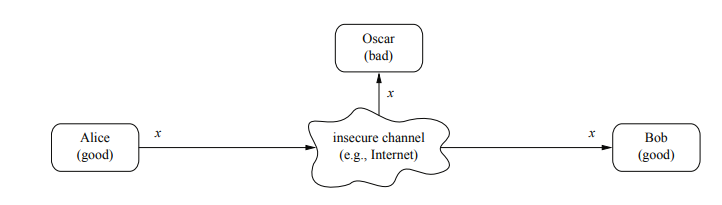
**Lecture 3: Symmetric Encryptions**

**Introduction**

Symmetric cryptographic schemes are also referred to as symmetric-key, secret-key, and single-key schemes or algorithms. Symmetric cryptography is best introduced with an easy-to-understand problem: There are two users, Alice and Bob, who want to communicate over an insecure channel Figure 1.

The term channel might sound a bit abstract but it is just a general term for the communication link: This can be the Internet, a stretch of air in the case of mobile phones or wireless LAN communication, or any other communication media you can think of. The actual problem starts with the bad guy, Oscar, who has access to the channel, for instance, by hacking into an Internet router or by listening to the radio signals of a Wi-Fi communication. This type of unauthorized listening is called **eavesdropping.**

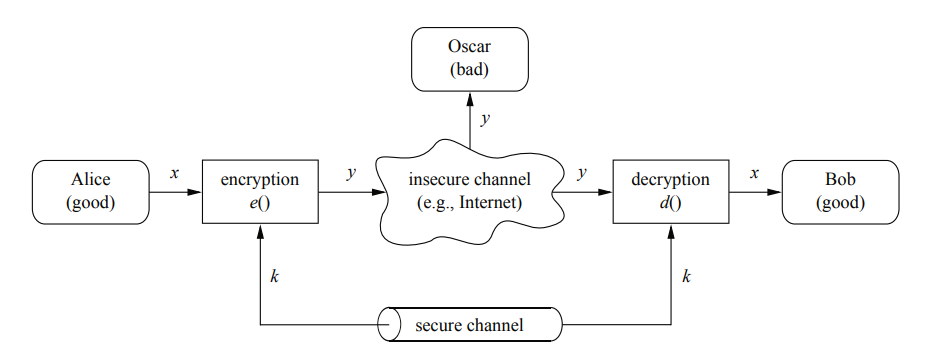
Obviously, there are many situations in which Alice and Bob would prefer to communicate without Oscar listening. For instance, if Alice and Bob represent two offices of a car manufacturer, and they are transmitting documents containing the business strategy for the introduction of new car models in the next few years, these documents should not get into the hands of their competitors, or of foreign intelligence agencies for that matter.



**Figure 1: Communication over an Insecure Channel**

**SYMMETRIC CIPHER MODEL**

In this situation, symmetric cryptography offers a powerful solution: Alice encrypts her message ***X*** using a symmetric algorithm, yielding the ciphertext ***Y***. Bob receives the ciphertext and decrypts the message. Decryption is, thus, the inverse process of encryption Figure 2. What is the advantage? If we have a strong encryption algorithm, the ciphertext will look like random bits to Oscar and will contain no information whatsoever that is useful to him.



**Figure 2: Symmetric-key cryptosystem**

A symmetric encryption scheme has five ingredients as indicated in Figure 2. This includes;

**Plaintext:** This is the original intelligible message or data that is fed into the algorithm as input. ***X*** is the plaintext.

**Encryption algorithm:** The encryption algorithm performs various substitutions and transformations on the plaintext. ***E () is the encryption algorithm.***

**Secret key:** The secret key is also input to the encryption algorithm. The key is a value independent of the plaintext and of the algorithm. The algorithm will produce a different output depending on the specific key being used at the time. The exact substitutions and transformations performed by the algorithm depend on the key. ***K is the secret key.***

**Ciphertext:** This is the scrambled message produced as output. It depends on the plaintext and the secret key. For a given message, two different keys will produce two different ciphertexts. The ciphertext is an apparently random stream of data and, as it stands, is unintelligible. ***Y is the ciphertext.***

**Decryption algorithm:** This is essentially the encryption algorithm run in reverse. It takes the ciphertext and the secret key and produces the original plaintext. ***D () is the decryption algorithm.***

The system needs a secure channel for distribution of the key between Alice and Bob. The secure channel shown in Fig. 1.5 can, for instance, be a human who is transporting the key in a wallet between Alice and Bob. This is, of course, a somewhat cumbersome method. An example where this method works nicely is the pre-shared keys used in Wi-Fi Protected Access (WPA) encryption in wireless LANs.

Secure symmetric encryption depends on two critical factors;

1. We need a strong encryption algorithm. At a minimum, we would like the algorithm to be such that an opponent who knows the algorithm and has access to one or more ciphertexts would be unable to decipher the ciphertext or figure out the key.
2. Sender and receiver must have obtained copies of the secret key in a **secure fashion** and must keep the key secure. If someone can discover the key and knows the algorithm, all communication using this key is readable.

We assume that it is impractical to decrypt a message on the basis of the ciphertext plus knowledge of the encryption/decryption algorithm. In other words, we do not need to keep the algorithm secret; we need to **keep only the key secret.** This feature of symmetric encryption is what makes it feasible for widespread use.

**CLASSIFICATION OF SYMMETRIC ENCRYPTION**

Symmetric encryption is classified into two categories. The stream cipher and block cipher encryption.

**Stream Cipher**

Stream ciphers encrypt bits individually. This is achieved by adding a bit from a key stream to a plaintext bit. There are synchronous stream ciphers where the key stream depends only on the key, and asynchronous ones where the key stream also depends on the ciphertext.

**Block Cipher**

Block ciphers encrypt an entire block of plaintext bits at a time with the same key. This means that the encryption of any plaintext bit in a given block depends on every other plaintext bit in the same block. In practice, the vast majority of block ciphers either have a block length of 128 bits (16 bytes) such as the advanced encryption standard (AES), or a block length of 64 bits (8 bytes) such as the data encryption standard (DES) or triple DES (3DES) algorithm. All of these ciphers are introduced in later chapters. This chapter gives an introduction to stream ciphers.

**CLASSICAL ENCRYPTION ALGORITHMS**

**Substitution Technique**

In this section and the next, we examine a sampling of what might be called classical encryption techniques. The two basic building blocks of all encryption techniques are **substitution and transposition.**

A substitution technique is one in which the letters of plaintext are replaced by other letters or by numbers or symbols. If the plaintext is viewed as a sequence of bits, then substitution involves replacing plaintext bit patterns with ciphertext bit patterns.

**Modular Arithmetic and Classical Algorithms**

In this section, we use two historical ciphers to introduce modular arithmetic with integers. Even though the historical ciphers are no longer relevant, modular arithmetic is extremely important in modern cryptography, especially for asymmetric algorithms.

A very popular special case of the substitution cipher is the Caesar cipher, which is said to have been used by Julius Caesar to communicate with his army. The Caesar cipher simply shifts the letters in the alphabet by a constant number of steps. When the end of the alphabet is reached, the letters repeat in a cyclic way, similar to numbers in modular arithmetic.

To make computations with letters more practicable, we can assign each letter of the alphabet a number. By doing so, an encryption with the Caesar cipher simply becomes a (modular) addition with a fixed value. Instead of just adding constants, a multiplication with a constant can be applied as well. This leads us to the affine cipher.

**Modular Arithmetic**

Almost all crypto algorithms, both symmetric ciphers and asymmetric ciphers, are based on arithmetic within a finite number of elements. In the following we introduce modular arithmetic, which is a simple way of performing arithmetic in a finite set of integers.

***Example 1: we consider a set of the nine numbers:***

***{0,1,2,3,4,5,6,7,8}***

We can do regular arithmetic as long as the results are **smaller than 9**. For instance:

***2 x 3 = 6***

***4 + 4 = 8***

But what about 8+4? Now we try the following rule: Perform regular integer arithmetic and divide the result by 9. We then consider only the **remainder** rather than the original result. Since 8+4 = 12, and 12/9 has a remainder of 3, we write:

***8+4 ≡ 3 mod 9***

We now introduce an exact definition of the modulo operation:

***Definition of Modulo Operation***

Let a, r, m ∈ Z (where Z is a set of all integers) and m > 0. We write

***a ≡ r mod m***

**Example 2**. Let m = 9, i.e., we are dealing with the ring Z9 = {0,1,2,3,4,5,6,7,8}.

Let’s look at a few simple arithmetic operations:

***6+8 = 14 ≡ 5 mod 9***

***6×8 = 48 ≡ 3 mod***

**Caesar Cipher**

The earliest known, and simplest, use of a substitution cipher was by Julius Caesar. The Caesar cipher involves replacing each letter of the alphabet with the letter **standing three places** further down the alphabet. For example,

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Plaintext | 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 |
| Cipher | 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 | C |

Using Caesar Cipher encrypt the following statement.

**Plain: meet me after the toga party**

**Cipher: PHHW PH DIWHU WKH WRJD SDUWB**

Let us assign a numerical equivalent to each letter:

|  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **a** | **B** | **C** | **D** | **E** | **f** | **G** | **H** | **I** | **J** | **K** | **L** | **M** |
| **0** | **1** | **2** | **3** | **4** | **5** | **6** | **7** | **8** | **9** | **10** | **11** | **12** |

|  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **N** | **O** | **P** | **Q** | **R** | **S** | **T** | **U** | **V** | **W** | **X** | **Y** | **Z** |
| **13** | **14** | **15** | **16** | **17** | **18** | **19** | **20** | **21** | **22** | **23** | **24** | **25** |

Then the algorithm can be expressed as follows. For each plaintext letter **P**, substitute the ciphertext letter **C**

***C= E (3, P) = (P+3) mod 26***

A shift may be of any amount so that the general Caesar algorithm is

**C = E (k, p) = (p + k) mod 26**

where takes on a value in the range of 1 to 25. The decryption algorithm is simply

**p = D (k, C) = (C - k) mod 26**

If it is known that a given ciphertext is a Caesar cipher, then brute-force cryptanalysis is easily performed: simply try all the 25 possible keys.

**Modulo Cipher/Shift Encryption**

**Definition 1.2. Shift Cipher**

Let x, y, k ∈ Z26.

**Encryption: ek**(x) ≡ x + k mod 26.

**Decryption: dk**(y) ≡ y − k mod 26.

***Example 3:*** Let the key be k = 17, and the plaintext is:

**ATTACK = x1,x2,...,x6 = 0,19,19,0,2,10.**

The ciphertext is then computed as

***y1,y2,...,y6 = 17,10,10,17,19,1 = rkkrtb***

***TASK: Perform the decryption on this operation.***

**Playfair Cipher**

The Playfair algorithm is based on the use of a 5 × 5 matrix of letters constructed using a keyword.

***Example 4: Encrypt the plaintext***

***Instrument***

***Key: Monarchy***

1. The matrix is constructed by filling in the letters of the keyword (minus duplicates) from left to right and from top to bottom, and then filling in the remainder of the matrix with the remaining letters in alphabetic order. The letters I and J count as one letter.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **M** | **O** | **N** | **A** | **R** |
| **C** | **H** | **Y** | **B** | **D** |
| **E** | **F** | **G** | **I/J** | **K** |
| **L** | **P** | **Q** | **S** | **T** |
| **U** | **V** | **W** | **X** | **Z** |

1. Plaintext is split into diagraphs ***{in st ru me nt}***
2. Plaintext is encrypted two letters at a time, according to the following rules;
3. Repeating plaintext letters that are in the same pair are separated with a filler letter, such as x, so that **balloon** would be treated as ***ba lx lo on.***
4. Two plaintext letters that fall in the same row of the matrix are each replaced by the letter to the right, with the first element of the row circularly following the last.
5. Two plaintext letters that fall in the same column are each replaced by the letter beneath, with the top element of the column circularly following the last.
6. Otherwise, each plaintext letter in a pair is replaced by the letter that lies in its own row and the column occupied by the other plaintext letter.

**HILL CIPHER**

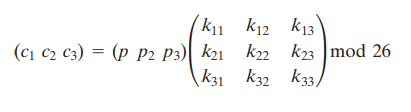
This encryption algorithm takes successive plaintext letters and substitutes for them ciphertext letters. The substitution is determined by ***M*** linear equations in which each character is assigned a numerical value (**0=A, 1=B, 2=C………..25=Z**. For, the system can be described as.

***C1 = (K11p1 + K12p2 + K13p3) mod 26***

***C2 = (K21p1 + K22p2 + K23p3) mod 26***

***C3 = (K31p1 + K32p2 + K33p3) mod 26***

This can be expressed in terms of row vectors and matrices.

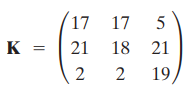


***C=E(P, K) =PK mod 26***

***P=D(K,C) = CK-1mod 26 = PKK-1 = P***

where ***C*** and ***P*** are row vectors of length 3 representing the plaintext and ciphertext, and ***K*** is a matrix representing the encryption key. Operations are performed mod 26

For example, consider the plaintext “paymoremoney” and use the encryption key



**Steps**

1. **Split the plaintext**

**Pay mor emo ney**

1. **Substitute numerical equivalence (0=a, 1=b…..25=z)**

|  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **P** | **A** | **Y** | **M** | **O** | **R** | **E** | **M** | **O** | **N** | **E** | **Y** |
| **15** | **0** | **24** | **12** | **14** | **17** | **4** | **12** | **14** | **13** | **4** | **24** |

The first three letters of the plaintext are represented by the vector ***[15 0 24]***. Then **(15 0 24) K = (303 303 531) mod 26 = (17 17 11)** = ***RRL.*** Continuing in this fashion, the ciphertext for the entire plaintext is s **RRL MWB KAS PDH.**

**NOTE: THE CALCULATION WILL BE DONE IN CLASS BOARD**