

Inhoud

Deze map bevat de oplossingen voor de opgave van Codetheorie voor 2024-2025.

We hebben het werk verdeeld als volgt: - Vigenère: Tibo Verreycken - ADFGVX: Jesse Daems - Playfair: Daria Matviichuk - Enigma: Tibo Verreycken, Jesse Daems, Daria Matviichuk

ADFGVX

Alle code bestaat uit 3 aparte python bestanden, uit te voeren in de adfgvx map zonder argumenten

Om deze code te ontcijferen heb ik in drie stappen gewerkt.

1. Mogelijkheden transpositie uitschrijven

We hadden gekregen dat de sleutel maximaal van lengte 10 zou zijn. Daarom waren er voor de kolomtranspositie slechts $\sum_{k=1}^{10} k! = 4.037.913$ mogelijkheden: voor elke mogelijke sleutellengte de mogelijke volgordes van de kolommen.

Als eerste idee heb ik geprobeerd deze mogelijkheden allemaal te overlopen en de digramfrequenties uit te schrijven naar een bestand om later te kunnen analyseren. Dit bleek ineens al goed genoeg te werken (zie volgende stap)

Met frequency_calculator.py doe ik dit in een 50tal minuten op mijn PC. Het resultaat is een bestand van 1,51GB.

2. Digramfrequenties transpositie onderzoeken

Bij het onderzoeken van de digramfrequenties heb ik rekening gehouden met het volgende: - Bij een juiste transpositie zou de hoogste frequentie veel hoger zijn dan bij de meeste foute transpositions, die een meer gelijkmatige verdeling zouden produceren. - Bij een juiste transpositie zou het ook logisch zijn dat er bepaalde digrammen niet voorkwamen (bv. het is onwaarschijnlijk dat in een literaire tekst, alle getallen zouden voorkomen).

In enkele minuten vind ik met frequency_analyzer.py vier ordeningen waarbij de maximale frequentie mijn *threshold* overschrijdt. De threshold is experimenteel bepaald. Van de vier ordeningen is er één die hoogstwaarschijnlijk de juiste was: (4, 2, 5, 0, 7, 1, 6, 8, 3). Bij deze volgorde kwamen 12 digrams niet voor, terwijl bij de andere transpositions alle digrams minstens één keer voorkwamen. Deze volgorde heb ik bij de substitutie verder gebruikt en bleek uiteindelijk de juiste te zijn.

Toen ik eenmaal de plaintext had teruggevonden en wist over welke tekst het ging (zie volgende stap), kon ik de sleuteltekst voor de transpositie gemakkelijk raden: GENTLEMAN komt overeen met de gevonden kolomvolgorde.

3. Digramsubstitutie ongedaan maken

De substitutie bleek moeilijker te zijn. Ik heb eerst een aantal varianten van bruteforce uitgeprobeerd, met beperkingen op de lengte van de tekst of het aantal digrams om mogelijkheden voor uit te proberen, om de runtime enigzins te beperken. Ik gebruikte woordenlijsten om met het aantal gematchte woorden te berekenen hoe hard de gesubstitueerde tekst leek op een juiste tekst. Dit gaf geen goede resultaten.

Ook heb ik geprobeerd te vertrekken vanuit een frequentietabel en van daar de n ($n=2, n=3$) meest waarschijnlijke mappings van digram naar letter uit te proberen. Ook hierbij geen succes en een zeer lange runtime om iets betekenisvolks te kunnen uitproberen.

Ten slotte besefte ik dat ik een meer gestructureerde aanpak nodig had. Ik ben gegaan voor het Hill Climbing algoritme. Dit algoritme begint van een manueel gekozen startmapping van digrams naar letters (ik heb gekozen voor de mapping van een tekst die exact de frequentietabel volgt) en gaat dan telkens de mapping van twee digrams omwisselen. Aan de hand van een fitness-score berekent het algoritme of deze mapping beter of slechter is dan de huidige mapping. Als de nieuwe mapping beter is, wordt deze behouden en wordt het proces herhaald. Ik heb de Stochastic variant op dit algoritme gebruikt: deze kiest de twee digrams om te wisselen telkens willekeurig en stopt wanneer alle opties slechter zijn dan de huidige mapping. Op dat moment is een (lokaal) maximum bereikt. Voor de fitness-functie leek de woordenlijst-aanpak mij niet geschikt aangezien deze geen duidelijk verschil zou geven in score bij één aanpassing aan de substitutie, wat net belangrijk is voor Hill Climbing.

Daarom heb ik gekozen voor een variant op de fitness functie hier beschreven: <http://practicalcryptography.com/cryptanalysis/text-characterisation/quadrgrams/>. Deze gebruikt een vooraf berekende n-gramfrequentietabel en verhoogt voor elke n-gram in de gegeven plaintext de fitness score met de frequentie uit de tabel. Quadrgrams bleken het beste te werken.

Met deze aanpak kon ik de plaintext vinden.

Aangezien mijn implementatie Stochastic Hill Climbing gebruikt geeft het script (substitution_solver.py) niet telkens dezelfde output. Meestal kom ik met een fitness score van 75.12 op deze tekst:

LETRANFEVOXAFAILAVAITSIGIENCOMMENCECEPENDANTPOURMAPARTZENENBISZAMAISQUISANNONCATSOUSDEPLUSHEUREUJA
USPICESLAPROVENCEESTUNTRANSATLANTIQUERAPIDECONBORTAGLECOMMANDEPARLEPLUSABBAGLEDESHOMMESLASOCIETELA
PLUSCHOISISEXTROUVAITREUNIEDESRELATIONSSEBORMAIENTDESDIVERTISSEMENTSSORFANISAIENTNOUSAVIDONSSETTEIM
PRESSIONEJQUISEDETRESEPARATESDUMONDERREDUITSANOUSMEMESCOMMESURUNEILEINCONNUEOGLIFESPARNSEQUENTDENOU
SRAPPROCHERLESUNSDESAUTRESETNOUSNOUSRAPPROCHIONSAVEYVOUSZAMAISSONFEACEQUILXADORIFINALTDIMPREVUDAN

SCEFRUPEMENTDETRESQUILAVEILLEENCORENESECONNAISSAIENTPASETQUIDURANTQUELQUESZOURSENTRELECIELINBINIE
 TLAMERIMMENSEONTVIVREDELAVIELAPLUSINTIMEENSEMGLEVONTDEBIERLESCOLERESDELOCEANLASSAUTTERRIBIANTDESV
 AFUESETLECALMESOURNOISDELEAUENDORMIECESTAUBONDVECUEENUNESORTEDERACCOURCITRAFIQUELAVIEELLEMEMEAVECS
 ESORAFESETSFRANDEURSSAMONOTONIEETSADIVERSITEETVOILAPOURQUOIPUTE TREONFOUTEAVECUNEHATEBIEVREUSEET
 UNEVOLUPTEDAUTANTPLUSINTENSECECOURTOXAFEDONTONAPERCOITLABINDUMOMENTMEMEOUILCOMMENCEMAISDEPUISPLUS
 IEURSANNESQUELQUECHOSESEPASEQUIAZOUTESINFULIEREMENTAUJEMOTIONSDELATRAVERSEELAPETITEILEBLOTTANTED
 EPENDENCOREDECEMONDEDONTONSECROXAITABBRANCHIUNLIENSUGSISTEQUINESEDENOUEQUEPEUAPEUENPLEINOCEANETPEU
 APEUENPLEINOCEANSERENOUELETEFRAPHESANSBILAPPELSDUNAUTREUNIVERSDOULONRECVRAITDESNOUVELLESDELABAC
 ONLAPLUSMXSTERIEUSEQUISOITLIMAFINATIONNAPLUSLARESSOURCEDEVOQUERDESBILSDEBERAUCREUJDESQUELSFLISSELI
 NVISIGLEMESSAFELEMXSTEREESTPLUSINSONDAGLEENCOREPLUSPOETIQUEAUSSIETCESTAUJAILESDUVENTQUILBAUTRECOUR
 IRPOUREJPLIQUERCENOUVEAUMIRACLEAINSILESPREMIERESHEURESNOUSSENTIMESNOUSSUVISESCORTESPREEDESMEMEPA
 RCETTEVOIJLOINTAINEQUIDETTEMPSCHUCHOTAITALUNDENOUSQUELQUESPAROLESDLAGASDEUJAMISMEPARLERENTD
 IJJAUTRESVINFTAUTRESNOUSENVOXERENTATOUSTRAVERSLESPACELEURSADIEUATTRISTESOUSOURIANTSORLESECONDZOUR
 ACINQCENTSMILLESDESCOTESBRANCAISESPARUNAPRESMIDIORAFEUJLETELEFRAPHESANSBILNOISTRANSMETTAITUNEDEPEC
 HEDONTVOICILATEURARSENELUPINAVOTREGORDPREMIERECLASSECHEVEUJGLONDGLOSSUREAVANTGRASDROITVOXAFESEU
 LSouslenomderacemomentprecisuncoupdetonnerreviolenteclata dans leciel somgrelesondeelectriquesburent
 INTERROMPUESLERESTEDELADEPECHENENOUSPARVINTPASDUNOMSouslequelsecachaitarsenelupinonnesutquelinittia
 LESILSEBUTAFIDETOUTEAUTRENNOUVELLEZENEPOINTQUELESECRETENEUTETESCRUPULEUSEMENTFARDEPARLEEMPLOX
 ESDUPOSTETELEGRAPHIQUEAINSIQUEPARLECOMMISSAIREDUGORDETPARLECOMMANDANTMAISILESTDECESEVENEMENTSQUISE
 MGLENTBORCERLADISCRETIONLAPLUSRIFOURUSELEZOURMEMESANSQUONPUTDIRECOMMENTLACHOSESEAVAITTEEGRUITENOU
 SSAVIONSTOUSQUELEBAMEUJARSENELUPINSECACHAITPARMINOUS

en dit met substitutie ||A|D|F|G|V|X||—|—|—|—||A|Y|G|J|Q|T|C||D|M||D|V||F|F|X|R|U|||G||O||
 B||||V||E||||L||X|A|H|S|P|N|Z|

met 12 lege vakken voor de 12 ontbrekende digrams.

Deze substitutie was nog niet perfect maar leek wel hard op een Franse tekst, en kon dus gemakkelijk manueel verbeterd worden:

Digram Waarde gevonden dmv hill climbing Manuele verbetering

FA	f	g
GG	b	f
AD	g	b
AF	j	x
FD	x	y
XX	z	j
AA	y	z

met ||A|D|F|G|V|X||—|—|—|—||A|Z|B|X|Q|T|C||D|M||D|V||F|G|Y|R|U|||G||O||F|||V||E|||
 ||L||X|A|H|S|P|N|J|

Dit geeft de volgende tekst:

letrangevoyage ilavaitsibiencommencecependantpourmapartjenenfisjamaisquisannoncatsousdeplusheureuxauspicesla provence est un transatlantique rapide confortable commandeparle plus affable des hommes la societe la plus choisiesy trouvaitreunie des relations se formaient des divertissementssorganisaient nous avions cette impressionexquisedetresparesdumondereduitsanousmemes commesuruneileinconnueobligesparconsequentdenousrapprocherlesunsdesautresetnousnousrapprochionsavezvousjamaissongeacequilyadoriginaletdimprevudansce groupementdetresquilaveilleencoreneconnaissaientpasetquidurantquelesjournsentrelecielinfinietlamerimmenseontvivredelavielaplusintimeensemblevontdefierlescoleresde loceanlassautterrifiantdesvaguesetlecalmesournoisdeleauendormiecestaufondvecueenunesortederaccourcitrage quelelavieellememeavec esoragesetsesgrandeurssammonotonieetsadiversiteetvoilapourquoipute treongouteavecunehatiefvreuseetunevoluptedautantplusintensececourtvoyagedontonapercoitlafindumomentmemeouilcommencemaisdepuisplus ieursannesquelquechose sepasequiajoutesingulierementauxemotionsdelatradeelapetiteileflottantedependencoredecemondedontonsecroyaitaffranchiunliensistequinesedenouuequepeuapeuenpleinoceanetpeuapeu enpleinocean sereno ueletelegraphesansfilappelsdunautreuniversdoulonreceivingt desnouvellesdelafaconlaplusmysterieusequisoitlimaginationnaplusslaressource devoquerdesfilsdeferauxrueuxdesque lsglisseli nvisible message mystereestplusinsondableencorepluspoetiqueaussietcestauxailesduventquifautrecourirpourepliquercentouveaumiracleainsilespremieresheuresnoussentimesnoussuvise escortespreeedesmemeparcettevoixlointainequidetempschuchotaitalundenosquelesparolesdelabasdeuxamismeparlerentdixa utresvingtautresnousenvoyerentatoustraverslespaceleursadieuattristesousouriantsorlesecoudjour acinqcentsmillesdescotesfrancaisesparunapresmidiorageuxletelegraphesansfilnouistranmettaitudedepec hedontvoicilateneurarsenelupinavotrebordpremiereclassecheveuxblonds blessure avant bras droit voyage seu lsouslenomderacemomentprecisuncoupdetonnerreviolenteclata dans leciel sombre les ondeselectriques furent interrompueslerestes de la depecheneno usparvintpasdunomsouslequelsecachaitarsenelupinonnesutquelinittialesilse futagide toutes autrenouvelles jene doute point que le secret eut ete scrupuleusement garder par le employ esdupo stetelegraphiqueainsiqueparlecommissairedubordetparlecommandantmaisilestdecesevenementsquise mblentforcerladi discretionlaplusrigoureuselejourmemesansquonputdirecommentlachoseseavaitteebruiteenou ssavionstousquelefameuxarsenelupinsecachaitparminous

Deze komt uit de verzameling verhalen van Maurice Leblanc Arsène Lupin, gentleman-cambrioleur, specifiek het deel L'arrestation d'Arsène Lupin.

Enigma

Hoe uit te voeren

Voer de pythonfile main.py uit.

Oplossing

alice della rocca odiava la scuola di sci odiava la sveglia alle sette e mezzo del mattino anche nelle v
di natale e suo padre che a colazione la fissava e sotto il tavolo faceva ballare la gamba nervosa mente
su s briga ti odiava la calza maglia di lana che la pungeva sulle cosce le offrii o le che non le lasciav
le dita il casco che le schiaccia vale guance e puntava con il ferro sulla mandibola e poi quegli scarpo
troppo stretti che la facevano camminare come un gorilla

Deze tekst komt uit La solitudine dei numeri primi door Paolo Giordano.

Rotors: [1, 3, 4], ['E', 'F', 'I']

Plug mapping: KBCXEUGNSWALMHOPQRITFVJDYZ

Oplossingsmethode

Enigma wordt ontcijferd door het implementeren van de Advanced Turing Bombe.

Permutatiematrix

De file Permutatiematrix.py bevat de implementatie van de permutatiematrix. Deze matrix volgt het algoritme van de geavanceerde Turing Bombe. Stroom wordt gesimuleerd door op elke positie van de matrix een PermutationNode te plaatsen. Ieder node object heeft een lijst van connections (propagations). Deze lijst bevat alle andere nodes, waarmee de gegeven node verbonden is (verbindingen worden gemaakt volgens het algoritme van de cursus). Voor het onder stroom zetten van een node, wordt de trigger() method gebruikt. Dit zet de gegeven node 'onder stroom', en zet recursief alle connecties onder stroom. Indien deze node al onder stroom stond, zal die deze trigger negeren. Dit voorkomt hetzelfde werk meerdere keren te doen en vermindert ook oneindige recursie. Er is een clear() method, om alle stroom te laten verdwijnen. Het is makkelijk te checken welke nodes onder stroom worden gezet: indien er 1 node van de gegeven rij onder stroom staat, zullen de rotors en rotorposities juist zijn.

Moeilijkheden

Het evenredig verdelen van Enigma, zodat iedereen een deel kon bijdragen aan het oplossen van deze cipher.

References

- Course Material Code Theory from Professor S. Symens
- Enigma tool from <https://atlas.uantwerpen.be/~ssymens/enigma.php>

Playfair

:warning: Disclaimer: de code in deze folder gebruikt the infamous **BIG BALL OF MUD** pattern. Wees gewaarschuwd.

De oorspronkelijke mappenstructuur zag er als volgt uit:

Laten we beginnen met de root folder. Naast deze README.md file kan je er de volgende belangrijke bestanden vinden:

- bigram_test.png: bigramverdeling van de ciphertext.
- decrypted.txt: output van de vorige runs van de Playfair solver
- graphic_generator.py: script dat ik heb gebruikt om de grafieken voor de bigramverdelingen te genereren
- playfair.py: Playfair solvers (run die om de code te cracken!)
- text_splitter: bevat een handig scriptje om een string in woorden te splitsen en dus om de decryptie meer leesbaar te maken

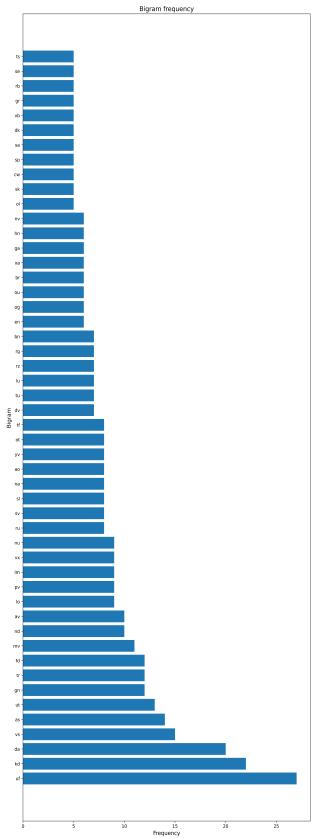
Bepalen van de taal

Voor de opgave hebben we als hint gekregen dat het originele bericht in het Nederlands, Frans, Duits, Engels, Italiaans of Spaans geschreven is. Om te bepalen met welke van deze talen ik verder moet werken, heb ik een frequentieanalyse toegepast.

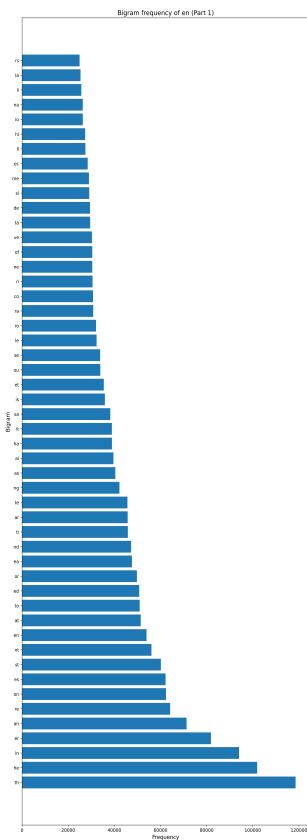
Frequentieanalyse

Aangezien dat Playfair een bericht bigram per bigram encrypteert, vond ik het logisch om een frequentietabel op te stellen voor elk van de gegeven talen en te vergelijken met de frequentie van de ciphertext om te bepalen of de taal min of meer hetzelfde patroon volgt als de gegeven ciphertext. Hiervoor heb ik een dataset van 100000 zinnen van elke taal gepakt van <https://wortschatz.uni-leipzig.de/en/download>. In corpus/original folder kan je die datasets vinden. Daarna heb ik met corpus/corpus_preprocessing.py alle bestanden verwerkt zodat ze alleen lowercase letters bevatten (daarnaast heb ik j met i vervangen en en ook de letters met diakritische tekens met overeenstemmende letters van het Latijnse alfabet). Het resultaat kan je in de corpus/preprocessed folder vinden.

Tot slot, heb ik grafieken van de bigramfrequenties gegenereerd met graphic_generator.py. Voor de ciphertext zelf was de volgende afbeelding de output:



Mijn conclusie was dat de meest waarschijnlijke taal het Engels is. Hier is de grafiek hiervan (voor de andere talen kan je grafieken in de stats folder vinden):



Simulated annealing

Simulated annealing is een optimalisatiealgoritme dat kan gebruikt worden om een globaal extremum te vinden (in ons geval het globale maximum, maar daarover verder meer). Om gevallen te voorkomen waarbij het model in een lokaal extremum blijft hangen (wat in het geval van hill climbing zou kunnen gebeuren), wordt er een concept van *temperatuur* geïntroduceerd. Hoe hoger de temperatuur, hoe groter de kans dat een key met een slechtere score geaccepteerd wordt. In de loop van de simulated annealing algoritme daalt de temperatuur geleidelijk, wat tot een *greedier* search leidt.

Samengevat, hier is een overzicht van hoe het algoritme in het algemeen werkt:

1. Begin met een start key k en temperatuur t .
2. Bij elke stap wordt t afgekoeld volgens temperatuurrooster r .
3. Bij elke stap, bereken een neighbour van de huidige sleutel
4. Bereken de score van de nieuwe sleutel:
 1. Als de score hoger is dan de huidige score, wordt de neighbour sleutel de huidige sleutel.
 2. Zo niet, kan de nieuwe sleutel toch geaccepteerd worden als huidige sleutel met de kans $e^{(huidige score - nieuwe score)/(temperatuur)}$
5. Herhaal 2-6 tot $t=0$ wordt.
6. Return de beste sleutel met de beste score.

Een van de grootste striukelpunten voor mij was om een goede heuristiek en parameters te vinden. Helaas, van wat ik heb gelezen, zijn er geen vaste regels hoe je dat moet doen, waardoor mijn zoektocht naar de juiste oplossing een beetje langer was dan ik had verwacht.

Hier is een korte overzicht van de parameters:

Start key

Op zich speelt het geen grote rol, maar in het geval van deze ciphertext geeft het gemiddeld betere resultaten als je met de string abcdefghijklmnopqrstuvwxyz begint dan met een willekeurige sleutel. Maar opnieuw, door de aard van het algoritme, werkt het in beide gevallen.

Begintemperatuur

Hangt sterk van de evaluatiefunctie en de afkoelingsstrategie af. Ik heb gekozen om die niet te hoog te zetten, omdat anders het aantal van totale stappen onnodig hoog zou liggen. In de huidige implementatie is deze parameter gelijk aan 50.

Afkoelingsstrategie

Er bestaan verschillende afkoelingsstrategieën ([paper](#)), maar de simpelste daarvan, namelijk de lineaire daling, geeft vaak ook de beste resultaten. Dus in dit project heb ik gekozen om niet het wiel opnieuw uit te vinden en gewoon deze strategie gepakt.

Aantal stappen

Bepaalt hoeveel iteraties met dezelfde temperatuur er worden uitgevoerd. Wordt experimenteel bepaald. Voor de huidige

implementatie heb ik dit getal aan 500 gelijk gesteld.

Aantal iteraties

Bepaalt hoeveel keer je de simulated annealing reset voor je opnieuw vanaf nul begint. Is gelijk aan 3 in de huidige implementatie (ook experimenteel bepaald).

Neighbour generation function

In het begin dacht ik om alleen swaps van letters te beschouwen (elke neighbour ligt dus op Hamming afstand 2). Als een andere mogelijkheid beschouwde ik om een kans toe te voegen dat er twee kolommen of rijen worden geswapped, maar ik heb gemerkt dat het soms zelfs tot slechtere resultaten kan leiden, dus in mijn eindimplementatie heb ik gekozen om die weg te halen.

Evaluatiefunctie

Waarschijnlijk de belangrijkste parameter. In de loop van het project heb ik verschillende manieren uitgeprobeerd:

- Tweenorm afstand berekenen tussen frequentie van bigrams van de ontcijferde tekst en de zelfopgestelde frequentie lijst (te minimaliseren)
- Som van alle frequenties van de woorden die in het ontcijferde bericht voorkomen (te maximaliseren)
- Som van frequenties van de bigrams die in de ontcijferde tekst voorkomen
- ...
- Som van de frequenties van de quadgrams van de ontcijferde tekst

Voor mijn eindimplementatie heb ik het laatste gekozen (omdat dat het beste werkte). Voor de quadgrams frequenties heb ik een lijst van hier gepakt: <http://practicalcryptography.com/cryptanalysis/text-characterisation/quadgrams/> en naar log probabilities omgezet. Ik heb het ook eerst uitgeprobeerd met zelfafgeleide frequentie lijst (met Laplacian smoothing, want sommige combinaties kwamen niet voor in de dataset), maar dat gaf geen goede resultaten (ik vermoed dat er waarschijnlijk een groter corpus nodig was).

Oplossing

Bij een van de runs heb ik het volgende resultaat gekregen:

```
1733165260.5431437
New best score: -7039.003283658883
Best key found: ['u', 'r', 's', 'o', 't', 'h', 'c', 'k', 'l', 'f', 'y', 'i', 'g', 'm', 'b', 'q', 'w', 'x'
Split text: you dont know about meu nl esx s you have read a box ok by the name of the adventures of tom
that box ok was writ x ten by mark twain and he holds the truths mainly not all parts of the story are t
but most of it is i dont know anyone who tel xl s the truth all the time except perhaps aunt polly or th
douglas or tom sawyers x sister mary these people are written about in the adventures of tom sawyer that
ends like this to mandi find money that was x stolen and we are allowed to k exe pit we become rich we e
six thousand x do lx lars in gold i ud get hatcher put the money in a bank for us and we can have a doll
that is more money than a person can knowhow to spend the widow douglas to x ok me into her home to live
note nio y living in a nice house i put on my old clothes and ran away and was fr exe and hap xp y but x
found me and said that if i wanted to io in his club and befriends i would have to return to live with t
this reason i returned to live with x her the widow cried over me and gave me new clothes to wear but i
new clothes i felt too warm in the mandi could not move my arms and legs f rex ely when supper was being
widow always ran gabel xl and i had to come quickly i was happier when i could eat whenever i chose to t
meant i had to make meals of the bits of food other people had thrown away when i asked permission to sm
widow said no she thought x that smoking was a dirty habit and told me that i must not smoke her sister
swat son a woman who had never marx ried and who had no children of her own came to live with x her she
x that she could change me and make me a bet xt er person by educating me and teaching me to sp el xl sh
with me for an hour until the widow made her stop miss watson complainedabout everything i did dont put
ex et up there huckleberry sit straight in your chair why cant you improve the way x you act dont be so
disrespectful to those who are trying to co rx rec t you x
```

Ziet er goed uit! Dit fragment komt uit The Adventures of Huckleberry Finn geschreven door Mark Twain.

Extra

Om de juiste decryptie gemakkelijker te kunnen herkennen, heb ik een scriptje gemaakt dat gebaseerd is op Zipf's law en dat een string (met de taal gegeven) spaties toevoegt. Zoals je boven kan zien, werkt het best ok (buiten het feit dat het "complainedabout" als één woord beschouwt).

Cracking the Vigenere Cipher

Hoe uit te voeren

Voer de pythonfile main.py uit. Verslag is hieronder te vinden.

Oplossing

Rauwe ontcijferde tekst: text
DOEMAARGEWONALSOFIKERNIETBENZEIIKTEGENHETKINDATVANDEHONGERAANHETSTERVENWASENDATIKPROBEERDETEFOTO
GRAFERENIKWASZENUWACHTIGENWOUDATIKEENPILTESLIKKENHADDIEHETBEVENVANMIJNHANDENZOUSTOPPENERGENSVOELDE
IKDATDITMIJNFOTIZOUWORDENDEFOTODIEFOTODIEMIJNGROTEDOORBRAAKZOUNLUIDENWAARDOORIKMIJNMARKTWAARDEKON
OPDRIJVENDIEHETMIZZOUTOESTAANDEGROTEBAASVANREUTERSTERVAGENOFHIJMJEENSTERUGKONBELLENWANNEERHETMIJB
ETERPASTEEENFOTOGRAAFVOELTZOIETSDEWEREELDBEROEMDEHENRICARTIERBRESSONVOELDEHETTOENHIJDATJONGETJEMED
ETWEEWIJNFLESSENINDEPARIJSERUEMOUFFETARDVASTLEGDEELLIOTTERWITTVOELDEHETTOENDIENEGERVOORHETOOGVANDEC
AMERAZIJNTONGUITSTAKALFREDSTIEGLITZVOELDEHETTOENDATMOOIMEISJEMETDENOGMOOIEREVINGERSHAARJASHADDICH
TGEKNOOPTOPHETJUISTEMOMENTENEDWARDSTEICHENHADHONDERDENKIEKJESVANGRETAGARBOGESCHOTENMAARHADNOGTIJDE
NSHETSCHERPSTELLENVANZIJNLENSGEVOELDDITWORDTHETENIGEWARESCHONEULTIEMEPORTRETVANDEGODINHETZELFDEALS
WATIKVOELDEMETHETUITGEHONGERDEKINDINMIJNVIZIERZALIGOPAVONDENDIENERGENSVOORDEUGENDANVOORFLAUWEKULHO
ORJEWEELEENSBEWERENDATFOTOGRAFIEVEELZONIETALLESMETGELUKVANDOENHEEFTENDANBEGINNENZEOVERDEMAKERVANDEF
OTODIEIEDEREENKENTHETNAAKTEMESISJEVERBRANDRENNENDMETDEARMENOPENCHRISTUSMETEENKUTALSDEFOTOGRAAFNIETT
OEVALLIGOPDEPLAATSVANHETNAPALMBOMBARDEMENTWASGEWEESTZOREDENERENZEDANHADHIJNOOITDIEFOTOKUNNENSCHIET
ENENDUSHEEFTHETTEMAKENMETGELUKTJAUGAATTOCHNIETMOPPERENDATKHETGELUKHADDATERVOORMIJNNOGENEENKINDLAGT
ECREPERENIKHADDATGELUKNIETIKHADDATTALENTZOALSROBERTCAPAHTTALENTHADDENEUSHADMETZIJNCAMERAOPDEPLAAT
STEZIJNWAAREENSOLDAATDEHERSENENUITDEKOPWERDENGESCHOTENGELUKZEGGENBERGBEKLIIMMERSDIEEENMOORDENDESTEE
NLAWINEOPDRIECENTIMETERVANHUNSMIKKELZAGENVOORBIJRAZENGELUKISOPDENDUURENKWESTIEVANBEKWAAMHEIDIKWEE
TDATZEDAARGELIJKINHEBBENI

Ontcijferde tekst (met spaties): text DOE MAAR GEWOON ALSOF IK ER NIET BEN ZEI IK TEGEN HET KIND DAT VAN DE HONGER AAN HET STERVEN WAS EN DAT IK PROBEERDE TE FOTOGRAFEREN IK WAS ZENUWACHTIG EN WOU DAT IK EEN PIL TE SLIKKEN HAD DIE HET BEVEN VAN MIJN HANDEN ZOU STOPPEN ERGENS VOELDE IK DAT DIT MIJN FOTO ZOU WORDEN DE FOTO DIE MIJN GROTE DOORBRAAK ZOU INLUIDEN WAARDOOR IK MIJN MARKTWAARDE KON OPDRIJVEN DIE HET MIJ ZOU TOESTAAN DE GROTE BAAS VAN REUTERSTE VAGEN OF HIJ MIJ EENS TERUG KON BELLEN WANNEER HET MIJ BETER PAST EEN FOTOGRAAF VOELT ZOIETS DE WERELD BEROEMDE HENRI CARTIER BRESSON VOELDE HET TOEN HIJ DAT JONGETJE MET DE TWEË WIJNFLESSEN IN DE PARISE RUE MOUFFETARD VASTLEGDE ELLIOT ERWITT VOELDE HET TOEN DIE NEGER VOOR HET OOG VAN DE CAMERA ZIJN TONG UIT STAK ALFRED STIEGLITZ VOELDE HET TOEN DAT MOOIE MEISJE MET DE NOG MOOIERE VINGERS HAAR JAS HAD DICHTGEKNOPT OP HET JUISTE MOMENT EN EDWARD STEICHEN HAD HONDERDEN KIEKJES VAN GRETA GARBO GESCHOTEN MAAR HAD NOG TIJDENS HET SCHERPSTELLEN VAN ZIJN LENS GEVOELD DIT WORDT HET ENIGE WARE SCHONE ULTIEME PORTRET VAN DE GOD IN HET ZELFDE ALS WAT IK VOELDE MET HET UITGEHONGERDE KIND IN MIJN VIZIER ZALIG OP AVOND EN DIE NERGENS VOOR DEUGEN DAN VOOR FLAUWEKUL HOOR JE WELEENS BEWEREN DAT FOTOGRAFIE VEEL ZONIET ALLES MET GELUK VAN DOEN HEEFT EN DAN BEGINNEN ZE OVER DE MAKER VAN DE FOTO DIE IEDEREEN KENT HET NAAKTE MEISJE VERBRAND RENNEND MET DE ARMEN OPEN CHRISTUS MET EEN KUT ALS DE FOTOGRAAF NIET TOEVALLIG OP DE PLAATS VAN HET NAPALMBOMBARDEMENT WAS GEWEEST ZO REDENEREN ZE DAN HAD HIJ NOOIT DIE FOTO KUNNEN SCHIETEN EN DUS HEEFT HET TE MAKEN MET GELUK TJAU GAAT TOCH NIET MOPPEREN DAT IK HET GELUK HAD DAT ER VOOR MIJN OGEN EEN KIND LAG TE CREPEREN IK HAD DAT GELUK NIET IK HAD DAT TALENT ZOALS ROBERT CAPAHT TALENT HADDENEUS HAD MET ZIJN CAMERA OP DE PLAATS TE ZIJN WAAR EEN SOLDAAT DE HERSENEN UIT DE KOP WERDEN GESCHOTEN GELUK ZEGGEN BERG BE KLIMMERS DIE EEN MOORDENDE STEENLAWINE OP DRIE CENTIMETER VAN HUN SMIKKEL ZAGEN VOORBIJ RAZEN GELUK IS OP DEN DUUR EEN KWESTIE VAN BEKWAAMHEID IK WEET DAT ZE DAAR GELIJK IN HEBBEN I

Van het boek 'Problemski Hotel' van 'Dimitri Verhulst'

Oplossingsmethode

Eerst heb ik de enkel-kolomtranspositie achterhaald. Vervolgens heb ik [Simon Singh cracking the vigenere cipher](#) gebruikt om de laatste sleutel (VERHULST) en de tekst te achterhalen.

Tot slot, heb ik een gelijkwaardig systeem geïmplementeerd, zodat het automatisch de tekst ontcijfert zodat de website van Simon Singh niet meer nodig is.

Enkel kolomtranspositie

Een eigenschap van Vigenere is dat deze cipher veel herhaling heeft (vooral woorddelen van lengte 3). Deze eigenschap wordt geëxploiteerd voor het oplossen van de enkel kolomtranspositie. Hierbij wordt elke enkel-kolomtranspositie uitgevoerd. Vervolgens wordt er door middel van **Suffix Arrays** ([suffix array explanation](#)) makkelijk gezocht naar common prefixes (CP) in de tekst. Als metric pakken we de CP van de enkel-kolomtranspositie. We nemen de decryption dat de hoogste CP-hoeveelheid heeft van lengte 3. Gelukkig voor ons kwam dit na de eerste poging uit op de ontcijfering van de enkel-kolomtranspositie.

Vigenere Cipher

De Vigenere cipher voert steeds een Caesar Cipher uit, zodat voor iedere groep posities (met dezelfde mod key lengte), het meest voorkomende karakter gelijkgesteld is aan 'E'. We maken wel gebruik van het feit dat we de key lengte weten (kan via trial and error/ Simon Singh tool). Door steeds het meest frequente karakter te linken aan het karakter 'E', komen we snel tot het juiste antwoord.

Concrete functies

- CommonPrefixCounter.py zorgt ervoor dat gegeven een string, het makkelijk is te tellen hoe vaak dat een zekere 3-letter combinatie voorkomt.
- TranspositionCipher.py voert de enkel kolomtranspositie encryptie en decryptie uit.
- Vigeneresolver.py kraakt de vigenere zelf, door steeds Caesar ciphers uit te voeren, en steeds het meest frequente karakter gelijk te stellen aan 'E'
- main.py hier is de functie find_highest_cp, deze functie voert alle mogelijke enkel-kolomtranspositions uit, en behoudt de key van de resulterende transpositie decryptie die de meeste common prefixes heeft. Deze file voert ook de hele decryptie uit.

Moeilijkheden

Niet aanwezig, het werkte na de eerste poging.

References

- Course Material Code Theory from Professor S. Symens
- Vigenere cracking tool (https://www.simonsingh.net/The Black Chamber/vigenere_cracking_tool.html) from Simon Singh
- Suffix Array (<https://cp-algorithms.com/string/suffix-array.html>) Mainly from Jakobkogler