Codetheorie en Cryptografie

Rapport

3 Ba INF 2018-2019

Benjamin Vandersmissen
Frank-Jan Fekkes

May 3, 2019

1 Enigma

1.1 Werkwijze

De werkwijze die gebruikt wordt in de code om de enigma code te kraken, is de werkwijze beschreven in de les, door een graph te maken van 26x26 knooppunten en de knooppunten te linken met enigma machines aan de hand van de Crib. Voor elke mogelijke combinatie vam rotors en standen wordt dan gezien of de volledige graph 1 lampje op elke rij heeft of niet.

1.2 Resultaat

Uiteindelijk kwamen we na het algoritme uit te voeren een graph uit waarbij in elke rij buiten de Y-rij 1 lampje brandde, in de Y-rij brandde er geen. We konden dan heel gemakkelijk inferreren uit de rest van de graph dat Y ook gewoon gemapt moest worden op Y. We hebben geluk gehad met het populeren van de graph dat A effectief op A gemapt werd, omdat dit de waarde was die we initieel gebruikten om de rest van de graph te populeren. Als dit niet zo was, dan moesten we nog een beetje meer werk verrichten.

Rotors en Rotorstand : 2-3-4, D-H-G
Deel van de oplossing; spaties zijn toegevoegd voor leesbaarheid:

```
DE EERSTE OPGAVE VOOR ENIGMA EINE INFACHER JUNGER MENSCH REIST (...)
```

2 Vigenere Plus

2.1 Werkwijze

Voor Vigenere Plus moesten we 2 stappen doen, eerst de juiste permutatie vinden van de letters en daarna een paar simpele caesar sleutels oplossen. Om de permutatie te vinden, hebben we geopteerd om een brute force algoritme te gebruiken dat elke mogelijke permutatie van lengte 2 tot lengte 10 afloopt, de tekst op die manier permuteert en dan probeert te bepalen of de gepermuteerde tekst overeen komt met een vigenere versleuteld bericht. Om te bepalen of een bericht vigenere versleuteld is, splitsen we de gepermuteerde tekst op in groepjes, waar we dan een analyse op doen om te zien of die tekst eigenschappen vertoont die een 'normale tekst' ook vertoont. Immers, als we de tekst met de correcte lengte van het codewoord zouden kunnen splitsen, dan is elk groepje een simpele caesarsubstitutie en die behoudt de eigenschappen van de taal van de oorspronkelijke tekst.

De eerste manier die we probeerden om uit te vinden was gewoon te kijken naar de maximum en de minimum frequentie van letters in een groepje en als die voldoende ver uit elkaar lagen, dan was het de tekst die we zochten. We hebben geexperimenteerd met verschillende tresholds, die we gebaseerd hebben op frequentietabellen van de talen.

Een andere manier was dan te zien naar de som van de afstanden tussen de frequentie van elke letter en de frequentie als de letters uniform verdeeld zouden zijn. De gepermuteerde tekst met de hoogste som, zou dan wel de juiste tekst moeten zijn.

Deze eerste 2 werkwijzes gaven geen fatsoenlijke resultaten. Achteraf bleek dat er nog een programmeerfout in zat in het berekenen van de frequenties en dat dat de reden was waardoor die 2 methodes niet werkten, maar toen waren we al begonnen met het implementeren van nog een andere methode.

Ook bij de laatste methode waren er een hoop problemen, tot we doorhadden dat we een denkfout hadden gemaakt. We gingen er immers vanuit dat het omzetten van een tekst met enkele kolom-transpositie naar ciphertext en terug dezelfde operatie was. Maar dit is niet zo.

De laatste manier die we probeerden was het gebruiken van de zogenaamde 'Index of Coincidence', dit is een percentage berekend voor de hele tekst en des te hoger het percentage, des te meer kans dat de tekst geschreven is in een taal en niet gewoon random characters is.

2.2 Resultaat

Index of coincidence gaf wel een goed resultaat, oorspronkelijk hadden we gewoon gekozen om de tekst met de grootste index of coincidence te kraken, maar dit bleek een nonsens tekst te zijn. Dan hebben we een treshold ingevoerd, waarbij we alle teksten bijhielden met een hogere index of coincidence dan 0.7, dit gaf dan ons wel het verwachte resultaat. De frequentie analyse en raden van het codewoord gebeurde niet in onze code, maar via een externe site (`https://f00l.de/hacking/vigenere.php`)

Deel van de oplossing; spaties zijn toegevoegd voor leesbaarheid:

```
KLOOP ZO HARD ALS MIJN KORTE BLOTE BENEN MIJ DRAGEN KUNNEN (...)
```

3 Playfair

3.1 Eerste ideeën

Voordat we op de gevonden werkwijze terecht zijn gekomen, hadden we nog een hoop andere ideeën. Hier zijn er enkele samen met de reden waarom ze niet werkte.
Search algoritms, zoals breath en depth search waren als eerst bedacht als mogelijke oplossing. Het probleem is dat de search space ongeloofelijk groot is (25!). Ookal bestaan er verschillende keys die de correcte oplossing geven, is het te inefficiënt om breath search te implementeren. Depth search lost het probleem van geheugen verbruikt op. Er duikt wel een nieuw probleem op: de makkelijkste manier om te weten of twee keys hetzelfde zijn is door ze de tekst te laten ontcijferen. Dit is maar een stap verwijderd van brute force waarvoor alleen de score nog maar nodig is. Een laatste oplossing zou het A-ster algoritme kunnen zijn maar daar heeft men een correcte heuristiek nodig. Deze hebben we niet kunnen vinden. Dit type zoek algoritmes zijn dan ook achterwege gelaten.
Pure Hillclimber algoritme, het Hillclimber algoritme hadden we eerst echt geïmplementeerd. Het probleem is dat er geen goede oplossingen uit kwamen ookal werden alle kinderen nagegaan. Dit komt omdat dit algoritme naar locale minima/maxima gaat zoeken en geen garantie kan geven dat het om een globaal extremum gaat. Bij het ontcijferen van de tekst zijn er juist zeer veel locale extrema. Hiermee is Hillclimber op zichzelf ook niet de correcte methode. Met een kleine aanpassing zijn we dan terecht gekomen op simulated annealing.

3.2 Simulated Annealing

In de vorige sectie was het duidelijk gemaakt dat Hillclimber wel op het goede pad zat maar dat het alleen locale extrema gaf. Om dit te vermijde bestaat er een variant: simulated annealing. I.p.v. altijd een stap in de beste richting te zetten, wordt er nu een kans berekend om een verkeerde stap te nemen. Op deze manier wordt er breder gezocht in de search space. Er is een grotere kans dat

men op een "globaler" maximum vindt. Wanneer het algoritme begint wordt er een random key gegenereerd. Vanuit deze key worden er buren (neighbours) gegenereerd. De scores van de keys worden dan vergeleken met de originele key. Als de buur beter is dan de oude key wordt deze onmiddelijk geaccepteerd. Zoniet, wordt er een kans opgesteld om te zien of we deze slechte key toch gaan accepteren. De kans (P) dat we een slechte stap gaan zetten wordt kleiner als het algoritme langer bezig is (temperatuur wordt kouder: T wordt lager) en als het verschil ($diff$) met de goede key groter wordt.

$$P_{accept\ bad\ key} = e^{diff/T} \quad (1)$$
$$diff = verschil\ scores \quad (2)$$
$$T = temperatuur \quad (3)$$

Na iedere zoveel verbeteringen wordt de temperatuur verlaagt. Er wordt meer locaal gezocht naar verbeteringen. We stoppen met verbeteren als er geen verbeteringen meer gevonden worden na een x aantal buren.

3.2.1 Score

Om verschillende sleutels/keys met elkaar te kunnen vergelijken is er een score systeem opgezet. We kunnen het feit dat playfair met bigrammen werkt, uitbuiten. Twee letters worden namelijk altijd samen omgezet in dezelfde twee andere letters. Als we dan een tekst omzetten met een willekeurige sleutel, dan kunnen de bigrammen frequenties geteld worden. Deze worden dan vergeleken met frequenties in de taal. Hoe dichter deze frequenties bij elkaar liggen, des te meer kans dat we een goede sleutel te pakken hebben.
De score is de standaard afwijking van de gevonden frequenties met de verwachte frequenties. De constante term $\frac{1}{n-1}$ is weggelaten omdat deze constant is en alleen een invloed heeft op de kans op het accepteren van een slechte stap.

3.2.2 Buren

Buren van de key worden gegenereerd door een willekeurige transformatie toe te passen. Dit kunnen rij en kolom wisselingen zijn alsook letter wisselingen en andere.

3.3 Resultaat

Gevonden sleutel:

```
SXDPC
WBQUN
TYGIM
HVERO
```

ZAKFL

Deel van de oplossing; spaties zijn toegevoegd voor leesbaarheid; X-en zijn laten staan:

IN MY YOUNGER AND MORE VULNERABLE YEARS MY FATHER GAVE
ME SOME ADVICE THAT I VE BEEN TURNING OVER IN MY MIND EVER
SINCE WHENEVER YOU FEXEL LIKE CRITICIZING ANYONE HE TOLD
ME UST REMEMBER THAT ALL THE PEOPLE IN THIS WORLD HAVENT
HAD THE ADVANTAGES THAT YOU VE HAD HE DIDNT SAY ANYMORE BUT
WE VE ALWAYS BEXEN UNUSUALXLY COMXMUNICATIVE IN A RESERVED
WAY AND I UNDERSTOOD THAT HE MEANT A GREAT DEAL MORE THAN
THAT IN CONSEQUENCE I M INCLINED TO RESERVE ALXL UDGMENTS A
HABIT XTHAT HAS OPENED UP MANY CURIOUS NATURES TO ME AND
ALSO MADE ME THE VICTIM OF NOT A FEW VETERAN BORES THE
ABNORMAL MIND IS QUICK TO DETECT AND ATXTACH IT SELF TO
THIS QUALITY WHEN IT APXPEARS IN A NORMAL PERSON AND SO
IT CAME ABOUT THAT IN COLXLEGE I WAS UNUSTLY ACXCUSED OF
BEING AP OLITICIAN BECAUSE I WAS PRIVY TO THE SECRET GRIEFS
OF WILD UNKNOWN MEN MOST OF THE CONFIDENCES WERE UNSOUGHT
FREQUENTLY I HAVE FEIGNED SLEXEP PREOCXCUPATION OR A HOSTILE
LEVITY WHEN I REALIZED BY SOME UNMISTAKABLE SIGN THAT AN
INTIMATE REVELATION WAS QUIVERING ON THE HORIZON FOR THE
INTIMATE REVELATIONS OF YOUNG MEN OR ATLEAST XTHE TERMS IN
WHICH THEY EXPRESXS THE MARE USUALXLY PLAGIARISTIC AND
MARRED BY OBVIOUS SUPXPRESSIONS RESERVING UDGMENTS IS A
MATTER OF INFINITE HOPE I AM STILL A LITXTLE AFRAID OF
MISSING SOMETHING IF I FORGET (...)