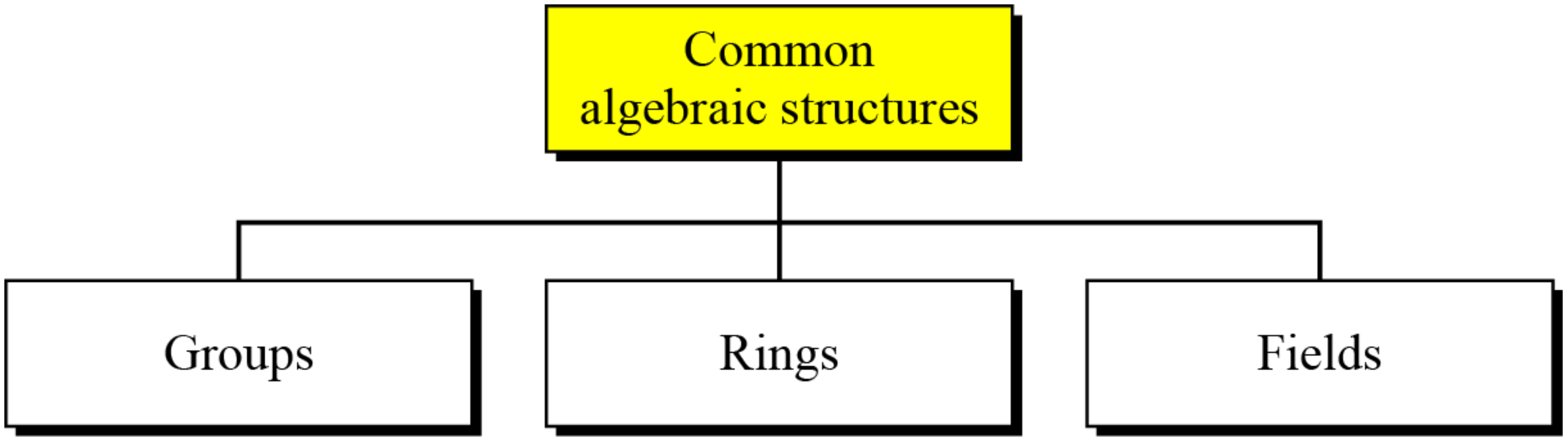
Algebraic Structures

- Cryptography requires sets of integers and specific operations that are defined for those sets.
- The combination of the set and the operations that are applied to the elements of the set is called an algebraic structure.

Common algebraic structure

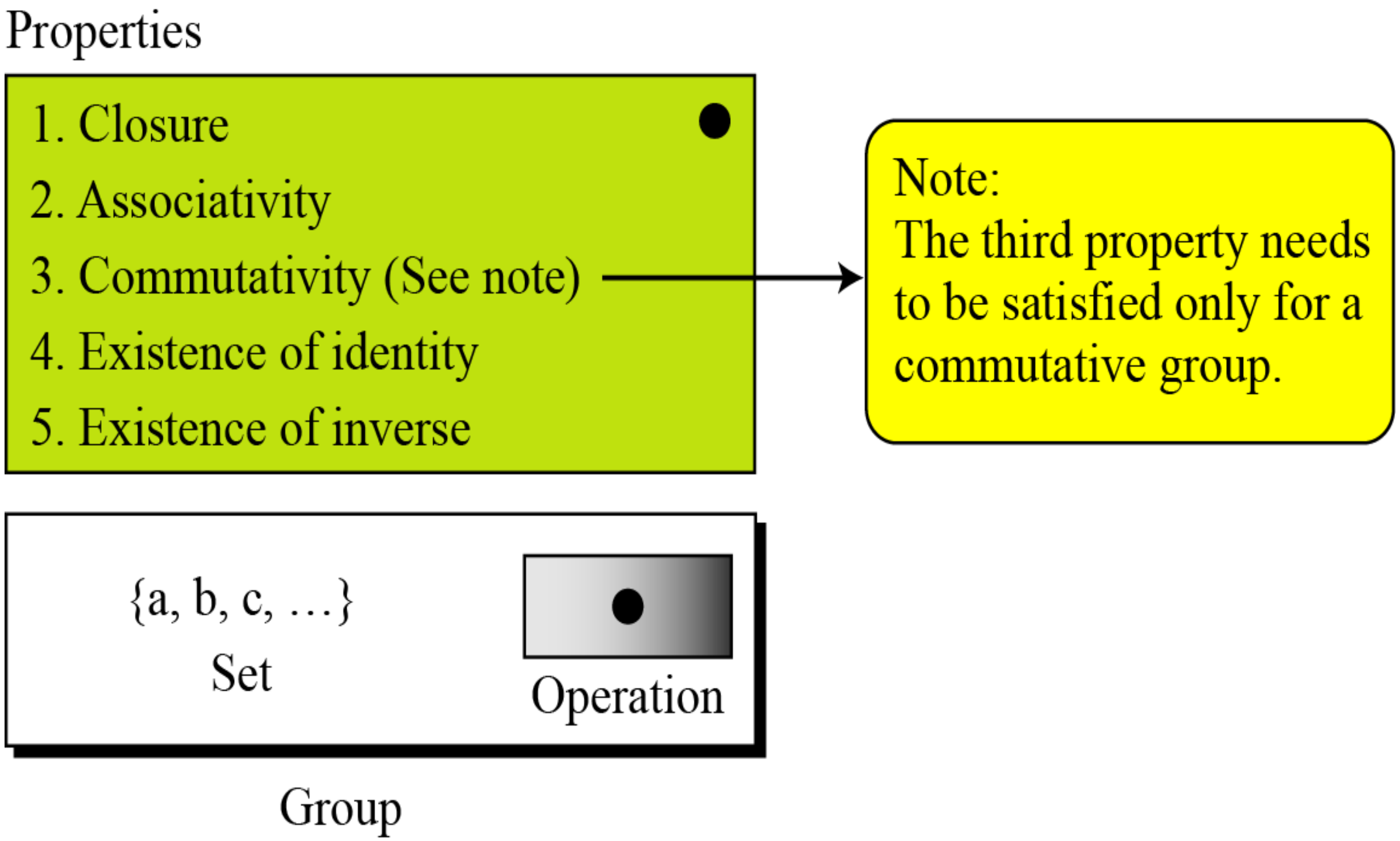


Group

- A group (G) is a set of elements with a binary operation (•) that satisfies four properties. A commutative group satisfies an extra property, commutativity.

- Closure: If a and b are elements of G, then c=a•b is also an element of G.
- Associativity: If a.b & c are elements of G, then (a•b) •c=a•(b•c)
- Commutativity: For all a & b in G, a•b=b•a
- Existence of Identity: For all a in G, there exist an identity element e such that e•a=a•e=a
- Existence of Inverse : For all a in G, there exist an element a’ such that a•a’=a’•a=e

Group

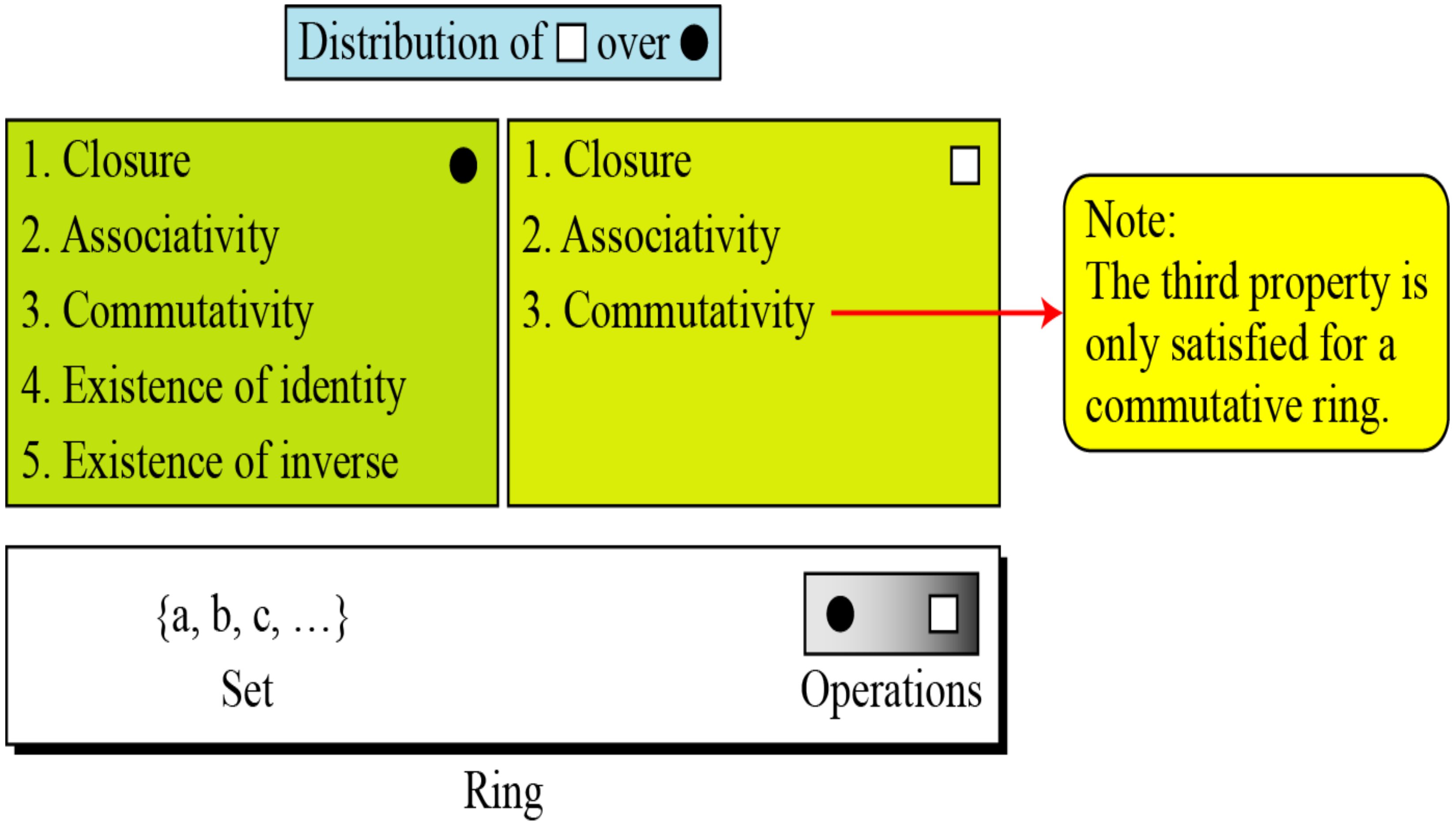

Properties
1. Closure ●
2. Associativity
3. Commutativity (See note)
4. Existence of identity
5. Existence of inverse
Note:
The third property needs to be satisfied only for a commutative group.
{a, b, c, …}
Set
●
Operation
Group

Examples of Group

- The set of residue integers with the addition operator, $G = < Z_n , + >$, is a commutative group.
- The set Z_{n^*} with the multiplication operator, $G = < Z_{n^*}, \times >$, is also an abelian group.
- Let us define a set $G = < \{a, b, c, d\}, \bullet >$ and the operation as shown in Table below.

•	*a*	*b*	*c*	*d*
a	*a*	*b*	*c*	*d*
b	*b*	*c*	*d*	*a*
c	*c*	*d*	*a*	*b*
d	*d*	*a*	*b*	*c*

Ring


Distribution of □ over ●
1. Closure ●
2. Associativity
3. Commutativity
4. Existence of identity
5. Existence of inverse
1. Closure □
2. Associativity
3. Commutativity
Note: The third property is only satisfied for a commutative ring.
{a, b, c, …}
Set
● □
Operations
Ring

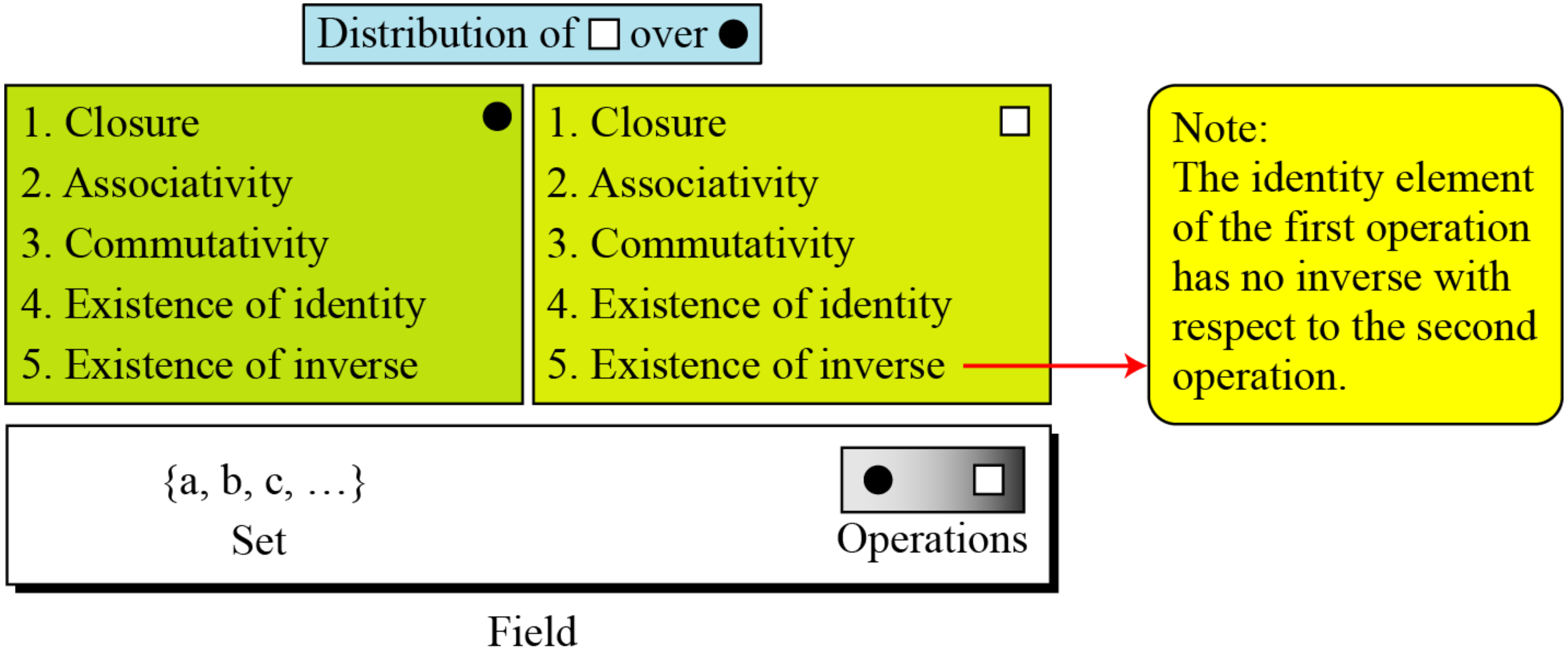
Ring

- A ring, R = <{…}, •, □ >, is an algebraic structure with two operations.
- The first operation must satisfy all the five properties of abelian group and second operation must satisfy only two or three properties for ring & abelian ring respectively.
- In addition, second operation must be distributed over the first one.
- ∀a,b, & c in R, we have

  a □ (b•c)=(a □ b) •(a □ c) &   (a•b) □ c=(a □ c) •(a □ b)
- The set Z with two operations, addition and multiplication, is a commutative ring. We show it by R = <Z, +, ×>. Addition satisfies all of the five properties; multiplication satisfies only three properties.

Field

- A field, denoted by **F** = <{…}, •, □ > is a commutative ring in which the second operation satisfies all the five properties defined for the first operation except that the identity of the first operation has no inverse (sometimes called the zero element) with respect to the second operation.

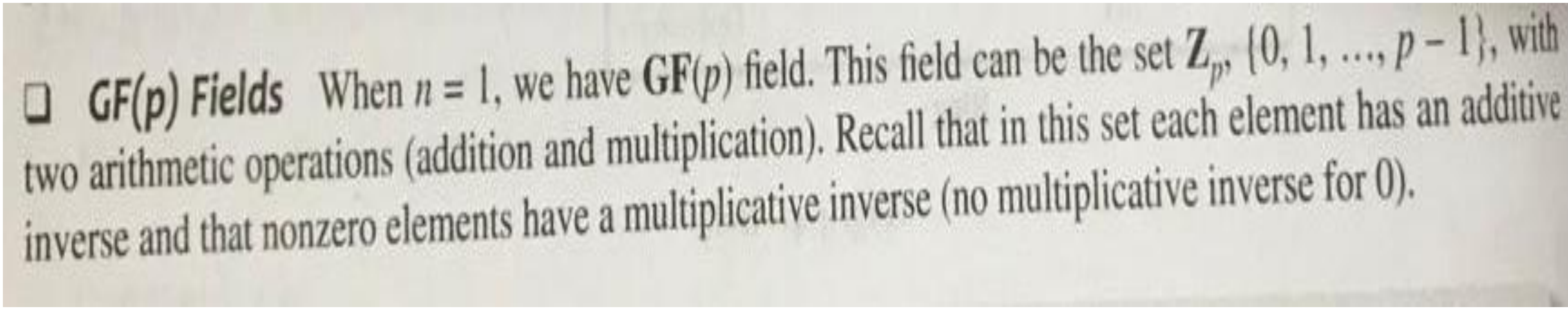


Finite Fields

- Galois showed that for a field to be finite, the number of elements should be p^n, where p is a prime and n is a positive integer.

A Galois field, GF(p^n), is a finite field with p^n elements.

Finite Fields



❑ ***GF(p) Fields*** When $n = 1$, we have **GF**(p) field. This field can be the set $\mathbf{Z}_p$, $\{0, 1, \ldots, p-1\}$, with two arithmetic operations (addition and multiplication). Recall that in this set each element has an additive inverse and that nonzero elements have a multiplicative inverse (no multiplicative inverse for 0).

- A very common field in this category is GF(2) with the set {0, 1} and two operations, addition and multiplication, as shown in Figure below.

GF(2)

{0, 1}   + ×

+	0	1
0	0	1
1	1	0

Addition

×	0	1
0	0	0
1	0	1

Multiplication

a	0	1
−a	0	1

a	0	1
a^{-1}	—	1

Inverses

Finite Fields

- We can define GF(5) on the set Z_5 (5 is a prime) with addition and multiplication operators as shown in fig below.

GF(5)

$\{0, 1, 2, 3, 4\}$ $+ \times$

+	0	1	2	3	4
0	0	1	2	3	4
1	1	2	3	4	0
2	2	3	4	0	1
3	3	4	0	1	2
4	4	0	1	2	3

Addition

×	0	1	2	3	4
0	0	0	0	0	0
1	0	1	2	3	4
2	0	2	4	1	3
3	0	3	1	4	2
4	0	4	3	2	1

Multiplication

Additive inverse

a	0	1	2	3	4
$-a$	0	4	3	2	1

a	0	1	2	3	4
a^{-1}	–	1	3	2	4

Multiplicative inverse

Message Authentication & MAC

- In the context of communication across the network, following attacks can be identified.

1. Disclosure
2. Traffic Analysis
3. Masquerade
4. Content Modification
5. Sequence Modification
6. Timing Modification
7. Source Repudiation
8. Destination Repudiation

Authentication Requirements

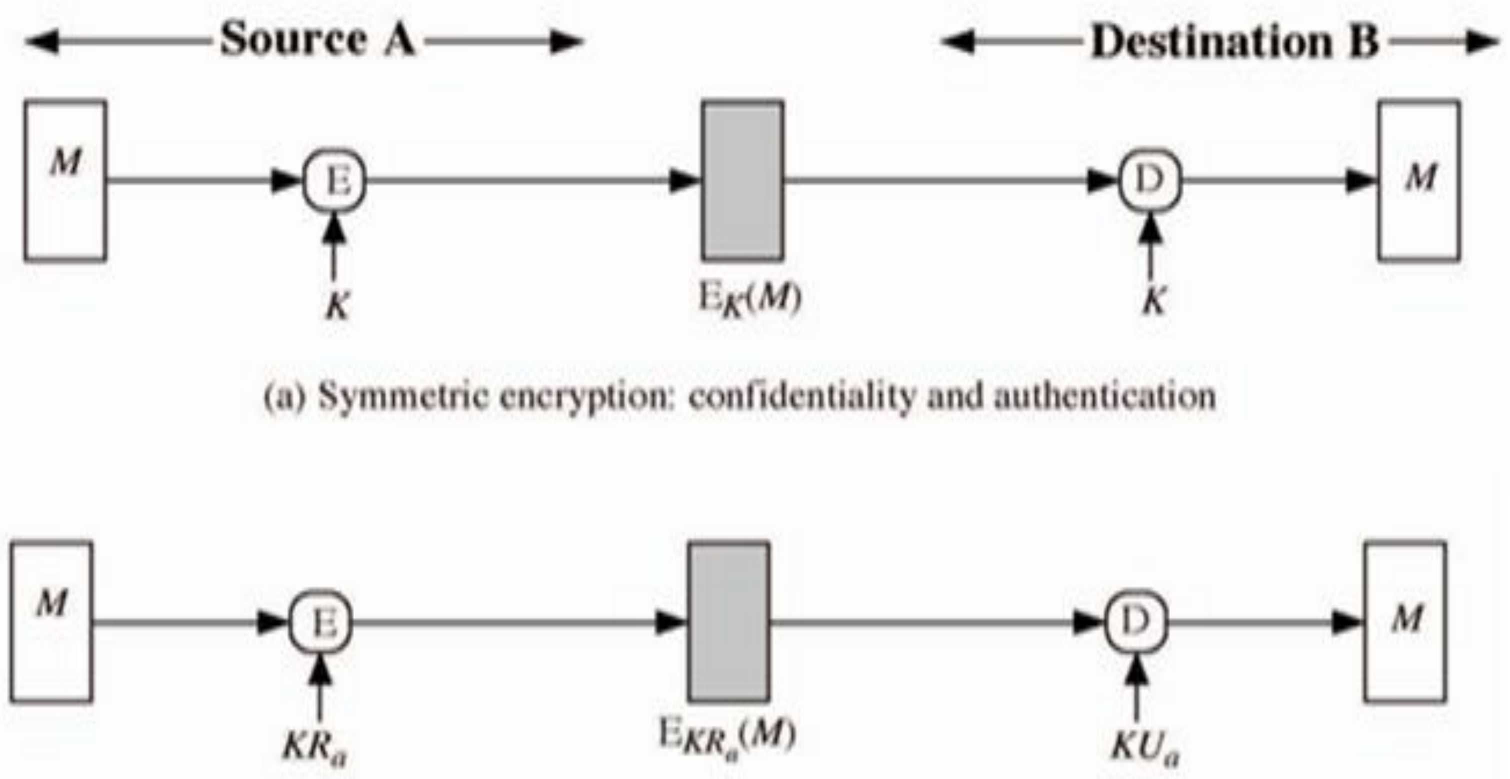
- Measures to deal 1 & 2 attacks comes under message privacy.
- Measures to deal with 3 to 6 comes under message authentication.
- Measures to deal with 7 comes under digital signature. However, may also be used to address attacks from 3 to 6.
- Measures to deal with 8 requires digital signature along with few other protocols.

Authentication Functions

- Any message authentication is fundamentally having two levels.
- At the lower level, some authenticator is generated & at the next level, generated authenticator is used to authenticate the message.
- Three types of functions that may be used to produce the authenticator.

- Message encryption: Ciphertext of the entire message serves as authenticator.
- Message authentication Code (MAC) : Function of the message and a secret key that produces a fixed length value that serves as authenticator.
- Hash function: Function that maps a message to fixed length value that serves as authenticator.

Message Encryption



(a) Symmetric encryption: confidentiality and authentication

(c) Public-key encryption: authentication and signature

Basic Uses of Message Encryption

Message Authentication Code (MAC)

- Here MAC function C creates a small fixed-sized block depending on both message M and a shared secret key K.
- MAC is appended to the message M

$$MAC=C_K(M)$$

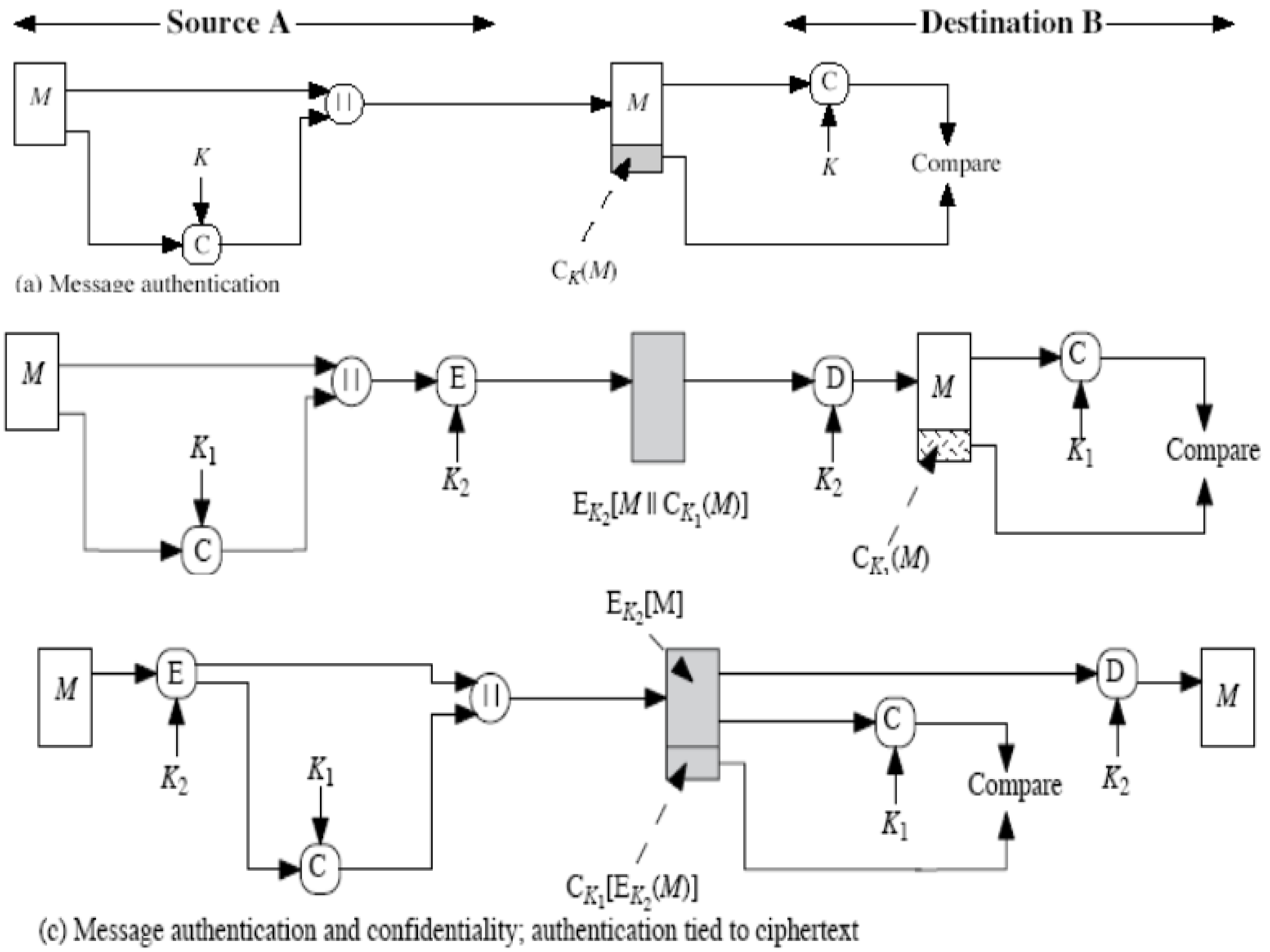
M: Message

K: Secret key

C: MAC function

MAC: Message authentication code

Message Authentication Code (MAC)



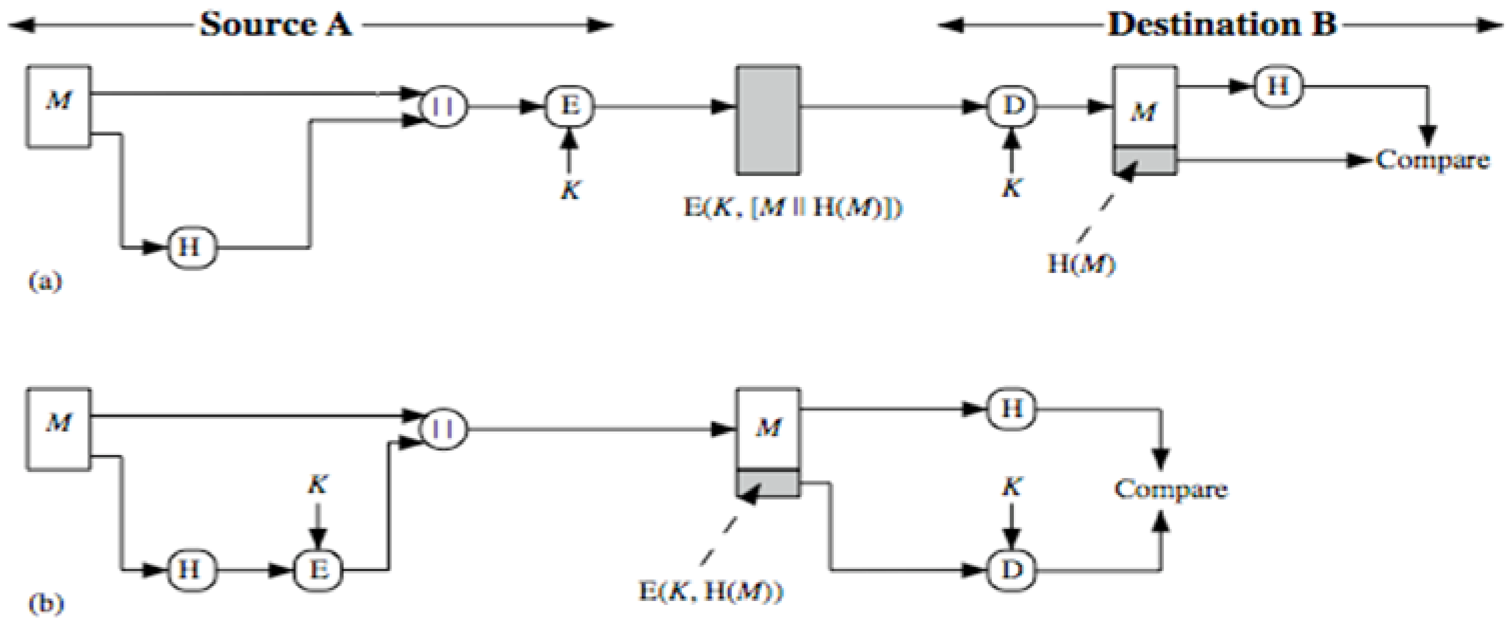
(c) Message authentication and confidentiality; authentication tied to ciphertext

Hash Functions

- A variation of the MAC is a one way hash function.
- Here hash function accepts a variable size message M as input and produces a fixed size output, referred to as H(M).

(a) Encrypt message plus hash code.

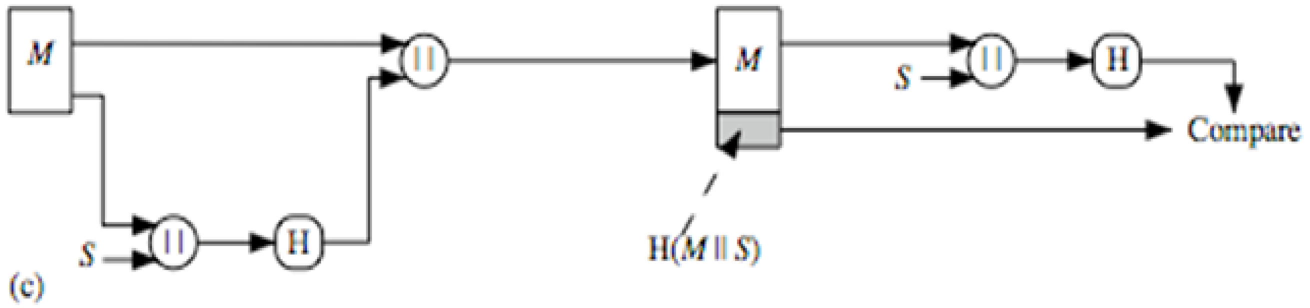
(b) Encrypt hash code - shared secret key

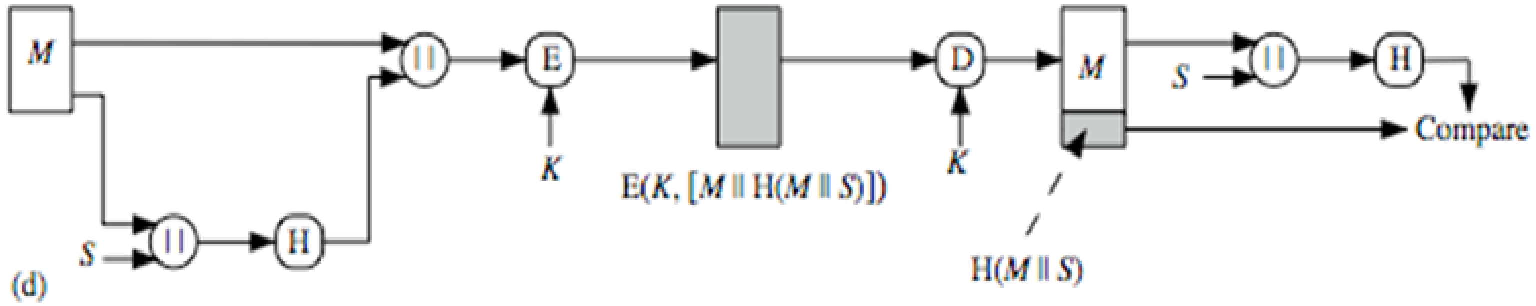


Hash Functions

(c) Compute hash code of message plus secret value.

(d) Encrypt result of (c).





MAC function Analysis

- Cryptanalysis can be done as follows:
- Suppose k>n where k is the key size and n is the MAC size.
- Given a known M_1 and MAC_1 with $MAC_1=C_{K1}(M_1)$.
- The cryptanalysis can perform $MAC_i=C_{Ki}(M_1)$ for all possible key values.
- At least one key is guaranteed to produce a match of $MAC_i=MAC_1$.
- As there are total of 2^n MAC with $2^n<2^k$ keys i.e. a number of keys will produce the correct MAC.
- On an average, $2^k/2^n=2^{k-n}$ keys will produce a match.

Requirements of MAC function

- If an opponent observes M and $C_k(M)$, it is computationally infeasible for the opponent to construct the message M’ such that $C_k(M')=C_k(M)$.

- $C_k(M)$ should be uniformly distributed such that with randomly chosen messages M and M’, the probability $C_k(M)=C_k(M')$ is 2^{-n} where n is the number of bits in the MAC.

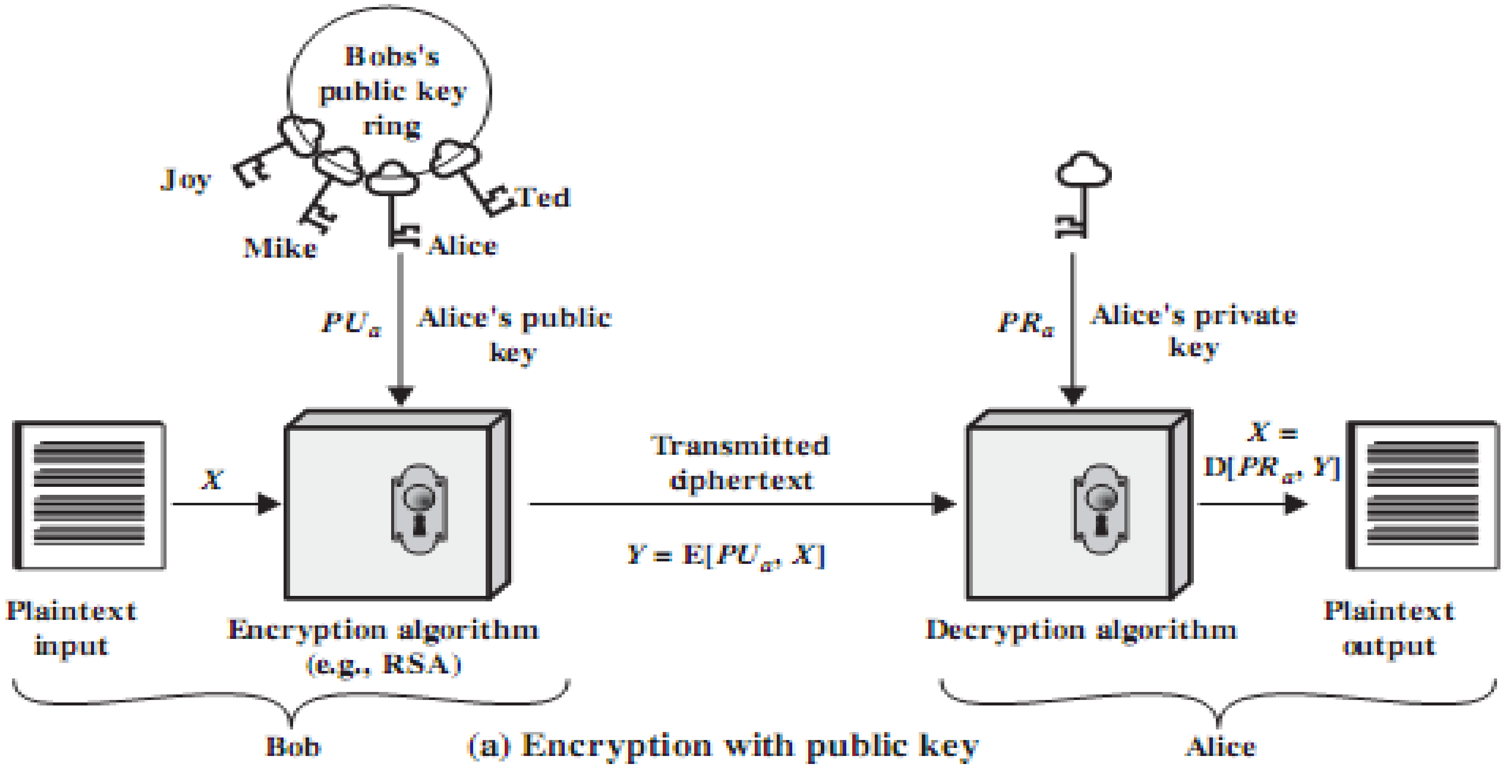
Requirements of Hash function

- H can be applied to block of data of any size.
- H produces fixed-length output.
- H(x) is relatively easy to compute for any given x.
- For any given h, it is computationally infeasible to find x such that H(x)=h. This is referred to as one-way property.
- For any given block x, it is computationally infeasible to find y≠x with H(y)=H(x). This is referred to as weak collision resistance.
- It is computationally infeasible to find pair (x,y) such that H(x)=H(y). This is referred to as strong collision resistance.

Asymmetric Key cryptosystem

- Main ingredients of Public Key Cryptosystem:
  - Plaintext
  - Encryption algorithm
  - Public and private key
  - Ciphertext
  - Decryption algorithm

Public Key Cryptography: Encryption



(a) Encryption with public key

Public Key Cryptography: Authentication

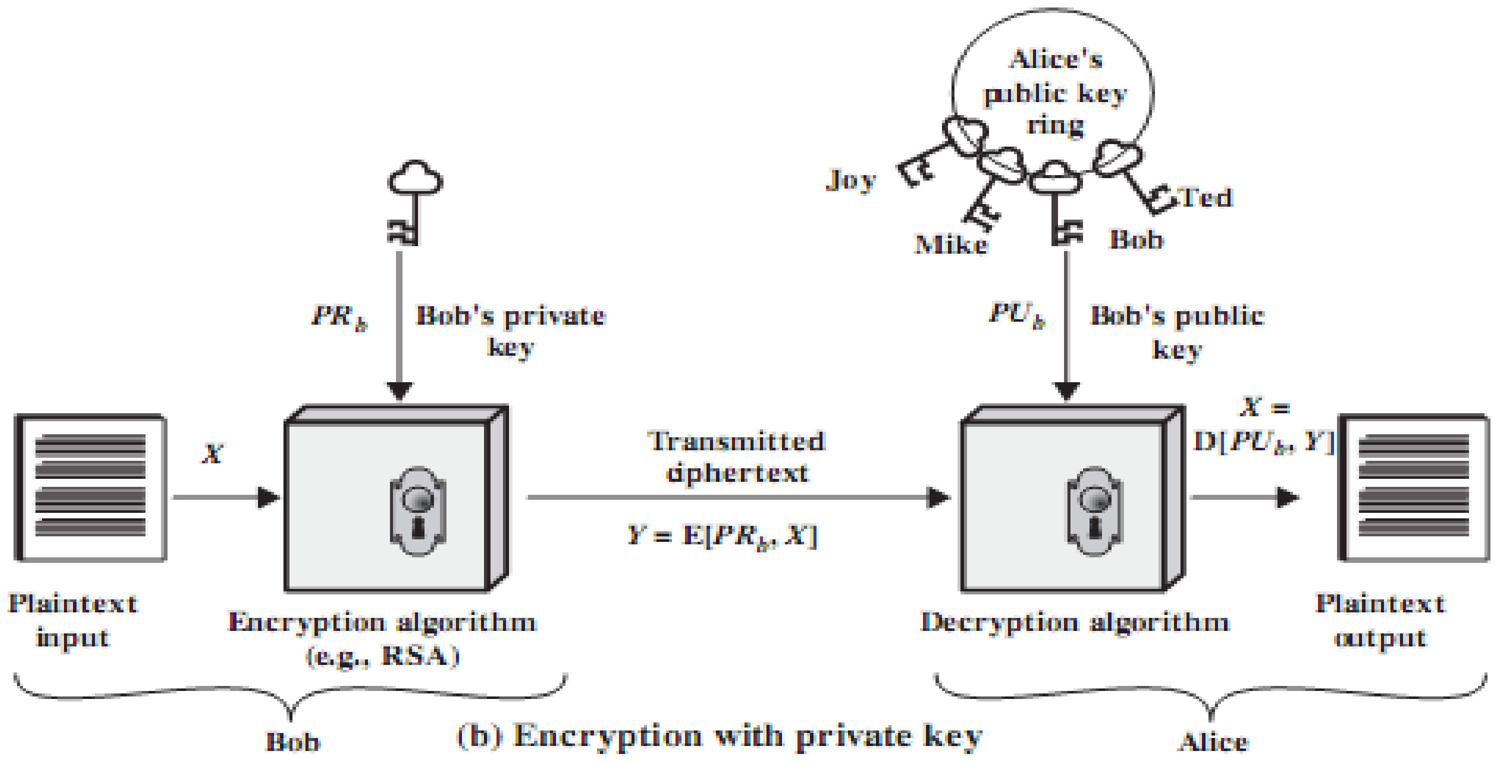
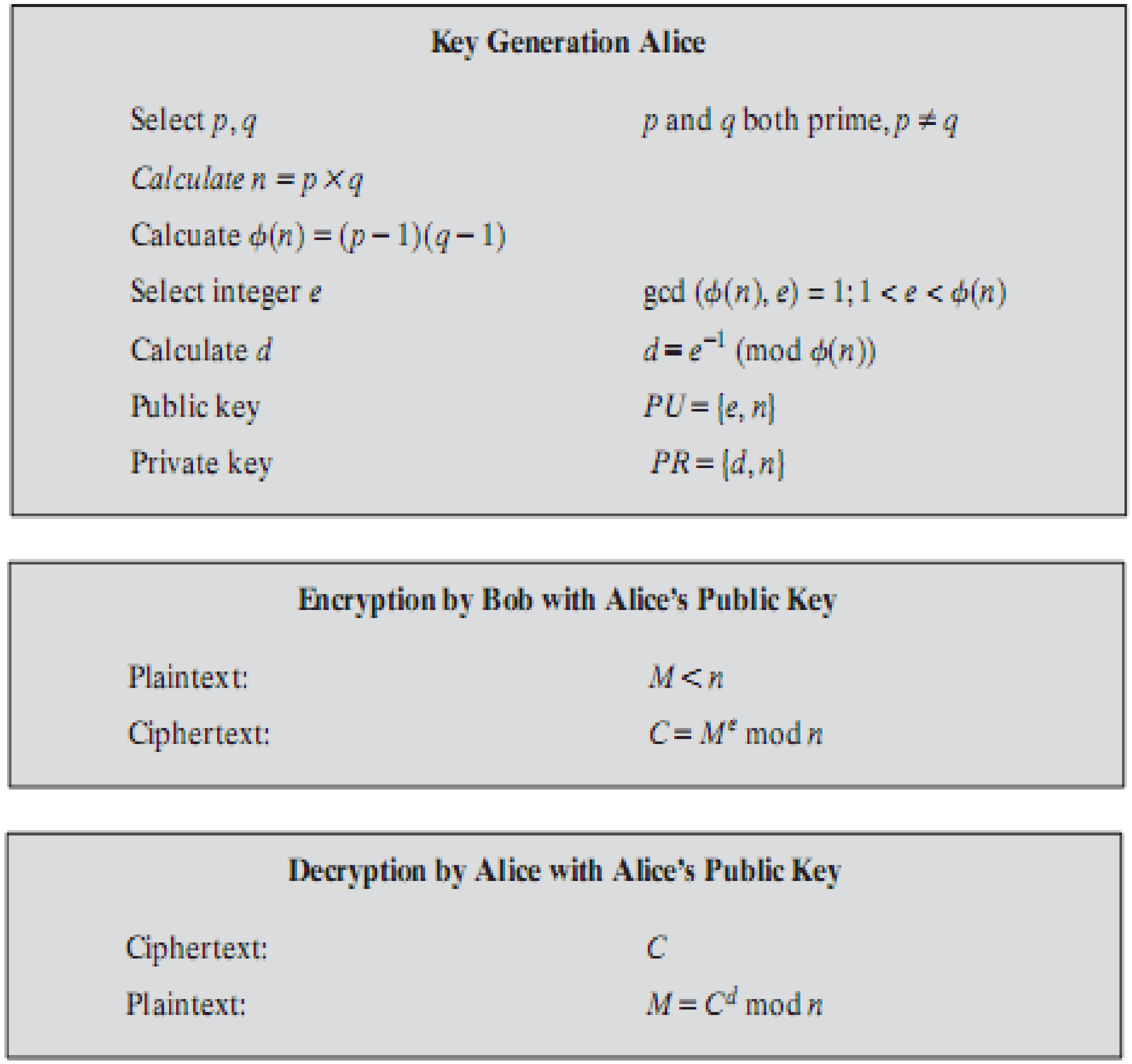


Figure 9.1 Public-Key Cryptography

RSA algorithm

- Named after inventors Ron Rivest, Adi Shamir and Len Adleman.
- RSA is a block cipher between 0 and n-1 for some n.
- Typical size of n is 1024 bits or 309 digits.

RSA Algorithm



Key Generation Alice

Select p, q	p and q both prime, $p \neq q$
Calculate $n = p \times q$	
Calcuate $\phi(n) = (p-1)(q-1)$	
Select integer e	$\gcd(\phi(n), e) = 1; 1 < e < \phi(n)$
Calculate d	$d \equiv e^{-1} \pmod{\phi(n)}$
Public key	$PU = \{e, n\}$
Private key	$PR = \{d, n\}$

Encryption by Bob with Alice's Public Key

Plaintext:	$M < n$
Ciphertext:	$C = M^e \bmod n$

Decryption by Alice with Alice's Public Key

Ciphertext:	C
Plaintext:	$M = C^d \bmod n$

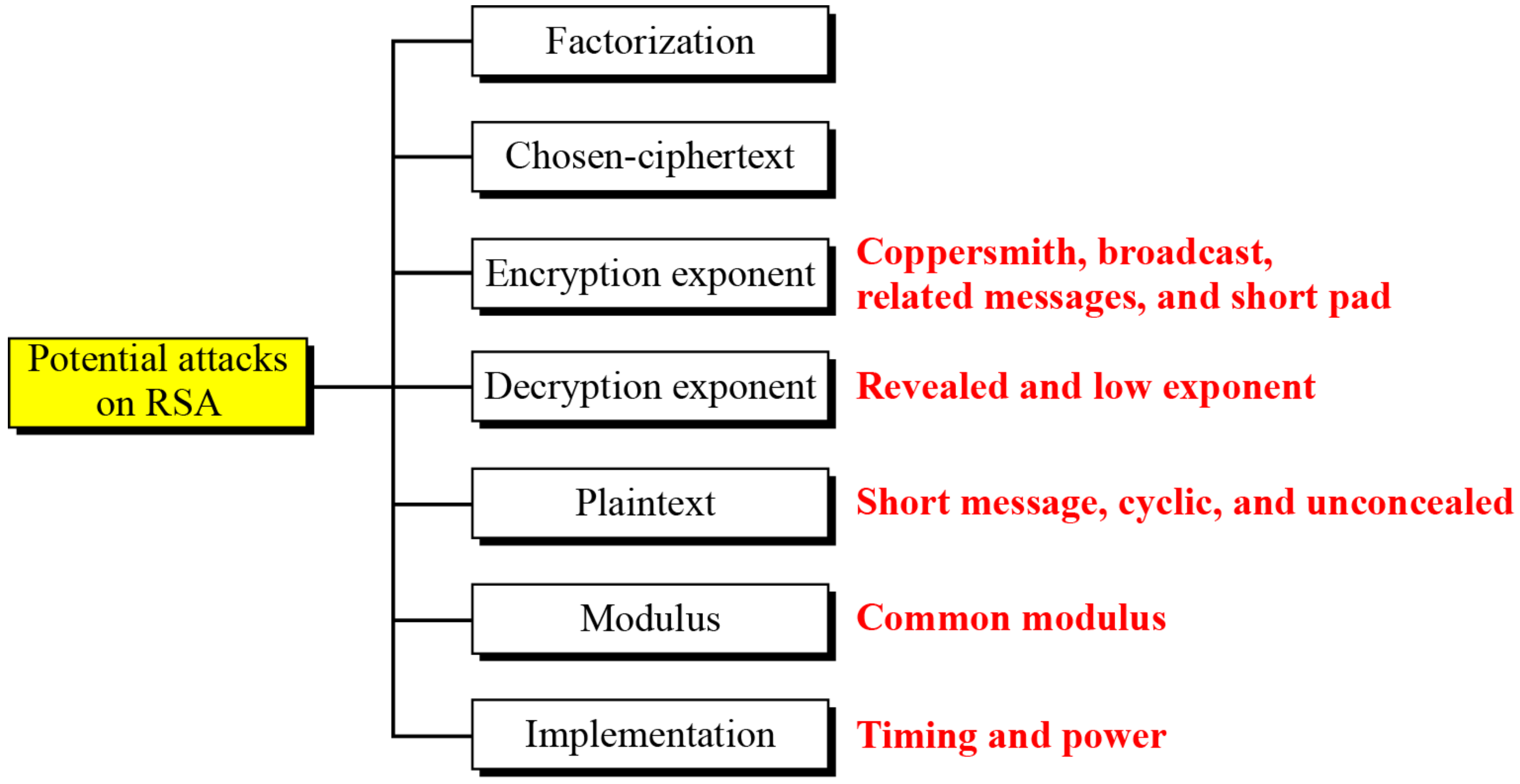
Figure 9.5 The RSA Algorithm

RSA Example

- Select two prime numbers: p=5, q=7.
- n=5*7=35
- $\phi(n)=24$
- e=5;d=5; (e*d) mod $\phi(n)=1$.
- Take plaintext M=2, then after encryption 2^5 mod35= 32.
- Ciphertext 32^5 mod 35=2. (Hint: 33554432)

Potential Attacks of RSA

Figure 10.8 *Taxonomy of potential attacks on RSA*



Attacks on RSA

- **Factorization** : Eve can factor n to obtain p & q and can calculate $\phi(n)=(p-1)*(q-1)$. It can then calculate inverse d such that $e*d \bmod \phi(n)= 1$.

- **Chosen Ciphertext Attack**:

  Assume that A creates ciphertext **C**$=P^e$ mod n and send **C** to B. Eve intercepts C and uses the following to find P.

  1. Eve choses a random integer X in Z_n^*.
  2. Eve calculates $Y=C * X^e$ mod n.
  3. Eve sends Y to B for decryption and get $Z=Y^d$ mod n. (**Chosen Cipertext**)
  4. Eve can find P because

     $Z=Y^d \bmod n= (C * X^e)^d \bmod n= (C^d * X^{ed}) \bmod n=$

     $(C^d * X) \bmod n=(P * X) \bmod n.$

     $Z= (P * X) \bmod n \rightarrow P=(Z * X^{-1}) \bmod n$

Attacks on RSA

- **Attack on Encryption Exponent**:

  The broadcast attack can be launched if one entity sends the same message to a group of recipients with the same low encryption exponent.

  For example: A sends the same message to three recipients with the same public exponent e=3 and the moduli n_1, n_2 and n_3.

  $C_1=P^3$ mod n1 $C_2=P^3$ mod n2 $C_3=P^3$ mod n3

  Apply CRT algorithm, to find the values of P^3 and thus can calculate different values for C_1, C_2 & C_3

Attacks on RSA

- **Attack on Decryption Exponent**:

  If intruder can find the decryption exponent d, then it can decrypt the current encrypted message.

  However, if Eve knows the value of d, it can use probabilistic algorithm to factor n and find the values of p and q.

  This means that if B finds out that the decryption exponent is compromised, he needs to create new value of n, public key and private key.

Attacks on RSA

- **Attacks on the Modulus**:

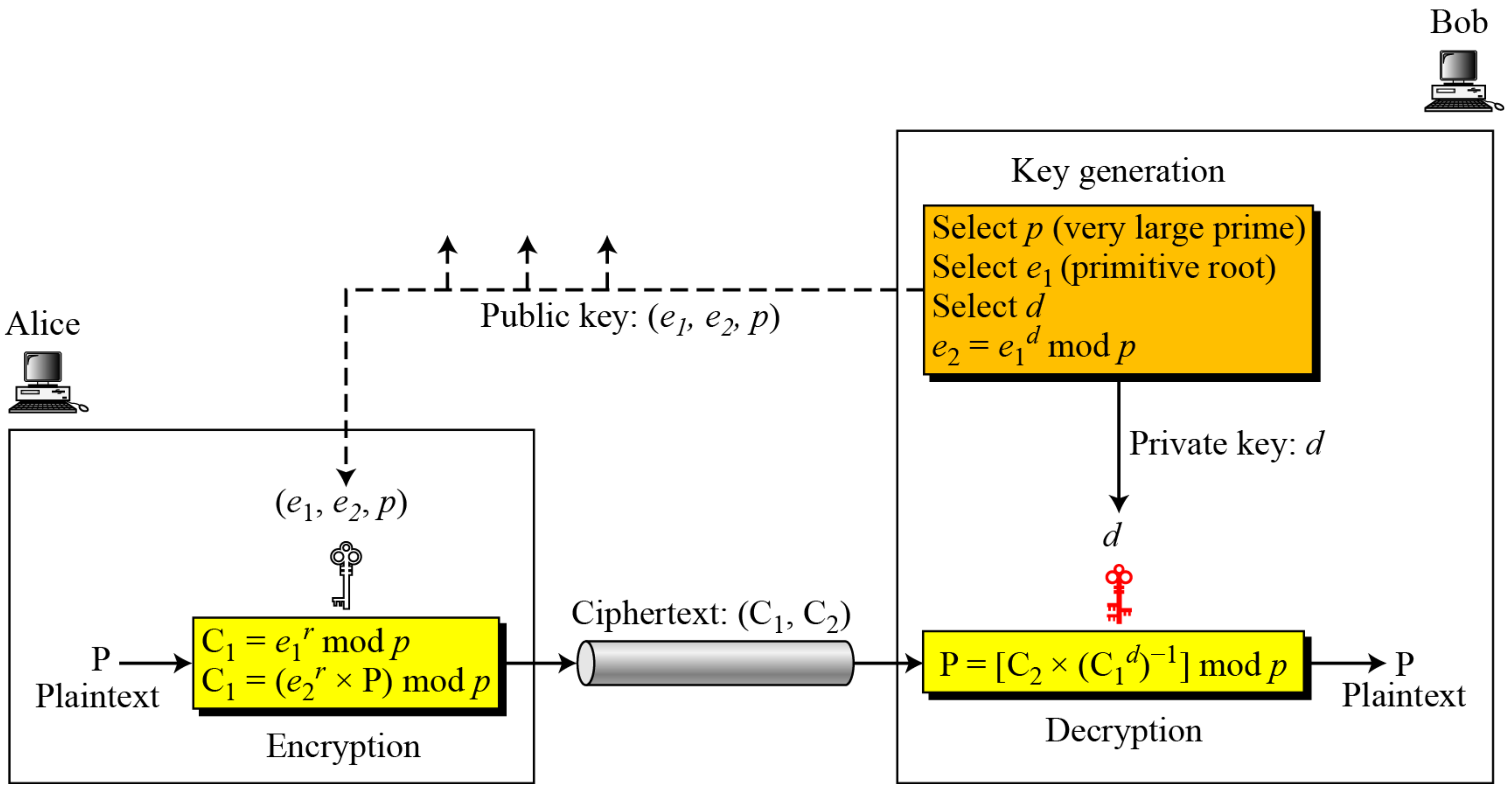
  Common modulus attack: This attack can be launched if a community uses a common modulus like n.

  For example: People in a community might let a trusted party select p & q, calculate n and $\phi(n)$ and create a pair of exponents for each entity.

  Using its own exponents, eve can launch probabilistic attack to factor n and find B’s private key. (Assumtion Eve is also a part of community).

Elgamal Cryptosystem

- Figure 10.11 *Key generation, encryption, and decryption in ElGamal*



Elgamal Cryptosystem

Algorithm 10.9 *ElGamal key generation*

```
ElGamal_Key_Generation
{
    Select a large prime $p$
    Select $d$ to be a member of the group $\mathbf{G} = <\mathbf{Z}_p{}^*, \times>$ such that $1 \le d \le p - 2$
    Select $e_1$ to be a primitive root in the group $\mathbf{G} = <\mathbf{Z}_p{}^*, \times>$
    $e_2 \leftarrow e_1{}^d \bmod p$
    Public_key $\leftarrow (e_1, e_2, p)$                    // To be announced publicly
    Private_key $\leftarrow d$                               // To be kept secret
    return Public_key and Private_key
}
```

Elgamal Cryptosystem

Algorithm 10.10 *ElGamal encryption*

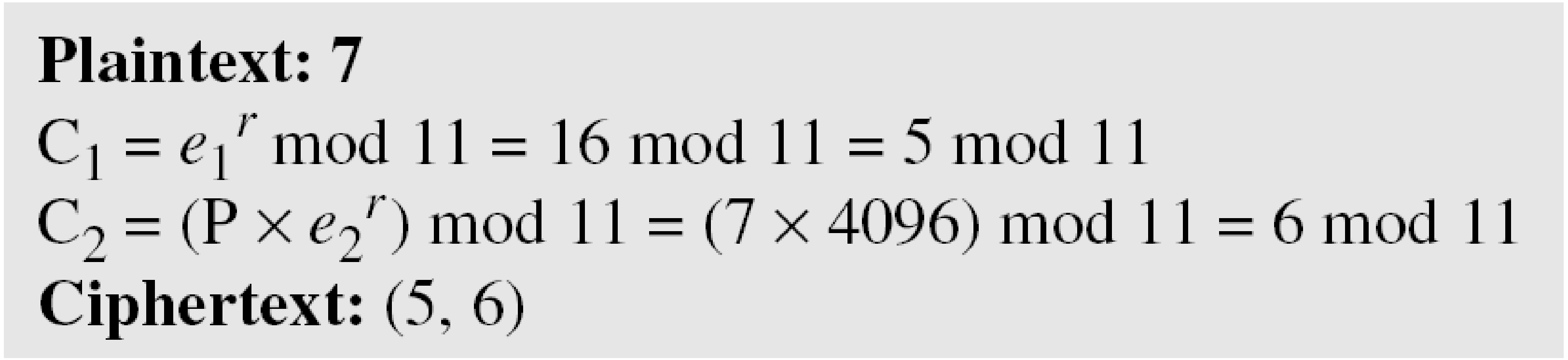
```
ElGamal_Encryption ($e_1$, $e_2$, $p$, P)                    // P is the plaintext
{
    Select a random integer $r$ in the group $\mathbf{G} = < \mathbf{Z}_p^*, \times >$
    $C_1 \leftarrow e_1^r \bmod p$
    $C_2 \leftarrow (P \times e_2^r) \bmod p$                 // $C_1$ and $C_2$ are the ciphertexts
    return $C_1$ and $C_2$
}
```

Algorithm 10.11 *ElGamal decryption*

```
ElGamal_Decryption ($d$, $p$, $C_1$, $C_2$)                  // $C_1$ and $C_2$ are the ciphertexts
{
    $P \leftarrow [C_2 (C_1^d)^{-1}] \bmod p$                 // P is the plaintext
    return P
}
```

Example of Elgamal Cryptosystem

- *Here is a trivial example. Bob chooses $p = 11$ and $e_1 = 2$. and $d = 3$ $e_2 = e_1^d = 8$. So the public keys are (2, 8, 11) and the private key is 3. Alice chooses $r = 4$ and calculates C1 and C2 for the plaintext 7.*



Plaintext: 7

$C_1 = e_1^{\,r} \bmod 11 = 16 \bmod 11 = 5 \bmod 11$

$C_2 = (P \times e_2^{\,r}) \bmod 11 = (7 \times 4096) \bmod 11 = 6 \bmod 11$

Ciphertext: (5, 6)

Bob receives the ciphertexts (5 and 6) and calculates the plaintext.

Inverse calculation : $P=[C_2 x(C_1^{d})^{-1}] \bmod p$ $P=[C_2 x C_1^{p-1-d}] \bmod p$

EXPONENTIATION AND LOGARITHM

Exponentiation: $y = a^x \quad \rightarrow \quad$ **Logarithm:** $x = \log_a y$

Discrete Logarithm

Cyclic Group *If g is a primitive root in the group, we can generate the set Z_n* as $Z_n* = \{g^1, g^2, g^3, \ldots, g^{\phi(n)}\}$*

Example 9.52

The group $G = \langle Z_{10}^*, \times\rangle$ has two primitive roots because $\phi(10) = 4$ and $\phi(\phi(10)) = 2$. It can be found that the primitive roots are 3 and 7. The following shows how we can create the whole set Z_{10}^* using each primitive root.

$g = 3 \rightarrow$	$g^1 \bmod 10 = 3$	$g^2 \bmod 10 = 9$	$g^3 \bmod 10 = 7$	$g^4 \bmod 10 = 1$
$g = 7 \rightarrow$	$g^1 \bmod 10 = 7$	$g^2 \bmod 10 = 9$	$g^3 \bmod 10 = 3$	$g^4 \bmod 10 = 1$

The group $G = \langle Z_n^*, \times\rangle$ is a cyclic group if it has primitive roots.
The group $G = \langle Z_p^*, \times\rangle$ is always cyclic.

Discrete Logarithm

The idea of Discrete Logarithm

Properties of $G = \langle Z_p{}^*, \times \rangle$:

1. *Its elements include all integers from 1 to* $p - 1$.

2. *It always has primitive roots.*

3. *It is cyclic. The elements can be created using* g^x *where* x *is an integer from 1 to* $\phi(n) = p - 1$.

4. *The primitive roots can be thought as the base of logarithm.*

Discrete Logarithm

5. *If the group has k primitive roots, calculations can be done in k different bases.*

6. *Given* x = $\log_g$ y for any element y in the set, there is another element x that is the log of y in base g.

7. *This type of logarithm is called discrete logarithm.*

Discrete Logarithm

Solution to Modular Logarithm Using Discrete Logs

Now let us see how to solve the problem of type
$y = a^x \pmod{n}$ when y is given and we need to find x.

Tabulation of Discrete Logarithm: One way to solve above mentioned problem is to use a table for each Z_p^* and different bases. This type of table can be precalculated and saved.

Table 9.6 *Discrete logarithm for* $\mathbf{G} = \langle \mathbf{Z}_7^*, \times \rangle$

y	*1*	*2*	*3*	*4*	*5*	*6*
$x = L_3\, y$	6	2	1	4	5	3
$x = L_5\, y$	6	4	5	2	1	3

Discrete Logarithm

Given the tabulation for other discrete logarithms for every group and all possible bases, we can solve any discrete logarithm problem.

This is similar to the past with traditional logarithms.

Before the era of calculators and computers, tables were used to calculate logarithms in base 10.

Discrete Logarithm

Example 9.53

Find x in each of the following cases:

a. $4 \equiv 3^x \pmod 7$.

b. $6 \equiv 5^x \pmod 7$.

Solution

We can easily use the tabulation of the discrete logarithm in Table 9.6.

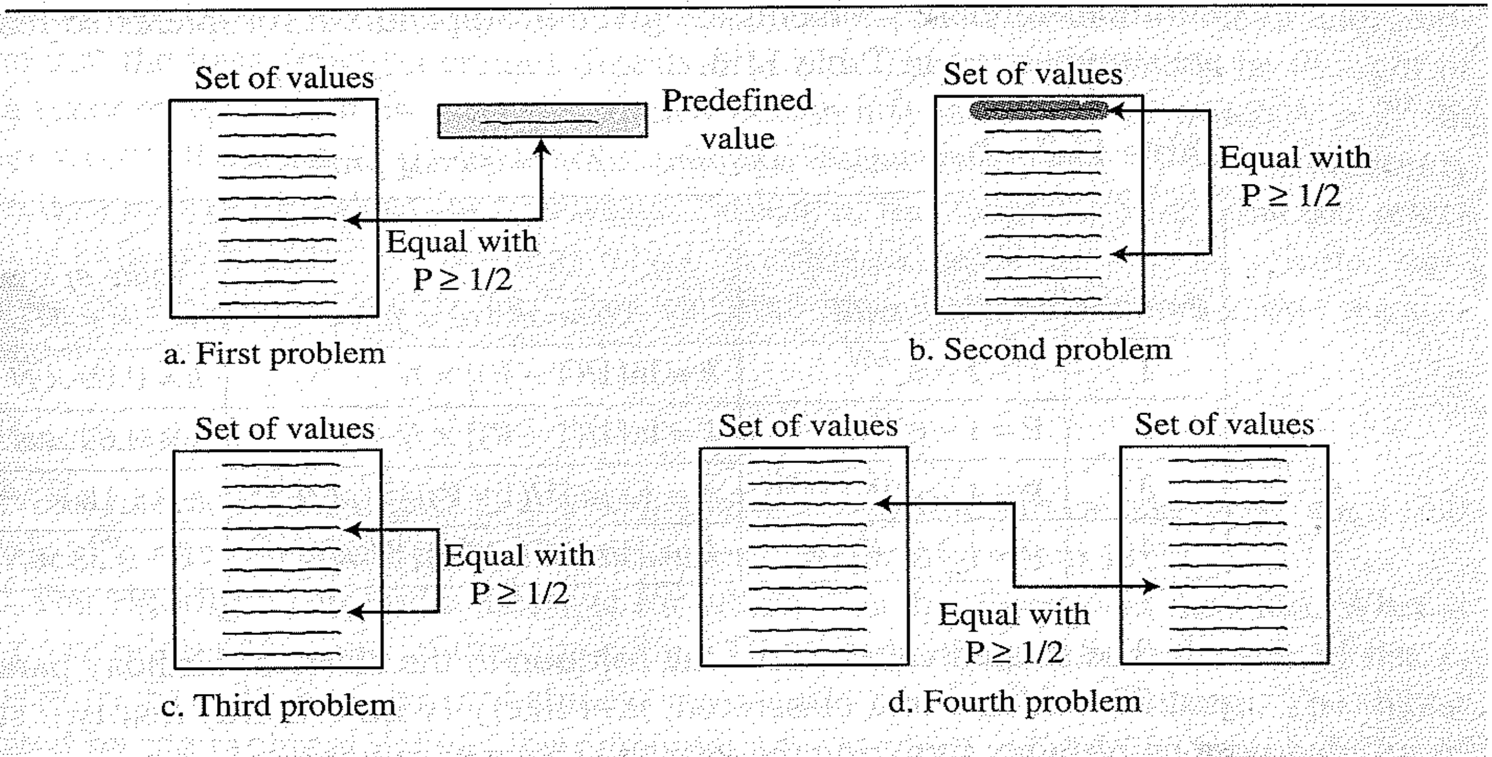
a. $4 \equiv 3^x \bmod 7 \rightarrow x = L_3 4 \bmod 7 = 4 \bmod 7$

b. $6 \equiv 5^x \bmod 7 \rightarrow x = L_5 6 \bmod 7 = 3 \bmod 7$

Birthday Problems

- The four different birthday problems usually there in probability.
- The third problem is refereed to as Birthday paradox.

Figure 11.7 *Four birthday problems*



Birthday Problems

- **Problem 1**: What is the minimum number, k, of students in classroom such that it is likely that at least one student has a predefined birthday.
- We have a uniformly distributed random variable with N possible values, then what is the minimum number of instances k such that it is likely that at least one instance is equal to a predefined value.
- **Problem 2**: What is the minimum number, k, of students in classroom such that it is likely that at least one student has the same birthday as the student selected by the professor.
- We have a uniformly distributed random variable with N possible values, then what is the minimum number of instances k such that it is likely that at least one instance is equal to the selected one.

Birthday Problems

- **Problem 3**: What is the minimum number, k, of students in classroom such that it is likely that at least two students have the same birthday.
- We have a uniformly distributed random variable with N possible values, then what is the minimum number of instances k such that it is likely that at least two instances are equal.
- **Problem 4**: We have two classes, each with k students. What is the minimum value k so that it is likely that at least one student from the first classroom has the same birthday as a student from the second classroom
- We have a uniformly distributed random variable with N possible values. We generate two sets of random values each with k instances. What is the minimum number k such that it is likely that at least one instance from the first set is equal to one instance on the second set.

Birthday Problems

- **23** is the solution to the classical Birthday paradox problem. If there are just 23 students in a classroom, it is likely that (with P >1/2) that the two students have the same birthday (ignoring the year of birth).

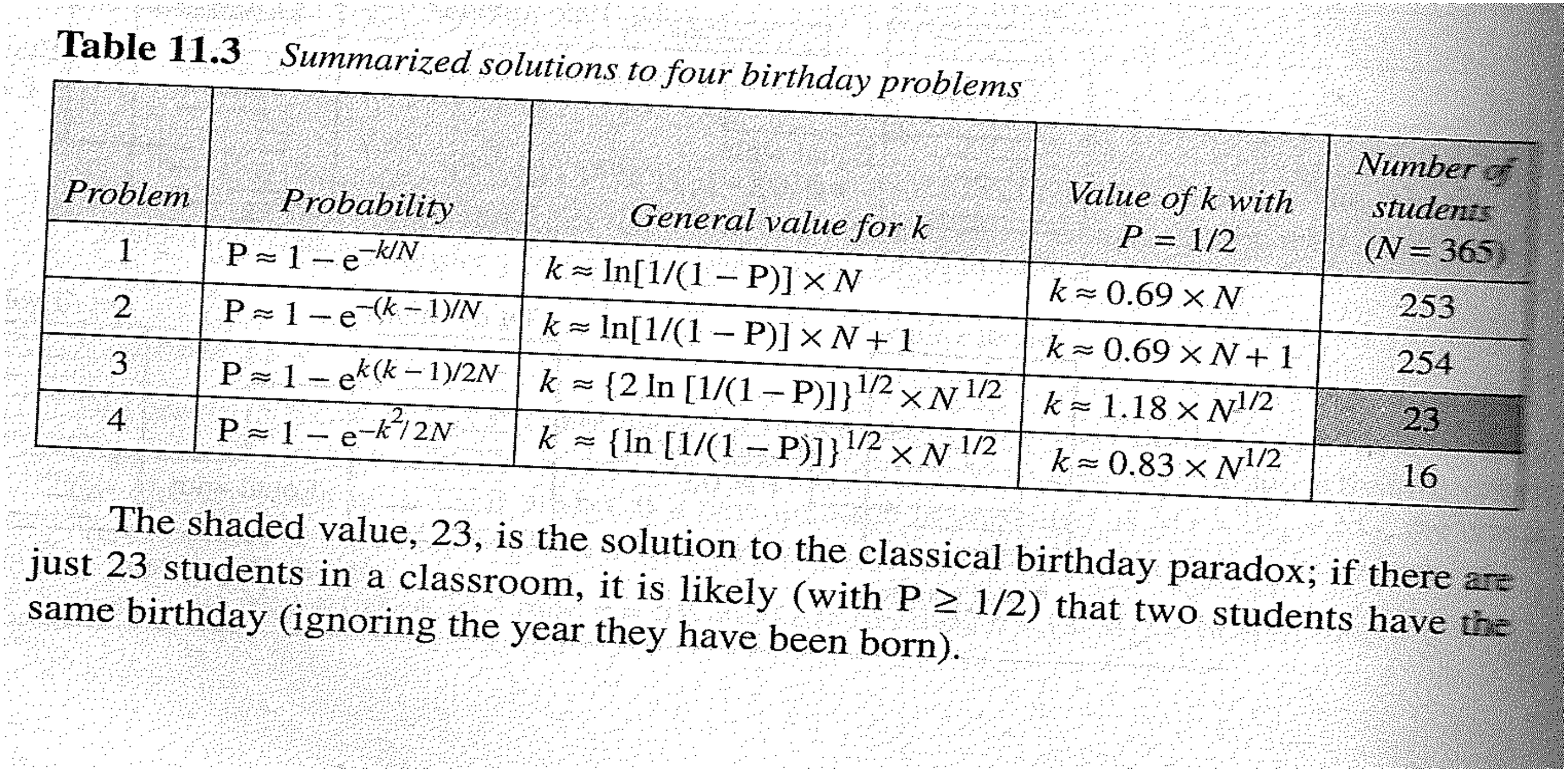


Table 11.3 *Summarized solutions to four birthday problems*

Problem	*Probability*	*General value for k*	*Value of k with P = 1/2*	*Number of students (N = 365)*
1	$P \approx 1 - e^{-k/N}$	$k \approx \ln[1/(1-P)] \times N$	$k \approx 0.69 \times N$	253
2	$P \approx 1 - e^{-(k-1)/N}$	$k \approx \ln[1/(1-P)] \times N + 1$	$k \approx 0.69 \times N + 1$	254
3	$P \approx 1 - e^{k(k-1)/2N}$	$k \approx \{2 \ln[1/(1-P)]\}^{1/2} \times N^{1/2}$	$k \approx 1.18 \times N^{1/2}$	23
4	$P \approx 1 - e^{-k^2/2N}$	$k \approx \{\ln[1/(1-P)]\}^{1/2} \times N^{1/2}$	$k \approx 0.83 \times N^{1/2}$	16

The shaded value, 23, is the solution to the classical birthday paradox; if there are just 23 students in a classroom, it is likely (with $P \geq 1/2$) that two students have the same birthday (ignoring the year they have been born).

Security of Hash Function & MACs

- We can group attacks on hash function and MAC's into two categories: brute force attacks and cryptanalysis.
- **Brute force attack:**
- Hash functions:
- For any given h, it is computationally infeasible to find x such that H(x)=h. This is referred to as one-way property.
- For any given block x, it is computationally infeasible to find y≠x with H(y)=H(x). This is referred to as weak collision resistance.
- It is computationally infeasible to find pair (x,y) such that H(x)=H(y). This is referred to as strong collision resistance.

Security of Hash Function & MACs

- For a code of length n, the level of effort required is proportional to the following:
  - One way: 2^n
  - Weak collision resistance: 2^n
  - Strong Collision resistance: $2^{n/2}$

Security of Hash Function & MACs

- **Brute force attack:**
- MAC:
- Given a fixed message x with n-bit MAC code h=H(x), a brute force method of finding a collision is to pick a random bit string and check if H(y)=H(x).
- There are two lines of attacks possible: Attack the key space and attack the MAC values.
- **Attack on Key space**:
- Suppose the key size is k bits and that the attacker has one known text MAC pair. Then the attacker can compute the n-bit MAC on the known text for all possible keys.
- At least, one key is guaranteed to produce the correct match.
- This phase of attack takes a level of effort proportional to 2^k

Security of Hash Function & MACs

- **Attack on MAC space**:
- The objective is to generate a valid MAC value for a given message or to find a message that matches a given MAC value.
- The level of effort is comparable to that for attacking the one way or weak collision resistance property of a hash function (2^n).
- Finally the level of effort for brute force attack on a MAC algorithm can be expressed as min(2^k,2^n).