Context-Agile Encryption for High Speed Communication Networks

Lyndon G. Pierson, Edward L. Witzke
Sandia National Laboratories
Mail Stop 0806
P.O. Box 5800
Albuquerque, New Mexico 87185-0806
{lgpiers, elwitzk}@sandia.gov

Mark O. Bean, Gerry J. Trombley
National Security Agency
9800 Savage Road, Ste 6516
Ft. Meade, Maryland 20755-6516
{mobean, gjtromb}@alpha.ncsc.mil

Abstract

Different applications have different security requirements for data privacy, data integrity, and authentication. Encryption is one technique that addresses these requirements. Encryption hardware, designed for use in high-speed communications networks, can satisfy a wide variety of security requirements if the hardware implementation is key-agile, key length-agile, mode-agile, and algorithm-agile. Hence, context-agile encryption provides enhanced solutions to the secrecy, interoperability, and quality of service issues in high-speed networks. Moreover, having a single context-agile encryptor at an ATM aggregation point (such as a firewall) reduces hardware and administrative costs. While single-algorithm, key-agile encryptors exist, encryptors that are agile in a cryptographic robustness sense, are still research topics.

1.0 Introduction

Different applications have different security requirements for data privacy, data integrity, and authentication. Encryption is one technique that addresses these requirements. Encryption can protect proprietary information as it passes from one end of a complex computer network to the other, even through untrusted intermediate systems, such as on the Internet. Encryption technology has many other uses including encrypting disk files and producing digital signatures.

These various applications often have different needs. For example, certain applications may be able to tolerate long times to encrypt/decrypt information, but may also need to protect that information for a long period of time. Other applications, dealing with data that is sensitive while useful, but quickly becomes stale, might benefit from short encryption/decryption times that may accompany a less cryptographically robust algorithm. Digital signature systems typically demand rapid generation or verification of signatures. Depending on how frequently a signature must be verified, the system may need to be optimized for rapid signature generation or rapid signature verification. Efficient high speed communication systems, being of a real-time nature, often require encryption systems that optimize throughput while minimizing network traffic delay. Additional requirements may include minimizing error magnification, deterring message playback attacks, interoperability between faster and slower encryptors/decryptors, and quick recovery from cryptographic synchronization loss.

Just as different applications have different security needs, different users and communication sessions can have different needs. Symmetric end-to-end network encryption requires separate keys for each pair of communicating confidants. Each and any pair of communicating confidants can have multiple sessions (file transfer, virtual terminal, interprocess communication, etc.) proceeding simultaneously [22]. Each of these communication sessions therefore, can have different needs regarding session keys, cryptographic robustness, and other encryption and communication characteristics.

This paper discusses context-agile hardware for end-to-end encryption systems designed for use in high speed communications networks, such as those employing Asynchronous Transfer Mode (ATM) technology. Section 2 compares ATM switch designs with ATM encryptor designs and defines three types of agile encryption to meet the varying needs regarding keys, cryptographic robustness, algorithms, and other characteristics. The application area and advantages are outlined for each type of agility. Section 3 discusses implementation issues, such as high-speed context-switching and the potential effects on ATM Quality of Service (QoS). Section 4 briefly covers management

and administration issues. Section 5 details a proposed architecture for a context-agile ATM encryptor. Finally section 6 summarizes the work.

2.0 Context-Agile Encryption

Agile is an adjective meaning "moving quickly and easily" [4]. Hence, a context-agile encryption system can switch between various cryptographic contexts (key, initial variable, present state, key length, algorithm, mode of operation, etc.) quickly and easily. It is important to bound the limits of the available contexts per implementation, since a fully context-agile encryptor could have a nearly limitless combination of parameters.

The context-agile ATM encryption process resembles the ATM switching process. In particular, context-agile encryptors are similar to two-port ATM switches. For comparison, ATM switches modify cell headers and switch cells based on the "switching context" associated with each VPI/VCI. The initial association of switching information with a virtual circuit may be a manual operation for Permanent Virtual Circuits (PVCs). The initial association might also occur automatically at connection setup time for Switched Virtual Circuits (SVCs). Then, for each incoming cell, the ATM switch performs an associative lookup, of switching information, based on the VPI/VCI found in each cell's header. This switching information maps the incoming VPI/VCI into the appropriate outgoing VPI/VCI. It also conditions the hardware to switch the cell out the proper port.

Context-agile ATM encryptors resemble ATM switches in that encryptors must retrieve information and make decisions based on the cryptographic context associated with each VPI/VCI. The initial association of the cryptographic variables, state, algorithm, etc. with each virtual circuit may be a manual operation or be performed at SVC connection setup time (or later) via the methods invoked for key management [19]. Once a cryptographic context is established for a virtual circuit, for each incoming cell, the encryptor performs an associative lookup of the cryptographic context, based on the VPI/VCI found in each cell's header. The encryptor then uses that cryptographic context to transform the incoming cell payload (plaintext or ciphertext) into the appropriate outgoing payload (ciphertext or plaintext). Finally, the encryptor typically routes the cell out the opposite port of a two-port device. Hence, in certain aspects regarding context lookup, signaling, and cell I/O, context-agile ATM encryptors resemble a two-port ATM-switch.

2.1 Key-Agile Encryption

Key-agile encryption implementations limit the context parameters to items such as key, initial variable, and present state. Key-agile software-implementations of cryptographic algorithms are usually straightforward. However, software-based encryption can raise both performance and security concerns. Hardware implementations provide higher performance; but, the efficient implementation of high-speed context switching in hardware is not as obvious.

Key-agile encryption hardware provides obvious benefits in both computer systems and high-speed communication networks. High-performance computers are often shared resources. Key-agile encryption allows each user on a workstation (or server) to use different key material. This cryptographically separates the users' traffic. High-speed network resources are also often shared resources. Indeed, each session, through a common network interface, may require separate keys. For example, an ATM encryptor may support end-to-end hardware encryption of multiple Asynchronous Transfer Mode Virtual Circuits (VCs) across a network operating at speeds from 155 Mbps (OC-3) to 10 Gbps (OC-192) and beyond.

Key-agile encryption also has some limitations. First, different systems may implement different security policies. Hence, a user that communicates with several other end-systems might need access to several different, shared key-agile encryptors. Robustness-agile encryptors (described below) solve this problem. A second problem is that each ATM VPI/VCI (Virtual Path Identifier/Virtual Channel Identifier) combination (virtual circuit) has a different cryptographic context associated with it. Hence, any context-agile encryptor, whether key-agile or robustness-agile, must be able to switch encryption contexts very quickly as the ATM cells associated with various virtual circuits arrive for processing. Section 3 discusses context-switching in more detail.

Single algorithm, key-agile encryptors have been prototyped [9] [12] [16] and several products (CellCase [10], FASTLANE [8]) are on the market.

2.2 Robustness-Agile Encryption

A robustness-agile encryption system can quickly and easily change the confidentiality services offered, such as to an ATM virtual circuit, to an arbitrary level of strength (or cryptographic robustness). To do this, robustness-agile encryptors must switch rapidly (on the basis of the cell streams) between contexts of different cryptographic strength. This switching has at least three forms. The first is between different cryptographic algorithms. The second is between variants of the same algorithm implementing different "modes of operation" with different protection characteristics. The third form is between variants of the same algorithm which may utilize different key and other cryptographic state variable lengths. Other variables relating to cryptographic algorithms and their implementation may also be varied as part of the cryptographic context. Thus, robustness-agility requires even more information in the cryptographic context associated with each virtual circuit than encryptors that are only key-agile. This implies that larger context memories are required.

Communications networks often require varying degrees of data protection (cryptographic robustness) because of political boundaries. A workstation at a branch office in the United States may want to communicate with a file server at the corporate headquarters (also in the United States) using the Data Encryption Standard (DES) [3] with a 56 or 112 bit key. At the same time, that workstation may have a session, with another workstation at an overseas office, that uses exportable DES with a 40 bit key. This example illustrates cryptographic robustness-agility via one algorithm (DES) with a varying key length. (Some algorithms, such as RC2 [15], RC4 [15], and RSA [13], exhibit robustness-agility through variable key lengths, but the authors have seen these implementations only in software.)

Different traffic types may also require different degrees of cryptographic robustness. For example, a file transfer may need a high degree of robustness (to protect the data for a relatively long time), with a corresponding delay in encrypting and decrypting the data. Another application, within the same workstation and/or network, may involve real-time data acquisition. While that application may not be able to tolerate the delay imposed by highly robust encryption, it also may only require a modest degree of data protection (as its data may become "stale" relatively quickly). This example varies the cryptographic algorithm to change the level of robustness or cryptographic strength.

In both examples, variable robustness encryption provides clear advantages if it can easily and quickly vary the appropriate parameters of the cryptographic context. In these cases, one shared hardware-based encryption device can accommodate the varying cryptographic robustness requirements in a stream of cells, or packets, from many users.

2.3 Algorithm-Agile Encryption

Encryption/decryption hardware that contains multiple algorithms (or variants of algorithms) and can switch between them quickly enough to service adjacent cells or packets in a high speed data stream is said to be algorithm-agile. An algorithm-agile encryptor could also implement different modes of operation of the same encryption algorithm, such as the DES feedback modes [6], as separate algorithms.

Algorithm-agile encryption provides several advantages in addition to variable cryptographic strength. It also allows for compatibility with a broader range of devices. Each may implement various different algorithms because of a wide variety of application and policy requirements. For example, a workstation may contain an encryption/decryption module supporting algorithms (or variants) *A* and *B*. This allows that workstation to engage in both a secure file transfer session with a server that only supports algorithm *A* and a remote terminal session to a system that only supports algorithm *B*. This algorithm-agile encryption module provides both initial cost savings and ongoing administrative cost savings, as discussed in Section 4.

As a further example of the above case, some commonly used encryption algorithms do not scale well for high speeds. This includes the class of encryption functions with feedback based on a combination of key and plaintext or ciphertext. These do not scale (in a parallel manner) *and* interoperate with units having different degrees of parallelism [12]. An example of such an algorithm in wide-spread use is DES in cipher block chaining (CBC) mode [6]. Filter generator [14] algorithms have been found to both scale well for high speed operation and to interoperate

with unscaled or lesser scaled implementations [12]. An example of such an algorithm is DES in counter mode [15]. In networks containing various kinds of encryptors, it is important for an encryptor to be agile between widely used algorithms, and scalable, high performance algorithms.

Algorithm-agility can also support one or more public-key encryption (asymmetric) algorithm(s) (Diffie-Hellman [7], RSA [13]) and one or more secret-key (symmetric) encryption algorithm(s) (DES [3], Idea [15], Vernam [11] [5]) in the same hardware unit. This allows a workstation to engage in public-key encrypted sessions and secret-key encrypted sessions simultaneously. This property is useful for hybrid encryption systems [15] [22] where a public-key algorithm is used to exchange a session key for a symmetric algorithm. For added security, the selected (symmetric) session algorithm could be appended to the (secret) session key.

Agility between various encryption algorithms can also satisfy requirements that could vary from one communication session to another, such as error magnification and ATM Quality of Service (QoS) parameters. Several of these QoS parameters -- Cell Transfer Delay (CTD) and Cell Delay Variation (CDV, the difference between best-case and worst-case Cell Transfer Delay) -- are affected by algorithm-agile encryption. The differing latencies through different encryption algorithms can affect both CTD and CDV. The CDV is particularly important for voice and video traffic, while the CTD is important to data throughput. Section 3 discusses these QoS issues further, while reference [17] provides a full treatment.

3.0 Implementation Concerns

3.1 Large Context Space

All types of agile-encryption have some common issues. First, the number of potential contexts (VPI/VCIs) and the amount of information per context may both be large. In general, there may be either 2^{24} possible UNI VPI/VCI combinations or 2^{28} possible NNI VPI/VCI combinations. Hence, implementing the cryptographic-context lookup with “straight indexed” (flat) memory may be too costly. Because the number of simultaneously active contexts is likely to be small, an efficient key-agile encryptor could use an associative memory lookup to determine the key and other cryptographic state information, associated with each cell stream. Clearly, encryption algorithms that must associate larger keys and greater state information will be more cumbersome (expensive) to implement than algorithms that require a minimum of key and state information. In either case, large content addressable memories (or the even larger sequential memories required) with access times on the order of ATM cell header processing times are expensive and/or unavailable. Hence, until large, inexpensive, and fast content addressable memories do become available, current designs compromise either the virtual circuit space over which circuits can be encrypted, or the cell processing latency, or both.

3.2 Cryptographic Synchronization

Another common issue is cryptographic synchronization. When an encryptor and decryptor pair have lost synchronization, the decrypted data stream is scrambled, which leads to excessive data loss. While some cryptographic algorithms or modes of operation are “self synchronizing”, others require both initial synchronization and resynchronization after each cell loss. (Various specific methods of synchronization are addressed elsewhere [12] [18].) Since each virtual circuit has independent synchronization, the synchronization state information adds to the amount of information that must be associatively maintained for each encrypted virtual circuit.

3.3 Throughput

A third common issue is throughput. If the encryption or decryption process cannot keep up with the maximum possible cell arrival rate, then the cell traffic throughput on that virtual circuit must be throttled in some fashion to avoid cell loss. This can be done via Call Admission Control (CAC) at virtual circuit setup time (for constant, variable, and unspecified bit rate traffic) or by participation in the flow control after virtual circuit setup (for available bit rate traffic). In either case, the encryption/decryption devices must participate in the establishment and/or control of the VC, making it no longer “transparent” to the switching network.

3.4 Quality of Service

Robustness- and algorithm-agile encryptors can indirectly affect the ATM Quality-of-Service (QoS), since different algorithms, modes, and key lengths may affect delay, throughput, error magnification, and/or sensitivity to synchronization upset. In that case, the ATM QoS negotiation, that occurs at connection setup, must incorporate knowledge of the encryption methods used. One interesting effect is due to the different relative latencies of each algorithm. Algorithm-agile encryption preserves cell order within an individual virtual circuit and among virtual circuits that use the same encryption algorithm. However, it may cause re-ordering of cells that use different encryption algorithms. As these differently delayed streams are queued and re-combined into a single cell stream, this may also introduce additional Cell Delay Variation among cells of the same virtual circuit that use the same encryption algorithm. Again, reference [17] addresses this topic in detail. That research indicates that, until the encryptors participate in the QoS negotiation, algorithm-agile encryptors may need to append output buffering to low latency algorithms. This delay-equalization, with the higher latency algorithms, trades lowered CDV for increased CTD. If algorithm-agile encryptors can participate in QoS negotiations then the set of available algorithms may be dynamically restricted to maintain a previously negotiated CDV bound.

3.5 Assurance

There are two types of assurance mechanisms to consider. The first is of use-control – verification that the user has the authority to utilize specific services or contexts (eg. algorithm, key length). The second is confidence that the appropriate (and authorized) service is invoked correctly.

The issue of use-control becomes significant for algorithm-agile and robustness-agile encryptors. Along with the flexibility brought about by algorithm-agile and robustness-agile encryptors, comes the concern that certain algorithms (particularly the cryptographically stronger algorithms) may be used by unauthorized personnel. Hence, strong use-control techniques, like cryptographic authentication, are required to ensure that encryption algorithms are only used by those persons or processes that are authorized by the site's security policy. That authentication, using the algorithms built into the encryption hardware, is itself a form of key-agile, algorithm-agile cryptography. Furthermore, these same types of use controls, involving multiple keys and multiple algorithms, can also authenticate the commands to the encryption hardware regarding system operational modes (as opposed to cryptographic operational modes), resynchronization, and self-destruction-upon-compromise.

One research project at Sandia National Laboratories [20], used an external device (a smart card from Siemens) to control access to various encryption algorithms. Successful entry of a pin associated the card to a user. This is one possible way to bind control to the user, not specifically to the information.

Beyond use-control, proper invocation of services is critical. The authorization process is a required first step. In addition, the encryptor must guarantee that the data from a given user utilizes the specific context for which it is authorized. For example, once a user is authorized to use algorithm *A* for a given session, the encryptor must ensure that algorithm *B*, a less robust algorithm, is not inadvertently used. Once a single system implements multiple algorithms or data paths, this type of assurance becomes an area of concern and must be addressed adequately in implementation. Some mechanisms that may be used include redundancy, labeling and process monitors.

4.0 Management and Administration

Context-agile encryptors can be a tool to reduce costs and simplify administration. Putting encryptors at an ATM aggregation point, such as a firewall, enables virtual private networks (VPNs) across an untrusted network. By using context-agile encryptors at these aggregation points, traffic across the VPN can be encrypted using an encryption algorithm with particular characteristics and a certain key. Interspersed with this, traffic destined for sites (having different encryptors) off the VPN, can pass through the same encryptors, operated on by the appropriate (but different) encryption algorithm.

The aggregation point example above, and the earlier examples showing the advantages of context-agile encryption, serve to illustrate how deployed encryption hardware can be minimized by being agile across multiple characteristics. By deploying context-agile encryptors, the purchase, maintenance, and administration of encryption hardware is minimized. Key materials and key management may also be reduced. Less equipment, less maintenance, and less administration, all translate into lower costs.

5.0 Proposed Architecture

The general architecture of a context-agile, Asynchronous Transfer Mode encryptor, addressing the issues described above, has been developed. The architecture is scalable in data rate, but is targeted at a proof-of-concept implementation operating at 10 Gbps (OC-192). This architecture (and the project developing it and the implementation) is called OCTANE (for OC-192 ConText-Agile Network Encryptor).

The proposed architecture consists of the following major components. There will be both an input and an output physical I/O module, a cell Identification and Association Module, a Cell Router, one or more Cryptographic Modules, a Cell Combiner, a Key Management Module, and a Non-Real-Time Control Module. Figure 1 shows the connectivity relationship among the modules.

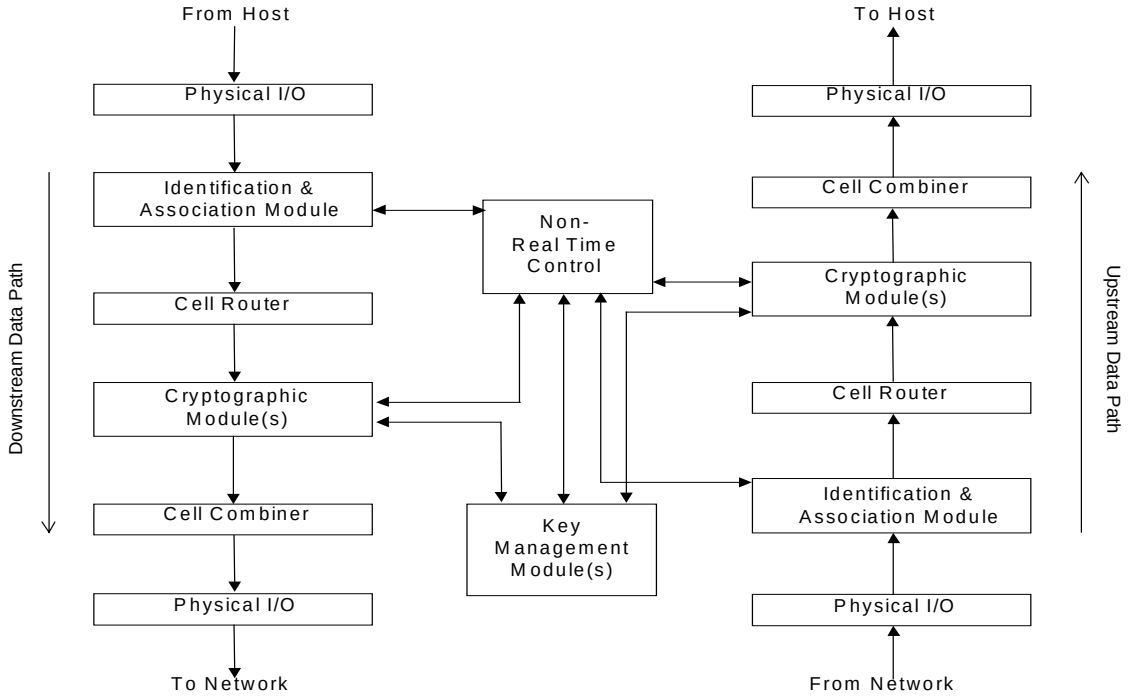

Figure 1. OCTANE Top Level Architecture.

These modules or functions can be broken down into real-time and nonreal-time services. The separation of real-time and nonreal-time traffic is critical to the overall system design. Specifically, the system must be designed to accommodate all real-time traffic in a very efficient manner such that excessive delays are not experienced. All user traffic is handled in real-time through the physical I/O, identification and association, and cryptographic services. The nonreal-time services consist of network management functions such as signaling and certain types of OAM (Operation, Administration and Maintenance) processes. The real-time data path is referred to as the high-speed path, and the nonreal-time data path is referred to as the lower-speed path. A distinction is also made between general, ATM-specific functions and security services. The ATM specific functions can be implemented in a shell, which surrounds the security services. The purpose of the shell is to provide a generic, non-proprietary definition for a large portion of the encryptor. This may lead to commercially available devices into which a

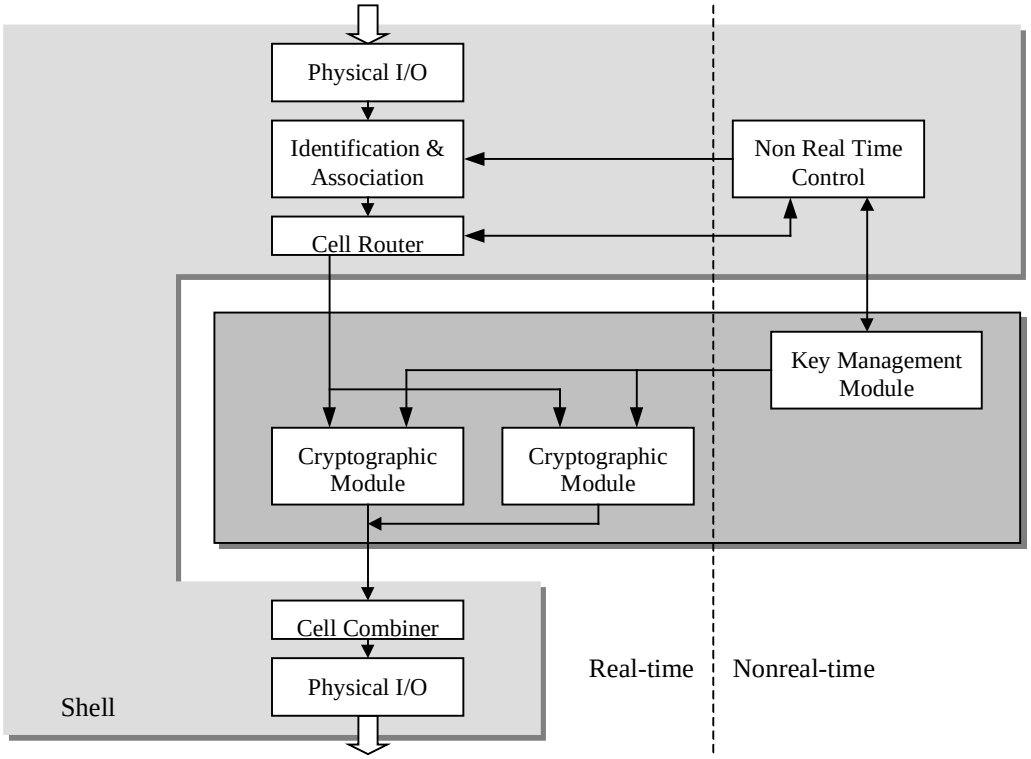

Figure 2. Shell and Security Module Relationships.

generic or custom (e.g. proprietary) security module may be inserted. Figure 2 shows the functional blocks, the shell boundaries, and the separation of real-time and nonreal-time functions.

5.1 The Shell

The primary data path (that path that carries user data cells) through the shell is from the physical I/O through the identification and association function to the cell router. User traffic continues through the cryptographic module (outside the shell), and is returned to the shell at the cell combiner. Finally, the physical I/O output port provides data to the network. Additional support data paths include interaction between the nonreal-time control and the identification and association function for the purpose of maintaining connection and association information. Additionally, nonreal-time cells such as signaling cells are diverted from the real-time path at the cell router to the nonreal-time control for processing. Similarly, the nonreal-time control may insert cells via the cell router. Each functional block is discussed in the following sections.

5.1.1 Physical I/O

The physical I/O is the link between the physical transmission of data across a network and the ATM encryptor. On the sending side, the input is coming from the host system, and the encrypted output is sent to an ATM network. Conversely, the receiving end accepts data from the ATM network, and provides decrypted data to the host system. It is assumed that the physical layer for high-speed communications will be SONET (Synchronous Optical NETwork). Therefore, SONET interfaces will be referenced throughout the paper, and in many cases will be used to imply ATM throughput rates. For example, the primary target of this architecture is a 10 Gbps encryption rate, which corresponds to a SONET interface of OC-192. The physical I/O is considered outside the scope of the encryptor development as it is assumed to be commercially available and requires no developmental or architectural design. Further, it is assumed that the SONET network interface will be translated to a standardized UTOPIA interface (or other ATM Forum recommended interface) for use by the encryptor. Currently, there are contributions within the ATM Forum for UTOPIA standards that will cover OC-192 data rates [2].

5.1.2 Identification and Association

The first function that the encryptor must perform is identification and association of ATM cells. In this architecture, the encryptor will take advantage of the connection-oriented properties of ATM by using connection information to specify parameters in the cryptographic context. Therefore, the ATM specific VPI/VCI (Virtual Path Identifier/Virtual Channel Identifier) is used not only for connection information, but also to determine the cryptographic handling of the cell.

5.1.2.1 Random Access Memory

As stated in section 3.0, one of the approaches for the identification and association function is to perform a table lookup using conventional memory (RAM). Some current encryptors such as FASTLANE (NSA/GTE Government Systems) use this approach [8]. However, the memory that would be accessed for a context lookup table must be considerably faster than in FASTLANE, which currently supports the OC-12 data rate (622 Mbps). For OC-192c (concatenated), the minimum cell period is only 42.6 ns in order to sustain full cell rate. All accesses to the connection table must be performed during this period. In addition, the maintenance of this table must also be performed without interfering with normal traffic. That is, when connections are initiated and/or terminated, the table must be updated appropriately. The table maintenance is a nonreal-time function. Therefore, these memory accesses will occur at unspecified times. One possibility is to allow the maintenance access during idle cell times only. However, given high traffic volume and frequent connection setup/teardown, this approach may be unacceptable, as it would create a backlog of required maintenance accesses. An alternate approach is to increase the memory access rate to allow for both a connection lookup and a maintenance access to be performed within a single cell period. For OC-192c, this requires the memory to be capable of both a read and write access within 42.6 ns. Additional memory access options are explored in [21].

5.1.2.2 Content Addressable Memory

Another approach identified in section 3.0 for connection table lookup, is to use a Content Addressable Memory (CAM). This associative memory is the most efficient method to perform the table lookup function because the CAM needs to be only as deep as the number of connections to be identified, plus a select set of generic cells common to all connections such as cryptographic resynchronization cells. Content Addressable Memory is used by loading the search VPI and VCI fields into any unused location in CAM. At this time, the appropriate association information is loaded into a RAM corresponding to the now filled CAM location. (The CAM finds the correct location when a cell's VPI and VCI is presented. The corresponding RAM then outputs the appropriate association information.)

CAM memory associations limit the number of connections due to limitations in CAM depth. Conventional SRAM does not limit the number of connections, but does restrict the size of the VPI and VCI field used for the address. In short, CAM is optimal for very wide associative lookups whereas RAM is preferred for narrower, deeper lookup tables On the surface, the advantage of the CAM approach may not be obvious. However, consider that each connection (fixed VPI/VCI) contains user data as well as OAM cells. The OAM cells require very different processing than the user data cells. The use of a ternary (1,0, don't care) CAM is more efficient at identifying specific types of cells on a given connection than a conventional RAM approach. Using CAM, there may be a single CAM entry identifying a cryptographic resynchronization cell (for all connections) and a single entry for each specific connection. When the CAM matches on both entries, it has identified the cell as being associated with a given connection as well as a specific cell type. Using a RAM approach, either a resynchronization entry is required for every connection (dramatically increasing memory size and address width requirements), or a secondary cell type lookup would be necessary.

Although a CAM implementation is more efficient, currently, there are no known CAM devices that will support operation at 10 Gbps. Therefore, although architecturally CAM would be the preferred technology, to construct a proof-of-concept implementation in the near term, Synchronous RAM and a reduced VPI/VCI table lookup approach may be used for the identification and association function.

5.1.2.3 ATM Cell Format

A final consideration of the identification and association function is the method in which to pass the association information (or cryptographic context) to the remainder of the system. There are two options. First, the context may be appended to the ATM cell header for use internal to the encryptor. Second, a separate dedicated path can be implemented to provide the context to the required components of the encryptor. Currently, FASTLANE appends this information to the header. This architecture will assume (although not require) that this strategy will continue.

The high-speed path of the encryptor will be a minimum of 64 bits wide to satisfy block algorithms like DES, and quite likely as large as 128 bits wide. These widths are necessary to maintain reasonable clock rates for data transfer. First, consider a 128-bit wide data path (Figure 3a). Given a 53-byte ATM cell, there are an additional 11 octets per cell that will not be used for ATM traffic. These 11 octets may be used to carry context information. Second, a 64-bit wide data path will provide three octets per ATM cell for context information (Figure 3b). In the event that three octets is insufficient to completely define the context, an additional 64-bit word may be appended to the header which would allow for 11 octets of context (Figure 3c). All cell formats considered are larger than the 53 octet ATM cell. Therefore, an increase in transfer rate is required to maintain the ATM throughput rate. For the 64 bit width with no additional context word (seven 64-bit words per 42.6 ns cell time), the transfer clock rate must be 164 MHz. Considering an additional 64-bit word for context information, the transfer clock rate must be 188 MHz. The 128-bit word width option must transfer data at a clock rate of 94 MHz (4 words per 42.6 ns cell time).

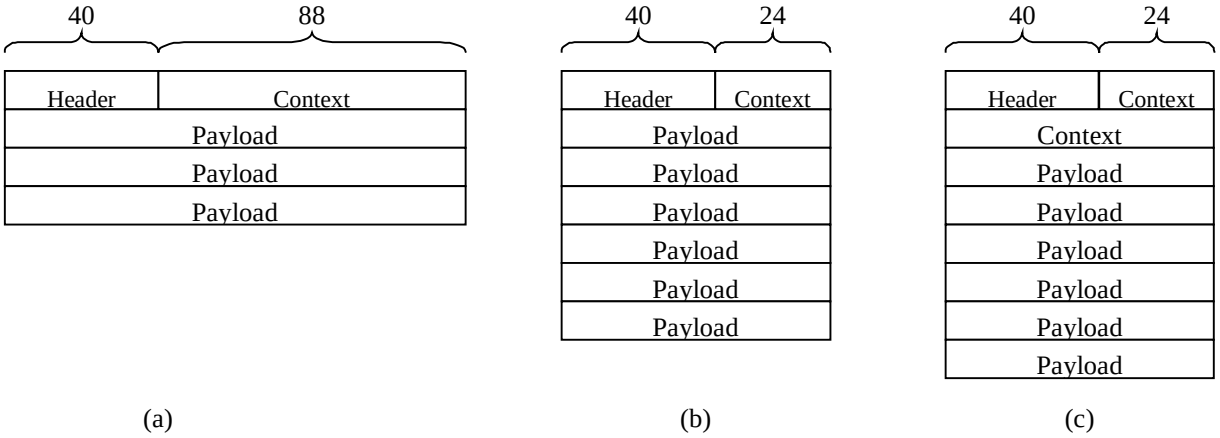

Figure 3. Internal ATM Cell Format

5.1.3 NonReal-Time Control

The primary purpose of the nonreal-time control (NRTC) is to provide control functions for the encryptor that do not need to be performed in real time (e.g., connection management and maintenance, and encryptor “housekeeping” functions). Subsequent to identification of a cell, nonreal-time cells (signaling cells, for example) are diverted to the NRTC. This requires that the cells be removed from the high-speed data path to the lower speed data path. The decoupling between the two paths is performed using FIFOs. Given the high speed nature of the real-time path, the FIFO must be able to accept data at full rate. Assuming a 64 bit data path, the FIFO must be able to accept data at a clock rate of up to 188 MHz. Alternatively, a 128-bit data path would require FIFO operation at 94 MHz. In addition, it must be deep enough to provide for enough storage given the relatively slow access from the NRTC. The NRTC must process assembled messages rather than ATM cells. In other words, a Segmentation And Reassembly (SAR) function must be performed prior to the NRTC processing. Additional glue logic may be necessary to properly interface the FIFO to the SAR.

It is desirable to have the SAR device support as many connections as possible. This is true because the encryptor may need to support many PNNI signaling connections to other switches. For high-speed networks, it is likely that there are a very large number of users multiplexed on the same high-speed connection, thus taking full advantage of the context-agility. The greater the number of users, the more likely the SAR will be segmenting and assembling multiple messages simultaneously. Currently, FASTLANE has demonstrated 64 simultaneous SAR channels to be adequate for operation at 0.622 Gbps in today's networks. Assuming linear scaling between throughput and number of concurrent SAR channels, it is recommended that the SAR function for a 10 Gbps device support a minimum of 1024 connections. In addition, the SAR must be able to access the decoupling FIFO at a rate such that the FIFO depth is minimized. Further traffic analysis is required to provide accurate estimates of adequate FIFO depth and SAR access rates.

5.1.4 Cell Router

When considering an implementation containing multiple algorithms, there should be as much separation as possible among the algorithms to prevent inadvertent misuse or intentional abuse. Therefore, a unique cryptographic path (also referred to as crypto path) for each algorithm is desirable. Similarly, each crypto path will maintain its own unique hardware to include: memory for the state vector and the cryptovariable, cell processor, key generator, and key/plaintext mixer. Another advantage to the hardware separation is that multiple vendors can supply a unique security solution and provide the exact hardware required by their implementation. For example, a less robust algorithm may require far less hardware resources resulting in a less expensive implementation. In order to gain access to a given crypto path, a cell router is required to properly route each cell to its appropriate crypto path.

Once a cell has been properly identified and associated with a given connection and cell type, this information is used to route the cell to the proper destination. In the case of OCTANE, there are multiple destinations: the NRTC or one of multiple crypto paths. All nonreal-time cells are destined for the NRTC; all real-time cells are destined for a specific crypto path.

In the most general sense, each cryptographic path consists of a unique cryptographic algorithm. However, the architecture does not prohibit identical algorithms in multiple paths for the purpose of either demultiplexing a high rate channel, or for the purpose of supporting multiple modes of the same algorithm. It is the responsibility of the cell router to send each cell to the appropriate cryptographic path, and it is the responsibility of the NRTC to properly program the identification and association function with the correct cryptographic path information.

In order to support the required data rates, a point to point architecture between the cell router and cryptographic path is recommended. The alternate approach is a bus architecture. However, bus technology is capable of operating frequencies of up to 66 MHz at a width of 64 bits (based on PCI Local Bus technology) without requiring extraordinary, custom design effort. This yields a theoretical bandwidth of only 4.2 Gbps, which is insufficient to support OC192 data rates. Recently, the CompactPCI bus has been identified as a possible bus implementation. This standard is attractive due to the relatively large number of user-defined pins which would allow for very wide (i.e. 128 bit or wider) busses. However, the problem of high frequencies across physical connectors remains unresolved. In addition, the point-to-point architecture provides a greater level of separation among the multiple cryptographic paths. Providing proper assurance mechanisms on the cell router will ensure that data is not misdirected through the incorrect algorithm, which could result in a serious security risk.

The primary concern of the cell router design is the very large I/O count, especially when considering the point-to-point architecture. The cell router must have a single input port and as many output ports as crypto paths. To support 10 Gbps, the interconnect must be configured as 64 bits at 188 MHz or 128 bits at 94 MHz. First, consider the 64 bit width interconnect. Support of three cryptographic paths requires 256 bits dedicated to the data path. Adding a nominal 20% overhead for power/ground pins plus miscellaneous signals (e.g. destination port information) brings the I/O pin count to 308. Now, consider the same case for 128 bit wide interconnects and three crypto paths. The total I/O pin count rises to 615. Therefore, 128 bit widths are difficult due to exceedingly large I/O count, and 64 bit widths are high risk due to the high frequency requirement.

In order to address both large I/O count and high operating frequency requirements, a hierarchical synchronous de-multiplexing scheme is proposed. In this scheme, a very simple 1 to 2 demultiplexer can be implemented as a single device. Multiple instantiations of this device are cascaded to achieve the desired fanout. By doing so, 128 bit wide interconnects operating at a lower clock rate can be implemented while still maintaining reasonable I/O count. In addition, great flexibility is achieved in terms of the number of crypto paths supported. Additional de-multiplexers can be added to support additional crypto paths.

5.1.5 Cell Combiner

Similar to the Cell Router, a Cell Combiner is required on the backend of the cryptographic modules. The Cell Combiner must select data from multiple crypto paths and combine them into a single high-speed stream. As in the Cell Router, the Cell Combiner has the implementation difficulty of a very large I/O count especially if there is a dedicated point-to-point connection to each crypto path. Based on the Quality-of-Service discussion in section 3.0, there are three distinct implementations for a Cell Combiner: First Come First Served (FCFS) queuing, latency matching, and priority servicing.

5.1.5.1 First Come First Served (FCFS) queuing

Of all cell combiner implementation options, FCFS queuing is the simplest to implement. In this case, cells from the encryption modules are queued in a FIFO which is served by the SONET framer. Although this is simple to implement, it is shown in [17] that variation in encryption delays can induce jitter in the cell streams when using the FCFS queuing discipline. However, because artificial delays are not added to the encryptor, the mean cell encryption delay is minimized.

5.1.5.2 Latency Matching

Latency matching is far simpler to implement. It assumes that all crypto paths maintain a constant and identical latency. Therefore, the extraction ordering and timing will match the insertion ordering and timing. Combining multiple data streams becomes trivial except for the high-speed operation requirements. The limitation of this approach is that the worst case latency path defines the latency of the cryptomodule. In other words, the longest latency crypto path defines the fixed latency required by all crypto paths. Therefore, use of a slow algorithm (i.e. large latency) will limit the capabilities of faster algorithms within the same system.

5.1.5.3 Priority Servicing

Priority servicing provides the capability to accept data from multiple streams on a priority basis, and is not necessarily related to the order in which cells were inserted to the multiple paths. There is a great deal of complexity involved with this type of implementation. Consideration must be given to a wide range of parameters. For example, if a single channel is being serviced by multiple cryptographic paths, then the Cell Combiner must ensure that the cells are inserted into the cell stream in the same order as they were extracted, since ATM guarantees cell ordering. Another issue is the fact that once a cell is displaced in the data stream, there is no guarantee that there will be an available insertion point in the stream in a timely manner. This is especially true for data communications, which is bursty in nature, and can cause significant problems for constant bit rate (CBR) traffic. Refer to [17] and [20] for an additional discussion of the implications of a priority servicing implementation.

5.1.5.4 Final Verification

An additional function of the Cell Combiner is to perform verification that the cell being processed is valid. Specifically, it is a redundant lookup as performed in the identification and association process. The result is compared with the context information that was passed through the system. The redundancy is included to add confidence that the ATM cell was processed correctly. As with any system, there is no absolute guarantee that data

was not inadvertently mishandled. However, this verification process can greatly reduce the possibility of such an occurrence as it adds an additional layer of failure detection. The verification process is the final step prior to passing the ATM cell -- with context information removed -- to the output physical I/O interface.

5.2 Cryptographic Functions

There are two primary cryptographic functions that an ATM encryptor must perform. First, key management must be provided for generating and maintaining traffic encryption keys. For the purposes of this discussion, it is assumed that the key management functions are performed locally rather than distributed across the network. Second, the network device must perform confidentiality services, i.e. encryption/decryption. Further security service requirements such as traffic flow security and authentication are handled by the nonreal-time controller through ATM signaling or by other higher layer entities/applications.

5.2.1 Confidentiality Services

Depending on the specific mode of operation, providing confidentiality services requires five basic components: key generator, key/plaintext mixer, state vector (SV) memory, cryptovariable (CV) memory and a cell processor. Ongoing exploration into Application Specific Integrated Circuit (ASIC) technology, Field-Programmable Gate Array (FPGA) technology, I/O capabilities, printed circuit boards and multi-chip modules will dictate the proper partitioning of these components. For the purposes of this discussion, it will be assumed that each component is a discrete ASIC, FPGA, or COTS device, as applicable. Figure 4 shows the relationship of these components.

The cell processor must provide processing on a per cell basis in real time. Fundamentally, the cell processor must perform one of two functions for each cell. Real-time cells are categorized as security related OAM cells (e.g. resync cells) or user data cells. The cell processor must process the cell differently based on the cell type. If it is a user cell, the cell processor must provide data appropriately to the key generator and mixer for proper encryption/decryption. If the cell is a resynchronization cell, OAM cell processing must be performed. The processing of the resync cells is described in the ATM Forum Security Specification [1], and is addressed subsequently.

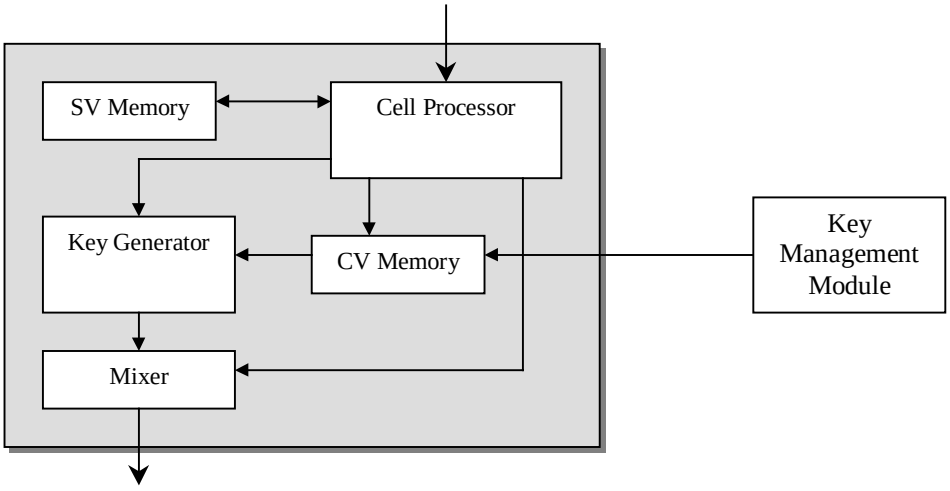

Figure 4. Cryptographic Module

5.2.1.1 Encryption/Decryption

In order to effectively encrypt or decrypt data at very high speeds, a non-feedback mode of operation is desired [23]. The two modes best suited for this application are Counter Mode and Electronic Codebook Mode. The following discussion assumes Counter Mode, as Electronic Codebook Mode does not protect against dictionary lookup or replay attacks.

User data cells are placed in a FIFO buffer while the key stream is generated. First, the cell processor fetches the current SV from memory. This memory access is a second look-up as opposed to the initial Identification and Association look-up. The correct SV is selected based on the context information, which was passed to the cell processor along with the cell header (as described previously). Then, the SV is written to the key generator, updated and written back to SV memory. Assuming a 128 bit word-width interface to the key generator, the cell processor must update and write the local SV a minimum of three times to cover the length of an ATM cell payload. (Cell headers are never modified as they are required by the network.) Only the final value of SV is written back to SV memory.

Due to the cryptographic sensitivity of the CV, sometimes referred to as "key" [15], the cell processor only provides an index, or address, to the CV memory and never handles the CV data. Instead, the data is fed directly to the key generator. The cell processor is responsible for properly synchronizing the SV and CV data arrival times at the key generator. Finally, the key stream generated by the key generator is passed to the key/plaintext mixer along with the user data cell payload, which was buffered in a FIFO. The resultant mixer output is ciphertext, which is appended to the original, unmodified cell header to form the encrypted ATM cell.

5.2.1.2 Cryptographic Resynchronization

In addition to user cell processing, the cryptomodule is required to process cryptographic information. Specifically, cryptographic resync cells must be extracted from and inserted into the cell stream, and processed accordingly. In order to eliminate bottlenecks and resulting backpressure on network traffic, all resync cell processing must be performed within a single cell period of 42.6 ns, for OC192 data rates. Further, the cryptomodule must process two types of resync cells -- those destined for the cryptomodule (cell reception) and those originating from within the encryptor (cell insertion.)

The insertion of resync cells requires stepping the current SV to the new value and writing the new SV to memory and into the resync cell. Stepping the SV is described in the ATM Forum Security Specification [1]. It involves incrementing the jump number, setting the I/R (Initiator/Responder) bit accordingly, resetting the Sequence Number and Segment Number to all zeros and the LFSR (Linear Feedback Shift Register) to its preset value. The new SV is then written to the appropriate SV memory location and to the resync cell. In addition, a CRC-10 is computed and inserted at the end of the cell. The new SV stored in memory is used on the next cell arriving on the given connection.

The reception of resync cells requires verification of the CRC-10. Similarly, the jump number is checked to verify that it is greater than the current jump number. If the CRC-10 value or jump number is invalid, the cell is ignored and no resynchronization occurs. If both values are valid, the jump number is extracted and the other SV fields are reset. The new SV is stored in SV memory and is ready for use by the next cell on the given connection.

5.2.2 Key Management

As in the nonreal-time control, the Key Management Module provides support services to the encryptor. The exact implementation is beyond the scope of this paper. However, the following details are considered relevant to the architectural discussion.

The primary purpose of the Key Management Module to the encryptor is to generate a traffic encryption key (TEK) for each connection. The TEK is also known as the cryptovariable (CV). The security message exchange protocols are described in [1]. The security messages exchanged are necessary for the generation of a properly authenticated TEK. However, the exact algorithm or method of generating the TEK is not specified and may be proprietary.

As has been mentioned, the CV is critically sensitive information. Therefore, the visibility of this data is limited as much as possible. To this end, there is a protected path from the key management module to the cryptographic module. The scope of "protected path" is left to implementation. In some situations, such as if the key management services were provided remotely, the protected path may be cryptographically protected (i.e. encrypted). However, for local key management, a protected path may dictate a dedicated point-to-point connection between the key management module and the cryptographic module. For this discussion, we will consider the latter case.

The cryptomodule interface is directly connected to a dual port memory device. The key management module controls one read/write port, while the cell processor of the cryptomodule controls the other read-only port. Again, the motivation for a read-only port is to limit the exposure and possible modification of the CV data. While protecting the CV data is one issue, protecting the address of this data also presents unique security and assurance concerns. Ultimately, both the key management module and the cryptographic module receive the CV address from a single source, the nonreal-time control. The validity and trustworthiness of the address is directly related to the trustworthiness of the NRTC.

6.0 Conclusions

This paper discussed the usefulness of key-agile, robustness-agile, and algorithm-agile encryption hardware. Several key-agile encryptor prototypes, such as the Milkbush [16], Enigma2 [9], and Scalable ATM Encryptor [12] have been built. Single algorithm, key-agile encryption products are becoming available (CellCase [10], FASTLANE [8]). Robustness-agile (via variable key length) algorithms do exist (RC2 [15], RC4 [15], RSA [13]), but the authors have only seen software implementations. This paper has then pointed out several obstacles to building general algorithm-agile encryption devices. Those problems include the unavailability of fast content-addressable memories, the interactions with QoS (Quality of Service) negotiation, and proper assurance mechanisms. After those implementation difficulties are overcome, context-agile encryption hardware will further enrich a host of secrecy, interoperability, and quality of (both communications and cryptographic) service functions in high speed communications networks, while minimizing cryptographic hardware, key management, and crypto administration costs.

Towards that end, an architecture for a context-agile encryptor has been presented in detail. An important concept of that architecture is a common shell to implement ATM specific functions and operate with one or more, custom or off-the-shelf, cryptographic modules. Parts of the proposed architecture have already been demonstrated. A full proof-of-concept implementation of the 10 Gbps, context-agile, ATM encryptor is under development.

Acknowledgements

The work described in this paper was a joint effort performed by Sandia National Laboratories (a multiprogram laboratory operated by Sandia Corporation, a Lockheed Martin Company) under contract number DE-AC04-94AL85000 to the United States Department of Energy, and the National Security Agency. The authors would like to thank Tom Tarman of Sandia National Laboratories, Jeff Ingle, Bryan Weeks, and Troy Young of NSA, Peter Sholander, formerly of Sandia National Laboratories, and Hans Rodriques de Miranda, formerly of RE/SPEC, for their contributions to this work.

Bibliography

1. The ATM Forum Technical Committee, ATM Security Specification Version 1.0, Straw Ballot, STR-SECURITY-01.00, The ATM Forum, Mountain View, CA, December 1997.

2. The ATM Forum Technical Committee, Scaleable Parallel Interface for UTOPIA, ATM97-0537, ATM Forum, Mountain View, CA, December 1997.

3. Data Encryption Standard (FIPS PUB 46), Federal Information Processing Standards Publication 46, National Bureau of Standards, Washington, D. C., January 15, 1977.

4. Davies, Peter (ed.), The American Heritage Dictionary of the English Language, Paperback edition, Dell, New York, 1979.

5. Denning, Dorothy Elizabeth Robling, Cryptography and Data Security, Addison-Wesley, Reading, MA, 1982.

6. DES Modes of Operation (FIPS PUB 81), Federal Information Processing Standards Publication 81, National Bureau of Standards, Washington, D. C., December 2, 1980.

7. Diffie, Whitfield, and Martin E. Hellman, "New Directions in Cryptography," IEEE Transactions on Information Theory, Vol. IT-22, No. 6, p. 644-654, November 1976.

8. http://www.nsa.gov:8080/programs/missi/kg75.html, June 1998.

9. http://www.mcnc.org/HTML/ITD/ANR/Enigma2.html, November 1996.

10. http://www.secantnet.com/product1.html, January 20, 1998.

11. Kahn, David, The Codebreakers, Macmillan, New York, 1967.

12. Pierson, Lyndon G., et al., Scalable End-to-End Encryption Technology for Supra-Gigabit/second Networking, SAND94-1622, Sandia National Laboratories, Albuquerque, NM, April 1997.

13. Rivest, R. L., et al., "A Method for Obtaining Digital Signatures and Public Key Cryptosystems," Communications of the ACM, Vol. 21, No. 2, p. 120-126, February 1978.

14. Rueppel, Rainer A., "Stream Ciphers," in Gustavus J. Simmons (ed.), Contemporary Cryptology: The Science of Information Integrity, IEEE, New York, 1992.

15. Schneier, Bruce, Applied Cryptography, 2nd edition, John Wiley & Sons, New York, 1996.

16. Semancik, William, et al., "Cell Level Encryption for ATM Networks and Some Results from Initial Testing," Conference Proceedings, DoD Fiber Optics '94, March 1994.

17. Sholander, Peter, et al., "The Effect of Algorithm-Agile Encryption on ATM Quality of Service," GLOBECOM 97, IEEE, Piscataway, NJ, November 1997.

18. Tarman, Thomas D., et al., "Algorithm-Agile Encryption in ATM Networks," IEEE Computer, Vol. 31, No. 9, p. 57-64, September 1998.

19. Tarman, Thomas D., et al., Final Report for the Protocol Extensions for ATM Security Laboratory Directed Research and Development Project, SAND96-0657, Sandia National Laboratories, Albuquerque, NM, March 1996.

20. Tarman, Thomas D., et al., Final Report for the Robustness-Agile Asynchronous Transfer Mode (ATM) Encryption Laboratory Directed Research and Development Project, SAND97-2902, Sandia National Laboratories, Albuquerque, NM, November 1997.

21. Trombley, G. J. and M. O. Bean, Technology Trends Influencing High-Speed INFOSEC Requirements, R2 Technical Report R22-003-98, National Security Agency, Ft. Meade, MD, February 1998.

22. Witzke, Edward L., and Lyndon G. Pierson, "Key Management for Large Scale End-to-End Encryption," Proceedings, 28th Annual International Carnahan Conference on Security Technology, IEEE, New York, October 1994.

23. Witzke, Edward L., and Lyndon G. Pierson, "The Role of Decimated Sequences in Scaling Encryption Speeds Through Parallelism," Conference Proceedings of the 1996 International Phoenix Conference on Computers and Communications, IEEE, New York, 1996.