

Crittografia e OpenSource

Matteo Carli

matteo@matteocarli.com

<http://www.matteocarli.com>

<http://www.lug-acros.org>

Chi sono

- Studente di “Sicurezza dei sistemi e delle reti informatiche” presso il polo di Crema (Uni Milano)
- Collaboratore per aziende che forniscono servizi per il web
- Appassionato di sicurezza e networking (membro del CiscoUserGroup di AreaNetworking)

Crittografia

- Scienza delle scritture segrete
- Nata come raccolta di tecniche e sistemi per nascondere messaggi
- Sviluppata nei primi anni del '900 con l'avvento del concetto di informazione
- Migliorata dopo il 1969 e successivamente con l'introduzione del cifrario RSA

Perchè OpenSource?

- Non si può basare la sicurezza di un sistema sulla segretezza dei dettagli tecnici dell'algoritmo.
- La crittografia (considerata come scienza) è sicura perché basata sulla matematica.
- Le applicazioni crittografiche hanno i loro difetti perché sviluppate dall'uomo.
- Solamente con la circolazione degli algoritmi è possibile correggere eventuali errori.

Cifrari

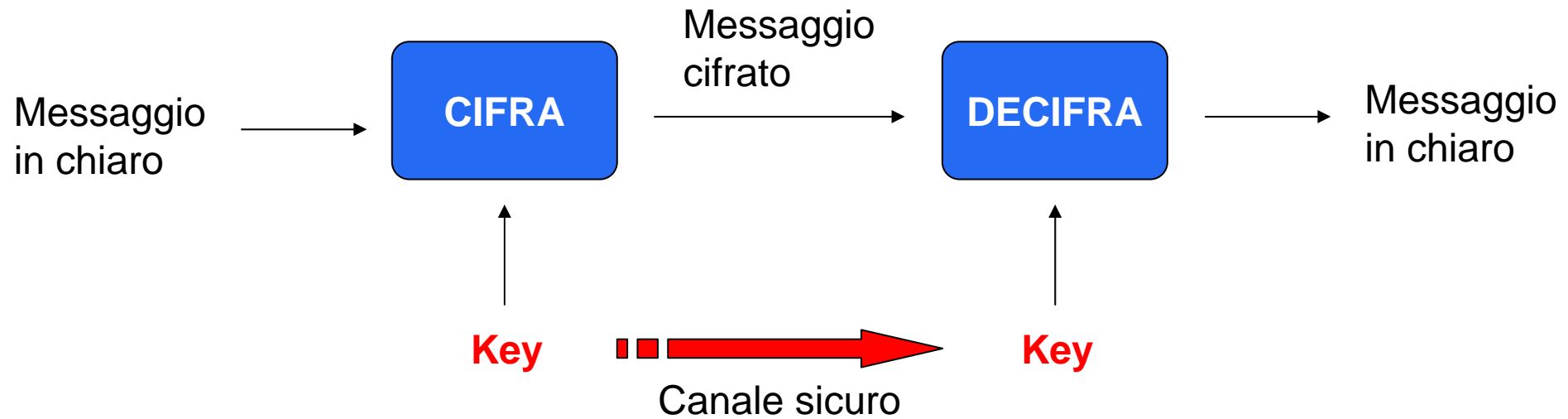
- **Simmetrici:** viene utilizzata una sola chiave per cifrare e decifrare un testo
- **Asimmetrici:** viene utilizzata una chiave per cifrare (chiave pubblica) e una chiave per decifrare (chiave privata) un testo

Un cifrario è composto da un algoritmo che consente di cifrare e decifrare un testo grazie all'uso di una o più chiavi.

Cifrari simmetrici

- La chiave, per la cifratura e la decifratura, deve essere comunicata (tramite un canale sicuro)
- La velocità di elaborazione è elevata
- Alcuni esempi: DES, 3DES, Blowfish, Rinjndael

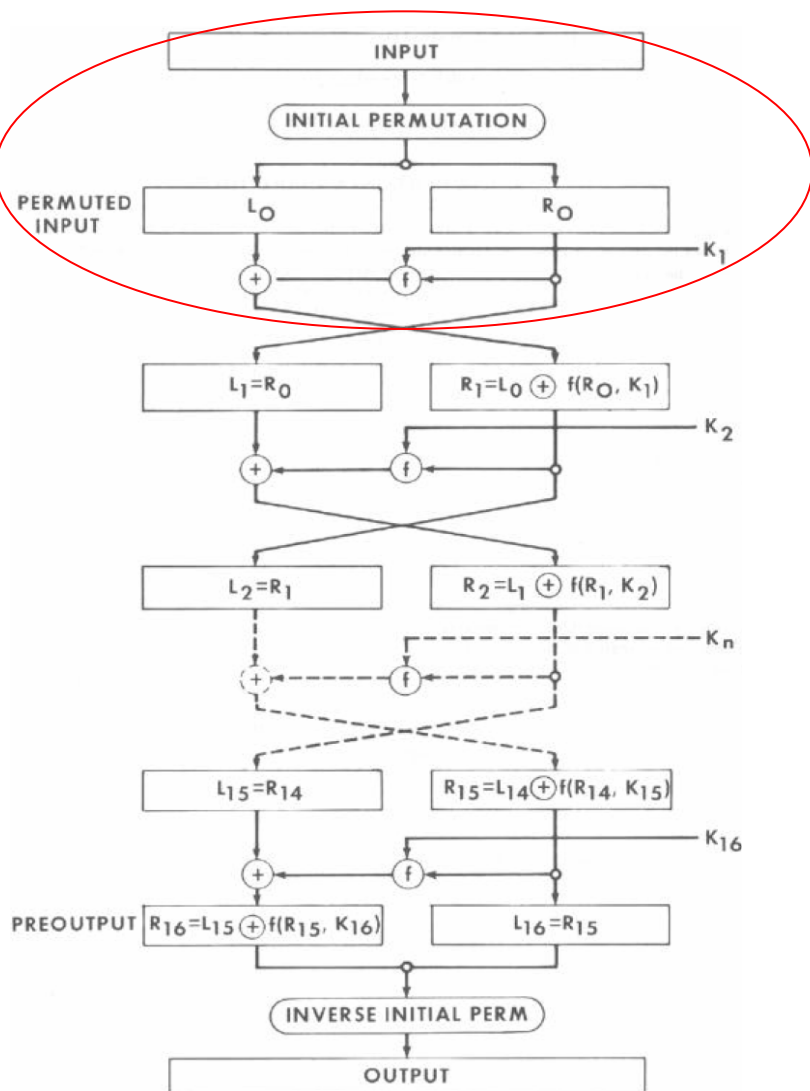
Cifrari simmetrici (2)



DES

- Sviluppato da IBM all'inizio degli anni '70
- Diventato nel 1976 lo standard mondiale per la protezione delle comunicazioni **commerciali**
- Algoritmo di dominio pubblico, la sicurezza è affidata alla chiave
- Attaccato con successo nel 1998 tramite "Deep Crack", quindi altamente insicuro.

DES (2)

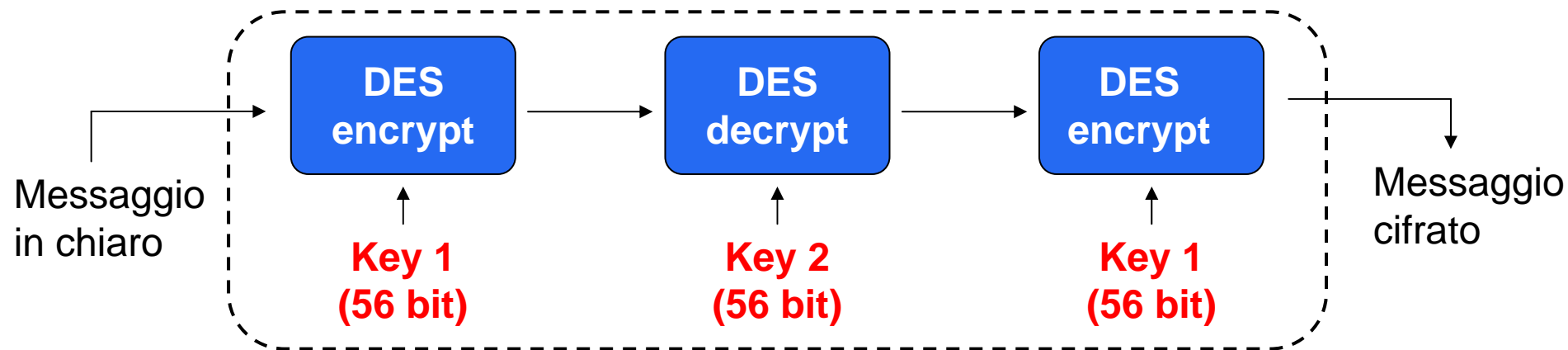


Caratteristiche:

- Utilizzo ciclico di 16 reti di Feistel (il cerchio rosso rappresenta una rete)
- Decifrazione mediante ordine inverso delle SOLE sottochiavi
- Il crittogramma ha la stessa lunghezza del messaggio in chiaro
- Block cipher da 64bit
- Chiave da 56 bit + 8 bit di controllo
- 16 sottochiavi da 48 bit

3DES

- Evoluzione del DES
- Basato sulla tecnica EDE (encrypt, decrypt, encrypt)
- Utilizzo di due chiavi a 56 bit = 112 bit



Cifrari asimmetrici

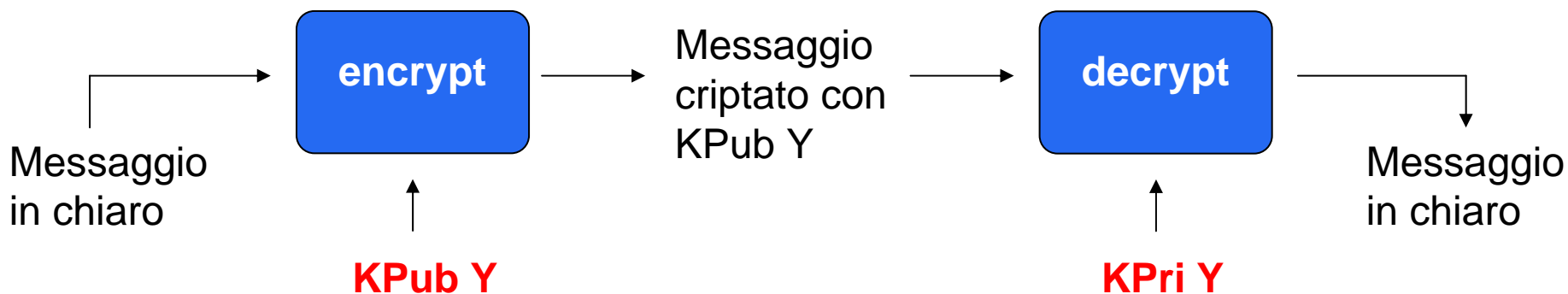
- Viene introdotta una seconda chiave, chiamata “**chiave pubblica**”
- La chiave privata non deve essere più comunicata
- La chiave pubblica può essere liberamente distribuita (es: Key Server)
- Possibilità di “**firmare digitalmente**” i messaggi

Cifrari asimmetrici (2)

- La chiave pubblica e la chiave privata sono legate matematicamente
- Introduzione del concetto di CA (Certification Authority)
- La sicurezza è garantita da sistemi matematici (non si conoscono algoritmi per attaccare i cifrari)
- I due algoritmi più famosi: Diffie-Hellman e RSA

Cifrari asimmetrici (3)

- L'utente X vuole spedire un messaggio cifrato all'utente Y
- L'utente X cifra il messaggio in chiaro con la **chiave pubblica** di Y
- L'utente Y che riceve il messaggio lo decifra con la propria **chiave privata**

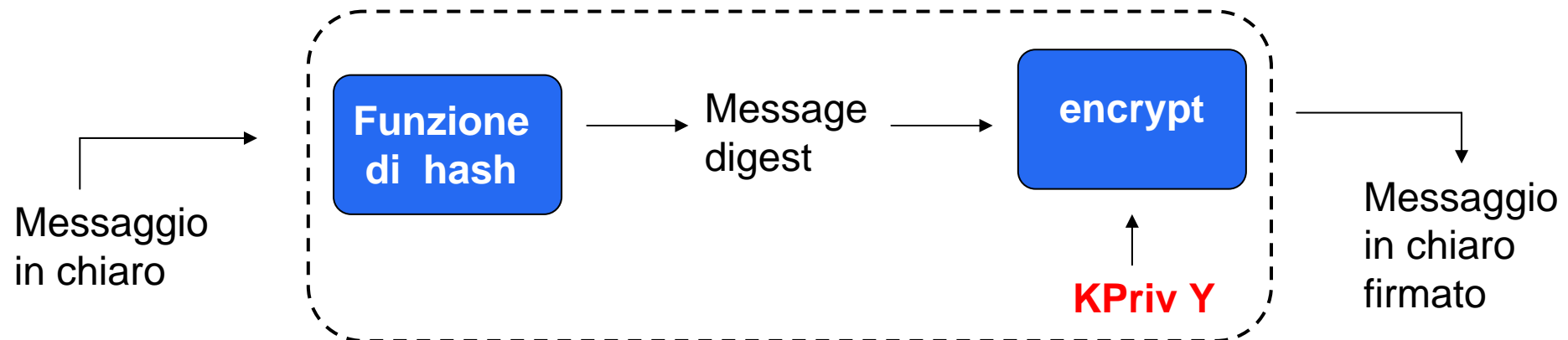


Firma digitale

- Stesso scopo delle firma convenzionale, ma diversa concettualmente.
- Viene creata “l'impronta” del testo originario mediante la propria chiave privata.
- Il testo del messaggio rimane in chiaro.
- La firma ha lunghezza fissa.

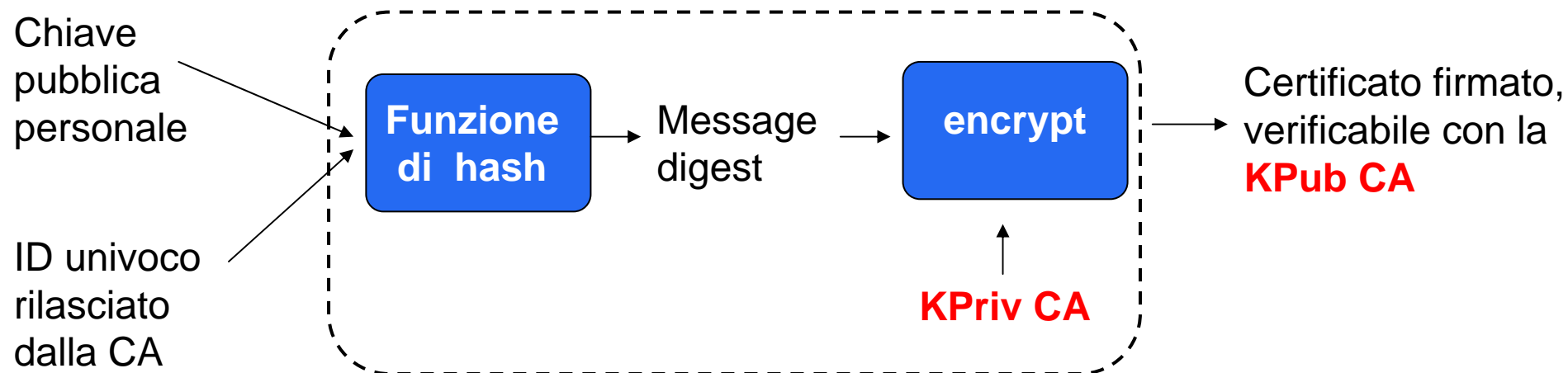
Firma digitale (2)

- L'utente Y vuole spedire un messaggio firmato all'utente X
- Tramite la **funzione di hash** viene generato un message-digest a partire dal messaggio originale
- Il message digest viene crittografato tramite chiave privata dell'utente Y
- L'utente X può comprovare l'autenticità usando la chiave pubblica dell'utente Y



Firma digitale (3)

- La firma è ritenuta **legalmente valida** solo se la CA ha rilasciato un certificato per la coppia **persona-chiave pubblica**
- La CA permette di associare in modo sicuro la coppia **persona-chiave pubblica**



Funzioni hash

- Sono “one way” ovvero non reversibili
- Generano un checksum di lunghezza fissa
- Vengono utilizzate per constatare l'integrità di un testo
- Punto debole: collisioni (checksum identici con testi diversi)
- Alcuni soffrono attacchi di tipo “brute forcing”
- Alcuni algoritmi di hash: MD4, MD5, SHA1, SHA2

Funzioni hash (2)

Attenzione però ad usare funzioni di hashing sicure

- **MD4 è ormai obsoleto**
- **MD5 è considerato insicuro**
- **SHA-0 è altamente insicuro**
- **SHA-1 da un recente studio si è scoperto vulnerabile ad attacchi crittoanalitici (si ottengono collisioni con 2^{63} possibilità)**
- **SHA-2 è la nuova famiglia SHA (SHA-256, SHA-384 e SHA-512)**

GnuPG

- Fratello libero di PGP (licenza GPL)
- Supporta OpenPGP (dalla versione 1.9.19 supporta anche S/MIME)
- Non utilizza algoritmi brevettati
- Tradotto in varie lingue
- Supporto per la scadenza delle chiavi
- Supporta tutti i sistemi operativi più utilizzati
- Sistema ibrido: usa cifrari simmetrici e asimmetrici

GnuPG: creare una chiave

1. `$ gpg --gen-key`
2. Fino alla scelta della durata della chiave compresa potete usare i valori di default.
3. Create un vostro userID, in un secondo momento ne possono essere aggiunti altri.
4. Scegliete un passphrase con rigore e attenzione usando parole non esistenti in un dizionario.

GnuPG: la chiave pubblica

Esportare la propria chiave pubblica usando il proprio id:

```
$gpg --armor --export uid > my_pubkey.asc
```

Importare la chiave pubblica di un altro utente:

```
$gpg --import < another_pubkey.asc
```

GnuPG: la chiave pubblica (2)

- Il punto debole dei cifrari asimmetrici
- Nessuno ci assicura che una chiave pubblica di utente sia realmente sua

Per risolvere questo problema GPG adotta:

- Firma delle chiavi pubbliche
- Rete di fiducia

GnuPG: firmare una chiave pubblica

1. Importare la chiave pubblica
2. `$ gpg --edit-key uid`
3. `Command> fpr`
4. Verificare su un **canale sicuro** la chiave di **fingerprint**
5. `Command> sign`
6. Inserire il proprio passphrase
7. `Command> quit`

GnuPG: crittografare un testo

Criptare:

```
$ gpg --encrypt -r uid --armour < in.txt -o out.txt
```

Decriptare:

```
$ gpg --output doc.txt --decrypt doc_encrypt.txt
```

Criptare con cifrario simmetrico:

```
$ gpg --armor --output out.txt --symmetric in.txt
```

GnuPG: firmare e verificare un testo

Firmare e crittografare:

```
$ gpg --armor --output out.sign --sign in.txt
```

Verificare e decriptare:

```
$ gpg --output doc --decrypt doc.sig
```

GnuPG: firmare e verificare un testo (2)

Firmare:

```
$ gpg --clearsign text.txt
```

Verificare:

```
$ gpg --verify < message.txt.asc
```

GnuPG: estensioni per email client

- Mutt: inserendo delle direttive nel `.muttrc` i due programmi possono cooperare
- Thunderbird e Mozilla: mediante Enigmail, potente plug-in, si possono far cooperare con GnuPG
- Outlook Express: mediante WinPT, front-end e plug-in, si può far cooperare con GnuPG

Riferimenti web

- www.gnupg.org
- winpt.sf.net
- www.schneier.com
- www.eff.org/descracker.html
- theory.csail.mit.edu/~yiqun/shanote.pdf
- www.stachliu.com/collisions.html
- passcracking.ru
- md5.rednoize.com/

Dubbi, domande??

Sono a disposizione per eventuali chiarimenti.

matteo@matteocarli.com