9.2 Computer and Network Security Integrity & Public Key Cryptography

INFO1112

Bob Kummerfeld

For network connections we wish to provide:

- **Confidentiality**
  - Protecting information from interception
  - Ensure information remains confidential if intercepted
- **Integrity**
  - Detecting whether information has been tampered with
- **Authenticity**
  - Ensuring we know who the sender is
  - Non-repudiation

Message Integrity

To check if a message has changed you can calculate a type of checksum called a "hash" or "message digest". This is a short value (eg 128 bits) calculated from a long message.

Different messages should have different digests

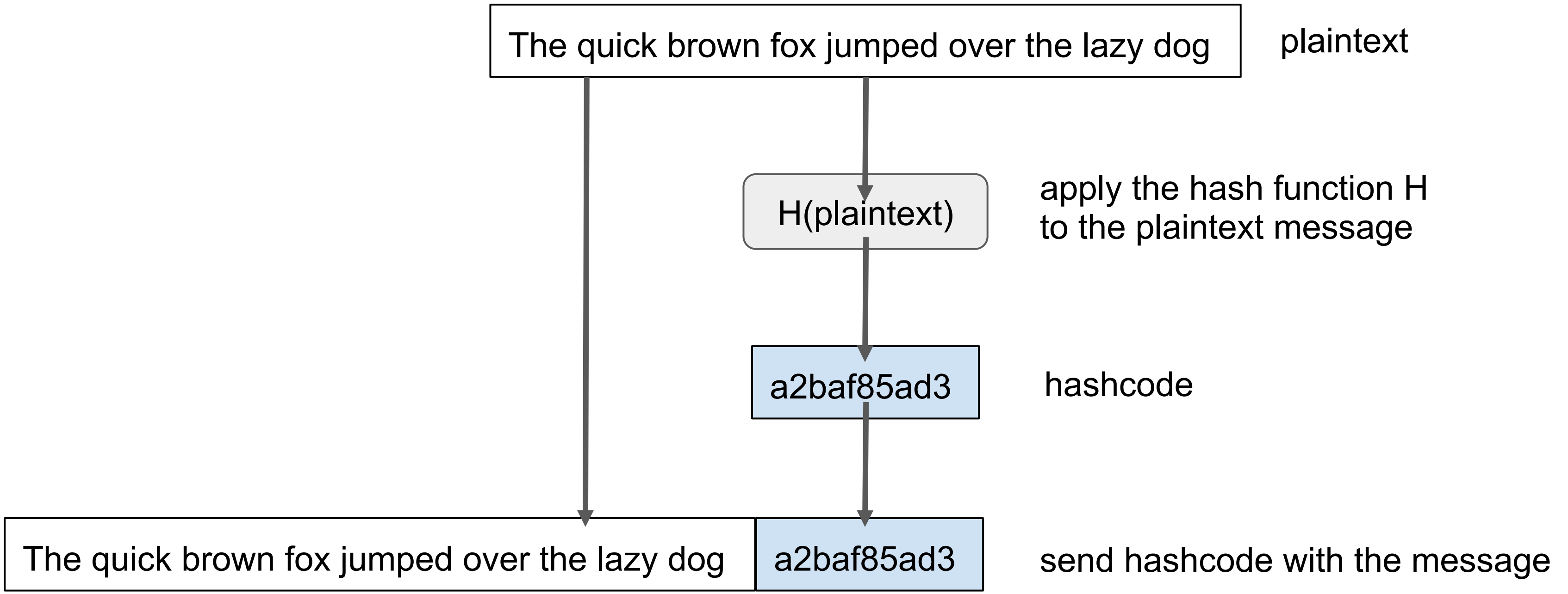
- If the hash function is H, it is hard to find a message N for which H(N) = H(m)
- A good H(m) should have a “one-way function property”
  - Inverse function is very hard to compute

Example algorithms include the USA standard, SHA-256

Message digests are used to check the integrity of a message.

Python provides a library module, `hashlib`, for this.

```
$ python
Python 2.7.16 (default, Oct 10 2019, 22:02:15)
[GCC 8.3.0] on linux2
Type "help", "copyright", "credits" or "license" for more
information.
>>> import hashlib
>>> h = hashlib.sha256("green eggs")
>>> h.hexdigest()
'6a3d501466f63d04d8ecd9ff3efe376e7784d0a3bbb21a9ce2fe4be12f77fbd2'
>>> h.update(" and ham")
>>> h.hexdigest()
'a113a9854ab71a4914e219e181ca8bfd48d7d65bdb1c3cb1bad6235c5f1acf23'
>>>
```


The quick brown fox jumped over the lazy dog
plaintext
H(plaintext)
apply the hash function H
to the plaintext message
a2baf85ad3
hashcode
The quick brown fox jumped over the lazy dog
a2baf85ad3
send hashcode with the message

For network connections we wish to provide:

- **Confidentiality**
  - Protecting information from interception
  - Ensure information remains confidential if intercepted
- **Integrity**
  - Detecting whether information has been tampered with
- **Authenticity**
  - Ensuring we know who the sender is
  - Non-repudiation

Public-key cryptography

- Traditional ciphers depend on the key being known to Alice and to Bob but not to Carol
  - They are called symmetric or shared-secret codes
    - Both E and D functions make use of the same key k
  - It is hard to arrange safe sharing, especially between parties who are not in close physical contact
    - Eg between customer and an ecommerce site
- A different approach uses secrets which are known to only one party - Public Key cryptography
  - Also known as asymmetric encryption

Public key cryptography

Alice wants to send Bob a message. Bob knows a secret or private key $k_{PRIV,Bob}$, and a related (but different) public key $k_{PUB,Bob}$

- Bob announces $k_{PUB,Bob}$ (eg on a web page) to everyone, including Alice
- Alice produces ciphertext using an encryption algorithm E with Bob's public key $k_{PUB,Bob}$ - ciphertext: $c = E(m, k_{PUB,Bob})$
- When Bob receives ciphertext c, he decrypts using an algorithm D with parameter $k_{PRIV,Bob}$
- A public key crypto system has the property that $D(E(m, k_{PUB,Bob}), k_{PRIV,Bob}) = m$
- Also, it must be difficult to work out $k_{PRIV,Bob}$ from knowing $k_{PUB,Bob}$

Note that Alice doesn't need any secret information, to send a confidential

Alice

The quick brown fox jumped over the lazy dog

plaintext

everyone knows this

E(plaintext, Bob's **PUBLIC** key)

everyone knows
the algorithms

7d38b5cd25a2baf85ad3bb5b9311383e67….

ciphertext

Carol

only Bob knows this

D(ciphertext, Bob's **PRIVATE** key)

Bob

The quick brown fox jumped over the lazy dog

plaintext

Example public-key algorithms

- RSA system is the most famous
  - Invented by Ron Rivest, Adi Shamir, Leonard Adleman at MIT in 1977
    - Also done before that by Clifford Cocks (British intelligence services) in 1973 but not disclosed to public
  - Based on prime numbers, and computer arithmetic (exponentiation)
  - it is easy to find prime numbers, and easy to multiply numbers, but hard to factor numbers that are not prime
- Other systems based on more complicated number theory and mathematics
  - Discrete logarithm system
  - Knapsack algorithms
  - Elliptic curve algorithms
- Implementations are slow
  - Many thousand times slower than symmetric encryption algorithms
  - not practical for most communication (but we'll come back to this …)