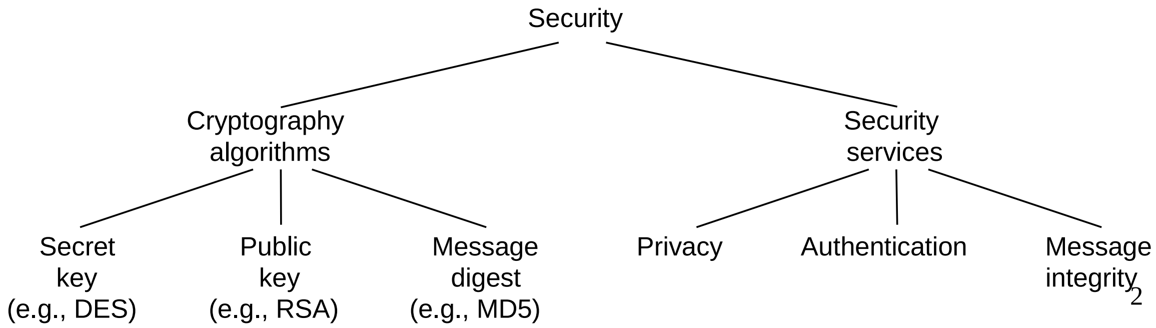
Security

Security services and cryptography

- Security services
  - Privacy: preventing unauthorized release of information
  - Authentication: verifying the identity of the remote participant
  - Message integrity: making sure that message has not been altered
- Cryptographic algorithms are used as fundamental building blocks
  - common algorithms: Data Encryption Standard (DES), Rivest, Shamir, and Adleman (RSA), Message Digest 5 (MD5)
  - most algorithms rely on the use of a secret key ⇒ key distribution problem
- Security services are implemented by using secure protocols
  - PGP, HTTPS, IPSec, …



Secure systems

- To build a secure system you need the right combination of algorithms and protocols + something that technology/science can not solve!
  - To implement privacy, authentication and integrity services, a number of protocols and algorithms are used
  - Even though you have the best protocols money can buy, there's always the human factor
    - one can get "forgotten" passwords by just calling local help desk
    - any kind of inside information (spying) helps in breaking security

$\Rightarrow$ Protocols and cryptography only solve some of the problems

$\Rightarrow$ Appropriate security policies and working processes are needed to achieve "full" security

- Here we only look at the technology part of security (cryptography and protocols)

Outline

- Cryptographic algorithms
- Security mechanisms
  - Authentication protocols
  - Message integrity protocols
  - Key distribution
- Secure protocols and systems
- Firewalls, security attacks

Cryptographic algorithms

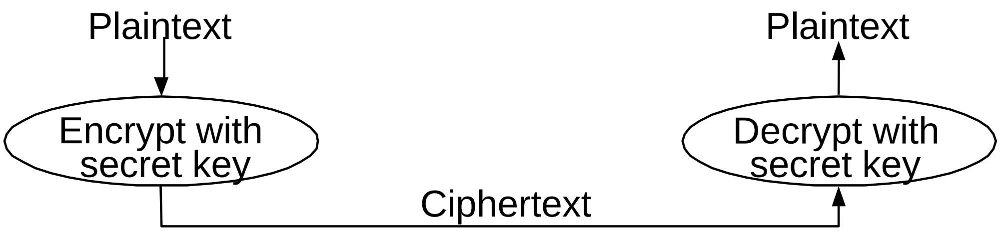
- Secret key algorithms
  - symmetric, both participants share a single key
- Public key algorithms
  - private key (not to be shared) and public key (published to everyone)
  - encrypt with public key and decrypt with private key
- Hash or message digest algorithms
  - no keys, think of as “cryptographic checksum” of a message
  - protects the receiver from malicious changes to the message (message integrity)

Requirements for algorithms

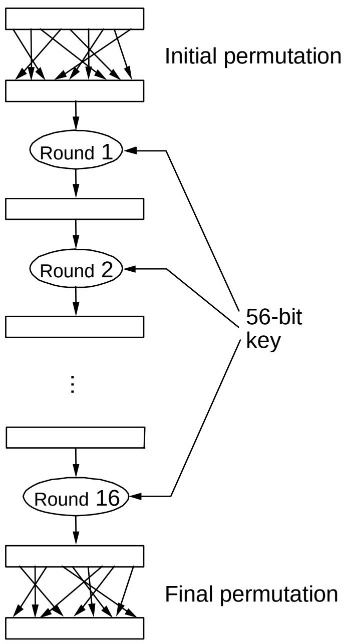
- Algorithm itself is known, only the key is secret
  - need to know why the algorithm works
    - algorithm unbreakable until somebody breaks it and announces it $\Rightarrow$ no news is good news (should not change algorithm very often)
  - key distribution/management becomes a problem
- Breaking the algorithm is easier if there is additional information available
  - be prepared for “known plaintext” or “chosen plaintext” attacks
  - bad keys are easier to break
    - security hole in a www browser: a combination made from process ID and time of day as a seed to generate a random number used for key calculation
- Best algorithms: “impossible” to find the key even if the plaintext and the ciphertext (=encrypted plaintext) are known
  - “impossible” = searching the key space takes simply too long
- For message digest algorithms: one-way functions, given the output it is computationally infeasible to find the corresponding input
  - note: usually produces a short output from a long message input (so not one-to-one, but many-to-one)
  - message digest algorithms should be fast to compute

Data Encryption Standard (DES)

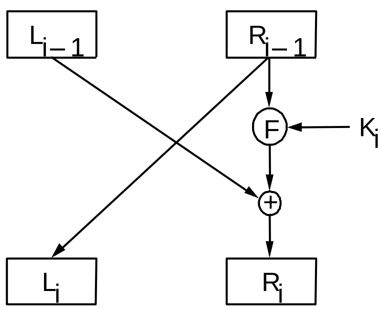
- Encrypts a 64-bit block with a 64-bit key (actually 56 bits are useful, 8 parity bits)
- Complicated algorithms, several stages
  - uses “diffusion and confusion”
  - design principles of DES are not public knowledge
  - no published mathematical proof that DES is secure
  - designed such that none of the structure of original text is left in the ciphertext ⇒ attacker must try out all possible key combinations
    - use long enough key and make single DES encyption/decryption process computationally expensive enough
- Nowadays, basic DES considered only marginally secure
  - key can be found in a “reasonable” time with powerful parallel computing
  - triple-DES: encrypt data three times (just first-aid or a real solution?)
  - AES (Advance Encryption Standard): new secret key algorithm (128, 192, 256 bit keys)



Diffusion and confusion in DES



- DES has 3 phases:
  - 64 bits in the block are permuted
  - 16 rounds of an identical operation are applied to the resulting data and key
  - inverse of the original permutation applied to the result
- Operation on each round:
  - (L_i, R_i) = left/right-most 16 bits
  - K_i = i^{th} 48 bit subset of original key K
  - F = (complex) transformation operation



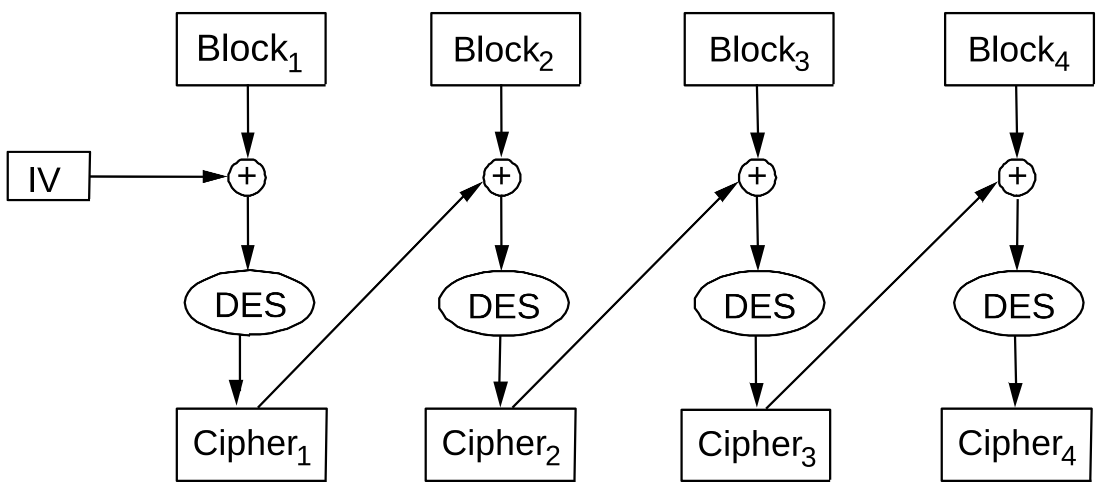
Diffusion and confusion in DES (cont.)

- Operations in DES algorithm
  - XOR operations
  - permutations, selections
  - expanding
  - all in all, simple bit operations repeated over and over…. hard to get a picture of the complete algorithm and why it works (and there is no formal proof that it works…)

- DES does not distinguish between encryption and decryption - only difference is that keys in 16 rounds are applied in a reverse order

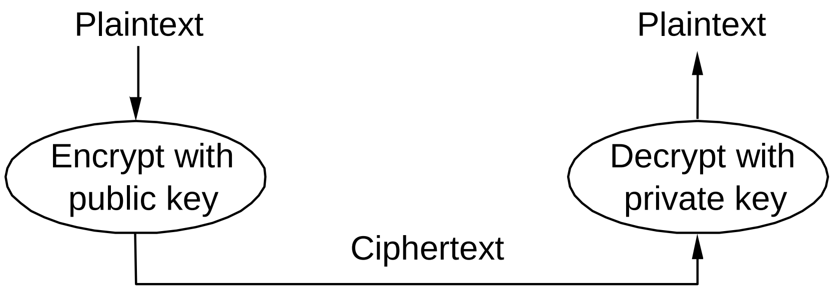
DES for long messages

- Cipher Block Chaining (CBC): Ciphertext for block i is XORed with the plaintext for block i+1 before running through DES
  - initialization vector (IV) needed for the first block
  - random number sent along with the “initial” message



RSA

- Encryption with public key, decryption with private key
- Grounded in number theory and computational complexity of factoring two large primes (that are needed to find the key)
- Simple formulas, only a few steps (but not fast to calculate)
  - computationally much more complex than DES
- First broken in 1994 (competition announced in 1977)
  - only 17 years after introduction (RSA initially believed virtually unbreakable)
  - massive parallel processing and efficient factorization algorithms



RSA (cont)

- Choose two large prime numbers p and q (each 256 bits)

$$n = p \times q$$

- Choose encryption key e, such that e and (p - 1) x (q - 1) are relatively prime
  - two numbers are relatively prime if they have no common factor greater than one
- Compute decryption key d such that

$$d = e^{-1} \bmod ((p - 1) \times (q - 1))$$

- Construct **public key** as (e, n), and **private key** as (d, n)

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

- Discard (do not disclose) original primes p and q

Simple RSA example

- Computing public and private key
  - we pick primes p=7 and q=11 (in real encryption you pick LARGE primes)
  - multiply the primes, n=7 x 11= 77 and also (p-1) x (q-1) = 60
  - pick e that is relatively prime to 60 ⇒ take e=7
  - d= 7^{-1} mod 60, i.e., 7 x d = 1 mod 60 ⇒ one solution is d=43
  - public key is (e,n)=(7,77) and private key (d,n)=(43,77)

- Ready to encrypt:
  - let's encrypt message m=9
  - encrypted message: $c = m^e \bmod n = 9^7 \bmod 77 = 37$.

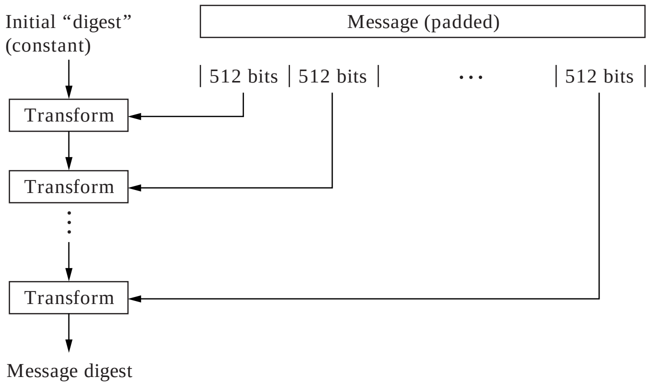
- Decryption:
  - decrypted message: $m = c^d \bmod n = 37^{43} \bmod 77 = 9$.

Message Digest

- Usually faster to compute than DES or RSA
- Usually don't have a formal mathematical foundation, rely on complexity of the algorithm (like DES)
- Cryptographic checksum
  - just as a regular checksum protects the receiver from accidental changes to the message, a cryptographic checksum protects the receiver from malicious changes to the message
- One-way function
  - given a cryptographic checksum for a message, it is virtually impossible to figure out what message produced that checksum
  - in other words, it is not computationally feasible to find two messages that hash to the same cryptographic checksum
- Relevance
  - if you are given a checksum for a message and you are able to compute exactly the same checksum for that message, then it is highly likely this message produced the checksum you were given (message integrity)

Message Digest Algorithms

- Commonly used MD4, MD5, SHA
- Basic operation in MD5
  - transformations in 512 byte chunks until whole message is handled
  - at each transformation: input = current value of 128-bit digest and 512 bits of message, output = new 128-bit digest
  - each transformation: 4 different sets of operations
    - operations: bitwise OR, AND, NOT, XOR, addition and rotation



Outline

- Cryptographic algorithms
- Security mechanisms
  - Authentication protocols
  - Message integrity protocols
  - Key distribution
- Secure protocols and systems
- Firewalls, security attacks

Security mechanisms

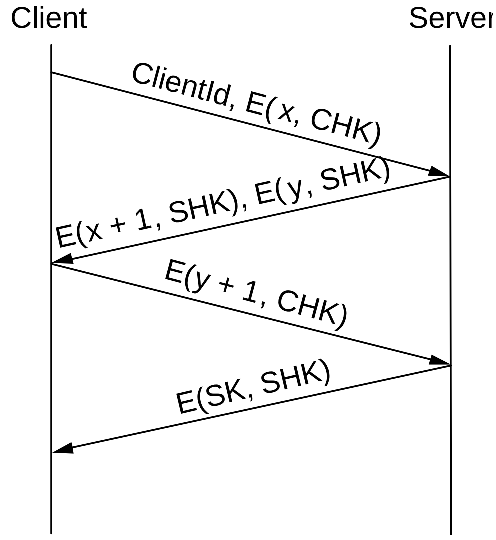
- Security mechanisms needed for
  - authentication of participants
  - assuring the integrity of messages
  - distributing public keys
- Remarks about algorithms:
  - DES and MD5 much faster than RSA when implemented in software
  - RSA too slow for encrypting data messages - instead used to deliver the most valuable part of the data, i.e., signature or secret key
  - hybrid algorithms, combinations of different algorithms for different tasks

Authentication Protocols

- Establish identity of the participants (server ⇔ client)
  - first step in secure communications
- 3 approaches:
  - three way handshake
  - trusted 3rd party
  - public key auhentication
- Need to establish Session Key (SK) to be used during further communication
  - using SK limits the number of messages actually encrypted with actual client/server secret keys ⇒ harder for attacker to gather data to determine the key

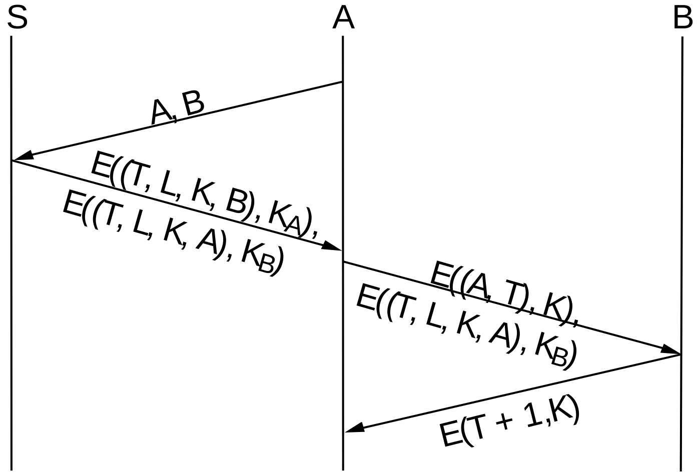
Three way handshake

- Three-way handshake
  - participants already share a secret key
  - E(m,k) = encryption of message m with key k
  - D(m,k) = decryption of message m with key k
  - x, y = random numbers, CHK = client handshake key, SK = session key, SHK = server handshake key = CHK (at least should be)
  - 1. Send ClientId and encrypted msg.
    2. Server checks ClientId for corresponding SHK.
    3. If client receives msg x+1 decrypted with CHK, server authenticated.
    4. Encrypt y+1 with CHK.
    5. If server receives msg y+1 decrypted with SHK, then client authenticated.
    6. Server sends SK to client.
  - Where does CHK (or SHK) come in the first place?
    - ex. obtained from user password via transformation



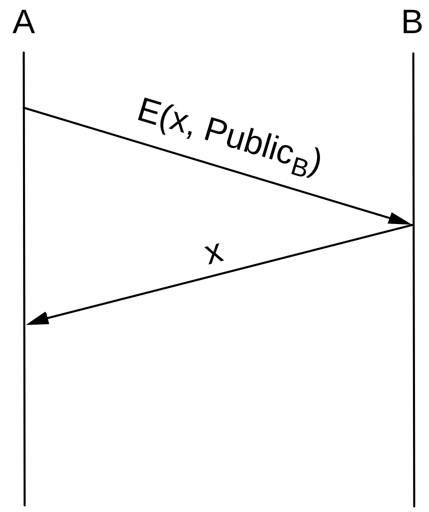
Trusted Third Party (Kerberos)

- Participants A and B both trust on S (authentication server)
- A and B share a secret key with S
- T=timestamp (like random number in 3-way handshake), L=lifetime (limits the life time of K), K=new session key



Public key authentication

- Nice feature: two sides need not share a secret key!
- A uses B’s public key, B decrypts using corresponding secret key and returns x ⇒ B is autheticated
  - A can authenticate itself in the same way



Message integrity protocols

- Setting:
  - participants do not care if some third party can read their messages, but want to be sure that messages DO come from the source they claim
- Digital signature using RSA
  - special case of a message integrity where the code can only have been generated by one participant
  - compute signature with private key, receiver verifies with sender’s public key (inverse use of RSA than in privacy)
  - inefficient because RSA is slow (encryption with private key as slow as RSA)
- Use of just MD5 not enough for integrity (imposter can send messages and apply MD5 on that)
  - to implement integrity, MD5 must be combined with some keyed cryptography
  - 2 approaches Keyed MD5 and MD5 with RSA signature
  - both approaches overcome RSA’s performance problems

Message integrity protocols (cont)

- Keyed MD5 with public key cryptography:
  - m = message, k = random key
  - sender: m + MD5(m + k) + E(k, private)
  - receiver
    - recovers random key, k, using the sender's public key
    - applies MD5 to the concatenation of m+k, OK if result equals received check sum
- MD5 with RSA signature
  - sender: m + E(MD5(m), private)
  - receiver
    - decrypts signature with sender's public key to get MD5 check sum
    - compares result with MD5 applied to m

Public key distribution

- How does A learn about B's public key?
  - ITU-T solution X.509
  - adapted to Internet by IETF Public Key Infrastrucrure Working Group (PKIX)
- Certificate
  - special type of digitally signed document:
    - "I certify that the public key in this document belongs to the entity named in this document, signed X."
  - contains:
    - name of the entity being certified
    - public key of the entity
    - name of the certified authority
    - a digital signature (see slide 22)
- Certificates do not solve the key distribution problem
  - certificate is useless, unless you trust the entity that provided the certificate and produced the signature

Key Distribution (cont)

- Certified Authority (CA)
  - administrative entity that issues certificates
  - useful only to someone that already holds the CA's public key
- Chain of trust
  - if X certifies that a certain public key belongs to Y, and Y certifies that another public key belongs to Z, then there exists a chain of certificates from X to Z
  - someone that wants to verify Z's public key has to know X's public key and follow the chain
  - here X is the root CA and its public key must be "well known"
  - Internet root CA called IPRA (Internet Policy Registration Authority)
- Note! Possession of a certificate says nothing about your identity
  - to prove who you are, you need to demonstrate that you have the private key that corresponds to the public key in the certificate (authentication!)
- Certificate Revocation List (CRL)
  - your certificate must be cancelled if somebody has obtained your private key
  - CRL = digitally signed list of certificates that have been revoked
  - periodically updated and publicly available (posted on bulleting board)
  - certificates have expiration dates

Outline

- Cryptographic algorithms
- Security mechanisms
  - Authentication protocols
  - Message integrity protocols
  - Key distribution
- Secure protocols and systems
- Firewalls, security attacks

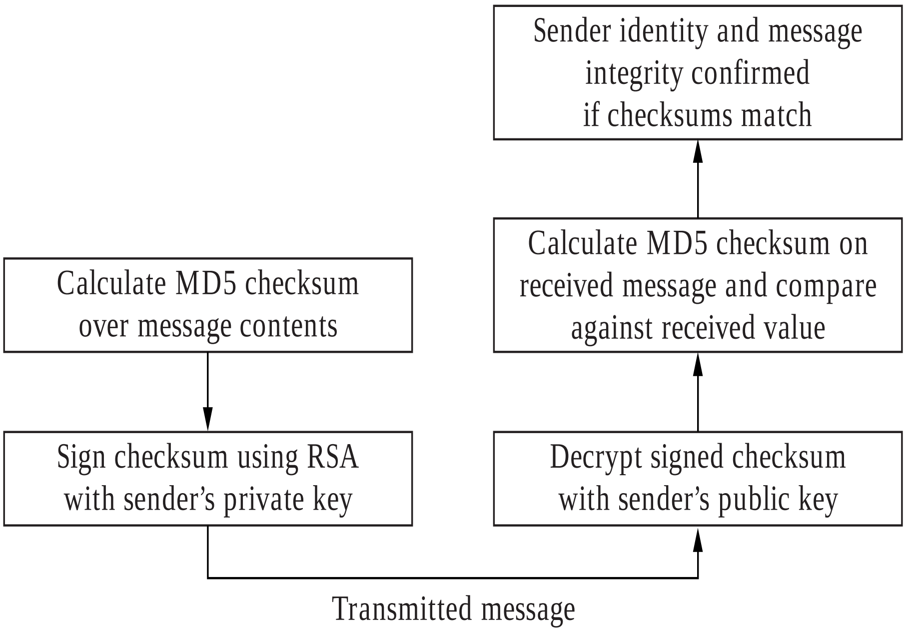
Some example systems

- Components of a secure system
  - Cryptographic algorithms
  - Authentication protocols
  - Key distribution mechanisms
- Systems that use these components can be categorized by the protocol layer at which they operate
  - Application level: secure e-mailing (PEM, PGP)
  - Transport level: TLS, HTTPS
  - Network level: IPSec

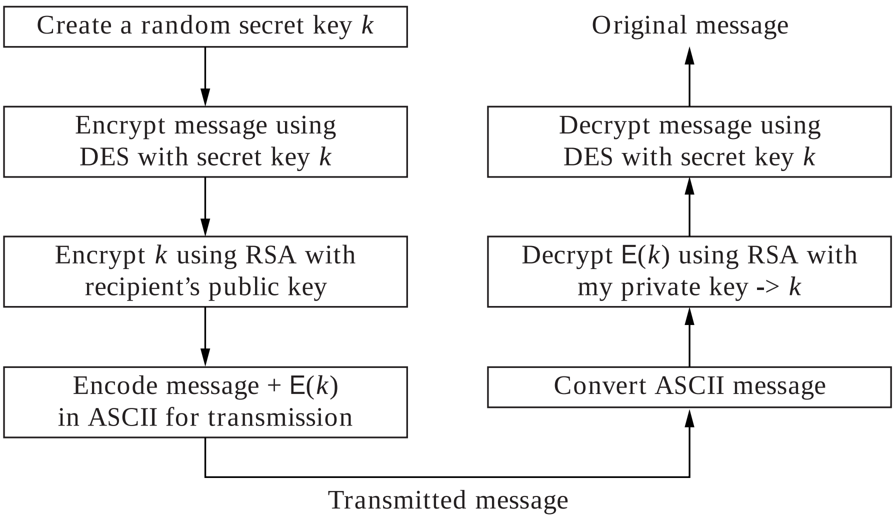
Privacy Enhanced Mail (PEM)

- Set of 4 RFCs that specify
  - format of the PEM message
  - hierarchy of certification authorities
  - set of cryptographic algorithms to be used
  - message formats for requesting and revoking certificates
- General challenges when securing email
  - most mail systems take only ASCII characters (cryptographic algorithms usually output binary data)
  - line breaks may destroy the message digest
  - handling mailing lists (mails sent to many receivers)
- PEM certification hierarchy: tree-structured hierarchy of CAs
  - need trust from one CA to another (chain of trust)

PEM message integrity and authentication



PEM Message Encryption



- Mail list problem: not whole message, but only k (which is short) is encrypted with each recipient's public key

PEM message

- Security operations given in header (authenticated, encrypted, both)
  - MIC = message integrity code

----BEGIN PRIVACY-ENHANCED MESSAGE----
PEM header; includes mode (MIC-CLEAR, MIC-ONLY, ENCRYPTED)
Initialization vector for DES-CBC
Certificate of sender (signed by sender's CA)
Certificate of sender's CA (signed by next level CA)
⋮
Certificate of PCA signed by IPRA
Message integrity code
Per-message key, encrypted with recipient's public key
Message body (clear, encrypted, or encoded)
----END PRIVACY-ENHANCED MESSAGE----

- Problem: complicated certification hierarchy needed

Pretty Good Privacy (PGP)

- Encryption and authentication for email
- Arbitrarily meshed certificates allowed (compare: strict hierarchy in PEM)
  - certificates collected,e.g., at IETF PGP key-signing parties
  - allows each user to decide for themselves how much trust to place on given certificate
  - user will collect a set of certificates (stored in key ring -file)
- Encryption of message similar to PEM
  - allows a wide variety of different cryptographic algorithms - algorithm used specified in the header
  - allows user to list his favorite cryptographic algorithm in the key ring – file
- Decryption
  - PGP's key management software used to find sender's public key
  - if checksum OK, PGP tells the level of trust of the (used) public key based on number of certificates for sender and how trustworthy the signatures are

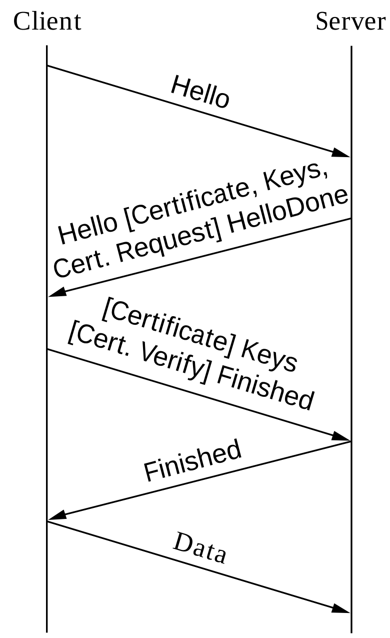
Transport Layer Security (TLS, SSL, HTTPS)

- What can happen when making a credit card purchase in the Internet?
  - Information can be intercepted in transit and used later to make unauthorized purchases
  - details of transaction can be modified
  - to whom did you actually send your credit card information
  - ⇒ Need for PRIVACY, INTEGRITY and AUTHENTICATION
- Solution: a general-purpose protocol that sits between the application protocol and the transport protocol, called “transport layer security”
  - TLS = Transport Layer Security, RFC2246
    - previously SSL (Secure Socket Layer)
    - defines protocols to achieve transport layer security
  - HTTPS = SSL-protected HTTP transfer; uses port 443 (instead of HTTP's normal port 80), and is identified with a special URL method “https”
  - offers a secure and realiable byte stream

Application (e.g., HTTP)
Secure transport layer
TCP
IP
Subnet

TLS (Transport Layer Security)

- Difference between TLS protocol and secure email: TLS allows real-time negotiation
- TLS broken into two parts:
  - handshake, used to negotiate parameters
  - a “record” protocol, used for the actual data transfer
- In handshake: agree on cryptographic algorithms (& session keys, initial vectors etc.) and compression algorithm (if needed), exchange certificates, …
- Handshake takes > 2 RTTs and up to dozen messages
  - in picture: [optional message]
- Record protocol performs fragmentation, integrity protection, encryption ⇒ to lower layer (TCP)



TLS (cont)

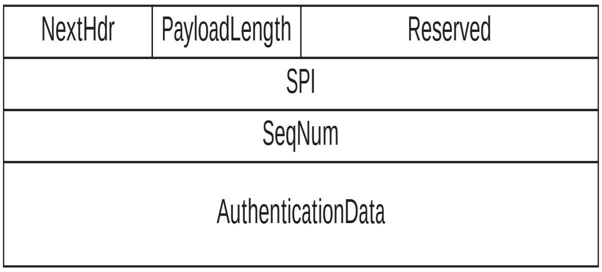
- Ability to negotiate cryptographic algorithms ⇒ “man-in-the- middle” attacks are possible
  - initial negotiation of algorithms not secure ⇒ intermediary can change the choice of algorithms into weaker ones
  - well-designed algorithm aborts the transaction if protection is not strong enough (attacks becomes “denial-of-service”)
- Ability to “resume” sessions
  - recall that handshake takes a long time
  - client includes the session ID from a previous session in initial handshake message
  - if server still has that session ID in cache, session can resume, otherwise need new session initialization
  - useful in web transactions over HTTPS
- Does not specify any particular key infrastructure (unlike PEM and PGP)

IP Security (IPSEC)

- A framework (instead of a single protocol) for providing all security services (privacy, integrity, authentication)
  - highly modular (system administrator can select suitable protocols and systems)
  - provides a large menu of security services
  - allows users to control granularity with which security services are applied
    - protect “narrow” (packets between two hosts) or “wide” (packets between two routers) streams
- Consists of 2 parts
  - protocols that implement the available security services
    - Authentication Header (AH)
    - Encepsulating Security Payload (ESP)
  - support for key management
    - ISAKMP = Internet Security Association and Key Management Protocol
    - defines procedures to establish, negotiate, modify and delete SAs
- SA (Security Association)
  - one-way “connection” that is protected by the security services
  - SA association identified by assigned SPI and host IP address

IPSEC Authentication Header (AH)

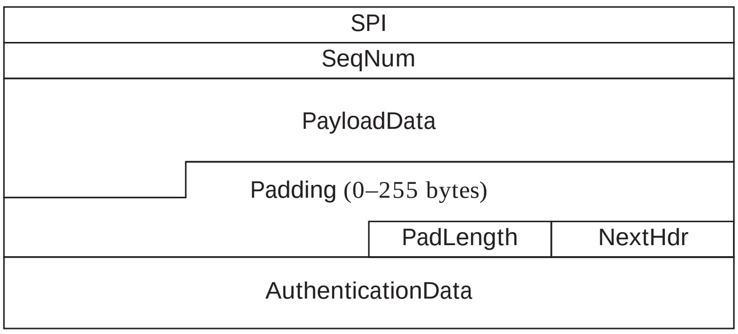
- Provides connectionless integrity and data origin authentication
- Either follows IPv4 header or is an IPv6 extension header



  - NextHdr=type of next payload after AH
  - Reserved=for future use, 0 now
  - SPI=security parameters index,
  - SeqNum=increasing counter, protection against replay
  - AuthenticationData=message integrity code for this packet

Encepsulating Security Payload (ESP)

- Designed to provide a mix of security services in IPv4 or IPv6.
  - can be applied alone, or with AH
  - ESP header inserted after IP header and before upper-layer protocol (between a pair of hosts) OR before an encapsulated IP header (tunnel between a pair of security gateways)
  - provides confidentiality, data origin authentication, connectionless integrity, and antireplay service
- A popular way to use ESP is to build an “IPSEC tunnel” between two routers

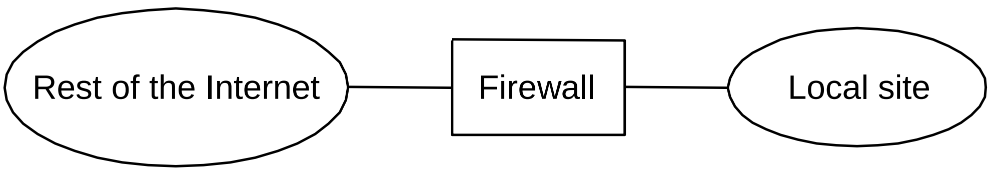


Outline

- Cryptographic algorithms
- Security mechanisms
  – Authentication protocols
  – Message integrity protocols
  – Key distribution
- Secure protocols and systems
- Firewalls, security attacks

Firewalls

- Firewall = specially programmed router that sits between a site and the rest of the network
- Actions
  – forwards packets
  – filters packets (e.g., based on source IP address, to prevent "denial-of-service" attack)
- Why needed?
  – security mechanisms are not widely deployed
  – allows the system administrator to implement a security policy in one centralized place (end-to-end security requires a distributed policy)
- Protects internal users from external users
- Two types: filter-based and proxy-based

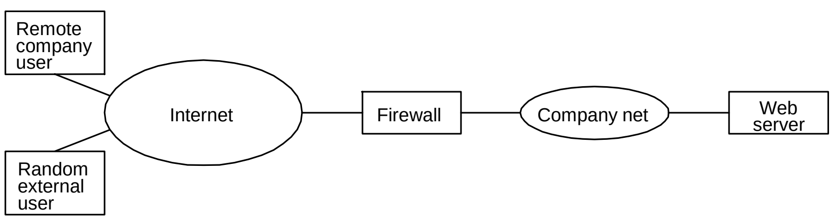


Filter-Based Firewall

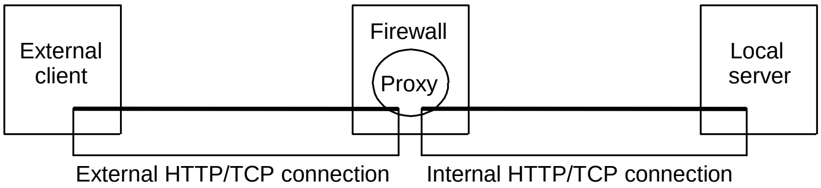
- Simplest and most widely deployed type of firewall
- Configured with a table of addresses that characterize packets that will, or will not, be forwarded
- Each table entry a 4-tuple: IP address and TCP port number for source and destination
  - example
    - (192.12.13.14, 1234, 128.7.6.5, 80)
    - (*,*, 128.7.6.5, 80) wild cards possible
  - sometimes called layer 4 switching (forwarding decision based on IP address and transport layer port number)
- Either forwards everything unless specifically filtered or the opposite (forward by default or drop by default)
- Filter specified when the system is booted or new filters can be inserted into a running system
  - FTP establishes a new TCP connection for each file transfer
  - need for “dynamic port selection” (if using drop by default)

Proxy-Based Firewalls

- Proxy = process that sits between a client process and a server process
  - to the client, proxy appears to be a server
  - to the server, proxy appears to be a client
  - so, proxy has application knowledge build into it
- Example: company web server, some pages accessible to all external users, some pages only for company user (at one or more remote sites)
  - no way to express this as a filter, depends on the URL in the HTTP request



Proxy-Based Firewall (cont.)



- Solution: HTTP proxy
  - remote users establish HTTP/TCP connection to the proxy, which looks at the URL
    - if allowed, proxy establishes a second HTTP/TCP connection to the server and forwards the page request. Then proxy forwards the response in the reverse direction
    - if not allowed, error message to the source
- A proxy
  - has to understand HTTP protocol
  - can be used to balance loads among servers
  - may cache hot Web pages
  - is classified either “transparent” (application does not see proxy) or “classical” (aplication needs to address proxy explicitly)

Security attacks

- Aims
  - fun, getting business knowledge, harming business
- How to achieve goals
  - viruses or trojan horses, breaking into systems, denial-of-service attacks
- How to avoid
  - increase personnel security knowledge, check files, be active in security updating, restrict services per computer
- Firewalls protect insiders from outsides, what if the security threat comes from inside
- Who makes attacks
  - hackers, own employees, business rivals, knowledge sellers, information agencies, terrorists

Denial of service attack

- Security mechanisms prevent any adversary from obtaining unwanted information
  - sometimes an adversary just wants to tease you, to keep you from using your network/computer resources ⇒ denial of service attack
- SYN attack
  - attacker floods the target with SYN packets (TCP connection setup packet), e.g., to port 80 (HTTP port)
  - each SYN requires nontrivial processing, target spends all its time in setting up connections
- IP address attack
  - flood ISP's router with IP packets carrying a serial sequence of IP addresses ⇒ router's first-level route cache blows up, processor spends all its time in building new forwarding tables
- Protection against attacks
  - account for all resources consumed by each user
  - detect when consumption exceeds given policy
  - reclaim the consumed resources using as few additional resources as possible (too massive a reaction ⇒ denial-of-service state)