Encrypting Network Traffic

Mark Lomas
Computer Security Group
University of Cambridge
Computer Laboratory

Encryption may be used to maintain the secrecy of information, to help detect when messages have been tampered with, or to prove the identity of the originator of a message. Why, we might ask, are so many messages still sent unencrypted?

There are several possible explanations. Many people are unaware of the potential benefits of encrypting messages before transmission. Alternatively they may not be aware how insecure their computers are; for example most of the computers in use within the Computer Laboratory here in Cambridge transmit user's passwords unencrypted so that they may be read by anybody who cares to monitor network transmissions; this is typical of many if not most computer networks. Another possibility is that the cost of encrypting network traffic is considered too high for the potential benefits.

If people are unaware of the benefits or do not realise that their machines are insecure then they should be educated so that they can make an objective decision as to whether or not they should encrypt their messages. In this paper I aim to address the idea that encryption may be too costly; I believe that many forms of transmission may be encrypted at a much lower cost than many people realise and that their decisions regarding the use of encryption may change if they are shown the true cost.

Network Overheads

Before we can evaluate the cost of encrypting network traffic we need some measure against which we can compare this cost. The network over which we will send this traffic imposes costs of its own. If the time taken to encrypt a message is much greater than the time taken to transmit it then encryption will cause a noticeable degradation in the performance of the network. If we find that the transmission is much more costly than the encryption then the encryption will cause very little degradation in performance. In such a case I would suggest that encryption is worthwhile.

For my measure of network performance I chose to analyse the round-trip time for a series of transmissions. These measurements were performed between two VAXstation-2000 computers connected by an Ethernet local area network. Both machines were running the Ultrix operating system and they used the TCP/IP protocol to manage their transmissions.

My experiments indicate that we cannot describe the performance of such a network configuration in terms of a single number. This was not particularly surprising since I had expected network performance to depend upon a variety of factors; these include the load upon the network and the load upon the machines themselves. Although the round-trip time did not vary greatly I achieved a much higher throughput when the packet size was large rather than small.

For example, typically it would take 46 seconds to transmit and receive 10,000 packets of length 128 bytes (a total of 20,000 packets). When the packet size was increased to 1,024 bytes then the total time increased only slightly to 50.5 seconds. These represent a throughput of 56,000 and 406,000 bytes per second respectively; the larger packet size resulted in 7.3 times as much throughput.

DEA Overheads

An implementation of the Data Encryption Algorithm [ANSI81, NBS77] was supplied with the VAXstations. This proved to be much too slow. Each encryption required 120ms which means that it could perform 8.33 encryptions per second. Each encryption operated upon eight bytes of plaintext so it could encrypt 66.67 bytes per second which is at best a factor of 840 slower than the network. This was obviously unacceptable.

I designed a rather faster implementation of the DEA. This implementation was a factor of 14 faster than that supplied with the machine; although this was a reasonable improvement it was still too slow to be used to encrypt network traffic.

An Alternative Encryption Scheme

Let us consider using a one-time-pad to encrypt network traffic. As with all one-time-pads this would suffer from the disadvantage that a potentially vast amount of key information would have to be sent by some other means. There is a variant on the one-time-pad that does not suffer from this problem.

A conventional pad requires a stream of randomly chosen bits. If we could produce a pseudo-random stream that may be described in the form of an algorithm then we could execute this algorithm on both machines. The result of these calculations would then be used as the encryption key. Unfortunately most pseudo-random number generators are not sufficiently secure to be used for this purpose. If however we use the DEA to encrypt such numbers then the ciphertext would be suitably random for our purposes.

This may seem a strange observation. If we have to apply the DEA to produce each of our random numbers then we cannot produce random numbers any more rapidly than we could have encrypted a message of the same length. There is however a method by which we can use this new system.

One-time-pads have an interesting characteristic. If we can send the key to a one-time-pad over a slow channel well before the transmission that we ultimately wish to make then this second transmission may take place far more quickly. We can pretend that we have a true random number generator and that the DEA implementation is merely a slow channel between this generator and the two clients. Whether this is an appropriate scheme depends upon characteristics of the data that is most commonly transmitted between machines.

Typical Network Clients

Within the University Computer Laboratory, as at many other sites, there are a variety of machines connected together via a computer network. Here in Cambridge we use more than one type of network but I chose for the purpose of our experiments to use only an Ethernet network as this is more typical of the networks found at other sites than the other experimental networks that were also available.

As might be expected there are a variety of different clients that use the Ethernet however two types of network traffic seem to dominate at our site. Users often log in to a computer system from either a workstation or a terminal connected to a terminal concentrator; even when they use a workstation this is typically used merely to communicate with the user rather than as a machine on which to perform computation. The network traffic that these two types of device produce I classify as 'terminal traffic'.

The other form of network traffic that we experience is communication between computers and associated file-servers. Most users have files that are physically resident on a particular machine but may be accessed from any of a large number of machines. This requires that files be transmitted from one machine to another as quickly as possible. I call these types of transfer 'file-server traffic'.

There are other forms of traffic with which I am less concerned since they are less frequent. For example each machine sends out regular information showing which users are logged into the machine; these transmissions also serve to prove that the machine has not crashed. The interval between such transmissions is usually of the order of seconds or even minutes. There are also messages such as boot-server requests that take place only when a computer is rebooted and even then only if the machine does not have a copy of the boot file on a local mass-storage device such as a disc or tape. The interval between these transmissions could be of the order of days.

Terminal and file-server traffic are far more frequent than the other forms of transmission. Much of the time users are editing files and so most of the transmissions taking place on their behalf are those associated with their terminal or workstation. Even though these may be frequent they usually consist of very small amounts of information – perhaps just a single key-press. The amount of information transferred is likely to increase as workstations become more common but is still smaller than file-server traffic.

File-server traffic represents most of the data that might in the past have been transferred between a computer and a local disc. A particular file transfer might consist of several megabytes of data to be transferred as quickly as the network can manage. It is an interesting challenge to be able to encrypt this amount of data and at this rate; it is fortunate that these transfers tend to be bursty, they aren't sustained for long periods of time, otherwise this might not be practical.

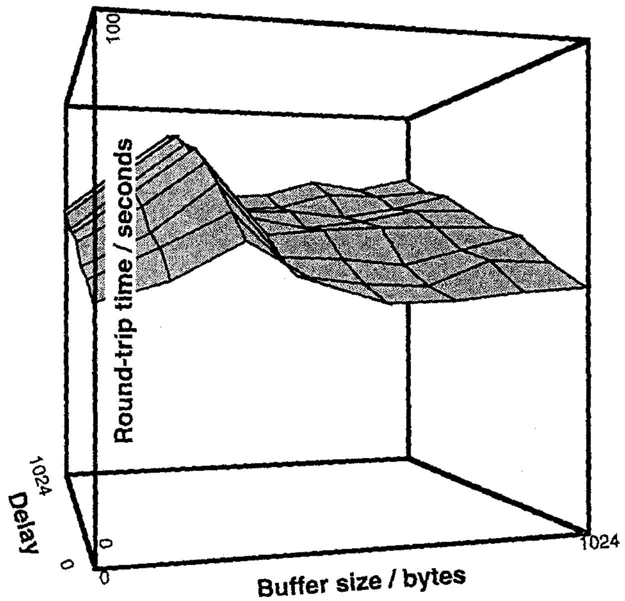
Atypical Network Clients

Since our typical clients must share the network with less typical clients we ought to consider whether these other clients might influence our measurements. My experience tends to suggest that such clients have little influence.

The series of measurements that I took to characterise network performance were repeated at different times of the day and on different days, both during the week and at the weekend; never did I see any significant variation in these readings – repeating a particular experiment produced a change in measurement of less than 1%. This observation tends to suggest that the other network clients that I consider less typical have very little effect upon the performance of the more common form of client.

The preceding graph shows how the round-trip time varies as the buffer size (the size of packet to be transmitted) and delay between packets are varied. Notice the characteristic hump to the left of the graph: the throughput can be increased to a large degree by choosing a packet size sufficiently large to avoid this hump.

I know that it would be possible to influence the measurements by sending large amounts of data over the same network. For example I also ran my measurement programs concurrently between two pairs of machines; although there was no direct communication between the pairs they shared a single Ethernet cable. The measurements under these circumstances showed a degraded network transfer rate between both pairs of machines. Sharing the network leads to



Time to transmit 10,000 packets

degraded performance which is what I would expect. Fortunately the network load that is more usual on the networks that I used does not appear to cause such degradation.

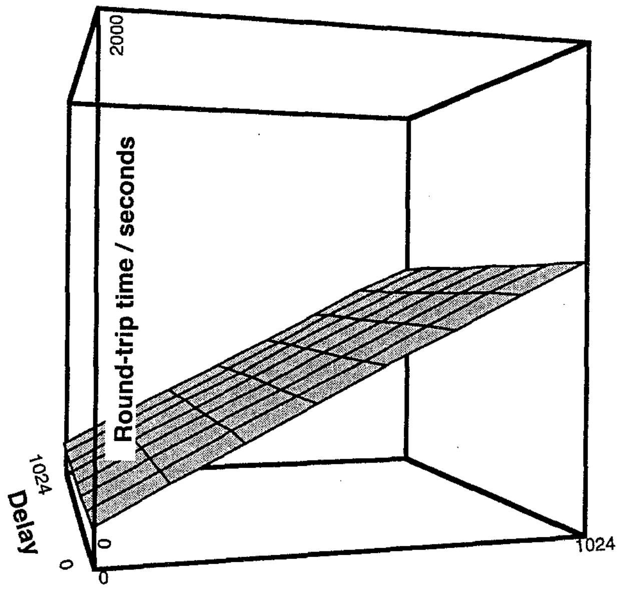
The results of these experiments do not prove that there is not a client or group of clients that exert a constant load upon the network, degrading performance at all times of the day and night. Even if this were the case I would consider it as an inherent characteristic of the network; experiments under such conditions would provide a reasonably accurate estimate of expected network performance. In any case I know of no clients on our network that I would expect to behave in this manner.

Encryption Speed

The encryption system that I have described has a non-linear encryption speed; by this I mean that the time taken to encrypt an amount of data is not a linear function of the length of this data. This is to be expected. By comparison conventional encryption using a method such as the DEA has a very predictable encryption time. If the new scheme were to be plotted the result would be indistinguishable from the graph above; the conventional scheme is very different, however.

The following graph shows the transmission time for sending 1,000 packets of various lengths and with various inter-packet delays when encrypted using the DEA in a conventional manner. I chose to time only 1,000 packets rather than 10,000, as in the previous case, because the encryption time was so much greater than the transmission time that the experiments would have taken much too long to complete; even so the experiments took a number of days. In any case the variations caused by changes in packet size or delay are insignificant when compared to the encryption time.

It should also be noted that only one encryption took place for each packet transmitted. In a practical rather than experimental system we would expect each packet to be encrypted, transmitted, then decrypted; the reply would also be encrypted, transmitted, then decrypted. If the reply was the same length as the original packet then this entire process would take four times as long as the already very large times that I measured. The transmission time using DEA in a conventional manner is directly proportional to the amount of data to be transmitted.



Time to transmit 1,000 encrypted packets

Consider two machines connected by an Ethernet that will encrypt communications between them using this scheme. Imagine that they have completed whatever startup procedure is necessary to agree upon an encryption key; this initialisation might use any one of a number of authentication protocols.

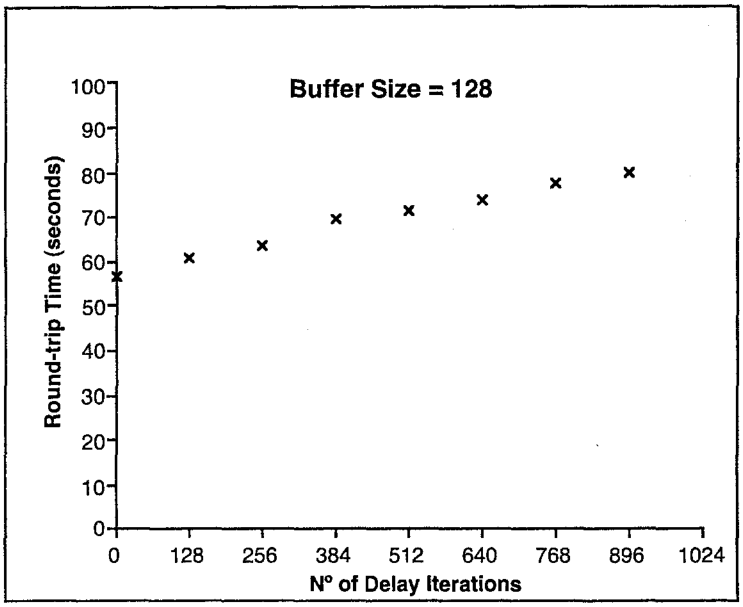
Before either machine can encrypt any data it must generate part of the required one-time-pad; this requires at least one application of the DEA and so introduces a delay at least as great as encrypting the data using the DEA would have done. On the machines that I used this requires approximately 8.5 milliseconds which is therefore the minimum delay before the first communication.

Now imagine that little communication takes place between the two machines for a period of time during which they can perform a number of DEA encryptions. The first transmission after this delay can take place far more quickly as encryption requires the application of the exclusive-or operator rather than a far more time-consuming function.

This is where the classification of network traffic into terminal traffic and file-server traffic becomes useful. Traffic from terminals arrive at a rate far slower than 8 bytes in 8.5 ms, which is the rate at which new encryption keys can be generated. If we can use the time between transmissions to generate new keys then we can transmit new data almost immediately.

My measurements show that the time to encrypt data using this method where the keys have been computed in advance is lower than the random variation in network performance. Essentially it may be considered negligible. For simplicity I assumed that a fixed packet-size of 128 bytes would be sensible for terminal traffic; it might be slightly more efficient to use a smaller packet-size from the terminal to the computer compared to the size sent from the computer back to the terminal, although this would have very little effect in practice. I experimented with a variety of delays before transmission and these readings represent the time required to send and receive 10,000 packets.

The following graph shows the results of such experiments. The round-trip for each packet even after encryption is several orders of magnitude smaller than the timings shown on the preceding graph; it is considerably faster to encrypt in this manner. I should point out however that the total amount of processing is about the same in either case. Using conventional encryption we require a large amount of computation immediately before transmission but using this scheme the computation may take place well before the transmission – before the data to be transmitted is available for encryption.



I suggest that since this encryption method does not slow down such transmissions to a measurable degree it is well suited for encrypting information to be sent at this rate. If data is to be sent at a rate of less than one byte per millisecond, using equipment comparable to that which I used, then this should be a suitable method of encryption. The idle time between events such as key presses may be usefully used to compute the necessary encryption keys.

File-server traffic appears to present a more difficult problem. As I suggested earlier people aim to perform file transfers as quickly as the combination of network and other equipment will allow. I estimate that using the equipment and network protocols that I had available to use, it would require 3.2 ms to send a 1,024 byte packet; this is several orders of magnitude faster than I could encrypt the same amount of data using the DEA.

Fortunately this rate of transmission is seldom sustained for long periods of time; the figure of 3.2 ms to send 1,024 bytes represents the peak data transfer rate. If it were to be sustained then this would be the rate at which we would use up our supply of pre-computed encryption keys; we would eventually exhaust our supply. Once the supply has been exhausted the maximum rate for encryption would be approximately the same as that to apply DEA directly. This is the reason that the encryption time is not a direct linear function of the amount of data.

When to Encrypt Using this Method

The factor that determines whether the method is suitable is not the peak transmission rate but the average. Provided the average rate does not exceed about one byte per millisecond the method may provide an efficient encryption mechanism.

If the average transmission rate does exceed this value over a long period of time then the pre-computed keys will eventually become exhausted. When this happens it will become necessary to perform at least one time-consuming encryption to generate the key necessary before the next packet can be encrypted. The performance of the network will be degraded to that which it would have had if conventional encryption had been used.

Had I used different equipment then the value of this threshold might differ but the method for determining the correct value should be very similar: the average transmission rate for unencrypted data should be determined as should the speed of the underlying cryptosystem; if the average rate is less than the encryption rate then this encryption scheme should not degrade network throughput, providing there is spare processing capacity to generate the keys.

We should note, however, that this represents a limit above which the scheme becomes impractical rather than a limit that determines that it will be practical. If the average rate is too high then the system cannot be more efficient but, if the rate is not too high, it may still be impractical because there is no spare processing capacity on the host computer. It may be necessary to add hardware encryption devices to compensate.

The experiments were carried out using computers that have since been replaced but I have subsequently repeated them using newer and faster machines with similar results. The technique that I have described is just as applicable on more modern machines. It should be noted, however, that like a one-time-pad this method of encryption provides confidentiality but no provision for preserving message integrity: it would be relatively easy for an attacker to corrupt a message in a predictable way. Inverting a bit in the ciphertext would have the effect of inverting a bit in the plaintext resulting from decryption. I recommend, therefore, appending some form of MAC (Message Authentication Code) before encrypting.