

Final Project

Group Members:

Christopher Erhard (cje5298@psu.edu)

James Zukowski (jjz5192@psu.edu)

Arib Hyder (aah5469@psu.edu)

Anoop Soodini (ars6375@psu.edu)

Professor:

Mark Mahon

Course:

CMPEN 462 - Wireless Security

Spring 2021

Abstract	3
Flow Chart	4
Introduction	5
Functional Blocks	5
Main	5
Encrypt	5
Decrypt	6
Problems	6
Testing	7
Result Images	8
Work Performed and Roles	11

Abstract

For our final project, we built off what we have learned in class regarding cryptography and modulation. Our goal was to simulate an encryption/decryption program in Python which would also modulate and demodulate the encrypted binary data. For the encryption function, the program prompts the user to input a text file, converts this text file from characters to binary, encrypts the binary data with a key, prompts the user to choose between 3 modulation schemes, modulates this data by converting it into OFDM symbols, and finally stores the OFDM symbols along with the encryption key and modulation scheme into a csv file. For the decryption portion, the program prompts the user to input a csv file that was outputted by the encryption function of our project, converts the OFDM symbols back into encrypted binary data using the given modulation scheme, decrypts this data using the given key, converts the decrypted binary data to characters, and stores the output text result in a text file. To build this, we used Python 3 along with several libraries.

Flow Chart

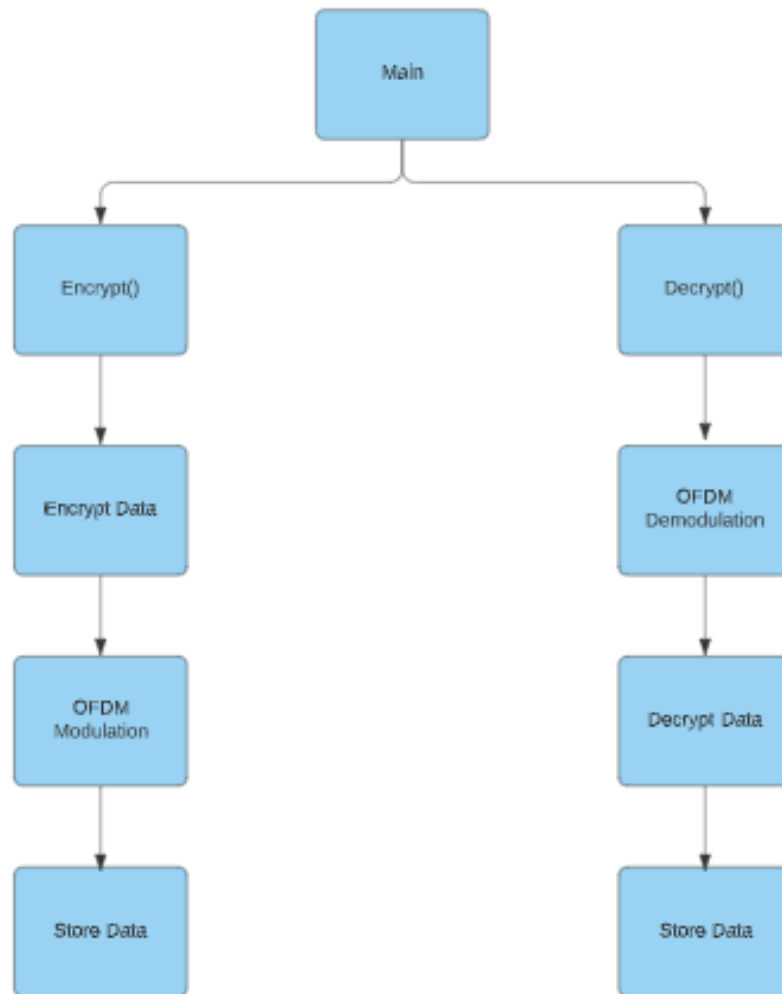


Figure 1: Application Flowchart

Introduction

Our final project's main goal was to build a user-friendly application that takes an input from the user, encrypts it, and then decrypts it whenever the user wants. In order to do this we began by looking over the two mini-projects that we built this semester. We knew that based on the two mini-projects, we would be able to adapt the code to work together in order to encrypt and decrypt data entered by the user. The program starts by asking if the user would like to encrypt or decrypt a file. Next, the user can upload a file for the application to encrypt or decrypt. The user will then be asked what modulation or demodulation technique they want to be used. Finally, the program will store the data into a separate file that can be accessed by the user.

Functional Blocks

Main

The main function controls the overall flow of our application. The function is encompassed in a while loop to continuously run until the user no longer wishes to encrypt or decrypt a file. The application first asks the user if they would like to encrypt or decrypt a file. The application then asks for the name of the file and searches for it in the current working directory. If the user wishes to decrypt a file, the modulation type and key are both taken from it and passed into the decrypt function. After every iteration of encryption or decryption, the application will ask the user if they wish to encrypt or decrypt another file. If they do not wish to, the program ends successfully.

Encrypt

For our encrypt function, we are passed a text file and the name of that file. From here, we extract the name of the file without the extension for naming output files (this is stored in a variable is called "base") and create variables to store our data for later in the function. Next we loop through our input file to store all of the text in a string. From here, we convert this string of ASCII characters to its equivalent string of binary characters for encryption and store this in `base_toBinary.txt`. Next, using Fernet encryption from the Python cryptography library, we generate a key, encode our data using the `encode()` function (this is required for using the Fernet encrypt function), and encrypt our data using the `fernet.encrypt()` function. After this, we convert our encoded encrypted data back from encoded characters and back into encrypted binary bits for modulation. We store our encrypted binary data in a text file called `base_encryptedBinary_in.txt`. After this, we have our encrypted binary data so all that is

left to do is modulate this data. The user is prompted to choose between BPSK, QPSK, and 16QAM modulation schemes. Once one is chosen, we loop through the bits of our encrypted binary data to convert the bits to OFDM symbols. Finally, when this is finished we store the list of OFDM symbols, the chosen modulation scheme, and the encryption key (for use later on with the decryption function) and a csv file called `base_MODULATION.csv` (Where MODULATION is the modulation scheme chosen for that encryption) for output. This output will be the input for our decrypt function.

Decrypt

For our decrypt function, we pass in the name of the file (this is stored in a variable called “base”), a list of OFDM symbols, the modulation type, and the encryption key (all generated by the encrypt function and stored in its output). The encryption key is then UTF-8 encoded for use with the Python cryptography functions later. The function next checks the modulation type and demodulates the input list of OFDM symbols by converting them into binary bits based on the modulation type. After this step, our demodulated binary data is still encrypted but no longer in the form of OFDM symbols, and this encrypted binary data is now stored in a file called `base_encryptedBinary_out.txt`. Next, we convert our provided encoded key into a Fernet key for decryption using the Fernet function. We next convert our binary data from binary bits to ASCII characters for use with the Python cryptography decryption function. We then UTF-8 encode these ASCII characters using the `encode()` function and decrypt this data using the `fernet.decrypt()` function. Our decrypted binary data is then stored in a file called `base_decryptedBinary.txt`. Finally, we loop through our decrypted binary data and convert the bits to 8-bit ASCII characters to receive our final desired decrypted text. This final decrypted text is then stored in an output text file called `base_decryptedText.txt`.

Problems

Our initial plan was to implement this application on a Raspberry Pi. However, due to time constraints and shipping issues, we were unable to receive the Raspberry Pi in a timely fashion. Due to this, we decided to implement the application in a fashion that would be similar to how it would behave on the device. To test the application, we used the Windows and Mac command lines and ran the program using the following command: `python3 FinalProject.py`. We kept our input files in the same working directory as the Python file.

Other problems encountered in this project had to do mostly with going from text to bits and back. Converting our original encryption text input from characters to binary bits was not too difficult, but once the `fernet.encrypt()` and `fernet.decrypt()` functions were introduced more problems soon arose. When trying to use them initially, we did not realize that we would not be able to just pass our string of binary characters to the `fernet.encrypt()` function for encryption. Upon learning more about these functions, we learned that we needed to encode the input to these functions to use them. This also introduced the problem of converting the encrypted data back to binary for modulation. Once this was solved, a similar process was done in the decryption function that was essentially the reverse of the solution found to fix this problem in the encryption function.

One last small problem we encountered was in installing the appropriate Python libraries needed for our project. We found that using the Python pip tool was the best way to solve this.

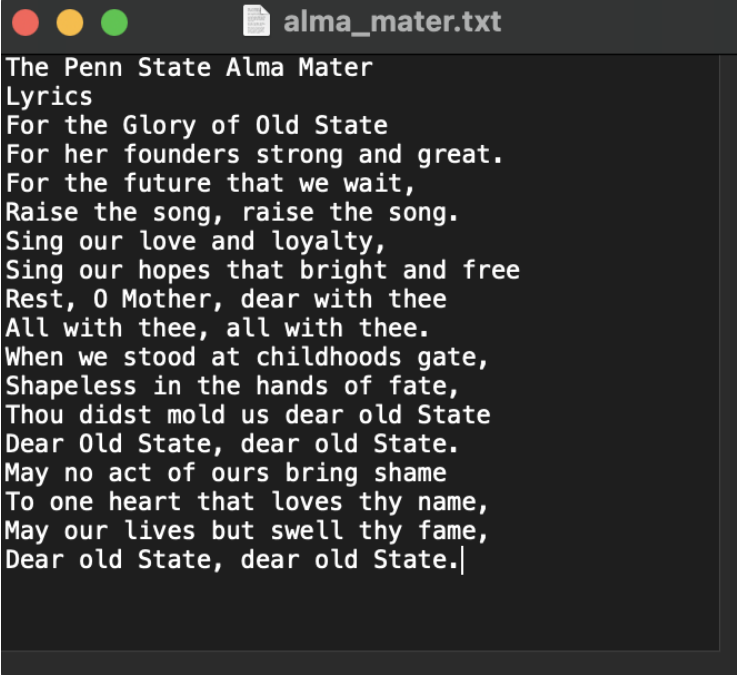
Results and Functionality

Testing

In order to extensively test the functionality of our program, we began by using a couple of different text files. The first test we ran was using the Penn State Alma Mater. We had to check to make sure that the result of our decrypt function is equivalent to the input text file. We also stored our results at several points during our functions. For example, we stored the binary output after encryption and before modulation in our encrypt function and also our encrypted binary data after demodulation in our decrypt function to compare similarity. We did this to make sure that we were getting the same demodulated data from our decrypt function as we were getting before modulation in our encrypt function. This helped us to narrow down where things were going wrong if we didn't receive the desired output (i.e. if these two were the same, our modulation/demodulation were correct and the error lies elsewhere). We also did something similar, storing the initial binary data received from converting our input text to binary in encrypt to compare to the binary in our decrypt right before converting from binary to characters for our final output. This would show us whether errors were occurring in the step where we went from binary to ASCII characters or not. Once we were sure our application was functioning as expected, we tried with a much larger input file, a text file of the Declaration of Independence. This file also was encrypted and decrypted as expected. Both of these input files are included in our submission. We omitted screenshots of the test of the Declaration of Independence due to their large

size, but our results can be confirmed by running our application using this input as provided.

Result Images



```
The Penn State Alma Mater
Lyrics
For the Glory of Old State
For her founders strong and great.
For the future that we wait,
Raise the song, raise the song.
Sing our love and loyalty,
Sing our hopes that bright and free
Rest, O Mother, dear with thee
All with thee, all with thee.
When we stood at childhoods gate,
Shapeless in the hands of fate,
Thou didst mold us dear old State
Dear Old State, dear old State.
May no act of ours bring shame
To one heart that loves thy name,
May our lives but swell thy fame,
Dear old State, dear old State.
```

Figure 2: Input text file


```

Chriss-MacBook-Pro-2:CPMENA462 chrisherhard$ ls
FinalProject.py README.md alma_mater.txt declaration.txt
Chriss-MacBook-Pro-2:CPMENA462 chrisherhard$ python3 FinalProject.py
Welcome to our Encryption/Decryption Application!

Would you like to Encrypt (1) or Decrypt (2) a file?:
1
Please enter the name of the file you wish to encrypt or decrypt:
alma_mater.txt
What modulation scheme would you like to use to encrypt your file?
Enter (1) for BPSK, (2) for QPSK, or (3) for 16QAM:
1
Successfully Encrypted file! Encrypted binary data is stored in alma_mater_encryptedBinary_in.txt.
OFDM Symbols, modulation type, and encryption key are stored in alma_mater_BPSK.csv

Please enter (0) to encrypt/decrypt again. If not, type any other key.
x
Encrypt/Decrypt Driver Exited Successfully.
Chriss-MacBook-Pro-2:CPMENA462 chrisherhard$ ls
FinalProject.py alma_mater_BPSK.csv declaration.txt
README.md alma_mater_encryptedBinary_in.txt filekey.key
alma_mater.txt alma_mater_toBinary.txt
Chriss-MacBook-Pro-2:CPMENA462 chrisherhard$ python3 FinalProject.py
Welcome to our Encryption/Decryption Application!

Would you like to Encrypt (1) or Decrypt (2) a file?:
2
Please enter the name of the file you wish to encrypt or decrypt:
alma_mater_BPSK.csv
Successfully Decrypted file! Results are stored in alma_mater_BPSK_decryptedText.txt

Please enter (0) to encrypt/decrypt again. If not, type any other key.
x
Encrypt/Decrypt Driver Exited Successfully.
Chriss-MacBook-Pro-2:CPMENA462 chrisherhard$ ls
FinalProject.py alma_mater_BPSK_decryptedBinary.txt alma_mater_toBinary.txt
README.md alma_mater_BPSK_decryptedText.txt declaration.txt
alma_mater.txt alma_mater_BPSK_decryptedBinary_out.txt filekey.key
alma_mater_BPSK.csv alma_mater_encryptedBinary_in.txt
Chriss-MacBook-Pro-2:CPMENA462 chrisherhard$

```

Figure 3: Running Application

```

1
2
3
4
5
6
7
8
9
10
11
12
13
14
15
16
17
18
19
20
21
22
23
24
25
26
27
28
29
30
31
32
33
34
35
36
37
38
39
40
41
42
43
44
45
46
47
48
49
50
51
52
53
54
55
56
57
58
59
60
61
62
63
64
65
66
67
68
69
70
71
72
73
74
75
76
77
78
79
80
81
82
83
84
85
86
87
88
89
90
91
92
93
94
95
96
97
98
99
100
101
102
103
104
105
106
107
108
109
110
111
112
113
114
115
116
117
118
119
120
121
122
123
124
125
126
127
128
129
130
131
132
133
134
135
136
137
138
139
140
141
142
143
144
145
146
147
148
149
150
151
152
153
154
155
156
157
158
159
160
161
162
163
164
165
166
167
168
169
170
171
172
173
174
175
176
177
178
179
180
181
182
183
184
185
186
187
188
189
190
191
192
193
194
195
196
197
198
199
200
201
202
203
204
205
206
207
208
209
210
211
212
213
214
215
216
217
218
219
220
221
222
223
224
225
226
227
228
229
230
231
232
233
234
235
236
237
238
239
240
241
242
243
244
245
246
247
248
249
250
251
252
253
254
255
256
257
258
259
260
261
262
263
264
265
266
267
268
269
270
271
272
273
274
275
276
277
278
279
280
281
282
283
284
285
286
287
288
289
290
291
292
293
294
295
296
297
298
299
300
301
302
303
304
305
306
307
308
309
310
311
312
313
314
315
316
317
318
319
320
321
322
323
324
325
326
327
328
329
330
331
332
333
334
335
336
337
338
339
340
341
342
343
344
345
346
347
348
349
350
351
352
353
354
355
356
357
358
359
360
361
362
363
364
365
366
367
368
369
370
371
372
373
374
375
376
377
378
379
380
381
382
383
384
385
386
387
388
389
390
391
392
393
394
395
396
397
398
399
400
401
402
403
404
405
406
407
408
409
410
411
412
413
414
415
416
417
418
419
420
421
422
423
424
425
426
427
428
429
430
431
432
433
434
435
436
437
438
439
440
441
442
443
444
445
446
447
448
449
450
451
452
453
454
455
456
457
458
459
460
461
462
463
464
465
466
467
468
469
470
471
472
473
474
475
476
477
478
479
480
481
482
483
484
485
486
487
488
489
490
491
492
493
494
495
496
497
498
499
500
501
502
503
504
505
506
507
508
509
510
511
512
513
514
515
516
517
518
519
520
521
522
523
524
525
526
527
528
529
530
531
532
533
534
535
536
537
538
539
540
541
542
543
544
545
546
547
548
549
550
551
552
553
554
555
556
557
558
559
560
561
562
563
564
565
566
567
568
569
570
571
572
573
574
575
576
577
578
579
580
581
582
583
584
585
586
587
588
589
590
591
592
593
594
595
596
597
598
599
600
601
602
603
604
605
606
607
608
609
610
611
612
613
614
615
616
617
618
619
620
621
622
623
624
625
626
627
628
629
630
631
632
633
634
635
636
637
638
639
640
641
642
643
644
645
646
647
648
649
650
651
652
653
654
655
656
657
658
659
660
661
662
663
664
665
666
667
668
669
670
671
672
673
674
675
676
677
678
679
680
681
682
683
684
685
686
687
688
689
690
691
692
693
694
695
696
697
698
699
700
701
702
703
704
705
706
707
708
709
710
711
712
713
714
715
716
717
718
719
720
721
722
723
724
725
726
727
728
729
730
731
732
733
734
735
736
737
738
739
740
741
742
743
744
745
746
747
748
749
750
751
752
753
754
755
756
757
758
759
760
761
762
763
764
765
766
767
768
769
770
771
772
773
774
775
776
777
778
779
780
781
782
783
784
785
786
787
788
789
790
791
792
793
794
795
796
797
798
799
800
801
802
803
804
805
806
807
808
809
810
811
812
813
814
815
816
817
818
819
820
821
822
823
824
825
826
827
828
829
830
831
832
833
834
835
836
837
838
839
840
841
842
843
844
845
846
847
848
849
850
851
852
853
854
855
856
857
858
859
860
861
862
863
864
865
866
867
868
869
870
871
872
873
874
875
876
877
878
879
880
881
882
883
884
885
886
887
888
889
890
891
892
893
894
895
896
897
898
899
900
901
902
903
904
905
906
907
908
909
910
911
912
913
914
915
916
917
918
919
920
921
922
923
924
925
926
927
928
929
930
931
932
933
934
935
936
937
938
939
940
941
942
943
944
945
946
947
948
949
950
951
952
953
954
955
956
957
958
959
960
961
962
963
964
965
966
967
968
969
970
971
972
973
974
975
976
977
978
979
980
981
982
983
984
985
986
987
988
989
990
991
992
993
994
995
996
997
998
999
1000
1001
1002
1003
1004
1005
1006
1007
1008
1009
1010
1011
1012
1013
1014
1015
1016
1017
1018
1019
1020
1021
1022
1023
1024
1025
1026
1027
1028
1029
1030
1031
1032
1033
1034
1035
1036
1037
1038
1039
1040

```

Figure 4: Input converted to binary in encrypt (not yet encrypted)

	A	B	C	D
1	(0.7071067811865475+0.7071067811865475j)	(-0.7071067811865475-0.7071067811865475j)	(-0.7071067811865475-0.7071067811865475j)	(0.7071067811865475+0.7071067811865475j)
2	BPSK			
3	wmidhz2RxojuQuKvI-QpL5tdonaz2kx_DovbOier_tYz=			
4				
5				

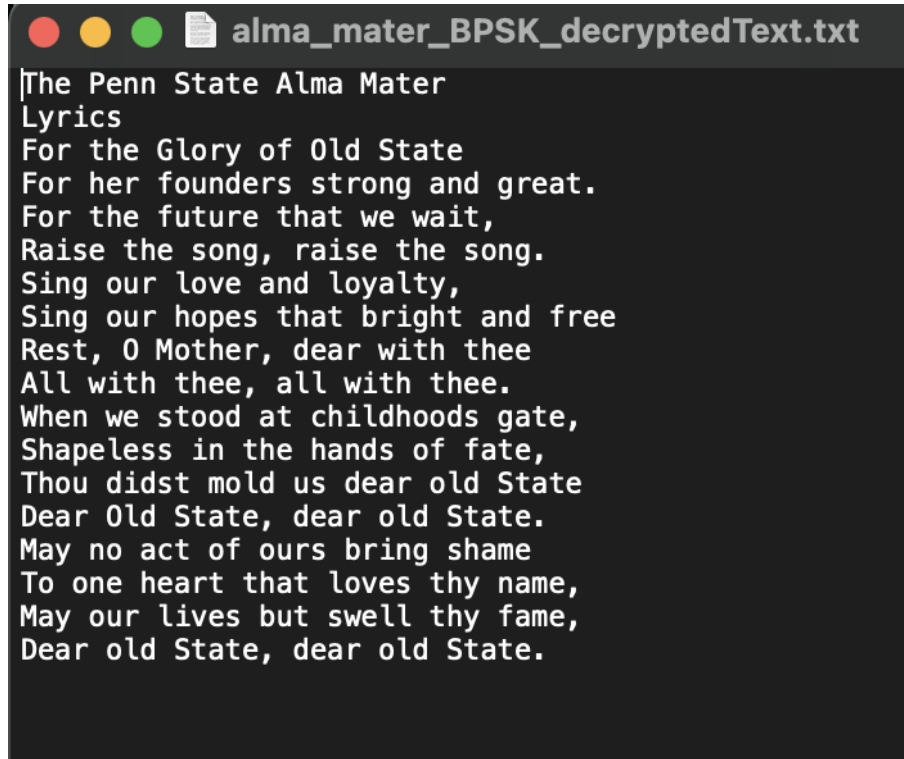
Figure 5: Output csv file produced by encrypt function (first row contains OFDM symbols, second row contains modulation scheme, third row contains encryption/decryption key)

```

010101000110100001100101001000000100000110010101101100110110001000000101001101101000110000101110
1000110010100100000010000010110100011011010100001001000000100110101000010111010001100101011001000
0010100100110001110010111001001100011011001100001010010001100110111011001000100000011010
00110100001100101001000000100011101101000110111011001001110010010000001101110100110001000000100
1111011011000110010001000000100010110100011000110100011001000110011011101101101101100100
010000011010000110010101110010001000000110011001101101101010110011001000110010111001001100
11001000000110011011010001110010011011101101100110011001000000110000101101100110010000100000011
0011101110010011001010110000101101000010100100011001101110110010001000000111010001101000
011001010010000001100110011101011010001101010110010011001000000110100011010000110000101110
100001000000111011101001001000000111011010000101101001110100001011000001010010100000101
10100101110011011001010010000011101000110100001100101011011101101100110011001100110110
0001000000111001001100001011010010110010100100000011101000110100011001010010000001100110110
1111011011100110011100100101001010110110011001100100000011011110110101011100100100100
010000001101100011011110110110010100100000011000010110110011001000010000001101100011011101110
01011000010101100011101000111100100101000000101001011011010010110110011001100100000110111011
101010111001000100000011010000110010101100110011001000000110100011010000110000101110100
00100000011000100111001001101001011011010000111010001000000110000101101110011001000010000001100
110011100100110010101100101000100011001010110011011010000101000010000001001111010000001
0011010110111011011010001101000011001010110010001100001000000110010001100101100001011100100010000
0011101101101001011101000110100001101000011001011001010001100101011001000010101100001011000110
11000010000001101101101001011101000110100001000000111010001101000011001010110010101100001000000
1100001010110001101100001000000111011011010010110100011010000100000011010001101000011001011001
0100101110000010100101011011010000110010101110001000000111011011001010010000001100101110100011
011110110111101100100001000000110000101110100001000000110011011010000110100101101000110010001101000
01101111011011101100100011100110010000110011101100001011010001100101100000010100101001101101
0000110000101110000011001010110110001100101110011001100100000011010011011011000100000011010001
1010000110010010001000000110100001100001011011001100100011100110011001100110011001100110011001
001100001010010001000110010101100001011001000100000010011110110110001100100001000000101001101100011
00001011101000110010101100001000000110010001100101000010111010001100001000000110111011010001100100
00100000010100110111010001100001011101000110010001100100011011001000100000011011101000110010000001101
11001101110110010000001100001011000110110100010000001101110110011000100000011011101101011011001001
1100111001000000110001001110010011011001100110010000001100110110100001100001011010101010010010
1000001010010101000110111001000000110111011011001100100000011010000110010101100001011100100111
010000100000011010001101000011000010110100011001101101101100110010010110011001000000
11010001101000011101000110000001101100100001011010100101001011000000101001001101010000101110
01001000000110111011010101110010001000000110100011010010111011001100110010000001100010011
101010111010000100000011001101110110110010101100011010000100000011010001101000011100100100000
01100110011000010110101011001010010100000010010001100101100010111001000100000010111101101
10001100100001000000100101110100011000010111010001100101000110000100000011001000110010110000101
1100100010000001101110110110001100100010000001010011011010001011101000110010100101110

```

Figure 6: Binary data after demodulation and decryption in decrypt function (final step before converting to ASCII)



```
alma_mater_BPSK_decryptedText.txt
The Penn State Alma Mater
Lyrics
For the Glory of Old State
For her founders strong and great.
For the future that we wait,
Raise the song, raise the song.
Sing our love and loyalty,
Sing our hopes that bright and free
Rest, O Mother, dear with thee
All with thee, all with thee.
When we stood at childhoods gate,
Shapeless in the hands of fate,
Thou didst mold us dear old State
Dear Old State, dear old State.
May no act of ours bring shame
To one heart that loves thy name,
May our lives but swell thy fame,
Dear old State, dear old State.
```

Figure 7: Final output after decryption

Work Performed and Roles

Task	Completed by
Main()	Anoop Soodini, James Zukowski, Chris Erhard
Encrypt()	Anoop Soodini, James Zukowski, Arib Hyder
Decrypt()	Chris Erhard, Arib Hyder
Encryption Testing	Chris Erhard, Anoop Soodini, James Zukowski
Decryption Testing	Chris Erhard, Anoop Soodini, James Zukowski
Functionality Testing	James Zukowski, Anoop Soodini, Chris Erhard
Writeup	Chris Erhard, Anoop Soodini, James Zukowski, Arib Hyder

Table 1: Worked Performed and Roles

While writing the script for the program, we began by first designing a flowchart of what needs to get done and then we split up the work as such. James and Anoop began writing the driver code in the main function, and Chris worked on improving the functionality of it. Encrypt and Decrypt were written by all members collectively. For testing, Chris took care of testing the encryption and decryption, while James and Anoop focused on testing the user interface. The write-up was done by all members of the group and work was divided evenly throughout.

While writing the program, we used Discord and text messages as our main form of communication. On Discord, we talked to each other and simultaneously shared our screens to work collaboratively. We created a Github repository to manage our code and keep track of commits and changes to our code. While working, we would help each other with individual sections and commit changes when finished so the next person could work on their portion. We used texting to decide when to meet and if anyone had any questions about any part of the project. To complete the report, we created a shared Google Doc and worked on together while on a Discord call.