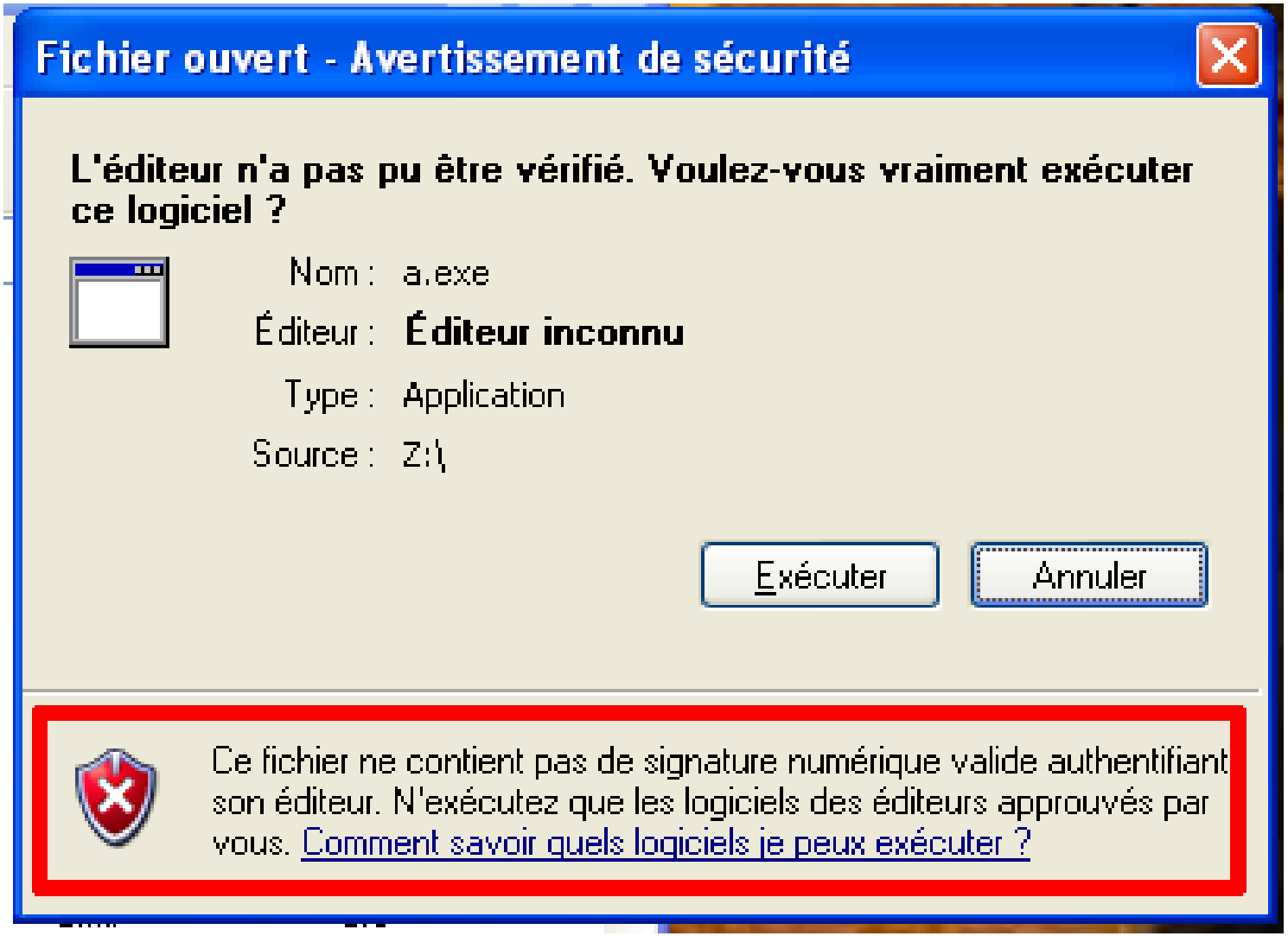
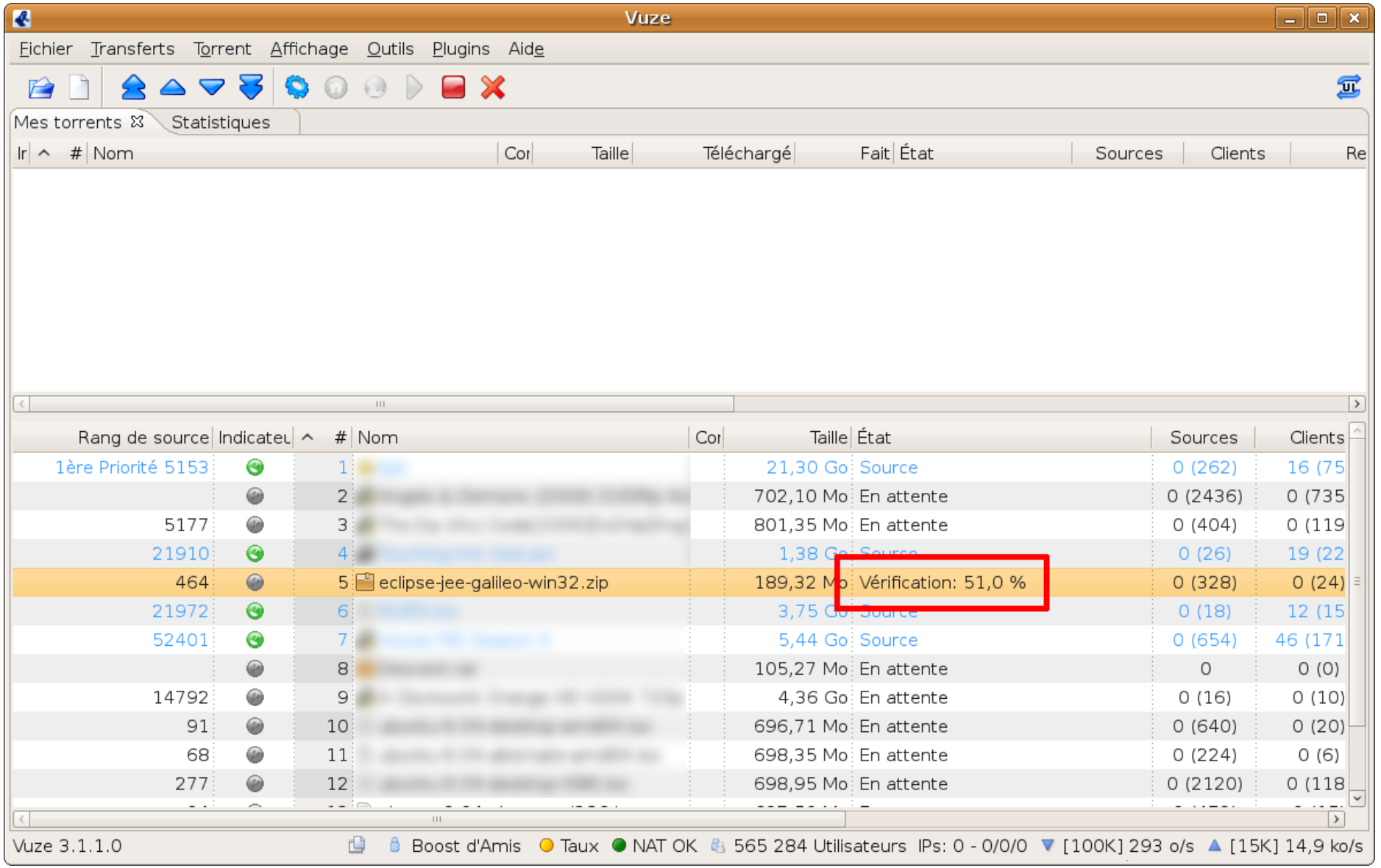
Hash functions

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

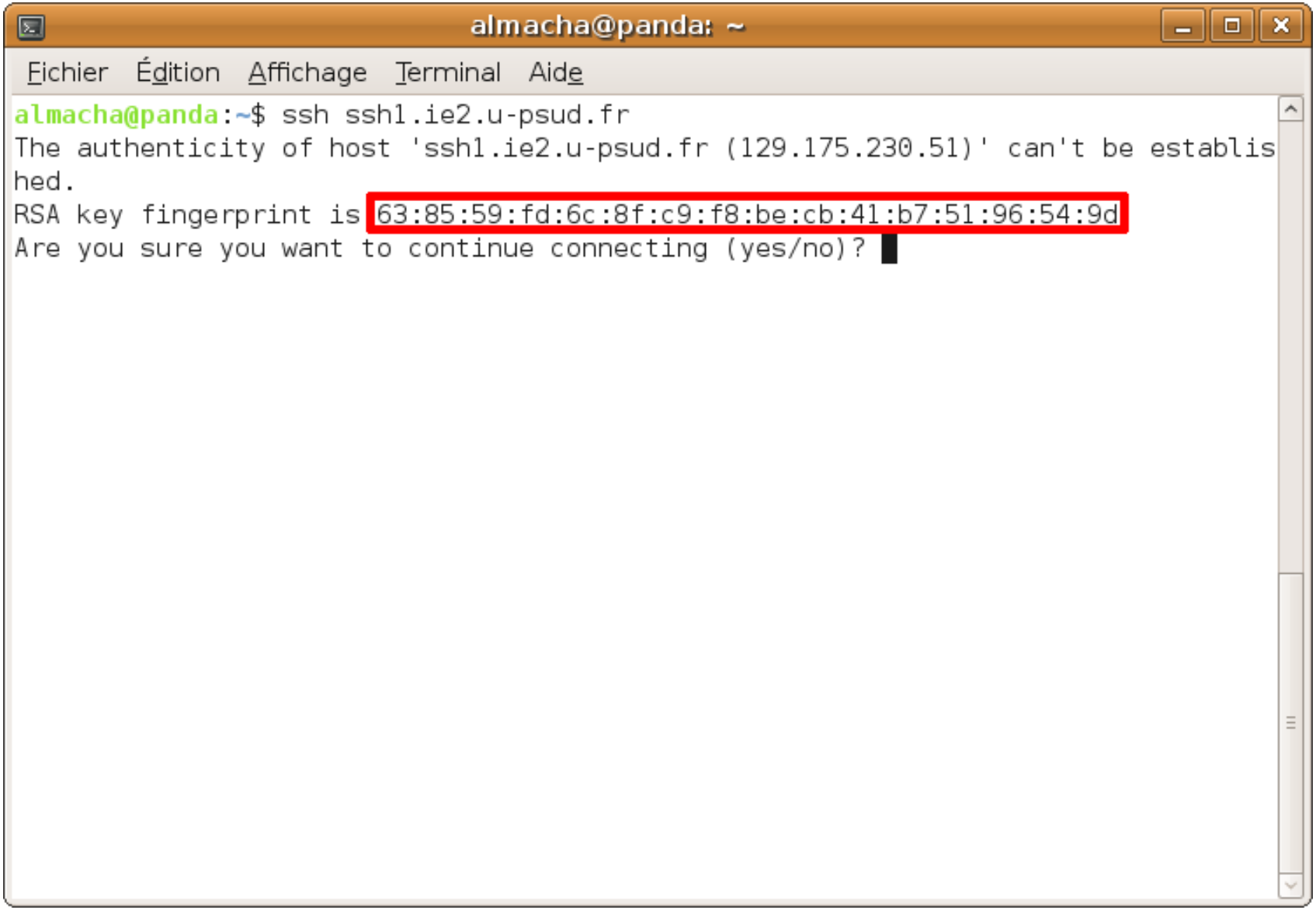
Hash functions are everywhere



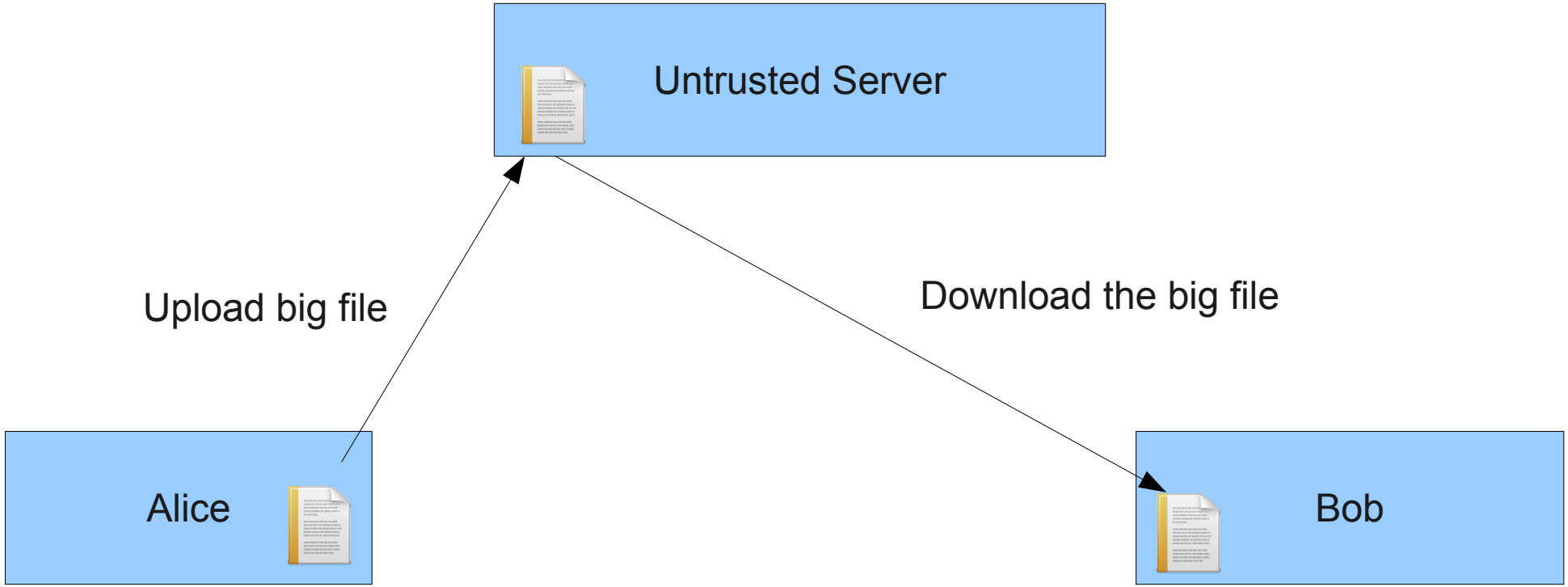
Hash functions are everywhere



Hash functions are everywhere



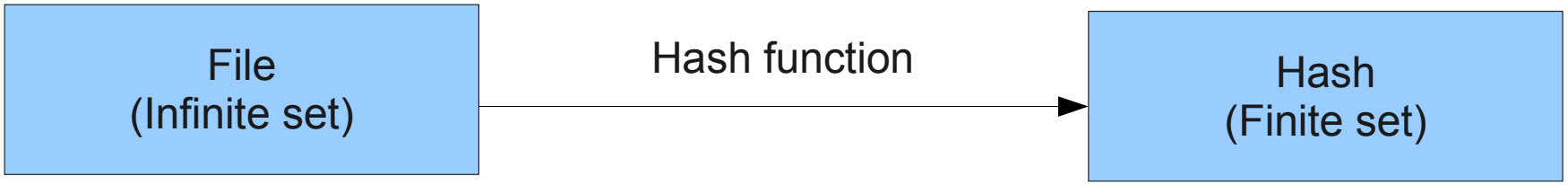
Without hash functions



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

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

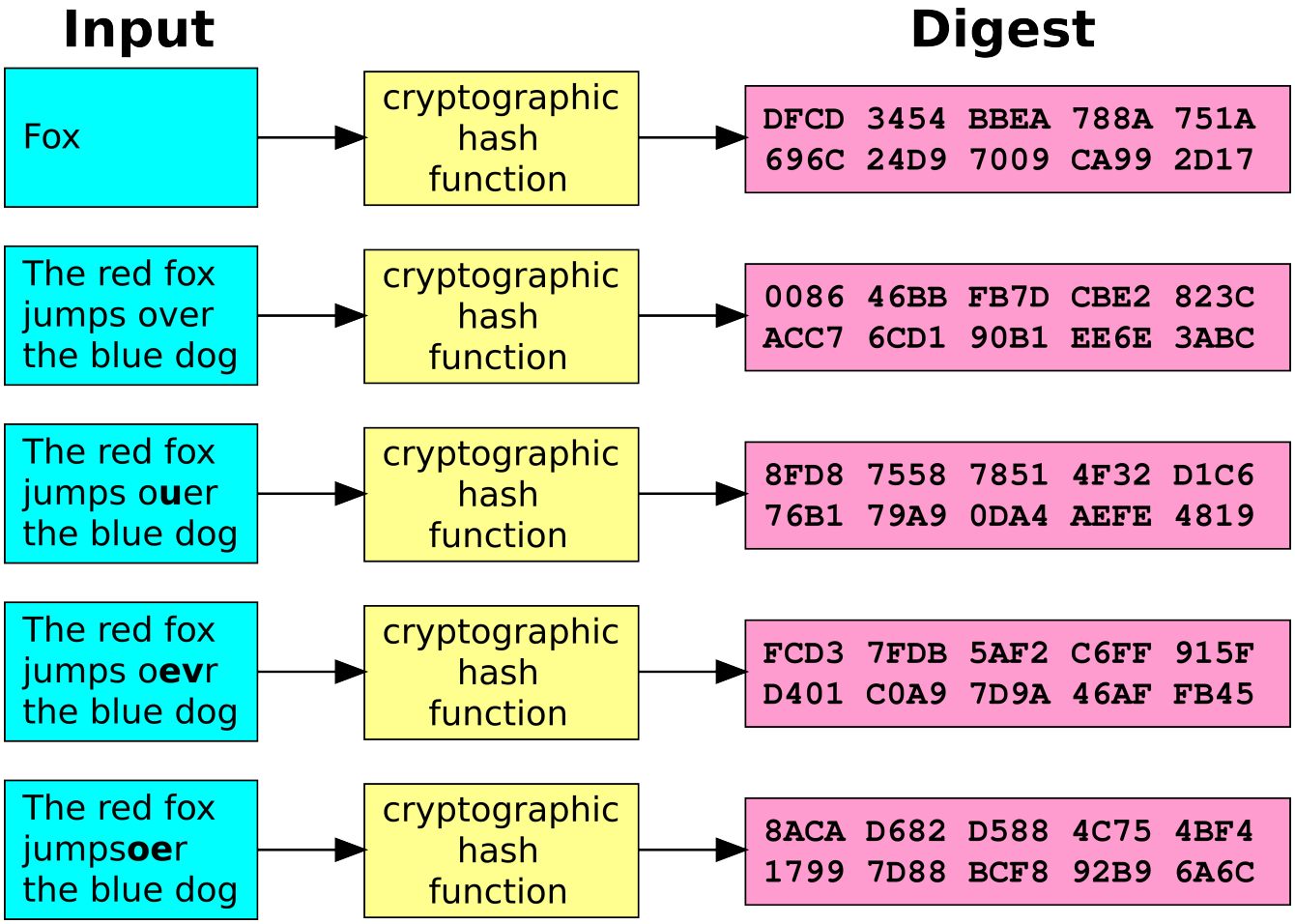
What is a hash function?



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

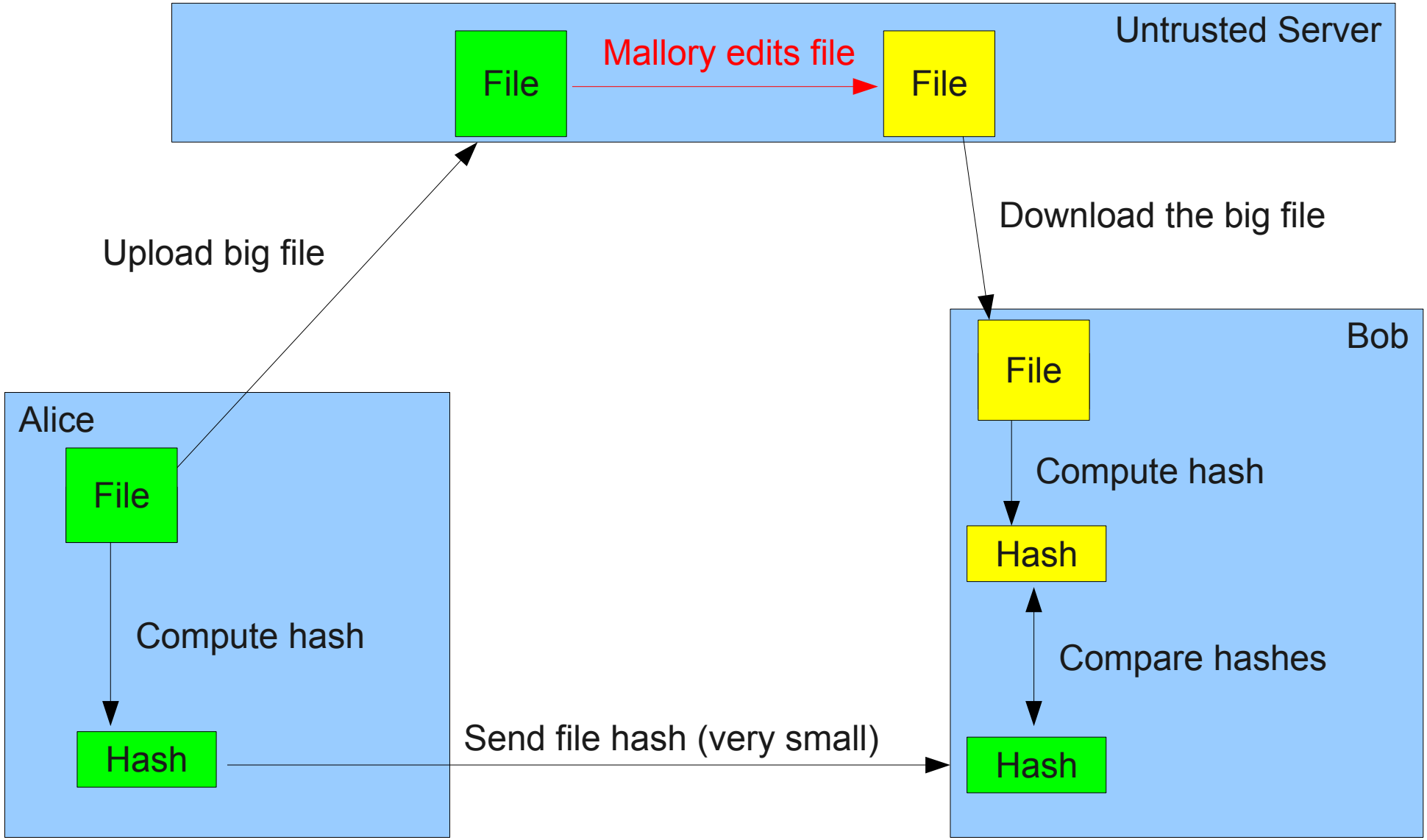
Example: The SHA-256 hash function takes a file and returns a 256 bits string.

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.

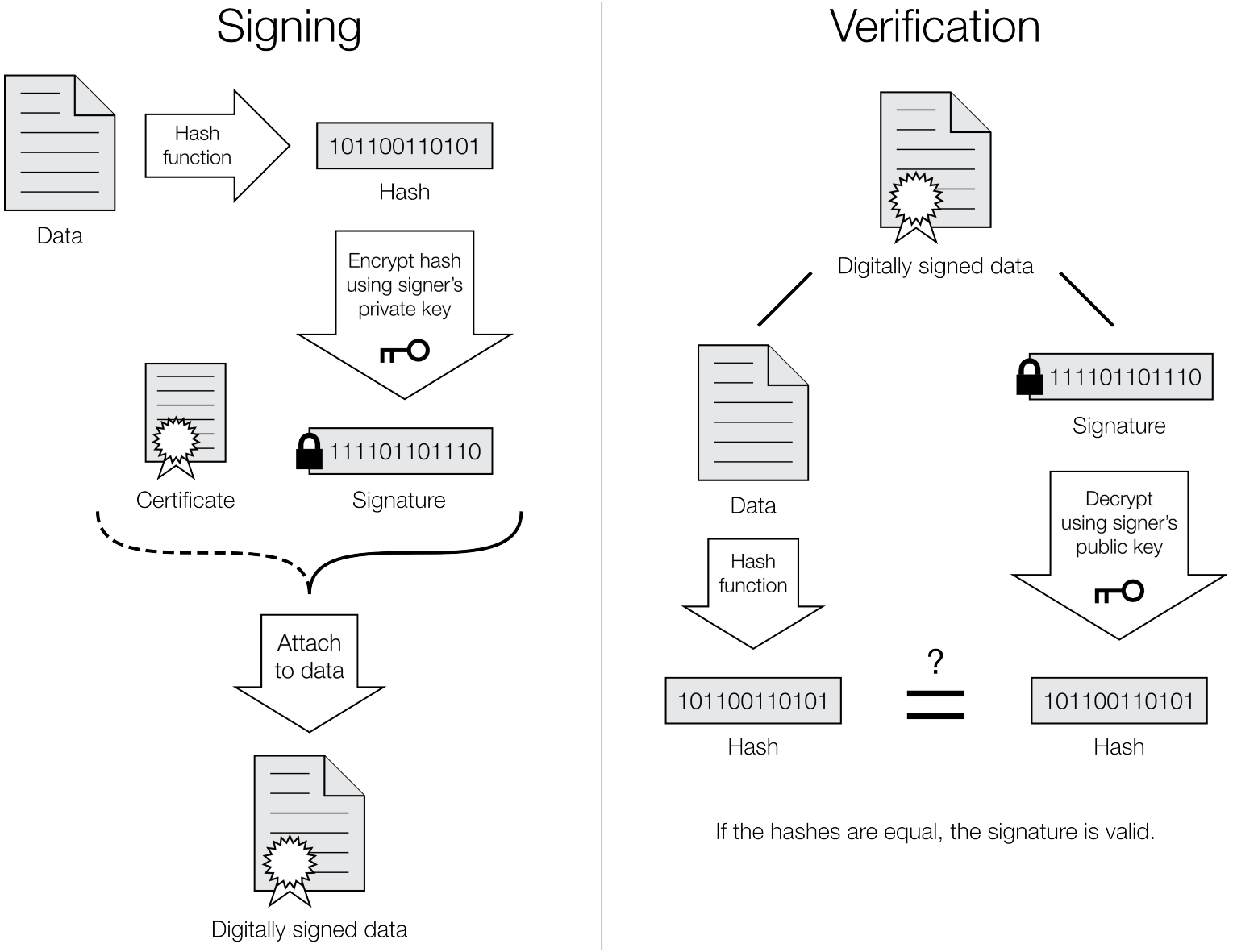
Consider the following hash function:

```
Function f(A[1..n]) =
    Let S = 0
    For i from 1 to n :
        S ← (S + A[i]) mod 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(someone has his birthday the same day as you) = 14%

P(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 =

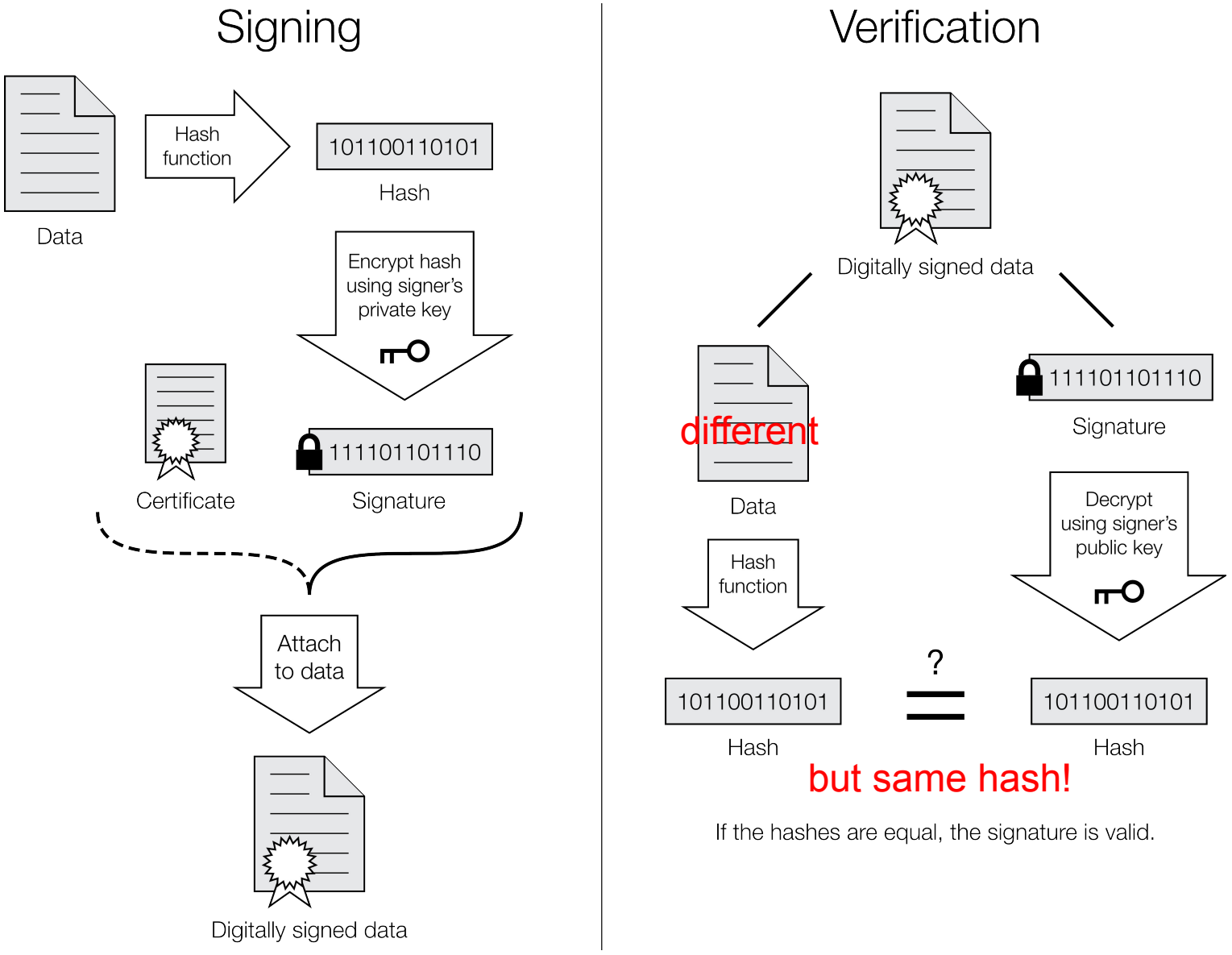
$$0.5 \times n$$

Time to find 2 files with the same hash =

$$1.25 \times \sqrt{(n)}$$

=> The hash has to be twice as long.

Birthday attack

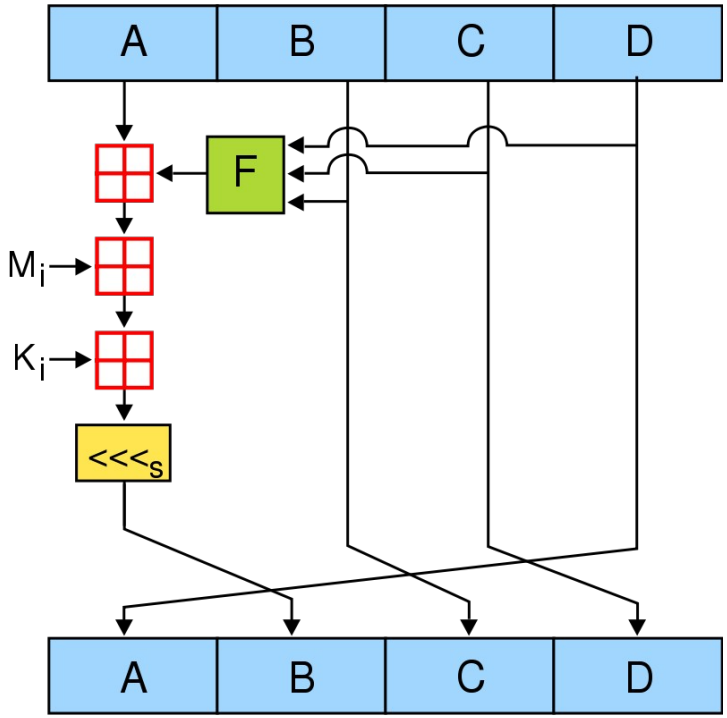


What we mean by broken

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.

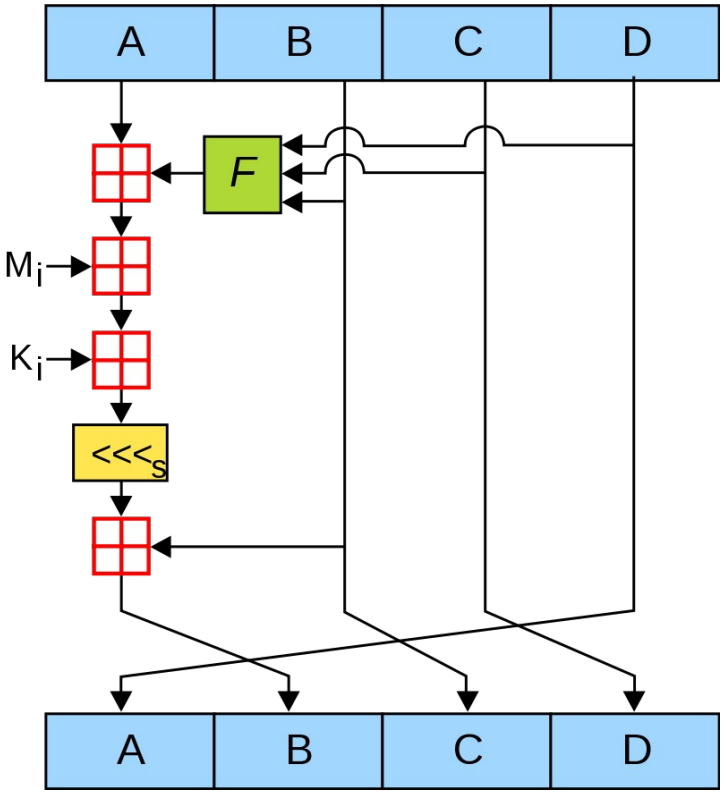
MD4

- 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.



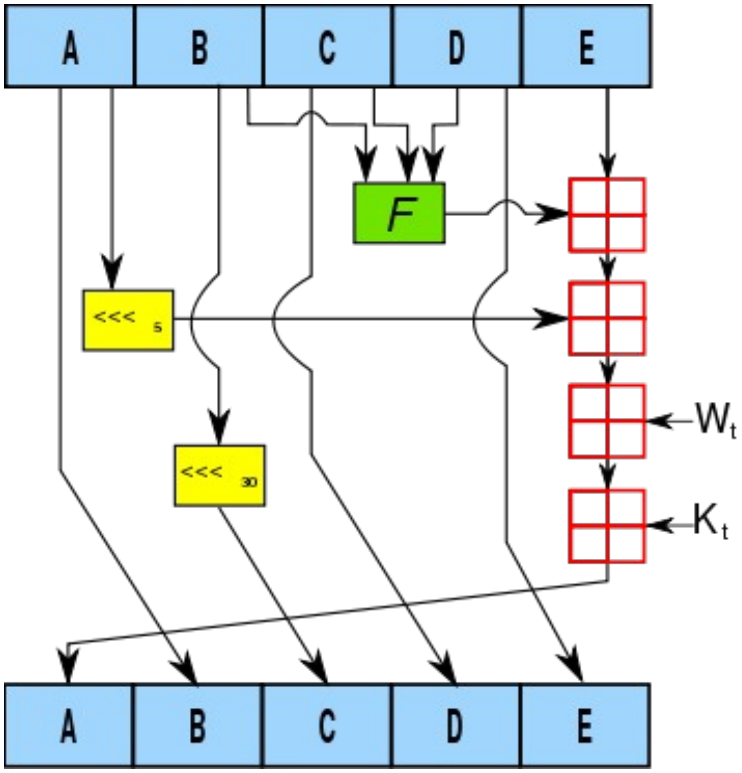
MD5

- 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.



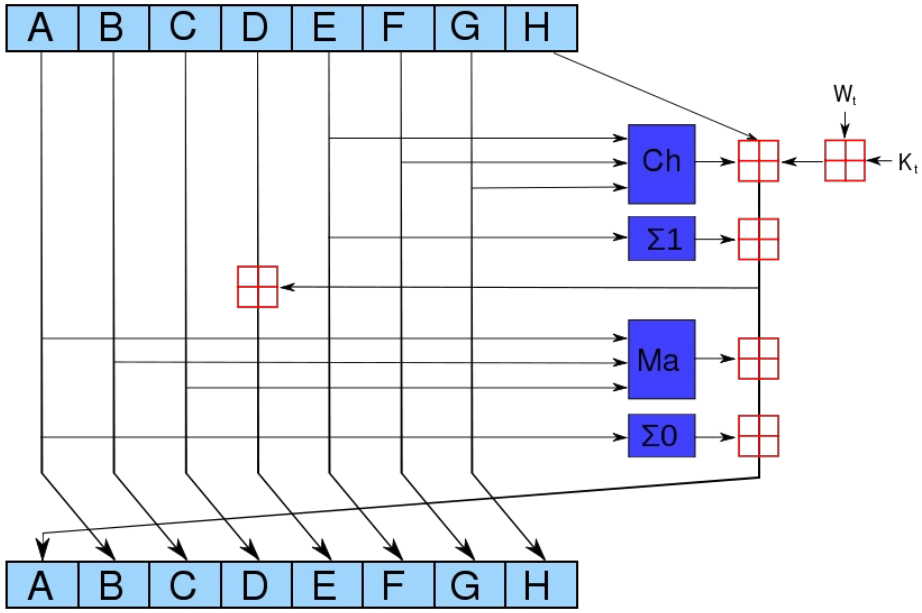
SHA-1

- 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!