Distributed Systems
COMP 212

Lecture 25

Othon Michail

UNIVERSITY OF
LIVERPOOL

Security

Security: Mechanisms and Policies

- Trying to build a system that is protected from *all possible types* of security threats does not make sense
  - First a Security policy
  - Then it becomes possible to implement security mechanisms that enforce the policy
- Whether a user considers a system to be secure is a matter of trust
- Difficult to get right, impossible to get perfect!

Security Topics

1. Providing *secure communication*
   - Secure channel
   - authentication, confidentiality and message integrity
2. Handling *authorisation*
   - Access control
   - who is entitled to use what in the system?
3. Providing effective *Security Management*
   - Mechanisms for achieving (1) and (2)
   - cryptographic keys distribution, add/remove users, certificates

Types of Security Threats

- **Interception**: unauthorised access to data, e.g.,
  - communication eavesdropping
  - Illegally copying data
- **Interruption**: a service or data become unavailable, e.g.,
  - lost/corrupted files
  - Denial of Service attacks
- **Modification**: unauthorised changes to, and tampering of, data, e.g.,
  - changing transmitted data
  - changing a program so that it logs user's activity
- **Fabrication**: non-normal, additional activity, e.g.,
  - intruder adding entries to a password database

Security Mechanisms

- **Encryption**: transforms data into something an attacker cannot understand
  - fundamental technique
  - used to implement confidentiality and integrity
  - can even detect modification of data (data integrity)

- **Authentication**: verifying claimed identity of users, client, server
  - before service is provided
  - Typically via passwords

- **Authorisation**: verifying allowable operations
  - e.g., a user may be allowed to log in and view a medical database but not necessarily to alter its entries

- **Auditing**: who did what to what and when/how did they do it?
  - Logging activities
  - Not direct protection, but useful for tracing and analysing a security breach

Design Issue: Simplicity

- Designing a secure computer system is considered a difficult task
- The ideal: A few simple mechanisms that are easily understood and trusted to work
- Importance of simplicity:
  - Users tend to trust simple systems, and therefore use them
  - Easier to convince designers that there are no security holes in the system
- Unfortunately, the real world is not this clear cut, as introducing security mechanisms to an already complex system can often make matters worse
- However, this is still a design goal to aim for!

A Fundamental Security Mechanism

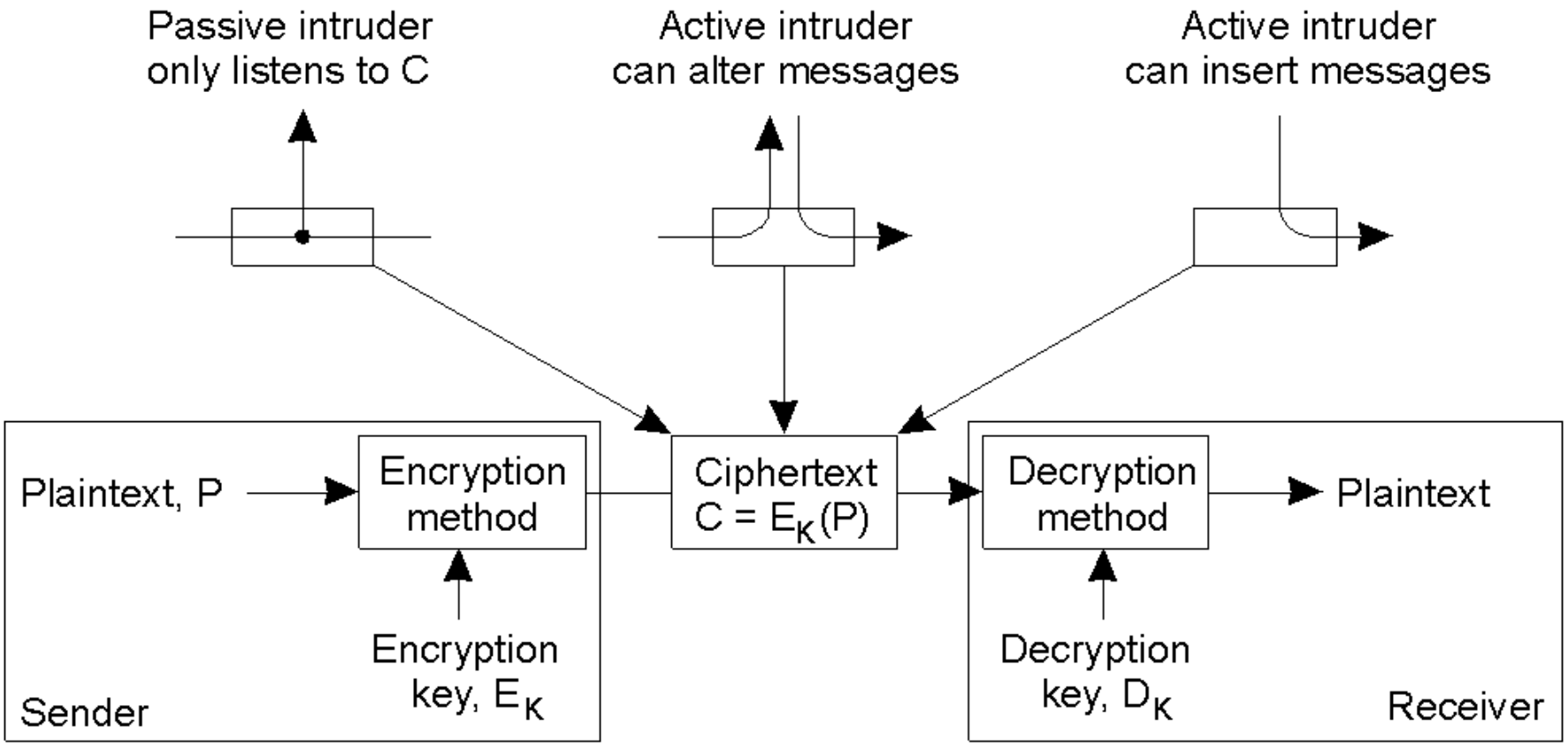
- Fundamental technique within any distributed system's security environment:

Cryptography

- Simple basic idea:
  - Alice wants to transmit a message *m* to Bob
  - But Chuck is constantly trying to ``attack'' the message

  To protect the message:
  - Alice encrypts it into an unintelligible *m'* and sends *m'*
  - Bob must decrypt *m'* in order to obtain *m*
  - Chuck does not know how to decrypt!

Participants/Components



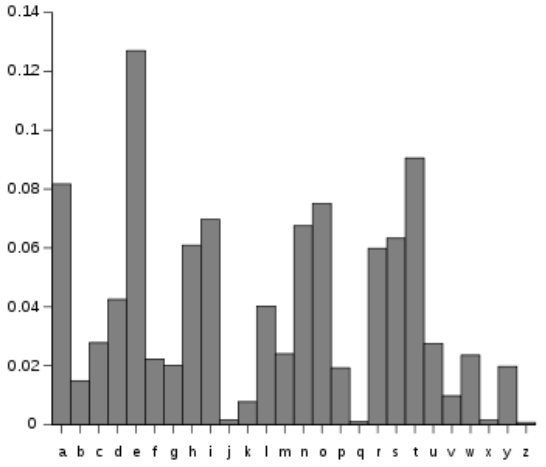
- Intruders and eavesdroppers in communication

Example: Caesar Cipher

- Used by Julius Caesar to communicate to his army
  - One of the oldest uses of cryptography in practice
- Shift each letter a certain number of spaces
  - For example, for a shift of 19

A B C D E F G H I J K L M N O P Q R S T U V W X Y Z
T U V W X Y Z A B C D E F G H I J K L M N O P Q R S

- Easily broken
  - e.g., by letter frequency analysis



Much Better: One-Time Pads

- Imagine two identical pads of paper
  - On each page there is a **random** text

- To encrypt:
  1. Take first/next character of plaintext message and first/next character from the pad
  2. Take their corresponding numbers, e.g., their position in the alphabet
  3. Add their numbers modulo 26 (# English letters)
  4. Convert the result to the corresponding letter and add this to the ciphertext. Go back to 1.

- To decrypt:
  - The inverse procedure

- To ensure security:
  - Burn pads so that they are never used again
  - Secret agents: highly flammable nitrocellulose

One-Time Pads Example

- Encrypt HELLO by using XMCKL from the pad

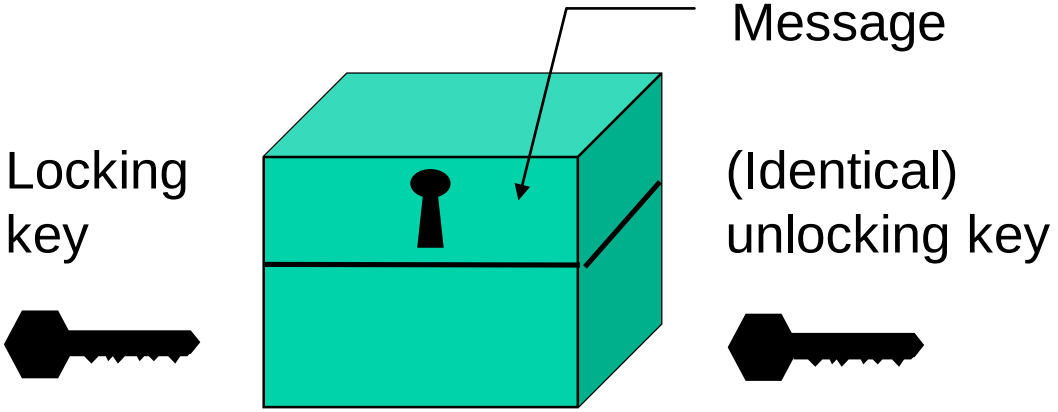
```
      H        E        L        L        O  message
    7 (H)    4 (E)   11 (L)   11 (L)   14 (O) message
+  23 (X)   12 (M)    2 (C)   10 (K)   11 (L) key
=  30       16       13       21       25     message + key
=   4 (E)   16 (Q)   13 (N)   21 (V)   25 (Z) (message + key) mod 26
       E        Q        N        V        Z  → ciphertext
```

- Decrypt transmitted cipher EQNVZ by using XMCKL from the pad

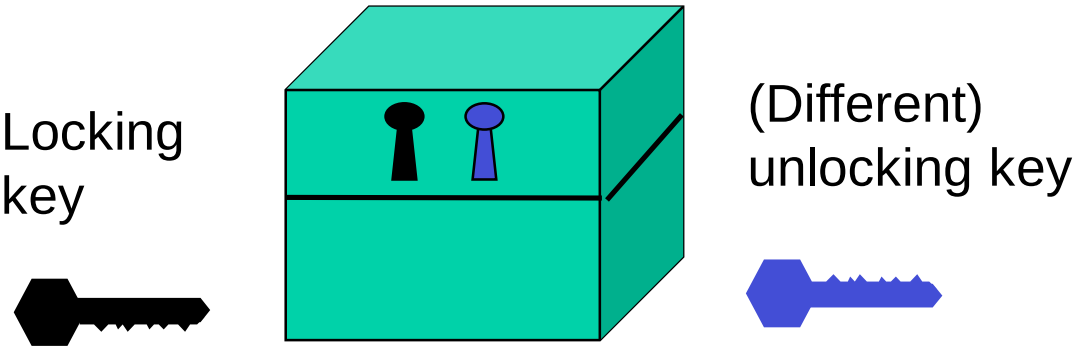
```
      E        Q        N        V        Z  ciphertext
    4 (E)   16 (Q)   13 (N)   21 (V)   25 (Z) ciphertext
-  23 (X)   12 (M)    2 (C)   10 (K)   11 (L) key
= -19        4       11       11       14     ciphertext - key
=   7 (H)    4 (E)   11 (L)   11 (L)   14 (O) ciphertext - key (mod 26)
       H        E        L        L        O  → message
```

Two Kinds of Keys

Symmetric lock and key



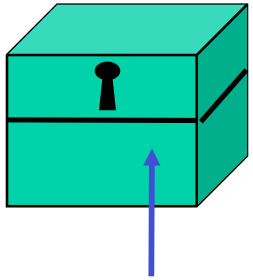
Asymmetric lock and key

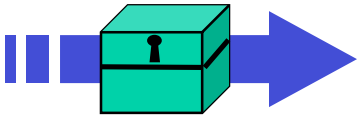


Symmetric case

Recipient opens using identical (symmetric) unlocking key

$E_k(M)$





$D_k(E_k(M)) = M$

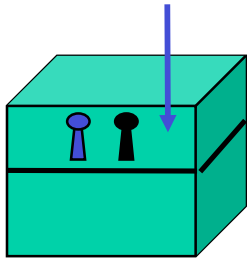
Put message in lockbox

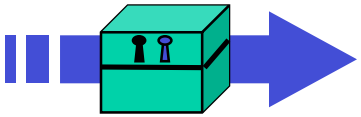
Close and lock using sender's locking key

Transport to recipient

$D_{k2}(E_k(M)) = M$

$E_k(M)$





Asymmetric case

Recipient opens using different (asymmetric) unlocking key

Secret Communication with a Shared Secret Key

- Alice and Bob share a secret key *k*
- Alice uses *k* and an agreed encryption function *Ek*(*M*) to encrypt and send any number of messages *M* to Bob
- Bob reads the encrypted messages using the corresponding decryption function *Dk*(*M*)

- Alice and Bob can go on using *k* as long as it is safe to assume that *k* has not been compromised

Added Value by a Public Key

- Asymmetric cryptosystems are also referred to as **public-key systems**
  - One of the two keys can be made public
- For Alice to send a confidential message to Bob, she uses Bob's public key
- Only Bob can decrypt this message (with his private key)
- In addition, Bob also wants to be sure the message is actually from Alice, so Alice uses her private key to sign the message, and Bob uses Alice's public key to decrypt it
- If a correctly formatted message appears, Bob knows Alice sent it

Notation for Cryptography

Notation	Description
$K_{A, B}$	Secret key shared by A and B
K_A^+	Public key of A
K_A^-	Private key of A

Symmetric Encryption Algorithms

- **DES**: The US Data Encryption Standard (1977)
  - No longer strong in its original form
  - Successfully cracked by brute-force attack in a competition, took about 12 weeks to break with tens of thousands PCs involved
  - 56-bit key, 350 kbytes/sec
- **Triple-DES**: Applies DES three times with two different keys
  - encrypt-decrypt-encrypt
  - quite safe
  - 112-bit key, 120 Kbytes/sec
- **IDEA**: International Data Encryption Algorithm (1990)
  - 128-bit key, 700 kbytes/sec
- **AES**: A proposed US Advanced Encryption Standard (1997)
  - 128/256-bit key
- The above speeds are for a Pentium II processor at 330 MHZ

Public-Key Cryptosystem

- **RSA**: The first practical algorithm (Rivest, Shamir and Adelman 1978) and still the most frequently used
  - Based on the fact that no efficient method is known to find the prime factors of large numbers
  - Each integer can be written as a product of prime numbers
    - e.g. 2100 = 2 x 2 x 3 x 5 x 5 x 7
  - Construct the keys based on large prime numbers
  - Key length is variable, 512-2048 bits. Speed 1-7 kbytes/sec. (350 MHz PII processor)

- **Elliptic curve**: A recently-developed method, shorter keys and faster

- Asymmetric algorithms are ~1000 x slower and are therefore not practical for bulk encryption, but their other properties make them ideal for key distribution and for authentication uses

Public Key vs Shared Secret Key

- Secret key encryption has better performance
  - Longer keys can be used in practice
  - Longer keys are harder to break
- Secret keys are more secure for the same size key
- Need a separate key for every two participants
  - e.g., 100 people would need 4950 different keys
- Public keys allow much more convenient key management
- Public keys support a greater range of security functions

Use Both!

- e.g., transmit a randomly generated secret key across a public key-based secure channel

Applications of Cryptography

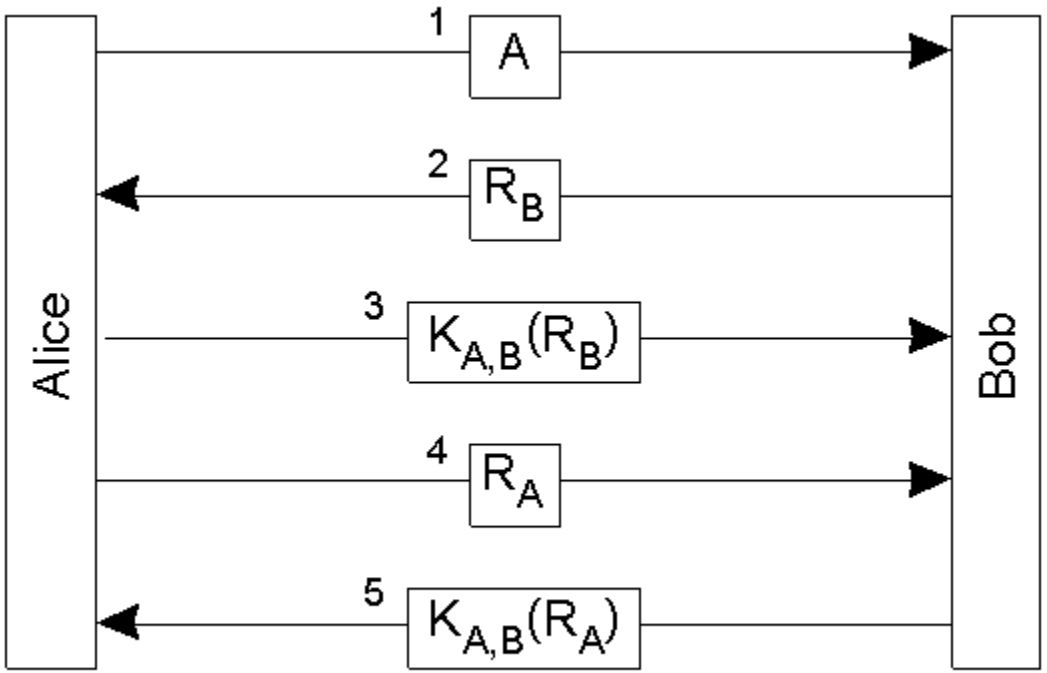
1. Authentication

2. Message Integrity

3. Confidentiality

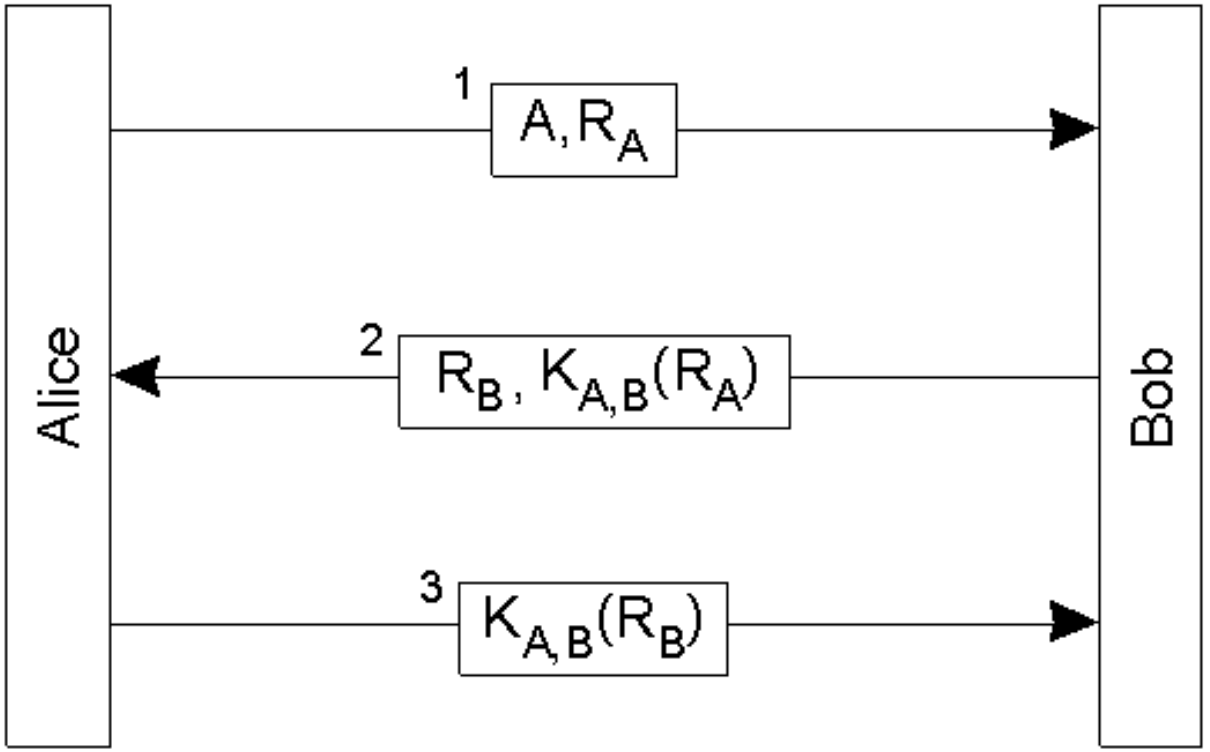
- Note that authentication and message integrity are technologies that rely on each other

Authentication



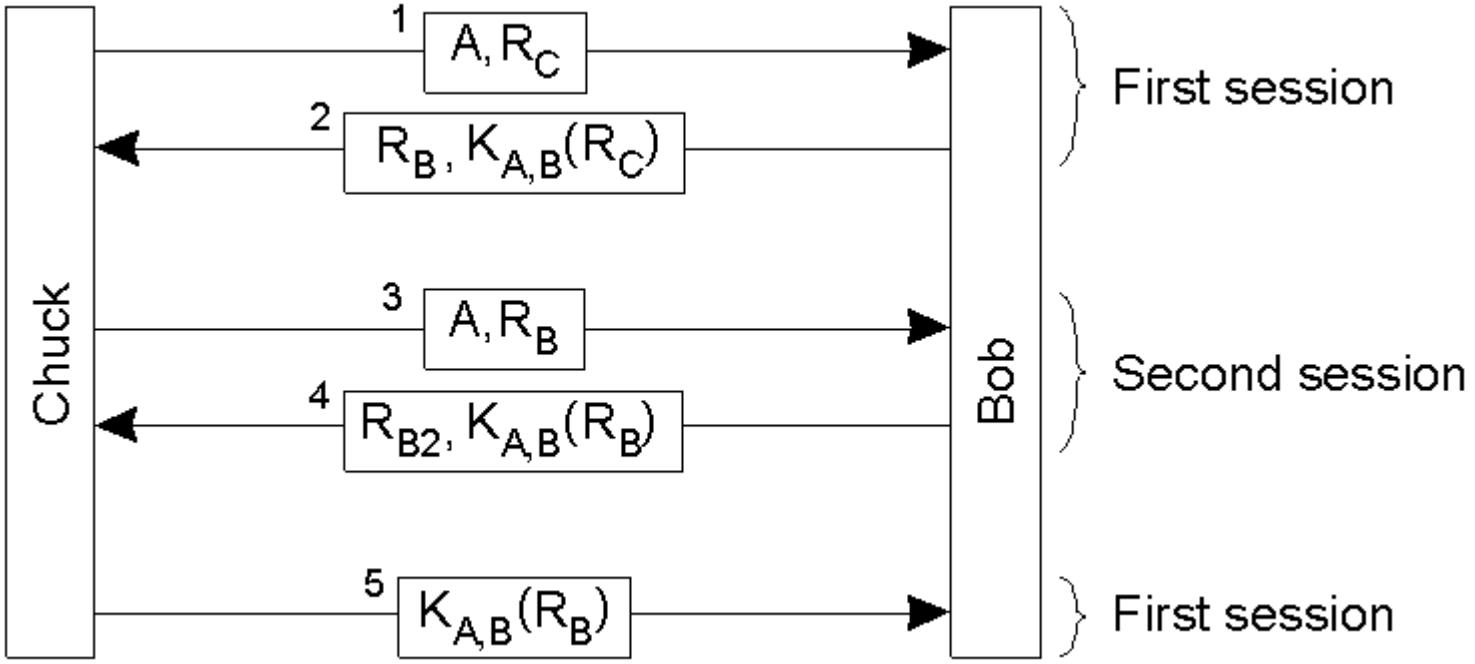
- Authentication based on a shared secret key, using a "challenge response" protocol
- Note: *R* is a random number

An Authentication Optimisation??



- Authentication based on a shared secret key, but using three instead of five messages
- Is this better??

An Authentication Attack!!



The "reflection attack":

- Chuck wants Bob to think he is Alice, so he starts up a second session to trick Bob

Lesson Learnt

- Designing security protocols is a tricky business!