Week 4 Friday Class 2

Pravesh Jain, 17CS10040

25th September 2020

Introduction

In this lecture, we look how to have perfect secrecy for a cryptographic system. Before diving into the lecture let us first define some common definitions.
The plain text is a probability distribution on the plain text letter space. Similarly keys also have a probability distribution

$\mathbf{p_P(x)}$: A priori probability of a plain text
$\mathbf{p_k(K)}$: A priori probability of the key

This is the second part of lecture. We arrived at the following conclusion after the first part of lecture.
A cryptosystem (P, C, K, E, D) is said to offer perfect secrecy if and only if every key is used with a probability $1/|K|$, and for every $x \epsilon P$, and every $y \epsilon P$ there is a unique key, such that y = $\mathrm{e_k}$(x)
The above is equivalent to

$$p_c(y|x) = p_c(y)$$

One-Time Pad

One-Time Pad is an example of another perfect secret cipher. We already that shift cipher is also perfect secret cipher. One-Time uses the following encryption

$$Plaintext \oplus Key = Ciphertext$$

All the above described Cipher such as Shift Cipher, One-Time Pad, are also known as *Unconditionally Secure Cipher*

Practical Problems with *Unconditionally Secure Cipher*

For unconditionally secure cipher, we need to have equi-probable keys for each, thus we face multiple issues.

- We need a large number of random keys. *For example, we need to send a package of 1000 bits, we need to apply randomness of 1000 bits, and also ensure that we have same random bits available at receivers end*
- Thus, due to this large number of randomness, transferring all the keys becomes a problem. We might as well transfer the plain string through the public channel by which we are exchanging our all the random keys.
- No assurance of message integrity. In case is there is any error due to communication channel, we don't have any method for integrity verification.

Practically we try to avoid the above problem by using a single key for cipher so that we get a unconditionally secure system (as all keys are equi-probable). This is because practically, not every attacker has boundless computation resource. Hence, we use single key standard algorithms. For example **Data Encryption Standard**

Entropy

Entropy (in layman terms) can be said to give us an overview, how well our probability distribution of cipher set is. It conveys how much information is begin conveyed by our set. Mathematically Entropy is defined as

$$H(x) = -\Sigma_{x \epsilon X} Pr[x] * log_2 Pr[x]$$

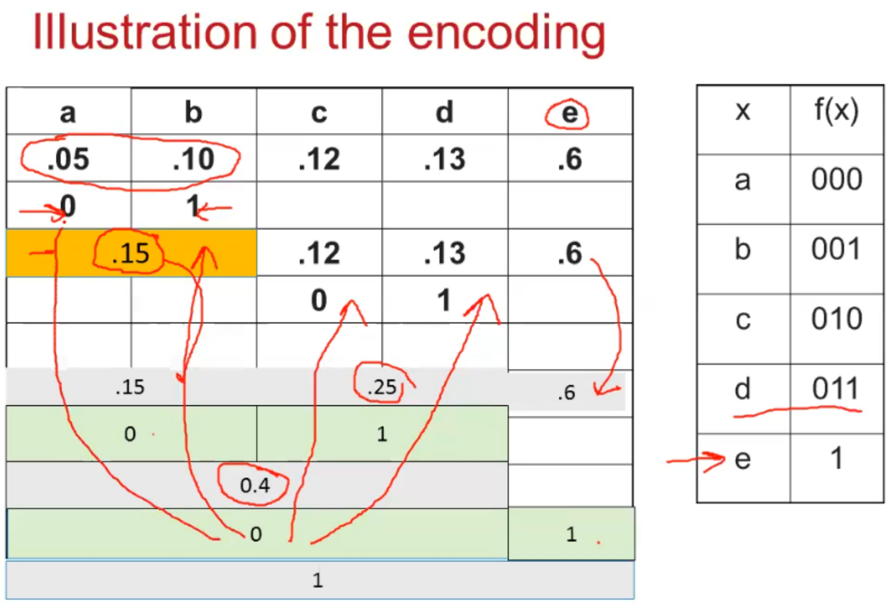
For example, let's say our plain text contains 2 alphabet $0, 1$ and with $P(1) = 1$. Entropy in this case will be $H(x) = 0$ Now consider our unconditionally secure cipher where $P(0) = 1/2$ and $P(1) = 1/2$. In this case the entropy will be $H(x) = 1$.

Huffman Encoding

Huffman Encoding provides a solution to encode messages such that their average length is as short as possible and nearly equal to $H(s)$. The following steps are used for Hoffman Encoding.

Encoding

- The message set X has a probability distribution. Arrange them in ascending order.
$$p(x1) \leq p(x2) \leq p(x3) ... \leq p(xj)$$
- Initially the codes for each element are empty.
- Choose two elements with minimum probabilities.
- Merge them into a new letter, say x12 with probability as sum of x1 and x2. Encode the smaller letter 0 and the larger 1.



x	f(x)
a	000
b	001
c	010
d	011
e	1

Figure 1: Huffman Encoding Example

- When one element remains, the code of each letter can be constructed by reading the sequence backward.

The Average length of Encoding comes out to be

$$l(f) = \Sigma_{x \epsilon X} Pr[x]|f(x)|$$

The following equality can be proved for $l(f)$

$$H(x) \leq l(f) \leq H(x) + 1$$

The above equality shows us that Hoffman Encoding is really optimised, and uses nearly the minimum amount of information (bits) required for encoding.

Decoding

It can be shown that Hoffman Encoding is indeed decodable, because Huffman Encoding in Injective, and since it is prefix free, it supports memory less decoding.

Other Results on Entropy

- Given that X and Y are random variables, we can say

$$H(X,Y) \leq H(X) + H(Y)$$

- When X and Y are independent
$$H(X,Y) = H(X) + H(Y)$$
- Conditional Entropy
$$H(X|Y) = -\Sigma p(x|y) log_2 p(x|y)$$
- $H(X,Y) = H(Y) + H(X|Y)$
- When X and Y are independent
$$H(X|Y) \leq H(X)$$

Theorem

Let (P, C, K, D, E) be an encryption algorithm. Then
$$H(K|C) = H(K) + H(P) - H(C) \quad (1)$$

where $H(K|C)$: Equivocation (ambiguity) of key given the cipher-text

Perfect vs Ideal Cipher

As we already know by now that Perfect cipher are not practical to achieve we strive to make our ciphers ideal. For an ideal cipher the idea is to make the uncertainty of the key given the cryptogram is the same as that of key without the cryptogram
$$H(K|C) = H(K)$$
To achieve this, we make $H(P) = H(C)$, and then from $eq.1$ it follows that $H(K|C) = H(K)$

Such ciphers are called Ideal Ciphers. And H(K—C) gives us an idea of security (or insecurity)

Unicity and Brute Force Attack

We define Unicity as the least amount of plaintext that can be deciphered uniquely from the corresponding ciphertext, given unbound resources by attacker.

Let's think of this way. When we try a key on our ciphertext we may get a plaintext that may not be right, but will still make semantic sense (even though the key may be wrong). To rectify this and obtain the unique key, we need more plain text corresponding to each key, so that we can verify that a key is correct or not. Unicity is the minimum of plaintext that is required to identify the correct key.

Conclusion

In this lecture, we looked at how to analyse if our crpytographic system is perfectly secure or not. We browsed various ways to judge a crpytographic system. Also understood how a perfect system should look like.