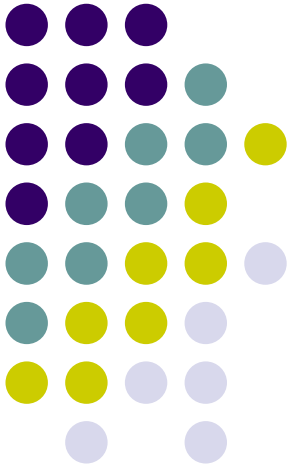
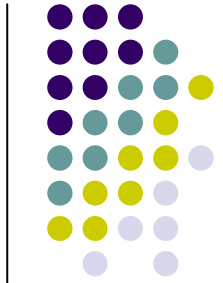
Autenticazione

Esempi e Approfondimenti

Fabrizio Martorelli
Daniele Giannetti
Fabio Cagnizzi
Federico De Felici



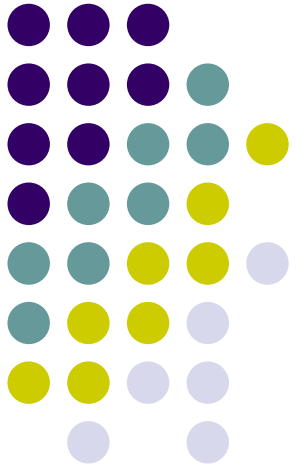
Sommario



- Autenticazione del Messaggio
  - Introduzione
  - MAC (Message Authentication Codes)
  - SSL (Secure socket layer)
- Autenticazione debole (funzioni hash)
- Autenticazione forte (Challenge-Response)
- Protocolli di tipo Zero-Knowledge
- Autenticazione client/server: Kerberos

Autenticazione

MAC e SSL



Fabrizio Martorelli

Servizi

Quali sono i servizi di sicurezza che vorremmo garantire sui nostri documenti

Il documento fisico (cartaceo)

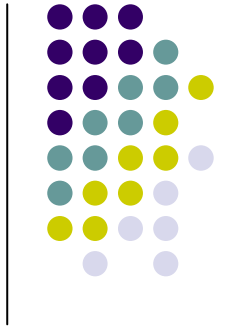
- Ha firma e data
- Talvolta deve essere: autenticato, registrato
- Spesso occorre proteggerlo da: divulgazione, falsificazione, distruzione

Oggi in molti campi societari la gran parte della documentazione trattata è di tipo elettronico

- In Italia l'articolo 15, comma 2, della legge 15 marzo 1997 n°59 ha stabilito che i documenti informatici sono validi e rilevanti a tutti gli effetti di legge

Vorremmo associare al documento elettronico le stesse funzioni che associamo al documento fisico.

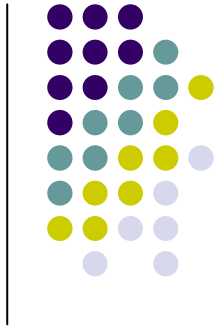
Servizi



Le caratteristiche del documento fisico che rendono difficile il problema sono:

- È abbastanza facile distinguere l'originale del documento da una fotocopia - un documento elettronico è solo una sequenza di bit, non c'è differenza tra originale e copie
- Le alterazioni sul documento fisico lasciano tracce; es. una cancellatura lascia il foglio ruvido o più sottile - alterare i bit non lascia tracce
- Ogni "prova" associata a un documento fisico ha la sua fisicità; es. firma, timbro - come "marcare" i bit?

Servizi



I servizi legati alla sicurezza informatica sono prevalentemente concentrati su:

- confidenzialità: accessibilità dell'informazione solo per coloro che sono autorizzati
- autenticazione: identificazione dell'origine
- integrità: modificabilità solo da parte di coloro che sono autorizzati
- non-ripudiabilità: l'autore del documento non ne può negare la paternità
- disponibilità: il sistema informatico è disponibile per coloro che sono autorizzati

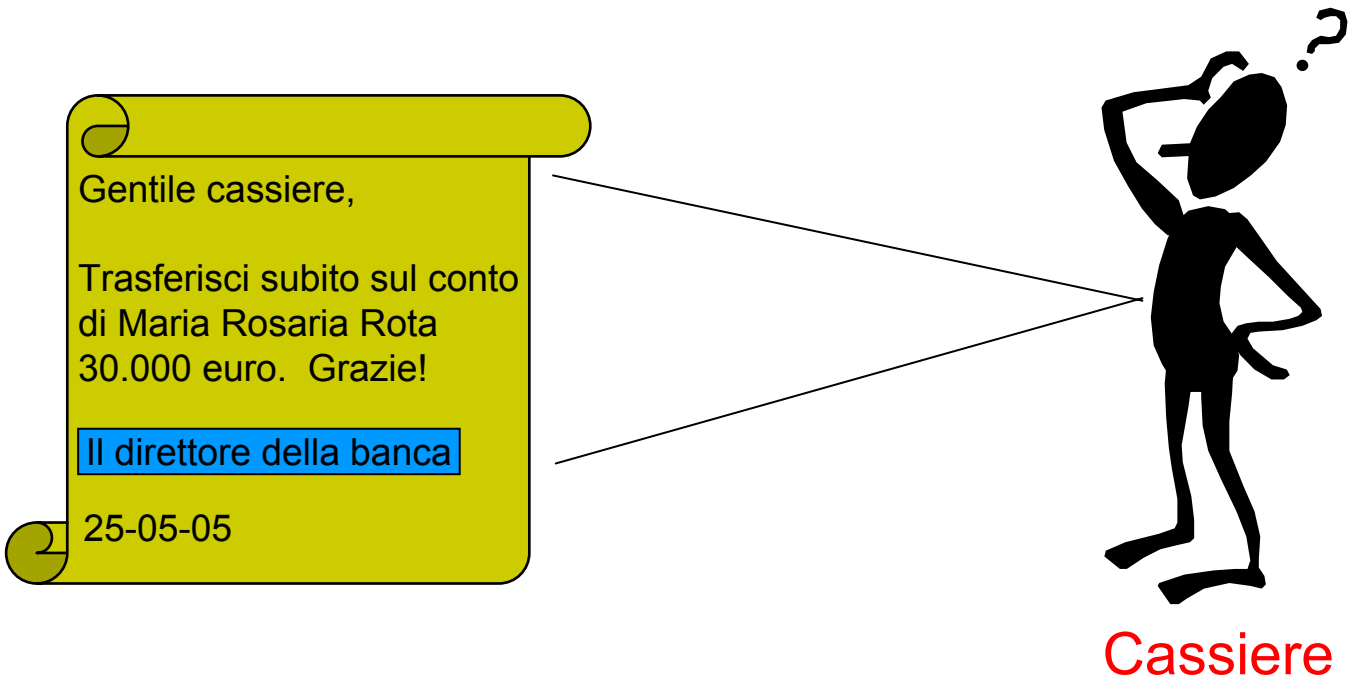
Servizi - confidenzialità

Accessibilità dell'informazione solo per chi è autorizzato

La confidenzialità di questa informazione è evidentemente fondamentale !

Servizi - autenticazione

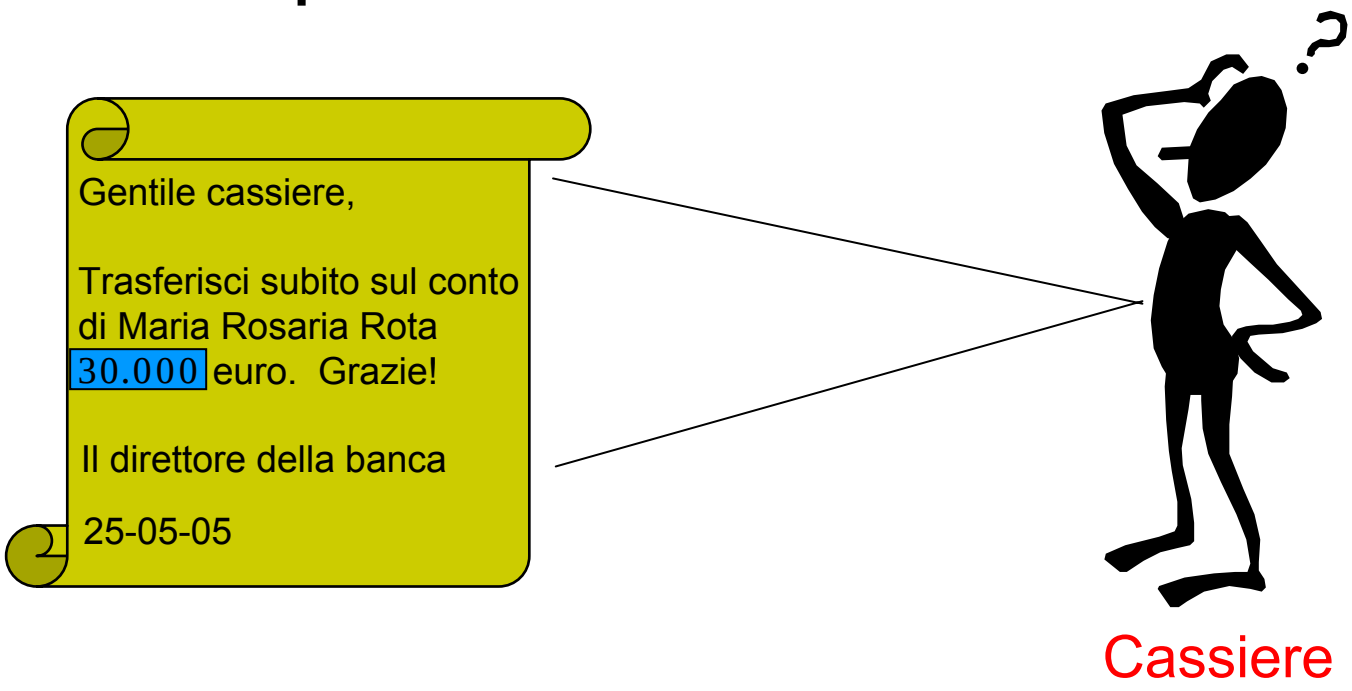
Identificazione dell'origine



Siamo certi che sia stato proprio il direttore della banca ad ordinare al povero cassiere di svolgere la transazione richiesta?

Servizi – integrita’

Modificabilità solo per chi è autorizzato



Siamo certi che sia proprio quella, la cifra che il direttore della banca ha ordinato al povero cassiere di accreditare alla cliente??

Servizi – non-ripudiabilità

L'autore del documento non ne può negare la paternità

A buon ragione il mal capitato azionista cerca di rinnegare la paternità del messaggio, per non incombere in una ingente perdita di denaro

Attacchi – prima differenza

Azioni che hanno l' obiettivo di compromettere la sicurezza delle informazioni possedute, non permettendo cosi di garantire l'integrità dei servizi prima menzionati

Attacchi passivi:

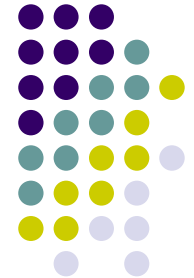
- Sono quelli destinati all'intercettazione dei messaggi con il solo scopo di infrangerne la confidenzialità.

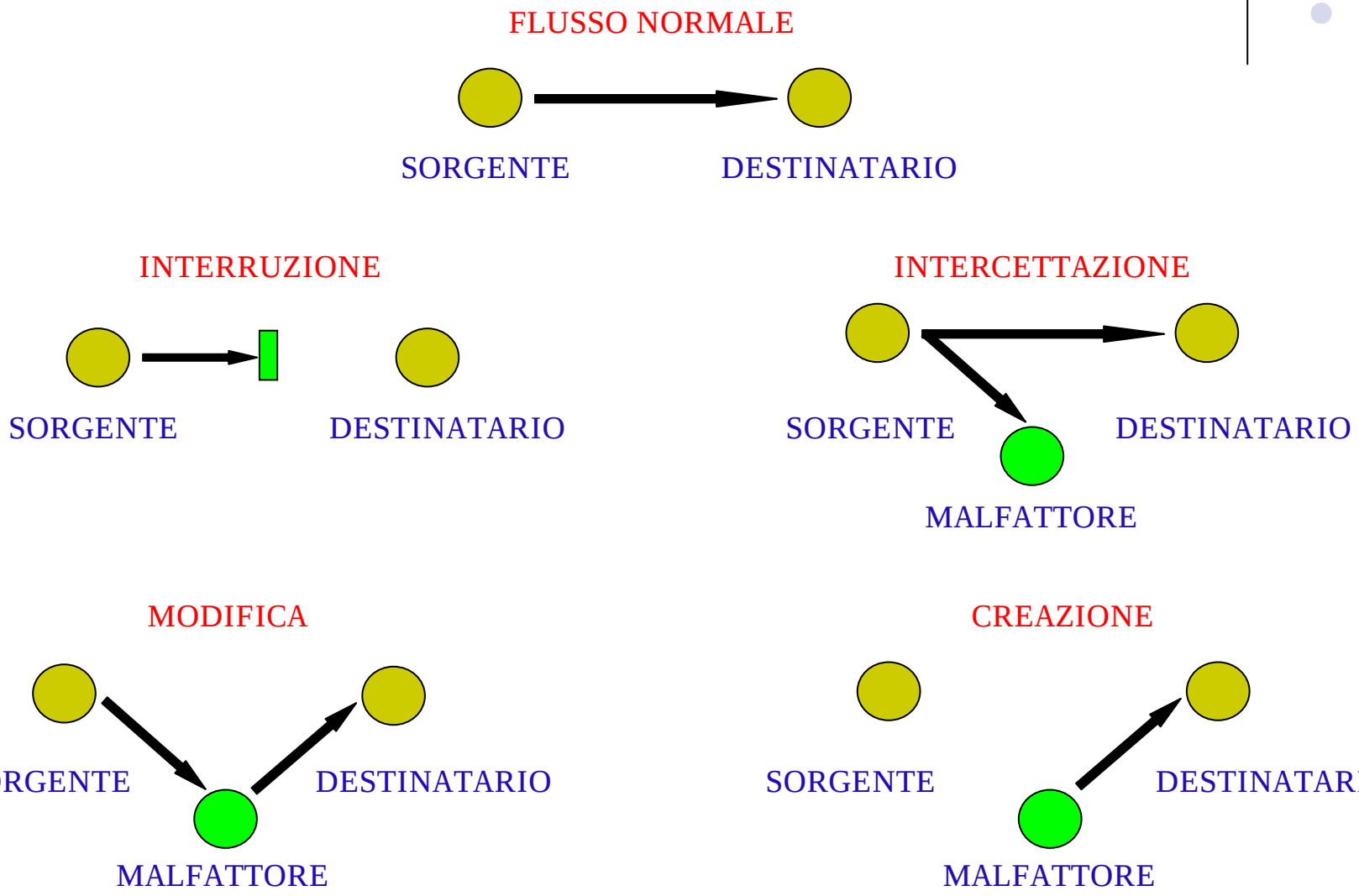
Attacchi attivi:

- Sono quelli destinati non solo all'intercettazione dei messaggi ma soprattutto alla loro modifica, al fine di poter trarre vantaggio (scopo personale) o per causare danni.

In entrambe le tipologie, le conseguenze possono essere molto serie

Attacchi - tipologie





Problema

Le tipologie di attacchi precedentemente menzionati comportano la necessità di dover garantire le seguenti caratteristiche sui messaggi scambiati (sicurezza informatica del messaggio)

Integrità del messaggio:

- Come sapere se il messaggio ricevuto è proprio quello trasmesso senza alcun cambiamento?

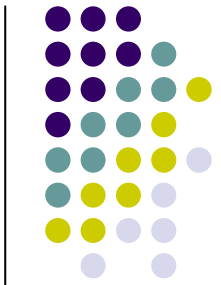
Autenticazione del messaggio:

- Il messaggio è stato mandato veramente dalla persona che il destinatario crede?

Autenticità del mittente:

- Riguarda il problema di una persona che vuole provare la propria identità. Il destinatario vorrebbe essere certo di parlare proprio con la persona che lui pensa.

Soluzione – come ottenere integrità e autenticità

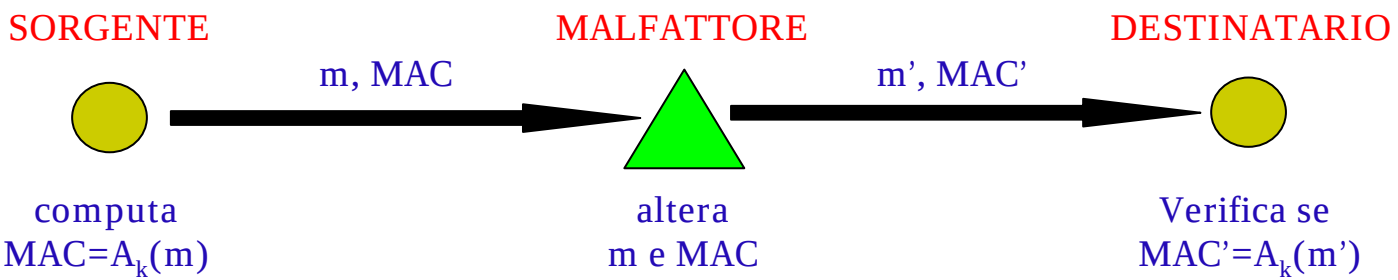


Per garantire le due proprietà sul messaggio il mittente deve fornire un'ulteriore informazione:

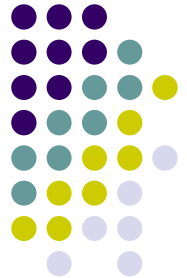
Il "codice di autenticazione di un messaggio" MAC :

- Protocollo basato su chiave segreta simmetrica k e su un algoritmo crittografico A. Il mittente invia oltre al messaggio m, anche il corrispondente MAC calcolato come segue.

MAC = $A_k(m)$ A: algoritmo k: chiave m: messaggio



Mettiamo in evidenza alcuni aspetti



Se $A_k(m') \neq MAC'$

- Il destinatario sa che è successo qualcosa ad m, non ritiene affidabile il messaggio e quindi lo respinge (verifica integrità).

Se $A_k(m') = MAC'$

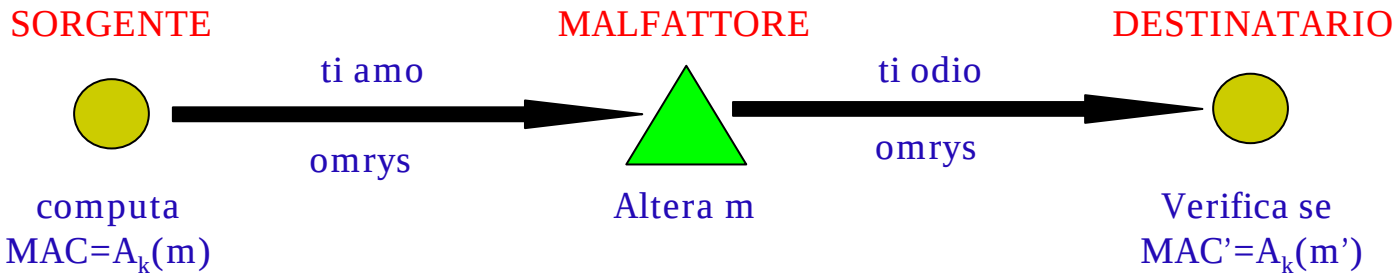
- È ragionevolmente sicuro che il messaggio non è stato cambiato. Questo dipende molto dalla forza dell' algoritmo A e dal numero delle possibili chiavi k.

Altri aspetti

- L'insidia del malfattore è contrastata, egli dovrebbe conoscere il valore della chiave per produrre il corretto MAC' sul suo m'
- Il destinatario può solo capire se il messaggio è stato contraffatto poiché non può da questo, risalire al messaggio originale (ritrasmissione messaggio).
- Il MAC garantisce anche l'autenticità di m in quanto il mittente è la sola persona che conosce la chiave k
- Non è importante la segretezza dell'algoritmo A utilizzato.

Un esempio concreto

A = cifrario di Vigenère
K = verme
m = ti amo MAC = omrys
m' = ti odio MAC' = omfpmj

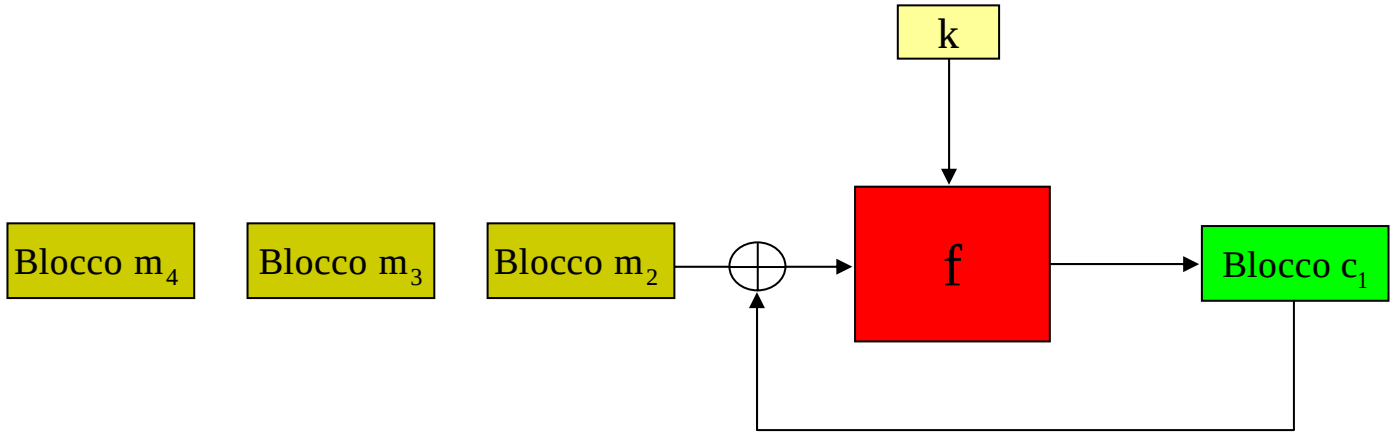


Il destinatario si accorge che omrys ≠ omfpmj quindi il messaggio è stato contraffatto.

Problema:

- Il metodo proposto ha lo svantaggio di richiedere che il messaggio sia trasmesso due volte, in chiaro e cifrato.

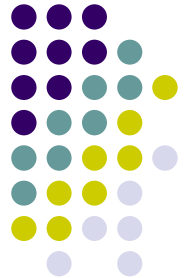
Soluzione – chipher block chaining mode



Il metodo per concatenare i blocchi cifrati

- Il messaggio m viene diviso in blocchi di lunghezza n fissata tipicamente n=64bits.
- *f*, algoritmo cifrante con chiave fissata *k*, trasforma un blocco *m* di *n* simboli, in un blocco *c* cifrato con *n* simboli.
- Iterativamente si applica $c_s = f_k (c_{s-1} \oplus m_s)$, con s≠1.
- Il MAC da utilizzare è dato dall'ultimo blocco così cifrato.

Vantaggi – chipher block chaining mode



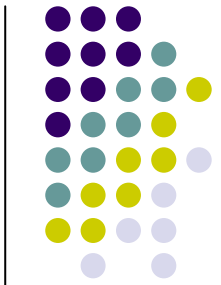
Il MAC

- Ha una lunghezza fissata ad n risultando cosi disaccoppiato dal messaggio m da inviare (guadagno in trasmissione) .
- Il MAC dipende da tutti i blocchi del messaggio e non solo da una sua parte.

Sicurezza

- Deve essere impossibile trovare un messaggio m partendo dal suo MAC. Se vale questa proprietà l'algoritmo A (o la funzione f) è detto *one-way hash function*.
- Deve essere praticamente impossibile trovare due messaggi distinti m ed m' aventi lo stesso MAC. Se vale questa proprietà l'algoritmo A (o la funzione f) è detto *libero da collisioni*.
- Algoritmi che soddisfano i punti di cui sopra sono il *triple-DES* e *AES*.

SSL – secure socket layer e MAC



SSL è un protocollo aperto e non proprietario

- È stato proposto da Netscape Communications al W3 Consortium come un possibile futuro approccio standard alla sicurezza per i browsers WWW e per i servers.

Garantisce

- *Privatezza del collegamento*, la crittografia è usata dopo un handshake iniziale per definire una chiave segreta. Per crittografare i dati è usata la crittografia simmetrica.
- *Autenticazione*, l'identità nelle connessioni può essere autenticata usando la crittografia a chiave pubblica, è prevista la certificazione sia del server che del client.
- *Affidabilità*, il livello di trasporto include un check dell'integrità del messaggio basato su un apposito MAC che utilizza funzioni hash sicure (MD5). In tal modo si verifica che i dati spediti tra client e server non siano stati alterati durante la trasmissione.

SSL- sessione e connessioni

Sessione

- Associazione logica tra client e server.
- Viene creata tramite il protocollo di handshake e definisce un insieme di parametri crittografici che possono essere condivisi da varie *connessioni.*
- La sessione evita la costosa rinegoziazione dei parametri di sicurezza per ciascuna connessione.

Protocollo di handshake

- Definisce chiavi condivise usate per cifrare i dati garantendo cosi la confidenzialità delle informazioni.
- Definisce anche chiavi usate per comporre i MAC garantendo cosi l'integrità dei dati.

SSL – stato di una sessione e connessione

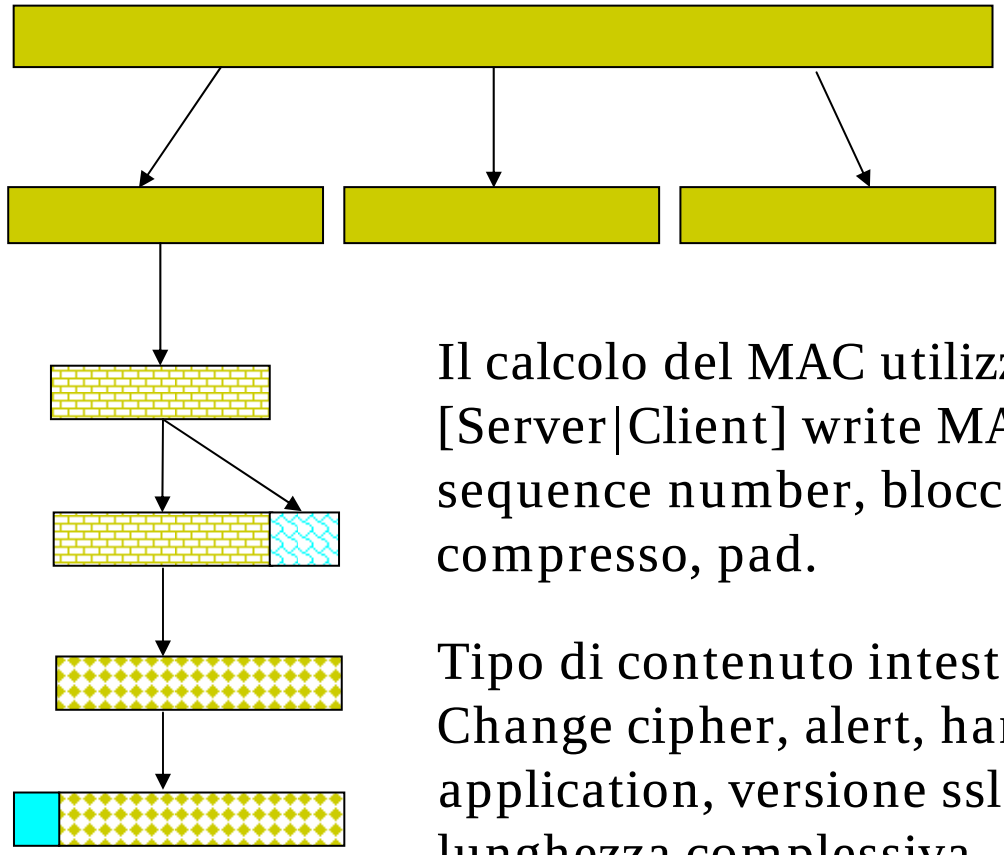
Valori condivisi da ciascun peer della sessione

- *Identificatore di sessione* (session identifier).
- *Certificato* (X.509v3) del peer (può non essere specificato).
- *Metodo di compressione* dei dati.
- *Algoritmo* di codifica ed algoritmo hash scelti (cipher spec)
- *Segreto principale* (master secret) [48 byte] condiviso tra i peer, usato, assieme a numeri random, per generare le varie chiavi necessarie.

Stato di una connessione

- Server/client *random number*
- Server/client *write MAC secret* (Message Authentication Code: "firmadigitale")
- Server/client *write key* (chiave convenzionale)
- *Sequence number* (per tenere traccia dei messaggi spediti e di quelli ricevuti)

SSL- il protocollo record



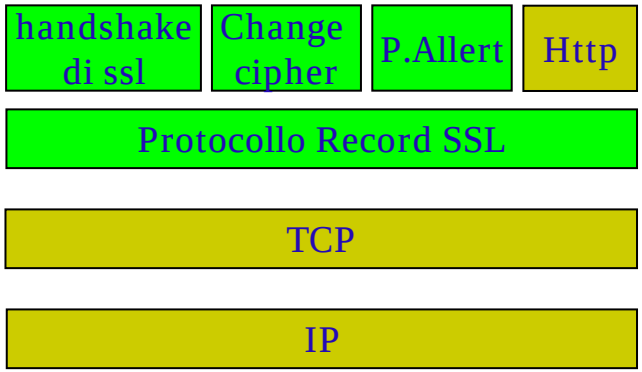
Il calcolo del MAC utilizza: [Server|Client] write MAC secret, sequence number, blocco compresso, pad.

Tipo di contenuto intestazione: Change cipher, alert, handshake, application, versione ssl, lunghezza complessiva

La cifratura utilizza la [Server|Client] write key, può essere a blocchi o a caratteri - crittografia convenzionale.

SSL- architettura

Posizione di ssl nella pila tcp/ip



Oss. Solo uno per volta tra i protocolli applicativi possono viaggiare nei pacchetti tcp/ip.
http + ssl = https

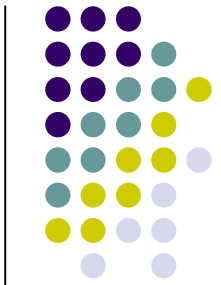
Protocollo change cipher spec

- Consiste in un solo messaggio
- Un singolo byte dal valore 1
- Viene reso operativo l'insieme di protocolli di cifratura negoziati

Protocollo Allert

- Usato per comunicare al peer i messaggi d'allarme di ssl
- bad_certificate , handshake_failure, certificate_expired, unsupported_certificate, decompression_failure, bad_record mac

SSL- fasi dell'handshake



fase 1: establish security capabilities

- Client spedisce un client_hello con :versione ssl, un numero random da 32 byte (28 random +timestamp"), un identificatore di sessione, una cipher suite (algoritmi di cifratura da usare, ordine di prefer.) metodo di compressione
- Il server spedisce un server_hello con parametri analoghi.
- I numeri random scelti da client e server sono diversi.

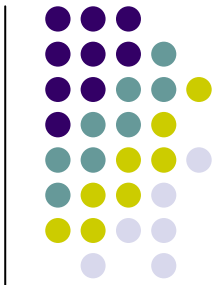
fase 2: server authentication and key exchange

- Il server spedisce il suo certificato
- Certificate request al client (opzionali)

fase 3: establish security capabilities

- Fase di client authentication opzionale, analoga alla precedente
- Il client sceglie un numero random (pre master secret) e lo spedisce al server cifrandolo con la chiave pubblica del server
- Il server scopre il numero random scelto dal client decifrandolo con la sua chiave privata

SSL- fasi dell'handshake



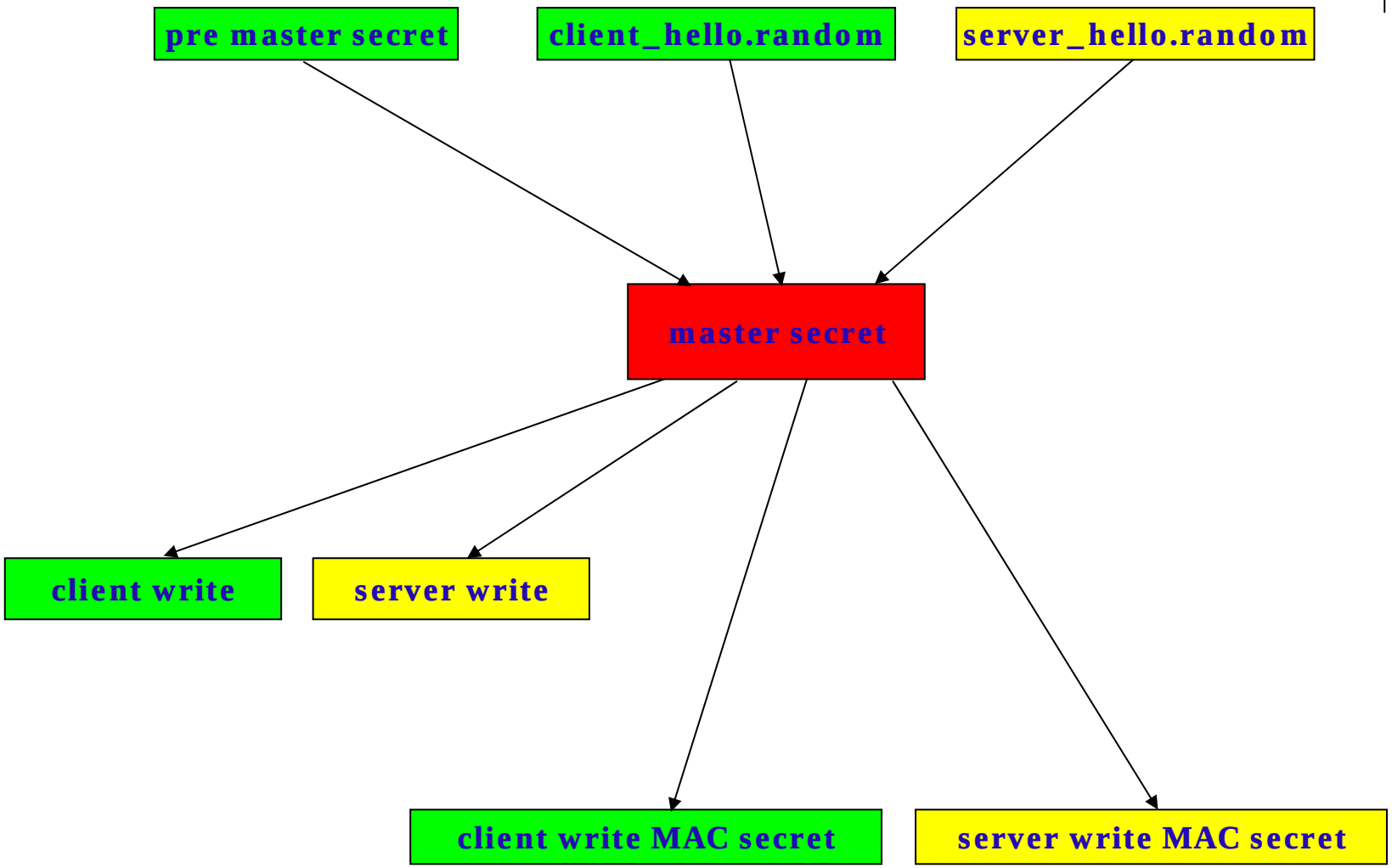
fase 4: finish

- Il client spedisce il change cipher specification rendendo operativa la suite di cifratura

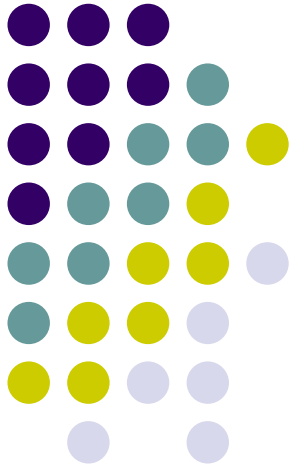
Quindi il protocollo handshake…….

- Permette di stabilire una sessione e cioè permette al client ed al server di autenticarsi a vicenda e di negoziare una suite di cifratura (cipher suite)
- Costituisce un metodo per lo scambio delle chiavi
- Algoritmo di cifratura (utilizzato nel Protocollo Record)
- Algoritmo per il MAC (utilizzato nel Protocollo Record)
- Viene eseguito prima di inviare qualunque dato applicativo

SSL- gerarchia delle chiavi

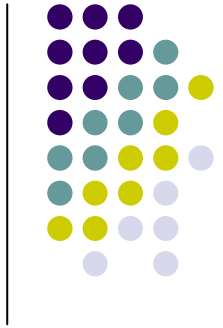


Funzioni Hash



Daniele Giannetti

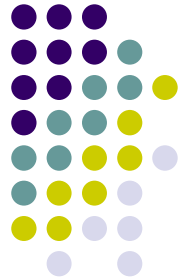
Login



- Nei sistemi UNIX-Like, non si memorizzano le password ma il risultato dell'applicazione su di esse di funzioni hash

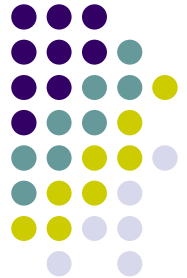
- il sistema **non ha bisogno di conoscere la password**: quando l'utente la fornisce ne calcola l'hash e lo confronta con quello memorizzato

Funzioni Hash



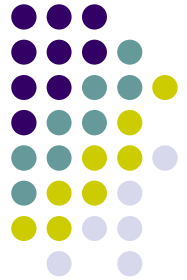
- Hash, nella sua accezione più comune, si riferisce ad una funzione univoca operante in un solo senso (ossia che non può essere invertita) atta alla trasformazione di un testo di lunghezza arbitraria in una stringa di lunghezza relativamente limitata.

Utilizzi



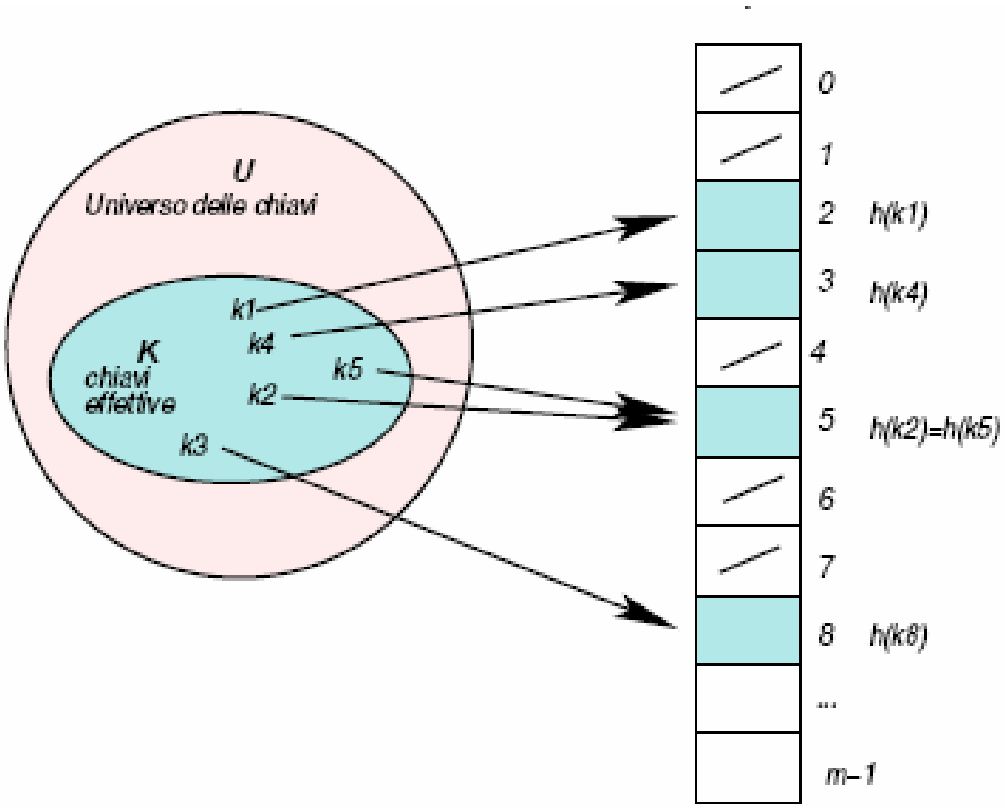
- Nelle strutture dati
- Consideriamo una tavola per la memorizzazione dei dati relativi agli studenti di una facoltà. Supponiamo che gli studenti siano al massimo 3000 e che la chiave sia il cognome degli studenti (non più di 15 caratteri)
- Il numero delle chiavi possibili sarebbe 26^{15} quindi sarebbe necessario un array di:
- 1.67725934228573e + 21 ~ 2^{70} componenti

Utilizzi

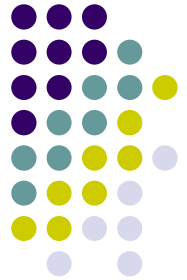


- Una funzione hash definisce una corrispondenza tra l'universo U delle chiavi e gli indici della tabella hash T[0…m-1]
- h:U -> {0, 1, …, m-1}
- Esiste la possibilità che più chiavi corrispondano alla stessa posizione nell'array
- h deve essere scelta in modo tale che le chiavi vengano distribuite in modo uniforme sugli m indici a disposizione

Utilizzi

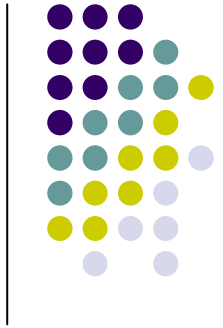

U
Universo delle chiavi
K
chiavi
effettive
k1
k4
k5
k2
k3
0
1
2
h(k1)
3
h(k4)
4
5
h(k2)=h(k5)
6
7
8
h(k8)
...
m–1

Utilizzi



Una buona funzione hash è tale quando per chiavi simili vengono generati indici diversi

Utilizzi



- In crittografia
- Funzioni hash crittografiche sono funzioni hash che sono strongly collision resistant
- Non c'è chiave segreta
- Devono essere veloci da controllare ma deve essere difficile trovare collisioni

Funzioni hash crittografiche

- Vengono utilizzate per ottenere una stringa associata al messaggio da spedire (o al dato da occultare) per il quale una volta applicata la funzione non dovrebbe essere più possibile ritornare al testo originale

Funzioni hash crittografiche

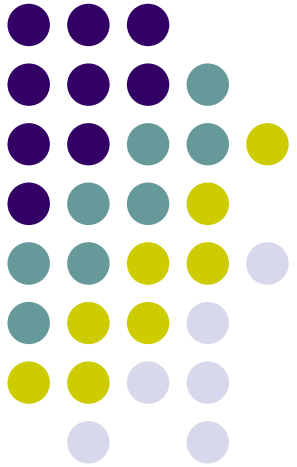
- **MD2 (Message Digest 2)** (Rivest)
  - Hash di 128 bit e richiede come input multipli di 16 byte
- **MD4 (Message Digest 4)** (Rivest 1990)
  - Hash a 128 bit più veloce della funzione precedente, richiede pad a multipli di 512 bit
- **MD5 (Message Digest 5)** (Rivest 1991)
  - Valori hash di 128 bit, messaggi di input di massimo 264 bit e blocchi di 512 byte
- **SHA (Secure Hash Algorithm)** (NISTe NSA 1994)
  - Valori hash di 160 bit, messaggi di input di massimo 264 bit e blocchi di 264 bit.
  - Più lento di MD5, ma message digest più grosso
- **SHA-1 è** (1995)
- **SHA-2** (2002)
  - Valori hash di 256 bit
- **RIPEMD (Race Integrity Primitives Evaluation Message Digest)** (1996)

MD5: algoritmo

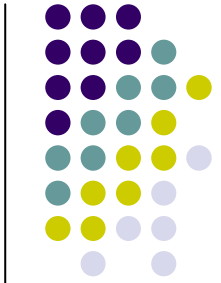
MD5 (Rivest, 1992)

Input: messaggio m di lunghezza arbitraria

Output: message digest (128 bit)



MD5: algoritmo



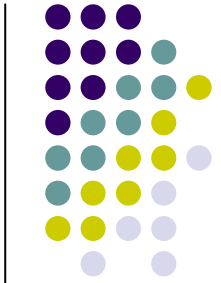
- Dato m di k bit:

  1) Appendere ad m una stringa di bit 100… t.c. la lunghezza ottenuta sia congrua a 448 mod 512. Il padding va aggiunto anche se m è già congruo a 448 mod 512.

  2) Appendere altri 64 bit indicanti la lunghezza di m (i.e. k). Se k >= 264 scrivere k mod 264.

  Usiamo M[0, … N-1] per indicare i blocchi da 32 bit del messaggio ottenuto.

MD5: algoritmo



3) Inizializzazione del buffer MD. E' un buffer di 128 bit: MD = (A,B,C,D) con:

A = 67 45 23 01 (parola da 32 bit)

B = EF CD AB 89

C = 98 BA DC FE

D = 10 32 54 76

MD è memorizzato nel formato little-endian (es. A = 01 23 45 67)

4) Processare il messaggio ottenuto a blocchi di 512 bit (Y0, Y1, … YL-1)

MD5: algoritmo

In sequenza gli Yq per q = 0,1, … L-1 vengono processati (4 round). La struttura dei round è la stessa:

input round: Yq, MD, una porzione di 16 elementi della tabella T composta da parole di 32 bit (T[i]= 232 abs (sin i), i=1, ..,64), una funzione logica con 3 input da 1 bit e output di 1 bit

output round: MD aggiornato

MD5: algoritmo

MD5: per q = 0, …, L-1 esegui la procedura descritta a lato

CV_0 = MD (128 bit) inizializzato al passo 3)

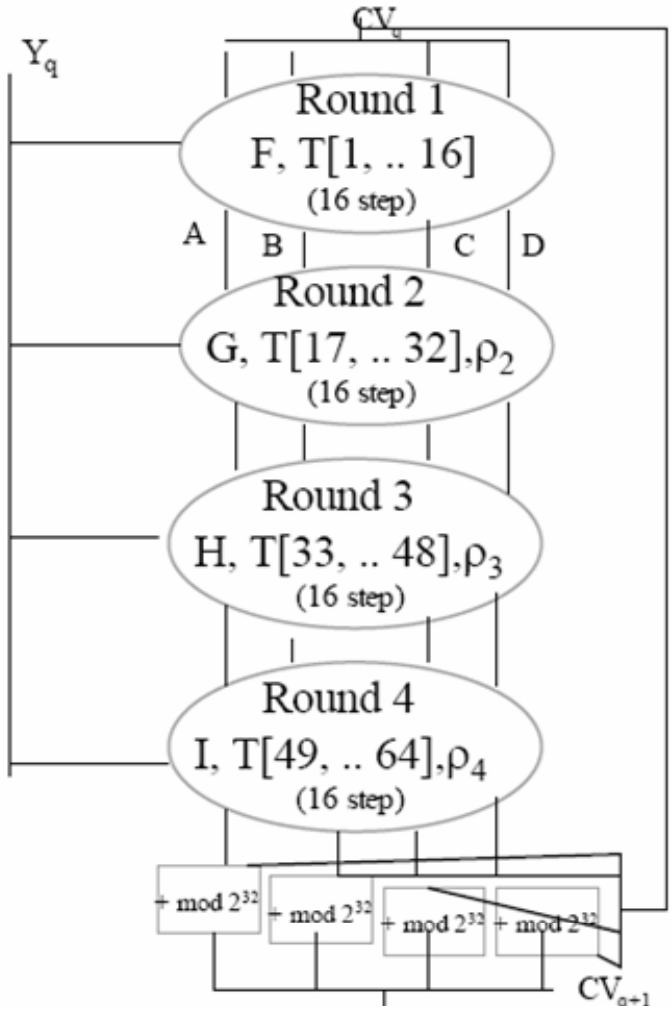
Y_q (512 bit)

F, G, H, I: funzioni logiche

ρ_2, ρ_3, ρ_4 : permutazioni

ρ_1 : id

Output MD5 = CV_L

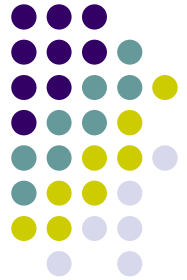


MD5: algoritmo

- **Funzione di compressione**:
  E' usata in ogni round per 16 volte:
  A = B + ((A+g(B,C,D)+X[r(i)]+T[i]<<<s), per i=1,..16
  A,B,C,D sono permutate ad ogni step, in particolare viene fatto uno shift (a dx) circolare a livello di parole (ABCD->DABC) (=> ogni parola viene aggiornata 4 volte per round)

  g: funzione logica primitiva (associata al round)
  applicata bit a bit su B, C e D
  <<<s: shift circolare a sinistra da s bit
  X[k]: k-esima parola di 32 bit del blocco Yq
  T[i]:i-esima parola della porzione di tabella del round

MD5: algoritmo



- **Funzioni logiche**:

  F(b,c,d) = if b then c else d — round 1

  G(b,c,d) = if d then b else c — round 2

  H(b,c,d) = bit di parità di bcd — round 3

  I(b,c,d) = c + (b or not d) — round 4

MD5: resistenza agli attacchi

- Trovare m, m' distinti con lo stesso digest ha una complessità dell'ordine di 2^{64} operazioni, mentre dato un MD trovare un m che lo produce ha complessità dell'ordine di 2^{128} operazioni. Rivest congetturò che MD5 ha la massima robustezza ottenibile con digest di 128 bit.

MD5: resistenza agli attacchi

- Usando crittoanalisi differenziale, è stato provato che è possibile trovare, in tempi ragionevoli, due messaggi m ed m' distinti t.c. l'applicazione di uno qualsiasi dei 4 round produce lo stesso MD. Al momento non è ancora stato provato se sia possibile trovare m ed m' distinti che producono lo stesso digest quando questi sono applicati a tutti e 4 i round [Berson].

MD5: resistenza agli attacchi

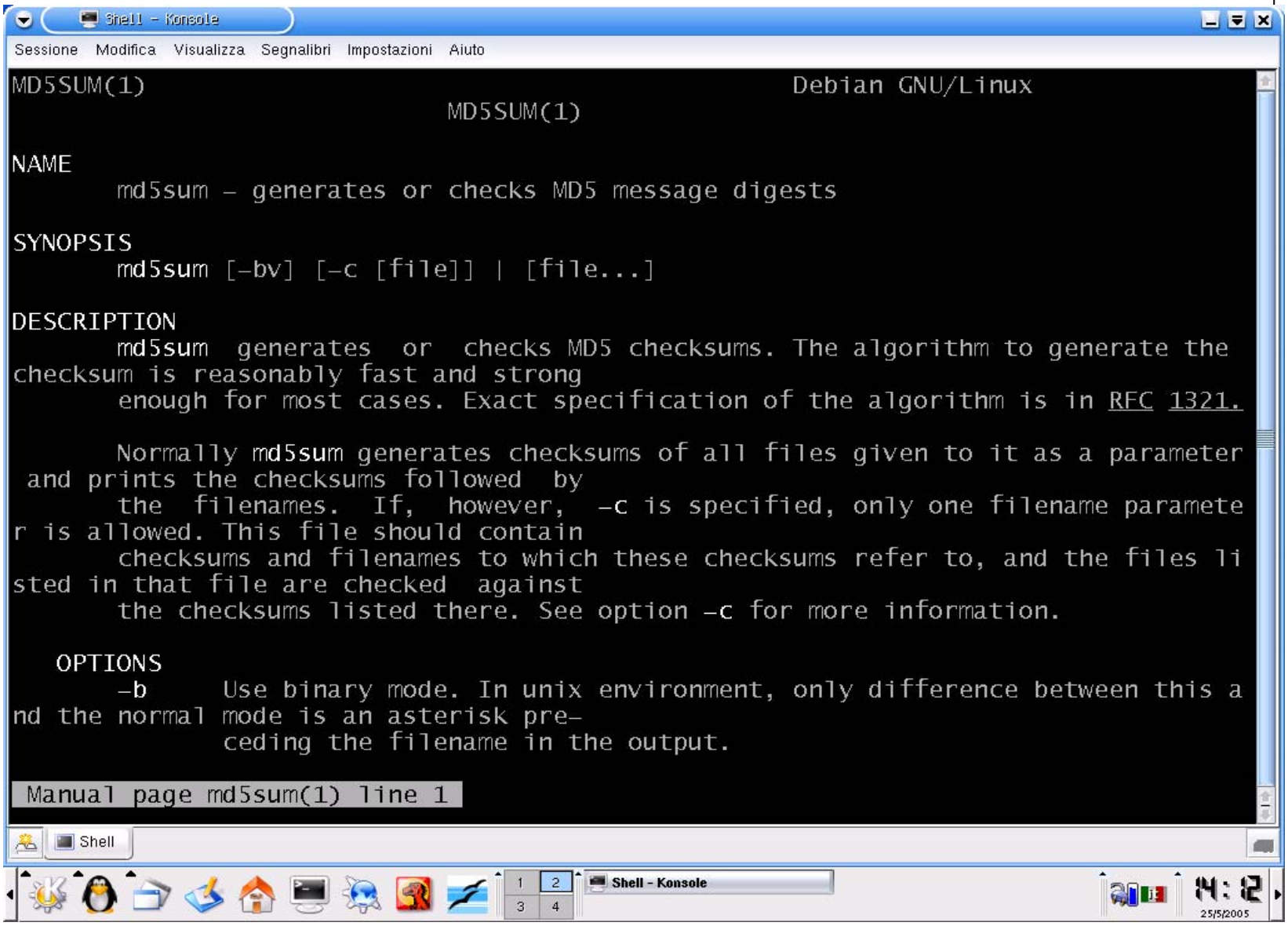
- È possibile trovare un blocco X di 512 bit e due relativi CV, CV' distinti (valori temporanei del MD finale) che, dati in input con X ad uno dei 4 round producono lo stesso output. Al momento questa proprietà non sembra offra una via per attaccare MD5 [Boer & Bosselaers].

MD5: resistenza agli attacchi

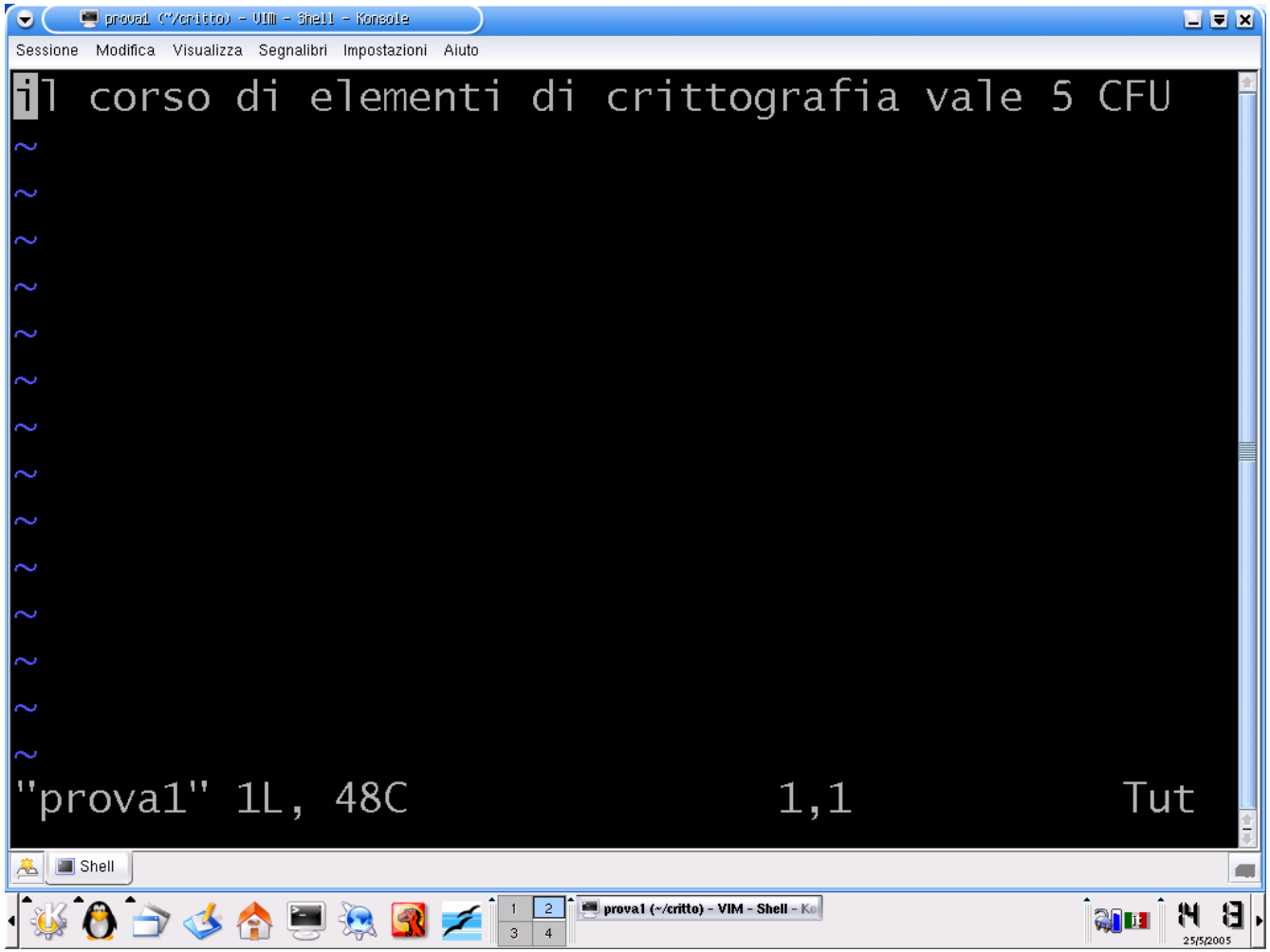
- E' possibile trovare, dato m di 512 bit, un m' diverso da m t.c. applicando interamente MD5 su m ed m' si ottiene lo stesso digest. Nonostante non esista per ora un modo per estendere questa tecnica a m di lunghezza arbitraria, il risultato ottenuto è molto forte e costituisce l'attacco piu' serio portato a MD5 [Dobbertin].

Con la potenza dei calcolatori attuali MD5 è considerato vulnerabile (2^{64} operazioni per trovare una collisione) e l'unica soluzione è aumentare la lunghezza del digest.

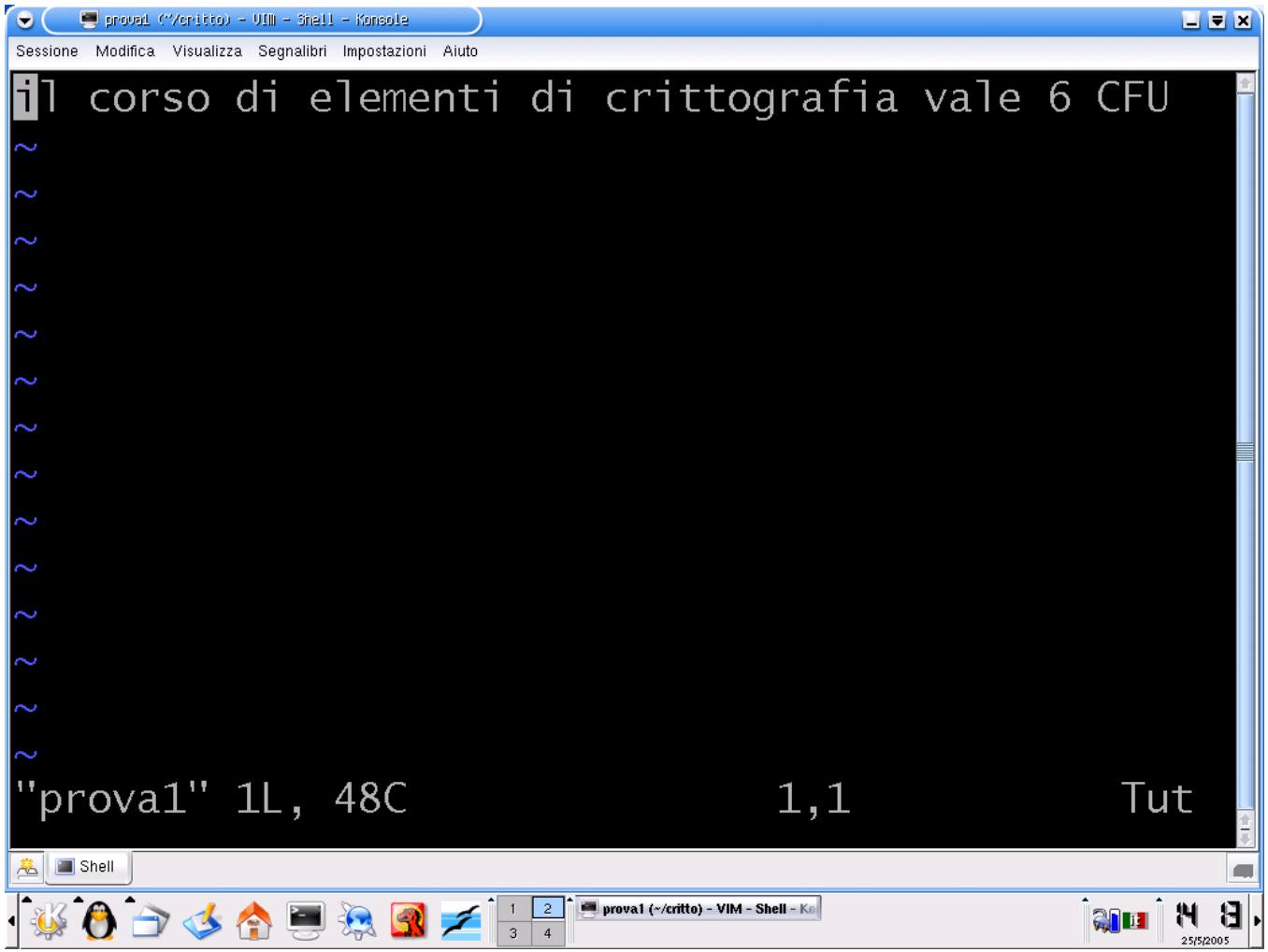
MD5 sotto Linux



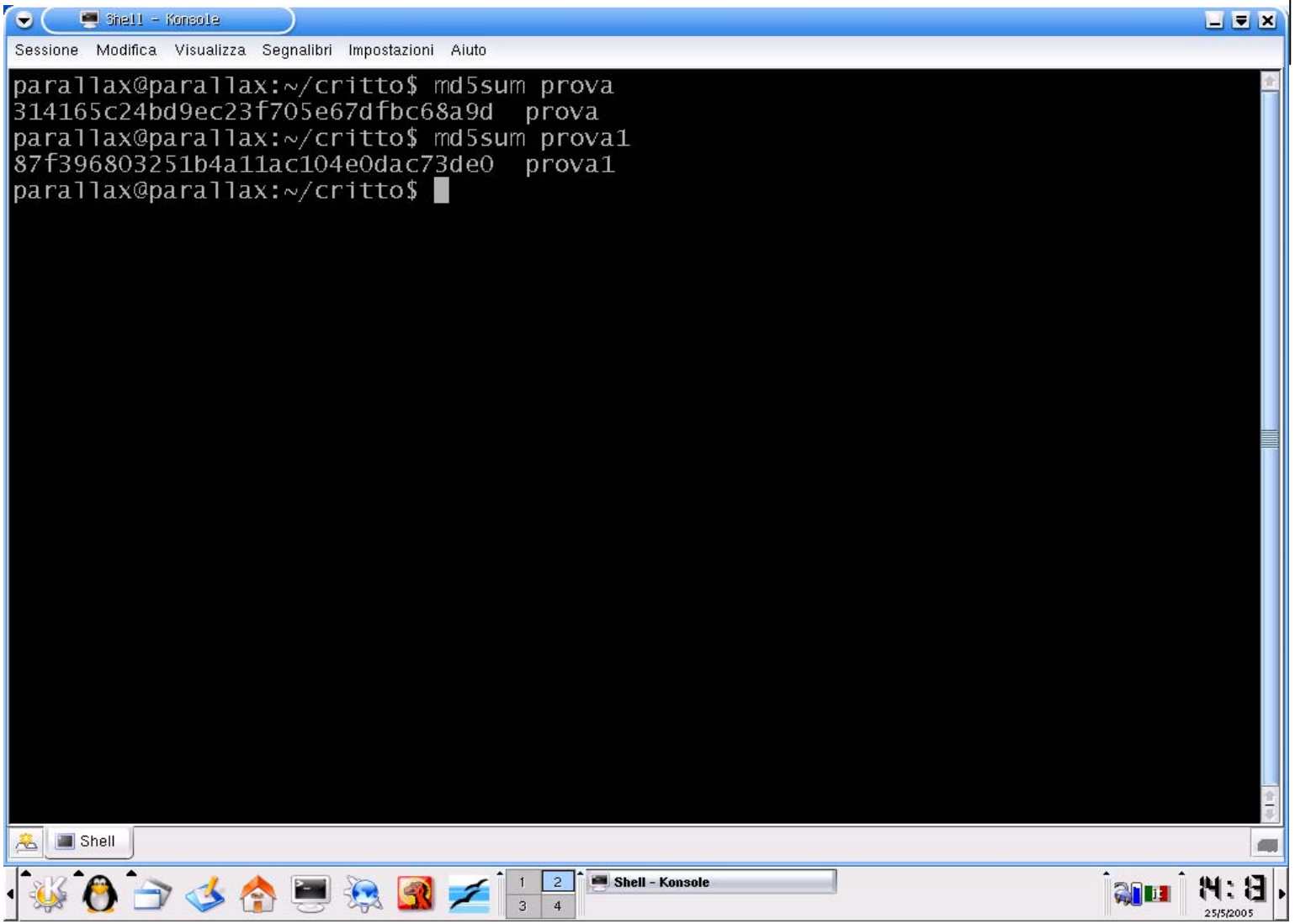
MD5 sotto Linux



MD5 sotto Linux

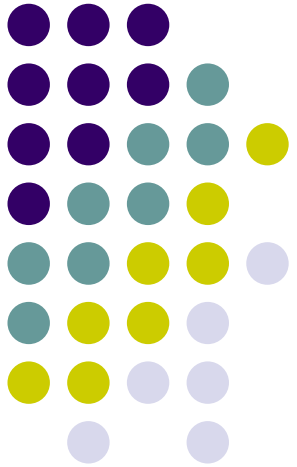


MD5 sotto Linux



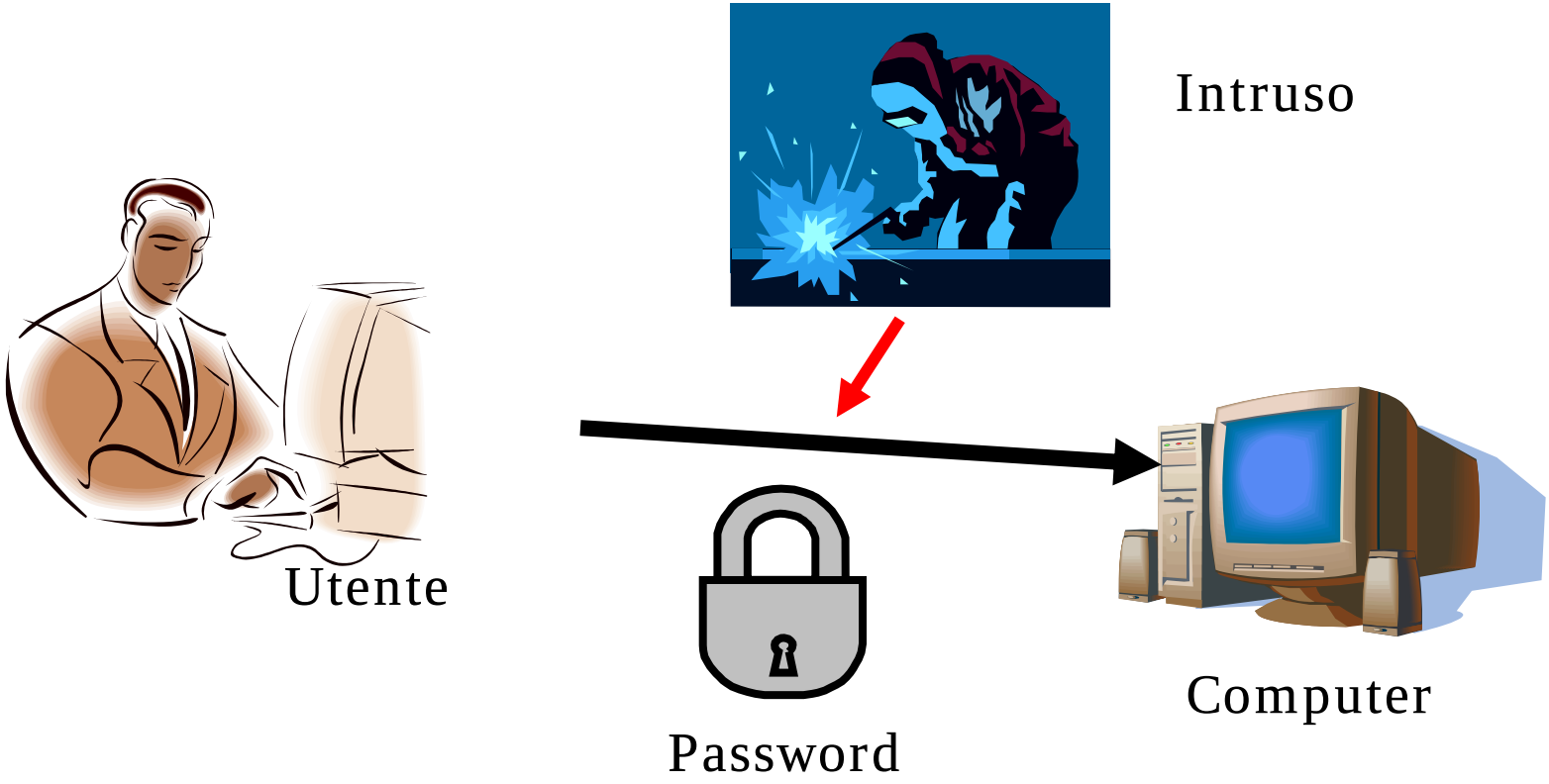
Autenticazione Forte

Paradigma challenge-response



Fabio Cagnizzi

Contesto Applicativo



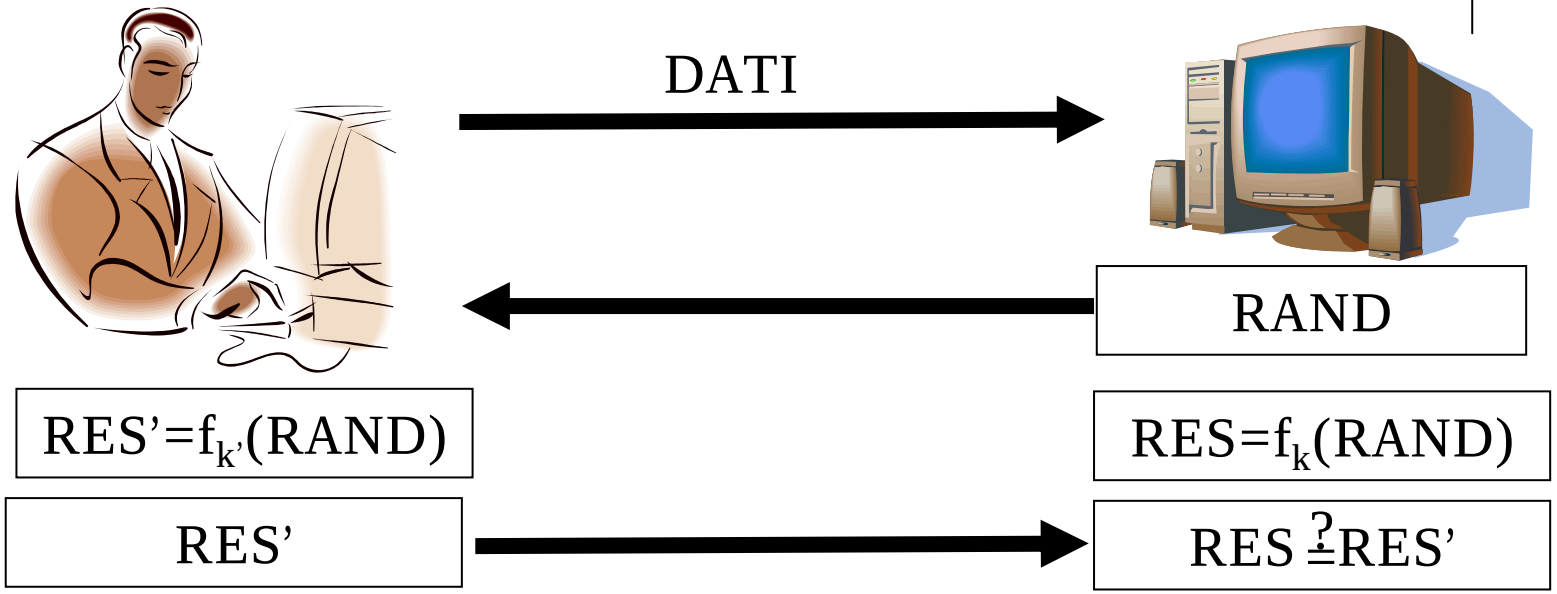
Svantaggi nell'utilizzo di password

- Dal punto di vista della sicurezza, i sistemi a parole di accesso sono imperfetti.
  - Difficoltà nella memorizzazione di **password** che non siano **corte o riconducibili a cose reali**;
  - **Staticità** dei sistemi.
- Come risolvere tali problemi ?
- Le informazioni scambiate tra utente e sistema di autenticazione dovrebbero **variare continuamente**.

Protocollo Challenge - Response

- Ogni volta una nuova domanda e una nuova risposta;
- La risposta dell'utente deve essere formulata in modo che nessun altro possa darla;
- L'utente deve eseguire una trasformazione basata su un segreto, altrimenti anche l'intruso potrebbe rispondere;
- Il protocollo definito prende il nome di **"challenge-response"**, ed è basato su chiavi segrete;
- **Viene evitato di trasmettere la password sulla rete**.

Protocollo Challenge - Response

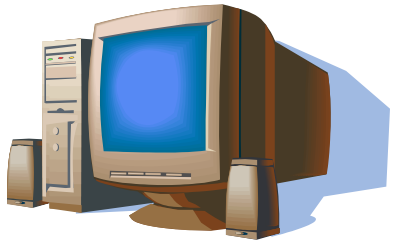


- Sia il computer che l'utente conoscono una funzione unidirezionale ***f*** ed una chiave segreta ***k***;
- Il computer deve stabilire indirettamente che l'utente ha la chiave *k*, **verificando che *k* = *k*'**.

Protocollo Challenge - Response (esempio)

- La funzione *f* è ***k + RAND mod 15***;
- La chiave di “Fabio Cagnizzi” è ***k*=9**.

“Fabio Cagnizzi”



RAND = 12

RES’= $f_{k'}$(RAND) = 12 + 9 mod 15 = 6

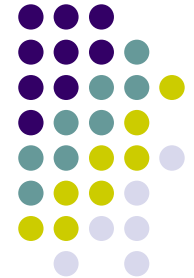
RES=f_k(RAND)

RES’

RES $\stackrel{?}{=}$ RES’

Esito positivo

Protocollo Challenge - Response (dubbi)



- Se il numero RAND invece che casuale fosse **costante** ?

  Si ripresenterebbero i problemi dell'autenticazione debole, in quanto l'intruso potrebbe agevolmente intercettare i messaggi e scoprire la chiave visto che la funzione dà sempre lo stesso risultato.

- Se invece fosse soltanto **prevedibile**

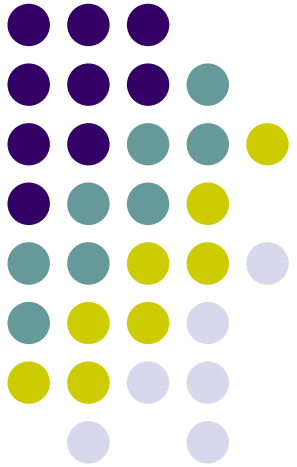
  (es. $RAND_{nuovo} = RAND_{vecchio} + 1$) ?

  Intercettando una serie significativa di messaggi, l'intruso, con una probabilità altissima, potrebbe individuare la chiave sulla base dell'analisi dei vari risultati della funzione.

Protocollo Challenge - Response

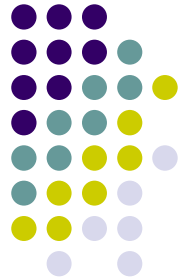
- Riepilogando, i protocolli challenge-response:
  - costituiscono una soluzione sicuramente più affidabile dell'autenticazione debole (userid-password);
  - purtroppo però, attraverso un'analisi attenta, si possono ottenere informazioni riguardo l'autenticazione, in quanto la chiave segreta è posseduta sia dall'utente che dall'autenticatore.

Protocollo Zero Knowledge



Fabio Cagnizzi

Protocollo Zero - Knowledge



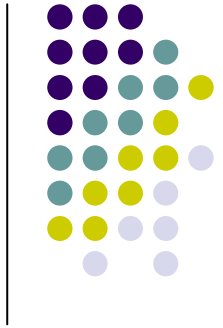
- Si può convincere qualcuno di avere un segreto senza assolutamente rivelarlo ?

  Si.

- Esistono dei metodi con i quali si può convincere qualcuno di possedere informazioni riservate senza dare la minima idea di quali siano.

  I PROTOCOLLI CON CONOSCENZA ZERO

Protocollo Zero - Knowledge



- La situazione è questa:
  - L'utente vuole convincere il sistema di autenticazione di una asserzione (detta **dimostrazione di conoscenza**);
  - Il sistema può accettare o rifiutare tale dimostrazione;
  - La dimostrazione non è intesa con la sua accezione matematica (assoluta) ma **probabilistica**.

Zero – Knowledge
Gioco della radice quadrata

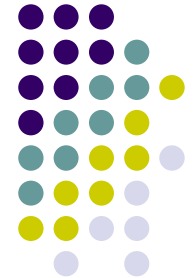
- Un esempio di applicazione di protocolli con conoscenza zero è il cosiddetto gioco della radice quadrata.

- Si conosce un numero, per esempio **s**, il cui quadrato mod 55 è 34;
- Il nostro interlocutore non lo conosce;
- Dobbiamo dimostrargli che lo conosciamo.

Basi Matematiche

- $\boldsymbol{Z_n}$ insieme delle classi resto modulo n;
- Sia ***a*** un intero appartenente a Z_n, allora a è un ***quadrato*** se esiste ***x*** tale che $\boldsymbol{a = x^2 \bmod n}$ e x è una ***radice quadrata mod n;***
- L' insieme dei quadrati in Z_n è denotato con $\boldsymbol{Q_n}$***;***

Zero – Knowledge
Gioco della radice quadrata



Si conosce un numero **s**, il cui quadrato mod 55 è 34 e il nostro interlocutore non lo conosce;

Per dimostrargli che lo conosciamo, gli comunichiamo un numero random r, dicendogli che r^2 mod 55 = 26;

Dopodichè il nostro interlocutore deve decidere se vuole sapere il valore di r oppure di r • s mod 55;

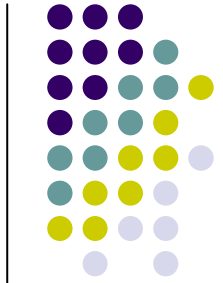
Se conosco s sono in grado di rispondere a tutte le domande dell'interlocutore dimostrandogli (prima o poi) che lo conosco;

Se non lo conoscessi, avrei ad ogni domanda, il 50% di possibilità di indovinare, ma per t domande la probabilità è $(½)^t$

Zero – Knowledge Caratteristiche

- Dal gioco della radice quadrata appaiono le caratteristiche fondamentali dei protocolli a conoscenza zero:
  - i protocolli sono interagenti, perché entrambe le parti fanno scelte a caso (noi scegliamo r, l'interlocutore quello che vuole sapere);
  - la probabilità che un inganno abbia successo dipende dal numero di volte che ripetiamo il procedimento (ogni volta è la metà della precedente).

Zero – Knowledge
Fiat - Shamir



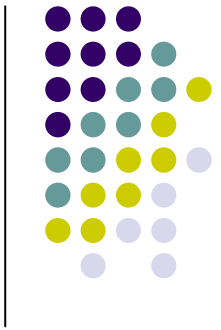
Primo esempio di applicazione di protocolli zero knowledge

↓

Protocollo di Fiat-Shamir

Il protocollo di Fiat-Shamir si basa sull' ipotesi che trovare la radice quadrata di un numero modulo ***n*** sia ***computazionalmente equivalente*** a trovare la fattorizzazione di un numero ***n***.

Zero – Knowledge
Fiat - Shamir

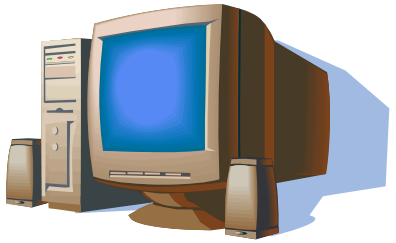


Fase di Setup

- Un centro di distribuzione delle chiavi sceglie due numeri primi, p e q, e calcola il prodotto n = p • q;
- È fondamentale che p e q rimangano segreti, mentre n è pubblico;
- n deve essere tanto grande da rendere impossibile per l'eventuale intruso la fattorizzazione;
- Il centro di distribuzione comunica all'utente un numero s segreto, il numero n e $v = s^2 \bmod n$;
- v servirà come identificazione pubblica dell'utente ed s come parola di accesso

Zero – Knowledge
Fiat - Shamir

L'utente deve convincere il centro di autenticazione di conoscere il segreto s senza rivelarlo



Per t volte !

Random r

$x = r^2 \bmod n$

x

e (0 || 1)

Bit b

$y = r \cdot s^e \bmod n$

y

Verifica

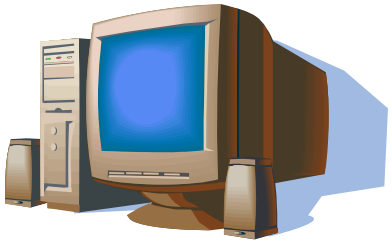
$y^2 \equiv x \cdot v^e \bmod n$

Zero – Knowledge
Fiat – Shamir (esempio)

n=55;

s mod 55 = 23;

v = s^2 mod 55 = 34.



Random r = 31

$x = r^2 \bmod n$

26 →

← e (1)

Bit b

$y = r \cdot s \bmod n = 53$

53 →

Verifica

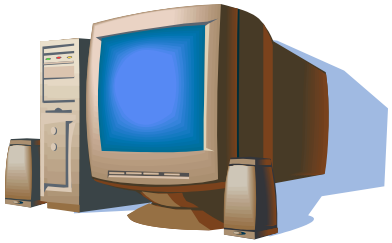
$y^2 \equiv x \cdot v \bmod n$

Zero – Knowledge
Fiat – Shamir (esempio)

n=55;

s mod 55 = 23;

v = s^2 mod 55 = 34.



Random r = 8

x = r^2 mod n

9

e (0)

Bit b

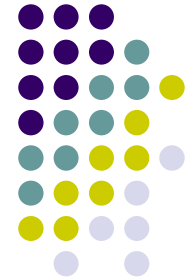
y = r mod n = 8

8

Verifica

$y^2 \equiv$ x mod n

Zero – Knowledge
Fiat – Shamir (esempi)



n=55, s mod 55 = 23, s^2 mod 55 = 34;

1) r=31

   a) $x = r^2 = 961 \equiv_{55} 26$ (utente → computer)

   b) e = 1 (computer → utente)

   c) $y = r \cdot s \equiv_{55} 31 \cdot 23 \equiv_{55} 53$ (utente → computer)

   d) il computer verifica che $(r \cdot s)^2 \equiv_{55} 4 \equiv_{55} x^2 \cdot s^2$ ($v = s^2$ mod n)

2) r=8

   a) $x = r^2 = 64 \equiv_{55} 9$ (utente → computer)

   b) e = 0 (computer → utente)

   c) y = r mod n = 8 (utente → computer)

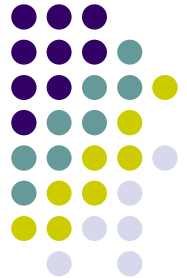
   d) il computer verifica che $x = 9 \equiv_{55} y^2$

Zero – Knowledge
Fiat – Shamir Attacchi

x va trasmesso prima di ***e***;

Se l'intruso prova ad autenticarsi al posto dell'utente, ha una probabilità del 50% di superare la prova.

Zero – Knowledge
Fiat – Shamir Attacchi (1)



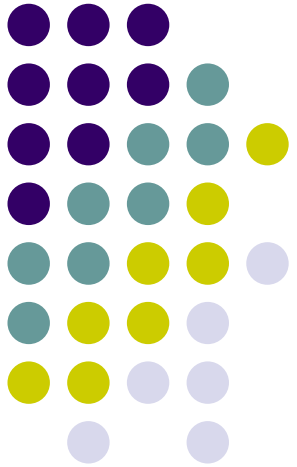
- L'intruso sceglie r;
- Calcola e invia r^2;
- Riceve il bit e:
  - se e = 0, sa rispondere, essendo a conoscenza di r;
  - se e = 1, non sa rispondere, non conoscendo il valore di s.

Zero – Knowledge
Fiat – Shamir Attacchi (2)

- L'intruso sceglie r;
- Calcola e invia $r^2 = x^2 / s^2$;
- Riceve il bit e
  - se e = 0, non sa rispondere: r= x / sqrt(s^2), ma calcolare la radice di s^2 è proprio ciò che non è in grado di fare;
  - se e = 1, sa rispondere con x, infatti
  - $r^2 = x^2 / s^2 \rightarrow r \cdot s = x$.

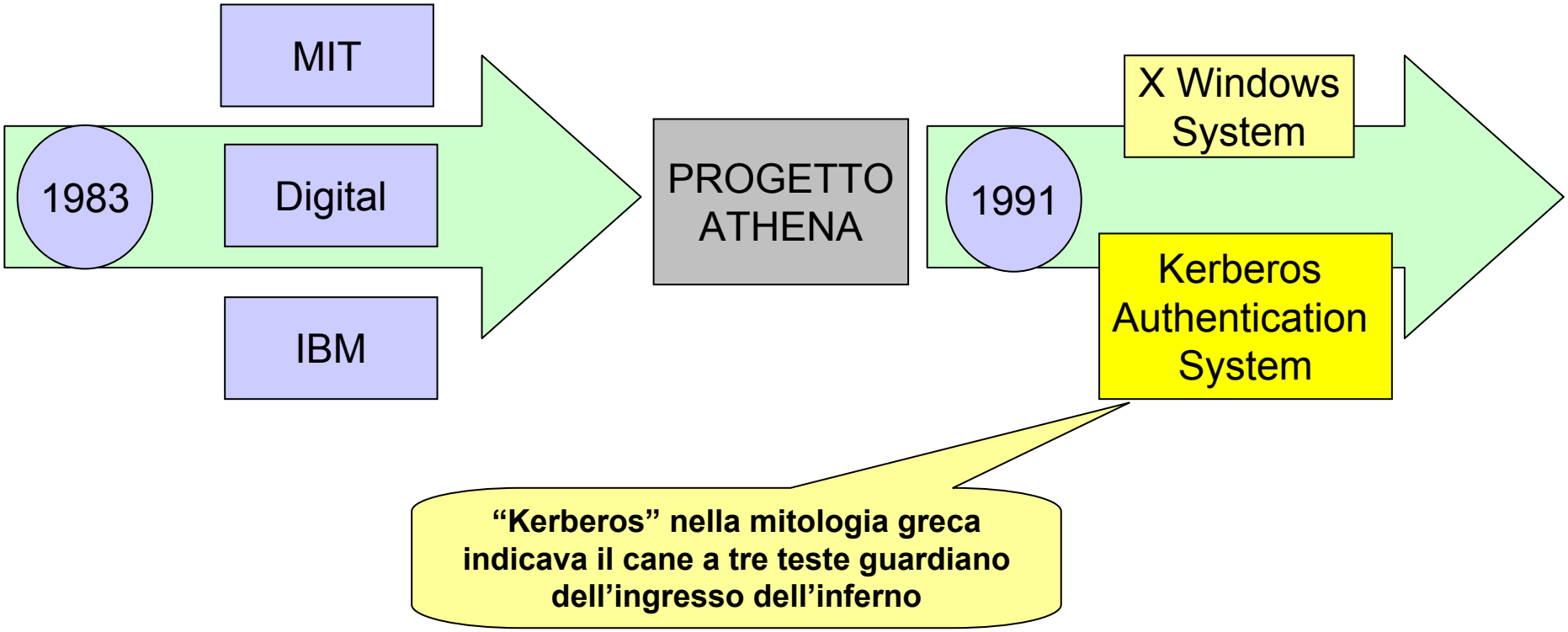
Kerberos

Sicurezza in rete



Federico De Felici

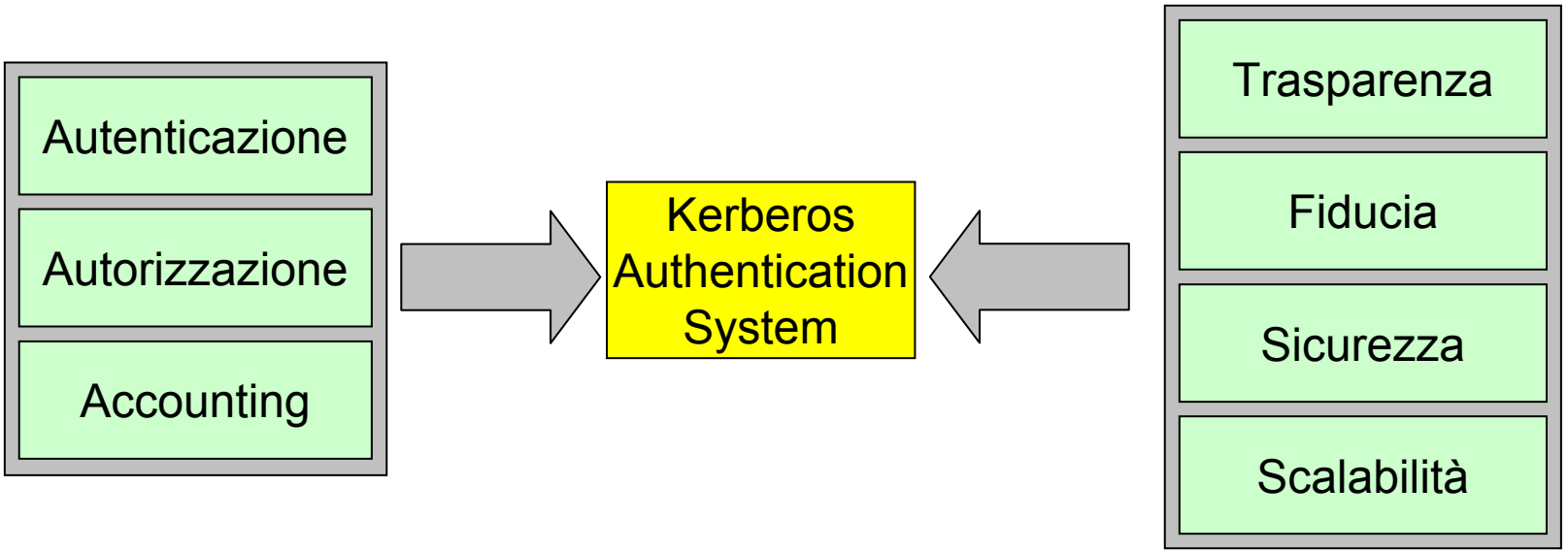
Introduzione



Introduzione

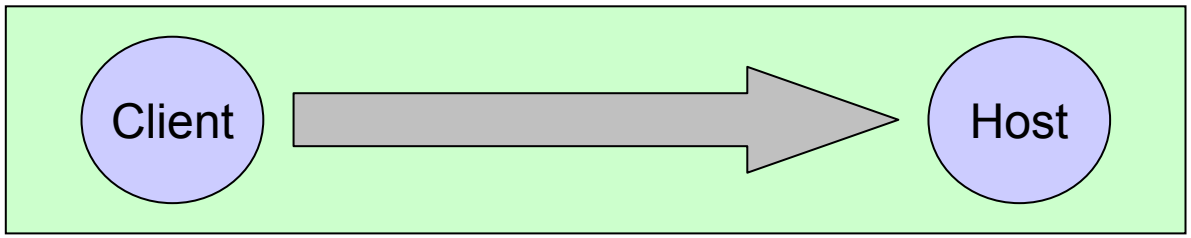
- Kerberos è un protocollo di autenticazione per i servizi di rete

- Obiettivi

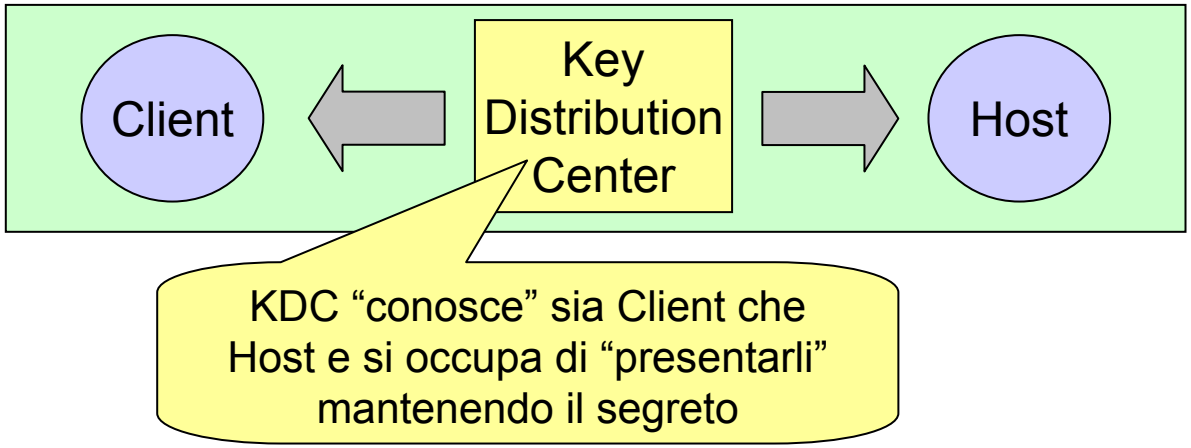


Introduzione

- Modello di Fiducia a due parti

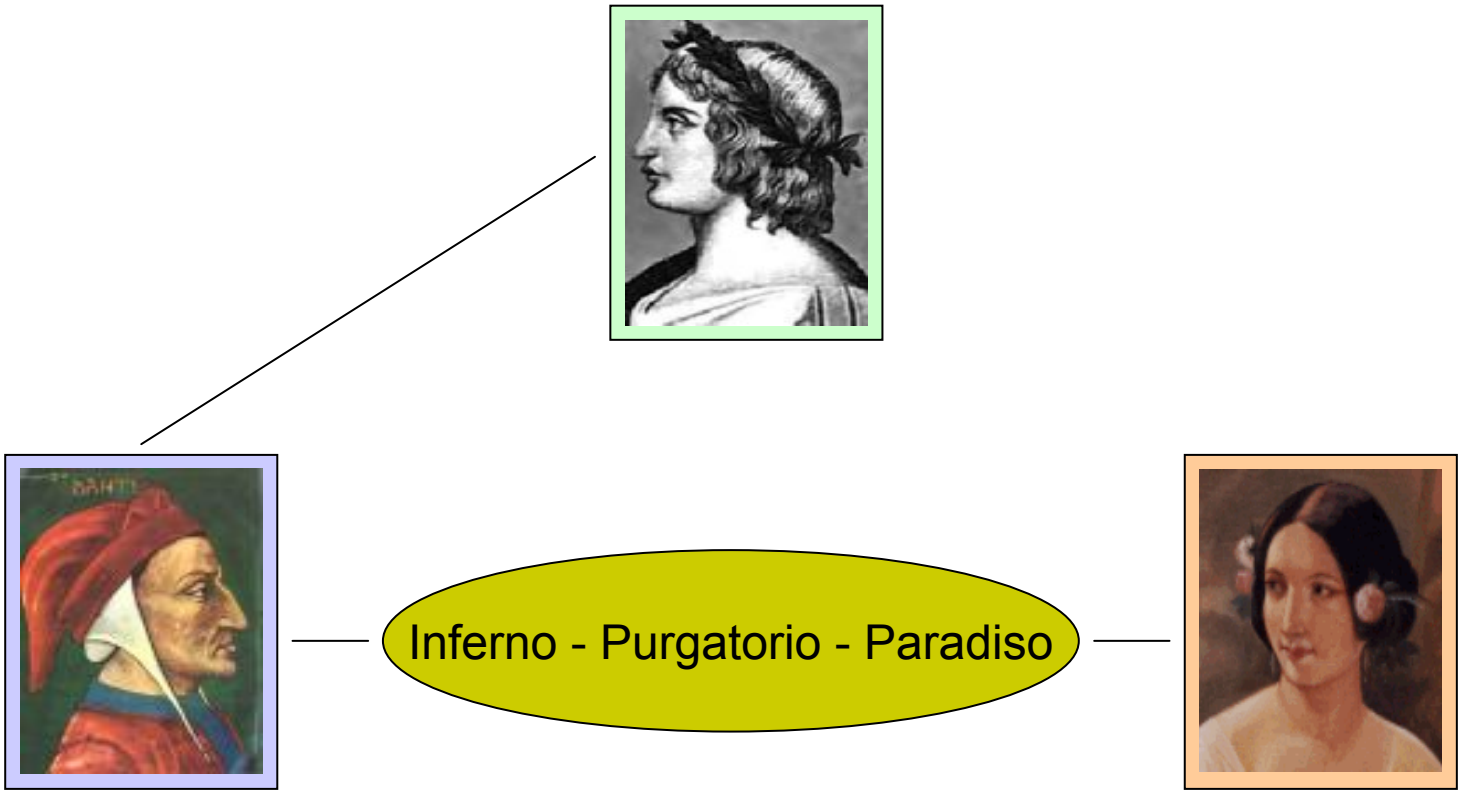


- Modello di Fiducia a tre parti

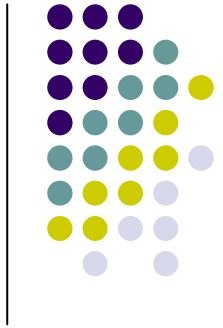


Protocollo Semplificato

- Visione allegorica



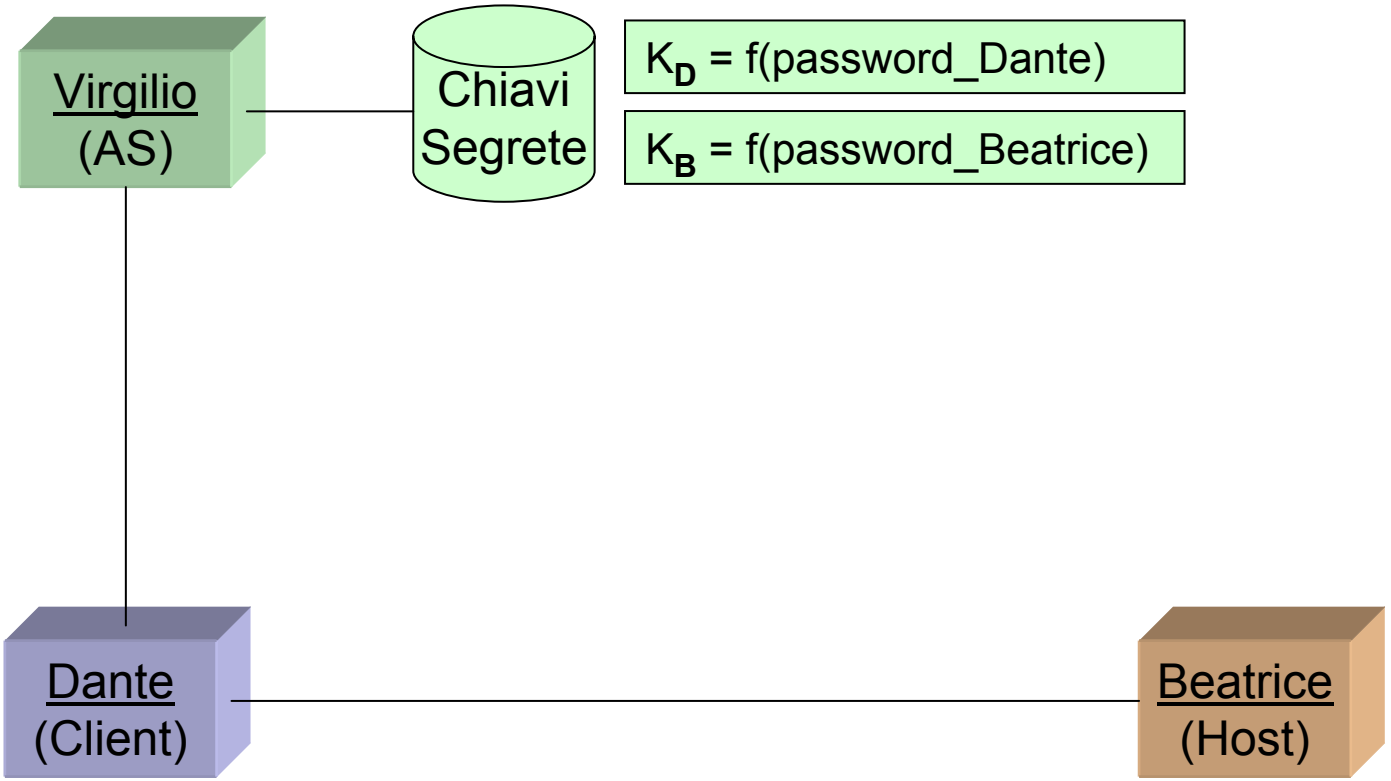
Protocollo Semplificato



- Special Guest
  - Dante: Client che vuole accedere all'Host (Beatrice)
  - Beatrice: Host (server) che offre servizi
  - Virgilio: Authentication Server (AS)
    - AS è la terza parte di cui Client e Host devono fidarsi in quanto possiede le informazioni (chiavi segrete) per poterli autenticare

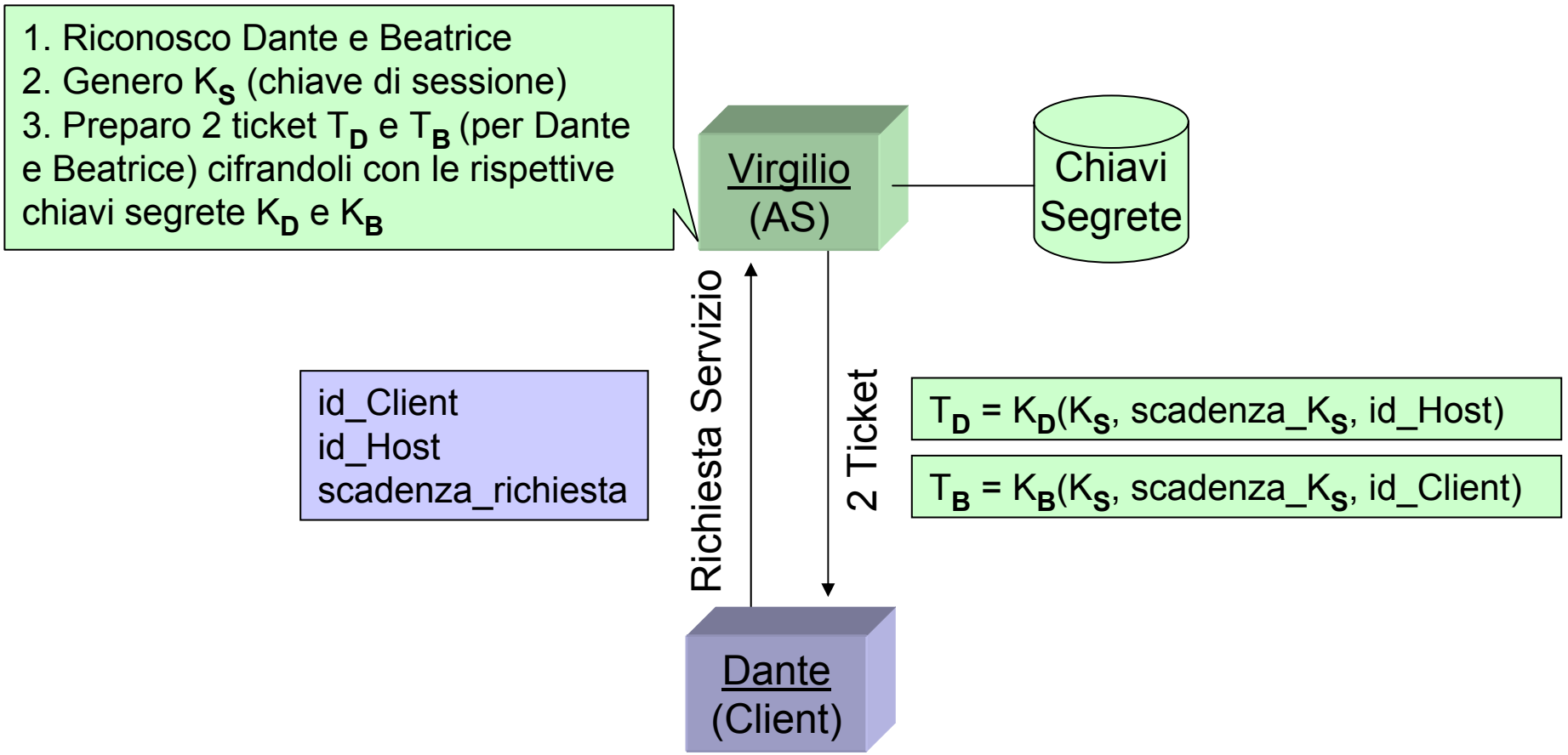
Protocollo Semplificato

- Aumentiamo il dettaglio con l'assunzione che chiave segreta K_i = f(password_i)



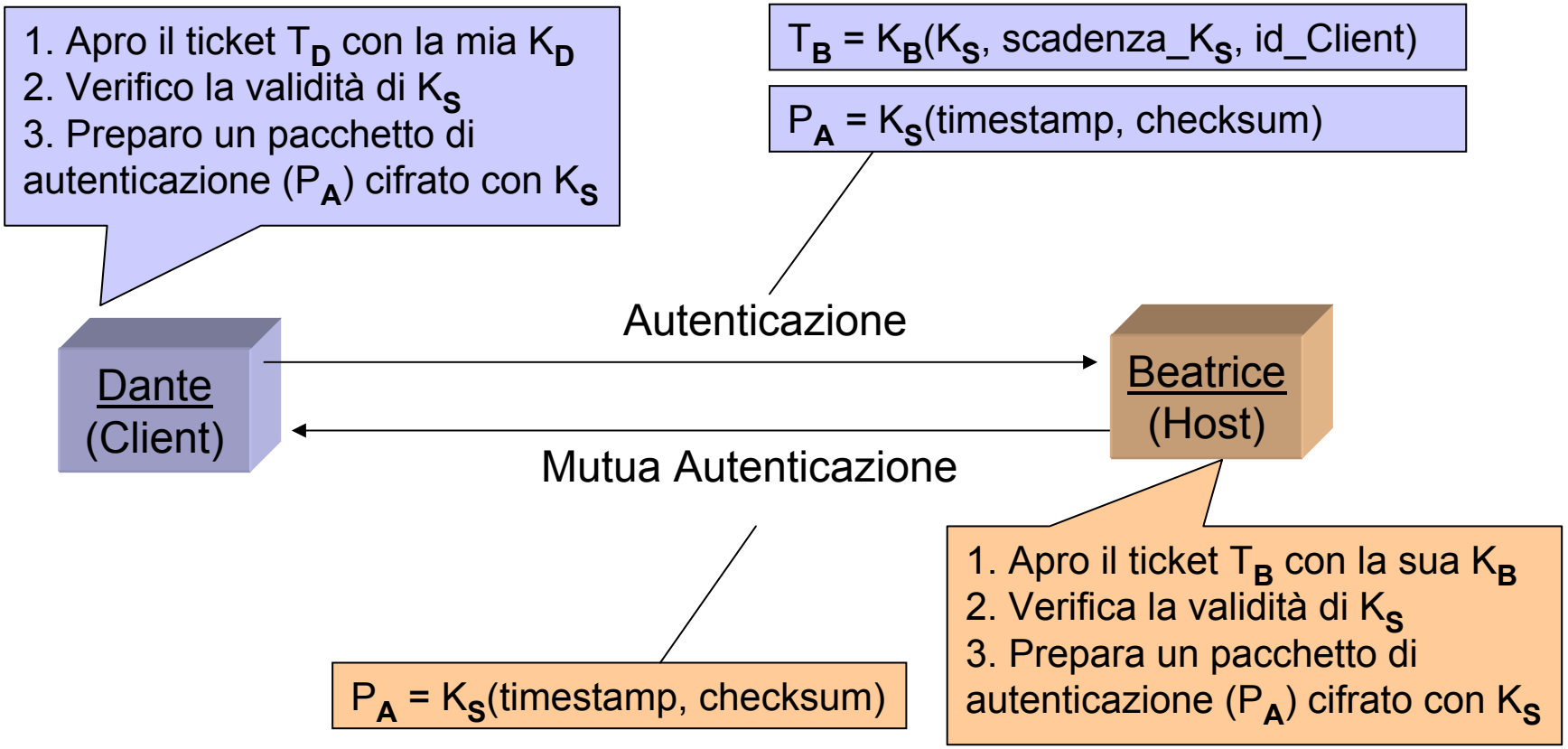
Protocollo Semplificato

- Dialogo Dante - Virgilio



Protocollo Semplificato

- Dialogo Dante - Beatrice



Protocollo Semplificato

- Problema
  - Per via dell'uso di timestamp e scadenza è importante una corretta sincronizzazione tra i clock dei vari componenti del sistema
  - Per ogni richiesta che Dante fa, esso deve "autenticarsi" a Virgilio. Ciò comporta un uso frequente della chiave segreta condivisa tra Dante e Virgilio
  - Collo di bottiglia rappresentato dall'AS

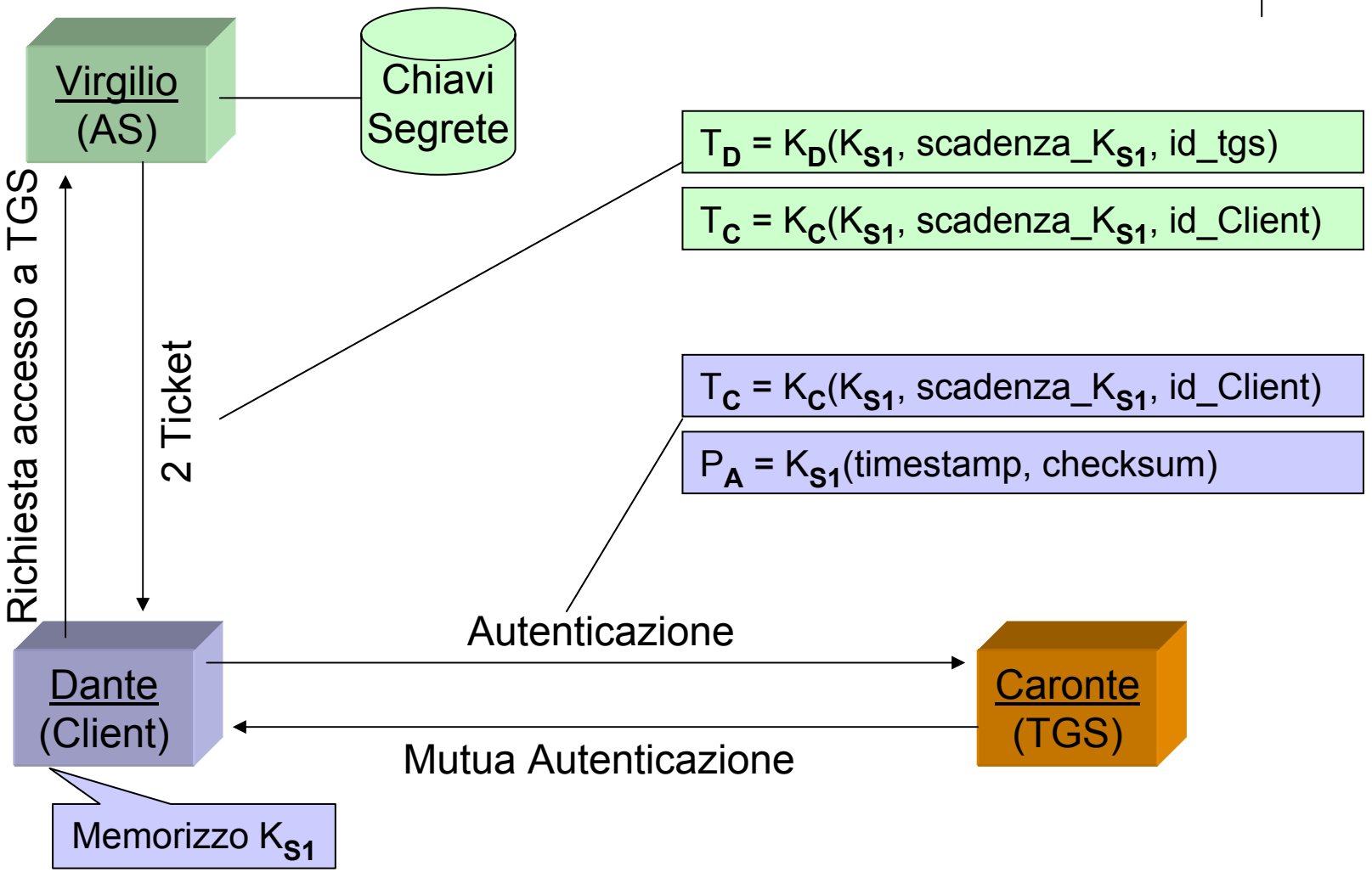
Protocollo Semplificato

- Soluzione
  - Mantenere la chiave segreta (password) in una cache sul Client (meglio di no!)
  - Viene previsto l'impiego di un servizio aggiuntivo definito Ticket Granting Server (TGS)

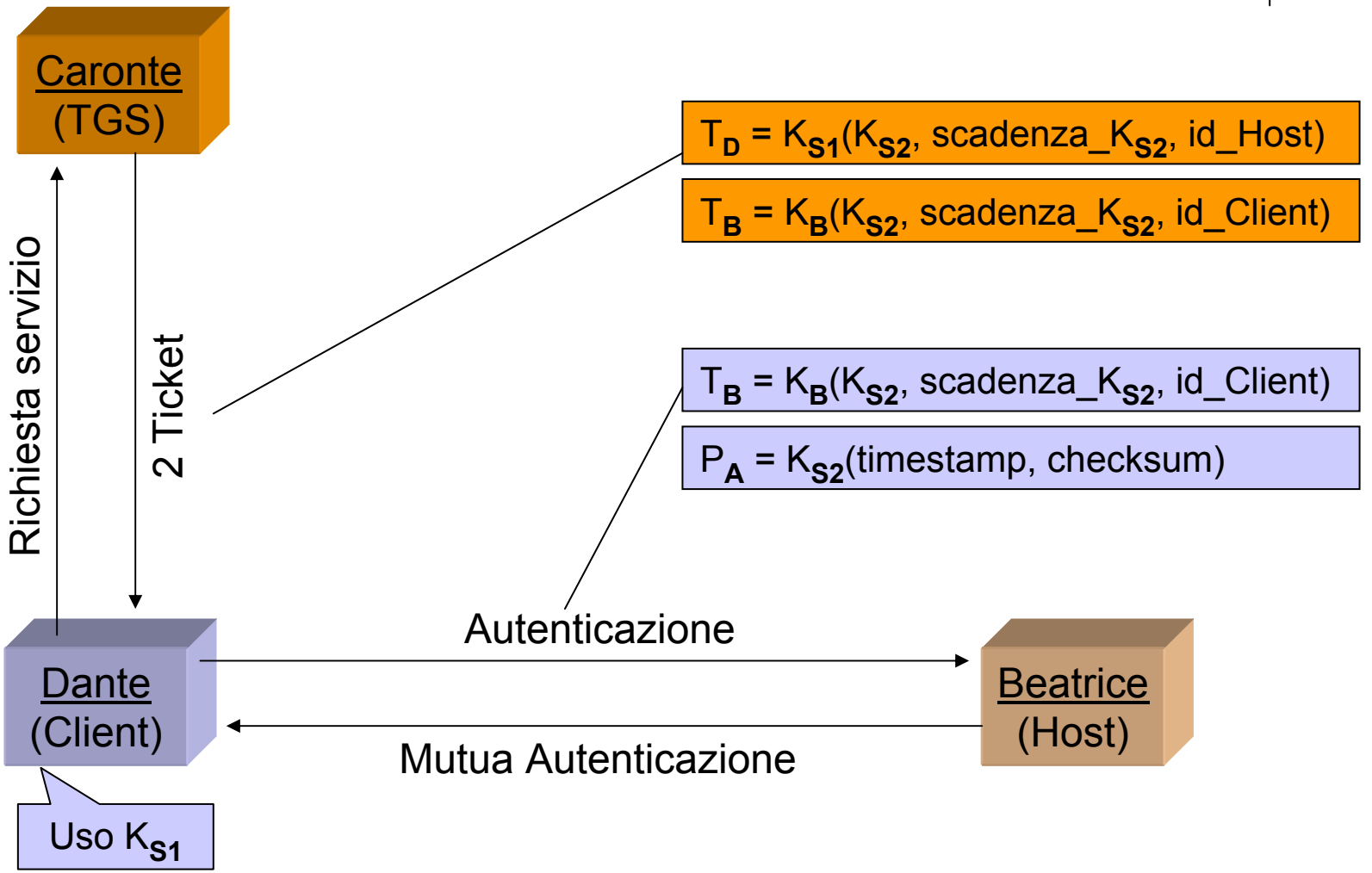
Protocollo Modificato

- Ticket Granting Server (TGS)
  - logicamente distinto dall'AS
  - può risiedere sullo stessa macchina dell'AS oppure su elaboratore distinto

Protocollo Modificato



Protocollo Modificato



Protocollo Modificato

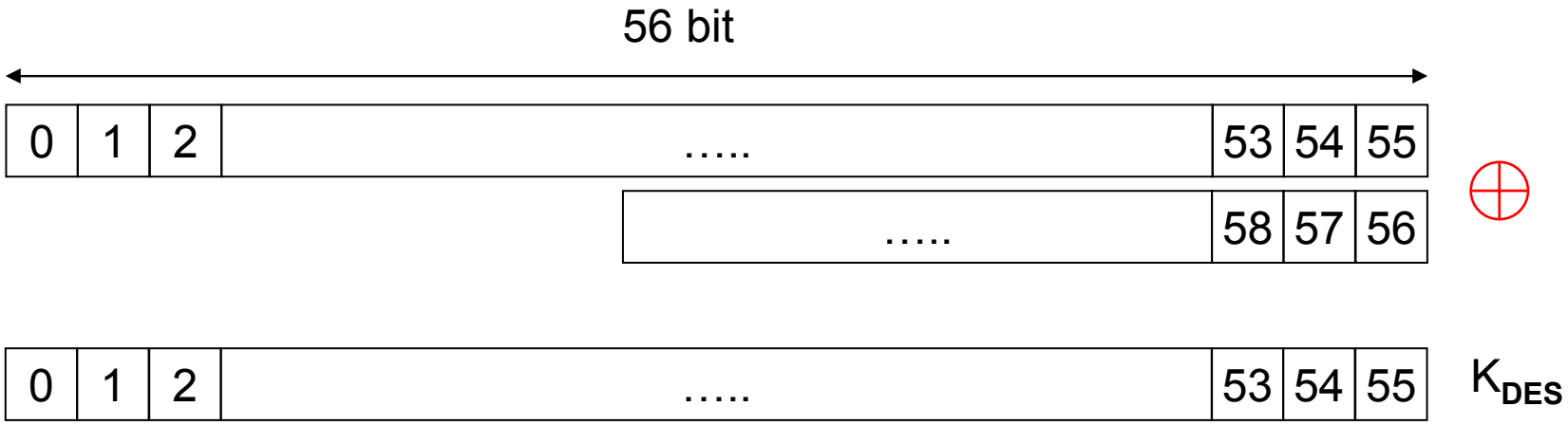
- Riassumendo
  - Quando Dante (Client) vuole accedere a un servizio, chiede prima un ticket a Virgilio (AS) per accedere a Caronte (TGS)
  - Viene cosi generata la chiave di sessione K_{S1} che viene usata da Dante per richiedere accesso a servizio a Caronte
  - Da questo momento fino alla scadenza di K_{S1}, Dante può fare richieste a Caronte per accedere a servizi (invece che a Virgilio), poiché i ticket che riceve sono codificati proprio con K_{S1}

Calcolo della Chiave Segreta

- Viene calcolata sulla base delle password di Dante e Beatrice
- Ogni password è codificata con codice ASCII a 7 bit

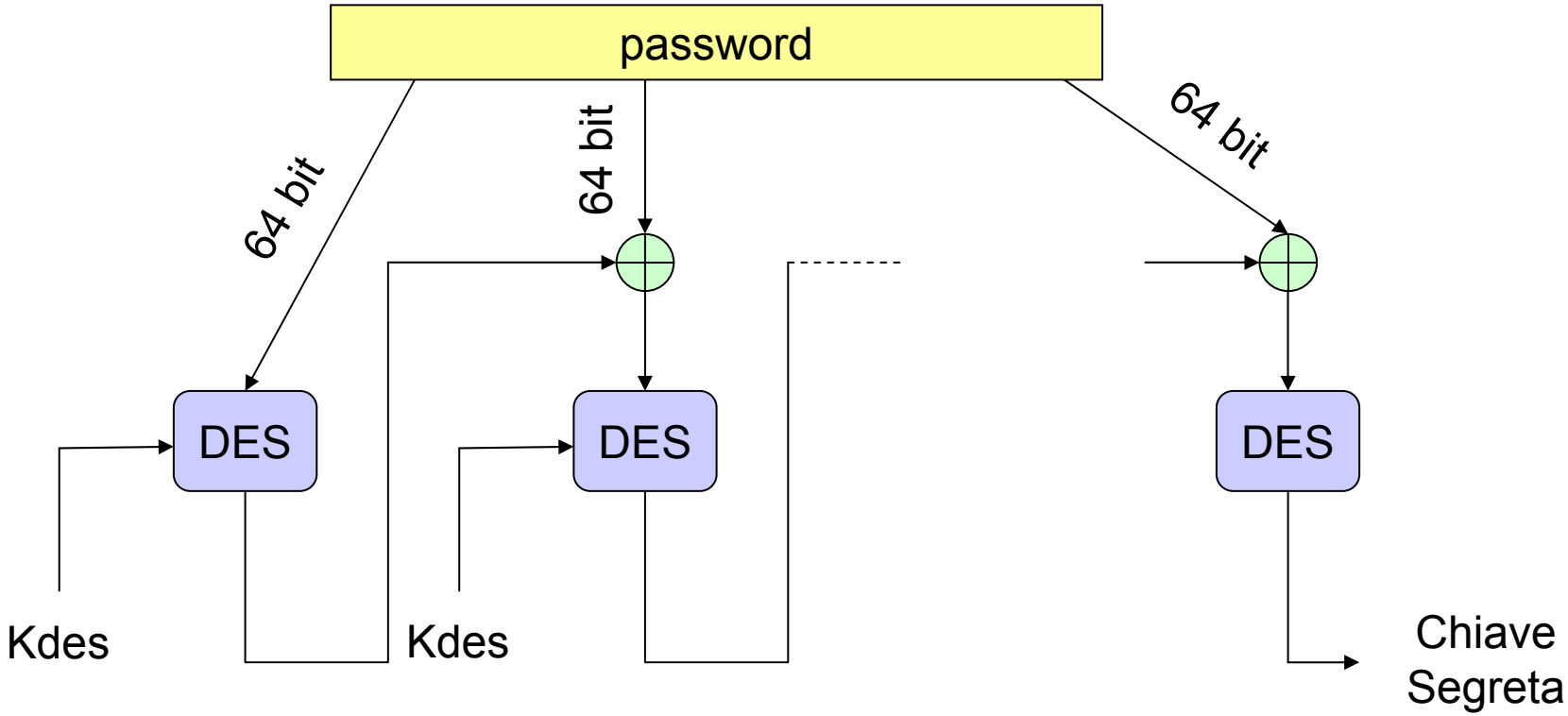
Calcolo della Chiave Segreta

- La stringa della password viene “ripiegata a ventaglio” su una lunghezza di 56 bit
- Quindi si procede con lo XOR tra le diverse “pieghe” ottenendo una chiave (K_{DES}) lunga 56 bit

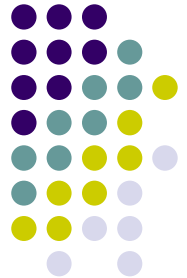


Calcolo della Chiave Segreta

- K_{DES} viene usata come chiave DES per cifrare la stringa della password originaria (64 bit alla volta)



Conclusioni



- L'algoritmo di cifratura della password garantisce una buona resistenza agli attacchi di un crittoanalista
- Ma…Kerberos si basa su delle assunzioni fondamentali
  - La rete non è sicura
    - la password non viene trasmessa nella rete
  - I Client e Host sono sicuri
    - se cosi non fosse un attaccante potrebbe impadronirsi di una macchina della rete e pregiudicare il funzionamento del protocollo…ma questa è un'altra storia