MONASH University
Information Technology

FIT2093 INTRODUCTION TO CYBER SECURITY

MONASH University
Information Technology

FIT2093 INTRODUCTION TO CYBER SECURITY

Lecture 7:
Public Key Cryptography

Unit Structure

- **Introduction to security of**
- **Authentication**
- **Access Control Fundamental**
- **Fundamental concepts of cryptography**
- **Symmetric encryption techniques**
- **Introduction to number theory**
- **Public key cryptography**
- **Integrity management**
- **Practical aspects of cyber security**
- **Hacking and countermeasures**
- **Database security**
- **IT risk management & Ethics and privacy**

Previous Lecture

- **Prime number**
- **Coprime or relative prime**
- **Understand the idea of factorisation.**
- **Modular arithmetic / clock arithmetic**
- **Modular arithmetic properties.**
- **Eulers Totient function**
- **Use of modulo arithmetic properties in cryptography**

Modular Arithmetic in Cryptography

- y & n are relatively prime integers and $y \pmod{ø(n)} = 1$, for any M < n, then

  $M^y \bmod n = M$

- Let y = a *b, then

  $M^y \bmod n = M$

  $= M^{a*b} \bmod n$

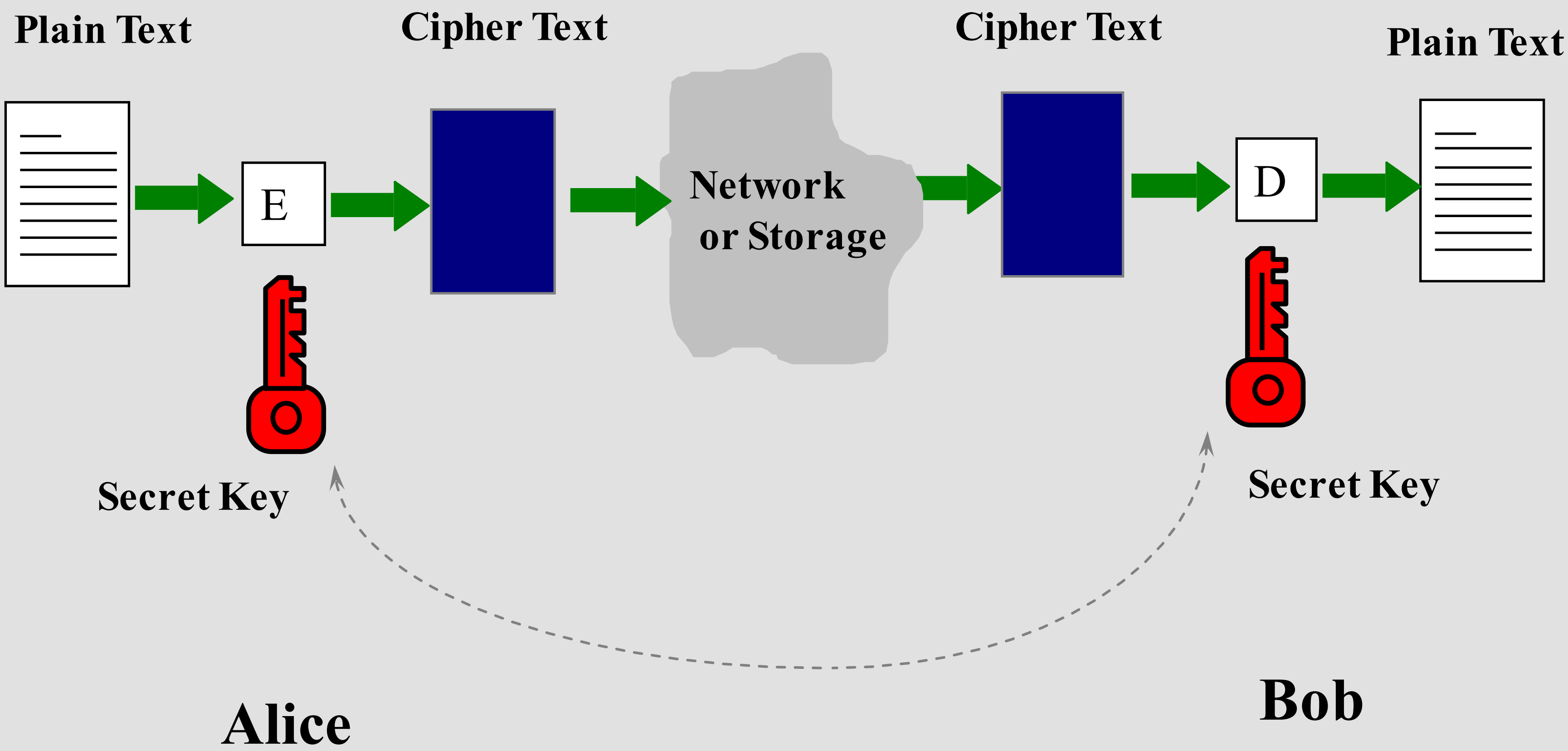
  $= M$ if $a * b \bmod ø(n) = 1$

- a & b are multiplicative inverse in modulo $ø(n)$ arithmetic.
- If message/data M need to be protected, transform M as $M^a \bmod n$ and store it and to get it back, do $(M^a)^b = M \bmod n$
- Not knowing b, it will be difficult to get M from M^a.

Outline

- **Why public key cryptography ?**
- **General principles of public key cryptography**
- **Diffie-Hellman key exchange**
- **The RSA public key cryptosystem**

Private (Symmetric) key cipher



Attacking Symmetric Encryption

- **cryptanalysis**
  - rely on nature of the algorithm
  - plus some knowledge of plaintext characteristics
  - even some sample plaintext-ciphertext pairs
  - exploits characteristics of algorithm to deduce specific plaintext or key
- **brute-force attack**
  - try all possible keys on some ciphertext until get an intelligible translation into plaintext

Problems with private key ciphers

Data (Advanced) Encryption Standard

- **In order for Alice & Bob to be able to communicate securely using a private key cipher, such as DES/AES, they have to have a shared key in the first place.**
  - Question:
    What if they have never met before ?
- **Alice needs to keep *100* different keys if she wishes to communicate with *100* different people**

Motivation of Public Key Cryptography

- **Is it possible for Alice & Bob, who have no shared secret key, to communicate securely ?**
- **This led to the SINGLE MOST IMPORTANT discovery of public key communications:**
  - Diffie & Hellman's ideas of public key cryptography: <private-key, public-key>

Main ideas

- **Bob:**
  – publishes, say in Yellow/White pages, his
    > public (for encryption) key, and
    > encryption algorithm.
  – keeps to himself
    > the matching secret (for decryption) key – also called private key (do not confuse with symmetric key cipher).

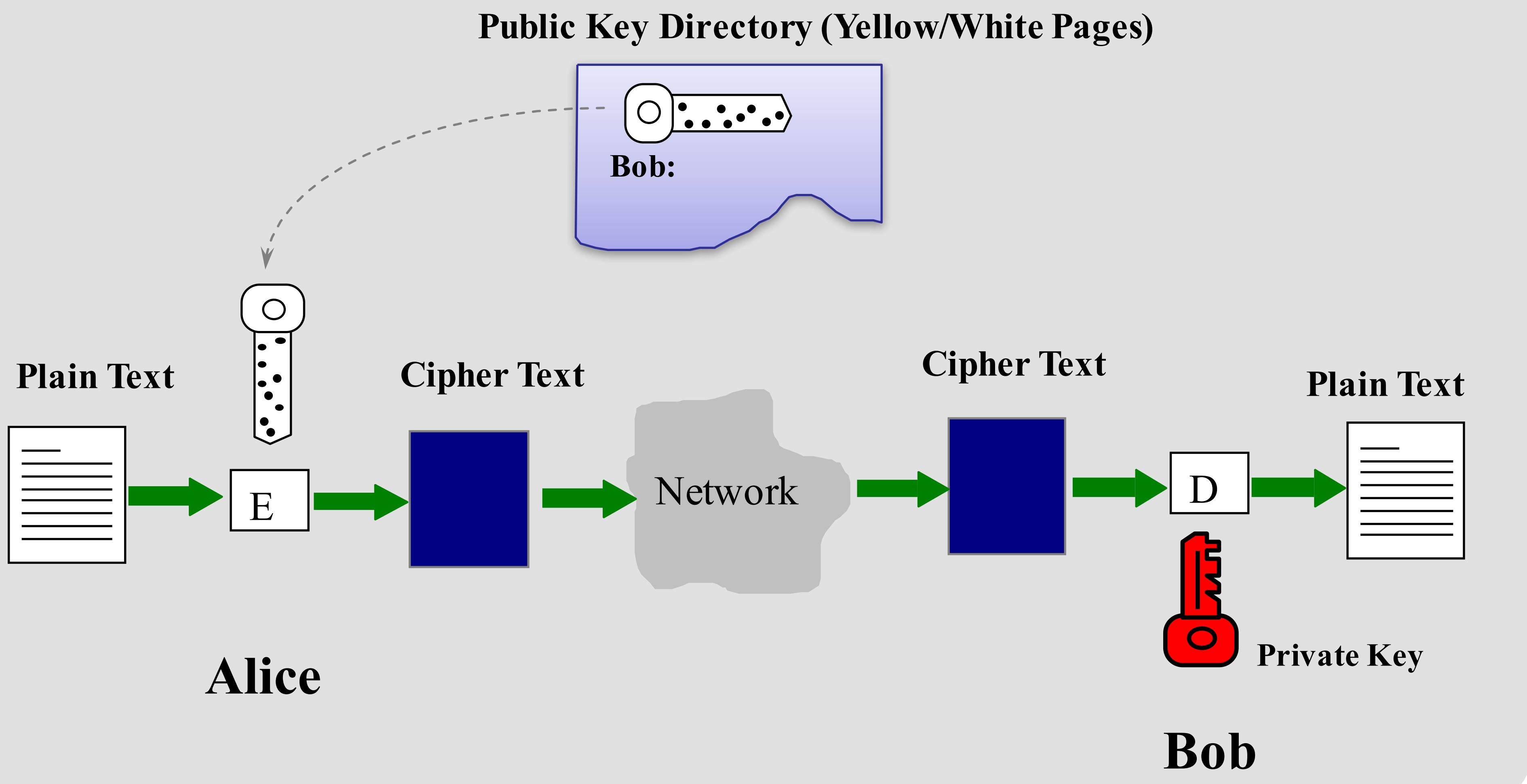
Main ideas (2)

- **Alice:**
  - Looks up the phone book, and finds out Bob's
    - > public key, and
    - > encryption algorithm.
  - Encrypts a message using Bob's public key and encryption algorithm.
  - sends the ciphertext to Bob.

Main ideas (3)

- **Bob:**
  - Receives the ciphertext from Alice
  - Decrypts the ciphertext using his secret/private key, together with the decryption algorithm

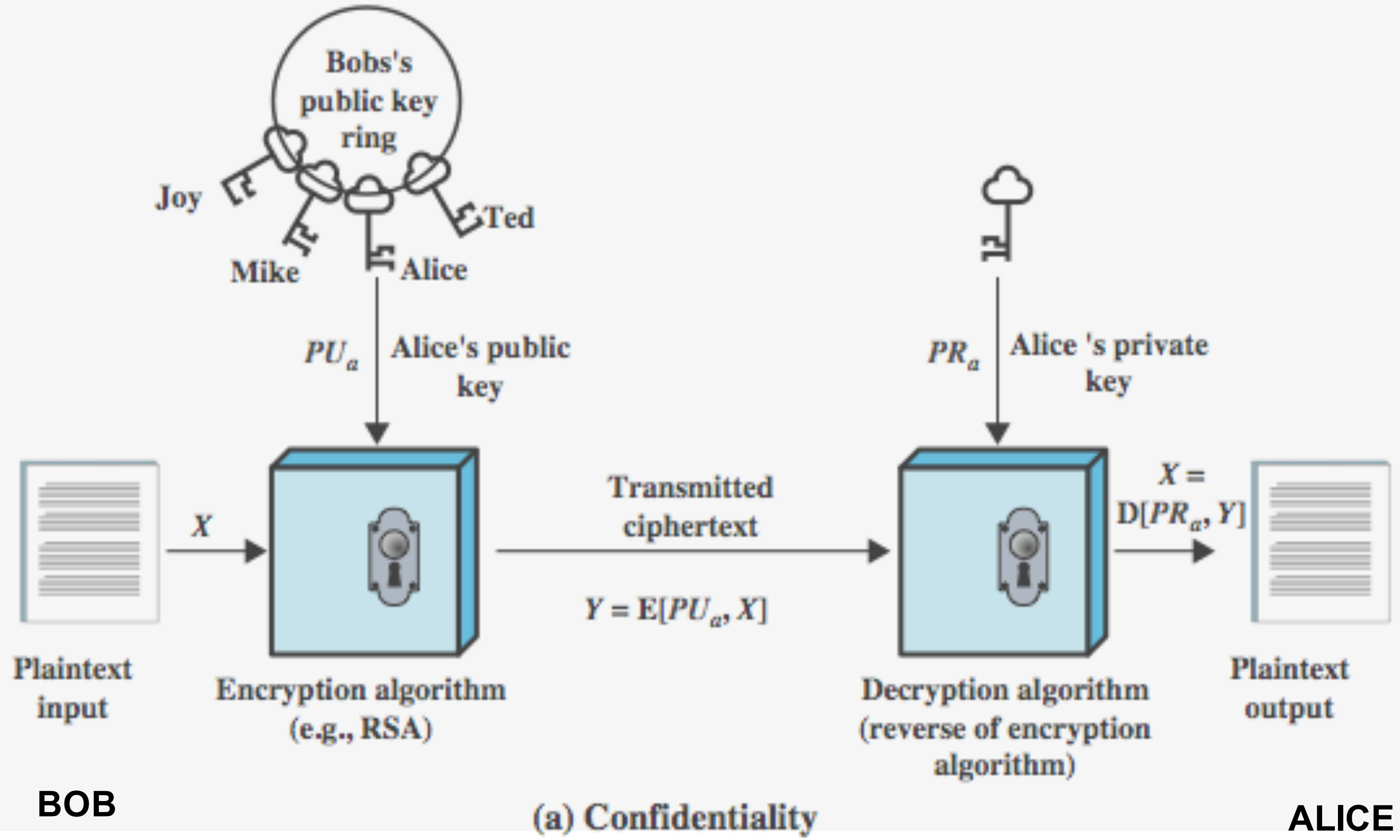
Public Key Cryptosystem



Main differences with DES

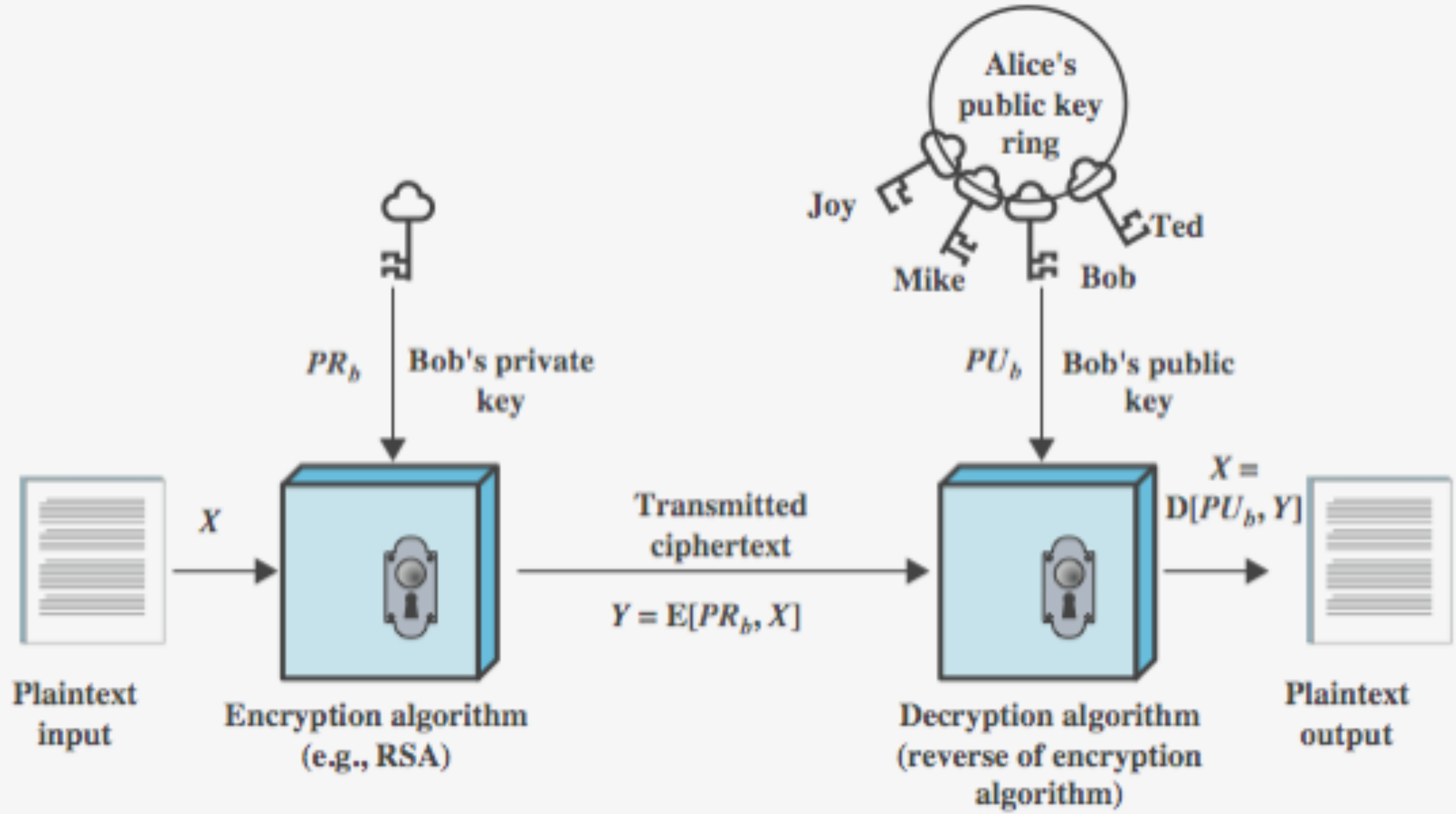
- **The public key is different from the secret or private key.**
- **Infeasible for an attacker to find out the secret key from the public key.**
- **No need for Alice & Bob to distribute a shared secret key beforehand !**
- **Only one pair of public and secret keys is required for each user !**

Public Key Encryption



(a) Confidentiality

Public Key Authentication



(b) Authentication

Public Key Requirements

1. **computationally easy to create key pairs**
2. **computationally easy for sender knowing public key to encrypt messages**
3. **computationally easy for receiver knowing private key to decrypt ciphertext**
4. **computationally infeasible for opponent to determine private key from public key**
5. **computationally infeasible for opponent to otherwise recover original message**
6. **useful if either key can be used for each role**

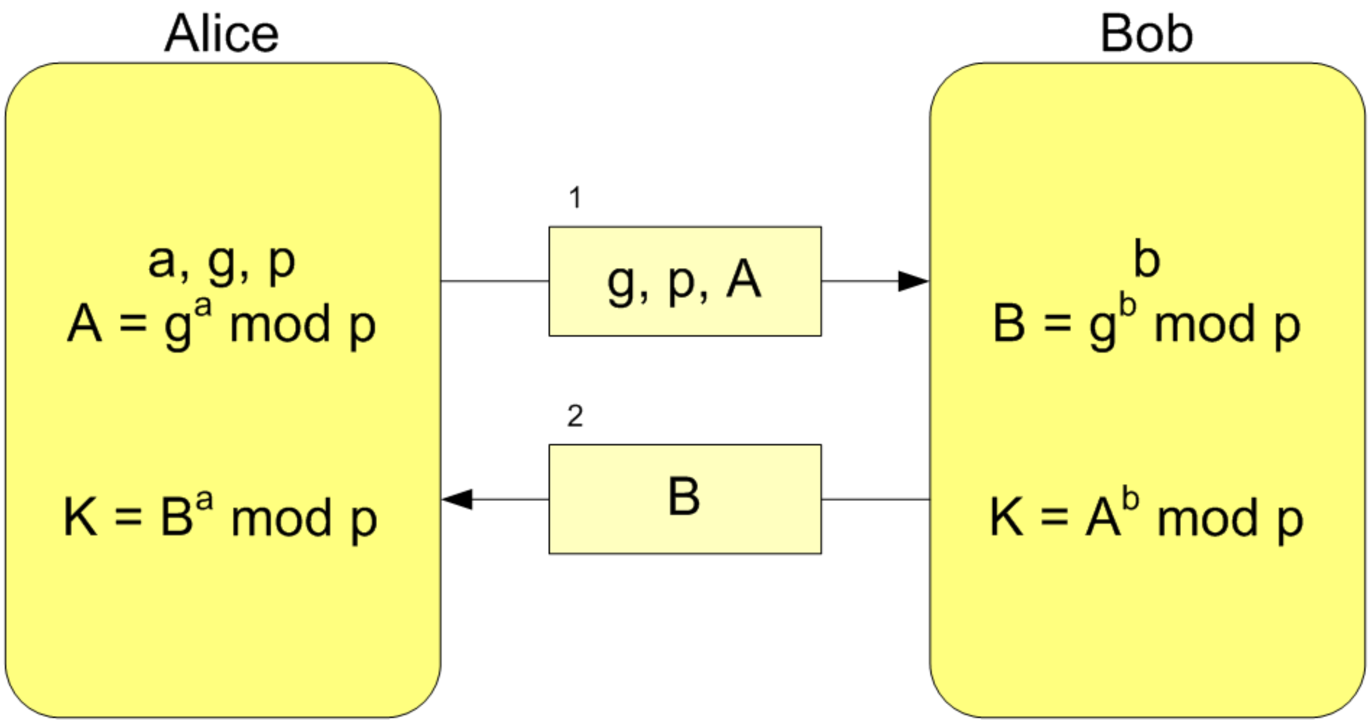
Public Key Algorithms

- **RSA (Rivest, Shamir, Adleman)**
  – developed in 1977
  – only widely accepted public-key encryption alg
  – given tech advances need 1024+ bit keys
- **Diffie-Hellman key exchange algorithm (1976)**
  – only allows exchange of a secret key
- **Digital Signature Standard (DSS) (1991)**
  – provides only a digital signature function with SHA-1
- **Elliptic curve cryptography (ECC) (1985)**
  – new, security like RSA, but with much smaller keys

Diffie-Hellman Key Exchange

- **first public-key type scheme proposed**
- **by Diffie & Hellman in 1976 along with the exposition of public key concepts**
  - note: now know that Williamson (UK CESG) secretly proposed the concept in 1970
- **practical method to exchange a secret key**
- **used in a number of commercial products**
- **security relies on difficulty of computing discrete logarithms**

Diffie-Hellman Key Exchange – Main Idea



$$K = A^b \bmod p = (g^a \bmod p)^b \bmod p = g^{ab} \bmod p = (g^b \bmod p)^a \bmod p = B^a \bmod p$$

Wikipedia.org

- **g,p are known to both Alice and Bob. $g<p$**
- **p is a prime number.**
- **a is Alice's secret number, $a<p$**
- **b is Bob's secret number, $b<p$**

Diffie-Hellman Key Exchange

- **Answer to the problem of key distribution in private key.**
  - The shared-key can be generated by using the combination of the non-secret values of p and g.
  - The shared-key can only be generated by knowing either the value of A (Alice's public key) and b (Bob's private Key) or B and a.

Diffie-Hellman Example

- **Alice and Bob agree to use a prime number p=23 and base g=5.**
- **Alice chooses a secret integer a=6, then sends Bob (g^a mod p)**
  - A=5^6 mod 23 = 5^2X5^2X5^2 mod 23= 2X2X2 mod 23 = 8.
- **Bob chooses a secret integer b=15, then sends Alice (g^b mod p)**
  - B=5^{15} mod 23 =$(5^2)^7$ X 5 mod 23 = 2^7X5 mod 23 = 128X5 mod23 =13X5 mod 23 = 65 mod 23 = 19.
- **Alice computes (g^b mod p)^a mod p = B^{a}mod p**
  - K=19^6 mod 23 = $(19^2)^3$ mod 23=$(361)^3$mod23 =16^3mod23= 16^2X16mod23=256 X16 mod23 = 3X16mod23= 2.
- **Bob computes (g^a mod p)^b mod p = A^{b}mod p**
  - K=8^{15} mod 23 =$(8^4)^3$X8^3mod23=2^3X8^3mod23=8^4mod23 = 2.
- **The shared secret key is 2.**

Wikipedia.org

Diffie-Hellman Key Exchange : Example (2)

- **Will Marvin be able to guess the value of the shared key?**
- **Marvin will know:**
  - $5^a \bmod 23 = 8$.
  - $5^b \bmod 23 = 19$
  - $19^a \bmod 23 = s$
  - $8^b \bmod 23 = s$
  - $19^a \bmod 23 = 8^b \bmod 23$
- **Finding the value of an exponent such as *a* or *b* knowing the integer and modulo values are not trivial, => It will take Marvin a long time to calculate either a or b!**
  - See slide 23 on number theory

Diffie-Hellman Key Exchange

- **The model shows that it is possible to create a secret key based on "known" or "published" information.**
- **The introduction of this key exchange protocol triggers the interests in the exploration of modular mathematics and prime number theory in cryptography.**
- **Still have the need to agree on *p* and the *g* before secret key can be generated, i.e reaching an agreement process is still needed.**
- **What can we do about it?**

Realising public key ciphers

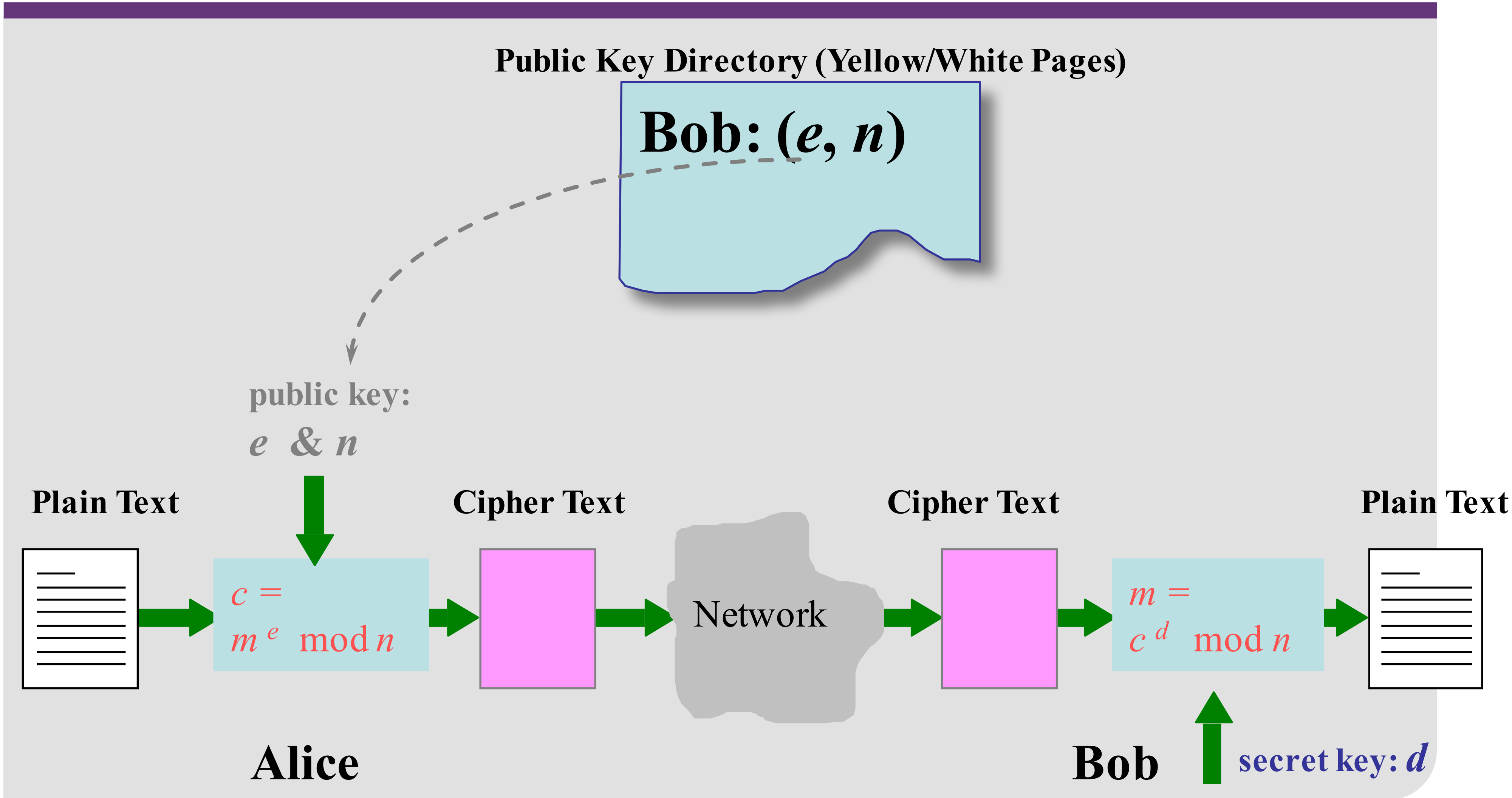
- **The most famous system that implements Diffie & Hellman's ideas on public key ciphers is due to**
  - Ronald Rivest
  - Adi Shamir
  - Leonard Adleman
- **This public key cryptosystem is called RSA.**

1977, at MIT

2003, RSA

MONASH University
Information Technology

RSA Public Key Cryptosystem



Look at this → $M^y \bmod n = M$

- Let y = a *b, and y & n are relatively prime, then For any M < n,

  $$M^y \bmod n = M$$
  $$= M^{a*b} \bmod n$$
  $$= M \text{ if } a^* b \bmod \phi(n) = 1$$

  $$a^* b = 1 \bmod \phi(n)$$

- a & b are multiplicative inverse in modulo $\phi(n)$ arithmetic.

- **Refer to slide 30 from number theory**

RSA Key Setup

- **Bob:**
  - chooses 2 large prime numbers: *p, q*
    multiplies p and q: $n = p*q$
  - finds out two numbers *e* & *d* such that
    - > $(e * d) \bmod ø(n) = 1$
    - *Or* $(e * d) \bmod [(p-1)*(q-1)] = 1$
  - public key (published in the phone book)
    - > 2 numbers: (e, n)
    - > encryption alg: modular exponentiation
  - secret key: (d,n)

RSA : Encryption

- **Alice has a message *m* to be sent to Bob:**
  – finds out Bob's public encryption key (e, n)
  – calculates

    m^e (mod n) -> c

  – sends the ciphertext c to Bob

RSA : Decryption

- **Bob:**
  - receives the ciphertext *c* from Alice
  - uses his matching secret decryption key *d* to calculate

$$c^d \ (mod\ n) \quad \text{->} \ m$$

RSA ---A simple example (1)

- **Bob:**
  - chooses 2 primes: $p=5, q=11$
    multiplies p and q: $n = p*q = 55$
  - $\varnothing(n)$ = (p-1)(q-1)=4X10=40
  - finds out two numbers $e=3$ & $d=27$ which satisfy
    $(3 * 27) \bmod 40 = 1;$
  - Bob's public key
    - > 2 numbers: (*3, 55*)
    - > encryption alg: modular exponentiation
  - secret key: (*27,55*)

RSA --- A simple example (2)

- **Alice has a message *m=13* to be sent to Bob:**
  - finds out Bob's public encryption key (*3, 55*)
  - calculates c:

$$
\begin{aligned}
c &= m^e \ (mod\ n) \\
&= 13^3 \ (mod\ 55) \\
&= (169\ (mod\ 55) * 13 mod 55) mod 55 \\
&= (4*13) mod 55) \\
&= 52
\end{aligned}
$$

  - sends the ciphertext *c=52* to Bob

RSA --- A simple example (3)

- **Bob:**
  - receives the ciphertext *c=52* from Alice
  - uses his matching secret decryption key *27* to calculate m:

$$m = 52^{27} \pmod{55}$$

$$= (4*13)^{27} mod55 = 4^{27}*13^{27} mod55$$

$$= 4^{27}*(13^{2X13})*13mod55$$

$$= 4^{27}*4^{13}*13mod55$$

$$= 4^{40}*13\ mod55 = 1mod55*13mod55$$

[$a^{\phi(n)} = 1 \pmod{n}$], if a & n are relatively prime-Euler's theorem

$$= 13 \text{ (Alice's message)}$$

How does RSA work?

- $n = p*q \Rightarrow Ø(n) = Ø(p*q) = (p-1)*(q-1)$
- **We choose *d* & *e* such that**
  - $(e * d) \bmod ø(n) = 1$;
  - for any $m < n$: $m^{de} = m \bmod n$;
  - **an RSA** encryption consists of taking m and raising it to e; and decrypting the ciphertext by raising the result of the encrytion to d.
    > We have $(a*b) \bmod n = ((a \bmod n) * (b \bmod n)) \bmod n$
    > hence : $(m^e \bmod n)^d \bmod n = (m^e)^d \bmod n = (m^{ed}) \bmod n = m \bmod n = m$

MONASH University
Information Technology

Constraints on RSA

- **The message m has to be an integer between the range [1, (n-1)].**
- **To encrypt long messages we can use modes of operation as for block private key ciphers, or a hybrid cryptosystem.**
- **The values of e & d are large else it will be vulnerable to a brute force attack and to other forms of cryptanalysis**

Why RSA is Secure?

- **Attack Scenario:**
  - Marvin wants to read Alice's private message (m) intended to be read only by Bob.
  - However, Alice used RSA to encrypt m using Bob's public key (e, n), into the ciphertext $\boldsymbol{c} = m^e \pmod{n}$.
  - Marvin is a determined attacker and managed to intercept the ciphertext c on its way from Alice's to Bob's computer.
  - Marvin also looked up Bob's public key (e,n) to help him in his attack.

Why RSA is Secure?

- **Marvin now has (c,e,n) and wants to find out m.**
- **How can Marvin proceed to find m?**
  - Approach 1: If Marvin could also find out Bob's secret key d, he could decrypt c into m in the same way as Bob does.
    - > Suppose Bob guards his secret key d very well, what can Marvin do then?
  - Approach 2: Marvin knows that $\boldsymbol{c} = m^e \pmod{n}$. He knows that m is a number between 0 and n-1. So he could use exhaustive search through all n possible messages m.
    - > But if n is large this takes a long time!

Why RSA is Secure?

- **Marvin's Attack options (cont):**
  - Approach 3: Marvin can try to ***compute*** **Bob's secret key d from (e,n) and then use Approach 1.**
    - > Remember that (e * d) mod ((p-1)*(q-1)) = 1
    - > Marvin found in a 'Number Theory' book a very fast algorithm called *EUCLID* to solve the following problem: Given two numbers (r,s), the algorithm outputs a number x such that

      (r * x) mod s = 1.

Why RSA is Secure?

- **Approach 3 is the most efficient known method Marvin can use to attack RSA!**
- **The time taken for Marvin to execute the attack in Approach 3 is essentially the time to factorize n=p*q into the prime factors p and q.**
- **Therefore, we say that RSA is *based on* the *factorization problem*:**

  While it is easy to multiply large primes together, *it is computationally infeasible to factorize or split a large composite into its prime factors* !

Why RSA is Secure?

- **Therefore, when both p and q in RSA are of at least 155 digits, the product n=p*q is 310 digits.**
- **Then no one can factorize n in less time than a few thousand years, not even Marvin!!**
- **Thus the *only* person who can extract the plaintext m from the ciphertext c is Bob, as only he knows the secret decryption key d !**

Marvin's New Attack Idea

- **Instead of just eavesdropping, Marvin can try a more *active* attack!**
- **Outline of the New Attack:**
  – Marvin generates an RSA key pair
    > Public key = Kpub_* = (N_*, e_*)
    > Secret key = Ksec_* = d_*
  – Marvin sends the following email to Alice, pretending to be Bob:
    > Hi Alice,
      – Please use my new public key from now on to encrypt messages to me. My new public key is Kpub_*.
      – Yours sincerely, Bob.
  – Marvin decrypts any messages Alice sends to Bob (encrypted with Kpub_*), using Ksec_*.

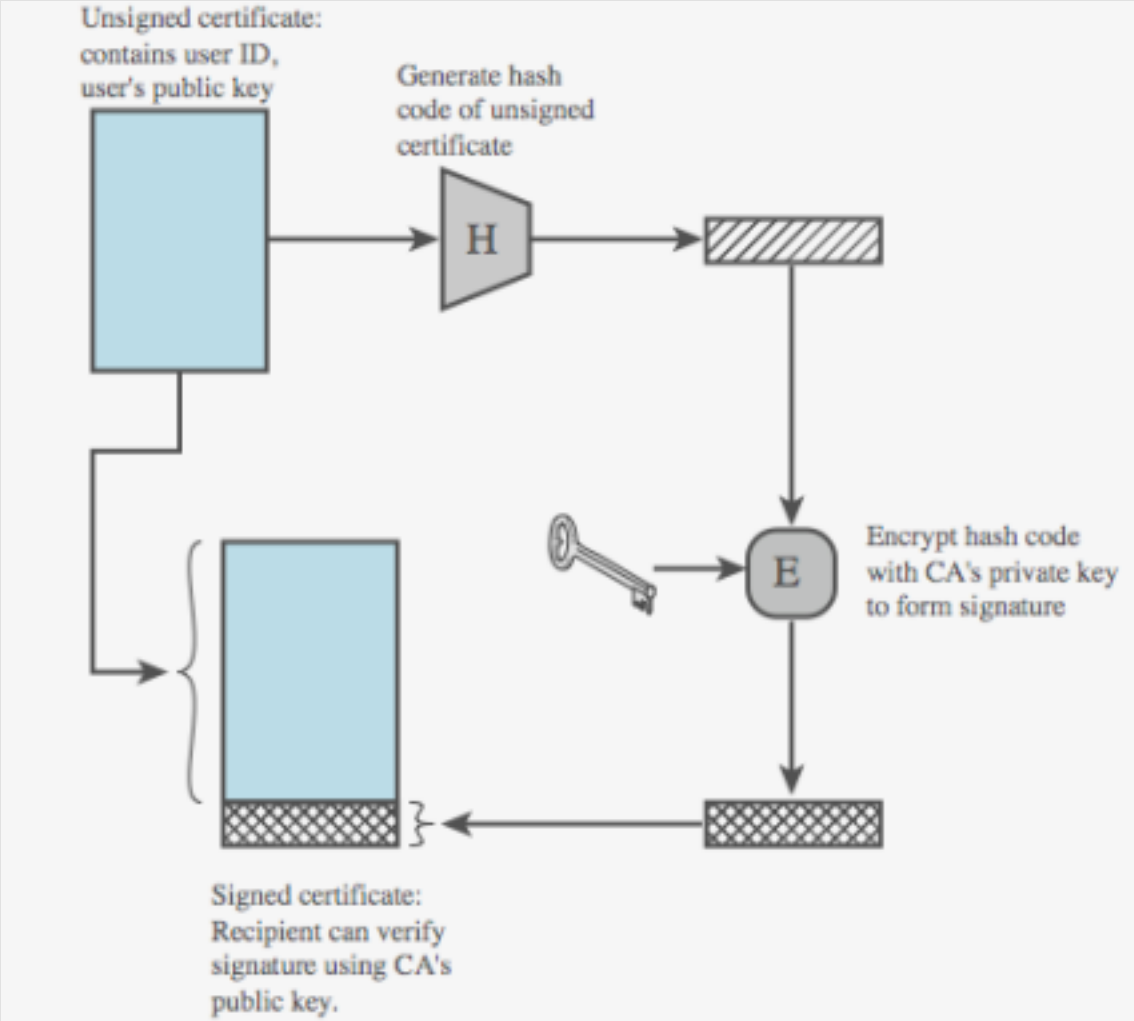
Preventing Marvin's Active Attack

- **The active attack works because:**
  - Alice was tricked by Marvin into encrypting a message intended for Bob using a "fake" public key which is NOT Bob's public key (in fact it was Marvin's).
- **To prevent the attack:**
  - Before Alice encrypts a message for Bob, she must make sure she has Bob's CORRECT public key (and not a fake one).
  - Alice needs a way of testing the truth of any "Bob's key message" informing Alice of Bob's Public Key.
  - No one besides Bob should be able to produce such a message so that it will pass Alice's Test.

Preventing Marvin’s Active Attack (2)

- **This is a setting where Alice and Bob have a <u>message integrity</u> security requirement!**
  - i.e. Alice and Bob want to prevent fabrication and/or modification of a “Bob’s key message” (a message informing Alice of Bob’s public key) by unauthorised parties (like Marvin).
- **The main cryptographic tool used to achieve message integrity is “Authority Certificates”.**
- **Later we shall see how Digital Signatures can be used to prevent Marvin’s Attack!**

Public Key Certificates



Private key ciphers

- **Strengths**
  - in-expensive to use
  - fast
  - low cost VLSI chips available
- **Weakness**
  - key distribution is a problem
  - If there are *n* people who need to communicate, then *n(n-1)/2* keys are needed for each pair of people to communicate privately.

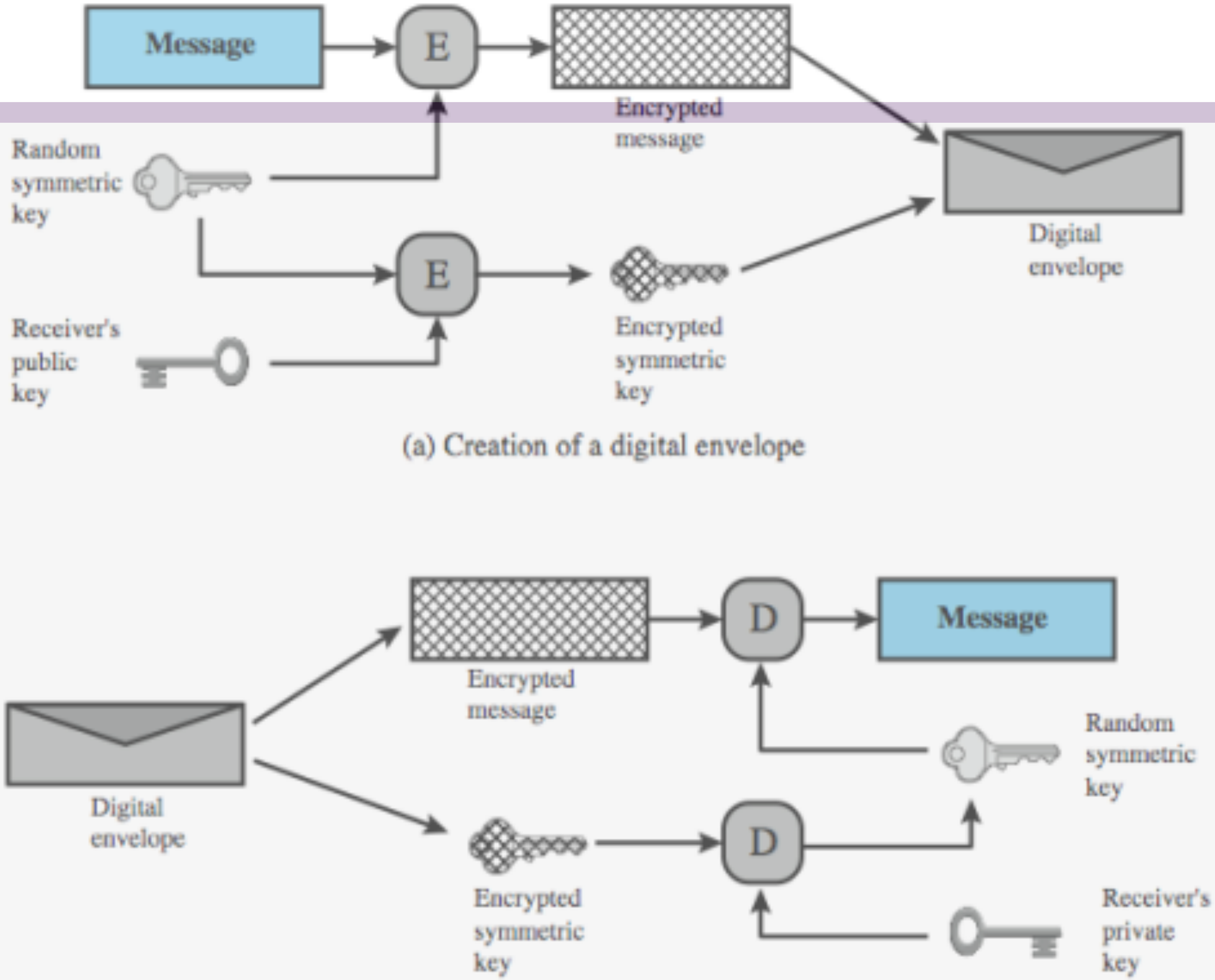
Public key ciphers

- **Strengths**
  - key distribution is NOT a problem
  - Only 2 keys required for communicating with *n* users
- **Weaknesses**
  - relatively expensive to use
  - relatively slow
  - VLSI chips not available or relatively high cost

Combining 2 Type of Ciphers

- **In practice, we can**
  - use a public key cipher (such as RSA) to distribute key for symmetric key cipher
  - use a symmetric key cipher (such as AES) to encrypt and decrypt messages

Digital Envelopes



(a) Creation of a digital envelope

(b) Opening a digital envelope

Summary

- **Why public key cryptography ?**
- **General principles of public key cryptography**
- **Diffie-Hellman public key exchange**
- **RSA public key cryptosystem**
- **RSA Security**

Further Reading

- **Chapters 2 & 21 of the textbook: *Computer Security: Principles and Practice" by William Stallings & Lawrie Brown,* Prentice Hall, 2015**

- **Acknowledgement: part of the materials presented in the slides was developed with the help of Instructor's Manual and other resources made available by the author of the textbook.**