Managing Enterprise Cybersecurity MIS 4596

Unit#8

Agenda

- Some useful Linux commands
- Symmetric cryptography
- Block versus Stream ciphers
- Block ciphers
- Block ciphers mode of operations
- Hashes

Some useful Linux commands

File Commands

ls – directory listing
ls -al – formatted listing with hidden files
cd *dir* - change directory to *dir*
cd – change to home
pwd – show current directory
mkdir *dir* – create a directory *dir*
rm *file* – delete *file*
rm -r *dir* – delete directory *dir*
rm -f *file* – force remove *file*
rm -rf *dir* – force remove directory *dir* *
cp *file1 file2* – copy *file1* to *file2*
cp -r *dir1 dir2* – copy *dir1* to *dir2*; create *dir2* if it doesn't exist
mv *file1 file2* – rename or move *file1* to *file2*
if *file2* is an existing directory, moves *file1* into directory *file2*
ln -s *file link* – create symbolic link *link* to *file*
touch *file* – create or update *file*
cat > *file* – places standard input into *file*
more *file* – output the contents of *file*
head *file* – output the first 10 lines of *file*
tail *file* – output the last 10 lines of *file*
tail -f *file* – output the contents of *file* as it grows, starting with the last 10 lines

System Info

date – show the current date and time
cal – show this month's calendar
uptime – show current uptime
w – display who is online
whoami – who you are logged in as
finger *user* – display information about *user*
uname -a – show kernel information
cat /proc/cpuinfo – cpu information
cat /proc/meminfo – memory information
man *command* – show the manual for *command*
df – show disk usage
du – show directory space usage
free – show memory and swap usage
whereis *app* – show possible locations of *app*
which *app* – show which *app* will be run by default

Shortcuts

Ctrl+C – halts the current command
Ctrl+Z – stops the current command, resume with **fg** in the foreground or **bg** in the background
Ctrl+D – log out of current session, similar to **exit**
Ctrl+W – erases one word in the current line
Ctrl+U – erases the whole line
Ctrl+R – type to bring up a recent command
!! - repeats the last command
exit – log out of current session

Agenda

- ✓Some useful Linux commands
- Symmetric cryptography
- Block versus Stream ciphers
- Block ciphers
- Block ciphers mode of operations
- Hashes

Symmetric and asymmetric algorithms

…both are 2-way functions that support encryption & decryption

- Symmetric cryptography
  - Use a copied pair of symmetric (identical) secret keys
  - The sender and the receive use the same key for encryption and decryption functions
- Asymmetric cryptography
  - Also know as “public key cryptography”
  - Use different (“asymmetric”) keys for encryption and decryption
  - One is called the “private key” and the other is the “public key”

A strong cipher contains

2 main attributes

1. **Confusion:** usually carried out through substitution
2. **Diffusion:** Usually carried out through transposition

Harris, S. and Maymi, F. (2016) All-In-One CISSP Exam Guide, McGraw Hill Education

Symmetric cryptography

Strengths:

- Much faster (less computationally intensive) than asymmetric systems.
- Hard to break if using a large key size.

Weaknesses:

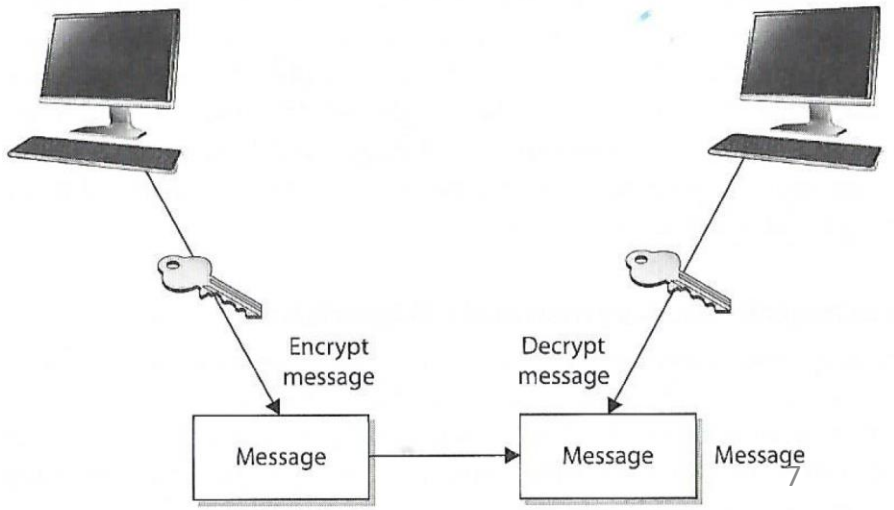
- Requires a secure mechanism to deliver keys properly.
- Each pair of users needs a unique key, so as the number of individuals increases, so does the number of keys, possibly making key management overwhelming.
- Provides confidentiality but not authenticity or nonrepudiation.

Two types: Stream and Block Ciphers

- **Stream Ciphers** treat the message a stream of bits and performs mathematical functions on each bit individually

- **Block Ciphers** divide a message into blocks of bits and transforms the blocks one at a time

Symmetric encryption uses the same keys.



Block Ciphers versus Stream Ciphers

Stream ciphers work on a single bit at a time:

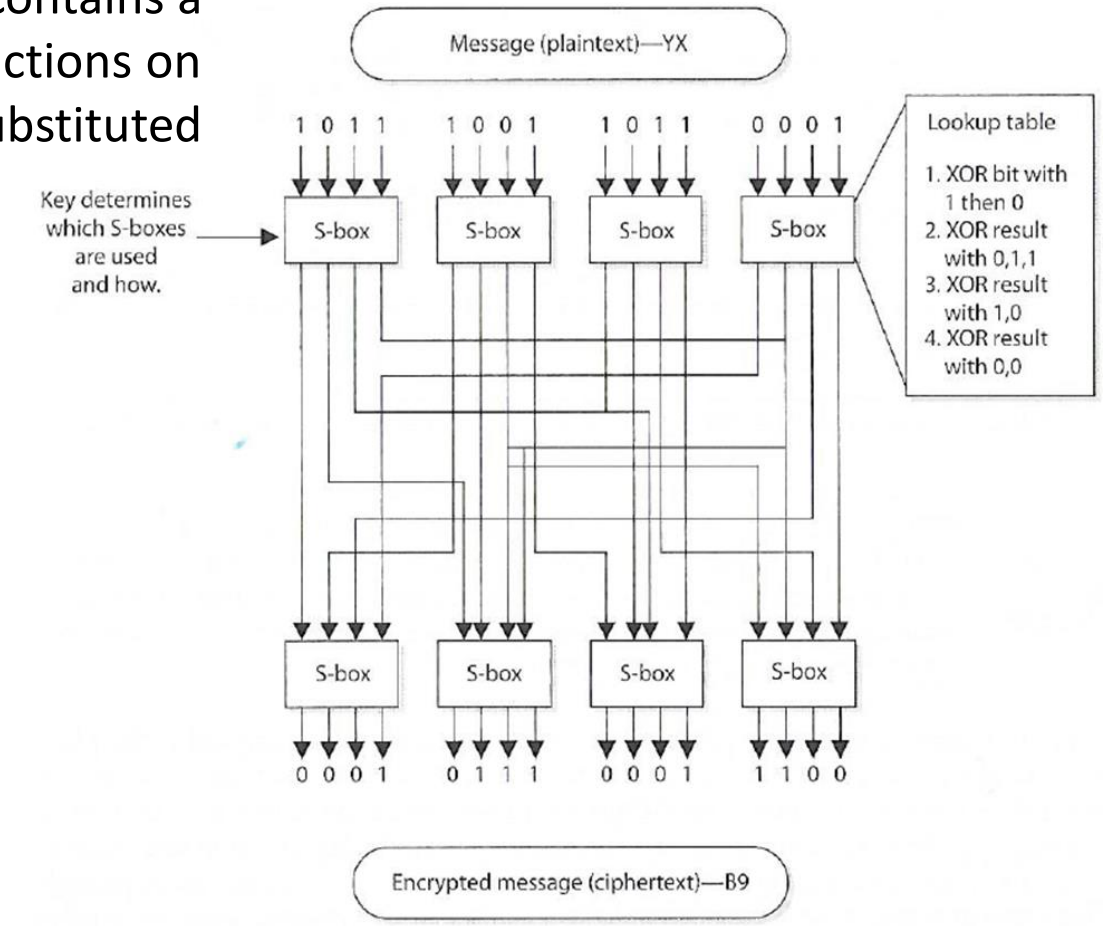
0	1	1	1	0	0	1	1	Plaintext
XOR								
1	0	0	1	0	1	0	1	Key
1	1	1	0	0	1	1	0	Ciphertext

PRNG = Pseudo Random Number Generator

Block Ciphers versus Stream Ciphers

In contrast, block ciphers encrypt a block of bits at a time

In this example, each Substitution Box (S-box) contains a lookup table used by the algorithm as instructions on how the bits are substituted



Plaintext	Ciphertext
0000	1110
0001	0100
0010	1101
0011	0001
0100	0010
0101	1111
0110	1011
0111	1000
1000	0011
1001	1010
1010	0110
1011	1100
1100	0101
1101	1001
1110	0000
1111	0111

Encryption table

Ciphertext	Plaintext
0000	1110
0001	0011
0010	0100
0011	1000
0100	0001
0101	1100
0110	1010
0111	1111
1000	0111
1001	1101
1010	1001
1011	0110
1100	1011
1101	0010
1110	0000
1111	0101

Decryption table

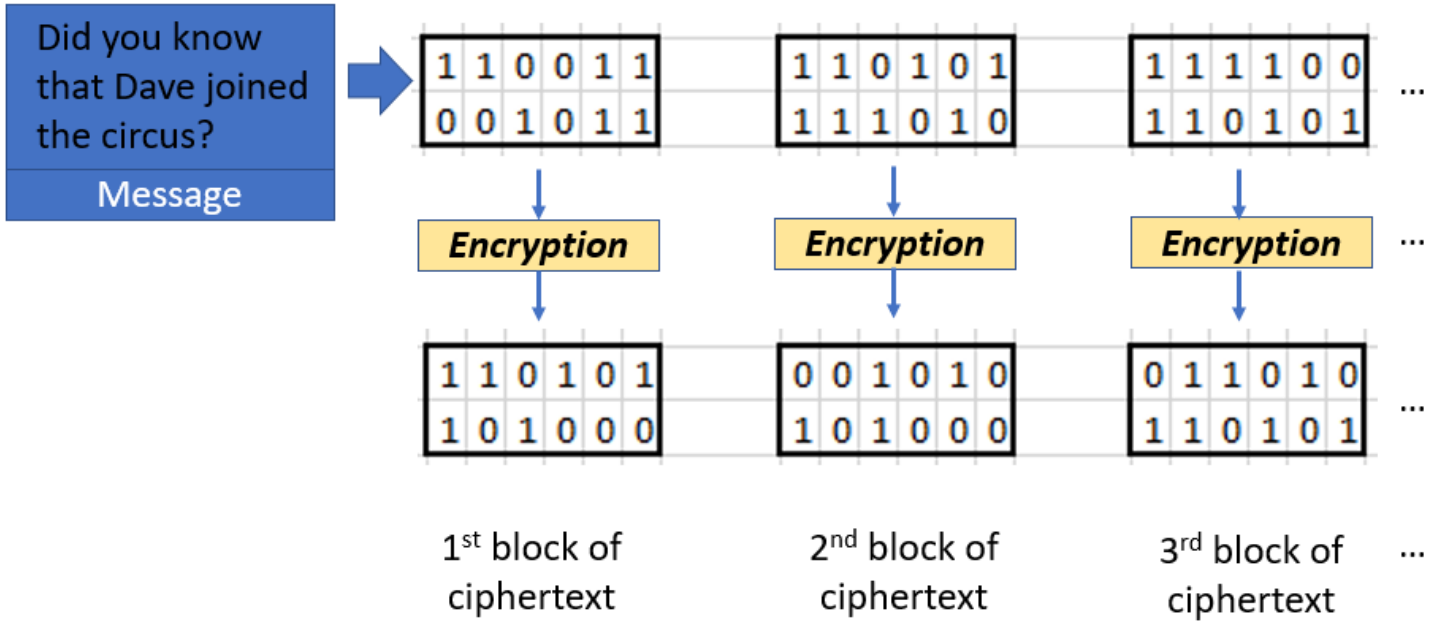
…followed by transposition…

Block Cyphers ("Cipher")

- Message is divided into blocks of bits
- Blocks are put through encryption functions 1 block at a time

Suppose you are encrypting a 648-bit long message to send to your mother using a block cypher that uses 12 bits

- Your message would be chopped up into 54 blocks each 12 bits long
- Each block, in turn, would be run through a series of encryption functions (substitution and transposition)
- Ending up with 54 blocks of ciphertext



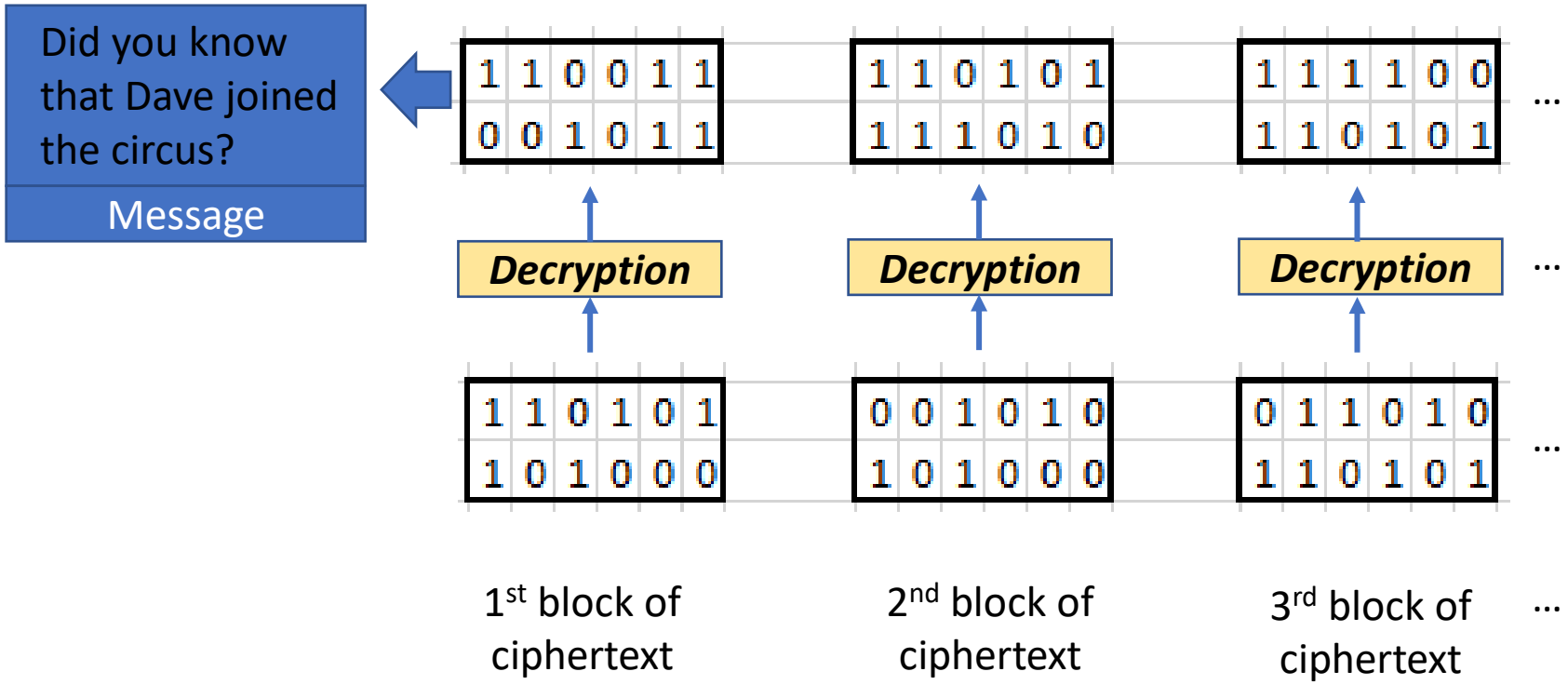
Note: Block ciphers typically use 64, 128, 512 bits at a time

Block Ciphers

- Message is divided into blocks of bits
- Blocks are put through mathematical functions 1 block at a time

You send the message to your mother. She uses the same block cipher and key (symmetric) to decipher the message

- The 54 ciphertext blocks go back through the algorithm in the reverse sequence
- Resulting in your original plaintext message your message



Modern Block Ciphers

- Use block sizes of 128-bits or greater
  - Examples of Block Ciphers that can be used are:
    - AES (NIST's 2001 Advanced Encryption Standard – originally known as Rijndael)
      - 128 bit block size, but 3 different key lengths: 128, 192, and 256 bits
    - Blowfish
    - Twofish
    - Serpent
- Do not use these examples of block ciphers which have a 56 bit key length, which is too small to provide secure encryption:
  - DES (Data Encryption Standard)
  - 3DES

Practical Cryptanalysis

DES Cracker:

- A DES key search machine
- Contains 1,536 chips
- Cost: $250,000
- Searches 88 billion keys per second
- Won RSA Laboratory's "**DES Challenge II-2**" by successfully finding a DES key in 56 hours

Agenda

- ✓Kali Linux workaround – Alternative way to bring up a terminal window
- ✓Some useful Linux commands
- ✓Symmetric cryptography
- ✓Block versus Stream ciphers
- ✓Block ciphers
- Block ciphers mode of operations
- Hashes

Block cipher's "mode of operation"

5 modes of operation are used to tailor them for use in different applications:

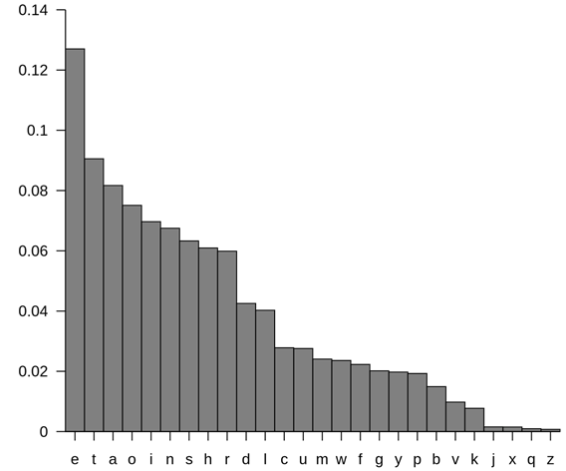
1. ECB – Electronic Code Book mode
2. CBC – Cipher Block Chaining mode
3. CFB – Cipher FeedBack mode
4. OFB – Output FeedBack mode
5. CTR – CounTeR mode

ECB – Electronic Code Book mode

- A data block of a certain size (e.g. 64 bits or 128 bits or…) is entered into the algorithm with the key, and a block of cipher text is produced

$C_i = Encrypt(Key, P_i)$
for $i = 1, ..., k$

Where:
- *Ci is block i of ciphertext*
- *P, is a block of plaintext*

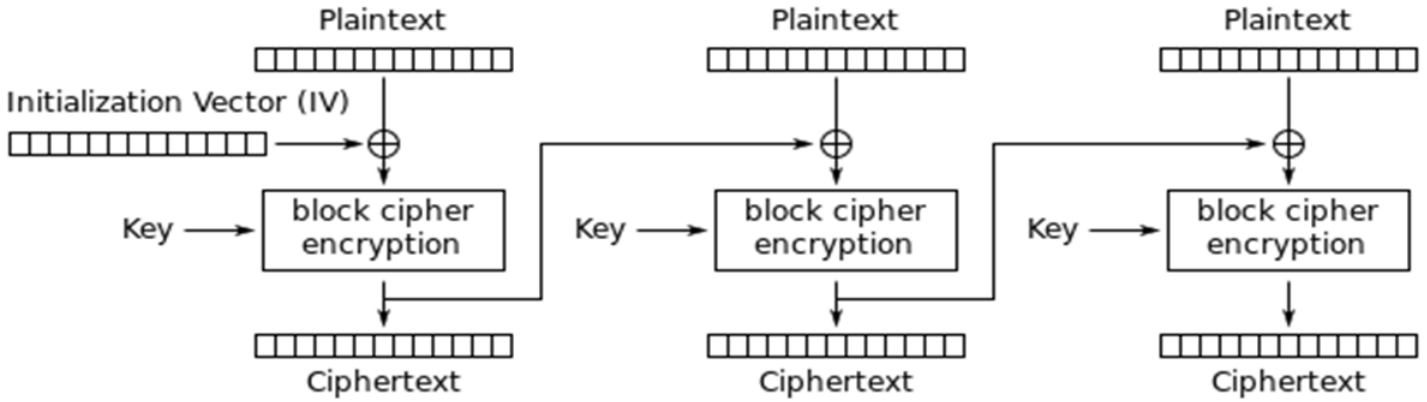


- Encrypts every block <u>the same way every time for a given key</u>

- Why is this a problem?
  - ➢ This is a problem because **frequency analysis** of the encrypted text can reveal a lot of information
  - ➢ Not enough randomness

Solution: CBC – Cipher Block Chaining mode

- Is much more secure
- Does not reveal a pattern of encryption for frequency analysis
- Each block of text, the key, and the value based on the previous block are processed in the algorithm and applied to the next block of text



- XORs a plaintext with the **last** encrypted block before encrypting it. This ensures that the same plaintext is encrypted differently every time.
- Requires an initialization vector (or IV) to get started, since the first block doesn't have a previous encrypted block to XOR against.

A similar concept to ***diffusion*** is known as the

The Avalanche Effect

A changed to a single plaintext bit should have an influence over several of the resulting ciphertext bits

In a strong block cipher, if 1 plaintext bit is changed, it will change every ciphertext bit with the probability of 50%

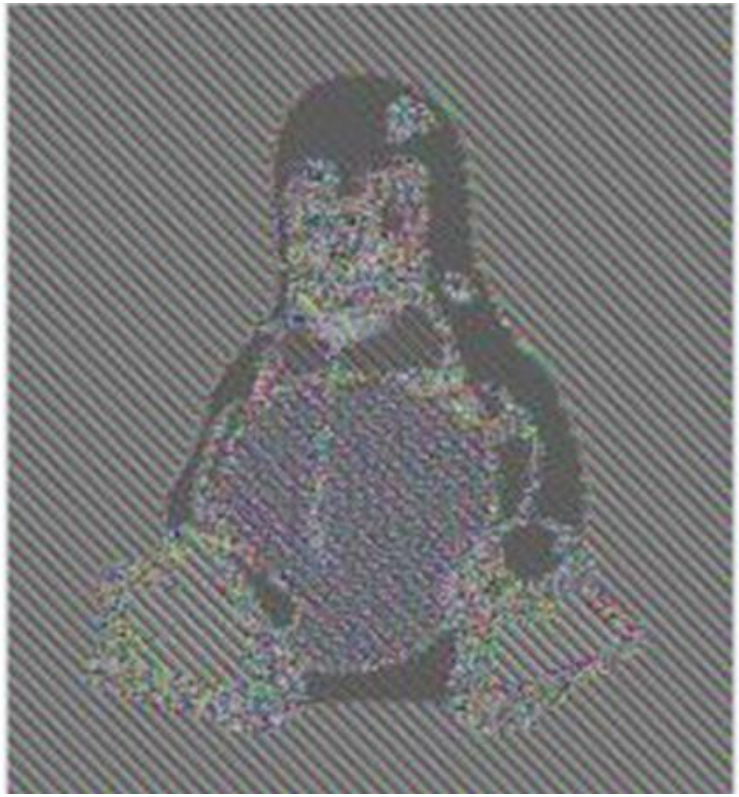
That is, if 1 plaintext bit changes, then about ½ of the ciphertext bits will change

Avalanche Effect: A small change to the key or to the plaintext should cause drastic changes to the resulting ciphertext

Harris, S. and Maymi, F. (2019) All-In-One CISSP Exam Guide, McGraw Hill Education

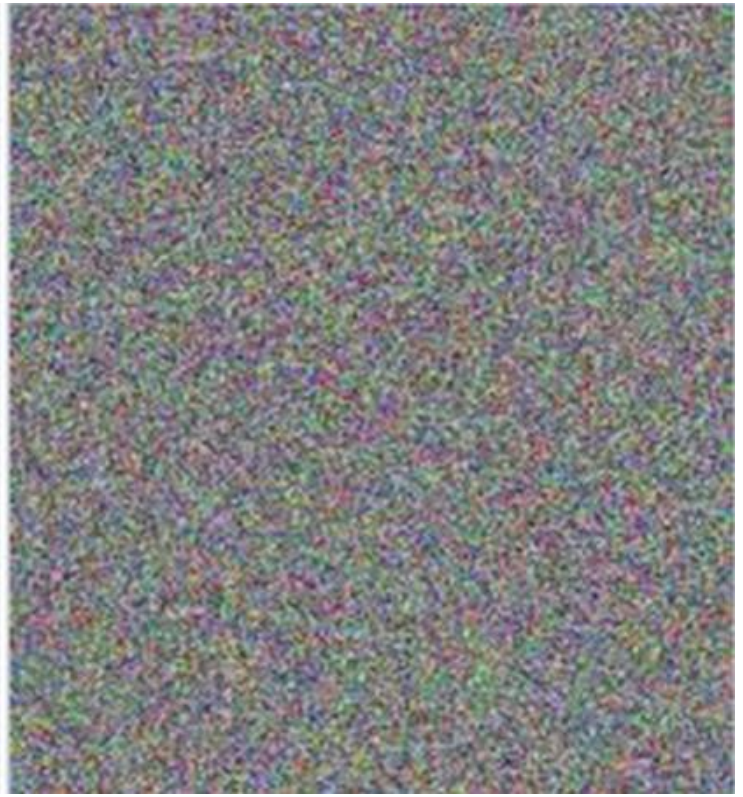
Visa

Original Image



Block cipher with ECB (Electronic Code Book) encryption

Not good!



Block cipher with CBC (Cipher Block Chaining) or any of the other modes of encryption

These are good!

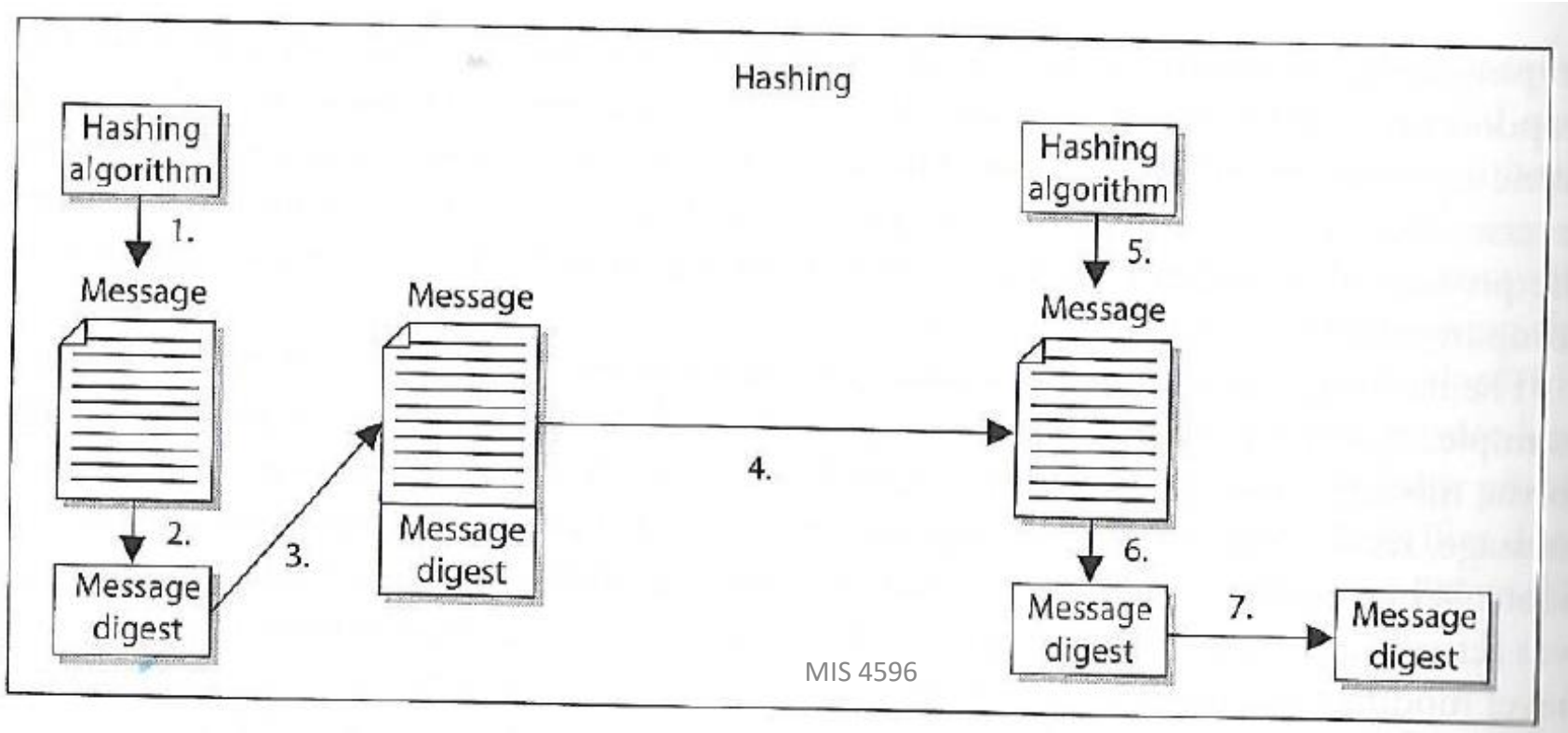
Agenda

- ✓Kali Linux workaround – Alternative way to bring up a terminal window
- ✓Some useful Linux commands
- ✓Symmetric cryptography
- ✓Block versus Stream ciphers
- ✓Block ciphers
- ✓Block ciphers mode of operations
- Hashes

One-way Hash function

- Assures message **integrity**
- A function that takes a variable-length string (i.e. message) and produces a fixed-length value called a hash value
- Does not use keys

1. Sender puts message through hashing function
2. Message digest generated
3. Message digest appended to the message
4. Sender sends message to receiver
5. Receiver puts message through hashing function
6. Receiver generates message digest value
7. Receiver compares the two message digests values. If they are the same, the message has not been altered



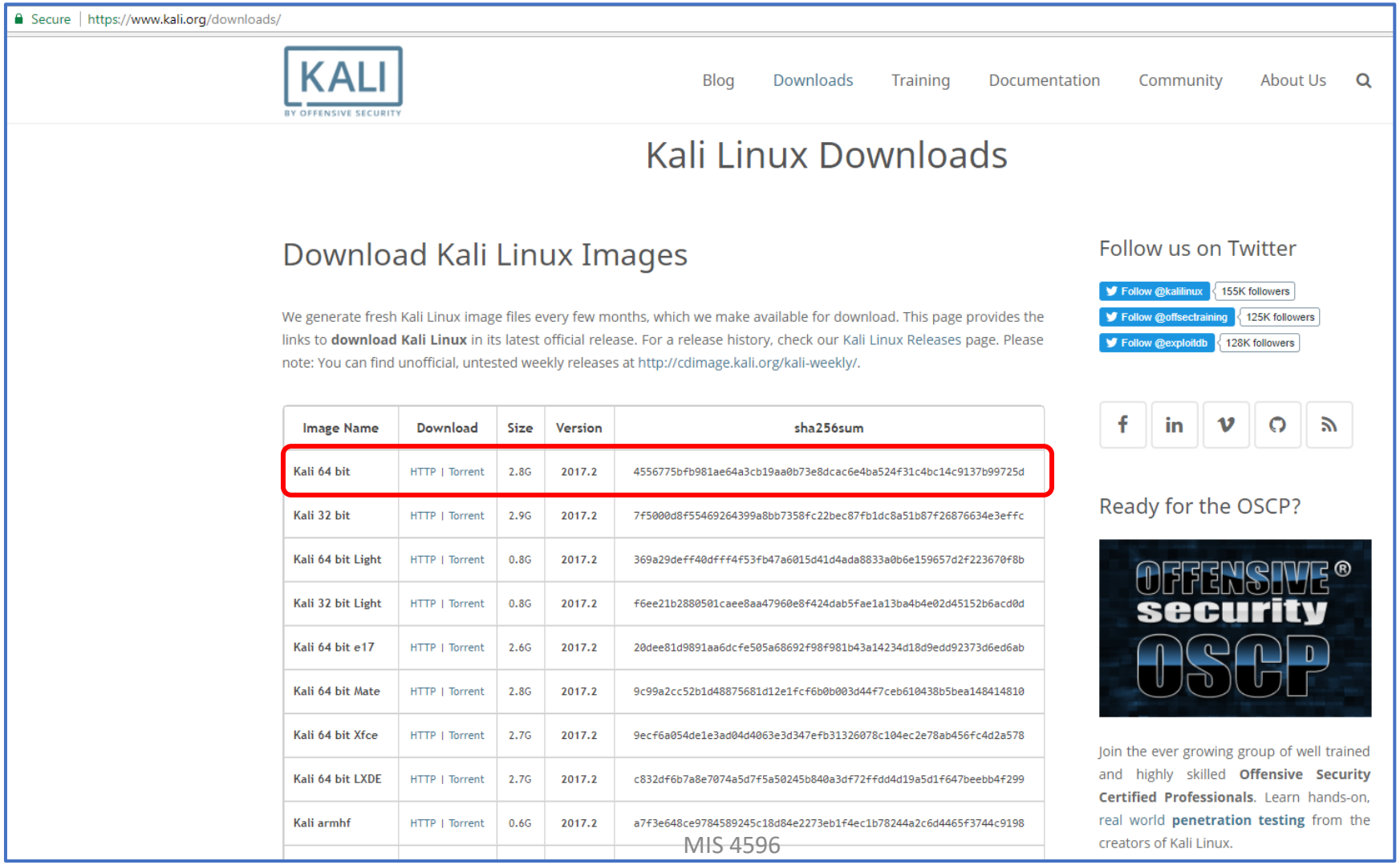
Hashing results in **fixed-sized output**

Names for the output of a hashing functions include “hash” and a *message digest* **(md)**, because a hash “digests” an input of any size down to a **fixed-sized output**

- No matter the size of the input, the output is the same, for example the md5 hash function’s output:
  - Letter ‘a’ in binary: 01000001 => md5 hash => 32-character string
  - Blu-ray disk digest => md5 hash => 32-character string
  - 6 TB hard drive digest => md5 hash => 32-character string

One-way hash example…

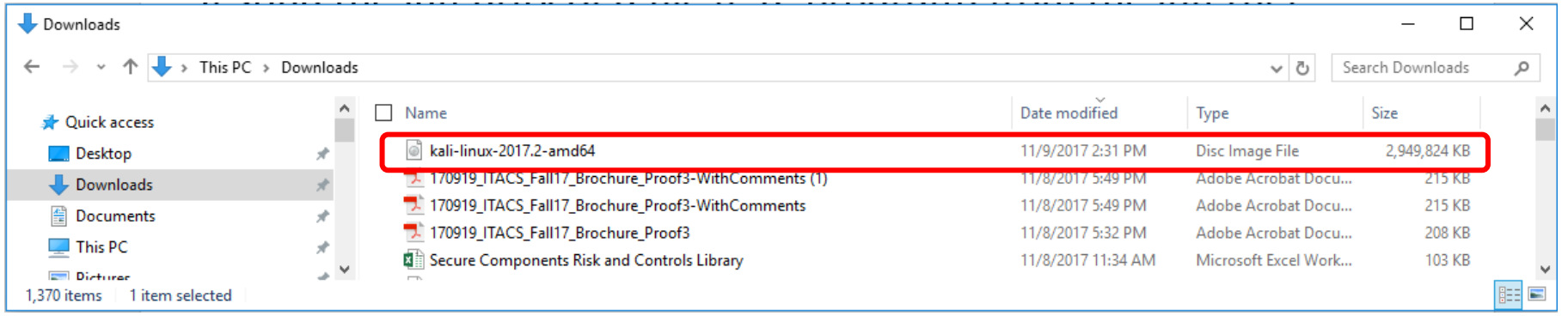
Testing the integrity of a file (e.g. program) downloaded from the internet…



One-way hash example…

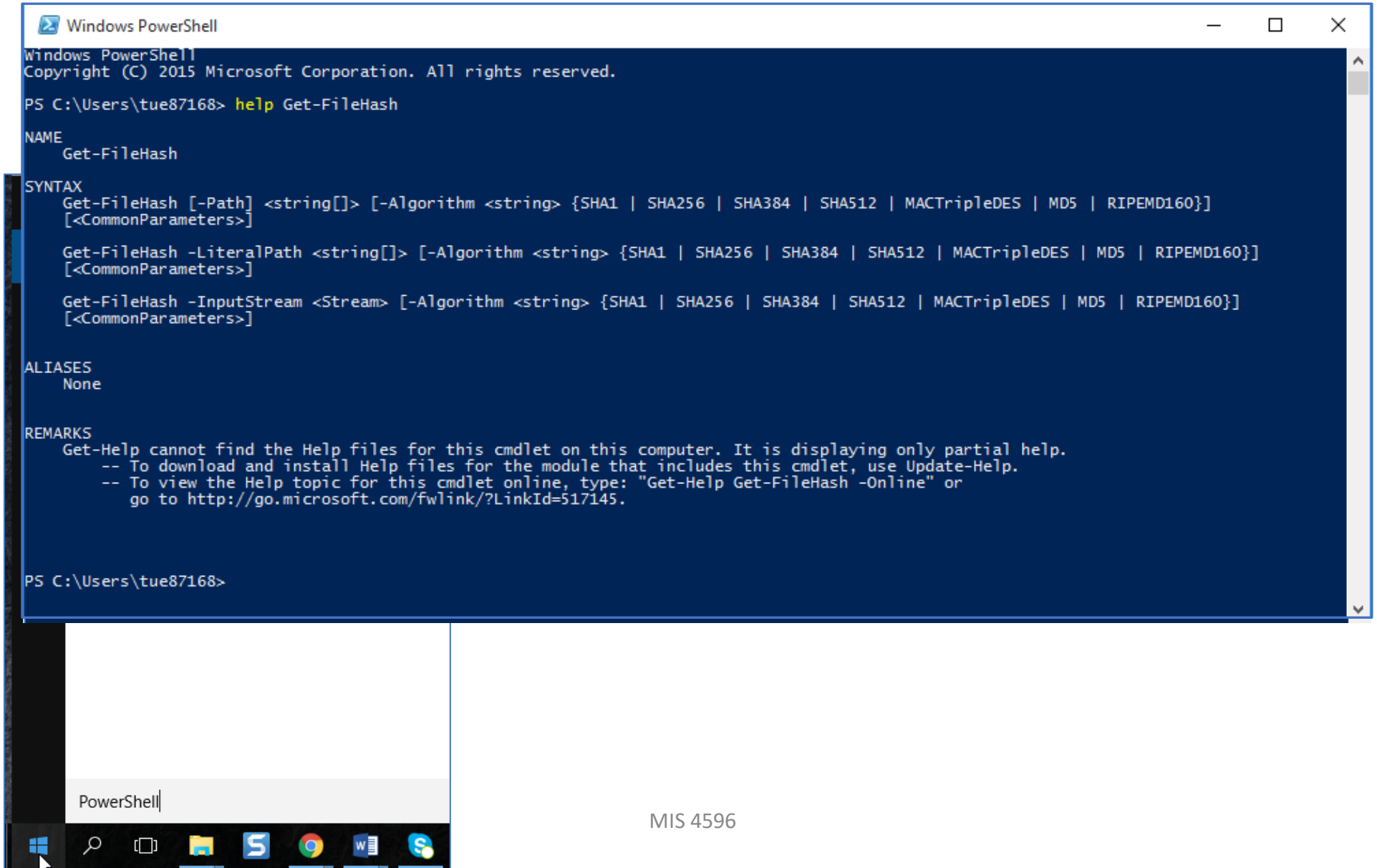
Testing the integrity of a file (e.g. program) from the internet…

Image Name	Download	Size	Version	sha256sum
Kali 64 bit	HTTP | Torrent	2.8G	2017.2	4556775bfb981ae64a3cb19aa0b73e8dcac6e4ba524f31c4bc14c9137b99725d



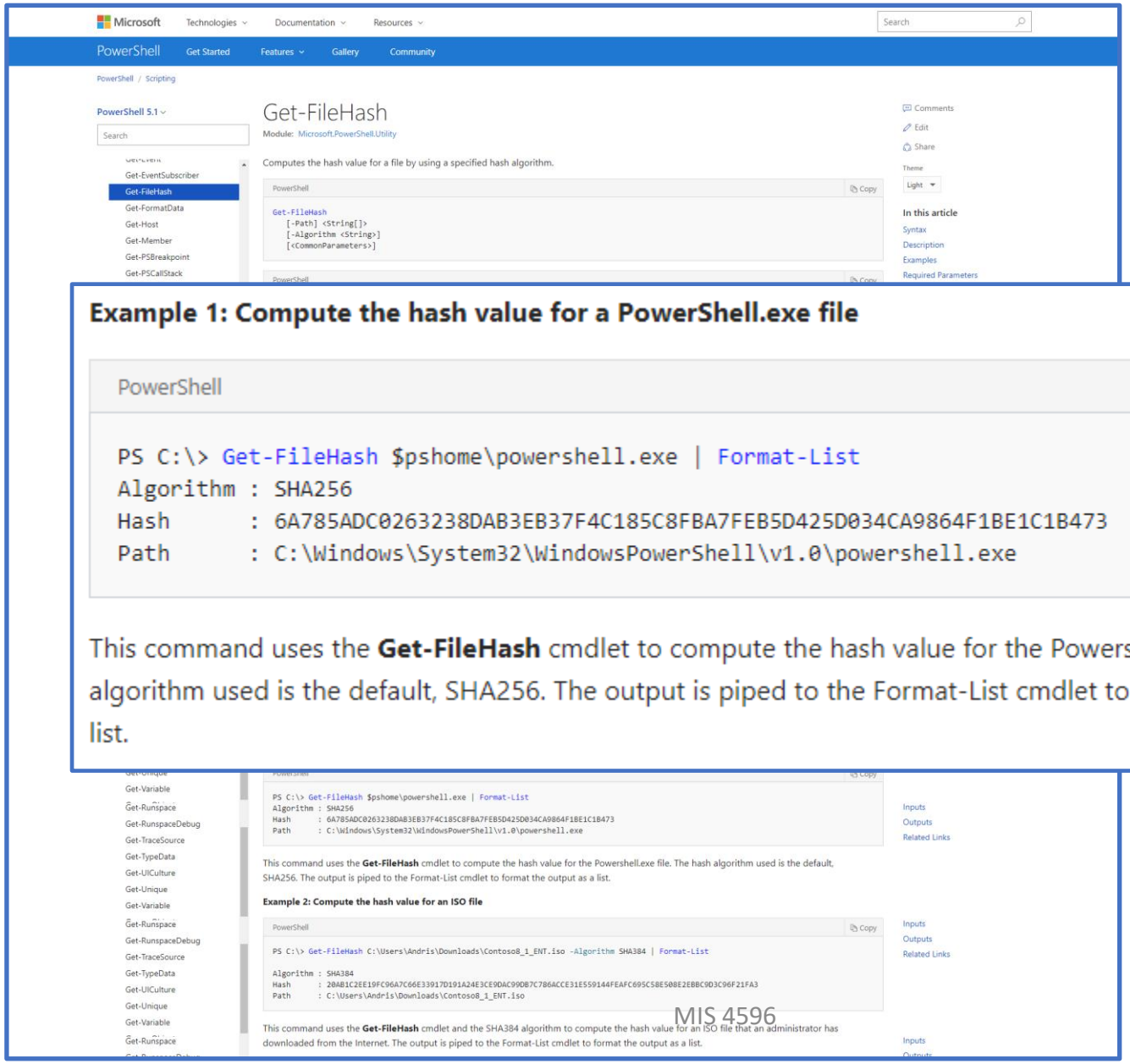
Is the Kali I downloaded the same Kali that was published?

One-way hash example…



One-way hash example…

https://docs.microsoft.com/en-us/powershell/module/microsoft.powershell.utility/get-filehash?view=powershell-5.1



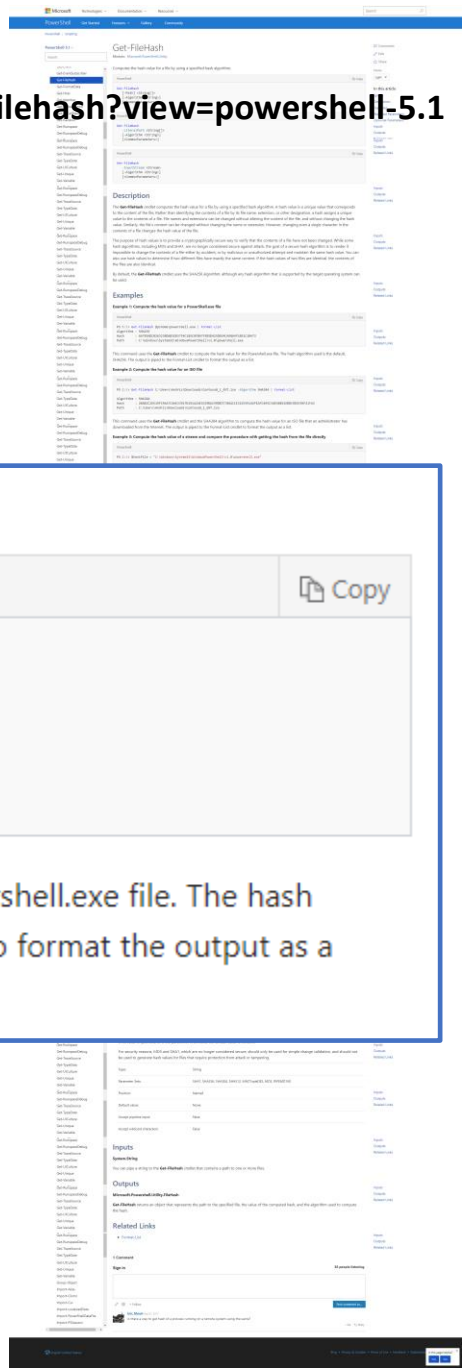
Example 1: Compute the hash value for a PowerShell.exe file

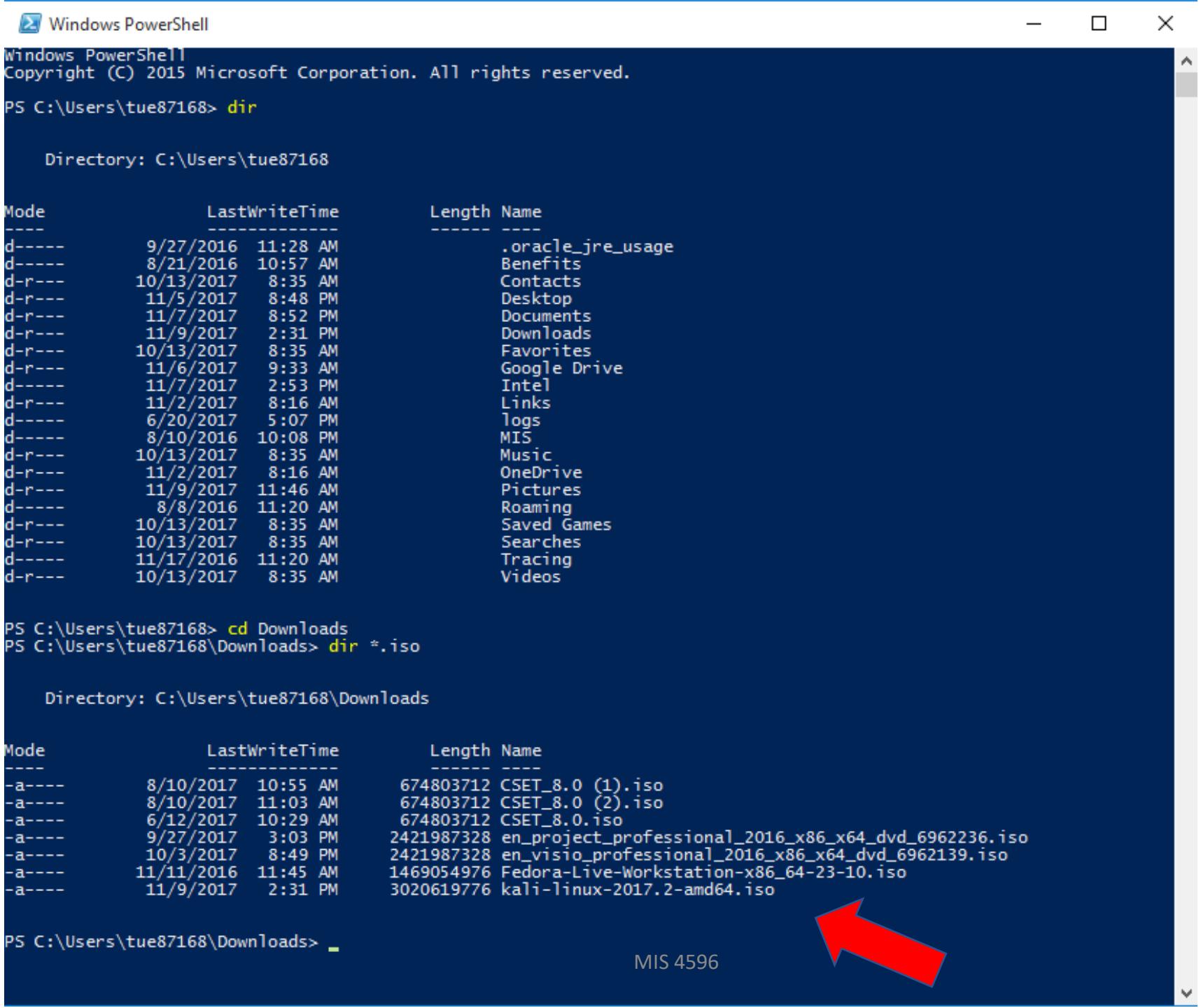
PowerShell

Copy

```
PS C:\> Get-FileHash $pshome\powershell.exe | Format-List
Algorithm : SHA256
Hash      : 6A785ADC0263238DAB3EB37F4C185C8FBA7FEB5D425D034CA9864F1BE1C1B473
Path      : C:\Windows\System32\WindowsPowerShell\v1.0\powershell.exe
```

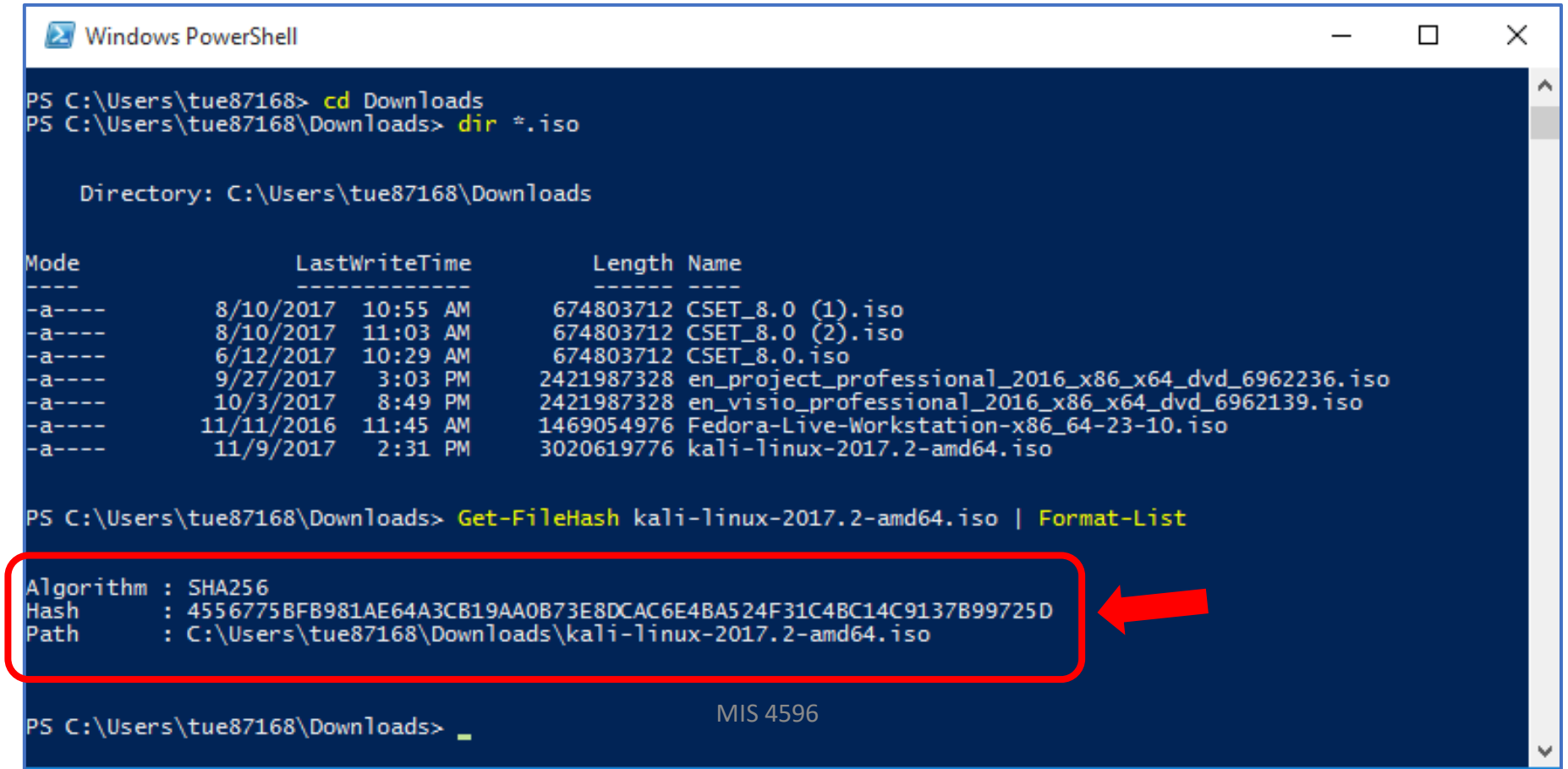
This command uses the **Get-FileHash** cmdlet to compute the hash value for the Powershell.exe file. The hash algorithm used is the default, SHA256. The output is piped to the Format-List cmdlet to format the output as a list.



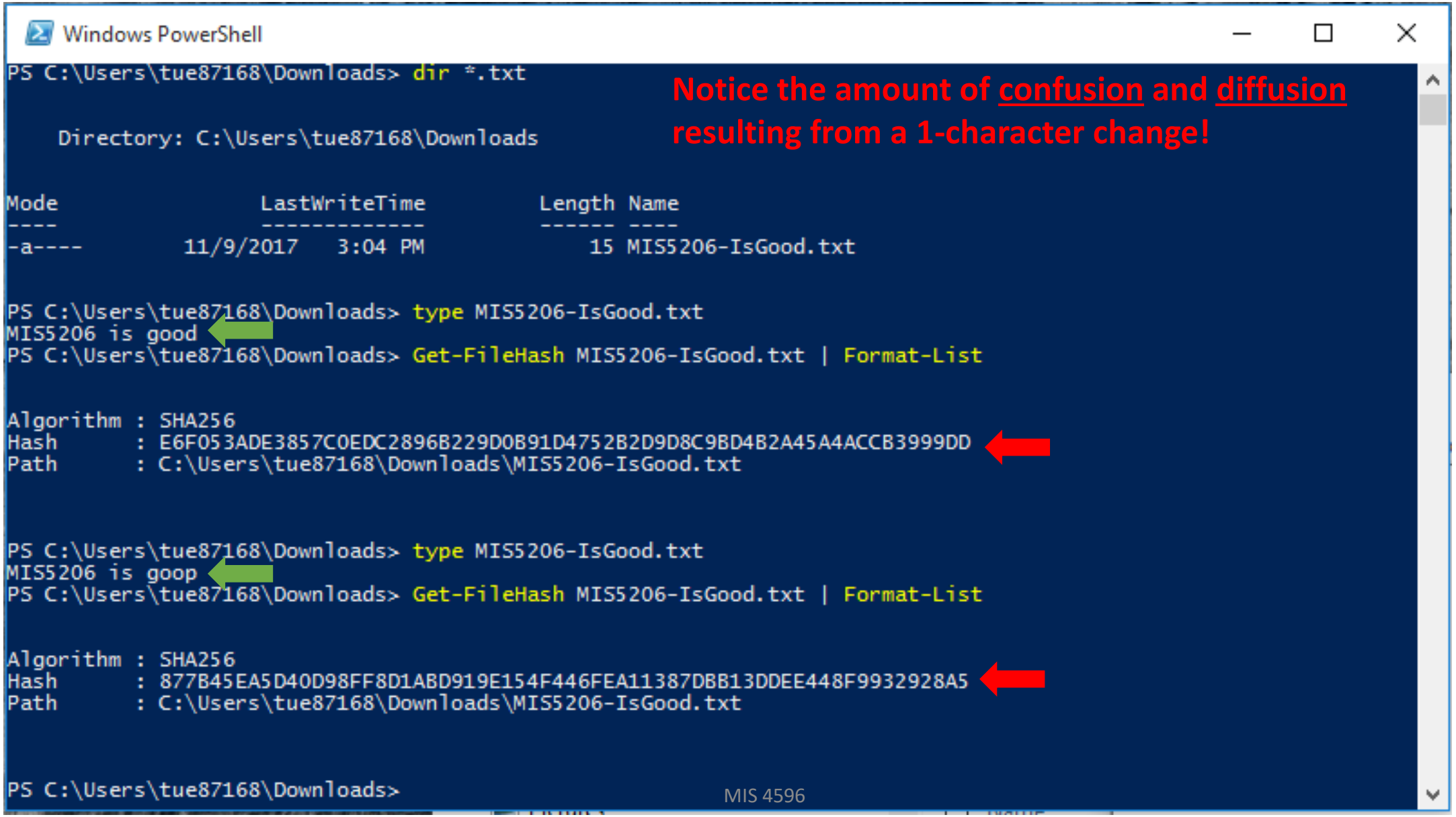

Windows PowerShell
Windows PowerShell
Copyright (C) 2015 Microsoft Corporation. All rights reserved.
PS C:\Users\tue87168> dir
Directory: C:\Users\tue87168
Mode LastWriteTime Length Name
d----- 9/27/2016 11:28 AM .oracle_jre_usage
d----- 8/21/2016 10:57 AM Benefits
d-r--- 10/13/2017 8:35 AM Contacts
d-r--- 11/5/2017 8:48 PM Desktop
d-r--- 11/7/2017 8:52 PM Documents
d-r--- 11/9/2017 2:31 PM Downloads
d-r--- 10/13/2017 8:35 AM Favorites
d-r--- 11/6/2017 9:33 AM Google Drive
d----- 11/7/2017 2:53 PM Intel
d-r--- 11/2/2017 8:16 AM Links
d----- 6/20/2017 5:07 PM logs
d----- 8/10/2016 10:08 PM MIS
d-r--- 10/13/2017 8:35 AM Music
d-r--- 11/2/2017 8:16 AM OneDrive
d-r--- 11/9/2017 11:46 AM Pictures
d----- 8/8/2016 11:20 AM Roaming
d-r--- 10/13/2017 8:35 AM Saved Games
d-r--- 10/13/2017 8:35 AM Searches
d----- 11/17/2016 11:20 AM Tracing
d-r--- 10/13/2017 8:35 AM Videos
PS C:\Users\tue87168> cd Downloads
PS C:\Users\tue87168\Downloads> dir *.iso
Directory: C:\Users\tue87168\Downloads
Mode LastWriteTime Length Name
-a---- 8/10/2017 10:55 AM 674803712 CSET_8.0 (1).iso
-a---- 8/10/2017 11:03 AM 674803712 CSET_8.0 (2).iso
-a---- 6/12/2017 10:29 AM 674803712 CSET_8.0.iso
-a---- 9/27/2017 3:03 PM 2421987328 en_project_professional_2016_x86_x64_dvd_6962236.iso
-a---- 10/3/2017 8:49 PM 2421987328 en_visio_professional_2016_x86_x64_dvd_6962139.iso
-a---- 11/11/2016 11:45 AM 1469054976 Fedora-Live-Workstation-x86_64-23-10.iso
-a---- 11/9/2017 2:31 PM 3020619776 kali-linux-2017.2-amd64.iso
PS C:\Users\tue87168\Downloads>

One-way hash example…

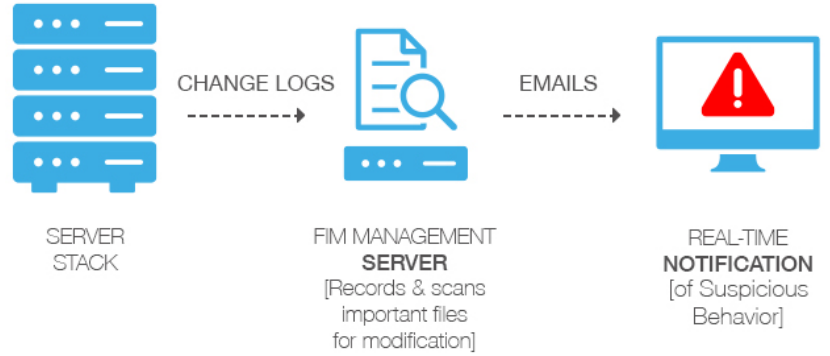
Image Name	Download	Size	Version	sha256sum
Kali 64 bit	HTTP | Torrent	2.8G	2017.2	4556775bfb981ae64a3cb19aa0b73e8dcac6e4ba524f31c4bc14c9137b99725d



One-way hash example…



File Integrity Monitoring



- An internal control process
- Validates the integrity of operating system and application software files
- Uses hash verification to compare the current file state and a known good baseline state
- Involves calculating and storing a hash value of a known good version of the file (“original baseline”)
- Compares the baseline with the calculated hash of the current state of the file to detect unauthorized changes

How are passwords stored in Linux?

Passwords are hashed!
Don't store plaintext passwords

```
cat /etc/passwd
root:x:0:0:root:/root:/bin/bash
daemon:x:1:1:daemon:/usr/sbin:/bin/sh
bin:x:2:2:bin:/bin:/bin/sh
sys:x:3:3:sys:/dev:/bin/sh
sync:x:4:65534:sync:/bin:/bin/sync
games:x:5:60:games:/usr/games:/bin/sh
man:x:6:12:man:/var/cache/man:/bin/sh
lp:x:7:7:lp:/var/spool/lpd:/bin/sh
mail:x:8:8:mail:/var/mail:/bin/sh
news:x:9:9:news:/var/spool/news:/bin/sh
uucp:x:10:10:uucp:/var/spool/uucp:/bin/sh
proxy:x:13:13:proxy:/bin:/bin/sh
www-data:x:33:33:www-data:/var/www:/bin/sh
backup:x:34:34:backup:/var/backups:/bin/sh
list:x:38:38:Mailing List Manager:/var/list:/bin/sh
irc:x:39:39:ircd:/var/run/ircd:/bin/sh
gnats:x:41:41:Gnats Bug-Reporting System (admin):/var/lib/gnats:/bin/sh
nobody:x:65534:65534:nobody:/nonexistent:/bin/sh
libuuid:x:100:101::/var/lib/libuuid:/bin/sh
dhcp:x:101:102::/nonexistent:/bin/false
syslog:x:102:103::/home/syslog:/bin/false
klog:x:103:104::/home/klog:/bin/false
sshd:x:104:65534::/var/run/sshd:/usr/sbin/nologin
msfadmin:x:1000:1000:msfadmin,,,:/home/msfadmin:/bin/bash
bind:x:105:113::/var/cache/bind:/bin/false
postfix:x:106:115::/var/spool/postfix:/bin/false
ftp:x:107:65534::/home/ftp:/bin/false
postgres:x:108:117:PostgreSQL administrator,,,:/var/lib/postgresql:/bin/bash
mysql:x:109:118:MySQL Server,,,:/var/lib/mysql:/bin/false
tomcat55:x:110:65534::/usr/share/tomcat5.5:/bin/false
distccd:x:111:65534::/:/bin/false
user:x:1001:1001:just a user,111,,:/home/user:/bin/bash
service:x:1002:1002:,,,:/home/service:/bin/bash
telnetd:x:112:120::/nonexistent:/bin/false
proftpd:x:113:65534::/var/run/proftpd:/bin/false
statd:x:114:65534::/var/lib/nfs:/bin/false
```

Password file

```
cat /etc/shadow
root:$1$/avpfBJ1$x0z8w5UF9Iv./DR9E9Lid.:14747:0:99999:7:::
daemon:*:14684:0:99999:7:::
bin:*:14684:0:99999:7:::
sys:$1$fUX6BPOt$Miyc3UpOzQJqz4s5wFD9l0:14742:0:99999:7:::
sync:*:14684:0:99999:7:::
games:*:14684:0:99999:7:::
man:*:14684:0:99999:7:::
lp:*:14684:0:99999:7:::
mail:*:14684:0:99999:7:::
news:*:14684:0:99999:7:::
uucp:*:14684:0:99999:7:::
proxy:*:14684:0:99999:7:::
www-data:*:14684:0:99999:7:::
backup:*:14684:0:99999:7:::
list:*:14684:0:99999:7:::
irc:*:14684:0:99999:7:::
gnats:*:14684:0:99999:7:::
nobody:*:14684:0:99999:7:::
libuuid:!:14684:0:99999:7:::
dhcp:*:14684:0:99999:7:::
syslog:*:14684:0:99999:7:::
klog:$1$f2ZVMS4K$R9XkI.CmLdHhdUE3X9jqP0:14742:0:99999:7:::
sshd:*:14684:0:99999:7:::
msfadmin:$1$XN10Zj2c$Rt/zzCW3mLtUWA.ihZjA5/:14684:0:99999:7:::
bind:*:14685:0:99999:7:::
postfix:*:14685:0:99999:7:::
ftp:*:14685:0:99999:7:::
postgres:$1$Rw35ik.x$MgQgZUuO5pAoUvfJhfcYe/:14685:0:99999:7:::
mysql:!:14685:0:99999:7:::
tomcat55:*:14691:0:99999:7:::
distccd:*:14698:0:99999:7:::
user:$1$HESu9xrH$k.o3G93DGoXIiQKkPmUgZ0:14699:0:99999:7:::
service:$1$kR3ue7JZ$7GxELDupr5Ohp6cjZ3Bu//:14715:0:99999:7:::
telnetd:*:14715:0:99999:7:::
proftpd:!:14727:0:99999:7:::
statd:*:15474:0:99999:7:::
```

Shadow file

Cryptanalysis Attacks

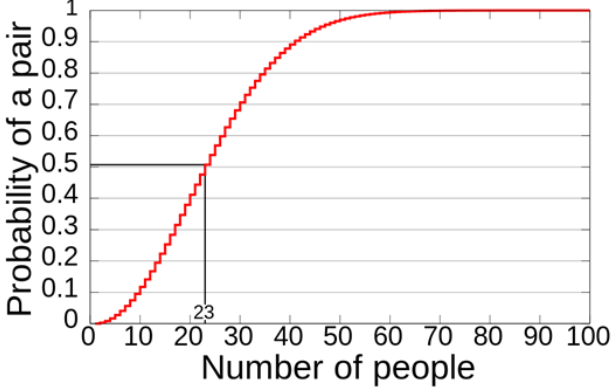
- Brute force
  - Trying all key values in the keyspace
- Frequency Analysis
  - Guess values based on frequency of occurrence
- Dictionary Attack
  - Find plaintext based on common words
- Known Plaintext
  - Format or content of plaintext available
- Chosen Plaintext
  - Attack can encrypt chosen plaintext
- Chosen Ciphertext
  - Decrypt known ciphertext to discover key
- Random Number Generator (RNG) Attack
  - Predict initialization vector used by an algorithm
- Social Engineering
  - Humans are the weakest link

Cryptanalysis Attacks

- Collisions
  - *Two different messages with the same hash value*
  - Based on the "birthday paradox"
  - Hash algorithms should be resistant to this attack

The birthday paradox, also known as the birthday problem, states that in a random group of 23 people, there is about a 50 percent chance that two people have the same birthday.

Is the Birthday Attack Real?



There are multiple reasons why this seems like a paradox

One is that when in a room with 22 other people, if a person compares her/his birthday with the birthdays of the other people it would make for only 22 comparisons—only 22 chances for people to share the same birthday.

When all 23 birthdays are compared against each other, it makes for much more than 22 comparisons. How much more?

Well, the first person has 22 comparisons to make, but the second person was already compared to the first person, so there are only 21 comparisons to make.

The third person then has 20 comparisons, the fourth person has 19 and so on.

If you add up all possible comparisons (22 + 21 + 20 + 19 + … +1) the sum is 253 comparisons, or combinations. Consequently, each group of 23 people involves 253 comparisons, or 253 chances for matching birthdays.

MD5 (Message Digest 5)

- A 128-bit hash algorithm, still in common use
- Has been broken
- 128-bit hash, but only need $2^{128/2} = 2^{64}$ to find a collision
- Not strong enough for modern computers

SHA -1 (Security Hash Algorithm 1)

- A 160-bit hash algorithm, still in common use
- Has been broken
- 160-bit hash, but only need $2^{160/2} = 2^{80}$ to find a collision
- No longer strong enough for modern computers

Windows PowerShell

```
PS C:\Users\Dave\Desktop\MD5-Hash-Collision-Example> get-filehash ProgramA.exe -Algorithm SHA256

Algorithm       Hash                                                                   Path
---------       ----                                                                   ----
SHA256          60D13913155644883F130B85EB24D778314014C9479AEDB5F6323BF38AD3A451       C:\Users\Dave\Desktop\MD5-Hash-Collision-Example\ProgramA.exe


PS C:\Users\Dave\Desktop\MD5-Hash-Collision-Example> get-filehash ProgramB.exe -Algorithm SHA256

Algorithm       Hash                                                                   Path
---------       ----                                                                   ----
SHA256          1316543942A8C6CD754855500CD37068EDBBD8B31C4979D2825A4E799FED6102       C:\Users\Dave\Desktop\MD5-Hash-Collision-Example\ProgramB.exe
```

```
Hello, world!

(press enter to quit)
```

ProgramA run

```
This program is evil!!!
Erasing hard drive...1Gb...2Gb... just kidding!
Nothing was erased.

(press enter to quit)
```

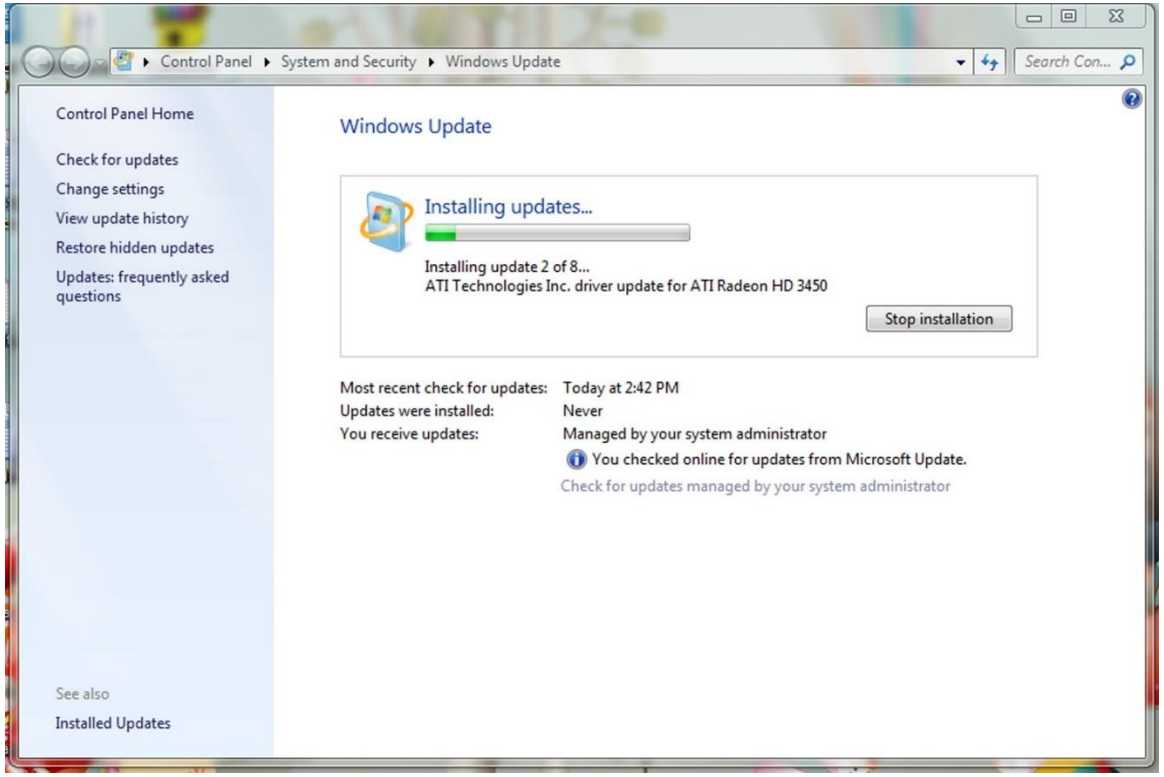
ProgramB run

Windows PowerShell

```
PS C:\Users\Dave\Desktop\MD5-Hash-Collision-Example> get-filehash ProgramA.exe -Algorithm MD5

Algorithm       Hash                                                                   Path
---------       ----                                                                   ----
MD5             CDC47D670159EEF60916CA03A9D4A007                                       C:\Users\Dave\Desktop\MD5-Hash-Collision-Example\ProgramA.exe


PS C:\Users\Dave\Desktop\MD5-Hash-Collision-Example> get-filehash ProgramB.exe -Algorithm MD5

Algorithm       Hash                                                                   Path
---------       ----                                                                   ----
MD5             CDC47D670159EEF60916CA03A9D4A007                                       C:\Users\Dave\Desktop\MD5-Hash-Collision-Example\ProgramB.exe
```

The malware Flame used a MD5 hash collision to hijack Microsoft Windows Update and spread itself across networks

- Flame collected **audio, keystrokes, screenshots** which it sent to a malicious server
- Found a collision within a single millisecond
- Cost ~$200k computing time just for 1ms
- Attributed to advanced persistent threat group Equation Group
- Espionage attacks on countries in and around Iran



PWN (verb)

1. An act of dominating an opponent.

2. Great, ingenious; applied to methods and objects.

- Originally dates back to the days of WarCraft, when a map designer mispelled "Own" as "Pwn“

- What was originally suppose to be "player has been owned." was "player has been pwned"

Use of the term “Pwn’ grew and is now used throughout the online world, especially in online games:

1. "I pwn these guys on battlenet"

2. "This strategy pwns!" or "This game pwn."

https://www.urbandictionary.com/define.php?term=pwn

Equation Group’s Flame malware won 2012 “Epic 0wnage” Pwnie award

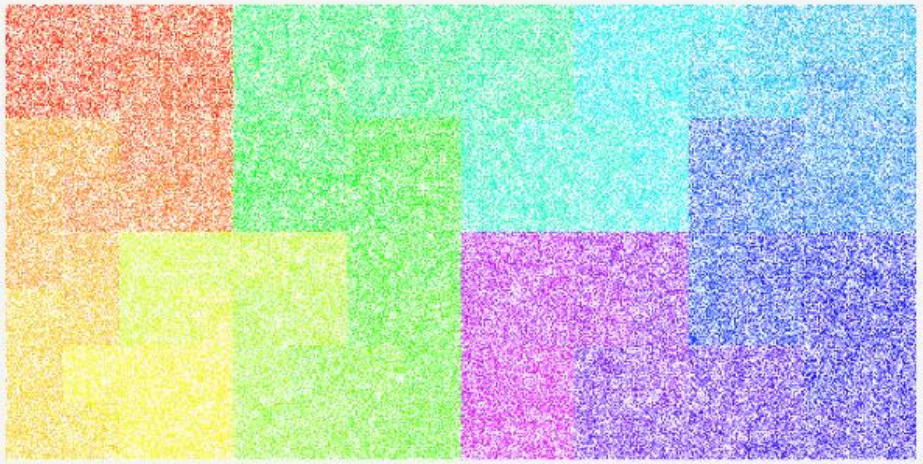
- Pwnie Awards recognize both excellence and incompetence in the field of information security
- Awards are presented yearly at the Black Hat Security Conference

Hashing algorithms are used for browser ssl (secure sockets layer)

- In 2014, many sites were still using SHA-1, at the time known to be dangerously vulnerable
- Google declared state of emergency to push companies to upgrade

Why Google is Hurrying the Web to Kill SHA-1

published by Eric Mill on September 7, 2014, 58 comments



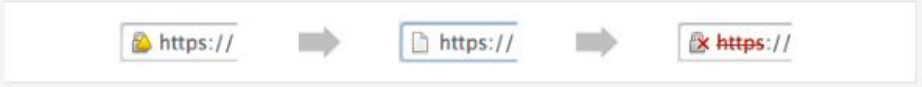
Hilbert map of hashing algorithms, by Ian Boyd

Most of the secure web is using an insecure algorithm, and Google's just declared it to be a slow-motion emergency.

Something like 90% of websites that use SSL encryption — https:// — use an algorithm called SHA-1 to protect themselves from being impersonated. This guarantees that when you go to https://www.facebook.com , you're visiting the real Facebook and not giving your password to an attacker.

Unfortunately, **SHA-1 is dangerously weak**, and has been for a long time. It gets weaker every year, but remains widely used on the internet. Its replacement, SHA-2, is strong and supported just about everywhere.

Google recently announced that if you use Chrome, then you're about to start seeing a progression of warnings for many secure websites:



What's about to befall websites with SHA-1 certificates that expire in 2017, in Chrome.

The first set of warnings will hit before Christmas, and will keep getting more stern over the

SHA-2 uses 224, **256**, 384, and 512-bit hashes

- But… it is built using the design of SHA-1, and prone to the same weaknesses
- It's believed to be a matter of time before SHA-2 is also exploited

- SHA-3 was just ratified recently by NIST, the U.S. National Institute of Standards and Technology
  - It was the result of a six-year hashing competition. Also uses 224-, 256-, 384-, 512-bit hashes

Why does this matter for businesses?

Business needs a reliable way to prove integrity of data, files, programs, that can be trusted

Agenda

- ✓Some useful Linux commands
- ✓Symmetric cryptography
- ✓Block versus Stream ciphers
- ✓Block ciphers
- ✓Block ciphers mode of operations
- ✓Hashes