Iterative decoding of Gold sequences
Mathieu Des Noes, Valentin Savin, Jean-Marc Brossier, Laurent Ros

To cite this version:
Iterative decoding of Gold sequences

Mathieu des Noes and Valentin Savin
CEA, LETI, Minatex campus
Grenoble, France
Email: {mathieu.desnoes, valentin.savin}@cea.fr

Jean Marc Brossier and Laurent Ros
GIPSA-Lab, BP46, 38402 Saint-Martin d’Hères, France
Email: {laurent.ros, jean-marc.brossier}@gipsa-lab.grenoble-inp.fr

Abstract—Gold sequences are widely used in communications and positioning systems for synchronization purposes or spread spectrum transmissions. This paper addresses the decoding of the initial state of a Gold sequence. This can be used to detect a harmful interferer closed to a 3G femtocell base station and implement interference mitigation techniques. The decoder implements an iterative message-passing algorithm which is built upon a parity check matrix. Thus, it depends on the coding properties of Gold codes. In this paper, we synthesize the coding properties of Gold codes and use them to compute the number of parity check equations of weight \( t = 3, 4 \) or \( 5 \). Eventually, the impact of the parity check equations used for decoding is highlighted.

INTRODUCTION

Gold sequences form a family of binary sequences with excellent correlation properties [1]. Hence, they are widely used for synchronization purpose in wireless communications and positioning systems, and also for scrambling in multi-user asynchronous CDMA systems [2][3]. A Gold sequence \( z \) is generated with a preferred pair of m-sequences \( x \) and \( y \) [1].

The conventional method to synchronize with a Gold sequence is to correlate the received signal with a replica of the searched sequence [4][5]. If a correlation peak is observed and is above a given threshold, the synchronization is declared. An alternative method consists in performing detection through a decoding of the received sequence. In fact, a m-sequence generator can be regarded as a linear code generator. It is thus possible to detect a transmitted sequence with a suitable decoder. This solution was originally proposed in cryptography for fast correlation attacks on stream ciphers [6][7]. This has been applied more recently in wireless communications and localization [8][9][10][11]. Exploiting the unique properties of m-sequences, an iterative message-passing algorithm can be implemented to decode the received signal [12]. All these papers focus on the decoding of m-sequences [1]. A notable exception is [13] which proposes a decoding procedure for Gold sequences. The approach consists in searching parity check equations which are multiple of \( g_z(x) \), the generator polynomial of sequence \( z \). The authors did not detailed the coding properties of Gold sequences. One objective of this paper is to clarify this point.

The decoding procedure is known to be sensitive to the weight of the parity check equations [14]. It is given by the number of non zero coefficients of the equation. It is thus fundamental to study these parity check equations for Gold sequences. In this paper, we first provide a synthesis of Gold sequence properties from the viewpoint of coding theory. A Gold sequence is a codeword of a cyclic linear code characterized by the generator polynomials of m-sequences \( x \) and \( y \). For the decoding procedure, we are interested in the properties of the dual code, since it defines the parity check equations used for decoding. This work will be helpful for evaluating the number of parity check equations of weight \( t = 3, 4 \) or \( 5 \). This tells us, if these equations at least exist and if yes, how many can be used for decoding. The goal is to use parity check equations having the smallest weight \( t \) for decoding.

The number of parity check equations of weight \( t = 3 \) and \( 4 \) has already been computed by Kasami in [15]. In this paper we provide an analytical expression for the number of parity check equations of weight \( t = 5 \) when the degree of the generator polynomial \( r \) is odd. To the best of our knowledge, this has not been given in the literature yet. Knowing this number is important because there is no parity check equation of weight \( t < 5 \) when \( r \) is odd. Then, we have measured the probability of missed detection as a function of \( t \) and the number of equations used for decoding. This highlights the decisive impact of selecting an even or an odd degree \( r \).

The paper is organized as follows. Section I details the generation of Gold sequences and the selection of the “preferred pair” of m-sequences used for generating a Gold sequence. Section II recalls some properties of linear cyclic codes and apply them to Gold codes. The number of parity check equations of weight \( t = 3, 4 \) or \( 5 \), when \( r \) is even or odd, is also given. Section III details the message-passing algorithm used for decoding. Section IV presents the decoding performance as a function of the degree \( r \), the weight \( t \) and the length of the message at the input of the decoder. Eventually, Section V concludes this paper.

Notations: a sequence will be written with upper case with the index \( k \) and with lower case \( x \) or \( y \). The index of a sequence is computed modulo the sequence’s length \( N \): \( s(k) = s(k \mod N) \). The modulo 2 binary addition is noted with symbol \( \oplus \).

I. GENERATION OF GOLD SEQUENCES

A Gold sequence \( z \) is obtained by the binary addition of a “preferred pair” of m-sequences \( x \) and \( y \) [16]:

\[
z(k) = x(k) \oplus y(k)
\]

We will first review the main properties of m-sequences and then explain the meaning of “preferred pair”.

A m-sequence is generated with a Linear Feedback Shift Register (LFSR) sequence generator. The feedback taps are
given by the generator polynomial \( g(x) = \sum_{k=0}^{\ell} g_k x^k \). It is primitive for a m-sequence, so that the sequence has the largest period among all the sequences that can be generated with a LFSR having the same number of registers. This is why they are named maximal length sequences, and in short m-sequences. If the primitive polynomial \( g(x) \) has degree \( r \), the period is \( n = 2^r - 1 \). There exist two structures for generating these sequences: Galois and Fibonacci generators [17]. They are depicted in Fig. 1 and Fig. 2. These two generators give sequences which are simply shifted with respect to each other. For instance, if the shift registers of the two generators are loaded with the same values, the Fibonacci architecture will generate sequence \( x(0), \ldots, x(n-1) \), while the Galois one will generate sequence \( x(i), \ldots, x(n-1), x(0), \ldots, x(i-1) \) for some \( i > 0 \).

There exists a decimation factor \( d \) between any two m-sequences \( x \) and \( y \) having the same length [1]: \( y(k) = x(dk) \). A “preferred pair” of m-sequences is such that \( d = 2^r + 1 \) and \( e \) satisfies [16]: \( \gcd(2^r + 1, 2^e - 1) = 1 \). This condition is met if:

\[
r \neq 0 \mod(4) \Leftrightarrow r \text{ is odd or } r = 2 \mod(4)
\]

and

\[
\gcd(e, r) = \begin{cases} 
1 & \text{if } r \text{ is odd} \\
2 & \text{if } r = 2 \mod(4)
\end{cases}
\]

Let \( SR_0(i) \) be the content of the \( i^{th} \) shift-register of sequence \( x \). The state of sequence \( x \) is the vector \( A_n = (SR_0(0) \cdots SR_0(r-1)) \). With the Fibonacci generator, the initial state of a sequence \( x \) is given by its first \( r \) chips: \( x(0) = SR_0(0), \ldots, x(r-1) = SR_0(r-1) \). This property is not valid for the Galois generator.

In this article, we will often use the polynomial representation of a codeword. This will be very useful to handle parity check equations. It is defined as follows: each codeword \( c = (c_0, \ldots, c_{n-1}) \) is associated with the polynomial \( c(x) = c_0 + c_1 x + \cdots + c_{n-1} x^{n-1} \). In the sequel, we use the conventional and polynomial representations indistinctly. Assume that \( f(x) \) is the generator polynomial of code \( C \), then every codeword \( c(x) \) is a multiple of \( f(x) \) modulo \( x^n - 1 \).

A cyclic code \( C \) fulfills the following property: if \( (c_0, \ldots, c_{n-1}) \in C \) then \( (c_{n-1}, \ldots, c_0, c_{n-2}) \in C \). As a consequence, shifting one bit of the codeword \( c \) is represented by the polynomial \( xc(x) \). Hamming, BCH, simplex and Gold codes belong to this family of linear cyclic codes.

Let \( C \) be a linear code \( [n, k] \), its dual \( C^\perp \) is the set of vectors orthogonal to every codeword of \( C \):

\[
C^\perp = \{ u | u.v = 0 \text{ for all } v \in C \} 
\]

\( u.v \) is the conventional scalar product between vectors \( u \) and \( v \).

When \( c \in C^\perp \), \( c(x) \) is referred as a parity check polynomial of \( C \). The weight of a codewords is defined by the number of non-zero elements (Hamming weight). By extension, this is also used for parity check polynomials. The dual code of a cyclic code is also cyclic. Assume that \( f(x) \) is the generator polynomial of code \( C \), the check polynomial of \( C \) is defined by: \( h(x) = (x^n - 1)/f(x) \). The generator polynomial of \( C^\perp \) is the reciprocal of the check polynomial: \( f^2(x) = x^r h(x^{-1}) \).

B. Gold sequence coding properties

In the domain of error correcting codes, a m-sequence is a codeword of a simplex code [18]. A simplex code \( S_k \) is a cyclic linear code \( [n = 2^r - 1, k = r] \) which check polynomial is \( g_k(x) \). A Gold code is also a cyclic linear code \( [n = 2^r - 1, k = 2r] \). The information bits are loaded at initialization in the \( r \) shift registers of the 2 m-sequences \( x \) and \( y \). The codeword is the generated sequence of length \( n = 2^r - 1 \) bits. The dual of a Gold code is the intersection of the dual codes of the two simplex codes \( S_k \) and \( S_y \) used to generate the sequences \( x \) and \( y \): \( S_k^\perp = S_k \cap S_y^\perp \). It is known from the literature that the dual code of simplex code \( S_k \) is the Hamming code \( H_k = [n = 2^r - 1, k = 2^r - r - 1] \) generated by \( g(x) \) [18]. Let \( \alpha \) denote the primitive root of polynomial \( g_k(x) \). If \( c(x) \in S_k^\perp \), it must verify \( c(\alpha) = 0 \) [18]. Now, let’s consider the case of \( c(x) \in S_k^\perp \). Due to the decimation property between sequences \( x \) and \( y \), the primitive root of \( g_y(x) \) is \( \beta = \alpha^d \). Thus, we shall have \( c(\alpha) = c(\alpha^d) = 0 \). \( \alpha \) and \( \beta \) being primitive roots of \( g_k(x) \) and \( g_y(x) \), \( c(x) \) must be a multiple of \( g_k(x)g_y(x) \). As a conclusion, \( S_k^\perp \) is the set of codewords which polynomial is a multiple of \( g_k(x)g_y(x) \).

C. Number of parity check equations

Let define \( N_t^* \) the total number of parity check equations of weight \( t \). Since \( S_k^\perp \) is a cyclic code, it is sufficient to enumerate only the parity check equations having a constant term equal to 1. The others are obtained by a simple cyclic shift. We note \( N_t \) the number of equations having a constant term. It is related to \( N_t^* \) by the following equation [19]:

\[
N_t = \frac{t}{2^r-1} N_t^*
\]
Let $A_j$ denote the number of Gold sequences of weight $j$. The Pless equality links $A_j$ and $N_j^*$ by the following equation [20]:

$$\sum_{j=0}^{N} (N-j)^t A_j = \sum_{j=0}^{t} \gamma_j N_j^*$$

(7)

where

$$\gamma_j = \sum_{k=0}^{t} k! S(t, k) 2^{2r-k} \binom{N-j}{N-k}$$

(8)

$S(t, k)$ is a Stirling number of the second kind:

$$S(t, k) = \frac{1}{k!} \sum_{i=1}^{k} (-1)^{k-i} \binom{k}{i} i^t$$

(9)

If the $A_j$’s are known, $N_j^*$ are computed by solving iteratively the set of $t$ equations:

$$\gamma_j N_j^* = \sum_{j=0}^{N} (N-j)^t A_j - \sum_{j=0}^{t-1} \gamma_j N_j^*$$

(10)

Since all '0' codeword belongs to $S_x$, we have $A_0 = N_0^* = 1$. Moreover $N_1^* = 0$ otherwise the Gold sequence would be equal to the all '0' vector whatever the initial state. Similarly, $N_2^* = 0$ otherwise, due to the “add and shift” property of m-sequences, they should exist a $r$ such that $x(k) \oplus y(k+r) = 0$ for all $k$. This is not possible because sequences $x$ and $y$ are generated by distinct primitive polynomials.

Coefficients $A_j$ have been evaluated by Kasami when he studied the cross-correlation properties of m-sequences. If $\delta = \text{gcd}(r, e) = \text{gcd}(r, 2e)$, then [15]:

$$A_0 = 1$$

$$A_{2^r-1} = 2^{(2r-1)} + 2^{(r-1)} + 1 N$$

$$A_{2^r+2(2r-1)} = 2^{(2r-1)} + 2^{(r-1)} + 1 N$$

$$A_0 = 0$$

elsewhere

(11)

It is remarkable that these numbers do not depend on the choice of the preferred pair of m-sequences. It is sufficient to satisfy the decimation property defined for selecting a preferred pair (see Section 1).

Practically, $\delta$ is defined as follows:

- if $r = 2 \mod (4)$: $\delta = 2$
- if $r$ is odd : $\delta = 1$

Inserting (11) in (10), we have the following results:

- if $r$ is odd : $N_3^* = N_4^* = 0$
- If $r$ is even and $r = 2 \mod (4)$:

$$N_3^* = 1$$

$$N_4^* = (2^r - 4)/3$$

(12)

These results have been already obtained by Kasami using a different approach [15].

When $r$ is odd, there is no parity check equation of weight $t < 5$. This has an impact on the decoding performance which are better for smaller $t$ [14].

We will now derive a theoretical formula of $N_k$ when $r$ is odd. To do so, we need to solve (10) with $t = 5$. Considering that $N_0^* = 1$ and $N_1^* = N_2^* = N_3^* = N_4^* = 0$, this reduces to:

$$N_k^* = \frac{1}{\gamma_5} \left( \sum_{j=0}^{N} (N-j)^5 A_j - \gamma_0 \right)$$

(13)

It is thus required to compute the 3 terms : $\gamma_5$, $\gamma_0$ and $\sum_{j=0}^{N} (N-j)^5 A_j$. Applying directly (8), we get:

$$\gamma_5 = 5! 2^{2r-5}$$

$$\gamma_0 = 2^{2r-5} (N^5 + 10 N^4 + 15 N^3 - 10 N^2)$$

(14)

The third term is equal to:

$$\sum_{j=0}^{N} (N-j)^5 A_j = a^5(2^{2r-1} + 5a A(2^{2r-1} - 2r-1))$$

$$+ 10a^3(2^{r-2} - 2^{2r-2})$$

$$+ 5(2^{2r-4} - 2^{2r-3})$$

$$+ N^5 + 2^{2r+2} - 2^{2r+1} - 2^{2r+2}$$

(15)

where $a = 2^{r-1} - 1$

The powers of $a$ and $N$ need to be expanded to compute $N_k^*$. We eventually obtain:

$$N_k^* = \frac{2^{2r+5} - 11 2^{2r-5} + 26 2^{2r-5} - 16 2^{2r-5}}{5! 2^{2r-5}}$$

(16)

Finally, $N_5^*$ is given by:

$$N_5^* = \frac{(2^{r-1} - 1)(2^{r-1} - 4)}{6}$$

(17)

Table I summarizes the values of $N_1$, $N_4$ and $N_5$ according to the primitive polynomial degree $r$.

<table>
<thead>
<tr>
<th>$r$</th>
<th>$N_1$</th>
<th>$N_4$</th>
<th>$N_5$</th>
</tr>
</thead>
<tbody>
<tr>
<td>$r = 2 \mod 4$</td>
<td>1</td>
<td>$(2^4 - 4)/3$</td>
<td></td>
</tr>
<tr>
<td>$r$ is odd</td>
<td>0</td>
<td>0</td>
<td>$(2^{r-1} - 1)(2^{r-1} - 4)/6$</td>
</tr>
</tbody>
</table>

In addition, the only parity check polynomial existing for $t = 3$ and $r$ even is known :

$$h_3(x) = x^{2(2r-1)/3} + x^{(2r-1)/3} + 1$$

(18)

Indeed, if $r$ is even, $h_3(x)$ is divisible by every primitive polynomial of length $r$ [21]. This means that $h_3(x) = h_3(x^d) = 0$ and, invoking the minimal property of $g_5(x)$ and $g_6(x)$, $h_3(x)$ is a multiple of $g_5(x)g_6(x)$. It is thus the polynomial representation of the only parity check equation of weight $t = 3$. This confirms the parity check polynomial found in [13] for the Gold sequence used in the GPS.

Table II shows the number of parity check equations of weight $t = 4$ or $t = 5$, depending on $r$. For $r$ even, only $N_4$ has been computed since there are obviously enough parity check equations of weight $t = 4$. Using an exhaustive search strategy, these theoretical values have been validated for $r \leq 11$. One observes that there are plenty of parity check equations available. This will not be a limiting factor.
III. ITERATIVE MESSAGE-PASSING DECODING

A. Iterative message-passing algorithm

The principle for the decoding of a Gold sequence is to build a sparse parity check matrix $E$ and then apply an iterative message passing algorithm on the induced Tanner graph $[13][12][22]$. This provides an approximation of the MAP decoding of the sequence.

The received sequence is noted $z$ and $R(i) = (-1)^{z(i)} + w(i)$ is the observation of variable $z(i)$ at the decoder input. $w(i)$ is an additive white Gaussian noise of variance $\sigma^2$. It is assumed the receiver observes $M$ elements of the sequence: $R(0), \ldots, R(M-1)$. For practical reasons, the decoder implements a Min-Sum (MS) algorithm [23] since it is known to be insensitive to a uniform scaling of input variables $R(i)$'s.

B. Parity check matrix

Let consider a sparse parity check polynomial of degree $m$: $c(x) = \sum_{k=0}^m c_k x^k$. Since the dual code is cyclic, all the codewords $x^i c(x)$ belong to $S^+_n$ and are parity check equations of $S_z$. It is thus easy to build a parity check matrix based on $c(x)$.

Assuming the entire sequence is observed, the parity check matrix is circulant and square ($M = n$):

$$E = \begin{bmatrix} c_0 & \cdots & c_m & 0 & \cdots & 0 \\ 0 & \cdots & c_0 & \cdots & c_m & 0 & \cdots \\ \vdots & \cdots & \vdots & \cdots & \vdots & \cdots & \vdots \\ 0 & \cdots & 0 & \cdots & c_0 & \cdots & c_m \\ c_m & \cdots & \cdots & 0 & \cdots & c_0 & \cdots \\ \vdots & \cdots & \vdots & \cdots & \vdots & \cdots & \vdots \\ c_1 & \cdots & c_m & 0 & \cdots & 0 & c_0 \end{bmatrix}$$

(19)

In order to improve decoding performance, an usual design strategy is to increase the column weight of the parity check matrix, while keeping the row weight constant. Having a large degree improves the probability to correct an error on that variable because it receives more information from its neighboring nodes. This is achieved by concatenating parity check matrices such as (19), generated by different parity check polynomials. The overall parity check matrix becomes:

$$E_{eq} = \begin{bmatrix} E_0 \\ E_1 \\ \vdots \\ E_{N_{eq}-1} \end{bmatrix}$$

(20)

where $N_{eq}$ is the number of different parity check polynomials used for decoding.

C. Initial state estimation

The overall decoding operation is modeled as a function $\text{dec}(\cdot)$ that produces a status indicator $I_c$ and the estimated initial state of sequence $z$ ($\hat{A}_z$):

$$\{I_c, \hat{A}_z\} = \text{dec}(R(0), R(1), \ldots, R(M-1))$$

(21)

$I_c$ is the indication function of the decoder. It outputs a 1 if the decoder finds a valid codeword:

$$I_c = \begin{cases} 1, \text{if all parity check equations are satisfied} \\ 0, \text{otherwise} \end{cases}$$

(22)

If the decoding is successful ($I_c = 1$), the soft decision output of the decoder is converted into a binary value (0 or 1) with a hard decision rule. Then, according to the Fibonacci representation of Fig. 1, the first $2r$ bits of the codeword represent the state of the shift registers of sequence $z$ at initialization.

There exists a transposition matrix $T_{FG}$ between the state of sequences $z$ and $x$ and $y$. It is defined by (27) in the Appendix. However, it is valid only with the Galois representation. It is thus required to first defined the transposition matrix $T_{FG}$ between a Fibonacci and a Galois representation. It is defined by (24) in the Appendix. Eventually, the initial state of sequences $x$ and $y$ with the Fibonacci representation are obtained as follows:

$$\hat{A}_x = T_{FG,x}^{-1} \left[ 1 \ 0 \right] T_{GD,x}^{-1} T_{FG,x} \hat{A}_z$$

$$\hat{A}_y = T_{FG,y}^{-1} \left[ 0 \ 1 \right] T_{GD,y}^{-1} T_{FG,y} \hat{A}_z$$

(23)

IV. SIMULATION RESULTS

The performance of the algorithm is measured by the probability of missed detection $P_m$, defined as follows:

$$P_m = P( I_c = 1 \text{ and } \hat{A}_x = A_x \text{ and } \hat{A}_y = A_y)$$

$$P_m = 1 - P_d$$

The decoder stops when either all the parity check equations are satisfied or the maximum number of iteration $N_{iter} = 60$ is reached. The Gold sequence used for the evaluation are noted with the octal representation of the generator polynomials of sequence $x$ and $y$ (e.g.: 2157-3515). The parity check equations used for decoding are listed in tables III and IV. We have used the following notation: $c_i(x) = x^t + x^{t-1} + \cdots + x^1 + 1$ for $t = 5$ and $c_i(x) = x^t + x^{t-1} + \cdots + x^1 + 1$ for $t = 4$. For $t = 10$, the parity check equation of weight $t = 3$ is given by (18):

$$c(x) = x^6 + x^3 + 1$$

Fig. 3 shows the probability of missed detection as a function of the number of parity check polynomials used for decoding for $r = 10$. One observed a large improvement from $N_{eq} = 2$ to 4, then the gain decreases. The overall gain from $N_{eq} = 2$ to 10 is about 4.5 dB at $P_m = 0.01$. For $N_{eq} > 8$ the complexity of the decoder increases with the number of parity checks and there will be a trade-off between complexity and performance. We also plot the probability of missed detection measured for the Gold sequence (2011 - 3515) with $N_{eq} = 10$. One can observe that performances are identical. It means that from the decoding perspective, all Gold sequences of the same degree $r$ can be decoded with the same probability of detection. Fig. 4 shows the probability of missed detection measured with
the Gold sequence (4005,4445) of degree $r = 11$. We also observe a gain when $N_{eq}$ increases. Compared to Fig. 3, one can observe a loss of more than 2 dB for $N_{eq} = 10$. This is due to the weight of the parity check polynomials $t = 5$, which degrades the performance.

<table>
<thead>
<tr>
<th>TABLE III</th>
<th>Parity check equations for sequence (2157,3515).</th>
</tr>
</thead>
<tbody>
<tr>
<td>sequence (2157,3515)</td>
<td>sequence (2011,3515)</td>
</tr>
<tr>
<td>$c_1$</td>
<td>$117$</td>
</tr>
<tr>
<td>$c_2$</td>
<td>$185$</td>
</tr>
<tr>
<td>$c_3$</td>
<td>$105$</td>
</tr>
<tr>
<td>$c_4$</td>
<td>$222$</td>
</tr>
<tr>
<td>$c_5$</td>
<td>$280$</td>
</tr>
<tr>
<td>$c_6$</td>
<td>$238$</td>
</tr>
<tr>
<td>$c_7$</td>
<td>$271$</td>
</tr>
<tr>
<td>$c_8$</td>
<td>$293$</td>
</tr>
<tr>
<td>$c_9$</td>
<td>$307$</td>
</tr>
<tr>
<td>$c_{10}$</td>
<td>$314$</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>TABLE IV</th>
<th>Parity check equations for sequence (4005,4445).</th>
</tr>
</thead>
<tbody>
<tr>
<td>$m$</td>
<td>$i$</td>
</tr>
<tr>
<td>$c_1$</td>
<td>$114$</td>
</tr>
<tr>
<td>$c_2$</td>
<td>$116$</td>
</tr>
<tr>
<td>$c_3$</td>
<td>$227$</td>
</tr>
<tr>
<td>$c_4$</td>
<td>$195$</td>
</tr>
<tr>
<td>$c_5$</td>
<td>$131$</td>
</tr>
<tr>
<td>$c_6$</td>
<td>$181$</td>
</tr>
<tr>
<td>$c_7$</td>
<td>$150$</td>
</tr>
<tr>
<td>$c_8$</td>
<td>$155$</td>
</tr>
<tr>
<td>$c_9$</td>
<td>$158$</td>
</tr>
<tr>
<td>$c_{10}$</td>
<td>$183$</td>
</tr>
</tbody>
</table>

![Fig. 3. Probability of missed detection - $r = 10$](image1)

**V. Conclusion**

In this paper we have investigated the iterative decoding of Gold sequences. This allows to perform a blind estimation of the initial state of the sequence if the generator polynomials are known. We detailed the coding properties of Gold codes and gave the number of parity check equations of weight $t = 5$, which degrades the performance.

![Fig. 4. Probability of missed detection - $r = 11$](image2)

**Appendix**

**A. Conversion between Galois and Fibonacci generator initial states**

Let $a_{Gal}(k)$ and $a_{Fib}(k)$ be the state of the $k^{th}$ register with the Galois and Fibonacci’s representations. They are related as follows [17, page 106]:

$$a_{Gal}(k) = \sum_{i=0}^{k} a_{Fib}(i) g(k - i) \quad k = 0, \ldots, r - 1$$

This can also be written with a matrix-vector notation. Let $A_{Gal}$ and $A_{Fib}$ be the vectors containing the state of the shift registers according to the Galois and Fibonacci representations. They are related by a transposition matrix $T_{FG}: A_{Gal} = T_{FG} A_{Fib}$. The elements of matrix $T_{FG}$ are defined by:

$$T_{FG}(i,j) = \begin{cases} g(i - j) & \text{if } j \leq i \\ 0 & \text{if } j > i \end{cases}$$

On the other hand, the transposition from the Galois to the Fibonacci representation is obtained by the inversion of matrix $T_{FG}$:

$$T_{GF} = T_{FG}^{-1}$$

**B. Initial state of a Gold sequence**

In this section, the relation existing between the initial state of a Gold sequence $x$ and its preferred pair of m-sequences $x$ and $y$ is established. This relies on the property of LFSR sequence with the Galois representation. This is why sequences are designated with a subscript ‘Gal’ in the remaining part of this section.

Let $SR_{Gal}(k)$ be the state of the $k^{th}$ shift-register of sequence $z_{Gal}$ with the Galois representation and $K_{Gal}(x) = \begin{cases} g(i - j) & \text{if } j \leq i \\ 0 & \text{if } j > i \end{cases}$
Let define \( \sum_{k=0}^{2r-1} SR_{Gal}(k)x^k \) be its polynomial representation. Likewise, \( K_{Gal}(x) = \sum_{k=0}^{r-1} SR_{Gal}(k)x^k \) and \( K_{Gal}(x) = \sum_{k=0}^{r-1} SR_{Gal}(k)x^k \) are the equivalent representations for sequence \( x_{Gal} \) and \( y_{Gal} \).

Let also define \( x_{Gal}(x) = x_{Gal}(0) + x_{Gal}(1) + \cdots + x_{Gal}(N-1)x^{N-1} \) as the polynomial representation of sequence \( x_{Gal} \). Likewise, \( y_{Gal}(x) \) is the polynomial representation of sequence \( y_{Gal} \). With the Galois representation, the sequence polynomial is the result of the following division [17]:

\[
x_{Gal}(x) = \frac{K_{Gal}(x)}{g_x(x)}
\]

Since \( z_{Gal}(k) = x_{Gal}(0) + y_{Gal}(k) \), we have:

\[
z_{Gal}(x) = \frac{K_{Gal}(x)}{g_y(x)} + \frac{K_{Gal}(x)}{g_x(x)} = \frac{g_x(x)K_{Gal}(x) + g_y(x)K_{Gal}(x)}{g_x(x)}\]

This means that sequence \( z_{Gal} \) can be generated with polynomial \( g_x(x) = g_x(x)g_y(x) \), and the initial state is related to the ones of sequences \( x_{Gal} \) and \( y_{Gal} \) by:

\[
K_{z_{Gal}}(x) = g_x(x)K_{y_{Gal}}(x) + g_y(x)K_{x_{Gal}}(x)
\]

Let define \( A_{z_{Gal}} = (SR_{x_{Gal}}(0), \ldots, SR_{x_{Gal}}(2r-1))^T \), \( A_{z_{Gal}} = (SR_{x_{Gal}}(0), \ldots, SR_{x_{Gal}}(2r-1))^T \) and \( A_{z_{Gal}} = (SR_{y_{Gal}}(0), \ldots, SR_{y_{Gal}}(r-1))^T \), the initial state of sequences \( z_{Gal} \), \( x_{Gal} \) and \( y_{Gal} \).

Rewriting (25), \( A_{z_{Gal}} \), \( A_{x_{Gal}} \) and \( A_{y_{Gal}} \) are linked by:

\[
A_{z_{Gal}} = T_{Gold}(A_{x_{Gal}}^T A_{y_{Gal}}^T)^T
\]

\( T_{Gold} \) is the state transposition matrix defined by:

\[
T_{Gold} = [T_y \ T_x]
\]

with \( T_x \) (resp. \( T_y \)) being defined as follows:

\[
T_x = \begin{bmatrix}
ge_x(0) & 0 & \cdots & 0 \\
\vdots & \vdots & \ddots & \vdots \\
0 & g_x(r) & \cdots & g_x(0) \\
\vdots & \vdots & \ddots & \vdots \\
0 & 0 & \cdots & g_x(r)
\end{bmatrix}
\]

REFERENCES