

# 安全網路電話中 SRTP 的研究與金鑰交換的設計

## Study on SRTP and Design of Key Exchange for Secure VoIP

馮輝文\* 陳永慶 陳穎聰

國立台灣科技大學資訊工程系

Huei-Wen Ferng, Yung-Ching Chen and Ying-Tsung Chen

Department of Computer Science and Information Engineering

National Taiwan University of Science and Technology, Taipei 106, Taiwan

\*E-mail: hwferng@mail.ntust.edu.tw

### 摘要

本論文著重於具保密特性之網路電話的金鑰交換設計與相關協定增修與實作。首先，我們改進 Diffie-Hellman (D-H)演算法來讓金鑰交換可使監聽第三者獲取通話用之金鑰，以達監聽與簡易實作之目標。基於 D-H 機制我們設計了兩段式的金鑰交換機制，並增修 SIP (Session Initiation Protocol)相關訊息來達成此目的。另外，我們亦實作 SRTP (Secure RTP)於 SIP UA (User Agent)上。透過語音數據串流實驗與效能測試評估結果，我們將說明 SRTP對傳送與接收兩端效能影響相當微幅，亦即 SRTP 與此金鑰交換相當適合用於安全網路電話中。

**關鍵詞：** Voice over IP, Session Initiation Protocol, Real-Time Transmission Protocol, Secure RTP, Key Exchange

### 1、簡介

VoIP [6], [19], [25]是一種快速興起的技術，其利用 IP (Internet Protocol) [7]技術來實現新型的電話通訊。VoIP 透過語音信號數位化處理、壓縮編碼等讓其於網路上傳輸；然後再解壓縮，把數位信號還原成原音，來達到通話的目的。VoIP 將聲音從傳送端到目的端的基本過程是：1.聲電轉換—透過裝置將聲波變換為電信號；2.量化取樣—將模擬電信號按照某種取樣方式(例如：脈衝編碼調製，即 PCM)轉換成數位信號；3.製成封包—將一定時長數位化後的語音信號組合，隨後，按照國際電聯 (ITU-T)的標準，這些語音數據被封裝到 RTP [2], [4]封包中，接著並進一步封裝到 UDP (User Datagram Protocol) [7]封包和 IP 封包中；4.傳輸—IP 封包在 IP 網路由傳送端傳送

到目的端；5.接收封包並做電聲轉換。VoIP 技術有別於傳統電話網路，例如，公眾交換電話網路 (Public-Switched Telephone Network, PSTN)使用線路交換(Circuit Switching)的技術，它是使用分封交換(Packet Switching)的技術來傳輸語音，而規範語音封包的協定則是 RTP。雖然 VoIP 技術可減少語音及資料通訊的成本，但是網路上充斥著各種駭客攻擊，因此有許多安全研究的議題及改進的空間。

首先讓我們先說明 RTP。它是由 IETF (Internet Research Task Force)的多媒體傳輸工作小組所制定的一個網路傳輸協定，RTP 詳細說明了在網際網路上傳遞語音和視訊的標準數據封包格式。一開始 RTP 被設計為一個多播 (Multicast)協定，但後來被用在很多單播 (Unicast)應用中；另外 RTP 常用於串流媒體(配合 RTSP (Real Time Streaming Protocol) [9], [11])和視頻會議等系統(配合 H.323 或 SIP (Session Initiation Protocol) [14]-[27])，使它成為 VoIP 產業的技術基礎。通常 RTP 和 RTCP [2] (RTP Control Protocol)一起使用，而且它是建立在 UDP 協定之上。RTP 可以提供的服務包含 Payload-Type Identification、Sequence Numbering、Time Stamping、Delivery Monitoring 等。透過上述的服務，RTP 也確保了語音封包能在 IP 網路中按照順序地抵達接收端，達到語音的特性。然而 RTP 並沒有考慮到安全性，因此未加密封包易受竊聽者進行竊聽。另一方面 RTP 也沒有考慮保護訊息的完整性，攻擊者仍然可以偽造數據，若不能偽造數據，但是至少可以重覆過去傳輸過的數據，達到攻擊的目的。

目前，在大多數企業 LAN 的 VoIP 是不加密的，也就是說有人可以輕易地竊聽企業網路中任一部電話的通話內容。而現今網路電話並非僅僅在企業的內部網路中使用，而是推廣至網際網路(Internet)上，如此一來使用者與使用者間的通話被竊聽的風險便大大地提高了，所

以對於一些較重要的談話內容來說要如何達到私密性的保護，變得較為大眾所在意，而對其通話內容進行加密的動作彷彿是較為直接且簡單的想法。

鑑於上述原因，SRTP [5]是在 RTP 基礎上所定義的一個新的協定，旨在於為單播和多播應用程序中即時傳輸協定的數據提供加密、訊息認證、完整性保證和重傳保護。它是由 David Oran (Cisco)和 Rolf Blom (Ericsson)開發，最早由 IETF 於 2004 年 3 月作為 RFC 3711 發佈。SRTP 為 RTP 提供了安全配置檔(Security Profile)，使資料封包具備機密性和訊息認證，解決了網際網路上的電話技術的安全性與完整性問題。

SRTP 雖然可以提供上述安全保護，讓通話內容不被揭露。然而網路電話若需能夠商業化運行是必須能提供監聽之機制。所以若是單純採用使用者與使用者間金鑰的交換(如 Diffie-Hellman [5])，則它將無法滿足電信法規的要求。所以在 Diffie-Hellman [5]演算法之下，只有使用者雙方才會知道交換後的金鑰為何，這在資訊安全方面來講是很好的，可是對於政府單位來說卻是頭痛的。因為沒有金鑰便無法進行監聽的動作，所以此種做法也是不被允許。為了使用 Diffie-Hellman 這個簡易的技術，但又要讓合法的管理者知道兩兩使用者間所使用的金鑰。本論文將設計一個新的金鑰交換機制，配合 SRTP 與增修 SIP 相關訊息，此金鑰交換機制可以讓網路電話具安全性也具監聽之功能。

最後我們實作 SRTP 安全機制與此金鑰交換機制於 SIP UA [1]-[3]上，以及測試和觀察 RTP 與 SRTP 語音封包在網路傳輸的表現，來了解 SRTP 是否會影響傳送、接收兩端效能。

本論文其他章節之安排如下：第二節繼續說明相關之研究；第三節則說明我們的相觀研究；第四節說明測試結果與討論；最後一節則是結論。

## 2、相關研究

SRTP 與 RTP 在文獻上的討論，則主要有 IETF 之 RFCs [4], [5]；SRTP 最主要的功能就是要對語音數據封包做加解密，確保在網路傳輸的安全性，然而做加解密時需在網路傳送接收兩端使用相同的金鑰，故金鑰的分配也需要相關的協定。在 SRTP 的 RFC 中有附帶提及所謂 Key Management 的相關協定，例如：MIKEY [8]、KEYMGMT [9]、SDMS [12]、KINK [13]以及 GDOI [14]等。以上 Key Management

機制雖然完整，但是卻相對複雜許多。

為了以簡單的概念來達成快速實作但又有成效，且儘可能將程式開發的時程降到最低，以完成在 SIP 網路架構之下的金鑰交換，以及對兩兩使用者間的金鑰上的使用作記錄與管理，讓雙方所使用的 SIP UA 也藉由此金鑰進行加密來達到防止惡意第三者進行竊聽的動作，我們在本論文將以 Diffie-Hellman 為基礎，研發新的機制。

而關於實作方面，目前有許多廠商也著手於 RTP 或者 SRTP 的研發上。在實作上，我們將參考 Vovida [16]的 Open Communication Application Library，亦即所謂的 VOCAL (Vovida Open Communication Application Library) [16], [19]所提供的 WinRTP [1]-[3]和 LibSRTP [16]。我們將利用 LibSRTP 來整合原先無 SRTP 的 WinRTP 以建置具安全功能之 SIP UA。由於 LibSRTP 的原始碼是 Unix/Linux 版本，程式開發使用 C/C++。所以我們須先對程式架構與功能進行了解並加以 Porting 到 Windows 系統上與 WinRTP 做整合。最後，採用我們所設計之金鑰交換，監聽功能就可達成。

## 3、研究方法

SRTP 可以視為 RTP 之延伸，其對 RTP 封包提供機密性、訊息認證與重傳保護以達到語音數據串流(Data Flow)的安全性和完整性。為了提供對語音數據串流的保密，需要對數據流進行加密和解密。關於這點，SRTP 採用一標準加密演算法，即 AES (Advanced Encryption Standard)或者 Rijndael，它是區塊加密演算法(Block Cipher)標準；此類加密演算法有兩種加密模式：區段整數計數器模式(Segmented Integer Counter Mode)和 f8 模式(f8 Mode)。在訊息認證並保護訊息的完整性方面，SRTP 使用了 HMAC-SHA1 (keyed-Hash Message Authentication-Code Secure Hash Algorithm)作為訊息認證碼，此碼是利用密碼學的赫序函數(Hash Function)結合密鑰(Secret Key)所產生。

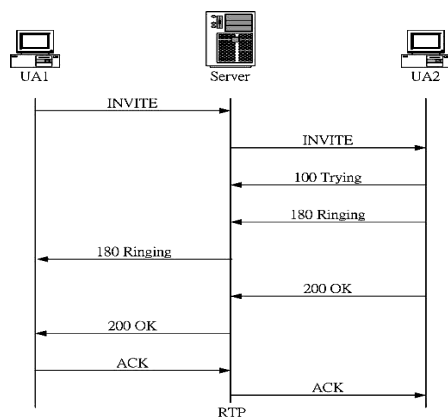
為了研究 SRTP，我們採用實作及實測的方式。我們採用 VOCAL 的開放原始碼，稱為 libSRTP，其由兩個最主要的副程式，srtp protect 和 srtp unprotect 以及其他相關副程式所組成。由於我們選定的實作平台為 Microsoft Windows，然而 libSRTP 原始碼是 Unix/Linux 版本，於是我們須先對協定原理與架構了解，之後再把需要的相關副程式 Porting 至

Windows 平台與原平台上的 WinRTP 整合，達到 SRTP 安全機制的目標。最後透過抓取語音數據串流實驗，我們可以檢驗 SRTP 安全機制及相關效能。而關於金鑰交換及管理部分，原有 VOCAL 所開放的 SIP Server 原始碼是功能上是屬於較為陽春的系統，其並沒有管理及記錄金鑰的功能，這將使得監聽單位無法獲得其加密雙方所使用的金鑰，這一點是許多國家所在網路電話使用上所不允許的。於是我們研發記錄金鑰的功能並可加到 SIP Server 之中，以符合電信相關規定。而在 SIP UA 方面，VOCAL 所提供的 SIP UA 功能較少，所以其也沒有提供金鑰交換的功能，為了安全的加密需求，我們研發了新且簡單的金鑰交換，使雙方的 SIP UA 在通話建立之前便完成金鑰的交換，以提供 SRTP 加密時所需要的金鑰使用，也提供監聽第三者相關金鑰的記錄。

### 3.1 SIP 基本訊息流程

當使用者想使用 SIP UA 時，首先必須先進行註冊的動作。UA 會傳送 Register 的訊息到 Server，當 Server 收到此訊息，Server 會先傳送 100 trying 訊息，若是此 UA 經檢驗後是該 Server 的合法使用者，則回傳 200 OK 的訊息，如此便完成註冊的動作。

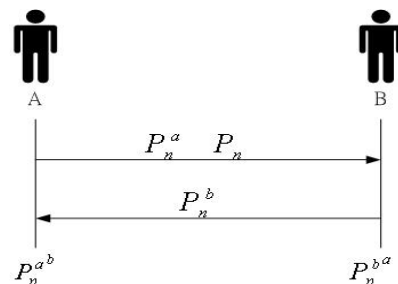
而當 UA1 使用者想要與 UA2 使用者進行通話時，UA1 會先傳送 INVITE 訊息到 Server，此時 Server 會先回傳 100 Trying 給 UA1，之後 Server 經解譯後會轉送 INVITE 訊息至 UA2，UA2 會透過 Server 回傳 100 Trying 給 UA1，之後再傳送 180 Ringing 與 200 OK 等。UA1 在回傳 ACK 後便可建立 RTP 通道，之後的通話將不用透過 Server，直到要結束通話。以上 SIP 訊息傳遞流程如圖一所示。



圖一：通話流程 SIP 訊息。

### 3.2 金鑰交換與記錄設計概念

在此我們所使用的金鑰交換技術是基於 Diffie-Hellman Key Exchange，D-H 是一個被普遍應用在金鑰交換的公開金鑰演算法，此演算法是仰賴離散對數的困難度，這和大整數分解因數一樣，都非常困難，其技術為一種公用基碼加密演算法，它讓通訊雙方經過公用資訊交換後來決定雙方所共用的金鑰。如圖二使用者 A 會先傳送自己的公鑰(Public Key)  $P_n^a$  與所使用的質數(Prime Number)  $P_n$ ，當使用者 B 收到後便以 A 的公鑰與自己的私鑰(Private Key)來算出所要共同使用的金鑰為何，並回傳自己的公鑰給 A，A 再依所收到的公鑰加上自己的私鑰來算出共同使用的金鑰，如此便完成了金鑰的交換。



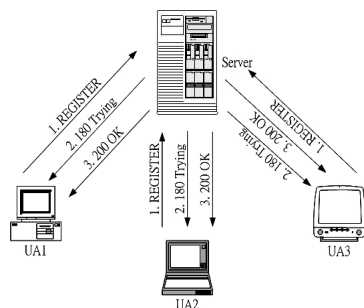
圖二：Basic Diffie-Hellman Key Exchange。

由於 D-H 演算法的困難度在於要推算出對方所使用的私鑰是十分困難的，加上所使用的私鑰只有使用者本身知道，換句話說私鑰應該是永遠不會讓第二者知道，所以更不可能做傳送私鑰的動作給記錄者，可是為了讓管理者獲得每位進行加密通話使用者所使用的私鑰，於是我們設計了兩段式的 D-H 金鑰交換，在第一階段中我們利用 D-H 在使用者與 Server 間先產生出私鑰，不再是由使用者自己亂數產生私鑰。在第二階段中的 D-H 便使用第一階段所產生的私鑰加上 D-H 演算法來算出雙方使用者所要的金鑰。以下兩小節讓我們詳細來描述此兩段式 D-H Key Exchange。

### 3.3 第一階段 D-H 金鑰交換

第一階段的 D-H Key Exchange 是 UA 與 Server 之間的動作，其主要目的在於產生使用者在進行通話時所真正使用的私鑰，我們在此稱之為**通話私鑰**，而第一階段中的 D-H Key

Exchange 所使用的私鑰我們在此稱之為**註冊私鑰**，所使用的公鑰我們在此稱之為**註冊公鑰**，而第二階段中的 D-H Key Exchange 所使用的公鑰我們則稱之為**通話公鑰**。所以當某位使用者要開始使用 SIP UA 時，使用者會自動的來進行註冊動作，在註冊的過程中通話私鑰的產生也將同步進行，如圖三所示，當使用者 UA1 要進行註冊時會先送出 REGISTER 的 SIP 訊息，在這 REGISTER 訊息中包含了 SDP 訊息，我們則在 SDP 訊息中加入了 UA1 的註冊公鑰與所使用的質數，藉由 REGISTER 將這些資訊傳送至 Server 端，當 Server 端收到後便會進行檢查，檢查結果若屬於可加密的 UA，則將記錄發送端 UA 所傳送過來的 USER ID、註冊公鑰與所使用質數到資料庫中，下一步 Server 則使用該 UA 所傳送過來的質數來做成 Server 與該 UA 間的 Server 註冊公鑰，然後將註冊公鑰加到 200 OK 之 SIP 訊息中(SDP 裡)，然後回傳至該 UA。當使用端 UA 收到 200 OK 訊息後會進行解析並將該 Server 註冊公鑰取出。接下來配合上自己的註冊私鑰來產生通話私鑰。而 UA2 或是其他使用者 UA 亦執行上述程序。不過在此過程中 Server 並不會去計算該 UA 的通話私鑰為何，只有當管理者想要知道該 UA 目前所使用的通話私鑰為何時，才會透過資料庫便計算出該 USER ID 的通話私鑰，以降低 Server 的負擔。



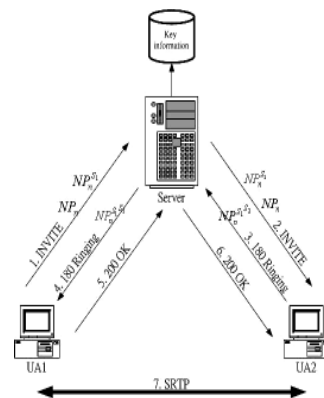
圖三：第一階段 D-H 金鑰交換(使用 Basic D-H Key Exchange)。

### 3.4 第二階段 D-H 金鑰交換

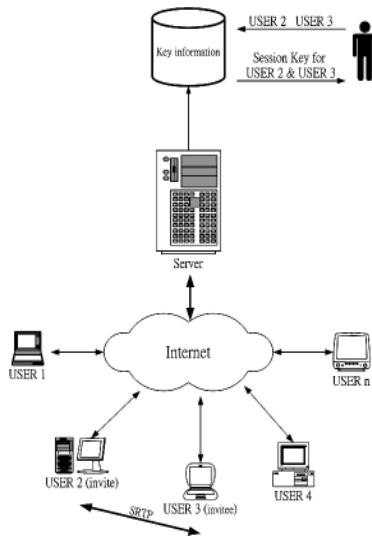
在第二階段中的 D-H Key Exchange 為 UA 與 UA 間的動作為主，使用的時機為使用者打算要進行加密的通話時，其主要的目的是要產生通話時所使用的金鑰，然後透過此金鑰與加密演算法來達成加密的安全通話。可是使用加密通話時管理者要如何知道正在通話的兩端使用者所使用的金鑰呢？如圖四所示(相關參數亦圖示於此)，當 UA1 的使用者想要與 UA2 的使用者進行通話時，此時 UA1 一開始

會先發送 INVITE 訊息到 Server 端，而在傳送這 INVITE 前會先使用在第一階段所產生的通話私鑰  $S_1 = P_n^{a \cdot b}$  然後配合上新產生

的質數來製成通話公鑰  $NP_n^{S_1}$ ，並將此通話公鑰與所使用的質數加到 INVITE 訊息中(SDP 裡)來傳送出去，當 Server 收到此 INVITE 訊息時，便會進行解析並且判斷是否為加密的通話要求，若不是則直接轉送到所要進行通話邀請端的使用者 UA，若有加密需求則將該 INVITE 訊息中的 USER ID、通話公鑰與所使用的質數記錄到資料庫中，記錄完成後則轉送該 INVITE 訊息到所要進行邀請的使用者 UA，被邀請端收到該 INVITE 訊息後，則會解析出 INVITE 訊息中的通話公鑰與所要使用的質數，然後配合本身在第一階段中所完成的通話私鑰，進一步的做出在通話時所要使用的金鑰及本身的通話公鑰，並且將該通話公鑰加到 180 Ringing 訊息中傳送到 Server，當 Server 收到此 180 Ringing 的訊息後一樣會進行解析與判別，若不是要進行加密的通話 SIP 訊息時便會直接轉送至進行 INVITE 的使用者 UA，如果判別結果為要加密的訊息後，便會將 USER ID 與其通話公鑰記錄到資料庫中，然後將訊息轉送到原 INVITE 端，INVITE 端的使用者 UA 收到後便從 180 Ringing 的訊息中取出被 INVITE 端的通話公鑰，利用此通話公鑰，配合上自己的通話私鑰便可以產生與被 INVITE 端所產生的相同金鑰，如此一來雙方便可利用所做出的金鑰順利的來進行加解密，當相關單位要針對某兩位使用者進行監聽的動作時(如圖五)，便可以根據 USER ID 從資料庫中取出 INVITE 端的通話私鑰以及被 INVITE 端的通話公鑰，由此便可獲得雙方所使用的加密金鑰。而由於金鑰交換的動作是在建立 RTP 通道之前完成，所以不會有來不及獲得金鑰的問題，因此監聽將可順利進行。



圖四：第二階段 D-H 金鑰交換。



圖五：金鑰取得流程。

### 3.5 SRTP 加密演算法模式說明

在 SRTP 方面，需要演算法對數據串流進行加密和解密，即 AES 演算法，它是新一代加密標準，屬於對稱性密碼技術，也就是加密鑰匙(Encryption Key)與解密鑰匙(Decryption Key)為同一把金鑰，其優點是加、解密的速度快，而缺點則為如何使雙方可以安全地共享同一把金鑰。AES 是採用 Rijndael 的理論背景，Rijndael 是一個採用反覆運算來對資料進行加密的加密演算法，它提供可變動的資料區塊長度及可變動的金鑰長度。

SRTP 採用兩種型態的金鑰：Session Key 和 Master Key。Session Key 是直接用來做加密或者是訊息認證，而 Master Key 為一亂數位元字串，可透過特定的 Key Derivation Function 和一些參數來產生 Session Key。這些參數包括 Key Derivation Rate 和 Salt Key。Key Derivation Rate 是(1, 2, 4, ..., 2<sup>24</sup>)整數集合而 Salt Key 是用來抵禦 Pre-Computation 和 Time-Memory Tradeoff 攻擊。Master Key 必須透過 Key Management 的機制來得到，所以 Master Key 是具機密性，它不能讓通話雙方以外的第三方知道。AES 演算法有兩種模式：區段整數計數器與 f8，我們必須要把 AES 兩種模式作了解，在程式的架構上才知道如何的修改與 Porting 到 Windows 作業系統。以下為相關模式說明：

- 區段整數計數器模式：一種典型的計數器模式，此模式能將原始的區塊密文(Block Cipher)轉換成串流密文(Stream

Cipher)，此模式會根據連續計數器的值產生下一個密鑰串流區塊(Keystream Blocks)，然後再與原文區塊(Plaintext Blocks)做互斥或運算(XOR)而得到密文(Ciphertext)。一般情況下，幾乎所有能產生數列的函數方程式都能當做計數器來使用，只要它在一次循環中重複的次數不要太多就可以。但是，用於 SRTP 數據加密的僅僅是一個簡單的整數遞增計數器，而且 SRTP 的 AES 加密演算法預設值就是此一模式，它使用的是預設 128 位元長度的 Session Key 和預設 112 位元長度的 Salt Key。區段整數計數器模式特色是它允許對任意區塊的隨機存取，這一點對於 SRTP 的數據串流在可能遺失封包的不可靠網路上進行傳輸是非常必要的。

- f8 模式：輸出回授模式(Output Feedback Mode)的一個變種，它與區段整數計數器模式相同，它與區段整數計數器模式功能相同，能將原始的區塊密文轉換成串流密文。其 Session Key 和 Salt Key 的預設值和計數器模式下的 AES 是一樣的。運行在這種模式下的 AES 亦被用於 UMTS 3G 移動網路。

除了 AES 加密演算法，SRTP 還允許徹底禁用加密，此時使用的是所謂的「零加密模式」(NULL Cipher Mode)。它可以被認為是 SRTP 支持的第二種加密演算法，或者說是它所支持的第三種加密模式。事實上，零加密模式並不進行任何加密，也就是說，加密演算法把密鑰串流想像成只包含「0」的串流，所以做互斥或運算後就等同於原封不動地將輸入串流複製到輸出串流。這種模式是所有與 SRTP 相容的系統都必須被實作出來的，因為它可以被用在不需要 SRTP 提供保密性保證而只要求它提供其它特性(例如：認證和訊息完整性)的場合。

### 3.6 SRTP 認證、完整性和重傳保護

以上列舉的加解密演算法本身並不能保護訊息的完整性，攻擊者仍然可以偽造數據，至少可以傳送過去傳輸過的訊息數據。因此，SRTP 同時還提供了保護數據完整性以及防止重傳的方法：

- 保護數據完整性：為了進行消息認證並保護訊息的完整性，SRTP 使用了 HMAC-SHA1 演算法(在 RFC 2104 中定

義)。這種演算法使用的是預設 160 位元長度的 HMAC-SHA1 認證密鑰，但是它不能抵禦重放攻擊。

- 防止重傳：重傳保護方法建議接收方維護好先前已接收到的封包索引，將它們與每個新接收到的封包進行比對，並只接收那些過去沒有被接收過的新封包。這種方法十分依賴於完整性保護的使用，以杜絕針對封包索引的欺騙技術。SRTP 定義每一個 SRTP 封包對應一個索引值，其值是由一個 32 位元的正整數計數器(unsigned rollover counter, ROC)和 16 位元的 RTP sequence number (SEQ)組成，正整數計數器是用來記錄 16 位元 RTP SEQ 計數超過 65536 次後 reset 的次數。所以索引值的計算方式為  $i=2^{16} \times \text{ROC} + \text{SEQ}$ 。

SRTP 封包格式，其標頭(Header)與 RTP 封包相同。其中加密部分為 SRTP 封包中 Payload 部分；而認證和完整性部分，是針對整個 SRTP 封包及包含標頭和 Payload 部分，經過 HMAC-SHA1 演算法運算後得到一 4 位元組大小的 authentication tag，最後加在 SRTP 封包的最後。

### 3.7 LibSRTP 程式碼 Porting

由於 Vovida (VOCAL)所提供的 SRTP 原程式碼是架構在 Unix/Linux 系統上，然而我們實作的平台是 Windows，而且所使用的開發軟體為 Microsoft Visual C++ 6.0，所以 SRTP 有部份原始碼是無法在 Visual C++ 6.0 上通過編譯器編譯而執行，所以我們必須針對這些有問題的部份原始碼做修改，以下是我們整理出來需要修改原程式碼語法三大類：

- 變數型態的調整：在 Vovida 所提供的 SRTP 原程式碼中宣告的整數變數型態為 long long，然而在 Visual C++ 編譯器不允許，因此我們作語法上的修改，以便能讓程式能正常執行。
- inline 的編譯：在 Vovida 所提供的 SRTP 原程式碼中的許多副程式前加了 inline，使程式執行時能減少執行時間，但是 Visual C++ 的編譯器是不允許的，所以我們在此做了修正拿掉 inline 以便能讓程式能執行。

- 指標變數的 cast：在 Vovida 所提供的 SRTP 原程式碼中使用許多變數型態的指標，而這些不同變數指標在互相引用時都沒有作變數型態的 cast，所以我們作了引用指標修正，在指標前作變數型態 cast 的動作；另外原程式碼中還使用了宣告成 void 的指標作位址位移的動作，但是在 Visual C++ 編譯器中是不允許這樣的動作，所以 void 的指標也需要作 cast 的動作，我們嘗試的結果，是要把 void 的指標 cast 成型態為 octet\_t，如此程式才能正常執行我們的結果。

我們針對上述的問題，在原程式碼裡修改，以下為部分修改的程式碼片段。

```
err_status_t
rijndael_icm_alloc(cipher_t **c, int key_len) {
    extern cipher_type_t rijndael_icm;
    octet_t *pointer; //void *pointer;
    int tmp;

    /* allocate memory a cipher of type rijndael_icm */
    tmp = (sizeof(rijndael_icm_context) + sizeof(cipher_t));
    pointer = (octet_t *)malloc(tmp); //pointer = malloc(tmp);
    /* set pointers */
    *c = (cipher_t *)pointer; // *c = pointer;
    (*c)->type = &rijndael_icm;
    (*c)->state = pointer + sizeof(cipher_t);
    /* increment ref_count */
    rijndael_icm.ref_count++;
    /* set key size */
    (*c)->key_len = key_len;

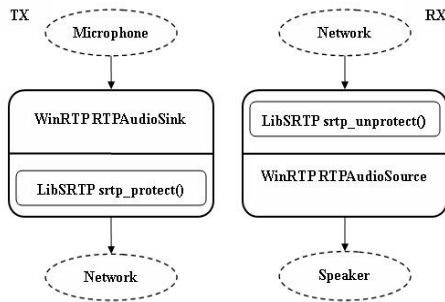
    return err_status_ok;
}
```

經修改過的部份程式碼可以讓 Vovida 所提供 SRTP 的 libSRTP 原程式碼中最重要兩個程式：加密程式 srtp protect 和解密程式 srtp unprotect 則能夠在 Windows 平台上正常執行。

### 3.8 SRTP 與 WinRTP 程式整合

我們將 Vovida 所提供的 libSRTP 原始碼 Porting 至 Windows 後，接著我們要與也是 Vovida 所提供的 WinRTP 做整合，使其最後變成所謂的 WinSRTP。圖六說明 SRTP 與 WinRTP 整合實作概念圖，分為傳送端和接收端兩部份。在傳送端部分，WinRTP 中的 RTPAudioSink 物件是將麥克風所輸入語音數據資料封裝成 RTP 封包格式送出，為了要實作 SRTP 機制，我們必須要在 RTPAudioSink 物件製作 RTP 封包後使用 libSRTP 的加密程式 srtp protect，使其最後變成 SRTP 封包後再送出。另一方面在接收端部分，SRTP 與 WinRTP 整合順序剛好與傳送端相反，從網路接收封包後馬上使用解密程式 srtp unprotect 將所接收到的 SRTP 封包轉變成 RTP 封包，接著再透過 WinRTP 的 RTPAudioSource 物件轉變成聲音由喇叭撥出，完成整個 WinSRTP 加解密的流

程。

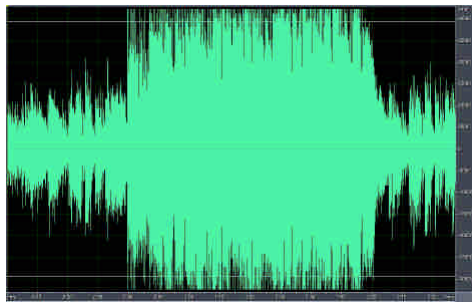


圖六：WinRTP與LibSRTP整合概念圖。

在libSRTP的加解密程式中所需要的金鑰部份，也就是Master Key，我們使用前述之兩段式D-H金鑰交換演算法（當然Basic D-H亦可，但其無法有監聽之功能），之後再配合SIP的訊息交換來達到加密金鑰的建立與交換。

### 3.9 SRTP 測試與驗證

整合完SRTP與WinRTP後，為了要測試與驗證SRTP的功能，我們特別針對WinRTP中不同的語音編碼(G711和G729)分別輸入固定時間長度大約一分多鐘的測試音樂檔，音樂內容為中文流行歌曲，其音樂波形表現如圖七所示，接著再分別觀察測試音樂檔在不同的語音編碼下，加解密前後的波形表現。驗證結果整理如下：

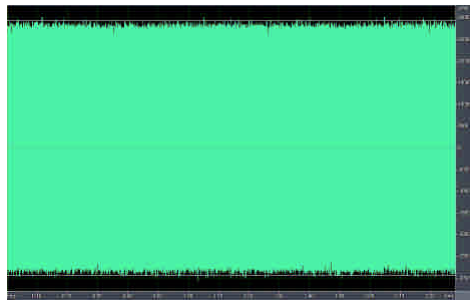


圖七：輸入測試音樂波形。

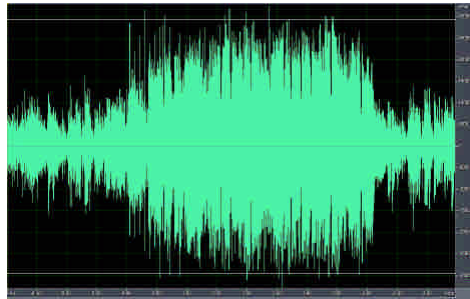
- 測試環境：使用兩台Windows平台的個人電腦，其中一台固定當傳送端而另一台則固定當接收端。為了要避免兩端長距離網路傳輸封包遺失問題，兩台測試電腦皆在區域網路內。

- G711編碼：圖八表示測試的音樂檔從傳送端經過編碼和加密後，再透過網路傳輸後抵達至接收端解密前的波形表現。實際聽到的聲音為連續的雜音，也確認RTP封包確實被加密而無法聽到測試的音樂。圖九則是接收端解密後的波形表現。

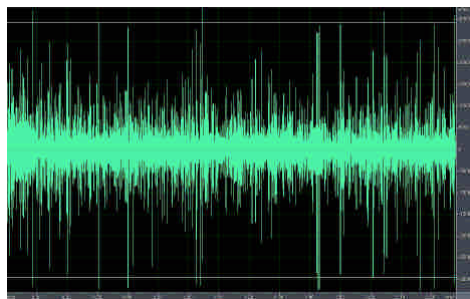
- G729編碼：圖十表示接收端接收到G729編碼後且解密前的測試音樂檔波形表現，實際聽到的聲音也是連續的雜音而且聲音較G711的來的小，圖十一則是接收端測試音樂檔解密後的波形表現。



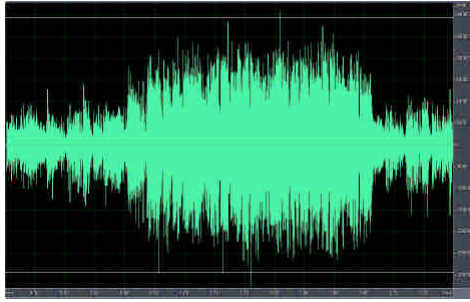
圖八：G711之RTP加密波形圖。



圖九：G711之RTP解密波形圖。



圖十：G729之RTP加密波形圖。



圖十一：G729 之 RTP 解密波形圖。

由於語音編碼 G729 資料壓縮率較 G711 大，所以理論上 G711 的語音品質會相較於 G729 來的好，也就是說在波形表現上，G711 的波形失真會比 G729 來的小。而在我們所作波形測試中可以觀察：以 G711 和 G729 解密後的波形跟原測試音樂檔波形作比較，G711 的波形較接近原測試音樂檔波形，如此也驗證理論的說法。

#### 4、測試結果與討論

由於 SRTP 的功能是使資料封包具備機密性，完整性和訊息認證，所以在 RTP 封包傳送到網路前，必須套用加密演算法作運算，故會多花費一些時間。為了觀察此花費時間對整個 WinRTP 系統效能的影響，因此我們做了兩個實驗來評估，第一個實驗為 SRTP 之處理時間效能測試，觀察製作 SRTP 封包需要花費多少時間。第二個實驗為 SRTP 與 RTP 之效能測試，觀察 WinRTP 在與 SRTP 整合前後的表現。另外，封包在網際網路傳輸時會有 Jitter 現象，為了觀察 SRTP 整合前後對 Jitter 的影響，所以最後的實驗為 SRTP 與 RTP 之 Jitter 測試，觀察兩者的 Jitter 表現。

##### 4.1 SRTP 之效能測試

為了得到製作 SRTP 封包時間，我們寫了一個測試程式，其內容先製作一個固定的 RTP 封包，RTP 封包的標頭和 Payload 都是預先給的固定值，接著針對不同的 Payload 長度，使用 srtp protect 副程式作迴圈來求得執行時間。數據結果(如表一)顯示，執行加密演算法的時間大約介於 250~281ns。

表一：SRTP 加密時間測試

Message Length(octets)	Time(ns)
256	281
128	260
64	256
32	256
16	250
8	250

##### 4.2 SRTP 與 RTP 之效能測試

將 SRTP 與原先的 WinRTP 整合後，稱作 WinSRTP，為了觀察整合前後的傳輸效能，我們做了以下的實驗：分別在 WinRTP 和 WinSRTP，輸入一測試音樂檔，觀察傳送不同的封包數時兩者各所花費的時間。測試結果(如表二)顯示，兩者時間差距不大。這是由於 WinRTP 的語音取樣率為 20ms，而 SRTP 所需多花費的加密時間單位(ns)遠小於取樣率時間單位(ms)，所以 SRTP 並不會影響 WinRTP 的傳輸效能。

表二：SRTP 與 RTP 之效能測試

Packet Number	SRTP Timer(ms)	RTP Timer(ms)
2000	38484	38438
4000	78468	78438
6000	118468	118438

##### 4.3 SRTP 與 RTP 之 Jitter 測試

在 WinSRTP 中，傳送端每隔 20ms 傳送一個 SRTP 封包，經過網際網路傳送後抵達接收端，封包透過一定距離的傳輸，接收端會有所謂的 Jitter，以下的實驗是要觀察 WinRTP 在與 SRTP 整合前後，是否對 Jitter 產生影響：傳送端輸入測試的音樂檔，分別在 WinRTP 和 WinSRTP 之接收端收集封包抵達時間，接著計算出 Jitter。Jitter ( $J$ )的計算公式為

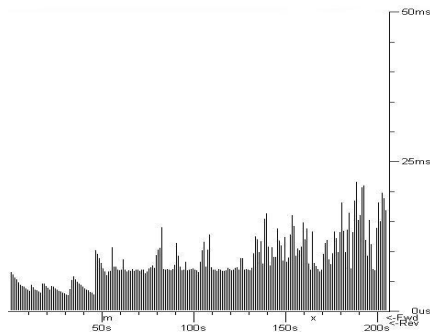
$$J = (1 - \frac{1}{16}) \times J + \frac{1}{16} \times |D(i-1, i)| \quad (1)$$

其中  $J$  為 Jitter 值，增益參數為 1/16，而  $D$  為兩封包間的延遲時間，其定義為

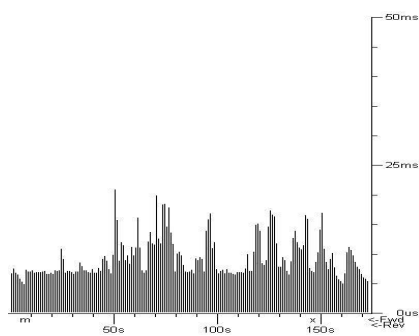
$$D(i, j) = (R_j - R_i) - (S_j - S_i) \quad (2)$$

$R_i$  代表第  $i$  個 RTP 封包的抵達時間， $S_i$  代表第  $i$  個 RTP 的 timestamp。因為封包透過網際網路傳輸，封包傳輸路徑不定，所以我們重複做了多次相同實驗來觀察，最後發現 WinRTP 與 WinSRTP 兩者的 Jitter 無特別的差異也無特

定的現象。圖十二和圖十三分別為收集 WinRTP 和 WinSRTP 其中一次的 Jitter 數據，橫軸為時間(s)，縱軸為 Jitter 值(ms)。



圖十二：Jitter over RTP。



圖十三：Jitter over SRTP。

#### 4.4 金鑰交換與記錄測試

將我們所提出的想法加入了 VOCAL 所提供的原始碼後，在本節中我們將說明經修改後的 VOCAL UA 依然可與未具加密功能的 VOCAL UA 來進行正常的通話，因為在我們所撰寫的程式中，針對加密部分我們所修改後的 UA 會自動的判別是否為有加密功能的 UA，以及是否為要進行加密的通話，經判別後其會自動的轉換需要對應模式，在 Server 部分則也是會自動地判別是否為加密的 VOCAL UA，並且觀察修改後的 Server 是否可正常的替使用者所使用的 UA 來做服務，無論他是否為我們經過加強後的 VOCAL UA 是原始的 VOCAL UA，甚至是其他符合 SIP 標準的 UA 如 X-Lite，只要是本 Server 的使用者，則須替使用者所使用的 UA 來進行服務，所以以下我們針對加強後的 VOCAL UA、加強前的 VOCAL UA 和 X-Lite 來做為測試與比較的對象，我們觀察其註冊、通話與金鑰記錄的狀態。結果如表三、表四所示。在舊版的 VOCAL Server 中，使用經過我們加強後的 UA 則在對 Server 的註冊與兩兩 UA 間金鑰的交換都皆可成功完成，可是舊版的 Server 並不會提供金鑰記錄，而舊版的 VOCAL UA 與 X-Lite 可以正常註冊但是只有普通未加密的通話

功，而舊版的 VOCAL Server 也不會提供金鑰記錄。而在加強後的 VOCAL Server 中，使用加強過的 UA 時，其結果不管是註冊、Server 金鑰的記錄與金鑰的交換都可以成功的運作，而使用舊版的 VOCAL UA 與 X-Lite 時對加強後的 VOCAL Server 在註冊與兩兩 UA 間金鑰的交換都依然可成功完成，不會因加強後而有所問題，可是因為舊版的 UA 與 X-Lite 本身就沒有加密功能所以自然也不會有金鑰記錄。

表三：OLD VOCAL Server 測試

SIP UA Version	Register State	Key Record	Key Exchange
NEW VOCAL UA	OK	NO	OK
Original VOCAL UA	OK	NO	NO
X-Lite UA	OK	NO	NO

表四：NEW VOCAL Server 測試

SIP UA Version	Register State	Key Record	Key Exchange
NEW VOCAL UA	OK	OK	OK
Original VOCAL UA	OK	NO	NO
X-Lite UA	OK	NO	NO

## 5、結論

我們於研究 SRTP 的相關文獻後，已成功修改由 Vovida (VOCAL) 提供在 Unix/Linux 平台上執行的 LibSRTP 部分原始碼，以 Porting 其至 Windows 平台，而和 WinRTP 的整合亦成功完成。透過語音波形的實驗驗證，我們可呈現其正常功能。所以完成之 SRTP 使通話雙方透過網路通話之內容具有機密性，確保不被竊聽者進行竊聽，同時也可避免遭受有意者的重傳攻擊。而藉由效能測試評估，我們觀察到 SRTP 幾乎不會降低和影響 WinRTP 傳輸接收端效能，因此 SRTP 確實值得被採用。此外，運用我們所提的兩階段式 D-H 金鑰交換方式，經過適當將 SIP 訊息的增修，UA 可在相當短的時間內完成安全性加密的建置，並且讓合法的管理者也可獲的使用者目前所使用的金鑰來進而達到監聽的目的。

## 6、致謝

感謝國科會電信國家型計畫(計畫編號 NSC 94-2219-E-011-006)對本論文研究在經費上的支持與國立交通大學資工系林一平教授和陳懷恩博士之協助與指導。

## 參考文獻

- [1] 古鴻炎, 邱舉明, 馮輝文, Oct. 2003, “PC 至 PC 之 VoIP 軟體開發,” 長冠科技建教合作案計畫結案報告, 台灣科技大學, 台北, 台灣。
- [2] 馮輝文, 邱舉明, 古鴻炎, 2004, “PC 至 Phone 之 VoIP 軟體開發,” 大景電訊建教合作案計畫結案報告, 台灣科技大學, 台北, 台灣。
- [3] 馮輝文, 邱舉明, 古鴻炎, 2004, “SIP 代理伺服器軟體開發,” 大景電訊建教合作案計畫結案報告, 台灣科技大學, 台北, 台灣。
- [4] IEIF, “RTP: A transport protocol for real-time applications,” <http://www.ietf.org/rfc/rfc3550.txt>, July 2003.
- [5] IETF, “The secure real-time transport protocol (SRTP),” <http://www.rfc-editor.org/rfc/rfc3711.txt>, March 2004.
- [6] U. D. Black, Voice Over IP, 2<sup>nd</sup> Edition, Prentice Hall, Jan. 2002.
- [7] A. S. Tanenbau, Computer Networks, 4<sup>nd</sup> Edition, Prentice Hall, 2003.
- [8] IETF, “MIKEY: Multimedia Internet Keying,” <http://www.rfc-editor.org/rfc/rfc3830.txt>, August 2004.
- [9] J. Arrko et al., “Key management extensions for session description protocol (SDP) and real time streaming protocol (RTSP),” Work in Progress.
- [10] IETF, “SDP: Session description protocol,” <http://www.rfc-editor.org/rfc/rfc2327.txt>, April 1998.
- [11] IETF, “Real time streaming protocol (RTSP),” <http://www.rfc-editor.org/rfc/rfc2326.txt>, April 1998.
- [12] F. Andreassen, M. Baugher, and D. Wing, “Session description protocol security descriptions for media streams”, Work in Progress.
- [13] M. Thomas and J. Vilhuber, “Kerberosized Internet negotiation of keys (KINK),” Work in Progress.
- [14] IETF, “The group domain of interpretation,” <http://www.rfc-editor.org/rfc/rfc3547.txt>, July 2003
- [15] Henning Schulzrinne’s SIP page <http://www.cs.columbia.edu/sip/>.
- [16] Vovida Open Communication Application Library (VOCAL) <http://www.vovida.org/>.
- [17] Radvision, “SIP: Protocol overview,” Radvision Technology White Paper.
- [18] Ubiquity, “Understanding SIP: Today’s hottest communications protocol comes of age,” a Ubiquity Technology White Paper.
- [19] L. Dang, C. Jennings, and D. Kelly, Practical VoIP Using VOCAL, O’Reilly & Associates Inc., 2002.
- [20] IEIF, “SIP: Session initiation protocol,” <http://www.ietf.org/rfc/rfc3261.txt>, Jun. 2002.
- [21] IEIF, “Session initiation protocol (SIP): Locating SIP servers,” <http://www.ietf.org/rfc/rfc3263.txt>, Jun. 2002.
- [22] IETF, “The session initiation protocol (SIP) and session description protocol (SDP) static dictionary for signaling compression (Sig-Comp),” <http://www.rfc-editor.org/rfc/rfc3485.txt>, Feb. 2003.
- [23] IETF, “Dynamic host configuration protocol (DHCP-for-IPv4) option for session initiation protocol (SIP) servers,” <http://www.rfc-editor.org/rfc/rfc3361.txt>, Aug. 2002.
- [24] H. Schulzrinne and J. Rosenberg, “The session initiation protocol: Internet-centric signaling,” IEEE Communications Magazine, pp.134-141, October 2000.
- [25] H. Schulzrinne and J. Rosenberg, “The IETF Internet telephony architecture and protocols,” IEEE Network, pp.18-23, May/June 1999.
- [26] J. Rosenberg, J. Lennox, and H. Schulzrinne, “Programming Internet telephony services,” IEEE Internet Computing, pp.63-72, May/June 1999.
- [27] IEIF, “Session initiation protocol (SIP): Locating SIP servers,” <http://www.ietf.org/rfc/rfc3263.txt>, Jun. 2002.