Mechanism for Encryption and Key Update Synchronization in ATM Networks

J. Gray, A. Kshemkalyani, M. Matyas, M. Peyravian, G. Tsudik
IBM Corporation
P. O. Box 12195
Research Triangle Park
NC 27709

Abstract

This paper presents a data compaction/randomization based approach as a mode of block encryption for ATM (Asynchronous Transfer Mode) cells. The presented approach converts a plaintext into pseudo-random plaintext before ciphering to conceal patterns in the plaintext. The underlying idea behind this scheme is the Shannon's principles of "confusion" and "diffusion" which involve breaking dependencies and introducing as much randomness as possible into the ciphertext. In this scheme, confusion and diffusion are introduced into the system by first compressing the ATM cell payload and then spreading a continuously changing random data over the entire content of the cell. As a mode of operation for block ciphering, this scheme offers the following attractive features: (i) plaintext patterns are pseudo-randomized and chained with ciphertext (thereby, preventing against "dictionary", "known plaintext", and "statistical analysis" attacks), (ii) it is self-synchronizing, (iii) cell loss has no additional negative effect, (iv) no IV (Initialization Vector) storage is required, (v) it is encryption-algorithm independent, (vi) there is no cell-to-cell dependency (no feedback from previous cells), and (vii) it is highly scalable (i.e., cells from the same stream can be ciphered and deciphered in parallel). This paper also presents a secure mechanism for in-band synchronization of encryption/decryption key updates using a "marker-cell" that is carried within the data channel. An important aspect of both the above mechanisms is that they do not require any changes to the ATM cell header or ATM infrastructure.

Key Words: Network, security, encryption, algorithm, ATM, key update synchronization.

1 Introduction

As critical and sensitive information ranging from patients' medical records to banking credit records and transactions get computerized and the evolving computer systems become more network-centric, it is becoming increasingly important to provide security to the information being stored at computers and transmitted across the network. Although considerable research has focused on preventing unauthorized access to confidential information [7, 19], there are security issues that are specific to new networking technologies and need separate treatment. Security mechanisms have been examined and developed for packet-based networks such as the Internet. However, newer security methods are required for the newer forms of cell-based high-speed networking technologies such as that used in Asynchronous Transfer Mode (ATM) networks [1, 2, 13, 15, 22].

In an ATM network, two end-systems communicate using a connection-oriented communication technology that transfers data using cells which are short fixed size blocks. ATM transfers different types of traffic such as voice, data, and video in an integrated manner using 53-byte cells. Each cell has a 48-byte data payload and a 5-byte header. The cells are delivered in the order sent.

Threats to an ATM network include leakage of information, manipulation of information, impersonation of valid network identity, and denial of the usage of resources [4, 5, 11, 12]. In general, four security services can be used to counter such threats to the ATM network: entity authentication, access control, information confidentiality, and information integrity [21]. These security services need to be seamlessly integrated into ATM so that security operations are transparent to both network users and equipments, and modifications to ATM protocols are kept to a minimum. The ATM Forum has started the work on ATM security specification [1]. As end-to-end security is required, [1] has specified that ATM security measures shall be provided at or above the ATM layer over VCCs (virtual channel connections) and VPCs (virtual path connections).

In this paper, we focus on providing information confidentiality for ATM user plane data. Cryptography (or encryption) is the sole method for secure end-to-end transmission over networks [3]. In cryptography, information, known as *plaintext*, is converted into a cryptic form, known as *ciphertext*, before being transmitted [6, 8, 10, 16, 17, 18]. The ciphertext is unintelligible unless it is decrypted using secret information, such as a key, known only to systems authorized to read and use the information. An intruder's attempt to procure the key using only the ciphertext is a "ciphertext only" attack; this is a computationally intractable problem. However, other forms of attacks such as "dictionary attack", "known plaintext attack", and "statistical analysis attack" simplify the task of the intruder to varying degrees.

1.1 Objectives and Solution Approach

The ATM Forum [1] has adopted DES (Data Encryption Standard) [23] as the "default" block encryption algorithm for user plane data confidentiality. In addition, [1] has specified that the ATM security infrastructure services shall support data confidentiality at the ATM cell level. As DES is a 64-bit block encryption algorithm with a symmetric secret key, for some applications it is not secure enough to encrypt each block of plaintext into a block of ciphertext. Since one block of plaintext always encrypts to the same block of ciphertext, this encryption mode is known as ECB (electronic codebook) mode. This mode of operation has a security weakness because it does not conceal plaintext patterns and repetition patterns in the plaintext can be inferred from patterns in the ciphertext. The intruder can make intelligent guesses about the plaintext to carry out dictionary attack, known plaintext attack, and traffic analysis attack. Although deducing the cipher key is computationally hard [9], it is possible to read the plaintext by skipping the search for the key and simply mapping the encoded characters back into plaintext. Note that this exposed security risk is further increased for low data-content cells such as those carrying Telnet traffic.

To overcome the heightened security exposure, some form of feedback mechanism can be used in the encryption process. Several feedback mechanisms, or modes of operation, exist. Some examples are: cipher block chaining (CBC) [24], cipher feedback (CFB), output feedback (OFB) and Counter modes. In these modes of operation, typically the result of encryption of a 64-bit block is propagated through the encryption of subsequent blocks. A plaintext block is combined using an exclusive-OR operation with the ciphertext of the previous block and the resulting block is encrypted using the DES substitution cipher. For encryption of the first plaintext block, a 64-bit random block is used as the initialization vector (IV). Thus, an encrypted block influences the encryption of all subsequent blocks and this is much more difficult to break than DES without a feedback mechanism.

None of the existing modes of operation is suitable for every ATM application and/or implementation. For example, CBC is self-synchronizing but does not scale well for high-speed implementations, counter mode is more scalable than CBC but requires periodic synchronization. It is particularly important that the encryption mode be scalable and self-synchronizing. If a solution does not scale well, then high-speed implementations become expensive because operations are forcibly serialized. If a solution requires periodic synchronization, then real-time applications such as voice, video, and multimedia will suffer as they cannot tolerate the loss of big blocks of cells between the synchronization periods.

The first problem we address is that of devising a self-synchronizing and highly scalable solution that is suitable for real-time or non-real-time. This is a data compaction/randomization based approach which converts a plaintext into pseudo-random plaintext before ciphering to conceal patterns in the plaintext. As a mode of operation for block ciphering, this scheme offers the following attractive features: (i) plaintext patterns are pseudo-randomized and chained with ciphertext (thereby, preventing against dictionary, known plaintext, and statistical analysis attacks), (ii) it is self-synchronizing, (iii) cell loss has no additional negative effect, (iv) no IV (Initialization Vector) storage is required, (v) it is encryption-algorithm independent, (vi) there is no cell-to-cell dependency (no feedback from previous cells), and (vii) it is highly scalable (i.e., cells from the same stream can be ciphered and deciphered in parallel). An important aspect of this solution is that no changes to the ATM cell header or ATM infrastructure is required.

The second problem we address is that of providing secure key update synchronization, defined next. The keys used for encryption/decryption have specific lifetimes because the longer a key is used, the greater the chances of a successful attack. The lifetime of a key is determined by factors such as the encryption algorithm used, the degree of confidentiality of the encrypted data, and the amount of data encrypted. For example, session encrypting keys have a shorter lifetime, such as a day, than key-encrypting keys which may be updated on a monthly basis. As ATM connections can be up for extended periods of time and have very high transmission rates, their session encryption keys need to be updated frequently. The two ends of an ATM connection can agree on and exchange the new key in a reliable and secure manner by using authentication protocols such as Kerberos [21], SPX, and KryptoKnight. However, there exists the problem of how the two ends of the ATM connection can reliably and securely synchronize the key update so that both ends begin using the new key for encrypting and decrypting the data payload of the ATM cells from the same ATM cell onwards.

The solution we present for the secure and reliable synchronization of the usage of new encryption/decryption keys between the two ends of an ATM connection is based on the method of compaction/randomization we proposed for encrypting ATM cells. Specifically, the synchronization is achieved by the usage of a "marker cell" which contains the old and the new keys. An important aspect of the solution is the ability to distinguish such cells from other encrypted cells without any changes to the cell header or ATM infrastructure.

The rest of paper is organized as follows: Section 2 discusses the design of the solution and the algorithm for compaction/randomization of ATM cells. Section 3 discusses the design of the solution and the algorithm for synchronizing encryption/decryption key updates in ATM connections. Section 4 gives the conclusions.

2 Randomization/Compaction Mode

The underlying idea behind the security of this compaction/randomization approach is the Shannon's principles of "confusion" and "diffusion" which involve breaking dependencies and introducing as much randomness as possible into the ciphertext [20]. The principle of "confusion" changes a piece of information so that the ciphertext has no obvious relation to the plaintext. The principle of "diffusion" spreads the correlation and dependencies among ciphertext as much as possible to maximize the length of plaintext needed to break the system. We introduce confusion and diffusion into the system by adding a continuously changing random data to the plaintext in every ATM cell and by spreading the randomness over the entire content of the cell.

The solution we present works by using data compression to compress the ATM cells, using the freed up space to convert the plaintext to pseudo-random plaintext, and chaining of the ciphertext with pseudo-random plaintext to conceal plaintext patterns. This method is applicable to all cells that can be compressed by a certain amount to achieve added encryption through compaction/randomization. There is now a need to distinguish between (i) the encrypted cells whose data payloads were not subjected to the compaction/randomization, and (ii) the encrypted cells whose data payloads were subjected to the compaction/randomization, at the receiving end of the ATM connection. This can be achieved by using a "compaction indication bit" in the 5-byte ATM cell header to indicate whether or not the data has been compressed/randomized. However, a bit in the ATM cell header is not available for such use. Except one payload type identifier (PTI) code point, all the bits in the ATM cell header are already used for other functions. Given that there is no available bit in the cell header and there is only one PTI code point left, which is reserved for future use, we had to design a solution that does not require a compaction indication bit in the cell header. An important aspect of the solution that we propose is the ability to distinguish cells that have undergone the compaction/randomization from those that have not, without requiring any changes to the cell header or ATM infrastructure. The proposed solution achieves this by using a Cyclic Redundancy Check (CRC) computed over a certain bit range of the payload and embedding it in the payload before encrypting the cell. The CRC acts like a compaction indicator to the receiver.

2.1 Algorithm

The algorithm first determines whether the payload of an ATM cell (shown in Figure 1(a)) to be encrypted can be compressed adequately to free enough space to be pseudo-randomized and to carry encoded information about whether the cell has udergone this compaction/randomization. This is performed by applying a data compression technique such run length encoding (RLE) base 0 or 1 to compress the data field of the ATM cell (see Section 2.2 for an overview of RLE). If the data field can be compressed by at least x bits, then the cell payload will be compressed and randomized before applying encryption; otherwise, the cell payload is simply encrypted. [1] When the data payload is compressed, the data field of the cell is replaced with the subfields shown in Table 1. The format of a compressed/randomized cell prior to encryption is shown in

[1] The derivation of x is based on the the length of the various subfields of the compacted/randomized payload.

Figure 1(b).

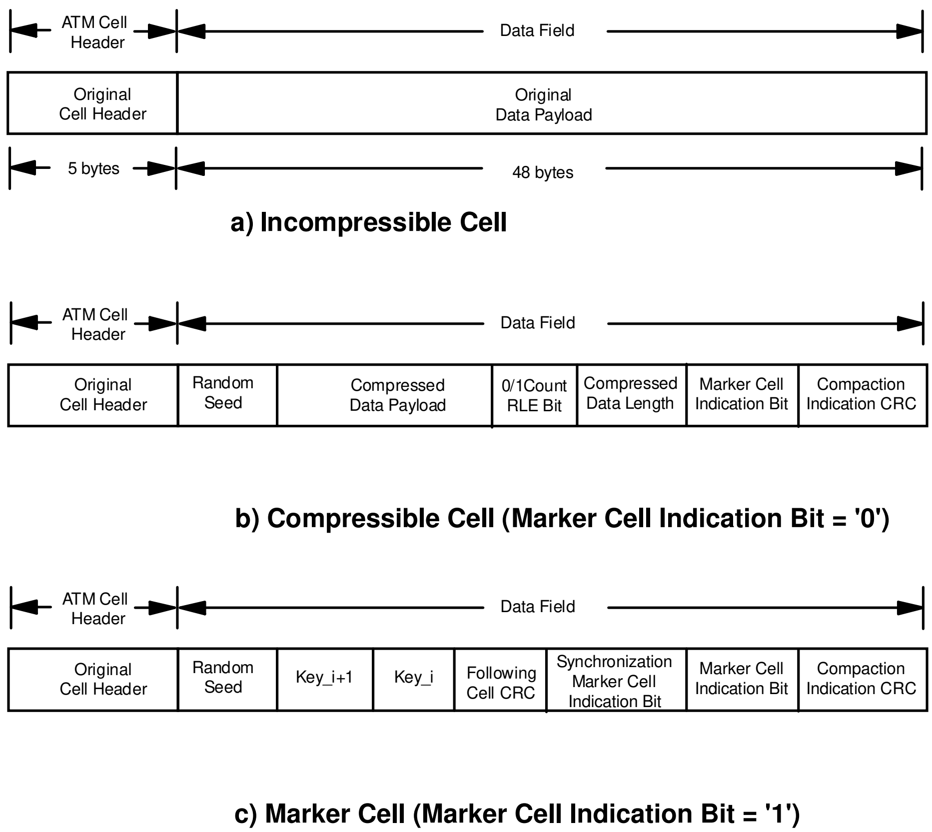

Figure 1: Formats of Cells.

The flow-chart in Figure 2 describes the encryption process. First the data field of the cell is compressed. If compression frees at least x bits, then the compressed data is placed in CDP subfield. If the RLE algorithm counted consecutive zeros, then 0/1-CRLEB is set to “0”, otherwise if it counted consecutive ones then 0/1-CRLEB is set to “1”. The CDL subfield is set to the length of the compressed data in bits and MCIB to “0”. A random seed is inserted in the RS subfield. Then a CRC is computed over the entire data field excluding the CI-CRC subfield. This CRC is then placed in the CI-CRC subfield. The data field is then encrypted in CBC mode, with initialization vector (IV) set to zero, starting with the first bit after the header (Figure 3). The CI-CRC subfield acts like a compaction indicator to the receiver, i.e., if the CRC computation over the data field (excluding the CI-CRC subfield) yields the same value as in the CI-CRC subfield then the receiver may assume that the data is compressed.

If the compression does not free enough bits then a CRC is computed over the entire data field excluding the CI-CRC subfield position. If this value does not match the content of the CI-CRC subfield position, then the data field of the cell is left unmodified in the form it was

Subfield	Explanation
Compaction Indication CRC (CI-CRC)	This subfield contains a cyclic redundancy check computed over the entire data field excluding the CI-CRC subfield.
Marker Cell Indication Bit (MCIB)	This bit indicates whether this ATM cell is an inserted “marker cell”. This bit is set to ’1’ if this ATM cell is a marker cell; it is set to ’0’ otherwise.
Compressed Data Length (CDL)	This subfield indicates the length of the compressed data in bits.
0/1 Count RLE Bit (0/1-CRLEB)	This bit indicates whether base 0 or 1 of RLE is used in this cell. This bit is set to ’1’ if RLE base 1 is used in this ATM cell; it is set to ’0’ otherwise.
Compressed Data Payload (CDP)	This is the compressed data.
Random Seed (RS)	This is a random number used to psuedo-randomize the data payload.

Table 1: Fields of a compressed cell.

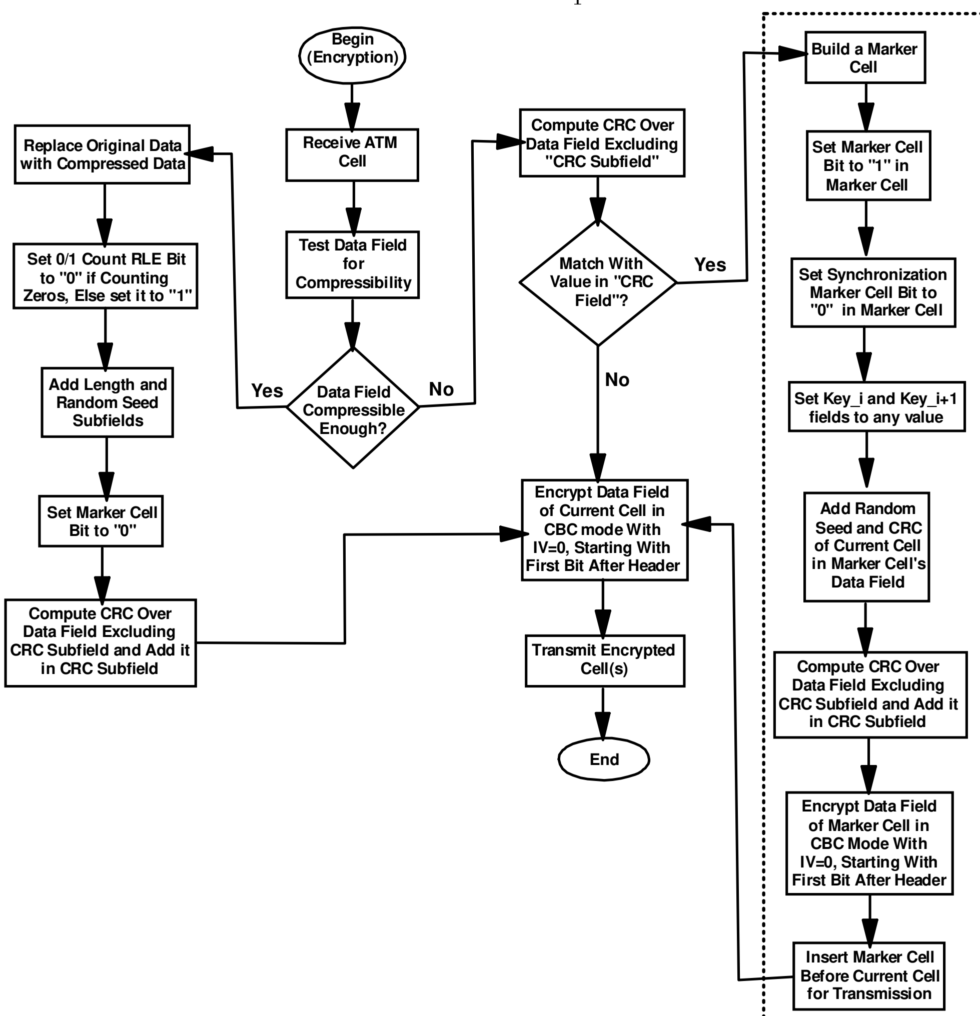

Figure 2: Flow Chart for Encryption for Compaction/Randomization.

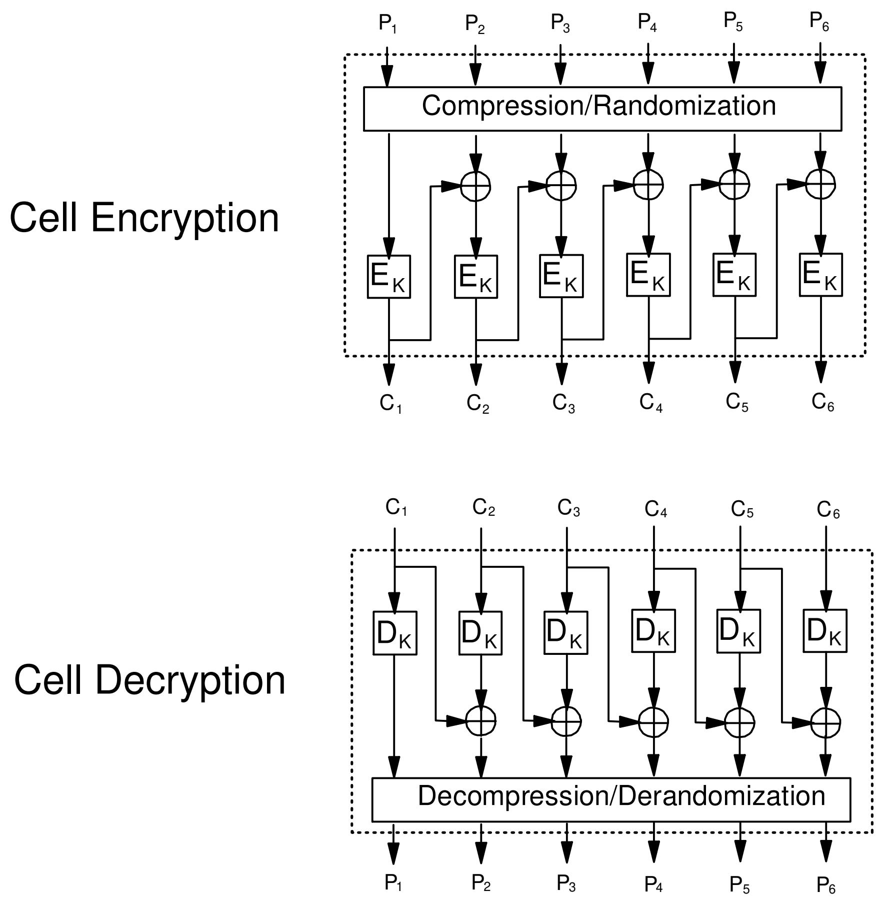

Figure 3: Encryption/Decryption.

Subfield	Explanation
Compaction Indication CRC (CI-CRC)	This subfield contains a cyclic redundancy check computed over the entire data field excluding the CI-CRC subfield.
Marker Cell Indication Bit (MCIB)	This bit indicates whether this ATM cell is an inserted "marker cell". This bit is set to '1' if this ATM cell is a marker cell; it is set to '0' otherwise.
Synchronization Marker Cell Indication Bit (SMCIB)	This bit indicates whether this marker cell is used to synchronize the encryption/ /decryption key update between the two ends of the ATM connection or is used to indicate that the following cell is a non-compressed cell with a CRC matching. This bit is set to '1' if this marker cell is used to synchronize key update; it is set to '0' otherwise. (See Section 3, Key Update Synchronization, for further explanation.)
Following Cell CRC (FC-CRC)	This subfield contains the CRC of the following cell when this marker cell is used to indicate that the following cell is non-compressed but has a CRC matching. It is set to an arbitrary value otherwise.
$\mathbf{Key_i}$	This subfield contains the old encryption/decryption key when this marker cell is used to synchronize key update. It is set to an arbitrary value otherwise. (See Section 3, "Key Update Synchronization" for further explanation.)
$\mathbf{Key_{i+1}}$	This subfield contains the new encryption/decryption key when this marker cell is used to synchronize key update. It is set to an arbitrary value otherwise. (See Section 3, "Key Update Synchronization" for further explanation.)
Random Seed (RS)	This is a random number.

Table 2: Fields of a marker cell.

before compression was attempted, as shown in Figure 1(a). The data field is then encrypted in CBC mode with IV=0, starting with the first bit after the header. When the computed CRC value matches the content of the "CI-CRC subfield" position, then the cell is processed as before with the exception that a "marker cell", constructed as follows, is inserted before the current cell. The marker cell tells the receiver to ignore the CRC matching of the following cell and treat it as a non-compressed cell. The marker cell is discarded by the receiver. The cell header of a marker cell is identical to the cell header of the current cell. The format of the marker cell's data field is shown in Figure 1(c) and is explained in Table 2.

A marker cell's data field is constructed as follows. The CRC of the non-compressed cell with CRC matching, which will follow the marker cell, is placed in the FC-CRC subfield. The MCIB subfield is set to "1". The SMCIB subfield is set to "0". The Key_i and Key_{i+1} subfields are set to arbitrary values. A random seed is added in the RS subfield. Then a CRC is computed over the entire data field excluding the CI-CRC subfield and is placed in the CI-CRC subfield. The data field is then encrypted in CBC mode with IV=0, starting with the first bit after the header. The encrypted marker cell is then transmitted before the cell under consideration.

Without the marker cell, a non-compressed cell that has a CRC matching will be treated as a compressed one by the receiver and as the result will not be decrypted back to the original payload. If the CRC is n bits, then the probability that the CRC of the first $(48 \times 8) - n$ bits of the payload just happens to match the data in the last n bits position of the payload is $2^{-((48\times8)-n)}$, which is very low if the length of CRC is sufficient. [2] Even though the probability that a non-compressed cell has a CRC matching is very low, a scheme (such as the marker cell approach) is needed to avoid this problem. This is because some non-real time applications (such as TCP) may want to provide reliable message delivery system which may not be possible in the presence of this problem – a retransmitted message can still have non-compressed cells with CRC matching.

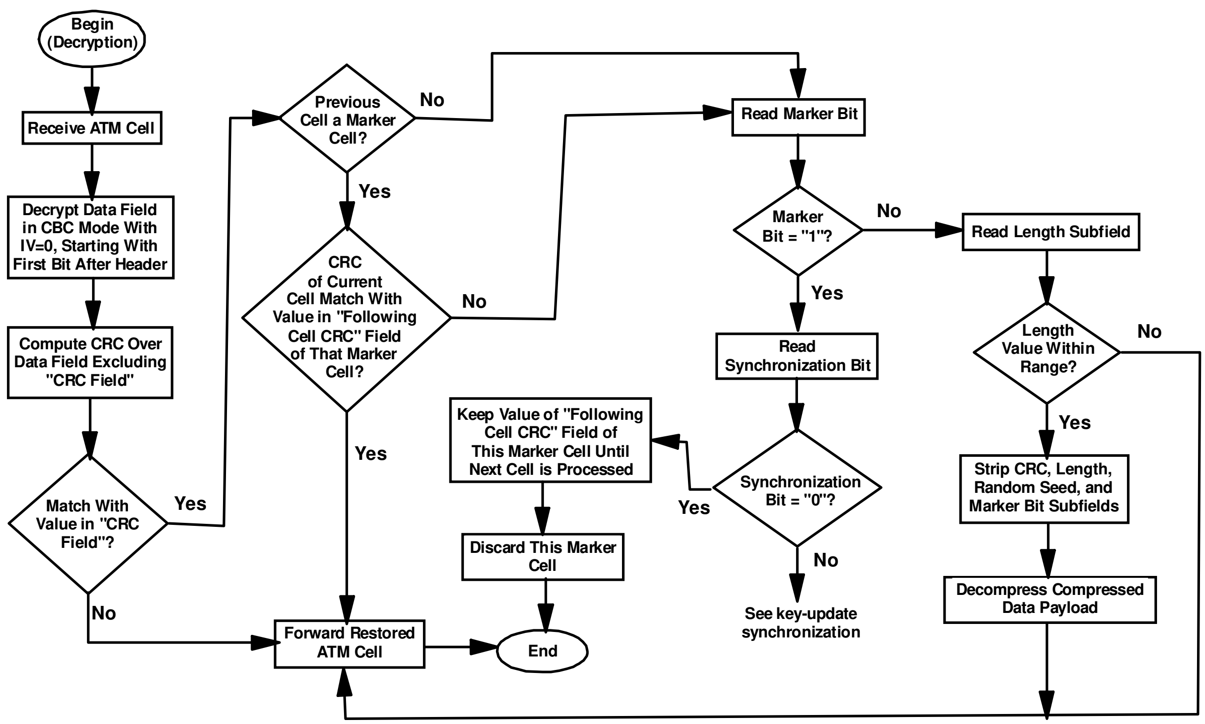

Figure 4: Flow Chart for Decryption for Compaction/Randomization.

The flow-chart in Figure 4 shows the decryption process. The data field of an ATM cell that is received is decrypted in CBC mode with IV=0, starting with the first bit after the header (See Figure 3). Then a CRC is computed over the entire data field excluding the CI-CRC subfield. If this computed value does not match the content of the CI-CRC subfield, then the

[2] The assumption in deriving this probability is that the payload bit patterns are random and the CRC bit patterns are uniformly distributed over all payload bit patterns.

data field contains non-compressed data and no further processing is required. Otherwise, the CRC computed over the entire data field excluding the CI-CRC subfield matches the content of the CI-CRC subfield and there are the following cases:

- If the previous cell was a compaction indicator marker cell, and the CRC for the current cell matches the FC-CRC subfield value in the marker cell, then the data field contains non-compressed data and no further processing is required.
- If the previous cell was a compaction indicator marker cell, and the CRC for the current cell does not match the FC-CRC subfield value in the marker cell, then there is an error situation in which the actual cell that was supposed to follow the marker cell was lost or reordered, and the present cell has to be treated as if the cell preceding it was not a marker cell.
- If the cell preceding this cell was not a compaction indicator marker cell, then if MCIB="1" and SMCIB="0" the current cell is a compaction indicator marker cell. The value of the FC-CRC subfield is extracted and stored until the next cell is processed, and the marker cell is discarded.
- If the cell preceding this cell was not a compaction indicator marker cell, then if MCIB="0" this cell should contain compressed data. Read the CDL field, and if its value is within a valid range, then extract the CDP subfield. Decompress the CDP subfield and place it in the data field. If the value of the CDL subfield is outside the valid range, there is an error situation in which the present cell is not a cell with compressed data but is a non-compressed cell with CRC matching and the compaction indicator marker cell which is supposed to precede the present cell is lost or reordered.

Note that this randomization/compaction encryption mode encrypts each cell's payload independently in CBC mode with IV=0. Cipher feedback (CFB) mode with IV=0 can also be used. But CFB is not as popular as CBC (and the width of feedback is not standardized). Some other modes such as output feedback (OFB) and Counter are not suitable because they do not spread the effect of the random seed over the entire cell payload.

2.2 Run Length Encoding

Run Length Encoding (RLE) is a simple method used to encode binary bit strings by compressing either substrings of consecutive zeros (RLE-base-0) or substrings of consecutive ones (RLE-base-1).

In RLE-base-0, each k-bit symbol indicates the number of consecutive "0" bits that occurred before the next "1" bit. To handle long "0" runs, the symbol consisting of all "1" bits means that the true distance is plus the value of the following symbol or symbols. For example, the bit string

100010000000000000010000001000000000000000100000010001000000011010000010 1

consists of "0" runs of length 0, 3, 13, 6, 14, 6, 3, 7, 0, 1, 5, and 1. It will be encoded using RLE-base-0 with a 3-bit symbols as

000011111110110111111000110011111000000001101001

for a saving of 32%. The original bit string is 71 bits and the compressed one is 48 bits.

Note that decompression of a bit string encoded using RLE-base-0 always ends with a "1". This is because RLE-base-0 counts the number of consecutive "0" bits that occurred before the next "1" bit. Therefore a receiver (that receives a compressed bit string) needs to use an additional mechanism to find out whether the original bit string actually ended with a "1". For ATM, the mechanism is very simple. Since the ATM cell data field has a fixed size of 384 bits, unless an error has occurred, the decompression process must always yield either 384 (when the original bit string ends with a "1") or 385 bits (when the original bit string ends with a "0"). When decompression results in 384 bits no further action is necessary. However, when it results in 385 bits, then the last bit (i.e., the 385th bit which is supposed to be a "1") is removed.

RLE-base-1 works exactly the same way as RLE-base-0 but it counts the number of consecutive "1" bits that occurred before the next "0" bit. To increase the probability that an ATM cell payload could be compressed, RLE-base-0 should be used when the cell payload consists mostly of "0" bits and RLE-base-1 when the cell payload consists mostly of "1" bits. In a compressed cell, the 0/1-CRLEB field (as described above) indicates to the receiver whether base 0 or 1 of RLE is used on a cell.

The ATM cell's data field has a fixed size of 384 bits. When the data field consists of all "0"s or all "1"s, the compressed data field would have a minimum length of 165 bits.

One could argue that RLE may not give good compression gains. There are other algorithms that could potentially give better compression gains. For example, adaptive Huffman coding based on statistical modeling, arithmetic coding, and adaptive dictionary coding schemes all yield good compression gains ([14] contains a good comparison study of compression techniques). However, for this randomization/compaction mode of operation for ATM cell encryption, the compression scheme needs to be simple and scalable. RLE is both simple and scalable.

3 Encryption/Decryption Key Update Synchronization

In this section, we devise a modification to the method of encrypting ATM cells, discussed in Section 2, to solve the following problem. The keys used for encryption/decryption have specific lifetimes because the longer a key is used, the greater the chances of a successful attack. The lifetime of a key is determined by factors such as the encryption algorithm used, the degree of confidentiality of the encrypted data, and the amount of data encrypted. For example, session encrypting keys have a shorter lifetime, such as a day, than key-encrypting keys which may be updated on a monthly basis. As ATM connections can be up for extended periods of time and have very high transmission rates, their session encryption keys need to be updated frequently. The two ends of an ATM connection can agree on and exchange the new key in a reliable and secure manner by using authentication protocols such as Kerberos [21], SPX, and KryptoKnight. However, there exists the problem of how the two ends of the ATM connection can reliably and securely synchronize the key update so that both ends begin using the new key for encrypting and decrypting the data payload of the ATM cells from the same ATM cell onwards.

One way to achieve this is to use a "Key Sync Bit" in the ATM cell header. When the sender begins encryption with the new key, it uses the complement of the value of the Key Sync Bit that was used with the old key. When the receiver detects a cell whose Key Sync Bit is the complement of the value on the prior cells, it uses the new decryption key from then on. The usage of a bit in the ATM cell header to perform the key update synchronization is not practical since no bit in the cell header is available. So, we propose a solution that does not require a bit in the ATM cell header to perform the key update synchronization. Specifically, the synchronization is achieved by the usage of a "marker cell" which contains the old and the new keys. An important aspect of the solution is the ability to distinguish such cells from other encrypted cells without any changes to the cell header or ATM infrastructure.

3.1 Algorithm

The secure and reliable synchronization of the usage of new encryption/decryption keys between the two ends of an ATM connection is achieved by using a "key update synchronization marker cell" which contains the old key K_i and the new key K_{i+1}. The marker cell is identifiable by the receiver without using any bit in the cell header using the technique of adding a CRC in the payload, as used in Section 2. The formats of the synchronization marker cell prior to encryption, is shown in Figure 1(c). The subfields within the marker cell payload are described in Table 2. The marker cell header is identical to the header of other user cells. A synchronization maker cell is constructed as follows.

The old and new keys are placed in the Key_i and Key_{i+1} subfields of the marker cell, respectively. The FC-CRC is set to any arbitrary value. The MCIB and SMCIB subfields are set to "1". The marker cell payload is randomized to enhance the security and therefore the confidentiality of the keys against attack. The randomization is achieved by introducing a random seed in the RS subfiled. The RS subfield is the left-most position in the data field for the effect of the random seed to propagate using the cipher block chaining performed later. A CRC is then computed over the entire data field excluding the CI-CRC subfield and is placed in the CI-CRC sufield. The data field is then encrypted using the old key (Key_i) in CBC mode with IV=0, starting with the first bit after the header. When the sender begins encryption with the new key (Key_{i+1}), it transmits this key update synchronization marker cell. This marker cell acts as an indicator to the receiver that from next cell onwards, the new key should be used for decryption.

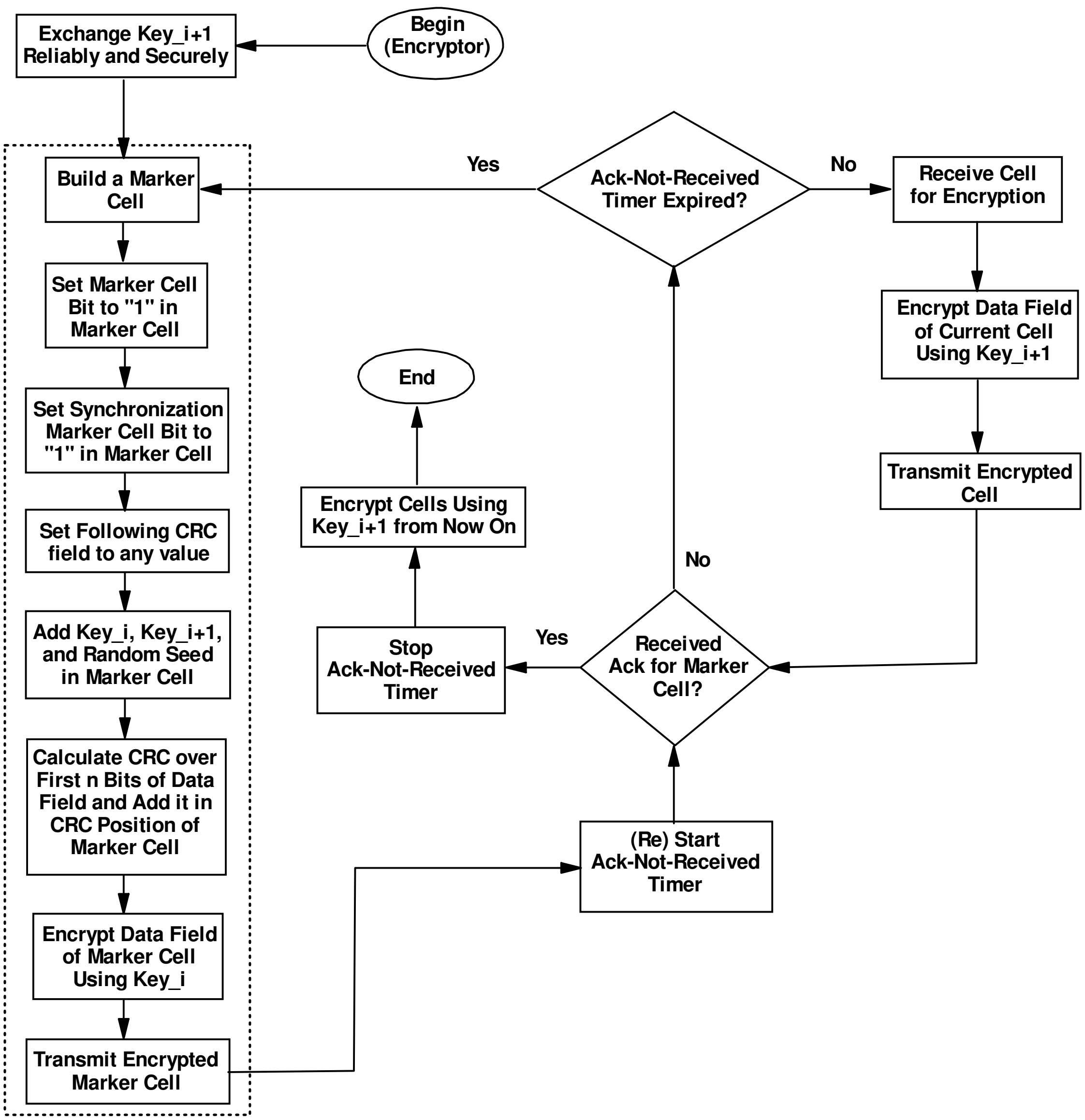

Figure 5: Flow Chart for Encryption for Key Update Synchronization.

The flow-chart in Figure 5 shows the processing for key update synchronization at the sender. When the sender encrypts a key update synchronization marker cell, it uses Key_i for encryption.

At the time it transmits the cell, it starts a timer $Ack\_Not\_Received$, and all subsequent cells are encrypted using Key_{i+1}. If the receiver successfully processes the key update synchronization marker cell, it uses Key_{i+1} for decryption henceforth and it sends back a reliable acknowledgement message, otherwise it continues to decrypt cells using Key_i. If the sender does not receive an acknowledgement within an appropriate timeout period, it rebuilds a key update synchronization marker cell, encrypts it with Key_i, sends it, restarts the timer, and encrypts subsequent cells with Key_{i+1}. This process is repeated until an acknowledgement is received. It is only in the case of the marker cell getting lost that the acknowledgement is not received by the sender. In this case, the cells sent after the marker cell and until the timer $Ack\_Not\_Received$ times out are lost. However, this error situation is rare.

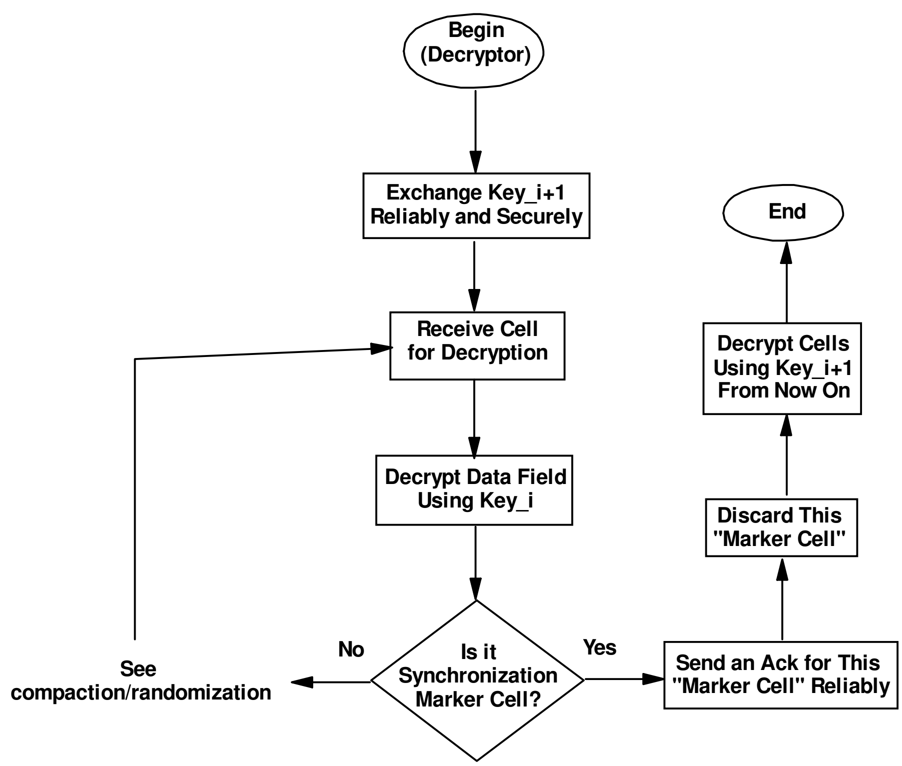

Figure 6: Flow Chart for Decryption for Key Update Synchronization.

The flow-chart in Figure 6 shows the processing for key update synchronization at the receiver. The data field of of an arriving ATM cell is decrypted using Key_i in CBC mode with IV=0, starting with the first bit after the header. Then a CRC is computed over the entire data field excluding the CI-CRC subfield position. If this computed CRC value does not match the content of the CI-CRC subfield position, then the cell contains regular data, otherwise it needs further examination to determine its contents. When the computed CRC matches the value in the CI-CRC subfield position, the following types of cells are distinguished from each other as indicated below:

- A key update synchronization marker cell – SMCIB = "1"; MCIB = "1"; and if the previous cell was a compression marker cell, the CRC of this cell does not match the value in the FC-CRC subfield of that compression marker cell. The values in the Key_i and Key_{i+1} subfields must match the known values of Key_i and Key_{i+1}; otherwise there is an error situation in which the preceding cell which should have been a compression marker cell was lost or reordered, and this present cell is actually a cell containing non-compressed data but has a CRC matching.

- A cell whose data has been compressed/randomized – MCIB = "0"; and if the previous cell was a compression marker cell, the CRC of this cell does not match the value in the FC-CRC subfield of that compression marker cell.

- A cell which contains non-compressed data but has a CRC matching – This cell is distinguishable because it immediately follows a compression marker cell and the CRC of this cell matches the value in the FC-CRC subfield of that compression marker cell.

- a compaction indication marker cell – the compression marker cell has SMCIB = "0" and MCIB = "1". and if the previous cell was a compression marker cell, the CRC of this cell does not match the value in the FC-CRC subfield of that compression marker cell.

When the receiver successfully process a key update synchronization marker cell, it sends back a reliable acknowledgment message and uses the new key (Key_{i+1}) for decryption of subsequent ATM cells.

4 Conclusions

This paper presented a secure, scalable, and self-synchronizing mode of operation for block-encryption of ATM cells which converts plaintext to pseudo-random plaintext before ciphering to conceal patterns in the plaintext. This mechanism is based on data compression and pseudo-randomization of the data payload of ATM cells. A notable feature of the presented mechanism is that it provides a means for encoding the information in the data field as to whether the data field of the cell had undergone compression/randomization. The presented mode of operation was shown to have the following attractive features: (i) plaintext patterns are pseudo-randomized and chained with ciphertext (thereby, preventing against "dictionary", "known plaintext", and "statistical analysis" attacks), (ii) it is self-synchronizing, (iii) cell loss has no additional negative effect, (iv) no IV (Initialization Vector) storage is required, (v) it is encryption-algorithm

Mode	Security	Implementation	Fault Tolerance	Crypto Synchronization
ECB	(-) plaintext patterns are not concealed	(+) no feedback (+) no IV storage (+) encryption and decryption are parallelizable	(+) cell loss has no additional negative	(+) self synchronizing
CBC, CFB	(+) plaintext patterns are concealed	(-) feedback from encryption output (-) IV storage (-) encryption is not parallelizable (+) decryption is parallelizable	(+) cell loss causes one additional block of plaintext to be corrupted	(+) self synchronizing
OFB	(+) plaintext patterns are concealed	(-) feedback from encryption output (-) IV storage (-) Encryption and decryption are not parallelizable	(-) cell loss causes loss of crypto synchronization	(-) requires periodic resynchronization
Counter	(+) plaintext patterns are concealed	(-) feedback from encryption output (-) IV storage (+) Encryption and decryption are parallelizable	(-) cell loss causes loss of crypto synchronization	(-) requires periodic resynchronization
Compaction/ Randomization (proposed)	(+) plaintext patterns are concealed	(+) no feedback (+) no IV storage (+) Encryption and decryption are parallelizable	(+) cell loss has no additional negative effects	(+) self synchronizing

Table 3: Comparison of Existing Modes of Operation with proposed Compaction/Randomization Mode.

independent, (vi) there is no cell-to-cell dependency (no feedback from previous cells), and (vii) it is highly scalable (i.e., cells from the same stream can be ciphered and deciphered in parallel). Table 3 summarizes a comparison of the proposed randomization/compaction mode with some well-known existing modes, namely, ECB, CBC, CFB, OFB, and Counter. The presented scheme is seen to be superior to known existing modes of operation such as ECB, CBC, CFB, OFB, and Counter, each of which has only some of the above-listed attractive features.

Based on the proposed technique of compression/randomization for encryption, the paper also presented a secure mechanism for in-band synchronization of encryption/decryption key updates. The mechanism used a marker cell within the data channel, whose original data payload that contains the old and new keys is pseudo-randomized and subsequently block ciphered before transmission. An important aspect of the solution is the ability to distinguish the marker cell from the other encrypted cells by encoding the information in the data field as to whether the cell is a marker cell.

Thus, both the above mechanisms do not require any changes to the cell header or the ATM infrastructure. The approaches given in this paper are applicable not just to ATM connections but to connections using any other cell-based communication protocol.

References

[1] ATM Forum, "Phase I ATM Security Specification (Draft)", ATM Forum/95-1473R4, Aug. 1996.

[2] R. Ballart, C. Ching, "SONET: Now It's the Standard Optical Network", IEEE Communications, Mar. 1989.

[3] D. Bertsekas, R. Gallager, "Data Networks", Englewood Cliffs, NJ, Prentice-Hall International, 1987.

[4] S. Chuang, "Securing ATM Networks", Tech. Report, University of Cambridge Computer Laboratory, Oct. 1995. (Available from http://www.cl.cam.ac.uk/Research/SRG/greenbook.html)

[5] R. Deng, L. Gong, A. Lazar, "Securing Data Transfer in Asynchronous Transfer Mode Networks", IEEE Globecom'95, Nov. 1995, 1198-1202. (Full paper appears as TR95-189, Inst. of Systems Science, National University of Singapore.)

[6] D. Denning, "Cryptography and Data Security", Addison-Wesley, Reading, MA, 1982.

[7] D. Denning, P. Denning, "Data Security", ACM Computing Surveys, Sept. 1979.

[8] W. Diffie, M. Hellman, "Exhaustive Cryptoanalysis of the NBS Data Encryption Standards", IEEE Computer, June 1977.

[9] M. R. Garey and D. S. Johnson, "Computers and Interactability", W.H. Freeman, San Francisco, 1979.

[10] S. Goldwasser, S. Micali, "Probabilistic Encryption", Journal of Computer and System Sciences, April 1984.

[11] ITU TSS Study Groups 13, Contribution Number T1S1.5/93-181.

[12] B. Lyles, "Authenticated Signaling for ATM", T1S1.5/94-118, Xerox Parc.

[13] D. E. McDysan and D. L. Spohn, "ATM: Theory and Application", McGraw-Hill, New York, 1994.

[14] M. Nelson, "The Data Compression Book", 2nd. edition, 1995

[15] "PNNI Draft Specification", ATM Forum 95-0471R14, December 1995.

[16] G. Popek, C. Kline, "Encryption and Secure Computer Networks", ACM Computing Surveys, Sept. 1979.

[17] R. Rivest, A. Shamir, L. Adleman, "A Method for Obtaining Digital Signatures and Public Key Cryptosystems", Communications of the ACM, Feb. 1978.

[18] B. Schneier, "Applied Cryptography", Wiley, New York, 1994.

[19] J. Seberry, J. Pieprzyk, "Cryptography: An Introduction to Computer Security", Prentice-Hall, Englewood Cliffs, N.J., 1989.

[20] C. Shannon, "Communication Theory of Secrecy Systems", Bell Systems Journal, Oct. 1949.

[21] J. Steiner, C. Neuman, J. Schiller, "Kerberos: An Authentication Service for Open Network System", Proc. of the Winter USENIX Conf., Feb. 1988.

[22] D. Stevenson, N. Hillery, G. Byrd, "Secure Communication in ATM Networks", Communications of the ACM, 38(2), 45-52, Feb. 1995.

[23] U.S. National Bureau of Standards, "Data Encryption Standard, Ferderal Information Processing Standards", (FIPS PUB) 46, Jan 1977.

[24] U.S. National Bureau of Standards, "DES Modes of Operation, Ferderal Information Processing Standards", (FIPS PUB) 81, Dec. 1980.