Funzioni hash

Esercizi con OpenSSL

Alfredo De Santis

Dipartimento di Informatica
Università di Salerno

ads@unisa.it

http://www.dia.unisa.it/professori/ads

Aprile 2017

Esercizi

- Creare due file testuali con contenuto diverso
  - **Esercizio 1** Creare l'hash MD5 dei due file e mostrare il risultato ottenuto
  - **Esercizio 2** Creare l'hash SHA384 dei due file e mostrare il risultato ottenuto
  - **Esercizio 3** Creare l'HMAC dei due file e mostrare il risultato ottenuto

Esercizi

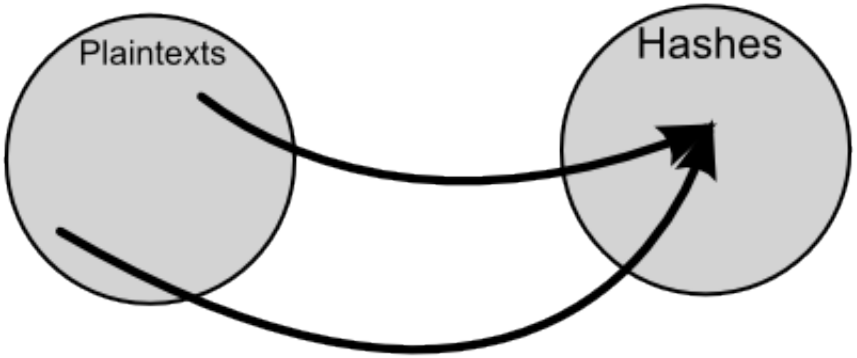
- **Esercizio 4** Creare un file testuale con il seguente contenuto
  - "Corso di Sicurezza, A.A. 2016/2017, Prof. A. De Santis, CdL in Informatica!"
- **Esercizio 5** Calcolare l'MD5 e lo SHA1 di tale file, memorizzando i risultati (i message digest) rispettivamente nei file MD5dgst ed SHA1dgst
- **Esercizio 6** Modificare leggermente il contenuto del file testuale, ad esempio cancellando il simbolo "!" e ricalcolare ciascuna delle due funzioni hash sul file modificato, memorizzando i risultati rispettivamente nei file MD5dgst_new ed SHA1dgst_new
- **Esercizio 7** Aprire i file MD5dgst e MD5dgst_new mediante Bless
  - Notare differenze e similitudini tra i due file
- **Esercizio 8** Ripetere la medesima operazione sui file SHA1dgst e SHA1dgst_new

Esercizi

- Ripetere i seguenti esercizi utilizzando sia MD5 che SHA224
  - **Esercizio 9.1** Creare un file testuale di lunghezza arbitraria
  - **Esercizio 9.2** Generate il valore hash H_1 per questo file, utilizzando una delle due funzioni suddette
  - **Esercizio 9.3** Utilizzando Bless, modificare un bit del file in input
  - **Esercizio 9.4** Generare il valore hash H_2 per il file modificato
  - **Esercizio 9.5** Osservare se H_1 ed H_2 sono simili o meno e descrivere le proprie osservazioni
  - **Esercizio 9.6** Quanti bit sono simili tra H_1 ed H_2?

Collisioni in MD5

- MD5 è stata una delle funzione hash più usate
- Tuttavia è stata dimostrata essere insicura
  - Mediante crittoanalisi sono stati proposti algoritmi efficienti per trovare collisioni
    - Coppie di messaggi che hanno lo stesso valore hash



Collisioni in MD5

(Esempio 1)

- Le prime collisioni furono annunciate il 17 agosto 2004, da Xiaoyun Wang, Dengguo Feng, Xuejia Lai e Hongbo Yu
  - https://eprint.iacr.org/2004/199

Dati i seguenti due messaggi in esadecimale

Messaggio 1 (File1.hex)

```
d131dd02c5e6eec4693d9a0698aff95c2fcab58712467eab4004583eb8fb7f89
55ad340609f4b30283e4888325 71415a085125e8f7cdc99fd91dbdf280373c5b
d8823e3156348f5bae6dacd436c919c6dd53e2b487da03fd02396306d248cda0
e99f33420f577ee8ce54b67080a80d1ec69821bcb6a8839396f9652b6ff72a70
```

Messaggio 2 (File2.hex)

```
d131dd02c5e6eec4693d9a0698aff95c2fcab50712467eab4004583eb8fb7f89
55ad340609f4b30283e4888325f1415a085125e8f7cdc99fd91dbd7280373c5b
d8823e3156348f5bae6dacd436c919c6dd53e23487da03fd02396306d248cda0
e99f33420f577ee8ce54b6708028 0d1ec69821bcb6a8839396f965ab6ff72a70
```


http://borax.polux-hosting.com/madchat/crypto/papers/md5_someday.pdf

Collisioni in MD5
(Esempio 1)

- Le prime collisioni furono annunciate il 17 agosto 2004, da Xiaoyun Wang, Dengguo Feng, Xuejia Lai e Hongbo Yu
  - https://eprint.iacr.org/2004/199

I valori in rosso sono quelli che differiscono tra i due file

due messaggi in esadecimale

Mess

```
d131dd02c5e6eec4693d9a0698aff95c2fcab58712467eab4004583eb8fb7f89
55ad340609f4b30283e4888325 71415a085125e8f7cdc99fd91dbdf280373c5b
d8823e3156348f5bae6dacd436c919c6dd53e2b487da03fd02396306d248cda0
e99f33420f577ee8ce54b67080a80d1ec69821bcb6a8839396f9652b6ff72a70
```

Messaggio 2 (File2.hex)

```
d131dd02c5e6eec4693d9a0698aff95c2fcab50712467eab4004583eb8fb7f89
55ad340609f4b30283e4888325f1415a085125e8f7cdc99fd91dbd7280373c5b
d8823e3156348f5bae6dacd436c919c6dd53e23487da03fd02396306d248cda0
e99f33420f577ee8ce54b6708028 0d1ec69821bcb6a8839396f965ab6ff72a70
```

http://borax.polux-hosting.com/madchat/crypto/papers/md5_someday.pdf

Collisioni in MD5
(Esempio 1)

- Creiamo due *binary file* a partire da tali messaggi

```
xxd -r -p File1.hex > file1
xxd -r -p File2.hex > file2
```

- Mediante il comando **cmp** verifichiamo che **file1** e **file2** siano diversi
- Calcoliamo l'hash MD5 di **file1** e **file2**, rispettivamente
- Notiamo che i due file hanno il medesimo hash MD5

```
$ openssl dgst -md5 file1 file2
MD5(file1)= 79054025255fb1a26e4bc422aef54eb4
MD5(file2)= 79054025255fb1a26e4bc422aef54eb4
```

Collisioni in MD5
(Esempio 2)

- In maniera del tutto analoga, è possibile verificare che i seguenti due messaggi producano lo stesso hash MD5

Messaggio 1 (File1.hex)

```
4dc968ff0ee35c209572d4777b721587d36fa7b21bdc56b74a3dc0783e7b9518afbf
a200a8284bf36e8e4b55b35f427593d849676da0d1555d8360fb5f07fea2
```

Messaggio 2 (File2.hex)

```
4dc968ff0ee35c209572d4777b721587d36fa7b21bdc56b74a3dc0783e7b9518afbf
a202a8284bf36e8e4b55b35f427593d849676da0d1d55d8360fb5f07fea2
```

Single-block collision attack on MD5, Marc Stevens, preprint, 2012
http://eprint.iacr.org/2012/040

Collisioni in MD5

(Esempio 3)

- È possibile trovare altre collisioni mediante uno strumento, chiamato *fastcool*, che permette di generare coppie di file distinti, ma aventi lo stesso hash MD5
- Strumenti necessari
  - *Boost (C++ libraries)*
    - sudo apt-get install libboost-system-dev libboost-program-options-dev libboost-filesystem-dev
  - *fastcoll*
    - http://www.cs.bu.edu/~goldbe/teaching/HW55814/static/fastcoll_v1.0.0.5_patched.zip
    - unzip fastcoll_v1.0.0.5_patched.zip
    - cd fastcoll/
    - make

Collisioni in MD5

(Esempio 3)

```
$ ./fastcoll -o file1 file2
MD5 collision generator v1.5
by Marc Stevens (http://www.win.tue.nl/hashclash/)


Using output filenames: 'file1' and 'file2'
Using initial value: 0123456789abcdeffedcba9876543210


Generating first block: ............
Generating second block: S10.........................
Running time: 3.12363 s
```

Collisioni in MD5

(Esempio 3)

```
$ ./fastcoll -o file1 file2
MD5 collision generator v1.5
by Marc Stevens (http://www.w                    lash/)


Using output filenames: 'file1' and 'file2'
Using initial value: 0123456789abcdeffedcba9876543210


Generating first block: ............
Generating second block: S10.........................
Running time: 3.12363 s
```

File creati da **fastcoll**

Tempo di esecuzione di **fastcoll**

Collisioni in MD5
(Esempio 3)

Mediante il comando **cmp** è possibile verificare se file1 e file2 sono uguali

```
$ cmp file1 file2
file1 file2 differenza: byte 20, riga 1
```

Collisioni in MD5
(Esempio 3)

Mediante il comando **cmp** è possibile verificare se file1 e file2 sono uguali

```
$ cmp file1 file2
file1 file2 differenza: byte 20, riga 1
```

Output restituito dal comando cmp

- Non restituisce alcun output quando i file comparati sono identici
- Quindi, in questo caso, **file1** e **file2** sono diversi
- Il primo byte in cui i due file differiscono è quello alla posizione 20
  (Ciò può essere verificato mediante Bless)

Collisioni in MD5
(Esempio 3)

Visualizzando il contenuto dei due file mediante il comando **xxd** (o Bless) è possibile notare le loro differenze

- I valori in rosso sono quelli che differiscono tra i due file

```
$ xxd file2
00000000: aebf 4de9 7f99 97a9 9ca3 fd5e 4529 7820  ..M........^E)x
00000010: 8bb8 f507 4901 2f1c 0546 421a b08d 25fd  ....I./..FB...%.
00000020: 4bcc f5e1 b8ed 0ea5 4742 6072 1422 c668  K.......GB`r."..h
00000030: d604 49ae 90f8 cc8c 015a b5fe bfeb 70c3  ..I......Z....p.
00000040: 49b9 fece 6aff ef8a 89ab 6e7d 6041 aa82  I...j.....n}`A..
00000050: b3b5 41e3 38db 9218 29b3 20c8 0660 68e8  ..A.8...). ..`h.
00000060: cfcd f253 9f43 a354 7b44 0e9c 0c68 e69f  ...S.C.T{D...h..
00000070: 5093 9259 46fb a668 bfae 0a4f fe7c e774  P..YF..h...O.|.t
$ xxd file1
00000000: aebf 4de9 7f99 97a9 9ca3 fd5e 4529 7820  ..M........^E)x
00000010: 8bb8 f587 4901 2f1c 0546 421a b08d 25fd  ....I./..FB...%.
00000020: 4bcc f5e1 b8ed 0ea5 4742 6072 14a2 c568  K.......GB`r...h
00000030: d604 49ae 90f8 cc8c 015a b57e bfeb 70c3  ..I......Z.~..p.
00000040: 49b9 fece 6aff ef8a 89ab 6e7d 6041 aa82  I...j.....n}`A..
00000050: b3b5 4163 38db 9218 29b3 20c8 0660 68e8  ..Ac8...). ..`h.
00000060: cfcd f253 9f43 a354 7b44 0e9c 0ce8 e69f  ...S.C.T{D......
00000070: 5093 9259 46fb a668 bfae 0acf fe7c e774  P..YF..h.....|.t
```

Collisioni in MD5

(Esempio 3)

La funzione MD5 sui due file restituisce il medesimo risultato, pur essendo i due file diversi tra loro

```
$ openssl dgst –md5 file1 file2
MD5(file1)= 0d60a9451e4e3da182a34033999ebc35
MD5(file2)= 0d60a9451e4e3da182a34033999ebc35
```

Collisioni in MD5
(Esercizio)

- **Esercizio 10.1** Mediante fastcoll, generare un numero arbitrario di collisioni MD5 e valutare i relativi tempi di esecuzione
  - Usare il comando time per rilevare i tempi
- **Esercizio 10.2** Visualizzare ed analizzare il contenuto dei file ottenuti
  - Mediante Bless o il comando **`xxd -p nomeFile`**
- **Esercizio 10.3** Quali sono i relativi hash MD5? Verificare che siano identici
- **Esercizio 10.4** Quali sono i relativi hash SHA-256? Verificare che siano diversi

Esercizi

- **Esercizio 11.1** Creare dei file che abbiano dimensioni (in byte) diverse. A tal fine si può usare il seguente comando
  - `openssl rand -out r.txt <numeroByte>`
- **Esercizio 11.2** Inserire nella tabella i tempi necessari al calcolo dei seguenti message digest (usare il comando **time**)

	100 B	10 kB	1 MB	100 MB	1 GB	10 GB
MD2						
MD5						
SHA1						
RMD160						
SHA256						
HMAC						

Collisioni in SHA (SHA-0)

(Esempio)

Eli Biham et al. "Collisions of SHA-0 and Reduced SHA-1." Annual International Conference on the Theory and Applications of Cryptographic Techniques. Springer Berlin Heidelberg, 2005

Dati i seguenti due messaggi in esadecimale

Messaggio 1 (File1.hex)

```
a766a602 b65cffe7 73bcf258 26b322b3 d01b1a97 2684ef53 3e3b4b7f 53fe3762
24c08e47 e959b2bc 3b519880 b9286568 247d110f 70f5c5e2 b4590ca3 f55f52fe
effd4c8f e68de835 329e603c c51e7f02 545410d1 671d108d f5a4000d cf20a439
4949d72c d14fbb03 45cf3a29 5dcda89f 998f8755 2c9a58b1 bdc38483 5e477185
f96e68be bb0025d2 d2b69edf 21724198 f688b41d eb9b4913 fbe696b5 457ab399
21e1d759 1f89de84 57e8613c 6c9e3b24 2879d4d8 783b2d9c a9935ea5 26a729c0
6edfc501 37e69330 be976012 cc5dfe1c 14c4c68b d1db3ecb 24438a59 a09b5db4
35563e0d 8bdf572f 77b53065 cef31f32 dc9dbaa0 4146261e 9994bd5c d0758e3d
```

Messaggio 2 (File2.hex)

```
a766a602 b65cffe7 73bcf258 26b322b1 d01b1ad7 2684ef51 be3b4b7f d3fe3762
a4c08e45 e959b2fc 3b519880 39286528 a47d110d 70f5c5e0 34590ce3 755f52fc
6ffd4c8d 668de875 329e603e 451e7f02 d45410d1 e71d108d f5a4000d cf20a439
4949d72c d14fbb01 45cf3a69 5dcda89d 198f8755 ac9a58b1 3dc38481 5e4771c5
796e68fe bb0025d0 52b69edd a17241d8 7688b41f 6b9b4911 7be696f5 c57ab399
a1e1d719 9f89de86 57e8613c ec9e3b26 a879d498 783b2d9e 29935ea7 a6a72980
6edfc503 37e69330 3e976010 4c5dfe5c 14c4c689 51db3ecb a4438a59 209b5db4
35563e0d 8bdf572f 77b53065 cef31f30 dc9dbae0 4146261c 1994bd5c 50758e3d
```

Collisioni in SHA (SHA-0)
(Esempio)

- Creiamo due *binary file* a partire da tali messaggi

```
xxd -r -p file1.hex > file1
xxd -r -p file2.hex > file2
```

- Mediante il comando **cmp** verifichiamo che i due file siano diversi

- Calcoliamo lo SHA di file1 e file2, rispettivamente
  - Notiamo che i due file hanno il medesimo SHA

```
$ openssl dgst -sha file1 file2
SHA(file1)= c9f160777d4086fe8095fba58b7e20c228a4006b
SHA(file2)= c9f160777d4086fe8095fba58b7e20c228a4006b
```

Collisioni in SHA-1
(Esempio)

- Miglior attacco: *Marc Stevens, New collision attacks on SHA-1 based on optimal joint local-collision analysis, Eurocrypt 2013*
  - Complessità stimata: $2^{57,5}$
- Partendo da tale attacco, nel Febbraio 2017 il team di ricerca di Google ha individuato una collisione effettiva in SHA-1
  - L'attacco è stato effettuato realizzando un apposito «header» PDF (noto anche come «prefisso» PDF)
  - Obiettivo: creare due PDF con contenuti visivi distinti, ma con valori hash SHA-1 identici

Fonti:
https://security.googleblog.com/2017/02/announcing-first-sha1-collision.html
https://shattered.io/
https://marc-stevens.nl/research/papers/SBKAM17-SHAttered.pdf

Google
SHA1TERED

Collisioni in SHA-1
(Esempio)

- La realizzazione pratica di questo attacco (teorico) ha richiesto un impegno notevole, anche per un'azienda come Google
  - È stato sfruttato il Cloud Computing per calcolare la collisione
    - Uno dei più onerosi calcoli mai completati
    - Ha richiesto la computazione di 9 quintilioni (9×10^{30}) di SHA-1
      - Circa 6500 anni di computazione CPU, oppure 110 anni di computazione GPU se i calcoli fossero avvenuti in parallelo

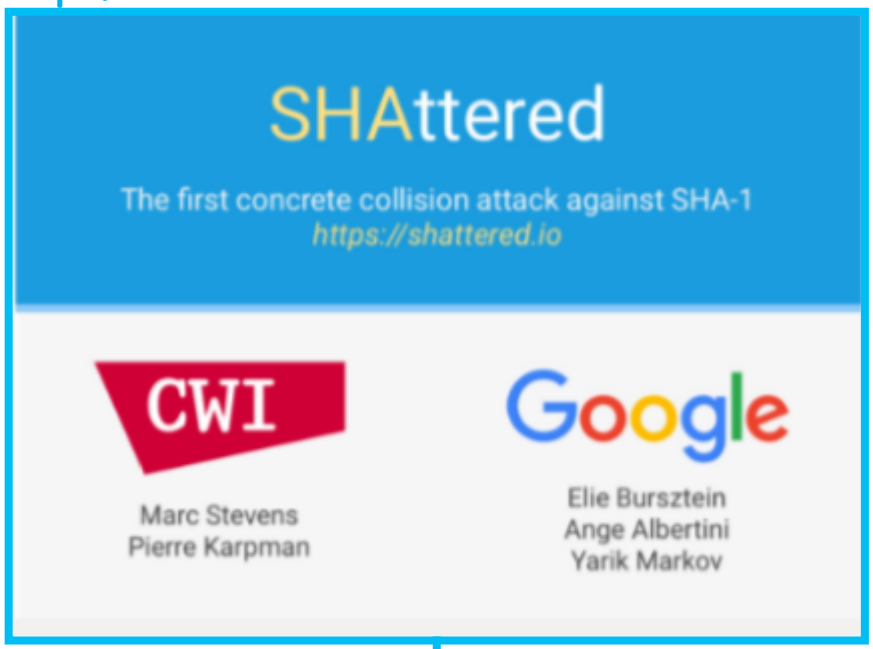
Fonti:
https://security.googleblog.com/2017/02/announcing-first-sha1-collision.html
https://shattered.io/
https://marc-stevens.nl/research/papers/SBKAM17-SHAttered.pdf

Google
SHA1TERED

Collisioni in SHA-1

(Esempio)

1.pdf



2.pdf

Stesso SHA-1

```
38762cf7f55934b34d179ae6a4c80cadccbb7f0a  1.pdf
38762cf7f55934b34d179ae6a4c80cadccbb7f0a  2.pdf
```

Fonti:
https://security.googleblog.com/2017/02/announcing-first-sha1-collision.html
https://shattered.io/
https://marc-stevens.nl/research/papers/SBKAM17-SHAttered.pdf

Google SHATTERED