

```
1  """
2  *FTPChat - Encrypted FTP-based Messaging Protocol
3  *Version 1.3 - September 2025
4  *Author: Ahmed Omar Saad
5  *Contact: ahmedomardev@outlook.com
6  *License: Custom MIT - Commercial use requires written permission
7  *All rights to the name "FTPChat" and its protocol specification are retained by the
  author.
8  *For more information, Check the `LICENSE` file."""
9
10 from datetime import datetime
11 from ftplib import FTP
12 from os import name, system
13
14
15 # !The start of the characters list and key for mac encryption layers
16
17
18 CHARACTERS = (
19     "a",
20     "b",
21     "c",
22     "d",
23     "e",
24     "f",
25     "g",
26     "h",
27     "i",
28     "j",
29     "k",
30     "l",
31     "m",
32     "n",
33     "o",
34     "p",
35     "q",
36     "r",
37     "s",
38     "t",
39     "u",
40     "v",
41     "w",
42     "x",
43     "y",
44     "z",
45     "A",
46     "B",
47     "C",
48     "D",
49     "E",
50     "F",
51     "G",
52     "H",
53     "I",
54     "J",
55     "K",
56     "L",
57     "M",
58     "N",
59     "O",
60     "P",
61     "Q",
62     "R",
63     "S",
64     "T",
65     "U",
66     "V",
```

```

67     "W" /
68     "X" /
69     "Y" /
70     "Z" /
71     "!" /
72     " " /
73     "#" /
74     "$" /
75     "%" /
76     "&" /
77     "'" /
78     "(" /
79     ")" /
80     "*" /
81     "+" /
82     "," /
83     "-" /
84     "." /
85     "/" /
86     ":" /
87     ";" /
88     "<" /
89     "=" /
90     ">" /
91     "?" /
92     "@" /
93     "[" /
94     "\" /
95     "]" /
96     "^" /
97     "_" /
98     "`" /
99     "{" /
100    "|" /
101    "}" /
102    "~" /
103    "0" /
104    "1" /
105    "2" /
106    "3" /
107    "4" /
108    "5" /
109    "6" /
110    "7" /
111    "8" /
112    "9" /
113 )
114 KEY_FOR_MAC_1 = (
115     "!" /
116     ")" /
117     "." /
118     "3" /
119     "(" /
120     "J" /
121     "b" /
122     "C" /
123     "A" /
124     ";" /
125     "n" /
126     " " /
127     "$" /
128     "0" /
129     "," /
130     "Z" /
131     "B" /
132     "c" /
133     "|" /

```

134 "g" /  
135 "7" /  
136 "^" /  
137 " " /  
138 "I" /  
139 "v" /  
140 "P" /  
141 "Y" /  
142 "M" /  
143 "i" /  
144 "O" /  
145 "k" /  
146 "L" /  
147 "%" /  
148 "Y" /  
149 "D" /  
150 "#" /  
151 "@" /  
152 "<" /  
153 "9" /  
154 "r" /  
155 "K" /  
156 "2" /  
157 "-" /  
158 "t" /  
159 "8" /  
160 "H" /  
161 "=" /  
162 "+" /  
163 "`" /  
164 "o" /  
165 "T" /  
166 "u" /  
167 "s" /  
168 "p" /  
169 "U" /  
170 "S" /  
171 "Q" /  
172 "~" /  
173 "!" /  
174 "G" /  
175 "\*" /  
176 "&" /  
177 ">" /  
178 "X" /  
179 "a" /  
180 "N" /  
181 "4" /  
182 "d" /  
183 "m" /  
184 "l" /  
185 "V" /  
186 "w" /  
187 "z" /  
188 ":" /  
189 "q" /  
190 "?" /  
191 "5" /  
192 "x" /  
193 "{" /  
194 "/" /  
195 "F" /  
196 "j" /  
197 "]" /  
198 "6" /  
199 "W" /  
200 "f" /

```
201     "R" /
202     "1" /
203     "}" /
204     "e" /
205     "[" /
206     "\\\" /
207     "h" /
208     "E" /
209 )
210 KEY_FOR_MAC_2 = (
211     "$" /
212     "i" /
213     "o" /
214     "c" /
215     "&" /
216     "L" /
217     "/" /
218     "~" /
219     "O" /
220     "@" /
221     "(" /
222     "[" /
223     "Q" /
224     "v" /
225     "a" /
226     "Y" /
227     "7" /
228     "3" /
229     "*" /
230     "X" /
231     "x" /
232     "8" /
233     "{" /
234     "K" /
235     "E" /
236     "4" /
237     "M" /
238     "q" /
239     "|" /
240     "J" /
241     "0" /
242     "<" /
243     "Z" /
244     "e" /
245     " " /
246     "V̄" /
247     "H" /
248     "m" /
249     "G" /
250     "A" /
251     "d" /
252     "y" /
253     "T" /
254     "." /
255     "S" /
256     "j" /
257     "I" /
258     "#" /
259     "]" /
260     "p" /
261     "\\\" /
262     "F" /
263     "b" /
264     "!" /
265     "W" /
266     "k" /
267     ";" /
```

```
268     "5" /
269     "N" /
270     "s" /
271     "2" /
272     "1" /
273     "1" /
274     ":" /
275     "r" /
276     "-" /
277     "`" /
278     "h" /
279     ">" /
280     "t" /
281     "}" /
282     "^" /
283     "+" /
284     "U" /
285     "B" /
286     "!" /
287     "6" /
288     ")" /
289     "," /
290     "P" /
291     "g" /
292     "=" /
293     "C" /
294     "D" /
295     "?" /
296     "R" /
297     "9" /
298     "z" /
299     "i" /
300     "n" /
301     "w" /
302     "f" /
303     "%" /
304     "u" /
305 )
306 KEY_FOR_MAC_3 = (
307     "h" /
308     "R" /
309     "-" /
310     "I" /
311     "4" /
312     "$" /
313     "Z" /
314     "c" /
315     "/" /
316     "Q" /
317     "O" /
318     "a" /
319     "E" /
320     "V" /
321     " " /
322     "q" /
323     "A" /
324     "!" /
325     "," /
326     "X" /
327     "N" /
328     "H" /
329     "`" /
330     "." /
331     "K" /
332     "e" /
333     "*" /
334     "v" /
```

```
335      "\\\" ,
336      "T" ,
337      "k" ,
338      "M" ,
339      "Y" ,
340      "^" ,
341      "6" ,
342      "l" ,
343      "j" ,
344      "G" ,
345      "i" ,
346      "m" ,
347      "x" ,
348      "7" ,
349      "3" ,
350      "L" ,
351      ";" ,
352      "C" ,
353      "1" ,
354      "P" ,
355      "2" ,
356      "S" ,
357      "l" ,
358      "r" ,
359      "%" ,
360      "g" ,
361      "@" ,
362      "0" ,
363      ")" ,
364      "}" ,
365      "B" ,
366      "W" ,
367      "|" ,
368      "t" ,
369      "=" ,
370      "?" ,
371      ">" ,
372      "{" ,
373      "b" ,
374      "d" ,
375      "i" ,
376      "u" ,
377      "f" ,
378      "+" ,
379      "P" ,
380      ":" ,
381      "z" ,
382      "i" ,
383      "F" ,
384      "~" ,
385      "[" ,
386      "(" ,
387      "&" ,
388      "o" ,
389      "n" ,
390      "D" ,
391      "#" ,
392      "5" ,
393      "8" ,
394      "J" ,
395      "w" ,
396      "s" ,
397      "U" ,
398      "9" ,
399      "<" ,
400      "y" ,
401      )
```

```
402 KEY_FOR_MAC_4 = (
403     " " /
404     " ( " /
405     "H" /
406     "7" /
407     "~" /
408     "t" /
409     "F" /
410     "D" /
411     "j" /
412     "5" /
413     "C" /
414     "#" /
415     "o" /
416     "^" /
417     "X" /
418     "z" /
419     "9" /
420     "-" /
421     "k" /
422     " " /
423     "N" /
424     "E" /
425     "+" /
426     "q" /
427     "P" /
428     "/" /
429     ";" /
430     "S" /
431     "W" /
432     "h" /
433     "s" /
434     "f" /
435     "." /
436     "A" /
437     "4" /
438     " " /
439     "\\ " /
440     "Q" /
441     "/" /
442     "m" /
443     "e" /
444     "$" /
445     "6" /
446     "p" /
447     "i" /
448     "y" /
449     "|" /
450     "U" /
451     "a" /
452     "3" /
453     "L" /
454     "g" /
455     "c" /
456     "]" /
457     "0" /
458     ")" /
459     "v" /
460     "x" /
461     "8" /
462     "T" /
463     "l" /
464     "u" /
465     "b" /
466     "[" /
467     "V" /
468     ":" /
```

```
469     "M" /
470     "r" /
471     "B" /
472     "G" /
473     "l" /
474     "K" /
475     "d" /
476     ">" /
477     "O" /
478     "`" /
479     "*" /
480     "I" /
481     "?" /
482     "2" /
483     "}" /
484     "n" /
485     "@" /
486     "<" /
487     "&" /
488     "Z" /
489     "{" /
490     "R" /
491     "%" /
492     "!" /
493     "Y" /
494     "w" /
495     "=" /
496     "J" /
497 )
498 KEY_FOR_MAC_5 = (
499     "W" /
500     "f" /
501     "v" /
502     "l" /
503     "i" /
504     "h" /
505     "@" /
506     "C" /
507     ";" /
508     "!" /
509     "a" /
510     "8" /
511     "5" /
512     "S" /
513     "G" /
514     "X" /
515     "-" /
516     "{" /
517     "^" /
518     "~" /
519     "|" /
520     "m" /
521     "=" /
522     "&" /
523     "H" /
524     "O" /
525     "M" /
526     "Q" /
527     "0" /
528     "d" /
529     "I" /
530     "g" /
531     "6" /
532     "J" /
533     "9" /
534     "E" /
535     "z" /
```



```
536 "K" /
537 "L" /
538 "?" /
539 "/" /
540 ">" /
541 "o" /
542 "]" /
543 "n" /
544 "Y" /
545 "F" /
546 "t" /
547 "i" /
548 "D" /
549 "w" /
550 "#" /
551 "(" /
552 "`" /
553 "y" /
554 "%" /
555 "r" /
556 "[" /
557 "p" /
558 "*" /
559 "b" /
560 "j" /
561 "<" /
562 "}" /
563 "!" /
564 "\\ " /
565 "," /
566 "B" /
567 "R" /
568 "q" /
569 "U" /
570 "x" /
571 ")" /
572 "$" /
573 " " /
574 "1" /
575 "+" /
576 "T" /
577 "N" /
578 "s" /
579 "7" /
580 "c" /
581 "k" /
582 ":" /
583 "4" /
584 "3" /
585 "e" /
586 "Z" /
587 "V" /
588 "P" /
589 "2" /
590 "." /
591 "u" /
592 "A" /
593 )
594 KEY_FOR_MAC_6 = (
595 "}" /
596 "X" /
597 "j" /
598 "%" /
599 "I" /
600 "T" /
601 "L" /
602 "s" /
```

603 " ^ " /  
604 " m " /  
605 " d " /  
606 " K " /  
607 " ) " /  
608 " - " /  
609 " B " /  
610 " D " /  
611 " o " /  
612 " E " /  
613 " @ " /  
614 " Z " /  
615 " ` " /  
616 " 2 " /  
617 " f " /  
618 " a " /  
619 " 6 " /  
620 " q " /  
621 " e " /  
622 " n " /  
623 " \* " /  
624 " R " /  
625 " [ " /  
626 " ; " /  
627 " v " /  
628 " ' " /  
629 " z " /  
630 " y " /  
631 " + " /  
632 " S " /  
633 " W " /  
634 " 7 " /  
635 " x " /  
636 " & " /  
637 " < " /  
638 " 9 " /  
639 " U " /  
640 " N " /  
641 " ] " /  
642 " # " /  
643 " = " /  
644 " b " /  
645 " Q " /  
646 " Y " /  
647 " ( " /  
648 " r " /  
649 " \ " /  
650 " M " /  
651 " A " /  
652 " v " /  
653 " H " /  
654 " { " /  
655 " w " /  
656 " p " /  
657 " 3 " /  
658 " , " /  
659 " k " /  
660 " i " /  
661 " t " /  
662 " : " /  
663 " u " /  
664 " J " /  
665 " 8 " /  
666 " F " /  
667 " > " /  
668 " i " /  
669 " G " /

```
670     "P" /
671     "0" /
672     "/" /
673     "~" /
674     "|" /
675     "h" /
676     "o" /
677     "g" /
678     " " /
679     "c" /
680     "l" /
681     "C" /
682     "4" /
683     "?" /
684     "1" /
685     "." /
686     "!" /
687     "$" /
688     "5" /
689 )
690 KEY_FOR_MAC_7 = (
691     "U" /
692     "E" /
693     "." /
694     "T" /
695     "n" /
696     "}" /
697     ")" /
698     "E" /
699     "I" /
700     "S" /
701     "{" /
702     "\\" /
703     ">" /
704     "B" /
705     "a" /
706     "j" /
707     "Z" /
708     "/" /
709     "<" /
710     "P" /
711     "e" /
712     "O" /
713     "|" /
714     "," /
715     "i" /
716     "0" /
717     "1" /
718     "6" /
719     "@" /
720     ";" /
721     "X" /
722     "M" /
723     "J" /
724     "#" /
725     "2" /
726     "7" /
727     "+" /
728     "s" /
729     "j" /
730     "h" /
731     "x" /
732     "&" /
733     "y" /
734     "-" /
735     "`" /
736     "G" /
```

```
737     "3" /
738     ":" /
739     "(" /
740     "L" /
741     "=" /
742     "f" /
743     "*" /
744     "H" /
745     "Q" /
746     "R" /
747     "8" /
748     "4" /
749     "z" /
750     "[" /
751     "^" /
752     "5" /
753     "g" /
754     "D" /
755     "Y" /
756     "!" /
757     "W" /
758     "$" /
759     "q" /
760     "9" /
761     "K" /
762     "k" /
763     "!" /
764     "d" /
765     "p" /
766     "A" /
767     "u" /
768     "w" /
769     "C" /
770     "-" /
771     "o" /
772     "b" /
773     "!" /
774     "r" /
775     "t" /
776     "N" /
777     "v" /
778     "?" /
779     "m" /
780     "l" /
781     "c" /
782     "%" /
783     "V" /
784     "~" /
785 )
786 KEY_FOR_MAC_8 = (
787     "u" /
788     "5" /
789     "-" /
790     "Y" /
791     "l" /
792     "p" /
793     "D" /
794     "L" /
795     "b" /
796     "k" /
797     "@" /
798     "9" /
799     "O" /
800     "0" /
801     "*" /
802     "}" /
803     "f" /
```

804 "w" /  
805 "C" /  
806 "d" /  
807 " " /  
808 "g" /  
809 "t" /  
810 "K" /  
811 "&" /  
812 ";" /  
813 "s" /  
814 "v" /  
815 "o" /  
816 "=" /  
817 "7" /  
818 "I" /  
819 "]" /  
820 "." /  
821 "R" /  
822 "c" /  
823 "?" /  
824 "a" /  
825 "^" /  
826 "V" /  
827 "x" /  
828 "%" /  
829 "n" /  
830 "\\ " /  
831 "z" /  
832 "B" /  
833 "i" /  
834 "Q" /  
835 "U" /  
836 "+" /  
837 "m" /  
838 "l" /  
839 "," /  
840 "F" /  
841 "M" /  
842 "\$" /  
843 "T" /  
844 "3" /  
845 "N" /  
846 ")" /  
847 "r" /  
848 "Y" /  
849 ">" /  
850 "8" /  
851 "P" /  
852 "q" /  
853 "2" /  
854 "<" /  
855 "|" /  
856 "`" /  
857 "4" /  
858 "~" /  
859 "/" /  
860 "A" /  
861 ":" /  
862 "J" /  
863 "{" /  
864 "(" /  
865 "H" /  
866 "-" /  
867 "h" /  
868 "i" /  
869 "j" /  
870 "G" /

```
871     "S" /
872     "X" /
873     "[" /
874     "6" /
875     "Z" /
876     "E" /
877     "W" /
878     "#" /
879     "e" /
880     "!" /
881 )
882 KEY_FOR_MAC_9 = (
883     "/" /
884     "T" /
885     "g" /
886     "B" /
887     "5" /
888     "n" /
889     "f" /
890     "s" /
891     "G" /
892     "W" /
893     "V" /
894     "`" /
895     "M" /
896     "[" /
897     "!" /
898     "e" /
899     "L" /
900     "-" /
901     "_" /
902     "Q" /
903     "9" /
904     "^" /
905     ")" /
906     "P" /
907     "0" /
908     "j" /
909     "2" /
910     "v" /
911     "?" /
912     "E" /
913     "Y" /
914     "Z" /
915     "1" /
916     "|" /
917     "." /
918     "#" /
919     "w" /
920     "D" /
921     "J" /
922     "l" /
923     "O" /
924     "@" /
925     "t" /
926     "{" /
927     "x" /
928     "+" /
929     "4" /
930     "S" /
931     ";" /
932     "u" /
933     "F" /
934     "~" /
935     "q" /
936     "}" /
937     "<" /
```

```
938 "N" /
939 "o" /
940 "(" /
941 "K" /
942 "/" /
943 "z" /
944 "a" /
945 "\\\" /
946 "p" /
947 "3" /
948 " " /
949 "i" /
950 "8" /
951 "7" /
952 "%" /
953 "6" /
954 "U" /
955 ">" /
956 "C" /
957 "X" /
958 "$" /
959 "R" /
960 "&" /
961 "d" /
962 "c" /
963 "I" /
964 "y" /
965 "H" /
966 "=" /
967 " " /
968 "h" /
969 "r" /
970 "m" /
971 "*" /
972 "b" /
973 ":" /
974 "A" /
975 "]" /
976 "k" /
977 )
978 KEY_FOR_MAC_10 = (
979 "0" /
980 "R" /
981 "h" /
982 "C" /
983 "~" /
984 "o" /
985 "i" /
986 "g" /
987 "," /
988 "k" /
989 "8" /
990 "H" /
991 "Q" /
992 "p" /
993 "9" /
994 "v" /
995 "$" /
996 "u" /
997 "5" /
998 "-" /
999 "a" /
1000 "{" /
1001 " " /
1002 "s" /
1003 "A" /
1004 ";" /
```

1005	"["
1006	"Y"
1007	"^"
1008	"r"
1009	"M"
1010	"b"
1011	"G"
1012	"_"
1013	"m"
1014	"e"
1015	"x"
1016	"I"
1017	"T"
1018	"n"
1019	">"
1020	"<"
1021	"P"
1022	"J"
1023	"U"
1024	"2"
1025	"3"
1026	"4"
1027	"#"
1028	"W"
1029	":"
1030	"w"
1031	"D"
1032	" "
1033	"("
1034	"z"
1035	"q"
1036	"K"
1037	"v"
1038	"B"
1039	"L"
1040	"j"
1041	"d"
1042	"&"
1043	"!"
1044	"1"
1045	"+"
1046	"\""
1047	"X"
1048	"/"
1049	"?"
1050	"@"
1051	"*"
1052	"t"
1053	"c"
1054	"}"
1055	"Z"
1056	"%"
1057	"."
1058	")"
1059	"E"
1060	"`"
1061	"f"
1062	"N"
1063	"y"
1064	"6"
1065	"7"
1066	"O"
1067	"="
1068	"]"
1069	"F"
1070	"l"
1071	"S"



```
1072     " " ,
1073 )
1074 KEY_FOR_MAC_11 = (
1075     "S" ,
1076     "}" ,
1077     "!" ,
1078     "N" ,
1079     "\\\" ,
1080     "C" ,
1081     "M" ,
1082     "[" ,
1083     "z" ,
1084     "?" ,
1085     "%" ,
1086     "7" ,
1087     "q" ,
1088     "Y" ,
1089     "." ,
1090     "$" ,
1091     "l" ,
1092     "D" ,
1093     "G" ,
1094     "f" ,
1095     "+" ,
1096     "/" ,
1097     "6" ,
1098     "#" ,
1099     "5" ,
1100     "w" ,
1101     "P" ,
1102     "O" ,
1103     "U" ,
1104     "*" ,
1105     "X" ,
1106     "d" ,
1107     "=" ,
1108     "3" ,
1109     ":" ,
1110     "i" ,
1111     "y" ,
1112     "h" ,
1113     "v" ,
1114     "(" ,
1115     "R" ,
1116     "W" ,
1117     "x" ,
1118     " " ,
1119     "1" ,
1120     "c" ,
1121     "A" ,
1122     "<" ,
1123     "J" ,
1124     " " ,
1125     "L" ,
1126     "|" ,
1127     "T" ,
1128     "^" ,
1129     ">" ,
1130     "H" ,
1131     "2" ,
1132     "`" ,
1133     "I" ,
1134     ";" ,
1135     "0" ,
1136     "k" ,
1137     "-" ,
1138     "u" ,
```

```

1139     "o",
1140     "&",
1141     "z",
1142     "j",
1143     "p",
1144     ")",
1145     ",",
1146     "r",
1147     "g",
1148     "i",
1149     "Q",
1150     "4",
1151     "]",
1152     "m",
1153     "t",
1154     "B",
1155     "a",
1156     "F",
1157     "V",
1158     "b",
1159     "8",
1160     "E",
1161     "n",
1162     "g",
1163     "@",
1164     "e",
1165     "s",
1166     "~",
1167     "{",
1168     "K",
1169 )
1170
1171
1172 # !The end of the characters list and key for mac encryption layers
1173
1174
1175 # !---The start for the all encryption and decryption functions---
1176
1177
1178 def reverser(text):
1179     """This function reverses the text"""
1180     return text[::-1]
1181
1182
1183 # ---The end of reverser function.---
1184
1185
1186 def mac1_encode(text):
1187     """Encryption layer for Mono-Alphabetic cipher - Number (1)."""
1188     result = []
1189     for char in text:
1190         if char in CHARACTERS:
1191             index = CHARACTERS.index(char)
1192             result.append(KEY_FOR_MAC_1[index])
1193         else:
1194             result.append(char)
1195     return "".join(result)
1196
1197
1198 def mac1_decode(text):
1199     """Decryption layer for Mono-Alphabetic cipher - Number (1)."""
1200     result = []
1201     for char in text:
1202         if char in KEY_FOR_MAC_1:
1203             index = KEY_FOR_MAC_1.index(char)
1204             result.append(CHARACTERS[index])
1205         else:

```

```

1206         result.append(char)
1207     return "".join(result)
1208
1209
1210 # ---The end for the encryption and decryption functions - Number (1)---
1211
1212
1213 def mac2_encode(text):
1214     """Encryption layer for Mono-Alphabetic cipher - Number (2)."""
1215     result = []
1216     for char in text:
1217         if char in CHARACTERS:
1218             index = CHARACTERS.index(char)
1219             result.append(KEY_FOR_MAC_2[index])
1220         else:
1221             result.append(char)
1222     return "".join(result)
1223
1224
1225 def mac2_decode(text):
1226     """Decryption layer for Mono-Alphabetic cipher - Number (2)."""
1227     result = []
1228     for char in text:
1229         if char in KEY_FOR_MAC_2:
1230             index = KEY_FOR_MAC_2.index(char)
1231             result.append(CHARACTERS[index])
1232         else:
1233             result.append(char)
1234     return "".join(result)
1235
1236
1237 # ---The end for the encryption and decryption functions - Number (2)---
1238
1239
1240 def mac3_encode(text):
1241     """Encryption layer for Mono-Alphabetic cipher - Number (3)."""
1242     result = []
1243     for char in text:
1244         if char in CHARACTERS:
1245             index = CHARACTERS.index(char)
1246             result.append(KEY_FOR_MAC_3[index])
1247         else:
1248             result.append(char)
1249     return "".join(result)
1250
1251
1252 def mac3_decode(text):
1253     """Decryption layer for Mono-Alphabetic cipher - Number (3)."""
1254     result = []
1255     for char in text:
1256         if char in KEY_FOR_MAC_3:
1257             index = KEY_FOR_MAC_3.index(char)
1258             result.append(CHARACTERS[index])
1259         else:
1260             result.append(char)
1261     return "".join(result)
1262
1263
1264 # ---The end for the encryption and decryption functions - Number (3)---
1265
1266
1267 def mac4_encode(text):
1268     """Encryption layer for Mono-Alphabetic cipher - Number (4)."""
1269     result = []
1270     for char in text:
1271         if char in CHARACTERS:
1272             index = CHARACTERS.index(char)

```

```

1273         result.append(KEY_FOR_MAC_4[index])
1274     else:
1275         result.append(char)
1276     return "".join(result)
1277
1278
1279 def mac4_decode(text):
1280     """Decryption layer for Mono-Alphabetic cipher - Number (4)."""
1281     result = []
1282     for char in text:
1283         if char in KEY_FOR_MAC_4:
1284             index = KEY_FOR_MAC_4.index(char)
1285             result.append(characters[index])
1286         else:
1287             result.append(char)
1288     return "".join(result)
1289
1290
1291 # ---The end for the encryption and decryption functions - Number (4)---
1292
1293
1294 def mac5_encode(text):
1295     """Encryption layer for Mono-Alphabetic cipher - Number (5)."""
1296     result = []
1297     for char in text:
1298         if char in characters:
1299             index = characters.index(char)
1300             result.append(key_for_mac_5[index])
1301         else:
1302             result.append(char)
1303     return "".join(result)
1304
1305
1306 def mac5_decode(text):
1307     """Decryption layer for Mono-Alphabetic cipher - Number (5)."""
1308     result = []
1309     for char in text:
1310         if char in key_for_mac_5:
1311             index = key_for_mac_5.index(char)
1312             result.append(characters[index])
1313         else:
1314             result.append(char)
1315     return "".join(result)
1316
1317
1318 # ---The end for the encryption and decryption functions - Number (5)---
1319
1320
1321 def mac6_encode(text):
1322     """Encryption layer for Mono-Alphabetic cipher - Number (6)."""
1323     result = []
1324     for char in text:
1325         if char in characters:
1326             index = characters.index(char)
1327             result.append(key_for_mac_6[index])
1328         else:
1329             result.append(char)
1330     return "".join(result)
1331
1332
1333 def mac6_decode(text):
1334     """Decryption layer for Mono-Alphabetic cipher - Number (6)."""
1335     result = []
1336     for char in text:
1337         if char in key_for_mac_6:
1338             index = key_for_mac_6.index(char)
1339             result.append(characters[index])

```

```

1340         else:
1341             result.append(char)
1342     return "".join(result)
1343
1344
1345 # ---The end for the encryption and decryption functions - Number (6)---
1346
1347
1348 def mac7_encode(text):
1349     """Encryption layer for Mono-Alphabetic cipher - Number (7)."""
1350     result = []
1351     for char in text:
1352         if char in CHARACTERS:
1353             index = CHARACTERS.index(char)
1354             result.append(KEY_FOR_MAC_7[index])
1355         else:
1356             result.append(char)
1357     return "".join(result)
1358
1359
1360 def mac7_decode(text):
1361     """Decryption layer for Mono-Alphabetic cipher - Number (7)."""
1362     result = []
1363     for char in text:
1364         if char in KEY_FOR_MAC_7:
1365             index = KEY_FOR_MAC_7.index(char)
1366             result.append(CHARACTERS[index])
1367         else:
1368             result.append(char)
1369     return "".join(result)
1370
1371
1372 # ---The end for the encryption and decryption functions - Number (7)---
1373
1374
1375 def mac8_encode(text):
1376     """Encryption layer for Mono-Alphabetic cipher - Number (8)."""
1377     result = []
1378     for char in text:
1379         if char in CHARACTERS:
1380             index = CHARACTERS.index(char)
1381             result.append(KEY_FOR_MAC_8[index])
1382         else:
1383             result.append(char)
1384     return "".join(result)
1385
1386
1387 def mac8_decode(text):
1388     """Decryption layer for Mono-Alphabetic cipher - Number (8)."""
1389     result = []
1390     for char in text:
1391         if char in KEY_FOR_MAC_8:
1392             index = KEY_FOR_MAC_8.index(char)
1393             result.append(CHARACTERS[index])
1394         else:
1395             result.append(char)
1396     return "".join(result)
1397
1398
1399 # ---The end for the encryption and decryption functions - Number (8)---
1400
1401
1402 def mac9_encode(text):
1403     """Encryption layer for Mono-Alphabetic cipher - Number (9)."""
1404     result = []
1405     for char in text:
1406         if char in CHARACTERS:

```

```

1407         index = CHARACTERS.index(char)
1408         result.append(KEY_FOR_MAC_9[index])
1409     else:
1410         result.append(char)
1411     return "".join(result)
1412
1413
1414 def mac9_decode(text):
1415     """Decryption layer for Mono-Alphabetic cipher - Number (9)."""
1416     result = []
1417     for char in text:
1418         if char in KEY_FOR_MAC_9:
1419             index = KEY_FOR_MAC_9.index(char)
1420             result.append(CHARACTERS[index])
1421         else:
1422             result.append(char)
1423     return "".join(result)
1424
1425
1426 # ---The end for the encryption and decryption functions - Number (9)---
1427
1428
1429 def mac10_encode(text):
1430     """Encryption layer for Mono-Alphabetic cipher - Number (10)."""
1431     result = []
1432     for char in text:
1433         if char in CHARACTERS:
1434             index = CHARACTERS.index(char)
1435             result.append(KEY_FOR_MAC_10[index])
1436         else:
1437             result.append(char)
1438     return "".join(result)
1439
1440
1441 def mac10_decode(text):
1442     """Decryption layer for Mono-Alphabetic cipher - Number (10)."""
1443     result = []
1444     for char in text:
1445         if char in KEY_FOR_MAC_10:
1446             index = KEY_FOR_MAC_10.index(char)
1447             result.append(CHARACTERS[index])
1448         else:
1449             result.append(char)
1450     return "".join(result)
1451
1452
1453 # ---The end for the encryption and decryption functions - Number (10)---
1454
1455
1456 def mac11_encode(text):
1457     """Encryption layer for Mono-Alphabetic cipher - Number (11)."""
1458     result = []
1459     for char in text:
1460         if char in CHARACTERS:
1461             index = CHARACTERS.index(char)
1462             result.append(KEY_FOR_MAC_11[index])
1463         else:
1464             result.append(char)
1465     return "".join(result)
1466
1467
1468 def mac11_decode(text):
1469     """Decryption layer for Mono-Alphabetic cipher - Number (11)."""
1470     result = []
1471     for char in text:
1472         if char in KEY_FOR_MAC_11:
1473             index = KEY_FOR_MAC_11.index(char)

```

```

1474         result.append(CHARACTERS[index])
1475     else:
1476         result.append(char)
1477     return "".join(result)
1478
1479
1480 # ---The end for the encryption and decryption functions - Number (11)---
1481
1482
1483 # !---The end for the all encryption and decryption functions---
1484
1485
1486 def all_mac_encryption(text):
1487     """12 Layers combined MAC encryption"""
1488     layer1 = mac5_encode(text)
1489     layer2 = reverser(layer1)
1490     layer3 = mac4_encode(layer2)
1491     layer4 = mac2_encode(layer3)
1492     layer5 = mac1_encode(layer4)
1493     layer6 = mac3_encode(layer5)
1494     layer7 = mac8_encode(layer6)
1495     layer8 = mac7_encode(layer7)
1496     layer9 = mac9_encode(layer8)
1497     layer10 = mac6_encode(layer9)
1498     layer11 = mac11_encode(layer10)
1499     return str(mac10_encode(layer11))
1500
1501
1502 def all_mac_decryption(text):
1503     """12 Layers combined MAC decryption"""
1504     layer1 = mac10_decode(text)
1505     layer2 = mac11_decode(layer1)
1506     layer3 = mac6_decode(layer2)
1507     layer4 = mac9_decode(layer3)
1508     layer5 = mac7_decode(layer4)
1509     layer6 = mac8_decode(layer5)
1510     layer7 = mac3_decode(layer6)
1511     layer8 = mac1_decode(layer7)
1512     layer9 = mac2_decode(layer8)
1513     layer10 = mac4_decode(layer9)
1514     layer11 = reverser(layer10)
1515     return str(mac5_decode(layer11))
1516
1517
1518 def encrypt(text):
1519     """Doubled 12 Layer MAC encryption"""
1520     return str(all_mac_encryption(all_mac_encryption(text)))
1521
1522
1523 def decrypt(text):
1524     """Doubled 12 Layer MAC decryption"""
1525     return str(all_mac_decryption(all_mac_decryption(text)))
1526
1527
1528 # !---The sending and reading messages functions---
1529
1530
1531 def send_message(
1532     username, message, ftp_host, ftp_user, ftp_pass, chat_input, chat_name
1533 ):
1534     """Send a message to the chat file on the FTP server."""
1535     try:
1536         if message == "REFRESH":
1537             read_messages(ftp_host, ftp_user, ftp_pass, chat_input, chat_name)
1538         else:
1539             msg = encrypt(f"{datetime.now()}:{username}: {message}\n")
1540             ftp = FTP(ftp_host)

```

```

1541         ftp.login(ftp_user, ftp_pass)
1542     try:
1543         with open(chat_input, "wb") as file:
1544             ftp.retrbinary(f"RETR {chat_name}", file.write)
1545     except:
1546         pass
1547
1548     with open(chat_input, "a", encoding="utf-8") as file:
1549         file.write(f"{msg}\n")
1550
1551     with open(chat_input, "rb") as file:
1552         ftp.storbinary(f"STOR {chat_name}", file)
1553
1554     ftp.quit()
1555 except Exception as error:
1556     print(f"Error sending message: {error}")
1557     input()
1558
1559
1560 def read_messages(ftp_host, ftp_user, ftp_pass, chat_input, chat_name):
1561     """Read messages from the chat file on the FTP server."""
1562     try:
1563         with FTP(ftp_host) as ftp:
1564             ftp.login(ftp_user, ftp_pass)
1565             with open(chat_input, "wb") as file:
1566                 ftp.retrbinary(f"RETR {chat_name}", file.write)
1567             with open(chat_input, "r", encoding="utf-8") as file:
1568                 lines = file.readlines()
1569                 if not lines:
1570                     print("No messages yet.")
1571                     return
1572                 print("-----Messages-----")
1573                 for line in lines:
1574                     decrypted_line = decrypt(line.strip())
1575                     print(decrypted_line)
1576     except Exception as error:
1577         print(f"Error reading messages. ({error})")
1578         input()
1579
1580
1581 def multiline_input(prompt):
1582     """Get multiline input from the user until 'END' is entered."""
1583     print(prompt)
1584     lines = []
1585     while True:
1586         line = input()
1587         if line.upper() == "END":
1588             break
1589         if line.upper() == "REF":
1590             line = "REFRESH"
1591             break
1592         if line.upper() == "CLS":
1593             system("cls" if name == "nt" else "clear")
1594             break
1595         lines.append(line)
1596     return "".join(lines)
1597
1598
1599 print(
1600     r"""
1601
1602     /\_/\
1603    / _  _ \
1604   / _  _ \
1605  / _  _ \
1606  / _  _ \
1607  / _  _ \
1608  / _  _ \
1609  / _  _ \
1610  / _  _ \
1611  / _  _ \
1612  / _  _ \
1613  / _  _ \
1614  / _  _ \
1615  / _  _ \
1616  / _  _ \
1617  / _  _ \
1618  / _  _ \
1619  / _  _ \
1620  / _  _ \
1621  / _  _ \
1622  / _  _ \
1623  / _  _ \
1624  / _  _ \
1625  / _  _ \
1626  / _  _ \
1627  / _  _ \
1628  / _  _ \
1629  / _  _ \
1630  / _  _ \
1631  / _  _ \
1632  / _  _ \
1633  / _  _ \
1634  / _  _ \
1635  / _  _ \
1636  / _  _ \
1637  / _  _ \
1638  / _  _ \
1639  / _  _ \
1640  / _  _ \
1641  / _  _ \
1642  / _  _ \
1643  / _  _ \
1644  / _  _ \
1645  / _  _ \
1646  / _  _ \
1647  / _  _ \
1648  / _  _ \
1649  / _  _ \
1650  / _  _ \
1651  / _  _ \
1652  / _  _ \
1653  / _  _ \
1654  / _  _ \
1655  / _  _ \
1656  / _  _ \
1657  / _  _ \
1658  / _  _ \
1659  / _  _ \
1660  / _  _ \
1661  / _  _ \
1662  / _  _ \
1663  / _  _ \
1664  / _  _ \
1665  / _  _ \
1666  / _  _ \
1667  / _  _ \
1668  / _  _ \
1669  / _  _ \
1670  / _  _ \
1671  / _  _ \
1672  / _  _ \
1673  / _  _ \
1674  / _  _ \
1675  / _  _ \
1676  / _  _ \
1677  / _  _ \
1678  / _  _ \
1679  / _  _ \
1680  / _  _ \
1681  / _  _ \
1682  / _  _ \
1683  / _  _ \
1684  / _  _ \
1685  / _  _ \
1686  / _  _ \
1687  / _  _ \
1688  / _  _ \
1689  / _  _ \
1690  / _  _ \
1691  / _  _ \
1692  / _  _ \
1693  / _  _ \
1694  / _  _ \
1695  / _  _ \
1696  / _  _ \
1697  / _  _ \
1698  / _  _ \
1699  / _  _ \
1700  / _  _ \
1701  / _  _ \
1702  / _  _ \
1703  / _  _ \
1704  / _  _ \
1705  / _  _ \
1706  / _  _ \
1707  / _  _ \
1708  / _  _ \
1709  / _  _ \
1710  / _  _ \
1711  / _  _ \
1712  / _  _ \
1713  / _  _ \
1714  / _  _ \
1715  / _  _ \
1716  / _  _ \
1717  / _  _ \
1718  / _  _ \
1719  / _  _ \
1720  / _  _ \
1721  / _  _ \
1722  / _  _ \
1723  / _  _ \
1724  / _  _ \
1725  / _  _ \
1726  / _  _ \
1727  / _  _ \
1728  / _  _ \
1729  / _  _ \
1730  / _  _ \
1731  / _  _ \
1732  / _  _ \
1733  / _  _ \
1734  / _  _ \
1735  / _  _ \
1736  / _  _ \
1737  / _  _ \
1738  / _  _ \
1739  / _  _ \
1740  / _  _ \
1741  / _  _ \
1742  / _  _ \
1743  / _  _ \
1744  / _  _ \
1745  / _  _ \
1746  / _  _ \
1747  / _  _ \
1748  / _  _ \
1749  / _  _ \
1750  / _  _ \
1751  / _  _ \
1752  / _  _ \
1753  / _  _ \
1754  / _  _ \
1755  / _  _ \
1756  / _  _ \
1757  / _  _ \
1758  / _  _ \
1759  / _  _ \
1760  / _  _ \
1761  / _  _ \
1762  / _  _ \
1763  / _  _ \
1764  / _  _ \
1765  / _  _ \
1766  / _  _ \
1767  / _  _ \
1768  / _  _ \
1769  / _  _ \
1770  / _  _ \
1771  / _  _ \
1772  / _  _ \
1773  / _  _ \
1774  / _  _ \
1775  / _  _ \
1776  / _  _ \
1777  / _  _ \
1778  / _  _ \
1779  / _  _ \
1780  / _  _ \
1781  / _  _ \
1782  / _  _ \
1783  / _  _ \
1784  / _  _ \
1785  / _  _ \
1786  / _  _ \
1787  / _  _ \
1788  / _  _ \
1789  / _  _ \
1790  / _  _ \
1791  / _  _ \
1792  / _  _ \
1793  / _  _ \
1794  / _  _ \
1795  / _  _ \
1796  / _  _ \
1797  / _  _ \
1798  / _  _ \
1799  / _  _ \
1800  / _  _ \
1801  / _  _ \
1802  / _  _ \
1803  / _  _ \
1804  / _  _ \
1805  / _  _ \
1806  / _  _ \
1807  / _  _ \
1808  / _  _ \
1809  / _  _ \
1810  / _  _ \
1811  / _  _ \
1812  / _  _ \
1813  / _  _ \
1814  / _  _ \
1815  / _  _ \
1816  / _  _ \
1817  / _  _ \
1818  / _  _ \
1819  / _  _ \
1820  / _  _ \
1821  / _  _ \
1822  / _  _ \
1823  / _  _ \
1824  / _  _ \
1825  / _  _ \
1826  / _  _ \
1827  / _  _ \
1828  / _  _ \
1829  / _  _ \
1830  / _  _ \
1831  / _  _ \
1832  / _  _ \
1833  / _  _ \
1834  / _  _ \
1835  / _  _ \
1836  / _  _ \
1837  / _  _ \
1838  / _  _ \
1839  / _  _ \
1840  / _  _ \
1841  / _  _ \
1842  / _  _ \
1843  / _  _ \
1844  / _  _ \
1845  / _  _ \
1846  / _  _ \
1847  / _  _ \
1848  / _  _ \
1849  / _  _ \
1850  / _  _ \
1851  / _  _ \
1852  / _  _ \
1853  / _  _ \
1854  / _  _ \
1855  / _  _ \
1856  / _  _ \
1857  / _  _ \
1858  / _  _ \
1859  / _  _ \
1860  / _  _ \
1861  / _  _ \
1862  / _  _ \
1863  / _  _ \
1864  / _  _ \
1865  / _  _ \
1866  / _  _ \
1867  / _  _ \
1868  / _  _ \
1869  / _  _ \
1870  / _  _ \
1871  / _  _ \
1872  / _  _ \
1873  / _  _ \
1874  / _  _ \
1875  / _  _ \
1876  / _  _ \
1877  / _  _ \
1878  / _  _ \
1879  / _  _ \
1880  / _  _ \
1881  / _  _ \
1882  / _  _ \
1883  / _  _ \
1884  / _  _ \
1885  / _  _ \
1886  / _  _ \
1887  / _  _ \
1888  / _  _ \
1889  / _  _ \
1890  / _  _ \
1891  / _  _ \
1892  / _  _ \
1893  / _  _ \
1894  / _  _ \
1895  / _  _ \
1896  / _  _ \
1897  / _  _ \
1898  / _  _ \
1899  / _  _ \
1900  / _  _ \
1901  / _  _ \
1902  / _  _ \
1903  / _  _ \
1904  / _  _ \
1905  / _  _ \
1906  / _  _ \
1907  / _  _ \
1908  / _  _ \
1909  / _  _ \
19
```



```

1608 )
1609
1610 all_filled = False
1611
1612 while True:
1613     ftp_host = input("Please, type the FTP host:\n")
1614     if not ftp_host:
1615         print("No Host entered, Exiting the app")
1616         input()
1617         break
1618
1619     ftp_user = input("Please, type the FTP username:\n")
1620     if not ftp_user:
1621         print("No FTP User entered, Exiting the app")
1622         input()
1623         break
1624
1625     ftp_pass = input("Please, type the FTP password:\n")
1626     if not ftp_pass:
1627         print("No Password entered, Exiting the app")
1628         input()
1629         break
1630
1631     username = input("Please, type your username:\n")
1632     if not username:
1633         username = "Anonymous"
1634         print("Username is empty, using 'Anonymous' as username.")
1635         continue
1636
1637     chat_input = input("Please, type the chat file name:\n") + ".txt"
1638     if not chat_input:
1639         print("No chat file name input, exiting the program")
1640         input()
1641         break
1642
1643     chat_name = f"/usb1_1/{chat_input}"
1644     all_filled = True
1645     break
1646
1647
1648 if all_filled:
1649     print(
1650         """
1651 Notes before start messaging:
1652 1. Type 'END' to finish your message.
1653 2. Type 'REF' to refresh messages.
1654 3. Type 'CLS' to clear the terminal.
1655 4. Don't turn off your FTP Server while using this chat.
1656 5. Messages are encrypted for security.
1657 6. To secure your chat, use a unique chat file name, and share the file name with the
1658 participants.
1659 """
1660     )
1661
1662     while True:
1663         read_messages(ftp_host, ftp_user, ftp_pass, chat_input, chat_name)
1664         message = multiline_input("Type your message:\n")
1665         send_message(
1666             username, message, ftp_host, ftp_user, ftp_pass, chat_input, chat_name
1667         )
1668

```