

Funzioni Hash con OpenSSL

Alfredo De Santis

Dipartimento di Informatica
Università di Salerno

ads@unisa.it



Maggio 2020

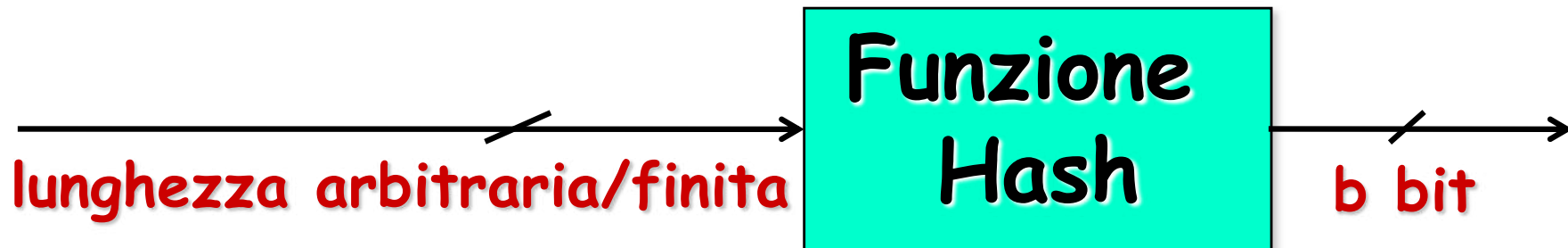
Outline

- Concetti Preliminari
- Funzioni Hash in OpenSSL

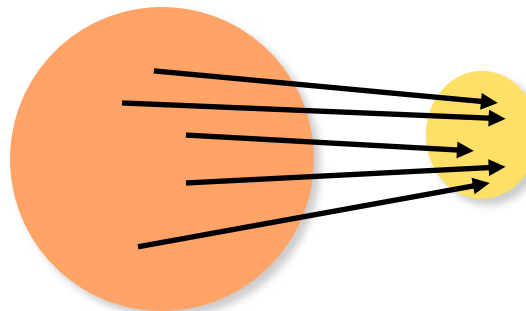
Outline

- Concetti Preliminari
- Funzioni Hash in OpenSSL

Funzioni Hash



- **Idea alla base:** il valore hash $h(M)$ è una rappresentazione non ambigua e non falsificabile del messaggio M
- **Proprietà:** comprime ed è facile da computare



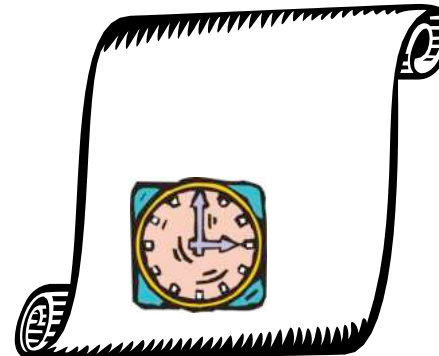
Uso delle Funzioni Hash

Firme Digitali



Integrità dei Dati

Certificazione del Tempo



Firme Digitali e Funzioni Hash

- **Problema:** firma digitale di messaggi lunghi
- **Soluzione naive:** Divisione in blocchi e firma per ogni blocco
 - Problema per la sicurezza: una permutazione/composizione delle firme è una nuova firma
- **Soluzione di uso corrente:** firmare il valore hash del messaggio
 - $[firma\ di\ M] = F_k(h(M))$
- **Vantaggi:** integrità dei dati ed efficienza degli algoritmi



Integrità dei Dati e Funzioni Hash

Tipico uso delle funzioni hash

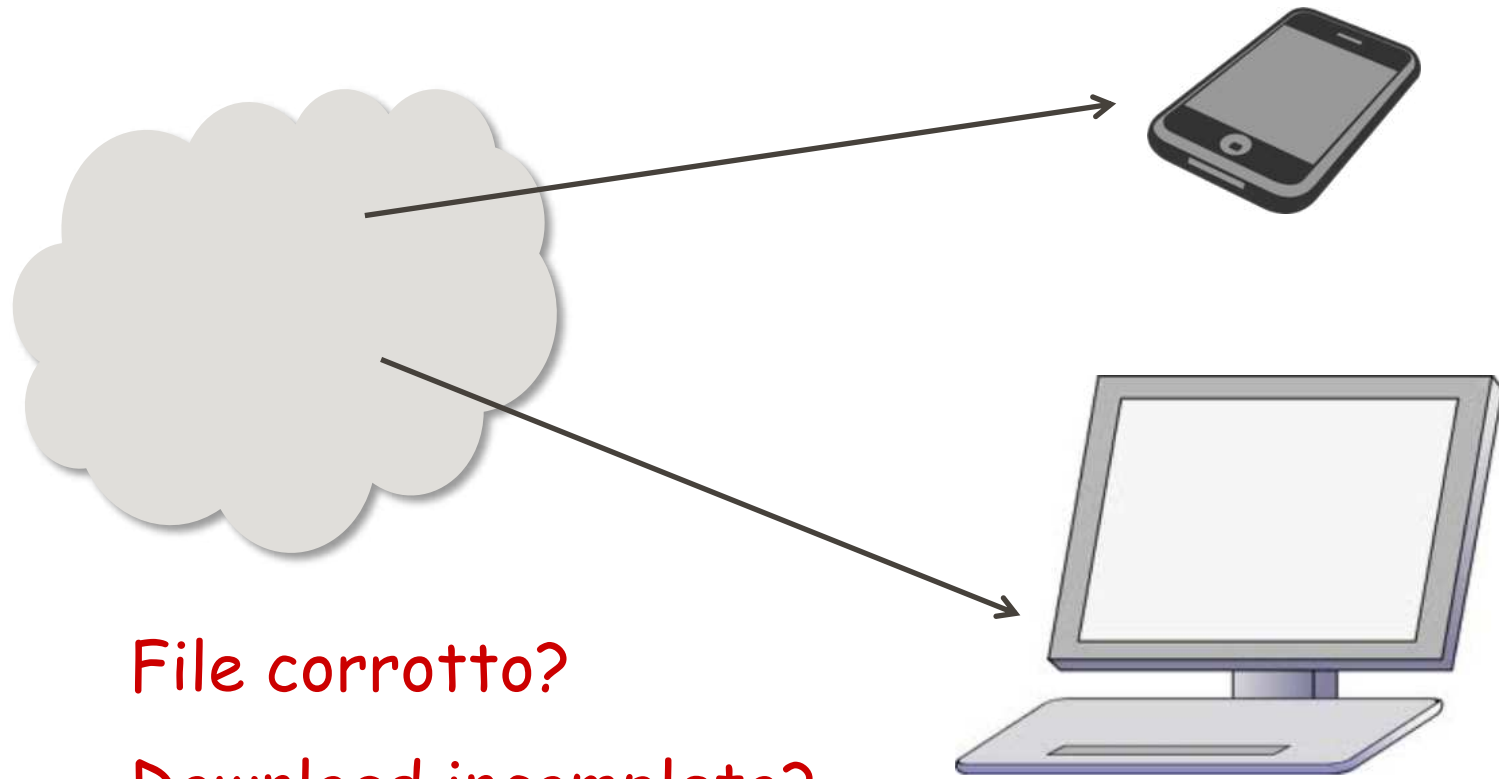
- Computo al tempo T il valore hash del file M
- Conservo $H = h(M)$ in un luogo sicuro
- Per controllare se il file è stato successivamente modificato, calcolo $h(M')$ e verifico se $H = h(M')$
- **$h(M)$ è l'impronta digitale del file**

Assicura se un file è stato modificato!



Integrità dei Dati e Funzioni Hash

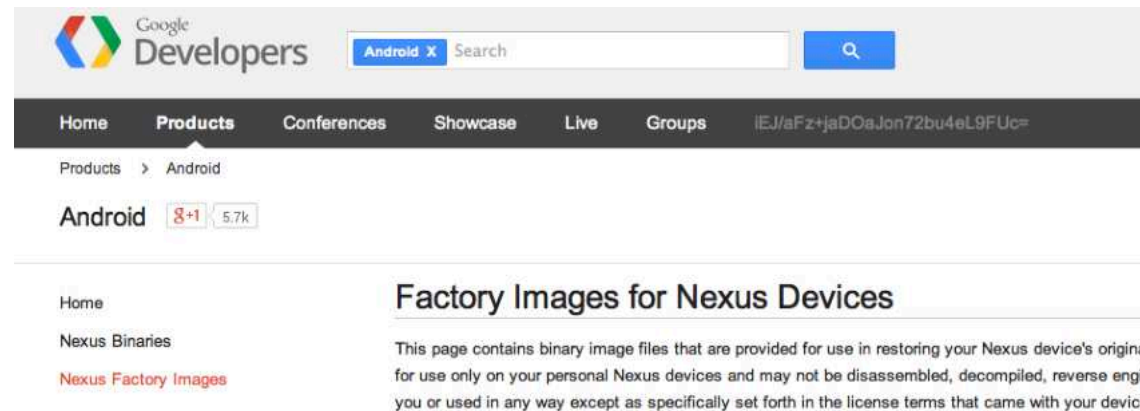
Download file



File corrotto?

Download incompleto?

Integrità dei Dati e Funzioni Hash



The screenshot shows the Google Developers website. The top navigation bar includes links for Home, Products, Conferences, Showcase, Live, and Groups. The 'Products' section is active, and the 'Android' sub-section is selected. Below the navigation bar, there is a search bar and a 'Products > Android' breadcrumb. The main content area is titled 'Factory Images for Nexus Devices'. A sidebar on the left contains links for 'Home', 'Nexus Binaries', and 'Nexus Factory Images'. The main text explains that the page contains binary image files for restoring Nexus devices and includes a disclaimer about the license terms.

Factory Images "razorg" for Nexus 7 [2013] (Mobile)

Version	Download	MD5 Checksum	SHA-1 Checksum
4.3 (JLS36C)	Link	186e8ac3b198276289a5ba3e1569a758	fb03a89feb7c60fb7e31e67c587da3c97c2bf56b
4.3.1 (JLS36I)	Link	344feabad51d3bedb75a6e4d1d451a75	ecb320cdea2fa980d8a944dd82227bb8f89d58fd
4.4 (KRT16S)	Link	20bee0445c67aa45c002a14e4bac1d77	bd6c92418035598c0815d19a2b21066d3f6fe9b6
4.4.2 (KOT49H)	Link	8e42ffe324a9109031dd4f9dd53fdc9a	49789b240cab9a8e91c1f5a3eb0b6be0b07b5732
4.4.2_r2 (Verizon) (KVT49L)	Link	2efd400254e657d0b72e698daff7ba4d	65bdbe0a9288802998ac8834341fa2b7a0ecb9b7

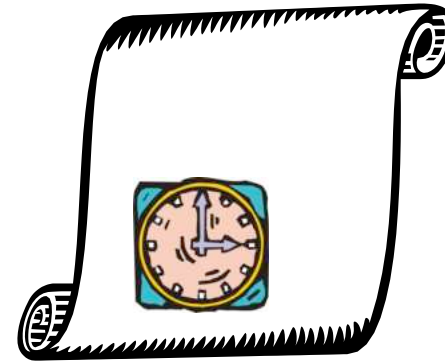
Factory Images "mantaray" for Nexus 10

Version	Download	MD5 Checksum	SHA-1 Checksum
4.2.2 (JDQ39)	Link	b7a1162fb4e617143306ef6c4ca6c040	d79f489e1001d183b31d8a407b47cd5b8e9505cd
4.3 (JWR66Y)	Link	60d0df743b44aee10f125be8e03e84f2	3d8252dd33af47ff7132734b053dfffbc2b58ce4
4.4 (KRT16S)	Link	7b308faf560cedd6970f27fc40828b2e	944139617036fca28848cf0ace67b81f93e08e4e
4.4.2 (KOT49H)	Link	6812260ac97283bd0053e09a05cd5825	174ba74f19a22c0e96467287c34cb63c6e9f751d

Certificazione del Tempo e Funzioni Hash

Il notaio digitale

Quando è stato creato
il documento D ?



Outline

- Concetti Preliminari
- Funzioni Hash in OpenSSL

Funzioni Hash in OpenSSL

- OpenSSL fornisce numerose funzioni hash (o Message Digest)
 - MD4, MD5, SHA1, SHA3, RIPEMD-160, etc.

- Mediante il seguente comando è possibile visualizzare le funzioni hash fornite

```
openssl list --digest-commands
```

- N.B. Alcune delle funzioni fornite hanno problemi di sicurezza

Funzioni Hash in OpenSSL

Il comando dgst

- Il comando `dgst` permette di accedere alle funzioni hash fornite da OpenSSL
 - Opera su dati letti dallo standard input, oppure su uno o più file
- Se alla funzione hash viene passato più di un file, viene calcolato un hash separato per ciascun file
- L'hash calcolato è scritto in formato esadecimale sullo standard output
 - A meno che non sia specificato un file di output

Funzioni Hash in OpenSSL

Opzioni principali del comando dgst

```
openssl dgst args file
```

➤ args

➤ -digest

- Funzione hash da usare per il calcolo del message digest
- digest può essere uno degli algoritmi mostrati dal comando
`openssl list --digest-commands`

➤ -out filename

- File in cui scrivere l'output della funzione. Altrimenti l'output viene scritto sullo standard output

➤ -hmac key

- Crea un hashed MAC usando una determinata chiave

➤ file

- File (uno o più) su cui deve essere applicata la funzione hash

Funzioni Hash in OpenSSL

Opzioni principali del comando dgst

```
openssl dgst args file
```

➤ args

➤ -digest

Per ottenere la lista completa delle opzioni del comando dgst è possibile utilizzare il comando `man dgst`

➤ Funzioni

➤ digests

➤ openssl

➤ -out filename

➤ File in cui scrivere l'output della funzione. Altrimenti l'output viene scritto sullo standard output

➤ -hmac key

➤ Crea un hashed MAC usando una determinata chiave

➤ file

➤ File (uno o più) su cui deve essere applicata la funzione hash

Funzioni Hash in OpenSSL

Esempi Calcolo Hash ed HMAC

- Mediante il seguente comando è possibile calcolare la funzione hash (SHA512 nell'esempio) di un file preso in input (`file.txt`)

```
openssl dgst -sha512 -out DigestOutput.txt file.txt
```

- Mediante il seguente comando è possibile calcolare l'HMAC di un file preso in input (`file.txt`), utilizzando la stringa **P1pp0B4udo** come chiave

```
openssl dgst -sha512 -out HMACOutput.txt -hmac P1pp0B4ud0 file.txt
```



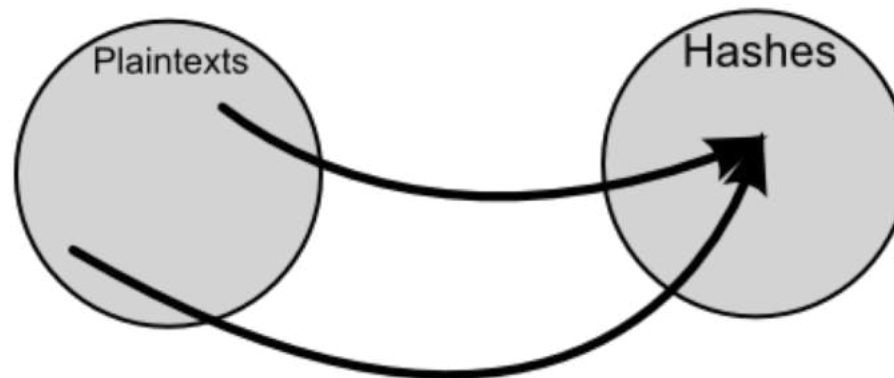
```
HMAC-SHA512(file.txt)=  
7ff9e2a6d40177e962bedd54c35a1b56e7cc557d73167451f59d27ed8ef9c26678bfbf  
c250333af4f9d9a5c78d376e71b04818e94b137ec61a8df6d764267e31
```

Contenuto del file HMACOutput.txt

Funzioni Hash in OpenSSL

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



Funzioni Hash in OpenSSL

Collisioni in MD5

- Prime collisioni annunciate il 17 agosto 2004, da Wang et al., nell'articolo «Collisions for Hash Functions MD4, MD5, HAVAL-128 and RIPEMD»
 - <https://eprint.iacr.org/2004/199>
- Ulteriori collisioni annunciate da Marc Stevens nel 2012, nell'articolo «Single-block collision attack on MD5»
 - <http://eprint.iacr.org/2012/040>

Funzioni Hash in OpenSSL

Collisioni in MD5 - Esempio 1

Dati i seguenti due messaggi in esadecimale

Messaggio 1 (File1.hex)

```
4dc968ff0ee35c209572d4777b721587d36fa7b21bdc56b74a3dc0783e7b9518  
afbfa200a8284bf36e8e4b55b35f427593d849676da0d1555d8360fb5f07fea2
```

Messaggio 2 (File2.hex)

```
4dc968ff0ee35c209572d4777b721587d36fa7b21bdc56b74a3dc0783e7b9518  
afbfa202a8284bf36e8e4b55b35f427593d849676da0d1d55d8360fb5f07fea2
```

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

Funzioni Hash in OpenSSL

Collisioni in MD5 - Esempio 1

- Creiamo due file binari a partire da tali messaggi

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

- Usando `cmp` verifichiamo che `file1` e `file2` siano diversi

- `cmp file1 file2`

- Calcoliamo l'hash MD5 di `file1` e `file2`

- Possiamo osservare che i due file hanno il medesimo hash MD5

```
$ openssl dgst -md5 file1 file2  
MD5(file1)= eddf167f1f2bc9da2ee843952e3fdf45  
MD5(file2)= eddf167f1f2bc9da2ee843952e3fdf45
```

Funzioni Hash in OpenSSL

Collisioni in MD5 - Esempio 2

- È anche possibile trovare collisioni tramite *fastcoll*
 - Strumento che permette di generare coppie di file distinti ma aventi lo stesso hash MD5
- Dipendenze ed installazione dello strumento
 - *Boost (C++ libraries)*
 - `sudo apt-get install libboost-system-dev libboost-program-options-dev libboost-filesystem-dev make make-guile g++`
 - *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`

Funzioni Hash in OpenSSL

Collisioni in MD5 - Esempio 2

```
$ ./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
```

Funzioni Hash in OpenSSL

Collisioni in MD5 - Esempio 2

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

Using output filenames: 'file1'
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

Funzioni Hash in OpenSSL

Collisioni in MD5 - Esempio 2

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

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


Funzioni Hash in OpenSSL

Collisioni in MD5 - Esempio 2

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

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

- Il comando `cmp` non restituisce alcun output quando i file comparati sono identici
 - 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

Funzioni Hash in OpenSSL

Collisioni in MD5 - Esempio 2

- Visualizzando il contenuto dei due file mediante `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
file 2 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: cfcf 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
file 1 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: cfcf f253 9f43 a354 7b44 0e9c 0ce8 e69f  ...S.C.T{D.....
00000070: 5093 9259 46fb a668 bfae 0acf fe7c e774  P..YF..h.....|.t
```

Funzioni Hash in OpenSSL

Collisioni in MD5 - Esempio 2

- La funzione MD5 sui due file restituisce il medesimo risultato
 - Pur essendo tali file diversi tra loro

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

Funzioni Hash in OpenSSL

Collisioni in SHA (SHA-0) - Esempio

Biham, Eli, 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

Messaggio 1 (File1.hex)

Dati i seguenti due messaggi in esadecimale

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	dc9dba a0	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

Funzioni Hash in OpenSSL

Collisioni in SHA (SHA-0) - Esempio

- Creiamo due file binari 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
```

Funzioni Hash in OpenSSL

Collisioni in SHA (SHA-0) - Esempio

- Creiamo due file binari a partire da tali messaggi

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

- Mediante il comando `cmp` verificiamo che i due file siano diversi

- N.B. OpenSSL 1.1.1 non fornisce più la funzione sha (SHA-0)

- Per svolgere questo esempio è necessario utilizzare versioni precedenti di OpenSSL

- Calcoliamo lo SHA di

- Notiamo che i due file hanno lo stesso SHA

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

Funzioni Hash in OpenSSL

Collisioni in SHA-1

- 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
 - Obiettivo: creare due PDF con contenuti visivi distinti, ma con valori hash SHA-1 identici
 - Attacco effettuato realizzando un apposito «header» PDF
 - Noto anche come «prefisso» PDF

Fonti:

<https://security.googleblog.com/2017/02/announcing-first-sha1-collision.html>

<https://marc-stevens.nl/research/papers/SBKAM17-SHAttered.pdf>



Funzioni Hash in OpenSSL

Collisioni in SHA-1

- 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

Fonti:

<https://security.googleblog.com/2017/02/announcing-first-sha1-collision.html>

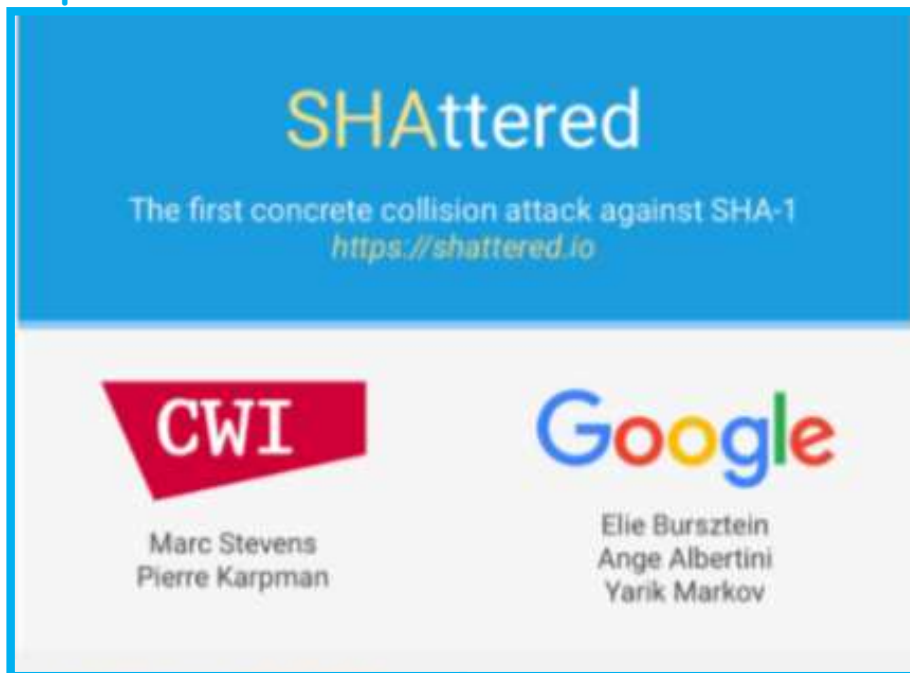
<https://marc-stevens.nl/research/papers/SBKAM17-SHAttered.pdf>



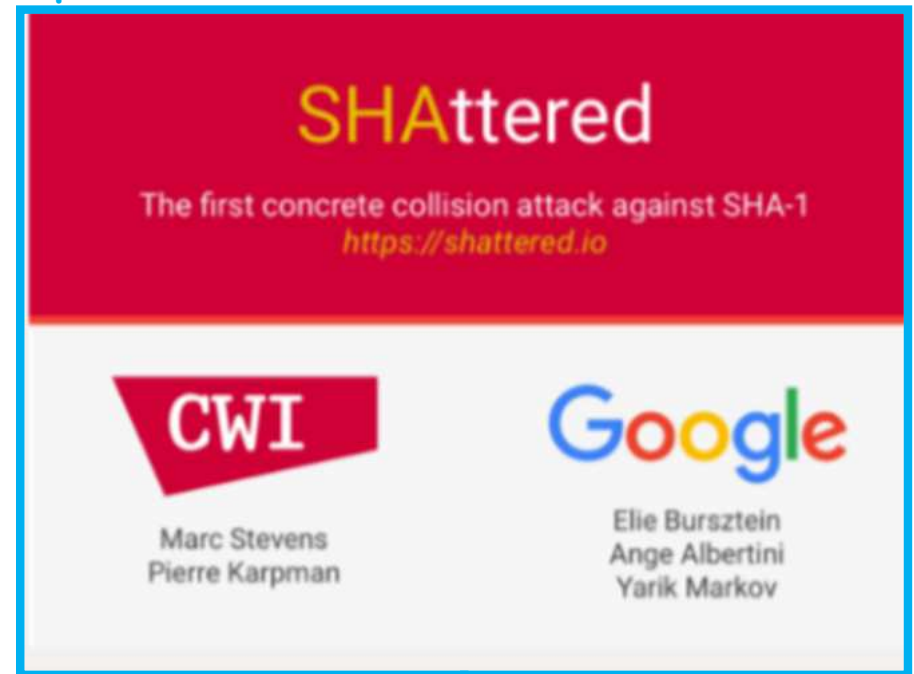
Funzioni Hash in OpenSSL

Collisioni in SHA-1 - Esempio

1.pdf



2.pdf



Stesso SHA-1

38762cf7f55934b34d179ae6a4c80cadccbb7f0a 1.pdf
38762cf7f55934b34d179ae6a4c80cadccbb7f0a 2.pdf

Fonte:

<https://shattered.io/>

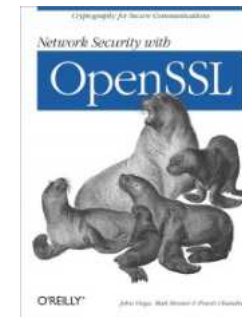


Bibliografia

➤ **Network Security with OpenSSL** Pravir Chandra, Matt Messier and John Viega (2002), O'Reilly

➤ Cap. 2.2

➤ Appendix A. Command-Line Reference



➤ **Documentazione su OpenSSL**

➤ <https://www.openssl.org/docs/>

Bibliografia

- Presentazioni Lezioni Corso di Sicurezza, Prof. De Santis
 - Funzioni Hash
 - MAC