Tecniche di Progettazione Automatica **Anno Acc. 1997-98**

PROF. ALFREDO DE SANTIS

PRETTY GOOD PRIVACY

a cura di Giovanna Abagnale, Massimo Dragonetti, Palmira Spiniello *20 Luglio 1998*

Indice

1 Introduzione

PGP (Pretty Good Privacy) è un pacchetto *freeware*, cioè disponibile a chiunque lo voglia utilizzare, realizzato da Philip Zimmermann tra il 1990 e il 1991 negli Stati Uniti, che fornisce un servizio di confidenzialità ed autenticità utilizzato per applicazioni di posta elettronica e memorizzazioni di file. L'idea di Zimmermann nel creare il PGP si basa essenzialmente su:

- selezionare i migliori algoritmi crittografici disponibili utilizzandoli come blocchi;
- integrare questi algoritmi in un'applicazione general purpose, che sia indipendente dal sistema operativo e dal processore e che si basi su un insieme ristretto di comandi facili da usare;
- fornire liberamente, via Internet o altre reti, il package con relativa documentazione e codice sorgente;

Dalla sua nascita il PGP ha subito notevoli trasformazioni ed ora è ampiamente utilizzato. Le ragioni di questo sviluppo sono:

1. È scritto in C, quindi portabile e versatile.
2. È disponibile su varie piattaforme, quali Dos/Windows, Unix, OS/2, Mac, ecc.
3. Si basa su algoritmi che sono considerati estremamente sicuri. In particolar modo il package utilizza **RSA** per le firme digitali e per la chiave di sessione, **IDEA** per cifrare i messaggi, **MD5** per calcolare l'hash del messaggio che si vuol firmare, un algoritmo di compressione **ZIP** per comprimere i messaggi che devono essere inviati.
4. Ha un ampio raggio di applicazione, da gruppi che vogliono selezionare e rafforzare uno schema standard per crittografare documenti, al singolo individuo che vuole comunicare in maniera sicura con altri individui su Internet o altre reti.
5. Non è stato progettato, nè manipolato, dal governo americano e, quindi, si è sicuri che non vi siano trapdoor al suo interno.

Dato il notevole uso, Zimmermann decise di stipulare un accordo con la società Viacript per fornire una versione commerciale del PGP a basso costo e totalmente compatibile. D'altra parte alcuni utenti hanno esitato ad utilizzare questo software poichè

1. Il PGP è un prodotto crittografico creato negli Stati Uniti, pertanto è vincolato dalle leggi di esportazione di materiale crittografico in vigore in tale paese. Tuttavia, le versioni americane più recenti sono state esportate legalmente fuori dagli USA.
2. La versione freeware del PGP è stata sviluppata e diffusa senza preoccuparsi di ottenere la licenza d'uso di RSA, che è necessaria negli Stati Uniti, ma non all'estero.

Inizialmente descriviamo le operazioni del PGP, esaminiamo come creare e memorizzare le chiavi, di seguito consideriamo la gestione della chiave pubblica ed, infine, esaminiamo i formati delle strutture dati della chiave del PGP. Partiamo da un approccio generale ed elaboriamo lo schema gradualmente fino ad una descrizione dettagliata di tutti i parametri del PGP.

2 Modalità del PGP

Il PGP realizza 5 servizi: autenticitá, confidenzialitá, compressione, compatibilitá con e-mail e segmentazione.

Funzione	Algoritmi usati	Descrizione sintetica
Cifratura del messaggio	IDEA, RSA	Un messaggio è cifrato con IDEA usando una session key one time generata dal mittente. La session key è cifrata usando RSA con la chiave pubblica del destinatario.
Firma digitale	RSA, MD5	Viene creato un codice hash di un messaggio con MD5, che viene cifrato usando RSA con la chiave privata del mittente.
Compressione	ZIP	Un messaggio è compresso per la memorizzazione e la trasmissione usando ZIP.
Compatibilità e-mail	Conversione Radix-64	Un messaggio cifrato è convertito in una stringa ASCII usando la conversione Radix-64 per fornire trasparenza alle applicazioni e-mail.
Segmentazione		PGP esegue la segmentazione e il riassemblaggio qualora la dimensione del messaggio sia proibitiva.

Tabella 1: Servizi del PGP

2.1 Autenticità

Supponiamo che A voglia inviare a B un messaggio firmato. La figura 1 mostra questo servizio.

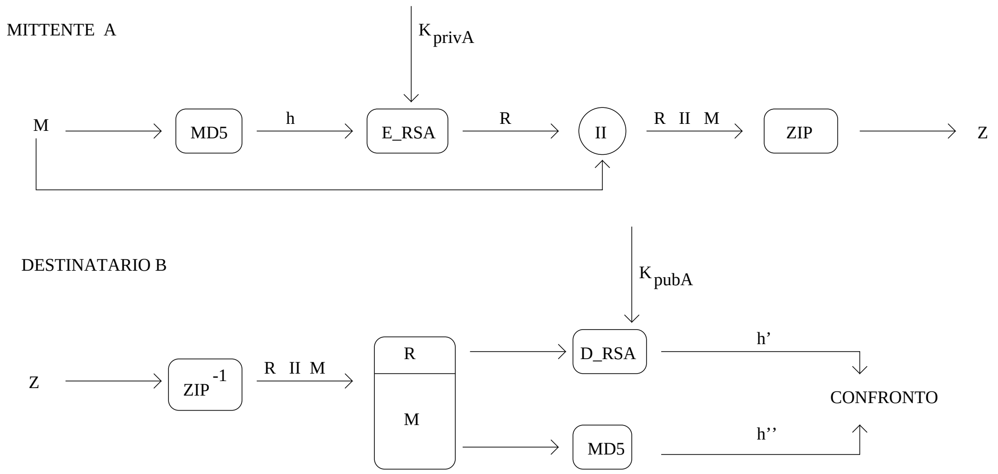

Figura 1: Autenticità

1. A genera un messaggio M.
2. Si applica a M l'algoritmo MD5 per generare un codice hash di 128 bit, ottenendo $h = MD5(M)$.
3. Il codice hash h è fornito in input a RSA usando la chiave privata di A (K_{privA}) il risultato R concatenato con M viene zippato e inviato a B. L'operazione di concatenazione è denotata con II.
4. B decomprime il messaggio Z, applica l'algoritmo di decifratura RSA a R usando come chiave la chiave pubblica di A (K_{pubA}) ed ottiene cosí il codice hash h' di M.
5. B applica ad M l'algoritmo MD5 per generare un nuovo codice hash h'' del messaggio e lo confronta con h'. Se coincidono ($h' = h''$), allora si è certi che il messaggio è autentico.

La forza di RSA garantisce che solo il possessore della chiave privata possa generare la firma, mentre la potenza di MD5 garantisce che nessun altro possa generare un nuovo messaggio M' il cui codice hash coincida con quello di M e, quindi, non si è in grado di generare la firma del messaggio originario. Sebbene le firme siano legate al messaggio o al file che firmano, esse possono anche essere *separate* dal messaggio stesso. La necessità di queste firme *separate* nasce quando l'utente desidera mantenere un insieme delle sole firme di tutti i messaggi, spediti o ricevuti, per verificare la presenza di virus o nel caso in cui un messaggio debba essere firmato da più persone.

2.2 Confidenzialità

Un altro servizio base fornito dal PGP è la confidenzialità , che consiste nel cifrare i messaggi sia per trasmetterli sia per memorizzarli localmente come file. In entrambi i casi viene utilizzato l'algoritmo IDEA, che fa uso di una chiave di 128 bit. In particolare, si adopera la modalità cipher feedback (CFB) a 64 bit con un vettore nullo di inizializzazione. Si è preferito l'IDEA al DES in quanto:

- risulta più resistente agli attacchi di forza bruta, data la lunghezza della chiave di 128 bit, invece dei 56 bit del DES;
- risulta più veloce poichè il DES è stato progettato per essere realizzato in hardware, per cui le sue implementazioni software sono dei compromessi;
- il DES viola le leggi sull'esportazione di materiale crittografico degli USA.

Nonostante il Triplo DES, denotato con TDES, sia più resistente del DES, data la chiave di lunghezza di 112 bit, si è preferito ancora l'IDEA in quanto si riscontrano gli altri due inconvenienti del DES sopra descritti.
Nel PGP ciascuna chiave convenzionale è utilizzata solamente una volta, cioè per ogni messaggio viene generata una nuova chiave, la session key, come un numero random di 128 bit. poichè è one time, la session key è limitata al messaggio e trasmessa con esso ed, inoltre, è codificata con la chiave pubblica del destinatario per renderla più sicura. La figura 2 mostra tale procedimento.

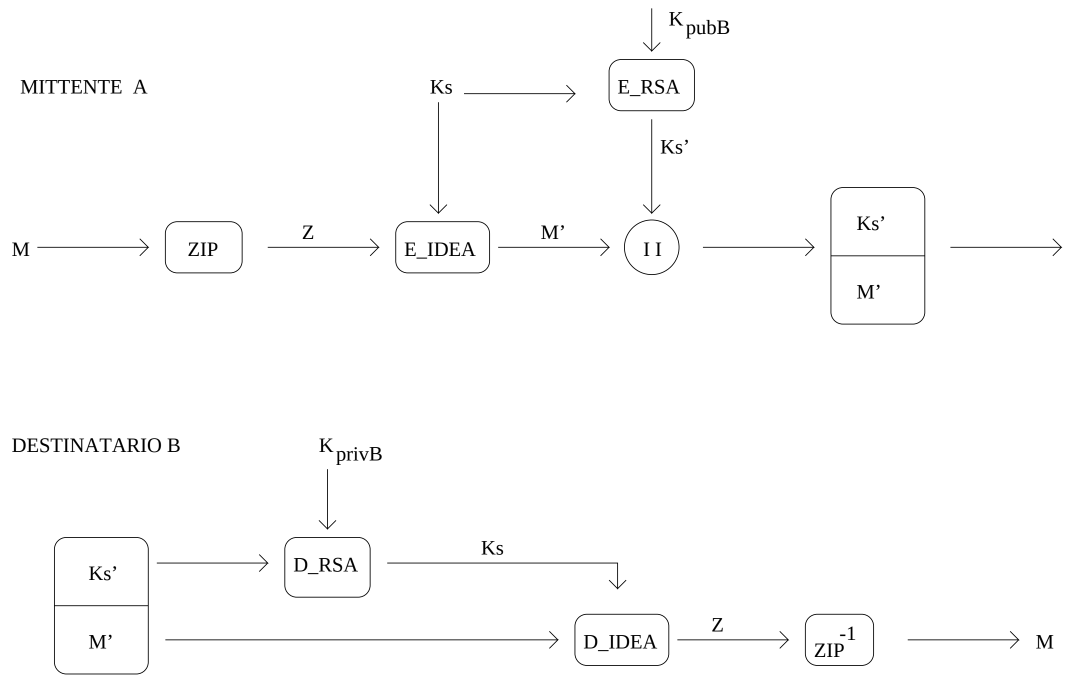

Figura 2: Confidenzialità

1. A genera un messaggio M ed un numero random di 128 bit, che rappresenta la session key, K_s, specifica per questo messaggio.

2. Dopo che M è stato compresso, viene cifrato usando IDEA con la session key come chiave. Il risultato è M'.

3. La session key è cifrata mediante RSA usando la chiave pubblica di B.

4. M' è concatenato con la cifratura della session key ed il risultato è inviato a B.

5. B usa RSA con la sua chiave privata per decodificare e ottenere K_s.

6. B decodifica M' tramite IDEA con K_s. Il risultato viene decompresso ottenendo M.

Si utilizza la combinazione IDEA/RSA, e non uno solo dei due, in quanto l'IDEA riduce i tempi di codifica (risulta essere più veloce di RSA), mentre RSA risolve il problema della distribuzione della session key, poichè solo il destinatario è in grado di recuperarla. Ogni messaggio è un evento indipendente, one time, con chiave propria. Poichè tale chiave è trasmessa tramite e-mail, non è possibile uno scambio diretto tra il mittente e il destinatario e, di conseguenza, non si è sicuri che abbiano la stessa session key. Nel PGP ciascuna chiave è utilizzata per cifrare solo una piccola parte del testo in chiaro e non esiste alcuna relazione fra le chiavi. In questo caso l'uso di una chiave convenzionale one time potenzia ciò che è già un approccio forte di crittografia convenzionale. Poichè RSA è sicuro, lo è anche l'intero schema. Per poter scegliere il grado di sicurezza del sistema (quanto più grande è la chiave tanto più sicuro è lo schema), PGP fornisce la possibilità di scegliere il formato della chiave tra i seguenti:

- Basso livello (512 bit)

- Alto livello (768 bit)

- Grado Militare (1024 bit)

2.3 Confidenzialità e Autenticità

Entrambi i servizi possono essere combinati per lo stesso messaggio. Il mittente firma il messaggio con la propria chiave privata, cifra il messaggio con la session key e poi cifra la session key con la chiave pubblica del destinatario.

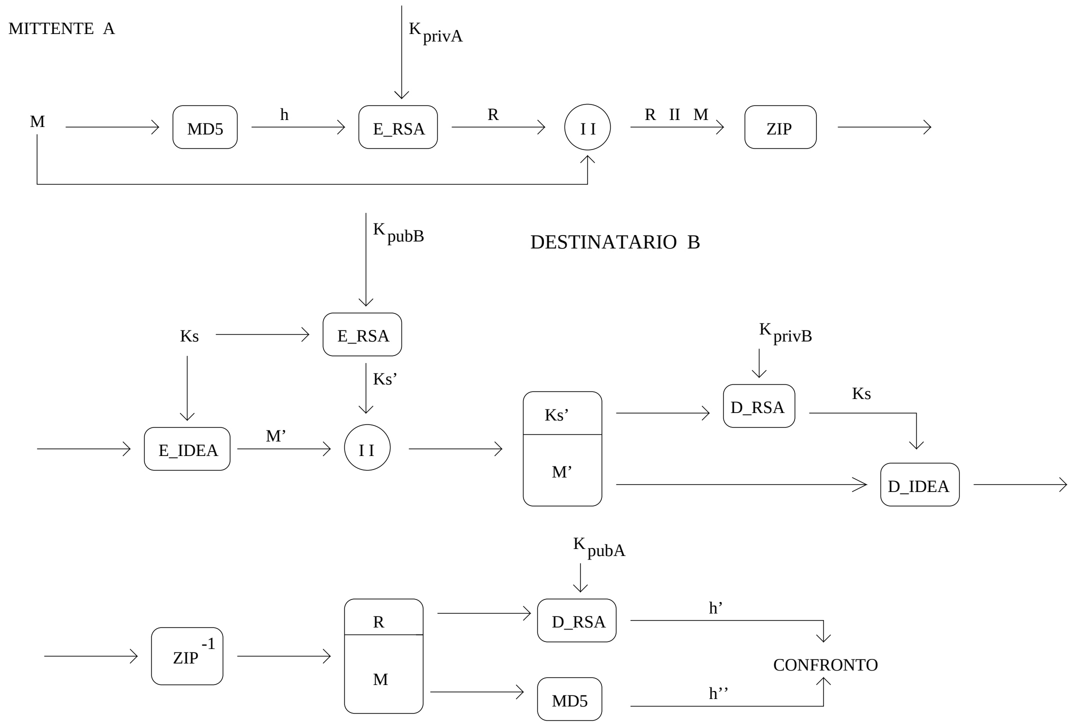

Figura 3: Autenticità e Confidenzialità

2.4 Compressione

Per default, PGP comprime il messaggio prima della cifratura e dopo la firma. In questo modo si risparmia spazio sia per la trasmissione e-mail sia per la memorizzazione. La firma è eseguita prima della compressione per due motivi:

1. Se si firmasse un messaggio compresso si dovrebbe memorizzare il messaggio originario con la sua versione compressa per verifiche successive della firma. Nel caso in cui venga memorizzato solo il messaggio originario si dovrebbe ricomprimere il messaggio ogni qualvolta venga richiesta la verifica della firma.

2. L'algoritmo di compressione del PGP è non deterministico, cioè il file compresso non è sempre lo stesso, ma è funzione del programma che si utilizza, mentre la decompressione fornisce sempre il messaggio originario. Firmando un messaggio compresso si vincolano tutte le implementazioni del PGP ad uno stesso algoritmo di compressione.

Si osserva, inoltre, che il messaggio viene cifrato dopo la compressione per rafforzare la sicurezza crittografica. Infatti, la crittoanalisi risulta essere più difficile poichè un messaggio compresso ha meno ridondanza del messaggio originario.

2.5 Compatibilità e-mail

Quando utilizziamo PGP, almeno una parte del blocco da trasmettere è cifrata. Se si utilizza solo il servizio di autenticità allora è cifrato il message digest con la chiave privata del mittente; se si utilizza il servizio di confidenzialità si cifra messaggio+eventuale firma con una chiave one time. Quindi, parte o l'intero blocco risultante è costituita da una stringa di ottetti (8 bit) arbitrari. Una volta cifrato e/o firmato il messaggio possiamo inviarlo tramite e-mail ad un altro utente. Con RSA e IDEA si ottiene una sequenza di bit che non sono tutti caratteri ASCII, mentre i programmi di e-mail usano file in formato ASCII, quindi si ha la necessità di convertire le stringhe di bit in caratteri ASCII. Questa conversione è detta **Radix-64**, mediante la quale la sequenza di bit è suddivisa in blocchi di tre ottetti (24 bit). Ogni gruppo di 6 bit del blocco di 24 bit è trasformato in un carattere, cioè in 8 bit, ottenendo una sequenza totale di 4 byte. La conversione Radix-64 è rappresentata da una tabella costituita da due colonne: la prima contiene il valore a 6 bit, la seconda il corrispondente carattere di conversione.

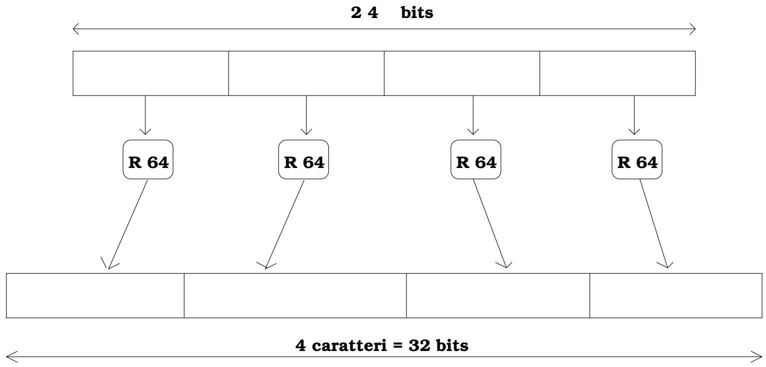

Figura 4: Codifica dei dati binari in formato RADIX-64

L'uso di Radix-64 espande un messaggio del 33%, senza causare alcun problema. Tale conversione è effettuata sull'output dell'algoritmo IDEA, che a sua volta prende in input un file compresso.
Supponiamo di aver compresso un file di lunghezza x della metá, per cui il file è 0.5 della lunghezza originaria. Con l'espansione è aumentato di 1/3, per cui il file è 1.33 di x.

$$1.33 \cdot 0.5 \cdot x = 0.665 \cdot x$$

Il file è diventato 0.665 della lunghezza originaria. Quindi la compressione compensa in maniera sufficiente l'espansione dovuta al Radix-64. Un aspetto rilevante di tale algoritmo è che converte la stringa input in formato Radix-64 anche se l'input è un testo ASCII. Quindi, se un messaggio è firmato ma non cifrato e la conversione è applicata all'intero blocco, l'output sarà incomprensibile all'osservatore casuale, il che fornisce un certo grado di confidenzialità .
Inoltre, PGP fornisce come opzione la possibilità di convertire in formato Radix-64 solo la parte della firma del messaggio in chiaro firmato. Ciò permette al destinatario di leggere un messaggio senza usare il PGP, il cui utilizzo è necessario per verificarne la firma.
I quattro servizi del PGP finora descritti sono correlati fra di loro.
In fase di trasmissione, se viene richiesta l'autenticità , la firma è generata usando un codice hash del messaggio in chiaro e testo in chiaro+firma viene compresso. Se viene richiesta la confidenzialità , il blocco ottenuto è cifrato con la chiave di cifratura convenzionale cifrata con RSA ed è convertito in formato Radix-64.

In fase di ricezione, il blocco ricevuto è prima convertito in formato binario, poi, se il messaggio è cifrato, il ricevente ricava la session key e decifra il messaggio; il blocco risultante è , dunque, decompresso. Se, invece, il messaggio è firmato, il ricevente calcola il codice hash trasmesso, ricava un proprio codice hash del messaggio e li confronta.

2.6 Segmentazione e riassemblaggio

Per essere trasmesso via e-mail, un messaggio non può superare una certa lunghezza (su Internet la lunghezza massima è di 50000 byte). Se si supera la lunghezza massima, si divide il messaggio in segmenti più piccoli che vengono inviati separatamente. Il PGP effettua questa segmentazione del messaggio automaticamente dopo tutti gli altri processi, compresa la conversione Radix-64. La session key e la firma appaiono una sola volta all'inizio del primo segmento. Alla fine della ricezione, PGP deve estrarre tutti gli header di e-mail e riassemblare l'intero blocco originario prima di eseguire i vari passi.

3 Chiavi crittografiche e key ring

PGP utilizza 4 tipi di chiavi di cifratura descritte nella tabella 2.

Chiave	**Algoritmo di cifratura**	**Uso**
Session key	IDEA	Viene utilizzata per cifrare messaggi per la trasmissione. Ciascuna session key viene utilizzata una sola volta ed è generata casualmente.
Chiave pubblica	RSA	Viene utilizzata per cifrare la session key per la trasmissione del messaggio. Sia il mittente che il destinatario devono mantenere una copia di ciascuna chiave pubblica.
Chiave privata	RSA	Viene utilizzata per cifrare il message digest per creare una firma digitale. Solo il mittente deve mantenere una copia della sua chiave privata.
Chiave basata sulla passphrase	IDEA	Viene utilizzata per cifrare e memorizzare le chiavi private del mittente.

Tabella 2: Chiavi di cifratura utilizzate dal PGP

Si osserva che:

- Si deve utilizzare un metodo per generare session key impredicibili.

- Un utente deve poter possedere più coppie di chiavi (pub/priv) in quanto potrebbe voler cambiare la sua coppia di chiavi di volta in volta o per interagire con diversi gruppi di utenti o semplicemente per aumentare la sicurezza limitando la quantità di materiale cifrato con una qualunque chiave. In questo modo non si ha una corrispondenza biunivoca tra gli utenti e le loro chiavi pubbliche, per cui bisogna utilizzare un metodo per individuare particolari chiavi.

- Ciascuna entità del PGP deve gestire un file delle proprie coppie di chiavi (pub/priv) e un file delle chiavi pubbliche degli utenti con cui desidera comunicare.

3.1 Generazione della session key

Ogni session key è associata ad un singolo messaggio ed è utilizzata solo per cifrare e decifrare quel messaggio tramite l'IDEA con una chiave di 128 bit. Questa chiave di 128 bit viene generata in msniera casuale usando IDEA stesso. Prima di descrivere come il PGP genera la session key e altri valori in maniera casuale forniamo la descrizione dell'ANSI X9.17, che è l'algoritmo base utilizzato dal PGP.

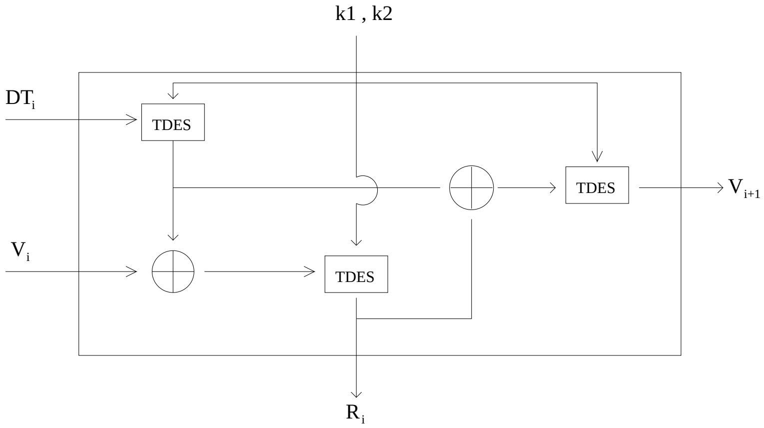

Figura 5: Generatore di numeri pseudocasuali ANSI X9.17

3.1.1 ANSI X9.17

L'ANSI X9.17 è un generatore di numeri pseudocasuali, il cui algoritmo, mostrato nella figura 5, utilizza il Triplo DES (indicato con TDES) per la cifratura. Tale algoritmo prende in input due numeri pseudocasuali, DT_i (una rappresentazione a 64 bit della data e dell'ora corrente, aggiornata ad ogni fase di generazione del numero) e V_i (un valore del seme a 64 bit, che è inizializzato ad un valore arbritario ed è aggiornato durante il processo di generazione), ed una coppia di chiavi (K1,K2), ognuna di 56 bit, che devono essere segrete e devono essere utilizzate solo per la generazione dei numeri pseudorandom. Si osserva che la coppia (K1,K2), fornita ai TDES utilizzati dall'ANSI X9.17, è sempre la stessa. L'output è costituito da un valore pseudocasuale (R_i) ed un valore seme (V_{i+1}), entrambi di 64 bit:

$$R_i = TDES_{K1,K2}(TDES_{K1,K2}(DT_i) \oplus V_i)$$

$$V_{i+1} = TDES_{K1,K2}(TDES_{K1,K2}(DT_i) \oplus R_i)$$

Diversi fattori contribuiscono alla forza crittografica di questo metodo. La tecnica utilizza una chiave a 112 bit e tre TDES per un totale di 9 cifrature DES. Lo schema è guidato da due input pseudocasuali, la data e l'ora, e da un seme prodotto dal generatore stesso. Se un nemico volesse alterare il risultato, dovrebbe modificare una grossa quantità di informazioni. Infatti anche se un R_i venisse alterato, dovrebbe essere impossibile dedurre V_{i+1} da R_i, in quanto viene utilizzata un'altra operazione di TDES per produrre V_{i+1}.

3.1.2 Generazione dei numeri casuali e pseudocasuali

PGP usa uno schema complesso e potente per generare numeri casuali e pseudocasuali per diversi scopi. PGP genera numeri casuali, considerando il tasto premuto e il tempo di battitura, e i numeri pseudocasuali, usando l'algoritmo specificato nell'ANSI X9.17.
I numeri casuali vengono usati per:

- generare le coppie di chiavi RSA;
- fornire il valore *seme* iniziale per il generatore dei numeri pseudocasuali;
- fornire un input aggiuntivo durante la generazione del numero pseudocasuale.

I numeri pseudocasuali vengono usati per:

- generare la session key;
- generare i vettori di inizializzazione per usare la session key nella modalità CFB.

Numeri casuali
PGP gestisce un buffer di 256 byte o dei bit casuali. Ogni volta si aspetta la digitazione di un tasto e registra, in un formato di 32 bit, l'istante in cui inizia ad aspettare. Quando riceve un input, registra l'istante in cui il tasto è stato premuto e il valore di 8 bit corrispondente al valore del tasto. L'informazione relativa al tempo e alla digitazione del tasto è usata per generare una chiave, che a turno è utilizzata per cifrare il valore corrente del buffer.

Numeri pseudocasuali
I numeri pseudocasuali utilizzano un valore seme di 24 ottetti e producono una session key di 16 ottetti, un vettore di inizializzazione di 8 ottetti ed un nuovo seme da utilizzare per la generazione del prossimo numero pseudocasuale. L'algoritmo è basato sull'ANSI X9.17, ma utilizza IDEA invece del TDES per la cifratura (come descritto nella sezione 2.2). L'algoritmo usa le seguenti strutture dati:

1. **Input**
   - *randseed.bin*: se questo file è vuoto viene riempito con 24 ottetti casuali.
   - *messaggio*: la session key e il vettore di inizializzazione IV, che dovranno essere usati per cifrare un messaggio, sono essi stessi funzione di quel messaggio, contribuendo ulteriomente alla casualità della chiave e di IV. Se un nemico conoscesse già il messaggio in chiaro, non ci sarebbe alcun bisogno apparente di catturare la session key one time.

2. **Output**

   - K (24 ottetti): i primi 16 ottetti, $K[0..15]$, contengono una session key, gli ultimi 8 ottetti, $K[16..23]$, contengono un IV.
   - *randseed.bin*: in questo file è memorizzato un nuovo valore seme.

3. **Strutture dati interne**

   - *dtbuf* (8 ottetti): i primi 4 ottetti, $dtbuf[0..3]$, sono inizializzati con il valore della data e ora corrente. Questo buffer è equivalente alla variabile DT_i nell' ANSI X9.17.
   - *rkey* (16 ottetti): indica la chiave di cifratura IDEA utilizzata in tutte le fasi dell'algoritmo.
   - *rseed* (8 ottetti): è equivalente alla variabile V_i dell'ANSI X9.17.
   - *rbuf* (8 ottetti): è un numero pseudocasuale generato dall'algoritmo. Questo buffer è equivalente alla variabile R_i dell'ANSI X9.17.
   - *K'* (24 ottetti): è il buffer temporaneo per il nuovo valore del randseed.bin.

L'algoritmo consiste di 9 passi. Il primo e l'ultimo passo sono di offuscamento, per ridurre il rischio che un nemico riesca a catturare il valore di un file randseed.bin. I rimanenti passi sono essenzialmente equivalenti alle iterazioni dell'ANSI X9.17.
Nell'algoritmo seguente indichiamo con E l'algorimo di cifratura IDEA in modalità ECB.

ALGORITMO

1. **Prelavaggio del seme precedente**

   a. Copiare il file randseed.bin in $K[0..23]$;

   b. Prendere il valore hash MD5 del messaggio (questo è già stato generato se il messaggio è stato firmato, altrimenti vengono usati i primi 4 ottetti di K);

   c. Usare il risultato come una chiave, usare un IV nullo e cifrare K nella modalità CFB;

   d. Memorizzare il risultato in K.

2. **Assegnazione del seme iniziale**

   a. Assegnare a $dtbuf[0..3]$ l'ora locale a 32 bit e a $dtbuf[4..7]$ tutti 0, copiare in rkey $K[0..15]$ e in rseed $K[16..23]$;

   b. Cifrare il dtbuf a 64 bit usando rkey a 128 bit nella modalità ECB e memorizzare il risultato in dtbuf.

3. **Preparazione per la generazione di ottetti casuali**
   $rcount \leftarrow 0$, $k \leftarrow 23$. Il ciclo costituito dai passi successivi da 4 a 7 verrà eseguito 24 volte ($k = 23, .., 0$), una per ogni ottetto casuale prodotto e memorizzato in K. La variabile rcount rappresenta il numero di ottetti casuali non usati in rbuf e conterà da 8 a 0, per tre volte, per generare i 24 ottetti.

4. **Verifica dei byte disponibili**
   Se $rcount = 0$ andare al passo 5, altrimenti al passo 7. I passi 5 e 6 eseguono un'istanza dell'ANSI X9.17 per generare nuovi 8 ottetti casuali.

5. **Generazione dei nuovi ottetti casuali**

   a. $rseed \leftarrow rseed \oplus dtbuf$.

   b. $rbuf \leftarrow E_{rkey}(rseed)$ nella modalità ECB.

6. **Generazione del seme successivo**

   a. $rseed = rbuf \approx dtbuf$.

   b. $rseed = E_{rkey}(rseed)$ nella modalità ECB.

   c. $rcount \leftarrow 8$.

7. **Trasferimento di un byte alla volta da rbuf a K**

   a. $rcount \leftarrow rcount - 1$.

   b. Generare un byte b casuale e $K[k] \leftarrow rbuf[rcount] \oplus b$.

8. **Controllo**

   Se $k = 0$ andare al passo 9, altrimenti $k \leftarrow k - 1$ e andare al passo 4.

9. **Postlavaggio del seme e risultato**

   a. Generare più di 24 byte usando i passi dell'algoritmo da 4 a 7 senza effettuare lo xor di un byte casuale nel passo 7 e memorizzare il risultato nel buffer K'.

   b. Cifrare K' con la chiave $K[0..15]$ e IV con $K[16..23]$ nella modalità CFB, memorizzare il risultato in randseed.bin.

   c. Restituire K.

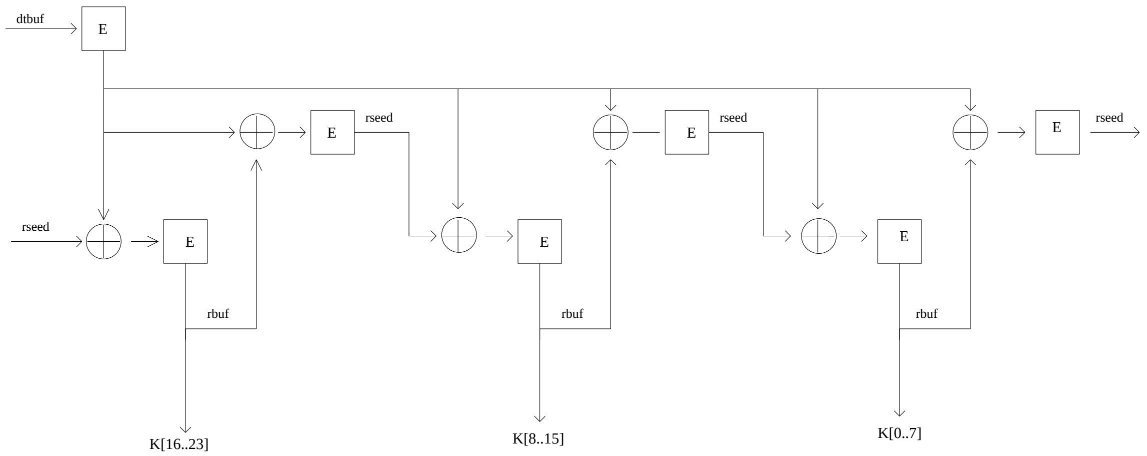

Figura 6: Generazione della session Key e di IV

Non dovrebbe essere possibile determinare la session key dai nuovi 24 ottetti generati nel passo 9. Comunque per essere sicuri che il file randseed.bin non fornisca alcuna informazione sulla session key più recente, i nuovi 24 ottetti sono cifrati ed il risultato è memorizzato come nuovo seme. Questo algoritmo dovrebbe fornire numeri pseudocasuali crittograficamente forti. Un'analisi preliminare di tale algoritmo indica che non esistono dipendenze interne fra i bit di una singola session key e che le session key successive sono indipendenti.

3.2 Identificativi di chiave

Precedentemente abbiamo detto che un messaggio cifrato viene concatenato ad una session key cifrata, utilizzata per cifrare. La session key stessa è cifrata con la chiave pubblica del destinatario, pertanto solo questi sarà in grado di ricavare la session key e, quindi, il messaggio. Ogni singolo utente possiede più coppie di chiavi pubblica/privata, quindi nasce il problema di come il destinatario possa individuare quale delle sue chiavi pubbliche sia stata usata per cifrare la session key. Per risolvere questo problema, il PGP assegna un identificativo (keyID) a ciascuna chiave pubblica che é, con altissima probabilitá, unico rispetto ad un identificativo dell'utente (UserID).
Data la chiave pubblica K_{pubA}, il keyID è $K_{pubA} \bmod 2^{64}$, cioè è ottenuto considerando i 64 bit meno significativi della chiave pubblica. La lunghezza è tale che la probabilità di avere due keyID uguali è molto bassa.
Ricordando che chi invia il messaggio può utilizzare una delle possibili chiavi private per cifrare il message digest, chi riceve il messaggio deve conoscere quale chiave pubblica deve usare, per cui è necessario un keyID anche per la firma digitale. Di conseguenza, la firma digitale di un messaggio include anche il keyID di 64 bit della chiave pubblica richiesta. Il destinatario, ricevuto il messaggio, verifica che il keyID sia relativo ad una chiave pubblica a lui nota rispetto a quel mittente, poi procede alla verifica della firma.

3.3 Formato del messaggio (trasmesso)

A questo punto è possibile descrivere dettagliatamente quelle che sono le componenti di un messaggio trasmesso:

1. la componente **messaggio**, che include i dati da memorizzare o trasmettere, il nome del file e il timestamp, che specifica il tempo di creazione;

2. la componente **firma** (opzionale), che include i seguenti componenti:

   - Il *Timestamp*, cioè l'ora in cui la firma è stata effettuata.
   - Il *Message digest*, di 128 bit, ottenuto con MD5. Il digest è calcolato concatenando il timestamp della firma con i dati della componente messaggio. La presenza del timestamp della firma in questa concatenazione garantisce la sicurezza rispetto ad attacchi di replay, mentre l'esclusione del nome del file e del timestamp della componente messaggio garantisce che le firme ricavate siano esattamente le stesse di quelle associate al messaggio. Le firme ricavate sono calcolate su un file separato che non ha nessun campo header della componente messaggio.
   - *I primi due ottetti del message digest* permettono al destinatario di determinare se è stata utilizzata la chiave RSA corretta per decifrare il message digest per l'autenticitá, confrontando questa copia dei primi due ottetti del testo in chiaro con quelli del digest decifrato. Questi ottetti vengono usati come sequenza di controllo di 16 bit per il messaggio.
   - Il *keyID della chiave pubblica del mittente* individua la chiave pubblica che dovrebbe essere usata per decifrare il message digest e quindi la chiave privata che è stata usata per cifrare il message digest.

3. la componente **session key** (opzionale) include la session key cifrata e l'identificativo della chiave pubblica del destinatario utilizzata dal mittente per cifrare la session key.

Osserviamo che le componenti messaggio e firma possono essere compressi.

3.4 Key ring

I keyID devono essere memorizzati ed organizzati in maniera sistematica per un uso reale ed efficiente da parte di tutti gli utenti. Il PGP considera due strutture dati per ogni utente:il private key ring e il public key ring.

3.4.1 Private key ring

Il *private key ring* è utilizzato per memorizzare le coppie di chiavi (pub/priv) posseduta da un utente U.
La chiave privata non viene memorizzata nel key ring in chiaro, ma è cifrata con IDEA, per renderla più sicura possibile.

Gestione della chiave privata
Per cifrare la chiave privata, da memorizzare nel private key ring, il PGP procede nel modo seguente:

1. L'utente sceglie una **passphrase** da usare per cifrare le chiavi private. La passphrase è una frase abbastanza grande e facile da ricordare.

2. Quando il sistema genera una nuova coppia di chiavi, usando RSA, chiede all'utente di digitare la passphrase. Usando MD5, viene generato un codice hash di 128 bit dalla passphrase che viene, poi, scartata.

3. Il sistema cifra la chiave privata usando IDEA con il codice hash della passphrase come chiave. Il codice hash è poi scartato e la chiave privata cifrata è memorizzata nel private key ring.

Quando un utente accede a questo ring, per ricercare una chiave privata, deve fornire la passphrase. PGP ricercherà la chiave privata cifrata, genererà il codice hash della passphrase e decifrerà la chiave privata cifrata utilizzando l'IDEA con il codice hash. La sicurezza di questo schema dipende dalla sicurezza della password. Per evitare di annotarla, l'utente dovrebbe usare una passphrase che sia facile da ricordare ma non da indovinare.

Struttura del private key ring
La figura 7 mostra la struttura generale di un private key ring.
Tale ring può essere visto come una tabella, in cui ogni riga rappresenta una delle coppie di chiavi possedute da U. Ciascuna riga contiene:

- **Timestamp**: la data/ora in cui la coppia di chiavi è stata generata.

- **KeyID**: identificativo di chiave formato dai 64 bit meno significativi della chiave pubblica per questa entry.

- **Public key**: la chiave pubblica della coppia.

Private Key Ring

TIMESTAMP	KEY ID	PUBLIC KEY	PRIVATE KEY CIFRATA	USER ID
⋮	⋮	⋮	⋮	⋮
Ti	$K_{pub,i} \bmod 2^{64}$	$K_{pub,i}$	$E\_IDEA_{H(Pi)}(K_{priv,i})$	User i
⋮	⋮	⋮	⋮	⋮

Figura 7: Private key ring

- **Private key**: la cifratura della chiave privata relativa alla coppia di questa entry.
- **UserID**: generalmente è l'indirizzo e-mail dell'utente, ma può essere un nome diverso associato a ciascuna coppia. Si può riusare lo stesso userID più di una volta.

Il private key ring può essere indicizzato sia dall'userID che dal keyID. Questo file viene memorizzato solo sulla macchina dell'utente che crea e possiede le coppie di chiavi ed è accessibile solo a questi.

3.4.2 Public key ring

Il *public key ring* è utilizzato per memorizzare tutte le chiavi pubbliche degli altri utenti noti ad U.
La figura 8 mostra la struttura generale di un public key ring.

PUBLIC KEY RING

TIMESTAMP	KEY ID	PUBLIC KEY	OWNER TRUST	USER ID	KEY LEGITIMACY	FIRMA/E	TRUST DI FIRMA
⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮
Ti	$K_{pub,i} \bmod 2^{64}$	$K_{pub,i}$	TRUST_FLAGi	USER i	TRUST_FLAGi		
⋮	.	⋮	⋮	⋮	⋮	⋮	⋮

Figura 8: Public key ring

Ogni riga contiene:

- **Timestamp**: la data/ora in cui questa entry è stata generata.
- **KeyID**: i 64 bit meno significativi della chiave pubblica per questa entry.
- **Public key**: la chiave pubblica di questa entry.
- **UserID**: identifica il proprietario di questa chiave.

- **Owner trust**: indica di quanto ci si fida del proprietario della chiave.
- **Key legitimacy**: indica di quanto ci si fida della validità della chiave.
- **Signature**: indica la firma.
- **Signature trust**:indica di quanto ci si fida della firma.

Similmente al private Key ring, il public Key ring può essere inidicizzato sia da UserId che da KeyId.

3.4.3 Utilizzo dei key ring nella trasmissione e nella ricezione del messaggio

Per semplicità non consideriamo la compressione e la conversione Radix-64.
Supponiamo che A voglia inviare a B un messaggio che sia cifrato.
Per **trasmissioni** di questo tipo il PGP effettua i seguenti passi:

1. *Firma del messaggio*

   a. Ricerca la chiave privata di A dal private key ring, usando l'userID di A. Se A non fornisce il proprio userID, PGP prende la prima chiave privata del ring.

   b. Richiede all'utente la passphrase per ricavare la chiave privata.

   c. Genera la firma del messaggio.

2. *Cifratura del messaggio*

   a. Genera la session key e cifra il messaggio.

   b. Ricava la chiave pubblica di B dal public key ring usando l'userID di B come indice.

   c. Costruisce la session key del messaggio.

Per la **ricezione** il PGP effettua i seguenti passi:

1. *Decifratura del messaggio*

   a. Ricava la chiave privata di B dal private key ring, usando il campo keyID della session key come indice.

   b. Richiede all'utente la passphrase per ricavare la chiave privata.

   c. Ricava la session key e decifra il messaggio.

2. *Autenticità del messaggio*

   a. Recupera la chiave pubblica di A dal public key ring usando il campo keyID della firma come indice.

   b. Recupera il message digest trasmesso.

   c. Calcola il message digest per il messaggio ricevuto e lo confronta con il message digest trasmesso.

3.5 Gestione delle chiavi pubbliche

Un utente A deve costruire un public key ring contenente le chiavi pubbliche degli altri utenti per interagire con essi tramite PGP.
Supponiamo che A abbia ottenuto da una BBS una chiave pubblica di B, che però è stata alterata da C, e l'abbia memorizzata nel public key ring. A questo punto C può inviare messaggi ad A, contraffare la firma di B e leggere ogni messaggio cifrato da A e inviato a B; A accetterà il messaggio come se fosse stato inviato da B. Esistono diversi approcci per minimizzare il rischio che il public key ring di un mittente contenga chiavi pubbliche false. Supponiamo che A voglia ottenere una chiave pubblica sicura da B.

1. *Ottiene fisicamente la chiave da B.* B memorizza la sua chiave pubblica su un floppy disk e lo dà ad A. Poi, A scarica la chiave dal floppy disk al suo sistema. Questo è un sistema molto sicuro, ma con ovvie limitazioni pratiche.

2. *Verifica la chiave per telefono.* Se A è in grado di riconoscere B al telefono, A telefona a B e gli chiede di dettargli la chiave in formato Radix-64. Un'alternativa più pratica consiste nella trasmissione della chiave da parte di B via e-mail. A genera un message digest della chiave di 128 bit con MD5, lo rappresenta in formato Radix-64 (impronta digitale). A richiede telefonicamente a B di dettargli l'impronta digitale da lui calcolata con un'operazione uguale a quella effettuata da A. Se le due impronte coincidono, la chiave è verificata.

3. *Ottiene la chiave pubblica di B da un individuo D che risulta di fiducia per A e per B.* Per questo motivo D crea un certificato firmato, che include la chiave pubblica di B, il tempo di creazione della chiave ed un periodo di validità per la chiave. D genera un MD5 digest del certificato, lo cifra con la sua chiave privata e aggiunge la firma al certificato. Poichè solo D può aver creato la firma, nessun altro può creare una chiave pubblica falsa e pretendere che sia stata firmata da D. Il certificato firmato può essere spedito direttamente ad A da B o D oppure inviato ad una BBS.

4. *Ottiene la chiave pubblica di B da un'autorità di certificazione fidata.* In tal caso viene creato e firmato da questa un certificato di chiave pubblica. A, fornendo il suo userID, accede all'autorità e riceve un certificato firmato da questa.

3.6 Revoca delle chiavi pubbliche

Un utente può voler rimuovere la sua chiave pubblica sia perchè sospetta che sia stata scoperta sia, semplicemente, per evitare l'uso della stessa chiave per un periodo di tempo prolungato. Notiamo che un nemico può , in qualche modo, aver ottenuto una copia della chiave privata in chiaro di un utente A o avere ottenuto la chiave privata dal private key ring di A e la sua passphrase.
Per revocare una chiave pubblica, il proprietario deve emettere un certificato firmato di revoca della chiave. Questo certificato ha lo stesso formato di un normale certificato di firma, con in più l'informazione della revoca della chiave pubblica. Notiamo che deve essere utilizzata la corrispondente chiave privata per firmare un certificato che revoca una chiave pubblica. Il proprietario dovrebbe, poi, diffondere questo certificato il più velocemente e ampiamente possibile per permettere ai potenziali utenti, che desiderano comunicare con questi, di aggiornare i propri public key ring. Inoltre, un nemico, che ha scoperto la chiave privata può anche

emettere un tale certificato. Tuttavia questo impedirebbe tanto al nemico quanto al legittimo proprietario l'uso della chiave pubblica.

4 L'uso del trust (fiducia)

Sebbene il PGP non includa alcuna specifica per stabilire le autorità di certificazione o stabilire la fiducia, fornisce uno strumento conveniente di utilizzo del trust.
Ciascuna entry nel public key ring é, in realtá, un certificato della chiave pubblica. Per ciascuna entry si ha oltre ad un campo *key legitimacy* (legittimià della chiave) calcolato dal PGP, che rappresenta quanto ci si possa fidare della validità della chiave pubblica per questo utente (più alto è il grado di fiducia, più forte è il legame di questo userID rispetto a questa chiave), anche 0 o più firme, che sono state collezionate dal proprietario del key ring e che firmano questo certificato.
Ad ogni firma è associato un campo *signature trust* (fiducia della firma) che indica quanta fiducia l'utente del PGP ha del firmatario per certificare le chiavi pubbliche.
Il valore del campo key legitimacy è calcolato considerando l'insieme dei valori dei campi signature trust nella entry.
Infine, ciascuna entry definisce una chiave pubblica, associata ad un particolare proprietario, ed è incluso un campo *owner trust* (fiducia del proprietario) che indica di quanto si ha fiducia di questa chiave pubblica per firmare altri certificati di chiave pubblica. Questo grado di fiducia è assegnato dall'utente.
Dato il public key ring di un utente A, descriviamo come elaborare il trust.

1. Quando A inserisce una nuova chiave pubblica nel public key ring, PGP deve assegnare un valore al campo owner trust che è associato al proprietario della chiave pubblica. Se il proprietario è A, a tale campo è assegnato automaticamente un valore di *completamente fidato* (trust ultimate); diversamente PGP chiede ad A la sua stima di trust da assegnare al proprietario di questa chiave e A deve digitare il grado di trust desiderato. L'utente A può specificare che questo proprietario è non noto, non fidato, parzialmente fidato o completamente fidato.

2. Una volta inserita la nuova chiave pubblica, possono essere attaccate ad essa una o più firme. Ulteriori firme possono essere aggiunte ad istanti successivi. Quando viene inserita una firma nella entry, il PGP cerca nel public key ring se l'autore di questa firma è un proprietario delle chiavi pubbliche note. Se è cosí al campo signature trust è assegnato il valore del campo owner trust per questo proprietario, altrimenti è assegnato il valore di *utente non noto*.

3. Il valore del campo key legitimacy è calcolato in base ai valori dei campi signature trust presenti in questa entry. Se almeno una firma ha un valore di ultimate per tale campo, allora il valore di key legitimacy è posto a *completo*; altrimenti PGP calcola una somma pesata dei valori trust. Viene assegnato un peso di 1/x alle firme che sono sempre fidate, di 1/y a quelle firme che sono di solito fidate, dove x e y sono parametri forniti dall'utente. Quando il totale dei pesi è 1, il legame tra userID e chiave è considerato attendibile e il valore di key legitimacy è posto a completo. Nel caso in cui non vi siano trust ultimate sono necessarie almeno x firme che siano sempre fidate o y firme che siano di solito fidate o una loro combinazione.

Periodicamente, PGP processa il public key ring per ottenere consistenza. Per ogni campo owner trust, PGP scandisce il ring per tutte le firme autorizzate da quel proprietario e aggiorna il campo signature trust uguagliandolo al campo owner trust. Questo processo inizia con le chiavi per le quali c'è un trust ultimate, poi tutti i valori dei key legitimacy sono calcolati in funzione delle firme attaccate.

5 Formati del PGP

Il PGP definisce un numero di formati che sono utilizzati per creare le entry del key ring, certificati e messaggi. Questi formati, considerati come pacchetti, sono dei blocchi da cui vengono costruite le entry, i certificati e i messaggi. Tutti i pacchetti iniziano con gli stessi due campi. Il campo *cipher type* indica di quale pacchetto si tratta e la lunghezza del prossimo campo. Il campo *packet lenght* indica la lunghezza del pacchetto in ottetti. Molti pacchetti contengono anche un campo *version* che specifica il numero di versione del PGP. Riassumiamo brevemente il contenuto di ogni pacchetto.

- Il **public key encrypted packet** è il formato per la componente session key di un messaggio. Contiene il keyID della chiave pubblica del destinatario considerato e la session key cifrata con quella chiave pubblica. Viene specificato anche l'algoritmo di cifratura, che per default è RSA.

- Un **signature packet** è utilizzato per diversi scopi, come messaggi e certificati; contiene il timestamp, il periodo di validitá, l'algoritmo per creare un message digest, il campo *signature classification* che indica la natura di questa firma ed un campo che indica quali fra i campi descritti sopra sono coinvolti nel calcolo del valore hash.

- Un **conventional key encrypted data packet** contiene un pacchetto di dati letterali o di dati compressi cifrato con un algoritmo convenzionale.

- Un **compressed data packet** contiene una specifica dell'algoritmo di compressione utilizzato (per default Zip), e un literal data packet compresso.

- Un **literal data packet** contiene il nome del file, il timestamp e un campo mode, che indica se è un file di testo o un file binario.

- Un **secret key certificate** definisce il formato usato per una entry in un private key ring.

- Un **public key certificate** è un elemento base di una entry del public key ring. Questo certificato non indica il proprietario della chiave pubblica. In un public key ring, il public key certificate deve essere sempre seguito da un **userID packet**.

- Un **userID packet** segue una chiave pubblica su un public key ring e una chiave privata su un private key ring. Quando un public key certificate è firmato, la firma maschera sia la chiave pubblica che l'userID.

- Un **key ring trust packet** specifica un livello di fiducia. L'interpretazione del byte del trust flag dipende dal contesto.

6 I keyserver

I keyserver sono degli enormi keyring, a cui tutti possono aggiungere la propria chiave pubblica e da cui è possibile estrarre una chiave di cui si ha bisogno. In pratica sono dei database automatici di chiavi pubbliche del PGP, accessibili e utilizzabili da tutti.
L'utilità dei keyserver è di aiutare il trasferimento delle chiavi tra gli utenti del PGP. Poichè il keyserver è essenzialmente un elenco di chiavi pubbliche, non si può fare alcun affidamento sulla validità delle chiavi che vengono reperite tramite essi, tranne nei casi in cui si fa riferimento alle firme di fiducia.
Esistono diversi keyserver in tutto il mondo e si aggiornano automaticamente tra loro e pertanto è sufficiente utilizzarne uno qualunque. In genere sono gestiti dalle università .
La sincronizzazione dei keyserver avviene a catena ogni volta che uno di questi riceve una nuova chiave. Questa sincronizzazione avviene tramite e-mail, che è l'unico tramite di comunicazione per i keyserver.
In genere un database di questo genere contiene circa 25000 chiavi PGP attive. La caratteristica fondamentale di questi database è il poter essere contattati tramite e-mail.
Il supporto dell'interfaccia e-mail per i keyserver risulta essere sconveniente nel caso in cui si desideri aggiornare il proprio key ring in maniera completa. In questi casi potrebbe essere vantaggioso effettuare operazioni FTP per reperire interi file di chiavi pubbliche.
La seguente tabella descrive i comandi principali per interagire con il keyserver.

Comando	**Descrizione**
HELP	Fornisce la lista delle istruzioni
ADD	Aggiunge la propria chiave pubblica
INDEX	Lista tutte le chiavi contenute nel keyserver
GET	Fornisce le chiavi del keyserver
GET[id]	Fornisce la chiave dell'utente con identificativo id
MGET stringa	Fornisce tutte le chiavi il cui identificativo contiene "stringa"

Tabella 3: Comandi principali e-mail

7 Debolezze del PGP

Come ogni sistema di crittografia, anche il PGP ha i suoi punti deboli che gli utenti devono conoscere per evitare situazioni che possano violare la propria privacy.
Elenchiamo di seguito i possibili attacchi che possono essere effettuati al PGP.

1. Impossessarsi della passphrase lasciata incustodita da un utente, in maniera tale da leggere e/o firmare i messaggi. È importante che la passphrase non sia costituita da una sola parola, in quanto potrebbe essere facilmente individuata utilizzando un computer, ma sia una frase abbastanza lunga.

2. Recuperare i file in chiaro non completamente cancellati. Se dopo aver criptato un file lo cancelliamo, il sistema operativo effettua una cancellazione logica del file e non fisica, per cui un nemico potrebbe recuperarlo prima che venga riscritta quell'area di memoria. Per ovviare a questo attacco, il PGP sovrascrive sul testo in chiaro subito dopo averlo cifrato.

3. Utilizzare **virus o worm** per mettere in crisi il PGP. Si potrebbe introdurre nel sistema un virus che possa individuare la passphrase e/o la chiave segreta e/o il testo in chiaro e inserire le informazioni ricavate in un file o inviarle in rete. Il virus potrebbe anche modificare la funzionalità del PGP stesso.

4. Sfruttare la conoscenza del codice sorgente del PGP per creare delle imitazioni contenenti insidie utilizzate per i propi scopi. Per evitare questa situazione è sufficiente procurarsi il PGP da una persona fidata che ne garantisca la non manomissione.

5. Analizzare il traffico dei messaggi cifrati in un ambiente multiutente (Unix, Vax/Vms). Un nemico non è in grado di leggere il contenuto di un messaggio crittato, ma può dedurre informazioni relative al messaggio osservando da dove viene, dove sta andando, la sua dimensione, il giorno in cui è stato spedito, ecc. Il PGP non ha difese contro questo tipo di attacco. Una possibile soluzione potrebbe essere quella di considerare speciali protocolli di comunicazione, che consentano di ridurre l'esposizione dell'ambiente di comunicazione all'analisi del traffico.

6. Rilevare i segnali elettromagnetici provenienti dal computer che sta trasmettendo. Questo tipo di attacco, detto **tempest attack**, consiste nel rilevare i segnali elettromagnetici generati dalla pressione sui tasti della tastiera del computer che si utilizza per scrivere. Per non permettere questo rilevamento, bisogna isolare l'ambiente in cui avviene la trasmissione e in cui sono collocati i cavi della rete per evitare l'emissione dei segnali.

7. **Crittoanalisi**. Questo tipo di attacco necessita di grandi risorse per poter essere attuato, in quanto il PGP utilizza IDEA e RSA, due crittosistemi per i quali non sono noti attacchi in grado di romperli dal punto di vista computazionale e, pertanto, può essere effettuato solo da organizzazioni specializzate, quali agenzie governative. Se qualche agenzia di stato trovasse il modo di rompere tali algoritmi, non lo renderebbe noto, in modo da sfruttare a suo favore la situazione. Per ridurre parte dei rischi dovuti ad un attacco di questo tipo il PGP inizialmente effettua la compressione del testo in chiaro.

8 Versioni del PGP

La prima versione del PGP è stata realizzata da Zimmermann negli USA. Successivamente sono state sviluppate altre versioni. Le versioni del PGP, disponibili su varie piattaforme, sono tutte compatibili fra loro: un file codificato con PGP per Unix può essere decodificato con il PGP del Dos e viceversa. Si ha una distinzione di tipo legale e non di compatibilità tra le versioni USA e quelle internazionali (identificate da una i dopo il numero di versione).
Le prime versioni sono le 2, che sono state prodotte allo scopo di garantire in maniera sicura e gratuita a tutti la privacy. L'ultima versione 2 internazionale è PGP 2.6.3i, il cui codice sorgente è stato pubblicato dal norvegese Stale Schumacher, e modifica il codice sorgente del PGP 2.6.2i.

Diversamente da questa versione, la 2.6.3i non usa la libreria RSAREF (libreria software che implementa le routine RSA), è totalmente compatibile con tutte le versioni precedenti e si compila su qualsiasi piattaforma.
Le versioni 5 sono quelle più recenti e sono state prodotte dalla PGP inc. Esse si basano su una versione commerciale americana, PGP 5.5 business. Ad esse sono state rivolte molte critiche poichè vengono a mancare garanzie sulla privacy personale. Tali versioni implementano l'algoritmo di crittografia a chiave pubblica DSS/DH e adoperano RSA con varie restrizioni (le versioni 2 utilizzavano prettamente RSA). Dal punto di vista della sicurezza RSA eDSS/DH possono essere considerati equivalenti.

8.1 PGP 5.0i

Il PGP 5.0i è stata la prima versione del PGP ad essere utilizzata in maniera perfettamente legale al di fuori degli Stati Uniti. Il regolamento di esportazione è relativo solo al software in forma elettronica,per cui il codice sorgente del PGP 5.0i è stato compilato, riportato in un libro pubblicato da Warthman Associates (12 libri di oltre 6000 pagine!) ed esportato legalmente fuori dagli USA. Dall' analisi di questi libri è stato ricavato il sorgente e ciò ha permesso di integrare il PGP nel sistema operativo Windows95. Il rilascio del PGP 5.0i è avvenuto il 18 Maggio 1997 e da allora si sono avuti molti aggiornamenti. Esso è diventato il sistema di crittografia a chiave pubblica più usato.

Riferimenti

[1] W. Stalling, *Network and internetwork security*, IEEE Press, 1995.

[2] P. Zimmermann, *PGP User's Guide*, vol.I: Essential Topics.
http://www.pegasus.esprit.ec.org/people/arne/pgpdoc1.html

[3] P. Zimmermann, *PGP User's Guide*, vol.II: Special Topics.
http://www.pegasus.esprit.ec.org/people/arne/pgpdoc2.html

[4] *International PGP Home Page.*
http://www.ifi.nio.no/staalesc/PGP

[5] *Frequently Asked Questions About the PGP.*
http://www.ifi.nio.no/staalesc/PGP/FAQ.shtml

[6] *FAQ from alt.security.pgp* by Jeff Licquia.
http://www.prairienet.org/jalicqui/pgpfag.txt

[7] *The PGP Attack FAQ*, Part A, Specifiche X9.17.
http://www.stack.ure.tue.nl/galactus/remailers/attack-4.html