Network Security: Email security, PKI

Tuomas Aura, Microsoft Research, UK

Outline

1. First security protocols: email security
2. Certificates
3. PGP web of trust
4. X.509 public-key infrastructure
5. Key-oriented PKIs

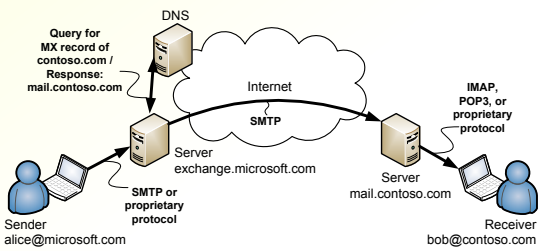
Email security

First question

- What kind of security is needed for email?
  - Confidentiality?
  - Authentication?
  - Non-repudiation?

  (PGP, S/MIME)
  - Mandatory access control / DRM?
  - Spam control?
  - Phishing prevention?
- We use email security as the first example because it is a fairly straightforward application of crypto and allows us to introduce many basic concepts
  - Crypto does not solve all email security problems

Internet email architecture



- Alice sends mail to bob@contoso.com

Observations about email

- Email is sent between human users (e.g. Alice and Bob)
- Users send and receive email using a software agent
- Sender agent connects to an SMTP server (TCP port 25), often without much security
- Email delivered over the Internet with SMTP without any security
- Receiver connects to an IMAP or POP3 server, usually with reasonable user authentication
- Local email within one domain may be secure, global email is not

➔ Best to secure email end-to-end rather than try to secure each step

➔ Need user authentication and user-to-user encryption

Email security

- Application-level network security protocols for encrypting and signing email
- Email software uses public-key encryption and/or signatures to protect email
- One must have the other end's authentic public key for sending encrypted mail or verifying signed mail
- Key distribution is a an issue, but let's first look at encryption and integrity

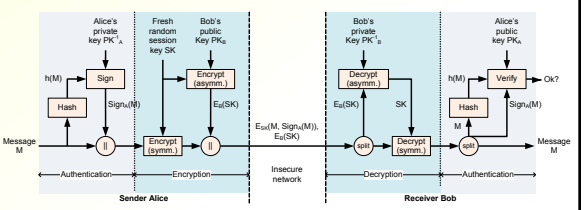
7

Order of signing, compression and encryption

- Opinions?
- Observations:
  - Signing without seeing content is dangerous → sign the plaintext
  - Attacker could change the signature on signed messages → sign the plaintext
  - Encryption only protects secrecy; thus, ciphertext might decrypt to multiple different plaintexts → sign the plaintext
  - Signature or MAC might reveal something about message contents → encrypt also the signature or MAC
  - Ciphertext does not compress → compress before encryption
  - Decompression might not guarantee unambiguous output in the presence of a malicious influence → sign the uncompressed plaintext
  - Receiver might decompress or recompress the signed data for storage; authentication of compressed messages prevents that → compress email after authentication
- Typical order: sign, compress, encrypt
- Exceptions common but need a good justification

8

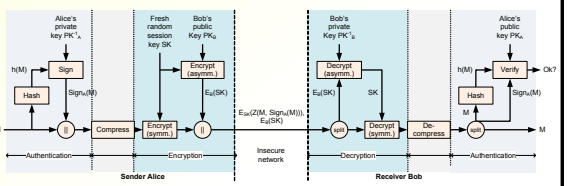
Sign, encrypt



- $E_{SK}(M, Sign_A(M)),\ E_B(SK)$

9

Sign, compress, encrypt



- Sender and receiver need to know each other's public keys
- Options to encrypt only or to sign only:
  - Possible to sign without knowing receiver's public key, or when sending to a mailing list
  - Possible to encrypt without identifying sender

10

Email integrity problem

- Email servers modify messages:
  - Each server adds headers
  - Old email systems were not 8-bit safe
  - Servers perform character-set conversions
  - Firewalls remove or replace suspicious attachments
  - Proxies compress text and images for mobile clients

  → Bits change, authentication fails
- Solution: encode the signed part of the message in "safe" characters that are not modified in transit
  - Around 64 safe ASCII characters give 6 bits per character
  - Base64, Radix64 etc.
- Remaining problems:
  - Signed message not human-readable text
  - 33% expansion in message size

11

Pretty Good Privacy (PGP)

- Zimmermann 1991–
- The sign-compress-encrypt process shown earlier, instantiated with the best available algorithms of the time:
  - IDEA (128-bit keys) in CBC mode (later 3DES, AES in CFB)
  - SHA-1 hash function
  - RSA public-key signatures
  - RSA and ElGamal Public-key encryption
  - Radix-64 conversion
  - Timestamp
- The first strong encryption product available to the public
- OpenPGP [RFC 4880]

12

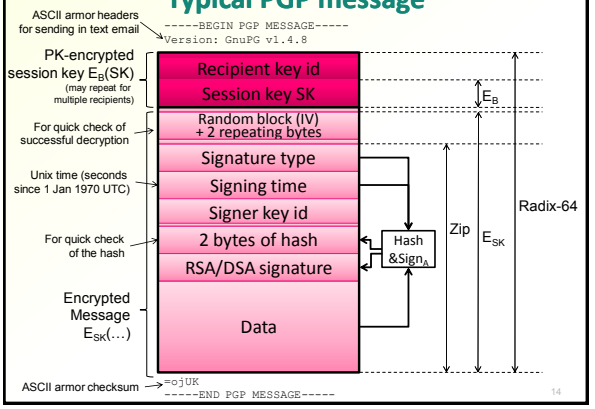
Example: PGP-encrypted message

```
-----BEGIN PGP MESSAGE-----
Version: GnuPG v1.4.8
Comment: Encrypted secret message.

hQEOA1e+1x6YuUMCEAQAoST11/obnXOB6fhIhmLnGVLhuxmsksKD+Efyk7ja9gOx
U5X98/25ZVDQz0EiOkRjW2LChuZt9Kesh1DSIRwB/llXCm3pbNX/V+ajkL4Fzxlw
jWCCedv527SUNTUP701hLbh4O2kHHxMdEn41zVo9TPUgtQlBIo32k/xP2RYtPCEE
AJDhcyp+COLaI4idibfSrDDtYcT+hVVFVveIteTIcznoUoS1yVyipE4mBwa380c6
TiwImq63hOhs62c9BOQv7G9cnaqEZNg0nLiVZD+K/JeN00zILm+TzdWZxrW0l9nA
+tsMwznUZ2V/kQZjS9xkPWjn7ZzPTyW6gLhjWQNlr93S0lcBT0CJy285ixFz9UrJ
qjK2azsBdXRcVuXFdh84LWlE/8/8DwdLgSK9X/jPNv3/WGLA4Ez2xTFIUorVi5Xe
M9dpriEQ0Jg2msnz2bjqRGZliXXo6m8ye/A=
=YWDi
-----END PGP MESSAGE-----
```

- "Meet me in the park at 6 PM."

Typical PGP message



Public-key distribution

- PGP public keys are usually distributed manually
  - Download from a web page or take from a received email
    → key distribution often insecure
- Users can endorse keys of others by signing them
  - Sign: key, name, level of trust, signing, expiry time
  - Mark friends and well-known people as trusted, derive trust to others from endorsements

  → PGP web of trust

Radix-64 encoding

- Use safe ASCII characters to represent values 0..63:

  `ABCDEFGHIJKLMNOPQRSTUVWXYZabcdefghijklmnopqrstuvxyz0123456789+/`
- Encode each 3 bytes as 4 characters:

```
+--first octet--+-second octet--+--third octet--+
|7 6 5 4 3 2 1 0|7 6 5 4 3 2 1 0|7 6 5 4 3 2 1 0|
+-----------+---+-------+-------+---+-----------+
|5 4 3 2 1 0|5 4 3 2 1 0|5 4 3 2 1 0|5 4 3 2 1 0|
+--1.char---+--2.char---+--3.cahr---+--4.char---+
```

- If the data length is not divisible by 3, pad with one or two = characters to indicate actual length

S/MIME

- PGP is mainly used by private persons and academia
- S/MIME is a similar standard used primarily by enterprises
- Message structure based on the MIME standards
  → Envelopes and signatures are new MIME types
  → Base64 encoding

Non-repudiation

- Proof of authenticity to third parties
  - Email sender should not be able to later deny sending it (i.e. repudiate the message)
  - Third party, such as a judge, needed to make the decision
  - The public key must be somehow registered to bind it to the person signing
- Uses:
  - Accountability for sent emails
  - Contract signing
- Questions:
  - Does the sender of normal emails want to go to extra lengths to be accountable for the emails you sent?
    → Incentives poorly aligned
  - Are business contracts signed using secure email?

How good is email security?

- Is it secure?
  - PGP public keys are rarely distributed through secure channels. Certificates don't necessarily mean that much
  - Absolute security not necessary for privacy or to prevent large-scale monitoring by governments
- Is it useful?
  - Many people sign email when sending. Few verify the signatures or take action if a signature is invalid
  - Email users rarely want non-repudiation
  - For most users, email security is more trouble than benefit
  - Spam filtering may require email authentication

19

Crypto Wars – Some History (1)

- Military origins:
  - Until '70s, encryption was military technology. In '70s and '80s, there was limited commercial use
  - American export restrictions and active discouragement prevented wide commercial and private use
- Arguments against strong encryption:
  - Intelligence agencies (NSA) cannot spy on encrypted international communications
  - Criminals, terrorists and immoral people use encryption
- US policies delayed availability of strong crypto (both encryption and authentication) for private and commercial use by up to 20 years

20

Crypto Wars – Some History (2)

- In the '90s, demand and availability exploded:
  - Encryption was needed to secure Internet commerce
  - Activist advocated encryption for privacy and security
- Anyone could download strong encryption products like PGP, SSH and SSL from the Internet
  - PGP source code printed as a book, taken abroad, scanned in and distributed freely outside the USA
  - SSH distributed from Finland 2005–
  - SSL on Netscape web browser 2005–
- Around 2000, most US restrictions were lifted
  - Strong encryption is now included in off-the-shelf products, such as web browsers and operating systems

21

Crypto Wars – Some History (3)

- Did encryption cause or solve problems?
  - Systems that use strong encryption have other security flaws
  - Serious fraud is committed by the end nodes, not by sniffers
  - Police and intelligence agencies found other ways, e.g. rubber-hose cryptanalysis

→ Encryption and authentication is just one building block in trustworthy systems, not the complete solution

22

Exercises

23

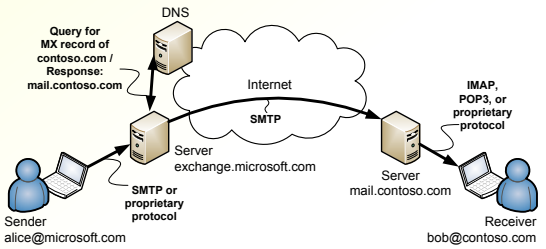
First question

- What kind of security is needed for email?
  - Confidentiality?
  - Authentication?
  - Non-repudiation?

  } PGP, S/MIME
  - Mandatory access control / DRM?
  - Spam control?
  - Phishing prevention?
- We use email security as the first example because it is a fairly straightforward application of crypto and allows us to introduce many basic concepts
  - Crypto does not solve all email security problems

25

Internet email architecture



- Alice sends mail to bob@contoso.com

26

Observations about email

- Email is sent between human users (e.g. Alice and Bob)
- Users send and receive email using a software agent
- Sender agent connects to an SMTP server (TCP port 25), often without much security
- Email delivered over the Internet with SMTP without any security
- Receiver connects to an IMAP or POP3 server, usually with reasonable user authentication
- Local email within one domain may be secure, global email is not

➔ Best to secure email end-to-end rather than try to secure each step

➔ Need user authentication and user-to-user encryption

27

Email security

- Application-level network security protocols for encrypting and signing email
- Email software uses public-key encryption and/or signatures to protect email
- One must have the other end's authentic public key for sending encrypted mail or verifying signed mail
- Key distribution is a an issue, but let's first look at encryption and integrity

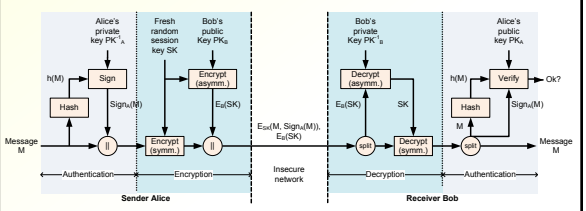
28

Order of signing, compression and encryption

- Opinions?
- Observations:
  - Signing without seeing content is dangerous → sign the plaintext
  - Attacker could change the signature on signed messages → sign the plaintext
  - Encryption only protects secrecy; thus, ciphertext might decrypt to multiple different plaintexts → sign the plaintext
  - Signature or MAC might reveal something about message contents → encrypt also the signature or MAC
  - Ciphertext does not compress → compress before encryption
  - Decompression might not guarantee unambiguous output in the presence of a malicious influence → sign the uncompressed plaintext
  - Receiver might decompress or recompress the signed data for storage; authentication of compressed messages prevents that → compress email after authentication
- Typical order: sign, compress, encrypt
- Exceptions common but need a good justification

29

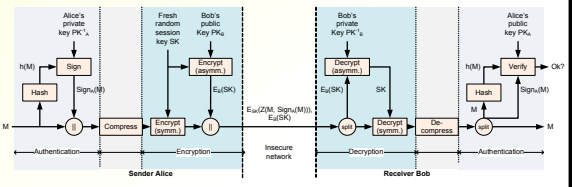
Sign, encrypt



- $E_{SK}(M, Sign_A(M)),\ E_B(SK)$

30

Sign, compress, encrypt



- Sender and receiver need to know each other's public keys
- Options to encrypt only or to sign only:
  - Possible to sign without knowing receiver's public key, or when sending to a mailing list
  - Possible to encrypt without identifying sender

31

Email integrity problem

- Email servers modify messages:
  - Each server adds headers
  - Old email systems were not 8-bit safe
  - Servers perform character-set conversions
  - Firewalls remove or replace suspicious attachments
  - Proxies compress text and images for mobile clients

  → Bits change, authentication fails
- Solution: encode the signed part of the message in "safe" characters that are not modified in transit
  - Around 64 safe ASCII characters give 6 bits per character
  - Base64, Radix64 etc.
- Remaining problems:
  - Signed message not human-readable text
  - 33% expansion in message size

32

Pretty Good Privacy (PGP)

33

Pretty Good Privacy (PGP)

- Zimmermann 1991–
- The sign-compress-encrypt process shown earlier, instantiated with the best available algorithms of the time:
  - IDEA (128-bit keys) in CBC mode (later 3DES, AES in CFB)
  - SHA-1 hash function
  - RSA public-key signatures
  - RSA and ElGamal Public-key encryption
  - Radix-64 conversion
  - Timestamp
- The first strong encryption product available to the public
- OpenPGP [RFC 4880]

34

Example: PGP-encrypted message

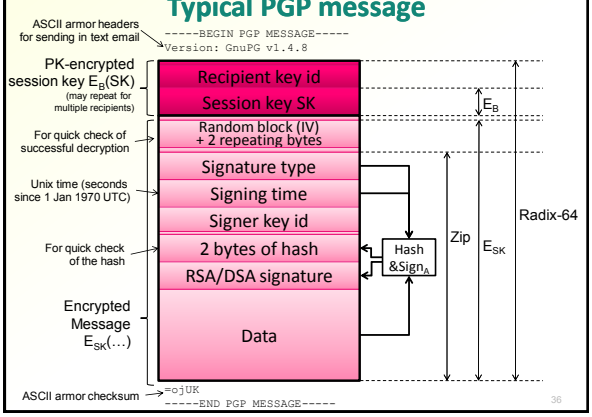
```
-----BEGIN PGP MESSAGE-----
Version: GnuPG v1.4.8
Comment: Encrypted secret message.

hQEOA1e+1x6YuUMCEAQAoST11/obnXOB6fhIhmLnGVLhuxmsksKD+Efyk7ja9gOx
U5X98/25ZVDQz0EiOkRjW2LChuZt9Keshl DSIRwB/llXCm3pbNX/V+ajkL4Fzxlw
jWCCedv527SUNTUP70lhLbh4O2kHHxMdEn41zVo9TPUgtQlBIo32k/xP2RYtPCEE
AJDhcyp+COLaI4idibfSrDDtYcT+hVVFVveIteTIcznoUoS1yVyipE4mBwa380c6
TiwImq63hOhs62c9BOQv7G9cnaqEZNg0nLiVZD+K/JeN00zILm+TzdWZxrW019nA
+tsMwznUZ2V/kQZjS9xkPWjn7ZzPTyW6gLhjWQNlr93SOlcBT0CJy285ixFz9UrJ
qjK2azsBdXRcVuXFdh84LW1E/8/8DwdLgSK9X/jPNv3/WGLA4Ez2xTFIUorVi5Xe
M9dpriEQ0Jg2msnz2bjqRGZliXXo6m8ye/A=
=YWDi
-----END PGP MESSAGE-----
```

- "Meet me in the park at 6 PM."

35

Typical PGP message



36

Public-key distribution

- PGP public keys are usually distributed manually
  - Download from a web page or take from a received email
    → key distribution often insecure
- Users can endorse keys of others by signing them
  - Sign: key, name, level of trust, signing, expiry time
  - Mark friends and well-known people as trusted, derive trust to others from endorsements

  → PGP web of trust

37

Radix-64 encoding

- Use safe ASCII characters to represent values 0..63:

```
ABCDEFGHIJKLMNOPQRSTUVWXYZabcdefghijklmnopqrstuv
xyz0123456789+/
```

- Encode each 3 bytes as 4 characters:

```
+--first octet--+-second octet--+--third octet--+
|7 6 5 4 3 2 1 0|7 6 5 4 3 2 1 0|7 6 5 4 3 2 1 0|
+----------+---+-------+-------+---+----------+
|5 4 3 2 1 0|5 4 3 2 1 0|5 4 3 2 1 0|5 4 3 2 1 0|
+--1.char---+--2.char---+--3.cahr---+--4.char---+
```

- If the data length is not divisible by 3, pad with one or two = characters to indicate actual length

38

S/MIME

- PGP is mainly used by private persons and academia
- S/MIME is a similar standard used primarily by enterprises
- Message structure based on the MIME standards
  → Envelopes and signatures are new MIME types
  → Base64 encoding

39

Non-repudiation

- Proof of authenticity to third parties
  - Email sender should not be able to later deny sending it (i.e. repudiate the message)
  - Third party, such as a judge, needed to make the decision
  - The public key must be somehow registered to bind it to the person signing
- Uses:
  - Accountability for sent emails
  - Contract signing
- Questions:
  - Does the sender of normal emails want to go to extra lengths to be accountable for the emails you sent?
    → Incentives poorly aligned
  - Are business contracts signed using secure email?

40

How good is email security?

- Is it secure?
  - PGP public keys are rarely distributed through secure channels. Certificates don't necessarily mean that much
  - Absolute security not necessary for privacy or to prevent large-scale monitoring by governments
- Is it useful?
  - Many people sign email when sending. Few verify the signatures or take action if a signature is invalid
  - Email users rarely want non-repudiation
  - For most users, email security is more trouble than benefit
  - Spam filtering may require email authentication

41

Crypto Wars – Some History (1)

- Military origins:
  - Until '70s, encryption was military technology. In '70s and '80s, there was limited commercial use
  - American export restrictions and active discouragement prevented wide commercial and private use
- Arguments against strong encryption:
  - Intelligence agencies (NSA) cannot spy on encrypted international communications
  - Criminals, terrorists and immoral people use encryption
- US policies delayed availability of strong crypto (both encryption and authentication) for private and commercial use by up to 20 years

42

Crypto Wars – Some History (2)

- In the '90s, demand and availability exploded:
  - Encryption was needed to secure Internet commerce
  - Activist advocated encryption for privacy and security
- Anyone could download strong encryption products like PGP, SSH and SSL from the Internet
  - PGP source code printed as a book, taken abroad, scanned in and distributed freely outside the USA
  - SSH distributed from Finland 2005–
  - SSL on Netscape web browser 2005–
- Around 2000, most US restrictions were lifted
  - Strong encryption is now included in off-the-shelf products, such as web browsers and operating systems

43

Crypto Wars – Some History (3)

- Did encryption cause or solve problems?
  - Systems that use strong encryption have other security flaws
  - Serious fraud is committed by the end nodes, not by sniffers
  - Police and intelligence agencies found other ways, e.g. rubber-hose cryptanalysis

→ Encryption and authentication is just one building block in trustworthy systems, not the complete solution

44

Certificates

Public-key certificates

- Signature verifier must know the signer's public key. For example, how to verify T_A, M, $S_A(T_A, M)$?
- Identity certificate is a signed message issued by a trusted entity, which binds a name and a public key to each other:

  $Cert_A = A, PK_A, T_{expiry}, S_{issuer}(A, PK_A, T_{expiry})$
- To verify message T_A, M, $S_A(T_A, M)$, $Cert_A$:
  - Verify signature S_A with PK_A
  - Check freshness of the timestamp T_A
  - Verify the certificate, and check $T_A < T_{expiry}$
- Security protocols often assume that the protocol participants have certificates, but who issues them?

46

PGP web of trust

About trust

- The word trust has many definitions:
  - Belief that an entity follows certain rules, does not behave maliciously, and is reasonably competent (belief that someone is honest, not an attacker)
  - Out-of-band information, which is taken as a fact and cannot be verified by other means (you must trust something that you cannot verify)
  - Infrastructure used to bootstrap authentication or authorization (shared key, PKI, trusted online server)
  - Fuzzy value that arises from human social interaction, or its imitation by machines
- Either define what you mean by "trust" in each context, or avoid using the word altogether

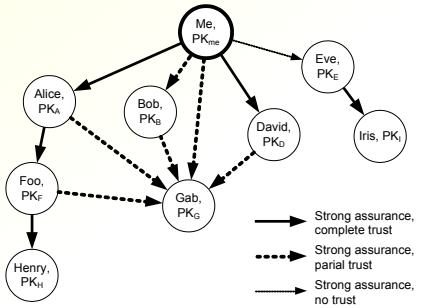
48

PGP key distribution

- PGP users need to know each other's public keys. But how to verify they are authentic?
  - Need to verify only the key fingerprint (hash value)
  - Personal verification: ask the person, print on business cards, etc.
  - PGP key ring signed by someone you trust
- PGP key ring contains public key, trust level, user id or name and one or more signatures. Each signature includes assurance level
  - Meaning: signers say that the public key belongs to the user
- Trust levels: none, partial trust, complete trust
  - Meaning: level of belief that entity tells the truth when it signs key rings
  - Signer can recommend the key owner as a person of high integrity to sign keys for others
- Assurance levels: unspecified, no, casual, heavy-duty
  - Meaning: how carefully did signer check that the key belongs to the user
- Assurance of key-person binding and trusting that person to tell the truth (sign keys of others) are separate issues

49

PGP web of trust



- Which keys should I use for sending confidential mail, for authenticating received mail, for contract signing?

50

Subkeys

- User's top-level PGP key is a signature key, which can sign key rings
- If RSA, the same key can be used for both signature and encryption
- Otherwise, the signature key can sign a separate subkey (in a key ring) for encryption

51

Revocation

- Certificate revocation:
  - Anyone who signed a certificate can revoke it
  - Similar to a certificate but assurance level "revocation"
- Key revocation:
  - Key can revoke itself (private key needed for this)
  - Used when private key compromised
  - Recommendation: sign a key revocation for your key and store it in a safe place just in case
- PGP key servers are email and ftp-based repositories for key rings, including revocations
- Certificates may have a validity period, after which revocation certificates no longer need to be revoked
  - Unfortunately, infinite validity is common practice → need to store revocations forever
  - Common to revoke keys when they are replaced with a new ones → many unnecessary revocations

52

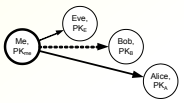
Issues with the web of trust

- Names can be arbitrary strings → how to tell apart two John Smiths?
  - Same issue in Facebook
- Two dimensions (trust and assurance) create complexity that is hard to understand
- Rules for evaluating evidence is are not fully defined → human judgment required at every step
- Even if the rules were fully defined, would they be an accurate model of human behavior?
- Design suggests transitive trust: "I trust Alice, Alice trusts Foo, Foo trust Henry. Thus, I also trust Foo and Henry."
  - In general, trust and assurance are not transitive
- Many pieces of weak evidence may accumulate to strong assurance → potential for misuse
- Goal a completely distributed system, but key servers maintain a global revocation list

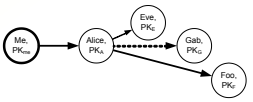
53

PGP web of trust in practice

- Use keys of which you have direct strong assurance



- Accept certificates signed those you trust completely



- Certificates give assurance for key-id binding, but no recursive trust

54

X.509 public-key infrastructure (PKI)

X.500 names

- ISO X.500 standard defines hierarchical directory
  - More advanced than DNS but not widely used
  - Hierarchical names used in X.509 certificates
- X.500 names:
  - C = country, S = state, L = locality, O = organization, OU = organization unit, CN = common name
- Names used in practice:
  - `CN = Tuomas Aura, O = Microsoft Corporation, L = Redmond, S = Washington, C = US`
  - `CN = Tuomas Aura, OU = UserAccounts, DC = europe, DC = microsoft, DC = com`
  - `CN = www.bankofamerica.com, OU = DMZUNIXAPPS, O = Bank of America Corporation, L = Charlotte, S = North Carolina, C = US`
- Hierarchical naming should ensure a 1-to-1 mapping between names and principals (unlike in PGP web of trust). Such names are called distinguished names

56

ASN.1, OID

- ASN.1 standard for defining protocol messages
  - Abstract notation for data structures
  - BER/DER encoding rules → standardized binary encoding with recursive <type tag, length, value> structure
  - Unambiguous parsing of binary messages
  - ASN.1 specification of protocol messages is directly compiled into C-code for encoding and decoding them
  - Encoded data unreadable to humans
  - Most Internet standards defined in RFCs use more light-weight bit-field or text-based syntax and manually encoded parsers
- X.509 certificates are encoded in ANS.1 DER
- One ASN.1 type is object identifier (OID)
  - Globally unique identifiers (similar to const or enum)
  - Variable length, each organization can get its own prefix

57

ASN.1 example

- ASN.1 (from RFC 3280)

```
PersonalName ::= SET {
   surname     [0] IMPLICIT PrintableString
                   (SIZE (1..ub-surname-length)),
   given-name  [1] IMPLICIT PrintableString
                   (SIZE (1..ub-given-name-length)) OPTIONAL,
   initials    [2] IMPLICIT PrintableString
                   (SIZE (1..ub-initials-length)) OPTIONAL,
   generation-qualifier [3] IMPLICIT PrintableString
                   (SIZE (1..ub-generation-qualifier-length))
                   OPTIONAL }
```

- Compare with RFC-style packet diagrams:

```
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|ST | 0 |  TYPE |    Reserved   |            n                  |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                        Router ID                              |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
| PrefixLength  | Prefix byte 1 | Prefix byte 2 |     ...       |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|      ...      | PrefixLength  | Prefix byte 1 | Prefix byte 2 |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|     ...                                                       |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

58

X.509 certificate fields (1)

Mandatory fields:

- Version
- Serial number — together with Issuer, uniquely identifiers the certificate
- Signature algorithm — for the signature on this certificate; usually *sha1RSA*; includes any parameters
- Issuer — name (e.g. CN = Microsoft Corp Enterprise CA 2)
- Valid from — usually the time when issued
- Valid to — expiry time
- Subject — distinguished name of the subject
- Public key — public key of the subject

- Standard notation for a certificate: CA<<Alice>>

59

X.509 certificate fields (2)

Common extension fields:

- Key usage — bit field indicating usages for the subject key (*digitalSignature, nonRepudiation, keyEncipherment, dataEncipherment, keyAgreement, keyCertSign, cRLSign, encipherOnly, decipherOnly*)
- Subject alternative name — email address, DNS name, IP address, etc.
- Issuer alternative name
- Basic constraints — (1) is the subject a CA or an end entity, (2) maximum length of delegation to sub-CAs after the subject
- Name constraints — limit the authority of the CA
- Certificate policies — list of OIDs to indicate policies for the certificate
- Policy constraints — certificate policies
- Extended key usage — list of OIDs for new usages, e.g. server authentiction, client authentication, code signing, email protection, EFS key, etc.
- CRL distribution point — where to get the CRL for this certificate, and who issues CRLs
- Authority info access — where to find information about the CA and its policies

60

Certificate example

[Wikipedia]

```
Certificate:
    Data:
        Version: 1 (0x0)
        Serial Number: 7829 (0x1e95)
        Signature Algorithm: md5WithRSAEncryption
        Issuer: C=ZA, ST=Western Cape, L=Cape Town, O=Thawte Consulting cc,
                OU=Certification Services Division,
                CN=Thawte Server CA/emailAddress=server-certs@thawte.com
        Validity
            Not Before: Jul  9 16:04:02 1998 GMT
            Not After : Jul  9 16:04:02 1999 GMT
        Subject: C=US, ST=Maryland, L=Pasadena, O=Brent Baccala,
                 OU=FreeSoft, CN=www.freesoft.org/emailAddress=baccala@freesoft.org
        Subject Public Key Info:
            Public Key Algorithm: rsaEncryption
            RSA Public Key: (1024 bit)
                Modulus (1024 bit):
                    00:b4:31:98:0a:c4:bc:62:c1:88:aa:dc:b0:c8:bb:
                    33:35:19:d5:0c:64:b9:3d:41:b2:96:fc:f3:31:e1:
                    66:36:d0:8e:56:12:44:ba:75:eb:e8:1c:9c:5b:66:
                    70:33:52:14:c9:ec:4f:91:51:70:39:de:53:85:17:
                    16:94:6e:ee:f4:d5:6f:d5:ca:b3:47:5e:1b:0c:7b:
                    c5:cc:2b:6b:c1:90:c3:16:31:0d:bf:7a:c7:47:77:
                    8f:a0:21:c7:4c:d0:16:65:00:c1:0f:d7:b8:80:e3:
                    d2:75:6b:c1:ea:9e:5c:5c:ea:7d:c1:a1:10:bc:b8:
                    e8:35:1c:9e:27:52:7e:41:8f
                Exponent: 65537 (0x10001)
    Signature Algorithm: md5WithRSAEncryption
        93:5f:8f:5f:c5:af:bf:0a:ab:a5:6d:fb:24:5f:b6:59:5d:9d:
        92:2e:4a:1b:8b:ac:7d:99:17:5d:cd:19:f6:ad:ef:63:2f:92:
        ab:2f:4b:cf:0a:13:90:ee:2c:0e:43:03:be:f6:ea:8e:9c:67:
        d0:a2:40:03:f7:ef:6a:15:09:79:a9:46:ed:b7:16:1b:41:72:
        0d:19:aa:ad:dd:9a:df:ab:97:50:65:f5:5e:85:a6:ef:19:d1:
        5a:de:9d:ea:63:cd:cb:cc:6d:5d:01:85:b5:6d:c8:f3:d9:f7:
        8f:0e:fc:ba:1f:34:e9:96:6e:6c:cf:f2:ef:9b:bf:de:b5:22:
        68:9f
```

61

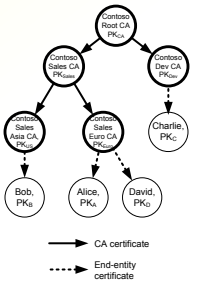
X.509 PKI

- ISO: X.509 standard; IETF: PKIX certificate profile [RFC3280]
- Certification authority (CA) issues certificates
  - Root CA (= trust root, trust anchor)
  - CA can delegate its authority to other CA → CA hierarchy
- Identity certificates bind a principal name to a public key
  - Little-used attribute certificates bind attributes to a name
- Users, computers and services are end entities
- CAs and end entities are principals
  - Each principal has a key pair

62

CA hierarchy

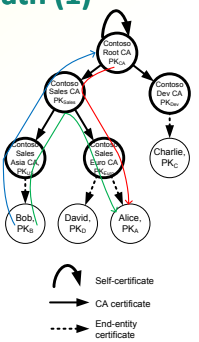
- One root CA
- Each CA can delegate its authority to sub-CAs
- All end-entities trust all CAs to be honest and competent
- Original hope:
  - One global hierarchy
- Reality:
  - One hierarchy per organization
  - Commercial CA without hierarchy, e.g. Verisign



63

Certificate path (1)

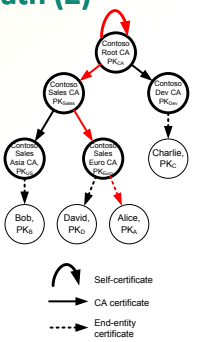
- How can Bob check Alice's PK?
- Original idea:
  - En-entities (like Bob) know their nearest CA
  - Each sub-CA certifies its parent CA in reverse direction
  - CA path from root to Alice meets reverse path from Bob's nearest CA to root at some point → path from Bob to Alice
- Practice:
  - End-entities (Bob) know the root CA
  - Root CA's PK stored as a self-signed certificate



64

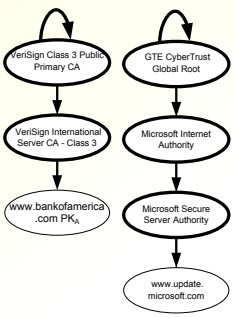
Certificate path (2)

- To verify Alice's signature:
  - Bob needs the entire certificate path from root CA to Alice (3 certificates + self-signed root certificate)
  - The root CA must be configured as Bob's trust root



65

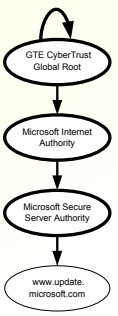
Commercial CAs and web sites



- Web browsers and OSs have a pre-configured list of root CAs
  - Multiple roots!
  - Being on the list enables business of selling certificates
- Some commercial CAs certify customers' CAs, some only end entities
  - Business reasons
  - Security issues of unconstrained delegation
- Wildcard names allow multiple servers to share one certificate
  - E.g. *.contoso.com for www.contoso.com, mail.contoso.com
  - Compromise for cost reasons
  - Not standard, supported by browsers

66

Constrained delegation



- Name constraints:
  - Constrain the authority of a sub-CA to specific subtrees of the name hierarchy
  - Examples: ".microsoft.com" = all MS hosts, "microsoft.com" = one host or all email addresses on that host
  - Permitted and excluded subtrees
- DNS name constraints apply to Subject and SubjectAltName
- Path length constraints limit the depth of the CA hierarchy
- Policy constraints control policies of sub-CAs
- Need to know which constrains are implemented and how they are interpreted by verifiers (e.g. web browsers)

67

Trust vs. authority

- When the same authority allocates names (e.g. host names or email addresses) and maintains the CA, it cannot really be wrong
  - It owns (a part of) the namespace and simply makes decisions about naming subjects
- When the CA merely certifies names given by someone else, as e.g. Verisign often does, it is not really an authority → certificate verifier must trust the CA to be honest and competent
- The commonly used term "trusted authority" makes little sense

68

Cross certification

- How to connect the PKIs of two organizations?
  - In practice, it is rarely done
- Merge into one hierarchy by creating (or hiring) a new root CA to certify both organizational root CAs
- Merge into one hierarchy by making one CA the root and the other a sub-CA
- Root CAs can cross-certify each other
  - Name constrains prevent leaking of authority
  - In effect, both become sub-CAs for each other
- Cross-certification can also be done at a lower level in the hierarchies

69

Certificate revocation

- CA may need to revoke certificates if the conditions for issuing the certificate no longer hold (reasons?)
- Certificate revocation list (CRL) = list of certificate serial numbers
- Who issues the CRL? How to find it?
  - By default, CRL is signed by the CA that issued the certificate
  - CRL distribution point and issuer can be specified in each certificate
- Unlike PGP, X.509 doesn't support key revocation → no mechanism for revoking the root key

70

X.509 CRL fields

- Signature algorithm
- Issuer — name
- This update — time
- Next update — time

For each revoked certificate:

- Serial number
- Revocation date — (how would you use this information?)
- Extensions — reason code etc.

- Signature

71

Revocation delay and CRL size

- Usually, CA issues the CRL and verifiers download it periodically → revocation delay: certificate may be accepted after it has been revoked
- CRL size grows over time until it reaches a stable level
  - Expired certificates can be removed from the CRL after some time (expiration time + maximum clock sync error)
  - Most revocations happen early in the certificate's lifetime
- Delta CRL = download only changes to the previous CRL
- Optimal frequency of CRL distribution depends on the risk caused by revocation delay and cost of CRL distribution
- Online revocation servers now common
- Realtime online validity checking would enable (almost) immediate revocation, but does it make sense?
  - The main advantage of certificates is that they can be used offline, or without frequent real-time access to a server

72

Setting up a PKI

- Potential root CAs:
  - Commercial CA such as Verisign, Thawte, etc. usually charges per certificate
  - Windows root domain controller can act as an organizational CA
  - Anyone can set up their own CA using Windows server or OpenSSL
- The real costs:
  - Distributing the root key (self-signed certificate)
  - Certificate enrolment — need to issue certificates for each user, computer, mobile device etc.
  - Administering a secure CA and CRL server

73

PKI and e-commerce

- In the 90's, PKI was seen as the philosopher's stone of Internet commerce
  - PKI → security → e-commerce → money
- What was successful?
  - Verisign and competitors enabled authentication of web sites → SSL encryption → sniffing of passwords and credit card numbers prevented
- What failed?
  - No global PKI for consumers
  - Internet crosses all organizational and authority boundaries
  - Even if the global PKI existed, what good would it do?
  - Binding contracts are possible without digital signatures
  - Sniffing and spoofing on the network are not the main problems in Internet commerce. Fraud by the store and customer are
  - Risk management and insurance, provided by credit cards, is more important than technical security measures

74

Key-oriented PKIs

Observations about PKIs

- In communication, principals are always represented by their public keys
  → let's redefine: principal = public signature key
- X.509 puts too much emphasis on names. What does a name mean anyway? What matters is access control
- Orange Book definition:
  access control = authentication + authorization.
  But authentication of public keys is easy → focus on authorization
- No global X.509 CA hierarchy exists → must live with local CAs → names have only local meaning
- Since the local authority is a principal, which is a key, all names are relative to a key

Key-oriented PKIs

- Influential research ideas, but no standards yet
- Simple distributed security infrastructure (SDSI):
  - Linked local name spaces: my doctor's secretary, PK_B's Alice
  - Name certificates: PK says PK's doctor is PK_D , PK_B says PK_B's Alice is PK_A
  - Algebra of names and name certificates
- Simple public-key infrastructure (SPKI):
  - Focus on delegation of access rights between public keys
  - Algebra of authorization certificates (5-tuples of <issuer, subject, authority, validity, delegation>)
  - ACL in a server is the root of all authority: PK is allowed to read fileA
  - Authorization certificate: PK delegates to PK2 the right to read fileA
- PolicyMaker:
  - Certificates are programs. They are evaluated by letting them run with the access request and each other as input
  - Later version called KeyNote is more constrained and similar to SPKI

77

Example of SPKI authorization

- ACL on a web page:
  <me, PK_{sales}, read/write, "always", delegate=yes>
- Sales department delegates to Alice:
  <PK_{sales}, PK_A, read, "this week", delegate=yes>
- Alice delegates Charlie:
  <PK_A, PK_C, read/write, "this week", delegate=no>
- Who can access the web page and how?
  - Delegation path me→PK_{sales}→PK_A→PK_B
  - Constrains accumulate along the path
    → PK_A (Alice's key) can read the page this week, and can delegate further
    → PK_B (Bob's key) can read this week but not delegate further

78

Example of SDSI name resolution

- ACL in file server:
  SalesDept's Salesman can read fileA
- Local policy in server:
  my SalesDept is Contoso's SalesDept
  my Contoso is $PK_{Contoso}$
- Head office issues a name certificate:
  $PK_{Contoso}$'s SalesDept is PK_{sales} (signed by $PK_{Contoso}$)
- Sales department issues name certificates:
  PK_{sales}'s Salesman is PK_A (signed by PK_{sales})
  PK_{sales}'s Salesman is PK_B (any name can be a group!)
- Bob requests fileA from the server. He signs the request with PK_B.
- Server resolves the name in the ACL:
  SalesDept's Salesman = Contoso's SalesDept's Salesman = $PK_{Contoso}$'s SalesDept's Salesman = PK_{sales}'s Salesman = {PK_A, PK_B}
  → access allowed
- SPKI and SDSI were merged into one system that is not so simple any more

79

Alternatives to PKI

- Not all authentication is based on a PKI. Other "trust roots":
  - Manual key distribution, e.g. for permanent IPsec tunnel or RADIUS
  - Password authentication of human users
  - Online authentication servers, e.g. Kerberos
  - Pseudonymity — create new id created for each service and authenticate returning users
  - Leap of faith -– assume there is no attacker on the first time, e.g. SSH
  - Self-certifying identifiers — public key as identifier

80

Exercises

- How to prevent SMTP spoofing without end-to-end cryptography? What can be filtered at SMTP servers and what cannot?
- Does signing of emails help spam control?

81

Exercises

- Install an OpenPGP implementation (e.g. GPG). How do you check that the binary or source code has not been tampered with? Would you use PGP to verify the signature or fingerprint of the installation package? Could there be other compromised software (spyware) on your machine?
- Set up your own CA e.g. using OpenSSL and issue certificates to your own web server or some other service that uses TLS/SSL authentication. What decisions did you have to make on the way? What open questions do you have after the experience?
- Consider setting up a PKI in a place where you have worked/studied. How would you distribute the root key and organize certificate enrolment?

82