

後量子時代的傳輸安全： 整合 ML-KEM 混合式金鑰封裝之實作與效能評估

廖威宥^{1,2,4}、王翊全^{1,3,4}、林昀毅^{1,2,4}、杜伯庸⁴、曾柏嚴^{1,4}、邱顯泰^{1,4,*}



¹國立成功大學化學系，台南市，台灣

²國立成功大學機械工程系，台南市，台灣

³國立成功大學工業與資訊管理學系，台南市，台灣

⁴台灣生技藥研團隊，台灣

*通訊作者與Email：hchiu@mail.ncku.edu.tw



摘要

大型量子電腦一旦可行，Shor 演算法將使目前廣泛使用的 ECDH (Elliptic Curve Diffie-Hellman，橢圓曲線迪菲-赫爾曼)、RSA 等金鑰交換機制失效。NIST (National Institute of Standards and Technology，美國國家標準與技術研究院) 已於 2024 年發布 FIPS 203，將基於模格 (module-lattice) 的 ML-KEM (Module-Lattice-Based Key-Encapsulation Mechanism，模塊格基金鑰封裝機制) 標準化為後量子金鑰封裝機制，並在 NIST IR 8547 (美國國家標準與技術研究院內部報告第 8547 號) 中規劃 2030 年起逐步淘汰傳統公鑰演算法。然而，實際系統該如何遷移、遷移後付出多少效能代價，仍缺乏貼近真實部署環境的量測資料。台灣生技藥研團隊在本研究上於 TLS 1.3 通道導入 ML-KEM-768 與 X25519 組合之混合式金鑰交換 (X25519MLKEM768)，在保留傳統演算法安全性下限的前提下取得抗量子能力；於 Kubernetes Ingress 層完成密碼套件與金鑰交換群組設定，建構一套完整的抗量子防護措施保證資料安全性，並探討握手延遲 (TLS handshake latency)、RTT (Round-Trip Time)、握手封包大小等指標進行驗證。由於 ML-KEM-768 的封裝金鑰為 1184 位元組、密文為 1088 位元組，遠大於 X25519 的 32 位元組，ClientHello 將超出單一 MTU (Maximum Transmission Unit，最大傳輸單元)，是否因此產生額外往返為本團隊研究的關注重點之一。

研究動機與研究問題

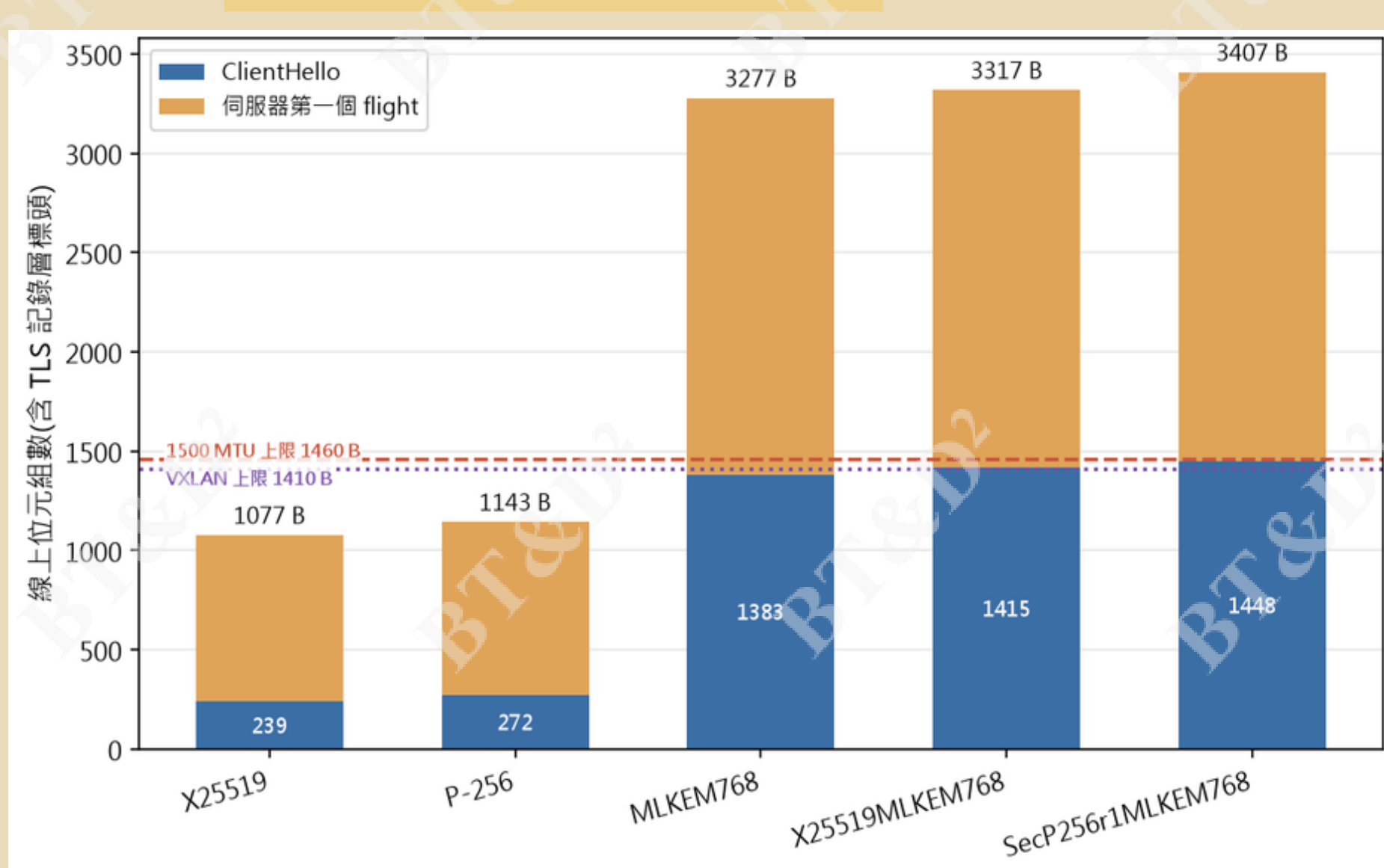
後量子遷移的進度呈現明顯不對稱。依 Cloudflare Radar 觀測，客戶端已有逾六成的人類 TLS 流量採用混合式金鑰協商 X25519MLKEM768 伺服器端卻僅約一成的來源主機支援。由於 TLS 1.3 的金鑰協商須雙方同時支援方能生效；瀏覽器早已就緒，連線卻因伺服器未跟進而退回傳統 ECDH。落後主因不在演算法，而在部署鏈-TLS函式庫的版本門檻、反向代理及其相依元件的升級成本，以及缺乏公開效能參考。既有統計多來自 CDN 的大規模觀測，能回答「有多少人做了」，卻難以回答「做了要付出多少代價」，而後者才是維運者的決策問題。

本研究自建可控之 TLS 1.3 測試環境，於同一硬體與網路條件下分別以純 X25519 與 X25519MLKEM768 建立連線並對照量測，使差異可歸因於金鑰交換機制本身。研究關注三點：一為部署門檻，即以現行主流軟體堆疊啟用混合式金鑰交換所需的前置條件與實際升級障礙；二為效能代價，即握手延遲與封包大小的增幅；三為 MTU 效應，ML-KEM-768 封裝金鑰 1184 位元組、密文 1088 位元組，遠大於 X25519 的 32 位元組，ClientHello 將超出單一 MTU 而遭分片，本研究欲確認此分片是否造成額外往返，並在高延遲或封包遺失下被放大，以補足大規模觀測難以觸及的部分。

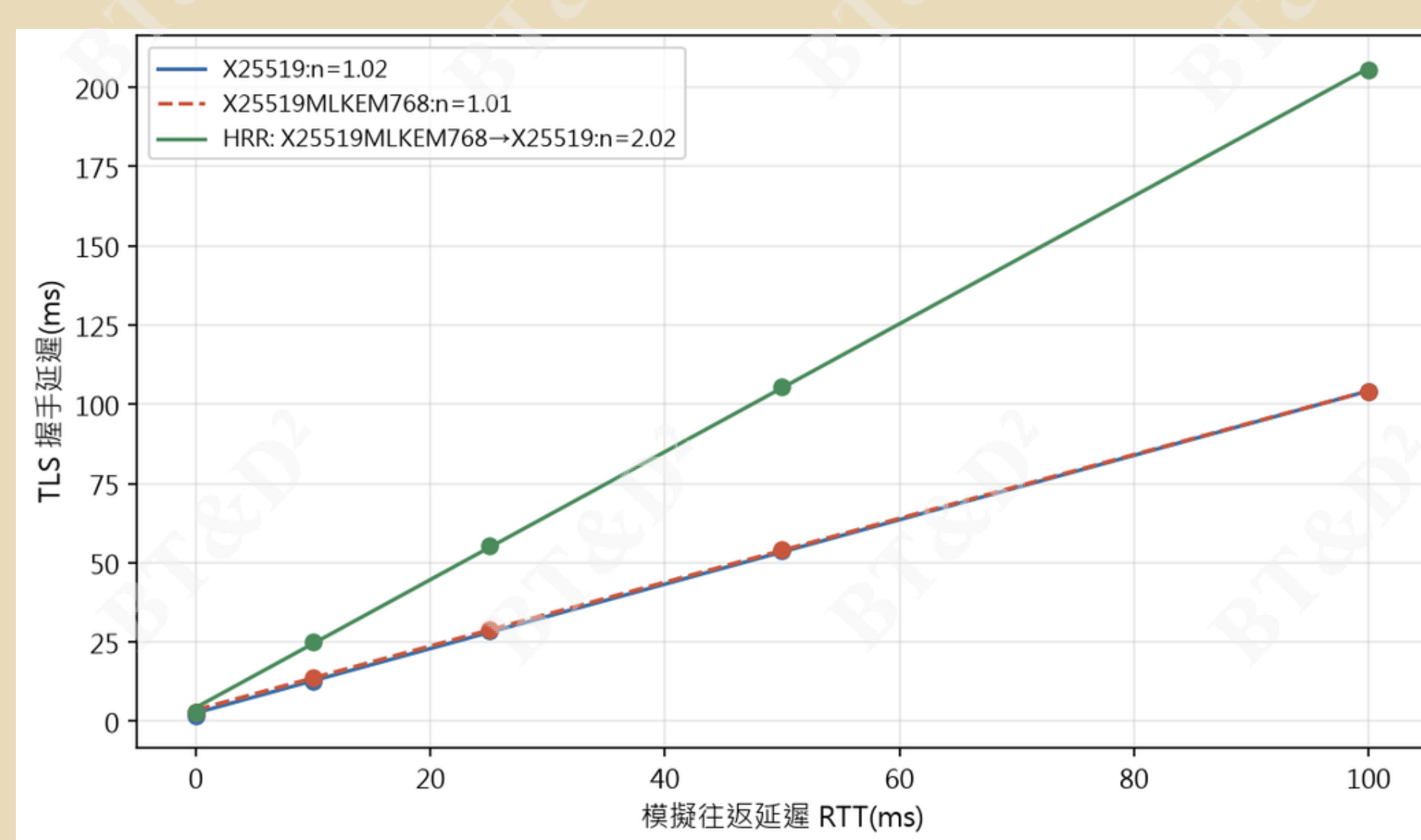
實驗方法

為了評估 TLS 1.3 導入 X25519MLKEM768 混合式金鑰交換之效能代價，我們結合 Kubernetes 叢集與自建 Python 量測工具進行實證分析。實驗運行於 Windows 11 之 loopback 網路，密碼學核心基於 OpenSSL 3.5，伺服器端配置 ECDSA P-256 憑證，且每組實驗皆取樣 1000 次以確保數據穩定性。在 Kubernetes Ingress 層的部署中，我們僅需將 ConfigMap 的 ssl-ecdh-curve 設定為指定群組，無須修改 nginx 原始碼，並刻意維持既有之 cert-manager 憑證流程，以避免憑證鏈過度膨脹而逼近初始壅塞視窗的上限。在效能與網路量測方面，為排除作業系統基準延遲與 TCP 底層行為的干擾，本研究採取多維度的隔離量測策略。我們透過自行開發具備 RTT 注入功能的 TCP 代理伺服器，以握手延遲對模擬 RTT 的線性迴歸斜率來精準量化協定實際消耗的往返次數。同時，我們實測各金鑰交換群組的 ClientHello 封包大小，探測其在標準 1500 MTU 與 Virtual Extensible LAN (MTU 1450) 環境下的封包切分臨界點。最後，為免去 loopback 網路約 1.5 ms 固定誤差的影響，本研究利用 openssl speed 隔離量測純演算法之運算成本，將其精確拆解為客戶端與伺服器端的微秒級花費，以呈現最客觀的演算法效能差異。

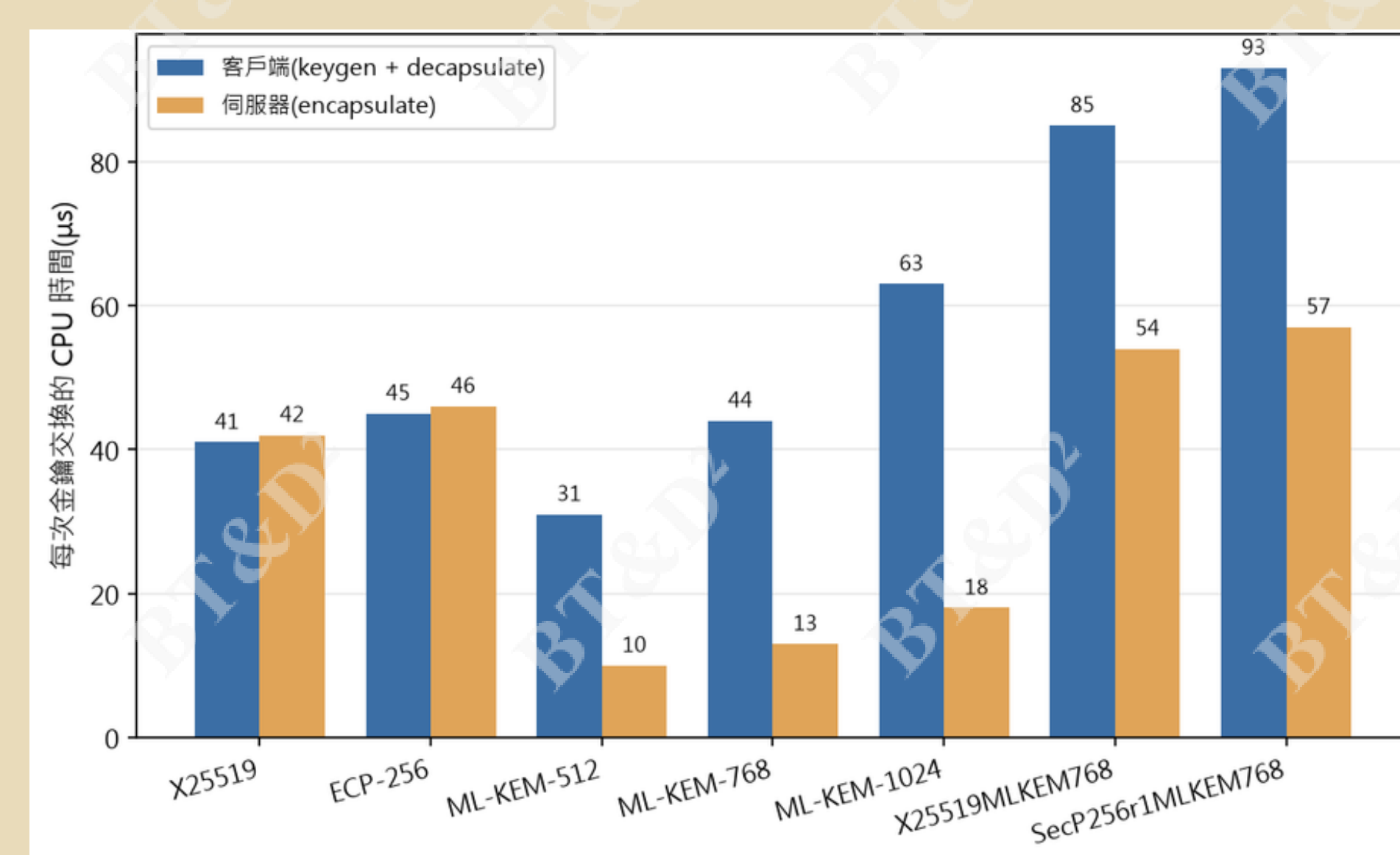
實驗結果



圖一：ClientHello 膨脹與路徑 MTU 界線



圖二：握手延遲對 RTT 的迴歸



圖三：金鑰交換的純運算成本 (不含網路 I/O)

實驗結論

封包膨脹: ML-KEM 的金鑰封裝體積使 ClientHello 逼近標準 1500 MTU 的上限，餘裕僅數十位元組。此一餘裕在真實環境中極易耗盡：稍長的 SNI，或 Kubernetes 叢集常見的 VXLAN overlay (MTU 1450)，都足以使 ClientHello 跨越單一封包。但分片本身未造成額外延遲，因此真正需要警惕的不是分片，而是餘裕消失後對路徑 MTU 異常，例如:PMTUD 黑洞 (Path MTU Discovery Black Hole，路徑最大傳輸單元發現黑洞)。

相容性會使往返次數增加:只要通訊雙方皆支援 X25519MLKEM768，握手仍維持在 1-RTT 的理想狀態；真正的效能風險出現在過渡期的不對稱組態——客戶端已提案 ML-KEM 而伺服器端 (如 Kubernetes Ingress) 尚未支援時，協商將觸發 HelloRetryRequest 而退化為 2-RTT。在高延遲鏈路上，這筆代價與 RTT 等比例放大：100 ms 的鏈路即多付出整整 100 ms。換言之，伺服器端「暫不升級」並非中性選擇，而是主動讓已就緒的客戶端承擔額外往返，這正是基礎設施端應優先佈署的理由。

運算成本的相對倍率: 混合式方案的 CPU 成本翻倍，源自必須同時執行兩種演算法的疊加，而非 ML-KEM 效率不佳；單獨量測時 ML-KEM-768 的運算時間與 X25519 相當甚至更低。



台灣生技藥研團隊
Taiwan BT&D² Team @ NCKU