

Computer Security

Security Protocols

Security Protocols

- Most programmers should never have to design cryptographic algorithms
- may have to design/implement applications which communicate using cryptography
- communication between applications uses a **protocol**

Security Protocols

- a security system consists of a number of principals
 - People, companies, computers, card readers, biometric devices, etc, etc, etc
- these principals communicate
- the security protocols are the rules that govern these communications

Problems

- many protocols have security flaws
- there is no one best protocol
- different protocols have different tradeoffs
- using protocols in ways not intended may result in vulnerabilities

Security Protocols

- security communication usually includes initial authentication
- some information is secret
- some information is public

5

Programmers

- need to understand potential flaws
- even slight changes to existing protocols can result in problems
- even with no changes weakness in original may be more important in new setting

6

Design

- start with some design
- check for problems
- fix them
- check again
- etc

7

Protocols

- Should have a standard internal format
- Should be as simple as possible
- Should have as few variants as possible

8

Examining Protocols

- Need to understand its assumptions
 - Should make as few as possible
- Need to examine each step
 - What does recipient know?
 - What is the next possible step(s)?

9

Representing Protocols

- No standard format exists
- I'll use
 - $\{M\}_K$ – message M is encrypted under **symmetric key** K
 - $[M]_K$ – message M is encrypted under **asymmetric key** K

10

Password Protocols - Outline

- initiator send name and password in clear
- verifier checks name and password
- communication occurs with no further security

11

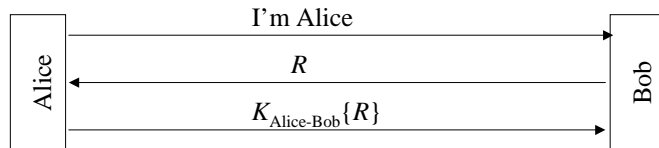
Password Protocols - Assumptions

- eavesdropping not a problem
- attackers unsophisticated
- not very realistic assumptions, really
- so let's try something else

12

Shared Secret - Protocol 1

A simple challenge-response protocol



Bob authenticates Alice based on a shared secret key $K_{\text{Alice-Bob}}$

13

Shared Secret - Protocol 1

- R is cryptographically transformed using Bob and Alice's shared secret
- secret may be key for DES or IDEA
- may use hashing - concatenate R and secret
- eavesdropper will see R and transformed R
- this should not enable attacker to derive secret

14

So What's Known?

- After message 1 Bob knows someone **claiming** to be Alice wishes to communicate
- After message 2 Alice knows some information which she **assumes** is a random number

15

So What's Known?

- After message 3 Bob knows the other party knows the shared key
 - **Assuming** the random was well chosen
 - Note Bob only can say it is Alice if he **assumes** only Alice also knows the key
- If random not well chosen attacker can build up a table of challenges and responses

16

Advantages of Protocol

- big improvement over passwords in clear
- eavesdropper can not impersonate Alice (if our assumptions are valid)

17

Disadvantages

- authentication is not mutual
- if no subsequent encryption, attacker can hijack
- off-line key guessing possible
- compromise of Bob's database means Alice can be impersonated

18

Bottom Line

- useful if only limited resources for adding security
- replacing cleartext password transmission very important enhancement

19

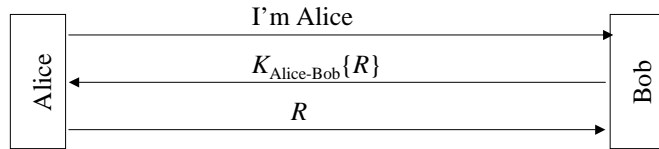
Assumptions

- If a protocol's assumptions are invalidated it can be useless (or worse than useless)
- For example, what happens if our assumptions about the secrecy of the key or how well chosen the random is are incorrect?
- The protocol will fail, no matter how good the cryptography

20

Shared Secret - Protocol 2

Another challenge-response protocol



Bob authenticates Alice based on a shared secret key $K_{\text{Alice-Bob}}$

21

So What's Known?

- After message 1 Bob knows someone **claiming** to be Alice wishes to communicate
- After message 2 Alice knows some information which she **assumes** is a random number encrypted using the shared key

22

So What's Known?

- After message 3 Bob knows the other party knows the shared key
 - assuming the random was well chosen
 - Note Bob only can say its Alice if he **assumes** only Alice also knows the key

23

Analysis

- requires reversible cryptography
- cannot be done using hash function (first protocol can)
- if R recognisable quantity (eg., 32 bit random then 32 zero bits) attacker can try key guessing

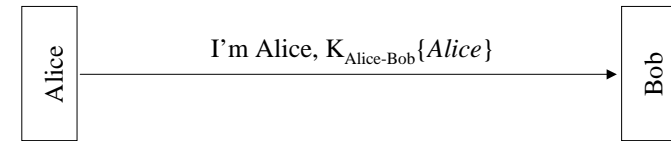
24

Analysis

- note first message can be sent by anyone and then attacker gets second message
- if R is recognisable quantity with limited lifetime Alice authenticates Bob as well

25

Shared Secret - Protocol 3



Bob authenticates Alice based on a shared secret key $K_{\text{Alice-Bob}}$
 This is a very bad protocol, but useful for illustration

26

Attacks

- there are many ways to undermine protocols
- we will consider three common ones
 - replay attacks
 - man in the middle attacks
 - reflection attacks

27

Replay Attack

- attacker records a message and later sends it to victim
- protocol 3 is very susceptible to this attack

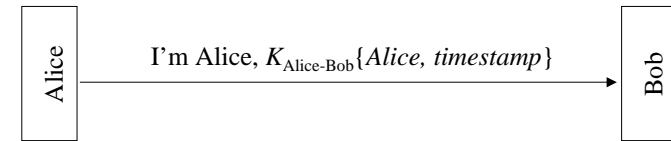
28

Timestamps

- help prevent replay attacks
- require reasonably synchronised clocks
 - but watch for quick replays

29

Shared Secret - Protocol 3a



Bob authenticates Alice based on synchronised clocks and a shared secret key $K_{\text{Alice-Bob}}$

30

Shared Secret - Protocol 3a

- Bob and Alice must have reasonably synchronised clocks
- Bob checks encrypted time is acceptable

31

Analysis

- can easily be added to protocol designed for sending clear text passwords
- efficient - few messages and no volatile state
- eavesdropper can impersonate Alice if quick
 - can be avoided if volatile state (unexpired timestamps) remembered
 - But this can require a lot of memory

32

Analysis

- if Alice contacting multiple servers eavesdropper can impersonate
- can be avoided by concatenating server name with timestamp
- if intruder can get server to reset clock old message can be reused
- if security relies on time then setting time requires security handshake

33

Problem

- clocks can not be precisely synchronised
- how does Bob know what is encrypted?
- not a problem for reversible cryptography, but is for hashes

34

Hashes

- would have to try all reasonable possibilities
- if timestamp in microseconds and acceptable clock skew is five minutes get 600 million possibilities
- need to send timestamp in clear

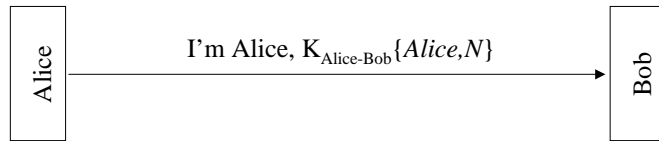
35

Nonces

- Another alternative is to use a nonce

36

Shared Secret - Protocol 3b



Bob authenticates Alice based on a shared secret key $K_{\text{Alice-Bob}}$
 N is a nonce

This is another poor protocol, as subject to replay if nonces not recorded

37

Nonces

- a quantity which any given user of a protocol uses only once
- many protocols use nonces
- different types of nonces have different properties
- A timestamp is a form of nonce (if the clock never goes backwards)

38

Examples

- large random number
- sequence number
- both can (and especially sequence numbers) require non-volatile state
 - And potentially a lot of it

39

Random Number

- cannot be guessed or predicted (others can)
- non-reuse is only probabilistic
- but if large enough (eg., 128 bits) negligible chance

40

Pseudo-Random Numbers

- true random numbers hard to get
- standard method for pseudo-random numbers
 - use as seed a key known only to node generating numbers

41

Sequence Numbers

- genuine messages replayed out of order can disrupt conversation
- can be avoided by putting sequence numbers in each message
- or by computing integrity not just on current message but on previous ones as well

42

Sequence Numbers

- reflection attacks can lead to misinterpretation if same sequence number valid in both directions
- avoided by different sequence numbers in different ranges for two directions, having a direction bit or using subtly different integrity algorithm in each direction

43

Sequence Numbers

- sequence numbers need to be from large space to avoid running out
- can avoid reusing sequence numbers by periodically changing keys in mid-conversation (key rollover)

44

Problems with Nonces

- In protocol 3b how does Bob know that the message is not being replayed?
- If using sequence numbers, is it good enough to remember last sequence number and accept next one?
 - But what if a message gets lost?
- With random numbers some systems can be broken if only look for current nonce to be different from last one
 - Attacker resends second last message

45

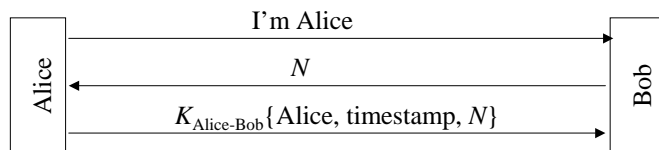
Nonces

- Nonces are better used as part of a challenge
- they can guarantee that a reply is in a response to a particular challenge

46

Shared Secret - Protocol 4

A simple challenge-response protocol



Compare to protocol 1

47

Nonce and Timestamp

- The timestamp is guaranteeing that the message has been recently put together
 - This is called **freshness**
- Without it the attacker may have seen the nonce (challenge) before
- The nonce is guaranteeing that the message is in response to a particular challenge, and not a replay of an earlier one

48

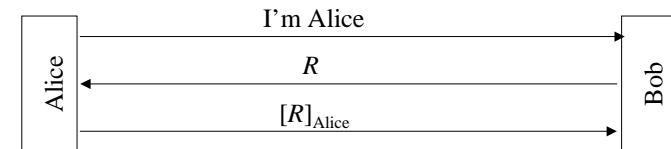
Database Attacks

- previous protocols use shared secret
- if attacker infiltrates Bob's database can impersonate Alice
- this can be avoided by using asymmetric keys

49

Public Key - Protocol 5

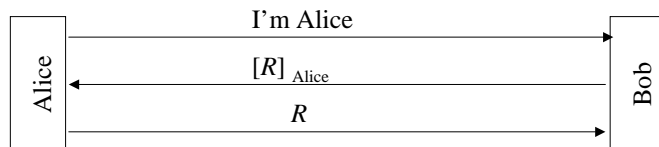
- Bob's database must be protected from unauthorised access – or he should use a signed certificate
- This is also a challenge-response protocol



Bob authenticates Alice based on her public key signature

50

Public Key - Protocol 6



Bob authenticates Alice if she can decrypt a message encrypted with her public key

51

Note

- some public key schemes (such as DSS) not reversible

52

Problem

- with protocol 5 can trick someone into signing something
- with protocol 6 can trick someone into decrypting something
- only need to be able to forge Bob's network address or otherwise persuade Alice you **might** be Bob

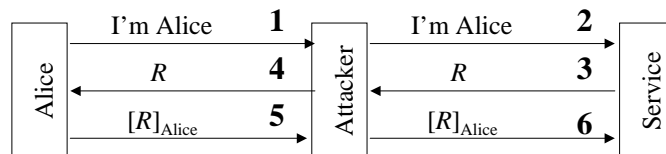
53

Man in the Middle

- assume protocol 5
- assume Alice registered with some legitimate server
- assume attacker can get Alice to register with some service provided by the attacker

54

Man in the Middle – Ex1



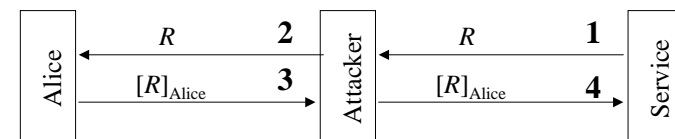
Note the ordering of the messages

Note also that the attacker has to wait until Alice does something before attack can begin

55

Man in the Middle – Ex2

This one is a little rarer, but still interesting
It depends upon being able to Challenge Alice



Note the ordering of the messages

56

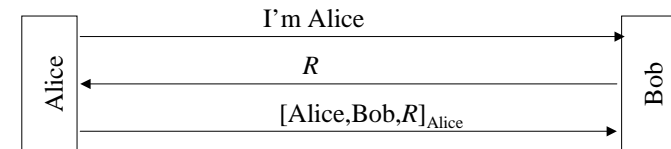
The Problem

- time stamps and nonces will not help against this form of attack
- we need to be able to identify what the creator of the message intended it to do
- and pay attention when we are creating messages
- Note that this attack works against symmetric key based protocols too

57

Public Key - Protocol 5a

- so an improvement to protocol 5 would be



58

Structure

- the last message in protocol 5a should be read with the understanding
 - senders name, then recipients name
- it does not matter which way round we have this, as long as we are consistent for all messages in the protocol
- this can help stop reflection attacks, as we will see later

59

Now

- in example 1 Alice will put the attacker's service name in message 5, not the other service's name, so the other service will reject
- in example 2 Alice will put the attacker's name, not the service's name, so again the service will reject

60

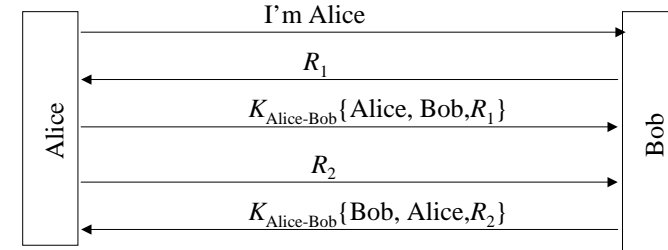
But

- if the attacker can trick Alice as to the attacker's identity, may still have problems
- but the protocols we've looked at up until now have only been for one way authentication

61

Mutual Authentication Protocol 1 (MA1)

Two challenge-response exchanges



Mutual authentication scheme based on a shared secret key $K_{\text{Alice-Bob}}$

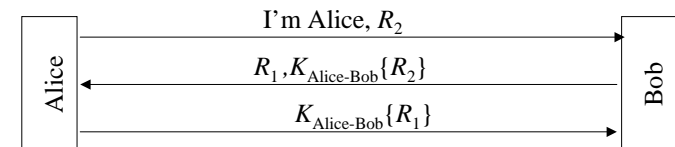
62

Mutual Authentication

- simple
- inefficient

63

Mutual Authentication Protocol 2 (MA2)



Optimised mutual authentication scheme based on a shared secret key $K_{\text{Alice-Bob}}$

64

Mutual Authentication

- three messages instead of five
- vulnerable to the reflection attack

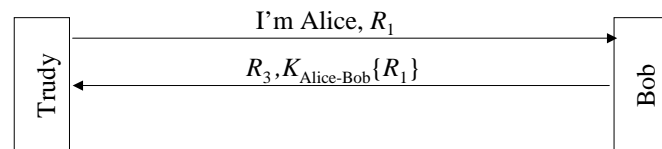
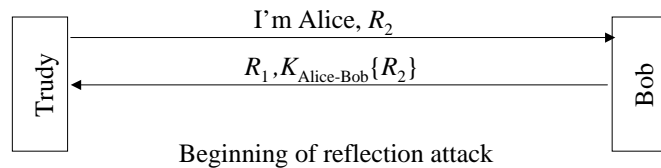
65

Reflection Attack

- Trudy wants to impersonate Alice
- starts protocol MA2
- when she receives challenge from Bob she cannot encrypt R_1
- Bob has encrypted R_2
- Trudy opens another session, and can then complete first

66

Reflection Attack



Second Session in reflection attack, can now complete first

67

Analysis

- this is a serious flaw
- A client can often open multiple sessions with a server
- There may be multiple servers with same secret for Alice
- so may not even need multiple sessions with one server

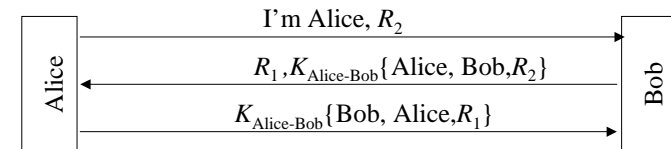
68

Solution

- do not have Alice and Bob do exactly the same thing
- use different keys to authenticate Alice and Bob
- use different forms of challenges
- eg., simply have name of party concatenated with R's

69

Mutual Authentication Protocol 2a (MA2a)



Optimised mutual authentication scheme based on a shared secret key $K_{Alice-Bob}$ with protection against reflection attack

70

Note

- An attacker can now not turn the second message around and use it in the place of the third, as the sender's and receiver's names will be in the wrong order
- Adding structure into the message is one important way of preventing a reflection attack

71

Note

- Protocol MA1 does not suffer from the reflection attack
- it follows principle that **initiator should be first to prove its identity**

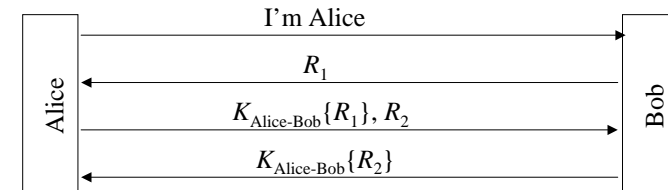
72

Password Guessing

- MA2 (but not MA1) vulnerable to off-line guessing without eavesdropping
- can be fixed with one extra message
- Attacker can then not get plaintext/cipher text pair

73

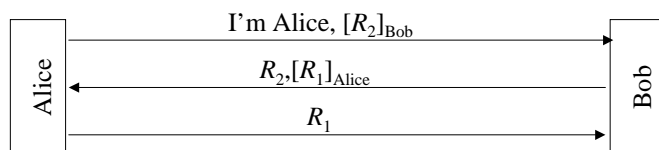
Password Guessing



Mutual authentication scheme based on a shared secret key $K_{\text{Alice-Bob}}$

74

Mutual Authentication with Public Keys – MA3



Mutual authentication with public keys

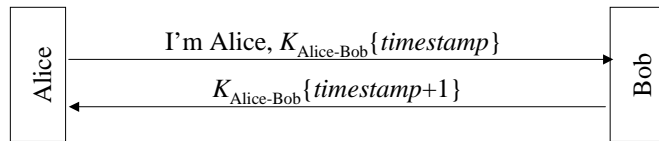
75

Knowing Keys

- how does Alice know Bob's key?
- how does Alice's workstation know Alice's private key?
- keys can be derived from passwords
- Bob's public key can be encrypted using Alice's password
- store a certificate for Bob's public key, signed with Alice's private key

76

Using Timestamps for Mutual Authentication



Mutual authentication based on synchronised clocks and a shared secret $K_{\text{Alice-Bob}}$

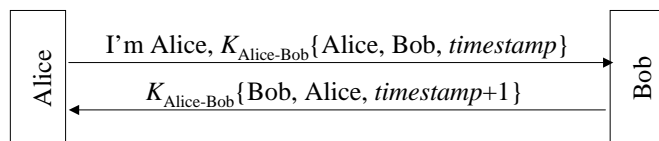
77

Timestamps

- same issues with time as in one-way authentication
 - easy to add onto existing protocols
 - must be used carefully
- note two timestamps slightly different
 - +1 may not be the best possibility
 - as eavesdropper can replay
- there should be an intrinsic difference between what Alice and Bob do

78

Using Timestamps for Mutual Authentication



Note second message could not be sent by an attacker pretending to be Alice

79

What to Look for in Protocols

- messages structured so they cannot be reflected or successfully replayed
- use of timestamp, nonce to guarantee freshness and place of message in protocol
- initiator should authenticate first

80

cont.

- the same key not used for two purposes
 - unless the uses co-ordinated so one protocol can not be used to break another
 - eg., ensure R and/or the full message has some structure
 - then a message in one protocol step can not be mistaken for a message in another

81

Lamport's Hash

- one time password scheme
- allows Bob to authenticate Alice
- neither eavesdropping nor infiltrating Bob's database allows impersonation
- does not use public key cryptography
- Alice remembers password

82

Database

- Bob stores (for each user)
 - user name
 - n , an integer which decrements each time Bob authenticates user
 - $\text{hash}^n(\text{password})$

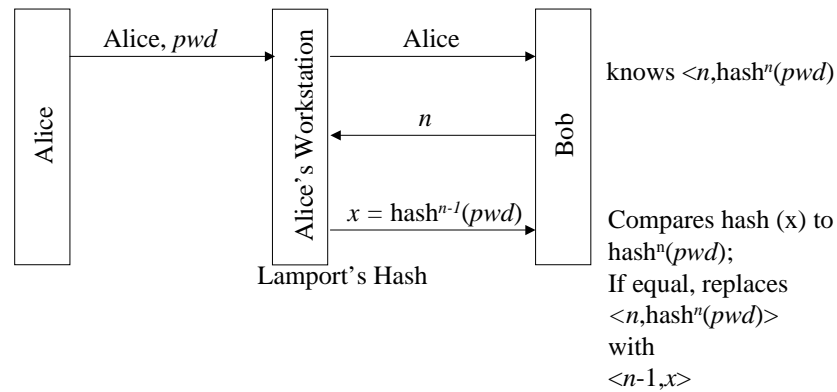
83

Setting Password

- Alice chooses password
- workstation chooses n
- computes $\text{hash}^n(\text{password})$
- sends this to Bob, along with n

84

Lamport's Hash - Protocol



85

Resetting Password

- when n reaches one, password must be reset
- no completely secure way of doing this in absence of encryption or integrity protection

86

Salt

- large random number
- chosen when password set
- stored at Bob
- salt concatenated with password before hashing

87

Advantages of Salt

- can use same password on different servers by using different salts
- do not need to change password when n reaches 1
- instead reinstall same password with different salt
- same advantage as with Unix - harder to crack stolen password file

88

Lamport's Hash - Analysis

- Bob's database is not security sensitive to reading
- can only log in a finite number of time before information at server must be reinstalled
- no mutual authentication

89

Small n Attack

- suppose an attacker can impersonate Bob's address
- when Alice attempts to log in attacker sends a small value for n
- attacker gets Alice's response
- can then impersonate Alice for some time, assuming actual n greater than attacker's chosen value

90

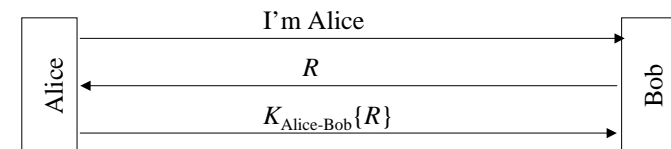
Session Key

- often (but not always) not much point to authentication if don't then get integrity & encryption
- we don't use long term secrets for this
- need to enhance authentication so that at end Bob and Alice share a session key
- eavesdropper must not be able to determine session key

91

Shared Secret - Protocol 1

A simple challenge-response protocol



Bob authenticates Alice based on a shared secret key $K_{\text{Alice-Bob}}$

92

Generating Session Key

- take shared secret, $K_{\text{Alice-Bob}}$, modify it in some way
- then encrypt R using the modified secret and use result as session key
- cannot use R encrypted by secret as session key as that is transmitted

93

Session Key - Desired Properties

- must be different for each session
 - So cannot just modify long term key
- unguessable by eavesdropper
- not consist of quantity X encrypted by secret where X can be predicted or extracted by attacker
- discarded at end of session

94

Generating Session Key

- for Two-Way Public Key Based Authentication
- Alice could choose random R
- encrypt it with Bob's public key
- attacker could hijack conversation by impersonating Alice's address and sending own R

95

Another Option

- As above, except Alice also signs result
- only problem if intruder later infiltrates one node and learns all its secrets
- can then decrypt recorded conversation

96

If Bob doesn't want Alice to do it all

- Alice picks R1, signs it, and sends it to Bob under Bob's public key
- Bob picks R2, signs it, and sends it to Alice under Alice's public key
- session key is exclusive or of R1 and R2
 - or some result using R1 and R2

97

Comments

- infiltration of either node after session ended and key discarded does not allow decryption of recorded conversation
- don't need signatures as intruder cannot decrypt the two halves of the key
- see text for Diffie-Hellman key exchange

98

One-Way Public Key Based Authentication

- only one party (Alice) has private/public key pair
- only Alice needs to prove identity
- Alice cannot be sure that she is talking to Bob
- following does ensure entire conversation occurs with a single entity

99

One-Way Public Key Based Authentication

- Bob chooses random R
- encrypts with Alice's public key
- sends result to Alice
- R is session key
- if attacker later infiltrates Alice can decrypt recorded conversation

100

One-Way Public Key Based Authentication

- Bob and Alice do a Diffie-Hellman exchange
- Alice signs her quantity
- more secure than former
- attacker cannot infiltrate and recover key

101

Lamport Hash

- do Diffie-Hellman key exchange after authentication
- attacker could hijack conversation after authentication and before exchange

102

Lamport Hash

- could do Diffie-Hellman key exchange first
- then do authentication as part of conversation protected by Diffie-Hellman key
- subject to bucket brigade attack
- session key based on shared secret not vulnerable to bucket brigade attack

103

Privacy and Integrity

- We know that there is no standard algorithm for providing integrity and privacy with single key and single cryptographic pass

104

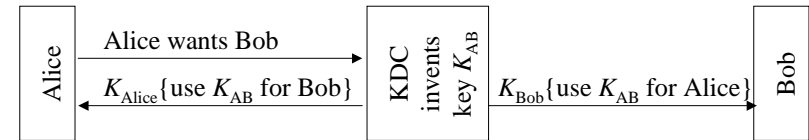
Possible Solutions

- develop two keys in authentication exchange
- make second key by modifying first in predictable manner
- use same key with different algorithms
- use weak checksum inside strong privacy algorithm

105

Mediated Authentication (with KDC)

- KDC does not know it is really Alice



KDC operation (in principle)

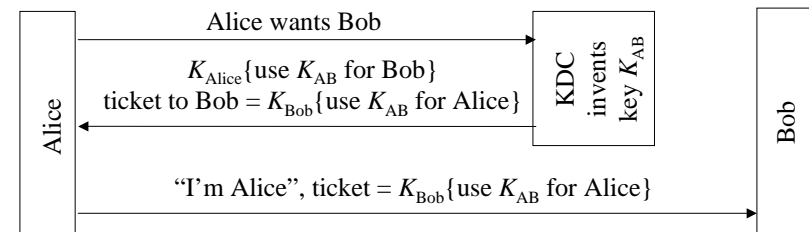
106

In Practice

- not done this way
- Alice's first message to Bob may arrive before KDC's message to Bob
- KDC gives Alice the information to be sent to Bob
- this is how Kerberos works
- needs to be followed by mutual authentication

107

KDC Operation



KDC operation (in practice)

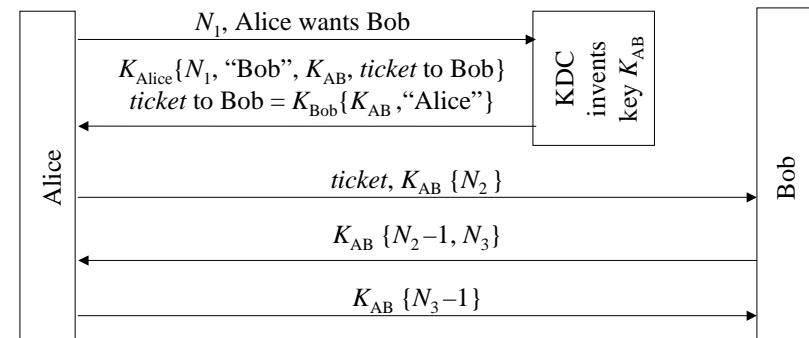
108

Needham-Schroeder

- uses KDC
- completes exchange by mutual authentication
- worth analysing
- Kerberos based on it

109

Needham-Schroeder



110

Message 1

- nonce N_1 is used to assure Alice she is talking to KDC
- avoids (unlikely) situation
 - attacker has old key of Bob's
 - has recorded old message 2
- while unlikely, adding nonce is easy, so why not void the problem?

111

Message 2

- KDC securely gives Alice K_{ab}
- string "Bob" stops attacker trying to substitute their own name in message 1
- ticket is unintelligible to Alice
- Needham-Schroder has been criticised for doubly encrypting ticket
- would be just as secure if ticket was not re-encrypted

112

Message 3

- N_2 is challenge to Bob for authentication
- Alice knows only someone who knows Bob's key can decrypt ticket and discover shared key
- Bob assumes Alice knows shared key as KDC put Alice's name in ticket

113

Messages 4 & 5

- Bob replies to challenge and issues own
- Alice replies in 5

114

Attack

- would be vulnerable to reflection attack if encryption in message was done using (eg.,) DES in ECB mode
- suppose each nonce is 64 bits long
- N_2-1 and N_3 separately encrypted

115

Attack

- suppose Trudy wants to impersonate Alice to Bob
- first she eavesdrops on an authentication handshake where Alice talks to Bob
- Trudy sees messages 3 & 4
- later replays message 3 to Bob
- Bob will respond with $K_{ab}\{N_2-1, N_4\}$

116

Attack

- Trudy can not return $K_{ab}\{N_4-1\}$
- she can open new connection to Bob
- splices in $K_{ab}\{N_4\}$ instead of $K_{ab}\{N_2\}$
- Bob will return $K_{ab}\{N_4-1, N_5\}$
- Trudy can then take first encrypted block, which will have $K_{ab}\{N_4-1\}$ returns that as message 5 of first connection

117

Attack

- Trudy will never know N_4 , but all she needs is $K_{ab}\{N_4-1\}$
- this could not be done if encryption is in CBC mode instead of ECB mode

118

CBC

- if encryption were done in CBC mode decrementing of nonces is not needed
- if pieces of a message sent together the entire message must be integrity protected so parts of it cannot be modified

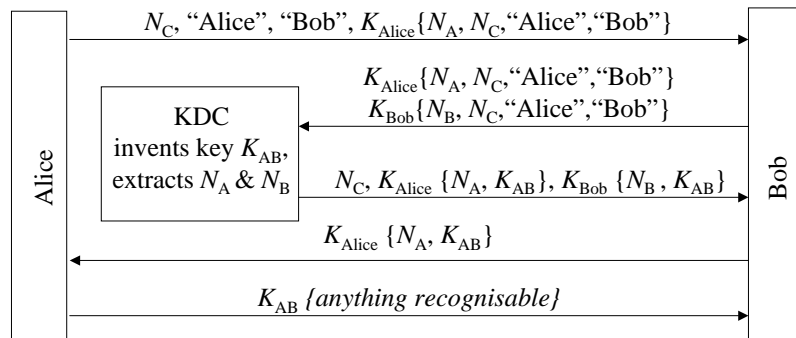
119

Otway-Rees

- solves a problem if attacker has old key of Alice's
- does mutual authentication
- based on idea that suspicious party should always generate challenge

120

Otway-Rees



121

Otway-Rees

- N_C should not just be nonce but should also be unpredictable
- otherwise attacker can add one to current value and intercept next exchange at message 2

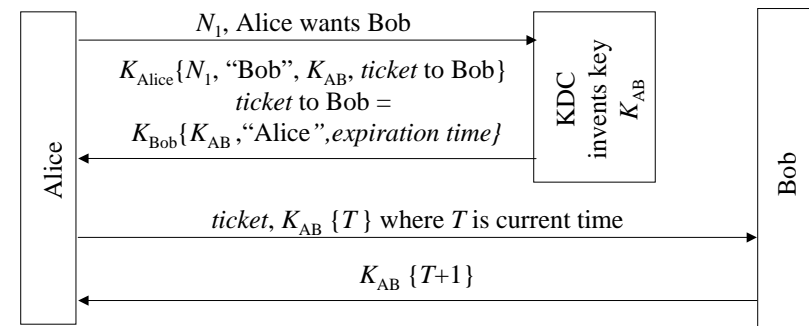
122

Kerberos

- based on Needham-Schroeder
- simpler
- assumes universal idea of time
- includes expiration dates in messages

123

Kerberos Protocol



124

Passwords and Protocols

- in most of the above the user's key could be derived from a password
- whenever a key is derived from a password there is the potential for off-line guessing
- if the attacker can impersonate Bob or Alice may be able to obtain information for off-line guessing

125

Bellovin-Merritt

- designed to prevent off-line guessing
- authentication based on user-chosen password

126

Bellovin-Merritt Scheme 1

- prevents password guessing when attacker
 - eavesdrops
 - pretends to be Alice
 - pretends to be Bob

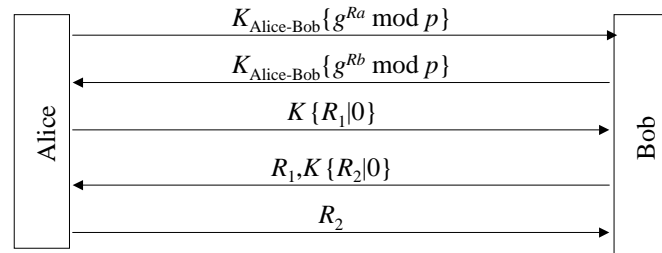
127

Scheme 1

- Alice and Bob do Diffie-Hellman key exchange
- encrypt the values they send to each other with a password
- note that the user's already need a shared secret for this to work

128

Bellovin-Merritt Scheme 1



129

Analysis

- eavesdropping does not work
- attacker can not verify that a guessed password is correct
- encryption of messages 1 & 2 allows authentication
- also without this encryption attacker could do password guessing by impersonating Alice or Bob

130

Scheme 2

- scheme 1 requires Bob to know Alice's password
- scheme 2 requires Bob to store hash of Alice's password

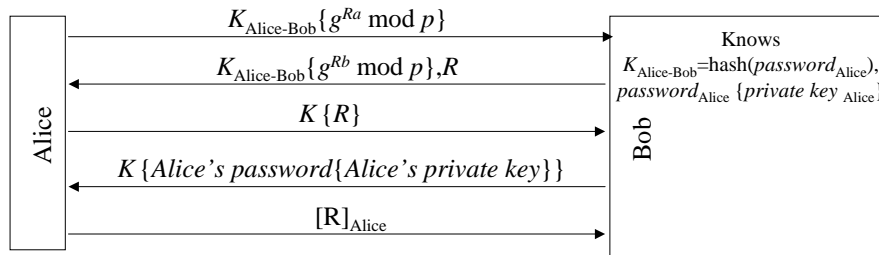
131

Scheme 2

- Alice has a private key
- Bob stores this encrypted under Alice's password
- Bob sends this to Alice after she proves she knows a hash of her password
- she then authenticates again using private key, proving she knows password

132

Scheme 2



133

Scheme 2

- note the following principle
- someone needs to prove they know a low quality secret before being given the high quality secret

134

Low Quality to High Quality

- passwords low quality
- vulnerable to off-line guessing
- private key, unguessable secret key
- high quality
- how to get from one to the other?

135

Basic Outline

- Alice's high quality secret stored at B
- encrypted with Alice's password
- Alice's workstation (A) needs to retrieve this
- then decrypt it

136

Scheme 1

- Alice's information available to anyone who requests it
- A asks for it
- B transmits it to A

137

Scheme 1

- this is often done with private keys, encrypted with password, stored in a directory
- eavesdropper and active attacker can get the stored information

138

Scheme 2

- A must prove knowledge of Alice's password (eg., via hash)
- B then transmits Alice's encrypted high quality secret
- eavesdropping still possible
- active attack does not work (without eavesdropping)

139

Scheme 3

- as scheme 2, with additions
- exchange between A and B encrypted
- key established "on the fly"
- eg.,
 - A or B transmits a public key with which to encrypt the exchange
 - do a Diffie-Hellman exchange

140

Scheme 3

- prevents eavesdropper or active attacker impersonating A getting password-guessing quantity
- attacker impersonating B can still get it

141

Scheme 4

- as in 2, but encrypt exchange
- A has secret or public key for B
- secure, but difficult to administer

142

Scheme 5

- use first two messages of Bellovin-Merritt
- this establishes secret session key between A and B
- then use scheme 1 or scheme 2
- 1 as secure as 2 here as A must know Alice's password to do Bellovin-Merritt
- as secure as 4, but much easier to administer

143