REPRINT

DEPARTMENT OF THE NAVY

NAVAL RESEARCH LABORATORY

A Pump for Rapid, Reliable, Secure Communication

Myong H. Kang and *Ira S. Moskowitz*

FROM:
Proceedings of the 1st ACM Conference on Computer & Communications Security, Fairfax, VA, Nov 3-5 1993, pages 119-129, ACM Press.

CONTACT:
Myong H. Kang, Information Technology Division, Mail Code 5542, Naval Research Laboratory, Washington, DC 20375.

Ira S. Moskowitz, Information Technology Division, Mail Code 5543, Naval Research Laboratory, Washington, DC 20375.

E-MAIL:
mkang@itd.nrl.navy.mil
moskowit@itd.nrl.navy.mil

A Pump for Rapid, Reliable, Secure Communication

Myong H. Kang and Ira S. Moskowitz
Naval Research Laboratory
Information Technology Division
Washington, D.C. 20375

Abstract

Communication from a low- to a high-level system without acknowledgements will be unreliable; with acknowledgements, it can be insecure. We propose to provide quantifiable security, acceptable reliability, and minimal performance penalties by interposing a device (called the Pump) to push messages to the high system and provide a controlled stream of acknowledgements to the low system.

This paper describes how the Pump supports the transmission of messages upward and limits the capacity of the covert timing channel in the acknowledgement stream without affecting the average acknowledgement delay seen by the low system or the message delivery delay seen by the high system in the absence of actual Trojan horses. By adding random delays to the acknowledgment stream, we show how to further reduce the covert channel capacity even in the presence of cooperating Trojan horses in both the high and low systems. We also discuss engineering tradeoffs relevant to practical use of the Pump.

1 Introduction

Currently many computer system users who deal with confidential/sensitive information use systems that are dedicated to a high level of security, i.e., system-high systems. All information residing in the system is marked with the system high level, even if some of the information is in fact innocuous. Sharing information between different security levels is quite cumbersome in this kind of system, which impedes high speed data transfer. Multilevel secure (MLS) systems promise to ease this problem.

Unfortunately, the security constraints in MLS systems often damage overall performance. We propose to add non-uniform random noise to response time to permit a high degree of security, while maintaining efficient performance standards. This paper is a first step in a research plan that we will eventually implement on an actual MLS system. We also plan to make further study of our ideas through various simulations. However, in this paper we present the system engineer with some rules of thumb to get both security and performance requirements within certain guidelines.

Let us consider a particular, either centralized or distributed, MLS system where two processes (nodes), one high and one low, are single-level processes (nodes). Two kinds of communication can exist between these processes (nodes):

- The high process sends information to the low process (this is sometimes called downgrading).
- The low process sends information to the high process.

The first type of communication grossly violates the Bell-LaPadula constraint [BeL76] that is used by most of the existing MLS systems. The second type of communication does violate Bell-LaPadula in a subtle manner if acknowledgements of a message being passed are provided to the low process. This, in fact, results in a covert channel. In this paper, we only consider the second type of communication, which we refer to as a low-to-high communication. Most MLS systems that are based on the Bell-LaPadula model accomplish this low-to-high communication through *read-down* or *blind write-up*. However, there are some reliability problems in these methods.

In secure computer systems, in addition to security, the following characteristics are required in most communication protocols: (1) reliability[1], (2) reasonable performance, and (3) a reasonable way to collect garbage. Sometimes the recoverability[2] in the case of system crash is a desirable feature. These are the goals that our method achieves in a secure fashion.

The rest of the paper is organized as follows: In section 2, we examine several communication methods in detail and discuss some problems inherent in them. A quasi-secure low-to-high communication mechanism that can achieve all of the above desirable properties is introduced in section 3. Section 4 describes the relevant security features of this mechanism. The capacity of the covert channel is analyzed in section 5. Section 6 summarizes this paper.

2 Background and Motivation

In this section, we present several methods of inter-process communication and show the difficulty in simultaneously achieving the goals in section 1.

2.1 A Non-Secure (Conventional) Communication Protocol

Communication protocols used in non-secure computer systems typically achieve reliability and reasonable performance. They also contain a reasonable way to collect garbage through the following typical scenario that is shown in figure 1.

[1] The communication is reliable if the source sends a message then it knows that either the message was delivered safely or that it has to retransmit due to errors.

[2] The communication is recoverable if the message is delivered to the destination even if the system fails, i.e., the system will recover and continue to deliver from the failed point.

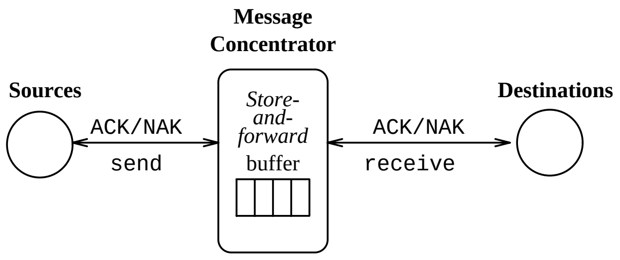

Figure 1: Message passing from sources to destinations.

If the message passing occurs in a distributed environment, then the source and destination processes may reside in two different computers and the message concentrator itself may be yet another computer; if the message passing occurs in a single computer, then the operating system may play the role of the message concentrator (e.g., *pipes* in a UNIX system).

A typical message passing between a source and the message concentrator, and the message concentrator and a destination, goes as follows:

1. Establish transmission/connection.
2. Send a message.
   - If the sender receives an ACK, then discard the message from the sender's memory.
   - If the sender either receives a NAK or times out, then retransmit the message.
3. If there are more messages to send, then go to step (2).
4. Signoff/Disconnection.

If the source sends messages faster than the destination can take (either due to slow processing or failure at the destination) then the buffer in the message concentrator may be filled. The source then will be unable to send any more messages until the destination empties some messages from the message concentrator. We say that the source has been *blocked* in this case. In a non-secure environment, determining the size of buffer that keeps the message blocking probability within specified design limits has been widely studied [Sch77].

In a secure environment, if the source, which resides in a low system, sends messages to the destination, which resides in a high system, then the sender cannot use the same protocol because ACK/NAK arrival times can be used to send signals covertly. The capacity of this covert channel is analyzed in section 3.2.

2.2 Read-Down

Read-down allows the high level user/process (High) to read the low level user's/process's (Low) memory. But it does not allow High to send signals after it reads Low's message. Consider the following implementation of a mechanism that passes information from Low to High using only *read-down*. Low inserts its message into a low message buffer. High reads the message out of the buffer. However, Low has no indication that the message has been read. The message sits unchanged in the buffer until Low deletes it from the buffer.

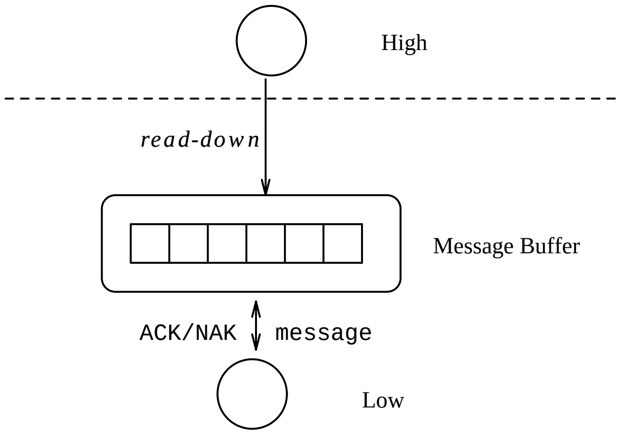

Figure 2: Message passing from Low to High using *read-down*.

Assuming that no error has occurred in the *read-down* procedure, there are two typical ways to achieve this communication:

- *High keeps polling the low buffer.* The disadvantage of this method is that the polling can waste resources (e.g., CPU time).
- *High periodically performs a read-down (e.g., every Δ time).* In this case, Low cannot send more than one message per Δ. Otherwise, (unless there is an infinite buffer) Low may delete messages which have not been read by High. If Δ is too small, then, like the polling method, this method will waste resources. If Δ is too large, then the message rate will be reduced (i.e., the message rate of this communication is less than or equal to $\frac{1}{\Delta}$ *messages/unit time*).

Another drawback of this method is that the low process cannot detect if the high process is ready to receive messages or not. For example, if the high process crashes, there is no way for the low process to detect the situation and stop sending messages (otherwise Low may delete messages that High has not read yet).

2.3 Blind Write-Up

Blind write-up allows the low level user/process to write on the high level user's/process's memory. But it does not allow High to send an ACK/NAK to Low. We could implement a blind write-up mechanism as follows in figure 3.

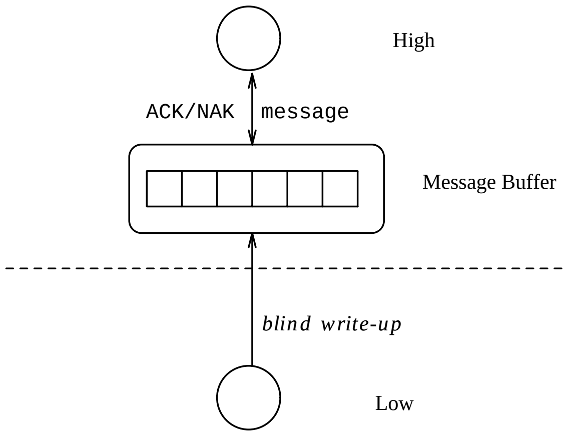

Figure 3: Message passing from Low to High using *blind write-up*.

The low process writes its message into the high message buffer and the high process reads messages from the buffer. Since the low process does not know the condition of the high process, it has to send a message and hope that the high process receives it. Hence, this mechanism is unreliable because even if there was an error during transmission, there is no way for the low process to discover it and retransmit the message.

3 A Quasi-Secure Low-to-High Communication Channel

The communication mechanisms presented in the previous section all have undesirable characteristics. The read-down and blind write-up methods are unreliable because there is no way of knowing what happens to the message after the source sent it. The **Pump** that will be introduced in this section is a variation of the conventional communication protocol that was introduced in section 2.1. We already mentioned that the conventional communication protocol has a covert channel. One way to circumvent this timing channel problem is to limit the ACK/NAK sending rate to meet the NCSC covert channel capacity guidelines [Dod] for B3/A1 classes. However, if it is desirable to send more messages (or ACK/NAK) than what the NCSC guideline specified, and the communication channel can handle this traffic, then this limitation causes a performance penalty for the communication system.

The Pump adds random noise to conventional communication methods to reduce the covert channel capacity. There have been other attempts to reduce timing channel capacity by introducing random noise to the system [CoMo91, Gra93, Hu91]. Our approach is different from the others in the sense that ours pays almost no performance penalty in the benign situation (i.e., there is no Trojan horse in the system). Our approach reduces timing channel capacity when Trojan horses attempt covert communication.

3.1 A Pump

This process can be used as a communication channel between any two security levels. Even though the Pump can reside in either source or destination, in this paper we assume that the Pump resides in the security level of the destination. This Pump needs to be trusted in the sense that the system designer has an assurance that the Pump will do only what it is supposed to do (i.e., the Pump sends to Low only ACK/NAK and does not repeat High's message). In a sense, the Pump is blocking any message flow from the destination process to the source process.

In our model of communication between high (destination) and low (source) processes, the location (i.e., either in the same computer or in two separate computers) of these two processes is not important.

The Pump has 3 basic components which work in conjunction with the (Pump independent) low and high processes to allow data to be passed from Low to High. In actuality, there is a subtle violation of Bell-LaPadula which allows a covert channel to exist. We will examine this later in the paper.

The components are the the trusted low buffer (TLB), the trusted high buffer (THB), and a communication buffer (CB). The Pump works as follows:

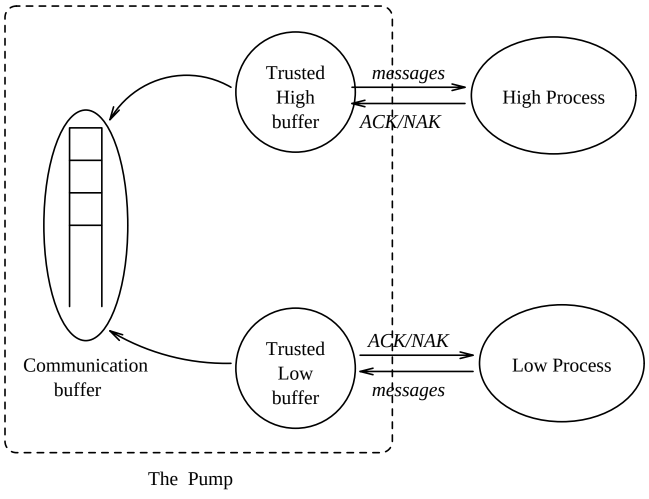

Figure 4: Message passing from Low to High using the *Pump*.

Low process: (Exterior to the Pump)
Low sends a message to the TLB and waits for an ACK from the TLB. Once an ACK arrives, then the message is removed from the low process (i.e., do the garbage collection) and a new message is sent (i.e., if the low process receives NAK or no response (i.e., time-out) then it will retransmit the same message). Note that Low may prepare a new message while Low waits for an ACK/NAK.

Trusted low buffer:
When a message arrives from the low process, the TLB inserts the message in the CB and then sends an ACK to the low process if the insertion is successful (i.e., there is space in the CB). Since the Pump is configured as a high process, this sending ACK violates the Bell-LaPadula constraints and a Trojan horse can exploit this procedure.

High process: (Exterior to the Pump)
When High receives a message from the THB, it stores the message and then sends an ACK/NAK to the THB.

Trusted high buffer:
This process sends a message to the high process if there is a message in the CB. Once an ACK arrives from the high process, the message is removed from the CB. If the THB receives NAK or no response (i.e., time-out) then it will retransmit the same message. Since the Pump is configured as a high process, this does not violate the Bell-LaPadula constraints.

Communication Buffer:

This is a regular FIFO buffer whose length is n. This buffer is shared between the TLB and the THB. It is also possible for the the TLB and the THB to learn certain statistical information from the CB (e.g., the average response time of the high process).

It is easy to see that any process that communicates with the Pump can collect garbage because this process receives ACK. Also, any communication with the Pump is reliable due to ACK/NAK being sent. If the sender receives either NAK or is timed out, then it will retransmit the same message. In the following, we show other desirable features of the Pump.

This communication method is also recoverable if we implement the CB in non-volatile storage and each message has an associated message number. We consider the following four cases:

case 1: *The system crashes after Low sends a message to the Pump but before the Pump receives it.* Since Low never receives an ACK, it will resend the message as the system recovers.

case 2: *The system crashes after the Pump receives a message but before Low receives an ACK.* Since Low never receives an ACK, it will resend the message as the system recovers. However, the Pump will notice that the message has already been received because of the message number. Hence, it will just send an ACK.

case 3: *The system crashes after the Pump sends a message but before High receives it.* This is similar to case 1.

case 4: *The system crashes after High receives a message but before the Pump receives an ACK.* This is similar to case 2.

Note that there may be many destination processes and many source processes that use the same Pump. However, in this paper, we just consider the case of one source process and one destination process which will have the worst case covert channel capacity. We have been denoting these two processes of interest as simply High and Low. Further, when we perform channel capacity analysis we assume that there are no NAK's. This makes the analysis easier but does not affect the capacity bounds.

The time from when Low sends its i-th message to the TLB until it receives an ACK back from the TLB is given by L_i. If the CB has space on it when the TLB receives the i-th message from Low then L_i is deterministic and simply equal to the fixed communication overhead time O_l. If the CB is full then the response time L_i is probabilistic and is given by the random variable S added onto the overhead O_l. S is the amount of time that the TLB has to wait, when the CB is full, for High to remove a message (by sending an ACK to the THB) so that the TLB can insert its i-th message into the CB. Note that it is possible for the TLB to attempt to insert a message into the full CB after the THB has sent a message to High and while the THB is still waiting for an ACK back from High. Therefore, the S values are less than or equal to the amount of time that the THB waits (the High ACK time). The distribution for S can be discrete, continuous, or mixed. Without loss of generality, we assume in this paper that it is discrete. Note, it is the ability of a Trojan horse to affect S that gives rise to a covert timing channel. In summary,

$$L_i = \begin{cases} O_l, & \text{if } \mathrm{CB}\emptyset, \\ O_l + S, & \text{if } \mathrm{CB}f. \end{cases}$$

where CB$\emptyset$ represents the event that the CB has space on it, whereas CBf represents the event that the CB is full. By using conditional probability we can summarize the above by expressing the behavior of L_i, when the CB has space for a message, by $P(L_i \leq t \mid \mathrm{CB}\emptyset) = P(O_l \leq t)$, which is 1 for $t \geq O_l$, (i.e. this is just a univalued discrete random variable), and the behavior of L_i, when the CB is full, is given by $P(L_i \leq t \mid CBf) = P(S + O_l \leq t)$. Admittedly, this is a rather cumbersome way of looking at such simple discrete random variables (one would normally look at mass functions instead of distribution functions). However, in section 4, when we adjust the distribution L_i, it is advantageous to view probabilities in this continuous manner.

3.2 Covert Timing Channels

A Trojan horse can exploit the present situation and create a covert timing channel. The Trojan horse controls when Low (via the low Trojan horse) sends a message and controls when High (via the high Trojan horse) sends an ACK back to the THB.

- The low Trojan horse fills the CB; this is done by having the high Trojan horse not remove messages from the CB. Now that the CB is full, there is a noiseless covert timing channel that exists between Low and High. Furthermore, this channel exists as long as the CB is full.

- Now Low sends a message to the TLB. The TLB cannot send an ACK back to Low until a spot opens up on the CB. This is totally in the control of High. We assume that $\epsilon \approx O_l$ is the smallest amount of time that High can remove a message from the Pump and for Low to get an ACK after it has sent a message to the TLB. Since the high Trojan horse knows the size of the CB (i.e., n) and how fast the low Trojan horse can send a message, High knows that Low has filled the CB and has just sent a new message to the TLB. If Low gets an ACK at time ϵ, Low interprets the signal as a zero. We further assume that 2ϵ is the next amount of time that High can remove an item from the CB and for Low to get an ACK. Therefore if the ACK to Low is at 2ϵ then Low will interpret the symbol being passed by High as a one. Since every time Low receives an ACK, the CB is full again, and Low can then attempt to insert its new message and High can send the binary symbols again. There is no noise in this channel.

We are looking at a worst case scenario with this example. High will try to send symbols as quickly as possible, hence the time values of ϵ and 2ϵ. The time units of our system are such that ϵ is an integer, i.e., ϵ is an integer number of system clock ticks. The channel capacity of this channel is given by

$$C = \limsup_{k\to\infty} \frac{\log N(k)}{k} \text{ bits per clock tick}$$

Where the logarithms are base two and $N(k)$ is the number of distinct sequences of zeroes and ones that High can send that take a total of time k. It can be shown [Sh48, MiMo93, Mo93] that $C = \log \omega$, where ω is the positive root of $x^{2\epsilon} - x^{\epsilon} - 1 = 0$. The polynomial arises from the recurrence relation $N(k) = N(k-\epsilon) + N(k-2\epsilon)$. By changing variables and letting $y = x^{\epsilon}$ we see that ω^{ϵ} is the positive root of $y^2 - y - 1 = 0$. Therefore, $\omega = (\frac{1+\sqrt{5}}{2})^{1/\epsilon}$, so $C = \epsilon^{-1} \log \frac{1+\sqrt{5}}{2}$.

In fact, there is nothing to limit the communication channel to just two symbols. High could send the symbols $\{b_1, b_2, \ldots, b_z\}$ by having the symbol b_i correspond to a response time of $i\epsilon$. In this case, the capacity is $\log \omega$ where ω is the positive root of $x^{z\epsilon} - x^{(z-1)\epsilon} - \cdots - x - 1$. As the number of distinct symbols increase from 2 to ∞, it can be shown that the positive root monotonically increases from $(\frac{1+\sqrt{5}}{2})^{1/\epsilon}$ to $2^{1/\epsilon}$, hence the capacity monotonically increases from $\epsilon^{-1}\log(\frac{1+\sqrt{5}}{2})$ to ϵ^{-1}. (The mathematical details to the above can be found in [Mo93].) Hence, by increasing the number of symbols from 2 to an infinite amount we can achieve an almost 50% increase in channel capacity. Of course, in practical usage, there is a limit to how long High will delay a response to Low, so the capacity of ϵ^{-1} can be interpreted as a worst case upper limit. It is this capacity that we shall attempt to "beat" by the other means that we will discuss in the rest of the paper. We summarize this by (since $O_l \approx \epsilon$)

$$\text{Worst Case Capacity Bound } = \frac{1}{O_l} \text{ bits per clock tick.} \tag{1}$$

Note that at present we have not added any security techniques to the Pump. When we do add these techniques we will see that we can substantially lower the channel capacity.

3.3 Performance/Security Goals

There are three cases that can limit the performance of this communication channel:

1. *Low is the bottleneck.* For example, the message sending rate of Low is slower than the rate that the Pump and High pass and receive messages. If this happens, the CB in the Pump can never be full. Consequently, no covert channel can exist.

2. *The Pump is the bottleneck.* For example, the message passing rate of the Pump is slower than the rate at which Low and High send and receive messages. If this happens, the CB in the Pump can be full. However, the time that Low receives ACKs does not reflect the response time of High because the ACK time from High is always faster than the ACK time to Low. Hence, no covert channel exists.

3. *High can be the bottleneck.* For example, the message receiving rate of High *can* be lower than the rate at which Low and the Pump can send and pass messages. If this happens, assuming that High and Low are Trojan horses, a covert timing channel can exist.

The use of better software or hardware may solve the first or the second case, but this is the beyond the scope of this paper. This paper focuses on the third case, where a covert timing channel can exist. Hence, the rest of this paper assumes that High is the bottleneck of this communication channel and Low and the Pump can send and pass messages much faster than High accepts them.

We wish to have a performance/security requirement that involves three goals. The first two attempt to mitigate the capacity of a timing channel, as described above, and the third goal attempts to make efficient use of system resources.

1. Prevent the CB from becoming/staying full.
2. Minimize the influence of High's actions on L_i.
3. Minimize system resources waiting in an idle state. For example, we would like to avoid the situation such as the Pump not sending an ACK, even though there is a space in the CB and Low is ready to send next message, in order to prevent a covert channel.

Often, in the security community, we are faced with making a trade-off between security and performance. We propose a method that approximately achieves all the above goals. We will do this by modifying the distribution L_i, but first let us formalize our performance/security requirement as follows.

Let $\bar{L_i}$ denote the mean (expectation) of the random variable L_i. When Low submits its i-th message to the TLB, we consider the last m messages that High has ACK'ed. We take the average of these m ACK times and denote it by $\bar{H_{m_i}}$. The term $\bar{H_{m_i}}$ is a moving average and the window size m should be chosen large enough to be somewhat insensitive to individual fluctuations of High but yet still able to reflect the current system workload.

We state our performance/security requirement as follows

$$\bar{L_i} \approx \bar{H_{m_i}} \; . \tag{2}$$

If we can get our system to obey Eq. (2) then we see that the CB is used in an efficient manner and any attempts of a Trojan horse speeding up Low and slowing down High, in order for the CB to become/stay full are mitigated by the approximate equality of the two "averages" in Eq. (2). (Note that $\bar{L_i}$ is an actual mean, whereas $\bar{H_{m_i}}$ is a numerical moving average.) More precisely, let us look at the two cases where Eq. (2) is not met.

Case 1: $\bar{L_i} > \bar{H_{m_i}}$
Since we assume that Low and the Pump can handle messages faster than High, the above condition implies that the Pump intentionally delays the ACK for message i, possibly to prevent covert channels. If this holds then, on the average, High is removing messages from the CB faster than Low puts them into the CB. This will result in High waiting for messages and Low waiting for an ACK to send the next message. In this case, there will be no covert channel. However we are wasting resources in the sense that the Pump is not fully utilized, and High and Low wait for either messages or ACKs.

Case 2: $\bar{L_i} < \bar{H_{m_i}}$
If this holds then, on the average, Low is sending messages to the buffer faster than High removes them. Unless the CB size is infinite, this will result in the CB becoming full. Once the CB is full then L_i becomes the same as High's response time. Also, if the CB becomes full, then a covert timing channel can be exploited as discussed previously.

Therefore, the only option left to us that both decreases exploitation of a covert timing channel and at the same time does not waste resources is Eq. (2). Note that we are allowing a slight "fudge" factor in our performance requirement due to practicality, and we use an approximate equality. The degree of "fudge" will be determined in practice.

Note that the system which obeys Eq. (2) pays almost no performance penalty in the benign situation. However, once Trojan horses decide to slow down High's response, to send signals to Low, then $\bar{H_{m_i}}$ will be increased and the capacity of the covert timing channel will be decreased. A system that comes close to meeting Eq. (2) assures one that the system is not a secure brick or a leaky vault.

4 Noisy Channels

The system which has a covert timing channel can nontheless be utilized if we understand the nature of the channel and make the channel noisy enough so that the channel capacity is less than a certain threshold. In this section, we introduce random noise to the Pump by modifying the TLB's response time to Low and analyze the covert channel capacities. The modification of the TLB's response time will be done by adding a probabilistic effect to L_i. Now, the ACK time to Low is given by

$$\text{ACK time} = \text{Old } L_i + A$$

where A is a random variable with mean $\bar{A}$. For the rest of this paper, we will denote this modified ACK time by L_i. In the benign case that we described in section 3.1, $A(t) \equiv 0$. A must be chosen to increase security without killing performance. By conditioning on whether or not the CB is full we obtain

$$\begin{aligned} P(L_i \leq t) &= P(L_i \leq t \mid \text{CB}\emptyset)P(\text{CB}\emptyset) + \\ &\quad P(L_i \leq t \mid \text{CB}f)P(\text{CB}f) \end{aligned} \tag{3}$$

By setting $P(\text{CB}f) = \mu$ and noting that we have $P(L_i \leq t \mid \text{CB}f) =$ $\sum_k P(L_i \leq t \mid \text{CB}f, S = s_k)P(S = s_k \mid \text{CB}f)$, since the events $\{S = s_k\}$ are disjoint, we see that Eq. (3) simplifies to

$$\begin{aligned} P(L_i \leq t) \;=\; P(L_i \leq t \mid \text{CB}\emptyset)\;(1-\mu) + \\ \sum_k P(L_i \leq t \mid \text{CB}f, S = s_k)P(S = s_k \mid \text{CB}f)\;\mu \end{aligned} \tag{4}$$

Since, by definition S is only defined when the CB is full, Eq. (4) reduces to

$$\begin{aligned} P(L_i \leq t) = (1-\mu)P(L_i \leq t \mid \text{CB}\emptyset) \;+ \\ \mu \sum_k P(L_i \leq t \mid S = s_k)p_k \end{aligned} \tag{5}$$

However, $P(L_i \leq t \mid \text{CB}\emptyset)$ is just $P(O_l + A \leq t) = P(A \leq t - O_l)$ and $P(L_i \leq t \mid S = s_k) = P(s_k + O_l + A \leq t) = P(A \leq t - s_k - O_l)$.

By taking the derivative of both sides of Eq. (5) and denoting the density function of L_i by $f_{L_i}(t)$ we arrive[3] at

$$f_{L_i}(t) = (1-\mu)f_A(t - O_l) + \mu \sum_k p_k f_A(t - s_k - O_l)$$

The mean wait for Low is

$$\begin{aligned} \bar{L_i} &= \\ \int_{-\infty}^{\infty} t &\left((1-\mu)f_A(t-O_l) + \mu \sum_k p_k f_A(t - s_k - O_l)\right) dt \end{aligned} \tag{6}$$

[3]In general, if c is a constant, we have that $P(c + A \leq t) = P(A \leq t - c) = \int_{-\infty}^{t-c} dA$, where dA is the measure associated with the random variable A. Note that $\frac{d}{dt}P(c+A \leq t) = f_A(t-c)$, if $dA = f_A(\tau)d\tau$. (We assume that the density function $f_A(\tau)$ of A always exists.)

We denote the mean of S, which is $\sum_k p_k s_k$, by $\bar{s}$. By simplifying Eq. (6) we obtain[4]

$$\begin{aligned} \bar{L_i} &= (1-\mu)(\bar{A} + O_l) + \mu \sum_k p_k(\bar{A} + s_k + O_l) \\ &= (1-\mu)(\bar{A} + O_l) + \mu\bar{A} + \mu \sum_k p_k(s_k + O_l) \\ &= (1-\mu)(\bar{A} + O_l) + \mu(\bar{A} + \bar{s} + O_l) \end{aligned}$$

and finally

$$\bar{L_i} = \bar{A} + O_l + \mu\bar{s} \tag{7}$$

Eq. (7) intuitively makes sense. If the CB is never full ($\mu = 0$) then $\bar{L_i} = \bar{A} + O_l$. If the CB is always full ($\mu = 1$) then $\bar{L_i} = \bar{A} + \bar{s} + O_l$.

We wish to choose A so that Eq. (2) is satisfied. This forces upon us the condition that

$$\bar{A} = \bar{H}_{m_i} - O_l - \mu\bar{s} \tag{8}$$

Note that some fixed overhead, O_h, is already included in $\bar{H}_{m_i}$. If for some reason $\bar{A} \leq 0$, we instead default $\bar{A}$ to a small value.

In the following sections, a specific random variable A will be chosen. Based on this A, the covert channel capacity of the Pump will be analyzed.

4.1 Choice of a Random Variable

We see, from the above discussion, that is it possible to add noise, via the random variable A, to L_i so that the performance/security requirement is met. We will now discuss an explicit choice of A and see how it affects the ability of High to communicate covertly, over a timing channel, to Low. The density function of the random variable that will be chosen should have the following two properties:

1. *The mean of this random variable should be controllable.* The density function should be sensitive to system feedback, in order to meet the performance/security requirement.

2. *There should be no upper bound.* If the support of the density function has an upper bound, then the upper bound can be exploited by Trojan horses. For example, if the uniform distribution is chosen, then A will be uniformly distributed between O_l and $2\bar{A} + O_l$. Hence, if the high Trojan horse decides to send a signal by sending an ACK after $2\bar{A} + O_l$, then the signal is delivered without any noise.

[4] $\int_{-\infty}^{\infty} t f_A(t-c)dt = \int_{-\infty}^{\infty} (u+c)f_A(u)du = \bar{A} + c$

Even though there are many random variables that satisfy the above properties we have chosen the exponential distribution because the capacity of the covert channel is relatively easy to analyze (due to the relatively simple density function). In fact, the exponential distribution has been well studied in other security work [Mo91, MoMi92a, MoMi92b] and is the basis of much work in queuing theory. The following is an actual implementation of the exponential distribution in the Pump.

5 A Noisy Scheme

First, we would like to consider a scheme that pays very little performance penalty. (By this we mean deviation from Eq. (2), erring on the side of system performance more than that of security).

To make this channel noisy, we consider the following scheme:

- The CB of the Pump computes $\bar{H_{m_i}}$ as a moving average for the last m values of High's ACK time to the THB.
- The distribution for A is given by the exponential random variable with density function

$$f(t) = \begin{cases} \lambda e^{-\lambda t}, & \text{if } t \geq 0, \\ 0, & \text{otherwise.} \end{cases}$$

This means that when the CB has space on it, L_i has a conditional density function given by

$$f(t) = \begin{cases} \lambda e^{-\lambda (t-O_l)}, & \text{if } t \geq O_l, \\ 0, & \text{otherwise.} \end{cases}$$

and when the CB is full and $S = s_k$, the conditional density is

$$f(t) = \begin{cases} \lambda e^{-\lambda (t-(s_k+O_l))}, & \text{if } t \geq s_k + O_l, \\ 0, & \text{otherwise.} \end{cases}$$

These above conditional densities are obtained by taking the convolution of the exponential density with a density function of the from $\delta(t-\alpha)$, α a constant (see footnote 3). We refer to any random variable that has a density function of the form $\lambda e^{-\lambda(t-\alpha)}, t \geq \alpha$ as a modified exponential distribution with shift α. The intuition behind the modified exponential distribution is that it decays just like the exponential density; however, the decay starts at α instead of 0. Of course a modified exponential distribution no longer has the (defining) memoryless property of the exponential distribution.

Note that the mean of A is $1/\lambda$ which we set, from Eq. (8), equal to $\bar{H_{m_i}} - O_l - \mu\bar{s}$. Of course in practice we would eventually have to bound the tail on the exponential distribution but this bound need not be a function of its mean, as it must be for the uniform distribution. So there is eventually a time out, perhaps quite large, such that Low finally does receive an ACK. This prevents the existence of a (theoretically possible) dangling message.

5.1 Analysis of Timing Channel Capacity

We see that our system satisfies the performance/security requirement. However it does much more. Our system is controllable due to the feedback to L_i, via the ability to change λ, due to the changes in $\bar{H_{m_i}}$. Assume, for example, that High wishes to covertly signal Low. There are two methods; one is the noiseless channel approach as described in section 3.2 and the other is the noisy channel approach. Let us try to get some quantitative bounds on the capacity for both methods. We assume in this analysis that $m \approx n$. In future work, we hope to remove this restriction and see if even further capacity reductions can be obtained.

As before High will attempt to signal Low by affecting the values of L_i. Say High tries the strategy that we discussed earlier of letting the CB get full and then removing messages within time ϵ or 2ϵ. Two factors make this an unfeasible Trojan horse strategy. The first is that High cannot get the CB full and keep it full without a severe time penalty being enacted upon L_i. This is because for the CB to become full, High must be removing messages at a slower rate than Low is getting ACKs back from the TLB. But after a certain number of messages the slow rate of High is manifested by forcing L_i to also slow down due to the moving average construction of λ. There are three basic problems with this approach. One is the noise that is involved when High tries to send a symbol to Low. The second is the time involved in sending the symbol due to large delays by High necessitated by High trying to send a symbol with as little noise as possible. The third is synchronization problems between High and Low. By this we mean the ability of Low to differentiate, via L_i values or the number of messages ACK'ed, between when High is getting ready to send a message (i.e., letting the CB get full) and when L_i is the actual symbol being passed by High. Let us consider three possible exploitations below.

Exploitation strategy 1:
High acts quickly (ACK time $= \epsilon$) m times. This has the effect of lowering the moving average and thus speeding up the L_i values. Now High does not send an ACK for $t = m\epsilon$ in the hopes of Low filling the CB. When Low finally does receive this delayed L_i value it is interpreted as a synchronization signal from High to Low — This means that the next L_i value is to be interpreted as a symbol being sent by High. The next High ACK time, via the S value chosen by High, is chosen so as to send a symbol to Low. However, due to the probabilistic nature of L_i and the fact that the CB may not even be full, this symbol is quite ambiguous. Note that now L_i is large because of the previous High delay of $t = m\epsilon$.

High wishes for the CB to become full again so that it can again send a symbol with as little noise as possible, so High repeats the above process of lowering L_i by acting quickly and then delaying and finally sending a symbol. We see that if a symbol is sent noiselessly it would take at least $t = (2m+1)\epsilon$. Therefore $C < \frac{1}{(2m+1)\epsilon}$.

Exploitation strategy 2:
Instead of High repeating the process of — filling the CB, delaying, and filling the CB again — after High sends the first symbol, it continues to send symbols. However, if High ACKs a message quickly to try to send Low small S values it will, in fact, end up only emptying out the CB and thus will not be able to send Low different S values. This is because the L_i values are, at this point, very large due to the effect of the previous large delay by High. Therefore, High must wait at least $t = m\epsilon$ and then the additional s_k times to attempt to send Low a symbol that is not too noisy. However, High's waiting this long to send an ACK has the side effect of keeping the moving average large. Therefore, all High is doing is sending noisy symbols in a very slow manner. Therefore, a worst case analysis would still have $C < 1/m\epsilon$.

Exploitation strategy 3:
High could attempt to send information to Low by simply affecting the moving average and having Low interpret its response times without High trying to make the CB full. A full analysis of this scenario is quite complicated and involves channels with continuous outputs (waveform analysis) which, up to now, have not been studied by the security community. Also there are severe practical coding issues when one quantizes the output space into many symbols. So even though a true capacity upper bound could be obtained, it would be impossible to build the proper code. So from a practical standpoint one could study the capacity just through finite decoding schemes (this is not to say that one should not see how the capacities differ). We can make some qualitative statements about the channel capacity based on present techniques. Low's ACK time is a modified exponential distribution with shift O_l. All High can do is to alter the mean, $1/\lambda + O_l$. Let us look at a simplifying example where High tries to send a symbol to Low by varying λ between two values. Low receives a response and wants to decide whether it came from a modified exponential distribution with mean $1/\lambda_1 + O_l$ or whether it came from a modified exponential distribution with mean $1/\lambda_2 + O_l$. If $\lambda_1 \approx \lambda_2$ then it is hard to make this decision and the symbol is very noisy. To make the symbol less noisy would require High to enlarge the difference between λ_1 and λ_2; this, however, would also increase the time that Low receives the symbol and in fact increase the time that Low receives future symbols due to the moving average construction of λ. Therefore we decrease the noise with which symbols are sent only by penalizing the time cost with which they are sent. Between the fidelity criterion of the symbols forcing a large difference between the λ values and the fact that the moving average moderates any change of High by a factor of $1/m$, we feel that $C < 1/m\epsilon$ is still a valid worst case bound.

In fact one could use a combination of the above exploitation strategies. However, we do not see any order of magnitude improvement by doing this.

We see that the size of the CB, n, is very important to the security of the channel. In the future, through both analytic techniques and simulations, we hope to obtain rules of thumb that a system designer could use to lower the capacity to within specified bounds. As in section 3.2, since $O_l \approx \epsilon$, and $m \approx n$, being conservative we may state

$$\text{Worst Case Capacity Bound with Noise added to the Pump} = \frac{1}{nO_l} \tag{9}$$

Thus far, the mean of the exponential distribution was a function of $\bar{H}_{m_i}$. However, if one wishes to reduce the channel capacity further, λ can be chosen not only as a function of $\bar{H}_{m_i}$ but also as a function of the current state of the CB. For example, if the CB is 80% full then λ may be a function of $2\bar{H_{m_i}}$, if the CB is 90% full then λ may be a function of $3\bar{H_{m_i}}$, and so on. This will have the effect of slowing down L_i when High tries to send covert signals with very little noise.

6 Summary

In this paper, we introduced the Pump. It is a generic communication mechanism in the sense that it can be used to pass messages between any two different security levels. The Pump has all desirable features (i.e., reliability, reasonable performance, garbage collectibility, recoverability, and practicality) of conventional communication mechanisms. At the same time, the covert channel capacity of the Pump is less than $\frac{1}{n}$ times that of the conventional communication mechanisms (by comparing Eq. (1) to Eq. (9)), where n is the buffer size.

The result that is presented in this paper is the worst case channel capacity. Our future plans are: (1) to tighten the bound of covert channel capacity, and (2) to provide the covert channel capacity as a function of n and m so that the system designer can choose the values, n and m, according to his/her security requirements.

7 Acknowledgements

We wish to thank Ruth Heilizer, Carl Landwehr, and John McLean for their time and helpful suggestions.

References

[BeL76] Bell, D. E., and LaPadula, L. J. Secure computer systems: Unified exposition and multics interpretation. The Mitre Corp. (1976).

[CoMo91] Costich, O. and Moskowitz, I. S. Analysis of a storage channel in the two-phase commit protocol. The Computer Security Foundations Workshop 4 (1991).

[Gra93] Gray, J. W. On introducing noise into the bus-contention channel. IEEE Symposium on Research in Security and Privacy (1993).

[Hu91] Hu, W. M. Reducing timing channels with fuzzy time. IEEE Symposium on Research in Security and Privacy (1991).

[MiMo93] Miller, A. R. , and Moskowitz I.S. Reduction of a Class of Fox-Wright Psi Functions for Certain Rational Parameters. Submitted for publication (1993).

[Mo91] Moskowitz, I. S. Variable noise effects upon a simple timing channel. IEEE Symposium on Research in Security and Privacy (1991).

[MoMi92a] Moskowitz, I. S., and Miller, A. R. The influence of delay upon an idealized channel's bandwidth. IEEE Symposium on Research in Security and Privacy (1992).

[MoMi92b] Moskowitz, I. S., and Miller, A. R. The channel capacity of a certain noisy timing channel. IEEE Transactions on Information Theory, 38, 4 (1992)

[Mo93] Moskowitz, I. S. and Miller, A. R. Simple Timing Channels. Preprint (1993).

[Dod] Department of Defense, National Computer Security. Trusted Computer System Evaluation Criteria 5200.28-STD (1985).

[Sh48] Shannon, C., and Weaver, W. The mathematical theory of communication. University of Illinois Press (1948).

[Sch77] Schwartz, M. Computer-communication network design and analysis. Prentice Hall (1977).