國立交通大學資訊工程學系

資訊專題競賽報告

以P4交換機實作封包酬載的快速置換加密

（Enhancing 5G/IoT Transport Security Through Content Permutation）

專題題目說明、價值與貢獻自評（限100字內）：

此專題利用P4交換機可程式化的能力與管線架構，實作以線路速率重新排列IoT封包酬載的端到端加密編碼，進行最高6.4Tbps位元速率的快速加解密。在此我學習到如何進行嚴謹的演算法證明，並深入研究SDN與P4語言撰寫。

專題隊員：

學號	姓名	手機	E-mail	負責項目說明	專題內貢獻度(%)
0516230	黃則睿	0972623895	raymond210129.cs05@g2.nctu.edu.tw	題目構想、設計/操作實驗、撰寫論文主文	100%

本專題如有下列情況則請說明：

1.為累積之成果(含論文及專利)、2.有研究生參與提供成果、3.為大型研究之一部份。

發表論文：Lin, Y.B., Huang, T.J. and Tsai, S.C., 2019. Enhancing 5G/IoT Transport Security Through Content Permutation. IEEE Access, 7, pp.94293-94299.

相關研究生資料（無則免填）：

級別年級	姓名	提供之貢獻	專題內貢獻度(%)

【說明】上述二表格之專題內貢獻度累計需等於100%。

指導教授簡述及簡評：

黃則睿同學的專題研究已超過碩士研究論文的水準。他在P4 switch進行permutation cypher 及decypher的實作，巧妙應用P4 switch pipeline architecture特性，寫出來的P4程式碼連P4晶片原廠Barefoot (現屬Intel)都極為讚嘆。更難能可貴的是，他能利用數學歸納法，以嚴謹的理論方式證明所設計的P4程式是正確的。此部分已有博士生水準。本專題由黃則睿同學獨力完成，成果發表於IEEE Access,是全世界速度最快的switch permutation方法。 指導教授簽名：林一平

中　華　民　國　一　〇　八　年　十　一　月　六　日

一、 關鍵詞

第五代行動通訊(5G)、物聯網(IoT)、P4 語言、置換加密(Secret Permutation)、安全(Security)、軟體定義網路(Software Defined Networking)

二、 專題研究動機與目的

此專題嘗試利用 P4 語言技術,嘗試讓 P4 交換機將封包酬載(Payload)切割成小區段並重新排序的區段數最大化，使 P4 交換機得以進行金鑰長度夠長的加密編碼，並應用於 5G 核心網路的封包傳輸。

三、 現有相關研究概況及比較

目前有一些針對 5G 安全的相關研究。M. A. Ferrag[6]研究對於 5G 對於安全性應做的考量；A. R. Prasad[7]研究 5G 遭受中間人攻擊(man-in-the-middle)、降級攻擊(bidding down)、重送攻擊(replay attack)、攻擊控制平面(control plane)與資料平面(data plane)的可能性；D. Basin[8]針對 5G 驗證進行嚴謹分析；R. P. Jover[9]則認為 5G 標準制定時考慮了不會發生的狀況,在沒有強制啟用特定安全機制的情況下存在安全漏洞。

對於加密方式的研究,I. Ahmad[10]認為 IPsec 是目前 4G 通訊中最常被使用的加密協定，並可以沿用至 5G 通訊；而近幾年，置換加密已經被使用於保護 IoT 資料與多媒體資料[11][12]。然而，這類的加密方式需要大量的運算資源與網路資源。因此此專題透過 P4 交換機進行線路速率的置換加密，並將其應用於保護 UPF 與 5G 核心網路的封包傳輸且不會需要額外的網路與運算資源。

四、 專題重要貢獻

此專題提出的加密方式，可以以線路速度(Line Rate)進行封包資料加解密，應用於 P4 交換機可達最高 6.4Tbps 的傳輸速率與 3.2Tbps 的加密速率，為目前最快的一種加密方式。

五、 設計原理、研究方法與步驟

1. 演算法採用與重構

此專題採用原自 T.-T. Lin[14]中的 CPA 演算法作為透過金鑰重新排序封包內

容次序的方式。此演算法容易實作於一般 CPU 架構的裝置上，利用一般程式語言使用指標與陣列結構撰寫。然而由於 P4 交換機具有一些限制，因此必須將其轉換為不需指標亦且空間效率更高的演算法。1.1.將介紹 CPA 演算法，1.2.提出不需指標但與 CPA 有相同排序結果的 eCPA 演算法，並於 1.3. 提出正確性證明。

1.1. CPA 演算法

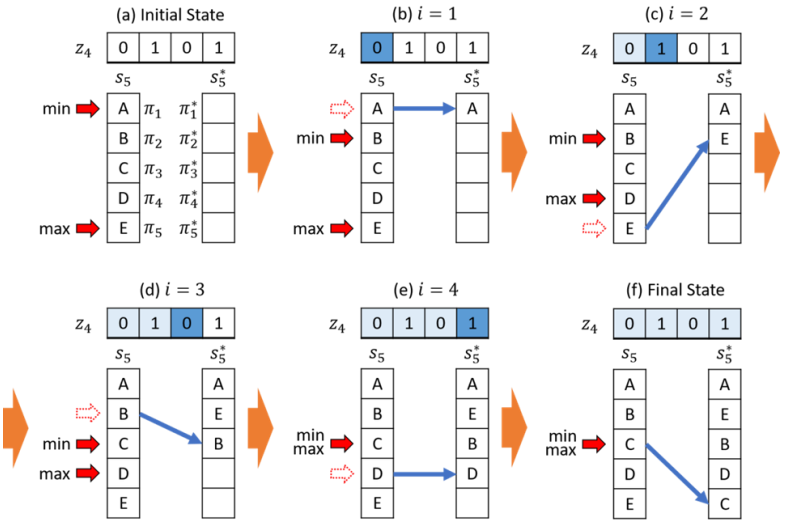
CPA 演算法可將各種金鑰對應到不同的酬載內容排序結果，演算法如圖：

```
1   Algorithm CPA(s_n, k_{n-1})
2       Input: s_n =< π_1, ..., π_n >, k_{n-1} = (x_1, ..., x_{n-1})
3       Output: s*_n =< π*_1, ..., π*_n >
4           let max = n; min = 1;
5           for i = 1 to n − 1 do {
6               if (x_i = 1)
7               then {π*_i = π_max; max = max − 1;}
8               else {π*_i = π_min; min = min + 1;}
9           }
10          π*_n = π_min;
11          Output < π*_1, ..., π*_n >;
```

其中，s_n為封包酬載內容，π_n為封包酬載的第n區段。k_{n-1}為長$n-1$位元的金鑰，x_i為金鑰中的第i位元且$x_i \in \{0,1\}$，$1 \le i \le n-1$。CPA 演算法利用k_{n-1}重新排序s_n並且將結果另存於s_n^*。max、min作為演算法中的陣列指標，針對每個x_i的值將max或min所指向的值複製到π_i^*並且對max或min進行遞減與遞增。此演算法的結果滿足以下方程式：

$$\pi_i^* = \begin{cases} \pi_{i-\theta_i}, & \text{for } x_i = 0, \\ \pi_{n-\theta_i+1}, & \text{for } x_i = 1, \end{cases} \quad 1 \le i \le n-1$$
$$\pi_n^* = \pi_{n-\theta_{n-1}}$$

其中$\theta_i = |\{l | \, x_l = 1, 1 \le l \le i\}|$。下圖為 CPA 演算法以$n = 5$運行的範例。

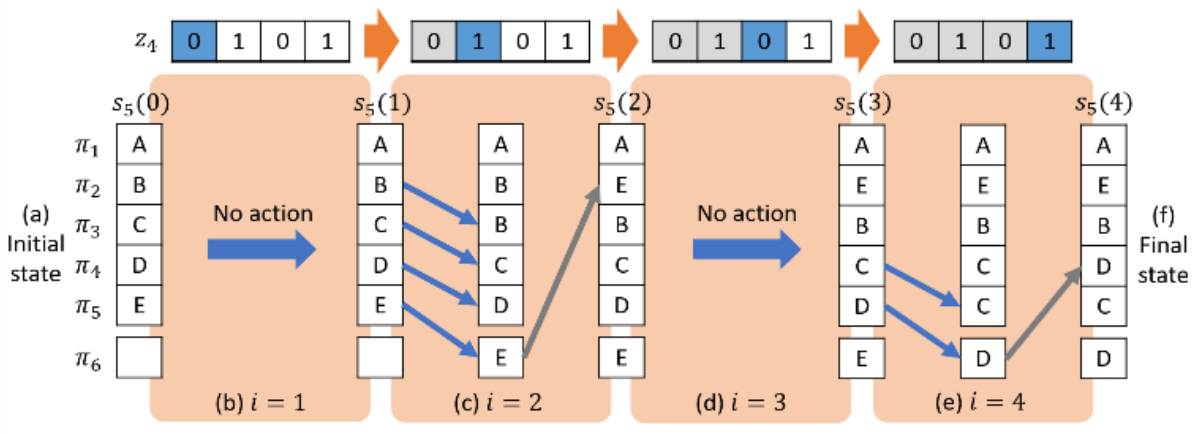


1.2. eCPA 演算法

P4 交換機由於存在下列限制而無法直接運行 CPA 演算法。第一，撰寫 P4 交換機程式的 P4 語言並無指標可以使用，因此無法實作 CPA 的max或min；第二，P4 交換機可定義的儲存空間有限，CPA 需要$2n|\pi_i|$的儲存空間才得以執行金鑰長度$n-1$的 CPA 演算法，造成空間的浪費。因此此專題提出 enhanced CPA (eCPA)演算法，以平移的方式將π_i中的資料重新排序，不需指標但保有與 CPA 完全相同的排序結果。演算法如下圖：

```
1   Algorithm eCPA(s_n, k_{n-1})
2       Input: s_n = s_n(0) =< π_1(0), ..., π_n(0) >,
3       k_{n-1} = (x_1, ..., x_{n-1}), π_{n+1}(0) = NIL
4       Output: s_n = s_n(n-1) =< π_1(n-1), ..., π_n(n-1) >
5           for i = 1 to n-1 do {
6               if x_i = 1 then {
7                   for j = i to n do {π_{j+1}(i) = π_j(i-1);}
8                   π_i(i) = π_{n+1}(i);
9               }
10          }
```

其中，π_{i+1}為額外的暫存空間。定義$s_n(i) =< \pi_1(i), \dots, \pi_n(i) >$為執行完第$i$位金鑰對應動作(第$i$個迴圈)後各酬載區段的內容。此演算法只需要$(n+1)|\pi_i|$的儲存空間即可達成與 CPA 完全一致的運行結果。下圖為 CPA 演算法以$n=5$運行的範例。



1.3. eCPA 正確性證明

此專題透過 Loop Invariant 方式證明 eCPA 與 CPA 有完全相同的計算結果。

理論一：

定義$\theta_i = |\{l|\, x_l = 1, 1 \le l \le i\}|$。當$1 \le j \le i < n$時，eCPA 每執行一個迴圈，$s_n(i)$的內容滿足下列恆定條件：

$$\text{For } 1 \le j \le i, \quad \pi_j(i) = \begin{cases} \pi_{j-\theta_j}(0), \text{if } x_j = 0 & (1) \\ \pi_{n-\theta_j+1}(0), \text{if } x_j = 1 & (2) \end{cases}$$

$$\text{For } i < j \le n, \quad \pi_j(i) = \pi_{j-\theta_i}(0) \qquad (3)$$

$$\text{For j = n + 1}, \quad \pi_{n+1}(i) = \begin{cases} \text{NIL}, & \text{if } \theta_i = 0 \quad (4) \\ \pi_{n-\theta_i+1}(0), & \text{if } \theta_i > 0 \quad (5) \end{cases}$$

證明一：

運用數學歸納法證明執行i次迴圈後條件(1)-(5)仍成立。

起始步驟：當$i = 1$時，以下成立：

$$\theta_i = \theta_1 = \begin{cases} 0 \text{ if } x_1 = 0 \\ 1 \text{ if } x_1 = 1 \end{cases}$$

若$x_1 = 0$，可得到$s_n(1) = s_n(0)$。因此得到：

$$\text{For } 1 \le \text{j} \le \text{n}, \quad \pi_j(1) = \pi_j(0) \qquad (6)$$

在此情況下，$\theta_1 = 0$。當$1 \le j \le i = 1$時，(6)式滿足恆定條件(1)。當$i < j \le n$時，$j - \theta_1 = j$，(6)亦滿足恆定條件(3)。而$\pi_{n+1}(1) = \text{NIL}$滿足條件(4)。

若$x_1 = 1$，則$\theta_1 = 1$。此時$\pi_n(0), \pi_{n-1}(0), \dots, \pi_1(0)$中的內容被平移至$\pi_{n+1}(1), \pi_n(1), \dots, \pi_2(1)$，且$\pi_1(1) = \pi_{n+1}(1)$。此時在$1 = i < j \le n + 1$條件下，可得：

$$\pi_j(1) = \pi_{j-1}(0) = \pi_{j-\theta_1}(0) \qquad (7)$$

當$1 < j \le n$時，(7)式滿足恆定條件(3)。同理，當$j = n + 1$時，可由(7)式得知$\pi_{n+1}(1) = \pi_n(0) = \pi_{n-\theta_1+1}(0)$而滿足恆定條件(5)。執行 eCPA 第八行後可得到下式：

$$\pi_1(1) = \pi_n(0) = \pi_{n-\theta_1+1}(0) \qquad (8)$$

而(8)式在$j = 1$時，滿足恆定條件(2)。因此起始步驟可滿足所有恆定條件。

推遞步驟：假設執行第i迴圈後可滿足所有恆定條件，接著推導執行第$i + 1$迴圈後亦滿足所有恆定條件。

若$x_{i+1} = 0$，可得$\pi_j(i + 1) = \pi_j(i)$，$1 \le j \le n$。由於恆定條件(1)、(3)於第i迴圈執行後可成立，因此上式可改寫為：

$$\pi_j(i + 1) = \pi_{j-\theta_{i+1}}(0) \qquad (9)$$

(9)式意味著恆定條件(1)與(3)於第$i + 1$迴圈執行後亦成立。在$j = n + 1$條件

下，由於恆定條件(4)、(5)於第i迴圈執行後成立，因此可得到：

$$\pi_{n+1}(i+1) = \pi_{n+1}(i) = \begin{cases} \text{NIL}, & \text{if } \theta_i = 0 \\ \pi_{n-\theta_i+1}(0), & \text{if } \theta_i > 0 \end{cases} \qquad (10)$$

由於$\theta_i = \theta_{i+1}$，(10)式可被替換為：

$$\pi_{n+1}(i+1) = \begin{cases} \text{NIL}, & \text{if } \theta_{i+1} = 0 \\ \pi_{n-\theta_{i+1}+1}(0), & \text{if } \theta_{i+1} > 0 \end{cases}$$

此式代表第$i+1$迴圈執行後亦滿足恆定條件(4)或(5)。

若$x_{i+1} = 1$，在$1 \le j < i+1$時可得$\pi_j(i+1) = \pi_j(i)$，因為執行第$i+1$迴圈並不改變$\pi_j(i)$內容。由於執行第i迴圈後恆定條件(2)可成立，上式可改寫為$\pi_j(i+1) = \pi_{n-\theta_j+1}(0)$。此式在不考慮$j = i+1$的情況下可以滿足恆定條件(2)，而該特例將進一步陳述。

$x_{i+1} = 1$使得$\pi_n(i), \pi_{n-1}(i), \dots, \pi_{i+1}(i)$中的內容被平移至$\pi_{n+1}(i+1), \pi_n(i+1), \dots, \pi_{i+2}(i+1)$。因此可得：

$$\text{For } i+1 \le j \le n+1, \qquad \pi_j(i+1) = \pi_{j-1}(i) \qquad (11)$$

因為

$$\theta_{i+1} = \theta_i + 1 \qquad (12)$$

而由於執行第i迴圈後滿足恆定條件(3)，因此透過(12)式可得：

$$\pi_j(i) = \pi_{j-\theta_i}(0) = \pi_{j-(\theta_{i+1}-1)}(0) = \pi_{(j+1)-\theta_{i+1}}(0) \qquad (13)$$

將(13)式與(11)式結合可得：

$$\pi_j(i+1) = \pi_{j-\theta_{i+1}}(0) \qquad (14)$$

$i+1 < j \le n+1$時，(14)式代表執行第$i+1$迴圈後可滿足恆定條件(3)與(5)。而$j = i+1$時，$\pi_{i+1}(i+1)$的內容被替換為$\pi_{n+1}(i+1)$。(14)式可被進一步寫為：

$$\pi_{i+1}(i+1) = \pi_{n+1}(i+1) = \pi_{(n+1)-\theta_{i+1}}(0) = \pi_{n-\theta_{i+1}+1}(0) \qquad (15)$$

而(15)式代表在$j = i+1$情況下，執行第$i+1$迴圈後可滿足恆定條件(2)。因此可得知在任何次數迴圈被執行後，皆滿足所有恆定條件。

理論二：

eCPA 與 CPA 有相同執行結果。

證明二：

由於理論一成立，在 eCPA 執行完第$n-1$個迴圈後可得到：

$$\text{For } j < n-1, \qquad \pi_j(n-1) = \begin{cases} \pi_{j-\theta_j}(0), \text{if } x_j = 0 \\ \pi_{n-\theta_j+1}(0), \text{if } x_j = 1 \end{cases}$$

$$\pi_n(n-1) = \pi_{n-\theta_{n-1}}(0)$$

因此$s_n(n-1)$滿足理論一，代表 eCPA 與 CPA 有相同執行結果。

2. 演算法實作

eCPA 不需指標定址即可完成與 CPA 效果相同的置換，而藉由 P4 交換機可自定義標頭欄位與運算的能力，此專題得以將封包酬載定義為標頭欄位並重新排序。然而實作 eCPA 於 P4 交換機時需要運用一些手法以配合該硬體的架構與限制。第一，各儲存空間在每個 P4 管線階段中只能進行一次讀取與寫入；第二，所有 P4 中的儲存空間被區分為數個群組，每個群組皆有專屬的 ALU 進行欄位的運算處理且不同 ALU 可同時進行運算。2.1. 2.2.將陳述此專題如何運用這些特性將置換效能最大化。

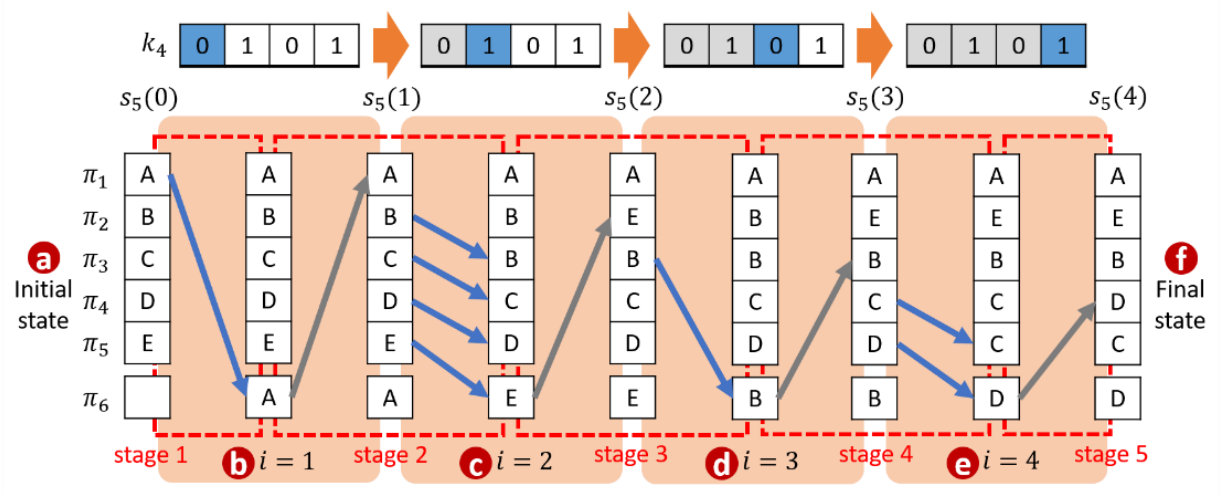
2.1. 交疊管線階段與 eCPA 迴圈

此專題將 eCPA 各迴圈遇到$x_i = 0$時原先的無動作，改為兩個冗餘動作如下：

$$\pi_{n+1}(i) = \pi_i(i-1) \qquad (16)$$

$$\pi_i(i) = \pi_{n+1}(i) \qquad (17)$$

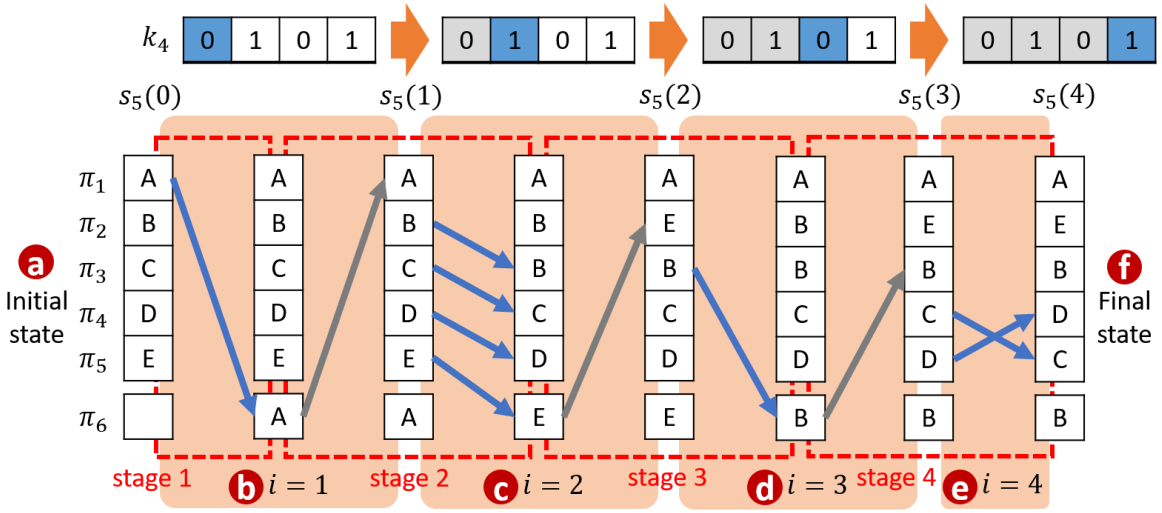
其中，(17)式與 eCPA 第 8 行的行為相同。新增上述兩個動作後，迴圈在$x_i = 0$或$x_i = 1$時的行為的不同之處，只有(16)式與 eCPA 第 7 行的執行。此時交疊 P4 交換機的管線階段與 eCPA 的迴圈，可得到以下結果：



其中，紅色框代表 P4 交換機的管線階段，橘色框代表 eCPA 迴圈。最前面

的管線階段只會執行 eCPA 的第七行或(16)式，eCPA 的第八行或(17)式則於下一個管線階段執行。由此可知，除了第一個管線階段僅執行 eCPA 迴圈的第一部分(eCPA 第 7 行或(16)式)，最後一個管線階段僅執行 eCPA 迴圈的第二部分(eCPA 第 8 行或(16)式)外，第i管線階段會執行第$i-1$迴圈的第二部分(eCPA 第 8 行或(17)式)與第i迴圈的第一部分。

若進一步將最後一個迴圈$x_i = 1$時的動作直接改為 P4 語言中的 swap()指令，則可直接省去 eCPA 最後一個迴圈的第二步驟而可減少使用一個階段。結果如下：

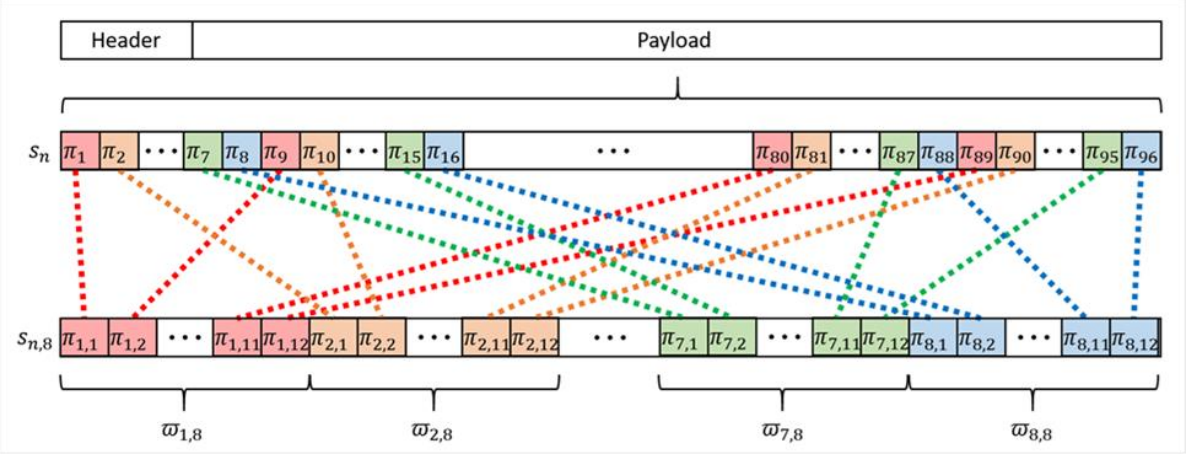


由此圖可知，P4 交換機可使用k個管線階段運行$n = k + 1$的 eCPA 演算法。目前現有的 P4 交換機具有 12 個階段，因此在實作上可以利用第一個階段讀取金鑰並利用剩餘的 11 個階段執行$n = 12$的 eCPA 演算法。

2.2. 平行處理多組演算法與欄位錯置

此專題利用各組儲存空間的 ALU 可平行處理的特性，將封包酬載區分成 8 組，每組具有$\varepsilon = 12$個酬載小區段指派至同組儲存空間。其中，由於 P4 交換機中每組儲存空間大小不盡相同，經實驗後得知 P4 交換機可以同時進行 2 組每個酬載小區段 32 位元的 eCPA 與 6 組每個酬載小區段 32 位元的 eCPA。在此情況下，P4 交換機最高可以將酬載長度$32 \times 12 \times 2 + 16 \times 12 \times 6 = 1920$位元組、96 個酬載小區段的封包，利用 8 組每組 11 位元的金鑰$k_{\varepsilon_m - 1}$，$1 \le m \le 8$，進行酬載加密。除此之外，為了增強其安全性，此專題將封包酬載的各小區段交錯對應到不同群組，相鄰的兩個小區段並不會被同一個 eCPA 演算法進行排

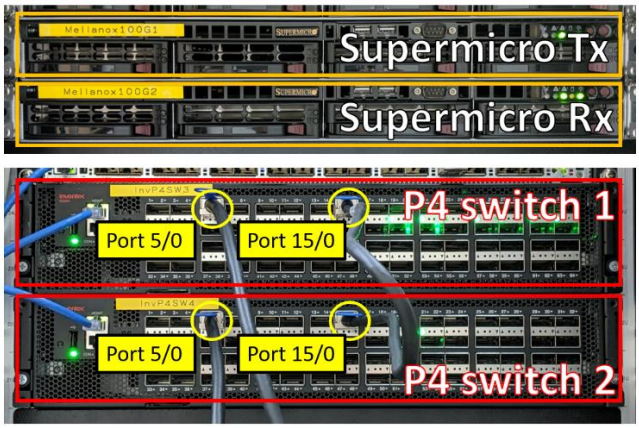
序。示意圖如下：

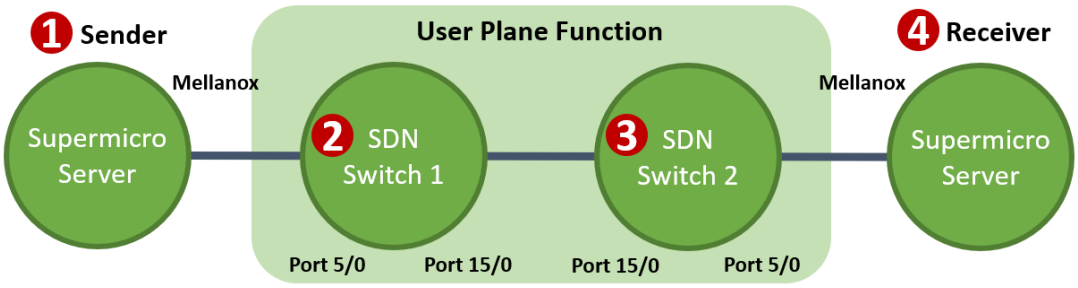


$s_{n,8} =< \varpi_{1,8}, \varpi_{2,8}, \ldots, \varpi_{8,8} >$ 為分組後的封包酬載，其中 $\varpi_{m,8} =< \pi_{m,1}, \pi_{m,2}, \ldots, \pi_{m,\varepsilon} >$ 為第 m 群組中的ε個小區段。$1 \le n^* \le n = 96$，$1 \le l \le \varepsilon = 12$，$1 \le m \le 8$。原始封包酬載中的第$n^*$個小區段以$n^* = 8 \times (m-1) + l$對應至第$m$組中的第$l$個酬載小區段$\pi_{m,l}$，並同時進行eCPA$(\varpi_{m,8}, k_{\varepsilon_m - 1})$，$1 \le m \le 8$。

六、 系統實現與實驗

此專題利用兩台配置 Mellanox Connect-X5 100G NIC、Intel(R) Xeon(R) E5-2675 v3 的 Supermicro server 與兩台 Inventec D5264 series P4 交換機模擬 5G 核心網路環境，連結方式與實際情形如下圖：





左側 Supermicro Server 以 100Gbps 位元速率向第一台 P4 交換機持續發送封包，模擬來自多個RAN的資料流。第一台P4交換機作為連接RAN與SDN WAN的角色，經由 SDN Controller 得到金鑰後，將接收到的封包進行 eCPA 置換加密，並透過 100G QSFP+光纖轉送給第二台 P4 交換機。兩台 P4 交換機之間的傳輸視為封包經過 SDN WAN 的過程。第二台 P4 交換機作為連接 SDN WAN 與 DN 的角色，也從 SDN Controller 接收相同的金鑰，並將封包進行 eCPA 反運算的置換解密，最後再將封包轉送至第二台 Supermicro Server，模擬在 DN 中的 IoT Server 接收 IoT 資料。

七、 效能評估與成果

由於實驗環境的限制，無法利用 Supermicro Server 以封包長度 250 byte (約等於 1920 bits 酬載 + 75 bits 標頭) 創造 100Gbps 的資料流，因此本專題改以封包長度 1536 byte 創造 100Gbps 的資料流進行實驗。實驗結果顯示未有任何封包遺失的狀況發生。而 Inventec D5264 每秒最高可以處理 5 Billion 個封包，因此可以假設每秒處理封包個數不超過此上限時，不會有明顯的封包遺失狀況。

接著，將上述實驗結果推算出的封包處理效能與 AdvanTek SKY-8101 運行加密演算法的效能數據[19]進行比較。比較表格如下：

裝置		Inventec D5264	AdvanTek SKY-8101
處理器架構		Tofino 6.5T	x86 CPU (Xeon Platinum 8176)
最大耗能		995W	1200W (Power Supply)
運行加密演算法		eCPA 88-bit security level	AES128-CBC 128-bit security level
裝置網路配備		64 x 100G	2 x 50G + 4 x 40G
效能	250 bytes 封包	1250Gbps throughput on 32 rx 32 tx (理論速率)	50Gbps throughput on 2 rx 1 tx (實測速率)
	1536 bytes 封包	100Gbps throughput on 1 rx 1 tx (實測速率) 3200Gbps throughput on 32 rx 32 tx (理論速率)	100Gbps throughput on 2 rx 1 tx (實測速率)

其中，若 Inventec D5264 以每秒 5 Billion packets per sec 處理長度為 250 bytes 的封包，則最高可以處理 $\min(3200, 5 \times 250) = 1250\text{Gbps}$ 的資料流。若處理長度為 1536bytes 的封包，則可以處理 $\min(3200, 5 \times 1536) = 3200\text{Gbps}$ 的資料流。此表格顯示，在最大耗能差不多的情況下，用 Inventec D5264 進行加密 250 bytes 封包的

速率遠比 Advantek SKY8101 高。雖然於 Inventec D5264 運行 eCPA 的 security level 不如 Advantek SKY8101 運行的 AES128-CBC，但未來市場推出下一代搭載 Tofino 2 ASIC 的 P4 交換機後，其所具備更多的管線階段(20 個)與儲存空間可以運作更高 security level 的 eCPA 加密演算法與更快的加解密。

八、 結論

由上述實驗結果得知，P4 交換機有能力利用管線架構以線路速率進行高速置換加解密。若同時啟用現有 P4 交換機上的 64 個 100G 傳輸埠，則可讓 P4 交換機以總傳輸速率 6.4Tbps 的資料流進行置換加解密 1536 bytes 的封包；若以 250 bytes 封包進行加密，最大速率也可達 1250Gbps。以單一裝置而言為目前世界上最快也最有效率的一種加密方式。

未來新一代的 P4 交換機具有更強大的傳輸能力、更多儲存空間與管線階段，可以使封包酬載區分的區段數量增加，進一步提升此加密演算法的安全性與最大酬載長度。

九、 參考文獻

[1] Y.-B. Lin, H.-C. Tseng, Y.-W. Lin and L.-J. Chen, "NB-IoTtalk: A Service Platform for Fast Development of NB-IoT Applications", IEEE Internet of Things Journal, Vol.6, Issue 1, pp.928-939, February, 2019.

[2] Y.-B. Lin, L.-K. Chen, M.-Z. Shieh, Y.-W. Lin, and T.-H. Yen. CampusTalk: IoT Devices and Their Interesting Features on Campus Applications. IEEE Access. Volume: 6, Issue:1, Page(s): 26036-26046, December, 2018.

[3] W.-L. Chen, Y.-B. Lin, Y.-W. Lin, R. Chen, J.-K. Liao, F.-L. Ng, Y.-Y. Chan, Y.-C. Liu, C.-C. Wang, C.-H. Chiu and T.-H. Yen, "AgriTalk: IoT for Precision Soil Farming of Turmeric Cultivation", IEEE Internet of Things Journal, 2019.

[4] P. Bosshart, D. Daly, G. Gibb, M. Izzard, N. McKeown, J. Rexford, C. Schlesinger, D. Talayco, A. Vahdat, G. Varghese, and D. Walker, “P4: Programming protocol-independent packet processors,” SIGCOMM Comput. Commun. Rev., vol. 44, no. 3, pp. 87–95, Jul. 2014

[5] Y.-B. Lin, S.-Y. Wang, C.-C. Huang, C.-M. Wu. SDN Approach for Aggregation/Disaggregation of Sensor Data, Sensors, 18(7): 2025, 2018.

[6] M. A. Ferrag, L. Maglarasc, A. Argyrioud, D. Kosmanosd, and H. Janickec. Security for 4G and 5G cellular networks: A survey of existing authentication and privacy-preserving schemes. Journal of Network and Computer Applications, 101 (2018) 55–82.

[7] A. R. Prasad, S. Arumugam, S. B and A. Zugenmaier. 3GPP 5G Security. Journal of ICT, Vol. 6 1&2, 137–158, 2018.

[8] D. Basin, J. Dreier, L. Hirschi, S. Radomirović, R. Sasse and V. Stettler. A Formal Analysis of 5G Authentication. 2018 ACM SIGSAC Conference on Computer and Communications Security, Pages 1383-1396, Toronto, Canada — October 15 - 19, 2018.

[9] R. P. Jover and V. Marojevic. Security and Protocol Exploit Analysis of the 5G Specifications. IEEE Access, PP(99):1-1, February 2019.

[10] I. Ahmad, T. Kumar, M. Liyanage, J. Okwuibe, M. Ylianttila, and A. Gurtovk. 5G Security: Analysis of Threats and Solutions, 2017 IEEE Conference on Standards for Communications and Networking (CSCN), 2017.

[11] G. Lasry. "Solving the Double Transposition Challenge with a Divide-and-Conquer Approach," Cryptologia. 38 (3): 197–214, 2014.

[12] S. Li and K.-T. Lo. Optimal quantitative cryptanalysis of permutation-only multimedia ciphers against plaintext attacks, Signal Processing, Volume 91, Issue 4, Pages 949-954, April 2011.

[13] S. Majumdar, A. Maiti, B. Bhattacharyya, and A. Nath. A New Bit-level Columnar Transposition Encryption Algorithm, International Journal of Advance Research in Computer Science and Management Studies, Volume 3, Issue 7, July 2015.

[14] T.-T. Lin, S.-C. Tsai ; Wen-Guey Tzeng. Efficient encoding and decoding with permutation arrays, 2008 IEEE International Symposium on Information Theory, 2008

[15] Inventec D5264 series switch, available: http://productline.inventec.com/switch/Download/D5264.pdf.

[16] Y.-B. Lin, C.-C. Huang and S.-C. Tsai, "A SDN Soft Computing Application for Detecting Heavy Hitters", Accepted and to appear in IEEE Transactions on Industrial Informatics, 2019.

[17] Mellanox. Available online: http://www.mellanox.com/

[18] Pktgen. Available online: https://pktgen-dpdk.readthedocs.io/en/latest/index.html

[19]White Paper: IPsec Performance Boosts with Networking Platforms based on Intel Xeon Scalable processors: https://advantechfiles.blob.core.windows.net/cms/66636345-0b11-4a33-a7cc-c21fcea859be/Whitepaper%20%20PDF%20File/Advantech_White_paper_IPsec-_Intel_Xeon_Scalable.pdf