Blind estimation of scrambler offset using encoder redundancy

R. Gautier, Gilles Burel, Johnathan Letessier, O. Berder

36th Asilomar Conference on Signals, Systems and Computers, Pacific Grove, California, USA, Nov. 3-6, 2002
IEEE-ACSSC'2002, Vol 1, pp. 626-630

Blind Estimation of Scrambler Offset Using Encoder Redundancy

R. Gautier, G. Burel, J. Letessier, O. Berder

LEST - UMR CNRS 6165, University of Brest
BP 809, 29285 Brest cedex, France
Roland.Gautier@univ-brest.fr
http://www.univ-brest.fr/lest/tst/

Abstract

A self recovering receiver for encoded and scrambled binary data streams is proposed in this paper. The generating polynomial of the scrambling sequence is known, as well as the encoder structure and coefficients, but the scrambler time offset is unknown. Taking profit of redundancy introduced by the encoder, we propose a method which is able to estimate the scrambling sequence offset from the observed scrambled stream. The method is based on projection of the observed data on the encoder orthogonal subspace. Once the offset has been estimated, classical data descrambling and decoding can be used to recover the information stream.

1 Introduction

In many digital transmission systems, the encoded data is scrambled with a long pseudo-noise sequence in order to ensure a reasonable degree of privacy, as well as to provide a random binary stream with good spectral properties. As shown on figure 1, the binary data is first encoded for error protection (using, for instance, a block code or a convolutional code). Then the binary stream is scrambled by performing an Exclusive Or (XOR) with a long pseudo-random binary sequence. The scrambled binary data is finally fed to a digital transmitter which performs carrier modulation, filtering and amplification [2].

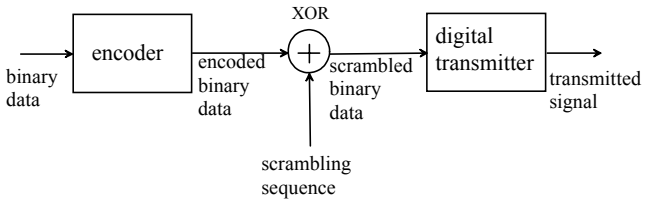

Figure 1: Principle of scrambling

On the receiver side, the received signal is demodulated to recover the scrambled binary data. Then, the data is descrambled and decoded. The classical receiver can perform data descrambling because it knows the scrambling sequence and its time offset. In this paper, we propose a receiver which is able to recover the information data stream without knowing the time offset of the scrambling sequence. Typical applications are multistandard adaptive receivers and spectrum surveillance [4]. The literature about self-recovering receivers is not very rich. In [4] a polyphase-based method to estimate an unknown short spreading sequence in spread spectrum transmission context has been proposed.

Let us come back to the scrambler. The scrambling sequence is a binary pseudo-random sequence generated by an m-stage shift register with linear feedback [3]. If the feedback coefficients are well chosen, the sequence length is $2^m - 1$. For instance, in the North American wireless communications standard IS-95 [1], the scrambling sequence is generated by a 42-stage shift register, which leads to a sequence length $2^{42} - 1$ (this corresponds to a period of about 40 days).

The feedback coefficients are in the public domain, in order to ensure that any receiver can compute the sequence. However, for a given communication, a time offset which depends on the mobile series number is applied to the sequence. Trying to descramble the data by "brute force" would require to test the $2^m - 1$ possible offsets, which is practically infeasible. In this paper, we show that, using linear algebra and a subspace-based method, the time offset can be determined without requiring a high computational power.

The paper is organized as follows. In Section 2, the proposed method is described. Then, in Section 3, experimental results using standard encoders and scramblers are provided to illustrate the approach. Finally, a conclusion is drawn in Section 4.

In the sequel, for illustration purpose, we provide examples related to IS-95 forward link. However, the method is not limited to this particular protocol.

36th Asilomar Conference on Signals, Systems and Computers, Pacific Grove, California, USA, Nov. 3-6, 2002
IEEE-ACSSC'2002, Vol 1, pp. 626-630

2 Proposed method

In order to allow the reader to choose the degree of details he wants, the proposed method is presented as follows. Subsection 2.1 describes the basic idea, from an intuitive point of view, without any equation. Then, Subsection 2.2 provides a global mathematical description of the method. Finally, Subsections 2.3 and 2.4 provide more details on the way to compute matrices which are used in the method

2.1 Basic idea

The basic idea of the method is to take profit of redundancy of the encoded data stream. Indeed, the principle of any error-correcting code is to add redundancy in order to provide error protection. When the code is linear, the effect of this redundancy is to confine the encoded data stream to a subspace. For instance, if we consider an N-dimensional vector containing N successive encoded bits, and a $1/r$ convolutional encoder, the vector is confined to an N/r-dimensional subspace[1]. The principle of our approach is to project the observed data on the orthogonal subspace. In the orthogonal subspace, contribution of the encoded data stream is null. Hence, only contribution of the scrambling sequence remains. Finally, by solving a system of linear equations, we determine the scrambling sequence offset.

2.2 Mathematical description of the method

Let us note m the number of registers in the scrambling sequence generator, N the number of observed binary symbols (typically, $N = 4m$), $1/r$ the encoder rate and K the encoder constraint length. Consider the binary vectors below:

h: the vector containing N samples of the scrambled data stream;

g: the corresponding outputs of the convolutional encoder (dimension N);

f: the corresponding outputs of the scrambling sequence generator (dimension N);

g_0: the corresponding unknown information data (dimension $N/r + K - 1$);

f_0: the binary representation of the scrambling sequence offset (dimension m).

According to figure 2, which summarizes the mathematical model, we have $h = f + g$ where $+$ stands for XOR. From the principle of linear encoders and random sequences generation, it can be shown that there exist matrices F and G such that $f = Ff_0$ and $g = Gg_0$. Hence, we have:

$$h = (Ff_0) + (Gg_0) \tag{1}$$

Matrices F and G can be computed as described in subsections 2.3 and 2.4, and vector h is observed. Vectors f_0 et g_0 are unknown. To estimate the scrambling sequence offset, we have to determine f_0. Globally, we have N equations (the dimension of h) and $m+N/r+K-1$ unknowns. Hence, we can expect to determine f_0, despite the fact that the equations are not classical linear equations, but equations on the Galois field GF(2).

[1] If border effects are neglected. See next subsection for more precise values.

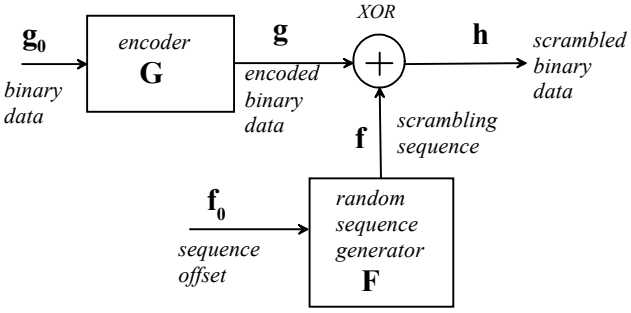

Figure 2: Mathematical model

Let us compute an $(N/r - K + 1) \times N$ full-rank matrix $\widetilde{G}$ such that:

$$\widetilde{G}G = \mathbf{0} \tag{2}$$

where $\mathbf{0}$ stands for the $(N/r - K + 1) \times (N/r - K + 1)$ null matrix. The rows of $\widetilde{G}$ span the subspace orthogonal to the encoded data subspace. Computation of $\widetilde{G}$ can be performed using singular value decomposition. For actual implantation under matrix-oriented software such as Matlab or Octave, we can note that the equation above is equivalent to $G^T\widetilde{G}^T = 0$, hence, the columns of $\widetilde{G}^T$ span the null subspace of G^T. Computation of the null subspace can be performed using singular value decomposition and is embedded in function *null* of Matlab and Octave. However, a difficulty appears due to the fact that, here, computations have to be done on GF(2). A solution is to force a rational representation of matrix $\widetilde{G}$ (thanks to option 'r'), and then use a modulo 2 reduction:

```
Gtilde = null(G','r')';
Gtilde = mod(Gtilde,2);
```

Then, we can compute:

$$\widetilde{h} = \widetilde{G}h = \left(\widetilde{G}Ff_0\right) + \left(\widetilde{G}Gg_0\right) = \left(\widetilde{G}F\right) f_0 \tag{3}$$

The interest is that the encoded data has no impact on $\widetilde{h}$. Let us note $H = \widetilde{G}F$. Finally, f_0 is obtained by solving the linear system below:

$$\boxed{\widetilde{h} = Hf_0} \tag{4}$$

If the inverse of $H^T H$ exists, we have:

$$f_0 = \left(H^T H\right)^{-1} H^T \widetilde{h} \tag{5}$$

If its inverse does not exist, an alternative method for solving linear systems in GF(2) can be used. If H is not full rank, many equivalent solutions exist.

To implement this method, please do not forget that **all computations must be performed in the Galois field GF(2)**, not in the set of real numbers. Otherwise, all results would obviously be false. For those who are not familiar with computations in Galois fields, extracts of sample Matlab programs will be posted on our web site after the conference.

2.3 Computation of matrix F using the model of the random sequence generator

For illustration purpose, we consider, here, a random sequence generator based on an SSRG (Simple Shift Register Generator) structure. The method can be easily adapted to any other linear random sequence generator.

When an SSRG is used, the sequence generator is defined by a polynomial $c(x)$ ([1] p. 562):

$$c(x) = 1 + c_1 x + c_2 x^2 + ... + c_{m-1} x^{m-1} + x^m \tag{6}$$

where m stands for the number of registers and the binary coefficients c_i are the feedback coefficients. This polynomial defines a recurrence equation which provides the sequence. For instance, the scrambling sequence used in IS-95 is generated by the polynomial below ([1] p. 352):

$$\begin{aligned} f_S(x) = 1 &+ x^7 + x^9 + x^{11} + x^{15} + x^{16} + x^{17} + x^{20} \\ &+ x^{21} + x^{23} + x^{24} + x^{25} + x^{26} + x^{32} + x^{35} \\ &+ x^{36} + x^{37} + x^{39} + x^{40} + x^{41} + x^{42} \end{aligned} \tag{7}$$

If we define a vector $x(t)$ which contains the state of the registers at time t, we can write:

$$x(t+1) = T.x(t) \quad \text{where} \quad x(t) = \begin{bmatrix} x_m(t) \\ x_{m-1}(t) \\ \vdots \\ x_1(t) \end{bmatrix} \tag{8}$$

and matrix T is:

$$T = \begin{pmatrix} 0 & 1 & \cdots & 0 & 0 \\ 0 & 0 & \ddots & 0 & 0 \\ \vdots & \vdots & \ddots & \ddots & \vdots \\ 0 & 0 & \cdots & 0 & 1 \\ 1 & c_{m-1} & \cdots & c_2 & c_1 \end{pmatrix} \tag{9}$$

Hence, we can write:

$$x(t+m) = T^m x(t) \tag{10}$$

Since the output is taken on register m ([1] p. 562, fig 6.2), the PN sequence is $f(t) = x_m(t)$. Finally, if we consider a vector f containing $N = (p+1)m$ successive samples, where p is an integer, we can write:

$$f = F f_0 \quad \text{where} \quad F = \begin{pmatrix} I_m \\ T^m \\ \vdots \\ T^{pm} \end{pmatrix} \tag{11}$$

We have:

$$f = \begin{pmatrix} x(0) \\ x(m) \\ \vdots \\ x(pm) \end{pmatrix} \quad \text{and} \quad f_0 = x(0) \tag{12}$$

Note that f_0 is the m-dimensional vector containing the initial registers states (in our application, it is the vector to determine). Matrix F is known because it depends only on the coefficients of the generating polynomial.

2.4 Computation of matrix G using the model of the linear encoder

As an illustration, we will consider a convolutional encoder, but the method can be easily extended to any linear code, such as block codes (Hamming, BCH, etc.). A convolutional encoder is defined by its generating polynomials. For instance, the IS-95 forward link uses a rate $1/2$ convolutional code, with constraint length $K = 9$ and the generating polynomials coefficients below ([1] p. 906-907):

$$\begin{aligned} poly_1 &= (111101011) \\ poly_2 &= (101110001) \end{aligned} \tag{13}$$

The coded stream z is then ([1] fig. 8.48 p. 908):

$$\begin{aligned} z_{2n} &= y_n + y_{n-1} + y_{n-3} + y_{n-5} + y_{n-6} + y_{n-7} + y_{n-8} \\ z_{2n+1} &= y_n + y_{n-4} + y_{n-5} + y_{n-6} + y_{n-8} \end{aligned} \tag{14}$$

where y stands for the uncoded data stream.

Let us note g an N-dimensional vector containing N successive samples of the coded stream and g_0 a vector containing the information sequence which generated g (dimension $N/r + K - 1$). We can write:

$$g = G g_0 \tag{15}$$

where

$$G = \begin{pmatrix} 1 & 1 & 0 & 1 & 0 & 1 & 1 & 1 & 1 & & & \\ 1 & 0 & 0 & 0 & 1 & 1 & 1 & 0 & 1 & & & \\ & 1 & 1 & 0 & 1 & 0 & 1 & 1 & 1 & 1 & & \\ & 1 & 0 & 0 & 0 & 1 & 1 & 1 & 0 & 1 & & \\ & & 1 & 1 & 0 & 1 & 0 & 1 & 1 & 1 & 1 & \\ & & 1 & 0 & 0 & 0 & 1 & 1 & 1 & 0 & 1 & \\ & & & \vdots & & & & & & & & \vdots \end{pmatrix} \quad (16)$$

Note that G contains the submatrices G_{sub} below (dimension $r \times K$):

$$G_{sub} = \begin{pmatrix} 1 & 1 & 0 & 1 & 0 & 1 & 1 & 1 & 1 \\ 1 & 0 & 0 & 0 & 1 & 1 & 1 & 0 & 1 \end{pmatrix} \quad (17)$$

Each row of G_{sub} contains the coefficients of a generating polynomial in reverse order. In our application, matrix G is known (because it depends only on the generating polynomials), but g and g_0 are unknown. However, instead of trying to determine them, we cancel their impact by projecting the data on the null subspace of G^T.

3 Experimental results

First of all, we illustrate the method by showing the content of various matrices for a given scrambler and encoder (subsection 3.1). Thereafter, in subsections 3.2 and 3.3 we give some simulation results in the context of IEEE802.11a WLAN and IS-95 standards.

3.1 Illustration of the method using matrices graphical representations

Let us consider a rate $1/3$ convolutional encoder with constraint length $K = 6$ with the 3 generating polynomials below ([1] p.907):

$$polyG = \begin{pmatrix} 1 & 0 & 0 & 1 & 1 & 1 \\ 1 & 0 & 1 & 0 & 1 & 1 \\ 1 & 1 & 1 & 1 & 0 & 1 \end{pmatrix} = \begin{pmatrix} 47 \\ 53 \\ 75 \end{pmatrix}_{octal} \quad (18)$$

This yields to matrix G_{sub} shown on figure 3 (black=1, white=0).

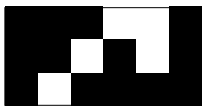

Figure 3: Graphical representation of G_{sub}

For illustration purpose, we consider a scrambler with $m = 15$ registers and the generating polynomial below:

$$f_S(x) = 1 + x^2 + x^6 + x^7 + x^8 + x^{10} + x^{15} \quad (19)$$

The offset of the scrambling sequence is:

$$f_0 = (110110000010011)^T \quad (20)$$

We will try to estimate the offset from $N = 2 \times m = 30$ scrambled bits. With G_{sub}, we can construct the encoder matrix G and from $f_S(x)$ given in Eq. 19 we can construct the scrambler matrix F. Equation 1 is graphically represented on figure 4.

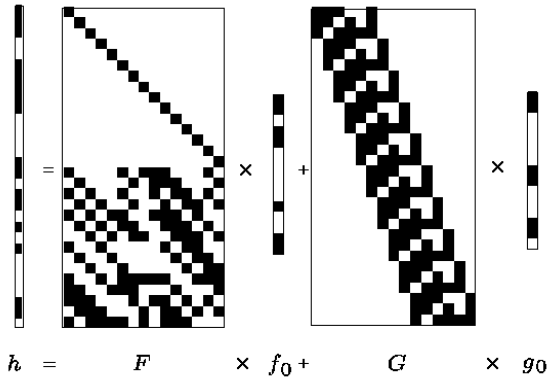

Figure 4: Graphical representation of the mathematical model (Eq. 1)

First, we compute $\tilde{G}$ (figure 5) as described in subsection 2.2. Then, we can compute $\tilde{h} = \tilde{G}h$, the projection of the observed symbols on the encoder orthogonal subspace. We also compute matrix $H = \tilde{G}F$.

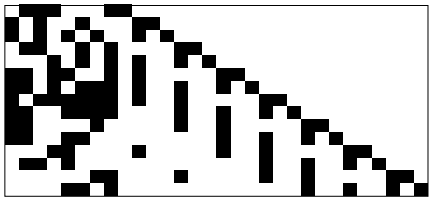

Figure 5: Graphical representation of $\tilde{G}$

Finally, the offset f_0 is obtained by solving the linear system (4). When the inverse of $H^T H$ exists, f_0 is directly provided by Eq. 5, which is illustrated on figure 6.

The estimated offset is exactly equal to the true offset used for the generation of the data stream, as shown on figure 7.

3.2 Example with IEEE802.11a W-LAN Standard

IEEE802.11a standard uses a rate $1/2$ and constraint length $K = 7$ convolutional encoder with the 2 generat-

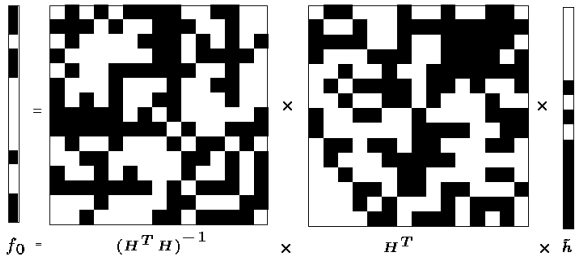

Figure 6: Graphical representation of Eq. 5

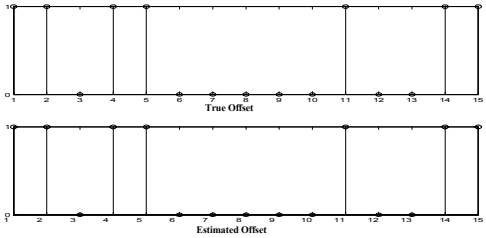

Figure 7: True and estimated offset for the illustration example

ing polynomials below:

$$polyG = \begin{pmatrix} 1 & 0 & 1 & 1 & 0 & 1 & 1 \\ 1 & 1 & 1 & 1 & 0 & 0 & 1 \end{pmatrix} = \begin{pmatrix} 133 \\ 171 \end{pmatrix}_{octal} \quad (21)$$

and the scrambler generating polynomial is:

$$f_S(x) = 1 + x^4 + x^7 \quad (22)$$

The number of registers in the scrambling generator is $m = 7$. We try to estimate the offset of the scrambling sequence from $N = 4 \times 7 = 28$ scrambled bits. We obtain a perfect estimation, as shown on figure 8.

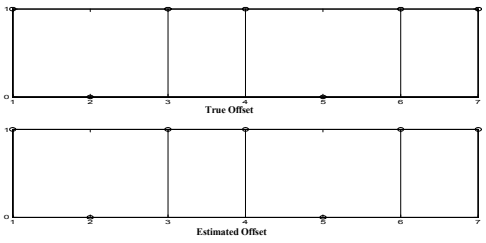

Figure 8: True and estimated offset for the IEEE802.11a W-LAN example

3.3 Example with IS-95 Standard

The IS-95 Standard (forward link) uses a rate $1/2$ and constraint length $K = 9$ convolutional encoder with generating polynomials 753 and 561 (see subsection 2.4) and a scrambler with $m = 42$ registers and generating polynomial given by Eq. 7.

We try to estimate the scrambler offset from $N = 5 \times 42 = 210$ scrambled bits. The method provides a perfect estimation, as shown on figure 9.

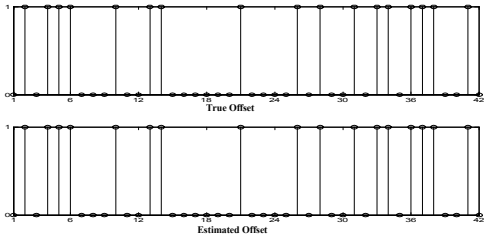

Figure 9: True and estimated offset for the IS-95 example

4 Conclusion

In this paper, we have considered the problem of determining the time offset of a long scrambling sequence. Since the binary data is usually encoded for error protection before scrambling, we have taken profit of redundancy introduced by the encoder to cancel the impact of the encoded data, and then to determine the scrambler time-offset. Thanks to linear algebra and computations in Galois fields, this idea can be mathematically expressed as projecting the scrambled data on the encoder orthogonal subspace, and then determining the time offset from the projected vector. Experimental results have been provided to illustrate and validate the approach.

References

[1] Jhong Sam Lee, Leonard E. Miller, *CDMA Systems Engineering Handbook*, Artech House Publishers, Boston, London, 1998, ISBN 0-89006-990-5

[2] John G. Proakis, *Digital Communications*, Third Edition, Mc Graw Hill Eds, 1995, ISBN 0-07-113814-5

[3] Dilip V. Sarwate, Michael B. Pursley, "Crosscorrelation Properties of Pseudorandom and Related Sequences", *Proceedings of the IEEE*, Vol. 68, No. 5, May 1980, pp. 593- 619.

[4] Michael K. Tsatsanis, Georgios B. Giannakis, "Blind Estimation of Direct Sequence Spread Spectrum Signals in Multipath", *IEEE Transactions on Signal Processing*, Vol. 45, No. 5, May 1997, pp. 1241- 1252.