Regular Paper

Communications of the CCISA
Vol. 31 No. 3 Aug. 2025

後量子加密演算法 ML-KEM 於嵌入式裝置上之深度學習型旁通道攻擊分析

郭崇韋[1*]、洪宇義[2]、莊恆豪[3]、劉嘉瑞[4]
[1,2,4]逢甲大學資訊工程學系
[3]逢甲大學資通安全碩士學位學程
[1] cwkuo@fcu.edu.tw、[2] M1221097@o365.fcu.edu.tw、[3] M1401451@o365.fcu.edu.tw、[4] D0909002@o365.fcu.edu.tw

摘要

隨著第五代行動通訊技術 (5G) 的推廣與物聯網 (Internet of Things, IoT) 應用的普及，越來越多微控制器設備承載關鍵性資料，然而傳統公鑰密碼機制已難以抵禦日益威脅的量子計算攻擊。為強化資安保護，美國國家標準暨技術研究院 (National Institute of Standards and Technology, NIST) 提出模組格金鑰封裝機制 (Module Lattice Key Encapsulation Mechanism, ML-KEM)，作為後量子密碼學 (Post-quantum Cryptography, PQC) 標準之一，具備良好之運算效率與量子安全性，並適用於嵌入式裝置環境。然而，即便 ML-KEM 理論上具有抗量子攻擊能力，其實體實作仍可能遭受旁通道分析 (Side-Channel Analysis, SCA) 威脅。為驗證此潛在風險，本文提出一套結合深度學習之功率分析攻擊流程，針對 ML-KEM 實作進行 SCA。我們以 ChipWhisperer CW308 平台搭配 STM32F415-RGT6 Cortex-M4 微控制器執行 ML-KEM，並透過 Picoscope 5244B 示波器擷取運算過程中之功率消耗波形，將軌跡資料上傳至圖形處理單元 (Graphics Processing Unit, GPU) 平台訓練神經網路模型。實驗結果顯示，所訓練模型可達 100% 驗證準確率與 99.98% 攻擊成功率，單次功率軌跡即可還原 32 Bytes 封裝金鑰。此研究結果證實 ML-KEM 現階段實作於微控制器中，仍暴露於深度學習型 SCA 之高風險，未來須搭配有效之物理層防禦技術，以提升整體後量子加密系統的實體安全性。

關鍵詞：第五代行動通訊技術 (5G)、物聯網 (IoT)、旁通道分析 (SCA)、ML-KEM、微控制器、後量子密碼學 (PQC)、深度學習

* 通訊作者 (Corresponding author.)

Regular Paper

Communications of the CCISA
Vol. 31 No. 3 Aug. 2025

Analysis of Deep Learning-Based Side-Channel Attacks on the Post-Quantum Cryptography Algorithm ML-KEM on Embedded Devices

Chung-Wei Kuo[1*], Yu-Yi Hong [2], heng-hao Zhuang[3], and Jia-Ruei Liu [4]
[1,2,4]Department of Information Engineering and Computer Science, Feng Chia University
[3]Master's Program of Information and Communication Security, Feng Chia University
[1]cwkuo@fcu.edu.tw, [2]M1221097@o365.fcu.edu.tw, [3]M1401451@o365.fcu.edu.tw,
[4]D0909002@o365.fcu.edu.tw

Abstract

With the rapid adoption of 5G and the proliferation of IoT applications, microcontroller-based devices are increasingly entrusted with sensitive data. However, traditional public-key cryptography is insufficient against emerging quantum computing threats. To address this, the National Institute of Standards and Technology (NIST) introduced the Module Lattice Key Encapsulation Mechanism (ML-KEM) as part of its Post-quantum Cryptography (PQC) standards, offering both efficiency and quantum resistance for embedded systems. Despite its theoretical resilience, practical ML-KEM implementations remain susceptible to side-channel analysis (SCA).

In this work, we propose a deep learning-based power analysis framework to evaluate the side-channel vulnerability of ML-KEM. Using the ChipWhisperer CW308 platform with an STM32F415-RGT6 Cortex-M4 microcontroller, we capture power traces with a Picoscope 5244B oscilloscope and train neural networks on a Graphics Processing Unit (GPU) platform. Our experiments achieve 100% validation accuracy and a 99.98% attack success rate, recovering a 32-byte encapsulation key from a single power trace.

These results highlight the high risk of deep learning-based SCA against ML-KEM on microcontrollers, underscoring the need for robust physical-layer countermeasures to secure future PQC implementations.

Keywords: 5G, Internet of Thing, Side-channel Analysis, ML-KEM, Microcontroller, Post-quantum Cryptography, Deep Learning

壹、前言

隨著物聯網 (Internet of Things, IoT) 與無線通訊技術的快速發展，資訊與通訊技術 (Information and Communication Technology, ICT) 受到巨大影響，許多場域開始廣泛導入 IoT 裝置以開發新應用，進而提升生活便利性。智慧都市、智慧工廠與智慧居家即是 IoT 改善生活品質的典型案例。根據 Statista 的統計 [1]，全球 IoT 連網裝置數量將於今年達到 18 億，並預估在 2033 年增長至 39.6 億。這些裝置多透過 Wi-Fi 與智慧中樞連結，使使用者能以行動裝置遠端控制家電與相關設備。然而，隨之而來的資安隱憂亦不容忽視。

在無線通訊過程中，若缺乏適當的加密防護，敏感資訊可能因傳輸過程遭攔截而洩漏。因此，採用強度足夠的加密演算法成為保護資料的必要措施。現行的進階加密標準 (Advanced Encryption Standard, AES) 與 RSA 演算法均屬高安全性技術，即便密文遭截獲，攻擊者仍需付出高昂成本與時間才能破解。然而，隨著量子計算技術快速演進，傳統公鑰密碼系統正面臨嚴重威脅。1994 年，Peter Shor 提出的 Shor 演算法 [2] 可利用量子電腦於多項式時間內有效求解大整數分解與離散對數問題，進而破解目前廣泛應用的 RSA 與橢圓曲線密碼編譯 (Elliptic Curve Cryptography, ECC) 系統，對現行公鑰密碼構成致命挑戰。為了預防此類風險，PQC 逐漸成為研究焦點。

NIST 於 2017 年啟動 PQC 標準化計畫，目標是發展可抵抗量子攻擊的公鑰加密演算法。2020 年 7 月，經過多輪篩選，共有七個決賽入圍者與八個備選方案進入第三輪，其中金鑰封裝機制 (Key Encapsulation Mechanism, KEM) 類別包含四個決賽演算法與五個備選方案。CRYSTALS-Kyber(Kyber) [3] 作為四個 KEM 決賽入圍者之一，最終在 2022 年 7 月被 NIST 選定為唯一進入標準化的 KEM，並於 2024 年 8 月 13 日正式命名為 Module Lattice Key Encapsulation Mechanism (ML-KEM) [4]，成為 PQC 中公鑰加密與金鑰交換的國際標準。

儘管 ML-KEM 的理論安全性已獲廣泛驗證，但其在實際硬體實作中仍存在潛在漏洞，其中最具威脅性的攻擊方式之一便是旁通道分析 (Side-Channel Analysis, SCA)。SCA 由 Paul Kocher 於 1996 年首次提出 [5]，攻擊者可透過分析設備在運算過程中洩漏的物理資訊，如功率消耗 [6]、電磁輻射 [7]、運算延遲 [6] 或溫度變化 [8]，以推測或破解敏感資料。

隨著人工智慧 (Artificial Intelligence, AI) 與深度學習 (Deep Learning, DL) 的快速發展，SCA 的攻擊能力進一步增強，對加密系統的硬體防護構成更嚴峻挑戰。因此，本研究聚焦於 ML-KEM 的硬體實作安全性分析，並特別探討如何運用深度學習方法強化 SCA 的效能。

貳、文獻探討

2.1 ML-KEM

ML-KEM 為新一代後量子公鑰加密/金鑰封裝機制 (PKE/KEM) 標準，其核心是建立在模學習誤差 (Module Learning-with-errors, MLWE) 難題之上。與基於環學習誤差 (Ring Learning-with-errors, RLWE) 的 NewHope [9] 相比，ML-KEM 在提供相同安全水準的同時，能產生更為精簡的公私鑰對，並且可以依據安全需求調整多項式向量的維度。為了提高多項式乘法的速度，ML-KEM 引入數論變換 (Number Theoretic Transform, NTT)，將多項式係數轉換到 NTT 域中進行計算，使得運算效率從 $O(n^2)$ 提升至 O(nlogn)，這也是其被選為標準的關鍵因素之一。

ML-KEM.CPAPKE 是一個針對選擇明文攻擊 (Chosen-plaintext attack, CPA) 具備 IND-CPA 安全性的公開金鑰加密方案。它還利用經過調整的 Fujisaki-Okamoto 轉換 (FO Transform)，將 IND-CPA 的安全性強化為 IND-CCA2，以抵禦更為強大的選擇密文攻擊 (Chosen-ciphertext attack, CCA)。具體的防禦機制是在解封裝時，對解出的訊息進行再次加密，並與原始密文進行比對，透過這種驗證程序，確保 ML-KEM 實現 IND-CCA2 的高度安全性。

ML-KEM.CPAPKE 的運作包含三個主要步驟：金鑰生成、加密和解密。在金鑰生成的環節中，會取樣多項式矩陣 $\boldsymbol{A}$ 和多項式向量 $\boldsymbol{s}$。公鑰則是將多項式矩陣 $\boldsymbol{A}$ 與多項式向量 $\boldsymbol{s}$ 乘積後，加上一個誤差多項式向量而取得。其中 $\boldsymbol{s}$ 和 $\boldsymbol{e}$ 都是包含 n 個多項式的向量，它們都會被轉換到 NTT 域以加速多項式乘法，每個多項式都包含k個係數。

在訊息加密的過程中，一個隨機訊息會被編譯成密文，這個密文由兩個壓縮過的部分組成，分別為多項式向量和一個多項式。進行解密的步驟時，首先會將密文解壓縮，接著將多項式向量轉換到 NTT 域，並與私鑰進行內積運算，然後再用該多項式與內積結果相減，即可還原原始的隨機訊息。ML-KEM.CPAPKE 在加密時，如 Algorithm 1所示，會將32個位元組的隨機訊息轉換為256維的多項式形式，以便後續的運算。此外，ML-KEM.CCAKEM.Dec 進行解封裝時，如 Algorithm 2所示，也會對解出的隨機訊息重新進行加密，同樣會使用到相同的演算法，而本研究也是針對此演算法進行 SCA。

Algorithm 1 ML-KEM.CPAPKE.Enc$(\boldsymbol{pk}, \boldsymbol{m}, \boldsymbol{r})$ (Simplified)

$\widehat{\mathbf{A}}^{T} \in \mathbf{XOF}(\mathbf{seed}_{A}) \in \mathcal{R}_q^{k\times k}$

$\boldsymbol{r} \leftarrow \boldsymbol{\beta}_{\mu}(\mathcal{R}_q^{k\times 1}; \boldsymbol{r});\ \boldsymbol{e_1} \leftarrow \boldsymbol{\beta}_{\mu}(\mathcal{R}_q^{k\times 1}; \boldsymbol{r}); \boldsymbol{e_2} \leftarrow \boldsymbol{\beta}_{\mu}(\mathcal{R}_q; \boldsymbol{r})$

$\hat{\boldsymbol{r}} = \mathbf{NTT}(\boldsymbol{r}) \in \mathcal{R}_q^{k\times 1}$

$\boldsymbol{u} = \mathbf{NTT}^{-1}(\widehat{\mathbf{A}}^{T} \circ \hat{\boldsymbol{r}}) + \boldsymbol{e_1} \in \mathcal{R}_q^{k\times 1}$

$\boldsymbol{v} = \mathbf{NTT}^{-1}(\hat{\mathbf{t}}^{T} \circ \hat{\boldsymbol{r}}) + \mathbf{Decompress}_q(\mathbf{Decode}_1(\boldsymbol{m}), 1) + \boldsymbol{e_2} \in \mathcal{R}_q$

$\boldsymbol{c_1} = \mathbf{Encode}_{d_u}(\mathbf{Compress}_q(\boldsymbol{u}, \boldsymbol{d_u}))$

$\boldsymbol{c_2} = \mathbf{Encode}_{d_v}(\mathbf{Compress}_q(\boldsymbol{v}, \boldsymbol{d_v}))$

return $\boldsymbol{c} := (\boldsymbol{c_1}||\boldsymbol{c_2})$

Algorithm 2 ML-KEM.CCAKEM.Dec$(\boldsymbol{c}, \boldsymbol{sk_{KEM}})$ (Simplified)

$\boldsymbol{m'} =$ **ML-KEM.CPAPKE.Dec**$(\boldsymbol{sk}, \boldsymbol{c})$

$(\bar{\boldsymbol{K}}', \boldsymbol{r}') = \mathbf{G}(\boldsymbol{m}'||\mathbf{H}(\boldsymbol{pk}))$

$\boldsymbol{c}' =$ **ML-KEM.CPAPKE.Enc**$(\boldsymbol{pk}, \boldsymbol{m}', \boldsymbol{r}')$

if $\boldsymbol{c} = \boldsymbol{c}'$ **then**

return $\boldsymbol{K} = \mathbf{KDF}(\bar{\boldsymbol{K}}||\boldsymbol{H}(\boldsymbol{c}))$

else

return $\boldsymbol{K} = \mathbf{KDF}(\boldsymbol{z}||\boldsymbol{H}(\boldsymbol{c}))$

end if

ML-KEM 定義了三種參數組態，分別為 ML-KEM-512、ML-KEM-768 與 ML-KEM-1024，其安全強度分別對應至 AES-128、AES-192 以及 AES-256 的等級。三種組態均基於相同的多項式環，如公式 (1) 所示。

$$R_q = \frac{\mathbb{Z}_q[X]}{(X^{256}+1)} \tag{1}$$

ML-KEM 在不同安全等級下，其金鑰與密文的長度各不相同。對於 ML-KEM-512，封裝金鑰長度為 800 Byte，解封裝金鑰長度為 1632 Byte，密文長度為 768 Byte，而共享

金鑰則為固定的 32 Byte。在 ML-KEM-768 的設定中，封裝金鑰長度提升至 1184 Byte，解封裝金鑰長度為 2400 Byte，密文長度為 1088 Byte，至於共享金鑰仍維持 32 Byte。不僅如此，在最高安全等級 ML-KEM-1024 中，封裝金鑰與解封裝金鑰的長度分別為 1568 Byte 與 3168 Byte，密文長度同樣為 1568 Byte，而共享金鑰依舊保持 32 Byte。

2.2 旁通道分析 (Side-channel Analysis, SCA)

SCA 是一種利用密碼演算法在加密設備上執行的過程中，加密設備不經意洩漏出一些物理特徵，例如：執行時間、功率消耗、電磁輻射、溫度和聲音，進而破解出敏感資訊。不同於傳統的密碼分析，SCA 之威脅在於不論密碼演算法的理論安全性有多麼堅固，實際的硬體實作時都有可能成為攻擊的弱點，此外我們堅信加密設備在執行不同的指令和不同的資料時，所消耗的功率會有所不同，攻擊者藉由加密設備不經意洩漏出的旁通道資訊進行分析，來破解密碼演算法的金鑰或加密訊息。

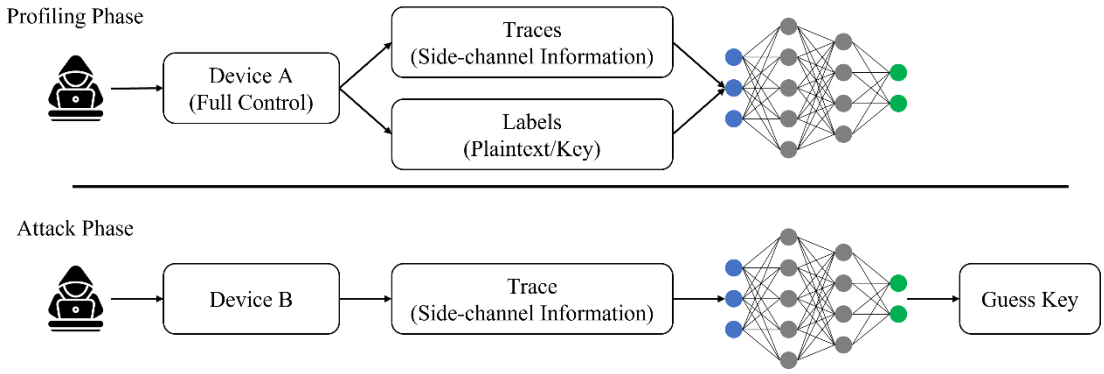
2.3 配置攻擊(Profiling Attack)

在 SCA 領域中，配置攻擊普遍被認為是最具威脅性的攻擊手法之一。其核心概念在於，攻擊者首先於一台可完全掌控的同型號裝置上蒐集資料並建立統計模型，隨後將該模型應用至目標裝置，以推斷敏感資訊，如明文或金鑰。

典型的配置攻擊可分為兩類。第一類為模板攻擊 (Template Attack, TA)，其方法是針對與金鑰相關的中間值建模，並假設其分布符合高斯分布，以此進行後續推測。第二類則是基於深度學習的配置攻擊，此方法以多層感知器 (Multilayer Perceptron, MLP)、卷積神經網路 (Convolutional Neural Network, CNN) [10] 等模型取代傳統的模板攻擊。深度學習方法能夠透過大量的旁通道資訊自動學習洩漏特徵，通常在攻擊效果上優於傳統的模板攻擊。

配置攻擊通常可劃分為兩個階段。第一階段為配置階段。假設攻擊者的目標為裝置 B，若攻擊者持有一台與目標裝置 B 相同或相似的裝置 A，並對其擁有完全控制權限，即可自由設定明文與金鑰以執行密碼演算法。在此過程中，攻擊者可量測裝置 A 在運行演算法時所洩漏的物理特徵，並蒐集大量特徵與對應標籤，用於訓練深度學習模型。

第二階段為攻擊階段。在此階段中，攻擊者僅需對目標裝置 B 執行演算法時所洩漏的物理特徵進行量測，並將所獲得的特徵輸入先前於配置階段訓練完成的模型，即可推斷出裝置 B 的敏感資訊，例如金鑰如圖一所示。



圖一：基於深度學習配置攻擊示意圖

在 SCA 的研究中，常用且重要的攻擊評估指標包括金鑰排名 (Key Rank)、猜測熵 (Guessing Entropy, GE) 以及成功率 (Success Rate, SR) [11]，用以量化攻擊效率與防禦效果。

在過去針對進階加密標準 [12] 的 SCA 研究中，常以第一回合 SubByte 的輸出值作為分析對象 [13]。該輸出值由隨機明文與固定金鑰計算而得，並透過 GE 或 SR 等指標來評估攻擊效能。然而，此方法的前提在於金鑰必須固定，方能進行有效分析。

相較之下，本研究所使用的攻擊 Trace 對應於 32 Bytes 的隨機訊息，並將其劃分為 32 個獨立片段進行分析。每一個 Trace 片段對應一個標籤，且其主要包含來自微控制器執行過程中的雜訊干擾。在模型具備足夠表現力的前提下，攻擊可達到相當高的準確率。

此外，由於 GE 與 SR 的傳統定義皆需要固定金鑰或訊息，因此本研究不採用 GE 作為評估指標，而是引入兩種改良的 SR 定義：

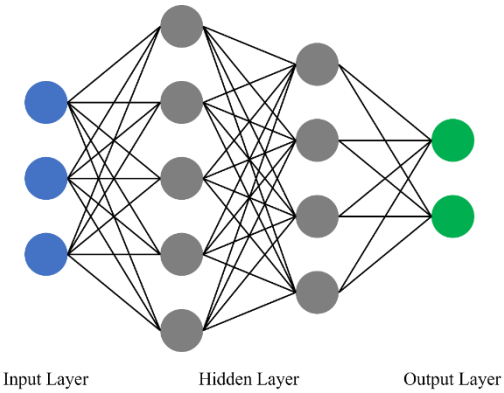
1. Byte-level Success Rate：衡量在所有攻擊 Byte 中，能成功預測正確 Byte 的比例，如公式(2)所示。
2. Message-level Success Rate：衡量在 32 Bytes 的隨機訊息中，能完整破解整條訊息的比例，如公式(3)所示。

其中，T 表示攻擊所使用的 Trace 條數，B 表示單條 Trace 所包含的 Byte 數(此處為 32)，$y'_{i,j}$ 為第 i 條 Trace 中第 j 個 Byte 的預測值，而 $y_{i,j}$ 為對應的正確 Byte 值。

2.4 多層感知器(Multilayer Perceptron, MLP)

多層感知器 (Multilayer Perceptron, MLP) 是一種前饋神經網路 (Feedforward Neural Network, FNN)，亦屬於人工神經網路 (Artificial Neural Network, ANN) 的基本形式之一，廣泛應用於分類、回歸等任務。

MLP 主要由以下幾個部分構成，**輸入層 (Input Layer)**：負責接收外部資料並傳遞至隱藏層，輸入層中的每個神經元對應輸入特徵的一個維度，在本研究中，輸入層即為功率軌跡 (Power Trace)。**隱藏層 (Hidden Layer)**：位於輸入層與輸出層之間，是 MLP 的核心部分，負責特徵提取與模式識別。每個隱藏層包含若干神經元，其數量與層數依具體應用與資料特性而定，隱藏層之間的非線性轉換使得 MLP 能處理複雜的非線性問題，隱藏層可為單層或多層。**輸出層 (Output Layer)**：生成最終預測結果，節點數取決於任務類型，例如分類問題的類別數或回歸問題的目標數。在本研究中，輸出層為預測秘密訊息單一 Byte 的機率，共計 256 類。綜上所述，多層感知器由上述三個部分組成，各層皆由若干神經元構成，且每層神經元與下一層神經元完全連接，如圖二所示。



圖二：MLP 示意圖

參、研究方法

3.1 攻擊分析 (Attack Analysis)

本研究採用 PQClean 開源函式庫 [14] 中的 ML-KEM 演算法。PQClean 提供了乾淨、具可攜性且經過驗證的版本。在 Kyber 尚未被正式標準化之前，多篇研究已指出 Kyber 演算法中 **poly.c** 檔案的 poly_frommsg 函式存在側信道攻擊風險，可能導致封裝訊息的洩漏。該函式會將 32 Bytes 的隨機訊息依序編碼為 256 維度的多項式，用於後續運算。

依據程式邏輯，函式會逐一處理 32 Bytes 隨機訊息中的每個 Bit：若 Bit = 0，則變數 mask 取值 0x0000；若 Bit = 1，則 mask 取值 0xFFFF。由於兩種情況下 mask 的漢明權重 (Hamming Weight, HW) 差異極大，且 HW 定義為二進位表示中 1 的個數，而 HW 與實際功率消耗具相關性，因此攻擊者可藉由功率消耗分析輕易還原隨機訊息的所有位元。

在 NIST 正式將 ML-KEM 納入金鑰封裝機制標準後，poly_frommsg 函式已經過修改。更新版本的設計可避免編譯器最佳化導致的分支，使得無論輸入 Bit 為 0 或 1，執行時間皆相同，從而有效抵禦時序攻擊 (Timing Attack)。

然而，本研究發現參數 b 的運算邏輯與原始變數 mask 相同，仍會產生明顯的 HW 差異：當 b = 0 時，暫存器值為 0x0000，其 HW = 0；當 b = 1 時，暫存器值為 0xFFFF，其 HW = 16。本研究僅將其劃分為 HW = 0 與 HW = 16 兩類，未使用中間 HW 值 (1–15)。儘管如此，HW 差異仍可能被攻擊者利用，以還原 32 Bytes 的隨機訊息。因此，本研究持續選擇 poly_frommsg 函式作為攻擊目標。

本研究採用 PQClean 開源函式庫 [14] 中的 ML-KEM 演算法。該函式庫提供經社群驗證、具可移植性且維護良好的後量子密碼實作版本。在 Kyber 尚未正式標準化之前，多篇研究已指出其程式碼中 **poly.c** 的 poly_frommsg 存在潛在的旁通道攻擊 (Side-Channel Attack, SCA) 風險。該函式將 32 字節的隨機訊息映射為 256 維的多項式表示，此過程若未採取適當的時間或功率均衡措施，可能導致敏感資料在實體層級遭到洩漏。此一特性成為攻擊者利用深度學習等現代技術進行封裝訊息重建的攻擊向量之一。

根據程式邏輯，函式會逐一處理 32 Bytes 隨機訊息中的每個 Bit：若 Bit = 0，則變數 mask 取值 0x0000；若 Bit = 1，則 mask 取值 0xFFFF。由於兩種情況下 mask 的 HW 差異極大，而 HW 定義為二進位表示中 1 的個數，且與實際功率消耗具相關性，因此攻擊者可透過功率消耗分析還原隨機訊息的所有位元。

在 NIST 將 ML-KEM 納入金鑰封裝機制標準後，poly_frommsg 函式已被修改。更新版本避免了編譯器最佳化所導致的分支，使得輸入 Bit 為 0 或 1 時執行時間相同，能有效抵禦時序攻擊。

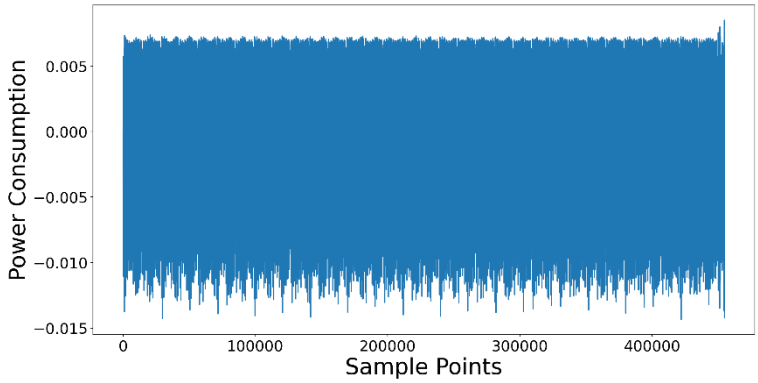
然而，本研究發現參數 b 的運算邏輯與原始變數 mask 相同，仍會產生明顯的 HW 差異：當 b = 0 時，暫存器值為 0x0000，其 HW = 0；當 b = 1 時，暫存器值為 0xFFFF，其 HW = 16。本研究僅將其劃分為 HW = 0 與 HW = 16 兩類，未考慮中間 HW 值 (1–15)。儘管如此，這樣的 HW 差異仍可能被攻擊者利用，以還原 32 Bytes 的隨機訊息。因此，本研究持續將 poly_frommsg 函式作為攻擊目標。

由於 ML-KEM 所封裝的是 32 Bytes 的隨機訊息，而非後續使用的對稱式會話金鑰 (Session Key)，本研究的目標在於破解該 32 Bytes 的隨機訊息。然而，一旦隨機訊息被成功破解，仍可藉由公鑰 (Public Key, PK)、密文，以及公開的雜湊函數，進一步推導出對應的 Session Key。

3.2 紀錄功率軌跡

進行 SCA 的第一步是執行多次目標演算法，以收集大量功率軌跡作為後續分析的基礎。本研究使用 ChipWhisperer CW308 板上的 STM32F415 MCU，紀錄其執行目標

演算法時的功率消耗。由於 CW-lite 的 ADC 上限僅能記錄 24,400 個取樣點，導致量測到的功率軌跡存在失真。為了更貼近實際情境，本研究外接 Picoscope5244B 示波器以量取功率軌跡。然而，STM32F415 MCU 內建的低壓差線性穩壓器 (Low-dropout Regulator, LDO) 會引入過多雜訊，影響功率軌跡的品質。為了改善此問題，本研究利用 Picoscope5244B 內建的 20 MHz 硬體低通濾波器，有效濾除高頻雜訊。最終記錄到的單條功率軌跡包含 455,000 個取樣點。此外，為了確認正確量測到 poly_frommsg 函式的功率消耗，本研究傳送 2000 條 32 Bytes 封裝的隨機訊息。將功率軌跡取平均後，可以觀察到 32 個 peak，對應於目標演算法外迴圈的 32 次執行，如圖四所示。

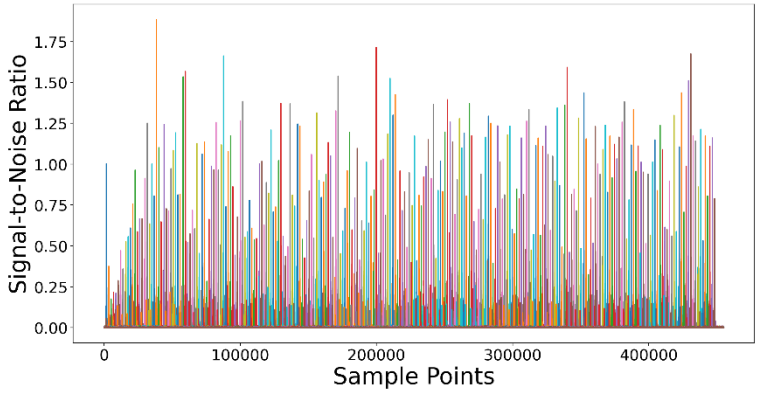


圖三：2000 條功率軌跡之平均

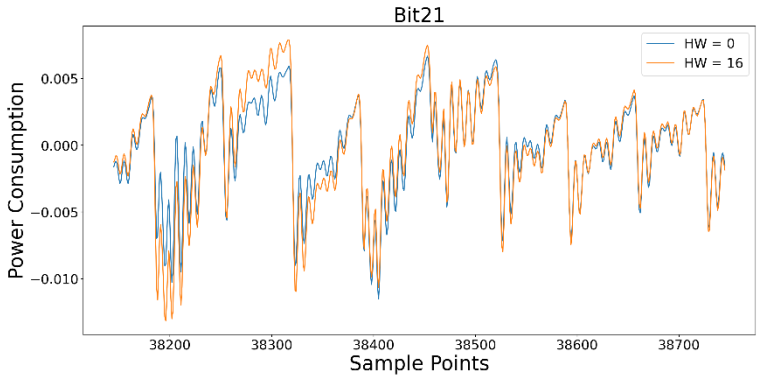
此外，本研究亦計算 32 Bytes 隨機訊息在 2000 條功率軌跡下的訊號雜訊比 (Signal-to-Noise Ratio, SNR)，以協助快速定位功率軌跡中敏感資訊洩漏的時間點，如公式(2)所示。

$$\mathrm{SNR} = \frac{\mathrm{Var}\left[\mathrm{E}[L|X]\right]}{\mathrm{E}\left[\mathrm{Var}[L|X]\right]} \tag{2}$$

由於本研究所攻擊的演算法是逐位元對 32 Bytes 隨機訊息中的所有位元進行操作，所以可以看到 SNR 總共有 256 個峰值 (Peak)，該峰值即為最具敏感之位置，如圖四所示。此外，由章節 3.1 描述，只會分成 HW 為 0 和 HW 為 16 兩群，在第 21 個位元上具有最高的 SNR 之值，將分群後的平均軌跡在最高 SNR 之位置的前後 300 個取樣點放大來看，可以看到 HW 為 0 和 HW 為 16 也明顯被分開，如圖五所示。



圖四：256 Bite 之訊號雜訊比

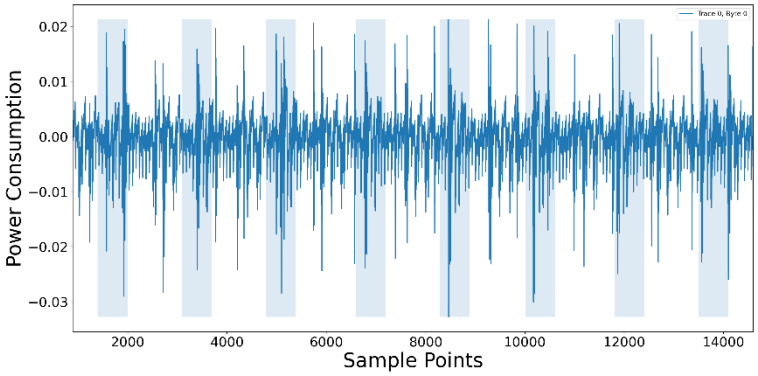


圖五：第 21 位元群分圖

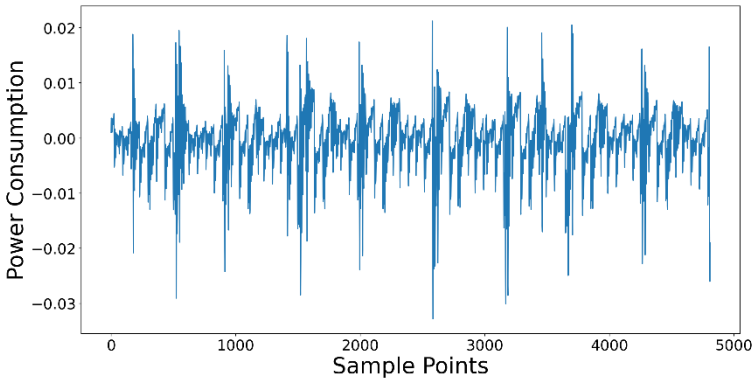
3.3 功率軌跡前處理

本研究記錄到的功率軌跡包含完整的 32 Bytes 的隨機訊息，採用逐位元組的方式進行破解，將功率軌跡拆成 32 等分，每一個 Byte 對應一個片段功率軌跡，並將所有切割後的片段功率軌跡和對應的 Byte 作為模型的資料與標籤，透過 Byte-by-byte 的方式逐一破解 32 Bytes 之隨機訊息。

本研究發現越後面的片段功率軌跡偏移越多才會有所對齊，但本研究不採用直接切割之預處理方法，因此不用考慮偏移之處理。在章節 3.2 中，本研究計算了功率軌跡集合之 SNR，找到最具洩漏敏感特徵的興趣點 (Point of Interest, PoI) 位置，取出 PoI 前後 300 個取樣點當作是該位元興趣點窗口 (PoI Window)，頭尾的分群會越來越不明顯，因此本研究認為再多的取樣點個數，例如 350 或 400，對訓練效果並沒有額外的幫助，也會提升模型的訓練負擔。計算出每個位元的 PoI Window 後，將每八個 Window 進行串接，作為每一個 Byte 的訓練資料集，相比直接進行分割，此方法更有效降低模型輸入的維度和訓練成本，PoI Window 的範圍與串接後的結果如圖六和圖七所示。



圖六：原始功率軌跡第一個 Byte 之 PoI Window



圖七：第一個 Byte 之 PoI Window 串接結果

此外，本研究對 ML-KEM 所封裝的 32 Bytes 隨機訊息進行還原，且採用 Byte-by-byte 的方式，對於運算有限的情境中，也可以進行 Bit-by-bit 的方式對 256 Bits 之隨機訊息進行還原，採用 Bit-by-bit 可以有效的降低模型輸入向量的維度，輸出分類結果也只有 0 或 1 兩種可能，有效降低模型的複雜程度，且也只需重複做 256 次亦即可以還原 32 Bytes 之隨機訊息。

3.4 模型訓練

在本節中，介紹深度學習模型的相關超參數設定和架構設計，本研究採用 MLP 作為分類模型，模型輸入層神經元數量由章節 3.3 中所選取的 PoI Window 決定，將 8 個 Bits 之 PoI Window 進行串接後構成長度為 4808 的輸入向量。為了探索最佳的網路架構與超參數配置，本研究採用隨機搜尋 (Random Search) 之方法，針對各項超參數進行搜尋與組合實驗，搜尋範圍涵蓋層數、各層神經元數量、激活函數、損失函數、Dropout Rate、優化器、批次大小、訓練輪數 (Epoch) 和學習率，並以驗證準確率 (Validation Accuracy) 作為模型優劣的評估指標。訓練結果顯示，在輸入層後接兩個隱藏層，維度

分別為 2048 和 512 具有最佳的表現，而輸出層神經元個數為 256，以對應破解單個 Byte 之 256 種可能值。在神經元激活函數方面，對比 ReLU 和 GeLU 後，實驗結果顯示，對於輸入層和隱藏層使用 ReLU 作為激活函數最效果最佳，輸出層則使用 Softmax 以產生機率分布並選取最大機率對應預測值。損失函數採用交叉熵 (Cross Entropy)，因其適合多類別分類問題。優化器選擇 Adam，並設定學習率為0.0001。為了減少過擬合，本研究在每個隱藏層後加入 BatchNormalization 正規化和 Dropout，並監控驗證集的損失與準確率。此外，實驗中比較了不同 Epoch 數，結果顯示 150 個 Epochs 足以收斂並獲得穩定且高的驗證準確度，同時避免過多 Epoch 帶來的過擬合風險，其他超參數與搜尋範圍如表一所示，最終模型架構與參數量如表二所示。

表一：模型超參數表

Hyperparameter	**Attempted Value**	**Optimal Value**
Number of Layers	[3, 4]	4
Outputs Size of Layers	[512, 1024, 2048]	[4808, 2048, 512, 256]
Activation Function	[ReLU, GeLU]	ReLU(Other Layers) Softmax(Output Layer)
Loss Function	[Crossentropy, MSE]	Crossentropy
Optimizer	[Adam, RMSprop]	Adam
Epochs	[100, 150, 300, 1000]	150

表二：MLP 模型架構

Layer	Output Shape	Parameter
dense	(None, 2048)	9,846,784
batch_normalization	(None, 2048)	8,192
activation	(None, 2048)	0
dropout	(None, 2048)	0
dense_1	(None, 512)	1,048,576
batch_normalization_1	(None, 512)	2,048
activation_1	(None, 512)	0
dropout_1	(None, 512)	0
dense_2	(None, 256)	131,328

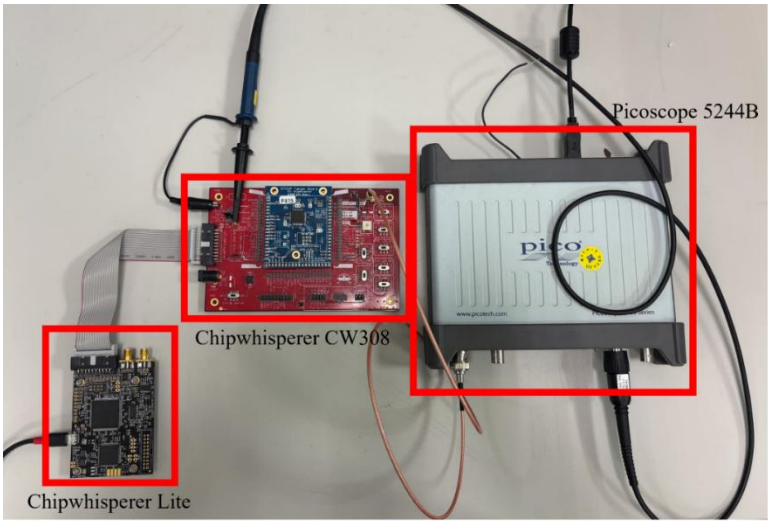
肆、實驗結果

4.1 實驗環境配置

本研究的實驗環境與系統架構配置如下。深度學習實驗所使用的軟體環境包括 CUDA 12.3、TensorFlow 2.16.1 與 Python 3.10.15，以確保圖形處理單元 (Graphics Processing Unit, GPU) 運算的相容性與穩定性。嵌入式目標裝置為 STM32F415 MCU，其時脈頻率設定為 7.37 MHz，並採用 -O3 編譯最佳化以模擬最接近實際應用的情境。硬體設備方面，實驗平台由控制主機(虛擬機)、GPU 運算伺服器、Picoscope 5244B 示波器以及 ChipWhisperer 平台組成。其中，控制主機 (Controller PC, Virtual Machine) 運行 Ubuntu 22.04.4 LTS 作業系統，搭載 Intel(R) Core(TM) i7-12700 處理器，並配置 20 GB DDR5 4800 MHz 記憶體。

為提升模型訓練與攻擊效率，本研究所使用之 GPU 運算伺服器搭載 Ubuntu 22.04.4 LTS 作業系統，硬體配置包含 Intel(R) Core(TM) i7-13700 處理器、雙通道 DDR4 3200 MHz 32 GB 記憶體模組，以及 NVIDIA GeForce RTX 3090 Ti 顯示卡，具備強大之深度學習運算能力。在資料收集階段，實驗流程如圖八所示，先由主控電腦 (Controller PC) 透過 UART 通訊傳送大量隨機產生之 32 Bytes 封裝訊息至 ChipWhisperer CW308 平台所搭載之 STM32F415-RGT6 微控制器。該 MCU 運行經

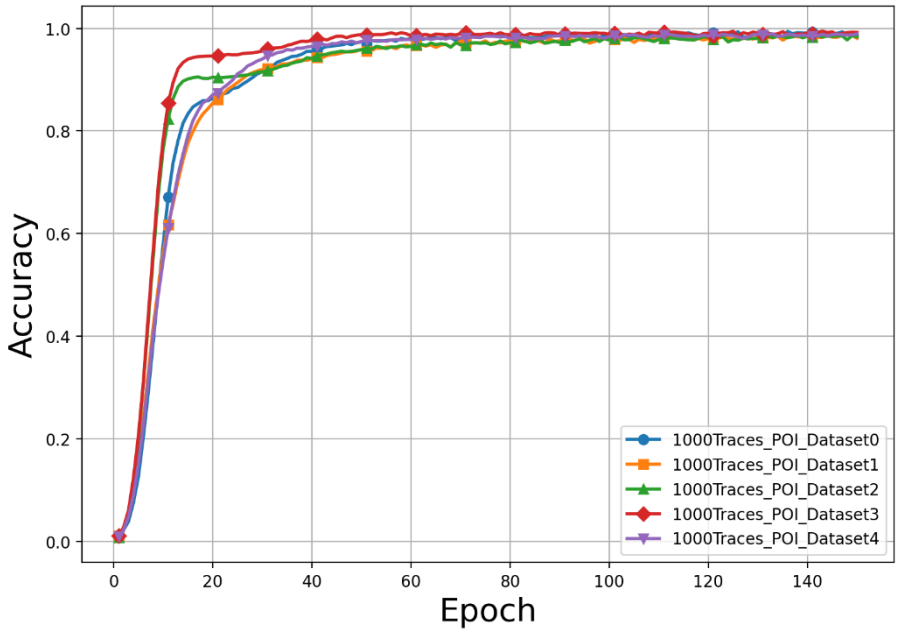
ML-KEM 封裝演算法，模擬後量子密碼機制於物聯網裝置中的實際應用場景。於演算法執行期間，藉由 Picoscope 5244B 高解析示波器進行即時功率軌跡量測，並同步擷取目標裝置之功率消耗波形。所有功率軌跡資料經擷取後，回傳並儲存至主控電腦，隨後上傳至 GPU 運算伺服器，以供本研究所設計之深度學習型旁通道攻擊模型進行訓練與驗證。整體流程確保數據來源準確且具時序一致性，有助於建立可遷移之攻擊模型。



圖八：SCA 架構圖

4.2 功率軌跡集合選擇

本章節展示對 STM32F415MCU 使用本論文提出的方法進行模型的訓練，圖九為模型訓練的結果，詳細數據統整於表三。



圖九：1000 條功率軌跡訓練成果

表三：STM32F415 各組資料集訓練結果

Number of Traces	1000 Traces
Dataset0	99.41%
Dataset1	98.87%
Dataset2	98.64%
Dataset3	99.45%
Dataset4	98.94%

4.3 模型訓練時間比較

本章節統計並比較了使用中央處理單元 (Central Processing Unit, CPU) 和 GPU 的模型訓練時間，如表四所示。從實驗結果中發現，對於運算資源不夠的環境中，我們的模型也可以使用 CPU 來完成訓練，訓練時間也在可以接受的範圍內，有效的實施 SCA。此外，對於章節 3.3 中的描述，本研究採用 Byte-by-byte 的方式對 32 Bytes 的隨機訊息進行還原，但如果採用 Bit-by-bit 的方式對 32 Bytes 中的每一個位元進行還原，其模型參數與架構會更簡易，更有較短的訓練時間。

表四：STM32F415 各組資料集訓練時間比較表 (mins)

Number of Traces	1000 Traces	
Platform	CPU	GPU
Dataset0	12.31	1.43
Dataset1	12.3	1.37
Dataset2	12.22	1.38
Dataset3	12.25	1.34
Dataset4	12.28	1.37

4.4 攻擊成果

在攻擊的情境中，假設攻擊者欲想破解封裝之 32 Bytes 隨機訊息，無法預先得知封裝之隨機訊息，因此不可直接從攻擊軌跡中準確定位 PoI 位置。基於此限制，採用 Profiling 設備，標記為 D0，上蒐集之 2000 條訓練功率軌跡，透過此資料集確定 PoI 位置對所有攻擊軌跡進行分割，並將分割後之片段功率軌跡輸入已訓練模型以進行訊息還原，如表五所示。

表五：設備之攻擊結果

Device	Type	Correct Count	Total Count	Success Rate
D0 (Profiling)	Byte	15997	16000	99.9813%
	Message	497	500	99.4%

伍、結論

本研究針對後量子密碼學金鑰封裝機制標準 ML-KEM 在嵌入式裝置上的實作安全性進行深入探討，選用 Cortex-M4 架構之 STM32F415 微控制器為目標平台，實施旁通道分析 (Side-Channel Analysis, SCA) 並結合深度學習模型以進行攻擊與評估。實驗結果顯示，透過所提出之深度學習架構進行模型訓練後，可達成 100% 的驗證準確率，以及高達 99.98% 的攻擊準確率，顯示其對於 ML-KEM 封裝階段中所產生的隨機訊息具有極高的還原能力。綜合實驗結果可知，ML-KEM 在現行嵌入式平台上的實作仍存在旁通道資訊洩漏的潛在風險。未來研究可進一步拓展至不同微控制器架構 (如 STM32F303、STM32F407 等)，以驗證攻擊模型之跨平台遷移能力與泛化性能，並進一步促進針對後量子演算法的實作安全性設計與防禦機制之發展。

參考文獻

[1] Statista, Number of IoT connections worldwide 2022-2033, with forecasts from 2024 to 2033.
Available:https://www.statista.com/statistics/1183457/iot-connected-devices-worldwide/ (2024/6/12).

[2] P. W. Shor, "Algorithms for quantum computation: discrete logarithms and factoring," *Proceedings 35th Annual Symposium on Foundations of Computer Science*, pp. 124-134, Nov. 1994.

[3] R. Avanzi, J. Bos, L. Ducas, E. Kiltz, T. Lepoint, V. Lyubashevsky, J. M. Schanck, P. Schwabe, G. Seiler, and D. Stehlé, "CRYSTALS-Kyber (version 3.02) – Submission to round 3 of the NIST post-quantum project," Aug. 2021. [Online] Available:https://pq-crystals.org/kyber/data/kyber-specification-round3-20210804.pdf

[4] National Institute of Standards and Technology, "Module-Lattice-Based Key-Encapsulation Mechanism Standard," Aug. 2024. [Online] Available: https://nvlpubs.nist.gov/nistpubs/FIPS/NIST.FIPS.203.ipd.pdf

[5] P. C. Kocher, "Timing Attacks on Implementations of Diffie-Hellman, RSA, DSS, and Other Systems," *16th Annual International Cryptology Conference*, pp.104-113, Aug. 1996.

[6] S. Mangard, E. Oswald, and T. Popp, "Power Analysis Attacks: Revealing the Secrets of Smart Cards," *Springer*, 2007, https://doi.org/10.1007/978-0-387-38162-6.

[7] C.W. Kuo, C.C. Lin, Y.Y. Hong, J.R. Liu, C.H. Yeh, and K.Y. Tsai, "Research and Analysis of the Effects of Different Shielding Materials on Resisting Side-Channel Attacks on IoT Device Microcontroller," *2024 8th International Conference on Cryptography, Security and Privacy (CSP)*, pp. 84-88, Apr. 2024.

[8] A. Aljuffri, M. Zwalua, C. R. W. Reinbrecht, S. Hamdioui, and M. Taouil, "Applying Thermal Side-Channel Attacks on Asymmetric Cryptography," *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, Vol. 29, No. 11, pp.1930-1942, Nov. 2021.

[9] E. Alkim, L. Ducas, T. Pöppelmann, and P. Schwabe, "Post-quantum key exchange: a new hope," *SEC'16: Proceedings of the 25th USENIX Conference on Security Symposium*, pp. 327-343, Aug. 2016.

[10] H. W. Feng, W.G. Lin, and W.Q. Shang, "MLP and CNN-based Classification for Points of Interest in Side-Channel Attacks," *2019 IEEE/ACIS 18th International Conference on Computer and Information Science (ICIS)*, pp. 456-460, Dec. 2019.

[11] F. X. Standaert, T. Malkin, and M. Yung, "A Unified Framework for the Analysis of Side-Channel Key Recovery Attacks," *Proceedings of the 28th Annual International*

Conference on Advances in Cryptology - EUROCRYPT 2009, Vol. 5479, pp. 443-461, Apr. 2009.

[12] National Institute of Standards and Technology, "Advanced Encryption Standard (AES)," Nov. 2001. [Online] Available: https://nvlpubs.nist.gov/nistpubs/FIPS/NIST.FIPS.197.pdf

[13] S. Mangard, E. Oswald, and T. Popp, "Power Analysis Attacks: Revealing the Secrets of Smart Cards," *Springer Publishing Company*, 2007. [Online] Available: https://link.springer.com/book/10.1007/978-0-387-38162-6

[14] M. J. Kannwischer, P. Schwabe, D. Stebila, and T. Wiggers, "Improving Software Quality in Cryptography Standardization Projects," *IACR Cryptology ePrint Archive*, Apr. 2022. https://eprint.iacr.org/2022/337