Schemi di Autenticazione per il Protocollo HTTP

a cura di
Nicola Ferrante

Challenge and Response

(Sfida e risposta)

- Basic Access Authentication:
  - Ancora oggi utilizzato, ma non troppo sicuro
- Digest Access Authentication:
  - In corso di standardizzazione, più sicuro

Basic Authentication Scheme

- L'utente A fa una richiesta (HTTP Request) al server su una risorsa specifica

```
GET /www.bancaditalia.it/codici_carte_di_credito.html HTTP/1.1
```

- Il server risponde (HTTP Response) ad A dicendogli che NON E' AUTENTICATO

```
HTTP/1.1 401 Unauthorized
...
WWW-Authenticate: Basic realm="InternetSecurity”
...
Corpo del messaggio di risposta
```

Basic Authentication Scheme

```
HTTP/1.1 401 Unauthorized
...
WWW-Authenticate: Basic realm="InternetSecurity“
...
Corpo del messaggio di risposta
```

- Il campo `WWW-Authenticate` deve indicare, oltre allo schema prescelto la sfida che è necessario completare per poter accedere alla risorsa richiesta

- Il valore del Challenge Realm (`realm`) non verrà allegato alla nuova richiesta prodotta dal client A

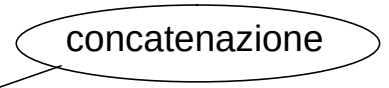
Basic Authentication Scheme

- A questo punto il client A farà una seconda richiesta inviando semplicemente la coppia Username/Password (in chiaro?)
- Se le credenziali sono valide (relative al dominio e alla risorsa richiesta) lo schema di autenticazione è superato con successo!

```
GET /www.bancaditalia.it/codici_carte_di_credito.html HTTP/1.1
...
Authorization: Basic QWxhZGRpbjpvcGVuIHNlc2FtZQ==
...
```

Basic Authentication Scheme

- La stringa che compare dopo la parola chiave `Basic` è ottenuta concatenando quanto digitato dall'utente A e convertito secondo codice radix-64



Es.: Username: Aladin

Password: apriti sesamo

```
Credenziali ="Alladin:apriti sesamo“
Codifica ASCII = 01000001 01101100 01100001 ...etc. etc.
Gruppi di 6-bit = 010000 010110 110001 100001 ...etc. etc.
Codifica Radix-64 = Q W x h ...
```

Vulnerabilità dello Schema di Autenticazione Basic

- Trasmissione in chiaro delle credenziali
- Memorizzazione delle credenziali dell'utente A sul server HTTP
- E' facilmente aggirabile da parte di un hacker che decide di sostituirsi al server HTTP

Digest Authentication Scheme

- L’autenticazione Digest risolve molti problemi di sicurezza (ma non tutti) che non trovavano soluzione con Basic

- Come sempre è l’utente A che invia la richesta HTTP al server

```
GET /www.bancaditalia.it/codici_carte_di_credito.html HTTP/1.1
```

- Questa volta il Challenge di risposta del server è decisamente più complesso di prima

Digest Authentication Scheme

- **qop**: stabilisce la “quality of protection” (RFC 2069)
- **nonce**: è la vera e propria sfida prodotta dal server HTTP (solitamente con algoritmo MD5)
- **opaque**: prodotta dal server e relativa alla specifica risorsa richiesta dal client A

```
HTTP/1.1 401 Unauthorized
...
WWW-Authenticate: Digest
       realm="testrealm@host.com",
       qop="auth-int",
       nonce="dcd98b7102dd2f0e8b11d0f600bfb0c093",
       opaque="5ccc069c403ebaf9f0171e9517f40e41“
 ...
```

Digest Authentication Scheme

- Alla ricezione, l' utente A digita le credenziali
- Il client A produce la risposta alla sfida e richiede nuovamente la risorsa

Es.: Username: Mufasa
Password: Cerchio della vita

```
GET /www.bancaditalia.it/codici_carte_di_credito.html HTTP/1.1
...
Authorization: Digest username="Mufasa",
        realm="testrealm@host.com",
        nonce="dcd98b7102dd2f0e8b11d0f600bfb0c093“,
        uri="/www.bancaditalia.it/codici_carte_di_credito.html ",
        qop=“auth-int”,
        nc=00000001,
        cnonce="0a4f113b",
        response="6629fae49393a05397450978507c4ef1",
        opaque="5ccc069c403ebaf9f0171e9517f40e41“
 ...
```

Digest Authentication Scheme

- **cnonce**: valore casuale. Contro ogni attacco di tipo “chosen plaintext” e/o meccanismo di mutua autenticazione
- **response**: riporta il digest prodotto dal client HTTP (MD5)

```
GET /www.bancaditalia.it/codici_carte_di_credito.html HTTP/1.1
...
Authorization: Digest username="Mufasa",
        realm="testrealm@host.com",
        nonce="dcd98b7102dd2f0e8b11d0f600bfb0c093“,
        qop=“auth-int”,
        opaque="5ccc069c403ebaf9f0171e9517f40e41“
        nc=00000001,
        cnonce="0a4f113b",
        response="6629fae49393a05397450978507c4ef1",  . . .
```

copia della response HTTP del server

Digest Authentication Scheme

- Il parametro **response**: è il risultato della operazione (H è una funzione one-way function)

```
“ H(H(A1) : nonce : nc : cnonce : qop : H(A2)) “
Dove
A1 = username : realm : user-password
A2 = Method : uri : H(body)
```

Vulnerabilità dello schema di autenticazione Digest

- Livello di sicurezza maggiore del Basic…
- … ma assenza di riservatezza
- “Man in the Middle”: il famoso Mister X

Bibliografia

- Berners-Lee, T., Fielding, R., and H. Frystyk, "Hypertext Transfer Protocol -- HTTP/1.0", RFC 1945, May 1996

- Rivest, R., "The MD5 Message-Digest Algorithm", RFC 1321, April 1992

- Franks J., Hallam-Baker P., Hostetler J., Lawrence S., Leach P., Luotonen A., Stewart L., "HTTP Authentication: Basic and Digest Access Authentication." 2 September 1998

MD5

a cura di
Nicola Ferrante

Cos’è il Digest

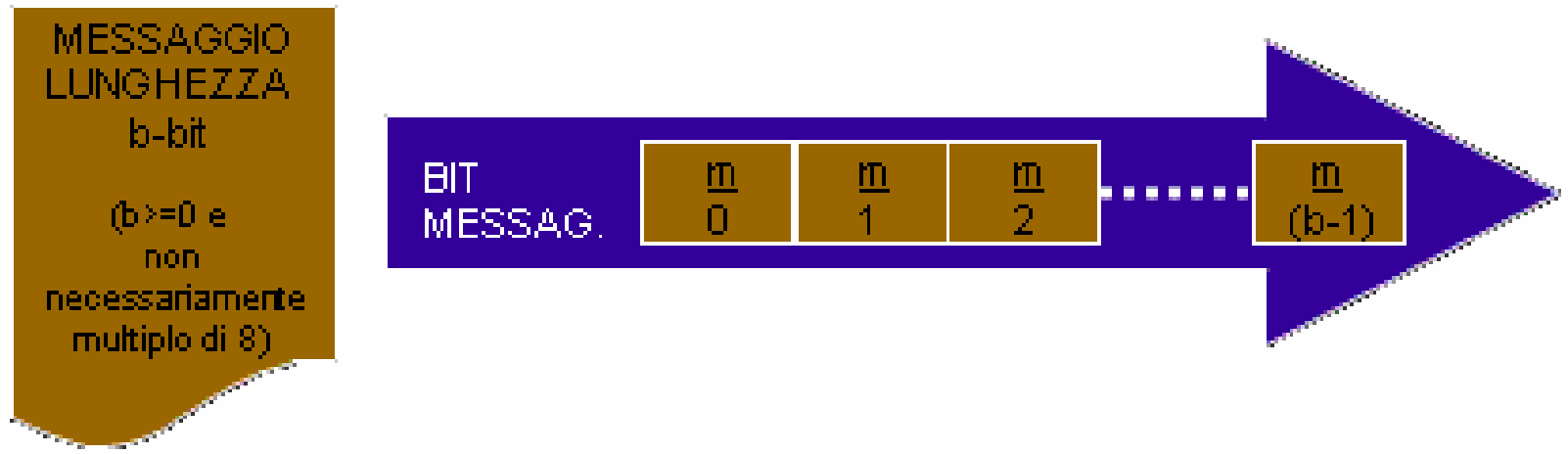
- Il digest (o impronta) di un messaggio M è un numero calcolato a partire da questo messaggio e che ha le seguenti proprietà:
  - ha lunghezza fissa che lo rende facile da manipolare e da trasmettere (128 bit)
  - è assai improbabile che due messaggi diversi abbiano lo stesso digest
  - Non è invertibile, o meglio one-way function

Utilità del Digest

- Verificare l'integrità di un messaggio trasferito
- Autenticazione di un utente con la tecnica del challenge (protocollo HTTP)
- Firma digitale

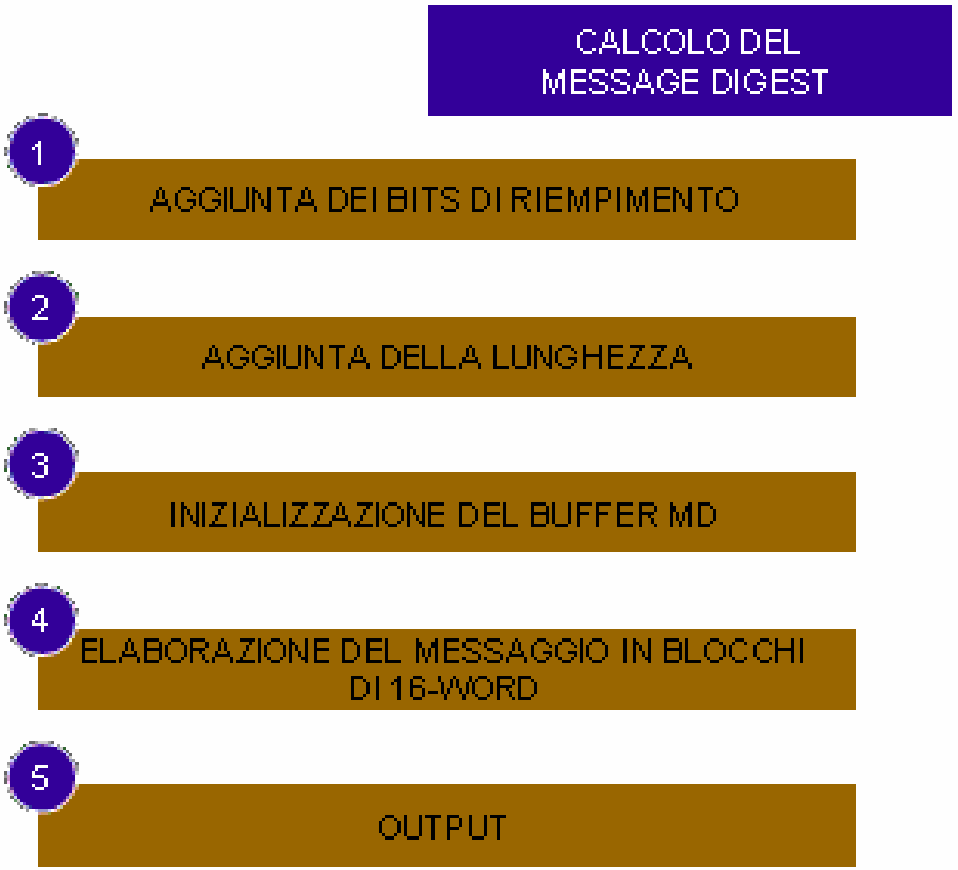
Il Digest con MD5

- In generale



Il Digest con MD5

- L'algoritmo è suddiviso in cinque fasi principali:



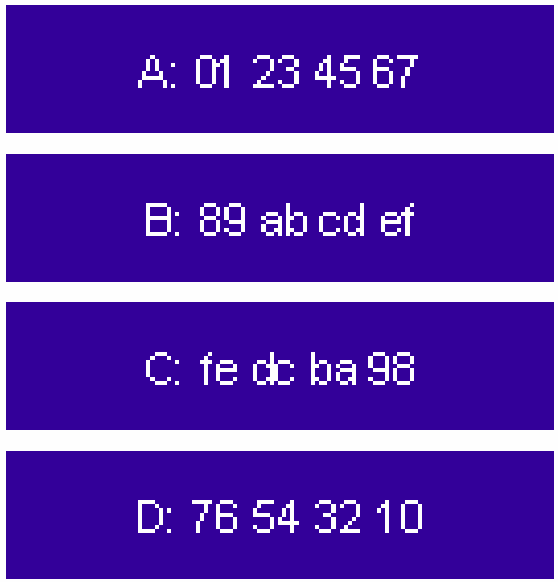
Il Digest con MD5

1. Aggiunta bits di riempimento (padding): così che la sua lunghezza in bits sia congruente a 448 mod 512

2. Aggiunta della lunghezza: viene aggiunta una rappresentazione a 64-bits della lunghezza del messaggio (b) prima del riempimento

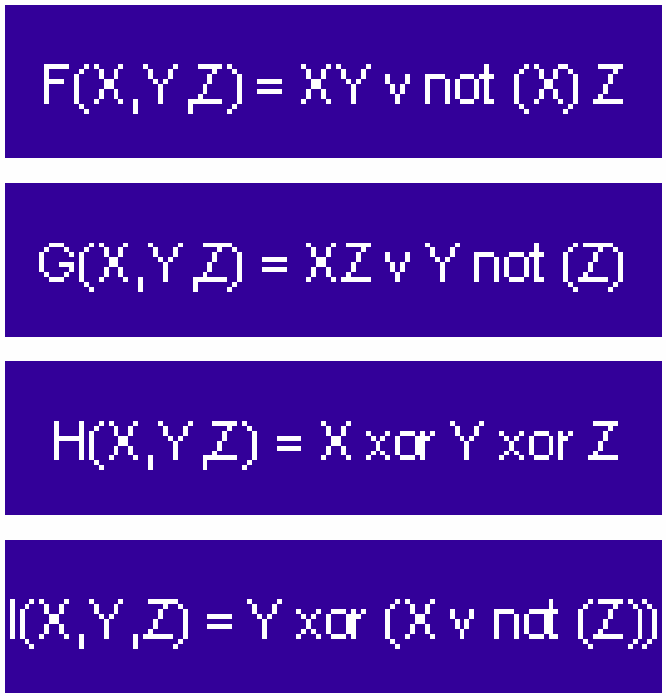
Il Digest con MD5

3. Inizializzazione del buffer MD (*initial variable/chaining variable*): si tratta di un buffer di quattro long-word (A, B, C, D)



Il Digest con MD5

4. Elaborazione del messaggio (*compression function*): vengono definite quattro funzioni ausiliare (F, G, H, I) che ricevono in ingresso tre long-words e producono in uscita una sola long-word:



Il Digest con MD5

```
/* Elabora ogni blocco da 16 word. (256 bit) */
For i = 0 to N/16-1 do

/* Copia il blocco i dentro X. */
For j = 0 to 15 do
     Set X[j] to M[i*16+j]
end /* del ciclo j */

/* Salva A come AA, B come BB, C come CC, e D come DD. */
AA = A
BB = B
CC = C
DD = D

/* Passaggio 1. */
/* [abcd k s i] indica l'operazione:
a = b + ((a + F(b,c,d) + X[k] + T[i]) <<< s). */
/* Effettua le seguenti 16 operazioni. */

[ABCD  0  7  1]  [DABC  1 12  2]  [CDAB  2 17  3]  [BCDA  3 22  4]
[ABCD  4  7  5]  [DABC  5 12  6]  [CDAB  6 17  7]  [BCDA  7 22  8]
[ABCD  8  7  9]  [DABC  9 12 10]  [CDAB 10 17 11]  [BCDA 11 22 12]

[ABCD 12  7 13]  [DABC 13 12 14]  [CDAB 14 17 15]  [BCDA 15 22 16]  ...
```

il valore a 32-bit ottenuto ruotando a sinistra il tutto di s bits

Il Digest con MD5

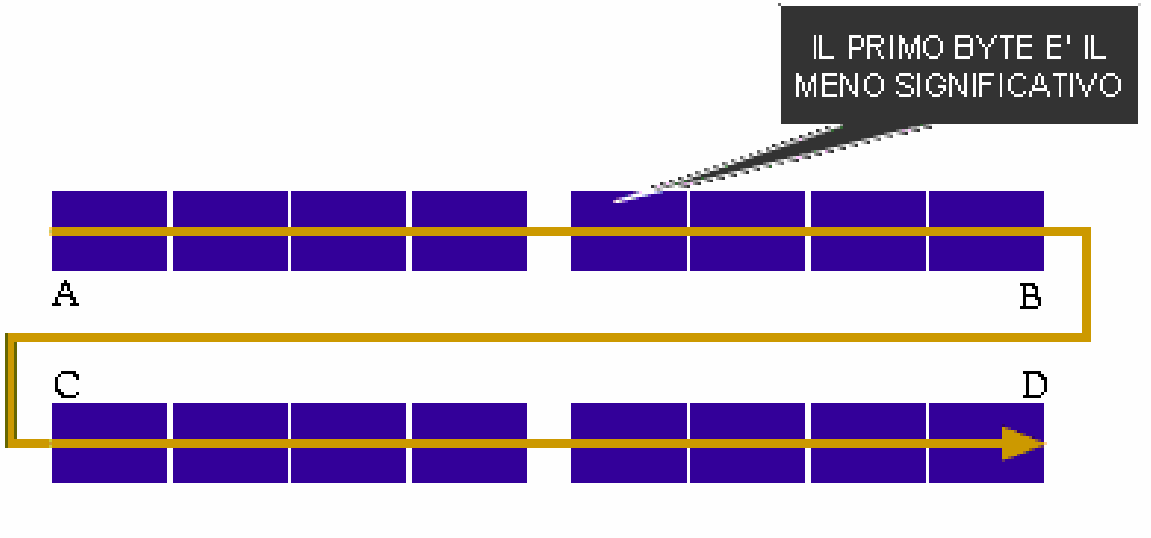
```
/* Passaggio 2. */
/* [abcd k s i] indica l'operazione:
a = b + ((a + G(b,c,d) + X[k] + T[i]) <<< s). */
/* Effettua le seguenti 16 operazioni. */

[ABCD  1  5 17]  [DABC  6  9 18]  [CDAB 11 14 19]  [BCDA  0  20 20]
[ABCD  5  5 21]  [DABC 10  9 22]  [CDAB 15 14 23]  [BCDA  4  20 24]
[ABCD  9  5 25]  [DABC 14  9 26]  [CDAB  3 14 27]  [BCDA  8  20 28]
[ABCD 13  5 29]  [DABC  2  9 30]  [CDAB  7 14 31]  [BCDA 12  20 32]

/* Passaggio 3. */
/* [abcd k s t] indica l'operazione:
a = b + ((a + H(b,c,d) + X[k] + T[i]) <<< s). */
/* Effettua le seguenti 16 operazioni. */

[ABCD  5  4 33]  [DABC  8 11 34]  [CDAB 11 16 35]  [BCDA 14 23 36]
[ABCD  1  4 37]  [DABC  4 11 38]  [CDAB  7 16 39]  [BCDA 10 23 40]
[ABCD 13  4 41]  [DABC  0 11 42]  [CDAB  3 16 43]  [BCDA  6 23 44]
[ABCD  9  4 45]  [DABC 12 11 46]  [CDAB 15 16 47]  [BCDA  2 23 48]

. . .
```

Il Digest con MD5

```
/* Passaggio 4. */
/* [abcd k s t] indica l'operazione:
a = b + ((a + I(b,c,d) + X[k] + T[i]) <<< s). */
/* Effettua le seguenti 16 operazioni. */

[ABCD  0  6  49]  [DABC  7 10  50]  [CDAB 14 15 51]  [BCDA  5  21 52]
[ABCD 12  6  53]  [DABC  3 10  54]  [CDAB 10 15 55]  [BCDA  1  21 56]
[ABCD  8  6  57]  [DABC 15 10  58]  [CDAB  6 15 59]  [BCDA 13  21 60]
[ABCD  4  6  61]  [DABC 11 10  62]  [CDAB  2 15 63]  [BCDA  9  21 64]

/* Quindi esegue le seguenti addizioni. (Ad ognuno dei quattro
 * registri viene aggiunto il valore che aveva prima dell'inizio
 * del processo) */

A = A + AA
B = B + BB
C = C + CC
D = D + DD

end /* fine del ciclo  i */
```

Il Digest con MD5

5. Output: Il message digest è ottenuto partendo dal byte meno significativo di A seguito da quelli di B, C, e terminato con il byte più significativo di D.



Bibliografia

- Rivest R., RFC 1321 "The MD5 Message-Digest Algorithm", The Internet Society, 1992