Cryptographic Support in a Gigabit Network

Jonathan M. Smith (jms@aurora.cis.upenn.edu)

C. Brendan S. Traw (traw@aurora.cis.upenn.edu)

David J. Farber (farber@cis.upenn.edu)

Distributed Systems Laboratory, University of Pennsylvania
200 South 33rd St., Philadelphia, PA 19104-6389

ABSTRACT

Many applications envisioned for ultra-high-speed networks require cryptographic transformations for data in transit. Security has often been an afterthought, and cryptographic support has generally not kept pace with performance increases in other network components. Two distinct experimental prototypes of high-speed DES boards were built to understand architectural issues in providing cryptographic support for the AURORA gigabit testbed. Combining cryptographic support with the host/network interface is indicated.

1. Introduction

Network usage and capabilities are both increasing at terrific rates. The traffic load on the NSFnet backbones in the United States doubles every few months; much of this is due to the increased connectivity provided by interconnection with other networks, large numbers of personal workstations connected to LANs, and large numbers of computers connected via 9600 bit per second (bps) dial-ins. Much of this traffic is ‘‘traditional’’ Internet traffic, e.g., electronic mail, netnews, and file transfer, although a significant percentage of traffic is due to remote terminal sessions via *rlogin* and *telnet.* Network capabilities are increasing in a number of ways. Most important of these is bandwidth, as bandwidth can be spent to achieve a variety of other quality-of-service (QOS) goals. In the very near future, network technologies with bandwidths near 1 Gbps will be deployed in several areas of the United States [1]. Research into associated technologies has been underway for several years.

1.1. Applications of Gigabit Networks

Among the applications envisioned for such networks are interactive teleconferencing, advanced multimedia systems [15] with support for sensory data such as tactile data, and advanced displays with extremely high-quality imaging capabilities. Many of these applications would not be possible without large available bandwidths; while video streams and images can be compressed, compression is only proportional to the underlying bit rate, which can be enormous.

These applications are extremely attractive, as they move us towards a world where remote medical diagnosis and robotic mining are possible, and a world where jet lag may become less of an issue! Many more applications can be imagined, and we expect that many may become commercialized.

1.2. Security/Privacy

Many proposed applications, however, will not be feasible without significant attention paid to the issue of security. The need for security, perversely, is best illustrated by examples of what could happen if it is lacking:

- Medical images are transmitted through a switched network, and poor administration of a routing point allows the images to be perused. Lung cancer is apparent on a particular set of images; this data is reported to an insurance company for an anonymous bounty, and insurance is refused for the imaged individual.
- A Telerobotic command stream is altered to introduce subtle defects into a competitor’s products, e.g., automobile transmissions or VLSI

masks.

- Monitoring of weather prediction visualizations allows foreign grain traders to buy US grain futures at a price which hurts US markets after the rains arrive.

Traditionally, security schemes relied on either administrative means (e.g., systems with extremely restricted access, careful monitoring and audits, etc.) or cryptographic support, or some combination of techniques.

1.3. Cryptography

Since networks, by their nature, require distributed control, cryptographic means have somewhat more utility than solely administrative means to achieve security. A detailed survey of cryptographic techniques is beyond the scope of this paper, but an introduction is available in several books; Denning's book [9] is quite accessible.

The major axes of choice are: choice of encryption algorithm, hardware or software implementation, and placement in a networking architecture. Of the methods available which have withstood significant cryptographic attack, the Data Encryption Standard [16] (DES) and the Rivest-Shamir-Adleman [17] (RSA) schemes seem most attractive. While the use of public-key technology holds many seductions, the poor performance of RSA implementations are a major problem. A recent survey of hardware implementations of the RSA algorithm[Brickell89] indicates that the fastest available implementations are no faster than 1 Mbps; in fact, the fastest implementation reported by Brickell operated at about 50 Kbps. Significantly faster implementations of the private-key DES algorithm are available in hardware [10, 13, 20] and architectural techniques for further improving its performance have been reported [4].

The remainder of this paper discusses the Aurora Gigabit Testbed [6] and research into the issue of appropriate cryptographic support.

2. The AURORA Gigabit Testbed

AURORA [6] is an experimental wide area network testbed whose main objective is the exploration and evaluation of network technologies. The Gbps network will link four sites:

- Bellcore's Morristown Research and Engineering Laboratory in Morristown, NJ
- IBM Research's Computer Science Laboratory in Hawthorne, NY
- MIT's Laboratory for Computer Science in Cambridge, MA
- University of Pennsylvania's Distributed Systems Laboratory in Philadelphia, PA

The logical topology of AURORA is illustrated in Figure 1.

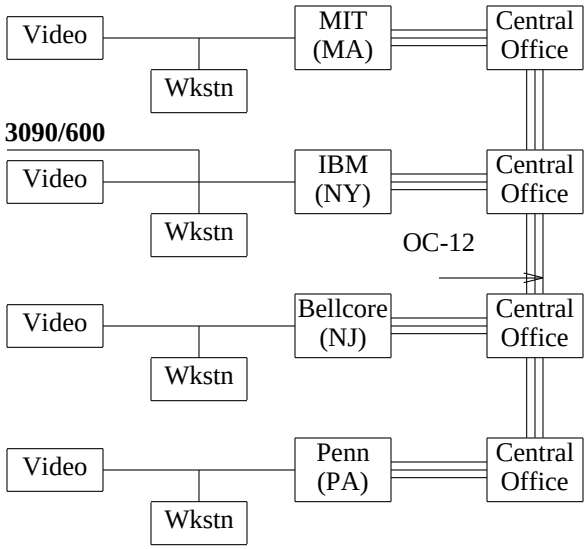

Figure 1: AURORA Logical Topology

Each of the three lines from sites to central offices in Figure 1 represents a logical OC-12. The point of this configuration is to first build independent plaNET (plaNET is the follow-on to PARIS [5]) and Sunshine [11] networks. These independent networks will be interconnected in order to understand interoperability of the technologies. Our host interface [21] is intended for the Sunshine-ATM logical topology.

The next three sections describe three designs for high-performance cryptographic subsystems. The first two have been implemented and the third will be implemented soon.

3. Exploitation of Parallelism in a Custom DES Design

The Data Encryption Standard (DES) is the most widely used publicly available secret-key algorithm. Since its introduction [16], DES implementations have improved in achieving high encryption rates. Computer data communications rates have also increased, and today's high-performance computer networks [1] further increase the encryption bandwidth needed to avoid performance bottlenecks in systems requiring security. Thus, architectural

means to increase DES throughput were studied, with the intent of producing a DES implementation to satisfy these bandwidth demands.

The key idea was to use the physical parallelism possible in a hardware realization to improve performance without resorting to exotic technologies. When such parallelism can be exploited cheaply using architectural innovations, the exploratory research would produce a useful artifact. The cryptographic subsystem was analyzed as a series of levels, ranging from the basic combinational logic of a single to the use of overlapped operation of the host processor and the cryptographic subsystem. The DES is implemented as sixteen ‘‘rounds’’ of key-directed encipherment; each round implements permutations and substitutions on the 64 bit input using the 56 bits of key for selection of the particular transformation. The 64 bits of output of a round are fed back as input for the next round until sixteen rounds are achieved, at which point the 64 bits are output.

At the combinational logic level, where a single round is implemented, the algorithm can be split into several parallel computations for increased speed. By generating subkeys one cycle in advance, the time required can be effectively overlapped with the use of the subkey in the rest of the round operation. An additional overlap can be made of the two stages of exclusive-or (XOR) gates at the expense of increased complexity and gate-count.

Use of multiple round implementations can increase computation bandwidth if the DES mode of operation chosen does not require feedback of ciphertext [19], ruling out all but the Electronic Code Book (ECB) method. Unfortunately, ECB is known to be susceptible to plaintext frequency-analysis based attacks since identical input blocks result in identical output blocks. A proposed operating method [14] resists this attack yet does not require feedback of ciphertext. The basic idea they propose is the use of sequence numbers XORed into plaintext blocks in a feedforward mode of operation. This produces a frequency-analyis thwarting many-to-one mapping as well as eliminating dependencies introduced by chaining which inhibit concurrent operation.

At the board level, concurrent I/O processing and DES computation provides for steady-state operation of the encryption unit. This can be accomplished using double-buffering and a small amount of extra logic. In addition to this buffering, the use of Direct Memory Access (DMA) for the encryption board allows the host processor to continue other work concurrently whatever the encryption unit is achieving.

One point worth a slight digression is the use of replicated hardware for implementing rounds; this provides an interesting range of cost/performance relations where realizable.

3.1. Unrolling rounds into hardware

Computation elements may be configured to implement a fraction of the rounds in a ‘‘pipeline’’-like approach. The main idea is the conversion of the iterative ‘‘rounds’’ of DES into a pipeline-like stream processing structure, analogous to hardware ‘‘loop unrolling’’. This can be accomplished by tagging each 64 bit input with the 56 bit key with which it undergoes the single-round transformation; a round can effectively be thought of as a 120 bit wide piece of combinational logic. A single-board implementation of the DES requires that the 16 prescribed rounds of DES be performed on the 64-bit data input and 56 bit key input before it is presented as output. The 16 rounds mean that 16 passes through the internal data path of any implementation be performed. Such a situation is illustrated in Figure 2:

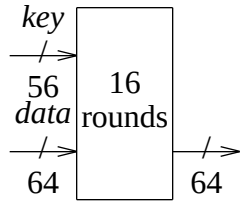

Figure 2: Single-Board, 16 round DES

However, if one employs multiple boards, for example four, as illustrated in Figure 3, one can reduce the number of rounds performed by each board. There is a small increase in the logic which must employed to manage the keying, but this is a second-order effect as far as performance is concerned. The primary relationship is this: up to a maximum of 16 DES units (one per round), there is a linear cost/performance tradeoff achievable by (1) doubling the number of DES units, and (2) halving the number of rounds each performs. For example, Figure 3 shows four units each performing four rounds. The upper limit is 16 units, each performing 1 round.

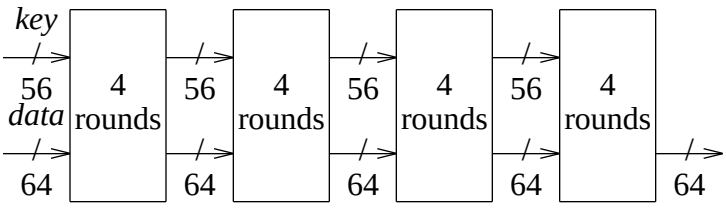

Figure 3: Four DES units, 4 rounds per unit

Ignoring second order effects, the major delay is induced by a round. Thus, as each ‘‘expansion’’ takes place, the unit count is doubled, and the round count is halved, meaning that the delay for the first bit through the sixteen unit case is almost exactly the same as the delay for the first bit through the single unit case. That is, at each point on the cost-performance line, from one unit to sixteen units, the delay through the configuration is unchanged.

When keying needs to be updated frequently, the pipeline style allows a matching of the datapath flow with a parallel keypath. In this way, data blocks are accompanied by their key throughout the computation. Switching keys between successive datablocks without flushing the pipeline is made possible since the key and data streams are synchronized.

For infrequent key changes, the tradeoff in keying interconnection may not be worthwhile as compared to maintaining separate key registers for each stage of the pipeline. Each round would then maintain its own key load (shadow) register in this approach. The standard’s key shifting sequence would be partially executed on each stage. Since a partial execution of the key schedule would not result in a complete cycling of the keytext, the key would be reloaded from the shadow register when it had completed its share of the computation on a data block.

A limitation of the pipeline approach is the bandwidth ceiling imposed by the number of rounds in the algorithm, sixteen. This means that a single pipeline of processors cannot provide more than sixteen times the encryption bandwidth of a single processor. Additionally, the pipeline suffers in scalability since the number of stages possible is restricted to be a factor of sixteen. This deficit is most notable when considering an upgrade of a pipeline: to gain any increase in bandwidth requires a doubling in computation resources.

3.2. Summary and Implementation Experience

Parallel aspects of the DES may be exploited at three levels: within the algorithm kernel, through duplication of the algorithm kernel, and in the encryption processor I/O design. Maintaining constant throughput rates requires careful consideration of the encryption system’s interface or input/output section. Overlap of the input, output and encryption processes of subsequent text blocks provides high throughput [23] Similarly, DMA support decouples the host processor from the encryption function to allow CPU processing of other tasks to proceed in parallel with the encryption request [2].

We developed a DES board [3] using SSI TTL and MSI PALs using the MUX key register approach. Testing of a wirewrapped prototype with a single algorithm kernel indicated an encryption rate of 93 Mbps. An interface to the Micro Channel Architecture bus of the IBM RISC System/6000 was also implemented.

This implementation taught us a great deal about the internal structure of the DES hardware itself, and was a surprisingly successful design exercise. Unfortunately, the parts cost of the implementation, potential performance limitations due to packaging, and the large amount of handwork required to replicate the work inhibited us. In addition, concurrently with the completion of the work and experimental evaluation, very high-speed DES chips were both implemented in research laboratories [10] and announced to be commercially available [20]. The availability of such chips both removed one of the motivations for implementing the board (high performance), and enabled us to redirect our energies towards architectural issues, such as where the chip should be incorporated.

4. Current Scheme

Experience with the wire-wrapped board described in Section 3 indicated that high encryption speeds were possible, and that architectural solutions could provide considerable leverage, independent of technology choices for the electronics. However, any solution was intended for incorporation in our Gigabit testbed, and thus several replicas were necessary for networking experiments, and the negatives described above (complexity and handwork) were thus even more important. VLSI Technology’s VM007 encryption chips [20] were chosen as the basis for a Micro Channel Architecture bus card.

This chip has a number of important features,

most notably the fact that it can be clocked at 30 Mhz, yielding Electronic Code Book (ECB) and Cipher Block Chaining (CBC) speeds of 192 Mbps. This is entirely adequate for the current generation host interface, which operates at OC-3c speeds (155 Mbps).

A key can be loaded in 8 clock cycles, or about 0.3 microseconds. Several registers for keying are provided.

The Micro Channel Architecture board has been implemented and tested, and software support has been written. It operates at rated speed, which exceeds the maximum transfer rate (130 Mbps) possible between the host and Micro Channel Architecture devices with the current generation Micro Channel Architecture I/O Channel Controller.

The major difficulty with this approach is the number of bus transactions necessary to move data to and from the network interface. This represents a serious performance limitation due more to the latency involved in roundtrips than in the bus throughput. Section 5.2 discusses alternatives; the nature of the traffic is illustrated in Figure 4

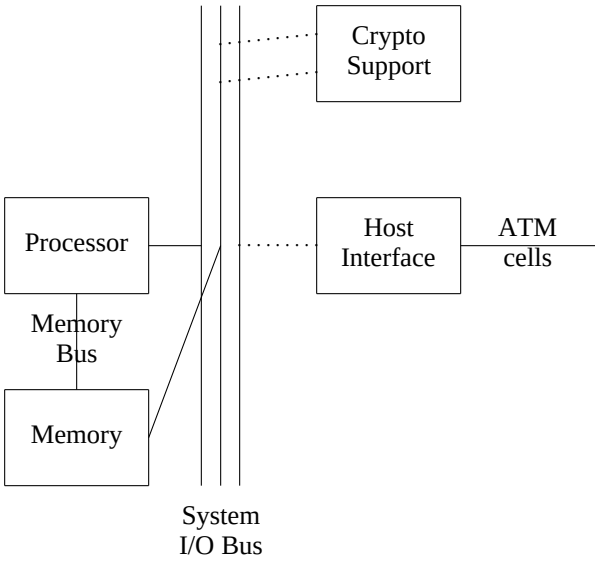

Figure 4: Bus-Connected Architectures

5. The right way

Our experience with the prototypes of Sections 3 and 4 leads us to believe that a truly high-performance encryption subsystem must be incorporated into the host/network adaptor.

5.1. An ATM host interface

We have designed and implemented a prototype ATM host interface [21] for the IBM RISC System/6000. The host interface is implemented as a pair of Micro Channel Architecture boards [7]. The Segmenter board serves to convert data objects, such as IP packets, to ATM cells. These cells are then sent over the network. ATM cells are 48 bytes of data plus 5 bytes of header, so that a reasonably sized data object (say 8192 bytes) on the machine requires many ATM cells to be put on the network medium. The Reassembler board takes ATM cells from the network and re-aggregates them into data objects such as packets.

Among the functions the Segmenter and Reassembler board provides are the computation of CRC checksums, multiplexing and demultiplexing of virtual circuits, and a small amount (64Kbytes) of buffering. The model for the host interface's role in a system is this: the host software makes any control decisions, while the host interface performs cellification, decellification, and any data movement to and from the host This significantly reduces the burden on the workstation, as it is organized for data processing, not memory manipulation.

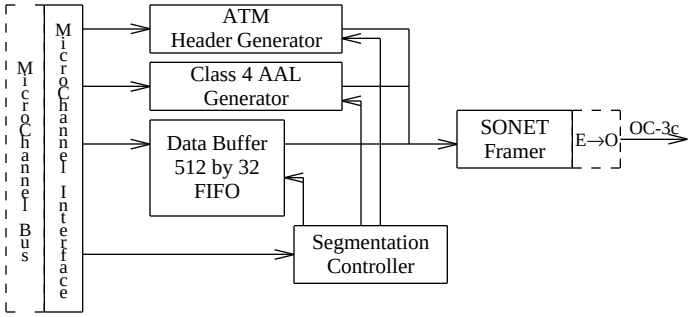

Figure 5: Segmenter

5.1.1. The Segmenter

A block diagram of the Segmenter is presented in Figure 5. When data is to be transmitted into the network, the virtual circuit identifier (VCI) to be used is loaded into the header generator. A multiplexing identifier (MID) is also loaded into the AAL generator if the data is to be transmitted via the Class 4 ATM Adaptation Layer (AAL4) [12]. The host then sets up a streaming mode [8] (an optimized bus transfer mode for contiguous data) transfer to move the data which is to be transmitted from a pinned buffer in host memory to the FIFO buffer on the Segmenter. The location and size of

the host's buffer are specified during stream set-up. While this transfer is being made, the Segmenter produces the header check CRC and formats the control information into the appropriate ATM and AAL4 header formats. As soon as sufficient data has been placed into the FIFO buffer, the segmentation controller removes the data for the first cell from the FIFO buffer and adds an ATM header, AAL4 header and AAL4 trailer. If the cell is carrying AAL4 data, the payload CRC is calculated as the data is moved to the SONET framer [18] and placed in the appropriate field at the end of the cell. This process is repeated until the FIFO buffer is drained.

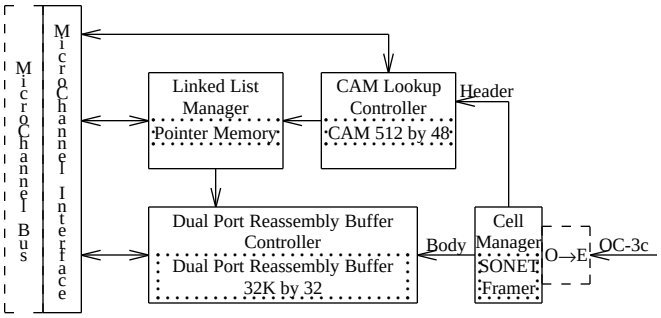

Figure 6: Reassembler

5.1.2. The Reassembler

The Reassembler is illustrated in Figure 6. It is composed of four major subsections which operate concurrently. The cell manager, CAM lookup controller, and the linked list manager (LLM) exploit this concurrency to form an ATM cell-processing ''pipeline''.

The cell manager verifies the integrity of the header and payload (if the cell is carrying AAL4 data) of cells received from the network by the SONET framer. It extracts the VCI from the ATM header and the MID and length from the AAL header. We currently ignore the 4-bit sequence number in the AAL4 as we believe it is insufficient to provide a reliable means for cell loss detection. The body of the cell is placed in a FIFO buffer for later movement into the dual port reassembly buffer (DPRB).

The CAM lookup controller manages two CAMs which provide lookup support for a total of 256 simultaneous virtual connections and the reassembly of 256 datagrams. The host is able to flush undesired virtual circuits and datagrams from the reassembler through the CAM lookup controller.

A reference resulting from the CAM lookup operation is passed to the LLM. The LLM, as its name suggests, establishes and maintains a linked list data structure for each of the virtual circuit and datagrams that is being received. Data received from the network is placed at the end of the appropriate list while data destined for host memory is read from the front of the list.

The LLM allocates space in the DPRB for data coming into the reassembler from the network and passes the location to the DPRB controller. The cell body which was placed into the FIFO by the cell manager is removed and written into the appropriate reassembly buffer.

The host is able to read data from a particular virtual circuit or datagram by specifying a list reference to the LLM which determines the location in the DPRB where data is stored. The location is passed to the DPRB controller which removes the data from the buffer for transfer into host memory over the MCA bus.

5.1.3. Discussion

Performance of our current architecture is quite good; we can send and receive at about 130 Mbps, measured by an AIX user process doing I/O to and from the host interface. Detailed performance data [22] is available.

5.2. Incorporating Cryptographic support: alternatives

There are essentially two ways in which encryption hardware can be employed. One method is to make the encryption card generally available for access on the Micro Channel Architecture bus, as was done for the solutions of Sections 3 and 4. This has the advantage that any application can employ the encryption hardware, for example, to encrypt and decrypt data written to disks, or to provide application-to-application cryptographic support. However, this approach (as illustrated in Figure 5) suffers from the need to access the bus multiple times for each network transaction, e.g., data must cross the bus to be encrypted, cross the bus after they are encrypted, and if the post-encryption transfer is to system memory, a third transfer (into the host interface) is required (although this is not strictly necessary on the Micro Channel Architecture as it supports card-to-card transfers). Since the considerable bus activity will severely inhibit

network performance, this seems somewhat undesirable.

The other alternative is to incorporate the cryptographic hardware into the host/network interface. This is often used when link encryption is desired, but has the drawback that the cryptographic hardware is private to the interface. In addition, link encryption itself has several drawbacks. First, it provides little control for applications, e.g., for customized key choices. Second, it makes key management for arbitrary host-to-host encryption in a switched network difficult; packet-switching exacerbates this problem since headers are encrypted and must be readable if used for routing.

In the next section, we describe a solution which combines some of the flexibility of the separate card architectures with the high-performance characteristics of a link-encryption solution.

5.3. Incorporating Cryptographic Support: A high-performance architecture

As shown in Figures 5 and 6, both the Segmenter and Reassembler portion of the host interface are broken into several functional blocks, which operate concurrently. This hardware concurrency permits high-performance solutions to be devised using inexpensive technology choices. The reassembler, in particular, makes great use of concurrency by setting up a ‘‘pipeline’’ which has four logical stages. The architectures we propose are illustrated in Figures 7 and 8.

The addition of the encryption chip in each case serves to add another functional block, which can operate concurrently with the other three blocks. Key lookup is done by adding an extra field to the ‘‘head-of-list’’ data structure in the pointer memory. Extra space for such purposes was reserved in the original interface design. These ‘‘head-of-list’’ structures are located using the CAM with the Virtual-Circuit Identifier as a tag. The cryptographic keys are loaded into the VM007 in parallel with data movement from the Cell Manager’s data buffer into the Dual Port Reassembly Buffer. This solution provides the following significant advantages:

1. Private keys for each virtual circuit (256 Virtual Circuits are currently supported in the host interface).
2. Real-time ‘‘agile’’ keying across Virtual

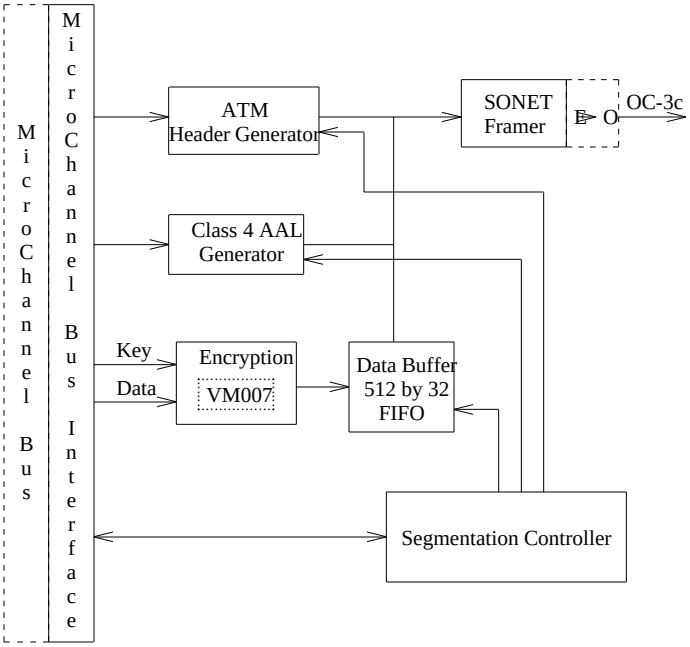

Figure 7: DES-augmented Segmenter

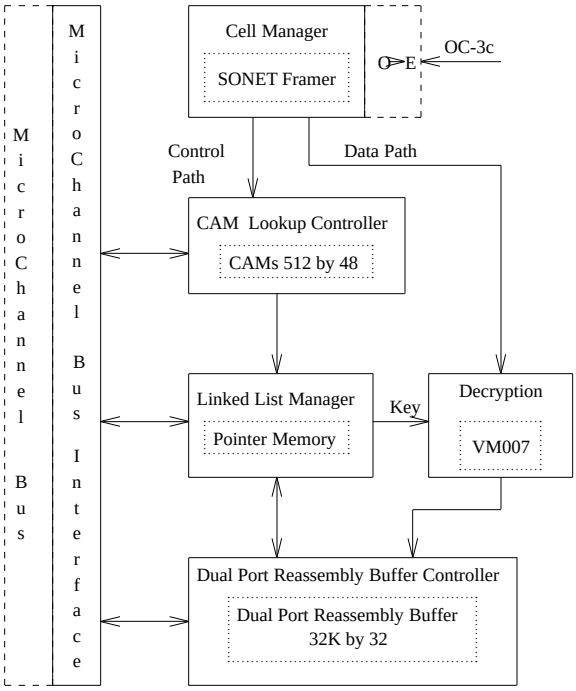

Figure 8: DES-augmented Reassembler Circuits.

3. Concurrent cleartext and ciphertext streams.
4. Application-specific control of cryptographic keying.
5. Cleartext headers and ciphertext payloads

(which ease transport through packet-switched fabrics).

6. Conclusions

Unless workstation architectures change greatly, cryptographic hardware should be incorporated into the host/network interface. This adds a marginal amount of complexity to the interface and meshes quite comfortably with functions it already performs, such as cellification, decellification, and computing checksums. The advantages to this approach stem from the fact that neither host memory access nor bus transactions are required beyond that necessary to move data from the application data areas to the host/network interface. Key management is closely correlated with ATM header manipulation, and can be performed efficiently with the addition of a small amount of logic in the host interface.

While we have used the DES algorithm to prototype our experimental solutions, our architectural proposals are not dependent on its use; they relate to the placement of cryptographic hardware in a workstation connected to a high-bandwidth network. These architectural solutions do not stand in the way of application-to-application (‘‘end-to-end’’) privacy, under the assumption that the workstation operating system is a ‘‘trusted’’ subsystem. Distinct keys can be loaded for each ATM virtual circuit (i.e., each application).

Many high-bandwidth applications require security, especially privacy. These requirements will drive transformations in workstation architectures, especially high-performance communications subsystems. Our proposal suggests an engineering approach which is both extremely cost-effective and provides high performance, while sacrificing little in flexibility.

7. Notes and Acknowledgments

Albert G. Broscius has been a rich source of ideas and critical judgement; he did the implementation described in Section 3. Balaji Srinivasan implemented the card employing the VLSI Technology’s VM007 described in Section 4.

AURORA is a joint research effort undertaken by Bell Atlantic, Bellcore, IBM Research, MIT, MCI, NYNEX, and Penn. AURORA is sponsored as part of the NSF/DARPA Sponsored Gigabit Testbed Initiative through the Corporation for National Research Initiatives. NSF (Cooperative Agreement Number NCR-8919038) and DARPA provide funds to the University participants in AURORA. Bellcore is providing support to MIT and Penn through the DAWN project. IBM has supported this effort by providing RISC System/6000 workstations, and this work was partially supported by an IBM Faculty Development Award. The Hewlett-Packard Company has supported this effort through donations of laboratory test equipment.

RISC System/6000, AIX, PC/RT, PS/2 and Micro Channel are trademarks of IBM. Ethernet is a trademark of Xerox. UNIX is a trademark of UNIX Systems Laboratories.

8. References

[1] *Gigabit Testbed Initiative Summary*, Corporation for National Research Initiatives, 1895 Preston White Drive, Suite 100, Reston, VA 22091 USA (January 1992). `info@nri.reston.va.us`

[2] David P. Anderson and P. Venkat Rangan, ‘‘High-Performance Interface Architectures for Cryptographic Hardware,’’ in *EUROCRYPT ’87 Proceedings*, Springer-Verlag, Amsterdam (1987).

[3] Albert G. Broscius and Jonathan M. Smith, ‘‘Exploiting Parallelism in Hardware Implementation of the DES,’’ in *Proceedings, CRYPTO 1991 Conference*, ed. Joan Feigenbaum, Santa Barbara, CA (August, 1991), pp. 367-376.

[4] Albert G. Broscius, *Hardware Analysis and Implementation of the NBS Data Encryption Standard*, University of Pennsylvania, School of Engineering and Applied Sciences (April, 1991). M.S.E. Thesis (CIS)

[5] Israel Cidon and Inder S. Gopal, ‘‘PARIS: An Approach to Integrated High-Speed Private Networks,’’ *International Journal of Digital and Analog Cabled Systems* **1**, pp. 77-85 (1988).

[6] David D. Clark, Bruce S. Davie, David J. Farber, Inder S. Gopal, Bharath K. Kadaba, W. David Sincoskie, Jonathan M. Smith, and David L. Tennenhouse, ‘‘The AURORA Gigabit Testbed,’’ *Computer Networks and ISDN Systems* **25**(6) (January 1993).

[7] IBM Corporation, *IBM RISC System/6000 POWERstation and POWERserver: Hardware Technical Reference, Micro Channel*

Architecture, IBM Order Number SA23-2647-00, 1990.

[8] IBM Corporation, *IBM RISC System/6000 POWERstation and POWERserver: Hardware Technical Reference, General Information Manual*, IBM Order Number SA23-2643-00, 1990.

[9] D.R. Denning, *Cryptography and Data Security*, Addison-Wesley (1982).

[10] Hans Eberle, *Personal Communication on 1Gbps GaAs DES Chip*, DEC Systems Research Center (April 1991).

[11] J. Giacopelli, J. Hickey, W. Marcus, W. D. Sincoskie, and M. Littlewood, ‘‘Sunshine: A High-Performance Self-Routing Broadband Packet Switch Architecture,’’ *IEEE Journal on Selected Areas in Communications* **9**(8), pp. 1289-1298 (October, 1991).

[12] CCITT Recommendation I.363, *B-ISDN ATM Adaptation Layer (AAL) Specification*, 1990.

[13] 1989 IC MASTER, *Fact Sheet, Western Digital WD20C03A*, 1989.

[14] Anthony McAuley and David C. Feldmeier, *Minimizing Protocol Ordering Constraints to Improve Performance*, Submitted for publication. Available via anonymous *ftp* from Internet host *thumper.bellcore.com*, 1991.

[15] Klara Nahrstedt and Jonathan M. Smith, ‘‘An Integrated Multimedia Architecture for High-Speed Networks,’’ in *Proceedings, Multimedia ’92 Conference*, Monterey, CA (April 1992).

[16] NBS, *Data Encryption Standard (FIPS Publication 46)*, National Bureau of Standards, U.S. Department of Commerce, Washington, DC (January, 1977).

[17] R. L. Rivest, A. Shamir, and L. Adleman, ‘‘A method of obtaining digital signatures and public-key cryptosystems,’’ *Communications of the ACM* **21**(2), pp. 120-126 (February 1978).

[18] Thomas J. Robe and Kenneth A. Walsh, ‘‘A SONET STS-3c User-Network Interface IC,’’ in *Proceedings, Custom Integrated Circuits Conference*, San Diego, CA (May, 1991).

[19] National Bureau of Standards, *Federal Information Processing Standard #81: Operational Modes of the DES*.

[20] VLSI Technology, Inc., *VM007 Data Encryption Processor:Advance Information Sheet*, 8375 South River Parkway (602)752-8574, October 1991.

[21] C. Brendan S. Traw and Jonathan M. Smith, ‘‘A High-Performance Host Interface for ATM Networks,’’ in *Proceedings, SIGCOMM 1991*, Zurich, SWITZERLAND (September 4-6, 1991), pp. 317-325.

[22] C. Brendan S. Traw and Jonathan M. Smith, ‘‘Implementation and Performance of an ATM Host Interface for Workstations,’’ in *Proceedings, IEEE Workshop on the Architecture and Implementation of High-Performance Communications Subsystems (HPCS ’92)*, Tucson, AZ (February 17-19, 1992).

[23] Ingrid Verbauwhede, Frank Hoornaert, Joos Vandewalle, and Hugo de Man, ‘‘Security and Performance Optimization of a New DES Data Encryption Chip,’’ *IEEE Journal of Solid State Circuits* **23**(3), pp. 647-656 (June 1988).