Encryption Details

COMP620

Goals for Today

- Understand how some of the most common encryption algorithms operate
- Learn about some new potential encryption systems

Substitution Permutation Ciphers

- A Substitution Permutation encryption algorithm typically involves three phases, which are often repeated
- **Substitution** – the substitution of a bit pattern with another
- **Permutation** – the rearrangement of the bits
- **Exclusive OR** with a key

Substitution Permutation stages

- The K box XORs the input with the key for that round
- The S box performs a substitution

Diagram from "Cryptography Theory and Practice", 3rd ed. by Douglas Stinson
S Box

• An S box performs a substitution.
• The substitution can be efficiently implemented by a look up table
• Example of a 3 bit to 3 bit substitution

<table>
<thead>
<tr>
<th>Input</th>
<th>000</th>
<th>001</th>
<th>010</th>
<th>011</th>
<th>100</th>
<th>101</th>
<th>110</th>
<th>111</th>
</tr>
</thead>
<tbody>
<tr>
<td>Output</td>
<td>101</td>
<td>010</td>
<td>110</td>
<td>000</td>
<td>111</td>
<td>001</td>
<td>011</td>
<td>100</td>
</tr>
</tbody>
</table>

S Box Expansion or Contraction

• The number of bits on the input of an S box does not have to match the number of bits on the output
• Example of a 3 bit input with 2 bit output

<table>
<thead>
<tr>
<th>Input</th>
<th>000</th>
<th>001</th>
<th>010</th>
<th>011</th>
<th>100</th>
<th>101</th>
<th>110</th>
<th>111</th>
</tr>
</thead>
<tbody>
<tr>
<td>Output</td>
<td>10</td>
<td>01</td>
<td>11</td>
<td>01</td>
<td>11</td>
<td>00</td>
<td>10</td>
<td>00</td>
</tr>
</tbody>
</table>

Permutation Expansion

• The number of bits on the output of a permutation does not need to match the input
• Some input bits can go to multiple output bits

Stage Keys

• The key used at each stage is a function of the original key
• Before each stage the key is modified to produce a unique key for that stage
• Some stages might use only some of the key bits
Data Encryption Standard

- Originally developed by IBM
- Adopted as a standard in 1977
- Was the most widely used cryptosystem in the world
- DES uses a 56 bit key
- 64 bit blocks of data are encrypted

DES Algorithm

- DES is a substitution permutation cipher
- There are 16 stages
- The data is split into the left and right half. Each 32 bit half is handled differently
- The 56 bit key is divided into two 28 bit halves which are used to create unique 48 bit keys for each stage

DES Stage

- The 56 bit key is split into two 28 bit halves
- Each half is rotated 1 or 2 bits to the left
- 48 of the 56 bits are selected for the stage key
- Each bit is used in 14 of the 16 stages

DES Key Schedule
**Advanced Encryption Standard**

- AES is also known as the *Rijndael* algorithm
- Selected in 2000 as the new standard after an open international competition
- Created by Belgian researchers Rijmen and Daemen
- Available world-wide royalty free
- AES encrypts blocks of 128 bits
- Keys can be either 128 bits, 192 bits or 256 bits
- AES operates on a 4×4 array of bytes, termed the *state*

---

**AES Algorithm**

**Initial Stage**
- AddRoundKey

**Each Stage**
- *SubBytes*—a substitution step
- *ShiftRows*—rows are shifted cyclically
- *MixColumns*—each column of the state is multiplied with a fixed polynomial
- *AddRoundKey*—each byte is XOR with the stage key

**Final Stage** (no *MixColumns*)
- *SubBytes*
- *ShiftRows*
- *AddRoundKey*

---

**AddRoundKey Step**

- Each byte of the data is XOR with the key for that stage

---

**SubBytes Step**

- Each byte in the array is updated using an 8 bit substitution box, the Rijndael S-box.
ShiftRows Step
• This cyclically shifts the bytes in each row by a certain offset.

MixColumns Step
• Each column is multiplied by a fixed polynomial. All four input bytes determine the four output bytes.

AES Implementation
• It is possible to speed up execution of this cipher by combining SubBytes and ShiftRows with MixColumns, and transforming them into a sequence of table lookups
• This requires 4KB of lookup tables
• A stage can now be done with 16 table lookups and 12 XOR, followed by four XOR in the AddRoundKey step
• Intel and AMD are adding instructions to the Pentium to perform an AES stage

RSA Algorithm
• RSA is an asymmetric encryption algorithm developed in 1977 by Rivest (author of COMP785 text), Shamir and Adleman
• Secretly developed by Clifford Cocks in 1973
• The most commonly used public key algorithm
• Now in the public domain
RSA Encryption

• The sender encrypts message $m$ as
  $$c = m^e \mod n$$
• $c$ is sent to the receiver
• The receiver computes
  $$m = c^d \mod n$$

RSA Algorithm

• Select secret primes $p$ and $q$ and form $n = p*q$
• Choose $e$ with $\gcd(e, (p-1)*(q-1)) = 1$
• Compute $d$ such that $d*e = 1 \mod (p-1)*(q-1)$
• The public key is $n$ and $e$
• The secret key is $d$

RSA Example

• Let $p = 101$ and $q = 113$
• $n = 11413$
• $\Phi(n) = (p-1)*(q-1) = 11200$
• Choose $e = 3533$ (gcd of $e$ and $\Phi(n)$ is 1)
• $d = e^{-1} \mod 11200 = 6597$
• Assume $m = 9726$
• Encryption: $c = 9726^{3533} \mod 11413 = 5761$
• Decryption: $m = 57616597 \mod 11413 = 9726$

Sizes

• Typically large values of $p$ and $q$ are selected to make it difficult to factor.
• Therefore $n$ is very large, $\log_2 n$ bits in length
• The pieces of plain text to encrypt, $m$, must be $0 < m < n$
• The cipher text can be up to $\log_2 n$ bits long
Another RSA Example

- Let $p = 885320963$ and $q = 238855417$
- $n = 211463707796206571$
- Choose $e = 9007$
- $d = e^{-1} \mod 11200 = 116402471153538991$
- Assume $m = 30120$
- $c = 113535859035722866$
- Note that $c$ is much larger than $m$
- It requires more space to store $c$ than $m$

RSA Key Concern

- For $n = pq$, if we know the first or last $(\log_2 n)/4$ digits of $m$, we can efficiently factor $n$
- If $p$ has 100 digits, then if we know the first or last 50 bits of $p$ or $q$, we can factor $n$
- Imagine we select $p$ and $q$ by picking a random 50 bit number and multiplying it by $2^{50}$ to get 100 bits. We then add 1 and test for primality until we get a prime
- We can guess the lowest 50 bits of the number

Knapsack Problem

- The knapsack algorithm is an NP-Complete problem
- Consider a set of integers $x_0 \ldots x_n$ and a number $E$ which is greater than any $x_i$
- Can you create a set of integers $a_0 \ldots a_n$ such that

$$E = \sum_{i}^{n} a_i \cdot x_i$$

Easy Knapsack Problem

- The super increasing knapsack algorithm has a linear solution
- Consider a set of integers $x_0 \ldots x_n$ such that each $x_i$ is greater than the sum of all $x_i$ before it.
- It is easy to solve this problem
Knapsack Encryption Algorithm

- R. Merkle and M. Hellman devised an asymmetrical encryption algorithm based on the knapsack problem.
- A super increasing set of $x_i$ is transformed into a regular knapsack set of $x_i$ by a method known only to the secret key holder.
- The public key is the normal knapsack set of $x_i$.

Knapsack Encryption

- To encrypt message $b$, let $b = b_1b_2...b_n$ (in binary format).
- Compute $y = b_1x_1 + b_2x_2 + ... + b_nx_n$ and send $y$.
- To decrypt, solve for $b_i$ using the super increasing knapsack values.
- Flaws have been found in the algorithm.

Current Cryptographic Research

- For the past year a small team has been working on developing an encryption algorithm based on a non-computable problem.

Harder than Hard

- The RSA algorithm is secure because it is difficult to factor large numbers.
- Computers are getting faster and mathematicians are getting smarter.
- Factoring a number is $O(\sqrt{n})$.
- We want to develop an algorithm based on a much harder problem.
No Possible Solution

- The halting problem is the best known non-computable problem. You cannot write an algorithm that reads a program and its data and can always tell if the program comes to the end.
- There are many instances of non-computable problems that can be easily solved, but you cannot solve all of them.
- There are several known non-computable problems.

Post correspondence problem

- Given a set of domino-like tiles with symbols on the top and bottom, can a series of tiles be placed side by side so that the string of symbols on the top matches the string of symbols on the bottom.
- Assume you have an infinite number of each type of domino.

Solution to Example

Simple solution to this instance of the problem requiring seven dominos.

```
100 1 100 100 1 0 0
1 00 1 1 00 100 100
```

More Difficult Problem

- If you change the bottom number on the last tile so that it only contains one zero, the problem becomes much more difficult.
- This simple change has a solution, but it requires 75 dominos.

```
100 1 100 100 1 0 0
1 100 100 00
```
Wang tiles

- When using a set of square tiles with colored edges, can the tiles be arranged without rotation or reflection so that they tile a plane with adjacent tiles having edges of the same color?
- Assume you have an infinite supply of each tile

```
  R  B  Y  G
  B  G  R  Y
  Y  R  G  B
  G  B  Y  R
```

Periodic and Aperiodic

- The previous solution has a 2 by 4 block that can be repeated forever
- Some sets of tiles can tile an infinite plane without repeating

```
  R  B  Y  G
  B  G  R  Y
  Y  R  G  B
  G  B  Y  R
```

Tiling Solution

```
  T  T  T  T
  T  T  T  T
  T  T  T  T
  T  T  T  T
```

Aperiodic Tiling
Domino Snakes

- Using tiles similar to Wang Tiles, can a given set of tiles form a path from two given points so that all adjacent tile edges have matching colors?

Use in Encryption

- Like the knapsack encryption algorithm, we are looking for a way to transform the tiles from a simple problem to a nearly impossible problem
- We have been developing heuristics to identify sets of tiles that do or do not tile the plane or form a snake