In this document we recreate all the examples from the article as well as demonstrate/test the code given in the article. Some examples are done twice, once "by hand" and once using a function that is more automatic, i.e. it combines all the needed steps. "By hand" means breaking it down into steps that use Matlab functions but not any custom functions (to the degree possible).

Table of Contents

Example 2.1 Sbox for ABCAES1

All Sbox values for ABCAES1

```
h=unique([25 28 23 4 2 15 10 19 26 16 22 25])
```

```
h = 1×11
     2     4    10    15    16    19    22    23    25    26    28
```

```
[~,hinv,~]=gcd(h,29);
[h;mod(hinv,29)]
```

```
ans = 2×11
     2     4    10    15    16    19    22    23    25    26    28
    15    22     3     2    20    26     4    24     7    19    28
```

```
Sbox=[h;mod(10.*hinv+17,29)]
```

```
Sbox = 2×11
     2     4    10    15    16    19    22    23    25    26    28
    22     5    18     8    14    16    28    25     0     4     7
```

Proof of Lemma 2.2 no fix pts Sbox ABCAES1

Analytic approach to showing that Sbox has no fixed points.

$$ah^{-1} + b = h \iff a + bh = h^2 \iff h^2 - bh - a = 0 \iff \sqrt{b^2 + 4a} \in Z_{29}$$

Matlab has the Jacobi symbol, which returns 1 if something is quadratic residue, and -1 if it is not.

```
a=10;
b=17;
mod(b^2+4*a,29)
```

```
ans = 10
```

```
jacobiSymbol(b^2+4*a,29)
```

```
ans = -1
```

Example 2.2 MixColumns for ABCAES1

```
MC=[1 4;4 1]
```

```
MC = 2×2
     1     4
     4     1
```

```
State=[22 08;16 18]
```

```
State = 2×2
    22     8
    16    18
```

```
MC*State
```

```
ans = 2×2
    86    80
   104    50
```

```
mod(MC*State,29)
```

```
ans = 2×2
    28    22
    17    21
```

Key expansion ABCAES1

```
W0 = [11 05];
W1 = [25 28];
```

```
rcon = [9 0];
rotword=[W1(2) W1(1)]
```

```
rotword = 1×2
    28    25
```

```
[~,hinv,~]=gcd(rotword,29);
subrot=mod(10.*hinv+17,29)
```

```
subrot = 1×2
     7     0
```

```
W2 = mod(W0+rcon+subrot,29)
```

```
W2 = 1×2
    27     5
```

```
W3=mod(W1+W2,29)
```

```
W3 = 1×2
    23     4
```

```
rcon = [23 0];
rotword=[W3(2) W3(1)]
```

```
rotword = 1×2
     4    23
```

```
[~,hinv,~]=gcd(rotword,29);
subrot=mod(10.*hinv+17,29)
```

```
subrot = 1×2
     5    25
```

```
W4 = mod(W2+rcon+subrot,29)
```

```
W4 = 1×2
    26     1
```

```
W5 = mod(W3+W4,29)
```

```
W5 = 1×2
    20     5
```

```
K0 = [W0 W1]
```

```
K0 = 1×4
    11     5    25    28
```

```
K1 = [W2 W3]
```

```
K1 = 1×4
    27     5    23     4
```

```
K2 = [W4 W5]
```

```
K2 = 1×4
    26     1    20     5
```

ABCAES1 two rounds step by step

```
K0 = [11 25; 5 28];
K1 = [27 23; 5 4];
K2 = [26 20; 1 5];
state = [20 19; 5 20];
state = mod(state+K0,29) % add key
```

```
state = 2×2
     2    15
    10    19
```

```
[~,hinv,~]=gcd(state,29);
state = mod(10.*hinv+17,29)
```

```
state = 2×2
    22     8
    18    16
```

```
state = [22 8; 16 18] % manually shift second row
```

```
state = 2×2
    22     8
    16    18
```

```
state = [28 22; 17 21] % look up MC results from above
```

```
state = 2×2
    28    22
    17    21
```

```
state = mod(state+K1,29)
```

```
state = 2×2
    26    16
    22    25
```

```
[~,hinv,~]=gcd(state,29);
state = mod(10.*hinv+17,29)
```

```
state = 2×2
     4    14
    28     0
```

```
state = [4 14; 0 28] % manually apply shift second row
```

```
state = 2×2
     4    14
     0    28
```

```
state = mod(state+K2,29)
```

```
state = 2×2
     1     5
     1     4
```

ABCAES1 decryption: Finding Aff inv and MC inv

```
a=10;
```

```
b=17;
[~,ainv,~]=gcd(10,29)
```

```
ainv = 3
```

```
mod(-ainv*b,29)
```

```
ans = 7
```

```
MCex = [1 4; 4 1];
[~,detinv,~]=gcd(det(MCex),29)
```

```
detinv = -2
```

```
MCinv=mod(detinv*[1 -4; -4 1],29)
```

```
MCinv = 2×2
    27     8
     8    27
```

```
mod(MCex*MCinv,29)
```

```
ans = 2×2
     1     0
     0     1
```

Example 5.1 Sbox calculations for ABCAES4

We do the first polynomial using steps that can be verified by hand

```
syms x
f= x^4 + 2*x^2 +15*x +2;
a = x^3+2*x^2+3*x+4;
b = 5*x^3+6*x^2+7*x+8;
h = 9*x^3+10*x^2+11*x+12
```

h = $9\,x^3+10\,x^2+11\,x+12$

```
[~,hinv,~]=gcd(h,f);
hinv=pmod(hinv,29);
aff = expand(a*hinv+b)
```

aff = $23\,x^6+58\,x^5+108\,x^4+186\,x^3+145\,x^2+136\,x+100$

```
aff = pmod(aff,29)
```

aff = $23\,x^6+21\,x^4+12\,x^3+20\,x+13$

```
aff = polynomialReduce(aff,x^4+1)
```

aff = $12\,x^3-23\,x^2+20\,x-8$

```
sbox =pmod(aff,29)
```

sbox = $12\,x^3+6\,x^2+20\,x+21$

Now we do all the polynomials at once and combine steps

```
syms x
f= x^4 + 2*x^2 +15*x +2;
a = x^3+2*x^2+3*x+4;
b = 5*x^3+6*x^2+7*x+8;

h = [9*x^3+10*x^2+11*x+12, 13*x^3+14*x^2+15*x+16,
    10*x^3+20*x^2+26*x+24, 01*x^3+26*x^2+13*x+16,
    21*x^3+10*x^2+12*x+23, 05*x^3+15*x^2+26*x+08,
    22*x^3+15*x^2+01*x+02, 14*x^3+21*x^2+20*x+14,
    02*x^3+01*x^2+23*x+10, 26*x^3+10*x^2+07*x+01,
    04*x^3+27*x^2+22*x+19, 15*x^3+13*x^2+07*x+00];
[~,hinv,~]=gcd(h,f);
hinv=arrayfun(@(h) pmod(h,29),hinv);
aff = a*hinv+b;
aff = arrayfun(@(h) polynomialReduce(h,x^4+1),aff);
Sbox = arrayfun(@(h) pmod(h,29),aff)
```

```
Sbox =
```

$$\begin{pmatrix} 12\,x^3+6\,x^2+20\,x+21 & 8\,x^2+11\,x+8 \\ 22\,x^3+23\,x^2+9\,x+24 & 22\,x^3+7\,x^2+11\,x+5 \\ x^3+3\,x^2+22\,x+1 & 11\,x^3+24\,x^2+8\,x+28 \\ x^3+5\,x^2+4\,x+10 & 8\,x^3+28\,x^2+7\,x+10 \\ 11\,x^3+18\,x^2+12\,x+18 & 6\,x^3+3\,x^2+19\,x+27 \\ 2\,x^3+14\,x^2+2\,x+25 & 26\,x^3+24\,x^2+5\,x+17 \end{pmatrix}$$

Next couple lines are for getting just the coefficients of the Sbox calculations.

```
hcoeffs = arrayfun(@(h) coeffs(h,'All'),h,'UniformOutput',false);
Sboxcoeffs = arrayfun(@(h) coeffs(h,'All'),Sbox,'UniformOutput',false);
```

Pad any list of coefficients that have fewer than 4 entries

```
for n = 1:12
    Sboxcoeffs{n} = [zeros(4-length(Sboxcoeffs{n}),1) Sboxcoeffs{n}];
end
```

Print the coefficients of each h, and then the Sbox output, in format ready to be pasted into LaTeX

```
for n=1:12
   fprintf('%02i%02i%02i%02i & ',double(hcoeffs{n}));
end
```

```
09101112 & 10202624 & 21101223 & 22150102 & 02012310 & 04272219 & 13141516 & 01261316 & 05152608 & 1421201
```

```
for n=1:12
   fprintf('%02i%02i%02i%02i & ',double(Sboxcoeffs{n}));
end
```

```
12062021 & 22230924 & 01032201 & 01050410 & 11181218 & 02140225 & 00081108 & 22071105 & 11240828 & 0828071
```

The BIG calculation: checking ABCAES4 Sbox for fixed points

```
f = [1 0 2 15 2]; % conway polynomial
a = 1:4; % from the affine map
b= 5:8; % from the affine map
BCwords = fliplr(gftuple([-1:29^4-2]',fliplr(f),29));
BCinvs=[BCwords(1:2,:); flipud(BCwords(3:end,:))];
BCsbox = zeros(size(words));
for n = 1:29^4
    ahinv=mod(conv(invs(n,:),a),29); % a*h^{-1}
    [~,ahinv]=deconv(ahinv,[1 0 0 0 1]); % reduce mod x^4+1
    ahinv = ahinv(4:7);
    BCsbox(n,:)=mod(ahinv+b,29); % add b
end
equalrows = zeros(1,29^4);
for n = 1:29^4
    equalrows(n)=isequal(BCwords(n,:),BCsbox(n,:));
end
idx=find(equalrows==1);
length(idx) % if this is >0 then we have a fixed point.
```

```
ans = 0
```

Double checking the big calculation

To run this section we need to have first run the previous section to fill "BCwords" and "BCsbox".

```
%% WARNING this takes about 1.5min to run.
tic
syms x
N = 1e3; % number of calculations we'll double check
idx = randi(29^4-1,1,N)+1;
BC_word_polys = BCwords(idx,:)*[x^3 x^2 x 1].'; % convert some random word inputs to po
BC_sbox_polys = BCsbox(idx,:)*[x^3 x^2 x 1].'; % convert the results of sbox calculati
f= x^4 + 2*x^2 +15*x +2; % start to take random word inputs and symbolically calculate
a = x^3+2*x^2+3*x+4;
b = 5*x^3+6*x^2+7*x+8;
h = BC_word_polys;
[~,hinv,~]=gcd(h,f);
hinv=arrayfun(@(h) pmod(h,29),hinv);
aff = a*hinv+b;
aff = arrayfun(@(h) polynomialReduce(h,x^4+1),aff);
Sbox = arrayfun(@(h) pmod(h,29),aff); % outputs of symbolic sbox calculation
isequal(BC_sbox_polys,Sbox)
```

```
ans = logical
   1
```

```
idx = randi(N,1,5); % let's look at 5 of them at random
BC_sbox_polys(idx) % the results of the previous section, converted to polynomials
```

```
ans =
```

$$\begin{pmatrix} 5x^3+x^2+26x+10 \\ 9x^3+5x^2+22x+4 \\ 18x^3+8x^2+22x+15 \\ 24x^3+10x^2+x+24 \\ 8x^3+18x^2+26x+13 \end{pmatrix}$$

```
Sbox(idx) % the symbolic double check we've done in this section
```

```
ans =
```

$$\begin{pmatrix} 5x^3+x^2+26x+10 \\ 9x^3+5x^2+22x+4 \\ 18x^3+8x^2+22x+15 \\ 24x^3+10x^2+x+24 \\ 8x^3+18x^2+26x+13 \end{pmatrix}$$

```
toc
```

```
Elapsed time is 94.681249 seconds.
```

Example 4.2 ABCAES4 Mix Columns

We do the calculations step by step so someone can check by hand

```
syms x
f = x^4+2*x^2+15*x+2;
M = [1 13*x^2; -13*x^2 1];

S = [x^3+03*x^2+22*x+1 x^3+05*x^2+4*x+10;...
    8*x^3+28*x^2+7*x+10 11*x^3+24*x^2+8*x+28];
MixCol = expand(M*S)
```

```
MixCol =
```

$$\begin{pmatrix} 104x^5+364x^4+92x^3+133x^2+22x+1 & 143x^5+312x^4+105x^3+369x^2+4x+10 \\ -13x^5-39x^4-278x^3+15x^2+7x+10 & -13x^5-65x^4-41x^3-106x^2+8x+28 \end{pmatrix}$$

```
MixCol = arrayfun(@(h) pmod(h,29), MixCol)
```

```
MixCol =
```

$$\begin{pmatrix} 17x^5+16x^4+5x^3+17x^2+22x+1 & 27x^5+22x^4+18x^3+21x^2+4x+10 \\ 16x^5+19x^4+12x^3+15x^2+7x+10 & 16x^5+22x^4+17x^3+10x^2+8x+28 \end{pmatrix}$$

```
MixCol = arrayfun(@(h) polynomialReduce(h,f),MixCol)
```

```
MixCol =
```

$$\begin{pmatrix} -29x^3-270x^2-252x-31 & -36x^3-428x^2-380x-34 \\ -20x^3-263x^2-310x-28 & -15x^3-274x^2-354x-16 \end{pmatrix}$$

```
MixCol = arrayfun(@(h) pmod(h,29), MixCol)
```

```
MixCol =
```

$$\begin{pmatrix} 20\,x^2+9\,x+27 & 22\,x^3+7\,x^2+26\,x+24 \\ 9\,x^3+27\,x^2+9\,x+1 & 14\,x^3+16\,x^2+23\,x+13 \end{pmatrix}$$

Next few lines are to extract the coefficients.

```
MCcoeffs = arrayfun(@(h) coeffs(h,'All'),MixCol,'UniformOutput',false);
MCcoeffs{1,1}=[0 MCcoeffs{1,1}];
[double(MCcoeffs{1,1}) double(MCcoeffs{1,2}) ;
    double(MCcoeffs{2,1}) double(MCcoeffs{2,2})]
```

```
ans = 2×8
     0    20     9    27    22     7    26    24
     9    27     9     1    14    16    23    13
```

Key expansion ABCAES4

```
W0 = 1:8;
W1 = 9:16;
rcon = [0 0 1 0 zeros(1,4)];
subrot = [0 8 11 8 12 6 20 21]% Look up Sbox of W1(5:8) and W1(1:4)
```

```
subrot = 1×8
     0     8    11     8    12     6    20    21
```

```
W2 = mod(W0+rcon+subrot,29)
```

```
W2 = 1×8
     1    10    15    12    17    12    27     0
```

```
W3=mod(W1+W2,29);
rcon = [0 1 0 0 zeros(1,4)];
subrot=[22 7 11 5 22 23 9 24] % Look up Sbox of W3(5:8) and W3(1:4)
```

```
subrot = 1×8
    22     7    11     5    22    23     9    24
```

```
W4 = mod(W2+rcon+subrot,29)
```

```
W4 = 1×8
    23    18    26    17    10     6     7    24
```

```
W5 = mod(W3+W4,29)
```

```
W5 = 1×8
     4     9    23    12    11     3    20    11
```

```
K0 = [W0 W1]
```

```
K0 = 1×16
     1     2     3     4     5     6     7     8     9    10    11    12    13⋯
```

```
K1 = [W2 W3]
```

```
K1 = 1×16
     1    10    15    12    17    12    27     0    10    20    26    24     1⋯
```

```
K2 = [W4 W5]
```

```
K2 = 1×16
    23    18    26    17    10     6     7    24     4     9    23    12    11···
```

ABCAES4 two rounds step by step

In this section we do the final assembly of parts for the complete encryption. Above we did the Sbox and Mix Column calculations. We use those here now, but just fill them in by hand as needed. The rest is key expansion, then adding the keys, and decoding the numbers into letters.

```
% we assume W0, W1, W2, W3, W4 and W5 have been calculated above
K0 = [W0(1:4) W1(1:4);
      W0(5:8) W1(5:8)]
```

```
K0 = 2×8
     1     2     3     4     9    10    11    12
     5     6     7     8    13    14    15    16
```

```
K1 = [W2(1:4) W3(1:4);
      W2(5:8) W3(5:8)]
```

```
K1 = 2×8
     1    10    15    12    10    20    26    24
    17    12    27     0     1    26    13    16
```

```
K2 = [W4(1:4) W5(1:4);
      W4(5:8) W5(5:8)]
```

```
K2 = 2×8
    23    18    26    17     4     9    23    12
    10     6     7    24    11     3    20    11
```

```
state = encode("this is message.");
state = [state(1:4) state(9:12);
    state(5:8) state(13:16)]
```

```
state = 2×8
    20     8     9    19    13     5    19    19
     0     9    19     0     1     7     5    27
```

```
state = mod(state+K0,29) % add key
```

```
state = 2×8
    21    10    12    23    22    15     1     2
     5    15    26     8    14    21    20    14
```

```
state = [1 3 22 1 1 5 4 10;
    11 24 8 28 8 28 7 10] % look up Sbox values from above
```

```
state = 2×8
     1     3    22     1     1     5     4    10
    11    24     8    28     8    28     7    10
```

```
state(2,:) = [state(2,5:8) state(2,1:4)] % manually shift second row
```

```
state = 2×8
     1     3    22     1     1     5     4    10
     8    28     7    10    11    24     8    28
```

```
state = [0 20 9 27 22 7 26 24;
    9 27 9 1 14 16 23 13]%  look up MC results from above
```

```
state = 2×8
     0    20     9    27    22     7    26    24
     9    27     9     1    14    16    23    13
```

```
state = mod(state+K1,29) % add K1
```

```
state = 2×8
     1     1    24    10     3    27    23    19
    26    10     7     1    15    13     7     0
```

```
state = [2 17 13 9 11 15 0 28;
    6 3 19 27 26 24 5 17] % look up Sbox results above
```

```
state = 2×8
     2    17    13     9    11    15     0    28
     6     3    19    27    26    24     5    17
```

```
state(2,:) = [state(2,5:8) state(2,1:4)] % manually apply shift second row
```

```
state = 2×8
     2    17    13     9    11    15     0    28
    26    24     5    17     6     3    19    27
```

```
state = mod(state+K2,29)
```

```
state = 2×8
    25     6    10    26    15    24    23    11
     7     1    12    12    17     6    10     9
```

```
state = [state(1,1:4) state(2,1:4) state(1,5:8) state(2,5:8)]
```

```
state = 1×16
    25     6    10    26     7     1    12    12    15    24    23    11    17···
```

```
decode(state)
```

```
ans = 
'YFJZGALLOXWKQFJI'
```

ABCAES4 decryption: Finding Aff inv and MC inv

```
syms x
a = x^3+2*x^2+3*x+4;
b = 5*x^3+6*x^2+7*x+8;
[~,ainv,~]=gcd(a,x^4+1);
ainv = pmod(ainv,29)
```

ainv = $19\,x^3+3\,x^2+26\,x+8$

```
pmod(polynomialReduce(-ainv*b,x^4+1),29)
```

ans = $8\,x^3+15\,x^2+10\,x+14$

```
MCmat = [1 13*x^2; -13*x^2 1];
% find the determinant, reduced mod f, and then mod 29
```

```
MCdet = pmod(polynomialReduce(det(MCmat),f),29)
```

MCdet = $10\,x^2 + 17\,x + 11$

```
% get the inverse, and then reduce again mod 29
[~,detinv,~]=gcd(MCdet,f);
detinv = pmod(detinv,29)
```

detinv = $3\,x^3 + 17\,x^2 + 26\,x + 6$

```
% Multiply the determinant inverse, then reduce mod f and 29
MCinv=arrayfun(@(h) pmod(polynomialReduce(h,f),29),detinv*[1 -13*x^2; 13*x^2 1])
```

MCinv =

$$\begin{pmatrix} 3\,x^3 + 17\,x^2 + 26\,x + 6 & x^3 + 21\,x^2 + 7 \\ 28\,x^3 + 8\,x^2 + 22 & 3\,x^3 + 17\,x^2 + 26\,x + 6 \end{pmatrix}$$

```
% double check product to get identity
arrayfun(@(h) pmod(polynomialReduce(h,f),29), MCmat*MCinv)
```

ans =

$$\begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix}$$

ABCAES4 Single Matlab function for Encryption Verbose mode

This tests the MATLAB functions we have created that create a full encryption in a single function call. 'verbose' mode displays each step of the process. Later we use this function repeatedly to look at confusion and diffusion, etc., but here we show the verbose output.

```
ABCstream("this is message.",[1:16],'verbose')
```

```
plaintext = 1×16
    20     8     9    19     0     9    19     0    13     5    19    19     1 ···
key = 1×16
     1     2     3     4     5     6     7     8     9    10    11    12    13 ···
plaintext after encoding and padding
plaintext = 1×16
    20     8     9    19     0     9    19     0    13     5    19    19     1 ···
key = 1×16
     1     2     3     4     5     6     7     8     9    10    11    12    13 ···
keyexpand mode:
W0 = 1×8
     1     2     3     4     5     6     7     8
W1 = 1×8
     9    10    11    12    13    14    15    16
RC1 = 1×4
     0     0     1     0
sbox(rotword(W1))=
ans = 1×8
     0     8    11     8    12     6    20    21
W2 = 1×8
     1    10    15    12    17    12    27     0
W3 = 1×8
    10    20    26    24     1    26    13    16
RC2 = 1×4
     0     1     0     0
```

```
sbox(rotword(W3))=
ans = 1×8
    22     7    11     5    22    23     9    24
W4 = 1×8
    23    18    26    17    10     6     7    24
W5 = 1×8
     4     9    23    12    11     3    20    11
K0 = 1×16
     1     2     3     4     5     6     7     8     9    10    11    12    13⋯
K1 = 1×16
     1    10    15    12    17    12    27     0    10    20    26    24     1⋯
K2 = 1×16
    23    18    26    17    10     6     7    24     4     9    23    12    11⋯
end keyexpand
K0 = 1×16
     1     2     3     4     5     6     7     8     9    10    11    12    13⋯
K1 = 1×16
     1    10    15    12    17    12    27     0    10    20    26    24     1⋯
K2 = 1×16
    23    18    26    17    10     6     7    24     4     9    23    12    11⋯
Next add K0
ciphertext = 2×8
    21    10    12    23    22    15     1     2
     5    15    26     8    14    21    20    14
Next apply WS
ciphertext = 2×8
     1     3    22     1     1     5     4    10
    11    24     8    28     8    28     7    10
Next apply SR
ciphertext = 2×8
     1     3    22     1     1     5     4    10
     8    28     7    10    11    24     8    28
Next apply MC
First column of words, input = [h1;h2]
Row 1
multiply 13*x^2*h2
row1 = 1×7
     0   104   364    91   130     0     0
reduce this modulo x^4+2x^2+15x+2
row1 = 1×7
           0           0           0        -117       -2158       -5668⋯
normalize length of 13*x^2*h2
row1 = 1×4
        -117       -2158       -5668        -728
add h1, and reduce mod 29
row1 = 1×4
     0    20     9    27
Row 2 of output (in one column)
multiply -13*x^2*h1
row2 = 1×7
     0   -13   -39  -286   -13     0     0
reduce this modulo x^4+2x^2+15x+2
row2 = 1×7
     0     0     0  -260   260   611    78
normalize length of -13*x^2*h1
add h2, and reduce mod 29
row2 = 1×4
     9    27     9     1
final result for this column
colout = 2×4
     0    20     9    27
     9    27     9     1
```

```
2nd column of words
Row 1
multiply 13*x^2*h2
row1 = 1×7
     0   143   312   104   364     0     0
reduce this modulo x^4+2x^2+15x+2
row1 = 1×7
           0           0           0        -182       -2405       -4966 ⋯
normalize length of 13*x^2*h2
row1 = 1×4
        -182       -2405       -4966        -624
add h1, and reduce mod 29
row1 = 1×4
    22     7    26    24
Row 2 of output (in one column)
multiply -13*x^2*h1
row2 = 1×7
     0   -13   -65   -52  -130     0     0
reduce this modulo x^4+2x^2+15x+2
row2 = 1×7
           0           0           0         -26         195        1001 ⋯
normalize length of -13*x^2*h1
add h2, and reduce mod 29
row2 = 1×4
    14    16    23    13
final result for this column
colout = 2×4
    22     7    26    24
    14    16    23    13
ciphertext = 2×8
     0    20     9    27    22     7    26    24
     9    27     9     1    14    16    23    13
Next add K1
ciphertext = 2×8
     1     1    24    10     3    27    23    19
    26    10     7     1    15    13     7     0
Next apply WS
ciphertext = 2×8
     2    17    13     9    11    15     0    28
     6     3    19    27    26    24     5    17
Next apply SR
ciphertext = 2×8
     2    17    13     9    11    15     0    28
    26    24     5    17     6     3    19    27
Next add K2
ciphertext = 2×8
    25     6    10    26    15    24    23    11
     7     1    12    12    17     6    10     9
Next decode
ans =
'YFJZGALLOXWKQFJI'
```

ABCAES Single Matlab function for decryption

```
plaintext="this is message.";
key = 1:16;
cipher = ABCstream(plaintext,key)
```

```
cipher =
'YFJZGALLOXWKQFJI'
```

```
ABCinv(cipher,key)
```

```
ans =
'THIS IS MESSAGE.'
```

Extra examples and observations

ABCAES4, message = Zs, key = As

```
key = ones(1,16);
plain = 26*ones(1,16);
decode(plain),decode(key)
```

```
ans =
'ZZZZZZZZZZZZZZZZ'
ans =
'AAAAAAAAAAAAAAAA'
```

```
ABCstream(plain,key)
```

```
ans =
'MSN?DKBTURIIKGOF'
```

ABCAES4, message = Zs, key = As + 1

```
key = ones(1,16);key(16)=2;
message = 26*ones(1,16);
decode(message),decode(key)
```

```
ans =
'ZZZZZZZZZZZZZZZZ'
ans =
'AAAAAAAAAAAAAAAB'
```

```
ABCstream(message,key)
```

```
ans =
'UEMVVZFPGGRCU EB'
```

ABCAES4, message = Zs+1, key = As

```
key = ones(1,16);
message = 26*ones(1,16);message(16)=25;
decode(message),decode(key)
```

```
ans =
'ZZZZZZZZZZZZZZZY'
ans =
'AAAAAAAAAAAAAAAA'
```

```
ABCstream(message,key)
```

```
ans =
'PTAFDKBTURIIZTLM'
```

ABCAES4 with extra MC, message = Zs, message = Zs+1

```
key = ones(1,16);
message = 26*ones(1,16);
decode(message),decode(key)
```

```
ans =
'ZZZZZZZZZZZZZZZZ'
ans =
'AAAAAAAAAAAAAAAA'
```

```
ABCstream_extraMC(message,key)
```

```
ans =
'U MWO?DZVEJTLHQP'
```

```
key = ones(1,16);
message = 26*ones(1,16);message(16)=25;
decode(message),decode(key)
```

```
ans =
'ZZZZZZZZZZZZZZZY'
ans =
'AAAAAAAAAAAAAAAA'
```

```
ABCstream_extraMC(message,key)
```

```
ans =
'XA AAZPW?WNA.UNW'
```

ABCAES4 with three rounds, message = Zs, message = Zs+1

```
key = ones(1,16);
message = 26*ones(1,16);
decode(message),decode(key)
```

```
ans =
'ZZZZZZZZZZZZZZZZ'
ans =
'AAAAAAAAAAAAAAAA'
```

```
ABCstream_r3(message,key)
```

```
ans =
'DRQJLAQSY.J M C?'
```

```
key = ones(1,16);
message = 26*ones(1,16);message(16)=25;
decode(message),decode(key)
```

```
ans =
'ZZZZZZZZZZZZZZZY'
ans =
'AAAAAAAAAAAAAAAA'
```

```
ABCstream_r3(message,key)
```

```
ans =
'QXIEDCDEDHUZPSBD'
```

Lee Metric basics

```
abs_lee([3,16,28,-60]) % the lee absolute value
```

```
ans = 1×4
     3    13     1     2
```

```
A = [1,2,3];
B = [1,1,1];
%% WARNING: d_Lee needs inputs that are numbers, not letters
d_lee(A,B)
```

```
ans = 3
```

Numerical evidence for Lee distances between random elements and vectors

Distance between two random elements of Z_{29}

```
N = 1e6;
data = zeros(1,N);
for n = 1:N
v = mod(randi(29,1,1),29);
w = mod(randi(29,1,1),29);
%% WARNING: d_Lee needs inputs that are numbers, not letters
data(n)=d_lee(v,w);
end
mean(data)
```

```
ans = 7.2458
```

Distance between two random elements of Z_{29}^{16}

```
N = 1e6;
data = zeros(1,N);
for n = 1:N
v = mod(randi(29,1,16),29);
w = mod(randi(29,1,16),29);
%% WARNING: d_Lee needs inputs that are numbers, not letters
data(n)=d_lee(v,w);
end
mean(data)
```

```
ans = 115.8702
```

Average Lee distance for all constant keys and constant plaintext

This calculates the average Lee distance between a plaintext and the resulting ciphertext, where the plaintext is allowed to vary over all constant messages and the key varies also over all constant keys.

```
%%%%%%%%%%% WARNING this takes about 7.5 minutes to run.
results = zeros(29,29);
tic
for n=  1:29
    for m = 1:29
```

```
        key = mod(n*ones(1,16),29);
        plain = mod(m*ones(1,16),29);
        cipher=encode(ABCstream(plain,key)); % encode returns numbers
%% WARNING: d_Lee needs inputs that are numbers, not letters
        results(n,m) = d_lee(cipher,plain);
    end
end
toc
```

```
Elapsed time is 469.331159 seconds.
```

```
mean(results,'all')
```

```
ans = 116.2699
```

Average Lee distance for confusion starting with all constant keys, one fixed plaintext.

This calculates the average Lee distance between two ciphertexts, which are produced by keys as close together as possible, one key being a constant key, and the plaintext is fixed at Zs. This shows that the amount of confusion created, even with such week keys and plaintext, is optimal.

```
%%%%%%%%%%% WARNING this takes about 8 minutes to run.
results = zeros(29,32);
tic
plain = 26*ones(1,16);
for n=  1:29
    key1 = mod(n*ones(1,16),29);
    cipher1 = encode(ABCstream(plain,key1)); % encode returns numbers
        for m = 1:32
        key2 = key1;
        idx = ceil(m/2);
        key2(idx)=mod(key2(idx)+(-1)^(m+1),29);
        cipher2=encode(ABCstream(plain,key2)); % encode returns numbers
        %% WARNING: d_Lee needs inputs that are numbers, not letters
        results(n,m) = d_lee(cipher1,cipher2);
    end
end
toc
```

```
Elapsed time is 529.734055 seconds.
```

```
mean(results,'all')
```

```
ans = 115.8427
```

Average Lee distance for diffusion starting with all constant plaintext, and using one fixed key.

This calculates the average Lee distance between two ciphertexts, which are produced by plaintext as close together as possible, one plaintext being a constant set of letters, and with a single key fixed at As. This shows that the amount of diffusion created, even with such week keys and plaintext, is optimal.

```
%%%%%%%%%%% WARNING this takes about 8 minutes to run.
results = zeros(29,32);
```

```
tic
key = ones(1,16);
for n=  1:29
    plain1 = mod(n*ones(1,16),29);
    cipher1 = encode(ABCstream(plain1,key)); % encode returns numbers
        for m = 1:32
        plain2 = plain1;
        idx = ceil(m/2);
        plain2(idx)=mod(plain2(idx)+(-1)^(m+1),29);
        cipher2=encode(ABCstream(plain2,key)); % encode returns numbers
        %% WARNING: d_Lee needs inputs that are numbers, not letters
    results(n,m) = d_lee(cipher1,cipher2);
    end
end
toc
```

```
Elapsed time is 535.530956 seconds.
```

```
mean(results,'all')
```

```
ans = 57.6067
```

```
function z = abs_lee(x)
x = mod(x,29);
z = min(x,29-x); % this works for vectors
end

function z = d_lee(x,y)
z = sum(abs_lee(x-y));
end
```