

# Authentication

- Motivation
- Authentication protocols in different scenarios
- X.509, Kerberos

# Authentication

- Alice needs to show her identity to Bob; she needs to get a service, or to access information, or a resource etc.
- Bob needs to be sure of Alice's identity:
  - Who is Alice?
- Trudy tries to impersonate Alice:
  - Once
  - Many times

# Authentication

- The process of reliably verifying the identity of someone (or something).
- In human interaction:
  - People who know you can recognize you based on your appearance or voice
  - A guard might authenticate you by comparing you with the picture on your badge
  - A mail order company might accept as authentication the fact that you know the expiration date on your credit card
- Two interesting cases
  - a computer is authenticating another computer
  - a person is using a public workstation that can perform sophisticated operations, but it will not store secrets for every possible user
    - the user's secret must be remembered by the user

# Closed world assumption

- Presumption that what is not currently known to be true, is false
  - the opposite of the closed world assumption is the open world assumption, stating that lack of knowledge does not imply falsity
- *Negation as failure* is related to closed world assumption, as it believes false every predicate that cannot be proved to be true

# Closed environment

- Authentication in a closed environment (e.g. company, home)
- We use a third party Carole (trusted server) distributing required information for authentication (before authentication or using secret communication)
- Trudy is a legal user and might use the connection Alice-Bob

# Human vs computer authentication

- What's the difference between authenticating a human and authenticating a computer?
- computer can store a high-quality secret, such as a long random-looking number, and it can do cryptographic operations
- a workstation can do cryptographic operations on behalf of the user, but the system has to be designed so that all the person has to remember is a password
- password can be used to acquire a cryptographic key in various ways:
  - directly, as in doing a hash of the password
  - using the password to decrypt a higher-quality key, such as an RSA private key, that is stored in some place like a directory service

# Authentication of people

## Use

- What you know (passwords)
- What you have (smart card)
- Who you are (biometric tools)
- Where you are (network address - weak, because of spoofing)

# Password

- Humans:
  - Short keys, that possibly allow to obtain long keys
  - Sometimes easy to guess
- Computers:
  - Long keys
  - Hidden (not stored in clear): either stored encrypted or indirectly
  - One-time password
- a well known threat is the login Trojan horse: the attacker prompts a fake login window that induces the innocent user to prompt the password that is collected by the attacker



# Biometric

- There are many devices that are used to authenticate:
- Retina examination, fingerprint reader, voice, or based on other characteristics (ex. hand written signatures, keystroke timing: timing in using the keyboard or the mouse etc.)
- Accuracy:
  - Error: false positive and false negative
  - They are used in specific scenarios (sometimes to enforce other methods)

# Biometric

- Fingerprinting:
  - Fake fingerprint, dead people
  - Not stable over time: children, people doing manual work (possible damage)
- Voice
  - Use of tape;
  - voices are affected: flu and colds; noise in the background, telephone etc

# Biometric

- Keystroke timing:
  - Each person has different speed in typing
  - Typing faster than personal limit is impossible
- Handwritten signature:
  - Poor quality
  - Electronic tablet, for signature + timing

# Authentication scenarios

In the following we will consider authentication using knowledge of a key (public key or secret key, possibly stored in a smart card)

1. Users (Alice and Bob) share a secret key
2. Users share a key with Carole, trusted authority (authentication server)
3. Users have a public key

# Techniques

Users are authenticated using a key

Techniques:

- timestamp
- nonce (or challenge): random number chosen by the person that authenticates to verify the other knows the key
- sequence number in protocols

# Cryptography vs. authentication

- Trudy attacks the protocols using non authentic messages (possibly messages that have been exchanged between Alice and Bob in previous application of the protocol)
- Need to guarantee authentication and integrity
- Encryption DOES NOT guarantee authenticity of the message

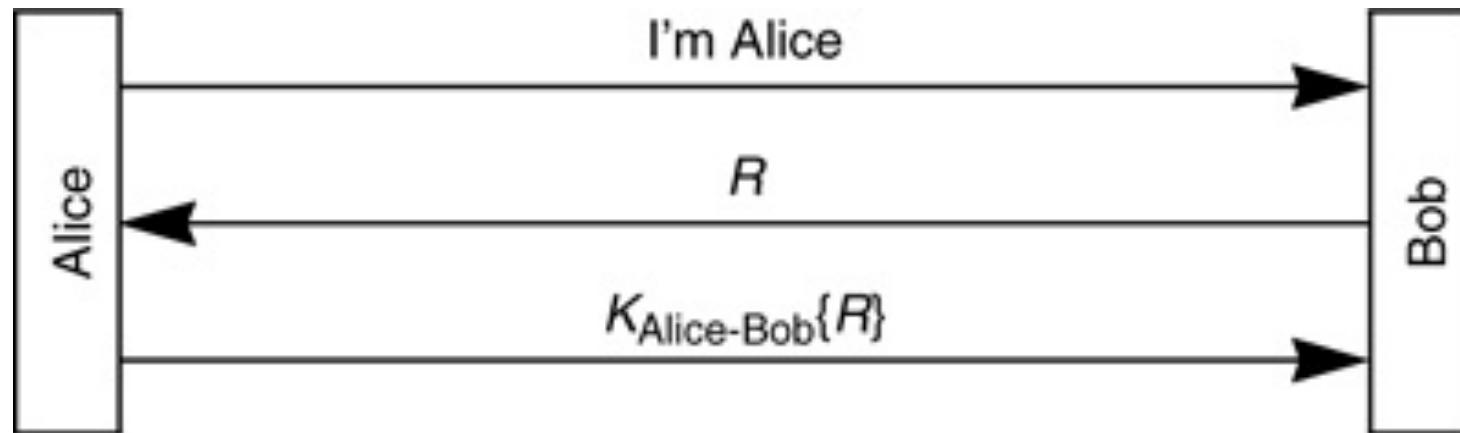
# Authentication by symmetric key

Alice and Bob share a secret  
key  $K_{\text{Alice-Bob}}$

$K\{M\}$  denotes  $M$  encrypted with secret key  $K$

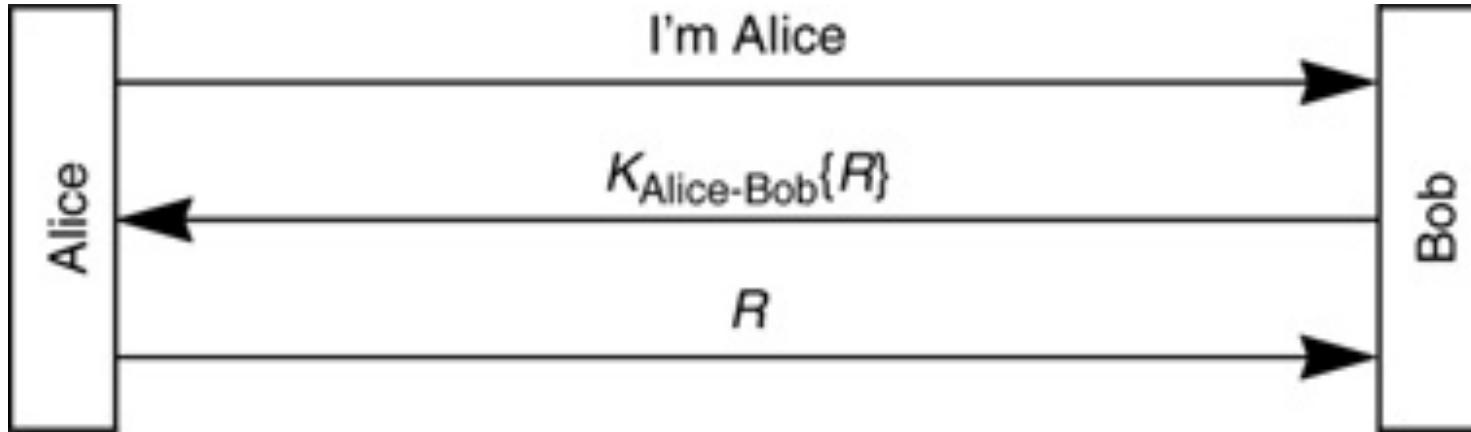
- One way authentication using nonce (challenge)
- One way authentication using timestamps
- Two ways (mutual) authentication using nonce

# challenge/response based on shared secret



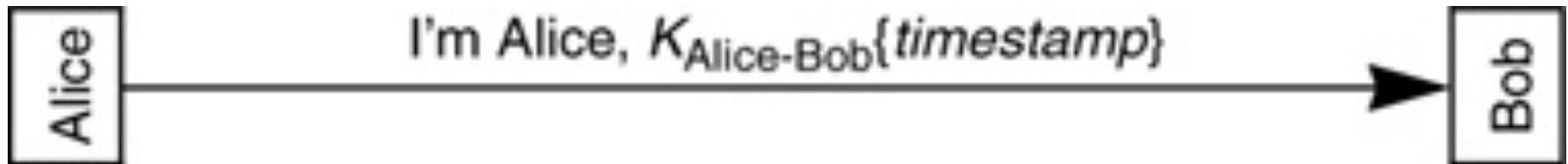
- authentication is not mutual. Bob authenticates Alice, but Alice does not authenticate Bob
- an eavesdropper could mount an off-line password-guessing attack (assuming  $K_{Alice-Bob}$  is derived from a password), knowing R and  $K_{Alice-Bob}\{R\}$
- R is a **nonce**

# variant



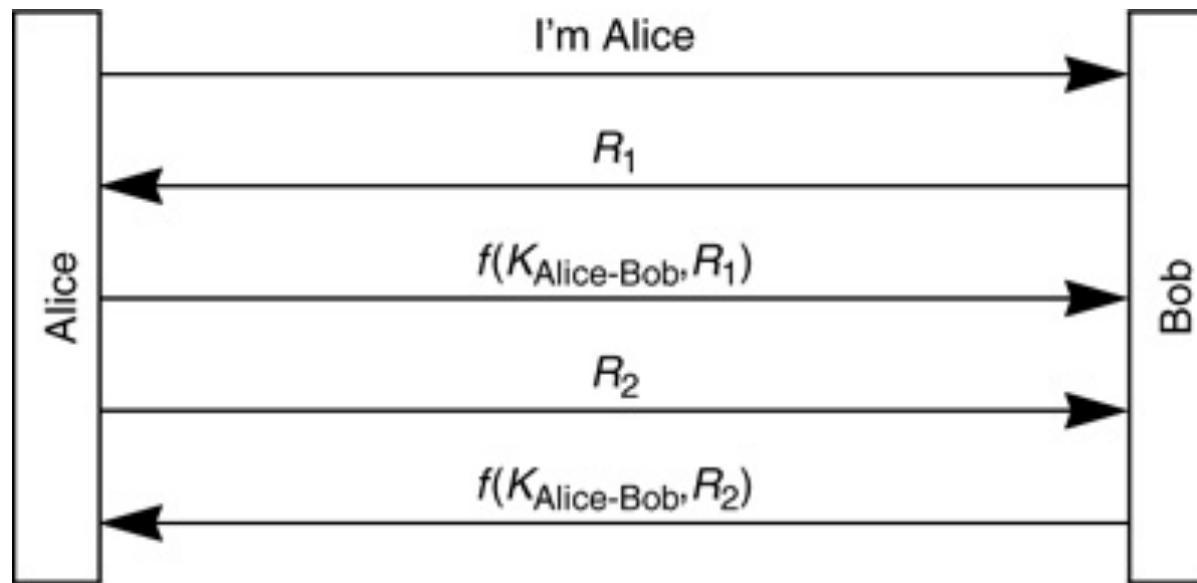
- requires reversible cryptography
- still possible the dictionary attack by eavesdropper
- if  $R$  has limited lifetime (e.g., random number + timestamp) Alice authenticates Bob because only someone knowing  $K_{\text{Alice-Bob}}$  could generate  $K_{\text{Alice-Bob}}\{R\}$ 
  - limited lifetime required to foil **replaying** of an old  $K_{\text{Alice-Bob}}\{R\}$

# timestamp based



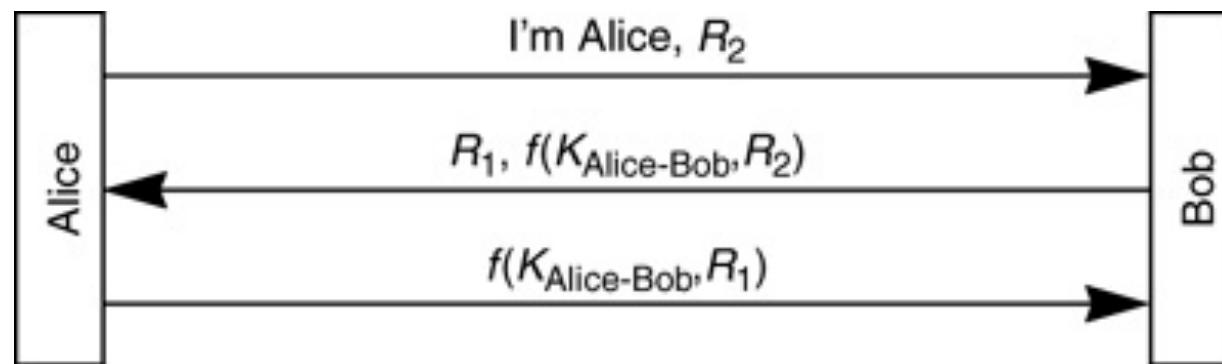
- Bob and Alice have reasonably synchronized clocks (can be a weakness)
- Alice encrypts the current time, Bob decrypts the result and makes sure the result is acceptable (i.e., within an acceptable clock skew)
- efficient, no intermediate states
- if Bob remembers timestamps until they expire, then no replaying attacks
- if multiple servers with same secret  $K$ , then Alice can send  $K\{\text{Bob} \mid \text{timestamp}\}$

# mutual authentication



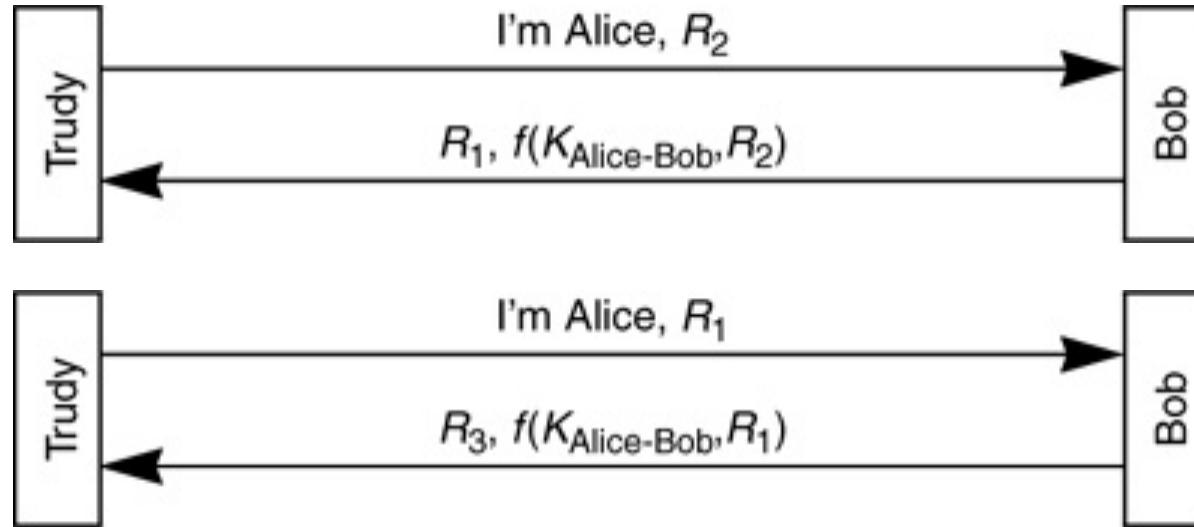
- many messages!
- perhaps a few ones can be saved...

# optimized mutual authentication



- save messages but weak to reflection attack

# reflection attack

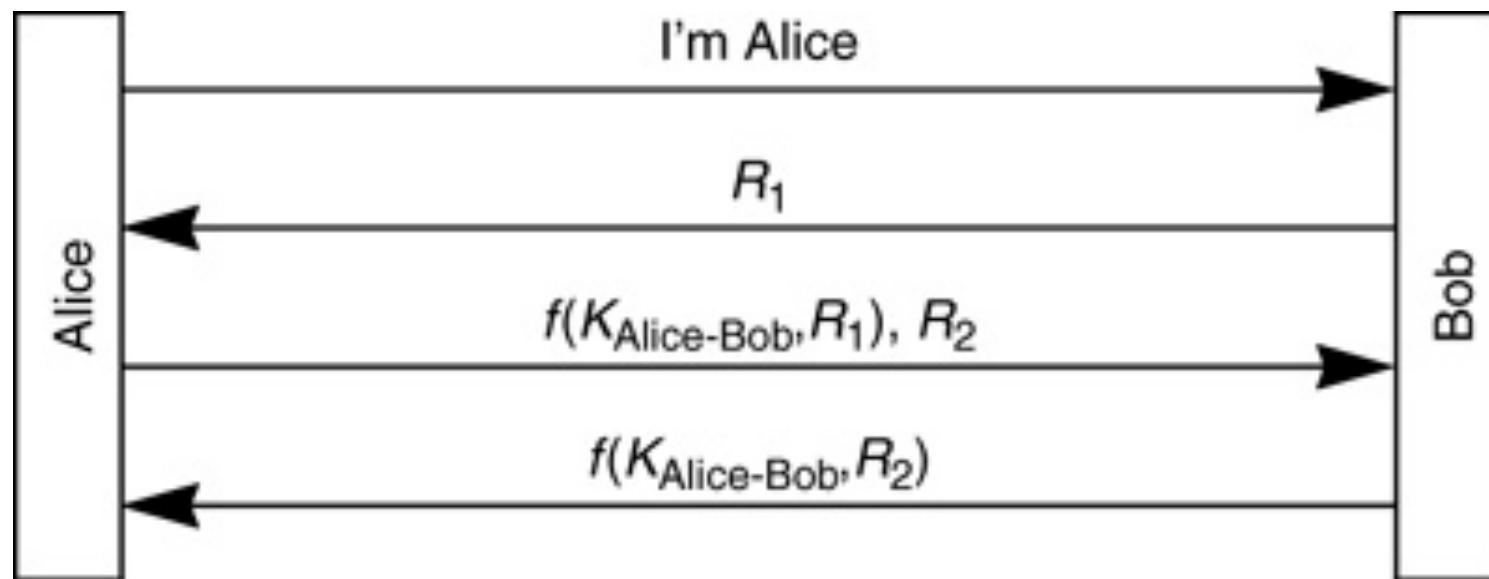


- Trudy wants to impersonate Alice to Bob
- two sessions, the 2<sup>nd</sup> will be incomplete but will allow Trudy to complete the 1<sup>st</sup> one

# how to prevent reflection attack?

- use different keys
  - $K_{\text{Alice-Bob}}$  and  $-K_{\text{Alice-Bob}}$ , or  $K_{\text{Alice-Bob}} + 1$
- different challenges, i.e. challenge from initiator different from challenge from responder
  - e.g., even number at initiator's side, odd number at responder's side

# less optimized mutual authentication



- Trudy can mount an off-line password-guessing attack by impersonating Bob's address and tricking Alice into attempting a connection to her

# Trusted party

It is not possible that all users share a secret key  
(quadratic number of keys, each user must have  
a different key for everyone)

Scenario: users share a key - password - with  
trusted authority  $C$  -  $C$  authentication server  
(aka Key Distribution Center (KDC))

- A and B share a secret key with  $C$  ( $K_{AC}$  e  $K_{BC}$ )
- A and B might not be human user but also entity  
of the system (e.g., printer, databases etc.)
- Goals: authenticate A (or A and B);
  - optional: decide on a session key to be used between  
A and B for short time

# Authentication server

Goal: Alice and Bob must authenticate and choose a secret session key  $K$  - used for short time (session)

At the end of the authentication protocol:

1. only A and B know  $K$  (besides C - trusted server)
2. Each should be sure of fact 1 above
3.  $K$  is a new key (randomly chosen, not used before)

# Authentication server - attacker's strength

Trudy (attacker) can

- be a legitimate user of the system (share a key with C)
- sniff and spoof messages
- concurrently run more than one session with A, B and C
  - different execution of the protocol can be done interleaved
  - T is able to convince A and/or B to start a new session with T
- Might know old session keys

# Authentication server - attacker's limits

Trudy

- is **NOT** able to guess random numbers chosen by A or B
- **does not know keys  $K_{AC}$  and  $K_{BC}$**  (in general does not know secret keys of other users)
- is not able to decode in a short time messages encrypted with unknown keys

# Authentication with trusted server

A and B share a key with C ( $K_{AC}$  and  $K_{BC}$ , respectively)

1. A sends to C: A, B
2. C chooses K - session key - and sends to A:  
 $K_{AC}(K)$  and  $K_{BC}(K)$
3. A decodes and computes K and sends to B:  
 $C, A, K_{BC}(K)$
4. B decodes  $K_{BC}(K)$  finds K and sends to A:  
 $K(\text{Hello A, this is B})$

# Attack - 1

Assume T can sniff, spoof and make MITM attack  
(T is in the middle of both A to C and A to B communication)

1. A sends to T (instead of A sends to C) : A,B
2. immediately T sends to C: A,T
3. C chooses K and sends to A:  $K_{AC}(K)$  and  $K_{TC}(K)$
4. A sends to B: C, A,  $K_{TC}(K)$  - T intercepts
5. T sends to A K(Hello A, this is B)

Possible modification of step 1 (previous slide):

A sends to C  $\langle A, K_{AC}(B) \rangle$  (in this way C knows that A wants to talk to B)

# Attack - 2

Modified protocol

1. A sends to C:  $A, K_{AC}(B)$
2. C chooses K, and sends to A:  $K_{AC}(K)$  and  $K_{BC}(K)$
3. A decodes K and sends to B:  $C, A, K_{BC}(K)$
4. B decodes K and sends to A : K(Hello A, this is B)

Note: T does not know that A wants to talk to B

# Attack - 3

Consider modified protocol

1. A sends to C : A,  $K_{AC}(B)$

T (as A) gets A's message and sends to C: A,  $K_{AC}(T)$   
(REPLAYS part of some previously exchanged message)

Note: T does not know whom A wants to talk to

2. C chooses K, and sends to A :  $K_{AC}(K)$  and  $K_{TC}(K)$

3. A decode K and sends to B : C, A,  $K_{TC}(K)$

4. T gets the message and finds that A wants to talk to B; T now acts in place of B and sends to A K(Hello A, this is B)

Note: T knows the identity the person A wants to talk to only at the end of the protocol

# Protocol Needham-Schroeder

*basis for the Kerberos protocol and aims to establish a session key between two parties on a network, typically to protect further communication*

N, nonce: random number

N.-S. protocol based on challenge-response :

1. A chooses N (nonce) and sends to C: A, B, N
2. C chooses K and sends to A:  $K_{AC}(N, K, B, K_{BC}(K, A))$
3. A decodes, checks N and B, and sends to B:  $K_{BC}(K, A)$
4. B decodes, chooses nonce N' and sends to A:  
 $K(\text{this is } B, N')$
5. A sends to B  $K(\text{this is } A, N' - 1)$   
*now B has checked A*

Note: B does not directly communicate with C

# Attacking N.-S. Protocol

1. A chooses N (nonce) and sends to C: A, B, N
2. C chooses K and sends to A:  $K_{AC}(N, K, B, K_{BC}(K, A))$   
now T acts in place of A
3. A decodes, checks N and B and sends to B:  $K_{BC}(K, A)$   
T (as A) replays to B:  $K_{BC}(K', A)$ , where K' is an older session key
4. B decodes, chooses nonce N' and sends to T (not to A) K' (this is B, N')
5. T sends to B: K' (this is A, N' - 1)

Note:

1. T uses old session key and acts in place of A
2. B is not sure that C is active: there is no direct exchange between B and C

# Authentication attack (Denning and Sacco, '81)

- two sessions
- assume that Trudy has recorded session 1 and that K is compromised
- after session 2, B is convinced that he shares the secret key K only with A

# session 1, session 2

1.  $A \rightarrow C: A, B, N$
2.  $C \rightarrow A: K_{AC}(N, B, K, K_{BC}(K, A))$
3.  $A \rightarrow I(B): K_{BC}(K, A)$

*assume that K is compromised*

4.  $I(A) \rightarrow B: K_{BC}(K, A)$       *replay*
5.  $B \rightarrow I(A): K(N')$
6.  $I(A) \rightarrow B: K(N' - 1)$

*B is now convinced that he shares secret key K only with A*

# Challenge-response symmetric key

How to guarantee data integrity

- timestamps (a message is valid only in a small time window)
- sequence number (A and B remember sequence number of exchanged messages to avoid replay attacks in which the attacker sends old messages)
- nonce should be used carefully

# Needham-Schroeder Protocol (variant)

Modified Challenge-Response (nonce and timestamp):

- C exchanges messages with both A and B
  - timestamp  $\dagger$  used only between B and C
  - K session secret key
1. A chooses N and sends to B:  $\langle A, N \rangle$
  2. B chooses  $N'$  and sends to C:  $\langle B, N', K_{BC}(N, A, \dagger) \rangle$
  3. C sends to A:  $\langle K_{AC}(B, N, K, \dagger), K_{BC}(A, K, \dagger), N' \rangle$
  4. A sends to B:  $\langle K_{BC}(A, K, \dagger), K(N') \rangle$

*Basis for the Kerberos protocol*

# Expanded Needham-Schroeder

