(12) UK Patent Application (19) GB (11) 2 415 798 (13) A

(43) Date of A Publication 04.01.2006

(21) Application No: **0414475.4**

(22) Date of Filing: **29.06.2004**

(71) Applicant(s):
Farhad Dalvi
69 Preston Road, WEMBLEY, Middlesex,
HA9 8JZ, United Kingdom

(72) Inventor(s):
Farhad Dalvi

(74) Agent and/or Address for Service:
Farhad Dalvi
69 Preston Road, WEMBLEY, Middlesex,
HA9 8JZ, United Kingdom

(51) INT CL:
G06F 12/14 (2006.01) **G06F 1/00** (2006.01)

(52) UK CL (Edition X):
G4A AAP A23A
G5X X30

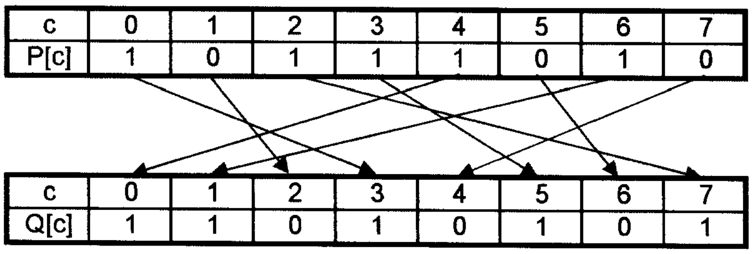
(56) Documents Cited:
GB 2194419 A
JP 610264936 A
US 4959863 A
US 20040015707 A1
US 20020027988 A1

(58) Field of Search:
UK CL (Edition W) **G4A, H4P**
INT CL^{7} **G06F, H04L**
Other: **Online: WPI, EPODOC, JAPIO**

(54) Abstract Title: **A non-deterministic secret key cipher using bit permutations**

(57) A non-deterministic, symmetric cryptosystem operating on a block of n binary digits where n is a positive integer greater than 2. Each bit is placed in an integer array P[c] (Figure 2) of n elements in length, i.e. 0≤c<n. Using the integer array K[c] (Figure 3), also n elements in length but where the constituent integers are randomly sequenced, unique and vary between 0≤K[c]<n, the integer array P[c] is re-arranged by substituting its index-position c, with a corresponding element of K[c] to produce the cipher-text integer array Q[c], Figure 7. The cipher-text array Q[c] is decrypted using the integer-array L[c], Figure 9, derived by re-sorting the constituent integers of K[c] in numerical order and using the resulting sequence of index-positions to populate L[c]. The cipher-text array is re-arranged by substituting its index-position with the corresponding element of L[c] to re-produce the plain-text array P[c]. For example, in figure 3 K[c] has integers 4, 6, 1, 0, 7, 3, 5, 2 resulting in the fifth bit of P[c=4] being placed in the first bit position of Q[c=o], the seventh bit of P[c=6] being placed in the second bit position of Q[c=1], and so on.

Figure 7



GB 2 415 798 A

Figure 1

c	0	1	2	3	4	---	n - 3	n - 2	n - 1
P[c]	1	0	0	1	1	---	1	0	1

Figure 2

c	0	1	2	3	4	5	6	7
P[c]	1	0	1	1	1	0	1	0

Figure 3

c	0	1	2	3	4	5	6	7
K[c]	4	6	1	0	7	3	5	2

Figure 4

c	0	1	2	3	4	5	6	7
Q[c]								

Figure 5

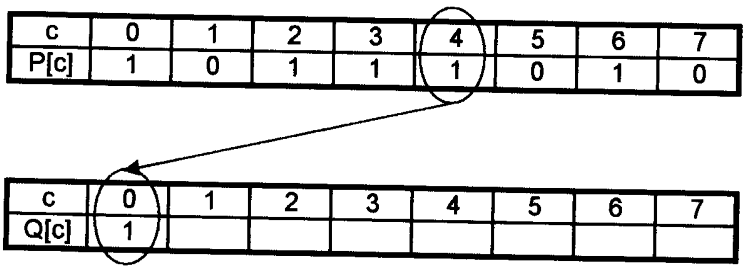


Figure 6

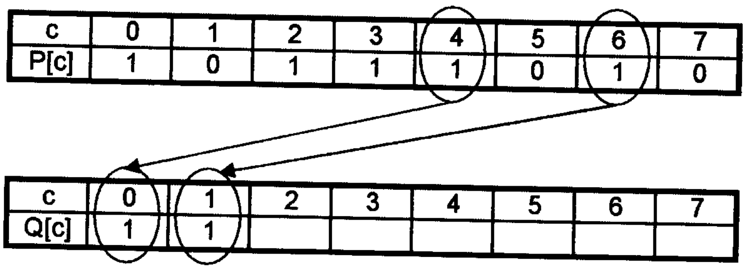


Figure 7

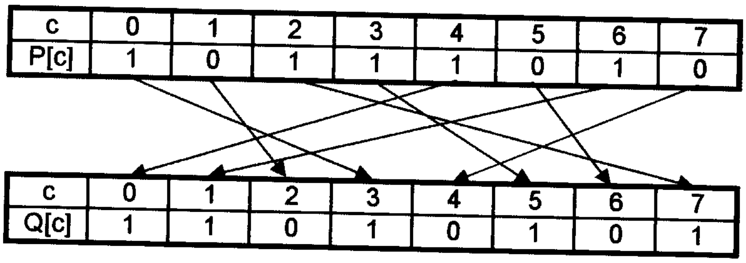


Figure 8

K[c]	4	6	1	0	7	3	5	2
c	0	1	2	3	4	5	6	7

Figure 9

c	0	1	2	3	4	5	6	7
L[c]	3	2	7	5	0	6	1	4

Figure 10

c	0	1	2	3	4	5	6	7
P[c]								

Figure 11

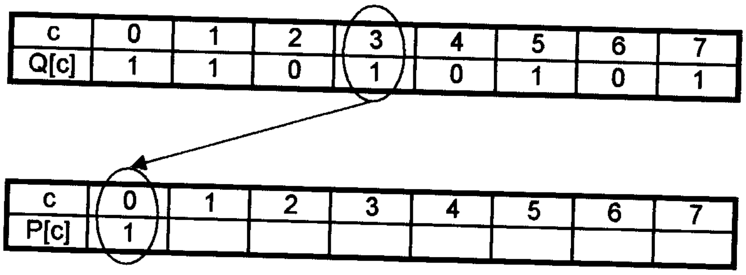
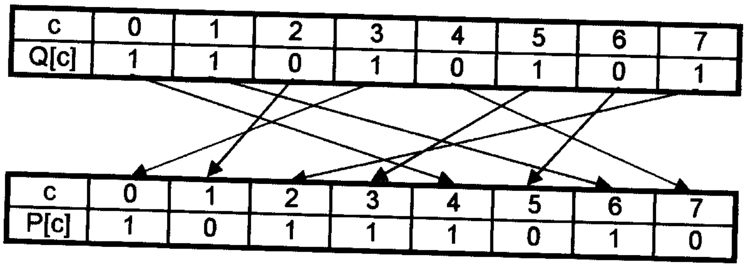


Figure 12



2415798

A NON-DETERMINISTIC SECRET KEY CIPHER

The invention described is a non-deterministic, symmetric cryptographic algorithm (cryptosystem) that operates on a block of n binary digits, where n is a positive integer greater than 2.

In essence, the cryptosystem re-arranges the position of every bit in the n bit plain-text block to another unique position within the block, to produce a corresponding n bit cipher-text block. The decryption process reverses the re-arrangement to produce the original n bit plain-text block. The re-arrangement pattern is dependent on a key, a random sequence of n unique integers where each integer varies between 0 and n-1. The number of possible keys (key-space) and hence, the number of possible ways the n bit plain-text block can be re-arranged is: n!-1.

One of the primary factors that determine the security of a cryptosystem is the length of the key used to encrypt a message. Larger the key, greater the number of possible keys (i.e. key-space). The easiest method of compromising a cryptosystem is to try every possible key to decrypt a given block of cipher-text. This form of attack is known as Brute-Force. A Brute-Force attack is harder on a cryptosystem that employs a large key-space as it takes more time to churn through every key. Modern-day algorithms employ a 128-bit integer key, yielding 2^{128} possible keys. A Brute-Force attack on such an algorithm would require a maximum of 2^{128} (approximately 3.40e+38) attempts to decrypt the given cipher-text block. These algorithms can be made more secure by increasing the key-space. However, for algorithms that employ a deterministic exponential formula, the computation time to process the plain-text would grow exponentially with key-space.

In its preferred configuration, the cryptosystem described here has a key-space of 4096!-1, approximately 3.64e+13019. As 3.64e+13019 is much larger than 3.40e+38, a Brute-Force attack on this cryptosystem would require proportionately greater time. Also since this cryptosystem does not involve any mathematical computation, increasing the key-space results only in a linear increase in computation time to process the plain-text.

An object of this invention is to provide an electronic mechanism to shred sensitive electronic-data stored on a personal-computer and un-shred the data using a private key. Accordingly, this invention provides the following algorithm.

The algorithm will now be described with reference to the accompanying figures in which:

FIGURE 1:	is an array P[c], a function that groups a set of n integers in a particular sequence.
FIGURE 2:	is the plain-text array P[c], representing the first 8 bits of the plain-text
FIGURE 3:	is the key array K[c] used to encrypt P[c]
FIGURE 4:	is the empty cipher-text array Q[c]
FIGURE 5:	shows the first element of the cipher-text array Q[0] being populated

FIGURE 6. shows the second element of the cipher-text array Q[1] being populated
FIGURE 7: shows the completely populated cipher-text array Q[c]
FIGURE 8. shows the flipped key-array K[c]
FIGURE 9: shows the complementary key array L[c] used to decrypt Q[c]
FIGURE 10: is the empty plain-text array P[c]
FIGURE 11. shows the first element of plain-text array P[0] being populated
FIGURE 12: shows the completely populated decrypted plain-text array P[c]

The algorithm employs integer-arrays, a function that stores a set of integers in a particular sequence. Every integer in the sequence is stored in separate elements. Hence, to store a sequence of n integers, an integer-array of n elements is required. Figure 1 shows an example of an integer-array P[c], n elements in length and where each element contains a constituent-integer. Each element and hence, its corresponding constituent-integer is given an index-position, c, a unique integer starting from 0 and increasing by 1 for the next n-1 elements. Hence, $0 \leq c < n$. Each constituent-integer can be accessed individually for computation by referencing its unique index-position, c. For example, the forth integer in the array P[c] is accessed by referring to its index-position, P[3] = 1.

To simplify the description of this cryptosystem, a plain-text block on n = 8 bits will be employed. However, the preferred configuration of this algorithm would be to use a plain-text block of n = 4096 bits.

The encryption algorithm begins by taking the first block of n=8 bits from the plain-text and places them in sequence into an array P[c], the plain-text array. This is shown in Figure 2.

Figure 3 shows the key-array K[c], n=8 elements in length and where each constituent-integer is unique, randomly arranged and varies between 0 and n-1, inclusive. Hence, $0 \leq c < n=8$ and $0 \leq K[c] < n=8$. There are 8!-1 = 40319 possible keys.

Figure 4 shows the cipher-text array Q[c], also n=8 elements in length but initially empty. The first element of the cipher-text array, Q[0] is determined by substituting the index-position of the plain-text array P[c] with the first element of the key-array, K[0].

Q[0] = P[K[0]] = P[4]

Hence the first element of the cipher-text array, Q[0] is equal to the forth element of P[c], i e. Q[0] = P[4]. The constituent-integer corresponding to the forth element of P[c] is now placed in the first element of Q[c], as shown in Figure 5.

The second element of the cipher-text array, Q[1] is similarly determined by substituting the index-position of P[c] with the second element of the key array, K[1], as shown in Figure 6:

Q[1] = P[K[1]] = P[6]

The constituent-integer corresponding to the sixth element of P[c] is now placed in the second element of Q[c].

The remaining elements of Q[c] are also determined by substituting the index-position of P[c] with the corresponding element of the key array K[c]:

Q[2] = P[K[2]] = P[1]
Q[3] = P[K[3]] = P[0]
Q[4] = P[K[4]] = P[7]
Q[5] = P[K[5]] = P[3]
Q[6] = P[K[6]] = P[5]
Q[7] = P[K[7]] = P[2]

Figure 7 shows the complete cipher-text array Q[c]. The encryption algorithm now processes the next n=8 bits until the entire plain-text has been processed.

The encryption algorithm can be summarized using the iterative formula:

```
for ( c=0, c<n, c=c+1 ) {
     Q[c] = P[K[c]]
}
```

For a block of n=8 bits:

```
for ( c=0, c<8, c=c+1 ) {
     Q[c] = P[K[c]]
}
```

To prepare for the decryption process, the recipient of the cipher-text has to reverse-map the key array K[c] to produce the complement key array L[c]. The first step is to flip K[c] such that the constituent-integers of K[c] become index-positions and the index-positions become the constituent-integers, as shown in Figure 8. The complementary key array L[c] is now deduced by sorting this array in the order of the new index-positions, as shown in Figure 9.

The decryption algorithm now begins by taking the first block of n=8bits from the received cipher-text Q[c], as deduced in Figure 7. Using the complementary key array L[c] shown in Figure 9 the plain-text array P[c] can be deduced using the same algorithm.

Starting with an empty integer array P[c] of n=8 elements in length, as shown in Figure 10, the first element, P[0] is determined by substituting the index-position of Q[c] with the first element of the complementary key array, L[0]:

P[0] = Q[L[0]] = Q[3]

Hence the first element of the decrypted plain-text array, P[0] is equal to the third element of the cipher-text array, Q[3], as shown is Figure 11. The constituent-integer corresponding to Q[3] is now placed in the first element of P[c], P[0].

The remaining 7 elements of the plain-text array P[c] are similarly determined by substituting the index-positions of the cipher-text array Q[c] with the corresponding elements of the complementary key array L[c]:

P[1] = Q[L[1]] = Q[2]
P[2] = Q[L[2]] = Q[7]
P[3] = Q[L[3]] = Q[5]
P[4] = Q[L[4]] = Q[0]
P[5] = Q[L[5]] = Q[6]
P[6] = Q[L[6]] = Q[1]
P[7] = Q[L[7]] = Q[4]

Figure 12 shows the complete plain-text array P[c] as deduced from the calculations above. This is identical to the plain-text array prior to encryption, as shown in Figure 2.

The decryption algorithm now processes the next n=8 bits until the entire cipher-text has been processed.

The decryption algorithm can be summarized using the iterative formula:

```
for ( c=0, c<n, c=c+1 ) {
     P[c] = Q[L[c]]
}
```

For a block of n=8 bits:

```
for ( c=0, c<8, c=c+1 ) {
     P[c] = Q[L[c]]
}
```

As previously mentioned, the preferred configuration of this algorithm would be to use a plaint-text block of n = 4096 bits. In this instance, the first 4096 bits of the plain-text are placed in sequence in an integer-array P[c], n = 4096 elements in length. That is, each bit is placed in a separate element and hence, $0 \leq c < n = 4096$. The corresponding cipher-text array Q[c], also n = 4096 elements in length is populated using a key-array K[c].

In this instance, the key-array K[c] is n = 4096 elements in length and the constituent-integers are unique, randomly sequenced and vary between 0 and 4095, inclusive. That is, $0 \leq K[c] < 4096$ and $0 \leq c < 4096$.

To decrypt the cipher-text array Q[c], the key-array K[c] is re-mapped as described above to produce the complementary key-array L[c], where $0 \leq L[c] < 4096$ and $0 \leq c < 4096$.

CLAIMS

1. An algorithm that encrypts a block of n binary digits by placing each bit in an integer array P[c], n elements in length where n is a positive integer greater than or equal to 2, and 0 ≤c<n, and re-arranges the position of each element of P[c] by taking each element in turn from P[0] to P[n-1], inclusive, and substituting c, the index-position of P[c] with the corresponding element of another integer array K[c] also n elements in length, in which each element contains an unique integer between 0 and n-1, inclusive, in a random sequence, to produce another integer array of n elements in length Q[c] = P[K[c]], where 0 ≤c<n.

2. A corresponding algorithm to Claim 1 that decrypts a block of n binary digits by placing each bit in an integer array Q[c], n elements in length where n is a positive integer greater than or equal to 2, and 0 ≤c<n, and re-arranges the position of each element of Q[c] by taking each element in turn from Q[0] to Q[n-1], inclusive, and substituting c, the index-position of Q[c] with the corresponding element of another integer array L[c] also n elements in length, in which each element contains an unique integer between 0 and n-1, inclusive, in a random sequence, to produce another integer array of n elements in length P[c] = Q[L[c]], where 0≤c<n.

3. An integer array K[c] as claimed in Claim 1 where the array is n elements in length, that is 0 ≤c<n, and the constituent integers are randomly sequenced, unique and vary between 0 and n-1, inclusive, that is 0≤K[c]<n.

4. An integer array L[c] as claimed in Claim 2 where the array is n elements in length, that is 0 ≤c<n, and the constituent integers are randomly sequenced, unique and vary between 0 and n-1, inclusive, that is 0≤L[c]<n, is deduced by re-sorting the integer array K[c] such that the constituent integers are in numerical order and employing the resulting re-sequence of index-positions to populate the array L[c].

The Patent Office

INVESTOR IN PEOPLE

Application No:	GB0414475.4	**Examiner:**	Matthew Nelson
Claims searched:	1-4	**Date of search:**	17 November 2004

Patents Act 1977: Search Report under Section 17

Documents considered to be relevant:

Category	Relevant to claims	Identity of document and passage or figure of particular relevance
X	1-4	GB 2194419 A (BBC) See definition of DES permutation algorithm and table.
X	1-4	US 2004/0015707 A1 (LEE) See figures 1-4.
X	1-4	US 2002/0027988 A1 (CALLUM) See figure 4, paragraph [0024] and table B.
X	1-4	US 4959863 A (AZUMA et al) See figures 1, 16, 18 and 19.
X	1-4	JP 61264936 A (FUJITSU) See WPI Abstract Accession No. 1987-004298 [01] and figure 1.

Categories:

X	Document indicating lack of novelty or inventive step	A	Document indicating technological background and/or state of the art
Y	Document indicating lack of inventive step if combined with one or more other documents of same category.	P	Document published on or after the declared priority date but before the filing date of this invention.
&	Member of the same patent family	E	Patent document published on or after, but with priority date earlier than, the filing date of this application.

Field of Search:

Search of GB, EP, WO & US patent documents classified in the following areas of the UKC[W] :

G4A; H4P

Worldwide search of patent documents classified in the following areas of the IPC[07]

G06F; H04L

The following online and other databases have been used in the preparation of this search report

Online: WPI, EPODOC, JAPIO