Project 1

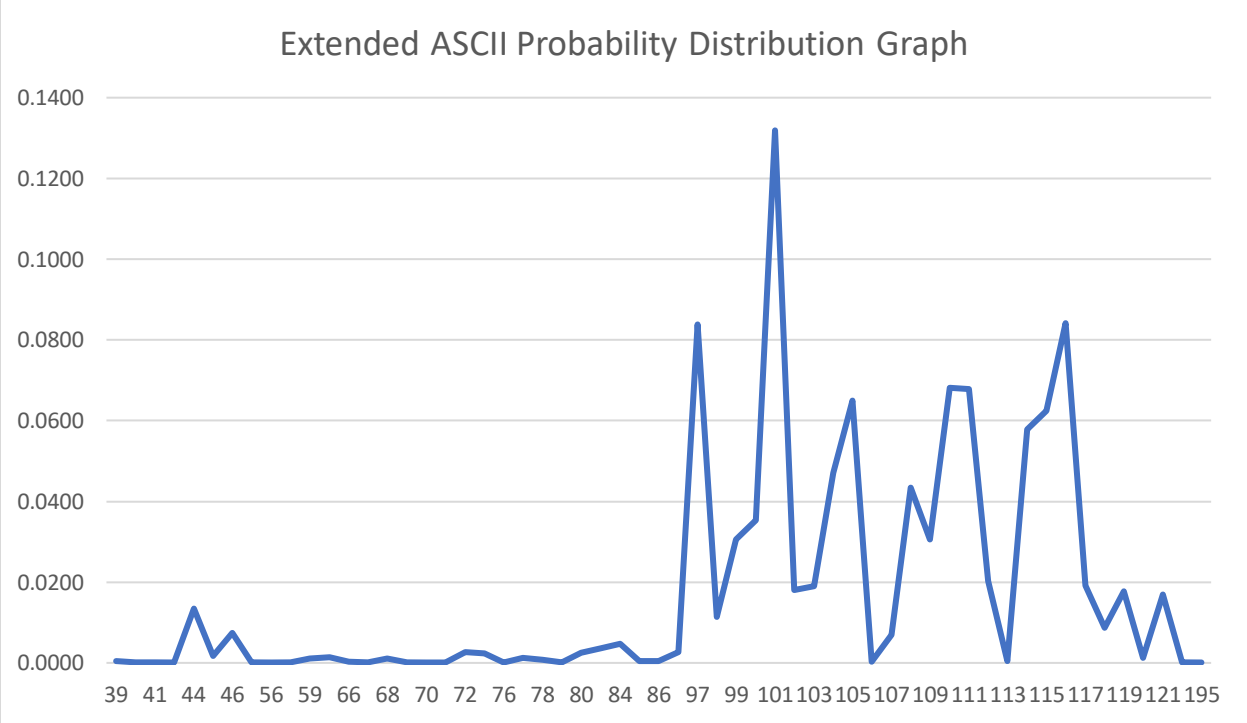
CECS 564

Spring 2020

Roman Millem

PROBABILITY DISTRIBUTION:

The graph included below shows the probability distribution of the 256 ASCII characters. This data was calculated using the provided “Darwin.txt” and “text data.txt” files. The function used to do the calculations is included in the source file and is named “PDF_ASCII”. The probability distributions are calculated for each text file and then averaged for the final result. This function is used to determine the probability distribution array used for attacking a Vigenere cipher and is called every time the program is executed to avoid hardcoding. This also allows for more text files to be added to further increase the accuracy of the calculated distribution.

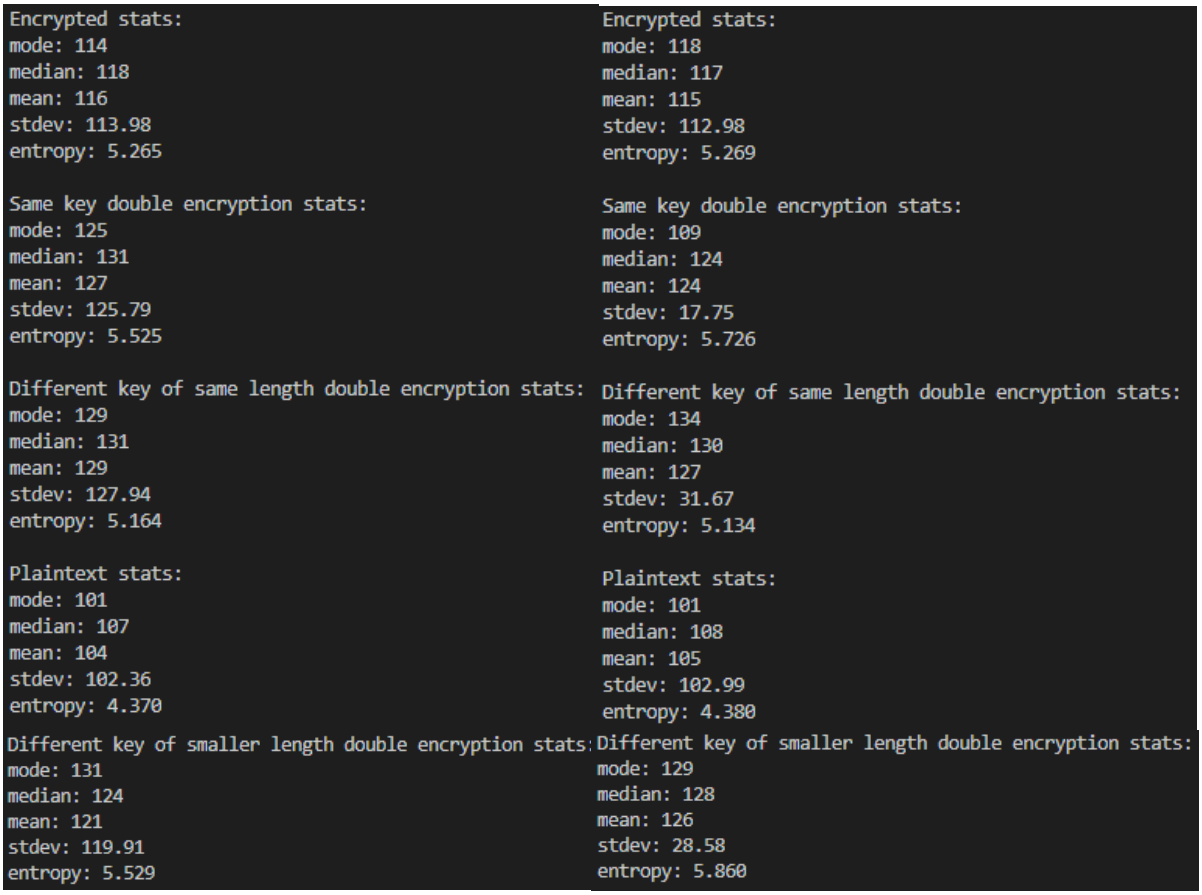


Note that the 256 ASCII character values are defined by the operating system (Windows 10), IDE (Visual Studio Code), and compiler (g++) used for the project. The CP 473 encoding scheme is used this system. A “PrintCodes” function is included in the source file that was used to output the determined characters for ASCII values 0-255. The results are shown in the figure provided below. These characters were used for all encryption, decryption, and attack techniques.

0:	33:	!	65:	A	97:	a	129:	ü	161:	í	193:	┴	225:	ß
1:	34:	"	66:	B	98:	b	130:	é	162:	ó	194:	┬	226:	Γ
2:	35:	#	67:	C	99:	c	131:	â	163:	ú	195:	├	227:	π
3:	36:	$	68:	D	100:	d	132:	ä	164:	ñ	196:	─	228:	Σ
4:	37:	%	69:	E	101:	e	133:	à	165:	Ñ	197:	┼	229:	σ
5:	38:	&	70:	F	102:	f	134:	å	166:	ª	198:	╞	230:	µ
6:	39:	'	71:	G	103:	g	135:	ç	167:	º	199:	╟	231:	τ
7:	40:	(	72:	H	104:	h	136:	ê	168:	¿	200:	╚	232:	Φ
8:	41:	)	73:	I	105:	i	137:	ë	169:	⌐	201:	╔	233:	Θ
9:	42:	*	74:	J	106:	j	138:	è	170:	¬	202:	╩	234:	Ω
10:	43:	+	75:	K	107:	k	139:	ï	171:	½	203:	╦	235:	δ
11:	44:	,	76:	L	108:	l	140:	î	172:	¼	204:	╠	236:	∞
12:	45:	-	77:	M	109:	m	141:	ì	173:	¡	205:	═	237:	φ
13:	46:	.	78:	N	110:	n	142:	Ä	174:	«	206:	╬	238:	ε
14:	47:	/	79:	O	111:	o	143:	Å	175:	»	207:	╧	239:	∩
15:	48:	0	80:	P	112:	p	144:	É	176:	░	208:	╨	240:	≡
16:	49:	1	81:	Q	113:	q	145:	æ	177:	▒	209:	╤	241:	±
17:	50:	2	82:	R	114:	r	146:	Æ	178:	▓	210:	╥	242:	≥
18:	51:	3	83:	S	115:	s	147:	ô	179:	│	211:	╙	243:	≤
19:	52:	4	84:	T	116:	t	148:	ö	180:	┤	212:	╘	244:	⌠
20:	53:	5	85:	U	117:	u	149:	ò	181:	╡	213:	╒	245:	⌡
21:	54:	6	86:	V	118:	v	150:	û	182:	╢	214:	╓	246:	÷
22:	55:	7	87:	W	119:	w	151:	ù	183:	╖	215:	╫	247:	≈
23:	56:	8	88:	X	120:	x	152:	ÿ	184:	╕	216:	╪	248:	°
24:	57:	9	89:	Y	121:	y	153:	Ö	185:	╣	217:	┘	249:	∙
25:	58:	:	90:	Z	122:	z	154:	Ü	186:	║	218:	┌	250:	·
26:	59:	;	91:	[	123:	{	155:	¢	187:	╗	219:	█	251:	√
27:	60:	<	92:	\	124:	|	156:	£	188:	╝	220:	▄	252:	ⁿ
28:	61:	=	93:	]	125:	}	157:	¥	189:	╜	221:	▌	253:	²
29:	62:	>	94:	^	126:	~	158:	₧	190:	╛	222:	▐	254:	■
30:	63:	?	95:	_	127:	⌂	159:	ƒ	191:	┐	223:	▀	255:	
31:	64:	@	96:	`	128:	Ç	160:	á	192:	└	224:	α		
32:														

SUMMARY STATISTICS:

Comprehensive summary statistics were gathered from plaintext, encrypted text, double encrypted text using the same key, and double encrypted text using a different key of the same length. The left figure below displays the summary statistics using “text data.txt” with primary encryption key “roman”, secondary same length key “donut”, and tertiary smaller length key “code”. The right figure below displays the summary statics using “Darwin.txt” with primary key “vigenere”, secondary same length key “computer”, and tertiary smaller length key “spider”.



As you can see, the Vigenere encryption increases security as encrypting the plaintext increases the values of all summary statistics including entropy. The mode, median, and mean simply increasing makes sense as encryption is done over the entire 256 character ASCII set. As most characters in the plaintext are A-z (65-122), encrypting with a key of characters a-z (97-122) is not likely to result in an encrypted character below ‘A’ (65). Double encrypting the plaintext is not always beneficial. Encrypting with two keys of different lengths will return the best results, especially if the two key lengths greatest common multiple is 1. This makes logical sense since the security of the Vigenere crypto system is dependent on key length grouping and the index of coincidence. Changing key lengths will shift groups twice in two different manners, increasing security greatly. Double encrypting with the same key will also increase security

albeit minimally. However, double encrypting with a different key of the same length will decrease the entropy of the result. This too makes sense as the grouping is the same so the index of coincidence could be ran once to find the key length. Also, different keys could result in overlapping shifts.

ERRORS:

I encountered an unsolvable error when dealing with extended ASCII characters. My program worked fine when incrementing a character with ASCII value less than 127 into a character with ASCII value > 127. However, when reading a character with value > 127, the compiler would split it into two seemingly random characters. Below is a screenshot of this problem.

```
int z = int('ü');
cout << z <<endl;
```

This code would result in the following compiling error.

```
warning: multi-character character constant [-Wmultichar]
```

```
int z = int('╠╝');
            ^~~~
```

I tried many ways around this, including variable types 'wchar_t', 'uint8_t', and 'uint16_t'. All these resulted in overflow errors. So, given an encrypted text file containing extended ASCII characters, I could not successfully decrypt it as extra characters would be inserted into the string. However, when I encrypted a text file and then attacked the resulting encrypted string, my program would accurately determine the decryption key.

Again, when an encrypted string is hardcoded, the output is altered by the compiler. It seems there is no work around for this on my system.

```
string s = "Y}üsrhwxlcÇé";
cout << s << endl;
```

The above code will result in the following ouput:

```
PS C:\Users\MillemRomanLee\Documents\Classes\Spring 20\564-Cryptography\Workspace\Project1> ./main.exe
Y}╠╝srhwxlc├ç├¬
```

transforming the string of length 12 into a string of length 15 as there are 3 ASCII characters with a value > 127.

SCREENSHOTS:

```
PS C:\Users\MillemRomanLee\Documents\Classes\Spring 20\564-Cryptography\Workspace\Project1> ./main.exe
Original Message: HouseWillVoteWednesdaytoSendImpeachmentArticles,PelosiSays.Thespeaker,whohaswaitednearl
teonWednesdaytosendtheSenateimpeachmentchargesagainstPresidentTrump,allowingalong-awaitedtrialtobegin,Spe
nputontrialintheSenate.Inaclosed-doorgatheringwithDemocraticlawmakersonTuesdaymorning,Ms.Pelosidetailedhe
Theofficialswhodescribedherprivateremarksspokeonconditionofanonymity.Unlessthingschange,hertimetablemeans
sentingthemandpromptingatrialtocommence.Thespeakersaidshewasnotyetreadytosharethenamesofthelawmakersshewo

Encrypted Message: Y}üsrhwxlcÇéqWru|qsqrçÇo`v|pIzüsmcu~sztNâéucyvü8Pr}}i`rç.aysprryqr9êv{hnäàmiüvrzenâzàa
|às{ndvrzeÇuoàt|äszdüys_e{réqizüsmcu~sztpyo~gräosavüÇPvüudré`ré~~8ay}}âi{xoxo{x;mwnzéqdüâwmlüÇpqgv:_prryq
qn}åé{nüâwmlvéte`v|mtr?Wzap}}eq>r{oxoÇhrâwzgäzétDr~}ornàwolnê{mkrâü{naåsdnè{{r{z|s,Zä<\eyÇüudràoulruvqr}}
?bte|wtucvrzwuÇrqspâwneqys~pzämtrâsya|üp||s{npÇ|piüz}zosr|{nå~wÇy;f|xeÇäéti{xüohnuq,uvÇÇizvémbyv{qa{äétaü
eÇv|Çi{xétezr|ppÇ{|tvumtzoxt|t}ymrqq.aysprryqrÇrwpsuvàms{Çéàeüâsmdåà}hnâsÇhroyeÇÇtÇhr}oâmn|s~sÇysâoé}reyv

Decrypted Message: HouseWillVoteWednesdaytoSendImpeachmentArticles,PelosiSays.Thespeaker,whohaswaitednear
oteonWednesdaytosendtheSenateimpeachmentchargesagainstPresidentTrump,allowingalong-awaitedtrialtobegin,Sp
enputontrialintheSenate.Inaclosed-doorgatheringwithDemocraticlawmakersonTuesdaymorning,Ms.Pelosidetailedh
.Theofficialswhodescribedherprivateremarksspokeonconditionofanonymity.Unlessthingschange,hertimetablemean
esentingthemandpromptingatrialtocommence.Thespeakersaidshewasnotyetreadytosharethenamesofthelawmakersshew
IOC[1] = 0.026
IOC[2] = 0.023
IOC[3] = 0.027
IOC[4] = 0.021
IOC[5] = 0.056
Suggested key length: 5

Suggested key: r o m a n
```

The suggested key is determined using the ‘AttackVigenere’ function run on the encrypted text and the suggested key length. The attack is ran independent of the set encryption key to originally encrypt the plaintext.