اونيۏرسيتي مليسيا ڤهڠ السلطان عبدالله
UNIVERSITI MALAYSIA PAHANG
AL-SULTAN ABDULLAH

Data & Network Security

Chapter 3 – Cryptography
Part 2

Outline

3.1 Introduction

3.2 Terminology

3.2.1 Plain text, cipher text

3.2.2 Encryption, decryption

3.2.3 Key

3.2.3 Symmetric and Asymmetric Cryptosystem

3.2.4 Cryptanalysis & Brute Force Attack

3.4 Cipher Types

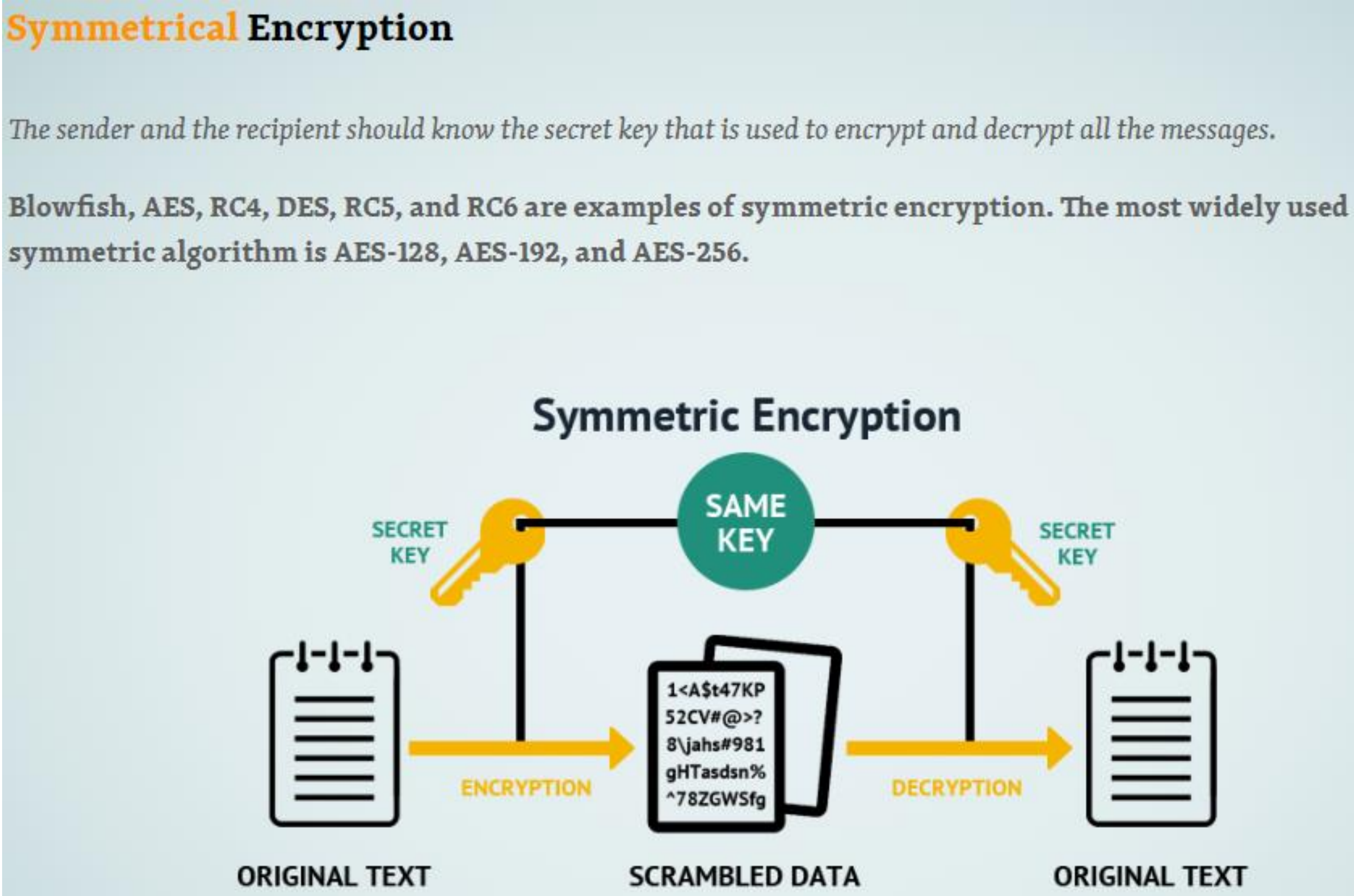
3.5 Classical Cryptography

3.5 Symmetric Encryption

3.6 Asymmetric Encryption

3.7 Public Key Cryptosystem (RSA, Key Exchange, Diffie-Hellman)

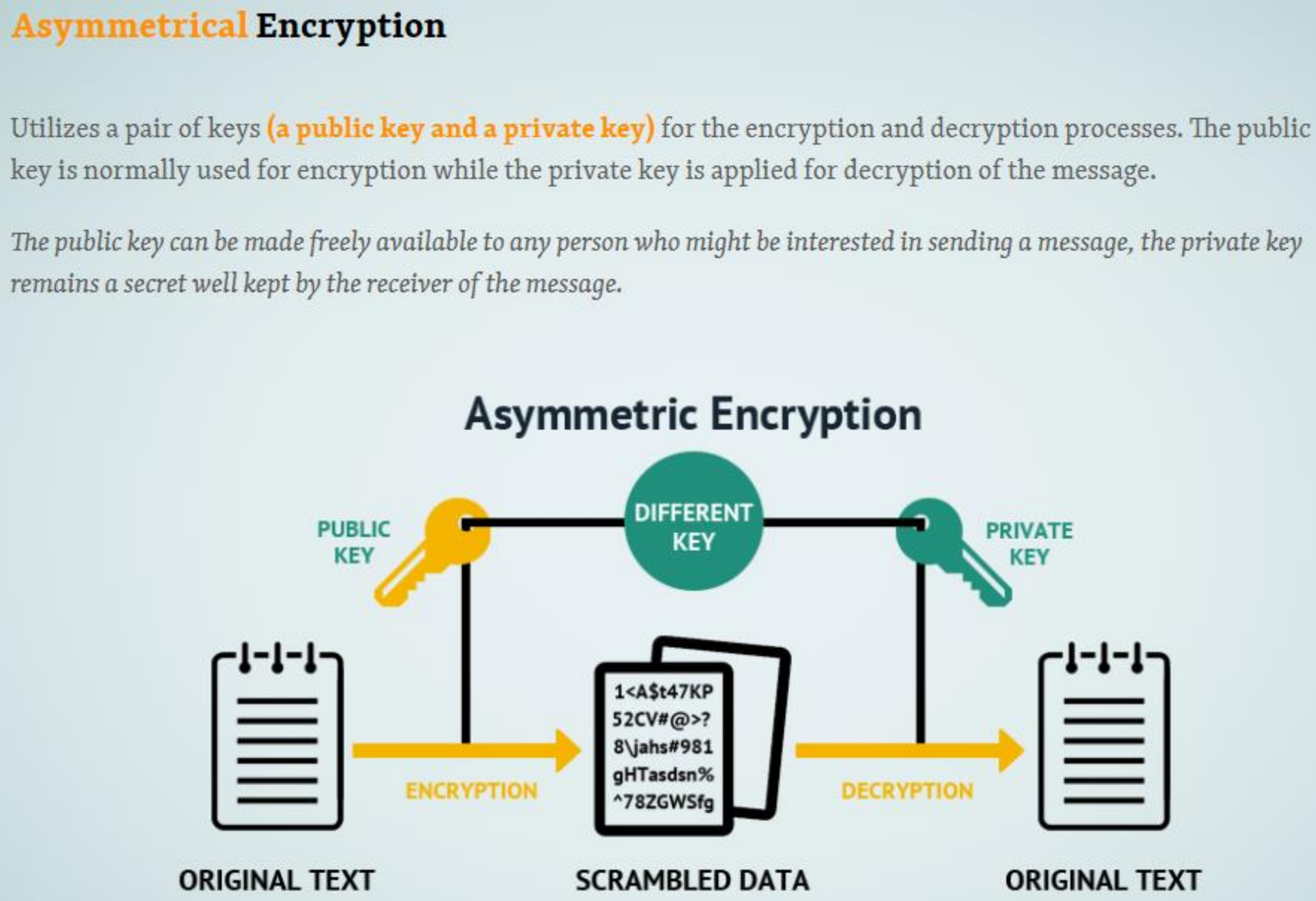
3.8 Others - Message Authentication, Hash Function, Digital Signature



Symmetrical Encryption

The sender and the recipient should know the secret key that is used to encrypt and decrypt all the messages.

Blowfish, AES, RC4, DES, RC5, and RC6 are examples of symmetric encryption. The most widely used symmetric algorithm is AES-128, AES-192, and AES-256.



Asymmetrical Encryption

Utilizes a pair of keys **(a public key and a private key)** for the encryption and decryption processes. The public key is normally used for encryption while the private key is applied for decryption of the message.

The public key can be made freely available to any person who might be interested in sending a message, the private key remains a secret well kept by the receiver of the message.

Asymmetric Encryption

- Also called as Public Key Cryptography, 2 different keys (which form a key pair) are used.
- One key is used for encryption and only the other corresponding key must be used for decryption.
- No other key can decrypt the message – not even the original (the first) key used for encryption!
- The beauty of this scheme is that every communicating party needs just a key pair.

Asymmetric Encryption

- One of the 2 keys is called as public key and the other is the private key.
- The private key remains with you as a secret.
- The private key must not be disclosed to anybody.
- However, the public key is for the public.
- It is disclosed to all parties that you want to communicate with.
- In this scheme each party publishes its public key.

Asymmetric Encryption

- Suppose A wants to send a message to B without having to worry about its security.
- Then, A and B should each have a private key and a public key.
  - √ A should keep her private key secret
  - √ B should keep her private key secret
  - √ A should inform B about her public key
  - √ B should inform A about her public key
- Thus, we have a matrix as shown next.

Asymmetric Encryption

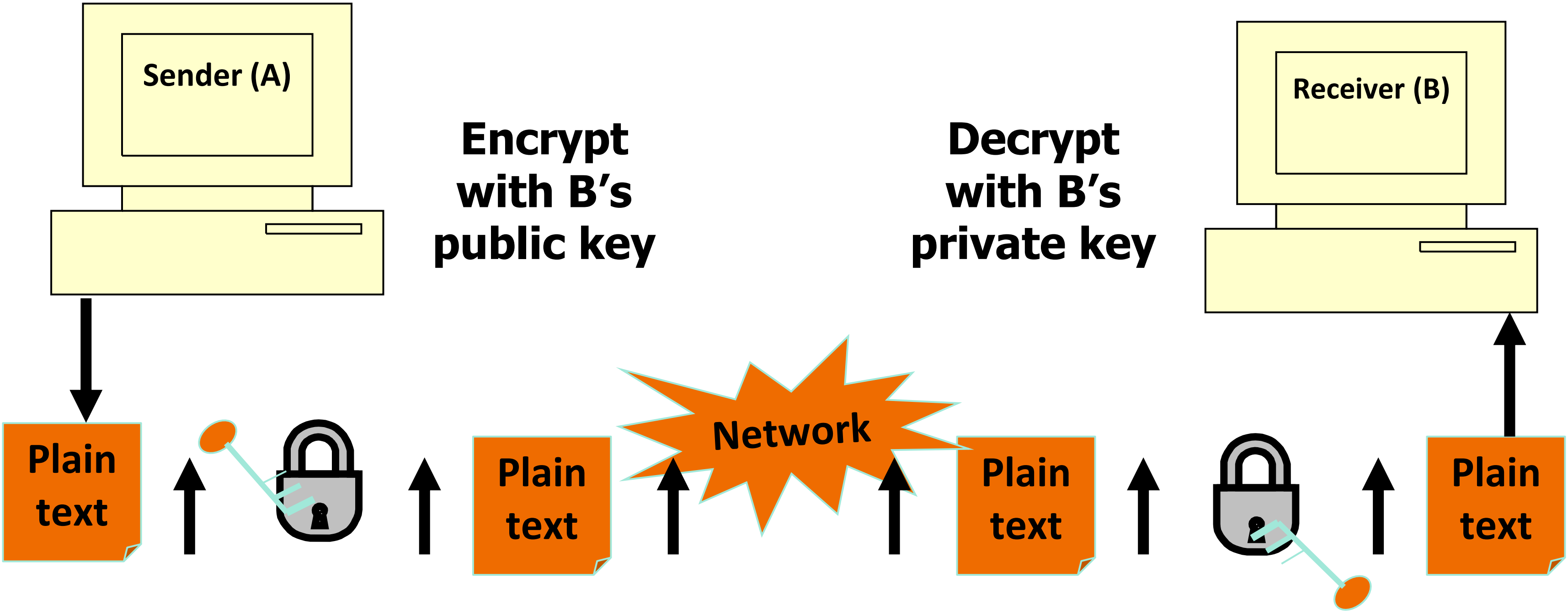
Key details	A should know	B should know
A’s private key	Yes	No
A’s public key	Yes	Yes
B’s private key	No	Yes
B’s public key	Yes	Yes

- Asymmetric key cryptography works as follows:
  - when A wants to send a message to B, A encrypts the message using B’s public key. This is possible because A knows B’s public key.
  - A sends this message (which was encrypted with B’s public key) to B.
  - B decrypts A’s message using B’s private key.

Asymmetric Encryption

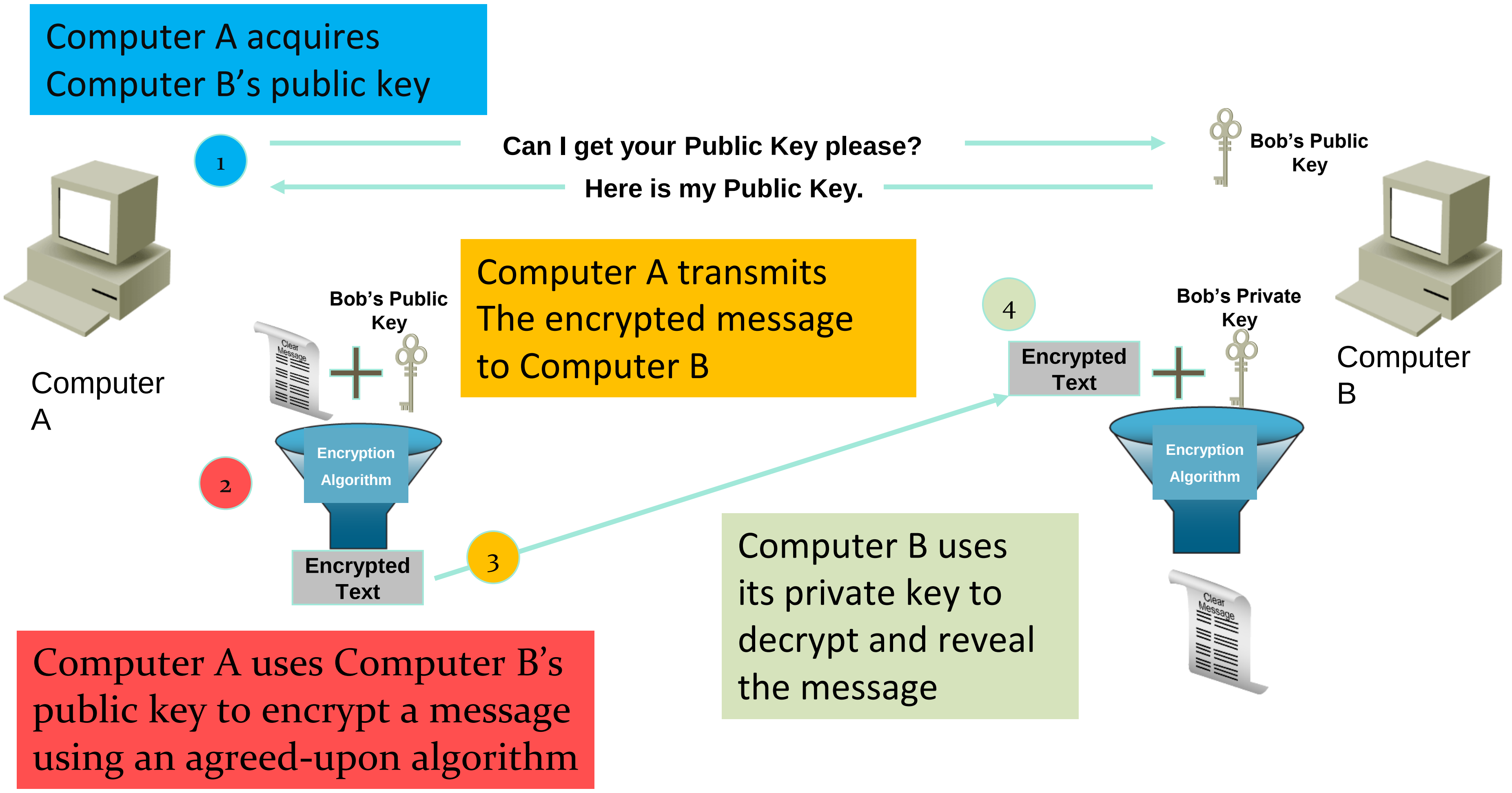
- Note that only B knows about her private key.
- Also note that the message can be decrypted only by B's private key and nothing else!
- Thus, no one else can make any sense out of the message even if one can manage to intercept the message.
- This is because the intruder (ideally) does not know about B's private key. It is only B's private key that can decrypt the message.
- Similarly, when B wants to send a message to A, exactly reverse steps take place.

Asymmetric Encryption



A encrypts the message using B’s public key. Therefore, only B can decrypt the message back to its original form, using her private key.

Public Key (Encrypt) + Private Key (Decrypt) = Achieved Confidentiality



Computer A acquires Computer B's public key

Computer A uses Computer B's public key to encrypt a message using an agreed-upon algorithm

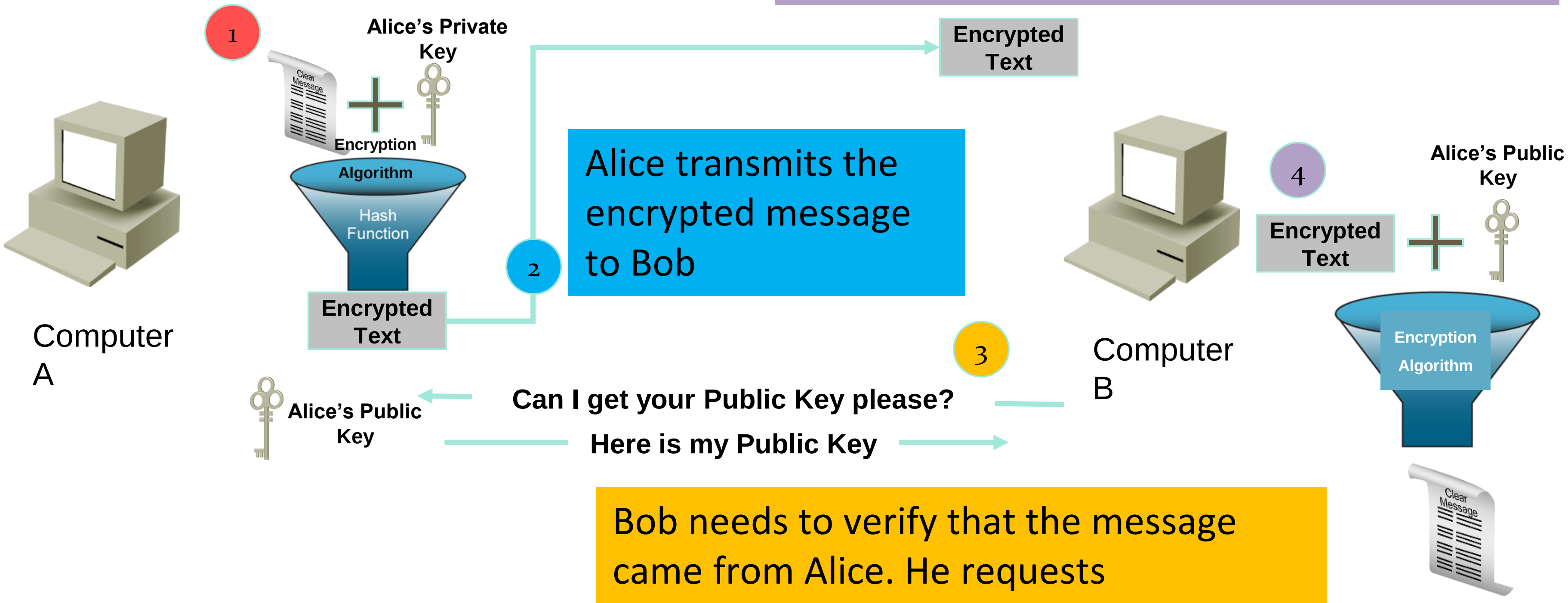
Computer A transmits The encrypted message to Computer B

Computer B uses its private key to decrypt and reveal the message

Private Key (Encrypt) + Public Key (Decrypt) = Achieved Authentication

Alice encrypts a message with her private key

Bob uses the public key to successfully decrypt the message and authenticate that the message did, indeed, come from Alice.



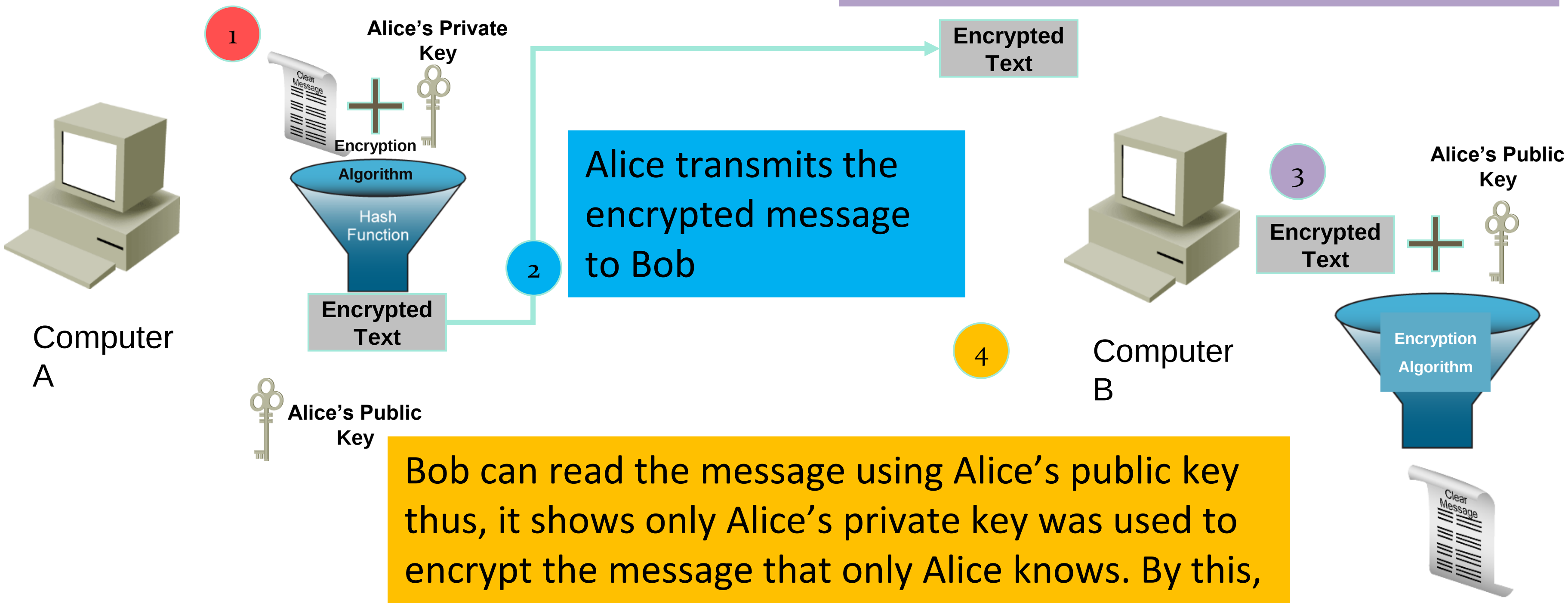
Alice transmits the encrypted message to Bob

Bob needs to verify that the message came from Alice. He requests and acquires Alice's public key

Private Key (Encrypt) + Public Key (Decrypt) = Achieved Non-Repudiation

Alice encrypts a message with her private key

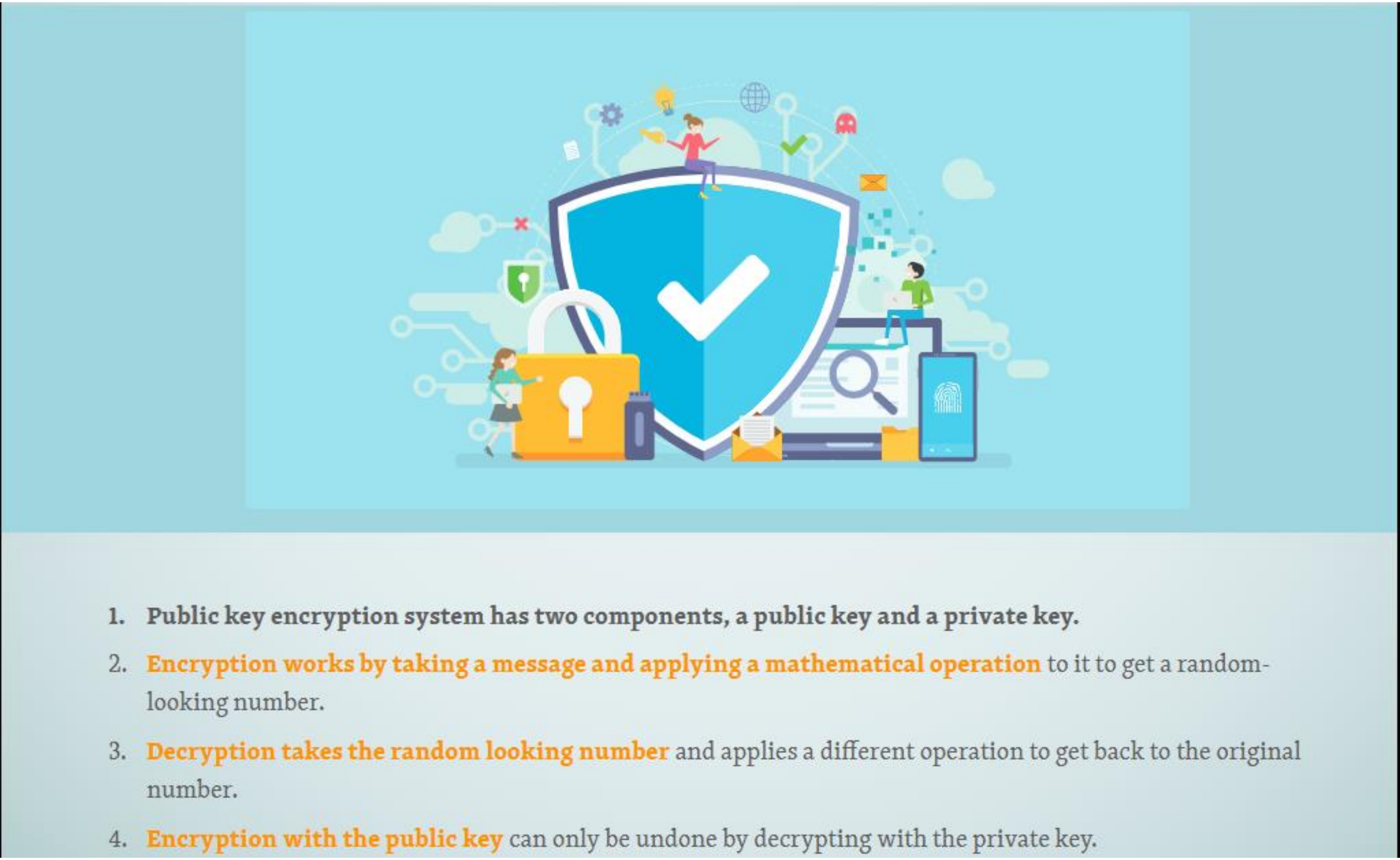
Bob uses Alice's public key to successfully decrypt the message.



Alice transmits the encrypted message to Bob

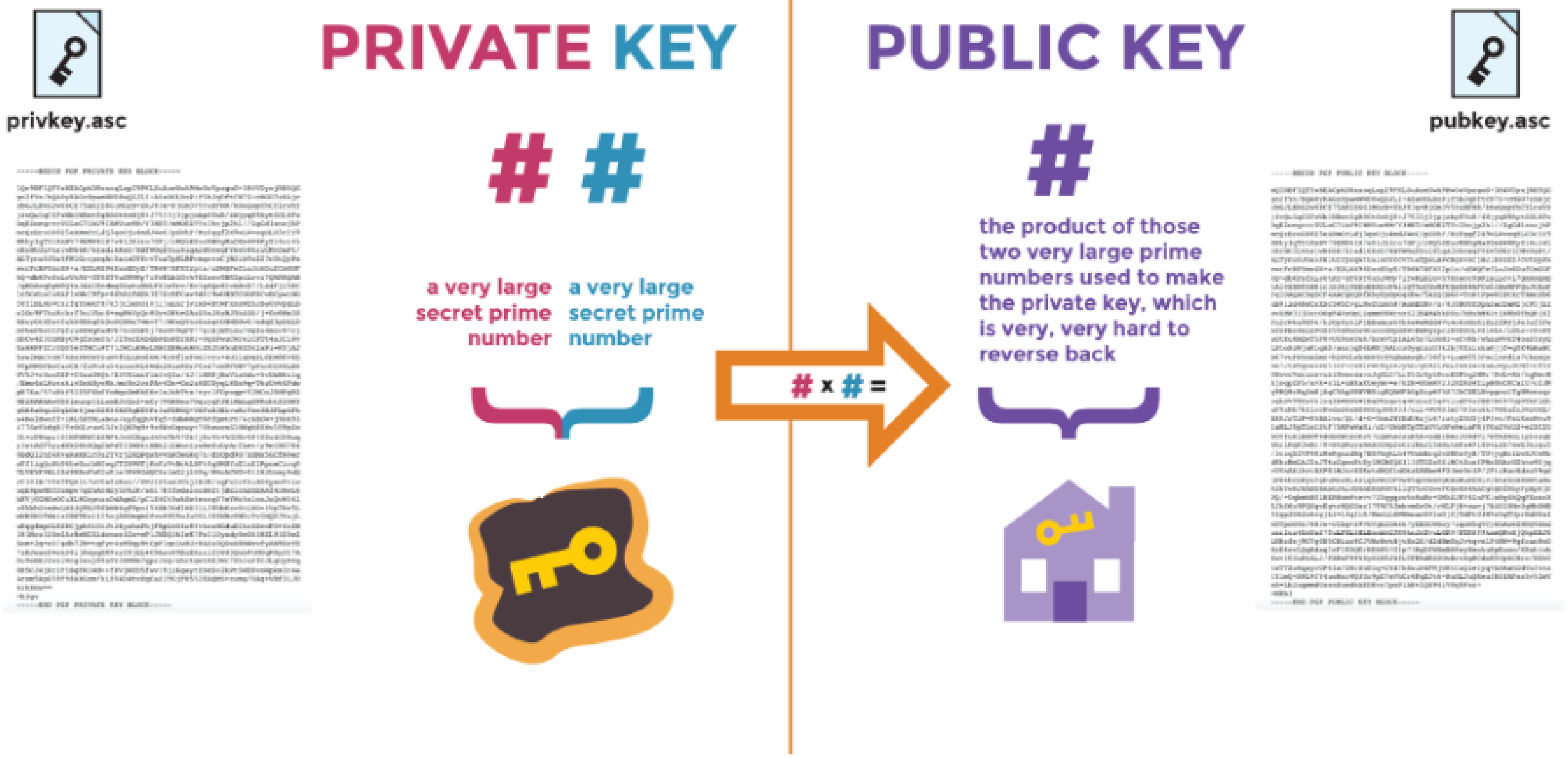
Bob can read the message using Alice's public key thus, it shows only Alice's private key was used to encrypt the message that only Alice knows. By this, it means Alice cannot deny later that she is the one who sent the message because no other people hold Alice's private key except herself.

Public Key Cryptosystem



1. **Public key encryption system has two components, a public key and a private key.**
2. **Encryption works by taking a message and applying a mathematical operation** to it to get a random-looking number.
3. **Decryption takes the random looking number** and applies a different operation to get back to the original number.
4. **Encryption with the public key** can only be undone by decrypting with the private key.

Public Key Cryptosystem



privkey.asc

-----BEGIN PGP PRIVATE KEY BLOCK-----

-----END PGP PRIVATE KEY BLOCK-----

PRIVATE KEY

# #

a very large secret prime number

a very large secret prime number

x # =

PUBLIC KEY

#

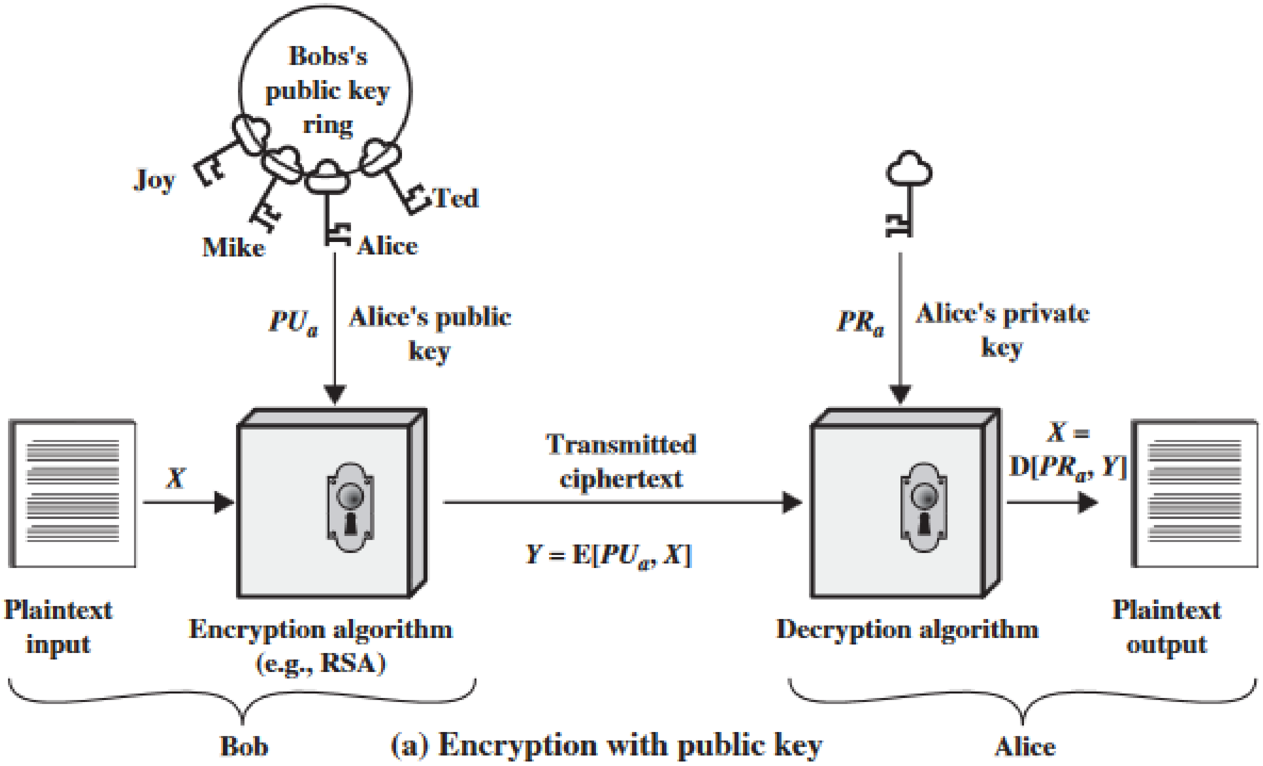
the product of those two very large prime numbers used to make the private key, which is very, very hard to reverse back

pubkey.asc

-----BEGIN PGP PUBLIC KEY BLOCK-----

-----END PGP PUBLIC KEY BLOCK-----

Public Key Cryptosystem



(a) Encryption with public key

Public Key Cryptosystem

The essential steps are the following:

1. Each user generates a pair of keys to be used for the encryption and decryption of messages.

2. Each user places one of the two keys in a public register or other accessible file. This is the public key. The companion key is kept private. As suggested, each user maintains a collection of public keys obtained from others.

3. If Bob wishes to send a confidential message to Alice, Bob encrypts the message using Alice’s public key.

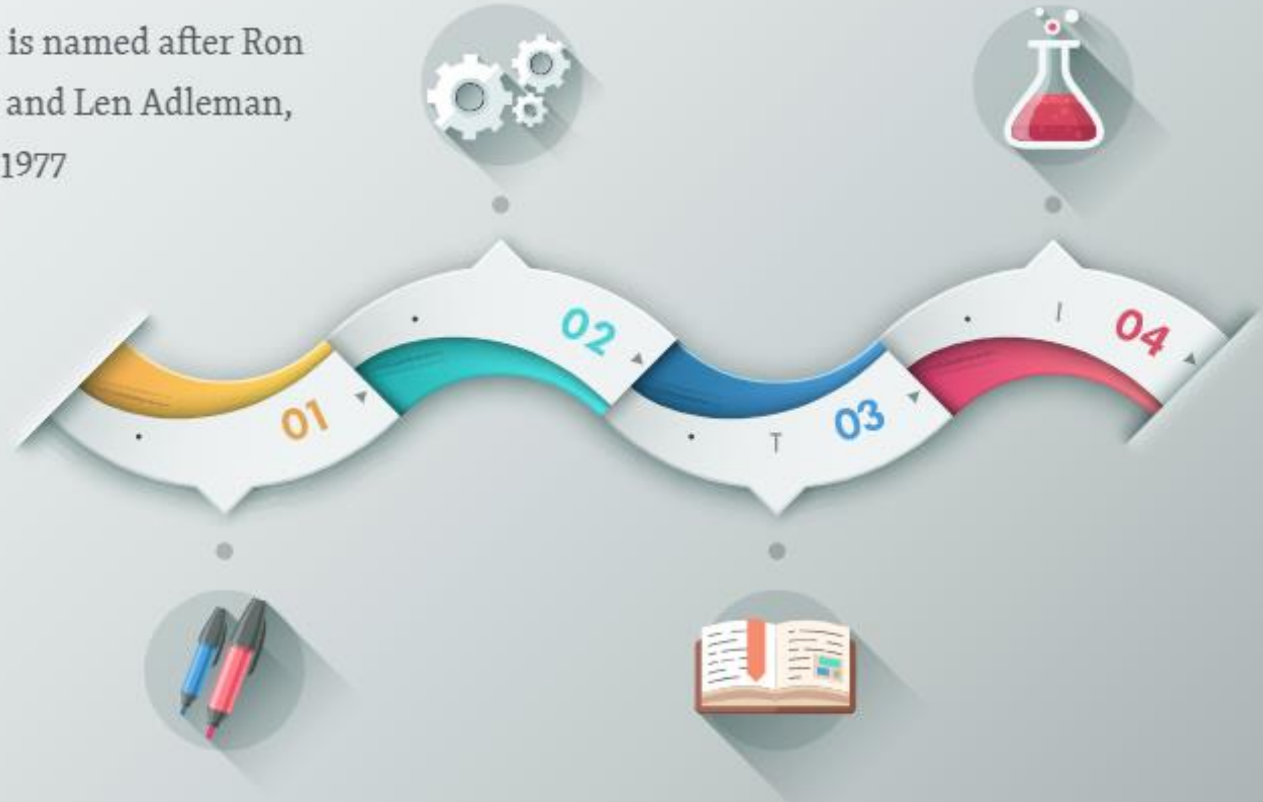
4. When Alice receives the message, she decrypts it using her private key. No other recipient can decrypt the message because only Alice knows Alice’s private key.

5. With this approach, all participants have access to public keys, and private keys are generated locally by each participant and therefore need never be distributed. As long as a user’s private key remains protected and secret, incoming communication is secure. At any time, a system can change its private key and publish the companion public key to replace its old public key.

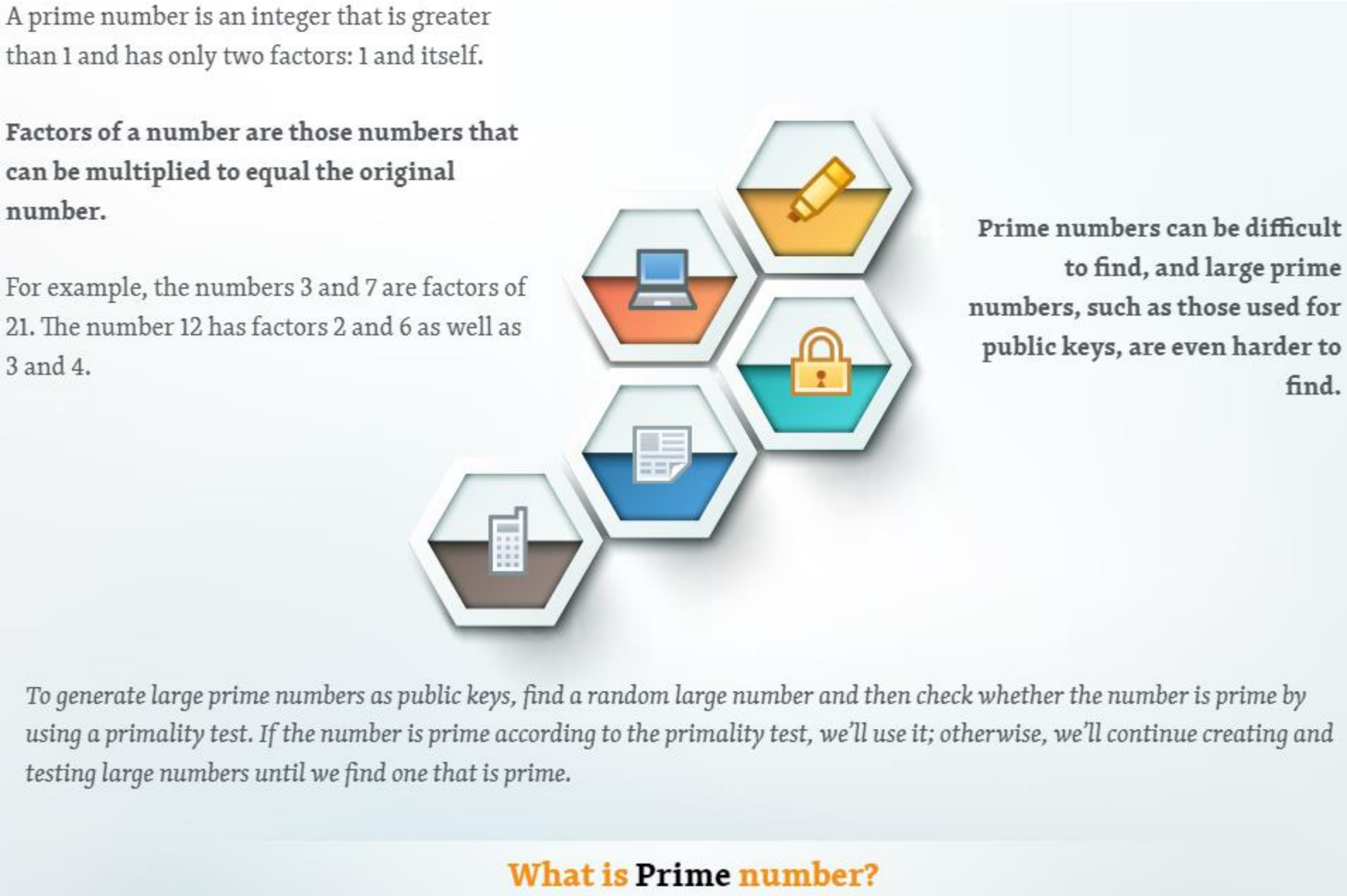
RSA Algorithm

RSA encryption is a public-key encryption technology developed by RSA Data Security. The RSA algorithm is based on the difficulty in **factoring very large prime numbers**. Based on this principle, the RSA encryption algorithm uses prime factorization as the trap door for encryption.

The RSA algorithm is named after Ron Rivest, Adi Shamir and Len Adleman, who invented it in 1977



The RSA algorithm can be used for both public key encryption and digital signatures. Its security is based on the difficulty of factoring large integers.



A prime number is an integer that is greater than 1 and has only two factors: 1 and itself.

Factors of a number are those numbers that can be multiplied to equal the original number.

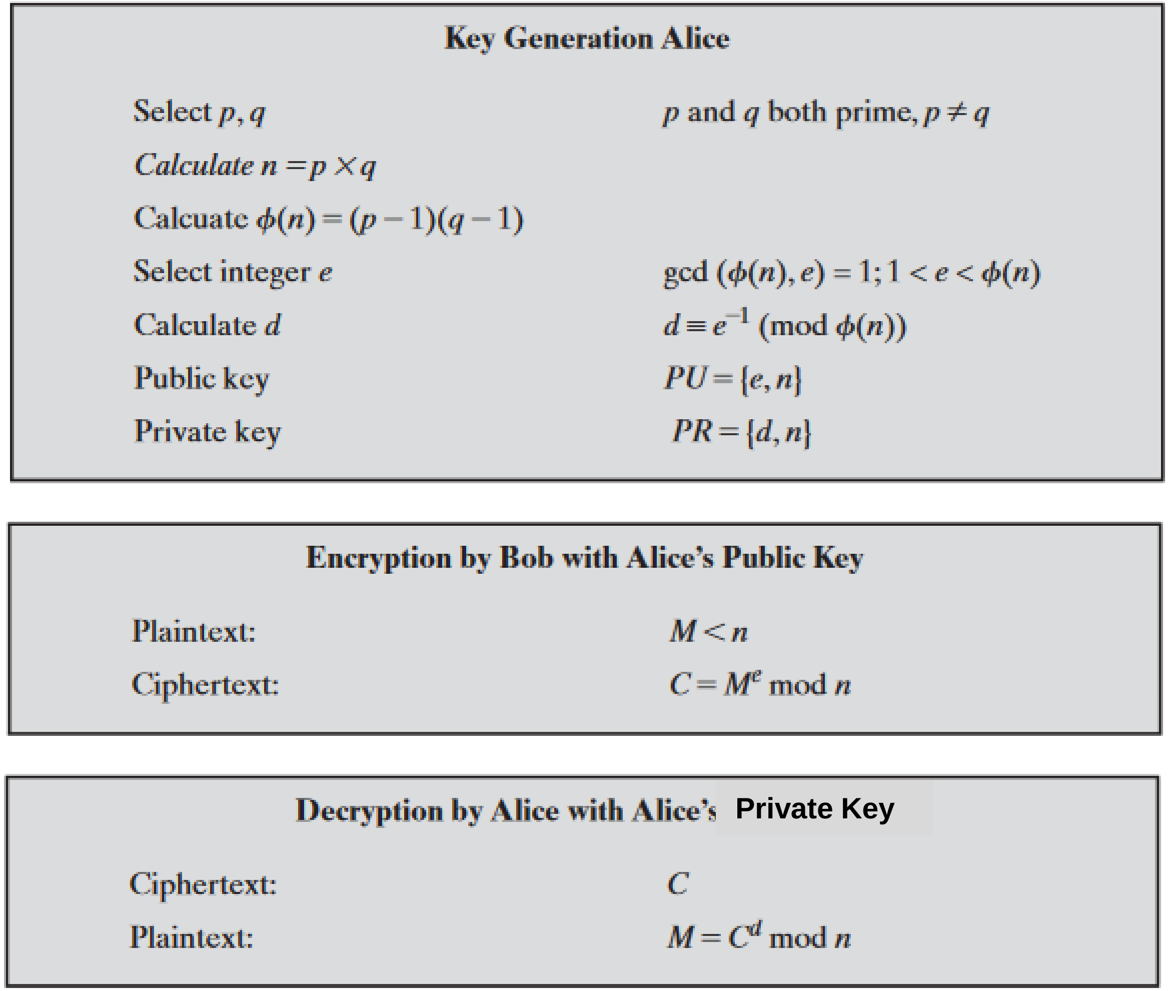
For example, the numbers 3 and 7 are factors of 21. The number 12 has factors 2 and 6 as well as 3 and 4.

Prime numbers can be difficult to find, and large prime numbers, such as those used for public keys, are even harder to find.

To generate large prime numbers as public keys, find a random large number and then check whether the number is prime by using a primality test. If the number is prime according to the primality test, we'll use it; otherwise, we'll continue creating and testing large numbers until we find one that is prime.

What is Prime number?

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

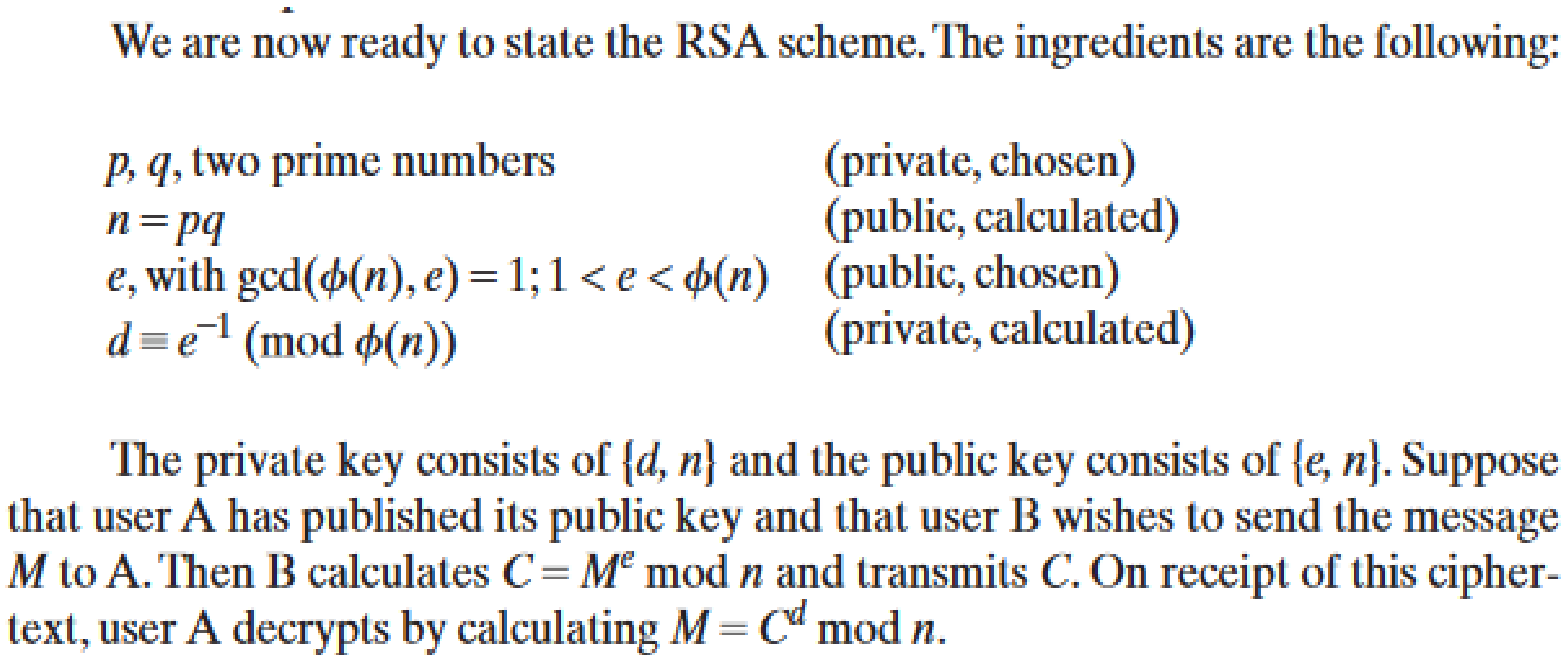
Plaintext:	$M < n$
Ciphertext:	$C = M^e \bmod n$

Decryption by Alice with Alice's Private Key

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

Page 294 – 297 Cryptography
and Network Security Book

RSA Algorithm

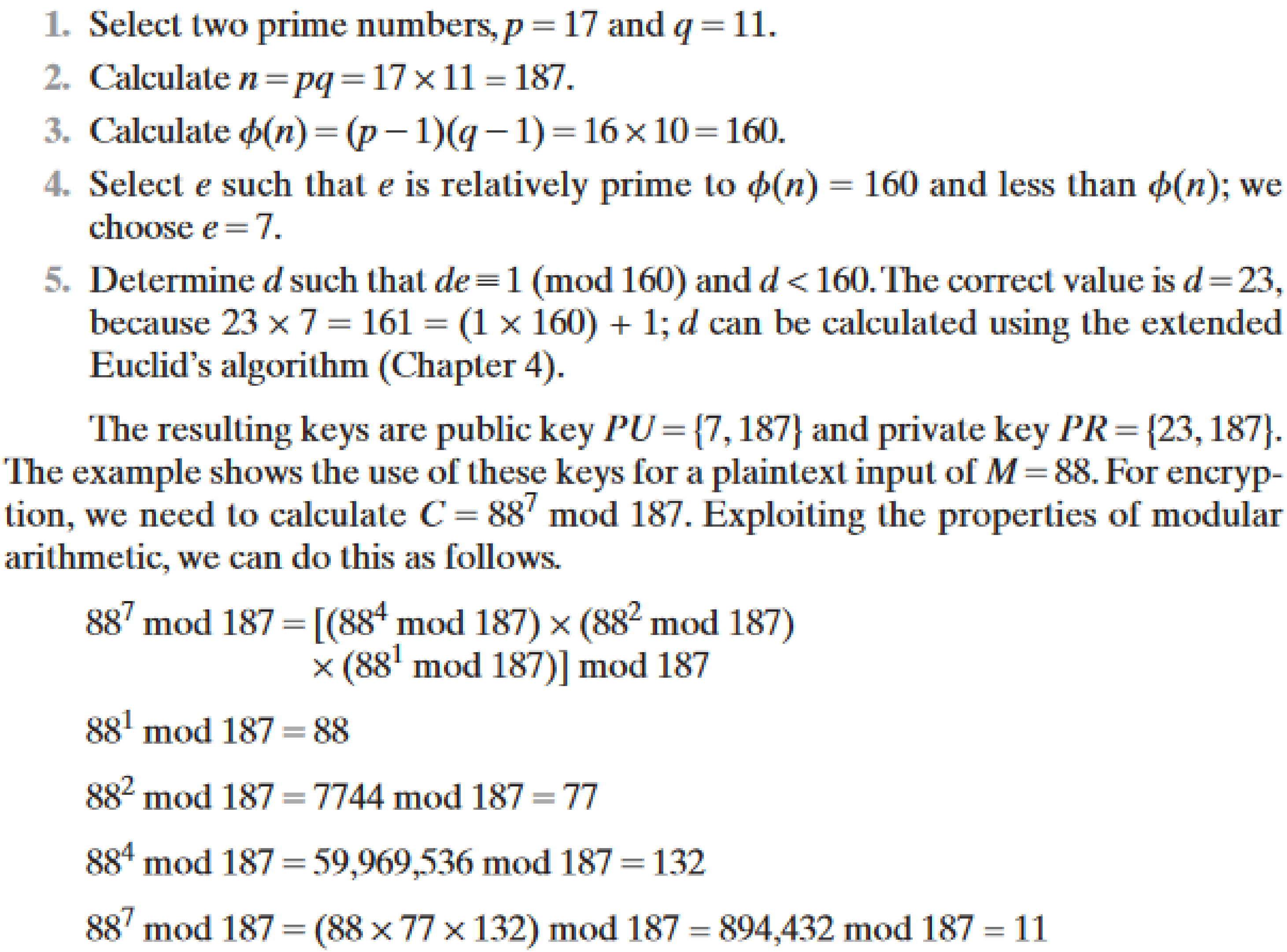


We are now ready to state the RSA scheme. The ingredients are the following:

p, q, two prime numbers	(private, chosen)
$n = pq$	(public, calculated)
e, with $\gcd(\phi(n), e) = 1; 1 < e < \phi(n)$	(public, chosen)
$d \equiv e^{-1} \pmod{\phi(n)}$	(private, calculated)

The private key consists of $\{d, n\}$ and the public key consists of $\{e, n\}$. Suppose that user A has published its public key and that user B wishes to send the message M to A. Then B calculates $C = M^e \bmod n$ and transmits C. On receipt of this ciphertext, user A decrypts by calculating $M = C^d \bmod n$.

RSA Algorithm - Example



1. Select two prime numbers, $p = 17$ and $q = 11$.
2. Calculate $n = pq = 17 \times 11 = 187$.
3. Calculate $\phi(n) = (p-1)(q-1) = 16 \times 10 = 160$.
4. Select e such that e is relatively prime to $\phi(n) = 160$ and less than $\phi(n)$; we choose $e = 7$.
5. Determine d such that $de \equiv 1 \pmod{160}$ and $d < 160$. The correct value is $d = 23$, because $23 \times 7 = 161 = (1 \times 160) + 1$; d can be calculated using the extended Euclid's algorithm (Chapter 4).

The resulting keys are public key $PU = \{7, 187\}$ and private key $PR = \{23, 187\}$. The example shows the use of these keys for a plaintext input of $M = 88$. For encryption, we need to calculate $C = 88^7 \bmod 187$. Exploiting the properties of modular arithmetic, we can do this as follows.

$$\begin{aligned} 88^7 \bmod 187 = {} & [(88^4 \bmod 187) \times (88^2 \bmod 187) \\ & \times (88^1 \bmod 187)] \bmod 187 \end{aligned}$$

$$88^1 \bmod 187 = 88$$

$$88^2 \bmod 187 = 7744 \bmod 187 = 77$$

$$88^4 \bmod 187 = 59{,}969{,}536 \bmod 187 = 132$$

$$88^7 \bmod 187 = (88 \times 77 \times 132) \bmod 187 = 894{,}432 \bmod 187 = 11$$

RSA Algorithm - Example

For decryption, we calculate $M = 11^{23} \bmod 187$:

$$11^{23} \bmod 187 = [(11^1 \bmod 187) \times (11^2 \bmod 187) \times (11^4 \bmod 187) \\ \times (11^8 \bmod 187) \times (11^8 \bmod 187)] \bmod 187$$

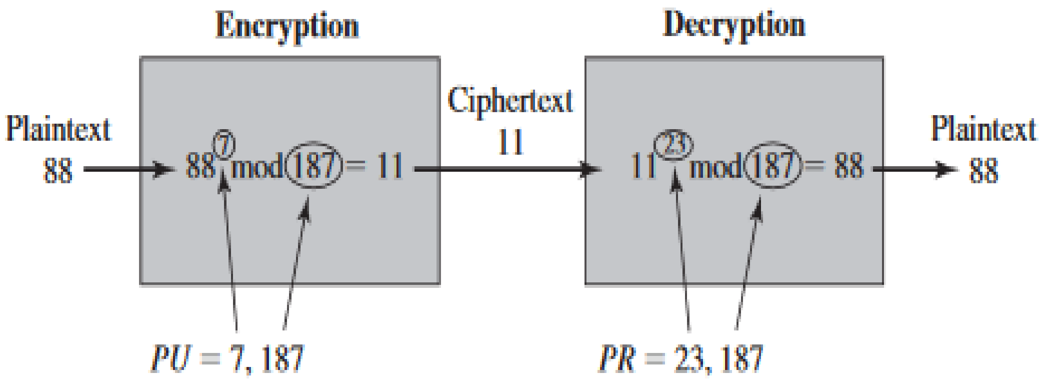
$$11^1 \bmod 187 = 11$$

$$11^2 \bmod 187 = 121$$

$$11^4 \bmod 187 = 14{,}641 \bmod 187 = 55$$

$$11^8 \bmod 187 = 214{,}358{,}881 \bmod 187 = 33$$

$$11^{23} \bmod 187 = (11 \times 121 \times 55 \times 33 \times 33) \bmod 187 = 79{,}720{,}245 \bmod 187 = 88$$



RSA Super Simplified

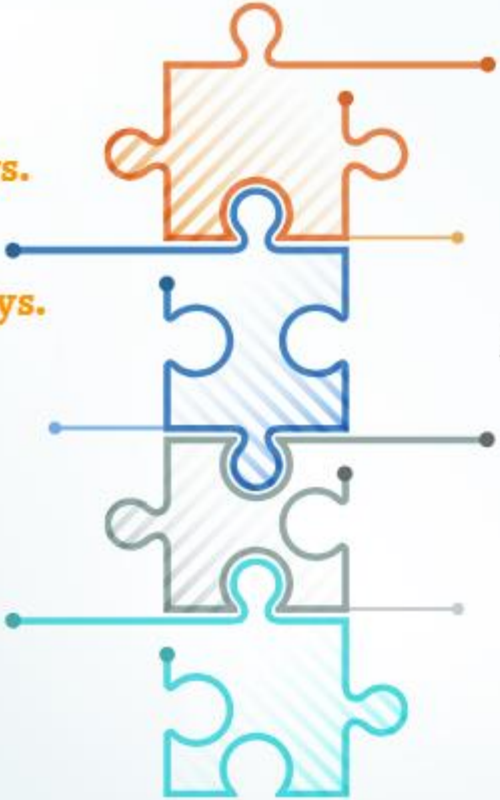
1. Select two prime numbers **p=7, q=19**.
2. **Calculate p * q = 133** *(Let's call this N)*

4. Calculate the Totient of N: (P-1)*(Q-1)
    1. (7-1) * (19-1) = 6 * 18 = **108** *(Let's call this T)*
5. **Select a Public Key**
    1. *The Public Key is a value which must match three requirements:*
    2. *It must be Prime. It must be less than the Totient. It must NOT be a factor of the Totient*
    3. *Let's select* **29 as our public key** *(Let's call this e)*
6. **Select a Private Key**
    1. *The Private Key only has to match one requirement: The Product of the Public Key and the Private Key when divided by the Totient, must result in a remainder of 1. The following formula must be true:*
    2. *(D*E) MOD T = 1 This validates (41*29) MOD 108 = 1*
    3. **41 is the private key**

Euler's Totient function Φ (n) for an input n is the count of numbers in {1, 2, 3, …, n-1} that are relatively prime to n, i.e., the numbers whose GCD (Greatest Common Divisor) with n is 1. A simple solution is to iterate through all numbers from 1 to n-1 and count numbers with gcd with n as 1.

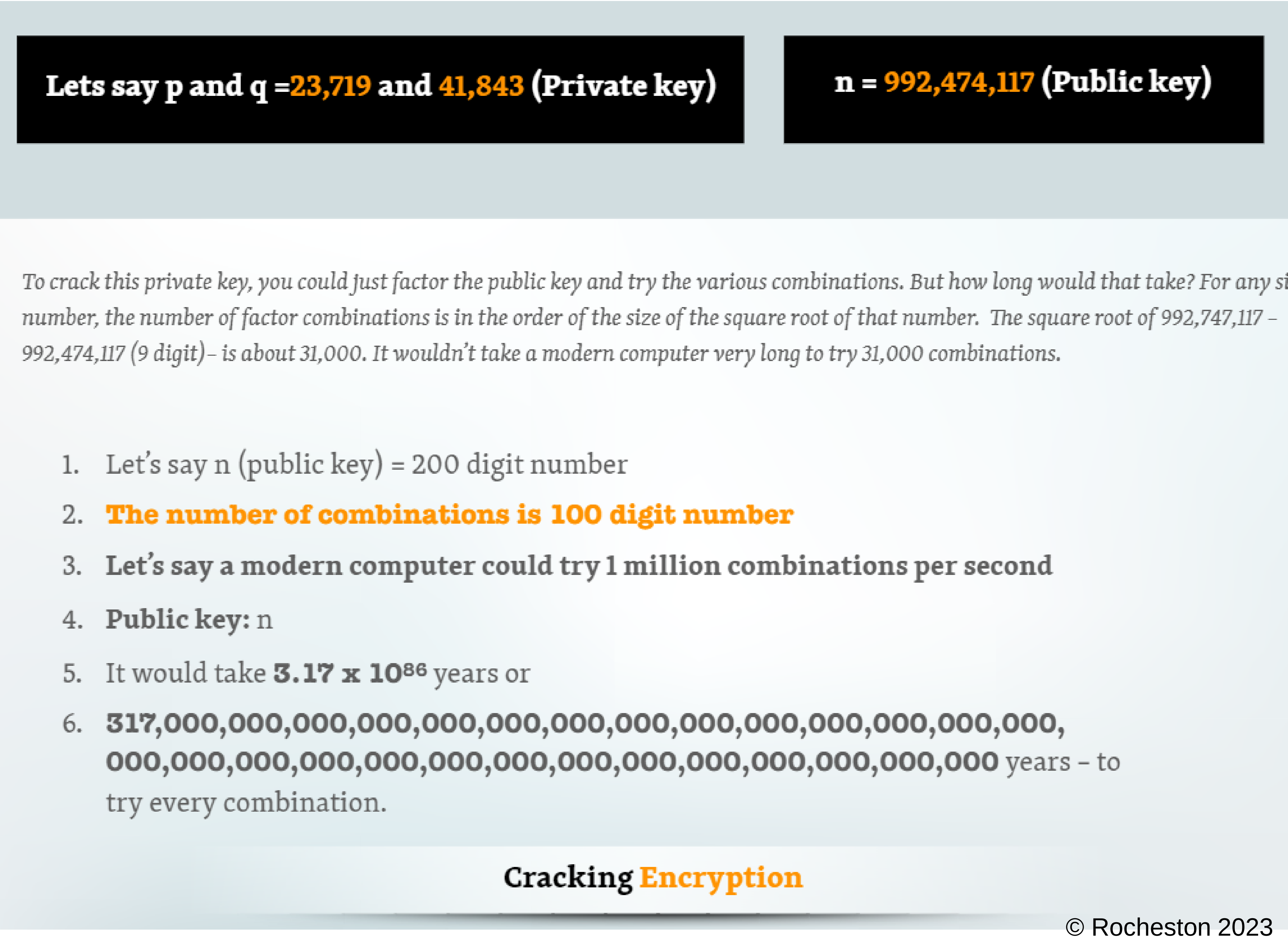
Two integers are relatively prime or Coprime when there are no common factors other than 1. This means that no other integer could divide both numbers evenly. Two integers (a & b) are called relatively prime to each other if gcd(a,b)=1. For example, 7 and 20 are relatively prime.

- ✓ 56-bit key was cracked in 250 days.
- ✓ 64-bit key was cracked in 1,757 days.
- ✗ 73-bit key? 2,080 years.
- ✗ 72-bit key is still being cracked
- ✗ 74-bit key? 4,160 years.



Cracking 128 bit key? Forget it.

Cracking Encryption



Lets say p and q =23,719 and 41,843 (Private key)

n = 992,474,117 (Public key)

To crack this private key, you could just factor the public key and try the various combinations. But how long would that take? For any si number, the number of factor combinations is in the order of the size of the square root of that number. The square root of 992,747,117 – 992,474,117 (9 digit)– is about 31,000. It wouldn't take a modern computer very long to try 31,000 combinations.

1. Let's say n (public key) = 200 digit number
2. **The number of combinations is 100 digit number**
3. **Let's say a modern computer could try 1 million combinations per second**
4. **Public key:** n
5. It would take **3.17 x 10^{86}** years or
6. **317,000,000,000,000,000,000,000,000,000,000,000,000,000,000,000, 000,000,000,000,000,000,000,000,000,000,000,000,000,000,000** years – to try every combination.

Cracking Encryption

National Institute of Standards and Technology (NIST) reviews the academic literature on attacking cryptographic algorithms and makes recommendations on the actual security provided by different algorithms.

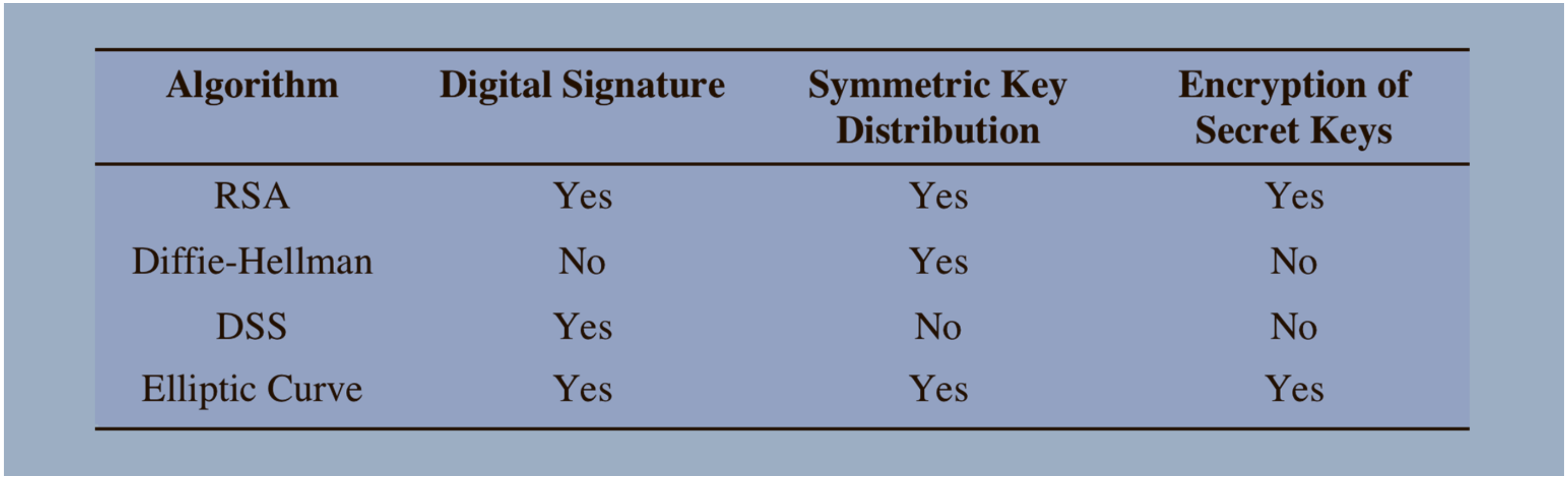
Bits of Security	Symmetric Key Algorithm	Corresponding Hash Function	Corresponding RSA Key Size	Corresponding ECC Key Size
80	Triple DES (2 keys)	SHA-1	1024	160
112	Triple DES (3 keys)	SHA-224	2048	224
128	AES-128	SHA-256	3072	256
192	AES-192	SHA-384	7680	384
256	AES-256	SHA-512	15360	512

NIST Recommended Key Sizes

Real Life Implementation

- We can consider a practical situation that describes asymmetric cryptography as used in real life.
- Suppose a bank accepts many requests for transactions from its customers over an insecure network.
- The bank can have a private key-public key pair. The bank can publish its public key to all its customers.
- The customers can use this public key of the bank to encrypt messages before they send them to the bank. The bank can decrypt all these encrypted messages with its private key, which remains with itself.

Applications of Public-Key Cryptosystems



Algorithm	Digital Signature	Symmetric Key Distribution	Encryption of Secret Keys
RSA	Yes	Yes	Yes
Diffie-Hellman	No	Yes	No
DSS	Yes	No	No
Elliptic Curve	Yes	Yes	Yes

Requirements for Public-Key Cryptosystems

computationally easy to create key pairs

computationally easy for sender knowing public key to encrypt messages

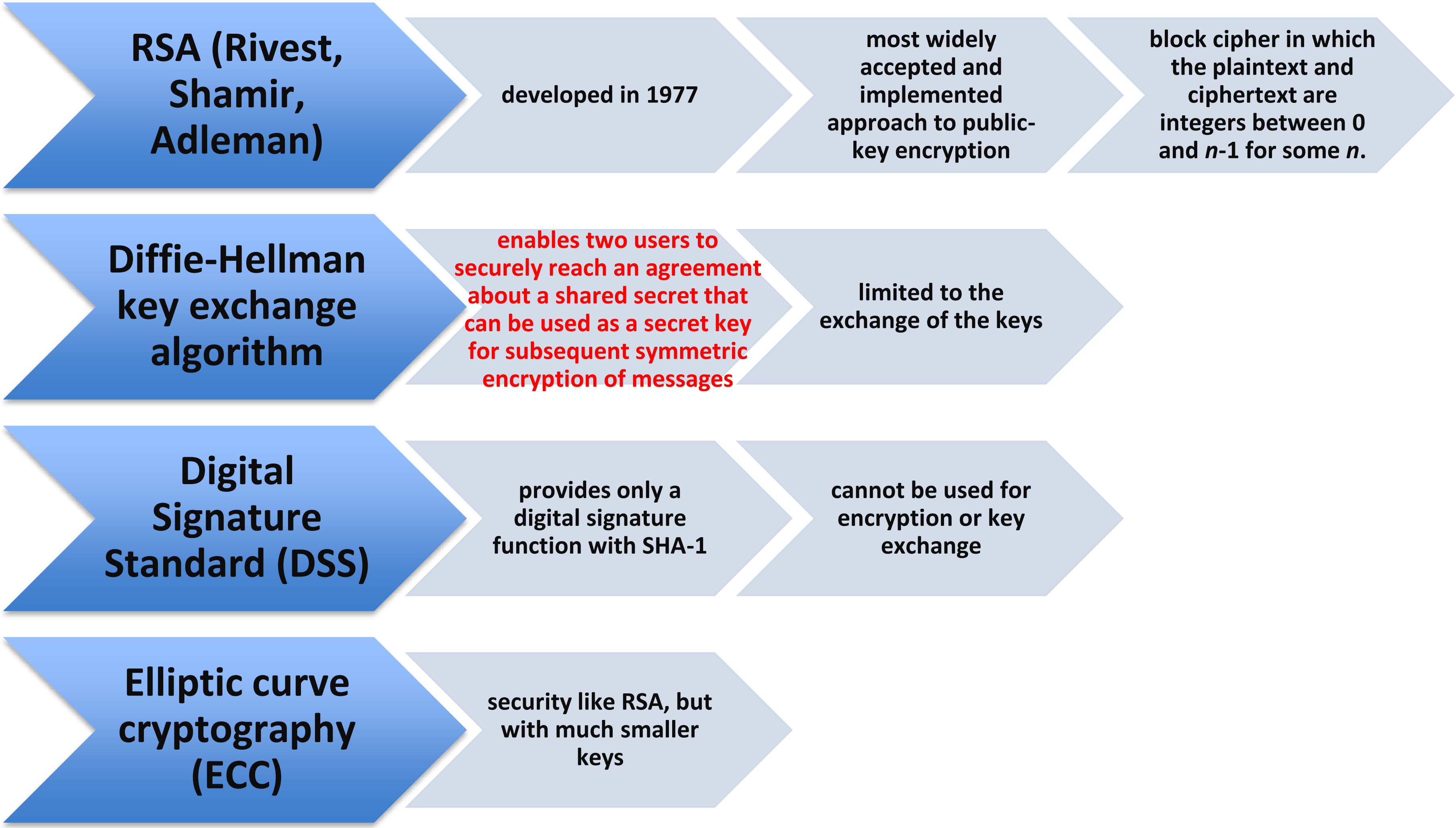
computationally easy for receiver knowing private key to decrypt ciphertext

computationally infeasible for opponent to determine private key from public key

computationally infeasible for opponent to otherwise recover original message

useful if either key can be used for each role

Asymmetric Encryption Algorithms



RSA (Rivest, Shamir, Adleman)

developed in 1977

most widely accepted and implemented approach to public-key encryption

block cipher in which the plaintext and ciphertext are integers between 0 and *n*-1 for some *n*.

Diffie-Hellman key exchange algorithm

enables two users to securely reach an agreement about a shared secret that can be used as a secret key for subsequent symmetric encryption of messages

limited to the exchange of the keys

Digital Signature Standard (DSS)

provides only a digital signature function with SHA-1

cannot be used for encryption or key exchange

Elliptic curve cryptography (ECC)

security like RSA, but with much smaller keys

Key Exchange/Distribution

- How nice to combine two cryptography mechanisms? Problems before?
- Combination must meet the following objective:
  - √ Solution completely secure
  - √ Encryption & decryption -> not take a long time
  - √ Generated cipher text -> compact in size
  - √ Solution scale to many users
  - √ Key distribution problem must be solved
- In practice, symmetric & asymmetric are combined -> very efficient security solution.

Key Exchange/Distribution

- Suppose you need to send a protected message to someone you do not know and who does not know you.
- Eg. Online income tax return.
- You want the information to be protected.
- And you do not necessarily know the person who is receiving the information.
- Situation: being able to exchange encryption key nobody can intercept it.

Diffie-Hellman Key Exchange

- Whitefield Diffie and Martin Hellman
  - devised an amazing solution to the problem
  - called Diffie-Hellman Key Exchange/Agreement Algorithm (DHKE/AA)
- The beauty of this scheme – two parties who want to communicate securely can agree on a symmetric key using this technique.
- However, must be noted DHKE/AA can be used only for key agreement but not for encryption or decryption of messages.

Example 1

- Alice and Bob want to agree upon a key to be used for encrypting / decrypting messages that be exchanged between them.
- 1. Firstly, Alice and Bob agree on one prime number, q and one root number, α. These 2 integers need not be kept secret. Alice and Bob can use an insecure channel to agree on them.

Let $q = 11$, $\alpha = 7$.

Diffie-Hellman Algorithm

Let $q = 11$, $\alpha = 7$

2. Alice chooses a private large random numberX_A, and calculates A: $Y_A = \alpha^{XA} \text{mod } q$

Let $X_A = 3$. Then we have, $Y_A = \alpha^{XA} \text{mod } q = 7^3 \text{ mod } 11 = 343 \text{ mod } 11 = 2$

3. Alice sends the number Y_A (public) to Bob

Alice sends 2 to Bob

4. Bob independently chooses another private large random number X_B and calculates B such that: $Y_B = \alpha^{XB} \text{ mod } q$

Let $X_B = 6$. Then we have, $Y_B = \alpha^{XB} \text{ mod } q = 7^6 \text{ mod } 11 = 117649 \text{ mod } 11 = 4$

Diffie-Hellman Algorithm

5. Bob sends the number Y_B (public) to Alice

Bob sends 4 to Alice

6. Alice now computes the secret key K1 as follows: K1 = $(Y_B)^{XA}$ mod q

We have, K1 = 4^3 mod 11 = 64 mod 11 = 9

7. Bob now computes the secret key K2 as follows: K2 = $(Y_A)^{XB}$ mod q

We have, K2 = 2^6 mod 11 = 64 mod 11 = 9

- Therefore, in this case, we have K1 = K2 = K.

Example 2: Diffie-Hellman

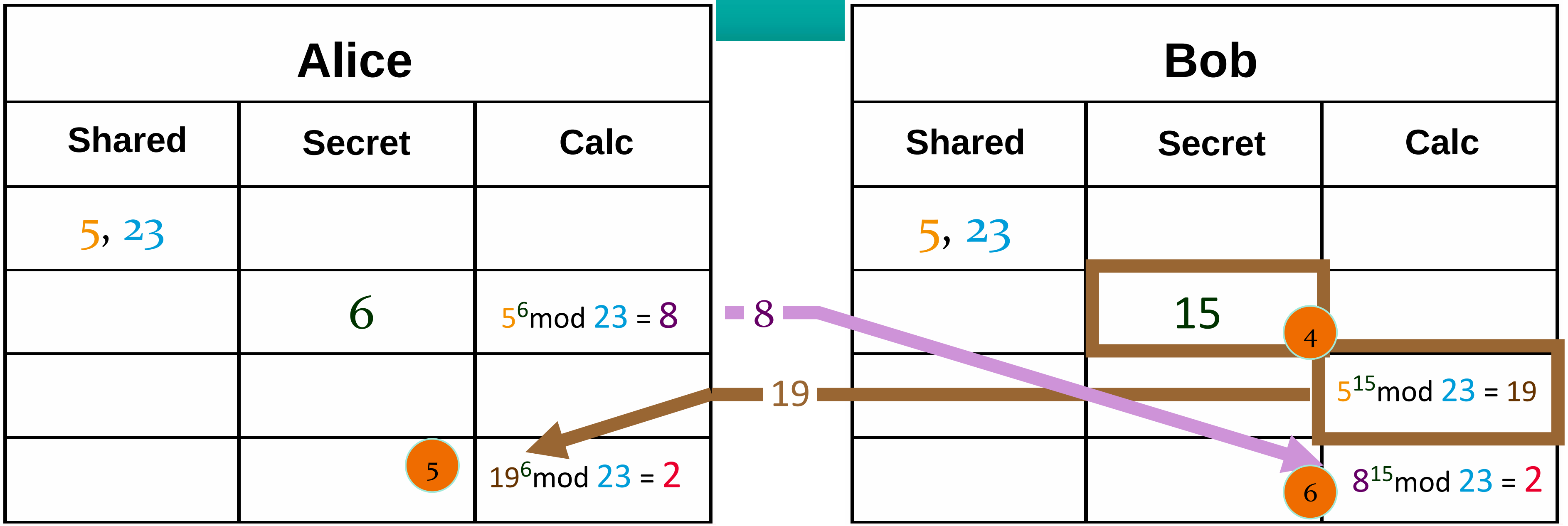
Alice		
Shared	Secret	Calc
1 23, 5		
2	6	3 5^6mod 23 = 8

8 →

Bob		
Shared	Secret	Calc
1 23, 5		

1. Alice and Bob agree to use the same two numbers. For example, the base number g=5 and prime number p=23
2. Alice now chooses a secret number x=6.
3. Alice performs the DH algorithm: g^x modulo p = (5^6 modulo 23) = 8 (Y) and sends the new number 8 (Y) to Bob.

Example 2: Diffie-Hellman



Alice		
Shared	Secret	Calc
5, 23		
	6	5^6mod 23 = 8
		19^6mod 23 = 2

Bob		
Shared	Secret	Calc
5, 23		
	15	
		5^{15}mod 23 = 19
		8^{15}mod 23 = 2

4. Meanwhile Bob has also chosen a secret number x=15, performed the DH algorithm: g^x modulo p = (5^{15} modulo 23) = 19 (Y) and sent the new number 19 (Y) to Alice.

5. Alice now computes Y^x modulo p = (19^6 modulo 23) = 2.

6. Bob now computes Y^x modulo p = (8^6 modulo 23) = 2.

The result (2) is the same for both Alice and Bob.
This number can now be used as a shared secret key by the encryption algorithm.

Diffie-Hellman Secure Part

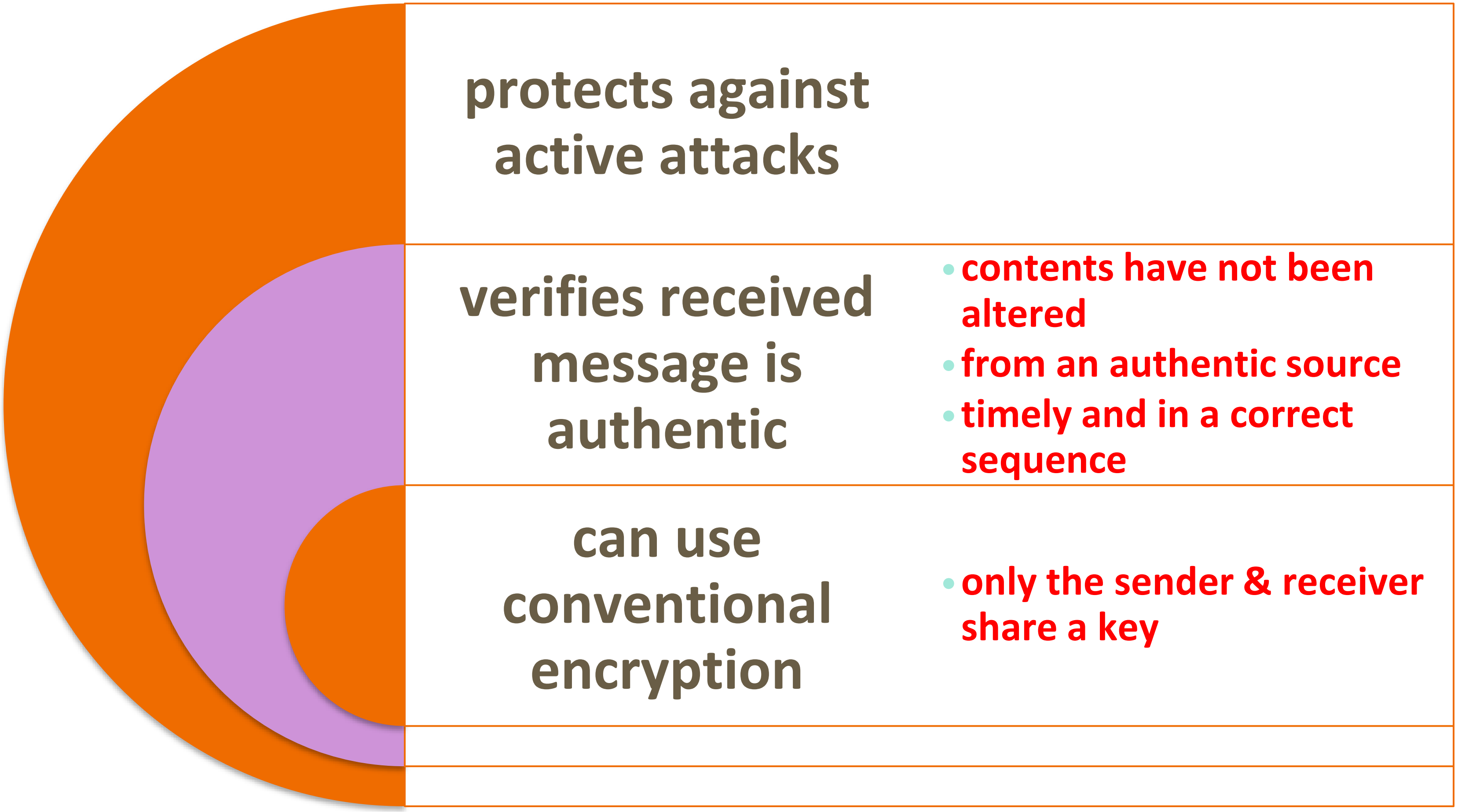
- An obvious question now is, if Alice and Bob can both calculate K independently, so can an attacker! What prevents this?
- The fact is, that Alice and Bob exchange q, α, Y_A and Y_B (public). Based on these values, $\mathbf{X_A}$ (Alice's private key) and $\mathbf{X_B}$ (Bob's private key) cannot be calculated easily.
- Rouge XYZ knows: q, α, Y_A and Y_B
- Try calculate = $Y_A(\text{pubA})^{\mathbf{X_B}(\text{privB})}$ mod n = $2^{\mathbf{X_B}}$ mod 11

  = $Y_B(\text{pubB})^{\mathbf{X_A}(\text{privA})}$ mod n = $4^{\mathbf{X_A}}$ mod 11
- $\mathbf{X_B}$ (Bob private key) = 6 , $\mathbf{X_A}$ (Alice private key) = 3

Diffie-Hellman Algorithm

- Mathematically, the calculations do find out **X_A** and **X_B** are extremely complicated, if they are sufficiently large numbers.
- Consequently, an attacker cannot calculate **X_A** and **X_B**, and therefore cannot derive key, **K**.

Message Authentication



protects against active attacks

verifies received message is authentic

- contents have not been altered
- from an authentic source
- timely and in a correct sequence

can use conventional encryption

- only the sender & receiver share a key

Message Authentication Codes

Achieved integrity

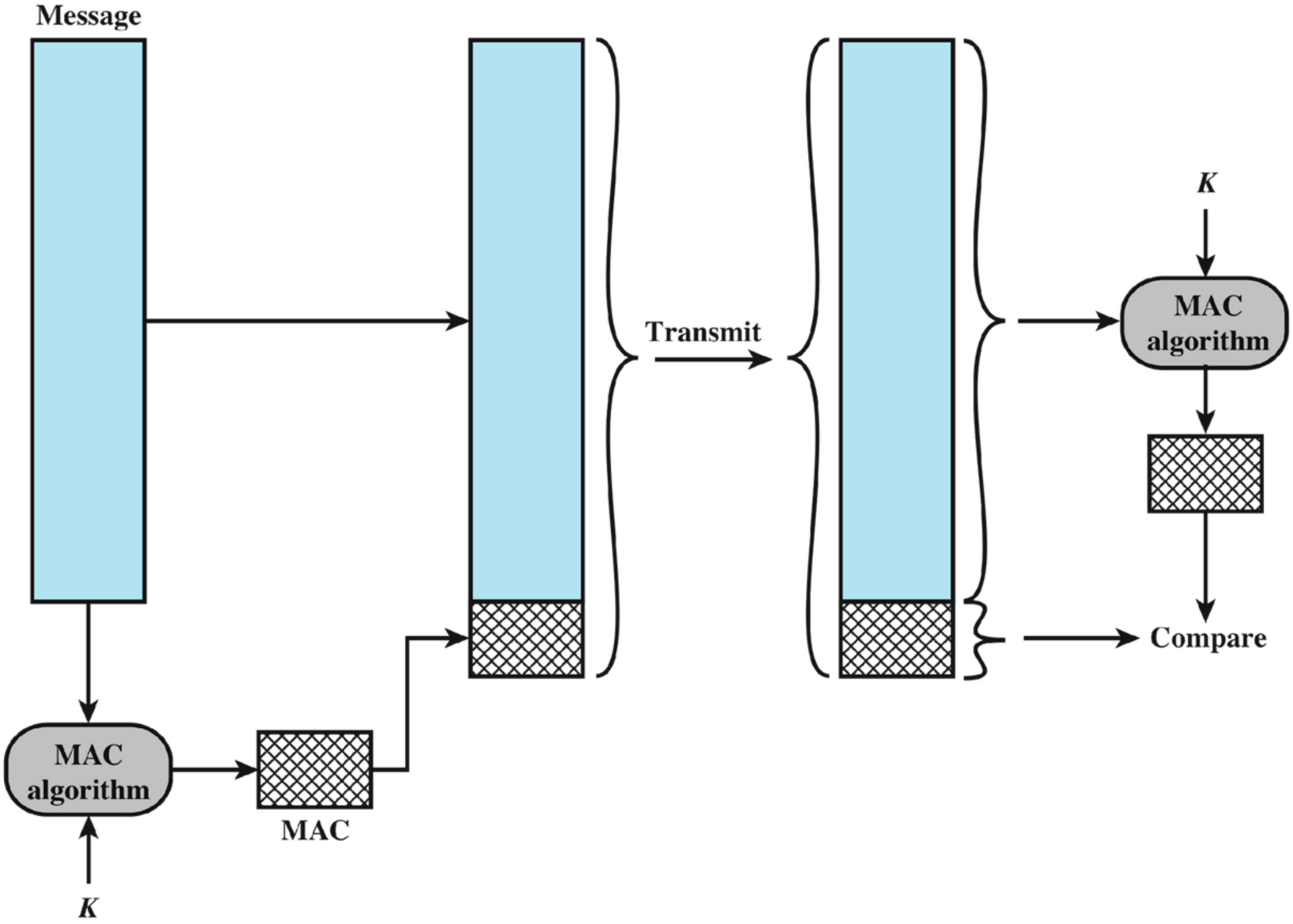


Figure 2.4 Message Authentication Using a Message Authentication Code (MAC). The MAC is a function of an input message and a secret key.

Secure Hash Function

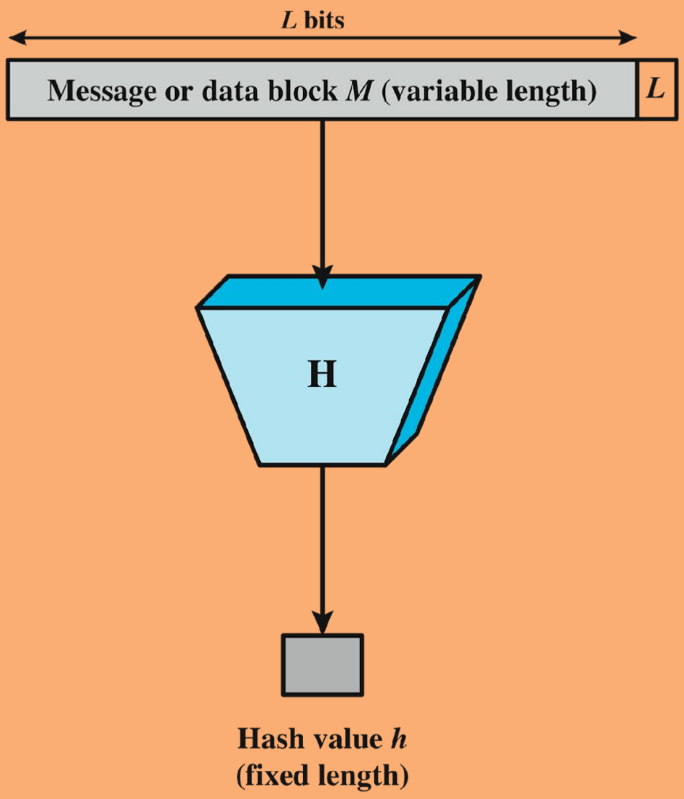
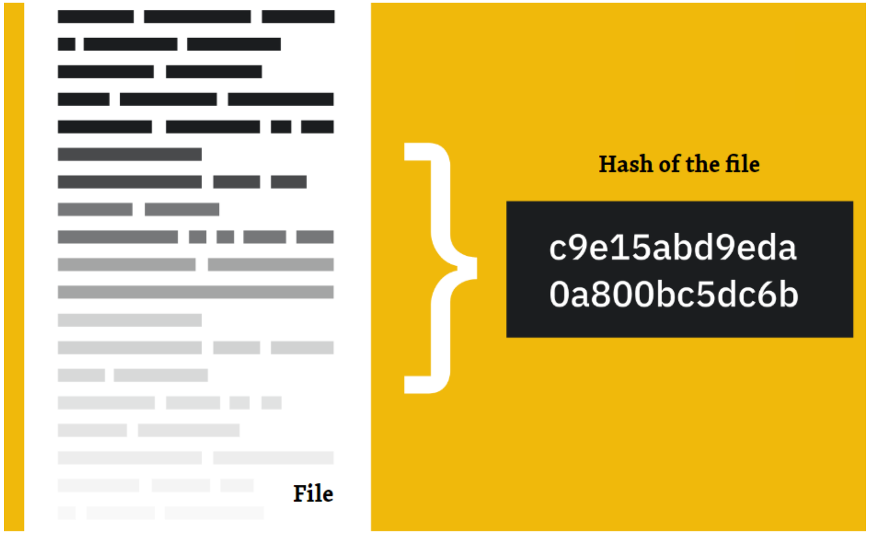


Figure 2.5 Block Diagram of Secure Hash Function; $h = \mathrm{H}(M)$



```
$ echo -n ‘magic is abracadabra’ | sha256sum

$ sha256sum <filename>
```

MD5 - An MD5 hash function encodes a string of information and encodes it into a 128-bit fingerprint.

SHA-2 – SHA-2, 256/512-bit developed by the National Security Agency (NSA), is a cryptographic hash function.

CRC32 – A cyclic redundancy check (CRC) is an error-detecting code often used for detection of accidental changes to data.

Hashing is an algorithm that calculates a fixed-size bit string value from a file. A good hashing algorithm would exhibit a property called the avalanche effect, where the resulting hash output would change significantly or entirely even when a single bit or byte of data within a file is changed.

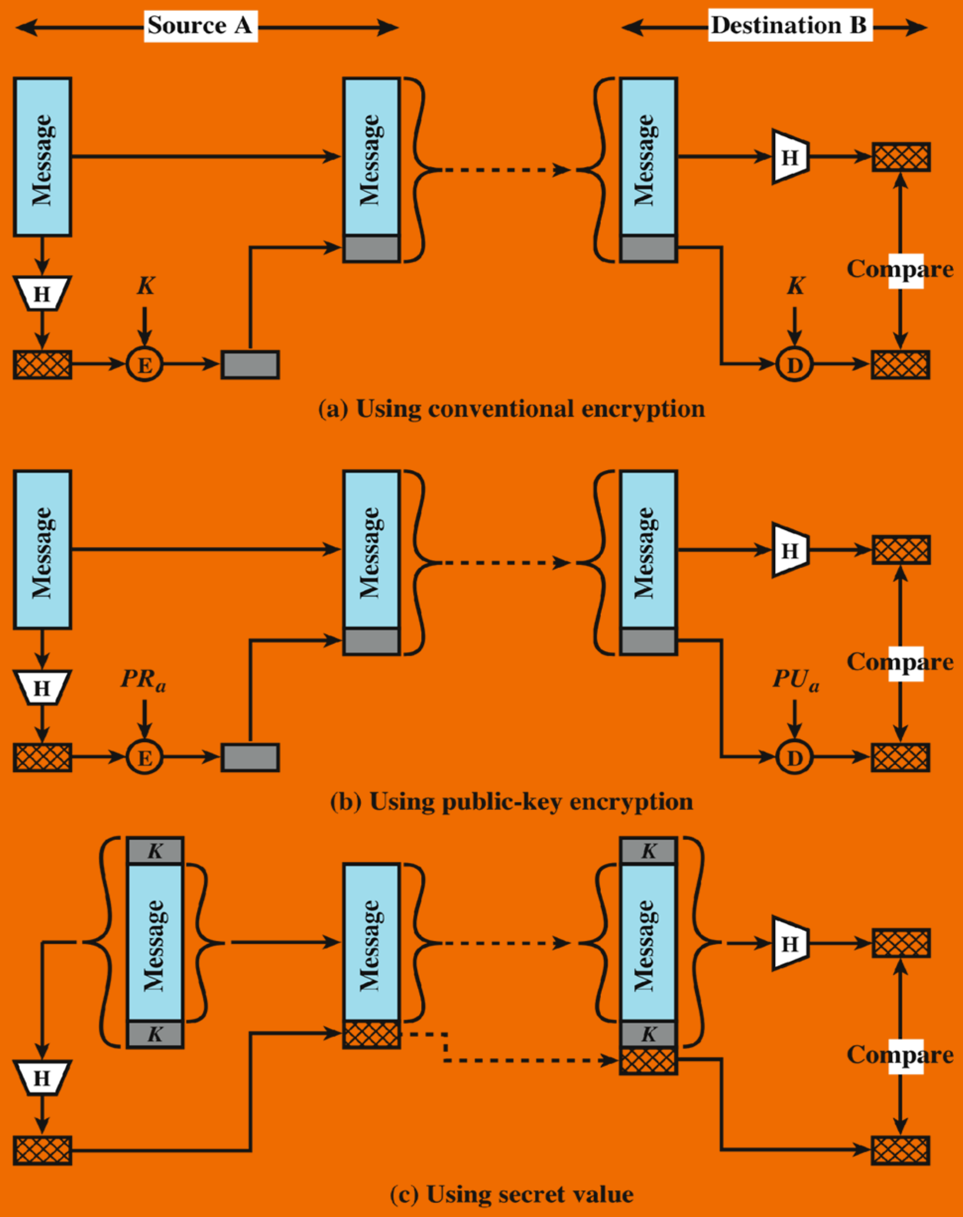
What is Hashing?

Security of Hash Function

- √ There are two approaches to attack a secure hash function:
  - ○ Cryptanalysis – exploit logical weaknesses in the algorithm
  - ○ Brute-force attack – the strength of the hash function depends solely on the length of the hash code produced by the algorithm
- √ SHA, MD5 and MD4 most widely used hash algorithm.
- √ Additional secure hash function applications:
  - ● Passwords – the hash of a password is stored by an operating system.
  - ● Intrusion detection – store H (F) for each file on a system and secure the hash values.

Message Authentication Using a One-Way Hash Function

Achieved integrity



(a) Using conventional encryption

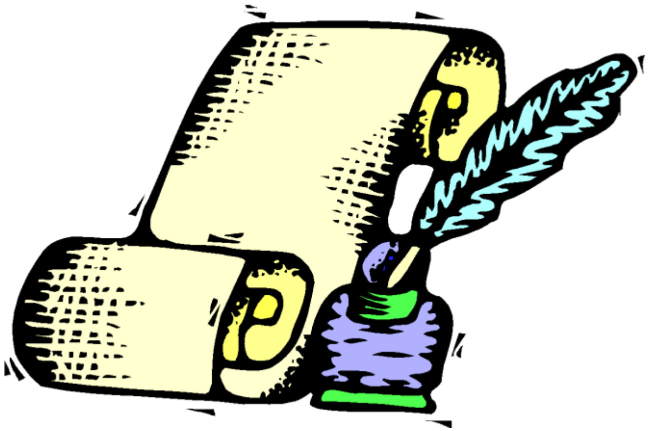
(b) Using public-key encryption

(c) Using secret value

Figure 2.6 Message Authentication Using a One-Way Hash Function. The hash function maps a message into a relatively small, fixed-size block.

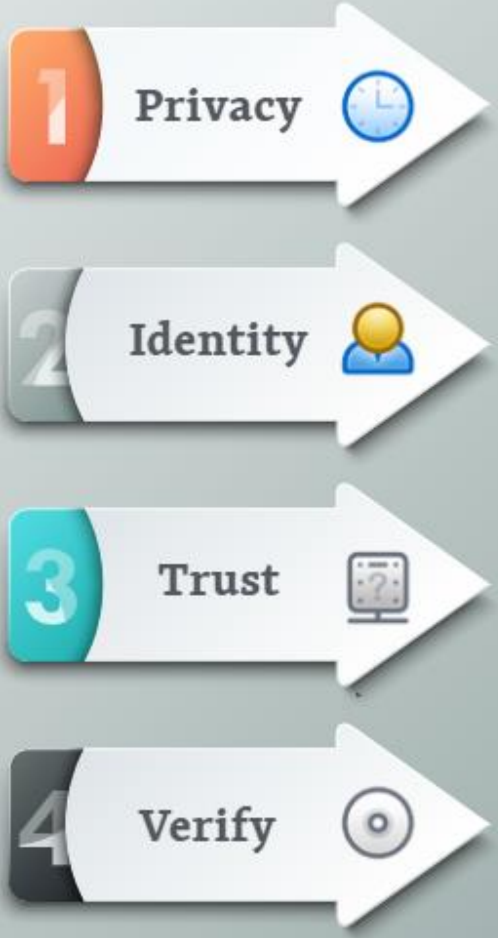
Digital Signature

- Used for authenticating both source and data integrity.
- Created by encrypting hash code with private key.
- Does not provide confidentiality
  - even in the case of complete encryption
  - message is safe from alteration but not eavesdropping



Digital signatures are the most advanced and secure type of electronic signature.

You can use them to comply with the most **demanding legal and regulatory requirements** because they provide the highest levels of assurance about each signer's identity and the authenticity of the documents they sign.



1 Privacy

2 Identity

3 Trust

4 Verify

There may be plenty of times and reasons why you might want or need to digitally sign a file.

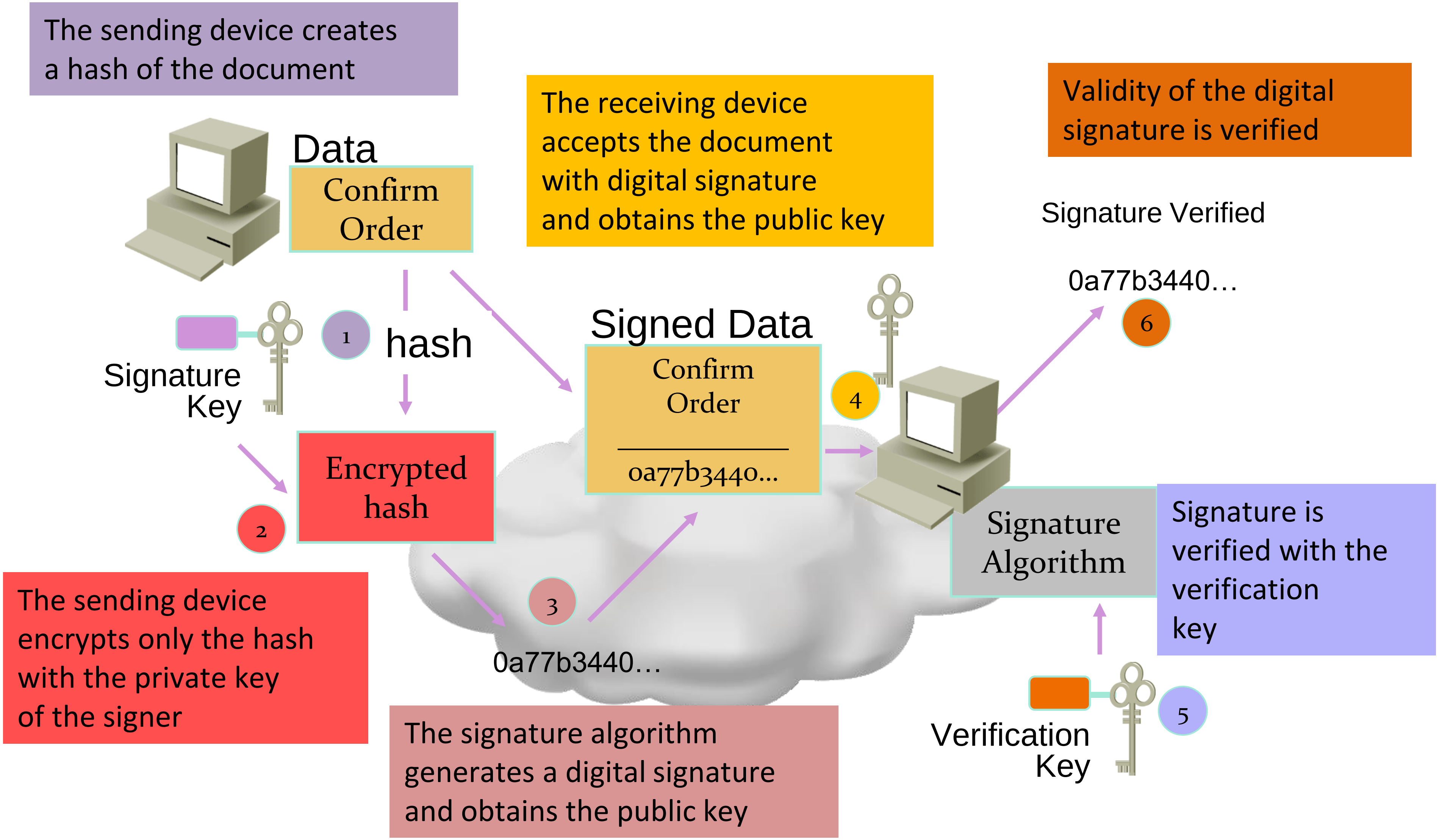
For instance, say you're sharing sensitive information and you want to ensure the recipient can trust the file that contains the data.

One way is to digitally sign that file prior to sending it to the client.

Once the client receives the file, they can verify the signature and if they trust the digital signature, extract the file within and use the data.

Digital Signature

Digital Signature Process


The sending device creates a hash of the document

The sending device encrypts only the hash with the private key of the signer

The signature algorithm generates a digital signature and obtains the public key

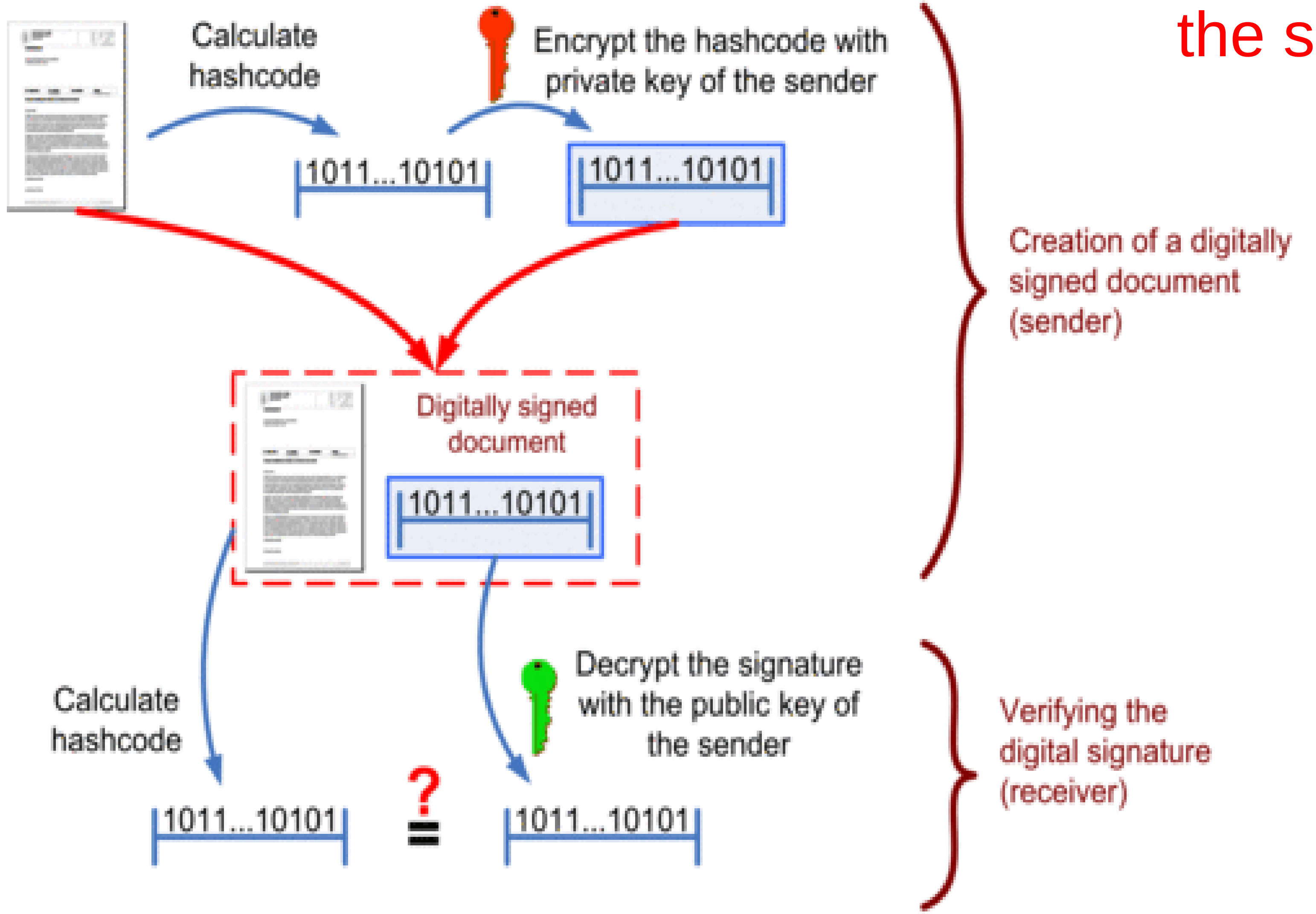
The receiving device accepts the document with digital signature and obtains the public key

Signature is verified with the verification key

Validity of the digital signature is verified

Achieved integrity and authenticated sender at the same time

Creating and verifying a digital signature


Calculate hashcode
Encrypt the hashcode with private key of the sender
1011...10101
1011...10101
Creation of a digitally signed document (sender)
Digitally signed document
1011...10101
Calculate hashcode
Decrypt the signature with the public key of the sender
Verifying the digital signature (receiver)
1011...10101
?
=
1011...10101

If the calculated hashcode does not match the result of the decrypted signature, either the document was changed after signing, or the signature was not generated with the private key of the alleged sender.

Summary

- Symmetric encryption
  - conventional or single-key only type used prior to public-key
  - five parts: plaintext, encryption algorithm, secret key, ciphertext, and decryption algorithm
  - two attacks: cryptanalysis and brute force
  - most used algorithms are block ciphers (DES, triple DES, AES)
- Public-key encryption (Asymmetric)
  - based on mathematical functions
  - Asymmetric six ingredients: plaintext, encryption algorithm, public and private key, ciphertext, and decryption algorithm

- Terminology – plaintext, cipher text, encryption, decryption, key, cryptography, cryptanalysis, cryptology.
- Classical cryptography – substitution and transposition cipher.
- RSA
- DHKE/AA
- Message authentication
- Hash function
- Digital signature