Bijective Function with Domain in N and Image in the Set of Permutations: An Application to Cryptography

Víctor M. Silva-García[†], Cornelio Yáñez-Márquez[††], and Juan L. Díaz de León-Santiago[††],

[†]CIDETEC, National Polytechnics Institute, México
[††]CIC, National Polytechnics Institute, México

Summary
In this work an algorithm is constructed that counts n! permutations in n-1 steps. Actually, the algorithm defines a bijective function from the natural numbers to the set of permutations. In addition, for any permutation π_L defined over the positions of a string of length L, where L is a multiple of 3, this permutation may be constructed by means of 3 permutations over strings of length ⅔L. This allows for the definition of an iterative cryptosystem over blocks of 96 characters, with numbers in the range of 10^{90} instead of 10^{154}, approximately. It is also shown that the set of keys grows factorially, so that the number of elements of the set reaches 2^{500} when working with strings of 96 characters. Finally, by means of an example the iterative cryptosystem using the DES boxes and strings of 96 characters is illustrated.

Key words:
JV Theorem, Factorial Theorem, Factorial Cryptosystem, Permutations.

1. Introduction

It is well known that many iterative systems like DES, triple-DES, SPN and AES employ basically three types of operations, that is, permutations, substitutions and the Boolean exclusive-or function (XOR) [4, 5]. Permutations are tabulated and considered fixed. Up to this time the possibility of representing a permutation by means of a nonnegative number has not been explored. Naturally, an algorithm must be constructed that relates a permutation to a natural number. Such an algorithm defines then a bijective function [14]. This function allows the permutation to be considered as a key since the permutation is then a variable. Then, in principle, the key may be represented by one or several nonnegative integers. Using this idea, iterative cryptosystems may be constructed that are of high computational complexity, but fast and moderately complex in their implementation [13].

In this work a cryptosystem is proposed, having an execution time of the same order of magnitude than triple-DES [5], but at a complexity level of 2^{500}, which is vastly superior to the AES[5]. Additionally it possesses the whitening property like the most recent cryptosystems [5]. This property avoids both linear and differential attacks [1-2].

2. Preliminaries

Before the JV and factorial theorems are proofed, it is necessary to present 2 examples in order to illustrate the proofs given below.

First example:

Suppose strings of 8 characters are used. A permutation of these characters consists in changing their positions in the string, that is, positions 0, 1, 2, 3, 4, 5, 6, and 7, to a new particular array; for instance: 5, 7, 6, 4, 2, 0, 1, and 3. Now assume a nonnegative integer n is given such that $0 \leq n \leq 8! - 1$; say n = 17777. This number may be expressed as follows:

$$17777 = 3(7!) + 3(6!) + 4(5!) + 0(4!) + 2(3!) + 2(2!) + 1(1!) \quad \textbf{2.1}$$

In fact, any integer n in the interval $0 \leq n \leq 8! - 1$ may be written uniquely, by using the algorithm of Euclid, as long as 7!, …, 1! remain fixed. Note that we use as the arithmetic base the numbers 7!, 6!, 5!, 4!, 3!, 2! and 1!. Denote the factors of 7!, 6!, 5!, 4!, 3!, 2! y 1! by C_0, C_1, C_2, C_3, C_4, C_5, C_6, respectively. Then, for this example the factors are:

$C_0 = 3$, $C_1 = 3$, $C_2 = 4$, $C_3 = 0$, $C_4 = 2$, $C_5 = 2$ and $C_6 = 1$.

As may be seen the values C_i are the coefficients of the divisions by 7!, …, 1!. Furthermore, by the algorithm of Euclid the factors must satisfy that $C_0 < 8$, $C_1 < 7$,…, $C_6 < 2$ [14]. By virtue of above the following algorithm may be constructed:

Step 0. Define an array in increasing order as follows:

X[0] = 0, X[1] = 1, X[2] = 2, X[3] = 3, X[4] = 4, X[5] = 5, X[6] = 6 and X[7] = 7.

Step 1. Take the value of $X[C_0 = 3] = 3$ and eliminate it from the array defined in step 0. The array is then reordered without including the value of $X[C_0]$. The result is:

Manuscript received April 5, 2007
Manuscript revised April 25, 2007

X[0] = 0, X[1] = 1, X[2] = 2, X[3] = 4, X[4] = 5, X[5] = 6 and X[6] = 7.

Step 2. Take the value of $X[C_1 = 3] = 4$ and eliminate it from the array defines in step 1. The array is the reordered without including the value of $[C_1]$. The result is:

X[0] = 0, X[1] = 1, X[2] = 2, X[3] = 5, X[4] = 6 y X[5] = 7

Step 3. As in step 2 take $X[C_2 = 4] = 6$ and eliminate it from the array defined in step 2. The new order is:

X[0] = 0, X[1] = 1, X[2] = 2, X[3] = 5 y X[4] = 7.

Step 4. Continue in the same way with $X[C_3 = 0] = 0$ and the resulting order is:

X[0] = 1, X[1] = 2, X[2] = 5 y X[3] = 7.

Step 5. In this step $X[C_4 = 2] = 5$ is eliminated and:

X[0] = 1, X[1] = 2, X[2] = 7.

Step 6. Following the same procedure $X[C_5 = 2] = 7$ is eliminated and X[0] = 1, X[1] = 2.

Step 7. Finally one eliminates $X[C_6 = 1] = 2$ and X[0] = 1.

If the (eliminated) values of $X[C_0]$, $X[C_1]$, $X[C_2]$, $X[C_3]$, $X[C_4]$, $X[C_5]$, $X[C_6]$ and the final value X[0] are written in order the result is 3, 4, 6, 0, 5, 7, 2, and 1. It is not difficult to see that the resulting array is a permutation of the numbers 0, 1, 2, 3, 4, 5, 6, and 7. In fact, it is the permutation 17777. It is also important to note that the number of steps required to assign a number to a permutation is 7.

Second example:

Now suppose that one is working with strings of 12 characters. A particular permutation of the positions of a string of that length could be:

5 4 11 0 8 6 3 2 1 7 10 9 **2.2**

Here we ask the following question: is there a way to apply permutations to strings of lesser length than 12 such that it is possible to obtain the permutation given by the expression **2.2**?

Fortunately the answer is yes. We illustrate the procedure graphically as shown in figure 1.

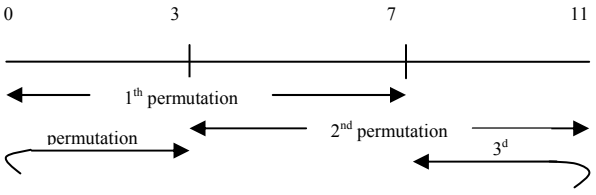

Fig. 1 Application of 3 permutations of length 8.

Now: may any permutation of 12 characters be obtained by applying 3 permutations according to the illustration? The answer is yes and it will be proofed below. In fact, the proof will be given for strings of length L, where L is a multiple of 3. The intention of this example is to describe the proof strategy.

We start with an ordered array, that is, 0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, and 11. Divide this set of numbers into 2, namely: **A** = {0, 1, 2, 3, 4, 5, 6, 7} and **B** = {8, 9, 10, 11}.

Furthermore, divide the given permutation, here 5 4 11 0 8 6 3 2 1 7 10 9, in 3 blocks as shown in figure 2.

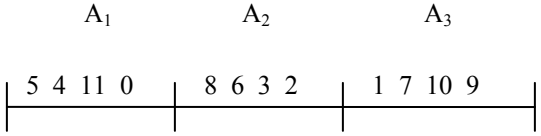

Fig. 2 Division of the permutation in 3 blocks.

The first permutation assigns the positions of the set **A** to the blocks A_1 and A_2, leaving out those positions that belong to the set **B**. This is shown below:

5 4 - 0 - 6 3 2 8 9 10 11

The missing values from set A are 1 and 7; lets place them at random in the holes. For example, first the 7 and than the 1. Note that this opens the possibility that there exist more than three permutations by which the given permutation may be constructed. The result of applying the first permutation is:

5 4 7 0 1 6 3 2 8 9 10 11

It follows that the first permutation is $\pi_1(y)$ = 5 4 7 0 1 6 3 2, with $0 \leq y \leq 7$.

The second permutation is applied to the characters 4 to 11. However, in order to carry this out it is necessary to define a displacement function as follows: $g_1(y) = 4-y$ with $4 \leq y \leq 11$. This is shown in the table below:

Table 1. The displacement function $g_1(y)$.

			$g_1(y)$=	0	1	2	3	4	5	6	7
5	4	7	0	1	6	3	2	8	9	10	11

According to the former ideas the permutation $\pi_2(g_1(y))$ is constructed as follows:

1. Positions that are in place are not modified.
2. Assign the positions from blocks A_2 and A_3 that are elements from set B, as is the case with 8, 9 and 10. Also, assign the positions from $\pi_1(y)$ with $4 \leq y \leq 7$ that should be in A_3, as is the case for 1. The positions of the form $\pi_1(y)$ with $0 \leq y \leq 3$ that should be in the block A_3 are substituted by the remaining positions. Here, the position $\pi_1(2) = 7$ must be in

block A_3 and is exchanged for the remaining position, which is:

$\pi_2(g_1(11) = 7) = 5$.

At this point the positions of block A_2 are in their place:

5 4 7 0 8 6 3 2 1 11 10 9

It follows that the second permutation is:

$\pi_2(g_1(y)) = 4\ 1\ 2\ 3\ 0\ 7\ 6\ 5$

In order to apply the third permutation, we define the displacement function $g_2(y)$:

$$g_2(y)=\begin{cases} y\text{-}8 & \text{if } 8 \le y \le 11 \\ y\text{+}4 & \text{if } 0 \le y \le 3 \end{cases}$$

This is shown in figure 3.

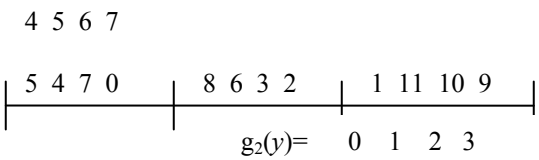

Fig. 3 The displacement function $g_2(y)$

The permutation $\pi_3(g_2(y))$ proceeds then according to the following steps:

1. Positions that are in their place are not modified.
2. Assign the positions that are members of the set B in block A1. This locates the position 11 in the position 6. Also, relocate positions of the form $\pi 1(y)$ with $0 \le y \le 3$ which should be in A3. This locates position 7 in position 1. It follows that the third permutation is $\pi 3(g2(y)) = 0\ 6\ 2\ 3\ 4\ 5\ 1\ 7$. The final result is:

5 4 11 0 8 6 3 2 1 7 1 0 9

Some comments are in order. The development of this kind of procedure allows us to work with numbers in the order of 10^{90} instead of 10^{150}, approximately, while using strings of 96 characters in length. In general, it may be said that this type of procedure reduces significantly the amount of computation. On the other hand, using this procedure but with 4 permutations instead of 3 in order to reduce even more the range of numbers, then some permutations of the string of 12 characters would not be included.

Example: given the permutation 3 4 5 1 0 11 10 6 2 7 8 9 it is not possible to construct this permutation using 4 permutations of strings of 6 characters since the number 11 may not be placed in the position 5 (recall that the positions are counted starting at 0).

3. Development

Define the set $\mathbf{N}_m$ as follows: $\mathbf{N}_m = \{n \epsilon \mathbf{N} \mid 0 \le n < m!\}$ with m a positive integer. For any $n \in \mathbf{N}_m$ the following iterative procedure will be applied:

Step 0.

$n = C_0(m\text{-}1)! + r_1$ and by the algorithm of Euclides [14], $0 \le r_1 < (m\text{-}1)!$ **3.1**

Then, $n < m! \Rightarrow \dfrac{n}{(m-1)!} = C_0 + \dfrac{r_1}{(m-1)!} < m$

Hence, $0 \le C_0 < m$

Step 1.

$r_1 = C_1 (m\text{-}2)! + r_2$ and by the same argument used above we have:

$0 \le r_2 < (m\text{-}2)!$ **3.2**

From expression **3.1** results

$$r_1 < (m-1)! \Rightarrow \frac{r_1}{(m-2)!} = C_1 + \frac{r_2}{(m-2)!} < m-1$$

It follows that $0 \le C_1 < m - 1$

Step i.

$r_i = C_i [m\text{-}(i\text{+}1)]! + r_{i+1}$ with $0 \le r_{i+1} < [m\text{-}(i\text{+}1)]!$. In the same way as for expressions **3.1** and **3.2** in step $(i\text{-}1)$ r_i must satisfy $0 \le r_i < (m\text{-}i)!$. From this last expression it follows that $\dfrac{r_i}{[m-(i+1)!]} = C_i + \dfrac{r_{i+1}}{[m-(i+1)!]} < m - i$.

Hence the following holds: $0 \le C_i < (m\text{-}i)$.

Note that this shows that for any i with $0 \le i \le (m - 2)$:

$C_i < (m\text{-}i)$.

If one continues with this iterative process, at the end one obtains the following:

$r_{m-2} = C_{m-2} 1! + r_{m-1}$ with $r_{m-1} = 0$.

As a conclusion of this iterative process it can be stated that given $n \in \mathbf{N}_m$ and $(m\text{-}1)! \ldots 1!$; then the number n may be uniquely written as:

$n = C_0(m\text{-}1)! + C_1(m\text{-}2)! + C_2(m\text{-}3)! + \ldots + C_{m-2}\, 1!$ **3.3**

Also, the following holds:

$0 \le C_i < (m\text{-}i)$, with $0 \le i \le (m - 2)$ **3.4**

Now, once the values of $C_0, C_1, \ldots, C_{m-2}$ are known, the following algorithm may be constructed:

Step 0. An array in increasing order is defined as follows:

X[0] = 0, X[1] = 1, X[2] =2,... X[m-1] = m-1.

Step 1. By expression **3.4** we have $C_0 < m$; hence X[C_0] is an element of the array constructed in step 0. This element X[C_0] is eliminated from the array of step 0 and a new array is constructed starting from X[0] up to X[m-2].

Step 2. Again, according to expression **3.4** we have $C_1 < m$-1; hence X[C_1] is an element of the array obtained in step 1. In the same way as in the previous step X[C_1] is eliminated from the array of step 1 and a new array is obtained starting with X[0] up to X[m-3].

Step m-1. By continuing in the same fashion one obtains in the end the following:

X[C_{m-2}] and X[0].

Finally, the string of eliminated numbers X[C_0], X[C_1], …, X[C_{m-2}] and X[0] is a permutation of the string 0, 1, 2, …, m-1. Hence, it is possible to say that to any $n \epsilon \mathbf{N}_m$ a permutation may be associated. At this point the following question arises: given two different numbers from the set $\mathbf{N}_m$, do they generate two different permutations? This question is answered by the JV theorem, as stated below.

JV theorem. Given the sets $\mathbf{N}_m$ and $\mathbf{\Pi}_m$ = {all the permutations of the array 0,1,…, m-1}. Then, the algorithm described above defines a bijective function π_m such that $\pi_m : \mathbf{N}_m \rightarrow \mathbf{\Pi}_m$.

The proof is by contradiction. Suppose $n_1 \neq n_2$ with $n_1, n_2 \epsilon \mathbf{N}_m \Rightarrow \pi_m(n_1) = \pi_m(n_2)$.

From expression **3.3** we know that n_1, n_2 may be written as:

$n_1 = C_{0,1}(m\text{-}1)! + C_{1,1}(m\text{-}2)! + C_{2,1}(m\text{-}3)! + \ldots + C_{m\text{-}2,1}$ 1! and

$n_2 = C_{0,2}(m\text{-}1)! + C_{1,2}(m\text{-}2)! + C_{2,2}(m\text{-}3)! + \ldots + C_{m\text{-}2,2}$ 1!

Now, if $\pi_m(n_1) = \pi_m(n_2)$ it follows that: $C_{0,1} = C_{0,2}$, $C_{1,1} = C_{1,2}$, …, $C_{m\text{-}2,1} = C_{m\text{-}2,2}$. Hence,

$n_1 = n_2$, which is a contradiction of the initial assumption. Consequently, if $n_1 \neq n_2$ with $n_1, n_2 \epsilon \mathbf{N}_m \Rightarrow \pi_m(n_1) \neq \pi_m(n_2)$. This shows that the function π_m is one to one.

That the function π_m is bijective follows from the fact that the number of elements of the sets $\mathbf{N}_m$, $\mathbf{\Pi}_m$ are equal.

We now proof the factorial theorem.

Factorial theorem. Given a permutation π_L over the positions of a string of length L, with L a multiple of 3. Then, π_L may be constructed by means of 3 permutations of length ⅔L.

Let be the following permutation of the positions of a string of L elements:

$$\pi_L = \sigma(0) = j_0, \sigma(1) = j_1, \ldots \sigma(L\text{-}1) = j_{L-1} \quad \textbf{3.5}$$

Now, separate the set of positions in 2, namely:

$$\mathbf{A} = \{0,1,\ldots, ⅔L\text{-}1\} \text{ and } \mathbf{B} = \{⅔L, ⅔L+1, \ldots, L\text{-}1\} \quad \textbf{3.6}$$

Divide the permutation **3.5** into three as follows:

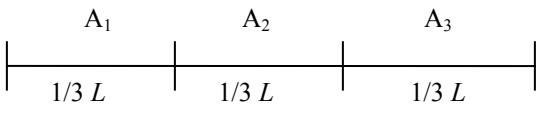

Fig. 4 Division of the string into three blocks.

The same strategy shown in figure 1 will be used. The first permutation $\pi_1(y)$ with $0 \leq y \leq ⅔L$-1 is constructed as follows:

1. Assign the positions that are elements of the set A to the blocks A1, A2.
2. The positions of set B, in case they exist and which should be in blocks A1, A2 are assigned at random by the remaining elements of A.

In order to apply the permutation π_2, we use the displacement function $g_1(y) = y - ⅓L$, for $⅓ L \leq y \leq L$-1.

The permutation $\pi_2(g_1(y))$ proceeds then as follows:

1. Positions that are in their place, in case they exist, are not modified.
2. Assign the positions, in case they exist, from blocks A2 and A3 that are elements of the set B. Assign the positions, in case they exist, of the form π1(y) for ⅓L ≤ y ≤ ⅔L-1 that should be in block A3. Positions, in case they exist, of the form π1(y) for 0 ≤ y ≤ ⅓L-1 that should be in the block A3, are exchanged with the remaining positions. At this point the positions of the block A2 are in their place.

In order to apply the permutation π_3 we use the displacement function:

$$g_2(y) = \begin{cases} y - \frac{2}{3}L \text{ if } \frac{2}{3}L \leq and \leq L\text{-}1 \\ y + \frac{1}{3}L \text{ if } 0 \leq and \leq \frac{1}{3}L\text{-}1 \end{cases}$$

The permutation $\pi_3(g_2(y))$ proceeds according to the following steps:

1. Positions that are in their place, in case they exist, are not modified.
2. Assign the positions, in case they exist, from block A1 that correspond to the set B as well as the positions of the form π1(y) for 0 ≤ y ≤ ⅓L-1 that should be in A3.

It follows that if the 3 permutations described above are applied the permutation **3.5** is constructed.

4. Proposal of a Cryptosystem

By using the JV and factorial theorems a cryptosystem may be proposed that has an execution time of the same order of magnitude than triple-DES but much more resistant to brute force attacks. The here proposed system is iterative in nature. In what follows, a high level description will be given:

1. Assume a string of 12 bytes of clear text is given, equivalent to a string of 96 bits. Chose 3 positive integers n_1, n_2 and n_3 such that $2 \leq n_i \leq 64!-1$ for i = 1, 2, 3.

2. According to the JV theorem, to the positive integers ni one may associate 3 permutations over strings of 64 positions in length. Then, it follows from the factorial theorem that it is possible to construct any permutation of the string of clear text of 96 bits, call this permutation π_{96}. We shall refer to the application of this permutation to the clear text as π_{96}(TC).

3. Since the string π_{96}(TC) is of 96 bits, it is possible to divide it into 2 substrings, one right substring and one left substring each of 48 bits in length. Call these substrings R_0 and L_0, respectively. Starting with these substrings, the following iterative procedure will be applied 8 times:

   $L_i = R_{i-1}$ and
   $R_i = L_{i-1} \oplus g(R_{i-1})$ for i = 1,2,..,8 where the symbol $\oplus$ denotes the Boolean exclusive-or function.
   The function g does the following:
   a. The right substring R_{i-1} of 48 bits is fed to the 8 boxes of the DES criptosystem [4].

   b. The result of the former step is a string of 32 bits, to which the DES expansion function E is applied [4]. The resulting string of 48 bits will be called $g(R_{i-1})$.

   This procedure is illustrated in the following figure:

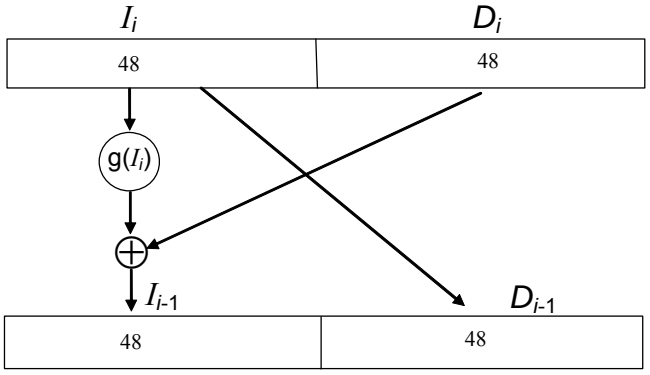

Fig. 5 Iteration *i* of the proposed algorithm.

4. After the 8th iteration $(\pi_{96})^{-1}(R_8L_8)$ is applied. Note that $(\pi_{96})^{-1}$ is the inverse permutation of π_{96} and that the substrings R_8, L_8 are inverted.

Some additional remarks:

1. As can be seen, the integers n_1, n_2, and n_3 act like keys, since the permutation π_{96} can be changed by altering one or some of the numbers n_1, n_2, and n_3.

2. Considering that each permutation is a key, the number of possible keys is approximately 10^{150}.

3. The proponed cryptosystem possesses the whitening property [5].

4. Decryption differs from the encryption process where the right substring is altered by the boxes, by altering the left substring. The figure shows the procedure:

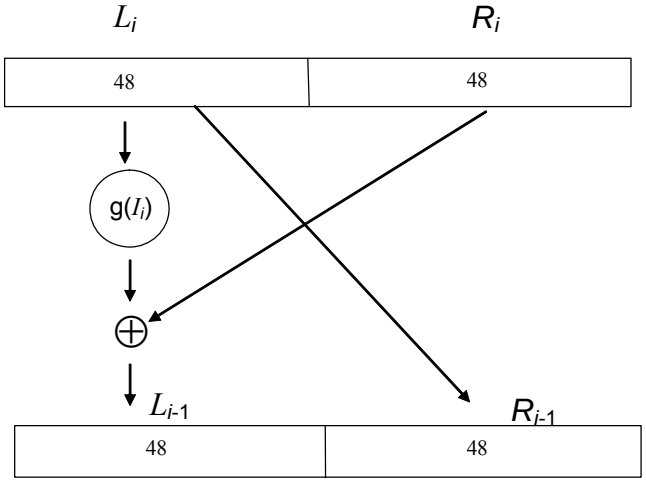

Fig. 6 Iteration *i* of the proposed algorithm during decryption.

To conclude this section, the authors propose the name "Factorial Cryptosystems" for all cryptosystems based on the JV and factorial theorems.

5. Results of the proposed algorithm.

In what follows the working of the algorithm will be explained by means of a specific example. Suppose the clear text **VíctorManuel** is to be encrypted and assume for n_1, n_2, and n_3 the following values:

n_1 = 12645790121264579012126457901212645790121264579012126457901212645790121264579012126457901212645790121264579012

n_2 = 12645791121264579012126457901212645790121264579012126457901212645790121264579012

n_3 = 126457901212645790121264579012126457901214579012126457901212645790121264579012126457901212645790121264579012

These numbers satisfy the condition $2 \leq n_i \leq 64!-1$. The permutations associated to these numbers follow:

π_1 = 63, 49, 22, 46, 56, 40, 35, 39, 23, 41, 38, 20, 55, 52, 31, 47, 34, 25, 54, 61, 17, 10, 53, 42, 44, 57, 14, 4, 19, 32, 28, 43, 16, 26, 45, 50, 2, 0, 51, 59, 12, 48, 18, 11, 5, 6, 7, 36, 37, 30, 62, 15, 24, 21, 13, 3, 33, 27, 60, 1, 29, 8, 9, 58.

π_2 = 0, 1, 2, 3, 4, 5, 37, 17, 41, 27, 13, 61, 9, 50, 32, 12, 24, 38, 54, 53, 40, 59, 58, 20, 35, 6, 39, 25, 51, 30, 62, 42, 44, 8, 43, 34, 52, 33, 23, 19, 56, 48, 22, 47, 14, 60, 21, 28, 55, 18, 10, 45, 29, 57, 16, 49, 36, 15, 63, 31, 46, 11, 7, 26.

π_3 = 0, 1, 2, 3, 4, 5, 9, 13, 56, 15, 34, 21, 52, 35, 48, 14, 26, 11, 43, 51, 44, 36, 22, 23, 29, 47, 58, 27, 8, 20, 30, 63, 37, 39, 50, 53, 54, 40, 33, 16, 59, 28, 6, 19, 10, 12, 60, 25, 18, 55, 24, 42, 41, 31, 45, 46, 49, 32, 62, 7, 57, 17, 38, 61.

The permutation π_{96} is obtained by applying a similar procedure to the one shown in figure 1, with the following result:

π_{96} = 40, 39, 54, 10, 53, 23, 49, 87, 4, 78, 3, 77,13, 7, 19, 36, 18, 42, 68, 38, 41, 60, 52, 31, 25, 63, 28, 15, 57, 62, 35, 32, 16, 26, 45, 50, 2, 0, 69, 30, 73, 1, 6, 93, 48, 82, 64, 5, 33, 70, 86, 85, 72, 91, 90, 24, 67, 51, 71, 27, 83, 9, 94, 74, 76, 12, 75, 66, 84, 65,80, 92, 44, 29, 22, 89, 17, 46, 34, 21, 95, 79, 20, 61, 55, 56, 37, 81, 11, 47, 14, 58, 88, 8, 59, 43.

The inverse permutation $(\pi_{96})^{-1}$ yields the following result:

$(\pi_{96})^{-1}$ = 37, 41, 36, 10, 8, 47, 42, 13, 93, 61, 3, 88, 65, 12, 90, 27, 32, 76, 16, 14, 82, 79, 74, 5, 55, 24, 33, 59, 26, 73, 39, 23, 31, 48, 78, 30, 15, 86, 19, 1, 0, 20, 17, 95, 72, 34, 77, 89, 44, 6, 35, 57, 22, 4, 2, 84, 85, 28, 91, 94, 21, 83, 29, 25, 46, 69, 67, 56, 18, 38, 49, 58, 52, 40, 63, 66, 64, 11, 9, 81, 70, 87, 45, 60, 68, 51, 50, 7, 92, 75, 54, 53, 71, 43, 62, 80.

The result of the encryption process in hexadecimal format is:

897A4FA1980E73CDF8BF937F.

6. CONCLUSIONS

As can be seen, the former procedure may be applied to many situations. As a matter of fact, it may be applied to all those cryptosystems that rely on permutations. On the other hand, the factorial function grows faster than the exponential function, which means that the number of available keys grows to extraordinary values, here to about 2^{500} (10^{150})[6].

Lastly, in contrast to the DES and triple-DES systems where the permutations are fixed, the here proponed cryptosystem is based on variable permutations which, when implemented in hardware, are costly both in the number of required gates and in execution time. In what follows it is assumed that the gates have a fanin of two and unlimited fanout. Also, the logical operation of negation is assumed to be incorporated into the input(s) and output of the gates, if required, and thus do not consume time nor does it require additional gates. Note that this analysis is intended only as a reference, since actual results vary with the fanin and fanout of the logic device actually used.

6.1. The permutation

Consider an implementation based on a crossbar switch as shown in the figure below. The input is applied to the columns and the output is obtained from the rows. Here, the permutation shown is the following:

0 →3, 1 →0, 2 →4, 3 →2, and 4 →1

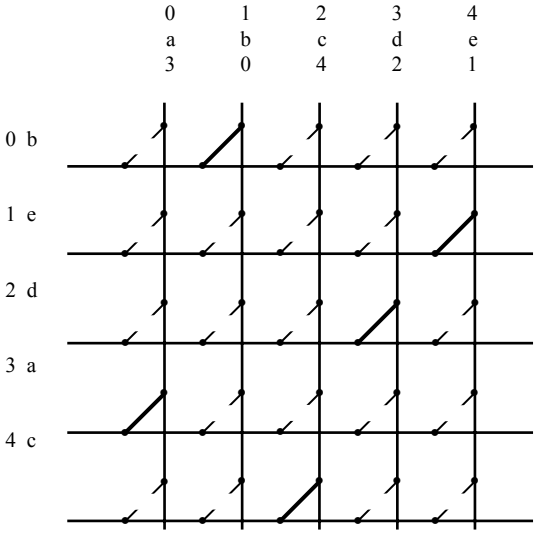

Fig. 7 A permutation executed by switch crossbar.

It is easy to see that N^2 switches are required. Associated to each switch is a decoder with lg(N) inputs, where lg(N) denotes the logarithm base 2 of N, that activates, if required, the switch. Finally, the output of the switches of a given row are combined by an N-input or gate to produce the output of the row. For $N = 2^n$, n a positive integer, and for the assumptions given above, the following values result:

1. The switches: N^2 gates, delay:1
2. Decoders: N^2D gates, where $\left(\frac{1}{2}\lg N\right)+1 \leq D \leq \left(\lg N\right)-1$, delay: $\lceil \lg(\lg N) \rceil$, where $\lceil x \rceil$ denotes the ceiling of x.

3. Or gate: N(N–1) gates, delay: (lgN)

Here, N = 64 and we obtain 4096 + 4096(5) + 64(63) = 28,608 gates and a global delay of 10 gate delays. Note that the number of gates may be reduced considerably by using other switching schemes (for instance, a multistage banyan network [15]) but at the expense of higher delays.

6.2. Estimation of the total execution time

Referring to figure 5, we obtain for the total execution time the following:

1. The delay of the initial permutation.
2. The delay of 8 iterations given by the delay of the boxes and the exclusive or function.
3. The delay of the final permutation.

The delays of the initial and final permutation are similar and are given by the delays of 3 permutations over 64 bits to which must be added the delay due to the replacement of bits in the blocks as described in the example 2. The latter is variable, but it is clear that each replacement cannot require more than 32 displacements and, furthermore, the maximum amount of replacements is 64. The permutation over 96 bits requires then in the order of 3(10) delays, plus the delays due to 64 replacements. Assuming a barrel shifter is used to generate the displacements, a delay in the order of 6 results for each displacement and the insertion. Hence, the total delay of the 96-bit permutation is equal to, or less, than 30 + 6(64) = 414.

The boxes are read-only memories of 2^6 words of 4 bits each. If implemented by gates, they generate a delay of $\lceil \lg(6) \rceil = 3$. It is then clear that the execution or the 8 iterations requires an order of magnitude less time than the permutations, and the total execution time may be estimated as 1000 or less gate delays.

As an example, suppose an FPGA device is used that specifies a delay of 0.2 ns per gate and associated wire delay. Then, the proposed algorithm executes in 200 ns for a 96 bit block, that is, with a speed of 480 Mb/s. As can be seen, the speed is limited mainly by the replacement of bits. Since these operations may be executed in parallel and depending of the resources of the device, if the number of barrel shifters is increased the execution time may be reduced significantly and speeds in excess of 1 Gb/s may be obtained.

Finally, the importance of the factorial theorem must not be underestimated. If the permutation were to be executed directly over 96 bits, the amount of gates required would increase from 28,608 to 73,632 and the algorithm would be difficult to implement with the resources of FPGA's available today [16].

Acknowledgments

The authors would like to thank the Instituto Politécnico Nacional (Secretaría Académica, COFAA, SIP, and CIC), the CONACyT, and SNI for their economical support to develop this work.

References

[1] Biham E. and Shamir A., 1993, "Differential cryptanalysis of the full 16-round DES", Lecturer Notes in computer Science.
[2] Matsui M, 1994, "Linear Cryptanalysis for DES cipher", Lecture Notes in Computer Science.
[3] Grabbe J. Orlin, 2003, "Data Encryption Standard: The DES algorithm illustrated", Laissez faire City time, vol. 2, no 28.
[4] Douglas R. Stinson, 1995, CRYPTOGRAPHY: Theory and practice, CRC Press, pp. 70-113.
[5] Douglas R. Stinson, 2002, CRYPTOGRAPHY: Theory and practice, CHAPMAN & HALL/ CRC Press, second edition, pp. 74-116.
[6] Rosen K., 2003, Discrete Mathematics and its Applications, Mc. Graw Hill, fifth edition.
[7] Koblitz M., 1987, A Course in Number Theory and Cryptography, Springer-Verlag, pp. 53-80, New York Inc.
[8] Sorking A., 1980, LUCIFER: A cryptographic algorithm, Cryptología 8, pp 22-35.
[9] Fúster Sabater A. et al, 2001, Técnicas Criptográficas de protección de datos, Alfaomega 2ª Edición, pp. 5-92.
[10] Ritter T, 2006, "Triple-DES is Proven to be Very Secure?", http://www.ciphersbyritter.com/NEWS5/PROVSEC.HTM.
[11] Stalling W, March 2006, "Encryption Options Beyond DES", www.commsdesign.com.
[12] Carlet C., 2005, "On highly nonlinear S-boxes and their inability to thwart DPA attacks", 6th International Conference on Cryptology of the Springer-Verlag, pp. 49-62
[13] Lindig Bos M., Silva García V.M., 2006, "Diseño de un dispositivo para encripción de datos en tiempo real", CIDETEC-ESIQIE-IPN., vol. 2.
[14] Herstein I.N., 1986, Álgebra Abstracta, Grupo Editorial Iberoamérica, pp. 22 y 11.
[15] T. Leighton, 1992, Introduction to Parallel Algorithms and Architectures: Arrays, Trees, Hypercubes, Morgan-Kaufmann Publishers, San Mateo, California, pp. 394.
[16] AX Detailed Specs_DS, 2005, Actel Corp.

Víctor M. Silva-García *Mexican Nationality. Bachelor Degree on Physics and Mathematics (1972) by the IPN ESFM. Degree of Ms. C. (1980) by the Chapingo Posgraduate College. He is currently a Ph. D student on Computer Sciences at the IPN CIC, and Director of the IPN Computer Innovation and Technological Research Center.*

Areas of Interest: Probability and Statistics, Cryptography, Hardware.

Cornelio Yáñez-Márquez *Mexican Nationality. Bachelor Degree on Physics and Mathematics (1989) by the IPN ESFM. Degrees of Ms. C. (1995) on Computer Engineering and Ph. D. (2002) on Computer Sciences, obtained both at IPN Computer Research Center. Currently he is a Titular C researcher at IPN Computer Research Center.. A member of the Researchers National System. Areas of Interest: Associative Memories, Neural Networks, Mathematical Morphology, Image Analysis.*
http://www.cornelio.org.mx

Juan L. Díaz de León-Santiago *Mexican Nationality. Degrees of M. Sc. (1993) on Automatic Control and Ph. D. (1996) on Mathematical Morphology, obtained both at IPN CINVESTAV, México. Currently he is a Titular C researcher at IPN Computer Research Center. A member of the Researchers National System. Areas of Interest: Mathematical Morphology, Image Analysis, Morpohological Neural Networks and Associative Memories, Control Theory, Mobile Robotics.*