Stream or Block Cipher for Securing VoIP?

Ashraf D. Elbayoumy and Simon J. Shepherd
(Corresponding author: Ashraf D. Elbayoumy)

Advanced Signals Laboratory, School of Engineering Design & Technology
University of Bradford, BD7 1DP, UK (Email: ademahmo@bradford.ac.uk)

(Received Nov. 30, 2005; revised and accepted Jan. 27 & Mar. 3, 2006)

Abstract

When the network is homogeneous, choosing the cipher type for a packet telephony application is simple. It is clear that stream ciphers perform better than block ciphers over landline, circuit-switched networks, since loss is negligible in these networks but corruption is not. Likewise, it is also clear that block ciphers perform better than stream ciphers over landline, packet-switched networks, since corruption is negligible in these networks but loss is not [9]. However, the choice of cipher is not so clear for a heterogeneous internetwork containing a mix of packet and circuit-switched networks. Additionally, this issue becomes even more confounded when heterogeneous internetwork also consists of wireless links. Existing encryption systems will degrade performance in a heterogeneous internetwork because such internetworks have appreciable loss and corruption. Thus, the error properties would degrade the subjective quality of the packet telephony application. In this paper we present an experimental results of comparing block and stream ciphers when used to secure VoIP in terms of end-to-end delay and subjective quality of perceived voice. We proposed a new technique, which provides automatic synchronization of stream ciphers on a per packet basis, without the overhead of an initialization vector in packet headers or without maintaining any state of past-encrypted data. We show that this technique mitigates the trade-off between subjective quality and confidentiality.

Keywords: AES, CBC, MOS, QoS, VoIP

1 Introduction

Security is a serious bottleneck for the future of VoIP. Because of the time-critical nature of VoIP most of the same security measures currently implemented in today's data networks could not be used in VoIP networks.

Encryption algorithms are classified into two categories: secret key and public key. Secret-key algorithms use the same key for encryption and decryption, and these algorithms usually perform bulk encryption of sensitive data. In contrast, public-key algorithms use one key (usually called the public key) for encryption and another key (usually called the private key) for decryption. Because of their slower performance, public-key algorithms usually perform encryption/decryption of sensitive credentials exchanged during authentication and/or key-exchange protocols but can't be used in a delay sensitive application like VoIP. So that we have to use secret-key algorithms for securing VoIP. Secret-key algorithms fall into two categories: block ciphers and stream ciphers. Block ciphers encrypt/decrypt blocks of bits at a time and stream ciphers encrypt/decrypt a single bit at a time.

In our recent work [1, 2, 3] aiming to implement a high grade secure VoIP system it has been shown that QoS is highly affected by adding confidentiality to VoIP systems. We tried to find the best QoS control mechanism and the best encryption algorithm to secure VoIP system. Our results showed that the computationally lighter algorithms achieved better throughput than the more expensive ones, but there is one question still in mind, which algorithms are better for securing Internet Telephony applications stream ciphers or block ciphers?

From an error propagation point of view many researchers show that block ciphers do not perform well for applications used in environments with high bit error rates because block ciphers multiply bit errors; that is, a single bit error in the encrypted data received at the input to the decryptor will result in multiple bit errors in the recovered plaintext at the output of the decryptor. That is due to the diffusion properties of any block cipher.

In contrast, stream ciphers do not multiply bit errors, but do propagate synchronization errors caused by insertion or deletion of bits. That is, if the decryptor loses synchronization with the encryptor, the decryptor will garble all the recovered plaintext bits after the synchronization error until synchronization is restored. For an Internet telephony service using stream encryption for confidentiality, in the worst case, if the encryption system does not restore synchronization, then the recovered speech would produce a persistent static sound. Because of this property, stream algorithms do not perform well for applications used in environments where the packet-loss rate is potentially high.

The choice of cipher for the different internetwork types is not so clear. Probably the most common example is a landline circuit and packet-switched internetwork, where one might connect to an ethernet LAN through PTSN using a voice-band data modem. This exact scenario applies to tens of millions of Internet users. Using a voice-band data modem, the user connects to his/her Internet service provider (ISP) through PTSN. Typically, their ISP is either a commercial organization that provides the service as a business, or some other organization that provides the service free of charge for their employees, members, and affiliates. Once the user has connected to their ISP, they have access to the resources that reside on their ISP's LAN, as well as access to the Internet through their ISP's Internet gateway.

The type of encryption system to deploy is not obvious because an end-to-end connection over this internetwork has both non-negligible loss and non-negligible corruption. Thus, a stream cipher would exhibit its error properties in the packet-switched part of the internetwork and a block cipher would exhibit its error properties in the circuit-switched part of the internetwork. Despite this fact, many packet telephony applications available today (which provide confidentiality) use block ciphers. This is so because the characteristics of a telephone channel vary slowly with time. Once a connection is established, we can assume that the underlying channel is stationary over a short duration (less than an hour). Therefore, for a connection over PTSN with a short duration, one could argue that corruption in this network is negligible. If we assume that the typical user only has one telephone line and does not wish to tie this line up indefinitely, then this argument is probably valid. However, with the explosion of the Internet, users are staying online much longer and many households have dedicated modem lines, which stay connected twenty-four hours a day. Thus, for long sessions, this argument is not valid and corruption in PTSN is non-negligible. Because of such internetworks, which have appreciable loss and corruption, existing encryption systems with these error properties will degrade the subjective quality of the packet telephony application.

In this paper we will compare block and stream ciphers when used to secure VoIP in terms of end-to-end delay and subjective quality of perceived voice. Our results show that the end-to-end delay and subjective quality of perceived voice are better in case of stream ciphers before loss of synchronization than in block ciphers.

If we take into account a heterogeneous network that includes wireless links, in such an environment the end-to-end path is subject to loss, corruption and re-ordering of packets. Because of corruption, block ciphers are not well suited for this scenario.

Although stream ciphers tend to have faster and more efficient (i.e., smaller code size) software implementations than block ciphers, Internet telephony applications to date exclusively use block ciphers to specifically avoid the problems of synchronization associated with stream ciphers.

Finally, we present a new technique that provides automatic synchronization of stream ciphers on a per packet basis, without the overhead of an initialisation vector in packet headers or without maintaining any state of past-encrypted data. We show that this technique mitigates the trade-off between subjective quality and confidentiality and more efficient for different internetwork scenarios.

This paper is organized as follows; Section 2 presents a quick overview of the error properties of secret-key ciphers, Section 3 describes the experimental environment, Section 4 presents the experimental results, Section 5 describes our proposed resynchronization technique, Section 6 explains some security considerations, and Section 7 concludes the paper and summarizes our findings.

2 Error Properties of Secret-key Ciphers

Secret-key algorithms perform bulk encryption of sensitive data in real-time applications such as VoIP. In contrast, because of their slower performance, public- key algorithms are usually reserved for non-real-time applications, such as providing confidentiality of sensitive credentials exchanged during authentication protocols.

2.1 Expansion of Bit Errors

Cryptographers design block algorithms to satisfy the *avalanche effect* [10]. The avalanche effect states that an average of one-half of the output bits should change whenever a single input bit changes. This property is a necessary condition for security and exhibited by all block algorithms that have found their way into practice [8]. It is a desirable property because it says that each output bit must depend on all the input bits. In other words, an algorithm that has this property does not exhibit any statistical correlation between input and output that an adversary might use in an attack. The consequence of this property is that block ciphers multiply bit errors. That is, a single bit error in the ciphertext received at the input to the decryptor will result in multiple bit errors in the recovered plaintext at the output of the decryptor.

Because of this property, block ciphers are seldom used for real-time services in environments that have appreciable corruption. However, block ciphers do perform well in environments that have appreciable loss, but negligible corruption. In the case of VoIP, the latter statement is true as long as the VoIP service maintains the block framing within a voice packet.

2.2 Propagation of Synchronization Errors

In contrast, stream ciphers do not multiply bit errors, but do propagate synchronization errors caused by insertion or deletion of bits. That is, if the decryptor loses synchronization with the encryptor, the decryptor will garble all the recovered plaintext bits after the synchronization error until the system restores synchronization. Because of this property, environments where packet re-ordering is common and packet loss is appreciable confound the task of re-synchronization.

3 Experimental Environment

An active VoIP QoS application measurement was performed with visual C++ software that transmits and receives full duplex VoIP streams between two hosts connected via an IP network. The program reads and encapsulates audio packets from the microphone of Host 1, send the packets to a remote Host 2. The two hosts are synchronized using NTP (Network Timing Protocol) because time synchronization provides receivers with precise information about end-to-end delays [7].

Packets are time-stamped and sequence-numbered so that various path criteria such as latency, jitter, packet loss, out of order packets, can be calculated.

We used the GSM 06.10 format, which is the speech compression algorithm used in GSM (the European standard for digital cellular telephones). GSM 06.10 is a 13 kbps, *lossy*, low bitrate, speech coder. It compresses 320 8-bit pulse-code modulated (PCM) samples into 260-bit GSM frames, which is a compression ratio of about 10:1. We chose this codec for our simulation because it is a one of the highest quality, error robust, low bit rate speech coders amongst the competing cellular standards. Additionally, it is widely used in practice.

4 Experimental Results

In order to compare between block and stream ciphers when used to secure VoIP, all packets are encrypted using the cryptographically powerful and computationally efficient AES cipher running in block cipher mode then in stream cipher mode. The AES is taken from the C++ open source cryptographic framework Crypto++ [4].

In a stream cipher mode the encryption is done by bitwise XORing the payload of the packet with a generated keystream segment. This is called an additive stream cipher.

As shown in Figure 1 AES in counter mode acts as a *keystream generato*r producing a pseudo-random keystream of arbitrary length that is applied in a bitwise fashion to the RTP payload by means of a logical XOR function, thus working as a classical stream cipher. AES itself is a block cipher with a block size of 128 bits and a key size of 128, 192, or 256 bits. In order to work as a pseudo-random generator AES is loaded at the start of each RTP/RTCP packet with a distinct initialisation vector (IV) that is derived by hashing a 112-bit salt_key, the synchronisation source identifier (SSRC) of the media stream, and the packet index. Encrypting this IV results in an output of 128 pseudo-random bits.

Next the IV is incremented by one and again encrypted, thus generating the next 128 bits of the keystream. By counting the IV up by increments of one

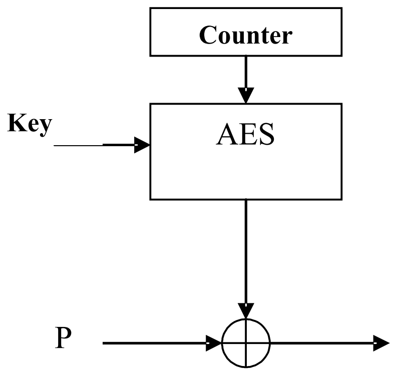

Figure 1: AES in counter mode acts as a keystream generator for an additive stream cipher

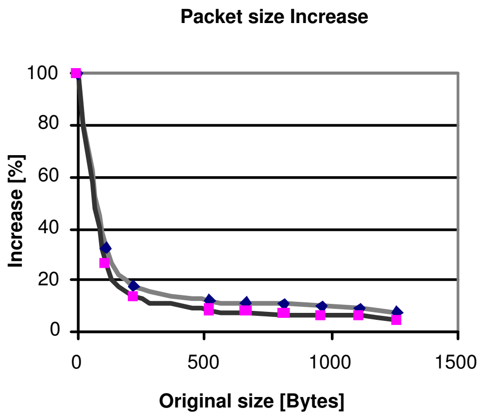

Figure 2: Packet size increase for block cipher mode and stream cipher mode, as a function of packet size in bytes

as many keystream blocks can be generated as are required to encrypt the whole RTP/RTPC payload. Any remaining bits from the last keystream block are simply discarded.

We now investigate the impact of both stream and block ciphers modes to encrypt the payload on the packet size and end-to-end delay.

1) *Packet size:*
   Our results show that the impact of block cipher mode on the packet size is greater than stream mode; especially as the packet size increases. "Figure 2" shows the percentage increase in packet size as a function of the original packet size for block cipher mode (top line) and for stream mode (bottom line).

   The packet size increase has negative effects not only on the bandwidth usage but it also impacts on the transmission delay, router internal delays, queuing delay, thus affecting jitter and overall packet delay.

2) *Crypto-engine:*
   In order to measure the maximum encoding rate,

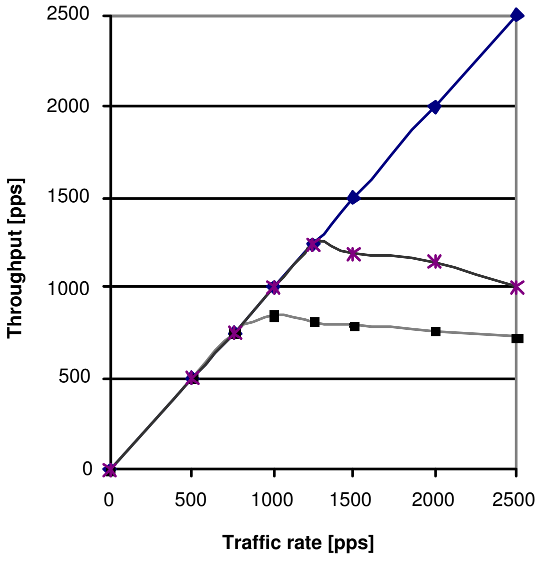

Figure 3: Throughput of the crypto-engine in pps as a function of linearly increasing traffic in pps for plain and encrypted traffic.

when both algorithms are used, we performed the following experiments. We considered both cryptographic algorithms and for each case we generated 4 packet flows with packets of size 60, 100, 250, 1000 bytes, respectively. Each flow starts from 0 pps and increases its rate of 25 pps every 30 s in order to saturate the crypto-engine. Figure 3 graphs the measured throughput as a function of the global traffic flow.

The straight line is the throughput for transmission of packets in the clear; therefore it increases linearly with traffic. The figure shows that when encryption is performed, throughput levels off or decreases after reaching a maximum value, which depends on the algorithm. It also shows that longer packets significantly improve the crypto-engine performance. It is clear that the performance of stream cipher mode (top line) is better than block cipher mode (bottom line) before stream cipher loss synchronization.

The negative slope throughput exhibits after reaching the maximum is due to packets discarded by the engine because it is saturated. Discarded packets contribute to lower the quality of the signal during the reconstruction phase.

In order to evaluate the effect of packet loss using both cryptographic algorithms on the QoS degradation we implement the mean opinion score (MOS) test (Figure 4). In voice communications, particularly Internet telephony, MOS provides a numerical measure of the quality of human speech at the destination end of the circuit. The scheme uses subjective tests (opinionated scores) that are mathemati-

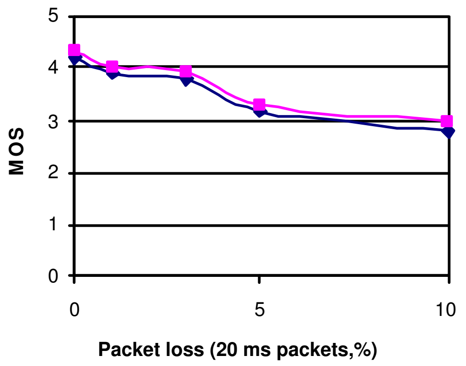

Figure 4: The effect of packet loss on the perceived quality

cally averaged to obtain a quantitative indicator of the system performance. It ranges from 1 to 5, 1 being the worst case. All traffic streams in our test are using G.711 PCM CODEC.

As shown from the results above, the end-to-end delay and subjective quality of perceived voice are better in case of stream ciphers than in block ciphers because stream cipher mode has the big advantage that the keystream can be precomputed before the payload becomes available thus minimizing the delay introduced by encryption. And ofcource by using a stream cipher instead of block cipher there is no need to pad the payload up to multiple of the block size which would add 15 overheaded bytes to the RTP packet in the worst case.

5 Our Proposed Resynch Technique

In this section, we will discuss the Automatic Synchronization Protocol (ASP), which is a technique that makes synchronous stream ciphers robust to synchronization errors. Our goal is to design a synchronization technique for synchronous stream cipher used in real-time, continuous media communications.

We say that the encryptor is synchronized with the decryptor when their states are identical for each corresponding cycle. In general, the decryptor will not be synchronized whenever there is a loss (or insertion) in the ciphertext stream. Although it is possible for losses to occur at the granularity of a bit, it is more common for losses to occur in larger data units like a packet.

Synchronous stream ciphers propagate synchronization errors. The mean length of a synchronization error event is a monotonically increasing function of the average packet loss rate (PLR) and the synchronization period (T), which is some integral number of packets. Thus, it is not advisable to use a synchronization period greater than one packet, particularly in networking environments that have non-negligible loss and corruption. ASP satis-

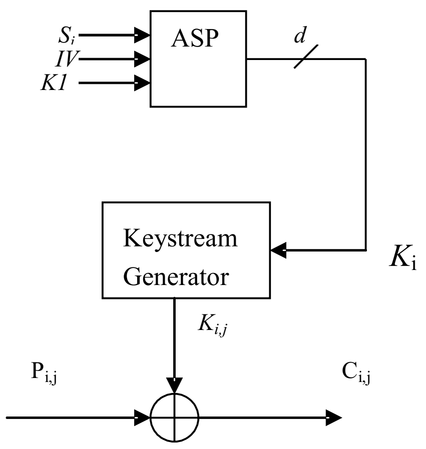

Figure 5: AGeneral description of ASP operation

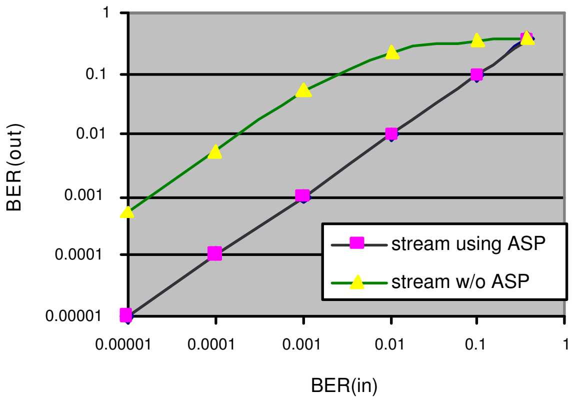

Figure 6: The average post-decryption BER for the three test cases

fies this objective by exploiting the existing packet header structure to synchronize the decryptor on a per packet basis, without adding any cipher synchronization bits to the packet or packet header.

In our synchronization technique we maintain synchronization at the decryptor by re-initializing the cipher for each received ciphertext packet to a different starting point by seeding it with a new key. Figure 5 depicts how ASP maintains synchronization for synchronous stream ciphers.

The ASP module produces a d-bit secret key (K_i) for each received ciphertext packet. Each K_i then re-initializes the keystream generator to a new starting point at the start of each packet. ASP takes as input an s-bit sequence number (S_i) that it extracts from each packet header, an n-bit random initialization vector (IV), and a k-bit secret key ($K1$). Both IV and $K1$ are exchanged during session establishment and last for the duration of the session.

We adopted our Internet Telephony software application by integrating ASP for error robust confidentiality and by modifying the packet parsing code to provide error checking on the RTP packet headers. Then we ran our application program for three test cases: 1) no encryption, 2) stream cipher with ASP 3) stream cipher without ASP. In Figure 6, we plot the results of this experiment for the average post-decryption bit error rate BER (BERout) versus the average pre-decryption BER (BERin). The results for the test case with ASP is indistinguishable from the test case without encryption. That is, there is zero empirical and subjective error expansion with ASP. In contrast, the third test case exhibits the familiar error expansion.

Our experiments verify that ASP immediately restores synchronization upon reception of the first valid RTP packet after a synchronization error occurs thereby solves the synchronization problem associated with synchronous stream ciphers by eliminating any error propagation. Subsequently, ASP-based synchronous stream ciphers can now be deployed in continuous-media applications to provide end-to-end confidentiality, without degrading subjective quality or sacrificing traffic capacity.

6 Security Considerations

Additive stream ciphers do not provide any security service other than confidentiality. In particular, they do not provide message integrity. If message integrity is required, it can be provided through the use of an authentication transform. Such a transform SHOULD be used.

An additive stream cipher is vulnerable to attacks that use statistical knowledge about the plaintext source to enable key collision and time-memory tradeoff attacks [6]. These attacks take advantage of commonalities among plaintexts, and provide a way for a cryptanalyst to amortize the computational effort of decryption over many keys, thus reducing the effective key size of the cipher. Protection against such attacks can be provided simply by increasing the size of the keys used. We encourage the use of keys that are as large as possible, and note that in many cases increasing the key size of a cipher does not affect the throughput of a cipher.

Block cipher encryption has the advantage that it is not as vulnerable to typical plaintext attacks as is encryption with an additive stream cipher. This is because block cipher encryption (CBC mode) is randomized through the use of an unpredictable IV. However, additive stream cipher encryption can achieve the same level of security as CBC mode encryption through an increase in key size [6]. This strategy of “putting all of the randomization in the key” provides an encryption method that can be as secure as CBC, while providing the advantages of using additive stream cipher outlined above.

7 Conclusion

The debate over the relative merits of block vs. stream ciphers for VoIP will no doubt be an ongoing matter. Shamir [5] has long predicted the death of stream ciphers, but current research such as ours into the engineering practicalities of secure VoIP suggest otherwise. As with many practical situations, not everything is black and white. Block ciphers have a place and so do stream cipher. Our research points to an optimum compromise that may give the best of both worlds.

References

[1] A. Elbayoumy and S. Shepherd, "A high grade secure VoIP system using the tiny encryption algorithm," in *Proceedings of 7th Annual International Symposium on Advanced Radio Technologies*, pp. 342-350, Colorado, USA, March 2005.

[2] A. Elbayoumy and S. Shepherd, "QoS control using an endpoint CPU capability detector in a secure VoIP system," in *Proceedings of 10th IEEE Symposium on Computers and Communications*, pp. 175-181, La Magna del Mar Menor, Spain, June 2005.

[3] A. Elbayoumy and S. Shepherd, "A High grade secure VoIP system using an endpoint CPU capability detector," in *Proceedings of ITA05 International Conference on Internet Technologies and Applications Wrexham*, pp. 173-180, North Wales, UK, Sept. 2005.

[4] http://www.eskimo.com/ weidai/cryptlib.html

[5] http://www.iris.re.kr/ac04/data/Asiacrypt2004/Invited%20Talk%201_Adi%20Shamir.pdf

[6] D. McGrew and S. Fluhrer, "Attacks on encryption of redundant plaintext and implications on Internet security," in *Seventh Annual Workshop on Selected Areas in Cryptography*, pp. 125-136, 2000.

[7] H. Melvin and L. Murphy, "Time synchronization for VoIP quality of service," *IEEE Internet Computing*, vol. 6, no. 3, pp. 57-63, June 2002.

[8] National Institute for Standards and Technology, *Data Encryption Standard (DES)*, FIPS PUB 46-2, US Department of Commerece, Dec. 30, 1993.

[9] J. Reason and D. Messerchmitt, "The Impact of Confidentality on Quality of Service in Heterogeneous Voice over IP Networks," in *IEEE Conf. on Management of Multimedia Networks and Services*, pp. 175-192, Chicago, IL, Nov. 2001.

[10] A. Webstar and S. E. Tavares, "On the design of s-boxes," in *CRYPTO'85*, pp. 523-534, Springer-Verlag, NY, 1986.

Ashraf D. Elbayoumy received the B.Eng. degree in electric engineering from the MTC University of Cairo, Egypt, in 1994 and the Master degree in speech security from the MTC University of Cairo, Egypt, in 1999, and the M.Phil. degree in VoIP security from the University of Bradford, UK, in 2004. Currently working toward the Ph.D. degree in the Advanced Signals Laboratory, School of Engineering, Design and Technology, University of Bradford, UK.

Simon Shepherd was educated at the Britannia Royal Naval College Dartmouth and the Royal Naval Engineering College Manadon. He holds a First Class Honours degree in Engineering and a Doctorate in Cryptography and Computing.

Professor Shepherd is a Chartered Engineer, a Chartered Mathematician, a Member of the London Mathematical Society, a Senior Member of the Institution of Electronic & Electrical Engineers, a Fellow of the Institute of Mathematics.

His interests include early music as principal harpsichord of the Bradford Chamber Ensemble, singing with the Bradford Chorale and as Cantoris Bass in the Choir of the Cathedral of St. Peter. He has a passion for the music of Ludwig van Beethoven and has recently performed several of the early sonatas. He enjoys fine food and wine and international travel.