Delay Characteristics of Statistical Cipher Feedback Mode

Howard M. Heys
Electrical and Computer Engineering
Faculty of Engineering and Applied Science
Memorial University of Newfoundland
St. John's, NF, Canada A1B 3X5
howard@engr.mun.ca

Abstract— **In this paper, we investigate the delay characteristics of a mode of operation of block ciphers, referred to as Statistical Cipher Feedback (SCFB) mode. We conclude that SCFB can be operated at rates close to maximum efficiency with modest buffer requirements and delays. As well, we examine the characteristics of the cipher mode for the new Advanced Encryption Standard based on 128-bit blocks and comment briefly on the security of the SCFB mode of operation.**

I. Introduction

Statistical Cipher Feedback (SCFB) mode has been proposed [1] to provide physical layer security for a SONET/SDH environment and is suitable for many other applications as well. It has the benefits of being self-synchronizing and yet being more efficient in its implementation than conventional Cipher Feedback (CFB) mode, a standardized method of configuring a block cipher for use as a stream cipher [2]. In [3], SCFB is analyzed in detail and the efficiency, error propagation characteristics, and synchronization recovery delay properties are discussed based on a 64-bit block cipher such as the Data Encryption Standard (DES) [4]. A mode of operation, with similar objectives and some similar characteristics, referred to as optimized cipher feedback mode, is introduced in [5].

When configuring a block cipher, such as DES or the recently selected Advanced Encryption Standard (AES) [6], as a stream cipher, encryption is performed by XORing the pseudo-random, highly unpredictable sequence of bits (referred to as the keystream) generated at the output of the block cipher with the plaintext bits. The advantages of such configurations can be minimized error propagation (as in the case of Output Feedback (OFB) or counter modes [2]) or self-synchronization to recover from bit slips in the communications channel (as in conventional CFB mode).

SCFB mode is a hybrid of conventional CFB and OFB modes of operation. Operation of the mode is illustrated in Figure 1, where E represents the block cipher encryption operation with block size B and both the encryption and decryption of SCFB mode are illustrated. In SCFB mode, the block cipher is configured in OFB mode until a special *sync pattern* is recognized in the ciphertext, at which point the cipher will collect the next block of bits for use as a new initialization vector (IV). This has the advantage of self-synchronization and yet, since the cipher is run in OFB most of the time, the scheme has close to the efficiency of OFB and is devoid of the inefficiencies of CFB mode.

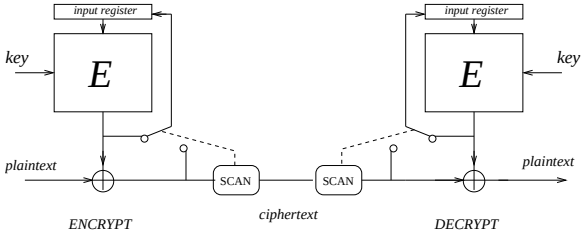

Fig. 1. SCFB Mode

The characteristics of the scheme are greatly influenced by the size of the sync pattern, represented in bits by n. In [3], it is demonstrated that, although it can take a long time to recover from a loss of synchronization for large values of n, for modest values of n the scheme can be quite efficient, while having only modest resynchronization delays. For example, for block size $B = 64$ and $n = 8$, the theoretical efficiency for SCFB mode is 91.1%, while for conventional CFB mode, the efficiency is only 1/64!

In this work, we investigate the relationship of cipher speed (as measured by the efficiency of SCFB mode with respect to straight block encryption) and the delay caused by the SCFB mode of operation. The efficient operation of SCFB mode is conditioned on the appropriate buffering of data to ensure that, should many resynchronizations occur closely in time, there is enough buffer available to avoid buffer overflow while storing

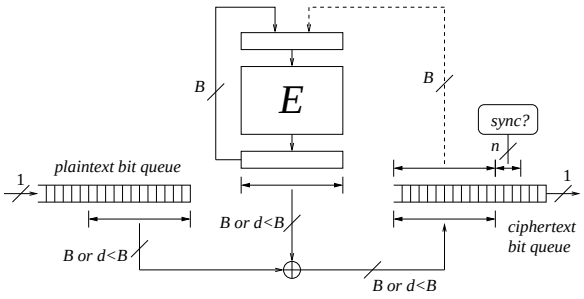

Fig. 2. SCFB Encryption System

plaintext data and there is enough ciphertext buffered to avoid underrun when production of ciphertext gets delayed due to the many required block encryptions in a short time span.

II. Implementation of SCFB System

In our study, we shall examine the buffering requirements of an implementation of SCFB mode. As we shall see, the buffering requirements are directly equivalent to the delay of data as it is transferred through the SCFB encryption system.

For the purposes of our work, we consider the SCFB mode to be implemented as shown for encryption in Figure 2. (A similar system can be envisioned for decryption.) In the system, there are two queues: one for incoming plaintext bits and one for outgoing ciphertext bits. Both queues are assumed to have buffer space of size M bits. When the system is operating in OFB mode (i.e., no sync pattern is detected), bits will collect in the plaintext queue, until an encrypted block is available and the number of bits in the queue is equal to the block size B. At this point, the B plaintext bits are XORed with the output of the block cipher and all B bits are placed into the ciphertext bit queue.

While bits are being collected into the plaintext queue, bits are being removed from the head of the ciphertext queue at exactly the same rate. The ciphertext queue is initialized with arbitrary data so that it is in the full state when the plaintext queue is empty. As the plaintext queue fills up, the ciphertext queue empties and if the plaintext queue gets full, the ciphertext queue will be empty. Consequently, the condition of an overflow in the plaintext queue and an underrun in the ciphertext queue are exactly the same and we need only consider the plaintext queue overflow condition.

Let d represent the number of bits in the plaintext bit queue. Under conditions of no resynchronizations, d will increment from 0 to B bits and then drop back down to 0, repetitively. The number of bits in the ciphertext queue is given by $M - d$. Hence, the delay through the system experienced by any bit is given by $d + (M - d) = M$ bit times.

In order to minimize delay, it is clearly desirable to keep the buffer size M as small as possible. However, M must be greater than B and must large enough to deal with the build up of data that can occur in the plaintext queue when resynchronizations occur. When the sync pattern is detected in the bits being placed into the ciphertext queue, the system continues encrypting until all B bits of IV are collected from the ciphertext. The last bit of IV is unlikely to occur on the boundary of a block cipher encryption and is, in fact, equally likely to occur anywhere within an encryption block. However, since all bits following the last bit of IV must be encrypted using the new output of the block cipher as a result of the block cipher encryption of IV, only part of the block (i.e, up to the last IV bit) needs to be XORed and there is a delay in XORing the following bits until the encryption of IV is complete. During this delay, incoming plaintext bits are buffered into the plaintext bit queue. The XOR of the partial block containing the last IV bit can be executed as soon as the last IV bit is available in the plaintext queue.

If many resynchronizations occur closely together, the plaintext queue will get full and may overflow, resulting in lost data bits. (This will manifest itself as underruns in the ciphertext queue.) In order to keep resynchronizations from continually growing the queue, the plaintext bits are removed from the queue at a rate greater than the rate of bits entering the queue.

The size of the queues must be large enough to ensure that the probability of an overflow is sufficently small so as not to create bit errors. Hence, there is a relationship between the rate at which the block cipher is run (and the corresponding rate at which bits are removed from the plaintext queue) and the size required for the buffer. This is investigated in Section IV.

There are other structures that may be envisioned to suitably implement an SCFB system. For example, incoming plaintext bits can be processed and moved immediately to the ciphertext queue by double buffering the output of the block cipher and encrypting the next OFB block while processing incoming bits of the current OFB block. This effectively moves all delay to the ciphertext queue. The analysis of overflow characteristics will be modified, but the general results and conclusions will be similar to the system considered in this paper.

III. Efficiency Concepts

In [3], the concept of "efficiency" is introduced as an indication of the rate at which the block cipher must encrypt in order to be able to process the incoming plaintext bits quickly enough to avoid queues growing

without bounds or buffers overflowing. In [3], efficiency refers to the minimum number of encryptions required to process the plaintext stream of bits (as would be the case for straight block encryption) divided by the expected number of block cipher encryptions required to generate the keystream bits used in the XOR for SCFB mode. This is actually representative of the theoretical maximum efficiency of the system and indicates that the system cannot be run any more efficiently.

In this paper, when we refer to efficiency, we shall generally be referring to the *full-queue efficiency*, which represents the efficiency at which the system operates while there is data in the plaintext queue to process. That is, if the queue has greater than B bits in it, bits are removed from the queue in blocks of B bits at a rate of $(1/\alpha) \times R/B$ blocks per unit time where $\alpha < 1$ is the cipher *full-queue efficiency* and R is the rate at which bits enter the plaintext queue in bits per unit time. Since data is arriving at the plaintext queue at a rate of R/B for each block of B bits, the full-queue efficiency represents the incoming plaintext data rate divided by the rate at which bits are removed from the plaintext queue. For example, for a queue with some data in it, an efficiency of $\alpha = 50\%$ implies that for every B bits entering the queue, $2B$ bits are removed from the queue, XORed with the output of the block cipher, and placed in the ciphertext queue.

In practice, there will be periods where the plaintext queue has fewer than B bits available at the completion of a block encryption and the system must wait until the queue has B bits. Hence, the *average* efficiency is greater than the *full-queue* efficiency, since during the full-queue situation the block cipher is encrypting at its peak rate and the corresponding efficiency is at a minimum. In either case, assuming that we have a stable system (i.e., the plaintext queue size does not grow without bound or a finite buffer does not overflow), the efficiency must be less than the theoretical maximum efficiency given in [3].

Assuming the time to execute a block encryption is fixed, a higher efficiency implies that the system is capable of operating at higher rates , i.e., larger values of R. The full-queue efficiency, α, gives a direct indication of the operating rate of the system since $R = \alpha \cdot r_{enc} \cdot B$ where r_{enc} represents the encryption rate of the block cipher while there is data to process in the plaintext queue in blocks per unit time. The parameter r_{enc} is generally constrained by the technology used to implement the block cipher.

IV. Buffer Requirements of SCFB Mode

In our work, we investigated the buffer requirements M (or equivalently the delay measured in bit times) of the SCFB encryption system by executing a number of simulations for systems with different buffer sizes and full-queue efficiencies, α. As the output of the simulations, we examined the probability of an overflow of the plaintext queue (or equivalently an underrun of the ciphertext queue). We observed that, although approaching speeds very close to the theoretical maximum efficiency, η, defined in [3], requires much buffer space and a resulting large delay, using speeds reasonably close (eg. at about 80-90% of maximum efficiency) requires modest buffering and causes small delays. As an example, consider Figure 3 and Figure 4, both of which are simulation results based on 64-bit DES with an 8-bit synchronization pattern of "10000000".

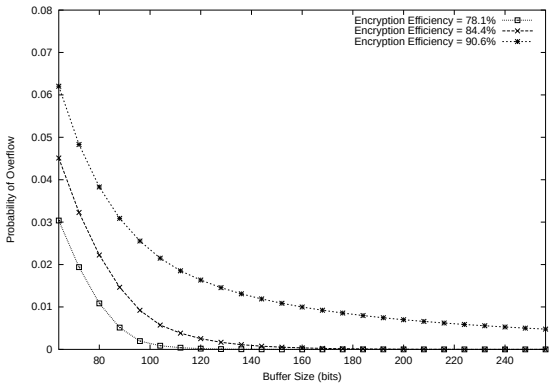

Fig. 3. Probability of Overflow vs. Buffer Size

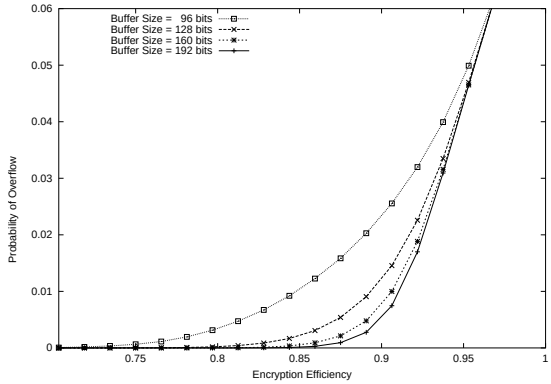

Fig. 4. Probability of Overflow vs. Encryption Efficiency

Figure 3 illustrates the effect on the probability of buffer overflow as a function of the buffer size for fixed full-queue efficiencies. Although, in theory, the efficiency can be as high as 91.1% [3], it is clear that running the system at close to that efficiency causes significant buffer overflows for buffer sizes up to 256 bits. However, for full-queue efficiencies of 78.1% and 84.4%, buffer overflow is very small for buffers of about 120 and 160 bits, respectively.

Figure 4 shows that the probability of overflow increases dramatically as the full-queue efficiency approaches the theoretical maximum efficiency of 91.1%. However, for large buffer sizes of 160 and 192, it can be seen that full-queue efficiencies of 80% to 85% can be achieved before significant overflow occurs. Even for small buffer sizes, such as 96, full-queue efficiencies can be 70% with little buffer overflow.

Of particular note is the efficiency value of 50%. For any sync pattern size, running the system at rates corresponding to 50% (or less) efficiencies guarantees that there is no buffer overflow for a buffer size of B. This occurs because the block cipher is being run at a rate that is two times (or more) the rate of incoming bits and, since all synchronization cycles consist of the IV collection phase followed by OFB encryption based on the IV, the worst case scenario requires no more than two block cipher encryptions for any $B + l$, $n \leq l \leq B$, bits. This is a significant point because it implies that running SCFB at 50% full-queue efficiency guarantees no overflow and a delay of exactly B bit times. This may be compared to conventional CFB, which, while being capable of providing delays of only 1 bit time, results in an efficiency of only $1/B$.

V. The Effect of Block Size on Efficiency

In [3], it is shown that the maximum implementation efficiency of the system is given by

$$\eta = \frac{\sum_{k=0}^{\infty} P(k) \cdot (k + n + B)/B}{\sum_{k=0}^{\infty} P(k) \cdot \lceil (k + n + B)/B \rceil}$$

where k represents the number of bits between the end of an IV block and the beginning of the next sync pattern. It is assumed that k may be approximated as a random variable following the geometric distribution. Hence, the numerator can be computed as $(n + B + E\{k\})/B$ where $E\{k\} = 2^n - 1$. We have computed the maximum efficiency η for block sizes of 64, 128, and 256 bits and these are plotted in Figure 5 as a function of the sync pattern size. It is clear that SCFB mode applied to block ciphers with large block sizes suffers in efficiency. For example, for $n = 8$, the maximum efficiencies are 91.1%, 85.3%, and 78.1% for 64, 128, and 256 bit blocks, respectively. However, these efficiencies are still many times more than conventional CFB mode and, in fact, the ratio of SCFB efficiency to conventional CFB efficiency increases as B increases.

VI. SCFB for 128-bit Block Ciphers

In [3], the characteristics of SCFB are only considered for block sizes of $B = 64$ and specifically the cipher DES. However, with the recent selection of the new AES cipher Rijndael [6] with a block size of $B = 128$, the suitability of SCFB as applied to 128-bit blocks is of interest.

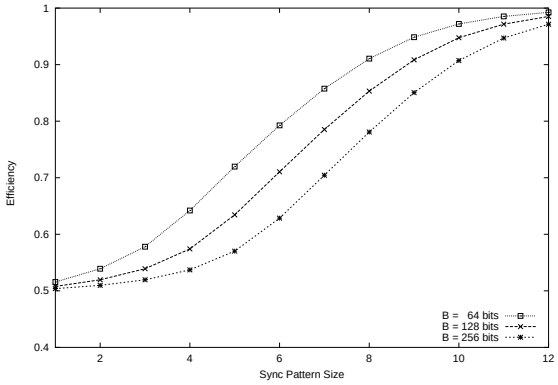

Fig. 5. Maximum Efficiency vs. Sync Pattern Size

The results of our simulation studies confirm the general conclusions of [3], based on $B = 64$, also apply to Rijndael with $B = 128$. It may be seen from Figure 6 which is based on simulations using Rijndael, that the synchronization recovery delay (SRD) of the SCFB mode is at a minimum of about 225 for a sync pattern size of $n = 5$. (The SRD is the time, in bits, required to recover from a loss of synchronization caused by a bit slip.) Also plotted on the graph is the lower bound on SRD [3], which is approximated by 2^n for large n. Note that the logarithm base 2 of SRD is plotted.

In Figure 7, the error propagation factor (EPF), defined as the expected number of bit errors at the decryption output caused by an isolated bit error in the communications channel, is presented as a function of n. The lower bound of $(n + B)/2$ is also plotted and, as expected, for large n, the EPF approaches the lower bound. The EPF characteristics of SCFB are similar to the EPF characteristics of conventional CFB mode, where the EPF is $B/2 + 1 = 65$ for Rijndael.

The results of Figure 6, in combination with a consideration of the efficiencies of the cipher implementation for different values of n, can be used to select an implementation which has a suitable compromise between synchronization recovery (modest values of n) and efficiency (high values of n).

The results of simulations exploring the relationship between buffer size and efficiency for Rijndael are given in Figure 8.

VII. Security Issues

Since IV is essentially a random block of B bits, one concern for the security of SCFB mode is that IV blocks might repeat, resulting in a large sequence of bits that is encrypted with the same keystream sequence as a previous sequence of bits. As a result, knowledge of

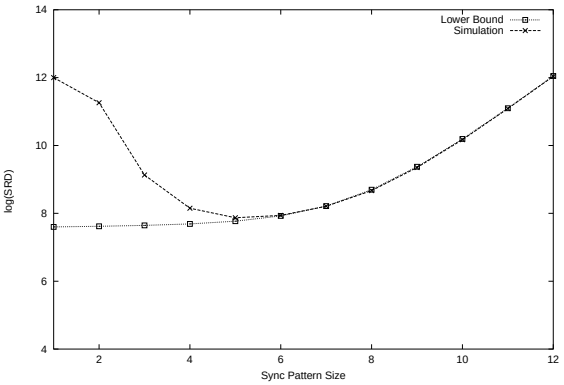

Fig. 6. Sync Recovery Delay vs. Sync Pattern Size ($B = 128$)

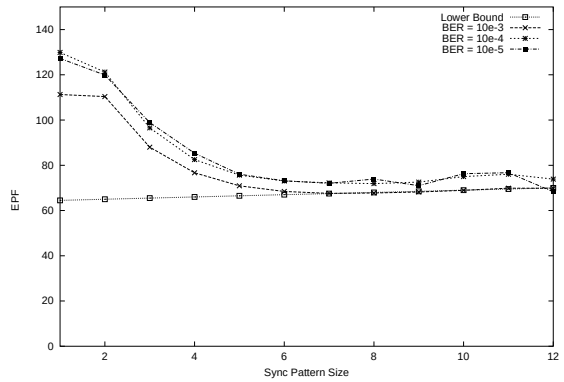

Fig. 7. Error Prop Factor vs. Sync Pattern Size ($B = 128$)

some plaintext might compromise the security of other encrypted unknown plaintext bits. This is particularly of concern for large n, say $n \geq 8$, where the number of bits encrypted under the influence of any one IV value is likely to be large and, hence, a repetition of an IV value could lead to the compromise of many bits. For example, in standard OFB mode, where resynchronizations are executed through the use of a signaling channel, it is generally good practice to avoid re-using IVs.

This security concern is eliminated through the use of large block sizes. By considering the "birthday paradox" [2], it can be rationalized that for $B = 64$ and $n = 8$, where the expected number of bits between consecutive resynchronizations is about $n + B + 2^n - 1 = 327$, about 1 to 2 terabits are required to be stored before any two IV values used in SCFB encryption can be expected to be the same with high probability (i.e., about 50%). For a block size of $B = 128$ and $n = 8$, there are 391 bits, on average, between resynchronizations and being able to store about 1 terabit would only result in data from two identical IVs with a probability of less than 10^{-20}. It is reasonable to conclude, therefore, that repetition of an IV value is not a problem, in a theoretical sense, for $B = 128$ and, in a practical sense, for $B = 64$.

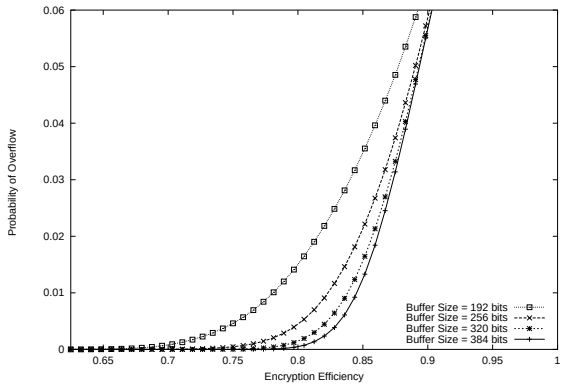

Fig. 8. Probability of Overflow vs. Efficiency ($B = 128$)

VIII. Conclusions

In this paper, we have considered the relationship of the efficiency and delay for an encryption system constructed for SCFB mode. We find that it is feasible to run the system at efficiences approaching the theoretical efficiencies of [3] and still maintain modest buffer sizes and resulting delays of about 2 or 3 blocks. Notably, if the SCFB system is run at 50% or less efficiency, the required buffer size and delay is exactly one block of B bits.

Further, the paper has considered SCFB mode as applied to the new 128-bit AES block cipher. As expected, the same general conclusions apply as for 64-bit block ciphers [3]. Although SCFB mode with 128-bit block ciphers suffers marginally in efficiency, synchronization recovery delay, and error propagation, these factors are small and SCFB is still a very desirable mode of encryption, applicable to communication channels susceptible to bit slips.

References

[1] O. Jung and C. Ruland, "Encryption with statistical self-synchronization in synchronous broadband networks", *Cryptographic Hardware and Embedded Systems - CHES '99*, Lecture Notes in Computer Science 1717, Springer-Verlag, pp. 340-352, 1999.

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

[3] H.M. Heys, "An Analysis of the Statistical Self-Synchronization of Stream Ciphers", *Proceedings of INFOCOM 2001*, Anchorage, Alaska, pp. 897-904, Apr. 2001.

[4] National Bureau of Standards, "Data Encryption Standard (DES)", *Federal Information Processing Standard 46*, 1977.

[5] A. Alkassar, A. Geraldy, B. Pfitzmann, A-R. Sadeghi, "Optimized Self-Synchronizing Mode of Operation", presented at Fast Software Encryption Workshop (FSE 2001), Yokohama, Japan, Apr. 2001.

[6] National Institute of Standards and Technology, AES web site: csrc.nist.gov/encryption/aes.