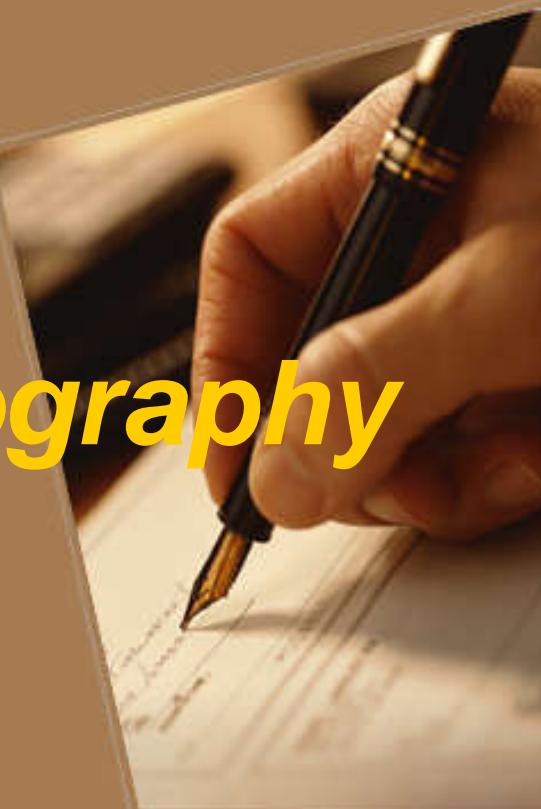


# *Introduction to Cryptography*

- ➊ Summarized from “*Applied Cryptography, Protocols, Algorithms, and Source Code in C*”, 2nd. Edition, Bruce Schneier, John Wiley & Sons, Inc.



# *Outline*

- Introduction
- Cryptographic Protocols

# *Introduction*



# Terminology

## ● Scenario

- A **sender** wants to sent a message to a **receiver** securely, that is, to make sure an **eavesdropper** cannot read the message

## ● Messages and Encryption

- **Plaintext:** the message
- **Ciphertext:** the encrypted message
- **Encryption:** disguising a message to hide its substance
- **Decryption:** turning ciphertext back into plaintext



# **Mathematical Notations**

## ● Symbols

- Plaintext:  $M$  (for message) or  $P$  (for plaintext)
- Ciphertext:  $C$
- Encryption function:  $E$
- Decryption function:  $D$

## ● Formulations

- $E(M)=C$ , the encryption function operates on plaintext to produce ciphertext
- $D(C)=M$ , the decryption function operates on ciphertext to produce plaintext
- $D(E(M))=M$ , the quality must hold in order to recover the original identity

# *Goals of Cryptography*

- Confidentiality
- Authentication
  - Receiver must be able to ascertain the message's origin
- Integrity
  - Receiver shall be able to verify that the message is not modified in transit
- Non-repudiation
  - Sender should not be able to falsely deny later that he sent a message

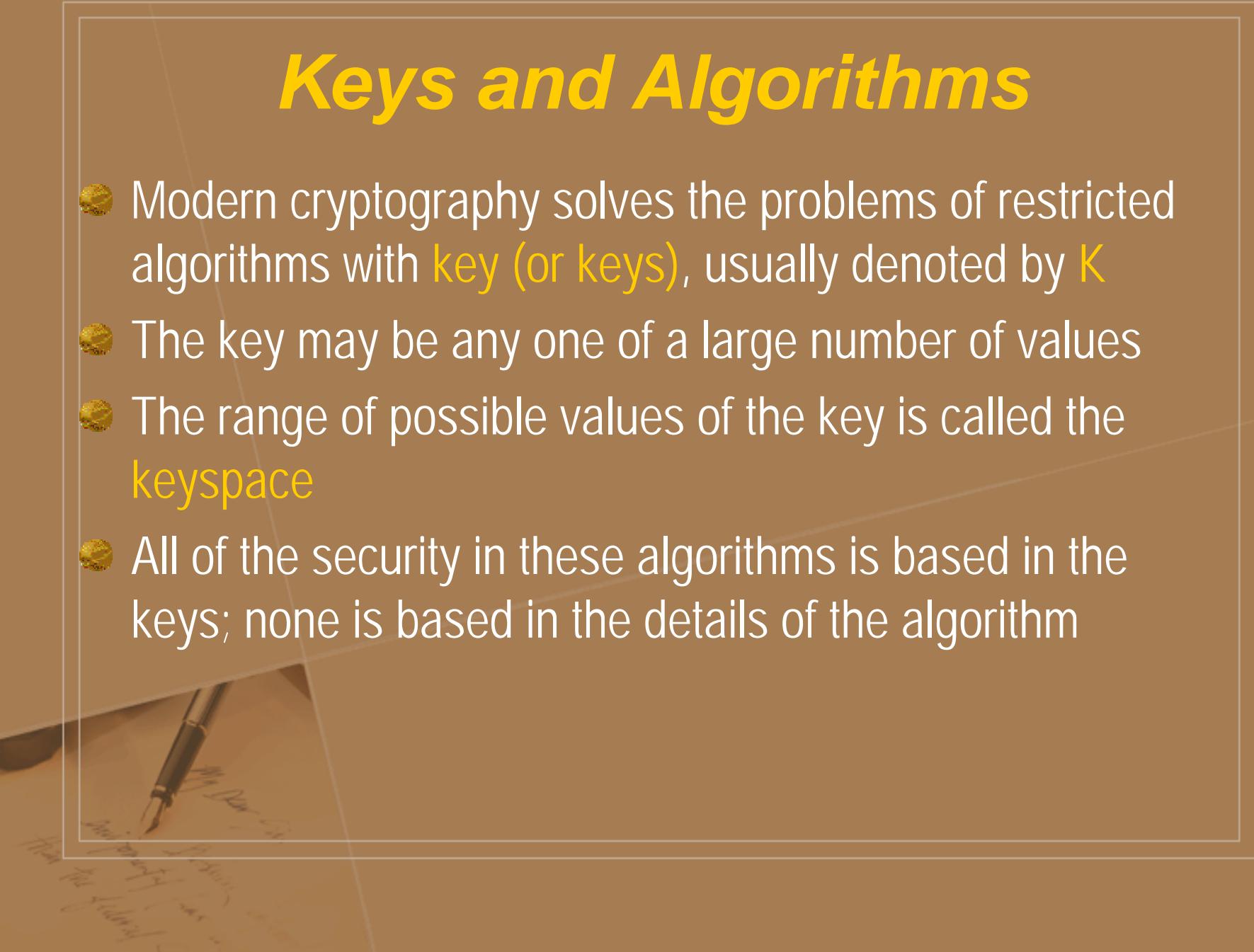
# ***Restricted Algorithms***

## ● Basics of Restricted Algorithms

- The security of an algorithm is based on keeping the way that algorithm works a secret
- Of historical interests only, inadequate in today's applications
  - Frequent changes of algorithm due to user-leaving
  - Difficult to test the security of adopted algorithms
- Widely used in low-security applications

# **Keys and Algorithms**

- Modern cryptography solves the problems of restricted algorithms with **key** (or **keys**), usually denoted by **K**
- The key may be any one of a large number of values
- The range of possible values of the key is called the **keyspace**
- All of the security in these algorithms is based in the keys; none is based in the details of the algorithm



# *Cryptosystem*

- A cryptosystem is composed of
  - An algorithm
  - All possible plaintexts
  - All possible ciphertexts
  - All keys



# **Encryption/Decryption with Keys**

- Both encryption and decryption operations use the same key

$$E_K(M)=C$$

$$D_K(C)=M$$

$$D_K(E_K(M))=M$$



- Different encryption and decryption keys are used

$$E_{K_1}(M)=C$$

$$D_{K_2}(C)=M$$

$$D_{K_2}(E_{K_1}(M))=M$$



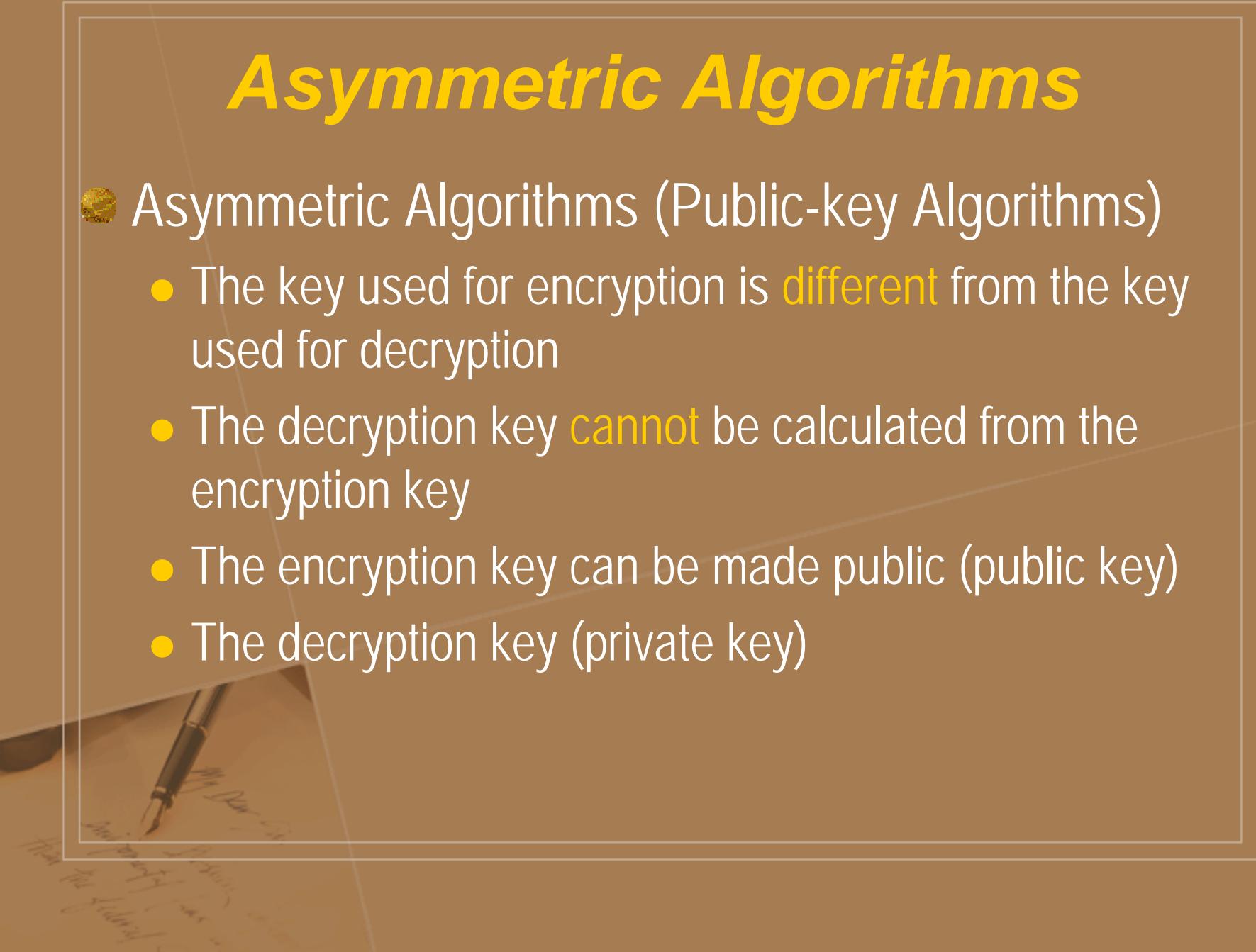
# **Symmetric Algorithms**

- Symmetric Algorithms (Conventional Algorithms)
  - Algorithms where the encryption key can be calculated from the decryption key and vice versa
    - In most symmetric algorithms, the encryption key and the decryption key are the same
  - Categories
    - Stream algorithms
    - Block algorithms

# **Asymmetric Algorithms**

## ● Asymmetric Algorithms (Public-key Algorithms)

- The key used for encryption is **different** from the key used for decryption
- The decryption key **cannot** be calculated from the encryption key
- The encryption key can be made public (public key)
- The decryption key (private key)



# *Cryptanalysis*

## ● Overview of cryptanalysis

- The science of recovering the plaintext of a message without access to the key
- Successful cryptanalysis may recover
  - The plaintext
  - The key
  - Weakness in a cryptosystem that eventually lead to the results above

# *Cryptanalytic Attacks*

## ● Basic Assumptions

- The secrecy must reside entirely in the key
- The cryptanalyst has complete knowledge of the encryption algorithms used

## ● General types of cryptanalytic attacks

- Ciphertext-only attack
- Known-plaintext attack
- Chosen-plaintext attack
- Adaptive-chosen-plaintext attack
- Chosen-ciphertext attack
- Rubber-hose cryptanalysis

# *Several Advises*

- ~~The strength of your new cryptosystem relies on the fact that the attacker does not know the algorithm's inner workings~~
- The best algorithms are the ones that have been made public, have been attacked by the world's best cryptographers for years, and are still unbreakable



# *Cryptographic Protocols*



# *Protocols*

## ● What is a protocol?

- Definition
  - A series of steps
    - A protocol has a **sequence**, from start to end
  - Two or more parties are involved
    - At least **two** persons are required to complete the protocol
  - Designed to accomplish a task
    - Otherwise, it's a **waste of time**



# ***Characteristics of Protocols***

- What is a protocol?

- Characteristics

- Everyone involved must know the protocol **in advance**
    - Everyone involved must **agree to follow it**
    - The protocol must be **unambiguous**
    - The protocol must be **complete**

- Steps in a protocol

- Computations by one or more of the parties
    - Messages sent among the parties

# **Cryptographic Protocols**

## • What is a cryptographic protocol?

- A protocol that uses cryptography
  - To prevent or detect eavesdropping or cheating
  - Involved parties can be friends and trust each other or they can be adversaries and not trust one another to give the correct time of day
  - It should not be possible to do more or learn more than what is specified in the protocol



# *Purposes of Protocols*

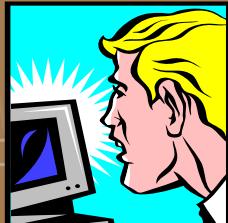
- Why do we need cryptographic protocols?
  - More and more **human interaction** takes place **over computer network** instead of face-to-face
    - It is naïve to assume that people on computer networks are honest
    - It is naïve to assume that managers of computer networks are honest
    - It is naïve to assume that designer of computer networks are honest
  - By formalizing protocols, subversion can be avoided.

# *Notations of Players*

- Alice, Bob, Carol, Dave
  - Participants in protocols
- Eve
  - Eavesdropper
- Mallory
  - Malicious active attacker
- Trent
  - Trusted arbitrator



Alice



Bob



Eve

- Walter
  - Warden
- Peggy
  - Prover
- Victor
  - Verifier



Mallory



Trent

# *Arbitrated Protocol*

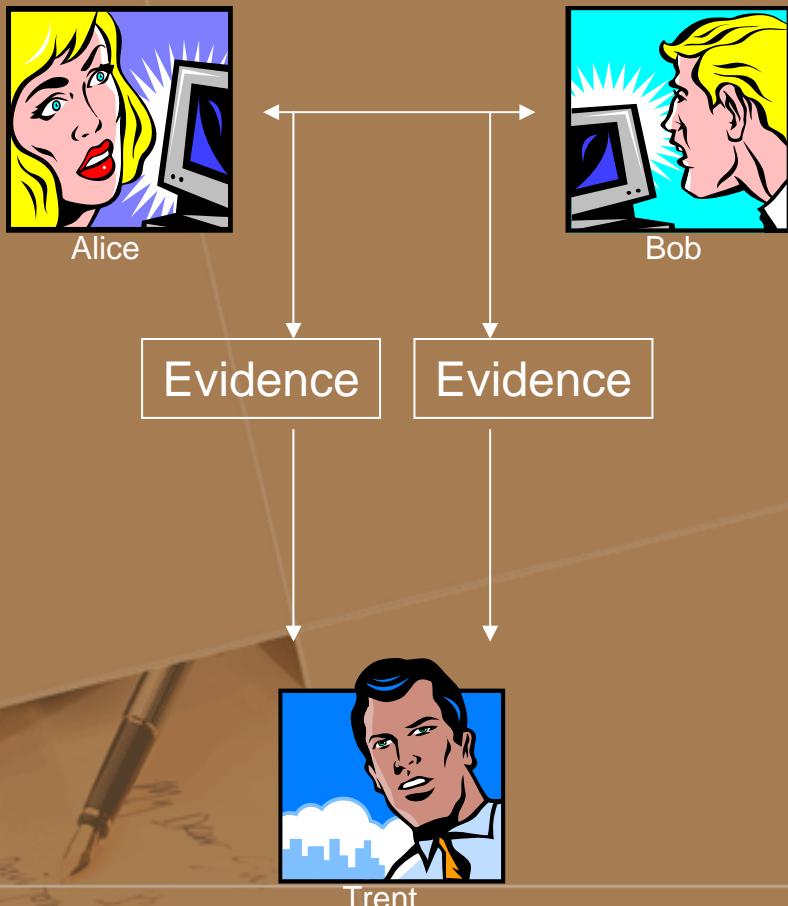


- An arbitrator is a **disinterested** third party **trusted** to complete a protocol
- Arbitrators can help complete protocols between two **mutually distrustful** parties
- An example of an arbitrated protocol in daily life (lawyers or banks as arbitrators )
  - Alice is selling a car to Bob, and Bob would like to pay by check
    1. Alice gives the title of the car to the lawyer
    2. Bob gives the check to Alice
    3. Alice deposits the check
    4. After the check clears, the lawyer gives the title to Bob. If the check is not clear after a specific period of time, the lawyer returns the title to Alice

# ***Problems about Arbitrators***

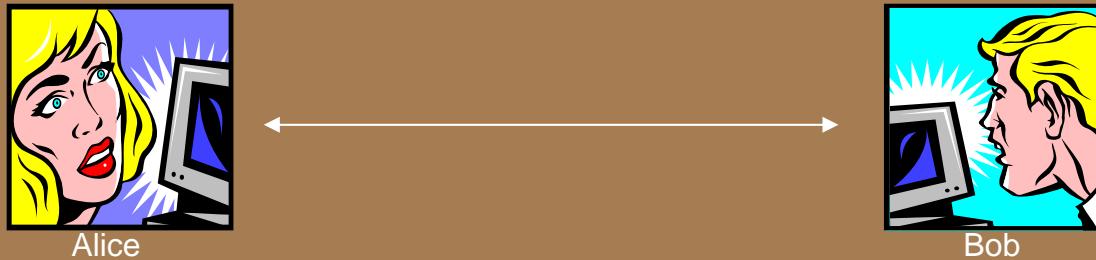
- The ideal of arbitrated protocol can be translated to the computer world, however, there are several problems about computer arbitrators:
  - Difficult to find and **trust** a neutral third party since you do not know who the party is
  - Who will bear the **cost** of maintaining an arbitrator?
  - The **inherent delay** caused by arbitrators
  - The arbitrators will be a **bottleneck** in large scale implementations of any protocol
  - Arbitrators will be a **vulnerable** point for attackers

# *Adjudicated Protocol*



- Due to the cost of hiring arbitrators, an arbitrated protocol can be divided into two low-level sub-protocols
  - non-arbitrated sub-protocols executed every time
  - adjudicated sub-protocol executed only in exceptional circumstance to check whether a protocol was performed fairly
- Judges are professional adjudicators
- An example
  - Contract signing

# *Self-Enforcing Protocols*



- A self-enforcing protocol is the **best** type of protocol.
- The protocol itself guarantees fairness
- No arbitrator is required to complete the protocol
- No adjudicator is required to resolve disputes
- If one of the parties tries to cheat, the other party immediately detects the cheating and the protocol stops.

Unfortunately, there is not a self-enforcing protocol for every situation

# ***Attack against Protocols***



## Targets of cryptographic attacks

- Cryptographic algorithms used in protocols
- Cryptographic techniques used to implement the algorithms and protocols
- **Protocols themselves**



## Passive attacks

- Eavesdropping
- Observe the protocol and attempt to gain information
- The attacker does not affect the protocol

# ***Attack against Protocols (cont.)***

## ● Active attacks

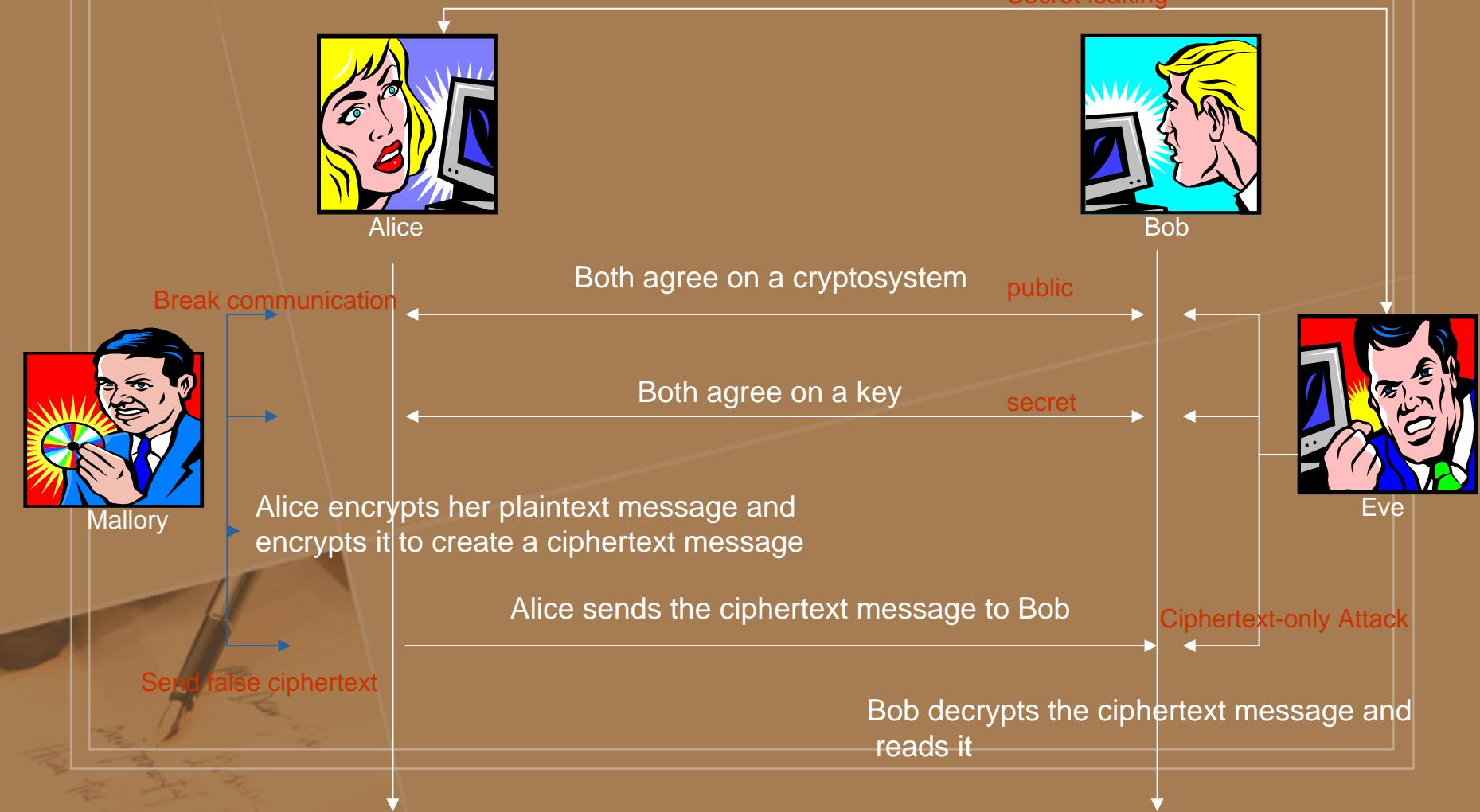
- An attacker could try to alter the protocol to his own advantage
- Active intervention is involved
  - E.g.
    - Pretend to be someone else
    - Introduce/delete/substitute messages in the protocol
    - Interrupt a communication channel

## ● Cheaters

- Attackers who are one of the parties involved in the protocol

# Communication Using Symmetric Cryptography

Secret leaking



# *Problems of Symmetric Cryptosystems*

- Keys must be kept secret before/during/after the protocol
  - Knowing the key → Knowing all the messages
- A separate key is used for each pair in the network, the total number of keys increases rapidly as the number of users increases
  - $N(N-1)/2$  keys for N users



# **One-Way Functions**

- The concept of a one-way functions is central to public-key cryptography
- One-way functions are relatively easy to compute, but significantly harder to reverse (maybe 1,000,000,000,000 years of computations)
  - Given  $x$ , we can easily compute  $f(x)$
  - Given  $f(x)$ , it's hard to compute  $x$

- Strictly speaking, there is no evidence that true one-way function really exists
- True one way function cannot be adopted in cryptography since messages encrypted with one way function is impossible to be decrypted

# **Trapdoor One-way Function**

- A trapdoor one-way function is a special type of one-way function with a secret trapdoor
  - It is easy to compute in one direction, and hard to compute in the other direction
  - But if you know the secret, you can easily compute the function in the other direction

# One-Way Hash Function

- One-way hash function
  - Compression function
  - Contraction function
  - Message digest
  - Fingerprint
  - Cryptographic checksum
  - Message integrity check (MIC)
  - Manipulation detection code (MDC)
- A hash function is a function that takes a variable-length input string (**pre-image**) and converts it to a fixed-length (usually smaller) string (called a **hash value**),
- The **hash value** **fingerprint**s the pre-image, that is, it can indicate that if a pre-image candidate is the same as the pre-image .
- A one-way hash function is a hash function that works for one direction
- Useful in transactions/login sessions



# ***Message Authentication Code***

## ● Message Authentication Code (MAC)

- The hash value is a function of both the pre-image and the key
- Only the one who has the key can verify the hash value

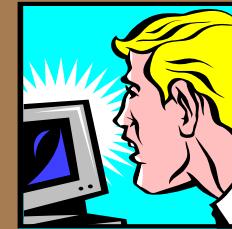


# *Communications Using Public-Key Cryptography*

Database



Alice



Bob

Both agree on a cryptosystem

Bob sends Alice his public key

Alice encrypts her message  
using Bob's public key

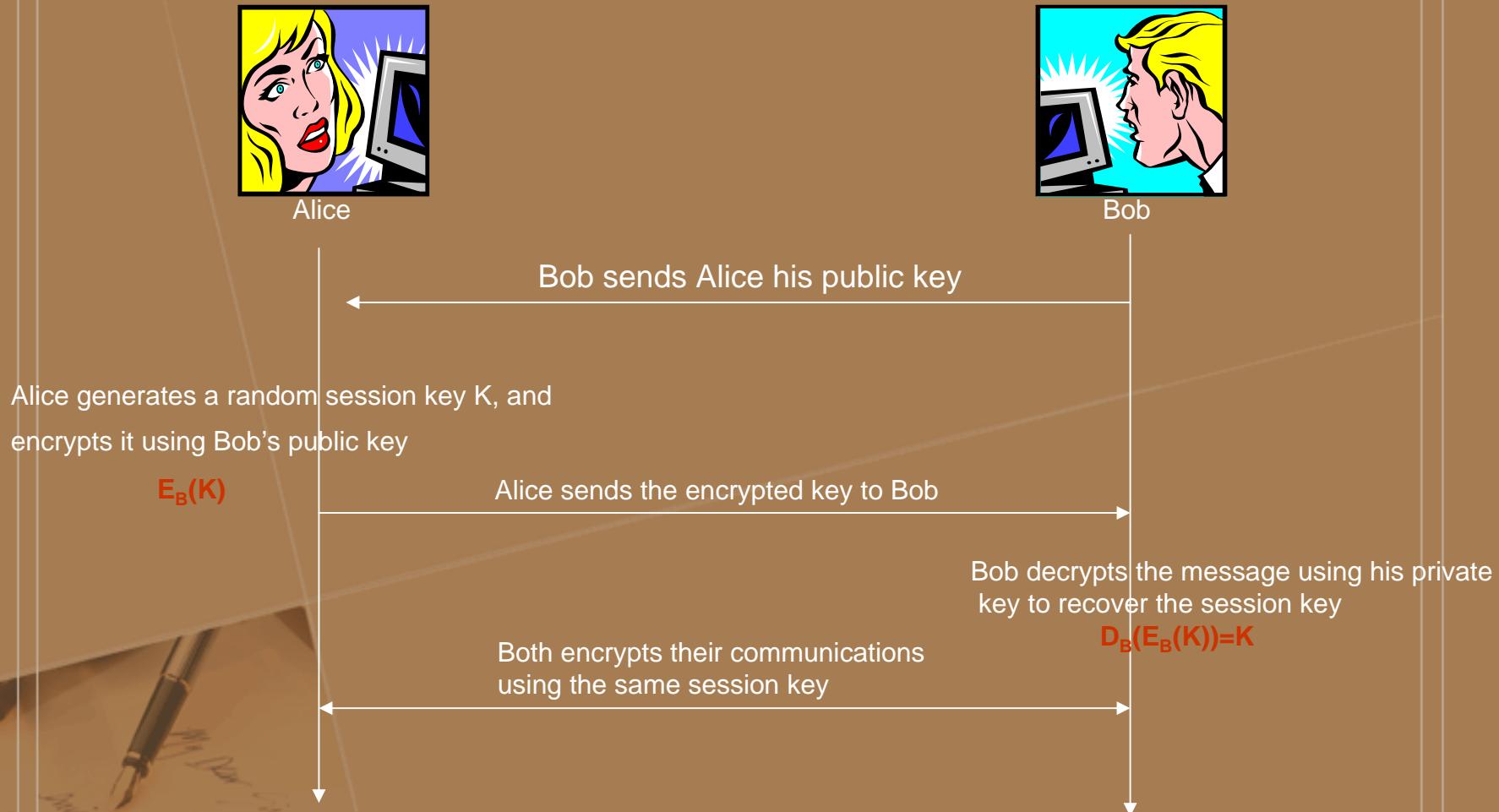
Alice sends the encrypted message to Bob

Bob decrypts the message  
using his private key

# ***Real-World Considerations***

- In the real world, public key algorithms are not substitute for symmetric algorithms because
  - Public-key algorithms are slow
    - Computing power
    - Bandwidth
    - New needs emerge
  - Public-key cryptosystems are vulnerable to chosen-plaintext attack
    - Assume  $C=E(P)$ , encrypt all possible  $P$  and compare to  $C$  can determine  $P$ , thus this type of attack is effective for few possible encrypted messages

# Hybrid Cryptosystems



Key management problem is solved.

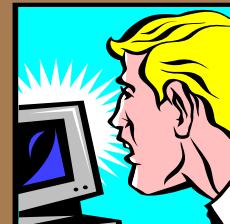
# Man-in-the-Middle Attack



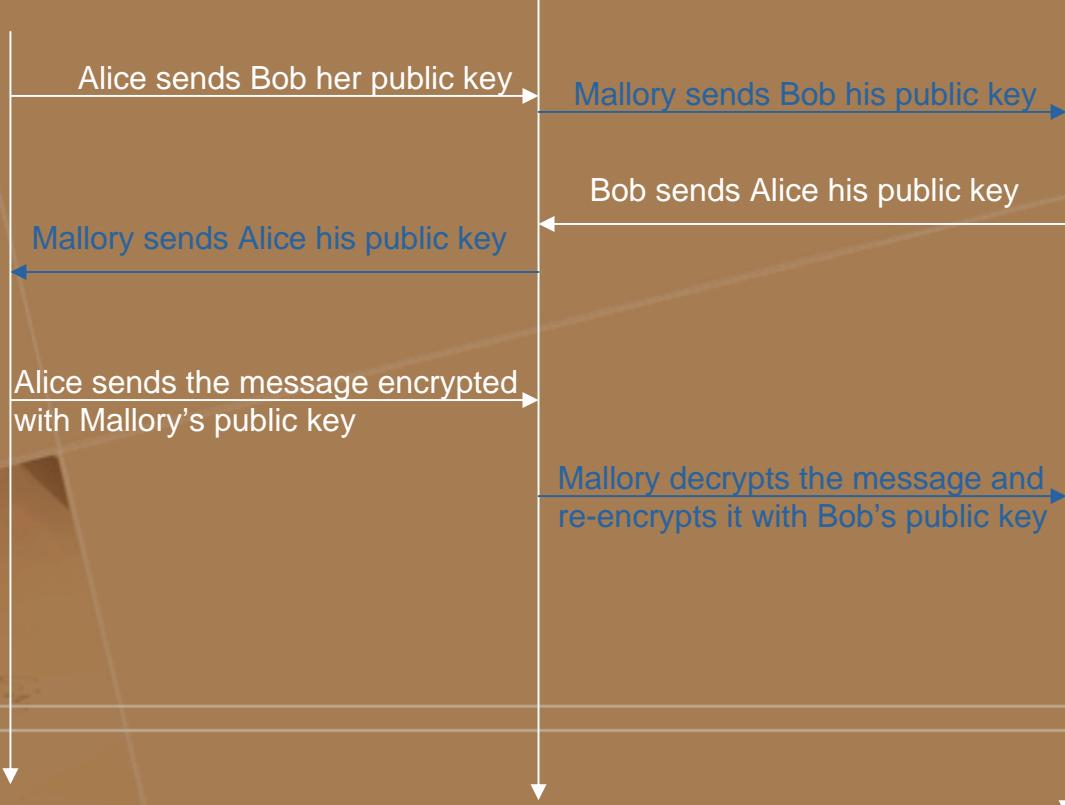
Alice



Mallory



Bob



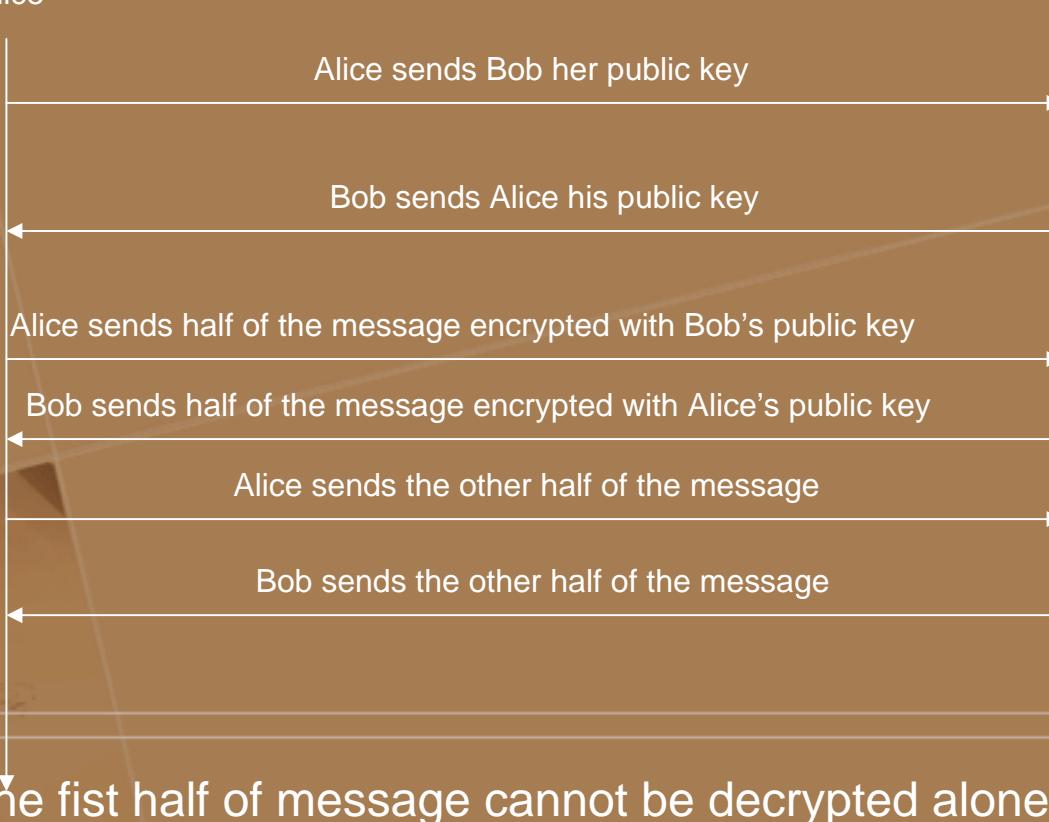
# *The Interlock Protocol*



Alice



Bob



# *Digital Signature*

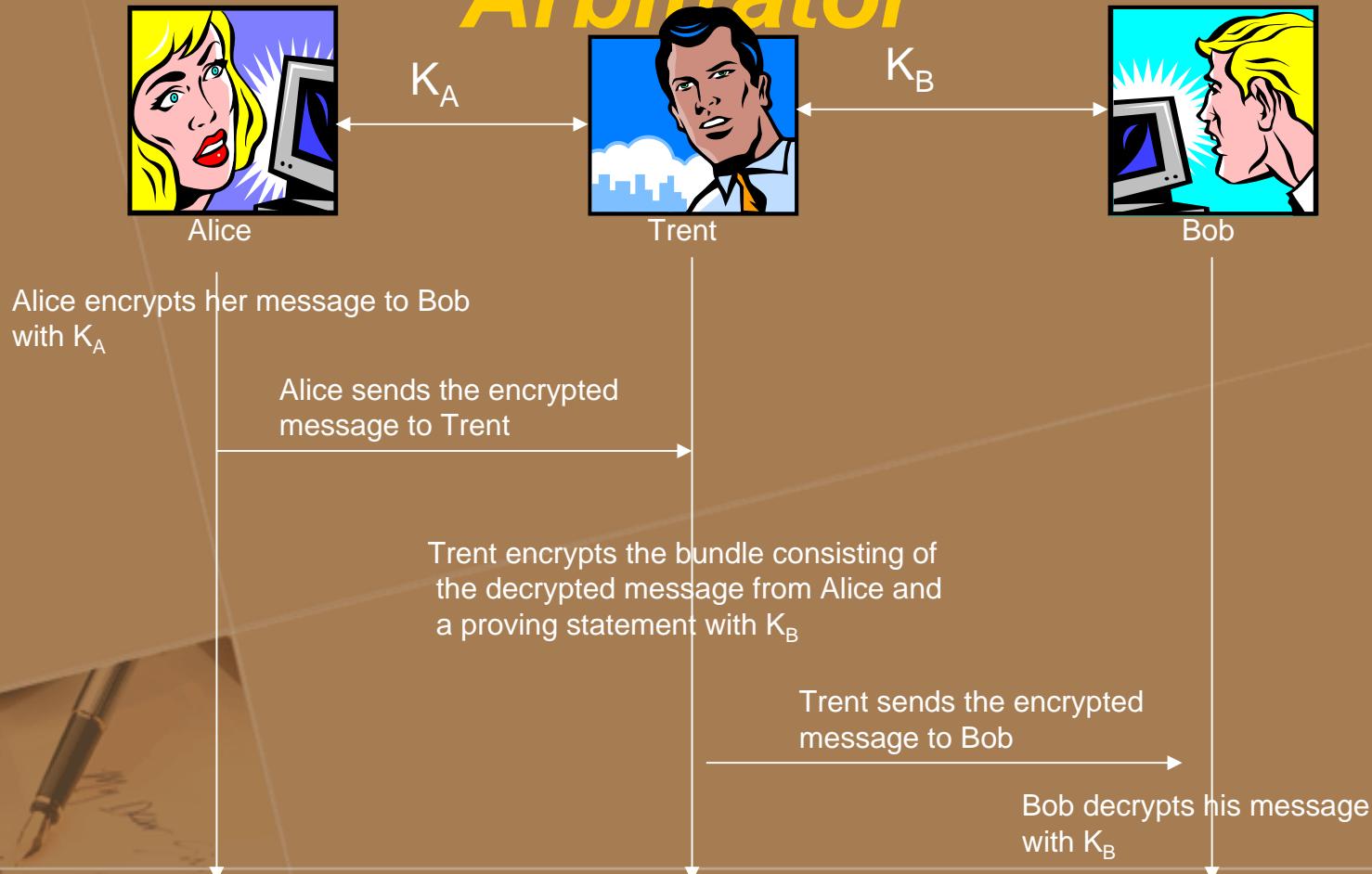
## ● Characteristics of a Signature

- The signature is authentic
- The signature is unforgeable
- The signature is non-repudiable
- The signature is unalterable
- The signature cannot be repudiated

## ● Problems of Signatures in the Computer World

- Files are trivial to copy
- Files are easy to modify after they are signed

# ***Signing with Symmetric Cryptosystems and an Arbitrator***

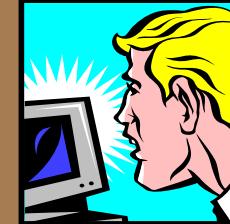


# ***Signing Documents with Public-Key Cryptography***



Alice

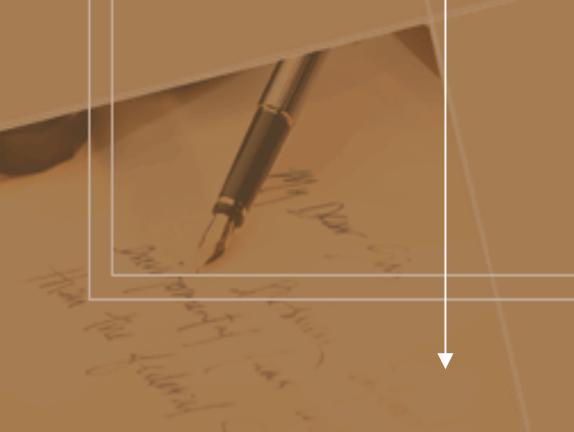
Alice encrypts her message  
with her private key



Bob

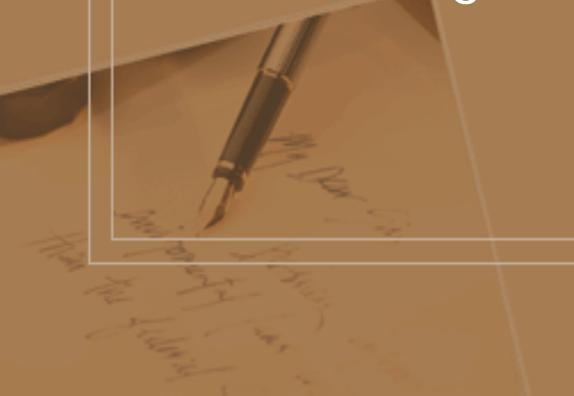
Alice sends the encrypted  
message to Bob

Bob decrypts his message  
with Alice's public key



# *Digital Signature and Timestamps*

- The document and signature may be reused
- Consequently, digital signatures often include timestamps.
  - The date and time of signature are attached to the message and signed along with the rest of the message



# *Digital Signature and One-Way Hash Functions*

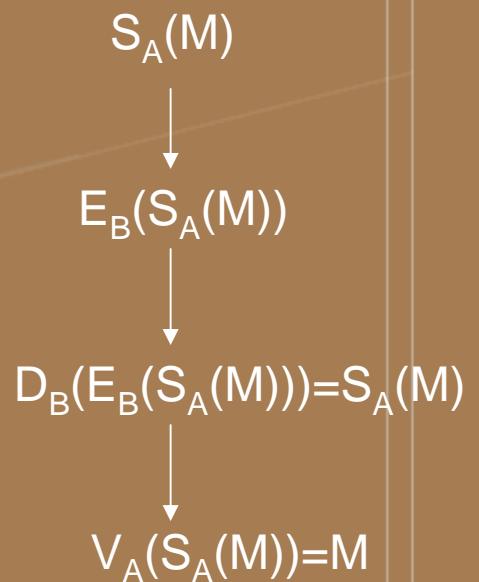
- In practical implementations, public-key algorithms are often too inefficient to sign long documents
  - Instead of signing the document, the hash value of the document is signed and transmitted.



# *Digital Signature and Encryption*

- Combining the security of encryption with the authenticity of digital signature

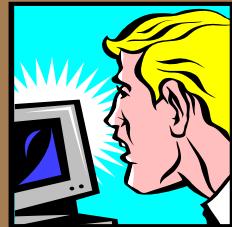
- Alice signs the message with her private key
- Alice encrypts the signed message with Bob's public key and sends it to Bob
- Bob decrypts the message with his private key
- Bob verifies with Alice's public key and recovers the message



# *Resending the Message as a Receipt*



Alice

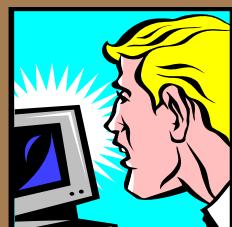


Bob

 $E_B(S_A(M))$  $V_A D_B(E_B(S_A(M))) = M$  $V_B(D_A(E_A(S_B(M)))) = M$  $E_A(S_B(M))$ 

Mallory

If  $V_x = E_x$ ,  $S_x = D_x$



Bob

 $E_B(D_A(M))$  $E_M D_B(E_B(D_A(M))) = E_M(D_A(M))$  $E_A(D_M(E_B(D_M(E_M(D_B(E_M(D_A(M)))))))) = M$  $E_M(D_B(E_M(D_A(M))))$