A Survey of Asynchronous Extensions of Block Cipher Modes of Operation

Jason Franklin Yael Peled
Department of Computer Science
University of Wisconsin-Madison
Madison, WI 53706

November 3, 2003

Abstract

While modes of operation and asynchronously clocked devices have been studied separately, the combination of the two ideas has received little attention. We first consider an asynchronously clocked mode of operation based on an extension of OFB mode [2]. After reviewing some of the weaknesses of our proposed mode of operation, we extend on our original design in an attempt to mask some of the identified security weaknesses. In the process of redesigning our initial insecure design, we survey a variety of options resulting from the combination of an asynchronous clock to a mode of operation. While some of our asynchronous devices display a potential for increased complexity which may translate to additional security, often only a slight modification of current cryptanalytic attacks will compromise the combined modes. Only one mode, Dual Stream Multiplexer mode, DSM, apparently increases the complexity of cryptanalysis. DSM mode selects between the key streams of two OFB devices to prevent an attacker from determining the origin of key stream blocks. DSM mode delivers increased complexity with minimal changes required for legacy block ciphers functioning as a stream cipher.

Contents

List of Figures

1 Introduction

Block ciphers are used in conjunction with a mode of operation, a design that allows for the combination of a cipher, feedback, and simple operations. The standard for these cryptographic devices is defined in the Federal Information Processing Standards Publication 81[2]. Implicit in the standard for a mode of operation is the idea of a clock which provides a synchronous pulse to drive the device. It appears to be a natural extension of the current modes to remove the assumption of a synchronous clock. We propose several such asynchronous designs and analyze the resulting key streams as well as describe possible cryptanalytic attacks.

We first considered an initial design that is a derivative of output feedback mode or OFB. This device uses a pseudorandom sequence as the clock to select between the block output from the forward cipher action, as in the traditional OFB mode, or the previous block of key stream. Since the resulting key stream of the device is the primary determinate of security, we analyzed the key stream focusing on randomness testing and periodicity.

We conjectured that an increase in the complexity of the cryptographic device, i.e. the inclusion of an asynchronous clock, would result in a more complex key stream, which would in turn increase the complexity of cryptanalysis. We realize that the complexity of the device and the complexity of cryptanalysis are not directly proportional and attempt to utilize sound cryptographic principles in modifying the modes of operation. Cosmetic changes or fixed permutations are not added as they merely modify the key stream in a constant and clearly insecure way. Changes that increase the complexity of current cryptanalytic attacks such as linear and differential cryptanalysis are our primary focus, as these represent positive security characteristics of a new mode, in addition to the typical characteristics such as efficiency and fault tolerance.

Upon analysis of the key stream of our initial device, we noticed several security flaws which compromise the new mode of operation, the primary flaw being the repetition of blocks of key stream. Clearly our original design is unusable and a mode that hides the repeated blocks of key stream should be constructed. Thus, we redesigned our device to eliminate the perceived security flaws and in turn survey the features of several of our devices.

2 Related Work

Several fields of current and past cryptographic research including modes of operation, pseudorandom bit generators, stream ciphers, and random number testing have contributed to the design and analysis of the asynchronously clocked modes of operation. Work relating to our proposed modes includes the original standard for DES modes of operation [8] containing the OFB specification and a more current recommendation [2]. In addition, current research [6, 7] into new modes of operation for AES has provided insight into the properties and design concepts of a practical mode of operation. We found the descriptions

of the current standard modes in [9] useful for comparison and benchmarking. The analysis of OFB mode in [1] recommended that the entire block of key stream be used as feedback to the shift register rather than a fixed portion of the block size. This recommendation is noted in our implementation of OFB mode. Additionally the idea of OFB mode functioning as a finite state device and the included theoretical analysis of the expected period [1] was useful when comparing the period of our asynchronous mode to the traditional OFB mode. Clock controlled pseudorandom bit generators [10] including the stop-and-go generator and the self-decimated generator inspired several of our asynchronous designs. The related studies of stream ciphers and random number testing were critical as often the key stream of a stream cipher is the primary subject of extensive statistical and random number testing. We implemented several random number testing algorithms inspired by Knuth [5] and used the results of such tests as a benchmark for our modes. Helpful information concerning side channel cryptanalysis, more specifically timing attacks, was provided by Kelsey and others [3].

3 Asynchronous OFB Mode

In our initial attempt to define an asynchronous mode of operation, we propose an extension of OFB mode called asynchronous OFB or AOFB. This mode of operation uses the forward cipher action, $E(x)$, to encrypt an initialization vector, IV, and produce k_1 which is the first block of key stream. Each k_i is then used in its entirety as feedback to a shift register, SR, the contents of which are encrypted to produce k_{i+1}.

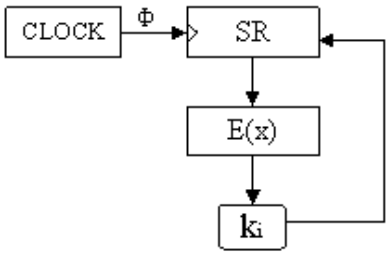

Figure 1: Asynchronous OFB Device

In order to provide the basis for a nontrivial security enhancement, we clock our device by using the key stream from another block cipher in OFB mode which is designated as CLOCK in Figure 1. The key stream of the clock should resemble a random sequence assuming a secure cipher is used. The clock controls the cipher action of the entire device, resulting in the production of one block of key stream, k_i, when the clock outputs a single binary value. The OFB device uses the feedback of the previous cipher action to produce k_{i+1} when the clock

outputs a 1 and produces the previous block of key stream, k_i, when the clock outputs a 0. Thus the device is said to be clocked when the clock produces a 1 and unclocked when the clock produces a 0.

3.1 Cryptanalysis

The analysis of the resulting key stream showed significant security faults which were imposed by the addition of the asynchronous clock. The primary security fault was the repetition of key stream blocks that increased the period to approximately one and a half times the birthday barrier while reducing the entropy of the sequence relative to a sequence from a device using OFB mode. When analyzing the entropy of the sequence, one can see that the repetition of a block adds the additional entropy associated with a single bit. This single bit represents the choice between repeating the block and not repeating the block. The repeated blocks can be removed and replaced with a bit representing the repetition of the previous block with no loss of entropy. Under the random oracle model, the probability that a block is repeated is $\frac{1}{2} + \frac{1}{2n}$, where n is the number of possible distinct blocks. We expect roughly half of the blocks to be repeated as the length of the message approaches infinity. If we remove the repeated blocks and replace them with a single bit, which is also random, the entropy reduction of a message with B bit blocks is $\frac{B+1}{2B}$. So the entropy reduction of a sequence approaches fifty percent as the block size approaches infinity. Analyzing the reduction in entropy with n being the number of possible distinct blocks, we have:

$$E = -\sum_{\forall i} P_i \log(P_i)$$

$$P_{old} = \frac{1}{2} + \frac{1}{2n} = \frac{n+1}{2n} \qquad \forall (i \neq old) P_i = \frac{1}{2n}$$

$$\begin{aligned} E &= -P_{old} \log\left(P_{old}\right) - P_{i \neq old} \log(P_{i \neq old}), \\ &= -\frac{n+1}{2n} \log \frac{n+1}{2n} - \sum_{i=1}^{n-1} \frac{1}{2n} \log \frac{1}{2n}, \\ &= -\frac{n+1}{2n} \log(n+1) + 1 + \log(n) \end{aligned} \tag{1}$$

For large n we can approximate:

$$E \approx -\frac{1}{2} \log(n) + \log(n) = \frac{1}{2} \log(n) + O(1)$$

3.1.1 Known Plaintext Attack

Not only does the repetition of $k_i s$ result in decreased entropy relative to a synchronous device, it also allows attackers to potentially recover several blocks of plaintext for each block of known plaintext. Each time a block of key stream

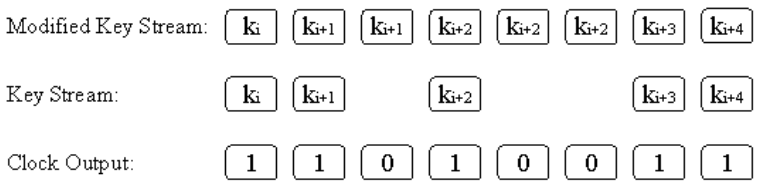

Figure 2: Key Stream of AOFB Device

is output from the device, there is a probability close to one half that the block selected will be the previous block, see Figure 2. An attacker with knowledge of a single plaintext block will be able to compromise the entire portion of plaintext which used the repeated block of key stream. For example, suppose that the jth plaintext block, P_j, is encrypted by xoring the ith key stream block, k_i, to produce ciphertext block $C_{j,i}$. An attacker with knowledge of P_j will be able to discover k_i and thus will be able to decrypt any ciphertext block of the form $C_{j+k,i} \forall k$, since this block of ciphertext is encrypted using the compromised block of key stream k_i.

3.1.2 Ciphertext Only Attack

With only the ciphertext, a device using AOFB mode is open to an attack hereafter referred to as "double block xor." A traditional cipher in synchronous OFB mode is not open to this attack as it does not repeat key stream blocks within the period of the sequence. Since blocks of key stream are repeated in AOFB mode, a skilled cryptanalyst could use this information alone to discover portions of or the entire message. Using two blocks of ciphertext encrypted using the same block of key stream, $C_{j,i}$ and $C_{j+1,i}$, an attacker could xor the two blocks to produce $P_{j,j+1}$, the double block xor with the key bits removed. Assuming the attacker is able to identify which blocks of the roughly $\frac{n}{2}$ block pairs contain the repeated key material through side channel cryptanalysis or by a brute force style attack, they could then xor ciphertext blocks which use the repeated key stream blocks and use a method similar to that used to cryptanalyze a reused one-time pad to identify the contents of the plaintext blocks.

3.1.3 Side Channel Cryptanalysis

Side channel cryptanalysis [3] of an asynchronous device may focus on timing attacks or alternative side channel methods including power consumption and processor load analysis to determine if the current block of key stream was a duplicate of the last block. For example, an implementation that simply output the stored value of key stream when the device was clocked zero would suffer from a major security flaw due to the timing differences associated with a forward cipher action (clocked one) versus a simple output operation (clocked

zero). In addition, careful analysis of the processor load and power consumption could determine if a block was repeated or was the result of a forward cipher action. Since encryption involves multiple operations in addition to the loads associated with the output of a stored value, it takes more power and time than a single output of a cached value. If the AOFB mode was a single component in a larger system, for example encrypting streaming data packets on a network, careful side channel analysis may not be able to ascertain weather the delays between encrypted packets was due to the cipher or network. More research should be done to explore side channel cryptanalysis of asynchronous modes.

3.2 Bitwise AOFB Mode

Up until now, we analyze AOFB mode under the assumption that the output of the AOFB device is a block of key stream. Hence the AOFB device repeats whole blocks of key stream, not just individual bits of key stream. When looking at the operation of a block cipher in OFB mode, this construction seems the most natural way to asynchronously clock the device.

A variant of our proposed method of clocking the OFB device would be to asynchronously clock the device in a bitwise fashion. A bitwise clocked AOFB device would output one block of k_i and an equal length block of clock output designated Φ_i. Within each block of clock output, a binary one in bit position j would cause a repetition of bit j in k_i. The blocks of key stream, k_i, would expand to the block size plus the Hamming weight of Φ_i. For example, given sequence k_i and Φ_i below, the output of the device would be α_i.

$$k_i : 01101001$$
$$\Phi_i : 01001011$$
$$\alpha_i : 011101100011$$

Under the random oracle model, we expect a uniformly distributed output of ones and zeros from both the clock and OFB device. Therefore, we expect an increase in the block size of the OFB device by approximately fifty percent. Thus, the period increases by the same amount as in blockwise AOFB, but since whole blocks are not repeated, patterns in the ciphertext are more difficult to detect.

Expanding on the bitwise idea, we notice that as the size of the repeated segment increases, the security of the device decreases. Security is reduced due to an attacker's ability to use repeated portions of key stream under the above mentioned attacks. Another way to view the security reduction of the device is to observe the reduction in entropy related to repeated segments of key stream. Using formula (1), we see that as the number of distinct blocks, n, increases, the difference between the maximum possible entropy $\log(n)$ and formula (1), widens. For small n, the entropy reduction of the device may only provide marginal cryptanalytic information. For instance when n equals 2, the entropy is reduced by approximately twenty percent. As n grows and the size of the repeated blocks grow, the reduction in entropy is sufficient to open the device

to our surveyed attacks. With large repeated blocks, formula (1) approaches $\frac{1}{2}\log(n)$, our lower bound on the entropy of the device.

With bitwise repetition instead of blockwise repetition, the device still provides information which could compromise the security of the device. A known plaintext attack on the expanded key stream, α_i, can recover additional unknown bits of key stream. The security flaw of this device derives from the fact that the probability of a bit j in α_i denoted $\alpha_{i,j}$ being the same as bit $\alpha_{i,j-1}$ is

$$\begin{aligned} P(\alpha_{i,j} = \alpha_{i,j-1}) &= P(\Phi_m = 1) + P(\Phi_m = 0) * P(k_m = k_{m-1}) \\ &= \frac{1}{2} + \frac{1}{4}. \end{aligned}$$

With such a high probability of repeated bits, the security of this device is neglible. No matter what size the repeated segment of key stream, any repetition provides an abundance of cryptanalytic information to an attacker.

3.3 Conclusion AOFB Mode

Clearly our original design is unusable in its current form due to the decrease in the entropy of the output. In general, AOFB clocking schemes are based on state transitions (forward cipher action) within a finite automaton where each state represents an output of the cipher. A general theory for the change in entropy of these clocking schemes is still an open problem. Continuing our analysis, we redesign our AOFB device to eliminate the repeated key stream information, in hopes of increasing the security of the device. We develop our new mode of operation with a focus on increasing security while maintaining efficiency and fault-tolerance.

4 Dual Stream Multiplexer Mode

In the analysis of AOFB mode, we conclude that the primary security problem is the repetition of key stream information which opens AOFB mode to several attacks which are highlighted in the last section. Dual Stream Multiplexer Mode or hereafter DSM mode is designed with these attacks in mind. Since the addition of the clock in AOFB mode reduces rather than increases the entropy of the sequence, we use the entropy associated with the clock in DSM mode to select between the key streams of several OFB devices, apparently increasing the complexity of cryptanalysis.

4.1 Implementation

The output of each synchronous OFB device, $s_1(t)$ and $s_2(t)$, connectes to a 2-to-1 multiplexer which uses the output of stream device S, designated $\Phi(t)$, to select between the streams of key bits. The clock is now a stream which consists of binary values that control the selection of key stream segments. If a value of 1 is output from the clock, a portion of key stream from OFB-2 is

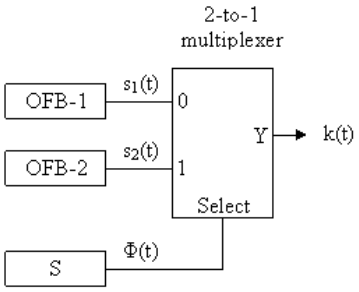

Figure 3: Dual Stream Multiplexer

used and if a value of 0 is output, a portion of key stream from OFB-1 is used. See Figure 3. Proper protocal for the implementation and usage of this device requires $\Phi(t)$ to be kept secret in addition to the secret keys of the OFB devices. Each OFB device should be uniquely keyed to prevent any possible repetition of key stream information, thus doubling the key length of a traditional OFB device. The same NIST-recommended cipher should be used in each OFB device to prevent distinguishing attacks which would allow an attacker to determine the origin of key stream blocks. The stream device S should be a device with the property that an attacker is not able to predict future bits of stream after having viewed any number of previous output bits and should be resistant to backtracking [4]. These properties are necessary to prevent an attacker from separating the multiplexed streams and thus reducing the security of the device to that of a single cipher in OFB mode.

4.2 Cryptanalysis of DSM Mode

Cryptanalysis of DSM mode focuses on two separate implementations of the device. The first implementation uses a single bit, Φ_t, to select between a block of key stream from OFB-1 or OFB-2. The sequence output consists of the key streams of each OFB device mixed in a blockwise fashion, hence the mode is named blockwise DSM. The second DSM mode uses a single bit, Φ_t, to select between a single bit of key stream from the OFB devices. In this mode, the key streams of the two OFB devices mix in a bitwise fashion hence, the mode is named bitwise DSM. Since the key stream of bitwise DSM has a greater increase in entropy than that of blockwise DSM mode, more time is spent analyzing bitwise DSM mode. The entropy increase of the DSM device is a function of the number of selections amongst key stream segments. With a block size $b > 1$, the entropy increase of blockwise DSM mode is $\frac{1}{b}$ the entropy increase of bitwise DSM mode.

4.2.1 Blockwise DSM Mode

In blockwise DSM mode, the key streams of the two OFB devices mix on a block by block basis. The device can be visualized as a pseudorandom selection between two size queues of key stream blocks. Let n_i be the period measured in number of blocks for device OFB-i. We show that selecting amongst blocks rather than bits eliminates most security added by the clock when a single key stream is compromised. We further analyze the security of the device by highlighting the complexity of likely attacks under different security assumptions including a known plaintext attack and a ciphertext only attack. In addition, we specify the complexity and method for launching a brute force attack given different amounts of key stream.

Ciphertext Only Attack First, we analyze the security of the device when only a trivial amount of key stream is known, in an attack similar to a ciphertext only attack. This attack assumes that an adversary is able to acquire at least one block of key stream, a practical assumption for large messages. Assuming the OFB device is implemented with full block size, b, feedback, an attacker could use one block of key stream as the initialization vector to the OFB device and brute force the key. If only one block of key stream is available, a brute force attack includes the additional complexity associated with trying all key stream positions in the ciphertext until a feasibe message is obtained. Being that this operation is on roughly half of the ciphertext, a large number of possible plaintexts can arise. The complexity of trying all key stream configurations assuming n_i blocks of the key stream are due to one OFB device is $\frac{(n_1+n_2)!}{n_1!n_2!}$.

Several blocks of key stream are required to verify with certainty that the correct key is obtained. Since the next block of key stream may or may not be derived from the same device and therefore key, an attacker has to search for the key that produces the correct sequence of additional key stream blocks, $k_{i+j}\forall j$, not the key that simply produces the next block of key stream k_{i+1}. If an attacker has an insufficient number of key stream blocks, absolute identification of the key may not be possible. False positive values or keys that produce the subsequent blocks of key stream which are not from the correct OFB device are possible although not likely.

Known Plaintext Attack A naive analysis would attempt to classify the complexity of brute force cryptanalysis under a known plaintext attack as proportionate to the complexity of a brute force attack against two separate OFB devices and the device producing stream S. Blockwise selection reduces the complexity of brute force cryptanalysis under a known plaintext attack to near that of one OFB device rather than two OFB devices. Since in a known plaintext attack, key stream blocks are clearly distinguishable from one another, once a single key stream is compromised, the selection of its position by stream $\Phi(t)$ adds no security. Even though stream S could permute blocks from one device $\frac{(n_1+n_2)!}{n_1!n_2!}$ ways, the blocks are identifiable therefore permutations are trivial.

4.2.2 Bitwise DSM Mode

Bitwise DSM mode selects between stream $s_1(t)$ and $s_2(t)$ in a bitwise fashion. Analogous to blockwise DSM mode, this device can be visualized as a pseudorandom selection amongst two size queues of key stream bits. This device maximizes the possible entropy increase realized by the selections of stream S. Also since a bit is the smallest possible unit to select, bitwise DSM maximizes the number of possible permutations of the key stream.

Ciphertext Only Attack We analyze bitwise DSM Mode assuming that a trivial amount of key stream has been discovered. For this device, around five times the block size number of bits are required to have seen two blocks from each OFB device with a high probability. A brute force style attack would try all permutations of the observed bits as the IVs to the OFB devices and then search for the next block of key stream. Even if one OFB device is compromised through a brute force style attack, this only provides the bits which will be permuted and says nothing about their location as determined by stream $\Phi(t)$. We know from our analysis of blockwise OFB that there are $\frac{(n_1+n_2)!}{n_1!n_2!}$ possible permutations when n blocks are generated by each OFB device. In blockwise, this function was not maximized since it related to n_i blocks, not n_i bits as in bitwise OFB. Hence, a brute force attack on bitwise DSM mode with minimal information has the maximum complexity increase associated with a DSM mode.

Known Plaintext Attack Next, we assume the attacker knows the plaintext of the device and therefore is able to identify the entire key stream. In traditional OFB mode, knowing the key stream is all the information required to launch a successful brute force or known plaintext attack. In bitwise OFB, knowledge of the plaintext alone is not sufficient to determine $s_1(t)$ and $s_2(t)$. The permutation of $\Phi(t)$ still provides a powerful cryptanalytic barrier since the number of permutations is significant and an attacker's ability to verify that they have determined the correct streams $s_1(t)$ and $s_2(t)$ is dependent on their ability to brute force each device. With a properly implemented stream device S, an attacker should not be able to predict any future outputs of the device and thus can not determine exactly the positions of any key stream bits.

Under a known plaintext attack with a compromised stream S, cryptanalysis of bitwise DSM mode reduces to the complexity of the best known plaintext attack on a single OFB device. Since in bitwise DSM mode, a single OFB device provides half of the bits of key stream, k_t, an attacker observing future output of the compromised device could use these bits of the plaintext to reduce the ciphertext to a partially decrypted message. Using statistical analysis and a dictionary attack, an attacker could cryptanalyze the remaining ciphertext. Once roughly half of the key stream bits have been obtained along with their positions, this device can no longer be considered secure. The additional OFB device merely adds a trivial amount of complexity as an attacker must searches for viable plaintext strings.

Period Length We previously set the period of each OFB device equal to n_i. When the two OFB devices have periods n_1 and n_2 and we exclude stream $\Phi(t)$, DSM mode repeats at the least common multiple of n_1 and n_2. Since stream $\Phi(t)$ permutes the two streams and for our analysis does not have period n_i, the security of the device is not compromised when the two OFB devices begin to repeat. When a traditional OFB device begins to repeat, visible repeated blocks of key stream arrive in the exact same order as they previously arrived. In DSM mode, the repeated blocks of key stream will be permuted in a different way up until the least common multiple of the periods of the two OFB devices and the stream device S. Only at this point does the device being to repeat the exact same sequence of key stream.

When one OFB device begins to repeat, subtle patterns should arise in the key stream. Depending how many times a single OFB repeats, continued usage of the device is possible. It is difficult to say at what point the security of the device is compromised. As a general rule, while the security of the device is not instantly compromised as in a traditional OFB device, the security of the device certainly degrades after a single OFB device begins repeating. In addition, it must be stated that if the device producing S has a period which is less than that of the OFB devices, an attacker who is able to determine the period of this device and use this information to seperate the key stream has reduced DSM mode to a single OFB device.

4.3 Conclusion DSM Mode

Assuming that Φ_t is kept secret and produced securely, bitwise DSM mode should double the key length and increase the complexity of cryptanalysis of a block cipher in OFB mode. This is an efficient way to increase the security of legacy block ciphers in OFB mode which requires minimal changes to old protocols. For legacy purposes, DSM mode is essentially just two OFB devices working in parallel. The fault tolerance of the device with respect to plaintext transmission errors remains the same as a single OFB device. One bit of plaintext error results in one bit of ciphertext error. The efficiency of DSM mode depends on the complexity associated with the production of stream S. The security of this device under a known plaintext attack is dependent on the security of stream device S. If an attacker is able to predict future bits of stream $\Phi(t)$ after having seen any number of previous output bits, then the security of this device is reduced to that of a single OFB device. The number of selections that stream $\Phi(t)$ must make over time depends on the size of the selected segment, block or bit. The recommended selected segment is a single bit, thus the block size of the OFB device number of bits of stream $\Phi(t)$ must be generated for every forward cipher action. Additionally, since the time complexity related to the rekeying of a cipher tends to be greater that the time complexity associated with producing a stream $\Phi(t)$, this device should represent a more efficient alternative to more frequent rekeying of OFB devices.

5 Conclusion

Although the initial assumption of increased complexity resulting in increased security was flawed, DSM mode represents an asynchronous mode that does not repeat blocks of key stream and is resistant to our survey attacks. We surveyed several designs and ultimately concluded that with additional research and analysis a mode similar to DSM mode provides an efficient security increase for legacy block ciphers operating in OFB mode.

6 Thank You

The authors of this paper would like to extend thanks to Professor Eric Bach of the University of Wisconsin-Madison, for his review, ideas, and assistance throughout the writing of this paper.

References

[1] D.W. Davis and G.I.P. Parkin, editors. *Cryptography, Proceedings of the Workshop on Cryptography*, volume The Average Size of the Key Stream in Output Feedback Encipherment, Burg Feuerstein, Germany, March 29-April2 1982. Springer-Verlag, 1983. pp. 263-279.

[2] Morris Dworkin. Recommendation for block cipher modes of operation. Nist special publication 800-38a, National Institute of Standards and Technology, 2001.

[3] J. Kelsey, B. Schneier, D. Wagner, and C. Hall. Side channel cryptanalysis of product ciphers. *Lecture Notes in Computer Science*, 1485:97–110, 1998.

[4] John Kelsey, Bruce Schneier, David Wagner, and Chris Hall. Cryptanalytic attacks on pseudorandom number generators. *Lecture Notes in Computer Science*, 1372:168–188, 1998.

[5] Donald Ervin Knuth. *The Art of Computer Programming*, volume Seminumerical Algorithms, chapter 3. Addison-Wesley, third edition, 1997.

[6] NIST. *First Modes of Operation Workshop*, http://csrc.nist.gov/CryptoToolkit/modes/workshop1/, July 2000.

[7] NIST. *Second Modes of Operation Workshop*, http://csrc.nist.gov/CryptoToolkit/modes/workshop2/, August 2001.

[8] Federal Information Processing Standards Publication. Fips pub 81, des modes of operation. Technical report, National Institute of Standards and Technology, December 1980.

[9] Bruce Schneier. *Applied Cryptography: Protocols, Algorithms, and Source Code in C*, chapter 9. New York: Wiley, c1996., 1996.

[10] Kencheng Zeng, Chung-Huang Yang, Dah-Yea Wei, and T.R.N. Rao. Pseudorandom bit generators in stream-cipher cryptography. Technical report, University of Southwestern Louisiana, http://crypto.nknu.edu.tw/publications/computer91.pdf, February 1991.