Lecture 04-27-09: Encryption

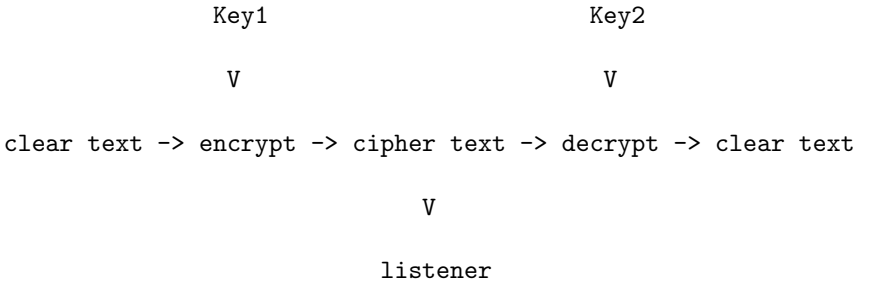
Matt Carroll

Discussion 2 in 118 Barrows for the rest of semester Exam 2 grades are up. Scores are still pretty high.
Recommended reading:

- "The Codebreakers". History of cryptography starting with romans/greeks through WW2.
- Whitfield, Diffie and Martin Hellman, "Privacy and Authentication: An Introduction to Cryptography", Proc. IEEE, 67, 3, March, 1979, pp. 397-427.

These are not assigned reading
A lot of papers are going online as PDFs soon.

Encryption
The main idea is that people are welcome to look, but they can't understand. Store and transmit information in an encoded form.
Encryption/cryptography is not new - has been used since the time of the Romans ("Caesar cypher").
The following diagram demonstrates the basic flow of information through an encyrption scheme:



The basic mechanism:

- Begin with the information to be protected. This is called the *clear text*.
- Encrypt the clear text so that it's unintelligible to a listener. This is called the *cipher text*. Encryption is controlled by a secret password or number, called the *encryption key*.
- To make sense of the cypher text, it must be decrypted back into the clear text. This is done with an algorithm that uses the other secret password or number, called the *decryption key*.

- The cipher text is exposed and incomprihensible to anybody who doesn't know how to decrypt it.

In order for an encryption scheme to work, three conditions must be met:

1. The encryption function can't easily be inverted (you can't get from the cipher text to the clear text without the decryption key).
2. The encryption must be done in some safe place so the clear text can't be stolen. If you're sitting out doing it on the beach...that's bad. Recent example: British Minister of Defence was photographed in public, with a highly classified document under his arm, and the text visible, resulting in a scandal and his resignation.
3. The keys must be secret. Usually, if you know one of the keys, you can compute the other. The algorithms are publicly known, but if either of the keys gets out, a listener can decrypt the cipher text.

Types of Cryptographic Systems
Substitution: A function $f(x)$ maps each letter of the plaintext into $f(x)$. This function must be 1 to 1, or 1 to many. The Caesar Cipher is a substitution scheme, where $f(x) = x + 1$ (the simplest substitution scheme).
For example:

```
The quick brown fox jumps over the lazy dog
||||||
vvvvvv...
uifarv...
```

To break this scheme, use frequency analysis of letters, doubles, triples, groups (etc.) Unless you're trying to do something weird, the top 5 or 10 letters are easy to identify by computing the frequencies, then doing the substitution. The most frequent symbol is probably "e", then the next most frequent is "t", and so on. After you've filled in half the letters, you can infer the rest if you know the language of the clear text.
Using a 1 to many function can disguise the frequency of symbols, making the scheme harder to break using the above method.

Transposition Permute (or transpose) the input in blocks to obtain the output. The following diagram demonstrates a simple transposition scheme:

```
| t | h | e |   | q |
| u | i | c | k |   |
| b | r | o | w | n |  ==>   tub_jhirfu...
|   | f | o | x |   |
| j | u | m | p | e |
```

Here, the input is placed in a 5 by 5 matrix, and the cipher text is produced by ordering the letters as they appear when traversing the matrix vertically, column by column, and from left to right.
If you know a transposition cipher is being used, it can be broken by looking for permutations that rejoin commonly used letter pairs, such as "th". This is not trivial to figure out.
Transpotiion ciphers were the state of the art circa the 17^{th} century.

Polyalphabetic ciphers These are an extension of substitution ciphers. The clear text is encoded with $f(i, x)$, a function of i where i is the sequence number of the letter in the text. Typically the function is periodic in i.
Example:

```
clear text: t h e   q u i c k   b r o w n   f o x
         i: 1 2 3 1 2 3 1 2 3 1 2 3 1 2 3 1 2 3 1
   f(i, x): u j h a s . . .
```

Question: Why make it periodic?
Answer: There's a tradeoff between the complexity of code and ease of encoding/decoding, possibly losing your mapping, things like that. Historically these were used prior to computers. A person had to be able to do the encryption on paper in a finite amount of time. Also, there wasn't any literature about codebreaking as there is today.
Usually, schemes get broken by stealing the code or torturing somebody until they tell you, rather than actually breaking the code technically. Example: Enigma machine used by Germany in WW2.

To break a polyalphabetic cipher, look for repeated strings, and count the number of symbols between them. Least common denominator between them is usually the period. Or, you can look at the frequency of letters K apart, until they look like they should, then K is the period.
Once you know the period, you can solve each of the N ciphers as a substitution cipher.
By the 20^{th} century, there were coding machines to do polyalphabetic ciphers. They had rotating wheels with a relatively prime number of cogs. The code was a product of a path through the wheels.

Running key cipher Use a long text as the key, but not random, like using a book or something. For example, suppose you want to encrypt The DaVinci Code. Use some other book as the key. Take the key as numbers 1 through 26, and do a mod 26 addition to the key text to get the cipher text. To decrypt, do a mod 26 subtraction of the DaVinci Code from the cipher text.

Thus far, we've discussed encryption methods that do things to individual letters. Now we'll talk about *codes*, encryption methods that use groups of letters

(words) as input. A code book is used to map words to output code words. This could even be used with whole phrases as inputs.
For example, this was a common method of transmitting messages accross the Atlantic during the early 1900s. Communication was expensive, so commercial codebooks were used to shorten the message to transmit. Another example of this type is substituting page, line, and word numbers for words in some book. These codes are very hard to break. The approach suggested is to try frequency analysis. They can be broken via the plaintext method: if you get a hold of a plaintext, a cipher text, and the codebook, you can determine the encryption method. But this is another example of a code being broken via some means other than cryptanalysis.

There are other approaches that may not break the code per se, but the security of the communication can be considered compromised. An example of this is traffic analysis: imagine in a war situation, one side observes a large quantity of encoded traffic, and the next day an artillery strike occurs. Now they can expect an artillery strike when they observe a lot of traffic.
For that particular security vulnerability, the solution is to generate a lot of false traffic, or play back old messages.
In most systems, key distribution is actually the biggest weakness. If you change the key, somehow the new key has to be distributed to everyone who needs it - if you transmit it, it could be vulnerable to decryption. If you send it via human courier, that person could get captured, etc.

An encryption scheme where the value of the effort required to break the code exceeds the value of the reward obtained by breaking it is considered secure. This is actually the definition of secure that is used.
The only truly perfect method of encryption is to use a random key that is as long as the message. The key must be truly random, not an English text for example (due to the fact that English texts have observable word frequencies). Getting truly random sequences is quite difficult. An encryption scheme that is complicated is not necessarily hard to break.

Error control is a major issue with encryption schemes. If some bits of your message are corrupted, it could be extremely difficult or impossible to recover the message. Standard error correcting codes can be used to mitigate the effects of this problem.

Key distribution
As mentioned above, key distribution is a major weakness of many systems. Here's a method for secure key distribution: there's a key server in the middle that knows everybody's keys, and is responsible for generating new keys. Messages are numbered with a message ID. The goal is allow two users to communicate directly, and securely, without going through the key server.
Let KS be the key server, and let KX be the key used to communicate between

user X and the key server KS. There are users A and B, and let ** denote encryption. ID is the message ID, and KAB is the new key that users A and B will use for secure communication without going through the key server.

1. A sends the following message to KS:

   ```
   {A,(ID,B)**KA}
   ```

   The message consists of A, and the pair ID, B encrypted with KA. This is A's request to the key server for a key that can be used to communicate with B.

2. KS sends the following message to A:

   ```
   {(ID,KAB,(KAB,A)**KB)**KA}
   ```

   The message consists of the ID, KAB (which is a new key generated by the key server), and the pair KAB,A encrypted with KB. The message is encrypted with KA.

3. Lastly, A sends the following message to B

   ```
   {(KAB,A)**KB}
   ```

   This message is just the new key KAB, and A, encrypted with KB. Now both A and B have securely received KAB, and they can now use KAB to communicate with each other.

Encryption Standards

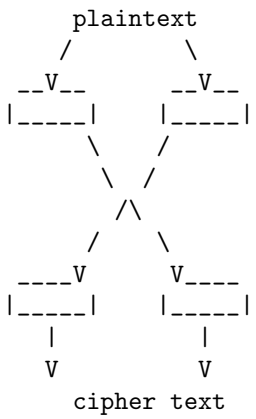
- The Federal Government uses the *DES* (data encryption standard). This can be implemented in hardware, and is relatively safe. It's a block cipher, encrypting 64 bits at a time using a 56 bit key. Developers IBM wanted to use a longer key, but the govt. didn't want it to be too secure (they wanted to be able to break the code if necessary). These chips can encrypt/decrypt megabits per second. The NSA doesn't consider this system secure enough, they have a better alternative called CLIPPER. But for most purposes, DES is good enough.

  The following diagram depicts the flow of information through the DES system:

  Normal:

  ```
  plaintext
      |
    __V__
   |_DES_|
      |
      V
  cipher text
  ```

Two-level encryption:



Sufficient security is obtained by two level encryp-tion in pairs.

Export control (ITAR (international traffic in arms regulations)) on DES products. Controls what you can import or export from the US. Example, if you own a tank, you need a permit to export it. Government decided that encryption products are covered unter ITAR - considered weapons. If you come up with a fancy encryption algorithm and email it to your friend in China, you can be prosecuted.

- *PGP* - pretty good privacy. This encryption system is based on DES. The govt. tried to prosecute the inventor of PGP, for the above mentioned reasons. Eventually they gave up the case.

  There are two keys, which are inverses of each other. Whatever is encoded with one can be decoded with the other. ("private key", "public key") Two keys are not derivable from each other. Each user keeps one key secret, publicizes the other. Can't derive private key from public key. Public keys are made available to everyone, in a phone book for example.

- Specific scheme for public key encryption (pages 471-472, chap 14, of Silberschatz and Galvin):

  Encode: $E(m) = (m^e) mod n = C$. e and n are public; d is private.

  Decode: $D(C) = (C^d) mod n$

  where: m is message, e, d, are between 0 and $n-1$; e, d, n positive integers. Must derive e, d, and n such that the above decode is inverse of encode. To do this:

  Let $n = pq$ (p, q large primes).

  d is large integer relatively prime to $(p-1)*(q-1)$

  (i.e. $GCD[d, (p-1)*(q-1)] == 1$)

  d is chosen such that $(e*d) mod((p-1)*(q-1)) == 1$

It Can Be Shown that this makes E and D inverses. This is safe because although n is known, p and q are not known, and so e cannot be derived. (factoring is known to be hard.) The proof requires some number theory and won't be detailed here.

Safe mail: Use public key of destination user to encrypt mail. Anybody can encrypt mail for this user and be certain that only the user will be able to decipher it. Encryption only saves the message. It doesn't prevent people from breaking into your computer. It's a nice scheme because the user only has to remember one key, and all senders can use the same key. However, how does receiver know for sure who it's getting mail from?

Digital signatures: can also use public keys to certify identity: To certify your identity, use your private key to encrypt a text message, e.g. "I agree to pay Mary Wallace $100 per year for the duration of life." You can give the encrypted message to anybody, and they can certify that it came from you by seeing if it decrypts with your public key. Anything that decrypts into readable text with your public key must have come from you! This can be made legally binding as a form of electronic signature.
Note that only encrypting with your private key permits the mail or message to be read by anyone. If you encrypt with your private key, and then someone else's public key, it can only be read by intended recipient.
One public key method believed to work: Publish a large composite number (public key). Private key is factors of the number. Factors hard to obtain.
Encryption appears to be a great way to thwart listeners. It doesn't help with Trojan Horses, though.
One Way Encryption - use to encrypt password file. Don't have to be able to decrypt it - just compare encryption of submit- ted password with stored one. Can't deduce what needs to be submitted. (I.e. encryption algorithm should not be invertable.). This is used by Unix in the password file.
General problem: how do we know that an encryption mechanism is safe? It's extremely hard to prove. For examle, for a while the knapsack algorithm was thought to be hard enough...then someone found a polynomial time algorithm to do it. A good way to know if your scheme is safe: offer a reward for breaking it. A hot topic for research: theorists are trying to find provably hard problems, and use them for proving safety of encryption.

CLIPPER Chip - came out in 1990s. After DES became dubiously strong, government came out with this. It's a very secure system, but the government holds the key, so they can read your mail if you use it.
CLIPPER Chip:

- contains 64-bit block encryption (algorithm classified)
- Uses 80 bit keys.

- Uses 32 rounds of scrambling (compared to 16 for DES)
- F - 80-bit key used by all Clipper chips
- N - 30-bit serial number (per chip)
- U - 80-bit secret decryption key for this chip only.

Secure conversation occurs this way:

- Session key K is negotiated (somehow).
- E(M;K) is encrypted message stream. (M = Message)
- E(E(K;U), N; F) is a "law enforcement block". With F, we can get E(K;U),N. From N, (the serial number), we can get U (held by federal agencies), and then can get K. From K, we can decrypt messages.
- Key U is xor of U1 and U2. U1 and U2 held by different federal agencies. Can get both U1 and U2 only with court ordered wiretap. Govt. ordered defense contractors to use this, but we don't know if anybody else is using it voluntarily.