Blind Synchronization of m-Sequences with Even Span

Richard A. Games and Joseph J. Rushanan

The MITRE Corporation, Bedford, MA, 01730, USA

Abstract. The problem of recovering the phase on a known binary m-sequence that is corrupted by a binary noise source is considered. This problem arises in the cryptanalysis of stream ciphers formed from a nonlinear combination of m-sequences. A synchronization procedure is developed for even span n. The procedure obtains a reliable estimate of the phase of an m-sequence of span n from unreliable estimates of the phases of a small number of shifts of a fixed m-sequence of span $n/2$. These latter estimates can be obtained from a variety of methods available in the literature. The procedure results in a reduction of complexity but requires observing on the order of the square root of the m-sequence's period.

1 Introduction

In this paper we focus on the problem of recovering the phase on a known binary m-sequence that is corrupted by a binary noise source. This problem arises in cryptanalysis and a number of methods have been suggested for its solution ([CS], [MS], [S], [ZH]). More precisely, we assume that we observe some terms of the binary sequence $\mathbf{r} = E^t\mathbf{s} + \mathbf{n}$, where $\mathbf{s}$ is the normal form of a known m-sequence of span n, E is the sequence shift-left operator (that is, $(E\mathbf{s})_i = (\mathbf{s})_{i+1}$), $\mathbf{n}$ is a sequence of independent and identically distributed binary random variables with probability p that $n_i = 0$, and the addition is modulo 2. We wish to recover the unknown phase t. We call this the blind synchronization problem to distinguish it from the usual synchronization process that takes advantage of approximate knowledge of the correct phase t.

We develop a blind synchronization procedure applicable when the span n of the m-sequence is even. We show how a reliable estimate of the phase of an m-sequence of span n can be obtained from unreliable estimates of the phases of the shifts of a fixed m-sequence of span $n/2$. A variety of approaches can be applied to obtain these latter estimates [CS], [MS], [S], [ZH]. The decrease in span from n to $n/2$ can result in a dramatic decrease in overall complexity. However, this complexity reduction is achieved at the expense of having to observe on the order of the square root of the m-sequence's period.

Section 2 reviews the array properties of m-sequences that we need, including the definition and properties of the shift sequence. Section 3 describes the new array blind synchronization procedure. Section 4 gives a performance analysis for the technique, and section 5 is the conclusion.

T. Helleseth (Ed.): Advances in Cryptology - EUROCRYPT '93, LNCS 765, pp. 168-180, 1994.

2 Shift Sequences of m-Sequences

Let f be a primitive polynomial of degree n, m an integer dividing n, $v = (2^n - 1)/(2^m - 1)$, and α a root of f. Then α is a primitive element of GF(2^n) and $\beta = \alpha^v$ is a primitive element of GF(2^m). Let Tr_m^n denote the trace mapping from GF(2^n) to GF(2^m). The shift sequence $\mathbf{e} = (e_0, e_1, \ldots, e_{2^n-2})$ of f for m dividing n is defined by

$$e_k = \begin{cases} \infty, & \text{if } \mathrm{Tr}_m^n(\alpha^k) = 0 \\ e, & \text{if } \mathrm{Tr}_m^n(\alpha^k) = \beta^e \end{cases}$$

The sequence $\mathbf{e}$ is called the shift sequence because when the m-sequence $\mathbf{s} = (\mathrm{Tr}_1^n(\alpha^i))$ is arranged in a $2^m - 1$ by v array, the nonzero columns of this array are comprised of shifts of the column m-sequence $\mathbf{c} = (\mathrm{Tr}_1^m(\beta^i))$. In particular, for $k = 0, 1, \ldots, v-1$, column k of this array is identically zero if $e_k = \infty$ and otherwise is equal to $E^{e_k}\mathbf{c}$.

The finite elements of the shift sequence can be viewed as elements of the integers modulo $2^m - 1$, denoted by $\mathbf{Z}_{2^m-1}$. By convention we have for any $e \in \mathbf{Z}_{2^m-1}$, $e \pm \infty = \infty$ and $\infty \pm \infty = \infty$. The shift sequence can be extended periodically, that is, the subscripts of $\mathbf{e}$ can be regarded modulo $2^n - 1$.

The shift sequence satisfies the following facts. The first two follow easily from the definition; the third is proved in [G].

FACT 1. *The first v terms of the shift sequence determine the remaining terms: For $a = 1, 2, \ldots, 2^m - 2$,*

$$(e_{av}, e_{av+1}, \ldots, e_{av+v-1}) = (e_0 + a, e_1 + a, \ldots, e_{v-1} + a).$$

FACT 2. *For $n = 2m$, $e_0 = \infty$ corresponds to the single zero column.*

FACT 3. *The shift sequence satisfies a uniform modular difference property: For a fixed integer difference offset $k \not\equiv 0 \pmod{v}$, the list of differences $\Delta_k = (e_{j+k} - e_j \pmod{2^m - 1} : j = 0, 1, \ldots, v-1)$ contains each element of $\mathbf{Z}_{2^m-1}$ exactly 2^{n-2m} times.*

3 Array Blind Synchronization

Let $\mathbf{s} = (s_i) = (\mathrm{Tr}_1^n(\alpha^i))$ be the normal form of a known m-sequence of span n, with $n = 2m$. We assume that an unknown phase t of $\mathbf{s}$ is transmitted and that $\mathbf{r} = E^t\mathbf{s} + \mathbf{n}$ is observed. We also assume the existence of a blind synchronization procedure for m-sequences of span n, denoted by BS(n, N), where N denotes the number of observed terms used. See, for example, the procedures described in [CS], [MS], [S], and [ZH]. In this section we describe a procedure that uses multiple applications of BS($n/2, l$), for some integer l, to produce an estimate for t.

The array blind synchronization procedure begins by collecting lv terms of $\mathbf{r}$, where $v = (2^{2m} - 1)/(2^m - 1) = 2^m + 1$, and by forming the $l \times v$ array

$$\mathbf{A}_l(\mathbf{r}) = \begin{bmatrix} r_0 & r_1 & \cdots & r_j & \cdots & r_{v-1} \\ r_v & r_{v+1} & \cdots & r_{v+j} & \cdots & r_{2v-1} \\ \vdots & \vdots & & \vdots & & \vdots \\ r_{(l-1)v} & r_{(l-1)v+1} & \cdots & r_{(l-1)v+j} & \cdots & r_{lv-1} \end{bmatrix}.$$

We assume that the array $\mathbf{A}_l(\mathbf{s})$ for the underlying m-sequence $\mathbf{s}$ of span n has the following form

$$\mathbf{A}_l(\mathbf{s}) = \begin{bmatrix} s_{uv+w} & s_{uv+w+1} & \cdots & s_{uv+w+j} & \cdots & s_{uv+w+v-1} \\ s_{(u+1)v+w} & s_{(u+1)v+w+1} & \cdots & s_{(u+1)v+w+j} & \cdots & s_{(u+1)v+w+v-1} \\ \vdots & \vdots & & \vdots & & \vdots \\ s_{(u+l-1)v+w} & s_{(u+l-1)v+w+1} & \cdots & s_{(u+l-1)v+w+j} & \cdots & s_{(u+l-1)v+w+v-1} \end{bmatrix},$$

where the integers u and w satisfy $0 \leq w \leq v-1$, $0 \leq u \leq 2^m - 2$, and $t = uv + w$. We wish to determine reliable estimates for the unknown sequence offsets u and w.

The nonzero columns of $\mathbf{A}_l(\mathbf{s})$ correspond to shifts of the column m-sequence $\mathbf{c} = (\mathrm{Tr}_1^m(\alpha^{vi}))$, where the shift sequence is denoted by $\mathbf{e} = (e_0, e_1, \ldots, e_{2^n-2})$. For $n = 2m$, shift-sequence fact 2 implies there is a single zero column corresponding to $e_0 = \infty$, but the position of this zero column is unknown in the array $\mathbf{A}_l(\mathbf{s})$. Also, for $n = 2m$, shift-sequence fact 3 implies that the modular differences in Δ_k are distinct for each difference offset k. Thus each Δ_k corresponds to a vector of length v with exactly two ∞ entries and with the remaining $2^m - 1$ entries corresponding to the distinct elements of $\mathbf{Z}_{2^m-1}$.

Since the column sequence $\mathbf{c}$ is known, the blind synchronization procedure BS(m, l) can be applied to the first two columns of $\mathbf{A}_l(\mathbf{r})$ to derive estimates $\hat{e}_0$ and $\hat{e}_1$ of the shifts e_{uv+w} and e_{uv+w+1}, respectively. The difference $\hat{d}_{10} \equiv \hat{e}_1 - \hat{e}_0 \pmod{2^m - 1}$ occurs in some unique position in the sequence of first differences Δ_1 of $\mathbf{e}$. If the column sequence estimates are both correct, then by shift-sequence fact 1, $\hat{d}_{10} \equiv \hat{e}_1 - \hat{e}_0 \equiv e_{uv+w+1} - e_{uv+w} \equiv e_{w+1} + u - (e_w + u) \equiv e_{w+1} - e_w \pmod{2^m - 1}$, and the unique position determined by $\hat{d}_{10}$ reveals the sequence offset w, as well as $w + 1$. It then follows that $u \equiv \hat{e}_1 - e_w \pmod{2^m - 1}$, and the sequence can be synchronized.

However, the estimates $\hat{e}_0$ and $\hat{e}_1$ are subject to error. If at least one of the estimates is in error, then the position P determined by $\hat{d}_{10}$ will be some other arbitrary value in the range of possible positions. To resolve this possibility, the blind synchronization procedure BS(m, l) is applied to the third column of $\mathbf{A}_l(\mathbf{r})$ to obtain the estimate $\hat{e}_2$ of the third column shift e_{uv+w+2}. If the estimates $\hat{e}_1$ and $\hat{e}_2$ are correct, the difference $\hat{d}_{21} \equiv \hat{e}_2 - \hat{e}_1 \pmod{2^m - 1}$ reveals the positions $w + 1$ and $w + 2$ in Δ_1. It is the consistent determination of position $w + 1$ by the differences $\hat{d}_{10}$ and $\hat{d}_{21}$ that signals that the estimates are most likely correct, and the computed sequence offsets are accurate.

If the positions determined by $\hat{d}_{10}$ and $\hat{d}_{21}$ are inconsistent, then at least one of the estimates $\hat{e}_0$, $\hat{e}_1$, and $\hat{e}_2$ is in error. It could be that only the second estimate is wrong, implying that the difference $\hat{d}_{20} \equiv \hat{e}_2 - \hat{e}_0 \pmod{2^m - 1}$ would correspond to the correct shift-sequence positions w and $w + 2$ in the sequence of unique second differences Δ_2. To verify this would require computing $\hat{e}_3$ using the blind synchronization procedure BS(m, l) on the fourth column of $\mathbf{A}_l(\mathbf{r})$. Then the differences $\hat{d}_{32}$, $\hat{d}_{31}$, and $\hat{d}_{30}$ would each yield a pair of positions to check.

In summary, the array blind synchronization procedure continues to estimate column shifts using BS(m, l) and then uses differences to obtain corresponding positions from the list of unique first differences Δ_1, second differences Δ_2, third differences Δ_3, etc. We assume that differences involving incorrect estimates will yield corresponding

positions that are uniformly spread over the range of possible positions. Note that as m increases it becomes less likely that two such randomly determined positions will collide. On the other hand, differences from correctly estimated column shifts will cluster and produce collisions at the correct sequence offset. A collision will occur as soon as three correct column estimates (or three consistent incorrect column estimates!) are obtained. When the first collision of positions occurs, we stop the process and use the three estimates involved to determine the estimates of the sequence offsets.

Before giving the precise algorithm for array blind synchronization, we construct the necessary data structures. Given integers k and d, $1 \leq k \leq 2^m - 1$ and $0 \leq d \leq 2^m - 2$, by the distinct modular difference property there is a unique pair of positions $\{P, P+k\}$ in the shift sequence, with $1 \leq P \leq 2^m$, such that $d \equiv e_{P+k} - e_P \pmod{2^m - 1}$. The position P corresponds to the position that d occurs at in the list of differences Δ_k. A $(2^m-1) \times (2^m-1)$ **position** array is defined by $\mathbf{position}(k, d) = P$. Since we exclude the difference ∞, the values of $\mathbf{position}(k, d)$ range from 1 to 2^m, except for position $2^m + 1 - k$, which would be paired with $e_{2^m+1} = \infty$. The algorithm creates for the jth estimated column shift $\hat{e}_j$ a list of up to j distinct positions (exactly j positions if the algorithm does not stop at the jth estimate). This list, denoted by $\mathbf{list}(j)$, contains the right-hand positions, that is, $P+k$, obtained from the j differences involving the jth estimate. The value $P + k$ is reduced modulo v in step 4 to account for the row length of $\mathbf{A}(\mathbf{s})$.

Array blind synchronization procedure:

1. Collect lv terms of $\mathbf{r}$ and form $\mathbf{A}_l(\mathbf{r})$;
2. Set $j = 0$, $\mathbf{list}(0) = \emptyset$, and $wrap = 0$;
3. Apply the blind synchronization procedure $\mathrm{BS}(m, l)$ to the jth column of $\mathbf{A}_l(\mathbf{r})$ to obtain the estimate $\hat{e}_j$ of the shift of the column m-sequence $\mathbf{c}$; if $j = 0$, then increment j by 1 and repeat step 3;
4. For each difference offset k, $1 \leq k \leq j$, compute the difference $\hat{d}_{j,j-k} \equiv \hat{e}_j - \hat{e}_{j-k} \pmod{2^m - 1}$; if $\mathbf{position}(k, \hat{d}_{j,j-k})$ occurs in $\mathbf{list}(j-k)$, then go to step 6; else add $\mathbf{position}(k, \hat{d}_{j,j-k}) + k \pmod{v}$ to $\mathbf{list}(j)$;
5. Increment j by 1 and go to step 3;
6. Set $P = \mathbf{position}(k, \hat{d}_{j,j-k})$.
7. The estimate for w is $\hat{w} = P + k - j$; if $\hat{w} < 0$, then add $v = 2^m + 1$ to the estimate $\hat{w}$ and set $wrap = 1$;
8. The estimate for u is $\hat{u} \equiv \hat{e}_j - e_{P+k} - wrap \pmod{2^m - 1}$;
9. Stop.

When step 6 is reached, $\mathbf{position}(k, \hat{d}_{j,j-k})$ corresponds to a position in the shift sequence corresponding to two consistent differences: adding k to it gives the rightmost position of the second consistent difference corresponding to the last estimate $\hat{e}_j$; subtracting j yields the estimate $\hat{w}$ of the position w involved in the first estimate: $\hat{e}_0 = e_{\hat{u}v+\hat{w}}$. The condition $\hat{w} < 0$ in step 7 is true when one of the estimated column shifts corresponds to the constant column. Adding v to the estimate produces the correct non-negative integer value of $\hat{w}$. The estimate for u can then be obtained as

in step 8, with the variable *wrap* set to 1 in the case where one of the column shifts corresponds to the constant column.

Before giving an example of the algorithm, we prove that after step 4 is completed, **list**(j) contains distinct entries.

PROPOSITION 1. *If all j values of k in step 4 are processed without satisfying the stopping condition, then* **list**(j) *contains j distinct entries.*

PROOF: We show that if **list**(j) contains a repeated entry, then the algorithm would have already stopped at the first occurrence of that entry. Suppose for some j, there exist k_1 and k_2, $1 \le k_1 < k_2 \le j$, such that $\textbf{position}(k_1, \hat{d}_{j,j-k_1}) + k_1 \equiv P \equiv \textbf{position}(k_2, \hat{d}_{j,j-k_2}) + k_2 \pmod v$. Suppose $\hat{e}_j \equiv e_P + u \pmod{2^m - 1}$. Then $\hat{d}_{j,j-k_1} \equiv \hat{e}_j - \hat{e}_{j-k_1} \equiv e_P - e_{P-k_1} \pmod{2^m - 1}$ implies $\hat{e}_{j-k_1} \equiv e_{P-k_1} + u \pmod{2^m - 1}$. Similarly, $\hat{e}_{j-k_2} \equiv e_{P-k_2} + u \pmod{2^m - 1}$. Thus, $\hat{d}_{j-k_1,j-k_1-(k_2-k_1)} = \hat{d}_{j-k_1,j-k_2} \equiv \hat{e}_{j-k_1} - \hat{e}_{j-k_2} \equiv e_{P-k_1} - e_{P-k_2} \pmod{2^m - 1}$, and so $\textbf{position}(k_2 - k_1, \hat{d}_{j-k_1,j-k_1-(k_2-k_1)}) + k_2 - k_1 \equiv P - k_1 \pmod v$ would have been added to **list**$(j - k_1)$ when the estimated shift $\hat{e}_{j-k_1}$ was processed. But then $\textbf{position}(k_1, \hat{d}_{j,j-k_1}) \equiv P - k_1 \pmod v$ occurs in **list**$(j - k_1)$, and the algorithm would have stopped at k_1 before completing all j values of k. This is a contradiction.

Example: Let **s** be generated by the primitive polynomial $f(x) = x^8 + x^4 + x^3 + x^2 + 1$. Then $v = 17$, and the column sequence is **c** = (000100110101111), generated by $g(x) = x^4 + x + 1$. The shift sequence is **e** = $(\infty, 2, 4, 2, 8, 12, 4, 0, 1, 9, 9, 14, 8, 5, 0, 3, 2)$. The first five difference offsets are:

j:	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16
e_j :	∞	2	4	2	8	12	4	0	1	9	9	14	8	5	0	3	2
$d_{j+1,j}$:	∞	2	13	6	4	7	11	1	8	0	5	9	12	10	3	14	∞
$d_{j+2,j}$:	∞	0	4	10	11	3	12	9	8	5	14	6	7	13	2	∞	1
$d_{j+3,j}$:	∞	6	8	2	7	4	5	9	13	14	11	1	10	12	∞	0	3
$d_{j+4,j}$:	∞	10	0	13	8	12	5	14	7	11	6	4	9	∞	3	2	1
$d_{j+5,j}$:	∞	2	11	14	1	12	10	8	4	6	9	3	∞	13	5	0	7

Note that offsets $k = 2, 3, 4$, and 5 use $(e_{18}, e_{19}, e_{20}, e_{21}, e_{22}) = (e_1 + 1, e_2 + 1, e_3 + 1, e_4 + 1, e_5 + 1) = (3, 5, 3, 9, 13)$.

The **position** array is:

difference:	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14
$k = 1$:	9	7	1	14	4	10	3	5	8	11	13	6	12	2	15
$k = 2$:	1	16	14	5	2	9	11	12	8	7	3	4	6	13	10
$k = 3$:	15	11	3	16	5	6	1	4	2	7	12	10	13	8	9
$k = 4$:	2	16	15	14	11	6	10	8	4	12	1	9	5	3	7
$k = 5$:	15	4	1	11	8	14	9	16	7	10	6	2	5	13	3

Suppose $(\hat{e}_0, \hat{e}_1, \hat{e}_2, \hat{e}_3) = (14, 6, 0, 3)$. The **list** array after iteration $j = 3$ and $k = 1$ is:

list(0)	$\emptyset$
list(1)	6
list(2)	12 1
list(3)	15

Then for $j = 3$ and $k = 2$, $\hat{d}_{31} = 3 - 6 = 12$ and $\mathbf{position}(2, 12) = 6$, which appears in $\mathbf{list}(1)$, and the algorithm jumps to step 6. Step 7 computes the estimate $\hat{w} = \mathbf{position}(k, \hat{d}_{j,j-k}) + k - j = 6 + 2 - 3 = 5$ and step 8 computes the estimate $\hat{u} = \hat{e}_3 - e_8 = 3 - 1 = 2$. Note that $(\hat{e}_0 - 2, \hat{e}_1 - 2, \hat{e}_3 - 2) = (12, 4, 1) = (e_5, e_6, e_8)$.

Next suppose $(\hat{e}_0, \hat{e}_1, \hat{e}_2, \hat{e}_3, \hat{e}_4, \hat{e}_5) = (2, 0, 8, 5, 4, 9)$. The **list** array after iteration $j = 5$ and $k = 1$ is:

$\mathbf{list}(0)$	$\emptyset$			
$\mathbf{list}(1)$	3			
$\mathbf{list}(2)$	9	13		
$\mathbf{list}(3)$	13	11	2	
$\mathbf{list}(4)$	16	6	8	2
$\mathbf{list}(5)$	11			

Then for $j = 5$ and $k = 2$, $\hat{d}_{53} = 9 - 5 = 4$ and $\mathbf{position}(2, 4) = 2$, which appears in $\mathbf{list}(3)$, and the algorithm jumps to step 6. Step 7 computes the estimate $\hat{w} = \mathbf{position}(k, \hat{d}_{j,j-k}) + k - j = 2 + 2 - 5 = -1$. Since $\hat{w} < 0$, the estimate is modified to $\hat{w} = -1 + 17 = 16$ and *wrap* is set to 1. Step 8 computes the estimate $\hat{u} = \hat{e}_5 - e_4 - 1 = 9 - 8 - 1 = 0$. Note that $(\hat{e}_0, \hat{e}_3, \hat{e}_5) = (2, 5, 9) = (e_{16}, e_{19}, e_{21}) = (e_{16}, e_2 + 1, e_4 + 1)$.

4 Performance Analysis

In this section we give a performance analysis for the array blind synchronization procedure. We assume that a given blind synchronization procedure $\mathrm{BS}(m, l)$ produces an estimate of the unknown phase of the jth column of the array $\mathbf{A}_l(\mathbf{r})$ with a probability of error $P_{\mathrm{col}}(m, l, p)$ and a computational complexity $C_{P_{\mathrm{col}}}(m, l, p)$, where recall p is the agreement probability that $r_i = s_{i+t}$. Our goal is to determine the probability of error P_{array} of the array blind synchronization procedure and its computational complexity. We begin with a noise only case that provides an upper bound on the expected number of applications of $\mathrm{BS}(m, l)$ required.

4.1 Uniformly Determined Shift Sequence Positions

Suppose that we are given a sequence of estimates $\hat{e}_0, \hat{e}_1, \ldots$ whose differences correspond to positions in Δ_k that are drawn uniformly and independently from the range 1 to 2^m, inclusive. We model the behavior of the array blind synchronization procedure in this situation using a multiple component version of the classic birthday repetition problem [F, page 31]. We use this model to determine how long we can expect the algorithm to run before we receive three consistent shifts.

Let $\mathbf{stop}(j)$ be the event that the array blind synchronization procedure stops while processing $\hat{e}_j$, and $\overline{\mathbf{stop}}(j)$ be the event that the algorithm does not stop. Define the probabilities

$$q_j = P(\overline{\mathbf{stop}}(j) \mid \overline{\mathbf{stop}}(j-1))$$
$$1 - q_j = P(\mathbf{stop}(j) \mid \overline{\mathbf{stop}}(j-1)).$$

Since three terms are required before stopping, $P(\mathbf{stop}(0))$ and $P(\mathbf{stop}(1))$ are both zero.

The array blind synchronization procedure can be described by the state diagram depicted in figure 1. Here open circles denote that the algorithm has not stopped yet, while closed circles are where the algorithm halts. The q_j are the transition probabilities. The probability of reaching any given state is the sum over all paths from the starting state to the ending state of the probability of following that path. (In figure 1 there is a unique path from the starting state to any given ending state.) The probability of a path is just the product of the transition probabilities multiplied by the probability of the starting state. Thus we see that for $j \geq 2$:

$$P(\overline{\textbf{stop}}(j)) = q_2 q_3 \cdots q_j$$
$$P(\textbf{stop}(j)) = q_2 q_3 \cdots q_{j-1}(1 - q_j)$$

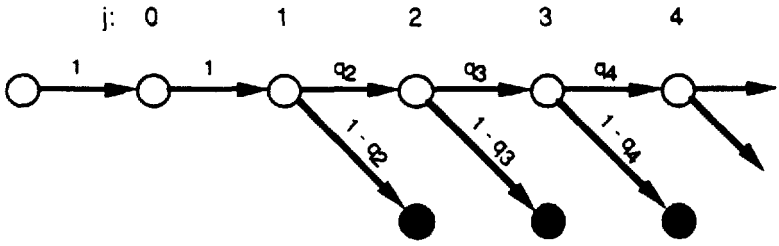

Figure 1. State Diagram for the Array Blind Synchronization Procedure

We wish to evaluate the q_j. Suppose that we do not stop at the jth estimated shift. Then for each difference offset k, $1 \leq k \leq j$ in step 4 of the array blind synchronization procedure, the difference $\hat{d}_{j,j-k} \equiv \hat{e}_j - \hat{e}_{j-k} \pmod{2^m - 1}$ determines a $\textbf{position}(k, \hat{d}_{j,j-k})$ that does not occur in $\textbf{list}(j-k)$, which by proposition 1 has $j-k$ distinct entries. If we assume that the new positions are determined uniformly in the range 1 to 2^m, then the probability that the new position is not in $\textbf{list}(j-k)$ is

$$\frac{M - (j-k)}{M},$$

where $M = 2^m - 1$ (there are 2^m possible positions, but one value is excluded because of the ∞ term).

Assuming independence, we take

$$q_j = \prod_{k=1}^{j} \frac{M-(j-k)}{M} = \frac{M-(j-1)}{M} \cdot \frac{M-(j-2)}{M} \cdots \frac{M-1}{M} \cdot \frac{M}{M}.$$

However, this value is not accurate because we have not forced the elements of the form $\textbf{position}(k, \hat{d}_{j,j-k}) + k \pmod{v}$ in $\textbf{list}(j)$ to be distinct as required by proposition 1. To accurately model this extra requirement, the probability of adding the second element to $\textbf{list}(j)$ would have to depend on which value was chosen for the first element and so forth. This soon leads to a large number of cases. For large values of M that we are interested in, simulations indicate that it is safe to ignore this extra requirement. This is because, as we shall see, the array blind synchronization process

halts before it is likely that a repeated element is encountered in the formation of **list**(j).

Figure 2 shows a graphical comparison between the experimentally determined probability of stopping and our simple model. The simulation, unlike the analysis, forced the elements of **list**(j) to be distinct. The solid curve in the figure is the theoretical prediction and the dashed curve is the probability distribution derived from 250,000 experimental trials for $M = 2^{16} - 1$ corresponding to span $n = 32$. The close agreement between the curves in figure 2 suggests that the assumptions used in the calculation of the q_j are accurate. (The rise at the end of the empirical curve is because lengths greater than or equal to 150 were accumulated in one bin.)

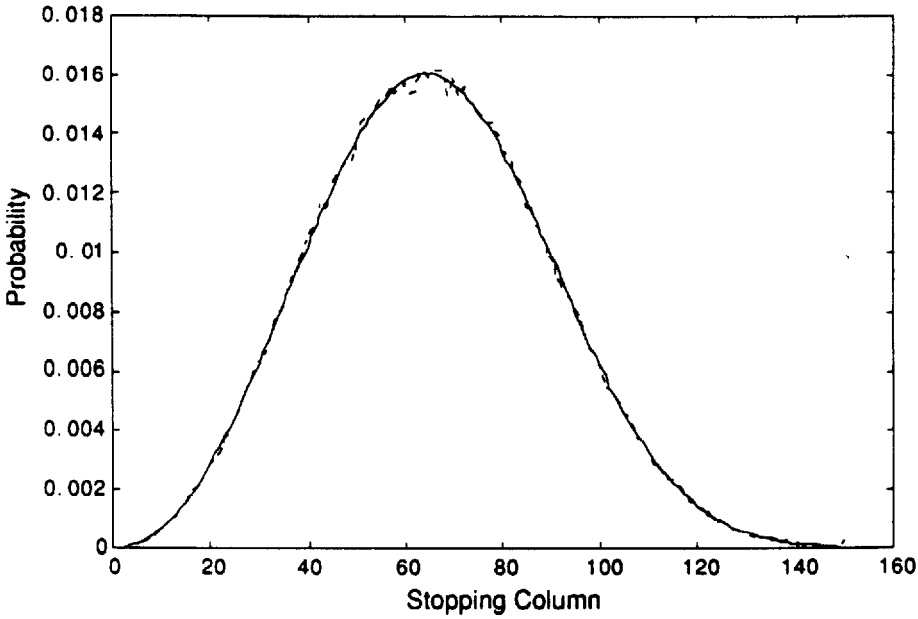

Figure 2. Probability Distribution for Stopping at Column j in Array Blind Synchronization Procedure ($n = 32$) (Theoretical Curve Solid; Empirical Curve Dashed)

The expected number of columns processed by the array blind synchronization procedure can be calculated using this simple model. These means represent an upper bound on the expected number of applications of the column blind synchronization procedure BS(m, l, p) when there is a sequence present. These means are listed below for some representative values of the span n:

span n	Mean
16	11.8
20	17.9
24	27.5
28	42.7
32	66.9

4.2 Probability of Error of Array Blind Synchronization

We next examine the case that the column blind synchronization procedure produces an estimate $\hat{e}_j$ that has a probability of error $P_{\text{col}}(m, l, p) = \alpha$ and thus is correct with probability $\beta = 1 - \alpha$. Figure 3 shows a transition diagram that distinguishes stopping correctly or incorrectly. The open circles represent non-terminal states. The closed circles represent stopping incorrectly due to three consistent shifts. Obtaining a correct estimated shift corresponds to moving down one row in the diagram. Finally, the last row of closed squares represent stopping correctly, that is, three correct estimated shifts have been obtained.

The transition probabilities q_j are the same as those in the previous section. The r_j give the probability that the array blind synchronization procedure halts correctly, given that the third correct estimated shift has been obtained. When the third correct estimated shift is encountered, the algorithm may still halt incorrectly. This is because the third correct estimated shift may be consistent with one or more earlier wrong estimated shifts, and this incorrect consistency is encountered first by the algorithm. Thus the probability of stopping incorrectly when two correct estimated shifts have been received is composed of two parts, based on whether the last shift is correct or not. In our experiments, we set r_j equal to q_j. This value is only an approximation, since it ignores the consistency that must occur with the third correct estimated shift.

It is straightforward to calculate the path probabilities to obtain for $j \geq 2$:

$$C_j = P(\text{stop at } j \text{ correctly}) = \binom{j}{2}\alpha^{j-3}\beta^3 q_2 q_3 \cdots q_{j-1} r_j$$

$$\begin{aligned} I_j &= P(\text{stop at } j \text{ incorrectly}) \\ &= \left(\alpha^j + j\alpha^{j-1}\beta + \binom{j}{2}\alpha^{j-2}\beta^2\right) q_2 q_3 \cdots q_{j-1}(\alpha(1-q_j) + \beta(1-r_j)). \end{aligned}$$

These probabilities satisfy

$$\sum_{j \geq 2}(C_j + I_j) = 1.$$

The probability of error for the array blind synchronization procedure is

$$P_{\text{array}}(n, P_{\text{col}}(m, l, p)) = \sum_{j \geq 2} I_j.$$

The probability of stopping correctly is $1 - P_{\text{array}}$, which is the sum of the C_j. Note that changing the value of r_j will change both C_j and I_j, but the sum $C_j + I_j$, which is the probability of stopping at the jth shift, remains unchanged.

To test our analytic results, we applied the array blind synchronization procedure to three actual m-sequences of spans 16, 24, and 32. The primitive polynomials were respectively $x^{16}+x^5+x^3+x^2+1$, $x^{24}+x^4+x^3+x+1$, and $x^{32}+x^{22}+x^2+x+1$. For each probability of column shift error α considered, an estimated shift sequence was derived from the actual shift sequence by selecting a correct shift with probability β; otherwise an incorrect shift was selected uniformly from the remaining $2^m - 2$ possibilities. The array blind synchronization procedure was then applied. This process was repeated 1000 times and statistics on the number of estimated shifts used and the number of correct phases obtained were tallied.

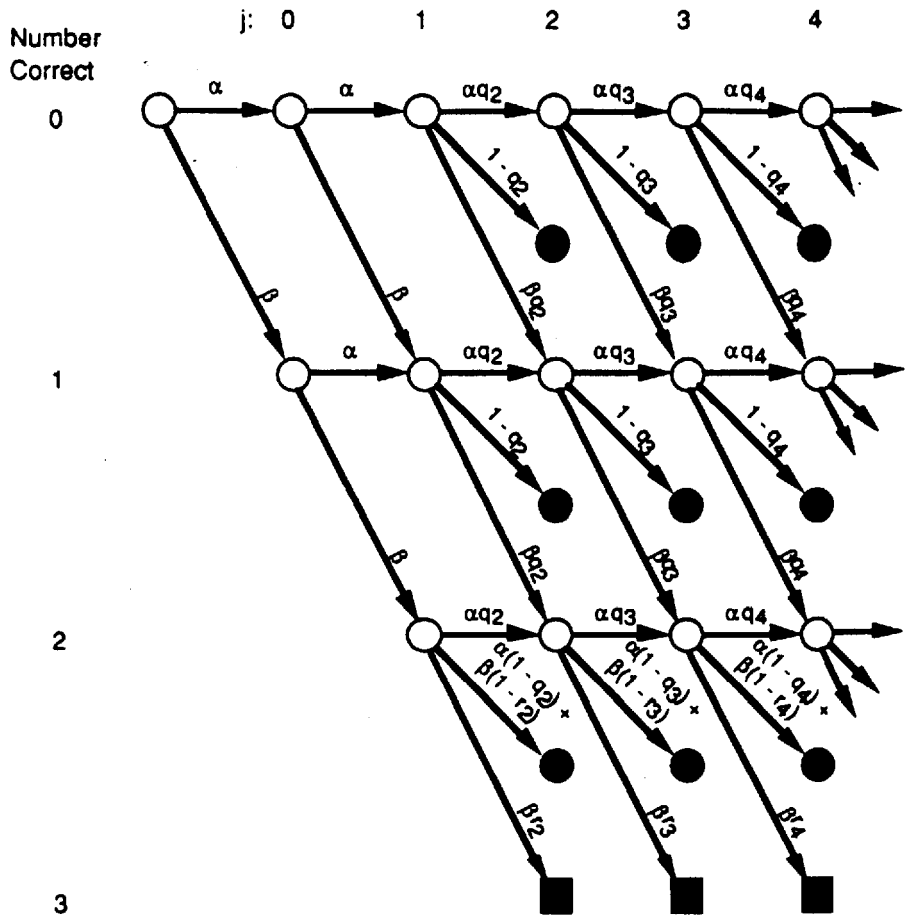

Figure 3. State Diagram for the Array Blind Synchronization Procedure

Figure 4 is a comparison of the analytic and simulation results for the expected number of columns processed by the array blind synchronization procedure as a function of the probability that the column estimates are in error. There is very good agreement between the analytic and simulation results. Also, as the column error probability approaches 1, the expected number of columns processed approaches the upper bound computed previously. Figure 5 is a comparison of the analytic and simulation results for the probability of error for the array blind synchronization procedure as a function of the probability that the column estimates are in error. Here we see that the agreement is not quite as good, due to our choice of r_j. Note that the closer fit between the experimental and theoretical curves in figure 4 can be explained by the fact that the expected number of columns processed is independent of the value of r_j, depending instead on the $C_j + I_j$.

Two facts are clear from these results. The modest expected number of columns processed means that the expected computational complexity of the array blind synchronization procedure is dominated by the computational complexity of the column

blind synchronization procedure $C_{P_{\text{col}}}(m, l, p)$. This should be much less than the complexity of applying the column blind synchronization procedure directly to the m-sequence of span n. Furthermore, to obtain a fixed probability of error P_{array}, a rather high probability of error $P_{\text{col}}(m, l, p)$ in the column shift estimation can be tolerated, with the situation improving for increasing span. Increasing the value of $P_{\text{col}}(m, l, p)$ decreases the computational complexity $C_{P_{\text{col}}}(m, l, p)$. Also, larger values of $P_{\text{col}}(m, l, p)$ permit smaller values of l. This is useful since the parameter l determines the amount lv of observed sequence required in the array blind synchronization procedure.

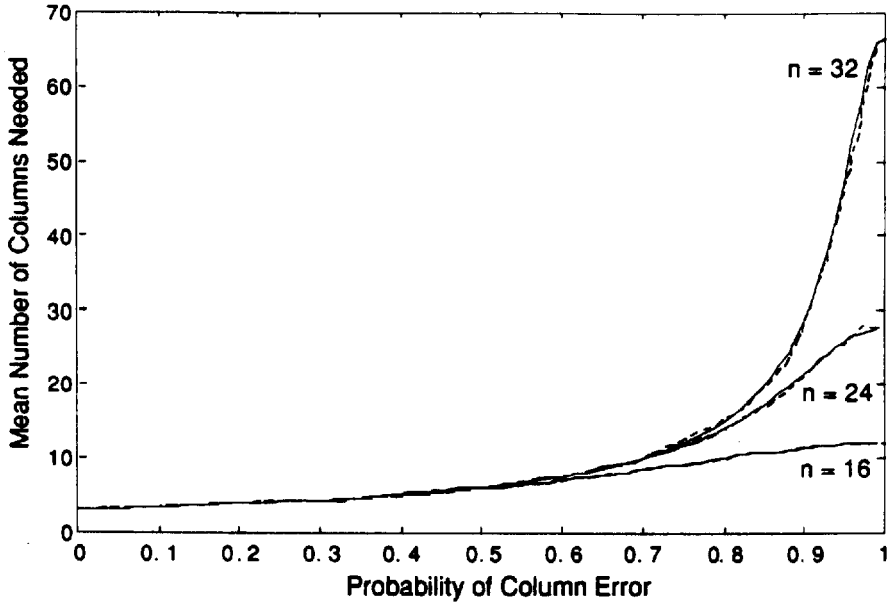

Figure 4. Average Number of Estimated Column Shifts used by Array Blind Synchronization as a Function of Probability of Error of Column Shift Estimation (Theoretical Curves Solid; Empirical Curves Dashed)

5 Conclusion

In this paper we developed a blind synchronization procedure applicable to m-sequences with even span n. In particular we showed how a reliable estimate of the phase of an m-sequence of span n can be obtained from unreliable estimates of the phases of a relatively small number of shifts of a fixed m-sequence of span $n/2$. The computational complexity of the procedure is dominated by the complexity of determining the phases of the smaller m-sequence of span $n/2$. The decrease in span from n to $n/2$ should result in a dramatic drop in complexity.

The procedure requires observing on the order of the square root of the period of the sequence. The observed terms are arranged in an array containing $2^{n/2} + 1$

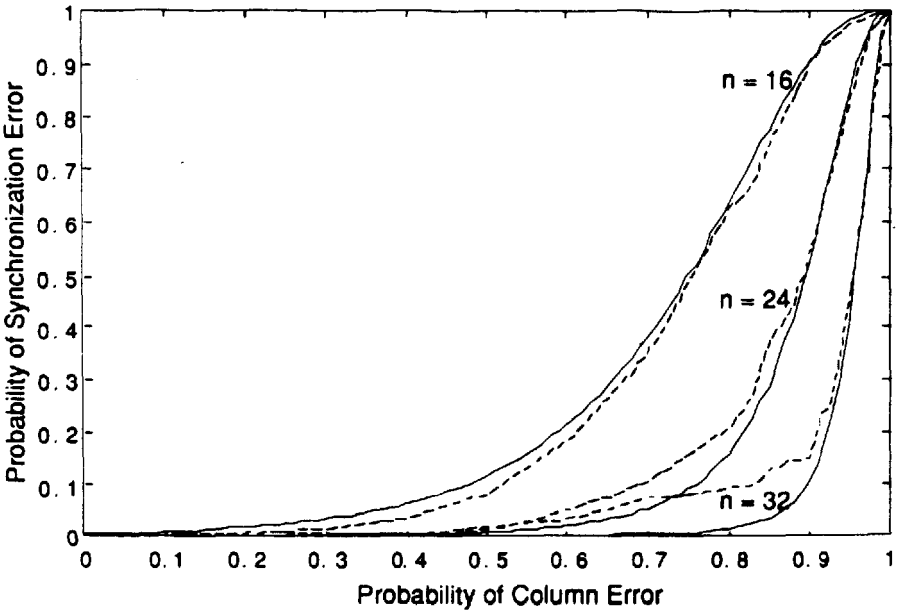

Figure 5. Probability of Error for Array Blind Synchronization as a Function of Probability of Error of Column Shift Estimation (Theoretical Curves Solid; Empirical Curves Dashed)

columns. The number of rows of this array can be minimized using the fact that the estimates of the phases of the shifts of the fixed m-sequence of span $n/2$ can have a high probability of error. Only a very small percentage of these terms are actually used in the procedure, and the required m-sequences of span $n/2$ can be collected directly using decimations of $2^{n/2}+1$. In the future, specific performance gains for the published blind synchronization procedures could be explored ([CS], [MS], [S], [ZH]), although we note that precise performance analyses of these techniques may have to be derived first.

The array blind synchronization procedure can be generalized to other factorizations of the span n, but the case $n = 2m$ is the most practical. This is mainly because for other factorizations, the long dimension of the array grows. It is also the case that the modular differences no longer are distinct when $n/m \neq 2$, which would further complicate the procedure.

References

[CS]. V. Chepyzhov and B. Smeets, "On a Fast Correlation Attack on Certain Stream Ciphers," *Advances in Cryptology—EUROCRYPT '91, Lecture Notes in Computer Science #547* (D. W. Davies, Editor), Berlin: Springer-Verlag, pp. 176–185, 1991.

[F]. W. Feller, *An Introduction to Probability Theory and Its Applications, Volume I*, New York: John Wiley & Sons, 1957.

[G]. R. A. Games, "Crosscorrelation of M-Sequences and GMW-Sequences with the same Primitive Polynomial," *Discrete Applied Mathematics*, Vol. 12, pp. 139–146, 1985.

[MS]. W. Meier and O. Staffelbach, "Fast Correlation Attacks on Certain Stream Ciphers," *J. Cryptography* , Vol. 1, pp. 159–176, 1989.

[S]. T. Siegenthaler, "Decrypting a Class of Stream Ciphers Using Ciphertext Only," *IEEE Transactions on Computers*, Vol. C-34, No. 1, pp. 81–85, 1985.

[ZH]. K. Zeng amd M. Huang, "On the Linear Syndrome Method in Cryptanalysis," *Advances in Cryptology—CRYPTO '88, Lecture Notes in Computer Science #403* (S. Goldwasser, Editor), Berlin: Springer-Verlag, pp. 469–478, 1990.