TCP/IP – part 4
Security. Cryptographic protection.

Last modification date: 15.04.2020

Introduction

- E-commerce, e-office, and traditional communication confidentiality needs are increasing.

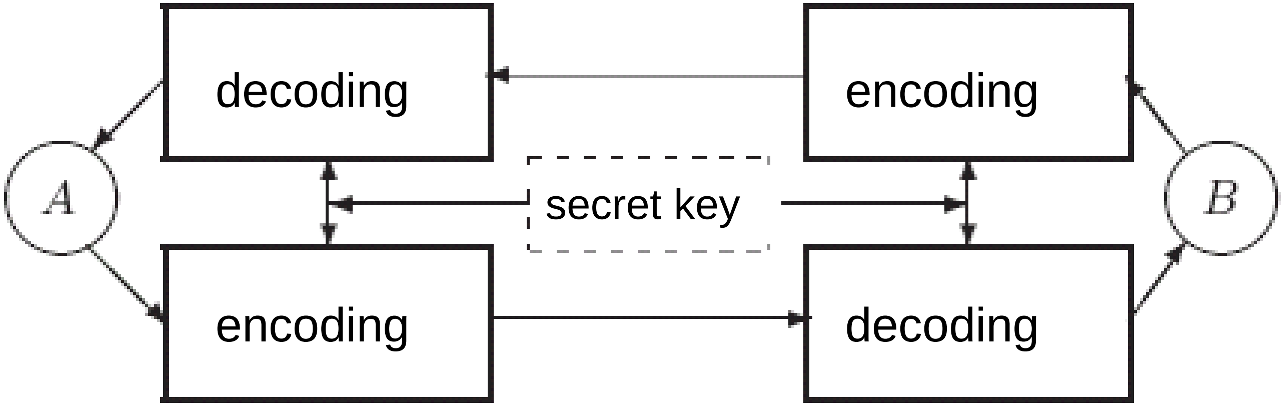
- Fundamental tasks of cryptographic protection of network applications:
  - Message origin authentication, message target authentication
  - Data confidentiality, confidentiality of content
  - Data integrity
    - Message content integrity
    - Message sequence integrity
  - Non-repudiation of message origin, non-repudiation of message receipt
  - Access control

Elementary cryptographic techniques

1. **Encryption/decryption**

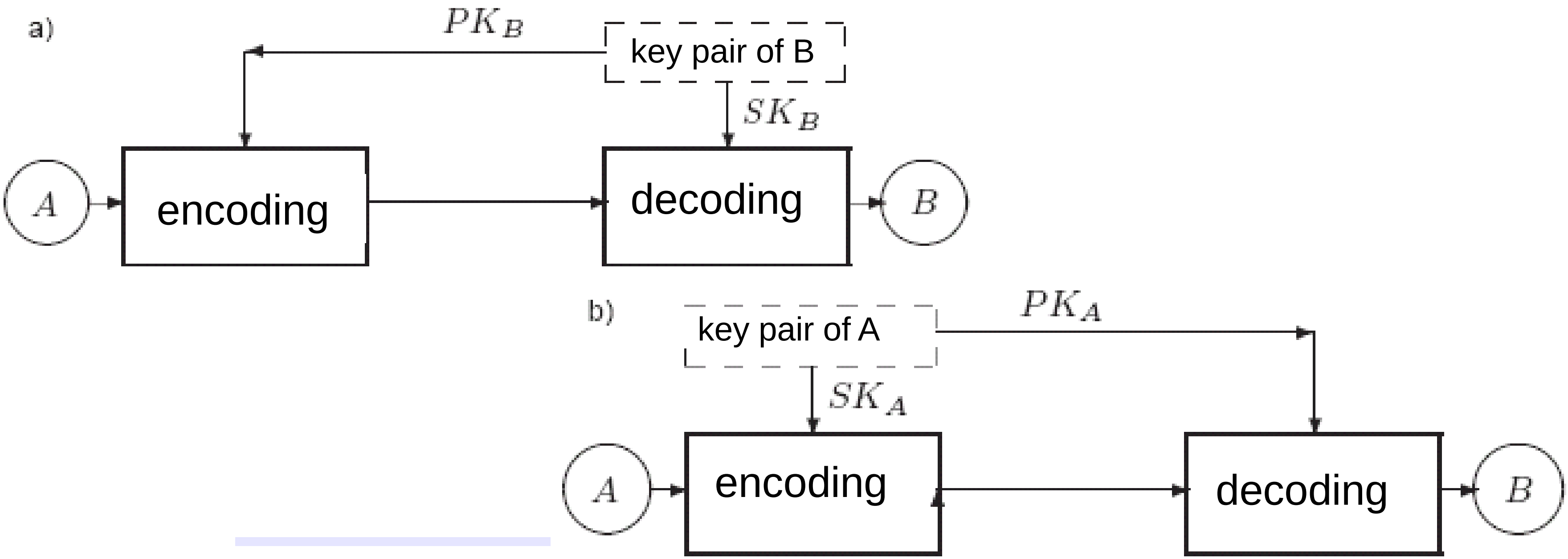
   a) **Symmetric cryptography** (classic), e.g.: DES, 3DES (TDES), IDEA, CAST, Blowfish, Twofish, SAFER, AES:

   - The same key (secret(is used to encrypt and decrypt a message
   - Key distribution – via a secure channel (e.g. by a courier)
   - the number of keys for N user pairs is: `N*(N-1)/2`
   - Relatively fast encryption/decryption



Elementary cryptographic techniques – cont.

1. Encryption/decryption
   b. **Asymmetric cryptography** (public key cryptography), e.g.. RSA, Diffie-Hellman, ElGamal, DSA/DSS
      - Key pair: public (PK) and private/secure (PK) define forward and inverse transformations
      - Two operation modes: a) confidentiality - a message that a sender encrypts using the recipient's public key can be decrypted only by the recipient's paired private key; b) non-repudiation of sending message



Elementary cryptographic techniques – cont.

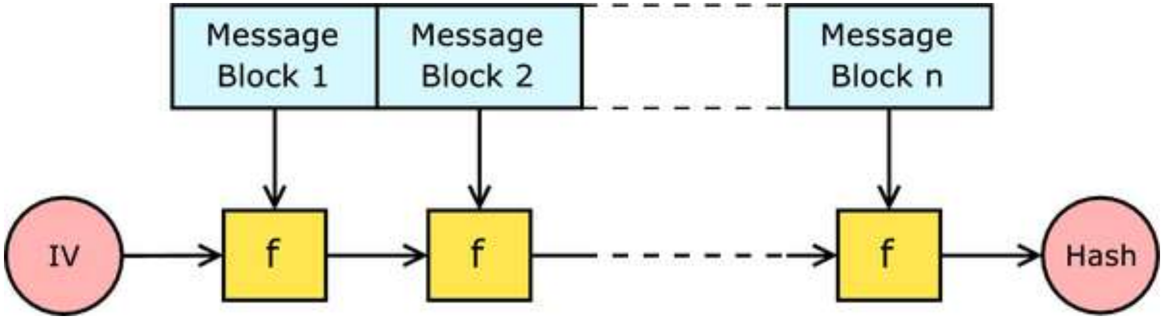
2. Computation of message digest (message integrity code) – is used for control of data integrity and for authentication..

   Properties of cryptographic hash function:

   - The function does not have any secret parameters and can be calculated for a message `M` of arbitrary length `D=h(M)`
   - It is not computationally possible to get the message `M` given `D`
   - Knowing the function `h` it is not possible to determine M_1, $M_2 != M_1$, such that $h(M_1) == h(M_2)$

   Hash value should be at least 2 times longer than the length of the key used for symmetric cryptography (for given application).

   Popular hash algorithms: SHA1 (160b – no longer recommended), SHA2, SHA3 (224-512b), RIPEMD160, HAVAL (128-256b), MD5 (128b – not recommended).



Merkle-Damgard technique of hash calculation for a message of arbitrary length

Elementary cryptographic techniques – cont.

3. **Generation of cryptographically secure random numbers ("true random numbers")** – is very important for many cryptographic algorithms (protocols), e.g. they impede reply attack (see RFC4086 „Randomness Requirements for Security").

   Software generators (pseudo-random number generators, PRNGs) have finite period.

   Basic requirements to cryptographically secure RNG:

   - There exists no algorithm of polynomial complexity which can infer from `I` bits of the generator its next bit – with probability which is significantly larger than 1/2.
   - If state of a generator is guessed at some point, than it is not possible to obtain previously generated bits.

   Example generators:

   - `/dev/random` – uses random system events; can block when insufficient entropy of registered events.
   - `/dev/urandom` – pseudo-random number generator, never low on numbers.
   - Daemon EGD (or prng) – pseudo-random number generators In user space; use randomness of events in a loaded systems (observations with such tools as 'w', 'last', 'vmstat')
   - Algorithm `Yarrow` (FreeBSD /dev/random) and algorithm `Fortuna` – use symmetric cryptography to generate pseudo-random numbers, accumulator of system events and a file with random numbers (for fast start after system re-init.)
   - `Mersenne Twister` – the most widely used general-purpose PRNG. For 32b implementation its period can be $2^{19937} - 1$

Cryptographic services - confidentiality

- Ciphers:
  - Stream ciphers; if KS – the key stream and M – message, then encoding: $C[i]=KS[i] \oplus M[i]$ and decoding $M[i]=KS[i] \oplus C[i]$ where $\oplus$ is most frequently XOR. Example stream cipher: RC4
  - Block ciphers (block size: 8-16 bytes). Example: AES, 3DES, RC2. Modes:
    - Electronic codebook (ECB); the message M is split into blocks $M[i]$; encoding: $C[i]=E(K,M[i])$; decoding: $M[i]=D(K,C[i])$, where E and D define encoding and decoding operations, K is the key.

      Problem: same input => same output
    - Cipher block chaining (CBC):

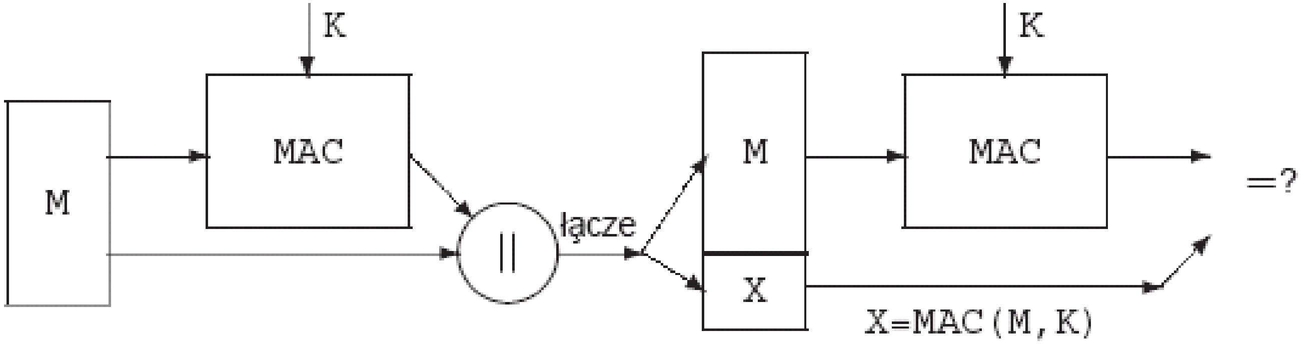
      $C[i]=E(K,M[i] \oplus C[i-1])$ where $C[0]=IV$ (random init. vector)

      $M[i]=D(K,C[i]) \oplus C[i-1]$

      Note: for b bit blocks the message length should not exceed $2^{(b/2)}$ blocks (possible *rollover*)
    - other modes: cipher feedback (CFB), or output feedback (OFB)
- **Kerckhoffs' principle**: A cryptosystem should be secure even if everything is known about the system, except the key. Common mistake about security: *security through obscurity*.
- It is believed that it is easier to provide secure distribution of asymmetric cryptography keys than keys for symmetric cryptography, but generation and encoding/decoding is faster for symmetric cryptography ciphers. It is frequently assumed that 1024b asymmetric cryptography key corresponds (in strength) to 80b key of symmetric cryptography.

Cryptographic services – data integrity

- Integrity of a single message is provided with a Message Integrity Code (Modification Detection Code, Message Authentication Code), e.g. a message hash value.



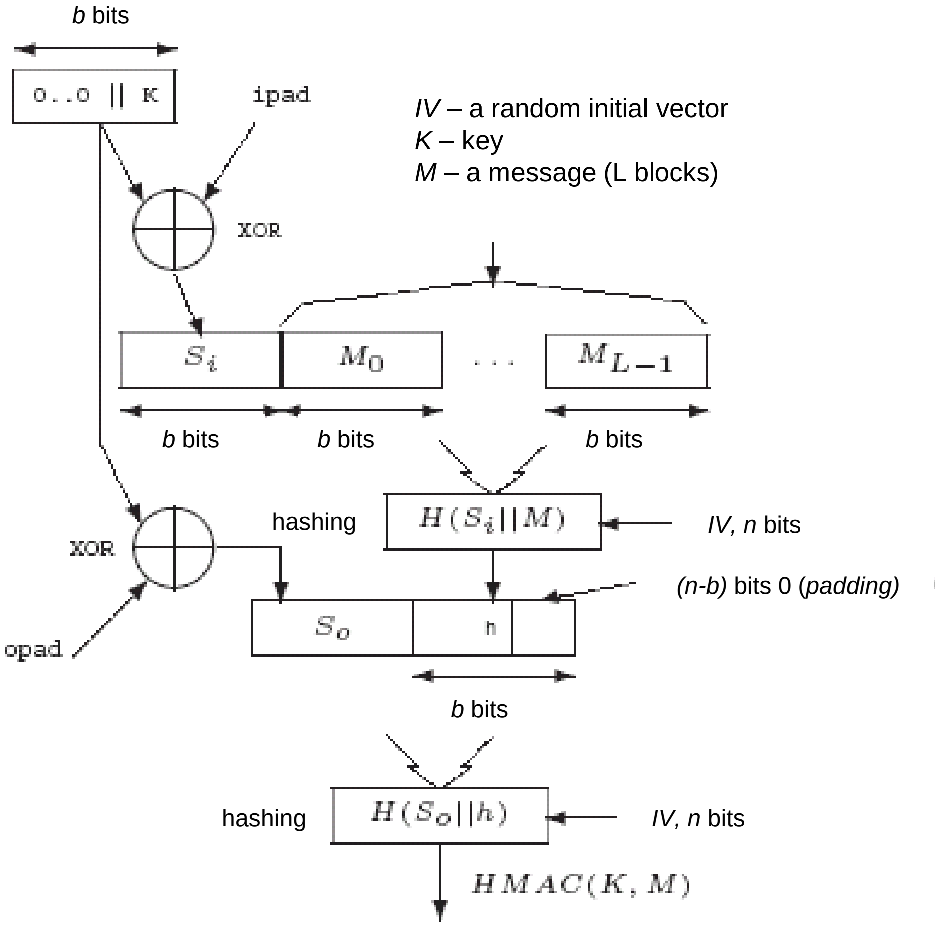
- Integrity of a sequence of message is provided by:
  - Protection of each message by MIC
  - Use of message sequence numbers, and also send time stamps
  - Confirmation of message reception

MAC calculation algorithm - HMAC

- **HMAC** (*Hashed Message Authentication Code*, RFC2104, FIPS-198)

HMAC(K,M)=H((K⊕ opad)||
H((K⊕ ipad)||M))

H(.) – a hash function



IV – a random initial vector
K – key
M – a message (L blocks)

- XOR with constants ipad (0x36 repeated b/8 times) and opad (0x5c repeated b/8 times). Typically b=512.
- IPSec and TLS use HMAC-SHA1 and HMAC-MD5

Cryptographic services – sender authentication

- Sender authentication enables confirmation of sender identity.

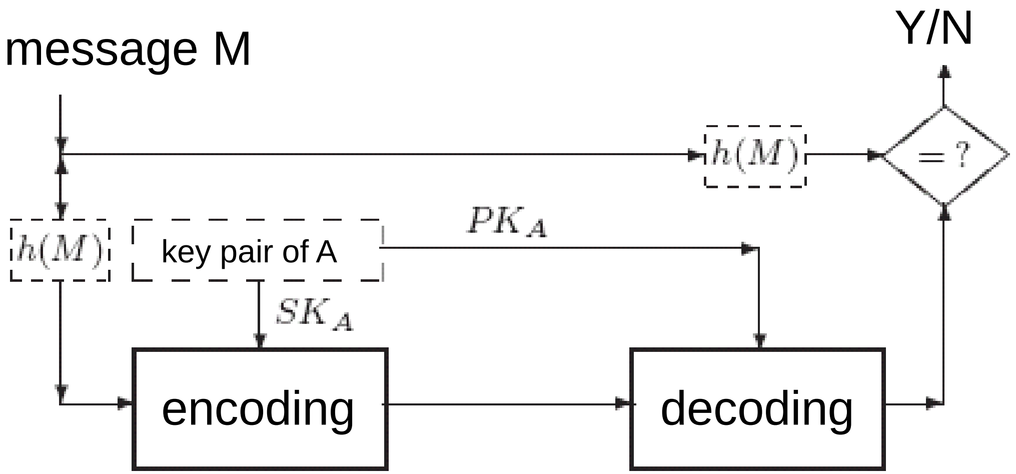
- Authentication can be implemented by appending a *message authentication code* (MAC) which depends on a secret key, which is known only to the sender and recipient..

- Authentication protocols which do not use a pre-defined secret are exposed to *Man-in-the-middle* (MITM) attacks.

Cryptographic services – sender non-repudiation

- Sender non-repudiation can be implemented with a digital signature which depends on the message and a secret, which is known only to the sender and recipient.

.



Note: distribution of public key (PK) has to guarantee validity and origin of the key.

In simple digital signature model it is possible to use the signature again and again (e.g. for recurrent banking operations) – so called reply attack.

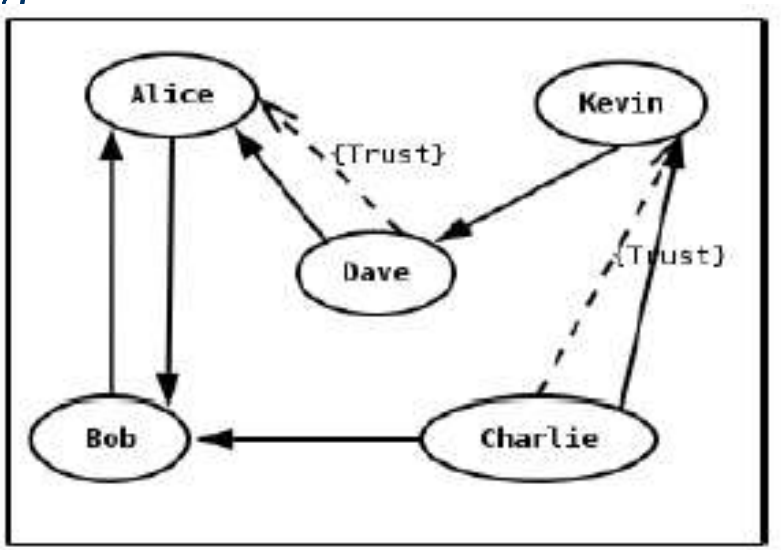
Solution: the message should be appended with a *sequence number*.

Cryptographic services – remarks

- Message integrity, authentication, confidentiality services – symmetric cryptography is better.
- Non-repudiation and key exchange – asymmetric cryptography is better.
- Because asymmetric ciphers are not efficient – usually symmetric-key encryption and public-key encryption are combined. The message is encrypted using a symmetric encryption algorithm, which requires a symmetric key. Each symmetric key (session key) is used only once. The message and its session key are sent to the receiver. The session key must be sent to the receiver so they know how to decrypt the message, but to protect it during transmission, it is encrypted with the receiver's public key. Only the private key belonging to the receiver can decrypt the session key.

Validation of public keys

1. **Distributed web of trust** , e.g. for PGP:
   - Each PGP user has his/her own database of public keys (*public key ring*)
   - A given public key (or more specifically, information binding a user name to a key) may be digitally signed by a third party user to attest to the association between someone (actually a user name) and the key.
   - Database keeps information on confidence level of a key :
     - unknown
     - untrusted
     - marginally trusted
     - completely trusted
   - A key can be added to the ring
     “manually” declaring the trust
   - The trust for a new key can be assigned automatically – if the key was signed by two “completely trusted users” or four “marginally trusted users”.



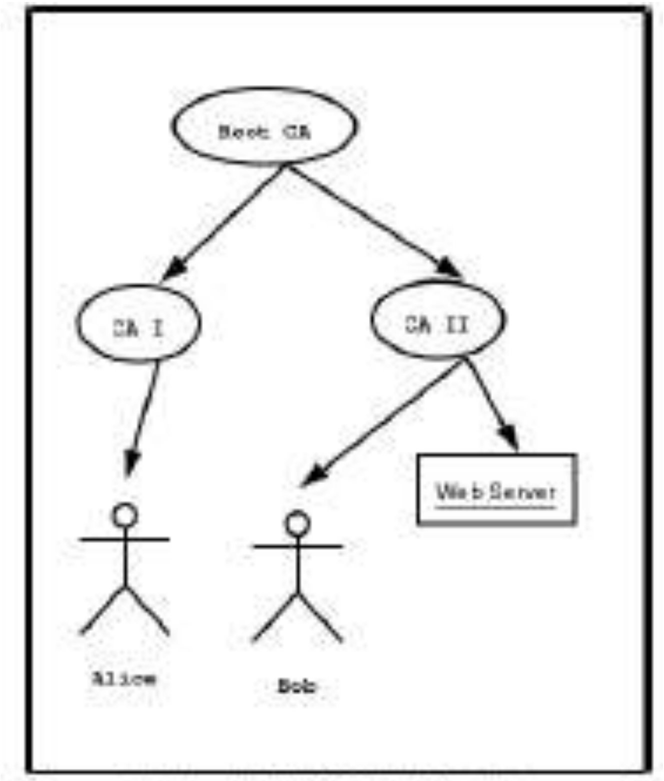
An example of the web of trust model

from N. Mavrogiannopoulos, S. Josefssoon „GNU TLS”

Validation of public keys

2. **Public Key Infrastructure (PKI)** handles certificates which are issued by hierarchical Certification Authorities (CA). CA perform the following activities:
   - Register users, which want to get authenticity certificates for their public keys.
   - Respond to questions regarding authenticity of relationship between a given key and a user.
   - Mange a Certification Revocation List (CRL), so that it is possible to invalidate certificates which are within their validity period.

   Each CA has its secret key (SK_{CA}) and a public key (PK_{CA}) – which can be known to each network user.. SK_{CA} is used to generate user certificates, which link (for a limited period of time) explicitly stated user identity, user key and digital signature of that message (can be verified with PK_{CA}).



Two typical X.509 Certification paths

An ITU-T standard, X.509 specifies formats for public key certificates, certificate revocation lists, attribute certificates, and a certification path validation algorithm.

From: N. Mavrogiannopoulos, S. Josefssoon „GNU TLS"

Remarks

- A CA can authenticate certificate of other CA; that way *certificate chains* are created

- Security of secret keys ($\mathtt{SK}_{\mathtt{CA}}$) and of public key ($\mathtt{PK}_{\mathtt{CA}}$) distribution for the CA at the top of CA hierarchy (which are self-signed) lend credence to the whole Public Key Infrastructure

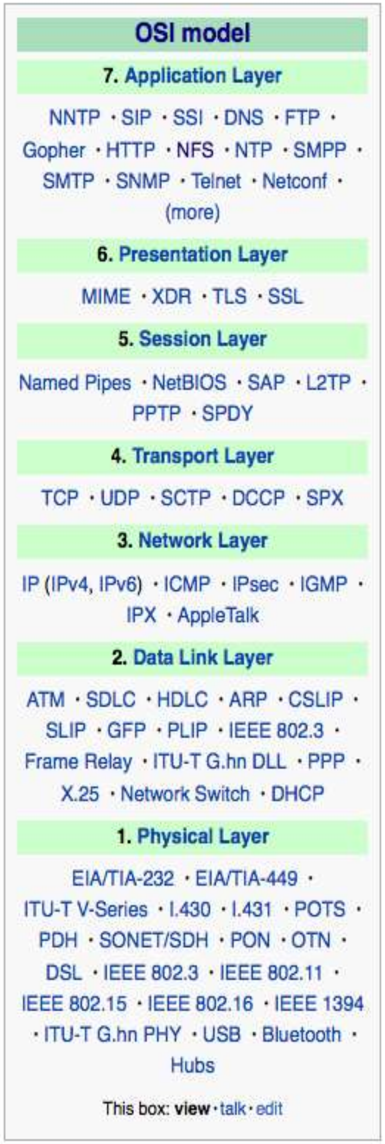
- Certificate users should check other user certificate validity with CA certificate, but also check validity period and CLR for each certificated involved.

Cryptography in network protocols

Cryptographic protection can be found in several layers of the OSI/ISO reference model

- Network layer: e.g. **IPSec + IKE** (key exchange).
  - The original IP packet is encoded, it gets a new IPsec header of IPsec and only than is forwarded..
  - A basis for **Virtual Private Networks (VPNs)**

    Two communication channels:
    - Key exchange channel – to manage authentication and key exchange
    - User data transmission channel.
- Presentation layer: e.g. **SSL/TLS.** Provides confidentiality, integrity, server (and possibly client) authentication. Uses asymmetric cryptography and X.509 certificates.



OSI model

7. Application Layer

NNTP · SIP · SSI · DNS · FTP · Gopher · HTTP · NFS · NTP · SMPP · SMTP · SNMP · Telnet · Netconf · (more)

6. Presentation Layer

MIME · XDR · TLS · SSL

5. Session Layer

Named Pipes · NetBIOS · SAP · L2TP · PPTP · SPDY

4. Transport Layer

TCP · UDP · SCTP · DCCP · SPX

3. Network Layer

IP (IPv4, IPv6) · ICMP · IPsec · IGMP · IPX · AppleTalk

2. Data Link Layer

ATM · SDLC · HDLC · ARP · CSLIP · SLIP · GFP · PLIP · IEEE 802.3 · Frame Relay · ITU-T G.hn DLL · PPP · X.25 · Network Switch · DHCP

1. Physical Layer

EIA/TIA-232 · EIA/TIA-449 · ITU-T V-Series · I.430 · I.431 · POTS · PDH · SONET/SDH · PON · OTN · DSL · IEEE 802.3 · IEEE 802.11 · IEEE 802.15 · IEEE 802.16 · IEEE 1394 · ITU-T G.hn PHY · USB · Bluetooth · Hubs

This box: **view** · talk · edit

Grafika z: http://en.wikipedia.org/wiki/OSI_model

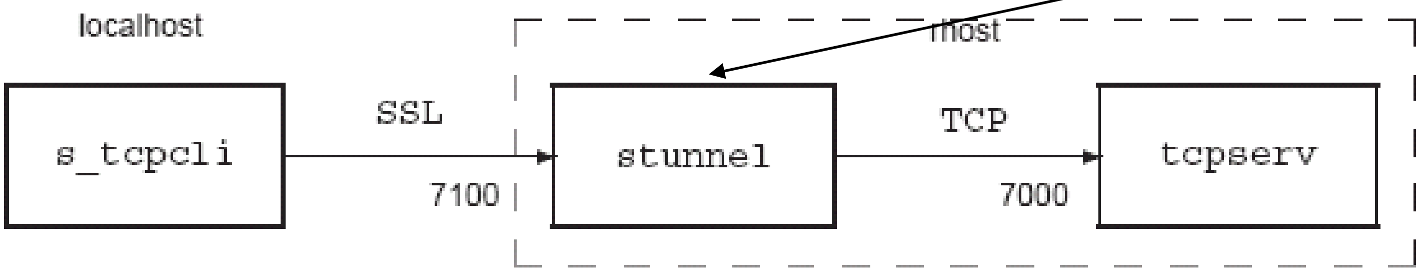
SSL and secure tunneling

- `stunnel` can make a secure tunnel for use by applications which communicate with TCP.
- `ssh` provides not only secure encrypted communications between two untrusted hosts over an insecure network; it also enables remote logging and remote execution of commands.
- `stunnel` requirements:
  - Connection-oriented protocol (TCP not UDP).
  - The protocol cannot use multiple connections (e.g.. FTP).
  - The protocol cannot use OOB data (e.g. TELNET)
  - Remote host cannot use an application-specific protocol (e.g.SSLTELNET, because SSL is a negotiable option).

Example (for `stunnel` version 3)

```
rhost> ./tcpserv 7000
rhost> /usr/sbin/stunnel -f -d 7100 -r 7000 -p server.pem
localhost> ./s_tcpcli rhost 7100
```

Server certificate



localhost
rhost
SSL
TCP
s_tcpcli
stunnel
tcpserv
7100
7000