Analysis of a New Shift Cipher

Nikolai V. Yakovenko

May 7, 2004

1 Abstract

This paper cryptanalyzes a new stream cipher, designed by former University of Maryland student Mark Newman. Once we show the details of the scheme, we will then analyze its security. In particular, we show that the cipher is not CCA1 secure, meaning that it is vulnerable to a non-adaptive chosen ciphertext attack.

Then, we proceed to show that the cipher is semantically secure, meaning that given two encrypted messages of the adversary's choice, he can not then determine which encryption corresponds to what message.

We conclude with a few notes on the possible uses for this cipher.

2 The Cipher

We demonstrate the cipher by showing how it works in the two-bit model. For the remainder of the discussion:

- All computations are performed in base 2.
- Random bits used for the cipher are denoted f_n.
- Message bits are denoted m_n.
- Ciphertext bits are denoted c_n.

We look at the example for which the security parameter $n = 2$. The symmetric key consists of the random bits (f_1,f_2,f_3,f_4). The sender then computes random bits (f_5,f_6) and sends:

$$\begin{aligned} c_1 &= f_1 f_3 + f_2 f_4 + f_5 \\ c_2 &= f_2 f_3 + f_5 f_4 + f_6 \\ c_3 &= f_5 f_3 + f_6 f_4 + m_0 \end{aligned}$$

The receiving party uses (f_1,f_2,f_3,f_4) and (c_1,c_2,c_3) to compute:

$$\begin{aligned} f_5 &= f_1 f_3 + f_2 f_4 + c_1 \\ f_6 &= f_2 f_3 + f_5 f_4 + c_2 \\ m_0 &= f_5 f_3 + f_6 f_4 + c_3 \end{aligned}$$

In order to encrypt (m_0,m_1), the sender continues the encryption by using (f_3,f_4,f_5,f_6) as the shared key. He generates new random bits (f_7,f_8) and sends:

$$\begin{aligned} c_4 &= f_3 f_5 + f_4 f_6 + f_7 \\ c_5 &= f_4 f_5 + f_7 f_6 + f_8 \\ c_6 &= f_7 f_5 + f_8 f_6 + m_1 \end{aligned}$$

Decryption occurs as before.

The matrix representation of the above encryption is as follows:

$$\begin{pmatrix} c_1 \\ c_2 \\ c_3 \end{pmatrix} = \begin{pmatrix} f_1 & f_2 \\ f_2 & f_5 \\ f_5 & f_6 \end{pmatrix} \begin{pmatrix} f_3 \\ f_4 \end{pmatrix} + \begin{pmatrix} f_5 \\ f_6 \\ m_0 \end{pmatrix}$$

$$\begin{pmatrix} c_4 \\ c_5 \\ c_6 \end{pmatrix} = \begin{pmatrix} f_3 & f_4 \\ f_4 & f_7 \\ f_7 & f_8 \end{pmatrix} \begin{pmatrix} f_5 \\ f_6 \end{pmatrix} + \begin{pmatrix} f_7 \\ f_8 \\ m_1 \end{pmatrix}$$

We would extend the cipher to three bits by sharing the key (f_1,f_2,f_3,f_4,f_5,f_6) and then computing:

$$\begin{pmatrix} c_1 \\ c_2 \\ c_3 \\ c_4 \end{pmatrix} = \begin{pmatrix} f_1 & f_2 & f_3 \\ f_2 & f_3 & f_7 \\ f_3 & f_7 & f_8 \\ f_7 & f_8 & f_9 \end{pmatrix} \begin{pmatrix} f_4 \\ f_5 \\ f_6 \end{pmatrix} + \begin{pmatrix} f_7 \\ f_8 \\ f_9 \\ m_0 \end{pmatrix}$$

As we decrypt line by line, we notice that for security parameter n, the bit f_k is determined by the corresponding ciphertext bit c_j and the bits (f_{k-2n},...,f_{k-1}). We thus we say that the bit f_k, $k > 2n$ is uniquely determined by its $2n$ preceding bits.

3 Note on Uniform Distribution

A careful reader might notice that $f_1 f_3 + f_2 f_4$ is not randomly distributed in $\{0, 1\}$. Therefore, even though all c_i are randomly distributed and so are all f_i, the correspondence between an f_i and a ciphertext bit encoding is not random. However, as $n \to \infty$, this distribution becomes negligibly different from random.

Consider the distribution for security parameter n. Denote the probability that $f_1 f_{n+1} + f_2 f_{n+2} + \ldots + f_n f_{2n} = 0$ as p_n. Some manipulation can show that $p_n = \frac{1}{4} + \frac{1}{2}(p_n - 1)$.

Substituting the initial condition $p_1 = 3/4$, we can see that $p_n = \frac{1}{2} + \frac{1}{2^{n+1}}$. Therefore, the distribution of $f_1 f_{n+1} + f_2 f_{n+2} + \ldots + f_n f_{2n}$ is negligibly close to random.

4 Chosen Ciphertext Insecurity

In the CCA1 model, the adversary A is allowed to submit a polynomial number of messages to a decryption oracle. Based on this information, we show that he will be able to differentiate between two future ciphertexts.

Since the decryption takes place line by line, we notice that any $n+1$ bit ciphertext will decrypt to a valid one bit message for security parameter n.

Without loss of generality, submit the ciphertext $(c_1, c_2, \ldots c_{n+1}) = (111\ldots11) \in \{0,1\}^{n+1}$ to the decryption oracle. Denote the message received as m_0.

Now submit the ciphertext $(c_1, c_2, \ldots c_n) = (111\ldots1101) \in 0, 1^{n+1}$. Recover message m_0'.

Although we do not know the key, nor do we know the values (f_{2n+1},...,f_{3n}) used in the first decryption, we do know how the values (f_{2n+1},...,f_{3n}) relate to the values (f'_{2n+1},...,f'_{3n}) in the second decryption.

$$\begin{aligned} c_1 = 1 &= f_1 f_{n+1} + f_2 f_{n+2} + \ldots + f_n f_{2n} + f_{2n+1} \\ c_1' = 1 &= f_1 f_{n+1} + f_2 f_{n+2} + \ldots + f_n f_{2n} + f'_{2n+1} \end{aligned}$$

Thus, $f_{2n+1} = f'_{2n+1}$. Similarly for all other f_k, $2n+1 \leq k \leq 3n-1$. However, if we add the last two lines of both decryptions, we get:

$$\begin{aligned} c_n + c_n' = 1 + 0 &= f_{3n} + f'_{3n} \\ c_{n+1} + c'_{n+1} = 1 + 1 &= (f_{3n} + f'_{3n}) f_{2n} + m_0 + m_0' \end{aligned}$$

which simplifies to

$$f_{2n} = m_0 + m_0'$$

An example for $n = 2$ shows that if we submit messages (111) and (101) for decryption, we get:

$$\begin{aligned} c_1 = 1 &= f_1 f_3 + f_2 f_4 + f_5 \\ c_1' = 1 &= f_1 f_3 + f_2 f_4 + f_5' \\ c_2 = 1 &= f_2 f_3 + f_5 f_4 + f_6 \\ c_2' = 0 &= f_2 f_3 + f_5' f_4 + f_6' \\ c_3 = 1 &= f_5 f_3 + f_6 f_4 + m_0 \\ c_3' = 1 &= f_5' f_3 + f_6' f_4 + m_0' \end{aligned}$$

We notice that $f_5 = f_5'$ and so:

$$\begin{aligned} c_2 + c_2' = 1 + 0 &= f_6 + f_6' \\ c_3 + c_3' = 1 + 1 &= (f_6 + f_6')f_4 + m_0 + m_0' \end{aligned}$$

and so

$$f_4 = m_0 + m + 0'$$

Thus ends the example.

Now that we know f_{2n}, we can use the shift nature of the cipher to compute $(f_{n+1} \ldots f_{2n-1})$ by working backwards, line by line. We just need to be careful to make sure that for every decryption query, we are changing exactly one bit in $(f_{n+1}, \ldots, f_{2n})$.

For example, suppose that we just determined that $f_{2n} = 0$. We will next submit the ciphertext $(111 \ldots 1011)$ for decryption. We note that flipping the bit f_{3n-1} will not affect the computation of f_{3n}, since $f_{2n} = 0$ and so $f'_{3n-1}f_{2n} = f_{3n-1}f_{2n} = 0$ in the second to last line. Conversely, if $f_{2n} = 1$, then we would need to submit the ciphertext $(111 \ldots 1001)$ in order to determine f_{2n-1}.

These calculations take a maximum computation time of $O(n^2)$, and compute $(f_{n+1},\ldots,f_{2n})$ in exactly $n + 1$ ciphertext submissions.

Claim 1 *If we can determine $(f_{n+1},\ldots,f_{2n})$ in polynomial time, then we can break the entire cipher in polynomial time.*

Proof The simplest way to break the cipher is to submit a two-bit message to the chosen ciphertext oracle, and determine $(f_{2n+1},\ldots,f_{3n})$ by the same process as above. Note that $(f_{2n+1},\ldots,f_{3n})$ is not part of the key. In fact, we may be unable to break the entire key (if $f_{n+1} = 0$, we have no way of determining f_1, for example), however, the entire key is not needed. Note that we can rewrite the cipher as:

$$\begin{aligned} c_1 &= d_1 + f_{2n+1} \\ c_2 &= d_2 + f_{2n+1}f_{2n} + f_{2n+2} \\ c_3 &= d_3 + f_{2n+1}f_{2n-1} + f_{2n+2}f_{2n} + f_{2n+3} \\ &\vdots \\ c_n &= d_n + f_{2n+1}f_{n+2} + f_{2n+2}f_{n+3} + \ldots + f_{3n-1}f_{2n} + f_{3n} \\ c_{n+1} &= f_{2n+1}f_{n+1} + f_{2n+2}f_{n+2} + \ldots + f_{3n}f_{2n} + m_0 \end{aligned}$$

where the bits $(d_1,\ldots,d_n)$ are calculated deterministically from the key.

With knowledge of the bits $(d_1, \ldots, d_n)$ and the bits $(f_{n+1}, \ldots, f_{2n})$ of the key, we can encrypt or decrypt any message. We just gained knowledge of $(f_{n+1}, \ldots, f_{2n})$ and also $(f_{2n+1},\ldots,f_{3n})$ for a ciphertext, so we can calculate the bits $(d_1,\ldots,d_n)$ in linear time. Thus, we have solved the entire cipher. ■

5 Single Ciphertext Security

Please note that this is not a formal proof. The following should be viewed as an arguement.

Before we go on to prove the semantic security of the scheme, we will make a few notes about the scheme. In particular, we need to show that the scheme can not be broken by examining a single ciphertext. We can not prove that the cipher is based on an NP-complete problem, so the proof is more of an argument.

Any encryption scheme is based on a mathematical assumption. Our assumption is that we can not break a single ciphertext without learning n consecutive bits $(f_k, \ldots, f_{k+n-1})$ for some k.

We had stated before that each f_k is defined in terms of the $2n$ previous bits, as well as a known ciphertext bit. It follows that if we know $2n$ consecutive f_k's, we will know all of the f_k's from then on. We consider such knowledge as breaking the cipher.

In general, a cipher is considered insecure if we can probabilistically learn any bits of the message based on the ciphertext. Upon examining the cipher as described above, we notice that line $n+1$ of the cipher is almost identical to line $n+2$. If we add those lines, we get the following equation:

$$c_{n+1} + c_{n+2} \quad = \quad m_0 + f_{3n+1}$$

For our $n = 2$ example, we notice that

$$c_3 + c_4 = m_0 + f_7$$

This tell us that solving for the message bit m_0 is equivalent to solving for f_{3n+1}. Thus the adversary's task is equivalent to determining a bit f_{kn+1} for any $k > 2$.

Of course, the adversary can simply guess that bit. However, in order the break the scheme, we require the adversary to predict a messsage bit with the probability of $1/2 + f(n)$ where $f(n)$ is not negligible.

Before we prove that the adversary can not solve this problem, let us make some observations.

Claim 2 *The only knowledge that A can attain is the values of particular bits f_k, or functions of several bits. In other words, he gains no probablisitic knowledge of other bits.*

Sketch of Proof (Informal)

Suppose A learns bits f_1 and f_3. From the expression $c_1 = f_1 f_3 + f_2 f_4 + f_5$, A can learn the exact value of $f_2 f_4 + f_5$. A also has probabilistic information about f_2, f_4, f_5, or $f_2 f_4$. However, we argue that this knowledge will not help A defeat the overall scheme. For randomly distributed elements, $f_2 f_4$ will be 0 with probability 3/4. If A uses this probabilistic knowledge to evaluate $f_2 f_4$, A will succeed with probability of at most $3/4 \cdot \Pr[A']$, where A' is the algorithm that takes as input the value of $f_2 f_4$.

We can see that if A makes k such probablitic choices throughout his algorithm, then A will succeed with probability of at most $(3/4)^k$. Therefore, since the adversary can never get better than 3/4 probabilistic knowledge of any bit, his probability of success will become negligible if he uses probabilistic knowledge of more than $O(logn)$ bits.

However, we will soon demonstrate another assumption, that A is allowed to guess the values of $O(\log k)$ consecutive bits. Therefore, since A is already allowed to guess $O(logn)$ bits, the probabilistic information that he can gain is irrelevant. Thus, we conclude that A has no probabilistic knowledge. His only information can be learned from the $O(logn)$ bits that he is allowed to guess. ■

Claim 3 *The adversary A is allowed to guess $O(logn)$ bits of his choice and still remain polynomial, provided that he still generates enough information to prove the value of message bit.*

Sketch of Proof (Informal) With this assumption, we give the adversary additional powers but now require him to solve the scheme with the probability of 1.

For the arbitarily long ciphertext, we can check any guess that we think has solved the scheme. If we predict $2n$ consecutive bits, we can easily check the ciphertext to make sure we are correct. We strengthen our assumption to say that we assume an adversary A can generate a full $2n$ consecutive bits from just n consecutive bits, so A can check any n consecutive bit string against the ciphertext in polynomial time.

Suppose we had $n-1$ consecutive bits. We can now solve the cipher by computing the solution for both possible values of the $n+1^{\text{st}}$ bit, which would still take polynomial time.

We generalize to say that we can learn any bit f_i in exponential time. This means that for every request for an f_i, we must double the length of the algorithm. Thus for $O(logn)$ bits, we allow the adversary to guess the value correctly.

However, we can not simply allow the adversary to guess the answer, nor can with allow the adversary to guess bit expressions of $O(n)$ bits. The guessing property is a simulation of the adversary's ability to check an answer with incomplete information. If the adversary does not have enough information to generate n consecutive bits of the f_i's, he can not check his answer, thus he can not guess the bits. ■

Claim 4 *For any PPT adversary A, the probability that A will correctly predict f_{kn+1} for any $k > 2$ is negligibly close to 1/2, subject to the following assumptions:*

1. *All bits f_k are randomly distributed.*
2. *The only knowledge that A can attain is the values of particular bits f_k, or functions of several bits.*
3. *A can request particular bits f_k, but does so at an exponential expense. (A is limited to a logarithmic number of such requests, as a PPT adversary.) The adversary can not request the bit f_{kn+1}.*
4. *A is given a single, arbitrarily long ciphertext.*
5. *A will consider to have succeeded if he predicts bits f_i for $(k-1)n \leq i \leq kn$*

Sketch of Proof (Informal)

Initially, A has no information about any f_i, and is not allowed to act probabilistically for reasons explained above. All that A can do is to request bits and to preform computations.

Expressions for each c_i are independent, so A has no initial insight from the ciphertext. A essentially has to start by requesting a bit.

Suppose A requests the value f_i. In the best case for A, $f_i = 0$. By substituting into the equations, A eliminates one term each from $2n$ consecutive equations. However, A is now in no better shape than solving a ciphertext for security paramenter $n-1$. Thus by requesting single bits, A can not provably solve for n consecutive bits or for the value f_{kn+1} in $O(log(n))$ bit requests.

If A requests for bits of the form $f_i f_j$, he will not do much better. Particular expressions of the form $f_i f_j$ can sometimes appear in two different lines of the cipher, but will never appear more than twice. In any case, learning such an expression can at best shorten the length of $2n$ consecutive expressions by two terms only, if $f_i f_j = 1$, and if $f_i f_j = 0$, all knowledge we have of other bits becomes probabilistic, and thus for our purposes invalid.

Requesting bit expression of three or more bits will not even generate a reduction in the equations of more than one ciphertext line. Since the lines are linearly independent (three and four bit expressions do not reappear), any such knowledge will be only probabilistic.

Thus we conclude that A can not determine n consecutive bits of the cipher, nor can he determine f_{kn+1} provably in polynomial time. ■

Therefore, the cipher can not be broken by examining one ciphertext.

6 Semantic Security

This section should also be viewed as an arguement rather than a rigorous proof.

In order to show that the cipher is semantically secure, we need to prove that given encryptions of two messages of the adversary's choice, he can not distinguish between the two ciphertexts.

Suppose the adversary requests ciphertexts for messages M_0 and M_1. The key bits $(f_1, f_2, \ldots, f_{2n})$ will be identical for both ciphertexts. We will denote the random bits as f_k^0 if they contribute to the encryption of M_0 and as f_k^1 if they conribute to the encryption of M_1. We also make the convention $f_k^* = f_k^0 + f_k^1$. Similarly for $c_k^* = c_k^0 + c_k^1$ and $m_k^* = m_k^0 + m_k^1$. Thus we receive the ciphertexts:

$$
\begin{array}{rcl@{\qquad}rcl}
c_1^0 & = & f_1 f_{n+1} + \ldots + f_n f_{2n} + f_{2n+1}^0 & c_1^1 & = & f_1 f_{n+1} + \ldots + f_n f_{2n} + f_{2n+1}^1 \\
c_2^0 & = & f_2 f_{n+1} + \ldots + f_{2n+1}^0 f_{2n} + f_{2n+2}^0 & c_2^1 & = & f_2 f_{n+1} + \ldots + f_{2n+1}^1 f_{2n} + f_{2n+2}^1 \\
 & \vdots & & & \vdots & \\
c_n^0 & = & f_n f_{n+1} + \ldots + f_{3n-1}^0 f_{2n} + f_{3n}^0 & c_n^1 & = & f_n f_{n+1} + \ldots + f_{3n-1}^1 f_{2n} + f_{3n}^1 \\
c_{n+1}^0 & = & f_{2n+1}^0 f_{n+1} + \ldots + f_{3n}^0 f_{2n} + m_0^0 & c_{n+1}^1 & = & f_{2n+1}^0 f_{n+1} + \ldots + f_{3n}^1 f_{2n} + m_0^1
\end{array}
$$

Similarly for m_1, m_2, etc.

By assumption, we can not solve the ciphertexts individually. However, we can sum the

sequences. If we sum the sequences, we get:

$$\begin{aligned}
c_1^* &= f_{2n+1}^* \\
c_2^* &= f_{2n+1}^* f_{2n} + f_{2n+2}^* \\
c_3^* &= f_{2n+1}^* f_{2n-1} + f_{2n+2}^* f_{2n} + f_{2n+3}^* \\
c_4^* &= f_{2n+1}^* f_{2n-2} + f_{2n+2}^* f_{2n-1} + f_{2n+3}^* f_{2n} + f_{2n+4}^* \\
&\vdots \quad\quad \vdots \\
c_{n+1}^* &= f_{2n+1}^* f_{n+1} + f_{2n+2}^* f_{n+2} + \ldots + f_{3n}^* f_{2n} + m_0^*
\end{aligned}$$

since the constant key-based sections $(d_1, \ldots, d_n)$ of both ciphertexts largely cancel out.

By previous assumption, breaking the scheme is equivalent to solving for $2n$ consecutive bits f_i. Note that solving for $(f_{2n+1}^* \ldots f_{3n}^*)$ is equivalent to solving for $(f_{n+1} \ldots f_{2n})$, as either result gives us the other vector.

We note that the ciphertext sum takes the regular rectangular form for message bits m_k, $k \geq 1$. We can not solve this sequence by assumption, so the only computational advantage for the adversary would be to efficiently compute $(f_{2n+1}^*, f_{2n+2}^*, \ldots, f_{3n}^*)$ from the ciphertext shown above.

The reader may wonder whether such a computation, if possible, helps the adversary to solve one of the original messages. In fact, if the values f_k^* are randomly distributed as required, then the value of $(f_{2n+1}^*, f_{2n+2}^*, \ldots, f_{3n}^*)$ is in fact useful. If we have $(f_{2n+1}^*, f_{2n+2}^*, \ldots, f_{3n}^*)$, we may be able to solve for many of the values $(f_{n+1}, f_{n+2}, \ldots, f_{2n})$. Intuitively, we can see that for every value of $f_k^* = 0$, we learn nothing, but we get a new relation between the two ciphertexts for every $f_k^* = 1$. We will not go into the rigors of this proof, as we will show that A cannot efficiently compute $(f_{2n+1}^*, f_{2n+2}^*, \ldots, f_{3n}^*)$.

Theorem 5 *Given the sequence above, no PPT adversary A can compute the vector $(f_{2n+1}^*, f_{2n+2}^*, \ldots, f_{3n}^*)$ with non-negligible probability, even if that the adversary has knowledge of m_0^*.*

Sketch of Proof (Informal) We will assume that an adversary A will try to solve the sequence by moving from top to bottom, row by row. The rows of the sequence (excluding the last one) form a triangle with clear symmetry, so we argue that if A is able to form a relation from a row k and some previous rows, this same method should allow him to solve a higher row, so there is no reason to start at the bottom or near the middle. We do not require the adversary to solve for all values in a row before proceeding, however, we requre A to get a new relation for a row before proceeding. Perhaps this relation will help A solve a future row and thus simplify the overall computation. However if A does not get a relation from row k other than the original equation, A will have no chance to solve a future row with the same algorithm. Therefore, at each row, A must get a new relation, or A must request a bit.

We also note that the last row of the sequence does not in general give a way to solve for a bit f_k, if we do not already know $O(n)$ bits. Nor with the last equation give us any weighted probabilities for any bits. Therefore, any bottom-up approach to solving this equation will require considering $O(2^n)$ possibilities on the first step, thus lending a exponential time algorithm immediately.

Lastly, we note that solving the sequence by starting in the middle is no better than solving a rectangular sequence for one message, with the rectangle of size $O(n)$ by $O(n)$ instead of strictly $(n+1)$ by n. Thus, we will assume that the adversary solves the triangular sequence from top to bottom.

The first line of the sequence gives us f^*_{2n+1}. If $f^*_{2n+1} = 0$, then we can eliminate one term from each expression of the sequence. What we have instead is an identically formed sequence of n lines instead of $n+1$. Similarly for $f^*_{2n+2} = 0$, etc.

Since all f_k are randomly distributed, we know that this offset created by high end bits being 0 will affect the complexity of our computation with only negligible probability. Thus, without loss of generality, we assume that $f^*_{2n+1} = 1$.

Thus, we now have the equations:

$$\begin{aligned} c^*_2 &= f_{2n} + f^*_{2n+2} \\ c^*_3 &= f_{2n-1} + f^*_{2n+2} f_{2n} + f^*_{2n+3} \\ c^*_4 &= f_{2n-2} + f^*_{2n+2} f_{2n-1} + f^*_{2n+3} f_{2n} + f^*_{2n+4} \end{aligned}$$

We claim that we can not proceed past these messages without requesting a bit. As in the computation of the previous section, guesses will be treated as doubling the length of the overall algorithm.

To show that we cannot proceed, consider the two cases: $c^*_2 = 0$ and $c^*_2 = 1$.

Suppose $c^*_2 = 0 = f_{2n} + f^*_{2n+2}$. Then $f^*_{2n+2} = f_{2n} = 0$ or $f^*_{2n+2} = f_{2n} = 1$, with equal distribution. However, then $f^*_{2n+2} f_{2n}$ also has random distribution, so we can not learn the value of $f_{2n-1} + f^*_{2n+3}$ without guessing f_{2n} or f_{2n+1}.

Now suppose $c^*_2 = 1 = f_{2n} + f^*_{2n+2}$, then $f^*_{2n+2} f_{2n} = 0$ therefore, now we have the equations:

$$\begin{aligned} 1 &= f_{2n} + f^*_{2n+2} \\ c^*_3 &= f_{2n-1} + f^*_{2n+3} \\ c^*_4 &= f_{2n-2} + f^*_{2n+2} f_{2n-1} + f^*_{2n+3} f_{2n} + f^*_{2n+4} \end{aligned}$$

Consider the value $K = f^*_{2n+2} f_{2n-1} + f^*_{2n+3} f_{2n}$. In the case that $c^*_3 = f_{2n-1} + f^*_{2n+3} = 0$, the value of K is randomly distributed as above. Even if $c^*_3 = f_{2n-1} + f^*_{2n+3} = 1$, the value of K is still randomly distributed. Therefore, we can not learn $f_{2n-2} + f_{2n+4}$ without taking a guess.

Thus, in order to proceed past the first four lines of the sequence, we have to take at least one guess. Suppose this one guess gave us all of the values $(f^*_{2n+2}, f^*_{2n+3}, f^*_{2n+4})$. Even in the case that all of these values are 0, which is the optimum case for solving the sequence, we still have a identical looking sequence of $n-3$ lines, instead of $n+1$. By the same assumption, we will have to make at least one guess to proceed, etc.

Thus, the adversary A will make $O(n)$ bit requests. However, A is a PPT algorithm, and as such can only make $O(log(n))$ guesses. Therefore, A can not efficiently compute $(f^*_{2n+1}, f^*_{2n+2}, \ldots, f^*_{3n})$ with better than negligible probability. ■

Thus, since the adversary can not learn $(f^*_{2n+1}, f^*_{2n+2}, \ldots, f^*_{3n})$, he learns nothing useful from the triangular sequence in polynomial time. He can not solve the individual sequences in polynomial time, and can not solve the combined sequence either. Thus, we conclude that the adversary can not distinguish between the ciphertexts for M_0 and M_1 in polynomial time.

7 Possible Uses and Future Considerations

Despite being semantically secure, the scheme has two glaring weaknesses: its chosen ciphertext insecurity and also the $(n+1)m$ ciphertext of trasmitting an m bit message with security parameter n. However, the scheme does have one significant strength. The computational power of the message receiver need not be very strong.

While the sender of the message must generate $n \cdot m$ random numbers for each transmission, the receiver need not generate any. In fact, the reciever's computations can be performed with a simple circuit, since all that he needs is to perform single bit multiplication and bit addition. The receiver does not need more than $4n$ bits of memory, either, half of which permanently stores the initial key.

Thus, this scheme could be useful for a device that needs to decode messages quickly and without much computational power. A television descrambler or possibly a cellular phone descrambler could find this scheme useful. The television probably makes more sense, due to the relative cheapness of sending $(n+1)m$ bits for an m bit message through a cable line.

There does not seem to be an obvious way of shortening the length of the ciphertext for this cipher without significantly changing it. However, it should be possible to reduce or eliminate the chosen ciphertext threat. One simply needs to introduce a checksum into the decryption process, so that it will be hard to generate a random yet valid ciphertext. Generating a checksum such that the specific attack shown above will fail should be simple. However, devising a scheme that makes the scheme provably chosen ciphertext secure may be difficult. It is unclear whether this result is possible without significantly increasing the length of the cipherext.

8 Acknowledgements

This project could not have been possible without my advisor, Dr. Larry Washington of the University of Maryland Department of Mathematics, who has taught me much more than math.

References

[1] O. Goldreich. *Foundations of Cryptography, vol. 1: Basic Tools*, Cambridge University Press, 2001.

[2] M. Newman. Extended Linear Feedback Shift Register Encryption Scheme. (unpublished)

[3] J. Rotman. *Advanced Modern Algebra*, Prentice Hall, 2002.