***Notes on cryptanalysis of the Hill Cipher***

*Jeudi 25/02:*

Beginning of the lecture first done by Alina Matyukhina.

***Intro:***

So the basics of Hill cipher is the separation of the plain text in blocks of size *d*, and to multiply each block by the key which is a matrix *d x d*. As this cipher is linear, it is easy breakable.

Hill cipher:

The plain text space is defined by the set of ALL meaningful English strings of length multiple of an integer d. (the length of the blocks). Each characters belong to Z/26Z.

To encode and get the cipher text C we do C = K \* X (where K is the key and X the plaintext) the whole modulo 26.

To decode as the matrix K of the Key is invertible, we just get X = K^-1 \* C and modulo 26 again.

So if you intercept d^2 pair of C/P you can easily do a mapping and find the matrix K.

Note on brute force*:*

To do an exhaustive search on the whole space of matrix K, you need 26^(d²) matrix multiplication because for each letter of the alphabet you need to test a matrix that is of size *dxd,* and then chose those who are meaningful when you multiply so O(d^3\*26^(d²)). Work with cipher text length = 1.27*d².*

Improvement of the brute force attack:

We can use the divide and conquer method instead of using the full redundancy of English language, so we could get to O(d^3\*26^(d²)/d!) as we didn’t get the good one but permutation, so it leads to O(d²\*26^(d²)/log(d)). Work with cipher text length at least 8.96d² - 0(log(d)).

Then we would get to O(d\*26^(d)) by removing repetitive calculations with a precomputation time of O(d²)

Last improvement:

With the Chinese Reminder Theorem, we could get to a final complexity of O(d\*13^d) (it applies as we are in Z/26Z) , but the length of the cipher text in the divide and conquer must be λ\*n.

***Goal of this project:***

So now the goal of this project is to present new cipher text-only attack. We first recover matrix K in Z/2Z. Observe the probability distribution of P in Z/2Z, or the bias of it. Then we proved that the largest bias (λ.X) can be obtained with *weight(*λ) = 1.

Preliminary mathematical concepts and application:

So we can first reduce due to the Chinese Remainder Theorem as Z/26Z = Z/2Z X Z/13Z.

If we take the characteristic function of the random variable X we get:\begin{align}
\phi_{X}(t)&=\mathbb{E}\left[e^{itX}\right]\\
&=\mathbb{E}\left[\cos (tX)\right]+i\ \mathbb{E}\left[\sin (tX)\right].
\end{align}


And bias(P) = ϕ(2π/p) in Z/pZ. We got p = 2 in our case and end up with E [(-1) ^X)]. By the formula of the expectation for a discrete random variable we end up with:

E [(-1) ^X)] = (-1)^0 Pr(X=0) + (-1) Pr(X=1) = Pr(X=0) - Pr(X=1) = 2Pr(X=0) – 1.

*Vendredi 26/02:*

Key recovery modulo 2:

*Note:* λ *is a fixed vector*

We first assume that all letters are independent and identically distributed in Z/2Z, and so λ.X = λ1.X1 + λ2.X2+ ...

A lemma (proof easy with the E [(-1) ^X)]) tells us that for a bloc of letters X taken from a random text bias(λ.X) = ε^(weight(λ)) , where the weight(λ) is the number or coordinates in vector λ with value 1. Where ε is the bias of a letter.

If the vector λ is null, we got a trivial solution of bias(λ.X)=1 , it’s the largest because the bias is bound to [-1,1].

We clearly see that the bias has exponential decay and so the largest non-trivial bias(λ.x) is with weight(λ)=1.

Another lemma use now the matrix K (so the cipher block at the end to be exact)

We have Y = K\*X , the d nonzero vectors μ with largest bias(μ.Y) are the colums of (K^T)^-1

Then by an easy proof we just get that this bias is equal to the previous one and thus the largest non-trivial bias(μ.Y) are obtain by μ= (K^T)^-1 .λ with weight(λ) = 1.

Recovering Key Matrix:

If we take a lot of cipher texts: Y1, … ,Yn , we can now with the previous relation find the d vectors μ such that the bias is the largest. The problem is to compute the vectors μ given that many sample vectors Y.

We can compute μ.Y1 , … , μ.Yn , and do the mean , if n is large engough so that bias(μ.Y) by the Hoeffding’s bounds. If we take Sn = = n.bias(μ.Y) = n

We know recall that is tge number of coordinates of vector with value 1, if they are > 1 let’s called it bad , and the number is

Now we take X1, … , Xn independent random variables. They are almost surely bounded (assumption) meaning that Pr(Xi [ai,bi])= 1.

We define Sn to be the sum of Xi and we get:

*Dimanche 28/02:*

The probability of fail of algorithm 1 part 1 can be calculated as following:

Pr(Sn(

With m = -

Probability of fail stays:

Pr(fail)

In our case we are in Z/2Z so amplitude is [-2 ;2] = 4, and #samples = n , #good = d , #bad and gap

Now if we take back μi=.λj , where the λj are unit vector colum and μi are obtained from Algorithm 1.

If we do . We see that this result gives only one letter of , indeed is a ligne vector with only one coordinate to 1.

Finding the correct order of vectors in Key Matrix:

To do this we’ll find the first one and the last one and find the others consectutively.

We assume that a pair of letters with same position of difference l from a random text are independent and identically distributed. We define pl to be the probability that this pair is (0,0).

If we take to indices I and I’, we have μi=.λj and μi’= .λj’ , so are 2 letters in at position j and j’.

Now there is 2 possibilites either they are at 1 from each other or at a distance l.

If . So the probability is called p1

When j’=j+l, the probability is pl

Assume we know vectors μi1 , μi2 ,… , μit-1 , to get μt you need to compute n00(i) = and take it such tat n00(it) is minimum.

Then we calculate the probability the proba to fail in this case and get formula

Getting First and Last vectors:

So to resume it we do n00(I,I’) and look at the first pair witth lowest n00 it should be the last , first vector pair.

***Algorithm 1 details:***

So it’s in 3 parts, first you find the element of μ thanks to the cipher texts and the mean.

Then you retrieve it in the right order and find in the meantime the first and last element of the set

And eventually you do iterations to find all the other μ vectors.

Complexity:

So we can improve this by doing a FFT , instead of Sn = we can do , where ny= #{k,}.

It’s more or less the number of time k that a value y is taken by the random variable . From this we have a total complexity of O(d\*.

Key recovery modulo 26: