

CPONGO

Tesina presentata da:

Andrea Bruno,
Davide Guida,
Marco Romano



Corso di Sicurezza su Reti
prof. Alfredo De Santis



Sommario

Introduzione

RSA

OpenSSL

cPongo

Bibliografia



Mappa testuale



Introduzione

RSA

OpenSSL

cPongo

Bibliografia



cPongo

- cPongo è un progetto open source che abbiamo sviluppato per l' Esame di Sicurezza su Reti del Corso di Laurea in Informatica dell'Università di Salerno.
- cPongo è l'acronimo di "**C** **P**riate **O**nline **N**ews**G**roup with **O**penssl
- cPongo è un newsgroup gestito su un canale di comunicazione riservato (crittografato e firmato), grazie al quale gli utenti non devono temere eventuali attacchi alla propria privacy.



Sicurezza

*Il tallone d'Achille delle reti informatiche è la **sicurezza**:*

- Non c'è garanzia di autoprotezione e difesa contro eventuali abusi
- Particolare sensibilità all'intercettazione ed all'alterazione dei dati trasmessi nonché alla violazione dei supporti informatici ad essa connessi.



Sicurezza (cont.)

Ciò che viene a mancare all'utente è la "*certezza del contesto*" :

- certezza dell'identità dei corrispondenti (*identificazione*), cioè garantire che la controparte sia realmente chi dice di essere
- certezza delle legittimità dell'operazione (*autorizzazione*), sapere cioè se la controparte sia autorizzata a compiere determinate operazioni
- certezza dell'integrità del messaggio (***autenticazione***), garanzia che il messaggio non sia stato modificato lungo il percorso e sia stato inviato dal legittimo proprietario
- certezza della segretezza della comunicazione (***riservatezza***), in modo che terzi non autorizzati non possano venire a conoscenza di informazioni riservate



Riservatezza

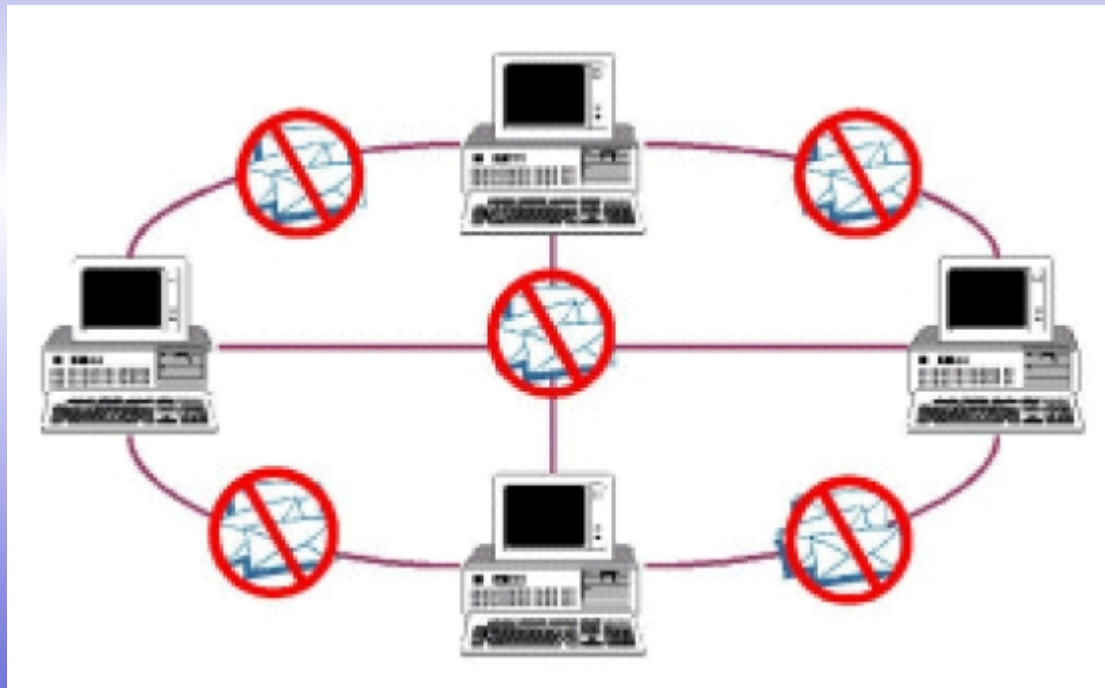
Soluzione: - **tecniche di cifratura e decifratura**

- 1) SISTEMI CRITTOGRAFICI A CHIAVE PRIVATA
(SIMMETRICA)
- 2) SISTEMI CRITTOGRAFICI A CHIAVE PUBBLICA
(ASIMMETRICA)
- 3) SISTEMI CRITTOGRAFICI MISTI.



Riservatezza (cont.)

Una rete sicura:





Riservatezza (cont.)

1) SISTEMI CRITTOGRAFICI A CHIAVE PRIVATA (SIMMETRICA)



- Mittente e destinatario utilizzano la stessa chiave per comunicare



Riservatezza (cont.)

2) SISTEMI CRITTOGRAFICI A CHIAVE PUBBLICA

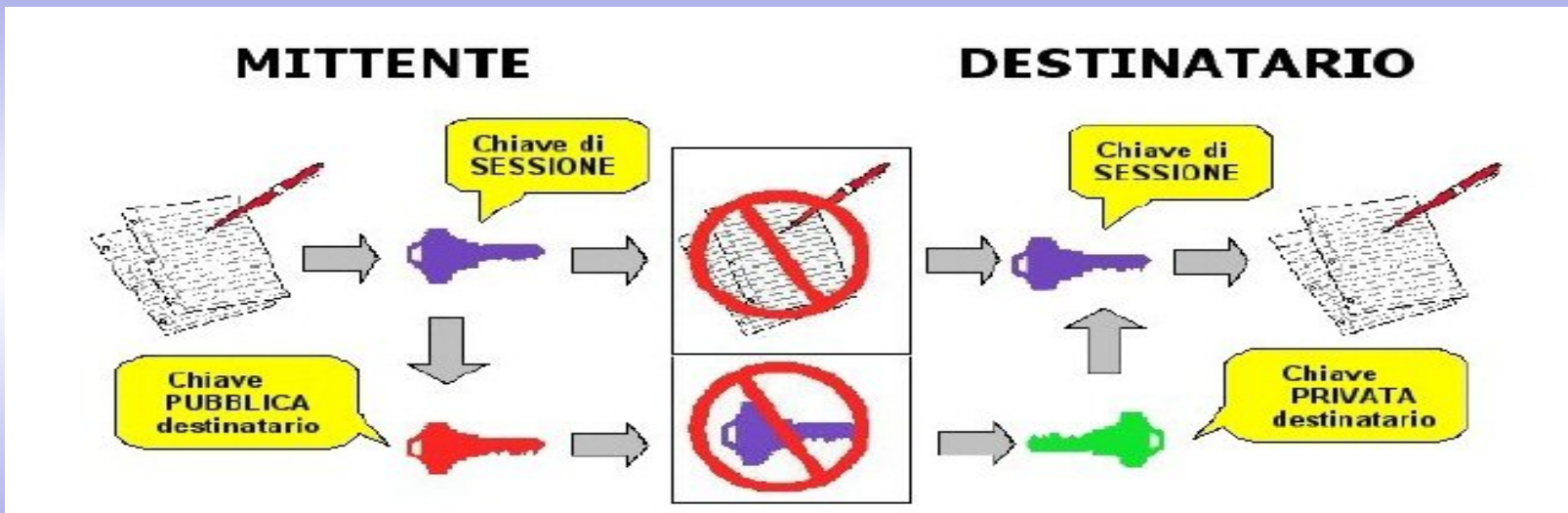


- Due chiavi differenti: pubblica e privata
- La chiave pubblica cifra il messaggio
- La chiave privata lo decifra
- Senza conoscere la chiave privata non si può decodificare



Riservatezza (cont.)

3) SISTEMI CRITTOGRAFICI MISTI



- Ogni utente ha una chiave pubblica e una privata
- Creazione di una chiave simmetrica per ogni sessione
- La chiave simmetrica viene cifrata e allegata al messaggio
- Il destinatario decifra la chiave simmetrica con la sua privata
- La usa per decifrare il messaggio



Autenticazione



L'autenticazione avviene tramite l'applicazione al nostro documento elettronico di una firma digitale.



Mappa testuale

- Introduzione
- *RSA*
- OpenSSL
- cPongo
- Bibliografia



RSA

RSA si basa sul prodotto di due numeri primi di grandi dimensioni, che possono superare le 300 cifre.

due fasi:

- generazione della coppia di chiavi
- utilizzo delle stesse.



RSA (cont.)

Prima fase:

1. vengono scelti due numeri primi p , q molto grandi;
2. viene calcolato $n=pq$, e la funzione di Eulero $\Phi(n) = (p-1)(q-1)$ dopo di che i due primi p , q vengono eliminati;
3. si sceglie un intero e minore di $\Phi(n)$ e primo con esso;
4. utilizzando la versione estesa dell'algoritmo di Euclide viene calcolato l'intero d così da avere $e * d = 1 \bmod \Phi(n)$;
5. vengono resi pubblici i valori e, n che costituiscono la chiave pubblica e mantenuto segreto d che, utilizzato con n rappresenta la chiave privata.



RSA (cont.)

$$C = M^e \text{ mod } n$$



Seconda fase:

C = messaggio cifrato

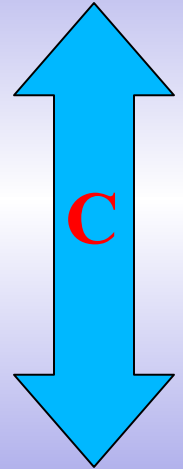
M = messaggio in chiaro

- *Cifratura:*

$$C = M^e \text{ mod } n$$

- *Decifratura:*

$$M = C^d \text{ mod } n$$



$$M = C^d \text{ mod } n$$





RSA (cont.)

- Metodo più efficiente di decifratura:

- Basato sul Teorema Cinese del Resto
- Dati p, q calcolo a, b t.c. $ap + bq = 1$
- $c = bq, d = ap$

- Dato un messaggio cifrato C
- $y = C^{d \bmod p-1} \bmod p$
- $z = C^{d \bmod q-1} \bmod q$
- $M = (cy + dz) \bmod n$

- ✓ al posto di un esponenziale modulo n abbiamo due esponenziali modulo p, q
- ✓ gli esponenti calcolati possono essere conservati per altre operazioni di decifratura



Firma Digitale RSA

Firma ← $M^d \bmod n$

// Firmiamo M utilizzando la *chiave*
// *privata (d ed n)*

Verifica (Firma)

if $M = \text{Firma}^e \bmod n$

// Verifichiamo la firma utilizzando la
// *chiave pubblica (e ed n)*

return TRUE

else

return FALSE





Mappa testuale

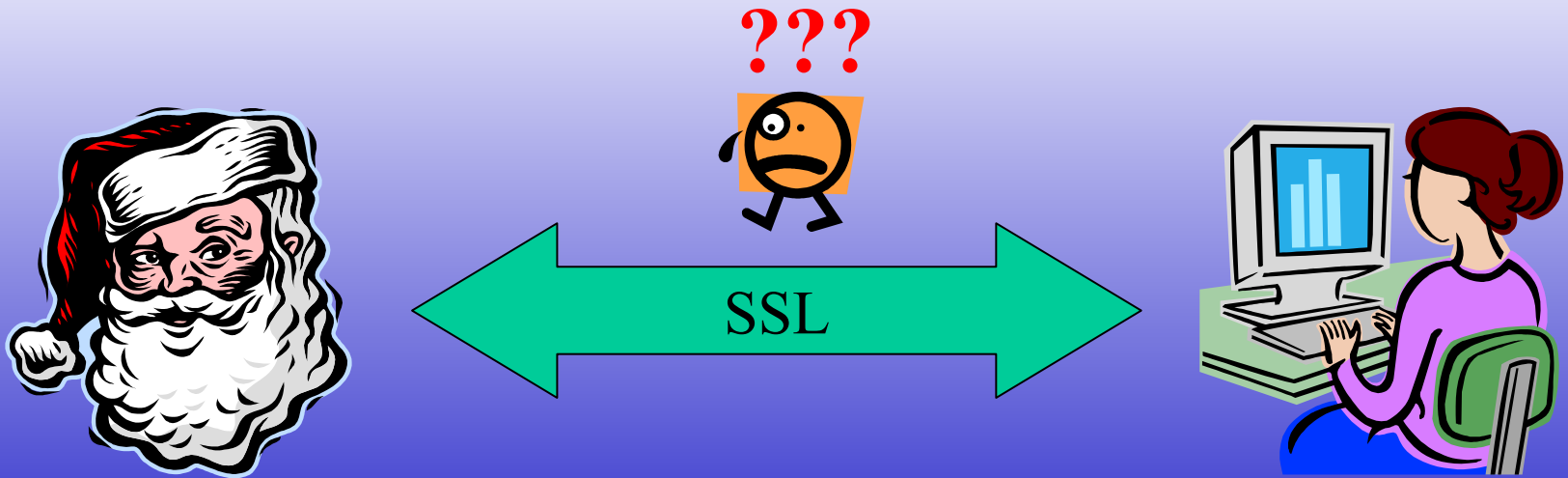
- Introduzione
- RSA
- **OpenSSL**
- cPongo
- Bibliografia



OpenSSL

Lo strumento di cui ci siamo avvalsi per lo sviluppo del nostro programma è l'OpenSSL

- Potente strumento per la sicurezza delle reti
- GNU Public Licence
- Multiplatforma





OpenSSL (cont.)

La struttura di una chiave RSA:

```
struct RSA
```

```
{  
    BIGNUM *n;           // public modulus  
    BIGNUM *e;           // public exponent  
    BIGNUM *d;           // private exponent  
    BIGNUM *p;           // secret prime factor  
    BIGNUM *q;           // secret prime factor  
    BIGNUM *dmp1;        // d mod (p-1)  
    BIGNUM *dmq1;        // d mod (q-1)  
};
```

- p, q, dmp1, dmq1 vengono usati per velocizzare le operazioni di decifrazione
- nella chiave privata possono essere anche NULL
- nella chiave pubblica l'esponente privato e i relativi fattori segreti sono NULL



OpenSSL (cont.)

Queste funzioni effettuano le operazioni di cifratura e decifratura di un messaggio:

```
int RSA_public_encrypt(int flen, unsigned char *from, unsigned char *to, RSA *rsa, int padding);
```

flen: numero di byte da cifrare
from: stringa da crittografare
to: buffer dove viene scritto l'output cifrato
rsa: chiave pubblica RSA
padding: specifica quale tipo di padding debba essere usato per completare la taglia del messaggio

Ritorna la grandezza dei dati crittografati

```
int RSA_private_decrypt(int flen, unsigned char *from, unsigned char *to, RSA *rsa, int padding);
```

flen: numero di byte da decifrare
from: stringa da decrittare
to: buffer che conterrà l'output in chiaro
rsa: chiave privata RSA
padding: questo parametro è posto uguale al valore di padding della funzione encrypt

Ritorna la grandezza del testo in chiaro



OpenSSL (cont.)

Queste funzioni implementano l'operazione di firma e di verifica del messaggio:

```
int RSA_sign(int type, unsigned char *m, unsigned int m_len, unsigned char *sigret, unsigned int *siglen, RSA *rsa);
```

type: indica il tipo di algoritmo che esegue la funzione hash
m: buffer che contiene i dati da firmare
m_len: numero di byte del buffer
sigret: buffer che conterrà la firma
siglen: numero di byte della firma
rsa: puntatore alla chiave privata

```
int RSA_verify(int type, unsigned char *m, unsigned int m_len, unsigned char *sigbuf, unsigned int siglen, RSA *rsa);
```

type: indica il tipo di algoritmo che esegue la funzione hash
m: buffer che contiene i dati da verificare
m_len: numero di byte del buffer
sigbuf: buffer contenente la firma da verificare
siglen: numero di byte della firma
rsa: puntatore alla chiave pubblica



OpenSSL (cont.)

OpenSSL supporta due formati standard per conservare le chiavi e per scambiarle:

- DER (Distinguished Encoding Rules):
 - Formato binario
 - Definito in RFC 2314
 - Utilizzato per i trasferimenti sulle reti
 - Non ideale per comunicazioni testuali (es. E-mail)
- PEM (Privacy Enhanced Mail):
 - Definito in *RFCs 1421, 1422, 1423, e 1424.*
 - Codificato in base 64 (Formato testuale)
 - Ideale per comunicazioni testuali

In generale se abbiamo la necessità di scrivere i dati sul disco, dovremmo utilizzare il formato PEM



OpenSSL (cont.)

Ecco come risulterebbe una chiave privata in formato PEM a 1024 bit :

```
-----BEGIN RSA PRIVATE KEY-----
MIICXQIBAAKBgQDWxE5+qNqE20NQWqRi71vkqroI+52CgzGmKv42BsE6kk1QiVSS
9Qm75bKfEP7Jz83NO/YO5DPHp6ZVr38nY6HCY3Ta/fNU00/xNE3UiOQSK8dn7lbw
7itshIwjPpK09cgL6wos4Sm+lchjdRk5DyTKcRNadWyXSxNsdrR5FWtImwIBAwKB
gQCPLYmpxecDPNeK5xhB9oftxyawp75XAiEZcf7Oryt8YYjgW423TgZ9Q8xqC1SG
iokzfU60mCKFGm7jylTE7RaAXuiq84NDz71wmFjOgqGzePHl4CH5WnPj0GTwfssI
cEBr5WjUu2DTNKAUdk+1jLJFXFY6UIws6tzZB1SPNNagewJBAPAvdSm765vsMcOL
zfIks0rMpx9c4QZrXAgm+IoAN3EQOvZ9KNAK1xkUYDPJEyJ93GY5ZjyPaFnM9t+x
n9711r8CQQDk6GVm8oN9Z5aMPNDSzNGrj+f+xxngEjxDzch2YfDcVy8cb8T4Daqt
vWSP2JWZm4YAWJzACuuQ+zAMBANjQ4ElAkeEAoB+jcSfyZ/LL17KJTBh3hzMaFOiW
BEeSsBn7BqrPoLV8pFNwirHku2LqzTC3bFPoRCZEKF+a5oiklSEVP0PkfwJBAJia
7kShrP5FDwgoizczNnJf7/8vZpVhfYKJK/mWoJLkyhL1Lfqzxx5+QwqQY7u9BAA7
Eyqx8mCndV1YAZotAMMCQQDYzg4eFgmTJuq/J/Ls7NusdBxuwh0fKRt2KPhCZw5r
5mEeCDflQ3ATBjeUt6Z77JG0aEmQUq5NOqd/ju1VtBst
-----END RSA PRIVATE KEY-----
```



OpenSSL (cont.)

Ed ecco la chiave pubblica relativa:

```
-----BEGIN RSA PUBLIC KEY-----  
MIGHAoGBANbETn6o2oTbQ1BapGLvW+Squgj7nYKDMaYq/jYGwTqSTVCJVJL1Cbvl  
sp8Q/snPzc079g7kM8enplWvfydjocJjdNr981TQ7/E0TdSI5BIrx2fuVvDuK2yE  
jCM+krT1yAvrCizhKVhPJmpxEsdRyYegH1p1bJdLE2x2tHkVa0ibAgED  
-----END RSA PUBLIC KEY-----
```



Mappa testuale

- Introduzione
- RSA
- OpenSSL
- *cPongo*
- Bibliografia



cPongo

- Il progetto è composto da due programmi: un server e un client capaci di scambiare messaggi su un canale sicuro
- Tutti i messaggi sono crittografati e firmati per evitare qualsiasi intromissione sul canale
- La crittografia è eseguita con il crittosistema RSA a 1024 bit implementato dalle librerie dell'OpenSSL
- Le chiavi (pubblica e privata) saranno salvate in formato testuale (PEM)
- Per lo scambio dei messaggi sono state usate le API Socket di Berkeley



cPongo (cont.)

Il server è stato progettato per una comunicazione concorrente con infiniti client a cui permette le seguenti operazioni:

- 1) Registrazione
- 2) Autenticazione
- 3) Cancellazione
- 4) Invio messaggi
- 5) Ricezione di tutti i messaggi



cPongo (cont.)



Fase di registrazione



Fase di login



Screenshots

Il server accetta la connessione del client e ne stampa a video la chiave

```
marco@marcolinux: ~/home/marco/Desktop/progetto che funziona/progetto - Shell - Konsole
Sessione Modifica Visualizza Segnalibri Impostazioni Aiuto

[marco@marcolinux marco]$ cd '/home/marco/Desktop/progetto che funziona/progetto'
[marco@marcolinux progetto]$ ./server
attendere l'inizializzazione...
Keys wrote
ho scritto le chiavi con successo!
inizializzazione completata
ok
la chiave è:-----BEGIN RSA PUBLIC KEY-----
MIGHAoGBANzwpQA0Qtqy0qUxn9qrCKeJBdfHoANANzjIzLVAHjzkeji9j0j9wfpazmfYeWx065pMwFWCwZHbiT5PMG1N0b1A4DtFYcGAePyGPETbqpIr07Fu+LX6Jgj7tanKEWRN+N4o60xJKKiSEOEwzTAEKmTaYrNcYcqBX04pSg7lmuzrAgED
-----END RSA PUBLIC KEY-----

DECRITTO: dentro decritto
```

Il client procede con la raccolta dei dati per la registrazione

```
marco@marcolinux: ~/home/marco/Desktop/progetto che funziona/progetto - Shell - Konsole <2>
Sessione Modifica Visualizza Segnalibri Impostazioni Aiuto

[marco@marcolinux progetto]$ ./client 127.0.0.1
.
inizializzo le impostazioni...
Keys wrote
ho scritto le chiavi con successo!
inserisci login password nome e cognome: pippo topolino tullio golia
```



Screenshots (cont.)

Il server ha appena autenticato il client ed è in attesa di una richiesta

```
marco@marco-linux: /home/marco/Desktop/progetto che funziona/progetto - Shell - Konsole
Sessione Modifica Visualizza Segnalibri Impostazioni Aiuto

Readed public key.
la firma è verificata
firma verificata
server: firma: Hpf`H8üý:4æ}0þ!º@ZÜJÚTÀè+èÀÉ!ÐÛPE¹T-ö-

utente non esiste

devo aprire publicpippo.pem
reading public key...
Readed public key.
Readed private key
registrato
aspetto il client

DECRIITTO: dentro decritto
```

Il client mostra il menù all'utente

```
marco@marco-linux: /home/marco/Desktop/progetto che funziona/progetto - Shell - Konsole <2
Sessione Modifica Visualizza Segnalibri Impostazioni Aiuto

DECRIITTO: msg ricevuto

Readed private key
ok nella decriptazione
VERIFICA: leggo
VERIFICA: letto

devo aprire publicnew
reading public key...
Readed public key.
la firma è verificata
ok del server
inserisci 0 per terminare:
inserisci 1 per scrivere un messaggio:
inserisci 2 per cancellarti:
inserisci 3 per riceverei messaggi del forum:
```

KDE 3 KRAFTWURM



Screenshots (cont.)

Validazione di un
messaggio scritto nel
database

```
Sessione Modifica Visualizza Segnalibri Impostazioni Aiuto
MESSAGGIO n.2: pippo 1 come va?
la chiave pubblica del proprietario del messaggio è:
-----BEGIN RSA PUBLIC KEY-----
MIGHAoGBANzwpQA0Qtqy0qUxn9qrCKeJBdfHoANANzjIzLVAHjzKEji9j0j9wfpazmfYeWx065pMWFwCwZHbit5PMG1N0b1A4DtFYcGAePyGPETbqIr07Fu+LX6Jgj7tanKEWRN+N4o60xJKKiSE0EWzTAEKMTaYrNcYcqBX04pSg71muzrAgED
-----END RSA PUBLIC KEY-----

la firma è:
jē7- āÿ:Üf^!SNI'A|_ŽZbKÓKĐÈ!ùē6²2ĐĀfj0`iäc[Ii:i

devo aprire publicpippo2.pem
reading public key...
Readed public key.
la firma è verificata

ATTENZIONE:il risultato della verifica è:1

Messaggio verificato
```



Screenshots (cont.)

Questo è ciò che contiene la cartella del server.

Nei file *.pem sono immagazzinate le chiavi pubbliche dei client registrati.





Manuale d'Uso

- Lanciare il server digitando **./server**
- Lanciare il client digitando **./client "IP del server"** (Loopback: 127.0.0.1)
- Seguire le indicazioni del client per l'accesso al server
- Utilizzare le opzioni offerte dal menù
- Per una rapida lettura dei database, digitare **./leggi**
- Per testare l'efficacia del rilevamento di contraffazioni, digitare:
./brake n, dove n è il numero delle firme dei messaggi, contenuti nel database del server, da corrompere



Mappa testuale

- Introduzione
- RSA
- OpenSSL
- cPongo
- *Bibliografia*



Bibliografia

- Pravir Chandra ,Matt Messier ,John Viega
Network Security with OpenSSL
O'Reilly 2002
- William Stallings
Crittografia e Sicurezza delle Reti
McGraw-Hill 2003
- OndaQuadra0A Elettronic Magazine:
OpenSSL/RSA di Paolo Ardoino
<http://ardoino.altervista.org/>
- Introduction to Cryptography
University of Haifa – Fall 2004
Benny Pinkas
www.pinkas.net/course.html
- La Sicurezza Delle Transazionie E Comunicazioni Attraverso Internet
Materiale prelevato dal sito <http://www.amicopc.com>
- Appunti del corso di sicurezza su reti del docente Alfredo De Santis
- Appunti del corso di reti di calcolatori del docente Roberto De Prisco
- Specifiche del formato PEM: *RFCs 1421, 1422, 1423, e 1424*
- Specifiche del formato DER: *RFC 2314*
- Sito ufficiale OpenSSL: <http://www.openssl.org>