Chapter 3

Conventional Cryptography Issues

In the last two chapters we saw how encryption has developed from the very simplistic Caesar cipher right up to the complexities of the Data Encryption Standard. It was seen that the road to the level of security enjoyed by present algorithms was not straightforward and times existed when encryption algorithms offered little or no security (due to the fact that the cryptanalysts were one step ahead of the cryptographers).

The perfect security algorithm, even today, seems like an elusive goal. However, cryptographers are always striving towards this ultimate encryption scheme and although today's encryption algorithms are far from it, they do offer a security level never before seen. Unfortunately, it is generally not possible to prove whether or not an algorithm is secure and although an algorithm appears so, it is extremely difficult to say whether or not it is completely secure. The security generally relies on some mathematical problem that is intractable (for example, RSA (the public key algorithm that will be met later in the course) relies on the fact that it is very difficult to factor very large numbers). This lack of proof adds a haziness the whole area of security and the only way people generally end up trusting an algorithm is if a lot of clever people, after a lot of time and effort, are unable to break it. This is the reason DES and RSA have been allowed to exist for so long (neither have been proven to be secure)[1].

Knowing that the encryption schemes in use are not perfect, it might not seem reasonable to ask the question, is there a perfect cipher available (one that is provably secure)? If there is, then surely everyone would be using this cipher? The answers are yes and no. Yes, there is the perfect cipher but no, this doesn't mean everyone would be using it. Why? This is what we are going to look at next.

3.1 One Time Pad

There is one (and only one) algorithm that is provably secure (and it is extremely simple). It is known as the **One Time Pad** and was invented by Gilbert Vernam (of AT&T) and Major Joseph Mauborgne (USA, chief of the Signal Corps). The one time pad can be used a number of ways but the one we will look at here will be using the bitwise exclusive OR operation ($\oplus$).

As usual, the plaintext to encrypt is P and the ciphertext is C. The encryption function

[1] Of course DES is no longer considered secure (due to its very small key) but it withstood the multitude of cryptanalytic tests for quite some time.

is given as:

$$C = E_K(P) = K \oplus P \tag{3.1}$$

where K in this case is a *randomly* generated, *one time* value that is as long as the plaintext.

The Decryption algorithm is therefore:

$$P = D_K(C) = K \oplus C \tag{3.2}$$

And that's it! Much more simplistic than DES, with complete security.

How does it offer complete security? Why doesn't everyone use it? Well, to answer the first question consider the simple example of encrypting the ASCII letter for $A = 0100\ 0001$ which means "attack by air". Choose a random number $K = 1010\ 0110$ (as long as the plaintext), then:

$$\begin{aligned} C &= E_K(P) = K \oplus P \\ &= 1010\ 0110 \oplus 0100\ 0001 \\ &= 1110\ 0111 \end{aligned}$$

As cryptanalysts we have intercepted the ciphertext $1110\ 0111$ and want to determine what it means. Because there is only 8 bits in the key it is very easy to do a brute force search as there are only $2^8 = 256$ values. At some point during the search we come across the key $1010\ 0110$ and run the decryption algorithm on it to receive the plaintext $0100\ 0001$ which we know is the ASCII character for A which means "attack by air". This is great because it is now known how the enemy are going to attack. However, a short time later the search comes up with the key $0001\ 0010$ which yields the plaintext $0101\ 0011$ which is known to equal the ASCII character S which means "attack by sea". Which one is correct? Assuming that the key was completely random (not related to the plaintext in anyway) and used only once then the probabilities that either one is correct are the same and no assumption can be made. Although a very simplistic example, the basic premise holds for all types of One Time Pads provided they are used correctly.

Unfortunately, choosing a key that is random is not very practical. First you have the problem of generating the random key that is as long as the plaintext (which can be quite long) and then you have to communicate this key to the recipient of the message. It is these practical problems that cause the One Time Pad only to be used by the most secret of secret communications. The word "Pad" is used because the random, one time keys used to be written on a pad that the sender(s) and recipient(s) would each receive. Once a key was used the page of the pad was ripped out and discarded[2].

[2] Another problem inherent with the one time pad is synchronisation, i.e. making sure the sender and receiver are on the same page of the pad.

3.2 Triple DES

So it is clear that the only perfectly secure algorithm we have is not very practical for use in the field. It is necessary therefore to turn back to some of the other algorithms and see what can be done to improve their security. It has always been thought that DES has had issues with its security and in 1998 a brute force search was done in less that 5 days. It was necessary to try and improve DES and to somehow increase the key size. The most secure thing to do would have been to devise a completely new algorithm however this was not the most practical thing to do. It was desired to keep DES in the picture and as a result the algorithm **Triple DES** was realised.

Triple DES is simply applying the DES algorithm three times using two different keys (see figure 3.1):

$$C = E_{K_1}[D_{K_2}[E_{K_1}[P]]] \tag{3.3}$$

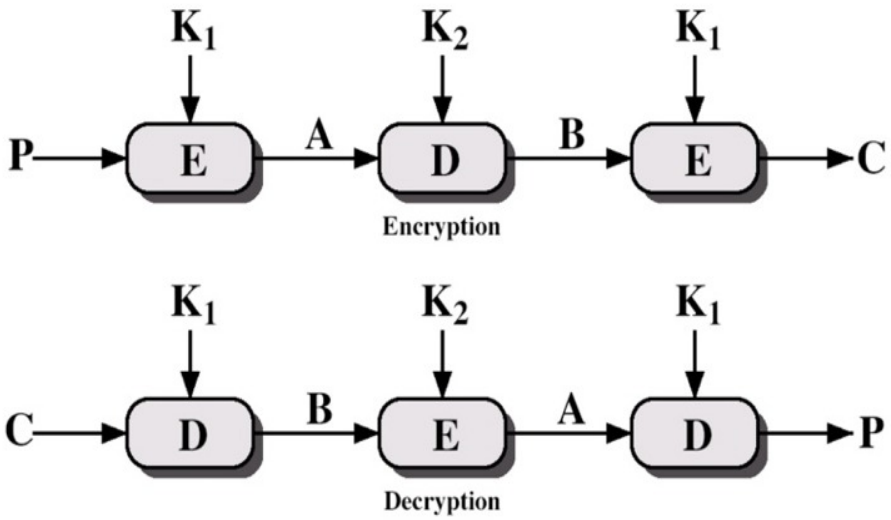

Figure 3.1: Triple DES.

As a result of the order of encryptions and decryptions this is known as EDE or Encrypt-Decrypt-Encrypt. The fact that a decryption is performed on the second iteration is simply for backward compatibility and offers no extra security to the algorithm. This can be seen from:

$$C = E_{K_1}[D_{K_1}[E_{K_1}[P]]] = E_{K_1}[P]. \tag{3.4}$$

One of the reasons the One Time Pad is not practical is the problem of key distribution - delivering the key to each party involved in the communication. This is not only a problem with the One Time Pad (although it is a lot more difficult in this case) but is in fact a problem with all symmetric cryptosystems. Also, assuming a particular

cipher has been chosen and the keys have been distributed to each party, where in the communications system do we place the encryption function itself? Both of these ideas will be discussed next with the latter being dealt with first.

3.3 Placement of the Encryption Function

Historically, cryptography has mainly used conventional encryption (single key) to provide confidentiality. Only in the last few decades have other considerations such as authentication, integrity, digital signatures and the use of public key encryption emerged. Before addressing these newer topics, elements of the problem of confidentiality are addressed and these include:

- Placement of the encryption function (link or end to end).
- Key distribution.

An attack can take place at many points in a communications channel. Examples include monitoring traffic on a LAN, where a message from one station to another is visible to all stations. Alternatively an attack could be focused on a wiring closet or other element of the network shown in figure 3.2.

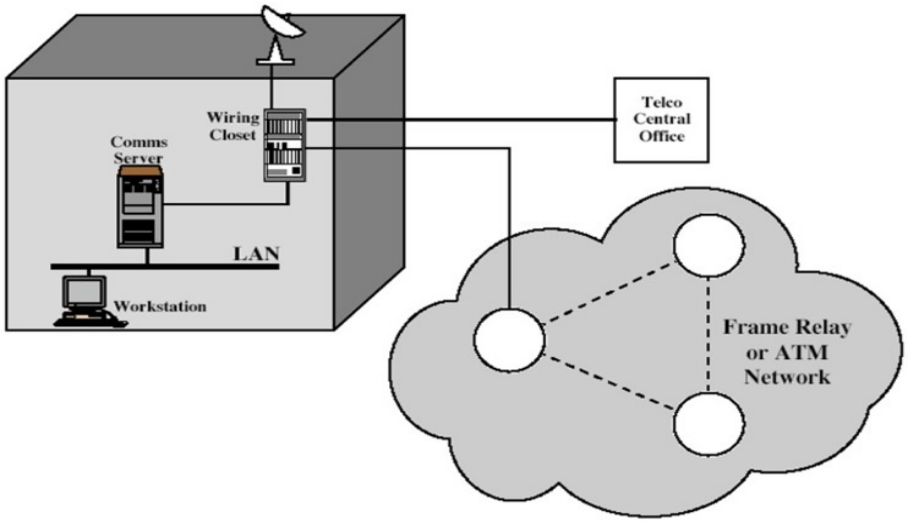

Figure 3.2: Points of vulnerability in a networking environment.

For active attacks the attacker must gain physical control/access to a portion of a link and be able to insert and capture transmissions. As mentioned in the first lecture, active attacks are easier to detect than prevent and the idea is to have some system in place that will allow recovery from these attacks. For passive attacks (which are easier to prevent than detect) the attacker only needs to monitor transmissions. Twisted pair and coax. may be breached by invasive (active or passive) or inductive (passive only)

taps. Satellite or microwave links are easily monitored without risk to the attacker. It will also be seen later in the course how the power consumption and electromagnetic radiation of an encryption device can be used to gain information from the system.

If encryption is to be used to counter all the attacks mentioned above then we need to decide what to encrypt and the location of the encryption devices. Figure 3.3 shows both link and end to end schemes which may be used alone or both together. With link

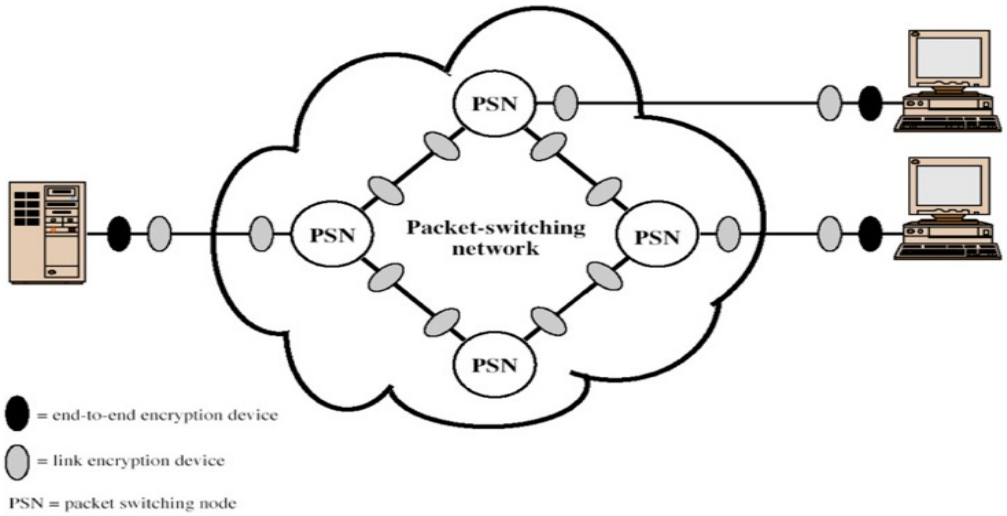

Figure 3.3: Link and end to end encryption.

encryption note the following:

- Each vulnerable link is equipped so there is a large number of encryption devices with a unique key for each pair.
- All traffic over all links is secure.
- Each message is vulnerable at each switch as the virtual circuit number is required for routing.

In the end to end case

- Encryption is carried out only at the two ends.
- It is secure against attacks on links and switches.

In this case however, the sending host must not encrypt the routing header of the packet. If it were encrypted, the routers and switches would have no way of determining the destination of each packet as they would be unable to decrypt this information. As a result, data are secure but the traffic pattern is not because an attacker has access

to header information. This means an attacker can determine which two parties are communicating with each other and the amount and regularity of the information. This can often reveal a great deal about the communication.

With link encryption, placement of the encryption function is at a low level in the communications hierarchy, in OSI terms, either physical or datalink layer. In the end to end case, several choices are possible from network to application layer. Deployment of encryption services on end to end protocols (such as network layer X25 or TCP) provides end to end security in a fully integrated network but cannot deliver the necessary services for traffic that crosses internetwork boundaries such as electronic mail etc. This is illustrated in figure 3.4.

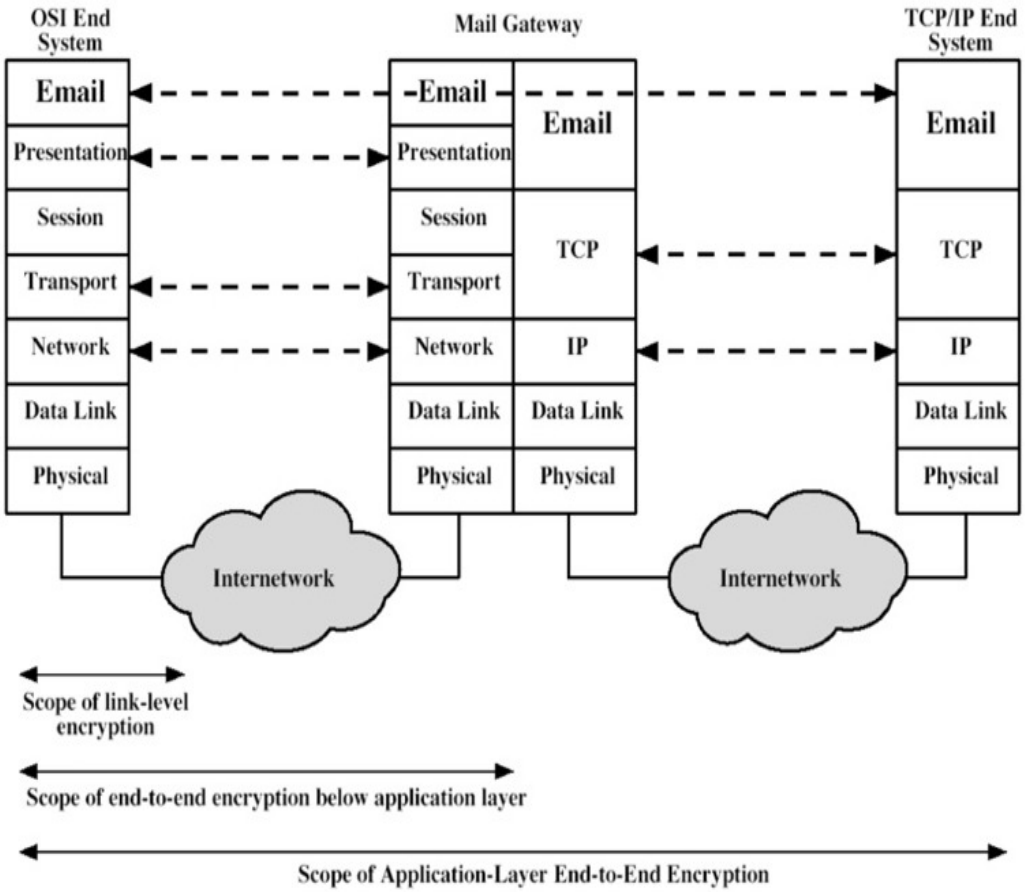

Figure 3.4: Implications of encrypting traffic that crosses Internetwork boundaries.

The drawback of application-layer encryption is that the number of entities to consider increases dramatically. A network that supports hundreds of hosts may support thousands of users and processes. Thus, the number of secret keys (at any point in time) is very large as will be seen below. Figure 3.5 illustrates the implications of the various encryption strategies.

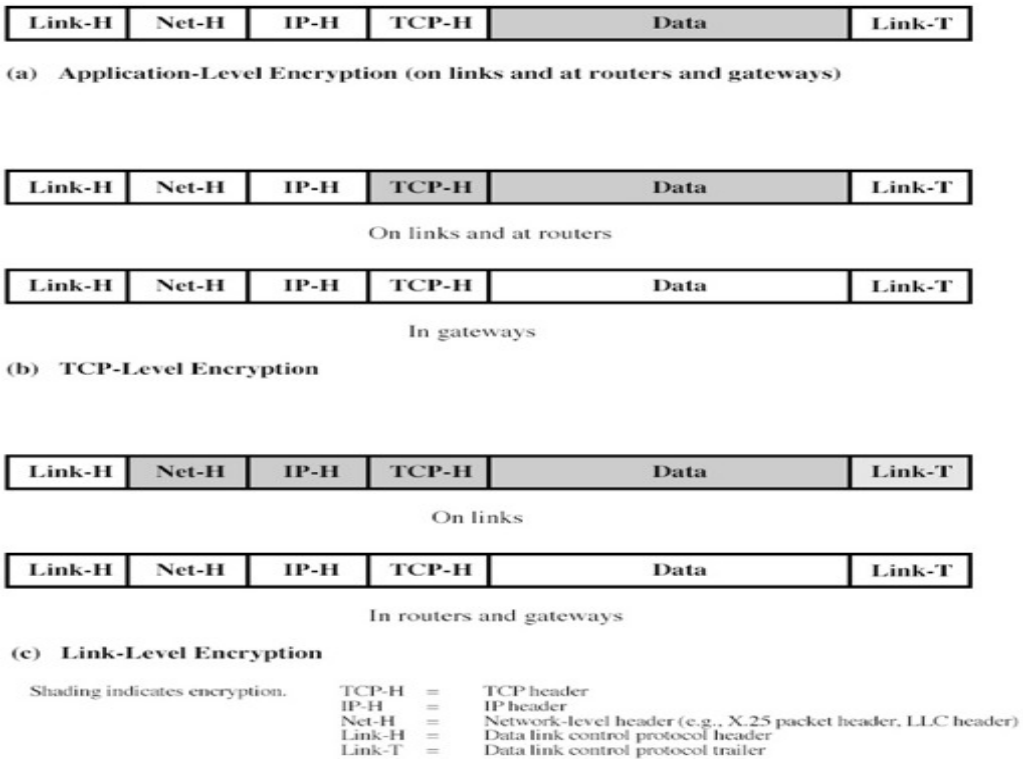

Figure 3.5: Implications of the various encryption schemes.

3.4 Key Distribution

In conventional encryption, the two parties of the exchange must have the same key and it must be protected from access by others. Also, frequent key changes are required to limit the amount of data compromised if an attacker learns the key. The strength of any cryptographic system rests with the **Key Distribution System (KDC)**.

Key distribution between two parties A and B can be achieved in a number of ways as follows:

1. A key may be selected by A and physically delivered to B.
2. A third party can select and physically deliver it to A and B.
3. If A and B have previously and recently used a key, one party can transmit a new key, encrypted using the old key.
4. If A and B each has an encrypted connection to a third party C, C can deliver a key on the encrypted links to A and B.

Options 1 and 2 call for manual delivery of a key which is ok in the link encryption case. However for the end to end case with many hosts/terminals (and distributed

also), many keys are required with each terminal needing to exchange with many other terminals. The scale of the problem depends on the number of communication pairs that must be supported. If end to end encryption is done at IP level then a key is required for each pair of hosts on the network. For N_h hosts:

$$N_k = \frac{N_h(N_h - 1)}{2} \tag{3.5}$$

where N_k is the number of keys required.

Equation 3.5 is easily seen to be true. If there are N_h hosts then this means that any one host can communicate with the $N_h - 1$ other hosts. If we consider each host separately and add up the number of connections we get $[(N_h-1)_1+(N_h-1)_2+\ldots(N_h-1)_{N_h} = N_h \times (N_h - 1)]$. As we have considered each connection twice (once for $(N_h - 1)_i$ and then again for $(N_h - 1)_{i+1}$) it is necessary to divide by 2.

If encryption is at end user level then a key pair is required between every communication pair. For example, 10,000 applications require approximately 50 million keys.

While option 3 is a possibility for both link and end to end encryption schemes, there is the problem that an attacker may gain access to one of the keys. If this happens all subsequent keys will be compromised. However, option 4 seems to be a better and more widely used alternative. In this scheme a Key Distribution Centre (KDC) is responsible for distributing keys to each pair of users as needed. A minimum of two levels of keys are required: **session keys** and **master keys**. Typically, a session key is a temporary key used for the duration of a logical connection and then discarded. Each session key is obtained from the KDC as needed over the same communication network used for end user communications. Therefore, session keys are transmitted in encrypted form using a master key which is shared by the KDC and each user. Only N master keys are required (one for each user).

Figure 3.6 shows a key distribution scenario. Assume that each user A and B shares a unique master key (K_a, K_b) with the key distribution centre (KDC). Assuming that user A wishes to establish a logical connection with B and requires a one-time session key to protect the data to be transmitted from A to B:

1. A requests a session key from the KDC. The message includes the identity of A and B and a unique identifier N_1 for this transaction - called a "**nonce**". This could be a timestamp or random number and it is desirable that it be difficult to guess.

2. The KDC responds with a message encrypted using K_a thus only A can decode it. It contains two items intended for A, the one-time session key K_s and the original request message including the nonce. The latter allows a match to be made between this response and the request. With this data, A can verify that its original request was not altered. As well as these two items, it includes another two items intended for B but encrypted using K_b: the session key K_s, and an identifier of A (its network address) **IDA**.

3. A stores the session key K_s and forwards the information encrypted with K_b to B:

   $$E_{K_b}[K_s||\mathbf{IDA}]$$

   where $||$ denotes concatenation.

   B now knows the session key K_s, that the other party is A, and that the information originated at the KDC (because it was encrypted using K_b). At this point K_s has been delivered securely to A and B for their session.

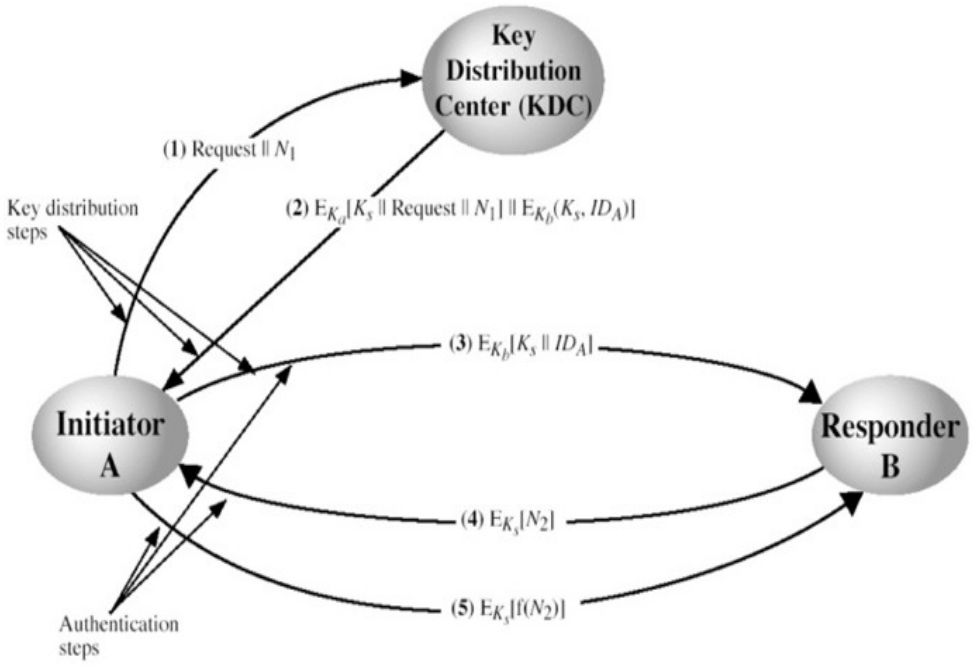

Figure 3.6: Key Distribution Scenario.

Two additional steps in the process are desirable:

4. Using the newly minted session key for encryption, B sends a nonce N_2 to A.
5. Also, using K_s, A responds with $f(N_2)$ where f is a function transforming N_2 (say add one).

These steps assure B that the original message it received was not a replay[3]. Note that the actual key distribution involves only steps 1-3 but that 4,5 as well as 3 perform an authentication function.

[3]This is were information from a previous exchange was captured by the attacker and forwarded to B at a later time.

For large networks a hierarchy of KDCs may be established. Also, the more frequently the session keys are changed, the more secure the complete system (but the higher the overhead on the network).