Authentication in Distributed Systems

Introduction

Crypto transforms (communications) security problems into key management problems.

To use encryption, digital signatures, or MACs, the parties involved have to hold the “right” cryptographic keys.

With public key algorithms, parties need authentic public keys.

With symmetric key algorithms, parties need shared secret keys.

Session Keys

Public key algorithms tend to be more expensive than symmetric key algorithm.

Cost factors: key length, computation time, bandwidth.

It is desirable to use long-term keys only sparingly to reduce the “attack surface”.

Potential problem: attacks that collect a large amount of encrypted material.

Solution: long-term keys establish short term session keys.

Key Usage

It is good cryptographic practice to restrict the use of keys to a specific purpose.

In key management, we may use key encrypting keys and data encrypting keys.

Examples for key usages:

Encryption	Decryption
Signature	Non-repudiation
Master key	Transaction key …

With RSA, don't use a single key pair both for encryption and for digital signatures.

Agenda

Remote user authentication

Definitions for key establishment

Diffie-Hellman key agreement

Man-in-the-middle attacks

STS – station-to-station protocol

AKEP

Needham-Schroeder

Perfect forward secrecy

Kerberos

Using Passwords Remotely

Sending passwords over the network.

Challenge-response protocols.

Off-line dictionary attacks.

RADIUS (RFC 2865).

HTTP Basic Authentication

Client: **`GET /index.html HTTP/1.0`**

Server:
```
HTTP/1.1 401 Unauthorized
WWW-authenticate Basic
realm="SecureArea"
```

Client:
```
GET /index.html HTTP/1.0}
Authorization: Basic
am9ldXNlcjphLmIuQy5E
```

Server: **`HTTP/1.1 200 Ok`** *(plus document)*

Password sent in the clear, base64 encoded.

Not really secure: anybody who can see the user's reply learns the password.

HTTP Digest Authentication

Challenge-response protocol (RFC 2617).
Server sends random challenge (nonce) to user.
User replies with hash (digest) of

username+password+nonce+uri:

request-digest =
$h(h(\text{username:realm:password}):$
nonce: $h(\text{method:digest-uri}))$

Better security but still vulnerable to off-line dictionary attacks.

Nonces

The term “nonce” was proposed Needham & Schroeder for unique values that are used only once.

Three ways of generating nonces:

- Random numbers
- Time stamps
- Sequence numbers

Nonces are used to prevent replay attacks

Nonces are used to guarantee “freshness”.

In some applications, nonces have to be unpredictable.

Terminology

Once, protocols establishing a session key were called authentication protocols.

After all, it is their purpose to let you know "whom you are talking to".

In the literature, in particular in older sources, you may still find this convention.

Today's convention in cryptology distinguishes between authentication and key establishment.

Types of Assurances

Reciprocity: unilateral or mutual authentication.

Key freshness: is there a protection against replay attacks?

Key control: who generates the key? Sometimes attacks are possible if one party can pick a key with specific properties.

Third party requirements: is a Trusted Third Party (TTP) involved, off-line or on-line?

Key Establishment (HAC)

Key establishment is a process whereby a shared secret becomes available to two or more parties, for later cryptographic use.

Key transport: one party creates the secret value and securely transfers it to the other(s).

Key agreement: both parties contribute to the generation of the secret value much that no party can predict the outcome.

Key Authentication

Key authentication: one party is assured that no other party aside from a specifically identified second party may gain access to a particular secret key.

Key confirmation: one party is assured that a second (possibly unidentified) party has possession of a particular secret key.

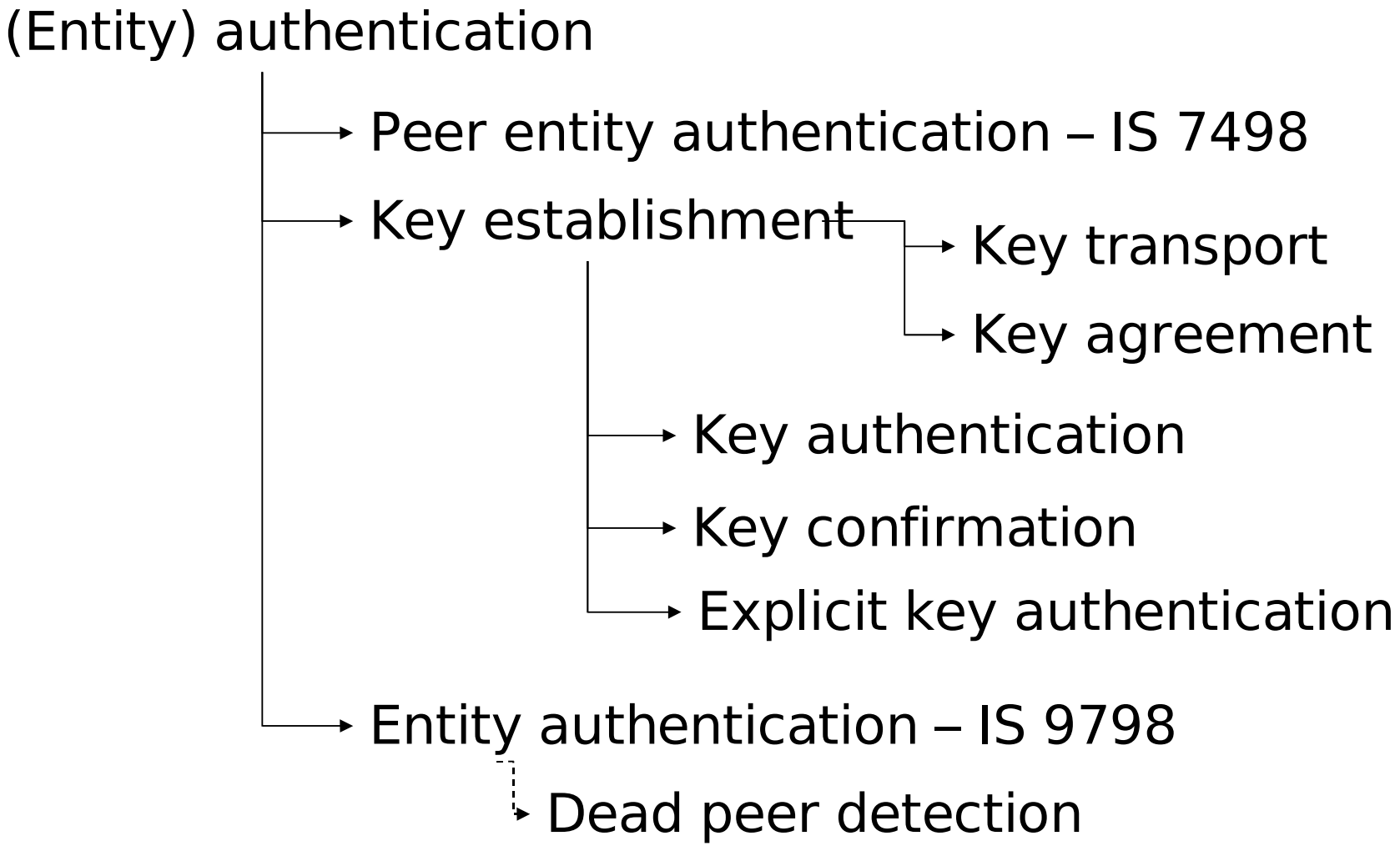
Explicit key authentication: both key authentication and key confirmation hold.

Key Establishment & TTPs

In a protocol like STS where key authentication is based on digital signatures, a Trusted Third Party (TTP) may have to vouch for the authenticity of verification keys.

In a protocol where authentication is based on symmetric cryptographic algorithms, a TTP may serve as a key distribution centre (KDC) supplying parties with session keys.

Authentication – Overview



Key Establishment Protocols

AKEP

Needham-Schroeder protocol

Kerberos

AKEP2

AKEP2: Authenticated Key Exchange Protocol 2

Uses symmetric authentication mechanisms but does not rely on a TTP.

Parties A and B share long-term symmetric keys K and K'.

They use a keyed hash function (MAC) h_K and a keyed one-way function $h'_{K'}$.

It is frequently a design criterion to avoid the use of encryption algorithms.

AKEP2

Let n_A and n_B be random nonces picked by A and B respectively.

AKEP2 is a three-pass protocol:

1. A → B: n_A
2. B → A: $B, A, n_A, n_B, h_K(B,A,n_A,n_B)$
3. A → B: $A, n_B, h_K(A,n_B)$

The shared key is $k = h'_{K'}(n_B)$

AKEP2 provides mutual entity authentication and (implicit) key authentication.

Reminder: DLP

Let p be a prime and a generator g of high order modulo p.

Exponentiation a $g^a \bmod p$ is a one-way function.

Discrete Logarithm Problem (DLP): given p, g, and y, find the discrete logarithm a so that $y = g^a \bmod p$.

Exponentiation mod p is commutative:

$(g^a)^b \bmod p = g^{ab} \bmod p = (g^b)^a \bmod p$

Diffie-Hellman Key Agreement

Parties A and B do not share an initial secret but agree on a prime p and a generator g.

A picks a random value a, $2 \quad a \quad p\text{-}2$, and sends $g^a \bmod p$ to B.

B picks a random value b, $2 \quad b \quad p\text{-}2$, computes the shared key $(g^a)^b = g^{ab} \bmod p$ and sends $g^b \bmod p$ to A.

Upon receiving $g^b \bmod p$, A computes the shared key $(g^b)^a = g^{ab} \bmod p$.

Diffie-Hellman Key Agreement

The "security" of this protocol depends on the difficulty of the DLP: an attacker able to compute discrete logarithms could obtain a and b from $g^a \bmod p$ and $g^b \bmod p$.

The "security" of the Diffie-Hellman protocol is not known to be equivalent to the DLP.

Computational Diffie-Hellman Problem (DHP):

Given p, g, $g^a \bmod p$ and $g^b \bmod p$, compute $g^{ab} \bmod p$.

Diffie-Hellman – Security

Which security properties do we get from Diffie-Hellman key agreement?

It is a key agreement protocol.

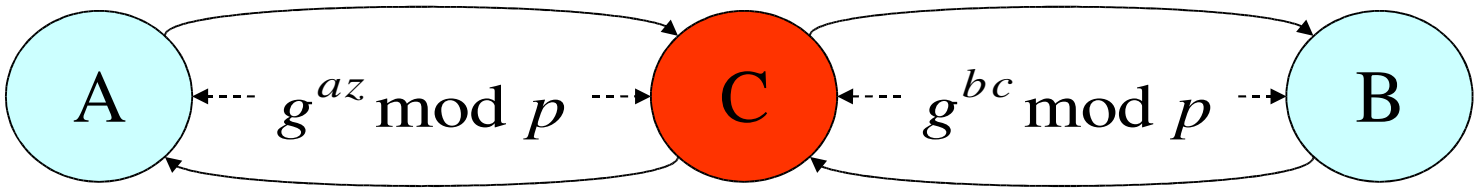
Secrecy: An attacker observing the messages exchanged does not learn the key.

No authentication: The parties do not know whom they are establishing a key with.

Man-in-the-middle Attacks

An attacker *C* sitting between *A* and *B* can mount a man-in-the-middle attack:

$g^a \bmod p$ $\qquad$ $g^c \bmod p$



$g^z \bmod p$ $\qquad$ $g^b \bmod p$

Station-to-station Protocol

Authenticated variant of Diffie-Hellman key agreement.

A and B use a digital signature algorithm: $S_A(m)$ denotes A's signature on m.

A and B use a symmetric encryption algorithm: $E_k(m)$ denotes encryption of m under key k.

A and B agree on a prime p and a generator g of order $p-1$ modulo p.

Station-to-station Protocol

A picks random value a, $2 \quad a \quad p\text{-}2$, and sends $g^a \bmod p$ to B .

B picks random value b, $2 \quad b \quad p\text{-}2$, computes the shared key $k = (g^a)^b = g^{ab} \bmod p$, and sends $g^b \bmod p$ and $E_k(S_B(g^b, g^a))$ to A.

A computes the key $k = (g^b)^a \bmod p$, decrypts $E_k(S_B(g^b, g^a))$ and verifies signature $S_B(g^b, g^a)$.

A replies with $E_k(S_A(g^a, g^b))$; B decrypts and verifies the signature.

Station-to-station Protocol

A → B: $g^a \bmod p$

B → A: $g^b \bmod p$, $E_k(S_B(g^b, g^a))$

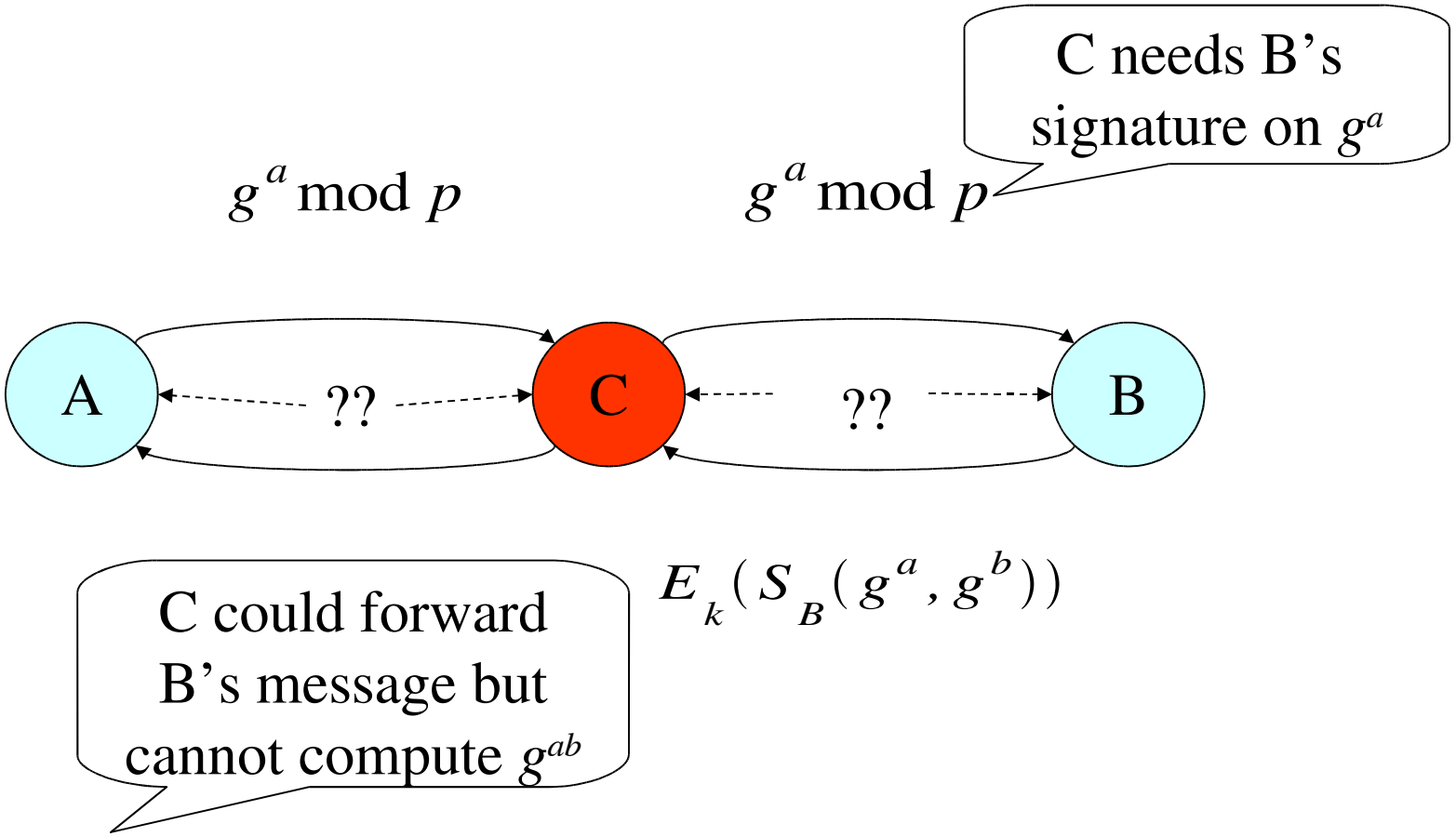
A → B: $E_k(S_A(g^a, g^b))$

shared key $k = g^{ab} \bmod p$

Security properties of STS:

- Key agreement
- Mutual entity authentication
- Explicit key authentication

Man-in-the-middle Attack?



Needham-Schroeder Protocol

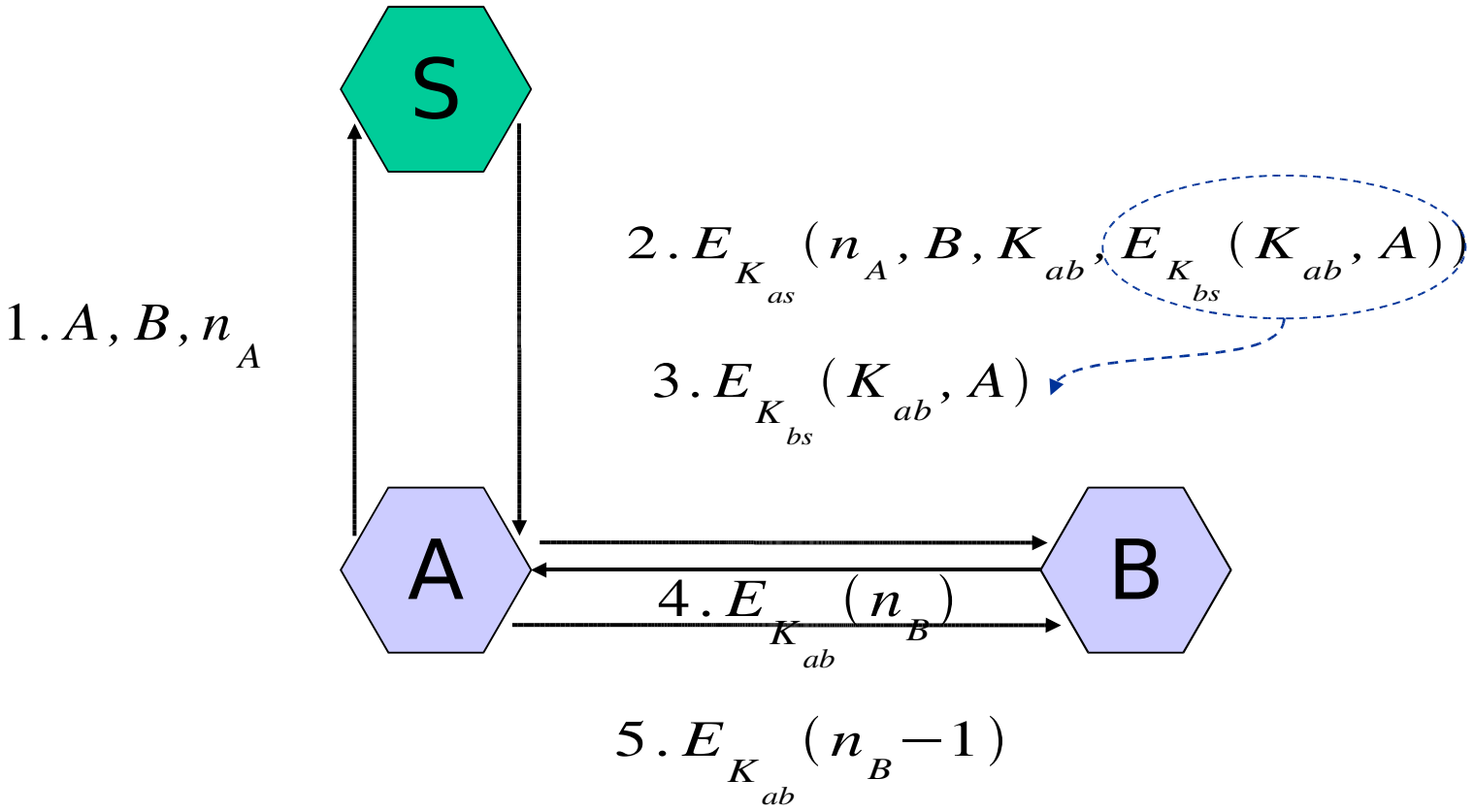
Proposed in a landmark paper in 1978 and basis for the widely used Kerberos protocol.

Key transport protocol based on a symmetric encryption algorithm: A and B obtain a session key K_{ab} from server S (Trusted Third Party).

A shares a secret key K_{as} with S, B shares a secret key K_{bs} with S.

Nonces n_A and n_B are used to prevent replay attacks.

Needham-Schroeder Protocol



Needham-Schroeder Protocol

The server (key distribution centre) has to be "trusted": it knows the session keys and could deceive A and B about the actual identity of the corresponding party.

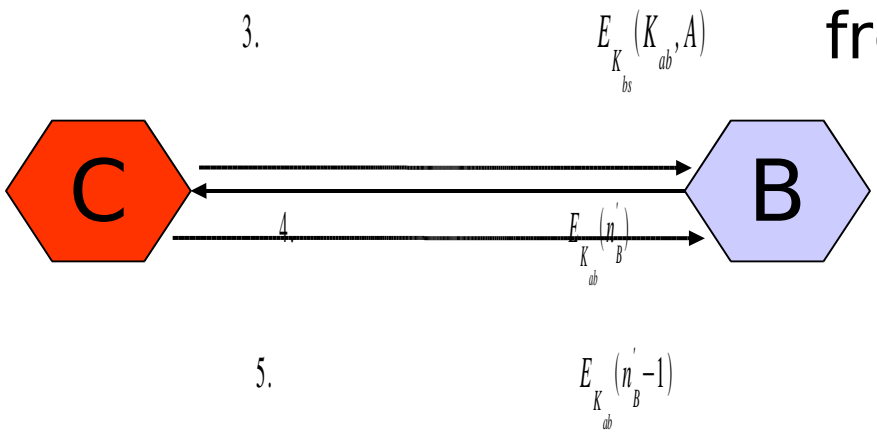
Achieves unilateral entity authentication of A to B (messages 4+5), key establishment, and key confirmation.

There exists also a public key version of the Needham-Schroeder protocol.

Denning-Sacco Attack

The NS protocol achieves its goals under the (standard) assumption that the long term keys K_{as} and K_{bs} are not compromised.

Denning & Sacco discovered an attack where the adversary C impersonates A re-using a compromised session key K_{ab}:



from original protocol run

Known Key Attack

The NS-protocol meets its goals if a single protocol run is considered.

Denning & Sacco found a problem when more than one protocol run is considered.

We have to consider:

- Compromise of long-term secret keys.
- Compromise of past session keys.

Known key attack: use a compromised past session key to compromise a future session.

Perfect Forward Secrecy

When a long-term key is compromised, we can no longer protect future sessions.

It is still desirable to design protocols where past sessions remain secure.

Perfect forward secrecy: compromise of long-term keys does not compromise past session keys.

"Forward secrecy" indicates that the secrecy of old keys is carried forward into the future.

Password-based Protocols

Use the password P to encrypt a randomly generated session key K_s; use session key to encrypt further data.

- $A \quad B$: $eP(K_s)$
- B A: eK_s(data)

Vulnerable to off-line dictionary attack.

Attacker guesses password P, decrypts first message and gets a candidate session key K'_s; decrypt the second message with K'_s.

When result is meaningful text, it is likely that the password had been guessed correctly.

Encrypted Key Exchange (EKE)

Symmetric encryption algorithm to encrypt data with the password as the key.

In a protocol run, user A generates a random public key/private key pair K_a, K_a^{-1}.

Step 1: A sends public key K_a to B, encrypted under the password P (symmetric encryption).

Step 2: B randomly generates session key K_s; sends K_s to A encrypted first under K_a (public-key enc.) and then under P (symmetric enc.):

- A B: e$P(K_a)$
- B A: eP(e$K_a(K_s)$)

Kerberos

Kerberos was developed at MIT for user authentication in a distributed system.

The parties involved are client *A*, server *B*, and Kerberos authentication server (KAS) *S*.

Based on the Needham-Schroeder key establishment ("authentication") protocol: the server provides *A* and *B* with a session key.

Uses a symmetric encryption algorithm.

Kerberos

Tickets: contain session keys, encrypted under a key shared by server and KAS.

Lifetime: validity time for tickets, to stop re-use of compromised session keys.

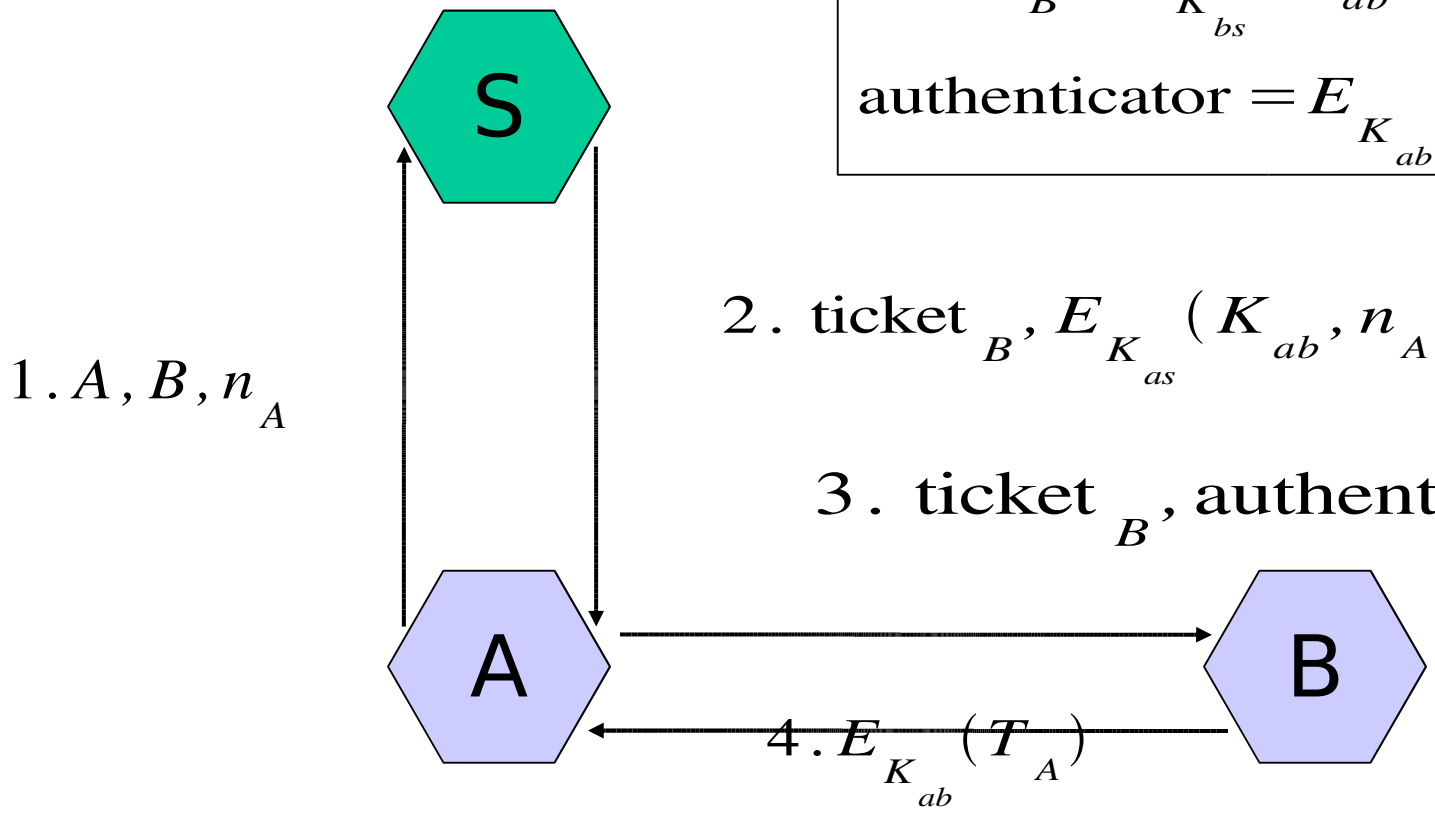
Authenticator: contains a timestamp, authenticate client to server.

Challenges (nonces) n_A and n_B are used to prevent replay attacks.

Kerberos (simplified)

$$\text{ticket}_B = E_{K_{bs}}(K_{ab}, A, L)$$

$$\text{authenticator} = E_{K_{ab}}(A, T_A)$$



Kerberos

1. Client A sends a request to S to log on to B.
2. KAS checks that it knows the client A; KAS then generates a session key K_{ab} and a ticket for B; KAS sends session key to A, encrypted under the key K_{as}, which is derived from the client's password.
3. A decrypts its part of the reply and checks the nonce; A sends ticket and authenticator to B.

Kerberos

1. B decrypts the ticket with K_{bs} and obtains the session key K_{ab}; B checks that the identifiers in ticket and authenticator match, that the ticket has not expired and that the time stamp is valid.
2. B returns the time stamp T_A encrypted under the session key K_{ab} to A.

   The validity period for time stamps has to consider the skew between the local clocks of A and B.

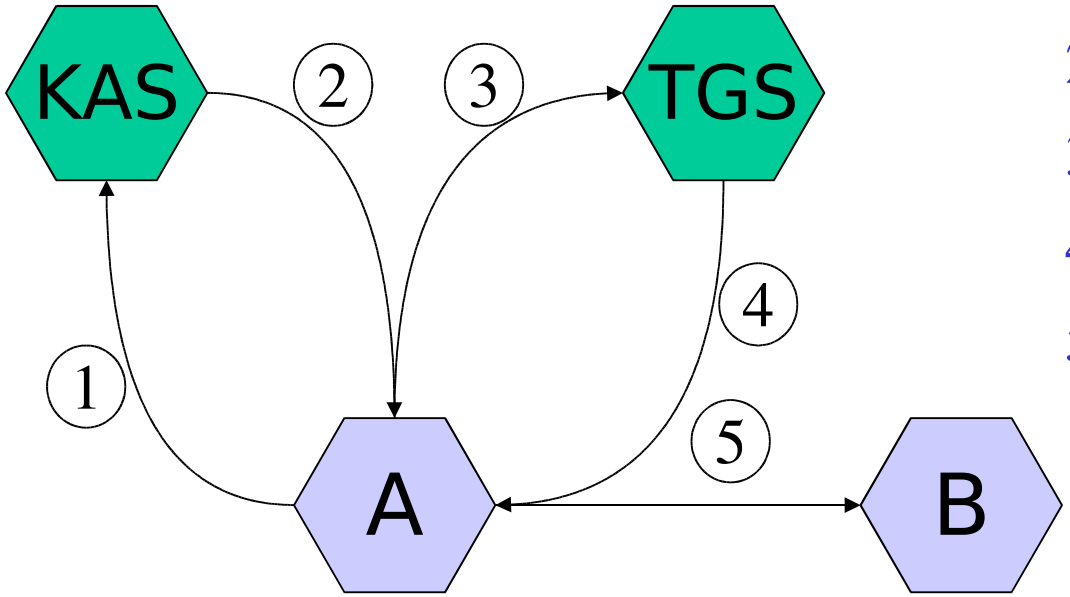
Ticket Granting Servers

The Kerberos authentication service at MIT employed Ticket Granting Servers:

In a first exchange, the client gets a ticket for the TGS.

In a next exchange, the client uses this ticket to get a service ticket from the TGS.

Ticket Granting Servers



1. Request ticket granting ticket
2. TGT
3. Request server ticket
4. Server ticket
5. Service request

Realms

A KAS with all its registered clients and servers (principals) defines a "realm".

A realm often corresponds to a single organisation.

Inter-realm authentication to get access to services in other organisations.

When a client in realm R_1 requests access to a server in realm R_2, KAS_1 issues a TGT for KAS_2.

Trust

This requires a ‘trust relationship’ between the authentication servers in different realms.

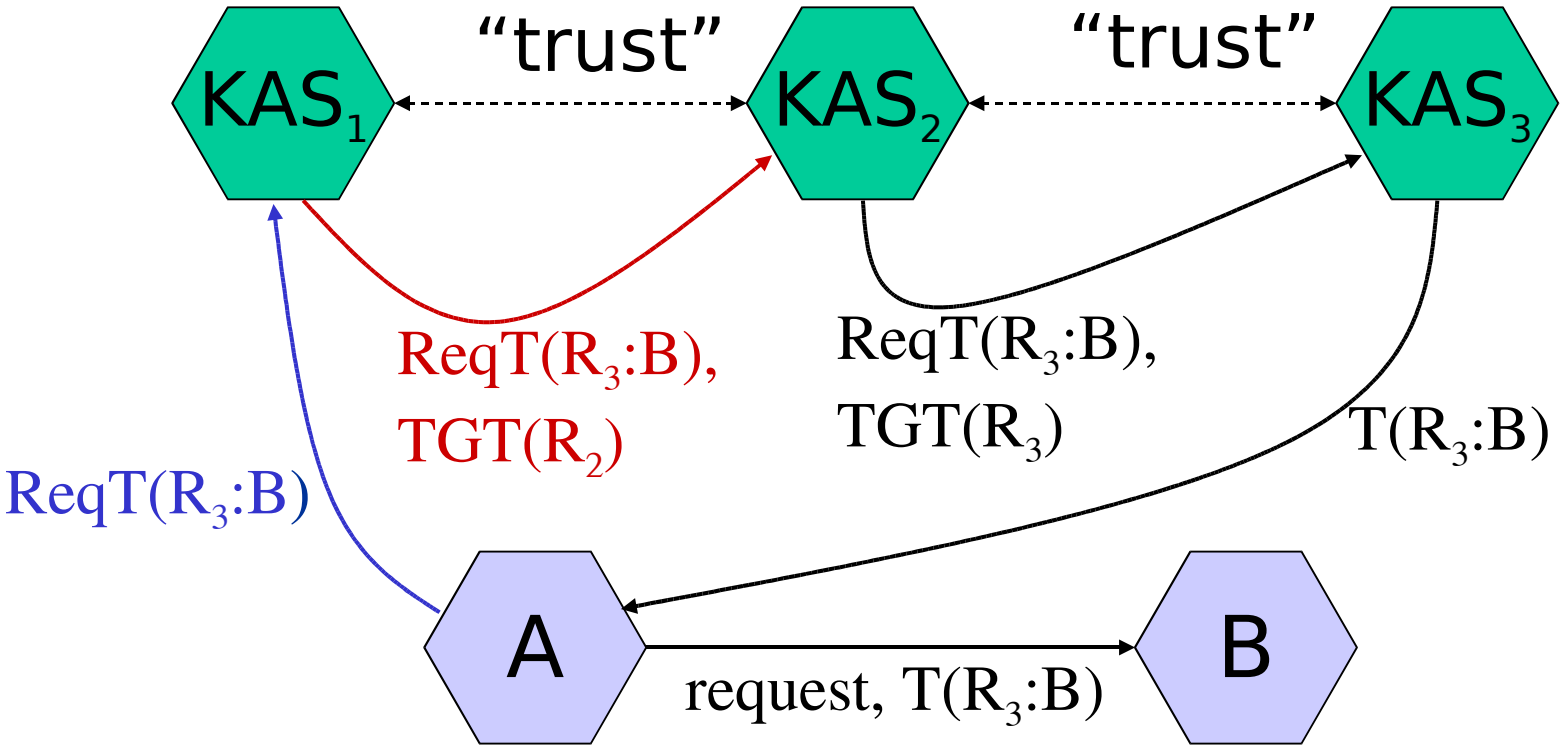
In this case, ‘trust’ is a shared secret key.

Between organisations, key sharing tends to be underpinned by contractual agreements.

Transitivity of trust: Assume there is trust between R_1 and R_2, and between R_2 and R_3; can a client in R_1 get access to a server in R_3?

The answer depends on the situation.

Inter-realm Authentication



Inter-realm Authentication

1. Client A in realm R_1 requests a ticket for a server B in realm R_3 from its KAS.
2. KAS_1 has a "trust relationship" with KAS_2, generates a TGT for realm R_2 and forwards this TGT together with A's requests to KAS_2.
3. KAS_2 has a "trust relationship" with KAS_3, generates a TGT for realm R_3 and forwards this TGT together with A's requests to KAS_3.
4. KAS_3 sends the ticket for B to A.
5. Client A presents this ticket when requesting a service from B.

Kerberos in Windows

Authentication protocol of choice in Windows.

Windows domains correspond to Kerberos realms; domain controllers act as KDCs.

Kerberos principals are users and machines.

Windows authentication is the basis for access control; principals in Windows access control: SID.

- Note that there are two definitions of principal!

Kerberos ticket [RFC 1510] contains mandatory field cname (client name) and optional field authorization-data.

Windows: cname holds principal's name and realm, e.g. diego@tuhh.de, authorization-data holds the group SIDs.

Delegation – Proxy Tickets

Alice needs a service from Bob, where Bob has to access servers on her behalf.

Alice knows in advance what Bob is going to need: she applies for proxy tickets for the relevant servers and gives the tickets and the corresponding session keys to Bob.

Proxy tickets contain special authorizations that limit how Bob can use Alice's credentials, e.g. state name of a file Bob is allowed to print.

Delegation – Forwarded Tickets

Alice does not know in advance what Bob is going to need.

Alice applies for a forwarded TGT for Bob and transfers this ticket and corresponding session key to Bob.

Alice 'delegates her identity' to Bob; Bob can now apply for tickets on her behalf.

Bob can impersonate Alice: "The fast and loose way to delegate credentials" [Brown]. Principals can be nominated as OK-AS-DELEGATE to have some control over the delegation of credentials (identities).

Revocation

Access rights revoked from a principal by updating KAS and TGS databases.

Revocation takes effect when the principal next requests a ticket from the TGS; tickets the principal has in possession are valid until they expire.

TOCTTOU problem!

Trade-off between convenience and security:

- Long ticket lifetime: TGS may occasionally be off-line, but revocation of access rights takes effect with a longer delay.
- Short ticket lifetime: principals have to update their tickets more regularly and the availability of the security servers is more important for system performance.

Kerberos

We have only sketched the basic features of Kerberos.

Kerberos version 5 has been specified by the IETF as RFC 1510.

Kerberos is used in the Windows operating system as the preferred replacement for proprietary authentication protocols.

The initial user request is not authenticated.

If this is deemed a problem, the AS may ask for pre-authentication before issuing a ticket.

Public Key Infrastructures

Certificates

X.509

Electronic Signatures

Certificates

How to bind a name to a public key?

Original suggestion: Public directory of user names and keys, just like a phone directory.

Kohnfelder [1978]: implement the directory as a set of digitally signed data records containing a name and a public key; he coined the term certificate for these records.

Certificates originally had a single function: binding between names and keys.

X.509 – ISO/IEC 9594-8 The Directory: Authentication Framework

ITU-T Recommendation X.509: part of X.500

X.500: intended as a global, distributed database of named entities: people, computers, printers, etc, i.e. a global, on-line telephone book.

The information held by the Directory is typically used to facilitate communication between, with, or about objects such as application-entities, people, terminals and distribution lists.

X.509

X.509 certificates: were intended to bind public keys [originally passwords] to X.500 path names (Distinguished Names) to note who has permission to modify X.500 directory nodes.

Geared towards identity based access control: *Virtually all security services are dependent upon the identities of communicating parties being reliably known, i.e. authentication.*

This view of the world predates applets and many new e-commerce scenarios.

X.509 certificates

User certificate (public key certificate, certificate): The public key of a user, together with some information, rendered unforgeable by encipherment with the secret key of the certification authority which issued it.

Attribute certificate: A set of attributes of a user together with some other information, digitally signed under the private key of the CA.

Certification authority: an authority trusted by one or more users to create and assign certificates.

X.509v3 Certificate Format

version (v3)
serial number
signature algorithm id
issuer name
validity period
subject name
subject public key info
issuer unique identifier
subject unique identifier
extensions

Extensions: added to increase flexibility

Critical extensions: if a critical extension cannot be processed, the certificate must be rejected

Critical extensions are also used to standardize

extensionID
critical: YES/NO
extensionValue

X.509v3 – Evaluation

Criticised for using ASN.1 formats: but now we have XML …

Criticised for not being flexible enough.

Criticised for being too flexible (extensions).

Different implementations of the standard are not necessarily interoperable:

- EEMA PKI Challenge, http://www.eema.org/

Distinguish between the X.509 certificate format and its intended application.

PKIX – RFC 3280
Internet X.509 Public Key Infrastructure

Public Key Certificate (PKC): A data structure containing the public key of an end-entity and some other information, which is digitally signed with the private key of the CA which issued it.

Attribute Certificate (AC): A data structure containing a set of attributes for an end-entity and some other information, which is digitally signed with the private key of the Attribute Authority which issued it.

PKIX

Public Key Infrastructure (PKI): The set of hardware, software, people, policies and procedures needed to create, manage, store, distribute, and revoke PKCs based on public-key cryptography.

Privilege Management Infrastructure (PMI): A collection of ACs, with their issuing Attribute Authorities, subjects, relying parties, and repositories.

Validity

Certificates have expiry dates, validity periods.

Misconception: a certificate cannot be used after it has expired.

Deciding what should be done with expired certificates is a policy decision.

Example: entry policies for EU passports

- the passport has to be valid x months beyond entry
- the passport has to be valid until exit (US)
- the passport has to be valid on entry
- the passport has expired less than a year ago (EU)

Validity

How to evaluate a certificate chain?

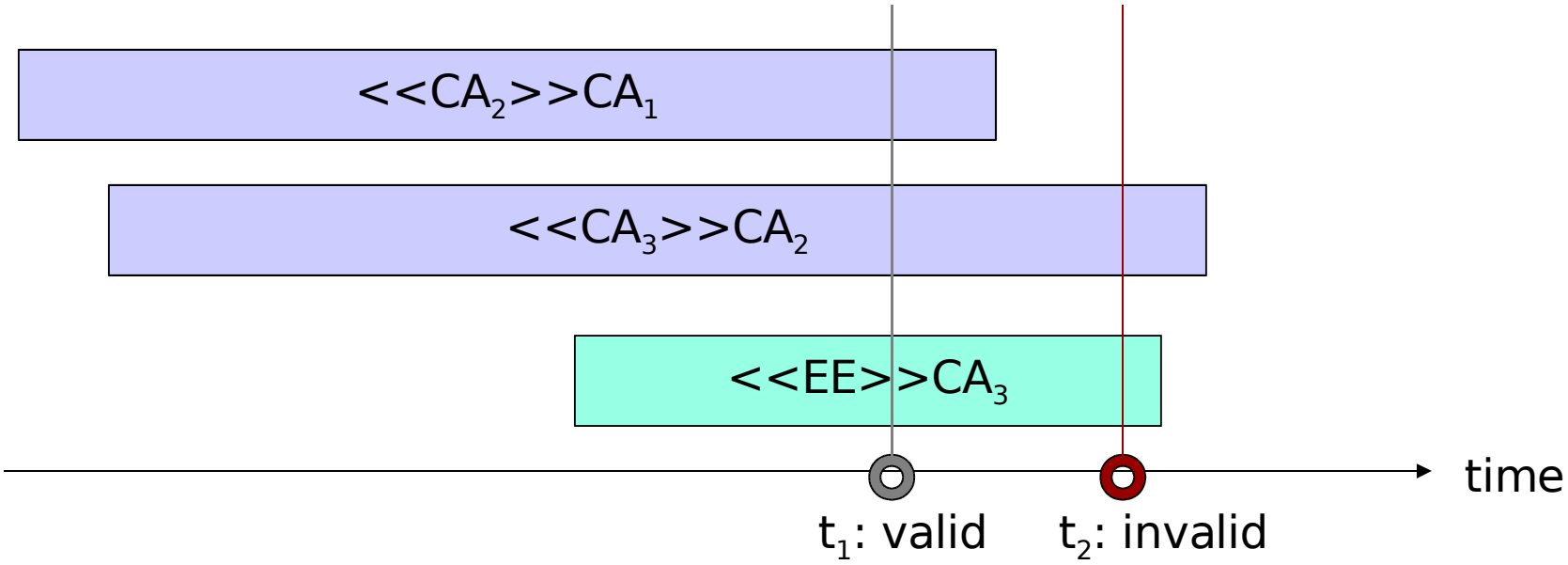
- certificates may expire
- certificates may be revoked

Shell model: all certificates have to be valid at the time of evaluation.

Chain model: the issuer's certificate has to be valid at the time the certificate was issued

Policies beyond the shell and chain model have been suggested.

Shell Model



Certificate <<EE>>CA_3 valid at time t_1 as all three certificates are valid.

Certificate <<EE>>CA_3 invalid at time t_2 as certificate <<CA_2>>CA_1 has expired.

Shell Model

Conservative approach.

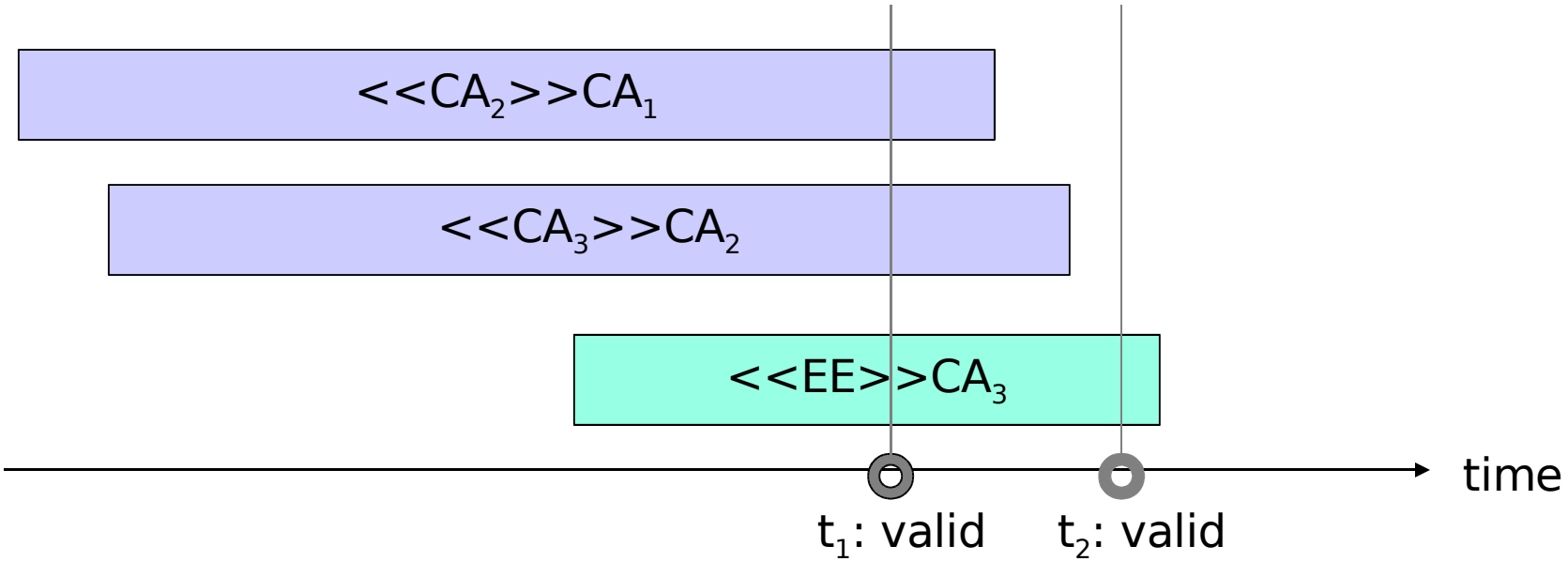
Policy implemented in SPKI.

CAs should only issue certificates that expire before their own certificate.

If a top level certificate expires or is revoked, all certificates signed by the corresponding private key have to be re-issued under a new key.

Appropriate for certificates defining hierarchical address spaces.

Chain Model



Certificate <<EE>>CA_3 is valid at times t_1 and t_2 :

<<CA_3>>CA_2 valid when <<EE>>CA_3 was issued

<<CA_2>>CA_1 valid when <<CA_3>>CA_2 was issued

Chain Model

Requires a time-stamping service (some means of reliably establishing when a certificate was issued).

If a top level certificate expires or is revoked, certificates signed by the corresponding private key remain valid.

Example: an organisation issues membership certificates signed by a manager; when the manager leaves and his certificate is revoked, should all membership certificates be re-issued?

Validity

A certificate cannot tell the end user what the end user's policy is.

A certificate can tell the end user what the CA's policy is and may limit the CA's liability.

Policy decisions have consequences:

- Shell model: certificates have to be re-issued.
- Chain model: certificates should be time-stamped.

Time Stamping

Applications may need independent evidence about the time documents are signed.

Time Stamp Authority (TSA): a TTP who provides a “proof-of-existence” for a particular datum at an instant in time.

A TSA does not check the documents it certifies.

TSP: PKIX Time Stamp Protocol [RFC 3161]

Revocation

Certificates may have to be revoked:

- if a corresponding private key is compromised,
- if a fact the certificate vouches for no longer is valid.

Certification Revocation Lists (CRLs):

- original solution proposed in X.509
- distributed in regular intervals or on demand,
- Delta-CRL: record only the changes since the issue of the last CRL.

CRLs make sense if on-line checks are not possible or too expensive.

Revocation On-line

When on-line checks are feasible, CRLs can be queried on-line

When on-line checks are feasible, certificate status can be queried on-line

- Online Certificate Status Protocol - OCSP [RFC 2560]
- positive lists in the German signature infrastructure

A CA issuing certificates for its own use (e.g. for access control) requires only a local CRL.

Alternative to revocation: short-lived certificates

Electronic Signatures

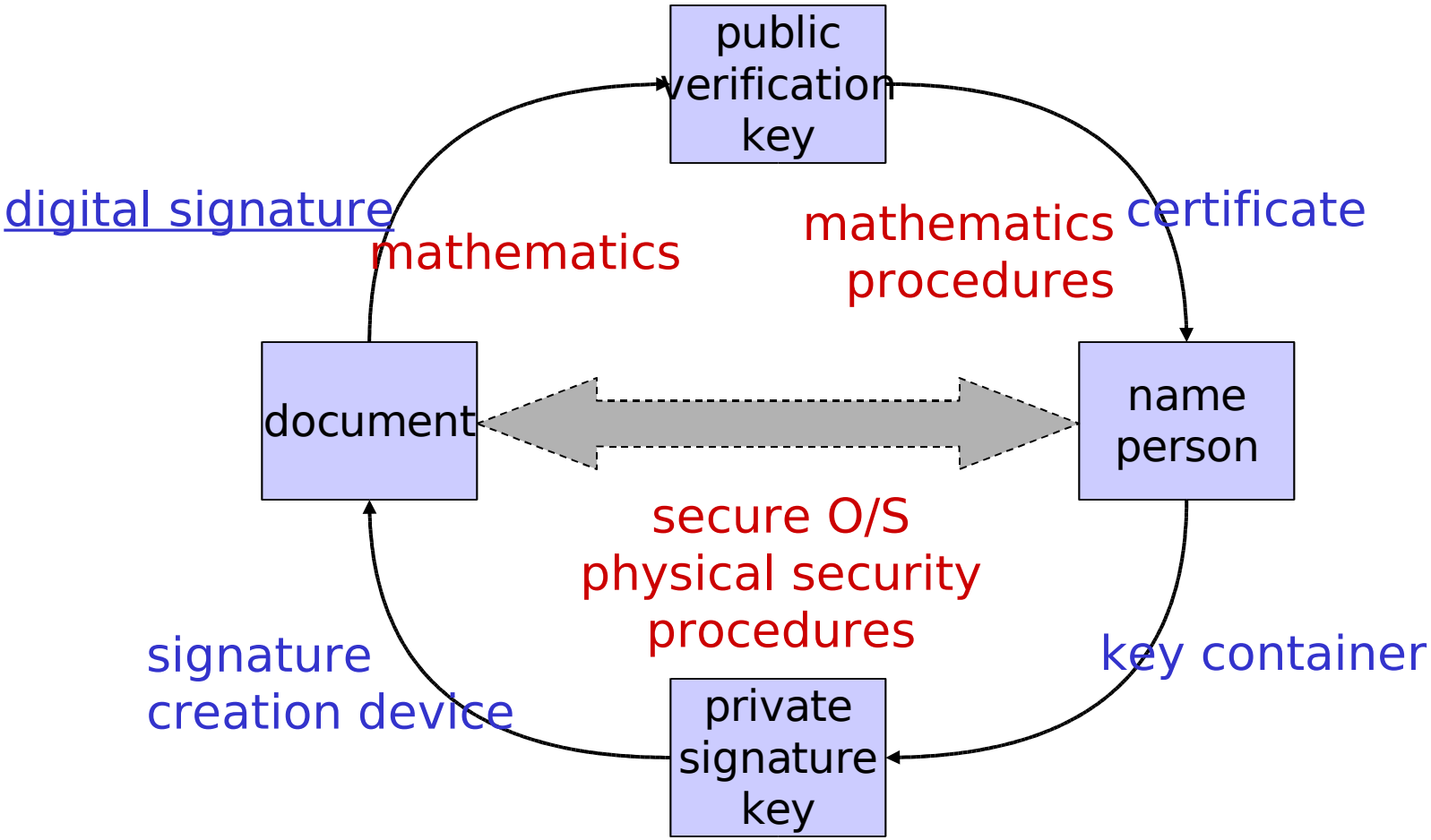
Digital signature: cryptographic mechanism for associating documents with verification keys.

Electronic signature: security service for associating documents with persons.

Electronic signature services usually use digital signatures as a building block but could be implemented without them.

Certificates can record the binding between the name of a person and a key.

Electronic Signatures



Trusted Computing: Attestation

Distributed application: request arrives from a remote source.

For access control decisions, we might want to know which application issued the request.

How can we "trust" any claim about the application making the request?

A system has to be able to make statements about the software it is running.

- Related to secure boot.

Other systems have to verify such statements.

Attestation

Trusted Platform Module (TPM): hardware module that can sign statements about the software it is running.

Signature key (endorsement key *EK*) installed by hardware manufacturer.

Certificates for public verification key issued by hardware manufacturers

- "This is a XYC Trusted Computing Module"

Hardware = "root of trust".

Attestation Keys

If all attestations from a TPM are signed by the same key, an observer could them link all.

To make attestations unlinkable, the TPM can create Attestation Identity Keys (AIKs).

An AIK is an RSA signature key pair generated by the TPM.

The TPM needs the services of a TTP, a so-called privacy CA (pCA) to get a certificate that confirms that the AIK belongs to a genuine TPM.

Attestation Keys

A protocol for obtaining such a certificate:

TPM sends its public endorsement key EK and the public part of the attestation identity key AIK_i to pCA.

The CA checks that EK belongs to a genuine TPM, stores the mapping between EK and AIK_i, and returns the certificate $Cert_{pCA}$ to the TPM.

The TPM uses the private part of AIK_i to sign the PCR contents in an attestation and includes $Cert_{pCA}$ in the message sent to the verifier.

- TPM pCA: EK, AIK_i
- pCA TPM: $Cert_{pCA}$
- TPM Verifier: AIK_i, $sAIK_i(PCR)$, $Cert_{pCA}$

Unlinkable Attestation

In the first message all attestation keys are linked to EK, and thus all attestations can still be linked.

There has been further work on this problem, e.g. on Direct Anonymous Attestation.

Full anonymity is not desirable. It must be possible to recognize attestations coming from TPMs known to be compromised.

Special cryptographic protocols exist that achieve these competing goals.