Module-III

PUBLIC KEY CRYPTOGRAPHY

PUBLIC KEY CRYPTOGRAPHY

MATHEMATICS OF ASYMMETRIC KEY CRYPTOGRAPHY

- Primes, Primality Testing, Factorization, Euclid's Algorithm, Euler's totient function, Fermat's Theorem, Chinese Remainder Theorem, Exponentiation and logarithm

ASYMMETRIC KEY CIPHERS

- RSA cryptosystem
- Key distribution& Key management
- Diffie Hellman key exchange
- ElGamal cryptosystem
- Elliptic curve cryptography

Chapter 2

Objectives

- To review integer arithmetic, concentrating on divisibility and finding the greatest common divisor using the Euclidean algorithm
- To understand how the extended Euclidean algorithm can be used to solve linear congruent equations, and to find the multiplicative inverses
- To emphasize the importance of modular arithmetic and the modulo operator, because they are extensively used in cryptography
- To emphasize and review matrices and operations on residue matrices that are extensively used in cryptography
- To solve a set of congruent equations using residue matrices

Part-I

Mathematics of Cryptography

Modular Arithmetic, Congruence, and Matrices

2-1 INTEGER ARITHMETIC

In integer arithmetic, we use a set and a few operations. You are familiar with this set and the corresponding operations, but they are reviewed here to create a background for modular arithmetic.

Topics discussed in this section:

2.1.1 Set of Integers

The set of integers, denoted by Z, contains all integral numbers (with no fraction) from negative infinity to positive infinity (Figure 2.1).

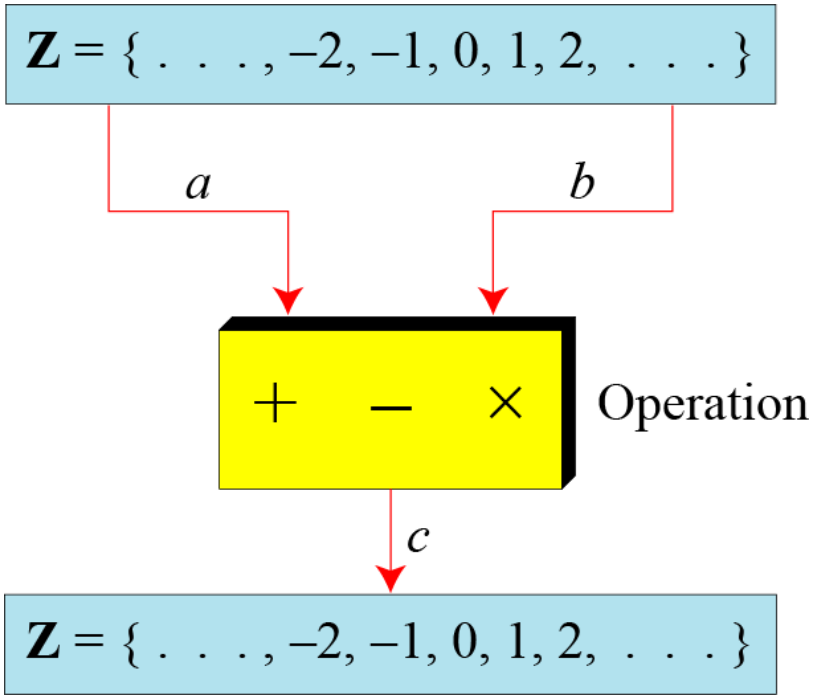
Figure 2.1 *The set of integers*

$$\mathbf{Z} = \{\ldots, -2, -1, 0, 1, 2, \ldots\}$$

2.1.2 Binary Operations

In cryptography, we are interested in three binary operations applied to the set of integers. A binary operation takes two inputs and creates one output.

Figure 2.2 *Three binary operations for the set of integers*



2.1.2 Continued

Example 2.1

The following shows the results of the three binary operations on two integers. Because each input can be either positive or negative, we can have four cases for each operation.

Add:	$5 + 9 = 14$	$(-5) + 9 = 4$	$5 + (-9) = -4$	$(-5) + (-9) = -14$
Subtract:	$5 - 9 = -4$	$(-5) - 9 = -14$	$5 - (-9) = 14$	$(-5) - (-9) = +4$
Multiply:	$5 \times 9 = 45$	$(-5) \times 9 = -45$	$5 \times (-9) = -45$	$(-5) \times (-9) = 45$

2.1.3 Integer Division

In integer arithmetic, if we divide a by n, we can get q And r . The relationship between these four integers can be shown as

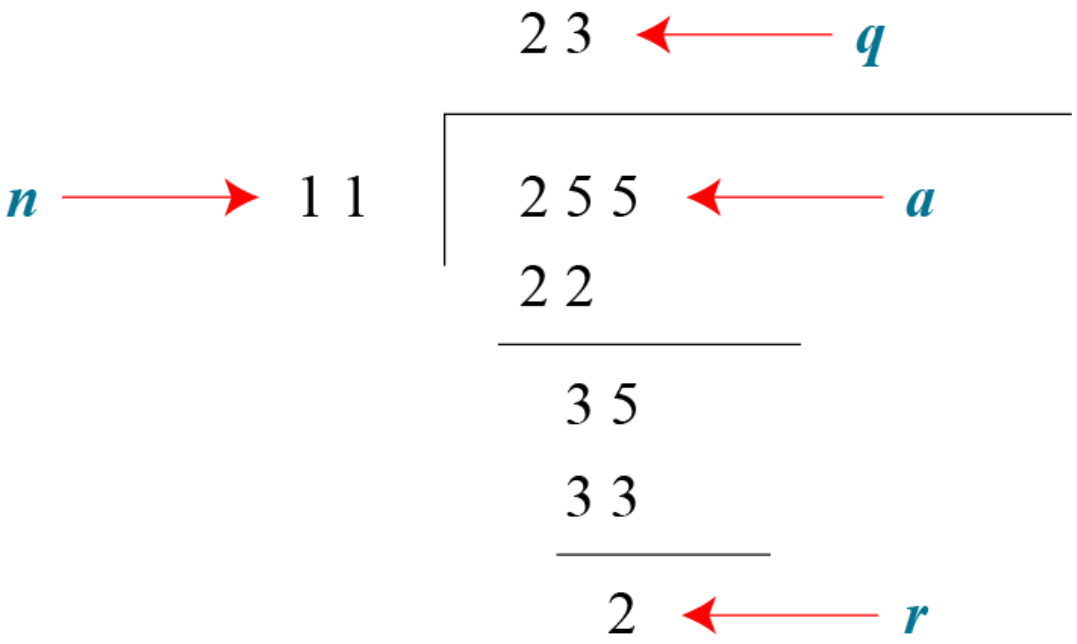
$$a = q \times n + r$$

2.1.3 Continued

Example 2.2

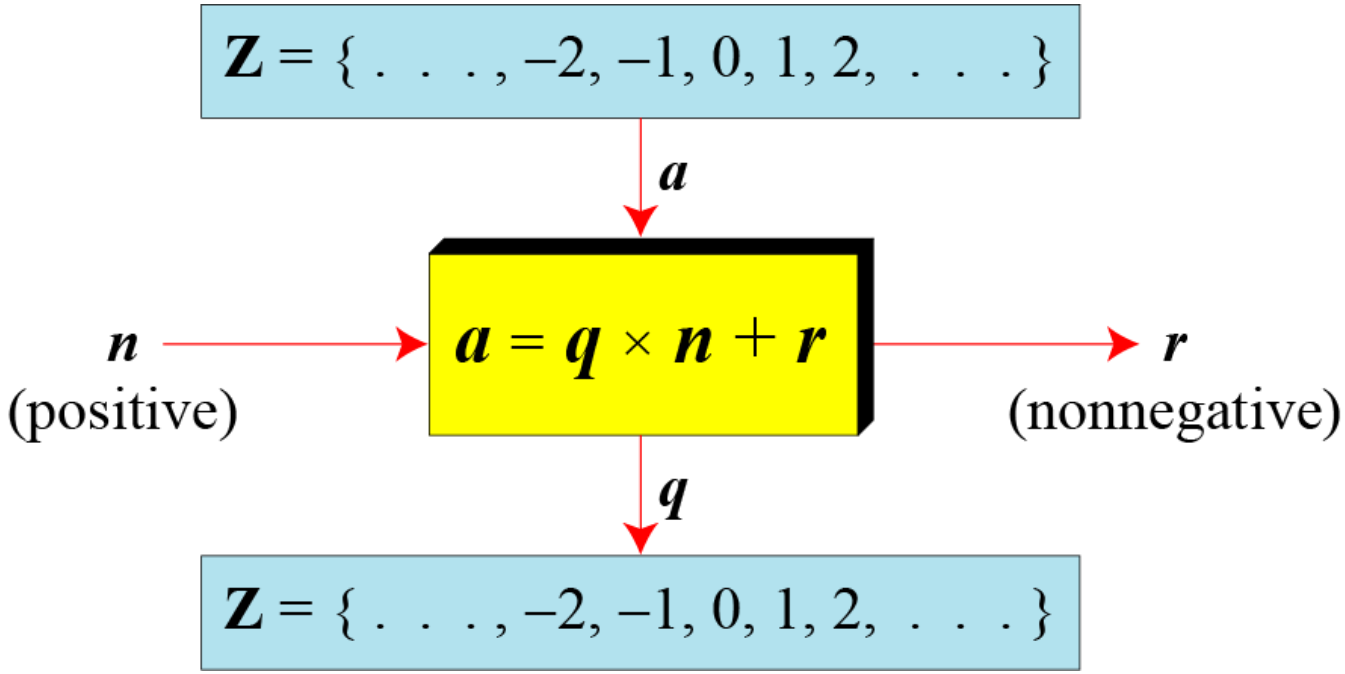
Assume that a = 255 and n = 11. We can find q = 23 and R = 2 using the division algorithm.

Figure 2.3 *Example 2.2, finding the quotient and the remainder*



2.1.3 Continued

Figure 2.4 *Division algorithm for integers*



2.1.3 Continued

Example 2.3

When we use a computer or a calculator, *r* and *q* are negative when *a* is negative. How can we apply the restriction that *r* needs to be positive? The solution is simple, we decrement the value of *q* by 1 and we add the value of *n* to *r* to make it positive.

$$-255 = (-\mathbf{23} \times 11) + (-\mathbf{2}) \quad \leftrightarrow \quad -255 = (-\mathbf{24} \times 11) + \mathbf{9}$$

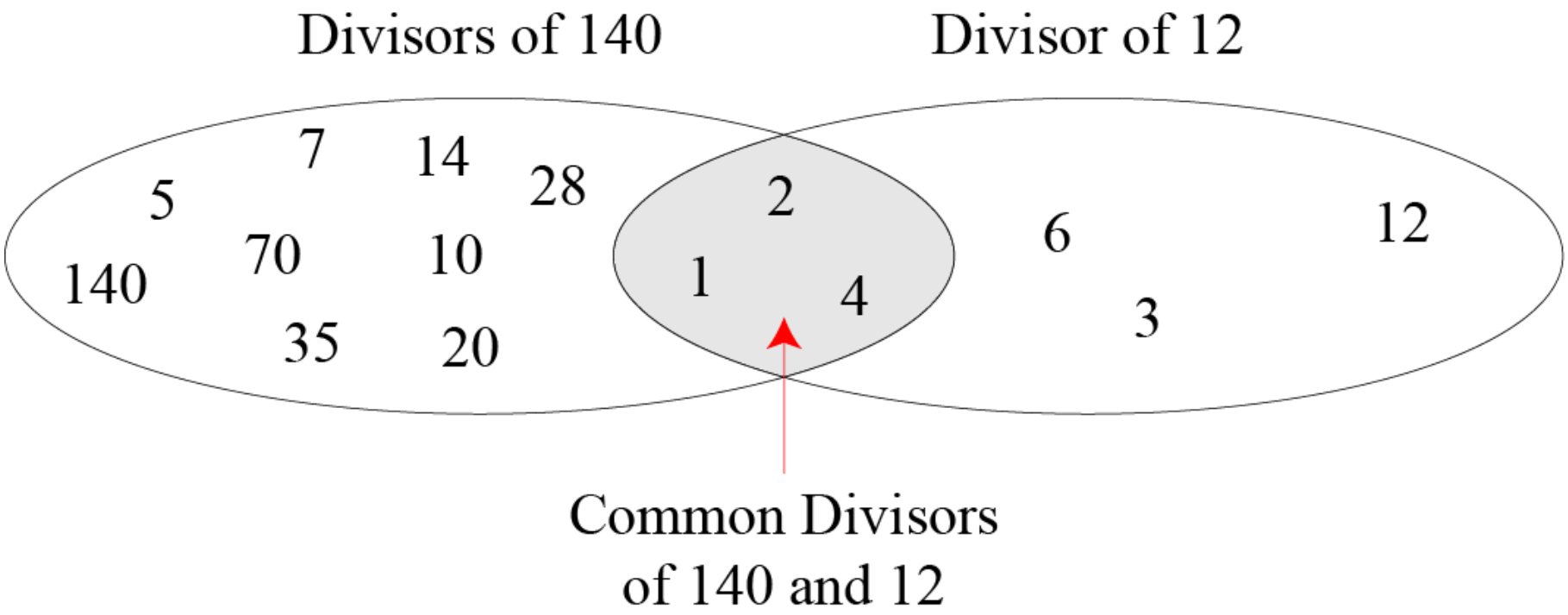
Note

Fact 1: The integer 1 has only one divisor, itself.

Fact 2: Any positive integer has at least two divisors, 1 and itself (but it can have more).

2.1.4 Continued

Figure 2.6 *Common divisors of two integers*



Note **Greatest Common Divisor**

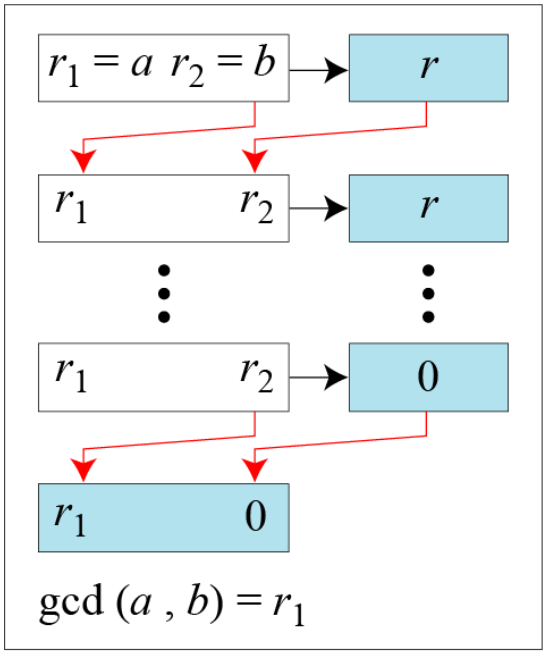
The greatest common divisor of two positive integers is the largest integer that can divide both integers.

Note **Euclidean Algorithm**

Fact 1: gcd (a, 0) = a
Fact 2: gcd (a, b) = gcd (b, r), where r is the remainder of dividing a by b

2.1.4 Continued

Figure 2.7 *Euclidean Algorithm*



a. Process

$r_1 \leftarrow a;$ $r_2 \leftarrow b;$ (Initialization)

while ($r_2 > 0$)

{

$q \leftarrow r_1 / r_2;$

$r \leftarrow r_1 - q \times r_2;$

$r_1 \leftarrow r_2;$ $r_2 \leftarrow r;$

}

gcd (a, b) $\leftarrow r_1$

b. Algorithm

Note

When gcd (a, b) = 1, we say that a and b are relatively prime.

Note

When gcd (a, b) = 1, we say that a and b are relatively prime.

2.1.4 Continued

Example 2.7

Find the greatest common divisor of 2740 and 1760.

Solution

We have gcd (2740, 1760) = 20.

q	r_1	r_2	r
1	2740	1760	980
1	1760	980	780
1	980	780	200
3	780	200	180
1	200	180	20
9	180	20	0
	20	0	

2.1.4 Continued

Example 2.8

Find the greatest common divisor of 25 and 60.

Solution

We have gcd (25, 60) = 5.

q	r_1	r_2	r
0	25	60	25
2	60	25	10
2	25	10	5
2	10	5	0
	5	0	

2.1.4 Continued

Extended Euclidean Algorithm

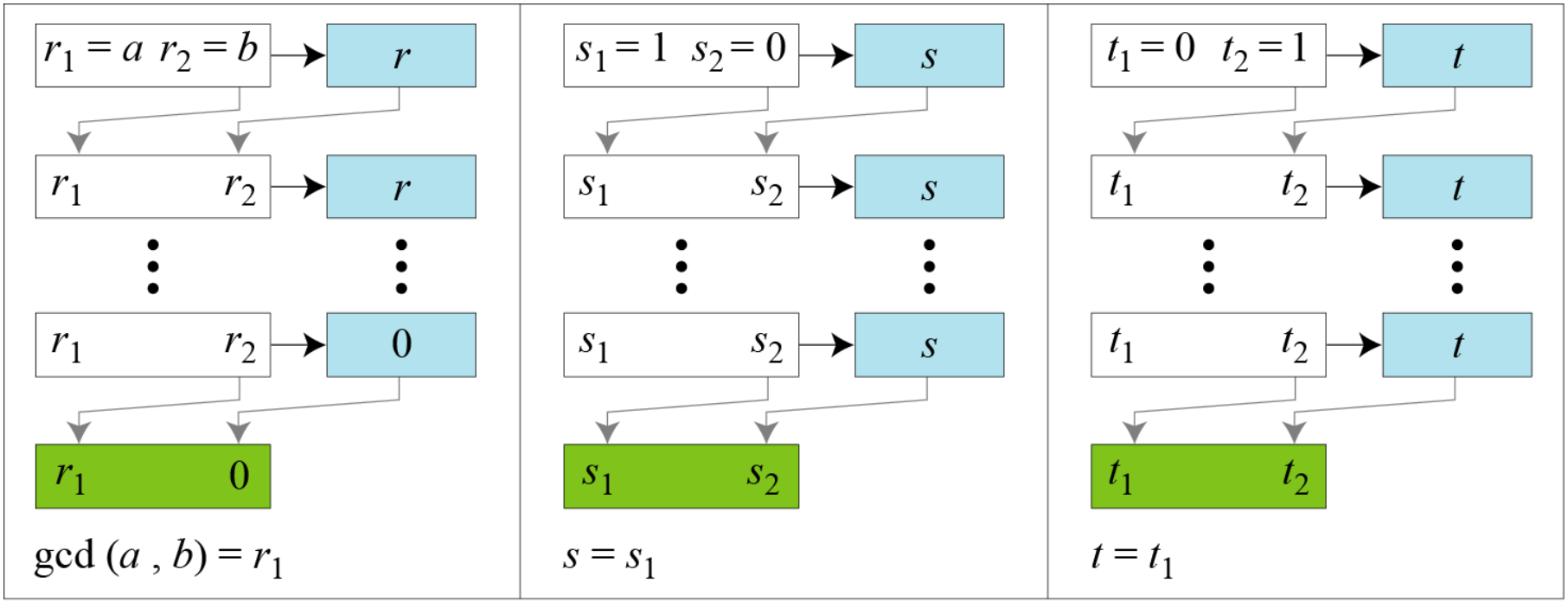
Given two integers *a* and *b*, we often need to find other two integers, *s* and *t*, such that

$$s \times a + t \times b = \gcd(a, b)$$

The extended Euclidean algorithm can calculate the gcd (*a*, *b*) and at the same time calculate the value of *s* and *t*.

2.1.4 Continued

Figure 2.8.a *Extended Euclidean algorithm, part a*



a. Process

2.1.4 Continued

Figure 2.8.b *Extended Euclidean algorithm, part b*

$r_1 \leftarrow a;$ $r_2 \leftarrow b;$
$s_1 \leftarrow 1;$ $s_2 \leftarrow 0;$
$t_1 \leftarrow 0;$ $t_2 \leftarrow 1;$ (Initialization)

while $(r_2 > 0)$

{

$q \leftarrow r_1 / r_2;$

$r \leftarrow r_1 - q \times r_2;$
$r_1 \leftarrow r_2;$ $r_2 \leftarrow r;$ (Updating r's)

$s \leftarrow s_1 - q \times s_2;$
$s_1 \leftarrow s_2;$ $s_2 \leftarrow s;$ (Updating s's)

$t \leftarrow t_1 - q \times t_2;$
$t_1 \leftarrow t_2;$ $t_2 \leftarrow$ t; (Updating t's)

}

gcd $(a, b) \leftarrow r_1;$ $s \leftarrow s_1;$ $t \leftarrow t_1$

b. Algorithm

2.1.4 Continued

Example 2.9

Given $a = 161$ and $b = 28$, find gcd (a, b) and the values of s and t.

Solution

We get gcd (161, 28) = 7, $s = -1$ and $t = 6$.

q	r_1	r_2	r	s_1	s_2	s	t_1	t_2	t
5	161	28	21	1	0	1	0	1	−5
1	28	21	7	0	1	−1	1	−5	6
3	21	7	0	1	−1	4	−5	6	−23
	7	0		**−1**	4		**6**	−23	

2.1.4 Continued

Example 2.10

Given $a = 17$ and $b = 0$, find gcd (a, b) and the values of s and t.

Solution

We get gcd (17, 0) = 17, $s = 1$, and $t = 0$.

q	r_1	r_2	r	s_1	s_2	s	t_1	t_2	t
	17	0		1	0		0	1	

2.1.4 Continued

Example 2.11

Given $a = 0$ and $b = 45$, find gcd (a, b) and the values of s and t.

Solution

We get gcd (0, 45) = 45, $s = 0$, and $t = 1$.

q	r_1	r_2	r	s_1	s_2	s	t_1	t_2	t
0	0	45	0	1	0	1	0	1	0
	45	0		0	1		**1**	0	

2-2 MODULAR ARITHMETIC

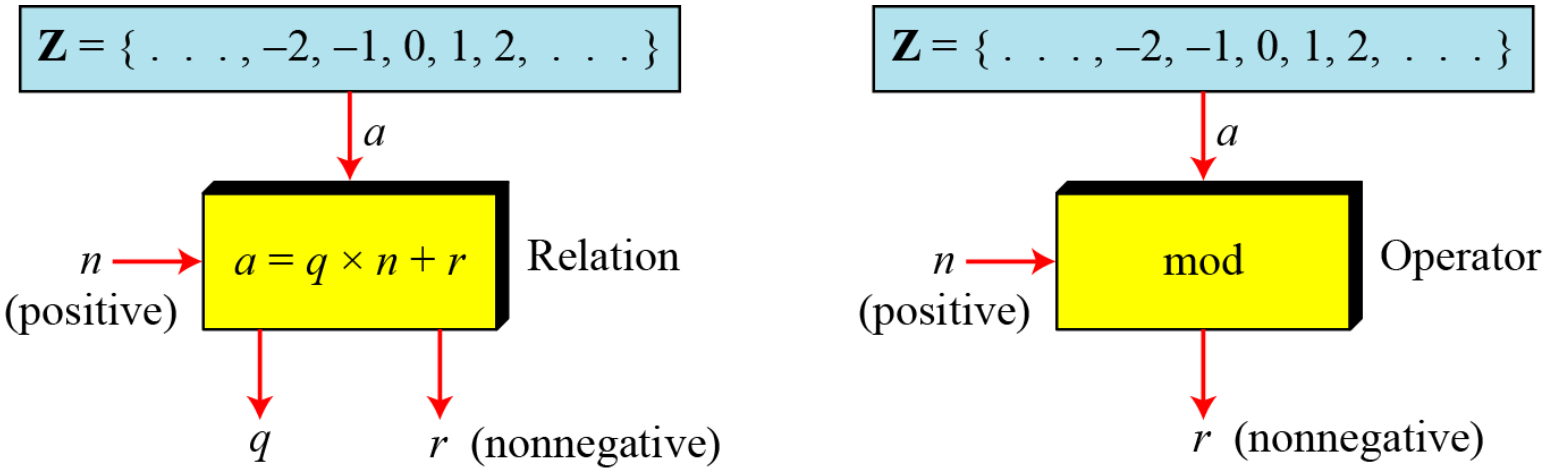
The division relationship ($a = q \times n + r$) discussed in the previous section has two inputs (a and n) and two outputs (q and r). In modular arithmetic, we are interested in only one of the outputs, the remainder r.

Topics discussed in this section:

2.2.1 Modulo Operator

The modulo operator is shown as mod. The second input (n) is called the modulus. The output r is called the residue.

Figure 2.9 *Division algorithm and modulo operator*



Example 2.14

Find the result of the following operations:

a. 27 mod 5

b. 36 mod 12

c. −18 mod 14

d. −7 mod 10

Solution

a. Dividing 27 by 5 results in $r = 2$

b. Dividing 36 by 12 results in $r = 0$.

c. Dividing −18 by 14 results in $r = -4$. After adding the modulus $r = 10$

d. Dividing −7 by 10 results in r = −7. After adding the modulus to −7, r = 3.

2.2.2 Set of Residues

The modulo operation creates a set, which in modular arithmetic is referred to as the set of least residues modulo n, or Z_n.

Figure 2.10 *Some Z_n sets*

$$\mathbf{Z}_n = \{ 0, 1, 2, 3, \ldots, (n-1) \}$$

$$\mathbf{Z}_2 = \{ 0, 1 \}$$

$$\mathbf{Z}_6 = \{ 0, 1, 2, 3, 4, 5 \}$$

$$\mathbf{Z}_{11} = \{ 0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10 \}$$

2.2.3 Congruence

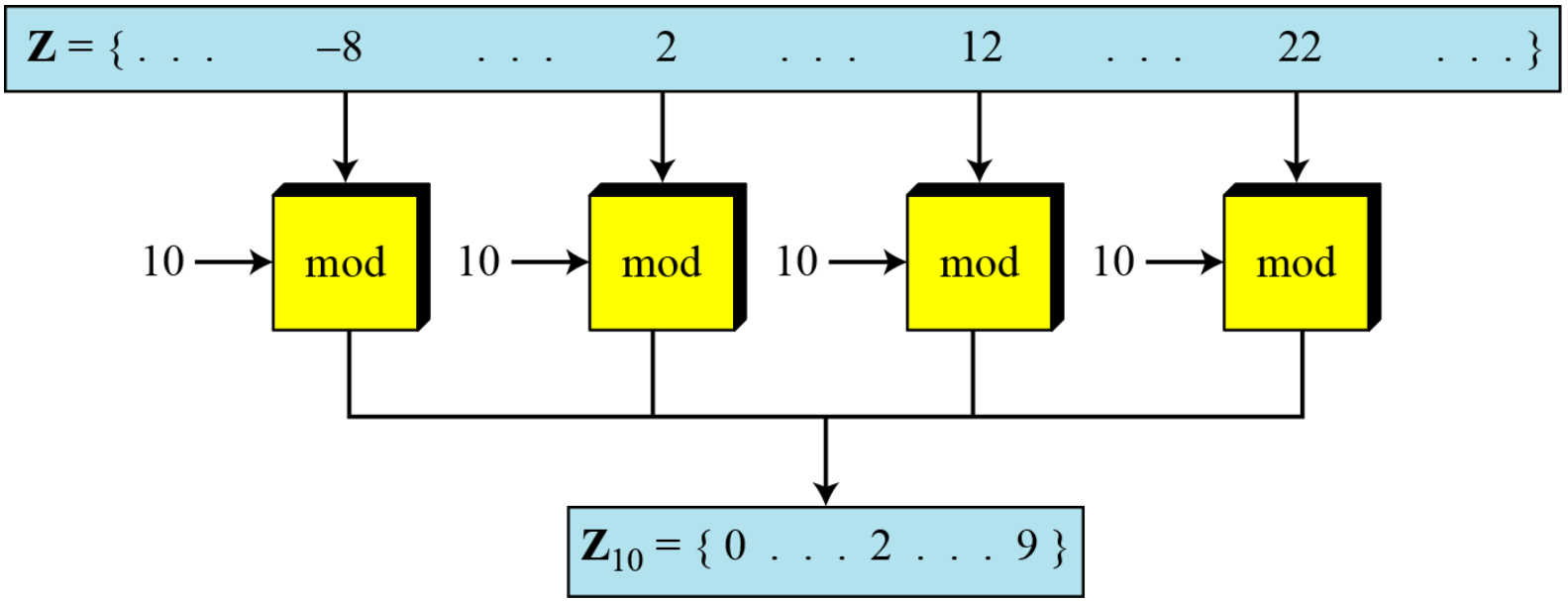
To show that two integers are congruent, we use the congruence operator (≡). For example, we write:

$$2 \equiv 12 \pmod{10} \qquad 13 \equiv 23 \pmod{10}$$

$$3 \equiv 8 \pmod{5} \qquad 8 \equiv 13 \pmod{5}$$

2.2.3 Continued

Figure 2.11 *Concept of congruence*



$$-8 \equiv 2 \equiv 12 \equiv 22 \pmod{10}$$

Congruence Relationship

2.2.3 Continued

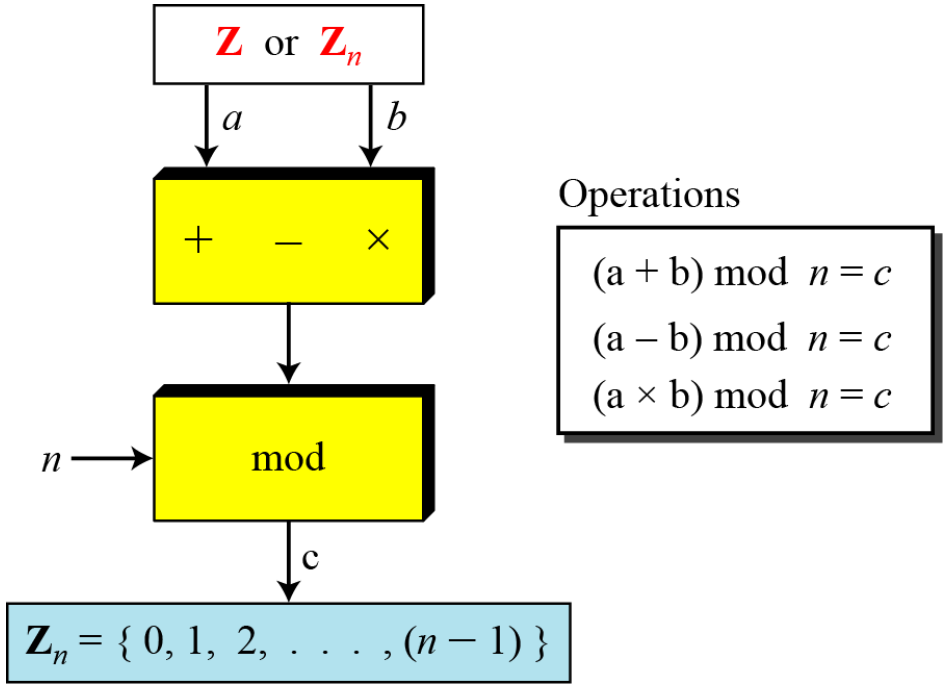
Example 2.15

We use modular arithmetic in our daily life; for example, we use a clock to measure time. Our clock system uses modulo 12 arithmetic. However, instead of a 0 we use the number 12.

2.2.4 Operation in Z_n

The three binary operations that we discussed for the set Z can also be defined for the set Zn. The result may need to be mapped to Zn using the mod operator.

Figure 2.13 *Binary operations in Z_n*



2.2.4 Continued

Example 2.16

Perform the following operations (the inputs come from Zn):

a. Add 7 to 14 in Z15.

b. Subtract 11 from 7 in Z13.

c. Multiply 11 by 7 in Z20.

Solution

$$(14 + 7) \bmod 15 \rightarrow (21) \bmod 15 = 6$$

$$(7 - 11) \bmod 13 \rightarrow (-4) \bmod 13 = 9$$

$$(7 \times 11) \bmod 20 \rightarrow (77) \bmod 20 = 17$$

2.2.4 Continued

Example 2.17

Perform the following operations (the inputs come from either Z or Z_n):

a. Add 17 to 27 in Z_{14}.

b. Subtract 43 from 12 in Z_{13}.

c. Multiply 123 by −10 in Z_{19}.

Solution

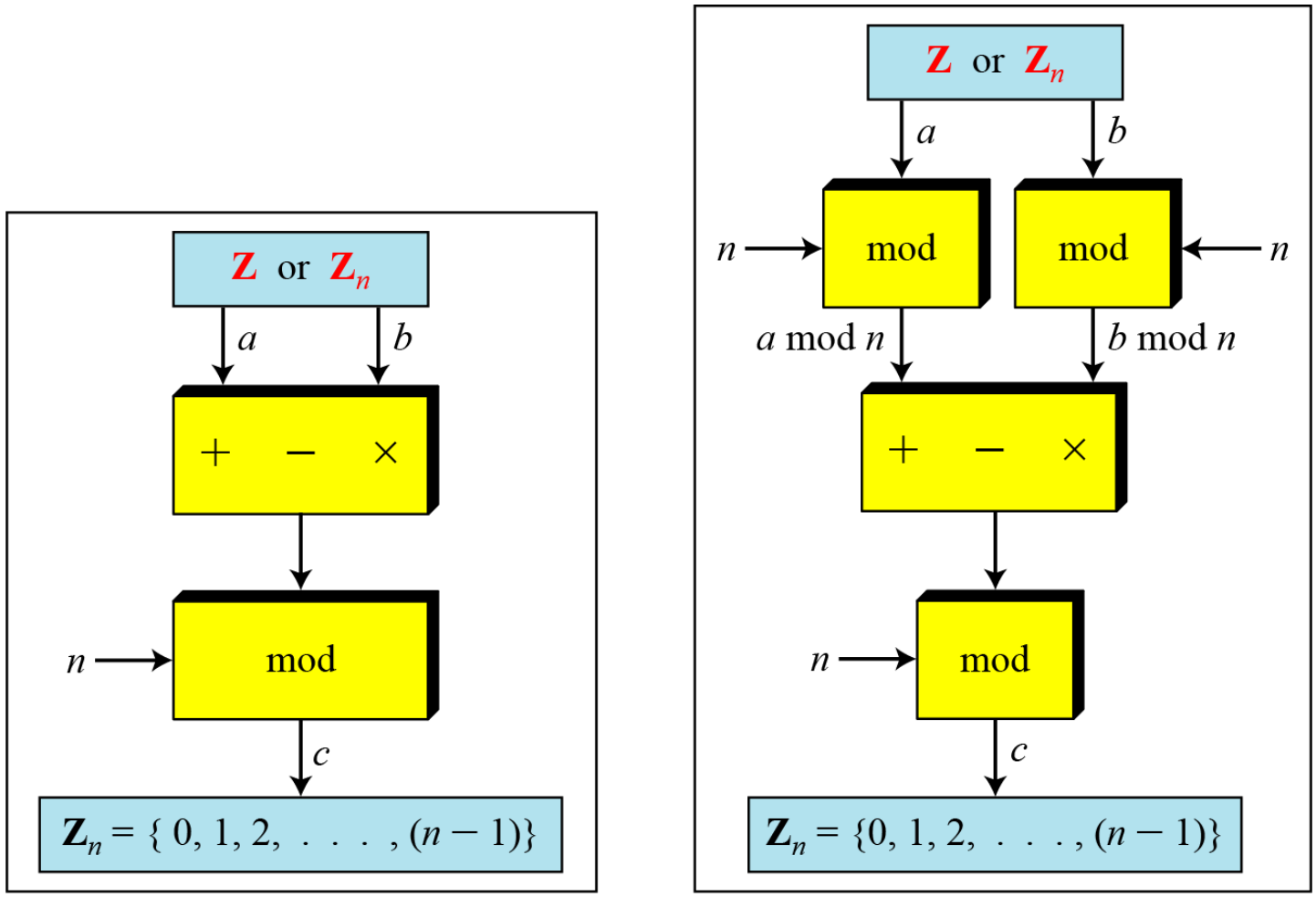
2.2.4 Continued

Properties

First Property:	$(a + b) \bmod n = [(a \bmod n) + (b \bmod n)] \bmod n$
Second Property:	$(a - b) \bmod n = [(a \bmod n) - (b \bmod n)] \bmod n$
Third Property:	$(a \times b) \bmod n = [(a \bmod n) \times (b \bmod n)] \bmod n$

2.2.4 Continued

Figure 2.14 *Properties of mode operator*



a. Original process

b. Applying properties

2.2.4 Continued

Example 2.18

The following shows the application of the above properties:

1. (1,723,345 + 2,124,945) mod 11 = (8 + 9) mod 11 = 6

2. (1,723,345 − 2,124,945) mod 16 = (8 − 9) mod 11 = 10

3. (1,723,345 × 2,124,945) mod 16 = (8 × 9) mod 11 = 6

2.2.5 Continue

Multiplicative Inverse

In Z_n, two numbers *a* and *b* are the multiplicative inverse of each other if

$$a \times b \equiv 1 \pmod{n}$$

Note

In modular arithmetic, an integer may or may not have a multiplicative inverse. When it does, the product of the integer and its multiplicative inverse is congruent to 1 modulo n.

2.2.2 Set of Residues

The modulo operation creates a set, which in modular arithmetic is referred to as the set of least residues modulo n, or Z_n.

Figure 2.10 *Some Z_n sets*

$$\mathbf{Z}_n = \{ 0, 1, 2, 3, \ldots, (n-1) \}$$

$$\mathbf{Z}_2 = \{ 0, 1 \}$$

$$\mathbf{Z}_6 = \{ 0, 1, 2, 3, 4, 5 \}$$

$$\mathbf{Z}_{11} = \{ 0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10 \}$$

Example 2.22

Find the multiplicative inverse of 8 in Z_{10}.

Solution

There is no multiplicative inverse because gcd (10, 8) = 2 ≠ 1. In other words, we cannot find any number between 0 and 9 such that when multiplied by 8, the result is congruent to 1.

Example 2.23

Find all multiplicative inverses in Z_{10}.

Solution

There are only three pairs: (1, 1), (3, 7) and (9, 9). The numbers 0, 2, 4, 5, 6, and 8 do not have a multiplicative inverse.

2.2.5 Continued

Example 2.24

Find all multiplicative inverse pairs in Z_{11}.

Solution

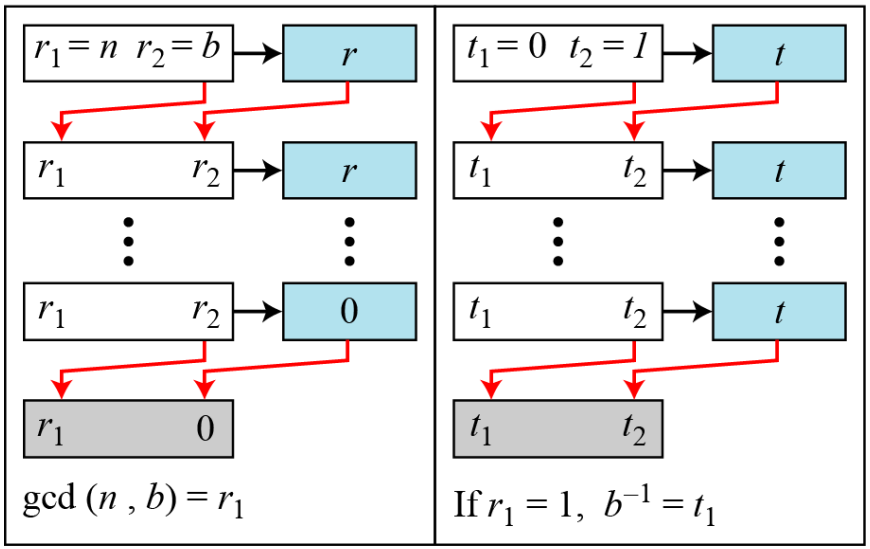
We have seven pairs: (1, 1), (2, 6), (3, 4), (5, 9), (7, 8), (9, 9), and (10, 10).

Note

The extended Euclidean algorithm finds the multiplicative inverses of b in Z_n when n and b are given and gcd (n, b) = 1. The multiplicative inverse of b is the value of t after being mapped to Z_n.

2.2.5 Continued

Figure 2.15 *Using extended Euclidean algorithm to find multiplicative inverse*



a. Process

$r_1 \leftarrow$ n; $r_2 \leftarrow b$;
$t_1 \leftarrow 0$; $t_2 \leftarrow 1$;

while $(r_2 > 0)$
{
$q \leftarrow r_1 / r_2$;

$r \leftarrow r_1 - q \times r_2$;
$r_1 \leftarrow r_2$; $r_2 \leftarrow r$;

$t \leftarrow t_1 - q \times t_2$;
$t_1 \leftarrow t_2$; $t_2 \leftarrow t$;

}
if $(r_1 = 1)$ then $b^{-1} \leftarrow t_1$

b. Algorithm

2.2.5 Continued

Example 2.25

Find the multiplicative inverse of 11 in Z_{26}.

Find the multiplicative inverse of 11 in Z_{26}.

Solution

q	r_1	r_2	r	t_1	t_2	t
2	26	11	4	0	1	–2
2	11	4	3	1	–2	5
1	4	3	1	–2	5	–7
3	3	1	0	5	–7	26
	1	0		–7	26	

The gcd (26, 11) is 1; the inverse of 11 is –7 or 19.

2.2.5 Continued

Example 2.26

Find the multiplicative inverse of 23 in Z_{100}.

2.2.5 Continued

Example 2.26

Find the multiplicative inverse of 23 in Z_{100}.

Solution

q	r_1	r_2	r	t_1	t_2	t
4	100	23	8	0	1	−4
2	23	8	7	1	−4	19
1	8	7	1	−4	9	−13
7	7	1	0	9	−13	100
	1	0		−13	100	

The gcd (100, 23) is 1; the inverse of 23 is −13 or 87.

2.2.5 Continued

Example 2.27

Find the inverse of 12 in Z_{26}.

Solution

q	r_1	r_2	r	t_1	t_2	t
2	26	12	2	0	1	–2
6	12	2	0	1	–2	13
	2	0		–2	13	

Module-III

Public Key Cryptography and RSA

Private-Key Cryptography

- traditional **private/secret/single key** cryptography uses **one** key
- shared by both sender and receiver
- if this key is disclosed communications are compromised
- also is **symmetric**, parties are equal
- hence does not protect sender from receiver forging a message & claiming is sent by sender

Public-Key Cryptography

- probably most significant advance in the 3000 year history of cryptography
- uses **two** keys – a public & a private key
- **asymmetric** since parties are **not** equal
- uses clever application of number theoretic concepts to function
- complements **rather than** replaces private key crypto

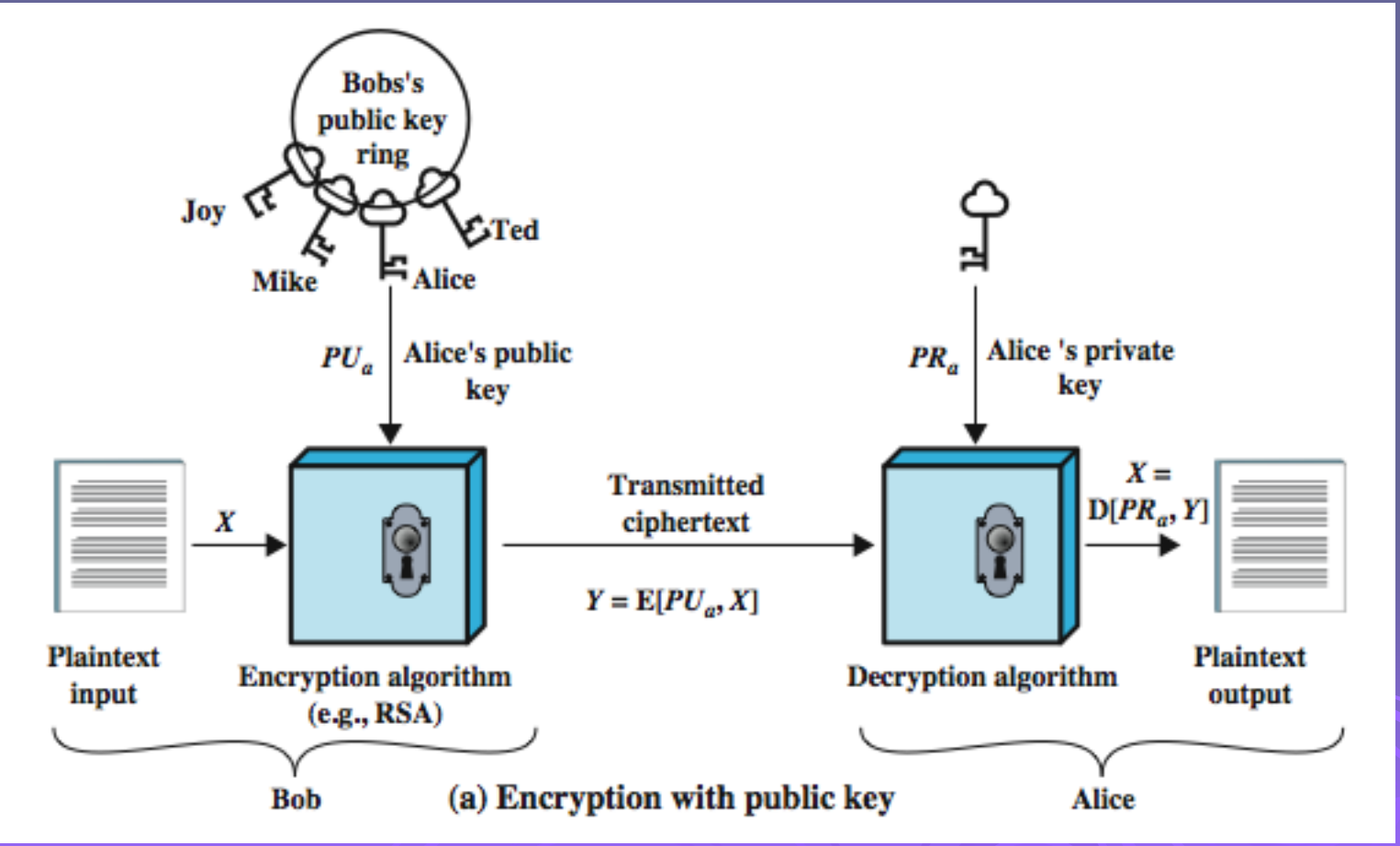
Why Public-Key Cryptography?

- developed to address two key issues:
  - **key distribution** – how to have secure communications in general without having to trust a KDC with your key
  - **digital signatures** – how to verify a message comes intact from the claimed sender
- public invention due to Whitfield Diffie & Martin Hellman at Stanford Uni in 1976
  - known earlier in classified community

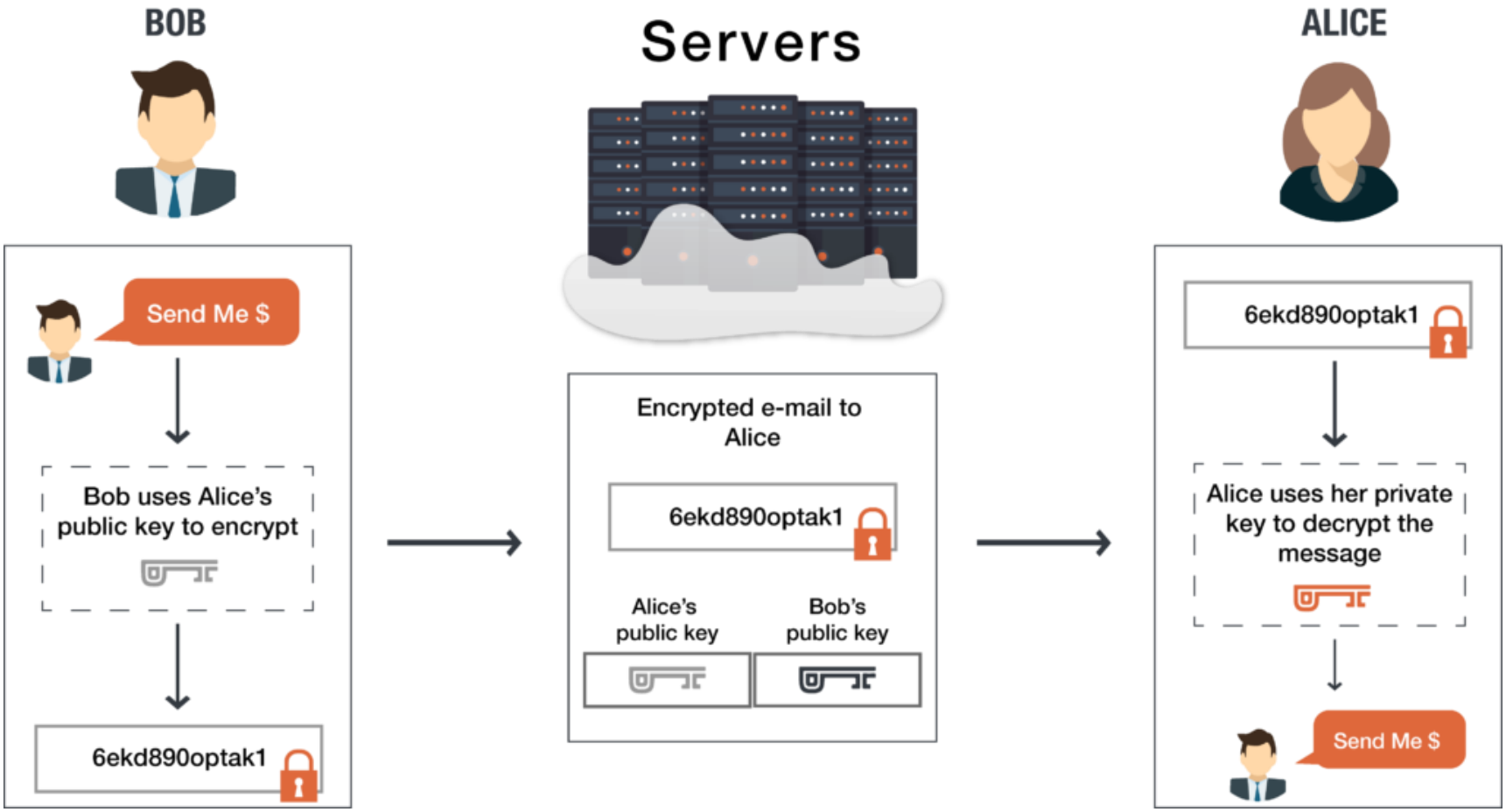
Public-Key Cryptography

- **public-key/two-key/asymmetric** cryptography involves the use of **two** keys:
  - a **public-key**, which may be known by anybody, and can be used to **encrypt messages**, and **verify signatures**
  - a related **private-key**, known only to the recipient, used to **decrypt messages**, and **sign** (create) **signatures**
- **infeasible to determine private key from public**
- is **asymmetric** because
  - those who encrypt messages or verify signatures **cannot** decrypt messages or create signatures

Public-Key Cryptography



Asymmetric key cryptography



Public-Key Applications

- can classify uses into 3 categories:
  - **encryption/decryption** (provide secrecy)
  - **digital signatures** (provide authentication)
  - **key exchange** (of session keys)
- some algorithms are suitable for all uses, others are specific to one

Algorithm	Encryption/Decryption	Digital Signature	Key Exchange
RSA	Yes	Yes	Yes
Elliptic Curve	Yes	Yes	Yes
Diffie-Hellman	No	No	Yes
DSS	No	Yes	No

Public-Key Requirements

- Public-Key algorithms rely on two keys where:
  - it is computationally infeasible to find decryption key knowing only algorithm & encryption key
  - it is computationally easy to en/decrypt messages when the relevant (en/decrypt) key is known
  - either of the two related keys can be used for encryption, with the other used for decryption (for some algorithms)

RSA

- by Rivest, Shamir & Adleman of MIT in 1977
- best known & widely used public-key scheme
- uses large integers (eg. 1024 bits)
- security due to cost of factoring large numbers
  - factorization takes $O(e^{\log n \log \log n})$ operations (hard)

RSA En/decryption

- to encrypt a message M the sender:
  - obtains **public key** of recipient $PU=\{e,n\}$
  - computes: $C = M^e \bmod n$, where $0 \leq M < n$
- to decrypt the ciphertext C the owner:
  - uses their private key $PR=\{d,n\}$
  - computes: $M = C^d \bmod n$
- note that the message M must be smaller than the modulus n (block if needed)

RSA Key Setup

- each user generates a public/private key pair by:
- selecting two large primes at random: p, q
- computing their system modulus $n=p.q$
  - note $\phi(n)=(p-1)(q-1)$
- selecting at random the encryption key e
  - where $1<e<\phi(n), \gcd(e,\phi(n))=1$
- solve following equation to find decryption key d
  - $e.d=1 \bmod \phi(n)$ and $0 \le d \le n$
- publish their public encryption key: PU={e,n}
- keep secret private decryption key: PR={d,n}

RSA Example - Key Setup

1. Select primes: $p=17$ & $q=11$
2. Calculate $n = pq = 17 \times 11 = 187$
3. Calculate $\phi(n) = (p-1)(q-1) = 16 \times 10 = 160$
4. Select e: $\gcd(e,160)=1$; choose $e=7$
5. Determine d: $de=1 \bmod 160$ and $d < 160$
   Value is $d=23$ since $23 \times 7 = 161 = 10 \times 160 + 1$
6. Publish public key $PU=\{7,187\}$
7. Keep secret private key $PR=\{23,187\}$

RSA Example - En/Decryption

- sample RSA encryption/decryption is:
- given message $M = 88$ (nb. $88<187$)
- encryption:

  $C = 88^7 \bmod 187 = 11$
- decryption:

  $M = 11^{23} \bmod 187 = 88$

RSA Key Generation

- users of RSA must:
  - determine two primes at random - `p, q`
  - select either `e` or `d` and compute the other
- primes `p,q` must not be easily derived from modulus `n=p.q`
  - means must be sufficiently large
  - typically guess and use probabilistic test
- exponents `e, d` are inverses, so use Inverse algorithm to compute the other

Progress in Factoring

Number of Decimal Digits	Approximate Number of Bits	Date Achieved	MIPS-years	Algorithm
100	332	April 1991	7	quadratic sieve
110	365	April 1992	75	quadratic sieve
120	398	June 1993	830	quadratic sieve
129	428	April 1994	5000	quadratic sieve
130	431	April 1996	1000	generalized number field sieve
140	465	February 1999	2000	generalized number field sieve
155	512	August 1999	8000	generalized number field sieve
160	530	April 2003	—	Lattice sieve
174	576	December 2003	—	Lattice sieve
200	663	May 2005	—	Lattice sieve

Summary

- have considered:
  - principles of public-key cryptography
  - RSA algorithm, implementation, security

Key Management

Diffie-Hellman Key Exchange Algorithm

by

M.K.Chavan

Asst. Professor

Dept. of Computer Science

The Problem of Key Exchange

- One of the main problems of symmetric key encryption is it requires a secure & reliable channel for the shared key exchange.

- The Diffie-Hellman Key Exchange protocol offers a way in which a public channel can be used to create a confidential shared key.

Modular what?

- In practice the shared encryption key relies on such complex concepts as *Modular Exponentiation*, *Primitive Roots* and *Discrete Logarithm Problems*.

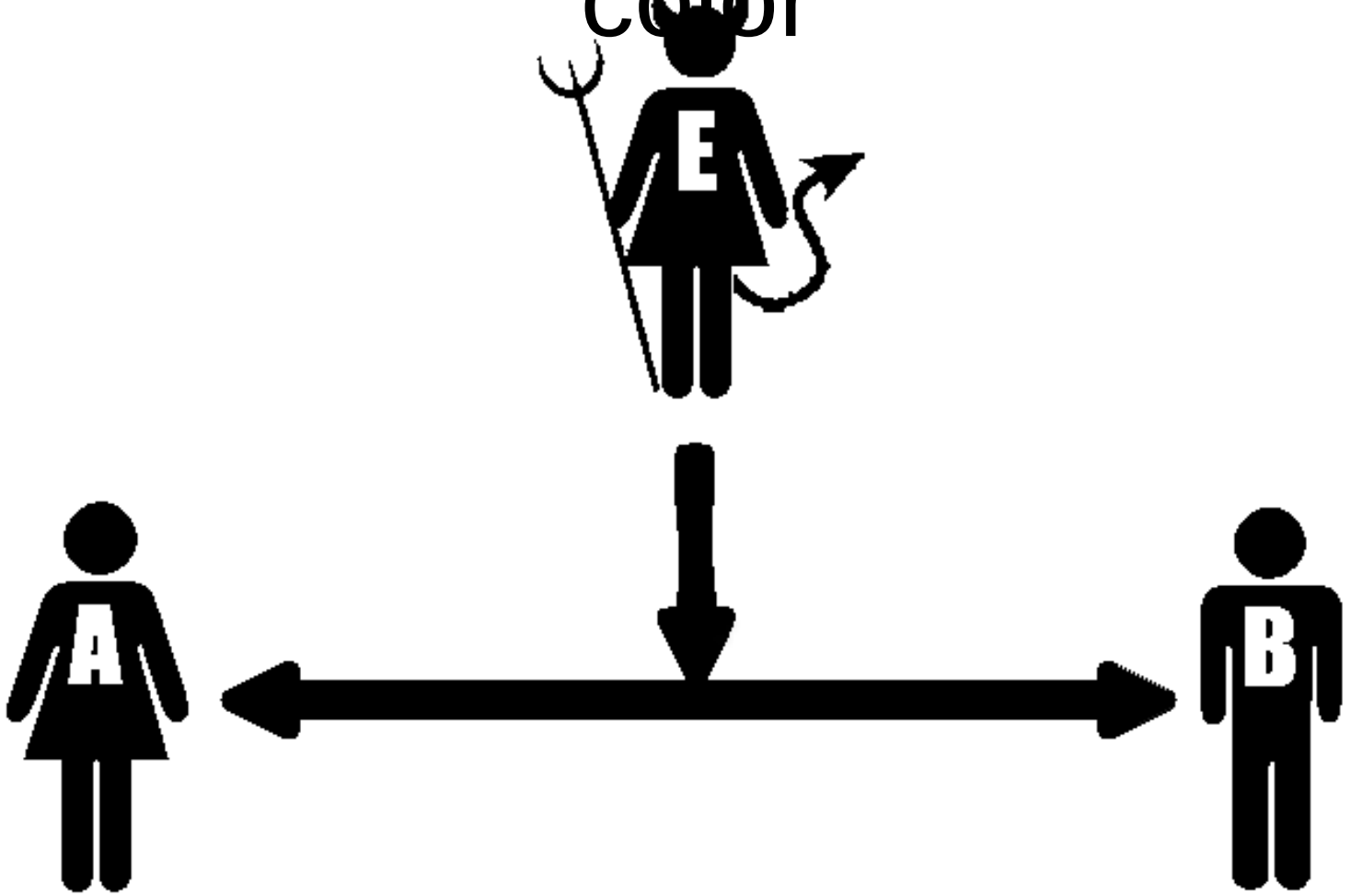
- Let's see though is we can explain the Diffie-Hellman algorithm with no complex mathematics.

A Difficult One-Way Problem

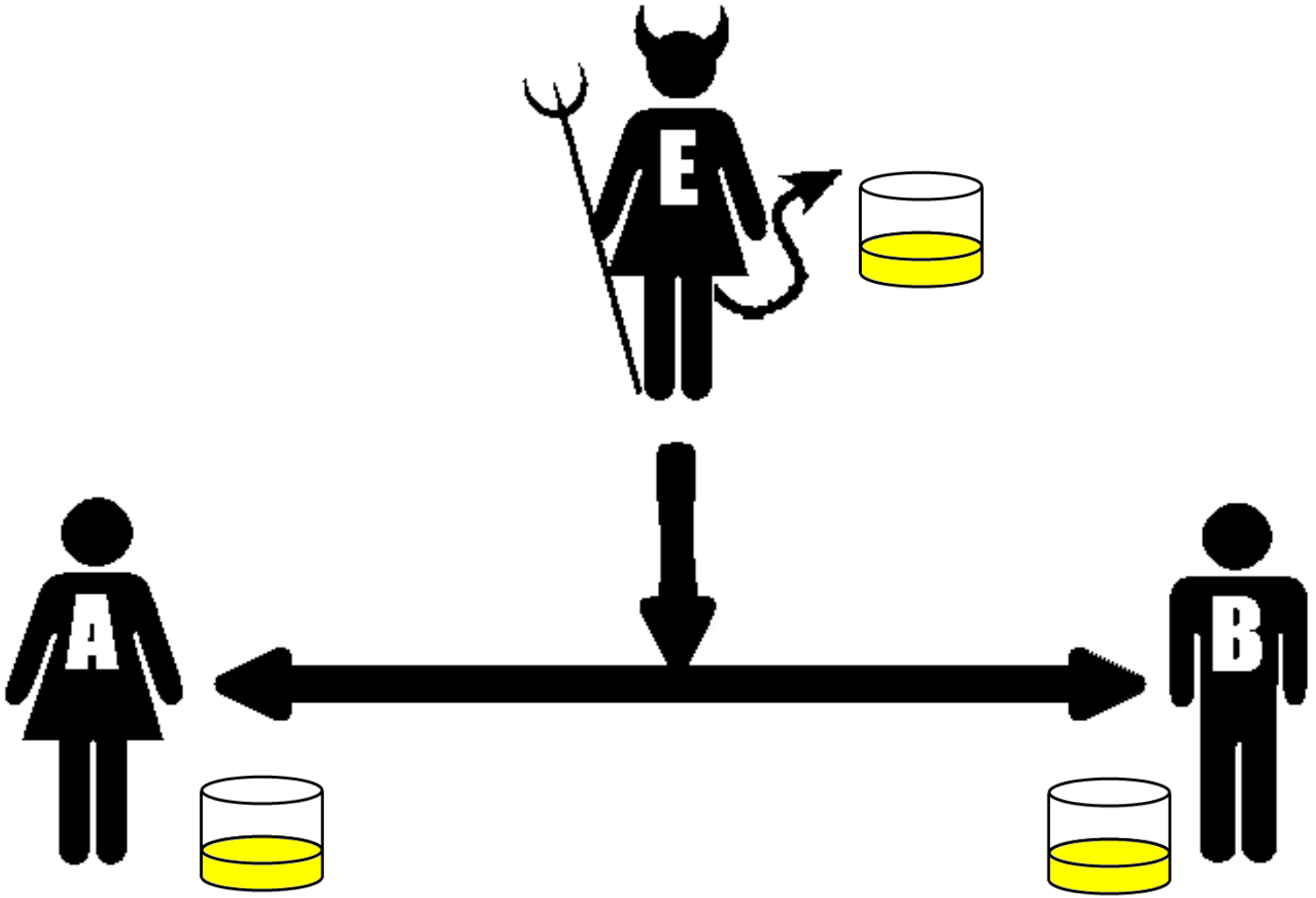
- The first thing we require is a simple real-world operation that is easy to ***Do*** but hard to ***Undo***.
  - You can ring a bell but not unring one.
  - Toothpaste is easy to squeeze out of a tube but famously hard to put back in.

- In our example we will use *Mixing Colors*.
  - Easy to mix 2 colors, hard to unmix

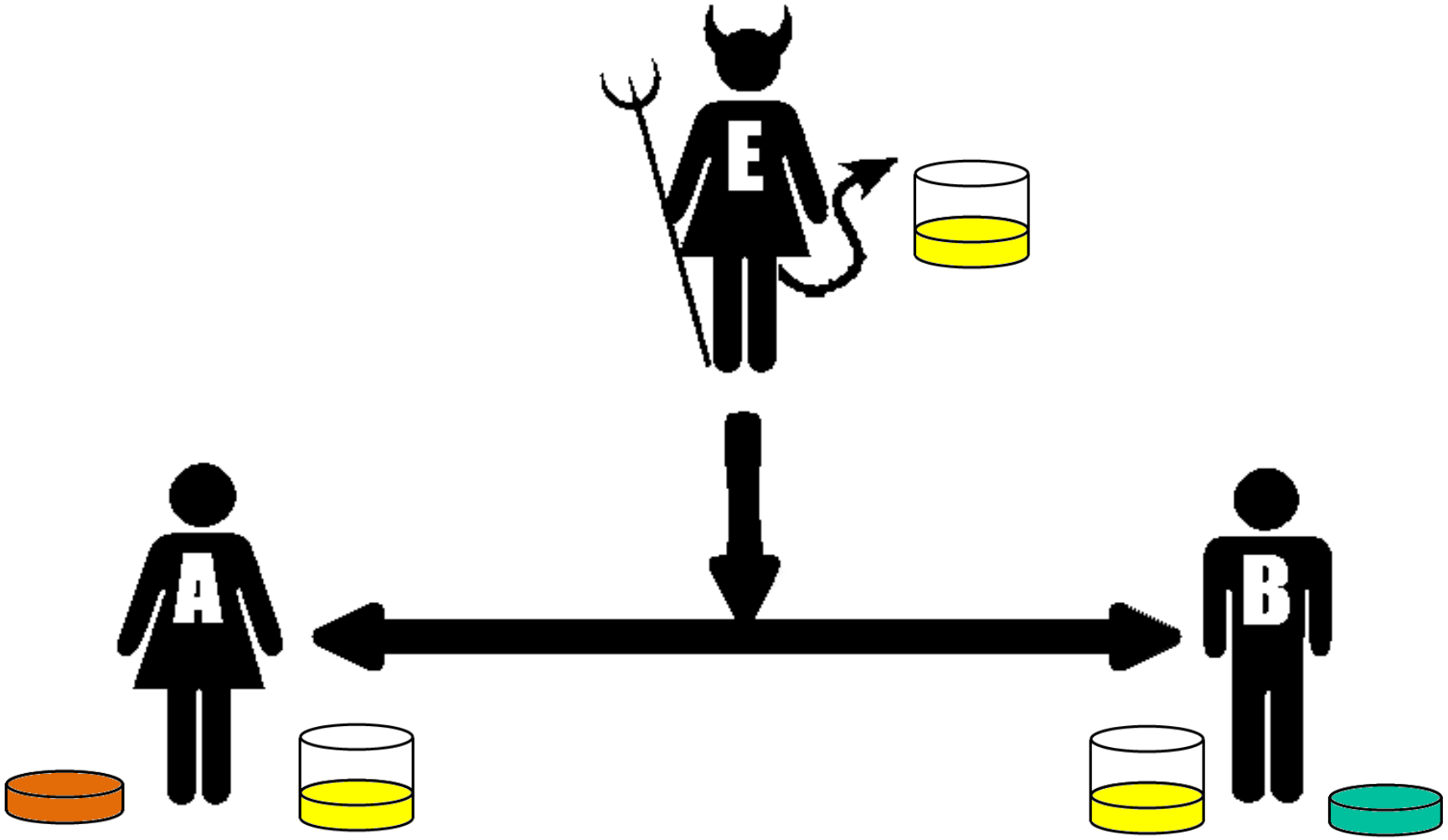
Alice & Bob with Eve listening wish to make a secret shared color



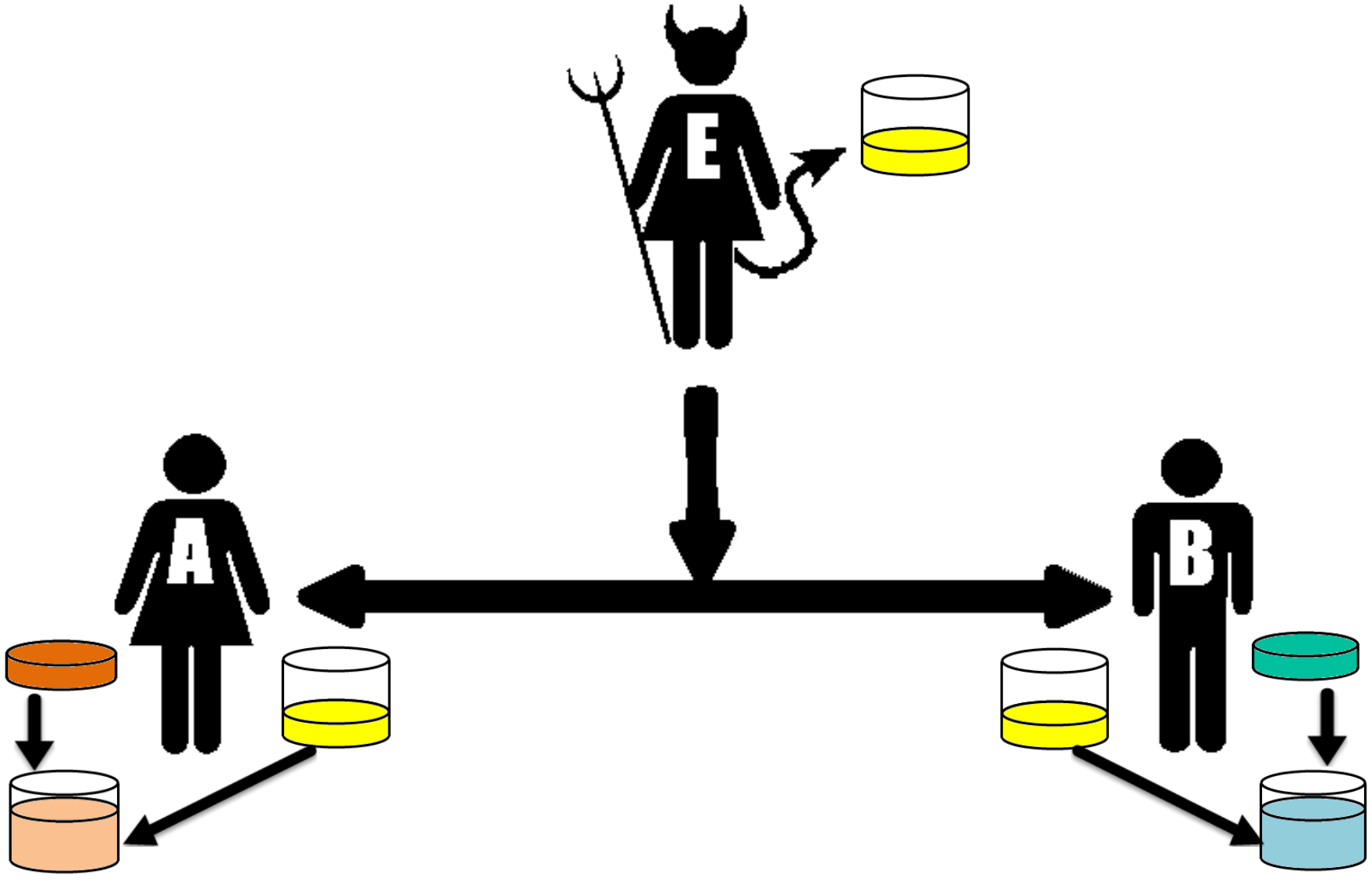
Step 1 - Both publicly agree to a shared color



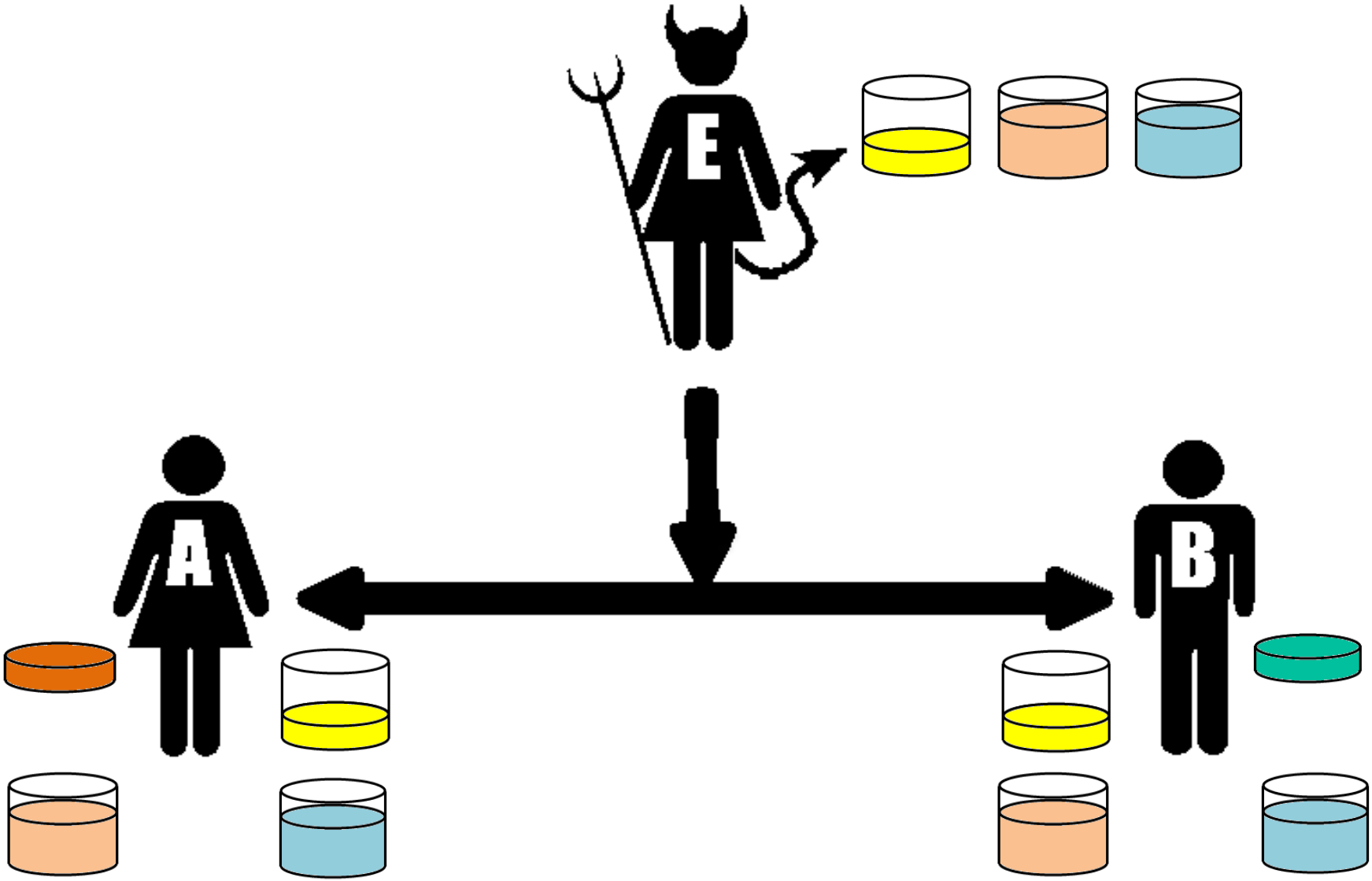
Step 2 - Each picks a secret color



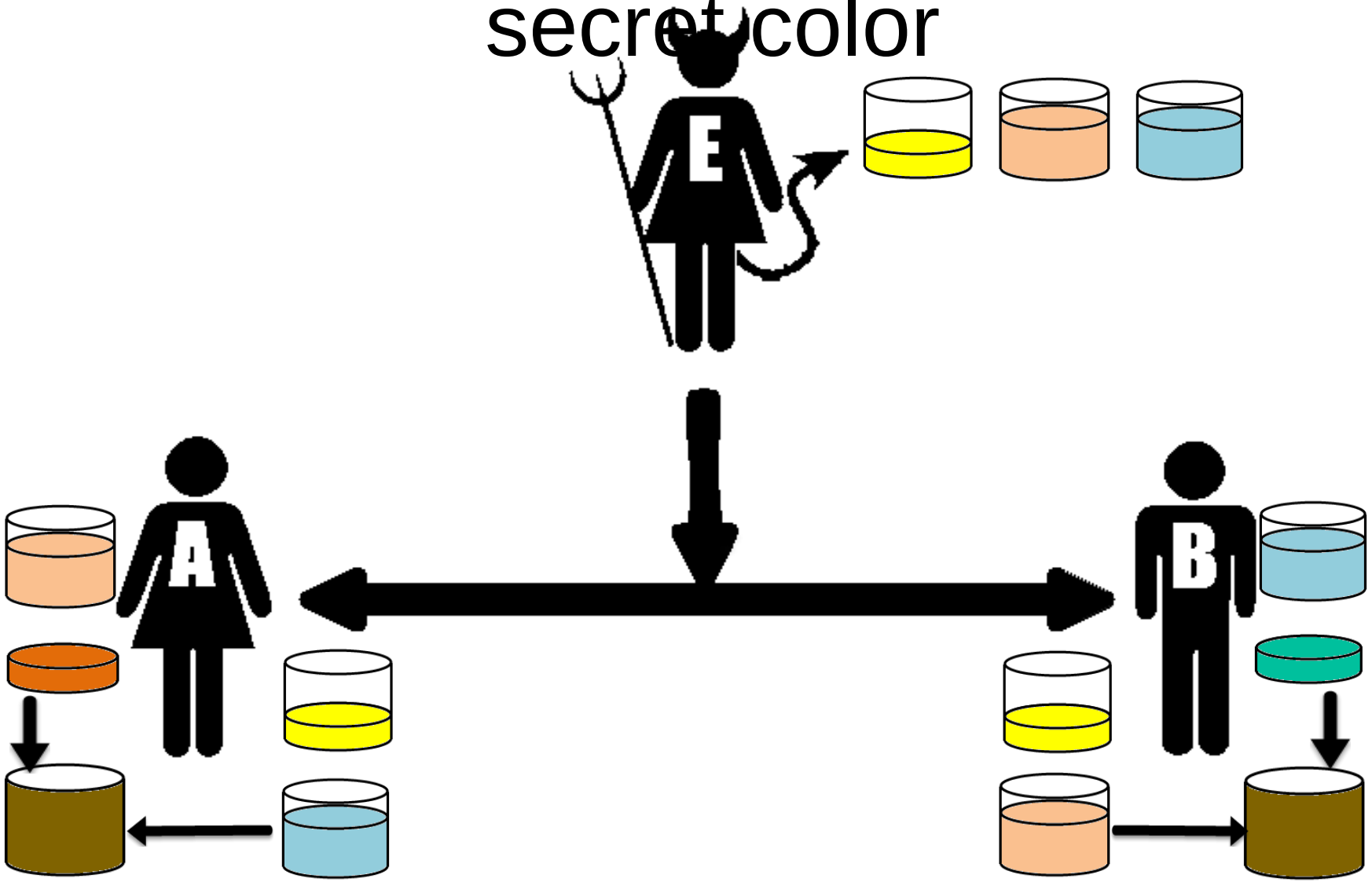
Step 3 - Each adds their secret color to the shared color



Step 4 - Each sends the other their new mixed color



Each combines the shared color from the other with their own secret color



Alice & Bob have agreed to a shared color unknown to Eve

- How is it that Alice & Bob's final mixtures are identical?

- Alice mixed
  - [(Yellow + Teal) $_{\text{from Bob}}$] + Orange

- Bob mixed
  - [(Yellow + Orange) $_{\text{from Alice}}$] + Teal

Alice & Bob have agreed to a shared color unknown to Eve

- How is it that Alice & Bob's final mixture is secret?
- Eve never has knowledge of the secret colors of either Alice or Bob
- Unmixing a color into its component colors is a hard problem

Alice & Bob have agreed to a shared color unknown to Eve

- How is it that Alice & Bob's final mixture is secret?
- Eve never has knowledge of the secret colors of either Alice or Bob
- Unmixing a color into its component colors is a hard problem

Diffie-Hellman Key Exchange

- first public-key type scheme proposed
  - For key distribution only
- by Diffie & Hellman in 1976 along with the exposition of public key concepts
  - note: now know that James Ellis (UK CESG) secretly proposed the concept in 1970
- is a practical method for public exchange of a secret key
- used in a number of commercial products

Diffie-Hellman Setup

- all users agree on global parameters:
  - large prime integer or polynomial q
  - α a primitive root mod q
- each user (eg. A) generates their key
  - chooses a secret key (number): $x_A < q$
  - compute their **public key**: $y_A = \alpha^{x_A} \bmod q$
- each user makes public that key y_A

Diffie-Hellman Key Exchange

- shared session key for users A & B is K:

  $K = y_A^{x_B} \bmod q$ (which **B** can compute)

  $K = y_B^{x_A} \bmod q$ (which **A** can compute)

  (example)

- K is used as session key in private-key encryption scheme between Alice and Bob
- if Alice and Bob subsequently communicate, they will have the **same** key as before, unless they choose new public-keys
- attacker needs an x, must solve discrete log

Diffie-Hellman Example

- users Alice & Bob who wish to swap keys:
- agree on prime $q=353$ and $\alpha=3$
- select random secret keys:
  - A chooses $x_A=97$, B chooses $x_B=233$
- compute public keys:
  - $y_A=3^{97} \bmod 353 = 40$ (Alice)
  - $y_B=3^{233} \bmod 353 = 248$ (Bob)
- compute shared session key as:

$K_{AB}= y_B^{x_A} \bmod 353 = 248^{97} = 160$ (Alice)

$K_{AB}= y_A^{x_B} \bmod 353 = 40^{233} = 160$ (Bob)

Reference

- Cryptography and Network Security Principles and Practices, William Stallings,4th Edition.