HIGH-PERFORMANCE INTERFACE ARCHITECTURES FOR CRYPTOGRAPHIC HARDWARE

David P. Anderson & P. Venkat Rangan

Computer Science Division
Department of Electrical Engineering and Computer Sciences
University of California, Berkeley
Berkeley, CA 94720

1. INTRODUCTION

In general, secure communication in a distributed system that spans physically insecure networks and hosts must be implemented using cryptography. Software implementations of cryptographic algorithms such as DES are much slower than typical network bandwidths. However, fast hardware implementations of these algorithms are being developed [4, 6] and are projected to have encryption speeds comparable to network bandwidths (i.e., 10-100 megabits per second).

Current efforts at increasing the performance of hardware encryption are directed largely at increasing the speed of encryption within the device itself [5]. Less attention is being paid to the efficiency of the interface between the cryptographic hardware and the rest of the computer system.

This research was supported by the Defense Advanced Research Projects Agency (DoD), ARPA Order No. 4871, monitored by the Naval Electronic Systems Command under Contract No. N00039-84-C-0089, by the IBM Corporation, by Olivetti S.p.A., by MICOM-Interlan, Inc., by CSELT S.p.A., and by the University of California under the MICRO Program. Venkat Rangan is also supported by an IBM Doctoral Fellowship. The views and conclusions contained in this document are those of the authors, and should not be interpreted as representing official policies, either expressed or implied, of any of the sponsoring agencies or corporations.

While implementing a secure network communication system [1, 2] using commercially available components, we found that interface to the encryption device, rather than the encryption speed of the device, imposed the major limits on performance. Specifically, CPU speed was both the bandwidth bottleneck and the major source of delay, and CPU overhead was significant.

In this paper we address the problem of designing an interface to encryption hardware that removes many of the performance limitations we encountered. With such an interface, the performance of secure network communication is determined by memory bandwidth, encryption speed, and network performance. We feel that these interface considerations should influence future hardware implementations of cryptographic algorithms.

2. COMPONENTS OF NETWORK PERFORMANCE

Three components are of primary interest in evaluating the performance of network communication:

- Message latency: the interval from the time a message is generated to the time it is received by its destination process.
- Throughput: the average data transfer rate that can be sustained between processes.
- Processor overhead: the fraction of processor time spent in network communication.

File server access (virtual memory paging and access to user files) is the dominant component of network communication in current distributed systems. Low latency is critical to the performance of network file access [3], and to applications involving real-time control and user interfaces. High bandwidth is required for many applications, such as those involving graphic and audio user interfaces. Processor overhead can have a significant effect on local processing speed, and also affects the latency and throughput components.

3. A CURRENT INTERFACE AND ITS LIMITATIONS

Our secure communication system was developed using widely available hardware: Sun-3 workstations with a built-in interface for the Zilog 8068 DES encryption processor. The operation of the DES encryption processor and its interface to the Sun-3 workstation is as follows: the CPU first loads the DES chip with a key. Starting from the beginning of the data, the CPU loads an 8-byte block of data into the processor, waits for the encryption operation to finish, then removes the encrypted data from the DES chip. The CPU repeats this process until there is no more data left in the input. Once the encryption is completed, the message can be transmitted over the network.

In the above sequence of operations, data transfers in and out of the DES chip are done one byte at a time. During the encryption of a long message, the CPU is devoted entirely to operating the cryptographic hardware, either copying data or polling the status of the cryptographic hardware for completion of an operation.

Our measurements show that during the encryption of long messages, 90% of the time is spent copying data to and from the chip, and the remaining time is spent polling the chip for completion. Maximum throughput of the encryption operation alone is 2.88

megabits per second, and maximum network throughput of secure messages (including encryption, transmission, and decryption) is 2.80 megabits per second. Maximum network throughput of unencrypted messages is 7.60 megabits per second.

The way in which the DES chip is interfaced have the following implications for the various performance components:

Message Throughput:

Because of software copying of data, the maximum encryption throughput of the system is limited by processor speed. Even if encryption time were zero, the encryption throughput would be only 3.18 megabits per second.

Message Latency:

Latency of encrypted packets is the sum of latency without encryption, the time for encryption, and the time for software data copying. In our system, the total latency for an 1024 byte message is 7.60 milliseconds, of which 5.60 milliseconds is due to encryption. Of this, 5.04 milliseconds is spent in software data copying and would be present even if encryption time were zero.

Processor Overhead:

Under a communication workload taken from traces of a real system (a heavily-used file server), the CPU overhead due to communication is 47.0% with encryption and 20.6% without. Of this, 42.0% is due to software copying and would be present even if encryption time were zero.

Therefore, the limitations on the performance of secure network communication in this system are imposed by the way in which the encryption hardware is interfaced, rather than by the speed of the encryption hardware.

4. PROPOSED FEATURES OF ENCRYPTION HARDWARE INTERFACES

We propose the following hardware architectural features for the interface between any cryptographic hardware and rest of a computer system.

4.1. Interface to Main Memory

Software data copying should be avoided. This can be achieved either by integrating the cryptographic hardware with the network interface (discussed in the next section) or by providing a *direct memory access* (DMA) interface to the cryptographic hardware. If the network interface hardware is fixed, only the latter alternative is possible. Whether or not DMA is used, the width of the data interface should be a word (32 or 64 bits) instead of a byte.

A DMA interface would work as follows: the CPU loads the interface with the start and end addresses of a data area in memory, and instructs it to begin an encryption, decryption, or cryptographic checksumming operation. During the operation, the interface fetches data during memory cycles ''stolen'' from the CPU. The CPU is free to do other work during encryption. The interface interrupts the CPU after completing the operation.

If the memory bandwidth is high enough to support the demands of both the CPU and the cryptographic hardware, the CPU and cryptographic hardware can operate at full speed in parallel. In this case the CPU overhead is essentially eliminated, and throughput is limited by memory bandwidth, encryption speed, and network bandwidth, rather than by CPU speed. If memory bandwidth is not this high, CPU operation is slowed by encryption DMA, but there will still be improvements in throughput, latency, and CPU overhead relative to software copying. The DMA technique is inherently limited by memory speed; if encryption speed is significantly greater, other approaches are required.

4.2. Pipelined Operation

The operation of the cryptographic hardware should be *pipelined* with that of the network interface, so that the encryption of a long message is overlapped with its transmission. In a non-pipelined system, the latencies for encryption and decryption are added directly to the total latency. In a pipelined system, the latency due to the combination of encryption and transmission is the maximum of the two latencies, rather than their sum. The same applies for reception and decryption.

This pipelining can be achieved in several ways. First, if the network interface and the cryptographic hardware are independent DMA devices, their operations in sending a particular packet can potentially be done in parallel. This solution is effective if the memory bandwidth is greater than that required by either operation alone, and is maximally effective when the memory bandwidth is at least the sum.

In this *DMA pipelining* technique, the devices must be synchronized so that (a) at the transmitting host the network interface does not transmit data yet to be encrypted, and (b) at the receiving host the cryptographic hardware decrypts data only after it has been received by the network interface. This synchronization is automatic if the appropriate device (the sender's encryption device, and the receiver's network interface) is faster and has higher DMA priority. If the second device in the pipeline is slower than the first, synchronization can be ensured by giving it a sufficient head start. A third alternative is to use a special-purpose *synchronizing DMA controller* that can perform multiple operations simultaneously, and in addition can delay the first operation in the pipeline to prevent it from advancing through the data faster than the second.

The second pipelining approach is to combine cryptographic and network functions in a single hardware device, within which the two operations are pipelined and synchronized. The unit would require a complex control interface, since different regions within a network packet may need to be encrypted with different keys or not encrypted at all. This approach has the significant advantage that no extra memory bandwidth is used for encryption.

Both of the above designs can be extended to include other I/O devices. As was mentioned previously, the latency of disk I/O is significant in network file access. Ideally, this latency could be overlapped with, rather than added to, that of network transmission and encryption. This could be done by either 1) having a single interface unit control all three devices, or 2) interfacing the disk via a common DMA controller capable of synchronizing 3 independent operations (disk access, encryption, and network transmission). In the latter case, memory bandwidth is again a limiting factor on the effectiveness of the technique.

4.3. Cryptographic Checksumming

In situations where authentication rather than secrecy is needed, cryptographic checksumming (using chained encryption and retaining only the final encrypted block) may be used rather than complete encryption. This reduces memory traffic by a factor of two, since data needs to be copied into, but not out of, the encryption hardware. This reduction yields an improvement in throughput, latency and CPU overhead, particularly in the cases mentioned above in which memory bandwidth is a limiting factor.

To exploit this efficiency, the encryption hardware and its interface must support the checksumming operation. This is not the case with the Zilog DES chip, which requires that all encrypted data be read from the chip.

4.4. Large Key Bank

The cryptographic hardware should have a large number of write-only registers for key storage. Keys can be loaded by software as secure communication channels are established. Encryption operations identify their key by an index into the register bank. The bank should have as many entries as the largest number of secure channels commonly in use (perhaps 256 or so).

This scheme has the following advantages: 1) it saves time since there is no need to load a key before each cryptographic operation; 2) the write-only property and the fact that keys are not kept in main memory ensure that keys are not compromised if an intruder gains control of the kernel on the host computer.

References

1. D. P. Anderson, D. Ferrari, P. V. Rangan and S. Tzou, The DASH Project: Issues in the Design of Very Large Distributed Systems, *UCB/Computer Science Dpt. Technical Report 87/338*, January 1987 .
2. D. P. Anderson and P. V. Rangan, A Basis for Secure Communication in Large Distributed Systems, *IEEE Symposium on Security and Privacy*, April 1987.
3. D. R. Cheriton, The V Kernel: A Software Base for Distributed Systems, *IEEE Software*, April 1984, 19-42.
4. M. Davio et. al., Efficient Hardware and Software Implementations for the DES, *Proceedings of the CRYPTO 84*, 1984, 144-147.
5. F. Hoornaert et. al., Efficient Hardware Implementation of the DES, *Proceedings of the CRYPTO 84*, 1984, 147-174.
6. M. Kochanski, Developing an RSA Chip, CRYPTO 1985.