

Reasons to Use Cryptography

- Privacy

When you want to ensure that a message is only seen by the intended recipient.

- Authenticity

When you want to be sure that only the person who claims to have sent the message can be the actual sender.

- Integrity

When you want to be sure that a message has not been tampered with while in transit between the sender and the recipient.

These correspond to protection against interception, modification and fabrication, but not interruption security threats.

Definition of Terms

- plaintext
the original message
- encryption algorithm
process by which the plaintext is rendered unintelligible to casual viewers.
- secret key
input to the algorithm along with the plaintext that affects the output of the algorithm
- ciphertext
the unintelligible output of the algorithm
- decryption algorithm
the encryption algorithm run in reverse

Definition of Terms (cont.)

- encoding

transforming input from one format type to another. e.g. The base64 encoding is used to encode 8-bit binary data into ASCII text using the letters a-z, A-Z, 0-9, / and +. This is not encryption.

- decoding

Reversing the encoding process. e.g. Converting base64 text into a binary file.

aGFwcHkgYmlydGhkYXk=

"happy birthday"

Cryptographic Systems Features

Three independent dimensions:

- **Type of operations used:**
substitution/transformation or product/feedback
- **Number of keys used:**
no key, single key or two key
- **Way plaintext is processed:**
input is processed in blocks or as a continuous bit stream

Cryptanalysis

Types of Attacks:

(ordered from least to most a priori knowledge)

- Ciphertext only

algorithm, target ciphertext

- Known plaintext

algorithm, target ciphertext, sample plaintext-ciphertext formed with secret key

- Chosen plaintext

algorithm, target ciphertext, another chosen plaintext message with corresponding ciphertext formed with secret key

- Chosen ciphertext

algorithm, target ciphertext, another chosen ciphertext message with corresponding plaintext formed with secret key

- Chosen text

combination of chosen ciphertext and chosen plaintext

Cryptography Strength

Referred to by size of the keys in bits.

Smaller key sizes are more vulnerable.

Secret and public cryptography algorithms are very different mathematically. As a result they are susceptible to different types of attacks. As such the key lengths that are considered weak are very different for each type.

An algorithm is "computationally secure", if the cost of breaking the cipher is greater than the value of the information, or the time required to break the cipher exceeds the useful lifetime of the information.

Basic System Types

- One Time Pads

No algorithm. Randomly generated pads are the shared knowledge necessary to communicate.

- No Key

Algorithm is the shared secret.

- Secret Key

Algorithm is public, but keys are secret.

- Public Key

Algorithm is public, as is half of the key pair. The other half is not shared with anyone.

Analogies For Basic System Types

- One Time Pads

Your locks require two people to operate, and each lock is inherently different from all others, but they are cheap.

- No Key

Everything you secure has a unique type of locking mechanism, and can only be accessed by those who understand particular lock they have to open. This scheme requires a lot of locks!

- Secret Key

Everything you secure has the same type of locking mechanism, and you can have as many copies of the keys as you like to share as you like. One well designed lock is enough.

- Public Key

Locks are replaced by trapdoors. Anyone can push stuff in to your secure area, but only you can operate the door that drops the secured items out the bottom.

Common Secret Key Algorithms

- DES 56 bits
- TDES 112 or 168 bits
Really 168, but effectively only 112.
- RC2 40, 64 or 128 bits
- RC5 variable to 2048 bits
- Blowfish variable to 448 bits
- AES (Rijndael) 128, 192 or 256 bits
This algorithm replaces DES as the US standard cipher.

Secret Key Math in a Nutshell

The bits of the plaintext are shuffled repeatedly in specific patterns with bits from the key shuffled in repeatedly.

The value of the key alters the shuffling process.

This whole process is repeated a number of times.

The difference between a good and a bad shuffling algorithm is judged by how closely the shuffled values resemble truly random data.

Secret key is also known as symmetric key.

This is because the algorithm is basically reversed for the decryption process.

Secret Key Strengths and Sizes

Assuming that the algorithm is mathematically sound, the number of attacks against shared secret keys is few. Basically you are left with brute force attacks. These are attacks where you try all possible secret keys until you find the correct one. It is also referred to as an exhaustive search.

In general you will have to try half of the possible keys before you will find the key you are looking for.

Assuming you can try 1 million keys per second, which is what the EFF DES Cracker could do in 1998, then you have the following times for exhaustive key searches at various key lengths.

32 bits	2.15 milliseconds
56 bits	10 hours
128 bits	5.4×10^{18} years
168 bits	5.9×10^{30} years

Drawbacks of Secret Key

1. Both parties must agree upon a key before secure communication can begin.
2. Both parties must store the secret securely; The sender until the message is encrypted, and the recipient until the message is decrypted.
3. A separate key is needed for each private channel of communication between individuals. For 10 people who want to communicate privately with one another, that is 45 keys that each participant must manage!
4. It is not possible to prove who sent a message. Both parties have the SAME SHARED key. This is a problem when the secure communication must be revealed to a third party. The third party cannot tell the difference between authentic and forged messages.

Common Public Key Algorithms

- **RSA**

Most popular and most analyzed. Hampered in its adoption until its patent expired.

- **DSS/DSA**

Another public key algorithm that was designed for authentication only. It is not useful for encryption.

Public Key Math in a Nutshell

The basic assumption that it relies upon is that "modular exponentiation" is relatively easy, but the inverse operation, calculating the "discrete logarithm", is very hard.

This is definitely true of conventional digital computers, but not true of quantum computers. There are no practical quantum computers yet, so for now public key algorithms are safe for the foreseeable future.

The same mathematical process is repeated for both the encryption and decryption process. The only difference is that in one case the private key is used, and the other the public key.

"Modular exponentiation" is the act of raising a number X to its Y th degree and then performing modular division on it with Z .

Note that there is no passphrase or secret key associated with public key algorithms. Possession of the bit pattern is paramount.

Use of Secret Keys

Since there is no passphrase built into a public key, the image of the public key on disk is usually protected using a symmetric algorithm such as TDES. This is why you need a passphrase to unlock your keys for use.

Also public key encryption is a slow process when compared with secret key. Therefore messages are actually encrypted with a completely random session key. This session key is then encrypted using the public key.

The session key and hashes are the only things encrypted by public key algorithms regularly.

Use of Hashing

MD5 and SHA-1 are cryptographic hash functions. They are similar to encryption algorithms in that the calculated value appears to be strongly random in relation to the data being hashed. This makes it very difficult for a hash value to be forged.

One key difference though is that the output of a hash algorithm is always the same size regardless of the size of the input. 128 bits for MD5 and 160 bits SHA-1.

Theoretically there are many plaintexts that have the same hash value. This is not a problem in practical terms, because the hashes are very large numbers and are very unlikely to collide.

If you have a particular hash value, it is correspondingly difficult to construct a file that will produce the hash value.

Use of Entropy

Good entropy, i.e. a source of truly random data, is essential to public key systems in two ways.

1. The large primes that go into making the key pair must not be predictable. If they are, then knowing one key pair would allow you to guess others.

2. The session keys must not be predictable. If they are then, you can guess them and decrypt the message without even needing to recover the public key pair.

With a complete lack of entropy, the same public key pairs would be generated each time a new one was needed. Similarly the session keys would all be identical as well.

Putting It All Together - Signatures

○ Signing:

1. Compose a message, the plaintext.
2. Calculate the MD5 hash of the message.
3. Encrypt the hash value with your private key.
4. Concatenate the encrypted hash to your message.
5. Send it to the recipient.

○ Signature Verification:

1. Receive a message with plaintext and an encrypted hash.
2. Separate the encrypted hash from the plaintext.
3. Decrypt the MD5 hash using the sender's public key.
4. Calculate the MD5 hash for the plaintext.
5. Compare the received hash to the one just calculated. If it matches, the message is authentic. If not, it has been altered in transit.

Putting It All Together - Encryption

○ Encrypting:

1. Compose a message, the plaintext.
2. Create a random session key.
3. Encrypt the message using TDES or AES and the session key.
4. Encrypt the session key with the recipient's public key.
5. Concatenate the encrypted session key to your ciphertext.
6. Send it to the recipient.

○ Decryption:

1. Receive a ciphertext message with an encrypted session key.
2. Separate the encrypted session key from the ciphertext.
3. Decrypt the encrypted session key using your private key.
4. Decrypt the ciphertext using the session key.
5. Read the plaintext.

Public Key Strengths and Sizes

Public keys are a product of very large random prime numbers. If you are able to factor the primes that go into the keys, you have effectively recovered the missing key pair.

Factoring is well understood and as better algorithms are found and computers get larger and faster, larger keys will become breakable.

To factor a 1024 bit key (roughly 300 digits) with current technology would cost in the millions of billions of dollars. 512 bit public key sizes (roughly 150 digits) were used at one time, but the cost of recovering this only one million dollars today. This is feasible today.

Factoring cannot be done statistically or efficiently using brute force on such large numbers.

Drawbacks of Public Key

1. The algorithms involve large numbers, e.g. 1024 bit, which makes them slow.
2. Requires a good source of entropy.
3. Finding a public key for a recipient may be hard.
4. You need a third party to vouch for public keys, or you are never really sure who it belongs to. i.e. Who has the private key.
5. Key generation is slow.

Advantages of Public Key

1. One key pair per participant allows you to communicate with all other participants securely. e.g. 10 people, 10 key pairs.
2. A public key can be retrieved from any source, even an malefacting one and still be useful to secure communication.
3. The private key can have a long lifetime if properly secured.
4. You do not need to communicate with the sender by an alternate mechanism before you begin sending them secure messages.
5. Digital signatures provide third party recognizable authenticity.