Autenticazione

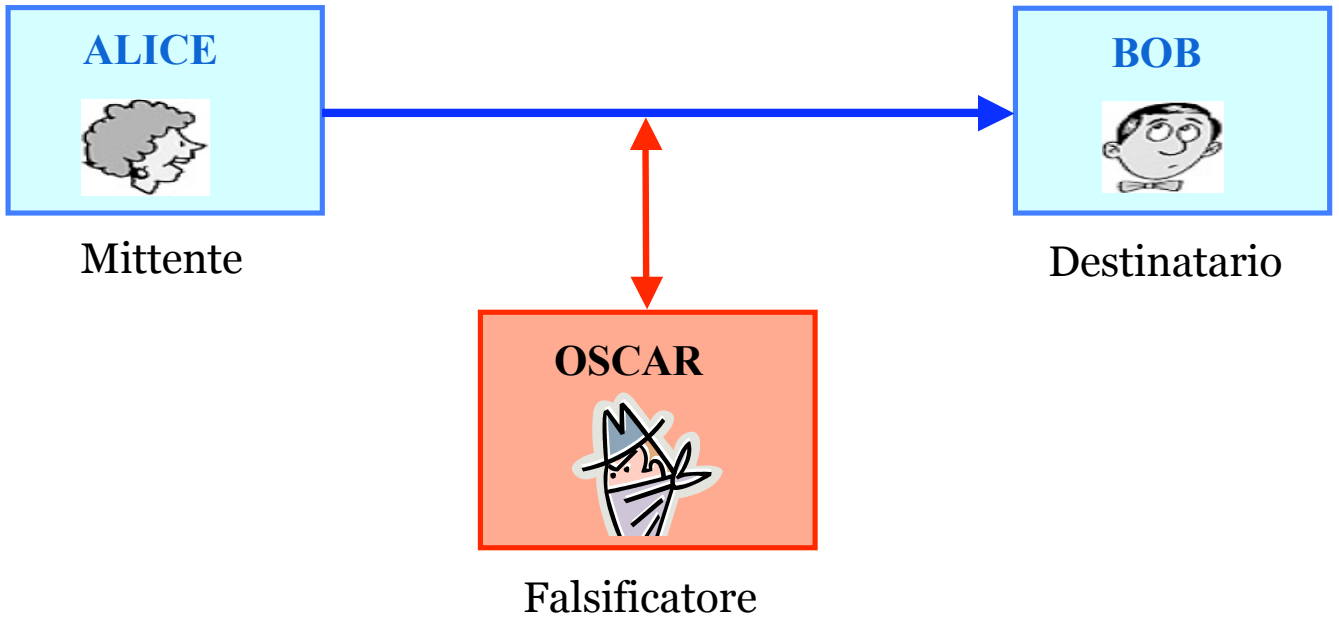
A cura di :
Mara Corona
Francesco Scarano
Ilaria Scarano

SOMMARIO

- Message Authentication
  - Firma digitale (Cenni)
  - Funzioni di Hash
  - MAC (Message Authentication Codes)
- Entity Authentication
  - Autenticazione debole (con Password)
  - Autenticazione forte (Challenge-Response)
  - Protocolli di tipo Zero-Knowledge
- Autenticazione Client/server: Kerberos

MESSAGE AUTHENTICATION

Obiettivo: garantire l'integrità dei messaggi scambiati anche in presenza avversari attivi sulla rete che mandano i propri messaggi



L'Autenticazione è ortogonale alla segretezza per cui spesso in un sistema si devono **garantire ambedue**

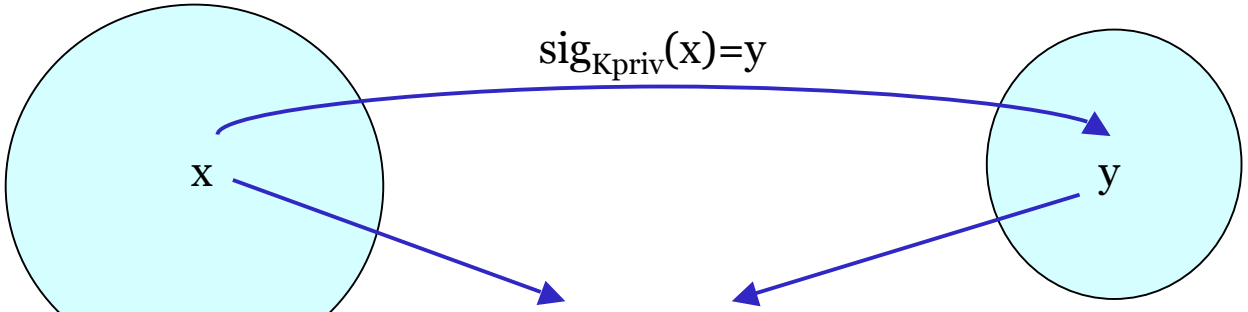
FIRMA DIGITALE

- Idea simile a quella di una normale firma
- Il messaggio x sarà caratterizzato da un'unica firma digitale che è funzione del messaggio e attaccata ad esso.

E' una funzione

Spazio del messaggio

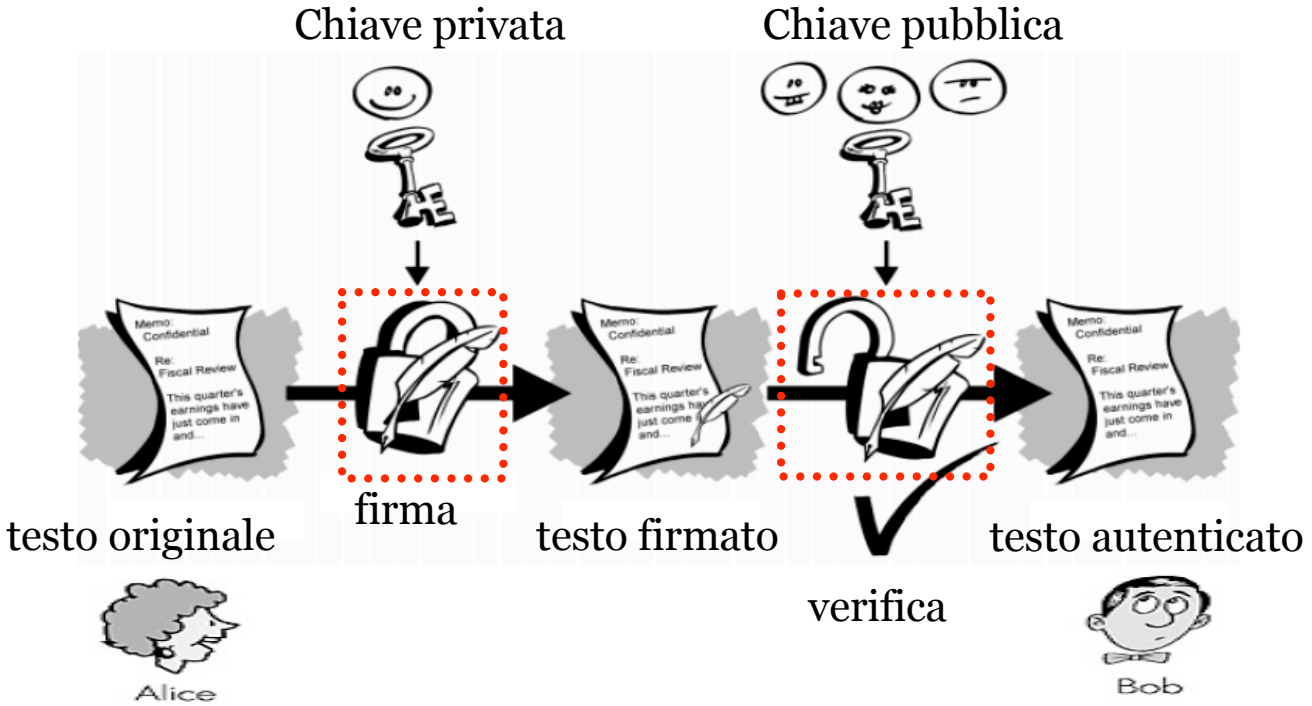
Spazio della firma



$$ver_{Kpub}(x,y)= \begin{cases} \text{true se } y = sig(x) \\ \text{false se } y \neq sig(x) \end{cases}$$

FIRMA DIGITALE

- Alice firma il suo messaggio x con la sua chiave privata $\mathbf{K_{priv}}$, $\mathbf{y = sig_{Kpriv}(x)}$
- Alice invia **(x,y)** a Bob
- Bob applica la funzione di verifica $\mathbf{ver_{Kpub}(x,y)}$ con la chiave pubblica di Alice



FIRMA DIGITALE

Proprietà

- Solo Alice può firmare il suo documento con la chiave privata K_{priv}
- Chiunque può verificare la firma con la chiave pubblica K_{pub}
- **Autenticazione**: Bob è sicuro che sia stata Alice a firmare il messaggio
- **Integrità**: il messaggio x non può essere alterato dal momento che ciò verrebbe scoperto attraverso la funzione di verifica
- **Non ripudiabilità:** nessun utente deve poter ripudiare o negare i messaggi da lui spediti

La firma digitale è una garanzia del messaggio alla stregua della sottoscrizione di un documento cartaceo anche dal punto di vista legale

Schemi

- RSA
- El Gamal
- DSA

FUNZIONI DI HASH

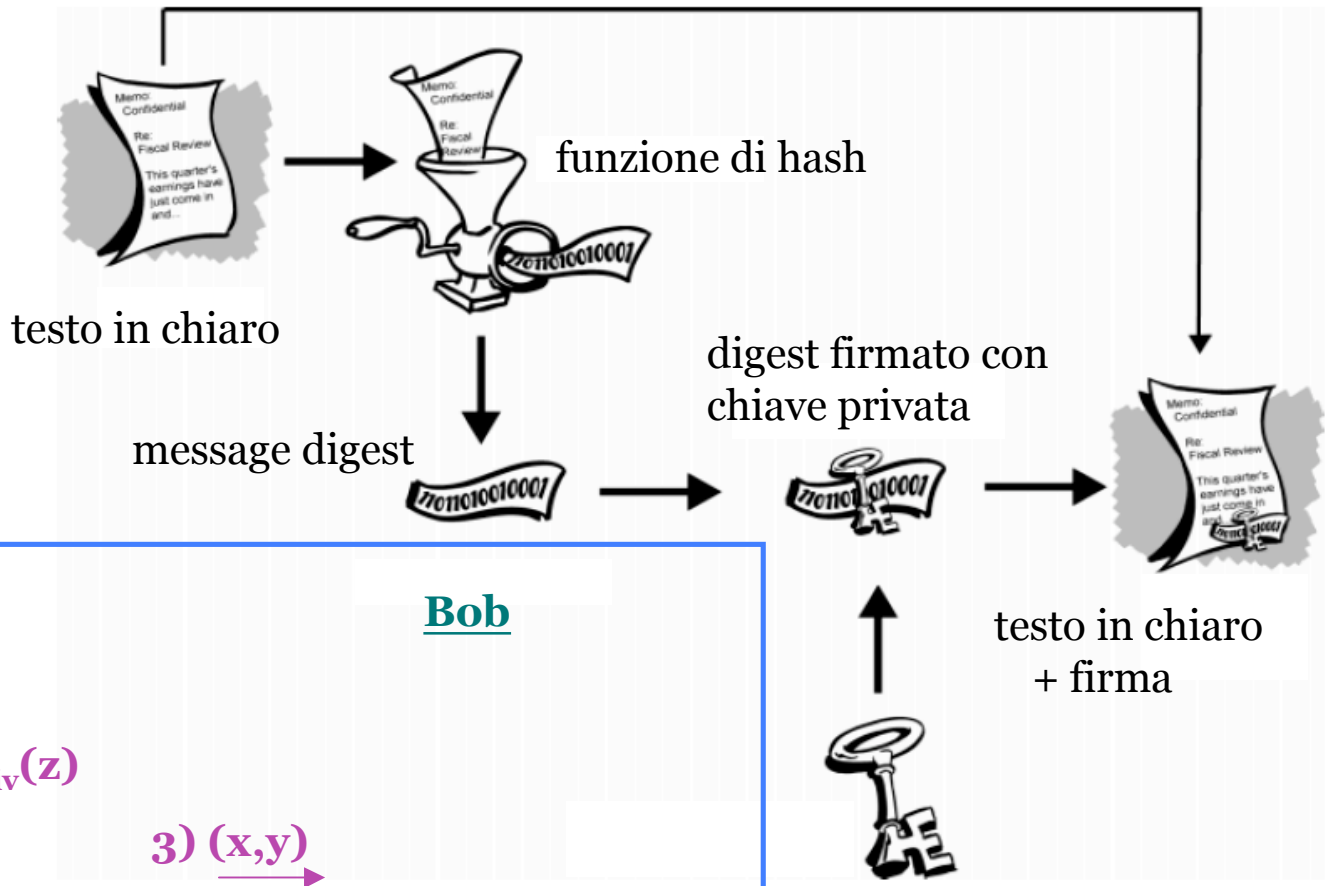
Funzione che, dato un qualunque messaggio di lunghezza arbitraria,ne produce un'impronta (message digest) di lunghezza prefissata (100-200 bit).

- **Utilità**: possibilità di utilizzare l'impronta come rappresentazione compatta del messaggio, firmando l'impronta anziché l'intero messaggio

CARATTERISTICHE

UNIDIREZIONALITA' **ASSENZA DI COLLISIONI**

FUNZIONI DI HASH



Protocollo:

Alice

1) $z = h(x)$

2) $y = sig_{Kpriv}(z)$

3) (x,y) →

Bob

4) $z = h(x)$

5) $ver_{Kpub}(z,y)$

FUNZIONI DI HASH:PROPRIETA'

- **One-way (unidirezionale)**:
  - se dato un qualsiasi messaggio x è computazionalmente semplice calcolare h(x)
  - se data un'impronta z, è computazionalmente impraticabile trovare un messaggio x tale che

$$h(x)=z$$

- **Weak collision resistant**: se dato un messaggio x è è computazionalmente impraticabile trovare un messaggio qualsiasi x' tale che

$$h(x) = h(x')$$

- **Strong collision resistant**: se è computazionalmente impraticabile trovare due messaggi x e x' tali che

$$h(x)=h(x')$$

FUNZIONI DI HASH: ATTACCHI

- Se h(x) non è unidirezionale, una intruso può calcolare x da h(x) nei casi in cui x è cifrato

- **Attacco a forza bruta "semplice"**. Se h(x) non è Weak collision free, un intruso può sostituire x con x'

Alice	Oscar	Bob
1) $z = h(x)$		
2) $y = sig_{Kpriv}(z)$		
	3) (x, y) →	
	4) (y, x') →	
		5) $z = h(x') = h(x)$
		6) $ver_{Kpub}(z,y) = true$

FUNZIONI DI HASH: ATTACCHI

- **Attacco del compleanno**: Se h(x) non è Strong collision free, un intrusore può trovare un messaggio che produca lo stesso hash indipendentemente dal valore di quest'ultimo.
  - sceglie il messaggio legittimo x_1 ed il messaggio fraudolento x_2
  - altera x_1 e x_2 a una posizione "non-visibile" fino a che $h(x'_1) = h(x'_2)$ (per es. 64 alterazioni di posizione → 264 valori differenti di hash)
  - Se Alice firma x'_1 → $(x'_1, sig_{Kpriv}(h(x'_1)))$
  - Oscar sostituisce x'_1 → x'_2 e $(x'_2, sig_{Kpriv}(h(x'_2)))$

- **La complessità degli ultimi due e tipi di attacco è molto diversa**

hash di m bit

Attacco a forza bruta semplice 2^m $2^{m/2}$ Attacco del compleanno

ALGORITMI CON FUNZIONI DI HASH

- **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)
  Varie versioni:
  - RIPEMD-128
  - RIPEMD-160
  - RIPEMD-256
  - RIPEMD-320

FUNZIONI DI HASH

- La lunghezza dei valori di *hash* varia a seconda degli algoritmi.
- Quelli a 128 bit sono i più comuni e i preferiti
- Una funzione *hash* a 4 bit non è di alcuna utilità per una verifica dell'integrità
- 1/16 dei possibili messaggi sono mappati su uno dei 16 possibili valori di *hash*
- Con un valore di *hash* da 160 bit, l'*hacker* dovrebbe modificare il messaggio in 2^{160} maniere diverse per ottenere il corretto valore di *hash*

Un hash non è né una firma né un MAC, di per sé non garantisce autenticazione e/o integrità

MAC (MESSAGE AUTHENTICATION CODES)

Funzioni di hash + chiavi segrete

***MAC* (Message Authentication Code) o keyed hash function**

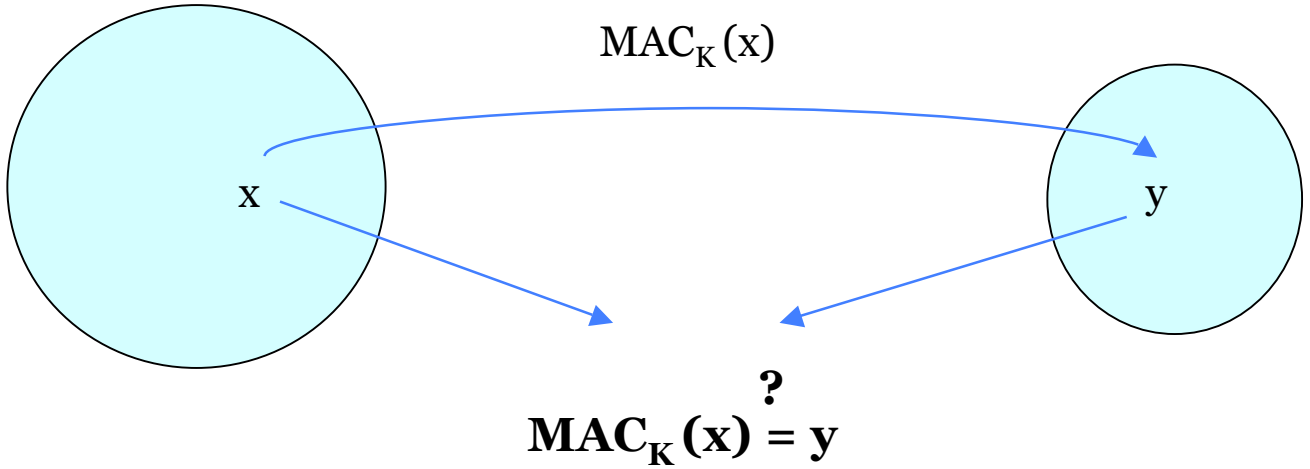
AUTENTICAZIONE **INTEGRITA'**

Proprietà:

- Dati una chiave k e un input x, è facile calcolare $h_k(x)$
- Dato un input x di lunghezza **qualsiasi** in output si ha $h_k(x)$ di lunghezza **prefissata**
- Noto un certo numero di coppie $(x_i, h_k(x_i))$ è computazionalmente impraticabile calcolare qualunque altra coppia $(x, h_k(x))$ per un qualsiasi $x \neq x_i$ senza conoscere k

Generati e verificati con la stessa chiave (simmetrica)

MAC

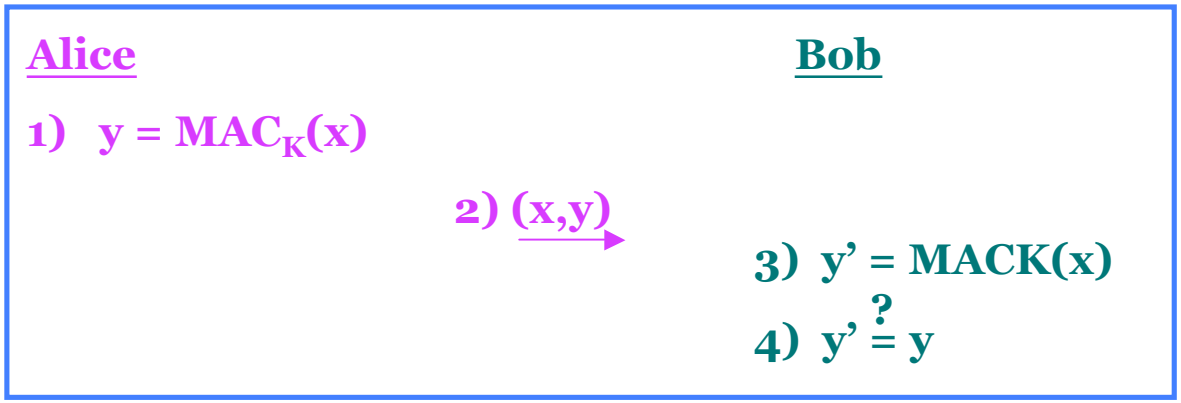


Esempio:

- m = i love you
- k = nazareno (13,0,25,0,17,4,13,14)
- Per calcolare il MAC usiamo un cifrario di Vigenere → MAC = VLNUVCBI
- Se un intrusore modifica il messaggio → m' = i hate you → MAC' = VLNUVCBI
- Quando il destinatario ricalcola il MAC → MAC* = VHZTVCBI
- MAC*≠ MAC' → Il destinatario si accorge dell'intrusore

MAC (APPLICAZIONE)

- Alice applica la funzione MAC sul messaggio x utilizzando la chiave K
- Alice invia a Bob la coppia (x,y) con y è impronta (authentication tag) del messaggio x calcolata dall'algoritmo di autenticazione MAC
- Quando Bob riceve la coppia (x,y) esegue un algoritmo di verifica per accettare o rifiutare il messaggio

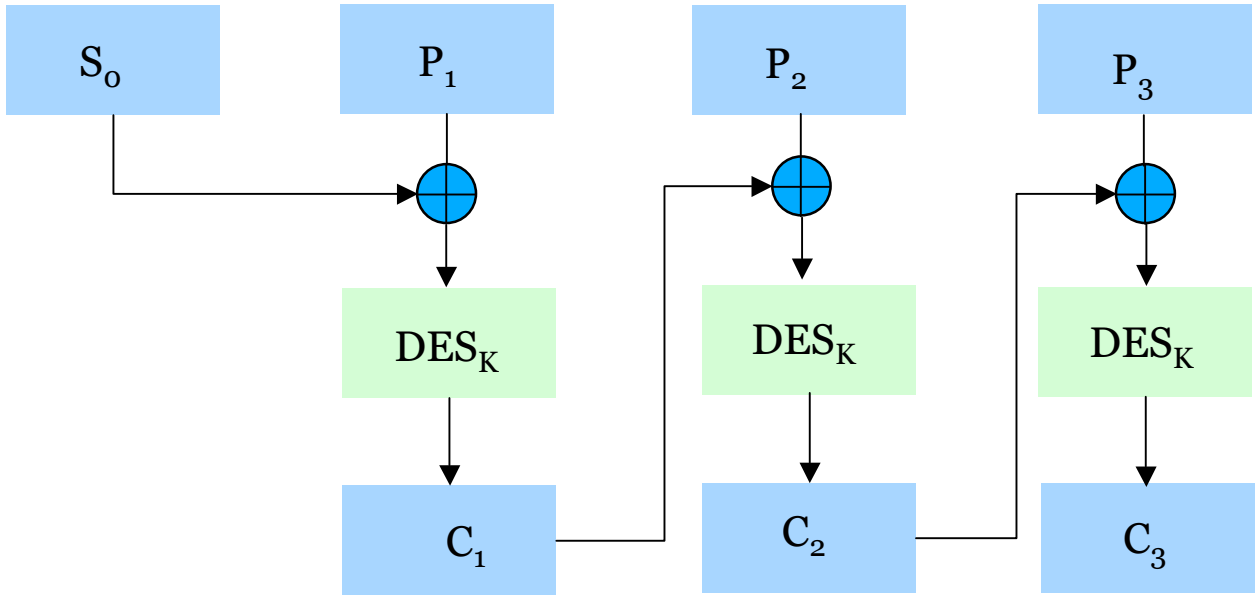


- Genera una firma di lunghezza fissata per un messaggio di lunghezza qualsiasi
- Basato su chiave privata

MAC CON CBC

CBC (Cipher Block Chaining)

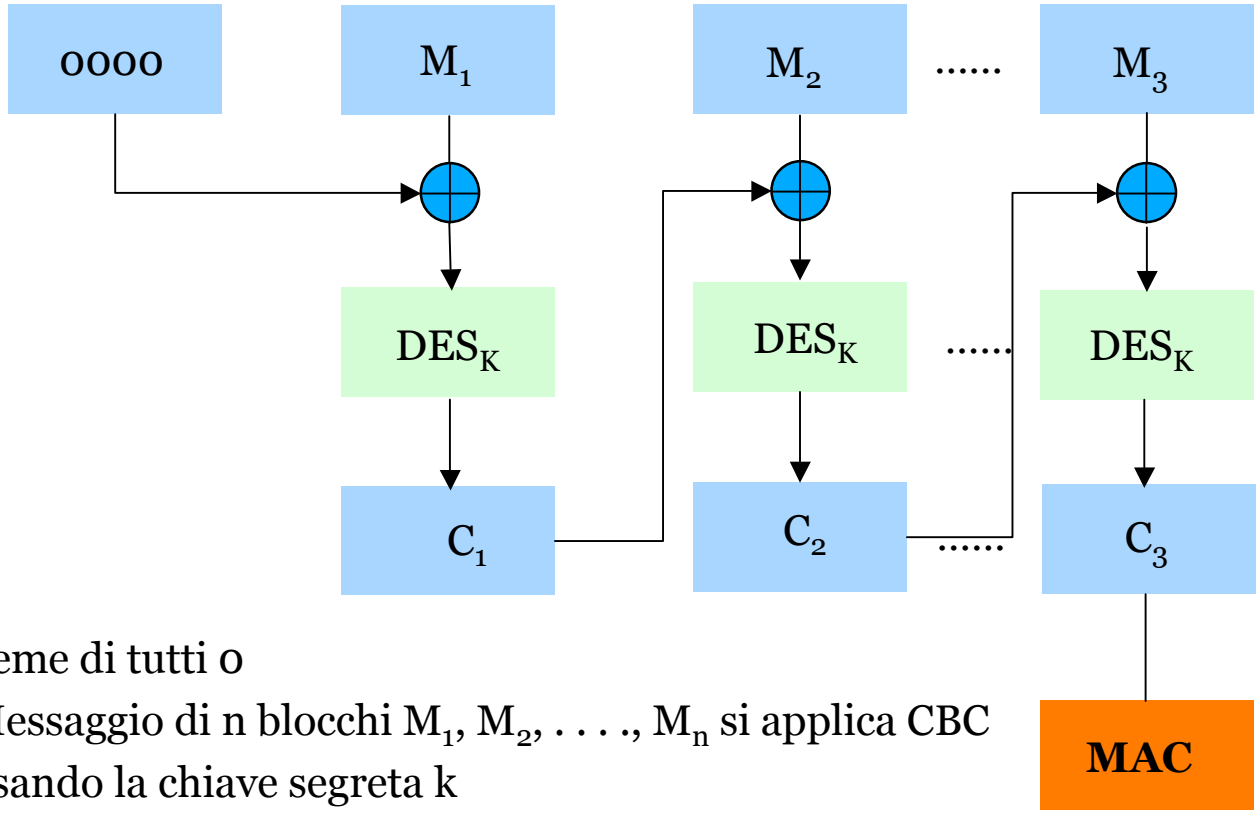
- Il messaggio viene cifrato con un algoritmo simmetrico a blocchi (es. DES-CBC).



- Chaining: collega le informazioni di ogni blocco a quelle dei precedenti

MAC CON CBC

Nella codifica DES-CBC si prende come MAC l'ultima porzione di C (C_n)



- Seme di tutti 0
- Messaggio di n blocchi $M_1, M_2, \ldots, M_n$ si applica CBC usando la chiave segreta k
- Si scartano i primi n-1 blocchi cifrati $C_1, C_2, \ldots$, e si usa C_n
- Si invia $M_1, M_2, \ldots, M_n$ e il tag di autenticazione $MAC_k(M) = C_n$

HMAC

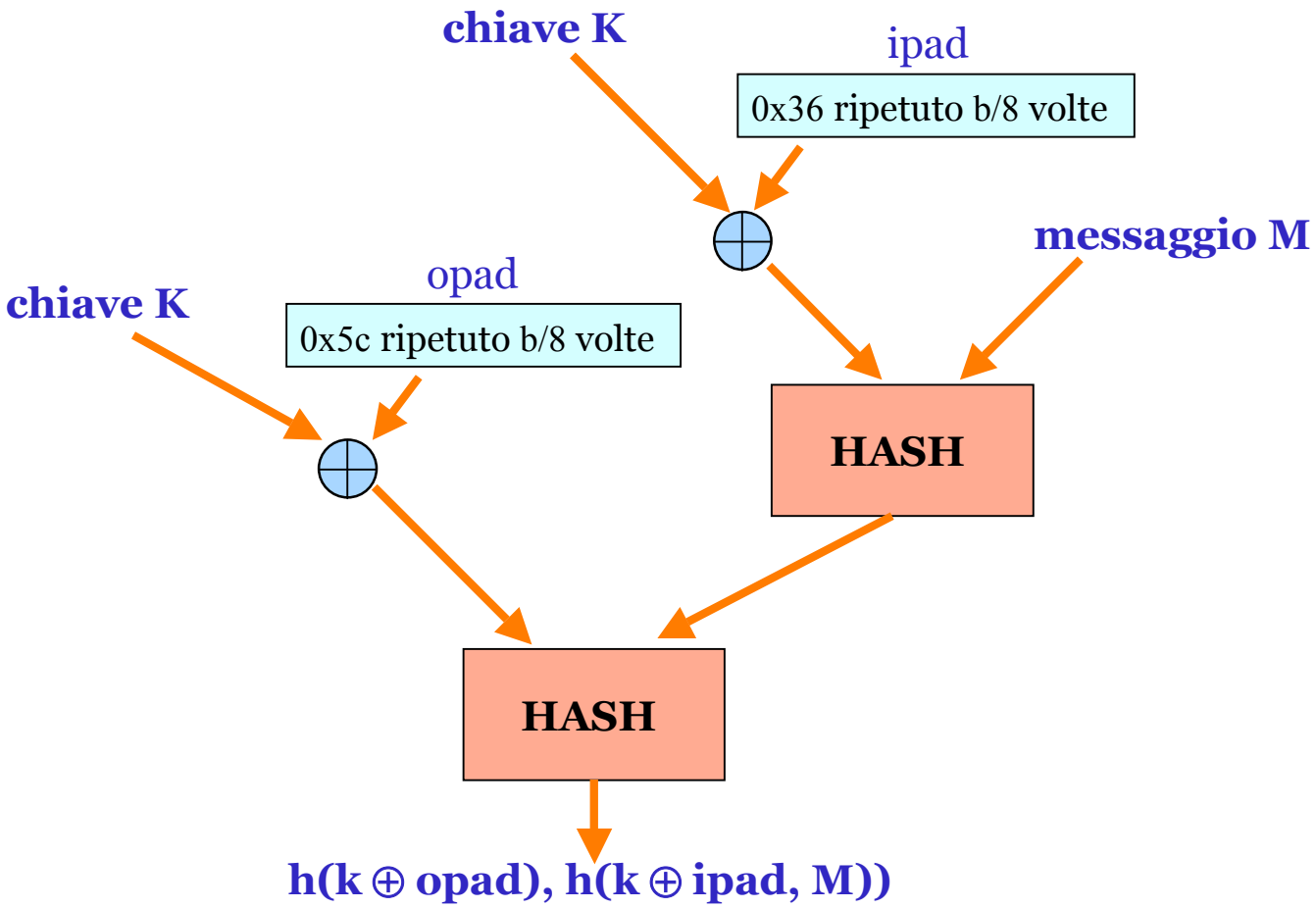
MAC hash function-based

- Una chiave e una funzione hash
- L'HMAC riceve in input un messaggio m, una chiave k ed una funzione hash h, calcola il MAC come

$$\mathbf{HMACK(m,h) = h(k \oplus opad), h(k \oplus ipad, M))}$$

- **opad** = **0x5c** ripetuto b/8 volte (con b dimensione del blocco)
- **ipad** = **0x36** b/8 volte

HMAC


chiave K
ipad
0x36 ripetuto b/8 volte
messaggio M
chiave K
opad
0x5c ripetuto b/8 volte
HASH
HASH
h(k ⊕ opad), h(k ⊕ ipad, M))

HMAC

- Se la chiave ha lunghezza minore di b allora si aggiungono ad essa tanti zeri a sinistra quanti ne servono per arrivare b
- Se la chiave ha lunghezza maggiore di b allora si effettua prima un hash della chiave

L'hash in output a volte viene **troncato** (t bit)

- Un vantaggio perché fornisce meno informazioni a chi attacca
- Uno svantaggio perché l'attaccante ha meno bit da predire

Il numero di bit t deve essere t ≥ b/2 per una funzione hash di b bit.

Più veloce del MAC con CBC

HMAC

Riassumendo:

- Il messaggio è dato in input ad una funzione hash che genera il digest quindi un algoritmo di cifratura cifra il digest con una chiave.
  Il valore ottenuto è aggiunto al messaggio.

- Chi riceve il messaggio ne verifica l'integrità decifrando l'HMAC con l'apposita chiave e confronta il digest ricevuto con quello calcolato sul momento relativamente al messaggio in chiaro ricevuto.

Esempi:

- HMAC-SHA1-80 usa solo i primi 80 bit dei 160
- HMAC-MD5 usa tutti i 128 bit

Entity Authentication

BASI PER L'ENTITY AUTHENTICATION

Le tecniche di Entity Authentication possono essere suddivise in tre categorie, in funzione di come viene garantita la sicurezza:

- **Qualcosa che si conosce**: Password, PIN (Personal Identification Numbers) e chiavi pubbliche e private.
- **Qualcosa che si possiede**: Smart Card (della carte contenenti un microprocessore o un circuito integrato) e generatori di password.
- **Qualcosa che si è**: impronte digitali, riconoscimento dell'iride, della voce, della calligrafia (biometria).

Idealmente, per garantire una corretta autenticazione, dovrebbero essere combinati insieme due o più di questi meccanismi

DIFFERENZE FRA M-A e E-A

- **Real time**: nell'Entity Authentication si rafforza l'identità del chiamante in tempo reale attraverso l'applicazione del protocollo mentre nella Message Authenticatin la verifica dell'autenticazione del messaggio non deve necessariamente avvenire in tempo reale (es. firma digitale) a differenza

- **Messaggio**: l'Entity Authentication non coinvolge alcun messaggio significativo se non l'accertamento di un particolare soggetto, la Message Authenticatin no.

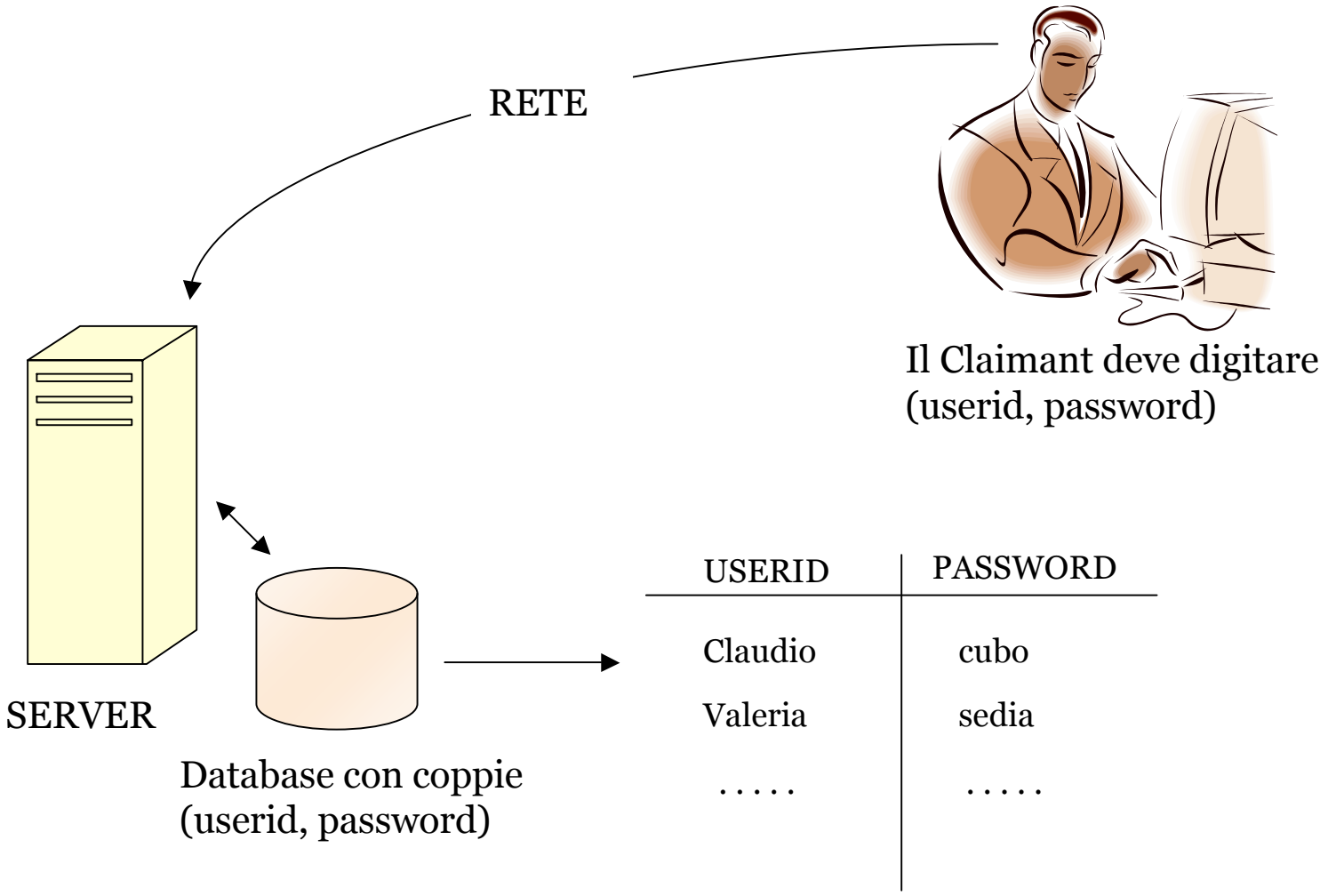
PROPRIETA' DEI PROTOCOLLI

- **Reciprocità dell'autenticazione:** una delle due parti o entrambe devono fornire rispettivamente un'autenticazione unilaterale o mutua.
- **Efficienza computazionale:** il numero di operazioni che il protocollo richiede che eseguire.
- **Efficienza della comunicazione:** il numero di passaggi (cambiamenti del messaggio) e l'ampiezza di banda richiesta.
- **Necessità di una terza parte**
- **Necessità di coinvolgere una terza parte in tempo reale:** es. i servizi per la distribuzione di chiavi pubbliche certificate, supportati da un'autorità di certificazione.
- **Natura della fiducia richiesta dalla terza parte**
- **Quali garanzie di sicurezza sono offerte**
- **Come e dove sono mantenute e conservate le chiavi e/o i segreti** (dischi locali, software, ecc.).

Autenticazione Debole

Paradigma username-password

PARADIGMA USERNAME-PASSWORD (1)



PARADIGMA USERNAME-PASSWORD (2)

- **Idea base:** Si associa ad ogni utente del sistema (entity) una password (fissa).
- La forma di autenticazione è quella basata su " qualcosa che si conosce ".
- Ciascuna password, è tipicamente una stringa formata da 6-8 caratteri che permettono l' identificazione di ciascuna entity e che devono essere ricordati a memoria. Essi costituiscono una sorta di "segreto" tra l' utente ed il sistema (generalmente tenuta in memoria dall' utente ed è contenuta in un file del sistema;
- PROPRIETA':
  - **Non reciprocità**: normalmente gli schemi classici che utilizzano il meccanismo delle password cadono nella categoria delle tecniche a chiave simmetrica e consentono un' identificazione unilaterale;
  - **Bassa complessità**: molto efficienti sia dal punto di vista computazionale che dal punto di vista della comunicazione;

ACCESSO TRAMITE PASSWORD

- Per guadagnare l' accesso ad una risorsa di sistema (i.e. computer account, stampante o applicazioni software), l' utente deve essere in possesso di una coppia (userid, password) da inserire e deve poter specificare, esplicitamente o implicitamente la risorsa a cui accedere.

- Lo userid può essere visto come una rivendicazione di identità mentre la password è la prova a sostegno della rivendicazione.

- Il sistema controlla la corrispondenza username password nel suo database e a seconda dell' esito del controllo garantisce o meno l' accesso alla risorsa.

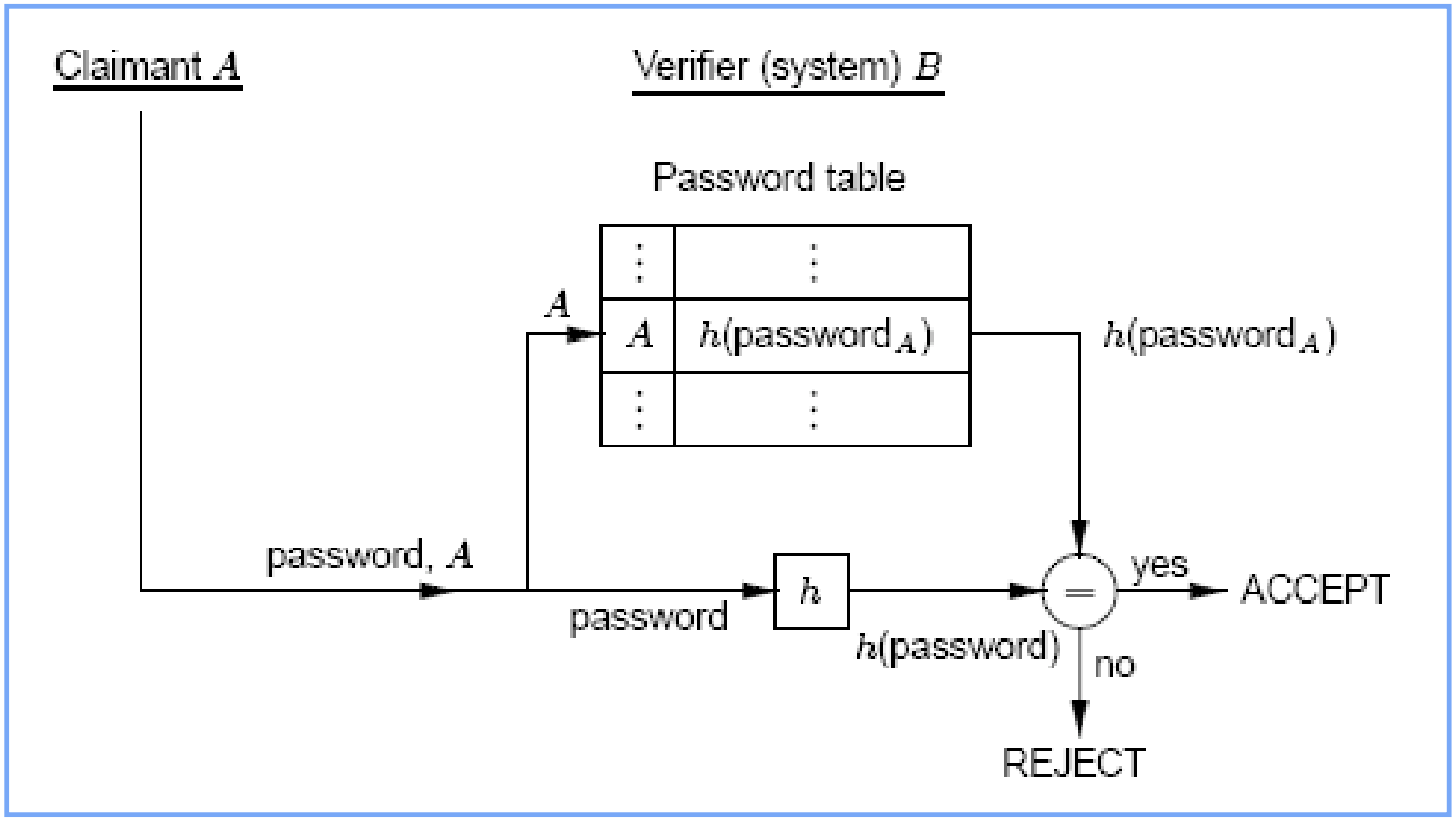
SCHEMI A PASSWORD FISSA

- Gli schemi a password fissa vengono suddivisi in base a:
  - Tipo di informazione a cui si vuole accedere e relativa localizzazione nel sistema;
  - Metodi di verifica.
- Le diverse tecniche di implementazione che vedremo sono :
  - Stored password files;
  - “Encrypted” password files;
  - Password rules;
  - Slowing down the password mapping;
  - Salting passwords;
  - Passphrases.

STORED PASSWORD FILES

- Questa tecnica utilizza files di password in chiaro nel sistema :
  - Il sistema memorizza una lista di coppie (userid, passwords) relative ai diversi users, in un file di sistema protetto in lettura e scrittura.
  - Quando un utente vuole identificarsi al sistema, immette la sua coppia (userid, password).
  - Il sistema, ricevuta la richiesta di accesso, opera un confronto tra quanto inserito e il valore della password memorizzata nel file in corrispondenza dello usid relativo all' utente.
- Tecnica non crittografica (no crittaggio informazioni, no chiavi segrete) .
- SVANTAGGI → non c'è protezione vs amministratori di sistema o superusers
  → tenere una copia di backup del file altrove potrebbe creare problemi per la sicurezza dei dati essendo informazioni in chiaro.
- VANTAGGI → maggiore velocità di accesso alle informazioni

"ENCRYPTED" PASSWORD FILES (1)



“ENCRYPTED” PASSWORD FILES (2)

- Questa tecnica utilizza files di password cifrate :
  - Anziché memorizzare la password dell’ utente in chiaro in un file, si memorizza una funzione one-way della password stessa.
  - Il sistema memorizza nel file di password una coppia (userid, h(password)) dove h(password) è una funzione hash.
  - L’utente che vuole accedere al sistema inserisce la sua coppia (userid, password).
  - Il sistema ricerca lo userid nel file, calcola la funzione hash della password inserita e la confronta con quella memorizzata. Solo se le due funzioni sono uguali l’ accesso viene consentito.

PASSWORD RULES

- Questa tecnica si basa su alcune regole fondamentali che le passwords dovrebbero rispettare. Esse dovrebbero essere :
  - Difficili da indovinare;
  - Facili da ricordare;
- Tuttavia qualcosa di facile da ricordare è anche facilmente indovinabile.
- Poiché esistono algoritmi di ricerca esaustiva e attacchi di tipo dizionario molto efficienti nel caso di scelta di password deboli, alcuni sistemi impongono delle ”regole della password” per scoraggiare l’ uso di questo tipo di password da parte degli utenti.

PASSWORD RULES – vulnerabilità delle pws

- Alcuni aspetti che rendono le passwords vulnerabili dipendono dai criteri di scelta Esse sono infatti generalmente costituite da:
  - parole comuni (in qualsiasi lingua);
  - nomi comuni (nomi di TV, nomi di musicisti, vezzeggiativi, nomi di amici, membri di famiglia o soprannomi);
  - informazioni facili da ottenere (date di nascita, telefono, CF, targa auto);
  - tasti consecutivi di tastiere(qwert);
  - password su altri sistemi;
  - permutazioni di queste lettere (ad es. al contrario);

PASSWORD RULES – possibili soluzioni

- Alcune soluzioni per risolvere il problema di vulnerabilità potrebbero essere:
  - mescolare lettere maiuscole e minuscole;
  - usare qualcosa di impronunciabile;
  - includere caratteri non alfanumerici;
  - usare lettere e numeri;
  - eseguire sostituzioni sistematiche come o con 0 oppure i con 1 ;
  - scegliere lettere da un periodo o da una frase;
  - generare una sequenza casuale con un computer;

PASSWORD RULES - regole principali

- Regole principali:
  - LB (8 - 12 caratteri);
  - Ogni password deve contenere almeno un carattere di ciascun insieme delle categorie (per esempio, maiuscole, minuscole numeri, non alfanumerici);
  - Le password scelte non devono essere presenti in dizionari on-line;
  - Le password non devono essere composte da informazioni relative all' account (userid o sue sottostringhe);

- Un' altra operazione tesa a migliorare la sicurezza delle passwords consiste nel suo "invecchiamento".

Definito il periodo di tempo come limite di una password (es. 30 o 90 giorni), si richiede che ciascuna di queste venga cambiata periodicamente.

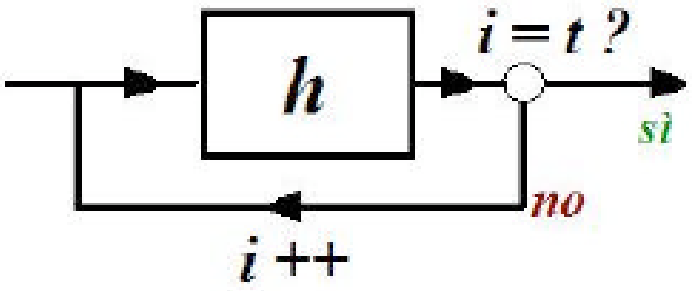
SLOWING DOWN THE PASSWORD MAPPING

- La funzione di verifica della password (ad es. una funzione one-way) può essere resa intensa dal punto di vista computazionale:

ad esempio, iterando una semplice funzione per t volte, con t > 1, con l'output dell' iterazione i usato come input dell' iterazione i + 1;

- SVANTAGGIO : Il numero totale di iterazioni deve essere limitato in modo da non imporre agli utenti un ritardo notevole o inammissibile.

- VANTAGGIO : rallenta e ostacola i tentativi del crittoanalista



SALTING PASSWORDS

- Per rendere gli attacchi dizionario meno efficaci, ogni password nella lista iniziale, può essere aumentata con una stringa random di *s* bit, chiamata "salt", prima di applicarvi la funzione one-way.
- Sia le passwords modificate che i bit di salt sono memorizzati nel file di password
- Quando un utente inserirà una password, il sistema cercherà il salt, lo aggiungerà alla password e applicherà una funzione one-way alla stringa ottenuta.
- La difficoltà di una ricerca esaustiva non aumenta (il salt è memorizzato in chiaro);
- Aumenta invece la complessità di un attacco dizionario che richiede di contenere 2^s variazioni di ciascun tentativo, implicando una grande richiesta di memoria per registrare un dizionario cifrato e anche molto tempo per la sua preparazione.

PASSPHRASES

- Per garantire una maggiore entropia, tenendo presente le capacità di memoria degli utenti (limite), le password possono essere estese in passphrases.

  Gli utenti digitano un detto o una frase anziché una semplice parola

- L' idea è che l' utente possa ricordare più facilmente una frase piuttosto che una stringa di caratteri casuali

- La passphrase viene troncata se supera la lunghezza prefissata ed ha stesso ruolo della password.

- Per evitare che il sistema tronchi la passphrase erroneamente si utilizzano frasi d' ordine

- Inconveniente: Si deve digitare una quantità di testo superiore

SCHEMI A PASSWORD FISSA : ATTACCHI

- **Reply alle passwords fisse**
- **Ricerca esaustiva di password**
- **Password-guessing**
- **Dictionary attack**

ATTACCHI : Reply alle passwords fisse

- Una debolezza dello schema che usa passwords fisse riproponibili è la possibilità che un avversario capisca la password di un utente osservando il modo in cui viene digitata;
- Altro problema è che le passwords (o funzioni one-way) inserite dall' utente sono trasmesse in chiaro sulla linea di comunicazione utente-sistema e siano disponibili sempre in chiaro durante la verifica del sistema;
- Un avversario curioso potrebbe registrare questi dati ed effettuare successive impersonificazioni

Schemi di password fissa sono usati solo su canali di trasmissione affidabili e sicuri dal monitoraggio e non su canali aperti

ATTACCHI : Ricerca esaustiva di password

- Nel caso di **attacchi "on-line":** l' avversario prova le password una alla volta con un sistema verificatore nella speranza di trovare quella corretta.

- Si può rendere più difficile l' attacco:

  - ampliando lo spazio in cui scegliere le password;

  - limitando il numero di tentativi non validi (on-line) consentiti in determinati intervalli di tempo;

  - rallentando il mapping della password;

ATTACCHI : Ricerca esaustiva di password

- Nel caso di **attacchi "off-line":** L' avversario prova ad applicare la funzione one-way ad una stringa alla volta (a caso o sistematicamente), fino a che non ne trova una che combacia con una password crittata nel sistema.

- Il problema è che viene lasciata grossa libertà d' azione al crittoanalista essendo il testo e il mapping one-way entrambe noti;

- Si può rendere più difficile l' attacco:

  - gestendo i dettagli di un mapping one-way;
  - tenendo segreto il file delle password;

- La flessibilità dell'attacco dipende dal numero di passwords che devono essere controllate prima di trovare quella giusta e dal tempo richiesto per testane una;

ATTACCHI : Password-guessing

- Consiste nell' indovinare la password ricercandola in ordine decrescente di probabilità:
  - ad esempio, password corte, sostantivi di uso comune, nomi propri, stringhe in minuscolo, ma soprattutto Nome, cognome e data di nascita.

ATTACCHI : Attacchi dizionario

- In realtà molto simile al precedente;
- Consiste nell' applicare la funzione one-way a tutte le parole di un dizionario, considerando che generalmente un utente utilizza solo un sottoinsieme dell' intero spazio delle password (password brevi, parole del dizionario, nomi propri, stringhe in minuscolo) :
  - Spesso la fatica del crittoanalista è ridotta al minimo grazie ai dizionari on-line;
  - Gli attacchi di questo tipo non hanno successo nel trovare la password di un particolare utente ma trovano molte password nella maggior parte dei sistemi.

ATTACCHI : Attacchi dizionario

→ C ↓ N	**26** (minuscole)	**36** (minuscole alfanumeriche)	**62** (miste alfanumeriche)	**95** (caratteri da tastiera)
5	23.5	25.9	29.8	32.9
6	28.2	31.0	35.7	39.4
7	32.9	36.2	41.7	46.0
8	37.6	41.4	47.6	52.6
9	42.3	46.5	53.6	59.1
10	47.0	51.7	59.5	65.7

Tabella 1: *Il numero di password composte da N caratteri, date C scelte per carattere, è C^N. La tabella fornisce il logaritmo in base 2 del numero di possibili password.*

ATTACCHI : Attacchi dizionario

→ C ↓ N	26 (minuscole)	36 (minuscole alfanumeriche)	62 (miste alfanumeriche)	95 (caratteri da tastiera)
5	0.67 ore	3.4 ore	51 ore	430 ore
6	17 ore	120 ore	130 giorni	4.7 anni
7	19 giorni	180 giorni	22 anni	440 anni
8	1.3 anni	18 anni	1400 anni	42000 anni
9	34 anni	640 anni	86000 anni	$4.0*10^6$ anni
10	890 anni	23000 anni	$5.3*10^6$ anni	$3.8*10^8$ anni

Tabella 2: *Tempo richiesto per effettuare una ricerca nell'intero spazio della password. La Tabella dà il tempo T (in ore, giorni, o anni) richiesto per una ricerca nell'intero spazio usando un singolo processore.* $T = C^N * t * y$, *dove t è il numero di volte che il mapping della password è iterato, e y è il tempo per ogni iterazione, C rappresenta le possibili scelte per i caratteri della password, N è il numero di caratteri che compongono la password. Per* $t = 25$, $y = 1/125000$ *sec. (ciò approssima il comando crypt di UNIX su un PC che esegue il DES a 1.0 MBytes/s).*

Verso l' autenticazione forte: ONE-TIME PWS

- Progresso naturale dagli schemi a password fissa ai protocolli di identificazione challenge-response;
- Ogni password viene usata solo una volta e poi buttata;
- Le one-time password sono di tre tipi:
  - Liste condivise di password one-time;
  - One-time password aggiornate sequenzialmente;
  - Sequenze di one-time password basate su funzioni unidirezionali (schema di Lamport).

LISTE CONDIVISE DI ONE-TIME PASSWORD

- Utente e sistema usano una sequenza o un insieme di t password segrete (ognuna valida per una singola autenticazione), distribuite come una lista pre-condivisa.
- Uno svantaggio è il dover mantenere una lista condivisa.

ONE-TIME PW AGGIORNATE SEQUENZIALMENTE

- Inizialmente solo un'unica password segreta viene condivisa;
- Durante l' autenticazione, usando la password i, l' utente crea e trasmette al sistema una nuova password (i+1) cifrata con una chiave derivata dalla password i.
- La difficoltà del metodo esce fuori se si verifica un crollo della comunicazione.

SCHEMA DI LAMPORT (1)

- Può essere visto come un protocollo challenge-response dove la sfida (challenge) è implicitamente definita dalla posizione corrente delle password nella sequenza (basato su funzioni unidirezionali);

- Sia A un utente che si vuole identificare ad un utente B; l' identificazione avviene estrapolando una one-time password da una sequenza.

1. **SETUP** :
   a) L'utente A inizia con un segreto w. Sia H una funzione one-way.
   b) Viene fissata una costante t, che definisce il max numero di identificazioni da eseguire (il sistema viene poi riavviato con un nuovo w per evitare attacchi di reply).
   c) L' utente A trasferisce $w_0 = H^t(w)$ a B, che inizializza un contatore $i_A=1$;

SCHEMA DI LAMPORT (2)

2. MESSAGGI DEL PROTOCOLLO (INVIATI) :

Durante l' i-esima identificazione (1<i<t), A invia a B un messaggio composto da:

a) Identificatore utente (A);

b) Numero dell'identificazione (i);

c) Password per l' i-esima identificazione $w_i = H^{(t-i)}(w)$

3. COMPUTAZIONI EFFETTUATE :

a) A calcola $w_i = H^{(t-i)}(w)$ (direttamente oppure servendosi di valori intermedi calcolati precedentemente);

b) B controlla che:

- $i=i_A$;
- $H(w_i) = w_{i-1}$

SCHEMA DI LAMPORT (3)

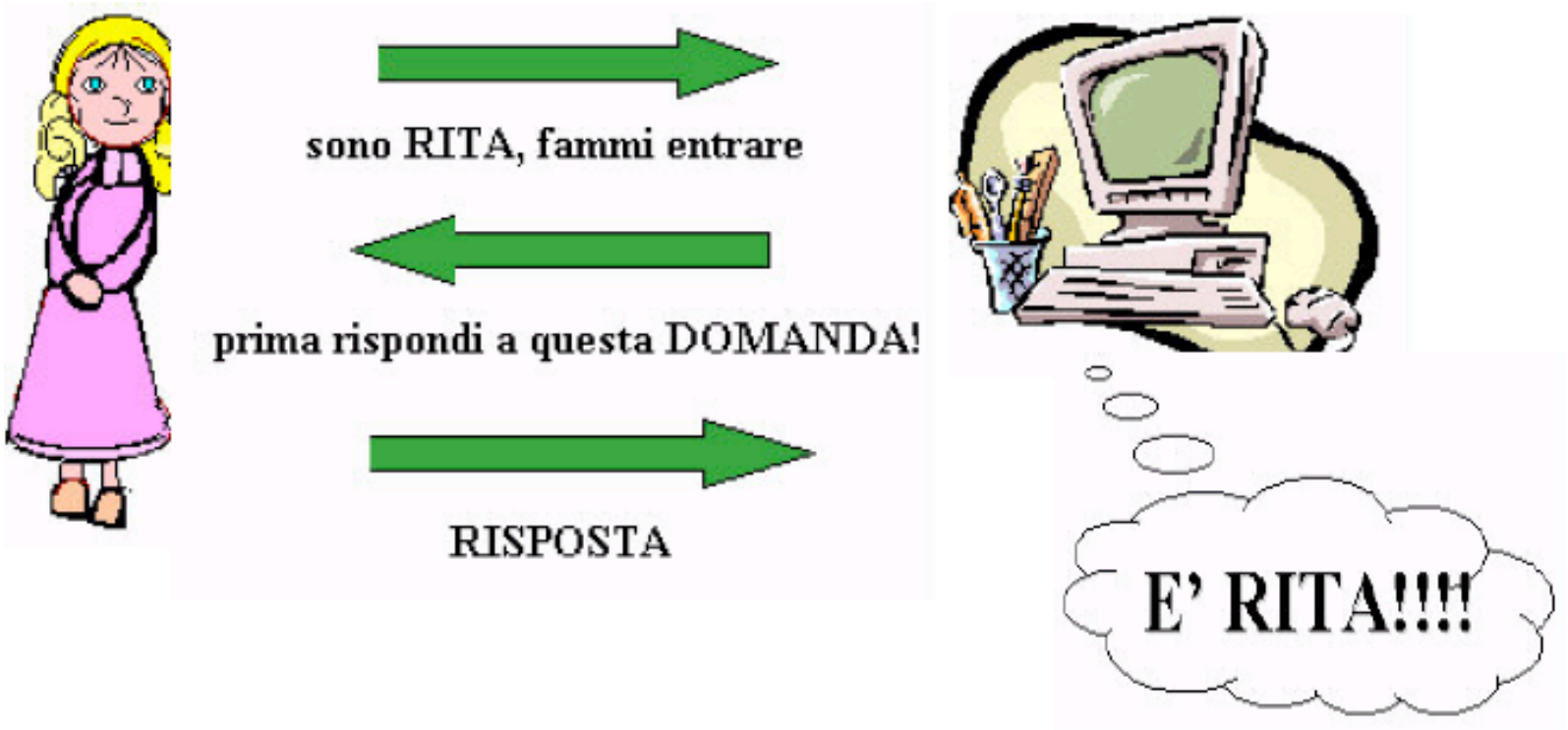
c) Se entrambe i controlli danno esito positivo, B:

- Accetta la password;
- Mette $i_A = i_A + 1$;
- Salva w_i per la prossima verifica di sessione.

Autenticazione Forte

Paradigma challenge-response

PARADIGMA CHALLENGE-RESPONSE



PARADIGMA CHALLENGE-RESPONSE

- In un ambiente di rete, il problema principale è la trasmissione della password in modo sicuro al server.
- I sistemi challenge-response sono stati sviluppati per evitare di rivelare la password sulla rete.
- Nella forma più semplice, di volta in volta:
  - Il Server invia un challenge all' utente, tipicamente una stringa casuale di byte;
  - L'utente calcola una risposta (response), di solito attraverso una funzione che prende in input il challenge e una password.
- Anche se un intruso cattura una coppia (challenge, response), non può violare il sistema perché i challenge successivi sono diversi e quindi anche i rispettivi response

PARADIGMA CHALLENGE-RESPONSE

- Il Claimant “prova” la sua identità al Verifier dimostrando di conoscere un segreto da loro condiviso, senza però rivelarlo durante l’ esecuzione del protocollo.

- Si fornisce di volta in volta una risposta ad un “time-variant” challenge inviato dal Verifier. Tale risposta dipende sia dal segreto che dal challenge.

- Il challenge è tipicamente un numero scelto dal server in modo random e segreto all’ inizio del protocollo e modificato ad ogni conseguente identificazione.

PARAMETRI TIME-VARIANT

- Esistono tre diversi tipi di parametri time-variant:
  - Random Numbers (nonce);
  - Sequence Numbers (serial numbers o counter values);
  - Timestamps (stampe del clock allegate al messaggio originale);
- Servono per :
  - prevenire attacchi : usando combinazioni di random numbers concatenati con timestamps o sequence numbers;
  - garantire tempestività e unicità
    - usando direttamente i random numbers;
    - indirett. con serial number o time clock
  - distinguere istanze : usando random numbers di grandezza sufficiente;
- Sono definiti spesso con i termini: Nonces, unique numbers, non-repeating values

CHALENGE-RESPONSE: TIPOLOGIE

Esistono tre tipi di protocollo challenge-response:

- Basato su tecniche a chiave simmetrica:
  - Simmetric-key encryption;
  - One-way function;
  - Hand-held passcode generator;
- Basato su tecniche a chiave pubblica:
  - Decrittaggio a chiave pubblica con witness;
  - Needham-Schroedr PK modificato;
- Basato sullo schema della firma digitale.

C-R con TECNICHE A CHIAVE SIMMETRICA

- Claimant e Verifier devono condividere una chiave simmetrica;
- Per piccoli sistemi chiusi, con pochi utenti, ciascuna coppia di users può condividere una chiave a priori;
- Per sistemi più grandi, i protocolli di identificazione spesso utilizzano un server di fiducia on-line con il quale ogni parte condivide una chiave; Tale server fornisce una chiave per ogni sessione alle due parti non appena l'una vuole autenticarsi all' altra.

T. Chiave Simmetrica: SIMMETRIC-KEY ENCRYPTION

- Alla base del protocollo di Kerberos e di quello di Needham-Schroeder a chiave condivisa;
- Esistono tre semplici tecniche fondamentali:
  - Autenticazione unilaterale basata sul timestamp;
  - Autenticazione unilaterale usando random numbers;
  - Mutua autenticazione usando random numbers;
- Le tre tecniche :
  - Assumono l' esistenza di una chiave segreta condivisa;
  - Prevedono che il claimant dimostri la sua identità crittando un challenge.

T. Chiave Simmetrica: SIMMETRIC-KEY ENCRYPTION

- Autenticazione unilaterale con timestamp:

$$A \longrightarrow B : \; E_k\,(t_a, B^*)$$

- Autenticazione unilaterale con random numbers:

$$A \longleftarrow B : \; r_B$$

$$A \longrightarrow B : \; E_k\,(r_B, B^*)$$

- Mutua autenticazione con random numbers:

$$A \longleftarrow B : \; r_B$$

$$A \longrightarrow B : \; E_k\,(r_A, r_B, B^*)$$

$$A \longleftarrow B : \; E_k\,(r_B, r_A)$$

T. Chiave Simmetrica: ONE-WAY FUNCTIONS

- L'algoritmo di crittaggio può venir sostituito con una funzione one-way (o non reversibile) della chiave condivisa e del challenge (ad es. con proprietà simili al MAC).

- Questo tipo di algoritmo è preferibile in situazioni in cui l' algoritmo di crittaggio è indesiderato (ad es. per via dei costi computazionali) o non disponibile.

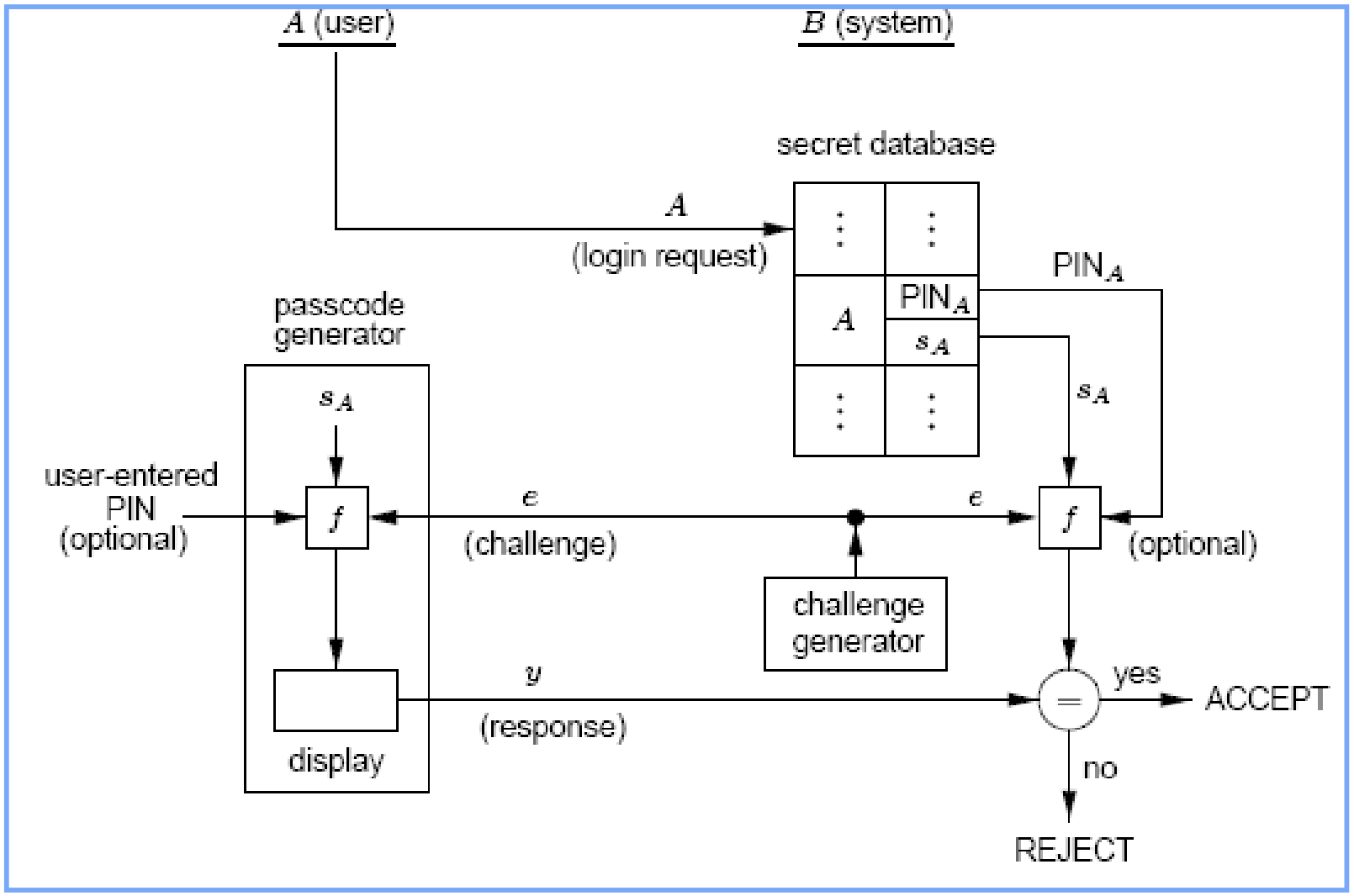
  - La funz. di crittaggio E_k viene rimpiazzata dall' algoritmo MAC h_k;
  - invece che decrittare e verificare la corrispondenza dei campi, il ricevente calcola indipendentemente il valore MAC a partire da quantità note e accetta il messaggio se il MAC del mittente corrisponde con quello calcolato;

$$A \longleftarrow B : r_B$$

$$A \longrightarrow B : r_A , h_k (r_A , r_B, B^*)$$

$$A \longleftarrow B : h_k (r_B , r_A, A)$$

T. Chiave Simm. : HAND HELD PASSCODE GENERATOR



C-R A CHIAVE PUBBLICA

Introduzione

Con il CR a chiave pubblica il client dimostra di conoscere il segreto attraverso la decrittazione di una sfida precedentemente crittata con la propria chiave pubblica.

- Le chiavi pubbliche che vengono utilizzate in un processo di autenticazione dovrebbero essere eliminate perché un eventuale riuso potrebbe compromettere la sicurezza della trasmissione.
- Inoltre il sistema deve prevenire attacchi sul testo cifrato, come l'estrazione di dati da parte di un avversario.
- Una soluzione per risolvere entrambi questi problemi è quella di usare un generatore di numeri random (***Confounder***).Tali dati devono essere disponibili al verifier (verificatore) nel testo in chiaro per permettere la verifica.

C-R A CHIAVE PUBBLICA

$A \leftarrow B : h(r), B, P_A(r, B)$

$A \rightarrow B : r$

Decrittazione a chiave pubblica e testimone

***B*:**

- Sceglie un valore random **r**
- Calcola il testimone **x = h(r)**, dimostrando la conoscenza di **r** senza rivelarlo
- Calcola il challenge **e = P_A (r, B)**

  P_A indica l' algoritmo di crittaggio a chiave pubblica

  h rappresenta una funzione one-way
- Invia il messaggio ad A

C-R A CHIAVE PUBBLICA

A:

- Decritta il challenge **e** per recuperare **r'** e **B'**
- Calcola **x' = h(r')**
- Chiude la comunicazione se **x ≠ x'** (che implica **r ≠ r'**) o se **B' ≠ B**; in caso contrario manda il suo **r'** a **B**

B:

- Procede all'autenticazione di A verificando che l'**r** ricevuto sia uguale a quello inviato

L'uso del testimone preclude la possibilità che si verifichino attacchi al messaggio

C-R A CHIAVE PUBBLICA

A → B : P_B(r1, A)

A ← B : P_A(r1, r2)

A → B : r2

Needham-Schroeder a chiave pubblica modificato

- Questo protocollo fornisce la possibilità di trasportare due chiavi distinte $\mathbf{k_1}$ e $\mathbf{k_2}$, rispettivamente, da A a B e da B ad A per permettere la mutua autenticazione.
- Se la caratteristica di definizione della chiave non è richiesta, $\mathbf{k_1}$ **e** $\mathbf{k_2}$ possono essere omesse.
- Con $\mathbf{P_B}$ denotiamo l'algoritmo di crittazione a chiave pubblica per B
- Con $\mathbf{P_A}$ denotiamo l'algoritmo di crittazione a chiave pubblica per A

C-R BASATO SU FIRMA DIGITALE

- ***Autenticazione unilaterale con timestamps t_A***

  A ⟶ B : $Cert_A$, t_A, B, $S_A(t_A, B)$

- ***Autenticazione unilaterale con numeri random***

  A ⟵ B : r_B
  A ⟶ B : $Cert_A$, r_A, B, $S_A(r_A, r_B, B)$

- ***Mutua autenticazione unilaterale con numeri random***

  A ⟵ B : r_B
  A ⟶ B : $Cert_A$, r_A, B, $S_A(r_A, r_B, B)$
  A ⟵ B : $Cert_B$, r_B, A, $S_B(r_B, r_A, A)$

Protocollo Zero-Knowledge

ZERO-KNOWLEDGE

Problema:

Comunicare una certa informazione è pericoloso perché così anche altri ne vengono a conoscenza.

Soluzione:

Un protocollo di tipo Zero-Knowledge permette all'utente A, un'autenticazione forte senza comunicare l'informazione che possiede.

ZERO-KNOWLEDGE

Basi

- B (sistema di autenticazione) pone ad A (utente) un certo numero di domande.
- Se A conosce il segreto **s** è in grado di rispondere correttamente a tutte le domande.
- L'obbiettivo di A è convincere B di un'asserzione, detta prova (o dimostrazione) di conoscenza (proof of knowledge).

Particolarità

- La tradizionale nozione matematica di dimostrazione è alterata, in questo caso deve essere corretta solo con una limitata probabilità, sebbene possibilmente vicina ad 1.

  Quindi si perde il concetto fondamentale di assolutezza.

ZERO-KNOWLEDGE

Proprietà (1)

- ***Completezza:*** un protocollo interattivo è completo se, dati un utente onesto ed un sistema onesto, il protocollo ha successo con schiacciante probabilità.
- ***Validità:*** un protocollo interattivo è valido se un utente A, che riesce ad autenticarsi, è in possesso dei dati (pubblici e privati) che gli consentono di eseguire il protocollo, allora potrà usare quei dati per autenticarsi in utilizzi successivi del protocollo.
- Se queste due condizioni sono rispettate un utente esterno che vuole impersonare l'utente A deve conoscerne il **segreto s**.

ZERO-KNOWLEDGE

Proprietà (2)

- ***Zero-Knowledge proprietà***: deve esistere un simulatore che è in grado di produrre, senza interagire con il vero prover, copie indistinguibli delle informazioni ottenute interagendo con il vero prover.

- ***Computazionalità***: Se considero un osservatore C che assiste ad una ZK dimostrazione che coinvolge A (prover) e B (verificatore), esso non riceve alcuna informazione utile.

  Allo stesso tempo una registrazione della comunicazione non fornisce alcuna garanzia a C sulla riproducibilità della stessa. Il protocollo raggiunge un livello di *computazionalità* se l'osservatore C non riesce a distinguere le trasmissioni reali da quelle simulate.

  Si dice *computazionalmente perfetto* se le probabilità tra transazioni reali e simulate sono identiche.

ZERO-KNOWLEDGE

Vantaggi

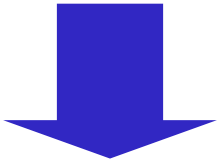
- Il protocollo non presenta alcun degrado dovuto al riutilizzo. Infatti non vengono mai rilasciate informazioni utili.
- Molte tecniche ZK evitano l'uso di algoritmi di crittazione espliciti riuscendo ad essere più leggeri.

Svantaggi

- Costi di esecuzione e memoria usata alti.
- Alcuni protocolli ZK sono basati su assunzioni matematiche non completamente dimostrate, ma solo riconosciute.

ZERO-KNOWLEDGE

Fiat-Shamir



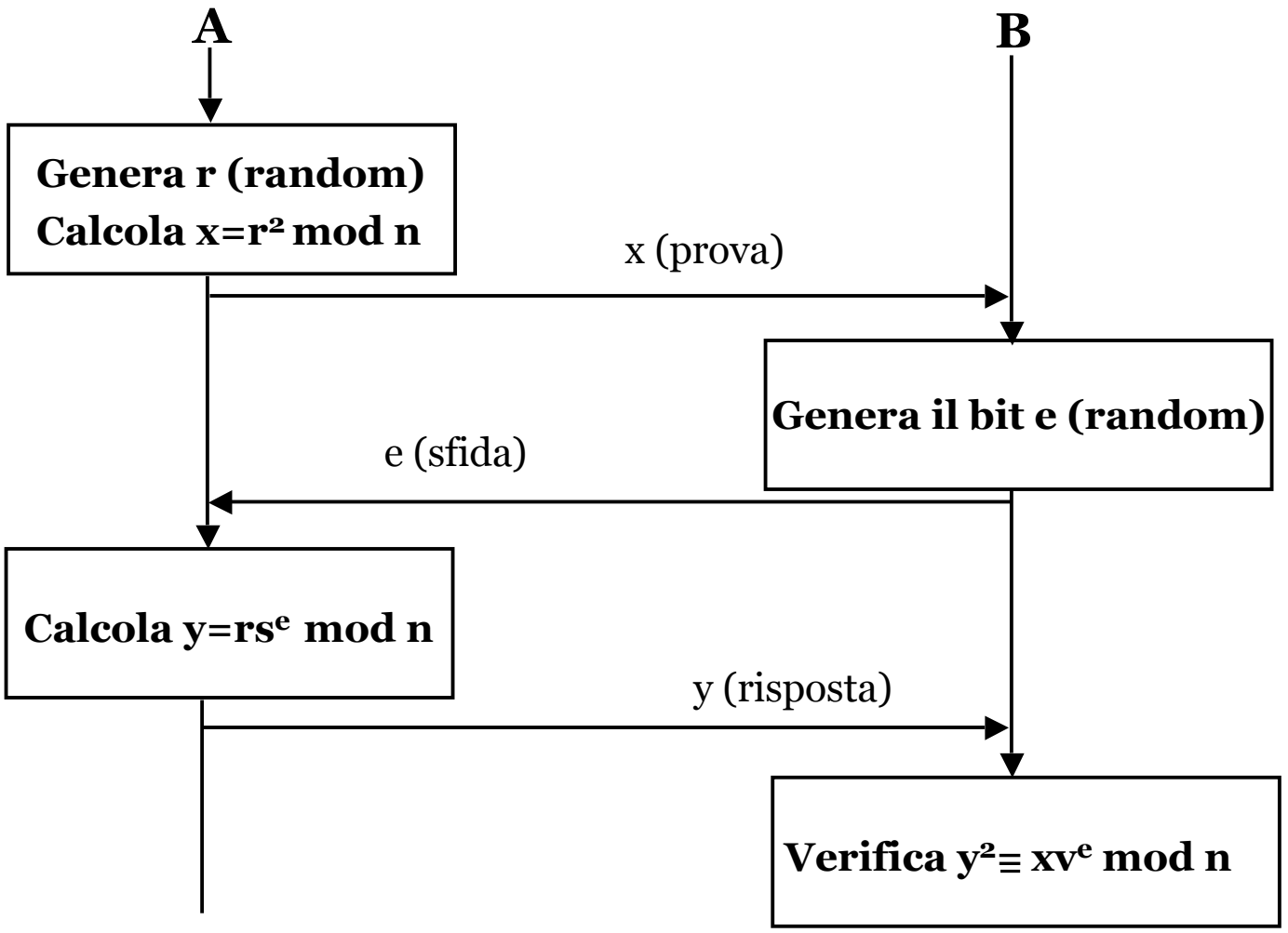
Si basa sull'idea di ZK proof dove B pone una singola sfida per ogni iterazione. La sua sicurezza è ottenuta dalla difficoltà di trovare la radice quadrata di un numero modulo n.

ZERO-KNOWLEDGE

Preparazione

- Una terza parte T genera casualmente i numeri primi **p** e **q** e calcola il modulo **n = p•q**.
- T rende pubblico **n** e mantiene segreti **p** e **q**.
- Ogni client A genera casualmente la chiave privata **s** (o< s < n), coprimo con **n** e calcola **v = s_ mod n** registrandolo presso T come sua chiave pubblica.

ZERO-KNOWLEDGE



ZERO-KNOWLEDGE

- A sceglie un numero r casuale (0< r < n) e calcola **x = r_ mod n**
- A spedisce **x** a B
- B genera casualmente il bit di sfida e=0 oppure e=1
- B manda il bit di sfida ad A
- A calcola **y= r •s^e mod n**
- A spedisce **y** a B
- B verifica la risposta di A con l'uguaglianza **y_= x•v^e mod n**

ZERO-KNOWLEDGE

Attacco di tipo 1

- C (utente esterno) sceglie r (0< r < n)
- Calcola ed invia **r**_
- Riceve il bit e di sfida
- Supererà la prova solo se **e = 0** poiché conosce **r**, mentre per **e = 1** verrà rifiutato in quanto è impossibile conoscere il valore di s a partire da **s**_

ZERO-KNOWLEDGE

Attacco di tipo 2

- C sceglie **x** (0< x < n)
- Calcola ed invia **r_ = x_/s_**
- Riceve il bit **e** di sfida
- Supererà la sfida solo se e = 1 poiché può calcolarsi **x** facendo la radice quadrata di **r•s**, mentre per e = 0 verrà rifiutato in quanto è impossibile conoscere il valore di **r** a partire da **s_**

ZERO-KNOWLEDGE

Conclusioni

Un utente esterno che prova ad autenticarsi senza conoscere il segreto s può rispondere ad una sola delle due domande. Ad ogni esecuzione dell'algoritmo la sua possibilità di autenticarsi si dimezza. Dopo t passi avrà $\mathbf{2^{-t}}$ possibilità.

Al crescere di **t** cresce la sicurezza del sistema.

ZERO-KNOWLEDGE

Altri tipi di protocollo a conoscenza zero:

- ***Feige-Fiat-Shamir:*** si basa sull'idea di ZK proof dove B pone **k** sfide per ogni iterazione. La probabilità che un utente esterno possa autenticarsi è data da $\mathbf{2^{-kt}}$.
- ***Guillou-Quisquater:*** è un estensione di Fiat-Shamir con riduzione del numero di messaggi scambiati e della memoria occupata.
- ***Schnorr:*** è basato sull'intrattabilità del logaritmo discreto.

Kerberos

KERBEROS

Definizione

Kerberos è un protocollo di autenticazione dei servizi di rete creato dal MIT che si serve della crittografia a chiave segreta evitando così la necessità di inviare password attraverso la rete. Autenticare mediante Kerberos impedisce agli utenti non autorizzati di intercettare le password inviate attraverso la rete.

Lo scopo principale di Kerberos è quello di eliminare la trasmissione delle informazioni di autenticazione. Il suo corretto utilizzo permette di ridurre drasticamente la possibilità di intercettazione da parte dei packet sniffer.

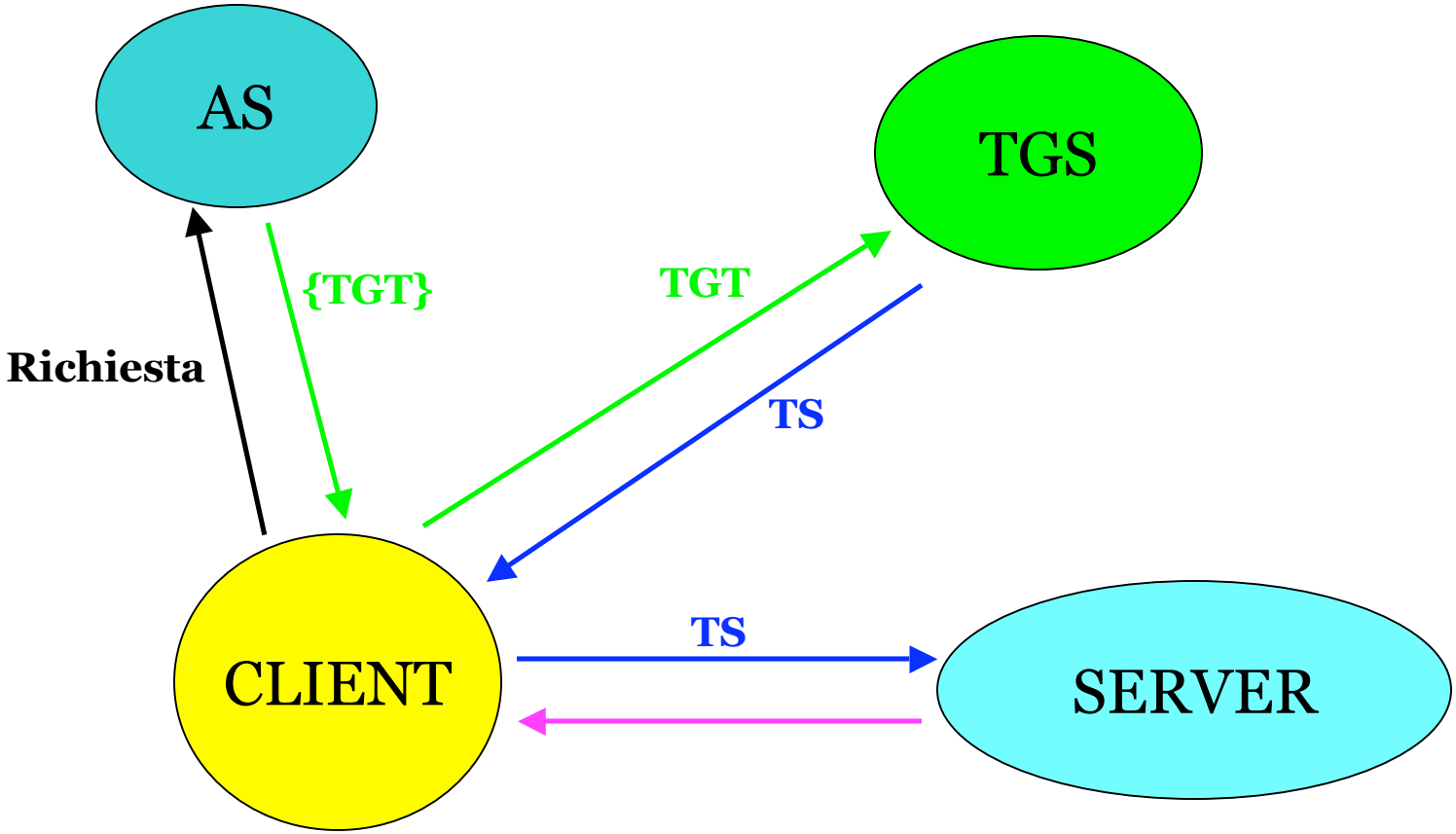
KERBEROS

Funzionamento

Kerberos differisce dagli altri metodi di autenticazione. Al posto di attuare un'autenticazione tra ogni dispositivo del client e ogni server, utilizza la cifratura simmetrica e un sistema fidato di terze parti — noto come Key Distribution Center o KDC — per autenticare gli utenti su una rete e consentire loro di accedere ai servizi desiderati.

Sia gli utenti che i servizi (detti ambedue principal) possiedono una chiave che deve essere conosciuta dal KDC.

KERBEROS



KERBEROS

- Il client fa un'autenticazione iniziale (di bootstrap) con il KDC (o meglio con il suo Autentication Server), che ne registra la chiave, chiedendogli un TGT (ticket granting ticket).
- Il KDC concede tale ticket.
- Il client che vuole utilizzare un servizio presenta il proprio TGT al KDC (o meglio al suo Ticket Granting Server) per ottenere l'accesso al servizio.
- Il KDC invia l' ST (service ticket) al client per quel particolare servizio.
- Il client invia l'ST al server ottenendo l'accesso.

Il TGT ha una durata limitata, di solito, ad 8 ore.

KERBEROS

Vantaggi

- Elimina la trasmissione di informazioni di autenticazione attraverso la rete.
- Login unico per i server appartenenti alla realm.
- Il meccanismo di ticket è utile per connessioni intermittenti (ISDN, portatile).
- Crescente supporto commerciale (Windows 2000).
- Il server fornisce un TGT incorruttibile.

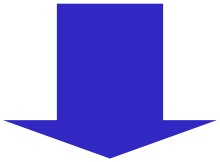
KERBEROS

Svantaggi

- Richiede la frequente sincronizzazione dei clock dei dispositivi sulla rete perché l'avversario potrebbe, modificandoli, riciclare i ticket.
- L'accesso remoto al KDC richiede password in chiaro (non crittata).
- Per avere una rete completamente sicura si dovrebbero "kerberizzare" tutte le applicazioni che inviano password in testo.
- Configurare un'applicazione di rete che può usare Kerberos richiede molta programmazione.
- Poiché immagazzina molte chiavi Kerberos stesso deve risiedere su un server altamente protetto.
- Il server dovrebbe conservare le richieste recenti e controllare eventuali riusi per prevenire il riciclaggio dei tickets durante le 8 ore di vita.

KERBEROS

Versione 5



Quanto detto fino ad ora vale per la versione di Kerberos 4. Alla metà degli anni 90 ne è stata sviluppata anche una versione 5 che presenta differenze e miglioramenti sia dal punto di vista tecnico che dal punto di vista organizzativo.

KERBEROS

4 vs. 5

Le differenze tra le due versioni sono le seguenti:

- la 4 è ristretta ad un unico ambiente
- la 4 usa il DES mentre la 5 non solo
- nella versione 5 i Ticket sono più lunghi ed estesi
- nella versione 5 i byte di messaggio hanno diverso ordine
- la 5 permette autenticazioni tra diversi ambienti
- la 5 si basa sul protocollo Internet
- la 5 è specificata nell' RFC1510 e usata da molti servizi