|  |
| --- |
|  |
| **Zero-Knowledge Proof：以 Schnorr Protocol 為核心的可驗證身份驗證系統** |
| 日期：114年6月5日  課程名稱：密碼學  系所：電機工程研究所  組別：資訊安全組  組員：r13921a13 林伯叡  d13921c02黃杬霆 |

****

**目錄**

[**前言** 1](#_Toc198837745)

[**一、** **背景介紹與動機說明** 2](#_Toc198837746)

[(一) **背景介紹** 2](#_Toc198837747)

[(二) **動機說明** 3](#_Toc198837748)

[**二、** **專題目標** 4](#_Toc198837749)

[(一) **傳統ZKP（互動式驗證機制）** 4](#_Toc198837750)

[(二) **精進版Schnorr Protocol（簡化版互動式驗證機制）** 4](#_Toc198837751)

[(三) **Fiat–Shamir Heuristic（非互動式驗證機制）** 5](#_Toc198837752)

[**三、** **系統設計** 6](#_Toc198837753)

[(一) **傳統ZKP — 以Graph 3-colorability（簡稱G3C）為例子** 6](#_Toc198837754)

[(二) **精進版Schnorr Protocol** 8](#_Toc198837755)

[(三) **Fiat–Shamir Heuristic（待研擬）** 10](#_Toc198837756)

[**四、** **專題成果** 11](#_Toc198837757)

[(一) **傳統ZKP — 以Graph 3-colorability（簡稱G3C）為例子** 11](#_Toc198837758)

[(二) **精進版Schnorr Protocol** 13](#_Toc198837759)

[(三) **Fiat–Shamir Heuristic（待研擬）** 14](#_Toc198837760)

[**五、** **結論（待研擬）** 15](#_Toc198837761)

[**六、** **未來展望（待研擬）** 16](#_Toc198837762)

[**七、** **參考文獻** 17](#_Toc198837763)

[**八、** **附錄** 18](#_Toc198837764)

[(一) **傳統ZKP — 以Graph 3-colorability（簡稱G3C）為例子** 18](#_Toc198837765)

[(二) **精進版Schnorr Protocol** 26](#_Toc198837766)

[(三) **Fiat–Shamir Heuristic（待研擬）** 31](#_Toc198837767)

**前言**

在資訊安全日益受到重視的當下，強調的安全性標準為「**零日攻擊**（zero - day）」，如何在遭遇未知攻擊，透過不洩漏關鍵個人資訊的前提下，完成身分驗證與機密確認，已成為密碼學中的重要課題。

傳統的驗證機制仰賴密碼、個人資訊、生物特徵或私鑰的直接揭露，然而這類方法存在極高的風險因子，可能導致機密外洩、遭遇重放攻擊，或被第三方竊取…等。為了解決此問題，「**零知識證明**（**Zero-Knowledge Proof**, **ZKP**）」技術於1985年應運而生，提供了一種創新方式，允許「證明者」（Prover）在不洩漏秘密的情況下，說服「驗證者」（Verifier）其確實擁有該秘密。

本專題以 ZKP 為核心出發，從傳統互動式的驗證模型著手研究，實作了以「**Graph 3-colorability**（**G3C**）」為基礎的零知識協定，觀察其完整性、可靠性與零知識性。接著，我們進一步導入**精進版Schnorr** **Protocol**，以**離散對數問題**為數學基礎，設計一套簡化版的互動式 ZKP，在保有原本ZKP安全性之下，同時大幅降低訊息傳輸成本，並精進了**原Schnorr Protocol** **[2]**所不足之處。最終因應資訊時代的資安需求，我們導入 **Fiat–Shamir Heuristic [3]**，實現「**非互動式**」的零知識證明模型，進一步提升其在真實應用場景中的彈性與可行性，亦為區塊鏈、加密貨幣等無法即時互動的場景，奠定了可行且安全的實作基礎。

1. **背景介紹與動機說明**
2. **背景介紹**

在早期的驗證系統中欲達成登入驗證，通常只能透過使用端直接揭露部分或全部資訊來讓目標端驗證，例如：輸入密碼、提交文件、或洩露原始資料…等方式，雖能達到驗證目的，但容易造成機密洩漏、重放攻擊，或讓未經授權的第三方竊取重要資訊。在高度依賴網路傳輸與重視資訊安全的現代，這類型作法已難以滿足對安全性及隱私性的雙重需求。為了解決這些問題，密碼學家 Shafi Goldwasser、Silvio Micali 與 Charles Rackoff 在 1985 年提出了**零知識證明**（**Zero-Knowledge Proof**， **ZKP**）的概念 [1]。他們設計了一種互動式的證明方法，讓「證明者」能在不揭露秘密本身的情況下，說服「驗證者」其確實知道某個秘密。這項技術徹底顛覆了傳統「揭露才能驗證」的模式，為密碼學、身分驗證與區塊鏈等未來領域帶來了全新的解決方案，以下為相關數學理論基礎及基本性質：

* + - * 數學理論基礎：
    1. 選用的一個數學難題：以Schnorr Protocol為例，其採數學上難解的問題—**離散對數問題**（**Discrete Logarithm Problem**，**DLP**）作為整個方法的核心基礎，駭客若想入侵則必須破解該議題。
    2. 數學背景：在一個有限域中 ，給定一個生成元（generator） 及任意元素 ，欲找到一個，使得：，則稱為相對於 的離散對數**（）**。
       - 基本性質：

1. **可靠性**（**Soundness**）：若陳述為假，不存在一個可證明策略，使證明者（Prover）說服驗證者（Verifier）相信。

例如：假設有人提出證明：「1 + 1 = 3」，證明系統應要求這種錯誤的證明不該存在。

1. **完整性**（**Completeness**）：若陳述為真，證明者（Prover）應該能找到一個可證明策略使驗證者（Verifier）相信。

例如：如果「1 + 1 = 2」是真的，那證明系統應該可以證明，而非找不到證明。

1. **零知識性**（**Zero-Knowledge**）：證明者（Prover）欲向驗證者（Verifier）證明知道「秘密（Security）」，必須在不直接揭露該秘密的情況下，僅能透過一些「陳述（Statement）」的交換，讓驗證者相信其知道該秘密，例如：證明者可以向驗證者證明自己知道密碼，而不需要直接告訴驗證者密碼。
2. **動機說明**

依據前述傳統驗證機制（無論是密碼式或生物特徵式）普遍仰賴資訊的直接揭露，導致使用者在追求安全驗證的同時，卻又必須承擔機密洩漏、重放攻擊、甚至嚴重的隱私外洩風險…等，此外，ZKP所提供的「不揭露秘密亦能驗證」，正好回應了當今資訊安全與隱私保護的雙重需求。本專題正是基於這樣的基礎，進一步探究 ZKP 的相關原理及成效。

1. **專題目標**

本專題旨在循序探討ZKP技術的發展脈絡，從基本架構出發，逐步導入實作效率與交互性的改善技術，藉由改善互動傳輸成本，最終達成兼顧安全性與可實用性的「非互動式驗證機制」，目標分為以下三階段：

1. **傳統ZKP（互動式驗證機制）**

初期的零知識證明為互動式機制，由證明者與驗證者透過多次隨機挑戰與回應的方式達成驗證。在理論上這種架構能確保完整性、可靠性與零知識性；然而，在實務應用上此類 ZKP 通常需多輪互動、傳輸成本高，不利於一對多、非同步或網路不穩定的環境中。為解決該問題，本專題進一步探討如何降低相關傳輸所造成的不穩定。

1. **精進版Schnorr Protocol（簡化版互動式驗證機制）**

原版Schnorr在結構上延續傳統互動式 ZKP 的安全性設計，但引入更簡潔的數學結構：**承諾（commit）**、**挑戰（challenge）**及**回應（response）**，大幅降低計算與通訊成本，並具有實作上的簡便性，並使其成為多種密碼學應用（如電子簽章）中的基礎原型。**本專題為解決挑戰值*c*存在著亂數長度過長影響傳輸效能及值可能相同的問題**，採用：**承諾值f + 時間戳記 + 回合編號 +Salt**，再丟入**SHA256**。然而，因Schnorr 協定仍屬於互動式協定，需證明者與驗證者同時在線互動，這在分散式系統或無法同步傳輸的場景中仍有侷限。故我們接續探討如何將其轉化為非互動式形式。

1. **Fiat–Shamir Heuristic（非互動式驗證機制）**

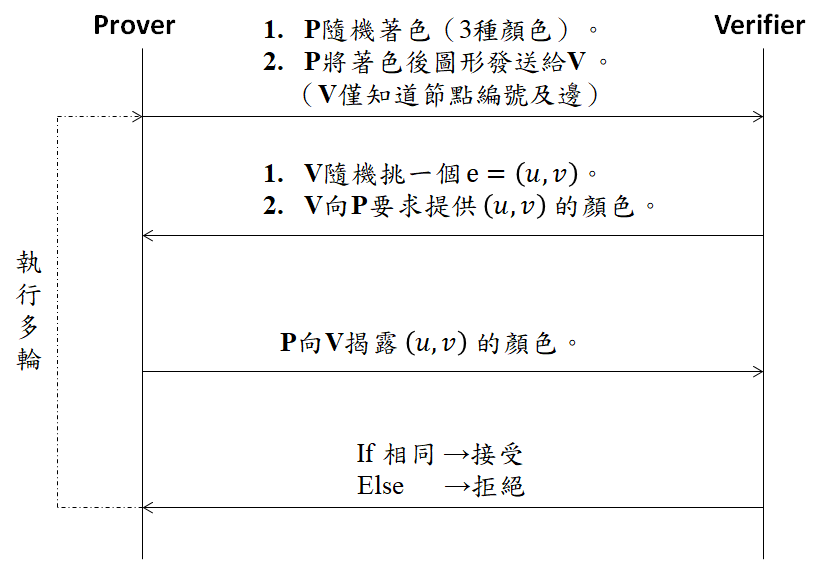
Fiat–Shamir 方法利用 hash 函數將互動式協定中的隨機挑戰值，改為由證明者本地端自行計算產生，實現 ZKP 的「非互動化」。它將原本需由驗證者提出的挑戰，改為自行產生 **hash** 值： ，消除雙方互動需求。

上述變更成功解決 Schnorr 協定的互動性限制，使得零知識證明可應用於區塊鏈（需支援**非互動式證明**的去中心化架構）、電子身份（如一次性提交的**非互動式驗證**流程）、無人監督驗證等場景，這些應用場域皆對**非互動性有高度依賴**，因此Fiat–Shamir 方法成為現代 ZKP 實作中的關鍵技術。

1. **系統設計**
2. **傳統ZKP — 以Graph 3-colorability（簡稱G3C）為例子**
   * + - **G3C定義**：假設有一G = ( V , E )，若存在一個映射（mapping），使得任一個邊，其兩個端點為，則稱G為「3-colorable」。
       - **前提假設**：
3. ：為G的一個3-coloring。
4. ：為 的隨機排列。
5. 定義 ：為一隨機的3-coloring。
6. 將每個放入一個標註為的箱子，並用金鑰（Key*v*）上鎖。
   * + - **ZKP過程**：
7. **P→V**：將前述箱子發送給驗證者，驗證端僅知道節點編號及邊，看不到點的著色。
8. **V→P**：隨機選擇一個邊，並且要求證明者提供兩端點和的顏色。
9. **P→V**：傳送存放和箱子的Key*u*及Key*v*給驗證者。
10. **V→P**：驗證端開啟箱子，若和顏色不同則「接受」，否則「拒絕」。

（重複多次，增加準確率）

* **可靠性（Soundness）：**假設G不為3-colorable，則至少有一個邊的兩個端點會是同色的（即），若P試圖作弊，驗證者會有 的機率選到不合規則的邊，且隨著重複多次執行，該機率將擴增至 以上，故V可以識破。
* **完整性（Completeness）：**若G真的為3-colorable，P及V依據前述協議進行交流後，可符合V任挑兩個端點的顏色皆不同，故V會接受。
* **零知識性（Zero-Knowledge）：**P可在不揭露「**節點顏色**」的情況下，向V證明圖G確實為3-colorable。
  + - * 互動圖：



1. **精進版Schnorr Protocol**

基於Schnorr Protocol改進的演算法，屬於**簡化版互動式零知識證明**，以**離散對數之難度**作為核心，Prover 知道一個秘密私鑰，計算，並提供 Verifier 公鑰，Prover 欲在不透漏的情況下，讓 Verifier 相信其知道秘密 ，以下為參數設定：

質數**：**：隨機挑選的大質數

：

生成元（）**：**

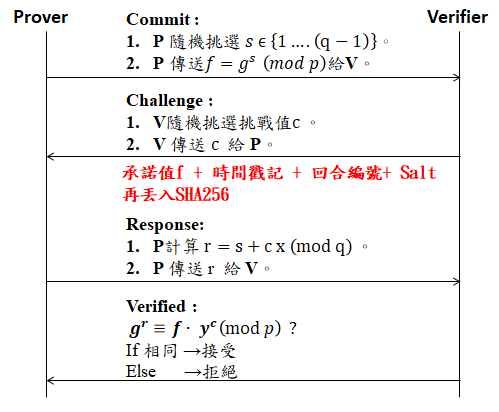
私鑰**： （隨機挑選）**

公鑰**：**

1. **承諾（commit）：**
2. Prover 隨機選擇一個 ephemeral key ，使得
3. 計算承諾值 ： 。
4. 傳送 給 verifier。
5. **挑戰（Challenge）：**
6. Verifier產生挑戰值c：將承諾值 放入SHA256產生c。
7. 傳送 給 Prover。
8. **回應（Response）：**
9. Prover 計算： 。
10. 傳送 給 Verifier。
11. Verifier 驗證：

（成立 → 接受；不成立 → 拒絕）

* **可靠性（Soundness）：**Prover在不知道 的情況下，其不可能找出所有 可以對應到有效的 值，故其透過亂猜的方式能夠回應滿足協議過程的 值機率僅為 ，故Prover難以欺騙Verifier。
* **完整性（Completeness）：**在前述協議執行一輪之後，Verifier將可確信Prover知道，首先 Prover 先隨機挑選 ，其後續只需傳送給Verifier，可使不知道的Verifier驗證，原因如下：
* **零知識性**（**Zero-Knowledge**）：Schnorr 協定滿足零知識性，即 Verifier 即使成功驗證 Prover 知道秘密 ，也無法從中得到任何的資訊。
  + - * 互動圖：

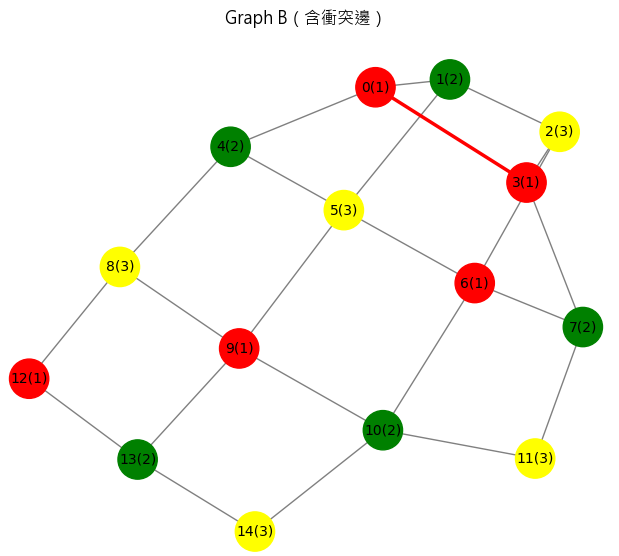
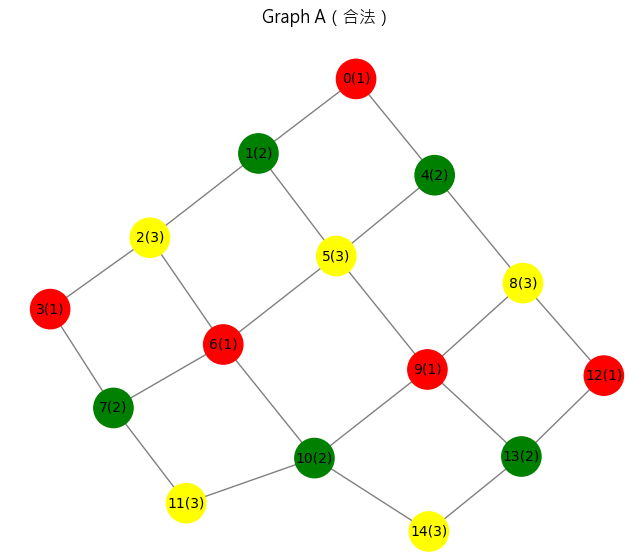


1. **Fiat–Shamir Heuristic（待研擬）**

**待研擬…**

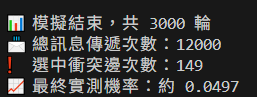
1. **專題成果**
2. **傳統ZKP — 以Graph 3-colorability（簡稱G3C）為例子**

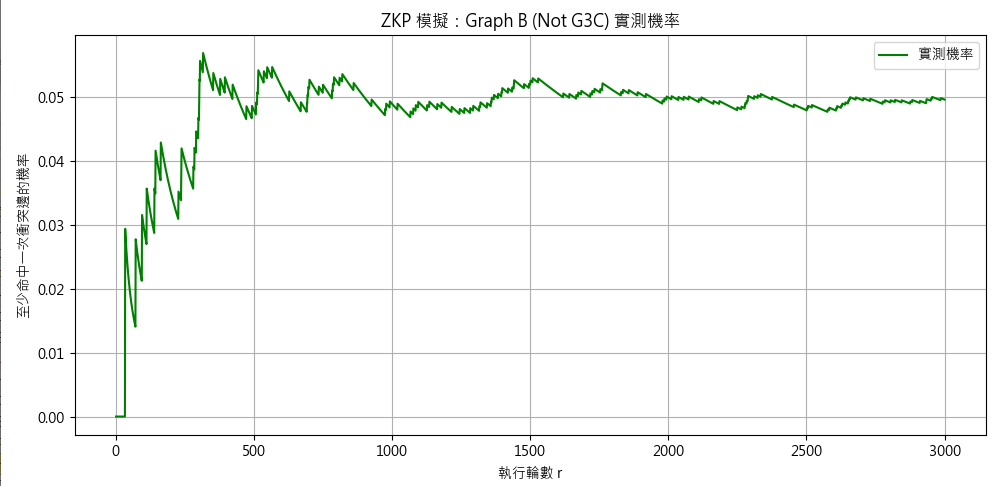
在本次模擬中，Graph A為3-coloring，Graph B為非3-coloring，針對Graph A模擬了20次，詳附錄八之（一）。



在Graph B模擬中，我們針對只有一個衝突邊（即不合法邊）的**非 3-colorable** 的圖（Graph B）進行 **3000 輪**的 ZKP 驗證實驗，並記錄每一輪中是否命中圖中唯一的一條不合法邊。如圖所示，橫軸為執行輪數 ，縱軸則為截至第 輪的「**至少命中一次衝突邊**」的累積實測機率：

* + - * 初始皆為0（未挑到衝突邊），後來逐漸變化劇烈。
      * 隨著「驗證次數」提高，挑到衝突邊機率快速上升，最終穩定於 **5%**，約等同於理論值 **。**
      * 執行多輪之後累積訊息傳達次數高達12000次。





1. **精進版Schnorr Protocol**

前述模擬，我們針對傳統互動式零信任機制採用 Graph 3-colorability（G3C） 問題作為理論基礎，實作並模擬 3000 輪互動，驗證 Prover 是否擁有一組合法著色，而不洩漏實際配色資訊。此方法雖具備完整性與零知識性，但在每一輪中需傳遞 4 筆訊息（承諾、挑邊…等），並且必須透過多輪的執行才能使可信任度提高，使得整體訊息傳輸成本隨輪數成長，總傳遞次數達 12000 次（見附錄八之一）。

為改善此問題，我們進一步研讀相關文獻並實作精進版Schnorr 協定，其核心為基於離散對數困難度建立的互動式 ZKP。相較於 G3C，精進版Schnorr 僅需交換一次承諾 ，收到挑戰 後，回傳 ，Verifier 再驗證 是否成立。整體來看，每輪僅需傳遞 3 筆訊息（承諾、挑戰、回應），大幅減少通訊負擔。

實驗結果顯示，模擬執行 100 輪精進版Schnorr 驗證，成功率達 100%（如附錄八之（二）），總訊息傳遞次數僅為 300 次，明顯低於傳統 ZKP 的 12000 次，在不犧牲安全性的前提下，顯示精進版Schnorr 協定在通訊效率上具有實質改善。

1. **Fiat–Shamir Heuristic（待研擬）**

**待研擬…**

1. **結論（待研擬）**

**待研擬…**

1. **未來展望（待研擬）**

**待研擬…**

1. **參考文獻**
2. Goldwasser, S., Micali, S., & Rackoff, C. (1985). The knowledge complexity of interactive proof-systems. *SIAM Journal on Computing*, *18*(1), 186–208. https://doi.org/10.1137/0218012
3. Schnorr, C. P. (1990). Efficient identification and signatures for smart cards. In G. Brassard (Ed.), Advances in Cryptology – CRYPTO ’89 (Lecture Notes in Computer Science, Vol. 435, pp. 239–252). Springer. https://doi.org/10.1007/0-387-34805-0\_21
4. Fiat, A., & Shamir, A. (1987). How to prove yourself: Practical solutions to identification and signature problems. In A. M. Odlyzko (Ed.), Advances in Cryptology – CRYPTO ’86 (Lecture Notes in Computer Science, Vol. 263, pp. 186–194). Springer. https://doi.org/10.1007/3-540-47721-7\_12
5. **附錄**
6. **傳統ZKP — 以Graph 3-colorability（簡稱G3C）為例子**
   * + - **執行結果**

====== Graph A（合法）ZKP 模擬 ======

🔁 第 1 輪：

- Prover 傳送承諾（1 次）

- Verifier 挑邊 (0, 1)（1 次）

- Prover 解鎖節點 0 色 1，節點 1 色 2（2 次）

✅ 驗證通過

📦 本輪傳遞訊息總數：4

🔁 第 2 輪：

- Prover 傳送承諾（1 次）

- Verifier 挑邊 (13, 14)（1 次）

- Prover 解鎖節點 13 色 2，節點 14 色 3（2 次）

✅ 驗證通過

📦 本輪傳遞訊息總數：4

🔁 第 3 輪：

- Prover 傳送承諾（1 次）

- Verifier 挑邊 (8, 9)（1 次）

- Prover 解鎖節點 8 色 3，節點 9 色 1（2 次）

✅ 驗證通過

📦 本輪傳遞訊息總數：4

🔁 第 4 輪：

- Prover 傳送承諾（1 次）

- Verifier 挑邊 (2, 3)（1 次）

- Prover 解鎖節點 2 色 3，節點 3 色 1（2 次）

✅ 驗證通過

📦 本輪傳遞訊息總數：4

🔁 第 5 輪：

- Prover 傳送承諾（1 次）

- Verifier 挑邊 (6, 10)（1 次）

- Prover 解鎖節點 6 色 1，節點 10 色 2（2 次）

✅ 驗證通過

📦 本輪傳遞訊息總數：4

🔁 第 6 輪：

- Prover 傳送承諾（1 次）

- Verifier 挑邊 (9, 10)（1 次）

- Prover 解鎖節點 9 色 1，節點 10 色 2（2 次）

✅ 驗證通過

📦 本輪傳遞訊息總數：4

🔁 第 7 輪：

- Prover 傳送承諾（1 次）

- Verifier 挑邊 (10, 14)（1 次）

- Prover 解鎖節點 10 色 2，節點 14 色 3（2 次）

✅ 驗證通過

📦 本輪傳遞訊息總數：4

🔁 第 8 輪：

- Prover 傳送承諾（1 次）

- Verifier 挑邊 (5, 9)（1 次）

- Prover 解鎖節點 5 色 3，節點 9 色 1（2 次）

✅ 驗證通過

📦 本輪傳遞訊息總數：4

🔁 第 9 輪：

- Prover 傳送承諾（1 次）

- Verifier 挑邊 (5, 6)（1 次）

- Prover 解鎖節點 5 色 3，節點 6 色 1（2 次）

✅ 驗證通過

📦 本輪傳遞訊息總數：4

🔁 第 10 輪：

- Prover 傳送承諾（1 次）

- Verifier 挑邊 (10, 11)（1 次）

- Prover 解鎖節點 10 色 2，節點 11 色 3（2 次）

✅ 驗證通過

📦 本輪傳遞訊息總數：4

🔁 第 11 輪：

- Prover 傳送承諾（1 次）

- Verifier 挑邊 (4, 8)（1 次）

- Prover 解鎖節點 4 色 2，節點 8 色 3（2 次）

✅ 驗證通過

📦 本輪傳遞訊息總數：4

🔁 第 12 輪：

- Prover 傳送承諾（1 次）

- Verifier 挑邊 (6, 7)（1 次）

- Prover 解鎖節點 6 色 1，節點 7 色 2（2 次）

✅ 驗證通過

📦 本輪傳遞訊息總數：4

🔁 第 13 輪：

- Prover 傳送承諾（1 次）

- Verifier 挑邊 (2, 6)（1 次）

- Prover 解鎖節點 2 色 3，節點 6 色 1（2 次）

✅ 驗證通過

📦 本輪傳遞訊息總數：4

🔁 第 14 輪：

- Prover 傳送承諾（1 次）

- Verifier 挑邊 (10, 14)（1 次）

- Prover 解鎖節點 10 色 2，節點 14 色 3（2 次）

✅ 驗證通過

📦 本輪傳遞訊息總數：4

🔁 第 15 輪：

- Prover 傳送承諾（1 次）

- Verifier 挑邊 (5, 9)（1 次）

- Prover 解鎖節點 5 色 3，節點 9 色 1（2 次）

✅ 驗證通過

📦 本輪傳遞訊息總數：4

🔁 第 16 輪：

- Prover 傳送承諾（1 次）

- Verifier 挑邊 (2, 6)（1 次）

- Prover 解鎖節點 2 色 3，節點 6 色 1（2 次）

✅ 驗證通過

📦 本輪傳遞訊息總數：4

🔁 第 17 輪：

- Prover 傳送承諾（1 次）

- Verifier 挑邊 (1, 5)（1 次）

- Prover 解鎖節點 1 色 2，節點 5 色 3（2 次）

✅ 驗證通過

📦 本輪傳遞訊息總數：4

🔁 第 18 輪：

- Prover 傳送承諾（1 次）

- Verifier 挑邊 (12, 13)（1 次）

- Prover 解鎖節點 12 色 1，節點 13 色 2（2 次）

✅ 驗證通過

📦 本輪傳遞訊息總數：4

🔁 第 19 輪：

- Prover 傳送承諾（1 次）

- Verifier 挑邊 (8, 12)（1 次）

- Prover 解鎖節點 8 色 3，節點 12 色 1（2 次）

✅ 驗證通過

📦 本輪傳遞訊息總數：4

🔁 第 20 輪：

- Prover 傳送承諾（1 次）

- Verifier 挑邊 (8, 9)（1 次）

- Prover 解鎖節點 8 色 3，節點 9 色 1（2 次）

✅ 驗證通過

📦 本輪傳遞訊息總數：4

📊 模擬結束，共 20 輪

📨 總訊息傳遞次數：80

* + - * **程式碼**

#Traditional ZKP with G3C Problem

#Author: 林伯叡、黃杬霆

#Date: 2025/05/15

import random

import copy

import networkx as nx

import matplotlib

import matplotlib.pyplot as plt

matplotlib.rcParams['font.family'] = 'Microsoft JhengHei'  # 微軟正黑體

matplotlib.rcParams['axes.unicode\_minus'] = False  # 避免負號變成亂碼

# === 圖與著色 ===

graph\_A = {

    0: [1, 4], 1: [0, 2, 5], 2: [1, 3, 6], 3: [2, 7],

    4: [0, 5, 8], 5: [1, 4, 6, 9], 6: [2, 5, 7, 10], 7: [3, 6, 11],

    8: [4, 9, 12], 9: [5, 8, 10, 13], 10: [6, 9, 11, 14], 11: [7, 10],

    12: [8, 13], 13: [9, 12, 14], 14: [10, 13]

}

colors\_A = {

    0: 1, 1: 2, 2: 3, 3: 1, 4: 2, 5: 3,

    6: 1, 7: 2, 8: 3, 9: 1, 10: 2, 11: 3,

    12: 1, 13: 2, 14: 3

}

graph\_B = copy.deepcopy(graph\_A)

graph\_B[0].append(3)

graph\_B[3].append(0)  # 非法邊

# === ZKP 模擬函式 ===

def simulate\_zkp\_rounds(graph, colors, is\_valid=True, rounds=20):

    total\_messages = 0

    conflict\_hits = 0

    conflict\_history = []

    log = []

    edge\_set = set()

    for u in graph:

        for v in graph[u]:

            if (v, u) not in edge\_set:

                edge\_set.add((u, v))

    edge\_list = list(edge\_set)

    for r in range(1, rounds + 1):

        messages = 0

        log.append(f"🔁 第 {r} 輪：")

        messages += 1

        log.append(f"  - Prover 傳送承諾（1 次）")

        u, v = random.choice(edge\_list)

        messages += 1

        log.append(f"  - Verifier 挑邊 ({u}, {v})（1 次）")

        messages += 2

        log.append(f"  - Prover 解鎖節點 {u} 色 {colors[u]}，節點 {v} 色 {colors[v]}（2 次）")

        if colors[u] == colors[v]:

            log.append(f"  ❌ 顏色衝突，驗證失敗")

            conflict\_hits += 1

            conflict\_history.append(1)

        else:

            log.append(f"  ✅ 驗證通過")

            conflict\_history.append(0)

        total\_messages += messages

        log.append(f"  📦 本輪傳遞訊息總數：{messages}")

        if not is\_valid:

            empirical\_p = sum(conflict\_history) / r

            theoretical\_p = 1 - (1 - (1 / len(edge\_list))) \*\* r

            log.append(f"  📈 累積實測機率：{empirical\_p:.4f}，理論機率：約 {theoretical\_p:.4f}")

    log.append(f"\n📊 模擬結束，共 {rounds} 輪")

    log.append(f"📨 總訊息傳遞次數：{total\_messages}")

    if not is\_valid:

        log.append(f"❗ 選中衝突邊次數：{conflict\_hits}")

        final\_empirical = sum(conflict\_history) / rounds

        final\_theoretical = 1 - (1 - (1 / len(edge\_list))) \*\* rounds

        log.append(f"📈 最終實測機率：約 {final\_empirical:.4f}")

        log.append(f"📈 最終理論機率：約 {final\_theoretical:.4f}")

    return log

# === 繪圖函式 ===

def draw\_single\_graph(graph, colors, title, highlight\_conflict=False):

    G = nx.Graph()

    for u in graph:

        for v in graph[u]:

            G.add\_edge(u, v)

    pos = nx.spring\_layout(G, seed=42)

    color\_map = {1: 'red', 2: 'green', 3: 'yellow'}

    node\_colors = [color\_map[colors[n]] for n in G.nodes()]

    conflict\_edges = []

    normal\_edges = []

    for u, v in G.edges():

        if colors[u] == colors[v] and highlight\_conflict:

            conflict\_edges.append((u, v))

        else:

            normal\_edges.append((u, v))

    plt.figure(figsize=(7, 6))

    nx.draw\_networkx\_nodes(G, pos, node\_color=node\_colors, node\_size=800)

    nx.draw\_networkx\_edges(G, pos, edgelist=normal\_edges, edge\_color='gray')

    if highlight\_conflict:

        nx.draw\_networkx\_edges(G, pos, edgelist=conflict\_edges, edge\_color='red', width=2.5)

    nx.draw\_networkx\_labels(G, pos, labels={n: f"{n}({colors[n]})" for n in G.nodes()}, font\_size=10)

    plt.title(title)

    plt.axis("off")

    plt.tight\_layout()

    plt.show()

# === 合法性檢查（推薦搭配） ===

def check\_graph\_validity(graph, colors):

    for u in graph:

        for v in graph[u]:

            if u < v and colors[u] == colors[v]:

                print(f"❌ 發現衝突邊 ({u}, {v})，同為色 {colors[u]}")

                return False

    print("✅ 著色合法，無衝突邊")

    return True

enable\_check = "TRUE"

assert enable\_check in [

    "TRUE",

    "FALSE",

], f"Unsupported enable\_check: {enable\_check}"

enable\_plotcurve = "FALSE"

assert enable\_plotcurve in [

    "TRUE",

    "FALSE",

], f"Unsupported enable\_plotcurve: {enable\_plotcurve}"

if enable\_check == "TRUE":

    # === 執行檢查與單圖繪製 ===

    print("\n🧪 檢查 Graph A 合法性：")

    check\_graph\_validity(graph\_A, colors\_A)

    print("\n🧪 檢查 Graph B 合法性：")

    check\_graph\_validity(graph\_B, colors\_A)

    draw\_single\_graph(graph\_A, colors\_A, "Graph A（合法）", highlight\_conflict=True)

    draw\_single\_graph(graph\_B, colors\_A, "Graph B（含衝突邊）", highlight\_conflict=True)

# === 執行模擬 ===

rounds\_A = 20

rounds\_B = 3000

log\_A = simulate\_zkp\_rounds(graph\_A, colors\_A, is\_valid=True, rounds=rounds\_A)

log\_B = simulate\_zkp\_rounds(graph\_B, colors\_A, is\_valid=False, rounds=rounds\_B)

print("\n====== Graph A（合法）ZKP 模擬 ======")

for line in log\_A:

    print(line)

print("\n====== Graph B（非法）ZKP 模擬 ======")

for line in log\_B:

    print(line)

if enable\_plotcurve == "TRUE":

    def extract\_conflict\_history(log):

        return [1 if "❌ 顏色衝突" in line else 0 for line in log if "✅ 驗證通過" in line or "❌ 顏色衝突" in line]

    conflict\_history = extract\_conflict\_history(log\_B)

    total\_edges = sum(len(neigh) for neigh in graph\_B.values()) // 2

    empirical\_probs = []

    conflict\_sum = 0

    for r in range(1, len(conflict\_history)+1):

        conflict\_sum += conflict\_history[r - 1]

        empirical\_probs.append(conflict\_sum / r)

    theoretical\_probs = [1 - (1 - 1 / total\_edges) \*\* r for r in range(1, len(conflict\_history)+1)]

    # 畫圖

    plt.figure(figsize=(10, 5))

    plt.plot(range(1, rounds\_B + 1), empirical\_probs, label="實測機率", color='Green')

    # plt.plot(range(1, rounds\_B + 1), theoretical\_probs, label="理論機率", linestyle='--', color='Red')

    plt.xlabel("執行輪數 r")

    plt.ylabel("至少命中一次衝突邊的機率")

    plt.title("ZKP 模擬：Graph B (Not G3C) 實測機率")

    # plt.title("ZKP 模擬：Graph B 實測機率 vs 理論機率")

    plt.grid(True)

    plt.legend()

    plt.tight\_layout()

    plt.show()

1. **精進版Schnorr Protocol**
2. **執行結果：**

**Round 1: s=13545, f=432246, c=985, r=41357, g^r=200258, f\*y^c=200258, pass=True**

**Round 2: s=42823, f=275753, c=45929, r=1384, g^r=147119, f\*y^c=147119, pass=True**

**Round 3: s=29967, f=59415, c=39885, r=13413, g^r=31255, f\*y^c=31255, pass=True**

**Round 4: s=22373, f=181677, c=23653, r=9624, g^r=303812, f\*y^c=303812, pass=True**

**Round 5: s=44178, f=71781, c=36923, r=25470, g^r=53175, f\*y^c=53175, pass=True**

**Round 6: s=3008, f=491525, c=44768, r=39193, g^r=162756, f\*y^c=162756, pass=True**

**Round 7: s=30091, f=431346, c=48953, r=46170, g^r=292132, f\*y^c=292132, pass=True**

**Round 8: s=32804, f=342342, c=20724, r=37269, g^r=25141, f\*y^c=25141, pass=True**

**Round 9: s=8610, f=211089, c=30510, r=35413, g^r=326131, f\*y^c=326131, pass=True**

**Round 10: s=30493, f=385727, c=41757, r=7302, g^r=251268, f\*y^c=251268, pass=True**

**Round 11: s=40044, f=50829, c=13418, r=45045, g^r=511306, f\*y^c=511306, pass=True**

**Round 12: s=50877, f=260326, c=6952, r=8612, g^r=58915, f\*y^c=58915, pass=True**

**Round 13: s=30463, f=278990, c=25565, r=1346, g^r=442410, f\*y^c=442410, pass=True**

**Round 14: s=49904, f=286218, c=50632, r=26719, g^r=176888, f\*y^c=176888, pass=True**

**Round 15: s=29255, f=490302, c=37228, r=47237, g^r=277090, f\*y^c=277090, pass=True**

**Round 16: s=39036, f=460494, c=51636, r=16863, g^r=160260, f\*y^c=160260, pass=True**

**Round 17: s=42482, f=310736, c=20894, r=18636, g^r=118677, f\*y^c=118677, pass=True**

**Round 18: s=203, f=307151, c=9755, r=47042, g^r=197886, f\*y^c=197886, pass=True**

**Round 19: s=4461, f=366124, c=4010, r=33451, g^r=451097, f\*y^c=451097, pass=True**

**Round 20: s=37083, f=110892, c=48285, r=12156, g^r=221366, f\*y^c=221366, pass=True**

**Round 21: s=25310, f=199721, c=12813, r=40512, g^r=158524, f\*y^c=158524, pass=True**

**Round 22: s=29494, f=509373, c=21302, r=31631, g^r=44745, f\*y^c=44745, pass=True**

**Round 23: s=39064, f=430168, c=21830, r=37991, g^r=286522, f\*y^c=286522, pass=True**

**Round 24: s=39686, f=33315, c=28448, r=28028, g^r=116961, f\*y^c=116961, pass=True**

**Round 25: s=44888, f=459214, c=2639, r=32995, g^r=120285, f\*y^c=120285, pass=True**

**Round 26: s=19633, f=100421, c=43202, r=42805, g^r=127213, f\*y^c=127213, pass=True**

**Round 27: s=30194, f=499088, c=33269, r=15911, g^r=514701, f\*y^c=514701, pass=True**

**Round 28: s=26731, f=480525, c=24987, r=52125, g^r=192042, f\*y^c=192042, pass=True**

**Round 29: s=36485, f=57401, c=45692, r=40961, g^r=25712, f\*y^c=25712, pass=True**

**Round 30: s=11242, f=479579, c=41718, r=45808, g^r=480075, f\*y^c=480075, pass=True**

**Round 31: s=41561, f=384799, c=3021, r=7184, g^r=169189, f\*y^c=169189, pass=True**

**Round 32: s=1753, f=462764, c=11799, r=5330, g^r=188896, f\*y^c=188896, pass=True**

**Round 33: s=24917, f=374609, c=28648, r=16787, g^r=266200, f\*y^c=266200, pass=True**

**Round 34: s=47223, f=397238, c=51852, r=14249, g^r=191900, f\*y^c=191900, pass=True**

**Round 35: s=37882, f=345350, c=3897, r=14783, g^r=462189, f\*y^c=462189, pass=True**

**Round 36: s=19394, f=478789, c=23766, r=6551, g^r=140318, f\*y^c=140318, pass=True**

**Round 37: s=44838, f=354704, c=35500, r=44862, g^r=291922, f\*y^c=291922, pass=True**

**Round 38: s=40368, f=462599, c=5034, r=39413, g^r=177993, f\*y^c=177993, pass=True**

**Round 39: s=9153, f=192718, c=11890, r=51907, g^r=98680, f\*y^c=98680, pass=True**

**Round 40: s=10780, f=244988, c=51981, r=15566, g^r=243890, f\*y^c=243890, pass=True**

**Round 41: s=6289, f=241925, c=34745, r=13868, g^r=99571, f\*y^c=99571, pass=True**

**Round 42: s=40453, f=137826, c=25807, r=49002, g^r=226666, f\*y^c=226666, pass=True**

**Round 43: s=17201, f=207033, c=42965, r=34105, g^r=439958, f\*y^c=439958, pass=True**

**Round 44: s=47649, f=51600, c=18535, r=19762, g^r=436581, f\*y^c=436581, pass=True**

**Round 45: s=2132, f=93769, c=11976, r=482, g^r=348532, f\*y^c=348532, pass=True**

**Round 46: s=51457, f=504529, c=10991, r=21995, g^r=44099, f\*y^c=44099, pass=True**

**Round 47: s=24481, f=287430, c=8842, r=5119, g^r=226642, f\*y^c=226642, pass=True**

**Round 48: s=14347, f=278349, c=21288, r=22499, g^r=336685, f\*y^c=336685, pass=True**

**Round 49: s=49930, f=157616, c=7125, r=19066, g^r=460022, f\*y^c=460022, pass=True**

**Round 50: s=28742, f=71273, c=48566, r=35907, g^r=19321, f\*y^c=19321, pass=True**

**Round 51: s=2143, f=473209, c=38865, r=51089, g^r=161519, f\*y^c=161519, pass=True**

**Round 52: s=20926, f=448204, c=18797, r=51931, g^r=160257, f\*y^c=160257, pass=True**

**Round 53: s=37114, f=205496, c=25511, r=23743, g^r=96243, f\*y^c=96243, pass=True**

**Round 54: s=40372, f=30001, c=16115, r=1106, g^r=14580, f\*y^c=14580, pass=True**

**Round 55: s=24578, f=373547, c=27400, r=38267, g^r=128705, f\*y^c=128705, pass=True**

**Round 56: s=8944, f=188436, c=25984, r=12266, g^r=67796, f\*y^c=67796, pass=True**

**Round 57: s=49033, f=264843, c=46013, r=23687, g^r=194087, f\*y^c=194087, pass=True**

**Round 58: s=44235, f=118451, c=32354, r=24248, g^r=467571, f\*y^c=467571, pass=True**

**Round 59: s=4693, f=376332, c=37207, r=5606, g^r=290552, f\*y^c=290552, pass=True**

**Round 60: s=30146, f=490657, c=13026, r=47018, g^r=129597, f\*y^c=129597, pass=True**

**Round 61: s=2844, f=280331, c=7855, r=16167, g^r=116867, f\*y^c=116867, pass=True**

**Round 62: s=29301, f=232551, c=23129, r=3134, g^r=230995, f\*y^c=230995, pass=True**

**Round 63: s=39017, f=211445, c=32769, r=15914, g^r=152412, f\*y^c=152412, pass=True**

**Round 64: s=18052, f=90656, c=45764, r=36322, g^r=83925, f\*y^c=83925, pass=True**

**Round 65: s=12569, f=470976, c=40424, r=51444, g^r=319165, f\*y^c=319165, pass=True**

**Round 66: s=26633, f=197136, c=46051, r=52053, g^r=370198, f\*y^c=370198, pass=True**

**Round 67: s=20159, f=354318, c=49351, r=51608, g^r=40506, f\*y^c=40506, pass=True**

**Round 68: s=38880, f=250102, c=168, r=18883, g^r=41851, f\*y^c=41851, pass=True**

**Round 69: s=34786, f=371421, c=41500, r=36284, g^r=172515, f\*y^c=172515, pass=True**

**Round 70: s=36838, f=514451, c=33488, r=51466, g^r=284979, f\*y^c=284979, pass=True**

**Round 71: s=15511, f=308142, c=12574, r=32759, g^r=485918, f\*y^c=485918, pass=True**

**Round 72: s=26277, f=302278, c=23880, r=9183, g^r=309925, f\*y^c=309925, pass=True**

**Round 73: s=35425, f=41852, c=12552, r=39761, g^r=357296, f\*y^c=357296, pass=True**

**Round 74: s=47270, f=251387, c=3610, r=17021, g^r=9148, f\*y^c=9148, pass=True**

**Round 75: s=1384, f=147119, c=14867, r=36120, g^r=488613, f\*y^c=488613, pass=True**

**Round 76: s=32046, f=96616, c=17532, r=51173, g^r=58663, f\*y^c=58663, pass=True**

**Round 77: s=11458, f=4847, c=12885, r=40454, g^r=167299, f\*y^c=167299, pass=True**

**Round 78: s=19718, f=114836, c=5625, r=14577, g^r=262139, f\*y^c=262139, pass=True**

**Round 79: s=17616, f=338543, c=50058, r=32314, g^r=238719, f\*y^c=238719, pass=True**

**Round 80: s=6681, f=206222, c=38807, r=35818, g^r=70323, f\*y^c=70323, pass=True**

**Round 81: s=34091, f=309869, c=25252, r=1540, g^r=218830, f\*y^c=218830, pass=True**

**Round 82: s=21387, f=428850, c=19290, r=38128, g^r=455081, f\*y^c=455081, pass=True**

**Round 83: s=20516, f=437484, c=38409, r=34283, g^r=342616, f\*y^c=342616, pass=True**

**Round 84: s=19443, f=194251, c=2947, r=31769, g^r=42902, f\*y^c=42902, pass=True**

**Round 85: s=8581, f=320966, c=36295, r=43502, g^r=112980, f\*y^c=112980, pass=True**

**Round 86: s=44419, f=122076, c=21464, r=51501, g^r=430251, f\*y^c=430251, pass=True**

**Round 87: s=28104, f=412172, c=37426, r=5745, g^r=466700, f\*y^c=466700, pass=True**

**Round 88: s=50780, f=381494, c=16882, r=6258, g^r=521055, f\*y^c=521055, pass=True**

**Round 89: s=48948, f=499607, c=41257, r=16937, g^r=336749, f\*y^c=336749, pass=True**

**Round 90: s=16656, f=306481, c=15681, r=7306, g^r=493200, f\*y^c=493200, pass=True**

**Round 91: s=15599, f=113262, c=243, r=49108, g^r=264074, f\*y^c=264074, pass=True**

**Round 92: s=38774, f=450289, c=2065, r=12581, g^r=80311, f\*y^c=80311, pass=True**

**Round 93: s=11556, f=304757, c=31756, r=24854, g^r=33229, f\*y^c=33229, pass=True**

**Round 94: s=8397, f=405195, c=26461, r=11784, g^r=453402, f\*y^c=453402, pass=True**

**Round 95: s=34155, f=401644, c=3861, r=4159, g^r=269169, f\*y^c=269169, pass=True**

**Round 96: s=22949, f=271373, c=18864, r=36350, g^r=411368, f\*y^c=411368, pass=True**

**Round 97: s=1912, f=283424, c=33465, r=7785, g^r=38808, f\*y^c=38808, pass=True**

**Round 98: s=22536, f=92287, c=19623, r=11754, g^r=97472, f\*y^c=97472, pass=True**

**Round 99: s=25813, f=71401, c=4090, r=35341, g^r=31945, f\*y^c=31945, pass=True**

**Round 100: s=51724, f=326315, c=15826, r=13622, g^r=441336, f\*y^c=441336, pass=True**

**Total successful rounds: 100/100**

1. **程式碼：**
2. #ZKP with 精進版Schnorr
3. #Author: 林伯叡、黃杬霆
4. #Date: 2025/05/08
5. from sympy import isprime
6. import random
7. import os
8. import hashlib
9. import time
10. # ===== 改進挑戰值：使用 (承諾值f + 時間 + 回合編號) 雜湊產生 c =====
11. def get\_challenge(f, round\_id, timestamp, q):
12. """
13. 模擬 Verifier 使用 (承諾值f + 時間 + 回合編號) 雜湊產生挑戰值 c
14. timestamp 為外部傳入，避免時間不一致問題
15. """
16. input\_str = f"{f}|{round\_id}|{timestamp}"
17. h = hashlib.sha256(input\_str.encode()).hexdigest()
18. return (int(h, 16) % (q - 1)) + 1
19. # ===== 公開參數設定 =====
20. def generate\_safe\_prime(bits=8):
21. """
22. 產生一組安全質數 p, q，使得 p = q \* r + 1，且 q, p 都是質數
23. bits: q 的位元長度
24. """
25. while True:
26. q = random.getrandbits(bits)
27. q |= 1  # 確保是奇數
28. if isprime(q):
29. for r in range(2, 20):
30. p = q \* r + 1
31. if isprime(p):
32. return p, q
33. # 產生參數 p, q
34. p, q = generate\_safe\_prime(bits=16)
35. def find\_generator(p, q):
36. """找一個生成元 g，使得 g^q ≡ 1 mod p"""
37. for g in range(2, p):
38. if pow(g, q, p) == 1:
39. return g
40. raise Exception("找不到生成元")
41. g = find\_generator(p, q)
42. x = random.randint(1, q - 1)
43. y = pow(g, x, p)
44. # ===== Schnorr 協定主程式 =====
45. def schnorr\_proof(rounds=100):
46. """執行 Schnorr 協定共 rounds 回合，顯示每輪資訊與統計成功次數"""
47. logs = []
48. success\_count = 0
49. for i in range(1, rounds + 1):
50. # Commit
51. s = random.randint(1, q - 1)
52. f = pow(g, s, p)
53. # 模擬 Verifier 在這一刻產生挑戰值，時間戳記記下來
54. timestamp = "{:.6f}".format(time.time())
55. c = get\_challenge(f, i, timestamp, q)
56. # Prover 回應
57. r = (s + c \* x) % q
58. # Verifier 驗證
59. left = pow(g, r, p)
60. right = (f \* pow(y, c, p)) % p
61. passed = left == right
62. if passed:
63. success\_count += 1
64. logs.append(f"Round {i}: s={s}, f={f}, c={c}, r={r}, g^r={left}, f\*y^c={right}, pass={passed}")
65. logs.append(f"\nTotal successful rounds: {success\_count}/{rounds}")
66. return logs
67. def write\_result(logs, filename="result.txt"):
68. """將結果寫入與此 .py 程式同一個資料夾"""
69. script\_dir = os.path.dirname(os.path.abspath(\_\_file\_\_))  # 此 .py 檔案的所在資料夾
70. filepath = os.path.join(script\_dir, filename)
71. with open(filepath, "w") as f:
72. f.write("\n".join(logs))
73. print(f"結果已寫入：{filepath}")
74. # ===== 主程式執行區塊 =====
75. if \_\_name\_\_ == "\_\_main\_\_":
76. result\_logs = schnorr\_proof(100)
77. write\_result(result\_logs)
78. **Fiat–Shamir Heuristic（待研擬）**

**待研擬…**