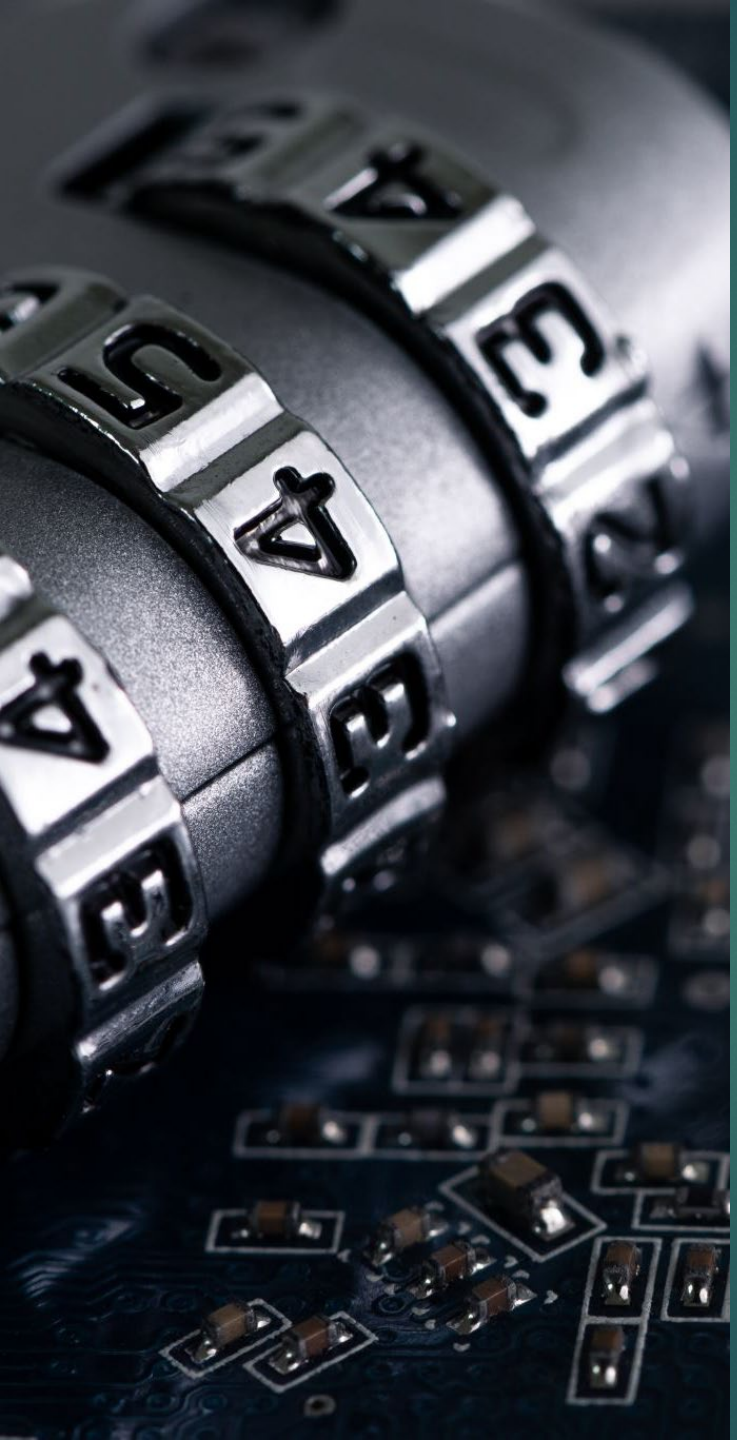
Cryptography



Cryptography

- Cryptography is the study of methods for sending secret messages.
- Encryption: a message, called plaintext, is converted into an unreadable form called ciphertext
- The ciphertext is sent over an open channel viewable by outside parties
- The receiver of the ciphertext uses decryption to convert the ciphertext back into plaintext.

Cryptography

- Systems for sending secret messages require both the sender and the receiver to know both the encryption and the decryption procedures.

- For instance, an encryption system once used by Julius Caesar, and now called the Caesar cipher, encrypts messages by changing each letter of the alphabet to the one three places farther along, with X wrapping around to A, Y to B, and Z to C.

Caesar Cipher

- Each letter of the alphabet is coded by its position relative to the others
  - A = 01, B = 02, … , Z = 26.
- The plaintext number for a letter is denoted M and the numeric version of the ciphertext is denoted C, then:

$$C = (M + 3) \; mod \; 26.$$

Python and Characters

- ord(character): Takes a string argument of a single Unicode character and returns its integer Unicode code point value.
  - For example, ord('A') returns 65 because 65 is the ASCII value for the character 'A'.
- chr(number): Takes an integer argument and returns a string representing a character at that Unicode code point.
  - For example, chr(65) returns 'A' because 65 is the ASCII value corresponding to the character 'A'.
- Using ord() to get the ASCII value of a character
  - print(ord('A')) # Output: 65
  - print(ord('B')) # Output: 66
  - print(ord('a')) # Output: 97
- Using chr() to get the character represented by an ASCII value
  - print(chr(65)) # Output: 'A'
  - print(chr(66)) # Output: 'B'
  - print(chr(97)) # Output: 'a'

Caesar Cipher

The receiver of such a message can easily decrypt it by using the formula

$$M = (C - 3) \ mod \ 26.$$

For reference, here are the letters of the alphabet, together with their numeric equivalents:

A	B	C	D	E	F	G	H	I	J	K	L	M
01	02	03	04	05	06	07	08	09	10	11	12	13
N	O	P	Q	R	S	T	U	V	W	X	Y	Z
14	15	16	17	18	19	20	21	22	23	24	25	26

Encrypting and Decrypting with the Caesar Cipher

- Use the Caesar cipher in Python to encrypt the message HOW ARE YOU.

- Use the Caesar cipher in Python to decrypt the message L DP ILQH.

Encrypt

- First, translate the letters of HOW ARE YOU into their numeric equivalents:
- 08 15 23 01 18 05 25 15 21.
- Next, encrypt the message by adding 3 to each number. The result is
- 11 18 26 04 21 08 02 18 24.
- Finally, substitute the letters that correspond to these numbers. The encrypted message becomes
- KRZ DUH BRX.

Decrypt

- First, translate the letters of L DP ILQH into their numeric equivalents:
- 12 04 16 09 12 17 08.
- Next, decrypt the message by subtracting 3 from each number:
- 09 01 13 06 09 14 05.
-
- Then, translate back into letters to obtain the original message: I AM FINE.

Python Caesar Encrypt

```
def caesar_encrypt(text, shift):
    result = ""

    for i in range(len(text)):
        char = text[i]

        # Encrypt uppercase characters
        if char.isupper():
            result += chr((ord(char) + shift - 65) % 26 + 65)

        # Encrypt lowercase characters
        elif char.islower():
            result += chr((ord(char) + shift - 97) % 26 + 97)

        # Other characters remain as they are (e.g. spaces)
        else:
            result += char

    return result
```

Python Caesar Decrypt

```
def caesar_decrypt(text, shift):
    return caesar_encrypt(text, -shift)

# Encrypt the message 'HOW ARE YOU'
encrypted_message = caesar_encrypt('HOW ARE YOU', 3)

# Decrypt the message 'L DP ILQH'
decrypted_message = caesar_decrypt('L DP ILQH', 3)

print(encrypted_message, decrypted_message)
```

XOR Encyption

- A simple encryption method based on the XOR bitwise operation
- A symmetric type of encryption where the same key is used for both encrypting and decrypting the data.
- Recall that the XOR operation takes two Boolean operands and returns true if and only if exactly one of the operands is true.
- In the context of encryption, it's typically applied to the binary representations of data.
- When you XOR a piece of data with a key, you get the encrypted data. To decrypt the data, you simply XOR it again with the same key.

XOR Encyption

Here's why XOR is particularly useful in encryption:

- Reversibility: $A \oplus K = B$ implies $B \oplus K = A$. So, if A is the original data and K is the key, B is the encrypted data. XORing B with K again yields the original data A.
- Symmetry: The operation is symmetrical, meaning $A \oplus B = B \oplus A$.
- Identity: Any data XORed with zero returns the original data, meaning $A \oplus 0 = A$.

Write the Python Code to Encrypt/Decrypt Using XOR (^)

WORK AS A TEAM OF 2 OR 3

USE THE ASCII VALUES

DEMONSTRATE A WORKING PROGRAM

HINT: `key[i % len(key)]`

SUBMIT TO BLACKBOARD

Properties of Congruence Modulo *n*

Properties of Congruence Modulo *n*

Theorem 8.4.1 Modular Equivalences

Let a, b, and n be any integers and suppose $n > 1$. The following statements are all equivalent:

1. $n \mid (a - b)$
2. $a \equiv b \pmod{n}$
3. $a = b + kn$ for some integer k
4. a and b have the same (nonnegative) remainder when divided by n
5. $a \bmod n = b \bmod n$

Properties of Congruence Modulo n

- Another consequence of the quotient-remainder theorem is this: When an integer a is divided by a positive integer n, a unique quotient q and remainder r are obtained with the property that $a = nq + r$ and $0 \leq r < n$.

- Because there are exactly n integers that satisfy the inequality $0 \leq r < n$ (the numbers from 0 through $n - 1$), there are exactly n possible remainders that can occur.

Properties of Congruence Modulo *n*

- These are called the *least nonnegative residues modulo n* or simply the *residues modulo n*.

Definition

Given integers a and n with $n > 1$, **the residue of a modulo n** is $a \bmod n$, the nonnegative remainder obtained when a is divided by n. The numbers $0, 1, 2, \ldots, n-1$ are called a **complete set of residues modulo n**. To **reduce a number modulo n** means to set it equal to its residue modulo n. If a modulus $n > 1$ is fixed throughout a discussion and an integer a is given, the words "modulo n" are often dropped and we simply speak of **the residue of a**.

Properties of Congruence Modulo *n* *(4/4)*

Theorem 8.4.2 Congruence Modulo *n* Is an Equivalence Relation

If n is any integer with $n > 1$, congruence modulo n is an equivalence relation on the set of all integers. The distinct equivalence classes of the relation are the sets [0], [1], [2], . . . , $[n-1]$, where for each $a = 0, 1, 2, \ldots, n-1$,

$$[a] = \{m \in Z \mid m \equiv a \pmod{n}\},$$

or, equivalently,

$$[a] = \{m \in Z \mid m = a + kn \text{ for some integer } k\}.$$

Modular Arithmetic

Modular Arithmetic

- A fundamental fact about congruence modulo *n* is that if you first perform an addition, subtraction, or multiplication on integers and then reduce the result modulo *n*, you will obtain the same answer as if you had first reduced each of the numbers modulo *n*, performed the operation, and then reduced the result modulo *n*. For instance, instead of computing

$$(5 \cdot 8) = 40 \equiv 1 \pmod 3$$

- You will obtain the same answer if you compute

$$(5 \bmod 3)\,(8 \bmod 3) = 2 \cdot 2 = 4 \equiv 1 \pmod 3.$$

Congruence modulo n

- Congruence modulo n is a fundamental concept in number theory. It refers to the relationship between two integers relative to a given modulus n.
- Two integers, a and b, are said to be congruent modulo n if they have the same remainder when divided by n.
- The formal definition is: $a \equiv b \bmod n$
  - This is read as "a is congruent to b modulo n" and means that n divides the difference of a and b evenly, or: $n \mid (a-b)$
- In other words, $a-b=k \cdot n$ for some integer k. The number n is called the modulus, and the congruence relation is an equivalence relation on the integers. This concept is widely used in various fields of mathematics and computer science, particularly in cryptography.

Modular Arithmetic

Theorem 8.4.3 Modular Arithmetic

Let a, b, c, d, and n be integers with $n > 1$, and suppose

$$a \equiv c \pmod{n} \quad \text{and} \quad b \equiv d \pmod{n}.$$

Then

1. $(a + b) \equiv (c + d) \pmod{n}$
2. $(a - b) \equiv (c - d) \pmod{n}$
3. $ab \equiv cd \pmod{n}$
4. $a^m \equiv c^m \pmod{n}$ for every positive integer m.

Getting Started with Modular Arithmetic

The most practical use of modular arithmetic is to reduce computations involving large integers to computations involving smaller ones. For instance, note that $55 \equiv 3 \pmod 4$ because $55 - 3 = 52$, which is divisible by 4, and $26 \equiv 2 \pmod 4$ because $26 - 2 = 24$, which is also divisible by 4.

Verify the following statements.

$55 + 26 \equiv (3 + 2) \pmod 4$

$55 - 26 \equiv (3 - 2) \pmod 4$

$55 \cdot 26 \equiv (3 \cdot 2) \pmod 4$

$55^2 \equiv 3^2 \pmod 4$

Example

To verify that $55+26 \equiv 81 \pmod 4$ and $3+2 \equiv 5 \pmod 4$, you need to show that when you subtract the right-hand side from the left-hand side of the equivalence, the result is divisible by 4.

To confirm the congruence $55+26 \equiv 81 \bmod 4$ is equivalent to $3+2 \equiv 5 \bmod 4$, check that the difference $81-5$ is a multiple of 4.

This is demonstrated as $81-5=76$, and since 76 is 19×4, it confirms the congruence.

Example 8.4.2 – Solution (2/2)

Compute 55 ·26 = 1430 and 3 ·2 = 6. By definition of congruence modulo *n*, to show that 1430 ≡ 6 (mod 4), you need to show that

$$4 \mid (1430 - 6).$$

But this is true because 1430 − 6 = 1424, and $4 \mid 1424$ since 1424 = 4 ·356.

Modular Arithmetic (3/5)

Corollary 8.4.4

Let a, b, and n be integers with $n > 1$. Then

$$ab \equiv [(a \bmod n)(b \bmod n)] \pmod{n},$$

or, equivalently,

$$ab \bmod n = [(a \bmod n)(b \bmod n)] \bmod n.$$

In particular, if m is a positive integer, then

$$a^m \equiv [(a \bmod n)^m] \pmod{n}.$$

Example 8.4.3 – *Computing a Product Modulo n*

- Note that $55 \equiv 3 \pmod 4$ and $26 \equiv 2 \pmod 4$. Because both 3 and 2 are less than 4, each of these numbers is a least nonnegative residue modulo 4. Therefore, $55 \bmod 4 = 3$ and $26 \bmod 4 = 2$. Use the notation of Corollary 8.4.4 to find the residue of $55 \cdot 26$ modulo 4.

Example 8.4.3 – *Solution (1/2)*

- We know that to use a calculator to compute remainders, you can use the formula $n \bmod d = n - d \cdot \lfloor n/d \rfloor$. If you are using a hand calculator with an "integer part" feature and both n and d are positive, then $\lfloor n/d \rfloor$ is the integer part of the division of n by d.

- When you divide a positive integer n by a positive integer d with a more basic calculator, you can see $\lfloor n/d \rfloor$ on the calculator display by simply ignoring the digits that follow the decimal point.

Example 8.4.3 – *Solution (2/2)* continued

- By Corollary 8.4.4,
- $(55 \cdot 26) \bmod 4 = \{(55 \bmod 4)(26 \bmod 4)\} \bmod 4$
- $\equiv (3 \cdot 2) \bmod 4$
  - because 55 mod 4 = 3 and 26 mod 4 = 2
- $\equiv 6 \bmod 4$
- $\equiv 2$
  - because $4 \mid (6-2)$ and $2 < 4$.

Modular Arithmetic

- When modular arithmetic is performed with very large numbers, as is the case for RSA crytography, computations are facilitated by using two properties of exponents. The first is

$$x^{2a} = (x^a)^2 \quad \text{for all real numbers } x \text{ and } a \text{ with } x \geq 0.$$

▶8.4.1

- Thus, for instance, if x is any positive real number, then

$$\begin{aligned} x^4 \bmod n &= (x^2)^2 \bmod n && \text{because } (x^2)^2 = x^4 \\ &= (x^2 \bmod n)^2 \bmod n && \text{by Corollary 8.4.4.} \end{aligned}$$

Modular Arithmetic (5/5)

- A second useful property of exponents is

$$x^{a+b} = x^a x^b \quad \text{for all real numbers } x, a, \text{ and } b \text{ with } x \geq 0.$$

8.4.2

- For instance, because 7 = 4 + 2 + 1,

$$x^7 = x^4 x^2 x^1.$$

- Thus, by Corollary 8.4.4,

$$x^7 \bmod n = \{(x^4 \bmod n)(x^2 \bmod n)(x^1 \bmod n)\} \bmod n.$$

Example

Find:

$$144^4 \ mod \ 713.$$

Example 8.4.4 – Solution

Use property (8.4.1) to write $144^4 = (144^2)^2$. Then

$$
\begin{aligned}
144^4 \bmod 713 &= (144^2)^2 \bmod 713 \\
&= (144^2 \bmod 713)^2 \bmod 713 \\
&= (20736 \bmod 713)^2 \bmod 713 && \text{because } 144^2 = 20736 \\
&= 59^2 \bmod 713 && \text{because } 20736 \bmod 713 = 59 \\
&= 3481 \bmod 713 && \text{because } 59^2 = 3481 \\
&= 629 && \text{because } 3481 \bmod 713 = 629.
\end{aligned}
$$

Example

Find:

$$12^{43} \ mod \ 713.$$

Example 8.4.5 – Solution (1/2)

First write the exponent as a sum of powers of 2:

$$43 = 2^5 + 2^3 + 2 + 1 = 32 + 8 + 2 + 1.$$

Next compute 12^{2^k} for k = 0, 1, 2, 3, 4, and 5.

$$12 \bmod 713 = 12$$

$$12^2 \bmod 713 = 144$$

$$12^4 \bmod 713 = 144^2 \bmod 713 = 59 \quad \text{by Example 8.4.4}$$

$$12^8 \bmod 713 = 59^2 \bmod 713 = 629 \quad \text{by Example 8.4.4}$$

$$12^{16} \bmod 713 = 629^2 \bmod 713 = 639 \quad \text{by the method of Example 8.4.4}$$

$$12^{32} \bmod 713 = 639^2 \bmod 713 = 485 \quad \text{by the method of Example 8.4.4.}$$

Example 8.4.5 – Solution (2/2)

continued

By property (8.4.2),

$$12^{43} = 12^{32+8+2+1} = 12^{32} \cdot 12^{8} \cdot 12^{2} \cdot 12^{1}.$$

Thus, by Corollary 8.4.4,

$$12^{43} \bmod 713$$
$$= \{(12^{32} \bmod 713) \cdot (12^{8} \bmod 713) \cdot (12^{2} \bmod 713) \cdot (12 \bmod 713)\} \bmod 713.$$

By substitution,

$$\begin{aligned} 12^{43} \bmod 713 &= (485 \cdot 629 \cdot 144 \cdot 12) \bmod 713 \\ &= 527152320 \bmod 713 \\ &= 48. \end{aligned}$$

Extending the Euclidean Algorithm

Extending the Euclidean Algorithm (1/1)

- An extended version of the Euclidean algorithm can be used to find a concrete expression for the greatest common divisor of integers a and b.

Definition

An integer d is said to be a **linear combination of integers** a and b if, and only if, there exist integers s and t such that $as + bt = d$.

Theorem 8.4.5 Writing a Greatest Common Divisor as a Linear Combination

For all integers a and b, not both zero, if $d = \gcd(a, b)$, then there exist integers s and t such that $as + bt = d$.

Example 8.4.6 – *Expressing a Greatest Common Divisor as a Linear Combination*

- We know how to use the Euclidean algorithm to find that the greatest common divisor of 330 and 156 is 6. Use the results of these calculations to express gcd(330, 156) as a linear combination of 330 and 156.

Example 8.4.6 – *Solution (1/3)*

- The first four steps were obtained by successive applications of the quotient-remainder theorem. The fifth step shows how to find the coefficients of the linear combination by substituting back through the results of the previous steps.
- **Step 1:** $330 = 156 \cdot 2 + 18$, which implies that
- $18 = 330 - 156 \cdot 2$.
- **Step 2:** $156 = 18 \cdot 8 + 12$, which implies that
- $12 = 156 - 18 \cdot 8$.
- **Step 3:** $18 = 12 \cdot 1 + 6$, which implies that $6 = 18 - 12 \cdot 1$.
- **Step 4:** $12 = 6 \cdot 2 + 0$, which implies that $\gcd(330, 156) = 6$.

Example 8.4.6 – *Solution (2/3)* continued

- **Step 5:** By substituting back through steps 3 to 1:

- $6 = 18 - 12 \cdot 1$ — from step 3
  - $= 18 - (156 - 8 \cdot 18) \cdot 1$ — by substitution from step 2
  - $= 9 \cdot 18 + (-1) \cdot 156$ — by algebra
  - $= 9 \cdot (330 - 156 \cdot 2) + (-1) \cdot 156$ — by substitution from step 1
  - $= 9 \cdot 330 + (-19) \cdot 156$ — by algebra.

Example 8.4.6 – *Solution (3/3)*

- Thus gcd(330, 156) = 9 ·330 + (−19) ·156. (It is always a good idea to check the result of a calculation like this to be sure you did not make a mistake. In this case, you find that 9 ·330 + (−19) ·156 does indeed equal 6.)

Finding an Inverse Modulo n

Finding an Inverse Modulo *n* (1/2)

Definition

Given any integer a and any positive integer n, if there exists an integer s such that $as \equiv 1 \pmod{n}$, then s is called **an inverse for a modulo n**.

Definition

Integers a and b are **relatively prime** if, and only if, $\gcd(a, b) = 1$. Integers $a_1, a_2, a_3, \ldots, a_n$ are **pairwise relatively prime** if, and only if, $\gcd(a_i, a_j) = 1$ for all integers i and j with $1 \leq i, j \leq n$, and $i \neq j$.

Corollary 8.4.6

If a and b are relatively prime integers, then there exist integers s and t such that $as + bt = 1$.

Example 8.4.7– *Expressing 1 as a Linear Combination of Relatively Prime Integers*

- Show that 660 and 43 are relatively prime, and find a linear combination of 660 and 43 that equals 1.

Example 8.4.7 – *Solution (1/4)*

- **Step 1:** Divide 660 by 43 to obtain $660 = 43 \cdot 15 + 15$, which implies that $15 = 660 - 43 \cdot 15$.
- **Step 2:** Divide 43 by 15 to obtain $43 = 15 \cdot 2 + 13$, which implies that $13 = 43 - 15 \cdot 2$.
- **Step 3:** Divide 15 by 13 to obtain $15 = 13 \cdot 1 + 2$, which implies that $2 = 15 - 13$.

Example 8.4.7 – *Solution (2/4)* continued

- **Step 4:** Divide 13 by 2 to obtain $13 = 2 \cdot 6 + 1$, which implies that $1 = 13 - 2 \cdot 6$.

- **Step 5:** Divide 2 by 1 to obtain $2 = 1 \cdot 2 + 0$, which implies that gcd(660, 43) = 1 and so 660 and 43 are relatively prime.

Example 8.4.7 – *Solution (3/4)* continued

- **Step 6:** To express 1 as a linear combination of 660 and 43, substitute back through steps 4 to 1:

- $1 = 13 - 2 \cdot 6$ ▶from step 4

▶$= 13 - (15 - 13) \cdot 6$ ▶by substitution from step 3

▶$= 7 \cdot 13 - 6 \cdot 15$ ▶by algebra

▶$= 7 \cdot (43 - 15 \cdot 2) - 6 \cdot 15$ ▶by substitution from step 2

Example 8.4.7 – *Solution (4/4)* continued

▶ $= 7 \cdot 43 - 20 \cdot 15$ ▶by algebra.

▶ $= 7 \cdot 43 - 20 \cdot (660 - 43 \cdot 15)$ ▶by substitution from step 1

▶ $= 307 \cdot 43 - 20 \cdot 660$ ▶by algebra.

▶Thus gcd(660, 43) = 1 = $307 \cdot 43 - 20 \cdot 660$. (And a check by direct computation confirms that $307 \cdot 43 - 20 \cdot 660$ does indeed equal 1.)

Finding an Inverse Modulo *n* *(2/2)*

Corollary 8.4.7 Existence of Inverses Modulo *n*

For all integers a and n, if $\gcd(a, n) = 1$, then there exists an integer s such that $as \equiv 1 \pmod{n}$, and so s is an inverse for a modulo n.

Example 8.4.8 – *Finding an Inverse Modulo n*

- a. Find an inverse for 43 modulo 660. That is, find an integer s such that $43s \equiv 1 \pmod{660}$.

- b. Find a positive inverse for 3 modulo 40. That is, find a positive integer s such that $3s \equiv 1 \pmod{40}$.

Example 8.4.8 – *Solution (1/4)*

- a. By Example 8.4.7,
- $307 \cdot 43 - 20 \cdot 660 = 1.$
  - Adding $20 \cdot 660$ to both sides gives that
  - $307 \cdot 43 = 1 + 20 \cdot 660.$
  - Thus, by definition of congruence modulo 660,
  - $307 \cdot 43 \equiv 1 \pmod{660}$,
  - so 307 is an inverse for 43 modulo 660.

Example 8.4.8 – *Solution (2/4)* continued

- b. Use the technique of Example 8.4.7 to find a linear combination of 3 and 40 that equals 1.

- **Step 1:** Divide 40 by 3 to obtain $40 = 3 \cdot 13 + 1$. This implies that $1 = 40 - 3 \cdot 13$.

- **Step 2:** Divide 3 by 1 to obtain $3 = 3 \cdot 1 + 0$. This implies that $\gcd(3, 40) = 1$.

Example 8.4.8 – *Solution (3/4)* continued

- **Step 3:** Use the result of step 1 to write
- $3 \cdot (-13) = 1 + (-1)40.$
- This result implies that −13 is an inverse for 3 modulo 40. In other words, $3 \cdot (-13) \equiv 1 \pmod{40}$.
- To find a positive inverse, compute 40 − 13. The result is 27, and
- $27 \equiv -13 \pmod{40}$
- because 27 − (−13) = 40.

Example 8.4.8 – *Solution (4/4)* continued

- So, by Theorem 8.4.3(3),
- $3 \cdot 27 \equiv 3 \cdot (-13) \equiv 1 \pmod{40}$,
- and thus by the transitive property of congruence modulo n, 27 is a positive integer that is an inverse for 3 modulo 40.

Theorem 8.4.3 Modular Arithmetic

Let a, b, c, d, and n be integers with $n > 1$, and suppose

$$a \equiv c \pmod{n} \quad \text{and} \quad b \equiv d \pmod{n}.$$

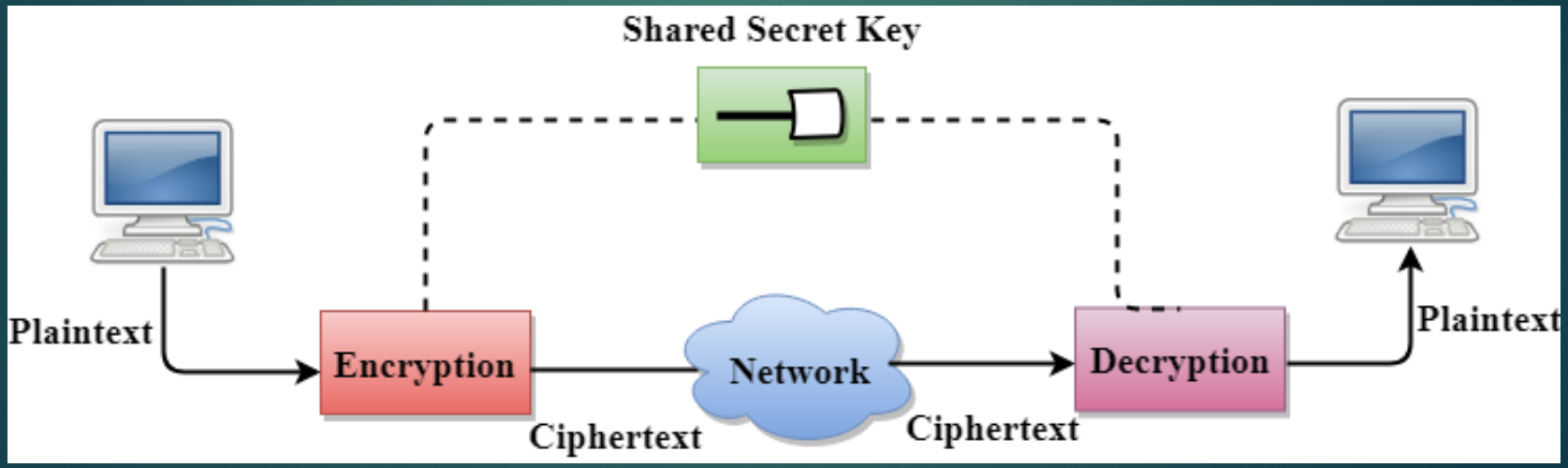
Then

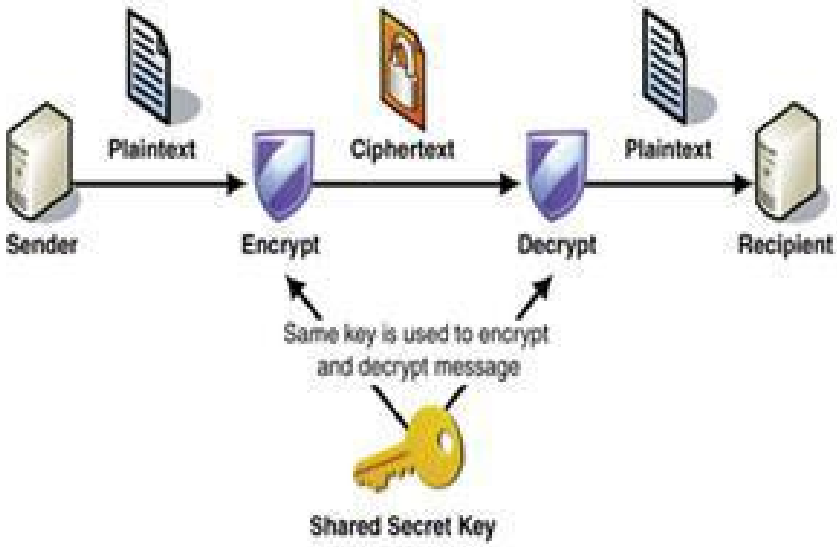
1. $(a + b) \equiv (c + d) \pmod{n}$
2. $(a - b) \equiv (c - d) \pmod{n}$
3. $ab \equiv cd \pmod{n}$
4. $a^m \equiv c^m \pmod{n}$ for every positive integer m.

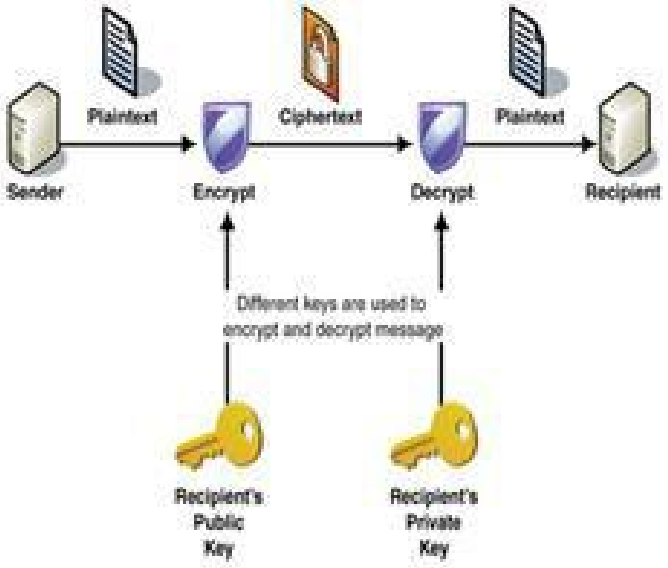
RSA Cryptography

Cryptography

- The art and science of concealing the messages to introduce secrecy in information security is recognized as cryptography.
- Plaintext ⇌ Cyphertext
- Encryption and Decryption
- Generally approached as puzzle





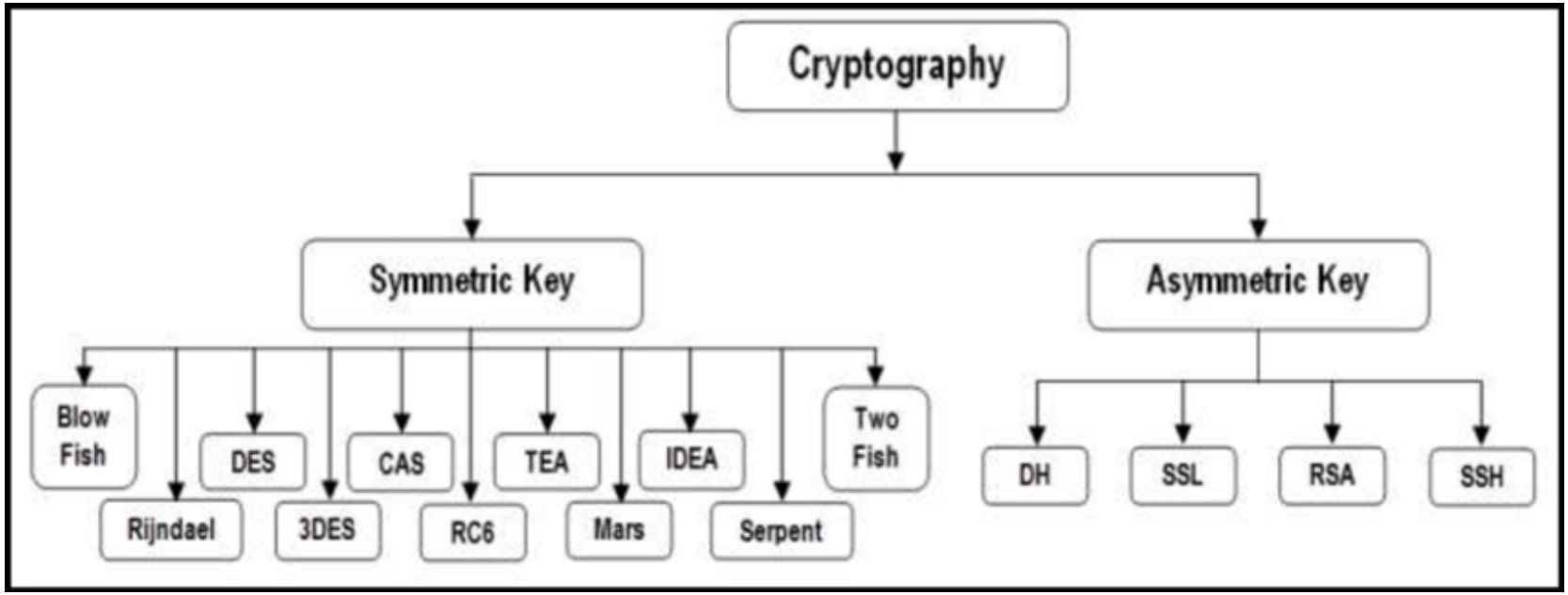


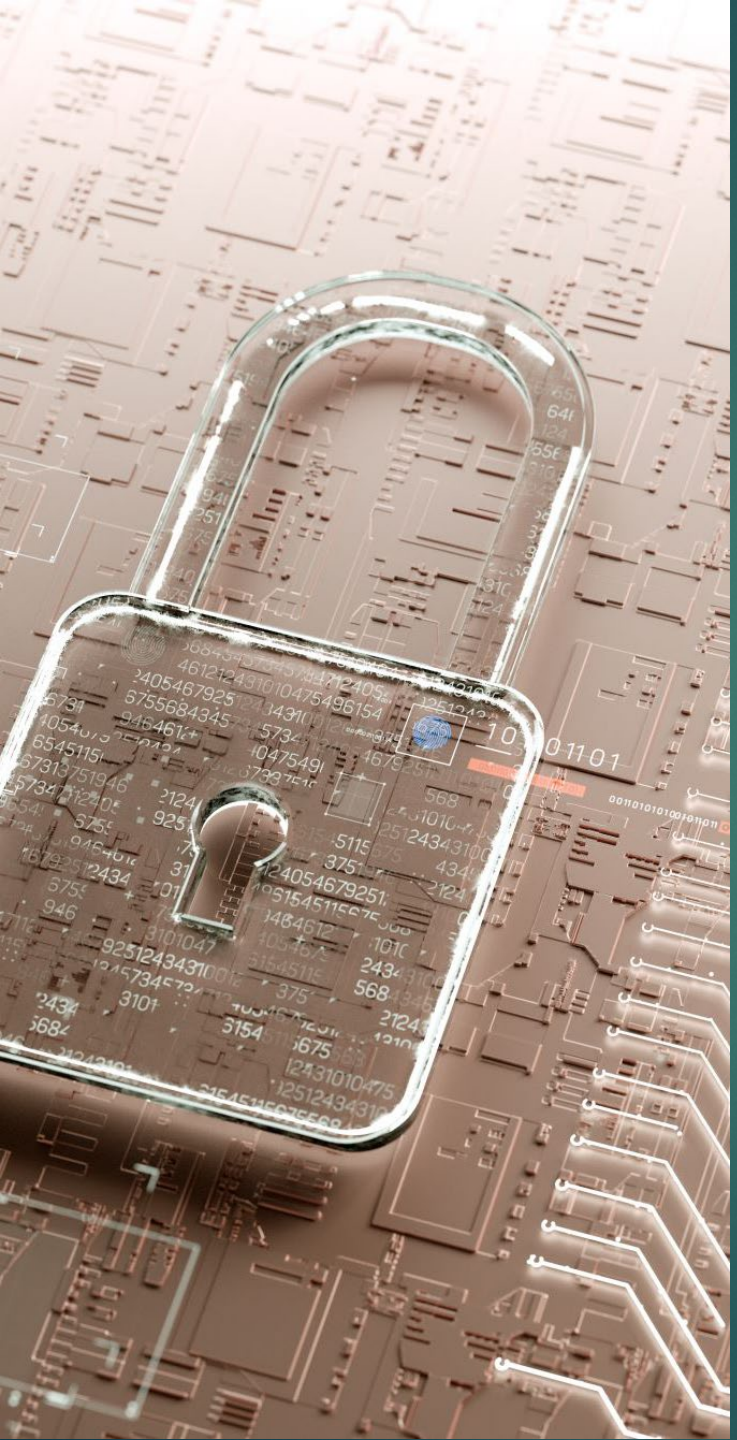
Cryptography

SYMMETRIC AND ASYMMETRIC KEYS

Cryptography

SYMMETRIC AND
ASYMMETRIC KEYS





Symmetric Encryption:

- Symmetric encryption uses a single key for both encryption and decryption processes.
- The same key is used by both the sender and the recipient.
- It's efficient for large volumes of data and generally faster than asymmetric encryption.
- Common symmetric encryption algorithms include AES (Advanced Encryption Standard) and DES (Data Encryption Standard).

Asymmetric Encryption:

- Asymmetric encryption employs a pair of keys: a public key and a private key.
- The public key is freely distributed and used for encryption, while the private key is kept secret and used for decryption.
- It enables secure communication between two parties without the need to share a secret key.
- Asymmetric encryption is slower than symmetric encryption due to the complexity of the algorithms involved.
- Common asymmetric encryption algorithms include RSA (Rivest-Shamir-Adleman) and ECC (Elliptic Curve Cryptography).

Symmetric vs Asymmetric

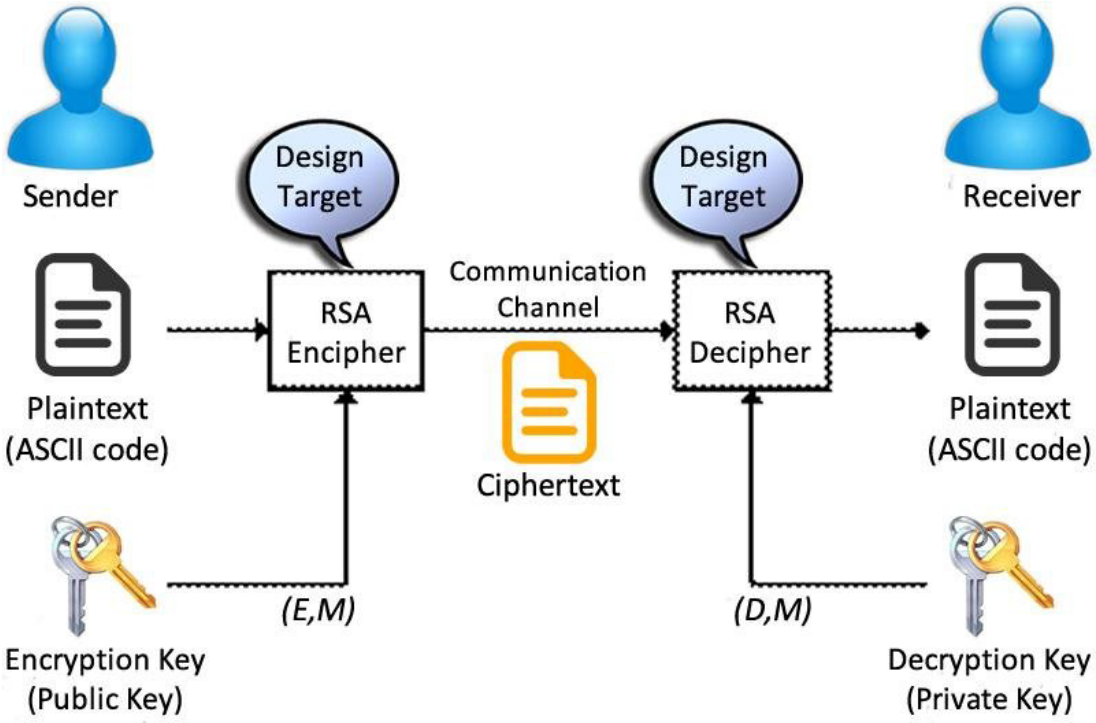
- **Comparison**:
- **Security**: Asymmetric encryption offers higher security because the private key never needs to be shared or transmitted, unlike symmetric keys. However, both methods can provide strong security when implemented correctly.
- **Key Management**: Symmetric encryption requires secure key distribution to all parties involved, which can be challenging in some cases. Asymmetric encryption simplifies key management as each user has a unique public-private key pair.
- **Speed**: Symmetric encryption is generally faster than asymmetric encryption because it involves simpler mathematical operations. Asymmetric encryption, with its complex algorithms, tends to be slower.
- **Use Cases**: Symmetric encryption is often used for bulk data encryption, such as securing files or hard drives. Asymmetric encryption is commonly used for secure communication channels, like SSL/TLS for web browsing, and digital signatures for authentication and data integrity.

Public Key Cryptography

RSA (Ronald Rivest, Asahi Shamir, Leonardo Adleman)

- Two keys
  - Private key known only to individual
  - Public key available to anyone
- Public key, private key inverses (we will see how)
- Idea
  - Confidentiality: encipher using public key, decipher using private key
    - Secret messaging
  - Authentication: encipher using private key, decipher using public one
    - Digital Signature

RSA


Sender
Design Target
Design Target
Receiver
Plaintext (ASCII code)
RSA Encipher
Communication Channel
RSA Decipher
Plaintext (ASCII code)
Ciphertext
(E,M)
(D,M)
Encryption Key (Public Key)
Decryption Key (Private Key)

RSA Cryptography

- To encrypt a message using the RSA cipher, a person needs to know the value of pq and of another integer e, both of which are made publicly available. But only a person who knows the individual values of p and q can decrypt an encrypted message.

- We first give an example to show how the cipher works and then discuss some of the theory to explain why it works.

RSA

- Exponentiation cipher
  - •Difficulty of determining the number of numbers relatively prime to a large integer n
- Totient function Ø(n)
  - •Number of positive integers less than n and relatively prime to n
    - •If n = p*q, Ø(n) = (p–1)(q–1)

  Example: Ø (10) = 4
    - •1, 3, 7, 9 are relatively prime to 10

  Example: Ø (21) = 12
    - •1, 2, 4, 5, 8, 10, 11, 13, 16, 17, 19, 20 are relatively prime to 21

RSA

- Algorithm
  - Choose 2 large prime numbers p, q
    - Let n = p*q, then Ø(n) = (p–1)(q–1)
    - Choose e < n | e is relatively prime to Ø(n)
    - Compute d such that e*d mod Ø(n) = 1
    - i.e., d is inverse of e mod Ø(n).
  - Public key: (e, n); private key: d
  - Encipher: $c = m^e \bmod n$
  - Decipher: $m = c^d \bmod n$

Why RSA is unbreakable? (Secure)

- **Factoring Large Prime Numbers:**
  - **RSA relies on the difficulty of factoring large composite numbers $n=p\times q$, where p and q are large prime numbers.**
  - **Efficiently factoring n into its prime factors is computationally infeasible, especially when p and q are large.**
- **Security Assumptions:**
  - **RSA's security assumes that factoring large numbers into primes is a hard problem.**
  - **Despite efforts, no efficient algorithm has been found to factor large numbers in polynomial time.**
- **Large Key Sizes:**
  - **RSA keys are typically generated with large sizes (e.g., 2048 or 4096 bits) to resist attacks.**
  - **Larger key sizes make it even more difficult to factorize the product of two large prime numbers.**
- **Cryptanalysis and Research:**
  - **RSA has undergone extensive cryptanalysis and security research.**
  - **While certain attacks pose risks, RSA's security fundamentally relies on the difficulty of factoring large numbers.**

http://certauth.epfl.ch/CA/rsa/node3.html

Why RSA is unbreakable?

- In 2009, it took 290 computers over the internet and a supercomputer 4 months to find that

n = 109417386415705274218097073220403576120037329454492059909138421314763499842889347847179972578912673324976257528997818337970765372440271467435315933543338 97 had prime factors

p = 0263959282974110577205419657399167590071656780803806680334193352179071130777 9

q = 1066034883801684548209272203600128786792079585759892915222706082371930628086 43

n had only 155 digits. Many values of n have over 200 digits, making the RSA algorithm nearly unbreakable.

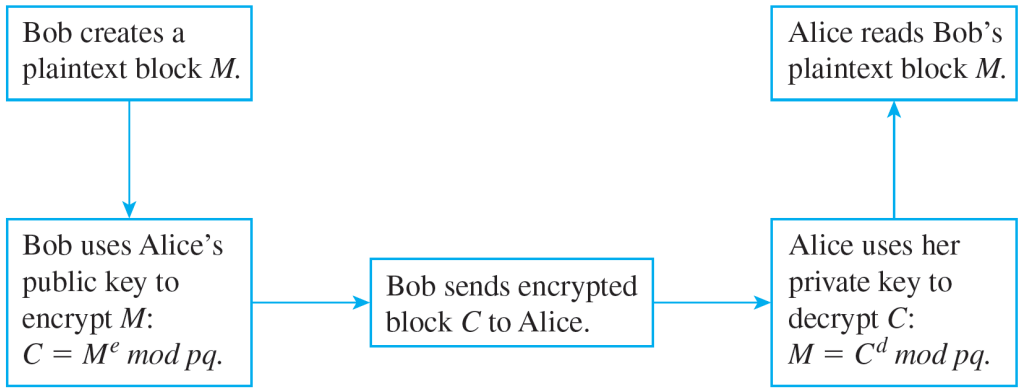
http://certauth.epfl.ch/CA/rsa/node3.html

Encrypting a Message Using RSA Cryptography

BOB WANTS TO SEND ALICE THE MESSAGE HI. WHAT IS THE CIPHERTEXT FOR HIS MESSAGE?

RSA Cryptography

▶Figure 8.4.1 illustrates the process of sending and receiving a message using RSA cryptography.



Using RSA cryptography

Figure 8.4.1

RSA Cryptography

Let us also assume that the messages Alice receives consist of blocks, each of which, for simplicity, is taken to be a single, numerically encoded letter of the alphabet.

Someone who wants to send Alice a message breaks the message into blocks, each consisting of a single letter, and finds the numeric equivalent for each block.

RSA Cryptography

The plaintext, *M*, in a block is converted into ciphertext, *C*, according to the following formula:

$$C = M^e \bmod pq.$$

8.4.5

Note that because (*pq*, *e*) is the public key, anyone who has it and knows modular arithmetic can encrypt a message to send to Alice.

RSA Cryptography

Suppose Alice decides to set up an RSA cipher. She chooses two prime numbers—say, $p = 5$ and $q = 11$—and computes $pq = 55$.

She then chooses a positive integer e that is relatively prime to $(p - 1)(q - 1)$. In this case, $(p - 1)(q - 1) = 4 \cdot 10 = 40$,

Choosing Public and Private Keys:

- Alice selects a public encryption exponent e=17.
- She computes her private decryption exponent d such that $e \times d \equiv 1 \pmod{\emptyset(n)}$.
- In this case, d=53 because $17 \times 53 \equiv 1 \pmod{60}$.

Example 8.4.9 – Solution (1/2)

Bob will send his message in two blocks, one for the H and another for the I. Because H is the eighth letter in the alphabet, it is encoded as 08, or 8.

The corresponding ciphertext is computed using formula 8.4.5 as follows:

$$C = 8^3 \bmod 55$$

$$= 512 \bmod 55$$

$$= 17.$$

Example 8.4.9 – Solution (2/2)

continued

Because I is the ninth letter in the alphabet, it is encoded as 09, or 9. The corresponding ciphertext is

$$C = 9^3 \bmod 55$$

$$= 729 \bmod 55$$

$$= 14.$$

Accordingly, Bob sends Alice the message: 17 14.

RSA Cryptography (6/7)

- To decrypt the message, the *decryption key* must be computed. It is a number d that is a positive inverse to e modulo $(p - 1)(q - 1)$.

- The plaintext M is obtained from the ciphertext C by the formula

$M = C^d \bmod pq$, where the number pair (pq, d) is Alice's **private key**.

- 8.4.6

Example 8.4.10 – Decrypting a Message Using RSA Cryptography

Imagine that Alice has hired you to help her decrypt messages and has shared with you the values of *p* and *q*. Compute Alice's private key (*pq*, *d*) and use the formula

$M = C^d \bmod pq$ to decrypt the following ciphertext for her: 17 14.

Example 8.4.10 – Solution (1/4)

Because $p = 5$ and $q = 11$, $(p - 1)(q - 1) = 40$, the decryption key d is a positive inverse for 3 modulo 40. Knowing that you would need this number, we computed it in Example 8.4.8(b) and found it to be 27. Thus to decrypt the ciphertext 17, you need to compute

$$M = 17^d \bmod pq = 17^{27} \bmod 55.$$

To do so, note that

$$27 = 16 + 8 + 2 + 1.$$

Example 8.4.10 – Solution (2/4)

continued

Next, find the residues obtained when 17 is raised to successively higher powers of 2, up to $2^4 = 16$:

$$17 \bmod 55 = 17 \bmod 55 = 17$$

$$17^2 \bmod 55 = 17^2 \bmod 55 = 14$$

$$17^4 \bmod 55 = (17^2)^2 \bmod 55 = 14^2 \bmod 55 = 31$$

$$17^8 \bmod 55 = (17^4)^2 \bmod 55 = 31^2 \bmod 55 = 26$$

$$17^{16} \bmod 55 = (17^8)^2 \bmod 55 = 26^2 \bmod 55 = 16$$

Example 8.4.10 – Solution (3/4)

continued

Then use the fact that

$$17^{27} = 17^{16+8+2+1} = 17^{16} \cdot 17^{8} \cdot 17^{2} \cdot 17^{1}$$

to write

$$\begin{aligned} 17^{27} \bmod 55 &= (17^{16} \cdot 17^{8} \cdot 17^{2} \cdot 17) \bmod 55 \\ &\equiv [(17^{6} \bmod 55)(17^{8} \bmod 55)(17^{2} \bmod 55)(17 \bmod 55)] \pmod{55} \\ &\qquad \text{by Corollary 8.4.4} \\ &\equiv (16 \cdot 26 \cdot 14 \cdot 17) \pmod{55} \\ &\equiv 99008 \pmod{55} \\ &\equiv 8 \pmod{55}. \end{aligned}$$

Example 8.4.10 – Solution (4/4)

Hence $17^{27} \ mod \ 55 = 8,$ thus the plaintext of the first part of Bob's message is 8, or 08. In the last step, you find the letter corresponding to 08, which is *H*.

In exercises 14 and 15 at the end of this section, you are asked to show that when you decrypt 14, the result is 9, which corresponds to the letter I, so you can tell Alice that Bob's message is HI.

RSA Cryptography

Suppose Alice decides to set up an RSA cipher. She chooses two prime numbers—say, $p = 7$ and $q = 11$—and computes $pq = 77$.

She then chooses a positive integer e that is relatively prime to $(p - 1)(q - 1)$. In this case, $(p - 1)(q - 1) = 6 \cdot 10 = 60$,

Choosing Public and Private Keys:

- Alice selects a public encryption exponent e=17.
- She computes her private decryption exponent d such that $e \times d \equiv 1 \pmod{\emptyset(n)}$.
- In this case, d=53 because $17 \times 53 \equiv 1 \pmod{60}$.

RSA Example

- p = 7, q = 11, so n = 77 and Ø (n) = 60
- Alice chooses e = 17, making d = 53
- Bob wants to send Alice secret message HELLO (07 04 11 11 14)
  - $07^{17} \bmod 77 = 28$
  - $04^{17} \bmod 77 = 16$
  - $11^{17} \bmod 77 = 44$
  - $11^{17} \bmod 77 = 44$
  - $14^{17} \bmod 77 = 42$
- Bob sends 28 16 44 44 42
  - Encrypted!
- Alice receives 28 16 44 44 42
- Alice uses private key, d = 53, to decrypt message:
  - $28^{53} \bmod 77 = 07$
  - $16^{53} \bmod 77 = 04$
  - $44^{53} \bmod 77 = 11$
  - $44^{53} \bmod 77 = 11$
  - $42^{53} \bmod 77 = 14$
- Alice translates message to letters to read HELLO
  - Decrypted!

Key Exchange

- A protocol by which two parties can exchange a secret key over an insecure channel without having any past shared secret information.
- Diffie-Hellman key exchange protocol:
  - Suppose that Alice and Bob want to share a common key.
  - Alice and Bob agree to use a prime p and an integer a.
  - Alice chooses a secret integer k_1 and sends $x = a^{k1} \bmod p$ to Bob.
  - Bob chooses a secret integer k_2 and sends $y = a^{k2} \bmod p$ to Alice.
  - Alice computes $z_1 = y^{k1} \bmod p = (a^{k2} \bmod p)^{k1} \bmod p = (a^{k2})^{k1} \bmod p$.
  - Bob computes $z_2 = x^{k2} \bmod p = (a^{k1} \bmod p)^{k2} \bmod p = (a^{k1})^{k2} \bmod p$.
  - At the end of the protocol, Alice and Bob have their shared key
    - $(a^{k2})^{k1} \bmod p = (a^{k1})^{k2} \bmod p$.

RSA Encryption in Python

```
def gcd(a, b):
    while b:
        a, b = b, a % b
    return a

def is_coprime(a, b):
    return gcd(a, b) == 1

def euler_totient(n):
    count = 0
    for i in range(1, n + 1):
        if is_coprime(i, n):
            count += 1
    return count

def mod_inverse(e, phi):
    for d in range(1, phi):
        if (e * d) % phi == 1:
            return d
    return None
```

RSA Encryption in Python

```
def rsa_encrypt(message, e, n):
    return pow(message, e, n)

def rsa_decrypt(ciphertext, d, n):
    return pow(ciphertext, d, n)

# Example RSA setup
p, q = 61, 53  # Two prime numbers
n = p * q
phi_n = (p - 1) * (q - 1)
e = 31  # Public key exponent (must be coprime to phi_n)
d = mod_inverse(e, phi_n)  # Private key exponent

# Message to encrypt
message = 42
ciphertext = rsa_encrypt(message, e, n)
decrypted_message = rsa_decrypt(ciphertext, d, n)

print(f"Public Key: (e={e}, n={n})")
print(f"Private Key: (d={d}, n={n})")
print(f"Original Message: {message}")
print(f"Encrypted Message: {ciphertext}")
print(f"Decrypted Message: {decrypted_message}")
```