A.A. 2006/2007
Sistemi Informativi

GPG E LA SICUREZZA NELLA POSTA ELETTRONICA

Federica Bardini
Emanuele Gaspari Castelletti

Indice degli argomenti:

1 - INTRODUZIONE

1.1 - Cos'è GnuPG

GnuPG è l'implementazione libera e completa dello standard OpenPGP definito dalla RFC 2240, offerta dal progetto GNU. GnuPG permette di cifrare e firmare digitalmente i propri dati e le proprie comunicazioni, offre un versatile sistema di gestione delle chiavi e moduli per l'accesso a tutti i tipi di elenchi di chiavi pubbliche. GnuPG, noto anche come *GPG*, è uno strumento a riga di comando che si integra facilmente con altre applicazioni, grazie alla disponibilità di molti frontend e librerie. La versione 2.0 di GnuPG offre anche supporto per S/MIME.
GnuPG è un software libero (nel senso che rispetta la vostra libertà). Può essere liberamente usato, modificato e distribuito sotto i termini della Licenza Pubblica Generica GNU.
GPG venne sviluppato inizialmente da Werner Koch: la versione 1.0.0 fu rilasciata il 7 Settembre 1999.
GnuPG attualmente è disponibile in 2 versioni: la 1.4.7, la versione stabile e portabile meglio conosciuta e la 2.0.4, la versione migliorata ma ancora instabile.
Nel 2000 il ministro dell'Economia e della Tecnologia della Germania Federale fa partire il progetto Gpg4Win, per la creazione della documentazione e del porting di GPG per Microsoft Windows; attualmente la versione di GnuPG per Windows comprende un installatore, vari frontend e manuali (per ora in tedesco).
Il progetto Aegypten ha sviluppato le funzionalità di S/MIME per GnuPG 2.0.

GPG è un programma stabile e maturo, ed è distribuito con parecchi sistemi operativi liberi come FreeBSD , OpenBSD, e NetBSD, e ovviamente con tutte le distribuzioni GNU/Linux. È disponibile anche per le varie versioni dei sistemi operativi proprietari Microsoft Windows e Mac Os X, e grazie alla sua portabilità ed alla disponibilità del codice sorgente è possibile crearne una versione per qualsiasi OS.
Nonostante la versione di base di GPG fornisca un'interfaccia a linea di comando completa, sono state sviluppate parecchie interfacce grafiche e metodi per integrarlo all'interno di programmi di posta elettronica, come Kmail, Evolution (i client e-mail di default per, rispettivamente, KDE e GNOME). Un plug-in apposito, Enigmail, permette l'integrazione con Mozilla e Thunderbird, semplificando l'utilizzo di GPG (per la posta) sotto Microsoft Windows, GNU/Linux ed altri sistemi operativi.

1.2 – Perchè utilizzare GPG

Attualmente si pensa alla posta elettronica, ed in generale al documento in formato digitale, alla stregua di un normale ed affidabile documento cartaceo; purtroppo questo non è vero per diversi motivi.

Parlando di posta elettronica, i protocolli e gli standard con cui viene trattata sono ideati ed

implementati per consentire la comunicazione attraverso un canale per assunto sicuro e fra alleati.

Questo era vero quando la Rete era agli inizi e si chiamava Arpanet, l'antenata destinata alla comunicazione fra le università e gli istituti di ricerca; non lo è più da quando i messaggi transitano sull'odierna Internet, che occorre iniziare a pensare come un luogo ostile.

I punti deboli più evidenti del protocollo possono essere riassunti nei seguenti punti:

- il mittente e il destinatario possono essere falsificati e cambiati senza troppa difficoltà sia in partenza che durante il trasferimento
- con la stessa facilità, il messaggio può essere letto durante tutto il percorso, senza alterarne in alcun modo il contenuto o l'aspetto
- non è previsto un meccanismo di verifica dell'integrità del messaggio, che può essere modificato in un momento qualsiasi del trasferimento fra mittente e destinatario, senza lasciare tracce evidenti
- possono essere fatte copie assolutamente identiche del messaggio senza che si noti differenza alcuna

Tutto questo dimostra che la posta elettronica, così com'è, non è sicura.

Gli stessi problemi li abbiamo di fronte ad un qualsiasi altro documento in formato digitale, in quanto questo è semplicissimo da modificare senza lasciare alcuna traccia.

Allo stesso modo non è dato sapere se il documento è stato effettivamente scritto, inviato o approvato dal mittente dichiarato nel documento stesso o indicato nel messaggio di posta elettronica che lo trasporta.

Allo stato attuale delle cose quindi gli strumenti di cui ci serviamo quotidianamente non sono sufficienti a fornirci un adeguato livello di sicurezza e riservatezza; per questo è necessario fare affidamento su programmi di crittografia quali GPG.

1.3 - COME FUNZIONA GPG

La crittografia può rispondere in parte alle basilari necessità di riservatezza e di autenticità. Non è la soluzione definitiva, che probabilmente non esiste e non esisterà mai, ma può almeno portare la posta elettronica ed i documenti digitali al livello minimo di riservatezza che possiede una lettera in busta chiusa.

La crittografia si occupa delle "scritture segrete", cioè modalità di scrittura non leggibili da chiunque ma solo da chi è in possesso di una informazione segreta, indicata con il termine "*chiave*" (key).

Con la potenza della matematica, basata principalmente su applicazioni di teoria dei numeri, si

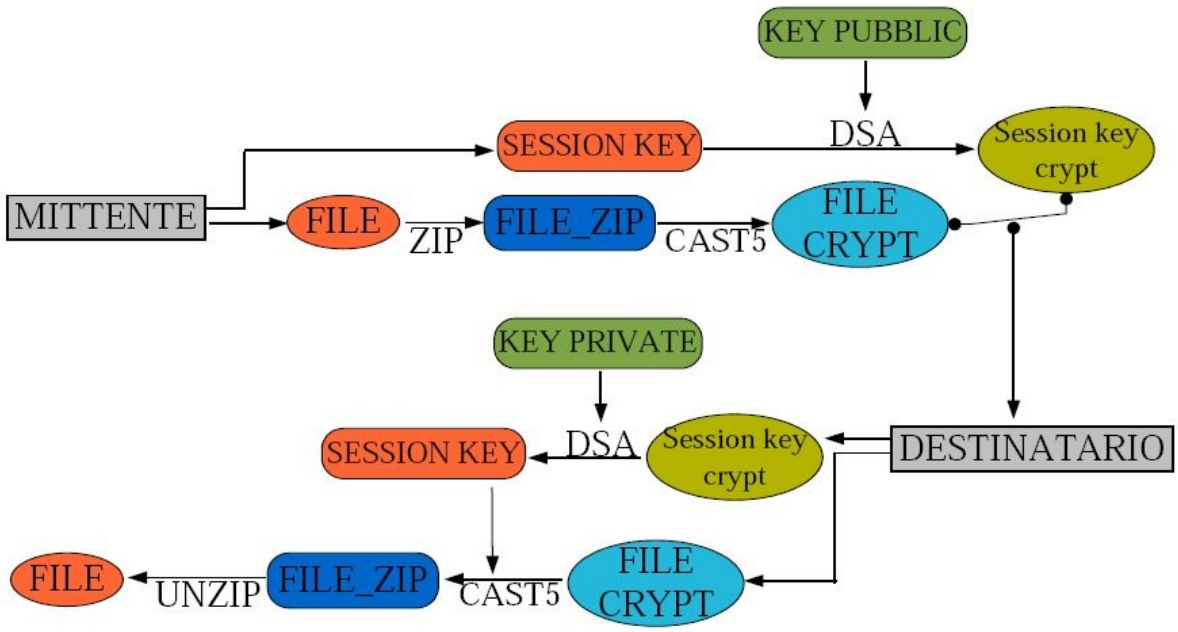
costruiscono algoritmi molto affidabili e sicuri poiché basati su teoremi ed applicazioni numeriche assolutamente dimostrabili.

GPG utilizza algoritmi ibridi, cioè algoritmi che utilizzano sia tecniche di crittografia asimmetriche (algoritmi a chiave pubblica) che simmetriche; questo perchè molti algoritmi simmetrici sono più forti dal punto di vista della sicurezza: infatti le operazioni di criptazione e decriptazione a chiave pubblica sono più costose delle corrispondenti operazioni dei sistemi simmetrici.
In particolare esso funziona utilizzando un algoritmo a chiave pubblica per condividere una chiave per il sistema simmetrico.

La crittografia a chiave asimmetrica prevede da parte di entrambe le parti la generazione di una coppia di chiavi collegate fra loro: una chiave pubblica ed una privata. La chiave pubblica del destinatario serve al mittente per cifrare una chiave comune (detta anche chiave di sessione) per un algoritmo di crittografia simmetrica; questa chiave viene quindi usata per cifrare il testo in chiaro del messaggio. Molte chiavi pubbliche di utenti GPG sono a disposizione di tutti dai numerosi key server (server delle chiavi) GPG sparsi per il mondo, che operano come mirror reciproci.

Il destinatario di un messaggio protetto da GPG lo decifra usando la chiave di sessione con l'algoritmo simmetrico, che è inclusa nel messaggio in maniera criptata e viene decifrata usando la chiave privata del destinatario. Ci sono inoltre delle vulnerabilità crittografiche nell'algoritmo a chiave asimmetrica utilizzato da GPG quando viene usato per cifrare direttamente un messaggio.

Vediamo nel dettaglio la procedura di cifratura:



1. il file del mittente viene compresso con un algoritmo di tipo zip

2. il file compresso viene criptato mediante l'algoritmo CAST5 (oppure un altro algoritmo, a seconda delle preferenze impostate)
3. viene generata una sequenza casuale di 128 bit (o più) chiamata session key con appositi algoritmi per ottenere una sequenza di numeri pseudo casuali che hanno determinate proprietà statistiche, devono essere equiprobabili (come se si tirasse un dado: escono numeri da 1 a 6)
4. la session key viene cifrata con l'algoritmo DSA (oppure con l'algoritmo RSA) utilizzando la chiave pubblica del destinatario ed il risultato è concatenato al documento
5. la session key criptata viene concatenata al file criptato infine, viene applicato l'algoritmo di trasformazione reversibile a testo ASCII chiamato Armor Radix-64 (a volte indicato Base64 o Armored). Questo algoritmo produce un documento formato da solo caratteri ASCII compatibile con tutti i server e client di posta elettronica.
6. Il destinatario esegue i passaggi al contrario.

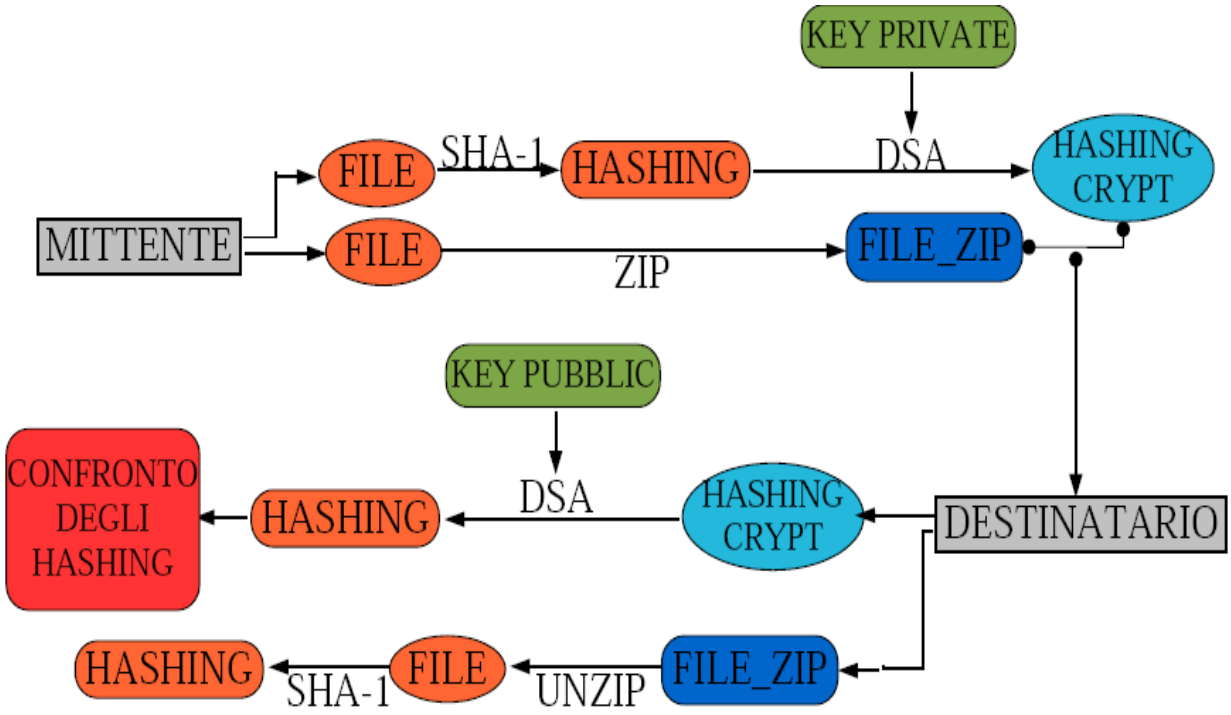
Una strategia simile può essere usata per capire se un messaggio è stato alterato, o se è stato mandato effettivamente da chi dice di essere il mittente. Per fare entrambe le cose, il mittente usa GPG per 'firmare' il messaggio con l'algoritmo di firma RSA o DSA. Per fare questo, GPG calcola un 'hash' (anche chiamato message digest) dal testo in chiaro, e da questo crea poi la firma digitale usando la chiave privata del mittente.

Il destinatario del messaggio calcola l'hash dal testo in chiaro decifrato, e poi usa la chiave pubblica del mittente ed il valore del message digest firmato con l'algoritmo di firma. Se la firma corrisponde al message digest del testo in chiaro ricevuto, si presuppone (con un grande margine di sicurezza) che il messaggio ricevuto non sia stato alterato né accidentalmente né volontariamente da quando è stato firmato.

Questa presunzione si basa su diverse considerazioni: è poco probabile, visti gli algoritmi ed i protocolli usati in GPG, che un avversario possa creare una firma per un messaggio qualsiasi, e quindi un messaggio ricevuto che superi questo test dev'essere stato mandato da chi dice di essere il mittente. Gli esperti in sicurezza infatti stabiliscono che la firma digitale assicuri:

- **autenticità** => viene intesa come "quella è proprio la mia firma"
- **integrità** => viene intesa come "il documento che ho firmato è proprio quello"
- **non ripudio** => viene inteso come "è firmato da te, non puoi dire che non è tua la firma"

Vediamo nel dettaglio i passaggi per la firma digitale:



1. il file del mittente viene compresso con un algoritmo di tipo zip
2. viene calcolato l'hashing del file mediante l'algoritmo SHA-1 (oppure con uno diverso). Il codice hash generato viene criptato tramite DSA con la chiave privata del mittente ed accodato al file di origine
3. eventualmente il file risultante dalle due operazioni precedenti viene trasformato in codice ASCII Armored
4. il destinatario effettua le operazioni al contrario è verifica se i codici di hashing sono identici.

Sia quando si cifra un messaggio che quando si verifica la firma, è fondamentale che la chiave pubblica utilizzata per mandare il messaggio ad una persona o ente appartenga effettivamente al destinatario. Scaricare una chiave pubblica da qualche parte non è per niente una garanzia di questa corrispondenza; lo spoofing (cioè il furto di identità) intenzionale od accidentale è possibile.

GPG include delle precauzioni per la distribuzione delle chiavi pubbliche introducendo il concetto di *modello di certificazione*, cioè il metodo con cui viene verificata l'identità del proprietario e certificata, appunto, l'associazione con la sua chiave pubblica.

L'organizzazione del sistema di certificazione viene spesso indicato con l'acronimo PKI (*Public Key Infrastructure*), che si fa carico sia della distribuzione delle chiavi pubbliche, che di fornire la prova di appartenenza delle chiavi stesse.

Al momento ne esistono di due tipi, usati e conosciuti:

- il modello ad Autorità di Certificazione (*Certification Authority*), in cui una organizzazione ha il compito di rilasciare sistemi di firma accertando l'identità delle persone e garantendo nei confronti degli altri. Per verificare la firma digitale fatta in questo modo devo riferirmi all'Autorità che ha rilasciato la firma e garantisce per il firmatario, certificando appunto l'identità e il possesso
- il modello a Rete della Fiducia (*Web Of Trust*), un po' più complesso. L'idea di base si appoggia sul concetto dei *sei gradi di separazione* ed alla *Teoria del Mondo Piccolo*: in sostanza, ipotizzando di costruire una catena di amici degli amici degli amici, sembra che sia necessaria una catena di sole *sei* persone per arrivare a qualsiasi altro essere umano attualmente vivente sulla Terra.
  Questo concetto viene sfruttato dal modello a Rete di Fiducia in questo modo: se mi giunge un messaggio firmato da qualcuno che non conosco, potrei però conoscere un suo amico di cui mi fido. Sulla base di questa idea, ognuno può convalidare l'associazione fra una chiave pubblica e la persona che la detiene quando la conosca di persona o quando l'abbia incontrata almeno una volta in una occasione particolare (un *Key Signing Party*). Di conseguenza, chi non conosca di persona colui che ha mandato il messaggio, potrebbe però conoscere qualcuno che lo ha incontrato, e quindi fidarsi di riflesso.

La principale differenza tra i due è che nel primo caso una sola entità certifica identità e associazione fra chiave e possessore, mentre nel secondo questa certificazione è distribuita fra i partecipanti all'infrastruttura, e nessuno ha a priori più autorità di altri.

Questo risulta essere in parte uno svantaggio per il primo modello, in quanto dal punto di vista puramente concettuale, è molto più semplice ingannare una entità singola che ingannarne molte indipendenti, per cui, andando a buon senso, il meccanismo della rete della fiducia è più difficile da indurre in errore, dato che l'identità di una persona viene controllata più volte, da persone diverse, in occasioni differenti.

GPG include comunque un modo per cancellare i certificati d’identità che possono essere diventati inutilizzabili, a causa di furti dei supporti USB e simili; questo è, più o meno, l’equivalente dei *certificate revocation list* (certificati di revoca) dei sistemi PKI più centralizzati; inoltre le recenti versioni di GPG supportano anche una data di scadenza per i certificati.

2 - ALGORITMI UTILIZZATI DA GPG

Analizziamo un esempio delle tre tipologie di algoritmi utilizzate da GPG: algoritmi a chiave pubblica (DSA&ElGamal), a chiave simmetrica (CAST5), e funzioni hash (SHA).
Inoltre vedremo anche l'algoritmo ASCII ARMOR RADIX-64 per formattare l'output.

2.1 - CAST5

CAST5 è un cifrario simmetrico a blocchi usato in una grande quantità di prodotti software, in particolare in GPG dove è usato come algoritmo di default. Questo algoritmo è stato creato nel 1996 da Carlisle Adams e Stafford Tavares usando la procedura di progettazione del CAST.
Secondo alcune fonti il nome "CAST" deriva dalle iniziali dei suoi creatori, sebbene Bruce Schneier affermi che l'idea del creatore era che "il nome avrebbe dovuto evocare immagini di casualità".

CAST5 è un cifrario Feistel di 12/16 cicli, composto di blocchi di 64 bit e con una chiave di dimensione compresa tra i 40 e i 128 bit (con possibili incrementi però solo di 8 bit).
I 16 cicli completi sono usati quando la chiave è più lunga di 80 bit.

I suoi componenti includono S-boxes con dimensioni 8x32 bit basate sulle bent functions, rotazioni dipendenti dalla chiave, addizioni e sottrazioni modulari e le operazioni di XOR.
CAST5 utilizza alternativamente tre tipi di funzioni cicliche, che hanno una struttura molto simile e differiscono solo nella scelta dell'esatta operazione (addizione, sottrazione, XOR) in diversi punti.

Sebbene l'Entrust detenga una licenza sulla procedura di creazione di CAST, CAST5 è disponibile in tutto il mondo su basi di mercato libere per usi commerciali e non.

2.2 - SHA

SHA (Secure Hash Algorithm, anche **SHS**, Secure Hash Standard): è un algoritmo hash pubblicato dal governo USA e produce un valore a 160 bit a partire da una stringa di lunghezza arbitraria. E' comunemente considerato abbastanza sicuro ed è relativamente nuovo.

Le **Funzioni Hash** di **SHA** fanno riferimento a cinque algoritmi approvati dalla **FIPS** (Federal Information Processing Standards) per ottenere in output una rappresentazione digitale ridotta (conosciuta come **message digest**) la quale è unica per una data sequenza di input (**message**) con un alto grado di probabilità.

Questi algoritmi sono definiti "sicuri" perchè "*per un dato algoritmo, è computazionalmente impossibile: 1) ricavare un messaggio a partire da un message digest e 2) trovare due differenti*

messaggi che producono lo stesso message digest. Qualsiasi cambiamento ad un messaggio, con una probabilità molto alta, comporterà un cambiamento del message digest."

I cinque algoritmi, chiamati *SHA-1*, *SHA-224*, *SHA-256*, *SHA-384 e SHA-512*, sono funzioni crittografiche di hash concepite dalla National Security Agency (NSA) e pubblicate dalla NIST sottoforma di standard governativo degli U.S. Le ultime quattro varianti sono solitamente denominate SHA-2.
SHA-1 è largamente impiegato in applicazioni e protocolli, incluso TLS e SSL, PGP, SSH, S/MIME e Ipsec. É stato considerato il successore dell'MD5, una funzione di hash usata agli inizi.

Alcune volte la sicurezza dell'SHA-1 è stata compromessa da ricercatori in campo crittografico, mentre nessun attacco ha avuto successo sulle varianti SHA-2 sebbene siano algoritmicamente simili. Considerati i recenti attacchi all'SHA-1, NIST sta concentrando gli sforzi su uno o più algoritmi hash attraverso progetti di carattere pubblico, similmente al processo che ha portato alla creazione di AES (Advanced Encryption Standard), per giungere ad un nuovo standard approssimativamente nel 2012.

2.3 - DSA e ELGAMAL

Digital Signature Algorithm (DSA) è uno standard FIPS per la firma digitale. Proposto dal National Institute of Standards and Technology (NIST) nell'agosto del 1991 per essere impiegato nel Digital Signature Standard (DSS) viene definitivamente adottato nel 1993. Consiste in un sistema di crittazione a chiave pubblica che assomiglia al metodo Elgamal.

El Gamal è un sistema di cifratura a chiave pubblica, proposto dal ricercatore egiziano americano ElGamal nel 1985. Lo schema è basato sulla difficoltà del calcolo del logaritmo discreto. Come per tutti gli argomenti il trasferimento dei dati si suddivide in 3 fasi: la creazione delle chiavi, la cifratura e la decifratura.

Creazione delle chiavi

1. Si sceglie a caso un numero primo **P** molto grande (10^{100} cifre);
2. Si sceglie un numero **G < P**;
3. Si calcola $\mathbf{G^A \% P}$ dove **0 < A < N-2**;

Si vengono così a formare le due chiavi: quella pubblica (**P**, **G**, $\mathbf{G^A \% P}$) e quella privata (**A**)

Cifratura del messaggio (M)

1. Avendo la chiave pubblica e il messaggio **M < P** si sceglie un numero **1 < B < P-2**;
2. Si calcola $\mathbf{G^B \% P}$;
3. Si calcola $\mathbf{M*(G^A)^B \% P}$;
4. Si inviano i dati criptati ($\mathbf{G^B \% P}$, $\mathbf{M*(G^A)^B \% P}$)

Decifratura del messaggio

Per la decifratura, sapendo l'inverso di **A** (facilmente ottenibile possedendo **A** e **P**) basta

1. Calcolare $(G^{B})^{-A}$ **%P**
2. Eseguire la semplice operazione $M*G^{BA}G^{-BA} = M$

Tutte le operazioni coinvolte sono algoritmicamente fattibili, in maniera efficiente. Il costo computazionale di Encryption e Decryption sono paragonabili all'RSA però questo algoritmo è resistente ad attacchi di tipo criptoanalitico: l'unico modo di ricavare informazioni segrete dai dati pubblici è effettuare il logaritmo discreto di o^{a} oppure di o^{k} mod p.
Ancora oggi non è conosciuto un algoritmo efficiente per calcolare tali valori.

2.4 - ASCII ARMOR RADIX-64

Il formato *ASCII Armored* (Base64 o Armor Radix-64) è un algoritmo definito nello standard di comunicazione OpenPGP e consiste nel produrre un file formato da solo caratteri ASCII compatibili con tutti i server e client di posta elettronica.

Questo formato converte un input a gruppi di 24 bit come una stringa di 4 caratteri codificati ASCII. I 24 bit vengono suddivisi in gruppi di 8 bit (quindi 3 byte) concatenati fra loro, e poi ogni gruppo di 6 bit è codificato con un carattere stampabile compreso tra un indice da 0 a 63.
Alla fine si ottengono dunque 4 caratteri stampabili.
Ogni singola linea, nel file di output, non avrà più di 76 caratteri.
Questi 64 caratteri fanno parte del cosiddetto ASCII 'basso', cioè i caratteri rappresentati dai numeri decimali nell'intervallo 0-127 (o esadecimali da 00 a 7F oppure ancora 27-1 bit).

Questa è la tabella di conversione del codice Armor Radix-64:

Value	Encoding	Value	Encoding	Value	Encoding	Value	Encoding
0	A	17	R	34	i	51	z
1	B	18	S	35	j	52	0
2	C	19	T	36	k	53	1
3	D	20	U	37	l	54	2
4	E	21	V	38	m	55	3
5	F	22	W	39	n	56	4
6	G	23	X	40	o	57	5
7	H	24	Y	41	p	58	6
8	I	25	Z	42	q	59	7
9	J	26	a	43	r	60	8
10	K	27	b	44	s	61	9
11	L	28	c	45	t	62	+
12	M	29	d	46	u	63	/
13	N	30	e	47	v		
14	O	31	f	48	w	(pad)	=
15	P	32	g	49	x		
16	Q	33	h	50	y		

La scelta di trasformare un file in codice ASCII Armored oppure no è a scelta dell'utente (in alcuni casi questa operazione è svolta dai sistemi software o hardware in modo trasparente se non l'ha fatto in modo esplicito l'utente).
In generale è consigliabile sempre utilizzare la conversione Radix-64, in quanto ciò evita problemi di visualizzazioni dei caratteri su sistemi differenti (Linux e Windows, Linux e Mac, e via dicendo).

Per rendere meglio un'idea si osservi l'esempio di cifratura di un semplice file che contiene la frase:

```
-----------------------------------------------
questo è un esempio di cifrato/firmato
-----------------------------------------------
```

Il risultato in binario su sistema Linux sarebbe questo:

```
  ###3m#ƒß d[##þ5¿#îì€Áõ6S^5#¡!*#W®8®#ú8æc##lÒ®×í(# ? (#G
25~V#äoûÊçb ùs#¢õÓq kcS, Eæ>¶Á ãjiÓMÌ8À. vàaÁ##Ÿ§ïð### #ÿK # .Í
ÿvT# Îš;S®úY@aÛa9##
  ð ê ßm¯ uÎ{×ÒJ{v;¬ Óbž.
##4F_[ÂÑdöÃð7Q#* lŠ-]#k"#P þ] v£:vÜ¡#ÑïŸµN#ÿò¥6>Íȼ 6rÓ(Œ - h ä'ù
œË. Õ#{íû " ,          #RÌ#GG #Ôõ&ÿÂ#ý#          <ëX)Gœ# POŽ X ÂÖlv# Y ##Ç#g
òÝ;3ßç Å!É8#bÁúSÚ ¡ù*Ò¿+SS-`ÿ77« Òï@5#!ý U#ÄiW##ßþÊLêž %é#ÎÙ¶Ì×
(šµ3+ã#HQ#ú}#ZWÐiß¯ *k f#$LL ËÈ·žTl586uoIþ2#4          ]LØžö# t9ìqa ŸÎ :S./
ÁåËÅ#žÑafH%ÇÚâDtø# q ¿ ZðAí^ÊiPÊ{~#ïxu bš: Š ;
¡œêÜ<É#cFçBÂ #n#8' mV \#; ¢ Sæ : Îq#xÔ-ó9,OêCúÃÒÀ##`Þ##3 d=30#gÕõ
#hÃ<°bï#Ó#Ç\4Ô#P-aò#ÎÅ®m# €l2 hRÒ¬Ï3~?mµÔÏô #ì#w xî» FP ¬¥Ç,~Á`œ]
ÎÉ#u~ò a¢PÈ {9yÝ×z ÆO1to}«¬< ,$Rd ÿ ÿä§·ç#ŒM#m+úW,E#~#`          âsµ2#ç5
É ^ÚKv |##û#çÙYèo Ûž_ë®:ÂòèYÌÌ¥"mÓã #i°#Q>ñ ×]%
```

Mentre in formato Radix-64 è:

```
-----BEGIN PGP MESSAGE-----
Version: GnuPG v1.2.4 (GNU/Linux)
hQIOAzNtCK/fgmRbEAf9FeYUFXNUZv/vmTPclQhiQ1lYe4do8y8delFDiJRwFIYA
59Zaz89V5z/mhDeoXWOsQ/viw38Bju44o5lAA2MZ5OHUpyT8oGb6k7NGRkYAVYwg
lzOhq7uYTXV4tnnTzoX5UF+y4+Cr0s3wyuWZ3aJpPaDhmf5AQUmGYmrYle/jvqcs
5U4oLgKMbmcNq64TEMYKE/YKPBCf53BZrwHcNlMKERYVjLWmCWKXAgQz06fqjS
3R
D44IiMocu+2F7F69aM+yIRB0wtdHsTu6l2PRJz/BT/ZryNGBtz63bNO5Ian7RIrK
u/XEDTGVBSetxsjGwVIM0yPrglMAQlcVKNMXgv9MpQf/fvo0Nq5geZnEYPOYupwi
ihIfDfdCul6D8Wk8DtoeKiScl8hiSyZHOHFOv5H0Q4ofYrlDsxjEcfn+wIKHPHtR
W+VRFa09fSbqHs8tjl+RoW9iZLdQGmBFsKdCJjVfcttlGLuspbShBuojT5WvuAWo
8jtyhe0oPjERbBYpb38TfPBiDLdBokM/zts/J8UdaStO5kN6GZ4xyBF/oqjRgVXA
LSgV/p20JQqKQLEgrYOTwb898opIjcqoC7dHXBzUNRUq/okH9U9vr9VwHri4gUWk
CROISi/mmRh/3q8gXc5+c70SBVkqsPB9CM/25apsIekKLi3XTg9Uh1E9MQOL/qhL
```

```
ItLAAgGKtu6surBZPoT32eitGZ1zMgiaXp/eXIk0bGFYs9+VbYT1m2hf1Fv9DxYf
s8PB88AkLiXzT0n7g3ipImCXLeVKU4G2quu4u/16dYG4gtzGW/wXdxwBP4WVOSiD
UHAYGHsPhiO/J5l4pUMZJc67o0d+8WNhncNwNvateW3LuWD1SfWoCREnvt1tpzAE
nSN9/rd3uaCAB6dIfEcJq3sxELYw6Wg4oefOq1jaCfzNLMsl89HV0fBVTAmtgCHA
ou5fV6+s
=iZXt
-----END PGP MESSAGE-----
```

Ad esempio, utilizzando la versione compilata per Mac (o per Windows) di GPG potrebbero esserci dei problemi di visualizzazione con il formato binario di Linux, mentre utilizzando Radix-64 il problema non si pone.

3 - USO DI GPG

3.1 - SICUREZZA

La chiave privata, i dati di convalida della firma nostra e delle firme dei nostri amici e conoscenti sono conservati su normali file, anche se in forma cifrata e questo è uno dei punti deboli di tutto il nostro sistema crittografico: occorre quindi che siano protetti al massimo contro il pericolo che qualcuno possa trafugarli.

Inoltre, dato che per definizione la chiave privata è unica e non ripetibile, la sua eventuale perdita è irrecuperabile, e siccome abbiamo detto che i file in cui GPG memorizza tutti i dati sono comuni file, questo significa che possono essere sottoposti a backup: GPG infatti non trova differenze a lavorare sugli originali o sulle copie.

Detto questo dobbiamo far coincidere esigenze contrastanti:

- protezione dei file da accessi non autorizzati => conservazione in un unico posto sicuro
- protezione dalla eventuale perdita o cancellazione dei file => copie di riserva in più supporti in posti differenti

Si può risolvere questo contrasto utilizzando come supporto un disco USB removibile, dal normale hard disk al disco *flash* a stato solido, la familiare “chiavetta USB”, da portare magari insieme alle chiavi di casa o dell'auto.

Si istruisce GnuPG ad usare il percorso giusto per i file di lavoro, prendendoli dal disco removibile, così se ci dimentichiamo di connetterlo al momento della firma, GnuPG avvisa soltanto che non riesce a trovare quello che serve, senza fare danni.

In questo modo sul computer non è conservato nulla, e se mai dovesse essere smarrito o trafugato, dentro non ci sarebbero dati così importanti.

L'elemento critico diviene in questo modo il disco USB, che va protetto da eventuali letture o manomissioni da parte di estranei, utilizzando eventualmente strumenti come la cifratura del filesystem, permessa sia dalle ultime versioni di Windows che da Linux.

Il problema della realizzazione dei backup dei file di lavoro di GnuPG può essere risolto tramite l'uso della crittografia a chiave simmetrica: anche a seguito della completa distruzione dei file di lavoro di GnuPG, per esempio a causa di un guasto al disco fisso, è possibile recuperare i file dal backup usando la sola chiave di cifratura, senza disporre delle chiavi private, che non sono necessarie con la cifratura a chiave simmetrica.

3.2 - INSTALLAZIONE

GnuPG presenta molte interfacce grafiche, quali KGPG per KDE e SeaHorse per GNOME, ma noi tratteremo solo la parte riguardante la riga di comando, in quanto i front-end non hanno la stessa potenza e completezza dei comandi e delle opzioni disponibili da terminale.

GPG, nelle varie distribuzioni di GNU/Linux è generalmente già installato; se non dovesse esserlo basterà fare riferimento alla documentazione della propria distribuzione e seguire le indicazioni.
L'unica cosa che rimane da fare, nel caso vogliamo usare il disco USB come supporto, è la creazione di un link simbolico nella home directory che punti al mount point del disco USB.

Creiamo innanzitutto la directory da collegare sul disco:

```
[pinco@pclinux ~]$ mkdir /media/usb/.miafirma
```

Ora creiamo il link simbolico che farà puntare la directory .gnupg (si trova nella home dell'utente e contiene i file di lavoro) alla directory sul disco USB:

```
[pinco@pclinux ~]$ ln -s /media/usb/.miafirma .gnupg
```
(questo comando deve essere lanciato dalla home directory)

3.3 - GENERAZIONE DELLE CHIAVI

Dobbiamo innanzitutto verificare quale versione è installata sul nostro sistema (generalmente sarebbe bene utilizzare sempre l'ultima versione stabile del programma):

```
[pinco@pclinux ~]$ gpg --version
gpg (GnuPG) 1.4.7
Copyright (C) 2006 Free Software Foundation, Inc.
This program comes with ABSOLUTELY NO WARRANTY.
This is free software, and you are welcome to redistribute it
under certain conditions. See the file COPYING for details.

Home: ~/.gnupg
Algoritmi gestiti:
A chiave pubblica: RSA, RSA-E, RSA-S, ELG-E, DSA
Cifrari: 3DES, CAST5, BLOWFISH, AES, AES192, AES256, TWOFISH
Hash: MD5, SHA1, RIPEMD160, SHA256, SHA384, SHA512, SHA224
Compressione: Non compresso, ZIP, ZLIB, BZIP2
```

Vengono elencati di seguito tutta la serie di algoritmi gestiti dal programma, sia quelli simmetrici che asimmetrici, fino agli hash; questo è molto utile nel caso ci si scambi dati con persone che hanno differenti versioni del software.

La procedure vera e propria per la creazione delle chiavi è questa:

```
[pinco@pclinux ~]$ gpg --gen-key
gpg: ATTENZIONE: si sta usando memoria insicura!
gpg: visitare http://www.gnupg.org/faq.html per ulteriori informazioni
gpg: directory `/home/pinco/.gnupg' created
gpg: creato un nuovo file di configurazione `/home/pinco/.gnupg/gpg.conf'
gpg: ATTENZIONE: le opzioni in `/home/pinco/.gnupg/gpg.conf' non sono ancora
attive durante questa esecuzione del programma
gpg: portachiavi `/home/pinco/.gnupg/secring.gpg' creato
gpg: portachiavi `/home/pinco/.gnupg/pubring.gpg' creato
Per favore scegli che tipo di chiave vuoi:
   (1) DSA e ElGamal (default)
   (2) DSA (firma solo)
   (4) RSA (firma solo)
Cosa scegli?
```

I file secring.gpg e pubring.gpg sono chiamati keyring (portachiavi), in quanto contengono rispettivamente le chiavi private (usate per firma e crittografia) e quelle pubbliche (le nostre e quelle dei conoscenti).
Si possono scegliere più tipologie di chiavi, a seconda degli algoritmi implementati: noi scegliamo quella di default, che permette sia la firma, sia la cifratura di dati generici.
Successivamente viene richiesta la dimensione ed il tempo di validità della chiave; viene poi costruito tramite software un user-ID che identifica la chiave (si genera inserendo un nome, un indirizzo mail ed un commento).
A questo punto è necessario scegliere una password di accesso alle chiavi private, la passphrase e poi il processo di generazione dei dati necessari (compresa la creazione del trustdb – database della fiducia alla base del Web of Trust) si compie in automatico.

```
gpg: /home/pinco/.gnupg/trustdb.gpg: creato il trustdb
gpg: key E4F4B420 marked as ultimately trusted
chiavi pubbliche e segrete create e firmate.

gpg: controllo il trustdb
gpg: 3 marginal(s) needed, 1 complete(s) needed, PGP trust model
gpg: depth: 0  valid:   1  signed:   0  trust: 0-, 0q, 0n, 0m, 0f, 1u
gpg: il prossimoi controllo del trustdb sarà fatto il 2037-03-13
pub   1024D/E4F4B420 2007-03-21 [expires: 2037-03-13]
      Key fingerprint = 9294 2CD5 02E3 7C21 7C19  F6D3 7372 A61F E4F4 B420
uid                  Pinco (uno qualsiasi) <pinco@mail>
sub   2048g/5B514DF0 2007-03-21 [expires: 2037-03-13]
```

Come si può notare alla chiave viene associato un fingerprint, “l'impronta digitale” unica e strettamente associata ad essa: se qualcuno manomettesse la chiave sarebbe immediatamente riconoscibile dal diverso fingerprint.

Se in seguito si vogliono ottenere le proprietà della chiave creata (nome a cui è associata, fingerprint, ecc.) si deve digitare il comando:

```
[pinco@pclinux ~]$ gpg --fingerprint Pinco
pub   1024D/E4F4B420 2007-03-21 [expires: 2037-03-13]
      Key fingerprint = 9294 2CD5 02E3 7C21 7C19  F6D3 7372 A61F E4F4 B420
uid                  Pinco (uno qualsiasi) <pinco@mail>
sub   2048g/5B514DF0 2007-03-21 [expires: 2037-03-13]
```

3.4 - CREAZIONE DI UN CERTIFICATO DI REVOCA

Al momento della creazione delle chiavi, o anche successivamente, è possibile creare un certificato di revoca, una chiave speciale che consente di revocare la validità della nostra chiave segreta principale dalla data in cui viene usato il certificato.

Dato che la chiave di cifratura è convalidata dalla firma digitale apposta su di essa al momento della generazione, anche il valore di tutte le sotto-chiavi di cifratura decade nello stesso momento in cui la chiave di firma viene revocata.

```
[pinco@pclinux ~]$ gpg --output revoca.asc --gen-revoke Pinco

sec   1024D/E4F4B420 2008-05-21 Pinco (commento) <pinco@mail>

Create a revocation certificate for this key? (y/N)
```

Per indicare a quale chiave ci si riferisce si può usare il nome, l'indirizzo mail o, meglio, l'user-ID della chiave stessa; viene poi richiesto il motivo della revoca (con un eventuale commento) ed infine la passphrase.

A questo punto il file revoca.asc viene creato ed il risultato sarà molto simile a questo:

```
----BEGIN PGP PUBLIC KEY BLOCK-----
Version: GnuPG v1.4.7 (GNU/Linux)
Comment: A revocation certificate should follow

iGcEIBECACcFAkYCr3kgHQJDaGlhdmUgY29tcHJvbWVzc2EsIG5vbiB1c2FyZSEA
CgkQc3KmH+T0tCAsywCfRYAvJzR0hU/RsmmDzgOlaIuyv0EAnjCVteultuelWm/Z
cFMsui7XdL5T
=GYEU
-----END PGP PUBLIC KEY BLOCK-----
```

L'uso del certificato di revoca è definitivo: se viene inviato ad un keyserver le chiavi ad esso relative vengono annullate con effetto immediato.

3.5 - ESPORTAZIONE DELLE CHIAVI

Per rendere una chiave pubblica utilizzabile è necessario distribuirla, utilizzando i public keyserver su internet (ad esempio “pgp.mit.edu” e “subkeys.pgp.net”), che mettono a disposizione tutte chiavi pubbliche che gli utenti hanno volontariamente distribuito.

Si può pubblicare la chiave in due modi:

- si può esportarla in un file (nell'esempio chiave.asc) ed inviarla al server (solo per i server che mettono a disposizione un'interfaccia web per questo genere di operazioni):

```
pinco@pclinux ~]$ gpg --output chiave.asc --armor --export Pinco
```

- si impartisce un comando a GPG per l'invio diretto al server:

```
[pinco@pclinux ~]$ gpg --keyserver subkeys.pgp.net --send-key Pinco
gpg: inviata con successo a `subkeys.pgp.net' (status=200)
```

Non è strettamente necessario inviare la chiave ad un keyserver per esportarla: si può anche incontrare la persona con cui si vuole effettuare lo scambio e consegnargli un foglio con i dati (ad esempio durante un key signing party).

3.6 - CONVALIDA DELLE CHIAVI

Dopo essersi scambiati la chiave la si può prelevare dal keyserver eseguendo il comando:

```
[pinco@pclinux ~]$ gpg --keyserver pgp.mit.edu --recv-key 0x3d739f0d
gpg: key 3D739F0D: public key "Caio <caio@server>" imported
gpg: Numero totale esaminato: 1
gpg:               importate: 1
```

In seguito si verifica l'autenticità della chiave, confrontando il fingerprint su terminale con quello ricevuto dalla persona:

```
[pinco@pclinux ~]$ gpg --fingerprint Caio
pub   1024D/3D739F0D 2007-03-21
      Key fingerprint = 1A50 D735 18A5 AA5B 2F65  0D76 BB51 4ED0 3D73 9F0D
uid                  Caio <caio@server>
sub   2048g/32F7C2EE 2007-03-21
```

Per rendere definitiva la verifica e rendere pubblico il fatto che si conosce la persona e quella è proprio la sua chiave pubblica si devono eseguire due operazioni:

- firma della chiave pubblica della persona in questione:

```
[pinco@pclinux ~]$ gpg --sign-key Caio
```

- invio al keyserver della chiave pubblica della persona aggiornata con la propria firma di convalida:

```
[pinco@pclinux ~]$ gpg --keyserver pgp.mit.edu --send-keys Caio
gpg: inviata con successo a `pgp.mit.edu' (status=200)
```

Il tempo di aggiornamento delle chiavi è variabile, da pochi secondi se i due interessati condividono lo stesso keyserver, ad alcuni giorni se il keyserver è diverso.

Per aggiornare la propria chiave pubblica scaricando le nuove firme aggiunte basta dare il comando:

```
[pinco@pclinux ~]$ gpg --refresh-keys Pinco
```

3.7 - FIDUCIA E WEB OF TRUST

Quando otteniamo la chiave pubblica di un'altra persona e la controfirmiamo, per completare la procedura bisogna aggiungere il livello di fiducia che si ha in questa persona.

Non si tratta però di una fiducia generica, ma di fiducia basata sulla comprensione che secondo noi l'altra persona ha del meccanismo del Web of Trust e sulla diligenza che applica nella verifica delle chiavi e delle identità altrui prima di controfirmarle.

Infatti fidarsi, nell'ambito della firma digitale con questo modello di certificazione, significa accettare una firma di convalida fatta da un nostro amico come se fosse nostra, cioè come se avessimo incontrato di persona l'altro e verificato noi stessi la chiave e l'identità.

La procedura per cambiare il livello di fiducia della persona è questa:

```
[tizio@miopc ~]$ gpg --edit pinco

pub  1024D/E4F4B420  created: 2007-03-21  expires: 2037-03-13  usage: SC
                     trust: sconosciuto   validity: full
sub  2048g/5B514DF0  created: 2007-03-21  expires: 2037-03-13  usage: E
[  full  ] (1). Pinco (uno qualsiasi) <pinco@mail>

Comando>
```

Questo comando di GPG mette a disposizione un'interfaccia molto semplice ed efficace che permette praticamente tutte le operazioni sulle chiavi, sia pubbliche che private (consultare il manuale per vedere quali comandi sono accettati); fornisce inoltre molte proprietà della chiave (quello che ci interessa è il valore “sconosciuto” dato a “trust”).

Digitando il comando “trust” si assegna un livello di fiducia:

```
Comando> trust
pub  1024D/E4F4B420  created: 2007-03-21  expires: 2037-03-13  usage: SC
                     trust: sconosciuto   validity: full
sub  2048g/5B514DF0  created: 2007-03-21  expires: 2037-03-13  usage: E
[  full  ] (1). Pinco (uno qualsiasi) <pinco@mail>

Please decide how far you trust this user to correctly verify other users' keys
(by looking at passports, checking fingerprints from different sources, etc.)

  1 = I don't know or won't say
  2 = I do NOT trust
  3 = I trust marginally
  4 = I trust fully
  5 = I trust ultimately
  m = back to the main menu
```

```
Cosa hai deciso?
```

GPG possiede un gruppo di regole predefinite per stabilire la validità di una chiave pubblica in base al livello di fiducia assegnato ad ogni persona; le regole sono configurabili a piacere, ma quelle predefinite sono:

- se una chiave è firmata da qualcuno in cui si ha fiducia piena è attendibile
- se la chiave è firmata da almeno tre persone con fiducia parziale è attendibile
- la profondità massima di una catena di firme convalidate con le due regole precedenti è di cinque salti; dal sesto in poi le firme saranno considerate inattendibili, indipendentemente dal numero e dalla fiducia riposta nelle firme che possiede la chiave in esame

Ovviamente tutti i dati riguardanti il livello di fiducia assegnato ad ogni chiave sono privati e strettamente personali; inoltre sono salvati nel file trustdb.gpg, il cui contenuto è accessibile solo a noi e a nessun altro.

Questo meccanismo di privatezza è costruito apposta per garantire la massima sincerità nel assegnare i livelli di fiducia e quindi una maggiore affidabilità del modello Web of Trust.

3.8 - USO DELLA FIRMA

Sappiamo che la firma digitale è un hash, cifrato con la chiave privata, quindi una sequenza di pochi byte: se il documento digitale che andiamo a firmare è un semplice file di testo puro, è possibile aggiungere la firma al messaggio stesso, in coda.
Il problema sorge ad esempio con file di applicazioni che usano un loro formato specifico per memorizzare i documenti, come OpenOffice Writer™ o Microsoft Word™, il cui formato di salvataggio dei file non consente modifiche arbitrarie al file stesso, anche solo per inserire i pochi byte di una firma digitale.
Nasce quindi il concetto di *detached signature*, firma separata, che altri non è che un file di pochi byte che contiene la sola firma digitale. Questo file è strettamente associato al documento firmato, nel senso che un differente documento non possiederà la stessa firma.

3.9 - FIRMA ALL'INTERNO DEL MESSAGGIO

Nel caso di un testo semplice, come può essere una e-mail, lo standard OpenPGP prevede tutto quello che serve, racchiudendo il testo del messaggio e la firma con intestazioni distinte.

Creiamo un messaggio semplice e lo salviamo nel file messaggio.txt; per firmarlo (lasciandolo però visibile) si utilizza questo comando:

```
[pinco@pclinux ~]$ gpg --clearsign messaggio.txt
```

```
You need a passphrase to unlock the secret key for
user: "Pinco (uno qualsiasi) <pinco@mail>"
1024-bit DSA key, ID E4F4B420, created 2007-03-21

Inserisci la passphrase:
```

Inserendo la password che sblocca la firma si ottiene un file in più dal nome messaggio.txt.asc che ha questo aspetto:

```
-----BEGIN PGP SIGNED MESSAGE-----
Hash: SHA1

Messaggio di prova.
Questo messaggio viene firmato con gpg
usando una firma internamente al messaggio.
Ciao
Pinco

-----BEGIN PGP SIGNATURE-----
Version: GnuPG v1.4.7 (GNU/Linux)

iD8DBQFGEOqic3KmH+T0tCARAl5JAJ9vYc2juhsgskIgJcQtXyTp1Zhl/wCfWlal
vy5wqAnDpD//Aivr4cAarjA=
=9Hi+
-----END PGP SIGNATURE-----
```

Chi riceve questo messaggio può verificarlo come segue:

```
[pinco@pclinux ~]$ gpg --verify messaggio.txt.asc
gpg: Signature made lun 02 apr 2007 13:36:02 CEST using DSA key ID E4F4B420
gpg: Good signature from "Pinco (uno qualsiasi) <pinco@mail>"
```

3.10 - LA FIRMA A PARTE (DETACHED SIGN)

Nel caso si voglia spedire invece un file che non contenga testo semplice (ad esempio un programma, un file.doc, ecc.) si deve come prima cosa firmare:

```
[pinco@pclinux ~]$ gpg --detach-sign --armor --output firma.asc utile.exe

You need a passphrase to unlock the secret key for
user: "Pinco (uno qualsiasi) <pinco@mail>"
1024-bit DSA key, ID E4F4B420, created 2007-03-21

Inserisci la passphrase:
```

dove :

- `-- detach-sign` indica la creazione di una firma separata
- `--armor` indica che il formato sarà *ASCII-armored*, ossia in testo stampabile protetto da due righe, una all'inizio ed una alla fine
- `--output` è seguito dal nome del file dove deve mettere la firma. Se non c'è questa opzione la firma viene stampata a video

- `utile.exe` è il file da inviare

Al termine dell'esecuzione, molto rapida, avremo un file in più il cui contenuto sarà simile al seguente:

```
-----BEGIN PGP SIGNATURE-----
Version: GnuPG v1.4.7 (GNU/Linux)

iD8DBQBGEQGEc3KmH+T0tCARAu5NAJ9Q6qtsqwXIVXPZVUebOJLDNByxwwCfWN1H
9oxL5/L/Mpw5ENUyJizwoWY=
=rtjQ
-----END PGP SIGNATURE-----
```

Per verificare la firma sul file, basta salvare gli allegati al messaggio ed usare questo comando:

```
[pinco@pclinux ~]$ gpg --verify utile.exe.asc utile.exe
gpg: Signature made lun 02 apr 2007 14:10:56 CEST using DSA key ID 3D739F0D
        gpg: Good signature from "Caio <caio@server>"
```

3.11 - UTILITÀ DELLA FIRMA DIGITALE

La firma digitale è uno strumento molto efficace ma non va utilizzato a sproposito: esistono casi in cui non c'è un reale bisogno di firmare il messaggio.

Vediamo quindi i casi in cui la firma digitale può essere estremamente utile:

- quando il messaggio o il documento ha un impatto economico di qualsiasi genere (ad esempio fatture, documenti fiscali, ricevute di pagamento, ecc.)
- quando si intende provare sia la provenienza che l'integrità di un qualsiasi file o documento (ad esempio quando si scaricano software da siti web)
- per diminuire l'incidenza dei virus: se ci giunge un allegato da qualcuno che conosciamo, spesso ci troviamo nell'imbarazzo di capire se è un messaggio vero o siamo davanti ad un virus che cerca di ingannarci per far aprire l'allegato; se i messaggi fossero firmati, allegato compreso, la verifica è immediata

3.12 - CRITTOGRAFIA SIMMETRICA

Se non si intende percorrere tutta la procedura di generazione e convalida delle chiavi si può usare la crittografia simmetrica messa a disposizione da GnuPG, usando cioè una singola chiave per cifrare e decifrare, anche se c'è lo svantaggio dell'estrema importanza che assume la chiave, che deve quindi essere comunicata tramite canali sicuri.

Per cifrare un messaggio o un file usando la crittografia simmetrica, supponendo che il file si chiami documento.txt, basta impartire il comando:

```
[pinco@pclinux ~]$ gpg --symmetric documento.txt
Inserisci la passphrase:
```

In questo caso la *passphrase* è la chiave di cifratura: viene chiesta due volte per sicurezza ed alla fine della cifratura viene generato un file con lo stesso nome, con aggiunta in fondo l'estensione .gpg (nel nostro caso documento.txt.gpg).

Il file è in binario e quindi contiene caratteri non stampabili a schermo; se si vuole l'output in ASCII stampabile, ad esempio per includerlo in una mail senza allegarlo, si può aggiungere l'opzione `--armor` come di consueto.

Il principio fondamentale per la riservatezza e sicurezza dei dati è il fatto che ogni messaggio cifrato è diverso: se proviamo infatti a cifrare due volte lo stesso messaggio con la stessa password otterremo due risultati totalmente differenti, in quanto il processo di cifratura coinvolge anche l'uso di numeri casuali, in modo da dare ogni volta messaggi differenti in presenza dello stesso testo e della stessa password.

Per decifrare il messaggio, il processo è ancora più semplice, dato che GnuPG individua automaticamente il formato del messaggio, che sia *ASCII armored* o binario:

```
[pinco@pclinux ~]$ gpg --decrypt documento.txt.gpg
gpg: dati cifrati con CAST5
Inserisci la passphrase:
```

Inserendo la password si ottiene il messaggio originale stampato a video.
L'algoritmo di cifratura si può cambiare, scegliendolo fra quelli conosciuti da GnuPG.

3.13 - CRITTOGRAFIA A CHIAVE PUBBLICA

Se si vuole comunicare con qualcuno ed essere certi che solo lui possa leggere il messaggio, senza comunicare password e senza incontrarsi prima, si utilizzano nuovamente le coppie di chiavi pubbliche/private.

Diversamente dalla firma digitale, dove si usa la chiave privata per firmare e chi vuole controllare l'autenticità della firma stessa deve usare la chiave pubblica, quando si vuole comunicare con qualcuno si usa la sua chiave pubblica per cifrare, e alla ricezione questi userà la sua chiave privata per decifrare e leggere il messaggio, e solo chi possiede quella chiave può farlo.

Si supponga di voler inviare questo messaggio, memorizzato in un file dal nome carta.txt.:

```
Caro Caio,
come da accordi ti invio i dati della mia carta di credito.
Carta Gold Special
numero ABC 123
scadenza 12/2010
```

```
A presto
Pinco
```

Il comando usato per crittare è:

```
[pinco@pclinux ~]$ gpg --armor --sign --encrypt carta.txt

You need a passphrase to unlock the secret key for
user: "Pinco (uno qualsiasi) <pinco@mail>"
1024-bit DSA key, ID E4F4B420, created 2007-03-21

Inserisci la passphrase: Qui viene digitata la password

Non hai specificato un user ID. (puoi usare "-r")

Current recipients:

Inserisci l'user ID. Termina con una riga vuota:
```

Non avendo indicazioni, GnuPG ha bisogno di sapere quale sia il destinatario del messaggio, per poterlo cifrare con la chiave pubblica giusta, cioè quella di chi riceve il messaggio.

Per identificare il destinatario si può usare il nome, l'indirizzo di posta elettronica o proprio l'ID in esadecimale; è possibile specificare più nomi: in questo caso il messaggio conterrà una copia cifrata del messaggio per ogni destinazione, in un unico blocco, ed ogni destinatario la decifrerà con la sua chiave privata.

Per semplificare le cose, si può anche specificare direttamente il destinatario nel comando, usando l'opzione `--recipient` seguita dal nome, l'ID o l'indirizzo di posta:

```
Inserisci l'user ID. Termina con una riga vuota: Caio

Current recipients:
2048g/32F7C2EE 2007-03-21 "Caio <caio@server>"

Inserisci l'user ID. Termina con una riga vuota:
```

Premendo “Invio” si ottiene il file carta.txt.asc, il cui contenuto è:

```
-----BEGIN PGP MESSAGE-----
Version: GnuPG v1.4.7 (GNU/Linux)

hQIOA/+Dnjwy98LuEAf/dnO0CLem/wPK5U77xV3ow3+EHXMcTQco+KzNBuc3fHTt
ClzZpJVDd02pfo/2B9KqaioRDl8TLMfiLg15rJW1hYWT/d0qXvYHNt6EjhDWfP52
n2YY33EbmpEkkB4/XDNQ1C9uOFtKD9W9iYZOK3ARmoC6/E/BGGpxagwFcUAai+sD
qNBVMP+o1gsTcq2/qJ9NUJdlQdnJE+pJuXUGNuw41H3Rr51Nl9z5uizMKTArqLfT
gI7drwwD30KXr8Np7JkDL0Ho8KM00HqHlCSatIZ2sEXLQDnWojnYsfmfG8zz2QS4
JkSdcNF+45rrM3FxJQ3l8TZsYV8ipRHxEdWOmBWQTwgAoB+RXktoR0x/+smTcfri
FFW37gz2HSczBhDtNa4JeBxSuDud1kjjTVrCBI9U1Ew6JQEG88xvECgLCEu1py53
VqGFWABY1vOeoKkCaVmkuuq9/CRvk6D2ygVxm1CTVAt7zh+e1tdu8Tvfq4Vs2Kpz
reVdbtXLRnB0vECsudn1UjDgmbSsUaZD0fGV4sOnnxkirdVCQHW2X6iodQmUXMCu
cF3vYuGryFIWy39nkXIFkQCk938dah3qgNkBeWZuBK3HLILDkaq5ohCHIOVpp0R1
PhT90wr5I0S3NohqYg8rvsSzB26xgTXXf7SXvXwSZnn/juLQq68VQOF8SMB7boVa
g9LATwG/0/42J85fm9To2tEV4zZlvx/WMip4GxYT5nljWstKFdcEMWwP2kbDoaBI
VqIoeTx/5vFgzvQpFZGnpnW7Umtt7GDfvmVtFC+bHleTUEWuvjWjxGy79Tkerlzf
```

```
oSJDCvZCrA+vOhHnsC/ueCdFXBRLxS4VvQvX4EQ5rBlCXdnR7IFv6jcDtbcP6gCP
6pgFL06llmL6wbsx4hus7rBdqqJE5C+OHHpK2EwYP4R1UFoRHDIK3W5ZIf7oZkgp
rH/NNT80rCVqEWx0MWS7MwyzLZRBosaqanwf1/ct6K77KaumRo2sROxkysDDXnDs
fxLhoyznGX7LTmO+36tNcqRWZ9S3rpsbIrXTgzWl9S4BWlY=
=qc6f
-----END PGP MESSAGE-----
```

Per leggere questo messaggio lo si salva in un file, ad esempio risposta-caio.txt, e per verificare la firma e decifrare il messaggio si digita il comando:

```
[pinco@pclinux ~]$ gpg --decrypt risposta-caio.txt

You need a passphrase to unlock the secret key for
user: "Pinco (uno qualsiasi) <pinco@mail>"
2048-bit ELG-E key, ID 5B514DF0, created 2007-03-21 (main key ID E4F4B420)

Inserisci la passphrase:
```

Si digita la password ottenendo:

```
gpg: encrypted with 2048-bit ELG-E key, ID 5B514DF0, created 2007-03-21
      "Pinco (uno qualsiasi) <pinco@mail>"
Caro Pinco,
ho ricevuto i dati della tua carta, ti includo i riferimenti della
transazione.
Grazie!

n. progressivo 10212011301
del 5/4/2007
Importo: 1500 euro
gpg: Signature made gio 05 apr 2007 14:28:53 CEST using DSA key ID 3D739F0D
gpg: Good signature from "Caio <caio@server>"
```

Per ragioni di sicurezza il messaggio non viene memorizzato, per cui, se lo si vuole salvato su un file, si usa la solita opzione `--output` seguita dal nome del file di destinazione, in questo modo:

```
[pinco@pclinux ~]$ gpg --output messaggio-in-chiaro.txt --decrypt risposta-caio.txt
```

Il fatto che il messaggio in chiaro sia disponibile solo per l'immediata lettura, a meno di diverse indicazioni, è dovuto al ragionamento che se un messaggio è tanto importante da richiedere una cifratura “forte”, il semplice scriverlo in chiaro su un supporto per definizione insicuro come un normale file è un controsenso.