US 20040096056A1

(19) **United States**

(12) **Patent Application Publication** (10) **Pub. No.: US 2004/0096056 A1**

Boren et al. (43) **Pub. Date: May 20, 2004**

(54) **METHOD OF ENCRYPTION USING MULTI-KEY PROCESS TO CREATE A VARIABLE-LENGTH KEY**

(76) Inventors: **Stephen Laurence Boren**, Vancouver (CA); **Andre Jacques Brisson**, Vancouver (CA)

Correspondence Address:
OYEN, WIGGS, GREEN & MUTALA
480 - THE STATION
601 WEST CORDOVA STREET
VANCOUVER, BC V6B 1G1 (CA)

(21) Appl. No.: **10/299,847**

(22) Filed: **Nov. 20, 2002**

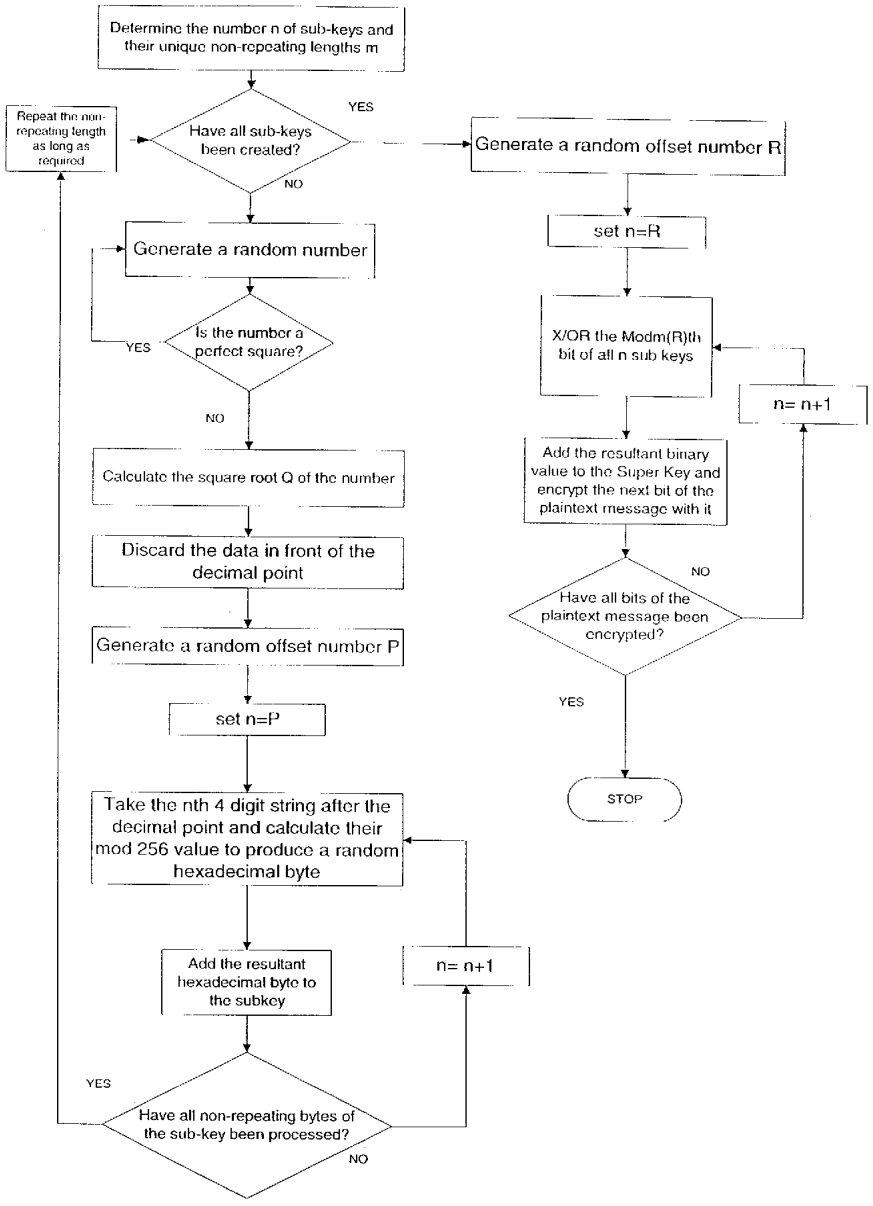
Publication Classification

(51) **Int. Cl.**[7] **H04L 9/00**

(52) **U.S. Cl.** **380/28**

(57) **ABSTRACT**

In symmetric methods of encryption the key should be as long as the plaintext message. Such a key is difficult to generate if the plaintext data to be encrypted is enormous. The present invention provides a method of creating a random key of variable length which may be extremely long. It is generated by consecutively applying sub-keys having shorter non-repeating random lengths.



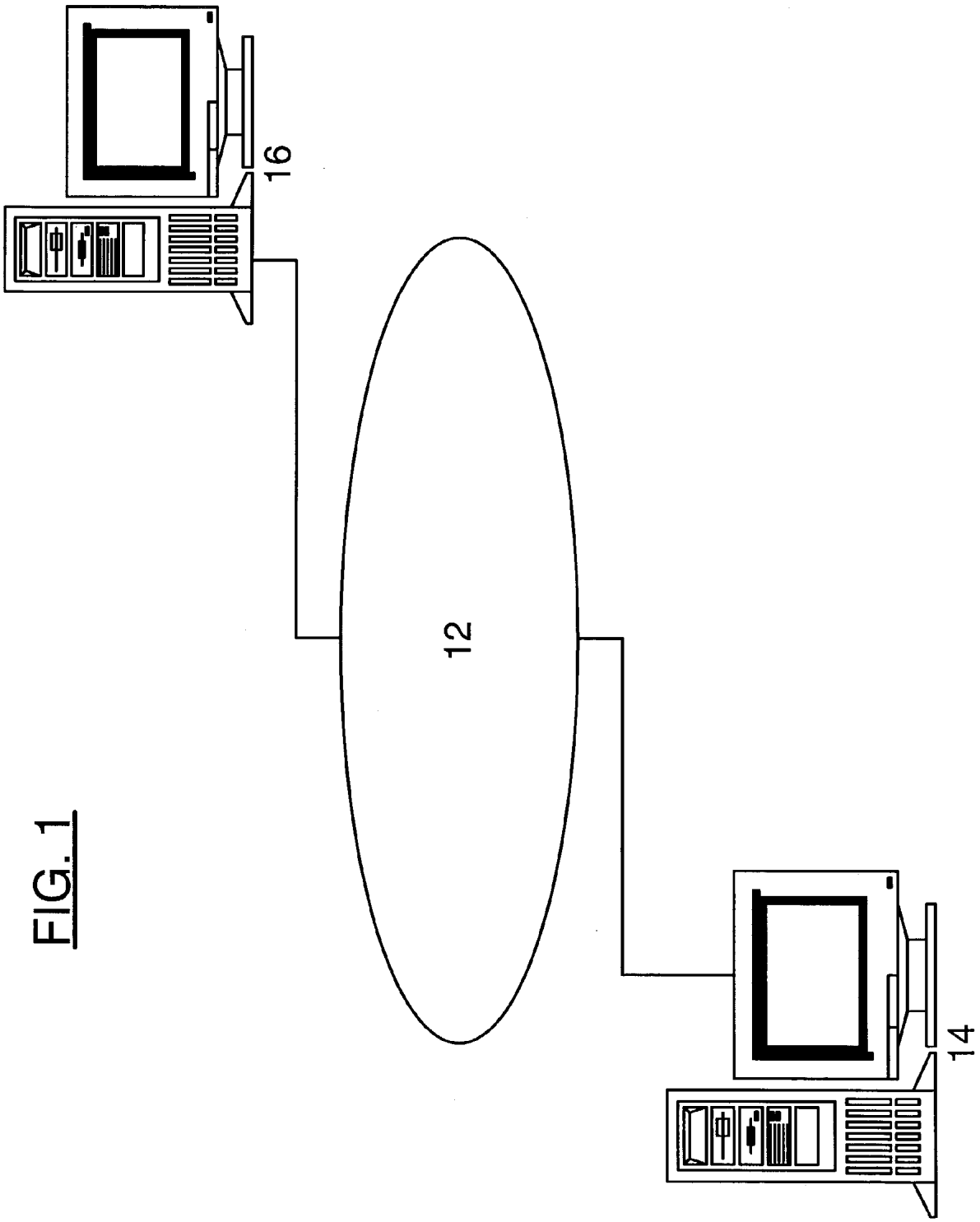


FIG. 1

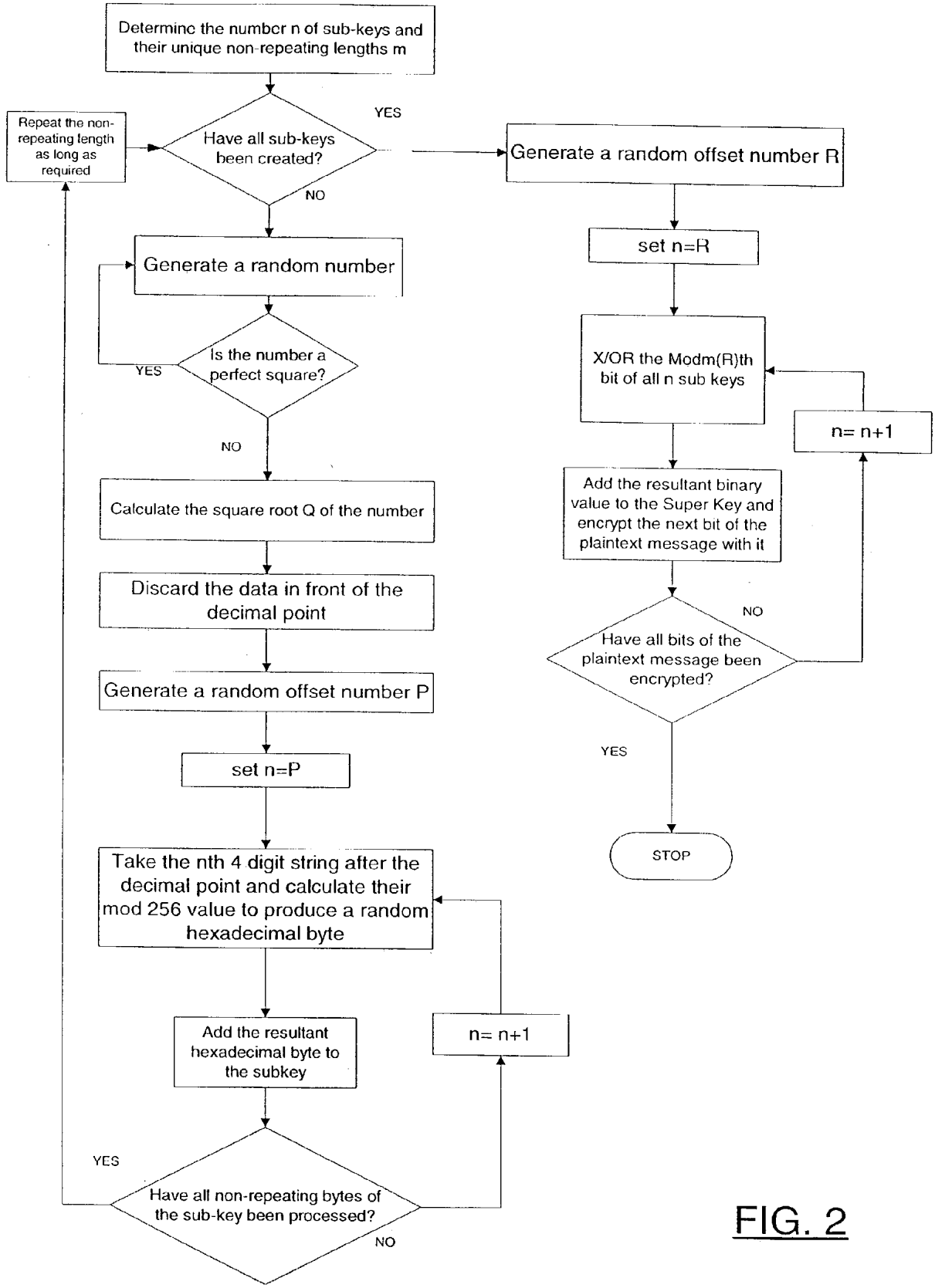

Determine the number n of sub-keys and their unique non-repeating lengths m
Have all sub-keys been created?
YES
NO
Repeat the non-repeating length as long as required
Generate a random number
Is the number a perfect square?
YES
NO
Calculate the square root Q of the number
Discard the data in front of the decimal point
Generate a random offset number P
set n=P
Take the nth 4 digit string after the decimal point and calculate their mod 256 value to produce a random hexadecimal byte
Add the resultant hexadecimal byte to the subkey
n= n+1
Have all non-repeating bytes of the sub-key been processed?
YES
NO
Generate a random offset number R
set n=R
X/OR the Modm(R)th bit of all n sub keys
n= n+1
Add the resultant binary value to the Super Key and encrypt the next bit of the plaintext message with it
Have all bits of the plaintext message been encrypted?
NO
YES
STOP

FIG. 2

METHOD OF ENCRYPTION USING MULTI-KEY PROCESS TO CREATE A VARIABLE-LENGTH KEY

TECHNICAL FIELD

[0001] The invention relates to the field of encryption methods and more particularly to a method for encrypting electronic communications using keys of variable length which may be extremely long.

BACKGROUND ART

[0002] Various methods of encryption to provide secure electronic communications are well known in the art. In symmetric methods of encryption, the sender and the recipient use the same code or key to encrypt and decrypt the message. The only completely secure cipher which cannot possibly be broken or deciphered is the One-Time Pad (OTP). A OTP takes a stream of bits that contains the plaintext message, and a secret random bit-stream of the same length as the plaintext (the key). To encrypt the plaintext with the key, each pair of bits from the key and plaintext is sequentially acted on by the exclusive-or function to obtain the ciphertext bit. The ciphertext cannot be deciphered if the key is truly random and the key is kept secret from an unauthorized party. The problem with this method is that the key should be at least the same length as the message. If a shorter key is used and repeated then the cipher can be broken. In some cases the data which needs to be encrypted is extremely large.

[0003] There is therefore a need for a method of generating a random key, or OTP, which is of variable length and that allows for encryption of very large amounts of data.

DISCLOSURE OF INVENTION

[0004] The present invention therefore provides a method of generating an encryption key having length x, the method comprising the steps of: i) selecting a number n of sub-keys each having a unique non-repeating length m; ii) generating n random numbers, one for each sub-key, each having length m; iii) generating a n+1 st random number R; iv) for each bit whose position in said nth random number is calculated as Modm(R) applying a function to all n bits to generate a binary value; v) concatenating said binary value to the end of the encryption key; and vi) repeating step iv) until the key is x bits in length. Preferably the selected length m of each sub-key is a prime number.

[0005] According to one aspect of the invention, each of the n random numbers is generated by: i) generating a first random number which is not a perfect square; ii) calculating the square root of the first random number; iii) generating a second random number; iv) commencing with a digit whose position in the first random number is calculated based on the second random number, taking finite strings of digits sequentially and converting each finite string into a hexadecimal byte; and vi) concatenating each hexadecimal byte sequentially to the random number until the selected length m of the random number has been reached.

[0006] The invention further provides a computer program product, an article for carrying out the method, and a data processing system for carrying out the method.

BRIEF DESCRIPTION OF DRAWINGS

[0007] In drawings which disclose a preferred embodiment of the invention:

[0008] **FIG. 1** is a schematic illustration of a computer system for carrying out the method of the invention; and

[0009] **FIG. 2** is a flow chart illustrating the method of the invention.

BEST MODE(S) FOR CARRYING OUT THE INVENTION

[0010] **FIG. 2** illustrates by way of a flowchart the method of generating the encryption key of the present invention. In particular an encryption key, a non-repeating key of indefinite length referred to herein as a Super Key, is formed by combining sub-keys. Any number n of sub keys K_n can be specified depending on the application. The greater the number of sub-keys, the greater the length of the non-repeating Super Key. The length of each sub key is a prime number of bytes (preferably with prime numbers larger than 10).

[0011] The first step in the process is to determine how large a Super Key, or cipher, to deploy. The number of sub-keys and the non-repeating length of each sub-key, in bytes, is selected. The sub-keys each have a unique non-repeating length. No two sub-keys are of the same non-repeating length. Preferably the sub-key non-repeating lengths are prime numbers of bytes. The selection may be done by manually entering the number of sub-keys and their prime number non-repeating lengths. Alternatively, the number of keys and their prime number non-repeating lengths is programmed into an application, or a program randomly selects the number of sub-keys and their non-repeating length. For n sub-keys K_n, the non-repeating length of the Super Key will be Size (K_1) X Size (K_2) X Size (K_3) . . . X Size (K_n). For example, assume **10** sub-keys of the following prime number non-repeating lengths are used:

[0012] Sub Key **1**=13 bytes=K_1

[0013] Sub Key **2**=17 bytes=K_2

[0014] Sub Key **3**=19 bytes=K_3

[0015] Sub Key **4**=23 bytes=K_4

[0016] Sub Key **5**=29 bytes=K_5

[0017] Sub Key **6**=31 bytes=K_6

[0018] Sub Key **7**=37 bytes=K_7

[0019] Sub Key **8**=41 bytes=K_8

[0020] Sub Key **9**=43 bytes=K_9

[0021] Sub Key **10**=47 bytes=K_{10}

[0022] The resulting non-repeating Super Key length is 13×17×19×23×29×31×37×41×43×47=266,186,053,068,611 bytes. Thus, using a small number of sub-keys, each of small prime number non-repeating length results in an extremely long non-repeating Super Key. The total definition for the size for the multi-key above is contained in 300 bytes and the header.

[0023] While preferably the non-repeating length of each sub-key is a prime number of bytes, to improve the randomness of the resulting cipher, the method will also work if non-prime number lengths are used, as long as the resulting cipher is very large.

[0024] Each sub-key of the multi-key process may be created as follows. First a random number which is not a perfect square is generated, preferably by a computer random number generator. This serves as a "first seed value" O. Random number generators that are included in the operating systems of most computers are pseudo-random and not very robust. These values, however, are sufficient as a starting point. It is verified that the selected value O is not a perfect square. If it is, then additional random values will be generated until one meets this criterion. A second random number P ("second seed value") is also generated by the computer's random number generator to serve as an offset to be utilized in this process. The square root Q of this first seed value O is calculated, resulting in an irrational number Q (one that extends infinitely after the decimal point since it is not evenly divisible). The resultant string of digits after the decimal point is potentially infinite in length and is highly random. The computer discards the digits in front of the decimal and computes the number Q up to P digits after the decimal. Then, starting at the Pth digit of Q after the decimal point, the computer sequentially selects **4** digits at a time, and calculates the Mod 256 value of the 4 digits. The single resultant random 8-bit byte may be represented in hexadecimal notation. This value is used as the first byte of the sub-key. This process is repeated **4** digits at a time, continuing with the next digits in sequence, until a string of random data equal to the prime number non-repeating length of the sub-key being created is completed. This process is repeated for all the sub keys until the non-repeating length for all the sub keys are created. Each sub-key then is formed by taking the non-repeating string of bytes thus created, and repeating it as often as necessary in combination with the other sub-keys to create the Super Key.

[0025] Once all the sub-keys are created as above, the Super Key (cipher) is created to the length required. This means the Super Key will continue to be created to encrypt the associated data to be encrypted, and continues to be created only until all the data is encrypted. First a random number R ("third seed value", or the starting offset for the Super Key, as opposed to the starting offset P for the number Q) is generated. Starting with any one of the n sub-keys, having length m, the Modm of R is calculated and the Modm(R)th byte of each sub-key is consecutively exclusive-or'd (X/OR'd) with the corresponding Modm(R)th byte of every other sub-key. For example, if R=100, and the length of the first sub-key is 97 bytes, then the 3rd byte of sub-key **1** is selected and X|OR'd with the corresponding bytes of the other remaining sub-keys based on R selected in the same way. The process is repeated until all the selected bytes from each sub-key have been X/OR'd. The resultant binary value is then added to the Super Key. The next, subsequent bytes of sub-key **1** is then X|OR'd with the next byte of Sub key **3** and so on. Again the process is repeated until all the selected bytes from each sub-key have been X/OR'd. The resulting binary value of each function is again added to the Super Key. While the X/OR function is preferred, it will be apparent that other functions can be applied. For example, mathematical functions of addition or subtraction can be used. As each byte of the Super Key is generated, the corresponding byte of the plaintext message is then encrypted with the corresponding byte of the Super Key by the exclusive-or function or some other mathematical function. Once all the bytes of the plaintext message have been encrypted the generation of the Super Key terminates. The encrypted message can then be decrypted applying the inverse of the encrypting function to it and the Super Key.

[0026] While preferably the random non-repeating string which forms each sub-key is generated as described above, the method will also work if the non-repeating string of each sub-key is simply generated by a random number generator to form each sub-key, as long as the overall resultant length of the Super key is sufficiently large so that the resultant Super Key is at least double the size of the data to be encrypted.

[0027] The present invention is described above as a computer-implemented method. It may also be embodied as a computer hardware apparatus, computer software code or a combination of same. The invention may also be embodied as a computer-readable storage medium embodying code for implementing the invention. Such storage medium may be magnetic or optical, hard or floppy disk, CD-ROM, firmware or other storage media. The invention may also be embodied on a computer readable modulated carrier signal.

[0028] As will be apparent to those skilled in the art in the light of the foregoing disclosure, many alterations and modifications are possible in the practice of this invention without departing from the spirit or scope thereof. Accordingly, the scope of the invention is to be construed in accordance with the substance defined by the following claims.

What is claimed is:

1. A method of generating an encryption key having length×bytes, the method comprising the steps of:

i) selecting a number n of sub-keys each having a unique non-repeating length m bytes;

ii) generating n random numbers, one for each sub-key, each having length m bytes;

iii) generating a n+1st random number R;

iv) for each byte whose position in said nth random number is calculated as Modm(R) applying a function to all n bytes to generate a value;

v) concatenating said value to the end of said encryption key; and

vi) repeating step iv) and v) until said key is x bytes in length.

2. The method of claim 1 wherein said selected length m of each said sub-key is a prime number.

3. The method of claim 1 wherein said selected length m of each said sub-key is a prime number greater than 10.

4. The method of claim 1 wherein said function applied to said n bytes of said sub-keys is the exclusive-or function.

5. The method of claim 1 wherein each of said n random numbers is generated by:

i) generating a first random number which is not a perfect square;

ii) calculating the square root of said first random number;

iii) generating a second random number;

iv) commencing with a digit whose position in said first random number is calculated based on said second

random number, taking finite strings of digits sequentially and converting each said finite string into a hexadecimal byte;

vi) concatenating each hexadecimal byte sequentially to said random number until the selected length m of said random number has been reached.

6. The method of claim 5 wherein said finite strings of digits are at least 4 digits long.

7. The method of claim 6 wherein said finite string is converted into a hexadecimal byte by applying a mod function.

8. The method of claim 7 wherein said finite string is converted into a hexadecimal byte by applying a mod **256** function.

9. A computer program product for generating an encryption key having length×bytes, said computer program product comprising a computer usable medium having computer readable program code means embodied in said medium for:

i) selecting a number n of sub-keys each having a unique non-repeating length m bytes;

ii) generating n random numbers, one for each sub-key, each having length m bytes;

iii) generating a n+1st random number R;

iv) for each byte whose position in said nth random number is calculated as Modm(R) applying a function to all n bytes to generate a value;

v) concatenating said value to the end of said encryption key; and

vi) repeating step iv) and v) until said key is x bytes in length.

10. The computer program product of claim 9 wherein said selected length m of each said sub-key is a prime number.

11. The computer program product of claim 9 wherein said selected length m of each said sub-key is a prime number greater than 10.

12. The computer program product of claim 9 wherein said function applied to said n bytes of said sub-keys is the exclusive-or function.

13. The computer program product of claim 9 wherein each of said n random numbers is generated by:

i) generating a first random number which is not a perfect square;

ii) calculating the square root of said first random number;

iii) generating a second random number;

iv) commencing with a digit whose position in said first random number is calculated based on said second random number, taking finite strings of digits sequentially and converting each said finite string into a hexadecimal byte;

vi) concatenating each hexadecimal byte sequentially to said random number until the selected length m of said random number has been reached.

14. The computer program product of claim 13 wherein said finite strings of digits are at least 4 digits long.

15. The computer program product of claim 14 wherein said finite string is converted into a hexadecimal byte by applying a mod function.

16. The computer program product of claim 15 wherein said finite string is converted into a hexadecimal byte by applying a mod **256** function.

* * * * *