Cybersecurity - Homework 1

Vlad Turno (1835365)

October 12, 2025

1) Introduction

In cryptography, frequency analysis is a technique that studies the number of occurrences in a given ciphertext in order to get the plaintext, thanks to the fact that in written language certain characters - such as vowels - occur in certain percentages (or at least in a meaningful speech). This technique is efficient for decrypting substitution ciphers, where the encoding of a message consists of a direct correspondence between a plaintext character and an encrypted one. Now consider an example of a ciphertext encrypted by substitution:

```
OUETOJIDECEJTCE OTUWSTDWCCILETIOTBIFATFWOUTITSYICCTNILTHATUWSTSUHRCBEJTT
IABTUET HCCHFEBTOUETAIJJHFTJHIBTOUIOTCEBTIQJHSSTOUET WECBSTFUEJETOUETLJISSTT
FISTSOWCCTFEOTFWOUTBEFTIABTOUETIWJTFISTQHHCTFUWCETOUETSHRABTH TNWJBSTT
EQUHEBT JHYTOUETOJEESTOUIOTSOHHBTIOTOUETEBLETH TOUET HJESOTIABTOUETSXGTT
LJEFTNJWLUOEJTFWOUTEDEJGTSOEPTISTOUETSRATJHSETNEUWABTOUETBWSOIAOTUWCCSTT
IABTOUETSUIBHFSTSUHJOEAEBTFUWCETOUETQHCHJSTH TOUETEIJOUTQUIALEBTSCHFCGTT
 JHYTLJEGTOHTLHCBTISTW TOUETCIABTWOSEC TFEJETIFIXEAWALTOHTLJEEOTUWYTHATT
UWSTKHRJAEGTOUIOTSEEYEBTOHTSOJEOQUTEABCESSCGTNE HJETUWYTCWXETITJWDEJTT
FUHSETQHRJSETFISTRAXAHFATGEOTPJHYWSEBTOHTCEIBTSHYEFUEJETNEGHABTOUETUHJWMHATT
FUEJETOUETQWOGTUETBJEIYEBTH TSOHHBTFWOUTFICCSTH TSOHAETIABTOHFEJSTOUIOTT
LCWOOEJEBTWATOUETCWLUOTH TOUETYHJAWALTSRATOUIOTSUHAETFWOUTITNJWCCWIAQETT
OUIOTYIBETUWYTOUWAXTH TSOHJWESTUETUIBTUEIJBTISTITQUWCBTFUEATUETSIOTNGTT
OUET WJETIABTCWSOEAEBTOHTUWSTYHOUEJTSPEIXTH TUEJHESTFUHTOJIDECEBT IJTT
IABT HRABTFHABEJSTOUIOTQUIALEBTOUEWJTCWDEST HJEDEJTFUWCETUETFHABEJEBTW TT
UWSTHFATSOEPSTFHRCBTHAETBIGTNJWALTUWYTOHTSRQUTPCIQESTHJTW TOUETJHIBTT
FHRCBTSWYPCGTQHAOWARETFWOUHROTEABTIQJHSSTDICCEGSTIABTHDEJTJWDEJSTIABTT
OUJHRLUT HJESOSTOUIOTLJEFTBIJXEJTISTOUETCWLUOTH TBIGT IBEBTIABTOUETYHHATT
IPPEIJEBTWATOUETSXGTOHTFIOQUTHDEJTUWYTFWOUTSWCEAOTEGESTOUIOTSEEYEBTOHTT
LRWBETUWST EEOTICHALTPIOUSTUETUIBTAEDEJTXAHFATEVWSOEBTRAOWCTOUETAWLUOTT
HPEAEBTOUEYTNE HJETUWYTIABTOUETSOIJSTINHDETYIJXEBTUWSTFIGTFWOUTOUEWJTT
BWSOIAOTCWLUOTOUIOTLCWOOEJEBTCWXETSWCDEJTBRSOTSQIOOEJEBTIQJHSSTITNCIQXTT
QCHOUTOUIOTQHDEJEBTOUETUEIDEASTIABTYIBETUWYT EECTNHOUTSYICCTIABTGEOTT
PIJOTH TSHYEOUWALTLJEIOEJTOUIATUWYSEC TISTW TOUETRAWDEJSETUIBTNEEATT
FIWOWALT HJTUWYTOHTFICXTOUWSTDEJGTJHIBTIABTOHTCEIJATOUIOTOUETKHRJAEGTT
FISTAHOTHACGTINHROTJEIQUWALTOUETQWOGTUETSHRLUOTNROTINHROTBWSQHDEJWALTT
OUETSOJEALOUTOUIOTCIGTFWOUWATUWYTIABTOUETQHRJILETOUIOTLJEFTFWOUTEIQUTT
SOEPTOUIOTUETOHHXTWAOHTOUETRAXAHFATFUEJETOUETPJHYWSETH TOHYHJJHFTT
FIWOEBTCWXETITCWLUOTOUIOTQHRCBTAEDEJTNETEVOWALRWSUEBTAHTYIOOEJTUHFTT
CHALTOUETAWLUOTHJTUHFT IJTOUETJHIBTOUIOTCEBTUWYTEDEJTHAFIJBTWAOHTOUETBWSOIAQETT
```

2) Frequency Analysis

With a simple C program it is possible to read the file and count how many times a certain character occurs within the ciphertext. Here is the output of the program I made (source code below):

```
character:"O" occurrences:179 frequency:08.548233%
character:"U" occurrences:149 frequency:07.115568%
character:"E" occurrences:220 frequency:10.506208%
character:"T" occurrences:414 frequency:19.770775%
character:"J" occurrences:102 frequency:04.871060%
character:"I" occurrences:123 frequency:05.873926%
character:"D" occurrences:021 frequency:01.002865%
character:"C" occurrences:068 frequency:03.247374%
character:" " occurrences:035 frequency:01.671442%
character:"W" occurrences:101 frequency:04.823305%
character:"S" occurrences:101 frequency:04.823305%
character:"L" occurrences:037 frequency:01.766953%
character:"B" occurrences:086 frequency:04.106972%
character:"F" occurrences:055 frequency:02.626552%
character:"A" occurrences:092 frequency:04.393505%
character:"Y" occurrences:032 frequency:01.528176%
character:"N" occurrences:018 frequency:00.859599%
character:"H" occurrences:121 frequency:05.778414%
character:"R" occurrences:027 frequency:01.289398%
character:"Q" occurrences:028 frequency:01.337154%
character:"X" occurrences:016 frequency:00.764088%
character:"G" occurrences:023 frequency:01.098376%
character:"P" occurrences:013 frequency:00.620821%
character:"K" occurrences:002 frequency:00.095511%
character:"M" occurrences:001 frequency:00.047755%
```

```c
#include<stdio.h>
#include<stdlib.h>

/* a simple structure to store information about the frequency analysis of the text
     */
struct frequencyAnalysis {
    int index;
    int total;
    char characters[27];
    int occurrences[27];
    float percentages[27];
};

/* a simple function that tells me if some character is already stored */
int find(char c, char chars[27]){
  int pos=-1;
  for(int i=0; i<27; i++) if(chars[i]==c) pos=i;
  return pos;
}

/* a simple function to sort the resulting percentages */
void structSort(struct frequencyAnalysis f){
  int i,j;
  float ftemp;
  char ctemp;
  int itemp;
  for(i=0; i<27; i++){
    for(j=0; j<27-i; j++){
      if(f.occurrences[j]<f.occurrences[j+1]){
        ftemp=f.percentages[j];
        f.percentages[j]=f.percentages[j+1];
        f.percentages[j+1]=ftemp;
        itemp=f.occurrences[j];
        f.occurrences[j]=f.occurrences[j+1];
        f.occurrences[j+1]=itemp;
        ctemp=f.characters[j];
        f.characters[j]=f.characters[j+1];
        f.characters[j+1]=ctemp;
      }
    }
  }
}

/* a simple function to dump on screen the resulting frequency analysis */
void structDump(struct frequencyAnalysis f){
  printf("\n|----------------------------------------|\n");
  printf("| CHARACTER |  OCCURRENCES |  FREQUENCY   |\n");
  printf("|----------------------------------------|\n");
  for(int i=0; i<(f.index-1); i++){
    if((f.occurrences[i]>=10)&&(f.percentages[i]>=10)){
      printf("|     %c     |      %d      |  %f%         |\n",
      f.characters[i], f.occurrences[i], f.percentages[i]);
    } else if(f.occurrences[i]>=10){
        printf("|     %c     |      %d      |  0%f%         |\n",
        f.characters[i], f.occurrences[i], f.percentages[i]);
    } else if(f.percentages[i]>=10){
        printf("|     %c     |      0%d      |  %f%         |\n",
        f.characters[i], f.occurrences[i], f.percentages[i]);
    } else {
      printf("|     %c     |      0%d      |  0%f%         |\n",
      f.characters[i], f.occurrences[i], f.percentages[i]);
    }
  }
  printf("|----------------------------------------|\n\n");
}

int main(){

  FILE *sourceFile;
  char sourcePath[100];
  char c;
  struct frequencyAnalysis f={0, 0, {(char) 0}, {0}, {0}};

  printf("enter source file path: ");
  scanf("%s", sourcePath);

  sourceFile=fopen(sourcePath, "r");
  if(sourceFile==NULL){
    printf("Cannot open file %s\n", sourcePath);
    exit(EXIT_FAILURE);
  }

  while((c=fgetc(sourceFile))!=EOF){
    int pos=find(c, f.characters);
    f.total+=1;
    if(pos==-1){
        f.characters[f.index]=c;
        f.occurrences[f.index]=1;
        f.index+=1;
    } else f.occurrences[pos]+=1;
  }

  f.total--;

  for(int i=0; i<(f.index-1); i++){
    f.percentages[i]=((float) f.occurrences[i]/(float) f.total)*100;
  }

  structSort(f);
  structDump(f);
  fclose(sourceFile);

  return 0;
}
```

Listing 1: frequencyAnalyzer.c

3) Decryption Attempt

After asking the internet what is the average percentage of the A-Z characters occurrence in the english language I made another C program to take the given ciphertext file and operate the substitution of characters according to those percentages. It's easy to note how the percentages are not that accurate... at this point I would look for an hint in the text, like some pattern of three letters "XYZ" and assume they can be substituted in the word "THE" or "AND" or something like that. But I cannot find any hint, plus I notice every sentence ending with "TT" (two equal characters side by side so often?) Maybe frequency analysis is not the correct approach in this case and we can call it a day. Anyway here's the result of a decryption attempt (source code below):

```
AOTEASIPTCTSECT AEOHREPHCCIMTEIAELIUDEUHAOEIERWICCEBIMENDEOHRERONGCLTSEE
IDLEOTE NCCNUTLEAOTEDISSNUESNILEAOIAECTLEIFSNRREAOTE HTCLREUOTSTEAOTEMSIRREE
UIRERAHCCEUTAEUHAOELTUEIDLEAOTEIHSEUIREFNNCEUOHCTEAOTERNGDLEN EBHSLREE
TFONTLE SNWEAOTEASTTREAOIAERANNLEIAEAOTETLMTEN EAOTE NSTRAEIDLEAOTERVYEE
MSTUEBSHMOATSEUHAOETPTSYERATKEIREAOTERGDESNRTEBTOHDLEAOTELHRAIDAEOHCCREE
IDLEAOTEROILNURERONSATDTLEUOHCTEAOTEFNCNSREN EAOTETISAOEFOIDMTLERCNUCYEE
 SNWEMSTYEANEMNCLEIREH EAOTECIDLEHARTC EUTSTEIUIVTDHDMEANEMSTTAEOHWENDEE
OHREJNGSDTYEAOIAERTTWTLEANERASTAFOETDLCTRRCYEBT NSTEOHWECHVTEIESHPTSEE
UONRTEFNGSRTEUIREGDVDNUDEYTAEKSNWHRTLEANECTILERNWTUOTSTEBTYNDLEAOTEONSHQNDEE
UOTSTEAOTEFHAYEOTELSTIWTLEN ERANNLEUHAOEUICCREN ERANDTEIDLEANUTSREAOIAEE
MCHAATSTLEHDEAOTECHMOAEN EAOTEWNSDHDMERGDEAOIAERONDTEUHAOEIEBSHCCHIDFTEE
AOIAEWILTEOHWEAOHDVEN ERANSHTREOTEOILEOTISLEIREIEFOHCLEUOTDEOTERIAEBYEE
AOTE HSTEIDLECHRATDTLEANEOHREWNAOTSERKTIVEN EOTSNTREUONEASIPTCTLE ISEE
IDLE NGDLEUNDLTSREAOIAEFOIDMTLEAOTHSECHPTRE NSTPTSEUOHCTEOTEUNDLTSTLEH EE
OHRENUDERATKREUNGCLENDTELIYEBSHDMEOHWEANERGFOEKCIFTRENSEH EAOTESNILEE
UNGCLERHWKCYEFNDAHDGTEUHAONGAETDLEIFSNRREPICCTYREIDLENPTSESHPTSREIDLEE
AOSNGMOE NSTRAREAOIAEMSTUELISVTSEIREAOTECHMOAEN ELIYE ILTLEIDLEAOTEWNNDEE
IKKTISTLEHDEAOTERVYEANEUIAFOENPTSEOHWEUHAOERHCTDAETYTREAOIAERTTWTLEANEE
MGHLTEOHRE TTAEICNDMEKIAOREOTEOILEDTPTSEVDNUDETVHRATLEGDAHCEAOTEDHMOAEE
NKTDTLEAOTWEBT NSTEOHWEIDLEAOTERAISREIBNPTEWISVTLEOHREUIYEUHAOEAOTHSEE
LHRAIDAECHMOAEAOIAEMCHAATSTLECHVTERHCPTSELGRAERFIAATSTLEIFSNRREIEBCIFVEE
FCNAOEAOIAEFNPTSTLEAOTEOTIPTDREIDLEWILTEOHWE TTCEBNAOERWICCEIDLEYTAEE
KISAEN ERNWTAOHDMEMSTIATSEAOIDEOHWRTC EIREH EAOTEGDHPTSRTEOILEBTTDEE
UIHAHDME NSEOHWEANEUICVEAOHREPTSYESNILEIDLEANECTISDEAOIAEAOTEJNGSDTYEE
UIREDNAENDCYEIBNGAESTIFOHDMEAOTEFHAYEOTERNGMOAEBGAEIBNGAELHRFNPTSHDMEE
AOTERASTDMAOEAOIAECIYEUHAOHDEOHWEIDLEAOTEFNGSIMTEAOIAEMSTUEUHAOETIFOEE
RATKEAOIAEOTEANNVEHDANEAOTEGDVDNUDEUOTSTEAOTEKSNWHRTEN EANWNSSNUEE
UIHATLECHVTEIECHMOAEAOIAEFNGCLEDTPTSEBTETVAHDMGHROTLEDNEWIAATSEONUEE
CNDMEAOTEDHMOAENSEONUE ISEAOTESNILEAOIAECTLEOHWETPTSENDUISLEHDANEAOTELHRAIDFTEE
```

```c
#include<stdio.h>
#include<stdlib.h>

int main(){
  FILE *sourceFile, *destFile;
  char sourcePath[100], destPath[100];
  char ch;
  char sub;

  printf("enter source file path: ");
  scanf("%s", sourcePath);

  sourceFile=fopen(sourcePath, "r");
  if(sourceFile==NULL){
    printf("Cannot open file %s\n", sourcePath);
    exit(EXIT_FAILURE);
  }

  printf("enter destination file path: ");
  scanf("%s", destPath);

  destFile=fopen(destPath, "w");
  if(destPath==NULL){
    printf("Cannot open file %s\n", destPath);
    fclose(sourceFile);
    exit(EXIT_FAILURE);
  }

  while((ch=fgetc(sourceFile))!=EOF){
    if(ch=='T') sub='E';
    else if(ch=='E') sub='T';
    else if(ch=='O') sub='A';
    else if(ch=='U') sub='O';
    else if(ch=='I') sub='I';
    else if(ch=='H') sub='N';
    else if(ch=='J') sub='S';
    else if(ch=='W') sub='H';
    else if(ch=='S') sub='R';
    else if(ch=='A') sub='D';
    else if(ch=='B') sub='L';
    else if(ch=='C') sub='C';
    else if(ch=='F') sub='U';
    else if(ch=='L') sub='M';
    else if(ch=='Y') sub='W';
    else if(ch=='Q') sub='F';
    else if(ch=='R') sub='G';
    else if(ch=='G') sub='Y';
    else if(ch=='D') sub='P';
    else if(ch=='N') sub='B';
    else if(ch=='X') sub='V';
    else if(ch=='P') sub='K';
    else if(ch=='K') sub='J';
    else if(ch=='M') sub='Q';
    else sub=ch;
    fputc(sub, destFile);
  }

  fclose(sourceFile);
  fclose(destFile);
  return 0;
}
```

Listing 2: characterSubstitution.c