COAC, TCB, and Cryptography

> *Created: 30 October 2025 | Last Updated: 30 October 2025*

Originator Controlled Access Control

Originator Controlled Access Control (ORCON), which addresses a very practical problem: how the creator of information can maintain control over it even after it's shared. This leads us directly into **Digital Rights Management (DRM)** systems.

中英雙語解釋 / Bilingual Explanation

創建者控制存取控制概述 / Originator Controlled Access Control Overview

核心問題： 創建文件的組織希望控制其傳播。
Core Problem: *Organization creating document wants to control its dissemination.*

- **例子：** 農業部長寫了一份備忘錄分發給她的直屬下屬，並且她必須**許可**該備忘錄才能被進一步傳播。
- *Example: Secretary of Agriculture writes a memo for distribution to her immediate subordinates, and she must give **permission** for it to be disseminated further.*
- 這就是"**創建者控制**"（這裡創建者是一個人）。

- *This is "**originator controlled**" (ORCON).*

ORCON要求 / ORCON Requirements

當主體s代表組織X將物件o標記為ORCON時：
When subject s marks object o as ORCON on behalf of organization X:

1. **傳播限制：** 未經X許可，o不能發佈給代表其他組織的主體。
   - *o cannot be released to subjects acting on behalf of other organizations without X's permission;*
2. **複製繼承：** o的任何**副本**必須具有相同的限制。
   - *Any **copies** of o must have the same restrictions placed on it.*

為什麼傳統方法失敗？ / Why Traditional Approaches Fail?

自主存取控制失敗：
Discretionary Access Control (DAC) alone fails:

- 物件的**所有者**可以設定任何所需的權限。
- *Owner of the object can set any desired permissions.*
- **注意：** 所有者**不一定**是物件的創建者/發起者。
- *Note that owner here is **NOT necessarily** the originator/creator.*
- 這使得要求(ii) **無法強制執行** - 接收者可以複製檔案並變更其權限。
- *This makes requirement (ii) **unenforceable** - recipients can copy the file and change its permissions.*

強制存取控制失敗：
Mandatory Access Control (MAC) alone fails:

- MAC缺乏**靈活性**。
- *MAC lacks **flexibility**.*
- ORCON的進一步傳播需求會引起**範疇爆炸問題** - 需要為每個新的傳播建立新的安全範疇。
- *Demands of further dissemination raise **category explosion issue** - need new security categories for each new dissemination.*
- 難以在MAC中按需靈活授權未預見的主體。
- *Hard to flexibly authorize unforeseen subjects on demand in MAC.*

解決方案：結合DAC和MAC / Solution: Combining DAC and MAC

混合方法：
Hybrid Approach:

元件 / Component	機制 / Mechanism	目的 / Purpose
MAC部分	物件的**所有者不能**變更物件的存取控制。	確保**穩定性**
	*The **owner** of an object **cannot change** the access controls.*	*Ensures **stability***
	複製物件時，源的存取控制限制應**複製並綁定**到目標。	強制執行**複製繼承**
	*When object is copied, access control restrictions are **copied and bound** to the target.*	*Enforces **copy inheritance***
DAC部分	**創建者**可以按每個主體和每個物件 basis 變更存取控制限制。	提供**靈活性**
	*The **creator (originator)** can alter access control restrictions.*	*Provides **flexibility***

在數位版權管理中的應用 / Application in Digital Rights Management (DRM)

ORCON的要求在DRM系統中得到了具體體現：
ORCON requirements have concrete implications in DRM:

DRM核心元素：
DRM Core Elements:

- **內容：** 受保護的資訊
- ***Content:** information being protected*
- **授權：** 描述內容允許使用的令牌
- ***License:** token describing allowed uses*
- **授予：** 授權的一部分，向一個或多個實體授予特定授權
- ***Grant:** part of license giving specific authorizations*
- **發行者：** 發行授權的實體（通常是創建者）
- ***Issuer:** entity issuing the license*

- **主體：** 實體的標識，用於標識授權適用於誰
- ***Principal:*** *identification of entity to whom license applies*
- **裝置：** 用於檢視內容的機制
- ***Device:*** *mechanism used to view content*

DRM系統範例 / DRM System Examples

Microsoft PlayReady:

- **設定：** 內容使用AES加密，金鑰提供給授權伺服器
- ***Setup:*** *Content enciphered using AES, key available to license server*
- **播放：** 用戶端下載內容，請求授權，伺服器驗證用戶端後發送授權
- ***Play:*** *Client downloads content, requests license, server authenticates and sends license*
- **特性：** 支援時間限制、複製約束、地理限制等
- ***Features:*** *Time limits, copy constraints, geographical restrictions*

Apple FairPlay:

- **授權系統：** 最多5個系統可以同時授權播放使用者的內容
- ***Authorization:*** *At most 5 systems at a time can be authorized*
- **金鑰管理：** 使用使用者金鑰加密主金鑰，確保內容保護
- ***Key Management:*** *User key encrypts master key to protect content*
- **離線播放：** 播放時不需要聯繫Apple伺服器
- ***Offline Play:*** *No need to contact Apple servers for playback*

DRM關鍵要求 / Key DRM Requirements

DRM系統必須滿足：
The system must satisfy all of:

1. **使用控制：** 必須實現對內容使用的控制，約束使用者可以對內容做什麼。
   - *Implement **controls on use** of content, constraining what users can do.*
2. **關聯規則：** 約束內容使用的規則必須與**內容關聯**，而不是與使用者關聯。
   - *Rules must be **associated with the content**, not the users.*
3. **持續控制：** 控制和規則必須在內容的整個生命週期內**持續存在**，無論內容如何分發或分發給誰。

- *Controls must **persist throughout the life** of the content, regardless of distribution.*

重要說明： 僅僅加密內容並向授權檢視者提供金鑰是**不夠的**，因為使用者可以隨意分發金鑰。
***Important:** Simply encrypting content and providing keys fails, as users can distribute keys indiscriminately.*

小測驗 / Test

1. 為什麼單純的自主存取控制無法實現ORCON的要求(ii) - "任何副本必須具有相同的限制"？
   - *Why does Discretionary Access Control (DAC) alone fail to enforce ORCON requirement (ii) - "Any copies must have the same restrictions"?*
2. 在ORCON的混合方法中，MAC元件和DAC元件分別負責什麼？為什麼需要兩者結合？
   - *In the ORCON hybrid approach, what do the MAC and DAC components respectively handle? Why is the combination necessary?*
3. DRM系統如何確保即使使用者將加密內容檔案複製給朋友，朋友也無法播放它？
   - *How does a DRM system ensure that even if a user copies the encrypted content file to a friend, the friend cannot play it?*
4. Microsoft PlayReady和Apple FairPlay在授權方法上有什麼主要區別？
   - *What is a key difference in the authorization approach between Microsoft PlayReady and Apple FairPlay?*

答案與解析 / Answers and Explanations

1. **DAC無法強制執行複製繼承**
   - **答案：** 因為在DAC中，檔案的**接收者成為新副本的所有者**。作為所有者，他們有權隨意變更檔案的權限設定，包括完全移除所有存取限制。創建者無法阻止接收者建立不受限制的副本。
   - *Answer: Because in DAC, the **recipient of the file becomes the owner of the new copy**. As the owner, they have the right to change the permission settings arbitrarily, including removing all access restrictions entirely. The originator has no way to prevent recipients from creating unrestricted copies.*
2. **MAC與DAC在ORCON中的分工**
   - **答案：**
     - **MAC元件**提供**穩定性**和**持續性**：確保存取控制規則不能被普通使用者（包括所有

者）更改，並且這些規則在複製時被繼承。
        - **DAC元件**提供**靈活性**：允許創建者根據需要靈活地新增或移除特定使用者和組織的存取權限。
    - **需要結合的原因：** 純MAC太僵化，無法處理ORCON所需的動態授權；純DAC太脆弱，無法保證控制的持續性。混合方法各取所長。
    - *Answer:*
        - ***MAC component** provides **stability and persistence**: ensures access control rules cannot be changed by ordinary users (including owners), and that rules are inherited during copying.*
        - ***DAC component** provides **flexibility**: allows the originator to flexibly add or remove access for specific users and organizations as needed.*
    - *Why combine: Pure MAC is too rigid for ORCON's dynamic authorization needs; pure DAC is too weak to guarantee persistent control. The hybrid takes the best of both.*

3. **DRM的複製保護機制**
    - **答案：** DRM系統使用**加密和授權綁定**。內容檔案本身是加密的。要解密和播放內容，使用者不僅需要加密檔案，還需要一個有效的**授權**。該授權通常與特定的**使用者帳戶**、**裝置或硬體**綁定。即使使用者複製了加密檔案，他們的朋友也缺乏與該檔案配對的、綁定到其自己帳戶或裝置的有效授權，因此無法解密和播放內容。
    - *Answer: DRM systems use **encryption and license binding**. The content file itself is encrypted. To decrypt and play the content, a user needs not only the encrypted file but also a valid **license**. This license is typically bound to a specific **user account, device, or hardware**. Even if a user copies the encrypted file, their friend lacks the valid license, bound to their own account/device, that pairs with that file, and thus cannot decrypt and play it.*

4. **PlayReady vs FairPlay授權區別**
    - **答案：** 一個關鍵區別在於**授權檢查的時機和頻率**。
        - **PlayReady**更傾向於**伺服器端授權和策略執行**。用戶端在播放特定內容前通常需要從授權伺服器取得或驗證授權，伺服器可以實施複雜的策略（如地理限制、播放次數）。
        - **FairPlay**使用**裝置授權和離線播放**。系統（如電腦）首先獲得播放使用者內容的通用授權（最多5台裝置）。一旦授權，播放特定內容時通常不需要再次聯繫伺服器，因為解密金鑰本地可用。
    - *Answer: A key difference lies in the **timing and frequency of authorization checks**.*
        - ***PlayReady** leans towards **server-side authorization and policy enforcement**.*

The client often needs to acquire or validate a license from a license server before playing specific content, allowing the server to enforce complex policies (e.g., geo-restrictions, play counts).

- ***FairPlay** uses **device authorization and offline playback**. A system (e.g., a computer) first gets general authorization to play the user's content (up to 5 devices). Once authorized, playing specific content usually doesn't require contacting the server again, as decryption keys are available locally.*

Trusted Computing Base

Trusted Computing Base (TCB) is about understanding what parts of a system we actually need to trust for security to work.

中英雙語解釋 / Bilingual Explanation

可信計算基概述 / Trusted Computing Base Overview

TCB定義： 系統中**強制執行安全策略**的部分（硬體、韌體、軟體）。
***TCB Definition:** The parts of a system (hardware, firmware, software) that **enforce a security policy**.*

關鍵設計原則： 良好的安全設計應嘗試使TCB**盡可能小**。
***Key Design Principle:** A good security design should attempt to make the TCB **as small as possible**.*

為什麼TCB要小？

- **最小化錯誤機會：** 減少其實作中出現錯誤的機會。
- ***Minimize errors:** Reduce the chance for errors in its implementation.*
- **簡化驗證：** 簡化仔細的驗證過程。
- ***Simplify verification:** Simplify careful verification.*
- **限制攻擊面：** TCB**之外**的故障不會幫助攻擊者違反安全策略。
- ***Limit attack surface:** Faults **outside the TCB** will not help an attacker violate the security policy.*

TCB示例：Unix工作站 / TCB Example: Unix Workstation

在Unix工作站中，TCB至少包括：

In a Unix workstation, the TCB includes at least:

1. **作業系統核心**及其所有裝置驅動程式
   - *The **operating system kernel** including all its device drivers*
2. 所有以**root權限**執行的行程
   - *All processes that run with **root privileges***
3. 所有由root擁有且設定了**set-user-ID位**的程式檔案
   - *All program files owned by root with the **set-user-ID-bit** set*
4. 用於建構上述內容的所有**函式庫和開發工具**
   - *All **libraries and development tools** used to build the above*
5. **CPU**
   - *The **CPU***
6. **大容量儲存裝置**及其韌體
   - *The **mass storage devices** and their firmware*
7. **檔案伺服器**及其網路鏈路的完整性
   - *The **file servers** and the integrity of their network links*

重要性： 這些元件中的任何一個安全漏洞都可能被用來**繞過整個Unix存取控制機制**。
*Any security vulnerability in these could be used to **bypass the entire Unix access control mechanism**.*

參考監視器 / Reference Monitor

參考監視器是一個軟體模型或抽象機器，它**仲裁**任何主體對任何物件的所有存取，並且**不能被繞過**。
*A **reference monitor** is a software model that **mediates all access** from any subject to any object and **cannot be bypassed**.*

安全核心是特定硬體基礎的參考監視器的實作。
*A **security kernel** is an implementation of a reference monitor for a specific hardware base.*

現代實作： 類似功能也由在更高或更低層級操作的環境提供：
Today, similar functions are provided by:

- **更高層級：** Java/C#虛擬機器（語言約束強制執行域分離）
- *Higher-level: Java/C# virtual machine (language constraints enforce domain separation)*
- **更低層級：** 虛擬機器監視器，如Xen或VMware
- *Lower-level: Virtual machine monitors like Xen or VMware*

參考監視器的三個屬性 / Reference Monitor's Three Properties

1. **完整性：** 它仲裁**每一次**存取 → **不能被繞過**
   - ***Complete:** Mediates every access → **cannot be bypassed***
2. **隔離性：** 與其他系統實體隔離 → **不能被修改**
   - ***Isolated:** Isolated from modification → **cannot be altered***
3. **可驗證性：** 只做它被程式設計要做的事 → **可以被測試為正確**
   - ***Verifiable:** Only does what it's programmed to do → **can be tested to be correct***

獨立系統中的信任環 / Rings of Trust in Stand-alone Systems

要求更高層級安全性的系統位於**內環**中。
*Systems requiring higher levels of security are located inside the **inner ring**.*

保護環模型：
Protection Ring Model:

環級別	**內容**	**信任級別**
Ring 0	**作業系統核心**	**最高信任**
	OS kernel	***Most trusted***
Ring 1	作業系統的非特權部分	高信任
	Non-privileged OS portions	*High trust*
Ring 2	I/O驅動程式、低階操作和公用程式	中等信任
	I/O drivers, low-level operations	*Medium trust*
Ring 3	**應用程式和行程**	**最低信任**
	Applications and processes	***Least trusted***

關鍵特性： 系統內的信任**從外向內單向移動**。
*Trust in a system moves from the outside to the inside in a **unidirectional mode**.*

網路環境中的信任環 / Rings of Trust in Networked Environments

網路主機被劃分為環，作為主機間信任的基礎。
The hosts of networks are divided into rings used as a basis for trust between hosts.

確定環層次的分析問題：
Questions to determine ring hierarchy:

- 主機在實體安全的電腦機房中嗎？
- *Is the host in a **physically secured** computer room?*
- 主機有普通（相對於特權）使用者帳戶嗎？
- *Does the host have **normal user accounts**?*
- 這個主機在遠端站點，因此比中央電腦機房的主機更不可信嗎？
- *Is this host at a **remote site** and less trustworthy?*
- 這個主機執行依賴從網際網路取得資料的軟體嗎？
- *Does this host operate software that relies on **data from the Internet**?*
- 這個主機提供關鍵任務服務嗎？公司中會有多少人受其停機影響？
- *Does this host provide **mission-critical services**?*

網路信任環的一般規則 / General Rules for Network Trust Rings

1. **向內信任：** 每個主機信任那些在**比它更內環**的主機。
   - *Each host trusts those hosts in a **more inner ring** than itself.*
2. **不向外信任：** 沒有主機信任任何在**比它更外環**的主機。
   - *No host trusts any host in a **more outer ring** than itself.*
3. **同級信任：** 每個主機可能信任那些在**同一環**中的主機。
   - *Each host may trust those hosts in the **same ring** as itself.*
4. **分段隔離：** 如果一個環被分割成單獨的子網，一個段中的主機不信任其他段中的主機。
   - *Where a ring is segmented, a host in one segment **does not trust** hosts in other segments.*

這與Biba嚴格完整性策略高度相關！
Think about Biba's Strict Integrity Policy - highly relevant here!

小測驗 / Test

1. 為什麼安全工程師要努力保持TCB盡可能小？請給出兩個主要原因。
   - *Why do security engineers strive to keep the TCB as small as possible? Give two main*

reasons.

2. 在保護環模型中，為什麼應用程式通常在Ring 3執行，而作業系統核心在Ring 0執行？
    - *In the protection ring model, why do applications typically run in Ring 3 while the OS kernel runs in Ring 0?*
3. 如果一個組織的內部資料庫伺服器被放置在網路信任環模型的Ring 1，而員工的桌上型電腦在Ring 2，根據信任環規則，資料庫伺服器應該信任桌上型電腦的存取請求嗎？為什麼？
    - *If an organization's internal database server is placed in Ring 1 of the network trust ring model, and employee desktop computers are in Ring 2, should the database server trust access requests from the desktops according to the trust ring rules? Why?*
4. 參考監視器的"完整性"屬性具體是什麼意思？如果這個屬性被違反會怎樣？
    - *What does the "completeness" property of a reference monitor specifically mean? What happens if this property is violated?*

答案與解析 / Answers and Explanations

1. **保持TCB小的原因**
    - **答案：**
        - **減少攻擊面：** TCB越小，包含可能被利用的安全漏洞的程式碼就越少。TCB外部的漏洞不會直接導致安全策略被違反。
        - **便於驗證和稽核：** 較小的程式碼庫更容易進行徹底的安全分析、形式驗證和測試，以確保其正確實作安全策略。
    - *Answer:*
        - ***Reduce attack surface:*** *The smaller the TCB, the less code that could contain security vulnerabilities to be exploited. Vulnerabilities outside the TCB cannot directly lead to security policy violations.*
        - ***Easier verification and auditing:*** *A smaller codebase is easier to thoroughly analyze, formally verify, and test to ensure it correctly implements the security policy.*
2. **應用程式與核心的環級別**
    - **答案：** 因為應用程式**更複雜**、**更不可預測且來自各種來源**，因此更可能包含漏洞或惡意程式碼。將它們限制在最低特權環（Ring 3）可以**限制損害** - 如果應用程式被破壞，

它不能直接修改關鍵的作業系統資料結構或干擾其他應用程式。作業系統核心需要最高特權（Ring 0）來管理所有系統資源，但正因如此，必須極其小心地設計和實作。

- *Answer: Because applications are **more complex, unpredictable, and from various sources**, making them more likely to contain bugs or malicious code. Confining them to the least privileged ring (Ring 3) **limits the damage** - if an application is compromised, it cannot directly modify critical OS data structures or interfere with other applications. The OS kernel needs highest privilege (Ring 0) to manage all system resources, but must be designed and implemented with extreme care because of this.*

3. **網路信任環的存取規則**

   - **答案： 不應該信任。**
   - **解析：** 根據信任環規則："沒有主機信任任何在比它更外環的主機。"資料庫伺服器在Ring 1（更內環），桌上型電腦在Ring 2（更外環）。因此，資料庫伺服器**不應該自動信任**來自桌上型電腦的存取請求。存取應該基於嚴格的身份驗證和授權來控制，預設不信任外環主機。這體現了**Biba完整性模型**的原則，防止低完整性（外環）實體污染高完整性（內環）資料。
   - *Answer: No, it should not.*
   - *Explanation: According to the trust ring rules: "No host trusts any host in a more outer ring than itself." The database server is in Ring 1 (more inner), and the desktops are in Ring 2 (more outer). Therefore, the database server **should not automatically trust** requests from the desktops. Access should be controlled based on strict authentication and authorization, distrusting outer ring hosts by default. This embodies the principle of the **Biba integrity model**, preventing low-integrity (outer ring) entities from contaminating high-integrity (inner ring) data.*

4. **參考監視器的完整性屬性**

   - **答案：** "完整性"屬性意味著參考監視器**仲裁主體對物件的每一次存取嘗試**。換句話說，在系統中**不存在**可以繞過參考監視器檢查而直接存取物件的路徑。
   - **如果違反的後果：** 如果存在繞過參考監視器的路徑，攻擊者可以利用這個"後門"直接存取受保護的資源，而無需經過安全檢查。這將使整個安全策略**無效**，因為策略執行可以被輕易繞過。
   - *Answer: The "completeness" property means the reference monitor **mediates every single access attempt** from a subject to an object. In other words, there is **no path** in the system to access an object directly, bypassing the reference monitor's check.*
   - *Consequence of violation: If a path exists to bypass the reference monitor, an*

*attacker can use this "backdoor" to access protected resources directly without security checks. This would **nullify** the entire security policy, as policy enforcement can be easily circumvented.*

Cryptography

中英雙語解釋 / Bilingual Explanation

密碼學概述 / Cryptography Overview

密碼學是一個豐富而複雜的學科。
Cryptography is a rich, complex subject.

我們的目標： 培養關於以下方面的直覺：
Our goal is to develop intuitions about:

- 密碼學的**關鍵概念**是什麼
- *What are the **key concepts** of cryptography;*
- 它如何作為安全**工具**被使用
- *How is it used as a **tool** for security;*
- 在這方面它的**有效性**如何
- *How **effective** is it in that regard.*

密碼分析範例：《金甲蟲》 / Cryptanalysis Example: "The Gold Bug"

愛倫·坡的《金甲蟲》是密碼分析的一個經典例子：
Poe's "The Gold Bug" is a classic example of cryptanalysis:

- 在19世紀初，一個人在南卡羅來納州海灘上發現一張羊皮紙碎片。
- *In the early 1800's, a man finds a scrap of parchment on a South Carolina beach.*
- 上面有一條奇怪的編碼資訊和一幅山羊頭圖畫。
- *On the parchment is a strange encoded message and a drawing of a goat's head.*
- 他懷疑這條資訊是否是通往臭名昭著的海盜基德船長埋藏寶藏位置的路線。
- *He wonders if the message could be directions to the location of a treasure buried by the infamous pirate Captain Kidd.*

編碼資訊：

```
53++!305))6*;4826)4+)4+).;806*;48!8]60))85;1+8*:+(;:+*8!83(88)5*!;46(;88*96*?;8)
*+(;485);5*!2:*+(;4956*2(5*-4)8]8*;4069285);)6!8)4++;1(+9;48081;8:8+1;48!85;4)
485!528806*81(+9;48;(88;4(+?34;48)4+;161;:188;+?;
```

密碼分析問題： 要開始破解，你應該問什麼問題？
To get started, what questions should you ask?

- **基礎語言：** 明文的可能基礎語言是什麼？
- ***Underlying language:*** *What is the likely underlying language of the plaintext?*
- **源文字特徵：** 可能源文字的哪些特徵是相關的？
- ***Source text characteristics:*** *What characteristics of the probable source text are relevant?*
- **語言特徵：** 源語言的哪些特徵是相關的？
- ***Language characteristics:*** *What characteristics of the source language are relevant?*
- **演算法複雜度：** 加密演算法的可能性質/複雜度是什麼？
- ***Algorithm complexity:*** *What is the likely nature/complexity of the encryption algorithm?*
- **預處理：** 在加密之前是否應用了任何變換/壓縮？
- ***Pre-processing:*** *Have any transformations/compressions been applied prior to encryption?*

加密/解密基礎 / Encryption/Decryption Basics

加密的目的： 使訊息對任何竊聽者**不那麼有用/有意義**。
*The purpose of encryption is to render the message **less useful/meaningful** to any eavesdropper.*

概念上， 加密過程非常簡單：
Conceptually, *the process of encryption is quite simple:*

- **明文 → 加密 → 密文**
- ***Plaintext → Encryption → Ciphertext***

解密過程同樣簡單：
As is the process of decryption:

- **密文 → 解密 → 明文**
- ***Ciphertext → Decryption → Plaintext***

資訊理論與密碼學 / Information Theory and Cryptography

資訊理論從幾個方面為密碼學提供資訊：
Information theory informs cryptography in several ways:

- **資訊內容：** 加密訊息對檔案的資訊內容有什麼影響？
- ***Information content:*** *What effect does encrypting a message have on the information content?*
- **雜訊通道：** 解密訊息的嘗試實際上是從（系統性的）雜訊通道中恢復訊息的嘗試。
- ***Noisy channel:*** *Decryption is an attempt to recover a message from a (systematically) noisy channel.*
- **冗餘性：** 源中的冗餘能否為解碼過程提供線索？
- ***Redundancy:*** *Can redundancy in the source give clues to the decoding process?*
- **完美加密：** 完美的加密是否可能（即理論上不可破解的）？
- ***Perfect encryption:*** *Is a perfect encryption possible?*

關鍵洞察：冗餘是安全加密的敵人，因為它為攻擊者提供了槓桿。
Key Insight: Redundancy is the enemy of secure encryption *because it provides leverage to the attacker.*

密碼學術語 / Cryptography Terminology

加密和解密是將一個文字轉換為另一個文字的函式：
Encryption and decryption are functions which transform one text into another:

- `C = E(M)` 和 `M = D(C)`
- 其中：
  - **C** 表示密文
  - ***C*** *denotes ciphertext*
  - **E** 是加密規則
  - ***E*** *is the encryption rule*
  - **D** 是解密規則
  - ***D*** *is the decryption rule*
  - **M** 是明文
  - ***M*** *is the plaintext*

顯然，能夠從密文中恢復原始訊息很重要：
It is obviously important to be able to recover the original message:

- $M = D(E(M))$

金鑰演算法 / Keyed Algorithms

通常加密和解密演算法使用一個**金鑰K**：
*Often the encryption and decryption algorithms use a **key K**.*

- **金鑰**從E定義的演算法族中選擇一個特定的演算法。
- *The **key** selects a specific algorithm from the family of algorithms defined by E.*
- 我們將其依賴關係寫為：
- *We write this dependence as:*
  - $C = E(M, K_E)$ 和 $M = D(C, K_D)$

演算法類型：
Algorithm Types:

- **對稱演算法：** 如果 $K_E = K_D$
  - ***Symmetric:*** *If* $K_E = K_D$
- **非對稱演算法：** 如果 $K_E \neq K_D$
  - ***Asymmetric:*** *If* $K_E \neq K_D$

一般來說： $M = D(E(M, K_E), K_D)$
In general: $M = D(E(M, K_E), K_D)$

替代表示法 / Alternative Notations

在密碼協定中經常使用替代表示法：
Alternative notations are often used, particularly in cryptographic protocols:

- M_K 表示 $E(M, K)$ ，有時也表示 $D(M, K)$
- *We often use* M_K *to denote* $E(M, K)$ *, and sometimes to denote* $D(M, K)$ *.*

例子：

- $M = D(E(M, K_E), K_D) = M_{KEKD}$

- $M_{KE} = Enc_{KE}(M) = C$
- $C_{KD} = Dec_{KD}(C) = M$

所有表示法都是合適的。
All notations are appropriate.

密碼分析任務 / Cryptanalysis Tasks

密碼分析員可能嘗試以下任何或所有任務：
A cryptanalyst may attempt to do any or all of the following:

1. **破解單個訊息**
   - ***Break a single message***
2. **識別加密訊息中的模式**
   - ***Recognize patterns in encrypted messages***
3. **在不破解演算法的情況下推斷某些含義**
   - ***Infer some meaning without breaking the algorithm***
4. **推導出金鑰**
   - ***Deduce the key***
5. **發現實作、環境或加密使用中的弱點**
   - ***Find weaknesses in the implementation or environment or use***
6. **發現演算法中的弱點，而不必截獲任何訊息**
   - ***Find weaknesses in the algorithm, without intercepted messages***

密碼分析工具 / Cryptanalysis Tools

分析員使用以下工具工作：
The analyst works with:

- **加密訊息**
  - ***Encrypted messages***
- **已知加密演算法**
  - ***Known encryption algorithms***
- **截獲的明文**
  - ***Intercepted plaintext***
- **已知或懷疑在密文訊息中的資料項**

    - ***Data items known or suspected to be in ciphertext***
- **數學和統計工具和技術**
    - ***Mathematical and statistical tools***
- **語言屬性**
    - ***Properties of languages***
- **電腦**
    - ***Computers***
- **創造力和運氣**
    - ***Ingenuity and luck***

小測驗 / Test

1. 為什麼說"冗餘是安全加密的敵人"？請用英語例子說明。
    - *Why is "redundancy the enemy of secure encryption"? Illustrate with an English example.*
2. 對稱演算法和非對稱演算法的關鍵區別是什麼？
    - *What is the key difference between symmetric and asymmetric algorithms?*
3. 如果一個密碼分析員發現了加密演算法本身的弱點，但沒有截獲任何訊息，這仍然被認為是成功的密碼分析嗎？為什麼？
    - *If a cryptanalyst finds weaknesses in the encryption algorithm itself without intercepting any messages, is this still considered successful cryptanalysis? Why?*
4. 在《金甲蟲》的例子中，為什麼了解可能的源語言（英語）對破解密碼有幫助？
    - *In the "Gold Bug" example, why would knowing the probable source language (English) help in breaking the cipher?*

答案與解析 / Answers and Explanations

1. **冗餘的問題**
    - **答案：** 冗餘為密碼分析員提供了**統計槓桿**。自然語言有高度可預測的模式。
    - **英語例子：**
        - 字母'e'是英語中最常見的字母。
        - 單字'the'是英語中最常見的三字母單字。

        - 某些字母組合（如'qu'）很常見，而其他（如'zx'）很罕見。
    - 如果加密演算法不能消除這些模式，分析員可以透過分析密文中符號的頻率和分佈來推斷明文。完美的加密應該使密文在統計上與隨機雜訊無法區分。
    - *Answer: Redundancy provides **statistical leverage** to the cryptanalyst. Natural languages have highly predictable patterns.*
    - *English Example:*
        - *The letter 'e' is the most common letter in English.*
        - *The word 'the' is the most common three-letter word.*
        - *Certain letter pairs (like 'qu') are common, while others (like 'zx') are rare.*
    - *If the encryption doesn't eliminate these patterns, an analyst can infer the plaintext by analyzing the frequency and distribution of symbols in the ciphertext. Perfect encryption should make ciphertext statistically indistinguishable from random noise.*

2. **對稱 vs 非對稱演算法**

    - **答案：** 關鍵區別在於**加密和解密使用的金鑰**。
        - **對稱演算法：** 使用**相同的金鑰**進行加密和解密（$K_E = K_D$）。這就像用一個鑰匙鎖箱子，用同一個鑰匙開箱子。
        - **非對稱演算法：** 使用**不同的金鑰**進行加密和解密（$K_E \neq K_D$）。這就像用一個鑰匙（公鑰）鎖箱子，但需要用另一個不同的鑰匙（私鑰）開箱子。
    - *Answer: The key difference lies in the **keys used for encryption and decryption**.*
        - ***Symmetric algorithms** use the **same key** for both encryption and decryption ($K_E = K_D$). It's like using one key to lock a box and the same key to unlock it.*
        - ***Asymmetric algorithms** use **different keys** for encryption and decryption ($K_E \neq K_D$). It's like using one key (public key) to lock a box, but needing a different key (private key) to unlock it.*

3. **發現演算法弱點**

    - **答案： 是的，這絕對是成功的密碼分析。**
    - **解析：** 密碼分析不僅關乎破解單個訊息，還關乎**評估加密演算法的整體強度**。發現演算法中的理論弱點（即使沒有具體訊息）是一個重大成就，因為它：
        - 破壞了**所有**使用該演算法保護的訊息的安全。
        - 促使向更強演算法的遷移。
        - 推進了密碼學領域的知識。
    - *Answer: Yes, this is absolutely successful cryptanalysis.*
    - *Explanation: Cryptanalysis isn't just about breaking individual messages; it's about*

***assessing the overall strength of the encryption algorithm**. Finding theoretical weaknesses in an algorithm (even without specific messages) is a major achievement because it:*

- *Compromises the security of **all** messages protected by that algorithm.*
- *Motivates migration to stronger algorithms.*
- *Advances knowledge in the field of cryptography.*

4. **了解源語言的重要性**

   - **答案：** 了解源語言（英語）提供了關於明文**統計特性**的關鍵資訊。密碼分析員可以利用英語的已知特徵：
     - **字母頻率：** 知道'e', 't', 'a', 'o', 'i', 'n'等是最常見的字母。
     - **常見單字：** 知道'the', 'and', 'of', 'to', 'a', 'in'等是常見單字。
     - **字母組合：** 知道哪些雙字母（'th', 'he', 'an'）和三字母組合是常見的。
     - **單字模式：** 知道單字母單字只能是'I'或'a'。
   - 這些模式為分析密文和測試可能的解密方法提供了起點。
   - *Answer: Knowing the source language (English) provides crucial information about the **statistical properties** of the plaintext. A cryptanalyst can leverage known characteristics of English:*
     - ***Letter Frequencies:** Knowing 'e', 't', 'a', 'o', 'i', 'n' are the most common letters.*
     - ***Common Words:** Knowing 'the', 'and', 'of', 'to', 'a', 'in' are common words.*
     - ***Letter Combinations:** Knowing which digraphs ('th', 'he', 'an') and trigraphs are common.*
     - ***Word Patterns:** Knowing that one-letter words can only be 'I' or 'a'.*
   - *These patterns provide a starting point for analyzing the ciphertext and testing possible decryption methods.*

Cryptanalysis by Using Information

Excellent! Now we're diving into the practical side of cryptography: **cryptanalysis** and how attackers use available information to break ciphers.

中英雙語解釋 / Bilingual Explanation

密碼分析攻擊分類 / Cryptanalysis Attack Classification

攻擊者的能力假設很重要： 對加密演算法的攻擊根據攻擊者可用的資訊進行分類。

Assumption of attacker's capability matters: *Attacks are classified according to what information is available to the attacker.*

攻擊類型	攻擊者擁有的資訊
唯密文攻擊	攻擊者只有**加密文字**
Ciphertext-only	*Attacker has only **encrypted text***
已知明文攻擊	攻擊者有一些**密文/明文對**
Known plaintext	*Attacker has some **ciphertext/plaintext pairs***
選擇明文攻擊	攻擊者可以**選擇明文**並取得其加密結果
Chosen plaintext	*Attacker can cause **messages of his choosing** to be encrypted*
自適應選擇明文攻擊	根據先前結果調整的選擇明文攻擊
Adaptive chosen plaintext	*Chosen plaintext attack **adjusted according to earlier results***
選擇密文攻擊	攻擊者可以**解密選定的密文**
Chosen ciphertext	*Attacker can **decrypt selected ciphertext***

破解密碼的任務 / The Task of Breaking a Cipher

密碼分析員的任務是：在給定有限資訊的情況下，從可能的解密空間中提取正確的解密。
*A cryptanalyst's task is **extracting the correct decryption from the space of possible decryptions**, given limited information.*

關鍵問題： 她能從密文和環境中收集多少資訊來**減少搜尋空間**？
*How much can she glean from the ciphertext and circumstances to **reduce the search space**?*

思想實驗：使用資訊減少搜尋空間 / Thought Experiment: Using Information to Reduce Search Space

讓我們透過一個例子來理解資訊如何幫助密碼分析：
Let's understand how information helps cryptanalysis through an example:

問題1：基本替換密碼 / Question 1: Basic Substitution Cipher

- **已知：** `xyy` 使用替換密碼對英文字母表（26個字母）中的字串進行編碼。
- *Suppose you know that* `xyy` *encodes a string using a* ***substitution cipher****.*
- **替換密碼：** 明文的每個字母獨立地交換為另一個字母。
- *In substitution cipher, each letter is exchanged for another* ***independently****.*
- **問題：** 有多少種可能的解密？
- *How many decryptions are possible?*

答案： 26^3 = **17,576**

- *每個位置有26種可能的選擇*
- *Each position has 26 possible choices*

問題2：簡單替換密碼 / Question 2: Simple Substitution Cipher

- **附加資訊：** 這是一個**簡單替換密碼**。
- *Suppose we add that it's a* ***simple substitution cipher****.*
- **簡單替換密碼：** 字母表到自身或另一個字母表的**單射**（一對一映射）。
- *A simple substitution cipher is an* ***injection (1-1 mapping)*** *of the alphabet.*
- **問題：** 現在有多少種可能的解密？
- *How many decryptions are possible now?*

答案： 26 × 25 = **650**

- *第一個字母：26種選擇*
- *First letter: 26 choices*
- *第二個字母：25種選擇（不能與第一個相同）*
- *Second letter: 25 choices (can't be same as first)*
- *第三個字母：必須與第二個字母相同*
- *Third letter: must be same as second*

資訊幫助將搜尋空間減少了約27倍！ (17,576 ÷ 650 ≈ 27)
The information helps reduce search space by a factor of ~27!

問題3：英語單字約束 / Question 3: English Word Constraint

- **附加資訊：** 明文是一個**英語單字**。
- *Suppose we add that the plaintext is an* ***English word****.*
- **問題：** 有多少個拼寫模式為xyy的3字母英語單字？

- *How many 3-letter English words spelled like xyy are there?*

答案： 大約**40**個（如"see", "too", "all", "add", "egg"等）

- *Answer: Around **40** (e.g., "see", "too", "all", "add", "egg", etc.)*

資訊幫助將搜尋空間減少了約439倍！ (17,576 ÷ 40 ≈ 439)
The information helps reduce search space by a factor of ~439!

完美密碼可能嗎？ / Is a Perfect Cipher Possible?

完美密碼是指即使知道以下資訊也無法減少搜尋空間的密碼：
*A **perfect cipher** would be one for which **no reduction of the search space** is gained from knowing:*

- 加密演算法
- *The **encryption algorithm**, and*
- 密文
- *The **ciphertext***
- 以及在攻擊者的不同能力下獲得的資訊（唯密文、已知明文等）
- *Obtained under attacker's different capabilities*

完美密碼的特性： 攻擊者對訊息的**不確定性**（猜測明文的可能性）無論她是否能夠存取密文都**完全相同**。
*The attacker's **uncertainty** about the message is **exactly the same** whether or not she has access to the ciphertext.*

現實世界的啟示 / Real-World Implications

這個思想實驗展示了密碼分析的基本原理：
This thought experiment demonstrates fundamental principles of cryptanalysis:

1. **每個資訊片段都很重要：** 即使是關於加密演算法類型或預期明文的微小資訊也能顯著減少搜尋空間。

- ***Every piece of information matters:** Even small information about the encryption type or expected plaintext can dramatically reduce the search space.*

1. **語言特性是弱點：** 自然語言的冗餘和模式為密碼分析員提供了統計槓桿。

- ***Language properties are weaknesses:*** *The redundancy and patterns in natural languages provide statistical leverage to cryptanalysts.*

1. **完美加密很難實現：** 建立對**所有類型**攻擊都安全的密碼非常困難。

- ***Perfect encryption is hard:*** *Creating ciphers secure against* ***all types*** *of attacks is extremely difficult.*

小測驗 / Test

1. 在問題1到問題3的思想實驗中，什麼類型的資訊對減少搜尋空間的貢獻最大？為什麼？
    - *In the thought experiment from Question 1 to Question 3, which type of information contributed most to reducing the search space? Why?*
2. 如果一個密碼系統能夠抵抗已知明文攻擊，它是否自動能夠抵抗唯密文攻擊？為什麼？
    - *If a cryptosystem is resistant to known plaintext attacks, is it automatically resistant to ciphertext-only attacks? Why?*
3. 為什麼說自然語言的"冗餘"是密碼分析員的優勢，但卻是密碼設計者的挑戰？
    - *Why is the "redundancy" of natural languages an advantage for cryptanalysts but a challenge for cipher designers?*
4. 完美密碼在現實世界中實際可行嗎？請解釋你的推理。
    - *Is a perfect cipher practically achievable in the real world? Explain your reasoning.*

答案與解析 / Answers and Explanations

1. **最有效的資訊類型**

    - **答案： "明文是一個英語單字"** 這一資訊貢獻最大。
    - **解析：** 這一資訊將搜尋空間從650種可能性減少到約40種，減少了約**16倍**（650 ÷ 40 ≈ 16）。相比之下，知道是"簡單替換密碼"而不是基本替換密碼將搜尋空間減少了約27倍（17,576 ÷ 650 ≈ 27）。關於**預期內容語義**的資訊通常比關於**加密機制**的技術資訊更強大，因為它直接限制了可能的有效解密集合。
    - *Answer: The information that* ***"the plaintext is an English word"*** *contributed the*

most.

- *Explanation: This information reduced the search space from 650 possibilities to about 40, a reduction of about **16x** (650 ÷ 40 ≈ 16). In comparison, knowing it was a "simple substitution cipher" rather than a basic one reduced the space by about 27x (17,576 ÷ 650 ≈ 27). Information about the **expected content semantics** is often more powerful than technical information about the **encryption mechanism** because it directly constrains the set of possible valid decryptions.*

2. **已知明文 vs 唯密文攻擊**

   - **答案： 是的。**
   - **解析：** 已知明文攻擊比唯密文攻擊**更強**。在已知明文攻擊中，攻擊者擁有額外的資訊（明文-密文對）。如果一個密碼系統能夠抵抗這種更強的攻擊，那麼它肯定能夠抵抗較弱的唯密文攻擊，因為唯密文攻擊者擁有的資訊更少。從搜尋空間的角度來看：如果即使擁有明文-密文對也無法減少不確定性，那麼僅擁有密文肯定也無法減少不確定性。
   - *Answer: Yes.*
   - *Explanation: A known plaintext attack is **stronger** than a ciphertext-only attack. In a known plaintext attack, the attacker has additional information (plaintext-ciphertext pairs). If a cryptosystem is resistant to this stronger attack, it must be resistant to the weaker ciphertext-only attack, where the attacker has less information. In terms of search space: if uncertainty cannot be reduced even with plaintext-ciphertext pairs, it certainly cannot be reduced with ciphertext alone.*

3. **冗餘的雙重性質**

   - **答案：**
     - **對密碼分析員是優勢：** 冗餘建立了**統計模式**（如字母頻率、常見單字、字母組合）。這些模式在密文中仍然存在，為分析員提供了破解密碼的起點。分析員可以利用這些已知的模式來測試假設並減少可能的解密數量。
     - **對密碼設計者是挑戰：** 設計者必須建立能夠**消除或掩蓋**這些自然語言模式的加密演算法。好的加密應該產生在統計上與隨機雜訊無法區分的密文，使得基於冗餘的模式識別失效。
   - *Answer:*
     - ***Advantage for cryptanalysts:** Redundancy creates **statistical patterns** (letter frequencies, common words, letter combinations). These patterns persist in the ciphertext, giving analysts a starting point for breaking the cipher. Analysts can leverage these known patterns to test hypotheses and reduce the number of*

possible decryptions.

- ***Challenge for designers:*** *Designers must create encryption that* ***eliminates or masks*** *these natural language patterns. Good encryption should produce ciphertext that is statistically indistinguishable from random noise, rendering pattern recognition based on redundancy ineffective.*

4. **完美密碼的可行性**

   - **答案：** 完美密碼在**理論上是存在的**（如一次一密亂碼本），但在**實際中通常不可行**用於大多數應用。
   - **解析：**
     - **理論存在：** **一次一密亂碼本** 如果正確使用，是理論上完美的（不可破解的）。它透過使用與訊息一樣長且真正隨機的金鑰來實現這一點。
     - **實際限制：**
       - **金鑰分發：** 安全地分發與所有訊息總和一樣長的金鑰是不切實際的。
       - **金鑰管理：** 產生、儲存和保護大量隨機金鑰非常困難。
       - **效率：** 對於大多數實際應用來說，金鑰大小與訊息大小相同是低效的。
     - 因此，現代密碼學使用**計算安全**的密碼：它們不是理論上完美的，但在實際計算能力下被認為是不可破解的。
   - *Answer: A perfect cipher is* ***theoretically possible*** *(e.g., the one-time pad) but* ***generally not practical*** *for most applications.*
   - *Explanation:*
     - ***Theoretical existence:*** *The* ***one-time pad****, if used correctly, is theoretically perfect (unbreakable). It achieves this by using a key as long as the message and truly random.*
     - ***Practical limitations:***
       - ***Key distribution:*** *Securely distributing keys as long as all messages combined is impractical.*
       - ***Key management:*** *Generating, storing, and protecting vast amounts of random keys is very difficult.*
       - ***Efficiency:*** *Having key size equal to message size is inefficient for most practical applications.*
     - *Therefore, modern cryptography uses* ***computationally secure*** *ciphers: they are not theoretically perfect but are considered unbreakable with practical computational power.*

A Perfect Cipher

中英雙語解釋 / Bilingual Explanation

完美密碼的定義 / Definition of a Perfect Cipher

完美密碼是指即使知道以下資訊也無法減少搜尋空間的密碼：
*A **perfect cipher** would be one for which **no reduction of the search space** is gained from knowing:*

- 加密演算法
- *The **encryption algorithm**, and*
- 密文
- *The **ciphertext***

夏農的證明： 夏農證明了一個完美密碼需要**與明文一樣多的可能金鑰**，且金鑰是隨機選擇的。
*Shannon proved that a perfect cipher requires **as many possible keys as plaintexts**, with the key chosen **randomly**.*

一次一密亂碼本 / One-Time Pad

一次一密亂碼本由Miller（1882年）以及Vernam和Mauborgne（1917年）獨立發明，是理論上的完美密碼。
*A **one-time pad**, invented by Miller and independently by Vernam and Mauborgne, is a **theoretically perfect cipher**.*

核心思想： 使用一個**與明文長度相同的金鑰**，並且**只使用一次**。
*The idea is to use a key that is the **same length as the plaintext**, and to use it **only once**.*

工作原理： 金鑰與明文進行**互斥或**操作。
*The key is **XORed** with the plaintext.*

特性： 明文空間、密文空間和金鑰空間都相同（例如，都是15位二進位字串）。
*The space of plaintexts, ciphertexts, and keys are all the **same**.*

為什麼一次一密是完美的？ / Why is the One-Time Pad Perfect?

考慮3位訊息的空間。假設攻擊者截獲了密文（`101`），並且知道正在使用一次一密。
Consider the space of three-bit messages. Suppose the attacker intercepts the ciphertext

`(101)` *and knows a one-time pad is in use.*

可能的明文	所需的金鑰	解釋
`000`	`101`	`000 ⊕ 101 = 101`
`001`	`100`	`001 ⊕ 100 = 101`
`010`	`111`	`010 ⊕ 111 = 101`
`011`	`110`	`011 ⊕ 110 = 101`
`100`	`001`	`100 ⊕ 001 = 101`
`101`	`000`	`101 ⊕ 000 = 101`
`110`	`011`	`110 ⊕ 011 = 101`
`111`	`010`	`111 ⊕ 010 = 101`

關鍵洞察： 每個可能的明文都可能是該密文在某個合理金鑰下的原像。
Key Insight: Every possible plaintext *could be the pre-image of that ciphertext under a plausible key.*

因此，搜尋空間的減少是不可能的！
Therefore, no reduction of the search space is possible!

攻擊者看到密文 `101` 時，完全無法判斷原始明文是 `000` 、 `001` 、 `010` 還是任何其他3位訊息。所有可能性都同樣合理。
The attacker seeing ciphertext `101` *has no way to tell if the original plaintext was* `000` , `001` , `010` *, or any other 3-bit message. All are equally plausible.*

為什麼金鑰必須是隨機的？ / Why Does the Key Need to be Random?

如果金鑰不是真正隨機的，就會引入**模式**和**偏見**，攻擊者可以利用這些來減少搜尋空間。
If the key is not truly random, it introduces ***patterns*** *and* ***biases*** *that an attacker can exploit to reduce the search space.*

例子： 如果金鑰是從一本小說中提取的文字，那麼金鑰將具有英語的統計特性（字母頻率、常見單字等），這些特性可能會在密文中洩露。
Example: If the key is text extracted from a novel, the key will have statistical properties of English (letter frequencies, common words, etc.) that might leak into the ciphertext.

金鑰分發問題 / The Key Distribution Problem

一次一密的主要問題是**實際的**，而不是理論的。
*The main problem with the one-time pad is **practical**, rather than theoretical.*

金鑰分發困境：
The Key Distribution Dilemma:

- 如果傳送方和接收方已經有一個安全通道，為什麼他們還需要金鑰？
- *If sender and receiver already have a **secure channel**, why do they need the key?*
- 如果沒有安全通道，他們如何安全地分發金鑰？
- *If they **don't**, how do they distribute the key **securely**?*

這是一個在實踐中很重要的問題。
*This is an **important issue in practice**.*

一次一密的近似實現 / One-Time Pad Approximation

解決方案： 使用**偽隨機數產生器**來產生金鑰。
*Approximate the one-time pad using a **PRNG** to generate a key.*

工作原理：

- **PRNG：** 偽隨機數產生器
- ***PRNG:** pseudorandom number generator*
- 接收端的電腦執行相同的亂數產生器函式，可以從種子（通常作為秘密）產生金鑰。
- *The receiver running the same PRNG function can produce the key from the **seed** (often kept secret).*

優勢： 偽隨機序列可能具有很長的周期。
*This works well because a pseudorandom sequence may have a **very long period**.*

啟發： 這啟發了現代串流密碼的設計。
*Inspired the modern **stream cipher** designs.*

例子： AES-CTR, AES-OFB, AES-CFB, RC4等。
Examples: AES-CTR, AES-OFB, AES-CFB, RC4, etc.

一次一密的工作原理範例 / One-Time Pad Working Example

假設明文是： `HELLO` （ASCII: 01001000 01000101 01001100 01001100 01001111）
Assume plaintext: `HELLO` (ASCII: 01001000 01000101 01001100 01001100 01001111)

步驟1：產生隨機金鑰（與明文等長）
Step 1: Generate random key (same length as plaintext)

- 金鑰： `5*Fq@` （ASCII: 00110101 00101010 01000110 01110001 01000000）
- *Key: `5*Fq@` (ASCII: 00110101 00101010 01000110 01110001 01000000)*

步驟2：逐位互斥或加密
Step 2: XOR encryption bit by bit

```
明文:       01001000 01000101 01001100 01001100 01001111  (HELLO)
金鑰:       00110101 00101010 01000110 01110001 01000000  (5*Fq@)
互斥或:     ----------------------------------------
密文:       01111101 01101111 00001010 00111101 00001111  (}o\n=▯)
```

步驟3：解密使用相同的金鑰
Step 3: Decryption uses the same key

```
密文:       01111101 01101111 00001010 00111101 00001111  (}o\n=▯)
金鑰:       00110101 00101010 01000110 01110001 01000000  (5*Fq@)
互斥或:     ----------------------------------------
明文:       01001000 01000101 01001100 01001100 01001111  (HELLO)
```

小測驗 / Test

1. 為什麼說一次一密是"理論上完美"的？這種完美性在什麼假設下成立？
   - *Why is the one-time pad said to be "theoretically perfect"? Under what assumptions does this perfection hold?*
2. 金鑰分發問題為什麼使一次一密在大多數實際應用中不切實際？
   - *Why does the key distribution problem make the one-time pad impractical for most real-world applications?*
3. 如果相同的金鑰被用於加密兩條不同的訊息，為什麼一次一密的安全性會被完全破壞？
   - *If the same key is used to encrypt two different messages, why is the one-time pad's security completely compromised?*
4. 現代串流密碼（如AES-CTR）如何近似實現一次一密的理想特性，同時避免其實際限制？

- *How do modern stream ciphers (like AES-CTR) approximate the ideal properties of the one-time pad while avoiding its practical limitations?*

答案與解析 / Answers and Explanations

1. **一次一密的完美性**
   - **答案：** 一次一密是理論上完美的，因為**給定一個密文，每個可能的明文都同樣可能是原始訊息**。攻擊者即使擁有無限的計算資源，也無法獲得任何關於明文的資訊。
   - **成立的假設：**
     - 金鑰是**真正隨機**的
     - 金鑰**與明文等長**
     - 每個金鑰**只使用一次**
     - 金鑰**絕對保密**
   - *Answer: The one-time pad is theoretically perfect because **given a ciphertext, every possible plaintext is equally likely to be the original message**. An attacker gains zero information about the plaintext, even with infinite computational resources.*
   - *Assumptions required:*
     - *Key is **truly random***
     - *Key is **same length** as plaintext*
     - *Each key is **used only once***
     - *Key is **kept completely secret***
2. **金鑰分發問題的實用性**
   - **答案：** 金鑰分發問題使一次一密不切實際，因為它要求**安全地分發與所有要傳送的訊息總長度相等的金鑰材料**。例如，要安全傳輸1GB的資料，你需要預先安全地共用1GB的隨機金鑰。如果已經存在能夠傳輸1GB金鑰的安全通道，那麼你可能不需要加密；如果沒有安全通道，你就無法建立金鑰。這導致了**先有雞還是先有蛋**的困境。
   - *Answer: The key distribution problem makes it impractical because it requires **securely distributing key material equal in length to the total of all messages to be sent**. For example, to securely transmit 1GB of data, you need to pre-share 1GB of random keys. If you already have a secure channel to transmit 1GB of keys, you might not need encryption; if you don't have a secure channel, you can't establish the keys. This creates a **chicken-and-egg dilemma**.*
3. **金鑰重用的危險性**

- **答案：** 如果相同的金鑰K用於加密兩條訊息M_1和M_2，攻擊者可以計算：
  - $C_1 \oplus C_2 = (M_1 \oplus K) \oplus (M_2 \oplus K) = M_1 \oplus M_2$
- 這**完全消除了金鑰**，留下了兩條明文的互斥或結果。攻擊者然後可以利用自然語言的統計特性（如英語中空格字元的已知模式、常見單字等）來分離出兩條原始訊息。這種攻擊被稱為"兩次墊密攻擊"。
- *Answer: If the same key K encrypts two messages M_1 and M_2, an attacker can compute:*
  - $C_1 \oplus C_2 = (M_1 \oplus K) \oplus (M_2 \oplus K) = M_1 \oplus M_2$
- *This **completely eliminates the key**, leaving the XOR of the two plaintexts. The attacker can then exploit statistical properties of natural language (known patterns of space characters in English, common words, etc.) to separate out the two original messages. This attack is known as a "two-time pad attack".*

4. **現代串流密碼的近似**

- **答案：** 現代串流密碼透過使用**短金鑰和偽隨機數產生器**來近似一次一密：
  - **短金鑰：** 使用一個較短的秘密金鑰（如128位元或256位元）而不是與訊息等長的金鑰。
  - **PRNG：** 使用密碼學安全的偽隨機數產生器從短金鑰擴展出**長的金鑰流**。
  - **類似操作：** 然後將這個產生的金鑰流與明文進行互斥或，就像一次一密一樣。
- 這種方法避免了分發大量金鑰的問題，同時如果PRNG是密碼學安全的，仍能提供強大的安全性。
- *Answer: Modern stream ciphers approximate the one-time pad by using a **short key and a pseudorandom number generator (PRNG)**:*
  - ***Short Key:** Use a short secret key (e.g., 128 or 256 bits) instead of a message-length key.*
  - ***PRNG:** Use a cryptographically secure PRNG to expand the short key into a **long keystream**.*
  - ***Similar Operation:** XOR this generated keystream with the plaintext, just like the one-time pad.*
- *This approach avoids the problem of distributing massive keys while still providing strong security if the PRNG is cryptographically secure.*

Properties of Ciphers

中英雙語解釋 / Bilingual Explanation

優秀加密的標準 / What is Good Encryption?

當前密碼學實踐中，以下被建議作為價值的測試標準：
The following are suggested as tests of worth for current cryptographic practice:

1. **基於可靠的數學**
   - ***Based on sound mathematics***
2. **經過有能力的專家分析並被發現是可靠的**
   - ***Analyzed by competent experts and found to be sound***
3. **經受了時間的考驗**
   - ***Stood the test of time***

可破解的加密 / Breakable Encryption

定義： 如果給定足夠的時間和資料，分析員能夠恢復明文，則加密演算法被稱為**可破解的**。
*An encryption algorithm is called **breakable** if, given enough time and data, an analyst can recover the plaintext.*

重要說明：

- **大多數加密演算法都是可破解的**，因為分析員可以系統地嘗試所有金鑰。
- ***Most encryption algorithms are breakable** since the analyst can try all keys systematically.*
- 分析員必須能夠**識別成功**。
- *The analyst must be able to **recognize success**.*
- 因此，通常需要擁有**明文/密文對**。
- *For that reason, having **plaintext/ciphertext pairs** is often required.*

關鍵區別： 可破解**並不意味著**破解是可行的！
Being breakable doesn't mean that it's feasible to break!

強加密 / Strong Encryption

定義： 如果**沒有比暴力破解快得多的分析方法**，則密碼系統是強的。
*A cryptosystem is **strong** if there is no analytic approach that is **substantially faster than brute force**.*

- **暴力破解：** 逐個嘗試所有金鑰。

- ***Brute force:*** *Trying all of the keys one by one.*
- **金鑰空間越大**，透過搜尋找到金鑰的時間越長。
- *The **larger the key space**, the longer to find the key by search.*

現代實踐：

- 許多密碼使用n位字串作為金鑰
- *Many ciphers use a n-bit string as key*
- 在現代對稱加密中，**256位元**是目前推薦的金鑰大小
- *In modern symmetric encryption, **256-bit** is the recommended key size*

重要說明： 大多數強演算法**仍然是可以破解的**（理論上），只是實際上不可行。
Most strong algorithms are still breakable *(theoretically), just not practically.*

密碼的建構模組 / Building Blocks of Ciphers

加密最簡單的建構模組是：
The simplest building blocks of encryption are:

1. **替換：** 每個符號被交換為另一個符號（不一定均勻地）
   - ***Substitution:*** *Each symbol is exchanged for another*
2. **置換：** 符號的順序被重新排列
   - ***Transposition:*** *The order of symbols is rearranged*

看似簡單但強大： 這些可能看起來太簡單而無效，但**幾乎所有現代商業對稱密碼**都使用某種替換和置換的組合進行加密。
*It might seem naive, but **almost all modern commercial symmetric ciphers** use some combination of substitution and transposition.*

AES範例 / AES Examples

AES替換步驟：
AES Substitution Step:

- 16位元組陣列中的每個位元組被替換為固定8位元查詢表中的對應條目
- *Each byte in a 16-byte array is replaced with a corresponding entry from a fixed 8-bit lookup table*

AES置換步驟：
AES Transposition Step:

- 重新排列16位元組陣列中的位元組順序
- *Reorders the bytes in a 16-byte array*

混淆與擴散 / Confusion and Diffusion

加密步驟可以提供兩樣東西：
Two things an encryption step can provide are:

1. **混淆：** 轉換明文中的資訊，使攔截者無法輕易提取它。
   - ***Confusion:*** *Transforming information so an interceptor cannot readily extract it.*
2. **擴散：** 將來自明文區域的資訊廣泛散佈到密文中。
   - ***Diffusion:*** *Spreading information widely over the ciphertext.*

分工：

- **替換**傾向於擅長**混淆**
- ***Substitution** tends to be good at **confusion***
- **置換**傾向於擅長**擴散**
- ***Transposition** tends to be good at **diffusion***

混淆與擴散的詳細解釋 / Detailed Explanation of Confusion and Diffusion

混淆 / Confusion

目標： 使金鑰和密文之間的關係盡可能複雜。
Goal: Make the relationship between the key and ciphertext as complex as possible.

如何工作：

- 明文的每個部分以複雜的方式依賴於金鑰的多個部分
- *Each part of the plaintext depends on multiple parts of the key in complex ways*
- 改變金鑰的一位應該導致密文發生不可預測的變化
- *Changing one bit of the key should cause unpredictable changes throughout the ciphertext*

例子： 在AES中，S-box（替換盒）提供了強大的混淆
Example: In AES, the S-box (substitution box) provides strong confusion

擴散 / Diffusion

目標： 將明文的統計特性散佈到密文中。
Goal: Spread the statistical properties of the plaintext throughout the ciphertext.

如何工作：

- 明文的每個部分影響密文的多個部分
- *Each part of the plaintext affects multiple parts of the ciphertext*
- 改變明文的一位應該導致密文大約一半的位發生變化
- *Changing one bit of the plaintext should change approximately half the bits in the ciphertext*

例子： 在AES中，行移位和列混合步驟提供了擴散
Example: In AES, the ShiftRows and MixColumns steps provide diffusion

---

實際例子：AES輪結構 / Practical Example: AES Round Structure

AES透過多輪操作實現強加密：
AES achieves strong encryption through multiple rounds:

每輪包含：
Each round contains:

1. **SubBytes** - 替換（提供**混淆**）
2. **ShiftRows** - 置換（提供**擴散**）
3. **MixColumns** - 置換（提供**擴散**）
4. **AddRoundKey** - 與輪金鑰進行XOR

透過重複這些操作，AES實現了：
By repeating these operations, AES achieves:

- **深度混淆：** 金鑰和密文之間的關係極其複雜
- ***Deep confusion:*** *Relationship between key and ciphertext is extremely complex*
- **完全擴散：** 明文的每一位影響密文的每一位
- ***Complete diffusion:*** *Every plaintext bit affects every ciphertext bit*

小測驗 / Test

1. 為什麼說"大多數加密演算法都是可破解的"，但這在現實中通常不是問題？
   - *Why are "most encryption algorithms breakable," but why is this usually not a problem in reality?*
2. "強加密"和"完美加密"之間有什麼區別？
   - *What is the difference between "strong encryption" and "perfect encryption"?*
3. 如果一個加密演算法只使用替換（沒有置換），它可能在哪個安全屬性上較弱？為什麼？
   - *If an encryption algorithm uses only substitution (no transposition), in which security property might it be weak? Why?*
4. 在AES中，如果改變明文的一個位，密文中大約一半的位會發生變化。這體現了哪個密碼學屬性？為什麼這個屬性很重要？
   - *In AES, changing one bit of plaintext changes approximately half the bits in the ciphertext. Which cryptographic property does this demonstrate? Why is this property important?*

答案與解析 / Answers and Explanations

1. **可破解性 vs 可行性**
   - **答案：** 大多數加密演算法在**理論上是可破解的**，因為攻擊者可以嘗試所有可能的金鑰（暴力攻擊）。然而，對於強加密演算法，金鑰空間如此之大，以至於即使使用所有現有的計算資源，暴力攻擊也需要**不可行的時間**（如數十億年）。因此，雖然理論上可破解，但**實際上不可行**。
   - *Answer: Most encryption algorithms are **theoretically breakable** because an attacker could try all possible keys (brute force). However, for strong encryption, the key space is so large that brute force would take **infeasible time** (e.g., billions of years) even with all existing computing resources. So while theoretically breakable, it's **practically infeasible**.*
2. **強加密 vs 完美加密**
   - **答案：**
     - **完美加密**（如一次一密亂碼本）是**資訊理論安全**的 - 即使攻擊者擁有無限的計算資源，也無法獲得關於明文的任何資訊。
     - **強加密**是**計算安全**的 - 沒有已知的攻擊方法比暴力搜尋所有金鑰更快。對於足夠大的金鑰大小，使用實際的計算資源進行暴力攻擊是不可行的。

- 完美加密提供絕對的安全性，但通常不實用；強加密提供基於當前計算可行性的安全性。
- *Answer:*
  - ***Perfect encryption*** *(e.g., one-time pad) is* ***information-theoretically secure*** *- even with infinite computational resources, an attacker gains zero information about the plaintext.*
  - ***Strong encryption*** *is* ***computationally secure*** *- no known attack is faster than brute force searching all keys. For sufficiently large key sizes, brute force is infeasible with practical computational resources.*
- *Perfect encryption provides absolute security but is often impractical; strong encryption provides security based on current computational feasibility.*

3. **純替換的弱點**
   - **答案：** 純替換演算法可能在**擴散**方面較弱。
   - **原因：** 在純替換中，明文的每個符號被獨立地替換。這意味著：
     - 明文的**統計特性**（如字母頻率）在密文中被保留。
     - 改變明文的一個位通常只影響密文中的一個位（或一個小區域）。
     - 攻擊者可以利用頻率分析和模式識別來破解密碼。
   - 沒有置換，就沒有機制將區域性變化傳播到整個訊息中。
   - *Answer: A pure substitution algorithm would likely be weak in* ***diffusion****.*
   - *Reason: In pure substitution, each symbol of plaintext is replaced independently. This means:*
     - ***Statistical properties*** *of the plaintext (like letter frequencies) are preserved in the ciphertext.*
     - *Changing one bit of plaintext typically affects only one bit (or a small local area) of the ciphertext.*
     - *Attackers can use frequency analysis and pattern recognition to break the cipher.*
   - *Without transposition, there's no mechanism to propagate local changes throughout the entire message.*

4. **雪崩效應的重要性**
   - **答案：** 這體現了**擴散**屬性，具體表現為**雪崩效應**。
   - **為什麼重要：**
     - **隱藏模式：** 防止攻擊者透過分析密文中的模式來推斷明文的結構。
     - **增加難度：** 使密碼分析更加困難，因為小的變化會產生大的、不可預測的影響。

        - **統計隨機性：** 確保密文在統計上與隨機雜訊無法區分，即使明文有明確的模式。
    - 這種屬性是強加密演算法的標誌，因為它意味著演算法徹底地打亂了輸入。
    - *Answer: This demonstrates the **diffusion** property, specifically the **avalanche effect**.*
    - *Why important:*
        - ***Hides patterns:** Prevents attackers from inferring plaintext structure by analyzing patterns in the ciphertext.*
        - ***Increases difficulty:** Makes cryptanalysis much harder because small changes have large, unpredictable effects.*
        - ***Statistical randomness:** Ensures ciphertext is statistically indistinguishable from random noise, even if the plaintext has clear patterns.*
    - *This property is a hallmark of strong encryption algorithms because it means the algorithm thoroughly scrambles the input.*

Symmetric vs. Asymmetric Encryptions

中英雙語解釋 / Bilingual Explanation

對稱 vs 非對稱加密 / Symmetric vs. Asymmetric Encryption

兩種基本加密類型：
Two basic types of encryption:

類型	別名	金鑰使用
對稱演算法	"私密金鑰"或"秘密金鑰"演算法	加密和解密使用**相同的金鑰**
Symmetric algorithms	*"Private key" or "secret key"*	*Use the **same key** for both encryption and decryption*
非對稱演算法	"公開金鑰"演算法	加密和解密使用**不同的金鑰**
Asymmetric algorithms	*"Public key"*	*Use **different keys** for encryption and decryption*

加密的主要挑戰 / Major Challenges in Encryption

對於任何加密方法，都有兩個主要挑戰：
For any encryption approach, there are two major challenges:

1. **金鑰分發：** 我們如何將金鑰傳遞給需要建立安全通訊的人。
    - ***Key distribution:*** *How do we convey keys to those who need them to establish secure communication.*
2. **金鑰管理：** 給定大量金鑰，我們如何保持它們的安全並在需要時提供。
    - ***Key management:*** *Given a large number of keys, how do we preserve their safety and make them available as needed.*

非對稱加密入門 / Asymmetric Encryption Primer

在非對稱或公開金鑰加密中，加密和解密使用不同的金鑰。
In asymmetric or public key encryption, ***different keys*** *are used for encryption and decryption.*

工作原理：
How it works:

- 每個主體A有一個**公開披露的金鑰PKA**（她的公鑰），任何人都可以使用它來加密。
- *Each subject A has a* ***publicly disclosed key PKA*** *that anyone can use to encrypt.*
- 每個主體A有一個**私人持有的金鑰SKA**（她的私鑰），只有她自己知道。
- *Each subject A has a* ***privately held key SKA*** *(her private key).*
- 關係是： $M = M_{PKA}{}_{SKA}$
- *The relationship is:* $M = M_{PKA}{}_{SKA}$

通訊流程：
Communication process:

- 任何希望向A發送保密訊息M的人傳送 M_{PKA} 。
- *Anyone wishing to send a message M confidentially to A sends* M_{PKA} *.*
- 只有SKA的持有者可以解密此訊息。
- *Only the holder of SKA can decrypt this message.*

非對稱加密在很大程度上解決了金鑰分發問題。
Asymmetric encryption largely solves the key distribution problem.

金鑰數量比較 / Key Quantity Comparison

對稱加密的金鑰數量 / Symmetric Encryption Key Count

給定一個有n個使用者的對稱系統，需要多少金鑰進行成對安全通訊？
Given a symmetric system with n users, how many keys are needed for pairwise secure communication?

- 每次向系統新增新使用者時，它需要與每個現有使用者共享一個新金鑰。
- *Each time a new user is added, it needs to share a new key with each previous user.*
- 因此，對於n個使用者，我們有：
- *Thus, for n users, we have:*
  - `1 + 2 + ⋯ + (n − 1) = n(n − 1)/2` 個金鑰
- 這是**O(n²)** 個金鑰。
- *This is **O(n²)** keys.*

例子：

- 10個使用者：10×9÷2 = **45個金鑰**
- 100個使用者：100×99÷2 = **4,950個金鑰**
- 1000個使用者：1000×999÷2 = **499,500個金鑰**

非對稱加密的金鑰數量 / Asymmetric Encryption Key Count

給定一個有n個使用者的非對稱系統，需要多少金鑰進行成對安全通訊？
Given an asymmetric system with n users, how many keys are needed?

- 每次向系統新增新使用者時，它只需要一個公鑰和一個私鑰。
- *Each time a new user is added, it needs only a public key and a private key.*
- 因此，對於n個使用者，我們有**2n**個金鑰，即**O(n)**。
- *Thus, for n users, we have **2n** keys, which is **O(n)**.*

根據演算法， 每個使用者可能需要獨立的金鑰對用於保密性和簽章：
Depending on the algorithm, each user may need separate pairs for confidentiality and signing:

- 即**4n**個金鑰，仍然是**O(n)**。
- *i.e., **4n** keys, which is still **O(n)**.*

例子：

- 10個使用者：**20個金鑰**（或40個用於雙重用途）
- 100個使用者：**200個金鑰**（或400個）

- 1000個使用者：**2000個金鑰**（或4000個）

金鑰特性比較 / Key Characteristics Comparison

對稱金鑰特性 / Symmetric Key Characteristics

通常，在對稱加密系統中，金鑰是：
Typically, in a symmetric encryption system keys are:

- **隨機產生的k位字串**
- ***Randomly generated k-bit strings***
- **簡單產生**
- ***Simple to generate***
- **沒有特殊屬性**
- ***Have no special properties***

非對稱金鑰特性 / Asymmetric Key Characteristics

在公開金鑰系統中，金鑰：
In a public key system, keys:

- **具有特殊結構**（例如，是大質數）
- ***Have special structure*** *(e.g., are large primes)*
- **產生昂貴**
- ***Are expensive to generate***

重要說明： 兩種方法的金鑰大小**不可比較**。
Key sizes are not comparable between the two approaches.

- 一個**256位對稱金鑰**可能在強度上等同於一個**2048位公開金鑰**。
- *A **256-bit symmetric key** may be equivalent in strength to a **2048-bit public key**.*

實際應用場景 / Practical Application Scenarios

對稱加密使用場景 / When to Use Symmetric Encryption

- **大資料加密**（檔案、磁碟、資料庫）

- ***Bulk data encryption** (files, disks, databases)*
- **高效能要求**的場景
- ***High-performance requirements***
- **雙方已經安全共享金鑰**的情況
- ***When parties already securely share a key***

非對稱加密使用場景 / When to Use Asymmetric Encryption

- **安全金鑰分發**
- ***Secure key distribution***
- **數位簽章**
- ***Digital signatures***
- **安全初始連線建立**
- ***Secure initial connection establishment***
- **需要與多方安全通訊**
- ***Need to communicate securely with multiple parties***

現實世界例子：HTTPS / Real-World Example: HTTPS

HTTPS結合了兩種加密類型：
HTTPS combines both encryption types:

1. **非對稱加密：** 用於**安全地交換對稱金鑰**
   - ***Asymmetric encryption:** Used to **securely exchange a symmetric key***
2. **對稱加密：** 用於**加密實際的資料傳輸**
   - ***Symmetric encryption:** Used to **encrypt the actual data transmission***

這種混合方法利用了兩種類型的優勢。
This hybrid approach leverages the strengths of both types.

小測驗 / Test

1. 為什麼對稱加密在使用者數量增長時會出現"金鑰爆炸"問題？
   - *Why does symmetric encryption suffer from "key explosion" as the number of users grows?*

2. 非對稱加密如何解決金鑰分發問題？它引入了什麼新挑戰？
    - *How does asymmetric encryption solve the key distribution problem? What new challenge does it introduce?*
3. 在一個有50個使用者的組織中，對稱加密和非對稱加密分別需要多少金鑰？展示你的計算。
    - *In an organization with 50 users, how many keys are needed for symmetric vs. asymmetric encryption? Show your calculations.*
4. 為什麼說"256位對稱金鑰可能等同於2048位公鑰"？這告訴我們關於兩種加密類型的什麼資訊？
    - *Why might a "256-bit symmetric key be equivalent to a 2048-bit public key"? What does this tell us about the two encryption types?*

答案與解析 / Answers and Explanations

1. **對稱加密的金鑰爆炸問題**
    - **答案：** 因為對稱加密需要**每對使用者共享一個唯一的金鑰**。當新增第n個使用者時，它需要與所有現有的(n-1)個使用者建立新的共享金鑰。金鑰總數以**組合數n(n-1)/2**增長，這是**二次方增長O(n²)**。對於大型組織，這導致需要管理數萬甚至數百萬個金鑰。
    - *Answer: Because symmetric encryption requires **each pair of users to share a unique secret key**. When adding the nth user, it needs to establish new shared keys with all existing (n-1) users. The total number of keys grows as the **combination formula n(n-1)/2**, which is **quadratic growth O(n²)**. For large organizations, this leads to managing tens of thousands or even millions of keys.*
2. **非對稱加密的解決方案與新挑戰**
    - **答案：** 非對稱加密透過使用**公鑰/私鑰對**解決金鑰分發問題。使用者可以將他們的公鑰自由分發給任何人，而保持私鑰秘密。想要傳送加密訊息的人使用接收者的公鑰，只有接收者可以用他們的私鑰解密。
    - **引入的新挑戰： 金鑰認證** - 如何確保你獲得的公鑰確實屬於聲稱擁有它的人，而不是攻擊者提供的假公鑰。這透過**公開金鑰基礎建設(PKI)** 和**數位憑證**來解決。
    - *Answer: Asymmetric encryption solves key distribution by using **public/private key pairs**. Users can freely distribute their public keys while keeping their private keys secret. Anyone wanting to send an encrypted message uses the recipient's public key, and only the recipient can decrypt it with their private key.*
    - ***New challenge introduced: Key authentication** - How to ensure the public key you*

*obtain genuinely belongs to who it claims, and isn't a fake provided by an attacker. This is solved through **Public Key Infrastructure (PKI)** and **digital certificates**.*

3. **50個使用者的金鑰計算**
    - **對稱加密：**
        - `n(n−1)/2 = 50 × 49 / 2 = 2,450 / 2 = 1,225`個金鑰
    - **非對稱加密：**
        - `2n = 2 × 50 = 100`個金鑰 （僅用於保密性）
        - 或 `4n = 4 × 50 = 200`個金鑰 （用於保密性和簽章）
    - **差異：** 非對稱加密需要的金鑰數量**少12倍**（或6倍，如果考慮雙重用途）。
    - *Symmetric encryption:*
        - `n(n−1)/2 = 50 × 49 / 2 = 2,450 / 2 = 1,225 keys`
    - *Asymmetric encryption:*
        - `2n = 2 × 50 = 100 keys` (for confidentiality only)
        - `or 4n = 4 × 50 = 200 keys` (for confidentiality and signing)
    - *Difference: Asymmetric requires **12x fewer keys** (or 6x fewer considering dual purpose).*

4. **金鑰大小等價性的原因**
    - **答案：** 這種差異存在是因為兩種加密類型基於**不同的數學問題**：
        - **對稱加密**通常基於將輸入徹底打亂的複雜操作（混淆和擴散）。
        - **非對稱加密**基於**計算困難的數學問題**，如大數分解（RSA）或離散對數問題。
    - 攻擊對稱加密通常需要暴力搜尋整個金鑰空間，而攻擊非對稱加密可以利用其數學結構的潛在弱點。
    - 這告訴我們非對稱加密**計算成本更高**，但對於金鑰分發更實用，而對稱加密**更高效**，但面臨金鑰管理挑戰。
    - *Answer: This difference exists because the two encryption types are based on **different mathematical problems**:*
        - ***Symmetric encryption** typically relies on complex operations that thoroughly scramble the input (confusion and diffusion).*
        - ***Asymmetric encryption** relies on **computationally hard mathematical problems** like integer factorization (RSA) or discrete logarithms.*
    - *Attacking symmetric encryption usually requires brute force searching the entire key space, while attacking asymmetric encryption can exploit potential weaknesses in its mathematical structure.*

- *This tells us that asymmetric encryption is **computationally more expensive** but more practical for key distribution, while symmetric encryption is **more efficient** but faces key management challenges.*

Basic Crypto Primitives Symmetric Encryption Scheme

Excellent! Now we're diving into the practical details of **symmetric encryption**, focusing on the most important modern algorithm: **AES (Advanced Encryption Standard)**.

中英雙語解釋 / Bilingual Explanation

現代對稱加密方案 / Modern Symmetric Encryption Scheme

現代對稱加密演算法具有以下特點：
Modern symmetric encryption algorithms are:

- **區塊加密：** 以**固定大小的區塊**處理輸入
- ***Block cipher:** Take input in **fixed size blocks***
- **輪次實作：** 對"狀態"**重複執行**相似的操作
- ***Implemented in rounds:** Perform **similar operations repeatedly** to a "state"
- 這種演算法被稱為**迭代區塊加密**。
- *Such an algorithm is called an **iterated block cipher**.*

設計目標： 快速處理大量文字，使用**廉價且易於實作**的機器操作（算術、位元操作、查表）。
*Designed to process large volumes of text quickly using **cheap and easy to implement** machine operations.*

進階加密標準 / Advanced Encryption Standard (AES)

歷史背景： 1995年，NIST開始尋找一種新的、快速、安全的對稱加密演算法，要求：
In 1995, NIST began a search for a new algorithm that was:

- **未分類的**（非軍事機密）
- ***Unclassified***
- **公開揭露的**
- ***Publicly disclosed***
- **全球範圍內免授權金使用**

- ***Available royalty-free worldwide***
- **128位元區塊的對稱區塊加密演算法**
- ***Symmetric block cipher for 128-bit blocks***
- **可使用128、192和256位元金鑰大小**
- ***Usable with 128, 192, and 256-bit keys***

選擇結果： 從15個競爭者中，荷蘭研究人員Vincent Rijmen和Joan Daemen的**Rijndael演算法**被選為進階加密標準。
*From 15 contenders, the **Rijndael algorithm** was chosen as the AES.*

AES的安全性 / Security of the AES

- AES被納入大量商業加密產品中。
- *AES is incorporated in a large number of commercial encryption products.*
- 該演算法相對較新，但已經過**廣泛分析**。
- *The algorithm has been subjected to **extensive analysis**.*
- **尚未發現缺陷**，但這並不意味著不存在缺陷。
- ***No flaws have been discovered**, but that doesn't mean none exist.*
- AES是**模組化**的，金鑰長度可以擴展，輪次數也可以增加。
- *AES is **modular** - key length can be extended and number of rounds can be increased.*

操作模式的重要性 / Modes of Operation Matter

AES是一個**建構區塊**，其安全性依賴於正確的操作模式：
*AES is a **building block** - its security relies on correct modes of operation:*

- **區塊加密模式：** AES-CBC
- ***Block encryption modes:** AES-CBC*
- **金鑰流生成模式：** AES-CFB, AES-CTR, AES-OFB等
- ***Key stream generation modes:** AES-CFB, AES-CTR, AES-OFB, etc.*

對稱加密的形式化表示 / Formal Notation for Symmetric Encryption

通常，任何對稱（私密金鑰）加密方案都是一對函式：

Generally, any symmetric encryption scheme is a pair of functions:

加密函式： $Enc: \{0,1\}^k \times \{0,1\}* \to \{0,1\}*^{+v}$
Encryption function: $Enc: \{0,1\}^k \times \{0,1\}* \to \{0,1\}*^{+v}$

- **高效且非確定性的函式**（隨機化演算法）
- ***Efficient and non-deterministic function*** *(randomized algorithm)*
- 將k位私密金鑰K和變長明文訊息M映射到比明文長v位的密文訊息 $C = Enc_K(M)$
- *Maps a k-bit private key K and variable-length plaintext M to ciphertext C*
- **密文更長**是由於加密前需要低階訊息填充
- ***Longer ciphertext*** *due to message padding needed before encryption*

解密函式： $Dec: \{0,1\}^k \times \{0,1\}*^{+v} \to \{0,1\}*$
Decryption function: $Dec: \{0,1\}^k \times \{0,1\}*^{+v} \to \{0,1\}*$

- **高效確定性演算法**
- ***Efficient deterministic algorithm***
- 不知道K的攻擊者**不能**從觀察 $C = Enc_K(M)$ 中了解到關於M的任何新資訊（除了M的長度）
- *Attacker must* ***not be able to learn anything new*** *about M from observing C*

AES詳細工作原理 / AES Detailed Working Principle

AES參數 / AES Parameters

參數	值
區塊大小	**128位元**（16位元組）
Block size	***128 bits*** *(16 bytes)*
金鑰大小	**128, 192, 256位元**
Key sizes	***128, 192, 256 bits***
輪次數	10（128位元金鑰），12（192位元），14（256位元）
Number of rounds	*10 (128-bit), 12 (192-bit), 14 (256-bit)*

AES輪結構 / AES Round Structure

每輪包含4個操作：

Each round contains 4 operations:

1. **SubBytes - 位元組替換**（提供**混淆**）
   - ***SubBytes - Byte substitution*** *(provides **confusion**)*
2. **ShiftRows - 行移位**（提供**擴散**）
   - ***ShiftRows - Row shifting*** *(provides **diffusion**)*
3. **MixColumns - 列混合**（提供**擴散**）
   - ***MixColumns - Column mixing*** *(provides **diffusion**)*
4. **AddRoundKey - 輪金鑰加**（與輪金鑰進行XOR）
   - ***AddRoundKey - Round key addition*** *(XOR with round key)*

AES加密過程 / AES Encryption Process

```
明文（128 位元）
    ↓
AddRoundKey（使用初始輪金鑰）
    ↓
[重複9–13輪]:
    SubBytes → ShiftRows → MixColumns → AddRoundKey
    ↓
最後一輪（無MixColumns）:
    SubBytes → ShiftRows → AddRoundKey
    ↓
密文（128 位元）
```

AES操作模式範例 / AES Operation Mode Examples

AES-CBC（密文區塊連結）

AES-CBC (Cipher Block Chaining)

- **特點：** 每個明文區塊在加密前與前一個密文區塊進行XOR
- ***Feature:*** *Each plaintext block is XORed with previous ciphertext block before encryption*
- **優點：** 隱藏明文模式
- ***Advantage:*** *Hides plaintext patterns*
- **缺點：** 不能平行加密
- ***Disadvantage:*** *Cannot parallelize encryption*

AES-CTR（計數器模式）

AES-CTR (Counter Mode)

- **特點：** 加密計數器值，然後與明文進行XOR
- ***Feature:*** *Encrypts counter values, then XORs with plaintext*
- **優點：** 可以平行加密和解密
- ***Advantage:*** *Can parallelize both encryption and decryption*
- **缺點：** 必須確保計數器永不重複
- ***Disadvantage:*** *Must ensure counters never repeat*

小測驗 / Test

1. 為什麼AES被稱為"迭代區塊加密"？這種設計有什麼優勢？
   - *Why is AES called an "iterated block cipher"? What is the advantage of this design?*
2. AES的SubBytes和ShiftRows步驟分別主要提供什麼安全屬性？
   - *What security properties do the SubBytes and ShiftRows steps in AES primarily provide?*
3. 在形式化表示中，為什麼密文比明文長（ $\{0,1\}*^{+v}$ ）？這個額外的長度v通常用於什麼？
   - *In the formal notation, why is the ciphertext longer than the plaintext ($\{0,1\}*^{+v}$)? What is this extra length v typically used for?*
4. 如果一個應用程式需要加密串流資料（如視訊串流），AES-CBC和AES-CTR哪個更合適？為什麼？
   - *If an application needs to encrypt streaming data (like video), would AES-CBC or AES-CTR be more suitable? Why?*

答案與解析 / Answers and Explanations

1. **迭代區塊加密的設計優勢**
   - **答案：** AES被稱為迭代區塊加密是因為它**重複執行相似的操作輪次**來處理每個資料區塊。
   - **優勢：**
     - **簡化設計：** 只需設計和驗證少量基本操作，然後重複它們。
     - **增強安全性：** 每輪都增加更多的混淆和擴散，使密碼分析更加困難。
     - **靈活性：** 可以透過增加輪數來輕鬆增強安全性。

        - **高效實作：** 可以在硬體和軟體中最佳化少量操作的重複執行。
    - *Answer: AES is called an iterated block cipher because it **repeats similar operations in rounds** to process each data block.*
    - *Advantages:*
        - ***Simplified design:** Only need to design and verify a small set of basic operations, then repeat them.*
        - ***Enhanced security:** Each round adds more confusion and diffusion, making cryptanalysis harder.*
        - ***Flexibility:** Security can be easily enhanced by increasing the number of rounds.*
        - ***Efficient implementation:** Can optimize the repetition of few operations in hardware and software.*

2. **SubBytes和ShiftRows的安全屬性**

    - **答案：**
        - **SubBytes主要提供混淆：** 它透過S-box（替換盒）非線性地替換每個位元組，使金鑰和密文之間的關係變得複雜。這防止了攻擊者透過分析輸入-輸出關係來推導金鑰。
        - **ShiftRows主要提供擴散：** 它透過移動行來打亂位元組的位置，確保單個位元組的變化會影響到多個列。這有助於將明文中的變化傳播到整個密文區塊中。
    - *Answer:*
        - ***SubBytes primarily provides CONFUSION:** It non-linearly substitutes each byte through S-boxes, complicating the relationship between the key and ciphertext. This prevents attackers from deriving the key by analyzing input-output relationships.*
        - ***ShiftRows primarily provides DIFFUSION:** It scrambles byte positions by shifting rows, ensuring that a change in one byte affects multiple columns. This helps propagate changes throughout the ciphertext block.*

3. **密文額外長度的用途**

    - **答案：** 額外的長度v通常用於**填充和初始化向量**。
    - **具體用途：**
        - **填充：** 區塊加密需要固定大小的區塊。如果明文長度不是區塊大小的整數倍，需要新增填充位元組。
        - **初始化向量：** 在許多操作模式（如CBC）中，需要隨機化加密過程，通常透過新增IV來實現。
        - **認證資料：** 在一些認證加密模式中，可能需要額外的空間用於認證標籤。

- 這個額外的長度確保了加密方案可以處理任意長度的輸入，並提供語意安全性。
- *Answer: The extra length v is typically used for **padding and initialization vectors**.*
- *Specific uses:*
  - ***Padding:** Block ciphers require fixed-size blocks. If plaintext isn't a multiple of block size, padding bytes are added.*
  - ***Initialization Vectors:** In many modes (like CBC), encryption needs randomization, often achieved by adding an IV.*
  - ***Authentication data:** In some authenticated encryption modes, extra space may be needed for authentication tags.*
- *This extra length ensures the encryption scheme can handle arbitrary-length inputs and provide semantic security.*

4. **串流資料加密的模式選擇**
   - **答案： AES-CTR更合適**用於串流資料加密。
   - **原因：**
     - **隨機存取：** CTR模式允許解密任意部分的資料而無需解密之前的所有內容，這對視訊串流的隨機搜尋很重要。
     - **平行性：** CTR模式可以平行加密和解密，提高了串流媒體應用的效能。
     - **無錯誤傳播：** 在CTR模式中，一個位元的傳輸錯誤只影響該位元，而在CBC中錯誤會傳播到整個區塊。
     - **與串流媒體協定相容：** CTR模式更像串流加密，自然適合連續資料流。
   - *Answer: **AES-CTR would be more suitable** for streaming data encryption.*
   - *Reasons:*
     - ***Random access:** CTR mode allows decrypting any part of the data without decrypting everything before it, important for random seeking in video streams.*
     - ***Parallelism:** CTR mode can parallelize both encryption and decryption, improving performance for streaming applications.*
     - ***No error propagation:** In CTR, a single bit transmission error affects only that bit, while in CBC errors propagate through the entire block.*
     - ***Compatibility with streaming protocols:** CTR mode behaves more like a stream cipher, naturally fitting continuous data streams.*

Basic Crypto Primitives Cryptographic Hash Function

Excellent! Now we're exploring **Cryptographic Hash Functions**, which are fundamental

building blocks for ensuring data integrity in security systems.

中英雙語解釋 / Bilingual Explanation

密碼學雜湊函式概述 / Cryptographic Hash Function Overview

雜湊函式是將可變大小的文字轉換為小資料（通常是固定大小的整數）的函式。
*A **hash function** converts variable-sized text into a small datum, usually a fixed size integer.*

密碼學雜湊函式具有額外的品質要求：
*A **cryptographic hash function** has additional qualities:*

1. **原像抵抗：** 難以構造具有給定雜湊值的文字
   - ***Preimage resistant:*** *Difficult to construct a text that has a given hash*
2. **弱碰撞抵抗：** 難以修改給定文字而不改變其雜湊值
   - ***Weak collision resistant:*** *Difficult to modify a given text without changing its hash*
3. **強碰撞抵抗：** 兩個不同的訊息不太可能具有相同的雜湊值
   - ***Strong collision resistant:*** *Unlikely that two different messages will have the same hash*

雜湊值有時被稱為**訊息摘要**。
*The hash value is sometimes called a **message digest**.*

主要用途： 密碼學雜湊函式用於**保護完整性**。
*Cryptographic hash functions are used to **protect integrity**.*

形式化表示 / Formal Notation

雜湊函式 $h: \{0,1\}^* \to \{0,1\}^n$ 使用高效確定性演算法將變長位字串映射到短的、固定長度的位字串，使得：
A hash function $h: \{0,1\}^* \to \{0,1\}^n$ maps variable-length bit strings to short, fixed-length bit strings such that:

- 對於典型輸入 $X \neq Y$，碰撞 $h(X) = h(Y)$ 的機率被最小化
- *The probability of a collision $h(X) = h(Y)$ for typical inputs $X \neq Y$ is minimized*

密碼學雜湊函式h旨在具有**碰撞抵抗性**，即：
*A cryptographic hash function aims to be **collision resistant**, i.e.:*

- 使任何人在計算上不可行找到任何一對輸入字串 X 和 Y，使得 $h(X) = h(Y)$ 且 $X \neq Y$

- *To make it computationally infeasible to find any pair* `X` *and* `Y` *such that* `h(X) = h(Y)` *with* `X ≠ Y`

實際應用場景 / Practical Application Scenarios

密碼學雜湊函式通常用於**完整性**，而不是保密性。
*Cryptographic hash functions are used for **integrity**, not confidentiality.*

應用例子：

- **文件檢索系統：** 包含法律記錄的系統，知道檢索的副本與儲存的副本相同可能很重要。
  - *In a document retrieval system, it may be important to know the copy retrieved is identical to that stored.*
- **安全通訊系統：** 訊息的正确傳輸可能優先於保密性關注。
  - *In a secure communications system, correct transmission of messages may override confidentiality concerns.*

密碼學雜湊函式以使得任何對檔案的更改都變得明顯的方式將檔案的位元組綁定在一起。
A cryptographic hash function binds the bytes of a file together in a way that makes any alterations apparent.

- 我們說我們**密封**檔案以使其防篡改（實際上是抗篡改）。
- *We say that we **seal** the file to make it tamper-resistant.*

常用標準 / Commonly Used Standards

推薦使用的安全雜湊函式：
Recommended secure hash functions:

- **SHA-2家族**（n = 224, 256, 384, 512）
- ***SHA-2 family** (n = 224, 256, 384, 512)*
  - 用於32位元CPU：SHA-224, SHA-256（約20週期/位元組）
  - *For 32-bit CPUs: SHA-224, SHA-256 (~20 cycles/byte)*
  - 用於64位元CPU：SHA-384, SHA-512, SHA-512/224, SHA-512/256（約10週期/位元組）
  - *For 64-bit CPUs: SHA-384, SHA-512, SHA-512/224, SHA-512/256 (~10 cycles/byte)*
- **SHA-3**（任意輸出長度）

- ***SHA-3*** *(arbitrary output length)*

已棄用的早期嘗試： MD4, MD5（n = 128）和SHA-1（n = 128）已遭受碰撞攻擊，因此**不再推薦**用於需要碰撞抵抗的應用。
Earlier attempts MD4, MD5 and SHA-1 have suffered collisions and are ***no longer recommended*** *for applications requiring collision resistance.*

密碼學雜湊函式的屬性詳解 / Detailed Properties of Cryptographic Hash Functions

1. 原像抵抗 / Preimage Resistance

- **定義：** 給定雜湊值h，難以找到任何訊息M使得h(M) = h
- ***Definition:*** *Given hash value h, hard to find any message M such that h(M) = h*
- **也被稱為：** 單向性
- ***Also known as:*** *One-way property*

2. 第二原像抵抗 / Second Preimage Resistance

- **定義：** 給定訊息M1，難以找到不同的訊息M2使得h(M1) = h(M2)
- ***Definition:*** *Given message M1, hard to find different message M2 such that h(M1) = h(M2)*
- **也被稱為：** 弱碰撞抵抗
- ***Also known as:*** *Weak collision resistance*

3. 碰撞抵抗 / Collision Resistance

- **定義：** 難以找到任何兩個不同的訊息M1和M2使得h(M1) = h(M2)
- ***Definition:*** *Hard to find any two different messages M1 and M2 such that h(M1) = h(M2)*
- **也被稱為：** 強碰撞抵抗
- ***Also known as:*** *Strong collision resistance*

實際應用示例 / Practical Application Examples

檔案完整性驗證 / File Integrity Verification

原始檔案 → 計算雜湊 → 儲存雜湊值
下載檔案 → 計算雜湊 → 與儲存的雜湊值比較

密碼儲存 / Password Storage

使用者密碼 → 計算雜湊 → 儲存雜湊值（而非明文密碼）
登入時輸入密碼 → 計算雜湊 → 與儲存的雜湊值比較

數位簽章 / Digital Signatures

訊息 → 計算雜湊 → 用私鑰加密雜湊 → 數位簽章
驗證時 → 計算訊息雜湊 → 用公鑰解密簽章 → 比較兩個雜湊值

SHA-256範例 / SHA-256 Example

SHA-256是SHA-2家族中最常用的雜湊函式：
SHA-256 is the most commonly used in the SHA-2 family:

- **輸出大小：** 256位元（32位元組）
- ***Output size:** 256 bits (32 bytes)*
- **區塊大小：** 512位元
- ***Block size:** 512 bits*
- **內部狀態：** 八個32位元變數（A,B,C,D,E,F,G,H）
- ***Internal state:** Eight 32-bit variables*

範例輸出：

- `"hello"` → `2cf24dba5fb0a30e26e83b2ac5b9e29e1b161e5c1fa7425e73043362938b9824`
- `"hello!"` → `ce06092fb948d9ffac7d1a376e404b26b7575bcc11ee05a4615fef4fec3a308b`

注意： 即使輸入有微小變化，輸出也完全不同 - 這體現了**雪崩效應**。
*Notice: Even tiny changes in input produce completely different outputs - this demonstrates the **avalanche effect**.*

小測驗 / Test

1. 為什麼說密碼學雜湊函式提供的是"完整性"而不是"保密性"？
   - *Why do cryptographic hash functions provide "integrity" rather than "confidentiality"?*
2. "原像抵抗"和"碰撞抵抗"之間有什麼區別？哪個屬性更強？
   - *What's the difference between "preimage resistance" and "collision resistance"? Which property is stronger?*
3. 在密碼儲存系統中，為什麼儲存密碼的雜湊值比儲存明文密碼更安全？
   - *In password storage systems, why is storing hash values of passwords more secure than storing plaintext passwords?*
4. 如果一個雜湊函式的輸出長度是256位元，透過暴力攻擊找到碰撞的預期工作量是多少？為什麼？
   - *If a hash function has 256-bit output, what is the expected work to find a collision by brute force? Why?*

答案與解析 / Answers and Explanations

1. **完整性 vs 保密性**
   - **答案：** 密碼學雜湊函式提供完整性而不是保密性，因為：
     - **雜湊函式是單向的** - 從雜湊值無法恢復原始訊息，因此它們不隱藏資訊內容。
     - **它們檢測更改** - 如果訊息被篡改，其雜湊值會改變，從而揭示完整性被破壞。
     - **雜湊值本身不包含足夠資訊**來重構原始資料，因此不提供保密性。
   - *Answer: Cryptographic hash functions provide integrity rather than confidentiality because:*
     - ***Hash functions are one-way*** *- you cannot recover the original message from the hash value, so they don't hide information content.*
     - ***They detect changes*** *- if a message is tampered with, its hash value changes, revealing integrity violation.*
     - ***The hash value itself doesn't contain enough information*** *to reconstruct the original data, so it doesn't provide confidentiality.*
2. **原像抵抗 vs 碰撞抵抗**
   - **答案：**
     - **原像抵抗：** 給定雜湊值h，難以找到任何訊息M使得h(M) = h。
     - **碰撞抵抗：** 難以找到任何兩個不同的訊息M1和M2使得h(M1) = h(M2)。

- **碰撞抵抗是更強的屬性**，因為如果你能找到碰撞（兩個不同的訊息具有相同的雜湊值），你可以破壞需要唯一雜湊值的系統。事實上，碰撞抵抗意味著第二原像抵抗，但反之則不成立。
- *Answer:*
  - ***Preimage resistance:*** *Given hash h, hard to find any message M such that h(M) = h.*
  - ***Collision resistance:*** *Hard to find any two different messages M1 and M2 such that h(M1) = h(M2).*
- ***Collision resistance is the stronger property*** *because if you can find collisions (two different messages with same hash), you can break systems that rely on unique hash values. In fact, collision resistance implies second preimage resistance, but not vice versa.*

3. **密碼雜湊儲存的安全性**

   - **答案：** 儲存雜湊值更安全，因為：
     - **防止資料庫洩露**：如果攻擊者竊取密碼資料庫，他們只獲得雜湊值，而不是實際密碼。
     - **原像抵抗**：從雜湊值恢復原始密碼在計算上不可行。
     - **加鹽可以防止彩虹表攻擊**：透過為每個密碼新增隨機鹽值，即使兩個使用者有相同密碼，他們的雜湊值也不同。
   - 在實際系統中，還會使用**慢雜湊函式**（如bcrypt、Argon2）來進一步增加暴力攻擊的難度。
   - *Answer: Storing hash values is more secure because:*
     - ***Prevents database exposure:*** *If attackers steal the password database, they get only hash values, not actual passwords.*
     - ***Preimage resistance:*** *Recovering original passwords from hashes is computationally infeasible.*
     - ***Salting prevents rainbow tables:*** *By adding random salt to each password, even two users with same password have different hashes.*
   - *In practice,* ***slow hash functions*** *(like bcrypt, Argon2) are used to further increase the cost of brute force attacks.*

4. **256位元雜湊的碰撞攻擊工作量**

   - **答案：** 透過暴力攻擊找到碰撞的預期工作量約為$\mathbf{2^{128}}$次操作。
   - **原因：** 這是由於**生日悖論**。對於n位元雜湊函式，找到碰撞的預期工作量是2^(n/2)。對於256位元雜湊，就是2^(256/2) = 2^{128}。

- **規模感：** 2^{128}是一個巨大的數字（約3.4×10^{38}）。即使使用當今最強大的電腦，執行這麼多次操作也需要比宇宙年齡更長的時間。
- *Answer: The expected work to find a collision by brute force is approximately **2^{128}** operations.*
- *Reason: This is due to the **birthday paradox**. For an n-bit hash function, the expected work to find a collision is 2^(n/2). For 256-bit hash, that's 2^(256/2) = 2^{128}.*
- *Scale perspective: 2^{128} is an enormous number (~3.4×10^{38}). Even with today's most powerful computers, performing this many operations would take longer than the age of the universe.*

Basic Crypto Primitives Message Authentication Code

Message Authentication Codes (MACs) and **Authenticated Encryption** are crucial concepts for ensuring both confidentiality and integrity in secure communications.

中英雙語解釋 / Bilingual Explanation

訊息認證碼 / Message Authentication Code (MAC)

用途： 在具有預先共享秘密/私密金鑰的各方之間用於**訊息認證**。
*Used among parties with a pre-shared secret/private key for **message authentication**.*

形式化定義：
Formal definition:

- 訊息認證碼函式將k位私密金鑰K和變長訊息M映射到n位標籤MACκ(M)
- *A MAC function maps a k-bit private key K and variable-length message M to an n-bit tag MACκ(M)*
- $\text{MAC}: \{0,1\}^k \times \{0,1\}^* \to \{0,1\}^n$
- $\mathit{MAC}: \{0,1\}^k \times \{0,1\}^* \to \{0,1\}^n$

安全要求： 任何不知道金鑰K的攻擊者不能以優於$\sim2^{-n}$的機率正確猜測標籤MACκ(M)。
Any attacker who does not know key K cannot guess the tag MACκ(M) correctly with probability better than $\sim2^{-n}$.

常用標準函式：
Common standard functions:

- **基於對稱區塊加密：** CMAC-AES

- ***Symmetric block-cipher based:*** *CMAC-AES*
- **基於雜湊：** HMAC-MD5, HMAC-SHA-1, HMAC-SHA-2, HMAC-SHA-3
- ***Hash based:*** *HMAC-MD5, HMAC-SHA-1, HMAC-SHA-2, HMAC-SHA-3*

Basic Crypto Primitives Authenticated Encryption

認證加密 / Authenticated Encryption

背景回顧：
Recall that:

- **私密金鑰加密方案**可用於確保**訊息保密性**
- ***Private-key encryption** ensures **message confidentiality***
- **訊息認證碼**可用於提供**訊息完整性**
- ***Message authentication codes** offer **message integrity***

實際需求： 但在實踐中，我們需要在不可信通道（如TLS、IPsec、SSH等）上同時確保兩者。
*But in practice, we need to ensure **both** for communication over untrusted channels.*

重要警告： 需要一些注意來同時提供保密性和完整性。**正確實作這一點出奇地困難**：不要在沒有安全證明的情況下發明自己的方案。
*Getting this right is **surprisingly difficult**: don't invent your own schemes without security proof.*

認證加密方案定義 / Authenticated Encryption Scheme Definition

認證加密方案是一個**私密金鑰加密方案**，它：
An authenticated encryption scheme is a private-key encryption scheme that:

- 提供**密文完整性**
- *Provides **ciphertext integrity***
- 其解密函式將**拒絕**任何不是使用相同私密金鑰K由加密函式生成的密文訊息作為無效
- *Its decryption function will **reject as invalid** any ciphertext not generated by the encryption function using the same private key K*

常用標準AE方案： AES-GCM, AES-CCM, AES-EAX, AES-OCB
Common standard AE schemes: AES-GCM, AES-CCM, AES-EAX, AES-OCB

形式化表示和過程 / Formal Notations and Procedures

實現認證加密函式的一種方法是使用**金鑰派生函式**：
*One way to implement authenticated encryption is to use a **key-derivation function (KDF)**:*

- 從私密金鑰K派生出兩個不同的新金鑰K_e和K_m，分別用於加密函式Enc和訊息認證函式MAC
- *Derive from private key K two different new keys K_e and K_m for use with Enc and MAC*

認證加密函式然後：
The authenticated encryption function then:

1. **首先加密**明文訊息M
    - ***First encrypts** the plaintext message M*
2. **然後附加**密文（也可能覆蓋關聯的明文資料）的**訊息認證碼**
    - ***Then appends** the MAC of the ciphertext*

數學表示：

- $M_K = Enc_{Ke}(M) \,||\, MAC_{Km}(Enc_{Ke}(M))$
- 其中 $K_e \,||\, K_m = KDF(K)$
- $M_K = Enc_{Ke}(M) \,||\, MAC_{Km}(Enc_{Ke}(M)),\ where\ K_e \,||\, K_m = KDF(K)$

詳細工作原理 / Detailed Working Principles

MAC的工作原理 / How MAC Works

傳送方：
訊息M + 共享金鑰K → MAC函式 → 標籤T
傳送：訊息M || 標籤T

接收方：
接收：訊息M' || 標籤T'
訊息M' + 共享金鑰K → MAC函式 → 計算標籤T''
比較 T' 和 T''：
- 如果相等：訊息認證通過
- 如果不相等：訊息被篡改或認證失敗

認證加密的工作原理 / How Authenticated Encryption Works

```
加密端：
明文M + 主金鑰K → KDF → 加密金鑰Kₑ + MAC金鑰Kₘ
    ↓
加密：C = Enc_Kₑ(M)
計算認證標籤：T = MAC_Kₘ(C)
傳送：密文C || 標籤T

解密端：
接收：密文C' || 標籤T'
使用Kₘ驗證：T' = MAC_Kₘ(C')？
    – 如果不等：立即拒絕，不解密
    – 如果相等：使用Kₑ解密：M = Dec_Kₑ(C')
```

實際應用示例 / Practical Application Examples

HMAC結構 / HMAC Structure

HMAC是最常用的MAC構造：
HMAC is the most common MAC construction:

- $HMAC_K(M) = H((K \oplus opad) || H((K \oplus ipad) || M))$
- 其中H是雜湊函式（如SHA-256），opad和ipad是固定的填充值

AES-GCM / AES-GCM

AES-GCM是最流行的認證加密方案：
AES-GCM is the most popular authenticated encryption scheme:

- **加密：** AES計數器模式
- ***Encryption:*** *AES in counter mode*
- **認證：** Galois/計數器模式認證
- ***Authentication:*** *Galois/Counter Mode authentication*
- **優勢：** 平行化、高效、標準化
- ***Advantages:*** *Parallelizable, efficient, standardized*

為什麼認證加密困難？ / Why is Authenticated Encryption Difficult?

常見錯誤模式：
Common error patterns:

1. **加密然後MAC：** $C = Enc_{K_1}(M),\ T = MAC_{K_2}(M)$
   - 問題：暴露明文M給MAC函式
   - *Problem: Exposes plaintext M to MAC function*
2. **MAC然後加密：** $T = MAC_{K_2}(M),\ C = Enc_{K_1}(M \mid\mid T)$
   - 問題：可能遭受填充Oracle攻擊
   - *Problem: May suffer padding oracle attacks*
3. **加密然後MAC錯誤：** $C = Enc_{K_1}(M),\ T = MAC_{K_2}(C)$
   - 但使用相同金鑰：$K_1 = K_2$
   - *But using same key: $K_1 = K_2$*
   - 問題：可能存在金鑰衝突
   - *Problem: Potential key collisions*

正確方法： 使用**不同金鑰**進行加密和MAC，並遵循標準化的AE方案。
Correct approach: *Use **different keys** for encryption and MAC, and follow standardized AE schemes.*

小測驗 / Test

1. MAC和加密雜湊函式在提供完整性方面有什麼主要區別？
   - *What's the main difference between a MAC and a cryptographic hash function in providing integrity?*
2. 為什麼在認證加密中需要使用金鑰派生函式（KDF）來從主金鑰派生出兩個不同的金鑰？
   - *Why do we need a Key Derivation Function (KDF) in authenticated encryption to derive two different keys from the master key?*
3. 在認證加密的"加密然後MAC"方法中，為什麼是對密文計算MAC而不是對明文？
   - *In the "encrypt-then-MAC" approach, why do we compute the MAC on the ciphertext rather than the plaintext?*
4. 如果一個系統使用AES-CBC進行加密，然後使用HMAC-SHA256進行認證，但兩個操作使用相同的金鑰，可能存在什麼安全風險？
   - *If a system uses AES-CBC for encryption and HMAC-SHA256 for authentication, but both operations use the same key, what security risks might exist?*

答案與解析 / Answers and Explanations

1. **MAC vs 加密雜湊函式**
    - **答案：** 主要區別在於**金鑰的使用**。
        - **加密雜湊函式**是**無金鑰的** - 任何人都可以計算和驗證雜湊值。它們提供完整性，但不提供認證（無法確定訊息來源）。
        - **MAC**需要**共享金鑰** - 只有擁有金鑰的各方可以生成和驗證MAC標籤。它們同時提供完整性和認證（證明訊息來自擁有金鑰的傳送方）。
    - *Answer: The main difference is the **use of keys**.*
        - ***Cryptographic hash functions** are **keyless** - anyone can compute and verify the hash. They provide integrity but not authentication (can't determine message origin).*
        - ***MACs** require a **shared key** - only parties with the key can generate and verify MAC tags. They provide both integrity and authentication (proves message came from someone with the key).*

2. **KDF在認證加密中的必要性**
    - **答案：** 使用KDF派生出兩個不同的金鑰是**關鍵的**，因為：
        - **金鑰分離**：加密和認證應該使用**獨立的金鑰**，以防止一個元件的漏洞影響另一個元件。
        - **避免互動攻擊**：如果加密和MAC使用相同金鑰，可能存在密碼學攻擊利用兩者之間的互動。
        - **安全證明要求**：大多數認證加密方案的安全證明依賴於加密和MAC使用獨立金鑰的假設。
    - *Answer: Using a KDF to derive two different keys is **critical** because:*
        - ***Key separation:** Encryption and authentication should use **independent keys** to prevent vulnerabilities in one component affecting the other.*
        - ***Avoid interaction attacks:** If encryption and MAC use the same key, there might be cryptographic attacks exploiting interactions between them.*
        - ***Security proof requirements:** Security proofs for most AE schemes rely on the assumption that encryption and MAC use independent keys.*

3. **對密文計算MAC的原因**
    - **答案：** 對**密文**計算MAC而不是明文有幾個重要原因：
        - **防止選擇性密文攻擊**：如果攻擊者修改密文，MAC驗證會失敗，接收方可以在解密前拒絕惡意密文。

- **更好的安全屬性**：這種方法被證明在更廣泛的攻擊場景下是安全的。
- **避免明文暴露**：MAC函式不需要接觸明文，減少了資訊洩露的風險。
- **標準化**："加密然後MAC"是經過充分研究並被標準（如IPsec）採用的方法。

- *Answer: Computing MAC on the **ciphertext** rather than plaintext has several important reasons:*
  - ***Prevents chosen-ciphertext attacks:** If an attacker modifies the ciphertext, MAC verification fails and the receiver can reject malicious ciphertext before decryption.*
  - ***Better security properties:** This approach has been proven secure against a wider range of attack scenarios.*
  - ***Avoids plaintext exposure:** The MAC function doesn't need to touch the plaintext, reducing risk of information leakage.*
  - ***Standardization:** "Encrypt-then-MAC" is well-studied and adopted by standards like IPsec.*

4. **重複使用相同金鑰的風險**

- **答案：** 在加密和MAC中重複使用相同金鑰存在嚴重安全風險：
  - **金鑰衝突**：可能存在理論攻擊利用AES和HMAC-SHA256中相同金鑰的使用。
  - **違反安全假設**：AES-CBC和HMAC-SHA256的安全證明假設使用獨立金鑰。
  - **潛在互動攻擊**：攻擊者可能能夠利用加密和認證操作之間的意外互動。
  - **削弱整體安全性**：一個元件中的漏洞可能影響另一個元件。
- **正確做法**：總是使用KDF從主金鑰派生出獨立的加密金鑰和MAC金鑰。
- *Answer: Reusing the same key for both encryption and MAC poses serious security risks:*
  - ***Key collision:** There might be theoretical attacks exploiting the use of the same key in both AES and HMAC-SHA256.*
  - ***Violates security assumptions:** Security proofs for AES-CBC and HMAC-SHA256 assume independent keys are used.*
  - ***Potential interaction attacks:** Attackers might exploit unintended interactions between the encryption and authentication operations.*
  - ***Weakens overall security:** Vulnerabilities in one component might affect the other.*
- ***Correct approach:** Always use a KDF to derive independent encryption and MAC keys from the master key.*