Vernam Two Cryptographic EGngine

Author: Dan P. Milleville, dmilleville@comcast.net 978-772-2928

This cipher methodology utilizes 4 pointers pseudo-randomly set initially and pseudo-randomly advanced between blocks. These pointers, pointing to unique streams within an 8 Mbyte key, are then Xor'ed together producing an **E**ffective **K**ey **S**tream (**EKS**) and then Xor'ed with the plaintext. The fourth pointer is a mathematical construction of 3 byte accesses from each of 3 table accesses Xor'ed with the ciphertext block number. Both the sender and receiver can now possess a fixed key and still encrypt and decrypt text using the created-on-the-fly non-repeating pseudo-random **EKS**'s formed, just as the Vernam cipher engine does now.

This design is at least 5 times faster, and also experiences no loss of security, as compared to the AES. The same mathematical operator as the Vernam cipher (Xor) is used between the **EKS** and the plaintext. The **EKS** has multiple key solutions that do result in almost countless key sets that all translate correctly to the same **EKS** with no methodology that will ever be available to determine which one of these reconstructed key sets is correct. This is illustrated in Appendix A showing 5 of the almost countless reconstructed incorrect keys. When encrypting 128 characters, the 31,360 (AES Visual Basic version) programming steps executed by the AES are decreased to 640 for this Vernam Two. This results in at least a five-fold increase in speed that is needed in today's increasingly data-intensive and security-dependent world.

The (modified gkey made from 32 bytes) 8 Megabyte (8,389,631) key is hereafter referred to as the **Key.** Four pointers are pseudo-randomly determined and reference this **Key,** each using an exclusive 2,097,408 byte segment of this key. This yields a total of 1.93×10^{25} possible **EKS**'s. The four streams of numbers extracted are equal in length to the block size, 128 key numbers in this design, 29 displayed below. Examples:

Pointer 1 = **460,314** Pointer 2 = **1,620,751** Pointer 3 = **1,085,354**

Xor'ing the vertical numbers from these four streams forms an **EKS** that will eventually be Xor'ed with the plaintext, fully detailed in Appendix A, black text key streams. The example in red is 'AF' Xor '0B' Xor '96' Xor '3E' = '0C'. With the attacker not knowing the 4 key numbers, it is Algebraic law that this single equation with 4 unknowns will remain verifiably unsolvable. Note that each stream accesses a different 2Mbyte section of the 8Mbyte key. Here are 4 example key streams using the above pointer numbers and the resulting **EKS**:

```
Key Stream @2,557,721  - 09 B7 3C A5 E5 62 AF 11 48 82 86 59 EC 50 6F D3 97 2B D1 D0 79 63 FB 48 2A 49 F8 17 FE
Key Stream @1,620,751  - 19 4E A5 6C D6 0C 0B 2C 07 32 18 AB 51 7F 1A 3A 64 0C 90 FC 8F 2F 9A B9 8E 19 CA 9D 12
Key Stream @5,280,170  - DC F3 4F D5 13 2B 96 D9 32 83 92 07 4C AE D4 67 98 4D 66 E8 87 E2 9C C4 19 D7 2A A8 97
Key Stream @7,211,985  - 5D 6A 3E 14 EB 58 3E 9A 93 D6 B8 84 4D 25 5E E6 8F DF 57 94 68 A7 3E 85 59 37 7A FE CF
Effective Key Stream 1 - 91 60 E8 08 CB 1D 0C 7E EE E5 B4 71 BC A4 FF 68 E4 B5 70 50 19 09 C3 B0 E4 B0 62 DC B4
```

If the first pointer is advanced by 1 to 2,557,722, the following result is obtained. Note that pointer 4 is significantly changed because it depends on the key table values that each of the other 3 pointers are pointing. So since pointer 1 would be referencing 3 different values, pointer 4 reflects this change. The Effective Key Stream from above is copied for you to compare:

```
Key Stream @2,557,722  - B7 3C A5 E5 62 AF 11 48 82 86 59 EC 50 6F D3 97 2B D1 D0 79 63 FB 48 2A 49 F8 17 FE 82
Key Stream @1,620,751  - 19 4E A5 6C D6 0C 0B 2C 07 32 18 AB 51 7F 1A 3A 64 0C 90 FC 8F 2F 9A B9 8E 19 CA 9D 12
Key Stream @5,280,170  - DC F3 4F D5 13 2B 96 D9 32 83 92 07 4C AE D4 67 98 4D 66 E8 87 E2 9C C4 19 D7 2A A8 97
Key Stream @6,666,568  - 77 D1 4E 6D 1F 52 46 54 FC E7 99 50 E3 45 C2 34 B9 2A 7E 77 D7 F5 25 28 2C 0A 33 E9 73
Effective Key Stream 2 - 05 50 01 31 B8 DA CA E9 4B D0 4A 10 AE FB DF FE 6E BA 58 1A BC C3 6B 7F F2 3C C4 22 74
Effective Key Stream 1 - 91 60 E8 08 CB 1D 0C 7E EE E5 B4 71 BC A4 FF 68 E4 B5 70 50 19 09 C3 B0 E4 B0 62 DC B4
```

Two screen shots showing the app displaying the data above from the first example is provided in Appendix B. If pointer 1 was advanced to 2,557,721 through 2,557,730 the key stream shift, as well as the pointer 4 changes, would cause the first example above to show the **EKS**'s listed in Appendix B.

For the app to duplicate the first scenario, copy/paste this string into the demo app: 8,460313,1620751,1085354,129,143

Code/speed comparisons, AES vs Vernam Two

The following AES encrypt engine code was extracted from the Visual Basic version of the AES obtained from the internet:

```
For i = 1 To m_Nr - 1        ' 1 to 13
    For j = 0 To m_Nb - 1    ' 0 to 7
        m = j * 3
        Y(j) = m_ekey(k) Xor m_etable(X(j) And &HFF&) Xor _
            RotateLeft(m_etable(RShift(X(m_fi(m)), 8) And &HFF&), 8) Xor _
            RotateLeft(m_etable(RShift(X(m_fi(m + 1)), 16) And &HFF&), 16) Xor _
            RotateLeft(m_etable(RShift(X(m_fi(m + 2)), 24) And &HFF&), 24)
        k = k + 1
    Next
    t = X
    X = Y
    Y = t
Next
For j = 0 To m_Nb - 1    ' 0 to 7
    m = j * 3
    Y(j) = m_ekey(k) Xor m_etable(X(j) And &HFF&) Xor _
        RotateLeft(m_etable(RShift(X(m_fi(m)), 8) And &HFF&), 8) Xor _
        RotateLeft(m_etable(RShift(X(m_fi(m + 1)), 16) And &HFF&), 16) Xor _
        RotateLeft(m_etable(RShift(X(m_fi(m + 2)), 24) And &HFF&), 24)
    k = k + 1
Next
```

Executing a debug single-step of the 'y(i)…..' instruction caused it to go through 70 steps to obtain the data to Xor together to obtain the value of y(i). Considering that the 'j' loop has 8 passes to execute, 70 times 8 = 560. The 'i' loop has 13 loops plus the second loop, 14 times 560 = 7,840. This is excluding the 't = X, X = y and y = t' steps that would require more work than just a single instruction because these transfer entire structures. Considering that only 32 characters are encrypted by this AES function, and Vernam Two encrypts 128 characters per block, 4 times 7,840 = 31,360 steps would be required to encrypt the same 128 plaintext characters with the AES engine.

The following code executes the Vernam Two encryption loops, both Vernam and Transposition:

- 'i' is the loop counter
- 'ctx' is the string holder that is to contain the ciphertext output block
- 'ptxBuffer' is the string holder containing 128 characters of plaintext
- 'e1Key' is the Vernam Key Table
- 'p1' through 'p4' are the randomly set pointers that address each of 4 separate 2 Mbyte segments of the 8 Mbyte Vernam Key

```
For i = 1 To 128
    ctx = ctx + chr$(Asc(Mid$(ptxBuffer, i, 1)) Xor e1Key(p1) Xor e1Key(p2) Xor e1Key(p3) Xor e1Key(p4))
    p1 = p1 + 1: p2 = p2 + 1: p3 = p3 + 1: p4 = p4 + 1
Next i
```

When executed, it does not jump to any functions to prepare data to Xor together as the AES must do – the data is right there in the structures to use and calculate. The loop executes 5 instructions per loop, 128 loops, 5 times 128 = 640 instructions to encrypt 128 characters. At the completion of the loop, the 'ctx' string location contains the ciphertext for output.

Compare the AES's 31,360 steps to encrypt 128 characters to the Vernam Two's 640 steps – a 49 to 1 improvement. That is not including the extra steps in the AES code of 1) the byte to longword conversion of the plaintext input, 2) the structure transfers ('t = X', etc.) and 3) the step to convert the longwords back to bytes after the encryption is complete that are not shown. So the ratio is even greater, but this gives a conservative view of the drastic decrease in computational requirements for the two ciphers, helping drastically to save computational requirements.

The 32 byte random number system key

This design requires only a 32 byte key, but it is way too small for use during the execution of this algorithm. The AES's key expansion algorithm function (gkey) is borrowed from the AES cipher and enhanced for this design. It was modified to produce 2,097,408 longwords instead of the 120 it produces for the AES, and are split into 8,388,608 bytes this algorithm needs. The decrypt longwords normally produced by the 'gkey' are not needed.

The communication protocol of pointers to the decrypt engine

The 4 Vernam 2 pointers are set randomly by the encryption engine for each encrypt operation and the first 3 need to be transmitted to the decrypt engine for it to be able to decrypt the ciphertext. The 4th pointer is created from the first 3 so it is not necessary to include it in this process. The encrypt engine uses this sequence, 'ctx' contains at least the first 6 characters of the ciphertext, and 'p1' through 'p3' are pointers the function 'setPointers' initializes using the ciphertext characters for the first line only:

```
' Get the pointers using the first 6 characters of ciphertext to create the pointers, 2 steps
ptr = setPointers(ctx, p1, p2, p3)
' Insert dummy characters that will be replaced in the ciphertext line with the pointer characters
ptxBuffer = Left$(ptxBuffer, ptr - 1) + "nnnnnnnnn" + Mid$(ptxBuffer, ptr)
```

Here is the code of 'setPointers':

```
Function setPointers(s6Chrs As String, p1 As Long, p2 As Long, p3 As Long) As Byte
    Dim byte0 As Long, byte1 As Long, byte2 As Long, n As Long, ptr As Byte

    ' Get 3 byte values to be combined to form an address
    byte0 = Asc(Mid$(s6Chrs, 1, 1)): byte1 = Asc(Mid$(s6Chrs, 2, 1)): byte2 = Asc(Mid$(s6Chrs, 3, 1))
    ' Now to set the first pointer used to position the pointer field for the first block
    p1 = ((byte2 * &H10000) + (byte1 * &H100&) + byte0) And &H7FFFFF
    ' Get 3 byte values to be combined to form an address
    byte0 = v2Key(p1 + 10): byte1 = v2Key(p1 + 11): byte2 = v2Key(p1 + 12)
    ' Now to set the second pointer used to position the pointer field for the first block
    p2 = ((byte2 * &H10000) + (byte1 * &H100&) + byte0) And &H7FFFFF
    ' Get 3 byte values to be combined to form an address
    byte0 = v2Key(p2 + 10): byte1 = v2Key(p2 + 11): byte2 = v2Key(p2 + 12)
    ' Now to set the third pointer used to position the pointer field for the first block
    p3 = ((byte2 * &H10000) + (byte1 * &H100&) + byte0) And &H7FFFFF
    ' Get 3 byte values to be combined to form an address
    byte0 = v2Key(p3 + 10): byte1 = v2Key(p3 + 11): byte2 = v2Key(p3 + 12)
    ' Now to RESET the FIRST pointer used to position the pointer field for the first block
    p1 = ((byte2 * &H10000) + (byte1 * &H100&) + byte0) And &H7FFFFF
    ' Get the byte ASCII numbers from the second 3 characters of ciphertext
    byte0 = Asc(Mid$(s6Chrs, 4, 1)): byte1 = Asc(Mid$(s6Chrs, 5, 1)): byte2 = Asc(Mid$(s6Chrs, 6, 1))
    ' Get the position in the key table, forming an address in location 'n'.
    n = ((byte2 * &H10000) + (byte1 * &H100&) + byte0) And &H7FFFFF
    ' Get 3 byte values to be combined to form an address
    byte0 = v2Key(n + 10): byte1 = v2Key(n + 11): byte2 = v2Key(n + 12)
    ' Get the position in the key table, forming an address in location 'n'.
    n = ((byte2 * &H10000) + (byte1 * &H100&) + byte0) And &H7FFFFF
    ' Find a position that is within the acceptable range to use as the position of the pointer field
    Do: ptr = v2Key(n): n = n + 1: Loop While ptr < 10 Or ptr > 110
    ' Return the pointer found
    setPointers = ptr
End Function
```

The encrypt engine then executes the encrypt cycle. This next code instruction substitutes the data in the ciphertext, using the 'ptr' obtained above, then converts the 3 critical pointers to 3 characters, each inserted in the line where the 'n's were inserted in the plaintext:

```
' If the first line, insert the values as needed using ptr, p1, p2 and p3 initialized above
If ctxBlockNum = 1 Then Mid$(ctx, ptr, 9) = cvtPtr(e1p1, p1, p2, p3) + _
    cvtPtr(e1p2, p1 + 3, p2 + 3, p3 + 3) + cvtPtr(e1p3, p1 + 6, p2 + 6, p3 + 6)
```

This is the function 'cvtPtr':

```
Function cvtPtr(n As Long, p1 As Long, p2 As Long, p3 As Long) As String
    ' Take each byte field of pointer # passed and XOR it with the three table values, return 3 characters
    cvtPtr = Chr$((Int(n / &H10000) And &HFF&) Xor v2Key(p1) Xor v2Key(p2) Xor v2Key(p3)) + _
        Chr$((Int(n / &H100&) And &HFF&) Xor v2Key(p1 + 1) Xor v2Key(p2 + 1) Xor v2Key(p3 + 1)) + _
        Chr$((n And &HFF&) Xor v2Key(p1 + 2) Xor v2Key(p2 + 2) Xor v2Key(p3 + 2))
End Function
```

As you can see, each byte of each pointer is Xor'ed with 3 table values to disguise it within the ciphertext line and the resulting number converted to a character. When this is done, there is no way any attacker could ever distuinguish the difference between the pseudo-random pointer payload characters and the encrypt engine pseudo-random payload characters. Placing these 9 characters in a position that is key-table directed also gives no clue to anyone as to where this critical numerical stream starts. There are no limitations for any character as to possible characters in any position regardless of the pointer number being converted.

Appendix A

The black numbers on this page are the real key numbers; the red numbers (on the next 5 pages) are randomly generated key numbers, except the last key stream, calculated to duplicate the Internal Data Stream of the original sample here. Notice that all respective 'Plaintext ASCII Hex', 'Effective Key Stream' (both black and red), 'Ciphertext Out Line' are the same on all 6 pages. Definition: The 'Effective Key Stream' is the Xor of all 4 Key Streams.

```
Plaintext ASCII Hex  - 54 68 69 73 20 69 73 20 73 61 6D 70 6C 65 20 74 65 78 74 20 74 6F 20 64 65 6D 6F 6E 73 74
                       || || || || || || || || || || || || || || || || || || || || || || || || || || || || ||
Key Stream @2,557,721- 09 B7 3C A5 E5 62 AF 11 48 82 86 59 EC 50 6F D3 97 2B D1 D0 79 63 FB 48 2A 49 F8 17 FE 82
Key Stream @1,620,751- 19 4E A5 6C D6 0C 0B 2C 07 32 18 AB 51 7F 1A 3A 64 0C 90 FC 8F 2F 9A B9 8E 19 CA 9D 12 73
Key Stream @5,280,170- DC F3 4F D5 13 2B 96 D9 32 83 92 07 4C AE D4 67 98 4D 66 E8 87 E2 9C C4 19 D7 2A A8 97 DD
Key Stream @7,211,985- 5D 6A 3E 14 EB 58 3E 9A 93 D6 B8 84 4D 25 5E E6 8F DF 57 94 68 A7 3E 85 59 37 7A FE CF 69
                       || || || || || || || || || || || || || || || || || || || || || || || || || || || || ||
Ciphertext Out Line  - C5 08 81 7B EB 74 7F 5E 9D 84 D9 01 D0 C1 DF 1C 81 CD 04 70 6D 66 E3 D4 81 DD 0D B2 C7 31

72 61 74 65 20 74 68 65 73 65 20 73 74 65 70 73 20 6F 66 20 74 68 65 20 27 52 61 6E 64 6F 6D 20 43 69 70 68 65
|| || || || || || || || || || || || || || || || || || || || || || || || || || || || || || || || || || || ||
99 38 F7 35 A5 9D 12 57 0A 73 13 E5 D8 2A FF B5 B7 F9 68 9E 66 29 11 FD 9D ED A8 1D F3 FA 56 9F 6A C2 A1 AA CF
2B AC 0B C1 54 FD 28 CD 5F D1 2F FF 47 7A 7E 80 5D 40 1A 00 99 5E 42 2F 03 E7 CC 36 C9 7A DE 45 E2 D6 D5 AF CC
3F 73 4B 2E 70 A6 58 05 C7 FD F9 E8 55 FA B5 46 81 9D 2D 0B E7 75 AA E9 34 68 9E 4C 1E C0 09 91 21 B3 42 BF 51
77 54 2D 68 CB 40 C6 30 F5 DA 55 E6 4D 5E 18 C3 13 0B A1 25 0F 9F C9 82 31 1A 90 B5 4B E4 5F DC 3C 3D FD EE EC
|| || || || || || || || || || || || || || || || || || || || || || || || || || || || || || || || || || || ||
88 D2 EE D7 6A F2 CC CA 14 E0 B0 67 F3 91 5C C3 58 40 98 90 63 F5 55 99 BC 2A 0B BC 0B CB B3 B7 D6 F3 BB 3C DB

72 20 4F 75 74 70 75 74 73 27 20 6D 6F 64 65 20 75 73 69 6E 67 20 55 4E 41 54 54 41 43 4B 41 42 4C 45 20 41 6C
|| || || || || || || || || || || || || || || || || || || || || || || || || || || || || || || || || || || ||
5F B3 FD C5 53 A8 51 08 79 57 E4 BF 80 3F 7A D9 A9 2E 87 44 3E BD 0A 9D C4 EB 95 F7 06 4A 3F 38 59 F9 C2 FD 50
F3 DE 62 93 22 F1 9D D4 58 8F 1D 89 18 95 27 4D 74 30 08 4E 93 FC 3E 87 E9 22 7B 65 3F F7 8E 81 86 B6 EC 12 A4
15 1A BA 16 A4 5B 1C 43 5E EE 5A C2 C3 C3 51 25 B6 69 B8 10 6A F2 BB 0E AA FB 2A 2F 19 B9 95 7E 0C A3 2F E5 5A
7D 3B DE 19 A7 6E 38 54 F9 76 FB 47 B3 AE 77 96 2C 67 F5 A7 36 F7 40 EC D2 A8 9C D0 88 2E B0 D2 F5 15 6E CB 52
|| || || || || || || || || || || || || || || || || || || || || || || || || || || || || || || || || || || ||
B6 6C B4 2C 06 1C 9D BF F5 67 78 DE 87 A3 1E 07 32 63 AB D3 96 64 9A B6 14 CE 0C 2C EB 61 D5 57 6A BC 4F 80 90

67 65 62 72 61 69 63 20 6C 61 77 20 66 6F 72 20 73 65 63 75 72 69 74 79
|| || || || || || || || || || || || || || || || || || || || || || || ||
86 EA B6 29 D1 0E 09 A9 EE 74 D0 00 C0 F3 94 5D 07 B5 B7 99 EC 20 40 9F
47 71 C6 FC C8 6C 4F E4 5D B8 24 2C 60 B0 6A BF 9C 8E ED 56 BE F5 88 69
51 18 A6 04 BF 42 64 C7 7C 13 41 71 15 AB 67 33 90 97 69 99 6B BD 46 80
7B 56 9F AB 0D AD D8 7F 3B 42 07 53 5C B7 A0 65 AB F7 4C B7 03 6B 9C 21
|| || || || || || || || || || || || || || || || || || || || || || || ||
8C B0 2B 08 CA E4 99 D5 98 FC C5 2E 8F 30 4B 94 D3 3E 1C 94 48 6A 66 2E
```

```
Plaintext ASCII Hex  - 54 68 69 73 20 69 73 20 73 61 6D 70 6C 65 20 74 65 78 74 20 74 6F 20 64 65 6D 6F 6E 73 74
                       || || || || || || || || || || || || || || || || || || || || || || || || || || || || ||
Effective Key Stream - 91 60 E8 08 CB 1D 0C 7E EE E5 B4 71 BC A4 FF 68 E4 B5 70 50 19 09 C3 B0 E4 B0 62 DC B4 45
                       || || || || || || || || || || || || || || || || || || || || || || || || || || || || ||
Ciphertext Out Line  - C5 08 81 7B EB 74 7F 5E 9D 84 D9 01 D0 C1 DF 1C 81 CD 04 70 6D 66 E3 D4 81 DD 0D B2 C7 31

72 61 74 65 20 74 68 65 73 65 20 73 74 65 70 73 20 6F 66 20 74 68 65 20 27 52 61 6E 64 6F 6D 20 43 69 70 68 65
|| || || || || || || || || || || || || || || || || || || || || || || || || || || || || || || || || || || ||
FA B3 9A B2 4A 86 A4 AF 67 85 90 14 87 F4 2C B0 78 2F FE B0 17 9D 30 B9 9B 78 6A D2 6F A4 DE 97 95 9A CB 54 BE
|| || || || || || || || || || || || || || || || || || || || || || || || || || || || || || || || || || || ||
88 D2 EE D7 6A F2 CC CA 14 E0 B0 67 F3 91 5C C3 58 40 98 90 63 F5 55 99 BC 2A 0B BC 0B CB B3 B7 D6 F3 BB 3C DB

72 20 4F 75 74 70 75 74 73 27 20 6D 6F 64 65 20 75 73 69 6E 67 20 55 4E 41 54 54 41 43 4B 41 42 4C 45 20 41 6C
|| || || || || || || || || || || || || || || || || || || || || || || || || || || || || || || || || || || ||
C4 4C FB 59 72 6C E8 CB 86 40 58 B3 E8 C7 7B 27 47 10 C2 BD F1 44 CF F8 55 9A 58 6D A8 2A 94 15 26 F9 6F C1 FC
|| || || || || || || || || || || || || || || || || || || || || || || || || || || || || || || || || || || ||
B6 6C B4 2C 06 1C 9D BF F5 67 78 DE 87 A3 1E 07 32 63 AB D3 96 64 9A B6 14 CE 0C 2C EB 61 D5 57 6A BC 4F 80 90

67 65 62 72 61 69 63 20 6C 61 77 20 66 6F 72 20 73 65 63 75 72 69 74 79
|| || || || || || || || || || || || || || || || || || || || || || || ||
EB D5 49 7A AB 8D FA F5 F4 9D B2 0E E9 5F 39 B4 A0 5B 7F E1 3A 03 12 57
|| || || || || || || || || || || || || || || || || || || || || || || ||
8C B0 2B 08 CA E4 99 D5 98 FC C5 2E 8F 30 4B 94 D3 3E 1C 94 48 6A 66 2E
```

For the app to duplicate this scenario, copy/paste this string: 8,460313,1620751,1085354,129,143

#1 (randomly created incorrect key that does work for the example on page 4)

```
Plaintext ASCII Hex  - 54 68 69 73 20 69 73 20 73 61 6D 70 6C 65 20 74 65 78 74 20 74 6F 20 64 65 6D 6F 6E 73 74
                       || || || || || || || || || || || || || || || || || || || || || || || || || || || || ||
Key Stream @?,???,??- 14 14 40 2F F4 4D 86 79 3D C4 44 58 94 E3 57 AC 7E E3 6E 0F 16 AC D6 50 F2 F2 8A 2C 52 8C
Key Stream @?,???,??- 52 69 4F 17 6C 6A C4 11 EE 46 AB 00 9F 37 A2 2F 98 FC CC C0 3E 4F 6D 3B 7D 86 C9 55 46 86
Key Stream @?,???,??- AB EF 11 46 F0 AD A0 8E 0F 0D 3F E4 A0 17 89 97 4A 35 85 6D D0 F3 D3 05 5C F8 1A 6F 0A 44
Key Stream @?,???,??- 7C F2 F6 76 A3 97 EE 98 32 6A 64 CD 17 67 83 7C 48 9F 57 F2 E1 19 AB DE 37 3C 3B CA AA 0B
                       || || || || || || || || || || || || || || || || || || || || || || || || || || || || ||
Ciphertext Out Line  - C5 08 81 7B EB 74 7F 5E 9D 84 D9 01 D0 C1 DF 1C 81 CD 04 70 6D 66 E3 D4 81 DD 0D B2 C7 31

72 61 74 65 20 74 68 65 73 65 20 73 74 65 70 73 20 6F 66 20 74 68 65 20 27 52 61 6E 64 6F 6D 20 43 69 70 68 65
|| || || || || || || || || || || || || || || || || || || || || || || || || || || || || || || || || || || ||
7A 4E 8A 17 1A 89 B1 CB 2E 54 8A 9C D4 86 B1 94 D1 52 0A 6F 20 84 67 78 B8 FC 77 A2 9C 6B F6 EA 28 4D 12 EB 9A
3D FD 4A F0 9E AC F7 86 D1 ED B2 DE AB 2A 57 4A 0A B9 EE 06 33 26 AE A3 6E C1 36 B3 E0 6B 09 B5 A7 9C 4B 43 3B
B1 09 B7 AB 8B 9C 0B 91 68 B7 C4 E6 39 4A 44 64 D2 85 5D 08 D7 F2 5B CE E0 D1 53 B0 EE B3 AA E0 12 59 E7 44 5A
0C 09 ED FE 45 3F E9 73 F0 8B 6C B0 C1 12 8E 0A 71 41 47 D1 D3 CD A2 AC AD 94 78 73 FD 17 8B 28 08 12 75 B8 45
|| || || || || || || || || || || || || || || || || || || || || || || || || || || || || || || || || || || ||
88 D2 EE D7 6A F2 CC CA 14 E0 B0 67 F3 91 5C C3 58 40 98 90 63 F5 55 99 BC 2A 0B BC 0B CB B3 B7 D6 F3 BB 3C DB

72 20 4F 75 74 70 75 74 73 27 20 6D 6F 64 65 20 75 73 69 6E 67 20 55 4E 41 54 54 41 43 4B 41 42 4C 45 20 41 6C
|| || || || || || || || || || || || || || || || || || || || || || || || || || || || || || || || || || || ||
26 1B C2 DF 00 6E 24 9F 24 A3 BC 8A 17 F4 C9 E4 D8 F0 F8 4E 57 61 7F D6 38 7A 7A 42 C8 44 89 98 3F 4A A9 F0 37
B8 BA 79 04 93 EF A8 37 42 D5 3B F5 23 68 E8 A5 2E B2 1A 91 7D 03 9E BF 1A 6F 8D 11 07 8D 79 A0 93 6D DA 54 DB
CE BD 31 FF C2 53 6C CC 29 D6 5D D1 14 41 95 49 CF 6A 82 C7 3B 50 70 A7 6C A4 C3 B4 86 95 73 15 D3 DF C7 7D 29
94 50 71 7D 23 BE 08 AF C9 E0 82 1D C8 1A CF 2F 7E 38 A2 A5 E0 76 5E 36 1B 2B 6C 8A E1 76 17 38 59 01 DB 18 39
|| || || || || || || || || || || || || || || || || || || || || || || || || || || || || || || || || || || ||
B6 6C B4 2C 06 1C 9D BF F5 67 78 DE 87 A3 1E 07 32 63 AB D3 96 64 9A B6 14 CE 0C 2C EB 61 D5 57 6A BC 4F 80 90

67 65 62 72 61 69 63 20 6C 61 77 20 66 6F 72 20 73 65 63 75 72 69 74 79
|| || || || || || || || || || || || || || || || || || || || || || || ||
83 53 7E 6C 7F D2 DA B5 D5 E9 91 42 CB A5 F1 18 47 7E 10 C9 F5 B4 1E 77
DA BA C7 9E D0 9F 5B 45 40 B9 9A 31 02 4C CD 47 1A 5F DF B9 2F A4 72 16
15 55 92 7A 2E 39 9D BC 4E 94 73 FB D5 48 A2 BB AD F1 DF 93 CF 26 AF 06
A7 69 62 F2 2A F9 E6 B9 2F 59 CA 86 F5 FE A7 50 50 8B 6F 02 2F 35 D1 30
|| || || || || || || || || || || || || || || || || || || || || || || ||
8C B0 2B 08 CA E4 99 D5 98 FC C5 2E 8F 30 4B 94 D3 3E 1C 94 48 6A 66 2E
```

The above example with the 4 bogus key streams Xor'ed together into what is displayed below as the 'Effective Key Stream' is Xor'ed with the plaintext to produce the 'Internal Data Stream'. Notice the Plaintext ASCII hex and Internal Data Streams are identical in both the set above and this one and, most importantly, the Xor of the 4 streams above is identical to that on page 5:

```
Plaintext ASCII Hex  - 54 68 69 73 20 69 73 20 73 61 6D 70 6C 65 20 74 65 78 74 20 74 6F 20 64 65 6D 6F 6E 73 74
                       || || || || || || || || || || || || || || || || || || || || || || || || || || || || ||
Effective Key Stream - 91 60 E8 08 CB 1D 0C 7E EE E5 B4 71 BC A4 FF 68 E4 B5 70 50 19 09 C3 B0 E4 B0 62 DC B4 45
                       || || || || || || || || || || || || || || || || || || || || || || || || || || || || ||
Ciphertext Out Line  - C5 08 81 7B EB 74 7F 5E 9D 84 D9 01 D0 C1 DF 1C 81 CD 04 70 6D 66 E3 D4 81 DD 0D B2 C7 31

72 61 74 65 20 74 68 65 73 65 20 73 74 65 70 73 20 6F 66 20 74 68 65 20 27 52 61 6E 64 6F 6D 20 43 69 70 68 65
|| || || || || || || || || || || || || || || || || || || || || || || || || || || || || || || || || || || ||
FA B3 9A B2 4A 86 A4 AF 67 85 90 14 87 F4 2C B0 78 2F FE B0 17 9D 30 B9 9B 78 6A D2 6F A4 DE 97 95 9A CB 54 BE
|| || || || || || || || || || || || || || || || || || || || || || || || || || || || || || || || || || || ||
88 D2 EE D7 6A F2 CC CA 14 E0 B0 67 F3 91 5C C3 58 40 98 90 63 F5 55 99 BC 2A 0B BC 0B CB B3 B7 D6 F3 BB 3C DB

72 20 4F 75 74 70 75 74 73 27 20 6D 6F 64 65 20 75 73 69 6E 67 20 55 4E 41 54 54 41 43 4B 41 42 4C 45 20 41 6C
|| || || || || || || || || || || || || || || || || || || || || || || || || || || || || || || || || || || ||
C4 4C FB 59 72 6C E8 CB 86 40 58 B3 E8 C7 7B 27 47 10 C2 BD F1 44 CF F8 55 9A 58 6D A8 2A 94 15 26 F9 6F C1 FC
|| || || || || || || || || || || || || || || || || || || || || || || || || || || || || || || || || || || ||
B6 6C B4 2C 06 1C 9D BF F5 67 78 DE 87 A3 1E 07 32 63 AB D3 96 64 9A B6 14 CE 0C 2C EB 61 D5 57 6A BC 4F 80 90

67 65 62 72 61 69 63 20 6C 61 77 20 66 6F 72 20 73 65 63 75 72 69 74 79
|| || || || || || || || || || || || || || || || || || || || || || || ||
EB D5 49 7A AB 8D FA F5 F4 9D B2 0E E9 5F 39 B4 A0 5B 7F E1 3A 03 12 57
|| || || || || || || || || || || || || || || || || || || || || || || ||
8C B0 2B 08 CA E4 99 D5 98 FC C5 2E 8F 30 4B 94 D3 3E 1C 94 48 6A 66 2E
```

#2 (randomly created incorrect key that does work for the example on page 4)

```
Plaintext ASCII Hex  - 54 68 69 73 20 69 73 20 73 61 6D 70 6C 65 20 74 65 78 74 20 74 6F 20 64 65 6D 6F 6E 73 74
                       || || || || || || || || || || || || || || || || || || || || || || || || || || || || ||
Key Stream @?,???,???- 77 F3 5D E5 00 F6 EC 23 FD DB FD 6E 87 ED A0 3E 86 4C 9F 9D 94 B6 F9 11 14 95 9C CA 8D 48
Key Stream @?,???,???- 54 1A 83 DB 23 17 6D 41 2F 4B 85 D4 9F 04 87 DB 26 14 04 EC 6A 71 F3 23 B4 0D 7A A1 D9 F7
Key Stream @?,???,???- F0 03 27 D6 03 BB 5D 08 97 34 F7 B5 E1 F9 F3 F5 7C 01 C7 F9 24 DC C2 5C B0 21 A3 AB 50 16
Key Stream @?,???,???- 42 8A 11 E0 EB 47 D0 14 AB 41 3B 7E 45 B4 2B 78 38 EC 2C D8 C3 12 0B DE F4 09 27 1C B0 EC
                       || || || || || || || || || || || || || || || || || || || || || || || || || || || || ||
Ciphertext Out Line  - C5 08 81 7B EB 74 7F 5E 9D 84 D9 01 D0 C1 DF 1C 81 CD 04 70 6D 66 E3 D4 81 DD 0D B2 C7 31

72 61 74 65 20 74 68 65 73 65 20 73 74 65 70 73 20 6F 66 20 74 68 65 20 27 52 61 6E 64 6F 6D 20 43 69 70 68 65
|| || || || || || || || || || || || || || || || || || || || || || || || || || || || || || || || || || || ||
61 0C 4B B5 4C DE 38 0E 04 F3 F7 E2 90 D7 42 67 53 5E 84 06 4A 49 BA 27 83 3A 86 3E E0 DA 94 16 0D 47 31 5C BF
F9 48 67 B3 58 B4 14 D5 91 7C 62 E4 7F A9 97 6D E8 76 DC 65 C9 4A B2 C3 E1 F2 9D 5D 80 22 0B 03 90 2E 86 DA E3
7A 32 08 A1 BF 08 6D 6E B0 5C 9E 61 EF 01 84 3E 5B 38 4C 96 2D BC 25 F4 57 13 36 48 18 13 E2 93 E3 B2 3F D3 72
18 C5 BE 15 E1 E4 E5 1A 42 56 9B 73 87 8B 7D 84 98 3F EA 45 B9 22 1D A9 AE A3 47 F9 17 4F A3 11 EB 41 43 01 90
|| || || || || || || || || || || || || || || || || || || || || || || || || || || || || || || || || || || ||
88 D2 EE D7 6A F2 CC CA 14 E0 B0 67 F3 91 5C C3 58 40 98 90 63 F5 55 99 BC 2A 0B BC 0B CB B3 B7 D6 F3 BB 3C DB

72 20 4F 75 74 70 75 74 73 27 20 6D 6F 64 65 20 75 73 69 6E 67 20 55 4E 41 54 54 41 43 4B 41 42 4C 45 20 41 6C
|| || || || || || || || || || || || || || || || || || || || || || || || || || || || || || || || || || || ||
C3 28 6E 2E 9B E0 65 D4 54 B3 5F 66 1C 90 11 13 6F D3 8D FB 07 63 50 F2 09 0D 8F 76 6D 20 C1 ED AE 49 6F 22 B3
04 1A FA BE 6D 07 49 A6 84 AA 99 6A 76 A9 00 71 7B 5D 77 44 14 84 B6 5E 86 26 4E 32 96 C7 07 7D 85 CF 7C CE 4E
11 D3 31 25 CB 90 68 A0 7A 12 78 A3 8B 17 8B 2B 43 3F 27 CA 10 04 95 43 14 FB 91 69 EE 4C 32 39 CB 6E 1B 35 AD
12 AD 5E EC 4F 1B AC 19 2C 4B E6 1C 09 E9 E1 6E 10 A1 1F C8 F2 A7 BC 17 CE 4A 08 40 BD 81 60 BC C6 11 67 18 AC
|| || || || || || || || || || || || || || || || || || || || || || || || || || || || || || || || || || || ||
B6 6C B4 2C 06 1C 9D BF F5 67 78 DE 87 A3 1E 07 32 63 AB D3 96 64 9A B6 14 CE 0C 2C EB 61 D5 57 6A BC 4F 80 90

67 65 62 72 61 69 63 20 6C 61 77 20 66 6F 72 20 73 65 63 75 72 69 74 79
|| || || || || || || || || || || || || || || || || || || || || || || ||
B0 3B 01 FC 53 2F 92 10 A1 47 33 25 59 40 19 11 A5 A1 09 70 72 0A 52 16
35 F3 52 1E 72 81 E3 13 A5 5E 53 39 02 85 F5 C1 80 90 6D 83 C1 D1 B9 B6
27 E2 02 FA C4 BF 44 47 94 79 B9 65 AA 8E 94 AC D2 A1 1B 87 D3 28 84 40
49 FF 18 62 4E 9C CF B1 64 FD 6B 77 18 14 41 C8 57 CB 00 95 5A F0 7D B7
|| || || || || || || || || || || || || || || || || || || || || || || ||
8C B0 2B 08 CA E4 99 D5 98 FC C5 2E 8F 30 4B 94 D3 3E 1C 94 48 6A 66 2E
```

The above example with the 4 bogus key streams Xor'ed together into what is displayed below as the 'Effective Key Stream' is Xor'ed with the plaintext to produce the 'Internal Data Stream'. Notice the Plaintext ASCII hex and Internal Data Streams are identical in both the set above and this one and, most importantly, the Xor of the 4 streams above is identical to that on page 6:

```
Plaintext ASCII Hex  - 54 68 69 73 20 69 73 20 73 61 6D 70 6C 65 20 74 65 78 74 20 74 6F 20 64 65 6D 6F 6E 73 74
                       || || || || || || || || || || || || || || || || || || || || || || || || || || || || ||
Effective Key Stream - 91 60 E8 08 CB 1D 0C 7E EE E5 B4 71 BC A4 FF 68 E4 B5 70 50 19 09 C3 B0 E4 B0 62 DC B4 45
                       || || || || || || || || || || || || || || || || || || || || || || || || || || || || ||
Ciphertext Out Line  - C5 08 81 7B EB 74 7F 5E 9D 84 D9 01 D0 C1 DF 1C 81 CD 04 70 6D 66 E3 D4 81 DD 0D B2 C7 31

72 61 74 65 20 74 68 65 73 65 20 73 74 65 70 73 20 6F 66 20 74 68 65 20 27 52 61 6E 64 6F 6D 20 43 69 70 68 65
|| || || || || || || || || || || || || || || || || || || || || || || || || || || || || || || || || || || ||
FA B3 9A B2 4A 86 A4 AF 67 85 90 14 87 F4 2C B0 78 2F FE B0 17 9D 30 B9 9B 78 6A D2 6F A4 DE 97 95 9A CB 54 BE
|| || || || || || || || || || || || || || || || || || || || || || || || || || || || || || || || || || || ||
88 D2 EE D7 6A F2 CC CA 14 E0 B0 67 F3 91 5C C3 58 40 98 90 63 F5 55 99 BC 2A 0B BC 0B CB B3 B7 D6 F3 BB 3C DB

72 20 4F 75 74 70 75 74 73 27 20 6D 6F 64 65 20 75 73 69 6E 67 20 55 4E 41 54 54 41 43 4B 41 42 4C 45 20 41 6C
|| || || || || || || || || || || || || || || || || || || || || || || || || || || || || || || || || || || ||
C4 4C FB 59 72 6C E8 CB 86 40 58 B3 E8 C7 7B 27 47 10 C2 BD F1 44 CF F8 55 9A 58 6D A8 2A 94 15 26 F9 6F C1 FC
|| || || || || || || || || || || || || || || || || || || || || || || || || || || || || || || || || || || ||
B6 6C B4 2C 06 1C 9D BF F5 67 78 DE 87 A3 1E 07 32 63 AB D3 96 64 9A B6 14 CE 0C 2C EB 61 D5 57 6A BC 4F 80 90

67 65 62 72 61 69 63 20 6C 61 77 20 66 6F 72 20 73 65 63 75 72 69 74 79
|| || || || || || || || || || || || || || || || || || || || || || || ||
EB D5 49 7A AB 8D FA F5 F4 9D B2 0E E9 5F 39 B4 A0 5B 7F E1 3A 03 12 57
|| || || || || || || || || || || || || || || || || || || || || || || ||
8C B0 2B 08 CA E4 99 D5 98 FC C5 2E 8F 30 4B 94 D3 3E 1C 94 48 6A 66 2E
```

#3 (randomly created incorrect key that does work for the example on page 4)

```
Plaintext ASCII Hex  - 54 68 69 73 20 69 73 20 73 61 6D 70 6C 65 20 74 65 78 74 20 74 6F 20 64 65 6D 6F 6E 73 74
                       || || || || || || || || || || || || || || || || || || || || || || || || || || || || ||
Key Stream @?,???,???- 33 60 7F AB EE F1 AD B4 F9 71 42 97 32 F1 3B 72 7B A0 13 11 40 BB E0 FE 28 B9 AE C9 19 B7
Key Stream @?,???,???- 00 49 35 DC 54 D2 BB 5F A8 20 55 A3 57 B5 7D 7C A2 B3 2F B4 3D 10 9B E0 1F 18 0E 81 85 28
Key Stream @?,???,???- AC F1 20 BC E0 67 20 16 42 CF 08 48 F2 90 1E 40 5A 3A 19 17 8B D7 19 78 DA 73 EB 53 B6 E9
Key Stream @?,???,???- 0E B8 82 C3 91 59 3A 83 FD 7B AB 0D 2B 70 A7 26 67 9C 55 E2 EF 75 A1 D6 09 62 29 C7 9E 33
                       || || || || || || || || || || || || || || || || || || || || || || || || || || || || ||
Ciphertext Out Line  - C5 08 81 7B EB 74 7F 5E 9D 84 D9 01 D0 C1 DF 1C 81 CD 04 70 6D 66 E3 D4 81 DD 0D B2 C7 31

72 61 74 65 20 74 68 65 73 65 20 73 74 65 70 73 20 6F 66 20 74 68 65 20 27 52 61 6E 64 6F 6D 20 43 69 70 68 65
|| || || || || || || || || || || || || || || || || || || || || || || || || || || || || || || || || || || ||
1A 93 72 21 32 E0 84 A1 8C 9D 51 4C A5 70 87 0A 7B 3F BF 46 6D 02 F1 D7 52 CC 3B 2D D7 17 D2 BF 1D 50 48 B3 92
1D 5F 47 A9 14 22 5F 4A 9D 32 B6 80 B4 CF 31 42 CA 73 FB 01 37 67 14 E6 E5 A3 1A 4F 39 EB 56 8B 45 27 09 D4 2C
7D C2 CC EF 9F 3B 5D 64 FB F7 DE D8 57 A1 D1 5C 09 43 E2 8B BA 6C 45 D8 5E A5 4E 1A 72 16 51 54 78 74 BB C9 AF
80 BD 63 D5 F3 7F 22 20 8D DD A9 00 C1 EA 4B A4 C0 20 58 7C F7 94 90 50 72 B2 05 AA F3 4E 0B F7 B5 99 31 FA AF
|| || || || || || || || || || || || || || || || || || || || || || || || || || || || || || || || || || || ||
88 D2 EE D7 6A F2 CC CA 14 E0 B0 67 F3 91 5C C3 58 40 98 90 63 F5 55 99 BC 2A 0B BC 0B CB B3 B7 D6 F3 BB 3C DB

72 20 4F 75 74 70 75 74 73 27 20 6D 6F 64 65 20 75 73 69 6E 67 20 55 4E 41 54 54 41 43 4B 41 42 4C 45 20 41 6C
|| || || || || || || || || || || || || || || || || || || || || || || || || || || || || || || || || || || ||
A2 FF CE 0F 06 ED 9E 33 8B 91 E6 FE E1 35 4A B0 2C 65 3B E3 06 AE 24 E7 D5 9B D5 5D 4E ED 2E 16 49 19 10 2D C7
01 F8 8C D5 C4 35 BB D2 74 2F C9 92 29 FC 77 C5 B0 E5 23 10 64 32 3C F1 48 37 03 F4 69 9E 83 17 72 F6 D2 79 90
2D 71 3F 17 D6 68 93 A4 D4 85 7D C1 49 B6 A4 07 41 51 E4 3E 1C AB DE 94 3E FC C1 1C F5 72 57 89 AA 93 52 2B 4B
4A 3A 86 94 66 DC 5E 8E AD 7B 0A 1E 69 B8 E2 55 9A C1 3E 70 8F 73 09 7A F6 CA 4F D8 7A 2B 6E 9D B7 85 FF BE E0
|| || || || || || || || || || || || || || || || || || || || || || || || || || || || || || || || || || || ||
B6 6C B4 2C 06 1C 9D BF F5 67 78 DE 87 A3 1E 07 32 63 AB D3 96 64 9A B6 14 CE 0C 2C EB 61 D5 57 6A BC 4F 80 90

67 65 62 72 61 69 63 20 6C 61 77 20 66 6F 72 20 73 65 63 75 72 69 74 79
|| || || || || || || || || || || || || || || || || || || || || || || ||
06 80 63 40 07 1C BA 32 16 C3 54 E6 53 7F 8C 84 BB B6 55 59 2C 91 D3 67
7C 88 49 88 29 80 EE 15 FB CE 72 86 87 FF 34 98 BF 97 CD 91 FF C7 42 9D
D7 D7 29 F8 85 4B FF EC CE 5D 3B AD 27 53 73 76 EA 98 89 E4 30 86 22 DF
46 0A 4A 4A 00 5A 51 3E D7 CD AF C3 1A 8C F2 DE 4E E2 6E CD D9 D3 A1 72
|| || || || || || || || || || || || || || || || || || || || || || || ||
8C B0 2B 08 CA E4 99 D5 98 FC C5 2E 8F 30 4B 94 D3 3E 1C 94 48 6A 66 2E
```

The above example with the 4 bogus key streams Xor'ed together into what is displayed below as the 'Effective Key Stream' is Xor'ed with the plaintext to produce the 'Internal Data Stream'. Notice the Plaintext ASCII hex and Internal Data Streams are identical in both the set above and this one and, most importantly, the Xor of the 4 streams above is identical to that on page 6:

```
Plaintext ASCII Hex  - 54 68 69 73 20 69 73 20 73 61 6D 70 6C 65 20 74 65 78 74 20 74 6F 20 64 65 6D 6F 6E 73 74
                       || || || || || || || || || || || || || || || || || || || || || || || || || || || || ||
Effective Key Stream - 91 60 E8 08 CB 1D 0C 7E EE E5 B4 71 BC A4 FF 68 E4 B5 70 50 19 09 C3 B0 E4 B0 62 DC B4 45
                       || || || || || || || || || || || || || || || || || || || || || || || || || || || || ||
Ciphertext Out Line  - C5 08 81 7B EB 74 7F 5E 9D 84 D9 01 D0 C1 DF 1C 81 CD 04 70 6D 66 E3 D4 81 DD 0D B2 C7 31

72 61 74 65 20 74 68 65 73 65 20 73 74 65 70 73 20 6F 66 20 74 68 65 20 27 52 61 6E 64 6F 6D 20 43 69 70 68 65
|| || || || || || || || || || || || || || || || || || || || || || || || || || || || || || || || || || || ||
FA B3 9A B2 4A 86 A4 AF 67 85 90 14 87 F4 2C B0 78 2F FE B0 17 9D 30 B9 9B 78 6A D2 6F A4 DE 97 95 9A CB 54 BE
|| || || || || || || || || || || || || || || || || || || || || || || || || || || || || || || || || || || ||
88 D2 EE D7 6A F2 CC CA 14 E0 B0 67 F3 91 5C C3 58 40 98 90 63 F5 55 99 BC 2A 0B BC 0B CB B3 B7 D6 F3 BB 3C DB

72 20 4F 75 74 70 75 74 73 27 20 6D 6F 64 65 20 75 73 69 6E 67 20 55 4E 41 54 54 41 43 4B 41 42 4C 45 20 41 6C
|| || || || || || || || || || || || || || || || || || || || || || || || || || || || || || || || || || || ||
C4 4C FB 59 72 6C E8 CB 86 40 58 B3 E8 C7 7B 27 47 10 C2 BD F1 44 CF F8 55 9A 58 6D A8 2A 94 15 26 F9 6F C1 FC
|| || || || || || || || || || || || || || || || || || || || || || || || || || || || || || || || || || || ||
B6 6C B4 2C 06 1C 9D BF F5 67 78 DE 87 A3 1E 07 32 63 AB D3 96 64 9A B6 14 CE 0C 2C EB 61 D5 57 6A BC 4F 80 90

67 65 62 72 61 69 63 20 6C 61 77 20 66 6F 72 20 73 65 63 75 72 69 74 79
|| || || || || || || || || || || || || || || || || || || || || || || ||
EB D5 49 7A AB 8D FA F5 F4 9D B2 0E E9 5F 39 B4 A0 5B 7F E1 3A 03 12 57
|| || || || || || || || || || || || || || || || || || || || || || || ||
8C B0 2B 08 CA E4 99 D5 98 FC C5 2E 8F 30 4B 94 D3 3E 1C 94 48 6A 66 2E
```

#4 (randomly created incorrect key that does work for the example on page 4)

```
Plaintext ASCII Hex  - 54 68 69 73 20 69 73 20 73 61 6D 70 6C 65 20 74 65 78 74 20 74 6F 20 64 65 6D 6F 6E 73 74
                       || || || || || || || || || || || || || || || || || || || || || || || || || || || || ||
Key Stream @?,???,???- 77 02 02 51 A5 FE C2 96 AD DE 5A D0 C6 1E 32 A3 83 82 68 EA 11 BB A5 7B AF 5B 3D 3C 6D FA
Key Stream @?,???,???- 2C 29 98 03 81 D8 89 20 8A FF 0E BE 91 D8 59 1C CB CF 86 DE A6 78 06 AB 52 D4 4A 09 8D C4
Key Stream @?,???,???- 84 41 7B 45 A5 10 D1 E2 E8 07 20 E6 9E 68 81 05 55 E2 52 C0 F9 A5 EF 92 28 A2 B3 5A D4 21
Key Stream @?,???,???- 4E 0A 09 1F 4A 2B 96 2A 21 C3 C0 F9 75 0A 15 D2 F9 1A CC A4 57 6F 8F F2 31 9D A6 B3 80 5A
                       || || || || || || || || || || || || || || || || || || || || || || || || || || || || ||
Ciphertext Out Line  - C5 08 81 7B EB 74 7F 5E 9D 84 D9 01 D0 C1 DF 1C 81 CD 04 70 6D 66 E3 D4 81 DD 0D B2 C7 31

72 61 74 65 20 74 68 65 73 65 20 73 74 65 70 73 20 6F 66 20 74 68 65 20 27 52 61 6E 64 6F 6D 20 43 69 70 68 65
|| || || || || || || || || || || || || || || || || || || || || || || || || || || || || || || || || || || ||
8D 66 28 A9 91 17 C5 97 C4 1E AC 01 3C 02 88 5E 05 53 3C CB BD 35 28 B4 70 29 9D C3 B5 1D 9C 0D 08 B6 D6 7B 6A
B3 77 B3 CD BA CB 04 0E 5B CE 37 7E C6 15 3F 69 48 22 5D 2C 76 BA 66 2F 9C 88 55 23 64 F3 13 7D CA 3C 85 32 3F
2E 0B 1F 88 A7 92 72 6A D4 08 DF CE D8 18 B8 C0 7D B2 6F 1D B2 F0 B0 9B 47 48 BD 16 E6 F3 08 22 55 9C 16 CC E9
EA A9 1E 5E C6 C8 17 5C 2C 5D D4 A5 A5 FB 23 47 48 EC F0 4A 6E E2 CE B9 30 91 1F 24 58 B9 59 C5 02 8C 8E D1 02
|| || || || || || || || || || || || || || || || || || || || || || || || || || || || || || || || || || || ||
88 D2 EE D7 6A F2 CC CA 14 E0 B0 67 F3 91 5C C3 58 40 98 90 63 F5 55 99 BC 2A 0B BC 0B CB B3 B7 D6 F3 BB 3C DB

72 20 4F 75 74 70 75 74 73 27 20 6D 6F 64 65 20 75 73 69 6E 67 20 55 4E 41 54 54 41 43 4B 41 42 4C 45 20 41 6C
|| || || || || || || || || || || || || || || || || || || || || || || || || || || || || || || || || || || ||
DC E0 75 36 87 B6 D5 46 C8 72 0A 46 1A 4D 8B 57 E9 D6 7E E7 63 70 A8 CF 7E 11 98 AE 49 38 0E 84 77 2F B1 91 28
DA D6 B4 1B 19 06 2C 48 F1 B2 F8 2C 23 26 67 D6 9A A0 B3 6C 00 E5 3E D4 9E 09 10 A0 78 7A 90 3F A8 4E 82 61 FA
4D 92 B3 4D 95 B0 47 75 FD C4 5F DC AD 22 01 04 5A 37 6D 2F 72 E8 6A 5A 8A 2A D4 7A 07 E2 66 FA 0B FE C3 73 A2
8F E8 89 39 79 6C 56 B0 42 44 F5 05 7C 8E 96 A2 6E 51 62 19 E0 39 33 B9 3F A8 04 19 9E 8A 6C 54 F2 66 9F 42 8C
|| || || || || || || || || || || || || || || || || || || || || || || || || || || || || || || || || || || ||
B6 6C B4 2C 06 1C 9D BF F5 67 78 DE 87 A3 1E 07 32 63 AB D3 96 64 9A B6 14 CE 0C 2C EB 61 D5 57 6A BC 4F 80 90

67 65 62 72 61 69 63 20 6C 61 77 20 66 6F 72 20 73 65 63 75 72 69 74 79
|| || || || || || || || || || || || || || || || || || || || || || || ||
82 F1 70 FA BE 39 6E 4F 6E 59 2F E0 52 31 84 82 E2 A2 86 37 02 B1 F6 77
CE 41 AB C9 42 3D 9B 18 FE B6 BD F4 AD BA 0F 08 DC 5C B0 81 C9 35 CF 24
D4 CB AD 2D BA 1C D6 0D E5 E5 01 93 86 F2 E1 27 22 F4 C5 B8 32 11 D7 63
73 AE 3F 64 ED 95 D9 AF 81 97 21 89 90 26 53 19 BC 51 8C EF C3 96 FC 67
|| || || || || || || || || || || || || || || || || || || || || || || ||
8C B0 2B 08 CA E4 99 D5 98 FC C5 2E 8F 30 4B 94 D3 3E 1C 94 48 6A 66 2E
```

The above example with the 4 bogus key streams Xor'ed together into what is displayed below as the 'Effective Key Stream' is Xor'ed with the plaintext to produce the 'Internal Data Stream'. Notice the Plaintext ASCII hex and Internal Data Streams are identical in both the set above and this one and, most importantly, the Xor of the 4 streams above is identical to that on page 6:

```
Plaintext ASCII Hex  - 54 68 69 73 20 69 73 20 73 61 6D 70 6C 65 20 74 65 78 74 20 74 6F 20 64 65 6D 6F 6E 73 74
                       || || || || || || || || || || || || || || || || || || || || || || || || || || || || ||
Effective Key Stream - 91 60 E8 08 CB 1D 0C 7E EE E5 B4 71 BC A4 FF 68 E4 B5 70 50 19 09 C3 B0 E4 B0 62 DC B4 45
                       || || || || || || || || || || || || || || || || || || || || || || || || || || || || ||
Ciphertext Out Line  - C5 08 81 7B EB 74 7F 5E 9D 84 D9 01 D0 C1 DF 1C 81 CD 04 70 6D 66 E3 D4 81 DD 0D B2 C7 31

72 61 74 65 20 74 68 65 73 65 20 73 74 65 70 73 20 6F 66 20 74 68 65 20 27 52 61 6E 64 6F 6D 20 43 69 70 68 65
|| || || || || || || || || || || || || || || || || || || || || || || || || || || || || || || || || || || ||
FA B3 9A B2 4A 86 A4 AF 67 85 90 14 87 F4 2C B0 78 2F FE B0 17 9D 30 B9 9B 78 6A D2 6F A4 DE 97 95 9A CB 54 BE
|| || || || || || || || || || || || || || || || || || || || || || || || || || || || || || || || || || || ||
88 D2 EE D7 6A F2 CC CA 14 E0 B0 67 F3 91 5C C3 58 40 98 90 63 F5 55 99 BC 2A 0B BC 0B CB B3 B7 D6 F3 BB 3C DB

72 20 4F 75 74 70 75 74 73 27 20 6D 6F 64 65 20 75 73 69 6E 67 20 55 4E 41 54 54 41 43 4B 41 42 4C 45 20 41 6C
|| || || || || || || || || || || || || || || || || || || || || || || || || || || || || || || || || || || ||
C4 4C FB 59 72 6C E8 CB 86 40 58 B3 E8 C7 7B 27 47 10 C2 BD F1 44 CF F8 55 9A 58 6D A8 2A 94 15 26 F9 6F C1 FC
|| || || || || || || || || || || || || || || || || || || || || || || || || || || || || || || || || || || ||
B6 6C B4 2C 06 1C 9D BF F5 67 78 DE 87 A3 1E 07 32 63 AB D3 96 64 9A B6 14 CE 0C 2C EB 61 D5 57 6A BC 4F 80 90

67 65 62 72 61 69 63 20 6C 61 77 20 66 6F 72 20 73 65 63 75 72 69 74 79
|| || || || || || || || || || || || || || || || || || || || || || || ||
EB D5 49 7A AB 8D FA F5 F4 9D B2 0E E9 5F 39 B4 A0 5B 7F E1 3A 03 12 57
|| || || || || || || || || || || || || || || || || || || || || || || ||
8C B0 2B 08 CA E4 99 D5 98 FC C5 2E 8F 30 4B 94 D3 3E 1C 94 48 6A 66 2E
```

#5 (randomly created incorrect key that does work for the example on page 4)

```
Plaintext ASCII Hex  - 54 68 69 73 20 69 73 20 73 61 6D 70 6C 65 20 74 65 78 74 20 74 6F 20 64 65 6D 6F 6E 73 74
                       || || || || || || || || || || || || || || || || || || || || || || || || || || || || ||
Key Stream @?,???,???- 7A 6B 6E 90 69 9D 3A 6C DA FD A2 8C 94 01 DC 3A D1 DB 24 AB FA A7 21 4D A7 D6 79 4F D6 D9
Key Stream @?,???,???- 53 1D C6 99 C1 AC DF 26 48 B4 9B 23 86 86 CE E4 BD 3B 2C F7 9F 2F 36 45 58 BF 67 6F DC 46
Key Stream @?,???,???- F8 57 15 02 C0 FE 17 BD 5B 2B C2 6C 1E DB 4E 8B 71 B6 4E DE 72 93 EF 16 CA 4C 70 C5 CE C3
Key Stream @?,???,???- 40 41 55 03 A3 D2 FE 89 27 87 4F B2 B0 F8 A3 3D F9 E3 36 D2 0E 12 3B AE D1 95 0C 39 70 19
                       || || || || || || || || || || || || || || || || || || || || || || || || || || || || ||
Ciphertext Out Line  - C5 08 81 7B EB 74 7F 5E 9D 84 D9 01 D0 C1 DF 1C 81 CD 04 70 6D 66 E3 D4 81 DD 0D B2 C7 31

72 61 74 65 20 74 68 65 73 65 20 73 74 65 70 73 20 6F 66 20 74 68 65 20 27 52 61 6E 64 6F 6D 20 43 69 70 68 65
|| || || || || || || || || || || || || || || || || || || || || || || || || || || || || || || || || || || ||
B2 72 69 11 4B 0E E5 F7 1B 67 69 87 E8 31 EC F1 4B 60 BA 82 C2 34 30 CF 16 84 B6 24 EF 4A FD 2B B4 6F 7A E0 A8
9E 67 72 54 CE 4C 81 30 E5 8A 99 90 F9 32 C4 97 B8 96 0E 0F 11 29 52 E7 1F 95 46 EF 55 1C C9 D1 EF 33 6B 52 97
E1 84 D9 34 1F 61 E7 BD 0F 8E 75 2B 19 47 98 06 E7 04 5C DE 5F DF 5F D6 DE 47 AD 71 B5 E5 DD 9D E8 CE 6C FD 25
37 22 58 C3 D0 A5 27 D5 96 E6 15 28 8F B0 9C D0 6C DD 16 E3 9B 5F 0D 47 4C 2E 37 68 60 17 37 F0 26 08 B6 1B A4
|| || || || || || || || || || || || || || || || || || || || || || || || || || || || || || || || || || || ||
88 D2 EE D7 6A F2 CC CA 14 E0 B0 67 F3 91 5C C3 58 40 98 90 63 F5 55 99 BC 2A 0B BC 0B CB B3 B7 D6 F3 BB 3C DB

72 20 4F 75 74 70 75 74 73 27 20 6D 6F 64 65 20 75 73 69 6E 67 20 55 4E 41 54 54 41 43 4B 41 42 4C 45 20 41 6C
|| || || || || || || || || || || || || || || || || || || || || || || || || || || || || || || || || || || ||
D8 22 B4 42 B1 67 FF CC F7 72 AB 3E 4B A1 1E 5C F5 43 67 0D 36 91 AC FE D3 F0 D3 77 DE 65 6F 7C C6 58 EC 27 A1
5B 3D CA 57 43 DA E1 BE 1B B5 8B 55 3A A5 46 44 F3 AD 91 40 B4 10 AF B7 5F 95 9D 3E ED 16 50 3A 3F 4E 33 CD 47
EB 58 7A 02 A4 00 0B 7B D3 3A C3 AA E7 B6 1C C8 57 92 83 63 49 3D 23 E1 82 90 F5 35 91 29 08 DF 36 0A 0C 28 18
AC 0B FF 4E 24 D1 FD C2 B9 BD BB 72 7E 75 3F F7 16 6C B7 93 3A F8 EF 50 5B 6F E3 11 0A 70 A3 8C E9 E5 BC 03 02
|| || || || || || || || || || || || || || || || || || || || || || || || || || || || || || || || || || || ||
B6 6C B4 2C 06 1C 9D BF F5 67 78 DE 87 A3 1E 07 32 63 AB D3 96 64 9A B6 14 CE 0C 2C EB 61 D5 57 6A BC 4F 80 90

67 65 62 72 61 69 63 20 6C 61 77 20 66 6F 72 20 73 65 63 75 72 69 74 79
|| || || || || || || || || || || || || || || || || || || || || || || ||
8D FE 45 B8 5F 46 2E 53 3E 45 E9 C6 6A 86 7F BB F1 07 9F C1 E6 E7 72 15
E3 DF 03 57 B1 1C 3D C8 59 18 DC CF 0A E6 31 9A 8E 39 A4 B2 97 60 24 8C
73 25 9F A7 FC ED CE 4E F5 CA 41 DD 02 71 B8 34 F1 BA AB 34 DE A6 FE E2
F6 D1 90 32 B9 3A 27 20 66 0A C6 DA 8B 4E CF A1 2E DF EF A6 95 22 BA 2C
|| || || || || || || || || || || || || || || || || || || || || || || ||
8C B0 2B 08 CA E4 99 D5 98 FC C5 2E 8F 30 4B 94 D3 3E 1C 94 48 6A 66 2E
```

The above example with the 4 bogus key streams Xor'ed together into what is displayed below as the 'Effective Key Stream' is Xor'ed with the plaintext to produce the 'Internal Data Stream'. Notice the Plaintext ASCII hex and Internal Data Streams are identical in both the set above and this one and, most importantly, the Xor of the 4 streams above is identical to that on page 6:

```
Plaintext ASCII Hex  - 54 68 69 73 20 69 73 20 73 61 6D 70 6C 65 20 74 65 78 74 20 74 6F 20 64 65 6D 6F 6E 73 74
                       || || || || || || || || || || || || || || || || || || || || || || || || || || || || ||
Effective Key Stream - 91 60 E8 08 CB 1D 0C 7E EE E5 B4 71 BC A4 FF 68 E4 B5 70 50 19 09 C3 B0 E4 B0 62 DC B4 45
                       || || || || || || || || || || || || || || || || || || || || || || || || || || || || ||
Ciphertext Out Line  - C5 08 81 7B EB 74 7F 5E 9D 84 D9 01 D0 C1 DF 1C 81 CD 04 70 6D 66 E3 D4 81 DD 0D B2 C7 31

72 61 74 65 20 74 68 65 73 65 20 73 74 65 70 73 20 6F 66 20 74 68 65 20 27 52 61 6E 64 6F 6D 20 43 69 70 68 65
|| || || || || || || || || || || || || || || || || || || || || || || || || || || || || || || || || || || ||
FA B3 9A B2 4A 86 A4 AF 67 85 90 14 87 F4 2C B0 78 2F FE B0 17 9D 30 B9 9B 78 6A D2 6F A4 DE 97 95 9A CB 54 BE
|| || || || || || || || || || || || || || || || || || || || || || || || || || || || || || || || || || || ||
88 D2 EE D7 6A F2 CC CA 14 E0 B0 67 F3 91 5C C3 58 40 98 90 63 F5 55 99 BC 2A 0B BC 0B CB B3 B7 D6 F3 BB 3C DB

72 20 4F 75 74 70 75 74 73 27 20 6D 6F 64 65 20 75 73 69 6E 67 20 55 4E 41 54 54 41 43 4B 41 42 4C 45 20 41 6C
|| || || || || || || || || || || || || || || || || || || || || || || || || || || || || || || || || || || ||
C4 4C FB 59 72 6C E8 CB 86 40 58 B3 E8 C7 7B 27 47 10 C2 BD F1 44 CF F8 55 9A 58 6D A8 2A 94 15 26 F9 6F C1 FC
|| || || || || || || || || || || || || || || || || || || || || || || || || || || || || || || || || || || ||
B6 6C B4 2C 06 1C 9D BF F5 67 78 DE 87 A3 1E 07 32 63 AB D3 96 64 9A B6 14 CE 0C 2C EB 61 D5 57 6A BC 4F 80 90

67 65 62 72 61 69 63 20 6C 61 77 20 66 6F 72 20 73 65 63 75 72 69 74 79
|| || || || || || || || || || || || || || || || || || || || || || || ||
EB D5 49 7A AB 8D FA F5 F4 9D B2 0E E9 5F 39 B4 A0 5B 7F E1 3A 03 12 57
|| || || || || || || || || || || || || || || || || || || || || || || ||
8C B0 2B 08 CA E4 99 D5 98 FC C5 2E 8F 30 4B 94 D3 3E 1C 94 48 6A 66 2E
```

Appendix B

All the Effective Key Streams are copied to the bottom of this page for your comparison to see the changes resulting from very minor changes to just one of the first 3 pointers. The 4th pointer is calculated from the first three pointers, so since the first pointer changed, pointer 4 will change significantly.

```
Key Stream @2,557,721  - 09 B7 3C A5 E5 62 AF 11 48 82 86 59 EC 50 6F D3 97 2B D1 D0 79 63 FB 48 2A 49 F8 17 FE
Key Stream @1,620,751  - 19 4E A5 6C D6 0C 0B 2C 07 32 18 AB 51 7F 1A 3A 64 0C 90 FC 8F 2F 9A B9 8E 19 CA 9D 12
Key Stream @5,280,170  - DC F3 4F D5 13 2B 96 D9 32 83 92 07 4C AE D4 67 98 4D 66 E8 87 E2 9C C4 19 D7 2A A8 97
Key Stream @7,211,985  - 5D 6A 3E 14 EB 58 3E 9A 93 D6 B8 84 4D 25 5E E6 8F DF 57 94 68 A7 3E 85 59 37 7A FE CF
Effective Key Stream 1 - 91 60 E8 08 CB 1D 0C 7E EE E5 B4 71 BC A4 FF 68 E4 B5 70 50 19 09 C3 B0 E4 B0 62 DC B4

Key Stream @2,557,722  - B7 3C A5 E5 62 AF 11 48 82 86 59 EC 50 6F D3 97 2B D1 D0 79 63 FB 48 2A 49 F8 17 FE 82
Key Stream @1,620,751  - 19 4E A5 6C D6 0C 0B 2C 07 32 18 AB 51 7F 1A 3A 64 0C 90 FC 8F 2F 9A B9 8E 19 CA 9D 12
Key Stream @5,280,170  - DC F3 4F D5 13 2B 96 D9 32 83 92 07 4C AE D4 67 98 4D 66 E8 87 E2 9C C4 19 D7 2A A8 97
Key Stream @6,666,568  - 77 D1 4E 6D 1F 52 46 54 FC E7 99 50 E3 45 C2 34 B9 2A 7E 77 D7 F5 25 28 2C 0A 33 E9 73
Effective Key Stream 2 - 05 50 01 31 B8 DA CA E9 4B D0 4A 10 AE FB DF FE 6E BA 58 1A BC C3 6B 7F F2 3C C4 22 74

Key Stream @2,557,723  - 3C A5 E5 62 AF 11 48 82 86 59 EC 50 6F D3 97 2B D1 D0 79 63 FB 48 2A 49 F8 17 FE 82 99
Key Stream @1,620,751  - 19 4E A5 6C D6 0C 0B 2C 07 32 18 AB 51 7F 1A 3A 64 0C 90 FC 8F 2F 9A B9 8E 19 CA 9D 12
Key Stream @5,280,170  - DC F3 4F D5 13 2B 96 D9 32 83 92 07 4C AE D4 67 98 4D 66 E8 87 E2 9C C4 19 D7 2A A8 97
Key Stream @8,142,856  - BB 6C 49 AC EF A5 44 26 E2 05 CE 34 BA 81 91 CE F6 17 E3 8E 65 99 65 76 26 97 5F 1E C8
Effective Key Stream 3 - 42 74 46 77 85 93 91 51 51 ED A8 C8 C8 83 C8 B8 DB 86 6C F9 96 1C 49 42 49 4E 41 A9 D4

Key Stream @2,557,724  - A5 E5 62 AF 11 48 82 86 59 EC 50 6F D3 97 2B D1 D0 79 63 FB 48 2A 49 F8 17 FE 82 99 38
Key Stream @1,620,751  - 19 4E A5 6C D6 0C 0B 2C 07 32 18 AB 51 7F 1A 3A 64 0C 90 FC 8F 2F 9A B9 8E 19 CA 9D 12
Key Stream @5,280,170  - DC F3 4F D5 13 2B 96 D9 32 83 92 07 4C AE D4 67 98 4D 66 E8 87 E2 9C C4 19 D7 2A A8 97
Key Stream @8,169,359  - D9 26 C7 B6 80 23 E9 D8 8B A3 03 FE BA 70 87 33 27 CD 37 CA 8E AB 7C 99 DF 1C 4D 66 4C
Effective Key Stream 4 - B9 7E 4F A0 54 4C F6 AB E7 FE D9 3D 74 36 62 BF 0B F5 A2 25 CE 4C 33 1C 5F 2C 2F CA F1

Key Stream @2,557,725  - E5 62 AF 11 48 82 86 59 EC 50 6F D3 97 2B D1 D0 79 63 FB 48 2A 49 F8 17 FE 82 99 38 F7
Key Stream @1,620,751  - 19 4E A5 6C D6 0C 0B 2C 07 32 18 AB 51 7F 1A 3A 64 0C 90 FC 8F 2F 9A B9 8E 19 CA 9D 12
Key Stream @5,280,170  - DC F3 4F D5 13 2B 96 D9 32 83 92 07 4C AE D4 67 98 4D 66 E8 87 E2 9C C4 19 D7 2A A8 97
Key Stream @8,120,130  - EB 04 EB B3 75 A0 A9 35 26 AB FA 03 90 B1 BE AB EA BB 5A 67 80 BC FF C4 E3 12 CC 13 EF
Effective Key Stream 5 - CB DB AE 1B F8 05 B2 99 FF 4A 1F 7C 1A 4B A1 26 6F 99 57 3B A2 38 01 AE 8A 5E B5 1E 9D

Key Stream @2,557,726  - 62 AF 11 48 82 86 59 EC 50 6F D3 97 2B D1 D0 79 63 FB 48 2A 49 F8 17 FE 82 99 38 F7 35
Key Stream @1,620,751  - 19 4E A5 6C D6 0C 0B 2C 07 32 18 AB 51 7F 1A 3A 64 0C 90 FC 8F 2F 9A B9 8E 19 CA 9D 12
Key Stream @5,280,170  - DC F3 4F D5 13 2B 96 D9 32 83 92 07 4C AE D4 67 98 4D 66 E8 87 E2 9C C4 19 D7 2A A8 97
Key Stream @7,759,612  - 6A 99 DD 24 27 53 33 63 93 E6 D4 C0 34 93 E2 4F 3C 81 B1 A4 BE B9 51 E7 60 F5 E2 37 75
Effective Key Stream 6 - CD 8B 26 D5 60 F2 F7 7A F6 38 8D FB 02 93 FC 6B A3 3B 0F 9A FF 8C 40 64 75 A2 3A F5 C5

Key Stream @2,557,727  - AF 11 48 82 86 59 EC 50 6F D3 97 2B D1 D0 79 63 FB 48 2A 49 F8 17 FE 82 99 38 F7 35 A5
Key Stream @1,620,751  - 19 4E A5 6C D6 0C 0B 2C 07 32 18 AB 51 7F 1A 3A 64 0C 90 FC 8F 2F 9A B9 8E 19 CA 9D 12
Key Stream @5,280,170  - DC F3 4F D5 13 2B 96 D9 32 83 92 07 4C AE D4 67 98 4D 66 E8 87 E2 9C C4 19 D7 2A A8 97
Key Stream @6,861,221  - FE 49 65 9E 39 29 2F A4 52 E2 65 33 62 E4 99 DD 2E 69 60 AF 85 DE 5F FA 13 C0 F2 94 A6
Effective Key Stream 7 - 94 E5 C7 A5 7A 57 5E 01 08 80 78 B4 AE E5 2E E3 29 60 BC F2 75 04 A7 05 1D 36 E5 94 86

Key Stream @2,557,728  - 11 48 82 86 59 EC 50 6F D3 97 2B D1 D0 79 63 FB 48 2A 49 F8 17 FE 82 99 38 F7 35 A5 9D
Key Stream @1,620,751  - 19 4E A5 6C D6 0C 0B 2C 07 32 18 AB 51 7F 1A 3A 64 0C 90 FC 8F 2F 9A B9 8E 19 CA 9D 12
Key Stream @5,280,170  - DC F3 4F D5 13 2B 96 D9 32 83 92 07 4C AE D4 67 98 4D 66 E8 87 E2 9C C4 19 D7 2A A8 97
Key Stream @7,410,543  - B2 72 1E DA 6B 5D 1D 24 67 89 CB 14 ED B9 F2 46 E5 61 4D D9 7F 5B 25 37 71 23 DC 71 92
Effective Key Stream 8 - 66 87 76 E5 F7 96 D0 BE 81 AF 6A 69 20 11 5F E0 51 0A F2 35 60 68 A1 D3 DE 1A 09 E1 8A

Key Stream @2,557,729  - 48 82 86 59 EC 50 6F D3 97 2B D1 D0 79 63 FB 48 2A 49 F8 17 FE 82 99 38 F7 35 A5 9D 12
Key Stream @1,620,751  - 19 4E A5 6C D6 0C 0B 2C 07 32 18 AB 51 7F 1A 3A 64 0C 90 FC 8F 2F 9A B9 8E 19 CA 9D 12
Key Stream @5,280,170  - DC F3 4F D5 13 2B 96 D9 32 83 92 07 4C AE D4 67 98 4D 66 E8 87 E2 9C C4 19 D7 2A A8 97
Key Stream @8,080,491  - 80 62 89 DB C1 1E 00 E0 7F 11 3C 33 5C 81 83 62 76 4F 34 0F 5E 80 7B C2 68 02 40 1C D2
Effective Key Stream 9 - 0D 5D E5 3B E8 69 F2 C6 DD 8B 67 4F 38 33 B6 77 A0 47 3A 0C A8 CF E4 87 08 F9 05 B4 45

Key Stream @2,557,730  - 82 86 59 EC 50 6F D3 97 2B D1 D0 79 63 FB 48 2A 49 F8 17 FE 82 99 38 F7 35 A5 9D 12 57
Key Stream @1,620,751  - 19 4E A5 6C D6 0C 0B 2C 07 32 18 AB 51 7F 1A 3A 64 0C 90 FC 8F 2F 9A B9 8E 19 CA 9D 12
Key Stream @5,280,170  - DC F3 4F D5 13 2B 96 D9 32 83 92 07 4C AE D4 67 98 4D 66 E8 87 E2 9C C4 19 D7 2A A8 97
Key Stream @8,357,556  - B5 E3 23 14 08 04 67 54 8C D5 8B 27 68 19 A3 AD D3 71 39 25 C3 E9 07 44 54 EE 59 76 F1
Effective Key Stream 10- F2 D8 90 41 9D 4C 29 36 92 B5 D1 F2 16 33 25 DA 66 C8 D8 CF 49 BD 39 CE F6 85 24 51 23


Effective Key Stream 1 - 91 60 E8 08 CB 1D 0C 7E EE E5 B4 71 BC A4 FF 68 E4 B5 70 50 19 09 C3 B0 E4 B0 62 DC B4
Effective Key Stream 2 - 05 50 01 31 B8 DA CA E9 4B D0 4A 10 AE FB DF FE 6E BA 58 1A BC C3 6B 7F F2 3C C4 22 74
Effective Key Stream 3 - 42 74 46 77 85 93 91 51 51 ED A8 C8 C8 83 C8 B8 DB 86 6C F9 96 1C 49 42 49 4E 41 A9 D4
Effective Key Stream 4 - B9 7E 4F A0 54 4C F6 AB E7 FE D9 3D 74 36 62 BF 0B F5 A2 25 CE 4C 33 1C 5F 2C 2F CA F1
Effective Key Stream 5 - CB DB AE 1B F8 05 B2 99 FF 4A 1F 7C 1A 4B A1 26 6F 99 57 3B A2 38 01 AE 8A 5E B5 1E 9D
Effective Key Stream 6 - CD 8B 26 D5 60 F2 F7 7A F6 38 8D FB 02 93 FC 6B A3 3B 0F 9A FF 8C 40 64 75 A2 3A F5 C5
Effective Key Stream 7 - 94 E5 C7 A5 7A 57 5E 01 08 80 78 B4 AE E5 2E E3 29 60 BC F2 75 04 A7 05 1D 36 E5 94 86
Effective Key Stream 8 - 66 87 76 E5 F7 96 D0 BE 81 AF 6A 69 20 11 5F E0 51 0A F2 35 60 68 A1 D3 DE 1A 09 E1 8A
Effective Key Stream 9 - 0D 5D E5 3B E8 69 F2 C6 DD 8B 67 4F 38 33 B6 77 A0 47 3A 0C A8 CF E4 87 08 F9 05 B4 45
Effective Key Stream 10- F2 D8 90 41 9D 4C 29 36 92 B5 D1 F2 16 33 25 DA 66 C8 D8 CF 49 BD 39 CE F6 85 24 51 23
```

Appendix C

These pseudo-random key table-based functions are called to advance all 4 Vernam Two pointers:

```
Function sete1p4() As Long
    Dim n As Byte, p1 As Long, p2 As Long, p3 As Long, byte0 As Long, byte1 As Long, byte2 As Long

    ' Get the particular base number to use from the ciphertext block number, then form the base into the respective blocks
    n = ((ctxBlockNum - 1) Mod 24) + 1: p1 = ptrBase(1, n) + e1p1: p2 = ptrBase(2, n) + e1p2: p3 = ptrBase(3, n) + e1p3
    ' Form the byte pointers 1=467719, 2=1601534, 3=988317, 4=541927 861290
    Select Case (ctxBlockNum Mod 6)
        Case 0: byte0 = v2Key(p1) Xor v2Key(p2 + 1) Xor v2Key(p3 + 2)
                byte1 = v2Key(p1 + 1) Xor v2Key(p2 + 2) Xor v2Key(p3)
                byte2 = v2Key(p1 + 2) Xor v2Key(p2) Xor v2Key(p3 + 1)
        Case 1: byte0 = v2Key(p1) Xor v2Key(p2 + 1) Xor v2Key(p3 + 2) '
                byte2 = v2Key(p1 + 1) Xor v2Key(p2 + 2) Xor v2Key(p3)
                byte1 = v2Key(p1 + 2) Xor v2Key(p2) Xor v2Key(p3 + 1)
        Case 2: byte1 = v2Key(p1) Xor v2Key(p2 + 1) Xor v2Key(p3 + 2)
                byte0 = v2Key(p1 + 1) Xor v2Key(p2 + 2) Xor v2Key(p3)
                byte2 = v2Key(p1 + 2) Xor v2Key(p2) Xor v2Key(p3 + 1)
        Case 3: byte1 = v2Key(p1) Xor v2Key(p2 + 1) Xor v2Key(p3 + 2)
                byte2 = v2Key(p1 + 1) Xor v2Key(p2 + 2) Xor v2Key(p3)
                byte0 = v2Key(p1 + 2) Xor v2Key(p2) Xor v2Key(p3 + 1)
        Case 4: byte2 = v2Key(p1) Xor v2Key(p2 + 1) Xor v2Key(p3 + 2)
                byte0 = v2Key(p1 + 1) Xor v2Key(p2 + 2) Xor v2Key(p3)
                byte1 = v2Key(p1 + 2) Xor v2Key(p2) Xor v2Key(p3 + 1)
        Case 5: byte2 = v2Key(p1) Xor v2Key(p2 + 1) Xor v2Key(p3 + 2)
                byte1 = v2Key(p1 + 1) Xor v2Key(p2 + 2) Xor v2Key(p3)
                byte0 = v2Key(p1 + 2) Xor v2Key(p2) Xor v2Key(p3 + 1)
    End Select
    ' Form the pointer, Xor'ing it with the block number
    sete1p4 = (((byte2 * &H10000) + (byte1 * &H100&) + byte0) Xor ctxBlockNum) And &H1FFFFF
End Function
```

```
Sub advancePointers()
    Dim byte0P1 As Long, byte1P1 As Long, byte2P1 As Long, byte0P2 As Long, byte1P2 As Long, _
        byte2P2 As Long, byte0P3 As Long, byte1P3 As Long, byte2P3 As Long, n As Byte, _
        p1 As Long, p2 As Long, p3 As Long, origP1 As Long, origP2 As Long, origP3 As Long, _
        pBlockNum As Long

    ' Save the current values
    origP1 = e1p1: origP2 = e1p2: origP3 = e1p3
    ' Get the pointer to the base value
    n = ((ctxBlockNum - 1) Mod 24) + 1
    p1 = ptrBase(1, n) + e1p1: p2 = ptrBase(2, n) + e1p2: p3 = ptrBase(3, n) + e1p3
    ' Get the block number into our variable for possible modification
    pBlockNum = ctxBlockNum
    Do
        ' Select how the new pointers will be formed
        Select Case (pBlockNum Mod 6)
            Case 0: byte0P1 = v2Key(p1): byte1P1 = v2Key(p2): byte2P1 = v2Key(p3)
                    byte0P2 = v2Key(p2): byte1P2 = v2Key(p3): byte2P2 = v2Key(p1)
                    byte0P3 = v2Key(p3): byte1P3 = v2Key(p1): byte2P3 = v2Key(p2)
            Case 1: byte0P1 = v2Key(p1): byte1P1 = v2Key(p3): byte2P1 = v2Key(p2)
                    byte0P2 = v2Key(p3): byte1P2 = v2Key(p2): byte2P2 = v2Key(p1)
                    byte0P3 = v2Key(p2): byte1P3 = v2Key(p1): byte2P3 = v2Key(p3)
            Case 2: byte0P1 = v2Key(p2): byte1P1 = v2Key(p1): byte2P1 = v2Key(p3)
                    byte0P2 = v2Key(p1): byte1P2 = v2Key(p3): byte2P2 = v2Key(p2)
                    byte0P3 = v2Key(p3): byte1P3 = v2Key(p2): byte2P3 = v2Key(p1)
            Case 3: byte0P1 = v2Key(p2): byte1P1 = v2Key(p3): byte2P1 = v2Key(p1)
                    byte0P2 = v2Key(p3): byte1P2 = v2Key(p1): byte2P2 = v2Key(p2)
                    byte0P3 = v2Key(p1): byte1P3 = v2Key(p2): byte2P3 = v2Key(p3)
            Case 4: byte0P1 = v2Key(p3): byte1P1 = v2Key(p1): byte2P1 = v2Key(p2)
                    byte0P2 = v2Key(p1): byte1P2 = v2Key(p2): byte2P2 = v2Key(p3)
                    byte0P3 = v2Key(p2): byte1P3 = v2Key(p3): byte2P3 = v2Key(p1)
            Case 5: byte0P1 = v2Key(p3): byte1P1 = v2Key(p2): byte2P1 = v2Key(p1)
                    byte0P2 = v2Key(p2): byte1P2 = v2Key(p1): byte2P2 = v2Key(p3)
                    byte0P3 = v2Key(p1): byte1P3 = v2Key(p3): byte2P3 = v2Key(p2)
        End Select
        ' Form the pointers
        e1p1 = ((byte2P1 * &H10000) + (byte1P1 * &H100&) + byte0P1) And &H1FFFFF
        e1p2 = ((byte2P2 * &H10000) + (byte1P2 * &H100&) + byte0P2) And &H1FFFFF
        e1p3 = ((byte2P3 * &H10000) + (byte1P3 * &H100&) + byte0P3) And &H1FFFFF
        ' Form the 4th pointer - the VERY LAST setup to perform
        e1p4 = sete1p4
        ' Increment the private block number
        pBlockNum = pBlockNum + 1
    ' Loop if at least one pointer did not change
    Loop While e1p1 = origP1 Or e1p2 = origP2 Or e1p3 = origP3
End Sub
```

Appendix D: The effectiveness of the 'AdvancePointers' function

Not only are the pointers changed in a key table-based pseudo-random fashion, changing their position instead of their values will result in different sequences. Notice in these examples that 5 values are used, set to different pointers for the 6 possible arrangements. The Engine 2 pointers are held constant in all 6 scenarios. Pointer 4 and the two Engine 2 pointers rely on the order of the first 3 pointers for their incremental values, as shown here. Notice that block 1 illustrates the same starting values as the first scenario but in a different order. For example, notice that scenarios 1 and 2 hold the same starting values for pointer 1. But the next block's pointer 1 contains 1,976,443 and 1,124,470 respectively. The only items that changed were the order of pointers 2 and 3, not their value:

```
Initial pointers Block #1: P1 = 1,619,019, P2 = 663,180, P3 = 961,408, P4 = 1,835,494, E2P = 253, E2O = 92

Scenario #1: Block #1: P1 = 1,619,019, P2 = 663,180, P3 = 961,408, P4 = 1,835,494, E2P = 253, E2O = 92
             Block #2: P1 = 1,976,443, P2 = 1,793,576, P3 = 555,870, P4 = 1,390,795, E2P = 3,947, E2O = 294
             Block #3: P1 = 1,196,355, P2 = 225,857, P3 = 82,802, P4 = 1,154,972, E2P = 4,176, E2O = 165
             Block #4: P1 = 1,745,703, P2 = 506,531, P3 = 206,778, P4 = 492,789, E2P = 2,610, E2O = 204
             Block #5: P1 = 1,262,074, P2 = 1,717,057, P3 = 129,587, P4 = 646,007, E2P = 4,254, E2O = 71
             Block #6: P1 = 507,547, P2 = 1,771,454, P3 = 2,005,767, P4 = 1,991,660, E2P = 7,668, E2O = 71
             Block #7: P1 = 835,719, P2 = 486,592, P3 = 34,668, P4 = 517,206, E2P = 6,529, E2O = 199
             Block #8: P1 = 723,795, P2 = 1,248,011, P3 = 742,155, P4 = 1,873,764, E2P = 1,110, E2O = 98

Scenario #2: Block #1: P1 = 1,619,019, P2 = 961,408, P3 = 663,180, P4 = 1,873,764, E2P = 253, E2O = 92
             Block #2: P1 = 1,124,470, P2 = 1,495,336, P3 = 554,705, P4 = 1,010,917, E2P = 4,701, E2O = 246
             Block #3: P1 = 1,215,085, P2 = 856,714, P3 = 683,282, P4 = 1,405,354, E2P = 1,300, E2O = 282
             Block #4: P1 = 393,364, P2 = 1,361,408, P3 = 38,086, P4 = 1,903,192, E2P = 2,380, E2O = 160
             Block #5: P1 = 353,126, P2 = 435,555, P3 = 222,885, P4 = 297,704, E2P = 2,843, E2O = 146
             Block #6: P1 = 411,767, P2 = 1,549,896, P3 = 554,918, P4 = 1,874,194, E2P = 2,325, E2O = 217
             Block #7: P1 = 818,786, P2 = 183,422, P3 = 1,991,372, P4 = 1,552,946, E2P = 2,449, E2O = 164
             Block #8: P1 = 999,268, P2 = 266,047, P3 = 2,057,231, P4 = 457,019, E2P = 5,702, E2O = 209

Scenario #3: Block #1: P1 = 663,180, P2 = 1,619,019, P3 = 961,408, P4 = 457,019, E2P = 253, E2O = 92
             Block #2: P1 = 2,011,798, P2 = 1,466,034, P3 = 1,218,142, P4 = 1,579,229, E2P = 2,857, E2O = 212
             Block #3: P1 = 843,489, P2 = 126,174, P3 = 2,023,916, P4 = 975,197, E2P = 4,145, E2O = 310
             Block #4: P1 = 1,656,580, P2 = 276,807, P3 = 459,833, P4 = 1,382,808, E2P = 2,272, E2O = 176
             Block #5: P1 = 1,921,847, P2 = 1,523,027, P3 = 1,259,325, P4 = 995,023, E2P = 5,427, E2O = 217
             Block #6: P1 = 731,774, P2 = 2,026,282, P3 = 687,851, P4 = 2,015,732, E2P = 2,030, E2O = 231
             Block #7: P1 = 915,264, P2 = 3,575, P3 = 1,523,725, P4 = 926,012, E2P = 3,712, E2O = 314
             Block #8: P1 = 839,472, P2 = 1,101,007, P3 = 995,532, P4 = 26,339, E2P = 7,234, E2O = 268

Scenario #4: Block #1: P1 = 663,180, P2 = 961,408, P3 = 1,619,019, P4 = 26,339, E2P = 253, E2O = 92
             Block #2: P1 = 1,094,262, P2 = 1,495,218, P3 = 1,210,064, P4 = 1,493,389, E2P = 2,855, E2O = 282
             Block #3: P1 = 1,632,359, P2 = 522,472, P3 = 550,904, P4 = 810,636, E2P = 4,081, E2O = 127
             Block #4: P1 = 1,443,201, P2 = 87,557, P3 = 360,790, P4 = 1,233,220, E2P = 88, E2O = 227
             Block #5: P1 = 1,643,598, P2 = 956,692, P3 = 1,330,841, P4 = 1,765,597, E2P = 5,201, E2O = 202
             Block #6: P1 = 1,457,950, P2 = 1,996,351, P3 = 2,039,414, P4 = 745,847, E2P = 8,079, E2O = 263
             Block #7: P1 = 267,969, P2 = 115,734, P3 = 1,491,396, P4 = 1,353,211, E2P = 3,459, E2O = 120
             Block #8: P1 = 710,505, P2 = 600,791, P3 = 1,534,250, P4 = 435,246, E2P = 4,759, E2O = 213

Scenario #5: Block #1: P1 = 961,408, P2 = 1,619,019, P3 = 663,180, P4 = 435,246, E2P = 253, E2O = 92
             Block #2: P1 = 1,116,310, P2 = 1,495,304, P3 = 562,897, P4 = 1,216,357, E2P = 6,800, E2O = 136
             Block #3: P1 = 508,973, P2 = 853,956, P3 = 273,671, P4 = 1,165,000, E2P = 994, E2O = 75
             Block #4: P1 = 1,084,863, P2 = 2,044,045, P3 = 900,912, P4 = 1,661,515, E2P = 4,044, E2O = 264
             Block #5: P1 = 648,807, P2 = 502,246, P3 = 419,753, P4 = 1,430,904, E2P = 4,915, E2O = 217
             Block #6: P1 = 1,504,466, P2 = 1,234,676, P3 = 1,364,694, P4 = 1,756,084, E2P = 7,123, E2O = 155
             Block #7: P1 = 503,505, P2 = 1,140,654, P3 = 971,111, P4 = 1,835,448, E2P = 5,527, E2O = 108
             Block #8: P1 = 1,903,437, P2 = 892,171, P3 = 740,765, P4 = 1,279,295, E2P = 5,165, E2O = 154

Scenario #6: Block #1: P1 = 961,408, P2 = 663,180, P3 = 1,619,019, P4 = 1,279,295, E2P = 253, E2O = 92
             Block #2: P1 = 1,050,747, P2 = 1,822,728, P3 = 555,984, P4 = 1,039,779, E2P = 3,962, E2O = 224
             Block #3: P1 = 1,974,635, P2 = 728,609, P3 = 92,958, P4 = 1,507,839, E2P = 1,928, E2O = 301
             Block #4: P1 = 880,775, P2 = 494,960, P3 = 1,083,277, P4 = 436,727, E2P = 271, E2O = 291
             Block #5: P1 = 81,913, P2 = 1,646,911, P3 = 2,095,393, P4 = 21,057, E2P = 1,279, E2O = 292
             Block #6: P1 = 1,991,920, P2 = 1,064,548, P3 = 323,646, P4 = 1,579,756, E2P = 3,843, E2O = 228
             Block #7: P1 = 344,765, P2 = 1,926,466, P3 = 179,557, P4 = 1,393,223, E2P = 4,149, E2O = 225
             Block #8: P1 = 1,350,545, P2 = 1,168,539, P3 = 1,806,804, P4 = 1,347,639, E2P = 5,414, E2O = 122
```

The effectiveness of the ‘AdvancePointers’ function:

In several tests (153, 1 hr each) simulating the encryption of 2 billion blocks of plaintext (256 Gigabytes) in each test, it was learned that all values between 0 and 2,097,151 inclusive were selected for each of the first 3 pointers in usually less than 20 million blocks. The fourth pointer, due to its different initialization, takes longer to use all of its values:

E1P1 = 1,002,614, E1P2 = 1,521,525, E1P3 = 863,223, E2P = 193, E2O = 117:
Block count where all addresses used for pointer #1 - 15,183,700
Block count where all addresses used for pointer #2 - 15,406,500
Block count where all addresses used for pointer #3 - 16,555,000
Block count where all addresses used for pointer #4 - 35,122,700

E1P1 = 1,004,373, E1P2 = 786,647, E1P3 = 538,067, E2P = 150, E2O = 153:
Block count where all addresses used for pointer #1 - 17,282,500
Block count where all addresses used for pointer #2 - 16,396,400
Block count where all addresses used for pointer #3 - 15,649,600
Block count where all addresses used for pointer #4 - 32,882,200

E1P1 = 1,014,903, E1P2 = 1,350,259, E1P3 = 469,502, E2P = 125, E2O = 26:
Block count where all addresses used for pointer #1 - 16,255,800
Block count where all addresses used for pointer #2 - 17,077,400
Block count where all addresses used for pointer #3 - 16,002,100
Block count where all addresses used for pointer #4 - 31,966,500

E1P1 = 1,052,942, E1P2 = 234,539, E1P3 = 398,933, E2P = 171, E2O = 199:
Block count where all addresses used for pointer #1 - 16,012,200
Block count where all addresses used for pointer #2 - 16,548,500
Block count where all addresses used for pointer #3 - 18,478,200
Block count where all addresses used for pointer #4 - 29,425,100

E1P1 = 1,068,442, E1P2 = 1,448,842, E1P3 = 575,802, E2P = 19, E2O = 50:
Block count where all addresses used for pointer #1 - 17,401,100
Block count where all addresses used for pointer #2 - 16,229,800
Block count where all addresses used for pointer #3 - 17,989,200
Block count where all addresses used for pointer #4 - 31,781,600

E1P1 = 1,069,639, E1P2 = 874,755, E1P3 = 732,750, E2P = 199, E2O = 63:
Block count where all addresses used for pointer #1 - 17,187,100
Block count where all addresses used for pointer #2 - 18,344,900
Block count where all addresses used for pointer #3 - 16,260,500
Block count where all addresses used for pointer #4 - 33,311,100

E1P1 = 1,069,869, E1P2 = 1,466,703, E1P3 = 62,571, E2P = 8, E2O = 253:
Block count where all addresses used for pointer #1 - 16,377,600
Block count where all addresses used for pointer #2 - 16,461,100
Block count where all addresses used for pointer #3 - 15,850,200
Block count where all addresses used for pointer #4 - 35,395,500

E1P1 = 1,092,940, E1P2 = 628,083, E1P3 = 1,347,710, E2P = 4, E2O = 96:
Block count where all addresses used for pointer #1 - 15,000,100
Block count where all addresses used for pointer #2 - 15,567,400
Block count where all addresses used for pointer #3 - 15,562,800
Block count where all addresses used for pointer #4 - 35,864,000

E1P1 = 1,098,469, E1P2 = 1,745,191, E1P3 = 2,066,400, E2P = 122, E2O = 149:
Block count where all addresses used for pointer #1 - 16,256,200
Block count where all addresses used for pointer #2 - 15,000,100
Block count where all addresses used for pointer #3 - 18,781,300
Block count where all addresses used for pointer #4 - 28,592,900

E1P1 = 1,105,622, E1P2 = 1,243,604, E1P3 = 1,996,890, E2P = 36, E2O = 245:
Block count where all addresses used for pointer #1 - 17,551,100
Block count where all addresses used for pointer #2 - 17,494,300
Block count where all addresses used for pointer #3 - 15,809,900
Block count where all addresses used for pointer #4 - 31,304,200

E1P1 = 1,120,928, E1P2 = 1,414,902, E1P3 = 962,422, E2P = 45, E2O = 189:
Block count where all addresses used for pointer #1 - 18,335,800
Block count where all addresses used for pointer #2 - 17,641,300
Block count where all addresses used for pointer #3 - 16,052,400
Block count where all addresses used for pointer #4 - 30,695,400

E1P1 = 1,144,045, E1P2 = 631,950, E1P3 = 340,268, E2P = 161, E2O = 193:
Block count where all addresses used for pointer #1 - 16,548,300
Block count where all addresses used for pointer #2 - 18,098,300
Block count where all addresses used for pointer #3 - 17,333,100
Block count where all addresses used for pointer #4 - 31,200,300

E1P1 = 1,151,087, E1P2 = 1,326,731, E1P3 = 1,482,166, E2P = 178, E2O = 137:
Block count where all addresses used for pointer #1 - 18,091,600
Block count where all addresses used for pointer #2 - 16,918,500
Block count where all addresses used for pointer #3 - 18,009,700
Block count where all addresses used for pointer #4 - 30,588,700

E1P1 = 1,153,597, E1P2 = 160,926, E1P3 = 983,292, E2P = 208, E2O = 232:
Block count where all addresses used for pointer #1 - 18,094,400
Block count where all addresses used for pointer #2 - 15,591,200
Block count where all addresses used for pointer #3 - 18,916,000
Block count where all addresses used for pointer #4 - 39,986,400

E1P1 = 1,153,970, E1P2 = 561,474, E1P3 = 935,954, E2P = 218, E2O = 252:
Block count where all addresses used for pointer #1 - 17,645,700
Block count where all addresses used for pointer #2 - 15,000,100
Block count where all addresses used for pointer #3 - 15,725,900
Block count where all addresses used for pointer #4 - 31,077,400

E1P1 = 1,156,289, E1P2 = 203,412, E1P3 = 48,794, E2P = 62, E2O = 136:
Block count where all addresses used for pointer #1 - 19,871,400
Block count where all addresses used for pointer #2 - 16,281,000
Block count where all addresses used for pointer #3 - 17,821,200
Block count where all addresses used for pointer #4 - 33,951,000

E1P1 = 1,165,673, E1P2 = 523,165, E1P3 = 1,977,984, E2P = 185, E2O = 58:
Block count where all addresses used for pointer #1 - 15,395,300
Block count where all addresses used for pointer #2 - 15,000,100
Block count where all addresses used for pointer #3 - 17,367,500
Block count where all addresses used for pointer #4 - 35,185,000

E1P1 = 1,178,089, E1P2 = 527,387, E1P3 = 268,036, E2P = 217, E2O = 18:
Block count where all addresses used for pointer #1 - 15,161,000
Block count where all addresses used for pointer #2 - 17,985,300
Block count where all addresses used for pointer #3 - 18,553,300
Block count where all addresses used for pointer #4 - 32,805,700

E1P1 = 1,183,525, E1P2 = 1,645,161, E1P3 = 1,376,796, E2P = 161, E2O = 96:
Block count where all addresses used for pointer #1 - 15,515,100
Block count where all addresses used for pointer #2 - 18,017,900
Block count where all addresses used for pointer #3 - 18,367,200
Block count where all addresses used for pointer #4 - 30,396,800

E1P1 = 1,184,888, E1P2 = 593,903, E1P3 = 725,258, E2P = 241, E2O = 123:
Block count where all addresses used for pointer #1 - 22,501,000
Block count where all addresses used for pointer #2 - 17,178,700
Block count where all addresses used for pointer #3 - 16,918,300
Block count where all addresses used for pointer #4 - 29,613,600

E1P1 = 1,185,530, E1P2 = 888,937, E1P3 = 1,046,428, E2P = 194, E2O = 55:
Block count where all addresses used for pointer #1 - 16,518,700
Block count where all addresses used for pointer #2 - 16,377,600
Block count where all addresses used for pointer #3 - 17,711,200
Block count where all addresses used for pointer #4 - 30,585,000

E1P1 = 1,186,075, E1P2 = 1,744,645, E1P3 = 1,412,295, E2P = 42, E2O = 213:
Block count where all addresses used for pointer #1 - 16,475,300
Block count where all addresses used for pointer #2 - 15,000,100
Block count where all addresses used for pointer #3 - 16,010,600
Block count where all addresses used for pointer #4 - 34,434,400

E1P1 = 1,194,496, E1P2 = 550,870, E1P3 = 1,028,051, E2P = 19, E2O = 246:
Block count where all addresses used for pointer #1 - 15,250,100
Block count where all addresses used for pointer #2 - 17,344,200
Block count where all addresses used for pointer #3 - 16,365,800
Block count where all addresses used for pointer #4 - 30,295,100

E1P1 = 1,220,857, E1P2 = 1,441,643, E1P3 = 820,117, E2P = 40, E2O = 104:
Block count where all addresses used for pointer #1 - 16,280,700
Block count where all addresses used for pointer #2 - 19,139,500
Block count where all addresses used for pointer #3 - 16,215,700
Block count where all addresses used for pointer #4 - 34,426,100

E1P1 = 1,268,072, E1P2 = 372,768, E1P3 = 1,041,272, E2P = 93, E2O = 123:
Block count where all addresses used for pointer #1 - 19,116,200
Block count where all addresses used for pointer #2 - 17,554,100
Block count where all addresses used for pointer #3 - 17,075,000
Block count where all addresses used for pointer #4 - 36,694,100

E1P1 = 1,275,554, E1P2 = 1,753,202, E1P3 = 1,079,938, E2P = 84, E2O = 134:
Block count where all addresses used for pointer #1 - 16,698,400
Block count where all addresses used for pointer #2 - 17,213,900
Block count where all addresses used for pointer #3 - 16,623,900
Block count where all addresses used for pointer #4 - 30,195,800

E1P1 = 1,276,759, E1P2 = 1,797,842, E1P3 = 1,980,896, E2P = 223, E2O = 118:
Block count where all addresses used for pointer #1 - 15,656,000
Block count where all addresses used for pointer #2 - 17,860,300
Block count where all addresses used for pointer #3 - 16,275,700
Block count where all addresses used for pointer #4 - 31,660,000

E1P1 = 1,290,346, E1P2 = 1,715,352, E1P3 = 1,567,117, E2P = 136, E2O = 226:
Block count where all addresses used for pointer #1 - 17,619,400
Block count where all addresses used for pointer #2 - 18,156,400
Block count where all addresses used for pointer #3 - 17,799,200
Block count where all addresses used for pointer #4 - 33,109,500

E1P1 = 1,290,574, E1P2 = 680,556, E1P3 = 1,613,458, E2P = 167, E2O = 134:
Block count where all addresses used for pointer #1 - 18,401,200
Block count where all addresses used for pointer #2 - 17,252,900
Block count where all addresses used for pointer #3 - 16,447,700
Block count where all addresses used for pointer #4 - 29,178,900

E1P1 = 1,297,988, E1P2 = 1,663,882, E1P3 = 1,938,232, E2P = 129, E2O = 6:
Block count where all addresses used for pointer #1 - 16,269,300
Block count where all addresses used for pointer #2 - 18,204,500
Block count where all addresses used for pointer #3 - 16,839,500
Block count where all addresses used for pointer #4 - 32,801,200

E1P1 = 1,299,900, E1P2 = 930,723, E1P3 = 1,732,461, E2P = 205, E2O = 111:
Block count where all addresses used for pointer #1 - 15,164,000
Block count where all addresses used for pointer #2 - 16,647,800
Block count where all addresses used for pointer #3 - 18,027,900
Block count where all addresses used for pointer #4 - 38,955,800

E1P1 = 1,310,684, E1P2 = 1,890,114, E1P3 = 597,650, E2P = 47, E2O = 71:
Block count where all addresses used for pointer #1 - 17,849,600
Block count where all addresses used for pointer #2 - 16,631,600
Block count where all addresses used for pointer #3 - 15,145,100
Block count where all addresses used for pointer #4 - 31,426,500

E1P1 = 1,321,525, E1P2 = 1,113,143, E1P3 = 2,016,179, E2P = 245, E2O = 9:
Block count where all addresses used for pointer #1 - 15,299,000
Block count where all addresses used for pointer #2 - 15,000,100
Block count where all addresses used for pointer #3 - 18,101,100
Block count where all addresses used for pointer #4 - 34,507,800

E1P1 = 1,343,768, E1P2 = 1,853,967, E1P3 = 123,562, E2P = 65, E2O = 46:
Block count where all addresses used for pointer #1 - 16,901,100
Block count where all addresses used for pointer #2 - 17,238,900
Block count where all addresses used for pointer #3 - 18,358,000
Block count where all addresses used for pointer #4 - 29,110,200

E1P1 = 1,345,465, E1P2 = 1,878,317, E1P3 = 748,625, E2P = 61, E2O = 251:
Block count where all addresses used for pointer #1 - 16,306,000
Block count where all addresses used for pointer #2 - 15,742,500
Block count where all addresses used for pointer #3 - 15,679,600
Block count where all addresses used for pointer #4 - 33,455,900

E1P1 = 1,375,031, E1P2 = 923,570, E1P3 = 181,697, E2P = 149, E2O = 121:
Block count where all addresses used for pointer #1 - 15,834,200
Block count where all addresses used for pointer #2 - 15,993,800
Block count where all addresses used for pointer #3 - 17,114,800
Block count where all addresses used for pointer #4 - 31,348,400

E1P1 = 1,427,447, E1P2 = 918,129, E1P3 = 4,739, E2P = 189, E2O = 110:
Block count where all addresses used for pointer #1 - 15,248,600
Block count where all addresses used for pointer #2 - 16,310,400
Block count where all addresses used for pointer #3 - 16,537,900
Block count where all addresses used for pointer #4 - 31,624,700

E1P1 = 1,427,450, E1P2 = 335,719, E1P3 = 615,585, E2P = 157, E2O = 44:
Block count where all addresses used for pointer #1 - 16,755,600
Block count where all addresses used for pointer #2 - 16,893,800
Block count where all addresses used for pointer #3 - 15,026,700
Block count where all addresses used for pointer #4 - 29,835,300

E1P1 = 1,453,761, E1P2 = 1,887,635, E1P3 = 111,260, E2P = 117, E2O = 20:
Block count where all addresses used for pointer #1 - 18,531,000
Block count where all addresses used for pointer #2 - 15,730,600
Block count where all addresses used for pointer #3 - 15,959,000
Block count where all addresses used for pointer #4 - 35,746,600

E1P1 = 1,466,692, E1P2 = 2,011,403, E1P3 = 38,709, E2P = 3, E2O = 116:
Block count where all addresses used for pointer #1 - 15,228,300
Block count where all addresses used for pointer #2 - 17,562,500
Block count where all addresses used for pointer #3 - 16,308,000
Block count where all addresses used for pointer #4 - 30,973,700

E1P1 = 1,473,218, E1P2 = 601,488, E1P3 = 312,997, E2P = 231, E2O = 15:
Block count where all addresses used for pointer #1 - 16,644,800
Block count where all addresses used for pointer #2 - 15,397,700
Block count where all addresses used for pointer #3 - 16,589,600
Block count where all addresses used for pointer #4 - 38,510,600

E1P1 = 1,497,974, E1P2 = 1,780,852, E1P3 = 1,664,764, E2P = 153, E2O = 52:
Block count where all addresses used for pointer #1 - 15,308,700
Block count where all addresses used for pointer #2 - 15,109,800
Block count where all addresses used for pointer #3 - 16,873,800
Block count where all addresses used for pointer #4 - 31,063,300

E1P1 = 1,529,197, E1P2 = 1,839,121, E1P3 = 1,427,365, E2P = 119, E2O = 182:
Block count where all addresses used for pointer #1 - 15,886,200
Block count where all addresses used for pointer #2 - 16,607,100
Block count where all addresses used for pointer #3 - 15,523,000
Block count where all addresses used for pointer #4 - 30,026,900

E1P1 = 1,544,515, E1P2 = 1,957,134, E1P3 = 712,492, E2P = 209, E2O = 59:
Block count where all addresses used for pointer #1 - 16,226,600
Block count where all addresses used for pointer #2 - 16,130,100
Block count where all addresses used for pointer #3 - 17,977,800
Block count where all addresses used for pointer #4 - 30,052,600

E1P1 = 1,554,631, E1P2 = 1,078,914, E1P3 = 1,696,464, E2P = 197, E2O = 73:
Block count where all addresses used for pointer #1 - 18,435,600
Block count where all addresses used for pointer #2 - 16,274,300
Block count where all addresses used for pointer #3 - 16,484,300
Block count where all addresses used for pointer #4 - 33,120,000

E1P1 = 1,572,296, E1P2 = 2,067,838, E1P3 = 1,371,102, E2P = 53, E2O = 208:
Block count where all addresses used for pointer #1 - 15,801,300
Block count where all addresses used for pointer #2 - 15,640,200
Block count where all addresses used for pointer #3 - 16,179,400
Block count where all addresses used for pointer #4 - 28,792,600

E1P1 = 1,576,485, E1P2 = 114,535, E1P3 = 1,256,736, E2P = 202, E2O = 43:
Block count where all addresses used for pointer #1 - 15,223,600
Block count where all addresses used for pointer #2 - 18,192,800
Block count where all addresses used for pointer #3 - 15,699,900
Block count where all addresses used for pointer #4 - 28,247,800

E1P1 = 1,579,643, E1P2 = 489,703, E1P3 = 681,251, E2P = 100, E2O = 91:
Block count where all addresses used for pointer #1 - 16,446,300
Block count where all addresses used for pointer #2 - 21,735,300
Block count where all addresses used for pointer #3 - 16,033,500
Block count where all addresses used for pointer #4 - 30,016,100

E1P1 = 1,585,609, E1P2 = 912,125, E1P3 = 190,177, E2P = 251, E2O = 233:
Block count where all addresses used for pointer #1 - 17,165,600
Block count where all addresses used for pointer #2 - 15,418,800
Block count where all addresses used for pointer #3 - 15,739,800
Block count where all addresses used for pointer #4 - 32,321,300

E1P1 = 1,602,686, E1P2 = 295,260, E1P3 = 1,247,300, E2P = 43, E2O = 147:
Block count where all addresses used for pointer #1 - 16,241,000
Block count where all addresses used for pointer #2 - 17,124,100
Block count where all addresses used for pointer #3 - 19,452,200
Block count where all addresses used for pointer #4 - 28,292,400

E1P1 = 1,631,933, E1P2 = 39,969, E1P3 = 811,893, E2P = 242, E2O = 117:
Block count where all addresses used for pointer #1 - 15,000,100
Block count where all addresses used for pointer #2 - 17,151,300
Block count where all addresses used for pointer #3 - 17,972,700
Block count where all addresses used for pointer #4 - 32,104,700

E1P1 = 1,648,236, E1P2 = 1,945,748, E1P3 = 633,755, E2P = 247, E2O = 13:
Block count where all addresses used for pointer #1 - 16,207,700
Block count where all addresses used for pointer #2 - 16,945,300
Block count where all addresses used for pointer #3 - 16,444,600
Block count where all addresses used for pointer #4 - 33,847,300

E1P1 = 1,652,229, E1P2 = 1,472,200, E1P3 = 555,518, E2P = 88, E2O = 185:
Block count where all addresses used for pointer #1 - 18,660,500
Block count where all addresses used for pointer #2 - 15,868,600
Block count where all addresses used for pointer #3 - 16,345,700
Block count where all addresses used for pointer #4 - 32,329,400

E1P1 = 1,659,332, E1P2 = 1,474,954, E1P3 = 2,047,930, E2P = 208, E2O = 66:
Block count where all addresses used for pointer #1 - 15,883,400
Block count where all addresses used for pointer #2 - 18,482,100
Block count where all addresses used for pointer #3 - 17,076,200
Block count where all addresses used for pointer #4 - 30,674,400

E1P1 = 1,661,577, E1P2 = 1,862,203, E1P3 = 490,660, E2P = 124, E2O = 79:
Block count where all addresses used for pointer #1 - 18,825,700
Block count where all addresses used for pointer #2 - 17,370,500
Block count where all addresses used for pointer #3 - 17,062,600
Block count where all addresses used for pointer #4 - 31,053,300

E1P1 = 1,674,121, E1P2 = 2,064,315, E1P3 = 1,393,317, E2P = 216, E2O = 210:
Block count where all addresses used for pointer #1 - 15,825,300
Block count where all addresses used for pointer #2 - 15,784,800
Block count where all addresses used for pointer #3 - 15,143,500
Block count where all addresses used for pointer #4 - 31,617,500

E1P1 = 1,677,397, E1P2 = 212,697, E1P3 = 1,332,428, E2P = 231, E2O = 236:
Block count where all addresses used for pointer #1 - 17,290,400
Block count where all addresses used for pointer #2 - 18,861,400
Block count where all addresses used for pointer #3 - 16,650,200
Block count where all addresses used for pointer #4 - 31,551,200

E1P1 = 1,683,855, E1P2 = 413,225, E1P3 = 857,565, E2P = 71, E2O = 24:
Block count where all addresses used for pointer #1 - 16,630,700
Block count where all addresses used for pointer #2 - 16,486,700
Block count where all addresses used for pointer #3 - 18,763,300
Block count where all addresses used for pointer #4 - 33,386,800

E1P1 = 1,691,789, E1P2 = 597,681, E1P3 = 1,121,221, E2P = 171, E2O = 205:
Block count where all addresses used for pointer #1 - 16,828,700
Block count where all addresses used for pointer #2 - 17,161,400
Block count where all addresses used for pointer #3 - 16,019,200
Block count where all addresses used for pointer #4 - 31,829,000

E1P1 = 1,707,063, E1P2 = 189,618, E1P3 = 1,597,121, E2P = 175, E2O = 204:
Block count where all addresses used for pointer #1 - 18,970,000
Block count where all addresses used for pointer #2 - 16,409,300
Block count where all addresses used for pointer #3 - 16,023,100
Block count where all addresses used for pointer #4 - 30,807,200

E1P1 = 1,709,834, E1P2 = 643,129, E1P3 = 1,996,844, E2P = 24, E2O = 31:
Block count where all addresses used for pointer #1 - 18,053,100
Block count where all addresses used for pointer #2 - 19,073,300
Block count where all addresses used for pointer #3 - 15,209,600
Block count where all addresses used for pointer #4 - 41,661,200

E1P1 = 1,716,939, E1P2 = 309,877, E1P3 = 1,179,897, E2P = 204, E2O = 150:
Block count where all addresses used for pointer #1 - 16,400,300
Block count where all addresses used for pointer #2 - 20,065,900
Block count where all addresses used for pointer #3 - 15,262,200
Block count where all addresses used for pointer #4 - 34,829,700

E1P1 = 1,722,884, E1P2 = 323,658, E1P3 = 1,961,208, E2P = 49, E2O = 208:
Block count where all addresses used for pointer #1 - 18,428,800
Block count where all addresses used for pointer #2 - 19,750,500
Block count where all addresses used for pointer #3 - 17,918,100
Block count where all addresses used for pointer #4 - 32,444,500

E1P1 = 1,780,950, E1P2 = 902,486, E1P3 = 1,504,597, E2P = 243, E2O = 102:
Block count where all addresses used for pointer #1 - 16,245,800
Block count where all addresses used for pointer #2 - 16,385,000
Block count where all addresses used for pointer #3 - 16,698,200
Block count where all addresses used for pointer #4 - 35,196,700

E1P1 = 1,785,540, E1P2 = 377,099, E1P3 = 814,518, E2P = 158, E2O = 12:
Block count where all addresses used for pointer #1 - 17,993,900
Block count where all addresses used for pointer #2 - 16,528,700
Block count where all addresses used for pointer #3 - 15,622,200
Block count where all addresses used for pointer #4 - 30,453,700

E1P1 = 1,802,944, E1P2 = 2,023,062, E1P3 = 1,661,078, E2P = 146, E2O = 100:
Block count where all addresses used for pointer #1 - 15,000,100
Block count where all addresses used for pointer #2 - 17,142,900
Block count where all addresses used for pointer #3 - 15,886,800
Block count where all addresses used for pointer #4 - 27,379,000

E1P1 = 1,821,174, E1P2 = 485,237, E1P3 = 1,942,136, E2P = 237, E2O = 90:
Block count where all addresses used for pointer #1 - 18,682,700
Block count where all addresses used for pointer #2 - 17,685,100
Block count where all addresses used for pointer #3 - 16,046,400
Block count where all addresses used for pointer #4 - 35,500,400

E1P1 = 1,838,059, E1P2 = 788,117, E1P3 = 1,283,607, E2P = 163, E2O = 140:
Block count where all addresses used for pointer #1 - 18,921,100
Block count where all addresses used for pointer #2 - 15,061,100
Block count where all addresses used for pointer #3 - 18,107,100
Block count where all addresses used for pointer #4 - 35,870,100

E1P1 = 1,838,819, E1P2 = 863,916, E1P3 = 1,084,114, E2P = 130, E2O = 78:
Block count where all addresses used for pointer #1 - 16,574,700
Block count where all addresses used for pointer #2 - 16,691,200
Block count where all addresses used for pointer #3 - 16,518,700
Block count where all addresses used for pointer #4 - 29,480,300

E1P1 = 1,851,372, E1P2 = 1,684,755, E1P3 = 633,629, E2P = 182, E2O = 180:
Block count where all addresses used for pointer #1 - 17,478,600
Block count where all addresses used for pointer #2 - 18,194,900
Block count where all addresses used for pointer #3 - 15,954,200
Block count where all addresses used for pointer #4 - 30,284,800

E1P1 = 1,857,295, E1P2 = 1,469,099, E1P3 = 563,542, E2P = 182, E2O = 95:
Block count where all addresses used for pointer #1 - 15,478,700
Block count where all addresses used for pointer #2 - 18,933,300
Block count where all addresses used for pointer #3 - 16,724,300
Block count where all addresses used for pointer #4 - 33,101,800

E1P1 = 1,869,143, E1P2 = 455,377, E1P3 = 1,034,213, E2P = 151, E2O = 117:
Block count where all addresses used for pointer #1 - 17,424,800
Block count where all addresses used for pointer #2 - 16,938,100
Block count where all addresses used for pointer #3 - 16,170,500
Block count where all addresses used for pointer #4 - 35,469,400

E1P1 = 1,895,878, E1P2 = 1,455,492, E1P3 = 35,788, E2P = 168, E2O = 218:
Block count where all addresses used for pointer #1 - 16,278,500
Block count where all addresses used for pointer #2 - 17,318,300
Block count where all addresses used for pointer #3 - 16,247,500
Block count where all addresses used for pointer #4 - 47,667,400

E1P1 = 1,919,891, E1P2 = 1,721,759, E1P3 = 712,443, E2P = 3, E2O = 124:
Block count where all addresses used for pointer #1 - 15,991,800
Block count where all addresses used for pointer #2 - 20,071,400
Block count where all addresses used for pointer #3 - 17,448,100
Block count where all addresses used for pointer #4 - 28,114,100

E1P1 = 1,927,293, E1P2 = 526,814, E1P3 = 213,308, E2P = 114, E2O = 106:
Block count where all addresses used for pointer #1 - 16,716,000
Block count where all addresses used for pointer #2 - 16,987,200
Block count where all addresses used for pointer #3 - 17,538,900
Block count where all addresses used for pointer #4 - 32,585,800

E1P1 = 1,944,592, E1P2 = 129,574, E1P3 = 1,130,340, E2P = 103, E2O = 30:
Block count where all addresses used for pointer #1 - 18,373,700
Block count where all addresses used for pointer #2 - 19,269,200
Block count where all addresses used for pointer #3 - 15,743,300
Block count where all addresses used for pointer #4 - 29,403,800

E1P1 = 1,986,282, E1P2 = 1,353,881, E1P3 = 1,535,756, E2P = 93, E2O = 124:
Block count where all addresses used for pointer #1 - 18,829,000
Block count where all addresses used for pointer #2 - 20,771,100
Block count where all addresses used for pointer #3 - 16,506,000
Block count where all addresses used for pointer #4 - 34,175,100

E1P1 = 10,625, E1P2 = 1,627,858, E1P3 = 371,296, E2P = 68, E2O = 204:
Block count where all addresses used for pointer #1 - 17,387,800
Block count where all addresses used for pointer #2 - 18,164,500
Block count where all addresses used for pointer #3 - 15,969,900
Block count where all addresses used for pointer #4 - 30,953,500

E1P1 = 18,976, E1P2 = 507,384, E1P3 = 989,680, E2P = 238, E2O = 73:
Block count where all addresses used for pointer #1 - 15,966,000
Block count where all addresses used for pointer #2 - 17,981,300
Block count where all addresses used for pointer #3 - 22,480,300
Block count where all addresses used for pointer #4 - 31,567,200

E1P1 = 181,525, E1P2 = 635,416, E1P3 = 69,263, E2P = 36, E2O = 155:
Block count where all addresses used for pointer #1 - 17,295,100
Block count where all addresses used for pointer #2 - 19,012,000
Block count where all addresses used for pointer #3 - 15,315,300
Block count where all addresses used for pointer #4 - 30,042,500

E1P1 = 194,917, E1P2 = 1,920,679, E1P3 = 1,018,979, E2P = 88, E2O = 253:
Block count where all addresses used for pointer #1 - 16,125,400
Block count where all addresses used for pointer #2 - 17,031,900
Block count where all addresses used for pointer #3 - 19,428,000
Block count where all addresses used for pointer #4 - 29,996,600

E1P1 = 2,046,458, E1P2 = 1,167,336, E1P3 = 147,357, E2P = 242, E2O = 127:
Block count where all addresses used for pointer #1 - 17,291,300
Block count where all addresses used for pointer #2 - 16,840,400
Block count where all addresses used for pointer #3 - 16,317,400
Block count where all addresses used for pointer #4 - 30,083,100

E1P1 = 2,065,805, E1P2 = 741,552, E1P3 = 1,348,295, E2P = 7, E2O = 206:
Block count where all addresses used for pointer #1 - 17,027,200
Block count where all addresses used for pointer #2 - 17,372,000
Block count where all addresses used for pointer #3 - 16,609,100
Block count where all addresses used for pointer #4 - 30,200,800

E1P1 = 201,951, E1P2 = 1,772,728, E1P3 = 1,974,446, E2P = 15, E2O = 43:
Block count where all addresses used for pointer #1 - 18,157,400
Block count where all addresses used for pointer #2 - 16,477,900
Block count where all addresses used for pointer #3 - 16,598,600
Block count where all addresses used for pointer #4 - 31,381,800

E1P1 = 208,907, E1P2 = 967,349, E1P3 = 321,593, E2P = 110, E2O = 119:
Block count where all addresses used for pointer #1 - 16,552,000
Block count where all addresses used for pointer #2 - 15,825,300
Block count where all addresses used for pointer #3 - 17,270,000
Block count where all addresses used for pointer #4 - 30,286,500

E1P1 = 209,400, E1P2 = 723,054, E1P3 = 603,276, E2P = 93, E2O = 157:
Block count where all addresses used for pointer #1 - 16,547,200
Block count where all addresses used for pointer #2 - 16,427,400
Block count where all addresses used for pointer #3 - 19,911,600
Block count where all addresses used for pointer #4 - 29,929,500

E1P1 = 21,191, E1P2 = 1,986,818, E1P3 = 1,792,465, E2P = 105, E2O = 151:
Block count where all addresses used for pointer #1 - 18,162,300
Block count where all addresses used for pointer #2 - 16,066,200
Block count where all addresses used for pointer #3 - 18,625,700
Block count where all addresses used for pointer #4 - 32,580,900

E1P1 = 221,698, E1P2 = 1,103,314, E1P3 = 1,749,985, E2P = 99, E2O = 187:
Block count where all addresses used for pointer #1 - 16,339,000
Block count where all addresses used for pointer #2 - 16,567,600
Block count where all addresses used for pointer #3 - 15,909,700
Block count where all addresses used for pointer #4 - 31,536,900

E1P1 = 224,903, E1P2 = 1,220,544, E1P3 = 886,678, E2P = 138, E2O = 242:
Block count where all addresses used for pointer #1 - 17,501,900
Block count where all addresses used for pointer #2 - 16,615,000
Block count where all addresses used for pointer #3 - 19,040,300
Block count where all addresses used for pointer #4 - 29,810,900

E1P1 = 234,065, E1P2 = 1,432,674, E1P3 = 1,628,592, E2P = 130, E2O = 246:
Block count where all addresses used for pointer #1 - 18,178,000
Block count where all addresses used for pointer #2 - 17,085,900
Block count where all addresses used for pointer #3 - 16,887,300
Block count where all addresses used for pointer #4 - 31,617,800

E1P1 = 234,789, E1P2 = 1,825,128, E1P3 = 709,150, E2P = 100, E2O = 202:
Block count where all addresses used for pointer #1 - 15,593,400
Block count where all addresses used for pointer #2 - 17,368,600
Block count where all addresses used for pointer #3 - 15,480,400
Block count where all addresses used for pointer #4 - 30,199,400

E1P1 = 240,385, E1P2 = 1,421,396, E1P3 = 1,115,099, E2P = 198, E2O = 79:
Block count where all addresses used for pointer #1 - 16,822,100
Block count where all addresses used for pointer #2 - 16,785,700
Block count where all addresses used for pointer #3 - 15,990,400
Block count where all addresses used for pointer #4 - 28,520,000

E1P1 = 240,541, E1P2 = 773,505, E1P3 = 544,597, E2P = 144, E2O = 232:
Block count where all addresses used for pointer #1 - 15,622,800
Block count where all addresses used for pointer #2 - 16,118,600
Block count where all addresses used for pointer #3 - 17,715,900
Block count where all addresses used for pointer #4 - 31,293,500

E1P1 = 244,966, E1P2 = 434,084, E1P3 = 2,017,002, E2P = 143, E2O = 95:
Block count where all addresses used for pointer #1 - 19,811,500
Block count where all addresses used for pointer #2 - 20,096,400
Block count where all addresses used for pointer #3 - 16,359,100
Block count where all addresses used for pointer #4 - 29,907,600

E1P1 = 250,202, E1P2 = 1,696,073, E1P3 = 427,260, E2P = 16, E2O = 6:
Block count where all addresses used for pointer #1 - 15,083,200
Block count where all addresses used for pointer #2 - 18,763,100
Block count where all addresses used for pointer #3 - 17,106,700
Block count where all addresses used for pointer #4 - 32,314,700

E1P1 = 253,061, E1P2 = 1,104,585, E1P3 = 542,589, E2P = 38, E2O = 104:
Block count where all addresses used for pointer #1 - 18,356,300
Block count where all addresses used for pointer #2 - 15,000,100
Block count where all addresses used for pointer #3 - 19,211,000
Block count where all addresses used for pointer #4 - 35,771,700

E1P1 = 256,747, E1P2 = 1,187,607, E1P3 = 1,977,362, E2P = 224, E2O = 84:
Block count where all addresses used for pointer #1 - 15,201,400
Block count where all addresses used for pointer #2 - 18,673,100
Block count where all addresses used for pointer #3 - 16,886,900
Block count where all addresses used for pointer #4 - 34,833,800

E1P1 = 261,282, E1P2 = 55,025, E1P3 = 1,654,147, E2P = 201, E2O = 225:
Block count where all addresses used for pointer #1 - 15,685,200
Block count where all addresses used for pointer #2 - 16,741,700
Block count where all addresses used for pointer #3 - 17,394,400
Block count where all addresses used for pointer #4 - 33,046,600

E1P1 = 290,694, E1P2 = 1,576,517, E1P3 = 649,095, E2P = 33, E2O = 30:
Block count where all addresses used for pointer #1 - 20,299,200
Block count where all addresses used for pointer #2 - 17,521,000
Block count where all addresses used for pointer #3 - 15,392,600
Block count where all addresses used for pointer #4 - 29,160,300

E1P1 = 316,522, E1P2 = 1,446,296, E1P3 = 1,722,256, E2P = 222, E2O = 84:
Block count where all addresses used for pointer #1 - 18,448,500
Block count where all addresses used for pointer #2 - 18,834,600
Block count where all addresses used for pointer #3 - 15,874,300
Block count where all addresses used for pointer #4 - 29,774,600

E1P1 = 331,421, E1P2 = 1,309,697, E1P3 = 572,756, E2P = 94, E2O = 99:
Block count where all addresses used for pointer #1 - 15,121,400
Block count where all addresses used for pointer #2 - 16,203,600
Block count where all addresses used for pointer #3 - 17,064,500
Block count where all addresses used for pointer #4 - 32,268,600

E1P1 = 343,102, E1P2 = 885,148, E1P3 = 524,546, E2P = 228, E2O = 198:
Block count where all addresses used for pointer #1 - 17,086,700
Block count where all addresses used for pointer #2 - 16,634,900
Block count where all addresses used for pointer #3 - 17,868,200
Block count where all addresses used for pointer #4 - 31,058,700

E1P1 = 387,970, E1P2 = 930,513, E1P3 = 1,683,556, E2P = 193, E2O = 125:
Block count where all addresses used for pointer #1 - 16,887,400
Block count where all addresses used for pointer #2 - 17,243,100
Block count where all addresses used for pointer #3 - 18,719,400
Block count where all addresses used for pointer #4 - 32,727,000

E1P1 = 41,109, E1P2 = 1,906,967, E1P3 = 388,371, E2P = 53, E2O = 132:
Block count where all addresses used for pointer #1 - 19,571,100
Block count where all addresses used for pointer #2 - 16,007,800
Block count where all addresses used for pointer #3 - 16,257,800
Block count where all addresses used for pointer #4 - 39,329,300

E1P1 = 419,411, E1P2 = 1,678,814, E1P3 = 2,022,588, E2P = 67, E2O = 239:
Block count where all addresses used for pointer #1 - 15,765,600
Block count where all addresses used for pointer #2 - 19,244,400
Block count where all addresses used for pointer #3 - 16,156,400
Block count where all addresses used for pointer #4 - 32,038,900

E1P1 = 42,726, E1P2 = 1,587,492, E1P3 = 979,945, E2P = 195, E2O = 8:
Block count where all addresses used for pointer #1 - 16,130,100
Block count where all addresses used for pointer #2 - 16,990,700
Block count where all addresses used for pointer #3 - 15,639,500
Block count where all addresses used for pointer #4 - 30,587,200

E1P1 = 45,327, E1P2 = 1,070,634, E1P3 = 780,889, E2P = 61, E2O = 249:
Block count where all addresses used for pointer #1 - 16,019,400
Block count where all addresses used for pointer #2 - 17,682,400
Block count where all addresses used for pointer #3 - 17,062,500
Block count where all addresses used for pointer #4 - 35,355,200

E1P1 = 450,771, E1P2 = 120,542, E1P3 = 735,293, E2P = 114, E2O = 236:
Block count where all addresses used for pointer #1 - 15,460,000
Block count where all addresses used for pointer #2 - 17,900,900
Block count where all addresses used for pointer #3 - 17,755,000
Block count where all addresses used for pointer #4 - 29,861,800

E1P1 = 46,776, E1P2 = 1,452,335, E1P3 = 1,623,370, E2P = 176, E2O = 38:
Block count where all addresses used for pointer #1 - 17,000,100
Block count where all addresses used for pointer #2 - 19,532,100
Block count where all addresses used for pointer #3 - 15,994,600
Block count where all addresses used for pointer #4 - 31,992,500

E1P1 = 473,014, E1P2 = 794,165, E1P3 = 1,736,248, E2P = 49, E2O = 168:
Block count where all addresses used for pointer #1 - 16,658,800
Block count where all addresses used for pointer #2 - 18,360,100
Block count where all addresses used for pointer #3 - 15,669,500
Block count where all addresses used for pointer #4 - 33,783,000

E1P1 = 474,879, E1P2 = 94,939, E1P3 = 278,726, E2P = 108, E2O = 83:
Block count where all addresses used for pointer #1 - 19,804,300
Block count where all addresses used for pointer #2 - 15,397,100
Block count where all addresses used for pointer #3 - 16,917,800
Block count where all addresses used for pointer #4 - 32,744,400

E1P1 = 488,363, E1P2 = 1,737,943, E1P3 = 785,106, E2P = 236, E2O = 38:
Block count where all addresses used for pointer #1 - 15,308,100
Block count where all addresses used for pointer #2 - 15,687,200
Block count where all addresses used for pointer #3 - 16,197,200
Block count where all addresses used for pointer #4 - 30,213,300

E1P1 = 502,297, E1P2 = 244,619, E1P3 = 1,755,316, E2P = 206, E2O = 240:
Block count where all addresses used for pointer #1 - 16,923,800
Block count where all addresses used for pointer #2 - 20,301,500
Block count where all addresses used for pointer #3 - 16,852,500
Block count where all addresses used for pointer #4 - 35,348,700

E1P1 = 530,566, E1P2 = 1,029,444, E1P3 = 199,820, E2P = 3, E2O = 255:
Block count where all addresses used for pointer #1 - 19,564,600
Block count where all addresses used for pointer #2 - 15,958,000
Block count where all addresses used for pointer #3 - 17,765,800
Block count where all addresses used for pointer #4 - 30,046,700

E1P1 = 551,980, E1P2 = 1,342,161, E1P3 = 359,011, E2P = 7, E2O = 201:
Block count where all addresses used for pointer #1 - 16,938,100
Block count where all addresses used for pointer #2 - 17,258,400
Block count where all addresses used for pointer #3 - 15,386,300
Block count where all addresses used for pointer #4 - 38,954,400

E1P1 = 553,294, E1P2 = 884,846, E1P3 = 207,502, E2P = 141, E2O = 239:
Block count where all addresses used for pointer #1 - 17,143,100
Block count where all addresses used for pointer #2 - 15,728,900
Block count where all addresses used for pointer #3 - 16,925,000
Block count where all addresses used for pointer #4 - 31,869,200

E1P1 = 556,073, E1P2 = 1,867,482, E1P3 = 315,976, E2P = 102, E2O = 66:
Block count where all addresses used for pointer #1 - 17,219,600
Block count where all addresses used for pointer #2 - 16,124,500
Block count where all addresses used for pointer #3 - 15,805,000
Block count where all addresses used for pointer #4 - 29,929,300

E1P1 = 557,322, E1P2 = 341,047, E1P3 = 26,161, E2P = 71, E2O = 184:
Block count where all addresses used for pointer #1 - 18,560,700
Block count where all addresses used for pointer #2 - 16,412,600
Block count where all addresses used for pointer #3 - 16,621,000
Block count where all addresses used for pointer #4 - 30,183,400

E1P1 = 558,100, E1P2 = 1,430,297, E1P3 = 1,872,267, E2P = 82, E2O = 137:
Block count where all addresses used for pointer #1 - 16,629,900
Block count where all addresses used for pointer #2 - 15,652,700
Block count where all addresses used for pointer #3 - 17,657,900
Block count where all addresses used for pointer #4 - 33,251,500

E1P1 = 606,543, E1P2 = 478,697, E1P3 = 1,509,786, E2P = 129, E2O = 206:
Block count where all addresses used for pointer #1 - 17,474,600
Block count where all addresses used for pointer #2 - 15,465,800
Block count where all addresses used for pointer #3 - 16,144,500
Block count where all addresses used for pointer #4 - 29,303,300

E1P1 = 629,809, E1P2 = 212,674, E1P3 = 1,192,592, E2P = 254, E2O = 239:
Block count where all addresses used for pointer #1 - 19,320,300
Block count where all addresses used for pointer #2 - 16,598,000
Block count where all addresses used for pointer #3 - 17,236,300
Block count where all addresses used for pointer #4 - 32,010,200

E1P1 = 64,286, E1P2 = 1,014,398, E1P3 = 1,603,037, E2P = 48, E2O = 140:
Block count where all addresses used for pointer #1 - 15,849,900
Block count where all addresses used for pointer #2 - 19,214,900
Block count where all addresses used for pointer #3 - 16,946,300
Block count where all addresses used for pointer #4 - 32,861,600

E1P1 = 644,888, E1P2 = 1,261,966, E1P3 = 1,970,094, E2P = 243, E2O = 198:
Block count where all addresses used for pointer #1 - 17,230,700
Block count where all addresses used for pointer #2 - 18,044,500
Block count where all addresses used for pointer #3 - 18,317,100
Block count where all addresses used for pointer #4 - 36,283,300

E1P1 = 647,983, E1P2 = 276,809, E1P3 = 513,149, E2P = 217, E2O = 38:
Block count where all addresses used for pointer #1 - 16,600,700
Block count where all addresses used for pointer #2 - 17,849,500
Block count where all addresses used for pointer #3 - 18,592,300
Block count where all addresses used for pointer #4 - 30,185,800

E1P1 = 650,550, E1P2 = 1,680,948, E1P3 = 1,240,249, E2P = 158, E2O = 202:
Block count where all addresses used for pointer #1 - 20,086,700
Block count where all addresses used for pointer #2 - 16,113,400
Block count where all addresses used for pointer #3 - 17,504,000
Block count where all addresses used for pointer #4 - 35,279,600

E1P1 = 651,622, E1P2 = 505,766, E1P3 = 1,465,957, E2P = 195, E2O = 176:
Block count where all addresses used for pointer #1 - 19,081,800
Block count where all addresses used for pointer #2 - 18,651,800
Block count where all addresses used for pointer #3 - 15,523,500
Block count where all addresses used for pointer #4 - 32,532,000

E1P1 = 653,016, E1P2 = 190,671, E1P3 = 1,640,042, E2P = 14, E2O = 224:
Block count where all addresses used for pointer #1 - 16,211,100
Block count where all addresses used for pointer #2 - 19,152,300
Block count where all addresses used for pointer #3 - 15,324,600
Block count where all addresses used for pointer #4 - 38,170,300

E1P1 = 663,097, E1P2 = 1,367,851, E1P3 = 1,957,332, E2P = 104, E2O = 118:
Block count where all addresses used for pointer #1 - 19,085,600
Block count where all addresses used for pointer #2 - 16,811,700
Block count where all addresses used for pointer #3 - 15,962,600
Block count where all addresses used for pointer #4 - 31,429,400

E1P1 = 696,122, E1P2 = 1,329,320, E1P3 = 973,790, E2P = 179, E2O = 254:
Block count where all addresses used for pointer #1 - 15,784,500
Block count where all addresses used for pointer #2 - 16,920,600
Block count where all addresses used for pointer #3 - 17,857,700
Block count where all addresses used for pointer #4 - 29,217,600

E1P1 = 703,294, E1P2 = 1,706,781, E1P3 = 348,415, E2P = 110, E2O = 153:
Block count where all addresses used for pointer #1 - 15,000,100
Block count where all addresses used for pointer #2 - 19,418,400
Block count where all addresses used for pointer #3 - 16,065,500
Block count where all addresses used for pointer #4 - 29,927,200

E1P1 = 741,968, E1P2 = 222,568, E1P3 = 385,952, E2P = 226, E2O = 45:
Block count where all addresses used for pointer #1 - 15,262,600
Block count where all addresses used for pointer #2 - 18,264,100
Block count where all addresses used for pointer #3 - 16,576,800
Block count where all addresses used for pointer #4 - 33,174,100

E1P1 = 791,952, E1P2 = 1,950,759, E1P3 = 183,777, E2P = 219, E2O = 195:
Block count where all addresses used for pointer #1 - 20,172,500
Block count where all addresses used for pointer #2 - 19,369,100
Block count where all addresses used for pointer #3 - 15,864,800
Block count where all addresses used for pointer #4 - 33,349,600

E1P1 = 829,047, E1P2 = 267,762, E1P3 = 641,793, E2P = 21, E2O = 130:
Block count where all addresses used for pointer #1 - 16,497,400
Block count where all addresses used for pointer #2 - 16,171,900
Block count where all addresses used for pointer #3 - 15,467,200
Block count where all addresses used for pointer #4 - 33,668,900

E1P1 = 829,069, E1P2 = 542,000, E1P3 = 962,246, E2P = 45, E2O = 2:
Block count where all addresses used for pointer #1 - 18,188,700
Block count where all addresses used for pointer #2 - 16,122,300
Block count where all addresses used for pointer #3 - 17,720,500
Block count where all addresses used for pointer #4 - 30,364,400

E1P1 = 829,864, E1P2 = 569,949, E1P3 = 394,175, E2P = 62, E2O = 90:
Block count where all addresses used for pointer #1 - 17,162,300
Block count where all addresses used for pointer #2 - 18,673,600
Block count where all addresses used for pointer #3 - 16,471,300
Block count where all addresses used for pointer #4 - 33,448,900

E1P1 = 832,256, E1P2 = 1,877,334, E1P3 = 654,804, E2P = 67, E2O = 255:
Block count where all addresses used for pointer #1 - 16,608,800
Block count where all addresses used for pointer #2 - 16,640,600
Block count where all addresses used for pointer #3 - 20,387,300
Block count where all addresses used for pointer #4 - 29,668,700

E1P1 = 834,060, E1P2 = 1,601,331, E1P3 = 2,032,446, E2P = 225, E2O = 65:
Block count where all addresses used for pointer #1 - 16,707,900
Block count where all addresses used for pointer #2 - 16,606,900
Block count where all addresses used for pointer #3 - 17,308,900
Block count where all addresses used for pointer #4 - 31,554,800

E1P1 = 838,264, E1P2 = 1,154,288, E1P3 = 220,424, E2P = 91, E2O = 9:
Block count where all addresses used for pointer #1 - 18,053,800
Block count where all addresses used for pointer #2 - 15,940,500
Block count where all addresses used for pointer #3 - 17,224,300
Block count where all addresses used for pointer #4 - 29,725,000

E1P1 = 846,298, E1P2 = 2,091,208, E1P3 = 1,342,334, E2P = 96, E2O = 91:
Block count where all addresses used for pointer #1 - 16,555,200
Block count where all addresses used for pointer #2 - 18,232,600
Block count where all addresses used for pointer #3 - 15,116,400
Block count where all addresses used for pointer #4 - 31,318,200

E1P1 = 853,474, E1P2 = 1,729,457, E1P3 = 1,088,964, E2P = 7, E2O = 103:
Block count where all addresses used for pointer #1 - 17,156,700
Block count where all addresses used for pointer #2 - 16,958,400
Block count where all addresses used for pointer #3 - 16,278,500
Block count where all addresses used for pointer #4 - 30,192,300

E1P1 = 88,935, E1P2 = 352,161, E1P3 = 636,018, E2P = 140, E2O = 76:
Block count where all addresses used for pointer #1 - 16,557,900
Block count where all addresses used for pointer #2 - 16,470,100
Block count where all addresses used for pointer #3 - 18,909,100
Block count where all addresses used for pointer #4 - 29,589,000

E1P1 = 887,177, E1P2 = 1,893,309, E1P3 = 588,961, E2P = 161, E2O = 80:
Block count where all addresses used for pointer #1 - 16,833,500
Block count where all addresses used for pointer #2 - 17,471,800
Block count where all addresses used for pointer #3 - 16,520,000
Block count where all addresses used for pointer #4 - 32,550,400

E1P1 = 891,831, E1P2 = 429,490, E1P3 = 1,072,448, E2P = 87, E2O = 136:
Block count where all addresses used for pointer #1 - 19,542,200
Block count where all addresses used for pointer #2 - 16,822,200
Block count where all addresses used for pointer #3 - 17,797,900
Block count where all addresses used for pointer #4 - 32,376,400

E1P1 = 919,585, E1P2 = 1,206,771, E1P3 = 746,493, E2P = 96, E2O = 105:
Block count where all addresses used for pointer #1 - 19,671,900
Block count where all addresses used for pointer #2 - 16,341,600
Block count where all addresses used for pointer #3 - 16,364,500
Block count where all addresses used for pointer #4 - 34,660,500

E1P1 = 924,345, E1P2 = 1,786,156, E1P3 = 1,159,507, E2P = 21, E2O = 70:
Block count where all addresses used for pointer #1 - 17,213,800
Block count where all addresses used for pointer #2 - 16,186,900
Block count where all addresses used for pointer #3 - 16,955,500
Block count where all addresses used for pointer #4 - 27,906,600

E1P1 = 95,471, E1P2 = 1,411,722, E1P3 = 380,217, E2P = 235, E2O = 153:
Block count where all addresses used for pointer #1 - 15,805,900
Block count where all addresses used for pointer #2 - 17,169,900
Block count where all addresses used for pointer #3 - 18,346,300
Block count where all addresses used for pointer #4 - 29,920,700

E1P1 = 953,628, E1P2 = 1,326,212, E1P3 = 116,683, E2P = 26, E2O = 113:
Block count where all addresses used for pointer #1 - 15,859,900
Block count where all addresses used for pointer #2 - 17,637,100
Block count where all addresses used for pointer #3 - 17,350,900
Block count where all addresses used for pointer #4 - 32,394,100

E1P1 = 958,355, E1P2 = 1,438,238, E1P3 = 712,700, E2P = 11, E2O = 224:
Block count where all addresses used for pointer #1 - 15,242,200
Block count where all addresses used for pointer #2 - 15,804,700
Block count where all addresses used for pointer #3 - 16,422,000
Block count where all addresses used for pointer #4 - 28,762,100

E1P1 = 965,070, E1P2 = 1,765,740, E1P3 = 260,116, E2P = 174, E2O = 187:
Block count where all addresses used for pointer #1 - 15,603,400
Block count where all addresses used for pointer #2 - 16,293,400
Block count where all addresses used for pointer #3 - 22,069,200
Block count where all addresses used for pointer #4 - 30,089,200

E1P1 = 97,502, E1P2 = 302,526, E1P3 = 1,512,094, E2P = 81, E2O = 188:
Block count where all addresses used for pointer #1 - 15,644,400
Block count where all addresses used for pointer #2 - 18,505,500
Block count where all addresses used for pointer #3 - 19,538,600
Block count where all addresses used for pointer #4 - 38,379,700

E1P1 = 973,347, E1P2 = 1,278,574, E1P3 = 1,535,756, E2P = 95, E2O = 92:
Block count where all addresses used for pointer #1 - 16,469,400
Block count where all addresses used for pointer #2 - 15,796,200
Block count where all addresses used for pointer #3 - 16,481,900
Block count where all addresses used for pointer #4 - 28,988,500

E1P1 = 993,322, E1P2 = 1,492,312, E1P3 = 1,384,269, E2P = 3, E2O = 202:
Block count where all addresses used for pointer #1 - 15,320,600
Block count where all addresses used for pointer #2 - 15,283,900
Block count where all addresses used for pointer #3 - 18,931,300
Block count where all addresses used for pointer #4 - 30,496,800

E1P1 = 999,312, E1P2 = 1,808,166, E1P3 = 1,532,387, E2P = 60, E2O = 212:
Block count where all addresses used for pointer #1 - 15,147,400
Block count where all addresses used for pointer #2 - 17,777,500
Block count where all addresses used for pointer #3 - 17,733,000
Block count where all addresses used for pointer #4 - 32,524,700

Appendix E

This is the code used to produce the initial 32 byte key table for this Vernam Two system. ‘GRN’ accesses the true random number generator and returns a random value from 0 to 255 inclusive. ‘ls’ merely changes the number to an ASCII string and truncates the leading space character.

```
Sub MakeKeyTable()
    Dim i As Integer, chkStr As String

    ' Open the key table for writing
    Open App.Path + "\vernamTwo.tbl" For Output As #1
    ' Write the key table to the file, 31 numbers plus comma, then the last one without a comma
    For i = 0 To 30: Print #1, ls(GRN); ",";: Next i: Print #1, ls(GRN)
    ' Close the file so the input function operates
    Close #1
End Sub
```

The 32 key numbers used for this technology demonstration document:

```
68,107,254,204,57,147,157,136,34,67,58,152,151,79,52,184,89,95,10,93,15,211,153,198,67,103,238,135,142,48,155,24
```

The code used to convert this 32 byte key into an 8,389,631 byte key is as follows:

```
' This function loads the the key.
Sub loadKeyTable()
    Dim i As Byte, p1 as long, p2 as long, p3 as long

    ' Open the key table file, input the 512 random numbers and close the table file
    Open App.Path + "\vernamTwo.tbl" For Input As #1: For i = 0 To 31: Input #1, v2Key(i): Next i: Close #1
    ' Call the Vernam 2 version of gkey to expand these 32 bytes to 8,389,631 bytes
    Call vernam2gkey
    ' Erase ALL trace of the original 32-byte key from this 8 Megabyte key
    p1 = &H200100: p2 = &H400200: p3 = &H600300
    For i=0 to 31
        v2key(i) = v2key(i) Xor v2key(p1) Xor v2key(p2) Xor v2key(p3)
       p1 = p1 + 1: p2 = p2 + 1: p3 = p3 + 1:
    Next i
End Sub

Sub vernam2gkey()

    Dim i, p            As Long             ' Added location 'p' for Vernam 2 use
    Dim j               As Long
    Dim k               As Long
    Dim m               As Long
    Dim n               As Long
    Dim C1              As Long
    Dim C2              As Long
    Dim C3              As Long
    Dim Ciphedkey(7)    As Long
    Dim Y               As Long

    Dim m_Nk        As Long
    Dim m_Nb        As Long
    Dim m_Nr        As Long
    Dim m_ekey(2097407) As Long             ' Moved here, not needed outside this function for Vernam 2 use

    Call Class_Initialize: Call genTables   ' Added for Vernam 2 use

    Y = 1
    m_Nb = 8
    m_Nk = 8

    If m_Nb >= m_Nk Then
        m_Nr = 6 + m_Nb
    Else
        m_Nr = 6 + m_Nk
    End If

    C1 = 1
    If m_Nb < 8 Then
        C2 = 2
        C3 = 3
    Else
        C2 = 3
        C3 = 4
    End If

    For j = 0 To 7
        m = j * 3

        m_fi(m) = (j + C1) Mod 8
        m_fi(m + 1) = (j + C2) Mod 8
        m_fi(m + 2) = (j + C3) Mod 8
        m_ri(m) = (8 + j - C1) Mod 8
        m_ri(m + 1) = (8 + j - C2) Mod 8
        m_ri(m + 2) = (8 + j - C3) Mod 8
    Next
```

```
    n = 2097408                             ' Changed to 2097408 for Vernam 2 use from 120

    For i = 0 To m_Nk - 1
        'j = i * 4
        Ciphedkey(i) = PackFrom(v2Key, i * 4)
    Next

    For i = 0 To m_Nk - 1
        m_ekey(i) = Ciphedkey(i)
    Next

    j = m_Nk
    k = 0
    Do While j < n
        m_ekey(j) = m_ekey(j - m_Nk) Xor SubByte(RotateLeft(m_ekey(j - 1), 24)) Xor Y
        Y = xtime(Y)
        i = 1
        Do While i < 4 And (i + j) < n
            m_ekey(i + j) = m_ekey(i + j - m_Nk) Xor m_ekey(i + j - 1)
            i = i + 1
        Loop
        If j + 4 < n Then
            m_ekey(j + 4) = m_ekey(j + 4 - m_Nk) Xor SubByte(m_ekey(j + 3))
        End If
        i = 5
        Do While i < m_Nk And (i + j) < n
            m_ekey(i + j) = m_ekey(i + j - m_Nk) Xor m_ekey(i + j - 1)
            i = i + 1
        Loop

        j = j + m_Nk
        k = k + 1
        If (k Mod 1000) = 0 Then DoEvents
    Loop

    ' ************************************************************
    ' END OF AES gkey REGULAR CODE, BEGIN VERNAM 2 TRANSFERING CODE
    ' ************************************************************
    ' Load the 8,389,631 bytes to the module global key structure
    p = 0
    For i = 0 To 2097407
        v2Key(p) = (m_ekey(i) And &HFF&)
        v2Key(p + 1) = Int(m_ekey(i) / &H100&) And &HFF&
        v2Key(p + 2) = Int(m_ekey(i) / &H10000) And &HFF&
        v2Key(p + 3) = Int(m_ekey(i) / &H1000000) And &HFF&
        p = p + 4
    Next i
End Sub
```

Appendix F

The byte distribution of the resulting 8,389,631 byte pseudo-randomly expanded from the 32 byte key, produced by the modified 'gkey' function in Appendix E, whos numeric distribution was shown is as follows:

```
Byte 0 occurs 32,553 times.     Byte 64 occurs 32,987 times.    Byte 128 occurs 32,754 times.   Byte 192 occurs 32,992 times.
Byte 1 occurs 32,254 times.     Byte 65 occurs 32,973 times.    Byte 129 occurs 32,699 times.   Byte 193 occurs 32,785 times.
Byte 2 occurs 32,554 times.     Byte 66 occurs 32,681 times.    Byte 130 occurs 32,726 times.   Byte 194 occurs 33,148 times.
Byte 3 occurs 32,446 times.     Byte 67 occurs 32,785 times.    Byte 131 occurs 32,434 times.   Byte 195 occurs 32,861 times.
Byte 4 occurs 32,915 times.     Byte 68 occurs 32,723 times.    Byte 132 occurs 33,166 times.   Byte 196 occurs 32,842 times.
Byte 5 occurs 32,935 times.     Byte 69 occurs 32,315 times.    Byte 133 occurs 32,767 times.   Byte 197 occurs 32,888 times.
Byte 6 occurs 33,202 times.     Byte 70 occurs 32,896 times.    Byte 134 occurs 32,776 times.   Byte 198 occurs 32,651 times.
Byte 7 occurs 32,830 times.     Byte 71 occurs 32,868 times.    Byte 135 occurs 32,637 times.   Byte 199 occurs 33,008 times.
Byte 8 occurs 33,044 times.     Byte 72 occurs 32,707 times.    Byte 136 occurs 33,005 times.   Byte 200 occurs 32,703 times.
Byte 9 occurs 32,601 times.     Byte 73 occurs 32,727 times.    Byte 137 occurs 32,606 times.   Byte 201 occurs 32,919 times.
Byte 10 occurs 32,364 times.    Byte 74 occurs 32,944 times.    Byte 138 occurs 32,602 times.   Byte 202 occurs 32,814 times.
Byte 11 occurs 32,968 times.    Byte 75 occurs 32,558 times.    Byte 139 occurs 33,294 times.   Byte 203 occurs 32,463 times.
Byte 12 occurs 32,866 times.    Byte 76 occurs 33,038 times.    Byte 140 occurs 32,400 times.   Byte 204 occurs 32,884 times.
Byte 13 occurs 32,848 times.    Byte 77 occurs 32,961 times.    Byte 141 occurs 32,502 times.   Byte 205 occurs 32,719 times.
Byte 14 occurs 32,399 times.    Byte 78 occurs 32,608 times.    Byte 142 occurs 32,619 times.   Byte 206 occurs 32,806 times.
Byte 15 occurs 32,749 times.    Byte 79 occurs 32,524 times.    Byte 143 occurs 32,879 times.   Byte 207 occurs 32,527 times.
Byte 16 occurs 32,442 times.    Byte 80 occurs 33,041 times.    Byte 144 occurs 32,688 times.   Byte 208 occurs 33,017 times.
Byte 17 occurs 32,908 times.    Byte 81 occurs 32,573 times.    Byte 145 occurs 32,730 times.   Byte 209 occurs 32,482 times.
Byte 18 occurs 32,756 times.    Byte 82 occurs 32,854 times.    Byte 146 occurs 32,749 times.   Byte 210 occurs 32,665 times.
Byte 19 occurs 32,417 times.    Byte 83 occurs 32,675 times.    Byte 147 occurs 32,814 times.   Byte 211 occurs 32,619 times.
Byte 20 occurs 32,720 times.    Byte 84 occurs 32,614 times.    Byte 148 occurs 33,146 times.   Byte 212 occurs 32,891 times.
Byte 21 occurs 32,651 times.    Byte 85 occurs 32,829 times.    Byte 149 occurs 32,744 times.   Byte 213 occurs 32,852 times.
Byte 22 occurs 33,039 times.    Byte 86 occurs 32,802 times.    Byte 150 occurs 32,755 times.   Byte 214 occurs 32,768 times.
Byte 23 occurs 32,760 times.    Byte 87 occurs 32,565 times.    Byte 151 occurs 32,634 times.   Byte 215 occurs 32,671 times.
Byte 24 occurs 32,709 times.    Byte 88 occurs 32,731 times.    Byte 152 occurs 32,587 times.   Byte 216 occurs 32,664 times.
Byte 25 occurs 33,274 times.    Byte 89 occurs 32,826 times.    Byte 153 occurs 32,769 times.   Byte 217 occurs 32,521 times.
Byte 26 occurs 32,672 times.    Byte 90 occurs 32,807 times.    Byte 154 occurs 33,084 times.   Byte 218 occurs 33,050 times.
Byte 27 occurs 32,718 times.    Byte 91 occurs 32,845 times.    Byte 155 occurs 32,597 times.   Byte 219 occurs 32,862 times.
Byte 28 occurs 32,557 times.    Byte 92 occurs 33,190 times.    Byte 156 occurs 32,713 times.   Byte 220 occurs 32,881 times.
Byte 29 occurs 32,689 times.    Byte 93 occurs 32,890 times.    Byte 157 occurs 32,901 times.   Byte 221 occurs 32,828 times.
Byte 30 occurs 32,758 times.    Byte 94 occurs 32,969 times.    Byte 158 occurs 32,653 times.   Byte 222 occurs 32,777 times.
Byte 31 occurs 32,768 times.    Byte 95 occurs 32,664 times.    Byte 159 occurs 32,814 times.   Byte 223 occurs 32,766 times.
Byte 32 occurs 32,860 times.    Byte 96 occurs 32,953 times.    Byte 160 occurs 32,969 times.   Byte 224 occurs 32,913 times.
Byte 33 occurs 32,964 times.    Byte 97 occurs 32,738 times.    Byte 161 occurs 32,770 times.   Byte 225 occurs 32,592 times.
Byte 34 occurs 32,595 times.    Byte 98 occurs 32,699 times.    Byte 162 occurs 32,613 times.   Byte 226 occurs 33,009 times.
Byte 35 occurs 33,048 times.    Byte 99 occurs 32,675 times.    Byte 163 occurs 32,735 times.   Byte 227 occurs 32,686 times.
Byte 36 occurs 33,014 times.    Byte 100 occurs 32,964 times.   Byte 164 occurs 32,620 times.   Byte 228 occurs 32,980 times.
Byte 37 occurs 32,562 times.    Byte 101 occurs 32,829 times.   Byte 165 occurs 32,717 times.   Byte 229 occurs 32,582 times.
Byte 38 occurs 33,084 times.    Byte 102 occurs 32,704 times.   Byte 166 occurs 32,889 times.   Byte 230 occurs 32,562 times.
Byte 39 occurs 32,704 times.    Byte 103 occurs 32,781 times.   Byte 167 occurs 32,942 times.   Byte 231 occurs 33,083 times.
Byte 40 occurs 32,917 times.    Byte 104 occurs 32,622 times.   Byte 168 occurs 32,873 times.   Byte 232 occurs 32,798 times.
Byte 41 occurs 32,526 times.    Byte 105 occurs 33,005 times.   Byte 169 occurs 32,983 times.   Byte 233 occurs 32,862 times.
Byte 42 occurs 32,522 times.    Byte 106 occurs 32,676 times.   Byte 170 occurs 32,759 times.   Byte 234 occurs 32,617 times.
Byte 43 occurs 32,880 times.    Byte 107 occurs 32,433 times.   Byte 171 occurs 32,718 times.   Byte 235 occurs 32,507 times.
Byte 44 occurs 32,935 times.    Byte 108 occurs 32,614 times.   Byte 172 occurs 32,991 times.   Byte 236 occurs 32,858 times.
Byte 45 occurs 32,629 times.    Byte 109 occurs 32,777 times.   Byte 173 occurs 32,515 times.   Byte 237 occurs 32,770 times.
Byte 46 occurs 32,634 times.    Byte 110 occurs 32,449 times.   Byte 174 occurs 32,685 times.   Byte 238 occurs 32,750 times.
Byte 47 occurs 32,771 times.    Byte 111 occurs 32,937 times.   Byte 175 occurs 32,853 times.   Byte 239 occurs 32,761 times.
Byte 48 occurs 32,691 times.    Byte 112 occurs 32,860 times.   Byte 176 occurs 32,685 times.   Byte 240 occurs 32,912 times.
Byte 49 occurs 32,774 times.    Byte 113 occurs 33,002 times.   Byte 177 occurs 32,394 times.   Byte 241 occurs 32,983 times.
Byte 50 occurs 32,951 times.    Byte 114 occurs 32,775 times.   Byte 178 occurs 32,656 times.   Byte 242 occurs 32,663 times.
Byte 51 occurs 33,008 times.    Byte 115 occurs 32,545 times.   Byte 179 occurs 32,842 times.   Byte 243 occurs 32,936 times.
Byte 52 occurs 33,034 times.    Byte 116 occurs 32,688 times.   Byte 180 occurs 32,725 times.   Byte 244 occurs 33,001 times.
Byte 53 occurs 32,422 times.    Byte 117 occurs 32,632 times.   Byte 181 occurs 32,517 times.   Byte 245 occurs 32,987 times.
Byte 54 occurs 32,960 times.    Byte 118 occurs 32,739 times.   Byte 182 occurs 32,892 times.   Byte 246 occurs 32,788 times.
Byte 55 occurs 32,594 times.    Byte 119 occurs 33,070 times.   Byte 183 occurs 32,880 times.   Byte 247 occurs 32,492 times.
Byte 56 occurs 32,692 times.    Byte 120 occurs 32,693 times.   Byte 184 occurs 32,655 times.   Byte 248 occurs 32,748 times.
Byte 57 occurs 32,836 times.    Byte 121 occurs 32,780 times.   Byte 185 occurs 32,771 times.   Byte 249 occurs 32,925 times.
Byte 58 occurs 32,550 times.    Byte 122 occurs 33,092 times.   Byte 186 occurs 32,792 times.   Byte 250 occurs 32,885 times.
Byte 59 occurs 32,755 times.    Byte 123 occurs 32,841 times.   Byte 187 occurs 32,597 times.   Byte 251 occurs 32,958 times.
Byte 60 occurs 32,713 times.    Byte 124 occurs 32,777 times.   Byte 188 occurs 32,350 times.   Byte 252 occurs 32,633 times.
Byte 61 occurs 32,876 times.    Byte 125 occurs 32,834 times.   Byte 189 occurs 33,030 times.   Byte 253 occurs 32,597 times.
Byte 62 occurs 32,637 times.    Byte 126 occurs 32,776 times.   Byte 190 occurs 32,917 times.   Byte 254 occurs 32,872 times.
Byte 63 occurs 32,726 times.    Byte 127 occurs 32,992 times.   Byte 191 occurs 32,902 times.   Byte 255 occurs 32,765 times.
```

This key is divided into 4 sets of keys, each 2,097,407 bytes each. Each set was tested against all others and itself for stream repeating. Set 1 = 0 to 2,097,407, set 2 = 2,097,408 to 4,194,815, set 3 = 4,194,816 to 6,292,223, set 4 = 6,292,224 to 8,389,631. Set comparisons executed: Sets 1 with 1, 1 with 2, 1 with 3, 1 with 4, 2 with 2, 2 with 3, 2 with 4, 3 with 3, 3 with 4 and 4 with 4. The results of each test, including the time it took and the number of 'IF' statements that were executed to obtain the results, are listed on the next page. Notice that at no time did any number stream longer than 5 bytes was found to exist within any single or between any two groups. At the same time, the hexadecimal number streams found were also recorded in a different file but not included here. These files are available upon request, along with a file containing the hexadecimal 8,389,631 byte key pseudo-randomly expanded.

Appendix G: Testing performed to confirm this system's design

To test this system, a process was developed to simulate the encryption using 100 million blocks of text, 12.8 Gigabytes of plaintext. The process created just the Effective Key Streams (**EKS**) and recorded the 5 pointer values used to create each stream. This produced a single data file almost 30 Gigabytes in size, 10 files of this size were produced.

A process was created to take the first **EKS** and compare it with all the other 99,999,999 **EKS**'s. The process did not just test to see if the stream was equal, it compared all 128 numbers in the two streams and counted the quantity of numbers that were found to be equal. The process would then input **EKS** #2 and execute the same comparison with the remaining 99,999,998 **EKS**'s. This would continue comparing the first 100 **EKS** streams with all the remaining streams. It took several days for this process to complete, the following is a representative sample of the output of the first two tests of just the first of 10 files tested:

Comparing EKS #1 with EKS's 2 through 100,000,000, date: 08-26-2014, time: 09:49:25:
Key numbers matching 0 times were found 60,584,643 times.
Key numbers matching 1 times were found 30,417,944 times.
Key numbers matching 2 times were found 7,577,292 times.
Key numbers matching 3 times were found 1,251,176 times.
Key numbers matching 4 times were found 152,629 times.
Key numbers matching 5 times were found 15,015 times.
Key numbers matching 6 times were found 1,215 times.
Key numbers matching 7 times were found 82 times.
Key numbers matching 8 times were found 3 times.

Comparing EKS #2 with EKS's 3 through 100,000,000, date: 08-26-2014, time: 11:18:01:
Key numbers matching 0 times were found 60,604,502 times.
Key numbers matching 1 times were found 30,408,283 times.
Key numbers matching 2 times were found 7,570,353 times.
Key numbers matching 3 times were found 1,247,186 times.
Key numbers matching 4 times were found 153,409 times.
Key numbers matching 5 times were found 14,967 times.
Key numbers matching 6 times were found 1,202 times.
Key numbers matching 7 times were found 90 times.
Key numbers matching 8 times were found 6 times.

As you can see by the date/time stamp between the two, it took approximately 1 ½ hours to perform the comparisons of all numbers in the remaining **EKS** streams for each **EKS** tested. As you can see, over 90% of the remaining streams for each tested **EKS** had less than 2 numbers identical. The most important statistic of the first 100 streams in this file is that NO stream within the 100 million stream collection equaled ANY of the first 100 streams, providing evidence of this design's ability to create **non-repeating EKS** streams can be produced with this system in spite of a fixed key being used. The testing of the entire file would obviously take years to complete, but the author feels that further testing would be just as revealing. If there was ANY weakness in the pseudo-random algorithms for pointer advancements, this examination of the first 100 **EKS** streams would be sufficient to show that it would not be able to produce what it needs to produce.

Obviously, that chance, no matter how slight (1 chance in 8,000,000,000,000,000,000) still exists, hence the stage 2 transposition engine to literally shred the output of this Vernam Two engine.

Even if it should ever be exposed that two streams had every pointer (both Vernam 2 and Transposition) identical, reverse engineering of the streams and key used would result in total failure. This is because of the Algebraic Law prohibiting the solving of a 4-unknown single equation for 1 value for each unknown and the impossible variety of transposition keys to perform its Engine 2 operation with no ability to eliminate any of the keys produced. The lack of knowledge of the makeup of the Midstream between the two engines would also complicate any effort to attack this system.

For comparison, I had my non-pseudo-random number generator (that has passed the NIST standard for pseudo-random numbers) generate another block of 12.8 billion numbers from 0 to 255, the same quantity as the 100 million **EKS**'s generated by my app. The following results were obtained:

Comparing key stream #1 with key streams 2 through 100,000,000, date: 09-06-2014, time: 03:30:43:
Key numbers matching 0 times were found 60,591,110 times.
Key numbers matching 1 times were found 30,421,265 times.
Key numbers matching 2 times were found 7,570,745 times.
Key numbers matching 3 times were found 1,247,623 times.
Key numbers matching 4 times were found 152,911 times.
Key numbers matching 5 times were found 14,984 times.
Key numbers matching 6 times were found 1,252 times.
Key numbers matching 7 times were found 102 times.
Key numbers matching 8 times were found 6 times.

Comparing key stream #2 with key streams 3 through 100,000,000, date: 09-06-2014, time: 04:30:35:
Key numbers matching 0 times were found 60,593,444 times.
Key numbers matching 1 times were found 30,417,389 times.
Key numbers matching 2 times were found 7,573,638 times.
Key numbers matching 3 times were found 1,246,538 times.
Key numbers matching 4 times were found 152,943 times.
Key numbers matching 5 times were found 14,752 times.
Key numbers matching 6 times were found 1,208 times.
Key numbers matching 7 times were found 80 times.
Key numbers matching 8 times were found 4 times.
Key numbers matching 9 times were found 1 times.

Notice there is no entry 'Key numbers matching 256 times were found 'n' times' that would indicate 'n' match(es) found. The entire run showing all 100 comparisons is available for your examination; please request and I will be happy to send it to you. Notice that the results are virtually identical with the tests on the output of my application on the previous page that provides a stream of 100 million non-repeating pseudo-random numbers to use in the Vernam Two encryption engine from a fixed key of only 8 Mbytes expanded from a key of 2,048 bytes.

To test the non-repeatability within and between the 4 – 2-Megabyte key segments, tests were performed to find duplicate key streams of 4 or more numbers anywhere within and between the key segments. The 'IF statements per discovery' shows the number of checks that were performed that failed to find a duplicate string segment of 4 numbers or more for each one that was found.

Here are the results of those tests:

For testing key set 1:

Testing of key groups 1 and 1:
Final results - IF's executed to get this = 2,208,174,250,439, IF statements per discovery = 4,460,958,081
Strings of 4 found = 495, strings of 5 found = 0, strings of 6 found = 0
Took 2 hours 24 minutes and 51.3457 seconds

Testing of key groups 1 and 2:
Final results - IF's executed to get this = 4,416,348,501,328, IF statements per discovery = 4,538,898,768
Strings of 4 found = 972, strings of 5 found = 1, strings of 6 found = 0
Took 4 hours 48 minutes and 52.86719 seconds

Testing of key groups 1 and 3:
Final results - IF's executed to get this = 4,416,348,912,485, IF statements per discovery = 4,198,050,297
Strings of 4 found = 1,048, strings of 5 found = 4, strings of 6 found = 0
Took 4 hours 52 minutes and 31.69922 seconds

Testing of key groups 1 and 4:
Final results - IF's executed to get this = 4,416,348,648,659, IF statements per discovery = 4,182,148,341
Strings of 4 found = 1,047, strings of 5 found = 9, strings of 6 found = 0
Took 4 hours 54 minutes and 56.61719 seconds

Testing of key groups 2 and 2:
Final results - IF's executed to get this = 2,208,174,315,401, IF statements per discovery = 4,658,595,602
Strings of 4 found = 474, strings of 5 found = 0, strings of 6 found = 0
Took 2 hours 25 minutes and 59.42188 seconds

Testing of key groups 2 and 3:
Final results - IF's executed to get this = 4,416,348,439,279, IF statements per discovery = 4,359,672,694
Strings of 4 found = 1,010, strings of 5 found = 2, strings of 6 found = 1
Took 4 hours 48 minutes and 35.40625 seconds

Testing of key groups 2 and 4:
Final results - IF's executed to get this = 4,416,348,614,298, IF statements per discovery = 4,483,602,654
Strings of 4 found = 981, strings of 5 found = 4, strings of 6 found = 0
Took 4 hours 48 minutes and 40.61914 seconds

Testing of key groups 3 and 3:
Final results - IF's executed to get this = 2,208,174,456,809, IF statements per discovery = 4,279,407,862
Strings of 4 found = 516, strings of 5 found = 0, strings of 6 found = 0
Took 2 hours 25 minutes and 28.92969 seconds

Testing of key groups 3 and 4:
Final results - IF's executed to get this = 4,416,348,537,255, IF statements per discovery = 4,198,049,940
Strings of 4 found = 1,045, strings of 5 found = 7, strings of 6 found = 0
Took 4 hours 55 minutes and 40.63672 seconds

Testing of key groups 4 and 4:
Final results - IF's executed to get this = 2,208,174,304,324, IF statements per discovery = 4,469,988,470
Strings of 4 found = 493, strings of 5 found = 1, strings of 6 found = 0
Took 2 hours 26 minutes and 45.12891 seconds

For testing key set 2:

Testing of key groups 1 and 1:
Final results - IF's executed to get this = 2,208,174,239,890, IF statements per discovery = 4,104,413,085
Strings of 4 found = 535, strings of 5 found = 3, strings of 6 found = 0
Took 2 hours 20 minutes and 32.06445 seconds

Testing of key groups 1 and 2:
Final results - IF's executed to get this = 4,416,348,496,619, IF statements per discovery = 4,170,300,752
Strings of 4 found = 1,056, strings of 5 found = 3, strings of 6 found = 0
Took 4 hours 38 minutes and 54.71875 seconds

Testing of key groups 1 and 3:
Final results - IF's executed to get this = 4,416,348,723,234, IF statements per discovery = 3,975,111,362
Strings of 4 found = 1,105, strings of 5 found = 6, strings of 6 found = 0
Took 4 hours 43 minutes and 52.08594 seconds

Testing of key groups 1 and 4:
Final results - IF's executed to get this = 4,416,348,646,111, IF statements per discovery = 4,334,002,596
Strings of 4 found = 1,018, strings of 5 found = 1, strings of 6 found = 0
Took 4 hours 46 minutes and 51.33984 seconds

Testing of key groups 2 and 2:
Final results - IF's executed to get this = 2,208,174,164,657, IF statements per discovery = 4,488,158,871
Strings of 4 found = 492, strings of 5 found = 0, strings of 6 found = 0
Took 2 hours 23 minutes and 27.85938 seconds

Testing of key groups 2 and 3:
Final results - IF's executed to get this = 4,416,348,600,611, IF statements per discovery = 4,108,231,256
Strings of 4 found = 1,075, strings of 5 found = 0, strings of 6 found = 0
Took 4 hours 38 minutes and 25.14063 seconds

Testing of key groups 2 and 4:
Final results - IF's executed to get this = 4,416,348,631,182, IF statements per discovery = 4,291,883,995
Strings of 4 found = 1,022, strings of 5 found = 7, strings of 6 found = 0
Took 4 hours 38 minutes and 56.69922 seconds

Testing of key groups 3 and 3:
Final results - IF's executed to get this = 2,208,174,296,605, IF statements per discovery = 4,230,218,959
Strings of 4 found = 522, strings of 5 found = 0, strings of 6 found = 0
Took 2 hours 20 minutes and 20.10742 seconds

Testing of key groups 3 and 4:
Final results - IF's executed to get this = 4,416,348,650,991, IF statements per discovery = 4,104,413,244
Strings of 4 found = 1,074, strings of 5 found = 2, strings of 6 found = 0
Took 4 hours 45 minutes and 46.76367 seconds

Testing of key groups 4 and 4:
Final results - IF's executed to get this = 2,208,174,244,743, IF statements per discovery = 4,355,373,263
Strings of 4 found = 506, strings of 5 found = 1, strings of 6 found = 0
Took 2 hours 23 minutes and 22.43359 seconds

Appendix G

An actual encryption sequence showing the 4 register pointer sequences is on this page and the next 5 pages. On the next 5 pages are outputs of the sequence experienced by changing just the order of the three starting pointers. The 5 starting pointers for each of the 5 pages are on the bottom of this page for comparison.

```
Block # 1 >>> Pointer 1 =   499,880, Pointer 2 = 2,212,661, Pointer 3 = 4,652,065, Pointer 4 = 6,626,798
Block # 2 >>> Pointer 1 = 1,957,997, Pointer 2 = 2,989,792, Pointer 3 = 4,222,877, Pointer 4 = 7,157,527
Block # 3 >>> Pointer 1 = 1,218,116, Pointer 2 = 2,397,078, Pointer 3 = 5,654,162, Pointer 4 = 7,996,355
Block # 4 >>> Pointer 1 =    40,719, Pointer 2 = 3,096,991, Pointer 3 = 6,230,336, Pointer 4 = 8,139,025
Block # 5 >>> Pointer 1 =   351,673, Pointer 2 = 3,769,949, Pointer 3 = 6,142,853, Pointer 4 = 6,768,819
Block # 6 >>> Pointer 1 =   463,182, Pointer 2 = 3,074,065, Pointer 3 = 5,329,127, Pointer 4 = 8,076,557
Block # 7 >>> Pointer 1 =   494,129, Pointer 2 = 3,262,602, Pointer 3 = 4,862,919, Pointer 4 = 8,332,387
Block # 8 >>> Pointer 1 = 1,166,741, Pointer 2 = 3,502,797, Pointer 3 = 5,085,041, Pointer 4 = 7,490,035
Block # 9 >>> Pointer 1 = 1,946,439, Pointer 2 = 2,571,955, Pointer 3 = 5,458,237, Pointer 4 = 7,814,989
Block #10 >>> Pointer 1 = 1,077,056, Pointer 2 = 2,117,999, Pointer 3 = 5,194,320, Pointer 4 = 7,206,374
Block #11 >>> Pointer 1 = 1,879,490, Pointer 2 = 2,244,013, Pointer 3 = 5,096,508, Pointer 4 = 6,411,111
Block #12 >>> Pointer 1 = 1,921,957, Pointer 2 = 2,489,939, Pointer 3 = 5,482,493, Pointer 4 = 6,788,336
Block #13 >>> Pointer 1 = 1,701,005, Pointer 2 = 2,988,788, Pointer 3 = 5,541,785, Pointer 4 = 7,307,822
Block #14 >>> Pointer 1 = 1,880,331, Pointer 2 = 2,858,417, Pointer 3 = 5,311,900, Pointer 4 = 8,326,846
Block #15 >>> Pointer 1 = 1,253,734, Pointer 2 = 2,544,673, Pointer 3 = 4,286,675, Pointer 4 = 6,878,347
Block #16 >>> Pointer 1 =   558,724, Pointer 2 = 2,419,078, Pointer 3 = 4,622,056, Pointer 4 = 7,443,549
Block #17 >>> Pointer 1 =   337,105, Pointer 2 = 3,221,028, Pointer 3 = 4,510,501, Pointer 4 = 6,684,170
Block #18 >>> Pointer 1 =   931,930, Pointer 2 = 3,804,984, Pointer 3 = 5,790,734, Pointer 4 = 7,154,859
Block #19 >>> Pointer 1 =   689,647, Pointer 2 = 3,107,717, Pointer 3 = 4,583,786, Pointer 4 = 7,201,981
Block #20 >>> Pointer 1 =   921,293, Pointer 2 = 2,993,934, Pointer 3 = 5,164,974, Pointer 4 = 6,738,993
Block #21 >>> Pointer 1 =   500,622, Pointer 2 = 3,016,867, Pointer 3 = 4,427,783, Pointer 4 = 7,173,563
Block #22 >>> Pointer 1 =   358,471, Pointer 2 = 2,565,752, Pointer 3 = 5,785,893, Pointer 4 = 6,924,255
Block #23 >>> Pointer 1 = 2,038,282, Pointer 2 = 2,801,690, Pointer 3 = 5,901,503, Pointer 4 = 6,897,760
Block #24 >>> Pointer 1 =   353,721, Pointer 2 = 3,786,341, Pointer 3 = 4,570,053, Pointer 4 = 7,997,136
Block #25 >>> Pointer 1 =   164,137, Pointer 2 = 2,704,257, Pointer 3 = 4,270,914, Pointer 4 = 7,806,182
Block #26 >>> Pointer 1 = 1,488,848, Pointer 2 = 3,184,567, Pointer 3 = 5,755,542, Pointer 4 = 7,026,025
Block #27 >>> Pointer 1 =   889,068, Pointer 2 = 2,928,272, Pointer 3 = 5,303,981, Pointer 4 = 7,969,328
Block #28 >>> Pointer 1 =   774,576, Pointer 2 = 3,173,585, Pointer 3 = 5,354,091, Pointer 4 = 6,731,448
Block #29 >>> Pointer 1 =   124,088, Pointer 2 = 3,678,948, Pointer 3 = 4,504,097, Pointer 4 = 6,799,669
Block #30 >>> Pointer 1 =   950,040, Pointer 2 = 3,690,367, Pointer 3 = 6,232,654, Pointer 4 = 7,442,739
Block #31 >>> Pointer 1 =   925,844, Pointer 2 = 3,419,936, Pointer 3 = 4,232,750, Pointer 4 = 7,789,787
Block #32 >>> Pointer 1 = 1,670,504, Pointer 2 = 2,660,989, Pointer 3 = 6,122,137, Pointer 4 = 8,016,525
Block #33 >>> Pointer 1 =   149,943, Pointer 2 = 3,638,089, Pointer 3 = 4,831,618, Pointer 4 = 7,414,469
Block #34 >>> Pointer 1 = 1,553,443, Pointer 2 = 2,332,852, Pointer 3 = 5,514,647, Pointer 4 = 8,207,373
Block #35 >>> Pointer 1 =   470,431, Pointer 2 = 4,171,821, Pointer 3 = 5,087,655, Pointer 4 = 8,326,906
Block #36 >>> Pointer 1 =    92,105, Pointer 2 = 2,703,975, Pointer 3 = 4,705,089, Pointer 4 = 6,486,716
Block #37 >>> Pointer 1 = 1,282,297, Pointer 2 = 3,781,776, Pointer 3 = 5,307,315, Pointer 4 = 6,925,089
Block #38 >>> Pointer 1 = 1,664,582, Pointer 2 = 2,513,510, Pointer 3 = 4,606,041, Pointer 4 = 7,461,141
Block #39 >>> Pointer 1 = 1,696,195, Pointer 2 = 2,325,217, Pointer 3 = 4,310,393, Pointer 4 = 7,953,515
Block #40 >>> Pointer 1 = 1,463,121, Pointer 2 = 3,217,235, Pointer 3 = 5,460,758, Pointer 4 = 7,383,878
Block #41 >>> Pointer 1 = 1,286,070, Pointer 2 = 3,601,567, Pointer 3 = 6,273,267, Pointer 4 = 6,748,193
Block #42 >>> Pointer 1 = 1,357,259, Pointer 2 = 2,823,605, Pointer 3 = 5,623,060, Pointer 4 = 6,548,284
Block #43 >>> Pointer 1 = 1,960,735, Pointer 2 = 4,161,259, Pointer 3 = 4,923,773, Pointer 4 = 6,315,805
Block #44 >>> Pointer 1 = 1,108,518, Pointer 2 = 2,511,338, Pointer 3 = 4,859,984, Pointer 4 = 8,329,312
Block #45 >>> Pointer 1 =   762,151, Pointer 2 = 2,583,713, Pointer 3 = 4,270,443, Pointer 4 = 7,239,942
Block #46 >>> Pointer 1 = 1,859,170, Pointer 2 = 2,235,742, Pointer 3 = 6,186,012, Pointer 4 = 8,133,886
Block #47 >>> Pointer 1 = 1,384,496, Pointer 2 = 3,192,352, Pointer 3 = 4,207,285, Pointer 4 = 7,993,754
Block #48 >>> Pointer 1 = 1,892,156, Pointer 2 = 3,947,999, Pointer 3 = 6,241,852, Pointer 4 = 7,219,197
Block #49 >>> Pointer 1 =   126,746, Pointer 2 = 3,801,839, Pointer 3 = 5,184,513, Pointer 4 = 7,941,885
Block #50 >>> Pointer 1 =   980,045, Pointer 2 = 2,977,780, Pointer 3 = 5,525,358, Pointer 4 = 6,325,482
Block #51 >>> Pointer 1 =   890,106, Pointer 2 = 3,813,012, Pointer 3 = 5,569,581, Pointer 4 = 7,114,872
Block #52 >>> Pointer 1 =   254,099, Pointer 2 = 3,400,928, Pointer 3 = 4,232,675, Pointer 4 = 7,744,685
Block #53 >>> Pointer 1 = 1,995,790, Pointer 2 = 3,022,708, Pointer 3 = 5,509,150, Pointer 4 = 8,207,599
Block #54 >>> Pointer 1 =   956,632, Pointer 2 = 3,674,008, Pointer 3 = 5,822,990, Pointer 4 = 8,349,616
Block #55 >>> Pointer 1 = 1,418,069, Pointer 2 = 3,503,779, Pointer 3 = 4,413,301, Pointer 4 = 8,263,381
Block #56 >>> Pointer 1 =   164,348, Pointer 2 = 3,965,825, Pointer 3 = 4,324,994, Pointer 4 = 6,382,045
Block #57 >>> Pointer 1 = 1,242,832, Pointer 2 = 3,183,606, Pointer 3 = 5,690,002, Pointer 4 = 7,118,502
Block #58 >>> Pointer 1 =   233,886, Pointer 2 = 4,097,169, Pointer 3 = 5,349,507, Pointer 4 = 7,908,009
Block #59 >>> Pointer 1 = 1,623,864, Pointer 2 = 3,676,615, Pointer 3 = 4,667,928, Pointer 4 = 6,464,721
Block #60 >>> Pointer 1 =   278,321, Pointer 2 = 3,237,183, Pointer 3 = 6,239,076, Pointer 4 = 6,658,911

Block # 1 >>> Pointer 1 =   499,880, Pointer 2 = 2,212,661, Pointer 3 = 4,652,065, Pointer 4 = 6,626,798
Block # 1 >>> Pointer 1 =   499,880, Pointer 2 = 2,554,657, Pointer 3 = 4,310,069, Pointer 4 = 8,028,974
Block # 1 >>> Pointer 1 =   115,253, Pointer 2 = 2,597,288, Pointer 3 = 4,652,065, Pointer 4 = 7,378,275
Block # 1 >>> Pointer 1 =   115,253, Pointer 2 = 2,554,657, Pointer 3 = 4,694,696, Pointer 4 = 6,308,533
Block # 1 >>> Pointer 1 =   457,249, Pointer 2 = 2,212,661, Pointer 3 = 4,694,696, Pointer 4 = 7,787,663
Block # 1 >>> Pointer 1 =   457,249, Pointer 2 = 2,597,288, Pointer 3 = 4,310,069, Pointer 4 = 7,989,657
```

```
Block # 1 >>> Pointer 1 =   499,880, Pointer 2 = 2,554,657, Pointer 3 = 4,310,069, Pointer 4 = 8,028,974
Block # 2 >>> Pointer 1 = 1,237,042, Pointer 2 = 3,322,848, Pointer 3 = 4,207,794, Pointer 4 = 7,714,792
Block # 3 >>> Pointer 1 =    81,888, Pointer 2 = 2,122,303, Pointer 3 = 6,283,873, Pointer 4 = 6,415,666
Block # 4 >>> Pointer 1 = 1,763,108, Pointer 2 = 2,423,783, Pointer 3 = 4,663,034, Pointer 4 = 8,048,607
Block # 5 >>> Pointer 1 = 1,799,260, Pointer 2 = 3,988,596, Pointer 3 = 5,529,307, Pointer 4 = 7,126,002
Block # 6 >>> Pointer 1 =   114,249, Pointer 2 = 2,712,254, Pointer 3 = 6,179,681, Pointer 4 = 6,368,199
Block # 7 >>> Pointer 1 = 1,545,784, Pointer 2 = 3,733,654, Pointer 3 = 5,651,191, Pointer 4 = 6,766,580
Block # 8 >>> Pointer 1 = 1,626,420, Pointer 2 = 3,463,633, Pointer 3 = 5,322,456, Pointer 4 = 8,162,683
Block # 9 >>> Pointer 1 =   762,479, Pointer 2 = 3,099,810, Pointer 3 = 4,354,379, Pointer 4 = 6,950,739
Block #10 >>> Pointer 1 = 1,299,338, Pointer 2 = 2,798,803, Pointer 3 = 5,475,507, Pointer 4 = 6,367,845
Block #11 >>> Pointer 1 =   457,704, Pointer 2 = 2,672,635, Pointer 3 = 6,023,878, Pointer 4 = 6,567,981
Block #12 >>> Pointer 1 = 1,705,460, Pointer 2 = 3,447,557, Pointer 3 = 4,585,114, Pointer 4 = 7,471,997
Block #13 >>> Pointer 1 = 1,632,374, Pointer 2 = 3,561,960, Pointer 3 = 4,749,400, Pointer 4 = 7,352,973
Block #14 >>> Pointer 1 = 1,168,524, Pointer 2 = 2,921,172, Pointer 3 = 5,541,521, Pointer 4 = 7,731,229
Block #15 >>> Pointer 1 =   863,924, Pointer 2 = 3,460,654, Pointer 3 = 5,158,605, Pointer 4 = 8,230,507
Block #16 >>> Pointer 1 = 1,537,932, Pointer 2 = 2,930,807, Pointer 3 = 5,738,167, Pointer 4 = 6,651,320
Block #17 >>> Pointer 1 = 1,105,514, Pointer 2 = 2,789,854, Pointer 3 = 6,188,176, Pointer 4 = 6,740,077
Block #18 >>> Pointer 1 =   546,021, Pointer 2 = 2,459,988, Pointer 3 = 5,564,296, Pointer 4 = 8,381,358
Block #19 >>> Pointer 1 = 1,925,336, Pointer 2 = 3,735,136, Pointer 3 = 4,250,365, Pointer 4 = 7,836,200
Block #20 >>> Pointer 1 =   569,658, Pointer 2 = 3,803,569, Pointer 3 = 5,323,784, Pointer 4 = 8,055,658
Block #21 >>> Pointer 1 =    70,038, Pointer 2 = 3,588,625, Pointer 3 = 5,347,521, Pointer 4 = 6,317,026
Block #22 >>> Pointer 1 =   643,932, Pointer 2 = 3,975,891, Pointer 3 = 5,463,721, Pointer 4 = 6,506,869
Block #23 >>> Pointer 1 = 1,088,380, Pointer 2 = 3,936,667, Pointer 3 = 5,996,048, Pointer 4 = 8,203,437
Block #24 >>> Pointer 1 = 2,057,591, Pointer 2 = 3,653,733, Pointer 3 = 4,553,151, Pointer 4 = 7,982,401
Block #25 >>> Pointer 1 = 1,482,705, Pointer 2 = 3,266,463, Pointer 3 = 6,280,150, Pointer 4 = 7,005,478
Block #26 >>> Pointer 1 =   333,996, Pointer 2 = 2,934,296, Pointer 3 = 5,811,909, Pointer 4 = 7,420,078
Block #27 >>> Pointer 1 = 1,597,337, Pointer 2 = 3,799,391, Pointer 3 = 6,265,848, Pointer 4 = 8,216,803
Block #28 >>> Pointer 1 =   997,023, Pointer 2 = 4,173,878, Pointer 3 = 5,677,487, Pointer 4 = 7,782,162
Block #29 >>> Pointer 1 =   694,937, Pointer 2 = 3,763,098, Pointer 3 = 5,938,026, Pointer 4 = 8,263,474
Block #30 >>> Pointer 1 =   714,821, Pointer 2 = 2,477,032, Pointer 3 = 4,736,970, Pointer 4 = 6,530,876
Block #31 >>> Pointer 1 = 2,001,844, Pointer 2 = 3,432,331, Pointer 3 = 4,961,886, Pointer 4 = 6,520,801
Block #32 >>> Pointer 1 = 1,627,203, Pointer 2 = 2,357,716, Pointer 3 = 5,522,936, Pointer 4 = 6,747,562
Block #33 >>> Pointer 1 = 1,291,330, Pointer 2 = 2,233,524, Pointer 3 = 5,522,451, Pointer 4 = 6,306,979
Block #34 >>> Pointer 1 =   349,489, Pointer 2 = 3,212,885, Pointer 3 = 5,583,621, Pointer 4 = 8,202,762
Block #35 >>> Pointer 1 =    46,908, Pointer 2 = 3,973,559, Pointer 3 = 5,717,664, Pointer 4 = 8,059,601
Block #36 >>> Pointer 1 = 1,465,414, Pointer 2 = 2,520,924, Pointer 3 = 6,047,862, Pointer 4 = 6,964,695
Block #37 >>> Pointer 1 = 1,203,353, Pointer 2 = 3,748,700, Pointer 3 = 6,069,042, Pointer 4 = 6,316,548
Block #38 >>> Pointer 1 =   567,739, Pointer 2 = 3,901,865, Pointer 3 = 4,832,648, Pointer 4 = 7,919,868
Block #39 >>> Pointer 1 = 1,029,818, Pointer 2 = 3,805,366, Pointer 3 = 5,684,239, Pointer 4 = 8,094,613
Block #40 >>> Pointer 1 = 1,789,034, Pointer 2 = 2,784,332, Pointer 3 = 5,008,507, Pointer 4 = 8,128,670
Block #41 >>> Pointer 1 =   822,186, Pointer 2 = 2,772,363, Pointer 3 = 4,959,308, Pointer 4 = 6,653,592
Block #42 >>> Pointer 1 =   876,902, Pointer 2 = 2,518,625, Pointer 3 = 4,286,573, Pointer 4 = 7,738,228
Block #43 >>> Pointer 1 =   778,045, Pointer 2 = 4,017,375, Pointer 3 = 6,242,123, Pointer 4 = 7,075,749
Block #44 >>> Pointer 1 = 1,647,151, Pointer 2 = 3,086,882, Pointer 3 = 4,337,945, Pointer 4 = 8,320,454
Block #45 >>> Pointer 1 =   388,638, Pointer 2 = 4,065,006, Pointer 3 = 5,120,005, Pointer 4 = 6,437,492
Block #46 >>> Pointer 1 =   656,383, Pointer 2 = 4,156,163, Pointer 3 = 4,456,810, Pointer 4 = 6,731,303
Block #47 >>> Pointer 1 = 1,434,076, Pointer 2 = 3,995,361, Pointer 3 = 4,316,917, Pointer 4 = 7,473,190
Block #48 >>> Pointer 1 = 1,362,871, Pointer 2 = 3,618,251, Pointer 3 = 4,962,612, Pointer 4 = 7,042,773
Block #49 >>> Pointer 1 =   668,760, Pointer 2 = 3,713,844, Pointer 3 = 5,528,234, Pointer 4 = 6,827,770
Block #50 >>> Pointer 1 =   526,509, Pointer 2 = 3,008,776, Pointer 3 = 4,763,624, Pointer 4 = 7,057,859
Block #51 >>> Pointer 1 =   662,525, Pointer 2 = 4,033,307, Pointer 3 = 6,029,194, Pointer 4 = 7,804,022
Block #52 >>> Pointer 1 =    45,348, Pointer 2 = 2,359,729, Pointer 3 = 5,318,144, Pointer 4 = 6,857,359
Block #53 >>> Pointer 1 =    27,579, Pointer 2 = 3,924,331, Pointer 3 = 4,963,808, Pointer 4 = 7,541,108
Block #54 >>> Pointer 1 =   786,928, Pointer 2 = 3,149,057, Pointer 3 = 4,321,804, Pointer 4 = 6,756,971
Block #55 >>> Pointer 1 =   761,184, Pointer 2 = 2,149,533, Pointer 3 = 6,120,139, Pointer 4 = 8,384,486
Block #56 >>> Pointer 1 = 1,944,728, Pointer 2 = 3,702,444, Pointer 3 = 5,020,285, Pointer 4 = 6,435,641
Block #57 >>> Pointer 1 =   996,854, Pointer 2 = 3,567,669, Pointer 3 = 5,634,159, Pointer 4 = 6,640,933
Block #58 >>> Pointer 1 = 1,853,683, Pointer 2 = 3,358,024, Pointer 3 = 4,781,372, Pointer 4 = 7,189,033
Block #59 >>> Pointer 1 = 1,283,783, Pointer 2 = 2,585,750, Pointer 3 = 5,687,667, Pointer 4 = 7,219,290
Block #60 >>> Pointer 1 =   425,374, Pointer 2 = 4,122,493, Pointer 3 = 6,136,038, Pointer 4 = 7,088,451
```

If you do a side-by-side comparison of these 6 pages, you will see that not only is the advancement algorithm value-dependent of each pointer but position-dependent as well. On page 25 at the bottom is a copy of the first block pointers for each case on each page. You can see that only the position of the first 3 pointers was changed to each of the 6 possible arrangements of those values. Register 4 shows that it too is dependent upon the order of the numbers, not just their values, as it too shows non-consistency in spite of the values not changing, only their position. This algorithm for pointer advancement accomplishes the goal of providing a thorough mix of numbers for each pointer's range to do the best job possible of making certain a non-repeating key stream is fed to the Vernam Engine.

```
Block # 1 >>> Pointer 1 =   115,253, Pointer 2 = 2,597,288, Pointer 3 = 4,652,065, Pointer 4 = 7,378,275
Block # 2 >>> Pointer 1 = 1,965,523, Pointer 2 = 3,383,037, Pointer 3 = 6,149,533, Pointer 4 = 7,309,550
Block # 3 >>> Pointer 1 = 1,600,097, Pointer 2 = 2,177,386, Pointer 3 = 4,875,064, Pointer 4 = 7,552,947
Block # 4 >>> Pointer 1 = 2,038,716, Pointer 2 = 3,948,571, Pointer 3 = 6,012,479, Pointer 4 = 8,161,980
Block # 5 >>> Pointer 1 =    50,336, Pointer 2 = 2,138,564, Pointer 3 = 4,498,080, Pointer 4 = 6,978,236
Block # 6 >>> Pointer 1 = 1,166,275, Pointer 2 = 2,314,955, Pointer 3 = 4,965,713, Pointer 4 = 8,334,984
Block # 7 >>> Pointer 1 =   682,019, Pointer 2 = 2,296,680, Pointer 3 = 4,728,074, Pointer 4 = 6,565,130
Block # 8 >>> Pointer 1 =    34,810, Pointer 2 = 3,850,631, Pointer 3 = 4,717,760, Pointer 4 = 7,656,846
Block # 9 >>> Pointer 1 =   239,965, Pointer 2 = 4,039,849, Pointer 3 = 4,808,611, Pointer 4 = 8,136,410
Block #10 >>> Pointer 1 = 1,454,490, Pointer 2 = 3,831,601, Pointer 3 = 5,348,470, Pointer 4 = 7,493,653
Block #11 >>> Pointer 1 =   953,558, Pointer 2 = 3,592,076, Pointer 3 = 5,036,238, Pointer 4 = 6,393,631
Block #12 >>> Pointer 1 = 1,642,975, Pointer 2 = 4,192,785, Pointer 3 = 5,366,265, Pointer 4 = 6,370,593
Block #13 >>> Pointer 1 =   102,290, Pointer 2 = 3,277,455, Pointer 3 = 5,215,233, Pointer 4 = 7,723,065
Block #14 >>> Pointer 1 = 1,976,944, Pointer 2 = 3,194,666, Pointer 3 = 4,879,038, Pointer 4 = 7,037,090
Block #15 >>> Pointer 1 =   354,874, Pointer 2 = 3,802,730, Pointer 3 = 4,865,029, Pointer 4 = 7,394,202
Block #16 >>> Pointer 1 =   372,943, Pointer 2 = 3,122,864, Pointer 3 = 5,296,549, Pointer 4 = 7,974,419
Block #17 >>> Pointer 1 = 2,019,020, Pointer 2 = 2,924,494, Pointer 3 = 5,164,702, Pointer 4 = 7,074,266
Block #18 >>> Pointer 1 = 1,815,315, Pointer 2 = 3,390,643, Pointer 3 = 5,445,051, Pointer 4 = 6,915,191
Block #19 >>> Pointer 1 =   575,932, Pointer 2 = 3,959,241, Pointer 3 = 4,832,872, Pointer 4 = 7,534,120
Block #20 >>> Pointer 1 = 1,803,994, Pointer 2 = 3,849,350, Pointer 3 = 4,644,027, Pointer 4 = 8,187,355
Block #21 >>> Pointer 1 =   598,185, Pointer 2 = 2,697,760, Pointer 3 = 4,238,121, Pointer 4 = 7,399,635
Block #22 >>> Pointer 1 =   810,418, Pointer 2 = 3,329,373, Pointer 3 = 6,141,132, Pointer 4 = 6,548,371
Block #23 >>> Pointer 1 =   446,837, Pointer 2 = 3,524,561, Pointer 3 = 5,339,078, Pointer 4 = 8,351,834
Block #24 >>> Pointer 1 = 1,541,067, Pointer 2 = 2,857,091, Pointer 3 = 4,443,543, Pointer 4 = 6,827,048
Block #25 >>> Pointer 1 =   467,393, Pointer 2 = 2,213,921, Pointer 3 = 4,309,959, Pointer 4 = 6,508,842
Block #26 >>> Pointer 1 = 1,702,235, Pointer 2 = 3,922,681, Pointer 3 = 5,856,729, Pointer 4 = 6,427,268
Block #27 >>> Pointer 1 =   647,443, Pointer 2 = 3,361,505, Pointer 3 = 4,265,289, Pointer 4 = 6,860,698
Block #28 >>> Pointer 1 = 1,847,215, Pointer 2 = 3,087,663, Pointer 3 = 5,222,684, Pointer 4 = 6,561,916
Block #29 >>> Pointer 1 = 2,096,657, Pointer 2 = 3,219,710, Pointer 3 = 6,165,279, Pointer 4 = 8,240,028
Block #30 >>> Pointer 1 = 1,208,358, Pointer 2 = 2,528,112, Pointer 3 = 5,253,266, Pointer 4 = 7,694,654
Block #31 >>> Pointer 1 =   815,498, Pointer 2 = 2,813,297, Pointer 3 = 5,344,492, Pointer 4 = 6,516,098
Block #32 >>> Pointer 1 =   592,495, Pointer 2 = 3,082,762, Pointer 3 = 4,878,601, Pointer 4 = 7,165,924
Block #33 >>> Pointer 1 = 1,106,326, Pointer 2 = 3,551,713, Pointer 3 = 4,298,800, Pointer 4 = 8,333,078
Block #34 >>> Pointer 1 =   779,491, Pointer 2 = 2,354,404, Pointer 3 = 4,515,307, Pointer 4 = 6,538,866
Block #35 >>> Pointer 1 =   816,326, Pointer 2 = 2,551,156, Pointer 3 = 5,556,460, Pointer 4 = 7,397,628
Block #36 >>> Pointer 1 = 1,133,863, Pointer 2 = 2,568,781, Pointer 3 = 5,056,817, Pointer 4 = 8,140,199
Block #37 >>> Pointer 1 =   511,907, Pointer 2 = 2,296,015, Pointer 3 = 5,219,591, Pointer 4 = 7,635,604
Block #38 >>> Pointer 1 =   575,081, Pointer 2 = 2,689,478, Pointer 3 = 4,614,920, Pointer 4 = 8,150,251
Block #39 >>> Pointer 1 = 1,877,477, Pointer 2 = 2,489,765, Pointer 3 = 4,581,372, Pointer 4 = 7,963,983
Block #40 >>> Pointer 1 = 1,475,326, Pointer 2 = 4,102,018, Pointer 3 = 4,391,062, Pointer 4 = 7,230,100
Block #41 >>> Pointer 1 =   692,159, Pointer 2 = 4,131,727, Pointer 3 = 5,226,762, Pointer 4 = 7,303,939
Block #42 >>> Pointer 1 = 1,205,571, Pointer 2 = 2,298,725, Pointer 3 = 4,539,666, Pointer 4 = 7,053,481
Block #43 >>> Pointer 1 =    21,367, Pointer 2 = 3,653,971, Pointer 3 = 5,470,656, Pointer 4 = 6,457,807
Block #44 >>> Pointer 1 =   262,769, Pointer 2 = 3,253,506, Pointer 3 = 4,354,980, Pointer 4 = 6,342,540
Block #45 >>> Pointer 1 = 1,529,707, Pointer 2 = 2,873,431, Pointer 3 = 5,729,751, Pointer 4 = 6,453,587
Block #46 >>> Pointer 1 = 1,489,513, Pointer 2 = 2,734,010, Pointer 3 = 5,925,814, Pointer 4 = 6,752,686
Block #47 >>> Pointer 1 = 1,685,958, Pointer 2 = 2,554,553, Pointer 3 = 5,884,153, Pointer 4 = 7,944,361
Block #48 >>> Pointer 1 = 1,733,008, Pointer 2 = 3,160,945, Pointer 3 = 5,345,850, Pointer 4 = 7,656,232
Block #49 >>> Pointer 1 =   199,385, Pointer 2 = 3,785,738, Pointer 3 = 4,905,923, Pointer 4 = 8,341,682
Block #50 >>> Pointer 1 =   526,250, Pointer 2 = 2,812,167, Pointer 3 = 4,697,320, Pointer 4 = 7,514,133
Block #51 >>> Pointer 1 = 1,267,537, Pointer 2 = 3,216,471, Pointer 3 = 5,722,899, Pointer 4 = 6,786,413
Block #52 >>> Pointer 1 = 1,977,114, Pointer 2 = 3,866,411, Pointer 3 = 4,922,622, Pointer 4 = 7,408,509
Block #53 >>> Pointer 1 =   224,863, Pointer 2 = 4,129,902, Pointer 3 = 5,136,643, Pointer 4 = 7,074,061
Block #54 >>> Pointer 1 =   905,684, Pointer 2 = 3,428,049, Pointer 3 = 5,363,277, Pointer 4 = 6,355,038
Block #55 >>> Pointer 1 = 1,595,765, Pointer 2 = 3,537,241, Pointer 3 = 5,863,416, Pointer 4 = 6,842,120
Block #56 >>> Pointer 1 = 1,432,642, Pointer 2 = 2,234,076, Pointer 3 = 6,046,741, Pointer 4 = 7,915,428
Block #57 >>> Pointer 1 =   280,780, Pointer 2 = 2,934,088, Pointer 3 = 4,771,524, Pointer 4 = 6,699,322
Block #58 >>> Pointer 1 = 1,249,511, Pointer 2 = 2,593,808, Pointer 3 = 5,302,675, Pointer 4 = 8,105,143
Block #59 >>> Pointer 1 = 2,011,590, Pointer 2 = 2,531,249, Pointer 3 = 5,359,774, Pointer 4 = 6,948,833
Block #60 >>> Pointer 1 = 1,613,586, Pointer 2 = 3,332,511, Pointer 3 = 6,231,256, Pointer 4 = 6,706,204
```

The total number of **EKS**'s available is $2{,}097{,}152^4 = 1.9343 \times 10^{25}$. If each possible **EKS** were only 6 inches long and placed end-to-end, the length would be equal to over 9.8 trillion round trips between the earth and the sun. It is the author's belief that the **EKS** production capability of this design is adequate for the life of this product.

```
Block # 1 >>> Pointer 1 =   115,253, Pointer 2 = 2,554,657, Pointer 3 = 4,694,696, Pointer 4 = 6,308,533
Block # 2 >>> Pointer 1 = 2,030,898, Pointer 2 = 3,317,757, Pointer 3 = 6,108,318, Pointer 4 = 6,945,991
Block # 3 >>> Pointer 1 =   766,312, Pointer 2 = 2,649,265, Pointer 3 = 5,335,659, Pointer 4 = 8,105,746
Block # 4 >>> Pointer 1 = 1,856,884, Pointer 2 = 3,439,957, Pointer 3 = 5,600,892, Pointer 4 = 8,037,787
Block # 5 >>> Pointer 1 =   422,550, Pointer 2 = 3,549,042, Pointer 3 = 5,412,902, Pointer 4 = 7,127,172
Block # 6 >>> Pointer 1 =    29,144, Pointer 2 = 3,694,961, Pointer 3 = 5,364,320, Pointer 4 = 7,348,260
Block # 7 >>> Pointer 1 =   312,440, Pointer 2 = 3,704,260, Pointer 3 = 4,487,812, Pointer 4 = 6,735,574
Block # 8 >>> Pointer 1 = 1,758,806, Pointer 2 = 3,570,646, Pointer 3 = 5,658,746, Pointer 4 = 8,152,670
Block # 9 >>> Pointer 1 =   801,777, Pointer 2 = 3,271,995, Pointer 3 = 6,026,220, Pointer 4 = 8,350,002
Block #10 >>> Pointer 1 =    88,393, Pointer 2 = 2,712,153, Pointer 3 = 5,852,001, Pointer 4 = 7,868,639
Block #11 >>> Pointer 1 =   185,036, Pointer 2 = 2,933,714, Pointer 3 = 5,426,882, Pointer 4 = 8,359,167
Block #12 >>> Pointer 1 =   187,934, Pointer 2 = 4,105,182, Pointer 3 = 6,168,738, Pointer 4 = 7,979,553
Block #13 >>> Pointer 1 = 1,960,418, Pointer 2 = 2,236,137, Pointer 3 = 4,842,525, Pointer 4 = 6,859,061
Block #14 >>> Pointer 1 =   481,982, Pointer 2 = 4,114,522, Pointer 3 = 5,947,591, Pointer 4 = 6,880,896
Block #15 >>> Pointer 1 = 1,965,882, Pointer 2 = 3,866,367, Pointer 3 = 6,241,533, Pointer 4 = 8,062,221
Block #16 >>> Pointer 1 =   946,783, Pointer 2 = 4,140,914, Pointer 3 = 5,398,830, Pointer 4 = 7,784,513
Block #17 >>> Pointer 1 =   222,400, Pointer 2 = 2,098,276, Pointer 3 = 4,506,115, Pointer 4 = 6,963,012
Block #18 >>> Pointer 1 =   750,475, Pointer 2 = 2,821,235, Pointer 3 = 5,475,595, Pointer 4 = 7,306,526
Block #19 >>> Pointer 1 = 1,696,917, Pointer 2 = 3,529,444, Pointer 3 = 4,495,321, Pointer 4 = 6,551,424
Block #20 >>> Pointer 1 = 1,063,408, Pointer 2 = 3,174,713, Pointer 3 = 5,894,768, Pointer 4 = 6,675,019
Block #21 >>> Pointer 1 = 1,300,004, Pointer 2 = 2,381,014, Pointer 3 = 5,645,907, Pointer 4 = 6,407,480
Block #22 >>> Pointer 1 = 1,880,527, Pointer 2 = 3,120,561, Pointer 3 = 5,362,076, Pointer 4 = 7,544,772
Block #23 >>> Pointer 1 =   831,439, Pointer 2 = 3,083,695, Pointer 3 = 5,230,860, Pointer 4 = 7,117,828
Block #24 >>> Pointer 1 = 1,886,913, Pointer 2 = 2,170,314, Pointer 3 = 4,899,612, Pointer 4 = 6,424,849
Block #25 >>> Pointer 1 =   887,371, Pointer 2 = 2,821,770, Pointer 3 = 4,869,389, Pointer 4 = 8,202,641
Block #26 >>> Pointer 1 =   392,436, Pointer 2 = 3,458,812, Pointer 3 = 6,092,485, Pointer 4 = 6,876,640
Block #27 >>> Pointer 1 =   787,396, Pointer 2 = 2,419,971, Pointer 3 = 4,441,836, Pointer 4 = 7,550,269
Block #28 >>> Pointer 1 =   833,889, Pointer 2 = 2,174,393, Pointer 3 = 5,858,092, Pointer 4 = 8,160,120
Block #29 >>> Pointer 1 =   154,137, Pointer 2 = 3,752,794, Pointer 3 = 5,905,218, Pointer 4 = 8,084,275
Block #30 >>> Pointer 1 = 1,811,327, Pointer 2 = 4,160,675, Pointer 3 = 4,424,059, Pointer 4 = 7,493,210
Block #31 >>> Pointer 1 =   636,176, Pointer 2 = 3,189,429, Pointer 3 = 5,575,337, Pointer 4 = 7,411,994
Block #32 >>> Pointer 1 = 1,791,256, Pointer 2 = 3,710,037, Pointer 3 = 5,577,371, Pointer 4 = 7,703,556
Block #33 >>> Pointer 1 =   816,922, Pointer 2 = 3,837,303, Pointer 3 = 5,708,940, Pointer 4 = 7,819,246
Block #34 >>> Pointer 1 =   611,767, Pointer 2 = 3,656,277, Pointer 3 = 5,618,121, Pointer 4 = 8,355,877
Block #35 >>> Pointer 1 = 1,538,956, Pointer 2 = 2,939,003, Pointer 3 = 6,000,343, Pointer 4 = 6,363,234
Block #36 >>> Pointer 1 = 1,746,610, Pointer 2 = 3,341,222, Pointer 3 = 4,633,850, Pointer 4 = 7,542,482
Block #37 >>> Pointer 1 = 1,443,302, Pointer 2 = 2,496,261, Pointer 3 = 4,581,398, Pointer 4 = 6,844,554
Block #38 >>> Pointer 1 = 1,500,287, Pointer 2 = 4,143,076, Pointer 3 = 4,489,526, Pointer 4 = 7,458,079
Block #39 >>> Pointer 1 =   124,553, Pointer 2 = 2,736,870, Pointer 3 = 4,623,297, Pointer 4 = 7,023,127
Block #40 >>> Pointer 1 = 1,065,189, Pointer 2 = 2,462,016, Pointer 3 = 4,253,584, Pointer 4 = 7,626,883
Block #41 >>> Pointer 1 = 1,325,003, Pointer 2 = 2,839,863, Pointer 3 = 5,754,196, Pointer 4 = 7,601,762
Block #42 >>> Pointer 1 = 1,619,632, Pointer 2 = 3,152,310, Pointer 3 = 5,681,688, Pointer 4 = 7,408,616
Block #43 >>> Pointer 1 = 1,914,077, Pointer 2 = 4,005,428, Pointer 3 = 5,562,141, Pointer 4 = 6,421,327
Block #44 >>> Pointer 1 = 1,730,840, Pointer 2 = 3,677,033, Pointer 3 = 4,790,810, Pointer 4 = 6,937,395
Block #45 >>> Pointer 1 =   807,508, Pointer 2 = 3,435,858, Pointer 3 = 5,396,076, Pointer 4 = 7,458,775
Block #46 >>> Pointer 1 =   332,244, Pointer 2 = 3,417,617, Pointer 3 = 5,363,237, Pointer 4 = 8,386,158
Block #47 >>> Pointer 1 =   283,495, Pointer 2 = 2,598,227, Pointer 3 = 5,466,532, Pointer 4 = 7,056,076
Block #48 >>> Pointer 1 = 1,768,756, Pointer 2 = 3,447,805, Pointer 3 = 6,108,826, Pointer 4 = 6,984,880
Block #49 >>> Pointer 1 = 1,076,489, Pointer 2 = 2,740,589, Pointer 3 = 5,049,296, Pointer 4 = 8,297,706
Block #50 >>> Pointer 1 =   699,069, Pointer 2 = 4,049,834, Pointer 3 = 4,898,762, Pointer 4 = 7,992,979
Block #51 >>> Pointer 1 =   721,049, Pointer 2 = 3,755,008, Pointer 3 = 4,234,059, Pointer 4 = 7,694,860
Block #52 >>> Pointer 1 =   869,585, Pointer 2 = 3,223,108, Pointer 3 = 4,510,509, Pointer 4 = 8,047,175
Block #53 >>> Pointer 1 =   312,359, Pointer 2 = 2,606,532, Pointer 3 = 4,467,140, Pointer 4 = 8,304,882
Block #54 >>> Pointer 1 =   832,059, Pointer 2 = 3,927,474, Pointer 3 = 5,389,804, Pointer 4 = 6,531,963
Block #55 >>> Pointer 1 = 1,917,196, Pointer 2 = 2,948,673, Pointer 3 = 4,263,677, Pointer 4 = 7,634,526
Block #56 >>> Pointer 1 =   483,809, Pointer 2 = 2,222,177, Pointer 3 = 4,318,183, Pointer 4 = 8,250,237
Block #57 >>> Pointer 1 =   114,520, Pointer 2 = 3,687,103, Pointer 3 = 6,249,025, Pointer 4 = 6,696,720
Block #58 >>> Pointer 1 = 1,697,300, Pointer 2 = 3,439,334, Pointer 3 = 4,593,273, Pointer 4 = 8,076,399
Block #59 >>> Pointer 1 =   964,076, Pointer 2 = 2,887,605, Pointer 3 = 5,631,502, Pointer 4 = 7,115,997
Block #60 >>> Pointer 1 =   110,600, Pointer 2 = 2,646,704, Pointer 3 = 5,245,537, Pointer 4 = 6,301,994
```

Refering back to Appendix A, if changing only one pointer by the amount shown can cause such a drastic change in the **EKS**, then this display of how the pointer advancement function operates in a real encryption operation ought to show that the range of **EKS**'s ought to be vast enough to satisfy the requirement for the Vernam Engine that the key be non-repeating.

Block # 1 >>> Pointer 1 = 457,249, Pointer 2 = 2,212,661, Pointer 3 = 4,694,696, Pointer 4 = 7,787,663
Block # 2 >>> Pointer 1 = 2,013,805, Pointer 2 = 2,990,010, Pointer 3 = 5,926,814, Pointer 4 = 8,196,989
Block # 3 >>> Pointer 1 = 1,647,060, Pointer 2 = 3,414,561, Pointer 3 = 4,314,649, Pointer 4 = 6,511,922
Block # 4 >>> Pointer 1 = 1,180,856, Pointer 2 = 3,724,036, Pointer 3 = 4,504,274, Pointer 4 = 8,005,874
Block # 5 >>> Pointer 1 = 477,459, Pointer 2 = 3,401,801, Pointer 3 = 4,789,735, Pointer 4 = 6,368,754
Block # 6 >>> Pointer 1 = 313,956, Pointer 2 = 2,360,778, Pointer 3 = 4,875,780, Pointer 4 = 8,136,827
Block # 7 >>> Pointer 1 = 209,247, Pointer 2 = 4,178,993, Pointer 3 = 5,333,443, Pointer 4 = 7,120,643
Block # 8 >>> Pointer 1 = 791,587, Pointer 2 = 2,305,300, Pointer 3 = 5,514,540, Pointer 4 = 6,324,710
Block # 9 >>> Pointer 1 = 1,251,810, Pointer 2 = 2,241,561, Pointer 3 = 5,891,123, Pointer 4 = 8,058,305
Block #10 >>> Pointer 1 = 2,039,305, Pointer 2 = 2,727,966, Pointer 3 = 6,163,359, Pointer 4 = 6,346,195
Block #11 >>> Pointer 1 = 2,024,700, Pointer 2 = 3,973,092, Pointer 3 = 4,521,630, Pointer 4 = 7,091,672
Block #12 >>> Pointer 1 = 1,839,249, Pointer 2 = 3,243,280, Pointer 3 = 5,280,636, Pointer 4 = 7,014,366
Block #13 >>> Pointer 1 = 1,310,480, Pointer 2 = 3,208,447, Pointer 3 = 6,230,771, Pointer 4 = 8,105,791
Block #14 >>> Pointer 1 = 117,228, Pointer 2 = 2,908,873, Pointer 3 = 4,845,153, Pointer 4 = 7,613,406
Block #15 >>> Pointer 1 = 1,310,982, Pointer 2 = 2,503,937, Pointer 3 = 4,261,940, Pointer 4 = 6,750,953
Block #16 >>> Pointer 1 = 792,920, Pointer 2 = 3,697,945, Pointer 3 = 5,855,852, Pointer 4 = 8,317,170
Block #17 >>> Pointer 1 = 489,611, Pointer 2 = 2,820,216, Pointer 3 = 5,803,271, Pointer 4 = 7,911,922
Block #18 >>> Pointer 1 = 833,383, Pointer 2 = 2,575,799, Pointer 3 = 5,728,588, Pointer 4 = 6,496,033
Block #19 >>> Pointer 1 = 50,186, Pointer 2 = 2,752,964, Pointer 3 = 4,459,520, Pointer 4 = 7,925,132
Block #20 >>> Pointer 1 = 611,047, Pointer 2 = 2,615,890, Pointer 3 = 5,433,833, Pointer 4 = 8,287,873
Block #21 >>> Pointer 1 = 319,737, Pointer 2 = 3,745,248, Pointer 3 = 4,258,596, Pointer 4 = 7,739,654
Block #22 >>> Pointer 1 = 898,571, Pointer 2 = 2,854,582, Pointer 3 = 5,639,565, Pointer 4 = 6,836,578
Block #23 >>> Pointer 1 = 213,390, Pointer 2 = 3,023,937, Pointer 3 = 4,296,739, Pointer 4 = 7,706,428
Block #24 >>> Pointer 1 = 734,524, Pointer 2 = 3,984,437, Pointer 3 = 5,586,635, Pointer 4 = 6,428,688
Block #25 >>> Pointer 1 = 1,621,822, Pointer 2 = 4,102,591, Pointer 3 = 6,242,456, Pointer 4 = 8,234,394
Block #26 >>> Pointer 1 = 699,735, Pointer 2 = 3,648,429, Pointer 3 = 5,069,226, Pointer 4 = 6,908,285
Block #27 >>> Pointer 1 = 896,442, Pointer 2 = 3,804,845, Pointer 3 = 5,094,413, Pointer 4 = 8,144,556
Block #28 >>> Pointer 1 = 1,810,343, Pointer 2 = 2,571,423, Pointer 3 = 6,269,243, Pointer 4 = 7,509,475
Block #29 >>> Pointer 1 = 621,804, Pointer 2 = 2,910,844, Pointer 3 = 6,090,345, Pointer 4 = 6,309,886
Block #30 >>> Pointer 1 = 481,771, Pointer 2 = 2,852,953, Pointer 3 = 5,893,511, Pointer 4 = 7,518,721
Block #31 >>> Pointer 1 = 1,525,478, Pointer 2 = 2,529,350, Pointer 3 = 4,647,063, Pointer 4 = 7,865,405
Block #32 >>> Pointer 1 = 390,507, Pointer 2 = 2,844,405, Pointer 3 = 5,598,565, Pointer 4 = 6,853,608
Block #33 >>> Pointer 1 = 1,057,332, Pointer 2 = 3,412,258, Pointer 3 = 4,339,216, Pointer 4 = 7,984,828
Block #34 >>> Pointer 1 = 1,006,362, Pointer 2 = 3,854,427, Pointer 3 = 5,971,151, Pointer 4 = 6,797,897
Block #35 >>> Pointer 1 = 1,713,055, Pointer 2 = 4,160,291, Pointer 3 = 4,432,250, Pointer 4 = 7,795,562
Block #36 >>> Pointer 1 = 735,486, Pointer 2 = 4,082,744, Pointer 3 = 5,832,779, Pointer 4 = 6,438,438
Block #37 >>> Pointer 1 = 1,811,161, Pointer 2 = 3,775,650, Pointer 3 = 4,381,595, Pointer 4 = 7,393,476
Block #38 >>> Pointer 1 = 1,038,507, Pointer 2 = 2,838,744, Pointer 3 = 5,811,535, Pointer 4 = 6,325,242
Block #39 >>> Pointer 1 = 992,473, Pointer 2 = 3,772,452, Pointer 3 = 4,512,655, Pointer 4 = 8,254,095
Block #40 >>> Pointer 1 = 2,008,716, Pointer 2 = 2,940,838, Pointer 3 = 4,624,094, Pointer 4 = 6,541,606
Block #41 >>> Pointer 1 = 480,002, Pointer 2 = 2,238,547, Pointer 3 = 5,440,551, Pointer 4 = 7,511,874
Block #42 >>> Pointer 1 = 1,214,963, Pointer 2 = 3,396,489, Pointer 3 = 4,847,058, Pointer 4 = 7,994,878
Block #43 >>> Pointer 1 = 2,062,652, Pointer 2 = 3,965,049, Pointer 3 = 5,848,703, Pointer 4 = 8,316,892
Block #44 >>> Pointer 1 = 721,054, Pointer 2 = 4,107,264, Pointer 3 = 4,235,435, Pointer 4 = 7,890,785
Block #45 >>> Pointer 1 = 1,849,350, Pointer 2 = 2,522,424, Pointer 3 = 5,769,340, Pointer 4 = 7,050,758
Block #46 >>> Pointer 1 = 611,644, Pointer 2 = 3,951,189, Pointer 3 = 5,586,505, Pointer 4 = 8,259,408
Block #47 >>> Pointer 1 = 1,484,817, Pointer 2 = 3,250,088, Pointer 3 = 4,723,606, Pointer 4 = 6,472,672
Block #48 >>> Pointer 1 = 815,510, Pointer 2 = 3,558,769, Pointer 3 = 5,347,404, Pointer 4 = 7,445,718
Block #49 >>> Pointer 1 = 1,284,811, Pointer 2 = 2,872,474, Pointer 3 = 5,950,931, Pointer 4 = 6,484,754
Block #50 >>> Pointer 1 = 1,154,259, Pointer 2 = 3,363,484, Pointer 3 = 6,083,921, Pointer 4 = 7,132,557
Block #51 >>> Pointer 1 = 2,033,310, Pointer 2 = 4,071,430, Pointer 3 = 4,628,511, Pointer 4 = 8,269,079
Block #52 >>> Pointer 1 = 270,151, Pointer 2 = 2,557,215, Pointer 3 = 6,244,612, Pointer 4 = 6,637,349
Block #53 >>> Pointer 1 = 1,551,446, Pointer 2 = 3,569,836, Pointer 3 = 5,003,383, Pointer 4 = 8,384,422
Block #54 >>> Pointer 1 = 1,807,893, Pointer 2 = 3,513,494, Pointer 3 = 5,642,139, Pointer 4 = 7,227,665
Block #55 >>> Pointer 1 = 1,715,841, Pointer 2 = 2,186,030, Pointer 3 = 5,145,434, Pointer 4 = 7,480,436
Block #56 >>> Pointer 1 = 1,143,476, Pointer 2 = 3,428,978, Pointer 3 = 5,420,625, Pointer 4 = 7,227,167
Block #57 >>> Pointer 1 = 82,285, Pointer 2 = 3,007,041, Pointer 3 = 4,288,481, Pointer 4 = 7,515,138
Block #58 >>> Pointer 1 = 1,208,283, Pointer 2 = 3,879,791, Pointer 3 = 5,233,970, Pointer 4 = 6,670,043
Block #59 >>> Pointer 1 = 1,560,079, Pointer 2 = 3,111,118, Pointer 3 = 5,116,279, Pointer 4 = 6,360,448
Block #60 >>> Pointer 1 = 1,918,746, Pointer 2 = 3,817,031, Pointer 3 = 4,660,285, Pointer 4 = 6,413,402

Block # 1 >>> Pointer 1 = 457,249, Pointer 2 = 2,597,288, Pointer 3 = 4,310,069, Pointer 4 = 7,989,657
Block # 2 >>> Pointer 1 = 1,227,475, Pointer 2 = 3,388,346, Pointer 3 = 5,952,946, Pointer 4 = 8,221,737
Block # 3 >>> Pointer 1 = 315,888, Pointer 2 = 3,188,177, Pointer 3 = 5,370,532, Pointer 4 = 6,329,348
Block # 4 >>> Pointer 1 = 1,545,316, Pointer 2 = 2,390,164, Pointer 3 = 5,531,255, Pointer 4 = 6,312,767
Block # 5 >>> Pointer 1 = 318,816, Pointer 2 = 2,139,613, Pointer 3 = 6,120,100, Pointer 4 = 7,577,677
Block # 6 >>> Pointer 1 = 1,702,915, Pointer 2 = 2,300,668, Pointer 3 = 6,030,617, Pointer 4 = 7,402,255
Block # 7 >>> Pointer 1 = 1,617,302, Pointer 2 = 3,578,285, Pointer 3 = 5,085,336, Pointer 4 = 7,823,157
Block # 8 >>> Pointer 1 = 21,694, Pointer 2 = 4,120,916, Pointer 3 = 5,554,400, Pointer 4 = 7,894,823
Block # 9 >>> Pointer 1 = 1,493,473, Pointer 2 = 2,185,161, Pointer 3 = 4,842,326, Pointer 4 = 6,963,713
Block #10 >>> Pointer 1 = 1,743,602, Pointer 2 = 3,283,866, Pointer 3 = 5,960,730, Pointer 4 = 6,755,093
Block #11 >>> Pointer 1 = 394,089, Pointer 2 = 2,729,731, Pointer 3 = 4,418,470, Pointer 4 = 8,301,033
Block #12 >>> Pointer 1 = 1,526,415, Pointer 2 = 3,119,178, Pointer 3 = 4,886,935, Pointer 4 = 7,524,456
Block #13 >>> Pointer 1 = 260,181, Pointer 2 = 3,499,256, Pointer 3 = 5,789,539, Pointer 4 = 8,258,460
Block #14 >>> Pointer 1 = 236,181, Pointer 2 = 3,515,546, Pointer 3 = 5,937,059, Pointer 4 = 8,175,511
Block #15 >>> Pointer 1 = 432,956, Pointer 2 = 3,958,683, Pointer 3 = 5,979,750, Pointer 4 = 8,369,447
Block #16 >>> Pointer 1 = 50,190, Pointer 2 = 3,064,260, Pointer 3 = 4,460,736, Pointer 4 = 8,261,427
Block #17 >>> Pointer 1 = 1,593,040, Pointer 2 = 3,193,166, Pointer 3 = 5,165,752, Pointer 4 = 6,812,637
Block #18 >>> Pointer 1 = 316,188, Pointer 2 = 3,990,995, Pointer 3 = 5,447,396, Pointer 4 = 7,161,520
Block #19 >>> Pointer 1 = 1,376,442, Pointer 2 = 3,814,912, Pointer 3 = 4,242,485, Pointer 4 = 7,912,701
Block #20 >>> Pointer 1 = 1,224,253, Pointer 2 = 4,019,118, Pointer 3 = 5,128,018, Pointer 4 = 6,415,434
Block #21 >>> Pointer 1 = 1,173,386, Pointer 2 = 2,757,351, Pointer 3 = 4,688,913, Pointer 4 = 7,388,606
Block #22 >>> Pointer 1 = 1,971,367, Pointer 2 = 2,604,820, Pointer 3 = 5,548,478, Pointer 4 = 7,119,438
Block #23 >>> Pointer 1 = 451,860, Pointer 2 = 3,426,277, Pointer 3 = 4,527,686, Pointer 4 = 7,115,115
Block #24 >>> Pointer 1 = 2,087,854, Pointer 2 = 3,055,835, Pointer 3 = 6,008,991, Pointer 4 = 6,413,468
Block #25 >>> Pointer 1 = 1,642,927, Pointer 2 = 3,119,633, Pointer 3 = 5,353,881, Pointer 4 = 6,394,308
Block #26 >>> Pointer 1 = 1,492,186, Pointer 2 = 3,848,132, Pointer 3 = 4,512,950, Pointer 4 = 7,640,097
Block #27 >>> Pointer 1 = 1,553,023, Pointer 2 = 4,151,474, Pointer 3 = 5,407,063, Pointer 4 = 7,025,568
Block #28 >>> Pointer 1 = 1,254,622, Pointer 2 = 4,109,348, Pointer 3 = 4,513,971, Pointer 4 = 7,756,619
Block #29 >>> Pointer 1 = 235,902, Pointer 2 = 4,064,409, Pointer 3 = 5,865,475, Pointer 4 = 6,362,080
Block #30 >>> Pointer 1 = 982,706, Pointer 2 = 3,321,854, Pointer 3 = 6,206,638, Pointer 4 = 7,332,644
Block #31 >>> Pointer 1 = 611,939, Pointer 2 = 2,337,366, Pointer 3 = 5,662,121, Pointer 4 = 7,929,449
Block #32 >>> Pointer 1 = 1,044,979, Pointer 2 = 3,354,865, Pointer 3 = 5,371,183, Pointer 4 = 6,600,228
Block #33 >>> Pointer 1 = 899,487, Pointer 2 = 4,148,921, Pointer 3 = 5,873,997, Pointer 4 = 7,511,924
Block #34 >>> Pointer 1 = 1,607,241, Pointer 2 = 2,693,510, Pointer 3 = 4,606,744, Pointer 4 = 6,974,704
Block #35 >>> Pointer 1 = 2,067,133, Pointer 2 = 4,030,602, Pointer 3 = 4,898,687, Pointer 4 = 6,748,929
Block #36 >>> Pointer 1 = 661,173, Pointer 2 = 3,509,014, Pointer 3 = 5,683,082, Pointer 4 = 8,125,994
Block #37 >>> Pointer 1 = 1,042,143, Pointer 2 = 4,157,670, Pointer 3 = 4,645,231, Pointer 4 = 6,605,405
Block #38 >>> Pointer 1 = 1,103,359, Pointer 2 = 4,157,909, Pointer 3 = 5,636,464, Pointer 4 = 8,090,355
Block #39 >>> Pointer 1 = 1,898,301, Pointer 2 = 4,021,751, Pointer 3 = 5,717,852, Pointer 4 = 7,789,222
Block #40 >>> Pointer 1 = 952,634, Pointer 2 = 3,821,449, Pointer 3 = 4,799,566, Pointer 4 = 7,546,447
Block #41 >>> Pointer 1 = 304,794, Pointer 2 = 3,802,534, Pointer 3 = 4,627,460, Pointer 4 = 7,083,124
Block #42 >>> Pointer 1 = 97,831, Pointer 2 = 2,556,542, Pointer 3 = 6,170,881, Pointer 4 = 7,213,933
Block #43 >>> Pointer 1 = 149,445, Pointer 2 = 2,425,671, Pointer 3 = 4,704,002, Pointer 4 = 8,052,770
Block #44 >>> Pointer 1 = 998,080, Pointer 2 = 2,134,074, Pointer 3 = 5,948,047, Pointer 4 = 6,593,179
Block #45 >>> Pointer 1 = 1,530,445, Pointer 2 = 2,971,738, Pointer 3 = 5,918,551, Pointer 4 = 7,118,945
Block #46 >>> Pointer 1 = 2,050,659, Pointer 2 = 2,310,218, Pointer 3 = 4,875,583, Pointer 4 = 7,812,680
Block #47 >>> Pointer 1 = 382,099, Pointer 2 = 3,368,660, Pointer 3 = 5,543,269, Pointer 4 = 7,450,190
Block #48 >>> Pointer 1 = 1,661,846, Pointer 2 = 3,586,651, Pointer 3 = 6,002,873, Pointer 4 = 7,837,281
Block #49 >>> Pointer 1 = 1,565,774, Pointer 2 = 3,061,988, Pointer 3 = 4,477,111, Pointer 4 = 8,091,507
Block #50 >>> Pointer 1 = 1,560,139, Pointer 2 = 2,881,742, Pointer 3 = 5,131,767, Pointer 4 = 8,012,125
Block #51 >>> Pointer 1 = 579,324, Pointer 2 = 3,975,638, Pointer 3 = 5,701,288, Pointer 4 = 6,553,361
Block #52 >>> Pointer 1 = 1,395,188, Pointer 2 = 3,446,345, Pointer 3 = 4,847,253, Pointer 4 = 8,060,725
Block #53 >>> Pointer 1 = 1,628,148, Pointer 2 = 3,455,447, Pointer 3 = 5,764,792, Pointer 4 = 7,060,590
Block #54 >>> Pointer 1 = 1,107,576, Pointer 2 = 3,674,598, Pointer 3 = 4,618,768, Pointer 4 = 7,760,923
Block #55 >>> Pointer 1 = 686,183, Pointer 2 = 2,567,032, Pointer 3 = 5,794,090, Pointer 4 = 7,320,260
Block #56 >>> Pointer 1 = 1,304,571, Pointer 2 = 3,888,359, Pointer 3 = 4,717,907, Pointer 4 = 7,003,334
Block #57 >>> Pointer 1 = 1,313,970, Pointer 2 = 3,331,340, Pointer 3 = 5,027,028, Pointer 4 = 6,327,141
Block #58 >>> Pointer 1 = 889,776, Pointer 2 = 3,190,419, Pointer 3 = 5,485,229, Pointer 4 = 6,973,368
Block #59 >>> Pointer 1 = 844,678, Pointer 2 = 2,526,691, Pointer 3 = 4,425,868, Pointer 4 = 6,399,299
Block #60 >>> Pointer 1 = 1,458,554, Pointer 2 = 3,831,617, Pointer 3 = 4,291,702, Pointer 4 = 8,034,600

A test was run to just produce 2 billion (2,000,000,000) sets of pointers and test to see what the pointer values would be as well as see when all pointer values from 0 to 2,097,151 inclusive would be used (to test the thoroughness of the engine pointer advancement pseudo-algorithm). The following is a sample of what it found. All log files are available for examination if needed – contact the author. Each log file contained over 500 test points where the low 22 bits of the ciphertext block number were all equal to 1 to start the 10-record set, 6 of them are shown below. This will show the unlikelyhood of any two sets of pointers being equal since one of the factors that is incorporated in calculating pointer 4's value is the lower 22 bits of the block number.

```
Block #1 >          P1=[7,902,840], P2=[5,346,543], P3=[4,032,778], P4=[1,346,022], E2P=22,   E2O=109
Block #2 >          P1=[5,377,781], P2=[3,535,628], P3=[849,394  ], P4=[6,822,413], E2P=1267, E2O=159
Block #3 >          P1=[1,287,325], P2=[6,124,964], P3=[6,594,675], P4=[2,420,499], E2P=2950, E2O=156
Block #4 >          P1=[3,094,946], P2=[6,484,536], P3=[1,614,575], P4=[5,411,392], E2P=7044, E2O=289
Block #5 >          P1=[1,897,507], P2=[4,456,180], P3=[3,417,340], P4=[6,349,308], E2P=2119, E2O=68
Block #6 >          P1=[5,265,372], P2=[1,880,149], P3=[7,724,976], P4=[2,927,596], E2P=6186, E2O=69
Block #7 >          P1=[3,466,978], P2=[4,372,197], P3=[385,716  ], P4=[8,047,812], E2P=2507, E2O=300
Block #8 >          P1=[180,510  ], P2=[4,105,153], P3=[6,365,602], P4=[5,574,323], E2P=5523, E2O=68
Block #9 >          P1=[2,896,340], P2=[1,362,992], P3=[5,297,868], P4=[7,565,998], E2P=781,  E2O=89
Block #10 >         P1=[1,981,584], P2=[7,389,500], P3=[3,969,470], P4=[4,262,141], E2P=3043, E2O=254

Block #4,194,313 >  P1=[7,462,363], P2=[5,772,250], P3=[3,856,658], P4=[51,823   ], E2P=3437, E2O=66,  correct = 0
Block #4,194,314 >  P1=[5,451,658], P2=[2,765,873], P3=[887,348  ], P4=[6,600,625], E2P=6477, E2O=141, correct = 0
Block #4,194,315 >  P1=[426,089  ], P2=[4,466,818], P3=[6,317,862], P4=[2,734,529], E2P=1232, E2O=264, correct = 0
Block #4,194,316 >  P1=[2,964,703], P2=[8,351,803], P3=[1,826,672], P4=[5,339,994], E2P=5053, E2O=248, correct = 0
Block #4,194,317 >  P1=[263,623  ], P2=[4,523,525], P3=[2,475,528], P4=[8,279,627], E2P=369,  E2O=130, correct = 0
Block #4,194,318 >  P1=[5,074,229], P2=[1,297,773], P3=[7,026,383], P4=[2,595,607], E2P=4939, E2O=243, correct = 0
Block #4,194,319 >  P1=[4,124,446], P2=[6,103,279], P3=[924,961  ], P4=[7,573,609], E2P=3728, E2O=180, correct = 0
Block #4,194,320 >  P1=[1,011,011], P2=[2,322,541], P3=[7,161,456], P4=[4,984,249], E2P=3517, E2O=244, correct = 0
Block #4,194,321 >  P1=[3,099,336], P2=[569,161  ], P3=[4,836,015], P4=[7,271,163], E2P=1252, E2O=98,  correct = 0
Block #4,194,322 >  P1=[1,679,353], P2=[7,822,496], P3=[4,192,349], P4=[4,572,886], E2P=6141, E2O=191, correct = 0

Block #6,291,457 >  P1=[8,385,271], P2=[5,620,207], P3=[3,143,361], P4=[1,067,877], E2P=4333, E2O=318, correct = 0
Block #6,291,458 >  P1=[4,297,169], P2=[3,031,696], P3=[1,035,845], P4=[7,038,859], E2P=2024, E2O=273, correct = 0
Block #6,291,459 >  P1=[780,341  ], P2=[5,524,970], P3=[6,829,903], P4=[2,373,070], E2P=5833, E2O=195, correct = 0
Block #6,291,460 >  P1=[2,994,434], P2=[6,467,760], P3=[1,049,261], P4=[5,276,273], E2P=171,  E2O=64,  correct = 0
Block #6,291,461 >  P1=[1,769,717], P2=[5,594,370], P3=[2,160,220], P4=[7,838,709], E2P=490,  E2O=280, correct = 0
Block #6,291,462 >  P1=[5,511,209], P2=[521,238  ], P3=[7,744,244], P4=[2,133,508], E2P=3889, E2O=84,  correct = 0
Block #6,291,463 >  P1=[2,880,534], P2=[5,590,516], P3=[1,250,635], P4=[7,398,359], E2P=7365, E2O=215, correct = 0
Block #6,291,464 >  P1=[244,500  ], P2=[3,466,427], P3=[8,067,043], P4=[5,838,822], E2P=2503, E2O=142, correct = 0
Block #6,291,465 >  P1=[3,648,941], P2=[645,040  ], P3=[5,024,727], P4=[6,451,635], E2P=2223, E2O=299, correct = 0
Block #6,291,466 >  P1=[2,055,661], P2=[7,119,455], P3=[4,058,527], P4=[4,641,129], E2P=6951, E2O=309, correct = 0

Block #10,485,769 > P1=[7,309,432], P2=[5,771,653], P3=[2,455,823], P4=[253,478  ], E2P=951,  E2O=65,  correct = 0
Block #10,485,770 > P1=[4,902,934], P2=[3,599,312], P3=[923,371  ], P4=[7,547,698], E2P=2872, E2O=150, correct = 0
Block #10,485,771 > P1=[229,398  ], P2=[5,580,164], P3=[6,297,640], P4=[3,214,079], E2P=1354, E2O=76,  correct = 0
Block #10,485,772 > P1=[3,889,170], P2=[7,028,313], P3=[1,510,204], P4=[4,198,870], E2P=1899, E2O=290, correct = 0
Block #10,485,773 > P1=[655,303  ], P2=[4,426,753], P3=[4,179,083], P4=[7,352,750], E2P=2047, E2O=190, correct = 0
Block #10,485,774 > P1=[5,660,508], P2=[1,873,501], P3=[8,216,470], P4=[4,087,636], E2P=3758, E2O=238, correct = 0
Block #10,485,775 > P1=[2,968,402], P2=[5,230,419], P3=[675,789  ], P4=[7,681,370], E2P=1321, E2O=106, correct = 0
Block #10,485,776 > P1=[26,128   ], P2=[3,195,238], P3=[6,689,728], P4=[5,000,089], E2P=6211, E2O=214, correct = 0
Block #10,485,777 > P1=[3,540,719], P2=[1,013,260], P3=[4,583,798], P4=[6,823,134], E2P=189,  E2O=300, correct = 0
Block #10,485,778 > P1=[2,041,101], P2=[7,201,317], P3=[2,428,639], P4=[5,318,728], E2P=7973, E2O=198, correct = 0

Block #12,582,913 > P1=[7,981,227], P2=[4,946,885], P3=[2,468,985], P4=[1,525,344], E2P=3559, E2O=245, correct = 0
Block #12,582,914 > P1=[4,928,206], P2=[3,001,392], P3=[1,101,259], P4=[7,522,773], E2P=7449, E2O=166, correct = 0
Block #12,582,915 > P1=[1,447,935], P2=[6,068,252], P3=[7,864,214], P4=[3,723,628], E2P=767,  E2O=319, correct = 0
Block #12,582,916 > P1=[3,438,051], P2=[6,362,996], P3=[1,368,340], P4=[5,411,350], E2P=2498, E2O=172, correct = 0
Block #12,582,917 > P1=[1,153,822], P2=[6,189,979], P3=[3,874,675], P4=[7,405,232], E2P=3288, E2O=280, correct = 0
Block #12,582,918 > P1=[5,611,423], P2=[2,037,149], P3=[8,233,493], P4=[3,941,331], E2P=1895, E2O=300, correct = 0
Block #12,582,919 > P1=[3,852,574], P2=[6,151,368], P3=[531,930  ], P4=[6,585,572], E2P=119,  E2O=164, correct = 0
Block #12,582,920 > P1=[577,650  ], P2=[2,714,068], P3=[7,367,785], P4=[6,032,914], E2P=833,  E2O=134, correct = 0
Block #12,582,921 > P1=[3,054,716], P2=[1,855,134], P3=[5,996,112], P4=[6,313,261], E2P=1759, E2O=287, correct = 0
Block #12,582,922 > P1=[95,093   ], P2=[7,603,315], P3=[3,372,293], P4=[5,804,184], E2P=1485, E2O=179, correct = 0

Block #16,777,225 > P1=[6,870,105], P2=[5,884,625], P3=[3,234,505], P4=[1,489,925], E2P=2257, E2O=85,  correct = 0
Block #16,777,226 > P1=[4,905,823], P2=[4,139,993], P3=[1,662,763], P4=[6,747,549], E2P=2917, E2O=235, correct = 0
Block #16,777,227 > P1=[1,532,481], P2=[4,274,534], P3=[6,439,992], P4=[2,177,847], E2P=3780, E2O=82,  correct = 0
Block #16,777,228 > P1=[3,945,832], P2=[6,831,925], P3=[1,337,404], P4=[5,353,386], E2P=2513, E2O=198, correct = 0
Block #16,777,229 > P1=[278,300  ], P2=[5,973,567], P3=[4,135,973], P4=[7,985,854], E2P=2019, E2O=136, correct = 0
Block #16,777,230 > P1=[5,871,470], P2=[776,597  ], P3=[7,696,089], P4=[4,080,056], E2P=6903, E2O=106, correct = 0
Block #16,777,231 > P1=[2,539,253], P2=[5,613,757], P3=[1,963,432], P4=[7,908,093], E2P=1725, E2O=232, correct = 0
Block #16,777,232 > P1=[1,753,810], P2=[3,070,914], P3=[6,476,251], P4=[6,125,204], E2P=3803, E2O=75,  correct = 0
Block #16,777,233 > P1=[2,414,991], P2=[861,402  ], P3=[5,803,812], P4=[6,758,921], E2P=1129, E2O=100, correct = 0
Block #16,777,234 > P1=[1,480,138], P2=[6,887,829], P3=[3,525,142], P4=[4,440,228], E2P=3646, E2O=126, correct = 0
```