Cryptography

Homeworks

Ferro Demetrio	207872
Minetto Alex	211419

Contents

Homework 1 - Caesar code with permuted alphabet

Assignment

Decrypt the following message which has been encrypted using Caeser cipher with the alphabet letters in a permuted order that is part of the secret key.

```
 9, 13,  3,  4, 21, 30, 15, 28,  0, 24,  4,  0,  3, 14,  3, 21,
 4, 28, 12, 12, 30, 19,  3, 27,  4, 28,  4,  9, 21, 30, 17, 29,
12,  3,  4, 24,  5, 30,  9,  4,  9, 30,  4, 21,  3, 15, 28,  1,
 0,  4, 24,  1, 15,  5, 12, 16,  4,  9, 30,  4, 28, 14, 30, ...
```

The frequencies of the 26 alphabet letters plus 5 special characters (i.e. blank and punctuation marks) are orderly encoded as two digit numbers from 0 to 25, and: "blank" is encoded as 26, "-" is encoded as 27, "," is encoded as 28, "." is encoded as 29, ";" is encoded as 30, are:

"a",	5.856417935	"b",	0.915065302	"c",	1.746994285
"d",	3.161113468	"e",	8.734714250	"f",	1.622161218
"g",	1.688711422	"h",	4.999584061	"i",	5.814824058
"j",	0.116462856	"k",	0.457532651	"l",	2.753514683
"m",	1.256135097	"n",	5.606854671	"o",	6.239081607
"p",	0.873471425	"q",	0.066550200	"r",	3.660261209
"s",	5.257466101	"t",	7.528491806	"u",	2.104650195
"v",	0.856833874	"w",	1.514017137	"x",	0.124781632
"y",	3.105981199	"z",	0.058231428	" ",	15.97728974
"-",	0.066550200	",",	1.081440812	".",	6.738208136
"?",	0.016637550				

Problem Statement

Caeser cipher encrypting systems are usually based on a circular shift of the reference alphabet $\mathcal{A}$, generating a new set $\mathcal{A}_\sigma$, containing all the elements of $\mathcal{A}$ shifted by σ positions. In our case, we have a further permutation of the resulting shifted alphabet, call it $\mathcal{A}_{\sigma\pi_0}$.
Considering the set:

$$\mathcal{A} := \{a_n\}_{n=0}^{30} \equiv \{\ a\ b\ c\ d\ e\ f\ g\ h\ i\ j\ k\ l\ m\ n\ o\ p\ q\ r\ s\ t\ u\ v\ w\ x\ y\ z\ \quad -\ ,\ .\ ?\ \}$$

By applying Caeser ciphering $\Sigma(\cdot, \sigma)$ with $\sigma = 3$ we get to $\mathcal{A}_\sigma = \Sigma(\mathcal{A}, \sigma)$,
for every a_n, we define $a_m = a_{n+\sigma}\ \ \forall m, n = 0, \cdots, 30$:

$$\mathcal{A}_\sigma := \{a_m\}_{m=0}^{30} \equiv \{\ d\ e\ f\ g\ h\ i\ j\ k\ l\ m\ n\ o\ p\ q\ r\ s\ t\ u\ v\ w\ x\ y\ z\ \quad -\ ,\ .\ ?\ a\ b\ c\ \}$$

By applying a permutation $\Pi_0(\cdot)$ to the originary alphabeth we get to a new set $\mathcal{A}_{\pi_0} = \Pi_0(\mathcal{A})$.
By applying the same permutation $\Pi(\cdot)$ to the shifetd alphabeth $\mathcal{A}_\sigma$ we get to a new set $\mathcal{A}_{\sigma\pi_0} = \Pi_0(\mathcal{A}_\sigma) = \Pi_0(\Sigma(\mathcal{A}, \sigma))$.
We wanted to exploit the fact that applying a permutation to a shifted version of the alphabet is equivalent to apply a different permutation to the original alphabeth.
Calling Π the new permutation, $\Pi = \Pi_0(\Sigma(\cdot, \sigma))$, we get: $\mathcal{A}_{\sigma\pi_0} = \Pi_0(\Sigma(\mathcal{A}, \sigma)) = \Pi(\mathcal{A}) = \mathcal{A}_\pi$.

This can be shown even by using characters given by the assignment.

a	b	c	d	e	f	g	h	i	j	k	l	m	n	o	p	q	r	s	t	u	···	?

by applying a 3-characters shift, we get the following:

d	e	f	g	h	i	j	k	l	m	n	o	p	q	r	s	t	u	v	w	x	···	c

then we apply a random permutation:

v	f	c	p	k	.	g	-	q	n	h	e	,	u	y	?		z	s	o	a	···	l

the effect of the combination of the two operations may be interpreted as a single new permutation.

Given an encrypted text obtained by doing these operations, one can not determine either the permuted version of the alphabet comes by permuting the reference alphabet or by permuting a shifted version of it.

One can ignore the knowledge of the previous permutation rule $\Pi_0(\cdot)$ acting on the Caesar ciphered text, hence the knowledge of the Caesar Key σ, since they may be considered as a single permutation $\Pi(\cdot)$.

Cryptanalysis

The purpose of our cryptanalysis is to identify the permuted alphabet $\mathcal{A}_{\tilde{\pi}}$ in order to reconstruct the message by inverting the permutation: $\Pi^{-1}(\mathcal{A}_\pi) = \mathcal{A}$.

In order to do this, we start from the assumption that the *plaintext* $\mathcal{M}$ is written in English.

Considering $\mathcal{T}$ an exhaustive text containing all possible English words: $\mathcal{T} := \{t_i \in \mathcal{A}\}_{i=1}^{|\mathcal{T}|}$, The statistics of the occurrences of each character in the text may be defined as:

$$P(\mathcal{T}) = \frac{1}{|\mathcal{T}|}\sum_{i=1}^{|\mathcal{T}|}\delta(t_i = a_n), \quad \forall n = 0, \cdots, 30.$$

That essentially is the probability mass function of the English alphabet.

Since the text was encrypted by substitution cipher, we suppose that the distribution of the occurrences of the letters does not change even if they are substituted.

So, given our Encrypted message $\mathcal{E}$, we computed its probability mass function $P(\mathcal{E})$.

$$P(\mathcal{E}) = \frac{1}{|\mathcal{E}|}\sum_{i=1}^{|\mathcal{E}|}\delta(e_i = a_n), \quad \forall n = 0, \cdots, 30.$$

In order to find the correspondence between the occurrences of English letters and the letters of our message, we need to sort the probability mass functions (pmf), as shown in Figure 1.

By doing this, we get to a new disposition of the alphabet in both cases: $\mathcal{A}_\mathcal{T}$ and $\mathcal{A}_\mathcal{E}$, where the elements are sorted by their pmf.

From the Occurrences Analysis we can extract an Estimation of the permutation $\hat{\Pi}$, that can be used to determine $\hat{\Pi}^{-1}(\mathcal{E}) = \hat{\mathcal{M}}$, with the assumption that $\hat{\Pi} \simeq \Pi$.

Since the two mass functions $P(\mathcal{T})$, $P(\mathcal{E})$ may not be so reliable, due to the fact that for some characters the values assumed by the probability mass function are very close, the estimation of the permutation rule is altered.

In order to reconstruct the plaintext we need to read the message applying the inverse permutation rule and try to adjust it to improve the estimation of the permutation rule.

Implementation

```
Alphabeth0(1,:)=[1:1:31];
Alphabeth0(2,:)=['a', 'b', 'c', 'd', 'e', 'f','g', 'h', 'i','j', 'k', 'l','m', 'n', 'o',...
                'p', 'q', 'r','s', 't', 'u','v', 'w', 'x','y', 'z', '_','-', ',', '.','?'];
Alphabeth0(3,:)=[5.856417935 0.915065302 1.7469942850 3.1611134681 8.734714250 ...
                1.622161218 1.688711422 4.999584061  5.814824058  0.116462856 ...
                0.457532651 2.753514683 1.256135097  5.606854671  6.239081607 ...
                0.873471425 0.06655020  3.660261209  5.257466101  7.528491806 ...
                2.104650195 0.856833874 1.514017137  0.124781632  3.105981199 ...
                0.058231428 15.97728974 0.06655020 1.081440812 6.738208136 0.016637550]/100;


%% READ FROM ENCRYPTED NUMERICAL FILE

%EncText(1,:)=load('encrypted_text.txt')+1;
%EncText(1,:)=load('EncryptedText.txt')+1;

%EncText(1,:)=load('encrypted_text.txt')+1;
EncTextTemp=load('encrypted_text.txt')+1;
%EncTextTemp=load('EncMiltonParadiseLost.txt')+1;
EncText(1,:)=EncTextTemp;
EncText(2,:)=Alphabeth0(2,EncText(1,:));

fprintf('Encrypted Text:\n');
fprintf('%c',EncText(2,:))
fprintf('\n__________________________________________\n');

% BUILDING PERMUTED ALPHABETH - STILL HAS TO BE PERMUTED
AlphabethP(1,:)=Alphabeth0(1,:);
AlphabethP(2,:)=Alphabeth0(2,:);
AlphabethP(3,:)=histc( EncText(1,:), unique(Alphabeth0(1,:)) )./length(EncText(1,:));
% conta le occorrenze delle lettere codificate in ordine alfabetico (non permutato)

% SORTING THE PMF IN ORDER TO GUESS THE MAPPING
[pmfp, indp]=sort(AlphabethP(3,:));
[pmfa, inda]=sort(Alphabeth0(3,:));

fprintf('English Alphabeth Sorted By Pmf:\n');
AlphabethSortedByPmf=Alphabeth0(2,inda);
fprintf('%c',AlphabethSortedByPmf);
fprintf('\n__________________________________________\n');

fprintf('Encrypted Text Alphabeth Sorted By Pmf:\n');
EncTextSortedByPmf=Alphabeth0(2,indp);
fprintf('%c',EncTextSortedByPmf);
fprintf('\n__________________________________________\n');

% PERMUTED ALPHABETH CONSTRUCTION - PERMUTATION IN PROGRESS

AlphabethP(2,:)=InterleaveText(Alphabeth0(2,:),Alphabeth0(2,indp),Alphabeth0(2,inda));
% approccio statistico : MAPPING PROBABILISTICI
% e - _
% d - e

% PERMUTED ALPHABETH - PERMUTATION BY MANUAL RECONSTRUCTION
AlphabethPTrue(2,:)=double('nige_pk.ct,xlhvmyujwqr?fsz-dabo');
%AlphabethPTrue(2,:)=double('bcdefghijklmnopqrstuvwxyz_-,.?a');

% approccio manuale : SCAMBI FATTI A MANO
% simboli p i  certi - simboli per cui si scambia
% h - i
% o - .
% n - c
% r - c
% i - s
% g - w
% c - l
% w - m
% n - .
% v - b
% w - y
% y - .
% p - .
% j - -
% x - q

AlphabethP(1,:)=TextToIndices(AlphabethP(2,:),Alphabeth0(1,:),Alphabeth0(2,:));
AlphabethP(3,:)=AlphabethP(3,AlphabethP(1,:));

% COMPUTING THE NUMBER OF PERMUTATION GUESSES
Guesses=AlphabethP(2,AlphabethP(2,:)==double('nige_pk.ct,xlhvmyujwqr?fsz-dabo'));
%Guesses=AlphabethP(2,AlphabethP(2,:)==double('bcdefghijklmnopqrstuvwxyz_-,.?a'));
if(~isempty(Guesses))
GuessesSorted=Alphabeth0(2,sort(TextToIndices(Guesses,AlphabethP(1,:),AlphabethP(2,:))));
end
Guesses0=Alphabeth0(2,AlphabethP(2,:)==double('nige_pk.ct,xlhvmyujwqr?fsz-dabo'));
%Guesses0=Alphabeth0(2,AlphabethP(2,:)==double('bcdefghijklmnopqrstuvwxyz_-,.?a'));

fprintf('Alphabeth Mapping:\n');
fprintf('%c',Alphabeth0(2,:));
fprintf('\n');
fprintf('%c',AlphabethP(2,:));
fprintf('\n');
fprintf('%c',AlphabethPTrue(2,:));
fprintf('\n__________________________________________\n');
fprintf('Guesses: %d\n',length(Guesses));
if(~isempty(Guesses))fprintf('%c',GuessesSorted);end;
fprintf('\n__________________________________________\n');

% DECRYPTION BY USING THE PMF OBTAINED FROM STATISTICS
DecText(2,:)=InterleaveText(EncText(2,:),Alphabeth0(2,:),AlphabethP(2,:));
DecText(1,:)=TextToIndices(DecText(2,:),Alphabeth0(1,:),Alphabeth0(2,:));

```

```
% COMPUTING THE ALPHABETH DISTRIBUTION OF THE SUPPOSED PLAINTEXT
% IT IS DIFFERENT FROM THE ENGLISH ONE

AlphabethD(2,:)=InterleaveText(Alphabeth0(2,:),Alphabeth0(2,:),AlphabethP(2,:));
AlphabethD(1,:)=TextToIndices(Alphabeth0(2,:),Alphabeth0(1,:),Alphabeth0(2,:));
AlphabethD(3,:)=histc(DecText(1,:),unique(AlphabethD(1,:)))./length(DecText(1,:));
char(AlphabethD(2,:))

[pmfd, indd]=sort(AlphabethD(3,:));

fprintf('Decrypted message Alphabeth Sorted By Pmf:\n');
DecSortedByPmf=AlphabethD(2,indd);
fprintf('%c',DecSortedByPmf);
fprintf('\n__________________________________________\n');

fprintf('%c',EncText(2,:));
fprintf('\n');
fprintf('%c',DecText(2,:));
fprintf('\n');
fprintf('\n__________________________________________\n');

% DECRYPTING BY USING MANUAL RECOSTRUCTED PERMUTATION
DecTextTrue(2,:)=InterleaveText(EncText(2,:),Alphabeth0(2,:),AlphabethPTrue(2,:));
DecTextTrue(1,:)=TextToIndices(DecText(2,:),Alphabeth0(1,:),Alphabeth0(2,:));

% COMPUTING THE ALPHABETH DISTRIBUTION OF THE TRUE PLAINTEXT
% IT IS DIFFERENT FROM THE ENGLISH ONE BUT EQUAL TO THE SUPPOSED ONE
AlphabethDTrue(1,:)=Alphabeth0(1,:);
AlphabethDTrue(2,:)=Alphabeth0(2,:);
```

Listing 1: Matlab Implementation

Results

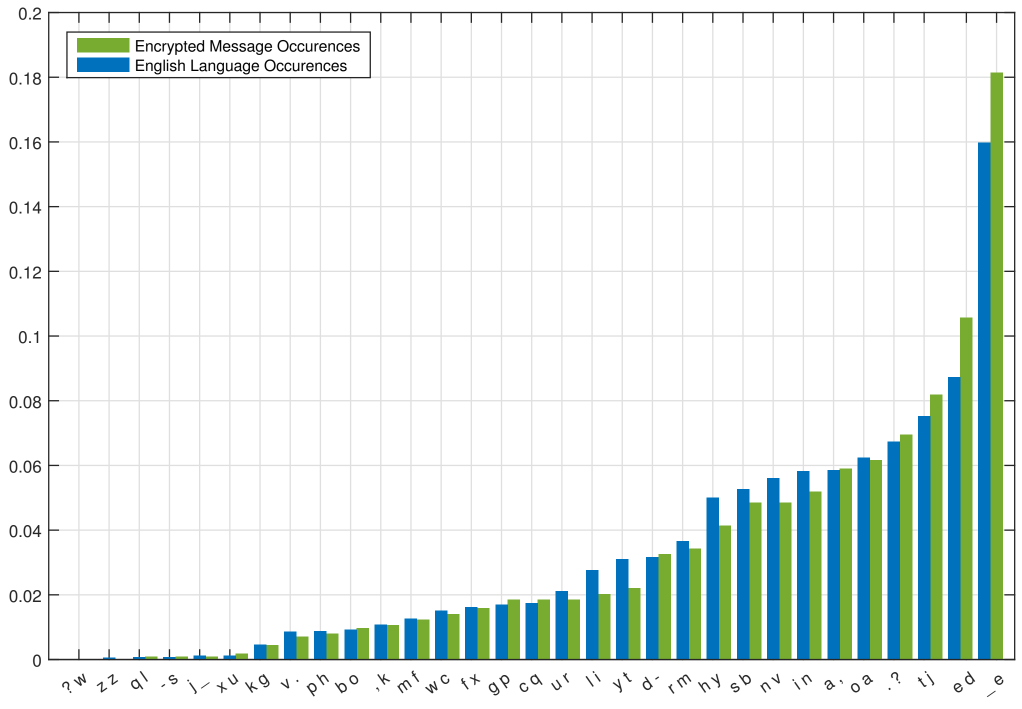

Figure 1: Probability Mass Functions of English Language and Encrypted Text.

The previously shown cryptanalysis formulation was implemented in Matlab®, the output is the following:

```
Encrypted Text:
________________________________________________________________

jndev?p,ayeadodve,mm?td-e,ejv?r.mdeyf?jej?evdp,baeybpfmqej?e,o?b-
ec?bacej?et,ve?odvebjke.di,rydejndqegadtejn,jet,vye-?aujesryjec?e
,t,qkejndqe,vde?amqef?yjf?ad-ej?ey?pd?adedmyduye,-o,aj,cdhejndvdx
?vdkejndqep,-det,vetbjnefnbmbfe,a-e,ajb?inryebaecvddidkebae?v-dve
a?jej?en,odej?exbcnjejndpebaebj,mqhhhejndqeadodvetdaje.qejn,jey,q
bacetnbineq?rei?ayj,ajmqend,vexv?pejndetbyd,ivdye?xe?rve-,qkejn,j
ejbpdend,mye,mmejnbacyhejndqejvryjd-ev,jndvejndbve?taein,v,ijdve,
a-efvr-daide_ega?tbacefdvxdijmqetdmmejn,jejbpdei?aj,bayejndeydd-y
e?xe,mmejnbacykec??-e,yetdmme,ye.,-hebje?rcnjej?e.devdpdp.dvd-ejn
,jejndvdebyea?jnbacep?vde-bxxbirmjej?ej,gdebaen,a-kep?vdefdvbm?ry
ej?ei?a-rijke?vep?vderaidvj,baebaebjyeyriidyykejn,aej?ej,gdejndem
d,-ebaejndebajv?-rijb?ae?xe,eadte?v-dve?xejnbacyhe.di,rydejndebaa
```

```
?o,j?ven,yex?vedadpbdye,mmejn?ydetn?en,ode-?adetdmmera-dvejnde?m-
ei?a-bjb?ayke,a-emrgdt,vpe-dxda-dvyebaejn?ydetn?ep,qe-?etdmmera-d
vejndeadthejnbyei??madyye,vbydyef,vjmqexv?pexd,ve?xejnde?ff?adajy
ketn?en,odejndem,tye?aejndbveyb-dke,a-ef,vjmqexv?pejndebaivd-rmbj
qe?xepdaetn?e-?ea?jevd,-bmqe.dmbdodebaeadtejnbacyerajbmejndqen,od
en,-e,em?acedlfdvbdaide?xejndph
```

```
Alphabeth Sorted By Pmf:
?zq-jxkvpb,mwfgculydrhsniao.te_
```

```
Encrypted Text Sorted By Pmf:
wzls_ug.hokfcxpqrit-mybvn,a?jde
```

```
Alphabeth Mapping:
abcdefghijklmnopqrstuvwxyz_-,.?
oswe_mkplt,qribgcu-yxn?fhzjdav.
```

```
Decoded message Alphabeth Sorted By Pmf:
z?jq-xkvpb,mwfcgulydrhnsiao.te_
```

```
Decrypted Text:

tie_n.gaoh_oeben_arr.yed_a_tn.uvre_hm.t_t._negaso_hsgmrc_t._ab.sd
_w.sow_t._yan_.ben_st,_velauhe_tiec_koey_tiat_yanh_d.oxt_-uht_w._
ayac,_tiec_ane_.orc_m.htm.oed_t._h.ge.oe_erhexh_adbaotawep_tienef
.ne,_tiec_gade_yan_ysti_misrsm_aod_aots.liuh_so_wneele,_so_.nden_
o.t_t._iabe_t._fswit_tieg_so_starcppp_tiec_oeben_yeot_vc_tiat_hac
sow_yisli_c.u_l.ohtaotrc_iean_fn.g_tie_yshealneh_.f_.un_dac,_tiat
_tsge_iearh_arr_tisowhp_tiec_tnuhted_natien_tiesn_.yo_lianalten_a
od_mnudeole_j_ko.ysow_menfeltrc_yerr_tiat_tsge_l.otasoh_tie_heedh
_.f_arr_tisowh,_w..d_ah_yerr_ah_vadp_st_.uwit_t._ve_negegvened_ti
at_tiene_sh_o.tisow_g.ne_dsffslurt_t._take_so_iaod,_g.ne_mensr.uh
_t._l.odult,_.n_g.ne_uolentaso_so_sth_hullehh,_tiao_t._take_tie_r
ead_so_tie_sotn.dults.o_.f_a_oey_.nden_.f_tisowhp_velauhe_tie_soo
.bat.n_iah_f.n_eoegseh_arr_ti.he_yi._iabe_d.oe_yerr_uoden_tie_.rd
_l.odsts.oh,_aod_rukeyang_defeodenh_so_ti.he_yi._gac_d._yerr_uode
n_tie_oeyp_tish_l..roehh_ansheh_mantrc_fn.g_fean_.f_tie_.mm.oeoth
,_yi._iabe_tie_rayh_.o_tiesn_hsde,_aod_mantrc_fn.g_tie_solnedurst
c_.f_geo_yi._d._o.t_neadsrc_versebe_so_oey_tisowh_uotsr_tiec_iabe
_iad_a_r.ow_eqmenseole_.f_tiegp
```

Then we proceeded by manually adjusting the permutation to get a better estimation:

```
Alphabeth Mapping:
abcdefghijklmnopqrstuvwxyz_-,.?
nige_pk.ct,xlhvmyujwqr?fsz-dabo
```

```
Decoded message Alphabeth Sorted By Pmf:
z?jx-qkb.v,pgfmuycwdlsirhanote_
```

```
Decrypted Text:

the_romans_never_allowed_a_trouble_spot_to_remain_simply_to_avoid
_going_to_war_over_it,_because_they_knew_that_wars_donqt_just_go_
away,_they_are_only_postponed_to_someone_elseqs_advantage._theref
ore,_they_made_war_with_philip_and_antiochus_in_greece,_in_order_
not_to_have_to_fight_them_in_italy..._they_never_went_by_that_say
ing_which_you_constantly_hear_from_the_wiseacres_of_our_day,_that
_time_heals_all_things._they_trusted_rather_their_own_character_a
nd_prudence_-_knowing_perfectly_well_that_time_contains_the_seeds
_of_all_things,_good_as_well_as_bad._it_ought_to_be_remembered_th
at_there_is_nothing_more_difficult_to_take_in_hand,_more_perilous
_to_conduct,_or_more_uncertain_in_its_success,_than_to_take_the_l
ead_in_the_introduction_of_a_new_order_of_things._because_the_inn
ovator_has_for_enemies_all_those_who_have_done_well_under_the_old
_conditions,_and_lukewarm_defenders_in_those_who_may_do_well_unde
r_the_new._this_coolness_arises_partly_from_fear_of_the_opponents
,_who_have_the_laws_on_their_side,_and_partly_from_the_incredulit
y_of_men_who_do_not_readily_believe_in_new_things_until_they_have
_had_a_long_experience_of_them.
```

Further Comments

First of all, we could observe that Ceaser ciphering encryption scheme is very weak, it can be broken easily by using occurrence analysis.
One could expect that the Permutation of the alphabet makes it stronger, since the permutation operation may be more difficult to invert.
This is not so true, since, from what we have seen, even if the permutation is applied, the probability mass function is preserved, but the order is not the same.
If we consider the sorted pmf, they look a lot like each other.
The estimation of the permutation rule $\hat{\Pi}(\cdot)$ is as much close to the true one $\Pi(\cdot)$, as the pmf of the encrypted message $P(\mathcal{E})$ is close to the English one $P(\mathcal{T})$.
In order to reach easily a good estimation, one may think to try clever character swappings, and use the Kullback-Leibler Divergence as a metric for the implementation of an algorithm that hopefully converges to the right computation of the permutation rule.
In order to make the system more robust, one may think to let the text be stronger to occurrences analysis attacks. For example, one thing that could be done is to substitute a given character (maybe one of the most recurrent) with a predefined character, (by setting proper rules) in order to let the encrypted text's pmf be uniform.
By doing this, the encrypting system may, in our opinion, be more robust to this kind of attacks.
Moreover, in order to try if the pmf of the encrypted message makes it weak - in the sense of occurrence analysis attacks - one may think to compress the message: if the pmf is uniform enough, the size of the message will not decrease that much, and this may be helpful for its security.

Homework 2 - LFSR Stream Ciphering Encryption using key encrypted with ElGamal PK based on Elliptic Curves

Assignment

Decrypt the message:

```
1, 1, 0, 1, 0, 0, 0, 1, 1, 0, 0, 1, 1, 1, 0, 0, 1, 0, 1, 0, 0, 1,
1, 1, 1, 0, 0, 0, 0, 1, 1, 0, 1, 1, 0, 0, 1, 0, 1, 0, 1, 0, 0, 0,
1, 1, 0, 1, 0, 1, 0, 1, 1, 1, 1, 1, 1, 0, 0, 0, 1, 0, 0, 1, 1, 0,
0, 0, 1, 0, 0, 0, 1, 0, 1, 1, 1, 0, 1, 0, 0, 0, 1, 1, 0, 1, 1, 1,
1, 0, 0, 1, 0, 1, 0, 0, 0, 0, 0, 1, 1, 0, 0, 0, 0, 1, 1, 0, 1, 1,
1, 0, 0, 1, 0, 1, 0, 0, 0, 0, 0, 1, 1, 0, 0, 0, 1, 0, 1, 1, 1 ...
```

that **Alice** sent to **Bob** encrypted using stream cipher.

The stream generator consists of a pair of binary LFSRs followed by an output NLF defined over $\mathbb{F}_2$.

Fibonacci LFSR:

$$F(x) = x^{23} + x^{20} + x^{19} + x^{15} + x^{12} + x^{8} + x^{7} + x^{6} + x^{5} + x^{4} + x^{2} + x + 1$$

Galois LFSR:

$$G(x) = x^{23} + x^{21} + x^{19} + x^{18} + x^{17} + x^{15} + x^{12} + x^{10} + x^{9} + x^{6} + x^{5} + x^{3} + 1$$

Output:

$$NLF = x_2 \cdot x_{10} \cdot y_7 \cdot y_{16} + x_1 \cdot x_5 + y_1 \cdot x_7 + y_{23} \bmod 2$$

where the variables x_i are taken from the Fibonacci LFSR and the variables y_i from the Galois LFSR. Both LFSRs have the same initial state (from left to right) which is specified by a binary secret key represented, for its encryption, as an integer:

$$K_s = b_1 + b_2 2 + b_3 2^2 + \cdots + b_{19} 2^{18}$$

K_s was encrypted using the ElGamal public-key system based on the elliptic curve defined over $\mathbb{F}_p$ as:

$$y^2 = x^3 + x^2 + 50x + 251$$

Known informations about the Elliptic curve are:

Modulo:	$p = 186733$	Message key:	$P_B = (64236, 69780)$
Generator point:	$P_0 = (15, 24353)$	Alice's secret key:	$ASK = [0110101000111011]$;

$$\text{Encrypted Secret Key:} \quad ek = K_s \cdot X_{com} \bmod p = 83533$$

Problem Statement

The stream ciphering system works with two different LFSR schemes: the Fibonacci and Galois ones. A subset of their memory cell's output is used to generate a stream by non linear combination whose rule is defined by a Non Linear Function (NLF).
Such a stream is then used as ciphering key for the Plain Text Message.
The two LFSRs are both initialized by the same starting state K_s that Alice sends to Bob by using elliptic curves based ElGamal Encryption system.

Bob	**Channel**	**Alice**
- Chooses his private key $BSK < p$		
- Computes his public key $P_B = BSK \cdot P_0$,	$P_B \rightarrow$	- Retrieves Bob's public key P_B
		- Chooses her private key $ASK < p$
		- Computes $P_{AB} = ASK \cdot P_B$
		- Takes only the coordinate X_{com} from $P_{AB} = (X_{com}, Y_{com})$
- Retrieves Alice's secret key P_A and the encrypted key ek	$\leftarrow P_A$ $\leftarrow ek$	- Encrypts the message by using the Encryption rule: $ek = K_s \cdot X_{com}$
- Computes $P_{BA} = BSK \cdot P_A = P_{AB}$		
- Takes only the coordinate X_{com} from $P_{AB} = (X_{com}, Y_{com})$		
- Decrypts the message inverting the Encryption rule: $K_s = \frac{ek}{X_{com}} \bmod p$		

The system works with the following known parameters:

- Elliptic curve equation:	$y^2 = x^3 + x^2 + 50x + 251$
- Order of the Group $\mathbb{F}_p$ of use:	$p = 186733$
- Initial Point:	$P_0 = (15, 24353)$

As a general convention, the Public Keys P_A and P_B are not sent over the channel but are retrievable from a shared directory.
The security of the system is provided by the elliptic curve's implementation of ElGamal public key system. If the plain starting state of the LFSRs was sent over the channel, the ciphering stream would be weaker.
Implementing Elliptic curves cryptosystem, whatever information sent over the channel is more protected, since its strength requires large primes factorization.

Elliptic Curves Implementation

Starting from the general equation for Elliptic curves over $\mathbb{F}_\mathbf{p}$:

$$E(x,y): \; y^2 + a_1xy + a_3y = x^3 + a_2x^2 + a_4x + a_6 \tag{1}$$

We may define the Doubling rule and the Addition rule for points over the elliptic curve.

Doubling rule : $\mathbf{2(x_1, y_1) = (x_3, y_3)}$

Intercepting the line with equation: $(y - y_1) = m(x - x_1)$.

$$\begin{cases} y = m(x - x_1) + y_1 \\ y^2 + a_1xy + a_3y = x^3 + a_2x^2 + a_4x + a_6 \end{cases} \tag{2}$$

Leads to the following derivation:

$$(m(x-x_1)+y_1)^2 + a_1x(m(x-x_1)+y_1) + a_3y \;=\; x^3 + a_2x^2 + a_4x + a_6$$

$$m^2(x-x_1)^2 + y_1^2 + 2m(x-x_1)y_1 + a_1x^2m - a_1xmx_1 + a_1xy_1 + a_3y = x^3 + a_2x^2 + a_4x + a_6$$

$$m^2x^2 + m^2x_1^2 - 2m^2xx_1 + y_1^2 + 2mxy_1 - 2mx_1y_1 + a_1x^2m - a_1xmx_1 + a_1xy_1 + a_3y = x^3 + a_2x^2 + a_4x + a_6$$

$$x^3 - (m^2 + a_1m - a_2)x^2 + (2m^2x_1 - 2my_1 + a_1mx_1 - a_1y_1 + a_4)x - (m^2x_1^2 + y_1^2 - 2mx_1y_1 + a_3y - a_6) = 0 \tag{3}$$

from Vieta's formulas, we know that, a Polynomial of degree 3: $p(x) = ax^3 + bx^2 + cx + d$, with roots x_1, x_2, x_3 has coefficient $b = x_1 + x_2 + x_3$. From this, we have:

$$m^2 + a_1m - a_2 = x_1 + x_2 + x_3 \tag{4}$$

The roots of the polynomial in Equation (3) belong to three points on the Elliptic Curve: x_1, x_2 are the abscissa of the point we are doubling, x_3 is the abscissa of the point we are looking for.
All we need to know is m, that can be obtained by derivation, since it represents the gradient of the tangent of the line in the point where it intersects the curve:

$$\frac{\partial y}{\partial x} = \frac{\partial E(x,y)/\partial x}{\partial E(x,y)/\partial y} = \frac{3x^2 + 2a_2x + a_4 - a_1y}{2y + a_1x + a_3}. \tag{5}$$

The actual value of m has to be computed by evaluating the gradient in the point (x_1, y_1):

$$m = \left.\frac{\partial y}{\partial x}\right|_{x=x_1, y=y_1} = \frac{3x_1^2 + 2a_2x_1 + a_4 - a_1y_1}{2y_1 + a_1x_1 + a_3} \bmod p \tag{6}$$

Knowing m, from Equation (4) we may think to solve for x_3, y_3:

$$\begin{cases} x_3 = m^2 + a_1 m - a_2 - 2x_1 \bmod p \\ y_3 = -(y_1 + m(x_3 - x_1)) \bmod p \end{cases} \tag{7}$$

We solve the intersection of the line with the equation of the elliptic curve.

Addition rule : $(\mathbf{x_1}, \mathbf{y_1}) + (\mathbf{x_2}, \mathbf{y_2}) = (\mathbf{x_3}, \mathbf{y_3})$

At the same way, it is done by intercepting the line with equation: $(y - y_1) = m(x - x_1)$. The Derivation (3) is still valid, hence the Equation (4) leads to the computation of x_3. This time m is known from the equation of the line crossing two points:

$$m = \frac{y_2 - y_1}{x_2 - x_1} \bmod p \tag{8}$$

Knowing m, we solve again for x_3, y_3:

$$\begin{cases} x_3 = m^2 + a_1 m - a_2 - x_1 - x_2 \bmod p \\ y_3 = -(y_1 + m(x_3 - x_1)) \bmod p \end{cases} \tag{9}$$

We solve the intersection of the line with the equation of the elliptic curve.

Rules of Interest

In our specific case, work over $\mathbb{F}_{186733}$ on the curve of equation:

$$E(x, y) : \; y^2 = x^3 + x^2 + 50x + 251 \tag{10}$$

That's what we had before with: $a_1 = 0, \quad a_2 = 1, \quad a_3 = 0, \quad a_4 = 50, \quad a_6 = 251.$

$$\begin{array}{cc} \textbf{Doubling: } \mathbf{2(x_1, y_1) = (x_3, y_3)} & \textbf{Addition: } \mathbf{(x_1, y_1) + (x_2, y_2) = (x_3, y_3)} \\ \begin{cases} m = \dfrac{3x_1^2 + 2x_1 + 50}{2y_1} & \bmod 186733 \\ x_3 = m^2 - 1 - 2x_1 & \bmod 186733 \\ y_3 = -(y_1 + m(x_3 - x_1)) & \bmod 186733 \end{cases} & \begin{cases} m = \dfrac{y_2 - y_1}{x_2 - x_1} & \bmod 186733 \\ x_3 = m^2 - 1 - x_1 - x_2 & \bmod 186733 \\ y_3 = -(y_1 + m(x_3 - x_1)) & \bmod 186733 \end{cases} \end{array} \tag{11}$$

The encryption key can be found by considering the rules of interest for Alice's secret key ASK starting from the point P_m.

$$P_{AB} = \sum_{i=0}^{15} 2^i \cdot ASK(i) \cdot P_B = 56406 \cdot P_B$$

The computation can be derived in a easier way by applying iteratively the doubling rule, then summing the points.

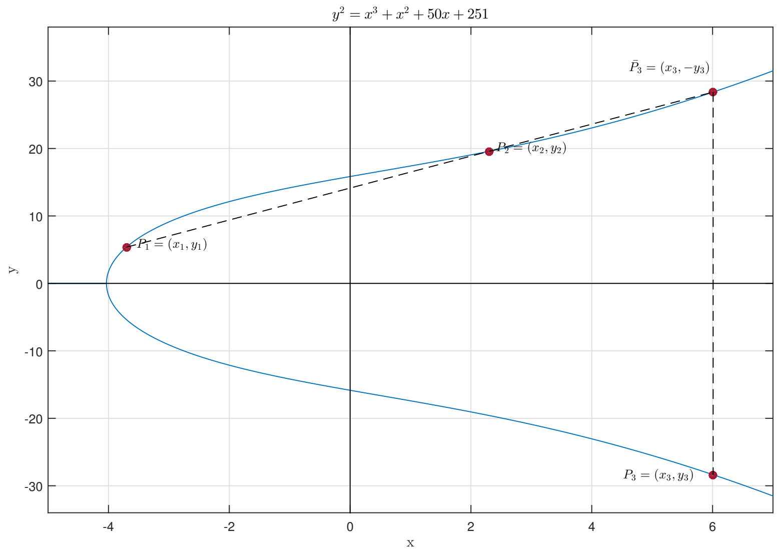

Figure 2: Elliptic Curve Additon Rule

$$\begin{aligned}
2^0 \cdot P_B &= (64236, 69780) \\
2^1 \cdot P_B &= 2 \cdot (64236, 69780) &&= (48020, 125549) \\
2^2 \cdot P_B &= 2 \cdot (48020, 125549) &&= (175330, 47928) \\
2^3 \cdot P_B &= 2 \cdot (175330, 47928) &&= (75937, 71730) \\
2^4 \cdot P_B &= 2 \cdot (75937, 71730) &&= (72852, 138370) \\
2^5 \cdot P_B &= 2 \cdot (72852, 138370) &&= (63618, 161) \\
2^6 \cdot P_B &= 2 \cdot (63618, 161) &&= (56218, 93364) \\
2^7 \cdot P_B &= 2 \cdot (56218, 93364) &&= (12054, 68385) \\
2^8 \cdot P_B &= 2 \cdot (12054, 68385) &&= (64182, 46455) \\
2^9 \cdot P_B &= 2 \cdot (64182, 46455) &&= (50647, 66194) \\
2^{10} \cdot P_B &= 2 \cdot (50647, 66194) &&= (23960, 824) \\
2^{11} \cdot P_B &= 2 \cdot (23960, 824) &&= (78063, 149336) \\
2^{12} \cdot P_B &= 2 \cdot (78063, 149336) &&= (135873, 15725) \\
2^{13} \cdot P_B &= 2 \cdot (135873, 15725) &&= (2026, 123847) \\
2^{14} \cdot P_B &= 2 \cdot (2026, 123847) &&= (15708, 84839) \\
2^{15} \cdot P_B &= 2 \cdot (15708, 84839) &&= (176136, 128991)
\end{aligned}$$

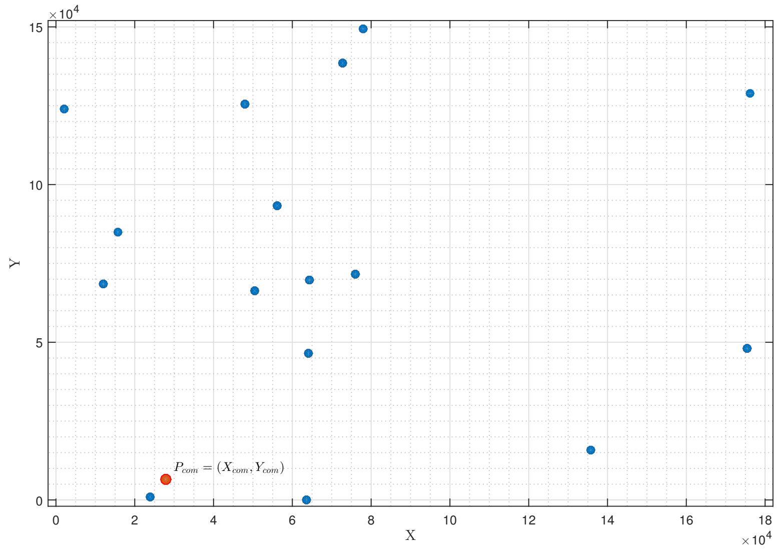

Figure 3: Set of points obtained by doubling P_B represented over $\mathbb{F}_p^2$.

$$
\begin{aligned}
P_{AB} &= (X_{com}, Y_{com}) = 56406 \cdot P_B \\
&= (2 + 4 + 16 + 64 + 1024 + 2048 + 4096 + 16384 + 32768) \cdot P_B \\
&= 2^1 \cdot P_B + 2^2 \cdot P_B + 2^4 \cdot P_B + 2^6 \cdot P_B + 2^{10} \cdot P_B \\
&\quad +2^{11} \cdot P_B + 2^{12} \cdot P_B + 2^{14} \cdot P_B + 2^{15} \cdot P_B \\
&= (27910, 6518)
\end{aligned}
$$

Once found the value assumed by $X_{com} = 27910$, we could retrieve the initial state for the LFSR K_s, by inverting its relationship with ek.

$$K_s = \frac{ek}{X_{com}} \bmod p = 116211,$$

Finding the inverse of X_{com} in mod$p$ was made by solving Diophantine equations.
To solve the equation:

$$X_{com} \cdot X_{com}^{-1} = 1 \bmod p$$

We just considered that X_{com} and p are relatively prime: $\gcd(X_{com}, p) = 1$,
They satisfy Bézout's identity: $uX_{com} + vp = 1$ where $u, v \in \mathbb{F}_p$ can be easily computed.
Since the operations are over $\mathbb{F}_p$, the previous equation becomes: $u \cdot X_{com} = 1 \bmod p$,
hence the value u that we find using gcd is equal to $X_{com}^{-1} = u$.

LFSR Implementation

Once we have found the starting state $K_s = 116211$ in decimal form by decrypting Alice's secret message ek, we need to read it in binary form in order to use it as starting state for LFSRs, padding it to fit the size of the whole registers.

$$K_s = (116211)_2 = b_1 + b_2 \cdot 2 + b_3 \cdot 2^2 + \cdots + b_{19} \cdot 2^{18} + \cdots b_{23} \cdot 2^{22};$$

$$\bar{x}(0) = \{b_i\}_{i=1}^{23} = [\ 1,\ 1,\ 0,\ 0,\ 1,\ 1,\ 1,\ 1,\ 1,\ 0,\ 1,\ 0,\ 0\ ,0\ ,1\ ,1\ ,1\ ,0\ ,0\ ,0\ ,0\ ,0\ ,0\ ]^T;$$

$$\bar{y}(0) = \{b_i\}_{i=1}^{23} = [\ 1,\ 1,\ 0,\ 0,\ 1,\ 1,\ 1,\ 1,\ 1,\ 0,\ 1,\ 0,\ 0\ ,0\ ,1\ ,1\ ,1\ ,0\ ,0\ ,0\ ,0\ ,0\ ,0\ ]^T;$$

We implemented LFSRs shift by using matrices represent1ation:

$$\mathbf{A}_F = \begin{pmatrix} 0 & 1 & 0 & 0 & \cdots & 0 \\ 0 & 0 & 1 & 0 & \cdots & 0 \\ 0 & 0 & 0 & 1 & \cdots & 0 \\ 0 & 0 & 0 & 0 & \cdots & 0 \\ 0 & 0 & 0 & 0 & \cdots & 0 \\ \vdots & \vdots & \vdots & \vdots & \ddots & \vdots \\ a_m & a_{m-1} & a_{m-2} & a_{m-3} & \cdots & a_1 \end{pmatrix} \quad \mathbf{A}_G = \begin{pmatrix} 0 & 0 & 0 & 0 & \cdots & a_m \\ 1 & 0 & 0 & 0 & \cdots & a_{m-1} \\ 0 & 1 & 0 & 0 & \cdots & a_{m-2} \\ 0 & 0 & 1 & 0 & \cdots & a_{m-3} \\ 0 & 0 & 0 & 1 & \cdots & a_{m-4} \\ \vdots & \vdots & \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & 0 & 0 & \cdots & a_1 \end{pmatrix}$$

By doing the following operation, we get iteratively the state update $\bar{x}(n)$ for each state transition n so that we generate a stream of length N which is the size of the encrypted message in binary form.

$$\bar{x}(n) = \mathbf{A}_F \cdot \bar{x}(n-1) \qquad \bar{y}(n) = \mathbf{A}_G \cdot \bar{y}(n-1)$$

Once we have generated the output streams: $\bar{x}(n)$, $\bar{y}(n), \forall n = 0, \cdots, N-1$, we have applied the NLF function by by combining a subset of elements of each stream by the following rule:

$$NLF(\bar{x}(n), \bar{y}(n)) = x_2(n) \cdot x_{10}(n) \cdot y_7(n) \cdot y_{16}(n) + x_1(n) \cdot x_5(n) + y_1(n) \cdot x_7(n) + y_{23}(n);$$

At the end, Bob can finally obtain the plaintext binary stream in order to read the Plain Text Message as shown in Figure .

$$E(n) = NLF(n) \oplus M(n)$$

is inverted easily by doing:

$$M(n) = NLF(n) \oplus E(n)$$

Since a simple 5 bits binary coding rule is applied to the plain text in character form, Bob has to read blocks of 5 bits and decode them using the relative mapping table.

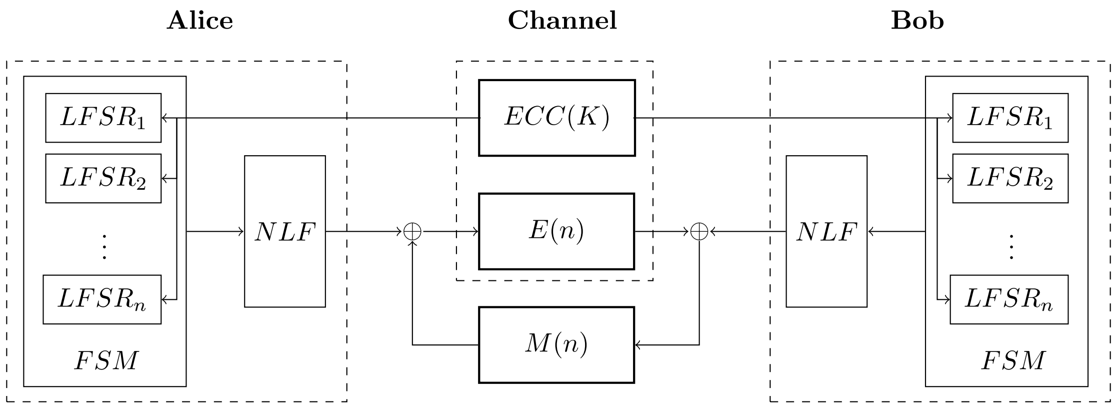

Figure 4: Stream Ciphering with Key encryption basic model

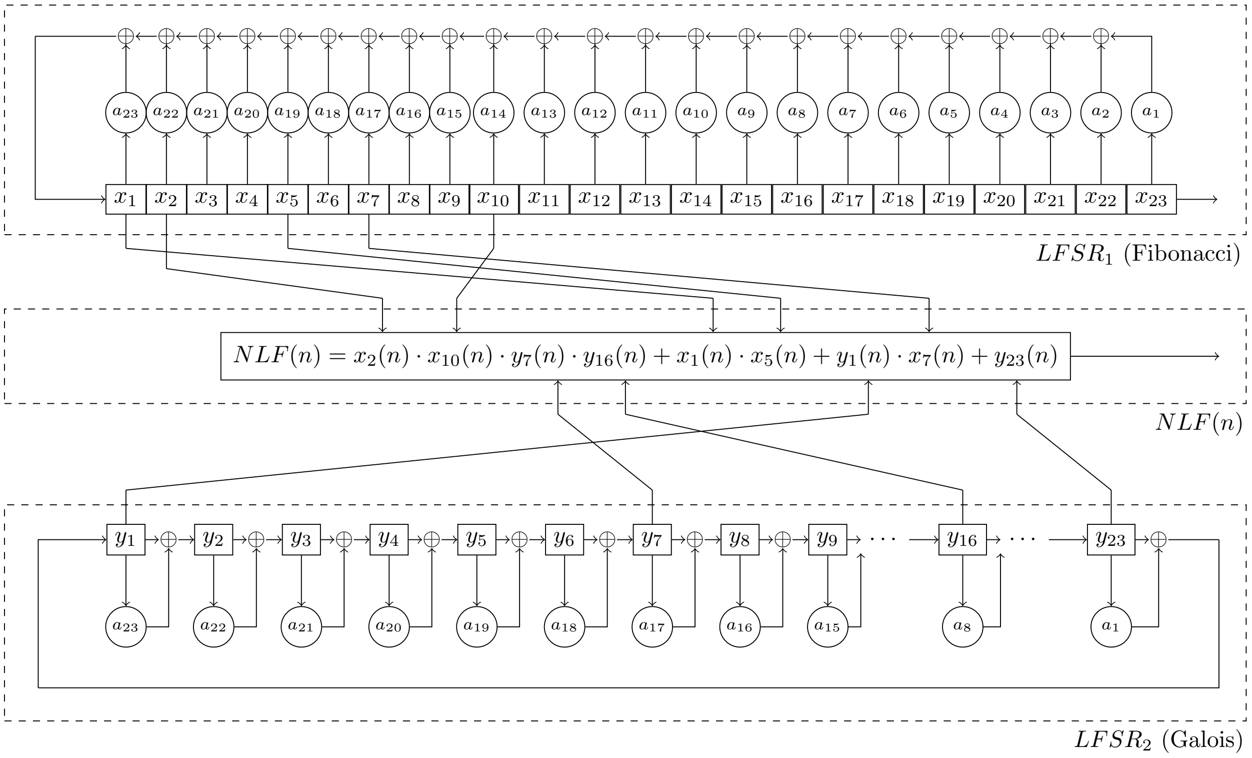
x_1 … x_{23}
a_{23} … a_1
$LFSR_1$ (Fibonacci)
$NLF(n) = x_2(n) \cdot x_{10}(n) \cdot y_7(n) \cdot y_{16}(n) + x_1(n) \cdot x_5(n) + y_1(n) \cdot x_7(n) + y_{23}(n)$
$NLF(n)$
y_1 … y_9 … y_{16} … y_{23}
$LFSR_2$ (Galois)

Figure 5: NLF Fibonacci, LFSR Galois and NLF visualization.

Results

The Output of the NLF function is the following:

```
0 1 0 0 1 1 0 1 1 1 1 1 0 1 0 0 1 1 1 0 1 0 1 0 1 0 0 1 0 1 1 0 1 0 1 1 1 1 1 0 1 0 0
1 1 1 0 1 0 1 1 1 1 0 0 1 0 0 0 0 1 1 1 0 1 0 1 1 1 1 1 1 1 1 1 1 0 1 1 0 0 0 1 1 0 1
0 1 0 0 0 1 1 1 0 1 0 0 1 1 0 1 0 1 0 1 1 1 1 0 1 1 1 0 0 0 0 0 0 1 1 0 1 1 0 0 0 0 1
1 1 0 1 1 1 1 1 0 0 0 1 0 0 1 1 0 0 1 0 0 0 1 1 1 1 0 1 0 1 0 0 0 1 0 0 0 1 0 1 0 0 0
0 1 0 1 0 0 1 0 0 0 1 1 1 1 0 1 0 1 0 1 0 0 1 1 1 1 1 0 1 1 0 0 0 1 0 0 1 1 0 1 0 1 1
0 0 0 0 0 1 1 1 1 1 1 0 1 1 1 0 1 0 0 1 0 1 1 1 0 1 0 0 1 0 0 0 1 0 0 1 1 1 1 0 1 0 1
0 0 1 0 0 1 0 1 0 1 0 0 1 1 1 1 1 0 0 0 0 1 1 1 0 1 0 0 1 0 0 0 1 1 0 1 1 0 0 0 0 0 0
1 1 1 1 0 0 0 0 1 0 0 0 1 1 1 0 0 0 0 0 0 1 1 0 1 0 0 1 0 1 1 1 1 1 0 1 0 0 0 0 0 1 0
1 0 0 0 0 0 0 1 1 1 1 1 1 1 1 0 0 0 1 1 0 1 0 1 0 1 1 0 1 0 1 1 0 0 1 0 1 1 0 1 0 0 0
1 0 1 1 0 0 1 0 1 0 1 0 0 1 1 1 1 0 1 0 1 1 1 1 0 0 0 0 1 1 1 0 1 1 1 0 0 1 1 0 0 0 1
1 0 1 0 1 0 1 0 0 0 1 1 1 1 1 0 1 0 1 0 0 1 0 0 0 1 0 1 0 0 1 0 1 0 0 1 1 1 0 1 0 1 0
0 1 1 1 0 1 0 0 0 1 1 0 0 0 1 0 0 0 1 1 0 0 0 0 0 1 1 0 0 1 1 0 1 1 1 0 1 1 0 1 0 0 1
0 0 1 0 0 0 0 1 1 0 1 0 1 1 0 1 0 1 0 0 0 1 0 1 0 1 1 1 1 1 0 0 1 1 0 1 0 1 1 0 0 1 0
0 0 1 1 0 0 1 0 0 0 0 0 1 0 0 1 1 1 0 1 0 0 0 0 0 0 1 0 1 0 0 0 1 1 0 0 1 1 0 0 1 1 1
1 1 0 0 1 0 0 1 0 0 1 1 0 0 0 0 0 1 1 0 1 0 0 0 1 0 1 1 0 0 1 0 0 0 1 0 0 1 0 0 1 0 1
1 0 0 0 0 1 0 1 1 1 1 1 0 0 1 0 0 1 0 1 1 1 1 0 0 1 1 0 0 0 0 1 0 0 0 1 0 1 0 1 1 1 1
1 0 0 0 0 0 1 1 0 1 1 1 1 0 0 1 1 0 1 0 0 0 1 1 0 0 1 1 0 1 0 1 0 0 1 0 0 0 1 1 0 0 0
0 0 0 1 0 1 0 0 1 0 0 0 0 0 1 0 1 1 0 0 1 1 1 1 1 1 1 1 0 0 1 0 0 1 1 1 0 1 1 1 1 1 0
1 0 0 1 0 1 0 0 1 1 1 1 0 0 0 0 1 0 1 1 1 0 0 0 0 0 1 1 1 1 1 0 1 0 1 0 1 1 1 0 1 1 0
1 0 0 1 1 0 0 1 0 0 1 1 0 1 0 1 0 1 0 0 0 0 0 1 1 1 0 1 1 1 1 0 1 1 1 1 1 1 0 0 0 0 1
1 0 0 1 0 1 0 1 0 1 1 1 0 1 1 1 0 1 0 1 1 0 1 0 0 1 0 0 0 1 1 1 1 0 0 0 1 0 1 1
```

The final output of the decryption is:

"true education makes for inequality: the inequality of individuality, the inequality of success, the glorious inequality of talent, of genius. felix e. schelling american educator."