GNU Privacy Guard (GnuPG) Mini Howto (italiano)

Brenno J.S.A.A.F. de Winter (inglese) <brenno@dewinter.com>, Michael Fischer v. Mollard (tedesco) <fischer@math.uni-goettingen.de>, Arjen Baart (olandese) <arjen@andromeda.nl>, Cristian Rigamonti (italiano) <cri@linux.it>
Versione 0.1.4 12 maggio 2003

Questo documento spiega come usare GNU Privacy Guard (GnuPG), un sistema di crittografia Open Source e compatibile con OpenPGP. Per mantenere il programma totalmente libero, si è evitato l'uso di RSA e di altri algoritmi brevettati. Il documento originale è scritto in tedesco da Michael Fischer v. Mollard, questa traduzione italiana, a cura di Cristian Rigamonti, è basata sulla traduzione inglese del testo originale.

Contents

1 Concetti

1.1 Crittografia a chiave pubblica

I metodi classici di crittografia usano una sola chiave per cifrare: il mittente cifra il messaggio con una chiave e il destinatario decifra il messaggio con questa stessa chiave, che gli deve essere fornita dal mittente con modalità che impediscano ad altri di entrarne in possesso. Se qualcun altro possiede la chiave, questi metodi crittografici diventano inutili.

L'uso della cosiddetta crittografia a chiave pubblica può risolvere questo problema. La crittografia a chiave pubblica prevede due chiavi: una chiave pubblica, che può essere diffusa con ogni mezzo, e una chiave privata, che non va diffusa e deve essere mantenuta segreta dal proprietario. Se il sistema è implementato bene, dalla chiave pubblica non si può risalire a quella privata. In questo modo, il mittente cifrerà il messaggio con la chiave pubblica del destinatario e quest'ultimo lo decifrerà con la propria chiave privata.

È cruciale che la chiave privata resti segreta e non sia a disposizione di nessuno all'infuori del proprietario: NON PUÒ ESSERE SPEDITA ATTRAVERSO INTERNET! È anche altamente sconsigliato usare GnuPG via `telnet` (è consigliabile non usare mai telnet, visti i suoi alti rischi di sicurezza).

1.2 Firme digitali

Per provare che un certo messaggio è stato inviato proprio dalla persona che dichiara di averlo fatto, è stato inventato il concetto di firma digitale. Con questa firma si può verificare l'autenticità di un messaggio, riducendo i rischi di imbattersi in un "cavallo di Troia" (ad esempio un messaggio che afferma di essere una patch per un certo problema, ma in realtà contiene un virus, o causa danni ai dati del vostro computer). Si può quindi verificare che qualsiasi informazione o dato provenga dalla fonte legittimata ad emetterlo.

Una firma digitale si ottiene dalla combinazione tra la chiave privata e il testo. Usando la chiave pubblica del mittente si può verificare un messaggio e controllare non solo che sia stato inviato dalla persona corretta, ma anche che il contenuto non sia stato modificato durante il trasporto.

1.3 Rete di fiducia

Un punto debole degli algoritmi a chiave pubblica è la diffusione delle chiavi. Un utente potrebbe spacciarsi per qualcun'altro, facendo circolare una chiave pubblica con un falso user ID e intercettando tutti i messaggi che venissero cifrati per quell'user ID. L'impostore potrebbe, dopo aver intercettato il messaggio, cifrarlo con la vera chiave pubblica del destinatario originale e inviarglielo, senza che quest'ultimo si renda conto dell'intercettazione.

La soluzione adottata da PGP (e quindi da GnuPG) è la firma delle chiavi: una chiave pubblica può essere firmata da altre persone, certificando così che la chiave appartiene veramente alla persona che sostiene di esserne il proprietario. La decisione su quanta fiducia riporre nelle firme spetta all'utente di GnuPG: sostanzialmente la fiducia in una firma dipende dalla fiducia che si ha nella chiave della persona che ha apposto la firma. Per essere assolutamente sicuri che una chiave appartenga a una certa persona, è consigliabile fare un controllo della cosiddetta "impronta digitale" della chiave ("fingerprint") usando un canale di comunicazione sicuro.

1.4 Limiti alla sicurezza

Se si possiedono dati considerati confidenziali, occorre preoccuparsi non solo del tipo di algoritmo di cifratura da usare, ma anche della sicurezza generale del proprio sistema. Fondamentalmente PGP può essere considerato sicuro e al momento della scrittura di questo documento non sono noti esempi di attacchi riusciti a PGP, ma ciò non implica una sicurezza assoluta (ad esempio, se la NSA riuscisse a rompere PGP non lo direbbe facilmente, né lo farebbero altre persone che cercano di rompere i sistemi con scopi criminali). Comunque, anche se PGP fosse completamente inattaccabile, resterebbero altre minacce per la sicurezza. Nel febbraio 1999 fu scoperto un cavallo di Troia che cercava chiavi private PGP sull'hard disk e le spediva via FTP: se le passphrase fossero state scelte male, le chiavi private avrebbero potuto facilmente essere scoperte.

Un possibile attacco (anche se poco probabile) è costituito da un cavallo di Troia che ritrasmetta ciò che si scrive sulla tastiera. È anche tecnicamente possibile (ma molto difficile) copiare il contenuto dello schermo: in questo caso non occorrerebbe decifrare alcun messaggio cifrato. Per tutti questi rischi occorre mettere in opera un buon sistema di sicurezza.

Non si tratta di diffondere la paranoia tra la gente, quanto piuttosto la consapevolezza che ci sono molti aspetti da considerare per essere più sicuri: la cosa più importante è tenere in mente che la crittografia è solo uno dei passi verso la sicurezza, non una soluzione totale. I cavalli di Troia, come il virus Melissa del marzo 1999, hanno spesso colto molte società impreparate.

2 Installazione

2.1 Sorgenti di GnuPG

Il sito ufficiale da cui scaricare è la

homepage di GnuPG `http://www.gnupg.org/download.html` . A questo indirizzo si troveranno anche link a siti mirror.

A causa di restrizioni legali è vietato scaricare GnuPG da server che si trovano negli USA: gli USA pongono restrizioni all'esportazione di software crittografico. È il motivo per cui PGP è disponibile in una versione internazionale e una nazionale (per gli USA). Per scrivere la versione internazionale, il codice sorgente è stato esportato sotto forma di libro stampato ed è stato scansito in Europa (Oslo); maggiori informazioni in proposito possono essere trovate sulla *homepage di PGP International* http://www.pgpi.com . La versione internazionale di PGP può essere importata liberamente negli Stati Uniti, a patto che non venga ri-esportata.

Se si ha già una versione di GnuPG o PGP, è consigliabile verificare il file di firma (si veda la sezione 5 (Firmare e verificare le firme)).

2.2 Configurazione

È possibile procurarsi GnuPG sotto forma di pacchetto Debian, di pacchetto RPM (Redhat Package Manager) o di codice sorgente. GnuPG è incluso nelle ultime distribuzioni RedHat. Per controllare se lo si ha già sul proprio sistema si usi:

```
rpm -q gnupg
```

I pacchetti binari sono installati usando gli strumenti caratteristici delle varie piattaforme Linux. Per installare GnuPG su altre piattaforme occorre compilarlo da sé. Chiunque renda disponibile al pubblico un metodo di installazione per una nuova piattaforma farà cosa gradita.

Visto che lo sviluppo principale viene fatto su piattaforma Linux (x86), non dovrebbe essere difficile portare GnuPG su altri sistemi. La lista dei sistemi operativi che attualmente supportano GnuPG si può trovare sulla *homepage di GnuPG* . La procedura descritta di seguito è abbastanza indipendente dalla piattaforma e può essere usata per installare GnuPG partendo dal file tar dei sorgenti.

Si decomprima il file tar con i comandi:

```
tar xvzf gnupg-?.?.?.tar.gz
```

Dopodiché si entri nella directory che contiene il codice sorgente e si scriva:

```
./configure
```

Non dovrebbe succedere nulla di speciale. Con

```
./configure --help
```

è possibile vedere le impostazioni disponibili per la compilazione. Se ci sono problemi di internazionalizzazione, è possibile usare le librerie gettext incluse nel sorgente, usando l'opzione `-with-included-gettext`, o disabilitarle usando l'opzione `-disable-NLS`.

2.3 Compilazione

A questo punto, è possibile compilare il programma con il comando

```
make
```

Se dovessero verificarsi problemi, è consigliabile nell'ordine: cercare di risolvere il problema facendo uso della documentazione disponibile, controllare che il problema non sia dovuto a un bug (controllando il file BUGS su http://www.gnupg.org), chiedere a qualche conoscente e solo a questo punto chiedere sulla mailing list di GnuPG (si veda la sezione 7 (Fonti di informazioni)). Se il problema riguarda i percorsi sul sistema, occorre eseguire `make clean`, quindi eseguire di nuovo `configure` e riprovare a compilare. Se ancora non funziona, siete autorizzati a farvi prendere dal panico!

2.4 Installazione

Si esegua:

```
make install
```

per copiare il programma e le pagine di manuale nella directory di installazione, che, a meno di modifiche apportate al ./configure, sarà `usr/local/share/gnupg/`. Si può trovare questa directory anche nel file `options.skel`, che può essere modificato e copiato come `~/.gnupg/options`, in modo che le modifiche apportate siano usate come standard. Questa copia viene eseguita automaticamente durante la creazione di `~/.gnupg/`. Le opzioni possibili sono tutte ben documentate e non è necessario spiegarle qui nel dettaglio.

È possibile eseguire GnuPG suid root, in modo che il programma giri con tutti i diritti dell'utente root e non scriva dati in locazioni di memoria teoricamente accessibili da altri. Non si può giudicare a priori quanto questo sia rischioso, d'altra parte si tenga anche conto che eseguire il programma suid root rende più pericolosi gli effetti di un eventuale cavallo di Troia (un cavallo di Troia che gira come utente root può danneggiare un intero sistema). Se si sceglie, per questo motivo o per altri, di non eseguire GnuPG suid root, è possibile disabilitare il relativo messaggio di avvertimento impostando `no-secmem-warning` in `~/.gnupg/options`.

3 Uso delle chiavi

3.1 Creare una chiave

Con

```
gpg --gen-key
```

verrà creata una nuova coppia di chiavi (una pubblica e una privata). Il primo problema è quale algoritmo usare: maggiori informazioni sugli algoritmi si trovano nella *PGP DH vs. RSA FAQ* `http://www.scramdisk.clara.net/pgpfaqfs.html` o in 7.4 (Applied Cryptography). La scelta predefinita (e la più usata) è quella di usare l'algoritmo DSA/ElGamal, che non è brevettato.

Il secondo problema è la lunghezza della chiave. In questo caso la scelta dipende dalle preferenze dell'utente tra sicurezza e tempo di calcolo: se una chiave è lunga, diminuisce il rischio di decifrare abusivamente un messaggio intercettato, ma aumenta il tempo di calcolo per cifrare e decifrare. Se si è preoccupati per il tempo di calcolo, ma si ha intenzione di usare la chiave per un periodo di tempo abbastanza lungo, si tenga conto dell'aumento di prestazioni dei processori, che diventano sempre più veloci. La lunghezza minima per una chiave GnuPG è di 768 bit; la lunghezza standard per le chiavi DSA è di 1024 bit, anche se molti raccomandano di usare chiavi da 2048 bit (che al momento è anche la lunghezza massima supportata da GnuPG). Se la sicurezza è il fattore di gran lunga più importante, è consigliabile scegliere la massima lunghezza di chiave disponibile.

Successivamente, il sistema chiederà di inserire nomi, commenti e indirizzi e-mail, che verranno usati per la costruzione della chiave; questi potranno essere modificati in parte anche successivamente, si veda la sezione 3.5 (Amministrazione delle chiavi).

Infine, occorre scegliere una password (di solito si usa il termine "passphrase", frase segreta, visto che sono ammessi gli spazi), che andrà immessa ogni volta che si useranno funzionalità che richiedono la chiave privata. Una buona passphrase ha le seguenti caratteristiche:

- è lunga
- contiene caratteri speciali (non alfanumerici)

- è una "parola" speciale (non un nome)
- è molto difficile da indovinare (quindi niente nomi, date di nascita, numeri di telefono o di carta di credito, nomi di bambini, ecc.)

Si può migliorare la sicurezza anche usando Le MAiusCOLe aLTernATe allE mInusCOLe. Nello scegliere la passphrase, assicurarsi di NON DIMENTICARLA: se succedesse, non sarebbe più possibile usare la propria chiave privata, ad esempio per leggere i messaggi ricevuti. È cosa saggia anche generare un certificato che contiene queste informazioni (e conservarlo con cura in un luogo sicuro). Si veda la sezione 3.4 (Revocare una chiave).

Dopo aver immesso tutti i dati, il sistema inizerà a generare le chiavi. Questa operazione richiederà un po' di tempo, durante il quale il sistema deve raccogliere molti dati casuali. Si può aiutare a generare dati casuali ad esempio lavorando su un'altra finestra. Ogni chiave generata è diversa dalle altre: se si genera una chiave e dopo cinque minuti se ne genera un'altra fornendo gli stessi dati (nome, email, passphrase, ecc.), si otterrà una chiave diversa. Ecco perchè è importante non dimenticarsi la passphrase.

3.2 Esportare le chiavi

Il comando per esportare la chiave di un utente è:

```
gpg --export [UID]
```

Se non si indica un UID (User ID), verranno esportate tutte le chiavi presenti nel portachiavi. L'output predefinito è lo `stdout`, ma è possibile dirigere l'output su un file usando l'opzione `-o`. In molti casi è consigliabile usare l'opzione `-a` per scrivere la chiave in un file ASCII a 7 bit, invece che in un file binario.

Per allargare il proprio orizzonte e permettere ad altri di inviare messaggi in modo sicuro, occorre esportare la propria chiave pubblica e pubblicarla sulla propria home page, o tramite finger, oppure caricarla su un key server come http://www.pca.dfn.de/dfnpca/pgpkserv/, o ancora usando altri metodi.

3.3 Importare le chiavi

Quando si riceve la chiave pubblica di qualcuno, prima di usarla occorre importarla nel proprio portachiavi (si può farlo anche per più di una chiave alla volta). Il comando è il seguente:

```
gpg --import [nomefile]
```

se si omette il nome del file, i dati verranno letti dallo `stdin`.

3.4 Revocare una chiave

Ci sono vari motivi per voler revocare una chiave esistente, ad esempio se la chiave privata è stata smarrita o è caduta nelle mani sbagliate, oppure se non si vuole più usare lo stesso UID, oppure se la lunghezza della chiave non è più sufficiente, ecc. In tutti questi casi, il comando per revocare la chiave è:

```
gpg --gen-revoke
```

che genera un certificato di revoca. *Per poter fare questo occorre la chiave privata*, altrimenti chiunque potrebbe revocare la chiave. Questo ha uno svantaggio: se si dimentica la passphrase, la chiave diventa inutilizzabile e non è neanche possibile revocarla! Per evitare questo inconveniente, è buona norma creare un certificato di revoca non appena si genera la chiave; esso andrà conservato in un luogo sicuro (può essere salvato su disco, su carta, ecc.), badando che non cada nelle mani sbagliate, altrimenti qualcun altro potrebbe revocare la chiave rendendola inutilizzabile.

3.5 Amministrazione delle chiavi

Il sistema GnuPG comprende dei file che servono per immagazzinare tutte le informazioni che accompagnano le chiavi (tutte tranne i valori di fiducia nel proprietario della chiave: per maggiori informazioni in proposito si veda la sezione 3.6 (Firmare le chiavi)). Con

```
gpg --list-keys
```

verranno mostrate tutte le chiavi esistenti. Per vedere anche le firme, si usi:

```
gpg --list-sigs
```

(si veda la sezione 3.6 (Firmare le chiavi) per maggiori informazioni). Per vedere le impronte digitali si usi:

```
gpg --fingerprint
```

Vedere le impronte digitali ("Fingerprint") serve ad assicurarsi che la chiave appartenga davvero alla persona che sostiene di esserne il proprietario (ad esempio al telefono). L'output di questo comando è una breve lista di numeri.

Per vedere la lista delle chiavi private si usi:

```
gpg --list-secret-keys
```

Si noti che non ha alcuna utilità vedere firme o impronte digitali di chiavi private.

Per cancellare una chiave pubblica si usi:

```
gpg --delete-key UID
```

Per cancellare una chiave privata si usi:

```
gpg --delete-secret-key
```

C'è un altro comando importante per la gestione delle chiavi:

```
gpg --edit-key UID
```

Usando questo comando è possibile modificare (tra le altre cose) la data di scadenza di una chiave, aggiungere UID o firmare una chiave (ovviamente per questo è necessaria la propria passphrase). Dopo aver eseguito il comando –edit-key si otterrà un prompt interattivo da cui digitare i comandi successivi.

3.6 Firmare le chiavi

Come accennato nell'introduzione, il principale tallone d'Achille del sistema è l'autenticità delle chiavi pubbliche: se si usa una chiave pubblica contraffatta si può dire addio alla crittografia sicura. Per evitare questo rischio, c'è la possibilità di firmare le chiavi, ossia di porre la propria firma digitale sulla chiave, certificandone la validità. In pratica, la firma certifica che lo user ID menzionato nella chiave corrisponde alla persona che possiede la chiave. Una volta che si è certi di questo, si può usare la chiave in tutta sicurezza.

Per firmare una chiave, si usi il comando `gpg -edit-key UID` e successivamente il comando `sign`.

Bisogna firmare una chiave solo quando si è ASSOLUTAMENTE CERTI della sua autenticità!!! Questa condizione può verificarsi quando si è ricevuta la chiave direttamente da una persona (ad esempio durante un "key signing party", un raduno per la firma delle chiavi) o quando la si è ricevuta per altri mezzi e se ne è controllata l'autenticità col metodo dell'impronta digitale (ad esempio al telefono). Non si dovrebbe mai firmare una chiave a priori.

GnuPG calcola la validità delle chiavi basandosi sulle firme disponibili e sui valori di fiducia nel proprietario ("ownertrust"). La fiducia nel proprietario di una chiave rappresenta la fiducia che si ripone nella capacità del proprietario di firmare correttamente altre chiavi. I valori possibili sono:

- 1 = Indefinito
- 2 = Nessuna fiducia
- 3 = Fiducia marginale
- 4 = Fiducia completa

Così, se non si ha fiducia nel proprietario di una chiave, le eventuali firme apposte da quest'ultimo su un'altra chiave veranno ignorate durante il calcolo del valore di validità della chiave. Le informazioni sulla fiducia non sono immagazzinate negli stessi file che contengono le chiavi, ma in un file separato.

4 Cifrare e decifrare

Dopo aver installato e configurato il programma nel modo voluto, è possibile iniziare a cifrare e decifrare.

Si può usare il programma con più di una chiave privata, selezionando per ogni operazione quale deve essere quella attiva, con l'opzione `-u UID` o l'opzione `-local-user UID`. In questo modo la chiave privata predefinita sarà sostituita da quella indicata.

Per modificare invece il destinatario, si può usare l'opzione `-r` o l'opzione `-recipient`.

4.1 Cifrare

Il comando per cifrare è

```
gpg -e destinatario [dati]
```

o

```
gpg --encrypt destinatario [dati]
```

Per evitare il rischio che qualcun altro si spacci per noi, è consigliabile firmare qualsiasi cosa si voglia cifrare; si veda la sezione 5 (Firmare e verificare le firme).

4.2 Decifrare

Il comando per decifrare è:

```
gpg [-d] [dati]
```

o

```
gpg [--decrypt] [dati]
```

Anche in questo caso, l'output predefinito è lo `stdout`, ma è possibile redirigere l'output a un file con l'opzione `-o`.

5 Firmare e verificare le firme

Per firmare dati con la propria chiave, si usa il comando:

```
gpg -s (o --sign) [dati]
```

Durante questa operazione, i dati vengono anche compressi, quindi il risultato non sarà leggibile. Per avere un risultato leggibile si può usare:

```
gpg --clearsign [dati]
```

In questo modo si firmeranno i dati lasciandoli leggibili.

Con

```
gpg -b (o --detach-sign) [dati]
```

è possibile scrivere la firma in un file separato. Questa opzione è utile specialmente per firmare file binari (come ad esempio archivi). Anche l'opzione `-armor` può tornare utile.

Spesso si troveranno dati cifrati e firmati allo stesso tempo. Il comando completo è:

```
gpg [-u mittente] [-r destinatario] [--armor] --sign --encrypt [dati]
```

L'utilizzo delle opzioni `-u` (`-local-user`) e `-r` (`-recipient`) è stato descritto in precedenza.

Quando dei dati cifrati sono stati anche firmati, la firma viene verificata dopo aver decifrato i dati. È possibile verificare le firme con il comando

```
gpg [--verify] [dati]
```

Ovviamente per fare questo è necessario possedere la chiave pubblica del mittente.

6 Front-end

L'uso di GnuPG è molto facilitato grazie all'esistenza di una vasta scelta di programmi che usano o supportano la crittografia di GnuPG. Ci sono front-end grafici che consentono di gestire le chiavi con semplici clic del mouse, e molti programmi di posta elettronica permettono di cifrare e decifrare i messaggi in modo automatico. Una lista quasi completa di front-end è disponibile sulla pagina *GnuPG front-end* . Ne illustreremo alcuni in questo capitolo.

6.1 Interfacce grafiche

6.1.1 GPA

GPA , *GNU Privacy Assistant* è un'interfaccia utente grafica per GNU Privacy Guard (GnuPG), sviluppata nell'ambito del progetto GnuPG. Con GPA, è possibile vedere il proprio portachiavi, importare ed esportare chiavi, generarle, modificarne le caratteristiche e cifrare, firmare o decifrare documenti. Installare GPA è facile: occorre scaricare il pacchetto, decomprimerlo ed eseguire la solita sequenza:

```
./configure; make; make install.
```

Il programma si esegue digitando

```
gpa
```

6.1.2 Seahorse

Seahorse è un front-end GNOME per GnuPG. Può essere usato per cifrare, firmare e decifrare testi e altri file. Il testo può essere preso dagli appunti o scritto direttamente con il piccolo editor incluso. Seahorse ha anche un gestore di chiavi che può essere usato per modificare quasi tutte le caratteristiche delle chiavi presenti nel proprio portachiavi. È possibile installare Seahorse da un pacchetto Debian (gli RPM non sono ancora disponibili al momento) o dai sorgenti. L'installazione dai sorgenti avviene come per gli altri programmi: scaricare, decomprimere, configure e make install. L'installazione pone seahorse in `/usr/local` e aggiunge un elemento al menù "Applicazioni" di GNOME.

6.1.3 Geheimnis

Geheimnis è un front-end per GnuPG basato su KDE. Dovrebbe avere tutte le funzioni di GPA o Seahorse. (Nota dell'autore: purtroppo quando ho provato a compilare il programma, mi sono bloccato a causa di una libreria mancante: /usr/lib/libfam.la).

6.2 Programmi di posta elettronica

La maggior parte dei programmi di posta elettronica supportano GnuPG. Tra i tanti:

- Mozilla
- Pine
- Kmail
- Eudora
- Mutt
- exmh

Ce ne sono probabilmente molti di più: è difficile elencarli tutti.

Usare un programma di posta elettronica con il supporto GnuPG permette di decifrare i messaggi che sono stati cifrati con la propria chiave pubblica, firmare i propri messaggi in modo che il destinatario possa verificarne l'autenticità e cifrare la propria posta con la chiave pubblica dei destinatari.

6.2.1 Mozilla e Enigmail

Mozilla non ha in sé il supporto per GnuPG. Per usare GnuPG con Mozilla occorre installare un "plug-in", come *EnigMail* . Enigmail è un "plug-in" per Mozilla/Netscape Mail che permette agli utenti di accedere alle capacità di autenticazione e crittografia fornite da programmi popolari come GPG e PGP. Enigmail può cifrare/firmare la posta in uscita e decifrare/verificare quella in entrata. Può anche importare ed esportare chiavi.

Installare EnigMail su un sistema Linux RedHat è facile, visto che sono disponibili RPM per l'ultima versione di Mozilla. Ci sono due pacchetti da installare: `mozilla-enigmail-0.39-3.i386.rpm` e `mozilla-ipc-0.99-0_rh7x.rpm`. Dopo aver installato questi RPM e aver riavviato Mozilla (o Netscape 6.x) si è pronti per usare GnuPG nella propria posta. Si noti tuttavia che EnigMail cifra solo i testi dei messaggi, non gli allegati; occorrerà cifrare separatamente i file che si vuole allegare. Su altri sistemi probabilmente occorre installare EnigMail dai sorgenti.

6.2.2 Kmail

Kmail, il programma di posta elettronica standard per KDE, ha il supporto integrato per GnuPG e PGP. Per impostare il programma in modo da poter firmare e cifrare i messaggi, occorre inserire il proprio user ID di GnuPG nella sezione "Identità" della configurazione di Kmail. Quando si spedisce un messaggio, per firmarlo o cifrarlo occorre usare i pulsanti "Firma messaggio" o "Cifra messaggio" sulla barra degli strumenti.

7 Fonti di informazioni

7.1 GnuPG

- La *homepage di GnuPG* `http://www.gnupg.org`
- La mailing list di GnuPG, gli archivi e le istruzioni sono disponibili sul *sito di GnuPG* .
- Le informazioni incluse nel progetto GnuPG (aggiornate fino alla versione 0.9.2), anche se non ancora complete. Da non dimenticare anche:

  ```
  gpg --help
  ```

  che fornisce informazioni molto utili.

7.2 PGP

PGP è un programma di crittografia più vecchio ma ancora molto diffuso ed utilizzato. Molti documenti prodotti negli scorsi anni per PGP sono ancora utili, perchè molte informazioni contenute sono generali e valide anche per GnuPG. Si controllino gli indirizzi seguenti:

- La *Home page di PGP International* `http://www.pgpi.com`
- La *PGP DH vs. RSA FAQ* `http://www.hertreg.ac.uk/ss/pgpfaq.html` , con informazioni sulle differenze tra questi due algoritmi, che sono usati da GnuPG.

7.3 Keyserver

- *Keyserver.net* `http://www.keyserver.net`
- `http://wwwkeys.eu.pgp.net`

7.4 Libri

- B. Schneier, "Applied Cryptography, Second Edition", Wiley, 1996

8 Informazioni su questo documento

8.1 Versioni

Versione 0.1 (tedesco)

- Questa è la prima versione in tedesco.

Versione 0.1.0 (inglese) 30 aprile 1999

- Questa versione è la traduzione in inglese della versione tedesca, con alcuni aggiustamenti.

Versione 0.1.1 (tedesco)

- Nuovo capitolo "Limiti alla sicurezza"
- Migliorata la spiegazione delle firme
- Modifiche dopo suggerimenti di Werner Koch (grazie!)

Versione 0.1.2 (inglese) 3 aprile 2002

- Corretti alcuni errori di stampa.
- Nuovo capitolo sui front-end.

Versione 0.1.3 (olandese) 17 maggio 2002

- Questa versione è una traduzione in olandese della versione inglese.

Versione 0.1.4 (italiano) 12 maggio 2003

- Questa versione è una traduzione in italiano della versione inglese.

For the English or Dutch version: All remarks for this document can be sent to Brenno J.S.A.A.F. de Winter (*brenno@dewinter.com*). or Arjen Baart (*arjen@andromeda.nl*). Comments help us make a better document and are greatly appreciated.

For the German version: Anregungen, Kritik, Verbesserungen und Erweiterungen einfach an Michael Fischer v. Mollard (*fischer@math.uni-goettingen.de*) senden, damit dieses Dokument weiter verbessert werden kann.

Per la versione italiana: inviare commenti e segnalazioni a Cristian Rigamonti (*cri@linux.it*).