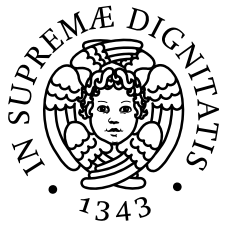
Università degli Studi di Pisa



Facoltà di Scienze Matematiche, Fisiche e Naturali
Corso di Laurea in Informatica

Prevenzione dello Spam via VoIP basata su Web of Trust

2 ottobre 2009

Relazione di Tirocinio

Candidato
Marco Cornolti
cornolti@cli.di.unipi.it

Tutore Accademico
Prof. Luca Deri
deri@ntop.org
Università di Pisa

Tutore Aziendale
Dott. Saverio Niccolini
Saverio.Niccolini@nw.neclab.eu
NEC Europe ltd.

Anno Accademico 2009/2010

Indice

Capitolo 1

Introduzione

1.1 Il tirocinio

Questa tesi per il Corso di Laurea in Informatica riguarda gli argomenti trattati, i problemi incontrati e le soluzioni trovate durante il tirocinio che il candidato ha svolto presso i NEC Laboratories Europe - Network Research Division, nella città di Heidelberg, Germania, tra il settembre e il dicembre 2008. Questo centro di ricerca, che conta circa 100 dipendenti, è la terza maggiore struttura di ricerca che l'azienda giapponese ha aperto in Europa. In questo centro si sviluppa il lato software delle tecnologie allo stato dell'arte sulle reti telematiche. Oltre allo sviluppo di protocolli e servizi, un ufficio si occupa di fare ricerche di marketing per indirizzare lo sviluppo nel mercato. Questo non impedisce tuttavia di produrre un importante lavoro teorico che spesso non ha ritorni immediati in termini economici.

Vengono sviluppati progetti, prototipi e viene data una grande importanza ai processi di standardizzazione. Effettivamente, gran parte del lavoro mira all'interazione con altre aziende o università per costruire standard, lavoro coordinato da organismi internazionali come IETF[1], IEEE[2], 3GPP[3], OMA[4] al quale il centro di Heidelberg dà una grande importanza.

Una caratteristica della ricerca industriale NEC è di puntare anche sulla formazione, infatti il centro di ricerca Neclab-NW dà spazio a circa 20 studenti di lauree triennali, specialistiche e dottorati provenienti da tutto il

[1] Il Internet Engineering Task Force si occupa di elaborare gli standard dei servizi Internet con dei documenti chiamati RFC.

[2] l'Institute of Electrical and Electronics Engineers è l'istituzione principale che si occupa di elaborare gli standard delle reti telematiche prevalentemente a livello fisico (hardware).

[3] Il 3rd Generation Partnership Project elabora gli standard che riguardano le reti di terza generazione, ovvero l'insieme di tecnologie che permettono forme di comunicazione mobile su reti cellulari avanzate rispetto agli standard GSM e GPRS, come ad esempio lo streaming video.

[4] La Open Mobile Alliance raggruppa i maggiori produttori di telefoni portatili e definisce standard aperti per queste forme di comunicazione.

mondo, selezionati in base alla carriera universitaria o alla particolare abilità dimostrata in un campo. Questi studenti svolgono il tirocinio o la tesi di laurea lavorando sui progetti portati avanti dai gruppi costituiti nel laboratorio, e vengono trattati come impiegati a tutti gli effetti: hanno un orario di lavoro di 8 ore al giorno e ricevono uno stipendio.

Gli studenti non svolgono mai mansioni secondarie o di routine. Al contrario, viene loro data una grande autonomia di lavoro e la responsabilità di sviluppare dei moduli di progetti principali del gruppo di cui fanno parte. Durante il tirocinio il candidato programmava delle brevi riunioni settimanali con il proprio supervisore Nico d'Heureuse, il suo collega Jan Seedorf e il manager del gruppo, Saverio Niccolini, nelle quali si faceva il punto sullo sviluppo del lavoro, ed il candidato proponeva delle scelte e dei dubbi che venivano discussi insieme, ma a parte questa supervisione le scelte principali vengono lasciate agli studenti. Questo ha reso il lavoro di ricerca molto stimolante. Il gruppo per il quale il candidato ha svolto il tirocinio, di nome RTC (Real-time Communication), diretto da Saverio Niccolini è formato da circa 10 persone, con alcune delle quali il candidato ha lavorato insieme. Al termine del tirocinio è stata consegnata al candidato una lettera di referenze, in cui il manager del gruppo valuta positivamente il lavoro svolto e ne suggerisce l'assunzione ad un ipotetico datore di lavoro.

Si segnala inoltre che dagli studi svolti ad Heidelberg è derivato un articolo scientifico [CdNS09] di prossima pubblicazione, che vede tra gli autori il candidato. L'articolo, attualmente in sottomissione alla conferenza GLOBECOM 2009[5], descrive il funzionamento del modulo con particolare attenzione all'analisi della performance.

1.2 Outline del lavoro svolto

Il lavoro svolto dal candidato durante il tirocinio si può dividere in due parti. La prima (che ha occupato i primi 3 mesi) riguarda il modulo di un server SIP per prevenire lo spam via telefono basato su Reti di fiducia o Web of Trust (WoT), la seconda (svolta nell'ultimo mese) riguarda un sistema di personalizzazione delle preferenze degli utenti per lo stesso server. Segue un elenco sommario dei compiti svolti durante il tirocinio. Tutti questi lavori sono stati portati a termine nella loro completezza dal candidato. I concetti presentati verranno ripresi nei capitoli successivi.

Prima parte - Sul modulo `wot` La prima parte del lavoro ha visto l'implementazione di un modulo per VoIP-SEAL, un sistema di prevenzio-

[5]Global Communications Conference è una delle conferenze principali organizzate da IEEE sul tema delle telecomunicazioni, a cui partecipano professionisti, aziende e università. L'edizione 2009, che avrà luogo alle Hawaii, ha come tema *Riding the wave of global connectivity*, e prevede la partecipazione di 1200 persone. link: http://www.ieee-globecom.org/2009/

ne dello SPIT (Spam over Internet Telephony). Il modulo è basato su una WoT, che è una rete di identità collegate da una relazione di fiducia. La sicurezza della rete deriva dal fatto che la relazione tra identità è garantita crittograficamente. Tra due identità può dunque esistere un percorso di fiducia (trust path), dal quale si determina quanto una identità si fidi di un'altra.

Il lavoro in questa parte ha compreso:

- La progettazione e l'implementazione di un algoritmo per trovare il trust path più breve da un'identità ad un'altra. L'algoritmo è stato implementato ponendo l'attenzione sulla performance. Il linguaggio utilizzato per l'implementazione è C.
- Implementazione della firma crittografica dei messaggi SIP e della sua verifica, utilizzando metodi di crittografia asimmetrica come OpenPGP.
- Integrazione del modulo che esegue l'algoritmo nel sistema VoIP-SEAL pre-esistente.
- Creazione di un'interfaccia grafica che mostra, durante il trattamento di un messaggio SIP, la ricerca del trust path. Questa interfaccia è implementata in Java utilizzando le librerie Awt e Swing e ha lo scopo di dimostrare il funzionamento del modulo.
- Test completi sulla performance, incluso lo studio dell'aggregazione dei dati e la loro elaborazione statistica. I test sulla performance sono mirati a mostrare la possibilità di messa in produzione di una reale applicazione del modulo. I dati sono stati raccolti ed elaborati utilizzando vari tool scritti in C.

Seconda parte - Sulla personalizzazione di VoIP-SEAL La seconda parte del lavoro si è concentrata sulla modellazione e la parziale implementazione del nucleo di un sistema in grado di fornire all'utente la possibilità di personalizzare il proprio profilo per il servizio anti-SPIT. Lo scopo di questo sistema è di permettere all'utente di definire la propria politica sul trattamento di una chiamata VoIP in arrivo attraverso un documento XML.

Le sfide della seconda parte hanno riguardato più che altro un design molto complicato ed astratto, ed è stata data un'attenzione particolare alla flessibilità, alla modellazione orientata agli oggetti e alla performance del parser XML. Sebbene anche questo lavoro abbia richiesto una dose di creatività, la presente relazione coprirà esclusivamente la prima parte del lavoro, quella principale e di interesse scientifico più rilevante.

Capitolo 2

Il problema dello Spam

Come accennato nell'introduzione, il fine del tirocinio è stato di progettare un sistema in grado di offrire un livello di sicurezza adeguato a impedire il manifestarsi di fenomeni di spam su VoIP[1], ovvero SPIT (Spam over Internet Telephony). Lo SPIT è la trasposizione del fenomeno dello spam via e-mail sulle tecnologie VoIP. Infatti, esattamente come è possibile generare messaggi non richiesti destinati ad una casella e-mail, è possibile generare delle telefonate VoIP non richieste.

Esistono moltissimi protocolli, liberi o proprietari, che permettono la comunicazione VoIP. Dato che SIP [RSC+02] è considerato il protocollo standard, è effettivamente il più utilizzato e su di esso si basa il sistema presentato in questa relazione, d'ora in poi per SPIT si intenderà lo spam convogliato tramite il protocollo SIP. Tuttavia, la maggior parte dei concetti espressi per lo spam su SIP valgono anche per gli altri protocolli di VoIP. Il protocollo SIP è presentato nell'Appendice A, dove ne vengono presentati i tratti necessari per la comprensione di questa tesi.

Il fenomeno dello SPIT non è ancora diffuso, ed esistono pochi casi documentati di telefonate di questo tipo. Questo perché le tecnologie VoIP sono molto recenti, e la loro diffusione, in particolare per l'utenza privata, è ancora molto ristretta. Tuttavia è in rapidissima crescita, e visto il minore costo economico rispetto alla telefonia tradizionale è facile ipotizzare che entro pochi anni i paesi più avanzati tecnologicamente sposteranno gran parte del traffico telefonico su Internet. Il lavoro di lotta allo SPIT è dunque preventivo rispetto al problema, ma sebbene la minaccia non sia ancora concreta, è inevitabile che col diffondersi di tecnologie VoIP si diffonderanno anche i tentativi di intrusione in questi sistemi.

Lo SPIT può avere, come vedremo, effetti ben più devastanti sulla diffusione del VoIP rispetto a quelli che lo spam ha avuto sull'e-mail. Il fenomeno dello SPIT, se non contenuto, può impedire lo sviluppo delle tecnologie VoIP

[1] Con "VoIP" si intende l'insieme dei servizi di telefonia via Internet

fino a provocarne il flop. Vi è per questo un forte interesse della comunità scientifica e dell'industria nella protezione anti-SPIT.

2.1 Lo spam via e-mail

Il fenomeno dello spam via e-mail e dello SPIT sono simili anche tecnologicamente tra loro, e dato che solo sul primo esiste una vasta letteratura, su questo si fornisce un breve excursus storico. Lo spam via e-mail si è diffuso soprattutto perché è stato in grado di sfruttare i punti deboli del protocollo SMTP, inventato nel 1971. Al momento della progettazione di questo protocollo, per ragioni storiche, gli inventori non si sono posti il problema che utenti maliziosi avrebbero potuto usarlo per convogliare messaggi non desiderati, concentrandosi piuttosto sulla costruzione di un protocollo semplice, scalabile e solido in grado di inviare messaggi a qualunque server del mondo. Basti pensare che nel protocollo SMTP non è prevista alcuna verifica sull'identità del mittente, quindi qualunque computer connesso ad un server SMTP può inviare e-mail indicando un mittente fasullo. Sebbene lo stato attuale delle tecnologie di sicurezza sarebbe in grado di creare un sistema di messaggistica in grado di limitare fortemente il fenomeno dello spam[2], tutti questi metodi non si sono affermati a causa della difficoltà di garantire la retro-compatibilità con i servizi e-mail già esistenti, clausola fondamentale per tutti i servizi Internet. Il protocollo SMTP è debole, ma di diffusione globale, e proprio la sua enorme diffusione ha nella pratica impedito di espandere le sue funzionalità con un sistema anti-spam omogeneo. Con lo SPIT si cerca di evitare la sottovalutazione del problema avvenuto con lo spam via e-mail, interagendo nello sviluppo di SIP prima che raggiunga il livello di diffusione dell'e-mail.

Il fenomeno dello spam via e-mail ha assunto proporzioni significative a partire dal 2002 quando, secondo uno studio di IDC[3], venivano inviati 2,4 miliardi di messaggi di spam al giorno, ma come si vede dal grafico 2.1 la crescita negli ultimi anni è stata esponenziale.

Vi sono diversi tipi di messaggi non desiderati. Riportiamo un breve elenco:

Spam con fini commerciali pubblicizzano siti, prodotti medici, prodotti finanziari, materiali per adulti e altro.

Phishing contengono messaggi fraudolenti, tipicamente con l'obbiettivo di ingannare l'utente in modo che invii le proprie credenziali come nome utente e password.

[2] Ad esempio, il sistema DomainKeys, definito in [ACD+07], si basa sull'autenticazione crittografica delle e-mail per certificare l'identità del server mittente

[3] IDC è un'agenzia di ricerca sui trend riguardo all'informatica e le telecomunicazioni.

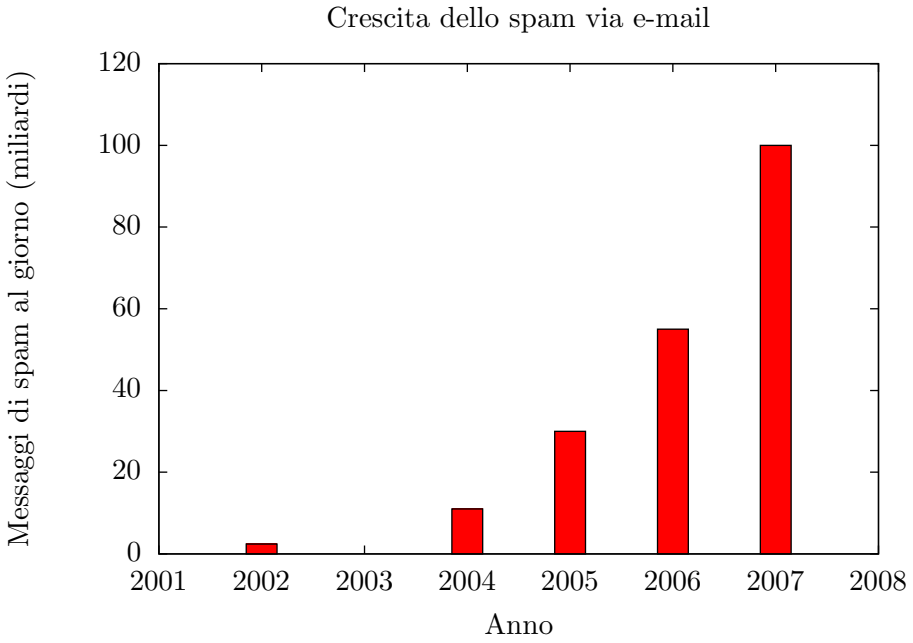


Figura 2.1: Crescita esponenziale dello spam via e-mail

Advance-fee scam contengono richieste di prestiti monetari o di investimenti in cambio di un grande guadagno.

Virus molti virus informatici si diffondono tramite e-mail. Tipicamente, dopo l'infezione di un computer, il virus invia una e-mail agli indirizzi contenuti nella rubrica del computer infetto. Ad esempio, il virus `W32.Sobig.F@mm` nel 2003 ha sfruttato questo meccanismo e altre debolezze di Windows per infettare milioni di computer.

Un problema fondamentale di chi costruisce i sistemi di spam è la costruzione di un database di indirizzi e-mail a cui inviare i messaggi. A parte l'acquisizione degli indirizzi da parte dei virus, il metodo più comune resta la ricerca degli indirizzi su web. È infatti relativamente semplice costruire un crawler, ovvero un programma che visita automaticamente e ricorsivamente le pagine web alla ricerca di pattern che possano rappresentare un indirizzo e-mail per costruire un elenco di indirizzi.

In molti si chiedono se chi fa spam di professione ottiene da questo un reale guadagno. Sicuramente la maggior parte degli utenti, di fronte ad un messaggio di spam, lo eliminano senza considerarne il contenuto. Tuttavia lo spam resta vantaggioso perché i costi di questi sistemi sono estremamente bassi, dato che ottenere indirizzi e inviare mail è praticamente gratuito. Ad esempio, per fare un confronto con altri mezzi pubblicitari, si pensi ai cataloghi inviati dai supermercati via posta ordinaria: anche in questo caso

la maggior parte degli utenti cestinano il catalogo senza leggerlo, ma i costi rispetto allo spam via e-mail sono nettamente maggiori (stampa del catalogo, distribuzione).

Lo spam ha anche delle importanti ricadute in termini economici. Secondo una stima presentata in [Rus05], nel 2004 soltanto negli Stati Uniti si sono persi 22 miliardi di dollari per il tempo sprecato ad eliminare messaggi indesiderati. È inoltre ovvio lo sbilanciamento dei costi tra la fonte dello spam e il destinatario: mentre un utente, per cancellare un messaggio di spam, deve perdere tempo a capire di cosa si tratti, per la sorgente dello spam l'invio è automatizzato e molto più veloce.

Le proprietà che mantengono basso il costo dei sistemi di spam sono:

1. Il basso costo del trasferimento di dati via Internet
2. L'alta velocità con cui i dati vengono trasferiti
3. La difficoltà nello scoprire la fonte fisica dello spam
4. La relativa semplicità con cui possono essere trovati gli indirizzi vittime dello spam
5. L'automazione del processo (durante l'invio dei messaggi non è necessario un significativo intervento umano)

Facendo una valutazione pratica, il semplice fatto che lo spam esista porterebbe a pensare che qualcuno ne ricavi del profitto.

2.2 Lo spam via VoIP

Come accennato, analogamente a quanto avviene per l'e-mail, anche con il VoIP è posibile ricevere spam sotto forma di telefonate indesiderate (SPIT). Tutte le proprietà che mantengono basso il costo dello spam via e-mail sono conservate: Internet continua a garantire costi bassi a alte velocità anche per volumi di traffico maggiori come sono quelli necessari a trasferire la voce[4], gli indirizzi dei destinatari possono essere trovati ancora più facilmente[5], e l'automazione resta possibile.

Trovare la fonte fisica dello spam resta un compito non banale, infatti, mentre nelle reti a commutazione di circuito come la PSTN il gestore di

[4]Ad esempio, si consideri l'invio da parte di una fonte di spam di messaggi vocali utilizzando il codec G.729 per la compressione audio. Il payload del messaggio voce è di 8kbit/s, a cui aggiungiamo altri 8kbit/s per gli overhead dei protocolli, per un totale di 16kbps. Da una normale ADSL domestica sarebbe dunque possibile inviare 40 telefonate contemporanee.

[5]Per retro-compatibilità con la rete PSTN, spesso come indirizzi degli account VoIP sono utilizzati dei numeri progressivi, dunque a partire da un indirizzo è possibile generarne altri validi.

telefonia conosce e tiene memoria del numero di telefono del chiamante, rendendo immediatamente rintracciabile una fonte di spam, su Internet, rete a commutazione di pacchetto, questo risulta molto più difficile, e non è scontato che i provider conservino i log delle connessioni. Questo permette allo SPIT di aggirare eventuali leggi o regolamentazioni a cui invece si devono attenere i tradizionali mezzi di telemarketing su PSTN. Scenario ancora peggiore è quello in cui i messaggi di SPIT sono propagati tramite virus che generano le telefonate, caso in cui la fonte dello SPIT sarebbe distribuita e quindi ancora più difficile da isolare.

Creare un sistema per inviare telefonate di SPIT su protocollo SIP è molto semplice. Il programma di SPIT dovrebbe solo inviare in parallelo messaggi di tipo `INVITE`, e nel momento in cui i telefoni vittime dello SPIT accettano la telefonata, può cominciare la trasmissione di un messaggio registrato, e al termine chiudere la conversazione. Un programma del genere si può implementare completamente su lato software e non ha bisogno di alcun hardware particolare – è sufficiente un personal computer con una connessione ad Internet –, inoltre non ha bisogno dell'intervento di un esperto se non durante la programmazione del software.

Tuttavia resta una fondamentale differenza tra un messaggio e-mail e una telefonata: la prima è una forma di comunicazione asincrona, la seconda è sincrona. Questo significa, per quanto riguarda i sistemi di prevenzione dello spam, che l'analisi di un messaggio è impossibile. Infatti, a differenza dell'e-mail, con il VoIP non si conosce il contenuto della telefonata in anticipo, per questo mentre la maggior parte dei filtri anti-spam per e-mail analizzano il corpo del messaggio per decidere se si tratti di spam, lo stesso approccio non può essere applicato al VoIP, in cui il corpo del messaggio – ovvero la voce della telefonata – non esiste prima della trasmissione. Una presentazione del problema, insieme con un'analisi delle differenze tra SPIT e spam via e-mail è presentata in [RJ08].

Come è facile immaginare, le telefonate indesiderate possono avere un effetto di disturbo anche peggiore rispetto alle e-mail. Quando un utente accede alle proprie e-mail ed alcune di queste sono spam, il disturbo consiste solamente nel cancellarle, ed è minore rispetto a quello provocato da una telefonata di spam, che costringe a rispondere al telefono che squilla. Per questo, se gli utenti che utilizzano le e-mail sono pronti a sopportare lo spam pur di continuare ad utilizzare un sistema globale, economico ed efficente, altrettanto non si può dire per il VoIP, tecnologia estremamente potente ma che ha un rivale, la vecchia telefonia PSTN, che resta più costosa ma quasi del tutto priva di spam. Se lo SPIT dovesse diffondersi, sia l'utenza domestica sia quella delle grandi organizzazioni deciderebbe semplicemente di tornare a PSTN, rinunciando alle potenzialità del VoIP e spendendo più soldi in cambio di un servizio affidabile.

2.3 Conclusioni

Negli ultimi anni il problema dello spam via e-mail ha raggiunto proporzioni enormi. Questo va imputato alle debolezze storiche dei protocolli usati e alla lentezza con cui vengono implementate le azioni di contrasto allo spam. Esattamente come l'e-mail e a differenza della telefonia PSTN, anche il VoIP può essere sottoposto allo spam, che in questa declinazione di chiama SPIT. Lo SPIT potrebbe affliggere gli utenti VoIP limitando la diffusione di questa tecnologia altrimenti molto potente, quindi è estremamente importante adesso - nel momento in cui questa tecnologia ha appena cominciato a diffondersi - studiare e implementare delle architetture sicure di prevenzione dello SPIT. Lo SPIT ha una fondamentale differenza dallo spam via e-mail: non è possibile fare controlli sul contenuto della telefonata prima che questa sia recapitata al destinatario.

Capitolo 3

Il sistema anti-SPIT di VoIP-SEAL

Nel febbraio 2007, al 3GSM World Congress che ha avuto luogo a Barcellona, i ricercatori di NEC Europe hanno presentato VoIP-SEAL, il sistema che fornisce una tecnologia di sicurezza per evitare lo SPIT. VoIP-SEAL è un server SIP che viene posto al perimetro di una rete da proteggere, come mostrato in Figura 3.1. Ogni chiamata proveniente da Internet e destinata a un telefono protetto dal server viene processata dal server che decide se scartarla (in caso di SPIT) oppure inoltrarla al telefono destinatario.

Si presti attenzione alla nomenclatura presentata nella Figura 3.1, che verrà ripresa negli esempi successivi: Alice è un utente che vuole telefonare a Bob. Bob e Charlie fanno parte di un'organizzazione, di nome ACME, che ha una rete protetta da VoIP-SEAL, mentre Alice è connessa a questa rete tramite Internet. Spad invece è una fonte di SPIT, anch'essa collegata tramite Internet.

VoIP-SEAL è un prototipo. Non essendo ancora diffuso lo SPIT, e non

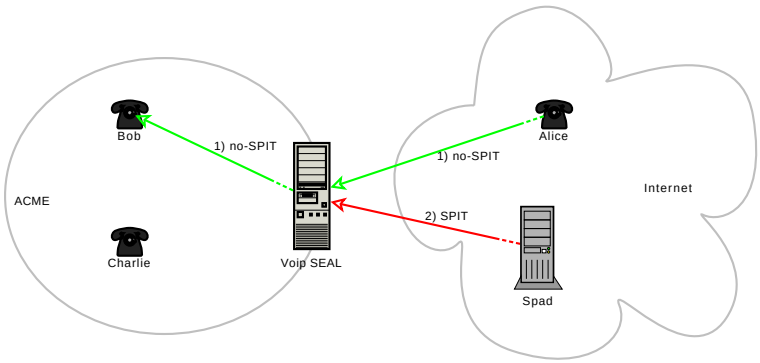


Figura 3.1: Schema di deployment di VoIP-SEAL

avendo di conseguenza un insieme definito di requisiti che questo servizio deve garantire, è stato necessario progettare un sistema molto flessibile, in grado di essere facilmente e velocemente aggiornabile con il concretizzarsi della minaccia. Per questo è stata scelta una struttura del software modulare: il server esegue vari test, ed ogni test è svolto da un modulo.

Sebbene il fenomeno dello SPIT non sia definito nel dettaglio, alcuni principi chiave di un sistema anti-SPIT sono chiari:

1. Deve minimizzare o ridurre a zero il numero di telefonate legittime scartate.

2. Deve fare in modo che il telefono destinatario squilli solamente se è molto alta la probabilità che il messaggio non sia SPIT, altrimenti, anche se il destinatario aggancerà subito il telefono, verrà comunque disturbato.

3. È preferibile appesantire il sistema anti-SPIT con elaborati calcoli che non richiedono l'interazione dell'utente piuttosto che chiedere un'interazione umana anche minima.

4. Nel caso di test intrusivi che richiedono l'interazione dell'utente, è preferibile sbilanciare questi test verso il chiamante piuttosto che verso il chiamato.

5. I controlli devono avvenire in tempo reale, quindi devono essere eseguiti in un tempo accettabile per l'utente.

L'articolo fondamentale in cui si descrive il funzionamento di VoIP-SEAL è [QNTS08].

3.1 I test di VoIP-SEAL

Tenendo conto di questi principi, i test di VoIP-SEAL sono raggruppati in 5 stage, che vengono applicati sequenzialmente dallo stage 1 allo stage 5. Ogni test, dopo aver analizzato la chiamata in arrivo, restituisce un punteggio $s \in \mathbb{R} \mid -1 \leq s \leq 1$. Il punteggio indica il giudizio fornito dal test sulla probabilità che la chiamata sia SPIT: $s = 1$ indica la certezza che si tratti di spam, $s = -1$ la certezza che la chiamata sia legittima, $s = 0$ indica che il test non ha elementi per dare giudizi sulla probabilità che si tratti di SPIT. Valori intermedi di s rappresentano proporzionali probabilità che la chiamata sia SPIT. Man mano che si procede con gli stage, i test diventano sempre più intrusivi, per questo i test dello stage n vengono eseguiti solamente se i punteggi forniti dai test dello stage $n-1$ non danno indicazioni abbastanza precise sulla legittimità della chiamata. Sono previsti 5 stage:

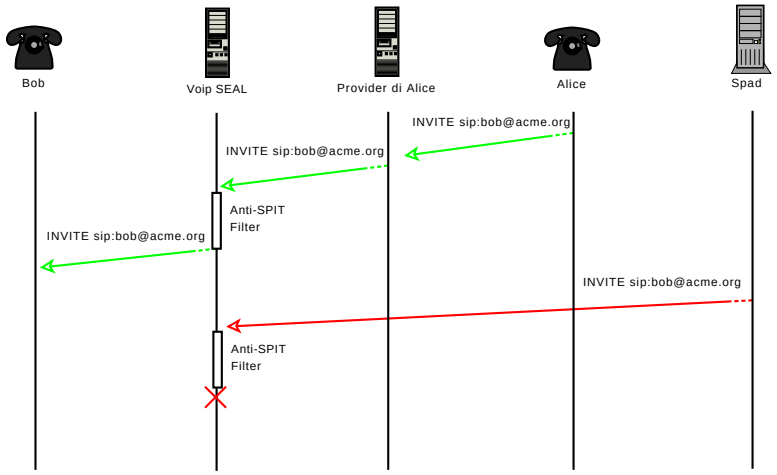


Figura 3.2: Scambio di messaggi SIP legittimi e illegittimi

Stage 1 i test sono non-intrusivi, ovvero non hanno bisogno dell'interazione umana e sono trasparenti sia al chiamante che al chiamato. Il modulo WoT, argomento della presente relazione, si colloca in questo stage.

Stage 2 i test hanno bisogno dell'interazione della persona chiamante, che deve dimostrare la legittimità della telefonata. Rientrano in questo stage i Turing test.

Stage 3 i test hanno bisogno dell'interazione della persona chiamata, ma hanno luogo prima che la telefonata venga stabilita.

Stage 4 i test hanno luogo durante la telefonata.

Stage 5 viene valutato il feedback dato dalla persona chiamata per potenziare la knowledge base del sistema.

A livello di protocollo SIP, quello che avviene durante l'analisi di una telefonata in arrivo è illustrato in Figura 3.2[1]: Alice invia al proprio provider VoIP un messaggio di `INVITE` destinato a Bob. Questo provider lo inoltra a VoIP-SEAL, implementato come un server SIP, che lo analizza, lo riconosce come telefonata legittima, e lo inoltra a Bob. Quando invece Spad invia il messaggio di `INVITE` a VoIP-SEAL, questo lo scarta senza inoltrarlo a Bob. E' importante notare come il server rispetti completamente gli standard del protocollo SIP, infatti nella RFC [RSC+02] che definisce SIP è previsto il forwarding dei messaggi `INVITE` da parte dei proxy verso proxy successivi o verso gli utenti.

[1]Nella figura sono riportati, per chiarezza di esposizione, esclusivamente i messaggi `INVITE`

Si riportano di seguito alcuni dei test che tra il 2007 e il 2008 sono stati implementati da NEC nel prototipo di VoIP-SEAL, e per ognuno di essi vengono presentati caratteristiche e difetti. L'elenco di questi test non è completo, è infatti riportato per ogni stage un solo test, limitando la descrizione a quelli più interessanti o caratteristici. Per un trattamento completo si faccia riferimento a [QNTS08]. Il sistema funziona con una knowledge base che può venire aggiornata in seguito ad ogni telefonata per potenziare il filtro anti-SPIT. Per questo esistono alcuni test (quelli degli stage 4 e 5) che non hanno il fine di prevenire lo SPIT della telefonata corrente ma solamente di aggiornare la knowledge base, e prevenire telefonate di SPIT future. Si veda, a proposito, l'esempio riportato per lo Stage 5.

Liste (Stage 1) Si colloca nello Stage 1, la fase in cui non è prevista l'interazione umana, un test molto semplice per le chiamate in arrivo. Ogni utente protetto da VoIP-SEAL ha due liste di indirizzi SIP, la Whiteliste e la Blacklist. Gli indirizzi della Whitelist sono gli indirizzi fidati: tutte le telefonate provenienti da questi indirizzi vengono accettate automaticamente. Simmetricamente, tutte le chiamate provenienti da indirizzi contenuti nella Blacklist sono considerate SPIT e scartate. Le blacklist possono essere integrate con altre blacklist gestite globalmente.

Questo test è non intrusivo, è computazionalmente facile ed è semplice da implementare. Il difetto maggiore è che, per aggirare il sistema, alla fonte di SPIT sarebbe sufficiente generare casualmente gli indirizzi mittenti in modo che non rientrino nella Blacklist[2].

Turing Test (Stage 2) Il test teorizzato da Alan Turing nel 1950 serve per scoprire, tramite un dialogo, se una delle parti coinvolte nella comunicazione è un umano oppure una macchina. Questo test può essere declinato per lo SPIT col fine di scoprire se la telefonata provenga da una fonte di SPIT oppure da un umano. Alcuni esempi di Turing Test per lo SPIT sono:

- Quando si riceve una chiamata, si risponde con un messaggio vocale in cui si chiede al chiamante di comporre una sequenza di numeri sulla tastiera, oppure delle semplici operazioni matematiche. Basandosi sull'ipotesi che la fonte di SPIT non sia in grado di elaborare questo messaggio, solamente gli umani potrebbero comporre la giusta combinazione di tasti. Il problema di questo test è che potrebbe portare a problemi di usabilità, infatti alcuni utenti potrebbero avere difficoltà nel comporre la sequenza di

[2]Infatti anche in SIP non è previsto un sistema di autenticazione degli utenti da parte del destinatario di una chiamata, per cui è sempre possibile generare telefonate da indirizzi inesistenti.

numeri richiesta (ad esempio, per la lingua del messaggio inviato dal test).

- Quando si riceve una chiamata, si risponde con un messaggio registrato che invita il chiamante ad attendere qualche secondo prima che la telefonata venga inoltrata. Ma mentre un umano resterebbe in silenzio e comincerebbe a parlare solamente dopo questo messaggio, una fonte di SPIT, ricevuto il messaggio di inizio della telefonata, non sapendo distinguere il messaggio registrato dalla risposta di un umano, comincerebbe subito l'invio del messaggio di SPIT. Sarebbe dunque necessario al sistema anti-SPIT un meccanismo di rilevazione della voce che faccia fallire il test nel caso che il chiamante parli prima della fine del messaggio di attesa. Un problema di questo test è che la rilevazione della voce ha un costo computazionale che ricadrebbe sul sistema anti-SPIT. Inoltre il test di rilevazione della voce porterebbe probabilmente a molti falsi negativi (si pensi alla possibilità che il chiamante sia in un ambiente rumoroso, e che il test confonda il rumore di fondo con un messaggio di SPIT).

Comunicazioni basate sul consenso (Stage 3) Questo test prevede che il destinatario della chiamata debba accettare esplicitamente una chiamata in ingresso. In seguito all'accettazione di una chiamata, l'indirizzo mittente può essere inserito nella Whitelist in modo che l'accettazione non debba essere ripetuta per una chiamata successiva. Il test si basa sull'ipotesi che, dato un destinatario di una chiamata, il numero dei possibili mittenti sia basso, assunzione vera per quanto riguarda l'utenza domestica, in cui normalmente si ricevono telefonate da un piccolo insieme di persone, ma che non si può applicare ad esempio ad un call center o un ufficio pubblico, che ricevono la maggior parte delle telefonate da utenti che chiamano per la prima volta. In questo caso, questo test risulterebbe del tutto inutile.

Filtro sui contenuti (Stage 4) Durante una telefonata, un test può cercare dei pattern nel messaggio in arrivo per verificare che si tratti di spam. Questi filtri hanno moltissimi problemi: è difficile, computazionalmente costoso e porta a falsi negativi la decodifica della voce ed il processo di ricerca di schemi di SPIT all'interno del messaggio; inoltre il test interviene nel momento in cui l'utente destinatario è già stato disturbato nel caso di SPIT, e in questo caso sarebbe più ragionevole chiedere all'utente una segnalazione esplicita di SPIT al termine della telefonata.

Sistema di reputazione (Stage 5) Al termine della telefonata, la persona chiamata pubblica in un servizio centralizzato un feedback positivo

o negativo per il mittente. In questo modo si crea un sistema di reputazione in cui ogni identità ha un punteggio che può essere usato in un test dello Stage 1. Un problema di questo test è che un nuovo utente privo di reputazione si vedrebbe penalizzato rispetto a utenti vecchi, inoltre resta il problema di creare un'associazione solida tra un indirizzo SIP ed un'identità reale: nulla infatti vieta, in SIP, che la fonte di SPIT sfrutti un indirizzo con alta reputazione per inviare telefonate di SPIT aggirando i controlli.

Prima di concludere la presentazione di VoIP-SEAL, si porta l'attenzione sul fatto che per funzionare, nessuno di questi test ha bisogno di terminali VoIP particolari, e tutto ciò che serve a questi test può essere dislocato sul server anti-SPIT, che mantiene anche la knowledge base. Ad esempio, per il test del sistema di reputazione (Stage 5), per dare all'utente la possibilità di fornire un feedback su una chiamata appena terminata, il sistema anti-SPIT può inviare all'utente un breve messaggio vocale in cui si chiede di premere un tasto per segnalare che la chiamata appena terminata è SPIT e un altro tasto per segnalare che è legittima; allo stesso modo nel caso della Whitelist (Stage 1), l'elenco degli indirizzi può risiedere sul server. Questa proprietà è fondamentale perché solo la garanzia della retro-compatibilità con i terminali VoIP già in circolazione può permettere ad un sistema anti-SPIT di diffondersi.

3.2 Conclusioni

NEC sta sviluppando un sistema anti-SPIT di nome VoIP-SEAL. Questo sistema è integrato nell'architettura SIP come un Proxy Server. L'obbiettivo di un server anti-SPIT è di filtrare le telefonate di SPIT e ammettere quelle legittime. Nel fare questo, deve rispettare i requisiti di tempo reale e interagire meno possibile con gli utenti umani. L'ideale sarebbe dunque un sistema completamente informatizzato in grado di distinguere tra telefonate legittime e lo SPIT.

Capitolo 4

La Web of Trust contro lo SPIT

La Web of Trust (rete di fiducia, d'ora in poi WoT) è un database che contiene delle identità e i collegamenti di fiducia tra queste identità. In altri termini, si tratta di un grafo orientato i cui nodi rappresentano un'identità e gli archi una relazione di tipo *"x ha fiducia nell'identità di y"*. In questo capitolo verrà illustrato come è possibile utilizzare una WoT per prevenire lo SPIT.

4.1 La WoT di OpenPGP

Il concetto di WoT nasce con OpenPGP, un sistema di autenticazione e crittaggio di messaggi[1]. OpenPGP, definito in [CDF+07], prevede che ogni utente Bob abbia una chiave pubblica, da distribuire, e una privata, da mantenere segreta. Quando un altro utente Alice critta un messaggio utilizzando la chiave pubblica di Bob, solamente chi possiede la chiave privata di Bob, ovvero soltanto Bob, potrà decrittare il messaggio. Inoltre OpenPGP permette ad Alice di autenticare il messaggio allegando ad esso una *firma*, ovvero un codice generato con la chiave privata di Alice e con il quale Bob, utilizzando la chiave pubblica di Alice, potrà avere la certezza che Alice sia effettivamente il mittente del messaggio.

Questo sistema di autenticazione non avrebbe bisogno di una WoT se Alice si fidasse direttamente di Bob, ovvero se i due utenti si fossero scambiati attraverso un mezzo sicuro le proprie chiavi pubbliche[2] e Alice avesse

[1] OpenPGP tipicamente è utilizzato per lo scambio di e-mail, ma ha applicazioni anche su altri tipi di messaggi.

[2] Tipicamente, l'unico mezzo sicuro è lo scambio delle chiavi pubbliche vis-à-vis; per questo vengono organizzati dei Key Signing Party, in cui gli utenti di OpenPGP si incontrano e, verificando l'identità degli altri utenti, firmano le loro chiavi.

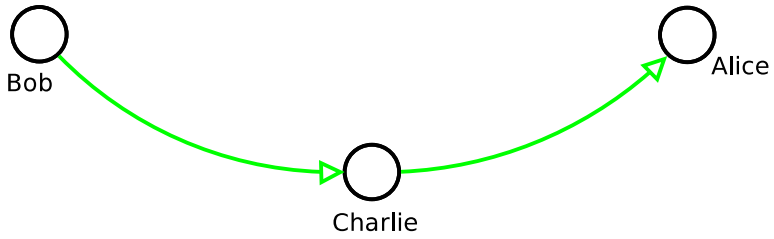


Figura 4.1: Un esempio di WoT: Alice e Bob non si fidano direttamente l'uno dell'altra, ma esiste comunque un percorso di fiducia da Bob ad Alice.

firmato[3] quella di Bob.

Tuttavia esistono dei casi in cui può essere utile la WoT. Supponiamo ad esempio che Alice e Bob non si siano mai incontrati, e per questo nessuno dei due abbia mai firmato la chiave dell'altro, ma che vi sia una terza persona, Charlie, di cui Bob ha firmato la chiave e che ha firmato la chiave di Alice. La situazione è schematizzata nel grafo in Figura 4.1. In questo modo, nella WoT esiste un *Trust Path* (percorso di fiducia, d'ora in poi TP) da Bob ad Alice: Bob si fida di Charlie, e Charlie si fida di Alice, quindi si può dire che *Bob si fida di Alice in 2 passi di fiducia*, ovvero che con ragionevole certezza Bob può verificare l'autenticità dei messaggi provenienti da Alice anche se nessuno dei due ha mai verificato direttamente l'identità dell'altro. È fondamentale notare che non è vero il viceversa, infatti non esiste un percorso di fiducia da Alice a Bob (ovvero Alice non può fidarsi dei messaggi firmati da Bob), la relazione di fiducia non è dunque sempre simmetrica ma, come in questo esempio, può essere orientata: Bob si fida di Alice ma Alice non si fida di Bob.

4.2 Un filtro anti-SPIT basato su WoT

Come è stato esposto nel Capitolo 3, i test meno invasivi e più efficaci sono svolti nello Stage 1, che non richiede in alcun modo l'interazione dell'utente. Ovviamente questi test devono essere applicati prima che la telefonata cominci, quindi analizzando solamente il messaggio SIP di tipo `INVITE`. La Whitelist è uno di questi metodi, ma ha il difetto che, essendo molto semplice come filtro, in molti casi fornisce un risultato ambiguo e il controllo sul chiamante è debole.

Per questo risulta utile costruire un test anti-SPIT basato su WoT. Avendo una WoT in cui i nodi sono indirizzi SIP e le relazioni di fiducia sono garanzie sul fatto che un'identità non è fonte di SPIT, il server anti-SPIT che protegge il destinatario della chiamata può cercare, dopo aver verificato

[3]Per *Alice firma la chiave di Bob* si intende che Alice, dopo aver verificato la corrispondenza tra Bob e la sua chiave, dichiara di avere fiducia in questa chiave.

l'autenticità di un messaggio rispetto al mittente, un percorso di fiducia dal destinatario al mittente, e decidere in base alla lunghezza di questo percorso di fiducia se e quanto fidarsi della telefonata sotto analisi.

Continuando il confronto con la Whitelist, risulta evidente come un sistema di fiducia dinamico come quello basato su WoT sia più potente rispetto ad un elenco statico di indirizzi. Posto che di media ogni utente dia la propria fiducia ad altri m utenti *nuovi*, cioè in precedenza non appartenenti alla WoT, in modo da inserirli nella rete, e detto l il numero di passi di un dato TP, il TP raggiunge di media $r = m^l$ utenti. Ponendo $m = 100$ (ogni utente si fida di altri 100 utenti) e $l = 5$ (il TP ha lunghezza 5), ogni utente raggiungerebbe $r_5 = 100^5 = 10^{10}$ utenti, ovvero più degli abitanti della terra. Inoltre, la verifica della firma fa sì che sia praticamente impossibile falsificare un'identità mandando un messaggio a suo nome, vulnerabilità a cui invece è soggetto il test della Whitelist.

Come visto, il numero di identità raggiungibili tramite TP è esponenziale rispetto alla lunghezza del TP, ma questi percorsi di fiducia hanno valori diversi, a seconda della lunghezza del TP. Ovviamente non è sufficiente, perché il test del WoT abbia successo, che esista un TP da un'identità ad un'altra, dato che in pochi passi di fiducia è possibile raggiungere tutte le identità di una rete. Al contrario, per stabilire il livello di fiducia, conta il numero di passi del percorso minimo da una identità all'altra[4]. Per il test WoT è necessario stabilire una lunghezza massima del path oltre alla quale il livello di fiducia è 0. Considerando che, secondo la teoria dello Small World[5], ogni persona può raggiungere mediamente in 6 passi di amicizia qualunque persona sulla terra, la lunghezza massima può essere fissata a 3 o 4 passi, a seconda delle caratteristiche della rete.

Il test del WoT può dare solo risultati positivi, infatti può scongiurare i sospetti che una telefonata in arrivo sia SPIT verificando che il TP sia sufficientemente corto. È però impossibile fondare dei sospetti sulla possibilità che, al contrario, la chiamata sia SPIT: se una chiamata in arrivo non è firmata, oppure la chiave non è presente nel WoT, oppure la chiave è presente ma il TP è troppo lungo, il test WoT non può che dare un risultato di incertezza. Il test WoT deve dunque essere combinato con altri test, ad esempio si può configurare il server anti-SPIT in modo che venga eseguito prima il test WoT; in caso che il test abbia successo (ovvero nel caso che esista un TP abbastanza corto) la chiamata viene accettata, altrimenti viene

[4]Un test più accurato potrebbe tenere conto del peso di ogni arco, ovvero "*quanto* X si fida di Y", oppure, dato un percorso minimo tra X e Y lungo n, "quanti altri percorsi minimi, ovvero di lunghezza n, vi sono tra X e Y", ma queste possibilità non sono state considerate.

[5]La teoria è stata proposta da Stanley Milgram, un sociologo che analizzando le reti sociali ha condotto una serie di esperimenti che hanno portato alla formazione della teoria dei 6 gradi di separazione.

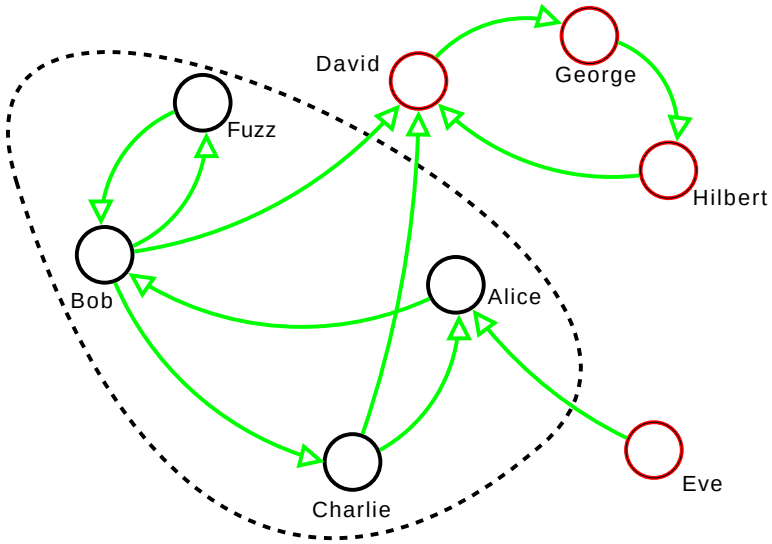


Figura 4.2: Un esempio di WoT. È messo in evidenza dalla linea tratteggiata uno Strongly Connected Set. Le identità rosse sono quelle escluse dallo SCS.

eseguito il Turing test, più intrusivo ma in grado di bloccare le telefonate di SPIT.

4.3 Lo Strongly Connected Set

All'interno di una WoT, a partire da una identità X, è possibile definire uno Strongly Connected Set (SCS), come il sottoinsieme più grande di WoT che comprende l'identità X più tutte le identità della WoT che hanno la proprietà di avere un percorso di fiducia da e verso tutte le altre identità dello SCS. Un esempio è riportato in Figura 4.2, in cui è messo in evidenza lo SCS generato a partire dall'identità Alice, da cui sono esclusi Eve (perché nessuna identità dello SCS ha un Trust Path verso di lei) e David (perché non esiste un TP da lui verso lo SCS), e di conseguenza George e Hilbert, che hanno relazioni di fiducia solo con David. È facile dimostrare che, scegliendo una identità di partenza diversa da Alice ma interna allo SCS, ad esempio Bob, si sarebbe ottenuto lo stesso SCS.

Alice e Bob hanno relazioni di fiducia asimmetriche: il cammino più breve da A a B è lungo 1, mentre il cammino opposto è lungo 2. Questo significa che *Alice si fida di Bob più di quanto Bob si fidi di Alice*. Con riferimento al sistema anti-SPIT, dato che è il mittente della chiamata a dover dimostrare la propria legittimità, nel caso di una chiamata da X a Y il test deve cercare le lunghezza del TP in direzione opposta alla chiamata, ovvero da Y a X, deve infatti determinare *quanto* Y *si fidi di* X. Nell'esempio di

Figura 4.2, questo significa che una chiamata proveniente da Alice a Bob avrà un punteggio maggiore (ovvero da parte di Bob sarà considerata più probabilmente SPIT) rispetto ad una chiamata da Bob ad Alice.

Nell'implementazione, il test WoT, nella ricerca dei percorsi dall'identità destinataria della chiamata all'identità mittente, considera solo le identità che fanno parte di uno SCS. Questo da una parte introduce una limitazione, dall'altra ha il vantaggio di diminuire consistentemente le dimensioni della WoT. L'esclusione dalla ricerca di una identità X che non ha percorsi di fiducia dallo SCS a X (come è il caso di Eve in Figura 4.2) non costituisce un problema: tutte le ricerche con il nodo sorgente $n_{src} \in$ SCS darebbero comunque esito vano, dato che nessuna identità dello SCS ha fiducia in X. Quindi l'eliminazione di queste identità riduce la dimensione della WoT senza introdurre semplificazioni. Al contrario, l'esclusione di una identità Y che non ha percorsi di fiducia in senso inverso, ovvero da Y allo SCS (caso di David), costituisce una approssimazione. A rigor di logica, infatti, Bob dovrebbe accettare le chiamate provenineti da David, di cui si fida direttamente. Questa situazione è comunque molto rara, e risolvibile facendo in modo che David dichiari la propria fiducia verso una qualunque identità dello SCS, operazione che farebbe diventare anch'esso parte dello SCS.

In un'unica Web of Trust possono esistere diversi Strongly Connected Set, fatto che pone il problema della scelta di quale degli SCS utilizzare. Ovviamente, detti SCS_1 e SCS_2 due SCS di una WoT, gli insiemi sono distinti[6]. Nell'esempio di Figura 4.2 esiste uno SCS diverso da quello evidenziato (chiamiamo questo SCS_{evid}), quello formato dalle identità David-George-Hilbert (chiamiamo questo SCS_{DGH}). Il modulo WoT prende in considerazione solo lo SCS che contiene più identità, nell'esempio SCS_{evid}, scelta che esclude per intero le identità di SCS_{DGH}. Tuttavia, è sufficiente un collegamento di fiducia da SCS_{DGH} ad un'identità di SCS_{evid} per fare in modo che le identità di SCS_{DGH} diventino parte dello SCS più grande.

4.4 Deployment del test WoT (W-BASA)

Si presenta in questa sezione un possibile schema di deployment dell'infrastruttura necessaria al test WoT. Lo schema presentato in seguito non è l'unico possibile, ma è realistico. L'architettura in questione si chiama W-BASA (Wot-Based Anti-Spit Architecture).

Il test della WoT, come già detto, viene eseguito nello Stage 1 di VoIP-SEAL, ovvero nel momento in cui al server anti-SPIT arriva la richiesta di

[6]Si dimostra facilmente per assurdo: se vi fosse un'identità X appartenente sia a SCS_1 che a SCS_2, significherebbe che tutte le identità dello SCS_1 avrebbero un TP (passante per X) da e verso tutte le identità dello SCS_2, il che contraddice la definizione di SCS per SCS_1, dato che esso conterrebbe solo alcune delle identità che hanno percorsi di fiducia da e verso le identità di SCS_1 e non sarebbe dunque massimo.

una nuova telefonata, prima che questa telefonata sia inoltrata al destinatario e il suo telefono cominci a squillare. Questa richiesta per una nuova telefonata è un messaggio SIP di tipo `INVITE`. Naturalmente, VoIP-SEAL è un elemento di W-BASA.

Assumendo che sia il terminale VoIP mittente che destinatario non abbiano funzioni particolari (requisito necessario per la retro-compatibilità), è evidente che il processo di autenticazione del messaggio e il processo di verifica debba essere svolto da altre entità dell'architettura, non potendo essere delegato ai terminali. Per questo è necessario la presenza in W-BASA di un Signing Proxy (SP), ovvero un altro server a cui il terminale mittente invia il messaggio e che ha il compito di apporvi la firma. Ovviamente, il SP deve essere certo dell'identità del mittente della chiamata. Per questo si può ipotizzare che ogni organizzazione debba avere un proprio SP e firmi esclusivamente le richieste dei terminali VoIP dell'organizzazione. Per una discussione più approfondita sul funzionamento del SP si faccia riferimento al Capitolo 5.

Dopo essere stato firmato, il messaggio di `INVITE` viene spedito tramite Internet a VoIP-SEAL, sul quale viene eseguito il test WoT. Questo test comprende una fase di verifica della firma, e nel caso che la firma risulti valida, una ricerca nella WoT del TP dal destinatario della telefonata verso il mittente. Nel caso che il test abbia esito positivo, il messaggio `INVITE` viene inoltrato al terminale destinatario, altrimenti, in base alla configurazione del server, possono essere svolti altri test.

Il test della WoT ha bisogno del database della WoT, ovvero il grafo che comprende le identità appartenenti allo SCS più grande e le loro relazioni di fiducia. La generazione di questo grafo non è un problema banale. Assumendo di utilizzare per il sistema anti-SPIT basato su WoT un'architettura analoga ma distinta da quella usata in OpenPGP, è necessario aggiungere a W-BASA dei keyserver globali, che hanno un database sincronizzato di tutte le chiavi pubbliche OpenPGP degli utenti VoIP e le loro relazioni di fiducia. Su uno di questi keyserver deve essere eseguito, con cadenza regolare (assumiamo giornaliera) un algoritmo che, a partire dalle informazioni delle chiavi, generi il grafo dello SCS. Ogni server VoIP-SEAL dovrebbe dunque scaricare in locale una copia di questo grafo, in modo da potervi fare delle ricerche. Inoltre, se VoIP-SEAL riceve per la prima volta una chiamata da un particolare mittente, deve scaricare da uno dei keyserver la chiave pubblica del mittente, in modo da poter verificare la firma.

La Figura 4.3 mostra lo schema di deployment in un esempio. Bob e Charlie fanno parte dell'organizzazione ACME, Alice fa parte dello Xavier Institute. Come al solito, Alice vuole telefonare a Bob. Sia la ACME che lo Xavier Institute utilizzano un'architettura W-BASA per prevenire lo SPIT. Il terminale di Alice comincia la chiamata verso Bob, e la indirizza al proprio provider VoIP, ovvero il SP della propria organizzazione. Il SP ha la certezza che si tratti di Alice, grazie ai meccanismi illustrati nella Sezione 5.5. Il SP

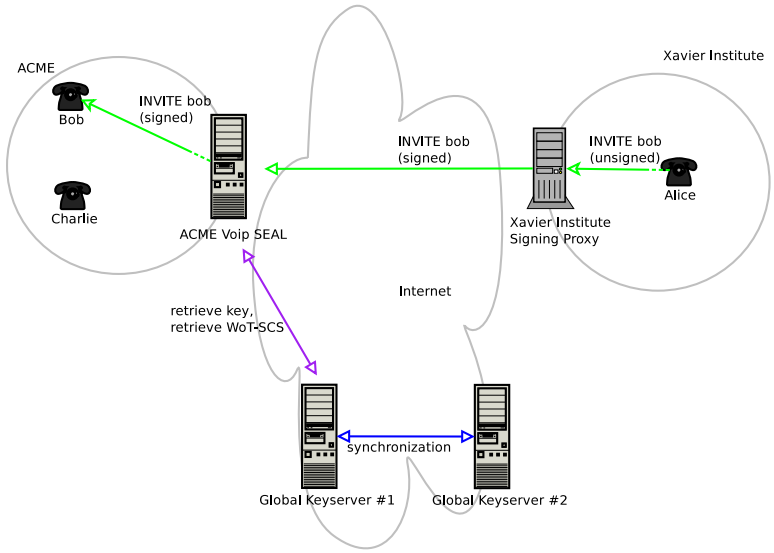


Figura 4.3: Un possibile schema di deployment per W-BASA, l'infrastruttura necessaria al test WoT.

dello Xavier Institute possiede tutte le chiavi private dei propri utenti, tra cui quella di Alice. Con questa chiave può generare la firma del messaggio, apporla al messaggio e inoltrarlo verso Bob. Quando il messaggio `INVITE` firmato arriva al provider di Bob, ovvero il VoIP-SEAL di ACME, questo svolge il test WoT, che comprende due fasi:

1. Viene verificata l'autenticità del messaggio, operazione che ha bisogno della chiave pubblica di Alice, che viene scaricata da un keyserver (a meno che non se ne abbia una copia in cache)

2. Viene cercato nello SCS della WoT, precendentemente scaricato da uno dei keyserver che lo ha calcolato, un TP da Bob verso Alice. A seconda della lunghezza del TP, VoIP-SEAL decide se inoltrare subito la telefonata a Bob oppure eseguire ulteriori test.

4.5 Aspetti di solidità e sicurezza del sistema

Il fatto che tutte le chiavi private degli utenti di un'organizzazione (come lo Xavier Institute) debbano risiedere nel SP dell'organizzazione può sembrare una degradazione del livello di sicurezza, infatti una delle prerogative di OpenPGP è che le chiavi private siano salvate esclusivamente negli endpoint, in modo che nessuno, a parte i loro proprietari, le possa utilizzare. In realtà, per quanto riguarda il sistema anti-SPIT, centralizzare le chiavi in un server relativamente sicuro (come è il SP della propria organizzazione) non

rappresenta un rischio per la privacy o per la sicurezza dei dati, visto che l'autenticazione dei messaggi SIP è utilizzata solo per il sistema anti-SPIT e non per garantire la privacy degli utenti. Si pensi a proposito al peggiore dei casi: se le chiavi private venissero rubate dal SP, una fonte di SPIT potrebbe utilizzarle per inviare messaggi di SPIT. I danni resterebbero in ogni caso limitati al sistema anti-SPIT. Lo scenario è tuttavia improbabile, infatti il furto delle chiavi non sarebbe un'operazione semplice, e andrebbe contro l'ipotesi che lo SPIT, perché convenga, debba avere un costo molto basso. Lo scenario si concluderebbe comunque con la revoca[7] delle chiavi rubate, meccanismo contemplato da OpenPGP. Tuttavia è fondamentale tener presente che le chiavi utilizzate per il sistema anti-SPIT non devono essere utilizzate per nessun altro scopo, in quanto non sarebbero sicure proprio perché conservate in un server remoto rispetto all'utente.

4.6 Costruzione della WoT anti-SPIT

Non si è trattato, per ora, il problema della formazione della WoT, ovvero di come funziona il processo per cui gli utenti VoIP possano dichiarare la propria fiducia nei confronti di altri utenti VoIP.

Attualmente, lo SCS della WoT di OpenPGP contiene 41.294 chiavi e 414.424 firme[8] (quindi di media ogni chiave ha firmato altre 10 chiavi). Questo potrebbe suggerire di utilizzare questo database già esistente per il sistema anti-SPIT, utilizzando come account SIP gli stessi indirizzi e-mail contenuti nei database di OpenPGP. Tuttavia questo pone un problema semantico non superabile, infatti nel tradizionale OpenPGP il significato delle firme è l'attestazione da parte di un'identità X dell'identità Y, ovvero della corrispondenza tra la persona fisica Y e la sua chiave PGP (col fine di permettere ad altri utenti di stabilire la credibilità di Y che non si è mai verificata personalmente), mentre nel caso del sistema anti-SPIT la firma esprime la fiducia di un'identità X sul fatto che un'altra identità Y non sia fonte di SPIT (col fine di permettere ad altri utenti di stabilire che Y non sia fonte di SPIT). Confondere il significato delle relazioni di fiducia tra indirizzi mail e indirizzi SIP porterebbe dunque a dei problemi.

Il fatto che la rete OpenPGP non sia utilizzabile come database nell'applicazione reale non impedisce che questa possa essere utilizzata nel prototipo del test: in effetti, non esistendo una WoT per un sistema anti-SPIT, per l'analisi della performance che verrà discussa nel Capitolo 6 è stata utilizzata la WoT di OpenPGP, che offre le caratteristiche di verosimiglianza, essendo una WoT reale.

Fare in modo che gli utenti SIP possano esprimere la propria fiducia

[7] Per *revoca delle chiavi* si intende l'operazione che può fare il proprietario di una chiave per revocarne la validità, in modo che nessun altro utilizzatore di OpenPGP la usi.

[8] Dato del 10 maggio 2009.

verso altri utenti[9] vede il proprio limite nella necessità da parte del sistema di permetterne l'utilizzo senza hardware particolare. Se non vi fosse questo vincolo, la soluzione più semplice sarebbe porre su ogni terminale 2 bottoni con i quali segnalare una telefonata SPIT oppure non-SPIT. In alternativa, si può pensare ad un'interfaccia (ad esempio web) tra l'utente e VoIP-SEAL, che permetta all'utente di dare la propria fiducia. Questo meccanismo sarebbe troppo dispendioso in termini di tempo per l'utente, e finirebbe per essere inutilizzato. Una buona soluzione sarebbe di impostare su VoIP-SEAL una risposta automatica ad una situazione: considerando che le telefonate di SPIT normalmente vengono interrotte dal destinatario dopo pochi secondi (appena ci si rende conto che la telefonata è un messaggio registrato), con un'approssimazione si potrebbero considerare come non-SPIT tutte le telefonate più lunghe di un tempo fisso, ad esempio 3 minuti. In questo modo la firma delle chiavi sarebbe trasparente all'utente e completamente gestito da VoIP-SEAL. A livello implementativo non sarebbe molto complesso, visto che VoIP-SEAL ha la visione di tutto il traffico VoIP tra Internet e la rete protetta, e può facilmente calcolare i tempi di una conversazione osservando i messaggi SIP di inizio e fine delle telefonate.

In fine, si fa cenno ad alcuni problemi aperti legati alla Privacy posti da un sistema architettato in questo modo. Il difetto principale è dovuto al fatto che gli algoritmi di ricerca di TP all'interno della WoT hanno bisogno di una visione globale della rete. Questo significa che i rapporti di fiducia tra le identità devono essere resi pubblici. Del resto, questo succede per la WoT di OpenPGP, in cui tutte le relazioni di fiducia sono pubblicate. Va considerato tuttavia che nel caso della WoT di OpenPGP è pubblicata solo la fiducia nell'identità, informazione che potrebbe essere considerata non personale, inoltre la pubblicazione deve avvenire in maniera esplicita con il consenso dell'utente. Potrebbero nascere più problemi nel caso della soluzione illustrata nel paragrafo precedente, in cui le relazioni di fiducia vengono determinare in base alle telefonate fatte, e di conseguenza con la WoT verrebbe pubblicato di fatto il tabulato delle telefonate, ovvero i legami tra le persone.

4.7 Conclusioni

La WoT, concetto mutuato dal sistema di crittografia OpenPGP, è una rete di fiducia garantita crittograficamente. Questa rete può essere utilizzata per prevenire lo SPIT, verificando alcune proprietà del chiamante. Il fatto che il chiamante faccia parte della WoT non garantisce che da lui non possa provenire lo SPIT, dunque è necessario, per ogni telefonata in arrivo, cercare

[9] Questo problema non è stato affrontato nel tirocinio argomento di questa relazione, verrà presentato qui per sommi capi.

il TP dal chiamato al chiamante, in modo da determinare il livello di fiducia dal primo verso il secondo.

È stata presentata W-BASA, un'architettura di supporto al test della WoT. Un'architettura del genere è necessaria per mantenere la retrocompatibilità con i terminali VoIP. Questa architettura comprende i keyserver, in cui sono salvate le chiavi pubbliche degli utenti e le loro relazioni di fiducia, e i SP, che mantengono le chiavi private e le usano per firmare i messaggi in uscita verso Internet. Inoltre, prima di apporre la firma, il SP deve essere in grado di accertarsi che un messaggio a lui recapitato provenga effettivamente dall mittente indicato nel messaggio.

Capitolo 5

Implementazione del modulo `wot`

Il server VoIP-SEAL si divide in due parti. La prima parte, il Core, si occupa di creare l'ambiente di esecuzione, gestisce l'invio e la ricezione dei messaggi SIP, carica la configurazione dei moduli e i moduli stessi, sceglie come eseguire i test dei moduli e come usare i risultati forniti dai test. La seconda parte è formata dai moduli, oguno dei quali implementa un test. I test sono dei controlli fatti sulle chiamate in arrivo per determinare la legittimità di un messaggio (vedi Capitolo 3). Ad esempio, il controllo della WoT è implementato all'interno di un modulo che si chiama, senza troppa fantasia, modulo `wot`.

L'organizzazione dei test in moduli offre a VoIP-SEAL un alto livello di personalizzazione e dinamismo, caratteristica necessaria per un sistema ancora in fase di prototipo. Infatti per aggiungere nuove funzionalità al sistema (ovvero, nuovi test) non è necessario modificare il funzionamento del complicato Core, bensì è sufficiente usare la semplice interfaccia tra il Core e i moduli e implementare il modulo.

Sia il Core che i moduli sono scritti in C o C++. Il codice del Core viene compilato in un eseguibile, mentre i moduli vengono compilati in shared libraries, ovvero oggetti che vengono caricati dinamicamente dal programma, in modo da poter permettere una sostanziale indipendenza tra il core e i moduli. VoIP-SEAL ha anche un'interfaccia grafica di amministrazione scritta in Java.

5.1 Il database della Web of Trust

Il modulo `wot` deve poter cercare i TP tra identità all'interno della WoT del sistema anti-SPIT. Per fare questo ha bisogno di una visione complessiva della WoT, che viste le caratteristiche di tempo reale dell'applicazione deve essere di accesso molto veloce. Per questo il grafo della WoT deve essere

caricato nella memoria del programma e l'informazione strutturata in modo da essere computazionalmente efficiente.

Ogni keyserver ha una visione complessiva della WoT globale. Da questa WoT, come visto, deve essere estratto lo SCS più grande, che in qualche modo il modulo dovrà prelevare e utilizzare per fare le proprie ricerche. L'estrazione dello SCS dalla WoT non è un'operazione semplice, al contrario ha bisogno di una grande quantità di tempo, per cui deve essere eseguita periodicamente. Per quanto riguarda la WoT di OpenPGP, esiste un ottimo lavoro svolto da Jörgen Cederlöf, che ha scritto `pks2wot`[1], un programma in grado di estrarre lo SCS dalla WoT. Questo programma viene eseguito ogni 24 ore su uno dei keyserver di OpenPGP (in particolare, lo svizzero `wwwkeys.ch.pgp.net`) e produce un file della dimensione di circa 1 MB, che contiene il grafo dello SCS e viene pubblicato su Internet. Il file è nel formato `.wot`[2], definito dallo stesso Cederlöf. Come abbiamo visto, la WoT di OpenPGP non può essere utilizzata per un sistema anti-SPIT, tuttavia offre delle caratteristiche di verosimiglianza con quella che potrebbe essere la WoT del sistema anti-SPIT, per questo il modulo `wot` di VoIP-SEAL è stato scritto in modo da caricare il database della WoT da file nel formato `.wot`. Per lo stesso motivo, per fare i test è stato utilizzato proprio il database prodotto da `pks2wot`.

Il modulo `wot` funziona dunque costruendo in memoria una struttura dati che rappresenti la WoT contenuta in un file `.wot`. Per ogni identità A contenuta nella WoT vengono caricati in memoria il key ID di 4 byte, una stringa contentente l'indirizzo SIP dell'identità, le relazioni di fiducia che ha A verso le altre entità ($A \rightarrow FS_i$) e le relazioni di fiducia dalle altre identità verso A ($BS_i \rightarrow A$). Chiamiamo l'insieme delle identità verso cui A ha una relazione di fiducia Foreward Star di A (FS_A), e quelle che hanno una relazione di fiducia verso A, Backward Star di A (BS_A). Il motivo per cui la struttura dati deve contenere non soltanto la FS ma anche la BS di ogni chiave è svelato nel paragrafo successivo, in cui si spiega il funzionamento dell'algoritmo usato per la ricerca dei TP.

5.2 L'algoritmo BIDDFS di ricerca del Trust Path

Il test della WoT deve cercare il TP, ovvero il percorso di fiducia minimo dall'identità chiamata verso il chiamante. Per scegliere quale sia l'algoritmo migliore da utilizzare, è stato fatto un confronto tra Breadth-First Search (BFS) e un algoritmo di ricerca Bidirezionale basato su Iterative Deepening Depth-First Search, che qui chiameremo BIDDFS[3].

[1] `pks2wot` è uno script Python contenuto nel pacchetto Wotsap, vedi [Ced]

[2] Il documento che definisce la sintassi del formato del file è pubblicato in [Ced07]

[3] Per mancanza di tempo, il confonto tra i due algoritmi è stato studiato solo a livello teorico, e non si è mai proceduto ad un benchmark che avrebbe portato certamente a

L'algoritmo comunemente più usato per trovare un percorso minimo in un grafo orientato è il BFS. Il funzionamento di questo algoritmo prevede una ricerca in ampiezza a partire dal nodo di origine lungo gli archi uscenti. A partire dalla profondità 0, vengono visitati prima tutti i nodi a profondità i, poi tutti quelli a profondità $i + 1$ e così via, fino al raggiungimento della destinazione o al fallimento della ricerca nel caso in cui la destinazione non sia raggiungibile. Con l'aumentare della profondità cresce geometricamente il numero di nodi da visitare. BFS ha bisogno come struttura dati ausiliaria di una pila FIFO nella quale sono contenuti i nodi raggiunti in precedenza che devono ancora essere visitati. Detta l la lunghezza del TP e m il numero medio di relazioni di fiducia che ha un'identità verso altre identità, sia lo spazio occupato dalla coda che il tempo impiegato al caso pessimo è $\sum_{i=0}^{l} m^i = O(m^l)$. Si dimostra che BFS è completo e, per grafi non pesati, trova la soluzione ottima.

Si porta l'attenzione su due caratteristiche della versione classica di BFS. La prima è che non utilizza le informazioni sulla Backward Star dei nodi, visto che la ricerca viene eseguita a partire dal nodo di origine e si propaga solo sugli archi uscenti. Dato che l'informazione sulla BS si può estrarre facilmente dalle WoT e può essere usata per rendere più efficiente l'algoritmo, l'uso di questa informazione costituisce per la particolare applicazione un miglioramento rispetto a BFS. La seconda è che ha bisogno di una struttura dati ausiliaria di dimensioni notevoli, e questo crea dei problemi di scalabilità rispetto alla dimensione della WoT più di quanti ne crei il tempo di esecuzione, considerando anche che VoIP-SEAL potrebbe dover eseguire diverse ricerche in parallelo (una per ogni chiamata in arrivo), e la dimensione delle code potrebbe rivelarsi un collo di bottiglia.

L'algoritmo di ricerca BIDDFS funziona in maniera diversa. Anche questo è un algoritmo completo e restituisce la soluzione ottima, ma a differenza di BFS sfrutta l'informazione riguardo alla BS dei nodi, raggiungendo una complessità di $O(m^{l/2})$. L'algoritmo in questione è il riassunto di diverse caratteristiche di BFS e Iterative Deepening DFS (IDDFS). Come vedremo, offre delle ottime performance e una sostanziale semplicità di implementazione. La ricerca avviene in due direzioni: dal nodo di origine n_{src} si percorre l'albero generato *in avanti*, ovvero utilizzando le FS dei nodi visitati, mentre dal nodo di destinazione n_{dest} si percorre l'albero *all'indietro*, ovvero usando la BS. Durante le visite, si tiene memoria del percorso fino al nodo analizzato correntemente. Se la ricerca all'indietro incontra un nodo visitato durante la ricerca in avanti, significa che questo nodo (il *nodo di mezzo*, n_m), è un nodo attraversato da uno dei percorsi minimi $n_{src} \rightarrow n_{dest}$. La lunghezza di questo percorso minimo viene calcolata sommando la distanza di n_m dal-

risultati interessanti. Inoltre, non sono state prese in considerazione le possibilità riguardo all'uso di algoritmi euristici, dato che non sembra possibile la creazione di una funzione euristica ragionevolmente efficiente di scelta dei percorsi.

l'origine (ottenuta dalla prima visita) e dalla destinazione (ottenuta dalla seconda visita). In questo modo l'algoritmo trova l'esistenza di un percorso di una certa lunghezza. Per trovare quale sia esattamente questo percorso, è necessaria una terza visita $n_{src} \rightarrow n_m$ che scriva in memoria il percorso minimo $n_{src} \rightarrow n_m$, mentre quello $n_m \rightarrow n_{dest}$ è già stato scritto durante la seconda visita. Più formalmente, i passi di ogni iterazione i dell'algoritmo BIDDFS sono descritti dai seguenti passi, con il valore iniziale di $i = 0$:

1. Se i è maggiore della distanza massima del path, restituisci errore (path inesistente o troppo lungo).

2. Conduci una visita DFS a profondità limitata (vedi sotto) a partire da n_{src} e utilizzando le FS dei nodi visitati (ricerca *in avanti*). La profondità è limitata a $\frac{i}{2}$ se i è pari, a $\frac{i}{2} + 1$ se i è dispari. Durante la visita si colorano i nodi impostando il valore dell'etichetta alla distanza minima del nodo dall'origine. Se la visita restituisce "success", restituisci "è stato trovato un path con lunghezza i".

3. Conduci una visita analoga ma *all'indietro*, ovvero partendo dal nodo destinazione e usando la BS dei nodi visitati. La profondità è limitata a $\frac{i}{2}$. Durante la visita, si colorano i nodi impostando il valore dell'etichetta alla distanza minima dal nodo destinazione, cambiata di segno.

4. Se durante la visita si incontra un nodo con etichetta positiva, quello è il nodo di mezzo, esiste dunque un percorso lungo i. Se ci si accontenta di sapere dell'esistenza di un percorso lungo i, restituisci i, altrimenti imposta il grafo per la prossima ricerca ed esegui una visita in avanti $n_{src} \rightarrow n_m$ e restituisci i.

5. Imposta il grafo per la prossima ricerca.

6. Incrementa i; Ritorna al passo 1.

La visita DFS a profondità limitata invocata sul *nodo iniziale* e con profondità massima p è descritta dai seguenti passi:

1. Se la profondità massima p dal nodo iniziale è stata superata, restituisci "failure".

2. Se il nodo visitato è il nodo destinazione, restituisci "success"

3. Scrivi in memoria nella struttura *trust-path* che il nodo visitato alla profondità p è il presente, colora il nodo analizzato con la distanza dal nodo iniziale.

4. Se la visita è all'indietro e il nodo visitato è stato visitato anche durante la precedente visita in avanti, restituisci "success".

5. Su ogni nodo della FS (nel caso che si tratti della visita in avanti) o della BS (nel caso della visita all'indietro), reinvoca la visita decrementando la distanza massima dal nodo iniziale;

6. Se il risultato di questa visita è "success" restituisci "success", se il risultato è "failure" passa al nodo successivo.

7. Se non ci sono altri nodi da visitare, restituisci "failure"

Uno schema che rappresenta il funzionamento dell'algoritmo Bidirezionale con IDDFS è mostrato in Figura 5.1.

5.2.1 Complessità di BIDDFS in termini di tempo

Per l'applicazione particolare, BIDDFS risulta molto migliore di BFS. Come si deduce dai passi descritti sopra, il numero di nodi visitati al caso pessimo per trovare il TP è quello della ricerca IDDFS lungo i due alberi in avanti e all'indietro per ogni iterazione, mentre nell'ultima iterazione va aggiunta la terza ricerca, ovvero, detta $V(l, m)$ la funzione che determina il numero massimo di nodi visitati per trovare un percorso lungo l in una rete in cui ogni nodo ha m archi uscenti e detta $v(i, m)$ la funzione che determina il numero medio di nodi visitati all'iterazione i nella stessa rete, è:

$$v(i,m) = \begin{cases} 2 \cdot \sum_{j=0}^{i/2}(m^j) & \text{per } i \text{ pari} \\ \sum_{j=0}^{\lfloor i/2 \rfloor}(m^j) + \sum_{j=0}^{\lceil i/2 \rceil}(m^j) & \text{per } i \text{ dispari} \end{cases}$$

$$V(l,m) = \begin{cases} \sum_{i=0}^{l} v(i) + \sum_{j=0}^{l/2}(m^j) & \text{per } l \text{ pari} \\ \sum_{i=0}^{l} v(i) + \sum_{j=0}^{\lceil l/2 \rceil}(m^j) & \text{per } l \text{ dispari} \end{cases}$$

V è dunque dato dalla somma parziale di progressioni geometriche di ragione m con esponente $\leq \lceil l/2 \rceil$, è dunque $V = O(m^{l/2+1})$. Da questa analisi si è esclusa l'operazione I: *Imposta il grafo per la prossima ricerca*, che verrà analizzata con maggior dettaglio. Il grande guadagno in termini di tempo rispetto a BFS è dovuto al fatto che i nodi visitati sono molti meno. La crescita degli alberi resta esponenziale, ma utilizzando due alberi invece di uno l'esponente è dimezzato rispetto a BFS. Per dare un'idea degli ordini di grandezza, si porta di nuovo l'esempio di $m = 100$ e $l = 6$. In questo caso si ha per BFS un numero massimo di nodi visitati di $100^6 = 10^{12}$ e di spazio ausiliario 10^{12}, mentre con l'algoritmo di ricerca BIDDFS si ha un valore al

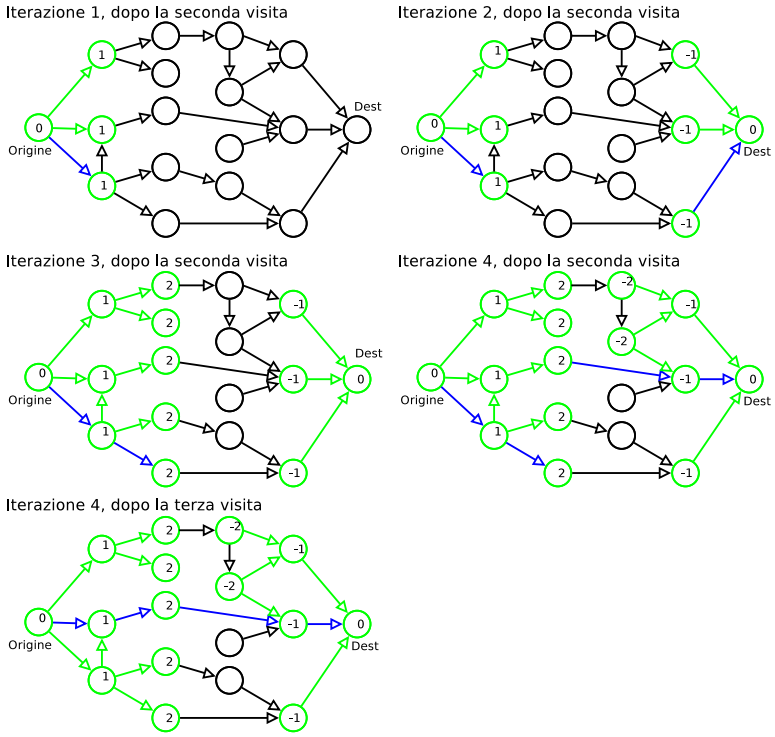


Figura 5.1: Esempio del funzionamento dell'algoritmo BIDDFS. È mostrato lo stato del grafo dopo ogni iterazione $i = 1 \dots 4$. In blu è mostrato il percorso temporaneamente salvato nella struttura *trust-path*. In ogni iterazione vengono eseguite sia la ricerca DFS a profondità limitata in avanti, sia all'indietro (prima e seconda visita). Si suppone che, durante la visita, i nodi siano scelti dal più alto al più basso. Il numero al centro dei nodi è il valore dell'etichetta (il colore) assegnato dall'algoritmo. Nell'iterazione 4, la ricerca lungo la BS a partire dalla destinazione ha individuato il nodo di mezzo già visitato dalla ricerca in avanti, quindi per trovare il percorso viene eseguita anche la terza visita, dal nodo sorgente al nodo di mezzo (lo stato dopo questa visita è mostrato nell'ultimo grafo). Si noti che, a parte per i casi particolari in cui si trova un percorso alle iterazioni $i = 0$ e $i = 1$, il nodo di mezzo è sempre individuato dalla ricerca all'indietro. Si noti inoltre che un percorso di lunghezza i viene trovato, se esiste, all'iterazione i.

caso pessimo di

$$\begin{aligned} V(6,100) &= v(0) + v(1) + v(2) + v(3) + v(4) + v(5) + v(6) + \sum_{j=0}^{3}(100^j) \\ &= 2 + 102 + 202 + 10202 + 20202 + 1020202 + 2020202 + 1010101 \\ &\simeq 4 \cdot 10^6 \end{aligned} \tag{5.1}$$

e - come si spiega successivamente - in spazio di $O(k)$ costante.

Si riportano nella tabella di seguito alcuni valori di $V(l, m)$ per $m = 10$. Come si vede, per l dispari il numero di nodi da visitare al caso pessimo subiscono un salto in positivo. Questo perché alle iterazioni dispari aumenta l'ordine di grandezza della prima e della terza visita in avanti.

l	$V(l, 10)$: n. di nodi visitati
1	25
2	47
3	269
4	491
5	2.713
6	4.935
7	27.157

Ad ogni iterazione i dell'algoritmo, i risultati trovati all'iterazione $i-1$, ovvero le distanze minime trovate nelle visite IDDFS vengono scartati. Questo permette di non avere strutture dati ausiliarie in cui salvare la distanza provvisoria dei nodi visitati dal nodo sorgente, ma può sembrare uno spreco. In realtà, nell'ipotesi che sia $m \geq 2$, lo spreco delle visite $0, 1, .., i-1$ è minore rispetto al numero di nodi visitati all'iterazione i. Ad esempio, per cercare un TP di $l = 6$, dei circa $4 \cdot 10^6$ nodi che vengono visitati, solo $1 \cdot 10^6$ sono visitati durante le iterazioni $i = 0, 1, .., 5$.

Si pone inoltre l'attenzione sul fatto che il caso pessimo analizzato sopra diventa tanto più improbabile quanto cresce l. Infatti, la progressione geometrica rappresenta il numero di nodi da visitare ad ogni profondità dell'albero in cui, di media, ogni padre ha m figli. Invece per quanto riguarda le WoT, essendo i nodi molto connessi, la visita lungo l'albero comprenderà, al crescere della distanza dal nodo sorgente, sempre più nodi che sono già stati incontrati da visite DFS fatte da BIDDFS nella stessa iterazione. Parte di questi nodi saranno già stati raggiunti con distanza minore o uguale, dunque non andrebbero visitati. Questo vuol dire che molti dei rami dell'albero vengono tagliati, e il reale numero di nodi da visitare è molto minore del massimo espresso dalla funzione V.

L'operazione I, che imposta il grafo per la visita successiva, è necessaria al funzionamento di BIDDFS per distinguere i nodi incontrati nella visita corrente da quelli incontrati in una visita precedente. Per quanto riguarda questa operazione, all'algoritmo BIDDFS si può apportare un'importante ottimizzazione. Per quanto visto finora, il tempo impiegato dall'algoritmo è $\Theta(V(l))$, dove $V(l) = O(m^{l/2+1})$ è il numero di nodi visitati. Per mantenere questo risultato, bisogna tuttavia fare in modo che anche l'operazione I, effettuata per ogni visita, abbia complessità $O(V(l))$. Ad esempio, se I fosse implementata con l'azzeramento del colore in tutti i nodi del grafo, questo porterebbe l'intero algoritmo ad essere di complessità $O(n)$ invece che $\Theta(V(l))$, dove n è il numero totale di nodi del grafo della WoT, questo significa che il tempo non dipenderebbe più dal numero di nodi visitati ma dalla dimensione della WoT, creando problemi di scalabilità rispetto alla sua espansione. L'ottimizzazione che si può applicare per risolvere praticamente questo problema consiste nell'aggiunta ad ogni nodo del *timestamp*, un numero intero senza segno da incrementare ad ogni visita e che indica il codice dell'ultima visita che è stata fatta sul nodo, in modo da poterla confrontare con le visite successive (l'algoritmo di visita presentato sopra deve dunque subire modifiche minori, ovvero l'aggiunta di un `if`, che è $O(k)$). Questo aumenta lo spazio utilizzato dai dati di $d \cdot n$, dove d è la dimensione in bit del timestamp, facendola rimanere $\Theta(n)$. Per quanto riguarda invece la complessità in termini di tempo, si consideri che, ad ogni invocazione dell'operazione I, potrebbe essere necessario, nel caso che il timestamp vada in overflow, l'azzeramento del timestamp in ogni nodo del grafo, per evitare conflitti di nuove visite con visite passate. In caso di overflow, I richiederebbe dunque $\Theta(n)$, mentre in caso contrario I dovrebbe semplicemente incrementare il timestamp per la visita successiva, sarebbe dunque $I \in O(k)$. Si definisce la probabilità $0 \leq p \leq 1$ come la probabilità che, per una generica invocazione di I, il timestamp vada in overflow. La complessità di I sarebbe dunque:

$$I \in \begin{cases} \Theta(k) & \text{con probabilità } (1-p) \\ \Theta(n) & \text{con probabilità } p \end{cases}$$

Bisogna considerare che p dipende dal numero di cifre binarie d con cui si rappresenta il timestamp, è infatti $p = \dfrac{1}{2^d}$. Ad esempio, nel caso che il timestamp sia rappresentato come un numero a 32 bit, varrebbe $p = \dfrac{1}{2^{32}}$, ovvero I avrebbe complessità $\Theta(n)$ in un caso su circa 4 miliardi. Aumentando la dimensione del timestamp a 64 bit, p diverrebbe definitivamente trascurabile, rendendo nella pratica $I \in \Theta(k)$. In definitiva, grazie a questa ottimizzazione è possibile mantenere BIDDFS $\in O(m^{l/2+1})$.

Tuttavia l'aggiunta del timestamp ha un lato negativo, dato che rende impossibile svolgere due ricerche in parallelo. Infatti nel caso di due ricerche

parallele che visitano uno stesso nodo, sorgerebbe un conflitto sull'assegnazione del timestamp per quel nodo. Questo significa che, anche se il processo complessivo di analisi della chiamata (vedi sotto) può restare parallelo, la ricerca dei TP deve avvenire in modo sequenziale.

5.2.2 Complessità di BIDDFS in termini di spazio

L'algoritmo BIDDFS utilizza strutture dati ausiliarie in $O(k)$. Infatti, oltre ad un numero limitato di variabili di cui ha bisogno, per la visita nel grafo utilizza una pila di stack che occupa $O(l \cdot s)$, dove l è la lunghezza del percorso cercato e s è la dimensione di un nuovo stack per la funzione di visita. Tuttavia si maggiorano sia s, che è costante, sia l, che è maggiorata dal percorso più lungo nella WoT e comunque dalla lunghezza massima del percorso considerata durante la visita. Risulta dunque $O(l \cdot s) = O(k)$ e BIDDFS $\in O(k)$ in termini di spazio.

5.3 Le operazioni del modulo `wot`

Dopo aver illustrato come funziona in particolare l'algoritmo di ricerca del TP, si spiega quali azioni deve intraprendere il modulo `wot` per funzionare correttamente. Infatti, il test della WoT non può limitarsi a calcolare la lunghezza del TP dal chiamato al chiamante e restituire il punteggio relativo, ma deve eseguire ulteriori operazioni e controlli, tra cui la verifica della firma rispetto al messaggio: non ha senso calcolare la lunghezza del TP se il mittente non dimostra la propria identità con una firma corretta. Le operazioni del modulo `wot` si possono condurre a due categorie: quelle *di modulo*, da eseguire al caricamento e allo scaricamento del modulo e quelle *di chiamata*, da eseguire per ogni chiamata. Segue un elenco delle operazioni.

Operazioni di modulo Queste operazioni possono anche avere un costo elevato, dato che vengono eseguite raramente e non hanno requisiti di tempo reale.

1. **Caricamento della configurazione:** VoIP-SEAL ha un file in cui sono contenuti alcuni valori di configurazione che devono essere letti e caricati all'avvio del modulo. Tra queste informazioni, vi è il nome del file `.wot` da caricare, il nome del keyserver da cui scaricare le chiavi per la verifica, la lunghezza massima del TP da considerare, alcuni valori personalizzati di punteggi da dare in alcuni casi particolari (es, se la chiamata in arrivo non è firmata).
2. **Caricamento della WoT:** Il file `.wot` contentente tutte le informazioni utili sulla WoT viene caricato in memoria in una struttura dati che rappresenta il grafo. Per ogni identità sono caricati il Key ID, la stringa di descrizione dell'identità (che contiene l'indirizzo SIP) e le relazioni di fiducia da e verso l'identità.

3. **Creazione e distruzione dei contesti per le librerie:** Le librerie ausiliarie utilizzate, in particolare CDK per la verifica delle firme OpenPGP e Sofia per l'analisi dei messaggi SIP, hanno bisogno di un ambiente di esecuzione che deve essere creato all'avvio del modulo e distrutto alla disattivazione del modulo.

Operazioni di chiamata Queste operazioni possono dover essere eseguite per ogni chiamata, dunque devono rispettare i requisiti di tempo reale della comunicazione, in altre parole, è particolarmente importante la loro performance. Non tutte le operazioni devono essere eseguite per ogni chiamata. In Figura 5.2 è presentato il diagramma di attività che illustra quali operazioni debbano essere eseguite nei vari casi. Per l'elenco di tutti i possibili percorsi della trattazione di una chiamata nel diagramma di attività, si veda anche il Capitolo 6 sull'analisi delle performance.

1. **Verifica della firma:** Prima di verificare la distanza nella WoT dal chiamato al chiamante, è necessario attestare l'identità del chiamante, verificando con la chiave pubblica del chiamante la validità della firma rispetto al messaggio. Per fare questo, la firma deve essere presente e integra[4], inoltre la chiave pubblica del chiamante deve essere disponibile.
2. **Download della chiave:** Nel caso che la chiave pubblica del chiamante non sia presente localmente, sarà necessario scaricarla da un keyserver, ovvero stabilire una connessione HTTP con il keyserver, inviare la richiesta per la chiave e aspettare una risposta. Evidentemente questa è l'operazione che richiede più tempo, ma bisogna considerare che viene eseguita solamente la prima volta che il chiamante telefona. Per le telefonate successive, il modulo salva la chiave in un database locale in modo da averla subito disponibile per verificare le telefonate future provenienti da quell'identità.
3. **Ricerca del Trust Path:** Se la verifica della chiave è andata a buon fine, e sia il chiamato che il chiamante sono presenti nella WoT, si può cercare il TP dal chiamato al chiamante e restituire il punteggio in funzione della lunghezza del percorso.

5.4 I valori restituiti dal test

Come illustrato nel Capitolo 3 tutti i moduli di VoIP-SEAL restituiscono, dopo essere stati invocati per analizzare una chiamata in arrivo, un punteggio $-1 \leq s \leq 1$che indica la probabilità che la chiamata sia spam secondo il

[4]Le firme PGP hanno un codice di controllo per verificarne l'integrità.

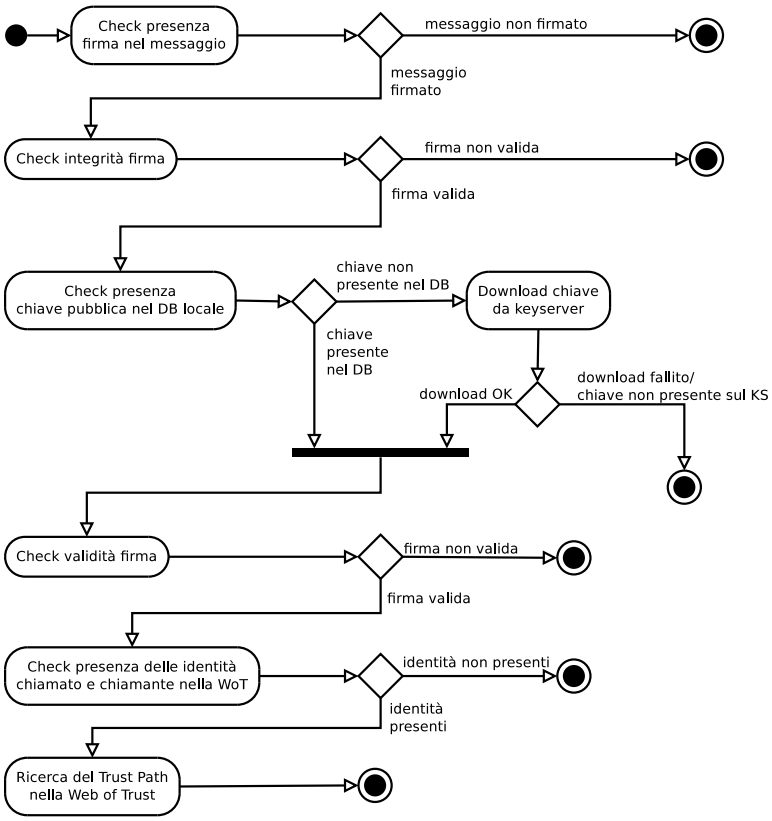


Figura 5.2: Il diagramma di attività che presenta il funzionamento del modulo `wot` nel trattamento dei possibili messaggi. Ogni stato di terminazione corrisponde ad un particolare valore restituito dal modulo. La ricerca del TP avviene solamente se il messaggio è firmato, la firma è integra, la chiave è presente nel DB locale oppure scaricabile dal keyserver, la firma è valida rispetto al messaggio ed entrambe le identità chiamante e chiamato sono presenti nella WoT.

test eseguito dal modulo. Per la semantica dei valori di s si veda la Sezione 3.1.

Come accennato, il modulo `wot` non può dare giudizi sulla probabilità che si tratti di spam ($s > 0$). Si pensi ad esempio al peggiore dei risultati che può dare il modulo `wot`, ad esempio che il TP sia troppo lungo o non esista, oppure che il messaggio non sia firmato o la firma non sia valida: in tutti questi casi, non si può comunque stabilire che un messaggio sia spam soltanto perché non implementa correttamente (o non implementa affatto) W-BASA. Per questo, i punteggi di spam restituiti da VoIP-SEAL saranno tutti $-1 \leq s \leq 0$.

I possibili risultati del test WoT si possono dunque dividere in due categorie:

Risultati dati dalla ricerca nella WoT Se la firma del messaggio è corretta, è possibile effettuare la ricerca. In questo caso, il punteggio restituito dal modulo è proporzionale alla lunghezza del percorso di fiducia. Detta l la lunghezza del TP per la telefonata analizzata, e m la lunghezza massima di un TP[5], vale:

- $l = 1 \Rightarrow s = -1$ (il messaggio è sicuramente non-spam se la persona chiamata si fida direttamente del chiamante)
- $1 < l < m \Rightarrow s = \frac{(l-1)}{(m-1)-1}$ (punteggio proporzionale alla lunghezza)
- $l = m \Rightarrow s = 0$ (se il TP è lungo quanto la soglia, il modulo non può dire alcunché sulla legittimità della chiamata)
- Per TP di lunghezza superiore a m, il punteggio resta $s = 0$

In formule, è

$$s = f(l, m) = \begin{cases} -1 & \text{se } l \leq 1 \\ \frac{(l-1)}{(m-1)-1} & \text{se } 1 < l < m \\ 0 & \text{se } l \geq 1 \end{cases}$$

Il grafico dei risultati è riportato in un esempio ($m = 6$) in Figura 5.3.

Casi in cui non è possibile la ricerca In questi casi, il punteggio è personalizzabile nel file di configurazione globale di VoIP-SEAL. In generale, i risultati di questi casi possono essere impostati a $p = 0$, dato che il test non è in grado di dare indizi sulla legittimità della chiamata. I casi sono qui elencati:

[5]Come già detto, è necessario porre alla lunghezza dei percorsi di fiducia un limite oltre il quale il percorso non è considerato. Questo limite può essere fissato tra 4 e 6, a seconda delle caratteristiche della rete. Il valore della lunghezza massima del Trust Path è personalizzabile nel file di configurazione globale di VoIP-SEAL.

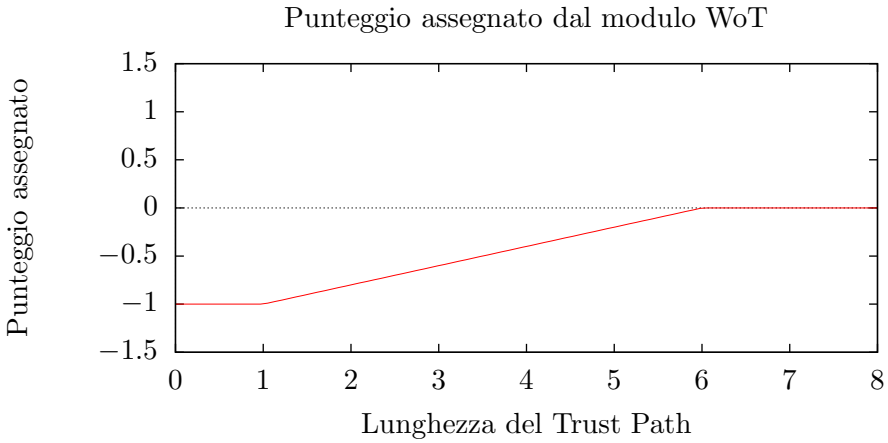


Figura 5.3: Punteggio assegnato ad una chiamata a seconda della lunghezza del TP nel caso che la firma sia corretta. La soglia è impostata a $m = 6$.

- Il messaggio non è firmato.
- La firma è presente ma non è integra.
- La chiave pubblica del mittente non è presente nel DB locale e non è stato possibile scaricarla dal keyserver.
- La firma non è valida.
- Il destinatario o il mittente non sono nello SCS della WoT (ovvero, non esiste un TP tra i due).
- Il TP esiste[6] ma è di lunghezza $l > m$

5.5 Il Signing Proxy

Il Signing Proxy è l'elemento del sistema che appone la firma ai messaggi in uscita (si veda a proposito la Figura 4.3 a pagina 24). Questo proxy è implementato come un server SIP stateless, infatti la sua unica funzione, ovvero la firma dei messaggi, non ha bisogno di mantenere lo stato delle comunicazioni. Le informazioni di cui ha bisogno per funzionare sono:

1. Le chiavi private OpenPGP delle identità del proprio dominio, necessarie per poter calcolare la firma da inserire nel messaggio;

2. Le informazioni fornite dal server Registrar riguardo alla registrazione dei terminali SIP, necessarie per stabilire l'autenticità dei messaggi.

[6]Se entrambe le identità sono presenti nella WoT, necessariamente esiste un percorso tra le due, visto che della WoT si considera solo lo SCS.

Prima di continuare la spiegazione del funzionamento del SP è bene illustrare come avviene la registrazione dei terminali SIP presso i server Registrar di competenza per il proprio *reame*[7]. Quando un utente vuole pubblicare la propria presenza presso il server Registrar, gli invia un messaggio di tipo `REGISTER` in cui è contenuto l'indirizzo SIP dell'utente. A questo punto, il server dovrà decidere se accettare la registrazione oppure rifiutarla. Ovviamente, perchè il server accetti la registrazione, deve essere certo dell'identità dell'utente, in modo da stabilire un'associazione tra l'indirizzo IP del telefono e l'utente. Questa verifica può avvenire in diversi modi, ma quello più diffuso e attualmente considerato più sicuro è quello del SIP Digest Authentication, definito in [RSC+02]. Questo protocollo di autenticazione basato sull'algoritmo di hashing MD5 permette di stabilire l'identità di un utente senza che questo debba inviare la propria password. Il protocollo è immune da attacchi di tipo *message replay* e di analisi *plaintext*.

L'algoritmo - illustrato a tratti molto generali - funziona con i seguenti passi:

1. Il client invia al server una richiesta di registrazione.
2. Il server risponde lanciando al client una *sfida* (challenge). Si tratta di una stringa casuale che il client deve combinare con la propria password e quindi crearne un hash con MD5, in modo da ottenere un codice da rispedire indietro. Il server calcola lo stesso codice per il confronto successivo.
3. Il client, se conosce la password segreta, risponde inviando il codice. Se il codice corrisponde a quello calcolato dal server, la registrazione avviene con successo.
4. Il server restituisce un messaggio di *Acknowledgement*.
5. Tutti i successivi messaggi inviati dal client al server conterranno un codice che permette al server di verificare l'autenticità del messaggio.

Questo meccanismo dunque permette al server Registrar di avere la certezza che i messaggi inviati da un telefono SIP provengano effettivamente da un determinato utente. Questa autenticazione ha valore esclusivamente all'interno del reame, ovvero nella comunicazione tra il client ed il server.

Quando il SP riceve un messaggio da firmare proveniente da un client, il server deve essere in grado di stabilire l'autenticità della richiesta, per questo vi sono due possibili soluzioni. La prima è che il SP funzioni anche come server Registrar, ovvero che svolga, oltre alla firma dei messaggi, anche tutte le operazioni di registrazione; la seconda è che il SP ed il server Registrar

[7]Con questo termine - in inglese *realm* - si intende lo spazio virtuale di un server Registrar e degli utenti che lo usano. In pratica, spesso corrisponde al dominio internet degli indirizzi, per cui il reame dell'università di Pisa si può chiamare unipi.it

siano entità separate, ma che il server Registrar comunichi al Signing Proxy le informazioni per verificare l'autenticità dei messaggi in arrivo[8]. Questo sistema garantisce la relazione tra un messaggio SIP ed un utente nella comunicazione tra il telefono mittente e il SP. Per garantire la stessa relazione fuori dal realm, ovvero nell'invio dal SP al destinatario, il SP, dopo aver verificato l'autenticità del messaggio, vi appone la firma PGP.

Il calcolo della firma avviene con una funzione di tipo $s = S(m, k_{prv})$, dove S è la funzione che genera la firma, m è il messaggio da firmare, k_{prv} è la chiave privata del mittente e s è la firma prodotta. La firma può essere verificata con una funzione di tipo $c = V(m, s, k_{pub})$ dove V è la funzione di verifica della firma, m è il messaggio, s è la firma prodotta dal mittente, k_{pub} è la chiave pubblica del mittente, c è un booleano che indica la correttezza della firma rispetto al messaggio. La firma s contiene anche meta-dati riguardo alla firma, quali la data in cui la firma è stata generata ed il Key ID del mittente.

Come messaggio m da firmare, il SP usa alcune parti del messaggio SIP di `INVITE`. In particolare, m è una stringa di alcuni dati concatenati e separati dal carattere '`|`'. I dati concatenati sono: indirizzi del mittente e del destinatario, ID della chiamata, campo `CSeq`, campo `Date`, campo `Contact` ed il payload del messaggio. Dunque, poniamo ad esempio che al SP arrivi il seguente messaggio:

```
INVITE sip:bob@acme.com SIP/2.0
Date:  Sun, 23 Aug 2009 14:54:40 GMT
CSeq:  1 INVITE
Via:  SIP/2.0/UDP bob.acme.org:5060;branch=z9hG4bK40ecd3;rport
User-Agent:  Ekiga/3.2.5
From:  Alice <sip:alice@xavier.org>;tag=44bf378a-628e
Call-ID: 44d8378a-628e@bob
To:  <sip:bob@acme.com>
Contact:  <sip:alice@xavier.org>
Allow:  INVITE,ACK,OPTIONS,BYE,CANCEL
Content-Type:  application/sdp
Content-Length:  875
Max-Forwards:  70

(...messaggio SDP...)
```

In questo caso, il testo m su cui calcolare la firma è:

[8]Durante la progettazione del prototipo di VoIP-SEAL il problema non è stato affrontato per mancanza di tempo, non essendo considerato particolarmente complesso e dunque degno di uno studio focalizzato. Nel prototipo, il Signing Proxy semplicemente appone la firma a tutti i messaggi in arrivo, senza considerare l'autenticità dei messaggi.

```
alice@xavier.org|bob@acme.com|44d8378a-628e-de11-9b7e-
00216b5dc72c@bob|1 INVITE|Sun, 23 Aug 2009 14:54:40
GMT|<sip:alice@xavier.org>|(...messaggio SDP...)
```

Dopo aver composto il messaggio m, il SP, avendo a disposizione le chiavi PGP private di tutti gli utenti del *realm*, può creare la firma $s = S(m, k_{prv})$. La firma s, prodotta da PGP in una stringa ascii di 101 caratteri, viene inserita dal SP nel campo `Authenticate`[9] del messaggio. Inoltre, come ogni volta che un messaggio SIP attraversa un proxy, viene aggiunto un campo `Via` e al campo `Via` precedente vengono aggiunte le informazioni `received` e `rport`[10], come definito in [RS03]. Nel complesso, il messaggio inviato dal SP verso la destinazione, ovvero in genere il server VoIP-SEAL di bob, è il seguente. Le parti sottolineate sono quelle aggiunte o modificate dal SP.

```
INVITE sip:bob@acme.com SIP/2.0
Date:  Sun, 23 Aug 2009 14:54:40 GMT
CSeq:  1 INVITE
Via:  SIP/2.0/UDP signing-proxy.acme.org:5060;branch=z9hG4bKh1u0aa;
rport
Via:  SIP/2.0/UDP bob.acme.org:5060;branch=z9hG4bK40ecd3;received=19
2.168.1.56;rport=9988
User-Agent:  Ekiga/3.2.5
From:  Alice <sip:alice@xavier.org>;tag=44bf378a-628e
Call-ID: 44d8378a-628e@bob
To:  <sip:bob@acme.com>
Authentication:  iEYEARECAAYFAkoSXoQACgkQzCa5QmSKyRGpwgCbBoI0HI4FXTL
ezl4FfS4KOurnEgoAoIeaUI57BgGxLFzQamLqzmyCARCf=kLWQ
Contact:  <sip:alice@xavier.org>
Allow:  INVITE,ACK,OPTIONS,BYE,CANCEL
Content-Type:  application/sdp
Content-Length:  875
Max-Forwards:  70

(...messaggio SDP...)
```

Quando il modulo `wot` del server VoIP-SEAL di Bob riceverà il messaggio SIP, potrà ricomporre la stringa m utilizzando gli stessi campi e usarlo per verificare la validità della firma contenuta nel campo `Authenticate`. Si ricorda che nella firma del campo `Authenticate` è contenuta come metadato anche il Key ID del chiamante, che viene usato dal modulo `wot` per interrogare il keyserver ed ottenere la chiave pubblica k_{pub} del mittente.

[9] Il campo Authenticate non è ancora standard SIP. Potrebbe tuttavia diventarlo se fosse accettato dalla comunità IETF. Per approfondire le considerazioni roguardo alla standardizzazione, si veda la Sezione 5.6.

[10] Queste informazioni servono al sistema SIP per costruire la route verso la destinazione, in modo che questa possa essere ripercorsa in direzione contraria per l'invio della risposta.

5.6 W-BASA è uno standard aperto

Alla luce delle scelte esposte in questo capitolo, è possibile approfondire alcune riflessioni enunciate in precedenza rispetto all'apertura del sistema. W-BASA fonda la propria base di conoscenza, ovvero la WoT, e di conseguenza la propria potenza, sulla diffusione del sistema stesso. Maggiore è il numero di persone che usano il sistema e fanno parte della WoT, più il sistema può garantire il corretto funzionamento delle telefonate legittime e l'isolamento delle telefonate di spam. Per questo, durante la progettazione del sistema, si è data una particolare attenzione all'uso di standard aperti, per fare in modo da creare un sistema universale, funzionante su scala globale, eterogeneo e indipendente dalle particolari implementazioni. In una parola, standardizzabile. Solo utilizzando uno standard aperto è possibile fare in modo che un gran numero di persone utilizzino il sistema e che la WoT cresca fino a comprendere un'alta percentuale di utenti VoIP in tutto il mondo. L'alternativa è un sistema proprietario, che potrebbe diffondersi solamente tra chi usa un particolare prodotto, che riuscirebbe - a meno di situazioni di monopolio - a coprire solo una bassa percentuale di utenti, rendendo il sistema inutile nella pratica.

VoIP-SEAL utilizza il protocollo SIP per l'inizializzazione della chiamata. SIP è stato scelto perché è open (è definito dalla RFC [RSC+02]), ed il suo utilizzo è in rapida crescita. Il protocollo si sta affermando nella maggior parte della applicazioni VoIP, con un gran numero di soft phone e apparecchi telefonici VoIP che lo utilizzano. Come spiegato, VoIP-SEAL rispetta fedelmente lo standard SIP. Ovviamente, l'aggiunta del campo `Authenticate` nei messaggi SIP non rispetta lo standard, ma ne è un'espansione. Certamente, se il sistema dovesse andare in produzione, si potrebbe avviare il processo presso la IETF per la standardizzazione del campo `Authenticate`, unico scoglio da superare perché W-BASA diventi standard.

Analogamente è stata fatta la scelta di OpenPGP come sistema per gestire la WoT, il trasferimento delle chiavi pubbliche e la firma dei messaggi. OpenPGP, oltre ad offrire delle caratteristiche di sicurezza e di performance tra le più avanzate, è completamente definito come standard aperto dalla RFC [CDF+07].

5.7 Conclusioni

VoIP-SEAL è un sistema anti-SPIT con una struttura modulare, in cui ogni modulo implementa un test per bloccare le telefonate di SPIT. Nel tirocinio è stato implementato il test basato su WoT come modulo di questo server. Per funzionare, il modulo ha bisogno di avere una conoscenza globale della WoT – che estrae dai keyserver – in cui cercare il TP. Per queste ricerche è stato ideato l'algoritmo BIDDFS che ha complessità $O(k)$ in spazio e $O(m^{l/2})$

in tempo, con una probabilità estremamente remota che la complessità sia $O(n)$. W-BASA, l'architettura di supporto al test WoT, è anche composta da un SP, che mantiene le chiavi private e ha il compito di apporre la firma ai messaggi `INVITE`. Per essere certo che un messaggio provenga effettivamente dal mittente indicato nel messaggio, deve usare i meccanismi SIP di autenticazione, dopodiché, in seguito all'inoltro del messaggio, la stessa garanzia è data dalla firma PGP aggiunta nel campo `Authenticate` del messaggio `INVITE` inoltrato. W-BASA può essere considerata un'architettura aperta e standardizzabile, visto che usa solo protocolli aperti.

Capitolo 6

Analisi della performance

Per verificare la possibilità effettiva di utilizzare il sistema della WoT per la prevenzione dello spam su VoIP sono stati portati a termine degli studi piuttosto accurati sulla performance del sistema. Questi studi hanno occupato una buona parte di tempo del lavoro complessivo. I test sono stati progettati e implementati completamente dal candidato, che ha anche analizzato i risultati ed elaborato la loro presentazione tramite i grafici.

I dubbi a cui si vuole trovare una risposta sono:

Domanda 1. *VoIP-SEAL può svolgere l'analisi dei messaggi in tempo reale?*

Domanda 2. *Può una particolare struttura o l'espansione della WoT pregiudicare il funzionamento del sistema?*

Come WoT per i test è stata utilizzata quella globale di OpenPGP. La scelta è ricaduta su questa perché è una delle poche reti di fiducia di dimensioni significative e facilmente accessibile. Inoltre, si è formulata l'ipotesi che una WoT sel sistema W-BASA conserverebbe caratteristiche molto simili alla WoT di OpenPGP.

Lo studio delle performance è stato diviso in due parti, presentate nelle prossime due sezioni di questo capitolo. Nella prima, vengono analizzate le performance complessive dell'analisi di una chiamata in tutti i casi possibili. Non studierà nel dettaglio il tempo di ricerca del TP, che verrà analizzato successivamente. In questa parte, si vuole rispondere alla Domanda 1. Nella seconda parte viene messo a fuoco il tempo necessario all'algoritmo per trovare il TP all'interno della WoT. Si è scelto di analizzare separatamente l'algoritmo di ricerca del TP perché questa è l'unica azione svolta dal modulo WoT che è influenzata dalla struttura e dalla dimensione della WoT. Quindi studiando i tempi delle ricerche svolte su WoT di dimensione diversa, si risponde alla Domanda 2.

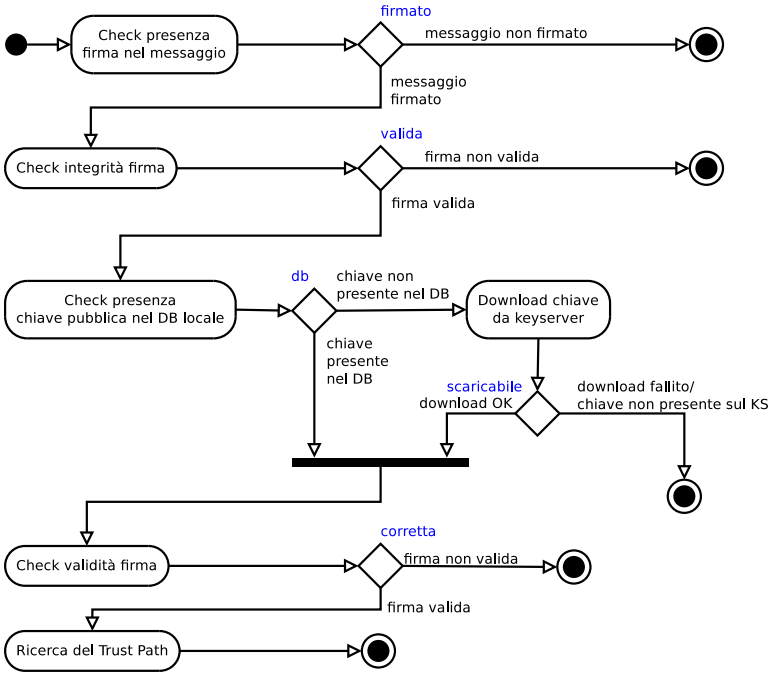


Figura 6.1: Diagramma delle attività che mostra l'albero dei possibili scenari di esecuzione del modulo WoT. Su ogni elemento di ramificazione (i rombi) è scritta in blu la variabile corrispondente.

6.1 Studio della performance degli scenari

Il diagramma delle attività in Figura 5.2 a pag. 38 presenta il comportamento del modulo WoT. Ogni possibile percorso dallo stato di inizio fino allo stato di terminazione è uno scenario corrispondente ad un possibile messaggio SIP in arrivo. Il diagramma viene riproposto in Figura 6.1 con l'aggiunta dei nomi delle variabili booleane che formano la configurazione di uno scenario. Le variabili hanno la seguente semantica:

- *firmato*: Il messaggio è firmato, ovvero è presente il campo `Authenticate`.
- *valida*: La firma è integra, ovvero è possibile estrarne il Key ID.
- *db*: La chiave pubblica del mittente che ha generato la firma è presente nel database locale, quindi non deve essere scaricata.
- *scaricabile*: La chiave è presente su un keyserver e può essere scaricata.
- *corretta*: La firma è corretta rispetto al messaggio.

Ogni possibile configurazione delle 5 variabili forma uno scenario di un possibile processamento di un messaggio. Tuttavia gli scenari possibili da

analizzare non sono $2^5 = 32$ ma solo 7, infatti alcune configurazioni non hanno senso. Ad esempio, se la firma non è presente nel messaggio (*firmato* = *false*) non ha senso chiedersi se la firma sia integra (*valida*). I 7 possibili scenari sono descritti nella tabella che segue. Le configurazioni sono rappresentate dalla tupla dei valori che assumono le 5 variabili, nell'ordine: (*firmato*, *valida*, *db*, *scaricabile*, *corretta*).

Configurazione	Descrizione dello scenario
C_1 =`(0,-,-,-,-)`	Il messaggio non è firmato.
C_2 =`(1,0,-,-,-)`	Il messaggio è firmato ma la firma non è valida.
C_3 =`(1,1,0,0,-)`	Il messaggio è firmato e la firma è valida, ma la chiave pubblica del mittente non è presente nel DB locale e non è scaricabile da un keyserver.
C_4 =`(1,1,0,1,0)`	Il messaggio è firmato e la firma è valida, la chiave non è presente nel DB locale ma è scaricabile dal keyserver. La verifica *non* ha successo.
C_5 =`(1,1,0,1,1)`	Come sopra, ma la verifica ha successo.
C_6 =`(1,1,1,-,0)`	Il messaggio è firmato e la firma è valida. La chiave pubblica del mittente è presente nel DB locale, la verifica *non* ha successo.
C_7 =`(1,1,1,-,1)`	Come sopra, ma la verifica ha successo.

Per il calcolo dei tempi necessari alle diverse operazioni, sono stati posti all'interno del codice della funzione di analisi dei messaggi del modulo WoT delle invocazioni alla funzione `int clock_gettime(clockid_t clk_id, struct timespec *tp)` della libreria `time.h`. Questa funzione scrive il tempo corrente nella struttura `tp` con la risoluzione di un nanosecondo. I tempi di invocazione di questa funzione si considerano trascurabili. A differenza della funzione `clock()`, la funzione `clock_gettime()` non misura l'utilizzo del processore da parte del processo, ma il tempo trascorso tra due punti del programma. Questa differenza che può sembrare sottile è fondamentale per la misurazione di alcune operazioni. Si pensi ad esempio all'operazione di download della chiave pubblica, che consiste in una connessione HTTP con il keyserver. Dopo aver inviato la richiesta, durante l'attesa della risposta il processore non è utilizzato (visto che il processo viene de-schedulato) ma il tempo che deve essere misurato è quello dall'inizio della transazione HTTP alla fine del trasferimento della chiave.

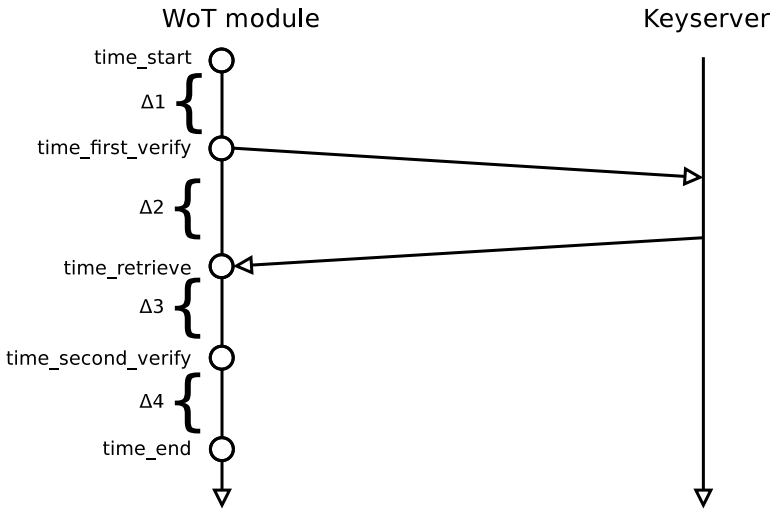


Figura 6.2: Gli istanti a cui vengono misurati i tempi per l'analisi della performance del modulo `wot`.

La struttura della funzione di analisi di una chiamata in arrivo da parte del modulo `wot`, che restituisce il punteggio s è la seguente:

```
1  double process_func(msg_t *incoming_call) {
2          clock_gettime(CLOCK_REALTIME, &time_start);
3          [prima verifica] //verifica della presenza della firma e
               prima verifica della firma
4          clock_gettime(CLOCK_REALTIME, &time_first_verify);
5          [download chiave]
6          clock_gettime(CLOCK_REALTIME, &time_retrieve);
7          [seconda verifica] //verifica della firma con la chiave
               appena scaricata
8          clock_gettime(CLOCK_REALTIME, &time_second_verify);
9          [ricerca Trust Path]
10         clock_gettime(CLOCK_REALTIME, &time_end);
11 }
```

Ovviamente, non tutti i blocchi tra parentesi quadre vengono eseguite per ogni chiamata analizzata, dunque questi blocchi contengono anche il codice di controllo del flusso, per calcolare la differenza tra i tempi e per restituire s.

In Figura 6.2 sono mostrati con dei pallini gli istanti a cui vengono presi i tempi del codice sopra. Sono anche definite le differenze $\Delta_{1...4}$ tra questi istanti, che sono i tempi di esecuzione delle *azioni di scenario*. Un'azione di scenario è un'azione che viene svolta tra un istante misurato e il successivo per un particolare scenario (quindi, lo stesso intervallo può misurare due azioni di scenario diverse in due scenari diversi, vedi tabella sotto). L'obbiettivo del test è analizzare il tempo delle azioni di scenario per osservare sotto quali condizioni sono rispettate le caratteristiche di real-time. Sono individuate 9 possibili azioni di scenario nella tabella che segue.

Azione di scenario	Descrizione	Intervallo
verifica$_1$	Verifica - firma non esiste	Δ_1
verifica$_2$	Verifica - firma non valida	Δ_1
verifica$_3$	Verifica - firma presente nel DB e corretta	Δ_1
verifica$_4$	Verifica - firma non presente nel DB	Δ_1
verifica$_5$	Verifica - firma presente nel DB, *non* è corretta	Δ_1
verifica$_6$	Verifica - chiave appena scaricata, firma corretta	Δ_2
verifica$_7$	Verifica - chiave appena scaricata, firma *non* corretta	Δ_2
connessione$_1$	Connessione al keyserver - firma scaricabile	Δ_3
connessione$_2$	Connessione al keyserver - firma *non* scaricabile	Δ_3
path	Ricerca del TP	Δ_4

Si definisce inoltre l'intervallo di esecuzione totale $\Delta_{tot} = \Delta_1 + \Delta_2 + \Delta_3 + \Delta_4$, che è il tempo complessivo di analisi per una chiamata.

Ognuno dei 7 scenari definiti in precedenza porta ad una lista di azioni di scenario che vengono eseguite:

Scenario	Configurazione	Lista azioni di scenario
Scenario 1	C_1 =`(0,-,-,-,-)`	verifica$_1$
Scenario 2	C_2 =`(1,0,-,-,-)`	verifica$_2$
Scenario 3	C_4 =`(1,1,0,0,-)`	verifica$_4$ $\rightarrow$ connessione$_2$
Scenario 4	C_4 =`(1,1,0,1,0)`	verifica$_4$ $\rightarrow$ connessione$_1$ $\rightarrow$ verifica$_7$
Scenario 5	C_5 =`(1,1,0,1,1)`	verifica$_4$ $\rightarrow$ connessione$_1$ $\rightarrow$ verifica$_6$ $\rightarrow$ path
Scenario 6	C_6 =`(1,1,1,-,0)`	verifica$_5$
Scenario 7	C_6 =`(1,1,1,-,1)`	verifica$_3$ $\rightarrow$ path

Dopo aver definito i 7 scenari, le 10 azioni di scenario ed i 5 istanti a cui viene misurato il tempo, si può procedere con il test. Durante il test, viene simulato l'inizio di un gran numero di telefonate processate sequenzialmente dal modulo WoT tramite altrettanti messaggi SIP di `INVITE` spediti ad un'istanza in esecuzione del server VoIP-SEAL. Sono stati preparati 7 messaggi SIP in modo da rispecchiare i 7 scenari (ad esempio, il messaggio inviato per simulare lo Scenario 1 non è firmato), e per avere un ampio campione statistico, ogni tipo di messaggio è stato inviato 15000 volte, per un totale di 105000 messaggi inviati durante i test. Durante l'analisi dei messaggi, il modulo `wot` prende nota degli istanti, calcola gli intervalli $\Delta_{1...4}$ e li scrive in un file, in modo che possano in seguito essere analizzati.

Per gli scenari in cui deve essere condotta la ricerca di un TP, si usa un valore generico per la misura di *path*, che è il tempo medio per la ricerca di un TP lungo 6 in una WoT standard di PGP che contiene circa 45.000 identità. Questa misura è riportata unicamente per confonto con le altre grandezze, mentre un'analisi più accurata della misura di *path* al variare della lunghezza del TP e della dimensione della WoT è presentata nella prossima sezione.

L'ambiente di esecuzione del test è stato un semplice PC con processore Athlon 2800 XP, 2 GB di RAM a 333MHz e sistema operativo Linux 2.6. Il computer era connesso ad Internet tramite un collegamento T3 (la connessione ad Internet influenza il tempo di download di una chiave pubblica). Il test, in totale, dura circa 8 ore, e deve essere eseguito a computer scarico, cioè senza altri programmi in esecuzione che potrebbero competere con l'uso delle risorse e influenzare i risultati del test. I messaggi sono stati inviati con la utility `sipp`[1]. In seguito l'elaborazione statistica dei dati è stata fatta da un programma C scritto ad-hoc.

I risultati del test ottenuti sono presentati nella tabella che segue. I valori sono espressi in μsec ed è indicato tra parentesi, per ogni valore, il coefficente di variazione $\sigma_r = \frac{\sigma_x}{\overline{x}}$ che fornisce una stima della deviazione relativa dei valori rispetto alla media.

Scenario	$\Delta_1(\sigma_{r1})$	$\Delta_2(\sigma_{r2})$	$\Delta_3(\sigma_{r3})$	$\Delta_4(\sigma_{r4})$	$\Delta_{tot}(\sigma_{r\_tot})$
Scenario 1	2(12%)				2(12%)
Scenario 2	3(10%)				3(10%)
Scenario 3	618(14%)	98.389(48%)	3.185(13%)	2.840(2%)	105.032(45%)
Scenario 4	3.802(24%)			2.840(2%)	6.642(15%)
Scenario 5	678(22%)	83.153(75%)			83.839(75%)
Scenario 6	584(12%)	100.161(53%)	160(52%)		100.905(53%)
Scenario 7	786(5%)				786(5%)

Dunque, facendo le medie dei tempi di esecuzione per le diverse azioni di scenario, si giunge alla tabella seguente, in cui per ogni azione di scenario è riportato il tempo medio di esecuzione e - nuovamente - la descrizione:

Azione di scenario	Descrizione	Tempo (μsec)
$verifica_1$	Verifica - firma non esiste	2
$verifica_2$	Verifica - firma non valida	3
$verifica_3$	Verifica - firma presente nel DB e corretta	3.801
$verifica_4$	Verifica - firma non presente nel DB	626
$verifica_5$	Verifica - firma presente nel DB, *non* è corretta	786
$verifica_6$	Verifica - chiave appena scaricata, firma corretta	3.185
$verifica_7$	Verifica - chiave appena scaricata, firma *non* corretta	160
$connessione_1$	Connessione al keyserver - firma scaricabile	99.275
$connessione_2$	Connessione al keyserver - firma *non* scaricabile	83.153
path	Ricerca del TP	2.850

[1]SIPp è una utility di testing per il protocollo SIP. Link: `http://sipp.sourceforge.net/`

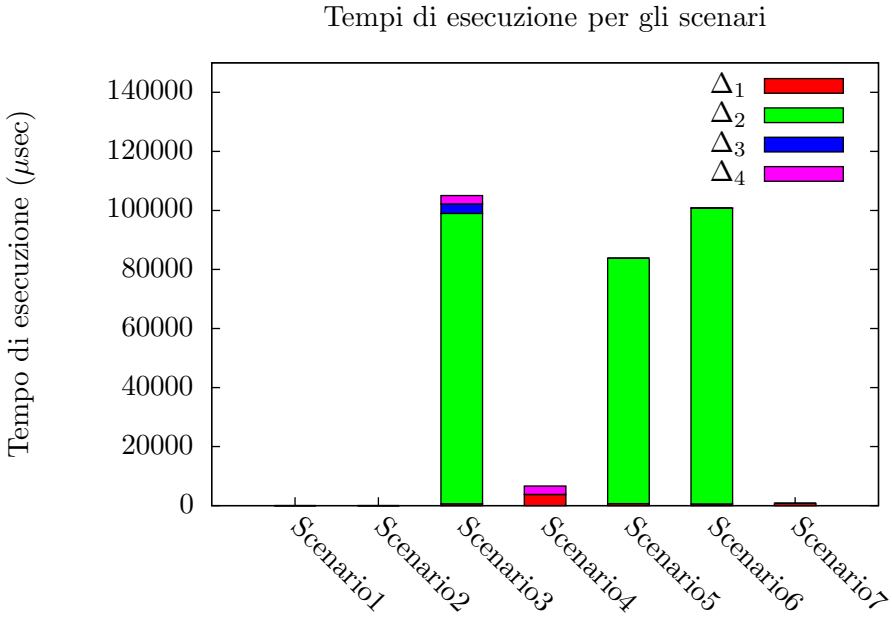


Figura 6.3: Grafico dei tempi di esecuzione per ogni scenario.

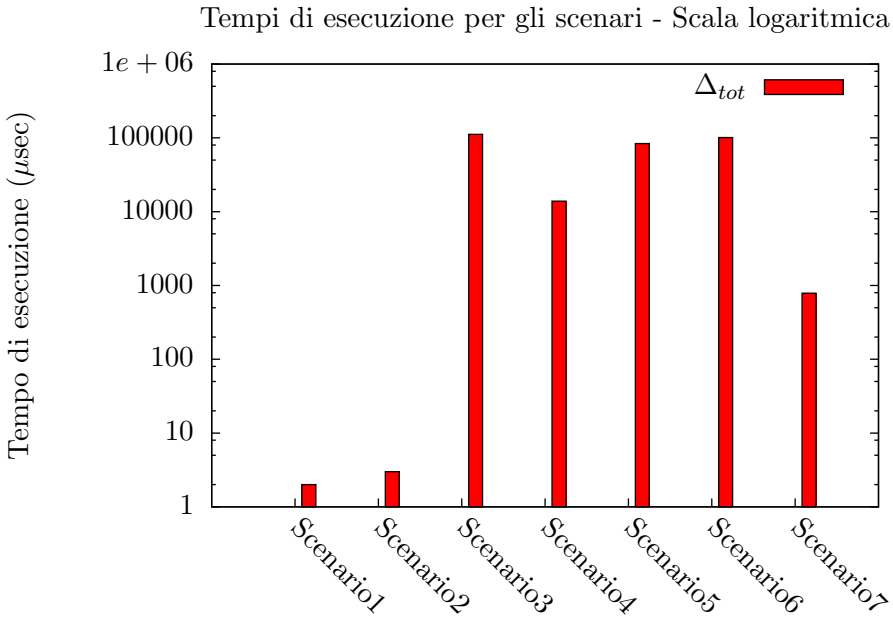


Figura 6.4: Grafico del tempo di esecuzione totale per ogni scenario, in scala logaritmica.

Osservando la ultime due tabelle ed i grafici in Figura 6.3 e 6.4, che mostrano i risultati fondamentali del test, saltano subito agli occhi alcune caratteristiche notevoli. Innanzitutto, le azioni hanno tempi di esecuzione che differiscono di ordini di grandezza. Per verificare che una firma non esiste ($verifica_1$) serve un millesimo del tempo che serve a verificare che una firma sia corretta ($verifica_3$, $verifica_6$). Come era da aspettarsi, le azioni che richiedono più tempo sono quelle che prevedono una connessione al keyserver tramite Internet. Ad esempio, scaricare una chiave ($connessione_1$) richiede ben 0.1 secondi - ma, si ricorda, il download avviene solo la prima volta che un nuovo utente chiama -, mentre collegarsi al keyserver ma ricevere una risposta negativa (quindi evitando il download della chiave) richiede $0,016$ secondi in meno. Il fatto che i due tempi siano simili è da ricondurre alla lentezza dello stabilimento di una connessione TCP, che affligge entrambe le transazioni. Verificando una firma, corretta o meno che sia, con una chiave appena scaricata ($verifica_6$, $verifica_7$) piuttosto che con una chiave scaricata in precedenza nel database ($verifica_3$, $verifica_5$), si guadagnano circa 500 μsec. Questo è dovuto al funzionamento della libreria OpenCDK, usata dal modulo WoT per gestire le chiavi e verificare le firme, che in seguito al download di una chiave la mantiene in una cache velocemente accessibile. 500 μsec è dunque il tempo necessario per recuperare una chiave pubblica non usata di recente dal database salvato come file sull'hard disk. Un altro elemento degno di nota è che, nello scenario più comune - ovvero lo Scenario 4, in cui la verifica ha successo e non è necessario stabilire la connessione - il tempo di ricerca del TP (*path*) incide per quasi la metà sul tempo totale di esecuzione. Questo indica che un'analisi più dettagliata di questo tempo è necessaria.

Osservando i valori del coefficente di variazione σ_r si nota che negli scenari 1, 2, 4, 7 si ha una sostanziale stabilità, con $\sigma_r \leq 15\%$. Gli scenari in cui invece il tempo di esecuzione subisce delle forti oscillazioni sono 3, 5, 6, ovvero quelli che richiedono una connessione con il keyserver. Infatti è proprio σ_{r2}, ovvero il coefficente di variazione dell'intervallo Δ_2, quello del download delle chiavi, ad assumere valori $\sigma_{r2} \geq 53\%$, rimanendo comunque $\sigma_{r2} \leq 75\%$, ad indicare che il tempo di interrogazione del keyserver può aumentare ma difficilmente vale più del doppio del valore medio. Le cause di questa instabilità sono da ricercare nelle condizioni della rete e nel carico del keyserver, non dipendono dunque dal modulo `wot`.

Questi dati mostrano che - a meno di crescite significative di *path*, vedi prossima Sezione - i requisiti di tempo reale sono rispettate in tutti gli scenari. Guardando i valori di Δ_{tot}, ovvero il tempo totale di esecuzione, che nel caso peggiore (Scenario 3) vale 105.042 con un coefficente di variazione del 45%, risulta che i tempi di analisi di una chiamata siano in ogni caso drasticamente inferiori al secondo. Per il "caso normale", ovvero una telefonata abituale che è già avvenuta in passato (Scenario 4), il tempo di esecuzione è di circa 7 msec.

Discorso a parte fa l'analisi del carico, ovvero l'analisi di quante telefonate possano essere processate contemporaneamente. Nei test svolti finora, le telefonate venivano analizzate sequenzialmente. Un test sul carico in cui vengono inviate un gran numero di chiamate che vengono analizzate in parallelo darebbe dei risultati interessanti. A priori si può dire che la concorrenza sull'uso delle risorse - che potrebbe dar luogo a scenari di congestione - non si dovrebbe verificare durante le azioni più lunghe, ovvero le connessioni al keyserver, dato che queste azioni non impegnano la macchina, rendendo possibile l'analisi di un'altra chiamata mentre si attende la lenta risposta dal server. Il collo di bottiglia sarebbe piuttosto l'esecuzione delle verifiche 3-6, ovvero quelle che impegnano l'hard disk (per il recupero della chiave dal DB) ed il processore (per la verifica della firma), e la ricerca del TP, che impegna processore e soprattutto memoria. Supponendo in linea teorica che l'assegnazione delle risorse da parte del sistema operativo avvenga in modo ottimale e non vi sia overhead, lo Scenario 4 (quello più lento escludendo i tempi di download della chiave) avrebbe un tempo totale di esecuzione di 6.652 μsec. Sotto questa ipotesi si può avere un ordine di grandezza del numero di telefonate - piuttosto incoraggiante - di questo tipo che il server può analizzare senza introdurre ulteriori ritardi di attesa: 150 telefonate/sec su un computer molto modesto.

6.2 Studio della performance del Path Finding

Dopo aver studiato i tempi complessivi di analisi di una chiamata, si sposta l'attenzione sull'analisi in dettaglio del tempo necessario al completamento dell'azione *path* definita nella sezione precedente, dove era compresa nell'intervallo Δ_4. D'ora in poi chiameremo questo intervallo di tempo per il path finding t_{pf}. L'analisi separata di questa azione è necessaria perché, come già spiegato, la performance di questa azione è l'unica ad essere influenzata dalla struttura della WoT. Per questo, mentre ci si aspetta che tutte le altre azioni richiedano dei tempi praticamente costanti, t_{pf} può variare molto a seconda del TP cercato e della WoT utilizzata. Con l'aumentare delle relazioni di fiducia nella WoT, aumentano anche il numero di nodi da visitare per le ricerche dei TP, e quindi il tempo necessario a trovarlo. Inoltre, con l'aumentare del numero di utenti, la WoT assumerebbe dimensioni sempre maggiori. Le domande a cui si vuole trovare risposta, tutte legate alla Domanda 2, sono:

Domanda 3. *Come sono distribuite le lunghezze dei Trust Path? In una WoT quanti TP hanno la stessa lunghezza l?*

Domanda 4. *Come varia t_{pf} al variare della lunghezza del Trust Path? (è lineare, esponenziale o altro?)*

Domanda 5. *Come sono distribuiti i valori di t_{pf}? Dati due TP $A \to B$ e $C \to D$ della stessa lunghezza, è necessario lo stesso tempo per trovare i due TP?*

Domanda 6. *Come varia t_{pf} al variare della dimensione della WoT in cui è effettuata la ricerca?*

Per rispondere ad ognuna di queste domande, sono stati impostati degli esperimenti. Stavolta non è stato necessario eseguire i test su un'istanza in esecuzione di VoIP-SEAL, né è stato necessario l'invio di un gran numero di messaggi SIP per provocare l'analisi. I test sono stati eseguiti con un programma ad-hoc che ha effettuato le ricerche e, sempre usando la funzione `clock_gettime()`, preso nota dei risultati di ogni ricerca (lunghezza del TP, tempo di esecuzione), in modo che potessero essere elaborati successivamente ed in fine usati per generare dei grafici.

Per ogni ricerca, sono stati scelti casualmente il nodo sorgente e il nodo destinazione. Il metodo utilizzato per scegliere casualmente[2] i nodi sorgente e destinazione richiede una spiegazione particolare. Si sarebbe potuto scegliere casualmente coppie di nodi (n_{src}, n_{dest}) all'interno della WoT ed eseguire la ricerca $n_{src} \to n_{dest}$ classificando il risultato (numero di nodi visitati e tempo impiegato dalla ricerca) a seconda della lunghezza, ma scegliere il campione del test in questo modo avrebbe privilegiato, in particolare per percorsi corti, la scelta di n_{src} con più archi uscenti, falsando i risultati. Per evitare questo, i nodi casuali sono stati scelti con il seguente algoritmo:

1. Scelta casuale di un nodo n_{src} nella WoT;
2. Colorazione di tutto il grafo a partire da n_{src}, colorando ogni nodo con il valore della distanza minima da n_{src};
3. Scelta casuale di un nodo n_{dest} per ogni lunghezza del TP, aggiunta al campione da analizzare della coppia (n_{src}, n_{dest}). Per la lunghezza si considerano i valori del colore dei nodi del grafo, in questo modo si ha la certezza che la lunghezza sia quella del percorso minimo;
4. Salto al passo 1.

Considerato questo, il programma che esegue il test è strutturato in questo modo:

1. Caricamento della Web of Trust in memoria;
2. Scelta casuale di 60.000 percorsi su cui fare i test, come descritto dall'algoritmo sopra. Scrittura in memoria delle coppie di nodi individuati.

[2]Ovviamente, per "scelta casuale" si intende in realtà scelta pseudo-casuale, infatti tutte le scelte di questo tipo sono state fatte con la funzione `random()` di `stdlib.h`.

3. Esecuzione di una ricerca sulle 60.000 coppie di nodi scelte casualmente, scrittura in memoria dei risultati di ogni ricerca (questa operazione richiede circa 2 ore);

4. Elaborazioni statistiche (suddivisione in classi di frequenza, calcolo media, varianza, deviazione standard, ecc);

5. Output dei dati riassunti in file compatibili con `gnuplot` per la generazione dei grafici.

Il test è stato eseguito su tre Web of Trust di diverse dimensioni, in modo da poter analizzare le performance al variare del numero di identità (Domanda 6). Le tre WoT sono tutte estratte da un keyserver del sistema OpenPGP ma in tempi diversi[3], conservano dunque le caratteristiche di verosimiglianza. Le tre Web of Trust sono state scelte con questo criterio: la prima è la più antica archiviata, la terza è la più recente, la seconda è scelta in modo da avere un numero di identità medio tra la prima e la terza. Le caratteristiche delle tre WoT scelte sono le seguenti:

WoT	Data estrazione	N. identità $\|WoT\|$	N. firme	media firme/identità m
WoT_1	25 feb 2005	25487	230445	$9,04$
WoT_2	1 giu 2006	33050	328912	$9,95$
WoT_3	7 ott 2008	40480	405289	$10,01$

6.2.1 Domanda 3 (la lunghezza dei Trust Path)

La risposta alla domanda 3 sulla distribuzione delle lunghezze dei TP nella WoT non richiede in realtà la misura di t_{pf}, ma costituisce comunque un dato importante per capire la struttura di una WoT. In Figura 6.5 è riportato il grafico che mostra la distribuzione della lunghezza dei TP tra tutte le possibili $|WoT|^2$ coppie del grafo (si considerano anche le coppie (n_{src}, n_{dest}) con $n_{src} = n_{dest}$). Il grafico in particolare si riferisce ai test condotti su WoT_3, ma non presenta grandi differenze rispetto alle altre WoT.

A priori si potrebbe pensare che il grafico dovrebbe assumere valori esponenzialmente crescenti: all'aumentare della lunghezza del TP, si dovrebbero poter raggiungere sempre più identità. In realtà bisogna ricordare che il TP è definito come il percorso *minimo* da un'identità all'altra, dunque, visto il grande numero di archi presenti in una WoT, risulta difficile trovare identità talmente poco connesse da avere cammini minimi molto lunghi. Piuttosto, il numero di nodi raggiunti cresce esponenzialmente per $l \leq 5$, per poi decrescere esponenzialmente. Mentre i TP di $l = 6$ sono 355 milioni, sono ben più

[3]Le diverse WoT sono state scaricate da [Ced], dove è disponibile l'archivio storico dei file `.wot` che contengono i grafi delle WoT generati dal 2005 in poi.

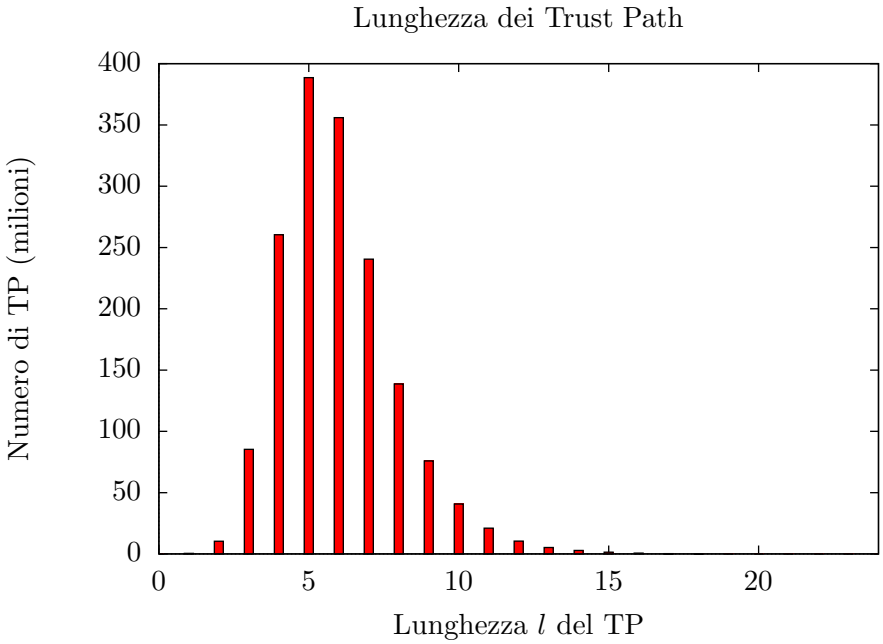


Figura 6.5: Distribuzione della lunghezza dei TP per WoT_3.

rari (soltanto 45) i TP di $l = 24$, e non vi sono TP con $l > 24$. Ovviamente, definito v_L come il numero totale di identità raggiungibili con cammini di lunghezze $l = 0, 1, .., L$, risulta v_L crescente, anche se non esponenzialmente (la WoT ha dimensioni finite, il numero di identità raggiungibili ha dunque un massimo uguale a |WoT|). Del grafico si nota anche un altro dato: la lunghezza per la quale vi è il massimo numero di percorsi è $l = 5$ (388 milioni), e con cammini di lunghezza $l \leq 6$ si comprendono in media i percorsi che permettono di raggiungere il 67% delle identità contenute nella WoT. Utilizzando una WoT come questa, probabilmente andrebbero considerati percorsi lunghi al massimo $4 \leq l_m \leq 6$, altrimenti si includerebbero porzioni troppo grandi della WoT. Per questo, negli esperimenti successivi, si darà una particolare attenzione alla performance di ricerche su percorsi di lunghezza $l \leq 6$, chiamata area *critica*, visto che nell'applicazione del modulo `wot` percorsi più lunghi sarebbero ignorati (dunque non verrebbero cercati).

6.2.2 Domande 4 e 5 (tempi di ricerca)

Prima di tutto bisogna considerare una caratteristica dei dati raccolti per t_{pf} durante i test: il tempo necessario alla visita del grafo è sostanzialmente proporzionale al numero di nodi visitati durante la ricerca, dunque è strettamente legato al funzionamento dell'algoritmo. In particolare, merge un

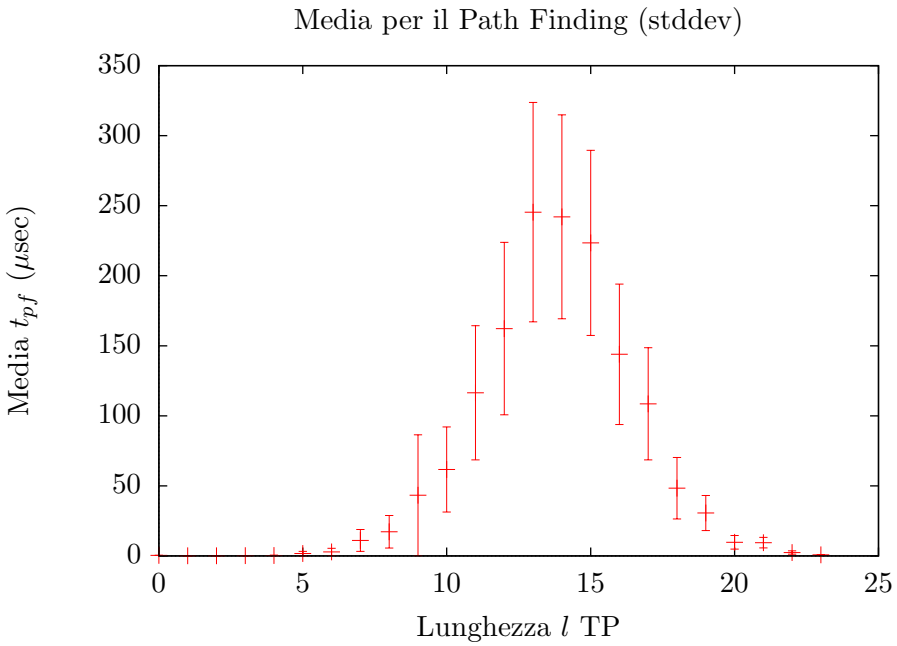


Figura 6.6: Variazione dei tempi di Path Finding rispetto alla lunghezza del TP.

dato fondamentale: usando la struttura dati in esame, l'algoritmo visita circa $17 \cdot 10^6$ nodi al secondo.

I tempi raccolti per l'esecuzione del Path Finding danno luogo al grafico in Figura 6.6. I risultati si riferiscono sempre alla WoT_3, e verranno messi a confronto con quelli delle altre WoT in seguito. Anche in questo caso non mancano risultati interessanti e in parte inaspettati. Guardando cosa avviene per $0 \leq l \leq 13$, si ha un risultato aspettato: t_{pf} cresce in modo esponenziale. Infatti, nell'ipotesi che tutti i nodi abbiano tra loro lo stesso numero m di archi uscenti ed entranti, il numero di nodi da visitare durante la ricerca è $O(m^{l/2})$, come risulta dalla sezione 5.2.

Questo dato sembra tuttavia essere sconfessato da ciò che avviene per $13 \leq l \leq 24$, dove t_{pf} è esponenzialmente decrescente. Vediamo il motivo di questo risultato. L'unico modo perché il percorso minimo sia molto lungo è che le identità del percorso siano poco connesse, in particolare abbiano pochi archi entranti, altrimenti probabilmente esisterebbero percorsi più brevi verso queste itentità. Tuttavia, in generale i nodi che hanno pochi archi entranti sono poco connessi in tutti i sensi, quindi hanno anche pochi archi uscenti, e questo riduce il numero di nodi che devono essere visitati per trovare il percorso durante le visite di BIDDFS. Un esempio della situazione è schematizzato in Figura 6.7.

Per quanto riguarda la parte *critica*, ovvero quella con $1 \leq l \leq 6$, che

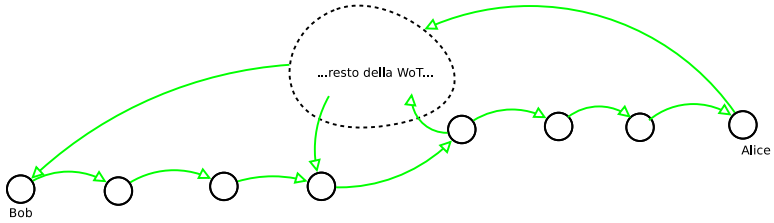


Figura 6.7: Esempio di TP molto lungo.

comprende il 67% dei percorsi, i valori di t_{pf} sono riportati nella tabella sotto. L'ultima colonna riporta la deviazione standard.

l	media t_{pf} (μsec)	Deviazione Standard σ (μsec)
1	84	98
2	88	44
3	170	263
4	303	501
5	1.669	3.173
6	2.849	5.196

Una misura importante è quella di quanto possano oscillare i dati (Domanda 5) discostandosi rispetto alla media. Nel grafico 6.6 sono riportati, per ogni valore della media, una misura di questa oscillazione. Si tratta della deviazione standard, che misura la dispersione dei dati intorno al valore medio. Come si vede dal grafico e dalla tabella sopra, l'ampiezza in cui possono ricadere questi dati è piuttosto grande. Questo comportamento è dovuto alla disomogeneità dei nodi della WoT (alcuni molto connessi, altri con pochi archi) e alla distanza tra il caso ottimo e il caso pessimo nella ricerca del TP. Nonostante questo, tutte le misure di t_{pf} sembrano limitate a valori molto bassi, specialmente per valori bassi di l.

L'oscillazione dei dati è resa in maniera più accurata nei grafici in Figura 6.8 e 6.9. I tempi di ricerca sono stati suddivisi in classi di frequenza. Nel grafico è resa, per ogni lunghezza del TP, la densità di ogni classe di frequenza, ovvero il numero di risultati per ogni classe (un'area più scura rappresenta un maggior numero di risultati che ricadono nella classe). Nei grafici è mostrata la grande differenza in termini di stabilità tra la ricerca per TP di diverse dimensioni: guardando la Figura 6.8 si vede che mentre i TP di lunghezza $1 \leq l \leq 6$ sono rappresentati da una sola linea blu, ad indicare che tutti i risultati ricadono in una sola classe, i TP di lunghezza maggiore hanno una classe con frequenza più alta che poi sfuma verso valori maggiori. Il grafico 6.9 è simile, ma riporta solamente i dati relativi ai percorsi di lunghezza $1 \leq l \leq 6$. Soltanto "zoomando" su questi percorsi si nota la distribuzione tra le classi (in questo grafico, le classi sono di ampiezza

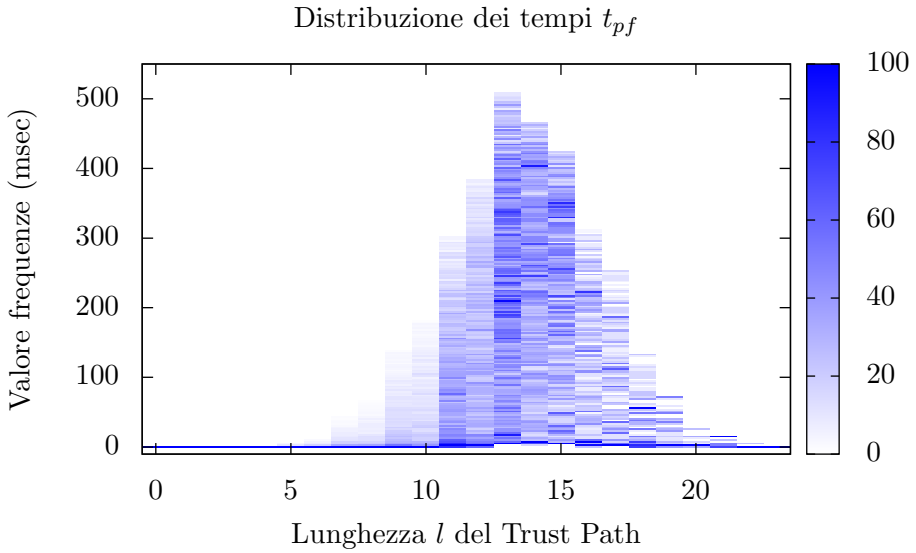


Figura 6.8: Distribuzione dei tempi di ricerca del TP (t_{pf}) per ogni lunghezza del TP (l).

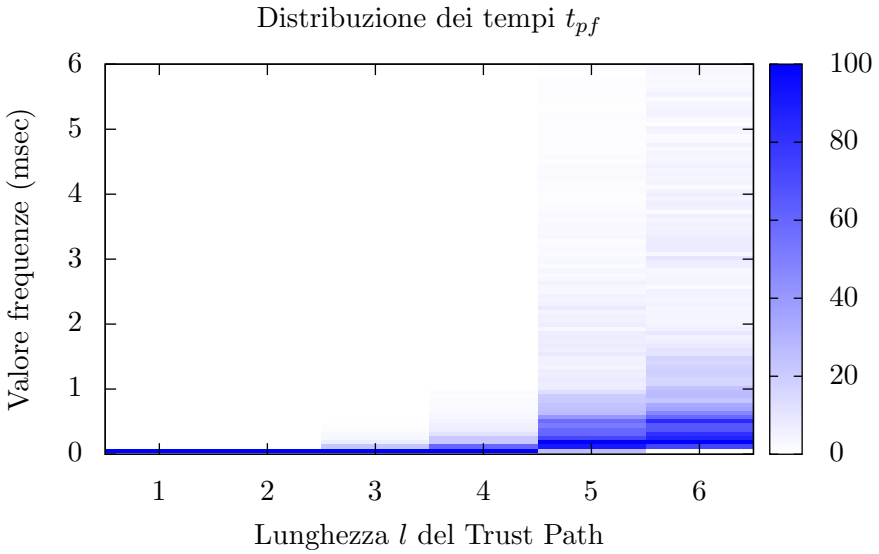


Figura 6.9: Distribuzione dei tempi di ricerca del TP (t_{pf}) per TP di lunghezza ($l \leq 6$).

minore rispetto all'altro, dunque l'analisi è più fine). Per ogni lunghezza, molte delle misure ricadono in poche classi centrali, mentre una parte sfuma verso classi di valore superiore.

6.2.3 Domanda 6 (dimensione della WoT)

Dopo i risultati incoraggianti trovati finora, rimane un dubbio. I test sono stati fatti sulla più grande WoT garantita crittograficamente esistente, ovvero quella di OpenPGP, che tuttavia ha attualmente meno di 45.000 utenti. È evidente che una WoT di un sistema anti-SPIT, per funzionare correttamente ed essere un efficace mezzo di prevenzione dello SPIT, dovrebbe contenere un numero ben maggiore di utenti. Il dubbio riguarda la scalabilità del sistema: se sono garantite delle performance buone per reti piccole, sarà possibile garantire performance analoghe con reti estese a tutti gli utenti del pianeta?

Per rispondere a questo dubbio sono stati confrontati i risultati del test principale (analisi del tempo medio per trovare un TP in funzione della sua lunghezza) svolto sulle tre WoT di OpenPGP WoT_1, WoT_2, WoT_3. I risultati sono riassunti nel grafico in Figura 6.10 e riproposti per $1 \leq l \leq 6$ in Figura 6.11. Come si vede, vi è una differenza notevole tra i risultati delle tre WoT. Sembra infatti che i tempi siano addirittura lineari rispetto al numero di utenti. Nella tabella che segue sono riportati come esempio i dati esatti relativi ai percorsi lunghi 4, 5, 6 e 10.

l	$\overline{t_{pf}}$ in WoT_1 (msec)	$\overline{t_{pf}}$ in WoT_2 (msec)	$\overline{t_{pf}}$ in WoT_3 (msec)
4	$0,232$	$0,252$	$0,303$
5	$0,857$	$1,283$	$1,669$
6	$1,363$	$2,243$	$2,850$
10	$28,245$	$46,054$	$61,661$

Questi risultati si spiegano con il numero n di nodi visitati durante le ricerche, che si riporta nella tabella seguente per $l = 4, 5, 6, 10$.

l	$\overline{V}$ in WoT_1	$\overline{V}$ in WoT_2	$\overline{V}$ in WoT_3
4	1.953	2.816	3.234
5	13.958	22.769	28.085
6	23.667	41.450	49.469
10	496.974	807.397	1.010.425

In effetti, a parità del numero medio di archi m, non vi dovrebbe essere alcuna differenza tra cercare un percorso lungo l in una rete di migliaia di elementi e una di miliardi di elementi.

Il motivo per cui dai dati sembra che cercare percorsi della stessa lunghezza in WoT più grandi richieda la visita di un numero maggiore di nodi

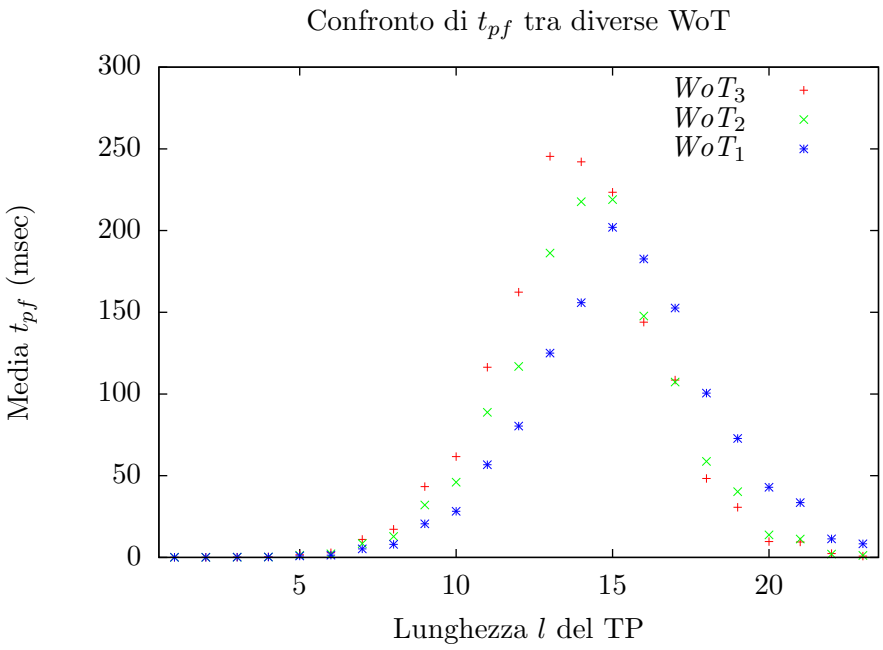


Figura 6.10: Confronto dei tempi di ricerca del TP per WoT di dimensione diversa.

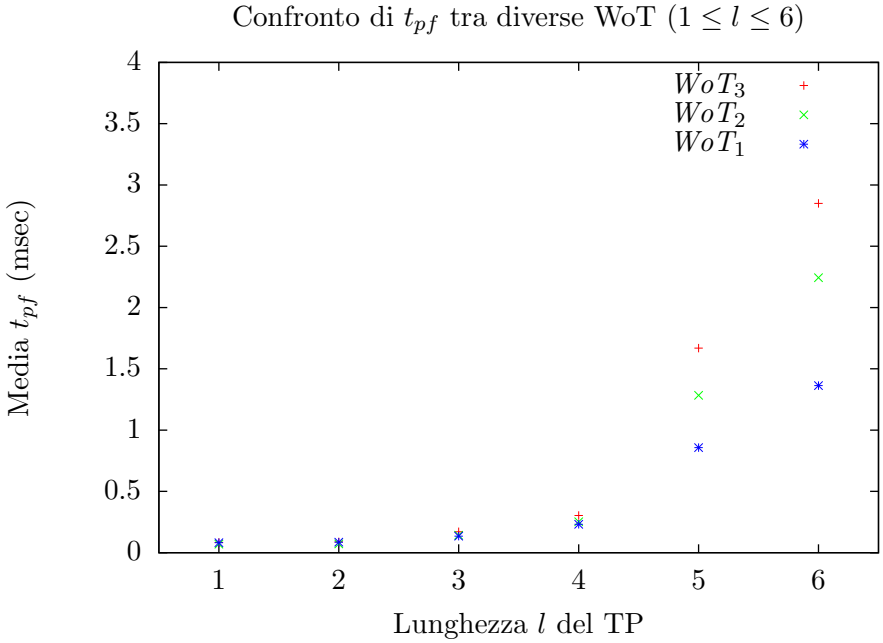


Figura 6.11: Confronto dei tempi di ricerca del TP per WoT di dimensione diversa, valori critici di l.

non è chiara e non trova una giustificazione teorica, per questo dovrebbe essere oggetto di ulteriori studi.

Purtroppo i dati a disposizione non sono sufficienti a stabilire sperimentalmente il comportamento dell'algoritmo all'aumentare della dimensione della WoT. Ad ogni modo, per quanto visto nella Sezione 5.2.1, dovrà essere rispettata la proprietà $V(m,l) = O(m^{l/2+1})$, quindi al crescere indefinito di $|WoT|$, n raggiungerà un massimo. In effetti, questi dati mostrano una coerenza con i risultati trovati nella Sezione 5.2.1.

Tuttavia, dimenticandosi la teoria e prendendo in esame solo i risultati sperimentali, ovvero la crescita di t_{pf} lineare rispetto alla dimensione della WoT, si deduce questo costituirebbe un collo di bottiglia. Ad esempio, interpolando i dati della tabella per $l = 6$, si avrebbe che per la ricerca di un TP di lunghezza 6 in una WoT di 100 milioni di utenti sarebbero necessari circa 10 secondi, non sarebbe dunque rispettato il requisito di tempo reale. Per quanto riguarda il possibile lavoro futuro per trovare conferma sperimentale alla teoria, si veda il Capitolo 7.

Il grafico 6.10 fornisce un altro risultato affascinante: mentre per $l \leq 13$ sembra che all'aumentare di $|WoT|$ aumenti t_{pf}, per $l \geq 16$ il risultato è ribaltato: in una WoT più grande, le ricerche per percorsi lunghi sono più veloci.

Come abbiamo visto nella Sezione 5.2, l'utilizzo in memoria di BIDDFS è $O(k)$, non presenta dunque un problema. Un ostacolo potrebbe invece essere rappresentato dalla dimensione della struttura dati su cui fare la ricerca. Bisogna infatti considerare che per fare le ricerche in modo efficiente, tutta la WoT viene caricata in memoria RAM e rappresentata tramite un grafo. Ogni nodo mantiene le seguenti informazioni:

- Il Key ID di 4 byte.
- Il timestamp (vedi Sezione 5.2.1); 4 byte.
- La stringa che rappresenta l'indirizzo SIP; 20 byte di media.
- Gli indirizzi di memoria dei nodi verso cui si ha una relazione di fiducia (FS); in un sistema a 32 bit $m \cdot 4 = 40$ byte.
- Analogamente, gli indirizzi della stella entrante (BS); 40 byte.
- La struttura base del nodo che mantiene i 3 indirizzi alle strutture dati sopra, escluso il KeyID ed il timestamp che sono memorizzati direttamente nella struttura base, essendo di lunghezza fissa; $4 \cdot 3 = 12byte$.

In totale, ogni nodo occupa 120 byte. Quindi, mentre una WoT di dimensioni contenute non presenta un problema - una WoT di 50.000 utenti occupa appena 6MB -, una WoT di 100 milioni di utenti occuperebbe 12 GB, e

difficilmente potrebbe essere utilizzata senza la memoria swap, o comunque potrebbe essere installata solo su macchine costose.

6.3 Conclusioni

Questo capitolo lascia dei problemi aperti. Alcuni pratici (*la crescita della WoT può portare a problemi di scalabilità rispetto al numero di utenti?*), altri del tutto teorici (*perché si ribaltano i valori nel grafico 6.10?*). Tuttavia, la teoria dimostra che esiste un limite piuttosto basso che non può essere superato con l'aumentare dei nodi della WoT.

Per quanto riguarda i tempi di esecuzione i dati sono comunque molto incoraggianti e l'algoritmo BIDDFS sembra ben strutturato, infatti per gli scenari individuati e per i quali sono stati svolti i test è assolutamente rispettato il requisito di tempo reale. Sui problemi relativi all'aumento dei tempi dovrebbero essere svolti dei test più accurati su reti più vaste (vedi Capitolo 7 per il possibile lavoro futuro), ma risulta comunque molto improbabile che la crescita dei tempi sia davvero lineare rispetto alla dimensione della WoT.

Ciò che preoccupa di più è lo spazio occupato in memoria dalla WoT, sul quale è difficile risparmiare senza perdere in performance, e che è destinato a crescere linearmente col numero di identità che fanno parte della WoT. Per mantenere l'informazione di una WoT molto grande sarebbe dunque necessaria una grande disponibilità di memoria oppure il trasferimento del grafo della WoT dalla memoria ad un database su disco fisso. Questo eliminerebbe nella pratica il problema della dimensione ma sicuramente rallenterebbe l'esecuzione della ricerca, anche se solo in modo lineare. Anche questa possibilità dovrebbe essere studiata come lavoro futuro.

Capitolo 7

Conclusioni e lavoro futuro

Si conclude la presentazione degli argomenti con una nota: W-BASA e il test anti-SPIT basato su WoT hanno delle enormi potenzialità. L'uso della WoT applicata alla prevenzione dello spam su VoIP è valida - specialmente se combinata con altre tecniche - e una sua diffusione potrebbe mettere in seria difficoltà chi fa SPIT e persino impedire la presentazione del problema. Alcuni aspetti sono da definire e probabilmente l'intero sistema ha bisogno di una base teorica più approfondita, tuttavia non ha niente che gli impedisca di andare in produzione e divenire un efficace mezzo di prevenzione dello SPIT.

Il percorso cominciato con questo lavoro può essere portato avanti proponendo altre sfide, di cui si fanno dei cenni.

Rimangono diversi problemi aperti, relativi soprattutto ai problemi su come gestire una WoT di dimensioni globali. A proposito, sarebbe da condurre uno studio sulle performance nel caso che il grafo della WoT risieda in un database piuttosto che in memoria, dato che questo approccio risolverebbe il problema della scalabilità, considerando che i database possono avere dimensioni praticamente illimitate.

Un altro dubbio irrisolto per mancanza di informazioni è l'aumento dei tempi di ricerca del TP con l'aumentare della dimensione della WoT. Un test su una più grande base di dati potrebbe essere svolto utilizzando come grafo quello di una qualsiasi rete sociale - non importa se garantita crittograficamente o meno - ad esempio una rete di conoscenze di un social network (in cui i nodi sono gli account e gli archi le relazioni di amicizia) oppure una rete basata sulle rubriche e-mail (in cui i nodi sono le caselle e-mail e gli archi rappresentano la presenza di un indirizzo nella rubrica). Questi esperimenti avrebbero il fine di confermare il risultato ottenuto dalla teoria, ovvero la limitazione del numero di nodi da visitare, e dunque dei tempi di ricerca.

7.1 Verso una Web of Trust decentralizzata?

Tutti questi dubbi sono derivati dalla scelta di centralizzare l'informazione riguardo alla WoT in un unico punto. La scelta è stata fatta per far sì che la ricerca potesse avvenire più velocemente, ma presenta i ben noti problemi di scalabilità rispetto al numero di utenti.

Un approccio alternativo potrebbe essere un sistema in cui l'informazione è distribuita. Illustrata per sommi capi, un'architettura del genere potrebbe funzionare come segue. Sui server di VoIP-SEAL non è presente l'intera WoT, ma soltanto - per ogni identità gestita dal server - due alberi:

L'albero di fiducia altrui (A_a) ovvero l'albero delle relazioni di fiducia *verso l'identità*, cioè delle persone che si fidano dell'identità.

L'albero di fiducia propria (A_p) ovvero l'albero delle relazioni di fiducia *dall'identità*, cioè delle persone di cui l'identità si fida.

Per non includere tutta la WoT, dovrebbe essere considerato solamente l'albero dei cammini minimi ad una certa profondità fissata p, dunque i due alberi conterrebbero di media $2 \cdot m^p$ identità. Gli alberi sono usati in questo modo: all'arrivo di una nuova chiamata da Alice a Bob, Alice deve dimostrare che Bob si fida di lui. Comincia così una fase di "handshake", in cui Alice e Bob cercano, rispettivamente nel proprio A_{a_alice} e A_{p_bob} un elemento comune ai due alberi. Se si trovasse un'identità M comune ai due alberi, significherebbe che M è un nodo di mezzo di un TP che comincia da Bob ed arriva ad Alice lungo massimo $2 \cdot p$ (dunque p dovrebbe essere scelto in funzione della lunghezza massima di un TP, in particolare, detta L questa lunghezza, dovrebbe essere $p = \frac{L}{2}$).

Naturalmente, Bob non può fidarsi delle relazioni di fiducia che Alice sostiene di avere, quindi queste informazioni dovrebbero essere garantite crittograficamente. Un modo interessante potrebbe essere la collezione, da parte di Alice, di una serie di *attestazioni*. Un'attestazione è un'informazione del tipo *C si fida di A* firmato con la chiave privata di C, che chiunque può verificare usando la chiave pubblica di C. L'albero A_{a_alice} è quindi in realtà un albero in cui ad ogni arco è associata un'attestazione. Alice invia a Bob le attestazioni che dimostrano il percorso di fiducia dall'identità M ad Alice, per dimostrare che esiste il percorso di fiducia. Una sessione di "handshake" potrebbe avere le seguenti fasi:

1. Alice invia un messaggio di `INVITE` a Bob, con scritto: "Si fidano di me con un TP di $l = 1$le identità M, N, O. Tu ti fidi di qualcuna di queste?"
2. Bob $\longrightarrow$ Alice: "Non ne conosco nessuna, inviane altre."
3. Alice $\longrightarrow$ Bob: "Si fidano di me con un TP di $l = 2$ le identità P, Q, R. Tu ti fidi di qualcuna di queste?"

4. Bob $\longrightarrow$ Alice: "Si, io mi fido di R con un TP lungo $l = 3$. Dimostrami che R si fida di te."

5. Alice $\longrightarrow$ Bob: invio delle attestazioni: "R si fida di P", "P si fida di Alice".

6. Bob verifica le 3 attestazioni rispettivamente con le chiavi pubbliche di R e P. Se sono corrette, stabilisce la comunicazione.

Ovviamente, nelle azioni di sopra per Alice e Bob non si intende i terminali VoIP ma piuttosto i server SIP che gestiscono la transazione in modo invisibile ai terminali. Le azioni di Alice dovrebbero quindi essere svolte dal proprio SP - che adesso non si limita a firmare il messaggio, ma deve anche gestire l'handshake -, mentre quelle di Bob dal server VoIP-SEAL da cui Bob è protetto.

Per un sistema così strutturato, la parte più complessa sarebbe probabilmente la soluzione dei problemi legati all'aggiornamento degli alberi - problemi, per altro, comuni a tutti i database distribuiti. Sarebbe inoltre da valutare il tempo di completamento di un'operazione come l'handshake. Tuttavia un approccio del genere risolverebbe ogni problema di scalabilità, evitando punti di centralizzazione dell'informazione, dato che ogni server dovrebbe mantenere soltanto le informazioni riguardanti i propri utenti, e forse il sistema nel complesso sarebbe anche più stabile.

Appendici

Appendice A

Il protocollo SIP

Dato che il server VoIP-SEAL fa parte dell'architettura SIP e gestisce messaggi SIP, si presenta per linee molto generali e grossolane il funzionamento del protocollo definito in [RSC+02]. Per una trattazione approfondita si consiglia [SJ01]. L'acronimo SIP (Session Initiation Protocol) sta per Protocollo di Inizializzazione della Sessione, il ruolo del protocollo infatti non è di trasportare i dati di una comunicazione, ma soltanto di mettere in comunicazione un certo numero di utenti per una qualunque applicazione che ha bisogno di una sessione. Per "sessione" si intende un contesto temporalmente limitato all'interno del quale avviene lo scambio di dati tra gli utenti. Sebbene SIP sia utilizzato principalmente per la telefonia Internet (per la quale una sessione corrisponde ad una telefonata), in realtà può essere utilizzato per molte altre applicazioni, come lo streaming video (in cui una sessione è una videoconferenza), giochi multi-giocatore (in cui una sessione è una partita), e così via.

Le funzioni fondamentali di SIP sono:

- Localizzare gli utenti, cioè tradurre un indirizzo SIP nel formato `utente@dominio` in una tripla (indirizzo IP, protocollo, porta) per permettere la connessione diretta con il destinatario.
- L'invito di un utente ad una sessione, che nel caso del VoIP corrisponde all'inizio di una telefonata. L'invito avviene tramite un messaggio di tipo `INVITE`.
- I messaggi di tipo `INVITE` comprendono anche un messaggio SDP (Session Description Protocol), definito in [HJ98], che contiene le informazioni per stabilire la sessione, come l'elenco dei protocolli utilizzabili per la sessione.
- L'indicazione della fine di una sessione, nel VoIP la fine di una telefonata, tramite il messaggio `BYE`.

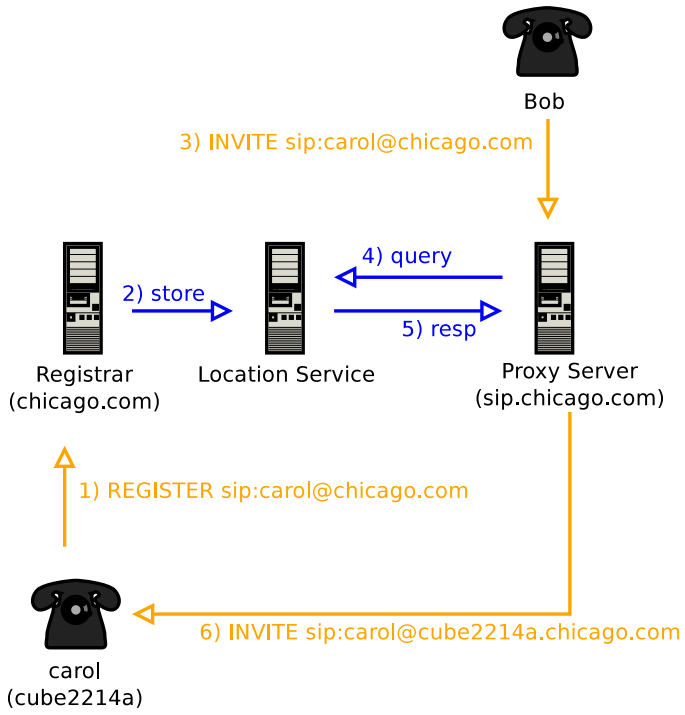


Figura A.1: Esempio di scambio di messaggi SIP per l'invio di una richiesta `INVITE`. In arancione i messaggi definiti dallo standard SIP, in blu i messaggi nel formato della particolare implementazione dei server Registrar e Proxy.

L'architettura SIP è scalabile e mutua molti concetti dal routing su Internet e dal protocollo HTTP. Nelle architetture SIP sono presenti 4 elementi del sistema, chiamati Transaction Users: gli User Agent (ovvero gli endpoint della comunicazione, in VoIP i telefoni coinvolti nella chiamata), i server Registrar (a cui gli utenti SIP segnalano la propria raggiungibilità) e i Proxy Server (che si occupano di ricevere i messaggi SIP dagli User Agent o da altri proxy, stabilirne la legittimità e instradarli verso la destinazione). L'architettura è peer to peer, infatti gli User Agent potranno funzionare sia da client che da server. Ad esempio, un telefono SIP svolge la funzione del client quanto invia un messaggio `INVITE`, e del server quando lo riceve.

Un tipico esempio di scambio di messaggi, estratto da [RSC+02], è riprodotto in Figura A.1. Si riferisce all'invito ad una sessione rivolto a Charlie da parte di Bob. Corrisponde all'azione svolta da Bob di chiamare Charlie al proprio telefono VoIP. In precedenza, Charlie ha segnalato la sua presenza (ovvero, la corrispondenza tra il proprio indirizzo SIP e l'indirizzo internet a cui l'utente è raggiungibile) al proprio server Registrar (messaggio 1), che ha salvato l'informazione sul Location Service (messaggio 2) con un metodo non definito da SIP. Nel momento in cui Bob telefona a Charlie, il suo messaggio `INVITE` viene spedito al Proxy Server responsabile del dominio, il

quale interroga il Location Service per ottenere l'indirizzo di Charlie, verso il quale il Proxy Server inoltra il messaggio di `INVITE` destinato a Charlie. La reazione da parte di Charlie alla ricezione della richiesta `INVITE` è l'invio di un messaggio `ACK` di risposta, grazie al quale può cominciare lo scambio di dati vero e proprio tra Charlie e Bob. Lo scambio dei dati avviene al di fuori del contesto SIP, ma al suo termine verrà inviata da una delle due parti un messaggio SIP `BYE` che indica la fine della sessione.

Appendice B

Acronimi utilizzati

Per facilitare la lettura, è riportata una lista degli acronimi utilizzati durante l'esposizione della relazione.

Istituti, gruppi di lavoro, cartelli

IETF: Internet Engeneering Task Force
IEEE: Institute of Electrical and Electronics Engineers
3GPP: Third Generation Protocol Project
OMA: Open Mobile Alliance
RFC: Request for Comments
Gruppo RTC: Real-time Communications

Reti

GPS: Global System for Mobile Communications
PSTN: Public Switched Telephone Network (rete pubblica a commutazione di circuito)
GPRS: General Packet Radio Service

Standard, protocolli

ADSL: Asymmetric Digital Subscriber Line
SIP: Session Initiation Protocol
PGP: Pretty Good Privacy
SMTP: Simple Mail Transfer Protocol

Telefonia via Internet, VoIP-SEAL

VoIP: Voice over IP (telefonia su Internet)
SPIT: Spam over Internet Telephony (spam inviato tramite telefonia su Internet)

W-BASA: Web of trust-Based Anti-Spit Architecture (Architettura anti-SPIT basata su Web of Trust)
SP: Signing Proxy

Reti di fiducia

WoT: Web of Trust (rete di fiducia)
SCS: Strongly Connected Set (insieme fortemente connesso)
TP: Trust Path (percorso di fiducia)
BS: Backward Star (stella entrante)
FS: Foreward Star (stella uscente)

Algoritmi

MD5: Message Digiest 5
BFS: Breadth-First Search
DFS: Depth-First Search
IDDFS: Iterative Deepening Depth-First Search
BIDDFS: Bidirectional Iterative Deepening Depth-First Search

Bibliografia

[ACD+07] E. Allman, J. Callas, M. Delany, M. Libbey, J. Fenton, and M. Thomas. *DomainKeys Identified Mail (DKIM) Signatures*. RFC 4871 (Proposed Standard), May 2007. URL `http://www.ietf.org/rfc/rfc4871.txt`.

[CDF+07] J. Callas, L. Donnerhacke, H. Finney, D. Shaw, and R. Thayer. *OpenPGP Message Format*. RFC 4880 (Proposed Standard), November 2007. URL `http://www.ietf.org/rfc/rfc4880.txt`.

[CdNS09] Marco Cornolti, Nico d'Heureuse, Saverio Niccolini, and Jan Seedorf. (NEC Laboratories Europe (Heidelberg, Germany), Università di Pisa (Pisa, Italy)). *Detecting Trustworthy Real-Time Communications using a Web-of-Trust*, 2009. Under Submission.

[Ced] Jörgen Cederlöf. *Wotsap*. URL `http://www.lysator.liu.se/~jc/wotsap/index.html`.

[Ced07] Jörgen Cederlöf. *The Web of Trust .wot file format, version 0.2*, November 2007. URL `http://www.lysator.liu.se/~jc/wotsap/wotfileformat.txt`.

[HJ98] M. Handley and V. Jacobson. *SDP: Session Description Protocol*. RFC 2327 (Proposed Standard), April 1998. Obsoleted by RFC 4566, updated by RFC 3266. URL `http://www.ietf.org/rfc/rfc2327.txt`.

[QNTS08] Juergen Quittek, Saverio Niccolini, Sandra Tartarelli, and Roman Schlegel. *On Spam over Internet Telephony (SPIT) Prevention*, 2008.

[RJ08] J. Rosenberg and C. Jennings. *The Session Initiation Protocol (SIP) and Spam*. RFC 5039 (Informational), January 2008. URL `http://www.ietf.org/rfc/rfc5039.txt`.

[RS03] J. Rosenberg and H. Schulzrinne. *An Extension to the Session Initiation Protocol (SIP) for Symmetric Response Rou-*

ting. RFC 3581 (Proposed Standard), August 2003. URL `http://www.ietf.org/rfc/rfc3581.txt`.

[RSC+02] J. Rosenberg, H. Schulzrinne, G. Camarillo, A. Johnston, J. Peterson, R. Sparks, M. Handley, and E. Schooler. *SIP: Session Initiation Protocol*. RFC 3261 (Proposed Standard), June 2002. Updated by RFCs 3265, 3853, 4320, 4916, 5393. URL `http://www.ietf.org/rfc/rfc3261.txt`.

[Rus05] Roland Rust. (Center for Excellence in Service at the University of Marylands Robert H. Smith School of Business and Rockbridge Associates, Inc.). *Spam Costs U.S. Nearly $22 Billion Annually*, February 2005. URL `http://www.rockresearch.com/news_020305.php`.

[SJ01] Henry Sinnreich and Alan B. Johnston. *Internet Communications Using SIP*. Wiley, 2001. ISBN: 978-0-471-41399-8.

Ringraziamenti

Per questa tesi ringrazio il mio professore, Luca Deri, che mi ha procurato il contatto per fare il tirocinio ad Heidelberg e ha letto la mia tesi da cima a fondo (o almeno così dice).

Ringrazio i colleghi che ho conosciuto ad Heidelberg: il mio tutore aziendale, Saverio Niccolini, e i miei supervisori, Jan Seedorf e Nico d'Heureuse, con i quali sviluppare software diventa una sfida piacevole e avvincente, i membri del gruppo Real-Time Communications, che sono sempre stati pronti ad aiutarmi, Michele, Christoph, Donato, Felix, Nadja, Emiliano e gli altri studenti conosciuti alla NEC, con cui ho condiviso ben più di un ufficio.

Ringrazio le comunità di sviluppo di tutti i software liberi che ho usato per questa tesi: LaTeX, `gnuplot`, GNU/Linux, e molti altri.

Ringrazio gli amici che ho avuto attorno in Germania e che mi hanno quasi convinto a rimanere là. Ringrazio i miei amici italiani che mi hanno convinto a tornare.

Ringrazio i fratelli e le sorelle di Rebeldìa, perché mi hanno dato un motivo per studiare e credere in un mondo migliore, e quelli del Cervinistan, perché è dalle piccole cose che nascono quelle grandi.

Ringrazio la mia famiglia, perché mi ha trasmesso l'amore per il sapere.

Ringrazio tutte le stelle uscenti ed entranti, e in particolare quella che è entrata nella mia vita.

Dedico il mio lavoro ad Alice e Bob, e a tutti quelli che vogliono soltanto comunicare in pace.

Dedico il mio lavoro ad Eve e Spad, perché sono sicuro che c'è del buono anche in loro.