Public Key Encryption and Message Authentication

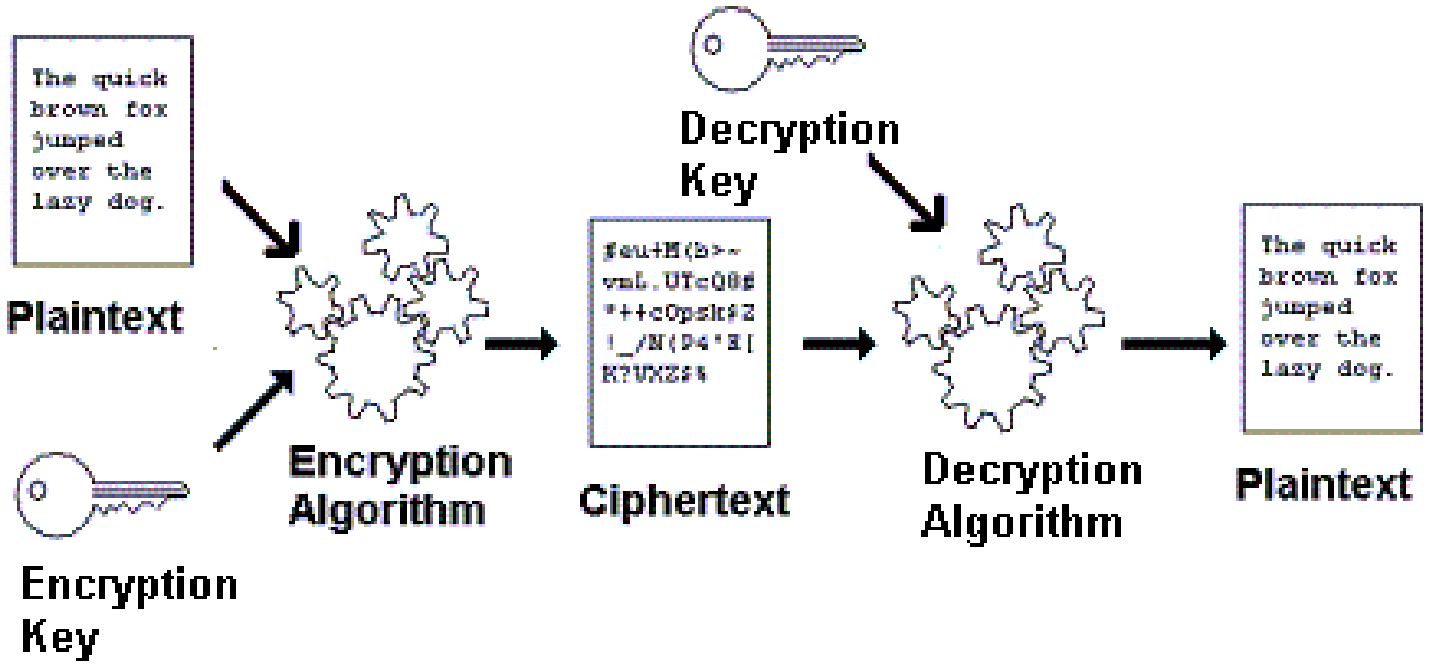
Lecture Summary

- Public-Key Cryptography Principles and Algorithm
- Digital Signatures
- Key Management
- Approaches to Message Authentication
- Secure Hash Functions

Asymmetric or Public Key Cryptography

- In the last two lectures we have looked at symmetric encryption.
- There are an alternative group of encryption algorithms which use different keys for encryption and decryption.
- This is asymmetric or Public Key Encryption.

Encryption using Public-Keys



In Asymmetric Encryption different keys are used for encryption and decryption. Asymmetric Encryption is also known as Public Key Encryption. The decryption algorithm is often the reverse or complement of the encryption algorithm.

Public-Key Cryptography Principles

- The use of two keys has consequences for
- confidentiality,
- what else?

Public-Key Cryptography Principles

- The use of two keys has consequences for
- confidentiality,
- message integrity
  - how does symmetrical encryption achieve this?
  - how does public key encryption achieve this?
- entity authentication

Asymmetric (Public Key) vs Symmetric Cryptography Techniques

- In symmetric cryptography a secret key is shared between 2 hosts.
- In asymmetric cryptography, the secret is personal, no secrets are shared with other hosts.
- In symmetric key cryptography, clear text are permuted, substituted, shifted, xored and swapped.
- In asymmetric key cryptography, large numbers are manipulated.

Public Key Cryptography is mathematically intensive

- Public Key cryptography algorithms use different processing methods than symmetric cryptographic algorithms.
- Much more mathematically intensive than symmetric cryptography.
- Uses exponentiation i.e. raising numbers to large powers.
- Certain mathematical shortcuts are used to perform the calculation.

Public Key Cryptography Security

- Based on the infeasibility of quickly solving some mathematical problems.
- e.g, **RSA** public key encryption scheme is based on the difficulty in finding the factorization of **two large prime** factors.
- The prime factor problem is also the basis of the security of some other public key systems.
- Other difficult mathematical problems are the basis of some public key systems.

Public Key Cryptography is slow

- Numerous public key algorithms have been proposed.
- Many are insecure and therefore, unusable.
- Many of the secure ones are impracticable because they require a very large key or their ciphertext is much larger than their plaintext.
- Only a few can be used for key distribution, encryption and digital signatures.
- Original Public Key Algorithm (Diffie-Hellman) is unsuitable for encryption but can be used for key exchange.

Comparison of Public Key and Symmetric Cryptography

- Symmetric Encryption is faster. Keys are shorter.
- Symmetric Encryption requires a secure method of distributing keys. Public Key encryption does not have this problem.
- In Public Key encryption only the private keys must be kept secret, and each private key is only known to 1 host.
- In a large network of n systems , a Public Key system requires fewer private keys than symmetric system n keys vs n*(n-1)/2 (Think about why?)

Applications for Public-Key Cryptosystems

- Encryption/decryption: The sender encrypts a message with the recipient's public key. The recipient decrypts it with their private key.
- Digital signature: The sender "signs" a message with their private key.
- Says something about public key operation
- Key exchange: Two sides cooperate to exchange a session key.

Commonly used Public Key Encryption Systems

Algorithm	Encryption/ Decryption	Digital Signature	Key Exchange
RSA	Yes	Yes	Yes
Diffie Hellman	No	No	Yes
DSS	No	Yes	No
Elliptic Curve	Yes	Yes	Yes

We will only examine RSA and Diffie-Hellman

Who really discovered public key encryption ?

- The idea of Public Key encryption was first posited by a mathematician working for GCHQ –James Ellis (not Bond) in 1969 but he was unable to develop the ideas further.
- In 1973 a newly recruited mathematics graduate employed at GCHQ – Clifford Cocks developed Ellis ideas further and discovered what was eventually known as RSA encryption.
- Another a newly recruited mathematics graduate at GCHQ – Malcolm Williamson discovered what was later known as Diffie-Hellman key exchange in 1974.
- GCHQ = Government Communications Headquarters –part of the British Secret Service concerned with codes and ciphers.

RSA Public Key Encryption

- First full-fledged public key algorithm –works for public key encryption, digital signatures and key exchange.
- Easiest public key algorithm to understand and implement.
- When implemented in specialised hardware can be very efficiently performed. Unfortunately, when implemented in software it is about 1000 times slower than DES.

Background

Theory 1:

$1 = m^{k(p-1)(q-1)} \bmod pq$

pq is prime number

$0 < m < pq$

k is any integer

Theory 2:

If $a = b \bmod n$

Then $a^c = b^c \bmod n$

pick two large prime numbers: p, q $\quad n = pq$

pick e, d :

e – encryption (part of a public key)

d – decryption (part of a private key)

$$e \times d = k(p-1)(q-1)+1$$

Complete public/private key, is formed from five numbers:

$$p, q, n, e, d$$

(e, n) – encryption keys (public key)

(d, n) – decryption keys (private key)

$$c = m^e \bmod n$$

$$m = c^d \bmod n$$

RSA Public Key Cryptography A Simple Demonstration

- Host B wants to send a message to A. Assume that they want send one character p, with a code number of 5.
- Host B gets A's public key n,e(77, 13).
- Host B calculates pemod n. The output of the calculation is the ciphertext(c).
- i.e. $5^{13} \text{mod } 77 = 1220703125 \text{ mod } 77 = 26$
- Host B sends the ciphertext(c) to A.

RSA Public Key Cryptography ASimple Demonstration

- Host A receives the ciphertextc from B.
- Host A uses the private key n,d(77, 37).
- Host A calculates $c^d \bmod n$
- i.e. $26^{37} \bmod 77 = 5$
- Host A has now recovered the message.

The security of RSA Public Key Cryptography

- The demonstration used a modulus of 77 (n) with factors of 7 and 11. The modulus is in the public key.
- If the modulus can be factored by a cryptanalyst, then they know d and e.
- If this happens the encryption is broken because.
- They can calculate the product of (d-1)*(e-1) = S.
- Knowing S, and the public key modulus (n) and exponent (x) , they can then calculate y. This because x * y mod n = 1
- Knowing y, they can decrypt the message.

The security of RSA Public Key Cryptography

Host A

public key = { modulus n, exponent x}

private key = { modulus n, and exponent y}

The modulus consists of a very large number with prime factors d and e.

The exponents x and y are two numbers chosen so that
x * y mod (d-1)*(e-1) = 1

Host B uses Host A's public key to encrypt messages it sends to A. That's all it needs to know.

If d and e can be calculated, then x and y can be calculated.

In practice d and e are LARGE numbers (512+ bits) , so the modulus n will also be large (1024+) bits. This makes the factorisation of n into its prime factors d and e a task which is computationally infeasible.

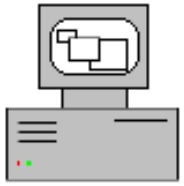
Strengthening RSA Public Key Cryptography

- The main defense (and it is an effective one) is to make the two original prime numbers very large (of the order of 512 bits each).
- The modulus will also be large (approximately 1024 bits).
- *p* and *q* should not be close to each other.
- Message may be padded with random data.
- Various other rules based on theoretical and experimental results are also applied.

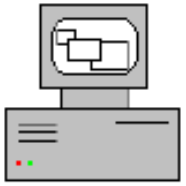
Key Exchange Protocols

- Key exchange is designed to allow 2 hosts to exchange a secret key securely. The secret key can then be used to exchange encrypted messages.
- The Diffie-Hellman protocol is a popular key exchange protocol.
- The keys exchanged between the 2 hosts can be used for symmetric or public key encryption.
- Establishes a shared key between 2 hosts, but does not automatically authenticate the hosts sharing the keys.

Diffie-Hellman Key Exchange



Host A



Host B

Host A and B decide on a modulus m and a number g. They exchange the values of these numbers publicly. So they are also known to eavesdroppers.

Host A makes up a number x.
Host A calculates

$R1 = g^x \bmod m$

and sends it to Host B

Host B makes up a number y.
Host B calculates

$R2 = g^y \bmod m$

and sends it to Host A

Host A calculates

the Key = $R2^x \bmod m$

Host B calculates

the Key = $R1^y \bmod m$

Both sides now have the key which is equal to = $g^{xy} \bmod m$.

Anyone watching the exchange sees g, m, R1 and R2. They do not see x or y and so they cannot calculate K.

Diffie-Hellman vulnerabilities

- No built in authentication of parties exchanging values. This makes it vulnerable to a **man-in-the middle attack.**
- This attack can be countered by using a Trusted Third Party (TTP) or X.509 Certificate.
- Also vulnerable to mathematical attack (Discrete Logarithm Attack).
- Discrete Logarithm attack can be overcome by using a modulus which is a Prime Number and very large (at least 300 decimal digits)

Digital Signatures

- Third area of application of Public Key Cryptography.
- Used to provide the lesser security goal of Message Origin Authentication.
- Can also be used to provide the major security service of message integrity.
- In order to understand Digital Signatures, we need to understand hashes.

Integrity

- Message has not been altered in transit.
- Integrity is also sometimes known as Message Authentication.

We may also wish to establish that:

- Message comes from its alleged source (Message Origin Authentication).
- Message has not been artificially delayed in transit (timeliness).

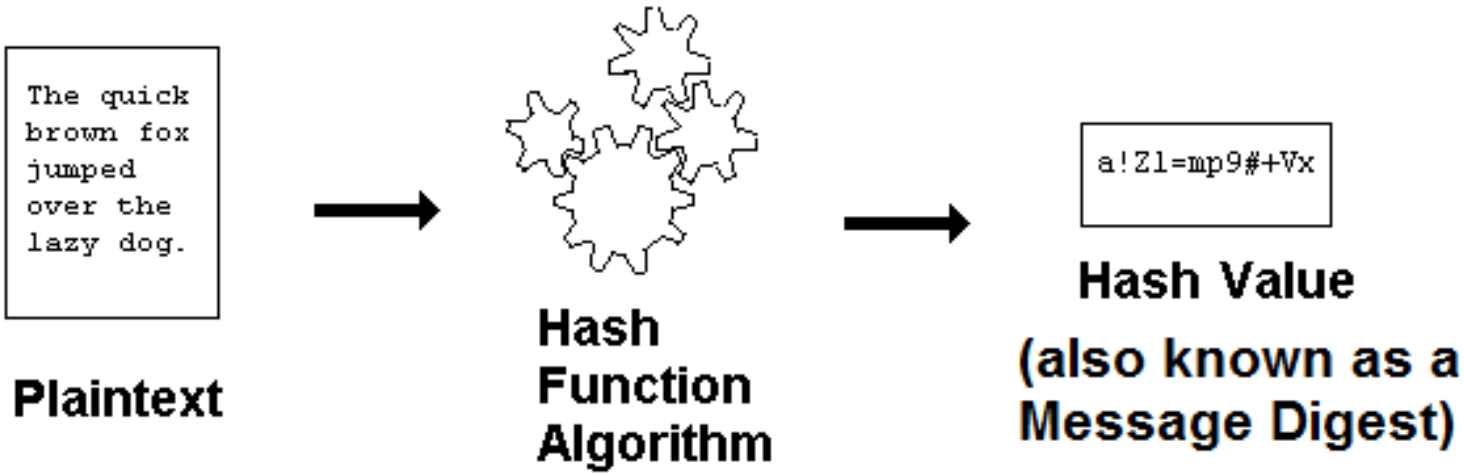
Message Integrity Tools

- The primary tool used to provide integrity is the Hash Function

There are 2 types of HASH functions

- Unkeyed Hash Functions –these are hash functions that do not require any input other than the message that is being hashed.
- Keyed Hash Functions –these require a key as well as the message to be hashed

Hash Functions

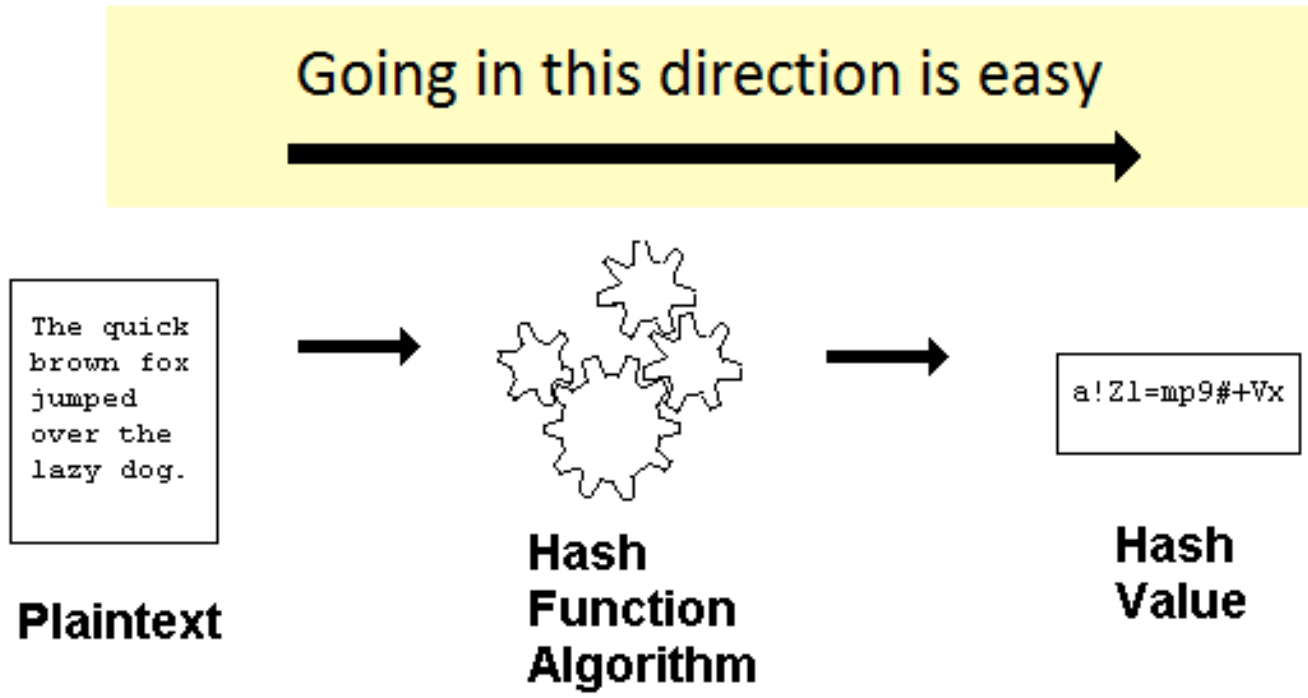


A hash function operates on a message to produce a (usually) much shorter value. Even a small change in the original message will produce a completely different hash value.

Knowing the value of the hash, it should be difficult to produce the original message or any other message that produces the same hash value.

A hash is an example of a one-way function - It is computationally difficult to produce an input message that produces a given output or hash value.

One Way Functions



Hash Function Collision

- If 2 different messages are fed into the same hash function and produce the same hash value this is known as a collision.
- The frequency of collisions is to be minimized.
- Good hash functions spread the outputs evenly over the set of possible results.

Mathematics of Hash Functions

- Imagine:
  - a hash function produces **128** bit hash
  - a pool of **1000** bit messages.
- Some messages are going to produce duplicate hashes, since it's not **1-to-1!**
- On average there will 2^{872} messages producing the same hash. This is a lot, but....
- Given any one 128 bit hash value, we will need to search $2^{128/2} = 2^{64}$ messages on average (why?) before we find a match. At the present time searching 2^{64} messages is computationally infeasible.
- Searching 2^{32} messages IS computationally feasible, so 64 bit hashes are insecure.
- All hashes currently used are at least 128 bits in length.

Birthday Attack

1. Original message (a contract of some sort) is copied into a large number of variations which differ in some minor detail e.g. space, punctuation, synonyms. The hash value of all variants of original is calculated.
2. False message is written with a lot of variations as outlined for original message. Hashes are calculated for all false messages.
3. Pool of original message variant hashes and false message variants are both searched for matching hashes - one from each pool.
4. Hashed original variant with hash is sent to receiver who signs it. False variant message which has the same hash value as the original signed variant is substituted.
5. It now appears that the receiver signed the false message.

Why is it called the Birthday Attack?

- Birthday is like a hash function!
- How many people do you need to put into a room before there is a 50% probability that at least 2 will have the same birthday (23 , 33 ,63, 103, 203) ?
- The number is surprisingly small. See MATLAB demo
- The birthday attack uses this high probability to find matching hashes from 2 pools.
- In general, a hash length of >=128 bits secures hash functions against a birthday attack.

Hash Standards

- SHA-1, SHA-2, SHA-3

SHA = Secure Hash Algorithm

- MD5

MD = Message Digest

- Other hash standards

SHA -1

- Released by NIST in 1993 as FIPS-180. Updated and re-released as FIPS-180-1. Hash length for FIPS-180-1 is 160 bits
- Weaknesses have been found in SHA-1 that enable more than one message with the same hash to be found in 2^{69} operations rather than the 2^{80} operations originally claimed.
- Notwithstanding this flaw, it is still regarded as reasonably secure and is widely used.

SHA-2

- Published by NIST in 2001
- More secure than SHA1.
- Several variants producing hashes of 224, 256, 284 and 512 bits.
- Not widely used for a number of reasons
  - SHA-1 is regarded as reasonably secure
  - Some users are waiting for the release of SHA-3 which is expected to happen in the next few years.
  - Not supported on Windows versions up to and including XP SP2

MD5 Hash Algorithm

- 128 bit hash code
- Developed by Ron Rivest
- Very widely used
- Brute Force and Cryptanalytic weaknesses have been discovered in the last few years.
- Has weakness to birthday attack
  -approximately 2^{64} operations required to create 2 different messages with the same hash.

Keyed Hashes

- All the hashes discussed so far only have the message as input.
- A second group of Hashes have a key as well as the message as the input.
- This group of hash functions make use of a secret key shared between the sender and receiver of the message.