Chapter 3: Hashing

Hashing Types.
Hashing Methods.
Salting.
Collisions.
LM and NTLM Hashes (Windows).
Hash Benchmarks.
Message Authentication Codes (MACs).
OTP/HOTP.

Prof Bill Buchanan OBE
http://asecuritysite.com/crypto03
http://asecuritysite.com/encryption
Chapter 3: Hashing

Hashing Types.
Hashing Methods.
Salting.
Collisions.
LM and NTLM Hashes (Windows).
Hash Benchmarks.
Message Authentication Codes (MACs).
OTP/HOTP.
Secret Shares.

Prof Bill Buchanan OBE
http://asecuritiesite.com/crypto03
http://asecuritiesite.com/encryption
How do we get a fingerprint for data?

With a fingerprint we can hopefully tell if Eve has modified any of the data.

Solved by Prof Ron Rivest with the MD5 hash signature.
Bob

Hashing Algorithm (MD5) - 128 bit signature

- hello → XUFAKrxlKn5cZ22REBFfkg
- Hello → ixqZU8RhepaoJ6v4xHqE1w
- Hello. How are you? → CysDE5j+Z0UbCYztTdsFiw
- Napier → j4NXH5Mkrk4j13N1MFXhtg

Base-64

- hello → 5D41402ABC4B2A76B9719D911017C592
- Hello → 8B1A9953C4611296A827ABF8C47804D7
- Hello. How are you? → CC708153987BF9AD833BEBF90239BF0F
- Napier → 8F83571F9324AE4E23D773753055C7B6

Hex

MD5 hash algorithm
Hello
Hello
Hello. How are you?
Napier

Base-64

AAF4C61DDCC5E8A2DABEDE0F3B482CD9EA9434D
F7FF9E8B7B2E09B70935A5D785E0CC5D9D0ABF0
3EE876026EFA5E18EA13995B4D6B70B2A6DD142F
BF81B135A5687766F4F464764EAC38CA8A4EBABA

SHA-1 hash algorithm
Security and mobility are two of the most important issues on the Internet, as they will allow users to secure their data transmissions, and also break their link with physical connections.

Base-64

F94FBED3DAE05D223E6B963B9076C4EC
+U++09rgXSI+a5Y7kHbE7A==

Hex

8A8BDC3FF80A01917D0432800201CFBF
iovcP/gKAZF9BDKAAGHPvw==
root@kali:~# echo -n "hello" | openssl md5
(stdin)= 5d41402abc4b2a76b9719d911017c592

root@kali:~# echo -n "hello" | md5sum
5d41402abc4b2a76b9719d911017c592  -

root@kali:~# openssl md5 pw
MD5(pw)= 859b6a9be3b45262c4414bd1696ba91b

root@kali:~# md5sum pw
859b6a9be3b45262c4414bd1696ba91b  pw

Hash methods supported:
md2       md4       md5       rmd160     sha
sha1
File/folders

Hashing Algorithm (MD5)
- 128 bit signature

- Hash signatures are used to gain a signature for files, so that they can be checked if they have been changed.

---

<table>
<thead>
<tr>
<th>Path</th>
<th>MD 5 sum</th>
</tr>
</thead>
<tbody>
<tr>
<td>[Path] / filename</td>
<td></td>
</tr>
<tr>
<td>C:\Windows\System32\</td>
<td></td>
</tr>
<tr>
<td>12520437.cpx</td>
<td>0a0feb9eb28bde8cd835716343b03b14</td>
</tr>
<tr>
<td>12520850.cpx</td>
<td>d69ae057cd82d04ee7d311809abefb2a</td>
</tr>
<tr>
<td>8point1.wav</td>
<td>beab165fa58ec5253185f32e124685d5</td>
</tr>
<tr>
<td>aclient.dll</td>
<td>ad45dedf5cf69a28cbeaf6a2ca84b5f1e</td>
</tr>
<tr>
<td>AC3ACM.acm</td>
<td>59fe83d1e4cd0b1ad5ae32ed627ae25f</td>
</tr>
<tr>
<td>Ac3audio.ax</td>
<td>4b87d889edf278e5fa223734a9bbee79a</td>
</tr>
<tr>
<td>ac3filter.cpl</td>
<td>10b27174d46094984e7a05f3c36ac2a</td>
</tr>
<tr>
<td>accessibilitycp1.dll</td>
<td>ac4cecc86eeb8e1cc2e9fe022cff3ac1</td>
</tr>
<tr>
<td>ACCTRES.dll</td>
<td>58f57f2f2133a2a77607c8ccc9a30f73</td>
</tr>
<tr>
<td>acledit.dll</td>
<td>0beee3f36752213d1b09d18e6938398</td>
</tr>
<tr>
<td>ZSHP1020.CHM</td>
<td>c671edf59d9c45a3c5566b5d932cd3d1b09d18e6938398</td>
</tr>
<tr>
<td>ZSHP1020.EXE</td>
<td>96e45a12520437.cpx</td>
</tr>
<tr>
<td>ZSHP1020.HLP</td>
<td>a0769312520850.cpx</td>
</tr>
<tr>
<td>ZSPool.DLL</td>
<td>f ae334f187ef58697c1f12520850.cpx</td>
</tr>
<tr>
<td>Ztag.DLL</td>
<td>7ca8368a4a97ac1f12520850.cpx</td>
</tr>
<tr>
<td>Ztag32.DLL</td>
<td>27b026f8b24f8b24f8b24f8b24f8b24f8b24f8b24f8b24f8b24f8b24f8b24f8b24f8b24f</td>
</tr>
</tbody>
</table>

---

Hash signature

- Hash signatures are used to gain a signature for files, so that they can be checked if they have been changed.

MD5 hash algorithm
Hashing Algorithm (MD5) - 128 bit signature

- Hash signatures are used to identify that a file/certificate has not been changed.

The digital certificate has an SHA-1 hash thumbprint (3f6a...89) which will be checked, and if the thumbprint is different, the certificate will be invalid.
**Windows login/authentication**

- Bob
- `/mypass` → NT hash (MD4) → `fa1bfa4fa13fa12fa10fa1ffa14fa12`

**Cisco password storage (MD5)**

- Bob
- `/mypass` → MD5 encoded password

**One-way hash**
- Hashes are used for digital fingerprints (see the next unit) and for secure password storage.
- Typical methods are NT hash, MD4, MD5, and SHA-1.

**NT-password hash for Windows NT, XP and Vista**

```
# configure
(config)# enable secret test

Current configuration : 542 bytes

| version 12.1
| no service single-slot-reload-enable
| service timestamps debug uptime
| service timestamps log uptime
| no service password-encryption
| hostname Router
| enable secret 5 $1$NwksknsEOYxZVenGjWOGj/TGkO
```
Windows login/authentication

Hashing suffers from dictionary attacks, where the signatures of well-known words are stored in a table, and the intruders does a lookup on this:

- mypast: effahd13fa12fa10fgffafa1ffafa14fa144
- mypass: fa1bfafa14fa13fa12fa10fa1ffafa14fa12
- mypose: ff12189043210954defff0123444512d
- test1: aabbfce023215546dfedddd0101001cd

One-way hash

- Hashing suffers from dictionary attacks, thus it is important that any passwords are not standard words, such as to change password for pA55wOrd.
Risk 4: One Password Fits All

Adobe

150 million accounts compromised

<table>
<thead>
<tr>
<th>#</th>
<th>Count</th>
<th>Cipher text</th>
<th>Plaintext</th>
</tr>
</thead>
<tbody>
<tr>
<td>2.</td>
<td>1981938</td>
<td>EQ7fj2p7fJyQ=</td>
<td>123456789</td>
</tr>
<tr>
<td>2.</td>
<td>446162</td>
<td>j9p+jWtzwT86aMnjgZFLzYg==</td>
<td>password</td>
</tr>
<tr>
<td>3.</td>
<td>435834</td>
<td>L8gb4D5j13j10xG6Cah3Hm==</td>
<td>adobe123</td>
</tr>
<tr>
<td>4.</td>
<td>211659</td>
<td>Bx4eFX+lUxG6CaLH3m==</td>
<td>adobe654</td>
</tr>
<tr>
<td>5.</td>
<td>201189</td>
<td>j9p+jWtzwT86aMnjgZFLzYg==</td>
<td>query</td>
</tr>
<tr>
<td>5.</td>
<td>130632</td>
<td>5dzlV7zCe2yvs=</td>
<td>123456789</td>
</tr>
<tr>
<td>7.</td>
<td>124253</td>
<td>4Q:0zsWPVvQ=</td>
<td>1234567</td>
</tr>
<tr>
<td>8.</td>
<td>113884</td>
<td>LwY2Kwe6I=</td>
<td>111111</td>
</tr>
<tr>
<td>9.</td>
<td>83411</td>
<td>PMDTtP0L2zc035wrFU/Y6A==</td>
<td>photoshop</td>
</tr>
<tr>
<td>10.</td>
<td>82694</td>
<td>e6MP9Q5G6x8=</td>
<td>123123</td>
</tr>
</tbody>
</table>

LinkedIn

47 million accounts

6.5 million accounts (June 2013)

One account hack ... leads to others

TJ-MEMK

Marshall's

677 million accounts - in plain text.

Dropbox

Compromised 2013

CitiGroup

200,000 client accounts
Brute Force - How many hash codes?

• 7 digit password with [a-z] ... how many?
  • Ans:
    • Time to crack - 100 billion per second:

• 7 digit with [a-zA-Z] ... how many?
  • Ans:
    • Time to crack – 100 billion per second:

• 7 digit with [a-zA-Z!@#$%^&*()] ... how many?
  • Ans:
    • Time to crack – 100 billion per second:
Chapter 3: Hashing

Hashing Types.

**Hashing Methods.**

- Salting.
- Collisions.
- LM and NTLM Hashes (Windows).
- Hash Benchmarks.
- Message Authentication Codes (MACs).
- OTP/HOTP.
- Secret Shares.

Prof Bill Buchanan OBE

http://asecuritysite.com/crypto03
http://asecuritysite.com/encryption
**LM Hash**

LM Hash. LM Hash is used in many versions of Windows to store user passwords that are fewer than 15 characters long.

**SHA-3**

SHA-3. SHA-3 was known as Keccak and is a hash function designed by Guido Bertoni, Joan Daemen, Michaël Peeters, and Gilles Van Assche. MD5 and SHA-0 have been shown to be susceptible to attacks, along with theoretical attacks on SHA-1. NIST thus defined there was a need for a new hashing method which did not use the existing methods for hashing, and setup a competition for competing algorithms. In October 2012, Keccak won the NIST hash function competition, and is proposed as the SHA-3 standard.

**Bcrypt**

Bcrypt. This creates a hash value which has salt.

**RIPEMD**

RIPEMD (RACE Integrity Primitives Evaluation Message Digest) and GOST. RIPEMD160. RIPEMD is a 128-bit, 160-bit, 256-bit or 320-bit cryptographic hash function, and was created by Hans Dobbertin, Antoon Bosselaers and Bert Preneel. It is used on TrueCrypt, and is open source. The 160-bit version is seen as an alternative to SHA-1, and is part of ISO/IEC 10118.

Tiger. Tiger is a 192-bit hash function, and was designed by Ross Anderson and Eli Biham in 1995. It is often used by clients within Gnutella file sharing networks, and does not suffer from known attacks on MD5 and SHA-0/SHA-1. Tiger2 is an addition, in which the message is padded with a byte of 0x80 (in a similar way to MD4, MD5 and SHA), whereas in Tiger it is 0x01. Otherwise the two methods are the same in their operation.

Author: Prof Bill Buchanan
Murmur

While hashing methods such as MD5 and SHA-1 use crypto methods, the Murmur and FNV hashes uses a non-cryptographic hash function. The Murmur hash, designed by Austin Appleby, uses a non-cryptographic hash function. This can be used for general hash-based lookups. It has a good performance compared with other hashing methods, and generally provide a good balance between performance and CPU utilization. Also it performs well in terms of hash collisions.

FNV

FNV (Fowler–Noll–Vo) is a 64-bit non-cryptographic hash function developed by Glenn Fowler, Landon Curt Noll, and Phong Vo. There are two main versions, of which 1a is the most up-to-date version.
Chapter 3: Hashing

Hashing Types.
Hashing Methods.
Salting.
Collisions.
LM and NTLM Hashes (Windows).
Hash Benchmarks.
Message Authentication Codes (MACs).
OTP/HOTP.
Secret Shares.

Prof Bill Buchanan OBE
http://asecuritysite.com/crypto03
http://asecuritysite.com/encryption
Adding salt

- Salt increases the range of the possible signatures

```
mypass
```

```
NT hash (MD4)
```

```
falbfa14fa13fa12fa10fa1fafa14fa12
```

NT-password hash for Windows NT, XP and Vista

Salt increase the range of the signatures

```
mypast
```

```
effahd13fa12fa10fgffafa1fafa14fa144
caaahdd3fa12ccfae34234550011aff
Dddee432969450310403010d0ae000100
```
Password: $1$fred$bATAk8UUH/IDAp9sd61Uv/

Python Code:
```python
C:\openssl>openssl passwd -1 -sal: fred password
$1$fred$bATAk8UUH/IDAp9sd61Uv/
```
OpenSSL

Cryptography and SSL/TLS Toolkit

```
$ openssl version
OpenSSL 1.0.1f 6 Jan 2014

$ openssl dgst -md5 file
MD5(file) = b1946ac92492d2347c6235b4d861184

$ openssl genrsa -out mykey.pem 1024
Generating RSA private key, 1024 bit long modulus
.......................................................
...++++++
e is 65537 (0x10001)

$ openssl rsa -in mykey.pem -pubout > mykey.pub
writing RSA key

$ cat mykey.pub
-----BEGIN PUBLIC KEY-----
MIIFGzIBADANBgkqhkiG9w0BAQEFAASCAIAADcQGfG9zGSIb3DQEBAQUAA4GNADCBiMQ
kJBgQoDxv9H5Fkpm+ZuoQcpdH8ziUwx3EZIKM0nsjgjc5ZTYVaf9CMLtmKtZe5p7aqX8o9
nePfTkQ73Ta9VOD6cX51/cgYXy2ts3hW0imrW1FVDFyJx7klmc0uwbFFC9ZmtJxIaxaa9SV2k
ARxOCTJ2uOjRTCCeXU09IyJHnThSNJeiTjQD4QAB

---END PUBLIC KEY-----
```

```
$ cat /etc/shadow
root:$1$Etg2ExUZ$F9NTP7omafhKl7qaBMqng1:15651:0:99999:7::

$ openssl passwd -1 -salt Etg2ExUZ redhat
$1$Etg2ExUZ$F9NTP7omafhKl7qaBMqng1
```
Chapter 3: Hashing

Hashing Types.
Hashing Methods.
Salting.

**Collisions.**
LM and NTLM Hashes (Windows).
Hash Benchmarks.
Message Authentication Codes (MACs).
OTP/HOTP.
Secret Shares.

Prof Bill Buchanan OBE
http://asecuritysite.com/crypto03
http://asecuritysite.com/encryption
A major factor with hash signatures is:

- **Collision.** This is where another match is found, no matter the similarity of the original message. This can be defined as a **Collision attack**.
- **Similar context.** This is where part of the message has some significance to the original, and generates the same hash signature. The can be defined as a Pre-image attack.
- **Full context.** This is where an alternative message is created with the same hash signature, and has a direct relation to the original message. This is an extension to a Pre-image attack.

In 2006 it was shown that MD5 can produce collision within less than a minute.

A 50% probability of a collision is:

$$\sqrt{N(\text{signatures})} = \sqrt{2^n} = 2^{n/2}$$

where $n$ is the number of bits in the signature. For example, for MD5 (128-bit) the number of operations that would be required for a better-than-50% chance of a collision is:

$$2^{64}$$

Note, in 2006, for SHA-1 the best time has been 18 hours.
The MD5 signature gives the same result

79054025255FB1A26E4BC422AEF54EB4
C:\openssl>openssl md5 hash01.jpg
MD5(hash01.jpg) = e06723d4961a0a3f950e7786f3766338

C:\openssl>openssl md5 hash02.jpg
MD5(hash02.jpg) = e06723d4961a0a3f950e7786f3766338

Nat McHugh

- 10 hours of computing on the Amazon GPU Cloud.
- Cost: 60 cents
- Used: Hashcat (on CUDA)
- Birthday attack: A group size of only 70 people results in a 99.9% chance of two people sharing the same birthday.
- M-bit output there are $2^m$ messages, and the same hash value would only require $2^{(m/2)}$ random messages.

18,446,744,073,709,551,616.
Chapter 3: Hashing

Hashing Types.
Hashing Methods.
Salting.
Collisions.
**LM and NTLM Hashes (Windows).**
Hash Benchmarks.
Message Authentication Codes (MACs).
OTP/HOTP.
Secret Shares.

**Prof Bill Buchanan OBE**
http://asecuritysite.com/crypto03
http://asecuritysite.com/encryption
C:> user2sid \pluto guest
S-1-5-21-7623811015-3361044348-030300820-501
C:> sid2user 5 21 7623811015 3361044348 030300820 500
Name is Fred
Domain is PLUTO

SID and RID

S - Security ID (SID)

Identifier Authority (48 bits)
5 = Login ID

Revision level

Sub-authority (21 – NT Non unique)

SA Domain ID

SA

Relative User ID (RID):
500 = Admin
501 = Guest
502 = Kerberos
1000 = First user
1001 = Second user

SIDs are used to uniquely identify users and their group memberships – identifies User, Group and Computer accounts.
HKLM\SAM

SAM Database (stores usernames and passwords)

Domain Controller - runs Active Directory which contains information of domain objects

Local Authority Subsystem (Lsass) - Windows Security mechanism - Attached by Sasser Worm which exploited a buffer overflow

Responsible for local security policy
- Controls access.
- Managing password policies.
- User authentication.
- Audit messages.

Windows domain
- LM Hash (Windows XP, 2003)
- NTLMv2 (Windows 7, 8, etc) – connect to Active Directory
- NTLM (Windows 7, 8, etc) – No salt

hashme gives:
FA-91-C4-FD-28-A2-D2-57-AA-D3-B4-35-B5-14-04-EE
FF2A43841C84518A18795AB6E3C8A62E (NTLM)

napier gives:
12-B9-C5-4F-6F-EC-EC-80-AA-D3-B4-35-B5-14-04-EE
307E40814E7D4E103F6A69B04EA78F3D (NTLM)

Root@kali:~# cat pw
myuser:500:12b9c54=6f0ec80aad3b435b51404ee:307e40814e7d4e103f6a69b04ea78f3d::
Root@kali:~# john pw
Loaded 1 password hash (LM DES [128/128 BS SSE2])
NAPTFR
(napier)
guesses: 1 time: 0:00:00:00 100% (1) c/s: 4850 trying: NAPIER - N4PI3R
Use the "--show" option to display all of the cracked passwords reliably.

Author: Prof Bill Buchanan
Registry: HKEY_Local_Machine\SAM

Root@kali:~# cat pw
myuser:500:12b9c54f6fe0ec80aad3b435b51404ee:307e10814e7d4e103f6a69b04ea78f3d:
Root@kali:~# john pw
Loaded 1 password hash (LM DES [128/128 BS SSE2])
NAPIER
(napier)
guesses: 1 time: 0:00:00:00 100% (1) c/s: 4850 trying: NAPIER - N4PI3R
Use the "--show" option to display all of the cracked passwords reliably

<user>:<id>:<LM hash>:<NTLM hash>:<comment>:<home_dir>:
password: 500:E52C4G7419A9A224A3B10B3F3FGC8G6D:0045F79E0E8
FB117AD668D3087586C5:
myuser:500:12b9c54f6fe0ec80aad3b435b51404ee:307e10814e7d4e103f6a69b04ea78f3d:

ophcrack
Hash Crackers/Bit Coin Miners

25 GPU Hash Cracker
- An eight character NTLM password cracked in 5.5 hours. 14 character LM hash cracked in six minutes. 350 billion hashes per second.

Fast Hash One
- 1.536TH/s – Cost 3-5,000 dollars.
Chapter 3: Hashing

Hashing Types.
Hashing Methods.
Salting.
Collisions.
LM and NTLM Hashes (Windows).
Hash Benchmarks.
Message Authentication Codes (MACs).
OTP/HOTP.

Prof Bill Buchanan OBE
http://asecuritysite.com/crypto03
http://asecuritysite.com/encryption
Hashes “The quick brown fox jumps over the lazy dog:

SHA-1: 2fd4e1c67a2d28fced849ee1bb76e7391b93eb12
SHA-256: d7a8fb307d7809469ca9abc0082e4f8d5651e46d3cdb762d02df37c9e592
SHA-512: 07e547d95866fa73f73fbac0435ed67951218fb7d0c8d788a309d7

85436bb642e93a252af954f3912547d1e8a3b5ed6e1b7f7097821233fa0538f3db854fee6

MD-5: 9e107d9d372bb6826b82a56c9fa95f3b
DES: ZDeS94Lcq/6zg
Bcrypt: $2a$05$2czCv5GYgkx3aobmEyewB.ejV2hePMdvTdCyNaS2WtlGPPjB2xx6
APR1: $apr1$ZDzPE45CS3PvRanPycmNc6c2Gw9T9b/
PBKDF2 (SHA1): $pbkdf2$5$Wkr6UE0NUM$0RB2bimWrMY.EPY1bpBT2q3HFg
PBKDF2 (SHA-256): $pbkdf2-sha256$5$Wkr6UE0NUM$yrJz2oJix7uBJWZ/50vWUgdE

Benchmark

Ultra fast:
Murmur: 545,716 hashes per second

Fast:
SHA-1: 134,412
SHA-256: 126,323
MD5: 125,741
SHA-512: 76,005
SHA-3 (224-bit): 72,089

Medium speed:
LDAP (SHA1): 13,718
MS DCC: 9,582
NT Hash: 7,784
MySQL: 7,724
Postgres (MD5): 7,284

Slow:
PBKDF2 (SHA-256): 5,026
Cisco PIX: 4,402
MS SQL 2000: 4,225
LDAP (MD5): 4,180
Cisco Type 7: 3,775
PBKDF2 (SHA-256): 2,348

Ultra-slow:
LM Hash: 733
APR1: 234
Bcrypt: 193
DES: 88
Oracle 10: 48

Postgres (MD5): md5d44c15daa11770f25c5350f7e5408dd1
Cisco PIX: kGyKNSqdfQ1qJUs
Cisco Type 7: 15260309443B3E2DB3875200108010D41505640135E1BE0E80
51957156401540035E460B594D1D53020B5C
Chapter 3: Hashing

Hashing Types.
Hashing Methods.
Salting.
Collisions.
LM and NTLM Hashes (Windows).
Hash Benchmarks.
Message Authentication Codes (MACs).
OTP/HOTP.

Prof Bill Buchanan OBE
http://asecuritysite.com/crypto02
http://asecuritysite.com/encryption
testing123' and a key of "hello", and you should get: AC2C2E614882E7158F69B7E3B12114465945D01

HMAC-MD5
HMAC-SHA-1
HMAC-SHA-256
HMAC-SHA-384
HMAC-SHA-512

MAC 3DES

Message

Receiver

HMAC

Message

Sender

HMAC

Secret key

Secret key

Receiver checks the HMAC code against received one – if they match the sender is validated, and the message is also confirmed

Author: Prof. Bill Buchanan
Chapter 3: Hashing

Hashing Types.
Hashing Methods.
Salting.
Collisions.
LM and NTLM Hashes (Windows).
Hash Benchmarks.
Message Authentication Codes (MACs).
OTP/HOTP.

Prof Bill Buchanan OBE
http://asecuritysite.com/crypto03
http://asecuritysite.com/encryption
Chapter 3: Hashing

Hashing Types.
Hashing Methods.
Salting.
Collisions.
LM and NTLM Hashes (Windows).
Hash Benchmarks.
Message Authentication Codes (MACs).
OTP/HOTP.

Prof Bill Buchanan OBE
http://asecuritysite.com/crypto03
http://asecuritysite.com/encryption