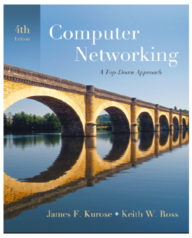
Gestion et sécurité des réseaux informatiques

Guy Leduc

Chapter 3: Securing applications



Computer Networking: A Top Down Approach,
4th edition.
Jim Kurose, Keith Ross
Addison-Wesley, July 2007.
(section 8.5)

Also based on:

Computer Networks, 4th edition
Andrew S. Tanenbaum
Pearson Education, 2003
(sections 8.8 and 8.9.2)

Network Security - PRIVATE Communication in a PUBLIC World
C. Kaufman, R. Pearlman, M. Speciner
Pearson Education, 2002
(chapters 20 and 22)

Chapter 3: Securing applications

Chapter goals:

- security in practice:
  - security in application layer (email)
  - securing DNS

Chapter Roadmap

- security in practice:
  - security in the application layer (email)
    - Mail infrastructure
    - Security services for emails
    - PGP, S/MIME
  - securing DNS

Architecture

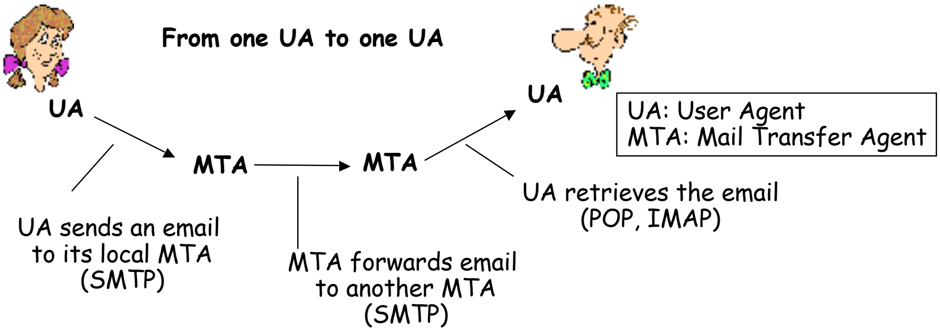
Secure application
TCP/IP

Alternative

Application
Secure TCP/IP

Would it be equivalent?

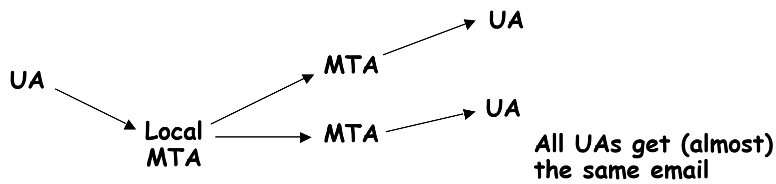
Mail infrastructure (simple case)



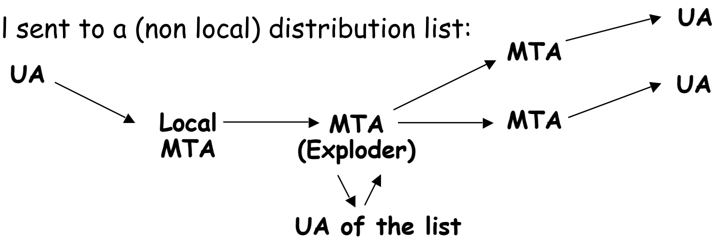
- An MTA is usually configured to only accept emails that are
  - Either coming from a local sender (local UA)
  - Or destined for a local recipient (local UA)
  - The MTA uses the IP address range to check this
- So usually no more than two MTAs on the path

Mail infrastructure (2)

- More general cases:
  - Email sent to multiple recipients (or a list managed locally by source UA):

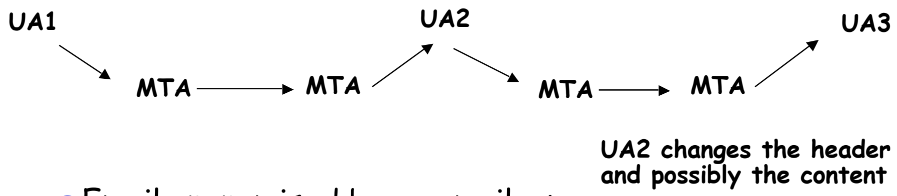


  - Email sent to a (non local) distribution list:

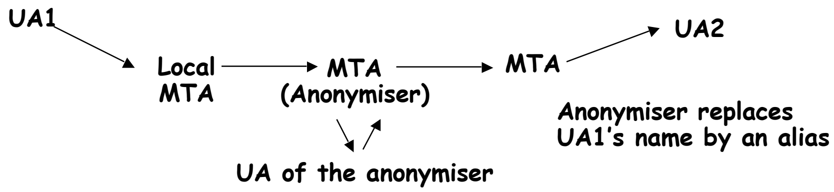


Mail infrastructure (3)

- More general cases (continued):
  - Email forwarded or redirected by recipient:



  - Email anonymised by a remailer:



Chapter Roadmap

- security in practice:
  - security in the application layer (email)
    - Mail infrastructure
    - Security services for emails
    - PGP, S/MIME
  - securing DNS

Security services for emails

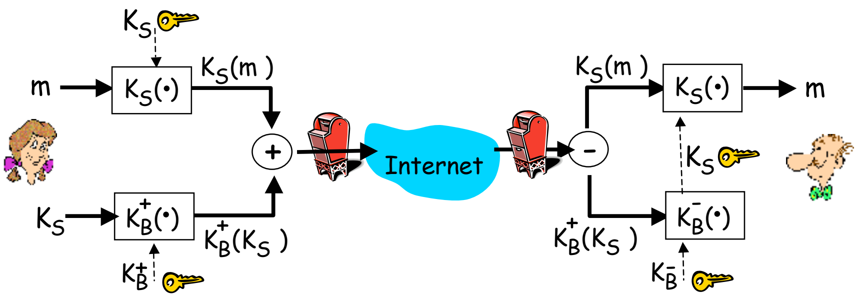
- In those contexts, we will focus on
  - End-to-end privacy
    - The ability to keep anyone but the intended recipient(s) from reading the message
  - Message integrity, including source authentication
    - Reassurance to the recipient(s) that the message (including the identity of the sender) has not been altered since it was transmitted by the sender
  - Non-repudiation
    - The ability of the recipient to prove to a third party that the sender really did send the message. The sender cannot later deny sending the message
    - The opposite of "plausible deniability"
  - Proof of submission
    - Verification given to the sender that the message was handed to the mail delivery system
  - Proof of delivery
    - Verification that the recipient received the message

End-to-end privacy

- Public key encryption versus shared secret key encryption
  - Public key encryption is far less efficient than secret key encryption
    - and emails can be quite long!
  - With public key encryption and multiple recipients the message would have to be encrypted once per recipient!
    - Encryption uses recipient's public key
  - Besides, it is not recommended to use a long-term key more than necessary

End-to-end privacy (one recipient)

- Alice wants to send confidential e-mail, m, to Bob.



Bob:

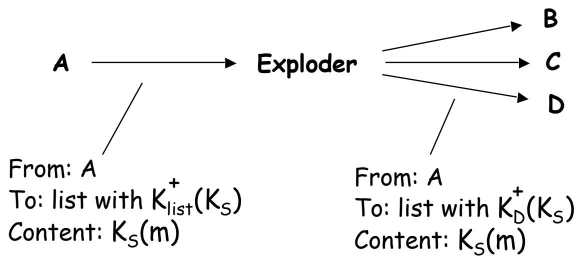
- uses his private key to decrypt and recover K_S
- uses K_S to decrypt $K_S(m)$ to recover m

End-to-end privacy (2 or more recipients)

- A chooses a random secret key K_S
- A encrypts m with K_S
- A encrypts K_S multiple times with public keys of B, C and D, getting $K_B^+(K_S)$, $K_C^+(K_S)$, $K_D^+(K_S)$
- A sends

From: A
To: B with $K_B^+(K_S)$, C with $K_C^+(K_S)$, D with $K_D^+(K_S)$
Content: K_S(m)

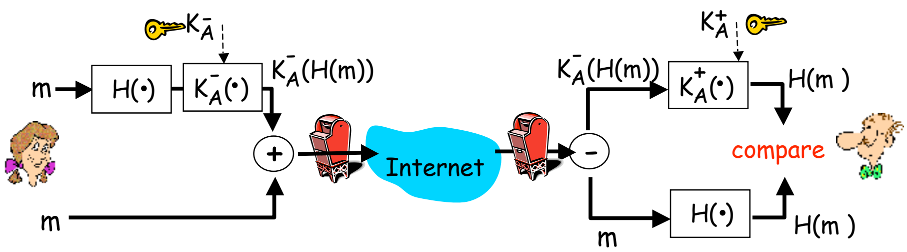
End-to-end privacy (with exploder)



- A encrypts K_S with the public key associated with the list, getting $K^+_{list}(K_S)$
- Exploder decrypts $K^+_{list}(K_S)$
- Exploder encrypts K_S with as many public keys as there are members in the list
- Exploder does NOT have to decrypt the content $K_S(m)$
  - Although it could
- Exploder redirects the many copies to actual recipients

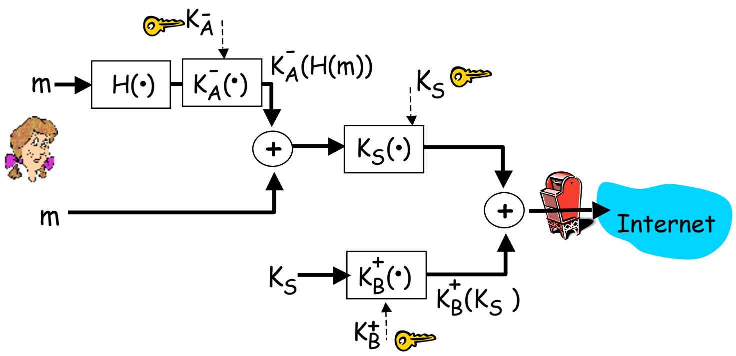
Message integrity and nonrepudiation

- Alice wants to provide sender authentication, message integrity, nonrepudiation



- Alice digitally signs message.
- sends both message (in the clear) and digital signature.

All security services together



Alice uses three keys: her private key, Bob's public key, newly created symmetric key

This works just as fine with an exploder

Plausible deniability

- What if A wants to ensure message integrity (including source authentication) while keeping plausible deniability?
- Solution:
  - A picks a secret key K_S
  - A encrypts K_S with B's public key, getting $K_B^+(K_S)$
  - A signs $K_B^+(K_S)$ with her private key, getting $K_A^-(K_B^+(K_S))$
  - A uses K_S to compute a MAC for m, getting $H(m,K_S)$
  - A sends

    From: A
    To: B
    Content: m, H(m, K_S**),** $K_A^-(K_B^+(K_S))$
- B will know the message came from A, because A signed $K_B^+(K_S)$
- But B can't prove to anyone else that A sent him m
  - B can only prove that at some point A sent some email using key K_S
  - Once B has $K_A^-(K_B^+(K_S))$, he can construct any m together with its integrity code using K_S

Proof of submission

- The mail system can simply compute H(m)
  - Possibly concatenated with any other information that might be useful (e.g. time of submission)
- and then sign H(m) + extra info

Proof of delivery

- Similar to "return receipt requested"
- Two possibilities:
  - 1. The destination signs H(m) + extra info (e.g. time of receipt)
    - Done after the destination UA has received m
    - But the recipient may not send a receipt even if he got the message!
  - 2. The mail system signs H(m) + extra info (e.g. time of receipt)
    - Done after transmitting m to the destination (UA)
    - m is considered transmitted to destination UA when the underlying TCP connection has been closed after the last byte has been acknowledged
    - Note that m may have been received while the last ACK is lost, in which case the message is considered as not received and the mail system does not send a receipt
    - So we get: if a receipt is provided, then the recipient got the message
    - The other direction may not always be true
- In addition, a receipt is itself a message that can be lost
- So, impossible to achieve "the recipient got the message if and only if a receipt is provided"

Annoying text format issues

- Encrypted and/or signed messages are not text files!
- But mailer have been designed with text format in mind
- And some mailers slightly adapt emails en route
  - Add line breaks
  - Convert tabs into spaces
  - Clear the high order bit of every octet (since ASCII characters are 7-bits…)
  - Add escape character '>' before a 'From' appearing at the beginning of a line
  - Consider '.' as a final delimiter of the message when '.' appears at the beginning of a line
  - …
- Even with non secured emails, this may be a problem
- So, for proper transfer, emails should ideally be converted into a canonical format
  - UNIX's uuencode
  - MIME
- We will refer to this function as 'encode / decode'

MIME

- MIME: Multipurpose Internet Mail Extensions
- Sort of presentation sublayer
- Designed to add structure in the message body of emails
  - To support languages with accents (e.g. French, German, Spanish), nonLatin alphabets (e.g. Hebrew, Russian, Greek), languages without alphabets (e.g. Chinese, Japanese), nontextual messages (audio, video)
  - To be encapsulated in emails, data had to be encoded so that the result is an ASCII message
    - Base64 encoding
    - Quoted-printable encoding (more efficient for texts that are almost ASCII)
- Can be used to structure the payload of many protocols (e.g. SMTP, HTTP)

Encoding secured emails

- When a message has to be sent encrypted
  - 1. Encrypt m
  - 2. Encode the result
- When a message is signed
  - 1. Sign H(m)
  - 2. Concatenate m and H(m)
  - 3. Encode the result
- When a message is signed and encrypted
  - 1. Sign H(m)
  - 2. Concatenate m and H(m)
  - 3. Encrypt the result of 2
  - 4. Encode the result of 3
  - This layering of encryption over signature allows to decrypt and encrypt again with another key (if need be) without invalidating the initial signature

From Network Security, by Kaufman et al. © Pearsons

Chapter Roadmap

- security in practice:
  - security in the application layer (email)
    - Mail infrastructure
    - Security services for emails
    - PGP, S/MIME
  - securing DNS

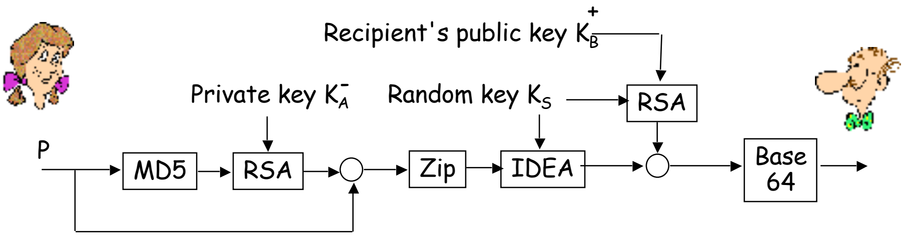
Pretty good privacy (PGP)

- Internet e-mail encryption scheme, de-facto standard
- uses symmetric key cryptography, public key cryptography, hash function, and digital signature as described
- provides secrecy, sender authentication, integrity
  - + data compression, key management
- PGP intentionally uses existing cryptographic algorithms (RSA, IDEA, MD5) rather than inventing new ones
- inventor, Phil Zimmerman, was target of 3-year federal investigation

A PGP signed message:

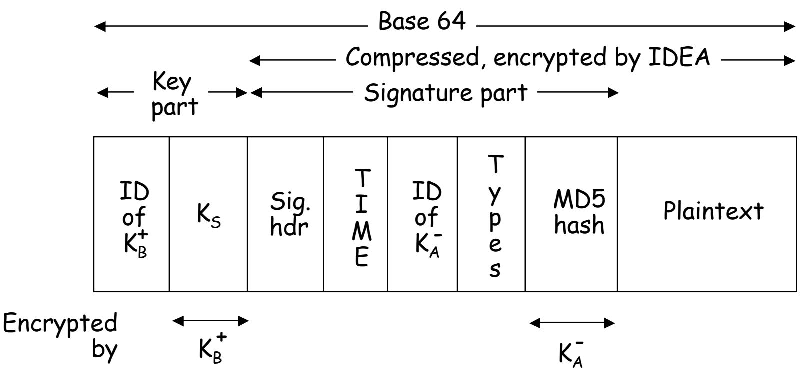
```
---BEGIN PGP SIGNED MESSAGE---
Hash: SHA1

Bob:My husband is out of town
   tonight.Passionately yours, Alice

---BEGIN PGP SIGNATURE---
Version: PGP 5.0
Charset: noconv
yhHJRHhGJGhgg/12EpJ+lo8gE4vB3mqJhFEvZ
   P9t6n7G6m5Gw2
---END PGP SIGNATURE---
```


PGP principle



- PGP hashes (MD5) the plaintext P and then signs the resulting hash (128 bits) using RSA
- The signature is concatenated to P, and the result is compressed
- A 128-bit key is generated and used to encrypt the compressed message with IDEA
- The random key is encrypted with RSA and appended to the encrypted message

PGP format



- IDs are used to indicate which key was used to encrypt K_S and which key should be used to verify the signature on the hash (notion of key rings)
- Types are used to identify the algorithms (RSA, MD5)

Key management in PGP

- Four RSA key lengths
  - Casual: 384 bits, can be broken easily
  - Commercial: 512 bits, breakable by NSA, etc
  - Military: 1024 bits, not breakable on earth
  - Alien: 2048 bits, not breakable elsewhere either
- No reason for not using Alien strength key
  - Only two encryptions of 128 bits
- Key rings
  - Allows to change the private/public key pairs regularly, without invalidating recent messages

PGP certificates - Trust

- Examples of PGP certificates:
  - {A's public key is K_A^+ } signed by K_B
  - {B's public key is K_B^+} signed by K_C
  - {A's public key is K_A^+ } signed by K_D
- Several issues:
  - How to find a chain leading from a known key to A's key?
  - There might be multiple chains, leading to different keys for A. So what?
  - How can I trust a chain if I find one?
    - Trust is not really transitive
- Each public key is associated with a trust level
  - Taken from a web page?
  - Given to me on a business card?
  - Communicated over the phone?
  - Handed to me on a disk?
- PGP public keys can also be certified (X.509)

Secure MIME - IETF S/MIME

- The approach is similar to PGP
- Based on PEM (Privacy Enhanced Mail) from IETF which nobody ever used
- With PGP a message in signed and encrypted, and then MIME encoded
- S/MIME provides the same functionality, but with standardized cryptographic message formats (different from PGP)
  - PKCS (Public Key Cryptography Standards)
  - For example, PKCS#7 defines the format and its encoding using the ASN.1 Basic Encoding Rules (BER)
- MIME is extended with some keywords to identify the encrypted and/or signed parts in the message

Variants of application security architectures

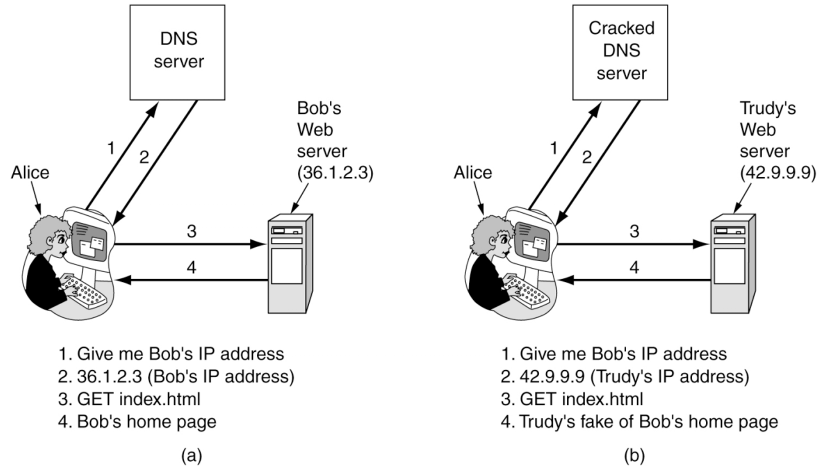
user process
PGP / MIME
SMTP/HTTP
TCP/IP

user process
S/MIME
SMTP/HTTP
TCP/IP

Chapter Roadmap

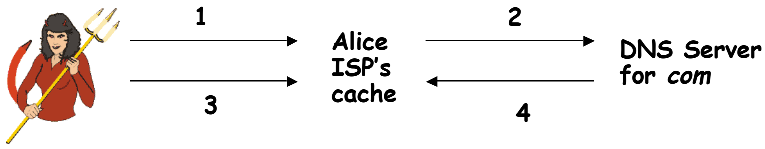
- security in practice:
  - security in the application layer (email)
  - securing DNS

Attack based on breaking DNS



(a) Normal situation
(b) An attack based on breaking into DNS and modifying Bob's record.

DNS Spoofing (simplified)

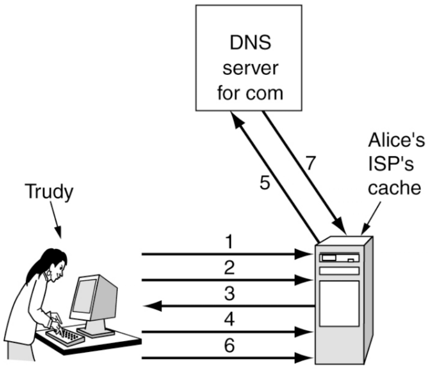


1: Lookup bob.com
2: Query for bob.com
3: Trudy's forged answer: "Bob is 42.9.9.9" (poisoned cache)
4: Real answer (rejected, too late)

- In message 3: IP spoofing is used (source address = DNS server for com)
- However, DNS requests carry a sequence number…
  - So,message 2 has a seq. nr. that message 3 has to carry!
  - How to guess it?
  - See next slide

DNS spoofing (real attack)

To learn the seq. nr., Trudy registers a domain herself
e.g., trudy-the-intruder.com
And Trudy runs a DNS server for it (on her PC)
e.g., dns.trudy-the-intruder.com



Then it goes like this:

1. Look up foobar.trudy-the-intruder.com
   (to force it into the ISP's cache)
2. Look up www.trudy-the-intruder.com
   (to get the ISP's next sequence number)
3. Request for www.trudy-the-intruder.com
   (Carrying the ISP's next sequence number, n)
4. Quick like a bunny, look up bob.com
   (to force the ISP to query the com server in step 5)
5. Legitimate query for bob.com with seq = n+1
6. Trudy's forged answer: Bob is 42.9.9.9, seq = n+1
7. Real answer (rejected, too late)

6'. Actually Trudy sends several 6's
with successive numbers: n+2, n+3,…

Solution to DNS spoofing

- Attack is also called DNS cache poisoning
- Solution: use random numbers to identify DNS requests, instead of sequential numbers
- Still:
  - Request id is only 16 bit long (65536 values)
  - If the attacker has time to bombard the DNS server with 100 answers before the real one comes back: 1 chance to succeed out of 655!

Bailiwick check

- The DNS allows the DNS answer to include additional info.
- Example:
  - user queries BadGuysAreUs.com
  - user gets IP address of BadGuysAreUs.com
  - user may also get additional pairs piggybacked in the answer, such as: (www.paypal.com, fake IP)
    - Would poison the cache!
- Bailiwick check:
  - extra info is ignored if it pertains to a domain that is different from the one that was asked about in the first place

Bailiwick check is not enough

- Attack:
  - Attacker asks aaa.paypal.com, then sends 100 answers with random ids: success probability = 1/655
  - Attacker asks aab.paypal.com, then sends 100 answers with random ids: success probability = 1/655
  - ...
  - until success, e.g. on apq.paypal.com
- Nothing to worry about, it seems, but answers could also contain piggybacked data for www.paypal.com, together with a fake IP for it!
  - Bailiwick check will allow it: same domain!
- Patch: randomize also the port numbers used for DNS requests: 100 answers are unlikely to succeed.

Secure DNS: DNSSEC

- DNSSEC goes beyond this
- It is based on public-key cryptography:
  - Every DNS zone has a public/private key pair
  - All info sent by a DNS server is signed with the originating zone's private key, for authenticity
- So, DNS clients need to know the zone's public keys to check the signatures
  - Clients may be preconfigured with the public keys of all the top-level domains
- In 2008: used in Sweden (.se domain) but not largely deployed (yet?)

Resource Record Sets (RRSets)

The unit of transmission sent to the client is the RRSet

An example RRSet for *bob.com*

Domain name	Time to live	Class	Type	Value
bob.com.	86400	IN	A	36.1.2.3
bob.com.	86400	IN	KEY	3682793A7B73F731029CE2737D...
bob.com.	86400	IN	SIG	86947503A8B848F5272E53930C...

The ***KEY*** record is Bob's (uncertified) public key

The ***SIG*** record is the top-level *com* server's signed hash of the *A* and *KEY* records to verify their authenticity.

Other security issues

- « Phishing 2.0 »
- Many web pages contain exploit code with malicious attachments that exploit bugs in the computer's software:
  - It changes one file (e.g. in the Windows registry settings) telling the PC to go to the criminal's DNS server instead of the ISP or enterprise DNS server
- Note:
  - End of 2007: several thousands such web pages
  - In 2008: 0.4% of DNS servers (i.e. ± 70000) are behaving badly
- Solution: antivirus software!

Security in the application layer (summary)

- Securing emails
  - Architecture takes intermediate systems into account
    - But ensures end-to-end security
  - Security services: Confidentiality, sender authentication, message integrity, non repudiation
    - Uses secret-key and public-key cryptography
  - Example: PGP, S/MIME
- securing DNS
  - DNS spoofing/cache poisoning
    - Shows that a mapping function (here names to addresses) may be the Achilles' heel
    - Random ids are weak
    - Bailiwick check is not enough
  - Use of public-key cryptography: DNSSEC
    - But has not caught on yet