國立暨南國際大學資訊工程學系

碩士論文

SRTP 與合法監聽相關議題

-以 SDES 密鑰協商交換機制為例

SRTP and Related Issues on Lawful Intercept

指導教授：吳坤熹 博士

研 究 生：林文仁

中 華 民 國 九十八 年 六 月

致謝

很榮幸能夠在這美麗的暨大山城讀書，除了感謝父母的栽培外，最重要的就是指導教授吳坤熹老師不辭辛勞地用心指導，讓我在待人接物與分析事情有很多不同角度的看法，在口語表達、實作訓練皆有豐碩的進步。更讓我學習到吳老師精準且實事求是的研究態度與方法以及學習精神。

感謝柏州學長技術上的指導，讓我在程式設計有很大進步。另外，非常感謝實驗室的夥伴們，除了冷冰冰的機器設備外，還有一股關懷的暖流在身邊。感謝在畢業典禮時幫忙我求婚與祝福的政霖、嘉裕、汎嘉、韋霖、韋勳、韋立、瑋勵、霓雅、信富、楹旋、筱璇、揮雄、文玲以及我的家人。特別感謝暨大慈青社師姑師伯們、慈青們、禪悅社指導老師姜義勝老師、黃光璿老師、紫鈞、嬓馨、雅方、文瀚等等，因為你們的關心使我更加茁壯。

最後，感謝我的未婚妻，妳的鼓勵與支持是我成長的動力。這些成果因為有你們才顯得甘甜美味，願與你們一同分享。

論文名稱：SRTP 與合法監聽相關議題-以 SDES 密鑰協商交換機制為例

校院系：國立暨南國際大學資訊工程學系　　頁數：57

畢業時間：98 年 6 月　　學位別：碩士

研究生：林文仁　　指導教授：吳坤熹博士

論文摘要

科技帶給人們極大的便利，同樣地，也成為恐怖份子好用的犯罪工具。網路電話(Voice over Internet Protocol, 簡稱 VoIP)就是其中之一。近年來，已經有恐怖份子透過網路電話配合電視轉播進行即時犯罪引導，以躲避警方查緝。這讓習於處理傳統犯罪模式的執法單位無所適從，不能在第一時間掌控歹徒行蹤，從而打擊犯罪。SIP(Session Initiation Protocol)已確定是 NGN(Next Generation Networking)所使用的通訊協定之一，可用來建立、變更、結束通話連線。目前市面上的 SIP 網路電話的通話內容大多數是未加密狀態，主要是因為 SRTP(Secure Real-time Transport Protocol)的密鑰協商交換機制種類繁多，但是比較明朗化且容易建置的是 SDES(Secure Description)，所以本篇論文將使用 SRTP 進行合法監聽系統的開發，並以此探討其中所涉及的技術困難與相關的通訊協定。

關鍵詞：SDES、SIP、SRTP、合法監聽、即時監聽

Title of Thesis：SRTP and Related Issues on Lawful Intercept

Name of Institute：Department of Computer Science and Information Engineering,
National Chi Nan University Pages：57

Graduation Time：06/2009 Degree Conferred：Master

Student Name：Wen-Jen Lin Advisor Name：Quincy Wu

Abstract

Technology has brought great convenience to us, but it may also become a tool for criminals. VoIP (Voice over Internet Protocol) is such an example. In recent years, it has been report that in some scenarios, terrorists were watching live news on television and using the Internet phone to guide their gang to avoid the police attack. For terrorists equipped with these new technologies, traditional law enforcement agencies can not effectively locate the terrorists and further given them a strike. At present, the content of most VoIP calls is non-encrypted, because it is difficult to find a common set of key agreement protocols for SRTP (Secure Real-time Transport Protocol). In this thesis, we would like to present SDES (Secure Description) which is clear and easy to implement. It will be used as the example to illustrate how a VoIP monitoring system can be develop to intercept SRTP audio streams.

Keywords: Lawful Interception, Real-Time Interception, SDES, SIP, SRTP

目錄

圖目錄

表目錄

1. 緒論

1.1 簡介

自從 1876 年美國貝爾實驗室開發出第一具電話機之後，進一步為了有效地讓更多人能夠相互通話，在 1878 年第一個電話交換機(Telephone Exchange)誕生在美國康乃狄格州紐哈芬市(New Haven, Connecticut)。當時的電話交換機需要人工手動來接線，透過電話先告訴接線生你要連到哪一個號碼，接線生就會幫你接通。而後因人工操作有隱私上的困擾，才發展成自動電路交換(Circuit Switching)系統。它的發明者 Almon Brown Strowger 聲稱：「從此不再因為競爭對手的妻子是貝爾公司的接線生，而偷走我所有生意」[11][26]。電話網路的構成方式是，在一個城鎮中，每一具電話都會接到當地的電話交換機而形成一個當地的網路；而地方的電話交換機會跟其他地方的電話交換機相連，以這樣階層式的架構，漸漸地擴展到跨城市之間相連，再來是跨國、跨大陸、跨海洋之間相連，逐漸形成公共交換電話網(Public Switched Telephone Network, 簡稱 PSTN)。

電話網路雖然提供給人們通訊上的便利，但伴隨而來的是法律方面的議題。政府以國防、社會等安全訴求，立法要求電信業者提供監聽的功能，因此多年來在公共交換電話網之下已發展出完整的監聽機制。直到 1961 年，加州大學洛杉磯分校的萊納德・克萊恩洛克(Leonard Kleinrock)發表了第一篇關於封包交換(Packet Switching, 簡稱 PS)理論[16]的文章，讓電路交換的系統上，原本一條線路只能一個用戶獨占頻寬，變成一條線路可以有多個用戶分享頻寬，充分有效地運用原本的線路。後來，於幾十年間發展出網際網路(Internet)，而在 1995 年由一家以色列 VocalTec 公司發行全球第一套商業版網際網路電話軟體[2]，就此展開網路電話時

代的來臨。同時，也表示傳統的監聽模式不再適用於網路電話了。

1.2 研究動機

網際網路日趨普遍，網路頻寬也越來越大，行動寬頻上網也慢慢地趨向成熟的服務，在這樣的基礎下，人們只要負擔少許的費用，就可以輕易地透過網路電話跟世界各個角落的人連絡。也因為這樣的便利性成為恐怖份子犯罪的工具。根據紐約時報指出[12]，近年來，孟買恐怖份子最有力的武器既不是槍也不是手榴彈，而是擁有全球定位系統(Global Positioning System, 簡稱 GPS)的行動衛星電話，恐怖份子事先都已經研讀過 Google 地圖。而在巴基斯坦的恐怖份子看著現場電視轉播，再透過網路電話即時引導現場的槍手躲避警察的追緝，並可以指示下一個地點進行另一波攻擊。而類似的手法也接連發生。

下一代網路(Next Generation Networking, 簡稱 NGN)將全面採用網際網路協定(All-Internet Protocol, 簡稱 All-IP)，網路電話在信號(Signaling)上所採用的標準協定為 SIP(Session Initiation Protocol)[14]，語音資料的傳輸協定則是使用 RTP(Real-Time Protocol)[8]。目前關於網路電話的監聽方式，多半僅適用於未加密的 RTP 語音通訊。雖然市面上大部份的網路電話產品尚未使用加密模式，可是一旦網路電話內容被加密了，那麼想要在第一時間打擊犯罪，就會困難重重。因此有必要未雨綢繆，及早發展針對加密模式的監聽機制。

在語音加密上有很多方法，其中由 RTP 發展而成的 SRTP(Secure Real-time Transport Protocol)[17]可以達到高傳輸量(High Throughput)與低封包擴張(Low Packet Expansion)，非常適合用在即時性多媒體的內容保護。目前支援 SRTP 的網路電話軟體，例如，eyeBeam，其密鑰產生的方式是由軟體本身所建立的，發話端與受話端經由 SIP 裡的 SDP(Session Description Protocol)來進行 SRTP 密鑰交換，以對方的密鑰來解開 SRTP 封包才能聽到聲音。雖然已有人提出經由可信任

的第三方(Trusted Third-Party, 簡稱 TTP)來產生密鑰[22]，這樣可以統一管理密鑰。不過，在整體上並沒有一個標準的文件來規範網路電話的密鑰管理機制，所以目前的 SRTP 密鑰都是由軟體自行產生，形成監聽上的困難。為此，本篇論文將以被動式偵測的方法取得 SRTP 密鑰，來達到即時(On-line)與離線(Off-line)監聽網路電話的內容。

1.3 論文架構

在論文架構上，第二章將介紹背景知識與相關研究。第三章是系統架構與實作成果。第四章是效能測試與結果。第五章則總結論文並提出未來可進一步深入探討的主題。

2. 背景知識及相關研究

2.1 Session Initiation Protocol (SIP)

IETF(Internet Engineering Task Force)組織提出一個主從式架構的會談初始協定，即是 SIP，定義在 RFC3261。SIP 是在應用層(Application Layer)的純文字信號協定，可用在網路電話、多媒體傳輸、視訊會議等相關的連線建立、變動及中止。它可以使用與電子郵件相似的定址方法(如：sip:alice@example.com)，稱為 SIP URI(Uniform Resource Identifier)，這讓使用者可以容易地記住對方 SIP URI。SIP 使用與 HTTP(Hypertext Transfer Protocol)相似的要求/答應模式(Request/Response Model)[20]，例如，有指定的方法(如：INVITE)、200 OK 等等。SIP 在傳輸層可以透過TCP 或UDP 來傳輸，在安全性方面，可以選擇TLS (Transport Layer Security)將 SIP 封包加密。在本篇論文所使用的 SIP 代理伺服器(SIP Proxy Server)就是支援

TLS 的功能，以確保如 SRTP 密鑰等資料的安全。其典型建立通話的 MSC (Message Sequence Chart) 如圖 1。

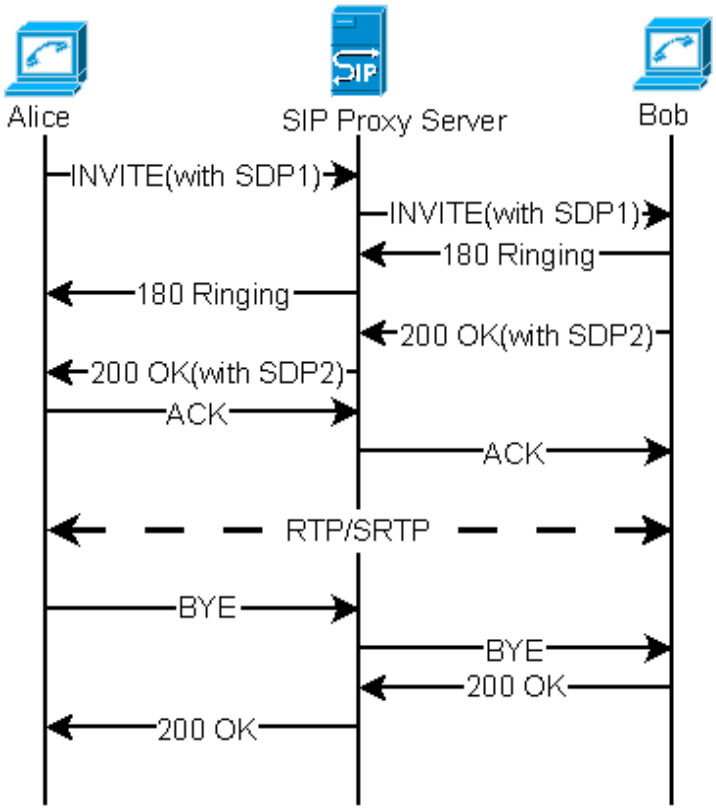

圖 1: SIP 訊息流程圖

2.2 Presence Service

在 IETF RFC2778 描述了一個系統模型，其主要有兩個服務：狀態服務(Presence Service)與即時傳訊服務(Instant Message Service)。這個系統允許使用者相互訂閱目前的狀態，而且在對方狀態改變時也能夠被通知，以及傳遞使用者的即時簡訊。狀態服務有兩類的使用者。第一類是稱為 PRESENTITY 的使用者，他們是狀態資訊(Presence Information)的來源者及發佈者；第二類是稱為 WATCHER 的使用者，他們是從狀態服務接收狀態資訊的使用者。其概念如圖 2[10]。

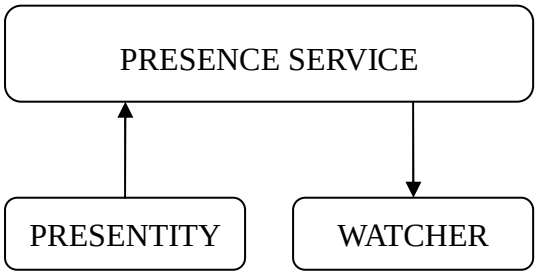

圖 2: Presence Service 概念圖

而 WATCHER 又分為兩種，一個稱為 FETCHER；另一個則稱為 SUBSCRIBER。FETCHER 的動作是從狀態服務取得某一位 PRESENTITY 的狀態資訊。而相對的，SUBSCRIBER 則是向狀態服務要求當某一位 PRESENTITY 的狀態資訊改變時就立即通知 SUBSCRIBER。

在本論文的架構中，SIP 伺服器必須支援狀態資訊的服務，也就是開啟 Presence Agent 模組(簡稱 PA)。PA 服務實現了即時傳訊(Instant Messaging, 簡稱 IM)所具備的查詢、更新和訂閱狀態資訊等功能。SIP-based PA 服務主要是由 SIP 的三種方法組成，分別是 SUBSCRIBER[3]、NOTIFY[3]、PUBLISH[1]。以本論文為例，SIP-based PA 服務的 MSC 如圖 3。內容說明如下：

第(1)、(2)步：中央伺服器(Central Server)發出 SUBSCRIBE 向狀態伺服器訂閱要求通知狀態變更的事件。然後狀態伺服器回覆 200 OK 確認已收到封包。

第(3)、(4)步：狀態伺服器回應當前的狀態給中央伺服器。

第(5)、(6)步：當監聽代理人(Spy Agent)偵測到有網路電話的時候，會將監聽相關的資訊(Intercept Related Information, 簡稱 IRI)PUBLISH 到狀態伺服器。

第(7)、(8)步：當狀態伺服器接收到監聽代理人 PUBLISH 的資訊會使用 NOTIFY 立即通知中央伺服器更新狀態，中央伺服器就會將資訊更新到資料庫的 IRI 資料表。

Presence 狀態資訊傳遞的格式是使用 XML 做為訊息主體(Message Body)，依然是使用 SIP 作為其通訊協定，其中 Content-Type 為 application/pidf+xml[9]。

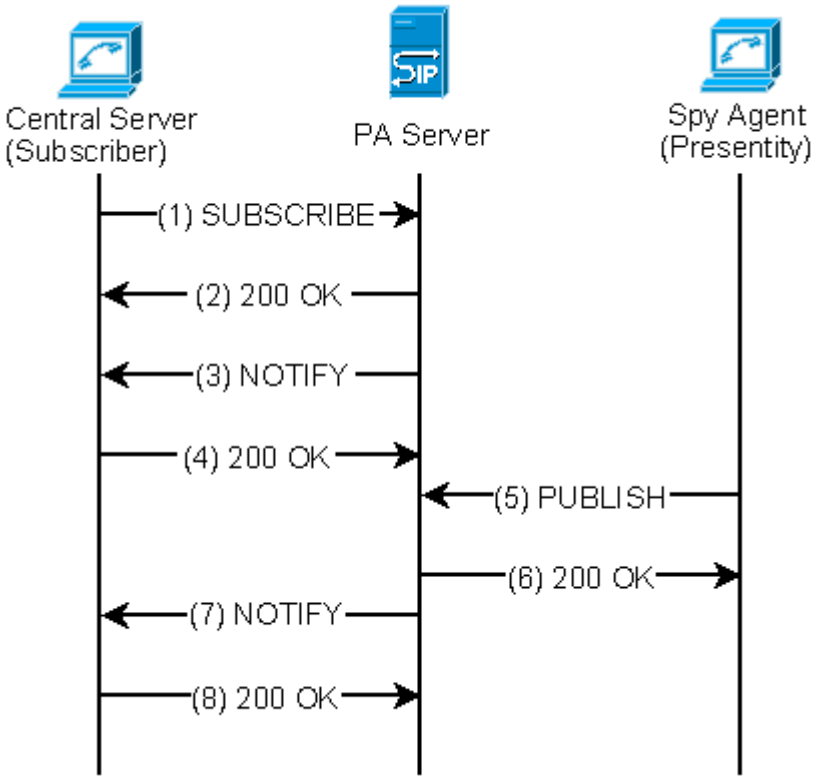

圖 3: SIP-based PA 服務訊息流程圖

2.3 RTP 與 RTCP

即時傳輸協定(Real-time Transport Protocol, 簡稱 RTP)[8]提供點對點(End-to-End)網路傳輸的功能，滿足應用程式傳輸即時性的資料，例如，聲音、影像或是需要有即時性的資料。RTP 本身只做資料傳輸的功能，並不能替資料串流提供可靠的傳送機制，也不提供流量控制或擁塞控制。而 RTCP(Real-Time Control Protocol)主要是提供資訊來輔助 RTP 傳輸資料，以達到流量控制和擁塞控制，其包含的資料像是已發送封包的數量、遺失封包的數量等統計資料。RTP 封包格式如圖 4。

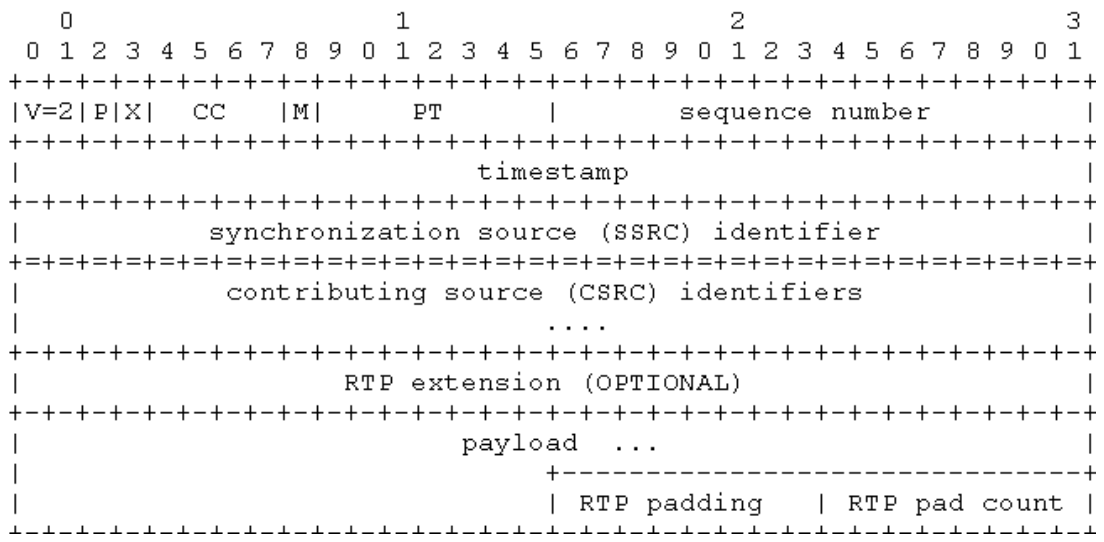

圖 4: RTP 封包格式

2.4 Secure Real-time Transport Protocol (SRTP)

在機密資料保護上，安全傳輸協定常見的應用是在電子商務交易的保全。而在串流(Streaming)媒體加密的技術中，有幾種安全協定可以使用在資料流上的保護，例如網路層(Network Layer)的 IP 安全協議標準(Internet Protocol Security, IPSec)、傳送層(Transport Layer)的安全傳輸層(Transport Layer Security, 簡稱 TLS)安全協定。但是這些安全協定都沒有考慮到即時應用的服務、手持設備的運算能力、有限的頻寬等問題，因此在諸如 3G 手機上網應用時，問題就會更加突顯。

針對上述問題及異質性網路環境，Secure Real-time Transport Protocol (SRTP)提出了一套適合的保護機制。SRTP 是以基礎架構(Framework)的方式來設計，其「實作編碼」與「整合保護」兩部分是獨立分開的，這使 SRTP 擴充能力很有彈性，並延長了協定可使用的壽命。SRTP 可以對單播(Unicast)或多播(Multicast)的 RTP 與 RTCP 資料流提供加密(Confidentiality)、訊息驗證碼(Message Authentication)及重傳保護(Replay Protection)。所以它非常適合用在網路電話。

即時性的資料流重視的是低延遲(Low Delay)，所以在加密的方法選擇就非常重要，因此對稱式(Symmetric)加密演算法是首選。Advanced Encryption Standard

(AES)不論是執行在軟體或硬體上，相對於非對稱式(Asymmetric)加密演算法是較快的，而且實作上也很容易。SRTP 將 AES 對稱式加密演算法訂定為預設的加密方法，且為必需實作的加密演算法。SRTP 的封包格式如圖 5，它只針對 RTP 的資料加密，並且保留原本的表頭(Header)。虛線的部分表示選擇性的(Optional)欄位。

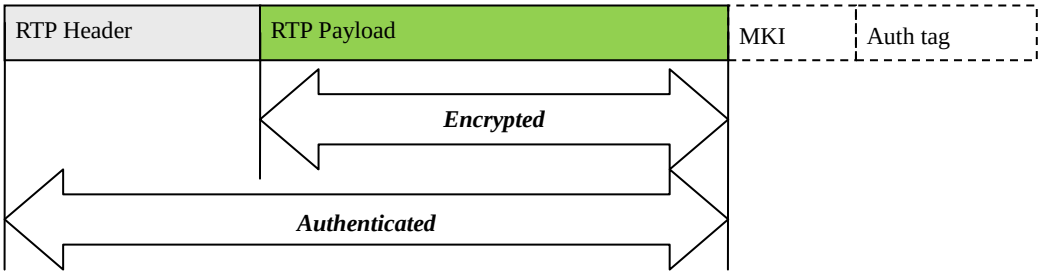

圖 5: SRTP 封包格式

另外，SRTP 與 SRTCP 的密鑰有生命週期，主要是以封包的數量來計算，如表 1。

表 1: SRTP 與 SRTCP 密鑰預設的生命週期

SRTP lifetime	2^{48} packets
SRTCP lifetime	2^{31} packets

發送端會根據每一個 RTP 封包的序號(Sequence Number)推算出封包的索引，SRTP 就以這個索引值作為 AES 加密演算法的初始向量(Initialization Vector)來將內容加密。它的計算公式如下：

$i = 2^{16} * ROC + SEQ$

- i：索引值。
- ROC：ROC 是 Rollover Counter 的縮寫，它是一個 32 位元的數值，用來記錄 16 位元的 RTP 序號被設為 0 的次數。RTP 序號超過 65535 時就會重設為 0，ROC 就要加 1。
- SEQ：RTP 的封包序號。

而接收端也要有適當的處理方法才能將 SRTP 解密，尤其是當 RTP 序號從 65535 變為 0 的時候。而索引值超過了 2^{48} 個就要重新產生密鑰(Re-keying)。

2.5 Session Description Protocol (SDP)

網路電話要能夠互通，必須要確定雙方所使用的語音編碼(Codec)相同，使用的網路位址，連接埠號等等，這些相關的資料都有了之後，才能正確的將語音封包送到正確的地方。而 SDP[18]主要的目的就是用來描繪多媒體串流所使用的相關資料，在初始化一個網路電話的通話扮演重要的角色。除了用在網路電話以外，SDP 也可以應用在多媒體格式的雙向通訊或單向串流等應用上。其交換資訊的方法則是透過諸如 SIP 等會談協定以要求/答應模式(Offer/Answer Model)[13]來達成。

SDP 透過數行的文字來傳遞連線資料，每一行的格式都是以<type>=<value>呈現，<type>(欄位)只能是單一字元而且分大小寫，<value>是有固定格式的字串，中間用「=」連接起來。以 c=IN IP4 224.2.17.12/127 為例，<type>就是 c，代表連線資訊(Connection Information)，<value>的部分為 IN IP4 224.2.17.12/127。「c=」的<value>包括三個欄位，即<network type> <address type> <connection address>。其中 IN 表示 Internet，IP4 表示位址型態是網際網路協定第四版，接下來就是連線的網路位址，最後 SDP 在每一行記錄都要加上換行符號(Carriage Return and Line Feed)。SDP 欄位在文件上有定義固定的順序，如圖 6 所示為常見的幾個 SDP 欄位，「*」表示選擇性的項目。

Session Description	
會議共同參數	v= (協定版本) o= (會議發起人) s= (會議名稱) c=* (連線資訊) t= (會議時間)
媒體參數 1	m= (媒體類型) c=* (連線資訊) a=* (媒體屬性)
媒體參數 2	m= (媒體類型) c=* (連線資訊) a=* (媒體屬性)

圖 6: SDP 結構與欄位表

2.6 SDP Security Descriptions for Media Streams (SDES)

2.6.1 SDES 簡介

由於 2.4 節所提到的 SRTP 專注在內容的安全，並沒有明確地定義要用什麼密鑰協商交換的方式，因此市面上百家爭鳴，光是密鑰交換的方式至少有十一種[7]。目前較多人實作也較容易使用是 SDES，發表於 IETF 的 RFC 4568 文件。雖然在現有的 SDP 提供 k 欄位做為傳遞加密的資訊，不過在格式上所能提供的資訊較缺乏彈性。所以 SDES 為 SDP 定義了一個新的屬性來描述密鑰及其他的參數，以提供在多媒體資料流上安全性的設定。SDES 文件裡更以 SRTP 為範例，說明如何使用這套機制來交換密鑰。

這個新屬性「crypto」表示的方法及說明如下：

a=crypto:<tag> <crypto-suite> <key-params> [<session-params>]

<tag>：是一個十進位的數字，作為加密的格式的索引，是協議的參數；如果

是 0 的話，代表不使用此加密格式。

<crypto-suite>：這個欄位是描述所提供的加密及驗證演算法，詳細格式如表 2：

表 2: Crypto-Suites

Crypto-Suites / Parameters	AES_CM_128_ HMAC_SHA1_80	AES_CM_128_ HMAC_SHA1_32	F8_128_ HMAC_SHA1_80
Master key length	128 bits	128 bits	128 bits
Master salt length	112 bits	112 bits	112 bits
SRTP lifetime	2^{48} packets	2^{48} packets	2^{48} packets
SRTCP lifetime	2^{31} packets	2^{31} packets	2^{31} packets
Cipher	AES Counter Mode	AES Counter Mode	AES F8 Mode
Encryption key	128 bits	128 bits	128 bits
MAC	HMAC-SHA1	HMAC-SHA1	HMAC-SHA1
SRTP auth. tag	80 bits	32 bits	80 bits
SRTCP auth. tag	80 bits	80 bits	80 bits
SRTP auth. key len.	160 bits	160 bits	160 bits
SRTCP auth. key len.	160 bits	160 bits	160 bits

<key-params>：這個欄位即是存放密鑰的地方，其格式為

<key-method> ":" <key-info>

key-method 在 RFC4568 裡唯一的定義是「inline」，實際密鑰即是放在 key-info 這個欄位裡，詳細格式如下：

"inline:" <key||salt> ["|" lifetime] ["|" MKI ":" length]

- <key||salt>就是由 SRTP master key 與 salt 所組成，以 base64 編碼[21]將二進位密鑰轉成可印出的文字。
- lifetime：是 SRTP master key 的生命週期，SRTP 或 SRTCP 最大可使用此 SRTP

master key 的封包數量，超過這個生命週期後，要更新 SRTP master key。

• MKI:length：SRTP 的 MKI(Master Key Identifier)與 MKI 的長度。

相關範例如下：

```
a=crypto:1 AES_CM_128_HMAC_SHA1_80
inline:d0RmdmcmVCspeEc3QGZiNWpVLFJhQX1cfHAwJSoj|2^20|1:32
```

<session-params>：是可選用(Optional)的參數，以提示接收端如何處理收到的封包。例如 UNENCRYPTED_SRTCP，表示 SRTCP 封包不加密，其他還有 UNENCRYPTED_SRTP、UNAUTHENTICATED_SRTP、KDR=n 等等，這些參數可指示接收端來建立特定連線。

2.6.2 SDES 密鑰協商交換機制

在 SIP 網路電話中，要指定使用 SRTP 加密連線需要在 SDP 媒體欄位(m=)設定傳輸協定，例如：RTP/SAVP(Secure Audio/Video Profile) 或 RTP/SAVPF，這樣就可以知道要建立加密連線。SRTP 的密鑰及相關參數存放在各別媒體描述(Media Description)之內。由於「crypto」是 SDP 「a=」欄位的新屬性，所以其交換方式也就是繼承 SDP 在 SIP 網路電話中的傳遞方法。其範例如圖 7。

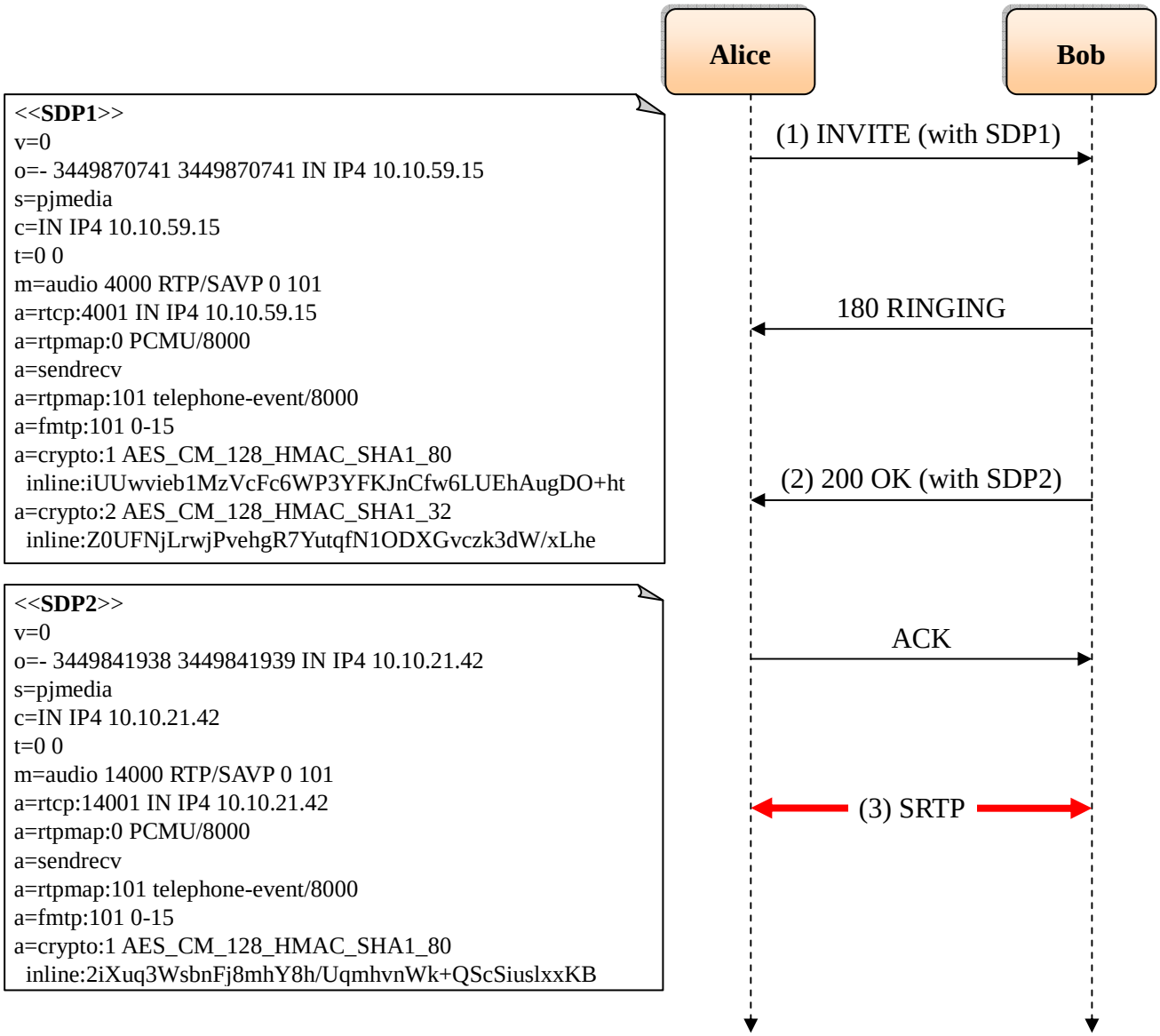

圖 7: SDES 訊息協商交換機制

圖例中步驟(1) Alice 發送 INVITE 給 Bob 請求通話，INVITE 的訊息主體 SDP1 會帶有 Alice 的 SRTP 密鑰。步驟(2)當 Bob 接起電話會回應帶有 SDP2 的 200 OK 訊息，同時也把 Bob 的 SRTP 密鑰傳給 Alice，這樣就完成了密鑰的交換。步驟(3)開始加密通話，雙方都可以用對方的密鑰解開 SRTP 封包。另外，SIP 訊息使用 TLS 加密，以確保密鑰不會被竊取。

2.7 監聽方法

當我們在建立一個合法監聽的解決方案，一定會遇到一個常見的問題，那就是主動式(Active)與被動式(Passive)監聽的問題，尤其是在網際網路的環境。首先我們必需了解主動式與被動式在合法監聽領域的意義。

主動式監聽是指調解設備(Mediation Device)的介面透過網路存取點監聽，例如：路由器、會談邊界控制器(Session Border Controller, 簡稱 SBC)、交換器(Switch)等網路元件，這存取點本身允許目標資訊的提供、會談資訊的交換和通話封包的複製。因為網路元件是根據調解設備的要求後，再由網路元件本身的裝置來辨識與複製目標封包，所以稱此介面為主動式。

被動式監聽是使用偵測器(Probe or Sniffer)來辨識及複製流量。其網路流量的取得方法可透過交換埠分析器(Switched Port ANalyzer, 簡稱 SPAN；或稱作 Port Mirroring)介面或是網路分流器(Network Tap)來達成。然後偵測器使用標的資訊動態地辨識與複製流量。因為偵測器是不是網路上主要的元件，也就是不負責網路封包交換或處理，所以它的存在與否並不會影響網路運作或服務提供。而且在觀念上是屬於網路外的元件，由外向內監視，所以稱為被動式。

這兩種監聽方式各有優缺點。主動式在資訊取得較為容易，而且設備可以再利用；但缺點是可能要做軟體升級或修改，以及設備的效能可能會受影響。被動式在網路內是容許任意移動的，不會因為移動而影響相關網路服務，不用在乎軟體的版本或是設備的模組，也就是平台獨立的；但缺點是設備較昂貴，以及可能會產生封包遺失的問題。

3. 系統架構與實作成果

我們主要是著重在既有的中央伺服器與監聽代理人的開發與改進。在前人的[27]中，已可以針對未加密的 RTP 通話進行監聽，卻無法聽到加密後的 SRTP 內容，這是目前網路電話監聽機制普遍的限制。因此我們將針對加密的部分進行處理以及程式臭蟲(Bugs)的修正。

本論文所採用的監聽方法是被動式監聽架構，並且是可分散式部署[27]監聽代理人，整體架構如圖 8。基本上，監聽代理人必須佈建在網路邊緣，除了截取 SIP 信號外，也是傳送監聽內容的重要元件。中央伺服器收集自監聽代理人傳來的 IRI 資料，而執法單位人員(Law Enforcement Agency, 簡稱 LEA)只要透過中央伺服器就可以要求監聽代理人將監聽內容傳送回來。在這個被動式監聽架構下，使用者不會注意到自己被監聽。相對地，如果他的封包被導向 RTPProxy 代為轉送[27]，雙方可以相互確認彼此的 IP 位址或使用適當的檢測工具，就會發現是否有被監聽的可能性。

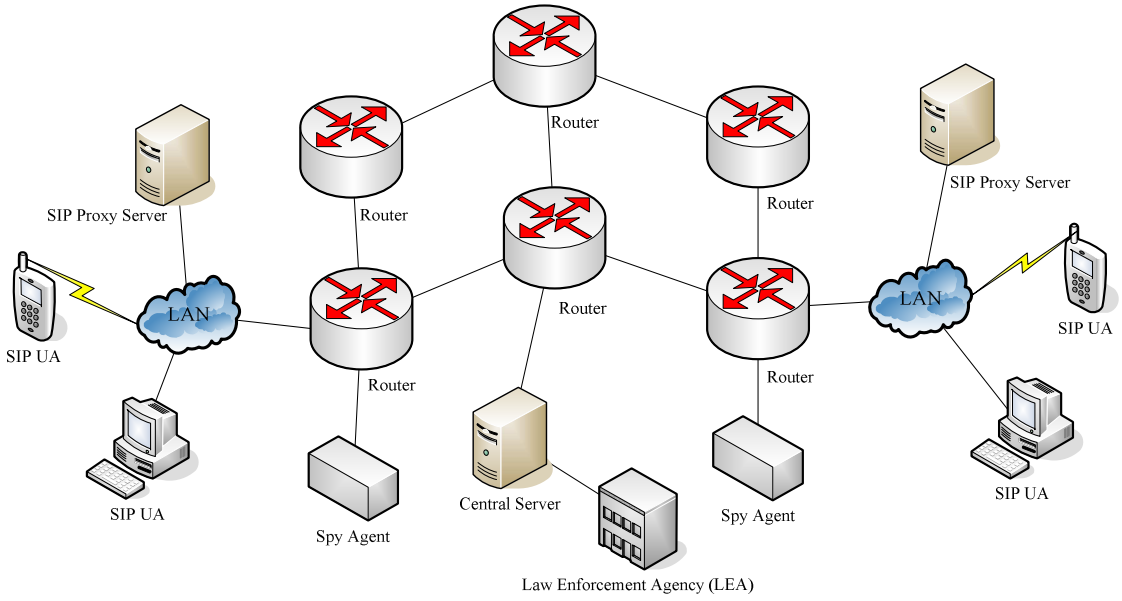

圖 8: 監聽架構示意圖

3.1 系統架構

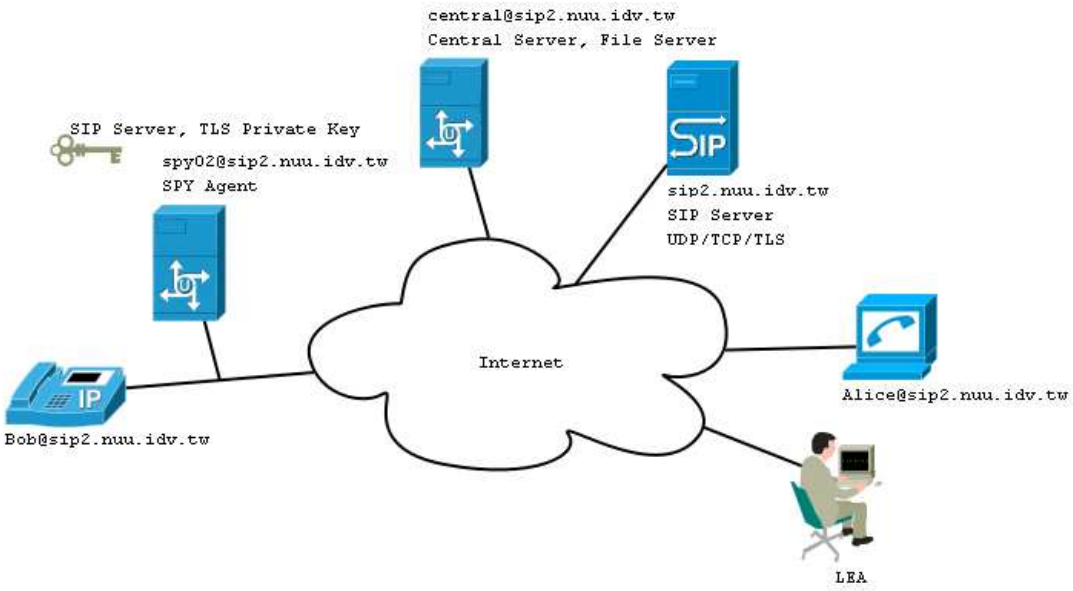

圖 9: 系統架構圖

在圖 9 所顯示的系統架構上，主要可分成四個元件來介紹：

1. 中央伺服器與網路電話伺服器(Central Server and SIP Server)：中央伺服器主要是儲存監聽內容、更新 IRI 資料表以及提供網頁介面給監聽人員，並且幫忙監聽代理人與監聽人員建立通話。網路電話伺服器提供註冊、狀態資訊等服務，並且支援 UDP/TCP/TLS 傳輸協定。在實測環境的 SIP 伺服器是使用 Open Source 的 OpenSIPS[19]。
2. 監聽代理人(Spy Agent)：主要部署在網路邊緣，收集信號、傳遞通話內容。它需要取得網路電話伺服器網路的私密金鑰(Private Key)，才能將 TLS 加密封包解開，進一步取出 SRTP 密鑰。
3. 監聽人員(Law Enforcement Agencies, 簡稱 LEAs)：他是執法單位人員，只須具備 SIP 網路電話以及網頁瀏覽器，就可以從網路上的任一端連到中央伺服器的監聽頁面，登入後點選目標進行監聽。
4. 網路電話 (User Agent, 簡稱 UA)：即被監聽的對象，可以是軟體或硬體電話。

3.2 系統功能加強及新增部分

3.2.1 離線監聽

離線(Off-line)監聽是本論文新開發的功能。如果通話內容是加密的，需要將離線檔案做解密的動作，其系統運作流程如圖 10。在 TLS 封包解密的部分，則是使用 Open Source 的 DSSL[25]函式庫進行解密，以取得 SRTP 密鑰。在啟動監聽代理人時需要提供 SIP 伺服器的 TLS 私密金鑰。離線監聽流程解說如下：

(1) 監聽代理人截取到 SIP 封包會將資訊以 PUBLISH 方式送到 SIP 伺服器，SIP 伺服器會 NOTIFY 中央伺服器以更新 IRI 資料表。而如果有偵測到密鑰，會一併更新 IRI 資料表。如果是 RTP 封包就會根據網路位址及連接埠號來更新 IRI 資料表。

(2) RTP 封包會經由 NFS(Network File System)透過 UDP 存到中央伺服器的硬碟。儲存的內容包括 RTP 表頭與資料的部分。而不將 RTP 表頭去掉的原故是因為 SRTP 在解密時需要表頭的資料。

(3) LEA 從監聽網頁介面登入並要求下載指定的離線通話內容，如圖 11 紅色實線框。若通話內容是加密過的，則可以看到所使用的加密演算法以及雙方的密鑰，如圖 11 藍色虛線框。另外，也提供了撥打的時間點。

(4) 收到 LEA 的要求後，中央伺服器會根據資料庫 IRI 資料找出指定的檔案。如果要求的內容是加密的，則會呼叫 decipher 程式來解密，並帶入相關密鑰將內容解密且暫存在硬碟。解密後會呼叫一個 Open Source 的 SoX[4]程式，將雙方聲音檔混音(Mix)。如果不是加密的，就將 RTP 表頭去掉，再將聲音檔混音。

(5) 最後經由網頁伺服器提供檔案給 LEA 下載。

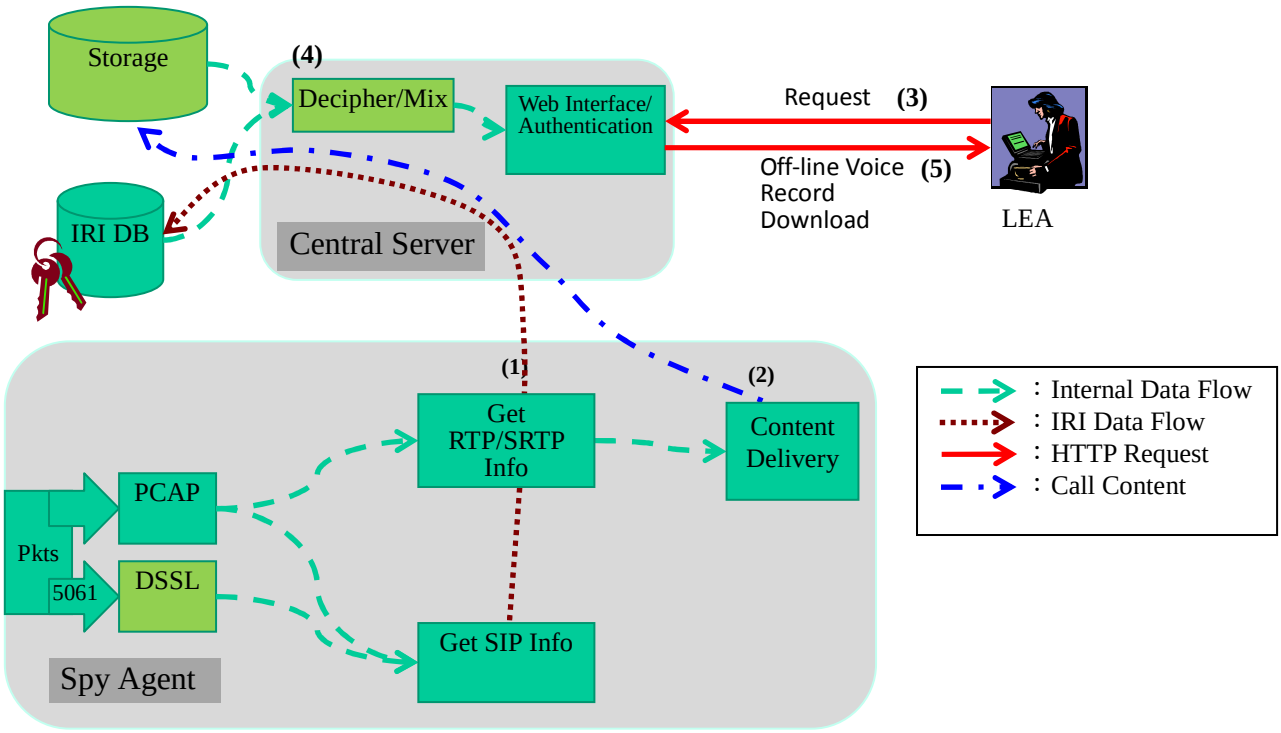

圖 10: 離線監聽系統運作圖

refresh logout online offline

LEA's SIP Address: alex@sip2.nuu.idv.tw 送出查詢

	No	Measurement_id(Click to download an off-line file.)	SPY_ID	CALL_ID	F_SIP_address	F_RTP_IP	F_RTP_Port
◉	6280	338436981_966367930	spy02@sip2.nuu.idv.tw	1-3817@10.10.21.103	1@10.10.21.103	10.10.21.103	6000
○	6278	338436981_966367930	spy02@sip2.nuu.idv.tw			10.10.21.103	6392

ort	State	Cipher Name	F_crypto_key	T_crypto_key	Node create time
	00001100	AES_CM_128_HMAC_SHA1_80	htRN/glbhcP7nbMNQY19aCsSCrmVcadL+mu2Y9tn	5VyVVy8GvxKilkqLmVRw5mrwZQPhjxyWP1zOovPt	June 19, 2009, 12:55 pm
	00000100				June 19, 2009, 2:50 am

圖 11: 監聽列表網頁畫面

3.2.2 即時監聽

在即時(On-line)監聽的部分需要即時將 SRTP 封包解密，然後傳送給 LEA。在未加入解密功能的系統裡，發話端與受話端各使用 160 個位元組儲存 G.711 的 RTP 資料，一有 RTP 封包進來就會更新。而在這裡我們改用環狀佇列(Circular Queue)儲存封包，以 G.711 的 SRTP 就要 182 個位元組(12 個位元組的 RTP 表頭+160 個位元組的已加密資料+10 個位元組的 SHA1_80 訊息驗證碼)，當儲存封包同時會將內容解密，如圖 12 的 `addQueue()`。虛線框的部分可以視為一個移動視窗(Sliding Window)，大小為 2 個封包，每次加入佇列就會移動一格。而 `fetchQueue()` 只讀取上一次所解密的封包，並且同時可多次讀取，以提供一個以上的 LEAs 監聽。

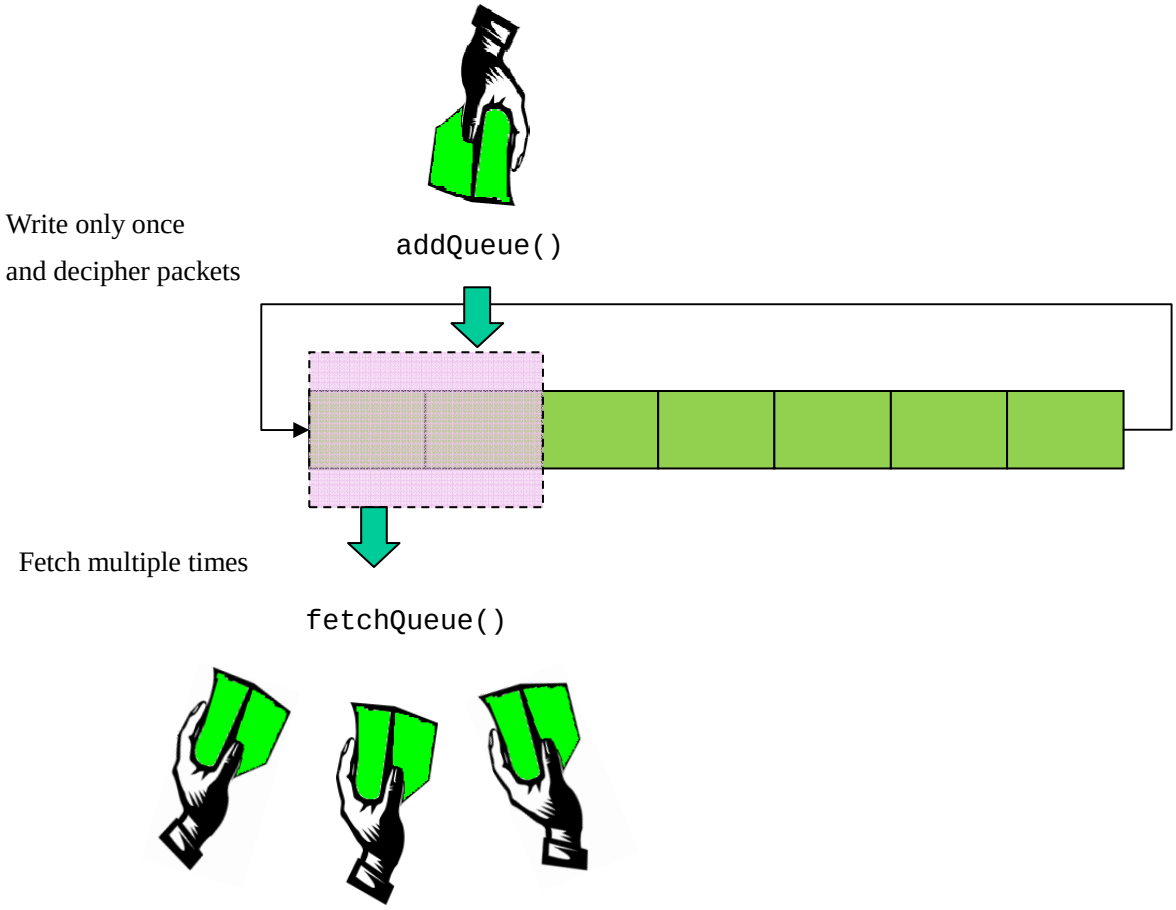

圖 12: 環狀佇列示意圖

以下是即時監聽流程(如圖 13)：

(1) 與離線監聽的部分一樣。通話中的 IRI 資訊會在記憶體中留存一份，直到通話結束。

(2) LEA 從監聽網頁列表(如圖 11)登入並要求監聽指定的通話中對象。

(3) 網頁伺服器呼叫一個 3PCC(Third Party Call Control) [15]外部程式並引入相關資料，這個外部程式會幫監聽代理人與 LEA 建立起通話連線。

(4) 通話建立成功後，監聽代理人開始將 RTP 封包加入佇列裡，如果是 SRTP，則同時也從記憶體中取得密鑰並將內容解密。

(5) 最後，將聲音混音並透過 Open Source 的 libortp[23]函式庫傳送給 LEA。

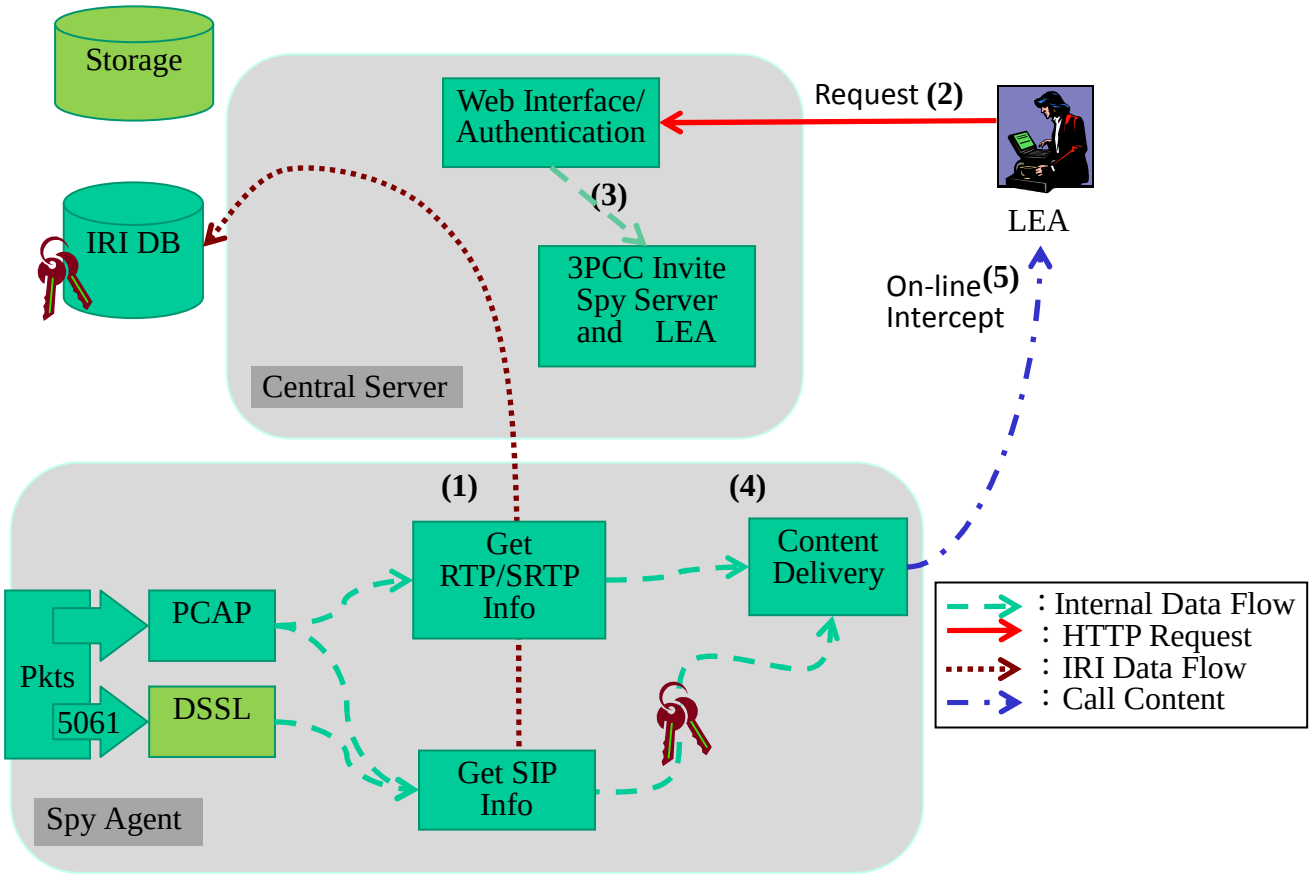

圖 13: 即時監聽系統運作圖

4. 效能測試

4.1 測試工具

4.1.1 PJSUA 簡介

PJSUA 是一個命令列(Command Line)模式下的 SIP 軟體電話，它是由 PJSIP、PJNATH、PJMEDIA 等處理 SIP、NAT、RTP 的函式庫所組成。函式庫的架構如圖 14。其函式庫具有跨平台的能力可在 Microsoft Windows、Windows Mobile、Linux、Unix、MacOS X、RTEMS、Symbian OS 等作業系統上執行；程式碼是以 C 語言來撰寫。

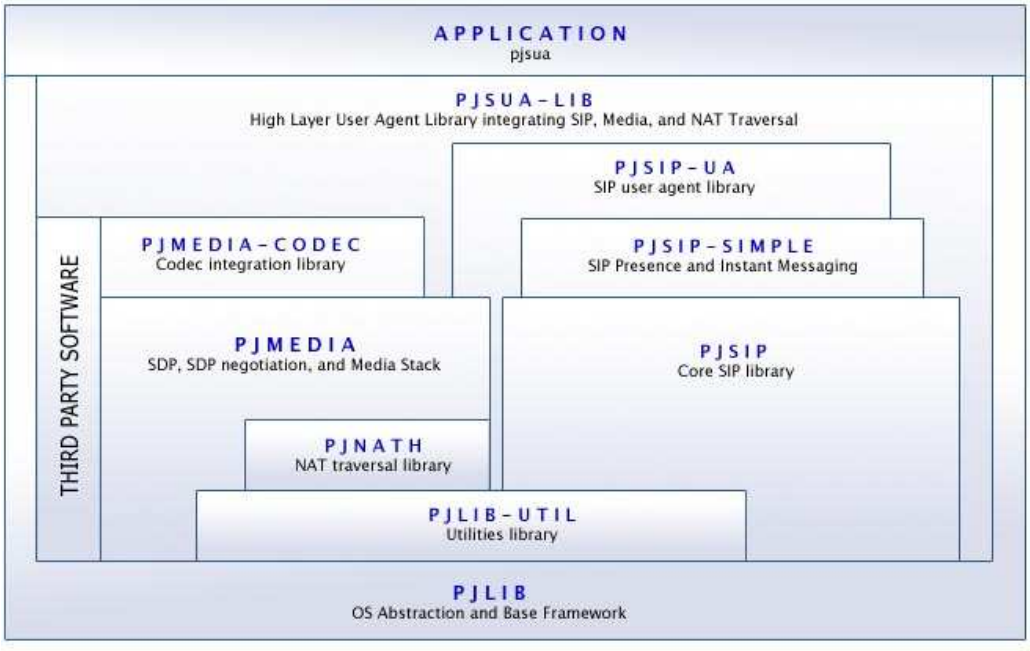

圖 14: 函式庫架構

儘管它只是命令列模式版本的軟體電話，沒有華麗的外觀，但是它支援的功能仍然非常齊全，請參考附件 C。在軟體授權上是以 GPL(GNU General Public

License)為主。

PJSUA 允許我們指定一個 WAV 聲音檔(單聲道，樣本大小：16-bit，樣本取樣率：16kHz，格式：PCM)做為對方接聽的內容。在各方面設定好之後，當有人打到 PJSUA 時就會自動接聽並播放此聲音檔。PJSUA 在 SIP 的傳輸協定支援 TLS，在媒體傳輸協定也支援 SRTP，因此非常適合做為我們實驗的工具。

4.1.2 SIPp 簡介

SIPp[24]是一套專門為 SIP 製作的效能測試軟體，它可以產生大量的 SIP 封包、RTP 封包與統計報告。我們使用 SIPp v3.1 版，這個版本支援 TLS，而 v1.0 不支援 TLS。SIPp 另外提供網頁界面，可以安排一連串的 SIP 測試，畫面如圖 15。SIPp 允許使用者自訂 XML 檔來描述 SIP 訊息的流程，從簡單到複雜的情況都可以模擬。例如附件 A，我們使用 SIPp 模擬 3PCC，用來測試監聽代理人。

另外，我們使用 SIPp 與監聽代理人測試時，發現監聽代理人並未正確截取到 SDP 內容。經過追蹤程式碼後，是 SIPp 從 XML 檔取出內容時最後一行的 CRLF 被略掉了。根據 RFC 4566，在每一行 SDP 記錄結尾都要加上 CRLF，而分析器(Parser)應該要能容錯。我們使用的「libosip2」SIP 函式庫並沒有容錯功能，導致 SDP 內容無法取得。在不修改原始碼的情況下，我們提供了一個函式來取得 SDP，如附件 B。

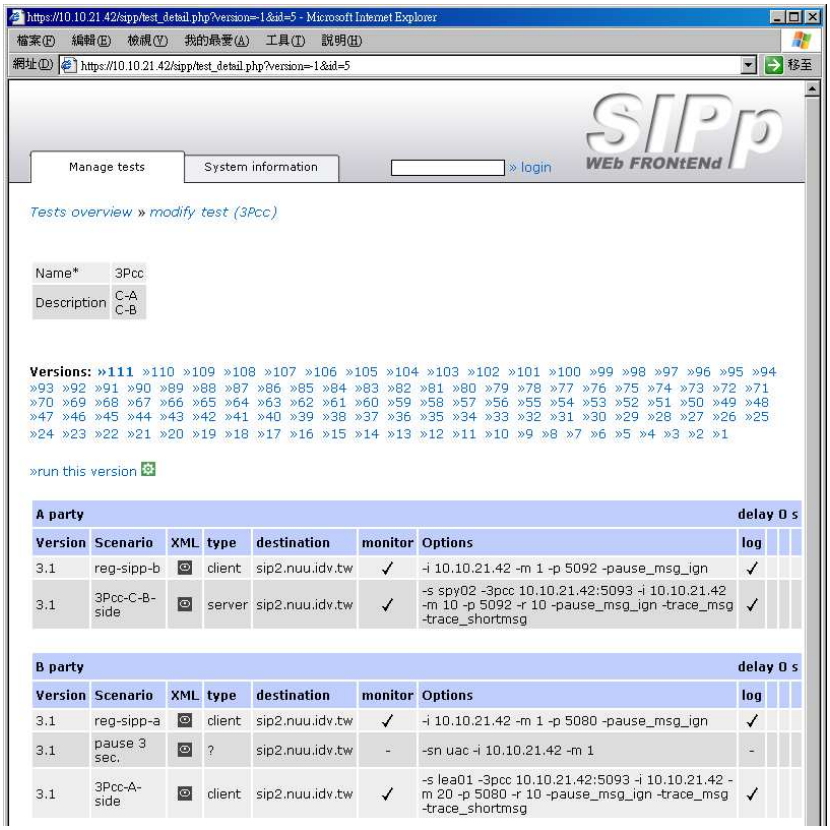

圖 15: SIPp 網頁介面

4.2 SRTP 效能測試

4.2.1 實驗環境

SRTP 使用 AES 對稱式加密演算法，所以在加解密處理時間上應該有較好的效能。為了測試它，我們寫了一個程式，並且在不同的中央處理器(Central Processing Unit, 簡稱 CPU)上執行加解密動作，同時記錄加解密所耗費的時間。另外，程式的輸入是一個包含 RTP 表頭及資料的 G.711 u-law 聲音檔，也就是說加密的資料長度為 160 個位元組。其封包數量為 30670 個。我們也分別測試了有 SHA1_80 驗證與沒有驗證的差別。以上內容整理如表 3。

表 3: SRTP 測試相關資料

機型	**CPU**	**RAM**
	Intel(R) Pentium(R) 4 CPU 3.20GHz	512MB
	AMD Athlon(tm) XP 1800+ 1541MHz	756MB
	AMD-K5(tm) Processor PR166	32MB
輸入檔	RTP 表頭 RTP 資料: G.711 u-law 總共 30,670 個 RTP 封包	
測試內容	分別測試有訊息驗證與沒訊息驗證所花費的加解密時間。訊息驗證碼的長度為 80-bit(SHA1_80)。	

4.2.2 量測結果

我們可以看到 AES 加解密的時間，以時脈最低的 AMD-K5-PR166 中央處理器來說，最長的時間不會超過 0.2ms，如果不做訊息驗證，更可以節省一半以上的時間。使用 Pentium 4 CPU 也顯示同樣的效果。因此我們在監聽代理人上解密選擇不做封包驗證，以減輕系統負荷。其結果如圖 16。

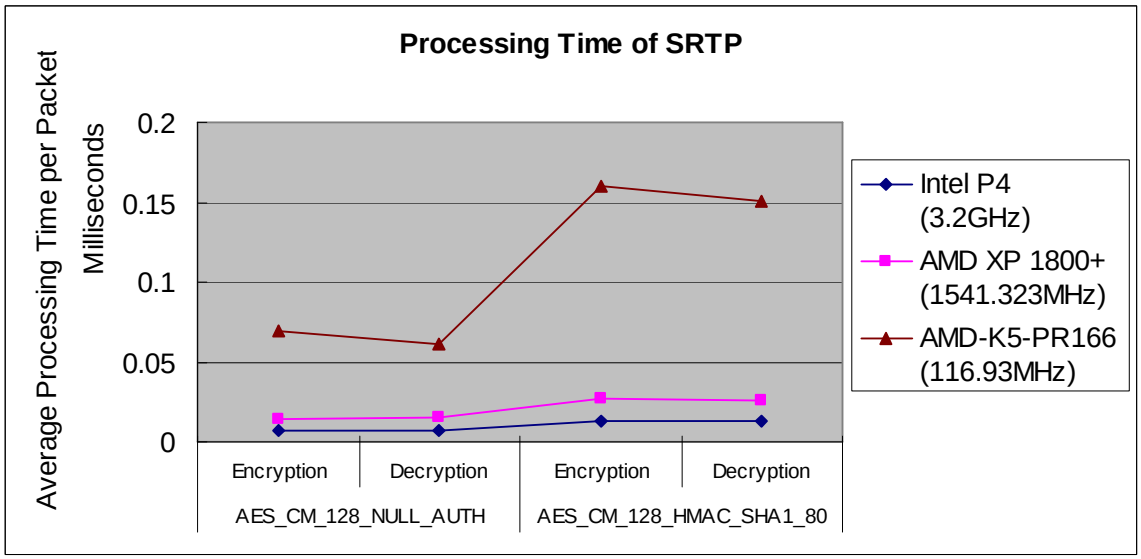

圖 16: SRTP 加解密處理時間

4.3 監聽代理人效能量測

被動式監聽的缺點是流量變大時會產生封包遺失。為了產生這個情況我們使用 SIPp 量測工具來模擬實際的通話情形。我們以一台電腦當作 UAC(User Agent Client)主動發話到另一台當 UAS(User Agent Server)的電腦，連線建立之後，就開始傳送 SRTP 封包，其持續時間為 40 秒。實驗環境分成兩種，實驗環境 1 的截取封包方式是透過較低階的集線器(Repeater Hub)；實驗環境 2 則使用高階交換器(Switch)的 SPAN 功能來截取封包。

4.3.1 實驗環境 1

量測的環境如圖 17。網路集線器廠牌為 Buffalo，型號為 LGH-M5P，半雙工(Half Duplex)，最大傳輸值為 10Mbps。紅色實線為監聽資料由監聽代理人所截取。藍色虛線為離線監聽資料從監聽代理人流向中央伺服器，傳送方式是使用架構在 UDP 傳輸層的 NFS 檔案系統。黑色粗線為被監聽者以 SIPp 模擬通話的內容流向，為雙向 SRTP 串流。

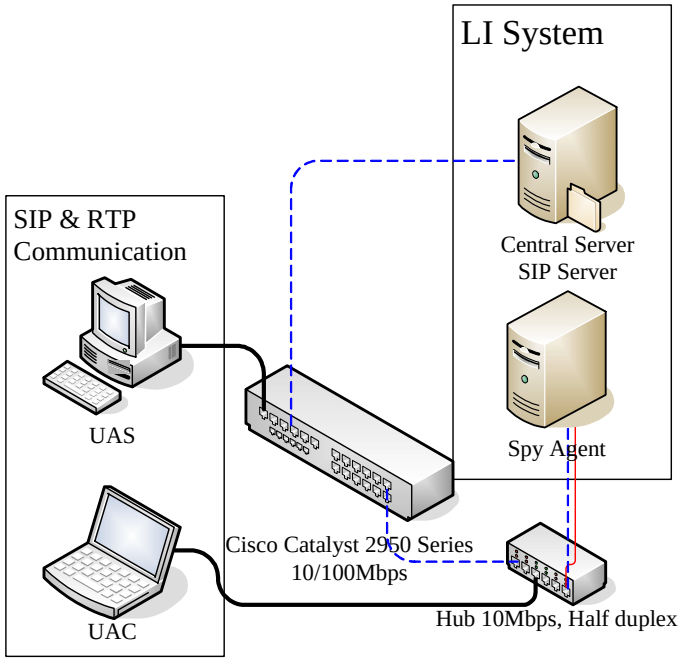

圖 17: 在集線器下實驗量測環境

監聽代理人系統平台如下：

硬體：

（1） CPU：Intel(R) Pentium(R) 4 CPU 3.20GHz

（2） RAM：512MB

（3） 網路卡 1：Broadcom Corporation NetXtreme BCM5753 Gigabit Ethernet PCI Express，10/100/1000Mbps

（4） 網路卡 2：D-Link Inc. DFE-530TX, Chip:VIA Technologies, Inc. VT6102 [Rhine-II]，10/100Mbps(在實驗 1 不使用，做為實驗 2 封包分析專用)

軟體：

（1） 作業系統：Linux Fedora Core 6

（2） libpcap：0.9.4-8.1

（3） libosip2：3.0.1-2.fc6

（4） libeXosip2：3.0.1-1.fc6

（5） ortp：0.13.1-3.fc6

（6） libsrtp：1.4.4

中央伺服器系統平台如下：

硬體：

（1） CPU：AMD Athlon(tm) XP 1800+ 1541MHz

（2） RAM：756MB

（3） 網路卡：Realtek Semiconductor Co., Ltd. RTL-8139/8139C/8139C+，10/100Mbps

軟體：

（1） 作業系統：Linux Fedora 9

（2） libosip2：3.1.0-1.fc9

（3） libeXosip2：3.1.0-1.fc9

（4） libxml2：2.7.2-1.fc9

（5） mysql：5.0.51a-1.fc9

UAC 系統平台如下：

硬體：

（1） 筆型電腦型號：Acer TravelMate 2350

（2） CPU：Intel(R) Celeron(R) M processor 1.30GHz

（3） RAM：1248MB

（4） 網路卡：Realtek Semiconductor Co., Ltd. RTL-8139/8139C/8139C+，10/100Mbps

軟體：

（1） 作業系統：Linux Fedora 10

（2） SIPp v3.1-TLS-PCAP

UAS 系統平台如下：

硬體：

（1） CPU：Intel(R) Pentium(R) Dual CPU E2180 @ 2.00GHz

（2） RAM：1GB

（3） 網路卡：Realtek Semiconductor Co., Ltd. RTL8111/8168B PCI Express Gigabit Ethernet controller，10/100/1000Mbps

軟體：

（1） 作業系統：Linux Fedora 8

（2） SIPp v3.1-TLS-PCAP

4.3.2 量測結果 1

圖 18 折線圖是以表 4 與表 5 的資料為基準所繪，顯示同時間通話量與封包截取遺失率的走勢。表 4 是關閉離線監聽功能，在同時通話數量中，UAC 與 UAS 於持續通話時間內傳輸的封包總量。而 SPY 欄位是監聽代理人所偵測到的封包數量。Packet loss 欄位是 SPY 減掉 UAC 與 UAS 的數量，即監聽代理人封包遺失量。我們可以看到圖 18 藍色線，當同時通話量到 60 通時，封包遺失就達到 21.57%。如果將 60 通的網路電話傳輸量計算出來，可 得 2(傳送與接收)*50(封包/秒)*224(每個封包大小=14(乙太表頭)+20(IP 表頭)+8(UDP 表頭)+12(RTP 表

頭)+160(RTP 資料)+10(SRTP Auth tag))*60(通話數)=1344000Bytes/s。而集線器的傳輸量為 10Mbps*1(半雙工)=1310720Bytes/s，所以同時有 60 通的傳輸量已經超出設備所能承受的限制，並且發生了大量的碰撞(Collision)，因此就產生封包遺失的情形。而 UAC 總傳輸量高於 UAS 不接近的原因是有部分的通話沒有辦法正常結束通話。因為當傳輸量到達集線器的限制時，同時也造成 SIP 封包的遺失，所以往往 UAS 雖然收到 BYE 封包，但 UAC 無法順利收到 UAS 的 200 OK 封包。在這期間 UAS 送完 200 OK 之後就停止傳送 RTP 封包了了，而 UAC 一直等不到 200 OK 封包，RTP 封包還是繼續在傳送，直到過了 200 OK 重送成功後才結束，所以造成傳輸總量的差異。

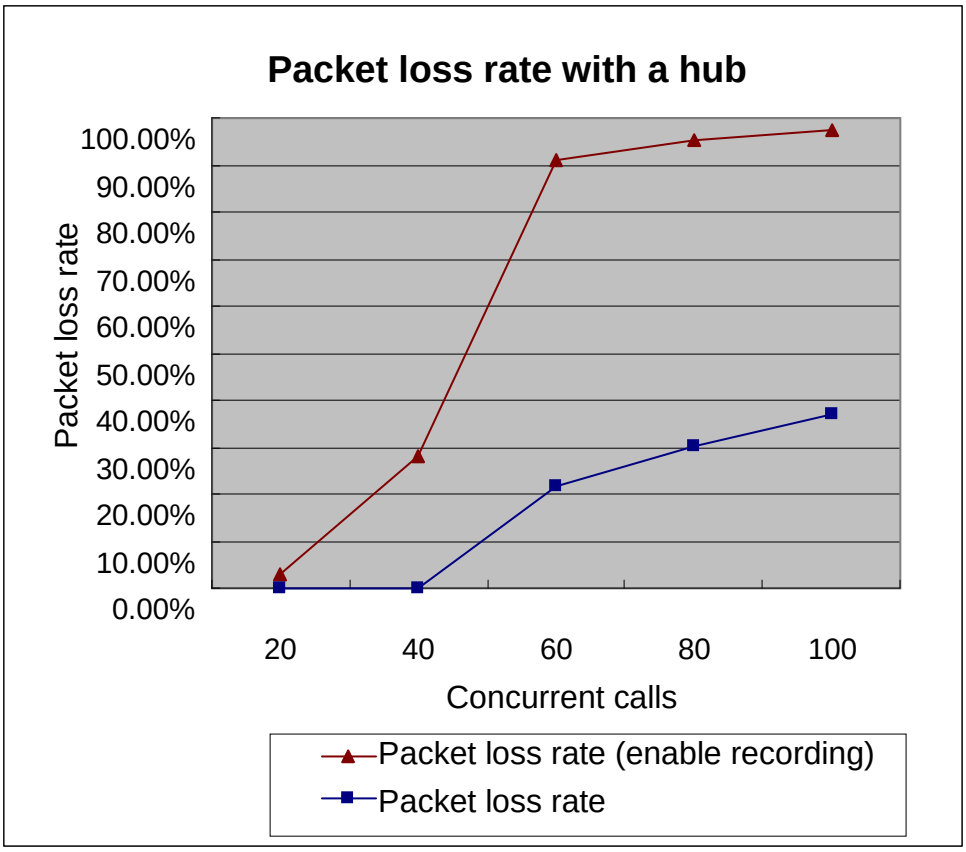

圖 18: 有無啟用離線監聽產生的封包遺失率(使用集線器)

表 4: 封包傳送量、封包遺失、CPU、Memory 狀態
(關閉離線監聽、使用集線器)

Concurrent Calls	UAC (Total send)	UAS (Total send)	SPY (Total recv)	Avg. CPU	Memory (Mbytes)	Packet loss	Packet loss rate
20	41920	43339	85259	3.86%	2.515	0	0.00%
40	87838	89494	177332	8.29%	2.515	0	0.00%
60	258687	214856	371382	7.06%	2.515	102161	21.57%
80	364352	235665	418567	7.30%	2.515	181450	30.24%
100	503243	337145	529169	7.19%	2.515	311219	37.03%

表 5 是有開啟離線監聽的功能，我們可以看到 40 通之後，封包遺失就非常嚴重。以 60 通同時的通話數來看就需要 1.34MBytes/s。除了網路電話本身的流量，同時監聽代理人也透過網路儲存封包，這樣還需要多一倍的頻寬來傳送，導致網路上的封包越來越多，互相搶占頻寬，而碰撞的機會也隨之增加，所以監聽代理人也就收不到。在這樣的情形下，連 SIP 訊息也無法傳遞成功，同時通話數也無法完全達到目標值。

表 5: 封包傳送量、封包遺失、CPU、Memory 狀態
(啟用離線監聽、使用集線器)

Concurrent Calls	UAC (Total packet sent)	UAS (Total packet sent)	SPY (Total packet recv)	Avg. CPU	Memory utilization (Mbytes)	Packet loss	Packet loss rate
20	41970	40988	80337	4.50%	3.018	2621	3.16%
40	84430	84306	121123	7.33%	3.520	47613	28.22%
60	155816	211550	33510	4.29%	11.064	333856	90.88%
80	195827	266046	21284	2.50%	6.538	440589	95.39%
100	207632	682566	22304	3.30%	3.520	867894	97.49%

4.3.3 實驗環境 2

我們第二個量測的實驗環境如圖 19。監聽代理人有兩張網路卡，一張是專門用來分析封包;另一張是連上網路傳遞內容。這次我們使用的是 Cisco Catalyst 2950 系列交換器，支援全雙工及 10/100Mbps 網路頻寬，並開啟 SPAN 功能，將 UAC 的封包複製一份到監聽代理人分析用網卡。紅色實線代表流經此交換器的封包會複製到監聽代理人。藍色虛線代表監聽代理人的監聽相關資料及離線監聽資料流向。黑色粗線是 UAC 與 UAS 使用 SIPp 模擬通話的線路，SRTP 為雙向傳輸。中央伺服器與監聽代理人的軟硬體規格跟實驗環境 1 一樣。

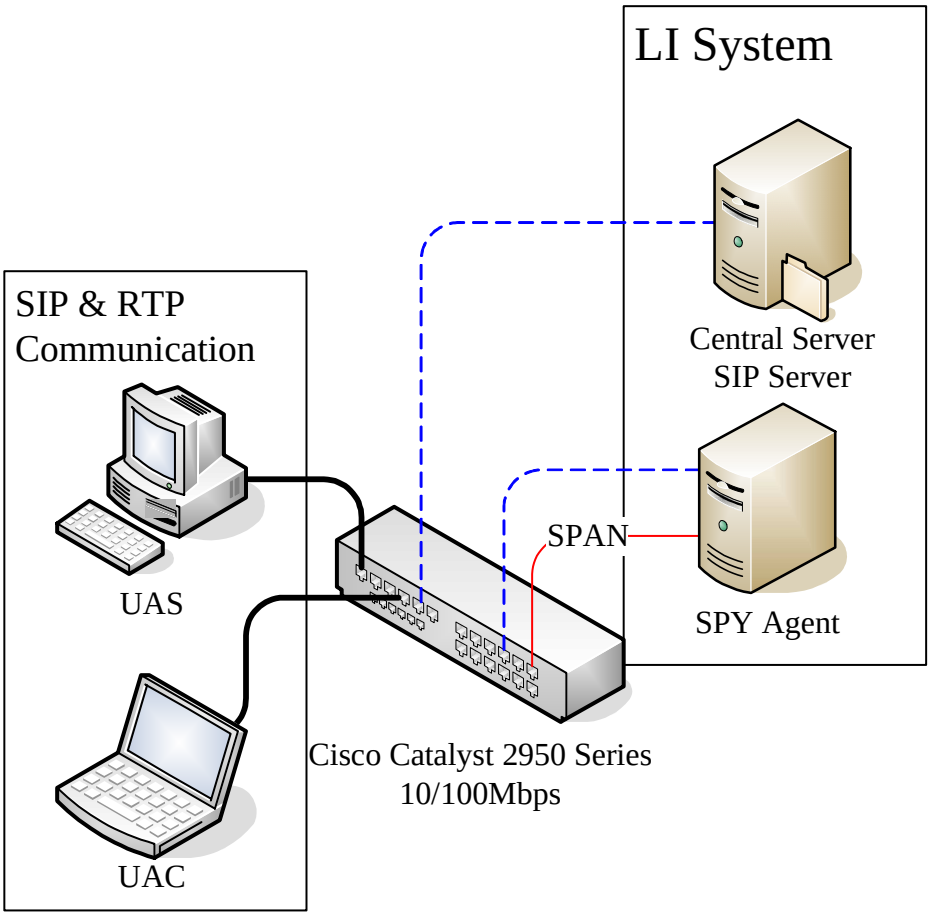

圖 19: 使用交換器的 SPAN 功能

4.3.4 量測結果 2

圖 20 是在這個實驗環境下，監聽代理人封包截取遺失率，詳細的資料可以參考表 6 與表 7。在 160 通同時通話數以下，封包遺失率與前次的實驗環境比起來大幅降低了 30 倍之多。這一方面是交換器的 100Mbps 頻寬比 Hub 的 10Mbps 提升了十倍，而且傳送及接收封包可同時進行，也就是全雙工(Full Duplex)的支援，封包碰撞的情形也沒有了。在提升網路設備的性能後，監聽代理人的 CPU 使用率因此也跟著提升，消耗更多的 CPU 資源在分析及處理封包上。而同時通話數可以順利達到預設目標值，連線可以正常建立及中斷，所以 UAC 與 UAS 的總傳送量是相近的。記憶體的消耗也是因此增加。

而在有開啟離線監聽功能下比起沒開啟的還是有少許的封包遺失，這很有可能是因為需要做儲存的動作而導致來不及處理封包。另外，有開啟離線監聽功能平均比關閉離線監聽還要多消耗37%倍的CPU資源，這是執行Disk IO(Input/Ouput)所造成。

在同時通話數 180 通以上，我們發現 UAC 這台筆記型電腦開始出現畫面稍微停滯的狀態。到了 200 通時，這個情形就更嚴重了，再到 220 通時，已經是動彈不得，而且通話建立無法順利達到預設目標，已經到了筆記型電腦的極限。因此保險估計 UAC 可同時通話數大約是在 180~160 通以下。由於 UAC 一時之間無法處理這麼多封包，所以 SIPp 顯示送出的封包與實際送出的封包是不相符的，這是設備限制造成的因素。

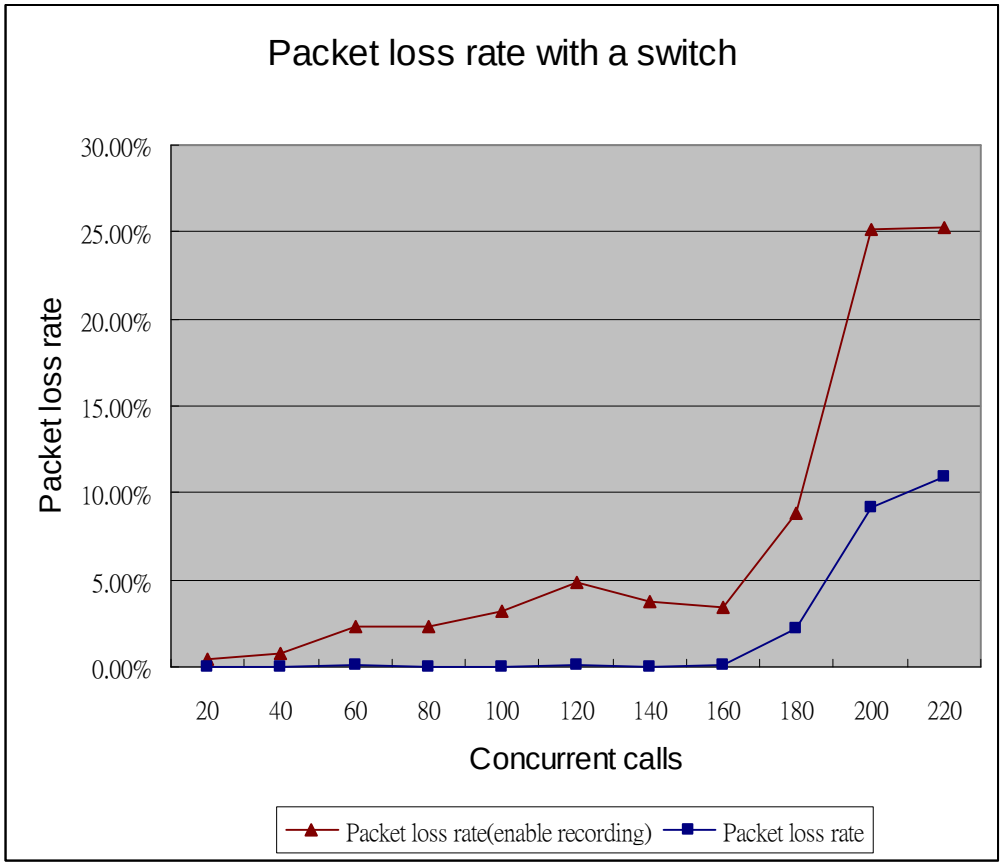

圖 20: 使用 SPAN 下，有無啟用離線監聽產生的封包遺失率

表 6: 封包傳送量、封包遺失、CPU、Memory 狀態
(啟用離線監聽、使用交換器)

Concurrent calls	UAC	UAS	SPY	Avg. CPU	MEM (Mbytes)	Packet loss	Packet loss rate
20	40210	41050	80924	2.08%	3.018	336	0.41%
40	80820	78712	158267	19.00%	3.520	1265	0.79%
60	121829	122456	238654	19.50%	4.526	5631	2.31%
80	163240	167240	322961	22.40%	5.029	7519	2.28%
100	204997	205670	397702	30.75%	6.035	12965	3.16%
120	241872	245690	464031	35.43%	5.029	23531	4.83%
140	282575	283336	544550	53.60%	5.029	21361	3.77%
160	323541	327225	628206	54.25%	6.035	22560	3.47%
180	654288	433894	992575	57.00%	6.035	95607	8.79%
200	1209945	1591763	2098252	53.33%	6.538	703456	25.11%
220	1192711	773964	1469728	54.69%	7.041	496947	25.27%

表 7: 封包傳送量、封包遺失、CPU、Memory 狀態
(關閉離線監聽、使用交換器)

Concurrent calls	UAC	UAS	SPY	Avg. CPU	MEM (Mbytes)	Packet loss	Packet loss rate
20	40208	39136	79343	5.10%	3.018	1	0.00%
40	80818	78637	159443	10.21%	4.023	12	0.01%
60	121829	124410	246077	16.28%	4.526	162	0.07%
80	161217	165140	326355	16.07%	5.532	2	0.00%
100	201323	202152	403386	24.53%	4.526	89	0.02%
120	244769	243629	488096	27.13%	7.544	302	0.06%
140	289874	287993	577666	38.00%	8.047	201	0.03%
160	330972	336118	666348	46.81%	7.041	742	0.11%
180	644673	413643	1035265	43.34%	8.550	23051	2.18%
200	959575	555912	1376053	38.32%	8.550	139434	9.20%
220	1464127	622190	1859052	37.84%	8.047	227265	10.89%

5. 結論及未來方向

理想的合法監聽不只是依賴設備而已，當它擴及到國家之間的監聽時，政治因素的層面影響更大。因此，國家的立法與國際的合作、協商左右了合法監聽的成敗。為了反恐、社會安全的考量，國際應共同組織合法監聽機構，來對抗犯罪事件。

即時監聽能夠提供執法人員在第一時間掌握罪犯的行蹤，本論文在這部分加入 SRTP/TLS 解密功能，即使通話是加密連線，仍然不會受到限制。而執法人員監聽的地點也不再受限於固定位置，行動力可以大幅提升。執法人員也並不需要一個客制化的設備，使用標準的 SIP 網路電話就可以監聽內容。我們也加入了離線監聽的功能，同樣可以將 SRTP 內容解密，讓每一通內容都可以成為法庭上最有利的證據。

聲音編碼除了常見的格式外，被動式監聽可能會遭遇到的問題就是當用戶使用自行定義的聲音編碼格式，如果沒有取得適當的編碼方法，將無法正確地聽到通話內容。我們所設計的離線監聽機制可將 RTP 封包內容儲存在伺服器裡，待套用正確的編碼方法即可播出內容。

未來我們可以將其他的密鑰協商機制加進來，以增加取得密鑰的範圍。我們也希望增加其他的聲音編碼，例如：G.729、GSM、iLBC、Speex 等等。

主動式監聽可以用在 SIP 信號的截取，若可分析並記錄每個 TLS 連線的加密方法(Cipher Spec)及用戶端私密金鑰，這樣監聽代理人就不需要截取每個 TLS 連線雙方交握(Handshake)的部分，也可以解出 TLS 封包。更進一步，在 NGN 的 IMS (IP Multimedia Subsystem)核心主要是 SIP 協定，可以使用主動式監聽，而監聽內容可透過主動式或被動式來截取，如此可以達到監聽範圍完全覆蓋的理想。

參考文獻

[1] A. Niemi, Ed., “Session Initiation Protocol (SIP) Extension for Event State Publication”, RFC 3903, IETF, Oct. 2004.

[2] Acceris Patent on Internet-to-Handset Telephone Calls, http://w2.eff.org/patent/wanted/acceris/acceris_prior.pdf, Sep. 1995

[3] B. Roach, “Session Initiation Protocol (SIP)-Specific Event Notification”, RFC 3265, IETF, Jun. 2002.

[4] Chris Bagwell, Sound eXchange, http://sox.sourceforge.net

[5] D. McGrew, libsrtp, http://sourceforge.net/projects/srtp/

[6] D. Wing et al., “SDP Security Descriptions for Media Streams”, RFC4568, IETF, Jul. 2006.

[7] D. Wing, “Overview of SIP Media Security Options”, IETF, Mar. 2006, http://www.ietf.org/proceedings/06mar/slides/raiarea-1/raiarea-1.ppt

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

[9] H. Sugano, et al, “Presence Information Data Format (PIDF)”, RFC3863, IETF, Aug. 2004.

[10] H. Sugano et al., “A Model for Presence and Instant Messaging”, RFC 2778, IETF, Feb. 2000.

[11] International Telecommunications Union - Historical Figures, http://www.itu.int/aboutitu/HistoricalFigures.html

[12] Jeremy Kahn, “Mumbai Terrorists Relied on New Technology for Attacks”, The New York Times, Dec. 2008.

[13] J. Rosenberg and H. Schulzrinne, “An Offer/Answer Model with the Session

Description Protocol (SDP)", RFC 3264, IETF, Jun. 2002.

[14] J. Rosenberg, H. Schulzrinne, G. Camarillo, A. Johnston, J. Peterson, R. Sparks, M. Handley, E. Schooler, "SIP: Session Initiation Protocol", RFC 3261, IETF, Jun. 2002.

[15] J. Rosenberg, et al, "Best Current Practices for Third Party Call Control (3pcc) in the Session Initiation Protocol (SIP)", RFC 3725, IETF, Apr. 2004.

[16] L. Kleinrock, "Information Flow in Large Communication Nets", RLE Quarterly Progress Report, Jul. 1961.

[17] M. Baugher, D. McGrew, M. Naslund, E. Carrara, K. Norrman, "The Secure Real-time Transport Protocol (SRTP)", RFC 3711, IETF, Mar. 2004.

[18] M. Handley, V. Jacobson, C. Perkins, "SDP: Session Description Protocol", RFC 4566, IETF, Jul. 2006.

[19] OpenSIPS, http://www.opensips.org/

[20] R. Fielding, J. Gettys, J. Mogul, H. Frystyk, L. Masinter, P. Leach, T. Berners-Lee, "Hypertext Transfer Protocol -- HTTP/1.1", RFC 2616, IETF, Jun. 1999.

[21] S. Josefsson, "The Base16, Base32, and Base64 Data Encodings", RFC 4648, IETF, Oct. 2006.

[22] Seokung Yoon et al., "Lawful Interception Scheme for Secure VoIP Communications using TTP", Computer Science and its Applications(CSA), IEEE, pages 149-152, Oct. 2008.

[23] Simon Morlat, libortp, http://www.linphone.org/index.php/eng/code_review/ortp

[24] SIPp. "A free Open Source test tool / traffic generator for the SIP protocol", http://sipp.sourceforge.net/

[25] SSLTech. DSSL Library, http://www.ssltech.net/dssl/index.html

[26] What's a Strowger?,

http://www.strowger.com/About-us/Strowger-Invention-of-Telephone-Switch.html

[27] 陳柏州,“支援網路電話即時監聽機制之分散式架構”, 國立暨南國際大學資訊工程學系, Jun. 2008.

附件

附件 A. SIPp 3PCC scenario XML code

這份附件是描繪 3PCC 的流程，用來測試監聽代理人效能的 XML 檔。圖 21 用來表示這整個通話信號流程。在 Controller-A-side 與 Controller-B-side 要使用參數「-3pcc 網路位址:網路埠號」來傳輸相關訊息。

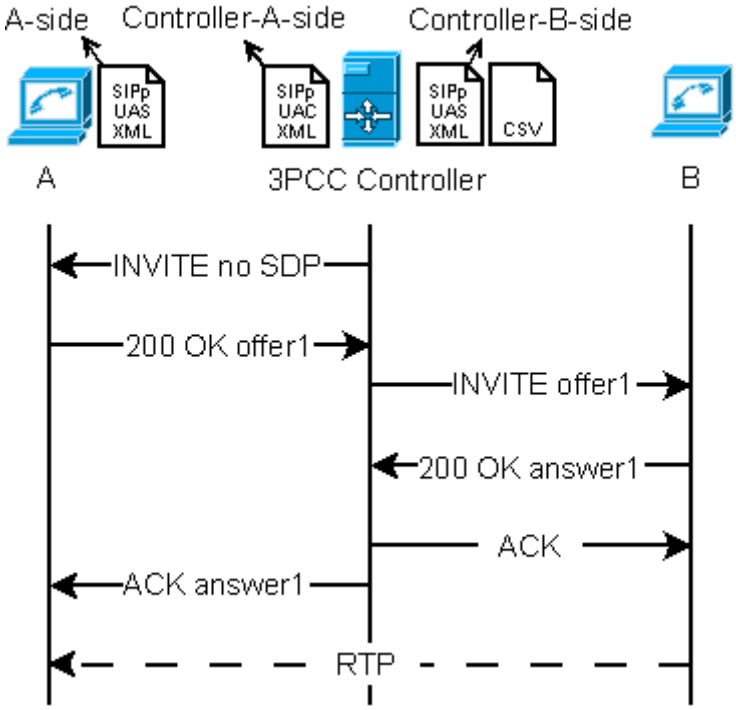

圖 21: 3PCC Call Flow

A.1 3PCC Controller-A-side

```
<scenario name="3PCC Controller - A side">
  <send retrans="500">
    <![CDATA[

      INVITE sip:[service]@[remote_ip]:[remote_port] SIP/2.0
      Via: SIP/2.0/[transport] [local_ip]:[local_port];branch=[branch]
```

```
      From: sipp-a
<sip:sipp-a@[local_ip]:[local_port]>;tag=[call_number]
      To: sut <sip:[service]@[remote_ip]:[remote_port]>
      Call-ID: [call_id]
      CSeq: 1 INVITE
      Contact: sip:sipp-a@[local_ip]:[local_port]
      Max-Forwards: 70
      Subject: Performance Test
      Content-Length: 0

    ]]>
  </send>

  <recv response="100" optional="true"> </recv>
  <recv response="180" optional="true"> </recv>
  <recv response="200" crlf="true" start_rtd="true">
    <action>
       <ereg regexp="Content-Type:.*"
             search_in="msg"
             assign_to="1"/>
    </action>
  </recv>

  <sendCmd>
    <![CDATA[
      Call-ID: [call_id]
      [$1]

     ]]>
  </sendCmd>

  <recvCmd>
    <action>
       <ereg regexp="Content-Type:.*"
             search_in="msg"
             assign_to="2"/>
    </action>
```

```
  </recvCmd>

  <send rtd="true">
    <![CDATA[

      ACK sip:[service]@[remote_ip]:[remote_port] SIP/2.0
      Via: SIP/2.0/[transport] [local_ip]:[local_port];branch=[branch]
      From: sipp-a
<sip:sipp-a@[local_ip]:[local_port]>;tag=[call_number]
      To: sut <sip:[service]@[remote_ip]:[remote_port]>[peer_tag_param]
      Call-ID: [call_id]
      CSeq: 1 ACK
      Contact: sip:sipp-a@[local_ip]:[local_port]
      Max-Forwards: 70
      Subject: Performance Test
      [$2]

    ]]>
  </send>

  <pause milliseconds="1000"/>

 <!-- The 'crlf' option inserts a blank line in the statistics report. -->
  <send retrans="500">
    <![CDATA[

      BYE sip:[service]@[remote_ip]:[remote_port] SIP/2.0
      Via: SIP/2.0/[transport] [local_ip]:[local_port];branch=[branch]
      From: sipp-a
<sip:sipp-a@[local_ip]:[local_port]>;tag=[call_number]
      To: sut <sip:[service]@[remote_ip]:[remote_port]>[peer_tag_param]
      Call-ID: [call_id]
      CSeq: 2 BYE
      Contact: sip:sipp-a@[local_ip]:[local_port]
      Max-Forwards: 70
      Subject: Performance Test
```

```
      Content-Length: 0

    ]]>
  </send>

  <recv response="200" crlf="true"> </recv>

</scenario>
```

A.2-1 3PCC Controller-B-side

```
<scenario name="3PCC Controller - B side">

<recvCmd>
  <action>
       <ereg regexp="Content-Type:.*"
             search_in="msg"
             assign_to="1"/>
  </action>
</recvCmd>

  <send retrans="500">
    <![CDATA[

      INVITE sip:[service]@[remote_ip]:[remote_port] SIP/2.0
      Via: SIP/2.0/[transport] [local_ip]:[local_port];branch=[branch]
      From: sipp-b
<sip:sipp-b@[local_ip]:[local_port]>;tag=[call_number]
      To: sut <sip:[service]@[remote_ip]:[remote_port]>
      Call-ID: [call_id]
      CSeq: 1 INVITE
      Contact: sip:sipp-b@[local_ip]:[local_port]
      Max-Forwards: 70
      Measurement_id: [field0]
      Subject: Performance Test
      [$1]
```

```
    ]]>
  </send>

  <recv response="100" optional="true"> </recv>
  <recv response="101" optional="true"> </recv>
  <recv response="180" optional="true"> </recv>
  <recv response="200" crlf="true">
    <action>
       <ereg regexp="Content-Type:.*"
             search_in="msg"
             assign_to="2"/>
    </action>
  </recv>
  <send start_rtd="true">
    <![CDATA[

      ACK sip:[service]@[remote_ip]:[remote_port] SIP/2.0
      Via: SIP/2.0/[transport] [local_ip]:[local_port];branch=[branch]
      From: sipp-b
<sip:sipp-b@[local_ip]:[local_port]>;tag=[call_number]
      To: sut <sip:[service]@[remote_ip]:[remote_port]>[peer_tag_param]
      Call-ID: [call_id]
      CSeq: 1 ACK
      Contact: sip:sipp-b@[local_ip]:[local_port]
      Max-Forwards: 70
      Subject: Performance Test
      Content-Length: 0

    ]]>
  </send>

  <sendCmd>
    <![CDATA[
      Call-ID: [call_id]
      [$2]

    ]]>
```

```
  </sendCmd>

  <pause milliseconds="1000"/>

<!-- The 'crlf' option inserts a blank line in the statistics report. -->
  <send retrans="500" rtd="true">
    <![CDATA[

      BYE sip:[service]@[remote_ip]:[remote_port] SIP/2.0
      Via: SIP/2.0/[transport] [local_ip]:[local_port];branch=[branch]
      From: sipp-b
<sip:sipp-b@[local_ip]:[local_port]>;tag=[call_number]
      To: sut <sip:[service]@[remote_ip]:[remote_port]>[peer_tag_param]
      Call-ID: [call_id]
      CSeq: 2 BYE
      Contact: sip:sipp-b@[local_ip]:[local_port]
      Max-Forwards: 70
      Subject: Performance Test
      Content-Length: 0

    ]]>
  </send>

  <recv response="200" crlf="true">
  </recv>
</scenario>
```

A.2-2 3PCC B-side CSV file

```
RANDOM
# Measurement_id: [field0=140966313_1709500062]
# field0
1104826931_438984759
```

A.3 3PCC A-side

```
<scenario name="3PCC A side">
  <recv request="INVITE" crlf="true">
  </recv>
  <send>
    <![CDATA[

      SIP/2.0 180 Ringing
      [last_Via:]
      [last_From:]
      [last_To:];tag=[call_number]
      [last_Call-ID:]
      [last_CSeq:]
      Contact:
<sip:lea01@[local_ip]:[local_port];transport=[transport]>
      Content-Length: 0

    ]]>
  </send>

  <send>
    <![CDATA[

      SIP/2.0 200 OK
      [last_Via:]
      [last_From:]
      [last_To:];tag=[call_number]
      [last_Call-ID:]
      [last_CSeq:]
      Contact:
<sip:lea01@[local_ip]:[local_port];transport=[transport]>
      Content-Type: application/sdp
      User-Agent: Sipp/3.1.0
      Content-Length: [len]

      v=0
```

```
      o=user1 53655765 2353687637 IN IP[local_ip_type] [local_ip]
      s=-
      c=IN IP[media_ip_type] [media_ip]
      t=0 0
      m=audio [media_port] RTP/AVP 0
      a=rtpmap:0 PCMU/8000/1
      a=sendrecv

    ]]>
  </send>

  <recv request="ACK" rtd="true" crlf="true"> </recv>

  <recv request="BYE" crlf="true"> </recv>

  <send>
    <![CDATA[

      SIP/2.0 200 OK
      [last_Via:]
      [last_From:]
      [last_To:]
      [last_Call-ID:]
      [last_CSeq:]
      User-Agent: Sipp/3.1.0
      Contact: <sip:[local_ip]:[local_port];transport=[transport]>
      Content-Length: 0

    ]]>
  </send>

  <!-- Keep the call open for a while in case the 200 is lost to be     -->
  <!-- able to retransmit it if we receive the BYE again.               -->
  <pause milliseconds="2000"/>
</scenario>
```

附件 B. Source Code of Getting SDP Function by Parsing SIP

Message

由於 SIPp 從 XML 檔取出 SIP 訊息時會略去最後一行的 CRLF，而 libosip2 對於這樣的情形並沒容錯的功能，導致無法成功取得 SDP。因此，在不改變兩邊原始碼的情況下，可用下面的程式碼取得 SDP。

```
sdp_message_t *get_sdp_way2(osip_message_t *osip)
{
        char *dest=NULL, *ckdest=NULL, *testsdp=NULL;
    char str[4000];
    int i, cnt=0;
    size_t length=0;
    sdp_message_t *sdp=NULL, *sdp2=NULL;

    if(osip==NULL)
        return NULL;
    i = osip_message_to_str(osip, &dest, &length);
    if(i || dest==NULL) { return NULL; }
    ckdest=dest;
    while(cnt!=4){ /* To find 0D0A0D0A (\r\n\r\n)*/
      if(*ckdest=='\r'||*ckdest=='\n')
        cnt++;
      else if(*ckdest=='\0') /* EOF */
      {
        free(dest);
        return NULL;
      }
      else
        cnt=0;
      ckdest++;
    }

    sprintf(str,"%s\r\n",ckdest);
    i = sdp_message_init(&sdp);
    if(i!=0)
    {
```

```
    DEBUG_print("SDP Initial failed!\n");
    free(dest);
    return NULL;
  }
  i = sdp_message_parse(sdp, str);
  if(i!=0)
  {
    DEBUG_print("SDP parse failed!section 1.\n");
    i = sdp_message_parse(sdp, ckdest);
    if(i!=0) {
      DEBUG_print("SDP parse failed!section 2.\n");
      i = sdp_message_to_str(sdp, &testsdp);
      if(i!=0) { DEBUG_print("SDP to_str failed!\n"); }
      else i = sdp_message_parse(sdp, testsdp);
      if(i!=0) {free(sdp); sdp=NULL;}
    }
  }

  if(testsdp!=NULL) free(testsdp);
  if(dest!=NULL) free(dest);
  return sdp;
}
```

附件 C. PJSUA Features

SIP features:

- Multiple lines/identities (account registrations).
- Multiple calls.
- IPv6 (added in version 1.2)
- PRACK (100rel, RFC 3262).
- UPDATE (RFC 3311).
- OPTIONS.
- Call hold.
- Call transfer (attended or unattended, with or without REFER subscription, RFC 3515, 3891, 3892, 4488).
- SIMPLE with PIDF and XPIDF support (SUBSCRIBE/NOTIFY, RFC 3265,

3856, 3863).

- Custom presence status text (RPID, RFC 4480).
- PUBLISH support (RFC 3903).
- Instant messaging (MESSAGE) and message composing indication (RFC 3428, 3994)
- UDP, TCP, and TLS transports.
- DNS SRV resolution for SIP servers (RFC 3263).
- DTMF with INFO (RFC 2976).
- STUN (RFC 3489bis).
- Digest AKA authentication (in development, RFC 3310, 4169).

Media features:

- Multiple Concurrent calls
- Conferencing
- Speex, iLBC, GSM, G711, G722, and L16 codecs.
- Wideband and ultra-wideband codec (Speex)
- More codecs via Intel IPP library: AMR-NB, AMR-WB, G.722.1 (Siren7), G.723.1, G.726, G.728, G.729A
- Stereo codecs (L16)
- WAV file playing, streaming, and recording.
- RTCP
- Call quality monitoring.
- RFC 2833
- Auto-answer, auto-play file, auto-loop RTP
- Tone generation.
- AEC (Acoustic echo cancellation).
- Adaptive jitter buffer.
- Adaptive silence detection.
- PLC (Packet Lost Concealment).
- Packet loss simulation.
- Multiple frames per RTP packet.
- SRTP (Secure RTP)

NAT traversal features:

- ICE (Interactive Connectivity Establishment, latest ICE draft).
- STUN (latest rfc3489-bis).
- TURN (latest draft-ietf-behave-turn)
- rport.

- SIP TCP and TLS keep-alive.
- Auto-detect and recover SIP UDP address change.
- Auto-detect ICE media transport change.