An Analysis of the Statistical Self-Synchronization of Stream Ciphers

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 examine a recently proposed mode of operation for block ciphers which we refer to as statistical cipher feedback (SCFB) mode. SCFB mode configures the block cipher as a keystream generator for use in a stream cipher such that it has the property of statistical self-synchronization, thereby allowing the stream cipher to recover from slips in the communications channel. Statistical self-synchronization involves feeding back ciphertext to the input of the keystream generator similar to the conventional cipher feedback (CFB) mode of block ciphers, except that the feedback only occurs when a special pattern is recognized in the ciphertext. In the paper, we examine the efficiency, resynchronization, and error propagation characteristics of SCFB and compare these to the conventional modes of CFB, output feedback (OFB), and counter mode. In particular, we study these characteristics of SCFB as a function of the synchronization pattern size. We conclude that, although it can take significantly longer to resynchronize, SCFB mode can be used to provide self-synchronizing implementations for stream ciphers that are much more efficient than conventional CFB mode and that have error propagation characteristics similar to CFB mode.**

Keywords—**cryptography, stream ciphers, block cipher modes of operation**

I. Introduction

In this paper, we discuss a structure for self-synchronizing stream ciphers, recently proposed in [1]. Stream ciphers are used to encrypt one symbol, typically one bit, at a time. They are often used in communication systems when high implementation efficiency (typically in hardware) is required. In particular, they are used when error propagation must be minimized or when the communication channel suffers from periodic bit slips or insertions.

The basic form of a stream cipher that is to operate at the bit level involves the generation of a keystream - a keyed, pseudo-random, unpredictable sequence of bits - that is XORed bit by bit with the plaintext to generate the ciphertext at the transmitter. At the receiver, the plaintext is recovered by generating the identical keystream such that it is exactly synchronized with the received ciphertext stream. Hence, the XOR of the keystream bits and received ciphertext bits produces the original plaintext bits.

In this paper, we shall concern ourselves with stream ciphers which are derived from block ciphers such as DES [2] and the new AES cipher [3]. We focus on an unconventional mode which we shall refer to as *statistical cipher feedback (SCFB)* mode, a form of self-synchronizing stream cipher. The characteristics of SCFB mode are examined and its merits are quantified.

II. Background

There are several conventional block cipher modes of operation that may be applied to derive a stream cipher, each with its own advantages and disadvantages [4]. We briefly review them here. In our notation, we let B represent the block size in bits of the block cipher. For example, for DES, $B = 64$, and for the new AES cipher, $B = 128$.

Output feedback (OFB) mode generates the keystream by feeding back the output of the block cipher to the input, as illustrated in Figure 1. It can be shown that, for security purposes, all B output bits of the block cipher output should be fedback, in order to ensure a long period before the output of the block cipher, and hence the keystream, begin to repeat. The period is expected to be 2^{B-1} in this case.

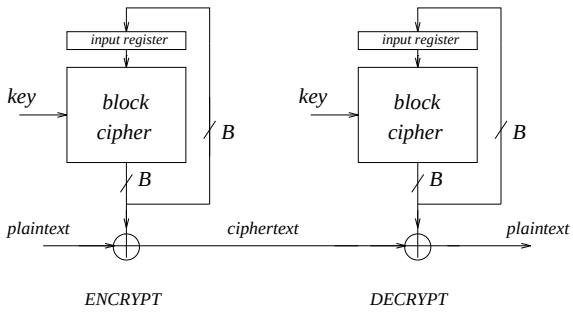

Fig. 1. Output Feedback Mode

The primary advantage of output feedback mode is that error propagation is minimized. In fact, a single bit error in the ciphertext in the communications channel results in only a single bit error in the recovered plaintext. Compare this to the scenario of using the block cipher for straightforward block encryption (referred to as *Electronic Codebook (ECB)* [4]). For ECB mode the plaintext is applied in blocks

of B bits directly at the input of the block cipher and the output of B bits is the ciphertext: a single bit error corrupts an entire recovered plaintext block resulting in $B/2$ bit errors on average.

The most significant disadvantage of OFB is that the system relies on the maintaining of synchronization between the transmitter and receiver. For example, if a single bit slip occurs (i.e., a bit is eliminated from the received ciphertext stream), synchronization loss will occur between the transmitter and receiver and half the bits are expected to be in error until synchronization is recovered. Resynchronization can be achieved by periodically sending an *initialization vector (IV)* from the transmitter to the receiver through the signaling channel of the communications system. Obviously, the price of such a scheme involves extra messaging overhead and the associated delays while synchronizing. As well, the rate at which synchronization messages are sent must balance the overhead of sending such messages frequently with the penalty of losing synchronization caused by slips for a long period of time should the messages be sent too infrequently. However, OFB (not considering the resynchronization messaging overhead) can be implemented as efficiently as ECB block encryption by using all B bits generated by the block cipher to XOR with B bits of plaintext, i.e., B bits of keystream are produced from one block cipher output. This is the configuration illustrated in Figure 1.

Counter mode is another configuration for a stream cipher, generated with a block cipher core, that is similar to OFB and is illustrated in Figure 2. This mode of operation, for example, is proposed as part of the ATM security specification [5]. For counter mode, inputs to the block cipher are generated by an independent counter which can be configured to give the maximum period of 2^B for the keystream. Counter mode has the same error propagation and resynchronization characteristics as OFB and can also be implemented efficiently with B bits of keystream generated from every block cipher output.

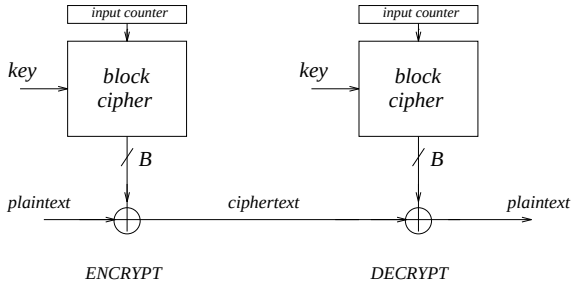

Fig. 2. Counter Mode

Cipher feedback (CFB) mode is another mode of a block cipher for use as a stream cipher and is shown in Figure 3. The properties of this mode are significantly different than OFB and counter modes. Most significantly, CFB mode allows for automatic resynchonization should bit slips or bit insertions occur in the communications channel and, hence, CFB stream ciphers fall into the category of self-synchronizing stream ciphers. In CFB, the input to the block cipher is driven by ciphertext data which is fedback into a shift register at the input of the block cipher in groups of m ($\leq B$) bits at a time. The plaintext is encrypted by XORing m bits at a time with m bits of the block cipher output. For decryption, similarly, ciphertext is fed into the register to provide the input to the block cipher, as shown in Figure 3.

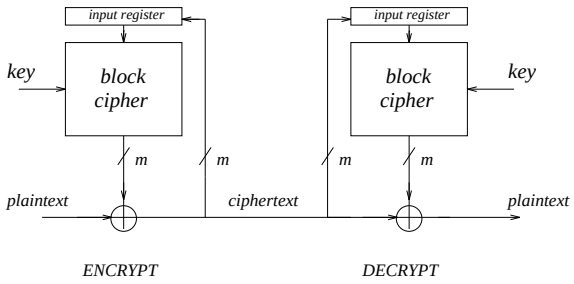

Fig. 3. Cipher Feedback Mode

Since the input to the block cipher is being generated from the ciphertext which is available to both ends of the communication, it is possible to recover from slips or insertions. Consider the case with $m = 1$. In this case, each bit encryption (or decryption) requires a complete encryption from the block cipher. This is very inefficient when compared to OFB or counter mode, where it is straightforward to implement the XOR operation with B bits in parallel. However, if any single or multiple bit slips/insertions occur, the CFB cipher with $m = 1$ can recover synchronization since the next B bits will be shifted into the register at the input to the block cipher and at this point the receiver will again be synchronized with the transmitter. Resynchronization therefore requires only B bits in CFB mode with $m = 1$.

Unfortunately, the self-synchronization property achieved by CFB mode is costly in terms of implementation efficiency. Essentially, CFB with $m = 1$ is capable of operating at only $1/B$ times the rate of ECB block encryption and, consequently, can only be implemented at $1/B$ times the rate of OFB and counter modes. This can be improved by increasing m but, if $m > 1$ and a single bit slip or insertion occurs, the input to the block cipher at the receiver will become misaligned and resynchronization will not occur. In fact, slips/insertions must occur as multiples of m bits or resynchronization will not occur. Hence, usually for CFB mode, $m = 1$ is the desirable configuration[1].

[1]There are cases where $m > 1$ make sense. For example, CFB with $m = 8$ can be used to encrypt an asynchronous communications link so that an 8-bit character can be encrypted with each block cipher output. Here CFB mode is used to recover from losses of synchronization due to asynchronous characters being lost, as opposed to individual bit slips or insertions.

Finally, it should be noted that the error propagation advantages of OFB and counter modes are no longer applicable to CFB mode. This occurs because a bit error must work its way through the shift register at the input to the block cipher. As a result, for CFB mode with $m = 1$, a single bit error will result in the corrupted bit plus the next B bits being randomly decrypted due to the corrupted keystream. So a bit error is expected to result, on average, in $B/2+1$ errors in the recovered plaintext.

In this paper, we analyze the properties of the hybrid scheme proposed in [1][2], which we shall refer to as *statistical cipher feedback (SCFB)* mode. This scheme has the desirable property that it is capable of self-synchronization for any number of bit slips. Specifically, we consider the implementation efficiency, the synchronization recovery delay, and the error propagation characteristics of SCFB.

III. Statistical Self-Synchronization

In [1], the concept of statistical self-synchronization is proposed as a mechanism to provide physical layer security for an SDH/SONET environment. Essentially, the concept of statistical self-synchronization involves a hybrid of OFB (or counter) mode and CFB mode: the cipher operates in OFB mode (or counter mode), while scanning the ciphertext for a special *sync pattern* of n bits in length. When this pattern is recognized, the next B bits are stored for a new initialization vector (IV) and, after all B bits have been collected, the input register for the block cipher is loaded with the new IV. The cipher then proceeds in OFB mode (or counter mode) until the next n bit sync pattern is received. During the collection of B bits for the new IV, the sync pattern scanning is turned off so that any n bits matching the sync pattern are ignored until the IV collection phase is complete. This process follows for both encryption and decryption and, since both the transmitter and receiver are examining the ciphertext, synchronization is achieved.

To provide enough detail for precise clarity of the operation of SCFB mode, a pseudocode representation for encryption at the transmitter using SCFB with OFB as the base mode is given in Figure 4. The sync pattern is given by $Q_0 \dots Q_{n-1}$ and $W_0 \dots W_{n-1}$ represents the window of n bits that is currently being compared to the sync pattern. In order for the algorithm, as presented, to work with the initialization of $W_0 \dots W_{n-1}$ to all zeroes, Q_0 must be 1. The function $ENC_K(\cdot)$ represents the block cipher encryption (using key K). $Z_0 \dots Z_{B-1}$ is used to collect the IV bits. The flags $loading\_IV$ and new_IV are used to indicate that IV is currently being collected (and sync pattern scanning is therefore suspended) and collection of IV has just completed, respectively. Note that the initial block cipher input $X_0 \dots X_{B-1}$ is given an initial value known to both the transmitter and receiver at the beginning of the communication. SCFB based on counter mode would be very similar except that the new cipher input $X_0 \dots X_{B-1}$ would be derived by incrementing the previous input value, rather than from the previous output of the block cipher.

[2] This scheme appears to have also been invented earlier and is referred to in [6].

```
loading_IV ← false
X_0 ... X_{B-1} ← initial value
W_0 ... W_{n-1} ← 0 ... 0
j ← 0
do
    Y_0 ... Y_{B-1} ← ENC_K(X_0 ... X_{B-1})
    new_IV ← false
    i ← 0
    do
        C_{j+i} ← P_{j+i} ⊕ Y_i
        if loading_IV then
            Z_k ← C_{l+k}
            k ← k + 1
            if k = B then
                loading_IV ← false
                new_IV ← true
                X_0 ... X_{B-1} ← Z_0 ... Z_{B-1}
                W_0 ... W_{n-1} ← 0 ... 0
        else
            W_0 ... W_{n-2} W_{n-1} ← W_1 ... W_{n-1} C_{j+i}
            if W_0 ... W_{n-1} = Q_0 ... Q_{n-1} then
                loading_IV ← true
                l ← j + i + 1
                k ← 0
        i ← i + 1
        if i = B and not new_IV then
            X_0 ... X_{B-1} ← Y_0 ... Y_{B-1}
    while i < B and not new_IV
    j ← j + i
while true
```

Fig. 4. Statistical Cipher Feedback Mode

From the pseudocode, it may be seen that the encryption of a bit would encounter a significant delay whenever a new block encryption is required since ENC_K can be expected to take much longer than any other operations in the algorithm. Hence, in practice, for a synchronous system an implementation would need a buffer in order to ensure that plaintext bits can be accepted at a uniform rate and ciphertext bits can be produced at a uniform rate at the output of the encryption process.

Since both the transmitter and receiver are using recognized bits within the ciphertext as a cue to resynchronize the stream cipher, SCFB mode is capable of self-synchronization and will clearly perform better in an environment where slips and insertions occur than either OFB or counter modes. Also, although individual bit errors will only cause one bit error if the bit is not part of the sync pattern or the initialization vector, there is the possibility that a bit error will cause a synchronization to be missed or a false synchronization to be detected at the receiver. In these cases, a single bit error in the communications channel will result in many bit errors at the output of the decryption as synchronization

will be lost until the next sync pattern is properly detected. Hence, clearly the error propagation characteristics of SCFB will be worse than OFB and counter mode.

Although SCFB mode was proposed in [1], the characterizations of efficiency, resynchronization, and error propagation were not fully developed. Specifically, the relationship of the sync pattern size to these properties was not examined. In Sections V and VI, we shall consider the resynchronization and error propagation characteristics of SCFB and, in particular, make a comparison to conventional CFB mode. In the next section, we outline the real value of SCFB over CFB mode by examining the implementation efficiency of SCFB mode.

IV. Implementation Efficiency

The principal advantage of implementing SCFB versus conventional CFB is that the efficiency (and hence the potential speed) of the implementation can approach that of OFB and counter mode, depending on the value of n. Letting M represent the number of bits transmitted, we can define efficiency for a stream cipher based on a block cipher core as:

$$\eta = \lim_{M\to\infty} \frac{M/B}{E\{\#\text{ block encrypts for } M \text{ bits}\}} \tag{1}$$

where $E\{\cdot\}$ represents the expectation operator. Hence, efficiency is essentially a measure of the rate at which the stream cipher can encrypt in comparison to ECB block encryption.

Hence, for OFB and counter modes when all B bits are used in the XOR operation, $\eta = 1$, and for conventional CFB with $m = B$, $\eta = 1$. However, if we are to be guaranteed to correct all synchronization losses, conventional CFB must operate with $m = 1$, and in this case, $\eta = 1/B \ll 1$. So conventional CFB is very inefficient in comparison to ECB block encryption.

Consider now SCFB. We can assume that for SCFB, the bits transmitted in the communications channel can be viewed as in Figure 5, where it is clear that some bits belong to the sync pattern (n bits), some belong to the subsequent IV (B bits), and the remaining bits, which we shall refer to as the *OFB block*, occur between the end of the IV and the beginning of the next sync pattern. We shall refer to the block of bits from the beginning of the sync pattern to the beginning of the next sync pattern as a *synchronization cycle* and, hence, a synchronization cycle consists of $n+B+k$ bits, where k is the size of the OFB block.

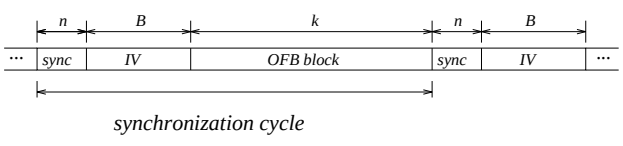

Fig. 5. Synchronization Cycle

The size k of the OFB block is variable and dependent on the position of the next sync pattern. Since we assume that the block cipher used in the SCFB configuration displays strong randomness properties (or else it would be insecure), k is a random variable with a probability distribution determined by assuming that each bit of ciphertext is equally likely to be a 0 or 1 and that each bit is independent. Strictly, the distribution of k is dependent on what sync pattern is used (eg. $11\ldots11$, or $10\ldots00$, etc.). However, we have verified experimentally that, for most sync patterns, the probability distribution of k can be approximated by the geometric distribution. We therefore use this distribution in our development and defer a more detailed discussion of the effect of the selection of the sync pattern to the Appendix[3].

Based on the geometric distribution, we have the probability distribution for k given by

$$P(k) = (1 - 1/2^n)^k \cdot 1/2^n \tag{2}$$

and the expected value of k given by

$$E\{k\} = 2^n - 1. \tag{3}$$

Consider now the scenario for Figure 5 where $k + n + B$ is not a multiple of B. In this case, after the second IV is collected, the input register will be loaded and a new block encryption will occur after a block encryption that is used for encrypting only a subset of a full B-bit block of plaintext. For example, let $k + n + B = \alpha B + \gamma$ where α and γ are integers and $\gamma < B$. Hence, the block cipher must execute $\alpha + 1$ block encryptions from the beginning of the OFB block to the end of the second IV, producing $(\alpha + 1)B$ bits, to encrypt only $\alpha B + \gamma$ plaintext bits (i.e., only γ bits of the last block cipher output are used in the XOR). Therefore, the block cipher must be run at a rate slightly greater than for ECB block encryption and a buffer must be used to accommodate scenarios where only partial outputs of the block cipher are used by the XOR operation.

Using (1) as the basic definition for efficiency, it is possible to define the efficiency of SCFB as

$$\eta = \frac{E\{\text{cycle size}\}/B}{E\{\#\text{ block encrypts per cycle}\}}. \tag{4}$$

This leads to

$$\eta = \frac{\sum_{k=0}^{\infty} P(k)\cdot(k+n+B)/B}{\sum_{k=0}^{\infty} P(k)\cdot\lceil(k+n+B)/B\rceil} \tag{5}$$

where the numerator can also be straightforwardly computed as $(n + B + E\{k\})/B$.

Now consider the efficiency of SCFB schemes characterized by the sync pattern size n. The results for $B = 64$

[3]It is clear that the geometric distribution must be an approximation since it is based on the assumption of independence between samples of n bits (which are compared to the sync pattern). However, since the n-bit samples overlap, there are clearly dependencies between samples.

n	η	$E\{k\}$	n	η	$E\{k\}$
1	.516	1	7	.857	127
2	.539	3	8	.911	255
3	.578	7	9	.948	511
4	.642	15	10	.972	1023
5	.720	31	11	.985	2047
6	.792	63	12	.993	4095

TABLE I
EFFICIENCY FOR DIFFERENT SYNC PATTERN SIZES

and various values of n are given in Table I (along with the expected length of the OFB block). It is obvious from the table that, as n increases, the efficiency of the implementation increases and, for large n, the efficiency approaches 100%, implying that the stream cipher can be run at a rate very nearly equivalent to the speed of ECB block encryption. In fact, a cipher can be run at a rate approaching $1/\eta$ times ECB block encryption with a suitably large enough buffer to account for the scenario of several block encryptions required for partial blocks (i.e., the XOR operation involving less than B bits) within a short time frame.

For small values of n, SCFB is significantly less efficient than ECB block encryption. For example, for $n = 1$, the cipher is resynchronizing after an expected OFB block size of 1 and as a result virtually every second block cipher output is used only for a small number of bits in the XOR operation. Hence, the efficiency is only about 50%.

The case of $n = 0$ is equivalent to the conventional CFB case with $m = B$ and this case achieves maximum efficiency of 100% but cannot properly resynchronize for single bit slips (or indeed, for any slip of a non-multiple of B bits!). For conventional CFB mode with $m = 1$ to accommodate sync recovery from any number of bit slips, the efficiency would only be 1.56% for a block size of $B = 64$.

Note that all efficiencies for SCFB mode are greater than 50%. This occurs because at least one full block is used in each synchronization cycle since B bits are associated with IV.

V. RESYNCHRONIZATION

A fundamental requirement of a self-synchronizing stream cipher is that resynchronization occur quickly to minimize the corruption of data due to a sync lost condition. Conventional CFB mode with $m = 1$, for example, will synchronize within B bits of a bit slip. Resynchronization delay for an OFB cipher relying on signaling messages to provide synchronization information will generally be very large since synchronization is usually relying on a low rate signaling channel and synchronization messages are exchanged relatively infrequently.

In this section, we examine the resynchronization properties of SCFB mode. We begin by considering a lower bound on the *synchronization recovery delay (SRD)*, defined as the expected number of bits following a sync loss due to a slip or insertion before synchronization is regained. Assume that a single bit slip occurs randomly within a synchronization cycle of $n + B + k$ bits as illustrated in Figure 5. The probability that a bit slip occurs within a synchronization cycle of size k is given by

$$P^*(k) = \frac{(n + B + k) \cdot P(k)}{n + B + E\{k\}} \tag{6}$$

where $P(k)$ and $E\{k\}$ are given by equations (2) and (3), respectively. Assuming that the receiver resynchronizes at the next sync pattern, i.e., at the end of the next IV, it will take an average of $(n + B + k)/2 + n + B$ bits to resynchronize. This is determined by the average position of a slip within the synchronization cycle plus the $n + B$ bits required at the beginning of the next synchronization cycle to resynchronize. Now if we consider the average over all OFB block sizes k then the synchronization recovery delay is lower bounded as in

$$\begin{aligned} SRD &> \textstyle\sum_{k=0}^{\infty} \frac{3(n+B)+k}{2} \cdot P^*(k) \\ &= \frac{3(n+B)}{2} + \frac{1}{2} \textstyle\sum_{k=0}^{\infty} \frac{(n+B)kP(k)+k^2P(k)}{n+B+E\{k\}} \\ &= \frac{3(n+B)}{2} + \frac{1}{2} \cdot \frac{(n+B)E\{k\}+E\{k^2\}}{n+B+E\{k\}} \end{aligned} \tag{7}$$

where $P(k)$ and $E\{k\}$ are given by (2) and (3), respectively, and the second moment for the geometric distribution is given by $E\{k^2\} = 2^{2n+1} - 3 \cdot 2^n + 1$. For large n, the sync recovery delay lower bound of (7) is approximated by 2^n.

Equation (7) represents a lower bound because it is possible that the position of the slip can result in scenarios which prevent resynchronization at the next sync pattern. For example, a slip could occur in such a way that the new sequence of ciphertext bits result in a false synchronization. If this occurs within B bits of the beginning the next valid sync pattern, the receiver will interpret the valid sync pattern bits as part of IV and will ignore them. As a result, resynchronization will be delayed until the next sync pattern. For small OFB block sizes, this could even happen in a manner such that several proper sync patterns are misinterpreted as part of the initialization vectors of several false synchronizations. This phenomenon is particularly prevalent for small values of n since small OFB block sizes are much more likely.

We have examined experimentally the sync recovery delay versus different values of n by running simulations with a sync pattern of the form $10\ldots00$. It is reasonable to assume that slips are infrequent so that it is very unlikely to have more than one slip in a synchronization cycle, and, in our simulations, we have used a slip rate of 1 bit slip every 10^5 bits. The results of the simulation (using DES with

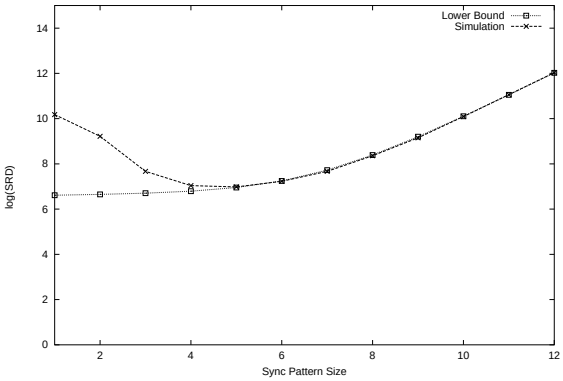

Fig. 6. Sync Recovery Delay vs. Sync Pattern Size

$B = 64$ as the block cipher) are shown in Figure 6, along with the lower bound as previously discussed. Note that the graph represents a plot of the logarithm base-2 of the sync recovery delay for convenience.

We can conclude from the graph that, as expected from the lower bound, as n increases, the synchronization recovery delay increases in an exponential manner. As n gets large, the synchronization recovery delay approaches the lower bound of (7). It should be noted that the lower bound is based on the geometric distribution for k, whereas the simulations are based on a sync pattern for which k follows closely, but not exactly, the geometric distribution. (See the Appendix for discussion of this point.)

For values of $n \leq 4$, the effects mentioned previously cause resynchronization problems and we find that the sync recovery delay is actually minimized for a value of $n = 5$ and has a value of about 126. This may be compared to the synchronization recovery delay of 64 for conventional CFB mode with $m = 1$. For large values of n, the synchronization recovery delay is much greater than for conventional CFB. For example, for $n = 10$, the sync recovery delay is about 2^{10} bits.

VI. Error Propagation

In this section, we consider the effect of SCFB mode on the error characteristics at the output of the decryption. As we shall observe, the error propagation is on the same order as for conventional CFB mode and, hence, is much poorer than OFB and counter modes.

We shall consider the *error propagation factor (EPF)* defined to be the bit error rate at the output of the decryption divided by the probability of a bit error in the communications channel (i.e., in the ciphertext). It is assumed that bit errors occur randomly and independently in the communications channel.

We consider first a simple lower bound on the error propagation factor. Let P_e represent the probability of a bit error in the ciphertext. For any block, the probability of an error occurring in either the sync pattern or the IV is given by

$$\begin{aligned} P(\text{sync/IV has error}) &= 1 - (1 - P_e)^{n+B} \\ &\approx (n + B) \cdot P_e \end{aligned} \tag{8}$$

where the approximation is valid if $n + B \ll 1/P_e$, which would typically be true for reasonable values of n, B and P_e. When an error occurs in the sync/IV portion of a synchronization cycle, it is expected that half of the OFB block and the next sync/IV block bits will be in error. This is irrespective of the size k of the OFB block portion of the synchronization cycle, and so we may conclude that the expected bit error rate at the output of the decryption process at the receiver will be lower bounded as

$$P(\text{error after decrypt}) > \frac{(n + B) \cdot P_e}{2}. \tag{9}$$

Hence, the lower bound on the error propagation factor is given by

$$EPF > \frac{n + B}{2}. \tag{10}$$

We might expect that other conditions for bit errors (i.e., other than in the sync/IV block) will only result in one bit error. However, this is not necessarily the case. Consider below a list of error conditions and the resulting effects:

1. error in $n + B$ bits of sync/IV block
   $\Rightarrow$ sync lost for entire cycle ($\sim \frac{k+n+B}{2}$ expected bit errors)
2. error in OFB block such that no sync pattern is falsely generated
   $\Rightarrow$ one bit error generated in recovered plaintext
3. error in OFB block such that false sync pattern generated in first $k - B$ bits of OFB block
   $\Rightarrow$ $i/2$ bit errors generated, where i is number of bits between end of false IV and end of next IV
4. error in OFB block such that false sync pattern generated in last B bits of OFB block
   $\Rightarrow$ next sync pattern missed because it is interpreted to be part of false IV causing 1/2 bits in error until next valid sync
5. error while sync already lost at receiver
   $\Rightarrow$ generates a possible bit error at the corresponding position of recovered plaintext

Note that cases 3 and 4 involve false detection of the sync pattern. This can occur if a bit error results in a sync pattern in the received ciphertext. The probability that a bit error causes a false synchronization is less than $n/(2^n - 1)$.

For small values of n, cases 3 and 4 become much more likely and the lower bound for the error propagation factor of $(n + B)/2$ is very loose. But as n increases, cases 1 and 2 dominate. Hence, case 1 becomes the significant factor in determining the error propagation factor and as a result the lower bound is a close estimate for the actual error propagation factor.

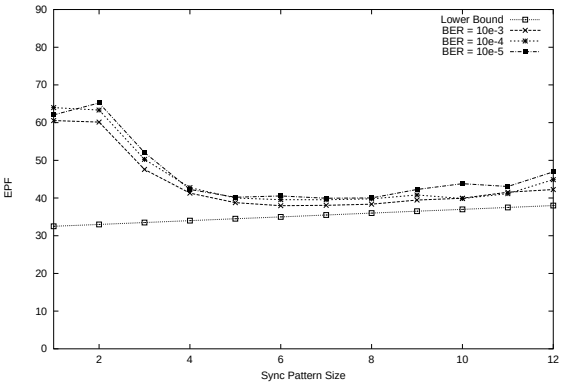

Fig. 7. Error Propagation Factor vs. Sync Pattern Size

We have investigated the error propagation factor experimentally and the results are illustrated in Figure 7, where the error propagation factor is given as a function of n for values of $P_e = 10^{-3}$, $P_e = 10^{-4}$, and $P_e = 10^{-5}$. The lower bound of (10) is also illustrated. Once again, the block cipher used in the simulations is DES with $B = 64$ and the sync pattern used is of the form $10\ldots00$. Clearly, the error propagation factor for most values of n is close to the case of conventional CFB mode with $m = 1$ where the error propagation factor is $B/2 + 1 = 33$ for DES.

VII. Conclusion

In the paper, we have examined the efficiency, synchronization recovery delay, and the error propagation chacteristics of a recently proposed encryption mode that we refer to as statistical cipher feedback (SCFB) mode. Most significantly, we have examined these characteristics as a function of the sync pattern size.

We conclude that, while SCFB mode does not suffer from the complete loss of synchronization in the case of bit slips or insertions as in OFB and counter modes, its recovery time from slips is dependent exponentially on the sync pattern size and can be significantly longer than conventional CFB mode, particularly for large values of n. However, this disadvantage is offset by the advantage that, as the sync pattern size increases, the efficiency of SCFB mode approaches efficient implementations of OFB and counter modes and, hence, SCFB is much more efficient than conventional CFB mode. Finally, it is shown that the error propagation characteristics are similar to CFB mode and are not heavily dependent on the synchronization pattern size. Hence, for a particular application, the actual sync pattern size can be selected by determining a suitable compromise between synchronization recovery and efficiency.

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

[3] National Institute of Standards and Technology, AES web site: www.nist.gov/aes.

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

[5] The ATM Forum, *ATM Security Specification*, Version 1.0, AF-SEC-0100.000, Feb. 1999

[6] U.M. Maurer, "New Approaches to the Design of Self-Synchronizing Stream Ciphers", *Advances in Cryptology - EUROCRYPT '91*, Lecture Notes in Computer Science 547, Springer-Verlag, pp. 458-471, 1991.

Appendix

Discussion on Synchronization Cycle Length

In the paper, we have used the geometric distribution to characterize the probability distribution of the the OFB block size k. However, the sync pattern is scanned for by essentially sliding along a window of n bits and comparing this to the sync pattern. Hence, there will be an overlap of windows and the detection of a sync pattern at a particular position is not independent of the sync pattern occurring at other locations.

In fact, assuming that each bit of ciphertext is equally likely to be a 0 or 1 and that each bit is independent, it is found that the distribution of the random variable k is actually dependent on the sync pattern. We illustrate this by examining, for varying sync pattern sizes n, two cases: sync patterns of the form $10\ldots00$ and of the form $11\ldots11$.

(a) *Sync Pattern* $10\ldots00$:

The probability distribution of k can be shown to be

$$P_a(k) = \begin{cases} [1 - \sum_{i=0}^{k-n} P_a(i)] \cdot \frac{1}{2^n} & , k > 0 \\ \frac{1}{2^n} & , k = 0 \\ 0 & , k < 0. \end{cases} \qquad (11)$$

Equation (11) is explained as follows. First, it should be noted that $P(k)$ is equivalent in meaning to the probability that the next sync pattern ends $k + n$ bits after the last bit of the previous IV. It is trivially true that k must be nonnegative. The case for $k = 0$ occurs when the first n bits following the last bit of IV correspond to the n bits of the sync pattern. The probability for the case of $k > 0$ comes from the fact that, in order to finish the sync pattern at the $(k + n)$-th bit, the $(k + 1)$-th to $(k + n)$-th bits must each have values corresponding to the bits of the sync pattern: the probability of this is $1/2^n$. As well, there must be no sequence of bits equivalent to the sync pattern in the first k bits: the probability of this is given by the complement of the probability that the next sync pattern finishes at bit n, or bit $n + 1$, etc., which is simply the sum of the probabilities up to $P(k - n)$. The resulting probability for the case of $k > 0$ is given by the product of the probability that the pattern does not appear in the first k bits and the probability that it does appear

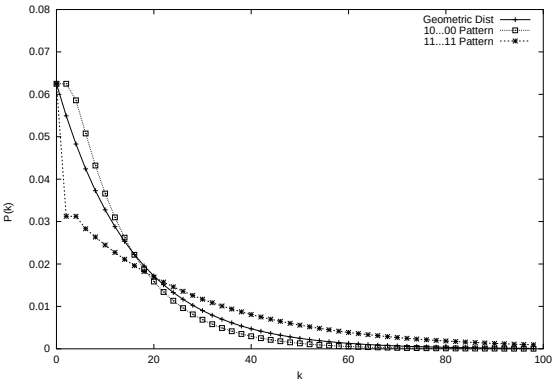

Fig. 8. Probability Distribution Comparison, $n = 4$

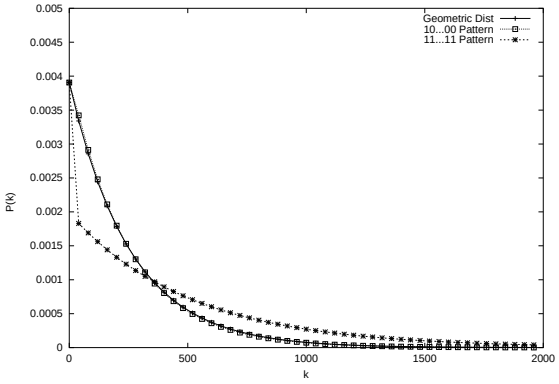

Fig. 9. Probability Distribution Comparison, $n = 8$

in the bits $k + 1$ to $k + n$. Particularly, it should be noted that the pattern cannot appear in bits $k + 1 - i$ to $k + n - i$, $1 \leq i \leq n - 1$, and also in bits $k + 1$ to $k + n$ because of the format of the bit pattern. This is accounted for in the calculation of the probability. (This is quite different than in the calculation of the geometric distribution, where it is assumed that if the pattern finishes at $k+n$, the probability that it does not finish at $k + 1 - i$ to $k + n - i$ must be accounted for.)

We have verified computationally that the expected value of k for the sync pattern $10 \ldots 00$ based on probability distribution P_a is given by

$$E_a\{k\} = 2^n - n \tag{12}$$

which differs from the value for the geometric distribution given by (3). However, this difference is small.

(b) *Sync Pattern* $11 \ldots 11$:

In this case, the probability distribution can be shown to be

$$P_b(k) = \begin{cases} [1 - \sum_{i=0}^{k-n-1} P_b(i)] \cdot \frac{1}{2^{n+1}} & , k > 0 \\ \frac{1}{2^n} & , k = 0 \\ 0 & , k < 0. \end{cases} \tag{13}$$

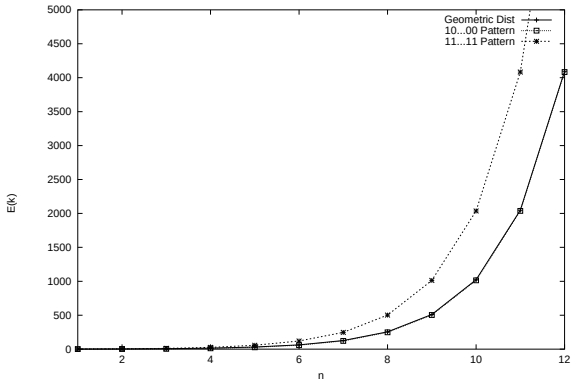

Fig. 10. Expected Value Comparison

The case for $k > 0$ varies from $P_a(k)$ of (11). It is derived from the fact that, in order to finish the sync pattern at the $(k + n)$-th bit, the k-th bit must have a value of 0 and the $(k + 1)$-th to the $(k + n)$-th bits must each have values of 1: the probability of this is $1/2^{n+1}$. As well, there must be no sequence of bits equivalent to the sync pattern in the first $(k-1)$-th bits: the probability of this is given by the complement of the probability that the next sync pattern finishes at bit n, or bit $n + 1$, etc.. The product of these probabilities results in the case for $k > 0$ in (13).

For sync pattern $11 \ldots 11$, the expected value of k is given by

$$E_b\{k\} = 2 \cdot (2^n - 1) - n \tag{14}$$

which differs significantly from the expected value in the geometric distribution.

(c) *Comparison of Distributions*:

For a comparison of the probability distributions between the two cases of sync patterns and the geometric distribution, see Figure 8 (for $n = 4$) and Figure 9 (for $n = 8$). As well, Figure 10 illustrates the expected values for the different distributions as a function of n.

It is clear that the sync pattern $10 \ldots 00$ follows very closely the distribution and the resulting expected values of the geometric distribution. (In fact, the distribution for $n = 8$ and the expected value plots follow virtually exactly the geometric distribution from the perspective of a visual interpretation of Figures 9 and 10.) We have found this to be the case for most sync patterns. However, the sync pattern $11 \ldots 11$ has a distribution significantly different than the geometric distribution and has an expected value for k that is about twice the expected value of the geometric distribution. In our experiments, in Figures 6 and 7, we have used the sync pattern $10 \ldots 00$ and so we expect that the theoretical values developed from the geometric distribution should be a reasonable approximation of the values for the actual sync pattern.