**3.1** A set of messages consists of diary entries. The daily diary entries are written in the English language and saved from a word processor in ASCII text files (\*.txt would be one possible format). The files have even parity in the most significant place of every byte for a total of 8 bits per symbol (one parity bit, seven ASCII code bits per byte). They are encrypted and transmitted over an insecure channel every day. There has been one such transmission every day for several years. The necessary keys were sent via a different and secure channel. To minimize the number of bits to be sent it is proposed that Huffman coding be applied to each diary file (in the set of files), with the key to the Huffman code also sent separately via the secure channel. The Huffman key from the first-ever diary is re-used with each subsequent diary entry, it being assumed that the style and vocabulary of diary entries do not change much from day to day. (Huffman code was discussed in class on 3/24/2023).

17/17

The Huffman coding technique is a well-known lossless data compression method. It is going to be combined with the AES method of symmetric key encryption. The key to the Huffman code and the key to the AES encryption will be pre-exchanged via a secure channel; thus, key exchange issues do not complicate or impinge on the solution to this problem.

**Which is the most advantageous order in which to combine these two coding methods, Huffman first or AES first? Please write a short paragraph giving your recommendation and your justification for that recommendation. Which factors should be considered as most important in this case and why?** The possible choices are detailed below.

A. HUFFMAN FIRST, then AES.

1) *Packetize*: Each daily diary file shall pass through an algorithm to delete all spaces, all punctuation, and all capitalization. After that process, the non-printing ASCII character SOH (start of heading) followed by the file name, followed by the non-printing ASCII character STX (start of text) shall be appended onto the beginning of the file.

2.) *Shuffle*: The above file shall be “shuffled” by computing the length of the file and dividing that number by two, and if necessary, truncating that to an integer value, *X*. Now remove the first *X* characters (bytes) from the file and re-append that at the end of the file. The file length has not changed.

3) *Pad*: The above file shall be padded by concatenating to the end of the file an ASCII EOT character followed by a sequence of random words (with no spaces, no punctuation, no capital letters) to consist of at least 33 characters (bytes) but not more than 71 characters (bytes).

4) *Huffman*: The result of the above shall be encrypted via the Huffman technique. Note that the Huffman encryption has variable-length symbols, so the resulting file will not necessarily be a multiple of 8 bits long. It will not necessarily be an integer number of bytes long. (Upon encoding the very first diary entry, the Huffman algorithm produced a key. This same Huffman key is used to encode and decode every subsequent diary file.)

5) *Truncate*: Remove just enough bits from the end of the file (between 0 and 7 bits) to make the file an integer number of bytes long and then remove an additional 0 to 15 *bytes* so that the file length in bytes is an integer multiple of 16 (for compatibility with the AES encryption block size).

6) *AES*: Taking the result from above one byte at a time as if each byte was still an ASCII symbol, encrypt using the AES method. (A key was randomly generated for the encryption of the first diary entry. This key is re-used for every subsequent diary entry.)

7) *Transmit*: Consider the result of Step 6 as the final cyphertext and send it to the recipient via the insecure channel.

8) *Decrypt*: The recipient will recover the original plaintext file from the received cyphertext by reversing the steps above, starting with the reversal of Step 6 and working back to the reversal of Step 1. At Step 5 there will be no way to replace truncated bits. Just ignore the fact that truncation happened and begin reversing Step 4. At Step 4 it is possible that the Huffman decryption will run out of bits before the last symbol is deciphered because Step 5 was not reversed. Just truncate the decryption at that point. The only loss will be one of the random text symbols previously concatenated to the message at step 3. When step 3 is reversed all those symbols will be discarded anyway. At Step 1 assume a computer algorithm is available to insert spaces, insert punctuation, and restore appropriate capitalization, extract the filename, and store the original file on the recipient’s computer.

B. AES FIRST, then HUFFMAN.

1) *Packetize*: Each daily diary entry shall pass through an algorithm to delete all spaces, all punctuation, and all capitalization. After that process, the non-printing ASCII character SOH (start of heading) followed by the file name, followed by the non-printing ASCII character STX (start of text) shall be appended onto the beginning of the file.

2.) *Shuffle*: The above file shall be “shuffled” by computing the length of the file. If the length is an odd number, insert a non-printing ASCII NULL character in a random place in the file but after the STX character. The overall length is now an even number. Dividing the file length by two and obtain integer *X*. Now remove the first *X* characters (bytes) from the file and re-append that at the end of the file. At the very end of the file add a non-printing ETX character. The file length is now one or two bytes longer than the packetized file was. Recompute the length of the file. If it is not a multiple of 16 bytes long, concatenate random characters to the end of the file to make it a multiple of 16 bytes long (in preparation for the block size of the AES encryption).

3) *AES*: The result of the above shall be encrypted via the AES technique. (Upon encoding the very first diary entry, an AES key was randomly generated. This same AES key is used to encode and decode every subsequent diary entry.)

4) *Huffman*: Taking the result from above one byte at a time, encrypt using the Huffman method. (A Huffman key was generated for the encryption of the first diary entry. This same Huffman key is re-used to encode and decode every subsequent diary entry.)

5) *Pad*: The above file shall be padded by concatenating to the end of the file a sequence of random bits, at least 32 and no more than 255 bits.

6) *Truncate*: Remove just enough bits from the end of the file (between 0 and 7 bits) to make the file an integer number of bytes long.

7) Consider the result of Step 6 as the final cyphertext and send it to the recipient via the insecure channel.

8) The recipient will recover the original plaintext file from the received cyphertext by reversing the steps above, starting with the reversal of Step 4 and working back to the reversal of Step 1. As Step 4 is reversed the decryption process might run out of bits prematurely due to the truncation process of Step 6 which could not be reversed. Just truncate the Huffman decryption at this point. It will not matter because these bits correspond to the pad that was added at Step 5 and will later be discarded. After step 3 is reversed the ETX character near the end of the file can be located. At this point Step 5, the padding process, can be reversed by discarding everything after the ETX character. Finally reverse steps 2 and 1 and the plaintext file is then stored on the recipient’s computer.

**3.2 In Problem 3.1 above, in either of the cases A (Huffman first) or B (AES first), what advantage does the *shuffle* step give?**

**3.3 In Problem 3.1 above, in either of the cases A (Huffman first) or B (AES first), what advantage does the *pad* step give?**