Fast Approximation of Anti-random Sequence Generator for Stream Ciphering

Shahram Esteki, Naser Movahhedinia, Ahmad Baraani
Computer Department, Faculty of Engineering
The University of Isfahan
Isfahan, Iran
naserm@ui.ac.ir

Abstract-Stream ciphering is an essential coding method for delay sensitive traffic such as voice and video. In this method the source sequence is combined with the output of a key stream generator using XOR. The strength and robustness of this ciphering method lays in complexity of the key stream generator. A well-known method for generating key stream is pseudo-random number generator, which is based on LFSR. Recently the anti-random sequence has attained attraction and shown to have better statistical properties compared to pseudo-random numbers, so it appears be suitable for stream ciphering. The existing methods for generating anti-random numbers are highly time and memory consuming. This paper presents an approximate method for generating anti-random sequence, which is very simple and does not need large capacity of memory, while preserves the properties of the sequence. Moreover, we show that our method presents enough statistical complexity to fulfill the required security for stream ciphering.

Keywords-Stream Ciphering; Anti-Random Sequence

I. INTRODUCTION

Acquiring security for digital voice and video transmissions has been an important issue during present decade. The difficulty of this job rises from the fact that these types of traffic (especially voice) are very delay and jitter sensitive. There are two basic methods for ciphering: Block ciphering and Stream ciphering. The symbol size used in stream ciphering is basically less than 32 bits, whereas block ciphering tends to have block size of 64, 128, 256 bits or even larger. Due to delay sensitivity, time consuming coding such as many block-ciphering methods can not be used for the purpose of voice encryption, even in data networks. The voice transmission systems mostly utilize stream ciphering, to fulfill the voice delay requirement [1].

In stream ciphering, to construct ciphered transmission sequence, a key is generated in pseudo-random fashion and combined with plain data using XOR. To decrypt received sequence, the same key is generated in the receiver side, and combined with encrypted data, using XOR again. To have a secure connection, key stream generator must have long period, with proper statistical characteristics so that it won’t be easily predictable. Moreover, not to impose large delay on ciphered voice, the key generation method should be simple, fast and easy to implement. A well-known method for this purpose is pseudo-random number generator, which is a system that takes a short string and expands it into a much larger string that ‘looks random’. The short string is called seed or key of ciphering, and the long pseudo random string is called key stream. The pseudo-random number generator is mostly based on Linear Feedback Shift Register (LFSR), which is a very simple and fast technique [2].

Recently another sequence of semi-random numbers, called Anti-Random (AR) sequence, has been introduced [3]. In this method each new number is generated in such a way that it has the largest distance from all the previously generated numbers. It has been shown that the correlation of this set of numbers is lower than pseudo-random numbers [4]. As a result, by having a subset of AR numbers, prediction of other parts of the sequence is more difficult. Having this property, one can imagine that if AR sequence is used in stream ciphering, the scheme will be more secure and robust.

However the main problem for using AR sequence in stream ciphering is its time-consuming generating algorithm. The AR sequence generators are mainly based on exhaustive search method, which is too computational, very memory hungry and too timely. As a result, despite of better properties, using AR sequence for online speech stream ciphering has been impractical.

In this paper, we present an approximate method for generating AR sequence. We call this method FARA (Fast Anti-Random Approximate) method, and show that it preserves the properties of AR sequence, while provides significant simplicity, speediness, and low memory requirement. The rest of the paper is arranged as followed; Section two introduces the definition of AR numbers and the way of generating them. Section three presents the approximate method for constructing AR sequence. In section four the computational complexity of the new approximation method is discussed, and section five presents simulation and numerical results. The paper is concluded in section six.
To generate AR sequence based on exhaustive search, one should look for the number with maximum distance from previously generated numbers. In a situation that AR sequence contains pairs of complementary vectors, we can reduce the computations by half [5]. The exhaustive method is too complex and requires long computation time and large capacity of memory.

An algorithm called Fast Anti-Random (FAR) has been proposed to ease the AR generation procedure [6]. Suppose that we express each number by a binary vector with M bits in each vector. The set of all M-bit binary vectors is a Euclidean M-space, with $2^M$ elements called M-vectors. Assuming that we have N anti-random vectors, a new, $(N+1)^{th}$ point in M-space, is found such that all vectors are distributed as evenly as possible. This is interpreted as finding a point with maximum distance from the existing N points in M-space.

To generate a new number by FAR algorithm, the first step is to calculate the binary centroid of all existing input vectors. By definition, a centroid of a set of vectors is their average. To obtain a binary centroid, FAR rounds vector elements to 0 for values less than 0.5, and to 1 if they are greater than 0.5. If the value equals 0.5, FAR randomly selects either 0 or 1. In the second step, the orthogonal vector with maximum distance from the centroid vector is found by inverting it.

### III. FAST ANTI-RANDOM APPROXIMATE (FARA) METHOD

Although FAR is much faster than exhaustive search, but still it needs on the order of $N \times M$ calculations and a large memory to generate each AR binary vector [5]. Therefore, FAR is not appropriate for real-time speech stream ciphering yet. To simplify the algorithm further, we propose the FARA method as described below.

To obtain the centroid vector, instead of calculating the average of all previous AR vectors and rounding the result, the difference between the number of ones and zeros in each bit position (column) of previously generated AR vectors is calculated, therefore there is no need for division. Since in a set of AR vectors the number of ones and zeros in each column are kept equal, the difference between the number of ones and zeros is either -1 or 0 or 1. If the mean value for a column is greater than 0.5, the number of ones in that column is greater than zeros, so the difference, and the corresponding centroid vector element will be equal to one. If the mean is less than 0.5, the number of zeros is greater than ones, and the difference will be equal to -1, so the corresponding element of centroid vector is set to 0. If the number of zeros and ones are equal, the difference is zero and either zero or one will be selected randomly. Thus, there is no need to store all previous vectors, and just $3M$ bits of memory is required to retain the difference between ones and zeros for each column. The next step, as in FAR, is to invert the binary centroid vector elements to generate the new AR number.

Since the speed of “shift” operation is higher than “add” (or “subtract”) operation, instead of subtraction and addition, shift to right (divide by 2), and shift to left (multiply by 2) are used respectively. To do so, we consider the values ‘1’, ‘2’ and ‘4’ instead of ‘-1’, ‘0’, and ‘1’ respectively.

The starting point of FARA method is to assign arbitrary zeros and ones to an initial AR vector. Then we set the values of zeros-ones differences to ‘1’ for corresponding AR vector element value ‘0’, and ‘4’ for value ‘1’. We proceed with calculating the binary centroid vector, and then the second AR vector by inverting its bits. The pseudo-code for generating M bits AR numbers by FARA method is given below.

#### Initialization:

```plaintext
for i=1 to M
    a[i]=rand(); // random number 0 or 1
    if a[i]=0
        sum[i]=1;
    else
        sum[i]=4;
endif
next i
```

#### Generating Anti-Random Sequence:

```plaintext
for i=1 to $2^M$
    for j=1 to M
        if a[j]=0 then
            sum[j]=sum[j]>>1; // divide by 2
        else
            sum[j]=sum[j]<<1; // multiply by 2
        endif
        if sum[j]=1 then
            a[j]=1;
            if sum[j]=4 then
                a[j]=0;
                if sum[j]=1 then
                    a[j]=rand(); // random number 0 or 1
            endif
next j
print a // a is the new anti-random vector
next i
```

### IV. COMPUTATION COMPLEXITY OF FARA METHOD

To generate a new M-bit AR number by FARA using $N$ previously generated AR numbers, we need only $M$ one bit shift operation, and a $3M$ bits memory to store the result of zeros-ones difference calculations. It’s worthy to mention that in exhaustive search method, the speed of generating a new number decreases exponentially as $N$ increases. In FAR this speed decreases linearly with $N$. However, in FARA method the speed of calculating a new AR number is constant, regardless of the value of $N$. Furthermore, to generate a complete set of $M$-bit AR sequence, the required capacity of memory in exhaustive search method, as in FAR, is in the order of $2^M \times M$ bit, while FARA method memory requirement is in the order of $3M$ bits.

### V. SIMULATION AND NUMERICAL RESULTS

#### A. Computation Complexity

Complete sets of AR sequence ($2^M$ AR vectors), with different number of bits ($M$), have been obtained by implementing exhaustive search, FAR and FARA methods, on a 2GHz Pentium-IV computer. Fig. 1 shows the computation
time in logarithmic scale (vertical axis) versus the number of bits for each method. As it is shown, for 20 bits AR sequence, the computation time for FARA method is about one and five orders of magnitude less than FAR and the exhaustive methods respectively. For instance, the time required for generating a set of 18 bits anti-random numbers is about one hour for exhaustive search method, 910 mili-second for FAR method, and 100 mili-seconds using FARA method. However, generating a set of 20 bits AR sequence, takes about 15.25 hours for exhaustive search and approximately 4.42 seconds for FAR method, and 430 mili-seconds for FARA method. This speediness of FARA method makes it appropriate for stream ciphering.

B. Statistical Properties

To be sure that statistical properties of AR sequence are preserved with FARA method, the autocorrelation function of 11 bits anti-random sequences generated by each of the exhaustive search and FARA, are obtained. The autocorrelations are calculated using window size 1024 points, and plotted in figure 2. This figure shows that both schemes result in sequences with auto-correlations similar to delta function, which is an indication of high randomness. Furthermore, the cross-correlation between two FARA sequences (with different initial values) is very low as shown in figure 3.

C. Properness for Stream Ciphering Key Generator

It is important to verify that FARA presents a sequence with enough statistical complexity that satisfies the required security for stream ciphering key. One of strong test algorithms for this purpose is Linear Complexity Test [7]. This test is based on the theorem that every sequence generator can be equalized to an LFSR sequence generator. The smallest length of an LFSR which is able to generate a sequence is called the linear complexity of that sequence. To find the linear complexity of a sequence, BMA (Berlekamp Massey Algorithm) can be used [7].
TABLE I presents the smallest LFSR length which is given by BMA algorithm, for eight sequences generated by FARA. For all the cases, the smallest LFSR length is equal to, or slightly more than half of the sequence length.

On the other hand, to break the security of a code, which is generated by an LFSR with length \( L \), at least \( 2L \) numbers of the sequence is required. In TABLE I, it is shown that for FARA the linear complexity is \( 2^{M-1} \) (or more). Thus, to predict any part of the sequence generated by FARA, the whole sequence is needed.

<table>
<thead>
<tr>
<th>M</th>
<th>Linear complexity</th>
<th>( 2^{M-1} )</th>
</tr>
</thead>
<tbody>
<tr>
<td>9</td>
<td>256</td>
<td>256</td>
</tr>
<tr>
<td>10</td>
<td>514</td>
<td>512</td>
</tr>
<tr>
<td>11</td>
<td>1028</td>
<td>1024</td>
</tr>
<tr>
<td>12</td>
<td>2052</td>
<td>2048</td>
</tr>
<tr>
<td>13</td>
<td>4102</td>
<td>4096</td>
</tr>
<tr>
<td>14</td>
<td>8196</td>
<td>8192</td>
</tr>
<tr>
<td>15</td>
<td>16387</td>
<td>16384</td>
</tr>
<tr>
<td>16</td>
<td>32768</td>
<td>32768</td>
</tr>
</tbody>
</table>

VI. CONCLUSION

In this paper, the anti-random sequence was considered as a candidate for key generator in stream ciphering. We proposed an approximate method, called FARA, to overcome the problem of latency in AR sequence generator. We showed that FARA is much faster than exhaustive search method, while preserves the statistical properties of AR sequence in terms of approximate delta shape autocorrelation and very low cross-correlation. Moreover, using linear complexity test, we showed that the sequence generated by FARA is enough statistical complex to be used as a secure key generator in stream ciphering.

REFERENCES