Network security

Cryptography Overview
Public/Private Key Cryptography
Authentication
Key Distribution
Message Integrity
Secure email

1

What is a cryptosystem?

- $K = \{0,1\}^l$
- $P = \{0,1\}^m$
- $C' = \{0,1\}^n$, $C \subseteq C'$
- $E: P \times K \rightarrow C$
- $D: C \times K \rightarrow P$
- $\forall p \in P, k \in K: D(E(p,k),k) = p$
- It is infeasible to find $F: P \times C \rightarrow K$

Lets start again. This time in English....

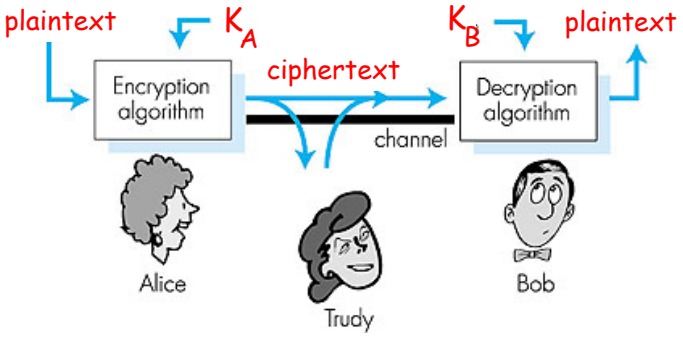
2

What is a cryptosystem (2.)?

- A pair of algorithms that take a key and convert plaintext to ciphertext and back
  - Plaintext: text to be protected
  - Ciphertext: should appear like random gibberish
- Requires sophisticated math!
  Do not try to design your own algorithms!

3

The language of cryptography



- Symmetric key crypto:
  sender and receiver keys identical and secret
- Public-key crypto:
  encrypt key public, decrypt key secret

4

Symmetric key cryptography: Example

Substitution cipher: substituting one thing for another

- Monoalphabetic cipher: substitute one letter for another

```
plaintext:  abcdefghijklmnopqrstuvwxyz
            |                       |
ciphertext: mnbvcxzasdfghjklpoiuytrewq
```

E.g.:

```
Plaintext: bob. i love you. alice
ciphertext: nkn. s gktc wky. mgsbc
```

- Attack: letter frequency analysis

5

Codes vs. Ciphers

- Ciphers operate syntactically, on elements of an alphabet (letters) or groups of "letters": A $\rightarrow$ D, B $\rightarrow$ C, etc.
- Codes operate semantically, on words, phrases, or sentences, e.g., per codebooks

6

Properties of a good cryptosystem

- There should be no way short of enumerating all possible keys to find the key from any reasonable amount of ciphertext and plaintext, nor any way to produce plaintext from ciphertext without the key
- Enumerating all possible keys must be infeasible
- The ciphertext must be indistinguishable from true random values

7

Milestones in modern cryptography

- 1883 Kerckhoffs' principles
- 1917-1918 Vernam/Mauborgne cipher (one-time pad)
- 1920s-1940s Mathematicization and mechanization of cryptography and cryptanalysis
- 1973 U.S. National Bureau of Standards issues a public call for a standard cipher; this led to the adoption of the Data encryption Standard (DES)
- 1976 Diffie and Hellman describe public key cryptography

8

Kerckhoffs' law

- „The system must not be required to be secret, and it must be able to fall into the hands of the enemy without inconvenience"
- In other words, the security of the system must rest entirely on the secrecy of the key not the algorithm itself

9

Vernam/Mauborgne cipher

- Exclusive-OR a key stream tape with the plaintext
- Online encryption of teletype traffic, combined with transmission
- For a one-time pad – which is provably secure – use true-random keying tapes and never reuse the keying material
- Problem: how to get good long one-time pads
  - Reuse of keying material ⇒ stream cipher
  - Key stream via algorithm ⇒ no one-time pad

10

Mathematicization and mechanization

- Mechanical encryptors
  (Vernam, Enigma, Hagelin, Scherbius)
- Mathematical cryptanalysis
  (Friedman, Rejewski et al., Bletchley Park)
- Machine-aided cryptanalysis
  (Friedman, Turing et al.)

11

Standardized ciphers

- Until the 1970s, most strong ciphers were government secrets
- Spread of computers ⇒ new threads
  (Reportedly, soviets eavesdropped on U.S. grain negotiators' conversations)
- NBS (now called NIST) issued public call for cipher; eventually IBM responded

⇒ eventual result – via secret process - DES

12

Public key cryptography

- Merkle invents a public key distribution scheme
- Diffie and Hellman invent public key encryption and digital signatures, but do not devise a suitable algorithms with all desired properties
- Rivest, Shamir, and Adelman invent their algorithm RSA soon after
- British GCHQ invented „non-secret encryption" a few years earlier
- There are claims, but no evidence, that the NSA invented it even earlier

13

What we have today

- Encryption is completely computerized and operates on bits
- Basic primitives can be combined to produce powerful results
- Encryption is by far the strongest weapon of computer security
  Host and OS software is by far the weakest link
- Bad software trumps good crypto

14

Block ciphers

- Operate on a fixed-length set of bits
- Output block size generally the same as input size
- Well-known examples
  - DES (56-bit keys; 64-bit block size)
  - AES (128-, 192-, and 256-bit keys; 128-bit block size)
- Basic structure
  - Optional key scheduling – converts key to internal form
  - Multiple rounds of combining plaintext with key
  - DES 16 rounds
  - AES 10-14 rounds (depending on key length)

15

Symmetric key crypto: DES

- US encryption standard [NIST 1973]
- 56-bit symmetric key, 64 bit plaintext input
- How secure is DES?
  - DES Challenge: 56-bit-key-encrypted phrase ("Strong cryptography makes the world a safer place") decrypted (brute force) in 4 months
  - No known "backdoor" decryption approach
  - Block size too small
  - Depends on bit-manipulation $\Rightarrow$ slow for software
- Making DES more secure
  - Use three keys sequentially (3-DES) on each datum
  - Use cipher-block chaining

16

DES

DES encryption

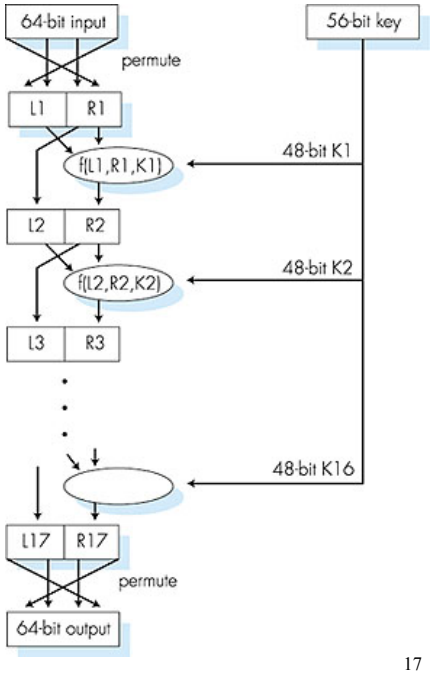
initial permutation

16 identical "rounds" of function application, each using different 48 bits of key

final permutation

DES decryption

run the "rounds" backwards



17

Advanced Encryption Standard (AES)

- NIST issued an open call for submissions
- 15 ciphers were submitted (from all over)
- Several open conferences (+ NSA private evaluation)
- 5 ciphers were eliminated as not secure enough
- 5 more were dropped for inefficiency or low security margin
- Of the 5 finalists, Rijndael – from Belgium – was chosen
  - Good security
  - Very high efficiency across a wide range of platforms

18

Rijndael

- Input block viewed as byte array (2-dim matrix)
- Each round consists of a series of simple, byte-oriented operations:
  - ByteSubstitution, ShiftRow, MixColumn, AddRoundKey
- Key is mixed with the entire block in each round
- Basic operations are individually reasonably tractable mathematically, but are combined in a hard-to-invert fashion
- Basis: Field theory and GF(2^8)

19

How to use a block cipher

- Direct use of a block cipher is inadvisable
  - Enemy can build up „code book" of plaintext/ciphertext equivalents
  - Only works for messages that are a multiple of the block size
- Solution: 5 standard modes of operation
  - Electronic Code Book (ECB)
  - Cipher Block Chaining (CBC)
  - Cipher Feedback (CFB)
  - Output Feedback (OFB)
  - Counter (CTR)

20

Electronic Code Book

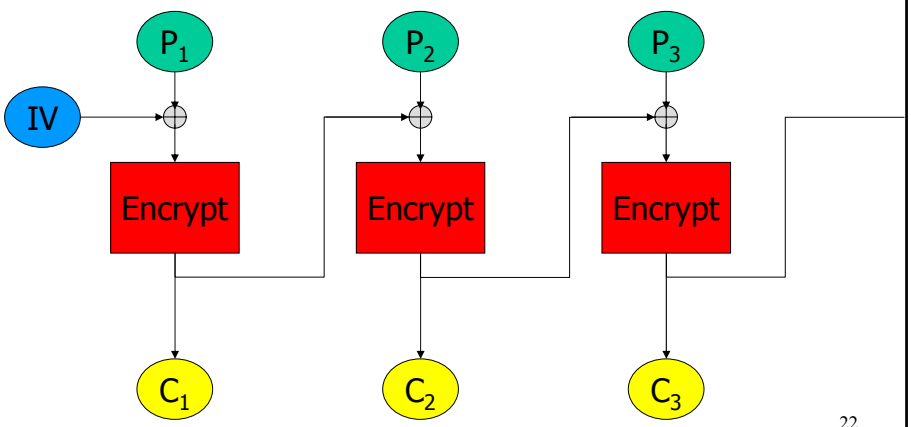
- Direct use of block cipher
- Used primarily to transmit encrypted keys
- Very weak for general-purpose encryption

- Problem: block substitution attack

21

Cipher Block Chaining (CBC)

- IV: Initialization vector, P: plaintext, C: ciphertext



22

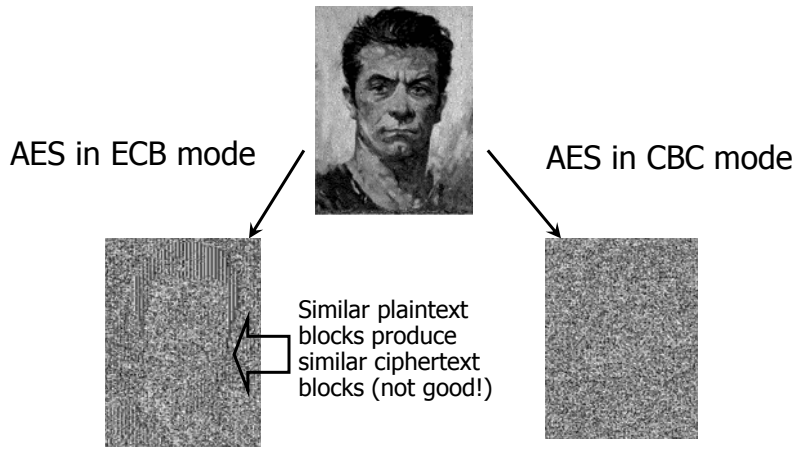
Cipher Block Chaining (2.)

- Properties of CBC
  - Ciphertext of each encrypted block depends on the plaintext of all preceding blocks
  - Subsets of blocks appear valid and will decrypt properly
  - Message integrity has to be done otherwise
- CBC and electronic voting
  [Kohno,Stubblefield,Rubin,Wallach]
  - Found in the source code for Diebold voting machines:
  - ```
    DesCBCEncrypt((des_c_block*)tmp,
    (des_c_block*)record.m_Data, totalSize,
    DESKEY, NULL, DES_ENCRYPT)
    ```

23

ECB vs. CBC

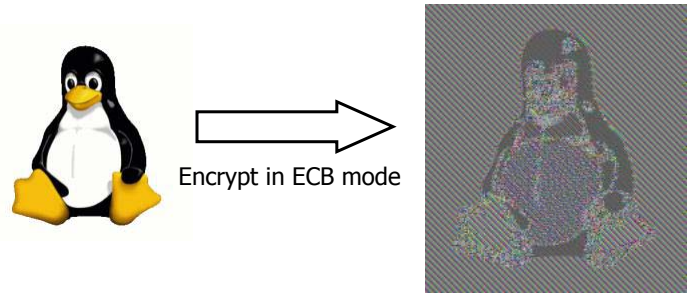
[Picture due to Bart Preneel]



24
    ```

Information leakage in ECB mode

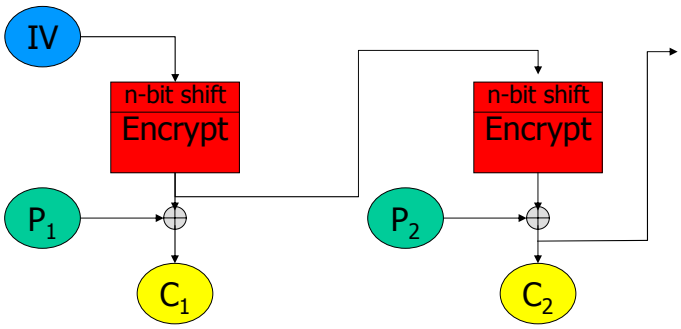
[Wikipedia]



25

n-Bit Cipher Feedback

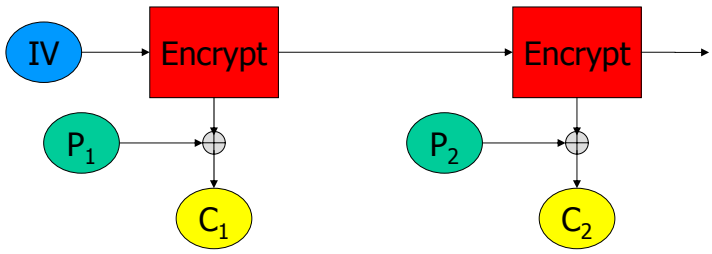
- Add n-bit shift and move Encrypt operation before X-OR operator
- Retains some of the previous cycle's ciphertext
- Copes gracefully with deletion of n-bit unit (bit errors)



26

n-Bit Output Feedback

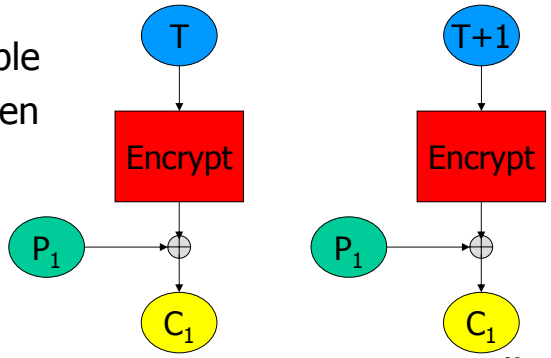
- No error propagation
- Active attacker can make controlled changes to plaintext
- OFB is a form of stream cipher



27

Counter mode

- Another form of stream cipher
- Counter often split in message and block number
- Active attack can make controlled changes to plaintext
- Highly parallelizable
- No linkage between stages
- Vital: Counter never to repeat



28

Which mode for what task

- General file or packet encryption: CBC
  $\Rightarrow$ Input must be padded to n $\times$ cipher block size
- Risk of byte or bit deletion: CFB_8 or CFB_1
- Bit stream: noisy line and error propagation is undesirable: OFB
- Very high-speed data: CTR
- Needed in most situations: integrity checks
  - Actually needed almost always
  - Attack on integrity $\Rightarrow$ attack on confidentiality
  - Solution: separate integrity check along with encryption

29

Combined modes of operation

- Integrity checks require a separate pass over data
- Cannot be parallelized: undesired burden
- Combined modes:
  - Galois Counter Mode (GCM)
  - Counter with CBC-MAC (CCM)

30

When is a cipher "secure"?

- Hard to recover the key?
  - What if attacker can learn plaintext without learning the key?
- Hard to recover plaintext from ciphertext?
  - What if attacker learns some bits or some function of bits?
- Fixed mapping from plaintexts to ciphertexts?
  - What if attacker sees two identical ciphertexts and infers that the corresponding plaintexts are identical?
  - Implication: encryption must be randomized or stateful

31

How can a cipher be attacked?

- Assume the attacker knows the encryption algorithm and wants to decrypt some ciphertext
- Main question: what else does attacker know?
  - Depends on the application in which cipher is used!
- Ciphertext-only attack
- Known-plaintext attack (stronger)
  - Knows some plaintext-ciphertext pairs
- Chosen-plaintext attack (even stronger)
  - Can obtain ciphertext for any plaintext of his choice
- Chosen-ciphertext attack (very strong)
  - Can decrypt any ciphertext except the target

32

Stream ciphers

- Operation:
  - Key stream generator produces a sequence S of pseudo-random bytes
  - Key stream bytes are combined (usually via XOR) with plaintext bytes
- Properties:
  - Very good for asynchronous traffic
  - Best-known stream cipher RC4 (used, e.g., in SSL)
  - Key stream must never be reused for different plaintexts

33

RC4

- Extremely efficient
- After key setup, it just produces a key stream
- Internal state: 256-byte array plus two integers

```
for (c=0; c < buffer_len; c++) {
      x = (x+1) % 256
      y = (state[x] + y) % 256;
      swap_byte(&state[x], &state[y]);
      xorIndex = (state[x] + state[y])% 256;
      buffer_ptr[c] ^= state[xorIndex];
}
```

- No resynchronization except via rekeying + starting over
- Note:
  known weaknesses if used other than as stream cipher

34

Cipher strength

- Never stronger than its key lengths: if there are too few keys, an attacker can enumerate all possible keys
- DES has 56 bits – arguably too few in 1976; far too few today
- Strength of cipher depends on how long it resists attacks
- No good reason to use less than 128 bits
- NSA rates 128-bit AES as good enough for SECRET traffic; 256-bit AES is good enough for TOP-SECRET traffic
- But a cipher can be considerably weaker! (A monoalphabetic cipher over all bytes has a 1684-bit key, but is trivially solvable.)

35

CPU speed vs. key size

- Adding one bit to the key doubles work for brute force attack
- Effect on encryption time is often negligible or even free
- It costs nothing to use a longer RC4 key
- Going from 128-bit AES to 256-bit AES takes (at most) 40% longer for en-/decryption but increases the attacker's effort by a factor of 2^{128}
- Using triple DES costs 3× more to encrypt, but increases the attacker's effort by a factor of 2^{112}
- Moore's Law favors the defender!

36

Public Key Cryptography

Symmetric key crypto

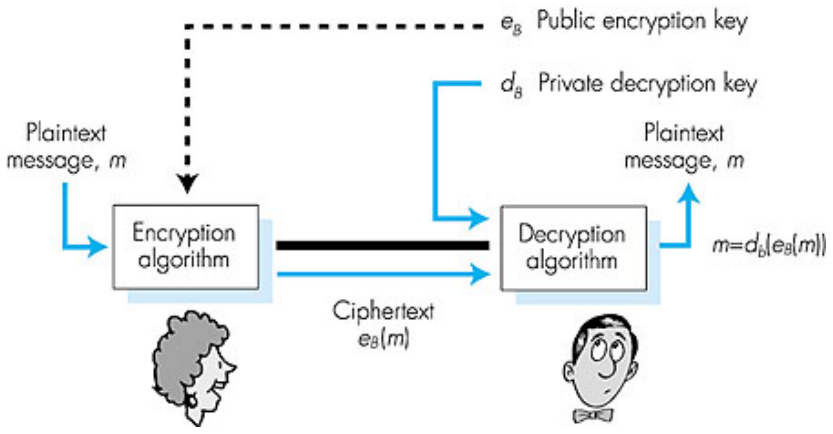
- Requires sender, receiver to know shared secret key
- Q: how to agree on key in first place (particularly if never "met")?
- Q: what if key is stolen?
- Q: what if you run out of keys?
- Q: what if A doesn't know she wants to talk to B?

Public key cryptography

- Radically different approach [Diffie-Hellman76, RSA78]
- Sender, receiver do *not* share secret key
- Encryption key *public* (known to *all*)
- Decryption key private (known only to receiver)
- Allows parties to communicate without prearrangement

37

Public key cryptography



38

Public key encryption algorithms

Two inter-related requirements:

1. need $d_B(\cdot)$ and $e_B(\cdot)$ such that

   $d_B(e_B(m)) = m$

2. need public and private keys for $d_B(\cdot)$ and $e_B(\cdot)$

Best-known public key system
RSA: Rivest, Shamir, Adelman algorithm

39

RSA: Choosing keys

1. Choose two large prime numbers *p, q.* (e.g., 1024 bits each)
2. Compute $n = pq$, $z = (p-1)(q-1)$
3. Choose e (with $e<n$) that has no common factors with z. (*e, z* are "relatively prime").
4. Choose d such that $ed-1$ is exactly divisible by z. (in other words: $ed \bmod z = 1$).
5. *Public* key is (n,e). *Private* key is (n,d).

Security of the system relies on the difficulty of factoring n.
Finding primes is easy; factoring is believed to be hard!

40

RSA: Encryption, decryption

0. Given (n,e) and (n,d) as computed above

1. To encrypt bit pattern, m, compute
   $c = m^e \bmod n$ (i.e., remainder when m^e is divided by n)

2. To decrypt received bit pattern, c, compute
   $m = c^d \bmod n$ (i.e., remainder when c^d is divided by n)

Magic happens! $m = (m^e \bmod n)^d \bmod n$

41

RSA example

Bob chooses $p=5$, $q=7$. Then $n=35$, $z=24$.
$e=5$ (so e, z relatively prime).
$d=29$ (so $ed-1$ exactly divisible by z.

encrypt:

letter	m	m^e	$c = m^e \bmod n$
l	12	248832	17

decrypt:

c	c^d	$m = c^d \bmod n$	letter
17	48196857210675091509141182522307169 7	12	l

42

RSA: Why? $m = (m^e \bmod n)^d \bmod n$

Number theory result: If p,q prime, $n = pq$, then

$$x^y \bmod n = x^{y \bmod (p-1)(q-1)} \bmod n$$

$$(m^e \bmod n)^d \bmod n = m^{ed} \bmod n$$

$$= m^{ed \bmod (p-1)(q-1)} \bmod n$$

(using number theory result above)

$$= m^1 \bmod n$$

(since we chose ed to be divisible by $(p-1)(q-1)$ with remainder 1)

$$= m$$

43

RSA practicalities

- Classical public key usage
  - Alice publishes her public key in the phone book
  - Bob prepares a message and encrypts it with that key by doing a large exponentiation
  - Alice uses her private key to do a different large exponentiation
  - It's not that simple...
- RSA complexities
  - RSA calculation is very expensive!
  - RSA is amenable to mathematical attacks; don't use the wrong numbers

44

RSA practicalities (2.)

- Use case
  - Bob generates a random key k for a conventional cipher
  - Bob encrypts the message with k
  - Bob pads k to k‘ to make it 512 bits long and encrypts it with Alice‘s public key
  - Bob transmits this and the ciphertext to Alice
  - Alice uses her private key to recover k‘, removes the padding and uses k to decrypt the ciphertext
- Problem: forward secrecy
  - If an endpoint is compromised, can an enemy read old conversations?
  - Solution use schemes that provide perfect forward secrecy, such as Diffie-Hellman key exchange

45

Diffie-Hellmann key exchange

- Agree on a large (> 1024-bit) prime p, usually of the form 2q+1 where q is also prime
- Find a generator g of the group „integers modulo p“.
- Alice picks a large random number x and sends Bob g^x mod p. Bob picks a large random number y and sends Alice g^y mod p.
- Alice (Bob) calculates k = $(g^y)^x = g^{xz}$ mod p;
- If x and z are really random, they cannot be recovered if Alice or Bob‘s machine is hacked
- Eavesdroppers cannot calculate x from g^x mod p, and hence cannot get the shared key.
  This is called the discrete logarithm problem (DLP).

46

Random numbers

- Crucial for cryptography
- As random as possible: prefer true-random numbers over pseudo-random ones
  - Physical process are best
    - Thermal noise in amplifiers (used by current CPUs)
    - Oscillator jitter
    - Radioactive decay
    - ...
- Often true-random numbers are used to seed a cipher

47

Authentication

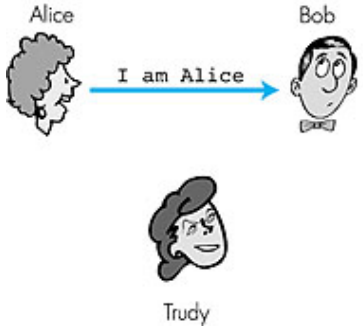
- With whom are you communicating?
  - Identification – who they claim to be
  - Authentication – proof
    - Something you know (i.e., passwords)
    - Something you have (i.e., token or smart card)
    - Something your are (biometrics)
  - Authorization – what they can do

48

Authentication – how to do it?

Goal: Bob wants Alice to "prove" her identity to him

Protocol ap1.0: Alice says "I am Alice"



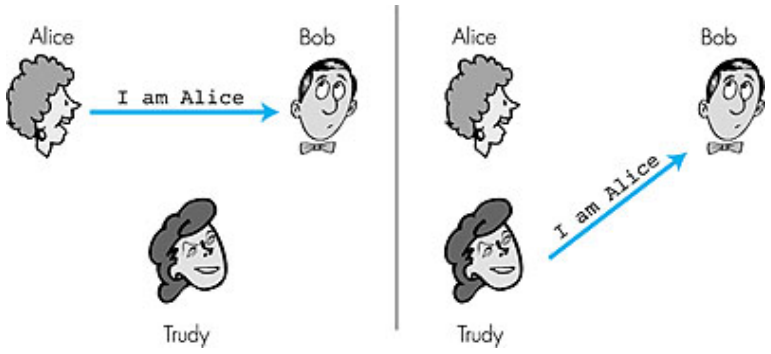
Failure scenario??

49

Authentication

Goal: Bob wants Alice to "prove" her identity to him

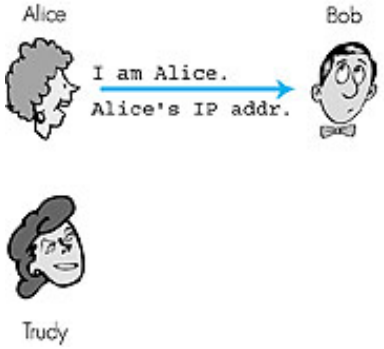
Protocol ap1.0: Alice says "I am Alice"



50

Authentication: another try

Protocol ap2.0: Alice says "I am Alice" and sends her IP address along to "prove" it.

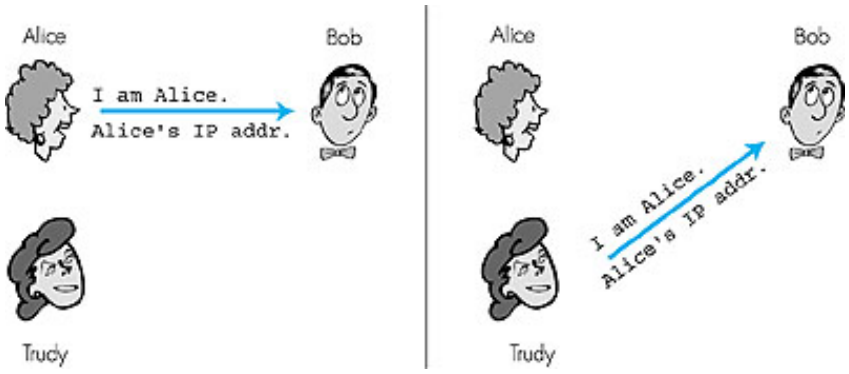


Failure scenario??

51

Authentication: another try

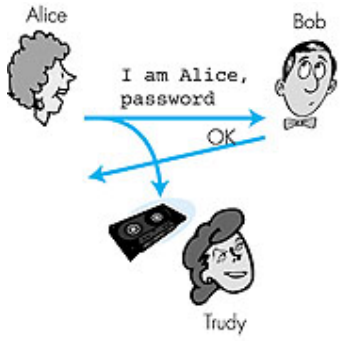
Protocol ap2.0: Alice says "I am Alice" and sends her IP address along to "prove" it.



52

Authentication: another try

Protocol ap3.0: Alice says "I am Alice" and sends her secret password to "prove" it.

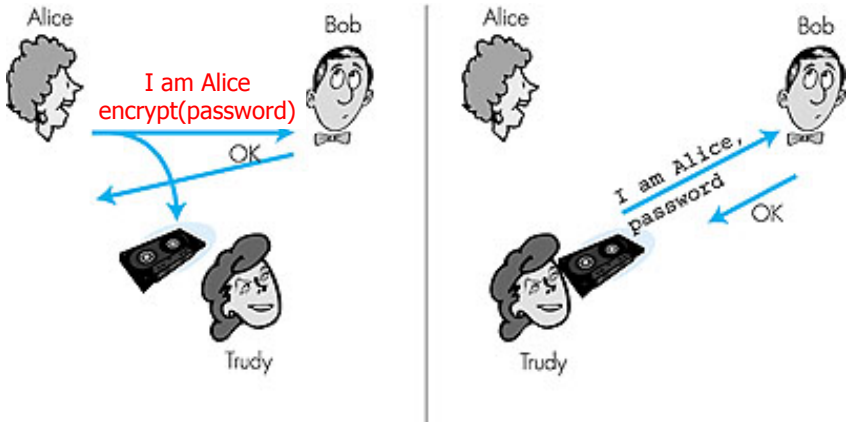


Failure scenario?

53

Authentication: yet another try

Protocol ap3.1: Alice says "I am Alice" and sends her *encrypted* secret password to "prove" it.



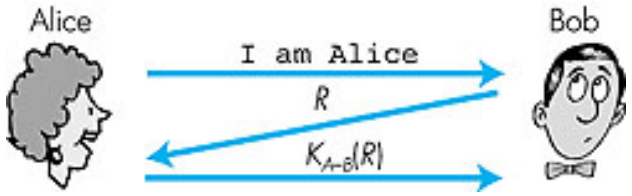
54

Authentication: yet another try

Goal: avoid playback attack

Nonce: number (R) used only once in a lifetime

ap4.0: to prove Alice "live", Bob sends Alice nonce, R. Alice must return R, encrypted with shared secret key



Failures, drawbacks?

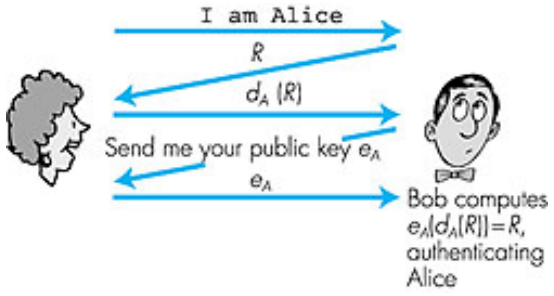
55

Authentication: ap5.0 challenge/response

ap4.0 requires shared symmetric key

- Problem: how do Bob, Alice agree on key
- Can we authenticate using public key techniques?

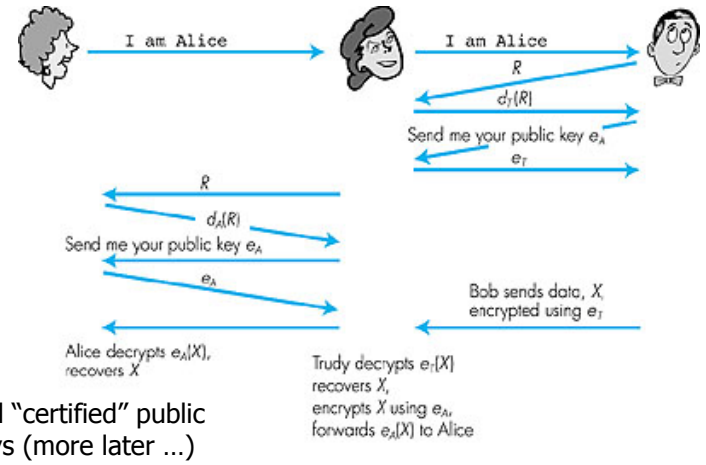
ap5.0: use nonce, public key cryptography (e.g., RSA)



56

Authentication: ap5.0 - security hole

Man (woman) in the middle attack: Trudy poses as Alice (to Bob) and as Bob (to Alice)



Need "certified" public keys (more later …)

57

Shared secrets

- Something you know or you have
- Passwords
  - Very guessable – 10-40%, according to studies [MorrisThompson]
    - Dictionary attack: passwords are not truly random
    - With 52 upper- and lower-case letters, 10 digits and 32 punctuation symbols, there are $94^8 \approx$ 6 quadrillion possible 8-character passwords
    - Humans like to use dictionary words, human and pet names ≈ 1 million common passwords
  - Forgettable – don't write them down!
  - Susceptible to eavesdropping (sniffing)
    - Trivial on wireless/easy on Ethernets
    - Best defense: encryption / one-time passwords schemes
  - Can be attacked via social engineering/phishing

58

Passwords (2.)

- Cryptographic key
- Tokens
  - Embedded shared secret used as cryptographic key
  - Challenge/response: server sends random number; token encrypts it
  - One-time passwords (never reused)
  - Frequently used together with a PIN (to guard against loss / theft)
  - Cannot be social engineered
    but – http://fob.webhop.net/

59

Original UNIX password system

- Uses DES encryption as if it were a hash function
  - Encrypt NULL string using password as the key
    - Truncates passwords to 8 characters!
  - Artificial slowdown: run DES 25 times
  - Modern UNIXes to use, e.g., salted MD5 hash function
- Problem: passwords are not truly random

60

Dictionary attack

- Password file /etc/passwd is world-readable
  - Contains user IDs and group IDs which are used by many system programs
- Dictionary attack is possible because many passwords come from a small dictionary
  - Attacker computes H(word) for every dictionary word; checks if the result is in password file (johntheripper)
  - 1,000,000-word dictionary; 10 guesses per second brute-force online attack
    => 50,000 seconds (14 hours) on average
    - This is very conservative. Offline attack is much faster!

61

Shadow Passwords

shmat:x:14510:30:Vitaly:/u/shmat:/bin/csh

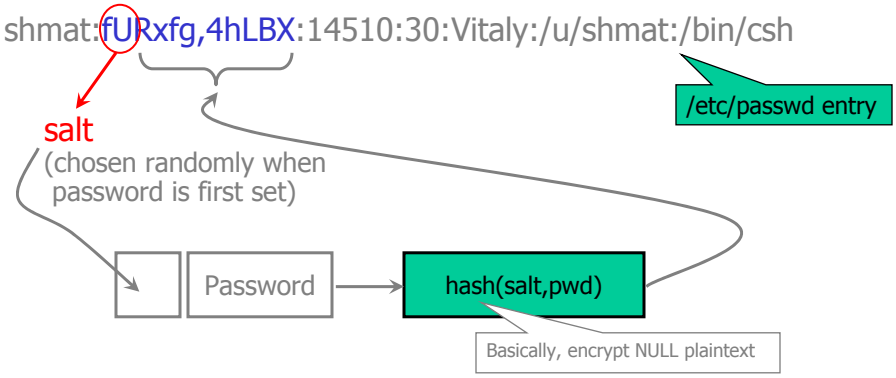
Hashed password is not stored in a world-readable file

/etc/passwd entry

- Store hashed passwords in /etc/shadow file which is only readable by system administrator (root)
- Add expiration dates for passwords
- Early shadow implementations on Linux called the login program which had a buffer overflow!

62

Salt



- Users with the same password have different entries in the password file
- Dictionary attack is still possible!

63

Advantages of salting

- Without salt:
  - Attacker can pre-compute hashes of all dictionary words
  - Same hash function on all UNIX machines; Identical passwords hash to identical values
- With salt:
  - Attacker must try all dictionary words for each salt value in the password file
  - With 12-bit random salt, same password can hash to 2^{12} different hash values

64

Strengthening passwords

- Add biometrics
  - For example, keystroke dynamics or voiceprint
  - Revocation is often a problem with biometrics
- Graphical passwords
  - Goal: increase the size of memorable password space
- Rely on the difficulty of computer vision
  - Face recognition is easy for humans, hard for machines
  - Present user with a sequence of faces, he must pick the right face several times in a row to log in

65

Graphical passwords

- Images are easy for humans to process and remember
  - Especially if you invent a memorable story to go along with the images
- Dictionary attacks on graphical passwords are difficult
  - Images are believed to be very "random" (is this true?)
- Still not a perfect solution
  - Need infrastructure for displaying and storing images
  - Shoulder surfing

66

Active attacks

- Simple one-time passwords not sufficient against active attackers
- Last-digit guessing attack
  - Watch user start to log in with SecurID
  - Start in parallel 10 sessions
  - Guess the last digit faster than the user
- Connection hijacking
  - Use ARP-spoofing to route traffic through you
  - Wait for victim to login
  - Imitate victim; do not send packets to real user
  - Demonstrated in lab in 1995;
    now doable with off-the-shelf hacker code!
  - Defense: cryptographically protect all packets!

67

Cryptography and authentication

- Some way to use cryptographic key to prove who you are
- Can go beyond simple schemes given above
- Can use symmetric or public key schemes
- Most public key schemes use certificates (More later)

68

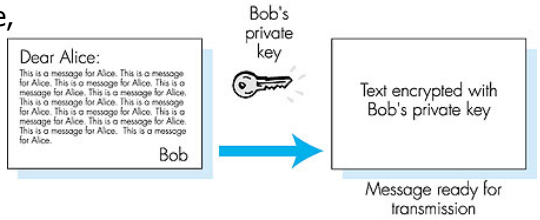
Digital signatures

Cryptographic technique analogous to hand-written signatures.

- Sender (Bob) digitally signs document, establishing he is document owner/creator.
- Verifiable, non forgeable: recipient (Alice) can verify that Bob, and no one else, signed document.

Simple digital signature for message m:

- Bob encrypts m with his private key d_B, creating signed message, $d_B(m)$.
- Bob sends m and $d_B(m)$ to Alice.



69

Digital signatures (more)

- Suppose Alice receives msg *m*, and digital signature $d_B(m)$
- Alice verifies *m* signed by Bob by applying Bob's public key e_B to $d_B(m)$ *then* checks $e_B(d_B(m)) = m$.
- If $e_B(d_B(m)) = m$, whoever signed *m* must have used Bob's private key.

Alice thus verifies that:

- Bob signed *m*.
- No one else signed *m*.
- Bob signed m and not *m'*.

Non-repudiation:

- Alice can take *m*, and signature $d_B(m)$ to court and prove that Bob signed *m*.

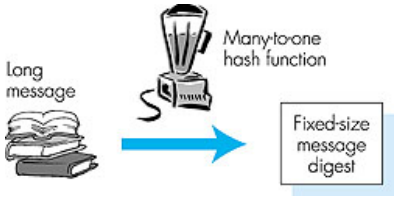
70

Digital vs. real signatures

- Real signatures are strongly bound to the person, and weakly bound to the data
- Digital signatures are strongly bound to the data, and weakly bound to the person – what if the key is stolen (or deliberately leaked)?
- A better term: digital signature algorithms provide non-repudiation

71

Message digests



Computationally expensive to public-key-encrypt long messages

Goal: fixed-length, easy to compute digital signature, "fingerprint"

- Apply hash function H to *m*, get fixed size message digest, *H(m).*

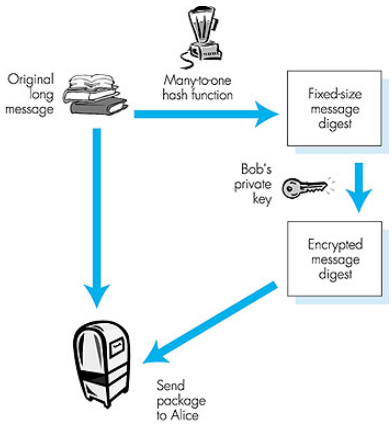
Hash function properties:

- Many-to-1
- Produces fixed-size msg digest (fingerprint)
- Given message digest x, computationally infeasible to find m such that x = H(m)
- Computationally infeasible to find any two messages m and m' such that H(m) = H(m'). (Collision!)

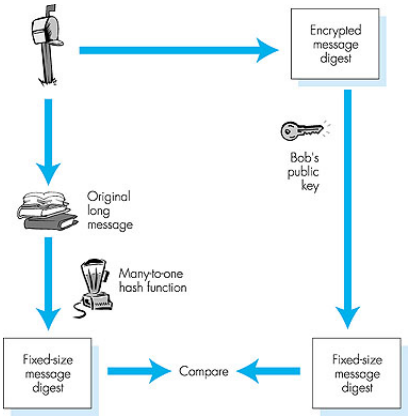
72

Digital signature = Signed message digest

Bob sends digitally signed message:



Alice verifies signature and integrity of digitally signed message:



73

Hash function algorithms

- Internet checksum would make a poor message digest.
  - Too easy to find two messages with same checksum.
- MD5 hash function widely used.
  - Computes 128-bit message digest in 4-step process.
  - Arbitrary 128-bit string x, appears difficult to construct msg m whose MD5 hash is equal to x.
- SHA-1 is also used.
  - US standard
  - 160-bit message digest

74

Cryptographic hash functions

- Produce relatively-short, fixed-length output string from arbitrarily long input
- Computationally infeasible to find two different input strings that hash to the same value (strong collision resistant)
- Computationally infeasible to find any input string that hashes to a given value (weak collision resistant)
- Strength roughly equal to half the output length
- Best-known examples:
  MD5 (128 bits), SHA-1 (160 bits), SHA-256/384/512
- 128 bits and shorter are not very secure for general usage

75

Recent developments

- At CRYPTO'04 several hash functions were cracked by Wang et al.
  - More precisely, collisions where found
  - MD4, MD5, HAVAL-128, RIPEMD, and SHA-0
  - SHA-0 was known to be flawed
    (replaced by SHA-1 in 1994)
- In 2005 Wang et al. showed that SHA-1 was considerably weaker than its design strength
- Are SHA-256/384/512 still secure?
- What about MD5
  - Commonly used (e.g., passwords)
  - Weaknesses suspected since a long time

76

The birthday paradox

- How many people need to be in a room for the probability that two will have the same birthday to be > 0.5?
  - Naive answer: 183
  - Correct answer: 23
  - The question is not „who has the same birthday as x"
  - The question is „are there two arbitrary folks who's birthday is on the same day"

77

The birthday attack

- Signature attacks:
  - Prepare lots of variant contracts and see if two have the same hash
- Block ciphers
  - How many blocks can be encrypted with one key before one gets collisions? $2^{B/2}$ (B = block size)
  - 2^{32} blocks for DES, 2^{64} for AES
  - 275 Gb; on a 1Gb/sec network less than 5 minutes!

78

Practical stunts from MD5 collision

- General style of attack for exploiting hash function collisions
- Create two prologues to a message file, using a collision assigned to some variable in each version
- In the body of the message, conditionally display one version or the other, depending on that variable
- Get the harmless version signed
- Transmit the harmful one
- If no conditionals in your page description language, put the collision in a font definition file or the like

79

Elliptic curve cryptography

- Public key and D/H algorithms, but based on more complex math
- Considerably more security per key bit; allows for shorter keys (e.g., 160 bits instead of 1024 bits)
- More importantly, allows for much more efficient computation
- Recently endorsed by NSA
- Many patents in this space

80

Rough table of key length equivalences

[Orman and Hoffman, RFC 3766]

Symmetric Key Size (bits)	RSA or DH Modulus Size (bits)
70	947
80	1228
90	1553
100	1926
150	4575
200	8719
250	14596

81

Message integrity

- Need a way to prevent tempering the message
- Use key and cryptographic hash to generate Message Authentication Code (MAC)
- Simpler solutions do not work
  - Example:
    - Append cryptographic hash to some plaintext, and encrypt all with CBC mode

      $\{P, H(P)\}_K$
    - Can be attacked via chosen plaintext attack

82

Message integrity attack

- Enemy picks some plaintext P and tricks you into encrypting it (this is real, e.g., WEP!)
- You transmit
  $\{prefix, P, suffix, H(prefix, P, suffix)\}_K$
- But P is of the form
  prefix, P', suffix, H(prefix, P' suffix)
- A CBC subset will have the checksum one is looking for!

83

HMAC

- Use cryptographic hash functions to build MACs
- HMAC: best known construct – provably secure under minimal assumptions
- HMAC(m, k) = H(k, H(k,m))
  where H is a cryptographic hash function
- Authentication key must be distinct from confidentiality key!
- Frequently, the output of HMAC is truncated

- (CBC MAC: use last CBC block as authentication
  CBC cut-and-past attack works if same key is used)

84

Order of encryption and MACing

- Three choices:
  - $\{P\}_K$, $M_{K'}(P)$
  - $\{P, M_{K'}(P)\}_K$
  - $\{P\}_K$, $M_{K'}(\{P\}_K)$
- Last option is the most secure (provably so) – always calculate a MAC on the ciphertext
- MACs are cheaper than decryption
  Integrity can be verified first
  Discard the message if MAC is bogus
  Otherwise decrypt
- Example: email signing and encryption

85

What to MAC?

- MAC should include all the ciphertext
- Often MAC should include plaintext metadata
- Example:
  - Plaintext message length (before padding)
  - MAC should cover that length

86

Key lifetimes

- Confidentiality key: as long as data is sensitive
- Digital signature:
  as long as one needs to prove authorship
- MAC key: only for the duration of the session

87

Trusted intermediaries

Problem:

- How do two entities establish shared secret key over network?

Solution:

- Trusted key distribution center (KDC) acting as intermediary between entities

Problem:

- When Alice obtains Bob's public key (from web site, e-mail, diskette), how does she know it is Bob's public key, not Trudy's?

Solution:

- Trusted certification authority (CA)

88

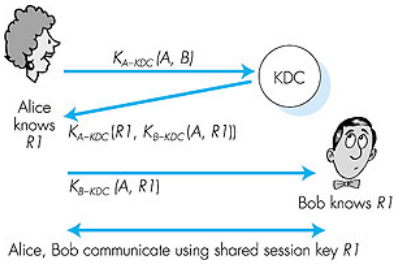
Certificates

- Digitally-signed message containing an identity and a public key – prevents tampering
- Why trust it
  - Who signed it?
  - Why do you trust them?
  - How do you know the public key of the Certificate Authority (CA)
  - Some public key (known as trust anchor) must be provided out-of-band – trust has to start somewhere

89

Key distribution center (KDC)

- Alice, Bob need shared symmetric key.
- KDC: server shares different secret key with each registered user.
- Alice, Bob know own symmetric keys, $K_{A\text{-}KDC}$ $K_{B\text{-}KDC}$, for communicating with KDC.

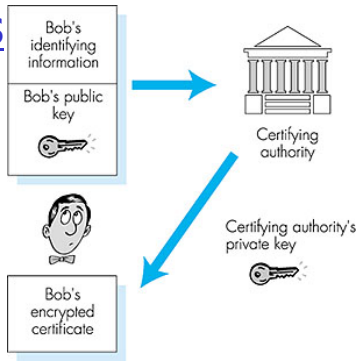


- Alice communicates with KDC, gets session key R1, and $K_{B\text{-}KDC}(A,R1)$
- Alice sends Bob $K_{B\text{-}KDC}(A,R1)$, Bob extracts R1
- Alice, Bob now share the symmetric key R1.

90

Certification authorities

- Certification authority (CA) binds public key to particular entity.
- Entity (person, router, etc.) can register its public key with CA.
  - Entity provides "proof of identity" to CA.
  - CA creates certificate binding entity to public key.
  - Certificate digitally signed by CA.



- When Alice wants Bob's public key:
- gets Bob's certificate (Bob or elsewhere).
- Apply CA's public key to Bob's certificate, get Bob's public key

Certification authorities (2.)

- Politics:
  - Who picks CAs? No one and every one
  - Your browser has some CAs built in – because the CA paid the browser vendor enough?
  - Matt Blaze: "A commercial certificate authority can be trusted to protect you from anyone from whom they won't take money"
- Who gets certificates?
  - How to prove your identity to the CA? How to verify it? How much liability?
- CAs are part of the Public Key Infrastructure (PKI)
- Alternatives?
  - Web of Trust
    - Get certificates from parties whom you know
  - Issue certificates to your own users

Certificate hierarchy vs. Web of Trust

- CA hierarchy
  - Most CAs are tree-structured
    (there exists a bottom up suggestion for building CAs)
  - Top-level CAs can use bridge CAs to cross-certify each other
- Web of Trust:
  - Arbitrarily complex graph
  - Users can verify path to as many trust anchors as they wish

93

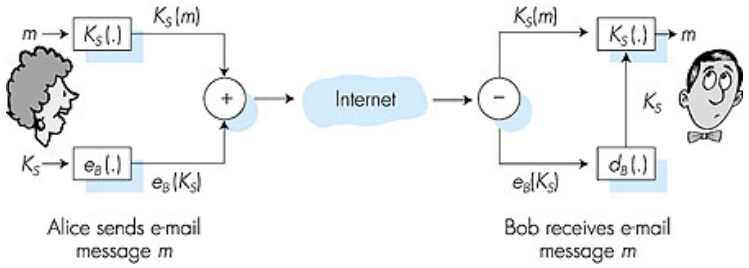
Certificates: what to keep in mind

- Who is the signer?
- What is the validity period? Expired?
- Which algorithm is used? RSA, SHA1, MD5?
- What is its intended purpose? Encryption? Authentication? Issuing other certificates?
- Is it on the certificate revocation list (CRL)?

94

Secure e-mail

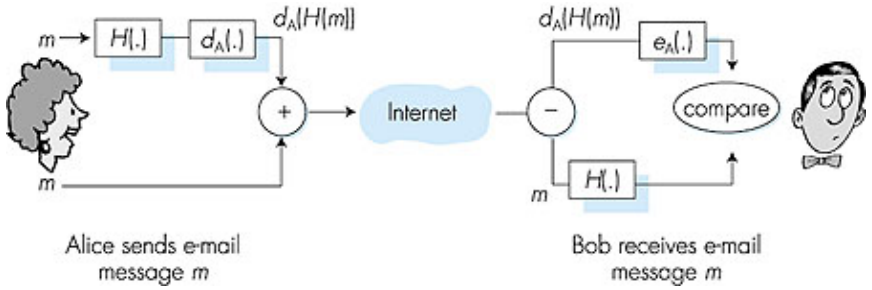
Alice wants to send secret e-mail message, m, to Bob.



- Generates random symmetric private key, K_S.
- Encrypts message with K_S
- Also encrypts K_S with Bob's public key.
- Sends both $K_S(m)$ and $e_B(K_S)$ to Bob.

Secure e-mail (2.)

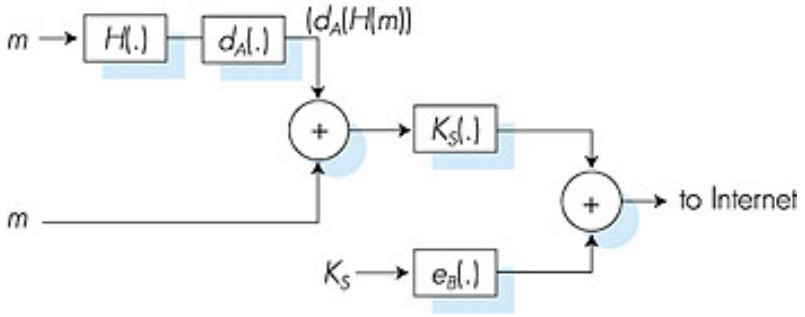
Alice wants to provide sender authentication message integrity



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

Secure e-mail (3.)

Alice wants to provide
secrecy, sender authentication, message integrity.



Note: Alice uses both her private key, Bob's public key.

97

Secure e-mail: PGP

- PGP = Pretty Good Privacy
- It is availiable free on a variety of platforms.
  - Inventor, Phil Zimmermann,
    was target of 3-year federal investigation.
- Based on well known algorithms.
- Not developed or controlled by governmental or standards organizations

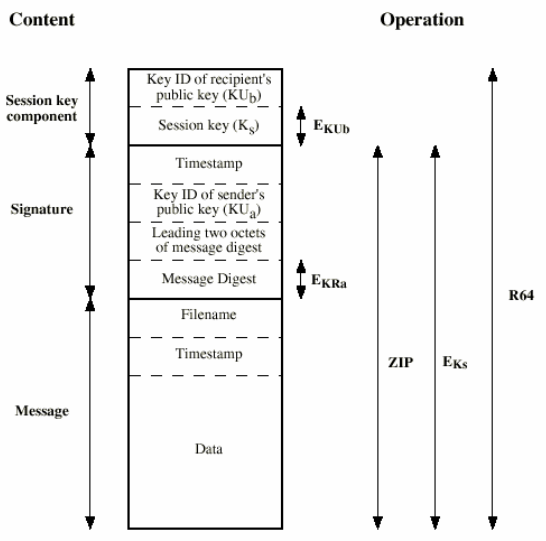
98

PGP offers five services

- Authentication
  - Hash: SHA-1, 160 bits
  - Public key cryptography: RSA
- Confidentiality
  - Session key: 128 bits
  - Symmetric encryption: CAST-128 or IDEA or 3DES
  - Public key encryption: RSA
- Compression
- E-mail compatibility
- Segmentation

99

Format of PGP Message



- Users may have multiple key pairs
  - Key ID's: last 64 bits of public key
- Message component
- Signature component
  - Timestamp
  - Key ID of K_A^+
  - Mess digest: $K_A^-(H(m))$
  - Leading two octets in clear, to verify correct key is being used
- Session key component
  - Key ID of K_B^+
  - Session key: K_S

100

PGP key rings

- Each node has two key rings:
  - Public/private key pairs owned by that node
  - Public key of other users
- For the keys of other users, for each key track:
  - Timestamp: date when key was generated
  - Key ID
  - Public key
  - User id: e-mail address
  - Key legitimacy
  - Signatures

101

PGP Trust

- No certificate authority

How does Alice obtain Bob's key?

- Alice physically gets key from Bob
- Or from phone conversation
- Or gets Bob's key from Claire, who Alice may or may not trust

For a key in your key ring:

- Can you trust that key really belongs to the person defined by the user-id?
- Can you trust that person to vouch for other keys?

For each key on ring:

- Key legitimacy field indicates how much you trust this key to be valid for the associated user.
  - **Determined by PGP algorithm**
- Signatures for PK. Each signature signed with private key of some user
- Also, key ring includes trust values for owners of keys in key ring
  - **Determined by you.**

102

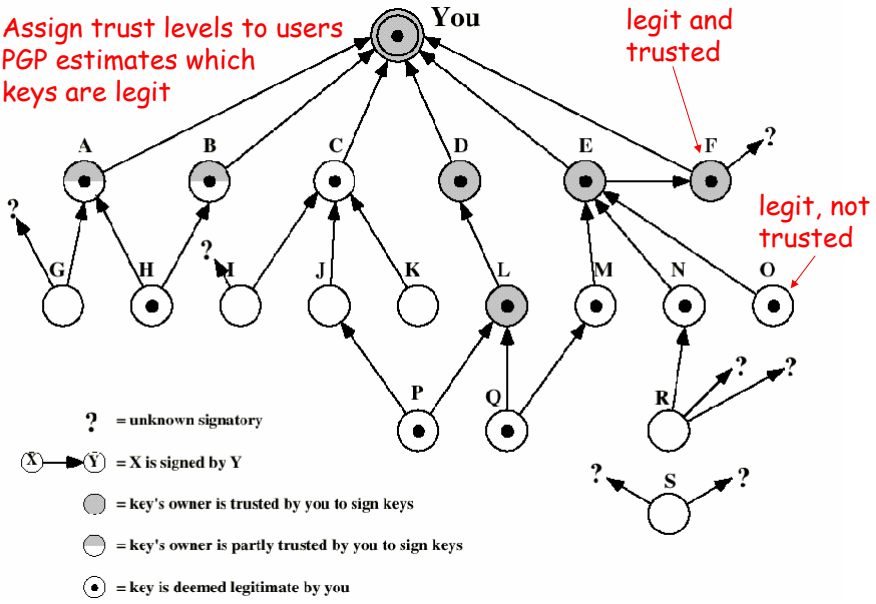
Public key management: example

- Suppose Alice inserts new public key on key ring. If Alice is owner, trust assigned to Alice is *ultimate*.
- Otherwise, Alice must assign trust value to owner of key:
  - unknown
  - untrusted
  - marginally trusted
  - completely trusted.
- New public key may come with signatures. For each signature, PGP searches ring if author of signature is owner of a key in key ring.
- Key legitimacy = legit if one signature completely trusted. Otherwise, determined from formula based on trust of signatures: above threshold, key is considered legit

103

Example

1) Assign trust levels to users
2) PGP estimates which keys are legit



PGP summary

- PGP provides security at application layer to a single application
- Provides:
  - Authentication, integrity, confidentiality
- Public key verification
  - Web of trust

105

S/MIME (RFC3850/RFC3851)

- Standard format for encrypting emails in MIME format
- Many minor syntactic difference to PGP
- Major difference how certificates are signed
  - Uses X.509 certificates / traditional PKI
  - Split of audience
    - Computer scientists like PGP
    - Mainstream users S/MIME
- S/MIME better for official use – makes it clearer when someone is speaking in an organizational role (uses that certificate!)

106

Course overview

- Introduction
  - Attacks and threads, cryptography overview
  - Authentication (Kerberos, SSL)
- Applications
  - Web, email, ssh
- Lower layer network security
  - IPsec, firewalls, wireless
- Information
  - Intrusion detection, network scans
- Availability
  - Worms, denial of service, network infrastructure

107