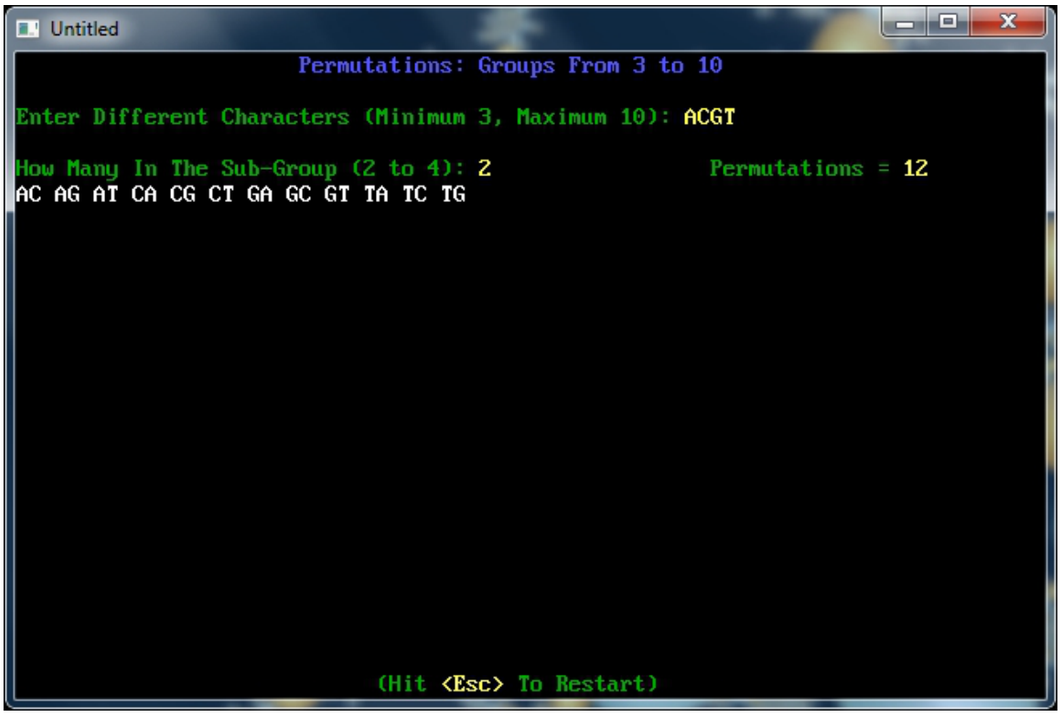
Encryption Using Only The DNA Alphabet: ACGT

File Reader to DNA Encrypted-Sequence Writer & DNA Encrypted-Sequence Reader to File Writer



If you do a normal permutation of ACGT with a sub-group of two (2), you get the above results, twelve (12) permutations. Notice that by definition, there are no duplications of any two letters in any of the sub-groups. However, by including "pseudo" permutations with duplications in the sub-groups, you get *sixteen* sub-groups as follows:

AA AC AG AT CA **CC** CG CT GA GC **GG** GT TA TC TG **TT**

This brings up interesting possibilities, since in binary, four bits represent sixteen decimal numbers 0 through 15 (hexadecimal 0 through F).

There are *four (4) Binary Bits per Nibble*, which is *one-half of a Byte*, and the binary key for these binary bits *taken two (2) at a time* is:

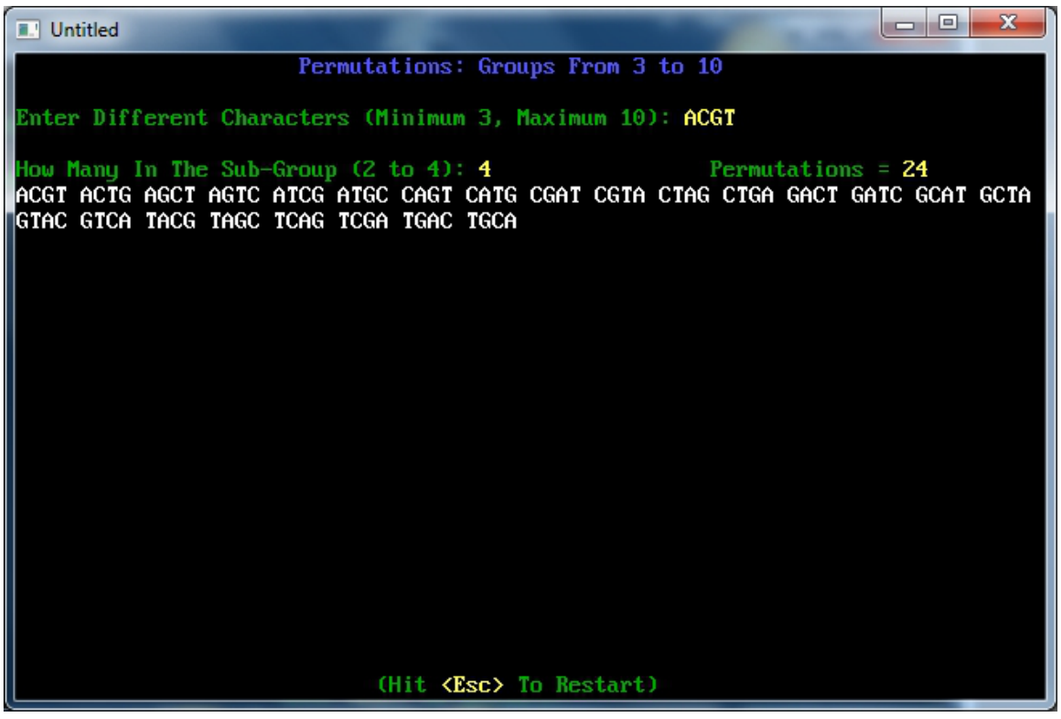
A = 00, C = 01, G = 10 and T = 11

From the key above, if the decimal value of a *Nibble* is 15 (hexadecimal F), this is represented in binary as 1111. From the chart below, this would result in the DNA written sequence: TT.

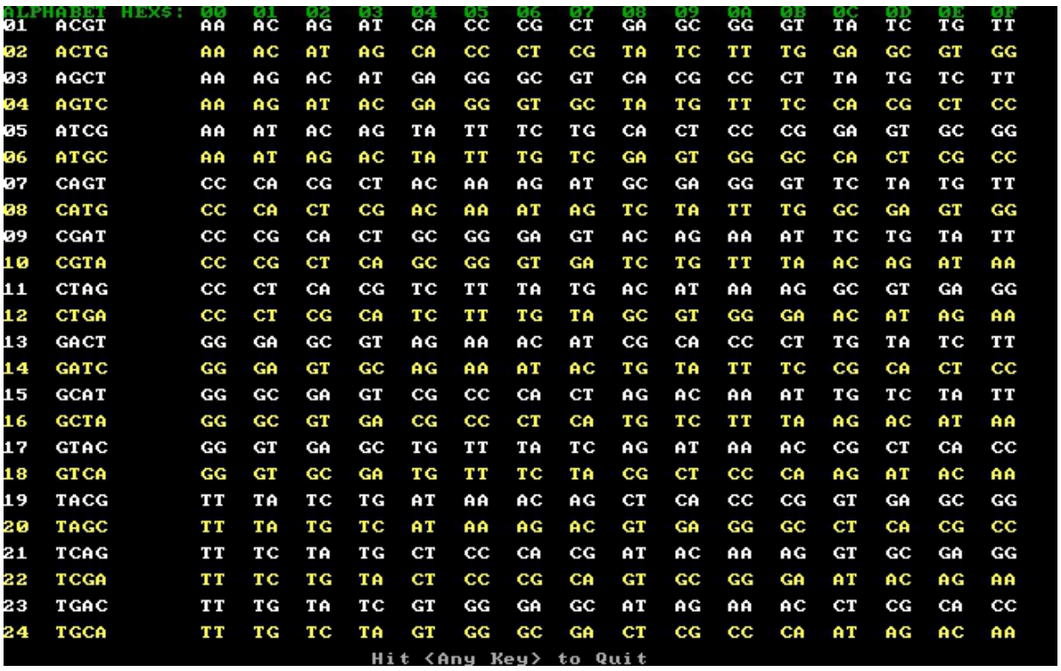
You can therefore see how an original file is made *four times larger*. However, it can be zipped back down to about the size of the original file for storage or transmission since it only contains four letters: A, C, G, and T.

By using a string array, DNA$(0) through DNA$(15) to compare with each half-byte nibble of an input file, a DNA alphabet sequence can be written to an output file as shown below:

Nibble Value		DNA$(0) through DNA$(15)
Decimal	Hex	Sequence Written
0	0	AA
1	1	AC
2	2	AG
3	3	AT
4	4	CA
5	5	CC
6	6	CG
7	7	CT
8	8	GA
9	9	GC
10	A	GG
11	B	GT
12	C	TA
13	D	TC
14	E	TG
15	F	TT



If you do a normal permutation of ACGT with a sub-group of four (4), you get the above results, twenty-four (24) permutations - alphabets. Therefore, the *specific* permutation from the twenty-four permutations above that is initially used will determine the *order* ot the sub-group set with two (2) permutations.

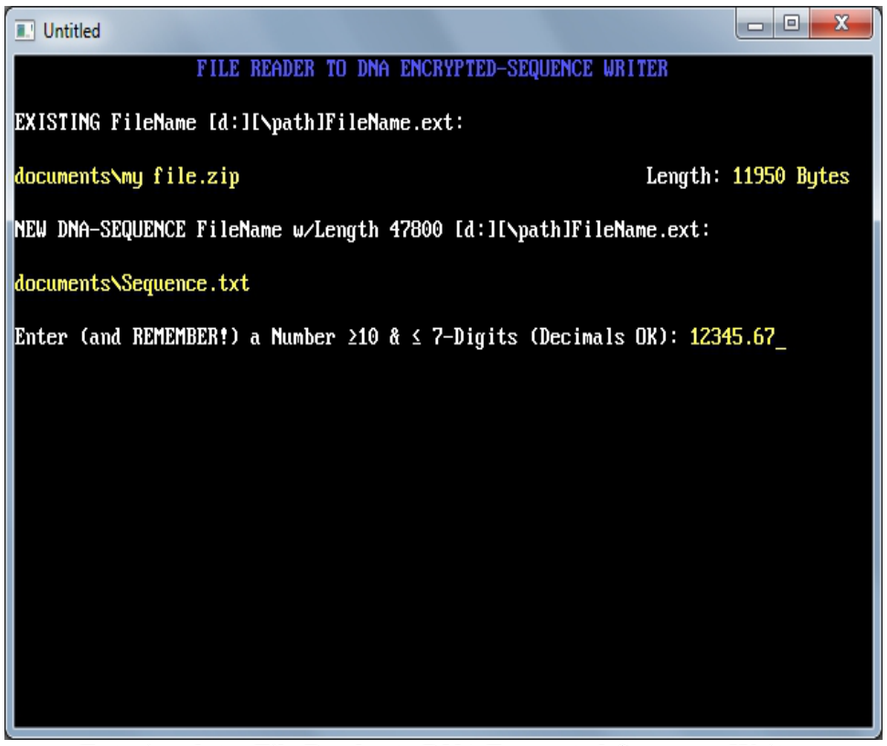


By including “pseudo” permutations with duplications in the sub-groups of two (2), you get *sixteen different sets of sub-groups* for *each* of the twenty-four (24) alphabets – even though *some* of the values in the different sub-group sets remain the same when comparing one alphabet against another.

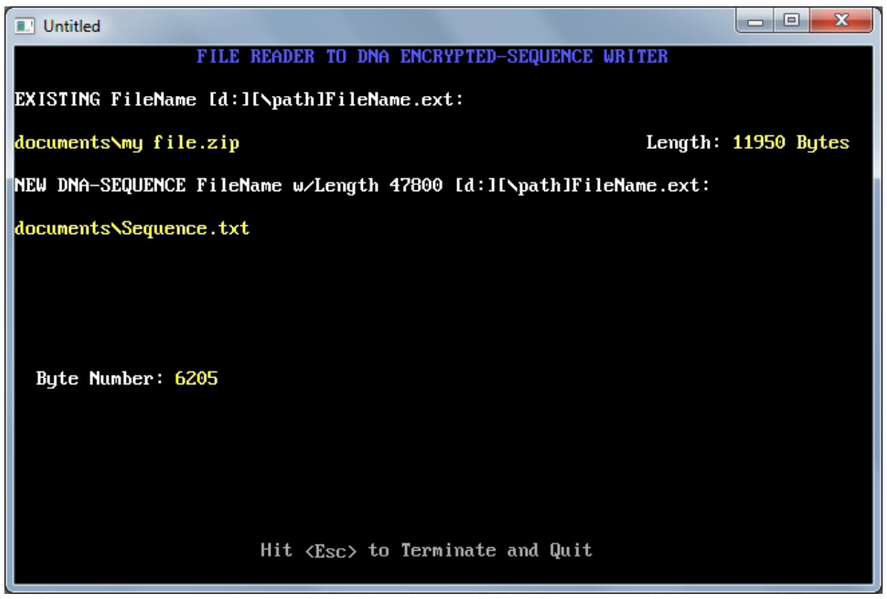
Encryption of a file is accomplished by using a pseudo-random generator to *change* to one of the twenty-four (24) alphabets for *each nibble* in the original file. Therefore, for each *byte* in the original file, *two* different alphabets are potentially used – depending on the pseudo-random generator – each having different sub-groups of two (2). Decryption is accomplished by reversing the procedure.

The pseudo-random generator is seeded by entering a number that has parameters defined in the screen prompt. The number chosen *MUST* be remembered in order to decrypt back to the original file. I have modified the normal sequence of the pseudo-random generator which can be seen when viewing my source code.

In the screen shots on the pages below, notice that when using the program “DNA Encrypted-Sequence Reader to File Writer”, decryption can be done to the *screen* instead of to a new output file. The reason for this is to decrypt a *text* file – it is not of much use if decrypting any file made up of binary characters such as a zip file, pdf file, jpeg file, etc. If you decrypt to the screen, I have chosen to use _FULLSCREEN and WIDTH 80, 50 because of a better rendering and readability of the fonts in the full screen mode.



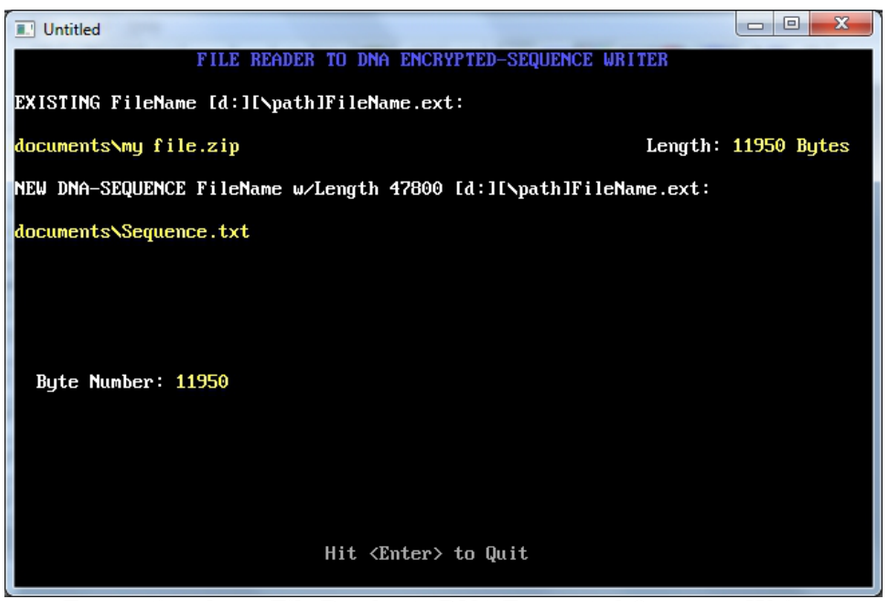
Entering data: File Reader to DNA Encrypted-Sequence Writer



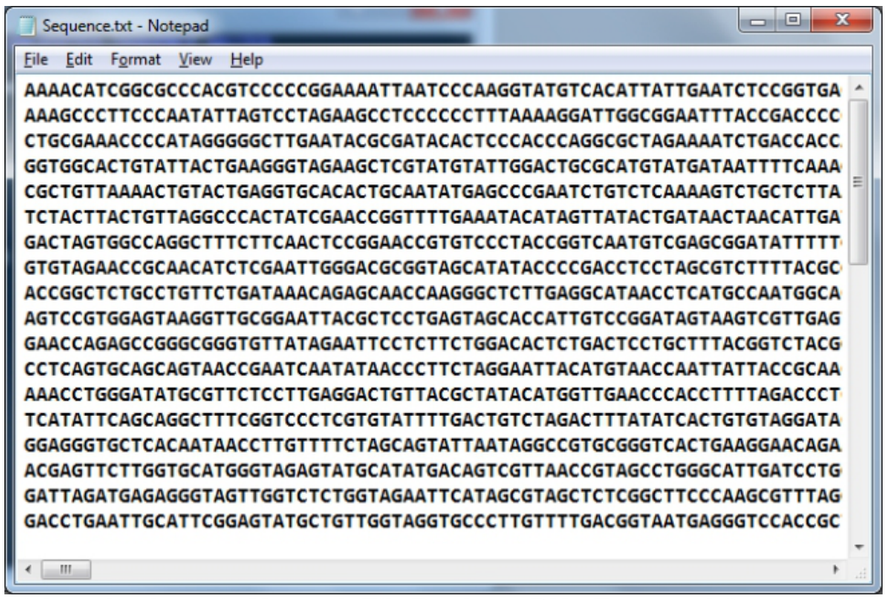
Encryption in progress: File Reader to DNA Encrypted-Sequence Writer

Note that the Number entered is cleared from the screen

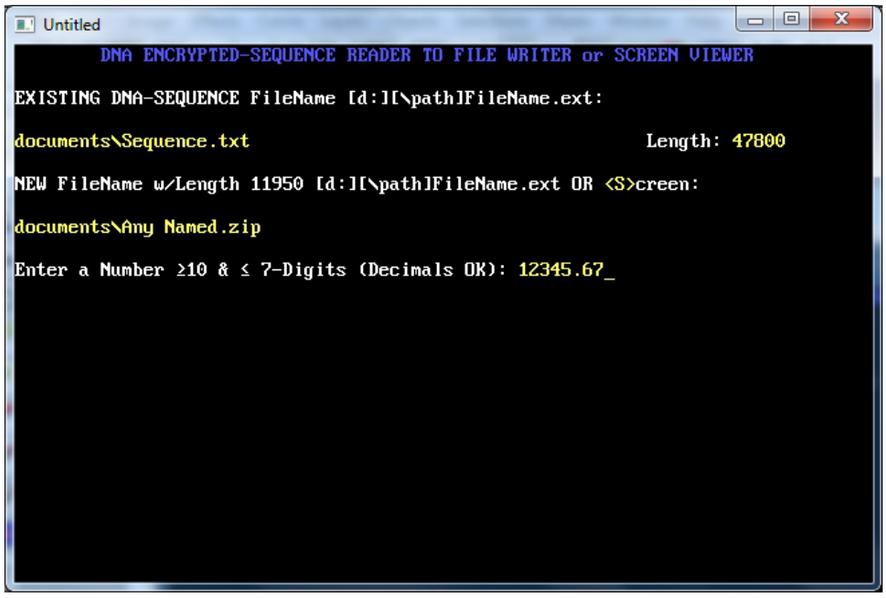
Also note that you can hit <Esc> while in progress to Kill the output file and terminate the program



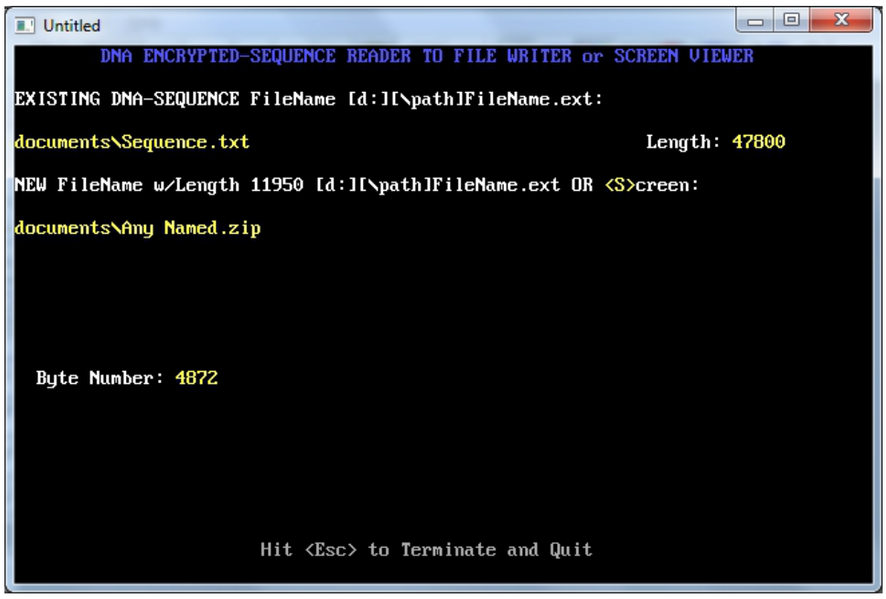
Encryption done: File Reader to DNA Encrypted-Sequence Writer



Partial results shown in Notepad: File Reader to DNA Encrypted-Sequence Writer



Entering data: DNA Encrypted-Sequence Reader to File Writer

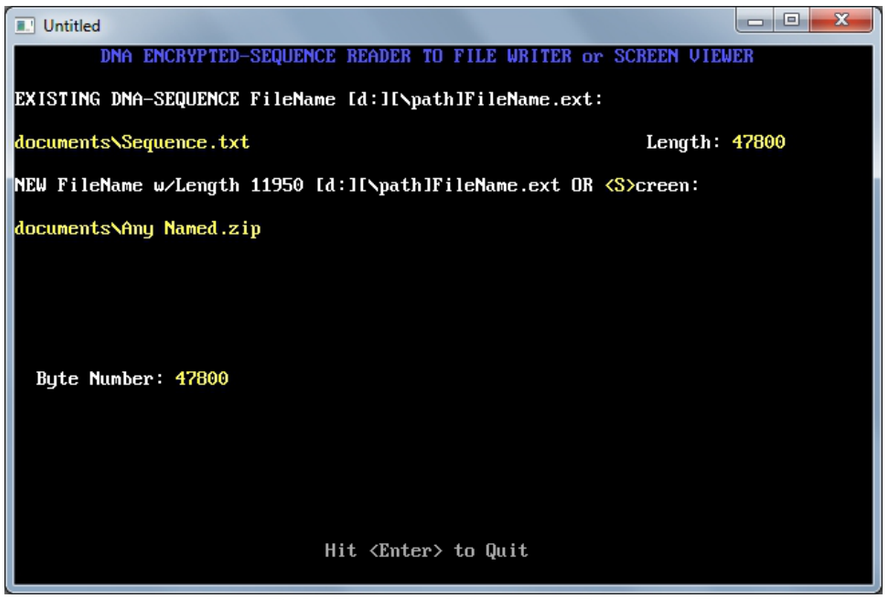


Decryption in progress: DNA Encrypted-Sequence Reader to File Writer

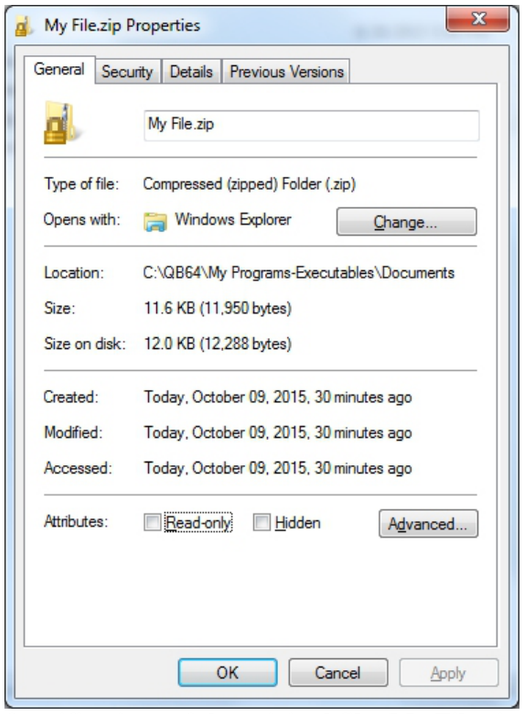
Note that the Number entered is cleared from the screen

Also note that you can hit <Esc> while in progress to Kill the output file and terminate the program

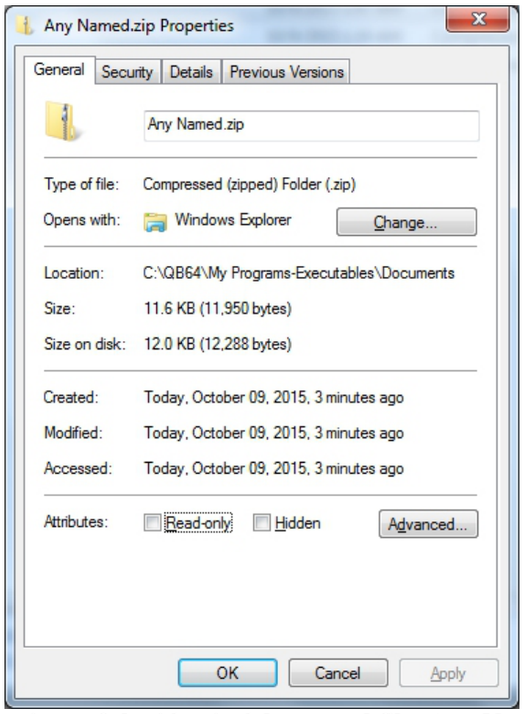
I *could* have chosen to decrypt to the screen. However, since the original file was a zip file with binary characters, it would have just been gibberish if viewed on the screen.



Decryption done: DNA Encrypted-Sequence Reader to File Writer



Specifications of the original file: My File.zip



Specifications of the *identical* decrypted file: Any Named.zip