DESIGN OF A CERTIFIABLE ONE-WAY DATA-FLOW DEVICE

Ronald L. Sharp

Ronald L. Sharp *is a member of technical staff in the Processor and Software Applications Group at AT&T Bell Laboratories, Whippany, New Jersey. He works primarily on communications security research, with emphasis on multilevel secure networks. Before joining AT&T in 1986, he was an officer in the U.S. Air Force working on intelligence and security-related projects. He received a B.S. in computer science from the University of Arkansas and an M.S. in computer science and systems design from the University of Texas.*

This paper describes a possible design for a device that would allow two hosts or networks of different security sensitivities (for example, Top Secret and Secret) to be connected so that information can move from the low-sensitivity system to the high-sensitivity system. The primary challenge was to allow return protocol messages from the high-sensitivity system while not permitting an unacceptable covert channel. Covert channels, as they relate to intersystem communication, are described to give the reader a better idea of the problem.

Introduction

Suppose there are two networks: one connects systems containing classified information, and the other connects systems containing only unclassified information. Further, suppose that users on the classified network routinely require information from a system on the unclassified network. The two networks cannot be directly connected by standard bidirectional communications channels because that might allow classified information to flow to the unclassified network.

A common solution to this problem is to dump the information from the unclassified network to tape and load it on the classified network. This seems to be a crude solution for an operation that does not violate common security principles.

The problem is not that the information cannot be sent from the unclassified system. Instead, the problem is that, given a bidirectional connection between the two networks, there is no assurance that a person or process will not try to transfer classified information from the classified network to the unclassified network.

With the advent of multilevel security (MLS) technology, there are now several automated solutions to this problem. A "trusted" system such as the Multics or SCOMP[1] system (trademarks of Honeywell Inc.) could be placed between the two networks to ensure, with reasonable confidence, that no information goes from the classified, or high-sensitivity (HS), network to the unclassified, or low-sensitivity (LS), network. These MLS systems are usually extremely expensive and rel-

atively slow in performance when compared with other systems on the market that use current technology. Most situations do not justify the expense of a large MLS system that will provide many features not needed to solve this simple problem.

In addition, these large MLS systems have much of their trust placed in software. As pointed out in Reference 1, this trust may be misplaced. No amount of source-level verification will be effective if a "Trojan horse" already exists in the object code of a program-handling program such as a compiler, assembler, loader, or even hardware microcode.

Periodic code verification will be partially effective. However, that verification procedure is likely to be software-based and thus also vulnerable to attack. This does not mean that software should not be trusted, but it should be understood that, with the flexibility that software offers, there is a heavy price in proving and maintaining integrity. Until there is run-time code validation (beyond the A1 level[2]) there will always be some doubt about the exact implementation of a security policy.

Several special-purpose communications systems are being developed.[2] However, they are environment-specific, and, like general-purpose MLS systems, they are usually software-based. This paper proposes a design for an environment-independent device that could act as a one-way gateway between two systems. The word *system* is used synonymously for a single computer and a network. All trusted functions are implemented directly in hardware. The name given to this device is the *data diode.*

Goals. The goals for the design study were to:

1. Design a one-way data transfer device that would:
   - Apply to many environments
   - Transfer data at speeds of 1 to 5 megabits per second (Mb/s)
   - Meet all applicable *Orange Book* requirements[3]
   - Provide highly reliable communication.
2. Identify and control all covert channels
3. Use standard devices and technology to reduce cost
4. Minimize trusted processes and implement them directly in hardware.

Problems. There were many hurdles to clear to meet these goals. They included:

1. Removing any overt channels that would, by design, permit information flow from the HS system to the LS system
2. Allowing protocol exchange to ensure data integrity while guaranteeing that no information is contained in, or represented by, the protocol from the HS system
3. Identifying, measuring, and deleting or minimizing all covert timing and storage channels
4. Reducing the overhead of security controls in order to increase throughput
5. Providing a common hardware and software interface that will be compatible with most systems currently on the market
6. Developing a security model and proving that the hardware implementation is consistent.

Overt and Covert Channels

An overt channel is any legitimate path to transfer information using system tools or resources in the manner for which they were designed. (*Legitimate,* within the scope of this paper, refers to actions, methods, and states that do not violate the system's security policy.)

A covert channel is a direct or indirect communications channel used to write (transmit) information in a manner inconsistent with the system's or network's security policy. Much has been written about covert channels.[4-8] This paper briefly describes the covert channels that relate to the problem of one-way data flow between systems of differing sensitivities.

It should be understood that it is effectively impossible to eliminate all covert channels where information is bidirectionally exchanged between systems. The best we can hope to achieve is to lower the bandwidth of the covert channel to such a degree that a would-be spy is forced to find other means to obtain the information. The other solution, although not as effective, is to monitor all communications, detect when covert information is being passed, and hopefully catch the perpetrator before too much information has leaked out.

The sections that follow include examples of covert channels as they relate to intersystem communications. They assume that there is a return protocol line and that the overt channel problem is solved and only protocol mes-

sages are being transmitted on that line.

Storage Channels. Storage channels are covert channels in which the information to be secretly transferred depends on changes in a characteristic of a shared physical resource. For a storage channel to exist, two processes must have concurrent access to a common resource, and a characteristic or parameter of that resource must be alterable by the transmitting process and readable or detectable by the receiving process.

One example of a storage channel is one in which a process alternately holds and releases a resource to represent a 1 and a 0, respectively, to a process at a lower level that is trying to grab the same resource. The information is actually represented by a flag in memory that indicates whether the file is available or not.

Previous methods of eliminating covert storage channels such as the one described in Reference 5 will not be effective for this problem. Reference 5 points out that, to remove the storage channel, one need only prevent any two processes of different sensitivities from sharing a resource at a given time. This is a reasonable method and is practical in many environments. For the one-way data flow problem, however, the two processes must share the same resource (the communication line) to perform their intended function.

Minimizing the variance of the shared parameters of a resource will reduce the effectiveness of the covert storage channel. If this is taken to an extreme (no variance of parameters), no information is passed, legitimately or covertly. The bandwidth of this channel depends on the number of combinations of alterable parameters and how rapidly changes in the parameters may be introduced and detected. This idea is brought out further in the examples of covert storage channels that follow.

Data stuffing. Data stuffing is the direct placement of covert information in a memory location or field that was not designed to hold general information. Restricting the content of these fields to a fixed set of responses is one possible way to prevent this method of covert communication. A Trojan horse would then be forced to use the fixed set of parameter values to encode the information. The term Trojan horse is expanded in this paper to represent any program or part of a valid program that acts in behalf of an individual to circumvent the security controls of a system or network.

Parameter encoding. To prevent detection by an operator or audit system, a process could send information by alternating valid protocol parameter values. The amount of information that can be sent in a given parameter depends on the number of possible parameter values. The effectiveness of a covert channel in intersystem communication is the ratio of the number of covert information bits transmitted to the number of valid (or apparently valid) bits transmitted. Limiting the number of valid values to a given parameter will greatly reduce the effectiveness of this covert channel. The effectiveness cannot be reduced to zero, because to do so would require that all parameters have only one possible value, and that would make the response useless as a means for communicating valid information. This proves the earlier statement that, if information is to be passed between two untrusted processes, then removal of all covert channels is nearly impossible.

Attribute encoding. Another method for sending covert information over storage channels is attribute encoding. This is where some attribute of a valid message is altered to represent information. The length or any other attribute could be altered in a message to represent covert information. This covert channel could be eliminated by requiring that all attributes of the message be fixed.

Response-type encoding. Response-type encoding is just a variation of parameter encoding in which the type of message is a parameter of the message. It is distinguished here because, as explained earlier, if all variations of parameters were reduced to one (effectively deleting them), then the only variation would be in the information field. For a return protocol message, this field would give the type of the response [that is, acknowledgment (ACK), negative acknowledgment (NAK), or flow control].

It would be difficult for a Trojan horse to piggyback covert information on a valid message by alternating the type of response. Changing the type of response on a valid message will probably cause an unexpected result that will eventually be detected. The Trojan horse would need to have temporary control of the communication line

to send its own bogus packets. These packets would be removed by another horse on the LS side. Response-type encoding is not easy to guard against and is the one covert channel allowed in the proposed solution.

Timing Channels. A timing channel is a covert channel that passes information by introducing and subsequently detecting variations in a shared concept of time. In a multiprocessing system, a process does a lot of waiting. The amount of waiting usually depends on the other processes in the system. This opens the opportunity for one process to signal another by affecting the amount of time the receiving process must wait for a particular resource. If many processes are sharing this resource, the channel will be very noisy, but still effective. A high-bandwidth covert channel can be established if the resource is shared by only the sending and receiving processes.

It could be argued that a timing channel is just a storage channel in which the attribute of time can be varied. The important difference is that this attribute does not depend on the content of a message or a storage element. It is for this reason that covert timing channels are very hard to detect.

Bandwidths. The bandwidth of a covert channel, as described in this paper, will depend on several factors:

- Bandwidth of the physical line
- Effectiveness rating (percentage of covert information to total information transmitted)
- Whether a Trojan horse must piggyback information on valid messages or can generate its own bogus messages that resemble valid messages
- Valid message load on the channel.

If we assume that the Trojan horse can generate its own bogus messages, then the bandwidth of the covert channel is the product of the effectiveness rating and the bandwidth of the physical line. A 0.0625 covert channel would have an effective bandwidth of 625 kilobits per second (kb/s) on a 10-Mb/s physical line. If the Trojan horse were forced to piggyback information on valid messages, then the bandwidth would depend on the current load on the line. If a Trojan horse on the receiving side could generate messages, then it could cause the return line to operate near maximum.

The National Computer Security Center (NCSC) recommends in its *Orange Book*[3] that the bandwidth of a unaudited covert channel be no greater than 0.1 bit per second (b/s). For a 10-Mb/s physical line, this would require restricting a covert channel to an effectiveness of 10^{-8}.

Bandwidth is measured in bits, not the actual amount of information being transferred. It is typically assumed that it takes 7 or 8 bits to represent one ASCII (American Standard Code for Information Interchange) character. A smart Trojan horse will use data-compression techniques to transmit even more information on a covert channel. This leaves us with the question of whether we should consider the bandwidth of a covert channel as bits per second or the maximum amount of information that could be transmitted using those bits. For this paper, a covert channel will be measured in bits per second so that comparison with the *Orange Book* can be made.

Solutions

Now let's look at some of the possible solutions to the one-way data-flow problem that use the ideas discussed in the preceding sections. They are presented briefly to show why some ideas that first appear to work are not acceptable.

Encryption. The standard way to secure a communications line is to use link or end-to-end encryption techniques. This will prevent anyone who taps the line from obtaining the information being transmitted. However, it does little to restrict covert channels. Link encryption would be implemented after a Trojan horse has been introduced and, at best, would inject some noise into a covert timing channel because of delay. End-to-end encryption will only encrypt the information field. All other parameters and attributes are still subject to use by a Trojan horse.

No Return Protocol. Not providing a return protocol is the one solution that has no covert channels. A one-way, possibly fiber-optic, line would link the two systems. No acknowledgment would be given by the HS side that it received the message. The LS system transmitting software would have to be changed so that it does not wait for a response. A very high forward bandwidth is possible with no corresponding reverse (or covert) channel bandwidth.

This is a very clean solution that will easily satisfy all security requirements at the cost of data integrity. Successful transmission would have to be accepted on faith or acknowledged by a human operator at regular intervals. Flow control is not possible, therefore the HS system must always be ready to accept messages at the rate the LS can send them. Retransmissions for lost or garbled messages would also have to be handled manually. Forward error-correction techniques could be employed to lower the message error rate. If error rates can be kept to a minimum and a system can be dedicated to receiving messages on the HS side, this solution could be very practical and cost-effective.

Protocol Filter and Audit. An alternative method uses the idea of a trusted "box" between the HS and LS systems that would:

1. Scan the reverse protocol line to ensure that only protocol messages are being returned.
2. Monitor the forward data line to see if protocol messages correspond to messages sent.
3. Scan protocol messages for embedded information.
4. Audit protocol traffic to ensure that variances of parameters, attributes, and response types are within prestated limits. If not, the system would sound an alarm, cut the line, or just record the occurrence.
5. Monitor time between messages and segments to ensure that it does not exceed "normal" variance limits.

The effectiveness of this approach would depend on the strictness of the limits set for the potential information-carrying elements. If they are set too tightly, many valid occurrences could cause an alarm. If they are set too loosely, covert channels could be possible.

This solution would require an MLS system with trusted software and hardware. It would also be difficult to prove that all covert channels had been identified and removed or restricted to an acceptable bandwidth.

Intermediary Trusted Device. In another alternative method, an intermediary trusted device (ITD) would sit between the two systems and communicate with them. It would accept a message from the LS system, acknowledge receipt, and pass the message on to the HS system, which would in turn acknowledge receipt. This would prevent return protocol messages from going directly from the HS system to the LS system. Retransmissions could be handled by the ITD, and the LS system would never know they occurred. This is no more than the store-and-forward station found in many networks, except that this one would have to be trusted.

At least one covert channel would still exist. If the HS system refused to receive any messages, the buffers in the ITD would eventually fill up. When this occurs, the ITD would have to tell the LS system to stop sending information. To the LS system, this would represent 1 bit of information generated by the HS system. The HS system could then decide to read or not read, depending on the information to be sent, and thus a high-bandwidth covert channel could be set up.

There is no practical way to remove this covert channel. However, its capacity can be greatly reduced. When the buffers become full, a rule could be implemented by which the LS system would not be allowed to continue to send additional information until the HS side had emptied the buffers. Another bit of information could not be sent until the buffers could be filled again by the LS system. This would reduce the covert channel transmission rate to 1 bit in the time it takes the LS system to fill the buffers. This covert channel could be reduced to an acceptable level if large enough buffers were used.

This solution would also require a complete trusted system, including both hardware and software. Proof of covert channel bandwidth would not be difficult, and reliability of information would be high. This is a very good solution; however, the trusted hardware and software make it more complicated and expensive than it needs to be.

Specialized Protocol. The proposed solution, described in the next section, uses the specialized protocol approach. The method involves placing two trusted boxes between the two systems. These boxes would communicate with their respective systems using any standard protocol, and with each other using a specially designed protocol. With the new protocol, the messages would be grouped into blocks large enough that return protocol messages would come too infrequently to be used as an effective covert channel. Trusted software or hardware would be used to enforce the time limits and to ensure that no other information is passed on the return protocol line. Return protocol would be limited to a very few possible messages, such as ACK, NAK, or flow control signals.

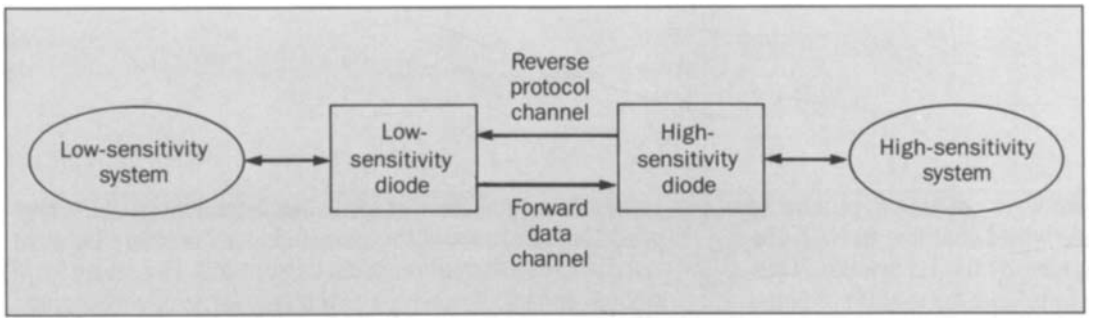

Figure 1. High-level layout for the controlled-response method.

The only other parameter would be type. All attributes would be fixed.

All covert channels could easily be identified and measured. The largest covert channel would be limited to the variations of response type and the time limit set for return messages. As discussed in the next section, "Controlled-Response Method," the response types can be limited to two, thus only 1 bit of information can be sent per response time period.

Controlled-Response Method

The controlled-response method (CRM) was chosen as the best approach for many reasons. Basically, this approach (Figure 1) offers all the integrity of the other solutions, is easier to implement, and does not require any trusted software. All trusted processes are implemented directly in hardware.

Procedure. The best way to explain the procedure is to run through an example. Suppose a user on the LS side wants to send a file to another user on the HS side. The user just sends the file to the other user's address via the LS and HS data diodes. The LS diode accepts the packets that make up the file from the sender's system. All layers of protocol on the sender's side are acknowledged; therefore it appears to the sender's system that the packets have arrived safely. The LS diode requests a retransmission from the LS system if it detects an error in the received packet.

The LS system continues to accept packets until it has 10 seconds' worth of information to pass on to the HS diode. The 10-second block of data is sent over a one-way fiber-optic cable to the HS diode. The diode checks for errors and generates an ACK if none are found or an NAK if an error is discovered. It then sends the appropriate response to a trusted clock mechanism. The clock mechanism holds this response until a multiple of 10 seconds has passed since the previous response, then regenerates the ACK or NAK, depending on what it received from the HS diode, and sends it to the LS diode.

The clock mechanism automatically generates an NAK if no response has been given by the HS diode at the end of the 10-second period. If the LS diode receives an NAK, it tells the LS system to stop sending and it retransmits the 10-second block. If it receives an ACK, it continues accepting packets from the LS system and starts sending the next 10-second block.

After receiving a good block, the HS diode breaks it up into packets and sends them to the HS system with the appropriate protocol. The HS system requests retransmission of a packet if an error is detected. The HS diode handles the retransmission.

The only way the HS system, or even the HS diode, can send covert information to the LS system is through the clock-controlled reverse protocol channel. The control clock will accept only an NAK or an ACK as input and will regenerate this signal only at 10-second intervals and send it to the LS diode.

The entire diode does not have to be trusted, only the control clock and the one-way fiber-optic forward data channel. To ensure one-way transmission only, the fiber-optic cable has transmit-only capability at the LS diode and receive-only capability at the HS diode.

Protocol. The new protocol for sending the 10-second block over the forward channel does not have to be restricted. Any protocol that can identify the beginning and end of a block and provide error detection (such as the cyclic redundancy check, or CRC) would fit the requirement. A protocol could also identify the beginning and end of an original packet so that the packets can be reassembled to be identical to the packets received by the LS diode. For many networks, however, that would not be necessary. The data could be encrypted, or more error-detection or error-correction code could be added.

The return protocol would have to be restricted to two responses, such as ACK and NAK. How these are represented is not important as long as the clock control recognizes and generates only two possible responses.

Flow control is possible using only the two

responses. If the HS system were not able to receive any more packets, the HS diode would continue to NAK the current block that is being sent by the LS system. This would cause the LS diode to tell the LS system to stop sending, and the LS diode would continually resend the same block; thus, no packets would be lost. When the HS system can resume receiving packets, the HS diode will ACK the current block and the process will continue.

Higher level requests. In the example above, only positive and negative acknowledgments and flow control were performed. What if the HS system lost many packets and wanted the file resent or a user wanted end-to-end acknowledgment of a file transfer? These and other high-level requests could be supported, but at a price. For example, a file-retransmission signal could be sent to the LS diode as a code of three NAKs followed by an ACK. The LS diode could be programmed to detect this signal and request a file retransmission from the LS system. Because of the control clock, this signal would take 40 seconds to get to the other side. However, if the file were a long one, the transfer would not be objectionable.

Other codes could be used to signal other high-level requests such as:

- Stop/start transmitting
- Acknowledge or retransmit file
- Make initial start-up line check.

The interesting thing about this capability is that a known covert channel is being used to perform a valid task. Shorter codes would reduce the time needed to send the request.

Another price for this capability is that the HS diode would have to watch for these codes in normal protocol exchange. For example, if the file-retransmission code were three NAKs followed by an ACK, and if during normal operation the HS diode sent three NAKs, then it would have to make sure that the next response sent was an NAK. This could cause some extra delay, depending on the number of special codes and the length of the codes. One NAK followed by an ACK would occur very often normally; however, five NAKs followed by an ACK would be rare in normal operation.

Windowing. One of the initial designs of the CRM solution had the control clock assure only that the responses were separated by not less than 10 seconds rather than send them at strict 10-second intervals. This would appear to solve the covert channel problem because only 1 bit of information could be sent in a 10-second period. In fact, however, a potentially large covert timing channel would still be available. A Trojan horse on the HS side could wait past the 10-second period and send the response at a particular second or microsecond past 10 seconds. An *a* could be represented by a response arriving at 10.1 seconds, a *b* at 10.2 seconds, and so on.

To solve this problem, responses were forced by the control clock to occur at set time windows past the initial 10-second period. Setting this window to 5 seconds still allowed a covert channel to operate at an average speed of 0.133 b/s using a 2-bit-per-event signaling channel. This is just over the *Orange Book* recommendation of 0.1 b/s. It was later determined that a window size of 10 seconds would be easier to implement and would not degrade performance.

This is a good time to point out that the 10-second period was chosen as the initial response separation to meet the *Orange Book* recommendation. A larger or smaller time could be used, depending on the requirements of the environment. A very sensitive environment might call for restricting covert channels even more by increasing this interval. Of course, memory requirements, described in the "Architecture" section, would also increase.

Trusted switches could be added to allow the timer period to be changed as the sensitivities of the systems change. Set to the shortest period, a switch would open the reverse line to normal protocol response periods. This could be used if the HS system were temporarily downgraded or the LS system upgraded. A switch set to the longest period would make this method functionally equivalent to the no-return-protocol method. This new control would need to be well-protected physically to prevent possible misuse.

Two-Clock Control. In the design described so far, a response would have to wait an additional 10 seconds if it missed the original 10-second window. In addition, the control clock would generate an NAK because no input was given at the 10-second mark. This would cause the LS diode to stop receiving packets from the LS system and retransmit the block. This is wasteful. For this reason, the

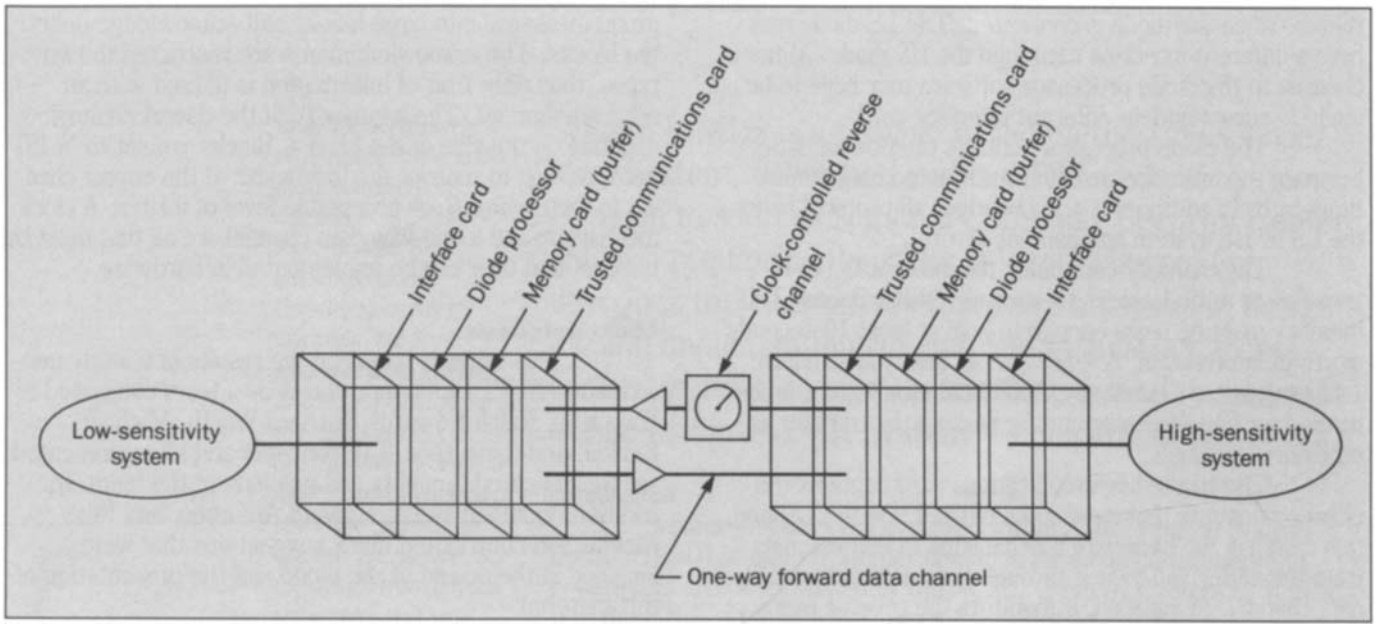

Figure 2. Data diode board-level design.

design was upgraded to ensure that the HS diode has plenty of time to respond.

The forward data channel between the diodes should be at least 20 percent faster than the line coming into the LS diode. This will allow the LS diode to send a 10-second block in only 8 seconds. The HS diode will then have 2 seconds to decide if an error has occurred and generate the appropriate response. This will allow the LS side to operate at 100 percent while no errors occur and still provide the reliability of obtaining a response before the next block of data is sent.

That solves the problem on the HS side, but what if the LS side delays sending the end-of-block or sends a block larger than 10 seconds? This could cause the HS side to miss the 10-second mark. To prevent this, a second clock was introduced. The second clock starts whenever an end-of-block is received by the HS diode. The clock goes for 2 seconds, and the clock control does not regenerate a response until both clocks have completed their sequence. This allows the HS diode at least 2 seconds to process the block, regardless of when the end-of-block is received.

This changes the initial idea that responses occur at 10-second intervals. However, it does not allow a covert timing channel. Extensions to the 10-second intervals can be caused only by the LS diode. The HS side has no control over the 2-second clock.

The 10-second clock is restarted after the response is regenerated and sent to the LS diode. An NAK is still generated if the HS diode does not respond by the time both clocks complete their sequence.

The 2-second clock would not have to be trusted. Only the device that detects an end-of-block condition would need to be trusted, to ensure that nothing on the HS side could cause an erroneous end-of-block condition. If that condition could occur, the equivalent of a 2-second window signal could allow a covert timing channel with a bandwidth of 0.1875 b/s. This is almost twice the recommended maximum bandwidth.

Architecture. Figure 2 shows a possible board-level layout of the diode device. The bus that joins these cards should be standard so that most of these circuit boards can be purchased off-the-shelf.

The choice of interface card depends on the sys-

tem to which the diode is connected. The LS diode may have a different interface card than the HS diode. Minor changes to the diode processor software may have to be made to accommodate different interface cards.

The diode processor controls the flow of data between the interface card and the trusted communications card. In addition, it acknowledges all protocol from the LS or HS system and handles errors.

The memory card holds the information to be transferred until the next 10-second window opens. This memory must be large enough to hold at least 10 seconds' worth of information. A 1-Mb/s input line would require 1.25 megabytes of memory. Additional memory would be needed for housekeeping and for making the transfer as efficient as possible.

The trusted communications card contains the only components that need to be trusted. On the LS side, this card has the hardware mechanisms to retrieve data from the buffer and pass it through the one-way forward data channel. In addition, it monitors the reverse protocol line and passes to the diode processor any response received. On the HS side, this card has the mechanisms to accept the data, check for errors, and pass the data and error status to the diode processor. In addition, it accepts protocol messages from the HS diode processor and passes them to the control clock.

The control clock resides on the same board. The 2-second clock is also included on this board. The output from the two clocks is combined and given to the device that regenerates the appropriate response.

The reverse protocol channel and the forward data channel are one-way enforced digital lines.

Summary

This paper has provided one interpretation of covert channels as they relate to the problem of trusted one-way data flow. It is effectively impossible to eliminate all covert storage channels where there is a bidirectional exchange of information. On the other hand, covert timing channels, usually considered harder to remove, could be eliminated.

A detailed description of a trusted one-way flow device has been presented. The primary concept is to group messages into large blocks and acknowledge only the blocks. The acknowledgments are restricted to two types, thus only 1 bit of information is passed with an acknowledgment. The bandwidth of the covert channel depends on the size of the blocks. Blocks are set to be 10 seconds long to restrict the bandwidth of the covert channel to the *Orange Book* acceptable level of 0.1 b/s. A clock mechanism and a one-way data channel are all that must be trusted, and they can be implemented in hardware.

Acknowledgments

This paper is based on the results of a study performed at AT&T Bell Laboratories by a team composed of Yong Kim, Ralph Edwards, Richard Wurth, Michael Becker, and the author. The concepts and ideas presented are the collected thoughts and opinions of the team. In addition, Jonathan Weiss, Edward Amoroso, and Ross McPherson contributed many suggestions that were included in the design of the diode and the presentation of this material.

References

1. K. Thompson, "Reflections on Trusting Trust," *Communications of the Association for Computing Machinery*, Vol. 27, No. 8, August 1984, pp. 761-763.
2. C. E. Landwehr, "The Best Available Technologies for Computer Security," *IEEE Computer*, July 1983, pp. 86-100.
3. *DoD Trusted Computer Systems Evaluation Criteria*, National Computer Security Center, DOD 5200.28-STD, December 1985.
4. S. B. Lipner, "A Comment on the Confinement Problem," *Proceedings of the Fifth Symposium on Operating System Principles, SigOps*, Vol. 9, No. 5.
5. K. Lopere, "Resolving Covert Channels within a B2 Class Secure System," *SigOps*, Summer 1985, pp. 9-28.
6. R. A. Kemmerer, "A Practical Approach to Identifying Storage and Timing Channels," *Proceedings of the 1982 Symposium on Security and Privacy*, Oakland, California, pp. 66-73.
7. M. Schaefer et al., "Confinement in KVM/370," *IEEE Computer*, July 1983, pp. 404-410.
8. L. K. Barker, *One Solution to Covert Timing Channels*, Master's Thesis, Texas A&M University, 1984.

(Manuscript received February 8, 1988)