Classic Cryptography: From Caesar to the Hot Line

WINLAB
WIRELESS INFORMATION NETWORK LABORATORY

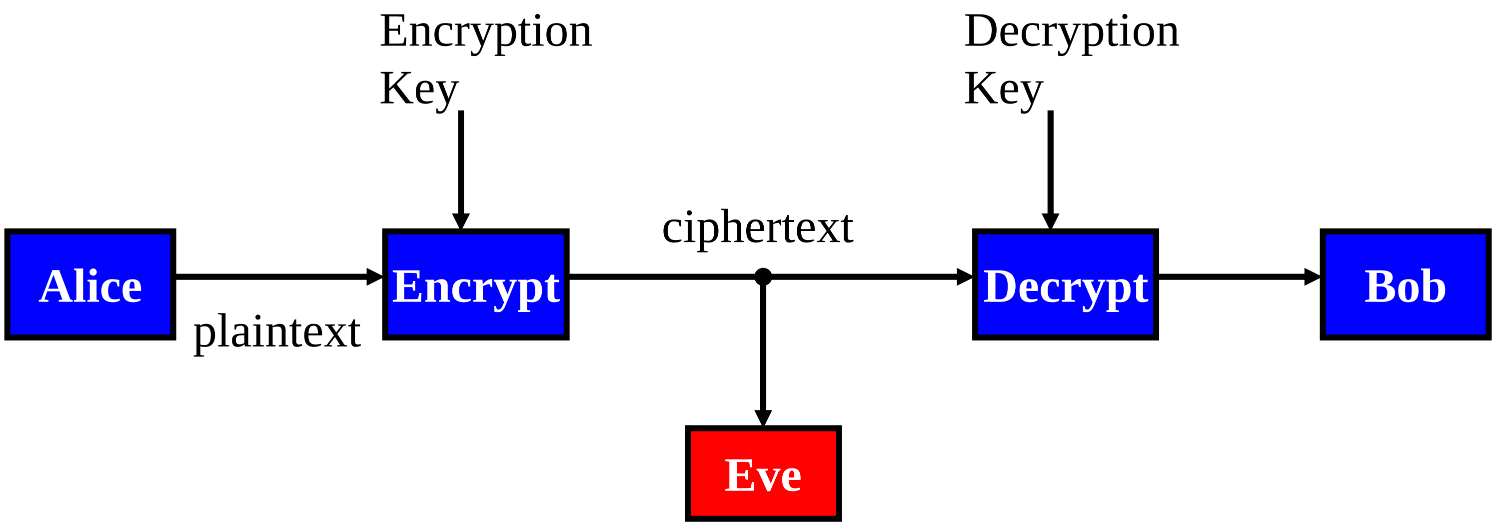
Overview of the Lecture

- Overview of Cryptography and Security
- Classical Cryptography:
  - Caesar/Shift Cipher
  - Affine Ciphter
  - Vigenere
- One-time Pads
- LFSR

What is Security?

- Security is a hard thing to explicitly define.
- Basically, its an assurance that some entity or thing is protected from possible harm
- In the context of:
  - **Information:** Assurance that information is not learned by entities not intended to learn this information
  - **Computer:** Assurance that the computer system and its files are not harmed by some form of outside attack
  - **Network:** Assurance that entities involved in a communication, as well as the information being shared, are kept “safe” or “hidden”
- This class will deal with all of these, and more!

The Basic Secure Communication Scenario



- Alice communicates with Bob via the channel.
- Eve is **evil** and **eavesdrops**. Her goals:
  - Read the message
  - Determine the keys
  - Corrupt the message to Bob gets something different than what Alice sent
  - Pretend to be Alice and fool Bob

The Basic Categories of Attacks

- There are four main strategies that Eve might employ to achieve her evil plans:
  - **Ciphertext only:** Eve only has a copy of the ciphertext
  - **Known Plaintext:** Eve has a copy of the ciphertext and the corresponding plaintext. Example: Alice always starts her messages the same way.
  - **Chosen Plaintext:** Eve somehow gains access to the encryption device. She can't open it up to get the key, but she can input whatever she wants.
  - **Chosen Ciphertext:** Eve somehow gains access to the decryption device. She can't open it up to get the key, but she can input whatever she wants.
- Most types of information security attacks can be loosely categorized as one of these attacks.

Shift Cipher

- Start with the plaintext alphabet. For example, it may be Z_{26} if we are using A, B, …, Z. Map these to numbers

A = 0 B = 1 C = 2 … Y = 24 Z = 25

- Plaintext is mapped to a numerical representation “x”.
- The key, k , is a letter/number in Z26.
- Encryption yields the ciphertext:

$$y = x + k \pmod{26}$$

- Decryption is

$$x = y - k \pmod{26}$$

- Caesar used k=3.

How to Attack the Shift Cipher?

- **Known Plaintext:** You know x and you know y, so you calculate the key easily by:

$$k = y - x \pmod{26}$$

- **Chosen plaintext:** You get to choose the plaintext x. For simplicity, take x = ‘a’ = 0. The ciphertext is

$$y = k \pmod{26}$$

- **Chosen Ciphertext:** You get to choose the ciphertext y. Choose y=‘A’, then plaintext is $x = -k \pmod{26}$.

$$x = -k \pmod{26}$$

- What about Known Ciphertext? This is the toughest one. We have the least information.

Ciphertext Only on Shift Cipher

- If all Eve gets is the ciphertext, she may try one of two strategies:
  - She could try all choices for k. There are only 26 and most likely (if message is long enough) only one key will produce something intelligent.
  - She could do a frequency counting:
    - *Use knowledge of the underlying language. For example, 'e' occurs the most often in English.*
    - *Whatever letter occurs the most in the ciphertext probably corresponds to the plaintext 'e'. (If you have a long enough message).*
    - *The key is probably the value needed to shift 'e' to the most frequent ciphertext letter.*
    - *If the most frequent fails, try the next most frequent…*

Affine Cipher

- The affine cipher involves a key (α,β) and maps the plaintext x to the ciphertext y via

$$y = \alpha x + \beta \pmod{26}$$

- **Example:** y= 9x+2 (mod 26)
- Decryption solves for x. This would normally be simple algebra

$$x = \frac{1}{\alpha}(y - \beta) \pmod{26}$$

- What does $(1/\alpha)$ mean?
- It is the inverse of α (mod 26)… ok, so what does that mean?

Inverses (mod n)

- When you think of a number (a^{-1}) in normal algebra, you think of division.
- What is really going on is that you are finding another number b such that ab=1.
- So, when we write a^{-1} (mod n) we really mean the number b such that ab=1 (mod n).
- How do we find this b?
  - We will see a fast way to do it for large n later… for now, just make a table!
  - Example, suppose a=7, n=26

    7*1 = 7 mod 26 7*4 = 28 = 2 mod 26

    7*2 = 14 mod 26 7*5 = 9 mod 26

    7*3 = 21 mod 26 7*6 = 16 mod 26 … and so on…

  Until you find a number b such that 7b=1 mod 26. (b= 15)
- Final Comment: You only have inverses when gcd(a,n) = 1

More on the Affine Cipher

- We need gcd(α,26)=1 in order to have an invertible function (see pg 15 for what can happen when you don't have this!)
- There are 12 choices for α since there are 12 numbers with gcd(α,26) = 1. (Check this!)
- We also need to choose β , and there are 26 possibilities.
- In total, we have 12*26 = 312 choices for the key k=(α,β)

Attacks on Affine Cipher

- **Ciphertext Only:** Just try all 312 possible keys.
- **Known Plaintext:** If you have two pieces of plaintext, you may set up a system of equations

$$y_1 = \alpha x_1 + \beta \pmod{26}$$
$$y_2 = \alpha x_2 + \beta \pmod{26}$$

Solving is just algebra!

- **Chosen Plaintext:** Choose ‘ab’ as the plaintext. The first character of the ciphertext will be β, while the second will be $\alpha+\beta$.
- **Chosen Ciphertext:** Similar to Chosen Plaintext.

Vigenere Cipher

- The Vigenere cipher is an extension of the shift cipher.
- Basically, your key is a secret word of unknown length. Encryption proceeds as:
  - Line your key up with your plaintext, repeating it as necessary.
  - Perform a letter-by-letter shift using the letter of the key beneath it.
- **Example:**

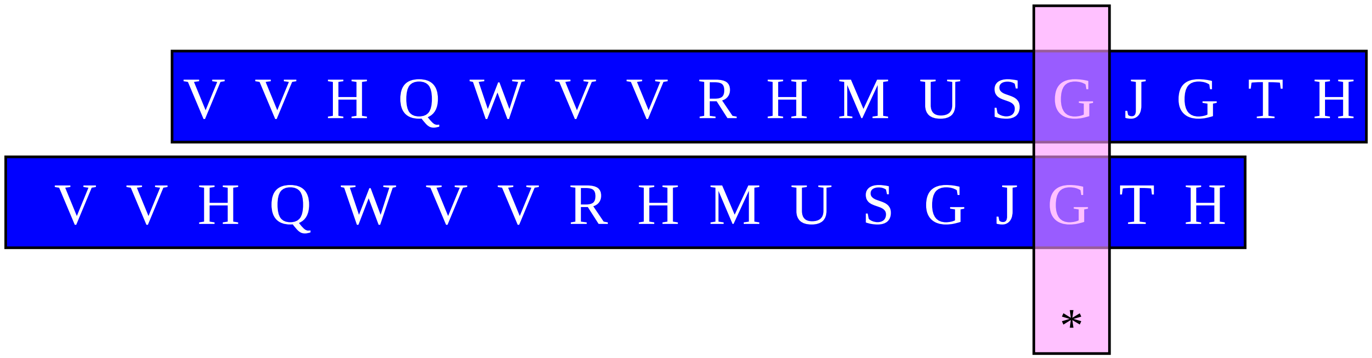
	H	e	r	e	i	s	h	o	w	i	t
Shift	21	4	2	19	14	17	21	4	2	19	14
	C	I	T	X	W	J	C	S	Y	B	H

How to Break the Vigenere

- There are two tasks to breaking Vigenere:
  - Find the length of the key
  - Find the key itself.
- If we know the length of the key (L), finding the key itself is not that hard:
  - All we need is to do to find the first letter of the key is grab every Lth letter from the ciphertext and perform frequency analysis
  - Then, to get the 2^{nd} letter of the key, we grab the 2^{nd}, L+2, 2L+2, … letter from the ciphertext
  - And so on…
- The trick lies in finding the length of the key!

Finding the Key Length in Vigenere

- Write the ciphertext on two strips of paper. Put one on top of the other, but displaced by a certain amount of places.



- Mark a * each time a letter and the one below it are the same.
- Count the total number of coincidences for different displacements
- The displacement with the most coincidences is the most likely key size.

Finding the Key Length Explained, pg. 1

- So why does this work?
- Suppose we write down the vector of frequencies for English letters:

$$\mathbf{A}_0 = (.082, .015, .028, \ldots, .020, .001)$$

- We may shift $\mathbf{A}_0$ by i spaces to the right to get $\mathbf{A}_i$, which corresponds to the probabilities we would get if we applied a shift cipher where we shifted by i letters
- Look at the dot products $< \mathbf{A}_i , \mathbf{A}_j >$, which depend only on $|i-j|$ (Why?)
- The maximum value is when $i-j = 0$.
  - This just says that a vector is most similar to itself.

Finding the Key Length Explained, pg. 2

- Now take a strip of ciphertext, and look at a random letter.
- This corresponds to a shift of the corresponding plaintext letter by an amount i, corresponding to an element of the key.
- Now, place a second strip below, and displace it. The letter below the first will correspond to a random letter of English shifted by j.
- What is the probability they are both “A”? This is $A_i(0) * A_j(0)$
- Similarly, the probability they are both “B” is $A_i(1) * A_j(1)$, and so on…
- The total probability that these letters are the same is $< \mathbf{A}_i , \mathbf{A}_j >$.
- When i=j, we have the maximum.
- This happens when the letters lying one above the other have been shifted by the same amount!

Binary and ASCII

- Generally, our data is not simply letters, but may more generally be characters, or they may be binary data generated by a program
- Binary numbers are a way to represent numbers base 2
- Example:

$$110101 = 2^5 + 2^4 + 2^2 + 1$$

- ASCII is the America Standard Code for Information Interchange:
  - Each character is represented using 7 bits
  - Total of 128 possibilities
  - Often an extra bit is used for parity checking, or used for extended characters

The One-Time Pad

- The one-time pad is an "unbreakable" cryptosystem developed by Vernam and Mauborgne in 1918.
- The **plaintext** message is represented using a sequence of bits
- The **key** is a random sequence of 0's and 1's as long as the message.
- The **ciphertext** is the XOR of the plaintext with the key.
- Example:

$$\begin{array}{ll} \text{(message)} & \ 00101001 \\ \text{(key)} & \oplus\ 10101100 \\ \hline \text{(ciphertxt)} & \ 10000101 \end{array}$$

- How to decrypt? Just XOR again with the key!

Vernam-style Ciphers

- The one-time pad belongs to a more general family of stream ciphers known that are often referred to as Vernam-style ciphers
- In a Vernam cipher
  - The plaintext x is a sequence of bits
  - The key sequence k is a sequence of bits
  - The ciphertext y is generated by

$$y = x \oplus k$$

- The key sequence is typically generated using a (cryptographic) pseudo-random number generator
- We will see many choices for generating k later, for now let us look at a popular (but weak) method

Linear Feedback Shift Registers

- Linear feedback shift registers (LFSRs) are a fast method for generating pseudo-random bits.
- Output bits depend on previous output bits using a linear recurrence.
- The general linear recurrence is:

$$x_{n+m} = c_0 x_n + c_1 x_{n+1} + \cdots + c_{m-1} x_{n+m-1} \pmod 2$$

where the initial values are

$$x_1, x_2, \cdots, x_m$$

- Why would we want to do this?
  - Its fast!
  - A small key (coefficients and/or initial values) can generate a key sequence with a large periodicity.

LFSR, the BAD!!!

- Why shouldn't we use LFSR? Answer: WEAK security
- LFSR succumbs easily to a known plaintext attack:
  - A few bits of plaintext and the corresponding ciphertext and we can solve for the recurrence relationship and generate all future bits in the key sequence.
- How to do this evil deed?

1. First, get the corresponding key sequence. (How?)
2. We don't know the length of the coefficient vector, so start with m=2. Set up system of linear equations.

   Solve linear equations for c-vector and then test to see if this generates the key sequence.
3. If not, try m =3 and so on…

Setting up the Equations to Attack LFSR

- This is best shown with an example.
- Example: Suppose we have the key sequence (011010111100)

m=2: Recurrence is $$x_{n+2} = c_0 x_n + c_1 x_{n+1}$$

Which yields the system of equations:

$$1 = c_0 \cdot 0 + c_1 \cdot 1$$
$$0 = c_0 \cdot 1 + c_1 \cdot 1$$
$$\Rightarrow \text{Solve to get}: c_0 = 1, c_1 = 1.$$

Try this out. Note that

$$x_{n+6} \neq x_4 + x_5$$

We must try m=3!

Setting up the Equations to Attack LFSR, pg 2

m=3: The recurrence is: $$x_{n+3} = c_0 x_n + c_1 x_{n+1} + c_2 x_{n+2}$$

The system of equations we get is:

$$\begin{pmatrix} 0 & 1 & 1 \\ 1 & 1 & 0 \\ 1 & 0 & 1 \end{pmatrix} \begin{pmatrix} c_0 \\ c_1 \\ c_2 \end{pmatrix} = \begin{pmatrix} 0 \\ 1 \\ 0 \end{pmatrix}$$

Do you see any problems with this?

There are no solutions!!!

Setting up the Equations to Attack LFSR, pg 3

m=4: The recurrence is: $x_{n+4} = c_0 x_n + c_1 x_{n+1} + c_2 x_{n+2} + c_3 x_{n+3}$

The system of equations we get is:

$$\begin{pmatrix} 0 & 1 & 1 & 0 \\ 1 & 1 & 0 & 1 \\ 1 & 0 & 1 & 0 \\ 0 & 1 & 0 & 1 \end{pmatrix} \begin{pmatrix} c_0 \\ c_1 \\ c_2 \\ c_3 \end{pmatrix} = \begin{pmatrix} 1 \\ 0 \\ 1 \\ 1 \end{pmatrix}$$

Solve this to get $c_0=1$, $c_1=1$, $c_2=0$, $c_3=0$.

The resulting recurrence is:

$$x_{n+4} = x_n + x_{n+1}$$

Checking this, we see that it works.