Security

Use these slides in conjunction with the material from 2004.

Some parts based on or inspired by the slides from
Prof. Claudia Eckert, Jouni Karvo, Pirkko Kuusela, Pasi Lassila
Past contributions by
Olaf Bergmann, Carsten Bormann, Dirk Kutscher

Security

- Background and security properties
- Cryptographic algorithms
- Security mechanisms
- Security protocols and systems
- Security Devices

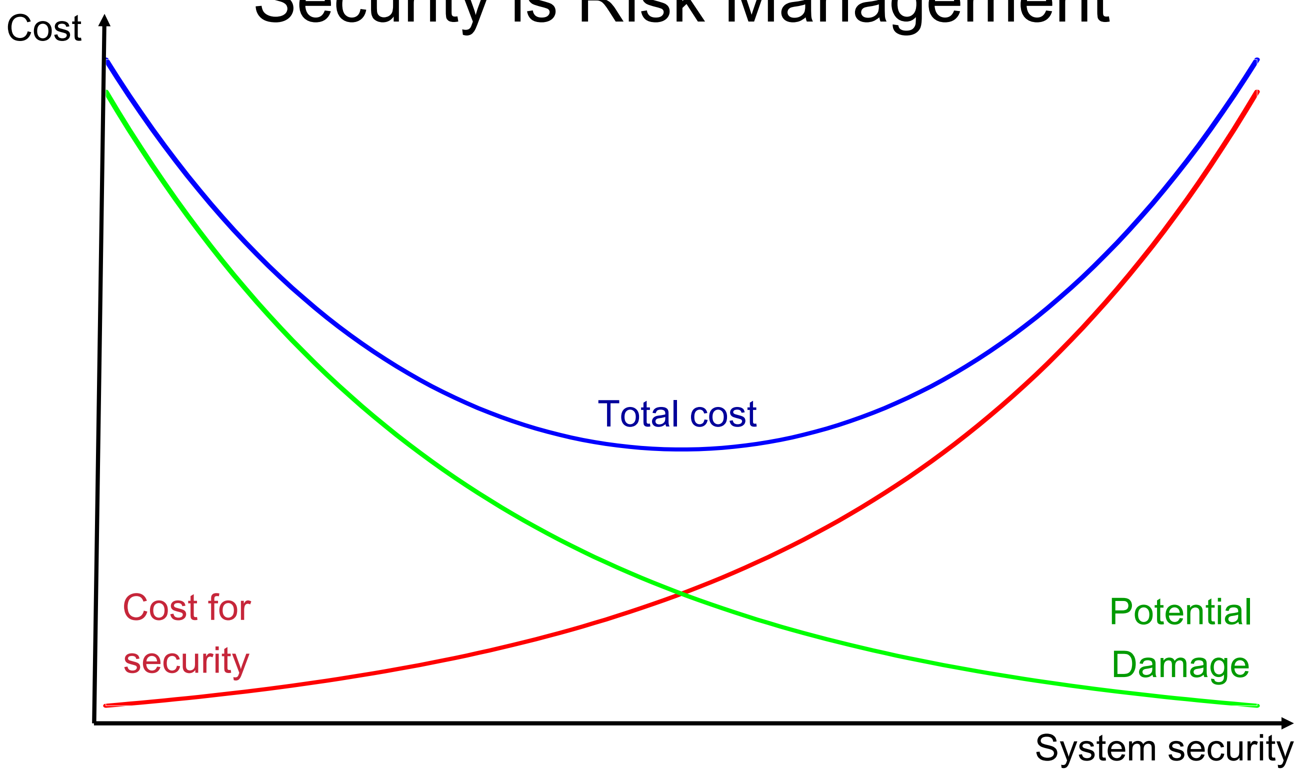
Motivation for Security in Communications

- Pretty much common sense today
- Open communications
  - Global access to information resources
  - Ubiquitous availability of communication networks
- Protection
  - Assets (information, other valuables)
  - Privacy of persons
  - Transactions in critical communications
- "Electronic business"
  - Authentication, Authorization, and Accounting (AAA)
  - Offer (digital) services to known (and unknown) customers
- (Legal) requirements

Some Terms in Brief

- (Communication) protocols / systems may have **weaknesses**
- A weakness may (but need not) cause a **vulnerability**
- A vulnerability can be exploited.
- **Threats** may makes use of one or more vulnerabilities.
- **Attacks** may result from threats to the system.
- Attacks may result in **damage** to the system.
- **Risk** = probability of an attack × amount of damage

Security is Risk Management



Some Sources of Attacks

- 50% of all attacks from the inside (employees)
  - Laziness, personal gain, frustration
  - May be triggered by third parties: social engineering!
- Hacker, "script kiddies"
  - Curiosity, thrill, addiction
  - Experiencing how far they can get
  - Most important: peer recognition in the community!
- "Professional": industry espionage, secret service
- (Organized) crime
  - Blackmailing: e.g., for online gambling sites
  - Competitive edge: DoS against your opponent

Some Security Threats

- Connection / session hijacking
- Masquerading / impersonation / identity spoofing
- Man-in-the-middle (MITM) attacks
- Theft of service

- Connection / session disruption or termination
- Replay

- Eavesdropping
  - Content sniffing, traffic analyses

- Denial-of-service
- Break in

Solutions

- Achieving security goals
  - Preventing attacks
  - Noticing attacks (e.g., monitoring, intrusion detection)
  - Limiting damage from attacks (e.g., isolation)

- Management aspects (planning, training, guidelines, policy, etc.)

- Technical measures
  - Overall system design
  - Protection of hosts and other devices
    - Physical; OS & software (e.g., regular updates)
  - Secure algorithms and protocols
  - Security devices (e.g., firewalls)

Security Properties

- Confidentiality of messages
  - Protection against eavesdropping
- Authentication of peers, messages
  - Protection against masquerading of senders
  - Prerequisite for authorization of actions
- Integrity of messages
  - Protection against modification in transit
- Non-repudiation of actions (accountability)
  - message origin
  - Reception of messages

Related Properties

- Availability of systems and services
  - Protection against maliciously caused overload
    - E.g., denial-of-service attacks
  - Distinction from dependability and safety
    - Different background
    - Ensure correctness, protect against malfunctioning
- Privacy of (personal) information
  - Partially in competition with authentication

Security

- Background and security properties
- Cryptographic algorithms
- Security mechanisms
- Security protocols and systems
- Security Devices

Cryptographic Algorithms

- Cipher algorithms
  - Symmetric algorithms: DES (56-bit), AES (128-bit and more)
  - Asymmetric (public-key) algorithms: e.g., RSA, PGP
- Digest algorithms
  - Compute a digest (“finger-print”) of a message: SHA-1, MD5
  - Basis for hashed message authentication code (HMACs)
- Pseudo random number generators
  - Generate keys for ciphers and HMAC generators

General

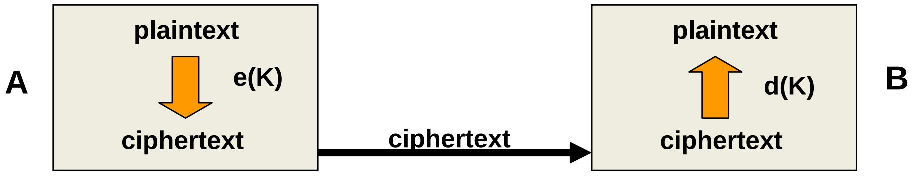
- Algorithms should be publicly known
  - No security by obscurity
  - Enables broad community review and attempts to break the algorithm
  - Strength should only depend on the strength of the chosen key
- Attacks against algorithms
  - Crypto-text only
  - Known plaintext attacks
  - Chosen plaintext attacks
- Algorithms should not reveal additional hints to the attacker
  - Allow for brute force searching of the entire key space as only resort
  - Large key space is important! (today: $\geq 2^{128}$ for symmetric, $\geq 2^{2048}$ for asymmetric)
  - Need to be computationally efficient; operate on blocks and (infinite) streams

General (2)

- Asymmetric ciphers usually rely on complexity of math operations
  - E.g., finding prime factors of a really large number
  - Issue: “really large” grows over time; keys need to get longer
    - Solely depends on computing power available
  - Economic aspect: effort invested vs. value gained
- One-way functions (message digests)
  - Map a large data portion onto a fixed length hash (or digest)
    - Many-to-one mapping, clashes possible
  - Needs to be defined so that it is not possible to construct a second (meaningful) messages that maps to the same output
    - Changing any bit in the input results in changing all the output
  - Need to be computationally efficient
    - If large amounts of data need to be handled

Symmetric Ciphers

- Encryption key = decryption key



- Transposition ciphers
  - Rearrange the occurrence of characters

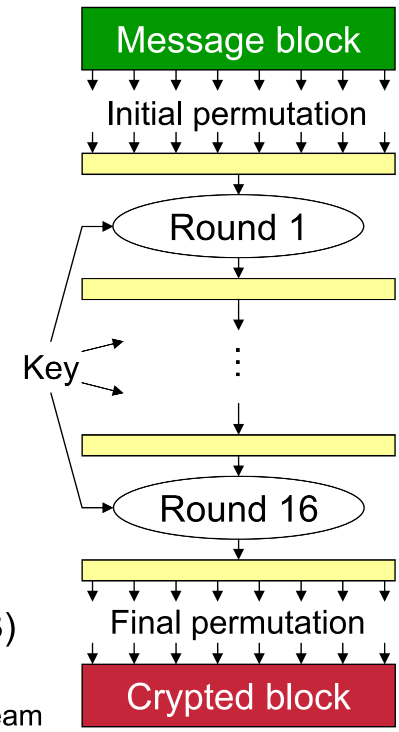
HELLOWORLD → HLOOL ELWRD → HLOOLELWRD

- Substitution ciphers
  - Simples case: Caesar chiffre: A ➔ D, B ➔ E, …, X ➔ A, Y ➔ B, Z ➔ C

HELLOWORLD → KHOORZRUOG

Example: DES

- Block cipher
  - 64 bit block length, 56 bit key length
  - Three phases
    - 64 bits in block are permuted
    - 16 rounds of identical operations
    - Inverse of the original permutation
- Four modes of operation
  - Electronic Code Book (ECB)
    - Substitution cipher for 64 bit blocks
  - Cipher Block Chaining (CBC)
    - Link encryption of each code block to its predecessor (XOR)
  - Output Feedback (OFB) and Cipher Feedback (CFB)
    - Use DES as a stream cipher
    - (Parts of) encryption result XORed with continuous input stream

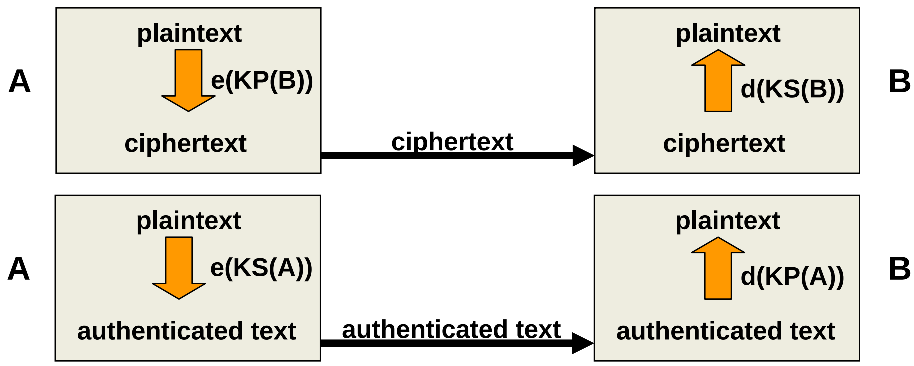


DES Successors

- Simple DES no longer considered secure
  - Can be “easily” broken with brute force
- Triple DES: Enhance the key space by running DES 3 times
  - DES-EEE_3: three different encryption keys
  - DES-EDE_3: three different keys: encryption, decryption, encryption
  - DES-EEE_2: two different keys: keys for first and third encryption identical
  - Effective key length up to 168 bits
- Official DES Successor: Advanced Encryption Standard (AES)
  - Multiple key lengths: 128, 192, 256 bits
- Other algorithms suitable for software: Twofish, Blowfish, RC5

Algorithms: Public-Key Cryptography

- Two keys for each communication partner
  - Public key ➔ published
  - Private key ➔ kept secret
- Messages encrypted with the public key can only be decrypted with the private key (and vice versa)
- Can be used to provide different security services
  - Encryption:
  - Authentication:



RSA: Principle

- Rivest, Shamir, Adleman (1978)
- Background: complexity of prime factorization of large numbers
- Algorithm:
  - Choose two large numbers p and q and calculate $n = p \times q$
  - Calculate an encryption key $e < n$ such that e and $(p – 1) \times (q – 1)$ are relatively prime
  - Compute decryption key d as $d = e^{-1} \times \text{mod } ((p – 1) \times (q – 1))$
  - Construct public key = (e, n) and private key = (d, n)

Encryption: $c = m^e \bmod n$
Decryption: $m= c^e \bmod n$

RSA: Example

- Let p = 7 and q = 11 [everyone seems to be using these]
- We get $n = p \times q = 7 \times 11 = 77$
  and $(p – 1) \times (q – 1) = 6 \times 10 = 60$
- Pick e = 7 [7 and 60 are relatively prime]
- We get $d = 7^{-1} \bmod 60$
  or $7 \times d \bmod 60 = 1$ [$7 \times 43 = 301 \bmod 60 = 1$]
- Public key (e, n) = (7, 77)
- Private key (d, n) = (43, 77)

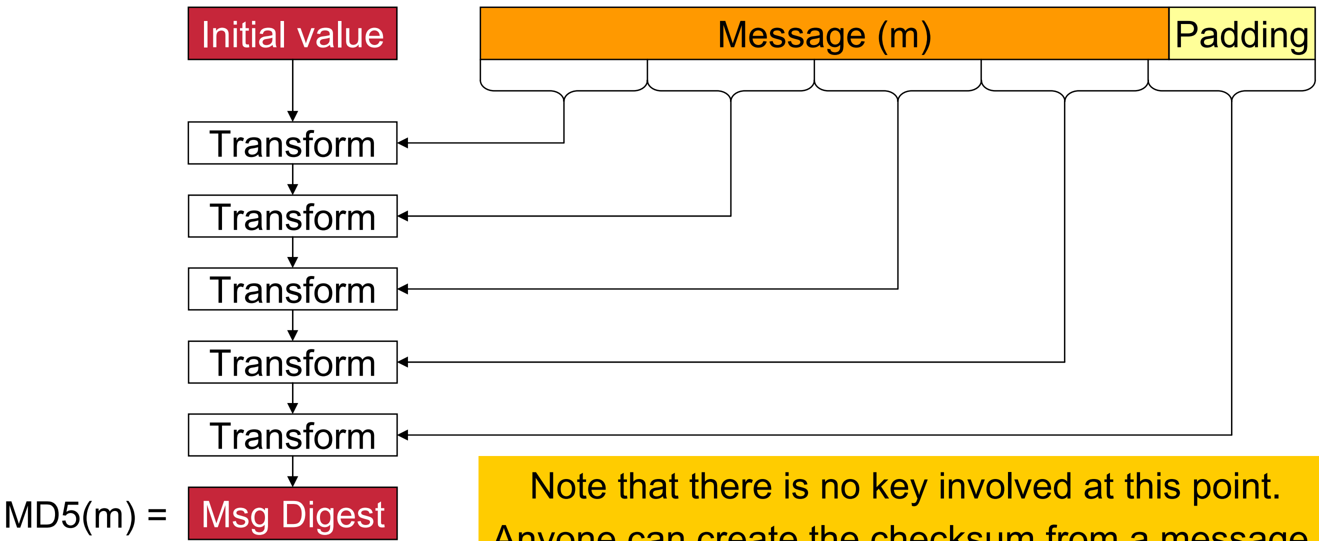
- Message: 9 encryption: $c=m^e \bmod n = 9^7 \bmod 77 = 37$
  decryption: $m=m^c \bmod n = 37^{43} \bmod 77 = 9$

Algorithms: Message Digest

- RSA computation is expensive
  - May be difficult to operate (imagine RSA key stored on a chipcard)
    - need to tunnel messages through card
- Seek representative for the full message
- Cryptographic checksum
  - Stronger than CRC (typically longer and more complex)
  - But: several messages map to the same checksum
- One-way function
  - Do not allow determining the original message based upon the digest
  - Do not allow determining another message that yields the same digest
- May be used for integrity protection
  - Encrypt the checksum with RSA or use a shared secret (later)

Message Digest Algorithms

- One-way function (examples: MD5, SHA-1)
  - Map a large data portion onto a fixed length hash (or digest)



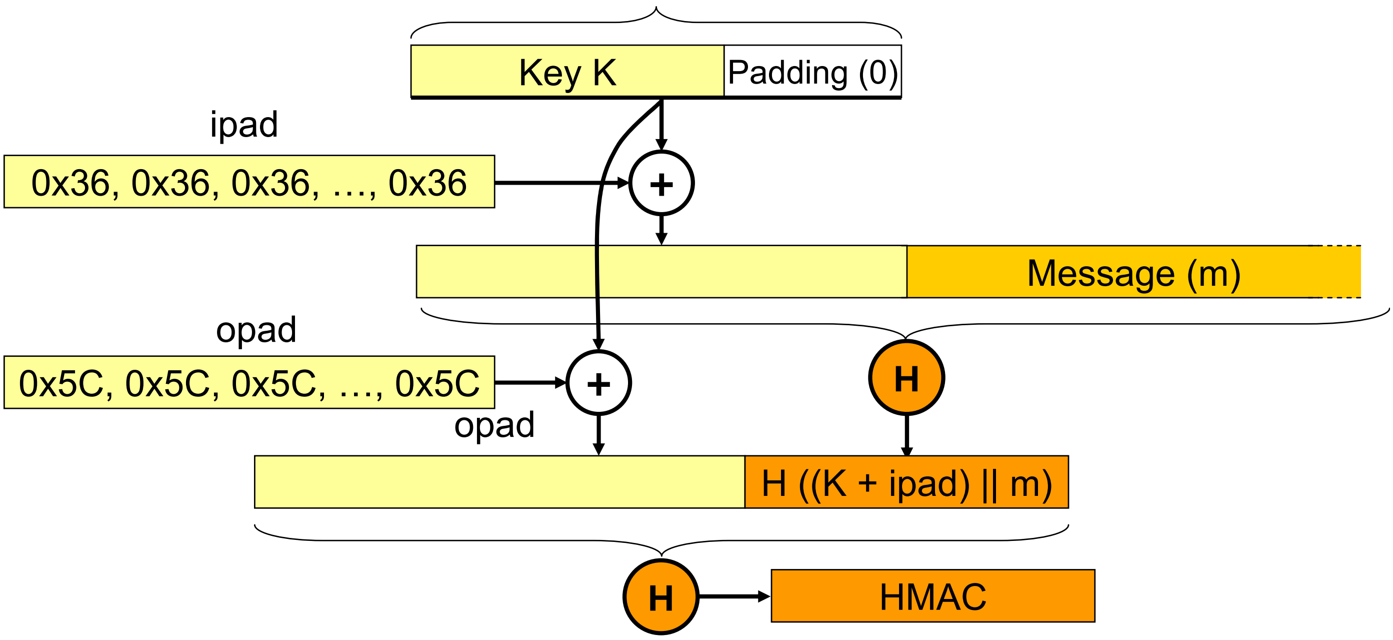
Note that there is no key involved at this point.
Anyone can create the checksum from a message.

Message Authentication Code (MAC)

- Combine integrity protection and message authentication
  - Base the outcome of the digest function on a key
- Use DES in CBC mode with a key, last output block is digest
- Keyed MD5: Append keying material to the message
  - Shared key k: sender sends **m + MD5(m+k)**
  - Random key k: sender sends **m + MD5(m+k) + E(k, public_receiver)**
- Hashed Message Authentication Code (HMAC)
  - Use shared or random key to vary initial values
  - Important: no cryptography involved ➔ no export regulations
  - HMAC-H(k, text) = H(k XOR opad || H(k XOR ipad || text))

HMAC Overview

- Hash function H takes blocks of size **B** as input



Algorithms: Random Number Generation

- Random numbers: dynamically generated keys, challenges, etc.
  - Pseudo-random numbers are predictable and therefore not suitable
- Time of days, process id, etc. are also somewhat predictable
  - Also an attacker knows the time
  - The process id is usually drawn from a very limited space
- Use user "input" (mouse movements, key strokes)
  - But not necessarily reliable (and there may not be input devices)
- Use hardware input
  - Measure entropy of the environment
    - Radioactive sources, atmospheric noise from radio, other
    - Project example: lava lamp image
  - Nice reference: www.random.org

Security

- Background and security properties
- Cryptographic algorithms
- Security mechanisms
- Security protocols and systems
- Security Devices

Some Security Mechanisms

- Message integrity protection and authentication
  - (Hashed) Message authentication codes: (H)MACs ✓
  - Digital signatures
- Key exchange/management
  - Obtain and validate public keys
  - Securely exchange keys between communication peers
  - Generate shared session keys
  - Usually address negotiation of algorithms and re-keying as well
- User Authentication
  - Establish the identity of communicating peers
  - Often prerequisite for (meaningful) key exchange

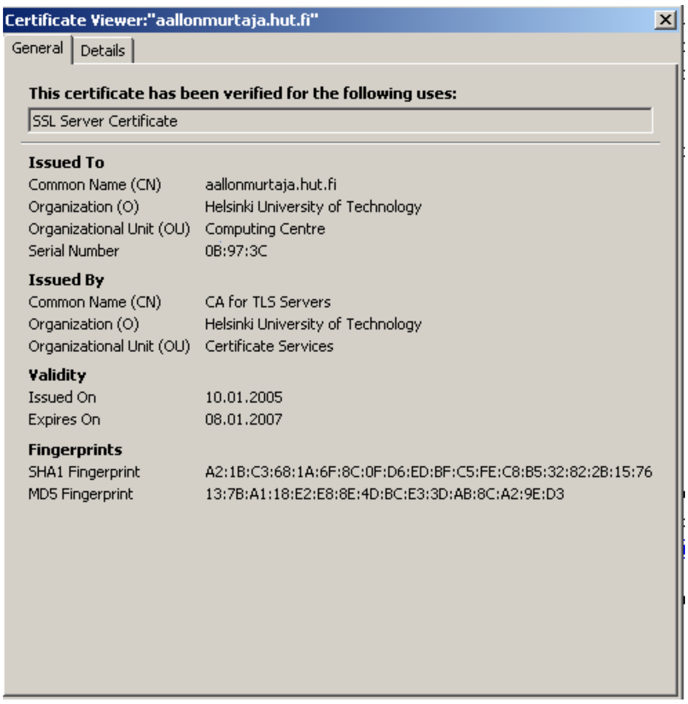
Digital Signatures

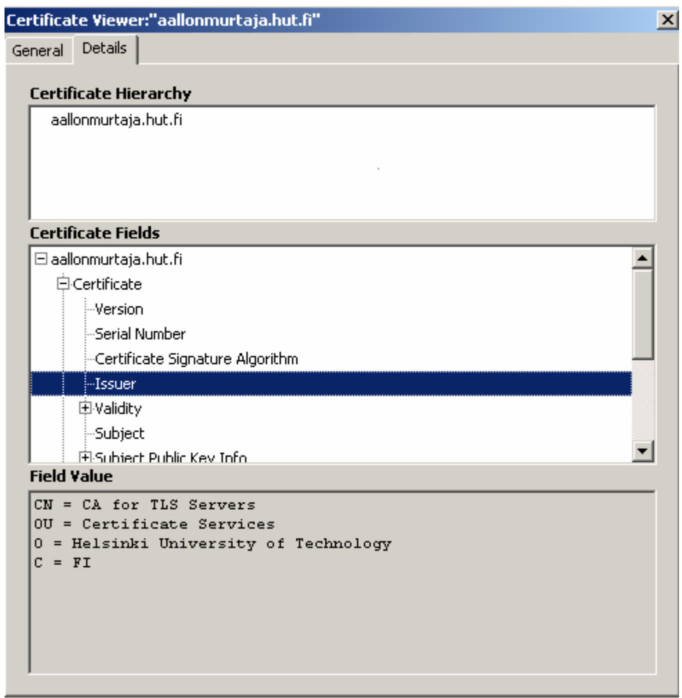
- An application of public-key cryptography (and message digests)
  - Main advantage of public-key cryptography
    - No need to transmit shared secrets (password, secret keys)
    - Private key does not have to be disclosed
- Simplest case: encrypt full message w/ private key: E(m, private)
  - Provide identity along with message
  - Receiver can validate message origin by decrypting with public key
- Issues:
  - Replay attacks possible ➔ ensure message uniqueness (e.g., timestamp)
  - Computational overhead incurred by RSA ➔ sign message summary
- Sender sends: m + E (MD5(m), private)
  - Includes information about digest algorithm, identity, timestamp, …

Issues with Public Key Cryptography

- Problem: Obtaining public keys of communication partners
  - Transmission over unsecured channels is not useful
  - Authentication (and integrity) is required
- Solution: **Certificates**
  - Certificates bind a public key to an entity
  - Digitally signed by a Certificate Authority (CA)
  - A CA can provide a certificate for its own public key
    - Signed by a well-known root CA
  - Hierarchy of CAs and corresponding directory services
    ➔Public Key Infrastructure (PKI)
    - important prerequisite for deploying public key cryptography

Sample Certificate





X.509 Certificates

```
TBSCertificate  ::=  SEQUENCE  {
       version          [0]  EXPLICIT Version DEFAULT v1,
       serialNumber          CertificateSerialNumber,
       signature             AlgorithmIdentifier,
       issuer                Name,
       validity              Validity,
       subject               Name,
       subjectPublicKeyInfo SubjectPublicKeyInfo,
       issuerUniqueID   [1]  IMPLICIT UniqueIdentifier OPTIONAL,
       subjectUniqueID [2]  IMPLICIT UniqueIdentifier OPTIONAL,
       extensions       [3]  EXPLICIT Extensions OPTIONAL
                             -- If present, version MUST be v3
       }

Extensions:
        Subject Alternative Names
        Issuer Alternative Names
```

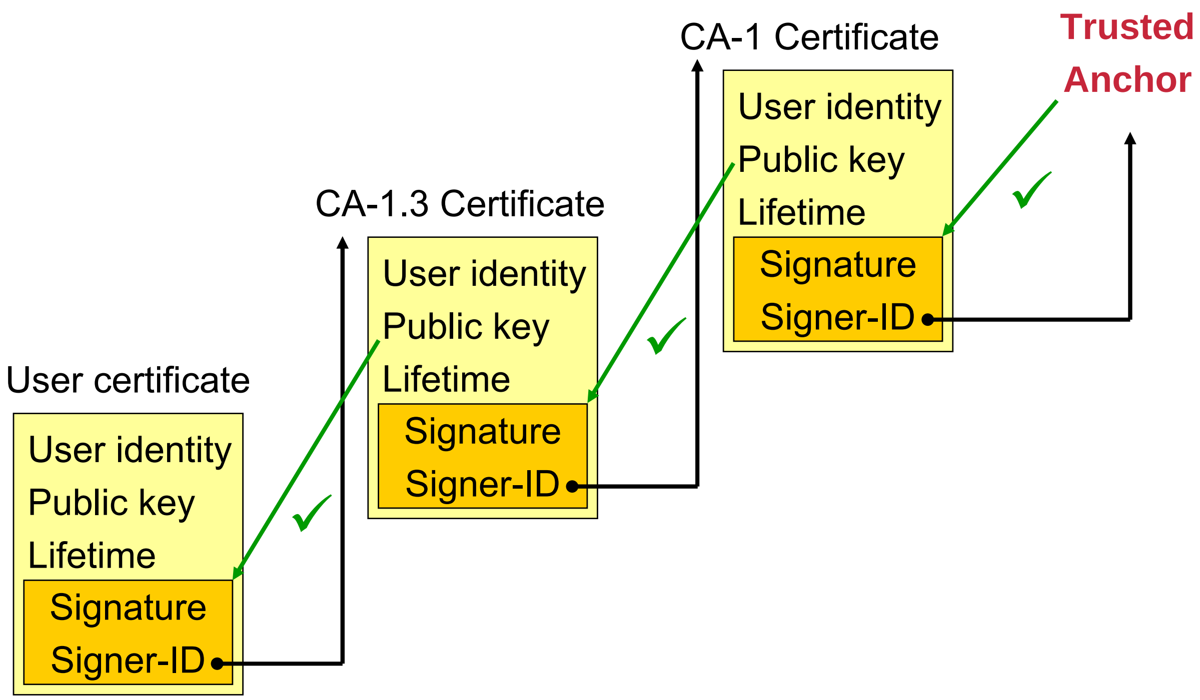

X.509 Certificates

```
SubjectAltName ::= GeneralNames
IssuerAltName ::= GeneralNames


GeneralName ::= CHOICE {
        otherName                       [0]     OtherName,
        rfc822Name                      [1]     IA5String,
        dNSName                         [2]     IA5String,
        x400Address                     [3]     ORAddress,
        directoryName                   [4]     Name,
        ediPartyName                    [5]     EDIPartyName,
        uniformResourceIdentifier       [6]     IA5String,
        iPAddress                       [7]     OCTET STRING,
        registeredID                    [8]     OBJECT IDENTIFIER }
```

Public Key Infrastructure (PKI)

- Have a (trusted) party certify the key's correctness:
  - That the key belongs to a particular **subject** (person, device)
  - That the key is (still) valid
- Certification Authority (CA)
  - Asserts a an identity – public key mapping + validity period in a **certificate**
  - Digitally signs the certificate to ensure authenticity and integrity
  - Signature can be validated by another CA
  - Validation "hierarchy" or "path"
- Provide a repository to store all this information
  - Interaction between CAs
  - Register, publish, and revoke certificates

CA Hierarchy and Certificate Validation

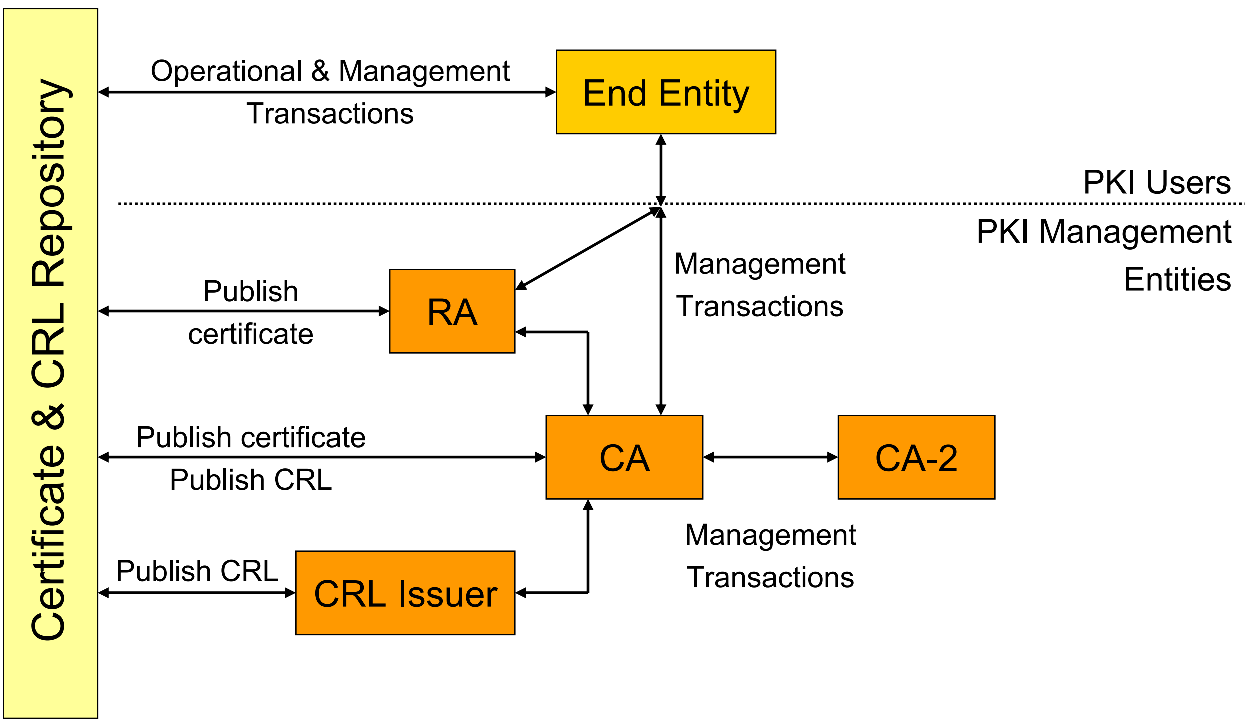


Public Key Infrastructure (PKI)

- Certification Authorities (CAs)
  - Optional: generate private – public key pairs
  - Validate a private–public key pair (authenticate subject)
  - Generate certificates
  - Publish certificates (to other CAs, to other entities for signature validation)
  - Revoke certificates (in case of compromise)
  - Management functions and inter-CA communications
  - Key pair management: update, recovery, …
- Registration Authorities (RAs)
  - Optional, for interaction with the user, key pair generation, etc.
  - Functionality may completely be integrated with CA (logical decomposition)
- End entity: acts on behalf of the user (user equipment)
  - Stores private keys “securely”, inquires certificates
- Certificate and CRL Repository
- RFC 2510, 2511, 2585, 3280(!)

HELSINKI UNIVERISITY OF TECHNOLOGY
NETWORKING LABORATORY

Public Key Infrastructure



Self-Signed Certificates

- Certificates usually handed out by CA
  - Workable for servers
  - Impractical + expensive + hard to deal with for end users
- Alternative: self-signed certificate
  - Self-generated key pair
  - Self-created certificate
    - Signed with the own key
  - Sufficient to satisfy (syntactical) protocol requirements (e.g., TLS)
  - Semantics may suffice for certain applications, too
    - Certificates convey during initial contact
    - Stored at both peers
    - Allows to validate that the same peer was contacted before

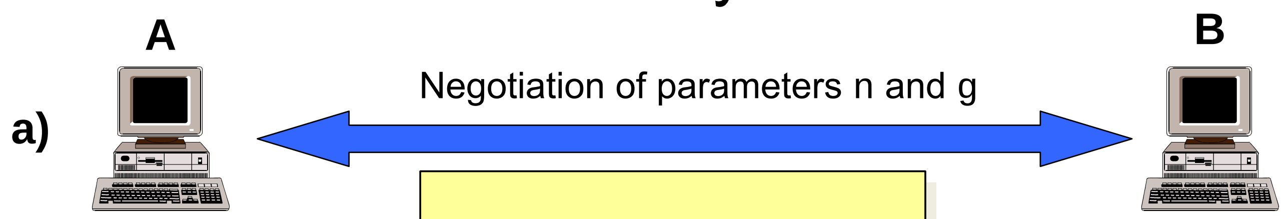
Key Exchange Protocols

- Public-key cryptography not very efficient for encrypting larger messages
  - Often used to securely agree a *session key* (key exchange)
  - Session encryption is done with symmetric cryptography

Alternatives for key exchange:

- Pre-shared key
  - Offline agreement of a shared key for a symmetric algorithm
  - All the problems of shared keys that have to be configured on two systems…
- Exchange session key using public-key crypto
- Diffie-Hellman key generation
  - A *key generation* algorithm
  - Allows two partners to securely compute a common secret key
  - No eavesdropping possible ➔ perfect forward secrecy
  - Preventing man-in-the-middle attacks requires authentication
    ➔ combine with public-key crypto

Diffie-Hellman Key Generation



A and B agree on a large prime, *n* and generator *g,* such that *g* is primitive mod *n*.

Example: g = 2, n =

```
FFFFFFFF FFFFFFFF C90FDAA2 2168C234 C4C6628B 80DC1CD1
29024E08 8A67CC74 020BBEA6 3B139B22 514A0879 8E3404DD
EF9519B3 CD3A431B 302B0A6D F25F1437 4FE1356D 6D51C245
E485B576 625E7EC6 F44C42E9 A63A3620 FFFFFFFF FFFFFFFF
```

(RFC2409)

One Moment – Primitive Root…?

- Just a quick note:
- R is a primitive root of P [R = prim_root (P)]

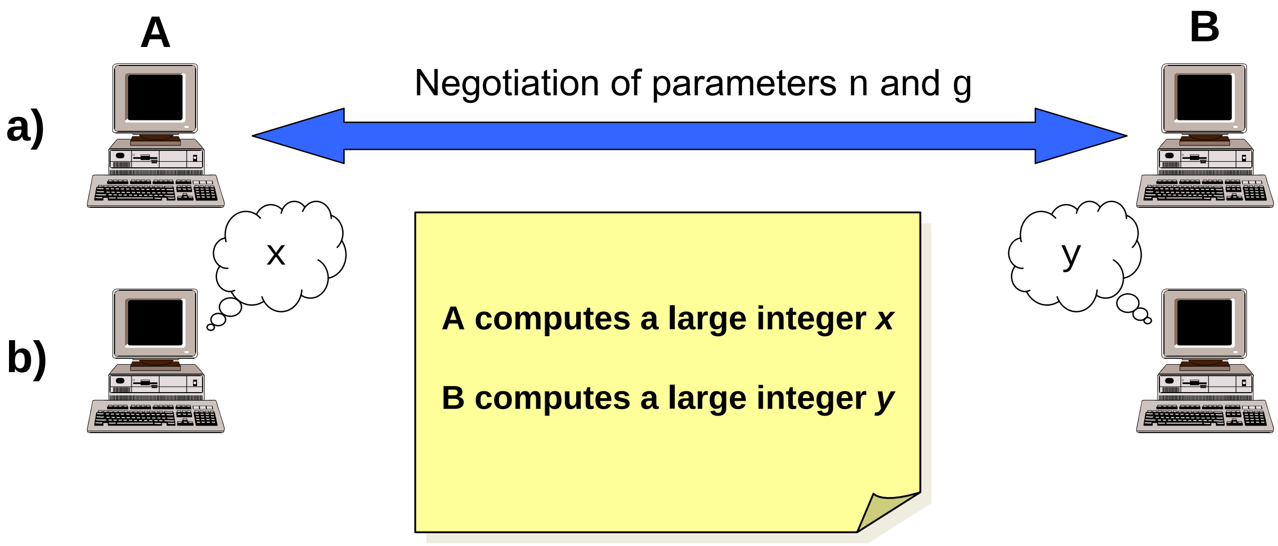
If $R^{P-1} \bmod P = 0$

AND

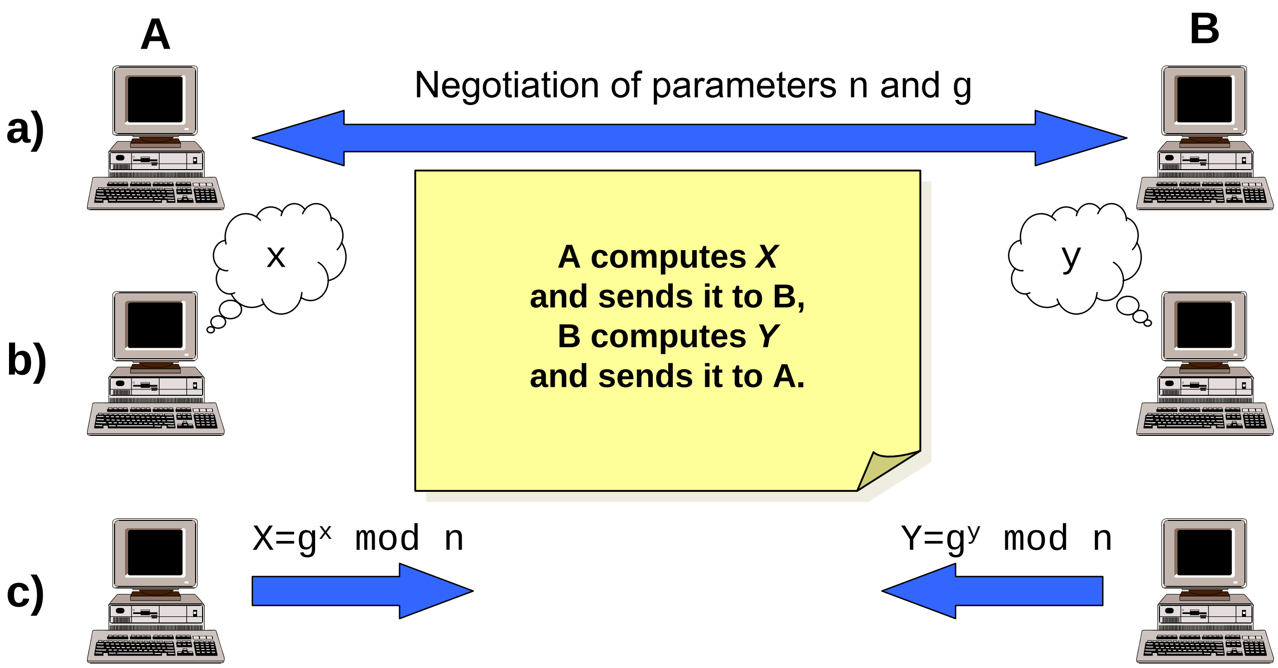
If for all $k < P-1$ $\quad R^k \bmod P \neq 0$

- Now let’s do a little brain exercise…

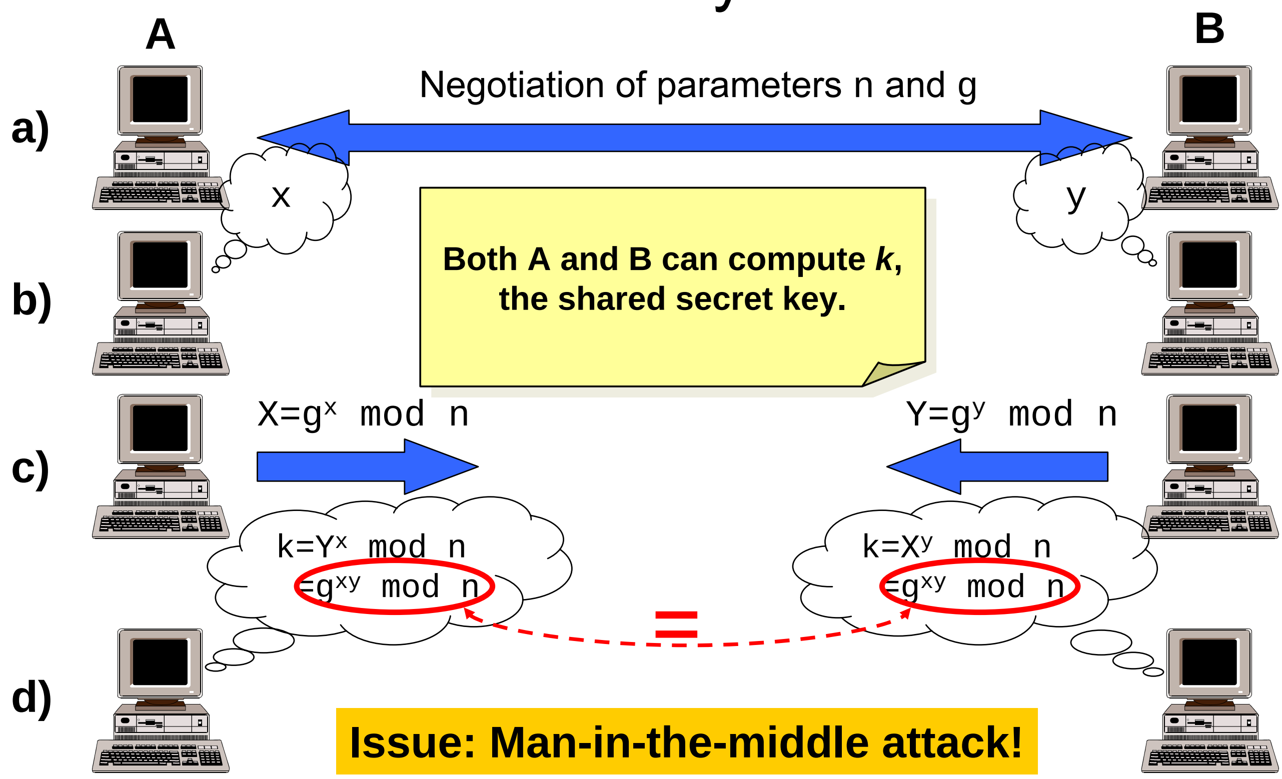
Diffie-Hellman Key Generation



Diffie-Hellman Key Generation



Diffie-Hellman Key Generation



HELSINKI UNIVERISITY OF TECHNOLOGY
NETWORKING LABORATORY

User Authentication

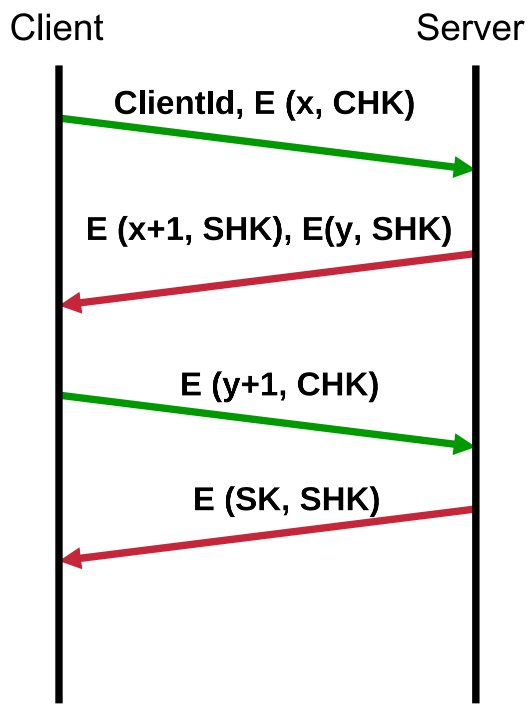
- Establish the identity of one or both peers (client, server)
  - Three-way handshake
  - Trusted third party
  - Public key authentication
- Exchange (or generate) session key along with authentication

<u>Related system aspects</u>

- Backend protocols to access security databases
  - Authentication, Authorization, and Accounting (AAA)
  - RADIUS (widely deployed)
  - DIAMETER (more complete successor)
- Important aspect: Identity management

Three-way Handshake

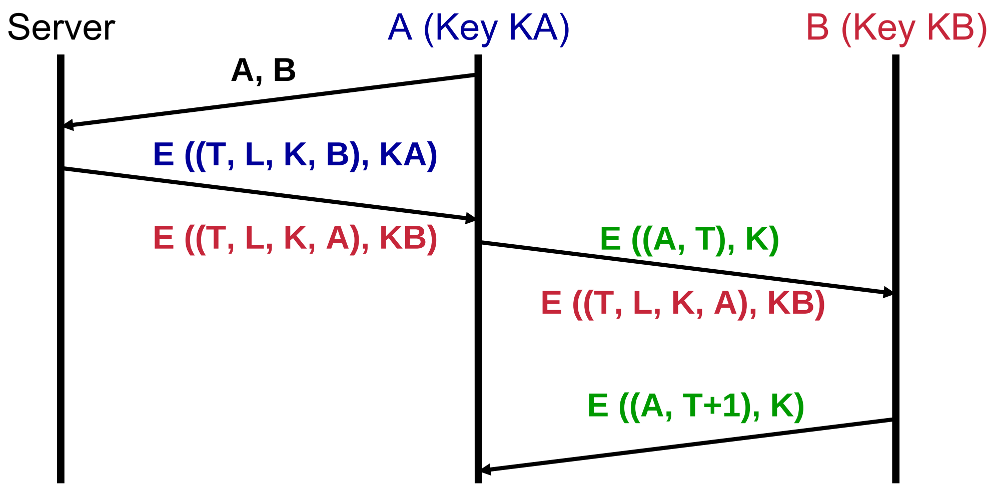
- Assumption: Shared secret key
  - SHK = CHK = Server/client handshake key
- E (m, k): encrypt message m with key k
- Client chooses random number x
  - Sends encrypted with shared secret
- Server uses ClientId to determine key
  - Decrypts and returns x+1 encrypted
  - Also chooses random number y
  - Sends encrypted with shared secret
- Client decrypts x+1 and y
  - If x+1 decrypts correctly: server authenticated
  - Returns y+1 encrypted
- Server decrypts y+1
  - If y+1 decrypts correctly: client authenticated
  - Generated session key SK and sends encrypted



HELSINKI UNIVERISITY OF TECHNOLOGY
NETWORKING LABORATORY

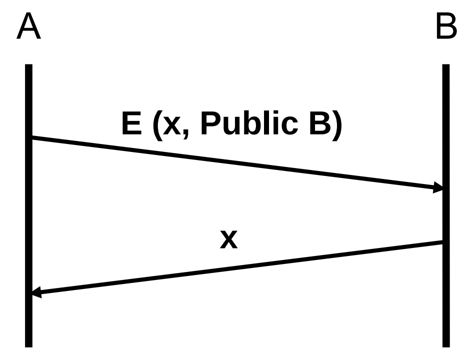
Trusted Third Party (Kerberos)

- Authentication server as trusted entity
  - Keys of A and B (KA and KB) are shared with the server
- A and B want to communicate securely (auth + encr)
- Server provides session K with lifetime L at time T



Simple Public Key Authentication

- No shared key necessary!
  - But a public key infrastructure of some sort
- A generates random number x
  - B decrypts and returns it ➔ authenticated
- Different related approaches conceivable



Security

- Background and security properties
- Cryptographic algorithms
- Security mechanisms
- Security protocols and systems
- Security Devices

Some Security Protocols

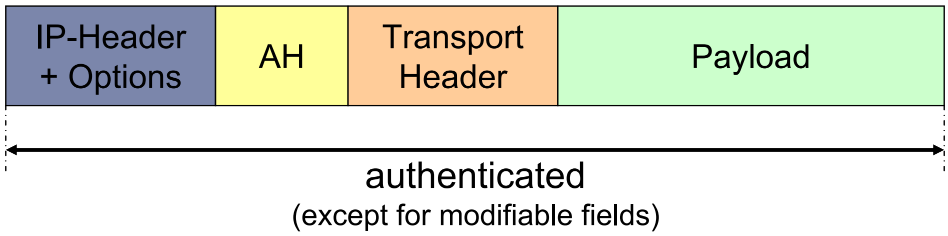
- Link layer
  - Link layer authentication: IEEE 802.1x
  - Wired Equivalent Privacy (WEP) for IEEE 802.11 Wireless LANs (insecure)
  - IEEE 802.11i as successor
- IP layer
  - IPsec
  - VPN tunneling alternatives: PPTP, L2TP
- Transport layer
  - Secure Shell (SSH) + associated protocols (file transfer, etc.)
  - Transport Layer Security (TLS)
- Application layer
  - HTTP digest authentication
  - Privacy enhanced mail (PEM)
  - Pretty good privacy (PGP)
  - S/MIME protection of message contents

IPSEC (1)

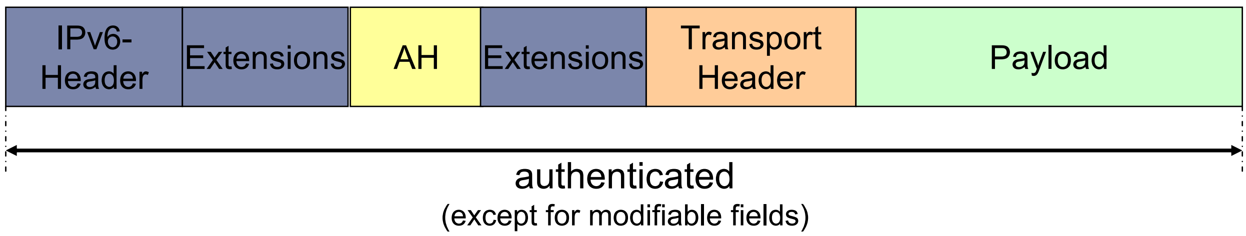
- Security Architecture for the Internet Protocol (RFC2401)
  - Provide security services at the IP layer (network layer)
  - Protect one or more paths between a pair of hosts
    - Special case: protect path between gateways
- Security Protocols
  - Authentication Header (AH, RFC2402)
    - Authentication
    - Integrity
    - Usually used with HMAC-MD5-96 or HMAC-SHA-1-96
  - Encapsulating Security Payload (ESP, RFC2406)
    - Authentication
    - Integrity
    - Usually used with HMAC-MD5-96 or HMAC-SHA-1-96
    - Encryption
    - Default cipher: DES (insecure), → 3DES, AES

IPsec Authentication Header (Transport Mode)

IPv4

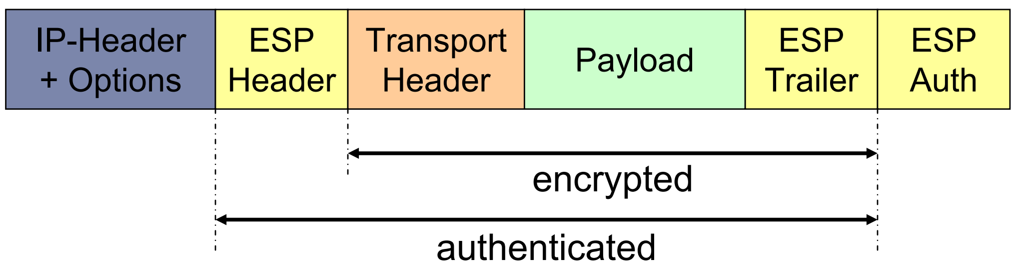


IPv6

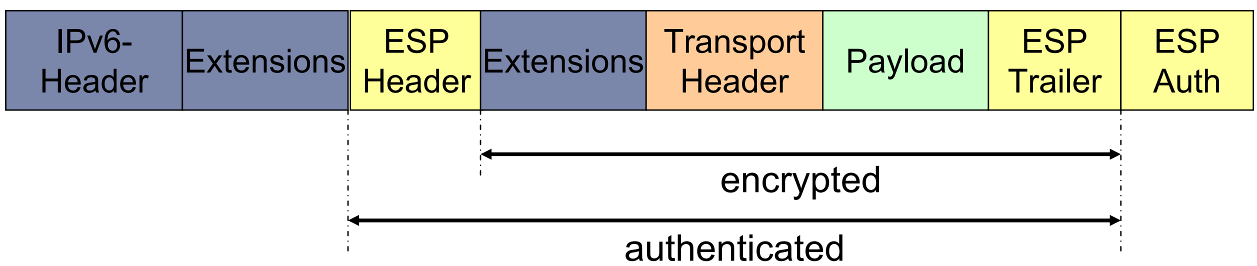


IPsec Encapsulation Payload (Transport Mode)

IPv4



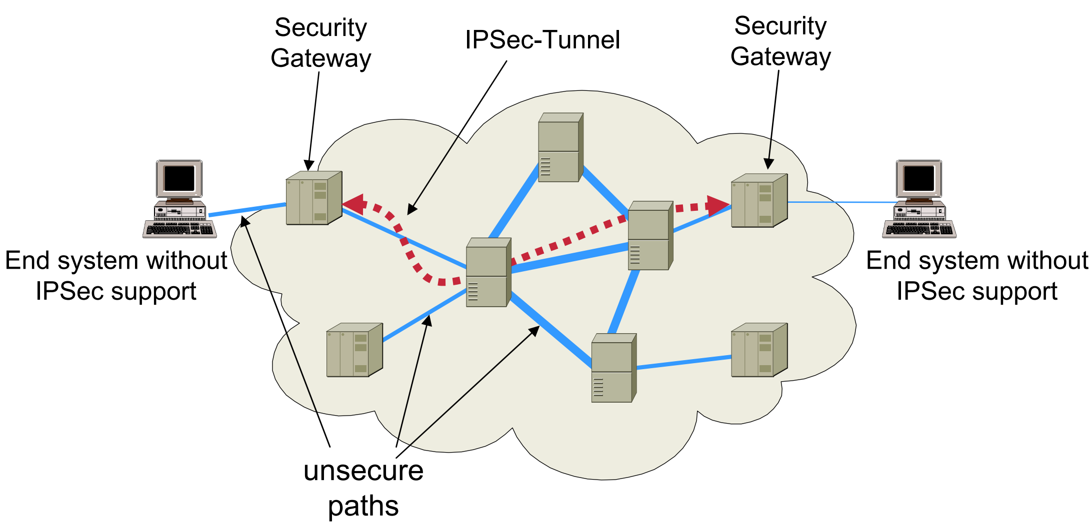
IPv6



IPSEC (2)

- Security Associations (SAs)
  - Fundamental abstraction for IPsec
  - A unidirectional “connection” offering security services
  - One SA for each service and for each direction
  - Combining SAs, if more than one service is needed
  - Identified by SPI, IP destination addr, protocol ID (ESP or AH)
- Two Types of SAs
  - Transport Mode
    - SA between two hosts
    - Protection for upper layer protocols (UDP, TCP, …)
  - Tunnel Mode
    - An SA applied to an IP tunnel
    - Tunnel IP packets on behalf of hosts (used by security gateways)

IPsec Tunnels



Key Management for IPsec

- Manual key management
  - Configure each system with keying material and SA parameters
  - Only feasible for static environments
  - A VPN tunnel between two sites in a common administrative domain
- Internet Security Association and Key Management Protocol (ISAKMP)
  - Generic key management framework defining procedures and Packet Format for SA management:
  - Establishment, negotiation, modification and deletion
- Internet Key Exchange (IKE)
  - Specialization of ISAKMP for IPsec
  - Two Phases:
    - Establishment of the ISAKMP SA (secure, authenticated bidirectional channel for SA negotiation)
    - SA negotiation for IPsec
  - Negotiable key exchange mechanisms

Security Protocols above IPsec

- IPsec deployment issues
  - IPsec between IP and transport layer
  - Need kernel support for IPsec
  - Most IPsec deployment so far in security gateways

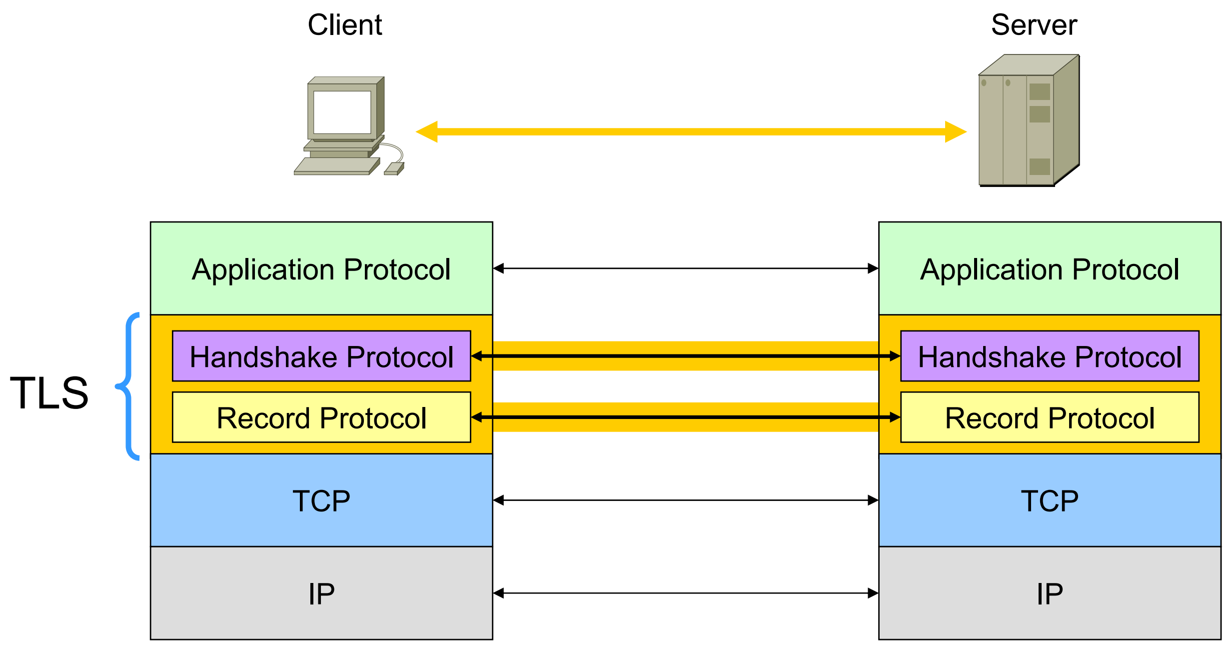
Easier to deploy alternatives:

- TLS = Transport Layer Security
  - Standardized form of SSL (Secure Socket Layer, Netscape)
  - Once established, similar to TCP
  - Secures *connection*, not application semantics
  - In practical use, usually authenticates server, not client
- Application Layer Security Protocols
  - Part of the application layer signaling (e.g., HTTP)
  - Secure the application messages
  - If they are MIME-encoded: S/MIME (secure MIME)
    - Alternative: PGP, PEM

Transport Layer Security (TLS)

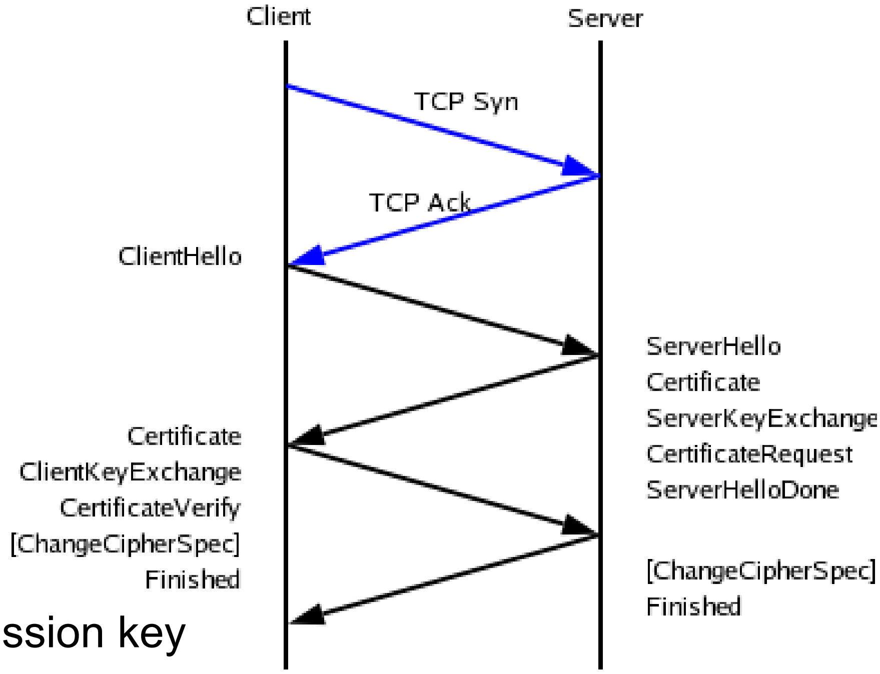
- Provide authentication, confidentiality and integrity between two communicating applications
  - Transparent to applications
  - Often used for client-server applications
  - Used for secure HTTP: https://
- Layered on top of a reliable transport protocol (e.g. TCP)
- Two layers:
  - TLS Record Protocol
    - Provides confidential and reliable (integrity) transport for higher layer protocols
    - Symmetric cryptography and HMACs
  - TLS Handshake Protocol
    - Allows for secure authentication of communication partners
    - Public key cryptography and certificates
    - Used on top of the TLS Record Protocol

TLS

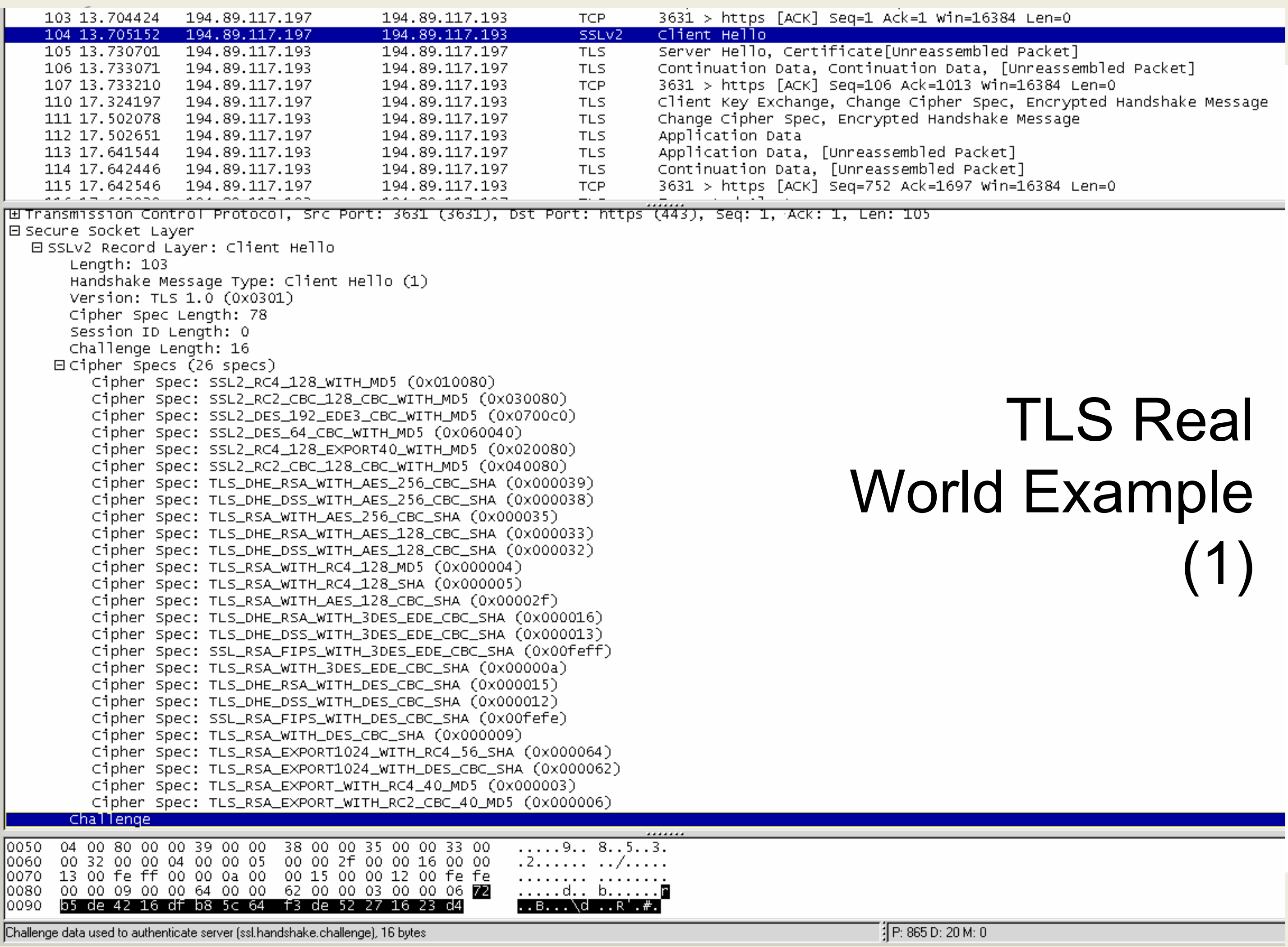


TLS Handshake Protocol Exchange

- ClientHello
  - Offers set of ciphers
  - Includes random value
- ServerHello
  - Selects cipher
  - Includes random value
  - Provides certificate
- Calculate master key
- Derive of session keys for the record layer
- Synchronize on new session key



TLS Real World Example (1)



```
103 13.704424  194.89.117.197  194.89.117.193  TCP    3631 > https [ACK] Seq=1 Ack=1 Win=16384 Len=0
104 13.705152  194.89.117.197  194.89.117.193  SSLv2  Client Hello
105 13.730701  194.89.117.193  194.89.117.197  TLS    Server Hello, Certificate[Unreassembled Packet]
106 13.733071  194.89.117.193  194.89.117.197  TLS    Continuation Data, Continuation Data, [Unreassembled Packet]
107 13.733210  194.89.117.197  194.89.117.193  TCP    3631 > https [ACK] Seq=106 Ack=1013 Win=16384 Len=0
110 17.324197  194.89.117.197  194.89.117.193  TLS    Client Key Exchange, Change Cipher Spec, Encrypted Handshake Message
111 17.502078  194.89.117.193  194.89.117.197  TLS    Change Cipher Spec, Encrypted Handshake Message
112 17.502651  194.89.117.197  194.89.117.193  TLS    Application Data
113 17.641544  194.89.117.193  194.89.117.197  TLS    Application Data, [Unreassembled Packet]
114 17.642446  194.89.117.193  194.89.117.197  TLS    Continuation Data, [Unreassembled Packet]
115 17.642546  194.89.117.197  194.89.117.193  TCP    3631 > https [ACK] Seq=752 Ack=1697 Win=16384 Len=0

⊞ Transmission Control Protocol, Src Port: 3631 (3631), Dst Port: https (443), Seq: 1, Ack: 1, Len: 105
⊟ Secure Socket Layer
  ⊟ SSLv2 Record Layer: Client Hello
      Length: 103
      Handshake Message Type: Client Hello (1)
      Version: TLS 1.0 (0x0301)
      Cipher Spec Length: 78
      Session ID Length: 0
      Challenge Length: 16
    ⊟ Cipher Specs (26 specs)
        Cipher Spec: SSL2_RC4_128_WITH_MD5 (0x010080)
        Cipher Spec: SSL2_RC2_CBC_128_CBC_WITH_MD5 (0x030080)
        Cipher Spec: SSL2_DES_192_EDE3_CBC_WITH_MD5 (0x0700c0)
        Cipher Spec: SSL2_DES_64_CBC_WITH_MD5 (0x060040)
        Cipher Spec: SSL2_RC4_128_EXPORT40_WITH_MD5 (0x020080)
        Cipher Spec: SSL2_RC2_CBC_128_CBC_WITH_MD5 (0x040080)
        Cipher Spec: TLS_DHE_RSA_WITH_AES_256_CBC_SHA (0x000039)
        Cipher Spec: TLS_DHE_DSS_WITH_AES_256_CBC_SHA (0x000038)
        Cipher Spec: TLS_RSA_WITH_AES_256_CBC_SHA (0x000035)
        Cipher Spec: TLS_DHE_RSA_WITH_AES_128_CBC_SHA (0x000033)
        Cipher Spec: TLS_DHE_DSS_WITH_AES_128_CBC_SHA (0x000032)
        Cipher Spec: TLS_RSA_WITH_RC4_128_MD5 (0x000004)
        Cipher Spec: TLS_RSA_WITH_RC4_128_SHA (0x000005)
        Cipher Spec: TLS_RSA_WITH_AES_128_CBC_SHA (0x00002f)
        Cipher Spec: TLS_DHE_RSA_WITH_3DES_EDE_CBC_SHA (0x000016)
        Cipher Spec: TLS_DHE_DSS_WITH_3DES_EDE_CBC_SHA (0x000013)
        Cipher Spec: SSL_RSA_FIPS_WITH_3DES_EDE_CBC_SHA (0x00feff)
        Cipher Spec: TLS_RSA_WITH_3DES_EDE_CBC_SHA (0x00000a)
        Cipher Spec: TLS_DHE_RSA_WITH_DES_CBC_SHA (0x000015)
        Cipher Spec: TLS_DHE_DSS_WITH_DES_CBC_SHA (0x000012)
        Cipher Spec: SSL_RSA_FIPS_WITH_DES_CBC_SHA (0x00fefe)
        Cipher Spec: TLS_RSA_WITH_DES_CBC_SHA (0x000009)
        Cipher Spec: TLS_RSA_EXPORT1024_WITH_RC4_56_SHA (0x000064)
        Cipher Spec: TLS_RSA_EXPORT1024_WITH_DES_CBC_SHA (0x000062)
        Cipher Spec: TLS_RSA_EXPORT_WITH_RC4_40_MD5 (0x000003)
        Cipher Spec: TLS_RSA_EXPORT_WITH_RC2_CBC_40_MD5 (0x000006)
      Challenge

0050  04 00 80 00 00 39 00 00  38 00 00 35 00 00 33 00   .....9.. 8..5..3.
0060  00 32 00 00 04 00 00 05  00 00 2f 00 00 16 00 00   .2...... ../.....
0070  13 00 fe ff 00 00 0a 00  00 15 00 00 12 00 fe fe   ........ ........
0080  00 00 09 00 00 64 00 00  62 00 00 03 00 00 06 72   .....d.. b......r
0090  b5 de 42 16 df b8 5c 64  f3 de 52 27 16 23 d4      ..B...\d ..R'.#.

Challenge data used to authenticate server (ssl.handshake.challenge), 16 bytes    P: 865 D: 20 M: 0
```

TLS Real World Example (2)

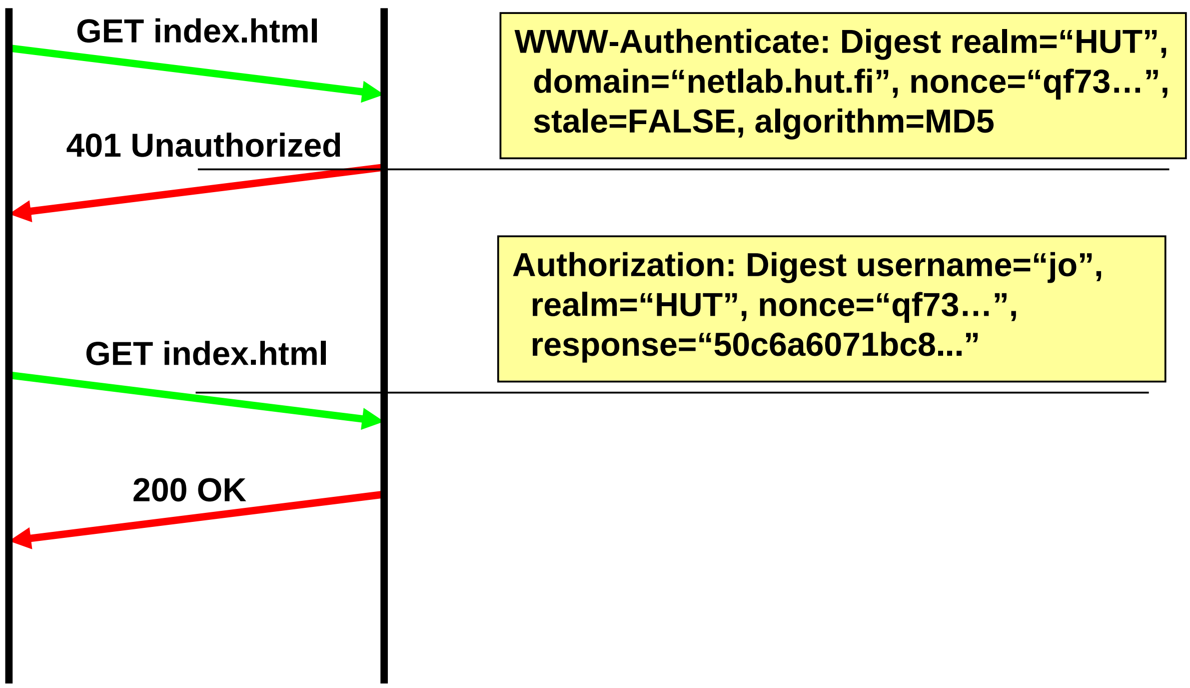
```
103 13.704424  194.89.117.197   194.89.117.193   TCP    3631 > https [ACK] Seq=1 Ack=1 Win=16384 Len=0
104 13.705152  194.89.117.197   194.89.117.193   SSLv2  Client Hello
105 13.730701  194.89.117.193   194.89.117.197   TLS    Server Hello, Certificate[Unreassembled Packet]
106 13.733071  194.89.117.193   194.89.117.197   TLS    Continuation Data, Continuation Data, [Unreassembled Packet]
107 13.733210  194.89.117.197   194.89.117.193   TCP    3631 > https [ACK] Seq=106 Ack=1013 Win=16384 Len=0
110 17.324197  194.89.117.197   194.89.117.193   TLS    Client Key Exchange, Change Cipher Spec, Encrypted Handshake Mess
111 17.502078  194.89.117.193   194.89.117.197   TLS    Change Cipher Spec, Encrypted Handshake Message
112 17.502651  194.89.117.197   194.89.117.193   TLS    Application Data
113 17.641544  194.89.117.193   194.89.117.197   TLS    Application Data, [Unreassembled Packet]
114 17.642446  194.89.117.193   194.89.117.197   TLS    Continuation Data, [Unreassembled Packet]
115 17.642546  194.89.117.197   194.89.117.193   TCP    3631 > https [ACK] Seq=752 Ack=1697 Win=16384 Len=0

⊞ Frame 105 (566 bytes on wire, 566 bytes captured)
⊞ Ethernet II, Src: 00:a0:8e:09:27:d4, Dst: 00:05:4e:44:b6:ac
⊞ Internet Protocol, Src Addr: 194.89.117.193 (194.89.117.193), Dst Addr: 194.89.117.197 (194.89.117.197)
⊞ Transmission Control Protocol, Src Port: https (443), Dst Port: 3631 (3631), Seq: 1, Ack: 106, Len: 512
⊟ Secure Socket Layer
  ⊟ TLS Record Layer: Server Hello
      Content Type: Handshake (22)
      Version: TLS 1.0 (0x0301)
      Length: 74
    ⊟ Handshake Protocol: Server Hello
        Handshake Type: Server Hello (2)
        Length: 70
        Version: TLS 1.0 (0x0301)
        Random.gmt_unix_time: Mar 21, 2005 21:32:22.000000000
        Random.bytes
        Session ID Length: 32
        Session ID (32 bytes)
        Cipher Suite: TLS_RSA_WITH_RC4_128_MD5 (0x0004)
        Compression Method: null (0)
  ⊟ TLS Record Layer: Certificate
      Content Type: Handshake (22)
      Version: TLS 1.0 (0x0301)
      Length: 919
    ⊟ Handshake Protocol: Certificate
        Handshake Type: Certificate (11)
        Length: 915
        Certificates Length: 912
      ⊟ Certificates (912 bytes)
          Certificate Length: 909
  [Unreassembled Packet: SSL]
```


Application Layer Security: HTTPS

- Web servers usually the “trusted” partners
  - Bank, government, insurance, etc.
- Web browsers (clients) usually the customers
  - “Few” web servers for “many” clients

- TLS for connection establishment
  - Web servers authenticated via certificates
  - Scales well as there are not so many
- HTTP Digest authentication via TLS connection
  - Server refuses request and challenges client
  - Client provides credentials typically based upon shared secret
    - Username, password
  - Response does not reveal secret

HTTP Digest Authentication



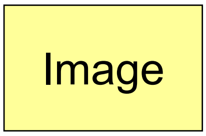
Secure/Multipurpose Internet Mail Extensions (S/MIME)

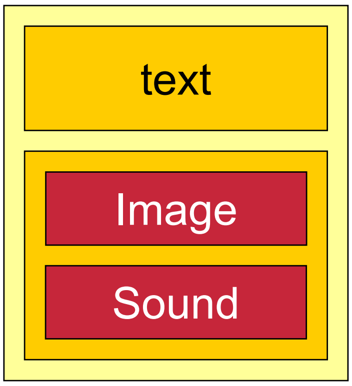
- Example for an *application layer* security mechanism.
- Security services for sending and receiving MIME data
  - Authentication, message integrity, non-Repudiation of origin, confidentiality
- Not restricted to e-mail
  - May be used with HTTP and other application protocols

- Relies on the *Cryptographic Message Syntax* (CMS)
  - Specification for authenticating, digitally signing and encrypting contents
  - Security functions are applied on *envelopes* containing message content (or other envelopes)
  - May embed certificates and other key management information
- Uses public key cryptography and certificates
  - Most useful with a PKI (e.g., CA hierarchy)

A Brief Excursion to MIME

- Multipurpose Internet Mail Extensions
  - Not just mail: used with HTTP and many other application protocols
- Define the purpose of a piece of content (in a message body)
  - Type, encodings
  - Intended interpretation
  - Specify additional parameters
  - Allow for references
- Allow for multipart contents
  - Arbitrarily nested pieces of contents
  - Specify the above for each part individually





A Brief Excursion to MIME

```
Content-Type: image/jpeg
Content-Length: 5489
Content-Transfer-Encoding: base64
Content-ID:  42
Content-Description: an image of a tree
Content-Disposition: attachment; ...
```

…cation protocols

…message body)

- Allow for references

Allow …

```
Content-Type: multipart/alternative
Content-Type: multipart/mixed
Context-Type: multipart/related
```

text

S/MIME Processing Steps

- S/MIME messages are combination of MIME bodies and CMS content types
  - Several MIME types as well as several CMS content types are used
- Data to be secured is always a canonical MIME entity
  - Secure MIME entity: sub-part, sub-parts, or an entire MIME entity
  - Canonicalization depends on the respective content type
- Optionally, apply transfer encoding
- MIME entity and security information as input to CMS processing
  - Will result in a CMS object
- CMS object is wrapped in MIME

Resulting S/MIME Messages

- Content-type: application/pkcs7-mime
  - PKCS#7: Cryptographic Message Syntax (CMS) Version 1.5 (RFC2315)
    - Current spec of CMS: RFC 3852; ASN.1-encoded cryptographic objects
  - ;smimetype= parameter to indicate what the object is used for
    - enveloped-data, signed-data, compressed-data
    - Concatenation possible: signed-encrypted-data
  - ;name=filename.suffix and ;filename=filename.suffix
    - application/pkcs7-mime: SignedData + EnvelopedData -> .p7m
    - application/pkcs7-mime: CompressedData -> .p7z
    - application/pkcs7-signature: SignedData -> .p7s
    - Filename is arbitrary; no longer than 8 characters
- Signing only: multipart/signed
  - Does not require S/MIME software on the receiver for viewing the contents

Sample Encoding: Encryption

```
Content-Type: application/pkcs7-mime; smime-type=enveloped-data;
            name=smime.p7m
Content-Transfer-Encoding: base64
Content-Disposition: attachment; filename=smime.p7m
```

rfvbnj756tbBghyHhHUujhJhjH77n8HHGT9HG4VQpfyF467GhIGfHfYT6
7n8HHGghyHhHUujhJh4VQpfyF467GhIGfHfYGTrfvbnjT6jH7756tbB9H
f8HHGTrfvhJhjH776tbB9HG4VQbnj7567GhIGfHfYT6ghyHhHUujpfyF4
0GhIGfHfQbnj756YT64V

Sample Encoding: Signing

```
Content-Type: multipart/signed;
          protocol="application/pkcs7-signature";
          micalg=sha1; boundary=boundary42

--boundary42
Content-Type: text/plain

This is a clear-signed message.

--boundary42
Content-Type: application/pkcs7-signature; name=smime.p7s
Content-Transfer-Encoding: base64
Content-Disposition: attachment; filename=smime.p7s

ghyHhHUujhJhjH77n8HHGTrfvbnj756tbB9HG4VQpfyF467GhIGfHfYT6
4VQpfyF467GhIGfHfYT6jH77n8HHGghyHhHUujhJh756tbB9HGTrfvbnj
n8HHGTrfvhJhjH776tbB9HG4VQbnj7567GhIGfHfYT6ghyHhHUujpfyF4
7GhIGfHfYT64VQbnj756

--boundary42--
```

Security

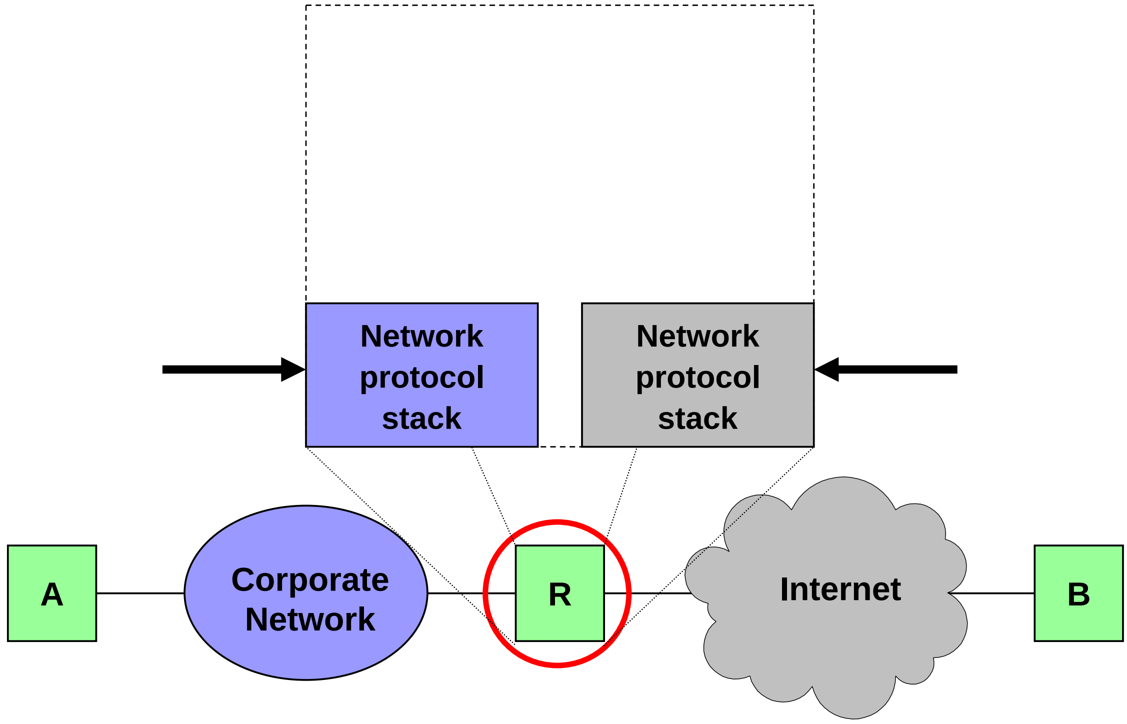
- Background and security properties
- Cryptographic algorithms
- Security mechanisms
- Security protocols and systems
- Security Devices

HELSINKI UNIVERISITY OF TECHNOLOGY
NETWORKING LABORATORY

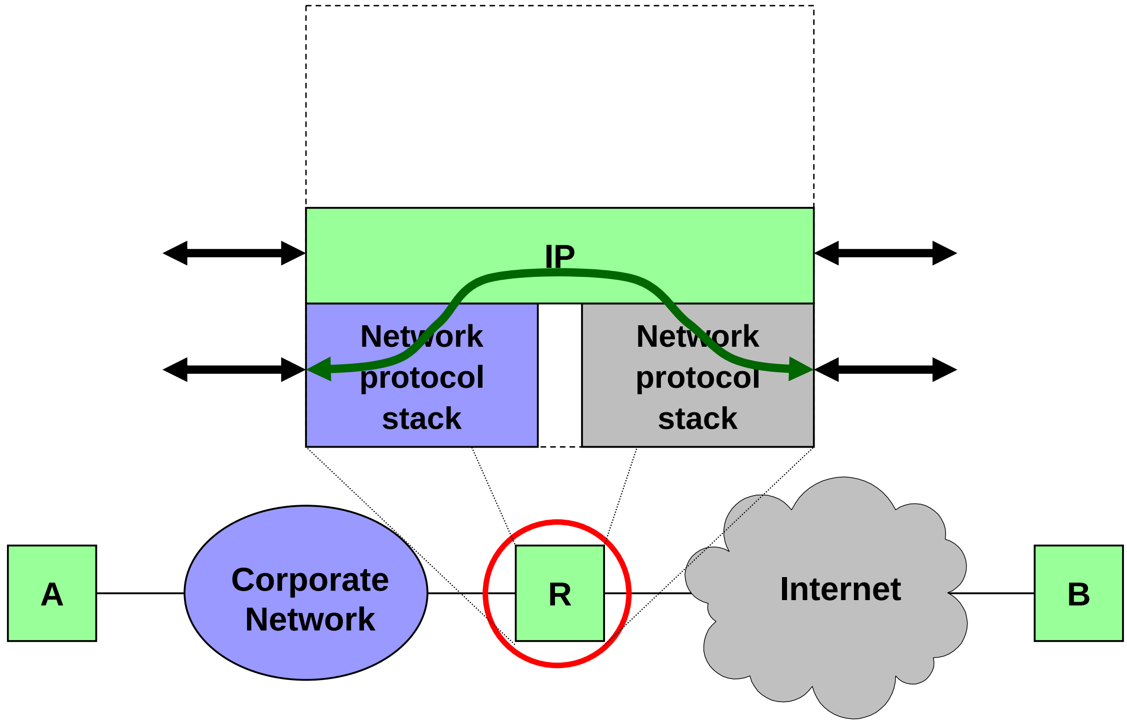
"Security Devices" for IP Networks

- Some protection against attacks from the outside
- Packet Filter
  - (dis)allow forwarding of packets to/from certain addresses
  - Protect networks from stray traffic
- Application Layer Gateway (ALG) / Proxy
  - control (and police) communications at application layer
- Firewall
  - Combination of the above
  - protect internal resources against access from the outside
- Network Address Translator (NAT)
  - minimize required fraction of "Internet" address space
  - hide internal IP addresses
  - perform packet filtering for unknown traffic

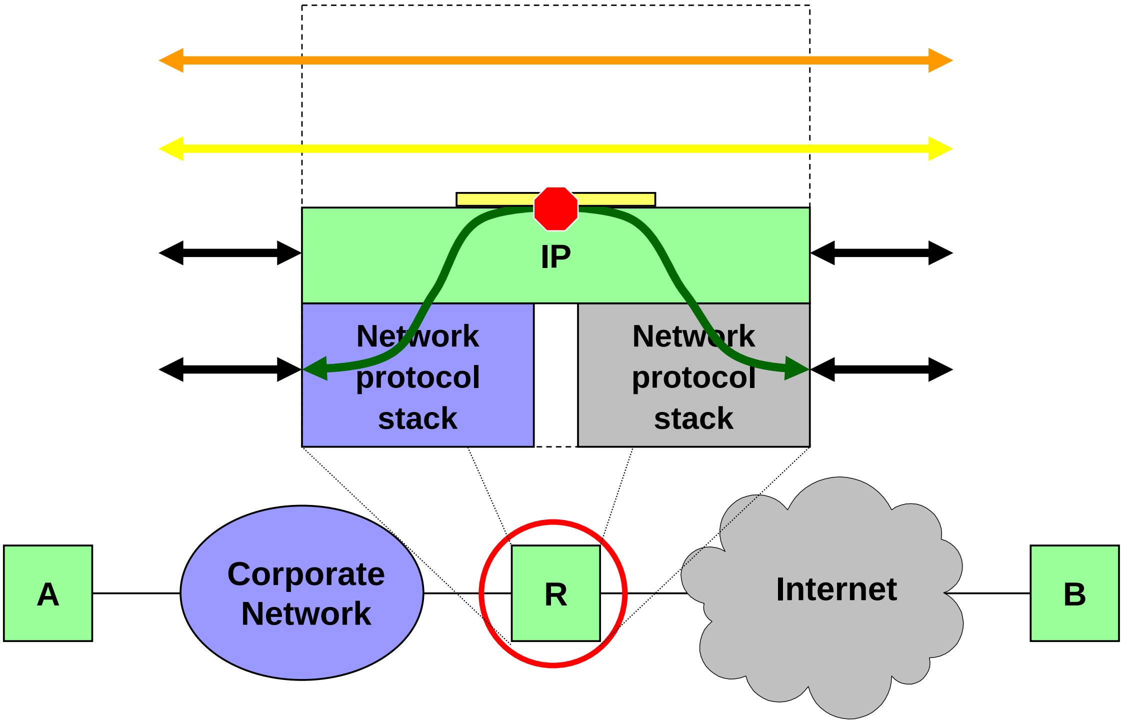
Coupling Networks with Security in Mind



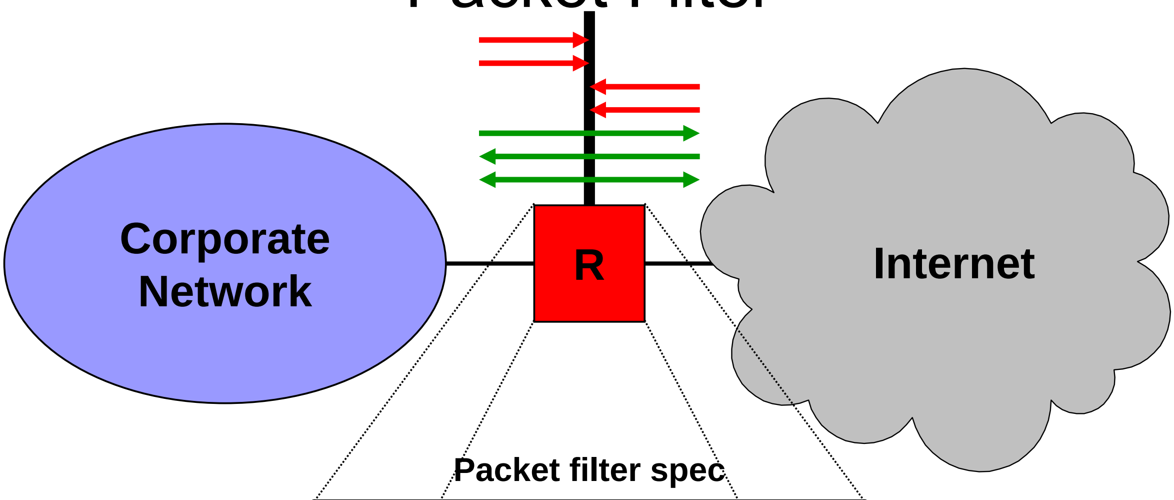
IP Layer: Router w/o Security



IP Layer: Packet Filter



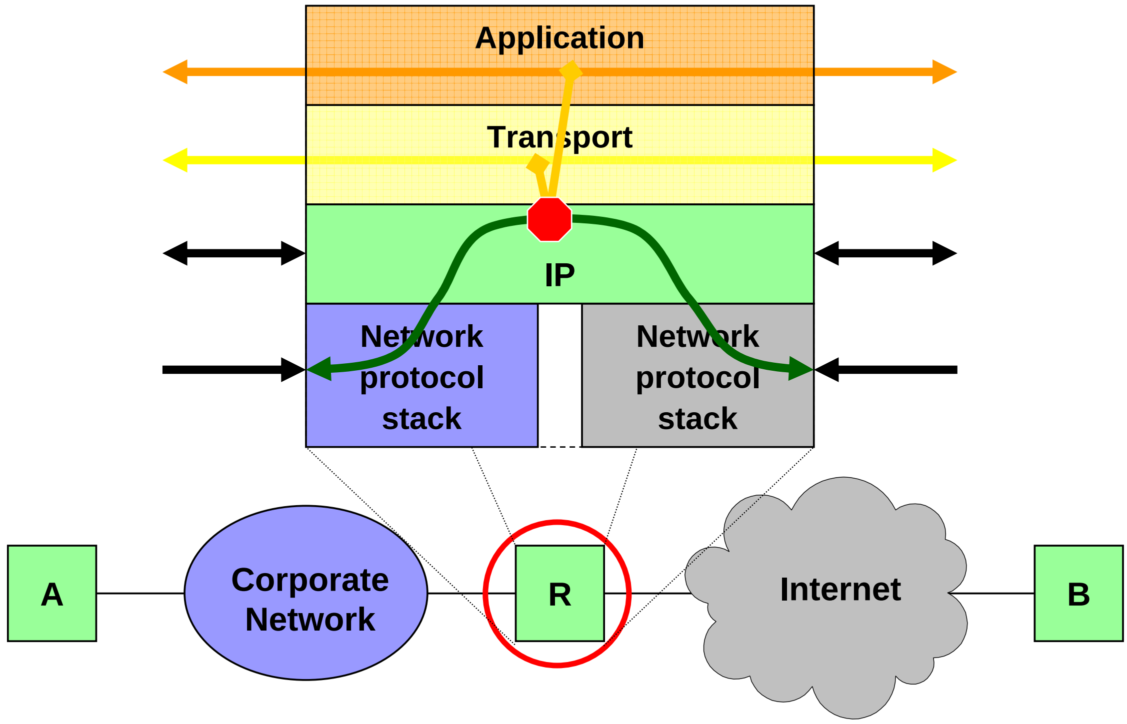
Packet Filter



- Source, destination IP address
- Protocol (UDP, TCP, ICMP)
- Source, destination port
- Direction of traffic

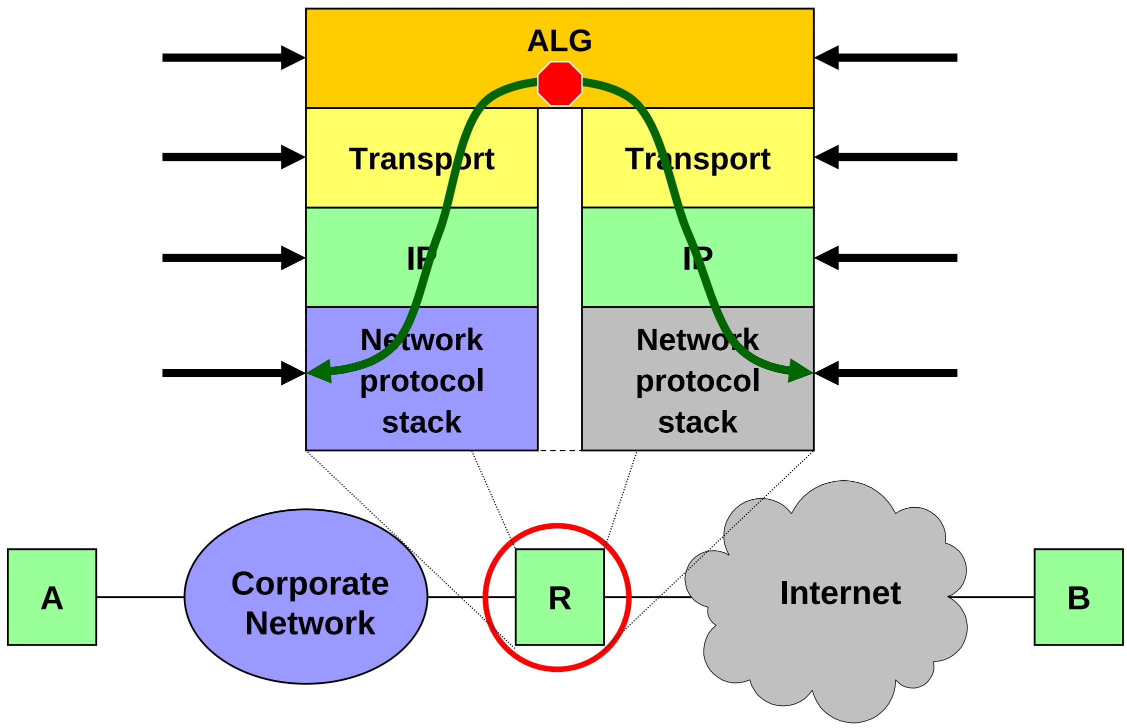
May be dynamically configured.

Stateful Packet Inspection

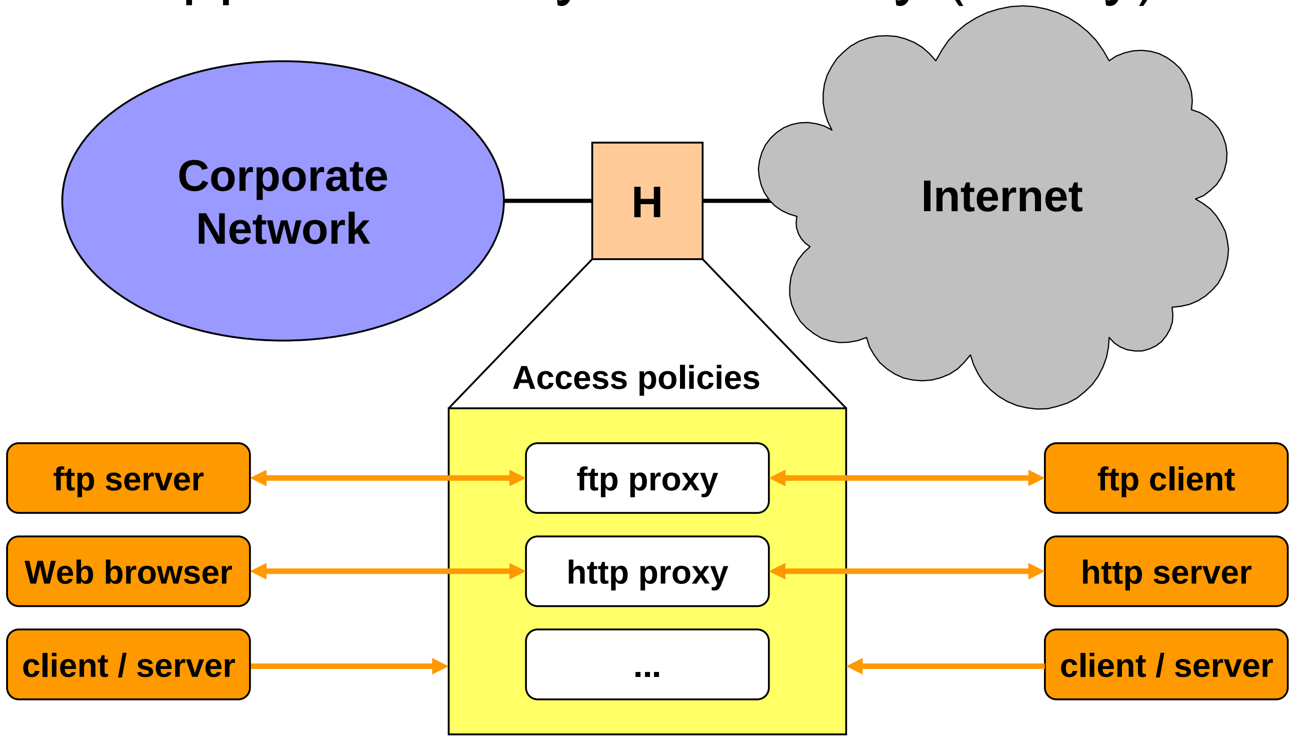


HELSINKI UNIVERISITY OF TECHNOLOGY
NETWORKING LABORATORY

Application Layer Gateway



Application Layer Gateway (Proxy)



HELSINKI UNIVERISITY OF TECHNOLOGY
NETWORKING LABORATORY

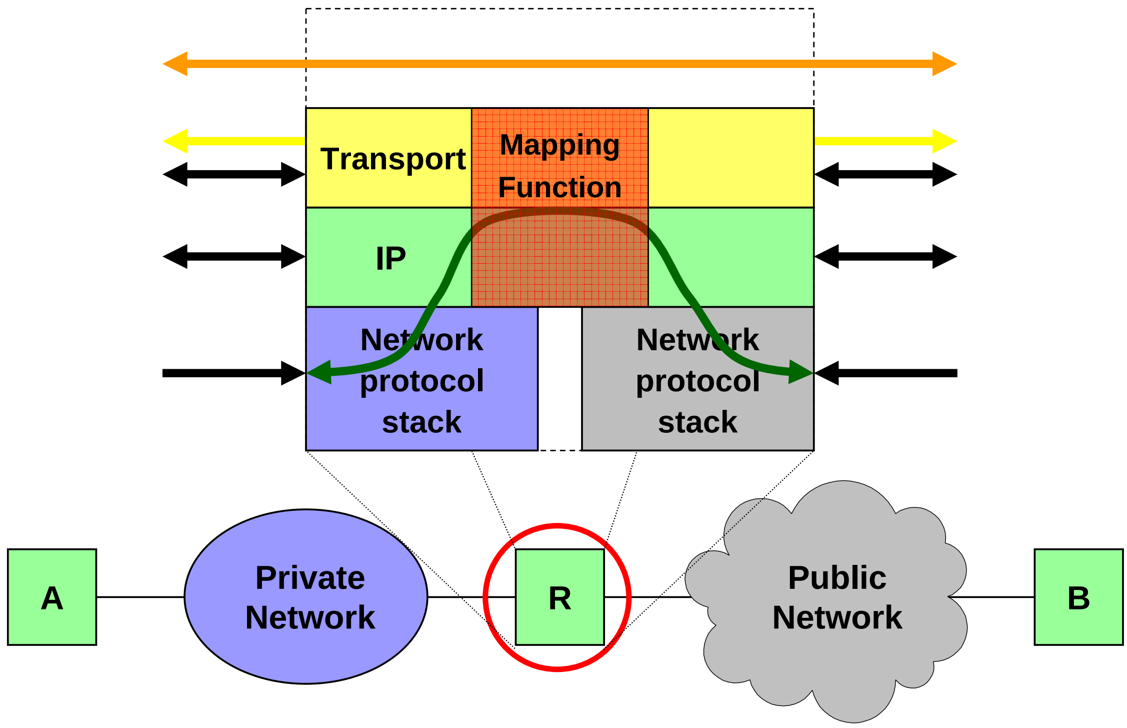
Firewalls

- Packet filters, enforcing packet altering/forwarding policies
  - Filter specification: Usually statically configured
  - Most configurations disallow packets for “non-standard ports”
- Stateful packet inspection
  - Detect transport or application context of packets
  - Dynamically adapt filter specification
- Application layer gateways
  - Terminate connections: act as transparent or explicitly visible proxies
  - Monitor connection: parse contents of application protocols
    - Functioning precludes end-to-end security!
  - Dynamically adapt filter specification
- Policies may be applied at all layers

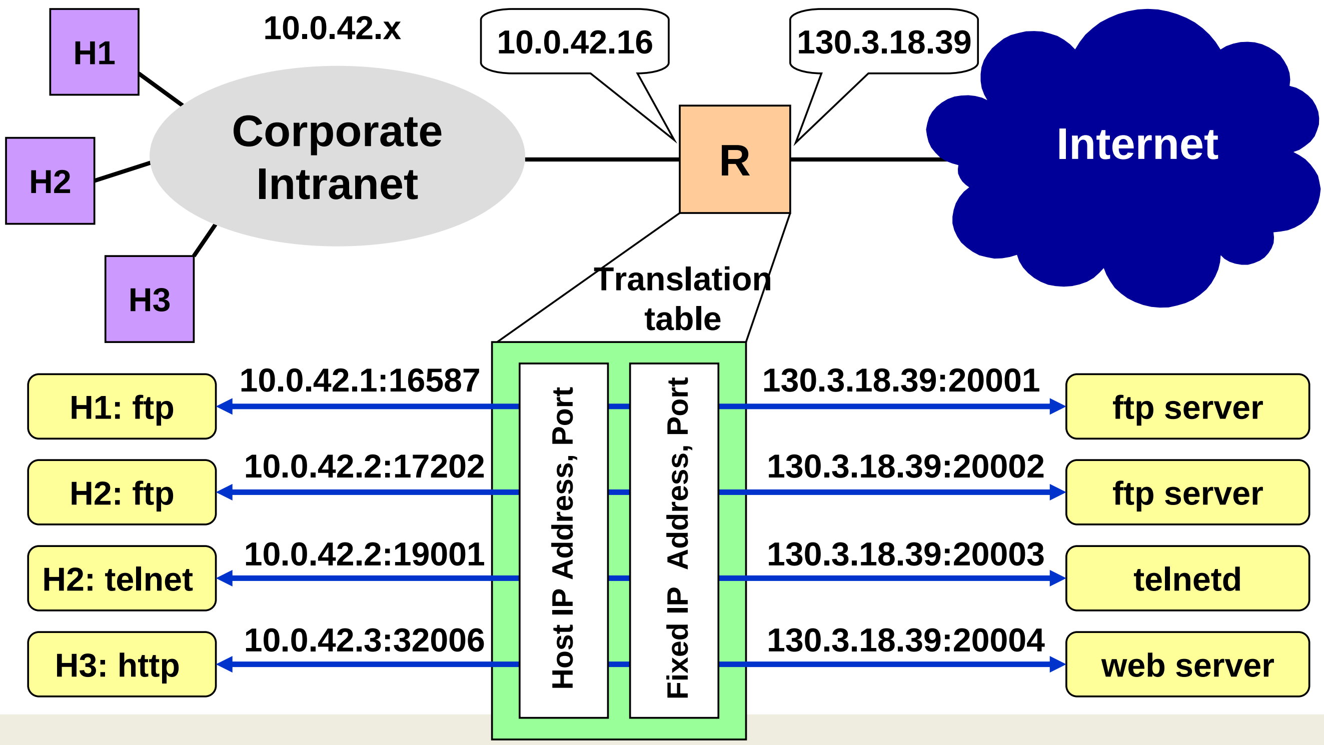
Network Address Translators

- Intermediate systems that can translate addresses (and port numbers) in IP packets
  - Often used to map global addresses to address/port number combination of hosts in a corporate network
- Different motivations
  - Efficient usage of address space
    - Share one globally unique address
    - Use a private address space in the enterprise (10.x.x.x, 192.168.x.x, …)
  - Security
    - Make internal host inaccessible from the public Internet
    - Hide addresses / address structure
- Include dynamically configured packet filters, stateful packet inspection

Network (+Port) Address Translators (NAT)



Network Address Translators



Firewall Applicability

- Firewalls and NATs help against unwanted traffic from the outside
  - Denial-of-Service attacks, port scans, break-in attacks, worms
  - ALGs against viruses
- But: Firewalls and NATs may also prevent legitimate traffic
  - Evil to IP communications: Break end-to-end model
  - Have many implicit assumptions about protocols
  - Do not work well with a number of protocols
    - Including their security features
- Just one piece in a security portfolio, to be applied wisely
- Applications and protocols still need security
- Users and their behavior still pose a significant risk