Data Structures and Algorithms Cryptography

Goodrich & Tamassia Sections 3.1.3 & 3.1.4

- Introduction
- Simple Methods
- Asymmetric methods: public and private keys.

Cryptography: Motivation

Many areas have sensitive information, e.g.

- Finance: many transactions encrypted, e.g. PIN on ATM card
- Commerce: sensitive negotiations, electronic commerce
- Military: battle plan, weapon control
- Diplomacy: sensitive negotiations

In each case:

- Trying to communicate at a distance.
- Others could easily eavesdrop, or insert messages
- You don't want them to know the content of your message (e.g., credit card details; battle plan).

So use some secret writing technique(s) so the message can only be understood by intended recipient.

Predates computing, although early computers were developed to break encryption, e.g. the Enigma code.

Secret Writing Methods

Two alternatives:

- **Steganography**: hidden writing, e.g. invisible ink, using low-definition bits of sound/graphics files. **Big advantage:** don't reveal to enemy that you have information to hide.
- **Cryptology**: scramble the message. Has two disciplines:
  - **Cryptography**: developing codes/ciphers
  - **Cryptanalysis**: breaking codes/ciphers

Cryptographic Terminology

There are two main encryption methods:

- **Transposition:** reorder character/word by another
- **Substitution:** replace 1 character/word by another

Either method can be applied to words or characters: **Codes** operate on words, where **Ciphers** operate on characters.

Encryption methods may be

- **Symmetric**: same key used by sender and receiver. Hence **vital** that key doesn't get into hands of the enemy.
- **Asymmetric**: sender uses one key to encrypt, receiver uses another to decrypt. Hence senders key may be **public**.

Encryption Example

```
PLAINTEXT:
'attack at dawn'
  +---+                        +---+
  |   |                        |   |
  +---+         KEY            +---+
    |       ===============>     |
    |                            |
    |          CIPHERTEXT        |
   / \      --------------->    / \
  /   \          \             /   \
 SENDER           \           RECEIVER
                   v
                 +---+
                 |   |
                 +---+
                   |
                   |
                   |
                  / \
                 /   \
                 ENEMY
```

A typical method:

- Sender transmits **key** to reciever (may need to be over a **secure channel**)
- Sender **encrypts plaintext** using key to produce **ciphertext**
- Sender transmits ciphertext over **insecure channel**
- Receiver uses key to **decrypt** ciphertext.
- **Enemy** or **cryptanalyst** intercepts message and attempts to **break** the encryption, i.e. decrypt it without the key.

Must be sufficiently hard to break the encryption that it isn't worth the enemies time!

Symmetric Methods
Columnar Transposition Cipher

To encrypt: Write the plaintext in columns of depth k (= key), padding (with *) the end as necessary, read off in rows, e.g. if key = 4.

Plaintext: `Transposition_ciphers_are_easy!`

```
T  s  i  n  p  s  e  s
r  p  t  _  h  _  _  y
a  o  i  c  e  a  e  !
n  s  o  i  r  r  a  *
```

Ciphertext:
`Tsinpsesrpt_h__yaoiceae!nsoirra*`

To decrypt: write ciphertext as rows of (message length/k), and read off columns.

To break: try every key between 1 and message length (easy)

Caesar Cipher
A Substitution Cipher

Key = k (some small integer)

To encrypt: replace Nth letter of alphabet with (N+k)th, e.g. k = 2

```
a b c d e f g h i j k l m n o p q r s t u v w x y z _
c d e f g h i j k l m n o p q r s t u v w x y z _ a b
```

So plaintext `attack` is encrypted as `cvvcem`.

To decrypt: Replace Nth letter by (N-k)th.

To break: Try all values of k until get something meaningful (very easy!)

Better Method

To encrypt: Use a fixed permutation of the alphabet as the key, i.e. map each character to another e.g.,

```
a b c d e f g h i j k l m n o p q r s t u v w x y z _
q w e r t y u i o p a s d f g h j k l z x c v b n _ m
```

So plaintext `badly` is encrypted as `wqrsn`.

To break: You might think this would be hard to break: there is an enormous choice of keys (26!), but providing there's enough cipher text it's easy to break using simple statistical methods.

- The occurrence frequencies of each character in a language is well known, e.g. in English 'e' is most common, then 't' etc.
- Also look for common short words, e.g. digrams ('an', 'at','as') and trigrams ('the', 'and')

Exercise: Use the permutation above to encrypt

`three_million`

Vigenere Cipher

Like simple Caeser cipher, but use a repeated key to specify "k" for each character in plaintext:

```
key: abc = (k values 1, 2 3)

plaintext:  attack
key         abcabc..
k           123123

ciphertext: bvwben
```

Harder to decrypt. Can't use character frequencies, and with long key, will take a long time to try all possible keys.

Exercise: Use the Vignere cipher to encrypt `three_million` using key `bad`

Vernam Cipher (One time pad)

- Use above method, but with key as long as plaintext
- Only use it once!

VERY secure; no way to decrypt as equally likely to end up with any message at all.

Problems memorizing/distributing key. But useful in situations where a key can be sent ahead of time.

Possible keys:

- A book: Alice in Wonderland, edition X, starting on page 102, line 4.
- Pads containing 100s of pages of keys.

Exercise: Use the Vignere cipher to encrypt `three_million` using key `sadbadmadladhad`
Exercise: Is `sadbadmadladhad` a good key? Why?

Simple Cryptanalysis Methods

How do we break ciphertexts?

- Brute force: try all possible keys.
- Statistical: look at character frequencies.
- Cribs: Look for known parts of message e.g., email/html header, date, time, spaces.
- Or a combination. Try all likely combinations, once other methods have suggested most likely ones.

Pseudo keys

How do we approximate the security of the one-time pad, without the overhead of needing a very long key?

- Generate a very long (longer than plaintext) key from a shorter ones, in a pseudo- random fashion.
- For example a simple encryption machine, takes the 'true' key (cryptovariable):
  - Generates long key from it as a stream of bits.
  - XORs this with the binary of the plaintext.

```
true key:   16256
pseudokey:  100101100011..
plaintext:  1001110100..
ciphertext: 0000101100
```

- Decrypt using exactly same method.

Exercise: Locate and experiment with a Java pseudorandom number generator

DES - The Data Encryption Standard

The Data Encryption Standard, or DES, was the first official U.S. government cipher intended for commercial use.

DES is the most widely used cryptosystem in the world.

DES is a block cipher, which operates on 64-bit data fragments, using a 56-bit key.

DES Operation

```
64-bit Plaintext
      ||
      \/                       56-bit key
Transposition 1                   ||
      \/                          \/
Substitution 1 <--------- Pseudo Key Generator
      \/                /
       .               /
       .              /
                     /
Substitution 16<-----
      \/
Transposition 2
      \/
Transposition 3
      ||
      \/
64-bit Ciphertext
```

The substitution stages in DES re-arrange the order of the bits from the previous stage, and use XOR to combine them with the key.

The effectiveness of DES is based on the complexity of the 19 stages.

In the above diagram, two identical 64-bit plaintexts will result in identical ciphertexts. This is called the Electronic Code Book (ECB) mode of operation.

DES In Practice

The ECB mode of operation is now rarely used, since it is now generally agreed that it is breakable given sufficient resources

In the Chain Block Cipher (CBC) mode, each block of plaintext is exclusive-ORed with the ciphertext output from the previous encryption operation. Thus, the next block of ciphertext is a function of its corresponding plaintext, the 56-bit key and the previous block of ciphertext. Identical blocks of plaintext no longer generate identical ciphertext, which makes this system much more difficult to break.

The CBC mode of DES is the normal method for encryption in modern business data communications.

DES Evaluation

The original (IBM) algorithm which was the basis of DES used a 128 bit key. The US government changed this to a 56 bit key. This made DES considerably less secure. Many people believe that the US government has a ”backdoor” (or trapdoor) decryption technique which is infeasible at 128 bits, but possible with 56. This has never been confirmed.

56 bit key DES is nowadays regarded as ”probably crackable”. A variation called triple encryption DES uses two keys (112 bits) and is currently considered sufficiently secure.

The Advanced Encryption Standard

Other encryption standards exist.

The Advanced Encryption Standard (AES) is intended to be a cipher that will remain secure for several decades from now.

AES supports key sizes of 128 bits, 192 bits, and 256 bits, in contrast to the 56-bit keys offered by DES.

The AES algorithm was published in November 2002, and is now gaining widespread use, and superseding DES.

See good article on Wikipedia.

Java Cryptography Engine

Java programs execute over insecure networks, and hence are vulnerable to security attacks

The Java Cryptography Engine(JCE) is part of the `java.security` package & contains methods to:

- Generate key pairs
- Construct `cipher` objects, with encryption/ decryption methods, e.g.

  ```
  Cipher ciph = Cipher.getInstance(‘‘DES’’)
  ```

- Initialise cipher objects, e.g.

  ```
  ciph.initEncrypt(keys)
  ```

- Encrypt/Decrypt the data, e.g.

  ```
  byte[] ciphertext = ciph.crypt(cleartext)
  ```

Asymmetric Methods: Public Key Cryptosystems

- How to safely send key from sender to receiver?
- Avoid problem by using different keys to decrypt/encrypt.
- Sender uses PUBLIC key to encrypt. Receiver uses PRIVATE key to decrypt.
- Ciphertext can NOT be decrypted using public key. So public key can be published, like telephone directory. Private keys never transmitted.

Public Key Example

Suppose you want to send a secret message M to John.

- Look up John's public key. Encrypt your message using it, and transmit to John.
- John then uses his private key to decrypt.

More formally, given message M, public key P, and private key S, then encrypted message is P(M) and decrypted message is S(P(M)). We must guarantee that S(P(M))=M, **and** can't decrypt knowing P.

Challenge: How do we come up with suitable public and private keys to make it work?

RSA Public Key Encryption

Developed by Rivest, Shamir and Aldeman at MIT around 1977, represents public and private keys as pairs of big numbers (N, P) and (N, S). Message also represented as a number M.

Prime number theory used to find suitable values for S, N and P, as follows:

Choose three prime numbers x,y,z (say 3, 5, 7).
S=largest (i.e. 7)
N=product of other two (i.e. 15)
P chosen so PxS % (x-1)(y-1) = 1
i.e. 7P % 8=1. e.g. P=23

To encrypt: $C = M^P \% \ N$.
e.g., M=2, $C = 2^{23} \% \ 15 = 8$

To decrypt: $M = C^S \% \ N$.
e.g., $= 8^7 \% \ 15 = 2$

To break: Very hard! We know from number theory that factoring the product of large primes is computationally very expensive.

In practice keys are primes with **hundreds** of digits.

As it's believed to be relatively secure, RSA is widely used: e.g. in Internet Browsers (e.g. Mozilla, Netscape) and PGP (Pretty Good Privacy).

RSA Session Keys

- Although secure RSA encryption needs hard maths to construct the keys! Encryption/decryption time consuming.
- So method often used to encrypt *keys* not whole message. You send a message with an RSA encrypted KEY. Decode it using RSA to get ordinary key. Then decode message with the ordinary key using symmetric methods.
- The ordinary key is referred to as the session key. Can be randomly generated, and only used once.

Internet Browser Encryption

Browsers like Mozilla & Netscape use:

- RSA key encryption to encode a session key.
- Session key used to encrypt/decrypt subsequent messages.

In practice:

- Client finds server's public key.
- Generates a random session key.
- Encrypts session key with server's public key (using RSA).
- Encrypts message with session key (using simpler methods)
- Sends encrypted session key + encrypted message.
- Server decrypts session key using private key. Decrypts message using session key.

Social and Ethical Issues

Encryption world is secretive: often organisations don't reveal if a code can be broken.

Governments

- monitor communications, e.g. GCHQ and phone satellites scan phone calls and emails for keywords like 'bomb' 'Columbia', 'CIA'
- seek to break codes

Individuals have a right to privacy, and commonly available software, e.g. PGP freeware, secure communication, encrypted disks: disliked by governments.

In the context of terrorism, crime, drug trafficking and money laundering should secure communication be widely available?

Summary

- Simple Symmetric methods: Columnar trans. Caesar, Vernam
- Simple codes&ciphers broken using frequencies, cribs, brute force.
- Pseudo-keys: long keys generated from shorter ones; harder to break.
- Standards: DES, AES
- Asymmetric methods, e.g.RSA public key encryption: needn't transfer key.
- WWW: public key encryption of session keys.
- Social & Ethical Issues