b

UNIVERSITÄT
BERN

b UNIVERSITÄT BERN

Network Security

V. Key Management

Prof. Dr. Torsten Braun, Institut für Informatik

Bern, 21.03.2022 – 28.03.2022

Key Management

Table of Contents

1. Introduction

1. Cryptographic Key Management

- Secure use of cryptographic key algorithms depends on protection of cryptographic keys
- Cryptographic key management: key
  - generation
  - creation
  - protection
  - storage
  - exchange
  - replacement
  - use
- Key management also involves monitoring and recording of each key's access, use, and context.
- Key management system includes
  - key servers
  - user procedures
  - protocols

1. Introduction

2. Symmetric Key Distribution

– Key distribution:
means of delivering a key to two parties, who wish to exchange data without allowing others to see the key

– For symmetric encryption to work, the two parties to an exchange must share the same key,
and that key must be protected from access by others.

– Frequent key changes are desirable to limit the amount of data compromised, if an attacker learns the key.

1. Introduction

3. Symmetric Key Distribution Alternatives

– A can select a key and physically deliver it to B.

– A third-party C can select the key and physically deliver it to both A and B.

– If A and B have previously and recently used a key, one party can transmit the new key to the other, encrypted using the old key.

– If both A and B have an encrypted connection to a third-party C (key distribution center), C can deliver a key on the encrypted links to A and B.

2. Symmetric Key Distribution with Symmetric Encryption

1. Third Party Key Distribution Options

Key Translation Center

– transfers keys between 2 entities.

– Decryption and encryption of keys

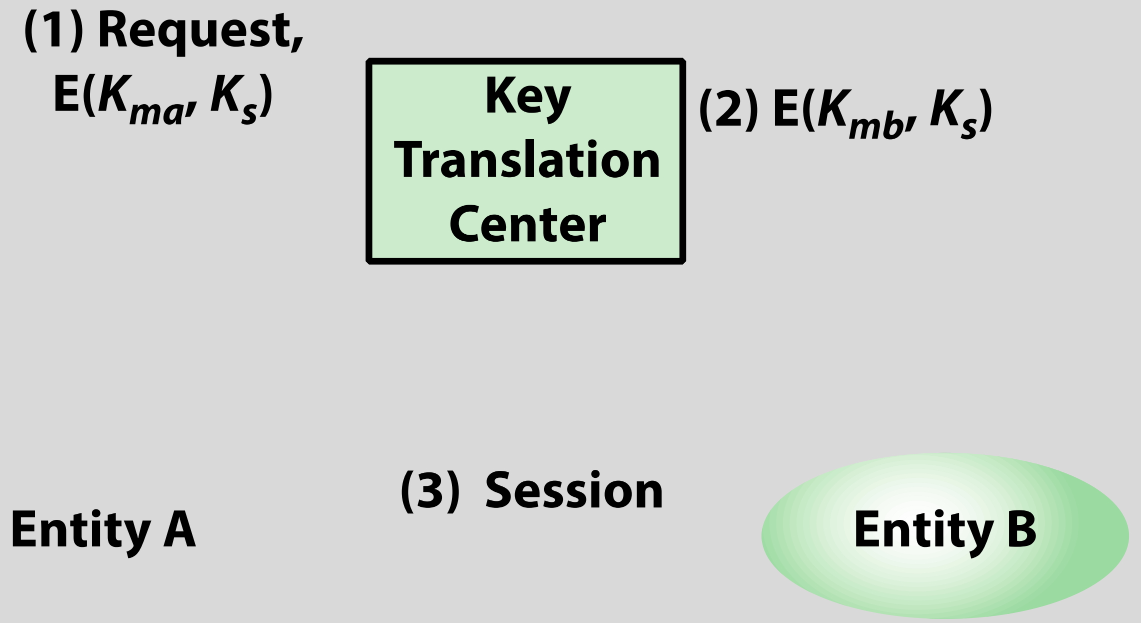
Key Distribution Center

– generation and distribution of session keys.

1. Key Translation
2. Key Translation with Key Forwarding
3. Key Distribution
4. Key Distribution with Key Forwarding

2. Symmetric Key Distribution with Symmetric Encryption

1.1 Key Translation



u^b

^{b} UNIVERSITÄT BERN

2. Symmetric Key Distribution with Symmetric Encryption

1.2 Key Translation with Key Forwarding

(1) Request,
E(K_{ma}, K_s)

(2) E(K_{mb}, K_s)

(3) E(K_{mb}, K_s)

(4) Session

2. Symmetric Key Distribution with Symmetric Encryption

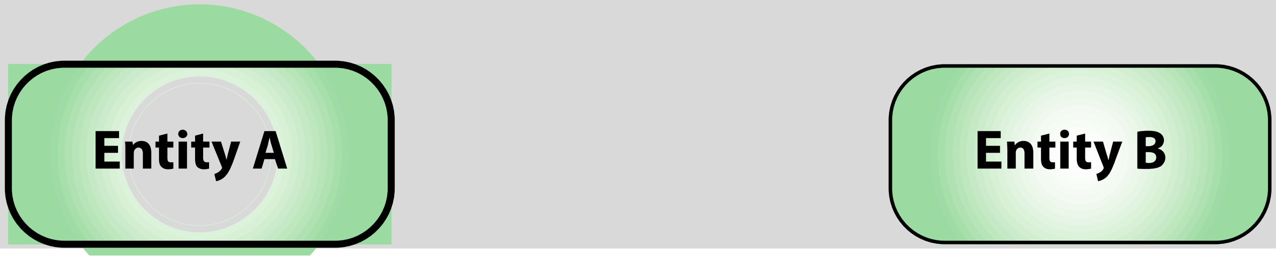
1.3 Key Distribution

(2b) E(K_{mb}, K_s)

(2a) E(K_{ma}, K_s)

2. Symmetric Key Distribution with Symmetric Encryption

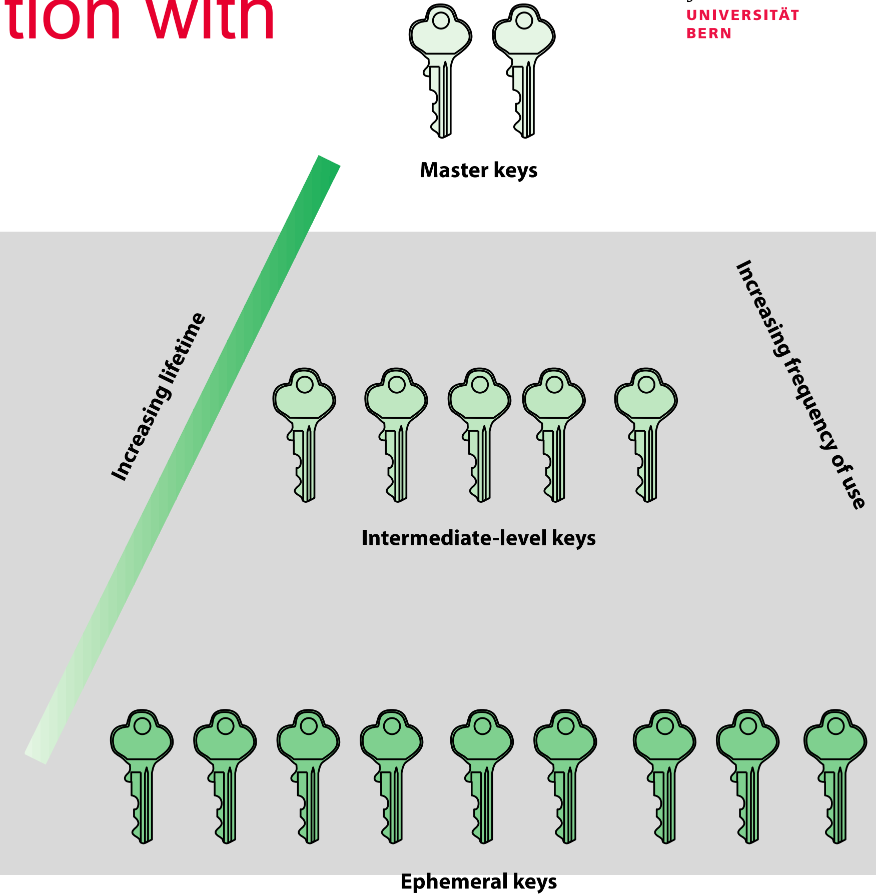
1.4 Distribution with Key Forwarding



2. Symmetric Key Distribution with Symmetric Encryption

2. Key Hierarchy

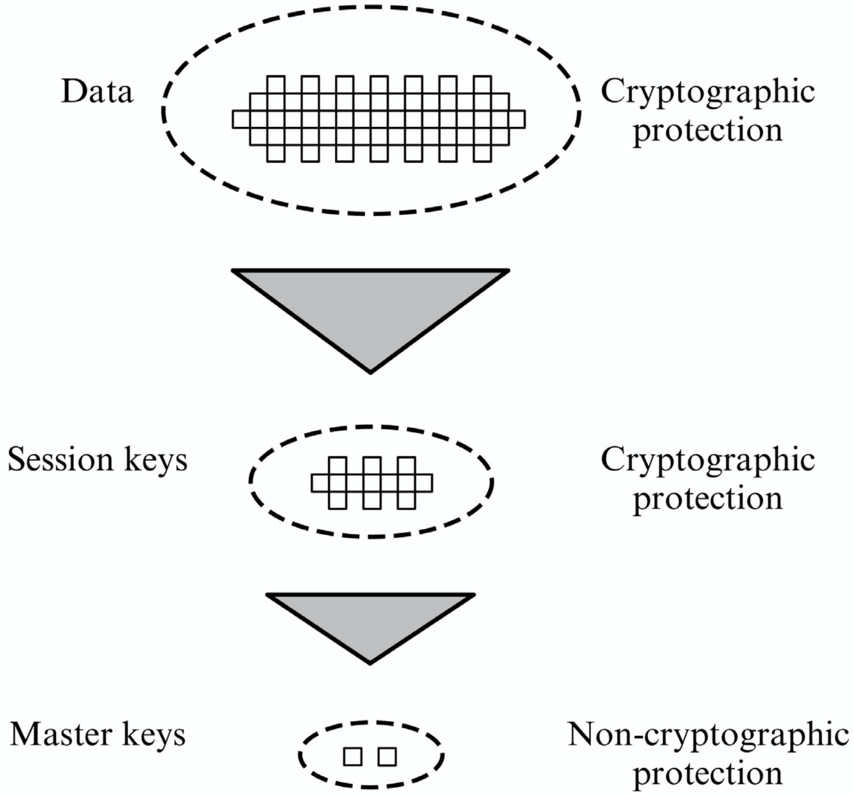
- Higher level protocols and keys are used to encrypt and exchange lower-level keys.
- Infrequently used higher level keys are more resistant to cryptanalysis.



2. Symmetric Key Distribution with Symmetric Encryption

2.1 Master and Session Key

- KDC is based on a hierarchy of keys.
- Session keys for the duration of a logical session, or for a certain time interval / number of messages
- Master key is used to encrypt session key transfer.
- Master key shared between KDC and user / end system



2. Symmetric Key Distribution with Symmetric Encryption

2.2 Hierarchical Key Control

- Local KDCs for small domain, e.g., local area network, responsible for key exchange between users of this small domain.
- If entities in two different domains need a key, the two local KDCs can communicate over a global KDC.
- Scheme can be extended to three or more layers.
- Advantages
  - Scalability
  - Limited damage in case of breaches

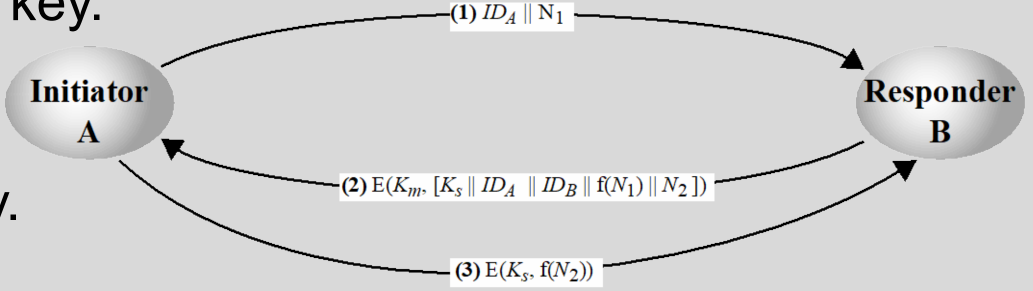
2. Symmetric Key Distribution with Symmetric Encryption

3. Decentralized Key Control

Session key establishment

1. A issues request for session key.
2. B responds with encrypted message including session key using shared master key.
3. A returns f(N_2) using new session key.

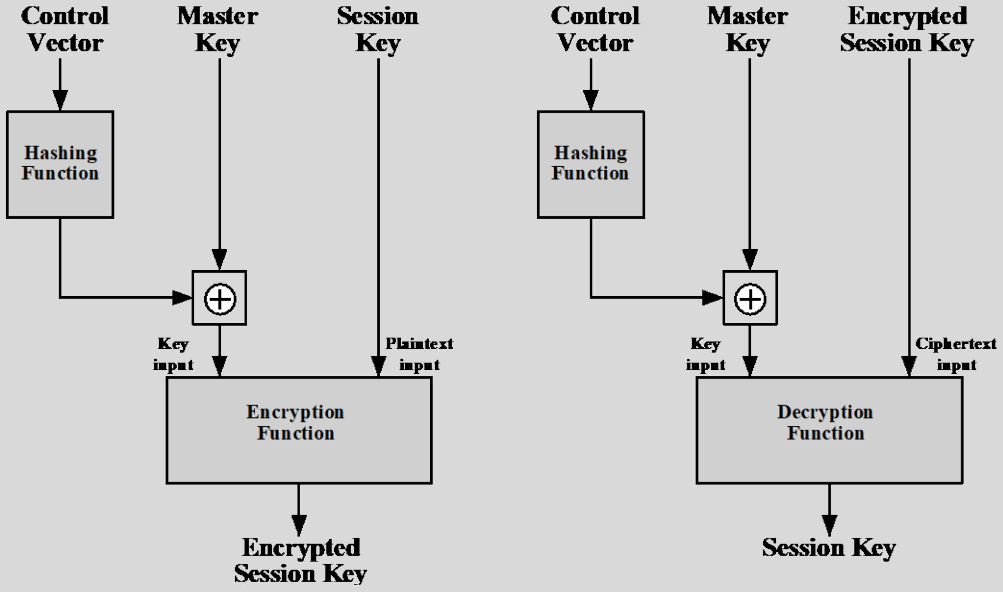
N (N-1)/2 master keys required



2. Symmetric Key Distribution with Symmetric Encryption

4. Controlling Key Usage

- Different types of keys for different applications, e.g.
  - Data communication encryption
  - PIN encryption
  - File encryption
- Tags with each key can indicate usage type.
- Control vectors can be used to specify uses and restrictions for session key.

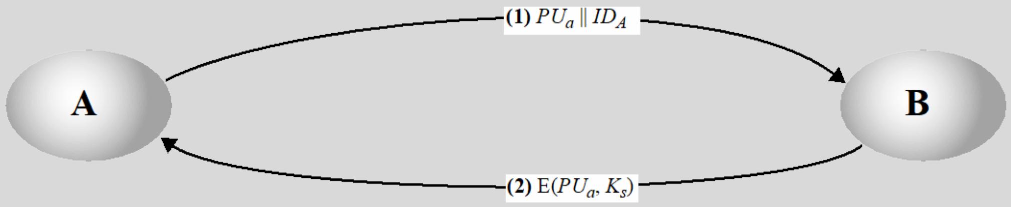


Control Vector Encryption / Decryption

3. Symmetric Key Distribution with Asymmetric Encryption

1. Simple Secret Key Distribution

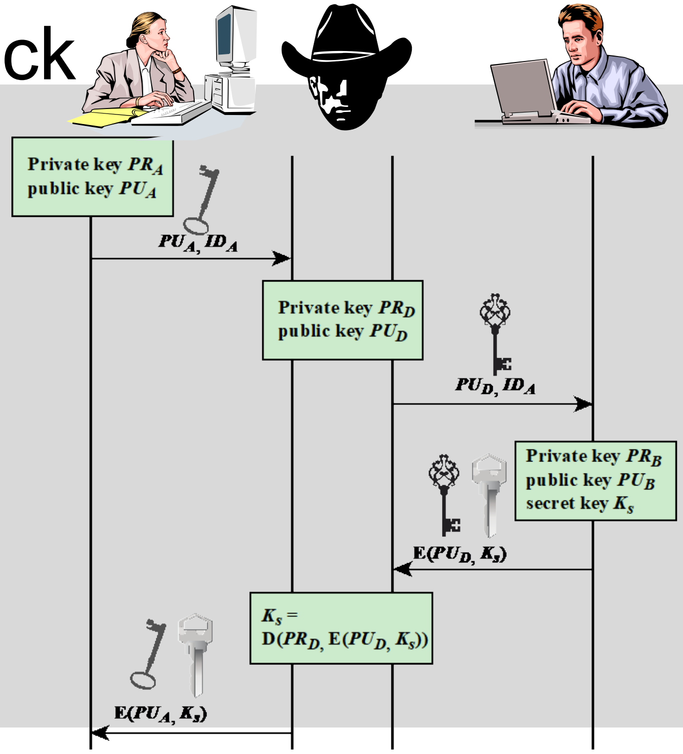
1. A generates public / private key pair (PU_a, PR_a) and transmits message with public key PU_a to B.
2. B generates secret key K_s and transmits encrypted message to A using A's public key PU_a
3. A computes $D(PR_a, E(PU_a, K_s))$ to recover secret key.
4. A and B discard PU_a and PR_a.



3. Symmetric Key Distribution with Asymmetric Encryption

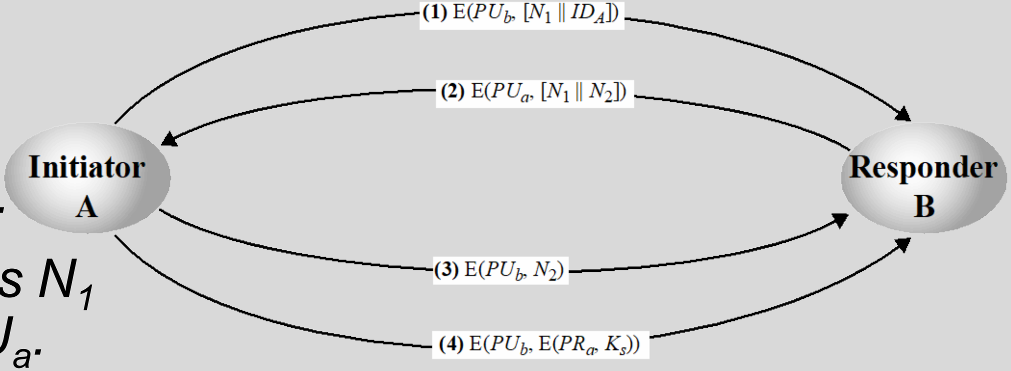
2. Another Man-in-the-Middle Attack

1. A generates public/private pair key and transmits message to B.
2. D intercepts message, creates own new public/private key pair, and submits PU_d, ID_A to B.
3. B generates secret key and transmits $E(PU_d, K_s)$ to A.
4. D intercepts message and learns secret key.
5. D transmits $E(PU_a, K_s)$ to A



3. Symmetric Key Distribution with Asymmetric Encryption

3. Secret Key Distribution with Confidentiality and *Authentication*

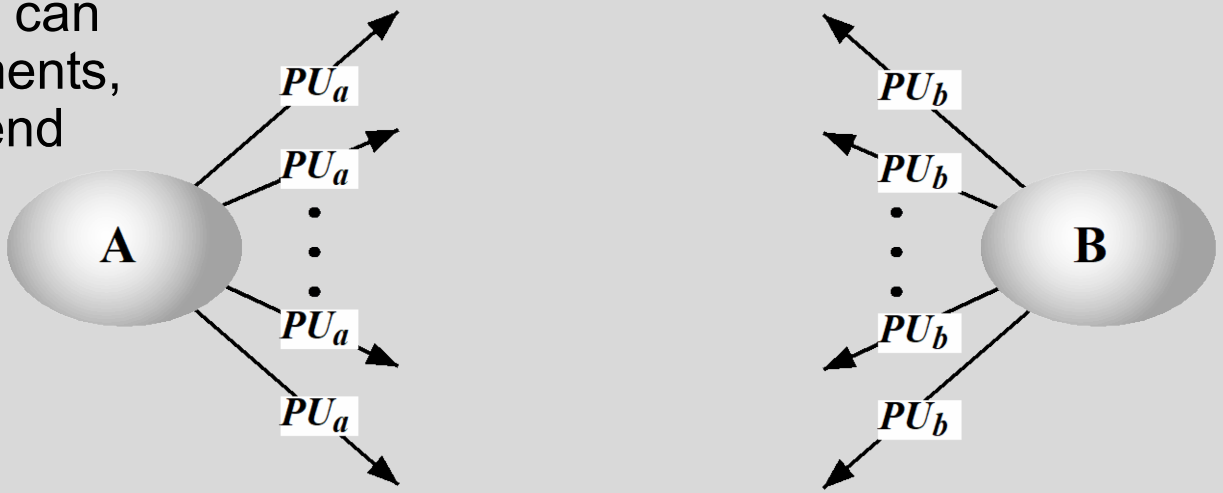

(1) E(PU_b, [N_1 ‖ ID_A])
(2) E(PU_a, [N_1 ‖ N_2])
Initiator A
Responder B
(3) E(PU_b, N_2)
(4) E(PU_b, E(PR_a, K_s))

1. *A uses B's public key PU_b to encrypt message with N_1 to B.*
2. *B sends message with nonces N_1 and N_2 to A encrypted with PU_a.*
3. *A returns nonce N_2 encrypted using B's public key PU_b.*
4. A selects secret key K_s and sends message to B.
5. B computes (D(PU_a, D(PR_b, M)) to recover key

4. Distribution of Public Keys

1. Public Announcement of Keys

Convenient, but anyone can forge public announcements, i.e., can users can pretend to be other users.



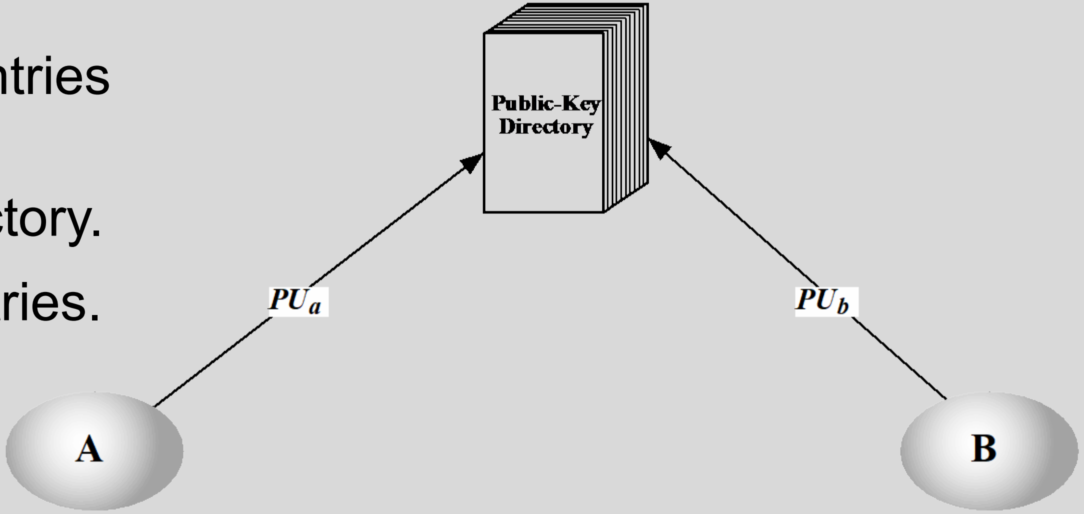
4. Distribution of Public Keys

2. Publicly Available Directory

– Public-Key Directory with entries [name, public key]

– Participants register at directory.

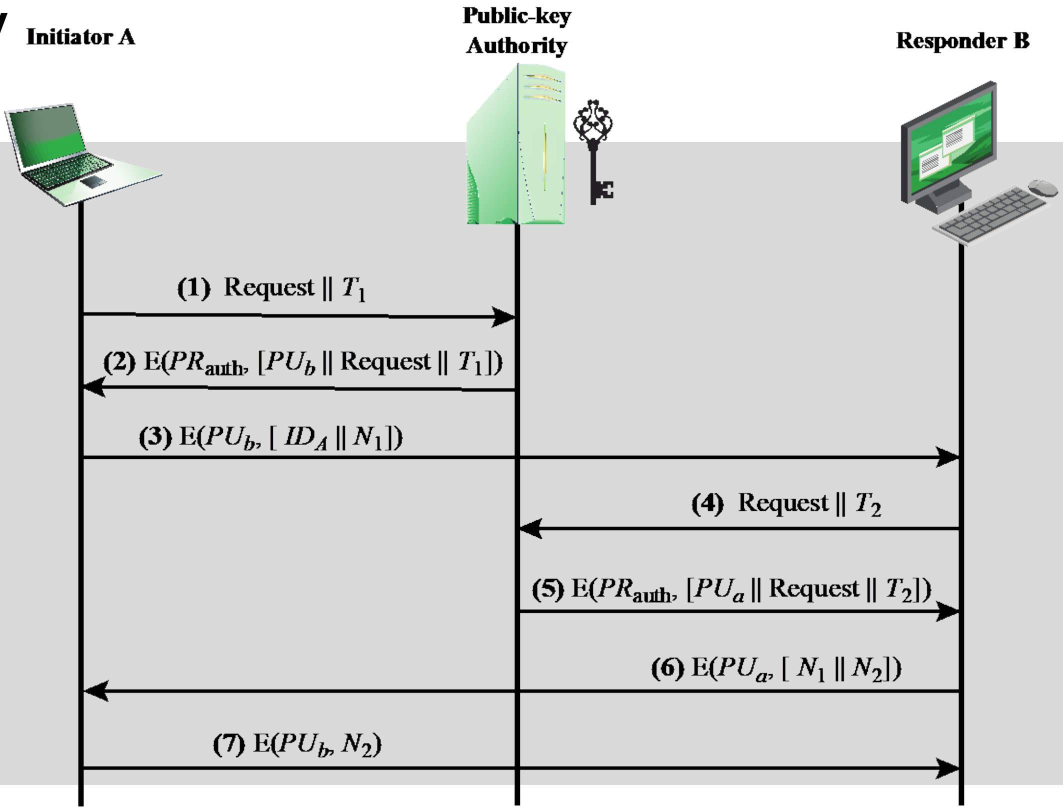
– Participants can replace entries.



4. Distribution of Public Keys

3. Public-Key Authority

- Again: directory with public keys of all participants
- In addition: participants know public key of PKA
- Disadvantage: PKA as bottleneck
- Alternative approach: direct key exchange but keys are signed by PKA


Initiator A
Public-key Authority
Responder B
(1) Request || T_1
(2) $E(PR_{auth}, [PU_b || Request || T_1])$
(3) $E(PU_b, [ID_A || N_1])$
(4) Request || T_2
(5) $E(PR_{auth}, [PU_a || Request || T_2])$
(6) $E(PU_a, [N_1 || N_2])$
(7) $E(PU_b, N_2)$

4. Distribution of Public Keys

4. Public-Key Certificates

Requirements

– Any participant can read certificates issued by Certificate Authority.

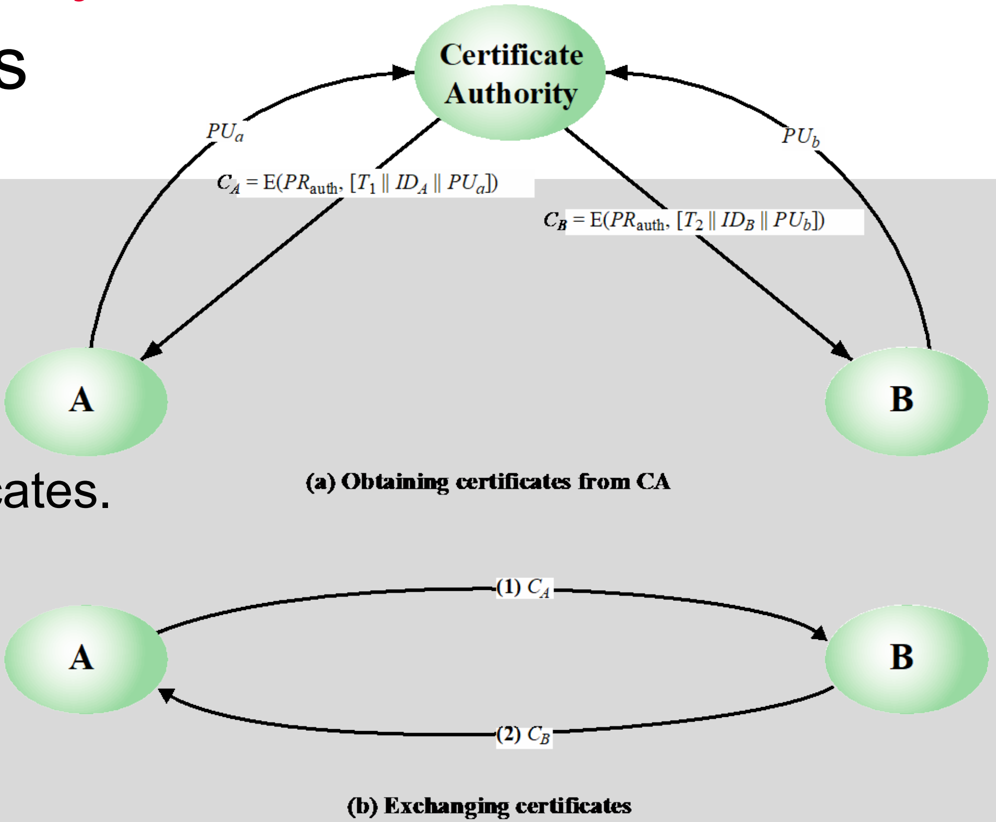
– Any participant can verify that certificates originated from CA.

– Only CA can create and update certificates.

– Any participant can verify time validity of certificates.

Certificate verification

– $D(PU_{auth}, C_A) = D(PU_{auth}, E(PR_{auth}, [T || ID_A || PU_a])) = (T || ID_A || PU_a)$


Certificate Authority
PU_a
PU_b
$C_A = E(PR_{auth}, [T_1 || ID_A || PU_a])$
$C_B = E(PR_{auth}, [T_2 || ID_B || PU_b])$
A
B
(1) C_A
(2) C_B
A
B

(a) Obtaining certificates from CA

(b) Exchanging certificates

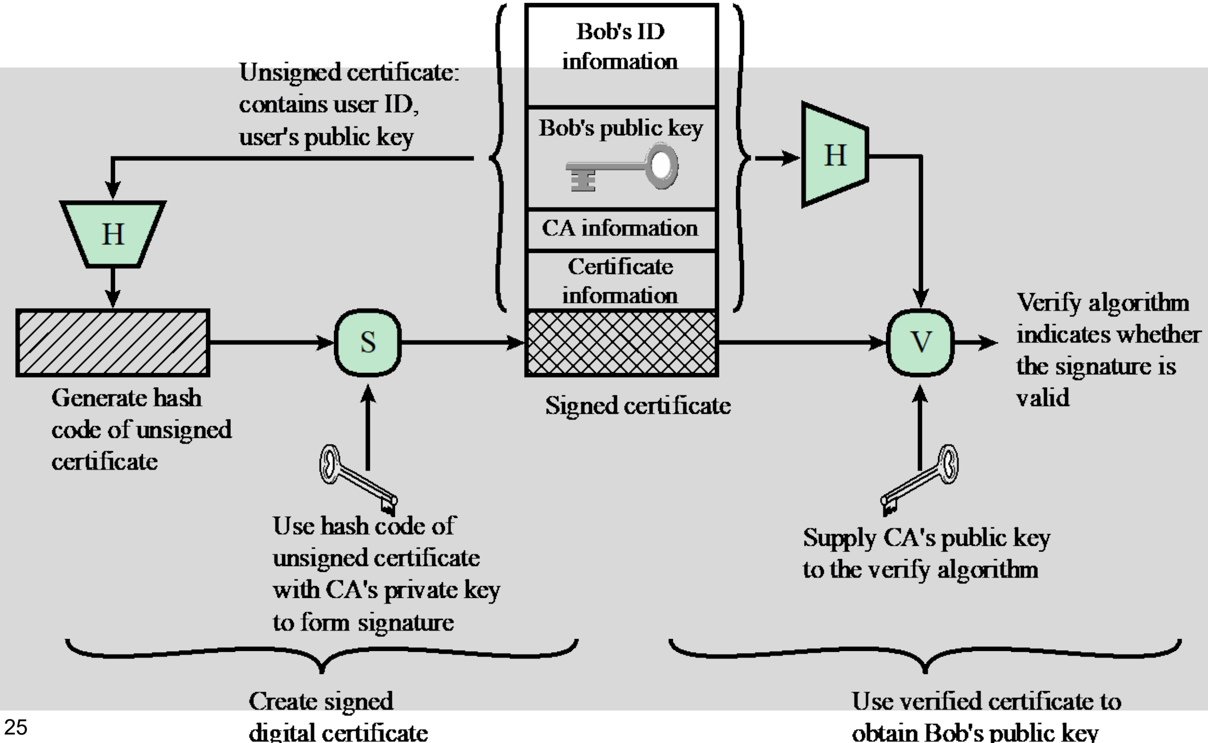
5. X.509 Certificates and Public Key Infrastructure

1. ITU Recommendation X.509

- Part of the X.500 series of recommen-dations that define a directory service
  - The directory is a server or distributed set of servers maintaining a database of information about users
- X.509 defines a framework for the provision of authentication services by the X.500 directory to its users
  - is based on the use of public-key cryptography and digital signatures
  - does not dictate the use of a specific algorithm but recommends RSA
  - does not dictate a specific hash algorithm
- Each certificate contains the public key of a user and is signed with the private key of a trusted certification authority.
- X.509 defines alternative authentication protocols based on the use of public-key certificates.

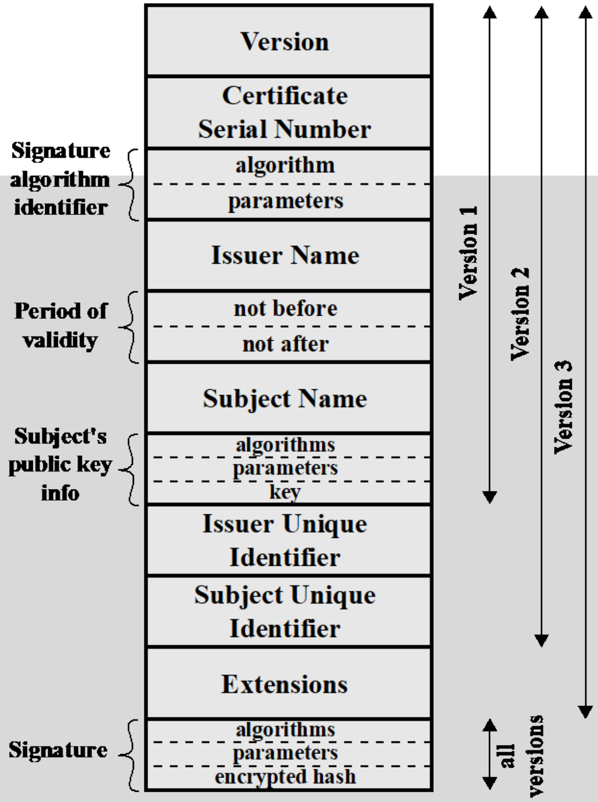
5. X.509 Certiticates and PKI

2. X.509 Public-Key Certificate Use

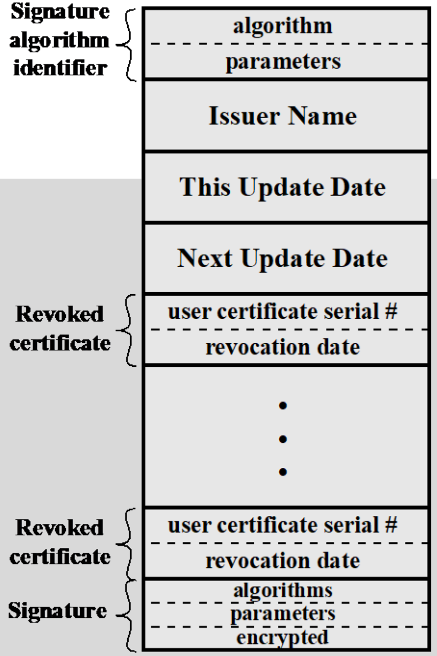

Bob's ID information
Bob's public key
CA information
Certificate information
Unsigned certificate: contains user ID, user's public key
H
H
Generate hash code of unsigned certificate
S
Signed certificate
V
Verify algorithm indicates whether the signature is valid
Use hash code of unsigned certificate with CA's private key to form signature
Supply CA's public key to the verify algorithm
Create signed digital certificate
Use verified certificate to obtain Bob's public key

5. X.509 Certificates and PKI

3.1 X.509 Formats



(a) X.509 Certificate



(b) Certificate Revocation List

5. X.509 Certificates and PKI

3.2 X.509 Formats

– **Version**: Differentiates among successive versions of the certificate format

– **Serial number**: An integer value unique within the issuing CA that is unambiguously associated with this certificate.

– **Signature algorithm identifier**:
The algorithm used to sign the certificate together with any associated parameters. Because this information is repeated in the signature field at the end of the certificate, this field has little, if any, utility.

– **Issuer name**: X.500 name of the CA that created and signed this certificate.

– **Period of validity**: Consists of the first and last date on which the certificate is valid.

– **Subject name**:
The name of the user to whom this certificate refers.

– **Subject's public-key information**: The public key of the subject, plus an identifier of the algorithm for which this key is to be used, together with any associated parameters.

– **Issuer unique identifier**: An optional-bit string field used to identify uniquely the issuing CA in the event the X.500 name has been reused for different entities.

– **Subject unique identifier**: An optional-bit string field used to identify uniquely the subject in the event the X.500 name has been reused for different entities.

– **Extensions**: A set of one or more extension fields

– **Signature**: Covers all the other fields of the certificate.

5. X.509 Certificates and PKI

4. Obtaining a User's Certificate

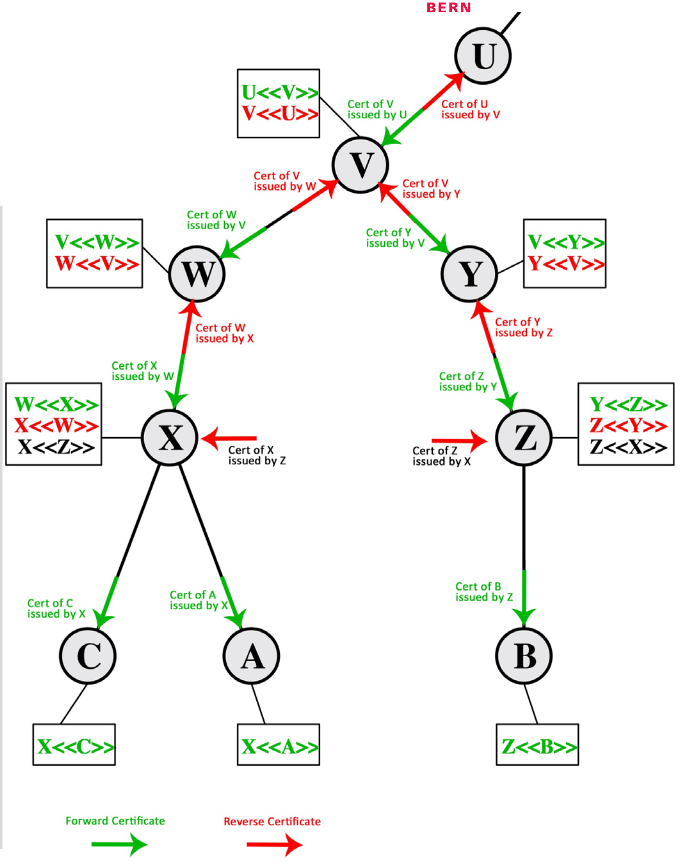
Characteristics of a user certificate generated by CA

- Any user with access to CA's public key can verify a user's public key.
- No party other than CA can modify a certificate without this being detected.
- If all users subscribe to one CA, there is a common trust.
- In case of large communities, users might not subscribe to the same CA.
- Chains of certificates can be used to obtain other users' keys.
- Example
  - A has certificate from X_1, B from X_2.
  - Assumption: X_1 and X_2 exchanged certificates.
  - A can obtain X_2's certificate signed by X_1.
  - A has then a trusted copy of X_2's certificate, A can verify B's certificate.
  - Certificate chain: X_1<<X_2>> X_2<<B>>

5. X.509 Certificates and PKI

5. X.509 Hierarchy

- Connected circles indicate hierarchical relationship among CAs.
- Associated boxes indicate certificates maintained in the directory for each CA entry.
- Directory entry for each CA includes 2 types of certificates
  - Forward certificates: Certificates of X generated by other CAs
  - Reverse certificates: Certificates generated by X that are certificates of other CAs
- Example: A can establish certification path to B
  - X<<W>> W<<V>> V<<Y>> Y<<Z>> Z<<B>>



UNIVERSITÄT BERN

5. X.509 Certificates and PKI

6. Certificate Revocation

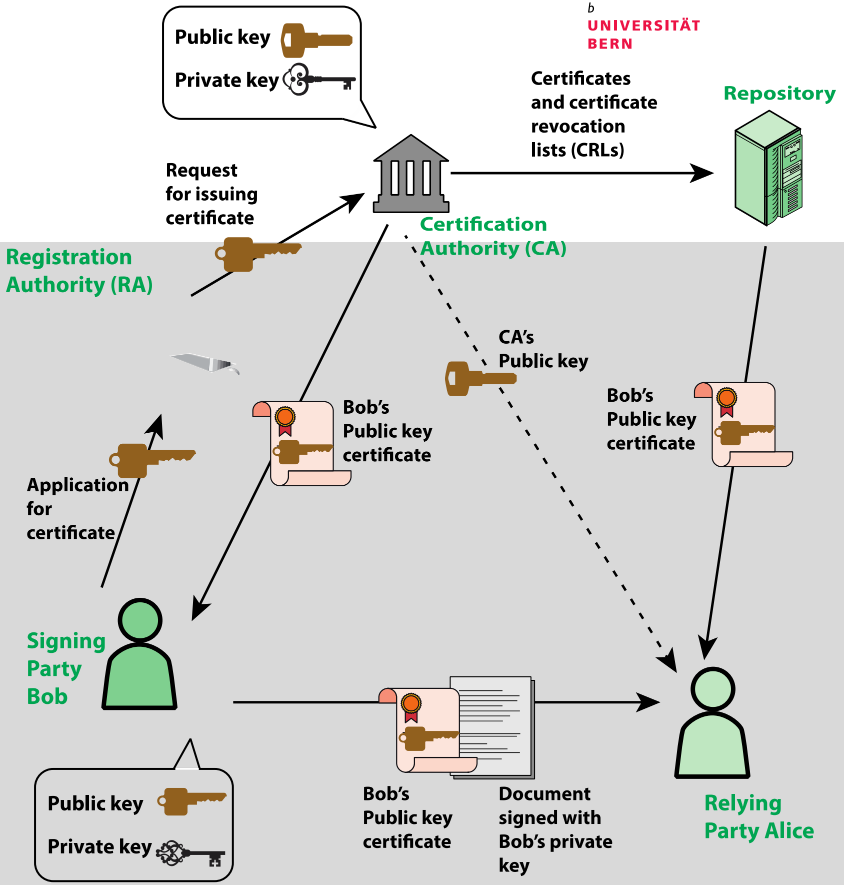
- Each certificate includes a period of validity and typically a new certificate is issued just before the expiration of the old one.
- It may be desirable on occasion to revoke a certificate before it expires, for one of the following reasons:
  - The user’s private key is assumed to be compromised.
  - The user is no longer certified by this CA.
  - The CA’s certificate is assumed to be compromised.
- Each CA must maintain a Certificate Revocation List consisting of all revoked but not expired certificates issued by that CA
- These lists should be posted on the directory.

5. X.509 Certificates and PKI

7. Public Key Infrastructure

Components

- End entity: users, devices
- Certification Authority: creation and signing public keys
- Registration authority: optional component to offload CA functions
- Repository: methods to store and retrieve PKI-related information
- Relying party: user or agent relying on certificate data in making decisions.



Public key
Private key
Certificates and certificate revocation lists (CRLs)
Repository
Request for issuing key certificate
Certification Authority (CA)
Registration Authority (RA)
CA's Public key
Bob's Public key certificate
Bob's Public key certificate
Application for certificate
Signing Party Bob
Public key
Private key
Bob's Public key certificate
Document signed with Bob's private key
Relying Party Alice

Thanks

for Your Attention

Prof. Dr. Torsten Braun, Institut für Informatik

Bern, 28.03.2022 – 04.04.2022