ΕΘΝΙΚΟ ΜΕΤΣΟΒΙΟ ΠΟΛΥΤΕΧΝΕΙΟ

ΣΧΟΛΗ ΗΛΕΚΤΡΟΛΟΓΩΝ ΜΗΧΑΝΙΚΩΝ ΚΑΙ ΜΗΧΑΝΙΚΩΝ ΥΠΟΛΟΓΙΣΤΩΝ

ΨΗΦΙΑΚΕΣ ΕΠΙΚΟΙΝΩΝΙΕΣ ΙΙ

3η ΣΕΙΡΑ ΑΣΚΗΣΕΩΝ

Ειρήνη Δόντη

ΑΜ: 03119839

9ο εξάμηνο

Αθήνα 2023-2024

1.

Για n = 31 και t = 3, ισχύει ότι: $n = 2^m - 1$ ή $m = 5$ και $n - k = 2t = 6$ ή $31 - k = 6$ ή $k = 25$. Χρειαζόμαστε το γενετήριο πολυώνυμο του κώδικα το οποίο προκύπτει από τη σχέση $g(x) = (x + a)(x + a^2)(x + a^3)(x + a^4)(x + a^5)(x + a^6) = \ldots = x^6 + a^{10}x^5 + a^9x^4 + a^{24}x^3 + a^{16}x^2 + a^{24}x + a^{21}$. Στη συνέχεια, για κάθε μήνυμα m που θέλουμε να στείλουμε, χρειάζεται να μετατρέψουμε τα bits του σε στοιχεία του σώματος galois $GF(2^5)$, έτσι ώστε κάθε 5 bits αντιπροσωπεύουν ένα στοιχείο του σώματος. Αναπαριστούμε το μήνυμα αυτό ως πολυώνυμο με συντελεστές των όρων του τα στοιχεία αυτά ξεκινώντας από τον μεγιστοβάθμιο όρο x^{24} και συνεχίζοντας μέχρι τον σταθερό όρο του πολυώνυμου. Πολλαπλασιάζουμε το πολυώνυμο m(x) που προέκυψε με $x^{n-k} = x^6$ και αυτό που προκύπτει το διαιρούμε με το γενετήριο πολυώνυμο g(x). Το άθροισμα του υπολοίπου b(x) και του γινομένου $x^6m(x)$ μας δίνει τελικά το πολυώνυμο u(x). Για την εύρεση του γενετηρίου πολυωνύμου, οι πράξεις πραγματοποιήθηκαν χειρωνακτικά. Για την επαλήθευση των παραπάνω πράξεων, παρατίθεται κώδικας στο MATLAB:

```
%R-S encoding
clear all;
close all;
clc;
n=31; %Αρχικοποιούμε τα χαρακτηριστικά του κώδικα
t=3;
m=log2(n+1);
k=n-2*t;

%%%finding generator polynomial%%%
gen_x=gf([1],m); %παρακάτω βρίσκουμε το γενετήριο πολυώνυμο του κώδικα
a=gf(2,m);
for i=1:2*t
p=gf([1,a^i],m);
gen_x=conv(gen_x,p); %(x+a)(x+a^2)(x+a^3)....
end
mes = randi([0 1],1,k*m); %μήνυμα μήκους 125 bits
gfelem=[]; %θα μετατρέψουμε τα bits του μηνύματος σε στοιχεία του αντίστοιχου σώματος
for i=1:m:k*m
    elem=0;
    for j = 0:m-1
        elem=elem+mes(i+j)*2^(m-1-j);
    end
    gfelem = [gfelem, gf(elem, m)];
end
gfelem=flip(gfelem);
np=gf([1,zeros(1,2*t)],m);
np=conv(np,gfelem);
[q, b] = deconv(np,gen_x);
u_x = np + b; % το κωδικοποιημένο μήνυμα
disp(u_x); %το κωδικοποιημένο μήνυμα
disp('G(x) = ');
disp(gen_x); %δεκαδική αναπαράσταση του γενετήριου πολυωνύμου
disp(log(gen_x)); %αναπαράσταση του γενετήριου πολυωνύμου με εκθέτες του B
disp(mes);
disp(u_x); %δεκαδική αναπαράσταση του κωδικού πολυωνύμου
```

```
for i=1:length(u_x) %αναπαράσταση του κωδικού πολυωνύμου με εκθέτες του α
  if u_x(i)==0
    fprintf('0*x^%d',n-i);
  else
    fprintf('a^%d*x^%d',log(u_x(i)),n-i);
  end
  if rem(i, 9)==0 && i<n %προσαρμογή στο command window του MATLAB
      fprintf('+\n+');
  elseif i<n
    fprintf('+')
  else
    fprintf('\n');
  end
end
```

i) Αποκωδικοποιούμε τον αλγόριθμο ΜΚΔ του Ευκλείδη

Για την αποκωδικοποίηση μηνύματος επιλέγουμε την κωδικολέξη:

$u(x) = a^{18}x^{30} + a^{20}x^{29} + a^{30}x^{28} + a^{3}x^{27} + a^{29}x^{26} + a^{8}x^{25} + a^{27}x^{24} + a^{3}x^{23} + a^{2}x^{22} + a^{5}x^{21} + a^{10}x^{20} + a^{18}x^{19} + a^{26}x^{18} + a^{26}x^{17} + a^{7}x^{16} + a^{18}x^{15} + a^{26}x^{14} + a^{20}x^{13} + a^{25}x^{12} + a^{24}x^{11} + a^{17}x^{10} + a^{7}x^{9} + a^{17}x^{8} + a^{18}x^{7} + a^{2}x^{6} + a^{30}x^{5} + a^{3}x^{4} + a^{24}x^{3} + x^{2} + a^{27}x^{1} + a^{2}$

Υποθέτουμε λάθος $e(x) = a^{4}x^{24} + x^{12} + a^{19}x^{10}$. Οπότε, τελικά προκύπτει το λαμβανόμενο από το δέκτη πολυώνυμο:

$r(x) = a^{18}x^{30} + a^{20}x^{29} + a^{30}x^{28} + a^{3}x^{27} + a^{29}x^{26} + a^{8}x^{25} + a^{16}x^{24} + a^{3}x^{23} + a^{2}x^{22} + a^{5}x^{21} + a^{10}x^{20} + a^{18}x^{19} + a^{26}x^{18} + a^{26}x^{17} + a^{7}x^{16} + a^{18}x^{15} + a^{26}x^{14} + a^{20}x^{13} + a^{21}x^{12} + a^{24}x^{11} + a^{22}x^{10} + a^{7}x^{9} + a^{17}x^{8} + a^{18}x^{7} + a^{2}x^{6} + a^{30}x^{5} + a^{3}x^{4} + a^{24}x^{3} + x^{2} + a^{27}x^{1} + a^{2}$

Αρχικά, υπολογίζουμε τα n-k = 6 σύνδρομα:

$S1 = r(a) = e(a) = a^{10}$

$S2 = r(a^{2}) = e(a^{2}) = a^{27}$

$S3 = r(a^{3}) = e(a^{3}) = a^{16}$

$S4 = r(a^{4}) = e(a^{4}) = a^{10}$

$S5 = r(a^{5}) = e(a^{5}) = a^{13}$

$S6 = r(a^{6}) = e(a^{6}) = a^{13}$

Οπότε, προκύπτει ότι $S(x) = a^{13}x^{5} + a^{13}x^{4} + a^{10}x^{3} + a^{16}x^{2} + a^{27}x^{1} + a^{10}$

Επιπλέον, αναζητούμε τα πολυώνυμα εντοπισμού και υπολογισμού σφαλμάτων σ(x), ω(x) με τη βοήθεια του αλγορίθμου Ευκλείδη για τον ΜΚΔ. Διαιρούμε τα πολυώνυμα $\frac{u_{i-2}(x)}{u_{i-1}(x)}$ και προκύπτει το υπόλοιπο $u_i(x)$ και το πηλίκο $\varphi_i(x)$. Συμπληρώνουμε τα $s_i(x), t_i(x)$ σύμφωνα με τις σχέσεις $s_i(x) = s_{i-2}(x) - \varphi_i \cdot s_{i-1}(x)$ και $t_i(x) = t_{i-2}(x) - \varphi_i \cdot t_{i-1}(x)$. Οπότε, σ(x) = $\lambda \cdot t_j(x)$ και ω(x) = $\lambda \cdot u_j(x)$, όπου j το βήμα του αλγορίθμου για το οποίο ισχύει για πρώτη φορά βαθμός($u_i(x)$) < t = 3.

Βήμα i	$u_i(x)$	$\varphi_i(x)$
-1	$x^{2t} = x^6$	
0	$a^{13}x^5 + a^{13}x^4 + a^{10}x^3 + a^{16}x^2 + a^{27}x + a^{10}$	
1	$a^{26}x^4 + a^{24}x^3 + a^{22}x^2 + a^{27}x + a^{28}$	$a^{18}x + a^{18}$
2	$a^{28}x^3 + a^3x^2 + a^{11}x + a^{30}$	$a^{18}x + a^{21}$
3	$a^{29}x^2 + a^{28}x + a^{29}$	$a^{29}x + a^{16}$

Βήμα i	$s_i(x)$	$t_i(x)$
-1	1	0
0	0	1
1	1	$a^{18}x + a^{18}$
2	$a^{18}x + a^{21}$	$a^5x^2 + a^3x + a^{20}$
3	$a^{16}x^2 + a^{12}x + a^{27}$	$a^3x^3 + a^9x^2 + a^{19}x + a^{19}$

Ο αλγόριθμος τερματίζει μόλις ο βαθμός του πολυωνύμου $u_i(x)$ γίνει μικρότερος από t =3 και γνωρίζοντας πως ο σταθερός όρος του σ(x) είναι ίσος με 1 υπολογίζουμε το λ: σ(x) = $\lambda \cdot t_3(x) = \lambda \cdot (a^3x^3 + a^9x^2 + a^{19}x + a^{19})$ ή $\lambda(a^{19}) = 1$ ή $\lambda = a^{12}$.

Επομένως, σ(x) = $a^{15}x^3 + a^{21}x^2 + x + 1$ και ω(x) = $\lambda \cdot u_3(x) = a^{10}x^2 + a^9x + a^{10}$.

Επειδή σ(x) είναι 3ου βαθμού, θα έχει 3 ρίζες. Αναζητούμε αυτές τις ρίζες δοκιμάζοντας τις τιμές x = 1, a, a^2, …, a^{30}

$\sigma(1) = a^{15} + a^{21} + 1 + 1 = a^{11}$

$\sigma(a) = a^{18} + a^{23} + \alpha + 1 = a^{23}$

$\sigma(a^2) = a^2$

$\sigma(a^3) = a^{13}$

$\sigma(a^4) = a^{17}$

$\sigma(a^5) = a^{26}$

$\sigma(a^6) = a^{27}$

$\sigma(a^7) = 0$

$\sigma(a^8) = a^{27}$

$\sigma(a^9) = a^{10}$

$\sigma(a^{10}) = a^{13}$

$\sigma(a^{11}) = a^{16}$

$\sigma(a^{12}) = a^{24}$

$\sigma(a^{13}) = a^{29}$

$\sigma(a^{14}) = a^3$

$\sigma(a^{15}) = a^{16}$

$\sigma(a^{16}) = a^8$

$\sigma(a^{17}) = a^{13}$

$\sigma(a^{18}) = 1$

$\sigma(a^{19}) = 0$

$\sigma(a^{20}) = a^{18}$

$\sigma(a^{21}) = 0$

Γνωρίζουμε πως υπάρχουν μόνο τρεις ρίζες, οι a^7, a^{19}, a^{21} και συνεπώς, για να βρούμε τις θέσεις των λαθών, αρκεί να υπολογίσουμε τα αντίστροφα των

ριζών. Οπότε, $\beta 1 = (a^7)^{-1} = a^{24}$, $\beta 2 = (a^{19})^{-1} = a^{12}$, $\beta 3 = (a^{21})^{-1} = a^{10}$. Άρα τα σφάλματα εντοπίστηκαν στους συντελεστές των όρων x^{24}, x^{12}, x^{10} του ληφθέντος πολυώνυμου r(x). Υπολογίζουμε τα λάθη με χρήση του τύπου $e_{jl} = \frac{\omega(\beta l^{-1})}{\sigma\prime(\beta l^{-1})}$ χρειάζεται να υπολογίσουμε την πρώτη παράγωγο του πολυωνύμου εντοπισμού σφαλμάτων: $\sigma'(x) = \sum_{j=1}^{3} \beta_j \prod_{k=1,\ k\neq j}^{3} (1 + \beta_k x)$

$= \beta_1(1 + \beta_2 x)(1 + \beta_3 x) + \beta_2(1 + \beta_1 x)(1 + \beta_3 x) + \beta_3(1 + \beta_1 x)(1 + \beta_2 x) = \alpha^{24}(1 + \alpha^{12}x)(1 + \alpha^{10}x) + \alpha^{12}(1 + \alpha^{24}x)(1 + \alpha^{10}x) + \alpha^{10}(1 + \alpha^{24}x)(1 + \alpha^{12}x) = \cdots = \alpha^{15}x^2 + (\alpha^8 + \alpha^4 + \alpha^{14})x + (\alpha^{24} + \alpha^{12} + \alpha^{10}) = \alpha^{15}x^2 + 1$

Οπότε, τελικά: $e_{24} = \frac{\omega((\alpha^{24})^{-1})}{\sigma\prime((\alpha^{24})^{-1})} = \frac{\omega(\alpha^7)}{\sigma\prime(\alpha^7)} = \frac{\alpha^{10}\alpha^{14} + \alpha^9\alpha^7 + \alpha^{10}}{\alpha^{15}\alpha^{14} + 1} = \frac{\alpha^7}{\alpha^3} = \alpha^4$

$e_{12} = \frac{\omega((\alpha^{12})^{-1})}{\sigma\prime((\alpha^{12})^{-1})} = \frac{\omega(\alpha^{19})}{\sigma\prime(\alpha^{19})} = \frac{\alpha^{10}\alpha^7 + \alpha^9\alpha^{19} + \alpha^{10}}{\alpha^{15}\alpha^7 + 1} = \frac{\alpha^7}{\alpha^7} = 1$

$e_{10} = \frac{\omega((\alpha^{10})^{-1})}{\sigma\prime((\alpha^{10})^{-1})} = \frac{\omega(\alpha^{21})}{\sigma\prime(\alpha^{21})} = \frac{\alpha^{10}\alpha^{11} + \alpha^9\alpha^{21} + \alpha^{10}}{\alpha^{15}\alpha^{11} + 1} = \frac{\alpha^{16}\alpha^3}{\alpha^{28}\alpha^3} = \alpha^{19}$

Πράγματι, είναι οι τιμές των συντελεστών των όρων του διανύσματος λάθους που είχαμε εισάγει στο κωδικό πολυώνυμο.

ii) <u>Αποκωδικοποίηση με αυτοπαλινδρόμηση στο MATLAB</u>

Τροποποιούμε τον κώδικα 9.24, αλλάζοντας τις παραμέτρους και τη σειρά γραφής των bit ώστε το αριστερότερο να είναι το MSB. Οπότε, προκύπτει ο παρακάτω κώδικας:

```
%%%Reed-Solomon decoding using Autoregression%%%
%RS decoding
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%A.Confirm that p(x) is a primpoly on the
%GF(2^5). Find all the elements of this field and fill out the table
%with all the representations of the elements
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
clear all; close all;
t=3;
m=5;
n=2^m-1;
k=n-2*t;
```

```
d=primpoly(m);%primitive polynomial of gf(2^m) with m=5 D^5+D^2+1=37
a=gf(2,m,d); %the first non-zero, non-one element of the field
%b will contain the non-zero elements of gf(2^5)
b=gf([],m,d);
b(n)=1;
for i=n-1:-1:1
    b(i)=a^i;
end
% All elements of b are discrete, thus p(x) is primitive
%
%B.Find the RS encoder on that field capable of correcting up to 3
%erroneous elements. Then suppose that an all-zero 60-bit block
%(encoded with this encoder) is transmitted. At the receiver,
%it is found 1(instead of 0) at bit positions 31, 95, 103, 104
%(counted from the left-most bit position 0, or MSB), that is,
%the following bit stream is received:
rb=zeros(1,m*n); %an all-zero codeword
e_pos=[31, 95, 103, 104]; %bit-error positions within the
                                %(binary) codeword, from MSB
for i=1:length(e_pos)
    rb(e_pos(i))=~rb(e_pos(i)); %We invert the bits at erroneous bit positions to get
                                %the received message
end
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%Combining the bits in  five (elements of the GF(2^5)), we get:
%r(x)=a^4*X^24+a^0*X^12+a^19*X^10
rm=reshape(rb,m,length(rb)/m);
r=gf(bi2de(rm','left-msb'),m)';
%{
gfr=[];
for i=1:m:n*m %μπορεί να γίνει και με χρήση της bi2de
    ele=0;
    for j=0:m-1
        ele=ele+rb(i+j)*2^(m-1-j);
    end
    gfr=[gfr, gf(ele,m)];
end
%gfr=flip(gfr);
%}
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%Syndrome Calculation
s=gf([],m,d);
for i=1:2*t
    s(i)=r(n);
    for j=1:n-1
        s(i)=s(i)+(a^i)^j*r(n-j);
    end
end
S=gf([],m,d);
cons_s=gf([],m,d);
for i=1:t
    for j=0:t-1
        S=[S s(i+j)];
    end
    cons_s(i)=-s(t+i);
end
S=reshape(S,t,length(S)/t)';
if det(S)~=0
    sigma=S^-1*cons_s';
    %Find roots of sigma
    s_roots_exp=[];   %vector of sigma roots exponents(of a)
    for i=1:n-1
        ai=a^i;
        %ai=a^(n-i); %
        sr=gf(1,m,d);
        for j=1:t
            sr=sr+sigma(j)*ai^(t-j+1);
        end
        if (sr==0)
            s_roots_exp=[s_roots_exp i];
        end
    end
    b=(a.^s_roots_exp).^-1;   %inverses of sigma roots
    er_pos=n-log(b);   %error positions from ms digit %
```

```
    B=gf([],m,d);
    cons_b=gf([],m,d);
    for i=1:t
        for j=1:t
            B=[B b(j)^i];
        end
        cons_b(i)=s(i);
    end
    B=reshape(B,t,length(B)/t)';
    er=(B^-1*cons_b')';
    u=r;
    for i=1:length(er)
        u(er_pos(i))=r(er_pos(i))+er(i);
    end
    r
    u
else
    disp('Determinant equal zero');
end
%{
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%Use high-level MATLAB functions for RS encoding & decoding
% ------------------------------------------------------------
msg=gf(zeros(1,k),m,d); % a message with k zero-value elements
r=rsenc(msg,n,k); % the rs-encoded message (n zeros)
r(3)=3;
r(9)=8;
r(12)=11; %the specified errors
[u,cnumerr]=rsdec(r,n,k); % u: the decoded data message
                          % cnumerr: the number of errors found
%}
```

Δοκιμάζουμε στην κωδικολέξη, με όλα τα στοιχεία μηδενικά, να προσθέσουμε το διάνυσμα λάθους $e(x) = \alpha^4x^{24} + x^{12} + \alpha^{19}x^{10}$ που αντιστοιχεί σε αντεστραμμένα τα bits 31, 95, 103, 104 μετρώντας από το αριστερότερο MSB. Με χρήση του παραπάνω κώδικα που πραγματοποιεί τη διαδικασία της αυτοπαλινδρόμησης, βρίσκουμε το πολυώνυμο εντοπισμού σφαλμάτων σ(x) και από τα αντίστροφα των ριζών που βρίσκουμε τις τιμές των σφαλμάτων αυτών. Έπειτα, προσθέτουμε στο ληφθέν πολυώνυμο τα σφάλματα στις σωστές θέσεις και όπως φαίνεται και στο πρόγραμμα, προκύπτει η αρχική κωδικολέξη με μόνο μηδενικά στοιχεία.

2.

Παρατηρούμε ότι για m = 8 κάθε σύμβολο του σώματος galois $GF(2^8)$ αποτελείται από 8 bits, δηλαδή 1 byte. Άρα τα k = 187 bytes είναι 187 σύμβολα, το ίδιο για τα 20 parity bytes. Εφόσον έχουμε 20 bytes ισοτιμίας, γνωρίζουμε ότι 2t = 20 και συνεπώς t = 10. Επίσης, n – k = 2t ή n = 20 + 187

= 207 bytes. Ο συγκεκριμένος κώδικας Reed-Solomon του προτύπου ATSC δεν είναι συμβατικής μορφής, δηλαδή $n = 2^m - 1 = 2^8 - 1 = 255 \neq 207$. Για τον υπολογισμό του γενετήριου πολυωνύμου του κωδικοποιητή που ορίζει το πρότυπο χρησιμοποιήθηκε ο παρακάτω κώδικας MATLAB:

```
%%RS_encoding_ATSC
clear all;
close all;
clc;
%Αρχικοποιούμε τα χαρακτηριστικά του κώδικα
m=8;
k=187;
t=10;
n=k+2*t;
%%%finding generator polynomial%%%
d=primpoly(8);
gen_x=gf([1],m,d); %παρακάτω βρίσκουμε το γενετήριο πολυώνυμο του κώδικα
a=gf(2,m,d);
for i=0:2*t-1
    p=gf([1,a^i],m,d);
gen_x=conv(gen_x,p);%(x+1)(x+a)(x+a^2)(x+a^3)...(x+a^(2*t-1))
end
gen_x
for i=1:2*t+1
    fprintf('a^{%d}*x^{%d}',log(gen_x(i)),2*t+1-i);
    if i<2*t+1
        fprintf(' + ');
    else
        fprintf('\n');
    end
end
```

Η έξοδος του προγράμματος είναι σε συμφωνία με τον πίνακα προτύπου. Για την εύρεση του γενετήριου πολυωνύμου, το πρότυπο ορίζει να σχηματιστεί ώστε οι ρίζες του να είναι 2t διαδοχικές δυνάμεις του α ξεκινώντας από τη μηδενική. Τροποποιούμε τον παραπάνω κώδικα, ώστε να ξεκινά από την πρώτη δύναμη του α, θα προκύψει ο εξής κώδικας:

```
%%RS_encoding_ATSC
clear all;
close all;
clc;
%Αρχικοποιούμε τα χαρακτηριστικά του κώδικα
m=8;
k=187;
t=10;
n=k+2*t;
%%%finding generator polynomial%%%
d=primpoly(8);
gen_x=gf([1],m,d); %παρακάτω βρίσκουμε το γενετήριο πολυώνυμο του κώδικα
a=gf(2,m,d);
for i=1:2*t %εκθέτη 1 ως 2*t
    p=gf([1,a^i],m,d);
gen_x=conv(gen_x,p);%(x+1)(x+a)(x+a^2)(x+a^3)...(x+a^(2*t-1))
end
gen_x
for i=1:2*t+1
    fprintf('a^{%d}*x^{%d}',log(gen_x(i)),2*t+1-i);
```

```
    if i<2*t+1
        fprintf(' + ');
    else
        fprintf('\n');
    end
end
```

Η έξοδος του προγράμματος δίνει το εξής γενετήριο πολυώνυμο:

$g(x) = x^{20} + a^{18}x^{19} + a^{62}x^{18} + a^{82}x^{17} + a^{54}x^{16} + a^{66}x^{15} + a^{169}x^{14} +$
$a^{33}x^{13} + a^{195}x^{12} + a^{211}x^{11} + a^{190}x^{10} + a^{232}x^{9} + a^{237}x^{8} + a^{96}x^{7} + a^{253}x^{6} + a^{171}x^{5} +$
$a^{180}x^{4} + a^{229}x^{3} + a^{230}x^{2} + a^{207}x^{1} + a^{210}$

Παρατηρούμε ότι τα δύο πολυώνυμα δεν είναι ίσα.

3.

Συνδυάζουμε τους παραπάνω κώδικες από τα ερωτήματα 1i) και 2):

```
%%RS_encoding_ATSC
clear all;
close all;
clc;
%Αρχικοποιούμε τα χαρακτηριστικά του κώδικα
m=8;
k=187;
t=10;
n=k+2*t;
%%%finding generator polynomial%%%
d=primpoly(8);
gen_x=gf([1],m,d); %παρακάτω βρίσκουμε το γενετήριο πολυώνυμο του κώδικα
a=gf(2,m,d);
for i=0:2*t-1
    p=gf([1,a^i],m,d);
gen_x=conv(gen_x,p);%(x+1)(x+a)(x+a^2)(x+a^3)...(x+a^(2*t-1))
end
gen_x
for i=1:2*t+1
    fprintf('a^{%d}*x^{%d}',log(gen_x(i)),2*t+1-i);
    if i<2*t+1
        fprintf(' + ');
    else
        fprintf('\n');
    end
end
mes=randi([0 1],1,k*m); %μήνυμα μήκους 125 bits
mes=zeros(1,k*m);
gfelem=[]; %θα μετατρέψουμε τα bits του μηνύματος σε στοιχεία του αντίστοιχου σώματος
for i=1:m:k*m %μπορεί να γίνει και με χρήση της bi2de
    elem=0;
    for j=0:m-1
        elem=elem+mes(i+j)*2^(m-1-j);
    end
    gfelem=[gfelem, gf(elem,m)];
end
gfelem=flip(gfelem);
np=gf([1,zeros(1,2*t)],m);
np=conv(np,gfelem);
[q,b]=deconv(np,gen_x);
u_x=np+b;%το κωδικοποιημένο μήνυμα
disp(u_x); %το κωδικοποιημένο μήνυμα
disp(gen_x); %δεκαδική αναπαράσταση του γενετήριου πολυωνύμου
disp(log(gen_x)); %αναπαράσταση του γενετήριου πολυωνύμου με εκθέτες του a
disp(mes);
```

```
disp(u_x); %δεκαδική αναπαράσταση του κωδικού πολυωνύμου
for i=1:length(u_x)     %αναπαράσταση του κωδικού πολυωνύμου με εκθέτες του a
    if u_x(i)==0
        fprintf('0*x^%d',n-i);
    else
        fprintf('a^%d*x^%d',log(u_x(i)),n-i);
    end
    if rem(i,9)==0 && i<n
            fprintf(' +  \n + ');
    elseif i<n
        fprintf(' + ');

    else
        fprintf('\n');
    end
end
```

Το γενετήριο πολυώνυμο που προκύπτει είναι παρακάτω:

$g(x) = x^{20} + a^{17}x^{19} + a^{60}x^{18} + a^{79}x^{17} + a^{50}x^{16} + a^{61}x^{15} + a^{163}x^{14} +$
$a^{26}x^{13} + a^{187}x^{12} + a^{202}x^{11} + a^{180}x^{10} + a^{221}x^{9} + a^{225}x^{8} + a^{83}x^{7} + a^{239}x^{6} + a^{156}x^{5} +$
$a^{164}x^{4} + a^{212}x^{3} + a^{212}x^{2} + a^{188}x^{1} + a^{190}$

i) <u>Αποκωδικοποιούμε τον αλγόριθμο ΜΚΔ του Ευκλείδη</u>

Θεωρούμε κωδικολέξη με όλα τα στοιχεία μηδενικά και διάνυσμα λάθους το
$e(x) = a^{58}x^{193} + a^{45}x^{175} + a^{72}x^{170} + a^{240}x^{155} + a^{160}x^{114} + a^{12}x^{99} +$
$a^{11}x^{43} + a^{4}x^{36} + a^{236}x^{17} + a^{101}x^{10}$
Οπότε, προκύπτει το παρακάτω λαμβανόμενο πολυώνυμο:
$r(x) = a^{58}x^{193} + a^{45}x^{175} + a^{72}x^{170} + a^{240}x^{155} + a^{160}x^{114} + a^{12}x^{99} +$
$a^{11}x^{43} + a^{4}x^{36} + a^{236}x^{17} + a^{101}x^{10}$

Υπολογίζουμε τα 2t = 20 σύνδρομα:
$S1 = r(a) = e(a) = a^{110}$
$S2 = r(a^2) = e(a^2) = a^{60}$
$S3 = r(a^3) = e(a^3) = a^{163}$
$S4 = r(a^4) = e(a^4) = a^{190}$
$S5 = r(a^5) = e(a^5) = a^{41}$
$S6 = r(a^6) = e(a^6) = a^{238}$
$S7 = r(a^7) = e(a^7) = a^{195}$
$S8 = r(a^8) = e(a^8) = a^{126}$
$S9 = r(a^9) = e(a^9) = a^{246}$

$S10 = r(a^{10}) = e(a^{10}) = a^{12}$

$S11 = r(a^{11}) = e(a^{11}) = a^{12}$

$S12 = r(a^{12}) = e(a^{12}) = a^{77}$

$S13 = r(a^{13}) = e(a^{13}) = a^{133}$

$S14 = r(a^{14}) = e(a^{14}) = a^{55}$

$S15 = r(a^{15}) = e(a^{15}) = a^{86}$

$S16 = r(a^{16}) = e(a^{16}) = a^{204}$

$S17 = r(a^{17}) = e(a^{17}) = a^{57}$

$S18 = r(a^{18}) = e(a^{18}) = a^{185}$

$S19 = r(a^{19}) = e(a^{19}) = a^{95}$

$S20 = r(a^{20}) = e(a^{20}) = a^{16}$

Οπότε, προκύπτει ότι:

$S(x) = a^{16}x^{19} + a^{95}x^{18} + a^{185}x^{17} + a^{57}x^{16} + a^{204}x^{15} + a^{86}x^{14} + a^{55}x^{13} + a^{133}x^{12} + a^{77}x^{11} + a^{12}x^{10} + a^{12}x^{9} + a^{246}x^{8} + a^{126}x^{7} + a^{195}x^{6} + a^{238}x^{5} + a^{41}x^{4} + a^{190}x^{3} + a^{163}x^{2} + a^{60}x^{1} + a^{110}$

Επιπλέον, αναζητούμε τα πολυώνυμα εντοπισμού και υπολογισμού σφαλμάτων σ(x), ω(x) με τη βοήθεια του αλγορίθμου Ευκλείδη για τον ΜΚΔ. Διαιρούμε τα πολυώνυμα $\frac{u_{i-2}(x)}{u_{i-1}(x)}$ και προκύπτει το υπόλοιπο $u_i(x)$ και το πηλίκο $\varphi_i(x)$. Συμπληρώνουμε τα $s_i(x), t_i(x)$ σύμφωνα με τις σχέσεις $s_i(x) = s_{i-2}(x) - \varphi_i \cdot s_{i-1}(x)$ και $t_i(x) = t_{i-2}(x) - \varphi_i \cdot t_{i-1}(x)$. Οπότε, σ(x) = $\lambda \cdot t_j(x)$ και ω(x) = $\lambda \cdot u_j(x)$, όπου j το βήμα του αλγορίθμου για το οποίο ισχύει για πρώτη φορά βαθμός$(u_i(x))$ < t = 10.

Με τη βοήθεια κώδικα σε MATLAB, πραγματοποιούμε τον αλγόριθμο του Ευκλείδη όπως φαίνεται παρακάτω:

```
%%%RS_decoding_Euclid's GCD algorithm ATSC
clear all; close all; clc;
t=10;
m=8;
k=187;
n=k+2*t;
d=primpoly(m);%primitive polynomial of gf(2^m)
a=gf(2,m,d); %the first non-zero, non-one element of the field
%b will contain the non-zero elements of gf(2^8)
b=gf([],m,d);
b(n)=1;
```

```
for i=2^m-1:-1:1
    b(i)=a^i;
end

e_x=[];
er_pos=[193, 175, 170, 155, 114, 99, 43, 36, 17, 10]; %θέσεις λαθών όπου οι τιμές
αντιπροσωπεύουν
                                %τους εκθέτες των όρων όπου τα λάθη εμφανίζονται
er=[a^58, a^45, a^72, a^240, a^160, a^12, a^11, a^4, a^236, a^101]; %τιμές λαθών
%er_pos=[24 12 10];
%er=[a^4 a^0 a^19];
count=1;

for i=1:n
    if er_pos(count)==n-i
        e_x=[e_x, er(count)];
        if count<t
            count=count+1;
        end
    else
        e_x=[e_x,0];
    end
end
r_x=e_x; %Μόνο αν θεωρήσουμε τη μηδενική κωδικολέξη


%%%%Υπολογισμός συνδρόμων
S=gf([zeros(1,2*t)],m,d);
for i=1:2*t
    for j=1:n-1
        S(i)=S(i)+(a^(rem(i*(n-j),2^m-1)))*e_x(j);
    end

end
log(S) %εκθέτες των συνδρόμων σε εκθετική μορφή

%%%%%%%%%%Euclid's algorithm
%αρχικοποιούμε βήματα -1 και 0
u_iminus2=gf([1,zeros(1,2*t)],m,d);
u_iminus1=flip(S);
phi_i=gf([],m,d);
s_iminus2=gf([1],m,d);
s_iminus1=gf([0],m,d);
t_iminus2=gf([0],m,d);
t_iminus1=gf([1],m,d);
i=1;

while length(u_iminus1)>t %συνθήκη τερματισμού του αλγορίθμου

    [p,q]=deconv(u_iminus2,u_iminus1);
    u_iminus2=u_iminus1;
    u_iminus1=q;
    phi_i=p;

    temp=[];
    no_zeros=0; %αφαιρούμε τα μηδενικά από την αρχή του πολυωνύμου
    for j=1:length(u_iminus1)
        if u_iminus1(j)==0 && no_zeros==0
            continue
        else
            temp=[temp,u_iminus1(j)];
            no_zeros=1;
        end
    end
    u_iminus1=temp;
    temp=[];
    no_zeros=0;
    for j=1:length(phi_i)
        if phi_i(j)==0 && no_zeros==0
            continue
        else
            temp=[temp,phi_i(j)];
            no_zeros=1;
        end
```

```
    end
    phi_i=temp;
    fprintf('For step %d we get u_%d, phi_%d',i,i,i);
    u_iminus1 %u_i(x) for step i
    phi_i %ϕ_i(x) for step i

    while length(s_iminus2)-length(conv(phi_i,s_iminus1))~=0
        s_iminus2=[0,s_iminus2]; %απαιτείται για να είναι τα πολυώνυμα ίδιων
διαστάσεων
                                 %και να μπορεί το MATLAB να τα προσθέσει
    end
    mu=s_iminus2+conv(phi_i,s_iminus1);
    s_iminus2=s_iminus1;
    s_iminus1=mu;

    while length(t_iminus2)-length(conv(phi_i,t_iminus1))~=0
        t_iminus2=[0,t_iminus2]; %απαιτείται για να είναι τα πολυώνυμα ίδιων
διαστάσεων
                                 %και να μπορεί το MATLAB να τα προσθέσει
    end
    mu=t_iminus2+conv(phi_i,t_iminus1);
    t_iminus2=t_iminus1;
    t_iminus1=mu;



    fprintf('And we also get s_%d, t_%d',i,i);
    s_iminus1 %s_i(x) for step i
    t_iminus1 %t_i(x) for step i
    i=i+1;
end
lambda=1/t_iminus1(length(t_iminus1));
sig_x=t_iminus1*lambda;
o_x=u_iminus1*lambda;
log(sig_x)%οι εκθέτες των συντελεστών του πολυωνύμου εντοπισμού λαθών
log(o_x)%οι εκθέτες των συντελεστών του πολυωνύμου υπολογισμού λαθών
```

Οπότε, λαμβάνουμε τον παρακάτω πίνακα:

Βήμα i	$u_i(x)$
-1	$x^{2t} = x^{20}$
0	$S(x) = a^{16}x^{19} + a^{95}x^{18} + a^{185}x^{17} + a^{57}x^{16} + a^{204}x^{15} + a^{86}x^{14} + a^{55}x^{13} + a^{133}x^{12} + a^{77}x^{11} + a^{12}x^{10} + a^{12}x^{9} + a^{246}x^{8} + a^{126}x^{7} + a^{195}x^{6} + a^{238}x^{5} + a^{41}x^{4} + a^{190}x^{3} + a^{163}x^{2} + a^{60}x^{1} + a^{110}$
1	$a^{148}x^{18} + a^{246}x^{17} + a^{137}x^{16} + a^{119}x^{15} + a^{165}x^{14} + a^{142}x^{13} + a^{205}x^{12} + a^{131}x^{11} + a^{170}x^{10} + a^{234}x^{9} + a^{185}x^{8} + a^{200}x^{7} + a^{192}x^{6} + a^{35}x^{5} + a^{168}x^{4} + a^{186}x^{3} + a^{207}x^{2} + a^{20}x^{1} + a^{173}$
2	$a^{43}x^{17} + a^{245}x^{16} + a^{145}x^{15} + a^{28}x^{14} + a^{92}x^{13} + a^{193}x^{12} + a^{87}x^{11} + a^{218}x^{10} + a^{196}x^{9} + a^{102}x^{8} + a^{23}x^{7} + a^{15}x^{6} + a^{205}x^{5} + a^{101}x^{4} + a^{128}x^{3} + a^{116}x^{2} + a^{162}x^{1} + a^{76}$

3	$a^{70}x^{16} + a^{123}x^{15} + a^{52}x^{14} +$ $a^{46}x^{13} + a^{96}x^{12} + a^{248}x^{11} + a^{208}x^{10} + a^{44}x^{9} + a^{254}x^{8} + a^{30}x^{7} + a^{236}x^{6} +$ $a^{210}x^{5} + a^{100}x^{4} + a^{91}x^{3} + a^{196}x^{2} + a^{248}x^{1} + a^{168}$
4	$a^{32}x^{15} + a^{151}x^{14} +$ $a^{143}x^{13} + a^{168}x^{12} + a^{11}x^{11} + a^{12}x^{10} + a^{54}x^{9} + a^{65}x^{8} + a^{165}x^{7} + a^{99}x^{6} +$ $a^{103}x^{5} + a^{97}x^{4} + a^{72}x^{3} + a^{246}x^{2} + a^{27}x^{1} + a^{59}$
5	$a^{109}x^{14} +$ $a^{173}x^{13} + a^{248}x^{12} + a^{207}x^{11} + a^{110}x^{10} + a^{68}x^{9} + a^{114}x^{7} + a^{155}x^{6} +$ $a^{95}x^{5} + a^{56}x^{4} + a^{222}x^{3} + a^{232}x^{2} + a^{160}x^{1} + a^{40}$
6	$a^{44}x^{13} + a^{188}x^{12} + a^{236}x^{11} + a^{156}x^{10} + a^{124}x^{9} + a^{230}x^{8} + a^{69}x^{7} +$ $a^{27}x^{6} + a^{114}x^{5} + a^{85}x^{4} + a^{24}x^{3} + a^{78}x^{2} + a^{167}x^{1} + a^{104}$
7	$a^{202}x^{12} + a^{142}x^{11} + a^{185}x^{10} + a^{127}x^{9} + a^{139}x^{8} + a^{178}x^{7} +$ $a^{239}x^{6} + a^{206}x^{5} + a^{202}x^{4} + a^{162}x^{3} + a^{120}x^{2} + a^{63}x^{1} + a^{79}$
8	$a^{223}x^{11} + a^{246}x^{10} + a^{128}x^{9} + a^{227}x^{8} + a^{89}x^{7} +$ $a^{248}x^{6} + a^{69}x^{5} + a^{21}x^{4} + a^{226}x^{3} + a^{87}x^{2} + a^{179}$
9	$a^{157}x^{10} + a^{153}x^{9} + a^{120}x^{8} + a^{128}x^{7} +$ $a^{139}x^{6} + a^{119}x^{5} + a^{229}x^{4} + a^{190}x^{3} + a^{146}x^{2} + a^{239}x + a^{192}$
10	$a^{20}x^{9} + a^{182}x^{8} + a^{159}x^{7} +$ $a^{147}x^{6} + a^{36}x^{5} + a^{233}x^{4} + a^{14}x^{3} + a^{31}x^{2} + a^{56}x + a^{216}$

Βήμα i	$\varphi_i(x)$
1	$a^{239}x + a^{63}$
2	$a^{123}x + a^{39}$
3	$a^{105}x + a^{230}$

4	$a^{228}x + a^{214}$
5	$a^{38}x + a^{121}$
6	$a^{178}x + a^{50}$
7	$a^{65}x + a^{42}$
8	$a^{97}x + a^{224}$
9	$a^{234}x + a^{231}$
10	$a^{66}x + a^{166}$

Βήμα i	$s_i(x)$
-1	1
0	0
1	1
2	$a^{123}x + a^{39}$
3	$a^{228}x^2 + a^{235}x + a^{224}$
4	$a^{201}x^3 + a^{197}x^2 + a^{229}x + a^{174}$
5	$a^{239}x^4 + a^{147}x^3 + a^{208}x^2 + a^{202}x + a^{78}$
6	$a^{162}x^5 + a^4x^4 + a^{245}x^3 + a^{93}x^2 + a^{157}x + a^{10}$
7	$a^{227}x^6 + a^{213}x^5 + a^{147}x^4 + a^{25}x^3 + a^{155}x^2 + a^{94}x + a^{250}$
8	$a^{69}x^7 + a^{107}x^6 + a^{33}x^5 + a^2x^4 + a^{155}x^3 + a^{30}x^2 + a^{69}x + a^{101}$
9	$a^{48}x^8 + a^{202}x^7 + a^{160}x^6 + a^{25}x^5 + a^{72}x^3 + a^{10}x^2 + a^{145}x + a^{54}$
10	$a^{114}x^9 + a^{167}x^8 + a^{78}x^7 + a^{43}x^6 + a^{126}x^5 + a^{36}x^4 + a^{110}x^3 + a^{181}x^2 + a^{152}x + a^{254}$

Βήμα i	$t_i(x)$
-1	0
0	1
1	$a^{239}x + a^{63}$
2	$a^{107}x^2 + a^{205}x + a^{221}$
3	$a^{212}x^3 + a^{159}x^2 + a^{77}x + a^{58}$
4	$a^{185}x^4 + a^{238}x^3 + a^{151}x^2 + a^{139}x + a^{204}$
5	$a^{223}x^5 + a^{87}x^4 + a^{47}x^3 + a^{29}x^2 + a^{220}x + a^{185}$
6	$a^{146}x^6 + a^{210}x^5 + a^{197}x^4 + ax^3 + a^{199}x^2 + a^{20}x + a^{249}$

7	$a^{211}x^7 + a^{100}x^6 + a^{225}x^5 + a^3x^4 + a^{186}x^3 + a^{65}x^2 + a^{55}x + a^{224}$
8	$a^{53}x^8 + a^{248}x^7 + a^{43}x^6 + a^{229}x^5 + a^{50}x^4 + a^{246}x^3 + a^{223}x^2 + a^{19}x + a^{69}$
9	$a^{32}x^9 + a^{55}x^8 + a^{37}x^7 + a^{105}x^6 + a^{171}x^5 + a^{142}x^4 + a^{38}x^3 + a^{232}x^2 + a^{222}x + a^{82}$
10	$a^{98}x^{10} + a^{94}x^9 + a^{169}x^8 + a^{226}x^7 + a^{242}x^6 + a^{116}x^5 + a^5x^4 + a^{72}x^3 + a^{229}x^2 + a^{72}x + a^{106}$

Ο αλγόριθμος τερματίζει μόλις ο βαθμός του πολυωνύμου $u_i(x)$ γίνει μικρότερος από t =10 και γνωρίζοντας πως ο σταθερός όρος του σ(x) είναι ίσος με 1 υπολογίζουμε το λ: σ(x) = $\lambda \cdot t_{10}(x)$ ή λ(a^{106}) = 1 ή λ = a^{149}. Επομένως, σ(x) = $a^{247}x^{10} + a^{243}x^9 + a^{63}x^8 + a^{120}x^7 + a^{136}x^6 + a^{10}x^5 + a^{154}x^4 + a^{221}x^3 + a^{123}x^2 + a^{221}x + 1$ και ω(x) = $\lambda \cdot u_3(x) = a^{169}x^9 + a^{76}x^8 + a^{53}x^7 + a^{41}x^6 + a^{185}x^5 + a^{127}x^4 + a^{163}x^3 + a^{180}x^2 + a^{205}x + a^{110}$. Επειδή σ(x) είναι $10^{ου}$ βαθμού, θα έχει 10 ρίζες. Αναζητούμε αυτές τις ρίζες δοκιμάζοντας τις τιμές x = 1, a, a^2, …, a^{254}. Η δοκιμή των 255 τιμών γίνεται με την προσάρτηση του παρακάτω κώδικα MATLAB:

```
solution_count=0;
solutions=[];
for i=0:2^m-2
    ai=a^i;
    v=1;
    for j=1:t
     v=v+sig_x(j)*ai^(t+1-j);
    end
    if v==0
        solutions=[solutions,a^i];
        fprintf('a^%d is a solution \n',i);
        solution_count=solution_count+1;
    end
end
```

Οπότε, λαμβάνουμε τις λύσεις a^{62}, a^{80}, a^{85}, a^{100}, a^{141}, a^{156}, a^{212}, a^{219}, a^{238}, a^{245} του πολυώνυμου εντοπισμού σφαλμάτων. Στη συνέχεια, για να βρεθούν οι θέσεις των λαθών αρκεί να βρεθούν τα αντίστροφα ριζών:

β1 = $(a^{62})^{-1} = a^{193}$, β2 = $(a^{80})^{-1} = a^{175}$, β3 = $(a^{85})^{-1} = a^{170}$, β4 = $(a^{100})^{-1} = a^{155}$, β5 = $(a^{141})^{-1} = a^{114}$, β6 = $(a^{156})^{-1} = a^{99}$, β7 = $(a^{212})^{-1} = a^{43}$, β8 = $(a^{219})^{-1} = a^{36}$, β9 = $(a^{238})^{-1} = a^{17}$, β9 = $(a^{245})^{-1} = a^{10}$

Στις παραπάνω θέσεις είχαμε εισάγει λάθη στο παράδειγμα που επιλέξαμε να αποκωδικοποιήσουμε. $\sigma'(x) = \sum_{j=1}^{10} \beta_j \prod_{k=1,\ k\neq j}^{10} (1 + \beta_k x)$

Χρησιμοποιούμε τον παρακάτω κώδικα MATLAB:

```
inv=solutions.^-1; %Αντίστροφα ριζών
error_pos=log(inv); %θέσεις των λαθών
der_sig=gf(zeros(1,t),m,d); %παράγωγος του πολυωνύμου εντοπισμού σφαλμάτων
for j=1:t
    temp=gf([1],m,d);
    for k=1:t
        if k~=j
            temp=conv(temp,gf([inv(k) 1],m,d));
        end
    end
    der_sig=der_sig+temp*inv(j);
end

for i=1:t
    if der_sig(i)~=0
        fprintf('a^%d*x^%d',log(der_sig(i)),t-i);
        if i<t
            fprintf(' + ');
        else
            fprintf('\n');
        end
    end

end
```

Οπότε, προκύπτει ότι: $\sigma'(x) = a^{243}x^8 + a^{120}x^6 + a^{10}x^4 + a^{221}x^2 + a^{221}$

Υπολογίζουμε τα λάθη με τον τύπο $e_{jl} = \frac{\omega(\beta l^{-1})}{\sigma\prime(\beta l^{-1})}$ με τον παρακάτω κώδικα MATLAB:

```
errors=gf(zeros(1,t),m,d);
for i=1:t
    errors(i)=polyval(o_x,solutions(i))/polyval(der_sig,solutions(i)); %τιμές
σφαλμάτων
end
errors==er %Η αποκωδικοποίηση πέτυχε ;
```

Οπότε, από τα παραπάνω, ισχύει ότι:

$e_{193} = \frac{\omega(a^{62})}{\sigma\prime(a^{62})} = a^{58}$

$e_{175} = \frac{\omega(a^{80})}{\sigma\prime(a^{80})} = a^{45}$

$e_{170} = \frac{\omega(a^{85})}{\sigma\prime(a^{85})} = a^{72}$

$e_{155} = \frac{\omega(a^{100})}{\sigma\prime(a^{100})} = a^{240}$

$$e_{114} = \frac{\omega(a^{141})}{\sigma\prime(a^{141})} = a^{160}$$

$$e_{99} = \frac{\omega(a^{156})}{\sigma\prime(a^{156})} = a^{12}$$

$$e_{43} = \frac{\omega(a^{212})}{\sigma\prime(a^{212})} = a^{11}$$

$$e_{36} = \frac{\omega(a^{219})}{\sigma\prime(a^{219})} = a^{4}$$

$$e_{17} = \frac{\omega(a^{238})}{\sigma\prime(a^{238})} = a^{236}$$

$$e_{10} = \frac{\omega(a^{245})}{\sigma\prime(a^{245})} = a^{101}$$

Επαληθεύουμε ότι πρόκειται για τις τιμές των συντελεστών των όρων του διανύσματος λάθους που είχαμε εισάγει στο κωδικό πολυώνυμο.

ii) Αποκωδικοποίηση με αυτοπαλινδρόμηση στο MATLAB

Τροποποιούμε τον κώδικα 9.24, αλλάζοντας τις παραμέτρους και τη σειρά γραφής των bit ώστε το αριστερότερο να είναι το MSB. Οπότε, προκύπτει ο παρακάτω κώδικας:

```
%%%RS_decoding_autoregression_ATSC
%%%Reed-Solomon decoding using Autoregression%%%

clear all; close all;
t=10;
m=8;
k=187;
n=2*t+k;
d=primpoly(m);%primitive polynomial of gf(2^m)
a=gf(2,m,d); %the first non-zero, non-one element of the field
%b will contain the non-zero elements of gf(2^m)
b=gf([],m,d);
b(n)=1;
for i=n-1:-1:1
    b(i)=a^i;
end
% All elements of b are discrete, thus p(x) is primitive
%An all-zero 60-bit block(encoded with this encoder) is transmitted. At the receiver,
%(counted from the left-most bit position 0, or MSB), that is,
%the following bit stream is received:
rb=zeros(1,m*n); %an all-zero codeword
e_pos=[106,107, 109, 112, 249, 250, 256, 290, 291, 294, 296, 411, 413, 414, 737, 738,
739, 742, 743, 1249, 1250, 1253, 1254, 1256, 1305, 1306, 1307, 1309, 1364, 1513, 1514,
1517, 1519, 1520, 1571, 1575]; %bit-error positions within the
                                %(binary) codeword, from MSB
for i=1:length(e_pos)
    rb(e_pos(i))=~rb(e_pos(i)); %We invert the bits at erroneous bit positions to get
                                %the received message
end
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%Combining the bits in fives (elements of the GF(2^m)), we get:
%gfr(x)=r(x)
```

```
gfr=[];
for i=1:m:n*m
    elem=0;
    for j=0:m-1
        elem=elem+rb(i+j)*2^(m-1-j);
    end
    gfr=[gfr, gf(elem,m)];
end

%{
%Το παραπάνω μπορεί να γίνει και με χρήση της bi2de
rm=reshape(rb,m,length(rb)/m);
r=gf(bi2de(rm','left-msb'),m)';
%}

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%Syndrome Calculation
s=gf([],m,d);
for i=1:2*t
    s(i)=gfr(n);
    for j=1:n-1
        s(i)=s(i)+(a^i)^j*gfr(n-j);
    end
end
S=gf([],m,d);
cons_s=gf([],m,d);
for i=1:t
    for j=0:t-1
        S=[S s(i+j)];
    end
    cons_s(i)=-s(t+i);
end
S=reshape(S,t,length(S)/t)';
if det(S)~=0
    sig=S^-1*cons_s';
    %Find roots of sigma
    s_roots_exp=[];   %vector of sigma roots exponents(of a)
    for i=1:2^m-1
        ai=a^i;
        sr=gf(1,m,d);
        for j=1:t
            sr=sr+sig(j)*ai^(t-j+1);
        end
        if (sr==0)
            s_roots_exp=[s_roots_exp i];
        end
    end

    b=(a.^s_roots_exp).^-1;   %inverses of sigma roots
    er_pos=n-log(b);   %error positions from ms digit
    B=gf([],m,d);
    cons_b=gf([],m,d);
    for i=1:t
        for j=1:t
            B=[B b(j)^i];
        end
        cons_b(i)=s(i);
    end
    B=reshape(B,t,length(B)/t)';
    er=(B^-1*cons_b')';
    u=gfr;
    for i=1:length(er)
        u(er_pos(i))=gfr(er_pos(i))+er(i);
    end
    gfr
    u
else
    disp('Determinant equal zero');
end
```