Practical Experiences with ATM Encryption

Gregory T. Byrd
NC State University
Raleigh, NC

Nathan Hillery
Cylink
Morrisville, NC

Jim Symon
Cylink
Morrisville, NC

Abstract

CellCase is a commercial high-speed encryptor for Asynchronous Transfer Mode (ATM) networks, available since 1997. It provides data confidentiality and entity authentication at the ATM layer, encrypting ATM cell payloads at rates from T1 (1.5 Mb/s) to OC-12c (622 Mb/s). Though deployed prior to the adoption of the ATM Forum Security Specification (1999), CellCase implements many of the mechanisms defined by that standard. In this paper, we describe how CellCase is deployed in actual networks, as well as customer experience with services such as counter-mode encryption, key exchange, and key update. Based on this experience, we also discuss possible changes to the ATM Forum specification.

1 Introduction

Asynchronous Transfer Mode (ATM) [4] is the most widely deployed network technology for broadband communications applications. Its presence has been less noticed since it generally has been placed only within the inner workings of the public carrier networks. Since no well-established alternatives exist for high-speed wide area communications, the continued growth of ATM in the future is very likely. Some analysts are predicting that ATM will be the common multi-service platform for consolidating the numerous application-specific networks that exist today.

ATM has powerful multiplexing capabilities that can be used to provide quality of service (QoS) guarantees to end-user applications. QoS guarantees allow ATM to deliver time-sensitive information such as voice, video, and telepresence applications over the same network infrastructure as time-insensitive applications such as bulk data movement. QoS has been difficult to achieve in packet networks, since packets monopolize the link while they are being transmitted, blocking other packets that are to be routed over the same link. The delayed packets may cause the end-user application to experience degraded performance.

The features of ATM present some unique challenges for encryption. At any one point in the network many connections (2^{28}, or more than 250 million) can exist at a time; therefore, an encryption device must be capable of managing a large number of security contexts. Furthermore, the security context can—and generally does—change with each cell. At the OC-12c rate (622 Mb/s), the cryptographic state information for the new security context must be retrieved, used, and updated in less than 705 nanoseconds (the transmission time of a single 53-byte cell).

While the time and memory demands of ATM encryption are daunting, the data movement operations are highly regular. This allows the bulk of the encryptor design to be straightforward, leading to a minimization of failure modes. The implementation of the encryptor design is partitioned so that the regular, very high-speed operations are carried out exclusively in hardware, while unpredictable, complex operations are carried out in software. Once a secure connection has been established, the software portion of the encryptor is no longer involved in the handling of the data transported over that connection.

The experiences we have obtained from helping customers use the encryptor in their networks indicate that the encryptor design and implementation are indeed robust, usable, and efficient. However, as in all cases when a device is moved from a laboratory environment to deployment in the real world, we have encountered a number of new situations at customer sites that have been educational. This paper shares our experience in implementing and deploying one of the few commercially-available ATM encryptors. We discuss the impact on the encryption system of in-band network management, aggregation of multiple channels into paths, and traffic policing. In addition, we discuss the ATM Forum specifications that relate to security, pointing out some consequences of their current provisions and offering several suggestions for how the specifications could be enhanced.

2 System Overview

The CellCase system provides key-agile ATM-layer encryption, for network rates from T1 (1.5 Mb/s) to OC-12c

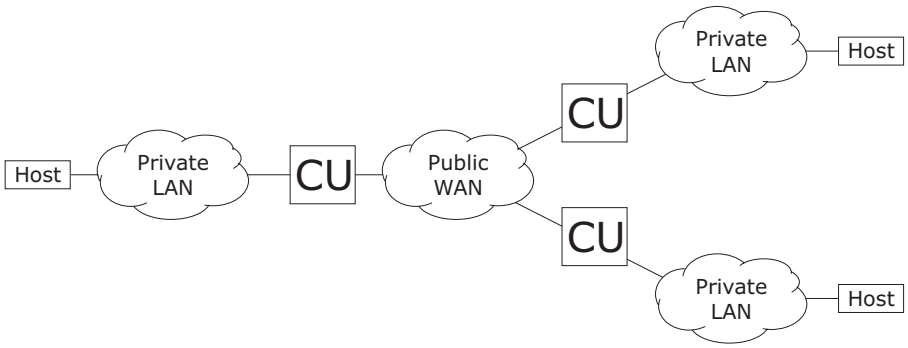

Figure 1. ATM Encryption. Each cryptographic unit (CU) encrypts data as it moves from the secure private LAN to the unsecure public WAN. The CU on the other side of the connection decrypts the data as it moves from the public WAN to the secure private LAN. Hosts may be any type of ATM end-user equipment. In key-agile systems, a separate key is used for each virtual connection through the public network.

(622 Mb/s). The cryptographic unit (CU) serves as a gateway between a secure private network (or host) and the unsecure public network. The 48-byte payload of each ATM cell is encrypted, while the five-byte header is left unchanged, so that the cell may be switched and routed through the public network normally. Each virtual connection, denoted by a virtual path identifier (VPI) and virtual channel identifier (VCI) in the cell header, has its own key and its own cryptographic state.

2.1 Hardware

The CU hardware is responsible for cell encryption and interfacing to the public and private networks. Higher-level functions, such as ATM signaling, key exchange, and key update, are performed by capturing selected cells, delivering them to software for processing, and inserting cells generated by software.

Figure 2 shows the hardware that controls the flow of cells between the private network and the public network. This is known as the "private flow." An identical set of hardware controls the public flow, from the public network to the private, except that cells are decrypted, rather than encrypted. Each flow is independent; there is no cryptographic state shared between the two pipelines.

First, the ATM Receive Interface converts the physical signal from the private network into ATM cells. The connection identifier (VPCI) is extracted from the cell header and used as an index for the VC Lookup Table (VLT). (The VPCI is a concatenation of the virtual path identifier (VPI) and the virtual channel identifier (VCI).) Other parts of the cell header are used to detect special cell types, such as Operation and Maintenance (OAM) cells. Such cells may require special processing, such as being extracted for software processing.

The VLT contains an entry for each enabled Virtual Connection (VC). (VC's are enabled by software, as described in the next section.) The VLT entries are sorted by VPCI, and a binary search is used to find the correct entry for the incoming cell. If no corresponding entry is found in the VLT, the cell is discarded. The VLT entry contains flags that determine the disposition of this cell: it may be extracted (either before or after encryption), encrypted, or passed through in the clear. If encryption is required, the flags also indicate the mode of encryption to be used. The flags travel with the cell through the rest of the pipeline.

The VLT entry also contains an index into the VC State Memory. This unsorted table contains an entry for each active connection. The entry contains the encryption keys and any other state information that must be kept for a connection. For each connection, two separate key banks are maintained in the state memory, with a bit that indicates which bank is currently being used. The unused bank is available for installing a new key as part of the key update procedure; after the new key is installed, the bank bit is changed and the new key is used for encryption/decryption. Some state must be updated after the cell is encrypted, such as the state vector for counter-mode encryption.

After the VLT, a pair of FIFO queues are available for cell extraction and insertion. Complete cells are inserted and removed from each FIFO, not individual words or bytes. If a cell is marked for capture and there is not enough room in the extraction FIFO, then the entire cell is discarded. A cell is removed from the insertion FIFO only when there is an idle cell (in the data stream) that can be replaced; a cell from the FIFO will never overwrite a data cell in the cell stream. (Therefore, it can be difficult to insert cells when a high percentage of the ATM link bandwidth is filled with data cells.)

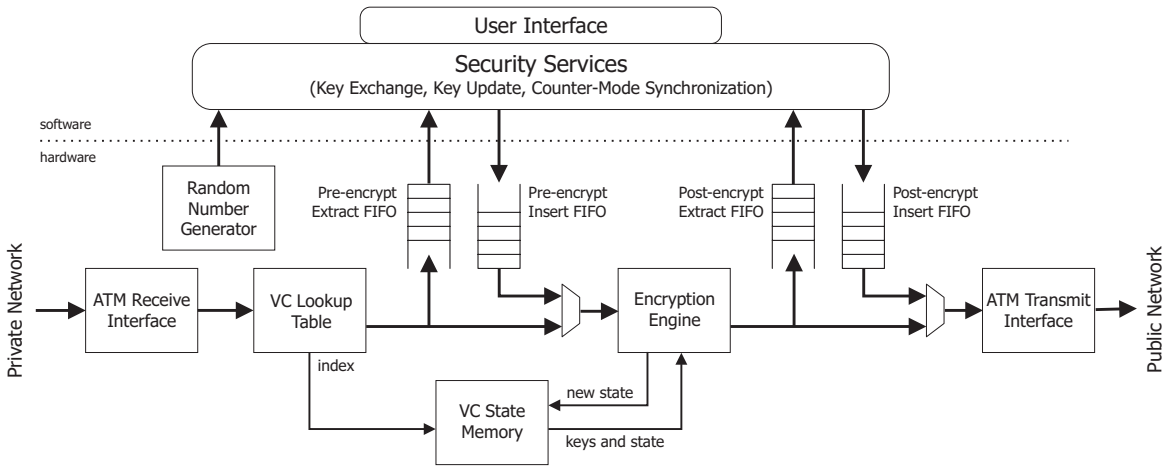

Figure 2. ATM encryptor—private-to-public flow.

Next, the received or inserted cell is passed to the encryption engine, along with the flags from the VLT and the keys and state information from the state memory. The payload of the cell is encrypted accordingly, while the header is unchanged. A cell that is not encrypted still flows through the engine, with the same latency as an encrypted cell.

After encryption is another pair of FIFO queues for cell extraction and insertion. As before, extraction is controlled by a flag in the VLT entry for this connection, and insertion occurs when an idle cell may be overwritten. Finally, the cell is transmitted to the public network by the ATM Transmit Interface.

In addition to the two cell pipelines described above, the hardware provides a single true random number generator. Random bits are accumulated into a 32-bit random register that may be read by software.

2.2 Software

The cryptographic unit software is responsible for managing the state of active connections, associating keys with secure connections, monitoring hardware status, and providing a management interface.

Connections are established through normal ATM signaling protocols, in the case of Switched Virtual Channels (SVC's), or by user action, in the case of Permanent Virtual Channels or Paths (PVC's and PVP's). In the case of SVC's, the calling and called end system addresses are matched against an access control list to determine whether a connection should be encrypted, passed in the clear, or disallowed. The cryptographic parameters of a PVP/PVC connection are set by the user when the connection is established.

Cryptographic parameters, such as key length, encryption mode, and key update interval, are independent for each connection. In this system, Triple-DES is the only algorithm provided, but the user may choose 56-bit, 112-bit, or 168-bit keys (equivalent to one-key, two-key, and three-key DES encryption, respectively). Two encryption modes are supported: electronic codebook (ECB) and ATM Forum counter mode (CM) [2], described in Section 2.3.1.

Output feedback modes, such as cipher-block chaining (CBC), reduce the throughput of Triple-DES by a factor of three. These modes are not supported, because no commercial encryption chips were available to match OC-3c or OC-12c cell rates when the system was designed. The initial research prototype [12] used experimental encryption chips [3] to achieve Single-DES with CBC at OC-12c rates.

Counter mode encryption and key update are implemented according to the ATM Forum Security Specification, described below. Both require the insertion of Operation and Maintenance (OAM) cells for cryptographic synchronization and for key distribution. The software is responsible for inserting these cells. As described above, hardware is responsible for capturing incoming OAM cells for processing by software.

Keys may be assigned explicitly by the user (for PVC's and PVP's only), or they may be generated automatically and distributed between cryptographic units through a key exchange protocol. The key exchange protocol is similar to the three-exchange protocol in the ATM Forum Security Specification. It provides confidentiality, mutual authentication, and protection from replay and man-in-the-middle attacks.

2.3 ATM Forum Security Specification

The ATM Forum is a consortium of ATM vendors and users that establishes standards for ATM equip-

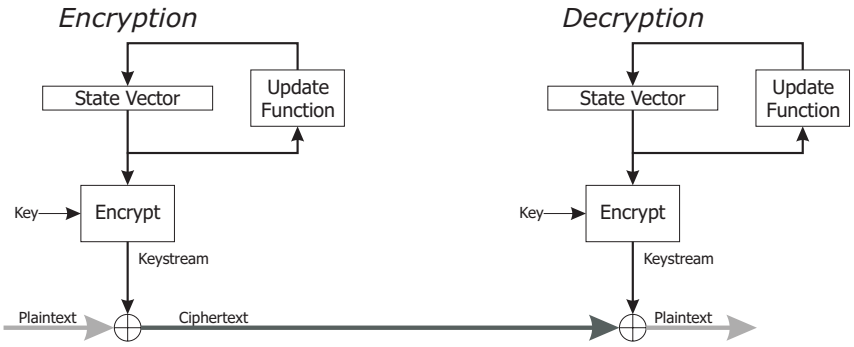

Figure 3. ATM Forum Counter Mode Encryption

ment. In February 1999, a specification for ATM security was approved by the Forum [2]. It establishes standard algorithms and protocols for data security in the user plane—providing transfer of user data across virtual connections—and the control plane—providing connection establishment, release and other connection functions.

The following security services are defined for the user plane: entity authentication, data confidentiality, data origin authentication and integrity, and access control. In the control plane, services for entity authentication and data origin authentication and integrity are defined. We will concentrate on data confidentiality and entity authentication in the user plane, since those are the services provided by the CellCase system.

Data confidentiality protects user data from unauthorized disclosure. The payload of each ATM cell is encrypted using a symmetric-key algorithm, such as DES [6], two-key EDE Triple-DES [1], or FEAL [11]. Symmetric-key algorithms are chosen because of their speed and because of the availability of high-speed hardware implementations. Encryption is performed at the ATM level, with the payload of each ATM cell encrypted and the cell header left in the clear. The standard specifies two modes of encryption: CBC and counter mode. (For more information about these algorithms and encryption modes, see Schneier [10], chapters 12, 13, and 15, or Menezes *et al.* [5], chapter 7.)

Entity authentication determines whether the identities of the called and calling parties in an ATM connection are genuine. This service is essential for establishing secure connections, since it protects against attacks that involve impersonation of a trusted entity. Approved algorithms for authentication include symmetric-key-based message authentication codes (MAC's), using DES and FEAL, and public-key digital signatures, such as those provided by RSA [9], DSA [7], and ESIGN [8]. (See Schneier [10], chapters 19–20, or Menezes *et al.* [5], chapters 9 and 11.)

To support these services, the Security Specification defines protocols for security message exchange, key exchange, and key update. Security message exchange is supported by the definition of standard information elements (IE's) that are used to construct and parse security messages. The information elements are defined to be consistent with IE's used in UNI (User-Network Interface) signaling [4].

The next three sections introduce the parts of the ATM Forum Security Specification that are most relevant to the CellCase system: counter-mode encryption, key update, and key exchange.

2.3.1 Counter-Mode Encryption

Counter mode is a method for implementing a stream cipher using a block cipher. The block cipher encrypts a changing state vector, and the result is used as a key stream. As shown in Figure 3, the key stream is bitwise exclusive-ORed with the plaintext to generate ciphertext. To decrypt, the same key stream is generated by the receiver (by encrypting the same sequence of state vector values with the same key) and is exclusive-ORed with the ciphertext, recovering the original plaintext.

Like CBC, counter mode encryption masks patterns in the plaintext. Unlike CBC, counter mode does not require feedback from the ciphertext stream into the plaintext stream. This is particularly important for Triple-DES, since the feedback reduces throughput by a factor of three. Counter mode takes full advantage of the fastest encryption chips available, leading to a system that scales naturally with technology to OC-12c and multi-gigabit transmission rates.

The state vector defined by the ATM Forum is designed for high-performance encryption of ATM cells. Each cell payload is 48 bytes, which corresponds to six 64-bit blocks. Part of the state vector is a counter from zero to five, identifying the six blocks in the cell. These state vector values can be generated independently, so all six blocks can be encrypted or decrypted in parallel. Another

part of the state vector is a 21-bit linear feedback shift register (LFSR), which is advanced one state for each cell. This definition allows $2^{21}-1$ key stream values to be generated with no repeated elements.

Most of the remainder of the state vector is devoted to a 35-bit jump counter. The sender and receiver are periodically resynchronized by incrementing the jump counter and resetting the LFSR to its starting value ($5A5A_{16}$). Resynchronization is necessary to recover from lost cells; without synchronization, counter mode encryption experiences infinite error extension when a ciphertext cell is lost.

Even without cell loss, synchronization must occur to avoid repeating the key stream. Therefore, a connection must be synchronized at least every 2^{21} cells. For a full-rate OC-12c connection, this corresponds to about 0.7 seconds worth of data.

Synchronization is accomplished through the use of special OAM cells, called Session Key Changeover (SKC) cells. The payload of the SKC cell contains a new jump number; upon receiving an SKC cell, the cryptographic unit sets the jump number for that connection to the value contained in the cell, and the LFSR and other parts of the state vector are reset to their starting values.

2.3.2 Key Update

The Security Specification defines a protocol for changing the data encryption key for an active connection without disrupting data flow. The new key is encrypted (using a previously negotiated shared key) and placed in another special OAM cell, called a Session Key Exchange (SKE) cell, which is sent along with user data on the connection. Upon receiving an SKE cell, the cryptographic unit decrypts the enclosed key and installs it in the unused key bank for that connection.

A Session Key Changeover (SKC) cell signals that the new key should be used for subsequent cells. This is the same SKC used for counter mode synchronization, described above. In addition to a new jump number, the SKC cell contains a key bank identifier, which indicates which key bank should be used for encryption for subsequent data cells. In connections that do not use counter mode, only the key bank identifier is used—the jump number is ignored.

2.3.3 Key Exchange Protocol

The Security Specification defines two key exchange protocols. The two-exchange protocol is intended for in-band key exchange, combined with the normal call setup procedure. It allows no negotiation of security parameters, and it relies on timestamps for replay protection. The three-exchange protocol performs key exchange on the user data channel, after it has been setup through the normal signaling mechanism. This protocol allows the responder to choose security parameters from a list presented by the initiator, and it uses challenge and response to protect against replay.

3 Customer Experiences

In this section, we illustrate the practical realities of deploying ATM encryption in customer networks. Generally, our experiences have been very good, validating that encryption can effectively be applied at the ATM layer. Full line rates for commonly deployed network bandwidths from 1.5 Mb/s to 622 Mb/s are supported. There is essentially no impact on traffic distribution patterns and negligible delay is introduced. Encryption has very small impact on delay variation. The degree of interoperation of ATM network equipment has proven to be high.

Of course, much can be learned from the unanticipated situations that occur in real-world environments. For the rest of this section, we describe some of those situations and the lessons learned from them.

3.1 Network Equipment Issues

A customer observed that channels encrypted in counter mode would occasionally have corrupted data. The error rate was correlated to the amount of traffic that the network switch was handling.

The switch supported two modes of handling OAM cells: it could be configured to intercept end-to-end OAM cells or let them pass through undisturbed. Even with the intercept function disabled, the problem behavior was still observed. However, when a virtual path connection (VPC) was established through the switch and the same counter mode traffic sent over it no errors were reported. The OAM cell intercept function was not available for VPC's.

An ATM analyzer was attached directly to the switch and configured to generate a stream of user cells with OAM cells periodically intermixed. The tester was also configured to receive the cell stream output from the switch and compare the content of the data cells to assure their values were correct.

At the start of the test, the traffic rate was low and the traffic distribution cell stream output from the switch matched the traffic distribution of the input cell stream. As the rate of data input to the switch was gradually increased, the distribution of cells in the output stream changed more and more from the pattern of the input stream. In particular, position of the OAM cells relative to the user cells shifted later in time. Eventually this shifting in time caused an OAM cell to leave the switch after a data cell that occurred later in the input cell stream.

Since OAM cells are used for counter mode synchronization, this re-ordering of the cell stream would cause

loss of synchronization between the encryptor and the decryptor. In other words, a cell would be decrypted with a portion of the key stream different from that used to encrypt the cell; the resulting plaintext cell would be corrupted.

The setting of end-to-end OAM cell interception did not seem to have much impact on this behavior. The problem was reported to the switch vendor, who determined that interception of end-to-end OAM cells was never completely disabled. The vendor created a test build of the switch software with a fix for this anomaly. When this test code was loaded on the switch in the test network and the ATM analyzer test rerun, the cell sequence integrity of OAM cells was preserved. Running the customer application with this test build also showed that the problem had been resolved.

The lesson here is that the intercept function can never be reliably used on a counter mode channel connection. A customer must choose between counter mode encryption for security and OAM cell interception for network management. In order to support both functions: (1) The switch must recognize synchronization OAM cells (SKC cells) and pass them without interception. (2) The encryptor must recognize non-SKC OAM cells and pass them in the clear. If the OAM cells are not encrypted, then they may be reordered with respect to data cells without affecting the decryption process. (This feature is supported by the CellCase encryptor.)

3.2 Virtual Path Connections

The CellCase system and its research prototype [12] were both designed to provide encryption for virtual channel connections (VCC's). ATM networks also support virtual path connections (VPC's), in which switching and routing is based only on a cell's virtual path identifier (VPI). From a management standpoint, we decided to support paths in the encryptor, partly to ease administration of large numbers of related channels and also to provide an interface that was similar to the management of ATM switches and interfaces.

In CellCase, a path is implemented as a collection of independent channels, managed as an integral unit. In the initial implementation, each channel in the path uses the same encryption key, but the state of each channel is maintained separately.

For a counter mode path, this means that each channel has an independently maintained state vector, as well. This has two important consequences. First, each channel must be synchronized independently. Second, using the same key for each channel means that the same key stream is generated to encrypt data on the separate channels.

When two plaintexts are encrypted with the same key stream, an attacker can exclusive-OR (XOR) the two resulting ciphertexts together to retrieve the XOR of the original plaintexts. In other words, for plaintexts A and B, and keystream K, $(A \oplus K) \oplus (B \oplus K) = (A \oplus B) \oplus (K \oplus K) = (A \oplus B) \oplus 0 = A \oplus B$. This may reveal information to the attacker that makes it easier to recover one or both plaintext messages. For instance, if one plaintext contains a well-known or easily guessed pattern, this is enough to recover part of the other plaintext.

One approach to solving the problem is to use different ranges of the jump number for each channel in the path. This reduces the effective lifetime of a key, however, since the number of cells that can be encrypted is determined by the number of unique values of the state vector. For a large path, one with many active secure channels, the effective key lifetime becomes impractically short if the state vector space is divided among the channels.

To solve this problem, the CellCase software was changed to use a separate key for each channel in the path. The key for each channel is derived from a path's session key and the relative position of the channel in the path. A key exchange or key update operation establishes the session key for the path as a whole, and this session key is then used to derive keys for each channel in the path.

Another customer-reported problem further illustrates the consequences of implementing a path as a collection of channels. The customer reported that they enabled a secure path that appeared to be passing data properly, but the decrypting CU reported failures on key updates. Obviously, the appropriate key values were installed, at least for the channels that carried the test data traffic. However, since the key update was reported as failing, the key value for at least one of the channels was not being set properly.

Further analysis showed that not all the channels in the range of the secure path had been established through the public network. In other words, a particular range of VCI values was enabled as a path in each CU, but not every channel in that range was provisioned in the network. For example, suppose the user at each end of the connection provisions path 10 with channels 32 through 255. In the network, however, channels 10/32 through 10/63 are provisioned as permanent virtual channels (PVC's). The other channels (10/65–10/255) are not provisioned in the network. Perhaps the user intends that they will be established later, and he does not wish to reconfigure the cryptographic state at that time.

During a key update operation, the new key is transmitted using SKE cells on one channel of the path, specifically the channel with the lowest VCI (32, in our example). However, since the state of each channel is maintained separately, SKC cells must be sent on each channel to complete the changeover process. Therefore, the key update process could not be completed on the channels that were not provisioned in the network (10/65 through

10/255), and the decrypting CU reported this as a failure.

The ATM Forum Security Specification views a VPC as a single connection, with a single cryptographic state. This is consistent with the treatment of VPC's in the ATM network: only the VPI of a cell is significant. (Certain VCI values are reserved for network use, however, and cells containing those VCI values must be passed in the clear.) This view of a VPC can be implemented with the current CellCase hardware by referring all channels in the path to a common state memory index. (Since each cell in the VPC updates the state vector, and thus the key stream, the duplicate keystream problem described above would not occur.) A more space-efficient approach, however, requiring fewer VLT entries, would be to change the hardware to examine only the VPI of a cell associated with a VPC.

3.3 Traffic Policing

One of the attractive features of ATM is its provisions for guaranteed bandwidth allocated to a connection. A bandwidth contract is established between the end user and the network when a connection is established, and the network equipment is expected to enforce that contract by giving priority to cells on that connection or discarding cells if the end user exceeds the allocated bandwidth. When the end user tries to match its traffic to the provisioned bandwidth, it is called traffic shaping. The actions taken by the network to enforce the bandwidth contract are called traffic policing.

Two behaviors can manifest themselves when using ATM encryptors in networks in which traffic flows are policed. The simplest behavior results from the rate of data from the source of traffic entering the network through the encryptor. During periods when the traffic source is transmitting data at exactly the policed rate, there is no bandwidth available for security functions, such as key exchange, key update, and counter mode synchronization. The CU inserts the cells required for these functions anyway, since it has no awareness of the bandwidth contract that the customer has with the service provider. Hence, both the data cells and the security function cells enter the public network. Cells exceeding the contracted rate may be discarded by the public network. The cells discarded might be data cells, security cells, or a mix of both.

During key exchange, a lost cell means that the key exchange message is not received properly at the peer CU. Depending on the protocol, the message may be resent or the entire key exchange may be abandoned. Since no data is being sent on the connection during a key exchange, problems with policing during this phase are typically due to the burstiness of the key exchange traffic.

After the connection is established, any data cell loss will result in corruption of the recovered plaintext. If the connection is encrypted in ECB mode, the impact of a data cell being discarded is the absence of six blocks in the recovered plaintext. For counter mode, however, loss of a data cell causes loss of synchronization between the encrypting and decrypting CU's. All subsequent data is corrupted until the next synchronization event. This is also true if the lost cell is a synchronization (SKC) cell, since the encrypting CU resets the state vector for this connection but the decrypting CU does not.

During key update, loss of an SKE cell (which carries the new key value) is unlikely to cause data corruption, because multiple SKE cells are sent for each key update event. Loss of an SKC cell does cause corruption, however, since the encrypting CU changes to the new key but the decrypting CU does not. Multiple SKC cells are also sent, however, so the corruption should cease when an SKC cell is successfully received by the decryptor.

We anticipated the effects of policing during design, but we didn't know how to quantify its impact. We also didn't have any basis for predicting how prevalent policing would be nor at what data rates it would be used. It is exactly this kind of unanticipated—and unanticipatable—factor that most challenge a design. As a result, the CellCase implementation was changed to allow the user to control the rate at which cells may be inserted into the data stream.

In our experience, traffic policing at a rate greater than 10% of the line rate with 250-microsecond cell delay variation tolerance (CDVT) is unlikely to cause problems for operation of the ATM encryptors. In practice, much lower rates can be supported without error.

4 Suggested Changes to the ATM Forum Specification

In the course of developing and marketing a commercial ATM encryptor, we have uncovered several subtle aspects of the security services that can hinder performance and customer satisfaction. In this section, we describe a few of those areas and suggest changes to the Forum specification to address them.

4.1 Negotiating Key Exchange Parameters

Key exchange requires authentication of the CU's, as well as encryption of the session keys. In the CellCase protocol, the RSA public-key algorithm is used to both sign and encrypt key exchange messages. The latency of these operations severely limits the rate at which secure connections can be established.

To increase call setup rates, a shared secret key encryption key (KK) is established during the initial key exchange. The KK is used for encryption during subsequent key exchanges between the same two CU's. Authentica-

tion is implied by the ability to encrypt using the KK.

Each CU is able to keep a limited number of KK's associated with other CU's. The number of peer CU's, however, is potentially unbounded. A KK from a previous exchange may be deleted in order to store a new KK from a more recent exchange.[1] The protocol must therefore deal with the case in which one CU has a KK (and therefore wants to use the fast key exchange protocol) and one does not (and therefore requires the public-key-based protocol).

Another consideration for the use of symmetric-key algorithms is that the CU closer to the call initiator (known as the calling CU) may not know the identity of the CU that will intercept the call for the call responder (known as the called CU). First, a CU may accept calls on behalf of many end users. Second, a different CU may handle successive calls to the same end user, due to changes in routing from one call to the next. Therefore, the called party address cannot reliably predict the identity of the CU on the responding end of the call. Since the KK is associated with a CU, not with an end system, the calling CU cannot determine which KK to use for a connection until it learns the identity of the called CU.

In the CellCase system, the called CU determines whether the fast key exchange protocol can be used. If it has a KK associated with the calling CU, it generates a key exchange message using that KK. When the calling CU receives the first key exchange message, it learns the identity of the called CU and whether that CU possesses a KK for this exchange.If the calling CU also possesses a KK, it continues the fast exchange protocol. If not, it initiates a new full authentication message sequence. The called CU then determines that the public-key protocol is being used and responds accordingly. A new KK pair is generated as a side-effect of executing the public-key protocol.

Consider the same scenario using the ATM Forum three-pass protocol. (The two-pass protocol does not allow security parameters to be negotiated.) The calling CU generates a list of encryption and authentication algorithms that may be used for key exchange. In this case, it lists a symmetric-key algorithm (e.g., Triple-DES) and a public-key algorithm (RSA), even though it does not yet know the identity of the called CU.

As before, the called CU looks in its cache to see if it has a KK established with the calling CU. If it does have a KK, it selects the symmetric-key algorithm and uses the KK to encrypt the key exchange message. If it does not have a KK, it uses the public-key algorithm. In either case, the choice of algorithm is conveyed in the response message.

Upon receipt of the second message, the calling CU knows the identity of the called CU. If the symmetric-key algorithm was chosen, then the calling CU must use the KK associated with the called CU. However, the CU may not have a KK—as described earlier, it may have discarded the KK to make room in its cache, or it may have been reset since the last exchange. Because it does not have a KK, the calling CU cannot complete the key exchange; it must be restarted, and this time only the public-key algorithm should be listed as an alternative.

Currently, there does not appear to be a way for the calling CU to restart the exchange without dropping and re-establishing the connection. To save this overhead, the Forum specification could allow the first message to be repeated, effectively renegotiating the security parameters for the exchange.

4.2 Key Update

The CellCase system implements the ATM Forum mechanism for changing the encryption key of an active connection, described in Section 2.3.2. One source of confusion for customers is that the update is a unidirectional process: only one directional flow is affected, and there is no feedback from the decrypting CU to indicate whether the key update was successful.

One CU initiates a key update by generating a new encryption key, installing it in the alternate hardware key bank, encrypting the key with a shared master key, and sending the encrypted key downstream in an SKE cell. Since the ATM connection does not insure delivery of cells, multiple copies of the SKE cell are sent. Still, there is no guarantee that the SKE cell is actually received by the peer CU downstream. Also, there is no information communicated to the initiating CU about whether the SKE was received.

When an SKE cell is received, the cell payload is decrypted (using the same master key) and the resulting session key is loaded into the peer CU's alternate key bank. The initiating CU waits an appropriate amount of time after sending its last SKE cell—the Forum specifies at least one second—and then initiates a key changeover. An SKC cell is inserted into the data stream before the encryption engine; the payload of the SKC cell causes the encryption hardware to switch to the alternate key bank for subsequent data cells. (For counter-mode connections, it also resets the state vector to its initial value.) The SKC cell then flows downstream to the peer CU, which also switches to the alternate key bank. As with the SKE, several SKC cells are sent (each indicating the same key bank) to reduce the probability of SKC cell loss.

The key update procedure described here only affects one direction of data flow, from the initiator to the downstream CU. To change the key in the other direction, a separate key update must be initiated by the downstream CU.

[1] In the CellCase system, KK's are not persistent across system reset, creating another situation in which one CU may not have a valid KK.

(a) Sender

(b) Receiver

Figure 4. Simplified state diagrams for the (a) sender and (b) receiver of a reliable key update.

Because there is no acknowledgement from the downstream CU, the initiating CU does not know whether the new key value has been installed at the decryptor. Therefore, the management interface on the initiating CU may report that the key update was successful, even though subsequent data passed on that connection will not be decrypted properly at the receiving end.

Feedback from customers indicates a need for a key update protocol that is reliable in two ways: (1) there is a high degree of confidence that the key update will complete, even in the presence of network errors, and (2) the sender of a key update knows whether the new key has been received at the downstream CU. Figure 4 shows simplified state diagrams for the sender and receiver in such a protocol. (Some error handling transitions are not shown.)

The sender initiates an update by sending an SKE cell, containing the new key to be installed (encrypted with a master key, as described above). It then waits to receive an acknowledgement cell (ACK) from the downstream CU. The ACK is a specially formatted SKE cell, containing the key number associated with the most recently received SKE cell. The sender periodically re-sends the SKE cell until an ACK is received.

When an ACK is received by the sender, it completes the update process by sending an SKC cell, which tells the receiver to start using the new key. An SKC cell is transmitted each time a valid ACK is received. If an improper ACK is received, such as one that contains the wrong key number, then the update process is restarted—a new key is generated and installed by the receiver, and a new SKE cell is sent downstream.

If no ACK is received within a user-specified timeout period, then the update is aborted and the sender reports an error. Upon failure, a new update is automatically scheduled after a user-defined time interval.

At the receiver, a new SKE cell indicates the start of an update transaction. The new key is retrieved and installed, and an ACK cell is sent upstream. An ACK cell is sent periodically, until an SKC cell arrives, indicating that the new key should be used for decryption. If no SKC cell is received within a user-specified timeout period, then an error is logged, but the receiver continues to send ACK's at a reduced time interval. In the case of a PVP connection, an SKC cell must be received on all allocated channels; if SKC's are received on some channels, but not all, when the timer expires, the receiver completes the update for all channels and logs an error.

This bidirectional approach does not work for connections with no bandwidth allocated in the upstream direction. Such connections are atypical, however, except for point-to-multipoint connections. Unidirectional connections can use the original unidirectional protocol, with the reliable protocol provided for the more typical case of a bidirectional point-to-point connection.

5 Conclusion

Encryption at the ATM layer is a relatively new technology, one that in many ways combines the speed and simplicity of a bulk link encryptor with the flexibility of an application-layer encryptor. We have attempted in this paper to pass on our experiences bringing this technology to market. These experiences can be summarized with a few observations:

- ATM encryption works. There are hundreds of encryptors in use today in production networks. The encryptors are transparent to the rest of the network equipment, requiring no modifications to hardware or software in the private or public networks. Full line rates are supported with very little added latency.

- Interoperability with other ATM network equipment is generally very good, but there are sometimes problems in the details, such as the OAM intercept problem (Section 3.1). In our experience, these are usually due to minor differences in interpretation of existing standards, exposed by the unique requirements of the security features.

- The security context for a virtual path connection (VPC) should be treated as a single entity, rather than as a collection of channel contexts. This simplifies

the key generation and key update processes for the path.

- Requirements for the CU to insert cells can conflict with traffic policing. Strategies to avoid this include shaping traffic after the encryptor, having the encryptor participate in the bandwidth contract, or reducing the rate at which cells are inserted by the CU, and allowing for the addition of security overhead when shaping traffic entering the CU.

- Renegotiation of security parameters should be supported during key exchange. This allows faster encryption and authentication algorithms to be used as appropriate, which is critical for sustaining high setup rates for secure connections.

- A reliable key update mechanism should be provided, or at least a mechanism that provides feedback as to whether the update is successful at both the encrypting and decrypting CU's.

- Counter mode allows high-throughput encryption, but it is more sensitive to network configuration and behavior. Cell loss and cell ordering have a more profound impact on counter mode than on other modes that either do not require synchronization (ECB) or are self-synchronizing (CBC). Also, the need to explicitly resynchronize adds the burden of inserting and processing synchronization cells to the CU's, and it can aggravate problems with policing on connections that operate near their contracted bandwidth.

Our experiences demonstrate that high data rate encryption at the ATM cell layer is practical. This design can be extended to even higher data rates through the use of either higher speed encryption devices or multiple encryption pipelines. These changes will have little impact on the overall architecture and on cell delay or delay variation characteristics of the current implementation.

This design offers a glimpse of the challenges that confront the designer of high performance packet encryption devices. Packet encryptors also have to handle rapid changes of security contexts from packet to packet. The data movement requirements are more demanding in packet encryption than in cell encryption, since packet payloads can be much larger (and smaller) than ATM cell payloads and are of unpredictable length. Therefore, the scheduling of memory and encryption resources will be much more complicated in a packet encryptor.

Since the ATM encryptor is a flow-through device, it can have a minimal impact on traffic distribution patterns. Specifically, it is possible for traffic streams with varying quality of service needs to be directed through an ATM encryptor without the QoS requirements being violated. We predict that the conflicting implications of the store-and-forward nature of packet handling and the emerging requirements for quality of service guarantees in packet networks will pose the greatest challenges to the designers of high performance packet encryptors.

References

[1] ANSI X9.17 (Revised). American National Standard for Financial Institution Key Management (Wholesale). American Bankers Association, 1985.

[2] ATM Forum. ATM security specification, version 1.0. ATM Forum Standard AF-SEC-0100.000, Feb. 1999. ftp:://ftp.atmforum.com/pub/approved-specs/af-sec-0100.000.pdf.

[3] H. Eberle and C. P. Thacker. A 1-Gbit/second GaAs DES chip. In *Proceedings of the IEEE 1992 Custom Integrated Circuits Conference*, May 1992.

[4] R. Händel, M. N. Huber, and S. Schröder. *ATM Networks: Concepts, Protocols, Applications*. Addison-Wesley, 3rd edition, 1998.

[5] A. J. Menezes, P. C. van Oorshot, and S. A. Vanstone. *Handbook of Applied Cryptography*. CRC Press, 1997.

[6] National Bureau of Standards. Data encryption standard. FIPS PUB 46-1, US Department of Commerce, Jan. 1988.

[7] National Bureau of Standards. Digital signature standard. FIPS PUB 186, US Department of Commerce, May 1994.

[8] T. Okamoto. A fast signature scheme based on congruential polynomial operations. *IEEE Transactions on Information Theory*, 36(1):47–53, Jan. 1990.

[9] R. Rivest, A. Shamir, and L. M. Adleman. A method for obtaining digital signatures and public-key cryptosystems. *Commun. ACM*, 21(2):120–126, Feb. 1978.

[10] B. Schneier. *Applied Cryptography: Protocols, Algorithms, and Source Code in C*. John Wiley and Sons, Inc., 2nd edition, 1996.

[11] A. Shimizu and S. Miyaguchi. Fast Data Encipherment Algorithm FEAL. In *EUROCRYPT '87*, pages 267–278. Springer-Verlag, 1988.

[12] D. Stevenson, N. Hillery, G. Byrd, F. Gong, and D. Winkelstein. Design of a key agile cryptographic system for OC-12c rate ATM. In *Symposium on Network and Distributed System Security*, Feb. 1995.