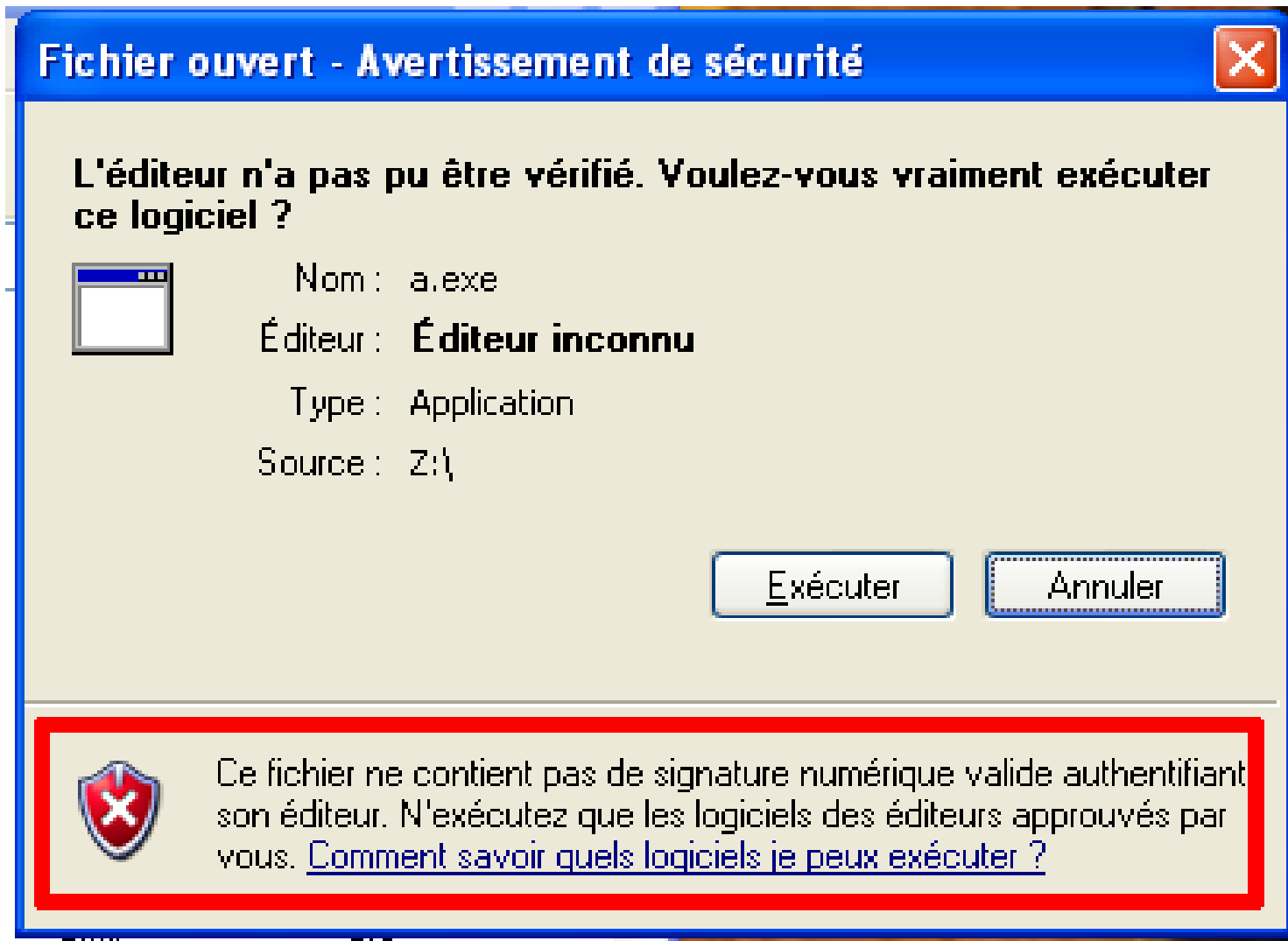


# Hash functions

1. Hash functions are everywhere
2. What a hash function is and why we need it
3. What their properties are and why it is not trivial to design a hash function
4. Hash functions in the real world: From MD5 to SHA-3

# Hash functions are everywhere



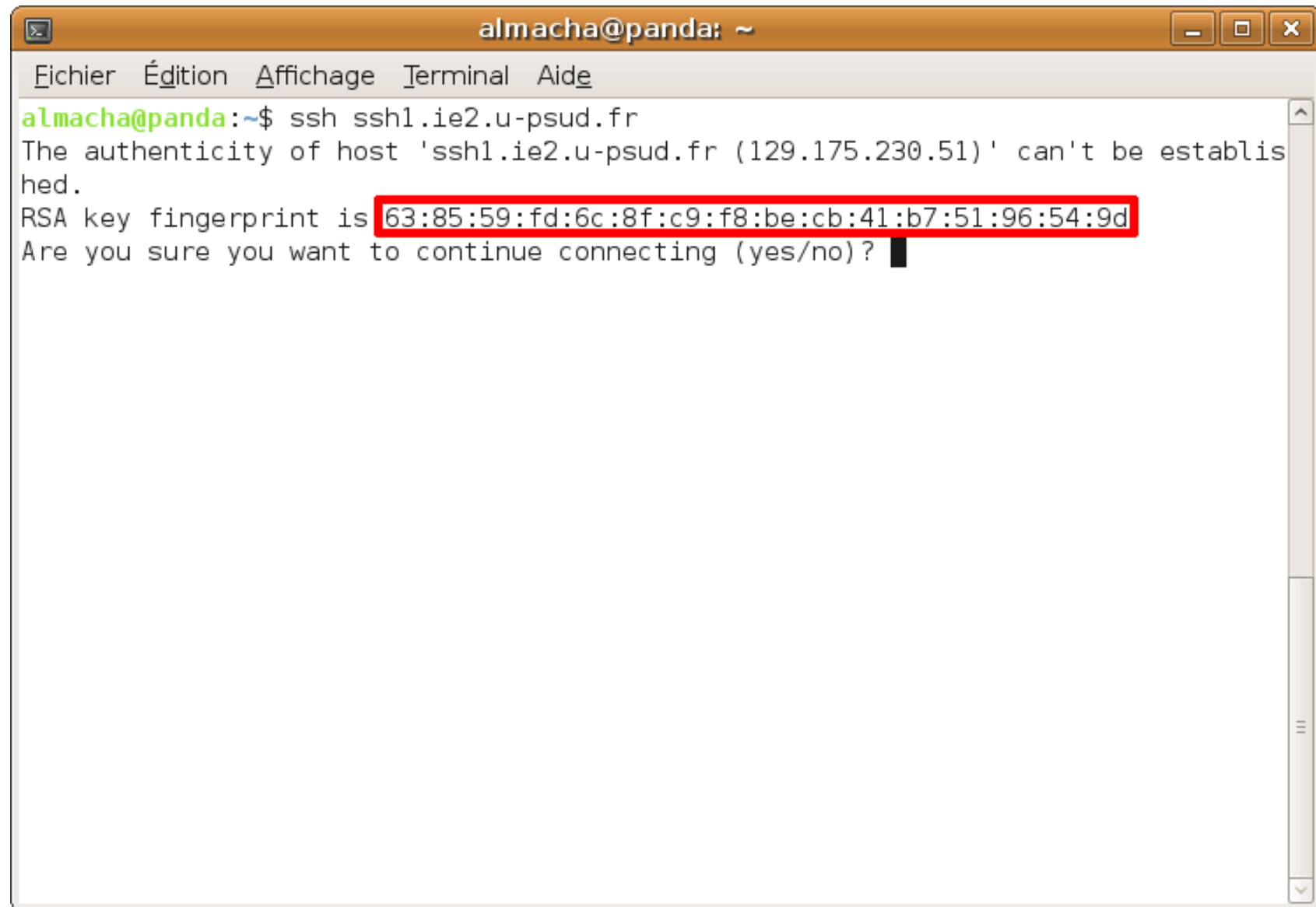
# Hash functions are everywhere

The screenshot shows the Vuze torrent client interface. The main window displays a list of torrents. The torrent 'eclipse-jee-galileo-win32.zip' is highlighted, and its verification progress 'Vérification: 51,0 %' is shown in a red box.

Rang de source	Indicateur	#	Nom	Cor	Taille	État	Sources	Clients
1ère Priorité 5153	🟢	1			21,30 Go	Source	0 (262)	16 (75)
	⚫	2			702,10 Mo	En attente	0 (2436)	0 (735)
5177	⚫	3			801,35 Mo	En attente	0 (404)	0 (119)
21910	🟢	4			1,38 Go	Source	0 (26)	19 (22)
464	⚫	5	eclipse-jee-galileo-win32.zip		189,32 Mo	Vérification: 51,0 %	0 (328)	0 (24)
21972	🟢	6			3,75 Go	Source	0 (18)	12 (15)
52401	🟢	7			5,44 Go	Source	0 (654)	46 (171)
	⚫	8			105,27 Mo	En attente	0	0 (0)
14792	⚫	9			4,36 Go	En attente	0 (16)	0 (10)
91	⚫	10			696,71 Mo	En attente	0 (640)	0 (20)
68	⚫	11			698,35 Mo	En attente	0 (224)	0 (6)
277	⚫	12			698,95 Mo	En attente	0 (2120)	0 (118)

Vuze 3.1.1.0 | Boost d'Amis | Taux | NAT OK | 565 284 Utilisateurs | IPs: 0 - 0/0/0 | [100K] 293 o/s | [15K] 14,9 ko/s

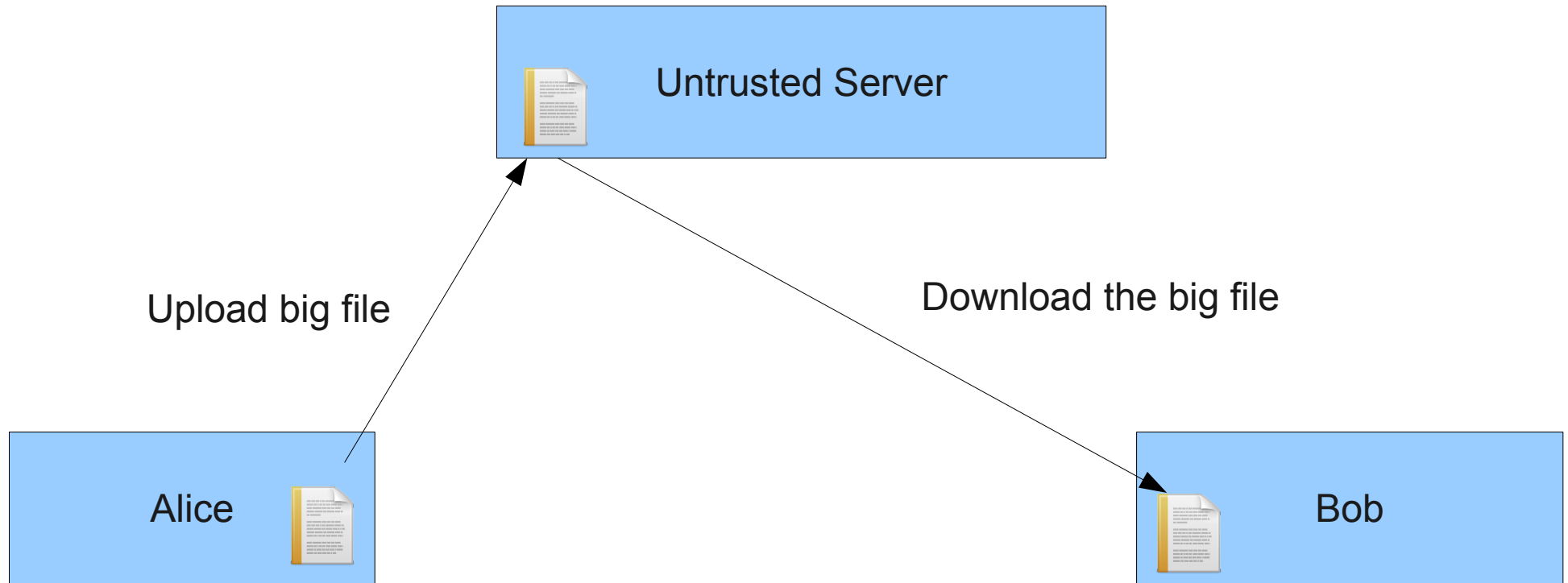
# Hash functions are everywhere



A terminal window titled 'almacha@panda: ~' with a menu bar containing 'Fichier', 'Édition', 'Affichage', 'Terminal', and 'Aide'. The terminal shows the command 'ssh ssh1.ie2.u-psud.fr' being executed. It displays a warning about host authenticity and an RSA key fingerprint, which is highlighted with a red box. The prompt asks for confirmation to continue connecting.

```
almacha@panda: ~$ ssh ssh1.ie2.u-psud.fr
The authenticity of host 'ssh1.ie2.u-psud.fr (129.175.230.51)' can't be established.
RSA key fingerprint is 63:85:59:fd:6c:8f:c9:f8:be:cb:41:b7:51:96:54:9d
Are you sure you want to continue connecting (yes/no)?
```

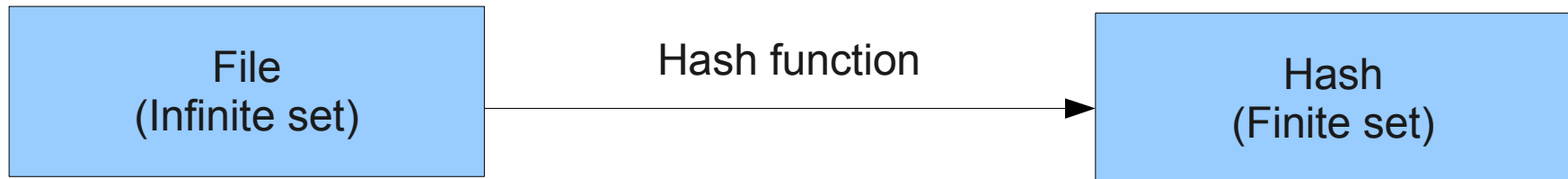
# Without hash functions



Is the file downloaded by Bob the same as Alice uploaded?

→ Same problem with mirrors (secondary servers) for Linux distributions.

# What is a hash function?



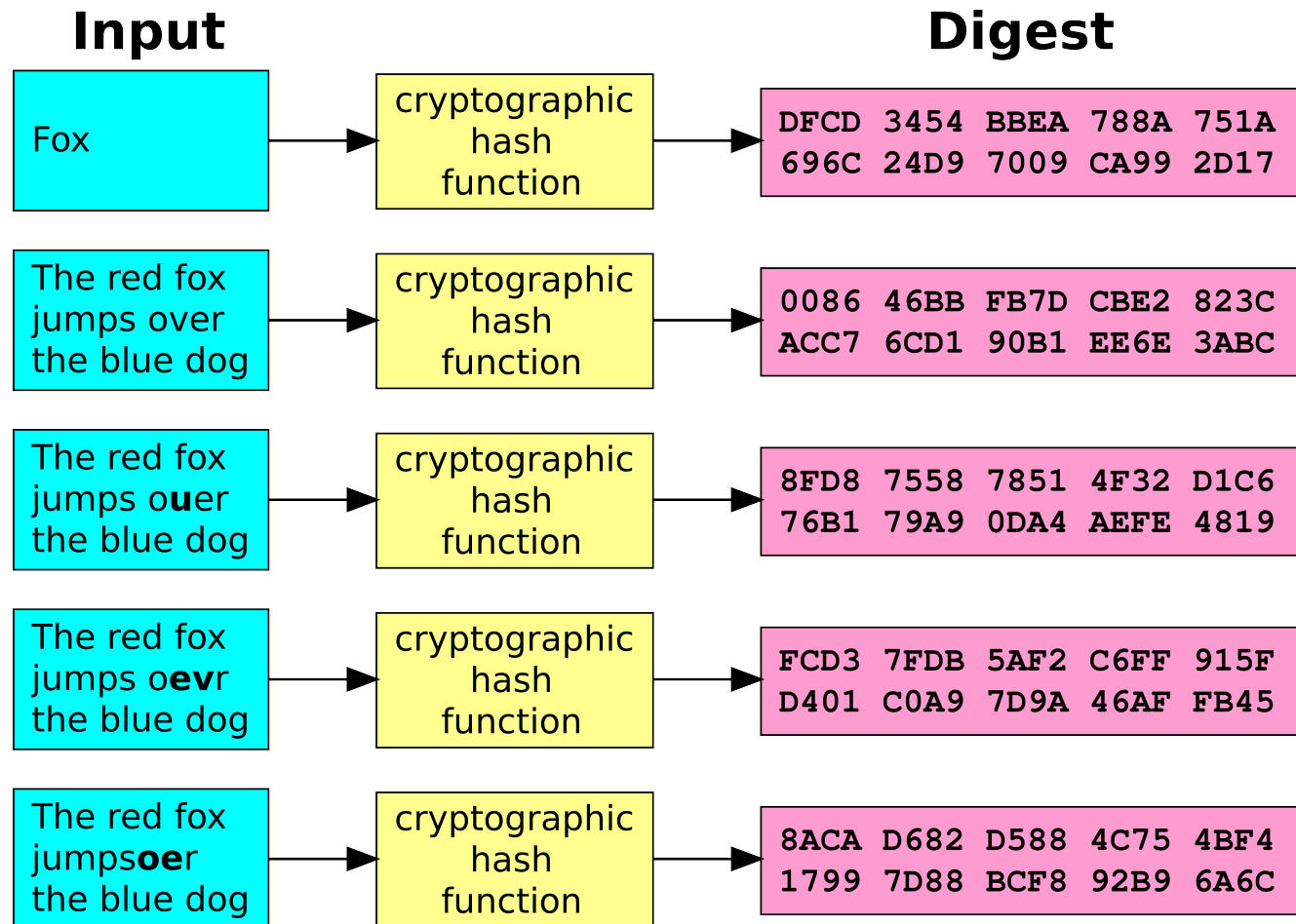
A hash function is a function that takes an unbounded string (like a file) and returns a small fixed-length (smaller than 1KB) string.

Example: The SHA-256 hash function takes a file and returns a 256 bits string.

```
almacha@panda: ~  
Eichier  Édition  Affichage  Terminal  Aide  
almacha@panda:~$ sha256sum A.java  
d24cedcaf539ccc4d96bd228b91373f1907a8a4431663c772e82b2fd9e2c055d  A.java  
almacha@panda:~$
```

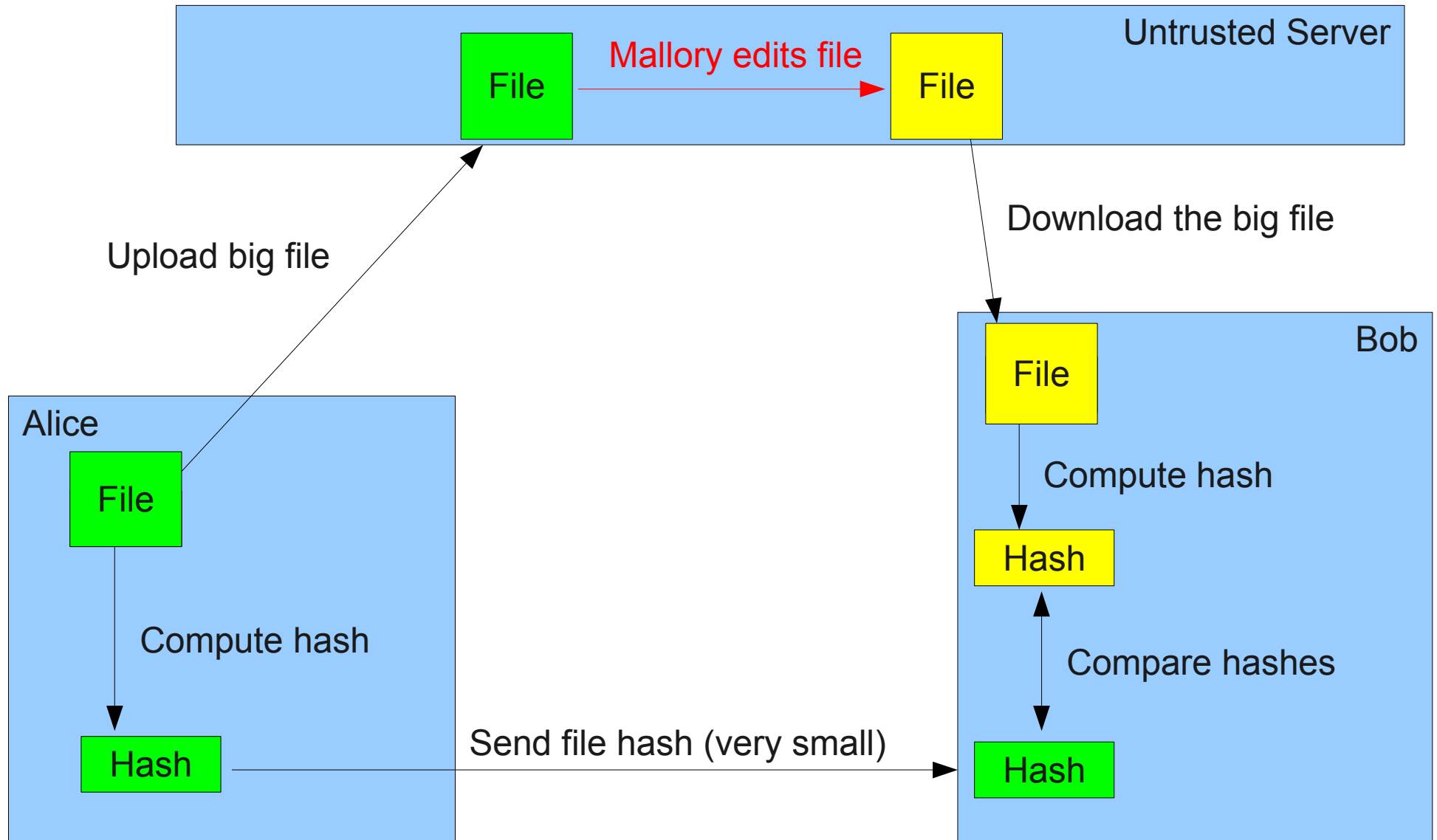
A screenshot of a terminal window titled "almacha@panda: ~". The window has a menu bar with "Eichier", "Édition", "Affichage", "Terminal", and "Aide". The terminal shows the command "sha256sum A.java" being executed, resulting in the output "d24cedcaf539ccc4d96bd228b91373f1907a8a4431663c772e82b2fd9e2c055d A.java". The prompt "almacha@panda:~\$" is visible before and after the command.

# We want to check file integrity



- Same input => Same hash
- Different input => Probably different hash (unless extremely unlucky)

# Why hash functions?





# A trivial (and bad) hash function

Let  $A[1..n]$  be an array of characters (as integers in  $[0,255]$ ) containing the input file.

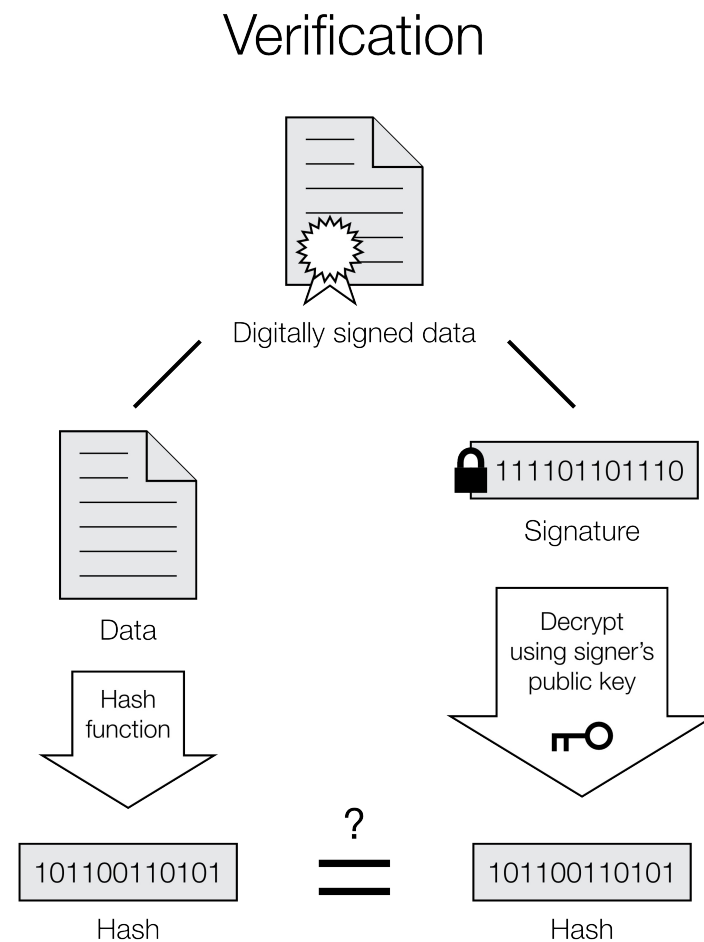
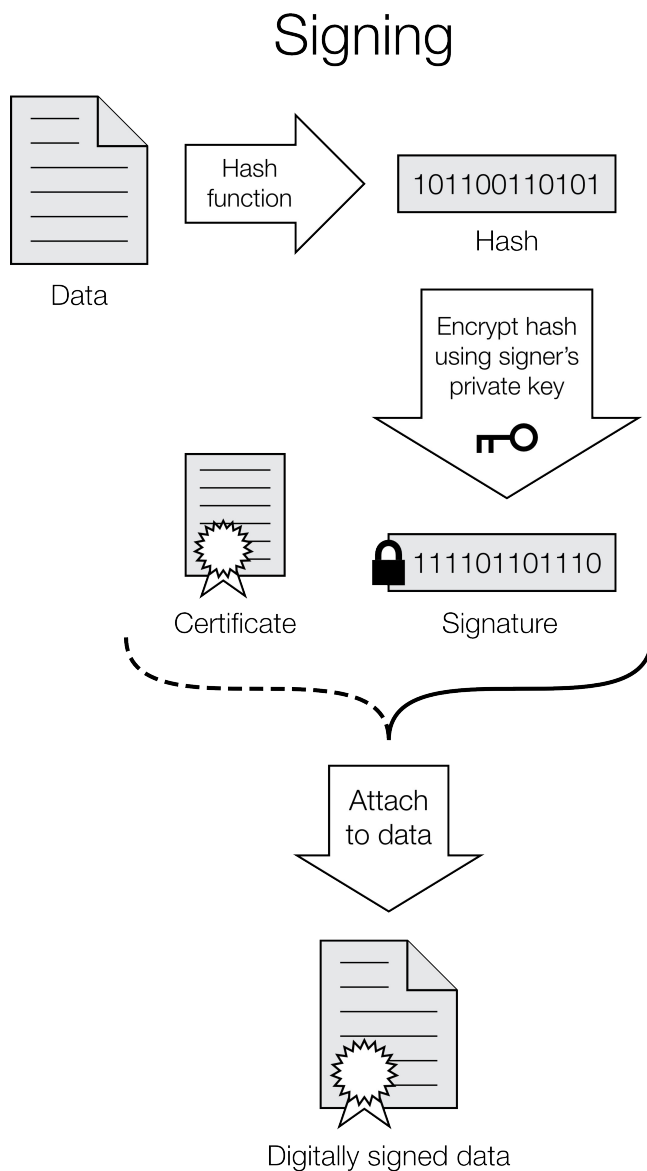
Consider the following hash function:

```
Function  $f(A[1..n]) =$   
  Let  $S = 0$   
  For  $i$  from 1 to  $n$  :  
     $S \leftarrow (S + A[i]) \bmod 1000$   
  Return  $S$ 
```

# Required properties

- It is infeasible to find a message that has a given hash. (*preimage resistance*)
- It is infeasible to modify a message without changing its hash. (*second preimage resistance*)
- It is infeasible to find two different messages with the same hash. (*collision resistance*)

# Digital signature



# Birthday problem

57 people in a room

$P(\text{someone has his birthday the same day as you})$   
 $= 14\%$

$P(\text{at least 2 people in the room have their birthday the same day}) = 99\%$

# Birthday problem

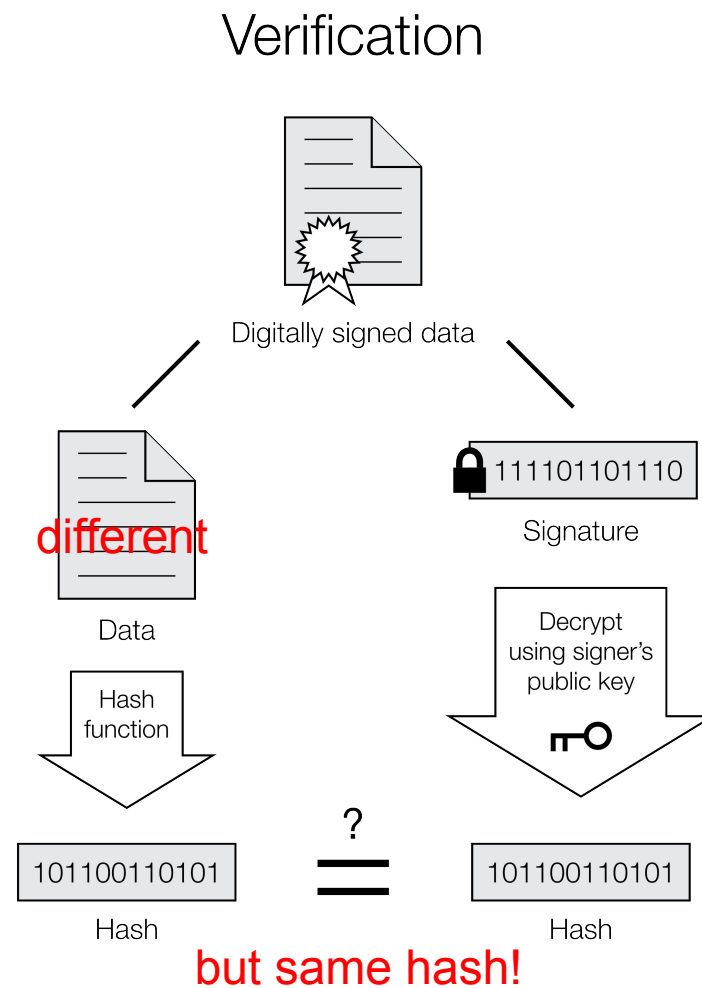
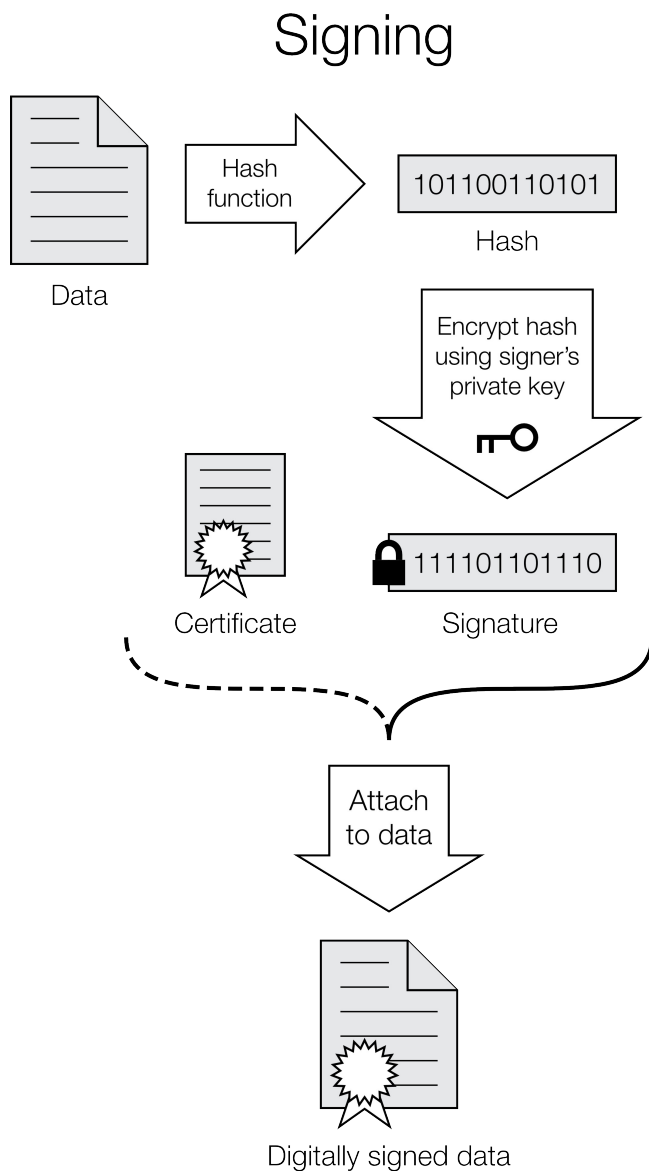
Hash function with  $n$  possible outputs

Given a hash,  
time to find a file with this hash =  
 $0.5 \times n$

Time to find 2 files with the same hash =  
 $1.25 \times \sqrt{n}$

=> The hash has to be twice as long.

# Birthday attack



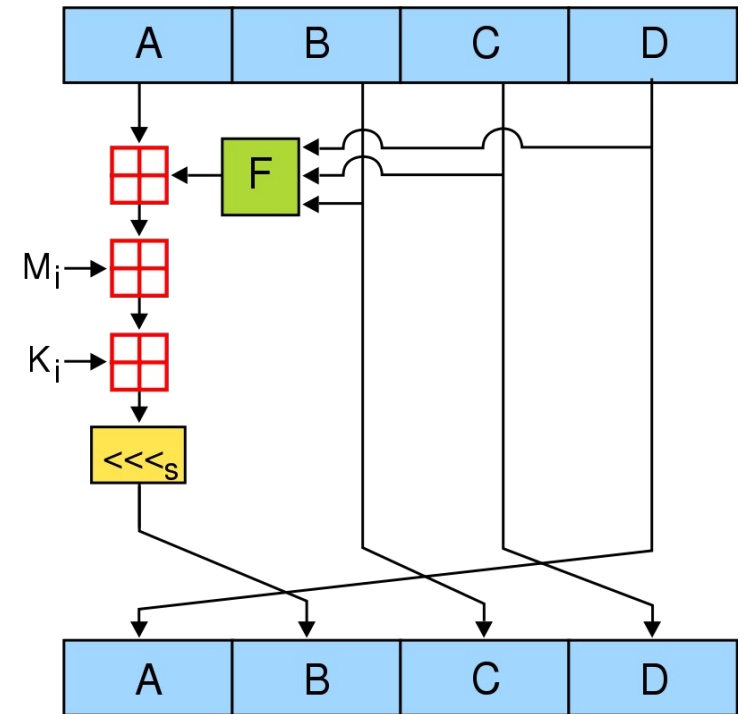
If the hashes are equal, the signature is valid.

# What we mean by broken

We say that a hash function is broken if it is possible to compute a collision faster than expected with a naive brute force attack.

# MD4

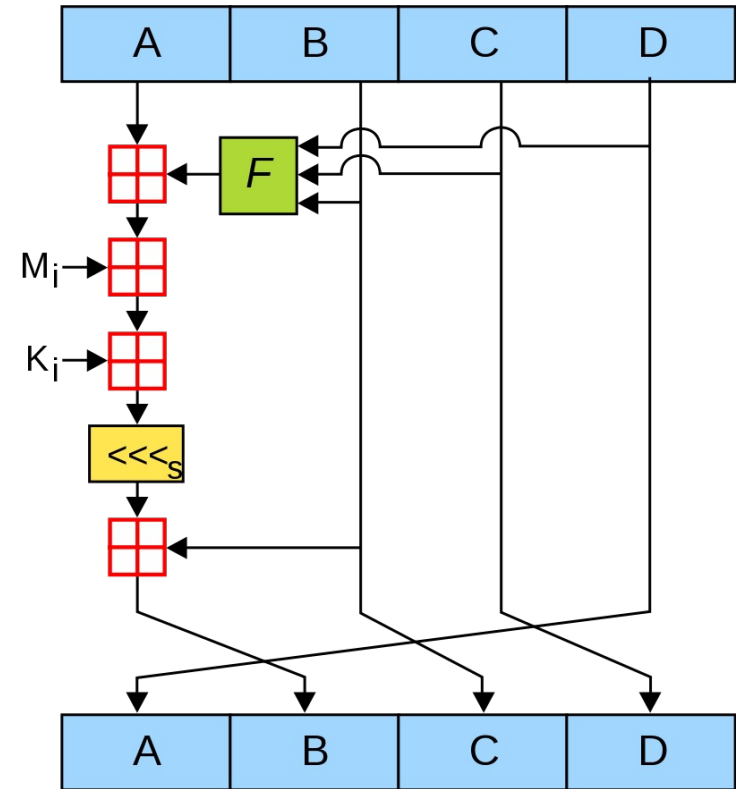
- Designed by Professor Ronald Rivest of MIT in 1990.
- Used by eMule to match files.
- Used by Windows NT to store passwords.
- Used by rsync to compare files.
- Weakness discovered in 1991.
- Generating a collision now takes only a few seconds.





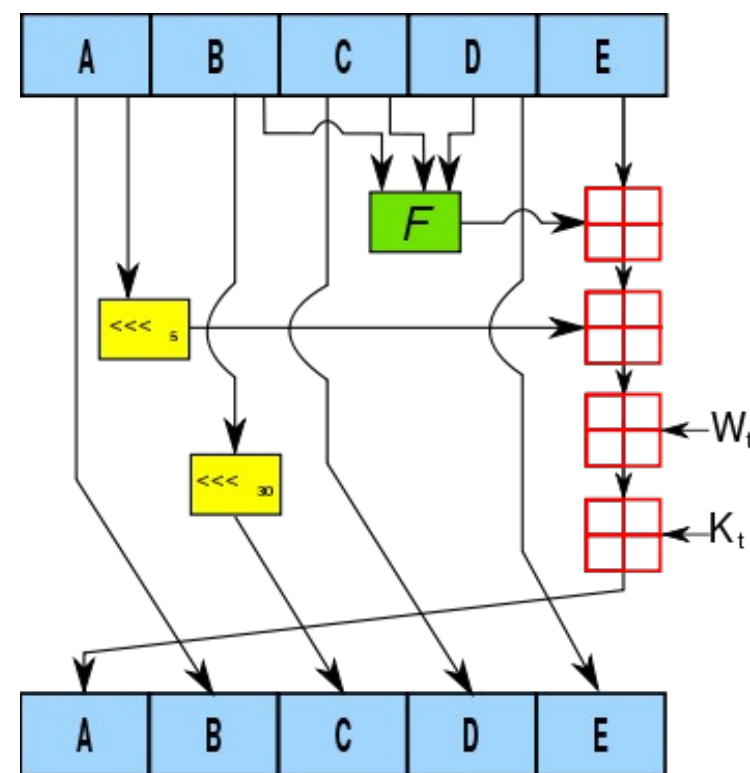
# MD5

- Designed in 1991 to replace MD4.
- Widely used for digital signatures, SSL certificates, password, file integrity checking...
- Minor flaw discovered in MD5 design in 1996.
- Serious flaw discovered in 2004.
- Successful attack on SSL in 2008.
- Deprecated by US Government and SSL researchers. Abandoned by VeriSign when issuing SSL certificates.



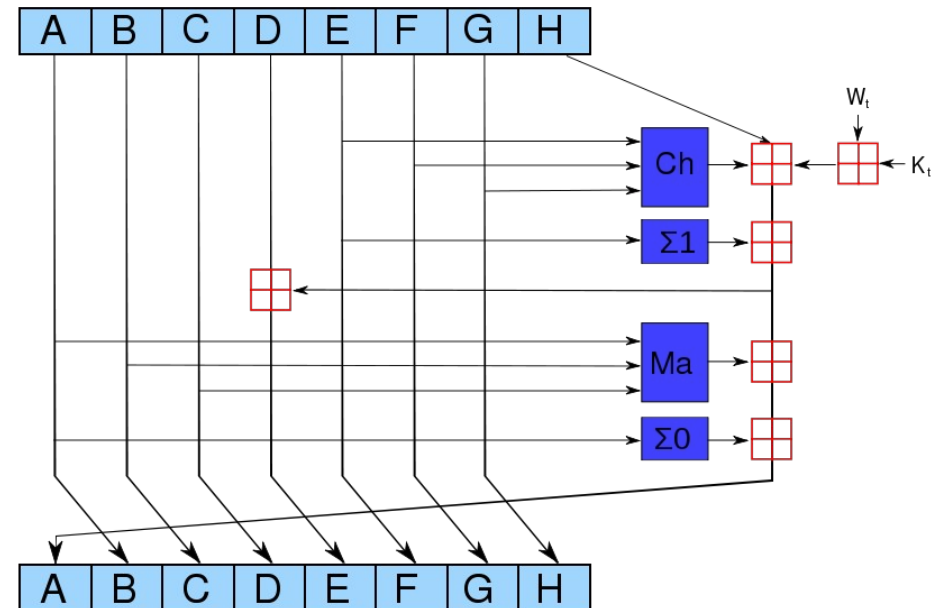
# SHA-1

- A earlier function called SHA-0 was released by the NSA in 1993, but was withdrawn and a modified version, SHA-1, was published in 1995.
- SHA-1 is widely used like MD5, which it has replaced in most SSL certificates.
- SHA-1 is used by BitTorrent to find files (when using DHT) and verify file chunks.
- The SHA-0 flaw was discovered in 1998.
- Algorithms to find collision faster than brute force were found in 2005.
- Better algorithm found in 2006, but time required to compute a collision is still too high in practice (hence the now abandoned BOINC project).
- Since 2004, NIST is trying to phase out the use of SHA-1 and replace it with SHA-2.



# SHA-2

- First published in 2001 by the NSA.
- Comes in 4 variants: SHA-224, SHA-256, SHA-384, and SHA-512 (numbers of bits in hash).
- SHA-2 are the best algorithms currently available.
- Not yet widely used because not as widely supported as MD5 and SHA-1.
- SHA-256 is used to authenticate Debian packages.
- SHA-512 is used to check archival video integrity from the International Criminal Tribunal of the Rwandan genocide.
- US Government agencies are required to switch to SHA-2.



# The future: SHA-3

- Open competition launched by NIST in 2007.
- List of candidates published in 2008.
- List of 14 candidates accepted for round 2 published in 2009.
- Announcement of final candidates scheduled for 2010.
- Proclamation of the winner in 2012.

# Conclusion

- Don't use MD5: it's now broken.
- If compatibility is a concern, use SHA-1, but don't plan it for the long term.
- If you use only modern systems or if your program can embed SHA-2, then choose SHA-2 (use SHA-256 or 512).
- Follow the SHA-3 competition!