



# GNU/Linux e la firma elettronica

*Paolo Garbin – [paolo@paolettopn.it](mailto:paolo@paolettopn.it)*

*02503CBF - 0CFF 04DF 17CD 8222 6E3D 8DFC 28A7 43D7 0250 3CBF*

51 anni

Tecnico Informatico, nella Pubblica Amministrazione

Dal 1990 si occupa di Software Libero, collaborando inizialmente con la Comunità di Ubuntu-it, successivamente in altre Associazioni di cultura informatica del software libero (ILS, PN LUG, Libreltalia, LibreOffice, ecc.).

Si interessa di divulgazione e sensibilizzazione sull'uso del software libero e dei formati liberi, sulla sicurezza informatica e sulla difesa della *privacy* in rete e





Queste diapositive sono state realizzate nel 2007 da Fabrizio Tarizzo e soggette alla licenza *Creative Commons*, nella versione *Attribuzione - Condividi allo stesso modo 2.5 Italia*; possono pertanto essere distribuite liberamente ed altrettanto liberamente modificate, a patto che se ne citi l'autore e la provenienza e che le versioni modificate siano distribuite sotto gli stessi termini di licenza.

*<http://creativecommons.org/licenses/by-sa/2.5/it/>*

Il relatore odierno si esprime a titolo personale e nell'ambito di proprie ricerche, aggiornamenti e sperimentazioni. Il contenuto tecnico della presentazione e le opinioni espresse nella stessa non rappresentano parere professionale e non sono, necessariamente, riconducibili alle affiliazioni, professionali e non, del relatore.

# La teoria

- *Il canale sicuro;*
- *La crittografia simmetrica e asimmetrica;*
- *La firma elettronica;*
- *Le funzioni hash;*
- *Le reti di fiducia*

# Canale di comunicazione “sicuro”



Si definisce ‘*canale sicuro*’ un canale di comunicazione in grado di garantire:

**Confidenzialità**: la garanzia che il messaggio possa essere letto solo dai legittimi destinatari

**Integrità**: la garanzia che il messaggio non sia stato alterato nel tragitto tra mittente e destinatari

**Autenticazione**: la certezza dell'identità del mittente

**Non ripudiabilità**: la garanzia che il mittente non possa in futuro disconoscere il contenuto del messaggio

Nella *crittografia simmetrica*, la forma classica di crittografia, i due corrispondenti devono *condividere* una stessa *chiave segreta*. Il principale algoritmo di crittografia simmetrica in uso oggi è **AES** (*Advanced Encryption Standard*), anche conosciuto come *Rijndael*. Usa chiavi segrete lunghe 128 o anche di 256 bit.

La crittografia simmetrica funziona e, nelle sue implementazioni attuali, si può considerare sicura, ma soffre di alcuni problemi pratici:

Prima di iniziare a comunicare, è necessario che i corrispondenti concordino in modo sicuro la chiave da usare

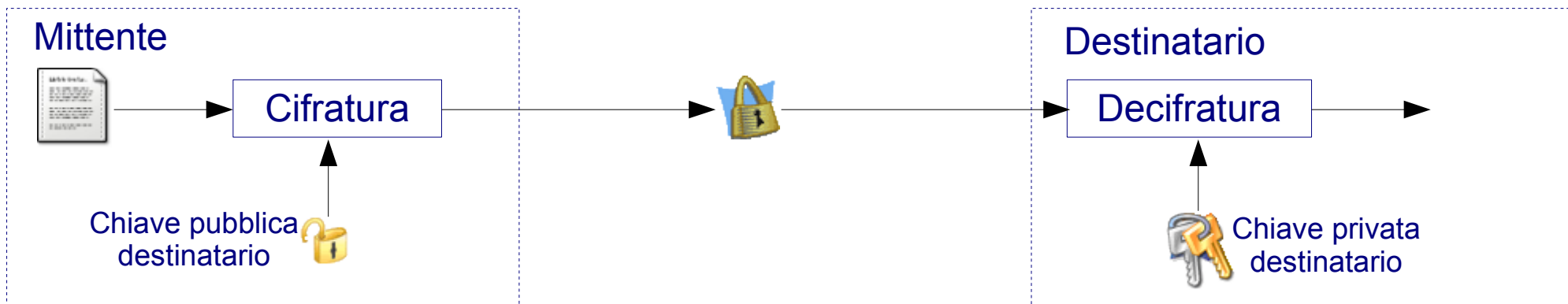
Occorre una chiave diversa per ciascuna coppia di corrispondenti

Per garantire la massima sicurezza, la chiave segreta dovrebbe essere cambiata con frequenza

È conosciuta anche come '*crittografia a doppia chiave*' oppure a '*chiave pubblica*'. Il principio di funzionamento è il seguente.

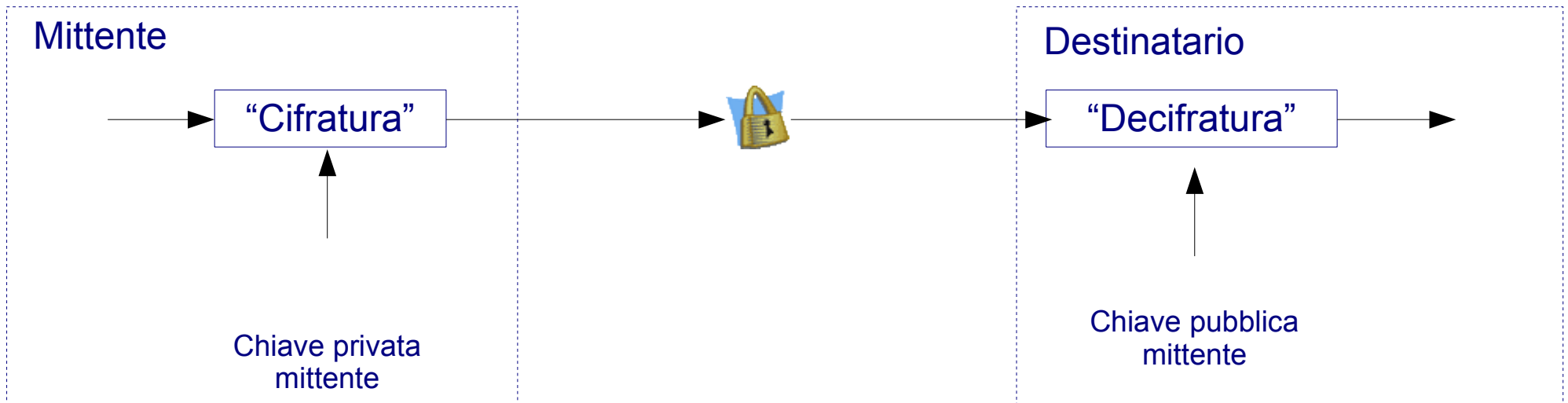
Ciascun utente dispone di *una coppia* di chiavi:

- ◆ una chiave **pubblica** usata per le operazioni di **cifratura**
- ◆ una chiave **privata** usata per le operazioni di **decifratura**



In pratica, chiunque può inviare un messaggio cifrato usando la chiave pubblica del destinatario, ma solo quest'ultimo è in grado di decifrarlo usando la sua chiave privata.

La *firma elettronica* si può ottenere utilizzando ‘al rovescio’ la crittografia asimmetrica.



In questo modo chiunque può leggere il messaggio (in quanto chiunque può conoscere la chiave pubblica) avendo però la certezza che solo il possessore della chiave privata corrispondente può averlo inviato.



Una funzione 'hash' trasforma un testo normale di lunghezza qualsiasi in una stringa di lunghezza fissa. Questa stringa rappresenta una sorta di 'impronta digitale' del messaggio ed è chiamata *checksum* oppure *message digest*.

Le funzioni *hash*, unitamente alla firma elettronica, ci permettono di verificare l'integrità dei messaggi:

Il mittente:

- Calcola il *digest* del messaggio e lo firma usando la propria chiave privata
- Invia il *digest* firmato insieme al messaggio

Il destinatario:

- Verifica la firma usando la chiave pubblica del mittente, ottenendo - se la verifica va a buon fine - il valore del *digest* che era stato calcolato dal mittente
- Calcola il *digest* sul messaggio ricevuto
- Se i due valori coincidono, potrà essere ragionevolmente certo che durante la trasmissione il messaggio sia rimasto inalterato

# Rete della fiducia (1)



Grazie alle funzioni *hash* ed alla crittografia asimmetrica siamo riusciti a garantire l'integrità del messaggio durante il trasporto e a stabilire con certezza l'associazione messaggio-chiave. Il passo successivo è riuscire a stabilire l'associazione chiave-mittente.

Se non conosciamo personalmente il mittente, dobbiamo affidarci ad un terzo soggetto (“*certificatore*”), del quale entrambi ci fidiamo, che garantisca (“*certifichi*”) questa associazione.

La certificazione avviene tramite la firma elettronica: il *certificatore* dispone della propria chiave che usa per firmare delle dichiarazioni (“*certificati elettronici*”) con le quali attesta di aver verificato personalmente l'identità dei possessori delle chiavi. I *certificati* vengono distribuiti insieme alle chiavi pubbliche.

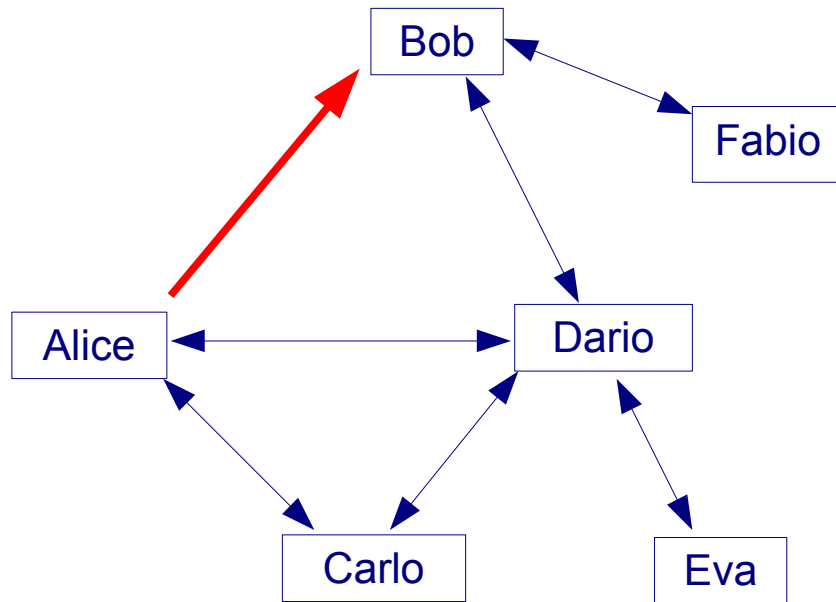
Sono stati ideati due diversi modelli di certificazione:

Rete della fiducia “*paritetica*” (OpenPGP)

Rete della fiducia “*gerarchica*” (PKI)

# Rete della fiducia (2)

Modello della *rete della fiducia* “paritetica”:



Ideato da Phil Zimmermann per il suo software PGP (primi anni '90)

Utilizzato dai sistemi che seguono lo standard OpenPGP

Tutti gli utenti possono anche essere certificatori

Ogni chiave può essere certificata da più persone

Basato sulla fiducia reciproca tra individui

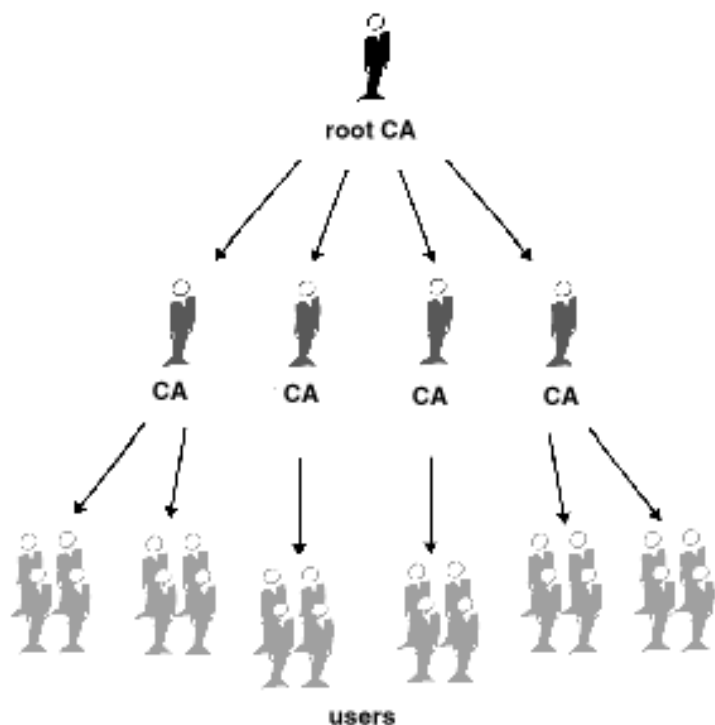
Modello delle *Public Key Infrastructure* (Infrastrutture a chiave pubblica):

Standard ITU X.509

Struttura gerarchica: tutti si fidano (per definizione) del vertice (“*Root Authority*”), che può delegare altri enti (“*Certification Authority*”) che a loro volta certificano le chiavi dei singoli utenti

Usato per la certificazione dei *server sicuri* su Internet (SSL, HTTPS, ...)

Sistema previsto dalle normative europee ed italiane sulla firma elettronica



I concetti visti finora non sono limitati alla sola trasmissione sicura di messaggi o documenti tra persone, ma costituiscono la base di molte tecnologie relative alla sicurezza, ad esempio:

Reti private virtuali (VPN)

Server sicuri (ad esempio: siti di e-commerce, e-banking)

Autenticazione client (SSH)

## Per approfondire

<http://it.wikipedia.org/wiki/Crittografia>

<http://www.ismprofessional.net/pascucci/documenti/gpg/>

C. Giustozzi, A. Monti e E. Zimuel - *Segreti, spie, codici cifrati* - Apogeo

N. Ferguson e B. Schneier - *Crittografia pratica* - Apogeo

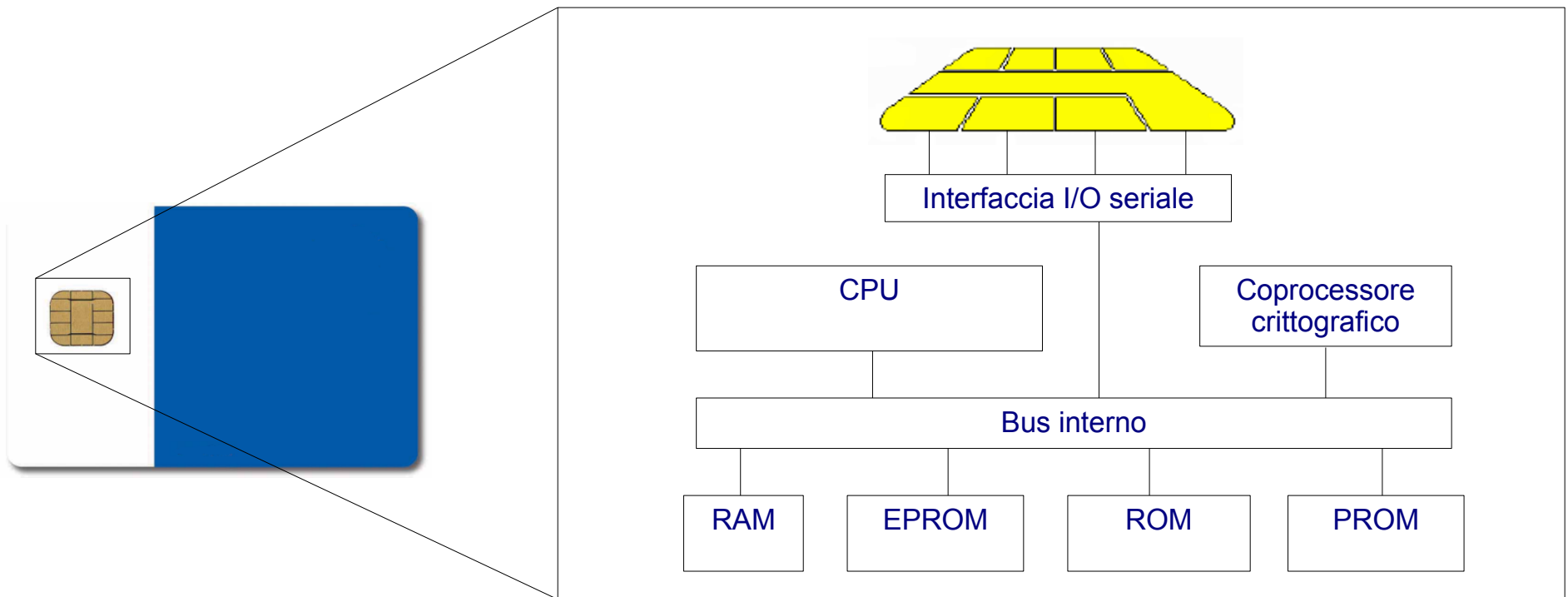
# Smart card

*Principi di funzionamento e utilizzo*

# Smart card (1)

Le *smart card* sono dispositivi a microprocessore con capacità di calcolo e memorizzazione (permanente o temporanea) di dati.

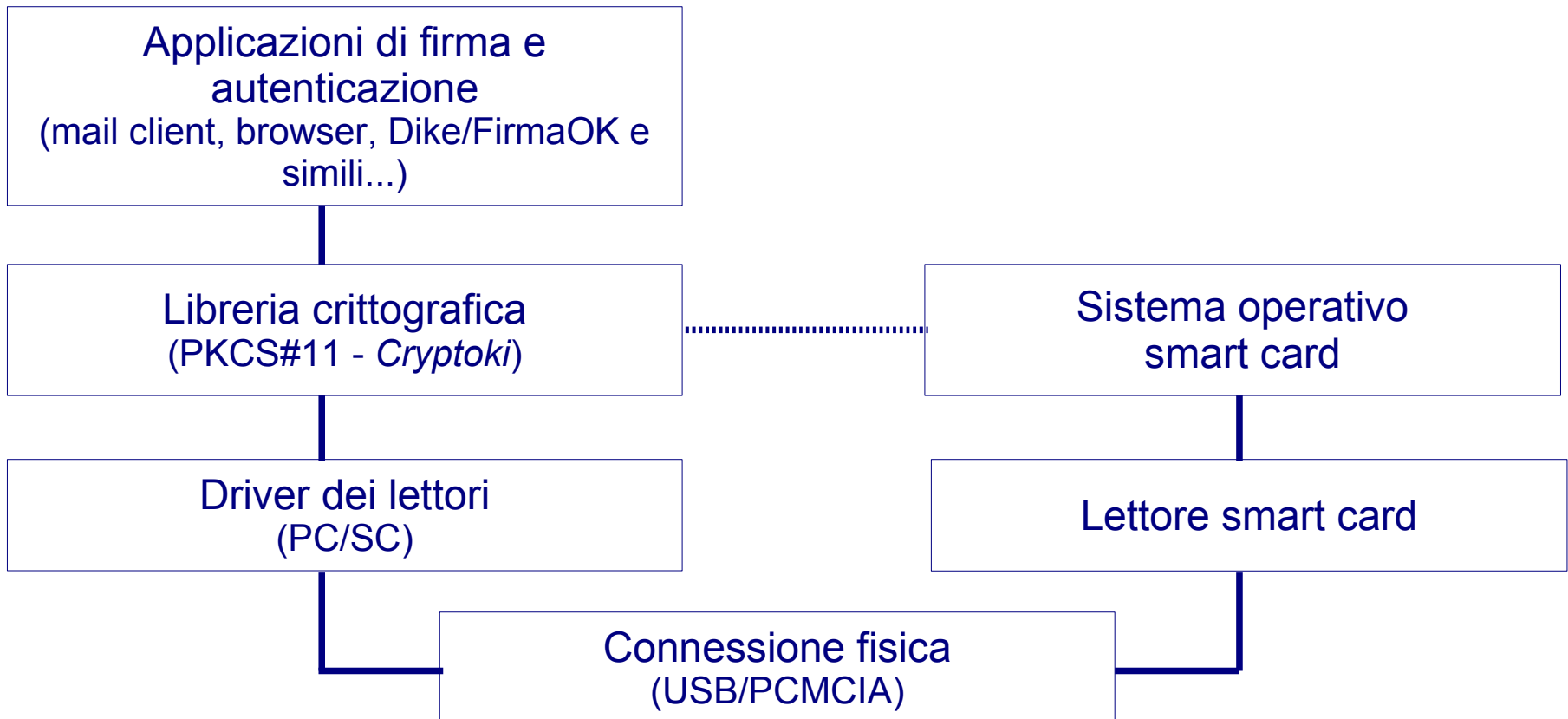
Nell'ambito della crittografia e della firma elettronica sono usate come dispositivi per la memorizzazione delle chiavi private e l'esecuzione sicura delle operazioni che le coinvolgono.



# Smart card (2)

L'architettura software dei sistemi crittografici basati su smart card è basata su *layer* che ricordano il modello dei protocolli di rete.

I principali problemi per gli utenti di software libero derivano dalle librerie crittografiche (*Cryptoki*) legate ai produttori delle smart card.





# In pratica...

*Utilizzo pratico di crittografia e firma elettronica  
in ambiente GNU/Linux*

<http://www.gnupg.org>

## Implementazione di OpenPGP con algoritmi liberi

Cifratura simmetrica: 3DES, AES, AES256, CAST, ...

Cifratura asimmetrica: RSA, ElGamal

Firma: RSA, DSA

*Hash*: MD5, SHA1, RIPEMD160, SHA256, SHA384, SHA512

Creazione e gestione di chiavi asimmetriche

Gestione della rete della fiducia (paritetica)

Cifratura, decifratura, firma e verifica delle firme

Utilizzabile da linea di comando e da interfaccia grafica (Seahorse, KGPG) e *client* di posta (nativamente o tramite *plugin*)

# GNU Privacy Guard - GnuPG (2)



GnuPG memorizza i dati nella directory nascosta `.gnupg` (nella home dell'utente):

`~/.gnupg/secring.gpg` - Chiavi private

`~/.gnupg/pubring.gpg` - Chiavi pubbliche

`~/.gnupg/trustdb.gpg` - Dati sulla rete della fiducia

`~/.gnupg/gpg.conf` - File di configurazione

Le versioni più recenti possono salvare le chiavi private anche su smart card

Ogni chiave è identificata da un ID (8 cifre esadecimali) e da una *fingerprint* (36 cifre esadecimali)

Esempio: 02503CBF (la chiave primaria di `paolo@paolettopn.it`)

`0CFF 04DF 17CD 8222 6E3D 8DFC 28A7 43D7 0250 3CBF`

## GnuPG da linea di comando: gestione delle chiavi

`gpg --gen-key` Generazione chiavi

`gpg --list-secret-keys` Elenca le chiavi private

`gpg --list-keys` Elenca le chiavi pubbliche in nostro possesso

`gpg --export` Salva una chiave pubblica in un file

`gpg --import` Importa una chiave pubblica da un file

`gpg --send-key` Invia una chiave pubblica su un *keyserver*

`gpg --recv-key` Scarica una chiave pubblica da un *keyserver*

`gpg --refresh-key` Aggiorna una chiave pubblica da un *keyserver*

`gpg --gen-revoke` Crea un *certificato di revoca* per la nostra chiave

## GnuPG da linea di comando: gestione della rete della fiducia

`gpg --fingerprint` Mostra la *fingerprint* di una chiave

`gpg --check-sigs` Elenca le firme (certificati) di una chiave

`gpg --edit-keys` Gestione avanzata delle chiavi, mostra tra le altre informazioni il livello di fiducia

`gpg --sign-key` Certifica una chiave altrui

`caff` Programma avanzato per la certificazione di una chiave altrui

<http://pgp-tools.alioth.debian.org/>

*PGP Pathfinder* mostra statistiche e grafici relativi alle chiavi

<http://pgp.cs.uu.nl/>

## GnuPG da linea di comando: cifratura e firma di file

`gpg --encrypt` Cifra un file

`gpg --sign` Firma un file

`gpg --detach-sign` Firma un file (firma in un file separato)

`gpg --clearsign` Firma in chiaro un file

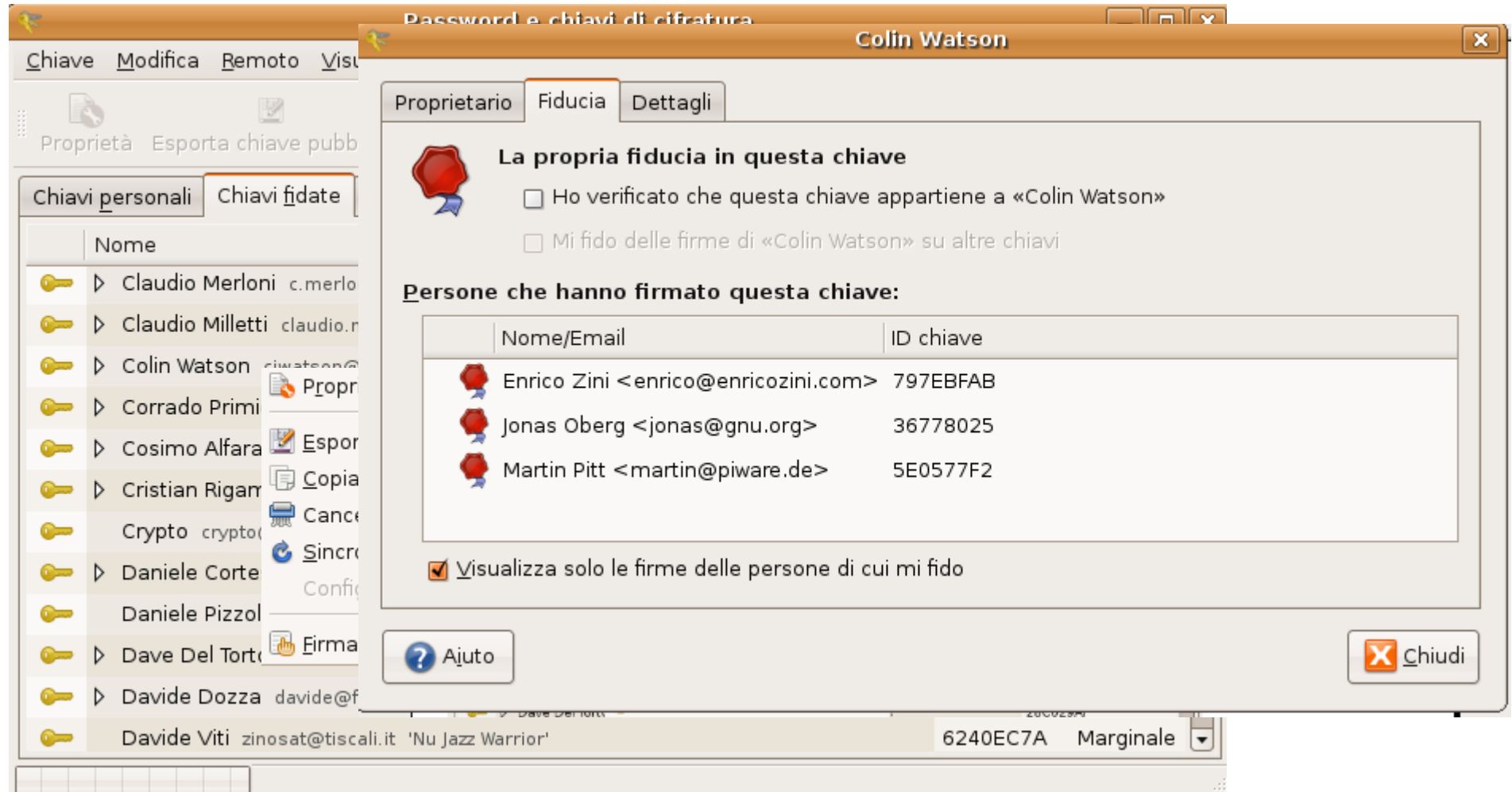
`gpg --sign --encrypt` Firma e cifra un file

`gpg --decrypt` Decifra un file

`gpg --verify` Verifica la firma di un file

`gpg --decrypt --verify` Verifica una firma e decifra un file

## GnuPG da interfaccia grafica: gestione delle chiavi con Seahorse



## GnuPG da interfaccia grafica: firma, verifica, cifratura e decifratura di file con Seahorse

Seahorse aggiunge alcune voci al menu contestuale di Nautilus:

### Cifra

Mostra l'elenco di tutte le chiavi pubbliche possedute, si selezionano i destinatari interessati e si prosegue

### Firma

Chiede la passphrase

### Apri con «Decifra file»

Chiede la passphrase

### Apri con «Verifica la firma»

Mostra il risultato della verifica



## GnuPG da interfaccia grafica: Thunderbird/Enigmail

L'estensione *Enigmail* aggiunge a Thunderbird le funzionalità per utilizzare nel client di posta le chiavi GnuPG:

Una finestra per la gestione delle chiavi, simile a quella di Seahorse

Nuove opzioni nella configurazione degli account di posta

Verifica automatica della firma dei messaggi ricevuti, con indicazione grafica del risultato

Richiesta della *passphrase* e decifratura automatica alla ricezione di messaggi cifrati

Opzioni di firma/cifratura nella finestra di composizione messaggi

# Certificati X.509 gratuiti



<https://www.thawte.com>

Divisione di Verisign

Rilascia certificati gratuiti di firma e cifratura email (solo per uso personale)

È incluso come certificatore fidato in quasi tutti i browser e client di posta



<https://www.cacert.org>

Associazione no-profit australiana

Rilascia certificati gratuiti di firma e cifratura email, certificati server e certificati per la firma del codice

Molti client non lo includono ancora di default tra i certificatori fidati

Per ottenere gratuitamente uno o più certificati occorre registrarsi sul sito e contattare gli *assurer* presenti nella propria zona. Ciascun *assurer*, previa verifica dell'identità, potrà assegnare da 10 a 35 punti. Raggiunti i 50 punti si potranno ottenere certificati nominativi, raggiunti i 100 punti si diventa a propria volta *assurer*.

# Usare certificati X.509 in Thunderbird



Mozilla Thunderbird gestisce nativamente (senza *plugin*) la firma e la cifratura con certificati X.509. Quando si riceve un messaggio firmato, Thunderbird mostra il risultato della verifica con un'icona.

Per installare e utilizzare un proprio certificato:

Creare il certificato con la procedura guidata sul sito (il certificato verrà salvato insieme alla chiave privata nel *certificate store* del browser)

Esportare certificato e chiave privata (in Firefox: *Preferenze > Avanzate > Cifratura > Mostra certificati > Archivia*)

Importare certificato e chiave privata nel *certificate store* di Thunderbird (*Preferenze > Privacy > Sicurezza > Mostra certificati > Importa*)

Configurare il certificato così importato nel proprio account di posta

Usare il pulsante *S/MIME* nella finestra di composizione per firmare o cifrare un nuovo messaggio

LibreOffice permette di firmare documenti in formato ODF utilizzando i certificati X.509 memorizzati nel *certificate store* di Firefox (comando *File - Firme digitali...*).

È necessario indicare ad LibreOffice il percorso del proprio profilo di Firefox utilizzato, digitando da terminale:

```
> export MOZILLA_CERTIFICATE_FOLDER=/home/nomeutente/.mozilla/firefox/nomeprofilo.default
```

```
> export MOZ_CERTIFICATE_FOLDER=/home/nomeutente/.mozilla/firefox/nomeprofilo.default
```

Queste sono due variabili-ambiente in modo che LO sappia dove trovare i certificati.

LibreOffice usa come formato per la firma quello definito dalla raccomandazione IETF/W3C XMLDSIG (RFC 3275).

## Installazione del software di base

### Reperire e installare i driver del lettore

Rilevare marca e modello, da terminale, con `lsusb` oppure `lspcmcia`

Installare il driver corretto (`libccid`, `libasedrive-*`, `libgempc*`, `libtowitoko2`, ...)

### Installare il *middleware* PC/SC

```
# apt install libpcsclite1, pcscd, openct
```

### Installare il *framework* OpenSC

```
# apt install libopensc2, opensc, mozilla-opensc
```

### Verificare la corretta rilevazione del lettore e della smart card

```
# opensc-tool --list-readers
```

```
# opensc-tool --name
```

```
# pkcs11-tool --list-slots
```

## Installazione eventuale *Cryptoki* proprietaria

esempio: smart card InCrypto recenti (CNS InfoCamere/InfoCert)

Scaricare e scompattare “DiKe Linux” dal sito InfoCert

*[http://www.firma.infocert.it/installazione/installazione\\_DiKe.php](http://www.firma.infocert.it/installazione/installazione_DiKe.php)*

Verificare e risolvere eventuali dipendenze mancanti

```
# ldd libincryptoki2.so
```

```
# apt-get install libstdc++2.10-glibc2.2
```

Creare link simbolici in /usr/lib/ per libpcsclite.so.0 e libcrypto.so.0.9.7

Installare la libreria

```
# ./pkcs11installer
```

Verificare la corretta rilevazione della smart card

```
# pkcs11-tool -L --module /usr/lib/libincryptoki2.so
```

## Configurazione Firefox e Thunderbird

In Firefox: *Preferenze > Avanzate > Cifratura > Dispositivi di sicurezza*; in Thunderbird: *Preferenze > Privacy > Sicurezza > Dispositivi di sicurezza*

Fare click su *Carica*, inserire come *Nome modulo* un nome a scelta e come *Nome file modulo* il percorso completo della libreria (`/usr/lib/libinccryptoki2.so`) e confermare

In Firefox, con la CNS inserita, provare a navigare su <http://impresa.gov.it/> (tentando di accedere a *La mia scrivania* dovrebbe essere richiesto il PIN)

In Thunderbird, sempre con la CNS inserita, configurare nel proprio account di posta il certificato di firma della CNS

Scaricare l'elenco dei certificatori dal sito CNIPA e importarlo in Firefox e Thunderbird

## Installazione di JavaSign

Installare la Sun Java Virtual Machine

Scaricare e scompattare javasign

*<http://jvasign.sourceforge.net>*

Scaricare j4sign (sorgente) e copiare *libpkcs11wrapper.so* nella directory di javasign

*<http://j4sign.sourceforge.net>*

Reperire una versione per GNU/Linux della libreria *jaccal-pcsc* e copiarla nella directory di javasign

Modificare il file *jvasign.sh* impostando il percorso della propria installazione di java

Avviare javasign col comando **`./jvasign.sh`**



## Configurazione di JavaSign

Dal menu *Options* > *Authentication* impostare *Smart Card cryptoki* per utilizzare certificati su smart card oppure *P12 file* per utilizzare certificati non qualificati su file

Dal menu *Options* > *Options* :

Se si usa il certificato su smart card, inserire nel tab *SC Cryptoki* il percorso della libreria nel campo *Library* ed il PIN della smart card nel campo *PIN*

Se si usa il certificato non qualificato su file, inserire nel tab *P12 file* il percorso del file e la relativa password

Nel tab *CRL (Certificate Revocation List)* selezionare *Download CRL*

Nel tab *CA (Certification Authority)*, inserire l'indirizzo dell'elenco dei certificatori dal sito CNIPA e fare click su *Download*

## Utilizzo di JavaSign

La finestra principale di JavaSign è sostanzialmente una finestra di browsing del file system. Dopo aver selezionato un file, tramite il menu *File* o il menu contestuale è possibile:

Aprire il file con la voce *Open*

Firmare il file con la voce *Sign* (viene generato un file con l'estensione aggiuntiva *.p7m*)

Se si tratta di un file *.p7m*, verificare la firma: compare una finestra di dialogo con l'esito della verifica, i dati del firmatario e la possibilità di esportare il file originale, il certificato e la chiave pubblica del firmatario e di aggiungere controfirme

Apporre una marcatura temporale (non qualificata) tramite un servizio gratuito (viene generato un file con l'estensione aggiuntiva *.m7m*)

# Links e altri progetti

---



<http://www.gnupg.org/>

<http://www.gnupg.org/gph/it/index.html>

<http://www.openssl.org>

<https://wiki.cacert.org/wiki>

<http://www.opensc-project.org>

<http://opensignature.sourceforge.net>

<http://j4sign.sourceforge.net>

<http://javasign.sourceforge.net>

<http://www.card.infocamere.it/utenti/verifica.php>

---

# Domande?

# Altri metodi utilizzati?

# Grazie per l'attenzione!

*Paolo Garbin*  
*paolo@paolettopn.it*

02503CBF - 0CFF 04DF 17CD 8222 6E3D 8DFC 28A7 43D7 0250 3CBF



**Il mondo di  
Paolettopn**

*<https://www.paolettopn.it>*