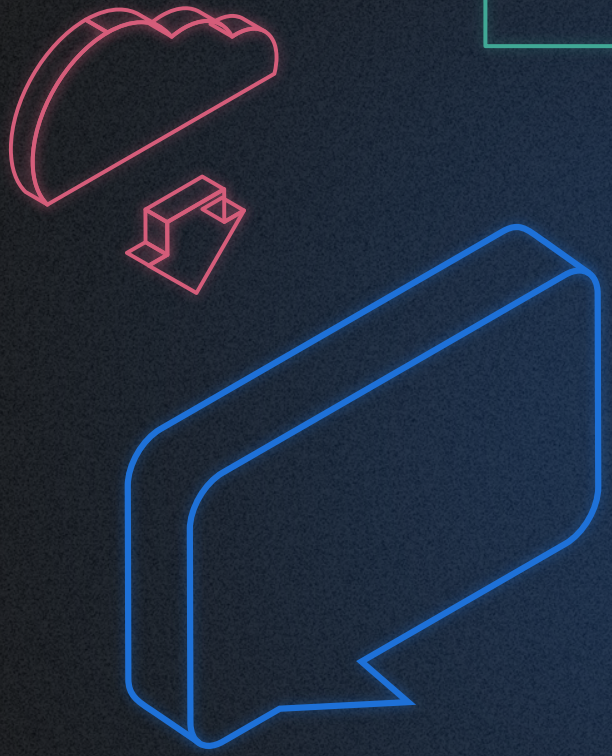
龍華科技大學
LUNGHWA UNIVERSITY
OF SCIENCE AND TECHNOLOGY

密碼學
Cryptography

授課老師：王昱晟 助理教授

報告組別：第一組

報告學生：楊敦傑、楊竣捷、張育丞、徐茂霖、葉俞君



01

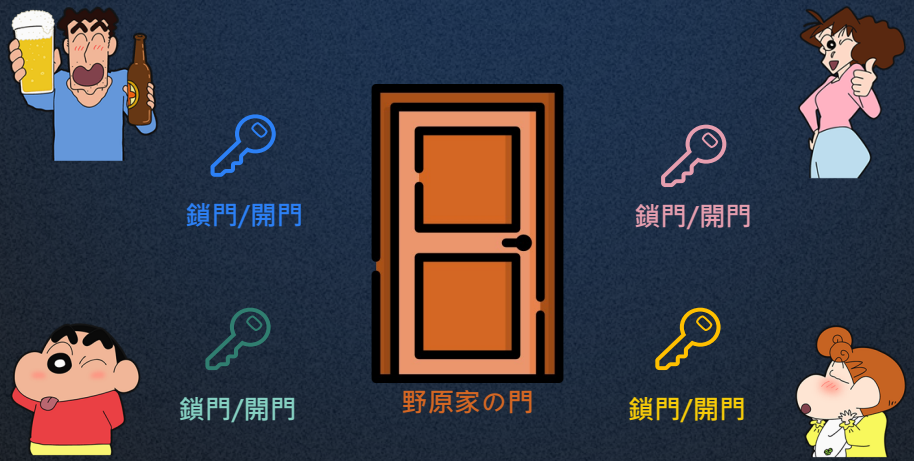
加密法

對稱加密與**非對稱加密**介紹及舉例！

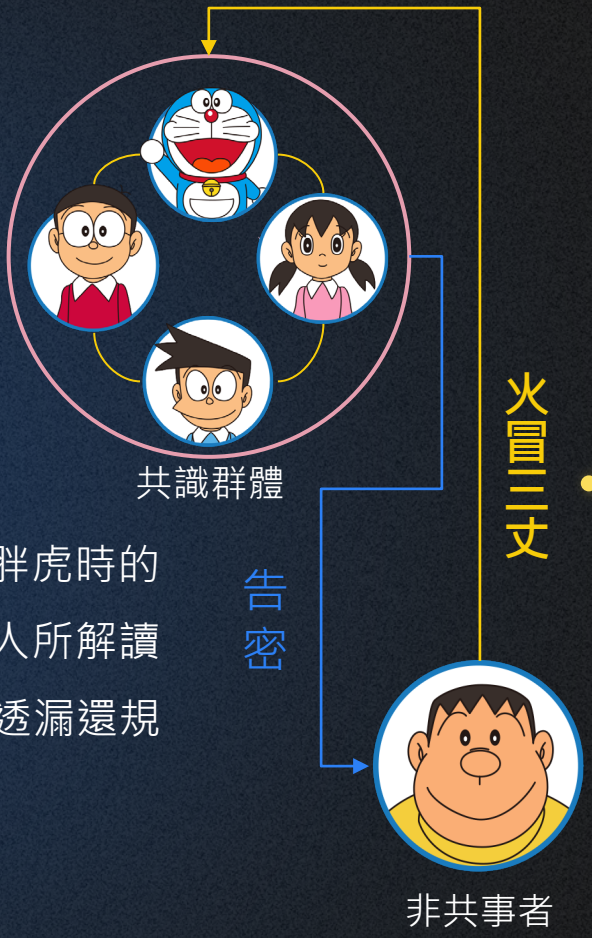
對稱式加密法

Symmetric Encryption

- **定義**：密鑰同時具備加密與解密功能。
- **舉例**：一扇家門，凡是同住者鑰匙皆是相通，具有鎖門及開門的功能。



非對稱式加密法
Asymmetric Cryptography



- **定義**：具有私密及公開的鑰匙，且鑰匙互不相通。
- **舉例**：由小叮噹為首的四人組(沒胖虎)討論關於謾罵胖虎時的密語規則(公鑰)，並在每次謾罵胖虎時將使用，在旁人所解讀為莫名奇妙的對話，其實暗潮洶湧，需透過內鬼小夫透漏還規則(私鑰)，否則非圈內人無法輕易了解。
- 補充：群體商討的加密策略稱之為「共識演算法」。

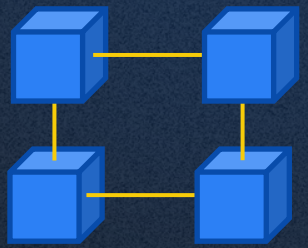
區塊與串流加密法

Block & Stream Encryption

區塊加密法

將明文分成多個等長的模組（block），使用確定的演算法和對稱金鑰對每組分別加密解密。

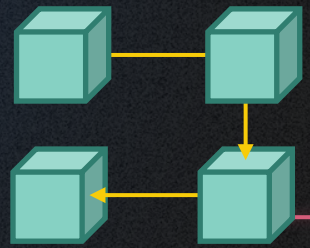
如：DES、AES。



串流加密法

對明文一個個位元加密，猶如流水般進入加密器中。具加密快速的特性，故有即時效果。

如：XOR Cipher、RC4。



優缺評判

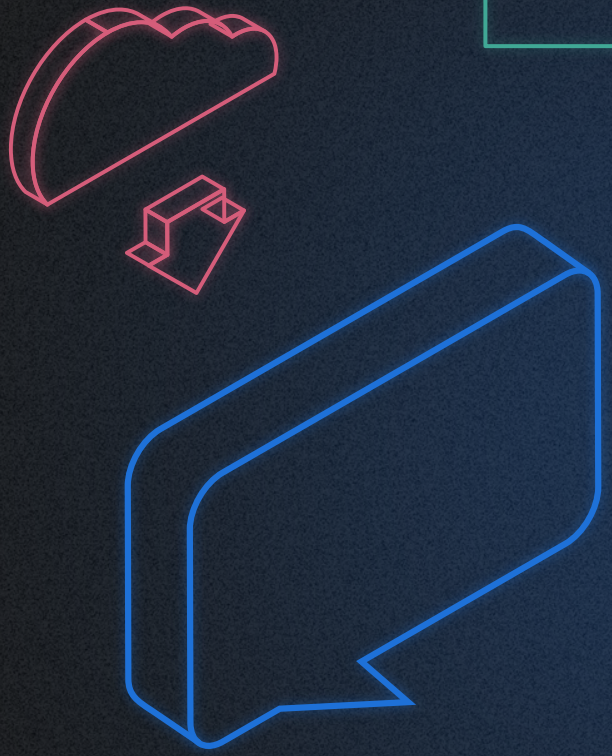
Advantages and Disadvantages Assessment

優 點

- 對稱加密法相較於非對稱擁有
- 高的安全級別。
- 能夠快速加密與解密。
- 消耗更少的運算資源。

缺 點

- 加解密都使用同個密鑰。
- 在不安全的網域當中容易被攔截密鑰。



02

程式實作

實際程式設計並體驗**對稱加密**及**解密**！

程式實作 (介面設計)
Program Implementation (User Interface Design)

操作介面圖

程式操作網頁

QR CODE

程式實作 (加密)
Program Implementation (Encryption)

A ☞ 0100 0001

Private ☞ 1111 0011

Result ☞ 1011 0010

輸入		輸入
A	B	C
0	0	0
0	1	1
1	0	1
1	1	0

```
function handleClick() {
        // 取得加密/解密文字與金鑰
        const text = document.getElementById('text').value;
        const key = document.getElementById('key').value;
        const mode = document.getElementById('mode').value;

        if (mode === 'encrypt') {
            // 將文字轉換為二進制
            let binaryText = "";
            for (let i = 0; i < text.length; i++) {
                let charCode = text.charCodeAt(i);
                let binaryChar = charCode.toString(2).padStart(8, "0");
                binaryText += binaryChar;
            }

            // 使用金鑰進行 XOR 加密
            let encryptedText = "";
            for (let i = 0; i < binaryText.length; i++) {
                let encryptedChar = (binaryText[i] ^ key[i %
key.length]).toString();
                encryptedText += encryptedChar;

            }

            // 將加密結果輸出到結果欄位
            document.getElementById('result').value = encryptedText;
        } else if (mode === "decrypt") {
            // 將二進制文字分割成每 8 個一組，並轉換為十進制
            let decimalText = [];
            for (let i = 0; i < text.length; i += 8) {
                let binaryChar = text.substring(i, i + 8);
                let decimalChar = parseInt(binaryChar, 2);
                decimalText.push(decimalChar);
            }

            // 使用金鑰進行 XOR 解密，並將結果轉換為文字
            let decryptedText = "";
            for (let i = 0; i < decimalText.length; i++) {
                let decryptedChar = (decimalText[i] ^ parseInt(key, 2));
                decryptedText += String.fromCharCode(decryptedChar);
            }

            // 將解密結果輸出到結果欄位
            document.getElementById('result').value = decryptedText;
        }
    }
```

程式實作 (解密)
Program Implementation (Decryption)

A ☞ 1011 0010

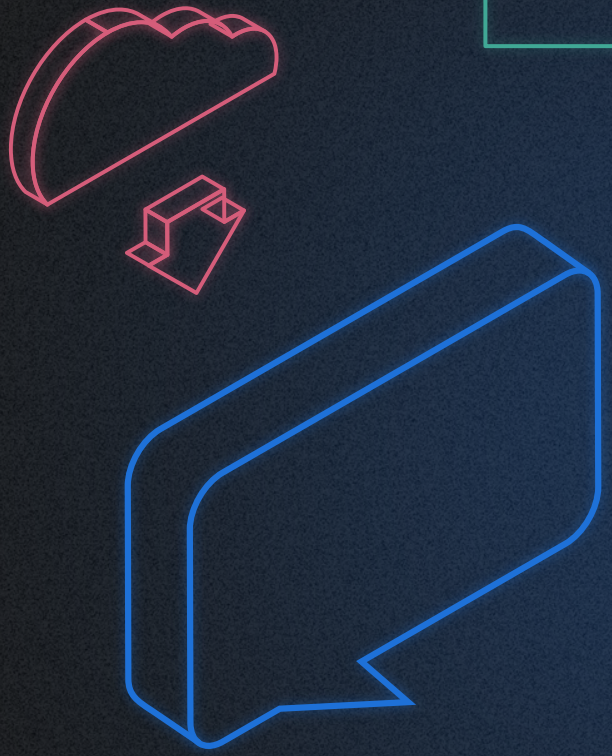
Private ☞ 1111 0011

Result ☞ 0100 0001

輸入		輸入
A	B	C
0	0	0
0	1	1
1	0	1
1	1	0

```
function handleClick() {
        // 取得加密/解密文字與金鑰
        const text = document.getElementById('text').value;
        const key = document.getElementById('key').value;
        const mode = document.getElementById('mode').value;

        if (mode === 'encrypt') {
            // 將文字轉換為二進制
            let binaryText = "";
            for (let i = 0; i < text.length; i++) {
                let charCode = text.charCodeAt(i);
                let binaryChar = charCode.toString(2).padStart(8, "0");
                binaryText += binaryChar;
            }

            // 使用金鑰進行 XOR 加密
            let encryptedText = "";
            for (let i = 0; i < binaryText.length; i++) {
                let encryptedChar = (binaryText[i] ^ key[i %
key.length]).toString();
                encryptedText += encryptedChar;

            }

            // 將加密結果輸出到結果欄位
            document.getElementById('result').value = encryptedText;
        } else if (mode === "decrypt") {
            // 將二進制文字分割成每 8 個一組，並轉換為十進制
            let decimalText = [];
            for (let i = 0; i < text.length; i += 8) {
                let binaryChar = text.substring(i, i + 8);
                let decimalChar = parseInt(binaryChar, 2);
                decimalText.push(decimalChar);
            }

            // 使用金鑰進行 XOR 解密，並將結果轉換為文字
            let decryptedText = "";
            for (let i = 0; i < decimalText.length; i++) {
                let decryptedChar = (decimalText[i] ^ parseInt(key, 2));
                decryptedText += String.fromCharCode(decryptedChar);
            }

            // 將解密結果輸出到結果欄位
            document.getElementById('result').value = decryptedText;
        }
    }
```

程式碼圖



03

破解案例

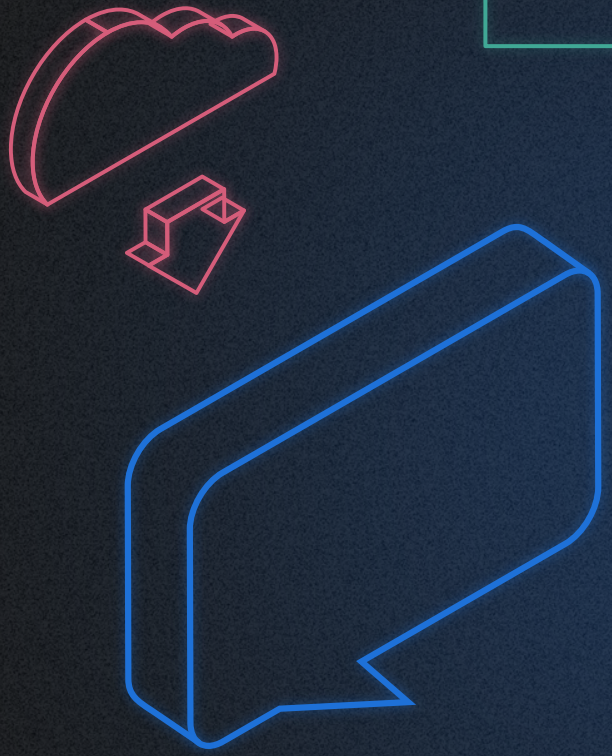
洞悉弱點進行攻擊，才能夠了解哪裡不足！

側通道攻擊
Side Channel Attack

側通道攻擊（SCA），又稱旁路攻擊，是指通過竊取電子設備實現加解密演算法時洩露的旁路資訊從而攻擊密碼系統的方法，例如通過分析密碼系統的計算時間、功率消耗、電磁輻射和聲音情況等“聽譯”密鑰破解密碼。 SCA並不直接攻擊已證明運算安全的密碼演算法本身，而採取了一種“曲線救國”的方式攻擊密碼演算法的實現技術，破解密碼系統的效果更為顯著。

側通道攻擊舉例
Example of a Side-Channel Attack

例如想知道一個人在家中是在洗衣服還是看電視，我們並不需要撬開門鎖進入屋子一探究竟，直接通過觀察室外的水錶和電錶便可推測出來，SCA正是這樣一種繞開門鎖的窺探方法。



04

防護手段

在了解如何攻擊後才能夠針對弱點有效防禦！

防護手段: VPN加密

透過高安全性的VPN協議加密以達到防護手段！

VPN是什麼?

VPN是保護使用者的資料不易外洩的一款軟體！

VPN通常所具備以下三點功能

個人隱私保護

加密網絡數據流量，使在網絡上的活動變得私密和安全。

防止數據竊取

加密網絡流量，使其更難以被駭客竊取。以達到保護個人敏感信息如銀行資料。

匿名上網

隱藏真實IP地址，使網絡上的活動難以被追蹤。避免受廣告追蹤、駭客攻擊和網絡監控等。

VPN加密及運作方式

Step.1 非對稱密鑰交換

用非對稱密鑰交換相互握手，取得資料創建公鑰和私鑰，使用者使用提供的公鑰，用只有使用者可以解密的方式來加密數據。

Step.2 對稱密鑰交換

創建一個新對稱密鑰，加密算法會使用此密鑰來傳輸實際數據。

Step.3 加密算法

加密算法會使用上述產生的對稱密鑰來加密所有數據。

Step.4 完整性算法

目的是確保通訊是否被篡改，利用數學哈希函數(Hash)來擾亂所傳送的部分訊息，接收方檢查此數據和私鑰做比對，若可以匹配則表示未受到干擾。

VPN協議

安全性取決於所使用的VPN協定

01 OpenVPN

高安全性、速度快、較廣泛使用。

SurfsharkVPN / NordVPN皆使用此協議

02 IPSec/IKEv2

提供加密或身份驗證、穩定性高、速度快以及不錯的安全性。穩定性高使得切換網路不會導致VPN斷掉。

03 WireGuard

速度極快、穩定性高，因代碼精簡使得有更多資源可以投入運算來達到高速。

04 SSTP

用於連線Windows設備。現今可支援其他作業系統，因程式只有內部人員可以查看，為避免偷開後門，絕大多數VPN不採用此協議。

05 L2TP/IPSec

不具備加密或身份驗證，支援大部分作業系統。

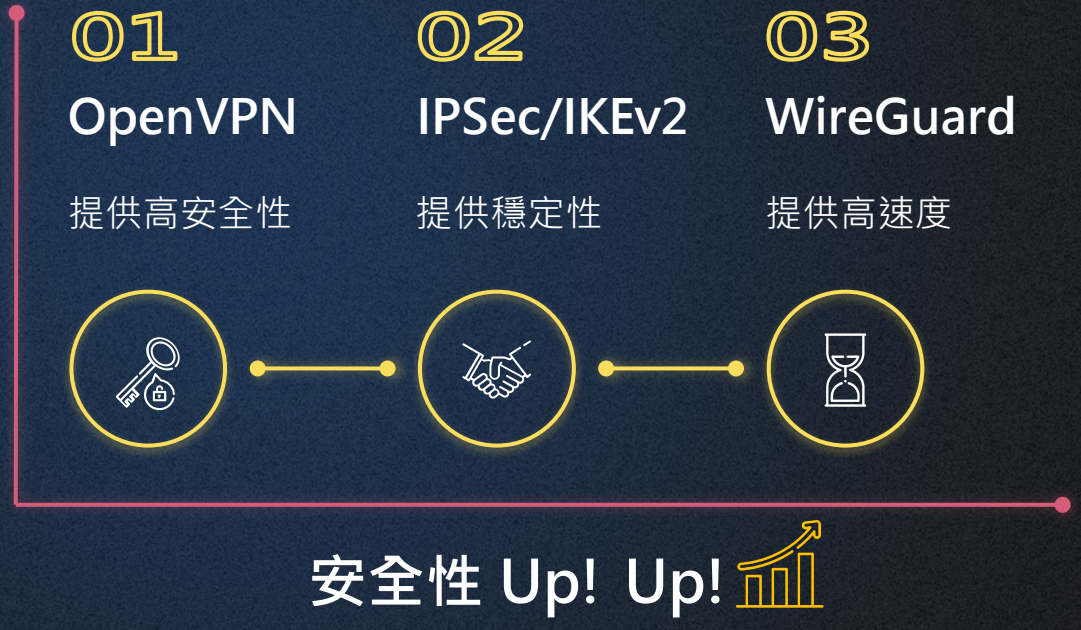
06 PPTP

第一個VPN協議，此協議在如今已經被證實協議過時老舊，有許多漏洞。

VPN多重協議加密

透過多重協議加密

知名VPN如 SurfsharkVPN、NordVPN皆使用多個協議加密來達到高安全性，這些協議通常皆使用高強度的AES-256算法進行加密。




龍華科技大學
LUNGHWA UNIVERSITY
OF SCIENCE AND TECHNOLOGY
密碼學
Cryptography
授課老師：王昱晟 助理教授
報告組別：第一組
報告學生：楊敦傑、楊竣捷、張育丞、徐茂霖、葉俞君
龍華科技大學．資訊網路工程系暨研究所
Department of Computer Information and Network Engineering

THANKS!