# Autenticazione Debole

**Claudio Gaz**

# Introduzione

L’identificazione di una persona può avvenire tramite:

- i suoi *attributi.*
- i suoi *possedimenti.*
- il suo *sapere.* 

# Autenticazione debole: le **Password**

- Associata ad ogni utente.
- Tipicamente stringa di 6 ~ 10 o più caratteri da ricordare a memoria.
- E’ (o almeno *dovrebbe essere…*) tenuta nascosta sia dall’utente che dal sistema.

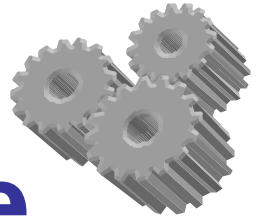
# Accesso tramite password

Per ottenere accesso alla risorsa voluta, l'utente inserisce la coppia:

**(nome utente, password)**

- **nome utente**: affermazione di identità
- **password**: prova che supporta tale affermazione

# Schemi a password fissa: **Tecniche implementative**



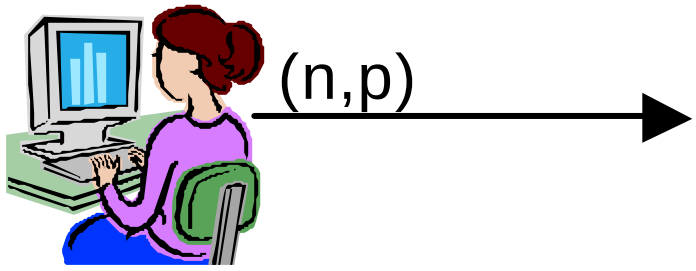
- Si distinguono per:
  - Scelta e immagazzinamento nel sistema,
  - Metodo di verifica.
- Sono:
  1. Uso di file di password in chiaro
  2. Uso di file di password crittate
  3. Regole di scelta
  4. Aumento del tempo computazionale
  5. Uso dei *sali*
  6. Uso di frasi d'ordine (*passphrases*)

# 1. File di password in chiaro

- Una lista di password (in chiaro) è contenuta in un file protetto in lettura e scrittura

\+ velocità di accesso

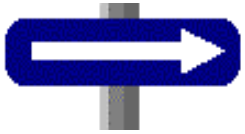
– nessuna protezione contro amministratori di sistema o *superutenti*




```
select password
where nomeutente = n


if (password == p)
    accept;
else reject;
```

# *One-way functions* (Funzioni unidirezionali)

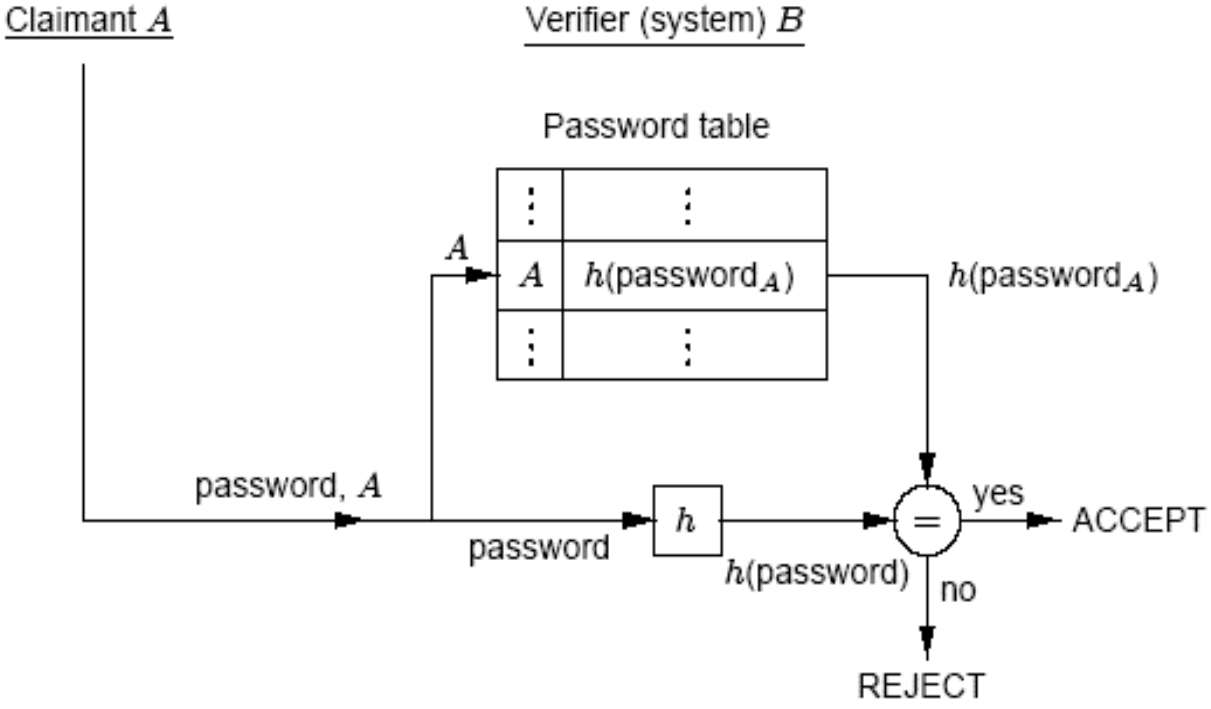


- Servono funzioni speciali non invertibili per impedire al crittoanalista di ricavare la chiave conoscendo il messaggio in chiaro e quello cifrato...
- Queste particolari funzioni sono dette *one-way functions*
- vedi RSA

- NB: questo procedimento non tutela le parole di accesso scelte male! (considera che non è raro che il crittoanalista conosca *f*...)

# 2. File di password crittate

- Una lista di password crittate (secondo una particolare *one-way function h* ) è contenuta in un file protetto in sola scrittura



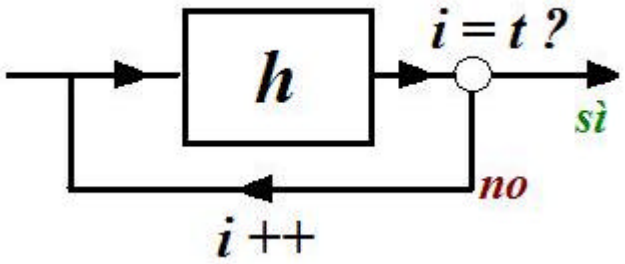
# 3. Regole di scelta (*Password rules)*

- Contro i cosiddetti *dictionary attacks*
- Alcuni sistemi impongono agli utenti la scelta di una password che soddisfi determinati criteri
  - lower bound (8~12 caratteri)
  - almeno un carattere per ogni insieme
  - divieto di utilizzo nella pw dello username
- Si vuole aumentare l'entropia $H(X) = -\Sigma_i \, p_i \log_2 p_i$
- *Password aging*

# 4. Aumento del tempo computazionale

+ rallenta e ostacola i tentativi del crittoanalista

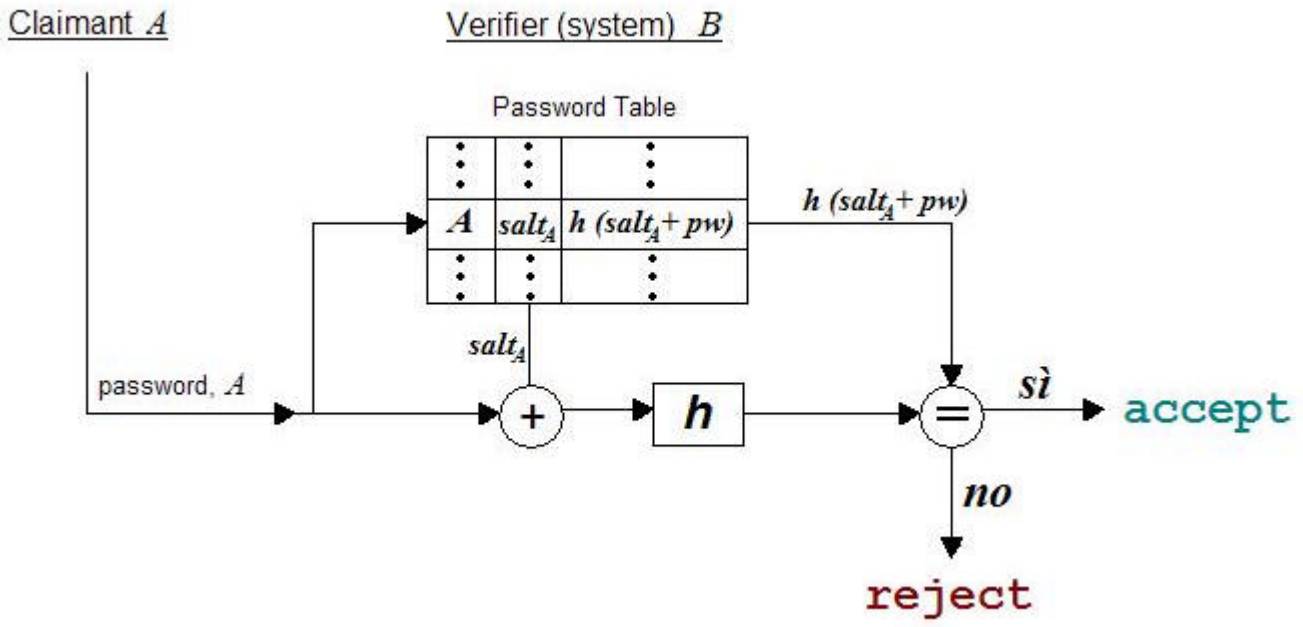
– introduce un leggero ritardo all'utente



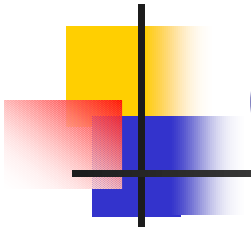
# 5. Uso dei sali (*Salting password*) (1)

- Consiste nell'aumento della pw (subito dopo l'inserimento) di una stringa casuale di *t* bit, chiamata *sale*
- Il sale è memorizzato in chiaro!!! 
- Ad ogni modo, la complessità per un *dictionary attack* aumenta di $2^t$ (dove *t* è il numero dei bit del sale) 

# 5. Uso dei sali (*Salting password*) (2)

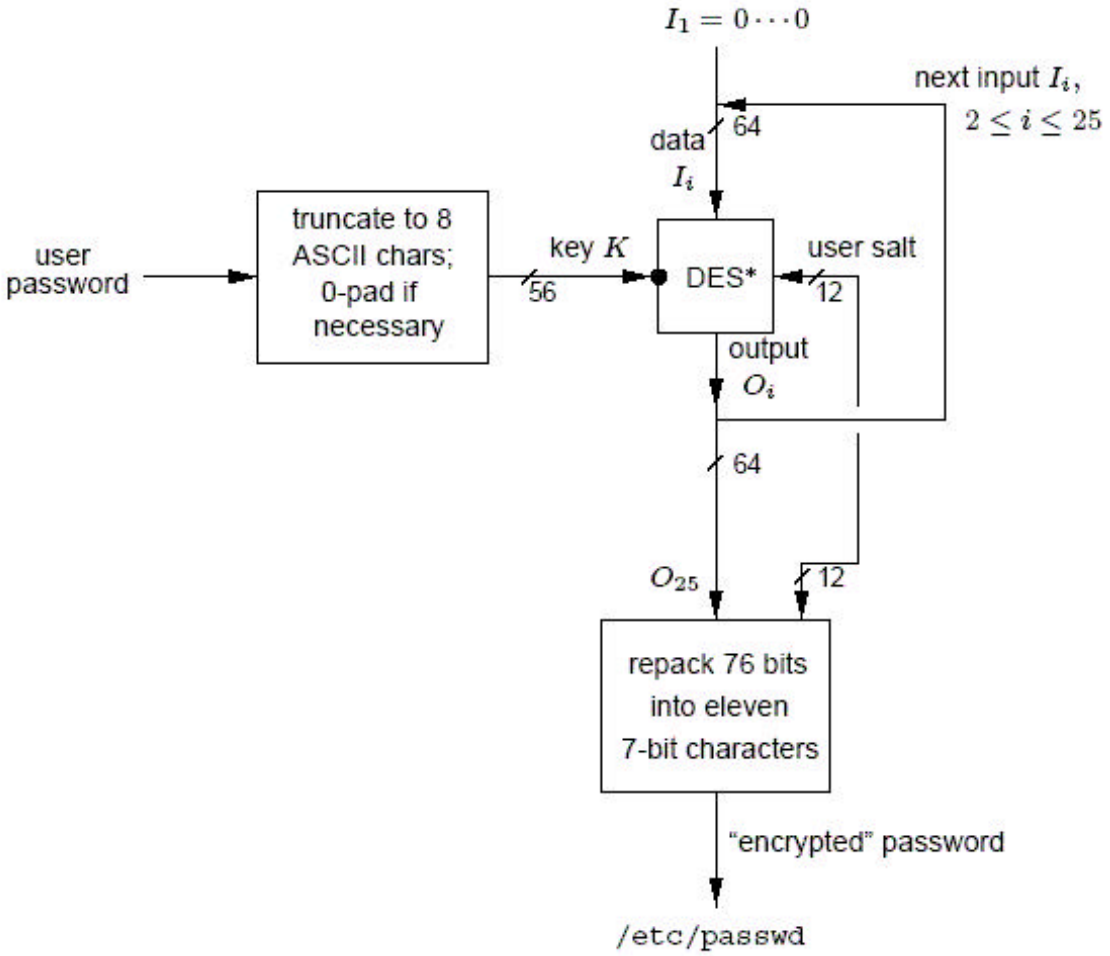


# 6. Frasi d’ordine (*Passphrases*)



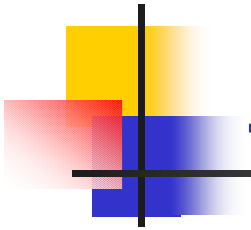
- Limite all’entropia: memoria umana
- Si ovvia a questo limite grazie alle frasi d’ordine
- Vengono troncate dal sistema, ma possiedono comunque altissima entropia

# Esempio: **Password UNIX**



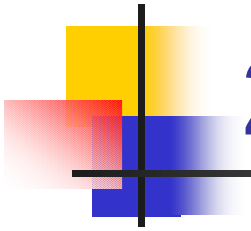
# Schemi a password fissa: **Attacchi**

1. Riuso di password fissa
2. Ricerca esaustiva di password
3. *Password-guessing*
4. *Dictionary attack*



# 1. Riuso di password fissa

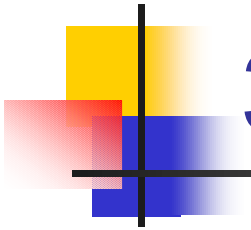
- Il crittoanalista può venire in possesso della password di un utente *A*
  - direttamente dalla fonte (ad es. vede fisicamente scrivere o digitare la pw)
  - impossessandosene lungo il canale (attenzione alle reti *wireless* !!!) mentre essa è in chiaro
  - impossessandosene direttamente dal server, nel breve intervallo di verifica
- A questo punto è facile per il Sig.X impersonificare *A* !



# 2. Ricerca esaustiva di password

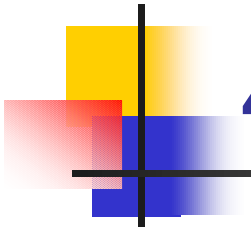
- L’avversario prova ad applicare la funzione unidirezionale a una stringa alla volta (a caso o sistematicamente) fino a che non ne trova una che combacia con una password crittata nel sistema

- Questo genere di attacco può essere vanificato utilizzando le *regole di scelta* o *l’aumento del tempo computazionale* oppure tenendo quanto più possibile segreta la funzione unidirezionale (difficilissimo…)



# 3. *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:
  - quanti di voi utilizzano come password il proprio nome o la propria data di nascita?



# 4. *Dictionary Attack* (1)

- In realtà molto simile al precedente
- Consiste nell'applicare la funzione *h* a tutte le parole di un dizionario:
  - l'esperienza insegna che il più delle volte la fatica è ben ripagata
  - spesso anche la fatica del crittoanalista è ridotta al minimo, grazie ai dizionari on-line
  - del tutto inutile, comunque, se si sta ricercando la pw di un particolare utente

# 4. *Dictionary Attack* (2)

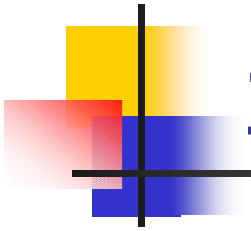
| $\rightarrow c$ <br> $\downarrow n$ | 26 (lowercase) | 36 (lowercase alphanumeric) | 62 (mixed case alphanumeric) | 95 (keyboard characters) |
|---|---|---|---|---|
| 5 | 23.5 | 25.9 | 29.8 | 32.9 |
| 6 | 28.2 | 31.0 | 35.7 | 39.4 |
| 7 | 32.9 | 36.2 | 41.7 | 46.0 |
| 8 | 37.6 | 41.4 | 47.6 | 52.6 |
| 9 | 42.3 | 46.5 | 53.6 | 59.1 |
| 10 | 47.0 | 51.7 | 59.5 | 65.7 |

| $\rightarrow c$ <br> $\downarrow n$ | 26 (lowercase) | 36 (lowercase alphanumeric) | 62 (mixed case alphanumeric) | 95 (keyboard characters) |
|---|---|---|---|---|
| 5 | 0.67 hr | 3.4 hr | 51 hr | 430 hr |
| 6 | 17 hr | 120 hr | 130 dy | 4.7 yr |
| 7 | 19 dy | 180 dy | 22 yr | 440 yr |
| 8 | 1.3 yr | 18 yr | 1400 yr | 42000 yr |
| 9 | 34 yr | 640 yr | 86000 yr | $4.0 \times 10^6$ yr |
| 10 | 890 yr | 23000 yr | $5.3 \times 10^6$ yr | $3.8 \times 10^8$ yr |

# Verso l'autenticazione forte: le ***one-time password***

- schemi a pw fissa → **o.t.pw** → *challenge-response* id.

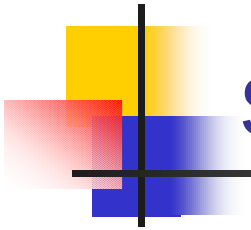
- La password è usata una sola volta, dopodiché viene buttata
- Si distinguono in tre tipi:
  1. Liste condivise di o.t.pw
  2. O.t.pw aggiornate sequenzialmente
  3. Sequenze di o.t.pw basate su funzioni unidirezionali (Schema di Lamport)



# 1. Liste condivise di *one-time pw*

- Utente e Sistema utilizzano una sequenza di *t* password segrete (preesistente) valide ognuna una sola volta
- Attenzione alla sequenzialità! (insieme invece che lista ordinata)
- In alcuni casi possiamo avere una tabella *sfida-e-risposta*

# *One-time-password* aggiornate sequenzialmente



- Inizialmente viene condivisa una sola password segreta
- Durante l'autenticazione che usa la password ***i***, l'utente crea e trasmette al sistema una nuova pw (***i*** + **1**) crittata secondo una chiave derivante da ***i***
- Attenzione alle cadute della connessione!

# 3. Schema di Lamport (1)

- Crea o-t-p basate su funzioni unidirezionali

- Utilizza il **Protocollo di Lamport**:

1. **Setup**
   a) l'utente *A* comincia con una parola segreta *w*. Sia *H* una funzione unidirezionale
   b) viene fissata una costante *t*, che definisce il numero di identificazioni max (poi si ricomincia da una nuova *w*)
   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**
   a) la i-esima identificazione, $1 \leq i \leq t$ , procede come segue:
      - $A \rightarrow B$ : $A, i, w_i \ (= H^{t-i}(w))$
3. **Azioni del Protocollo**
   a) $A$ calcola $w_i = H^{t-i}(w)$ (si può fare o direttamente oppure avvalendosi di valori intermedi calcolati precedentemente)
   b) $B$ controlla che:
      a) $i = i_A$
      b) $w_i = H(w_i) = w_{i-1}$

      Se entrambi i controlli danno esito positivo, $B$ :
      a) accetta la pw
      b) $i_A = i_A + 1$
      c) salva $w_i$ per la prossima verifica di sessione.