7 Cryptographic Techniques – A Brief Introduction

7.1 Introduction to Cryptography

7.2 Symmetric Encryption

7.3 Asymmetric (Public-Key) Encryption

7.4 Digital Signatures

7.5 Public Key Infrastructures

Literature:

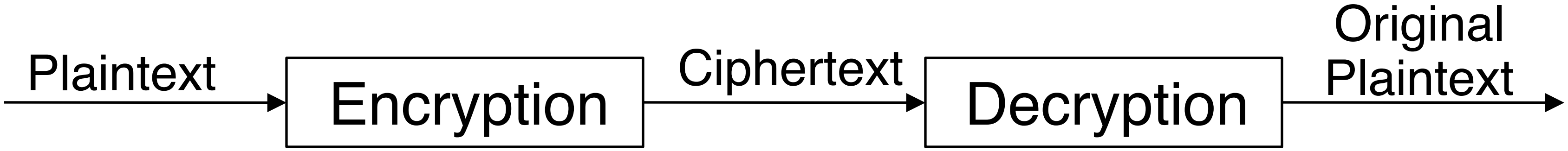
Bruce Schneier: Applied Cryptography, 2nd ed., John Wiley 1995

Klaus Schmeh: Krypografie, 5. Auflage, dpunkt 2013

Purpose of Cryptographic Techniques

- To protect the content of communication between two parties
  - Protection against various kinds of attacks
  - Preserving confidentiality and integrity of a message
  - Computer-equivalent to packaging and sealing
- To establish the identity of communication partners (*authentication*)
  - Computer-equivalent to hand-written signature
  - *Nonrepudiation (Zurechenbarkeit):*
    Avoiding false denial of the fact that someone has sent a message
- Applications for networked multimedia:
  - Encrypted content in DRM, decryption only for authorized users
  - Packaging keys and right specifications in DRM
  - Identifying business partners for payment procedures
  - Protecting electronic forms of money
  - Protecting important personal data

Encryption and Decryption



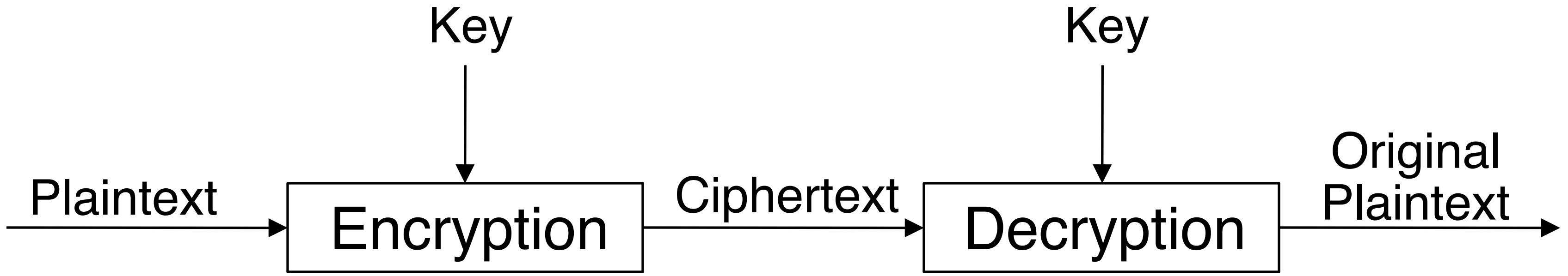
- A *sender* (often called Alice) wants to send a *message* to a *receiver* (often called *Bob*), in a way that an eavesdropper (often called *Eve*) cannot read the message.
  - Plaintext message (binary data) M
  - Ciphertext C
- Encryption E:

  $E(M) = C$
- Decryption D:

  $D(C) = M$

  such that $D(E(M)) = M$
- Encryption/Decryption should not rely on keeping the algorithms secret.
  - Kerckhoffs principle

Encryption and Decryption Keys



- Encryption *E:*

  $E(K_1, M) = C$

- Decryption *D:*

  $D(K_2, C) = M$

  such that $D(K_2, E(K_1, M)) = M$

- Special case:
  Identical keys for encryption and decryption *(symmetry)*

Attack Terminology

- Ciphertext-only attack
  - Recover the plaintext or the keys based only on the ciphertext
- Known-plaintext attack:
  - Deduce the keys from given plaintext and corresponding ciphertext
- Chosen-plaintext attack:
  - Attacker (cryptanalyst) can obtain the encoding result on an arbitrary plaintext
- Chosen-ciphertext attack:
  - Attacker (cryptanalyst) can obtain the decoding result on an arbitrary ciphertext

- Brute-force attack
  - Trying out all possible keys
  - Breakability depends on available computing power

7 Cryptographic Techniques – A Brief Introduction

7.1 Introduction to Cryptography

7.2 Symmetric Encryption

7.3 Asymmetric (Public-Key) Encryption

7.4 Digital Signatures

7.5 Public Key Infrastructures

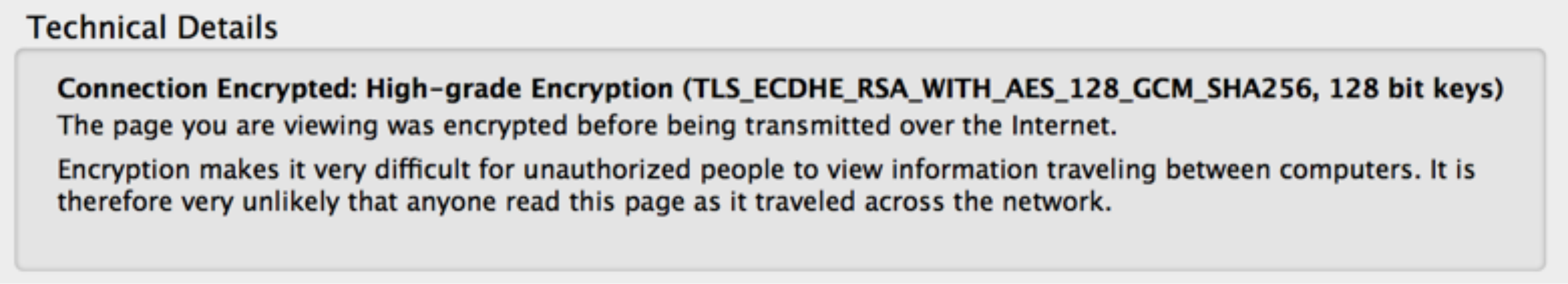
Literature:

Bruce Schneier: Applied Cryptography, 2nd ed., John Wiley 1995

Klaus Schmeh: Krypografie, 5. Auflage, dpunkt 2013

Symmetric Cryptographic Algorithms

- Encryption and decryption using the same key
  - Alternatively: One key can be computed from the other
- Stream algorithms or *stream ciphers:*
  - Operate bit-by-bit (or byte-by-byte)
- Block algorithms or *block ciphers:*
  - Operate on larger groups of bits (blocks)
  - Block size should not be too large - typical 64 bits



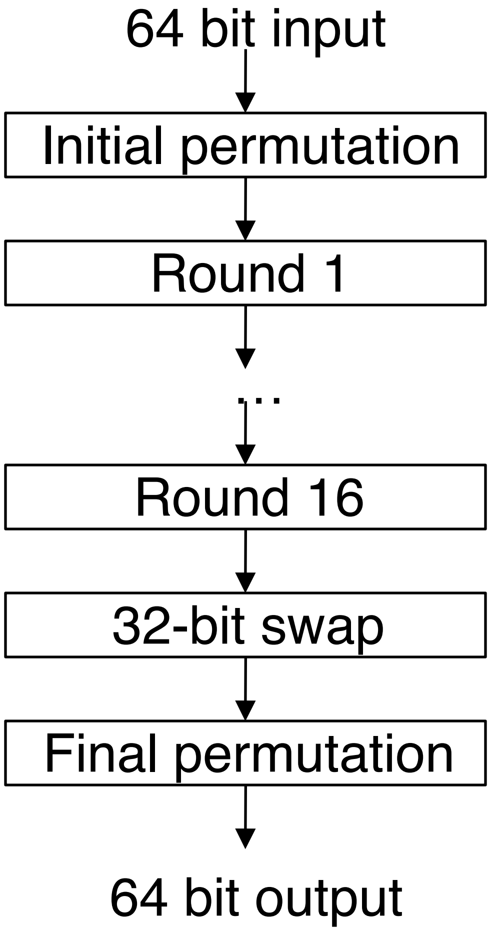
Technical Details

Connection Encrypted: High-grade Encryption (TLS_ECDHE_RSA_WITH_AES_128_GCM_SHA256, 128 bit keys)
The page you are viewing was encrypted before being transmitted over the Internet.

Encryption makes it very difficult for unauthorized people to view information traveling between computers. It is therefore very unlikely that anyone read this page as it traveled across the network.

Firefox encryption details message

Data Encryption Standard DES


64 bit input

Initial permutation

Round 1

...

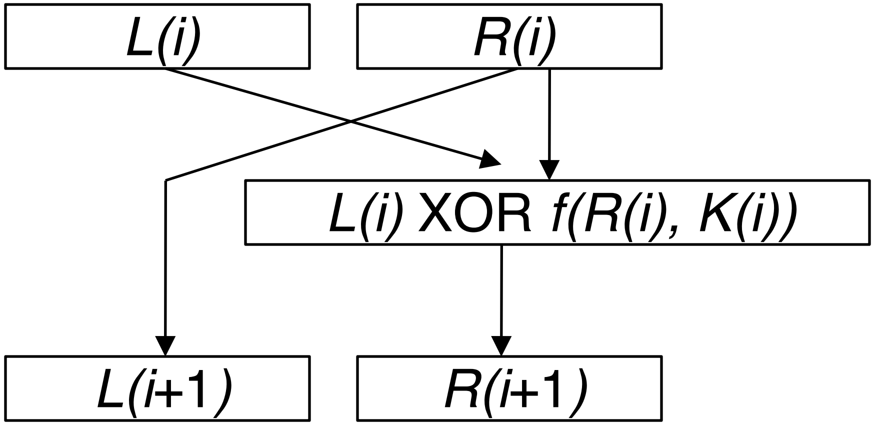
Round 16

32-bit swap

Final permutation

64 bit output

- Symmetric block cipher (64 bit blocks)
- Adopted by U.S. government in 1977, based on IBMs *Lucifer* algorithm
  - Designed for hardware realization
- Key length: 56 bits
- Each of the 16 “rounds”:


L(i)

R(i)

L(i) XOR *f(R(i), K(i))*

L(i+1*)*

R(i+1*)*

- *f* does a number of permutations and substitutions

Encoding and decoding algorithms identical

DES – Example for an Aging Standard

- Brute force attack to DES: 2^{56} permutations to be tried
  - 56 bit keys considered unbreakable in 1977
- Specialized hardware can test DES keys very fast
  - Rumours persist that the NSA (US National Security Agency) can break 56-bit DES in a few minutes time
  - 1997: DES Challenge
    - After 4 months, a DES-encrypted message could be decrypted
  - 2000: DES Challenge III won by “distributed.net” in 22 hours
    - Specialized supercomputer + CPU time from 100.000 PCs in the Net
    - Key test rate 240 billion keys/second
- Practical workaround: “Triple DES”
- Obstacle for unbreakable codes:
  - U.S. government apparently wants to be able to break the standard encryptions
- Strong cryptographic products are considered weapon technology by the U.S. government!

Advanced Encryption Standard AES

- U.S. National Institute of Standards and Technology (NIST)
  - 1997: Call for proposals for an unclassified, publicly disclosed symmetric encryption algorithm, key sizes 128, 192, and 256 bits
  - 15 submissions, 5 candidates selected
    (MARS, RC6, Rijndael, Serpent, Twofish)
  - 2000: Rijndael declared to be official AES
- Rijndael (Joan Daelen, Vincent Rijmen, Belgium):
  - Between 10 and 14 rounds, depending on key and block length
  - Operations in each round:
    - » XOR
    - » Byte substitution
    - » Row shift (in a grid representation)
    - » Mixing of columns based on polynomial (in a grid representation)
- Other common alternative symmetric algorithms: RC4, RC6 ((Ronald) Rivest Cipher)

7 Cryptographic Techniques – A Brief Introduction

7.1 Introduction to Cryptography

7.2 Symmetric Encryption

7.3 Asymmetric (Public-Key) Encryption

7.4 Digital Signatures

7.5 Public Key Infrastructures

Literature:

Bruce Schneier: Applied Cryptography, 2nd ed., John Wiley 1995

Klaus Schmeh: Krypografie, 5. Auflage, dpunkt 2013

Asymmetric or Public Key Encryption

- Main problem of symmetric cryptography:
  How to obtain the shared, secret key?
  - Off-line transportation
  - Key distribution architectures, e.g. Kerberos

- Public-key cryptography: Whitfield Diffie, Martin Hellman 1976
  - Each person gets a *private* (secret) key and a *public* key
- Public-Key Cryptosystem:

  Encryption with public key: $PK(M) = C$

  Decryption with secret key: $SK(C) = M$

  such that $SK(PK(M)) = M$
  - By publicly revealing PK, user does not reveal an easy way to compute SK.

- Mathematical background: “Trapdoor one-way function”
  - e.g. prime factorization of large numbers

Example Public Key

```
-----BEGIN PGP PUBLIC KEY BLOCK-----
Version: GnuPG v1.4.1 (Darwin)

mQGiBEIjHZ4RBADcsLcc9Cx1ZCfkcuI3GJBqiKbYPtsgD71hGorg9Q13ZWXLrgVt
0thGybaC0ftSYoAlqPmsWqBGAPXtNYB8igHaqq+qfbO99HVK5lQTptR58zbqPZ7S
wpvZdmx7/TujVKc9PwSWnCfjlUwh0HqY7AhSR86cCFocfkyfN4eafOCjqwCg+bb8
Ry/JlqbASmXz/bzOidfexVUD/3qxIzTse//r0p28bKfeyVFcleov5Z1diMTwNywm
K+56wqZLeE70va3g55t1Haed2t9OJ6rC6Hlik1tMWl1kfiW85NSj0Lb1Yz8LMs8C
k7FgswBbBrYJ7WjdMex1vpBjVKQ1geZGn9uph8/IdwtEdLBm9n49ADRXayDYR6W5
VZ12A/95DKFTOgE6VNiriEJ3c+LHGE8CQ0J8OL1UYyWL+iZdXvvUyKmu0JMSjxfA
J+05FR3BYpOKWtrgEZT/djPEjy4zsWi1vKZ7jgq7yb1dltPfy5rMNUnyk9PzGq78
6DQRzcb/GbLy+ocZyldhympfBIFAjMFQoIgZiYuFaPFhvmHRprQnSGVpbnJpY2gg
SHVzc21hbm4gPGh1c3NtYW5uQGlmaS5sbXUuZGU+iF4EExECAB4FAkIjHZ4CGwMG
CwkIBwMCAxUCAwMWAgECHgECF4AACgkQ4NBNTLlLJwWhewCfQmn5tkFKbggyyT2G
l4vAXurMqFwAn16AVV0VzJT64tsP1Pq36KAbqmYmuQENBEIjHaEQBACHR51/R2jf
QU128JBC6tlQaaREmqk6Zp49puu60tieSj3F6h+hrjn37sA0eltIuQJqjGJQjn90
RKEBoSHUliKITF+9v2n1t8UMf7VNQIbTvzO9X1ac/SqWYewllXlvRiFfTPwOhgD8
BEQ/yoiwbt6yCWbF8jNoDTipECgxFuyQfwADBQP/fNo9Hhl87uek8nTidbL0Di0D
+hEzeoA9rP/7rT+Y5330XjV+Sph2sXFlqc95Tj7VnJeI8lVJiWtu8Tdu3PM/X/xV
s+0y/nSqc9HGlYDzNEMGQHPdEMb1VDDP60/flKkQkrPvtiqG12NRBSHLOW/nyfPe
LHlH11ZALHtcLhYfB2uISQQYEQIACQUCQiMdoQIbDAAKCRDg0E1MuUsnBc3JAJ93
PayXxyXeBxtb40dxy32DHHtTVgCgrD/Kyc8D14xcbo6Sn18VXKCAqm0=
=klXH
-----END PGP PUBLIC KEY BLOCK-----
```

RSA: Mathematics

- Ronald Rivest, Adi Shamir, Leonard Adleman 1978 (MIT)
- Creating a public/secret key pair:
  - Choose two large primes p and q and compute the “modulus” $n = pq$
  - Randomly choose a number $e < n$, relatively prime to $\phi = (p–1)(q–1)$ (Eulers totient function)
    - » (n, e) is the public encryption key
  - Compute d as inverse of e (modulo ϕ): i.e. such that $(ed \equiv 1) \bmod \phi$
    - » (n, d) is the secret decryption key
- Encryption:

$$C = M^{e} \bmod n$$

- Decryption:

$$M = C^{d} \bmod n$$

For an example, see e.g. http://www.di-mgt.com.au/rsa_alg.html

RSA: Mathematics – Example

- Creating a public/secret key pair:
  - Choose two (large) primes *p* and *q* and compute the “modulus” $n = pq$
    - » *p = 11, q = 13, n = 143 (in practice much larger!)*
  - Randomly choose a number $e < n$, relatively prime to $\phi = (p-1)(q-1) = 120$
    - » E.g. *e = 23 (in practice, Fermat primes are used, e.g. 3, 17, 65537)*
    - » (*143*, *23*) is the public encryption key
  - Compute *d* such that $(ed \equiv 1) \bmod \phi$, i.e. $(ed-1) = k\,\phi$, i.e. $(23\,d-1) = k\;120$
    - » Apply extended Euclidian algorithm: *d = 47, k = 9*
    - » (*143*, *47*) is the secret decryption key
- Encryption:

  $C = M^{e} \bmod n$, *e.g.* $C = 7^{23} \bmod 143 = 2$ *(Modular arithmetic)*

- Decryption:

  $M = C^{d} \bmod n$, *e.g.* $M = 2^{47} \bmod 143 = 7$

RSA: Pragmatics

- Key size is variable, typical 1024 bits
- RSA relies on exponentiation which is computing-intensive
  - DES is at least 100 times as fast as RSA in software and 1000 to 10000 times as fast in hardware
- Security of RSA is conjectured to rely on factorization of large numbers into primes
- Hybrid usage of symmetric and asymmetric cryptosystems *(enveloping)*
  - Choose a symmetric key (e.g. for AES)
  - Encode the symmetric key with an asymmetric cryptosystem (e.g. RSA) to transmit the shared (symmetric) key to the communication partner
  - Combination of advantages:
    - » Use asymmetric system for keeping the secrets locally
    - » Use symmetric system for mass-data encoding
- RSA is part of many Internet protocols for secure interaction, e.g. S/MIME, SSL, TLS, IPsec, ...

7 Cryptographic Techniques – A Brief Introduction

7.1 Introduction to Cryptography

7.2 Symmetric Encryption

7.3 Asymmetric (Public-Key) Encryption

7.4 Digital Signatures

7.5 Public Key Infrastructures

Literature:

Bruce Schneier: Applied Cryptography, 2nd ed., John Wiley 1995

Klaus Schmeh: Krypografie, 5. Auflage, dpunkt 2013

Digital Signature with Asymmetric Cryptosystems

- Message authentication (digital signature):
  - To establish trust that a message actually originates from a certain sender
  - Must involve full message body, i.e. similar to message encryption
- Some asymmetric cryptosystems allow to use “inverse encryption” for a digital signature, e.g. RSA
  - For such cryptosystems, the inverse equation holds: $PK(SK(M)) = M$
  - Encryption with own secret key
  - Verification possible by anybody knowing the public key
- Example: Alice wants to send a message M to Bob ensuring the message’s integrity and that it is from her

  $S = M^{d} \bmod n$ (n, d) is Alice’s secret key – Equivalent to decryption algorithm

  - Alice sends M and S to Bob
- Bob verifies:

  $M = S^{e} \bmod n$ (n, e) is Alice’s public key – Equivalent to encryption algorithm
- Other digital signature standards exist, e.g. DSS/DSA (Digital Signature Standard/Algorithm by NIST)

Message Digesting or Hashing

- Sometimes not encryption, but integrity of message is the goal
  - Simpler algorithms similar to symmetric encryption
- Hash (or *digesting*) function for messages
  - Computes short code from long message
  - Difficult to invert (i.e. to obtain message from code)
  - Collision-resistant (i.e. unlikely to find two messages with same hash code)

- Examples of message digesting algorithms:
  - MD5 (Ron Rivest) (128 bit code)
  - Secure Hash Algorithm SHA (NIST) (160 bit code)

- Combination of message digest and signing the digest:
  - Faster way of authenticating a message

7 Cryptographic Techniques – A Brief Introduction

7.1 Introduction to Cryptography

7.2 Symmetric Encryption

7.3 Asymmetric (Public-Key) Encryption

7.4 Digital Signatures

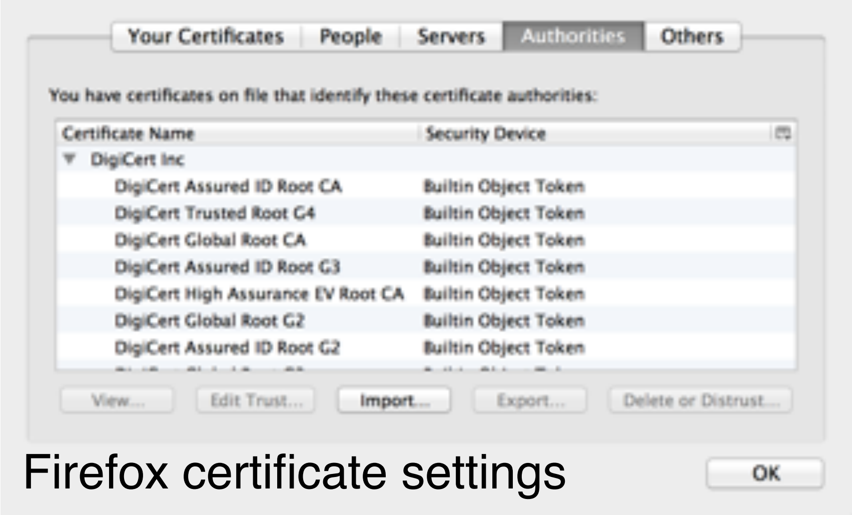
7.5 Public Key Infrastructures

Literature:

Bruce Schneier: Applied Cryptography, 2nd ed., John Wiley 1995

Klaus Schmeh: Krypografie, 5. Auflage, dpunkt 2013

Public Key Infrastructure



You have certificates on file that identify these certificate authorities:

Certificate Name	Security Device
▼ DigiCert Inc	
DigiCert Assured ID Root CA	Builtin Object Token
DigiCert Trusted Root G4	Builtin Object Token
DigiCert Global Root CA	Builtin Object Token
DigiCert Assured ID Root G3	Builtin Object Token
DigiCert High Assurance EV Root CA	Builtin Object Token
DigiCert Global Root G2	Builtin Object Token
DigiCert Assured ID Root G2	Builtin Object Token

Firefox certificate settings

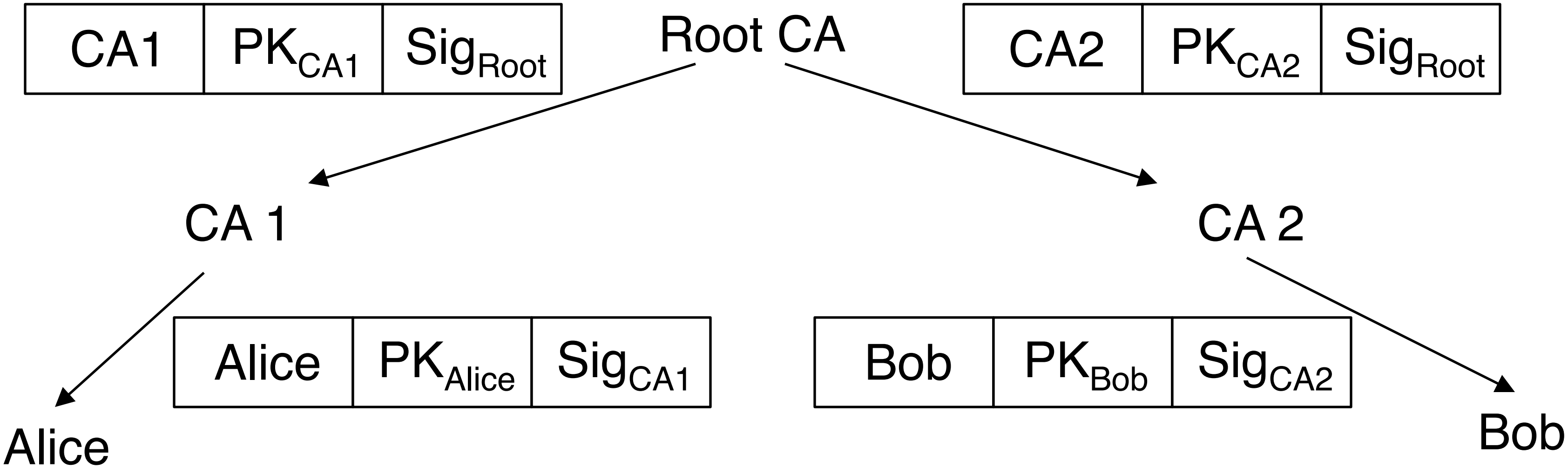
- Weak point in public-key cryptosystems
  - Association public key / real identity
- Establishing trust in public keys:
  - Trusted Third Party (TTP)
    - » e.g. governmental organization, financial institution
  - TTP issues a message *(certificate)* that contains
    - » User identity
    - » Public key
    - » Validity period
    - » Issuer (TTP identity)
  - TTP “signs” certificate
    - » This can be achieved by using the own public key
    - » All participants know the signatures (public keys) of TTP, i.e. can trust that the certificates actually come from the issuing TTP

Certificate

Identity	PK	Signature

Certification Authorities

- A TTP issuing certificates is a *Certification Authority* (CA)
- CAs are organized in a hierarchy, signature of root CA universally known



CA1 | PK_{CA1} | Sig_{Root}

Root CA

CA2 | PK_{CA2} | Sig_{Root}

CA 1

CA 2

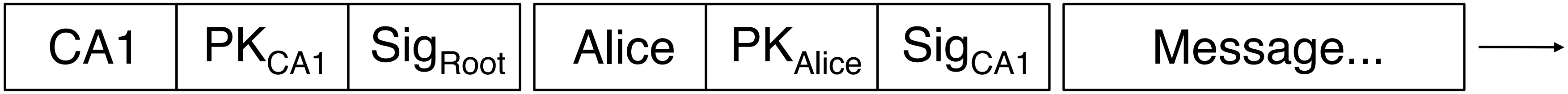
Alice | PK_{Alice} | Sig_{CA1}

Bob | PK_{Bob} | Sig_{CA2}

Alice

Bob

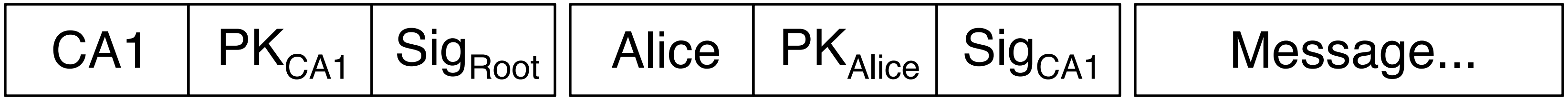
The certificates for the public key can be transferred with the message (or put on a website etc.)
E.g. message from Alice to Bob:



CA1 | PK_{CA1} | Sig_{Root} | Alice | PK_{Alice} | Sig_{CA1} | Message...

Digital Signatures and PKI

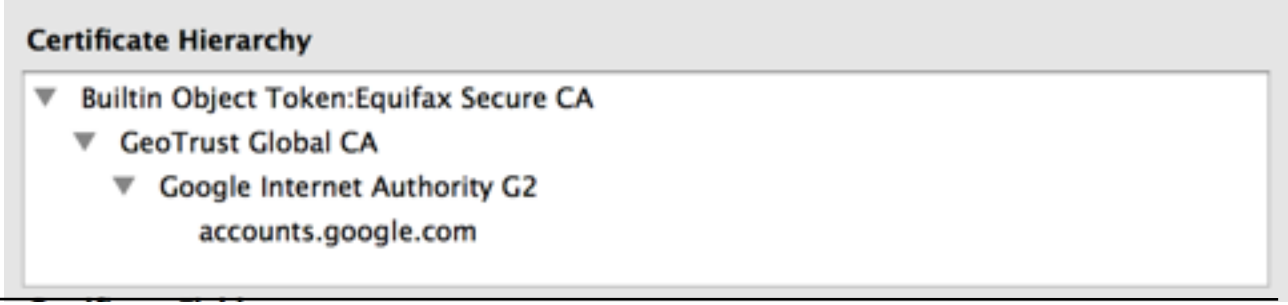
- The “chain of trust” in a PKI can be reduced to the single fact
  - Everybody knows the public key PK_{Root} of the Root CA
- Root CA signs CAx certificates using its secret key SK_{Root}
  - Everybody can verify the certificates using PK_{Root}
- CAx signs certificates using its secret key SK_{CAx}
  - Everybody can verify the certificate as soon as he has PK_{CAx}
  - ... which he can obtain from a Root-signed certificate



CA1 | PK_{CA1} | Sig_{Root}

Alice | PK_{Alice} | Sig_{CA1}

Message...



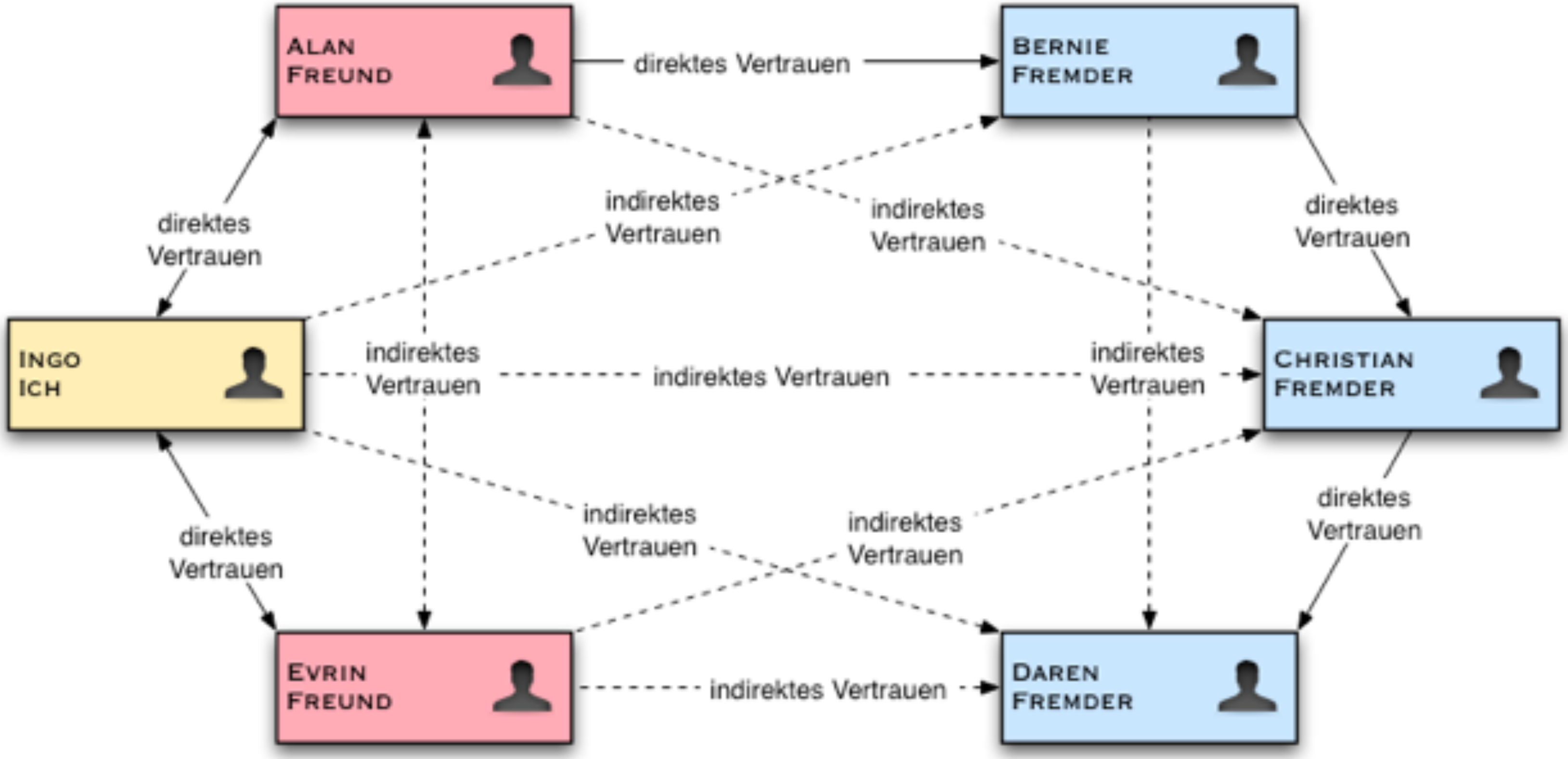
Certificate Hierarchy

- Builtin Object Token:Equifax Secure CA
  - GeoTrust Global CA
    - Google Internet Authority G2
      - accounts.google.com

Google certificate chain shown by Firefox

Web of Trust

- No central certification authority; mutual certification
- Users can define individual level of trust in the owner of a key
- Well-known implementations: PGP and GPG



Picture: Wikipedia