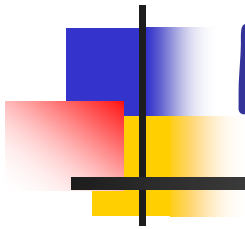
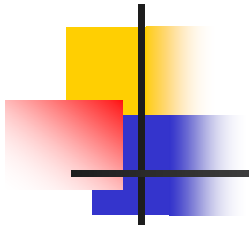


Protocolli di Rete

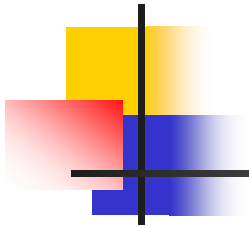
Clemente Galdi

Universita' di Napoli "Federico II"
c.galdi@na.infn.it



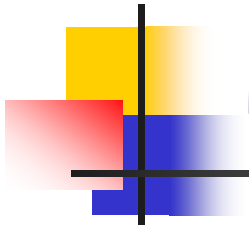
Firme Digitali

- Cos'e': Equivalente digitale della "firma" apposta su un documento cartaceo
- A cosa serve?:
  - Convincere che il firmatario "sottoscrive" con il contenuto del documento
  - Il documento non ha subito modifiche dal momento in cui e' stato firmato



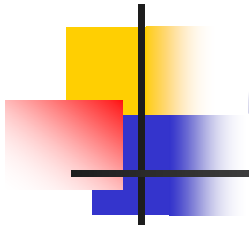
Firme Digitali: Proprieta'

- La firma digitale deve essere facilmente prodotta dal legittimo firmatario
- Il documento firmato non deve poter essere alterato
- Chiunque può facilmente verificare una firma
- La firma non deve poter essere falsificata
- La firma non deve poter essere riutilizzata
- La firma non deve poter essere ripudiata



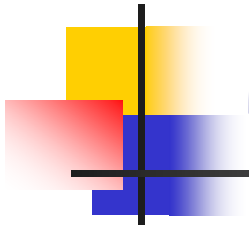
Firma Digitale: Soluzione naive

- Possibile soluzione (?): Concatenare al documento una firma digitalizzata del firmatario
- Facilmente producibile; Autenticita' verificabile
- Ma:
  - Il documento firmato puo' essere alterato
  - La firma non ha bisogno di essere falsificata
  - La firma puo' essere riutilizzata
  - La firma puo' essere ripudiata
- La firma digitale deve dipendere dal documento!



Firma Digitale: Building Block

- Cifrari Asimmetrici:
  - La "chiave" e' costituita da una coppia (public, private)
  - Cifratura mediante chiave pubblica
  - Decifratura mediante chiave privata
  - Le operazioni di cifratura e decifratura sono efficienti
  - Dato il ciphertext, e' difficile calcolare il plaintext *senza la chiave privata*



Firma Digitale: Idea

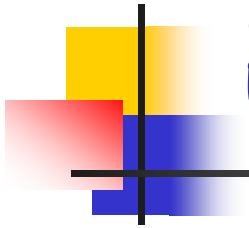
- Usare la chiave privata per firmare un messaggio
  - Firma Efficiente/Non riproducibile/Non falsificabile/Non ripudiabile/Documento non modificabile
- Usare la chiave pubblica per verificare una firma
  - Chiunque puo' verificare una firma

Esempio RSA: Key Setup

- Seleziona due grandi primi casuali - p, q
- Calcola N=p.q
  - nota $\Phi(N)=(p-1)(q-1)$
- Seleziona casualmente la chiave di cifratura e
  - Con $1<e<\Phi(N)$, $\gcd(e,\Phi(N))=1$
- Risolvi la seguente equazione per trovare la chiave di decifratura d
  - $e \cdot d=1 \bmod \Phi(N)$ and $0 \leq d \leq N$
- Pubblica la chiave pubblica : $K_U=\{e,N\}$
- Mantieni segreta la chiave : $K_R=\{d,p,q\}$

Esempio RSA: Encryption

- Per cifrare un messaggio M:
  - Ottieni la **chiave pubblica** del destinatatrio $K_U=\{e,N\}$
  - Calcola: $C=M^e \bmod N$, dove $0 \leq M < N$
- Per decifrare il testo cifrato C:
  - Usa la chiave privata $K_R=\{d,p,q\}$
  - Calcola: $M=C^d \bmod N$
- Nota che il messaggio M deve essere più piccolo del modulo N



Esempio RSA: Signature

- Per Firmare il testo messaggio M:
  - Usa la chiave privata $K_R=\{d,p,q\}$
  - Calcola: $S=M^d \bmod N$, dove $0 \leq M < N$
  - Invia (M,S)
- Per Verificare un coppia (M,S):
  - Ottieni la **chiave pubblica** del firmatario $K_U=\{e,N\}$
  - Calcola: $M'=S^e \bmod N$
  - Se M=M' allora output OK

"Piccolo" esempio: Chiavi RSA

chiave privata
(n=4897, d=3805)

file pubblico

utente	chiave pubblica
Alice	(n = 4897, e = 5)
…	…

4897 = 59 ·83
p = 59, q = 83

ed = 5 ·3805 = 1 mod 4756

(p-1)(q-1) = 58 ·82 = 4756

Alice

"Piccolo" esempio: Chiavi RSA

chiave privata
(n=4897, d=3805)

utente	chiave pubblica
Alice	(n = 4897, e = 5)
…	…

Alice

Devo firmare M=100

100
Alice

"Piccolo" esempio: Firma RSA

chiave privata
(n=4897, d=3805)

utente	chiave pubblica
Alice	(n = 4897, e = 5)
…	…

Alice

Firma di 100
= 100^{3805} mod 4897
= 1320

"Piccolo" esempio: Verifica RSA

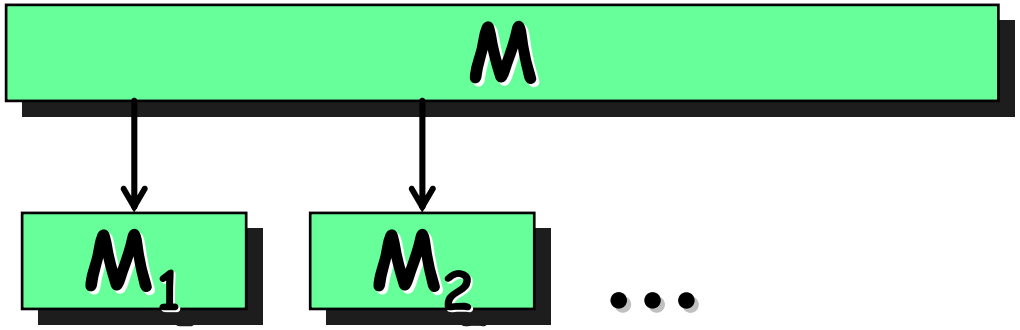
utente	chiave pubblica
Alice	(n = 4897, e = 5)
...	...

Verifica firma di 100

$$100 = 1320^5 \bmod 4897$$

Firme di messaggio grandi

Prima soluzione

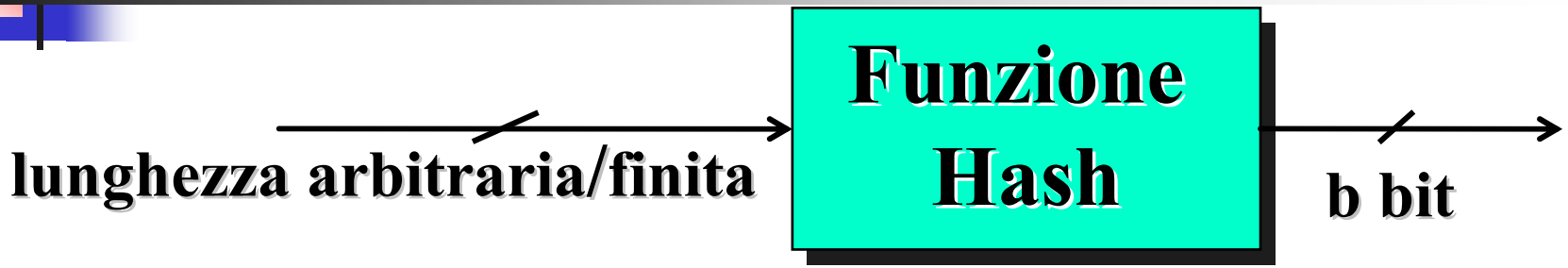


$M_i < n$

$$Firma(M) \leftarrow (Firma(M_1), Firma(M_2), \ldots)$$

Problemi
- Efficienza
- Permutazione/composizione delle firme → nuova firma

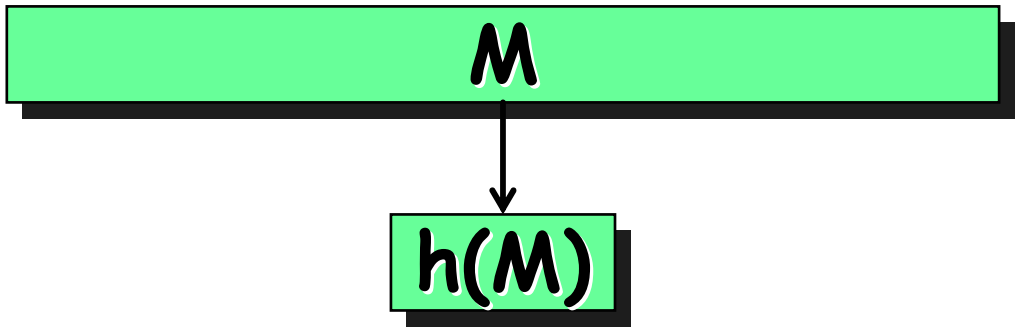
Funzioni Hash



- Il valore hash h(M) è una rappresentazione non ambigua e non falsificabile del messaggio M
- Proprietà:
  - comprime
  - facile da computare
  - **Sicurezza forte**: computazionalmente difficile trovare 2 diversi messaggi con lo stesso valore hash
  - **One-way**: dato y è computazionalmente difficile trovare M tale che y = h(M)

Firma di messaggi grandi

Soluzione

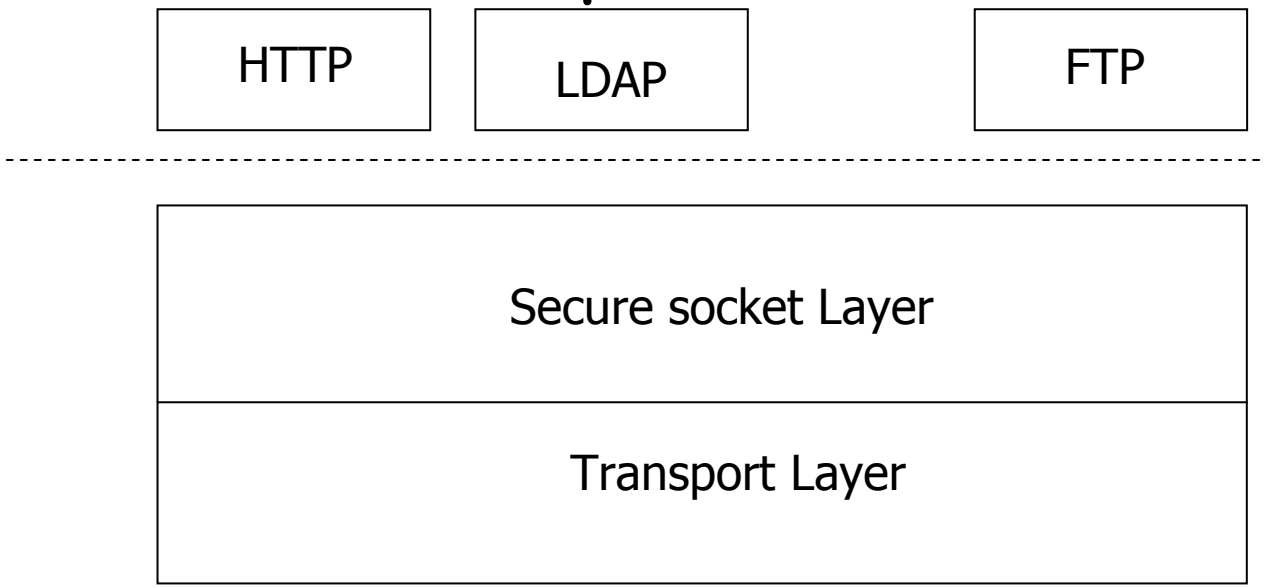


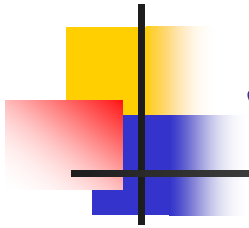
Firma(M) ← (Firma(h(M))

Vantaggi { Efficienza, Integrità

Secure Socket Layer (SSL)

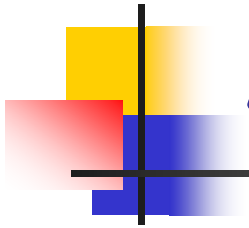
- Opera tra il livello applicazione ed il livello trasporto





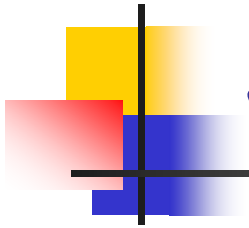
SSL: Caratteristiche

- Consente l'autenticazione per le applicazioni server e client
- Cifra i dati prima di inviarli su un canale pubblico
- Garantisce l'integrità dell'informazione
- Gli algoritmi crittografici utilizzati vengono negoziati tra le parti



Algoritmi usati da SSL

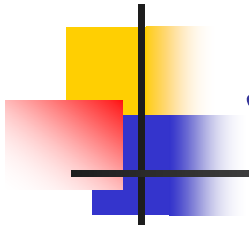
- DES, cifrario simmetrico usato negli USA
- AES, Advanced Encryption Standard, cifrario simmetrico standard NIST
- DSA , Digital Signature Algorithm, fa parte del digital authentication standard usato negli USA
- KEA , Key Exchange Algorithm, un algoritmo usato per lo scambio di chiavi negli USA
- MD5 , Message Digest algorithm
- RC2 e RC4, cifrarsviluppati da Rivest per la RSA Data Security
- RSA , cifrario asimmetrico
- RSA key exchange, un algoritmo per lo scambio di chiavi per SSL e basato su RSA
- SHA-1, Secure Hash Algorithm, una funzione di hash usata dagli USA
- SKIPJACK un cifrario simmetrico
- Triple-DES, DES applicato tre volte



STORIA

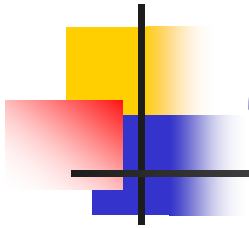
Sviluppato ed introdotto da Netscape

- 1994 versione 1: diversi problemi, mai utilizzata
- 1994 versione 2: implementata in Navigator 1
- 1996 versione 3: implementata in Navigator 3
- 1999 TLS (Transport Layer Security) RFC 2246 evoluzione di SSL proposta dall'IETF



SSL: Componenti

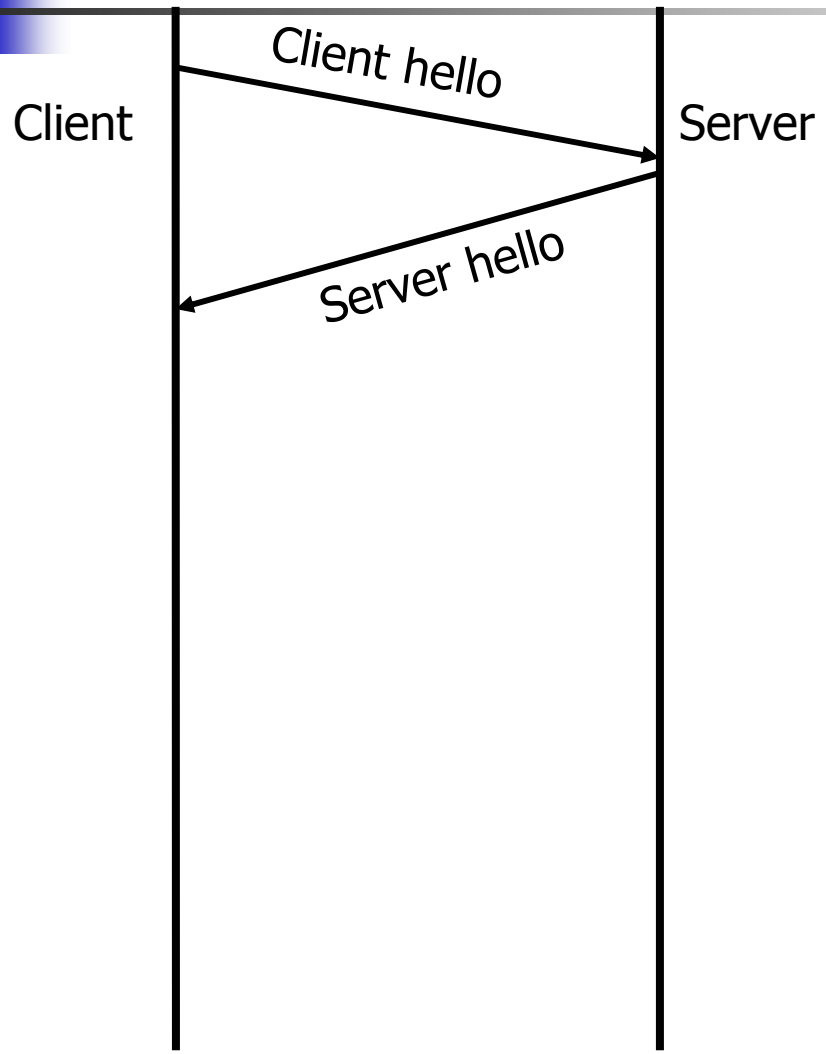
- Handshake*
  - Preposto alla negoziazione della ciphersuite
  - Consente l'autenticazione tra client e server
- Change Cipher Suite
  - Forza la rinegoziazione
- Alert
  - Notifica l'applicazione di eventuali problemi
- Record*
  - Gestisce la comunicazione vera e propria



Handshake di SSL

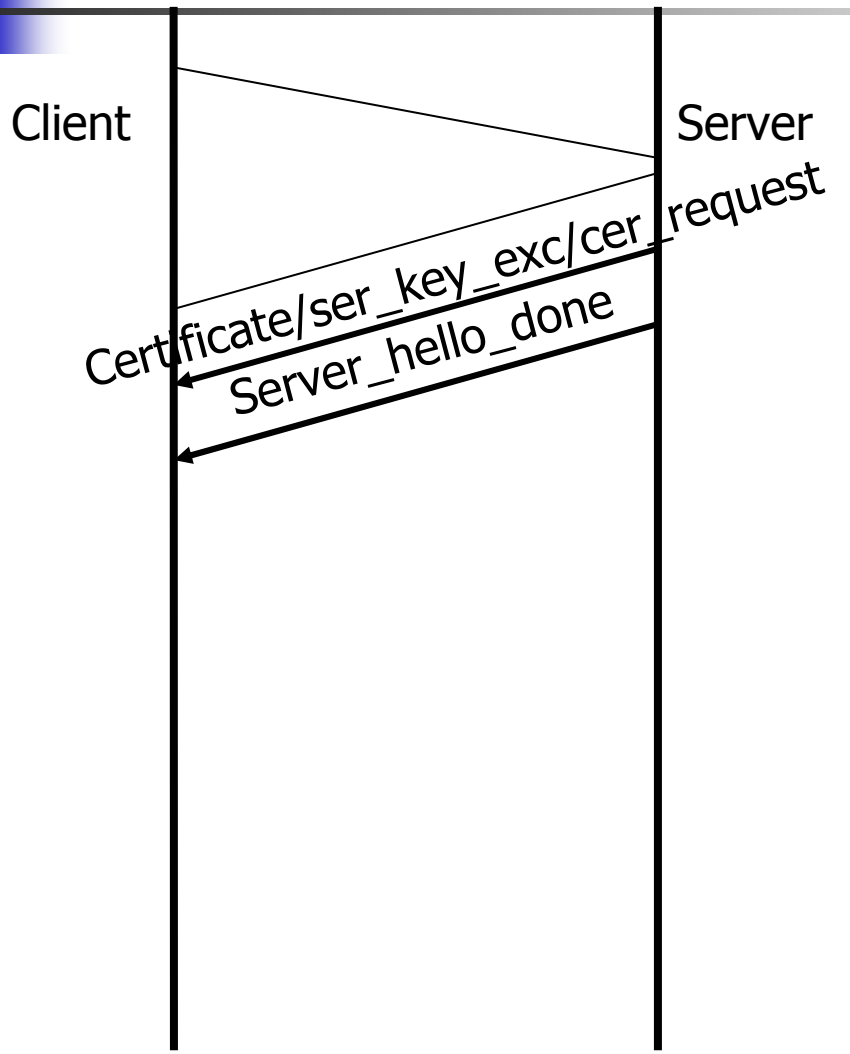
- Gestisce l'autenticazione delle parti
- Consente la negoziazione di:
  - Algoritmo per lo scambio di chiavi
  - Algoritmo per l'autenticazione
  - Algoritmo per la cifratura simmetrica
  - Algoritmo per il MAC

Handshake (fase 1)



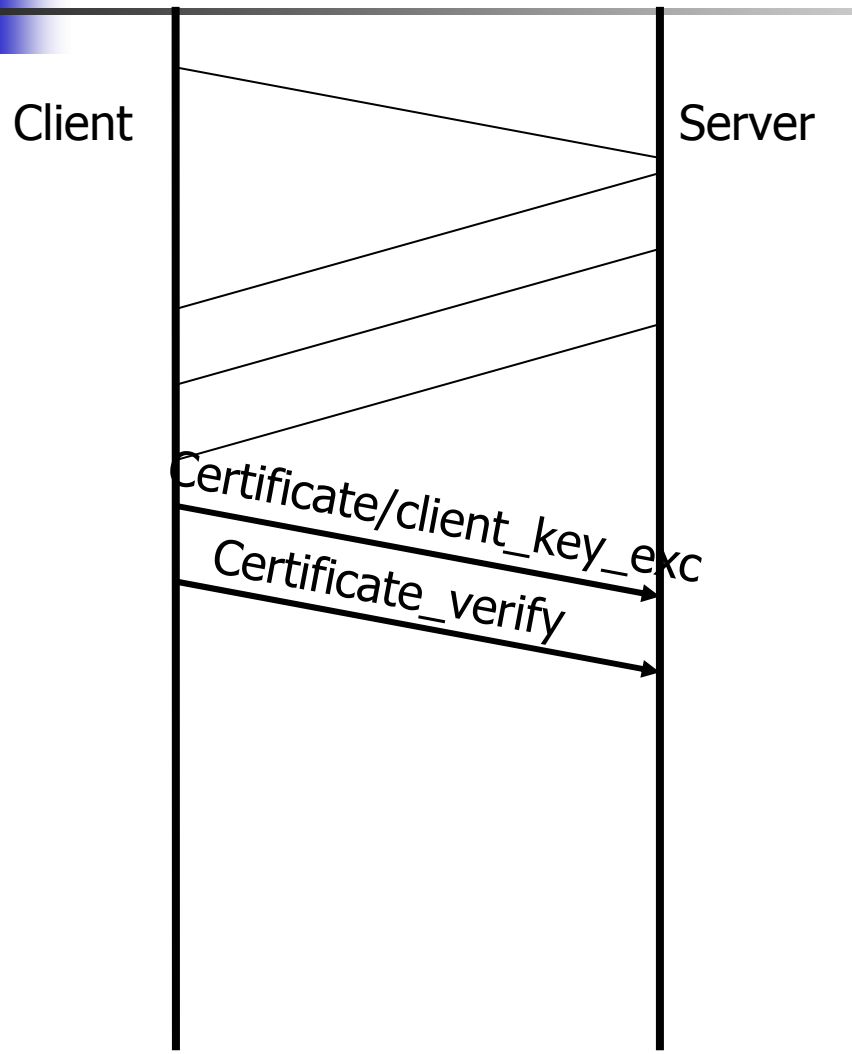
- Il client spedisce al server le informazioni sulla versione del protocollo SSL, ciphersuite, dati random ed altre informazioni (messaggio client hello)
- Il server spedisce al client le informazioni sulla versione del protocollo SSL, ciphersuite, dati random ed altre informazioni (messaggio server hello)
- A questo punto sono stabilite
  - Versione di SSL/TLS
  - Ciphersuite

Handshake (fase 2)



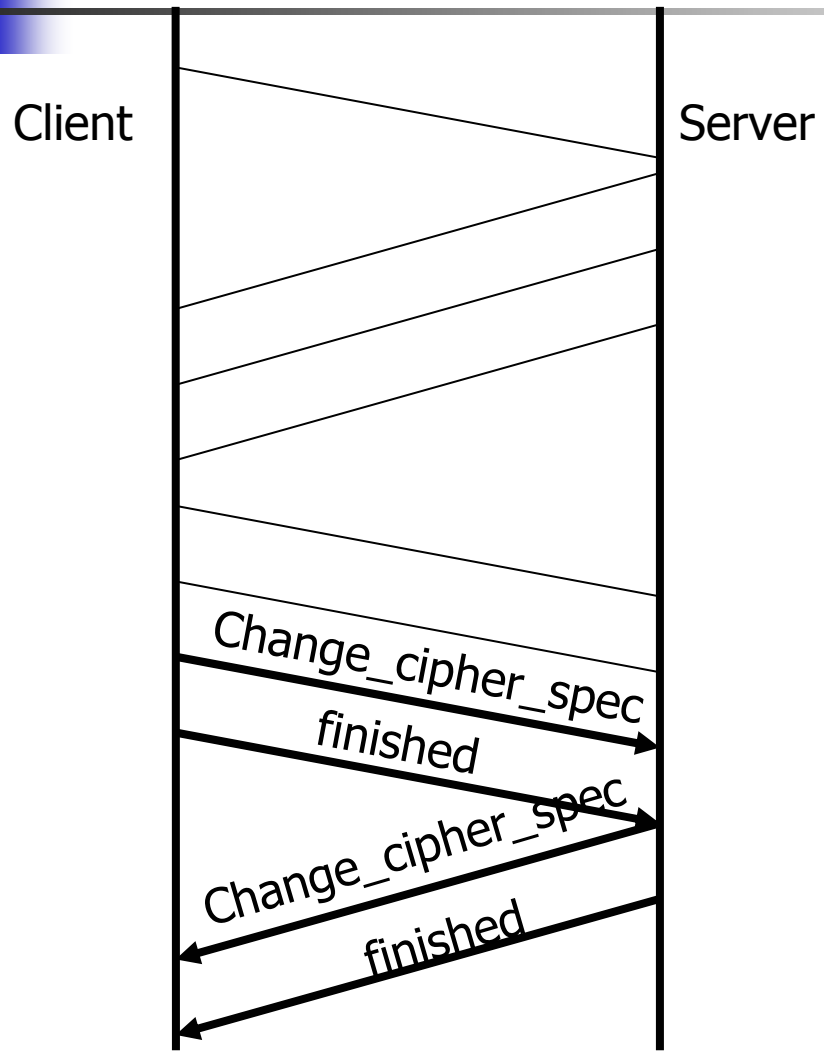
- Il server invia una lista di certificati X509, se richiesti (certificate)
- Il server se necessario invia le informazioni per lo scambio di chiave (server_key_exchange)
- Il server puo' richiedere l'autenticazione del client (certificate_request)
- Invia il messaggio server_hello_done per terminare la fase

Handshake (fase 3)



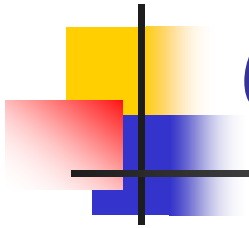
- Il client invia una lista di certificati X509 se richiesto (certificate)
- Il client, se richieste, invia il pre_master_secret (client_key_exchange)
- Il client invia una verifica del certificato (certificate_verify)
- A questo punto
  - Client e serve autenticati
  - E' stabilita una pre_master_secret.

Handshake (fase 4)



- Il client invia il messaggio change_cipher_spec ed il messaggio finished (cifrato)
- Il server invia il messggio change_cipher_spec ed il messaggio finished (cifrato)

Generazione del Key Block



- SSL usa chiavi private per MAC e Cifrari simmetrici e vettori di inizializzazione
- Dato il pre_master_secret (48 byte) client e server calcolano un master_secret (48 byte):

Master_secret =

MD5(pre_master_secret + SHA('A' + pre_master_secret + ClientHello.random + ServerHello.Random)) +

MD5(pre_master_secret + SHA('BB' + pre_master_secret + ClientHello.random + ServerHello.Random)) +

MD5(pre_master_secret + SHA('CCC' + pre_master_secret + ClientHello.random + ServerHello.Random))

Nota: Il master_secret dipende da ServerHello.Random.

Generazione del Key Block

- Data il master_secret (48 byte) viene calcolato il keyblock (la taglia dipende dal ciphersuite)

Keyblock =

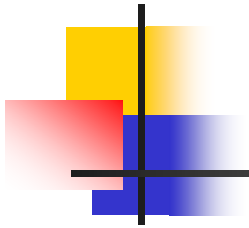
MD5(master_secret + SHA('A' + pre_master_secret +
ClientHello.random + ServerHello.Random)) +

MD5(master_secret + SHA('BB' + pre_master_secret +
ClientHello.random + ServerHello.Random)) +

MD5(master_secret + SHA('CCC' + pre_master_secret +
ClientHello.random + ServerHello.Random))+

...

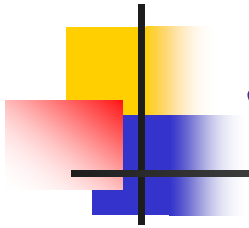
Diffie-Hellman Key Exchange



Sia p un primo ed g un generatore di Z_p^*

- Alice sceglie a<p; calcola $y_A = g^a \bmod p$; invia y_A a Bob
- Bob sceglie b<p; calcola $y_B = g^b \bmod p$; invia y_B ad Alice
- Alice calcola $(y_B)^a = (g^b)^a = g^{ba} \bmod p$
- Bob calcola $(y_A)^b = (g^a)^b = g^{ab} \bmod p$

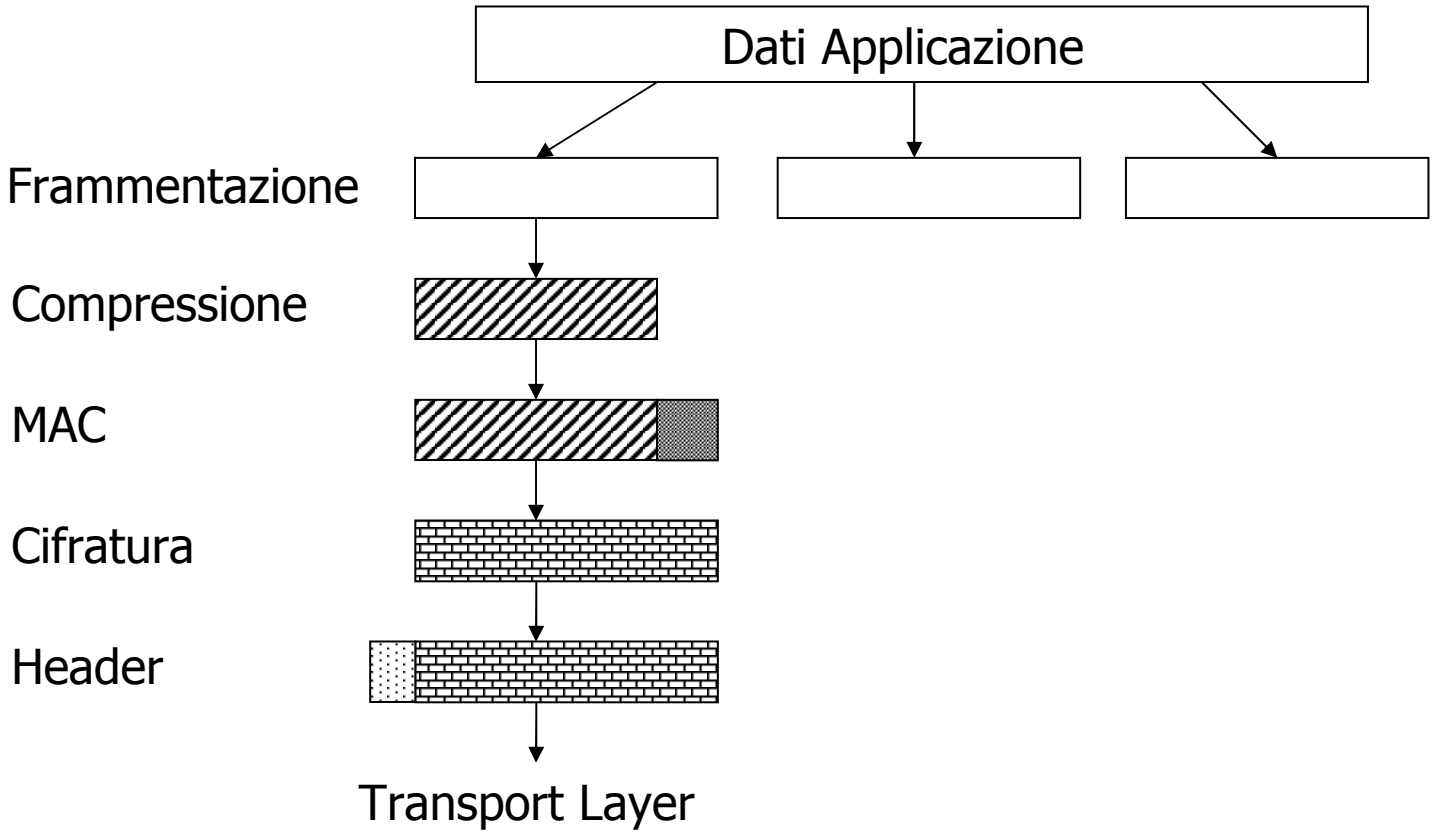
Sicurezza: Basata sulla difficolta' del logaritmo discreto, dato $g^a \bmod p$ e' difficile calcolare a

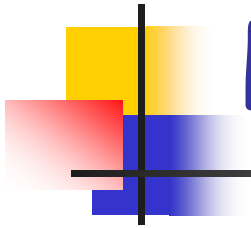


SSL: Key Exchange

- RSA: Il client invia il pre_master_secret cifrato
- Diffie-Hellman:
  - Anonimo (il server invia p,g ed Y_S)
  - Ephemeral (il server invia le informnazioni firmate)
  - Fixed DH (il server possiede un certificato DH contenente le p, g)

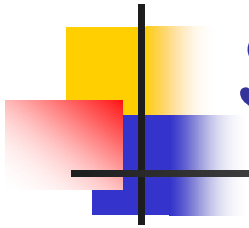
SSL Record


Dati Applicazione
Frammentazione
Compressione
MAC
Cifratura
Header
Transport Layer



Pretty Good Privacy (PGP)

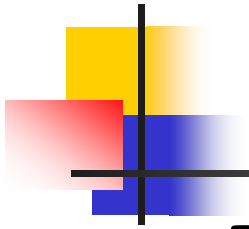
- È un software di crittografia per la posta elettronica e la protezione di file di uso personale
- Creato da Philip Zimmermann nel 1991 e distribuito gratuitamente su Internet
- Cosa fa?
  - Consente di firmare un messaggio lasciando il testo in chiaro
  - Permette di cifrare una email senza firmarla
  - Fare entrambe le cose.



Storia di PGP

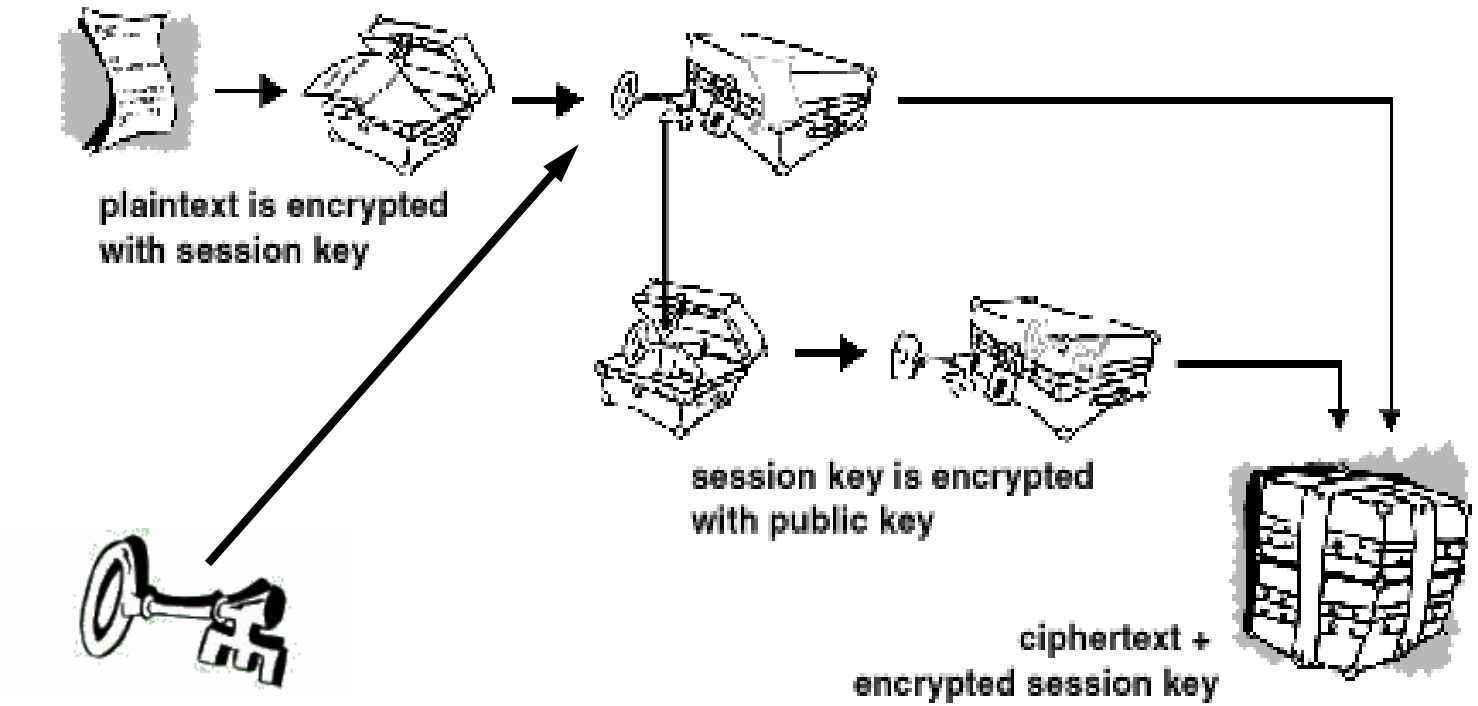
- *PGP 1.0*
  - Prima versione scritta da Zimmermann
  - Usa RSA, Bass-o-Matic, MD4, Lempel-Ziv-Huffman
- *PGP 8.0*
  - Ultima versione freeware ufficiale in ambiente Windows
- *PGP Desktop Professional 9.0*
  - Prima versione commerciale www.pgp.com

Confidenzialità



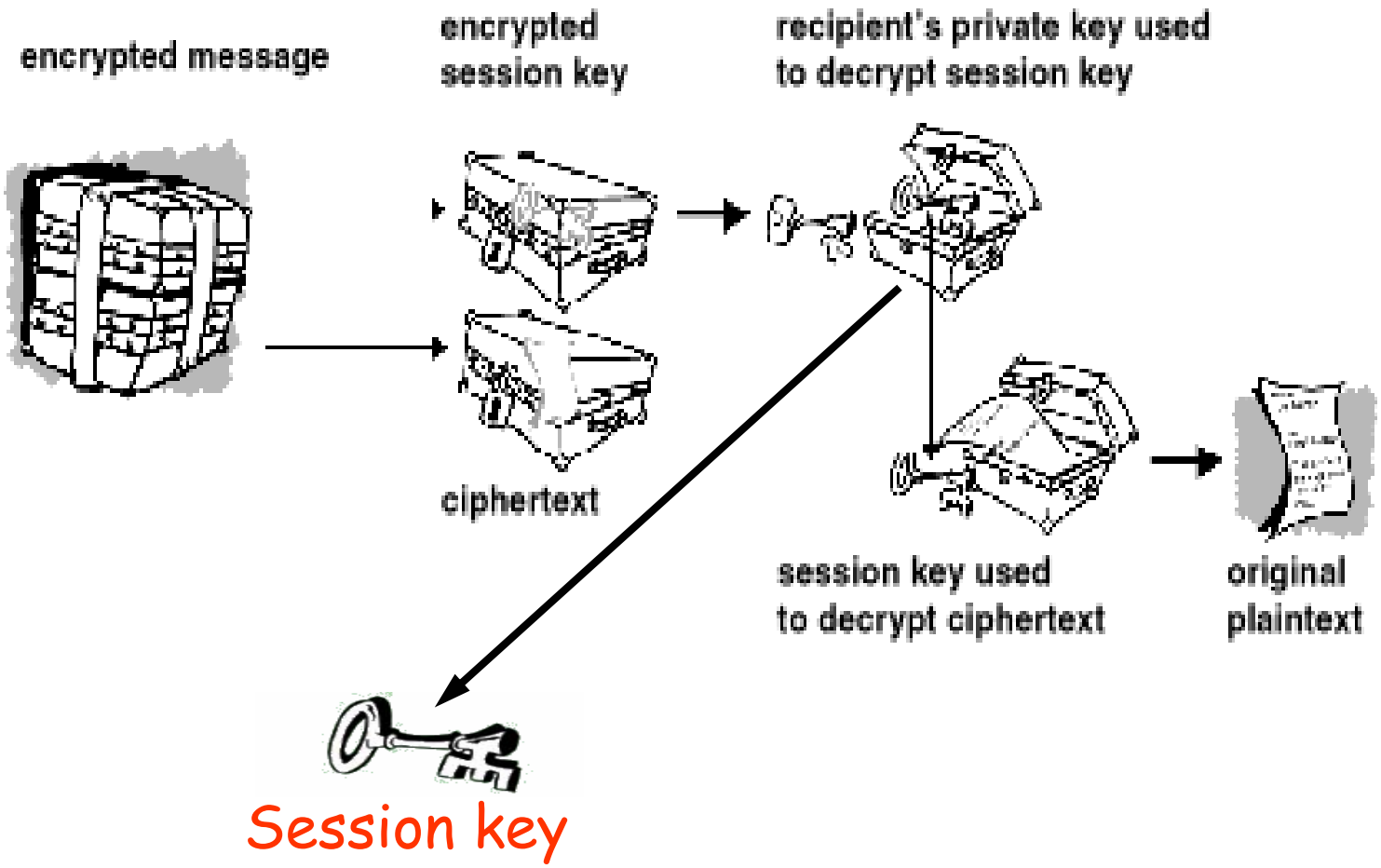
- Il messaggio è cifrato con un cifrario simmetrico
  - Algoritmi: AES, CAST, IDEA, 3DES, Twofish
- La chiave (session key) è generata dal mittente per ogni messaggio
- La session key è poi cifrata con la chiave pubblica del destinatario
  - Algoritmi usati: DH, RSA

Confidenzialità

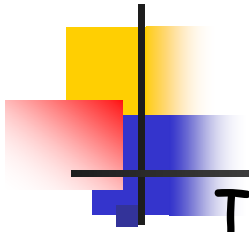


Session key

Decifratura

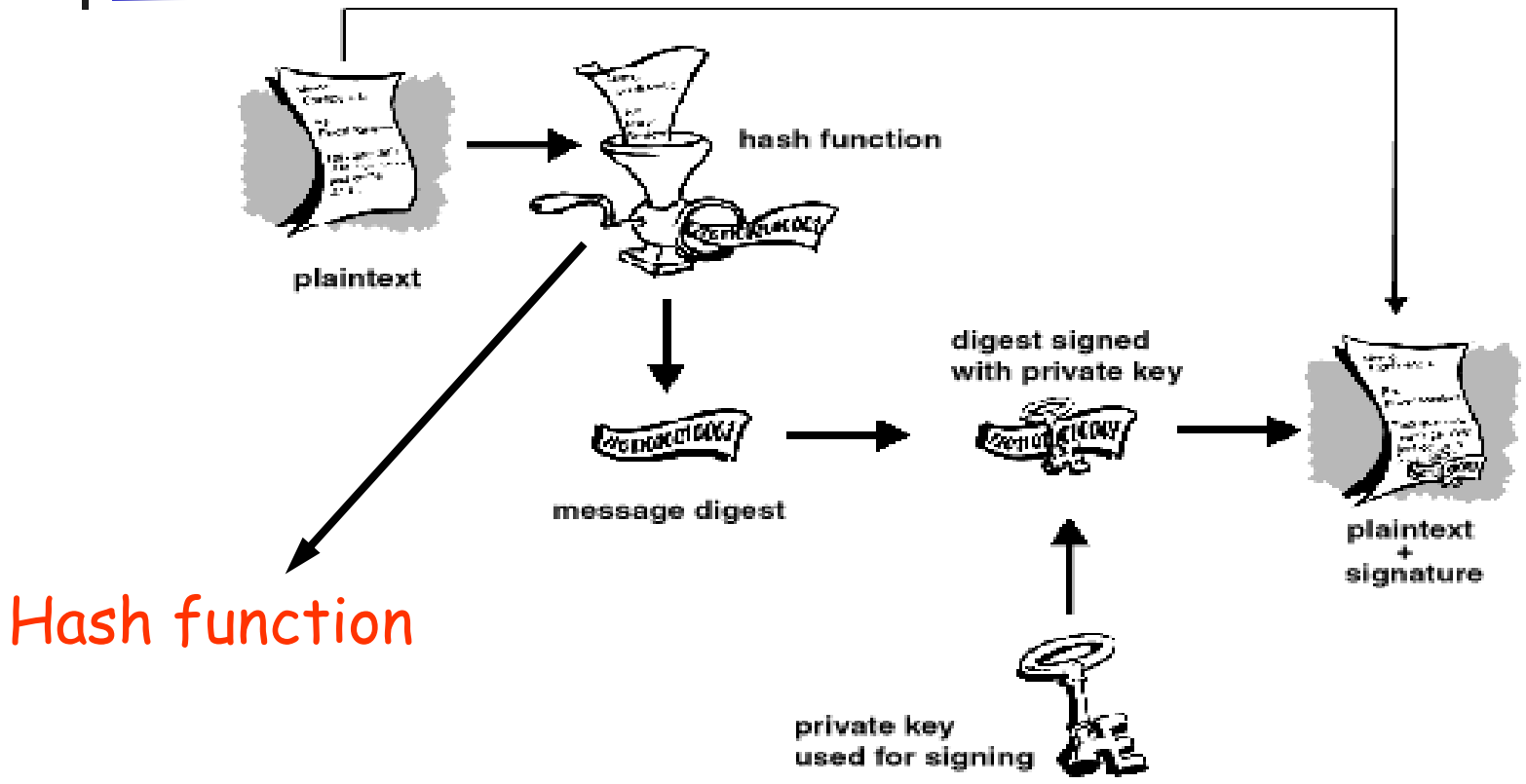


Autenticazione

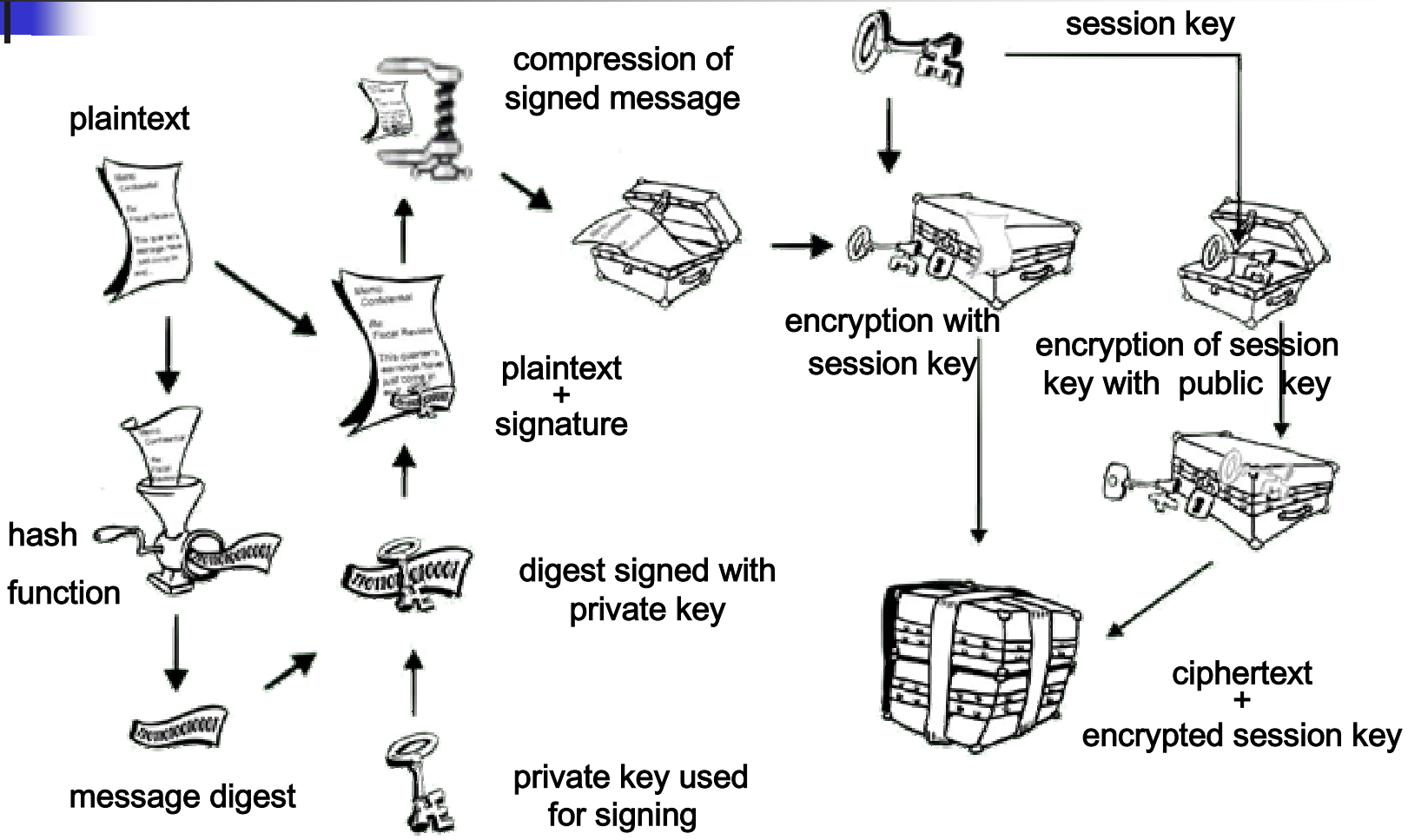


- Il messaggio viene compresso con una funzione hash e fornisce un message digest di lunghezza costante
  - Algoritmi usati: MD5 (128 bit) o SHA (160 bit)
- Il message digest viene firmato con la chiave privata del mittente
  - Algoritmi usati: RSA o DSA

Autenticazione



Confidenzialità e autenticità



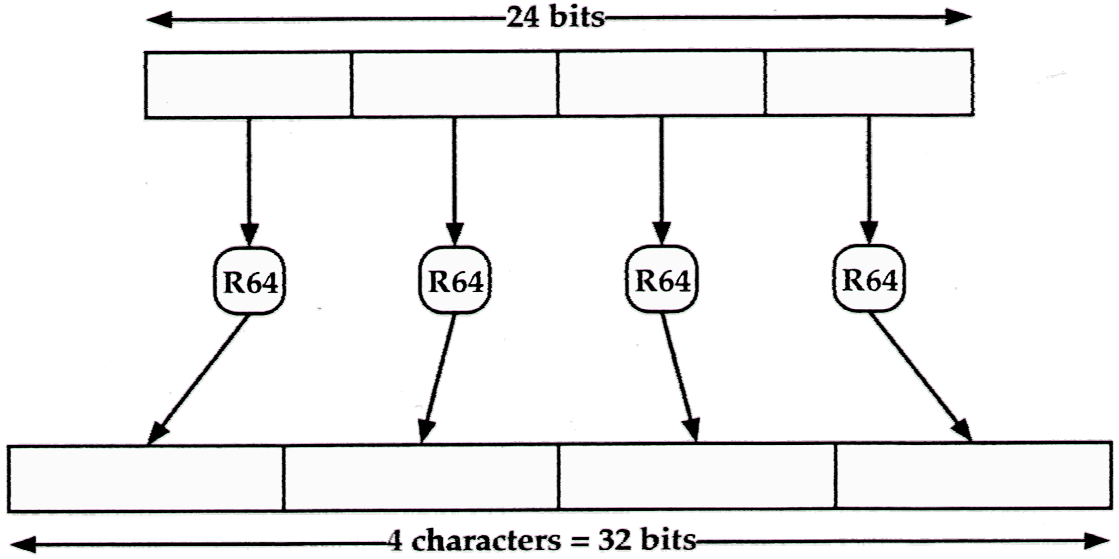
Compressione

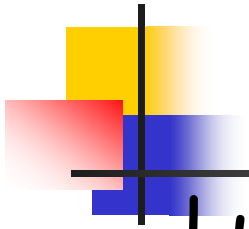
- PGP comprime il testo in chiaro prima di cifrarlo
  - Risparmio di tempo nella trasmissione
  - Aumenta la sicurezza della cifratura
  - Crittoanalisi più difficile
- La compressione è eseguita dopo la firma
  - Ciò evita la ricompressione in fase di verifica
  - Stesso algoritmo di compressione per tutte le implementazioni del PGP

Compatibilità e-mail

- PGP converte i messaggi cifrati in ASCII usando la conversione Radix-64 per fornire trasparenza alle applicazioni e-mail
- Converte il testo espandendo gruppi di 3 byte in 4 caratteri ASCII

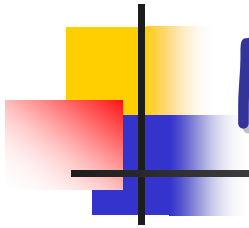
espansione del 33%





Segmentazione

- Lunghezza massima di messaggi via e-mail: 65000 byte
- RADIX-64 divide il messaggio in segmenti più piccoli che vengono inviati separatamente
- Il software del destinatario concatena i segmenti nel giusto ordine



PGP: un esempio

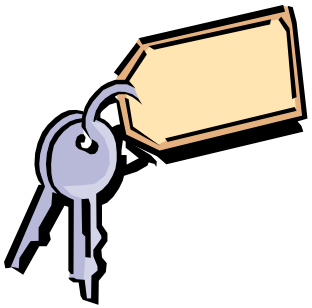
-----BEGIN PGP MESSAGE-----

Version: PGPfreeware 6.5.3 for non-commercial use

/K8xfzpEDp19J3tkItAjbBJstoXp18mAkKjX4t7eRdefXUkk+/lPpclJc8
zgH/1mBkQhVlwheylkGjqflOqe4/I6t7QZNtArJNVP/seDiYRFtC+duJ
me+pGMw68sn8myKFXGktDlR0FNjiGk4JJynoQHo21+mUQxaC2
TiknYlaV+zElHamuSfQu4y57oggFOSAogmgar66psSoaMJFVEafU
Wd7lbnDvnkyHPZri7JpXblFFVxpqbQRhvWkQVMy5JqoFCEuft080
xYbdFAW2ymzgkbpo2BalCEPLiRIQfE6y4L41nVhzIxnMKdhxmJ4V
GqVhKqakVnd8UhsPv515QfWoZAFuHa4YbL6bdXi3ZSY1XqtmnW
RaWnVupK/Z6PK+4gCrO9m0JQ6JnRRY+eY1+uygxm64XHTaCSf
hW2rac2BhsfJGqWdu8AlClpITeMFsY3wyzgJjbuenN9CtwiJAEYEG
BECAAYFAjcXkTcACgkQIuhy1aq5NlaiIQCcCquPR1HfI5QvHjA67
mBvqCXTVuoAnAoTl/gqVOy2VezwHWO4pH3eIJ5/=8kKG

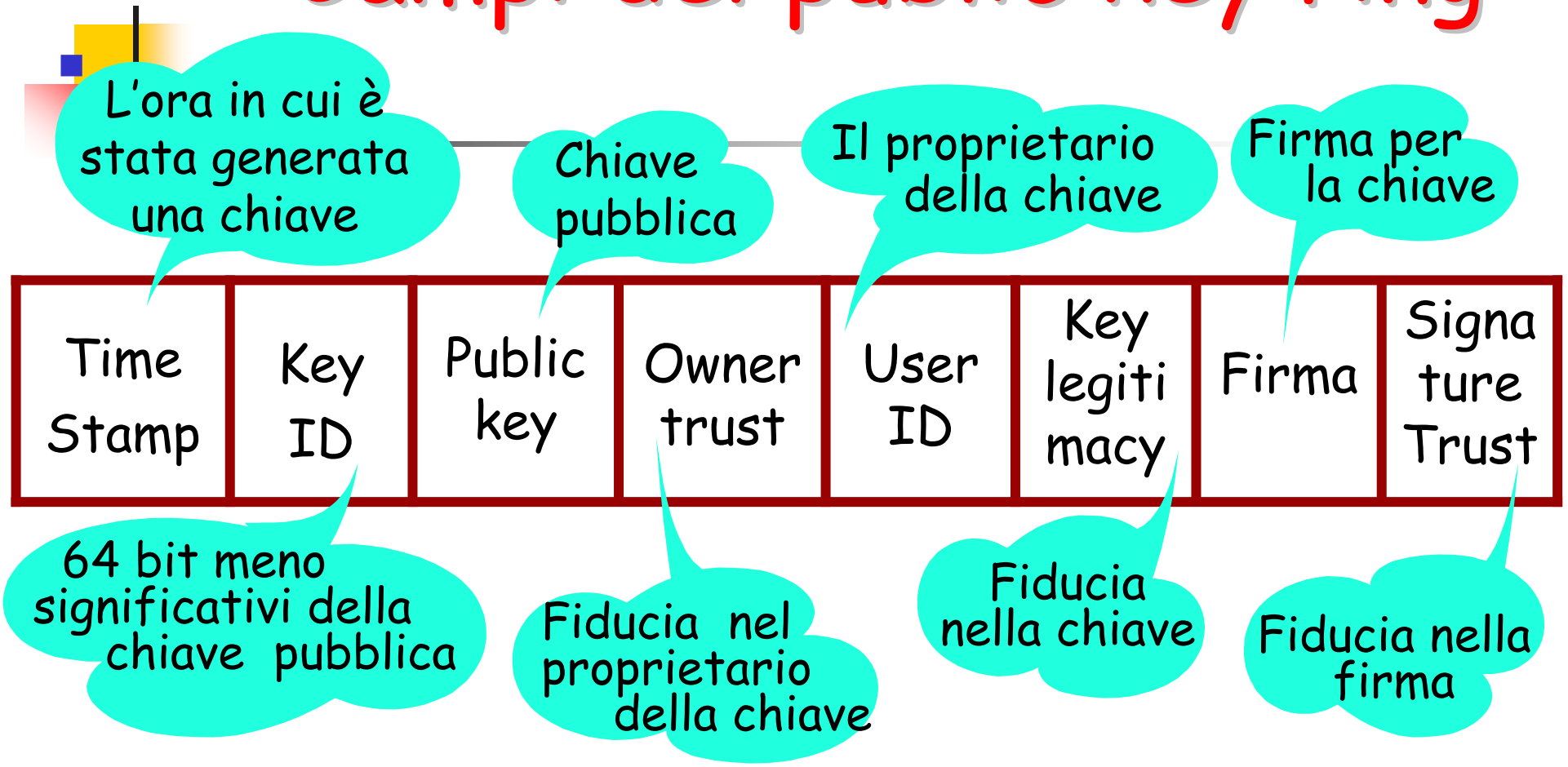
-----END PGP MESSAGE-----

Key ring

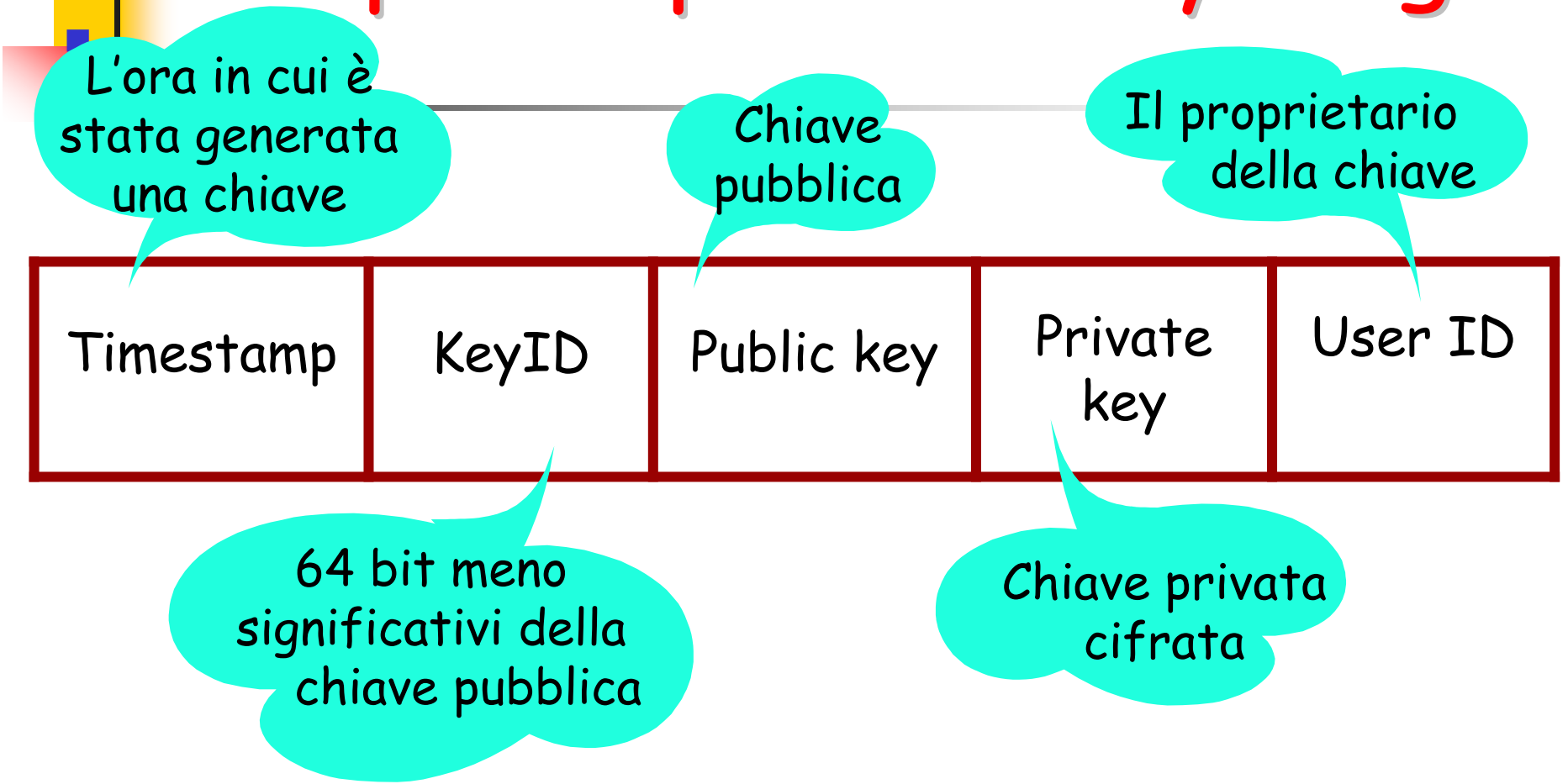


- Le chiavi sono memorizzate nei keyring
- Private Key Ring (secring.skr)
  - Memorizza la coppia di chiavi, pubblica e privata, dell'utente
  - La chiave privata è cifrata con la passphrase
- Public Key Ring (pubring.pkr)
  - Usato per memorizzare le chiavi pubbliche delle persone note all'utente.

Campi del public key ring

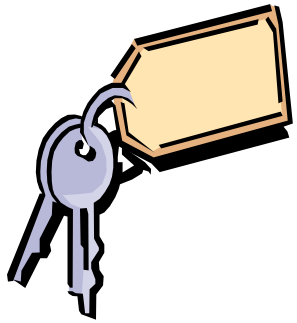


Campi del private key ring



Distribuzione delle chiavi

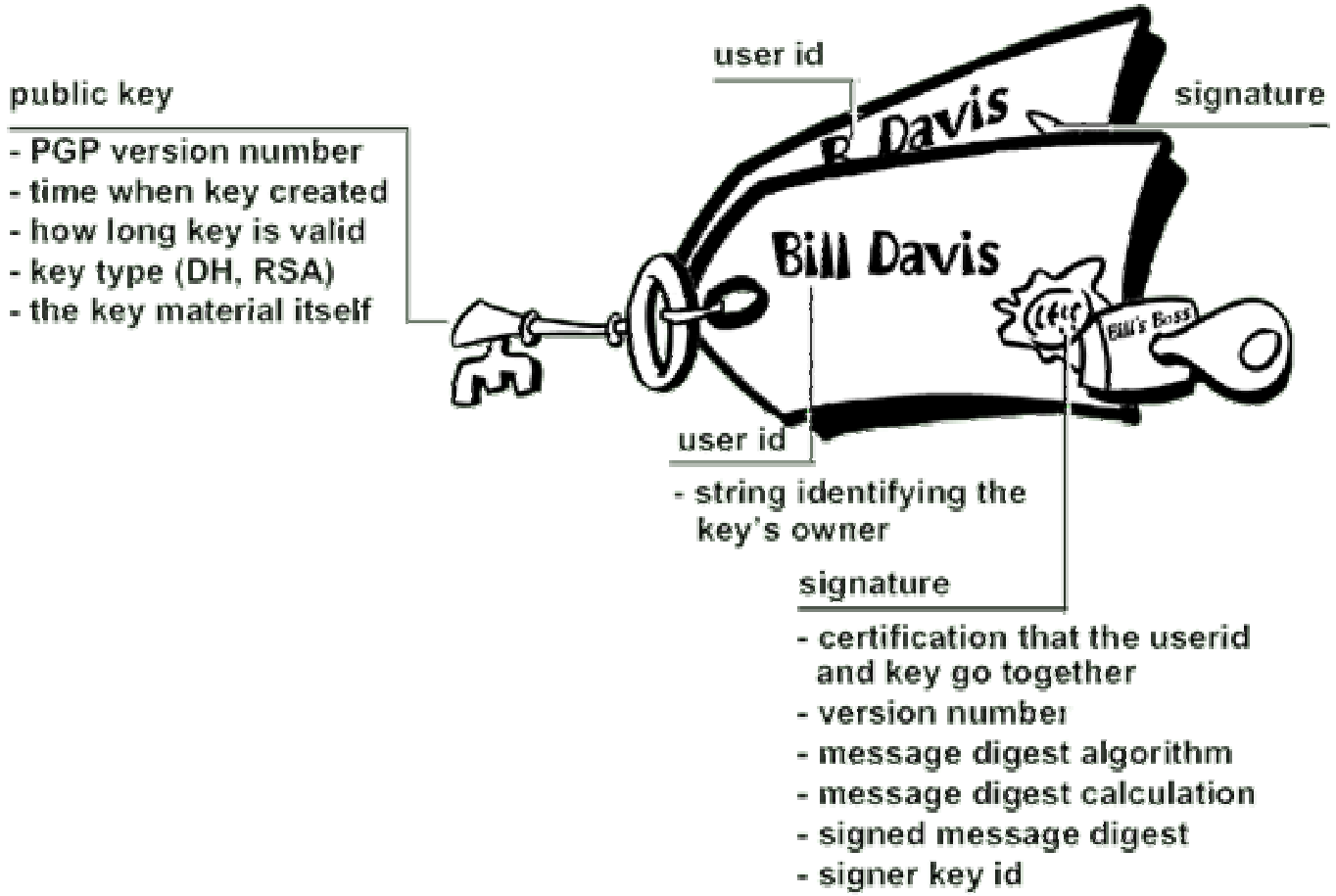
- E' necessario distribuire la nostra chiave pubblica
  - per consentire agli altri utenti di mandarci messaggi cifrati
  - per verificare la nostra firma
- Come ottenere le chiavi pubbliche ?
  - Dal proprietario
  - Da qualcuno che l'ha ottenuta in precedenza
  - Da un keyserver



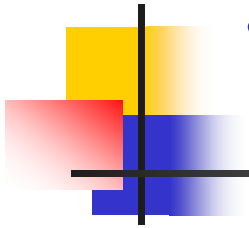
Certificati Digitali

- PGP riconosce due formati differenti:
  - certificati PGP
    - Ognuno puo' creare un certificato PGP
    - Un certificato PGP puo' essere firmato da vari utenti e revocato solo dal proprietario
  - certificati X.509
    - Solo una Certification Authority puo' creare un certificato X.509
    - Un certificato X.509 puo' essere firmato o revocato solo dalla CA

Certificati PGP

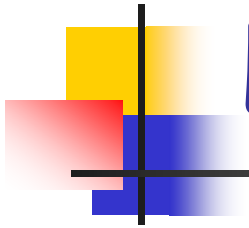


A PGP certificate



Web of Trust

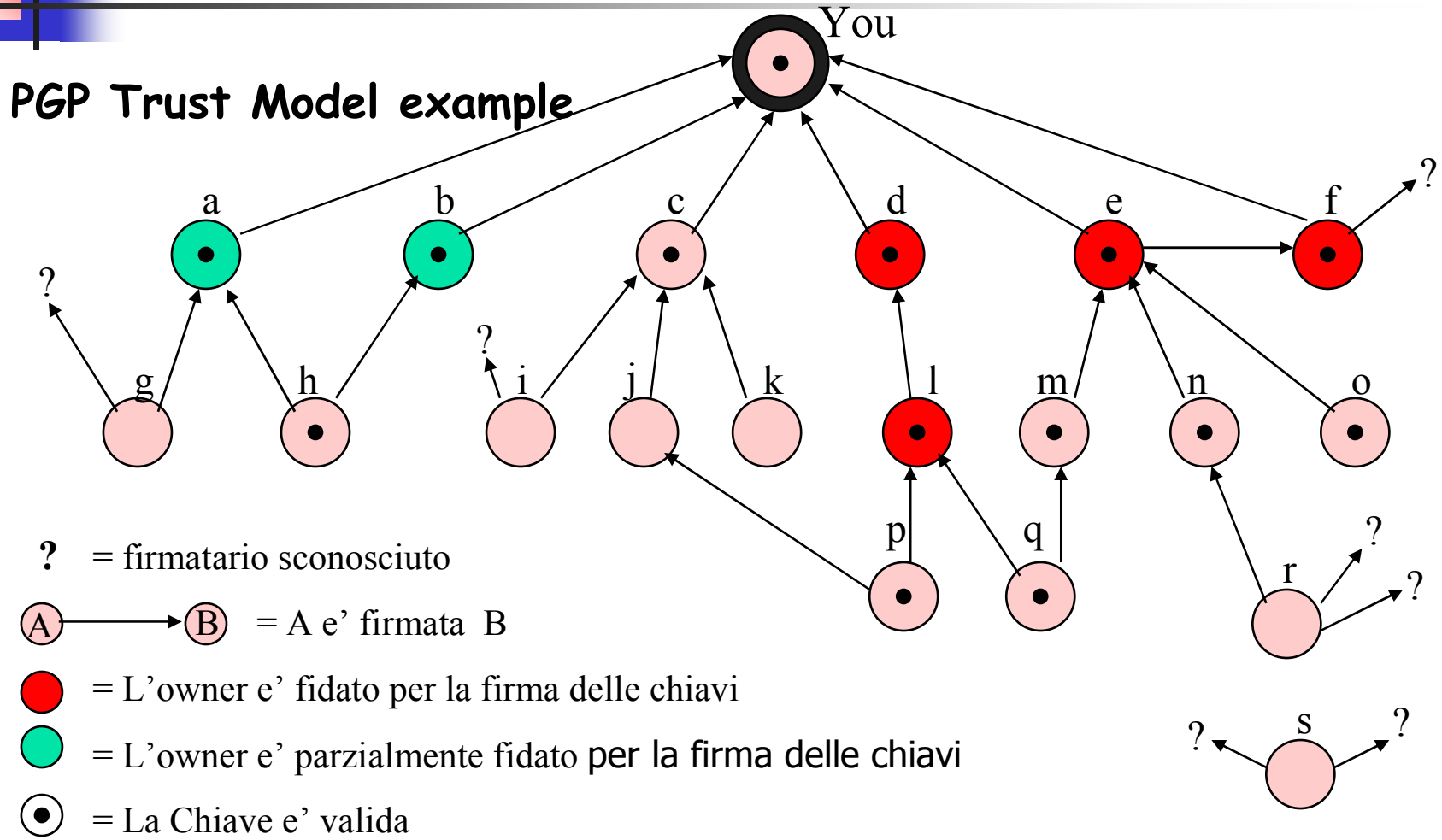
- PGP si basa su una gestione decentralizzata, in cui ciascuno si rende responsabile certificando una firma o un'altra
- Non bisognerebbe mai certificare una chiave di cui no si è perfettamente sicuri perché se fosse fasulla si comprometterebbe il "web of trust" che PGP mira a creare



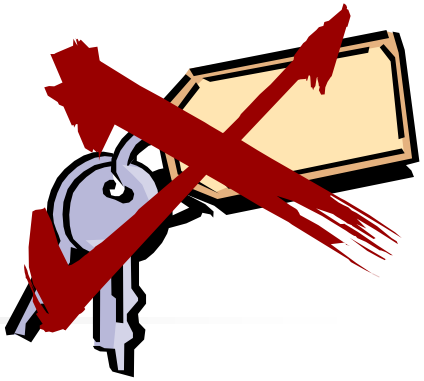
Livelli di Trust

- Ci sono quattro livelli di fiducia che si possono assegnare alla chiave pubblica in un key ring
  - Completamente fidato
  - Parzialmente fidato
  - Non fidato
  - Non noto
- Tutte le chiavi firmate con la propria chiave sono valide

Web of trust

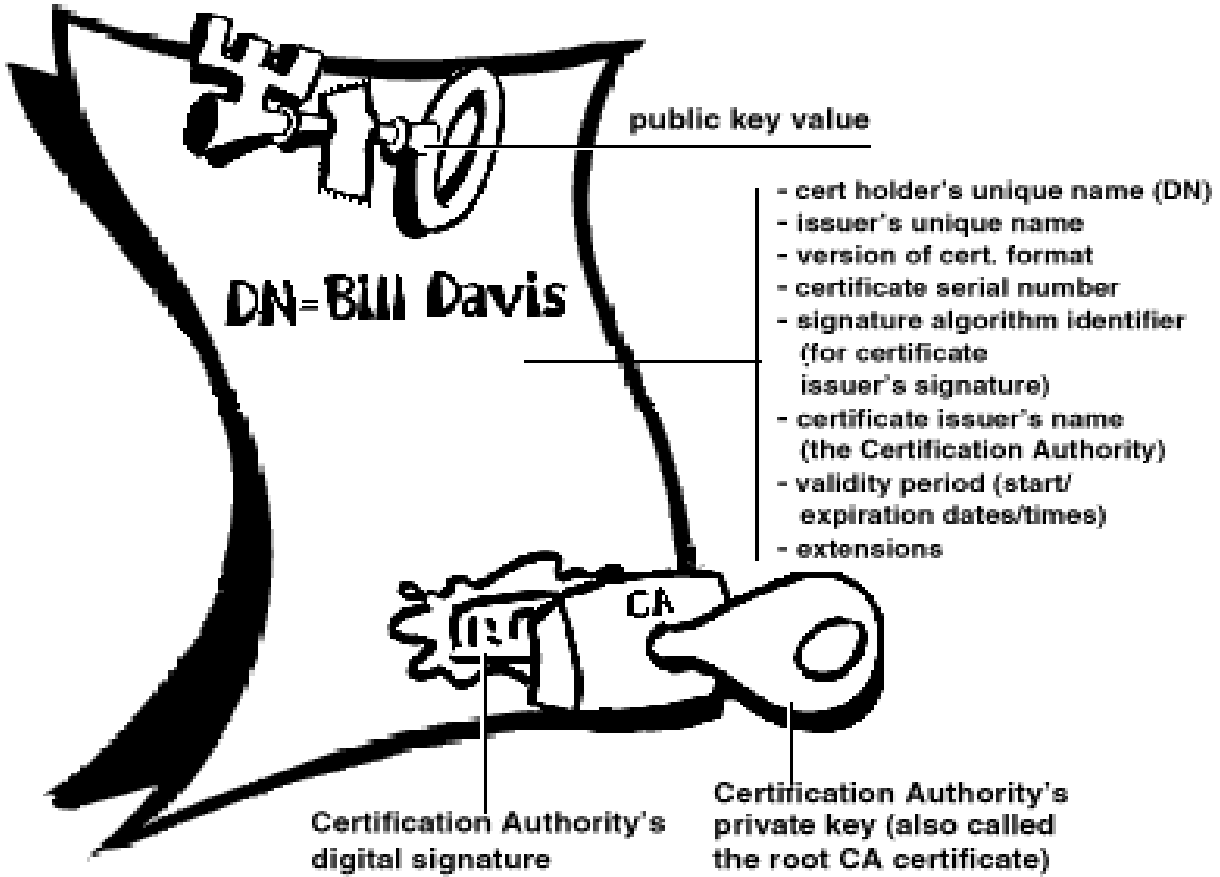

PGP Trust Model example
You
a
b
c
d
e
f
?
g
h
i
j
k
l
m
n
o
p
q
r
s
? = firmatario sconosciuto
A → B = A e' firmata B
= L'owner e' fidato per la firma delle chiavi
= L'owner e' parzialmente fidato per la firma delle chiavi
= La Chiave e' valida

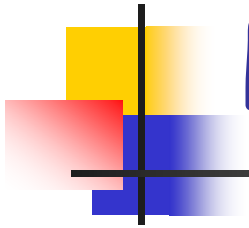
Certificati di Revoca



- Un certificato di revoca avvisa che una chiave pubblica non è più valida.
- Questo certificato è firmato con la chiave privata corrispondente a quella che si vuole revocare.
- PGP installerà il certificato nel keyring di chiunque lo riceva impedendo l'uso della chiave compromessa.

Certificati X.509

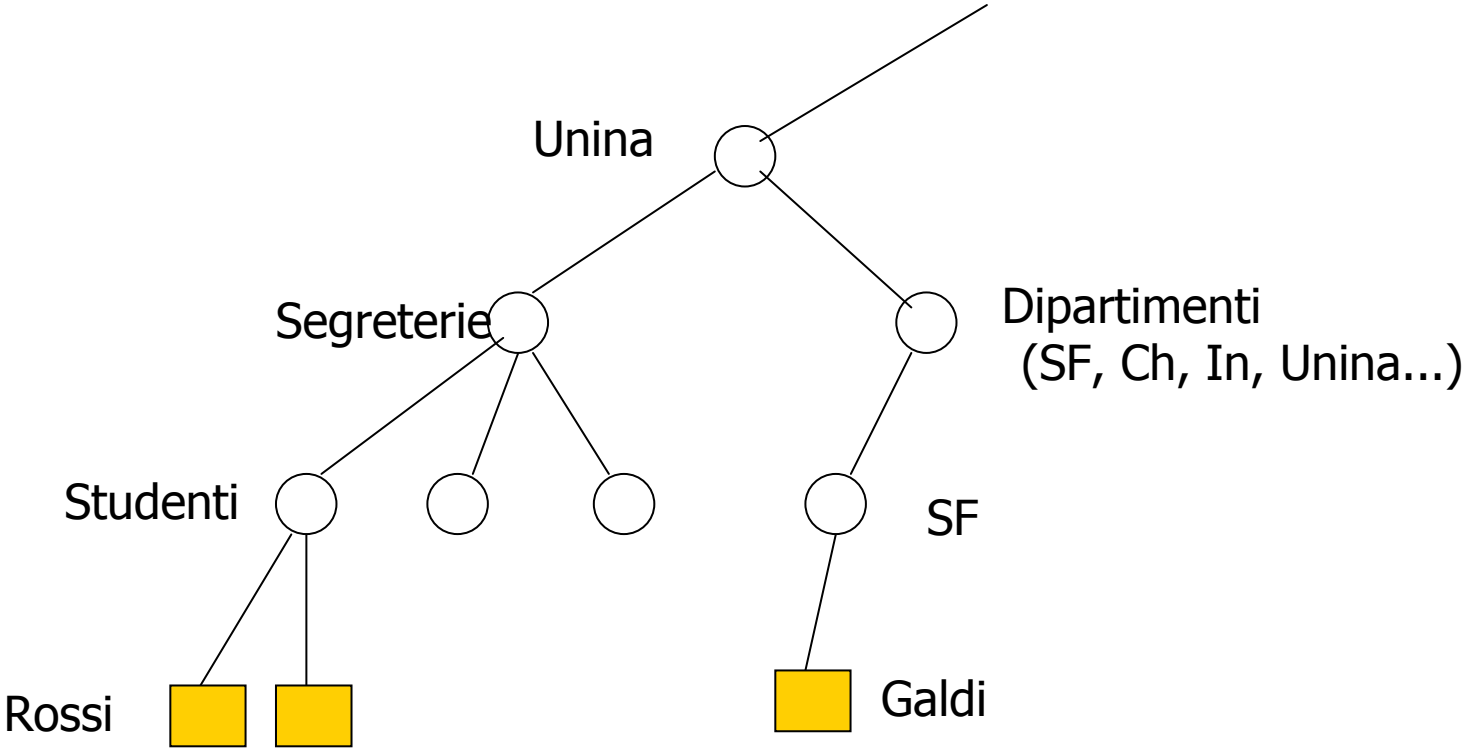


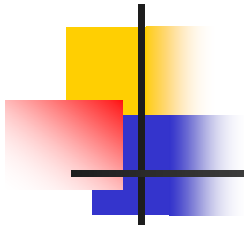


PKI

- I certificati X.509 vengono rilasciati da *Certification Authorities (CA)*
- La CA
  - Certifica l'identita' del proprietario di un certificato
  - Firma tutti i certificati emessi consentendo di verificarne l'originalita'
- Le CA sono organizzate in modo gerarchicho

PKI: Un esempio





Revoca certificati

- Se un certificato viene compromesso il proprietario chide la revoca alla *CA*
- La *CA* inserisce i certificati validi nella *Certificate Revocation List (CRL)*