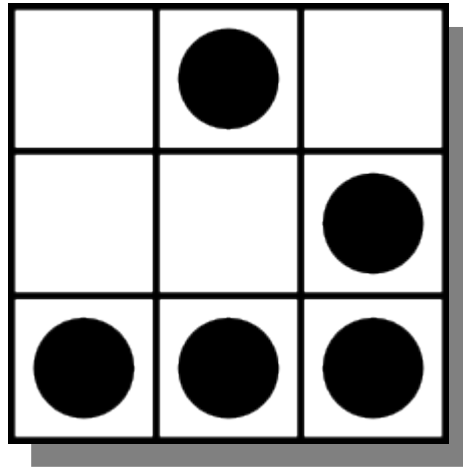


Key Signing Party HOW-TO



Come, cosa e perchè di un Key Signing Party

a cura di Giacomo Rizzo



Che cos'è un Key Signing Party?

- E' divertente
 - C'è tanta gente, ci si diverte, si mangia, si beve, si ride e si scherza. Un momento per evadere della ripetitività della vita di tutti i giorni.
- E' istruttivo
 - Vista la solitamente breve durata di un Key Signing Party, questo viene associato ad altri eventi, spesso e volentieri riguardanti la sicurezza informatica (per rimanere in tema). E' quindi una buona occasione per imparare qualcosa di nuovo.
- E' utile
 - Consente di ampliare/creare la propria “rete della fiducia”, facendosi firmare la propria “chiave pubblica” da altre persone, di cui potremo fidarci.

Rete della fiducia?

- Ehi ehi ehi... “rete della fiducia”, “firmare”, “chiave pubblica”... cos'è tutta questa roba?
 - Un Key Signing Party non è una cosa complicata, ma ci vogliono alcune conoscenze sul principio di funzionamento della crittografia a chiave pubblica, sul meccanismo della “rete della fiducia” ed un pò di pratica.
 - Lo scopo di questa presentazione è proprio quello di fornire queste nozioni “base”, in modo da consentire a tutti di poter partecipare ad un “Key Signing Party” senza alcun problema.
 - Pronti per cominciare? 3... 2... 1... SI PARTE!

Crittografia

- Cominciamo con chiederci cosa sia la “Crittografia”
 - Termine che deriva dal greco “kryptós” (nascosto) e “gráphein” (scrivere)
 - Significa banalmente rendere un messaggio illeggibile per tutte le persone non autorizzate a farlo (vedremo come).
 - E' vecchia quanto l'uomo: la usavano già gli Ebrei (codice atbash) e Giulio Cesare (cifrario di Cesare)
 - Non scenderemo nei dettagli del funzionamento della crittografia (esula dallo scopo di questa presentazione), ma è importante conoscere un minimo di storia della crittografia.

Crittografia: le origini

- Inizialmente venivano usate principalmente “cifrari a sostituzione monoalfabetica”
 - Ad ogni lettera dell'alfabeto ne veniva sostituita un'altra
 - E' il caso del cifrario ebraico “atbash” o del “Cifrario di Cesare”, o ancora dell'attuale “ROT13”
 - Esempio:
 - “Offuschiamo” con ROT13 la parola “CIAO”:
 - C lettera 3 + 13 = lettera 16 = **P**
 - I: lettera 9 + 13 = lettera 24 = **V**
 - A: lettera 1 + 13 = lettera 14 = **N**
 - O: lettera 15 + 13 = lettera 28 = **B**
 - E' oltretutto invertibile: $\text{rot13}(\text{rot13}(\text{TESTO})) = \text{TESTO}$
 - <http://rot13page.googlepages.com/>

Crittografia: lo sviluppo...

- I cifrari a “scorrimento” (o “sostituzione monoalfabetica”) sono debolissimi. Una volta noto l'algoritmo, è facile ricavare il testo iniziale (crittanalisi).
- Si passa ai cifrari “a sostituzione polialfabetica”:
 - Testo in chiaro: A B C D E F G H I J K L M N O P Q R S T U V W X Y Z
 - Alfabeto Z: Z Y X W V U T S R Q P O N M L K J I H G F E D C B A
 - Alfabeto N: N M L K J I H G F E D C B A Z Y X W V U T S R Q P O
- Per decifrare il testo, quindi, conoscendo la “chiave” (zn)
 - Chiave: Z N Z N Z N Z N Z N Z N Z N Z N Z
 - Cifrato: O F M L L A G W L J Z C O J H J G U V
 - Testo:

Crittografia: lo sviluppo...

- I cifrari a “scorrimento” (o “sostituzione monoalfabetica”) sono debolissimi. Una volta noto l'algoritmo, è facile ricavare il testo iniziale (crittanalisi).
- Si passa ai cifrari “a sostituzione polialfabetica”:
 - Testo in chiaro: A B C D E F G H I J K L M N O P Q R S T U V W X Y Z
 - Alfabeto Z: Z Y X W V U T S R Q P O N M L K J I H G F E D C B A
 - Alfabeto N: N M L K J I H G F E D C B A Z Y X W V U T S R Q P O
- Per decifrare il testo, quindi, conoscendo la “chiave” (zn)
 - Chiave: z N Z N Z N Z N Z N Z N Z N Z N Z
 - Cifrato: o F M L L A G W L J Z C O J H J G U V
 - Testo:

Crittografia: lo sviluppo...

- I cifrari a “scorrimento” (o “sostituzione monoalfabetica”) sono debolissimi. Una volta noto l'algoritmo, è facile ricavare il testo iniziale (crittanalisi).
- Si passa ai cifrari “a sostituzione polialfabetica”:
 - Testo in chiaro: A B C D E F G H I J K L M N O P Q R S T U V W X Y Z
 - Alfabeto Z: Z Y X W V U T S R Q P **O** N M L K J I H G F E D C B A
 - Alfabeto N: N M L K J I H G F E D C B A Z Y X W V U T S R Q P O
- Per decifrare il testo, quindi, conoscendo la “chiave” (zn)
 - Chiave: **z** N Z N Z N Z N Z N Z N Z N Z N Z
 - Cifrato: **O** F M L L A G W L J Z C O J H J G U V
 - Testo: L

Crittografia: lo sviluppo...

- I cifrari a “scorrimento” (o “sostituzione monoalfabetica”) sono debolissimi. Una volta noto l'algoritmo, è facile ricavare il testo iniziale (crittanalisi).
- Si passa ai cifrari “a sostituzione polialfabetica”:
 - Testo in chiaro: A B C D E F G H **I** J K L M N O P Q R S T U V W X Y Z
 - Alfabeto Z: Z Y X W V U T S R Q P O N M L K J I H G F E D C B A
 - Alfabeto N: N M L K J I H G **F** E D C B A Z Y X W V U T S R Q P O
- Per decifrare il testo, quindi, conoscendo la “chiave” (zn)
 - Chiave: Z **N** Z N Z N Z N Z N Z N Z N Z N Z
 - Cifrato: O **F** M L L A G W L J Z C O J H J G U V
 - Testo: L **I**

Crittografia: lo sviluppo...

- I cifrari a “scorrimento” (o “sostituzione monoalfabetica”) sono debolissimi. Una volta noto l'algoritmo, è facile ricavare il testo iniziale (crittanalisi).
- Si passa ai cifrari “a sostituzione polialfabetica”:
 - Testo in chiaro: A B C D E F G H I J K L M N O P Q R S T U V W X Y Z
 - Alfabeto Z: Z Y X W V U T S R Q P O N M L K J I H G F E D C B A
 - Alfabeto N: N M L K J I H G F E D C B A Z Y X W V U T S R Q P O
- Per decifrare il testo, quindi, conoscendo la “chiave” (zn)
 - Chiave: Z N Z N Z N Z N Z N Z N Z N Z N Z
 - Cifrato: O F M L L A G W L J Z C O J H J G U V
 - Testo: L I N C O N T R O E A L L E S E T T E

Crittografia: la chiave

- Con l'introduzione di una “chiave”, abbiamo reso il tutto più difficile.
- Maggiore è il numero di alfabeti utilizzati (la lunghezza della chiave), comunque, maggiore è la difficoltà nel decifralo.
- La crittanalisi basata sulle frequenze però è in grado di decifrare un testo di questo tipo senza grosse difficoltà (primo fu il colonnello prussiano Friedrich Kasiski, 1863)
- Una chiave lunga quanto il testo rende indecifrabile (a patto di non avere a disposizione il testo originale)
- Visto che le due parti condividono la stessa chiave, questa prende il nome di “crittografia simmetrica”

La Legge di Kerckhoffs

"La sicurezza di un crittosistema non deve dipendere dal tener celato il crittoalgoritmo. La sicurezza dipenderà solo dal tener celata la chiave." (legge di Kerckhoffs, 1883)

Kerckhoffs aveva le sue buone ragioni per dire questo, all'alba del 20° secolo:

- Gli algoritmi, a differenza delle chiavi, vengono usati da più individui, e sono quindi più esposti agli attacchi
- Oltre a risultare meno aggiornabili, perchè implementati in soluzioni hardware o software.
- La chiave, essendo unica, è sicuramente più facile da proteggere, oltre che da cambiare
- Inoltre un algoritmo aperto è più controllato, quindi potenzialmente più sicuro di uno “nascosto”

Crittografia: i problemi

- Problema: la chiave segreta deve rimanere tale!
 - Una volta svelata la chiave segreta, tutto è perduto.
 - Ma la chiave deve essere posseduta dal mittente (per cifrare) e dal destinatario (decifrare) se si vuole che la comunicazione sia possibile.
 - Problema quindi nel trasporto della chiave
- Per quasi 100 anni, il problema rimase tale. Solo nei tardi anni '60, grazie al lavoro di Hellis, Cocks e Williamson prima, di Diffie, Hellman poi e Rivest, Shamir e Adleman infine, furono resi possibili i meccanismi di crittografia a chiave pubblica, che risolsero gran parte del problema della distribuzione della chiave.

Crittografia: la svolta!

- In cosa consiste questa svolta?
 - Nell'ideazione della “Chiave Pubblica”.
 - Tutto il meccanismo si basa su un'operazione matematica, semplice da eseguire, ma “difficilmente reversibile” (a partire dal risultato è difficile risalire agli argomenti): ad esempio la fattorizzazione.
 - Nel corso degli ultimi anni, sono stati scoperti diversi metodi per rendere più veloce ed efficiente l'algoritmo di fattorizzazione, ma nessuno è mai riuscito a fare quel “salto di qualità” che consente di mettere in crisi gli algoritmi a chiave pubblica
 - E' comunque buona norma generare chiavi sufficientemente lunghe (almeno 1024, meglio 2048 bit).

La chiave asimmetrica

- Il meccanismo di funzionamento della chiave pubblica è piuttosto semplice. Ogni utente genera una coppia di chiavi (si tratta di una singola operazione, in quanto sono legate tra di loro):
 - Una chiave PUBBLICA
 - Una chiave PRIVATA
- La chiave pubblica deve essere inviata al “keyserver”, la chiave privata resta sul sistema dove è stata generata, e deve essere adeguatamente protetta (altrimenti tutto perduto).
- Le due chiavi sono tra di loro “inverse”: cifrare con una, si decifra con l'altra, e viceversa
- Le due chiavi non sono (ovviamente) uguali.

Cifratura

- Idealmente:
 - ALICE cifra il testo con la chiave pubblica di BOB (la chiave viene reperita su un keyserver dove BOB l'ha inviata)
 - BOB può decifrare il testo usando la propria chiave segreta, che è solo in suo possesso, non viene scambiata con nessuno.
 - EVE non può fare nulla, non possedendo che la chiave pubblica di BOB.
- Nella pratica:
 - BOB deve custodire in maniera adeguata la propria chiave privata.
 - ALICE deve essere sicura che la chiave in suo possesso sia effettivamente quella di BOB, e non un falso (di EVE?)

Firma digitale

- Idealmente:
 - ALICE cifra con la propria chiave privata un hash del testo contenuto nella email, generando così una “firma digitale”.
 - BOB può decifrare il testo usando la chiave pubblica di ALICE, che è solo in possesso di ALICE, verificando che il testo sia conforme a quanto ricevuto (integrità e non rupidio).
 - Se EVE modifica il contenuto del testo durante il tragitto, non potendo rigenerare l'hash tale quale (non ha la chiave privata di ALICE), “romperà” la firma digitale.
- Nella pratica:
 - La chiave di ALICE deve essere solo in suo possesso.
 - BOB deve essere sicuro del fatto che quella chiave sia proprio di ALICE

Problemi

- La prima delle due affermazioni (sicurezza della chiave privata) non può che ricadere sulle spalle di BOB e del suo Sistema Operativo.
 - Solitamente, se il sistema operativo utilizzato è fatto come si deve, bastano pochi accorgimenti per essere ragionevolmente sicuri della sicurezza della propria chiave privata (anche perchè viene ulteriormente protetta tramite una passphrase)
 - Generare inoltre dei “certificati di revola” ed archivarli su un dispositivo removibile consente inoltre di “riparare” nel malaugurato caso in cui la chiave venisse compromessa.
- La seconda delle due affermazioni (autenticità della chiave pubblica) viene facilmente risolta:
 - La chiave pubblica deve essere il piu pubblica possibile

Problemi

- Vi faccio inoltre notare un ulteriore problema, la cui soluzione non è così evidente e che quindi tralascieremo banalmente:
 - Il calcolo della firma digitale, non viene fatta da Alice, ma dal processore del suo computer
 - Pur ammettendo che la chiave si trova solo sul computer di Alice, non possiamo in alcun modo essere sicuri che sia proprio lei ad aver “confermato” l'invio di quel messaggio
 - Possiamo solo dire che è stato certamente inviato da quel computer, da qualcuno che conosceva la passphrase per sbloccare la chiave privata ivi contenuta
 - Questo apre la strada a: virus e malware di vario tipo, e rischia di mettere in discussione il non-ripudio.
 - Tralasciamo per stavolta...

I KeyServer

- Per rendere più pubblica possibile la “chiave pubblica” nascono i KeyServer, infrastrutture che contengono e condividono le chiavi pubbliche degli utenti.
 - Chiunque può inviare qualsiasi chiave, ed i keyserver se la scambiano tra di loro
 - Mettono a disposizione di tutti, tutte le chiavi con diversi protocolli, tra cui LDAP e la posta elettronica
- Questo garantisce che una chiave caricata sul keyserver (vedremo come fare), nel giro di poco tempo venga resa disponibile su tutti i KeyServer, e quindi disponibile a tutti gli utenti (e programmi) nel maggior numero di modi possibili.
- Ma chi garantisce sull'autenticità, se tutti possono inserire chiavi di chiunque? (necessario)

L'autenticità della chiave

- Ci sono 2 strade per accertare l'autenticità di una chiave pubblica:
 - Metodo verticale: la PKI
 - Le Public Key Infrastructure sono delle autorità (solitamente auto-referenziate) che “certificano” (spesso dietro compenso) l'autenticità di una determinata chiave pubblica, chiedendo a chi la vuole certificare una serie di dati, in modo da verificarne l'autenticità.
 - Ci si deve fidare della PKI. Ma nel mondo di oggi, sappiamo bene che far riferimento ad una unità centrale porta a grossi danni nel momento in cui questa fallisce.
 - Soluzione fare riferimento a PKI diverse (soluzione vera?)
 - Metodo orizzontale: il “Web of Trust”

Il “Web of Trust”

- Letteralmente la “rete della fiducia” è il metodo orizzontale di controllo dell'autenticità delle chiavi.
- Si organizzano dei Party, in cui ognuno porta la propria chiave pubblica (quasi) e un documento di identità.
- Gli altri partecipanti al Party controllano la corrispondenza dei dati, certificando di fatto l'associazione tra persona e chiave pubblica.
- Rientrati “ai propri alloggi”, gli utenti possono procedere a firmare la chiave pubblica ricevuta per mezzo della propria, certificando così quanto verificato al Key Signing Party.

Il “Web of Trust”



Chiave di
TIZIO

- TIZIO genera una coppia di chiavi e porta la sua chiave pubblica ad un Key Signing Party.

Il “Web of Trust”



- TIZIO genera una coppia di chiavi e porta la sua chiave pubblica ad un Key Signing Party.
- CAIO certifica l'autenticità della chiave pubblica di TIZIO, e TIZIO quella di CAIO

Il “Web of Trust”



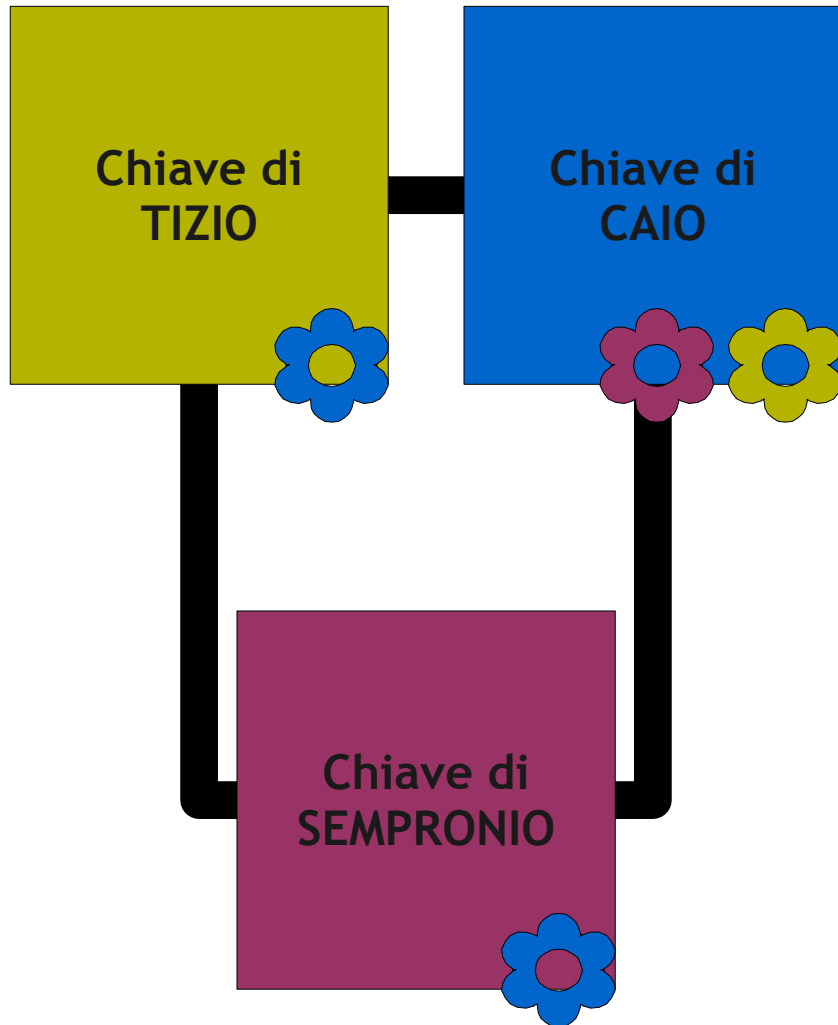
- TIZIO genera una coppia di chiavi e porta la sua chiave pubblica ad un Key Signing Party.
- CAIO certifica l'autenticità della chiave pubblica di TIZIO, e TIZIO quella di CAIO
- Ad un altro Party, SEMPRONIO firma la chiave di CAIO e viceversa.

Il “Web of Trust”



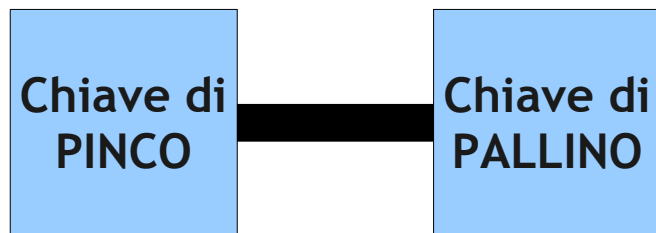
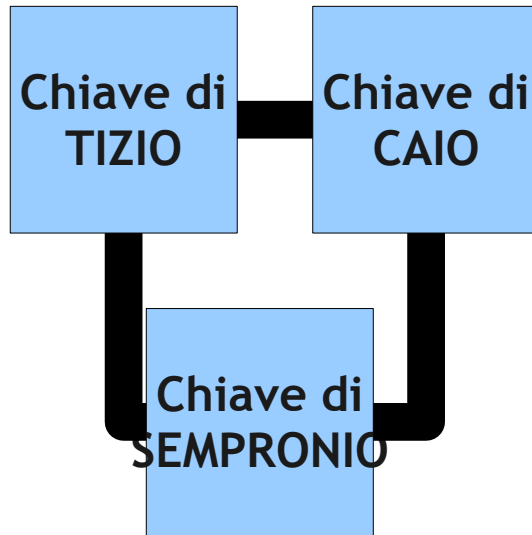
- Nel momento in cui SEMPRONIO vuole valutare l'autenticità della chiave di TIZIO, vedrà che questa è firmata dalla chiave di CAIO, di cui lui si fida, avendola firmata egli stesso.
- Potrà quindi fidarsi della chiave di TIZIO.

Il “Web of Trust”



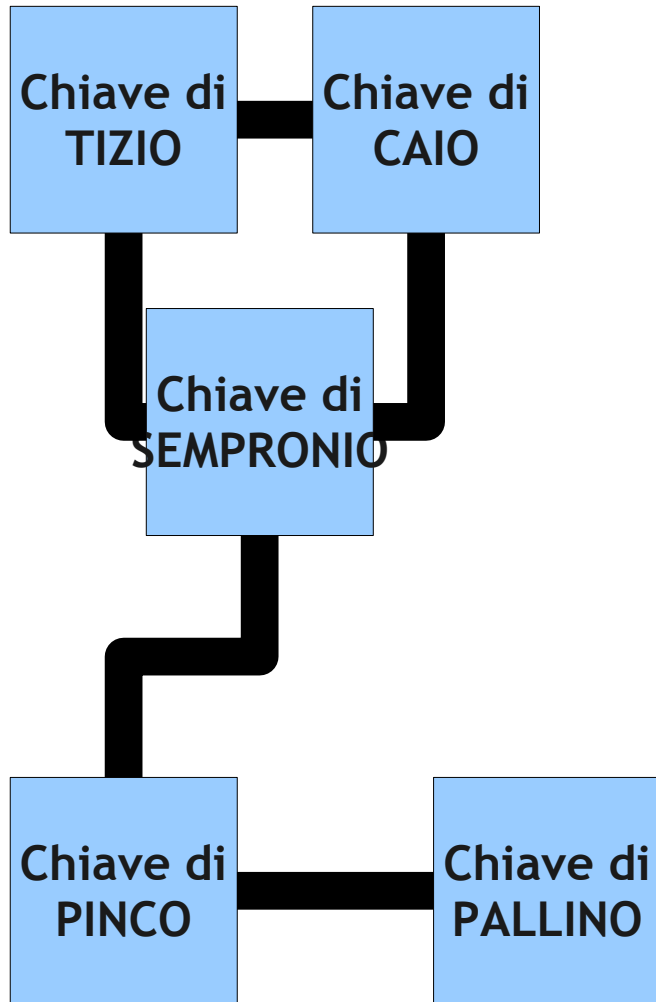
- Nel momento in cui SEMPRONIO vuole valutare l'autenticità della chiave di TIZIO, vedrà che questa è firmata dalla chiave di CAIO, di cui lui si fida, avendola firmata egli stesso.
- Potrà quindi fidarsi della chiave di TIZIO.
- Creando così delle vere e proprie “reti” di fiducia.

Il “Web of Trust”



- Nel momento in cui SEMPRONIO vuole valutare l'autenticità della chiave di TIZIO, vedrà che questa è firmata dalla chiave di CAIO, di cui lui si fida, avendola firmata egli stesso.
- Potrà quindi fidarsi della chiave di TIZIO.
- Creando così delle vere e proprie “reti” di fiducia.

Il “Web of Trust”



- Nel momento in cui SEMPRONIO vuole valutare l'autenticità della chiave di TIZIO, vedrà che questa è firmata dalla chiave di CAIO, di cui lui si fida, avendola firmata egli stesso.
- Potrà quindi fidarsi della chiave di TIZIO.
- Creando così delle vere e proprie “reti” di fiducia.
- Che oltretutto si possono interconnettere.

Piu facile a farsi che a dirsi

Nella pratica, le cose sono molto piu semplici. Generiamo una coppia di chiavi (da riga di comando):

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

Che lancia un piccolo wizard in cui sceglierete:

- Tipo della chiave (DSA + Elgamal)
- Dimensione della chiave (2048 bit)
- Validità e/o data di scadenza (0)
- Vostro nome e cognome, un eventuale commento (nick) e l'indirizzo di posta di riferimento per la chiave
- La pass phrase

Questo genera una coppia di chiavi, una pubblica, ed una privata, che risiedono nella directory `~/.gnupg/` e ne viene visualizzata la “fingerprint”.

Fingerprint?

- La fingerprint è un'impronta univoca ricavata dalla chiave pubblica. Consente di identificare la chiave più comodamente. E' una specie di “resoconto” della chiave.

```
pub 1024D/C6727BD0 2006-11-01
```

```
Key fingerprint = 3287 C83E E4DA 6F9A 966F 67D9 A717 F93B C672 7BD0
```

```
uid Giacomo Rizzo (alt-os) <alt-os@openlabs.it>
```

```
sub 2048g/88FBC125 2006-11-01
```

- Ora si invia la chiave pubblica al KeyServer:

```
gpg --send-keys C6727BD0 --keyserver pgp.mit.edu
```

Stampare la fingerprint

- Il passo successivo è quello di stampare la fingerprint della propria chiave da portare al Key Signing Party (e sempre appresso, che è una buona idea :P)

```
gpg --fingerprint C6727BD0
```

- Si copia il testo in output, e lo si stampa, in più copie, su carta.
- Alcuni Key Signing Party richiedono di inviare copia della propria chiave pubblica ad un utente/server indicato. Nell'eventualità, per esportare la propria chiave pubblica in un formato “utile”, si dà il comando

```
gpg --armor --export C6727BD0 > C6727BD0.asc
```

- Questo genera un file .asc contenente la chiave pubblica nella sua interezza, in modo che possiate spedirla.

Durante e dopo il Party

- Al Key Signing Party, vi scambierete le stampe delle fingerprint, e verificherete che il nome e cognome che compaiono su quelle degli altri corrispondono alle rispettive carte di identità.
- Una volta tornati a casa, recupererete dal server la chiave della persona in questione (il cui numero sta sulla stampa della fingerprint):

```
gpg --recv-keys 5001277D
```

- Verificate che la relativa fingerprint sia quella indicata sul foglio stampato (verifichiamo che il server ci abbia mandato quella giusta)

```
gpg --fingerprint 5001277D
```

Verifica della fingerprint

- La verifica della fingerprint è la parte probabilmente più delicata di tutto il processo, perchè la pigrizia può giocare davvero brutti scherzi.
- E' importantissimo verificare **INTERAMENTE** la fingerprint che vi viene consegnata, perchè una sua manomissione rischia di mettere in crisi tutto il vostro Web Of Trust
- Solitamente l'essere umano per verificare una sequenza di caratteri ne legge la testa e la coda, raramente la parte centrale.

3287 C83E E4DA 6F9A 966F 67D9 A717 F93B C672 7BD0

Verifica della fingerprint

Questo è il motivo per
cui riuscite a leggere
questo testo

Verifica della fingerprint

- La fingerprint della slide precedente, quindi, potrebbe essere agevolmente scambiata con una simile

3287 C83E E4DA 6F9A 966F 67D9 A717 F93B C672 7BD0

3287 C83E FT96 8Z9A 966F 67D9 H12Z P34T C672 7BD0

da qualsiasi malintenzionato che abbia compromesso il KeyServer.

- E' quindi importante che vi accertiate che la chiave che avete ricevuto sia **ESATTAMENTE** quella di cui avete verificato l'autenticità durante il Key Signing Party.

Durante e dopo il Party

- Una volta verificato che vi trovate di fronte alla chiave corretta, potete procedere alla firma:

```
gpg --sign-key 5001277D
```

- Si tratta di un banale wizard testuale, che vi chiede di confermare di aver verificato la validità della chiave, dopodichè appone la firma con la vostra chiave privata (vi chiede quindi la relativa passphrase) e memorizza la chiave così firmata. A questo punto, ci sono due modi di procedere:

- Inviare la copia della chiave all'utente a cui l'avete firmata

```
gpg --export --armor 5001277D > 5001277D.asc
```

- Oppure spedire la chiave firmata al KeyServer, perchè la distribuisca (pare sia considerata cattiva abitudine...)

```
gpg --send-keys 5001277D --keyserver pgp.mit.edu
```

Il gioco è finito

- Firmate tutte le chiavi che avete verificato, avrete terminato il vostro dovere. Nel caso qualcuno vi invii la vostra chiave firmata da lui, sarà sufficiente importarla nel vostro “keyring” e caricarla voi sul server:

```
gpg --import C6727BD0
```

```
gpg --send-keys C6727BD0 --keyserver pgp.mit.edu
```

- Ogni tanto (ed in particolare dopo i KSP) è buona abitudine fare un “refresh” delle chiavi, richiedendole tutte nuovamente al server: questo consente non solo di aggiornare le firme che tenete in locale per le chiavi pubbliche degli altri, allargando il Web of Trust, ma anche di verificare l'aggiunta di eventuali nuovi UID

```
gpg --refresh-keys --keyserver pgp.mit.edu
```

Fine della presentazione

Fine.

alt@free-os.it