

ICS 180
May 4th, 2004

Guest Lecturer: Einar Mykletun

Symmetric Key Crypto

Symmetric Key

- ❖ Two users who wish to communicate share a secret key

- ❖ Properties
 - High encryption speed
 - Limited applications: encryption
 - Based on permutations and substitutions
 - No mathematical assumptions

- ❖ Candidates: DES, 3-DES, AES, Blowfish

- ❖ Problem:
 - Key Distribution

Public Key Crypto

Solves Key Distribution problem

...but...

Public-Key Cryptography

- ❖ Each user has a unique public-private key pair

Alice - K_{Apriv}, K_{Apub}

Bob - K_{Bpriv}, K_{Bpub}

- ❖ The public key can be given to anyone
- ❖ The private key is not shared with anyone, including a trusted third party (authentication server)
- ❖ The public key is a one-way function of the private-key (hard to compute private key from public one)
- ❖ Used for key distribution/agreement, message encryption, and digital signatures

Origins of Public Key

- ❖ Concept credited to Diffie and Hellman, 1976 "New Directions in Cryptography"
- ❖ Motivation - wanted a scheme whereby Alice could send a message to Bob without the need for Alice and Bob to share a secret or for a Trusted Third Party -- called "public-key" because Alice & Bob need only exchange public keys to set up a secret channel
- ❖ Invented earlier by British at CESG
<http://www.cesg.gov.uk/about/nsecret.htm>

Public-Key Agreement

- ❖ Method whereby Alice and Bob can agree on a secret key to use with DES, AES, or some other symmetric encryption algorithm
 - Need a shared secret
- ❖ They do this after exchanging only public keys
- ❖ They each compute a secret session key K derived from their own private key and the other's public key. They both arrive at the same K independently

Diffie-Hellman Method

1) Shared prime p and generator g

Alice: private x_a and public $y_a = g^{x_a} \bmod p$

Bob: private x_b and public $y_b = g^{x_b} \bmod p$

$$x_a = \log_g y_a \bmod p \text{ (hard to compute)}$$

2) They swap public keys

Alice computes: $K = y_b^{x_a} \bmod p = g^{x_b x_a} \bmod p$

Bob computes: $K = y_a^{x_b} \bmod p = g^{x_a x_b} \bmod p$

What can K be used for?

Math Strength

Depends on difficulty of computing the discrete logarithm

The best known methods are exponentially hard
- same as factoring

e.g., given n , find p, q where $n = p * q$

Need to use numbers on the order of 768 bits
(230 digits) or bigger

Implementations typically use 512 (155), 1024
(310) or 2048 (621) bits (digits).

Sending Messages

To send message M to Bob, only Bob's key is used

Alice \rightarrow Bob: $C = E_{B_{\text{pub}}}(M)$

Bob decrypts: $M = D_{B_{\text{priv}}}(C)$

In practice, use to distribute symmetric key K

Alice \rightarrow Bob: $C_K = E_{B_{\text{pub}}}(K), C_M = E_K(M)$

Bob decrypts: $K = D_{B_{\text{priv}}}(C_K), M = D_K(C_M)$

Alice and Bob then use K to encrypt/decrypt messages

SSL: www.amazon.com

*Public Key Cryptography solved the
Key Distribution Problem
but introduces another one...*

Certificate Revocation

Certificates...

- ❖ Who is Alice? Bob?
- ❖ How do we tie Alice to her public key?
- ❖ Certificates are issued by Certificate Authorities
 - States that owner of this certificate has the following public key
 - Certificate is signed
- ❖ How about the signer of the certificate?

CA

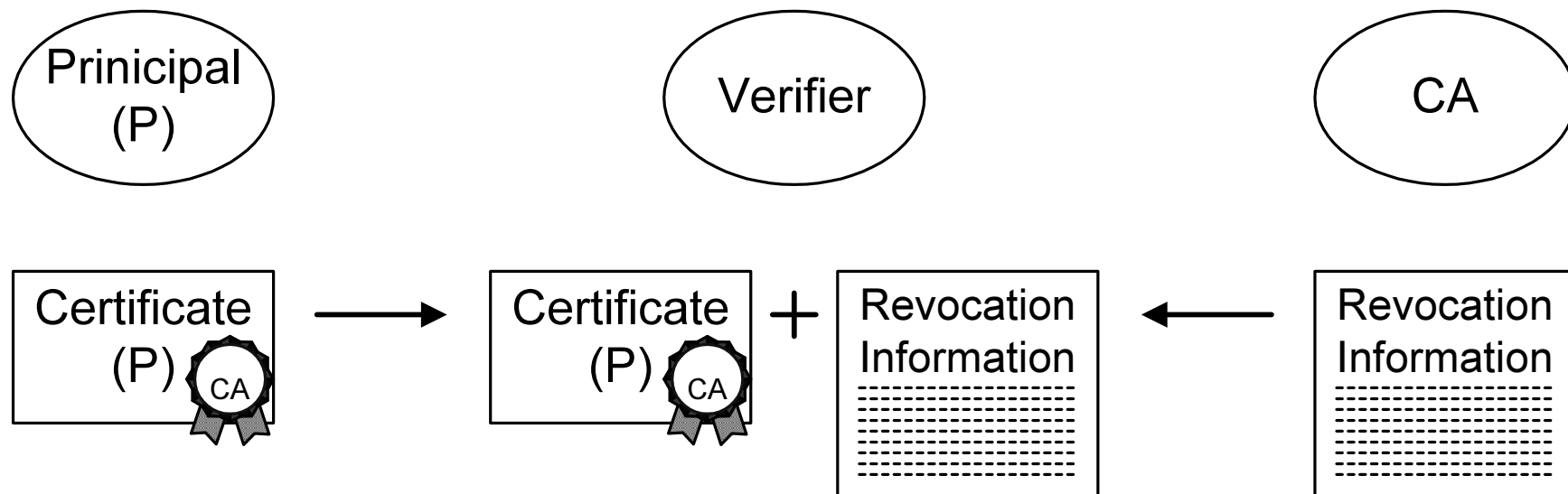
- ❖ CA checks that requesting user is who he claims to be (in the certificate request)
- ❖ CA's own certificate is signed by a higher-level CA. Root CA's certificate is self-signed and his identity/name is "well-known"
- ❖ CA is a critical part of the system and must operate in a secure and predictable way according to some policy

Who needs them?

- ❖ Certificates checked by verifiers to find out correct public key of the target entity
- ❖ A verifier must:
 - know the public key of the CA (CAs?)
 - trust all CAs involved
- ❖ Certificate checking is verification of the signature and validity of certificate

Verifying a PKC

- ❖ How can one verify a PKC ?
- ❖ Need a PKI
 - Otherwise, if PKC is self-signed (as in PGP) and must be trusted directly



PKI and Revocation

- ❖ CA's PKC must itself be validated
- ❖ Need CA PKC's issuer's PKC, etc.
- ❖ Every PKC has own validity period
- ❖ What if a certificate must be invalidated before the expiration date ? (e.g., private key compromise)

Requirements for revocation

❖ Timeliness

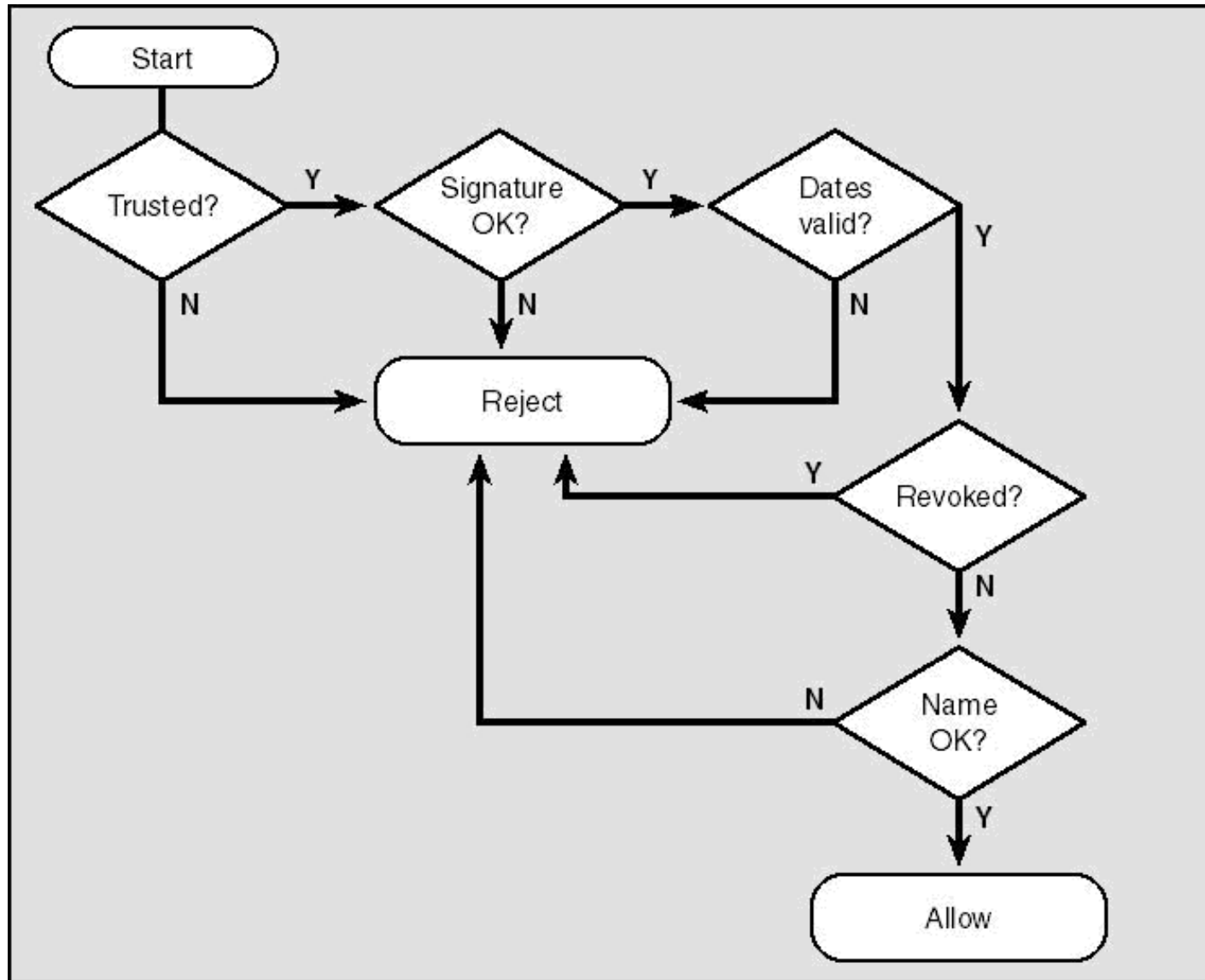
- ◆ Must check most recent revocation status

❖ Efficiency

- Computation
- Bandwidth and storage
- Availability

❖ Security

Verifying a certificate (assuming common CA)



Revocation methods

- ❖ CRL (Certificate Revocation List)
 - CRL-DP, indirect CRL, dynamic CRL-DP,
 - delta-CRL, windowed CRL, etc.
- ❖ OCSP → explicit
- ❖ CRS → implicit
- ❖ CRT
- ❖ Authenticated Data Structures
 - E.g., Certificate Update Scheme, skiplists

CRLs

- ❖ Off-line mechanism
- ❖ CRL = list of revoked certificates (e.g., SNs) signed by a revocation authority (RA)
- ❖ RA not always CA that issued the revoked PKC
- ❖ Periodically issued: daily, weekly, monthly, etc.

Pros & Cons of CRLs

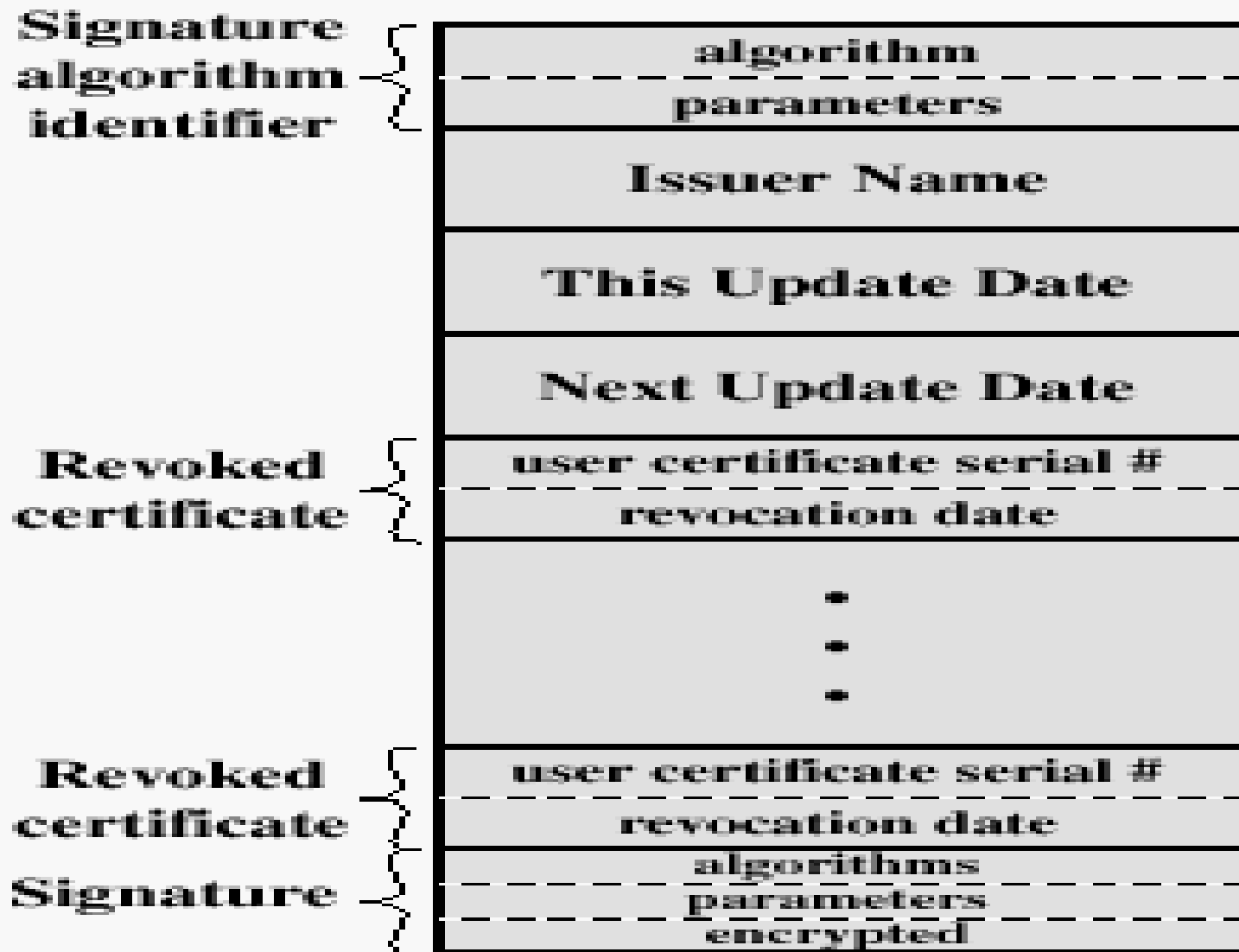
❖ Pros

- Simple
- Don't need secure channels for CRL distribution

❖ Cons

- Timeliness: "window of vulnerability"
- CRLs can be huge
- How to distribute CRLs reliably?

X.509 CRL Format



Variation: Indirect CRL

- ❖ CRL issuer is a distinct entity: revocation authority (TTP)
 - Can aggregate multiple CRLs from different CAs
 - verifier can avoid requesting different CRLs from different sources
 - problem: trust in the TTP

PKI and Revocation

- ❖ On January 29 and 30, 2001, VeriSign, Inc. issued two certificates for Authenticode Signing to an individual fraudulently claiming to be an employee of Microsoft Corporation.
- ❖ Any code signed by these certificates appears to be legitimately signed by Microsoft.
- ❖ Users who try to run code signed with these certificates will generally be presented with a warning dialog, but who wouldn't trust a valid certificate issued by VeriSign, and claimed to be for Microsoft?
- ❖ Certificates were very soon placed onto a CRL, but:
 - the code that checks the signatures for ActiveX controls, Office Macros, and so on, doesn't do any CRL processing.
- ❖ According to Microsoft:
 - since the certificates don't have a CRL Distribution Point (DP), it's impossible to find and use the CRL!

Delta CRL

- ❖ simple variation
- ❖ starting point: a base CRL (infrequent)
- ❖ distribution of delta CRL which contains the changes occurred
- ❖ delta CRLs are smaller
 - can be issued more frequently

Certificate Revocation Tree (CRT)

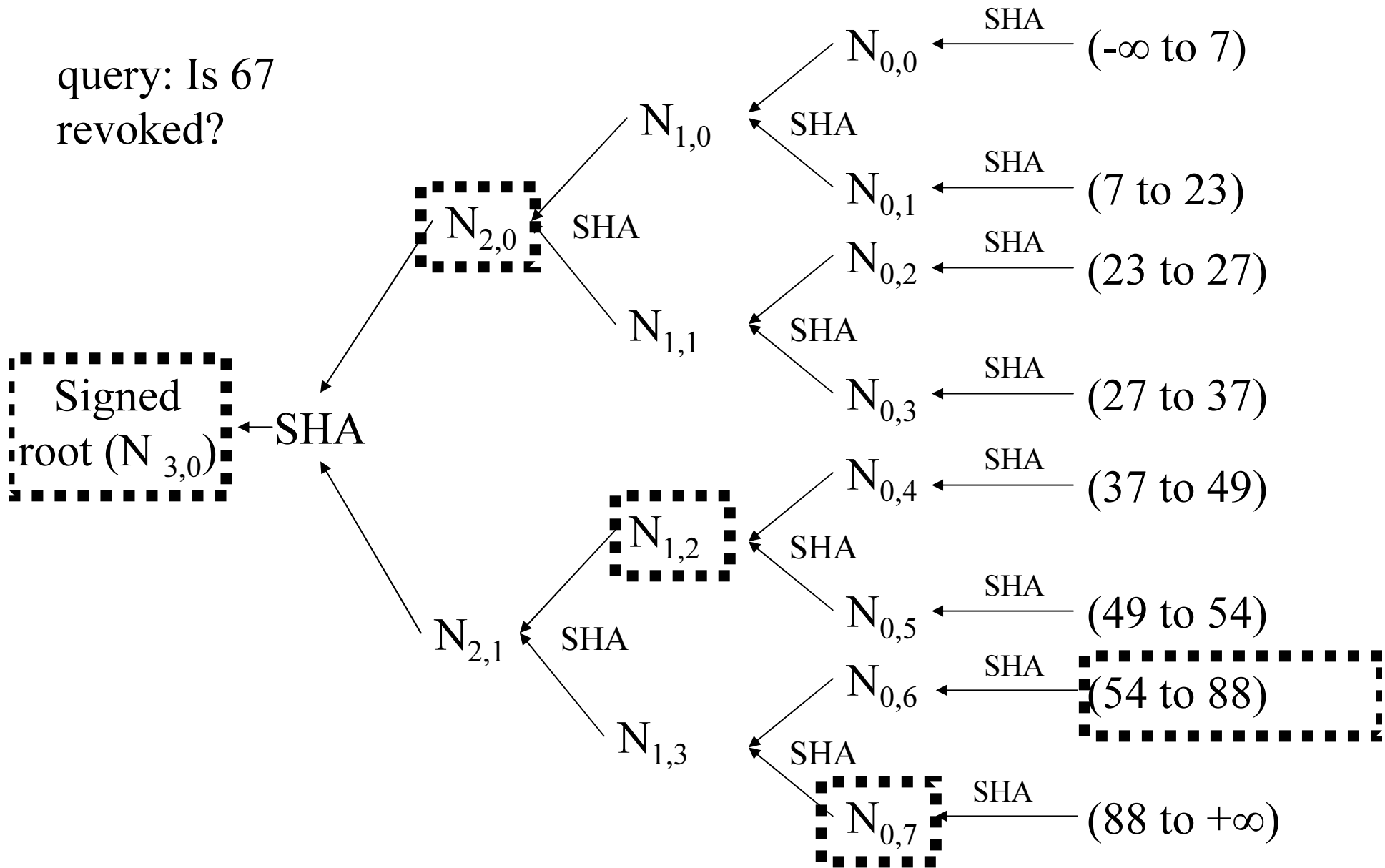
- ❖ proposed by Kocher (1998)
- ❖ based on hash trees
 - hash trees first proposed by Merkle in another context in 1979 (one-time signatures)
 - improvement to Lamport-Diffie OTS scheme
 - based on the following idea:
 - ◆ A wants to sign a bit of information. A gives B the image (y) produced as $y=F(x)$
 - ◆ Eventually A reveals the pre-image: x
 - ◆ B checks that: $y=F(x)$

CRT contd.

- ❖ express ranges of SN of PKC's as tree leaf labels:
 - E.g., (5 -- 12) means: 5 and 12 are revoked, the others larger than 5 and smaller than 12 are okay
 - Place the hash of the range in the leaf
- ❖ response includes the corresponding tree leaf, the necessary hash values along the path to the root, the signed root
- ❖ the CA periodically updates the structure and distributes to un-trusted servers called Confirmation Issuers

Example of CRT

query: Is 67
revoked?



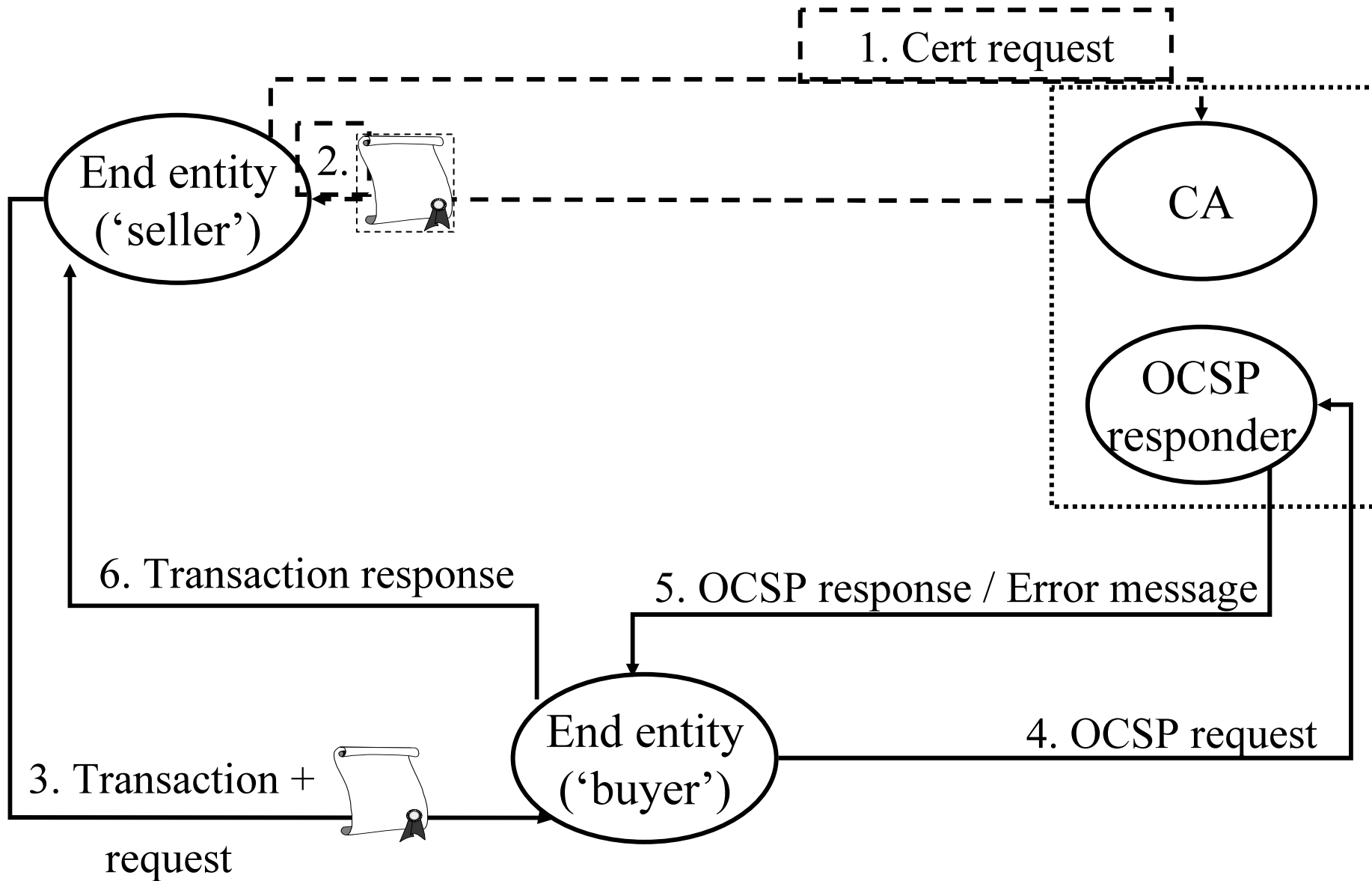
Characteristics of CRT

- ❖ response represents a proof
- ❖ length of proof is $O(\log n)$
- ❖ only one "real" signature for tree root
(could be done off-line)

Explicit Revocation: OCSP

- OCSP = On-line Certificate Status Protocol (RFC 2560) - June 1999
- In place of or, as a supplement to, checking CRLs
- Obtain instantaneous status of a PKC
- OCSP may be used for sensitive, volatile settings
 - ◆ (e.g., stock trades, electronic funds transfer)

OCSP players



OCSP definitive response

- all definitive responses have to be signed:
 - ◆ either by issuing CA
 - ◆ or by a Trusted Responder (OCSP client trusts the TR's PKC)
 - ◆ or by a CA Authorized Responder which has a special PKC (issued by the CA) saying that it can issue OCSP responses on CA's behalf

Responses for each certificate

❖ Response format:

- target PKC SN
- PKC status:
 - ◆ good - positive answer
 - ◆ revoked - permanently/temporarily (on-hold)
 - ◆ unknown - responder doesn't know about the certificate being requested
- response validity interval
- optional extensions

Security Considerations

- ❖ on-line method
- ❖ DoS vulnerability
 - flood of queries + generating signatures!
 - unsigned responses → false responses
 - pre-computing responses offers some protection against DoS
- ❖ pre-computing responses allows replay attacks (as no nonce included)
 - but OCSP signing key can be kept off-line

Open questions

- ❖ Consistency between CRL and OCSP responses
 - possible to have a certificate with two different statuses. How is handled such a thing?
- ❖ If OCSP is more timely and provides the same information as CRLs do we still need CRLs?
- ❖ Which method should come first - priority to OCSP or to CRL?

Implicit Revocation: Certificate Revocation System (CRS)

- ❖ proposed by Micali (1996)
- ❖ aimed to improve CRL communication costs
- ❖ basic idea: signing a message for every certificate stating its status
- ❖ use of off-line/on-line signature scheme to reduce update cost

CRS: creation of a certificate

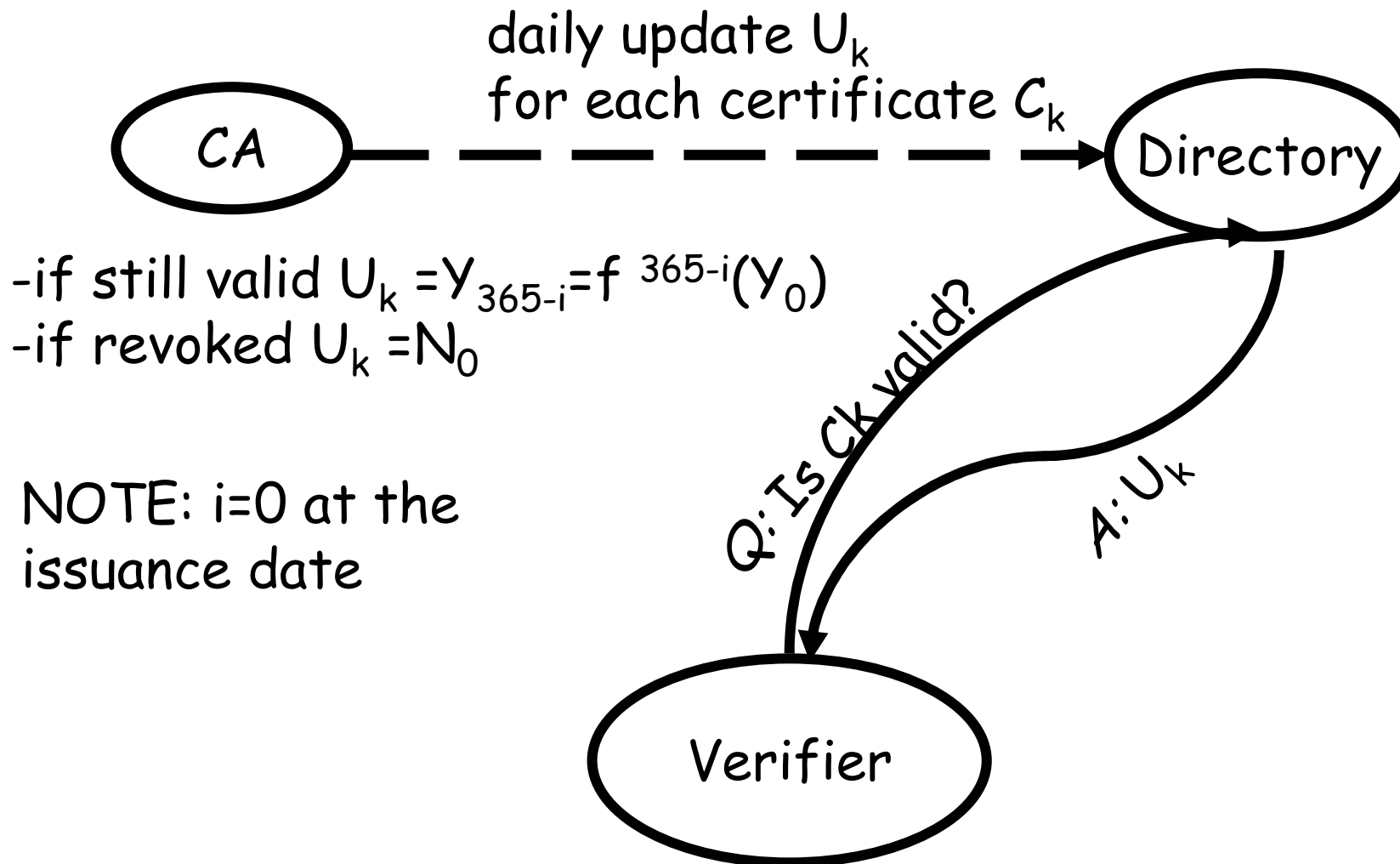
- ❖ Two new parameters in PKC: Y_{MAX} and N

$$Y_{MAX} = f^{MAX}(Y_0)$$

$$N = f(N_0)$$

- ❖ Y_0, N_0 - per PKC secrets stored by CA
- ❖ f is a public one-way function

How CRS works



What is cryptography?

❖ Some terminology

- Cryptography
- Cryptanalysis
- Cryptology

❖ What is cryptography used for?