Cryptographic Hash Function

Cryptographic Hash Function

- Also called
  - One-way Hash function
  - Message Digest
  - One-way Hash Digest
  - Hash Digest
- Strictly speaking the term digest refers to the output of the function.
- So the term ‘cryptographic’ is sometimes understood (assumed) to be present and not explicitly used.

Cryptographic Hash Function

- “A Hash function is a function that can be used to map data of arbitrary size to data of a fixed size.”
- “The values returned by a hash function are called hash values, hash codes, digests, or simply hashes.”
- A cryptographic hash function is a hash function with a particular set of properties.

Cryptographic Hash Function

- A cryptographic hash function is a mapping from data to a number of fixed length (say 128, 256,.. bits).
- The value associated with data is unique, i.e. changing just one character in the data changes the associated hash value dramatically (avalanche effect).
- The data can not be retrieved from the hash value (hence one-way).

Four Properties of a Cryptographic Hash Function

- It is easy to compute the hash value for any given message,
- It is infeasible to find a message that has a given hash,
- It is infeasible to modify a message without hash being changed,
- It is infeasible to find two different messages with the same hash.

Cryptographic Hash Function

- Normally a bunch of steps that mangle the input in a particular way.
- 'Swap every bit with the complement of the bit 11 bits to the right'
- 'Multiply every 12 bits by the number 384729'
- The process should <u>not</u> be reversible.

Hash Sizes

Hash sizes

- Should hash values be for example 32 or 64 or 128, 256, 512 bits?
- We want to determine the minimum hash sizes required so for example
  - it is not feasible to find a string that hashes to a particular hash value.
  - it is not feasible to find two strings that hash to the same value (collision).

Throw a Dice

- Find the avverage number of throws before throwing a 6.
  - $P(6) = 1/6$
  - $P(x,6) = 5/6 \cdot 1/6$
  - $P(x, x, 6) = 5/6 \cdot 5/6 \cdot 1/6$

- $P(k) = (1-p)^{k-1}\, p$
- [Prob that k throws required to throw a 6]

Throw a Dice

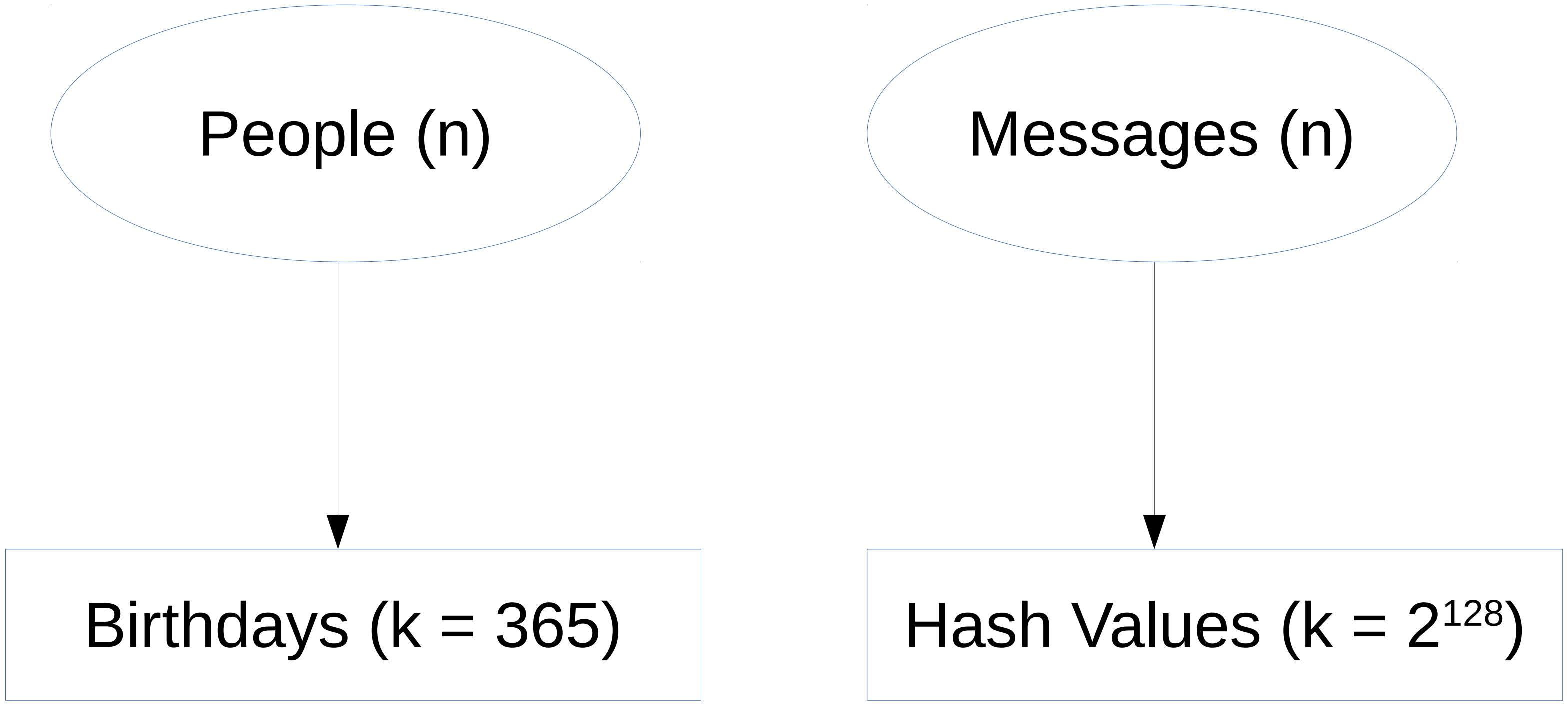
- Expected value of the number of throws required to throw a 6.
- $E(k) = \Sigma\, k \,.\, P(k)$

  $= 1.\, P(1) + 2.P(2) + 3.\, P(3) + \ldots$

Result

$E(k) = k$ (i.e. 6)

Birthdays & Cryptographic 64 bit Hash Values



Birthdays - Invert

- n – number people in a room (corresponds to messages)
- k - number outcomes
  - birthdays, k = 365
  - (corresponds to hash values)
- The expected number of people required so that we have someone with a particular birthday is also equal to k (365).

Hash Values - Invert

- Hash size – 128 bits
- Number hash values – 2^{128}
- The expected number of messages that we need to generate to find a message with a particular hash value is 2^{128}
- This is not feasible.

Birthdays – Clash

- For a room of n people there are n(n-1)/2 pairs of people [pairs ~ O(n2)].
- For 20 people there are 400 pairs.
- Each pair has a 1/365 chance of having the same birthday.
- Need only about 20 people to have a better than 50% chance of birthday clash
- Need only sqrt(365) for a 50% chance of a clash.

Hash Values - Clash

- Hash size – 128 bits
- Number hash values – 2^{128}
- The expected number of messages that we need to generate to find two message with the same hash value is $\mathrm{sqrt}(2^{128}) = 2^{64}$
- This is feasible.
- => 128 bit hash is not safe.
- (Messages with the same hash values have been generated)

Hash Sizes

- So hash algorithm must be greater than 128 bits, to be safe.
- MD5 is 128 bit hash but was broken (shown not to be collision resistant) in 2004 and many times since.
- No longer considered safe.
- SHA algorithms now preferred.
- Conclusion – hash sizes should be greater than 128.

Uses of Hash/Message Digests

Uses of Hash/Message Digests

- File CheckSums
- Downline Load Security
- Message Fingerprints
- Password Hashing
- Message Integrity (HMAC)
- Digital Signature Efficiency

File CheckSums

- Often software to be downloaded have a checksum quoted.
- Can be used to verify if you have the exact/correct copy of the software.

Downline Load Security

- Some devices connected to a network (routers, printers etc.) might not have enough persistent store to store the software they run.
- They often download the programs they run.
- And use Hashes to ensure that they have the right version of the program.

Message Fingerprints

- It you wanted to keep a master copy of a large piece of data/program so that you could always verify that you had the correct working copy.
- You could just store the Hash of the data.

Password Hashing

- Passwords should not be stored in cleartext.
- They are hashed and the hash value is stored.
- When authenticating a user, the password supplied by the user is hashed and then compared with the stored has value.
- So even if the password database is compromised, this is of no use to the attacker.

Digital Signature Efficiency

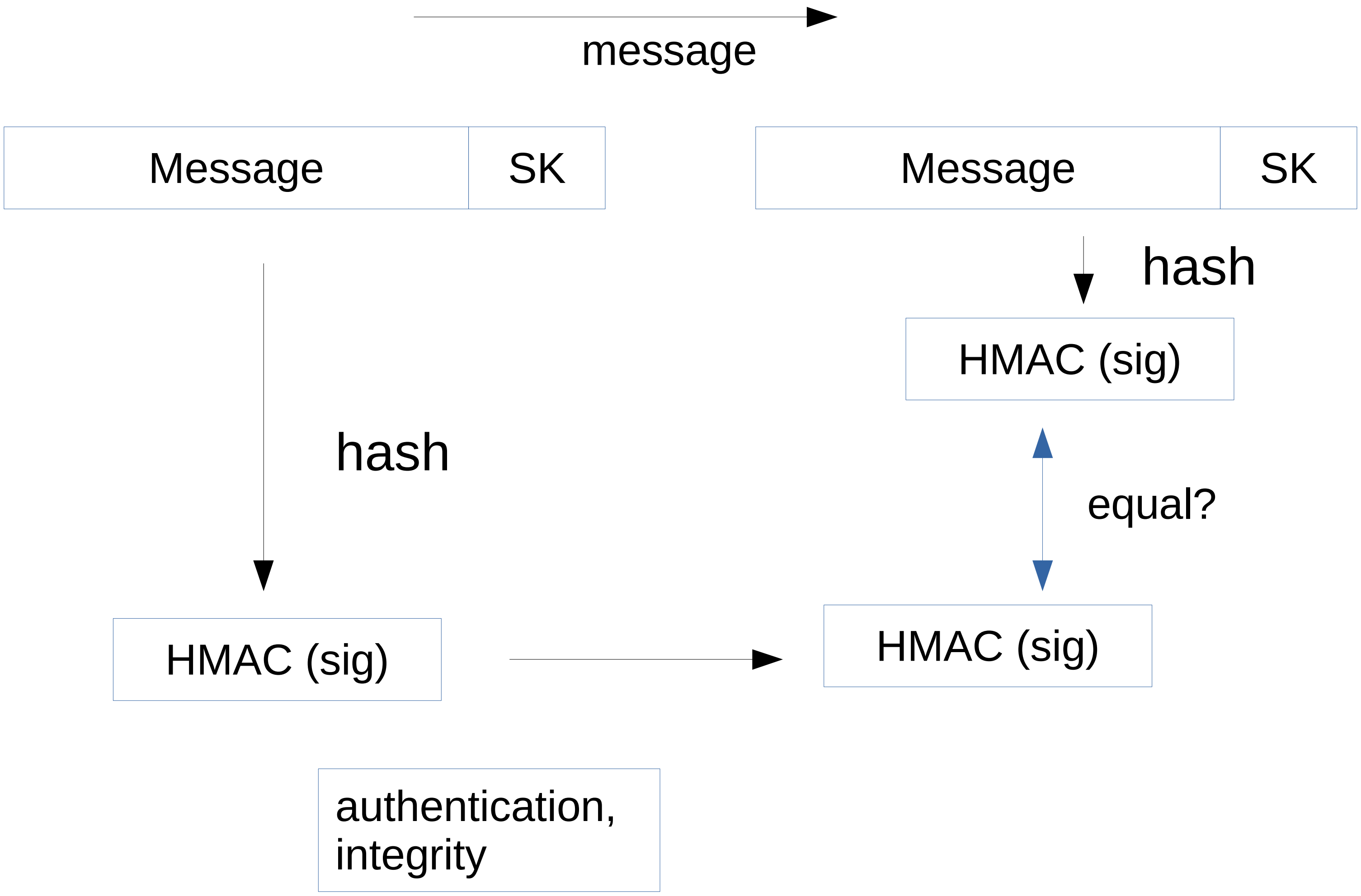
- (Later). Hashes are used in digital signatures.
- You sign a hash of a message rather than the message itself.
- Public/private key algorithms are computationally expensive.

Message Authentication & Integrity
Hashed Message Authentication Codes (HMAC)

Message Authentication & Integrity (HMAC)

- Hash Message Authentication Codes (HMAC)
- A technique for verifying the <u>integrity</u> and <u>authenticity</u> of a message.
- Used with <u>a shared secret key</u>.
- Take a hash of the message + secret key.
- Receiver does the same, and checks that the hashes match.

HMAC


message
Message
SK
Message
SK
hash
HMAC (sig)
hash
equal?
HMAC (sig)
HMAC (sig)
authentication,
integrity

Message Integrity (HMAC)

- If they do we know
  - The message has not been changed
  - It originated from the peer with which we share the secret key.
- HMACs are much more efficient than digital signatures (later).
- But require a shared secret.

Example Algorithms

Example Algorithms

- MD5
- SHA-1
- SHA-2

MD5

- Message-Digest algorithm 5 - 1991
- Generates a 128 bit value
- Commonly used to check the integrity of files.
- It has been shown that it is not collision resistant.
- Now vulnerable in this and other regards.

SHA-2

- Set of four with hash digests of size
  - 224, 256, 384 or 512 bits.
- Used in TLS and SSL, PGP, SSH, S/MIME, and Ipsec.

SHA-3

- Competition to find the best Hash Function.
- Started in 2008.
- Final candidates announced in December 2010.
- Result published in 2015.
- Not meant as a replacement for SHA-2 as no attack on SHA-2 has been demonstrated.

Cryptography / Java

Base64 Encoding

- A way of encoding binary data as text.
- Binary data is split into 6 bit parts with padding if necessary.
- (Padding is necessary if the number of bytes is not divisible by 3.)
- Each of these 6 bit values is represented by one of 64 characters.
- [$2^6 = 64$]

Base64 Encoding

Value	Char
0	A
1	B
25	Z
26	a
51	z
52	0
61	9
62	+
63	/

Base64 Encoding

- Binary values are padded with zeros.
- Zeros at the end ot the Base64 string are encoded as “=”.

Base64 Encoding and Decoding

```
String s = "qwerty" ;
byte[] sBytes = s.getBytes() ;
String encodedString = Base64.getEncoder().encodeToString(sBytes);
System.out.println("s is: " + s + "  Encoded: " + encodedString);

byte[] decodedBytes = Base64.getDecoder().decode(encodedString);
System.out.println("Encoded: " + encodedString +
                                        "  Decoded: " + new String(decodedBytes));



Outout:
s is: qwerty  Encoded: cXdlcnR5
Encoded: cXdlcnR5  Decoded: qwerty
```

Base64 Encoding and Decoding

Outout:
s is: qwerty Encoded: cXdlcnR5
Encoded: cXdlcnR5 Decoded: qwerty

Another example
s is: qwertyu Encoded: cXdlcnR5dQ==
Encoded: cXdlcnR5dQ== Decoded: qwertyu

Message Digests

Example – MD5

```
public class A1MessageDigestEx {
  public static void main(String[] args) {

    String password = "12345";
    MessageDigest algorithm = null;
    try {
      algorithm = MessageDigest.getInstance("MD5");
    } catch (NoSuchAlgorithmException e) {
      e.printStackTrace();
    }

    algorithm.reset();
    algorithm.update(password.getBytes());
    byte[] messageDigest = algorithm.digest();
```

Example – MD5 (cont)

```
    System.out.println("length " + messageDigest.length);

    String encodedDigest = Base64.getEncoder().encodeToString(messageDigest);;
    System.out.println("Base64 encoded message digest " + encodedDigest);

  }
}
```

java.security.MessageDigest

- update() - adds data to be hashed
- reset() - clears the data (not necessary in this case)
- digest() – calculates the hash digest

Apache Commons Codec Library

- commons-codec-1.6.jar
- http://commons.apache.org/codec/apidocs/index.html

- Has some convenience methods for getting digests.

Example – MD5 & SHA256

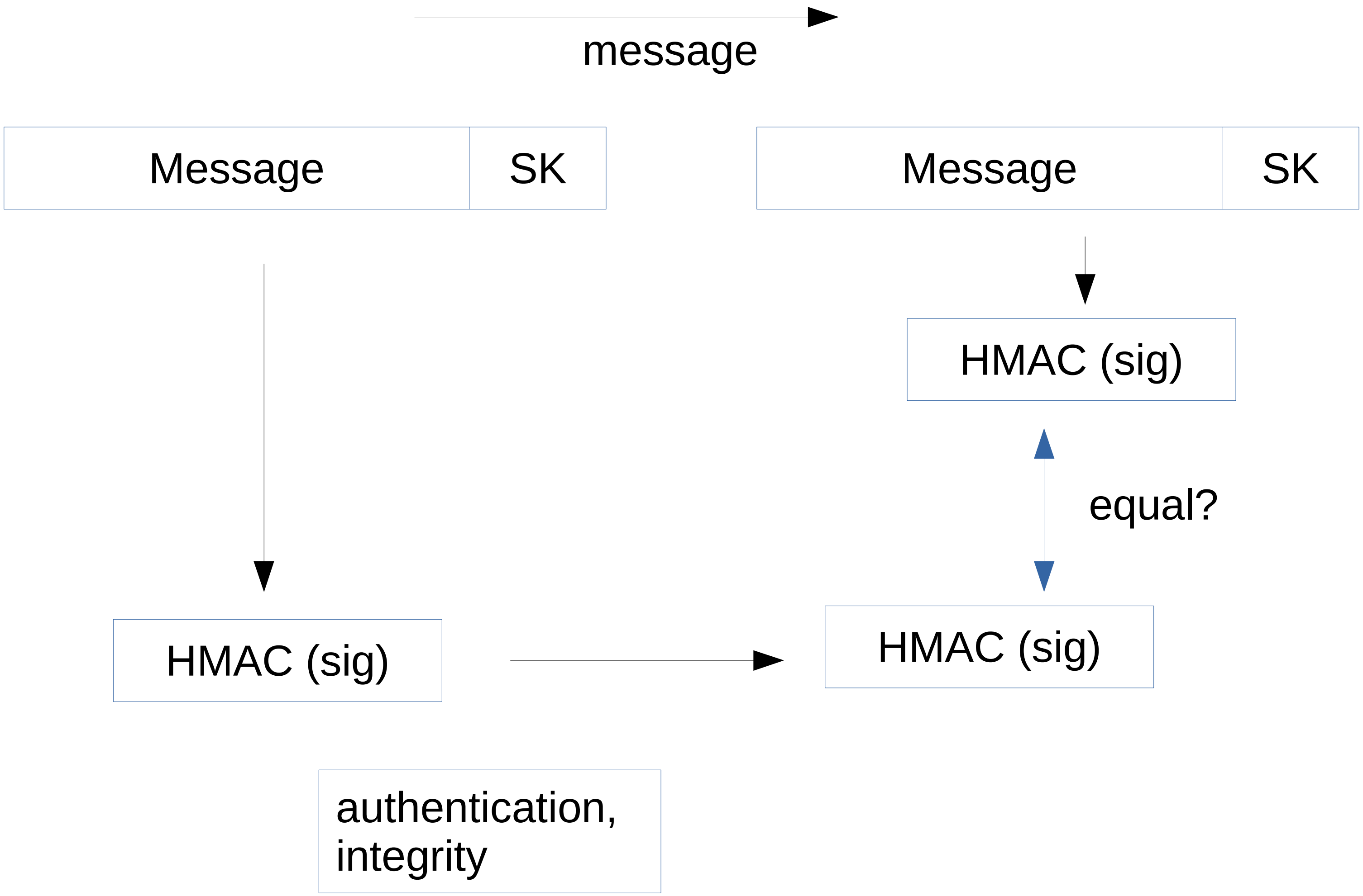
```
import org.apache.commons.codec.digest.DigestUtils;

public class E2MessageDigestEx {

      public static void main(String[] args) {

       String sessionid = "12345";
       String md5 = DigestUtils.md5Hex(sessionid);
       System.out.println("sessionid " + sessionid  +
                             " md5 version is " + md5);
       String sha256 = DigestUtils.sha256Hex(sessionid);
       System.out.println("sessionid " + sessionid +
                             " sha256 version is " + sha256);
      }
}
```

Hashed Message Authentication Code (HMAC)

HMAC

- Take the hash of a message + secret key.
- Receiver does the same, and checks that the hashes match.
- Authenticates the sender (only they have the secret key)
- Verifies the integrity of the message.

HMAC



message
Message
SK
Message
SK
HMAC (sig)
equal?
HMAC (sig)
HMAC (sig)
authentication, integrity

HMAC Example

```
KeyGenerator kg = KeyGenerator.getInstance("HmacSHA256");
SecretKey sk = kg.generateKey();

Mac mac = Mac.getInstance("HmacSHA256");
mac.init(sk);
byte[] result = mac.doFinal("Hi There".getBytes());
System.out.println(result.length);

/// Receiver
Mac mac2 = Mac.getInstance("HmacSHA256");
mac2.init(sk);
byte[] result2 = mac.doFinal("Hi There".getBytes());

System.out.println("Check: " +
          Arrays.equals(result, result2));
```

Base64 Encoded HMAC

```
byte[] hmac = mac.doFinal(textArray);
String encodedHmac =
        Base64.getEncoder().encodeToString(hmac);
System.out.println("Encoded HMAC :" + encodedHmac);


// Base64 decode a HMAC
byte[] decodedHmac =
        Base64.getDecoder().decode(encodedHmac);
```

Base64 Encoded Secret Key

```
// Base64 encode a secret key
String encodedKey =
Base64.getEncoder().encodeToString(sk.getEncoded());
System.out.println("Encoded Key :" + encodedKey);


// Base64 decode a secret key
byte[] decodedKey =
Base64.getDecoder().decode(encodedKey);
SecretKey sk = new SecretKeySpec(decodedKey, 0,
decodedKey.length,"HmacSHA256");
```

Summary

- What is a Hash Digest (or one way hash)
- Properties of a Hash Digest
- What size should a hash digest be.
- Uses of a Hash Digest
- What is a Hashed Message Authentication Code?
  - Provides authentication and integrity of a message without sending the shared secret (password)

Summary (Java)

- Calculate MD5 and SHA hashes (binary values).
- Get the Base64 encoded version of the hash value (text value)
- Calculate the HMAC value for a message.
- Print out the Base64 encoded version of the HMAC.