

Computer Security

Cryptography

Cryptography

- ability to send secret messages
- integrity checking
- authentication

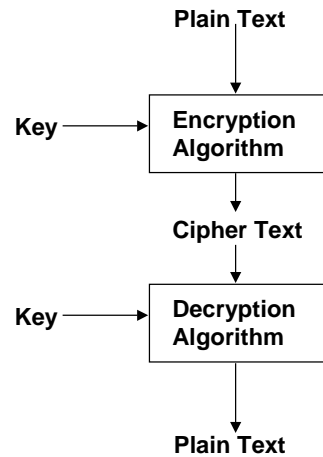
Uses

- cryptography can be used to support
 - confidentiality
 - integrity
 - authentication

Cryptographic Terms

- **plaintext** - the "normal" or readable version of a message
- **ciphertext** - the encrypted version of a message
- **encryption** - the transformation of plaintext to ciphertext
- **decryption** - the transformation of ciphertext to plaintext
- **key** - the secret information used in encryption and decryption

Basic Cryptography



5

Cryptography

- cryptographers - invent clever secret codes (algorithms)
- cryptanalysts - attempt to break these codes
- if lots of smart people have attempted to solve a problem and failed, then it probably won't be solved (soon)

6

Symmetric (Secret) Key Cryptography

- transmitting over an insecure channel
- this is the classic use of cryptography
- same key used for both encryption and decryption
- so both parties must share secret
- algorithms quite efficient
- key distribution problem

7

Asymmetric (Public) Key Cryptography

- different keys used for encryption and decryption
- so users do not have to share secret
- algorithms markedly less efficient than for symmetric key cryptography

8

Public and Private Keys

- in asymmetric key systems each user normally has two keys
 - private key (known by user and trusted authorities)
 - public key (known by everyone)
- can send message to someone by encrypting using their public key

9

Breaking Encryption Systems

- cryptographic algorithms can be broken without the key(s)
- it is simply a question of how long it takes
- a scheme can often be made more secure by making the keys longer

10

Are the Algorithms Publicly Known?

- if no, can say more secure as less information for attacker
- but difficult to keep an algorithm secret
- so perhaps security should not depend on algorithm being secret

11

Breaking an Encryption Scheme

- ciphertext only attack
- known plaintext attack
- chosen plaintext attack
- a cryptosystem should be proof against all three

12

Ciphertext Only

- attacker has a ciphertext
- ciphertext easy to obtain, by eavesdropping
- attacker needs enough of the ciphertext

13

Attacking

- can try each possible key
- sometimes do not need to try each key
- eg., in Kerberos keys based on passwords, so if password poorly chosen...

14

Known Plaintext

- attacker has a plaintext,ciphertext pair – is usually trying to determine key
- for some ciphers this is fatal
- for such ciphers need to ensure plaintext,ciphertext pair never available

15

Chosen Plaintext

- attacker may be able to get system to encrypt chosen plaintext
- for example, public encryption service

16

Other Uses of Cryptography – Storage

- secure storage on insecure media
- similar to standard cryptography
- except same person does encryption and decryption

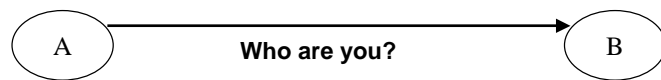
17

Other Uses of Cryptography – Authentication

- authentication – proving your identity
- strong authentication - proving knowledge of a secret without revealing it
- possible with computers

18

Authentication



How can I
be sure of
the answer?

19

Authentication

- the process of determining the correctness of a user's identity
- this is usually based on one or more of the following:
 - a user possession (eg., a card or certificate)
 - user knowledge (eg., a password or key)
 - user attribute (eg., fingerprint, retina pattern)

20

Strong Authentication

- suppose Alice and Bob both know K_{ab}
- each picks a random number (r_a and r_b) known as a *challenge*
- the value x encrypted with K_{ab} is known as the *response* to challenge x

21

Protocol

- Alice \rightarrow Bob : r_a
- Bob \rightarrow Alice : $[r_a]K_{ab}$
- Bob \rightarrow Alice : r_b
- Alice \rightarrow Bob : $[r_b]K_{ab}$
- actually this protocol have some problems, but we'll come to that later

22

Attack

- someone impersonating Alice could get Bob to respond to a challenge but could not respond in turn
- note this does give an attacker (plaintext,ciphertext) pairs
- challenges must be chosen from a large enough space (say 2^{64})

23

Kerberos

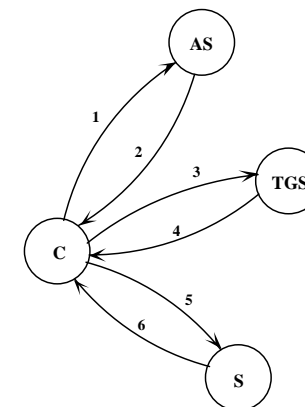


Figure 1: Kerberos Protocol

24

Kerberos Protocol

(1) C → AS : C, t_C, TGS

(2) AS → C : [K_{C,TGS}, <T_{C,TGS}>_{K_{TGS}}]_{K_C}

Where <T_{C,TGS}>_{K_{TGS}} = { C, TGS, t_{AS}, lifetime_{AS}, K_{C,TGS}, }_{K_{TGS}}

(3) C → TGS : S, [C, t_{C1}]_{K_{C,TGS}}, <T_{C,TGS}>_{K_{TGS}}

(4) TGS → C : [K_{C,S}, t_{C1}, <T_{C,S}>_{K_S}]_{K_{C,TGS}}

Where <T_{C,S}>_{K_S} = { C, S, t_{TGS}, lifetime_{TGS}, K_{C,S}, }_{K_S}

(5) C → TGS : [C, t_{C1}]_{K_{C,S}}, <T_{C,S}>_{K_{TGS}}

(6) S → C : [t_{C1}+1]_{K_{C,S}}

25

Authentication – Public Key Systems

- much easier with public key cryptography
 - owner has private key
 - everybody has public key
- to prove identity, simply encrypt something using private key
others can decrypt using public key
- at least, that's the basis

26

Other Uses of Cryptography - Integrity Check

- keeping things secret is not all we would like to do
- we would like some way of checking whether the message sent is the message received
- messages can be changed in transit
 - deliberately (by attackers)
 - accidentally (by transmission error)

27

MIC

- one way of checking whether the message sent is the message received is to calculate a quantity based on the message
- send it with the message and redo the calculation on the message received
- if the quantity sent and the quantity calculated by the receiver match we can assume the message sent is the one received
- such a quantity is called a *message integrity code* (MIC)

28

Secrecy & MIC

- The MIC must be calculated in such a way that an attacker *can not*
 - change a message *and*
 - calculate a MIC for the new message
- Attackers can always change messages – it's the requirements about not being able to calculate a MIC that's important

29

Secret Key and MIC

- secret key mechanisms can be used to generate a MIC
- in this case also called a cryptographic checksum
- given a key and a message the checksum algorithm produces a *fixed length* MIC

30

Use of MIC

- when message is received by someone who knows key
 - re-compute MIC from message received
 - check MIC computed matches that received

31

Security

- a secret checksum algorithm must ensure that an attacker not knowing the key can't alter a message and compute a correct new checksum for the corrupted message
- a typical MIC must be at least 48 bits long for computational security – as computer power increases this length will need to increase

32

Public Key Cryptography – Digital Signatures

- a digital signature is like a MIC
- digital signature can only be generated by someone who knows the private key

33

Integrity

- what if you encrypt something using your private key?
- everyone can decrypt as they know your public key
- so encrypt this cipher text and the plain text using the recipients key
- they can do both decryptions and, if the results match, then the message can not have been tampered with

34

Digital Signatures

- it must have been you that encrypted the message in the first place
- if you only send off the version encrypted with your private key then everyone can see that you sent it
- this is the basis for digital signatures

35

MIC and Digital Signature

- verification of a digital signature does not require knowledge of the secret used to create it - verifying a MIC does
- anyone who knows the key used to create the MIC can alter the message and create a new MIC

36

Non-repudiation

- digital signatures support non-repudiation
- if signed with private key, must have originated with that person

37

Hash Algorithms

- message digests
- one-way transformations
- takes message of arbitrary length and computes fixed length (short) number

38

Desirable Hash Properties

- $m, h(m)$
- given m is easy to compute $h(m)$
- given $h(m)$ no way to find m that hashes to $h(m)$ that is substantially easier than brute force (one way functions)
- it is computationally infeasible to find two m 's that hash to the same $h(m)$

39

Example

- treat m as a number
- add a large constant
- square it
- take the middle n digits as a the hash
- while this is not a particularly good function it is an example

40

Password Hashing

- instead of storing passwords, system stores hashes of passwords
- if attackers obtains password file does not get actual passwords

41

Message Integrity

- can't simply use hash to generate MIC as attacker knows algorithm - can alter message and generate new hash
- if parties to communication choose a password can use concatenation of message and password

42

MIC

- Can use hash to produce fixed length value dependent on message
- Can then encrypt that to produce MIC

43

Message Fingerprint

- can check whether a file has been modified by taking hash and comparing to old value
- can also be used to check security of storage

44

Efficiency

- remember that public key cryptography is not that efficient
- better to compute message digest and sign that rather than sign message directly