一個網路電話監聽機制之研究

以 RTCP 為基礎的 VoIP 數位簽章機制

A RTCP-based Digital Signature Mechanism for VoIP

一個網路電話監聽機制之研究
以 RTCP 為基礎的 VoIP 數位簽章機制

摘要

隨著網路電話(Voice over Internet Protocol,VoIP)技術的成熟及寬頻網路的普及，VoIP 已有逐漸取代傳統 PSTN 電話的趨勢。VoIP 應用大量的數位技術，能達到比傳統 PSTN 電話更安全的防護，目前已有許多金鑰交換及加密技術被大量應用在實際商品上，但仍缺少在 VoIP 中加入數位簽章來達到不可否認性的相關規範。我們認為在 VoIP 中加入數位簽章，必須滿足(1)擁有不可否認性(2)不影響通話品質，必須有很小的傳送端延遲(3)即使在低階手持裝置上也能運行的低運算工作負載(4)不佔用太多頻寬這四項需求，在研究數種串流簽章後，我們將 Efficient Multi-chained Stream Signature(EMSS)加入 VoIP 中。EMSS 運用在 VoIP 上可滿足前三項需求，但在頻寬使用上仍有改進空間。我們進一步提出一個根據 RTCP 動態調整 EMSS 參數的機制，經實驗驗證，在犧牲少許驗證率(4%以下)為條件之下，可以節省約 6%~15%以上的頻寬。

關鍵字：VoIP、數位簽章、Stream Signature、RTCP

A RTCP-based Digital Signature Mechanism for VoIP

Abstract

With the maturity of VoIP (Voice over Internet Protocol, VoIP) technology and the popularity of broadband Network , VoIP has the trends of replacing traditional PSTN phone gradually . VoIP applies a great deal of digital technologies, that can achieve more security protection than traditional PSTN phone, there are many key exchange mechanism and encryption technology has been widely used in the actual goods, but still lacking of some purposals to achieve non-repudiation through digital signatures . We believe that the digital signature suited for VoIP, must be met (1) non-repidiation (2) low sender delay (3)low workload (4) low communication overhead, In examining a number of streaming signature, we adopt Efficient Multi-chained Stream Signature (EMSS) to join in VoIP. EMSS can satisfy the first three demands, but on the aspect of bandwidth usage, there are still room for improvement. We propose a further mechanism about how to adjust EMSS parameter dynamically based on RTCP(Real-time Transport Control Protocol) ,after the experimental verification, we can save at least 4~16% bandwidth at the cost of less than 4% verification rate.

Keyword: VoIP、Digital Signature、Stream Signature、RTCP

目錄

圖目錄

表目錄

第一章、緒論

1.1 研究背景與動機

網路電話(Voice over Internet Protocol,VoIP)[22]隨著寬頻及軟硬體技術的逐漸成熟，在不久的將來，取代傳統電話的願景已不是不無可能。網路電話由於使用數位化技術，能做到比以往類比電話更強的安全性，包括語音資料的壓縮及修補、或對交談的內容進行加密…等。

現行網路電話的相關規範中，並沒有規定加入數位簽章，來達到不可否認性的機制。由於電子簽章法的關係，網路電話使用的語音封包可視為數位資料，若在其中加入簽章，則可以在法律上存有效力。比如兩個人在遠端透過網路電話談生意，交談內容由律師錄下，便可以代替紙本契約[2]。此外隨著二類電信的開放，司法單位亦要求二類電信業者必須提供網路電話監聽的功能，若在其中加入數位簽章，則錄製下的數位證據不需經由聲紋比對，即可成為法庭證據。若使用者透過 IVR(Interactive Voice Response)查詢股票、或進行下單的動作，若在其中加入數位簽章機制，可以確保該訊息的可靠性。

為了達到不可否認性的目的，一般來說我們會應用數位簽章的技術。但如果依循傳統作法，在每個語音封包加入數位簽章，會出現以下兩個問題：(1)computational overhead：文獻[3]中提到，使用 Pentium 300 的裝置，每秒只能簽署 80 個 RSA 數位簽章，而且這會耗去所有運算資源。(2)communication overhead：以 1024 bit RSA 簽章來說，每個語音封包必須額外夾帶 128 bytes 的簽章。因此可以採用有別於傳統簽章的串流簽章(Stream Signature)方法。我們認為該方法必須滿足下列四項特性：(1)不可否認性。(2)很小的使用者端延遲。(3)低運算資源負載。(4)不佔用太大的頻寬。

1.2 研究目的

本論文中根據上述的研究動機，於達成以下目的：

1. 研究現行的串流簽章方案，是否適用於 VoIP 中。

2. 在現行方案中尋找可行的解決辦法，並探討有何改進之處。

3. 模擬實驗，並討論其帶來的效益。

1.3 論文架構

本篇論文共分為五章，第一章為「緒論」，對研究動機、背景、目的以及論文整體架構作一簡單的說明。第二章為「文獻探討」，介紹與本研究相關的主題並給予小結論。第三章為介紹如何在 VoIP 中加入 EMSS，並藉由 RTCP 動態調整參數的機制。第四章為介紹系統實作的流程以及實驗的數據與結果。第五章為「結論與未來工作」，為總結本研究的研究成果，並針對未來可能的研究方向加以說明。

第二章、文獻探討

在這個章節裡，我們將對既有的文獻進行探討。首先我們介紹數位聲音的原理及編碼方式，以及數位聲音是如何在分封網路中傳遞，緊接著，我們會簡單介紹目前 VoIP 的協定家族，其中包括連線建立協定(Session Establishment Protocol)、RTP、RTCP 及現有的一些安全機制。最後，為了在 VoIP 中達到不否認性，我們會探討什麼是數位簽章及串流簽章。

2.1 數位聲音(Digital Voice)

2.1.1 語音(Speech)

語音為人類溝通的主要方式，並且是人類社會中彼此聯繫的媒體，因而語音通訊的發展也相當早。從貝爾發明電話起，人類就逐步實現遠距通話的夢想。從那時開始，電話的功能隨著電子、電機技術的發展愈來愈精緻、愈來愈便宜。同時，近代，通訊更往數位化的方向發展，使得語音編解碼技術的需求也就愈趨明顯。除此之外，音訊隨著朝向數位化發展，而成為多媒體通訊的重要一環。

語音的訊號，是非固定(non-stationary)且可視為一小段、一小段無法讓人準確預測的訊號，發音的基本原理是由肺部氣壓迫使聲帶張開，讓空氣通過，空氣通過聲門狹小縫隙，流速加快導致氣壓下降，使得聲帶彈回而閉合，如此週而復始的振動。當聲帶振動時，氣流脈波便會經過咽喉、口鼻，受到共鳴效果的修飾，最後從嘴唇與鼻孔輻射出去。語音依發音時聲帶是否振動，又可再細分為有聲音(voiced sounds)的 b、t、k 與無聲音(unvoiced sounds)的 s、th 等。

2.1.2 數位化(Digitalization)

從字面上來說，數位(Digital)就是以數字來描述事物，像一張桌子、兩張椅子這類可以很明確地用數字去表達它。而另一個經常伴隨數位出現的名詞是類比(Analog)，類比的意思即是用相似的東西去表達，像是傳統的類比式電話，就

是用電壓波動的連續起伏來代表人類發出的聲音。

如果我們想要在網際網路上傳遞聲音，有一必要的前提工作是，我們必須將聲音從類比格式轉化成數位格式。因為在電腦的世界裡，所有的事物都是由一連串的0與1所構成。而數位化帶來的另一個好處是方便保存，使資料不容易失真。只要記錄資料的數字大小不改變，記錄資料的內容也就不會改變。傳統類比的方式記錄訊號，如使用 LP 表面的凹凸起伏或是錄音帶表面的磁力強度來表達振幅大小，在我們複製資料時，無論電路設計多麼嚴謹，總是無法避免雜訊的介入。像這樣的雜訊會融合成複製資料後的其中一部份，造成失真，且複製的次數越多，訊噪比(訊號大小與雜訊大小的比值)更會隨之降低，有意義的資料細節也越來越少。

將聲音數位化的步驟，必須透過取樣(sampling)及量化(quantization)來達成。轉換原理如下，首先讓聲音透過麥克風或話筒等收音設備，轉換成一連串電壓變化的訊號，如圖 1 示。我們現在開始將這聲音波型的類比訊號數位化，方法是以等時距分割橫坐標。假設用每 0.01 秒分割，則得到如圖 2 的 PAM(Pulse Amplitude Modulation)格式。接著我們把坐標數字記錄下來，得到如下資料，(0.01，11.65)、(0.02，14.00)、(0.03，16.00)…etc，又因為我們知道記錄的時間間距，因此我們最後只需要記下縱坐標的數字，得到的結果就是11.65、14.00、16.00…etc，如此，我們完成了將聲音數位化的基本工作。

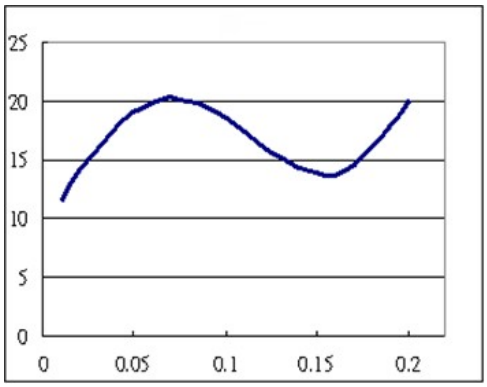


圖 1 類比聲音

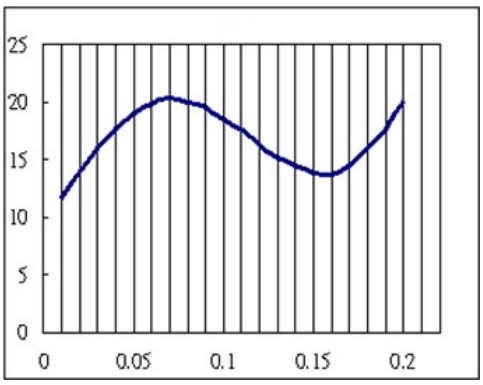


圖 2 聲音數位化

2.1.3 Coding/Compression

最後的工作，便是將這些 11.65、14.00、16.00…etc 轉換成電腦可接受的二進位(binary 格式，例如使用 8 個 bit 來記錄每一點，則會變成 00001100、00001110、00010000...etc。這就是最基本的 PCM 原理。不過通常我們還會進一步再將 PCM 壓縮成其它的 codec(code/decode)。以現在較常見的有 G.723、G.726、G.729、iLIBC..等等，除此之外還有許多私人企業自行開發的格式。

2.2 分封交換網路與封包遺失模型

要在個人電腦或終端裝置之間傳遞數位聲音，必須透過分封交換網路在點對點之間交換數位資料，我們在這個小節介紹什麼是分封交換網路，以及用來模擬分封交換網路封包遺失的數種模型。

2.2.1 分封交換網路 Packet Switch Network

和早期電話機房使用的線路交換網路(Circuit Switch Network)相比，分封交換網路著重在頻寬上的最大使用率。在分封交換網路的模式中，資料會在發送端被切割成一小塊、一小塊的封包，這些小塊的封包，會視網路當時的狀況經由相同、或不同的路徑路由(route)到目的地。由於在傳輸的途中，封包會被暫時存放在路徑節點的佇列(queue)中等待處理，與線路交換網路相比，這會造成可能影響通話品質的動態延遲(variable delay)及封包遺失(packet loss)，尤其在網路設備過於老舊時，這種情況更是明顯。目前我們使用的網際網路(Internet)就是採用這種資料交換方式。

2.2.2 Packet Loss/Loss Correlation Model

以現行 VoIP 的實作來看，為了效率上的考量，都以 UDP(User Datagram Protocol)來傳送語音封包。相較於 TCP(Transmission Control Protocol)來說，UDP 提供了較快速，但相對不可靠的傳送服務機制。且由於 UDP 傳輸的不可靠，會使得封包遺失的現象經常發生在網路中，尤其是當網路發生壅塞(congestion)時；這現象通常起因於網路節點的緩衝區溢位(buffer overflow)，即是它所接

收的已超過它所能處理的封包數量，以致於它必須被迫放棄處理某些封包。而網路中的某節點故障時，也有可能會造成連續性的封包大量遺失。除此之外，品質不佳的網路卡也會造成隨機性的封包遺失。另一個會造成封包遺失的因素是 bit error，這是指在傳輸途中訊號的失真毀損。不過除了在無線網路的環境中，bit error 發生的比率很低且較很難去預測，因此在本論文中所指的封包遺失僅表示壅塞(Congestion)一詞。

許多關於網路或修補演算法的研究，會需要模擬網路封包遺失的情境。我們在這個章節介紹三種有名、常被用來摸擬封包遺失的網路模型。分別是 Bernoulli Model、Gilbert Model、m-order Markov Model。

2.2.2.1 Bernoulli Model

Bernoulli[15]模型表示在網路中，封包遺失的機率會服從機率論裡的伯努利分配(Bernoulli Distribution)，也就是每一個封包遺失的事件都是獨立不相關(i.i.d.)。先前的研究指出，在下列數種情況之下，會讓封包遺失情形符合此模型

1. 每兩封包間，傳送時間的間隔大於 500ms(0.5 秒)。

2. 傳送的封包大小，僅佔全部頻寬的一小部份(約 1/10)。

3. 硬體不良產生的隨機性遺失。

但一般來說，我們較少使用 Bernoulli Model，因為在大部份的網路應用程式中，傳送封包的間隔並不會那麼大。

2.2.2.2 Gilbert Model

Gilbert Model[7][10]，又稱為 two-state Markov Model。它的基本想法是，現實網路中的封包遺失是會互相影響的，遺失的封包間，往往會存在短暫的相依性(temporary dependency)。在許多文獻中都建議使用 Gilbert Model 來模擬網路封包遺失。

Gilbert Model 經常被用來描述網路中突發封包遺失現象，它是由 E・N・Gilbert 在 1960 年提出的，如圖 3 所示。

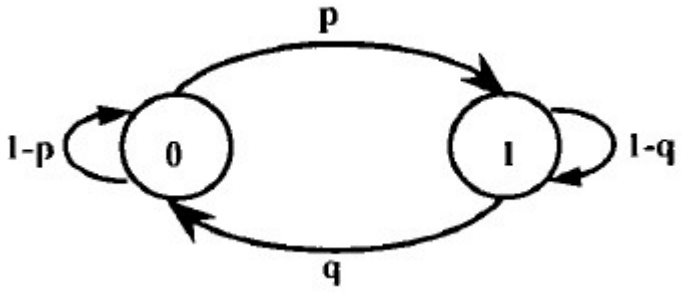


圖 3 Gilbert Model 狀態轉換圖(1)

Gilbert Model 用兩個獨立的件機率 p 和 q 來描述網路模型。

令一隨機變數 X 表代封包遺失的事件

$$l(X) = \begin{cases} 0: packet\ s\ is\ not\ lost \\ 1: packet\ s\ is\ lost \end{cases}$$

則機率 p 表示由“0”狀態轉變到“1”狀態的機率，q 表示由“1”狀態轉變到“0”狀態的機率。因此 p 又可以表示為 P_{01}，即機率

P(X = 1 | X = 0) = (封包遺失事件的次數)/(封包沒有遺失事件的次數)

其中“封包遺失事件的次數”是將連續若干封包遺失視為『一次』遺失事件，即為從 0 狀態轉變到 1 狀態的次數，同時也是從 1 狀態轉變到 0 狀態的次數。同樣地，q 又可以表示為 P_{10}，由於從 1 狀態轉變到 0 狀態的次數等於從 0 狀態轉變到 1 狀態的次數，因此得到

P(X = 0 | X = 1) = (發生遺失事件的次數)/(1 狀態出現的次數)

1-p 表示為 P_{00}，即 P(X = 0 | X = 0)，1-q 表示為 P_{11}，即 P(X = 1 |X = 1)。則 Gilbert Model 又可用圖 4 來表示:

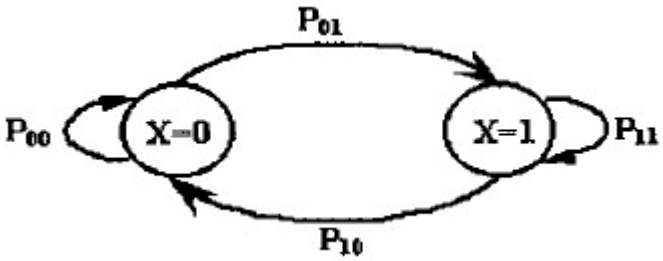


圖 4 Gilbert Model 狀態轉換圖(2)

設 P_1 為總平均遺失率，P_0 為總平均未遺失率，P_1 又稱為 ulp(Unconditional Loss Probability)，P_{11} 稱為 clp(Conditional Loss Probability)。

$$p_0 = \frac{0\text{ 狀態出現的次數}}{0\text{ 狀態出現的次數} + 1\text{ 狀態出現的次數}} = \frac{q}{p+q}$$

$$p_1 = 1 - p_0 = \frac{p}{p+q}$$

$$q = 1 - clp$$

$$p = \frac{ulp(1-clp)}{1-ulp}$$

如上推導出的公式，p 和 q 可以分別用 ulp 和 clp 來表示。實際上 ulp 即為一個網路的平均封包遺失率，而 clp 則反應了網路中遺失封包間的相依性，可以用 clp 和 ulp 兩個參數來描述一個網路的封包遺失特性。如果 p+q=1，則 Gilbert Model 會轉化為只有一個狀態的 Bernoulli Model，此時 ulp = clp。表 11[7] 為在兩所大學間，量測該段學術網路計算後得到的值。

表 1 Gilbert Model 參數量測表

封包間隔時間 (ms)	8	20	50	100	200	500
ulp	0.23	0.16	0.12	0.10	0.11	0.09
clp	0.60	0.42	0.27	0.18	0.18	0.09

2.2.2.3 m-order Markov Model

m-order Markov Model[15]對遺失封包之間的相依性做了更詳細的描述。可以這麼想，當 m =0 時，封包遺失率只有一種狀態，也就是 Bernoulli Model，當 m=1 時，即成為 Gilbert Model。m 的次數越大，對封包遺失的相依性會描述得越詳細(但在估算該網路時，必須耗用更多記憶體)。和 Gilbert Model 一樣，同樣有如下假設：令隨機變數 s 表示封包遺失的事件。

$$l(s) = \begin{cases} 0: packet\ s\ is\ not\ lost \\ 1: packet\ s\ is\ lost \end{cases}$$

定義 P(l(s) | l(s-k),……,l(s-2),l(s-1))為第 s 個封包的轉換機率，則所有 0 與 1 (l(s-m),…l(s-2),l(s-1_)的組合都會出現在狀態空間中。如圖 5 所示，P(l(s)=1|l(s-2),l(s-1) = 01)表示若在第 s-2 個封包未遺失及第 s-1 個封包遺失的條件之下，第 s 個封包遺失的條件機率，如此依序類推(詳細的估算方法參考[15])。

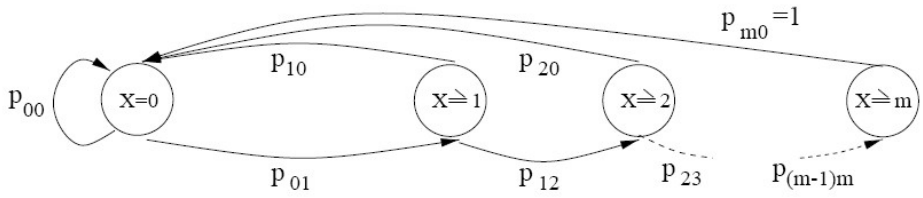


圖 5 m-order Markov Model

2.3 網路電話(VoIP)

網路電話[22]即是指透過網路，將數位聲音從發話方傳遞至受話方，達到類似傳統 PSTN 電話的功能，但實際上的運作卻有其複雜性。但由於網路電話利用了大量的數位技術，因此也比傳統的類比電話更具有附加價值。

要在網際網路中撥打數位電話，必須仰賴多種協定的互相合作。目前網路電話中的通訊協定大致上分為兩類，分別為點對點(Peer-to-Peer，P2P)和主從式

(Client-Server)兩大類，點對點通訊協定中以 H.323 和 SIP 最多人使用，適用在點對點的終端設備(例如個人電腦、或手持裝置)，而主從式通訊協定的 MGCP(Media Gateway Control Protocol)、Megaco 則適用在電信核心網路的局端設備。

完成一通網路電話必須由兩個階段的工作，第一階段是透過會話連線協定(Session Establishment Protocol)建立點對對連線，就像傳統 PSTN 網路的第七號信令(SS7)一般。首先撥打方向連線伺服器發出建立連線的要求(此要求會包含撥打方設定的通話參數，像是使用的 codec、ip、port…等等)，此要求會經由連線伺服器送達給受話方，當受話方接起話筒並送出相對的回應訊息之後，該通話的 Session 建立，意即雙方已成功地建立起通話管道。

第二個階段，進入交談狀態。雙方使用協調過後的 codec，經由先前所建立的通話管道傳送語音串流，當對話完畢(通話中的某一方掛掉話筒並送出結束訊息)之後，此會話亦隨之終結。圖 6 為目前網路電話經常使用的協定，我們在接下來的幾個小節快速地介紹 VoIP 中經常使用的 SIP(Session Initiation Protocol)、SDP(Session Description Protocol)、RTP(Real-time Transport Protocol)、RTCP(Real-Time Control Protocol)等通訊協定。

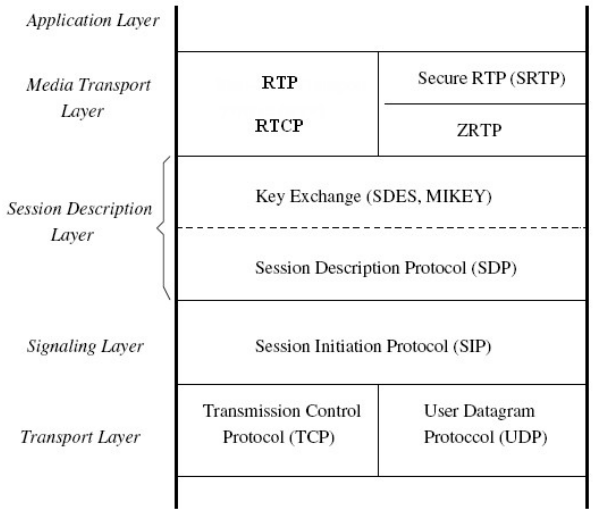


圖 6 VoIP Protocol Stack

2.3.1 SIP/SDP

如前所述，撥打網路電話的第一件工作，是必須在通話雙方之間建立連線。現行的網路使用 IP(Internet Protocol)位址來做為定址工具，想像在撥打每通電話之前，你都必須先知道對方的 IP 位址，再進行連線的話，那將會是一種非常沒有效率的作法。VoIP 發展至今，提出了許多建立連線的協定及架構，例如在 Windows 的 NetMeeting 軟體中使用 H.323 協定，其它又有如 MGCP 及 SIP 等等。這些協定目的其實都很類似，就是幫助使用者能夠非常快速、且方便地找到欲連線的對象。

在目前的諸多協定中，SIP(Session Initialization Protocol)[12]已漸漸成為主流，它以文字為基礎(Text-based)傳送訊息，訊息內容類似 HTTP 的格式(下一代的 SIP 採用 XML 格式，提供更強的擴充功能)，而且它也像 HTTP 一樣，使用一些數字代碼表示回應訊息(ex.200 OK、404 page not found…etc)。這種方式的優點就是簡單、易懂，缺點則是相較於 H.323 這類的二進位(binary)傳輸模式，會佔用更大的頻寬。目前比較新的網路電話均已採用 SIP。因此我們將在這個小節裡，簡單介紹 SIP 的基本原理及運作流程。

SIP 被規範於 RFC3261，是一種遵循擴散巴科斯形式(Backus－Naur form)的語法格式。它不僅止在 VoIP 中被使用，在其它一些關於網路會議、多媒體的應用中亦可看見它的身影。它定義有六項最基本的型式(SIP Method)如下：

1. REGISTER：用於登記聯繫資訊。
2. INVITE：用於邀請用戶加入對話。
3. ACK：對於收到的訊息作出回應。
4. CANCEL：用於取消未完成的請求
5. BYE：用於終止對話。
6. OPTIONS：用於詢問伺服器的性能(較不常被使用)。

我們實際看一個 INVITE 命令長成什麼樣子，如圖 7 所示：

```
INVITE sip:e9-airport.mit.edu SIP/2.0
From: "Dennis Baron"<sip:6172531000@mit.edu>;tag=1c41
To: sip:e9-airport.mit.edu
Call-Id: call-1096504121-2@18.10.0.79
Cseq: 1 INVITE
Contact: "Dennis Baron"<sip:6172531000@18.10.0.79>
Content-Type: application/sdp
Content-Length: 304
Accept-Language: en
Allow: INVITE, ACK, CANCEL, BYE, REFER, OPTIONS, NOTIFY, REGISTER,
  SUBSCRIBE
Supported: sip-cc, sip-cc-01, timer, replaces
User-Agent: Pingtel/2.1.11 (WinNT)
Date: Thu, 30 Sep 2004 00:28:42 GMT
Via: SIP/2.0/UDP 18.10.0.79
```

圖 7 SIP INVITE 命令

我們以表 2 簡敘圖 7 中 SIP 夾帶的重要資訊,完整參數列表請參閱 RFC3261。

表 2 SIP 參數說明表

參數	說明
INVITE	表示這是一個邀請他人加入對話的請求。
From	表示發話端的 SIP URI。
To	表示受話端的 SIP URI。From 和 To 的順序不可更換。
Call-ID	用於標識特定邀請或某個客戶端的註冊請求，每一次呼叫時會隨機產生一個唯一值，用於識別發話端的來源，若 Call-ID 值相同，那麼 Cseq 值必須各不相同，一個多媒體會議可產生多個 Call-ID 不同的呼叫。
Cseq	用於識別伺服器發出的不同請求，每次呼叫有一個 Command Sequence 值，以識別目前的回履是針對哪一次呼叫，如果重新發送 INVITE 時這個值需加大。
via	經過了那些位址，可以有多行，後來的由 Proxy Server 加註在前面。

SIP 命令之後經常還會伴隨 SDP(Session Description Protocol)[13]，SDP 被規範在 RFC2327 之中，它的內容包括下列四點：

1. 多媒體的種類，video、audio、etc。
2. 傳輸的協定(RTP/UDP/IP,etc)

3. 多媒體使用的 codec

4. 欲接收該多媒體串流所必須知道的資訊(IP 位址、通訊埠,etc)

一個實際的 SDP 內容就有如圖 8 所示的文字訊息：

```
v=0
o=Pingtel 5 5 IN IP4 18.10.0.79
s=phone-call
c=IN IP4 18.10.0.79
t=0 0
m=audio 8766 RTP/AVP 96 97 0 8 18 98
a=rtpmap:96 eg711u/8000/1
a=rtpmap:97 eg711a/8000/1
a=rtpmap:0 pcmu/8000/1
a=rtpmap:8 pcma/8000/1
a=rtpmap:18 g729/8000/1
a=fmtp:18 annexb=no
a=rtpmap:98 telephone-event/8000/1
```

圖 8 SDP 參數內容範例

我們同樣以表說明上述 SDP 的各項參數(完整參數列表請參閱 RFC)，如表 3 所示：

表 3 SDP 參數說明表

參數	說明
v	SDP 的版本，此處表示 SDP version = 0。
o	為 Origin 的縮寫，描述連線的來源訊息。例如此處的 Pingtel 為用戶名稱，後面的兩個 5 為隨機產生的戳記，最後表示 Internet IP Version 4，位址為 18.10.0.79。
c	c 為 Connection Information 的縮寫，內容類似 o。
t	t 定義協定訊息發送後的一段有效期間，表示 Repeat Times and Time Xones 的意思。此處的 t=0 0，是省略了對時間戳記的描述(但 SDP 規定 t 不可省略)。
m	m 是 Media Information 的縮寫，表示多媒體的類型及使用的通訊埠。

a	a 通常緊跟在 m 後面，表示 Attribute Line 的縮寫。即是發出此 SDP 訊息的使用者所偏好 codec 的優先順序(越上面的權限越大)。通話雙方可藉由 a 的值來決定此通話使用的 codec 為何。

SIP 使用類似 DNS 的機制，將使用者的 SIP 號碼與 IP 位址之間做映射(mapping)動作。即是使用 SIP 建立連線的用戶，會在開啟 VoIP 軟體的同時，在背景(background)向 SIP Server 發出『REGISTER』的登錄命令，告訴 SIP Server 說:『如果有人想撥打給我，請到 xxx.xxx.xxx.xxx 這個 IP 位址通知我，我使用 xxxxx 連接埠。』

另一個可能讓初學 SIP 的人混淆的問題是，為何會有 SIP 及 SDP 的分別?其實只要這樣想就不難了，SIP 就好像信封，而 SDP 則是被包裹在其內的信件。SIP 負責將 SDP 的參數傳送到適合的人手上。

最後我們用圖 9 展示一個簡單的 SIP 建立連線的流程，它僅需要下列四個參與者：撥打方(UserA)、Proxy Server、Location Server 以及受話方(UserB)，如下圖所示:

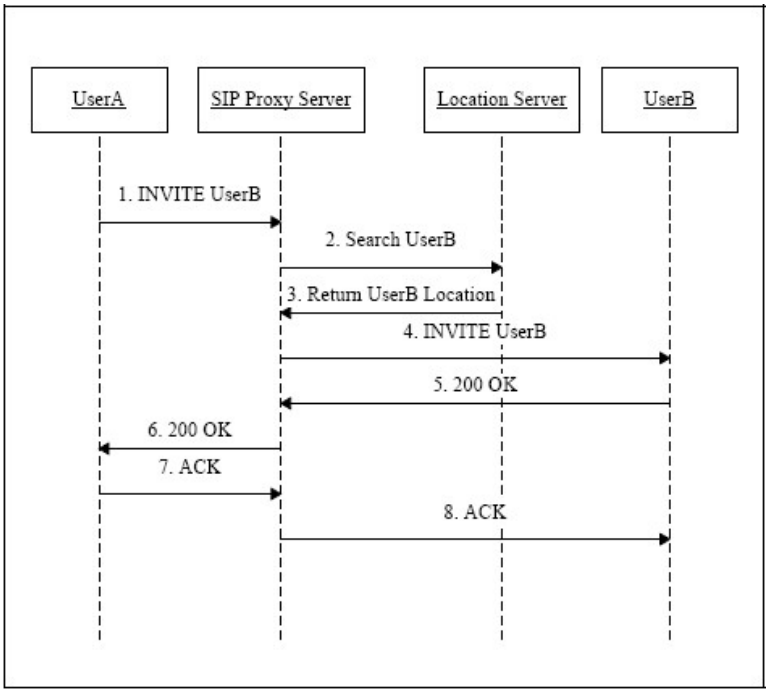


圖 9 SIP Signaling Diagram

1. 使用者 UserA 送出 INVITE 訊息至 SIP Proxy Server
2. SIP Proxy 收到後，詢問 Location Server UserB 的位址。
3. Location Server 回傳 UserB 的位址給 Proxy Server。
4. SIP Proxy Server 收到 Location Server 回傳的位置，送出 INVITE 訊息至 UserB。
5. UserB 收到 INVITE 訊息，如果可以進行通話(UserB 接起話筒)即回傳 200 OK 訊息給 SIP Proxy Server
6. SIP Proxy Server 再將 200 OK 訊息回傳至 UserA。
7. UserA 回傳 ACK 確認進行通話至 SIP Proxy Server。
8. SIP Proxy Server 將最後的 ACK 訊息轉送給 UserB。

完成了連線階段，雙方會對彼此產生某種程度的共識，UserA 與 UserB 接著便以點對點的方式，將語音資料封裝成 RTP 封包傳遞給對方。值得一提的是，SIP 雖然已成為目前主流的連線建立協定，但也有一些研究[4]提出透過 web services 之類的 middleware 將連線協定再進一步封裝，藉此達到跨平台、跨協定的功能。

2.3.2 Real-time Transport Protocol(RTP)

即時傳輸協定(Real-Time Transport Protocol,RTP)[5]被設計用來傳送即時(Real-Time)資料時使用的通訊協定，屬於 OSI Model 應用層(Application Layer)協定的一種。

通常像 VoIP 這類的即時應用程式，在傳輸效率的考量上，會有比較高的要求，因此幾乎都是以 UDP 為底層來傳送封包。UDP 雖也提供了簡單的檢查機制，但它太過精簡以致無法對上層的應用程式提供較有用的資訊，例如封包遺失及失序等問題。因此 RTP 設計一些有用的欄位來解決它。圖 10 是一個標準的 RTP 封包格式：

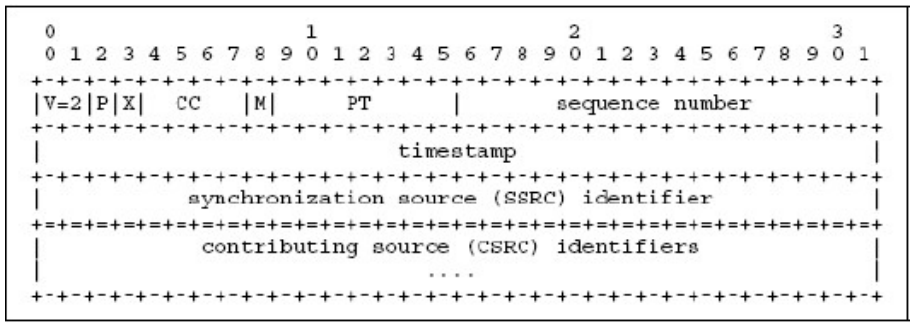


圖 10 RTP Header

其欄位的代表意義及長度如下：

1．Version(V):2bits，記錄 RTP 的版本資訊。

2．Padding(P):1 bit，若此欄位的值為 1，代表此 RTP 封包的 Payload 包含了 padding 的資料，padding 會在 Payload 之前補上 0 以符合某些加密演算法需要的資料長度。

3.Extension(X):1 bit,若此欄位值為 1,表示此 RTP 封包有 header extension。

4．CSRC count(CC):4 bits，此欄位記錄 CSRC identifier 欄位的長度。

5．Maker(M):1 bit，此欄位由 RTP profile 定義。

6．Payoload Type(PT):7 bits，此欄位表示此 RTP 封包的 Payload 編碼，例如使用 G.722 codec，PT=4。

7．Sequence number:16 bits：此欄位表示該 RTP 封包的序號，由傳送端產生。

8.timestamp 32 bits：記錄該 RTP 封包的時間戳記。

9.SSRC 32 bits:表示 RTP stream 的來源，每一個 RTP 封包都會用 32bits 的資料來辨別其 RTP stream 來源，若兩端點的 RTP 封包來自不同的 RTP session 就不會有相同的 SSRC。

10.CSRC list：會有 0 至 15 個項目，每項 32bits，由於不同來源的 RTP 封包會因為 mixer 的機制混合成為同一個 RTP 封包，而 RTP 封包會將這些尒同來源的 SSRC 結合成 CSRC 列表。

常見的 RTP 應用有：

1. Simple Multicast Audio conference

IETF 中的一個工作群討論過Multicast 的可行方法,他們定義了幾個 Multicast address 和port，其中一種方案就是利用RTP 來完成，也就是利用 RTCP的port 來進行之. Audio 會談應用是每個成員在固定週期會將audio data 放入一chunk 中，每個在chunk 中的audio data 都使用RTP Header，RTP header 和data會轉換成UDP 的封包再送出，RTP header 在每個封包中會包含此audio data 使用何種encoding 方法，故sender 可以在會談中視網路狀況來改變 encoding 方法，以達到較好的效能. RTP header 中還包含了 timing 的資訊和 sequence number，讓receiver 可以依此來重建收到的資料。基於會談群組中的成員可能會加入或退出，故必需對現在成員的資訊要有相當的了解，亦包括其目前的接收狀況,故audio 的應用程式要在固定的週期中使用RTCP 進行multicast，來對其他成員報告其狀況，而其他成員也可以參考RTCP的report 來對傳送的 audio data 進行適當的encoding。

2. Audio and Video Conference

RTP 也可使用於Audio 和Video 共同使用的環境,但若要同時使用audio 和 video 的交談，則必需將audio 和 video 分為兩個不同的RTP sessions 和RTCP 封包，並使用不同的UDP port 和/或 multicast address. 因為RTP 本身並無提供可以合併audio 和 video 的session。

3. Mixers and Translators

若在考慮會議成員是在一個低速的連結中要去跟高速的內容提供者做影音傳輸，內容提供者不能單單為了這個低速成員的需求，將影音的encoding 調低以配合其低速的連結，因為這樣會影響到其他較高速的成員，讓他們也相同的接收到較低品質的影音資料，故此時就需要一個元件來應對. mixer 可以置於低速端,功用為重組收到的audio 封包,將其重建為較低頻寬可使用的encoding 資料並傳送給低速的成員，並藉RTP header 的特性，讓接收者可以明確的知道mixer 的存在並和其建立良好的互動，以保障傳輸品質。有些會議成員可能本身連接到高速的連結，但不能直接收到IP multicast，比方說其位址是位於應用程式等級

的防火牆內，在此狀況下，而此防火牆不會讓任何的IP 封包通過，故mixer 在此就沒有其必要性了，真正需求的是一個稱為translator 的原件; translator 必需有兩個,一個位於防火牆的外面,一個位於防火牆的內部,外部的translator 收到 multicast 封包後，將其包裝成可被防火牆信任的格式，再行送出，在通過防火牆後，防火牆內的translator 會將其解開，再以multicast 送出。mixer 和 translator 亦可被用於在多種不同的需求上，比方說 mixer 可以將同傳給一個人的數個不同的video stream 結合成一個stream，來達到群體鏡頭的效果。

2.3.3 Real-Time Control Protocol(RTCP)

RTP協定僅負責資料流的傳送，在品質上的控制，則必須仰賴與RTCP[5]的合作。Real Time Control Protocol(RTCP) 是基於在週期性的傳送控制封包給會議的所有的參與者，通常我們使用(RTP連線埠+1)為RTCP的連線埠。RTCP 最主要有此四種功能：

1. 第一點,也就是其最主要的功能,就是提供傳送端一個資料傳輸的品質回報，此為RTP 協定中最不可或缺的一環,因其和流量及壅塞控制的程序有相當大的關係，此回報會直接對encoding 的控制有相當大的幫助，在IP multicast 方面，由實驗的結果也可以得知，在解決傳輸錯誤上，是有非常的重要性來從收到的回報中來解決傳輸的問題，會議所有的成員都會提出回報，也有助於傳送者可以明確的找出問題發生的點是位於區域端，還是全域端. 在系統提供者的方面，其可以提供一個元件位於網路中，其並未實際的參與會議，但亦會週期性的回報傳輸品質給傳送端或接收端,來幫助其有效找出傳輸問題點,此元件稱為third-party monitor。

2. RTCP 會帶著展現層等級的識別資訊，RTP 稱此為CNAME，在RTP 的SSRC識別資訊可能因衝突或程式重新啟動而改變的狀況下,CNAME 可以讓其繼續和其他成員繼續通訊,另一方面,接收端也可以藉著CNAME 來對數個datastream 做分辨，來判斷其屬於何個session。

3. 上述兩個功能需要所有成員傳送回報，故也可以藉此來對傳送的速率進行控制，藉由所有成員的回報，可以知道成員的個數，並可以藉此計算出傳送時的最佳速率為何。

4. 最後一點，選擇性的功能，其提供小量的session 控制資訊，比方說成員的辨別碼可以在使用者的界面展示，其提供了一個loosely controlled session 的方法，可以顯示成員會進進出出這個會議的狀態，並藉由RTCP 的回報可以找到特定成員的所在。

我們最後來看看，一個典型的RTCP Receiver Report的封包格式如圖11所示：值得注意的是，在VoIP中經常會參考的是interval jitter這個欄位，我們可以根據這個欄位的值去微調語音封包發送的速度，這麼做可以使得通話的品質更加平穩。

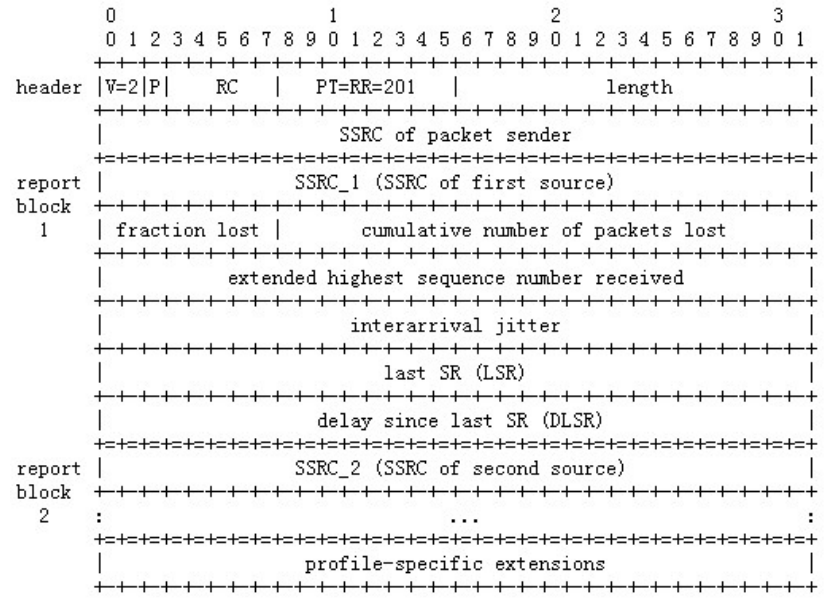


圖 11 RTCP Receiver Report Header

2.3.4 網路電話的安全機制

我們在先前的小節中介紹過網路電話的一些基本協定，包括 SIP/SDP、RTP、RTCP，這些協定的相互合作已經足夠讓我們完成一通網路電話。而在這個小節中，我們將進一步介紹與網路電話安全相關的協定[17]，包括 SDES、MIKEY、SRTP。其中 SRTP 是用來幫 RTP 及 RTCP 加密的協定，不過 SRTP 需要一把 TGK(TEK

Generation Key)來導出加密的 TEK(Traffic Encryption Key)，但是它本身並不包含產生 TGK 的機制，所以必須仰賴 SDES 或 MIKEY 來幫它完成交換 TGK 的動作。

2.3.4.1 SDES

SDES(Session Description Protocol Security Description)[16]是最簡單的金鑰交換機制，它被設計成在 SDP 中使用。我們簡單描述它的流程如下：

(1)連線的發起者首先產生一把金鑰。

(2)將這把金鑰及它的相關參數(演算法、生命週期)以明文的方式寫在 SDP 中。

(3)由於金鑰係以明文方式書寫，所以必須搭配 S/MIME 傳送訊息。

一個 SDES 的例子如圖 12 所示，注意粗體字的部份：

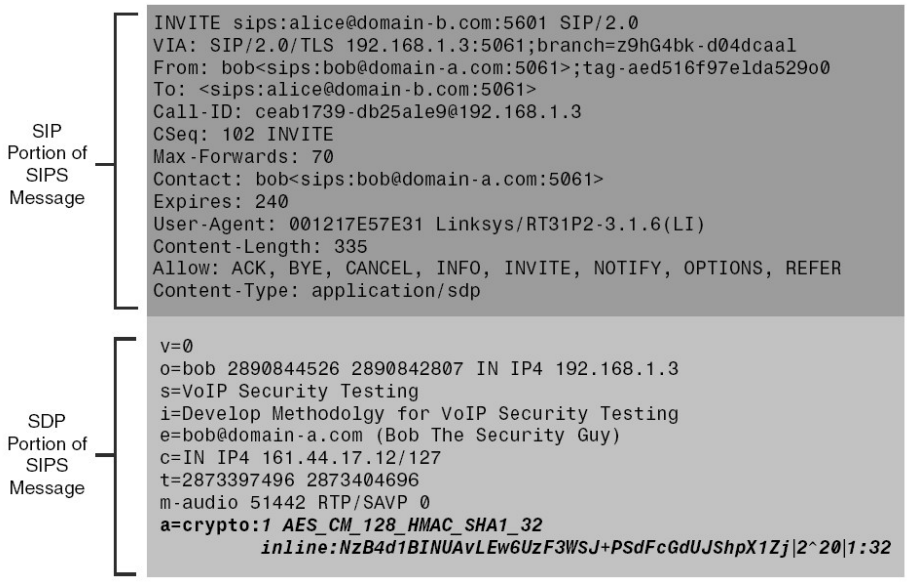


圖 12 在 SIP 中加入 SDES

2.3.4.2 MIKEY

MIKEY[9]的全名是 Multimedia Internet KEYing，它是一種上層的抽象協定，並不是像 SDES 那般限定使用 SIP 來實作。它有三種金鑰交換方式，包括：

(1)PSK(Pre-shared Secret Key)

(2)PKE(Public Key Encryption)

(3)Diffie-Hellman(DH) key exchange

它們的目的都相同，便是在訊息的一次來回(one round trip)之間交換 TGK 的相關訊息，我們以圖 13 展示 PSK 的運作流程(PKE 及 DH 運作流程相似，僅是參數不同，但必須有 PKI 的支援)：

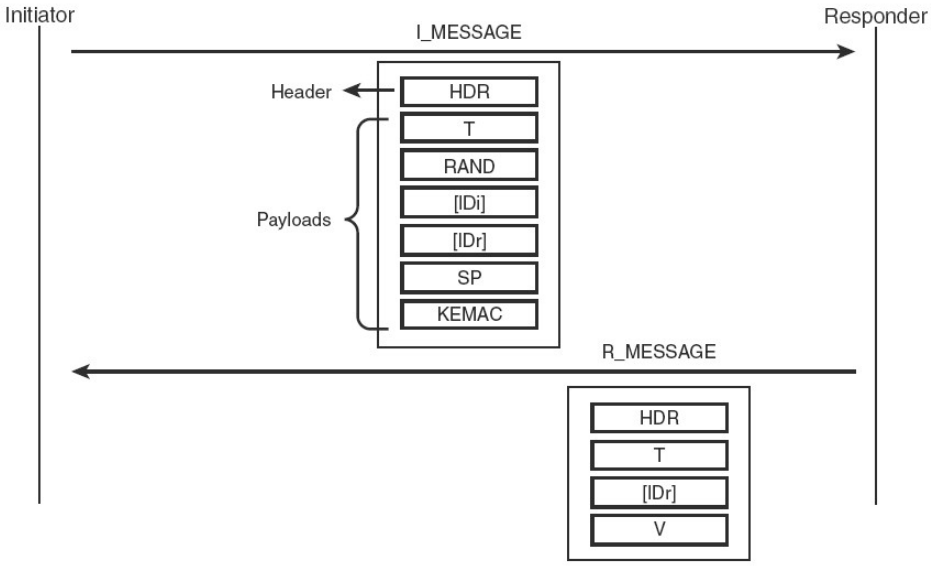


圖 13 MIKEY PSK

圖片來源 文獻[16]

在 PSK 中，雙方已經事先設定好一把共用的金鑰 encr_key。我們在這圖 13 中必須特別注意的是 SP 及 KEMAC 這兩個參數，SP 的意思是 Security Policy，必須指明使用的是哪一種加密機制(ex.SRTP)及相關參數，而 KEMAC 的值是 E(encr_key,{TGK})||MAC，也就是加密過的 TGK。至於其它參數，礙於篇幅的關係，我們並不在此說明，完整內容可以參考 RFC3830。

2.3.4.3 SRTP

SRTP[11]的全名是 Secure Real-Time Transport Protocol，被定義在 RFC3711 中，它被設計出來的目的是用來保證 RTP 及 RTCP 的安全。它可以根據 RTP/RTCP 的表頭資訊、TGK 及 AES 演算法，導出加密的 key stream。並可調用基於散列的消息認證碼(HMAC)，應用 SHA-1 演算法用於認證功能。(這兩個功能可以分別選用或合用)

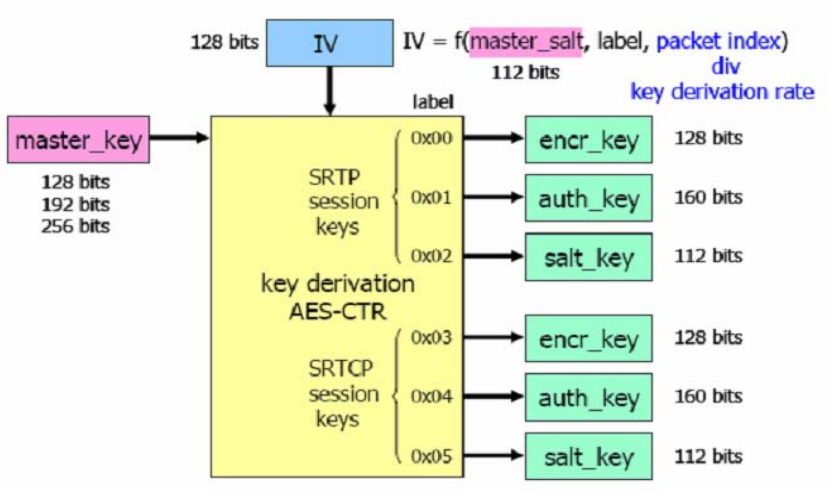


圖 14 SRTP key derivation

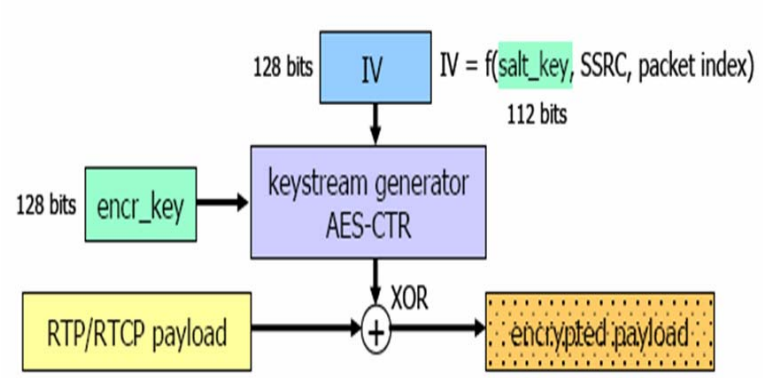


圖 15 SRTP encryption cipher

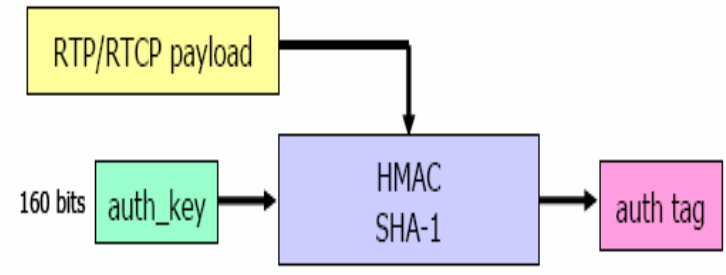


圖 16 SRTP authentication cipher

圖片來源 www2.cma.edu.tw/u_edu/dep_comp/演講 95 年/951011.ppt

2.4 數位簽章(Digital Signature)

數位簽章又稱為電子簽章，一種類似寫在紙上的普通個人簽名或蓋章，但使用公鑰加密(public key encryption)領域的技術，是一種用於鑒別數位訊息的方法。

數位簽章和傳統簽章最大的不同點，除了所欲簽署的文件之形式不同外，印章或親筆簽名與該文件內容是各自獨立且無關的，換言之，針對不同的文件，簽署者使用印章或親筆簽名所產生的簽章並不會隨著文件內容的不同而有所不同。而數位簽章則不同於實體簽章，即便是同一位簽署者，若簽署之文件不同的話，所產生的簽章亦不會相同。

一套數位簽章通常定義兩種互補的運算，一個用於簽名，另一個用於驗證。在這個章節，我們敘述數位簽章所必需使用的兩種演算法，包括雜湊演算法及簽章演算法，及它能夠達到所謂資料完整性(data integrity)、來源鑒別(Authentication)、以及不可否認性(no-repudiation)的目的。

2.4.1 雜湊演算法(Hash Algorithm)

雜湊，也經常被稱做摘要(Digest)。雜湊演算法就像數學式裡的 f(x)函式，當你以一串長度不等的數位內容做為輸入時，它藉由將輸入資料打亂、混合，再輸出成一串長度固定的值(此長度視會演算法而有所不同，例如使用 MD5 輸出 16 個 bytes，而 SHA-1 則輸出 20 個 bytes)。這個輸出值，被稱為是原始輸入資料的雜湊值(hash value)。

雜湊值可以視為一個檔案的電子指紋(digit digest)。在現實世界裡，我們每個人都擁有不同的指紋，有多少個人就會有多少種不同的指紋，電子指紋的想法亦是如此。一個設計良好的雜湊演算法，必須要求在輸入域中很少出現雜湊衝突(hash collision)，如此一來，這個雜湊值才足以代表該原始的輸入內容。概括來說，所有的雜湊演算法都必需滿足這個基本特性：如果兩個雜湊值不相同(根據同一演算法)，那麼這兩個雜湊值的原始輸入內容也是不相同的。典型的雜湊函數都有『無限定義域』，比如任意長度的位元組字串，和有限的『值域』。

由於雜湊演算法應用的多樣性，它們經常是專為某一應用而設計的。例如加密雜湊函數假設存在一個要找到具有相同雜湊值的原始輸入的敵人。一個設計優秀的加密雜湊函數是一種『單向』(one-way)操作，對於給定的雜湊值，很難找

出有效率的方法可以計算出原始的輸入值，也就是說要偽造原始資料的難度或成本很高。為加密為目的設計的雜湊函數(ex.MD5)，被廣泛地用作檢驗雜湊函數。例如在下載軟體的時候，就會對照驗證碼下載正確的部份。錯誤監測和修復函數主要用於辨別資料是否被更動。

目前來說， MD5 及 SHA-1 是一般較為常見的雜湊演算法，分別簡介如下。

MD5(Message Digest Algorithm 5)

Ron Rivest 於 1990 年十月以 RFC 的型態發表 MD4，為 MD5 的前身。MD5 於 1992 年四月發表為 RFC1321，為輸入任意長度訊息，分成數個 512 位元區段，運算後產生 128 位元長度的雜湊值。當暴力攻擊法(Brute Force)與密碼破解等議題未成熟前，MD5 是最多人使用的雜湊函數。1994 年 Oorschot 與 Wiener 花一千萬美元設計一台針對 MD5 的配對搜尋機 (Collision Search Machine)，只用 24 天就找出一組配對的資料區段，這說明 MD5 不具有強碰撞抵抗力(Strong Collision Resistance)

SHA-1(Secure Hash Algorithm-1)

安全雜湊演算法(SHA)是由美國國家標準與技術協會(NIST)所發展，並於 1993 年發佈成為第 180 項聯邦資訊處理標準(FIPS PUB 180)，爾後又於 1995 年頒發修訂版 FIPS PUB180-1，通稱為 SHA-1，其設計是以 MD4 為基礎。此演算法輸入值長度不可超過 2^{64} 個位元，處理流程類似 MD5，皆將原輸入資料切成數個 512 位元的區段，計算後的輸出結果為 160 bytes。

太細節的內部運算流程我們不予討論，僅需要把它想像成一個黑盒子就行了。我們在此以 java security package 實作並觀看其結果(為方便表示，轉換成十六進位顯示)如下表所示：

表 4 雜湊演算法實例(一)

輸入內容(以 Unicode 編碼)：這是輸入的第一個訊息	
MD2	86:6F:56:FA:5A:3A:C3:79:D9:4A:F9:FA:51:E5:C7:07
MD5	3F:7B:4A:B8:2D:26:E6:47:4B:0A:0E:E9:06:56:64:36

SHA-1	E2:5F:EA:26:AD:03:95:0C:88:B6:72:64:96:41:18:3A:86:3A:F2:AB
SHA-256	A5:74:45:D3:C8:67:C5:C5:D4:64:9E:D2:79:EC:7A:3C:42:36:86:39:76:47:6C:36:70:EA:6F:32:6F:0D:C3:02
SHA-384	2C:21:67:A3:C7:7E:95:DF:95:5F:54:80:98:E1:08:63:8F:95:00:A5:25:BE:42:A4:96:00:91:6C:6A:E2:C1:5C:AC:9C:38:6A:17:33:51:99:02:65:63:8C:DE:BC:9D:0C
SHA-512	C3:B3:AA:FC:B7:7A:E2:54:03:57:98:C2:20:E2:AF:A3:F1:D9:0A:73:49:CC:7F:1C:0F:55:21:0E:0A:83:FB:30:6F:36:E3:3E:51:7F:BE:BD:2E:96:A0:7D:B0:EA:35:72:20:86:BF:D6:3D:62:98:45:93:87:EF:83:12:A9:B9:19

表 5 雜湊演算法實例(二)

輸入內容(以 Unicode 編碼)：這是輸入的第二個訊息	
MD2	D5:C2:ED:55:B6:AF:36:22:95:13:62:8E:D2:5D:16:DA
MD5	E6:82:21:AE:5B:D2:60:64:CE:D0:42:2D:A7:35:4E:82
SHA-1	29:E0:A3:A7:44:88:0A:65:02:37:4D:39:A8:63:EB:AC:67:13:C7:EB
SHA-256	51:3C:3E:52:B5:C6:70:ED:0E:D0:CE:FE:5C:6F:C5:5C:E2:A8:B4:B8:D9:DF:B2:B9:8F:5B:72:E6:AC:11:0C:75
SHA-384	A7:52:D3:8D:6E:21:D2:71:38:15:0E:38:71:4F:A6:FB:8D:C9:B8:1D:F2:B3:FB:4C:AB:C2:AD:21:3A:C5:86:A2:7F:83:2A:4F:8F:51:46:18:9A:5E:54:34:38:FA:58:14
SHA-512	95:8B:AD:BA:37:E9:0C:A0:F3:03:72:92:4C:F8:5C:C1:ED:55:CC:C4:FE:47:5A:AB:26:BD:02:68:EF:21:50:C1:90:E8:93:71:92:0F:15:7D:91:56:8F:1C:9C:E9:D6:2D:DE:BC:A5:25:43:5A:09:CF:F8:11:85:4A:C9:BE:9C:C5

比較上面兩表列結果，我們可以發現到：以人類的角度來看，這兩個原始訊息僅不過相差一個字，但不論使用的是哪一種雜湊演算法，所計算出的結果都相差極遠。

2.4.2 簽章演算法(Signature Algorithm)

目前實作上的所有簽章演算法，都必須輔以密碼學中的一把公鑰(Public Key)及一把私鑰(Private Key)，此型式又稱為非對稱密碼系統（Asymmetric Cryptosystem)。公鑰及私鑰以數位化的形式儲存在個人電腦或網路中，使用者必須秘密地保存自己的私鑰，並且將公鑰公諸他人（通常會搭配數位憑證)。之後使用者就可以利用自己的私鑰對文件進行簽署;而數位簽章的接收者可以利用該簽署者的公鑰來驗證數位簽章的有效性。

一個安全且有效的數位簽章,除了簽署者必須要以正確且有效的方法來對電子文件進行簽署外,其所產生的數位簽章之有效性亦需要一個合適的驗證方法來驗證。目前較常使用的簽章演算法包括 RSA 及 DSA。

2.4.3 數位簽章的產生與驗證

數位簽章的產生步驟。如圖 17 所示：

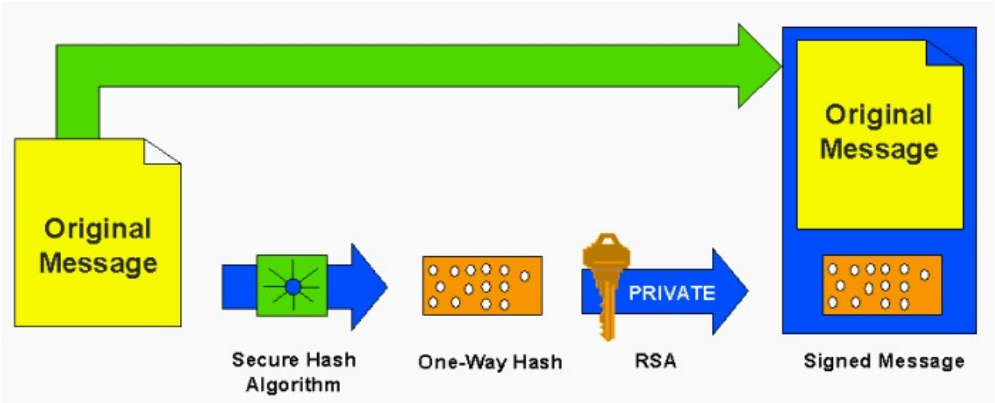


圖 17 產生數位簽章

圖片來源: http://sna.csie.ndhu.edu.tw/~cnyang/HA/sld021.htm

1.假設我們手上握有欲簽章的原始訊息(Original Message)。

2.將原始訊息透過雜湊演算法產生雜湊值，此值在這裡又被稱為 MAC(Message Authentication Code)。

3.我們以 MAC 及自己的私鑰為輸入,利用簽章演算法產出數位簽章(Digital Signature)。

4. 將數位簽章附加到原始訊息的後面，一起傳送給驗證方。

當接收方收到資料之後，可以利用類似的步驟來驗證簽章的正確性，如圖 18 所示：

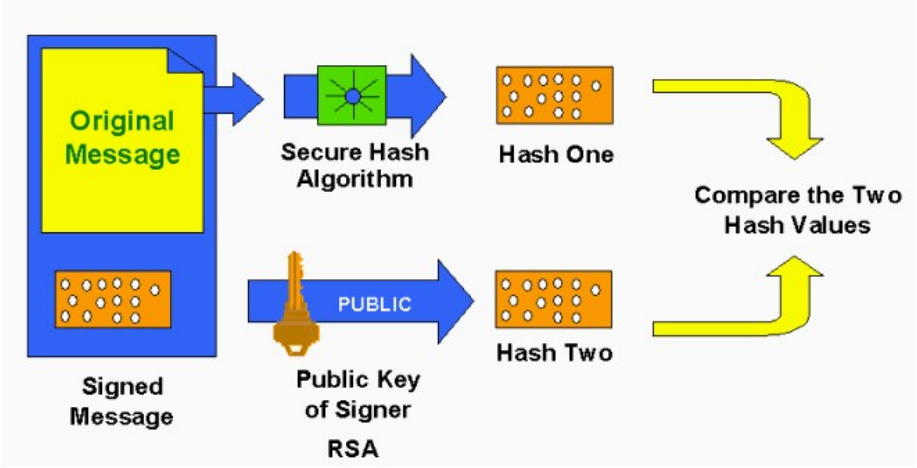


圖 18 驗證數位簽章

圖片來源：http://sna.csie.ndhu.edu.tw/~cnyang/HA/sld022.htm

1．將收到的原始訊息，以相同的雜湊演算法計算出 MAC 值。

2．用簽署者的公鑰，解開數位簽章，得到簽署者計算的 MAC 值。

3．比較步驟 1 及 2 的 MAC 值，若相同的話，則表示訊息及數位簽章都沒有被更改過，且可以認為是簽署者所傳送。

2.4.4 數位簽章的功能

傳送一份使用數位簽章技術的文件，可以達到下列目的：

1．資料完整性(Data Integrity)：文件接收者透過數位簽章之核對可確保此文件的完整性，避免被篡改、重送、遺失。

2．來源認證(Authentication)：文件接收者可確認此文件之發送者的身分，避免被冒名傳送假資料。

3．不可否認性(No-Repudiation)：因為只有文件發送者知道自己的私密金鑰，而且文件具有發送者之數位簽章，使其無法否認發送此文件的事實。

2.5 串流簽章(Stream Signature)

若我們想在 VoIP 中加入數位簽章，來達到不可否認性的目的，有一個很直覺的想法，就是在每個傳送出去的語音封包都加入數位簽章。但就實際情況的考量考，為每一個封包簽章，會造成電腦運算資源 (Computation Overhead)及網路頻寬的浪費(Communication Overhead)。文獻中提到，若使用 Pentium II 300 的電腦，每秒只能簽署 80 個 RSA 簽章，而且還會耗去所有的運算資源。因此必須採用有別於傳統簽章方式的串流簽章。我們研究數種串流簽章演算法，並討論何者最適合使用於 VoIP。

2.5.1 Star Chain

Star Chain[3]的基本想法是，將欲傳送的封包分成若干區塊，經由特殊的方式計算出某區塊的區塊摘要(block digest)，加以簽章後得到區塊簽章(block signature)。這個區塊簽章及一些額外資訊會被附加(append)在該區塊裡的所有封包之後，讓每個封包都能藉由它們來驗證自己。

由於它最初的目的，是希望能讓每個封包都能夠單獨被驗證。藉由在每個封包裡附加能自我驗證的額外資訊，當接收者一拿到這個封包，就能經由計算、比對，立刻驗證該封包。它的優點是可以百分之百驗證，但缺點是它浪費的頻寬很大。我們用圖 19~23 來說明 Star Chain 的方法：

傳送方(簽署者)：

1. 傳送方取出 n 個封包 P1、P2、P3…Pn，將這 n 個封包視為一個區塊(block)

圖 19 Star Chain 流程圖(1)

2. 透過雜湊演算法(MD5、SHA-1…etc)，計算每個封包的數位摘要(Digest)

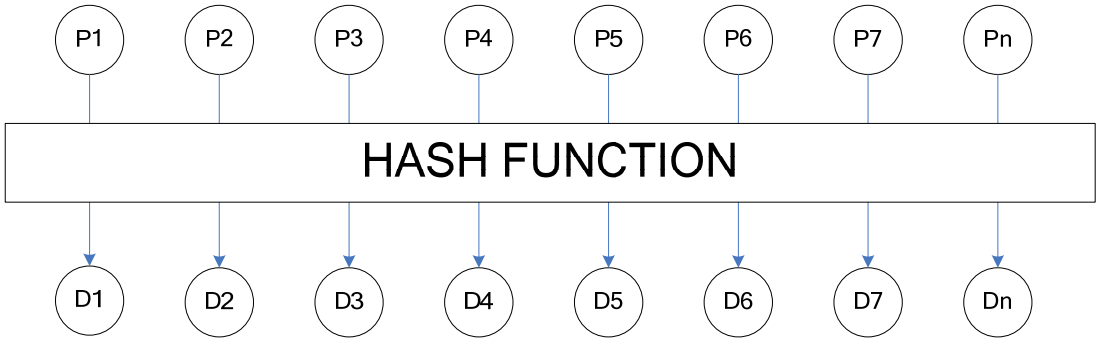


圖 20 Star Chain 流程圖(2)

3. 依序將每個摘要(D1,D2…Dn)當成雜湊函數的輸入值，計算區塊摘要 D1-n【Block Digest】，並用簽章演算法及發送者的私鑰計算區塊簽章 sign(D1-n)【Block Signature】。

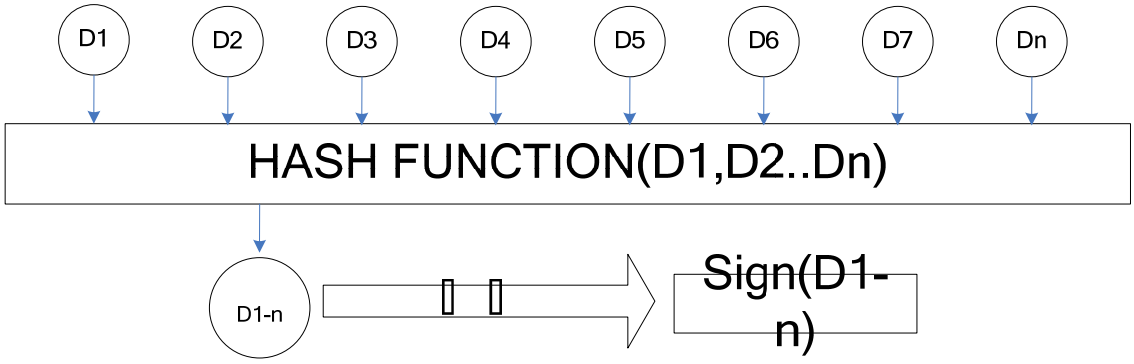


圖 21 Star Chain 流程圖(3)

4. 將每個封包 Pi 重新包裝為 Pi’ ，Pi’ 除了原本的 Pi 之外，還加上用來驗證的 D1、D2…Di-1、D_{i+1}…Dn、Pi 在區塊裡的位置、區塊簽章。

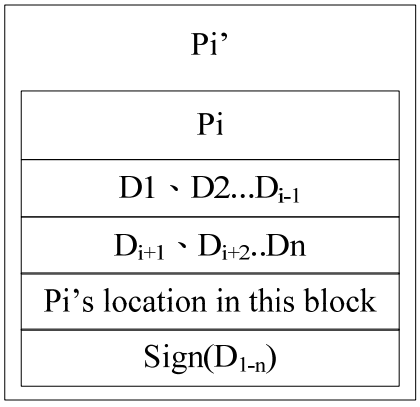


圖 22 Star Chain 流程圖(4)

1 接收方收到這個區塊中第一個到達的封包 Pi’ ，可經由如下步驟驗證：

- 用 Pi 計算 D’i。
- 由於 Pi’ 中有 Pi 在區塊的位置及其 D1、D2…D_{i-1}、D_{i+1}…Dn，可據此計算出 D’1-n。
- 用簽署者的公鑰解開 Sign(D1-n)並和計算出的 D’1-n 比對，若比對結果相同，表示該封包驗證成功。
- 將 D1、D2…Dn 儲存在記憶體中，在收到下一個封包 P_{i+1}’ 時，只需要計算 P_{i+1} 的 D’ $_{i+1}$ 即可驗證該封包。

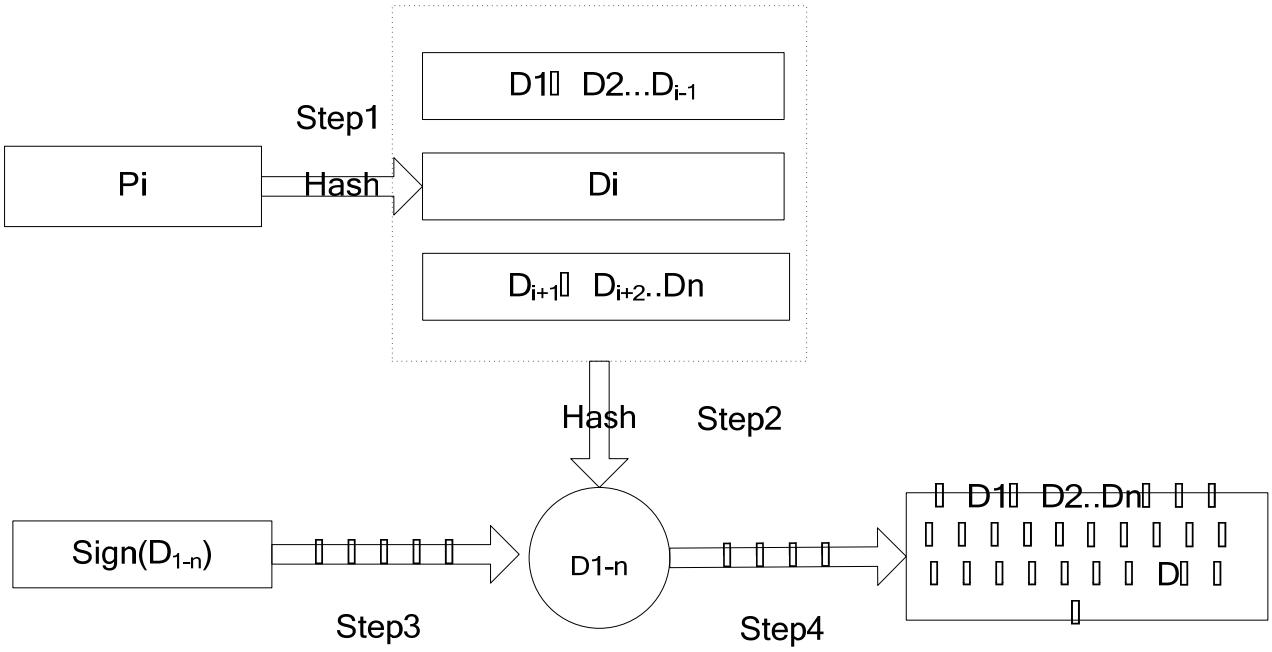


圖 23 Star Chain 流程圖(5)

2.5.2 Tree Chain

Tree Chain[3]是 Star Chain 的特殊形式。在 Star Chain 中，每個封包都必須夾帶相同區塊中，所有其它封包的摘要(Digest)訊息，這對頻寬來說是很大的浪費。Tree Chain 修改 Star Chain 的連結方式，它犧牲一點驗證端的運算資源，讓每個封包夾帶的摘要數量變得較少。

我們用圖 24~26 來說明 Tree Chain，為方便說明，以 8 個封包為一個區塊為例：

傳送方(簽署者)：

1. 依下圖的方式，建構一棵認證樹(Authentication Tree)，計算在這棵樹中所有節點的摘要，並對區塊摘要 D1-8 做簽章得到區塊簽章 Sign(D1-8)。

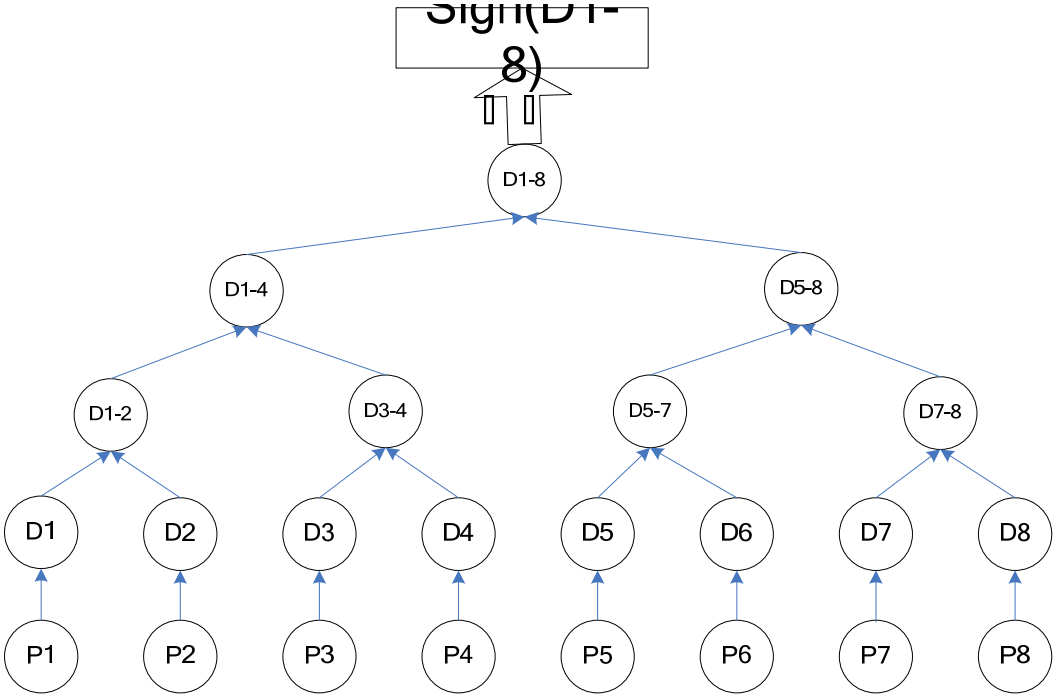


圖 24 Tree Chain 流程圖(1)

2. 以 P3 為例，畫出從 P3 到根節點(Root)的路徑，這條路徑上的節點包括(D3、D3-4、D1-4、D1-8)，如虛線所示。

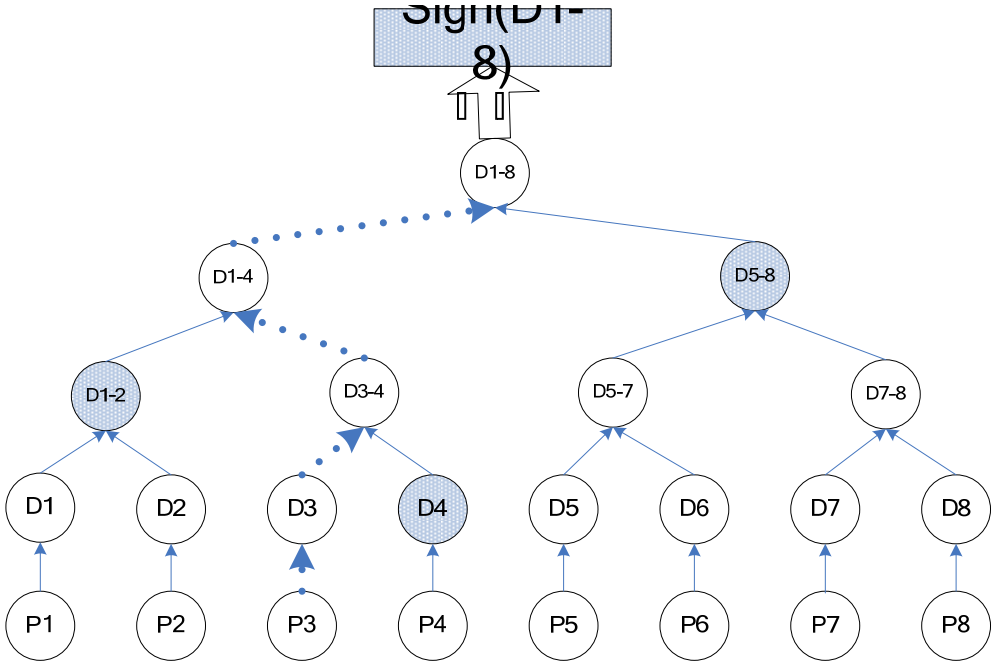


圖 25 Tree Chain 流程圖(2)

3. 將路徑節點的兄弟節點(sibling node)及區塊簽章附加到 P3，封裝成 P3’送出。

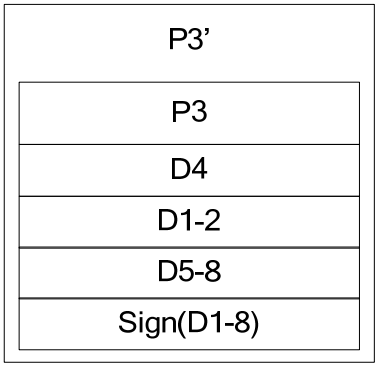


圖 26 Tree Chain 流程圖(3)

接收方(驗證者)：

若 P3’是接收方收到該區塊的第一個封包，則可依如下方式驗證：

1. 用 P3 計算 D’3，用得到的 D’3 及 D4 計算 D’3-4…如此依序計算，最後可以得到 D’1-8。
2. 用簽署者的公鑰解開 Sign(D1-8)並和 D’1-8 比對，若相同則表示該封包可被驗證無誤。
3. 將 D4、D1-2、D3-4、D1-4、D5-8、D1-8 儲存在記憶體中，若下一個收到的是 P4’封包，則只需要計算該封包的摘要 D’4 並和儲存在記憶體中的 D4 比對即可。

2.5.3 Efficient Multi-chained Stream Signature(EMSS)

EMSS[1]將一個簽章的成本，分攤給若干個之前發送出去的封包。它的基本想法如圖 27 所示：令封包按發送順序依次為 P1、P2、P3、P4…Pi…etc，計算出每個封包(Pi)的雜湊值 h(Pi)，並將該雜湊值附加到之後的封包(Pj)中，此時 Pj 被稱為是 Pi 的 supporting node，因為如果 Pj 可以被驗證，那被附加在 Pj 裡的 h(Pi)就可以用來驗證 Pi 。若將 Pi->Pj 這條線看成是一條 edge，| i-j |則用來表示這條 edge 的長度。如此串連下去，最後會形成若干條可用來驗證的路徑。在驗證某一點時，若從該點展開的任一條路徑中，有任一節點是被簽章封包，即表示該封

包可被驗證。

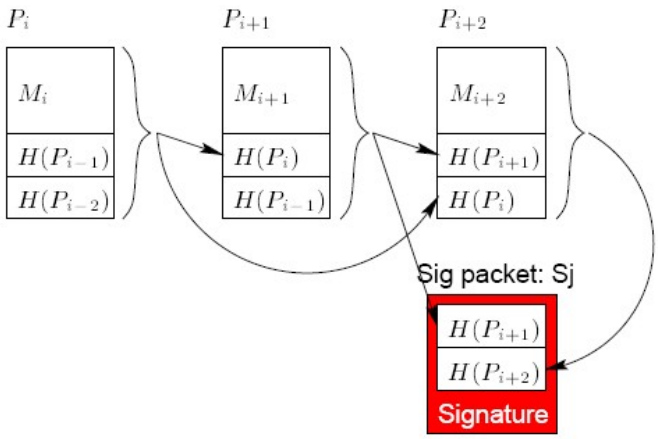


圖 27 EMSS Authentication chain

倘若驗證方已經收到了 n-1 個封包，現在收到被簽章過的封包 Pn，則可依圖 28 的方式展開驗證網路。

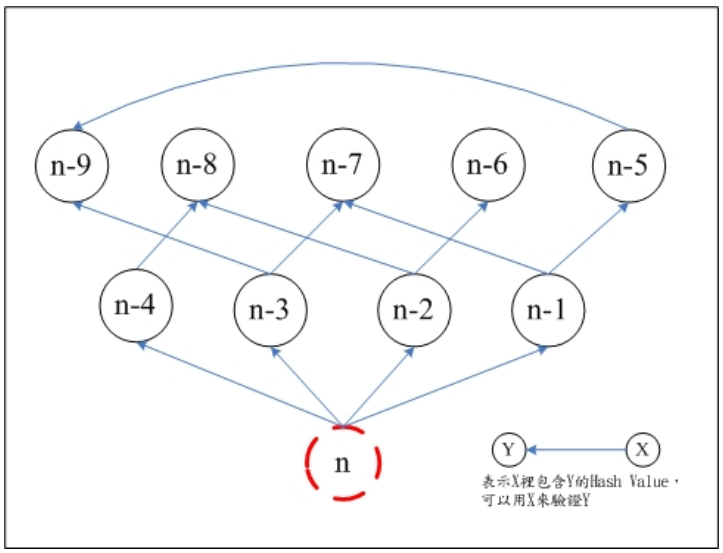


圖 28 EMSS 展開圖

影響 EMSS 驗證率的參數包括下列四點：

- Number of edges per node
- Length and distribution of edges
- Frequency of signature nodes
- Number and distribution of incoming edges in signature nodes.

分別解說如下：

Number of edges per node：如圖 29 所示，每一個 node 有兩個連出去的 edge。

表示每個封包會將自己的雜湊值附加到之後的兩個封包。這個值越大，能建構出的驗證網路也越強，但相對的會造成頻寬上的負擔。

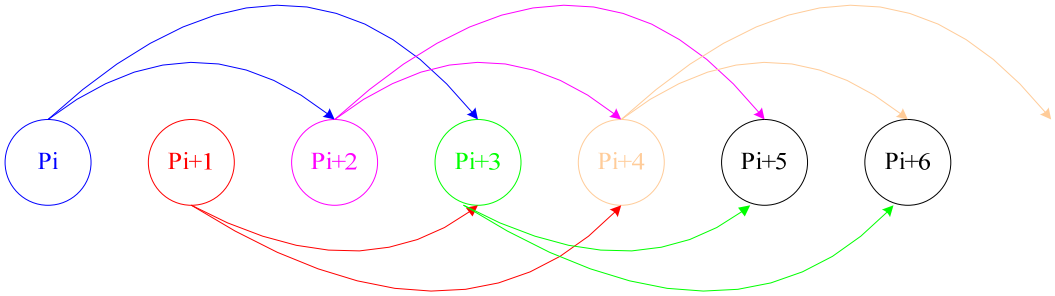


圖 29 EMSS 參數 Number of edges per node

Length and distribution of edges：如果我們用 Pi{s1,s2,s3…}來表示 Pi 有 edge 連到 P(i+s1)、P(i+s2)、P(i+s3)的話。以圖 30 為例，則我們會有 P1{2,3}、P2{1,3}…etc。

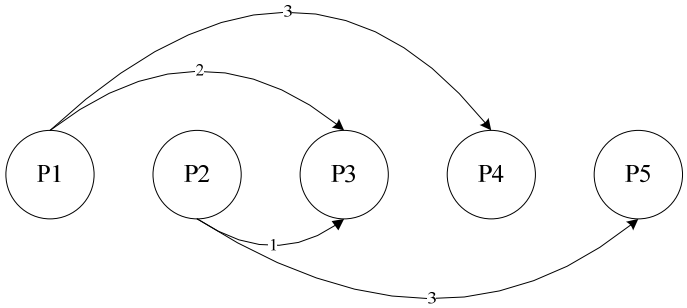


圖 30 EMSS 參數 distribution of length

Length and distribution of edges：指的是在選擇 si 上的策略要用哪一種機率分配模型。Adrian Perrig 等人認為很難導出數學公式去證明出哪一種分配會得到最佳的驗證率，所以他們根據大量實驗並做出如下結論：若考慮到封包的遺失之間具有相關性，建議使用均勻分配(Uniformly distribution)會得到不錯的結果，即給定 U(1,N)，si 的值會介於 1~N 之間。根據他們的實驗，U(1,50)可以得到不錯的結果。

Frequency of signature nodes：送出簽章的頻率，意即發送者多久簽章一次。這個參數會影響 ：

(1)傳送端的電腦運算資源：簽章的頻率越高，耗費的運算資源越多。

(2)驗證端的驗證延遲時間：驗證方必須在收到簽章後才能驗證之前收到的所有封包。

(3)驗證率：送出簽章封包的頻率越高，驗證率會比較高。

Adrian Perrig 等人建議 length of edge 使用均勻分配的另外一個理由是，使用均勻分配得到的驗證率，較不受 Frequency of signature nodes 的影響。

Number of incoming edges in signature nodes.：這個參數表示會在簽章封包裡，夾帶多少個封包的雜湊值。在頻寬足夠的情況之下，它的值似乎是越大越好。Adrian Perrig 等人建議一種決定它數量的方法：**將簽章封包的大小設計成和應用程式使用的封包大小相同**。以下面這個例子來說：若使用 RSA-1024 bit 為簽章演算法，MD5 為雜湊演算法，若應用程式傳送 512 bytes 的封包，那簽章就設計為夾帶【(512-128)/16】個封包的雜湊值。

EMSS 存在另一種變形，稱為 Extended EMSS，能夠在相同的 communication overhead 下得到更好的驗證率。它的想法是使用資訊分散演算法(IDA，2.5.5 小節介紹)將雜湊值分散，而不僅僅是單純附加在別的封包之後。但在文獻裡[8]提到，對每個發送出去的封包都做 IDA 運算，所付出的 computational overhead 會成為實際應用上的瓶頸。事實上，在光電領域中經常被使用的 Read-Solomon codec 通常會被製成晶片以加快運算速度，若是考量到手持裝置的運算能力及供電，我們並不使用這個方法。

2.5.4 Augmented Chain

Augmented Chain[18]的作法類似 EMSS，但它和 EMSS 最大的不同點在於 EMSS 將雜湊值儲存在亂數位置(服從均勻分配)，而 Augmented Chain 將雜湊值儲存在經設計過的關鍵位置，在相同 Communication Overhead 的情況下，Augmented Chain 可以得到比 EMSS 更好的驗證率。

Augmented Chain 的參數有兩個：分別為 a 和 p，這兩個參數會影響它能承受多少個連續封包遺失。在文獻中透過複雜的數學證明推導，它能承受的封包連續遺失數量是$a \times (p-1)$

它的鏈結建構過程分為兩個階段：在第一個階段，使用 a 參數將封包串連在一起，稱做 authenticate chain。串連的規則為(P_i->P_{i+1})及(P_i->P_{i+a})。圖 31 以 a=3 為參數建構出的 authenticate chain。

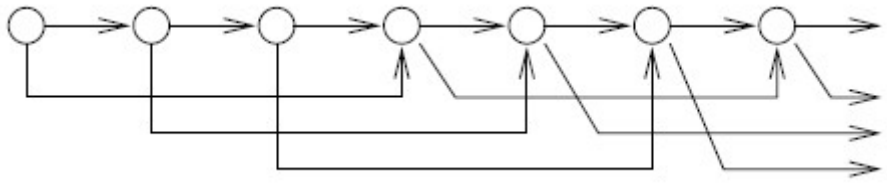


圖 31 Augmented Chain 第一階段

第二階段，在 authenticate chain 中的兩兩節點之間，插入(p-1)個新節點，插入的規則如下：按左、右、左、右…的順序插入新節點，每插入一個新節點，將該新節點的雜湊值給『前面』及『後面』的節點，按此規則直至加入(p-1)個節點為止(如圖 32 所示)。

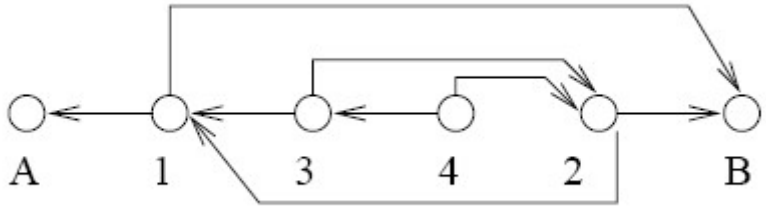


圖 32 Augmented Chain 第二階段

最後，依據實際應用上的需要，在合適的位置加上數位簽章送出。

2.5.5 Signature Amortization using IDA(SAIDA)

SAIDA[8]使用一種稱為 Information Dispersal Algorithm(IDA)的資訊分散演算法，可以大幅降低驗證需要的 Communication overhead，在說明 SAIDA 之前，我們先介紹 IDA 的觀念如下(引述文獻[24])：

資訊分散演算法，是利用線性代數的反矩陣特性所設計的，簡單的說，先將原始資料分散成 n 份(包含其它回復有關資料)，僅需取回其中的 m 份，即可回復資料。詳細說明如下：

假設必須設定欲分散之數目值 n 及還原所需數目值 m，假設欲分散之資料為

F，將 F 依所設定取回的 m 值，分散成 m*[|F| /m]的資料矩陣，以 n 個兩兩獨立的向量(1 * m)乘上資料矩陣，即可獲得 n 份資料區段。當需進行還原時，取回 m 個向量及其產生的資料區段，將 m 個向量堆疊成 m * m 的向量矩陣，進行反矩陣運算，再乘上其產生的資料區段所堆疊的資料區段矩陣(m *[|F|/m])，即可獲得原資料矩陣(m*[|F|/m])，再分解即可還原 F。

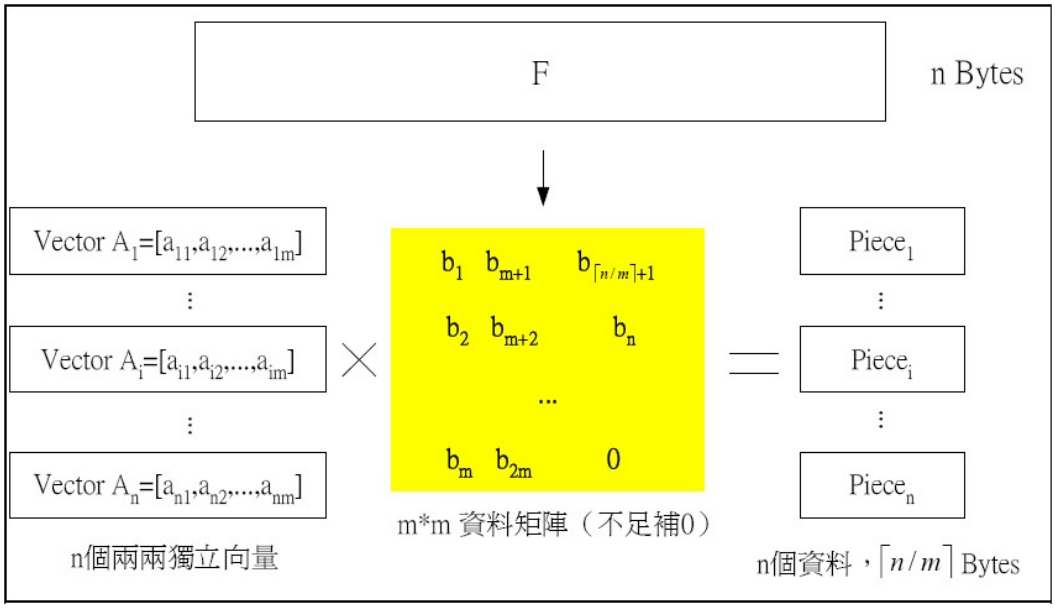


圖 33 IDA 流程圖(1)

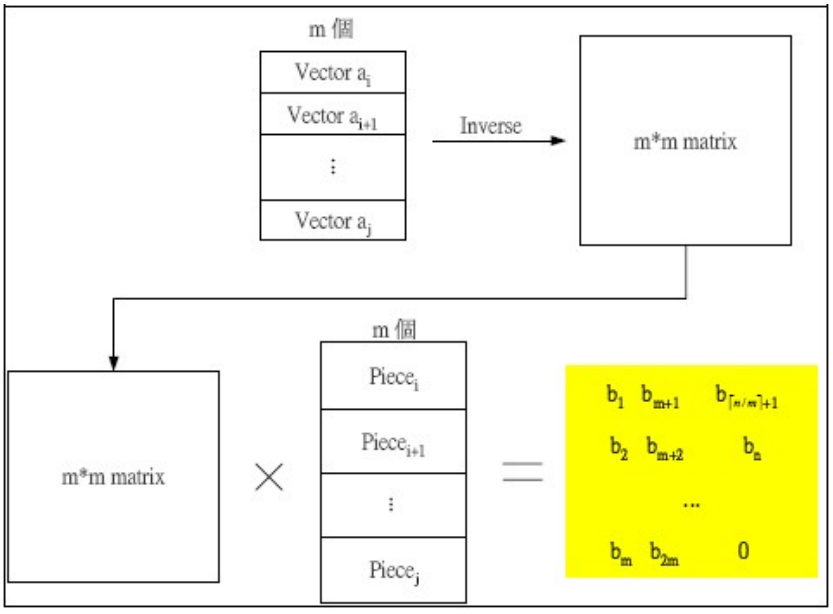


圖 34 IDA 流程圖(2)

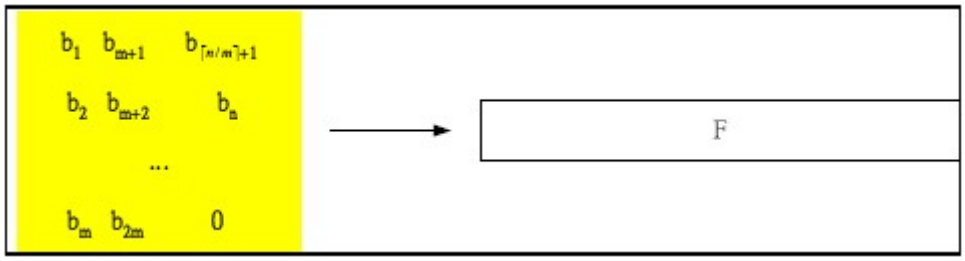


圖 35 IDA 流程圖(3)

有了 IDA 的基本觀念後，我們接著用圖 36~40 來說明 SAIDA 的作法

傳送方(簽署者)：

1. 首先，將 n 個封包個視為一個群組，計算出每個封包的雜湊值。

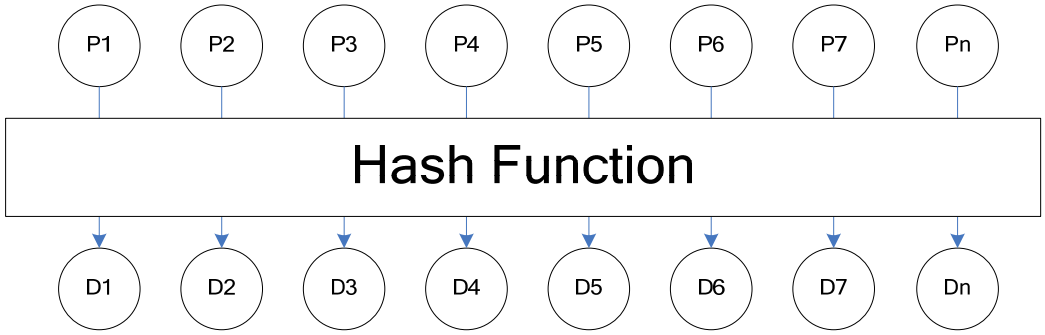


圖 36 SAIDA 流程圖(1)

2. 將 D1、D2、D3…Dn 連在一起變成一個大的 F，將 F 複製成兩份，一份使用傳送者的私鑰簽署，得到 Sign(F)，再用 IDA 切成 n 等份得到 s1, s2, s3…sn。而另一份 F 直接使用 IDA 切成 n 份，得到 f1, f2, f3…fn。

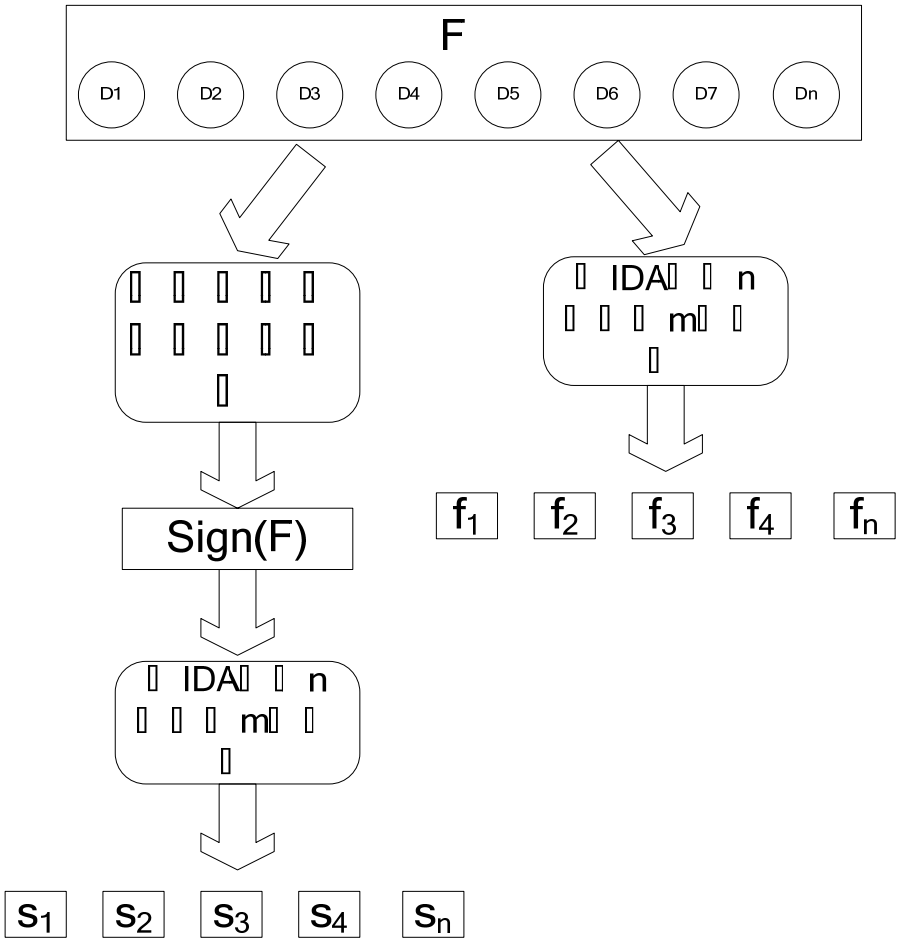


圖 37 SAIDA 流程圖(2)

3. 將 s_i 及 f_i 附加(append)在 Pi 的後面，重新封裝成為 Pi’ 送出。

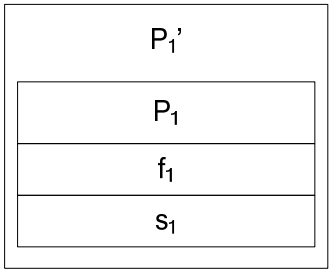


圖 38 SAIDA 流程圖(3)

接收方(驗證者)：

1. 驗證方在這 n 個封包中收到 m 個之後，即可經過 IDA 還原 F 及 Signed(F)，並用使用者的公鑰來驗證計算出的 F’ 是否正確：

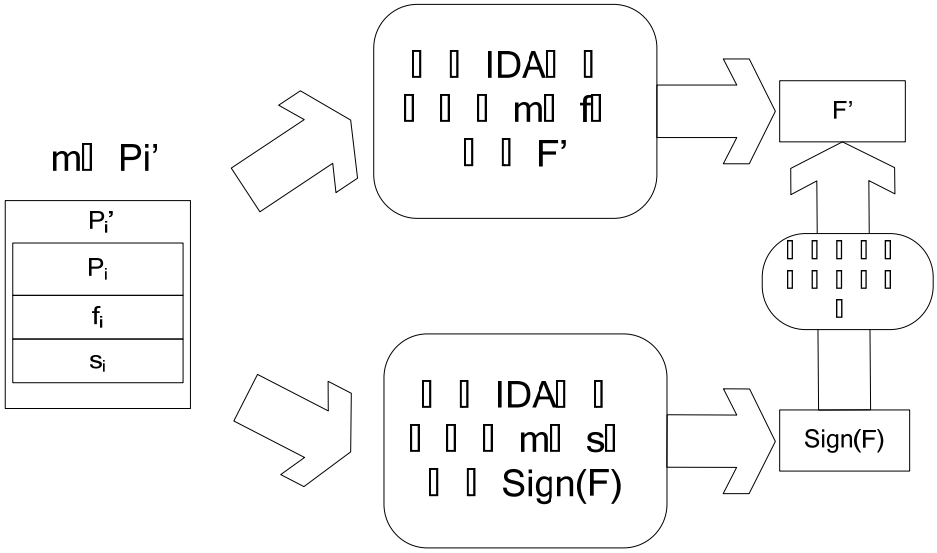


圖 39 SAIDA 流程圖(4)

2. 若 F’ 無誤，則可用 F 來驗證 Pi

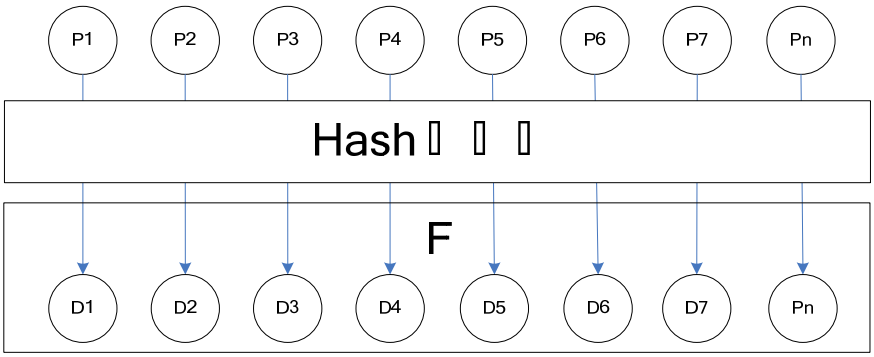


圖 40 SAIDA 流程圖(5)

2.5.6 適用於 VoIP 的串流簽章方法

我們在這個小節中介紹許多串流簽章的方法，其實每一種串流方法各自針對不同的考量點設計(如圖 41 所示)，Tree Chain 犧牲頻寬(Communication Overhead)，換取百分之百驗證率，而 SAIDA 犧牲大量的傳送端延遲，換取在低 Communication Overhead 及高封包遺失率的環境下，也能得到很好的驗證率，可以說是各有優缺點。而在這些方法中， EMSS 很適合使用在 VoIP 中。相對於其它種方法，它最大的優點是擁有很小的傳送端延遲(Sender Delay)，這讓它非常適用於即時產生的資料(data generated in real time)，傳送端在計算完語音封包的雜湊值之後，就可以立即將這個語音封包送出。

再加上它大部份的工作都只在做簡單的雜湊值運算，即使應用在硬體環境較差的手持裝置上也能運作得很好。

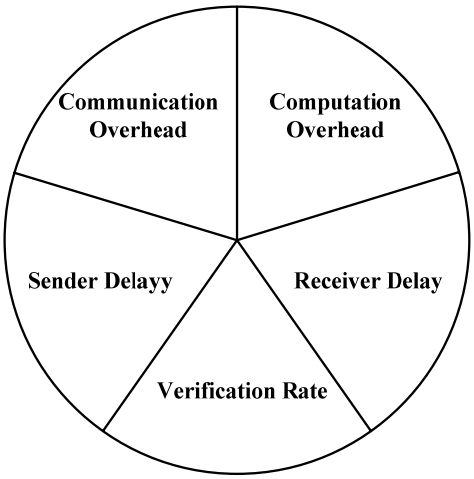


圖 41 設計 stream signature 演算法的考量點

第三章、一個以 RTCP 為基礎的 VoIP 數位簽章機制

3.1 問題定義

我們整理問題如下所述：現行與 VoIP 安全相關的規格書中，都只著重在金鑰交換及加密(Confidentiality)的部份，但我們認為在 VoIP 中加入數位簽章來達到不可否認性(No-Repudiation)確有其好處。然而傳統數位簽章的作法並不適用於 VoIP，雖然它能達到百分之百的驗證率，但付出的代價卻是龐大的 computational overhead 及 communication overhead。因此我們可以採用串流簽章的方法(此時會碰到的第一個問題是：不論是哪一種串流簽章法，它本身並沒有包含交換驗證所需資訊的機制)。並且該方法應該符合下列四項需求：

(1) 不可否認性：這是數位簽章最重要的功能，只有簽署者用自己的私鑰才能簽章，其他人無法假造簽章。

(2) 很小的傳送端延遲：這是在 VoIP 中是必須考慮的一點，因為語音是即時產生的資料，若對產生的語音封包做緩衝的動作，則可能會因回音導致通話品質降低。

(3) 低運算資源負載：若考慮到低階手機的運算能力，及目前手持裝置最難克服的電源問題，那麼低運算資源負載便是考量的重點。

(4) 希望能夠儘量節省頻寬。

3.2 解決方法

一、VoIP 整合 EMSS

我們可以在 VoIP 的架構中整合 EMSS，其中包括兩件工作：

(1) 在連線階段交換驗證 EMSS 的資訊。

就像 SRTP 沒有金鑰交換機制，必須仰賴 MIKEY/SDES 幫它處理一樣，EMSS

本身並沒有牽涉到交換驗證資訊的機制，所以必須在連線階段的時候告訴別人說:『我待會會在我的語音封包中加入 EMSS，驗證所所需的資訊如下...』。

(2) 在語音串流中加入 EMSS 簽章。

我們以圖 42 說明，在語音串流中加入 EMSS 簽章的流程：

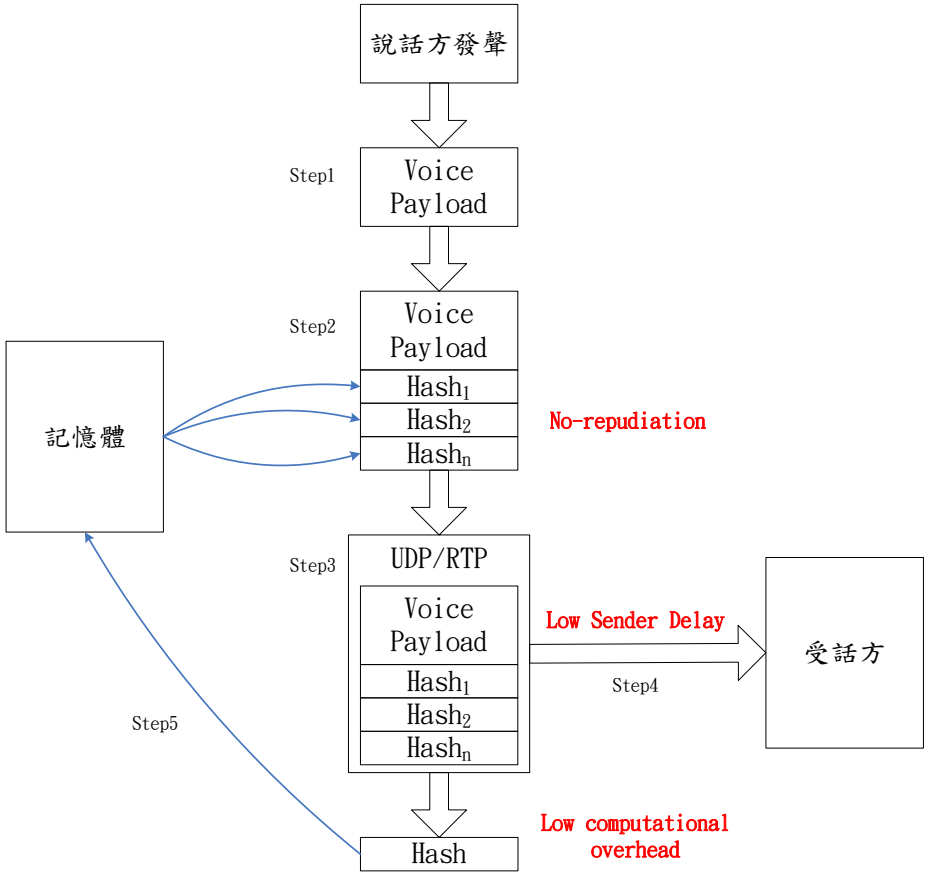


圖 42 在語音封包中加入 EMSS 的流程圖

Step1：VoIP Encoder 將使用者發出的聲音編碼成數位格式，成為 Voice Payload。

Step2：從記憶體中取出之前封包的雜湊值，附加到 Voice Payload 中，成為 RTP Payload。

Step3：將 RTP Payload 加上 RTP Header。

Step4：將 RTP 封包封裝成 UDP 格式傳送到受話方。

Step5：計算該 RTP 封包的雜湊值，並將該值儲存到記憶體中。

我們透過上述流程發現，在語音串流中加入 EMSS，可以達到下列三項目的：

- 語音串流的不可否認性(No-Repudiation) ：EMSS 是串流簽章的一種方法，只有私鑰擁有者才能在語音封包中建構 EMSS authentication chain，認證者使用簽署者的公鑰還原 EMSS 驗證圖形。
- 幾乎沒有傳送端延遲(no sender delay)： EMSS 不需要緩衝語音資料，它在產生語音資料之後，馬上從記憶體中取出之前運算的雜湊值，並附加在該語音資料之後即可送出，並不會有回音問題。

- 低運算資源負載(low computational workload)：EMSS 的大部份工作都只在運算雜湊值，不會對裝置產生太大的負擔。

二、加入 RTCP 動態調整參數的機制

但是當我們考慮到頻寬的負荷時，使用 EMSS 的問題便浮現出來了。主要的原因是 VoIP 的封包普遍都很小(參考下表六)，一個 16 bytes 的 MD5 雜湊值就佔一個語音封包約 6%~14%的大小，因此在每個語音封包中放多少個雜湊值便成了必需要考慮的問題。

表 6 常見的 VoIP codec 表

Codec		Packet Duration(ms)	Packet Size(Bytes)
name	Scheme		
G.711	PCM	10	158
G.711		20	238
G.711		30	318
G.711	VAD	180	82
G.723.1	MP-MLQ	30	102
G.723.1		38.5	108
G.723.1	VAD	180	82
G.723.1	AC-ELP	30	98
G.723.1		45	108

G.726	ADPCM	30	198
G.728	LD-CELP	30	138
G.729A	CS-CELP	20	98
G.729A	CS-CELP	30	108
G.729A	VAD	180	82

用一個情境來點明問題所在：假使我希望驗證率能滿足百分之九十五，如果不知道網路環境為何，為了保險起見，我讓每個語音封包的 number_of_hash(number of edges per node)越大越好，比如說 4，但實際上，目前網路的封包遺失率很低，number_of_hash 只需要 2 就可以達到 95%以上的驗證率，那多出來的兩個雜湊值就造成了浪費。而另一種相反的情況是，若我總是假設網路狀況很好(封包的遺失率很低)，並讓每個語音封包的 number_of_hash 為 2，但偏偏今天網路的封包遺失率很大，那麼我的驗證率將會變得很低。所以最理想的情況，應該是視網路的狀況，來動態調整 number_of_hash 的值才合理。VoIP 中使用 RTCP 的 SR(Sender Report)、RR(Receiver Report)來提供關於 QoS(Quality of Service)的參數，這些參數中包括：兩 RTCP 間隔時間內的封包平均遺失率(average fraction rate)為多少。若我們可以利用這點，來做為動態調整 EMSS 參數值的依據，那就可以節省頻寬上的浪費。

最後我們將(1)VoIP 整合 EMSS (2)及加入 RTCP 動態調整參數的機制整理如圖 43 所示：

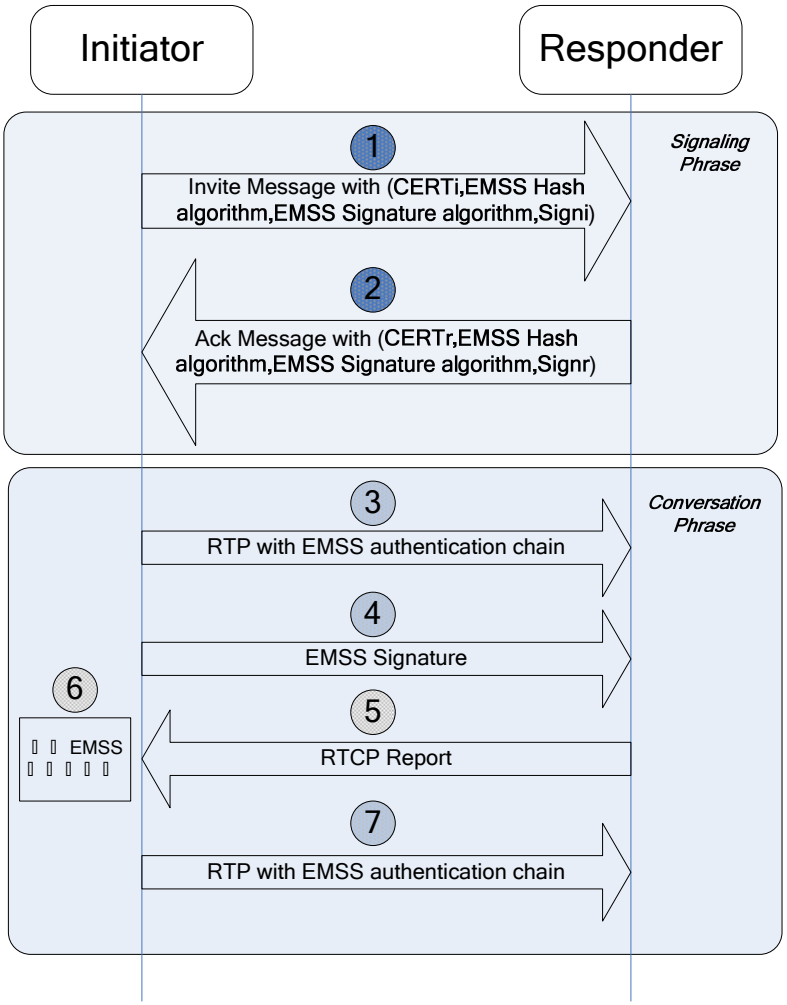


圖 43 在 VoIP 加入 EMSS 及動態調整機制

依序說明如下：

Signaling Phrase：

Step1：在連線建立階段，連線發起者透過 Session Establishment Protocol(SIP、H.323、web service…etc)，將自己的數位憑證及 EMSS 使用的雜湊演算法、簽章演算法，最後加上數位簽章傳送給回應者。

Step2：回應者亦透過相同的方式，將自己的數位憑證及 EMSS 使用的雜湊演算法、簽章演算法，最後加上數位簽章送給發起者。

Conversation Phrase：

Step3：雙方在傳送出去的語音封包中建立 EMSS authentication chain，這點會因為 EMSS 本身的特性滿足 **no sender delay** 及 **low computational overhead** 這兩項需求。

Step4：雙方在談話過程中送出 EMSS 簽章，這點會因 EMSS 本身的特性而滿足 **No-Repudiation** 這個需求。

Step5：雙方在談話過程中，不斷由背景向彼此發送 RTCP report 封包。

Step6：雙方根據 RTCP 中報告的平均封包遺失率去取得預估的雜湊值數量。

Step7：使用新的雜湊值數量為參數，在語音封包中建立 EMSS authentication chain。

Step3、Step4、Step7 我們可以直接使用 EMSS 方法在語音封包中加入串流簽章，關於 EMSS 的鏈結方法我們已經在 2.5.3 小節詳細討論過，在此便不多加贅述。我們接下來將在 3.3 小節討論如何在連線階段交換 EMSS 資訊(Step1、Step2)及 3.4 小節討論根據 RTCP 動態調整參數的機制(Step5、Step6)。

3.3 在連線階段交換驗證 EMSS 的資訊

在 VoIP 中整合 EMSS 很重要的一點是，必須**在 Signaling 階段**交換彼此的(1)數位憑證，及(2)EMSS 使用的雜湊演算法及(3)簽章演算法。驗證者如果有上述三項資訊，便可以解析 RTP 封包的內容並還原 EMSS 的認證圖形。

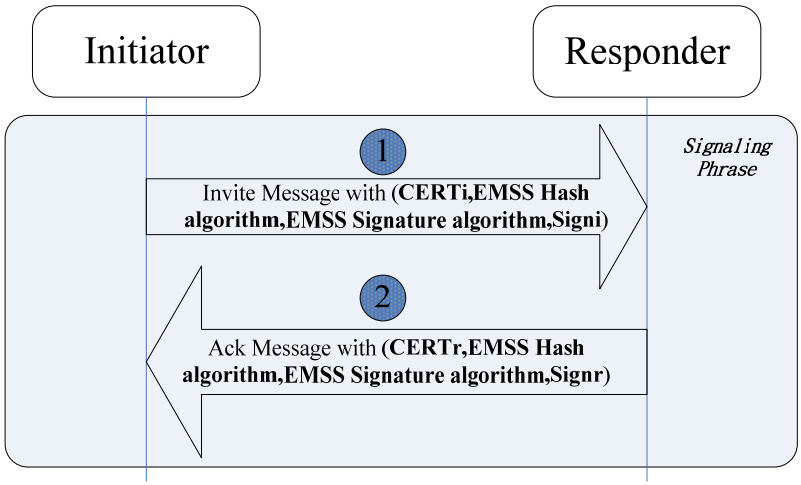


圖 44 在連線階段，傳遞驗證 EMSS 的資訊

我們參考 MIKEY DH-key exchange[9]的想法，設計如圖 44 的協定：會話的參與者們應該在 one round trip 之間交換彼此 EMSS 需要的訊息，而且為了減少建立連線的延遲(latency)，直接在訊息中加入憑證資訊，而不使用 X.500 之類

的目錄服務。在訊息的最後，為了增加安全性，應該加上發送者對該訊息裡重要欄位(如 From、To、SDP)的數位簽章。至於協定的具體實作應該視使用哪一種 Session Establishment Protocol 而定，我們以 SIP 為例來說明：

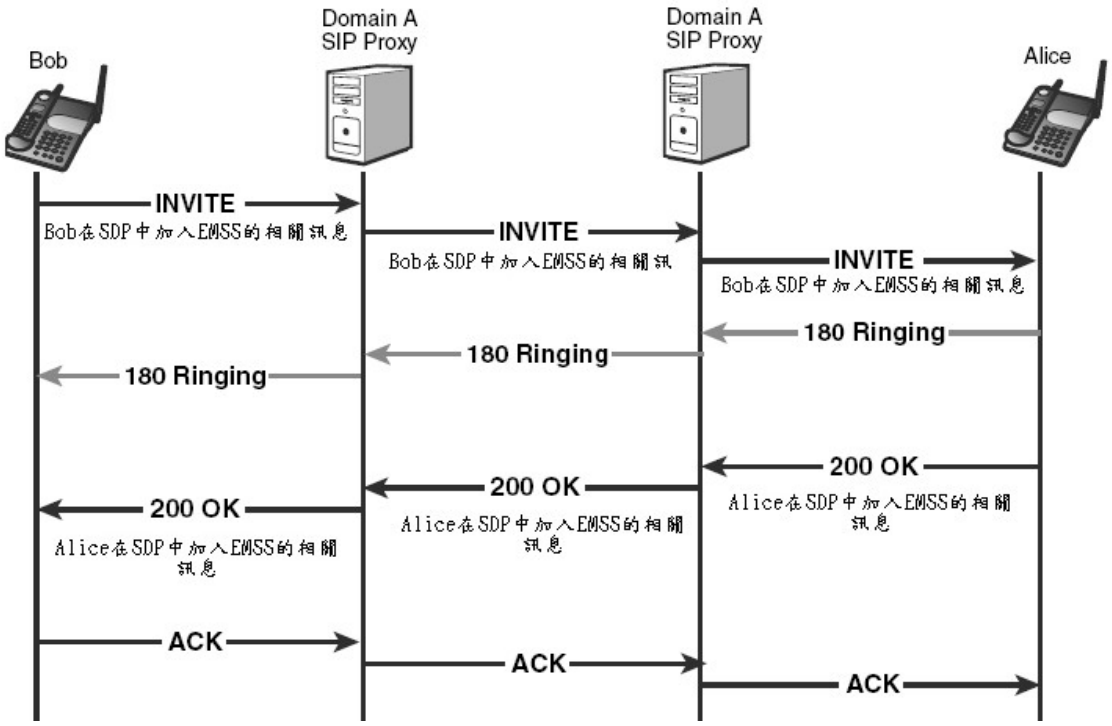


圖 45 在 SIP 中透過 SDP 傳遞驗證 EMSS 的資訊

如圖所示，Bob 在送出去的 Invite 命令裡，加入 Bob 的數位憑證及 EMSS 使用的雜湊演算法及簽章演算法，並且為這個 SIP 命令加上數位簽章。而 Alice 在 200 OK 的回應訊息中，亦帶有同樣的資訊，如此雙方便可以知道對方的公鑰及 EMSS 使用的雜湊、簽章演算法。(補充說明一點，為了避免頻寬的額外浪費，Alice 應在 200 OK 的訊息放入 SDP 回應，而不是 180 Ringing)。若是使用 H.323 或 web service 的話也是相同的道理，只不過在 H.323 中會被編碼成二進位格式，而 web service 則是被嵌在 SOAP 訊息之中。

3.4 根據 RTCP 動態調整參數的機制

我們在 2.5.3 小節中介紹過 EMSS，它擁有四個會影響驗證率的參數，包括：

1. Number of edges per node
2. Length and distribution of edges

3. Frequency of signature nodes

4. Number of incoming edges in signature node

關於(2)這個參數，在文獻中建議使用 U(1,50)的均勻分配，我們並不予以更動。而(3)表示簽署者多久會送出一個 EMSS Signature 封包，這個參數最主要的考量在『希望多久能夠驗證一次』，這點將視實作而有所不同。而(4)參數表示在 EMSS Signature 中會夾帶多少個雜湊值，文獻裡提出一個估計它的方案，那就是將它包裝得和應用程式傳送的封包大小一樣。

最後我們來看看(1)，也就是 Number of edges per node(number_of_hash)，它是影響驗證率及 communication overhead 最大的一個參數，我們在 3.2 中討論過，在 VoIP 中將它的值設定為太大或太少都會造成問題，所以應該要視網路狀況去動態調整它的值才合理。我們提出『根據 RTCP 動態調整的機制』，將它分成下面兩個小節來說明：(1)預估 number_of_hash、(2)動態調整 number_of_hash。

3.4.1 預估 number_of_hash

EMSS 的驗證率和它的四個參數有關：

1. Number of edges per node(number_of_hash)
2. Length and distribution of edges(distribution)
3. Frequency of signature nodes(interval)
4. Number of incoming edges in signature nodes(N)

為方便說明，我們分別給予這些參數括號中的簡稱，在已知 distribution、Interval、N 的情況之下，我們可以利用封包遺失模型及統計學上的假設檢定[23]來幫助我們估計出：『當封包的平均遺失率為 x 時，為滿足驗證率不小於 γ，所需要的最小 number_of_hash 為多少』。

在 2.2.2 節裡，我們介紹過三種可以利用的模型：Bernoulli Model、Gilbert Model、m-order Markov Model。m-order Markov Model 雖然對遺失封包之間的相依性提供最詳細的描述，但我們不需要這些過於細節的資訊。我們可以使用

Gilbert Model，因為它對封包的連續性遺失(bursty)做了不錯的描述，我們以下圖表示 Gilbert Model 的參數範圍：

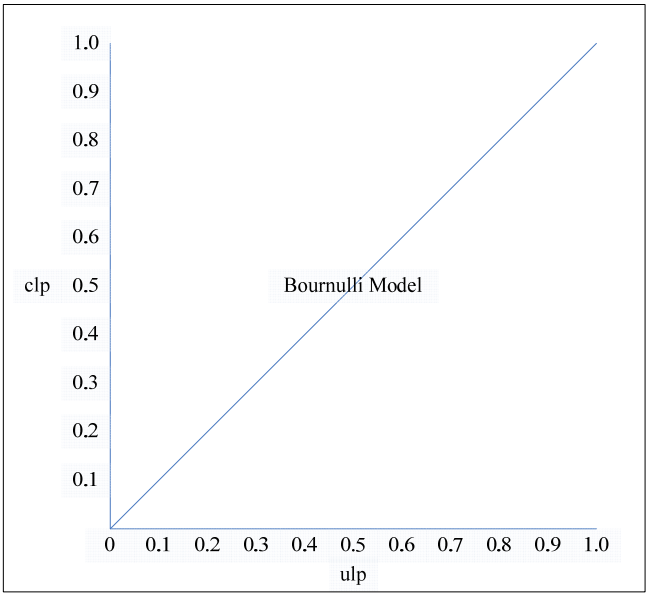


圖 46 以座標軸展開 Gilbert Model

由於真實世界裡的封包遺失通常具有短暫的相依性，所以 clp 的值總是大於或等於 ulp[8]。為了縮小問題範圍，我們進一步將圖形切割如下：(1)假設當網路狀況落在 ulp=0.4 這條線的右邊時，使用者會因為無法接受通話品質而掛斷電話 (2)根據文獻所測量(表一)，VoIP 的數據值約是(ulp=0.16,clp=0.42)，我們假設 VoIP 運行在大部份網路的 clp 值都不會超過 0.8。依上述三項原則將 Gilbert Model 切割如下圖所示：

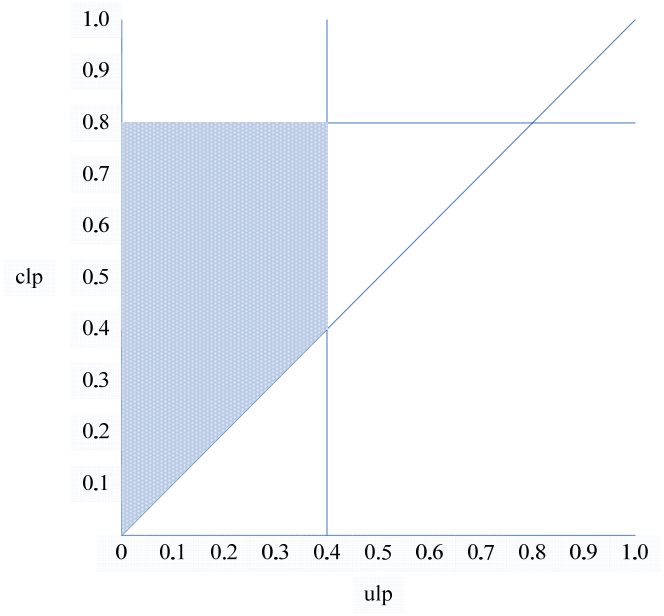


圖 47 以座標軸展開 Gilbert Model，限定環境參數

我們的假設是真實世界的 VoIP 會落在這個梯形裡，而我們的目標是要估計出，『給定另外三項參數 N、distribution、interval，當封包平均遺失率為 x 時，為滿足驗證率不小於 γ ，所需要的 number_of_hash 最小值為何』

首先只考慮這個梯形上的某一點(ex.ulp=0.1,clp=0.5)，如果網路的狀況落在這點，那我們可以用統計[17]的假設檢定方法做如下估計：

Step1：設定該點為網路的環境參數。

Step2：給定 distribution、N、interval。

Step3：給定驗證率的門檻值 γ 。

Step4：令 number_of_hash 的值為 i，分別執行 n 組數據後，計算樣本平均數 $\bar{x}_i$及樣本標準差s_i，在母體未知的情況下，根據統計學的中央極限定理[17]，當樣本數 n>=30 時，未知母體可視為常態分配，並可用樣本平均數$\bar{x}_i$估計母體平均數μ_i。

Step5：給定拒絕域 α ，利用 p-value 右尾檢定法，找出為滿足驗證率 γ 時所需的 number_of_hash 最小值為何。檢定步驟如下：

令虛無假設 H_0: $\mu_i \leqq \gamma$　　vs　　對立假設 $H1$: $\mu_i > \gamma$

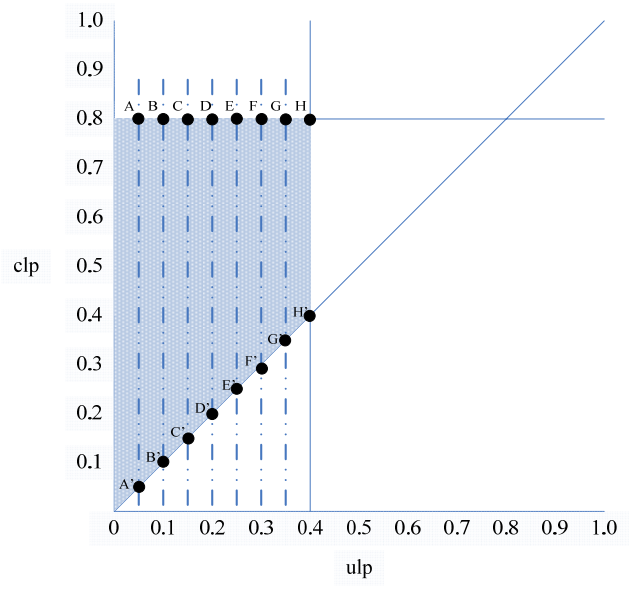
計算$p-value = \frac{\bar{x}_i - \gamma}{\frac{s_i}{\sqrt{n}}}$

給定 α，查表求得拒絕域的 z 值為何，若 p-value 大於 z，則拒絕 H_0，表示該 number_of_hash 可以滿足 γ。

Step6：依序令 i=2、3、4、5、6…，分別執行(4)、(5)，可求出最小的 i 值為何。

若是針對單一點的情況，我們可以用這種方法來檢定最小的 number_of_hash 為多少，但考慮到這個梯形上有無限多點，我們不可能在每一點都做相同的實驗。所以我們可以用另一種計算趨近值的方法：

Step1：將這塊梯形按橫軸等距切割成 m 個區塊(切割的區塊越多，估計得越精準)。



Step2：分別用這 m 個區塊裡的最差點(所謂最差，指的是在其它參數都相同的情況下，該點會造成最低的驗證率，即表示若在該點滿足門檻值 γ，則該區塊的所有點也會滿足 γ)去估計『給定 N、distribution、interval，為滿足 γ 時，這個區塊所需要 number_of_hash 的最小值為何』。

要找出區塊裡最差的一點，我們可以從下面兩個性質著手：

- 在給定相同 clp 的情況下，當要維持驗證率值在某程度時，若 ulp 越大，

需要的 number_of_hash 就會越多。我們可以依此推斷：給定相同 clp，越往右邊(ulp 越大)的點會越差。

- 在給定相同 ulp 的情況下，clp 的值越大，封包遺失率的『變異數』會越大。例如在 B'點時，封包遺失率會很穩定的一直維持在 0.1，但在 B 點時，雖然 ulp 相同，但封包遺失率可能會在 0.7~1.3 之間跳動。我們可以依此推斷：給定相同 ulp，越往上走(clp 越大)的點會越差。

根據上面的兩個性質，以 AA'BB'這個區塊為例，最差的一點就是 B，我們可以用 B 點來代表這個區塊去估計出，為滿足驗證率不小於 γ，所需要的 number_of_hash 最小值為何。

Step3：在對每個區塊都做相同的檢定之後，我們可以得到類似圖 48 的結果：

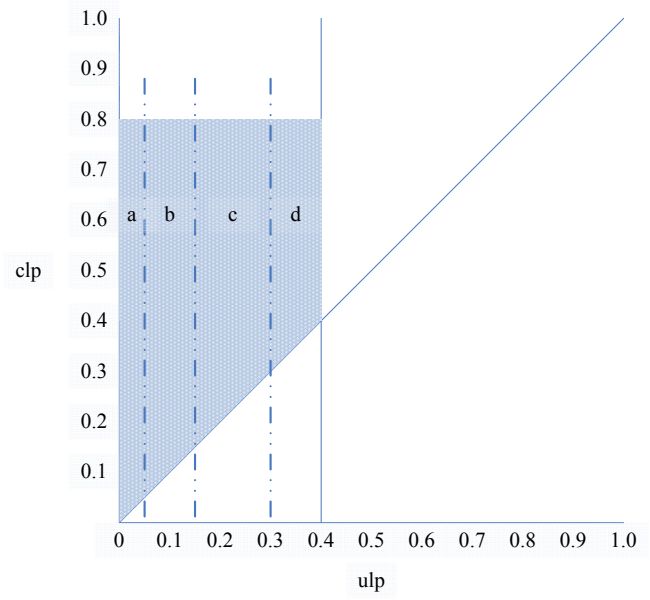


圖 48 預估 number_of_hash 的結果圖

圖 48 的解讀方式如下：給定 interval、N、distribution，當封包的平均遺失率為 0.05 時，為維持驗證率在 γ，我應該讓 number_of_hash 為 a；在平均遺失率為 0.15 時，number_of_hash 應該為 b…以下依此類推。

3.4.2 動態調整 number_of_hash

我們現在可以把 3.3 小節得到的結果套用在 VoIP 中。VoIP 使用 RTCP 來做 QoS 參數報告，SR 和 RR 中有接收端向發送端報告最近一段時間平均遺失率的

欄位，反應目前的網路狀況，但是 RTCP 封包中的遺失率一般不能直接用來做為標識量。首先，在網路發生擁塞時，RTCP 封包不可避免會受到影響，傳輸過程中產生延遲。這樣 RTCP 封包中的遺失率由於傳輸延遲而帶有滯後性，不能完全、及時、準確地反映目前網路狀況。同時，由於網路的不穩定性和負載變化的隨機性，遺失率可能出現較大的隨機波動，突然達到一個較大的值，然後馬上又恢復正常，但實際上並沒有發生擁塞。

為了解決這個問題，一般使用被稱為 low pass filter[21]的過濾器，配合目前收到 RTCP 封包中的遺失率去預測下一個時刻的遺失率為何。low pass filter 的公式如下所述：

- $\lambda_{n+1} = \lambda_n + a(b - \lambda_n)$

表 7 low pass filter 參數說明表

參數值	說明
λ_{n+1}	T_{n+1} 時刻所預測的封包遺失率
λ_n	T_n 時刻所預測的封包遺失率
a	平滑系數，一般來說控制在 1 以下
b	T_n 時刻 RTCP 匯報的封包遺失率。

由上述參數可知，當前封包遺失率的預測值是由前一時刻的預測值加上預測誤差和平滑系數 a 的值，若發生擁塞時 b 會長時間維持在較大的值，λ_n 會逐漸逼近 b。

必須特別注意的參數是 a，它是 low pass filter 的核心參數。a 的值如果太小，則對遺失率的反應會比較慢，發生擁塞時，判斷出擁塞就會相對較晚，無法及時進行調解； 倘若 a 的值過大，雖能對擁塞有較快的反映，但容易將遺失率較大的隨機波動誤認為是擁塞，造成誤判。因此，a 的值要視具體條件而恰當地取值，以提高預測的準確性。在文獻[6]中認為 a=0.3 時，效果會最好。

導入 low pass filter，我們可以根據 3.3 小節估計的結果去做調整，方法如下：

Step1：在一開始通話的時候，由於並不知道目前網路的狀況為何，所以假設目前網路處於最差的狀態，並將 number_of_hash 設為最大值。

Step2：隨著通話的進行，可以透過 RTCP 及 low pass filter 估算網路封包遺失狀況，逐漸調整 number_of_hash 的值，達到最合適的狀態。

3.5 小結

我們最後表列整理第三章的內容如下所示：

表 8 問題的整理與解決方法

<table>
<tr><th>問題</th><th>子問題</th><th colspan="3">解決方法：</th></tr>
<tr><td rowspan="5">VoIP 缺乏以數位簽章來達到不可否認性的相關規範，而且傳統簽章方法並不適用於 VoIP，應該使用串流簽章。</td><td>串流簽章本身缺乏交換驗證資訊的機制</td><td rowspan="5">一個以 RTCP 為基礎的數位簽章機制</td><td colspan="2">在連線階段交換 EMSS 資訊(3.3)</td></tr>
<tr><td>Non-Repudiation</td><td colspan="2">在語音串流中加入 EMSS(2.5.3)</td></tr>
<tr><td>Low sender delay</td><td colspan="2">在語音串流中加入 EMSS(2.5.3)</td></tr>
<tr><td>low computational overhread</td><td colspan="2">在語音串流中加入 EMSS(2.5.3)</td></tr>
<tr><td>low communication overhead</td><td rowspan="2">根據 RTCP 動態調整參數的機制</td><td>預估 number_of hash(3.4.1)</td></tr>
<tr><td>動態調整 number of hash(3.4.2)</td></tr>
</table>

第四章、效能與安全性分析

本章將依據我們所提出的方法進行模擬測試，我們的重點擺放在根據 RTCP 動態調整參數的機制，透過實驗數據探討利用動態調整 number_of_hash 可帶來的好處及可能的缺點為何，並於最後一個小節做安全性分析。

4.1 效能分析

我們在在這裡實作 3.4 小節的內容，包括預估 number_of_hash(4.1.4)及動態調 number_of_hash(4.1.5)，並針對實驗結果在 4.1.6 小節給予評論。

4.1.1 模擬環境說明

模擬實驗使用個人電腦模擬網路電話協定(RTP/RTCP)及語音封包遺失的行為。硬體設備及軟體環境如下。

開發平台：

CPU: Pentium4 1.8 KHz

RAM: 1G DDR RAM

HARDISK: WD 60G

作業系統: Windows XP sp2

設計平台： EclipseJAVA 整合開發工具

程式語音： JAVA

4.1.2 模擬網路的方法

實驗根據文獻[9]的方法(pseudo code 如下)模擬 Gilbert Model，並讓封包延遲時間服從均勻分配 U(50,100)的隨機值。根據 RFC1889 模擬 RTP 及 RTCP 封包，模擬語音封包傳輸的速度為 20ms。

```
define double ulp
define double clp

define double p = (ulp*(1-clp))/(1-ulp)

define q = 1-clp

double event = random(0,1)

switch(state){

case 0:if(event<=p){state =1;}else{send packet to receiver}break

case 1:{if(event<=q){send packet to receiver}else{drop the packet}}

}
```

4.1.3 環境限制

(1)我們必須假設通話雙方背後已經有 PKI 架構的支援。

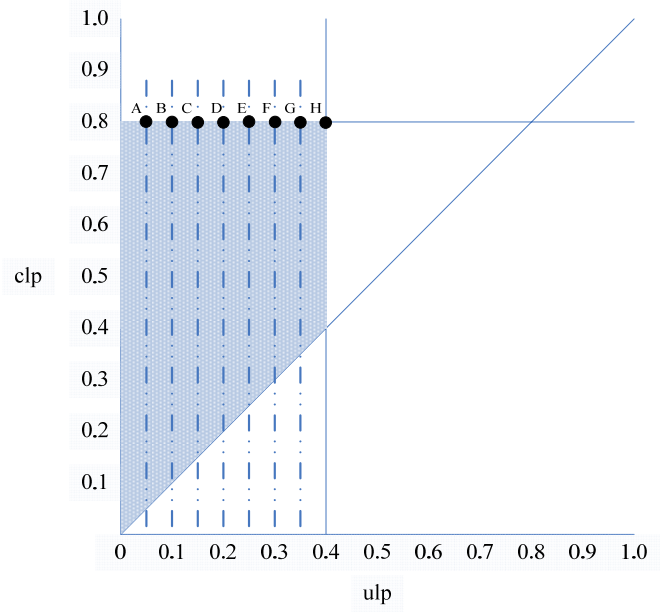
(2)比較早期的 VoIP 可能沒有實作 RTCP，我們在這裡假設使用的是較新的 VoIP 技術，並實作 RTCP。

(3)我們使用 Gilbert Model 模擬網路，並根據 3.4.1 小節提到的三條線切割網路參數。我們的假設是真實世界的 VoIP 大部份會落在這三條線所切割出來的梯形內。

4.1.4 預估 number_of_hash

我們根據 3.4.1 小節的方法，將梯形切割成 8 個區塊(切割得越多，可以得到越精準的估計值)，並且針對每個區塊的最差點，給定 N、interval、distribution(N 及 interval 視應用程式的實作而有所不同)，驗證率的門檻值

γ，估計出為滿足 γ，所需的最小 number_of_hash 為多少。



如上圖所示，將梯形切割成八個區塊後，可以用來估計的八個點分別是：A(0.05,0.8)、B(0.1,0.8)、C(0.15,0.8)、D(0.2,0.8)、E(0.25,0.8)、F(0.3,0.8)、G(0.35,0.8)、H(0.4,0.8)，各點實驗結果如下：

(1)A 點的環境參數如下：

ulp	clp	N	interval	distribution
0.05	0.8	15	10(秒)	U(1,50)

令 number_of_hash 為 2,3,4,5,6 各執行 1000 次，計算驗證率的平均值和變異數，結果如下表所示。

number_of_hash	2	3	4	5	6
組數	1000	1000	1000	1000	1000
平均數	0.965841	0.988773	0.994638	0.997285	0.998321
變異數	0.000617	0.000073	0.000025	0.000009	0.000007

以右尾檢定，門檻值 γ 為 0.95，拒絕域 α=0.95 時，z-value=1.96。

number_of_hash	2	3	4	5	6
P-value	20.1705	143.57212	280.097605	493.40241	568.70756

結果如上表所示，表示在 ulp=0.00~0.05 的這個區塊，為滿足 γ=0.95 的最小 number_of_hash 的值為 2。

(1)B 點的環境參數如下：

ulp	clp	N	interval	distribution
0.1	0.8	15	10(秒)	U(1,50)

令 number_of_hash 為 2,3,4,5,6 各執行 1000 次，計算驗證率的平均值和變異數，結果如下表所示。

number_of_hash	2	3	4	5	6
組數	1000	1000	1000	1000	1000
平均數	0.940694	0.981082	0.992258	0.995712	0.996712
變異數	0.002729	0.001151	0.000057	0.000035	0.001000

以右尾檢定，門檻值 γ 為 0.95，拒絕域 α=0.95 時，z-value=1.96。

number_of_hash	2	3	4	5	6
P-value	-5.633059	28.97718	176.5422	245.6514	46.70922

結果如上表所示，表示在 ulp=0.05~0.1 的這個區塊，為滿足 γ=0.95 的最小 number_of_hash 的值為 3。

(1)C 點的環境參數如下：

ulp	clp	N	interval	distribution
0.15	0.8	15	10(秒)	U(1,50)

令 number_of_hash 為 2,3,4,5,6 各執行 1000 次，計算驗證率的平均值和變異數，結果如下表所示。

number_of_hash	2	3	4	5	6
組數	1000	1000	1000	1000	1000
平均數	0.9004441	0.9713509	0.9884716	0.9939223	0.9965960
變異數	0.0071369	0.0016960	0.0001386	0.0000672	0.0000622

以右尾檢定，門檻值 γ 為 0.95，拒絕域 α=0.95 時，z-value=1.96。

number_of_hash	2	3	4	5	6
P-value	-18.549835	16.394559	103.32327	169.45949	186.8238

結果如上表所示，表示在 ulp=0.1~0.15 的這個區塊，為滿足 γ=0.95 的最小 number_of_hash 的值為 3。

(1)D 點的環境參數如下：

ulp	clp	N	interval	distribution
0.2	0.8	15	10(秒)	U(1,50)

令 number_of_hash 為 2,3,4,5,6 各執行 1000 次，計算驗證率的平均值和變異數，結果如下表所示。

number_of_hash	2	3	4	5	6
組數	1000	1000	1000	1000	1000
平均數	0.8495209	0.9558529	0.9799905	0.9893645	0.9924684
變異數	0.0133133	0.0047488	0.0029612	0.0021673	0.0024312

以右尾檢定，門檻值 γ 為 0.95，拒絕域 α=0.95 時，z-value=1.96。

number_of_hash	2	3	4	5	6

P-value	-27.537986	2.685858	17.428207	26.739151	27.236771

結果如上表所示，表示在 ulp=0.15~0.2 的這個區塊，為滿足 γ=0.95 的最小 number_of_hash 的值為 3。

(1)E 點的環境參數如下：

ulp	clp	N	interval	distribution
0.25	0.8	15	10(秒)	U(1,50)

令 number_of_hash 為 2,3,4,5,6 各執行 1000 次，計算驗證率的平均值和變異數，結果如下表所示。

number_of_hash	2	3	4	5	6
組數	1000	1000	1000	1000	1000
平均數	0.759502	0.931121	0.970691	0.983349	0.989656
變異數	0.031752	0.008157	0.004292	0.003988	0.00319

以右尾檢定，門檻值 γ 為 0.95，拒絕域 α=0.95 時，z-value=1.96。

number_of_hash	2	3	4	5	6
P-value	-33.8066	-6.61026	9.987258	16.69847	22.20235

結果如上表所示，表示在 ulp=0.2~0.25 的這個區塊，為滿足 γ=0.95 的最小 number_of_hash 的值為 4。

(1)F 點的環境參數如下：

ulp	clp	N	interval	distribution
0.3	0.8	15	10(秒)	U(1,50)

令 number_of_hash 為 2,3,4,5,6 各執行 1000 次，計算驗證率的平均值和變異數，結果如下表所示。

number_of_hash	2	3	4	5	6
組數	1000	1000	1000	1000	1000
平均數	0.640016	0.900065	0.958352	0.972527	0.982026
變異數	0.052854	0.014361	0.006679	0.008278	0.006997

以右尾檢定，門檻值 γ 為 0.95，拒絕域 α=0.95 時，z-value=1.96。

number_of_hash	2	3	4	5	6
P-value	-42.6384	237.5074	3.231424	7.829943	12.10738

結果如上表所示，表示在 ulp=0.25~0.3 的這個區塊，為滿足 γ=0.95 的最小 number_of_hash 的值為 4

(1)G 點的環境參數如下：

ulp	clp	N	interval	distribution
0.35	0.8	15	10(秒)	U(1,50)

令 number_of_hash 為 2,3,4,5,6 各執行 1000 次，計算驗證率的平均值和變異數，結果如下表所示。

number_of_hash	2	3	4	5	6
組數	1000	1000	1000	1000	1000
平均數	0.523805	0.859694	0.935768	0.966368	0.977454
變異數	0.056869	0.018689	0.011794	0.007238	0.00723

以右尾檢定，門檻值 γ 為 0.95，拒絕域 α=0.95 時，z-value=1.96。

number_of_hash	2	3	4	5	6

P-value	-56.5159	-20.8892	-4.14407	6.084047	10.21061

結果如上表所示，表示在 ulp=0.3~0.35 的這個區塊，為滿足 γ=0.95 的最小 number_of_hash 的值為 5。

(1)H 點的環境參數如下：

ulp	clp	N	interval	distribution
0.4	0.8	15	10(秒)	U(1,50)

令 number_of_hash 為 2,3,4,5,6 各執行 1000 次，計算驗證率的平均值和變異數，結果如下表所示。

number_of_hash	2	3	4	5	6
組數	1000	1000	1000	1000	1000
平均數	0.360909	0.786802	0.912002	0.955798	0.970545
變異數	0.05297	0.029386	0.011812	0.007452	0.006864

以右尾檢定，門檻值 γ 為 0.95，拒絕域 α=0.95 時，z-value=1.96。

number_of_hash	2	3	4	5	6
P-value	-80.9408	-30.1056	-11.0559	2.123834	7.841755

結果如上表所示，表示在 ulp=0.35~0.4 的這個區塊，為滿足 γ=0.95 的最小 number_of_hash 的值為 5。

總結 A、B、C、D、E、F、G、H 的數據，得到結果下圖所示：

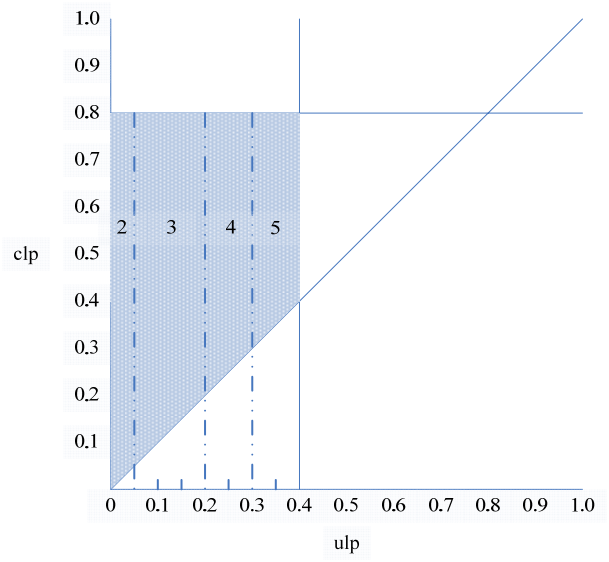


圖 49 給定 N、distribution、interval 及 γ，預估 number_of_hash

這個圖表示在給定 N=15，distribution=U(1,50)，interval=10(s)的情況下，為滿足平均驗證率能有 95%，各區塊所需要的 number_of_hash 值為多少。

我們在這裡示範『給定 N、distributiojn、interval 及 γ，用 Gilbert Model 及 p-value 檢定來估計最小的 number_of_hash 值』的技巧，若是使用其它的參數值，也一樣可以套用這個預估方法。此外必須特別說明的一點是，在這裡 interval 是拿來預估 number_of_hash 用的，而實際上簽署方可以在小於等於 interval 的時間送出簽章(因為 interval 越長，越會降低驗證率)，ex：我們在預估時將 interval 設成 10 秒，但簽署方在實作上可以 7 秒送出簽章而不會讓驗證率小於 γ，但若簽署方在實作上 20 秒才送出簽章，那就有可能會讓驗證率低於 γ。

4.1.5 動態調整 number_of_hash

在這個實驗裡，我們加入 3.4.2 小節的方法，並以 4.1.4 小節估計出的結果為依據，去動態調整number_of_hash。除了一組動態調整為實驗組(EMSS_dynamic)之外，另外再給予 4 組不調整的當做對照組 EMSS_fix(2)、EMSS_fix(3)、EMSS_fix(4)、EMSS_fix(5)【EMSS_fix(x)表示該對照組從頭到尾的number_of_hash 都是 x】，在不同的環境底下模擬數分鐘，並觀察結果。

每一組實驗的EMSS參數必須和4.1.4一樣：分別為：N=15、Interval=10(s)、distribution=U(1,50)我們設計六個Case如下列所示：

表 9 驗證動態調整實驗Case一覽表

Case 編號	環境參數	選擇參數考量	目的
Case1	將網路狀態維持在 ulp=0.02,clp=0.6	假設網路在極好的狀態	比較 EMSS_dynamic 與各對照組在驗證率與頻寬使用上的差別，藉此觀察是否能達到節省頻寬的目的。
Case2	將網路狀態維持在 ulp=0.09,clp=0.7。	在圖49的第一個區塊任選一點	
Case3	將網路狀態維持在 ulp=0.15,clp=0.7。	在圖49的第二個區塊任選一點	
Case4	網路狀態維持在 ulp=0.27,clp=0.75。	在圖49的第三個區塊任選一點	
Case5	將網路狀態維持在 ulp=0.35,clp=0.8。	在圖49的第四個區塊任選一點	
Case6	令程式每隔亂數秒緩慢移動 ulp及clp。	由於網路的狀況可能會改變，而且無法預測，所以我們設定程式在數秒之間自動改變參數。	記錄封包遺失狀態，並且和驗證率比較，藉此觀察 EMSS_dynamic 的缺點。

Case 1：將網路狀態維持在 ulp=0.02, clp=0.6 之下，執行 10 分鐘的平均驗證率如圖所示：

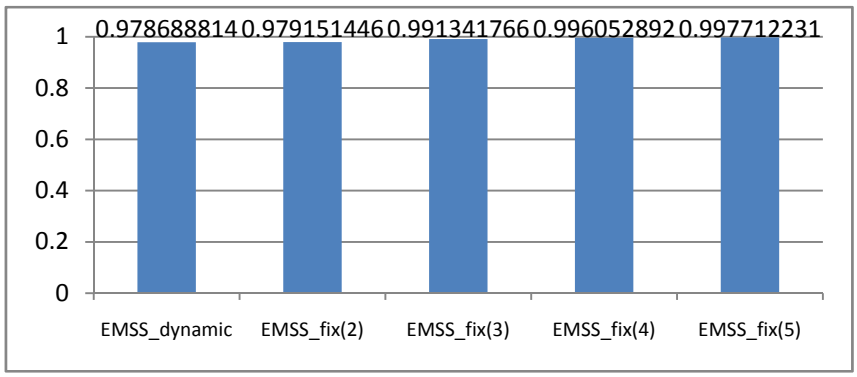
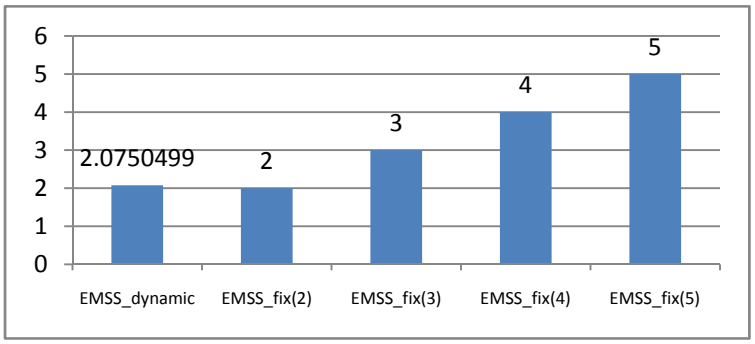


圖 50 Case1 各組實驗的驗證率比較圖

下圖表示各組執行 10 分鐘所計算的 number_of_hash 平均值：



Codec	封包	MD5	百分比
G.711PCM	238	16	6.70%
G.726 ADPCM	198	16	8.08%
G.729A CS-CLEP	108	16	15.00%

圖 51 Case1 各組實驗使用的平均雜湊值比較圖

觀察使用動態調整時，記錄 number_of_hash 的改變狀態如下圖所示：約在 34 秒的時候 number_of_hash 會調整到 2 的位置。

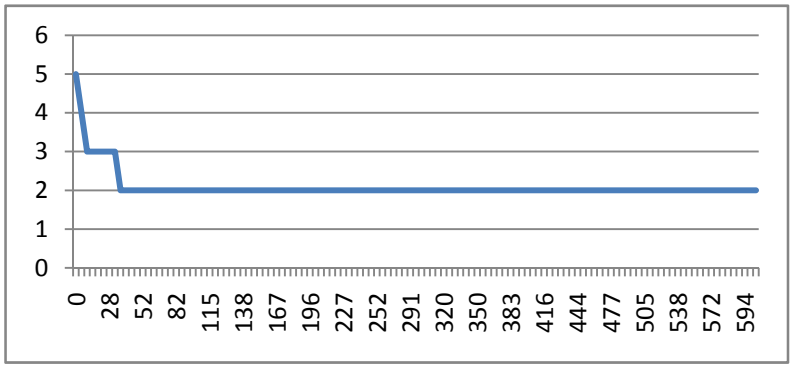


圖 52 Case1 使用動態調整，number_of_hash 的變化圖

Case 2：將網路狀態維持在 ulp=0.09, clp=0.7，執行 10 分鐘的平均驗證率如圖所示：

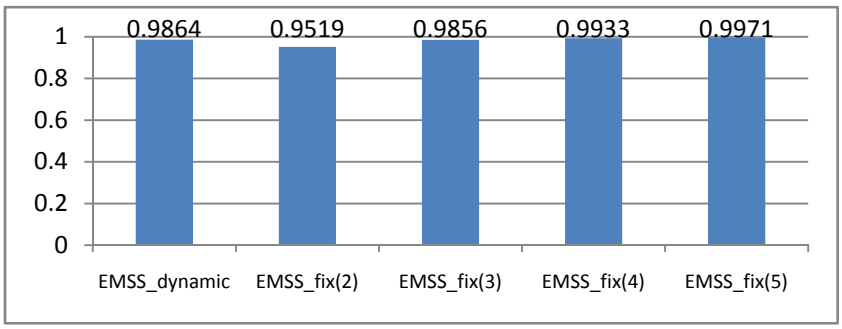
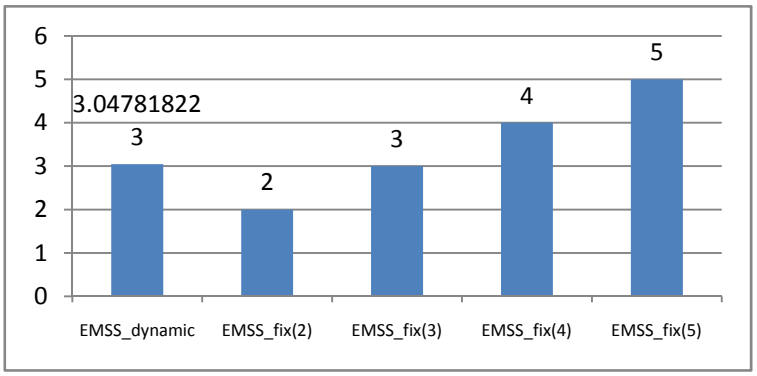


圖 53 Case2 各組實驗的驗證率比較圖

下圖表示各組執行 10 分鐘所計算的 number_of_hash 平均值：



Codec	封包	MD5	百分比
G.711PCM	238	16	6.70%
G.726 ADPCM	198	16	8.08%
G.729A CS-CLEP	108	16	15.00%

圖 54 Case2 各組實驗使用的平均雜湊值比較圖

觀察使用動態調整時，number_of_hash 的改變狀態如下圖所示：約在 16 秒的時候 number_of_hash 會調整到 3 的位置。

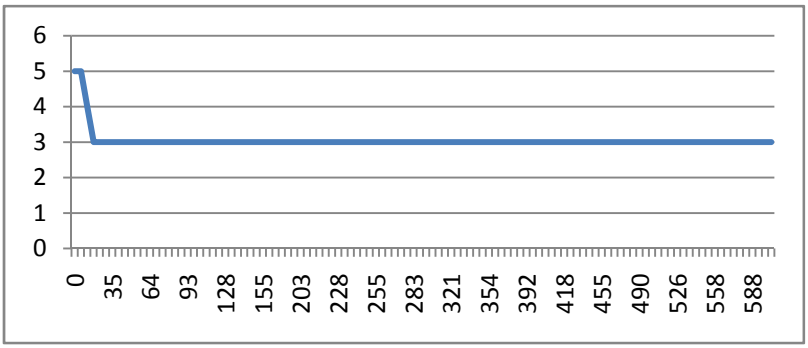


圖 55 Case2 使用動態調整，number_of_hash 的變化圖

Case 3：將網路狀態維持在 ulp=0.15, clp=0.7，執行 10 分鐘得到的平均驗證率如下所示：

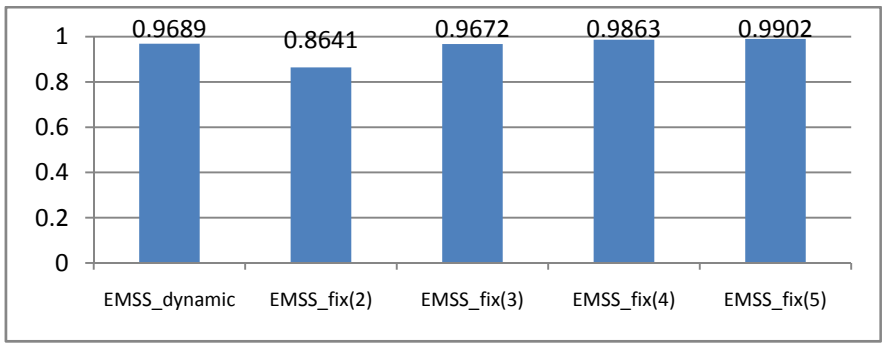
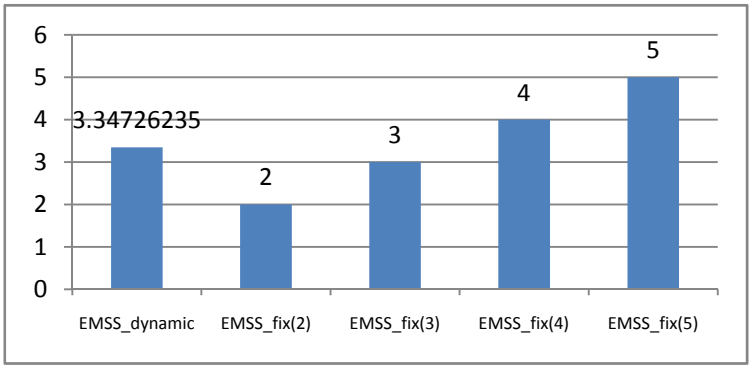


圖 56 Case3 各組實驗的驗證率比較圖

下圖表示各組執行 10 分鐘所計算的 number_of_hash 平均值：



Codec	封包	MD5	百分比
G.711PCM	238	16	6.70%
G.726 ADPCM	198	16	8.08%
G.729A CS-CLEP	108	16	15.00%

圖 57 Case 3 各組實驗使用的平均雜湊值比較圖

觀察使用動態調整時，number_of_hash 的改變狀態如下圖所示：number_of_hash 在 3~4 之間徘徊。

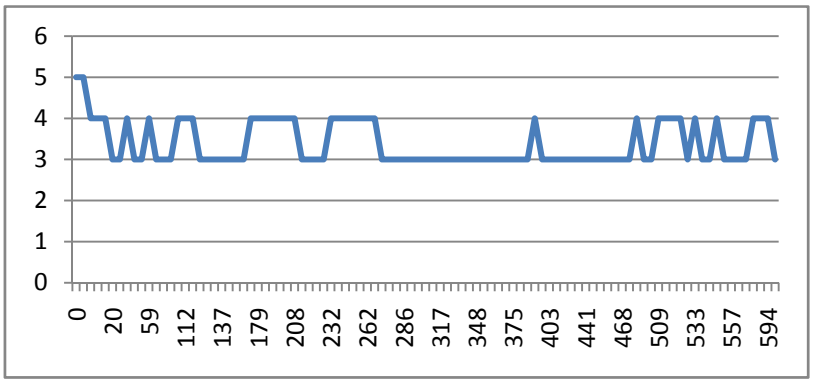


圖 58 Case3 使用動態調整，number_of_hash 的變化圖

Case 4：將網路狀態維持在 ulp=0.27,clp=0.75，執行 10 分鐘的平均驗證率如圖所示：

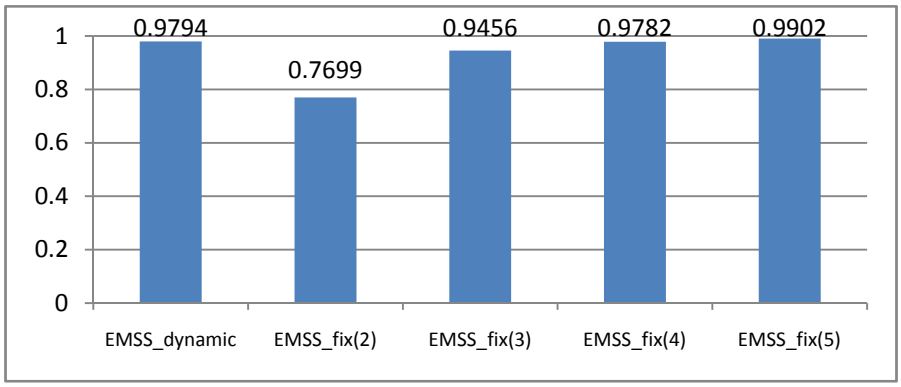
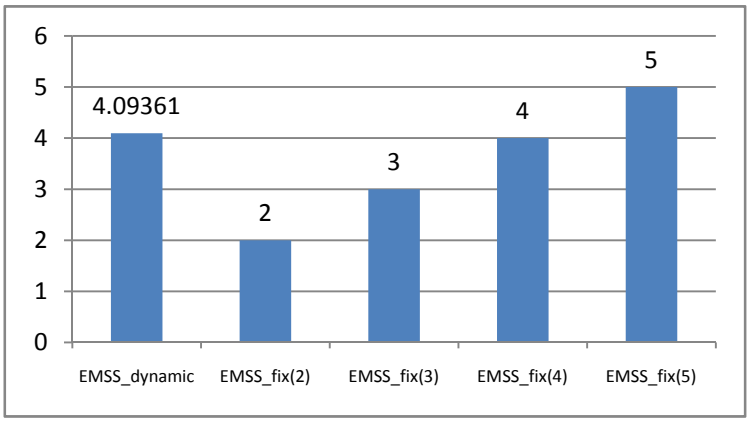


圖 59 Case4 各組實驗的驗證率比較圖

下圖表示各組執行 10 分鐘所計算的 number_of_hash 平均值：



Codec	封包	MD5	百分比
G.711PCM	238	16	6.70%
G.726 ADPCM	198	16	8.08%
G.729A CS-CLEP	108	16	15.00%

圖 60 Case 4 各組實驗使用的平均雜湊值比較圖

觀察使用動態調整時，number_of_hash 的改變狀態如下圖所示：

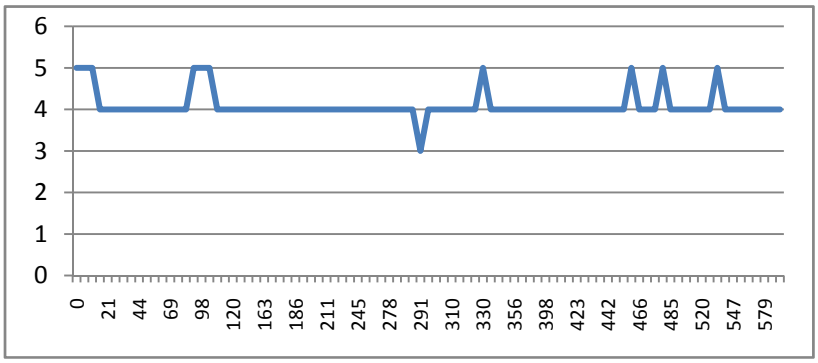


圖 61 Case4 使用動態調整，number_of_hash 的變化圖

Case 5：將網路狀態維持在 ulp=0.35, clp=0.8，執行 10 分鐘後的平均驗證率如下所示：

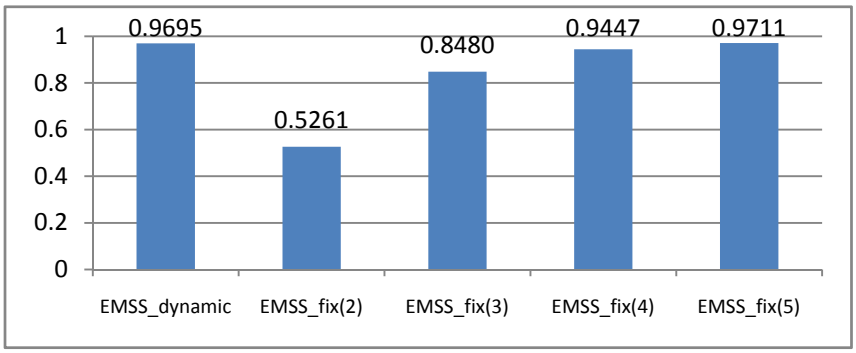
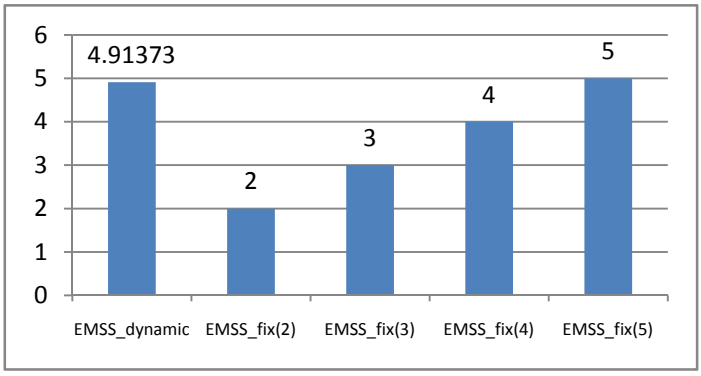


圖 62 Case5 各組實驗的驗證率比較圖

下圖表示各組執行 10 分鐘所計算的 number_of_hash 平均值：



Codec	封包	MD5	百分比
G.711PCM	238	16	6.70%
G.726 ADPCM	198	16	8.08%
G.729A CS-CLEP	108	16	15.00%

圖 63 各組實驗使用的平均雜湊值比較圖

觀察使用動態調整時，number_of_hash 的改變狀態如下圖所示：

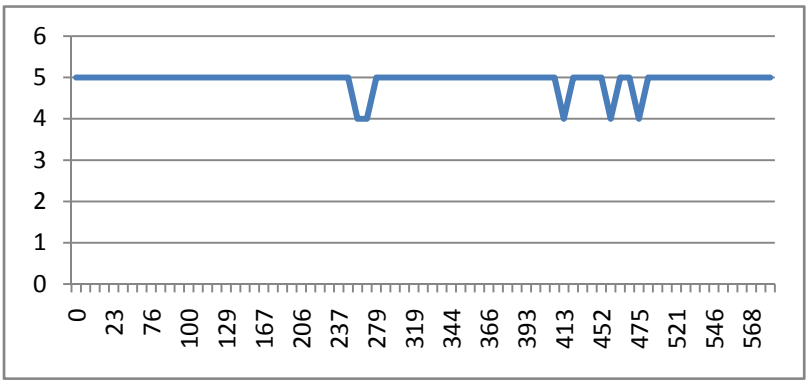


圖 64 使用動態調整，number_of_hash 的變化圖

Case 6：令程式每隔亂數秒緩慢移動 ulp 及 clp，我們想用這個 case 看出使用動態調整可能會出現的問題。執行 10 分鐘後我們記錄封包遺失率如下圖所示：

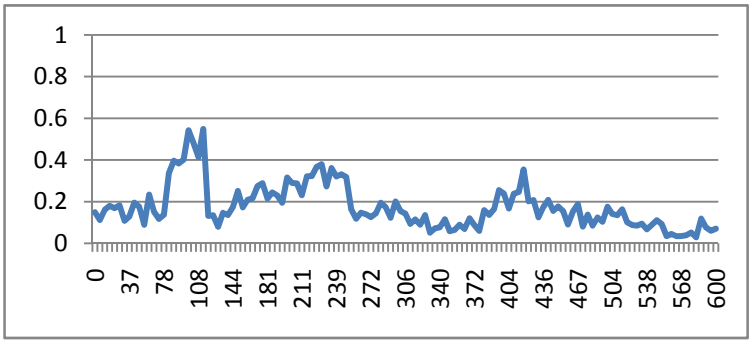


圖 65 Case6　封包遺失率記錄圖

以折線圖觀察每一組在每一時點的驗證率如下圖所示：

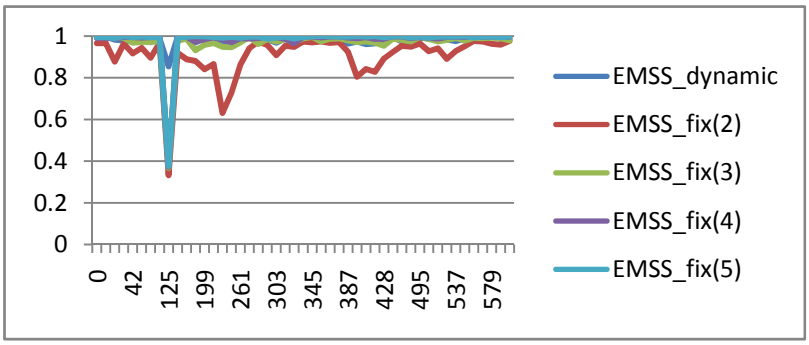


圖 66 Case 6 各組實驗的驗證率細部比較圖

我們可以在上圖察覺到，EMSS_fix(2)及 EMSS_fix(3)在約 100 秒左右時，因為無法反應封包的大量遺失，驗證率會驟降至 40%左右，而使用動態調整的話，仍然維持在 85%以上。

4.1.6 小結

我們透過 4.1.5 的實驗，可以看到根據 RTCP 去動態調整 number_of_hash 的值，可以在驗證率及 communication overhead 之間達到某種程度的平衡，平衡的意思並不代表它就是最好，回顧實驗 4.1.5 的第一個 Case，使用動態調整的實驗組 EMSS_dynamic 在 34 秒的時候才將值調整到 2，若拿來和對照組 EMSS_fix(2)相比較的話，可以說它前面 34 秒多使用的雜湊值反而造成了浪費，

但是和 EMSS_fix(5)相比，EMSS_dynamic 卻節省了很多頻寬。而在 Case 6 中我們也可以看到，對於網路封包遺失率的突然暴升，使用動態調整的方法雖然會在短時間之內無法維持驗證率在 95%而掉到 85%左右，但仍然比 EMSS_fix(2)及 EMSS_fix(3)好得多。由於一般正常的網路在短時間之內都會維持在某個穩定狀態，所以我們認為使用動態調整 number_of_hash 確實存在其效益。

必須再次強調的一點是，因為 VoIP 的封包大小約在 100 bytes~200 byte 之間，所以即使只節省一個約 16 bytes 雜湊值，對於整體頻寬也是不算小的節省。

4.2 安全性分析

我們在這個小節進行安全性分析，主要針對在 VoIP 中加入 EMSS 之後，確實能夠達到不可否認性的目的，以及並不會因為加入 EMSS 而破壞 VoIP 本身具有抵抗 Reply Attack 及 Eavesdropping Attack 的能力。

4.2.1 不可否認性

不可否認性分為兩個部份，包括連線的不可否認及對話的不可否認，我們以圖 64 說明：

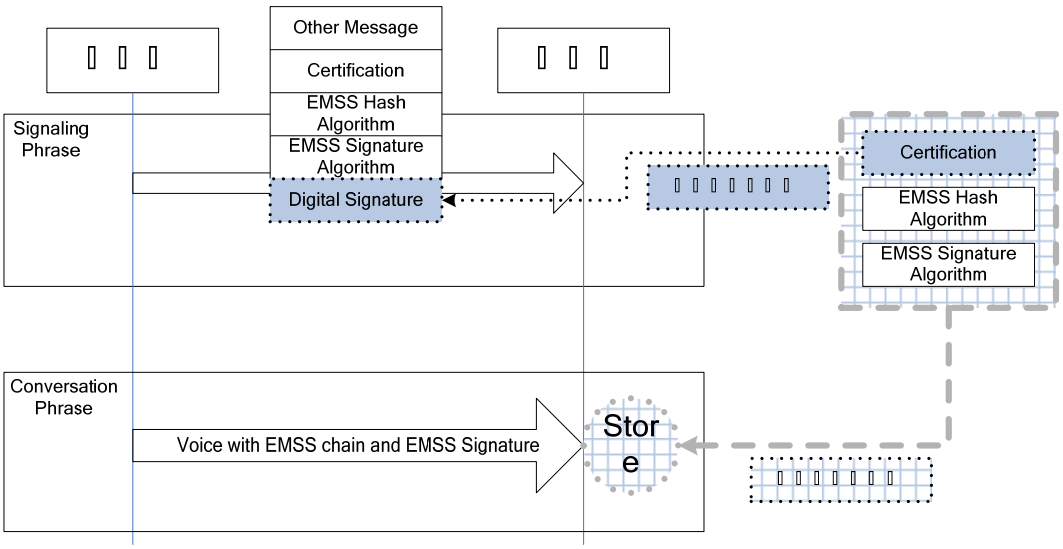


圖 67 在 VoIP 中加入 EMSS 的不可否認性

為了方便表示，我們將角色的名字改為簽署方及驗證方，參與談話的人可以同時是簽署方及驗證方。除此之外驗證方的角色也可以由第三者(如司法單位)居中扮演，總而言之，驗證方就是記錄下這些資訊並加以驗證的人。

(1)簽署方在 Session Establishment Protocol 的訊息中加入簽章(Digital Signature)，驗證方可以使用簽署方的憑證(Certification)去驗證該訊息，由於只有簽署方用自己的私鑰才能簽署，所以這構成了連線的不可否認性。

(2)簽署方在語音串流的RTP封包中加入EMSS簽章，驗證方使用Certification、EMSS Hash Algorithm、EMSS Signature Algorithm 去驗證語音封包，由於只有簽署方用自己的私鑰才能簽署，所以這構成了對話的不可否認性。

4.2.2 Replay Attack

我們在這個小節討論，在 VoIP 中加入 EMSS 並不會破壞它原本抵抗 Replay Attack 的能力。所以我們必須先說明 VoIP 是如何防制這項攻擊的。參考 MIKEY rfc 3830 的第 5.4 小節裡描述到：在交換 MIKEY 的命令中，必須加入 timestamp 這個欄位，而且所有參與 MIKEY 的主機(Host)，都必須實作下列數項：

(1)所有主機都必須在時間上有某種程度的同步化(loosely synchrnoized)

(2)這個時間同步化的動作可以透過網路及同步化協定來完成。

(3)所有的主機都有 replay cache，並且利用它記錄經認證且已經處理過的訊息。

(4)當收到新的訊息時，比對 replay cache 中的記錄及新訊息的 timestamp，若是已在replay cache 中或timestamp過期，則會認為該訊息可能是重送攻擊，應該丟棄不予處理。

簡單來說，就是利用訊息的 timestamp，及在所有的節點額外多做時間同步化及快取，來判斷一個訊息是否為重送。而我們在 VoIP 連線階段多加的額外參數

(Certification、EMSS Hash Algorithm、EMSS Signature Algorithm)並不會影響 timestamp 這個欄位，因此也不會影響到這個安全機制。

4.2.3 Eavesdropping Attack

VoIP 可以使用 SRTP 加密語音封包,去抵擋竊聽攻擊。我們可以由下圖看出，EMSS 所使用的雜湊值及簽章，會和語音 codec 一同被視為 RTP Payload 加密，所以並不會破壞原本 SRTP 的加密機制。

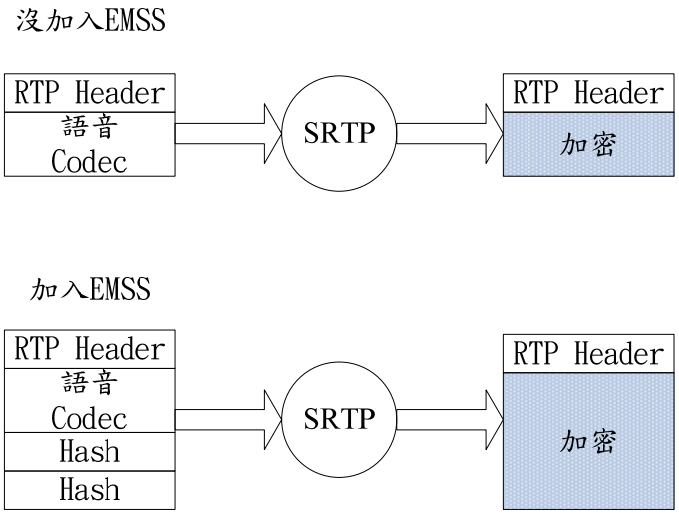


圖 68 VoIP 使用 SRTP 抵抗竊聽攻擊

第五章、結論及未來發展

5.1 結論

由於現行的網路電話中，並沒有加入數位簽章來達到不可否認性的相關規範，因此本研究的貢獻在於，我們針對這一個問題，將 EMSS 加入現有的 VoIP 架構中來達到(1)不可否認性(2)不影響通話品質的低傳送端延遲(3)能在低階手持裝置上運行的低工作負載，並且更進一步提出一個根據 RTCP 動態調整 EMSS 參數的機制，再經過模擬實驗驗證之後發現，當網路環境處於穩定的狀態時，可以藉由犧牲少許驗證率(小於 4%)為代價的情況之下，達到節省頻寬(5%~16%以上)的目的，而即使是在網路環境不穩定的情況下，也能在驗證率與頻寬的使用上達到平衡。最後我們經由安全性分析，說明將 EMSS 加入現有的 VoIP 架構中，除了達到不可否認的目的外，並不會破壞 VoIP 原本的安全機制。

5.2 未來展望

1.使用其它種簽章方案

我們在本研究中使用 EMSS，但其實不論哪一種簽章法，對語音封包很小的 VoIP 及資源有限的手持裝置來說，都會構成 communication overhead 或 computational overhead。因此我們認為，即便基於其它的考量，使用別種串流簽章演算法，也都能夠藉由(1)在連線階段加入驗證資訊(2)在語音封包中加入簽章(3)根據 RTCP 動態調整簽章參數這三個步驟，來達到節省資源的目的。

2. 多人會話中，根據 RTCP 動態調整的適用性

在我們的模擬實驗裡，僅考慮到兩人通話，但在多人會話時，RTCP 會根據參與會談人數的多寡調整發送速度，參與的人數越多，為避免網路擁塞，回報速

度會越慢，如此對依靠 RTCP 動態調整串流簽章演算法參數會產生影響，至於要多少人才會產生嚴重影響，我們並沒有研究相關議題。

3.加入 PKI 及合法監聽的架構

我們之前提過,在 VoIP 中加入簽章,可以讓第三者如司法單位或律師錄下，以做為數位證據的來源。因此可以結合現有的監聽機制及 PKI 架構,但是否必須額外修改，並沒有在我們的研究中考慮進來。

參考文獻

[1]A. Perrig, R. Canetti, J. D. Tygar, and D. Song, “Efficient authentication and signing of multicast streams over lossy channels,” IEEE Symposium on Security and Privacy, May2000, pp. 56–73.

[2]Christian Hett,Nicolai Kuntze,and Andreas U.Schmidt. “security and non-repudiation for voice-over-op conversations”, January 2006

[3]C. Wong and S. Lam, Digital Signatures for Flows and Multicasts, Technical Report TR-98-15, Department of Computer Sciences, University of Texas at Austin, May 1998, 25 pp.

[4]Ge Zhang; Hillenbrand, M.; Muller, P. “Facilitating the interoperability among different VoIP protocols with VoIP Web services,” Distributed Frameworks for Multimedia Applications, 2005. DFMA apos;05. First International Conference on Volume , Issue , 6-9 Feb. 2005 Page(s): 39 – 44

[5]H.Schulzrinne,S Casner,R.Frederick,V.Jacobson. “RTP:A Transport Protocol for Real-Time Applications”,RFC 3550,July 2003.

[6]I Busse,B Deffnere,H Schulzrinne. Dynamic Qos Control of Multimedia Applications Based on RTP.Computer Communications,1996,19;p49-58.

[7]JC Bolot,” Characterizing End-to-End Packet Delay and Loss in the Internet” Journal of High Speed Networks, 1993

[8]Jung Min Park,Edwin K.P.Chong,Howard Jay Siegel,” Efficient Multicast Packet Authentication Using Signature Amortization” proceedings of the 2002 IEEE Symposium on Security and Privacy

[9]J.Arkko,E.Carrara,F.Lindolm,M.Naslund,K.Norrman, “MIKEY:Multimedia Internet KEYing”,RFC 3830,August 2004.

[10]LAN Fan,ZHANG Yaobi,”Network Packet Loss Simulation Based on Gilbert Model” Dept. of Computer Science and Engineering,Shanghai Jiaotong University,Shanghai 200030

[11]M.Baugher,D.McGrew,M.Naslund,E.Carrara,K.Norrman, “The Secure Real-Time Transport Protocol (SRTP)”,RFC 3711,March 2004

[12]M.Handley, H.Schulzrinne, E.Schooler, and J.Rosenberg, ”Session Initiation Protocol, ” RFC 3261,June 2002.

[13]M. Handley,V. Jacobson, “SDP:Session Description Protocol”, RFC 2327 Aprial 1998

[14]M. Rabin, “Efficient dispersal of information for security, load balancing, and fault tolerance,” Journal of the ACM, Vol. 36, No. 2, Apr. 1989, pp. 335–348.

[15]M. Yajnik, S. Moon, J. Kurose, and D. Towsley. Measurement and modelling of the temporal dependency packet loss. In IEEE INFOCOM ’99, New York,NY, March 1999.

[16]Peter Thermos and Ari Takanen,”Securing VoIP Networks Chapter 7:Key Management Mechanisms” 09 Oct 2007.

[17]Prateek Gupta and Vitaly Shmatikov“Security Analysis of Voice-over-IP Protocols”,csf,pp.49-63,20th IEEEE Computer Security Foundations Symposium(CSF’07),2007

[18]P.Golle and N.Modaugu, ”Authenticating streamed data in the presence of random packet loss” Network and Distributed System Security Symposium(NDSS ’01),Feb.2001,pp.13-22

[19]Sanneck H,Carle G.A Framework Model for Packet Loss Metrics Based on Loss Runlengths[A] Proceedings of the SPIE/ACM SIGMM Multimedia Computing and Networking Conference 2000(MMCN 2000)[C],CA(USA):San Jose,2000:177-187

[20]S. Miner and J. Staddon, “Graph-based authentication of digital streams,” IEEE Symposium on Security and Privacy,May 2001, pp. 232–246.

[21]YU Su'◦SHENG Yan-Jinz，HUANG Kai ，WANG Wen-Jing,“ Research on the Algorithm of RTP Adaptive Transmission Control” COMPUTER ENGINEERING & SCIENCE v01．27，No．11，200

[22]SIP會談起始協議操典，賈文康編著，松崗出版社2006

[23]統計學, 白賜欽, 前程企業管理有限公司, 1997

[24]楊文超，“不信任區域網路中數位證據保留之研究”，國立中央大學資訊管理研究所碩士論文，2002年5月31日