CS211FZ: Data Structures and Algorithms II
10- Cryptography
RSA

LECTURER: Chris Roadknight

Chris.Roadknight@mu.ie

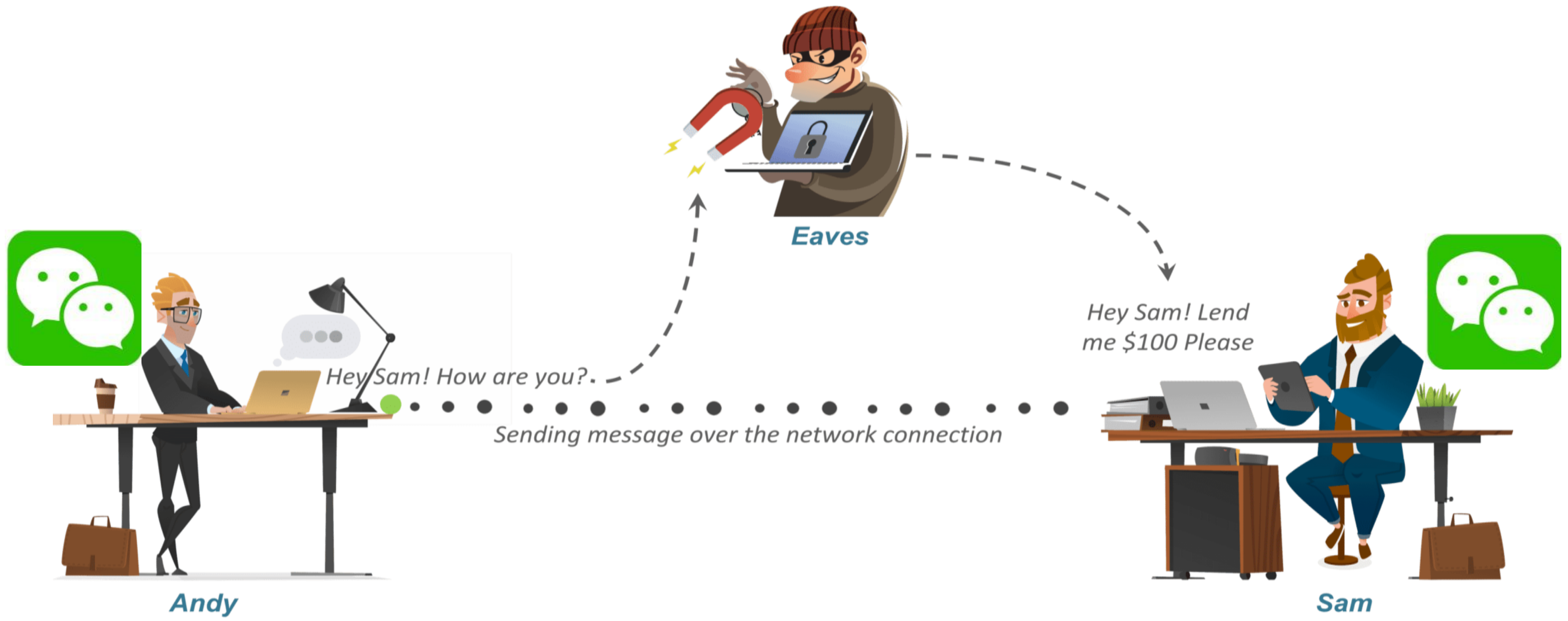
Introduction

Cryptography - Background

- Cryptography is derived from the Greek word, which means "Hidden Secrets."
- Through cryptography, we convert our data into unreadable secret codes, called "Cipher Text" and can read this data only, which will have the secret key to decrypt it.
- Cryptography is used since ancient times (3000 BC – AD 750).

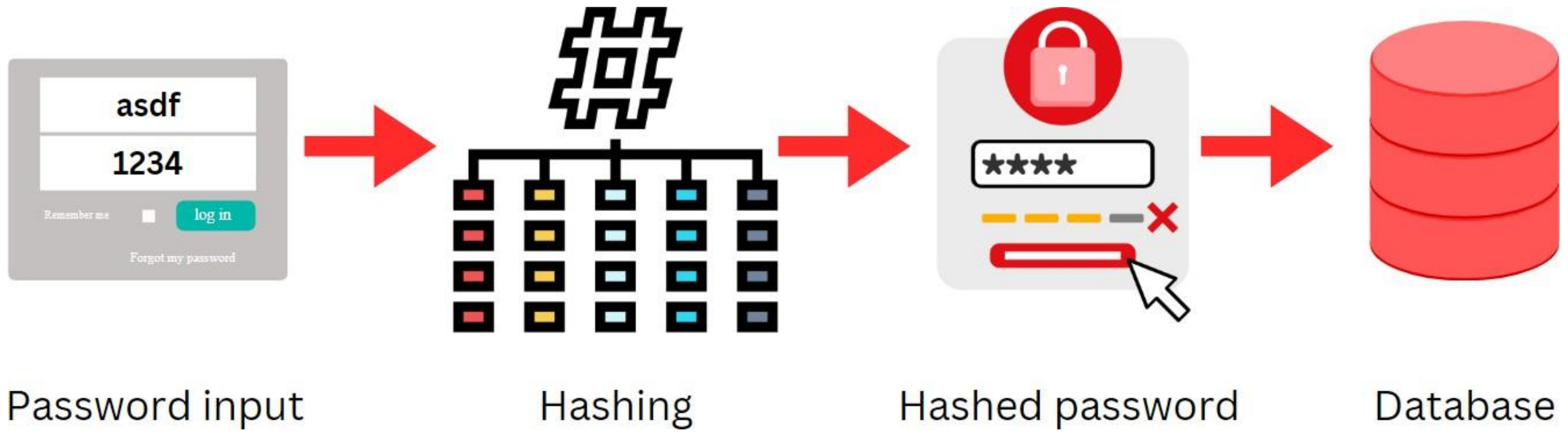
Problem Example 1: Secure Messaging

- Andy wants to send a private message to his friend Sam, ensuring no one else can read it.
- Eaves, a hacker, gains access to the communication channel and can attempt to read or change the message.
- If Eaves gets private information, the consequences could be severe.
- Encryption **(cryptography)** ensures only Sam can read Andy's message securely.


Eaves
Hey Sam! How are you?
Hey Sam! Lend me $100 Please
Sending message over the network connection
Andy
Sam

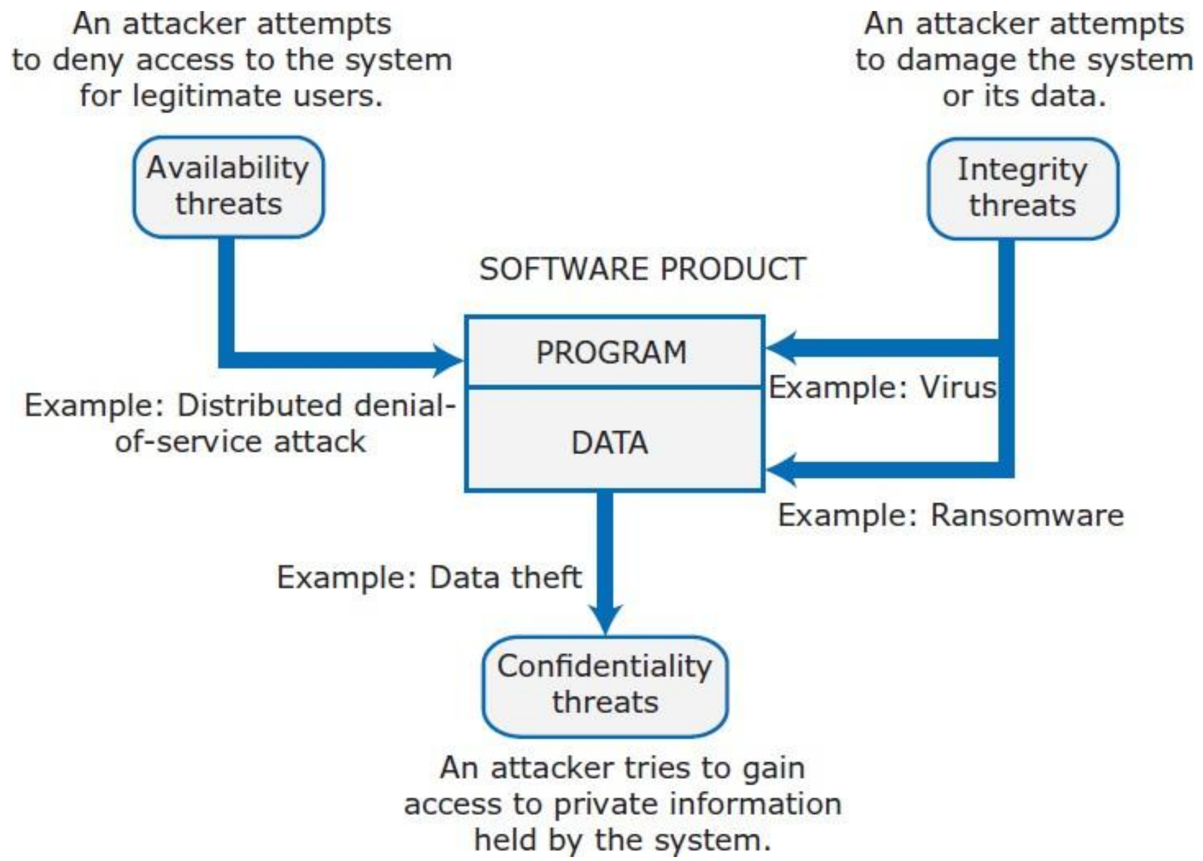
Problem Example 2: Password Storage

- An online platform must securely store user passwords to prevent unauthorized access.
- The platform hashes passwords using a **cryptographic hash function** instead of saving them in plain text.
- Even if hackers steal the database, they cannot reverse-engineer the original passwords from the hashed versions.



Main Threats to Software Systems

- Figure below shows the three main types of threat that software systems face.

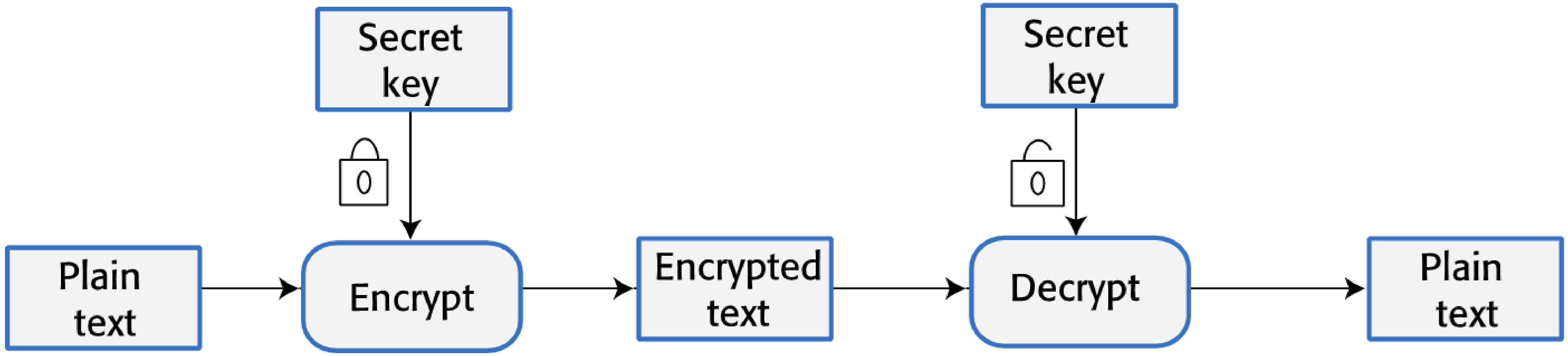


An attacker attempts to deny access to the system for legitimate users.
Availability threats
An attacker attempts to damage the system or its data.
Integrity threats
SOFTWARE PRODUCT
PROGRAM
DATA
Example: Distributed denial-of-service attack
Example: Virus
Example: Ransomware
Example: Data theft
Confidentiality threats
An attacker tries to gain access to private information held by the system.

Cryptography - Introduction

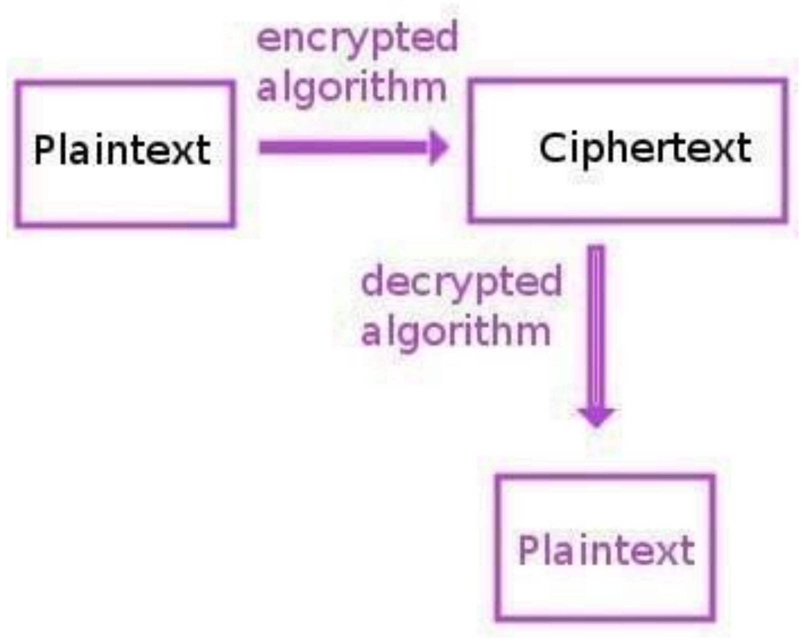
- Cryptography is an approach to make communication secure.
- It involves encoding messages or data to prevent unauthorized access
- It employs encryption algorithms and keys to transform plaintext into ciphertext.
- It involves encryption, decryption, hashing, and digital signatures.

Cryptography - Process



Cryptography Process – Plain Text

- In cryptography, plaintext is usually ordinary readable text before it is encrypted into cipher text, or readable text after it is decrypted.



Cryptography Process – Encryption

- Encryption is the process of making a document unreadable by applying an algorithmic transformation to it.
- It used for data encryption, authentication and digital signatures.
- One can decode the encrypted text by applying the reverse transformation.
- Modern encryption techniques are such that you can encrypt data so that it is practically uncrackable using currently available technology.

Cryptography Process – Encryption

- However, history has demonstrated that apparently strong encryption may be crackable when new technology becomes available.

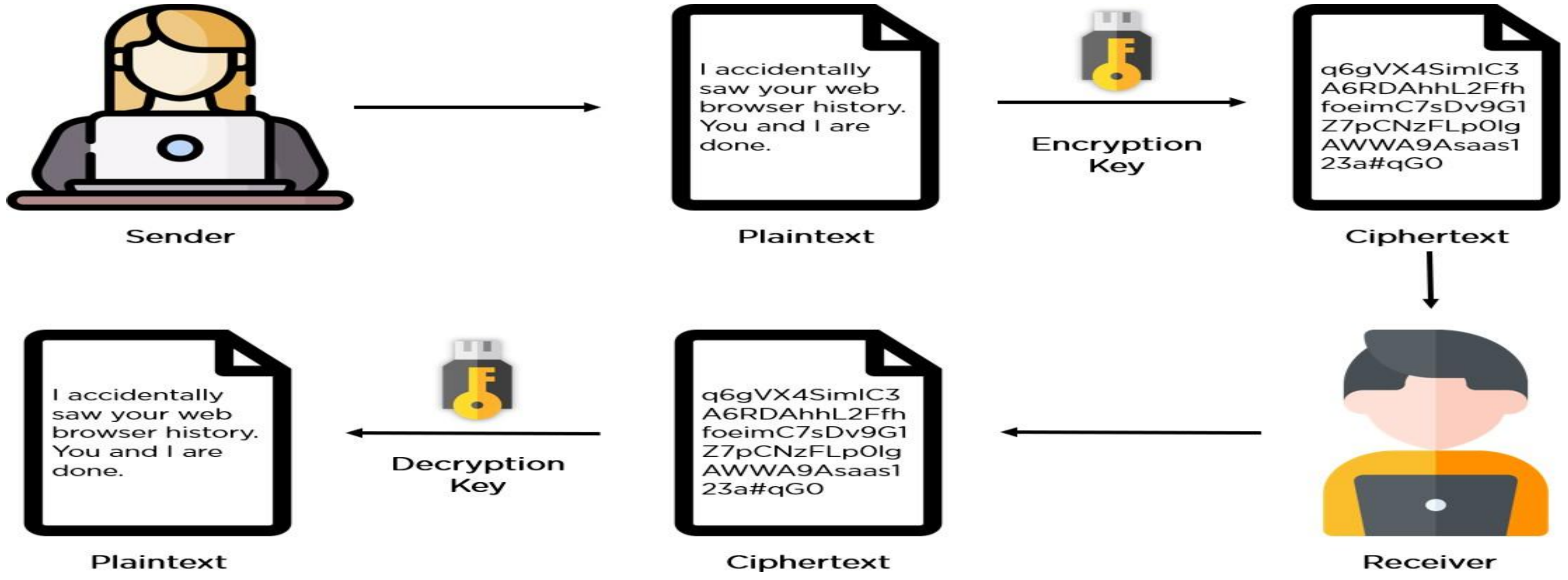
- If commercial quantum systems become available, we will have to use a completely different approach to encryption on the Internet.

Cryptography Process – Key generation

- Keys are crucial in cryptography.
- A key is a piece of information used by the encryption algorithm to transform plaintext into cipher text.

Cryptography Process – Transmission

- Once the message is encrypted, it can be transmitted over insecure channels such as the internet or public networks.
- The message is in cipher text form, even if intercepted it would remain unreadable without the decryption key.


Sender
Plaintext
I accidentally saw your web browser history. You and I are done.
Encryption Key
Ciphertext
q6gVX4SimIC3 A6RDAhhL2Ffh foeimC7sDv9G1 Z7pCNzFLp0lg AWWA9Asaas1 23a#qGO
Receiver
Ciphertext
q6gVX4SimIC3 A6RDAhhL2Ffh foeimC7sDv9G1 Z7pCNzFLp0lg AWWA9Asaas1 23a#qGO
Decryption Key
Plaintext
I accidentally saw your web browser history. You and I are done.

Cryptography Process – Cipher Text

- The cipher text is the result of encryption performed on plaintext using an algorithm.
- Cipher text is also known as encrypted or encoded information because it contains a form of the original plaintext that is unreadable by a human or computer without decryption.

Cryptography Process – Decryption

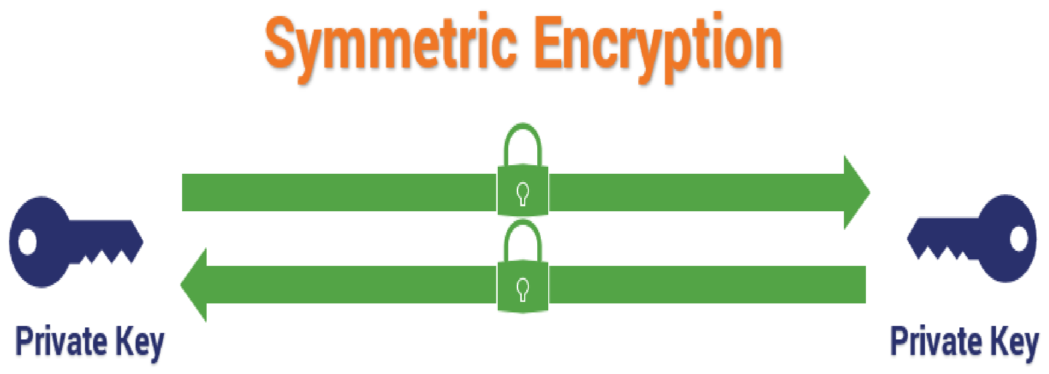
- Decryption algorithms are mathematical procedures used to transform encrypted data (Cipher text) back into its original form, known as plaintext.
- These algorithms typically require a decryption key, which may be the same used to encrypt the data.
- Only users who possess the correct decryption key can successfully decrypt the ciphertext and retrieve the original message.

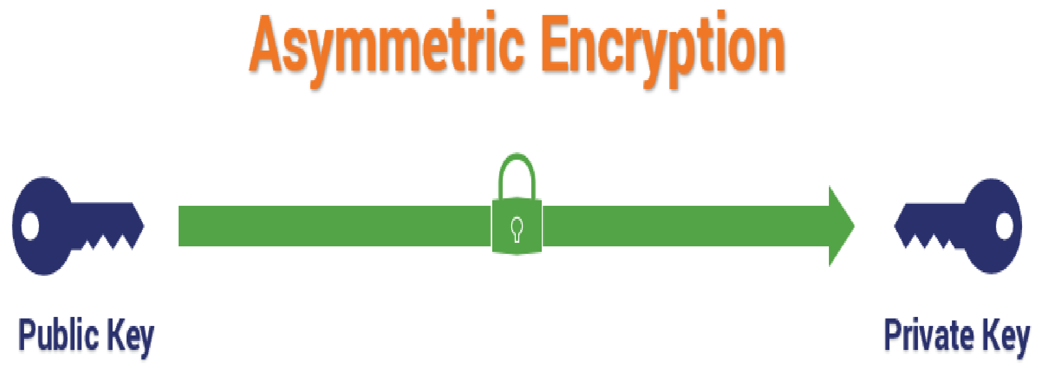
Encryption – Types

- There are two main types of encryption.
  - Symmetric

    Uses same secret key for encryption and decryption.
  - Asymmetric encryption

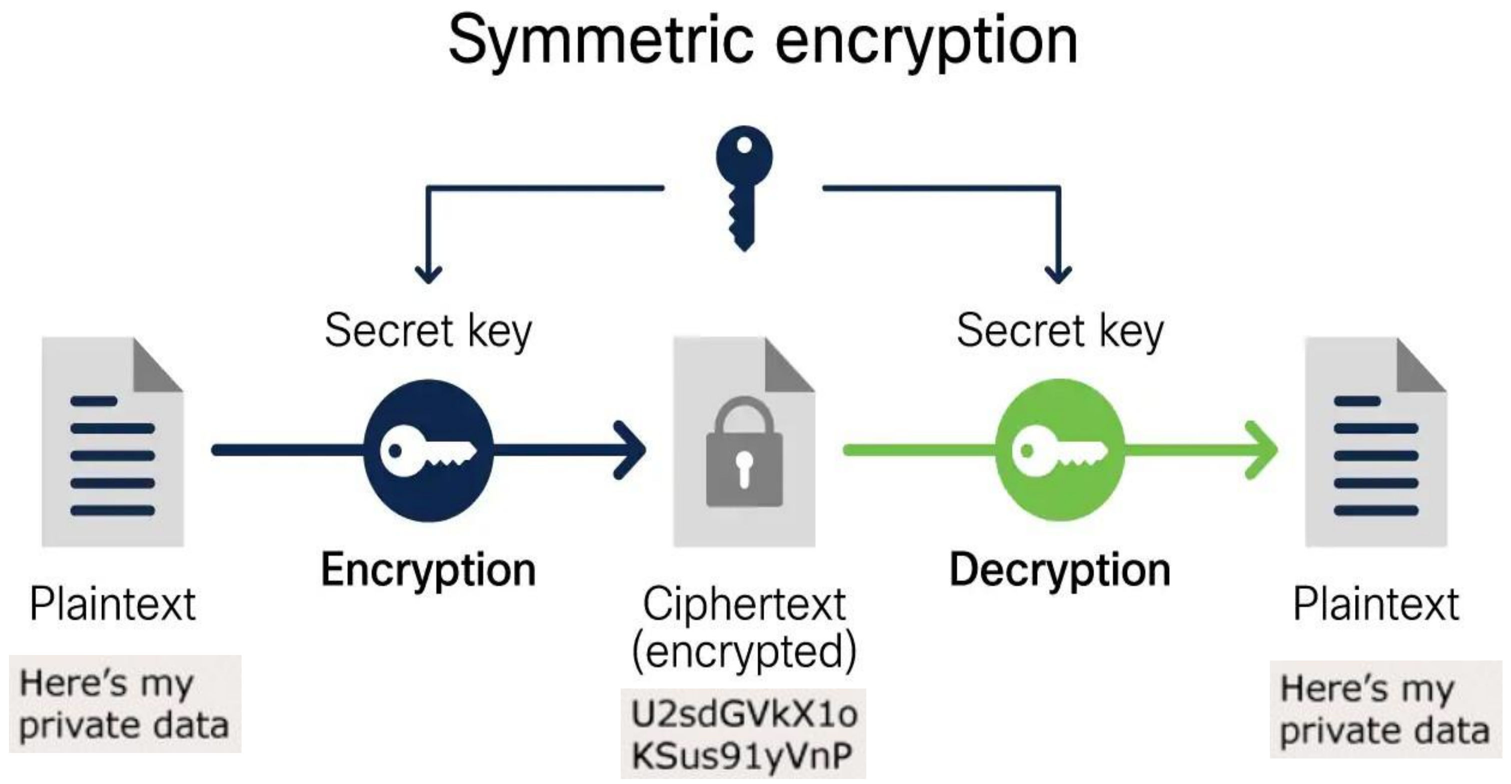
    Use pair of keys: one for encryption and second for decryption.





Symmetric Encryption

- In a symmetric encryption scheme, the same secret key is used for encryption and decryption of the information that is to be kept secret.



Symmetric Encryption

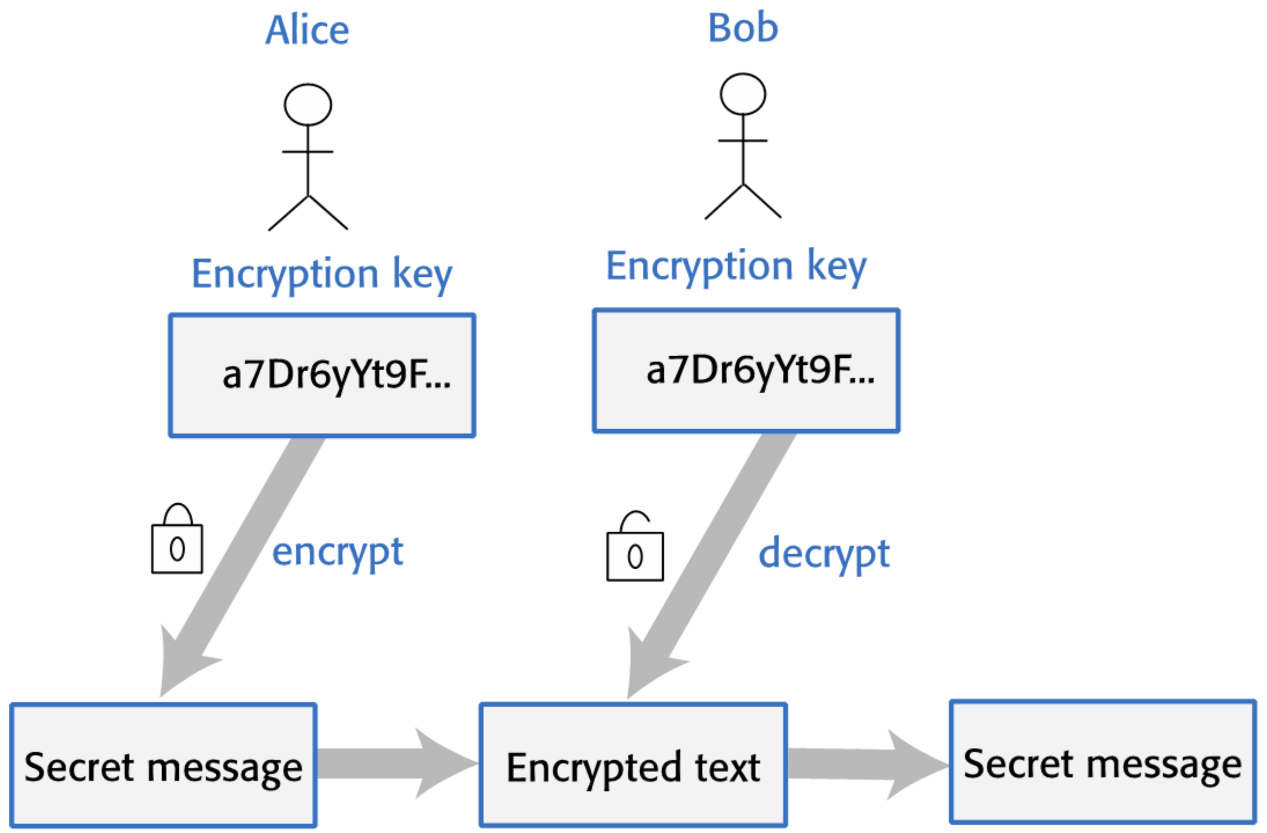
- If Alice and Bob wish to exchange a secret message, both must have a copy of the encryption key. Alice encrypts the message with this key. When Bob receives the message, he decodes it using the same key to read its contents.

- The fundamental problem with a symmetric encryption scheme is securely sharing the encryption key.

- If Alice simply sends the key to Bob, an attacker may intercept the message and gain access to the key. The attacker can then decode all future secret communications.

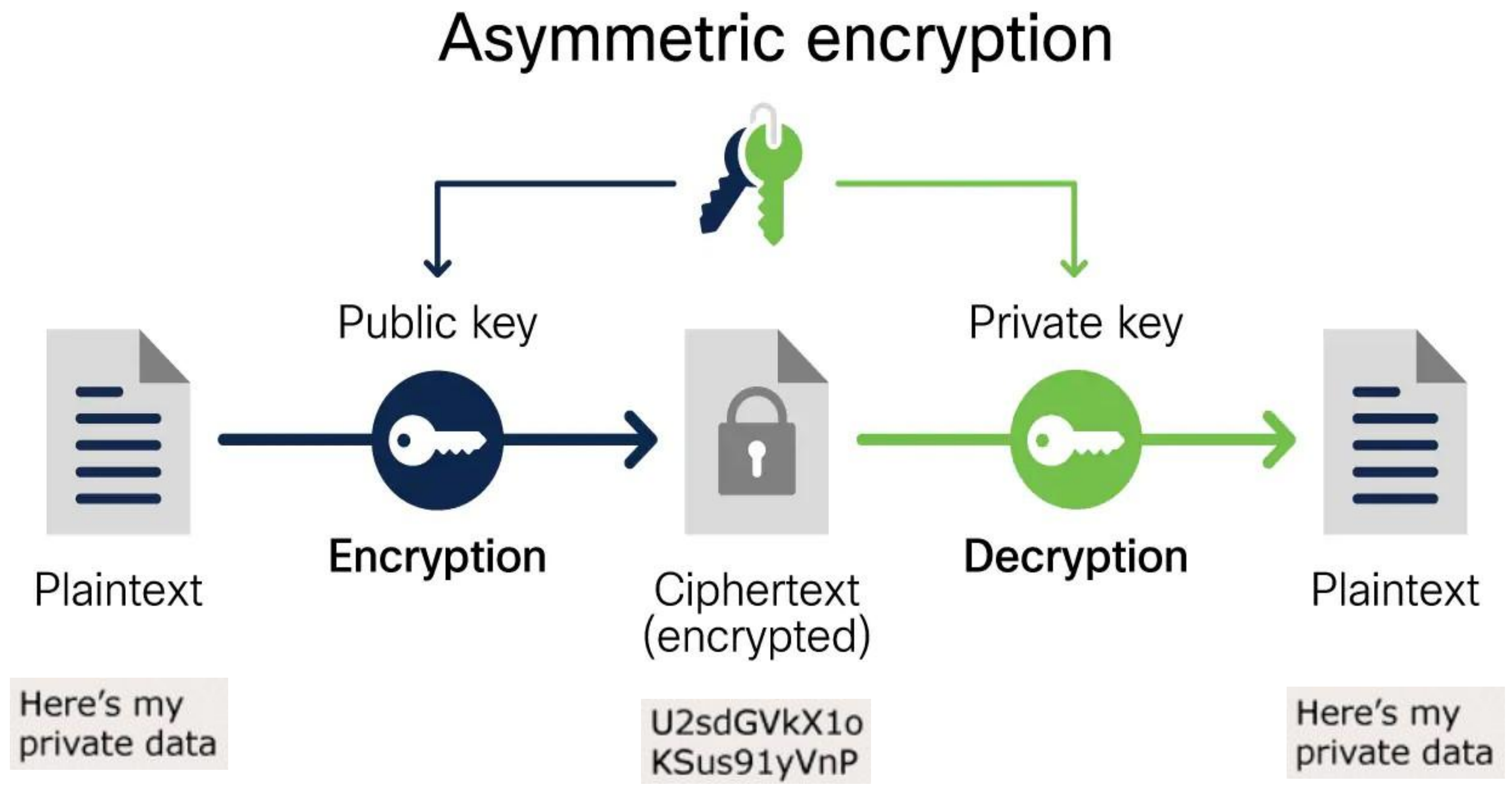
Symmetric Encryption

- Example of symmetric encryption:



Asymmetric Encryption

- An asymmetric encryption scheme uses different keys for encrypting and decrypting messages.



Asymmetric Encryption

- Asymmetric encryption, does not require secret keys to be shared.

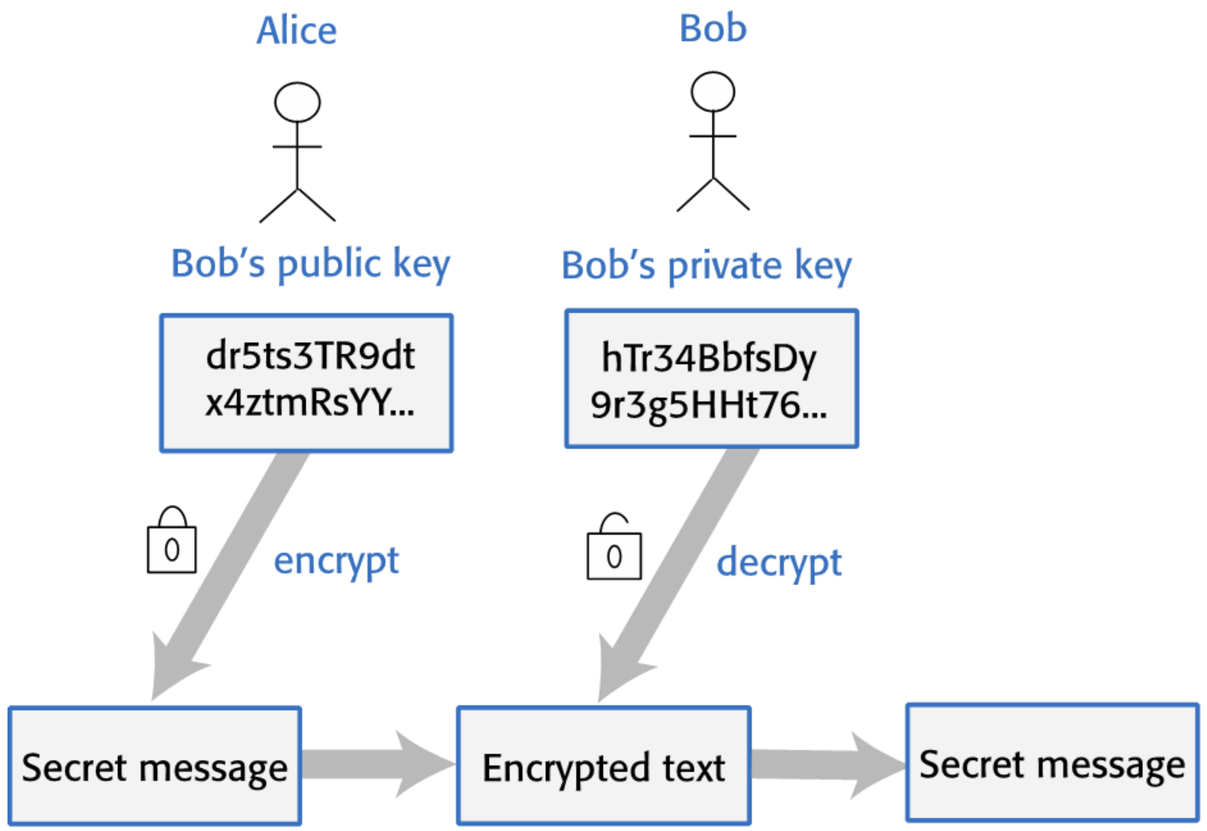
- Each user has a public and a private key. Messages may be encrypted using either key but can only be decrypted using the other key of same pair.
  - Public key is used for encryption
  - Private key is used for decryption.

- Public keys may be published and shared by the key owner. Anyone can access and use a published public key.

- However, messages can only be decrypted by the user’s private key so is only readable by the intended recipient.

Asymmetric Encryption

- Example of asymmetric encryption:



Asymmetric Encryption - Working

- Asymmetric encryption also called public-key cryptosystem.
- Public and secret keys determine functions that can be used to encrypt or decrypt a message.
- A set called D represents all allowed messages, such as all possible bit sequences.
- Public key cryptography involves one-to-one functions that map messages in D to other messages using these keys.
- Alice's public key function, $P_A()$, and secret key function, $S_A()$, transform messages based on her specific keys.

Asymmetric Encryption - Working

- Public and secret keys form a matched pair because they work as opposites (inverses) of each other.
- Encrypting a message with Alice's public key P_A and then decrypting it with her secret key S_A returns the original message M or vice versa. That is:

$$M = S_A\left(P_A(M)\right)$$
$$M = P_A(S_A(M))$$

- This relationship ensures that any message transformed by one key can be correctly recovered with the other key.
- As a result, using both keys in succession always reveals the original message.

Asymmetric Encryption - Working

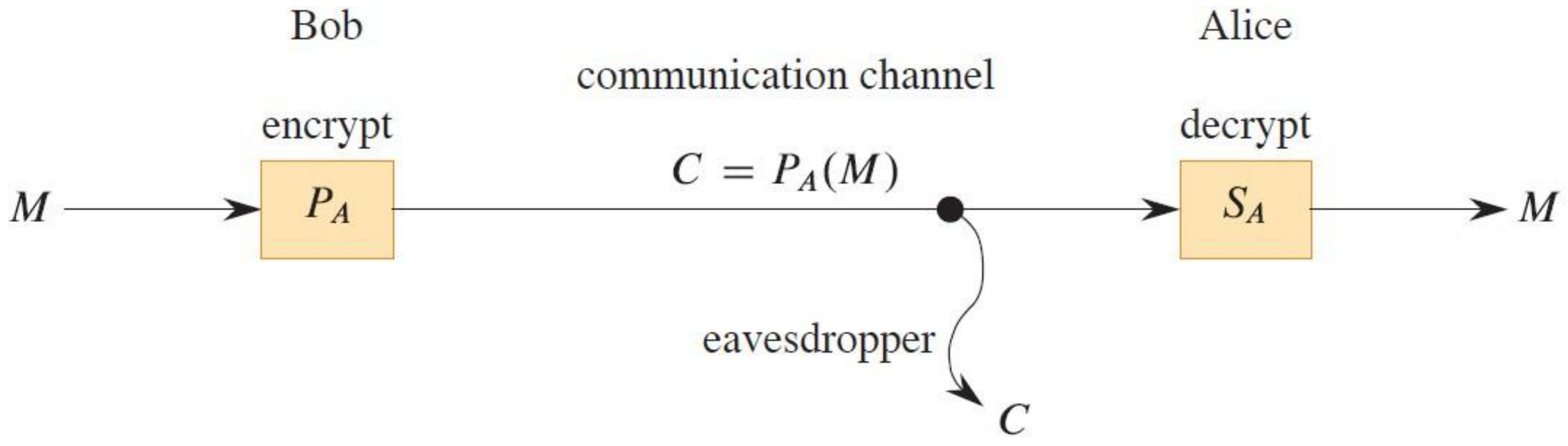
- A public-key cryptosystem requires that Alice, and only Alice, be able to compute the function $S_A()$ in any practical amount of time.

- This assumption is crucial to keeping encrypted messages sent to Alice private and to knowing that Alice’s digital signatures are authentic.

- Alice must keep her key S_A secret. If she does not, whoever else has access to S_A can decrypt messages intended only for Alice and can also forge her digital signature.

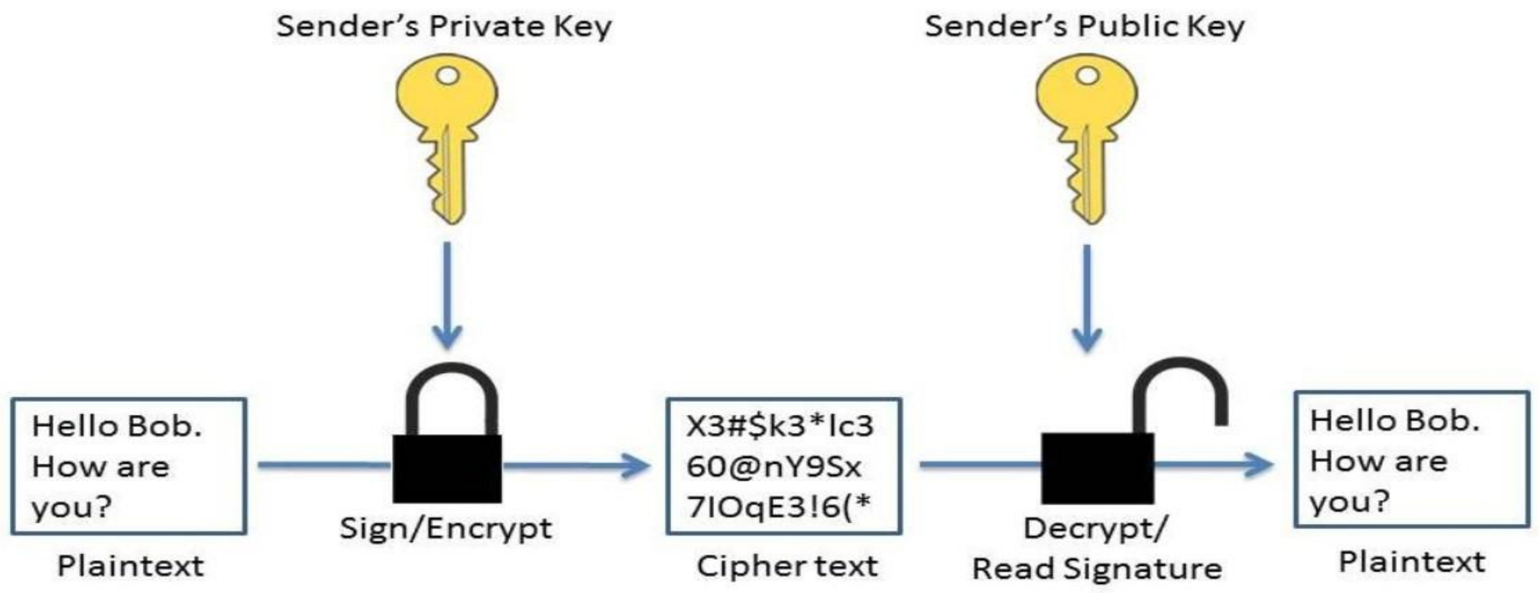
Asymmetric Encryption - Working

- Bob encrypts the message M using Alice's public key P_A and transmits the resulting ciphertext $C = P_A(M)$ over a communication channel to Alice.
- An eavesdropper who captures the transmitted ciphertext gains no information about M. Alice receives C and decrypts it using her secret key to obtain the original message $M = S_A(C)$.



Digital Signatures

- Digital signature is a cryptography algorithm used to verify the authenticity, integrity, and non-repudiation of digital messages.

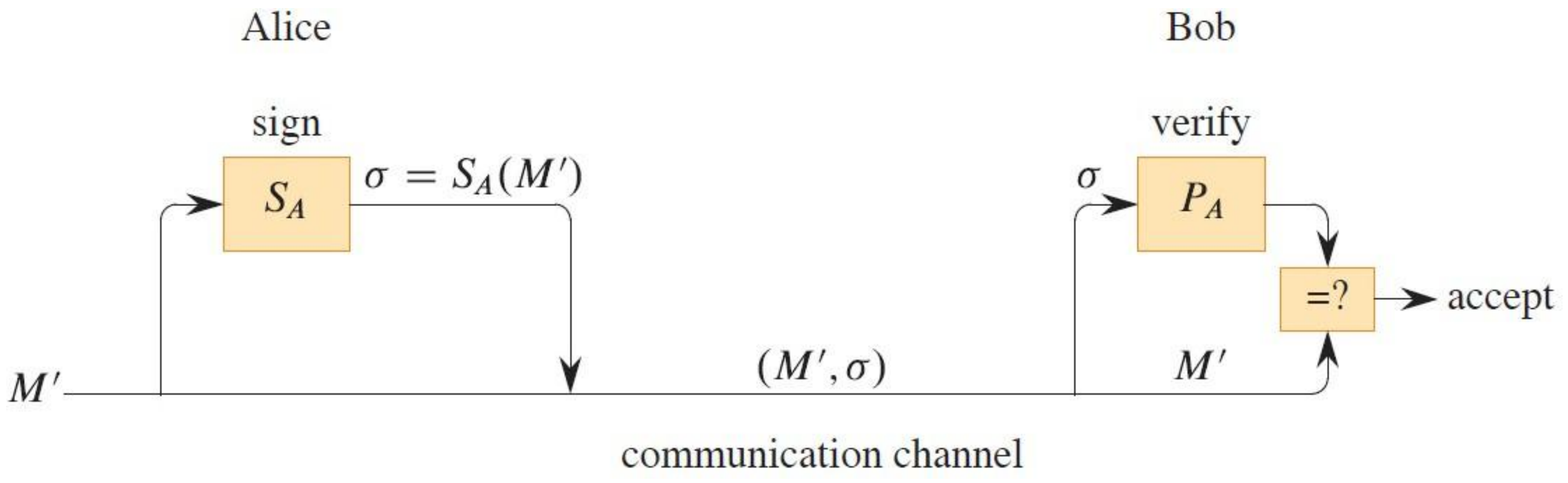


Digital Signatures

- Because a digital signature provides both authentication of the signer's identity and authentication of the contents of the signed message, it is analogous to a handwritten signature at the end of a written document.
- A digital signature must be verifiable by anyone who has access to the signer's public key. A signed message can be verified by one party and then passed on to other parties who can also verify the signature.
- For example, the message might be an electronic check from Alice to Bob. After Bob verifies Alice's signature on the check, he can give the check to his bank, who can then also verify the signature and effect the appropriate funds transfer.

Digital Signatures – Example

- Alice signs the message M by appending her digital signature $\sigma = S_A(M)$ to it. She transmits the message/signature pair (M, σ) to Bob, who verifies it by checking the equation $M = P_A(\sigma)$. If the equation holds, he accepts (M, σ) as a message that Alice has signed.



Digital Signatures – Example

- Alice computes her **digital signature** σ for the message M using her secret key S_A and the equation $\sigma = S_A(M)$
- Alice sends the message/signature pair (M', σ) to Bob.
- When Bob receives (M', σ), he can verify that it originated from Alice by using Alice's public key to verify the equation $M' = P_A(\sigma)$. (Presumably, M' contains Alice's name, so that Bob knows whose public key to use.) If the equation holds, then Bob concludes that the message M' was actually signed by Alice.
- If the equation fails to hold, Bob concludes either that the information he received was corrupted by transmission errors or that the pair (M', σ) is an attempted forgery.

Digital Signatures – Example

- A signed message may or may not be encrypted. The message can be “in the clear” and not protected from disclosure. By composing the above protocols for encryption and for signatures, Alice can create a message to Bob that is both signed and encrypted.
- Alice first appends her digital signature to the message and then encrypts the resulting message/signature pair with Bob’s public key. Bob decrypts the received message with his secret key to obtain both the original message and its digital signature.
- Bob can then verify the signature using Alice’s public key. The corresponding combined process using paper-based systems would be to sign the paper document and then seal the document inside a paper envelope that is opened only by the intended recipient.

Modular Exponentiation

Modular Exponentiation (1)

- Modular exponentiation calculates a number raised to an exponent, then finds the remainder when divided by a modulus.
- Given three numbers a, b and n, we need to find

$$a^b \; mod \; n$$

- When we exponentiate a number, we always get a bigger number, and we'll want to reduce modulo n.
- It's a computational tool we need to perform the RSA encryption algorithm.

Modular Exponentiation (2)

- The method of repeated squaring solves this problem efficiently.
- Repeated squaring is based on the following formula to compute a^b for nonnegative integers a and b:

$$a^b = \begin{cases} 1 & \text{if } b = 0 \, , \\ (a^{b/2})^2 & \text{if } b > 0 \text{ and } b \text{ is even} \, , \\ a \cdot a^{b-1} & \text{if } b > 0 \text{ and } b \text{ is odd} \, . \end{cases}$$

- The last case, where b is odd, reduces to the one of the first two cases, since if b is odd, then b–1 is even.
- The method ensures calculations remain manageable by repeatedly squaring and reducing.

Modular Exponentiation - Procedure

- The recursive procedure MODULAR-EXPONENTIATION computes a^b mod n using repeated square equation.

```
MODULAR-EXPONENTIATION($a, b, n$)
if $b == 0$
    return 1
elseif $b \bmod 2 == 0$
    $d$ = MODULAR-EXPONENTIATION($a, b/2, n$)    // $b$ is even
    return $(d \cdot d) \bmod n$
else $d$ = MODULAR-EXPONENTIATION($a, b-1, n$)   // $b$ is odd
    return $(a \cdot d) \bmod n$
```

Modular Exponentiation - Procedure

- If b is 0 (Base Case):
  - Return 1.
- If b is even:
  - Compute d = MODULAR-EXPONENTIATION(a, b/2, n).
  - Return $d.d \ mod \ n$.
- If b is odd:
  - Compute d = MODULAR-EXPONENTIATION(a, b-1, n).
  - Return $a.d \ mod \ n$.

Modular Exponentiation - Procedure

- Consider an example where we compute 3^4 mod 5 using the MODULAR-EXPONENTIATION procedure.
- **Step by Step Calculation for 3^4 mod 5 :**
  - Initial Call: MODULAR-EXPONENTIATION(3, 4, 5)

    b = 4 (even), so we need d = MODULAR-EXPONENTIATION(3,2,5).
  - Second Call: MODULAR-EXPONENTIATION(3, 2, 5)

    b = 2(even), so we need d = MODULAR-EXPONENTIATION(3,1,5).
  - Third Call: MODULAR-EXPONENTIATION(3, 1, 5)

    b = 1(odd), so we need d = MODULAR-EXPONENTIATION(3,0,5).
  - Fourth Call (Base Case): MODULAR-EXPONENTIATION(3, 0, 5)

    b = 0, return 1

Modular Exponentiation - Procedure

- **Now Reconstructing the Results:**
  - From b = 1:

    d = 1 (from b=0), Return (a . d) mod 5 = $(3 \cdot 1)$ mod 5 = 3.

  - From b = 2:

    d = 3 (from b=1), Return (d . d) mod 5 = $(3 \cdot 3)$ mod 5 = 4.

  - From b = 4:

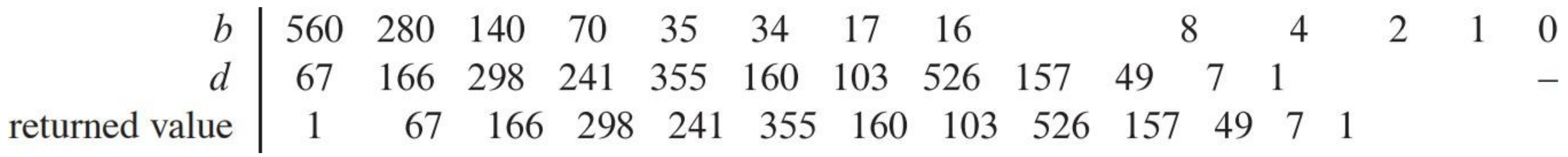
    d = 4 (from b=2), Return (d . d) mod 5= $(4 \cdot 4)$ mod 5 = 1.

- **Final Result:**

  3^4 mod 5 = 1

Modular Exponentiation - Procedure

- Another example where we compute $7^{560} \text{mod } 561$ using the MODULAR-EXPONENTIATION procedure.



b	560	280	140	70	35	34	17	16	8	4	2	1	0
d	67	166	298	241	355	160	103	526	157	49	7	1	–
returned value	1	67	166	298	241	355	160	103	526	157	49	7	1

- At each recursive level, the value of b (the exponent) is adjusted, d (the intermediate result) is squared, and the final outcome at each step is noted.
- This method simplifies and speeds up the calculation, returning 1 in the example of $7^{560} \text{mod } 561$.

The RSA Cryptosystem

The RSA Cryptosystem (1)

- In the RSA public-key cryptosystem, a participant creates a public key and a secret key with the following procedure:
  1. Select at random two large prime numbers p and q such that $p \neq q$. The primes p and q might be, say, 1024 bits each.
  2. calculate n by multiplying p and q together: $n = p.q$.
  3. Next, a small odd integer e is chosen, which has no common factors with $\phi(n) = (p-1) \times (q-1)$.
  4. Compute d as the multiplicative inverse of e modulo $\phi(n)$.

The RSA Cryptosystem (1)

- Publish the pair P=(e, n) as the participant's RSA public key.

- Keep secret the pair S=(d, n) as the participant's RSA secret key.

The RSA Cryptosystem (1)

- To transform a message M associated with a public key P=(e, n), compute

$$P(M) = M^e \; mod \; n \qquad \textit{(Equation 1)}$$

- To transform a ciphertext C associated with a secret key $S = (d, n)$, compute

$$S(C) = C^d \; mod \; n \qquad \textit{(Equation 2)}$$

- These equations apply to both encryption and signatures. To create a signature, the signer's secret key is applied to the message to be signed, rather than to a ciphertext. To verify a signature, the public key of the signer is applied to the signature rather than to a message to be encrypted.

Example

- Choose $p = 3$ and $q = 11$
- Compute $n = p * q = 3 * 11 = 33$
- Compute$\phi(n) = (p - 1) * (q - 1) = 2 * 10 = 20$
- Choose e such that $1 < e < \phi(n)$ and e and$\phi(n)$ are coprime. Let $e = 7$
- Compute a value for d such that $(d * e)\ mod\ \phi(n) = 1$. One solution is $d = 3[(3 * 7) mod\ 20 = 1]$
- Public key is $(e, n) => (7, 33)$
- Private key is $(d, n) => (3, 33)$
- The encryption of $m = 2$ is $c = 2^7 mod\ 33 = 29$
- The decryption of $c = 29$ is $m = 29^3\ mod\ 33 = 2$

Exercise

- Choose $p =??$ and $q =??$
- Compute $n = p * q =$
- Compute$\phi(n) = (p-1) * (q-1) =$
- Choose e such that $1 < e < \phi(n)$ and e and$\phi(n)$ are coprime.
- Compute a value for d such that $(d * e) \ mod \ \phi(n) = 1$.
- Public key is (e, n) => ( )
- Private key is (d, n) => ()
- The encryption of $m = 3$ is $c =$
- The decryption of $c =$ is $m =$

Exercise - Solution

- Choose $p = 5$ and $q = 7$
- Compute $n = p * q = 5 * 7 = 35$
- Compute$\phi(n) = (p - 1) * (q - 1) = 4 * 6 = 24$
- Choose e such that $1 < e < \phi(n)$ and e and$\phi(n)$ are coprime. Let $e = 5$ is coprime to 24.
- Compute a value for d such that $(d * e)\ mod\ \phi(n) = 1$. One solution is $d = 29$ $[(29 * 5) mod\ 24 = 1]$
- Public key is $(e, n) => (5, 35)$
- Private key is $(d, n) => (29, 35)$
- The encryption of $m = 3$ is $c = 3^5 mod\ 35 = 28$
- The decryption of $c = 28$ is $m = 28^{29}\ mod\ 35 = 3$

The Security of the RSA Cryptosystem (1)

- The security of the RSA cryptosystem rests in large part on the difficulty of factoring large integers.
- If an adversary can factor the modulus n in a public key, then the adversary can derive the secret key from the public key, using the knowledge of the factors p and q in the same way that the creator of the public key used them.
- Therefore, if factoring large integers is easy, then breaking the RSA cryptosystem is easy.
- The converse statement, that if factoring large integers is hard, then breaking RSA is hard, is unproven.

The Security of the RSA Cryptosystem (2)

- After two decades of research, however, no easier method has been found to break the RSA public-key cryptosystem than to factor the modulus n.
- And factoring large integers is surprisingly difficult. By randomly selecting and multiplying together two 1024-bit primes, you can create a public key that cannot be “broken” in any feasible amount of time with current technology.
- In the absence of a fundamental breakthrough in the design of number-theoretic algorithms, and when implemented with care following recommended standards, the RSA cryptosystem is capable of providing a high degree of security in applications.

The Security of the RSA Cryptosystem (2)

- To achieve security with the RSA cryptosystem, however, you should use integers that are quite long - more than 1000 bits - to resist possible advances in the art of factoring.

- In 2021, RSA moduli are commonly in the range of 2048 to 4096 bits. To create moduli of such sizes, you must find large primes efficiently.