PGP / GnuPG e reti di fiducia

Un approccio distribuito alla confidenzialità, sicurezza e fiducia

Enrico Bassetti

28 Ottobre 2017

Latina Linux User Group

Roadmap

1. Introduzione alla crittografia
2. Sistemi a chiave simmetrica e asimmetrica
3. Verificare l'integrità con la crittografia
4. La Fiducia in un sistema asimmetrico
5. PGP/GPG e Web-of-trust
6. Hands-on: Usare GPG

Introduzione alla crittografia

Introduzione

Quando Giulio Cesare comunicava con i suoi generali, come in ogni struttura militare, voleva mantenere i messaggi segreti. Non fidandosi dei messaggeri, iniziò quindi a scrivere i messaggi "spostando" (in inglese *shift*) di 3 ogni lettera del messaggio.

In questo modo, solo chi conosceva il sistema poteva, spostando le lettere di 3 posizioni indietro, leggere il messaggio.

Questo sistema è noto come **cifrario di Cesare**.

Definizioni

- **Chiave** (*key*): speciale stringa segreta, utilizzata per cifrare o decifrare un messaggio (nel caso di *Cesare* "3" era la *chiave*)
- **Testo in chiaro** (*plaintext*): il testo originale del messaggio, leggibile
- **Cifrare** (*encryption*): l'operazione che rende un testo in chiaro illegibile senza conoscere la chiave
- **Testo cifrato**, o *ciphertext*: testo "cifrato" (si presenta spesso come un "blob" di caratteri senza senso)
- **Decifrare** (*decryption*): operazione inversa della cifratura

Cifrario di Cesare: esempio

Testo in chiaro: **ATTACCARE** Chiave: **3**

- $A \rightarrow D$
- $T \rightarrow W$
- $T \rightarrow W$
- $A \rightarrow D$
- $C \rightarrow F$
- $C \rightarrow F$
- $A \rightarrow D$
- $R \rightarrow U$
- $E \rightarrow H$

Testo cifrato: **DWWDFFDUH**

Problemi con il cifrario di Cesare

ATTACCARE

DWWDFFDUH

Vedete qualche problema con il cifrario?

Problemi con il cifrario di Cesare

ATTACCARE

DWWDFFDUH

Vedete qualche problema con il cifrario?

- Le stesse lettere sono codificate allo stesso modo - in altre parole, se una lettera si ripete nel messaggio "in chiaro", allora si ripete anche nel messaggio cifrato.

Problemi con il cifrario di Cesare

ATTACCARE

DWWDFFDUH

Vedete qualche problema con il cifrario?

- Le stesse lettere sono codificate allo stesso modo - in altre parole, se una lettera si ripete nel messaggio "in chiaro", allora si ripete anche nel messaggio cifrato.
- Semplicemente provando tutte le 25 possibili combinazioni (35 se si considerano i numeri), si può risalire al numero di spostamenti necessari!!

Sistemi di crittografia moderni

Esistono sistemi più efficaci per generare "testo cifrato" dal testo non cifrato usando una chiave. I più importanti sono:

- AES
- RSA
- ElGamal

Tutti si basano sulla difficoltà di risolvere problemi matematici, ad esempio il *logaritmo discreto* o la *fattorizzazione in numeri primi*.

La crittografia nella vita quotidiana

Molti sistemi che utilizziamo abitualmente usano un sistema crittografico. Ad esempio:

- La rete GSM/UMTS (chiamate voce e SMS)
- Siti internet HTTPS (la banca, PayPal, Amazon, etc)
- Tutti i sistemi di posta elettronica forniscono l'accesso anche in modo sicuro con SSL
- ...

Sistemi a chiave simmetrica e asimmetrica

Definizioni

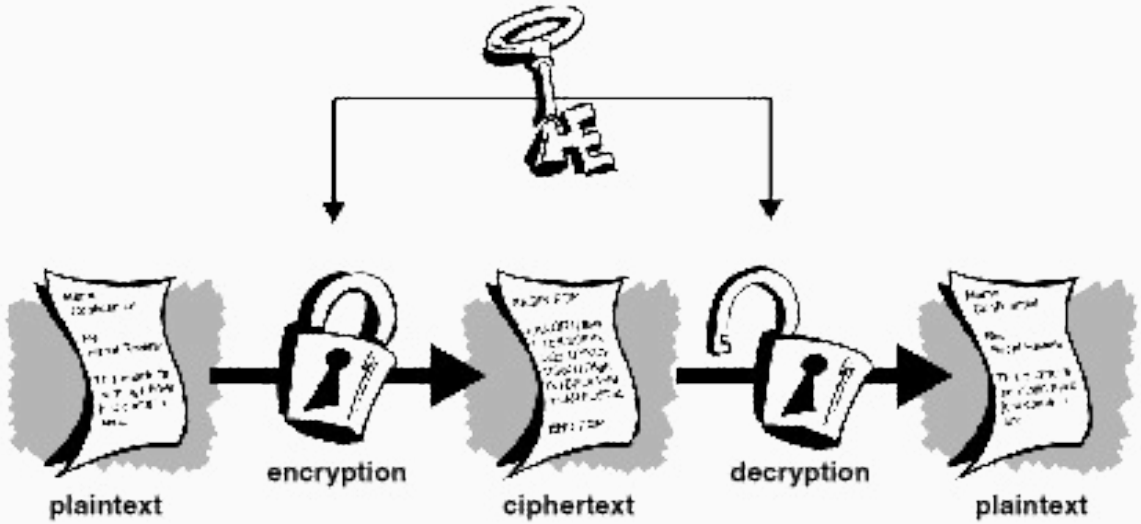
Per sistema a **chiave simmetrica** si intende un sistema di trasmissione che utilizza la stessa chiave per *cifrare* e *decifrare* il messaggio. Il cifrario di Cesare è un sistema a chiave simmetrica.

Un sistema che impiega due chiavi diverse per la crittazione e la decrittazione è detto a **chiave asimmetrica**. In genere si utilizza il termine **chiave pubblica** per la chiave di crittazione, e **chiave privata** per la chiave di decrittazione.

Chiave simmetrica

Un sistema di crittazione con chiave simmetrica si basa sulla presenza di un **segreto condiviso**, ovvero della chiave.

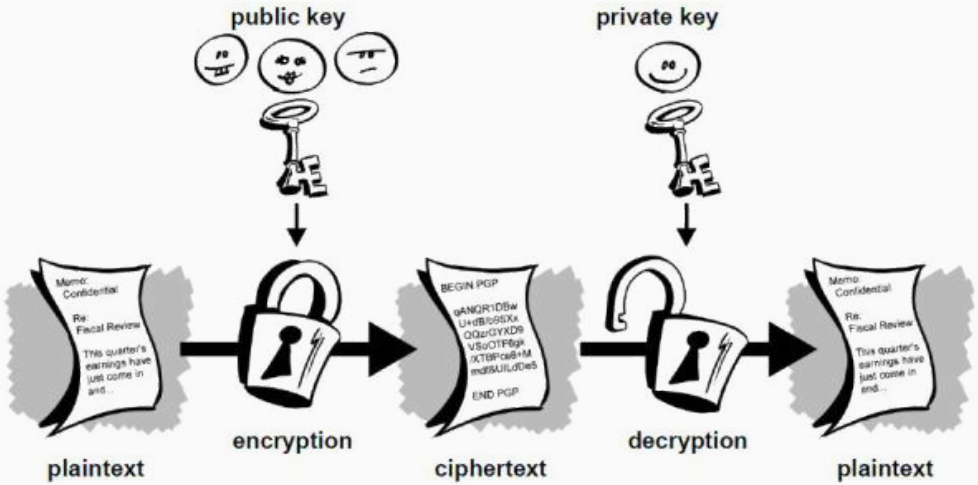
Questa chiave deve essere **già** condivisa tra i due (o più) soggetti della comunicazione in modo sicuro.



Chiave asimmetrica

La chiave **pubblica** è una chiave che può essere utilizzata per crittografare un messaggio, ma non può essere utilizzata per l'operazione inversa. In altre parole, una volta crittato un file con la chiave pubblica, non è possibile leggerlo se non utilizzando la chiave **privata**.

Questa operazione permette l'invio di messaggi tra due soggetti senza che venga condiviso un segreto.



Verificare l'integrità con la crittografia

Integrità del messaggio

In taluni casi è necessario verificare l'integrità del messaggio, ovvero verificare che non sia stato alterato (pensiamo ad un codice IBAN per un versamento).

Esistono algoritmi che generano un codice di verifica (chiamato **hash**) con il quale si può verificare che il testo non sia stato alterato. Ma come possiamo essere sicuri che anche questo codice non venga alterato?

Integrità del messaggio (2)

Usando un sistema crittografico (con chiave simmetrica o asimmetrica) possiamo crittografare il messaggio (o il codice di verifica): in questo modo il messaggio diventa inalterabile a meno di conoscere le chiavi.

Nel caso si usi un sistema a chiave **simmetrica**, è sufficiente crittare l'*hash* generato con la chiave e allegarlo al messaggio: solo il destinatario potra aprire, conoscendo la chiave, l'hash generato e verificare che il messaggio non è stato alterato.

Nel caso si usi un sistema a chiave **asimmetrica**, si invertono i ruoli delle chiavi **pubblica** e **privata**: la prima viene usata per decrittografare l'*hash* (in fase di verifica), mentre la seconda per **crittografare** l'*hash* (in fase di firma).

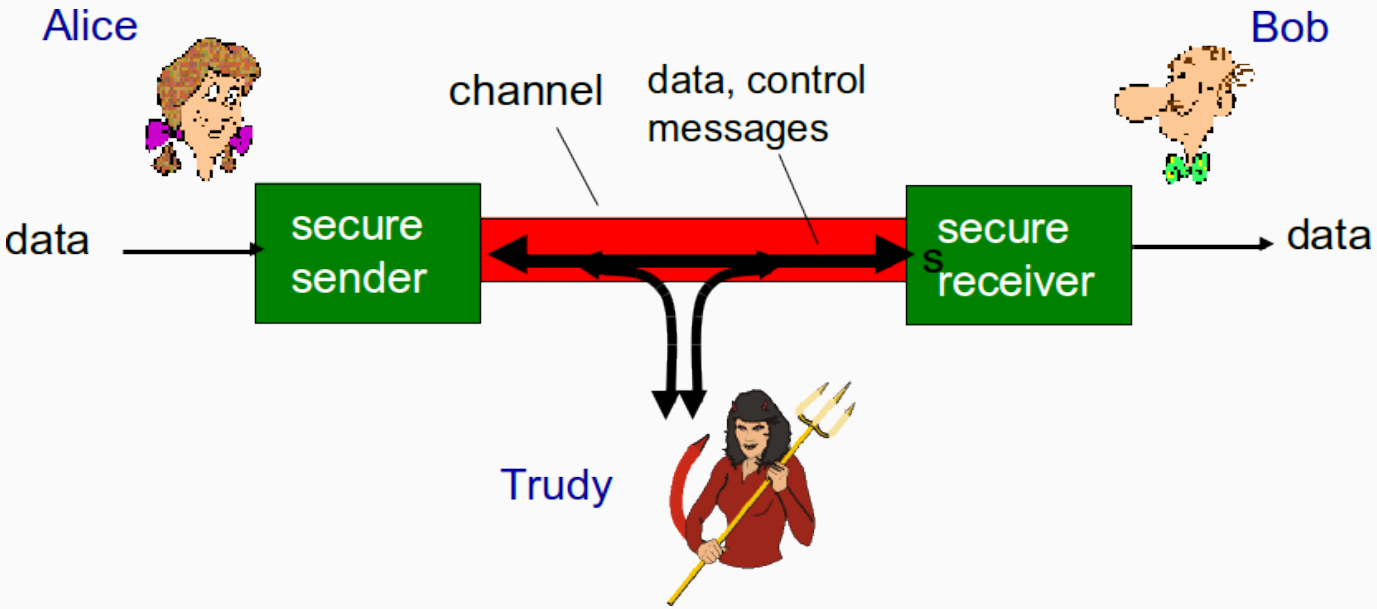
La Fiducia in un sistema asimmetrico

Vi presento Alice, Bob e Trudy

Immaginiamo che *Alice* debba inviare a *Bob* un file importante, e di volerlo crittografare con una coppia di chiavi asimmetriche per evitare che *Trudy* possa leggerlo.

Per farlo, *Alice* userà la sua **chiave pubblica**, così che **solo** *Bob* possa, con la sua **chiave privata**, leggere il contenuto del file.

Ma come possiamo avere la chiave pubblica di Alice in modo sicuro?



La chiave di Alice

Bob, per ottenere la chiave pubblica di *Alice*, può:

- Scaricarla da internet o farsela inviare via e-mail: in questo modo però *Trudy* potrebbe intercettare la comunicazione e scambiare la chiave pubblica di *Alice* con la sua; così *Bob* crittografa il file con la chiave di *Trudy* anziché di *Alice*, e quindi *Trudy* può leggere il contenuto
- Usare un server esterno: e se *Trudy* controllasse quel server?
- Incontrare *Alice* di persona: il metodo è sicuro, ma non sempre è fattibile
- Fidarsi di *Charlie* (del quale anche *Alice* si fida), per scambiare la chiave

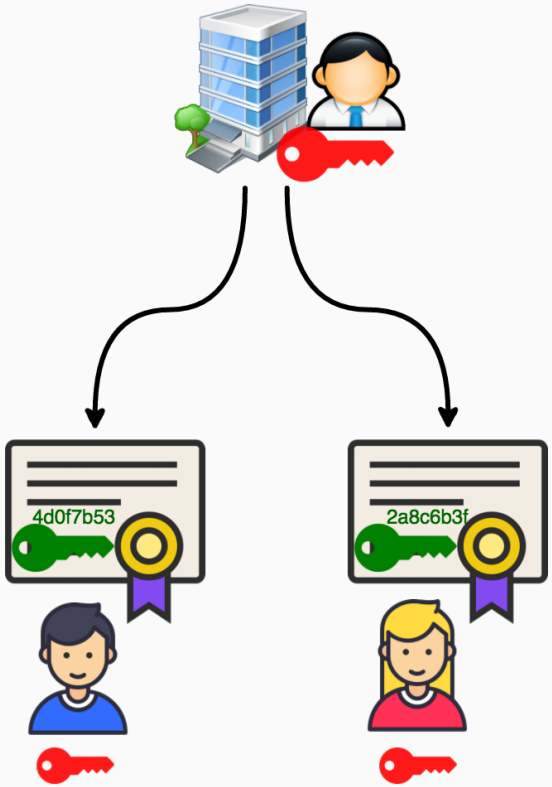
Ogni sistema ha le sue falle e le sue peculiarità. In generale due approcci sono usati per risolvere questo problema: **PKI** e **Web-of-trust**

PKI e web-of-trust

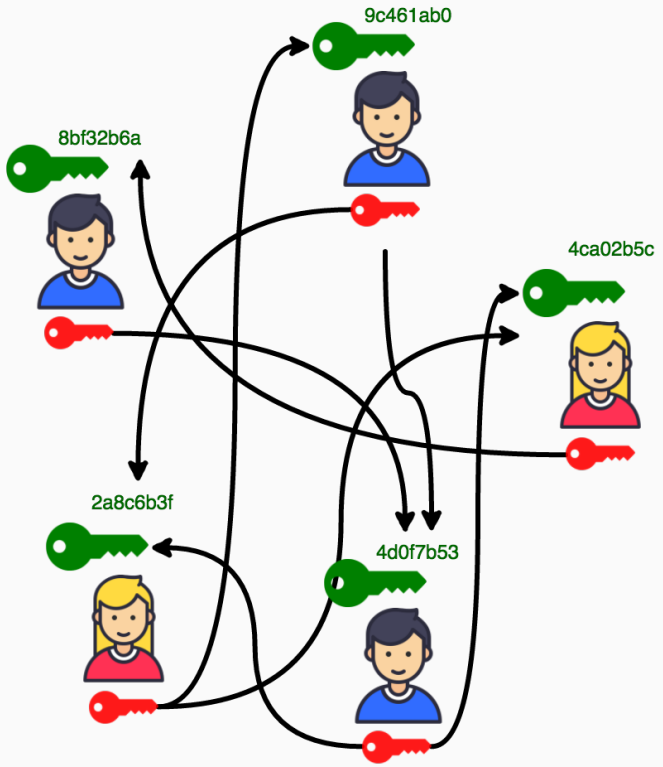
- **Public Key Infrastructure**: la fiducia in una chiave (e le informazioni in essa contenute) si basa sulla fiducia in un terzo (rispetto ad *Alice* e *Bob*) che garantisce per entrambi
- **Web-of-trust**: la fiducia in una chiave si basa su una rete di persone fidate tra *Alice* e *Bob*

PKI e web-of-trust

CERTIFICATE AUTHORITY



PGP / WEB OF TRUST



Public Key Infrastructure

Nella *Public Key Infrastructure* vengono generate delle coppie di chiavi pubblica-privata per un ente detto *Certification Authority*. Dopodiché la *CA* fornisce a tutti una copia della sua chiave pubblica (in modo "sicuro") e firma (con un procedimento che vedremo dopo) le chiavi delle persone di cui si fida.

In questo modo si crea una catena di certificati (detta *certificate chain*) che portano da *Alice* alla *CA*. *Bob*, fidandosi della *CA*, si fida anche della chiave di *Alice* firmata dalla *CA*

Public Key Infrastructure (2)

Le chiavi delle *autorità di certificazione* vengono inserite in modo predefinito all'interno del sistema operativo, anche se le applicazioni possono avere una loro lista "personale" (ad esempio i browser).
Installare nel proprio computer un certificato CA equivale quindi a fidarsi di tutti i certificati emessi da quella CA

In generale, sono inserite come *CA* enti importanti come:

- Verisign
- Microsoft
- Thawte
- GlobalSign
- ...

Web-of-trust

Usando un sistema di **web-of-trust** invece si crea una rete di persone che firmano e danno "fiducia" alle chiavi di altre persone. In particolare, ci sono due operazioni:

- **Firma della chiave** di una persona (di un conoscente) dopo che si è verificato che la persona sia veramente lui (ad esempio, se *Bob* è cresciuto insieme a *Charlie*, allora, dopo aver ottenuto a mano la chiave da *Charlie*, la firmerà così da confermare che la chiave è associata a lui)
- **Fiducia in una chiave**: si può impostare una fiducia (di default è nulla) su di una chiave, in base a quanto ci si fida della persona: questo cambia il modo in cui si considerano le chiavi firmate da quella persona (esempio *Bob* si fida completamente di *Charlie*, quindi **ha fiducia nel fatto che Charlie verifichi le chiavi che firma**)

PGP/GPG e Web-of-trust

PGP/GnuPG

PGP, acronimo di *Pretty Good Privacy* è un software scritto da *Phil Zimmermann* nel 1991 come sistema di crittografia a chiavi asimmetriche. Dal suo software sono derivati i prodotti commerciali **PGP Corp** (attualmente di *Symantec*) e lo standard libero **OpenPGP**.

GnuPG (GPG) è una implementazione libera (con licenza GPL) dello standard OpenPGP, fa parte dei progetti *GNU*. GPG è un software a riga di comando, ma esistono interfacce grafiche per facilitare l'utilizzo.

GPG e la fiducia

I livelli di fiducia a disposizione con PGP/GPG sono:

- **Sconosciuta**: nessun valore di fiducia associato (di default quando importiamo le chiavi dentro il nostro archivio locale)
- **Nessuna**: nessuna fiducia, la persona firma chiavi impropriamente (eg. senza verificarne il proprietario)
- **Marginale/parziale**: sappiamo che la persona comprende le implicazioni del sistema e firma propriamente le chiavi
- **Completa**: la persona conosce a fondo il sistema e possiamo fidarci completamente

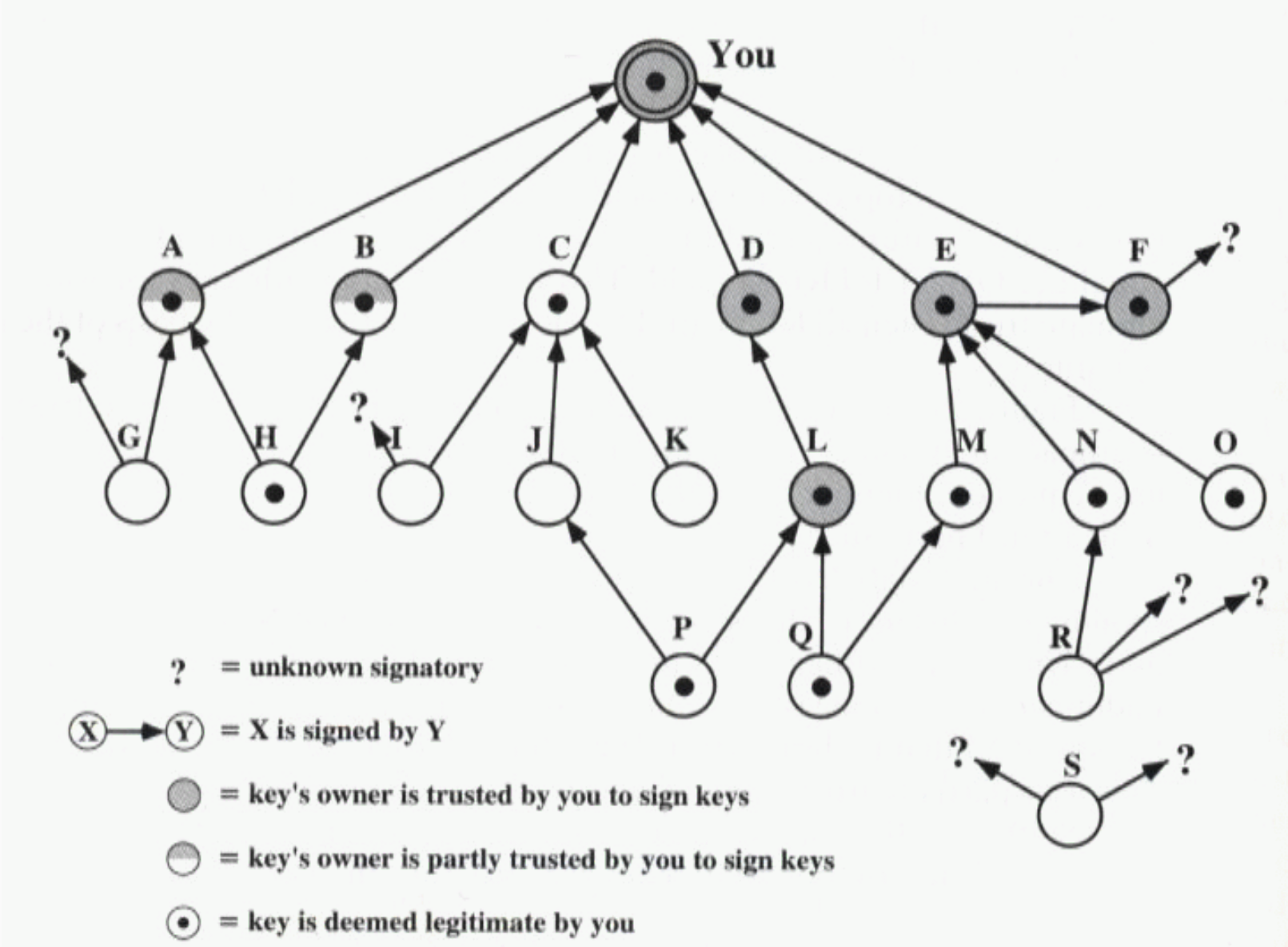
GPG e la rete di fiducia

I livelli di fiducia sono utilizzati per determinare se una chiave è affidabile, per transitività. Per far questo, GPG considera una chiave **K** valida se:

- **K** è firmata da un numero sufficiente di chiavi valide, ovvero:
  - **K** è firmata personalmente da te
  - **K** è firmata da una delle chiavi che ritieni **completamente** affidabili
  - **K** è firmata da almeno 3 chiavi **parzialmente** affidabili
- il percorso tra la tua chiave e **K** è al massimo di 5 passi (5 chiavi)

I numeri qui esposti sono i valori di default di GPG, ma possono essere modificati a piacimento

GPG e la rete di fiducia



Hands-on: Usare GPG

PRE: impostazioni consigliate

Configurazione consigliata per ‘ /.gnupg/gpg.conf‘:

```
personal-digest-preferences SHA256
cert-digest-algo SHA256
default-preference-list SHA512 SHA384 SHA256 SHA224
    AES256 AES192 AES CAST5 ZLIB BZIP2 ZIP
    Uncompressed
```

PS: da ‘default-preference-list‘ a ‘Uncompressed‘ è solo una riga

Generare una chiave con GPG

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

I passi (guidati) per la creazione sono:

1. Scelta degli algoritmi, tra DSA+RSA, DSA+ElGamal, DSA (solo firma) o RSA (solo firma)
2. Scelta della lunghezza della chiave (in bit)
3. Scelta della data di scadenza
4. Specifica dello user-id (nome, e-mail ed eventuale commento)
5. Password di protezione per la chiave privata

A questo punto vengono stampate diverse informazioni riguardo alla chiave: le più importanti sono l'e-mail dello user-id e l'ID della chiave.

Lista delle chiavi sul PC

Possiamo visualizzare la lista delle chiavi pubbliche presenti sul nostro PC con il comando:

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

Mentre per le chiavi private:

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

Crittografare un file: per noi

```
$ gpg --encrypt --recipient nostro@user.id nomefile.ext
```

Il sistema provvederà a crittografare il file (salvandolo come ‘nomefile.ext.gpg‘) con la chiave pubblica dello user-id indicato (in questo caso il nostro)

Per decrittografarlo, dobbiamo avere la chiave privata sul PC e dobbiamo digitare:

```
$ gpg nomefile.ext.gpg
```

GPG ci chiederà la password per sbloccare la chiave privata, e creerà il file ‘nomefile.ext‘ con il plaintext.

Crittografare un file: per qualcun altro

Immaginando che Bob abbia, come e-mail nello user-id della chiave, ‘amico@user.id‘:

```
$ gpg --encrypt --recipient amico@user.id nomefile.ext
```

Il sistema provvederà a crittografare il file (salvandolo come ‘nomefile.ext.gpg‘) con la chiave pubblica dello user-id indicato (in questo caso, **‘amico@user.id‘**).

Fatto questo, non possiamo leggere il file ‘*.gpg‘: per farlo ci serve la chiave privata di Bob, **che non abbiamo**! Solo lui potrà decrittografarlo con il comando visto nella precedente slide.

Integrità: firmare un file

E' possibile firmare un file in modo "autocontenuto", ovvero salvando firma e file originale all'interno di un solo file:

```
$ gpg --sign --recipient nostro@user.id file.txt
```

Il sistema provvederà a firmare il file (salvandolo come 'nomefile.ext.gpg') con la chiave privata del nostro user-id.

Se si vuole lasciare separato il file originale (ad esempio per file di grandi dimensioni può esser comodo):

```
$ gpg --armor --detach-sig --recipient nostro@user.id file.txt
```

In questo caso il file '*.gpg' conterrà **solo** la firma.

Integrità: verificare una firma

Per verificare la firma di un file autocontenuto (firma+originale):

```
$ gpg --verify file.txt.gpg
```

Mentre, per estrarre il file originale:

```
$ gpg --output file.txt --decrypt file.txt.gpg
```

Altrimenti, se si sono separati firma e file originale, è possibile verificare la firma con:

```
$ gpg --verify file.txt.gpg file.txt
```

Revocare una chiave

Se la chiave è stata compromessa, o se la si vuole per qualche motivo dismettere (cessato utilizzo, etc), bisogna dapprima generare la revoca:

```
$ gpg --armor --output revoke.asc --gen-revoke KEYID
```

E poi importare il file di revoca:

```
$ gpg --import revoke.asc
```

Questa azione è irreversibile.

E' possibile comunque continuare a decrittografare i file e verificare le firme, ma non è più possibile generare firme valide o crittografare nuovi file.

Web-of-trust: i keyserver

Per comodità, le chiavi pubbliche possono essere inviate/ricevute a/da "keyserver", ovvero server appositamente creati per ospitare chiavi. Per pubblicare (o aggiornare) la nostra chiave:

```
$ gpg --keyserver pgp.mit.edu --send-key IDCHIAVE
```

Per scaricare la chiave di qualcuno:

```
$ gpg --keyserver pgp.mit.edu --recv-key IDCHIAVE
```

I server più comuni sono ‘pgp.mit.edu‘ e ‘pool.sks-keyservers.net‘.

Web-of-trust: firmare le chiavi di altre persone

```
$ gpg --recipient nostro@user.id --sign-key KEYID
```

Dopo averle firmate, è conveniente re-inviare la chiave sui keyserver, in modo che la firma sia visibile a tutti.

Email e GPG

E' possibile firmare (e/o crittografare) le e-mail in uscita e verificare la firma (e/o decrittografare) le e-mail in entrata nei vari client di posta elettronica tramite plugins.

- Thunderbird: Enigmail
- MS Outlook: Gpg4win
- Apple Mail.app (macOS): GPGMail

Questions?

Conclusioni

https://github.com/Enrico204/ld2017-gpg