Trust in knowledge of human nature
 - $\text{Trust}(X,\geq 3)$, $\text{Rec}(X,Y,\geq 3)$: Hard to understand

Active Attacks against Certification Sending

- Active attacks against sending of certificates?

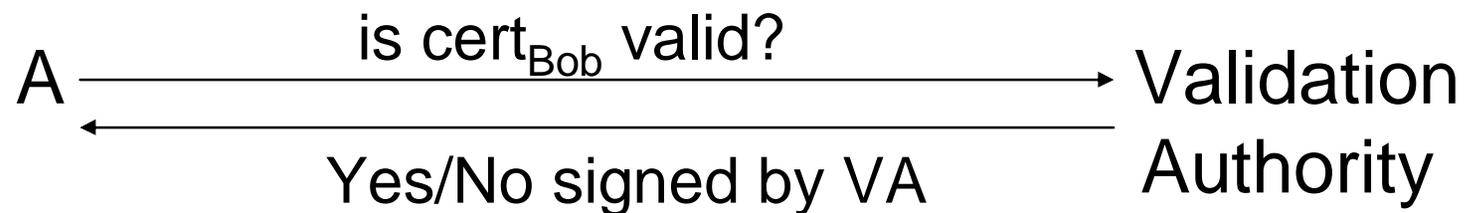


Certificate Revocation

- How to revoke A's certificate?
 1. Expiration date in credential (e.g., one year)
 2. CRL: Certificate Revocation List:
Post list of revoked certificates signed by the CA
 - Problem: CRLs become long
 - Δ -CRL: signed incrementally to CRL.

Certificate Revocation

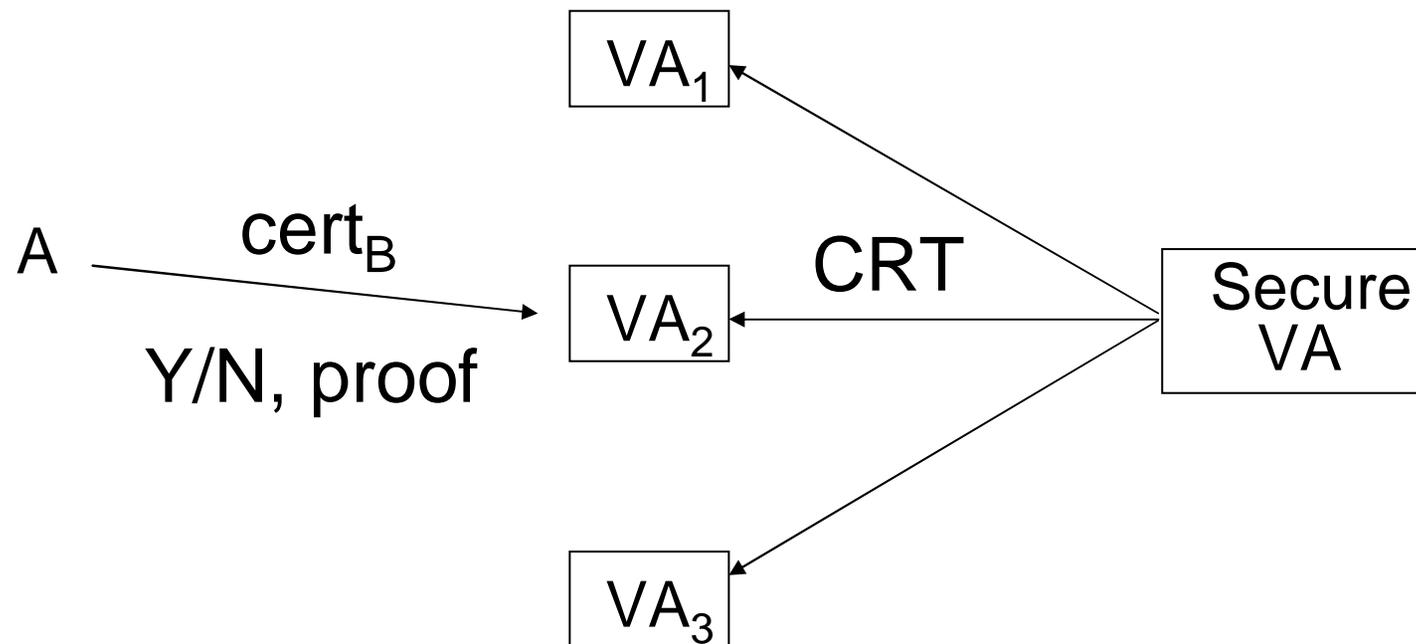
3. OCSP: Online Certificate Status Protocol



- Problems:
 - High load on VA → Replicate VA's key all over the world (but then key at risk)

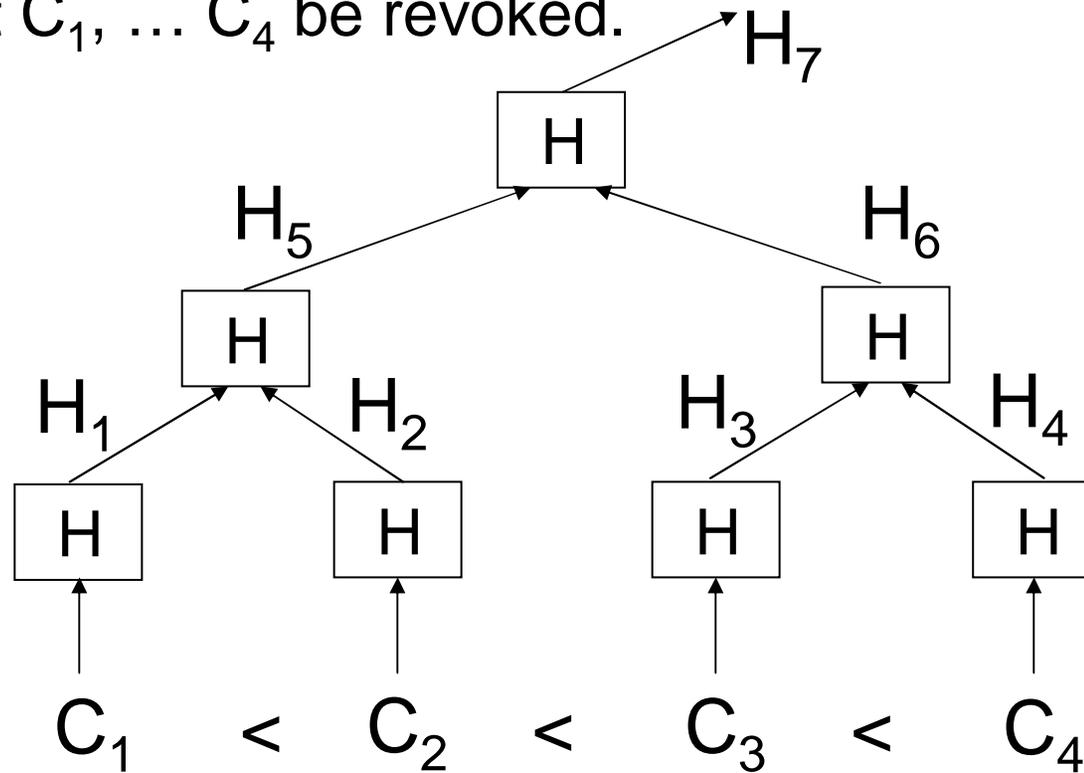
Certificate Revocation

4. Certificate Revocation Tree (CRT)



Certificate Revocation (cont'd)

- Let C_1, \dots, C_4 be revoked.



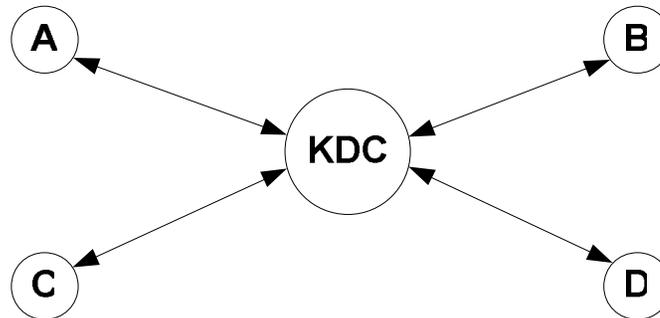
- $\text{CRT} = [C_1, C_2, C_3, C_4 + \text{sign}(\text{sk}_{\text{root-VA}}, (\text{tree-depth}, H_7))$

Certificate Revocation (cont'd)

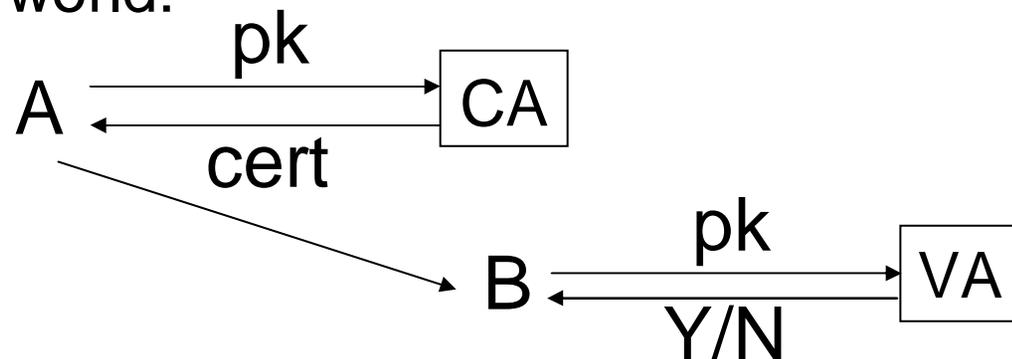
- Proof that, e.g., C_2 was revoked.
 - VA sends $[\text{sign}(\text{sk}_{\text{root-VA}}, (\text{tree-depth}, H_7)), H_1, H_6]$
 - Alice hashes C_2 , then compute tree and check depths and if root is H_7
 - For n revoked certificates, proof size is $O(\log n)$
- Proof that C was not revoked, i.e., $C \notin \text{CRT}$
 - Say $C_1 < C < C_2$
 - VA sends $[\text{sign}(\text{sk}_{\text{root-VA}}, (\text{tree-depth}, H_7)), H_1, H_2, H_6, C_1, C_2]$
- Answering a request incorrectly requires breaking the hash function (strictly: breaking 2^{nd} pre-image collision-resistance of H)

Distribution of Sym. and Pub. Keys

- Comparison: distribution of symmetric / public keys.
- Recall symmetric case: Key Distribution Center (KDC)



- Public-key world:



Distribution of Sym. and Pub. Keys

- KDC
 - is online: needed for every new session
 - is compromised → all past and future sessions are exposed
 - fast
- CA
 - is offline, but VA is online
 - is compromised → then future sessions exposed (forward secrecy)
 - slow