Chapter 9
Lecture 9 - More encryption

Warp factor 9...

Large and hitherto uncatalogued life-form on the windsceen, Jim.
Tut 7, Q1: Hamming

**Question:** Calculate extra bits needed for encoding a 64 bit value, with two-bit error *detection*.

**Answer:** Want a hamming distance of 3, so

\[(n + 1)2^m \leq 2^n\]
\[64 + 1 \leq 2^r - r\]
\[R = 7\]

Tut 7, Q2: Diffie-Hellman

**Question:** Give a worked example of Diffie-Hellman

**Answer:** Choose \( p = 1637 \), and \( g = 331 \). Alice chooses a secret key \( a = 1433 \), and Ted chooses \( b = 977 \).

(a) Alice sends \( m_a = g^a \mod p = 893 \).
(b) Ted sends \( m_b = g^b \mod p = 749 \).
(c) Alice calculates \( m_b^a \mod p = 441 \).
(d) Ted calculates \( m_a^b \mod p = 441 \). Have a shared key.
(d) Bob has the difficult problem to calculate \( g^{ab} \mod p \).
Tut 7, Q3: DES

- Same sort of confusion and diffusion algorithm. One round of 16, performed for each 64 bits of the input.

- \( \langle a, b \rangle \) are 32-bit registers. The \( x \) operations are XOR. Note \( b' \) is one-way... \( K \) is part of the key.

---

Tut 7, Q3: Feistel F function

56-bit key \( K \) is split into 16 subkeys - each 48 bits. S-Box and P-Boxes.
Tut7 Q4: Mono-alphabetic cipher

Recursive programs may require large numbers of procedure calls and stack operations and many such recursive programs exhibit exponential time complexity due to the timespent on recalculating already computed subproblems as a result. Methods which transform a given recursive program to an iterative one have been intensively studied. We propose here a new framework for transforming programs by removing recursion. The framework includes a unified method of deriving logarithmic order programs by solving recurrences derived from the program sources. Our prototypes system, PARTS1, is an initial implementation of the framework. Automatically finding simpler closed form versions of a class of recursive programs, though in general the solution of recurrences is easier if the functions have only a single parameter, we show a practical technique for solving those with multiple parameters.

https://www-appn.comp.nus.edu.sg/~cs3235/mono.cgi

Tut7 Q5: Poly-alphabetic cipher

Answer: Running the ic.perl script using shifts of 1, 2, 3 yields:

Offset k = 1: 660 characters, 26 match Estimated IC: 0.0393
Offset k = 2: 659 characters, 21 match Estimated IC: 0.0318
Offset k = 3: 658 characters, 23 match Estimated IC: 0.0349
Offset k = 4: 657 characters, 42 match Estimated IC: 0.0639
Offset k = 5: 656 characters, 32 match Estimated IC: 0.0487

Note the sudden jump when the offset is 4, 8, 12 etc. This is fairly good evidence of a Vigenere cipher with a key length of 4. Note also the values... \( \frac{1}{26} = 0.0385 \).
Tut7 Q5: Poly-alphabetic cipher

BOBBY HURRIED DOWN THE ROAD IN THE DIRECTION OF THE CEDARS ALWAYS HURRYING. HE DESPERATELY TO RECALL WHAT HAD OCCURRED DURING THOSE BLACK HOURS LAST NIGHT AND THIS MORNING BEFORE HE HAD AWAKENED IN THE EMPTY HOUSE NEAR HIS GRANDFATHER'S HOME. ALL THAT REMAINED WERE HIS SENSATIONS OF TRAVELING IN A SWIFT VEHICLE. HIS IMPRESSION OF STANDING IN THE FOREST NEAR THE CEDAR'S HILLS. HIS GLIMPSE OF THE MASKED FIGURE WHICH HE HAD CALLED HIS CONSCIENCE THE ECHO IN HIS BRAIN OF DREAM LIKE VOICES SAYING TAKE OFF YOUR SHOES AND CARRY THE MINI YOUR HAND ALWAYS DO THAT IT'S THE ONLY SAFE WAY THESE FACTS THEN A LONE WERE CLEAR TO HIM. HE HAD WANDERED UNCONSCIOUS IN THE NEIGHBOURHOOD. HIS GRANDFATHER HAD BEEN STRANGELY MURDERED THE DETECTIVE WHO HAD MET HIM IN THE VILLAGE PRACTICALLY ACCUSED HIM OF THE MURDER AND HE COULDN'T REMEMBER...

Final lectures...

1. Thursday Oct 20 - Protocols

2. Thursday Oct 27 - Unix and NT

3. Thursday Nov 3 - Hari Raya and Hugh away - make up necessary

4. Thursday Nov 10 - final

What is the best solution to the missed lecture?
Uses of asymmetric encryption

1. Generating encrypted passwords with 1-way functions
2. Checking integrity by appending digital signature
3. Checking the authenticity of a message.
4. Encrypting timestamps with messages to prevent replay attacks.
5. Exchanging a key.

Asymmetric encryption

- Participants each have private and public keys
- Keys cannot be derived from each other
**Asymmetric encryption**

- \( K_{pub} \) is public key for Bob, \( K_{priv} \) is his private key.
- Only Bob can decrypt

CS3235 notes.

Page number: 410

**Asymmetric authentication**

- \( J_{priv} \) is private key for Alice, \( J_{pub} \) is her public key.
- Only Alice could have encrypted message

CS3235 notes.

Page number: 411
Authentication and encryption

* Making sure of digital signatures. Hash function...

RSA (Rivest, Shamir, Adelman)

This public key system relies on the difficult problem of trying to find the complete factorization of a large composite\(^\text{14}\) integer whose prime factors\(^\text{15}\) are not known.

\(^{14}\text{An integer larger than 1 is called composite if it has at least one divisor larger than 1.}\)

\(^{15}\text{The Fundamental Theorem of Arithmetic states that any integer } N \text{ (greater than 0) may be expressed uniquely as the product of prime numbers.}\)
RSA hacks

Two RSA-encrypted messages have been cracked:

✶ The inventors of RSA published a 129-digits (430 bits) RSA public key. In 1994, it was factored with 5000 MIPS-years of computing time.

✶ A year later, a 384-bit PGP key was cracked. It needed 1300 MIPS-years to factor the key in three months.

Note that these efforts each only cracked a single RSA key.

RSA coding algorithms

Below are outlined the four processes needed for RSA encryption:

1. Creating a public key
2. Creating a secret key
3. Encrypting messages
4. Decoding messages
To create public key $K_p$

1. Select two different large primes $P$ and $Q$.

2. Assign $x = (P - 1)(Q - 1)$. (Does this ring a bell?)

3. Choose $E$ relative prime to $x$. (This must satisfy condition for $K_s$ given later)

4. Assign $N = P \times Q$.

5. $K_p$ is $N$ concatenated with $E$.

CS3235 notes. Page number: 416

To create private (secret) key $K_s$

1. Choose $D$: $D \times E \mod x = 1$.
   
   (a) (i.e. multiplicative inverses)
   (b) another way: $DE = k(P - 1)(Q - 1) + 1$

2. $K_s$ is $N$ concatenated with $D$.

CS3235 notes. Page number: 417
To encode plain text $m$

1. Pretend $m$ is a number.
2. Calculate $c = m^E \mod N$.

To decode $c$ back to $m$

1. Calculate $m = c^D \mod N$.
2. ....WHY?....
...Why?...

\[
c^D \mod N = m^{ED} \mod N = m^{k(P-1)(Q-1)+1} \mod PQ = m\]

\* \(m^{P-1} \mod P = 1\), SO \((m^{(P-1)})^{k(Q-1)} \mod P = 1\)

\* \(m^{Q-1} \mod Q = 1\), and so (tutorial) \((m^{(P-1)})^{k(Q-1)} \mod PQ = 1\).

CS3235 notes.

---

**RSA code**

```perl
#!/usr/bin/perl -sp
X+d*lMLa^*lN%0\]dsXx++lMlN/dsM0<j\]dsj
$=unpack('H*',$);$_=`echo 16dio\U$k"SK$/SM$n\EsN0p[lN*1
1K[d2%Sa2/d0$^Ixp"|dc`;s/\W//g;$=pack('H*','(\.)*$/$)
```

and then

\* `echo "squeamish ossifrage" | .rsa.perl -k=10001 -n=1967cb529 > msg.rsa`

\* `./rsa.perl -d -k=ac363601 -n=1967cb529 < msg.rsa`

CS3235 notes.

Page number: 420
New topic: Protocols...

Lots of opportunity to \textit{snoop, modify, DoS}, etc etc

Routing and packets...

Lots of opportunity to \textit{modify} routing information, \textit{spoo}f etc etc
Kerberos/Cerberus

✔ Network authentication protocol.

✔ Strong authentication for client/server applications using public key cryptography.

✔ Kerberos is freely available in source form

✔ Kerberos is also available in commercial products.

✔ Client can prove its identity to a server (and vice versa) across an insecure network connection.
After a client and server have used Kerberos to prove their identity, they can also encrypt all of their communications to assure privacy and data integrity as they go about their business.

Must have a Key Distribution Center (KDC)

Kerberos uses Needham-Schroeder protocol.
Kerberos

When a client first authenticates to Kerberos, she:

1. Talks to KDC, to get a *Ticket Granting Ticket*
2. Uses that to talk to the *Ticket Granting Service*
3. Uses the *ticket*, to interact with the *server*.

This way a user doesn’t have to reenter passwords every time they wish to connect to a Kerberized service. If the Ticket Granting Ticket is compromised, an attacker can only masquerade as a user until the ticket expires.

---

Kerberos protocol

- Two sorts of credentials: *tickets* and *authenticators*.

- A *ticket* $T_{c,s}$ contains the client’s name and network address, the server’s name, a timestamp and a session key. This is encrypted with the server’s secret key (so that the client is unable to modify it).

- An *authenticator* $A_{c,s}$ contains the client’s name, a timestamp and an optional extra session key. This is encrypted with the session key shared between the client and the server.
Kerberos protocol

✔ A key $K_{x,y}$ is a session key shared by both $x$ and $y$.

✔ When we encrypt a message $M$ using the key $K_{x,y}$ we write it as $\{M\}K_{x,y}$.

Alice wants session key for communication with Bob:

✶ Alice sends message to Ted containing her identity, Ted's TGS identity, and one-time value ($n$): $\{a, tgs, n\}$.

✶ Ted responds with a key encrypted with Alice's secret key (which Ted knows), and a ticket encrypted with the TGS secret key: $\{K_{a,tgs, n}K_a \{T_{a,tgs}K_{tgs}\}K_{tgs}$.

Alice now has ticket and session key: $\{T_{a,tgs}K_{tgs}, K_{a,tgs}$

✶ Alice can prove her identity to the TGS, as she has session key $K_{a,tgs}$, and Ticket Granting Ticket: $\{T_{a,tgs}K_{tgs\}$.
Kerberos protocol

Later, Alice can ask the TGS for a specific service ticket:

acious wants a ticket for a specific service (say with Bob), she sends an authenticator along with the Ticket Granting Ticket to the TGS:

\[ \{A_{a,b}\} K_{a,tgs} \{T_{a,tgs}\} K_{tgs}, b, n. \]

The TGS responds with a suitable key and a ticket:

\[ \{K_{a,b}, n\} K_{a,tgs} \{T_{a,b}\} K_{b}. \]

Alice can now use an authenticator and ticket directly with Bob:

\[ \{A_{a,b}\} K_{a,b} \{T_{a,b}\} K_{b}. \]

Weaknesses

**Host security:** Kerberos makes no provisions for host security; it assumes that it is running on trusted hosts with an untrusted network.

**KDC compromises:** Kerberos uses a principal’s password (encryption key) as the fundamental proof of identity.

**Salt:** This is an additional input to the one-way hash algorithm.
Voting protocols

A voting protocol is one in which

- independent systems vote in a kind of election, and
- afterwards we can check that the vote was correct.
- Each voter is only allowed a single vote, and
- the system should be corruption-proof.

Example with Alice, Bob and Charles (!), who vote and then encrypt and sign a series of messages using public-key encryption. For example, if Alice votes $v_A$, then she will broadcast to all other voters the message

$$R_A(R_B(R_C(E_A(E_B(E_C(v_A)))))))$$

where $R_A$ is a random encoding function which adds a random string to a message before encrypting it with $A$’s public key, and $E_A$ is public key encryption with $A$’s public key.
Voting protocols

✔ Each voter then signs the message and decrypts one level of the encryption.

✔ At the end of the protocol, each voter has a complete signed audit trail and is ensured of the validity of the vote.

Tossing a coin

✔ Alice and Bob want to toss a coin

✔ Alice calculates two primes $p, q$ and calculates $N = pq$, sends $N$ to Bob. $N = 35 = 5 \times 7$

✔ If Bob can factorize the number, then Bob wins a coin toss.

✔ Bob selects random $x$, and sends $x^2 \mod N = y$ to Alice. $y = 31^2 \mod 35 = 16$
Tossing a coin

Alice calculates the four square roots of 16:

* \(4^2 \text{ mbox } 35 = 16\)
* \(31^2 \text{ mbox } 35 = 16\)
* \(24^2 \text{ mbox } 35 = 16\)
* \(11^2 \text{ mbox } 35 = 16\)

This is easy for Alice, as she knows the prime factors of \(N\). She then sends one of these back to Bob.

CS3235 notes. Page number: 438

Tossing a coin

✔ If Bob receives \(x\) or \(-x\), then he learns nothing, but

✔ if Bob receives either of the other values, he can add this to \(x\), and then find the GCD of the result with \(N\):

\[
\text{GCD}(24 + 31, 35) = \text{GCD}(55, 35) = 5
\]

✔ Alice is unable to tell she has divulged the factor.
Oblivious transfer

✔ In an oblivious transfer, randomness is used to convince participants of the fairness of some transaction.

✔ In a coin-tossing example, Alice knows the prime factors of a large number, and if Bob can factorize the number, then Bob wins a coin toss.

✔ A protocol allows Alice to either divulge one of the prime factors to Bob, or not, with equal probability.

✔ Alice is unable to tell if she has divulged the factor, and so the coin toss is fair.

Contract signing

✔ Signing contracts can be difficult.

✔ If one party signs the contract, the other may not. We have one party bound by the contract, and the other not.

✔ In addition, both may sign, and then one may say “I didn’t sign any contract!” afterwards.
Contract signing

Oblivious transfer used for contract-signing where

- Up to a certain point neither party is bound
- After that point both parties are bound
- Either party can prove that the other party signed

Alice and Bob exchange signed messages, agreeing to be bound by a contract with ever-increasing probability.

CS3235 notes. Page number: 442

✔️ In the event of early termination of the contract, either party can take the messages they have to an adjudicator, who chooses a random probability value (42% say) before looking at the messages.

✔️ If both messages are over 42% then both parties are bound.

✔️ If less then both parties are free.

CS3235 notes. Page number: 443