Università degli
Studi di L'Aquila

Facoltà di Ingegneria

Corso di Laurea in Ingegneria Elettronica

Corso di Reti di calcolatori
del Prof. Gabriele Di Stefano

OpenPGP, PGP e GPG

studente: Mariano Spadaccini
matricola: 140769

Indice

[1] tratto da `http://www.philzimmermann.com/it/index.html`

Capitolo 1

Introduzione

Un messaggio email scambiato fra due siti molto distanti fra loro transita tipicamente attraverso diverse macchine prima di giungere a destinazione. Ognuna di queste macchine può potenzialmente leggere il messaggio e registrarlo: al contrario di quanto comunemente si pensa, sostanzialmente *la privacy non esiste*. Tuttavia si vorrebbero inviare email leggibili solo dal destinatario legittimo e da nessun altro. Questo desiderio ha stimolato persone e gruppi ad applicare alla posta elettronica i principi propri della crittografia per ottenere email sicure.
Uno dei sistemi più diffusi per ottenere email sicure è utilizzare i principi inclusi nell'**OpenPGP**, standard descritto nel *RFC 2240*[1].
OpenPGP deriva dal famoso **PGP**, frutto del lavoro di **Phil Zimmermann**. Il *PGP*[2] è un pacchetto completo per la sicurezza della posta elettronica in quanto fornisce *privacy*, *autenticazione*, *firma digitale* e *compressione*, completamente semplice da usare. Oggi *PGP* è largamente usato per le sue doti di qualità e disponibilità per diverse piattaforme, tra le quali *Linux*, *Unix*, *MS Windows* e *Mac OS*.

[1] datato Novembre 1998
[2] introdotto nel 1991

Capitolo 2

Un po' di storia

Prima dell'avvento di *PGP*, Phil Zimmermann era scontento della sicurezza che offrivano i pacchetti esistenti; a tal proposito creò il famoso *PGP*, acronimo di **Pretty Good Privacy**, il quale semplicemente condensava i migliori algoritmi crittografici esistenti affinché il pacchetto fornisse una sicurezza degna di tale nome.
Nel 1991 rese pubblico su Internet i dettagli del nuovo prodotto crittografico, ma ciò rese Zimmermann oggetto di un'indagine criminale durata tre anni perché il governo sostenne che le restrizioni all'esportazione statunitense di software crittografico erano state violate quando PGP si diffuse nel mondo.
Nonostante la persecuzione del governo, *PGP* divenne il software di codifica per la posta elettronica più usato al mondo. Dopo che il governo chiuse l'indagine all'inizio del 1996, Zimmermann fondò la *PGP Inc.*
L'ironia vuole che a seguito delle tante controversie che hanno caratterizzato la nascita di *PGP*, oggi ne sono in circolazione (con nomi diversi) più versioni, talune incompatibili tra loro.
Tra queste, si cita la **GnuPG**, acronimo di **Gnu Privacy Guard**, software prodotto con licenza GPL, quindi **software libero**, attualmente il software nato dalla filosofia *PGP*, più utilizzato tra i diretti concorrenti.
Interessante è l'altra questione che *PGP* ha affrontato, in quanto PGP è legato alla violazione dei brevetti. Difatti, l'azienda che deteneva il brevetto per *RSA*, la *RSA Security Inc.*, dichiarò che l'uso dell'algoritmo *RSA* in *PGP* violava il suo brevetto; nei primi tempi ci furono anche dei problemi con un altro algoritmo crittografico brevettato, *IDEA*.
La sua natura *open source* e tutte le difficoltà incontrate, stimolarono gruppi e singoli individui a cimentarsi in modifiche di *PGP*: alcune per aggirare leggi sul traffico di armi, altre per evitare l'uso di algoritmi brevettati e altre ancora per far diventare *PGP* un prodotto commerciale e prorpietario, non più open source. La versione di PGP che segue è focalizzata sul PGP classico,

nella sua versione più semplice e antica.

Capitolo 3

Le funzionalità

PGP utilizza di proposito algoritmi crittografici esistenti, al posto di inventarne di nuovi. Gli algoritmi utilizzati, infatti, hanno superato esaustive ricerche da parte di esperti del settore e non sono stati influenzati da nessuna agenzia governativa che poteva avere interesse ad indebolirli[1].
PGP supporta la *compressione del testo*, la *segretezza*, la *firma digitale*, e contiene funzioni evolute per la *gestione delle chiavi*; potrebbe sembrare strano, ma propriamente non ha funzioni per la gestioni delle email: si tratta più che altro di un preprocessore, che prende in input un testo in chiaro e produce in output un testo cifrato in *base64*. Naturalmente questo output può essere inviato per email.
Come ultimo passaggio, alcune implementazioni di *PGP* prevedono la chiamata ad un *agente* che invia il messaggio.

[1] per quelle persone che tendono a non fidarsi del governo, questa proprietà è considerata molto positivamente

Capitolo 4

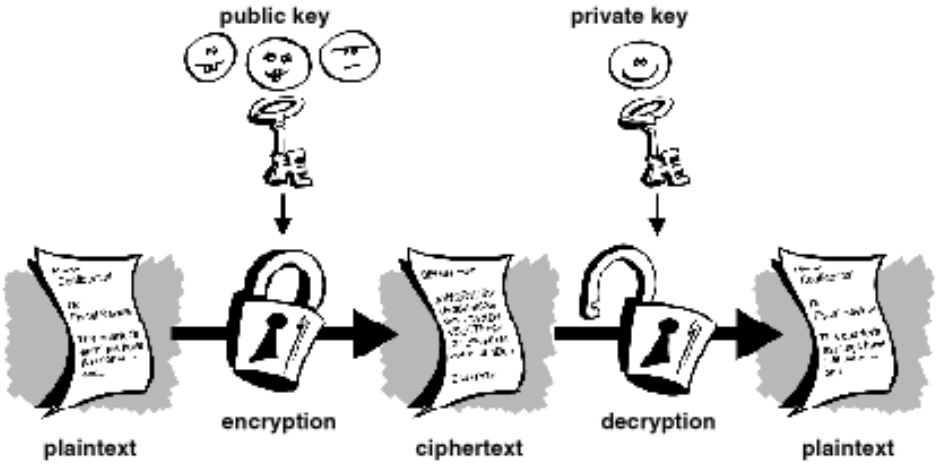
Gli algoritmi utilizzati

Prima di analizzare il funzionamento di *PGP* sarà utile introdurre alcuni algoritmi utilizzati al suo interno.

4.1 RSA

RSA è frutto della ricerca di un gruppo del *M.I.T.* e prende il nome dalle iniziali dei suoi tre ideatori ***R****ivest*, ***S****hamir* e ***A****dleman*. È sopravvissuto a tutti i tentativi di forzatura per più di un quarto di secolo, ed è considerato un algoritmo molto robusto. Il suo maggior svantaggio è che richiede chiavi di almeno 1.024 bit[1], il che lo rende abbastanza lento, ma rimane comunque l'algoritmo a chiave pubblica più diffuso.

Analizziamo ora i principi su cui si basa quest'algoritmo; inizialmente si



[1] contro i 128 bit degli algoritmi a chiave simmetrica

eseguono i seguenti passaggi:

1. si scelgono due numeri primi, p e q (tipicamente di 1024 bit);
2. si calcola $n = pq$ e $z = (p-1)(q-1)$;
3. si sceglie un numero primo di z, definito d;
4. si trova e tale che de mod z $= 1$.

Si suddivide il testo in chiaro (considerato come una stringa di bit), in modo che ogni messaggio in chiaro, P, cada nell'intervallo $0 \leq |P| < n$. Per fare questo, raggruppiamo il testo in chiaro in blocchi di k bit, in cui k è il più grande intero per cui vale $2^k < n$.
Per cifrare il messaggio P, calcoliamo $C = P^e \bmod n$). Per decifrare C, calcoliamo $P = C^d \bmod n$. Si può dimostrare che per ogni P nell'intervallo specificato, le funzioni di cifratura e decifrazione sono una l'inversa dell'altra. Per cifrare è necessario disporre di e e di n, per decifrare occorrono d e n. Quindi, la chiave pubblica consiste nella coppia (e, n), mentre la chiave privata consiste in (d, n). La sicurezza del metodo è basata sulla difficoltà di scomporre in fattori i numeri molto grandi. Se il criptoanalista riuscisse a fattorizzare il numero *noto* n, allora riuscirebbe a trovare anche p e q, e da questi anche z; attraverso la conoscenza di z, si possono trovare e e d tramite l'algoritmo di Euclide. Fortunatamente i matematici si occupano della fattorizzazione dei grandi numeri da più di 3000 anni, e, dalla conoscenza acquisita, si assume che questo sia un problema *difficile*.
Fattorizzare un numero di 500 cifre usando la forza bruta richiederebbe (disponendo della tecnologia attuale) un tempo improponibile, anche assumendo di usare il miglior algoritmo noto. Se i computer continueranno ad aumentare la loro velocità di un ordine di grandezza sarà sufficiente scegliere p e q più grandi.

4.2 MD5

MD5 è una delle funzioni *message digest*[2] più utilizzata; come tutte le funzioni della suddetta categoria, *MD5* ha le seguenti proprietà importanti:

- dato P, è facile calcolare *MD5(P)*;
- dato *MD5(P)*, è *difficile* trovare P;

[2] riassunto del messaggio

- dato *P*, è *difficile* trovare *P'*, tale che *MD5(P') = MD5(P)*;
- se l'input cambia anche di 1 bit, l'output diventa completamente diverso.

MD5 è il quinto di una serie di **message digest** progettati da *Ronald Rivest*. *MD5* mescola i bit d'ingresso in una maniera sufficientemente complessa da ottenere che ogni bit di uscita sia influenzato da tutti i bit di ingresso. In sintesi, l'algoritmo inizia riempiendo il messaggio fino ad una lunghezza di 448 bit modulo 512. Il valore della lunghezza originale del messaggio è aggiunto sotto forma di intero a 64 bit, per raggiungere un input di lunghezza 512 bit, e si inizializza a un valore prefissato un buffer di 128 bit.
A questo punto comincia il calcolo vero e proprio: ogni iterazione prende dall'input un blocco di 512 bit, che è profondamente alterato usando il buffer a 128 bit. Come extra, si aggiunge anche una tabella costruita con la funzione sin. L'uso di una funzione ben conosciuta come il *seno*, al posto del generatore di numeri casuali, serve per allontanare qualunque sospetto che l'autore abbia inserito delle *backdoor sofisticate*, che solo lui riesce ad usare. Per ogni blocco di input si effettuano quattro iterazioni, e il processo prosegue fino a che tutti i blocchi sono stati elaborati. Il contenuto del buffer a 128 bit costituisce il *message digest*.

4.3 Codifica Base64

La codifica **Base64** è ampiamente utilizzata per la codifica dei messaggi da trasportare secondo il formato *MIME*[3]. La codifica *Base64*, a volte definita *armatura ASCII*, prevede l'utilizzo di gruppi di 24 bit suddivisi in quattro unità da 6 bit; ogni unità è considerata come carattere ASCII legale. La codifica è *A* per 0, *B* per 1 e così di seguito, seguita dalle 26 lettere minuscole, dalle dieci cifre e infine da + e /, rispettivamente 62 e 63. Le sequenze == e = indicano che l'ultimo gruppo contiene, rispettivamente, solo 8 o 16 bit. I ritorni a carrello e gli avanzamenti riga sono ignorati, pertanto possono essere inseriti a piacere per rispettare la lunghezza delle righe. Con questo schema si può inviare in sicurezza un testo binario arbitrario.

4.4 IDEA: evoluzione del DES

IDEA (***I**nternational **D**ata **E**ncryption **A**lgorithm*) è nato nel 1991 sotto il nome di *IPES* (***I**mproved* P*roposed **E**ncryption **S**tandard*), ed è stato pro-

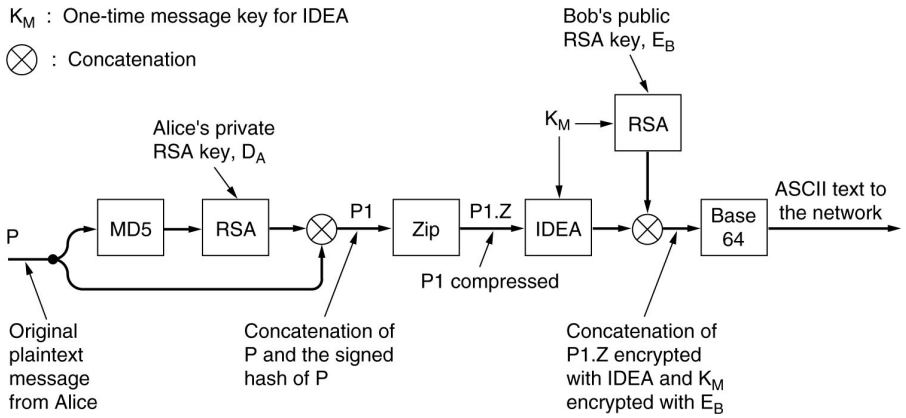
[3]Multipurpose Internet Mail Extensions

gettato da due famosi ricercatori in Svizzera: *Xuejja Lai* e *James L. Massey*. Come il **DES**, è un codice cifrato a blocchi di 64 bit; la differenza è che questa volta la chiave è di 128 bit, che dovrebbe eliminare qualsiasi possibilità di riuscita di ricerca della chiave procedendo per tentativi: le chiavi possibili sono infatti 2^{128}. *IDEA* è al momento il cifrario a chiave segreta più utilizzato per quanto riguarda i software commerciali di crittografia considerando la sua velocità di codifica e decodifica nonché la sua elevata sicurezza.

Capitolo 5

Il funzionamento

Per vedere come funziona *PGP*, consideriamo il seguente esempio.
Alice vuole inviare a Bob in modo sicuro un messaggio in chiaro firmato, P. Alice e Bob hanno entrambi una chiave *RSA* privata (D_x) e una pubblica (E_x). Ipotizziamo che ognuno dei due conosca la chiave pubblica dell'altro.



Alice inizia lanciando il programma *PGP* sul suo computer. *PGP* calcola innanzitutto l'*hash* del messaggio P usando *MD5*, e poi cifra il risultato usando la chiave *RSA* privata D_A.
Quando Bob riceve il messaggio, può decifrare l'*hash* usando la chiave pubblica di Alice e quindi verificarne la correttezza.
La robustezza di *MD5* garantisce che una terza persona (per esempio Trudy), capace di intercettare l'*hash* e decifrarlo con la chiave pubblica di Alice, non sarà in grado di produrre un messaggio con lo stesso *hash MD5* (è un problema computazionalmente troppo difficile).

L'*hash cifrato* e il messaggio originale sono concatenati in un singolo messaggio *P1*, e compressi usando il programma *ZIP*, che usa l'algoritmo di *Ziv-Lempel*. Chiamiamo l'output di questo passo *P1.Z*.
Successivamente *PGP* chiede ad Alice di generare un numero casuale in input. Il contenuto del messaggio e la velocità di battitura sono usati per generare una chiave di messaggio *IDEA* a 128 bit, K_M[1]. K_M serve a cifrare *P1.Z* con *IDEA* in modalità *chipher feedback*; inoltre K_M è cifrata con la chiave pubblica di Bob, E_B; queste due componenti sono concatenate e convertite in *Base64*. Il messaggio risultante contiene solo lettere, cifre e i simboli+, /, =, quindi si può inserire in un *body* conforme a *RFC 822* per inviarlo a destinazione senza temere alterazioni durante il transito.
Quando Bob riceve il messaggio, esegue a rovescio la codifica *base64* e decifra la chiave *IDEA* usando la sua chiave privata.
Con essa decifra il messaggio e ottiene *P1.Z*. Quindi passa alla decompressione, separa il testo in chiaro dall'*hash cifrato* e decifra l'*hash* usando la chiave pubblica di Alice.
Adesso Bob controlla se l'*hash* del testo in chiaro coincide con il valore del suo calcolo *MD5*: in caso affermativo, sa con certezza che P è il messaggio originale inviato da Alice.
È utile notare che *RSA* si usa solamente in due momenti: per cifrare l'*hash MD5* a 128 bit e per cifrare la *chiave IDEA* a 128 bit. Anche se *RSA* è lento, deve cifrare solamente 256 bit, quindi una quantità limitata di dati. D'altra parte, tutti i 256 bit cifrati con RSA sono altamente casuali, quindi Trudy dovrebbe fare una gran quantità di lavoro già per stabilire se ha indovinato correttamente una chiave. La cifratura della maggior parte dei dati è fatta usando *IDEA*, che è diversi ordini di grandezza più veloce di *RSA*. Quindi *PGP* fornisce sicurezza, compressione e firma digitale, e lo fa in modo molto più efficiente dello schema che prevede l'utilizzo della crittografia a chiave pubblica.

PGP supporta quattro diverse lunghezze di *chiavi RSA*, così l'utente può scegliere quella che ritiene più appropriata:

- *Casual* (384 bit): può essere forzata facilmente;
- *Commercial* (512 bit): può essere forzata dalle ben note organizzazioni governative;
- *Military* (1.024 bit): non può essere forzata da nessuno;

[1] è chiamata *chiave di sessione* nella letteratura *PGP*, ma in realtà il nome non è appropriato, in quanto non esiste una sessione

- *Alien* (2.048 bit): non può essere forzata da nessuno, neanche in altri pianeti.

Considerando che *RSA* è utilizzato solo in due piccoli calcoli, tutti dovrebbero sempre usare il livello *alien*.

Capitolo 6

Il formato

Il formato *PGP* classico è mostrato in Figura 6.1, ma si usano numerosi altri formati. Il messaggio *PGP* ha tre parti, che contengono la chiave *IDEA*, la *firma* e il *messaggio*. La parte con la chiave contiene anche un identificatore della chiave, considerando che gli utenti possono utilizzare più chiavi pubbliche.

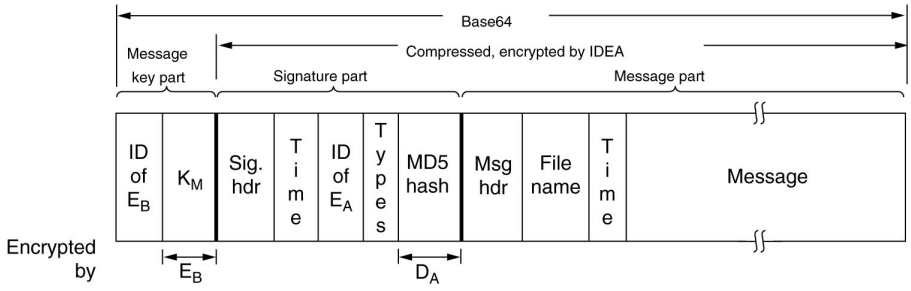


Figura 6.1: Classico formato PGP

La parte del messaggio *PGP* che contiene la firma è composta da un'*intestazione*, da un *timestamp*, dall'*identificatore* della chiave pubblica utilizzata dal mittente[1], da alcune informazioni che identificano gli algoritmi usati[2], e infine dall'*hash cifrato*.
La parte contente il messaggio è formata da *intestazione*, nome di default per il file (nel caso che il destinatario voglia scriverlo su disco), *timestamp* relativo alla creazione del messaggio, e infine dal *messaggio*.

[1] serve per decifrare la *firma hash*
[2] per consentire un eventuale utilizzo di versioni successive di *MD5* o *RSA*

Capitolo 7

La gestione delle chiavi

La gestione delle chiavi ha ricevuto una gran quantità di attenzioni durante lo sviluppo di *PGP*, poiché rappresenta il *tallone d'Achille* di tutti i sistemi di sicurezza. Ogni utente gestisce localmente due strutture dati: un insieme di chiavi private e un insieme di chiavi pubbliche. L'insieme di chiavi private (*portachiavi privato*) contiene una o più coppie personali di chiavi private e pubbliche. Per ogni utente sono supportate coppie multiple di chiavi, per consentire agli utenti di cambiare le loro chiavi pubbliche con cadenza periodica o quando pensano che una chiave sia compromessa, senza che questo annulli la validità dei messaggi in preparazione o in transito.
A ogni coppia di chiavi è associato un'identificatore, così il mittente può comunicare al destinatario quale chiave pubblica a usato per cifrare il messaggio. Gli identificatori di messaggio sono formati dai 64 bit di ordine più basso della chiave pubblica, e gli utenti hanno la responsabilità di evitare possibili conflitti fra gli identificatori delle loro chiavi pubbliche. Le chiavi private sono memorizzate su disco in forma cifrata utilizzando una *password* di lunghezza arbitraria, per proteggerle contro eventuali furti.
Per capire meglio il problema della gestione delle chiavi, consideriamo l'esempio in cui le chiavi pubbliche sono gestite da un *bullettin board*. Trudy, per leggere le email segrete di Bob, potrebbe attaccare la *bullettin board* e rimpiazzare la chiave pubblica di Bob con una di sua scelta. Quando più tardi Alice legge la chiave che crede appartenere a Bob, Trudy può far scattare un attacco di tipo *bucket brigate*[1] con Bob.
Per prevenire questo tipo di attacchi, o comunque minimizzarne le conseguenze, Alice deve sapere quanto si può fidare di quella cosa chiamata *chiave di Bob* che si ritrova nel suo insieme di chiavi. Per esempio la fiducia sarebbe al massimo valore nel caso in cui Bob avesse personalmente passato ad Alice

[1] uomo nel mezzo

un floppy contenente la sua chiave. È questo approccio di gestione delle chiavi decentralizzato e controllato dagli utenti che rende *PGP* essenzialmente diverso dagli schemi PKI centralizzati.
In ogni modo, in alcune occasioni le chiavi si ottengono con una richiesta ad un server fidato. Dopo la standardizzazione di X.509, *PGP* supporta anche questo tipo di certificati, oltre al meccanismo tradizionale dell'insieme di chiavi; tutte le versioni correnti di PGP supportano X.509.

Appendice A

Phil Zimmermann[1]

Philip R. Zimmermann è il creatore di *Pretty Good Privacy*, un pacchetto software crittografico. Originamente ideato come uno strumento per i diritti civili, *PGP* è stato pubblicato su Internet nel 1991. Ciò rese Zimmermann oggetto di un'indagine criminale durata tre anni perché il governo sostenne che le restrizioni all'esportazione statunitense di software crittografico erano state violate quando *PGP* si diffuse nel mondo. Nonostante l'insufficienza di fondi, la mancanza di personale pagato, l'assenza di un'azienda che lo supportò, e la persecuzione del governo, *PGP* divenne il software di codificazione per la posta elettronica più usato al mondo. Dopo che il governo chiuse l'indagine all'inizio del 1996, Zimmermann fondò la *PGP Inc.*. Questa compagnia fu acquistata dalla *Network Associates Inc.* (*NAI*) nel dicembre 1997, all'interno della quale egli ricoprì la posizione di *Membro anziano*. Nell'agosto 2002, *PGP* fu venduto dalla *NAI* alla nuova compagnia *PGP Corporation*, nella quale Zimmermann divenne *Consulente* e *Consigliere Speciale*. Attualmente Zimmermann offre consulenza per numerose compagnie ed organizzazioni legate all'industria riguardo problemi di crittografia ed è anche membro della *Scuola di Legge Stanford* per il *Center for Internet and Society*.
Prima di fondare la *PGP Inc.*, Zimmermann era un ingegnere sviluppatore di software con oltre 20 anni di esperienza, specializzato nella crittografia e nella sicurezza dei dati, nella comunicazione dei dati, e in sistemi integrati a tempo reale. Il suo interesse nell'aspetto politico della crittografia nacque

[1]tratto da `http://www.philzimmermann.com/it/index.html`

dalla precedente esperienza in temi della politica militare.
Egli ha ricevuto numerosi premi umanitari e tecnici per il suo lavoro pionieristico nel campo della crittografia. Nel 2001, Zimmermann fu introdotto nella *CRN Industry Hall of Fame*. Nel 2000 *InfoWorld* lo nominò tra i *Top 10 Innovators* nell'E-business. Nel 1999 egli ricevette il *Louis Brandeis Award da Privacy International*, nel 1998 un premio a vita da *Secure Computing Magazine*, e nel 1996 il riconoscimento *Norbert Wiener da Computer Professionals for Social Responsibility* per aver promosso l'uso responsabile della tecnologia. Nel 1995, egli ha anche ricevuto il *Chrysler Award* per *Innovazione nel Design*, il premio 1995 *Pioneer da Electronic Frontier Foundation*, il premio 1996 *PC Week IT Excellence*, il premio 1996 *Network Computing Well-Connected Award* per **Best Security Product**.
PGP fu selezionato da *Information Week* come uno dei prodotti *Top 10* più importanti del 1994. Newsweek ha anche nominato Zimmermann come uno dei *Net 50*, vale a dire le 50 persone più influenti in Internet nel 1995.
Oltre ai premi ottenuti con le versioni *PGP* sviluppate prima che Zimmermann iniziò a lavorare per la compagnia, le versioni successive del *PGP*, raffinate dal gruppo d'ingegneri della compagnia, continuano ogni anno ad essere riconosciute con molti altri premi dall'industria.
Zimmermann ottenne la sua laurea in scienza del computer nell'Università Florida Atlantic, nel 1978. Egli è un membro dell'*International Association of Cryptologic Research*, dell'*Association for Computing Machinery* e della *League for Programming Freedom*. Egli è il **Presidente dell'OpenPGP Alliance**, e fa parte del *Consiglio d'Amministrazione della Computer Professionals for Social Responsibility*, ed è membro del *Comitato Consultivo d'Anonymizer.com*, dell'*Hush Communications*, della *Veridis* e della *Qualys*.

Bibliografia

[1] rfc2240.txt - Request for Comments: 2440 – *OpenPGP Message Format*

[2] `http://openpgp.org/`
Sito curato dalla *OpenPGP Alliance*

[3] *Manuale GNU sulla privacy*, di Mike Ashley, tradotto da Lorenzo Cappelletti

[4] `http://www.enricozimuel.net/`
Sito ricco di algoritmi e documentazione, raccolta curata da Enrico Zimuel

[5] *Reti di calcolatori*, di Andrew S. Tanenbaum