分組名單(不足5個人空著就好):

姓名	學號
楊子睬	111550076
林悅楊	111550149
陳冠智	111550086
曾紹幃	111550040
林鈺誠	111550102

1. Name of the paper: New Directions in Cryptography

2. Summary:

這篇論文由Whitfield Diffie 和 Martin E. Hellman 共同撰寫, 它開啟了現代加密技術的新篇章。隨著電子通訊日益普及, 對安全加密技術的需求也不斷增長。這篇文章提出了兩個創新的概念:公鑰密碼系統和數字簽名。公鑰系統特點在於使用兩把鑰匙, 一把公開的公鑰用於加密, 一把私密的私鑰用於解密, 使得即使公鑰被人知曉, 通信的安全性仍然得到保障。數字簽名則為數字文件提供了一種確認真實性的方式, 這對於網路交易和法律文件來說至關重要, 因為它們需要在沒有實體文檔的情況下證明合法性。

3. Strength(s) of the paper:

(1). 創新性:這篇論文提出了公鑰密碼系統的概念, 徹底改變了密碼學的傳統模式, 提供了一種不需事先共享秘密鑰匙的新方法, 這在當時是一個革命性的突破。該論文中的理論創新為後續的研究提供了無數可能性, 使得信息安全技術得以快速發展。

(2). 應用廣泛:論文中介紹的技術在現代信息安全中得到了廣泛應用, 影響了數字簽名、SSL/TLS協議、電子商務等多個領域。公鑰密碼學的概念不僅在學術研究中得到了驗證, 也在實際應用中取得了巨大成功, 成為現代通信安全的基石。

(3). 理論基礎扎實:該研究在信息理論和計算理論的基礎上建立, 為後續的研究提供了堅實的理論框架。論文中對密碼學問題的深入分析和對新方法的數學證明, 為公鑰密碼系統的可信度提供了有力支持, 啟發了後續大量的研究工作。

(4), 解決實際問題:論文中提出的方法有效解決了密鑰分配問題, 這在實際應用中大大提升了通信的安全性和便捷性。通過公鑰加密, 消除了傳統密鑰分發中的許多困難, 實現了無需安全通道即可進行安全通信的目標, 大幅度提升了系統的實用性。

(5).影響深遠:該論文不僅在學術界有著深遠的影響, 也對工業界產生了巨大的影響, 推動了信息安全技術的發展和普及。它促進了數字時代的到來, 使得安全通信、數字簽名和電子交易成為可能, 徹底改變了我們的生活和工作方式。

4. Weakness(es) of the paper

(1). 實現複雜度:論文中提出的方法在實際實現中存在一定的複雜度, 尤其是對於當時的計算機硬體來說, 公鑰加密和解密的計算成本較高。這使得早期的實施變得困難, 並且在某些應用中可能受到計算資源的限制。

(2). 依賴數學難題:安全性依賴於某些數學難題(如大數因數分解和離散對數問題)的計算困難度。如果未來數學界或計算機技術有重大突破, 這些基於難題的系統可能面臨安全風險。例如, 量子計算的發展可能會威脅到當前的公鑰密碼系統。

(3). 效率問題:相比於傳統的對稱加密系統, 公鑰系統的運行效率較低, 這在某些應用場景中可能成為瓶頸。尤其在需要大量加密解密操作的情況下, 性能問題更加突出, 可能影響系統的整體效率和用戶體驗。

(4). 密鑰管理:雖然解決了密鑰分發的問題, 但公鑰基礎設施(PKI)的管理和維護仍然是一個複雜且昂貴的問題。尤其是在大規模應用中, 如何安全地管理和發佈公鑰, 如何防止公鑰目錄被篡改等問題仍需要妥善解決, 增加了實施的複雜性。

(5). 初期接受度:由於技術的創新性和複雜性, 在早期推廣過程中可能會遇到來自傳統方法支持者的質疑和抵制, 增加了技術普及的難度。一些組織和個人可能對新技術持懷疑態度, 認為其尚未經過充分驗證, 這在一定程度上延緩了公鑰密碼系統的廣泛應用。

5. Your own reflection, which can include but not limited to:

A. *What did you learn from this paper?*
從這篇論文中, 我學到了公鑰和私鑰在現代加密技術中的重要性和基

本應用，尤其是如何透過這些技術來保護數據安全與隱私。公鑰加密技術解決了傳統對稱加密中密鑰分發的問題，大大提升了通信的便捷性和安全性。此外，數字簽名的概念讓我理解到在數字化世界中，如何利用加密技術來確保文件的真實性和不可否認性，這對於進行數位交易和法律合約尤為關鍵。這些知識不僅加深了我對加密技術的理解，也為我後續在相關領域的學習和研究奠定了堅實的基礎。

B. *How would you improve or extend the work if you were the author?*

1. 先進的公鑰密碼系統

新算法：

引入新技術：介紹和分析自原出版以來出現的新型公鑰密碼算法，例如橢圓曲線密碼學（*ECC*）和後量子密碼學（*PQC*），並比較其安全性和效率。

加強安全性證明：提供更嚴格的安全性證明，使用現代密碼學理論中的新工具，如形式化方法來證明公鑰系統的安全性。

2. 應用和使用案例

現實應用：

案例研究：提供詳細的案例研究，展示公鑰密碼學在現代技術中的應用，例如區塊鏈技術、物聯網（*IoT*）安全和數字簽名。

跨學科整合：探索公鑰密碼學如何與量子計算、機器學習和其他新興技術相結合，以提高安全性和效率。

3. 密碼協議

增強協議：

改進現有協議：提出改進現有密碼協議的方法，解決潛在的漏洞並提高效率，例如基於零知識證明（*ZKP*）的協議和同態加密技術。

協議標準化：討論標準化密碼協議的重要性和國際組織在這一過程中的作用，並強調成功的標準化案例和經驗教訓。

4. 未來方向

抗量子加密：

量子計算影響研究：研究量子計算對現有加密系統的影響，並提出抗量子計算的加密替代方案，例如後量子密碼學算法。

隱私增強技術：研究可以與公鑰密碼學共同使用的新隱私增強技術，提供更強的用戶隱私和數據保護保障，例如差分隱私和多方計算（*MPC*）。

C. *What are the unsolved questions that you want to investigate?*

1. 公鑰密碼系統的開發：
   雖然作者提出了公鑰密碼系統的概念，但具體實現這種系統的方法仍然是開放性問題。他們建議需要進一步研究如何開發既能有效加密又能確保解密安全的公鑰密碼系統。

2. 數位簽名的實現:
論文提出了數位簽名的必要性, 但如何在實際系統中高效地實現數位簽名, 並確保其安全性和可靠性, 仍然需要深入研究。
3. 公鑰分發系統的改進:
儘管提出了一些公鑰分發的方法, 但這些方法的實際效率和安全性有待提高。尤其是如何在不犧牲安全性的前提下減少通信和計算成本, 是一個亟待解決的問題。
4. 計算複雜性理論在密碼學中的應用:
論文強調了計算複雜性在保證密碼系統安全性中的重要作用, 但具體哪些複雜性問題可以用來構建更安全的密碼系統, 以及如何證明這些問題在計算上是困難的, 仍需要更多的研究。
5. 認證和防偽機制的完善:
論文討論了在數位通信中認證和防偽的重要性, 但如何開發更高效、更安全的認證機制, 特別是在多使用者網路環境下, 是一個需要深入研究的問題。

D. *What are the broader impacts of this proposed technology?*
這篇論文提出的公鑰密碼加密與數位簽章系統, 在現今社會中都已經是相當成熟的技術, 並且已經有許多優化與改良應用在現實生活中, 但我們還是能夠觀察到這些技術為我們帶來的影響, 以及他在未來可能帶來的改變:
1. 金融產業:促成了網路交易以及電子商務的崛起, 並且數位簽章也推動了電子合約等等相關業務的實踐。
2. 政府法律:公私鑰加密影響了隱私權相關法律的制定, 在證據採樣時強制執行的法規更新, 以及在軍事相關加密通訊設備的進步。
3. 日常通訊:在各式各樣的通訊軟體上皆可採用簡單的加密就能保障通訊安全, 縮短現代人之間的距離。
這些技術延伸到現在也發展出許多密碼學的領域:
數位簽章在現今延伸成為Zero-knowledge proof的研究, 並且推動了區塊鏈的崛起, 再延伸到同態加密, 或是加密貨幣相關的應用, 未來也會在物聯網等等相技術上看到他的蹤影。
公鑰加密在現今則延伸至例如RSA等等加密方式, 以及促進量子電腦與量子加密相關的研究, 在未來不論是在雲端運算, 或是5G網路相關應用, 都可以見到公鑰加密的蹤影。

6. Realization of a technical specification or algorithm as a program:
這段code演示了包括Diffie-Hellman密鑰交換和RSA加密/解密在內的安全通信技術。這些實驗基於Whitfield Diffie和Martin E. Hellman的論文《密碼學的新方向》中提出的概念。

github:https://github.com/David810209/NYCU_cryptography_engineering/tree/main/critique3

- choose_large_prime_and_base(): Selects a large prime and a base for Diffie-Hellman key exchange.

```python
# 選擇一個大質數p和基數g
def choose_large_prime_and_base():
    primes = list(primerange(1000, 5000))
    p = random.choice(primes)
    g = random.randint(2, p-1)
    return p, g
```

- diffie_hellman_key_exchange(): 執行Diffie-Hellman密鑰交換並返回共享秘密和密鑰。

```python
# Diffie-Hellman Key Exchange Function
def diffie_hellman_key_exchange():
    p, g = choose_large_prime_and_base()
    a = random.randint(1, p-1)
    b = random.randint(1, p-1)
    alice_public_key = pow(g, a, p)
    bob_public_key = pow(g, b, p)
    # 交換並計算共享秘密
    alice_secret = pow(bob_public_key, a, p)
    bob_secret = pow(alice_public_key, b, p)

    print(f"Alice 的公鑰: {alice_public_key}")
    print(f"Bob 的公鑰: {bob_public_key}")
    print(f"Alice 的秘密: {alice_secret}")
    print(f"Bob 的秘密: {bob_secret}")

    assert alice_secret == bob_secret, "秘密不匹配，有錯誤發生！"
    dh_secret = alice_secret
    return dh_secret, alice_public_key, bob_public_key, p, g, a, b
```

- compute_shared_secret(public_key, private_key, p): 使用另一方的公鑰計算共享秘密。

```python
# Compute shared secret using another party's public key
def compute_shared_secret(public_key, private_key, p):
    return pow(public_key, private_key, p)
```

- generate_rsa_keys(): 生成RSA公鑰和私鑰。

```python
# RSA Key Generation
def generate_rsa_keys():
    private_key = rsa.generate_private_key(
        public_exponent=65537,
        key_size=2048,
        backend=default_backend()
    )
    public_key = private_key.public_key()
    return private_key, public_key
```

- rsa_encrypt(message, public_key): 使用RSA公鑰加密消息。

```python
# RSA Encryption
def rsa_encrypt(message, public_key):
    encrypted = public_key.encrypt(
        message.encode(),
        padding.OAEP(
            mgf=padding.MGF1(algorithm=hashes.SHA256()),
            algorithm=hashes.SHA256(),
            label=None
        )
    )
    return encrypted
```

- rsa_decrypt(encrypted, private_key): 使用RSA私鑰解密消息。

```python
# RSA Decryption
def rsa_decrypt(encrypted, private_key):
    original_message = private_key.decrypt(
        encrypted,
        padding.OAEP(
            mgf=padding.MGF1(algorithm=hashes.SHA256()),
            algorithm=hashes.SHA256(),
            label=None
        )
    )
    return original_message.decode()
```

- sign_message(message, private_key): 使用RSA私鑰簽署消息。

```python
# RSA Sign Message
def sign_message(message, private_key):
    signature = private_key.sign(
        message.encode(),
        padding.PSS(
            mgf=padding.MGF1(hashes.SHA256()),
            salt_length=padding.PSS.MAX_LENGTH
        ),
        hashes.SHA256()
    )
    return signature
```

- verify_signature(message, signature, public_key): 使用RSA公鑰驗證簽名。

```python
# Verify Signature
def verify_signature(message, signature, public_key):
    public_key.verify(
        signature,
        message.encode(),
        padding.PSS(
            mgf=padding.MGF1(hashes.SHA256()),
            salt_length=padding.PSS.MAX_LENGTH
        ),
        hashes.SHA256()
    )
    return True
```

- main()測試上面的function的功能，可以透過交換密鑰獲取一樣的內容

```python
def main():
    print("Starting secure communication simulation...")

    # Diffie-Hellman Key Exchange
    dh_secret, alice_public_key, bob_public_key, p, g, a, b = diffie_hellman_key_exchange()
    print(f"Diffie-Hellman Shared Secret: {dh_secret}")

    # Compute shared secret using Bob's public key and Alice's private key
    alice_computed_secret = compute_shared_secret(bob_public_key, a, p)
    print(f"Recomputed Shared Secret using Bob's public key: {alice_computed_secret}")

    # Compute shared secret using Alice's public key and Bob's private key
    bob_computed_secret = compute_shared_secret(alice_public_key, b, p)
    print(f"Recomputed Shared Secret using Alice's public key: {bob_computed_secret}")

    # RSA Encryption/Decryption
    private_key, public_key = generate_rsa_keys()
    message = input("Enter a message to encrypt: ")
    encrypted_message = rsa_encrypt(message, public_key)
    decrypted_message = rsa_decrypt(encrypted_message, private_key)
    print("Encrypted Message:", encrypted_message)
    print("Decrypted Message:", decrypted_message)

    # Digital Signature
    signature = sign_message(message, private_key)
    verification = verify_signature(message, signature, public_key)
    print("Signature Verified:", verification)
```

- 結果顯示，確實可以用上述function達到算出共同的secret的功能，在RSA加密解密跟digital signature功能也能實現。