checksum/hash

Matsuzaki ‘maz’ Yoshinobu

<maz@iij.ad.jp>

Checksum

- data integrity refers to maintaining and assuring the accuracy and consistency of data over the research lifecycle
- checksums provide a way to monitor the integrity of your data
- it’s also called as ‘Hash’ or ‘Fingerprint’

Data integrity

- what is a checksum/hash?
  - like a fingerprint of a file
  - produces a condensed representation of a message (hashing)
  - used to verify whether two copies of a file are identical
- the fixed-length output is called the checksum, hash or message digest
- each time you run a checksum, a string of digits is created for each file. If just 1 byte of data has been altered, the same process will generate a different string.

Hash Functions

- A hash function takes an input message of arbitrary length and outputs fixed-length code.
  - Given x, we can compute the value f(x).
  - Given f(x), it is hard to get the value of x.
- A form of signature that uniquely represents the data
  - Collision-free
- Uses:
  - Verifying file integrity - if the hash changes, it means the data is either compromised or altered in transit.
  - Digitally signing documents
  - Hashing passwords

Hash Functions

- Message Digest (MD) Algorithm
  - Outputs a 128-bit fingerprint of an arbitrary-length input
  - MD5 is getting obsolete
- Secure Hash Algorithm (SHA)
  - SHA-1 is obsolete
  - Widely-used on security applications (TLS, SSL, PGP, SSH, S/MIME, IPsec)
  - SHA-2 families - SHA-256, SHA-384, SHA-512 are commonly used, which can produce hash values that are 256, 384, and 512-bits respectively
- RIPEMD
  - Derived from MD4, but performs like SHA
  - RIPEMD-160 is the most popular version

Digital Signature

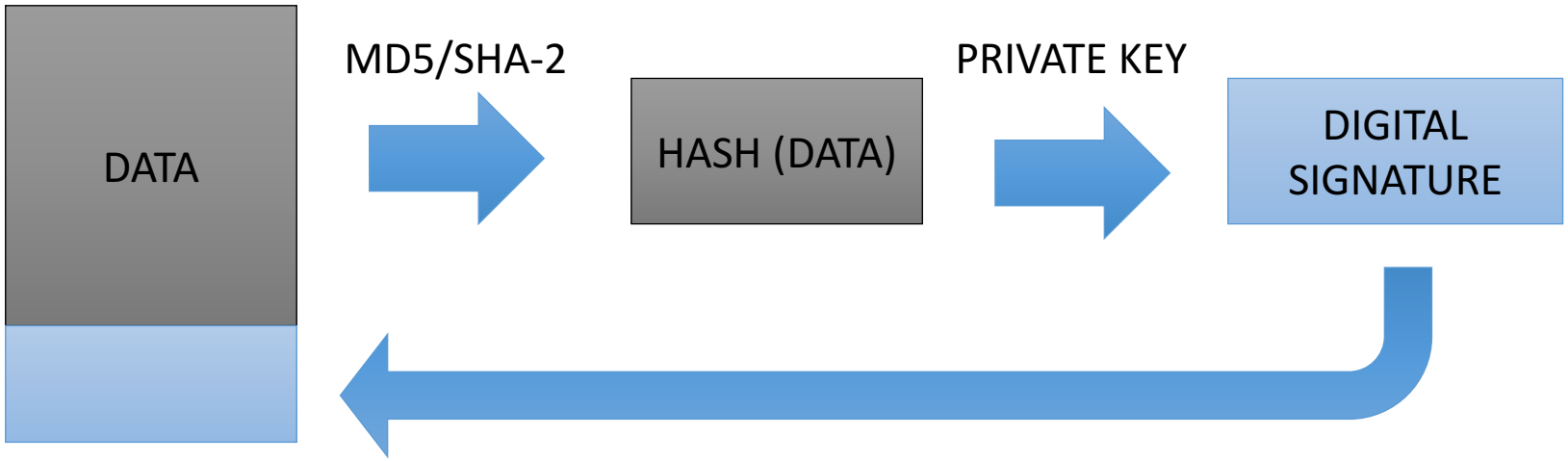
- A digital signature is a message appended to a packet
- The sender encrypts message with own private key instead of encrypting with intended receiver's public key
- The receiver of the packet uses the sender's public key to verify the signature.
- Used to prove the identity of the sender and the integrity of the packet

Digital Signature

- Two common public-key digital signature techniques:
  - RSA (Rivest, Shamir, Adelman)
  - DSS (Digital Signature Standard)
- Used in a lot of things:
  - Email, software distribution, electronic funds transfer, etc
- A common way to implement is to use a hashing algorithm to get the message digest of the data, then use an algorithm to sign the message

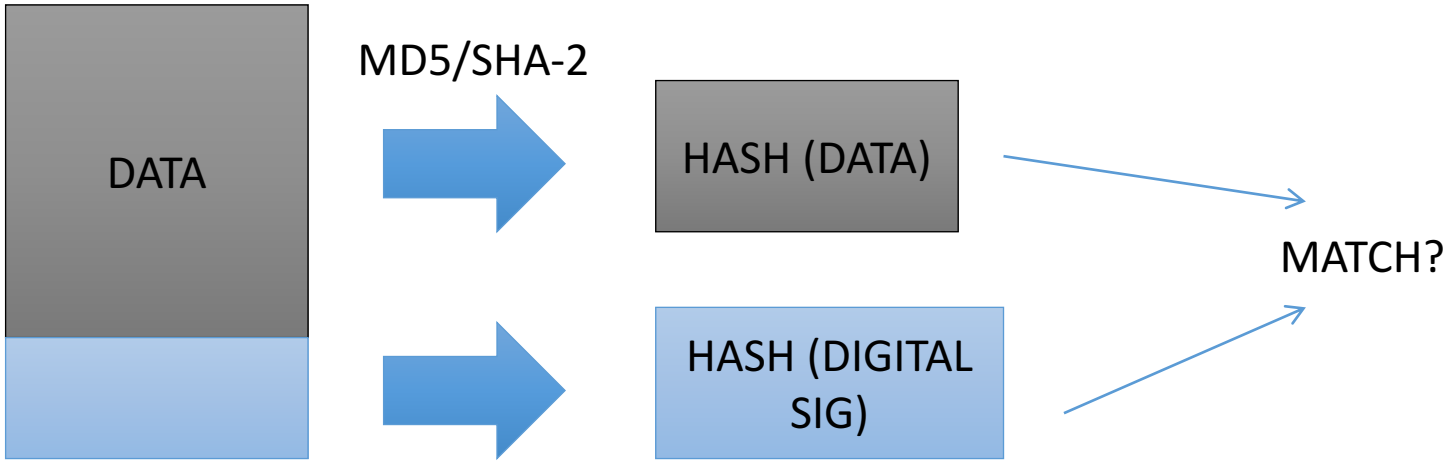
Digital Signature Process

1. Hash the data using one of the supported hashing algorithms (MD5, SHA-256)
2. Encrypt the hashed data using the sender's private key
3. Append the signature (and a copy of the sender's public key) to the end of the data that was signed)



Signature Verification Process

1. Hash the original data using the same hashing algorithm
2. Decrypt the digital signature using the sender's public key. All digital signatures contain a copy of the signer's public key
3. Compare the results of the hashing and the decryption. If the values match then the signature is verified. If the values do not match, then the data or signature was probably modified.



Hash collision

- hash(m1) and hash(m2)
  - hash is a checksum operation
  - m1/m2 are different messages
- hash(original_file) = hash(fake_file)
  - If there is a way to create a file that has the same checksum values as the target file, that's…problem

Think about a weak checksum

- Calculate the sum of each byte, and get the last digit in base 10 as a checksum
  - hash(10 21 30 31) is ‘2’
  - hash(11 22 33 09) is ‘5’
- It’s easy to produce collisions
  - Can not rely on this algorithm to check data integrity

- Use a modern checksum algorithms to check data integrity
  - like SHA256, SHA512

sha utilities

- on UNIX
  - SHA256: $ sha256sum <file>
  - SHA512: $ sha512sum <file>
- on Mac
  - SHA256: $ shasum -a 256 <file>
  - SHA512: $ shasum -a 512 <file>
- on Windows
  - SHA256: > certutil -hashfile <file> SHA256
  - SHA512: > certutil -hashfile <file> SHA512

Need GUI? (windows)

- Create a new bat file named ‘sha256.bat’ which contains the following 3 lines

```
@echo off
certutil -hashfile  “%~1” SHA256
pause
```

- Open explorer, type sendto on the address bar
- Put the bat file there
- How to use it
  - Select a file, right click, select sendto then sha256, you will get a SHA256 checksum of the file

md5 utilities (old)

- on UNIX
  - $ md5sum <file>
- on Mac
  - $ md5 <file>
- on Windows (7 or later)
  - > certutil -hashfile <file> MD5

sha1 utilities (old)

- on UNIX
  - $ sha1sum <file>
- on Mac
  - $ shasum <file>
- on Windows
  - > certutil -hashfile <file> SHA1

hands on

Exercise 1: text file

- Create a new text file with the following contents

```
hello!<enter>
```

- Get checksums of the file
  - sha256, sha512

Result should be:

- Mac&Linux:
  - sha256: 69b63d7c7f5e81dcc55ca4be4f45add1757f3b8992c8ac34dd3e80e2f8d814e7
  - sha512: ac3c78e58cda47f8df0bbe4bb1002689d82d8379b1d0636df1cf3dbcca63b73d59835d4c485c055ff94a9022f4cc0d3bea3967b4fc4edd00fa6121bce541455a
- Windows
  - sha256:
  - 1c0b8448b72c55f2f614640252d14ec132fa031d7c5c35d93b6bce780cfe0e92
  - sha512:
  - d91190cce04a17036bb57086c69bd8c377193880d4404d81c940624d6a3251d4623d2c53fc45934de9bef85f073b0acca8bd796fe4bec7a6c6eecfdee2334888

If you are interested in

- made on Windows
  - hello-win.dat
- made on Mac
  - hello-unix.dat

Why? It's "newline" problem

- on Mac&Linux
  - 68 65 6c 6c 6f 21 0a 0a |hello!..| -- <LF><LF>
- On Windows:
  - 68 65 6c 6c 6f 21 0d 0a |hello!..| -- <CR><LF>

- It looks the same, but actually different
- That's why ftp client has 'text' mode to translate newlines of text document accordingly
  - binary mode just transfer the data as is

Exercise 2: rename the file

- Change the filename to something else
- Get checksums of the files
  - sha256, sha512
- Compare the checksums with the result you get on the Exercise 1

Exercise 3: modify the file

- Delete ’o’ from the file, now the contents should be

```
hell!<enter>
```

- Get checksums of the files
  - sha256, sha512
- Compare the checksums with the result you get on the Exercise 1

Exercise 4: re-modify the file

- Add ’o’ to the file, now the contents should be

```
hello!<enter>
```

- Get checksums of the files
  - sha256, sha512
- Compare the checksums with the result you get on the Exercise 1

Exercise 5: downloaded file

- Get checksums of this pdf file
  - sha256, sha512
- Compare the checksums with your neighbors