

# Computer Security

## Authentication

# Authentication

- process of reliably verifying identity
- verification techniques
  - what you know (eg., passwords, crypto key)
  - what you have (eg., keycards, embedded crypto)
  - what you are (eg., biometric information)

# Authentication of People

- how authentication is done depends on capabilities of entity being authenticated
- two most important capabilities
  - ability to store a high-quality key
  - ability to perform cryptographic operations

# High Quality Key

- secret
- chosen from large space
- computationally infeasible to guess

## Computer vs. Person

- computer has both
- person has neither

## Passwords

- an old idea
- military uses
- everybody knows password of day

## Password Based Authentication

- in password based authentication we have a secret quantity
- you have to state it to prove you know it
- has a number of problems

## Problems with Passwords

- eavesdropping
- on-line guessing of password
- off-line cracking
- unguessable passwords unusable
- security of password file

## Eavesdropping

- passwords must be uttered to be used
- can then be overheard
- most people don't watch
- but they are not the people you are worried about
- wiretapping is a more sophisticated problem

## Eavesdropping

- if the password is sent from across a network then eavesdropping is possible
- for example, a traditional telnet connection is unsecured – no cryptography
- so an attacker who can eavesdrop, eg., on the port in use, simply gets to see the password

## Trojan Horses

- A trojan horse is a useful, or apparently useful, program, which also performs unwanted/harmful functions
- If a user can be induced to run a trojan horse which mimics the log in program then the trojan can capture the user's password
- The password can then be sent to the author of the trojan

## On-Line Guessing

- I can impersonate you if I can guess your password
- some systems enforce easily guessable passwords (not really a good idea, but some do it – would be better to disallow)
- some people use easily guessable passwords

## On-Line Guessing

- with enough guesses even obscure passwords can be guessed
- not a problem for the military
- get it wrong, get shot

13

## On-Line Guessing

- executing users who get their password wrong would probably be unacceptable
- can make sure that guesses have to be typed

14

## Locking Accounts

- can lock accounts after too many failed attempts
- but then easy for someone to deny access
- can cut-off connection after a number of failed attempts and require it to be reestablished
- can have system response be very slow

15

## Catching Attacker

- can attempt to catch guesser
- trace modem connection or otherwise locate
- especially when attempts are made to different accounts

16

## Off Line Password Guessing

- passwords more vulnerable if off-line guessing possible
- off line attack - an intruder captures a quantity that is derived from password
- attacker then takes their time trying to compute password
- if off-line guessing is possible then password must be from a much larger space than if it is not

17

## How Big?

- to thwart on line attacks only, secret does not have to be from large space
- for off-line attacks, secret needs 80+, maybe 128+, bits of randomness

18

## Problem

- humans usually are not willing to remember 64 bits
- a randomly chosen 11 character string will suffice for 64 bits
- but humans won't remember those either

19

## Memorable Passwords

- if the password was pronounceable, rather than fully random
- can get enough randomness in 16 characters
- humans won't remember those either

20

## User Chosen

- humans choose poor passwords
- to be proof against off-line attacks passwords need to be ~32 characters long
- humans certainly won't remember those

21

## Conclusion

- passwords will be vulnerable to off-line guessing

22

## Passwords

- system can assign random ones - users resent
- systems can check for easily guessed ones

23

## Alternatives

- one-time passwords
- but then occasionally need new list
- a set of passwords
- use only some at each login, on a challenge-response basis
- eavesdropper has to listen many times

24

## Password Files

- if password file disclosed all passwords vulnerable
- attacker usually wants any password, not a particular one
- can use dictionary to attempt to crack some passwords
- password files must be at least as well protected as anything else

25

## Protecting Passwords

- some system do not store the passwords themselves
- instead hold quantity derived from password, such as cryptographic hash

26

## Salt

- system chooses random number for each user - the salt
- stores salt and hash of combination of salt and password
- salt does not make it harder to guess password
- does it make it harder to use single guess on all passwords

27

## Salt

User ID	salt value	password hash
Mary	2758	hash(2758   passwd)
John	886d	hash(886d   passwd)
Ed	5182	hash(5182   passwd)
Jane	1763	hash(1763   passwd)

28

## Other Problems with Passwords

- users will give passwords to others
- will write them down
- put them in programs
- will forget them

## Multiple Uses

- users may have multiple accounts
- may use same passwords for all
- easier to remember
- break one, break into all

## Password Changes

- system can require regular password changes
- if attacker knows password, only until next change

## Problems

- too frequent changes lead to poorly chosen passwords
- users will write them down
- users will oscillate between two passwords

## Password Distribution

- how are passwords given to users in the first place?
- if you can impersonate user to system administrator may get their password

## In Person

- turn up to administrator's terminal
- authenticate using documents
- type in password
- inconvenient
- gives access to sensitive terminal

## Administrator Chosen

- administrator chooses good password
- gives it to user
- instructs them to change it immediately

## Pre-Expired

- as before, but password already expired
- must be changed immediately
- can be written down for distribution

## Variations

- distribute passwords by mail
- original password function of user attribute (eg., ID#)

37

## Passwords in Distributed Systems

- how do you access resources on different workstations?
- does your workstation remember your password and transmit it?
- but then you need same password on every workstation
- how do keep them co-ordinated?

38

## Storing User Passwords

- how does a server know Alice's password?
  - Alice's authentication information is individually configured into every server
  - one location stores Alice's information and servers retrieve it when needed

39

## Storing User Passwords (cont.)

- one location stores Alice's information and servers which want to authenticate Alice send Alice's information to the location which replies yes or no
- for the last two the server must be certain of the identity of the node which stores the authentication information

40

## Authentication Database

- such a database must be secure
- can encrypt passwords
- or hashes of the passwords
- if the key for that node is broken, the whole database will be vulnerable

41

## Password Based vs. Cryptographic Authentication

- some systems might be thought password based but are not
- if secret string is converted into a cryptographic key, it is not really password based

42

## Why Use Passwords?

- given that cryptographic authentication is more secure, why use password based authentication at all?
- having dumb terminals
- using a protocol designed without cryptography
- perhaps protocol would be too expensive if cryptography used

43

## Example

- early cellular phones transmitted the telephone number of the phone and a password when making a call
- if the password corresponds to the telephone number, call is allowed
- very easy to eavesdrop and clone phone
- a challenge response protocol would have been better

44

## Address-Based Authentication

- does not rely on sending passwords
- assumes identity of source can be determined from network address at which packets originate
- each node notes which accounts on other nodes should have access to its resources

45

## Implementation

- node B holds a list of network addresses of equivalent machine
- if node A is listed, then any account on A is equivalent to the same account name on B
- or

46

## Implementation (cont.)

- node B might instead have a list address, remote account name, local account name
- request from remote account on remote machine is equivalent to same request coming from local account

47

## Security Considerations

- if someone subverts a node, can access resources on other machines open to accounts on the subverted node
- of course, attacker needs to know what possibilities exist

48

## Security Considerations (cont.)

- however, access is usually complementary, so only needs to examine database on subverted node
- if attacker may be able to forge messages so that they appear to be from other nodes
- attacker can then access resources reserved for users from those nodes

49

## Authentication Tokens

- what you have
- subject to theft
- generally used either with passwords or biometric checks

50

## Examples

- keys (car, house)
- credit cards

51

## Magnetic Strips

- hold information
- not trivial to counterfeit
- people less likely to loan than share password

52

## Disadvantages

- requires custom hardware
- can be lost, stolen, damaged or destroyed
- require override when card misplaced
- as vulnerable as password to eavesdropping

## Smart Card

- same size as credit card
- embedded CPU and memory
- insert in smart card reader
- reader and card carry on conversation
- with magnetic strip simply dump information

## PIN Protected Memory Card

- information in memory of card
- only read after PIN input
- card locks after some number of wrong guesses
- more secure than magnetic strip

## Cryptographic Challenge/Response Card

- cryptographic key held in card memory
- card will encrypt or decrypt using key
- will not reveal key, even after PIN entered

## Challenge/Response

- computer that knows key can authenticate
- picks random
- challenges card to encrypt or decrypt
- cards unreadable for most practical purposes
- offer protection against eavesdropping

57

## Problems

- need special devices
- cards can be lost or stolen

58

## Cryptographic Calculator

- also called readerless smart card
- performs cryptographic calculations
- uses a key it will not disclose
- needs no reader
- has a display and keyboard
- all interaction through user
- popular for remote access to networks

59

## Physical Access

- provide authentication via human guards
- system authentication distinguishes legitimate users
- eg., bank tellers
- location can be part of authentication process
- rights granted depend on location

60

## Biometrics

- what you are
- measure physical characteristics
- hard to loan or steal

## Examples

- retinal scanner
- fingerprint scanner
- handprint reader
- voiceprints
- keystroke timing
- signatures

## Problems

- hard to keep biometric quantities secret
- reader needs tamper-resistant secret
- communicates with remote computer via cryptographically protected exchange

## Cryptographic Authentication Protocols

- can be much more secure than password or address based authentication
- Alice proves her identity to Bob by performing a cryptographic operation on a quantity Bob supplies
- the cryptographic operation is based on Alice's secret

## Who is Being Authenticated?

- a user, eg., when wanting to access the files at a server
- a node, eg., duplicate file servers authenticating each other
- a user and a node, eg., a bank teller at their assigned terminal

65

## The Difference

- a computer can store a high quality secret and do cryptographic operations
- a computer can do these on behalf of a user
- but system has to be designed so that all user has to remember is password

66

## Password to Key

- do a hash of password
- use password to decrypt higher quality key, such as RSA private key

67

## Passwords as Cryptographic Keys

- public key cryptographic keys are specially chosen very large numbers
- most users can not remember something like this
- can remember a password
- can convert password into DES key
- much harder to convert it into RSA key

68

## Possible Solution

- use password as seed for pseudo-random number generator
- remember RSA keys need to generate lots of numbers to test for primality
- this method can take a long time
- but can also get user to remember how many numbers generated to get each key in pair

69

## In Practice

- not done all that often because of time taken and knowledge of public key allows off-line password guessing
- usual practice is to encrypt user's private key with user's password and store it somewhere it can be recovered by workstation and decrypted using password

70

## Authentication Protocol

- Alice -> Bob : I'm Alice
- Bob -> Alice : random R
- Alice -> Bob : R signed with Alice's private key
- Bob checks using Alice's public key

71

## Attacks

- public key technology makes it easy to do authentication
- secure from both eavesdropping and server database reading
- eavesdropping useless
- accessing Bob's database only gives Alice's public key, which everybody knows anyway

72

## Without Public Key - Eavesdropping

- If protocol is  
Alice-> Bob : Alice's password
- Bob is machine authenticating Alice
- eavesdropper can get password
- Bob's database is hashed version of password - useless to intruder as they can not get password

73

## Without Public Key - Server Compromise

- If protocol is  
Alice-> Bob : I'm Alice  
Bob -> Alice : Random R  
Alice -> Bob : X
- X is cryptographic function of Alice's secret and R, for example  $[R]_{K_{AB}}$

74

## Attacks

- eavesdropper learns nothing
- but Bob must store Alice's secret to check X
- so intruder gets Alice's secret

75

## Getting a Key

- if A wants to communicate with B, needs B's public key
- when B is added to system could tell everyone B's key
- not very practical in large systems

76

## Trusted Intermediaries

- can have one system which stores everyone's public key
- when want to talk to another principal, go to the system with all the keys
- need to be aware of root of trust

77

## KDC

- one version of this idea is a Key Distribution Centre
- KDC knows keys for all nodes, users etc (ie all principals)
- when new principal installed, only it and KDC need to have key

78

## Conversations using KDC

- if node a wants to talk to node b, a talks to KDC
- this is secure as a and KDC share a secret key
- a asks KDC for key to talk with b
- KDC authenticates a

79

## Conversations using KDC

- chooses random  $R_{ab}$  for use as key between a and b
- encrypts  $R_{ab}$  with key shared between a and KDC
- encrypts  $R_{ab}$  with key shared between b and KDC
- sends both to a
- a sends encrypted  $R_{ab}$  to b

80

## Tickets

- usually with information such as expiry time and a's name
- this is usually called a **ticket**

81

## Disadvantages of KDC

- KDC can impersonate anyone
- KDC is a single point of failure
- KDC can be a performance bottleneck

82

## Possible Problem

- key distribution easier with public keys
- but how do you know the listed public key is the correct one for the listed entity?
- what if some attacker has substituted their public key?

83

## Certification Authorities

- generate certificates
- signed messages which specify a name (eg., Alice) and the corresponding public key
- all nodes need to know CA's public key
- can then verify signatures

84

## Certificates

- certificates can be stored anywhere
- a principal can provide its certificate as part of authentication
- if the CA is compromised it can destroy the integrity of the entire network

85

## Certificates

- have
  - user's name
  - user's public key
  - expiration time
  - serial number
  - issuing CA's signature

86

## Advantages of CAs over KDCs

- CA does not need to be on-line
- as not on line can be simpler and more secure
- if CA crashes only stops new users being added
- certificates themselves not security sensitive

87

## Advantages of CAs over KDCs

- a compromised CA cannot decrypt conversations
- a compromised CA can still impersonate a network entity

88

## Certificate Revocation

- what if Fred is a given a certificate with a life span is a year and is soon after fired?
- with KDC's, simply delete him from the KDC
- must somehow cancel certificate
- can publish lists of revoked certificates (a la credit cards)

89

## Certificate Revocation List

- A CRL lists serial numbers of certificates that should not be honoured
- a certificate is valid if
  - it has a valid CA signature
  - has not expired
  - is not listed in the most recent CRL
- CRL has an issue time
- an intruder could destroy CRL

90

## Lifespans & Revocation

- certificate lifetimes can not be too short or certificates have to be issued too often
- can not be too long, or will have to revoke too many certificates

91

## Lifespans

- note the tradeoff between certificate lifespan and CRL length
- could include extra field in CRL which gives first valid certificate all earlier certificates invalid
- requires serial numbers to be issued in order
- then do not need lifetimes for certificates

92

## Multiple Trusted Intermediaries

- problem with a KDC or CA is that there must be only one trusted authority
- as the system is scaled we reach the point where no one entity is trusted by everyone
- solution is to split world into **domains**

93

## Domains

- each domain has one trusted authority
- if Alice and Bob are in same domain, they authenticate as already described
- if they are not, they still can authenticate, but it is more complicated

94

## Communication across Domains

- say Alice is domain a
- Bob is in domain b
- the KDC's of a and b share a key,  $K_{ab}$
- $K_{ab}$  is used when a user in a wants to have a secure communication with a user in b

95

## Communication across Domains

- Alice tells her KDC she wants to talk to b's KDC
- a's KDC treats this much the same as a communication between two users in its domain
- a's KDC generates a key,  $K_{new}$
- encrypts this using Alice's key and  $K_{ab}$
- message also contains Alice's name

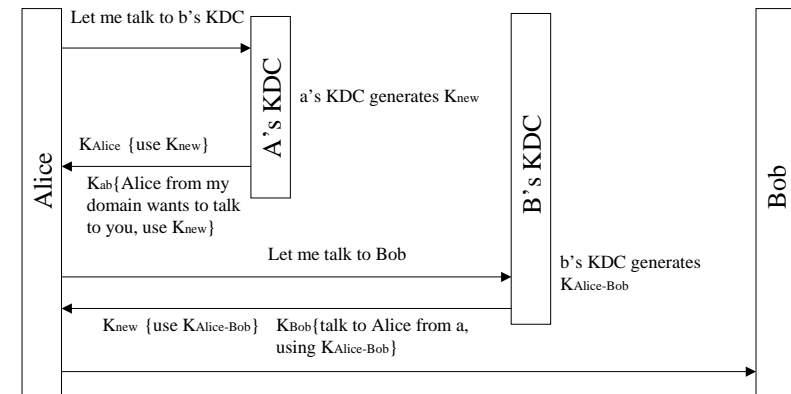
96

## Communication across Domains

- Alice sends all this to the KDC of b, along with Bob's name and a's name
- KDC of b knows which key to use (from a's name)
- KDC of b generates key and message for Alice to communicate with Bob and sends it to Alice as if Alice was part of b's domain
- Alice can then communicate with Bob

97

## Communication across Domains



98

## Many Domains

- there are many domains on the internet
- too many for all the KDC's to have keys for each other

99

## Chain of KDC's

- users can securely talk to each other even if their KDC's are not directly linked
- ie, do not have each other's keys
- need a chain of KDC's to be found
- when the chain is found Bob and/or the KDC in his domain should be informed of it when they are contacted

100

## How to find Chain?

- perhaps a hierarchy
- some pairs of KDC's are linked across branches

101

## Problems After Authentication

- if only do authentication
  - eavesdropper might overhear authentication
  - the messages might be intercepted in transit and modified or replayed
  - someone on path might hijack entire session by impersonating network address of one party
- cryptography can protect against these

102

## Which Key?

- could use public keys
- computationally expensive
- parties need secret key
- nice if authentication involved agreement on secret key

103

## Long Term Key

- could have a long term key
- keys wear out - the longer use, the more chance of being cracked
- old conversations could then be decrypted
- re-used keys allow more chances for replay
- can not give less trusted software the key

104

# Session Key

- the problems with long term keys solved by a session key
- used for one conversation only
- usually established during authentication