Tatra Mt. Math. Publ. **29** (2004), 17–29

**TATRA MOUNTAINS**
*Mathematical Publications*

# STREAM CIPHER CRYPTANALYSIS BASED ON THE EDIT-DISTANCE: A HARDWARE APPROACH

Slobodan Bojanić — Gabriel Caffarena —
Slobodan Petrović — Octavio Nieto-Taladriz

ABSTRACT. Pseudorandom generator schemes containing irregularly clocked linear feedback shift-registers (LFSRs) are widely used because of long period and large linear complexity of their output sequences. A cryptanalytic attack on such schemes that utilizes the divide-and-conquer paradigm can be based on the solution of the edit-distance problem. In this work, the time and space complexities of both serial and parallel approach to the solution of the edit-distance problem are discussed and the design of the special purpose FPGA hardware is presented. The implemented systolic array hardware architecture exploits the inherent parallelism of a dynamic programming algorithm and the results indicate a significant gain in performance over the software implementation.

## 1. Introduction

The pseudo-random sequence generator, which contains a Linear Feedback Shift Register (LFSR) whose clocking sequence is controlled by a subgenerator of general type, is often used in practice because its output sequence possesses good cryptographic properties: long period, high linear complexity, good statistical properties, etc. The scheme of the analyzed pseudo-random sequence generator is presented in Fig. 1.

However, if a noisy prefix of the output sequence of this generator is known to the cryptanalyst, it is possible to reconstruct the initial state of the LFSR by means of a generalized correlation attack. By making use of a special statistical model, it is possible to determine a set of candidate initial states of the LFSR, which could generate the intercepted output sequence [1]. This model employs

2000 Mathematics Subject Classification: 94A60.
Keywords: stream cipher, cryptanalysis, edit distance, reconfigurable hardware.

Work partially supported by the Spanish Ministry of Science and Technology under grants TIC2000-1395-C02-01, TIC2000-0583-C02-02 and TIC2001-0586. Dr. S. Bojanić is also supported through the "Ramon y Cajal" program from the same institution.

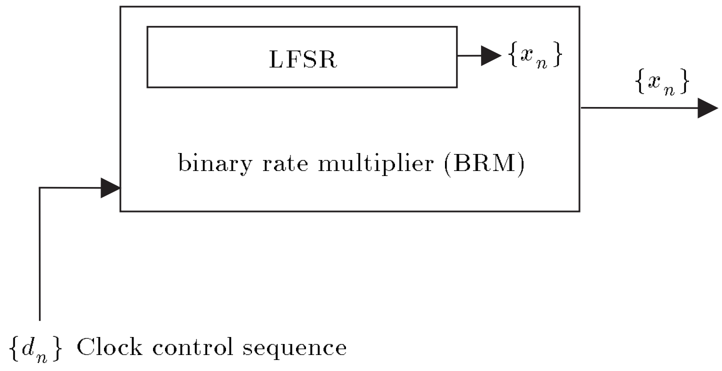


FIGURE 1. The scheme of the analyzed generator.

the edit distance with the constraint on the maximum length of the runs of deletions.

Once the set of candidate initial states is known, its element that generates the intercepted output sequence can be determined. The attack continues by determining the clock control sequence that, together with one of the candidate initial states of the LFSR, could generate the intercepted sequence [2].

The edit or Levenshtein distance is the smallest number of insertions, deletions and substitutions required to change one string into another [3]. A dynamic programming algorithm is one method for its computing and has been extensively applied in computational biology where the edit distance represents how "close" two proteins or DNA sequences may be in terms of functionality or origin [4, 5].

In this work we present the specially designed hardware that accelerates the cryptanalytic attack based on the edit distance calculation. We developed a systolic array hardware architecture that requires $O(n)$ time and $O(m)$ space, in respect with a sequential software implementation of the dynamic programming algorithm requiring $O\big(m(n-m)\big)$ time, where $m$ and $n$ are output lengths with and without decimation, respectively. Our hardware exploits the inherent parallelization of the algorithm and therefore achieves faster cryptanalytic attack.

The proposed hardware architecture was implanted in reconfigurable hardware, Field programmable gate arrays (FPGA). It offers a good compromise between software and hardware: while not as fast as integrated circuits, it can be significantly faster than software, and in small quantities is extremely cost-effective. These systems can be reconfigured, enabling the same hardware resource to be reused depending on its interaction with external components, data dependencies or algorithm requirements [6], [7].

For the implementation of the general purpose cryptanalysis devices, reconfigurable hardware has been identified as a suitable technology with the potential for improved performance over traditional microprocessor based approaches. Particularly, FPGA was successfully applied in exhaustive key search that is a highly parallelizable task [8].

The rest of the paper is organized as follows: In Section 2, the cryptanalytic attack is described, and in Section 3, the corresponding dynamic programming algorithm is presented. The parallelized hardware design is given in Section 4, and the results are shown in Section 5. Finally, Section 6 concludes the paper.

## 2. Cryptanalytic attack

The corresponding statistical model of the generator from Fig. 1 is presented in Fig. 2. Let $\{x_n\}$ be the binary sequence produced by the shift register $R$. Let $\{d_n\}$ be a sequence of integers, named decimation sequence, $0 \leq d_n \leq \mathcal{L}$, where $\mathcal{L}$ is given in advance.

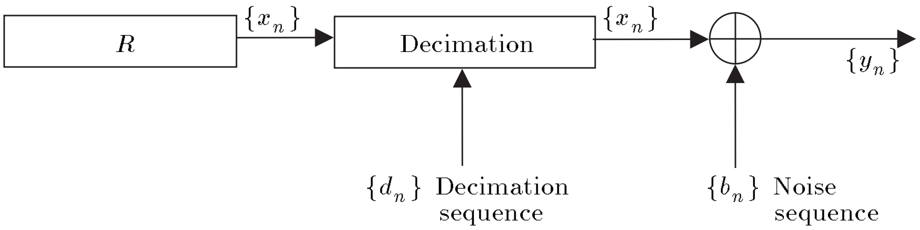


FIGURE 2. The statistical model of the analyzed generator.

In the decimation process, the sequence $\{z_n\}$ is obtained in the following way:

$$z_n = x_{f(n)}, \quad f(n) = n + \sum_{i=0}^{n} d_i, \qquad n = 0, 1, 2, \ldots \tag{2.1}$$

In the statistical model, it is supposed that $\{d_n\}$ is the realization of the sequence $\{D_n\}$ of independent and identically distributed (i.i.d.) random variables, with the probability $\Pr(D_n = i) = \frac{1}{\mathcal{L}+1}$, $0 \leq i \leq \mathcal{L}$, $\forall n$. The binary noise sequence, $\{b_n\}$, is the realization of the sequence of random i.i.d. variables $\{B_n\}$ with the probability $\Pr(B_n = 1) = p < 0.5$, $\forall n$, where $p$ is the correlation parameter.

The cryptanalyst possesses $M$ consecutive bits of the sum $\{y_n\}$ of the decimated sequence $\{z_n\}$ and the noise sequence $\{b_n\}$. His/her task is to determine the initial state of the generator that produced the $M$ intercepted bits of the sequence $\{y_n\}$.

The correlation attack described in [1] is based on the edit distance measure with the constraint on the maximum length of the runs of deletions. Let $X$ and $Y$ be two binary sequences of lengths $N$ and $M$, respectively. Let us consider the transformation of $X$ into $Y$ using the elementary edit operations—substitutions and deletions. The edit distance between $X$ and $Y$ used in [1] is defined as the minimum number of elementary edit operations needed to transform $X$ into $Y$, where the number of consecutive deletions is $\leq E$.

The edit distance defined above can be determined in a recursive way, by filling the edit distance matrix (see, for example, [9]). In the edit transformation, if $e$ represents the number of deletions and $s$ represents the number of substitutions, then the edit distance between the prefix $X_{e+s}$ of the sequence $X$ and the prefix $Y_s$ of the sequence $Y$ is given by the following recursion:

$$W[e,s] = \min\Big\{W[e-e_1,\, s-1] + e_1 d_e + d(x_{e+s}, y_s) : \\ \max\big\{0, e - \min\{N-M,\, (s-1)E\}\big\} \leq e_1 \leq \min\{e, E\}\Big\}$$

$$s = 1, \ldots, M\,, \quad e = 1, \ldots, \min\{N-M,\, sE\}\,, \tag{2.2}$$

where $d_e$ represents the elementary edit distance associated with a deletion, $d(x,y)$ represents the elementary edit distance associated with the substitution of the symbol $x$ by the symbol $y$, and $E$ is the maximum number of consecutive deletions.

The first phase of the attack consists of the following steps [1]:

1° Determine the length $N$ of the output sequence of the LFSR $R$ without decimation. $N$ depends on the maximum number of consecutive deletions $E$. For example, if $E = 1$ then $N = 3M/2$, which is the mathematical expectation of $N$ in that case. The threshold $T$ necessary for the classification of the initial state checks of $R$ is also determined. For this to be carried out, the probability of "false alarm" $P_f$ as well as the probability of "missing the event" $P_m$, are selected in advance. The probability $P_f$ determines the mathematical expectation of the cardinality of the set of candidate initial states of $R$. The threshold is computed by checking $1/P_m$ initial states of $R$, selected at random. For each of them, the edit distance defined above between the output sequence generated by the actual initial state without decimation and the intercepted output sequence is calculated. The threshold is selected to be greater than the maximum edit distance value obtained in this process.

2° For every possible initial state of $R$, not used in the step 1°, the constrained edit distance between the output sequence of length $N$ and the intercepted sequence of length $M$ is computed. All the initial states, whose edit distance is less than the threshold $T$, are included in the set of candidate initial states of $R$ that could produce the intercepted sequence.

For every candidate initial state of the LFSR $R$, it is necessary to determine its corresponding clock control sequence. Since this clock control sequence is not unique, all these sequences are to be checked by means of the additional postfix of the intercepted sequence.

The reconstruction of the necessary clock control sequences can be carried out by determining optimal and/or suboptimal paths of adequate length over the edit distance matrix.

In order to reconstruct the optimal and the necessary suboptimal paths, we first need to determine the points in the column pl, through which the optimal paths that start at $W[N-M, M]$ pass. To carry out this, every cell $W[e, s]$ has, besides the value $c$ of the edit distance, four associated vectors:

1° The vector of "primary" pointers vp to the cells $W\big[\mathrm{vp}(1), s-1\big], \dots,$ $W\big[\mathrm{vp}(k), s-1\big]$ from which it is possible to arrive to the cell $W[e, s]$ with the minimum weight increment.

2° The vector of "updated" pointers vu to the cells $W\big[\mathrm{vu}(1), \mathrm{pl}\big], \dots,$ $W\big[\mathrm{vu}(l), \mathrm{pl}\big]$, through which it is possible to arrive to the cell $W[e, s]$ (see for example [12]).

3° The vector of pointers ve to the cells $W\big[\mathrm{ve}(1), s-1\big], \dots, W\big[\mathrm{ve}(j), s-1\big]$ from which it is possible to arrive to the cell $W[e, s]$ regardless of the weight increment.

4° The vector of values vj of the edit distances corresponding to the elements of the vector ve. The cardinality of this vector is also $j$.

Obviously, ve contains the whole vp. In this paper, we keep them separated for easier understanding.

The numbers $k$, $l$, and $j$ depend on $E$ and pl.

The matrix $W$ is filled by means of the following algorithm:

**Algorithm**

**Input:**

- The sequences $X$ and $Y$ of lengths $N$ and $M$, respectively.
- The length pl of the clock control sequence necessary to reconstruct the initial state of the subgenerator that generates it.
- The maximum length $E$ of runs of deletions.
- The elementary distance $d_e$ associated with the deletion of a symbol.
- The elementary edit distance $d(x, y)$ associated with the substitution of the symbol $x$ by the symbol $y$, $\forall x, y$.

**Output:**

- The matrix $W$ of edit distances with the vectors vp, vu, vj, and ve associated with every cell.

**Initialization:**

`W[e,s].c =` $\infty$, $e = 0, \ldots, N - M$, $s = 0, \ldots, M$ :
The vectors `vp,vu,vj`, and `ve` associated with every cell `W[e,s]` are empty.
`W[0,0].c = 0`:

```
// The row 0 of the matrix W:
for s = 1 to M {
  W[0,s].k = 1;
  W[0,s].c = W[0,s-1].c + d(X[s],Y[s]) ;
  W[0,s].vp[1] = 0;
}
```

**Main loop:**

```
for s = 1 to M {
  for e = 1 to min{N-M,s*E} {
```

Let `q` be the minimum value of the expression

```
W[e-e1,s-1].c + e1*de + d(X[e+s],Y[s]),        (3)
e1 = max{0,e-min{N-M,(s-1)*E}} ... min{e,E}.
```

Let `nq` be the number of values of `e1` for which the expression (3) takes the value `q`. Then

```
W[e,s].c = q;
W[e,s].k = nq;
```

The vector `W[e,s].vp` is filled with `nq` values of the expression `e - e1` corresponding to the values `e1` for which the expression (3) takes the value `q`.

```
  }
```

The vector `W[e,s].vj` is filled with all the values (not necessarily minimum) of the expression (3).

The vector `W[e,s].ve` is filled with the values `e - e1` corresponding to the values of `W[e,s].vj`.

**Determining updated pointers**

```
  if s = pl + 1 then {
   for e = 0 to min{N-M,s*E}
     W[e,s].vu = W[e,s].vp;
  }
```

`else if s > pl + 1` {

For every element of the vector `W[e,s].vp`, `e = 0 ... min{N-M,s*E}` the corresponding set of updated pointers is determined. These are placed into the vector `W[e,s].vu` in the following way:

All the elements of `W[W[e,s].vp[i],s-1].vu`, are placed into the vector `W[e,s].vu`, deleting those that are repeated, `i = 1 ... W[e,s].k`.

} □

}

## 3. Dynamic programming algorithm

In general, the edit distance is the total cost to alter a pattern and its occurrence in the text in order to make them equal. The operations that can be used to do this are insert, delete and substitute; and each has a cost associated with it. These operations are applied to simple characters in either the pattern or the text. If each operation has a cost of 1, then the edit distance is the number of operations. If the operations all have different costs, then the general edit distance is the number of operations multiplied by their individual costs.

The edit distance between two strings can be calculated using a dynamic programming algorithm. The dynamic programming algorithm for computing the edit distance compares two strings $S$ and $T$ by performing a pairwise comparison of each element in the two strings, then computing a score to determine the similarity of the two strings. In a general case, an edit-distance matrix is filled with values that depend on a comparison between elements of $S$ and $T$ and on neighbor values and the deletion, insertion and substitution costs. All possible pair-wise comparisons are made while filling the edit-distance matrix.

In analyzed cryptanalytic attack only deletions and substitutions are involved and only one consecutive deletion is assumed ($E = 1$). The strings $S$ of length $M$ and strings $T$ of length $N = 3M/2$ were analyzed. Figure 3 gives a two-dimensional representation of the constrained edit-distance. The two strings $S$ and $T$ are compared, and intermediate values $a$ and $b$ are used to produce the intermediate result, $d$. This calculation is repeated once for each pairwise element comparison. The elements of $S$ are used to compute the values of the columns matching the index of the element, thus $S_i$ is used to compute column with index $i$. The same applies to $T$, but in this case, an element $T_j$ is used to compute the values of the antidiagonal with index $j$.

The matching algorithm itself is given in Figure 4. The values of a plus a deletion penalty del and b are compared. If the elements in the two strings are not the same, then the both values are increasing by a substitution penalty sub. The result value d is determined by taking the minimum of the two resulting values.In our case there are no insertion penalties.

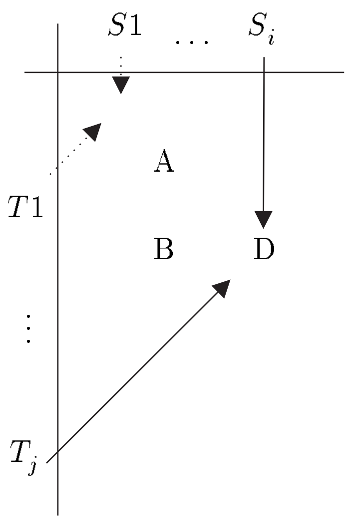


FIGURE 3. Matrix representation of the algorithm.

$$d = \min \begin{cases} a + \mathrm{del} + \begin{cases} 0 & \text{if } S_i = T_j\,, \\ \mathrm{sub} & \text{if } S_i \neq T_j\,, \end{cases} \\ b + \begin{cases} 0 & \text{if } S_i = T_j\,, \\ \mathrm{sub} & \text{if } S_i \neq T_j\,. \end{cases} \end{cases}$$

FIGURE 4. The dynamic programming algorithm.

In the case where string $S$ is of length $m$ and string $T$ is of length $n$, the algorithm begins by comparing $S_0$ and $T_0$ and proceeds onward until a final value of $d$ is calculated at the comparison of $S_m$ and $T_n$. This value of $d$ is the edit distance between the two strings.

The dynamic programming algorithm requires $O\big(m(n-m)\big)$ time for a sequential implementation. Yet the nature of the dependencies allows a systolic implementation, requiring $O(n)$ time and $O(m)$ space.

It is known that dynamic programming algorithms map well onto systolic arrays [10]. In particular, there is a tremendous potential for concurrency in the construction of an edit distance table; the entries on a given 45 degree diagonal can be computed simultaneously because they depend only on values up and to the left. This parallelism can be realized with a systolic array. All the values on 45 degree diagonals will be computed at the same time and all the values on $-45$ degree diagonals will be computed using the same processor. In this way the quadratic time serial algorithm becomes a linear time parallel algorithm requiring a linear number of processors.

## 4. Hardware design

Field programmable gate arrays (FPGA) are integrated circuits containing arrays of logic blocks and programmable interconnections between these blocks. The logic blocks can be configured to implement simple or complex logical functions and can be changed as required. Example functions are registers, adders, and multipliers. The programmable interconnection permits the construction of even more complex functions and/or systems.

FPGAs are highly adaptable and are able to handle complex circuits, with the ability to easily update and correct circuits. Each FPGA can quickly process large amounts of data in parallel. The basic architectures for a configurable component composed of multiple FPGAs are very similar, or identical, to those used for parallel processing systems. In many cases, a configurable computing system is the hardware equivalent of a software parallel processing system.

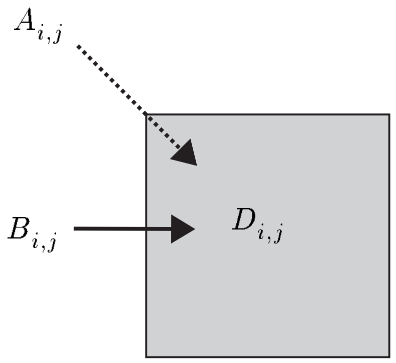


FIGURE 5. Locality property of edit-distance algorithm.

The particular data dependence of the algorithm makes possible the extensive use of parallelism. The elements in the distance matrix depend on the upper-left and left neighbours (see Fig. 5). Thus, it is viable to compute the elements of any antidiagonal in parallel at each time step. This is displayed in Fig. 6, where we can see that once $D_{1,1}$ is calculated, $D_{1,2}$ and $D_{2,2}$ can be computed simultaneously. The locality of the algorithm makes it suitable for systolic array implementations [10].

From Figure 6 we can conclude that we need a total of $M$ processors for the hardware implementation, given that each processor computes the distances of a particular row (as indicated in the figure) and assuming that $M \leq N$.

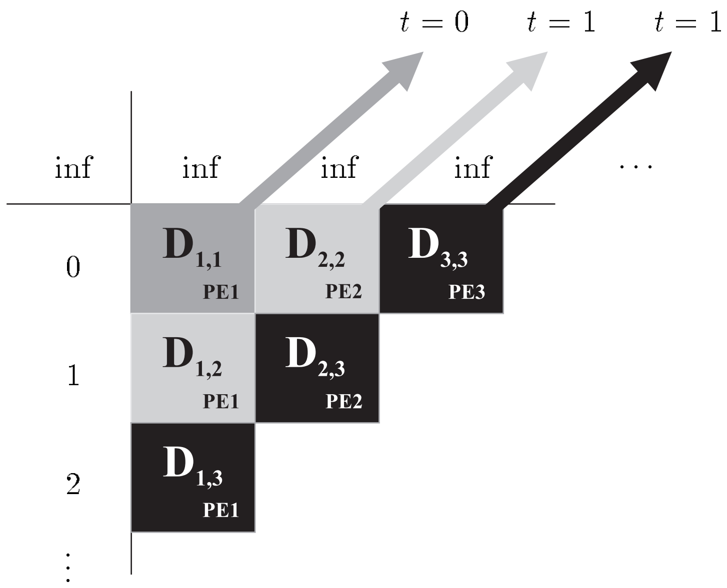


FIGURE 6. Parallelization of edit-distance algorithm.

Fig. 7 shows a diagram of the inner PEs blocks. The shadowed area includes the distance computation, given the values of the upper-left (a) and left (b) neighbours. As said before, each PE computes the distances of a column; PE1 computes the distance of the elements of the first column, PE2 deals with the second column, and so on. Each PE receives as inputs the last distance computed by its predecessor and $T$. PE number $X$ will need only the value $S(X)$ during the whole process, and this value is stored before starting the matching process. The distance $D_{i,n}$ corresponds with the value $b$ (left distance). After one clock cycle this value will become $a$, since by then it will be the upper-left distance.

Fig. 8. shows the interconnections between PEs. The signal *load* is used to preload the values of $S$ and it takes $M$ clock cycles. The signal *start* is a pulse propagated from left to right, which initializes each PE and indicates that the beginning of the edit-distance computation. After $M$ cycles the last PE will output the edit-distance. Notice that $T$ is input to all processors simultaneously.

In a sequential system the edit-distance computation time is $O(M \times N = 3M^2/2)$. The parallel system presented can compute the edit-distance in $O(N = M/2)$, which notably boosts the performance of the computation.

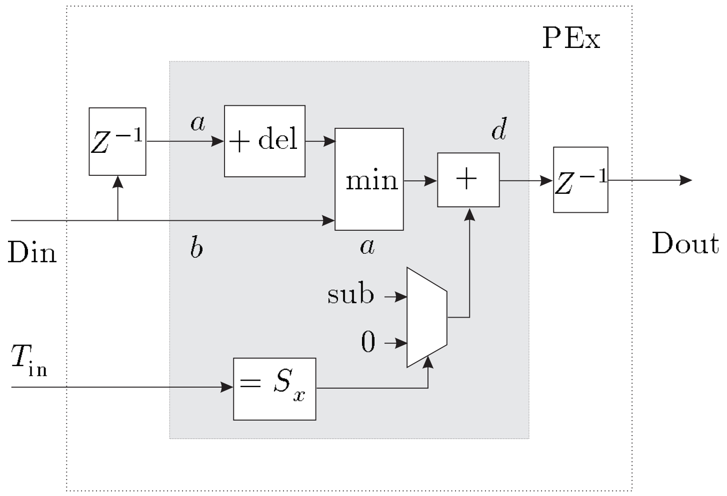


FIGURE 7. Inner blocks of PE.

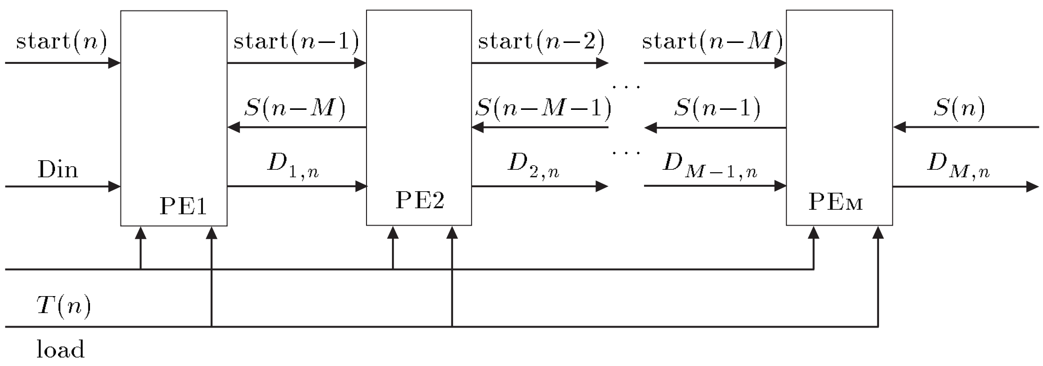


FIGURE 8. Interconnection between PEs.

## 5. Results

Table 1. shows some implementation results. All implementations were aimed on the FPGA device XC2V6000 (speed grade -6) from Xilinx. A maximum number of 1300 PEs were implemented on a single FPGA device. The occupied area was a 99 % of the total area (33792 slices) and the maximum achievable clock speed was 50 MHz.

The first column displays the lengths of strings $S$ and $T$. The second column contains the number of PEs that the device can hold. This number is multiplied

TABLE 1. The results of the implementation

| $M \times N$ | PEs per device | Strings/secs (HW) | Strings/secs (SW) |
|---|---|---|---|
| $650 \times 975$ | $650 \times 2$ | 51,282/102,560 | 6,837 |
| $1300 \times 1950$ | $1300 \times 1$ | 25,641 | 3,418 |

by the number of times a complete systolic array can be fitted in the device. If there are more than one array in the FPGA, two time results are given, one for the case of only one systolic array, and another for the totality of arrays working in parallel.

The performance of the system is measured in terms of strings processed per second. In the third column we find the result for the hardware system, and in the fourth the results for a sequential software implementation. The software version of the algorithm was tested on a Pentium–IV 1.7 GHz.

In a general case, the distance computed at each PE requires $\log_2$ (length of $N$) bits. Thus, the greater $T$, the more resources that each PE will need. But for concrete cases, the selection of the algorithm costs may lead to optimal implementations [10]. As an example, the Smith-Waterman algorithm (with costs $C_s = 2$, $C_i = 1$ and $C_d = 1$) enables to apply a simplification to the distance computation that can produce a dramatic improvement in both area and speed [10, 4, 5]. However, the costs applied in our implementation do not allow applying such simplifications. We see as a future research line to study the viability of using a set of costs capable of achieving area savings as well as speed increment.

## 6. Conclusion

In this work, the specially designed hardware was realized in order to accelerate cryptanalytic attack on a class of stream ciphers that use linear feedback shift registers with irregular clocking. The cryptanalytic attack is based on calculation of edit distance of two sequences and the calculation is performed by the means of dynamic programming algorithm. Our design exploits inherent parallelism of the algorithm and the results indicate a significant performance gain over the software implementation. The systolic array architecture was realized in reconfigurable hardware on the FPGA device XC2V6000 (speed grade $-6$) from Xilinx. Further research should be oriented to the study of using a set of edit distance costs that will enable further area minimizing as well as speed increase in FPGA hardware realization.

## REFERENCES

[1] GOLIĆ, H.—MIHALJEVIĆ, M.: *A generalized correlation attack on a class of stream ciphers based on the Levenshtein distance*, J. Cryptology **3** (1991), 201–212.

[2] PETROVIĆ, S.—FÚSTER-SABATER, A.—DURÁN, R.: *The use of edit distances in cryptanalysis*, in: Proc. 5th World Multiconference on Systemics, Cybernetics and Informatics (SCI 2001), Orlando, USA, 2001, pp. 511–516.

[3] LEVENSHTEIN, V.: *Binary codes capable of correcting deletions, insertions and reversals*, Doklady Akademii Nauk SSSR **163** (1965), 845–848.

[4] PUTTEGOWDA, K.—WOREK, W.—PAPPAS, N.—DANDAPANI, A.—ATHANAS, P.: *The a run-time reconfigurable system for gene-sequence searching*, in: Proc. International VLSI Design Conference, 2003.

[5] GUCCIONE, S. A.—KELLER, E.: *Gene matching using Jbits*, in: Proc. Field-Programmable Logic and Applications, 2003, pp. 1168–1171.

[6] MARCHAL, P.: *Field programmable gate arrays*, Communication of the ACM **42** (1999), 57–59.

[7] MENEZES, A. J.—VANOORSCHOT, P. C.—VANSTONE, S. A.: *Handbook of Applied Cryptography*, CRC Press, Singapore, 1996.

[8] TSOI, K. H.—LEE, K. H.—LEONG, P. H. W.: *A massively parallel RC4 key search engine*, in: Proc. 10th Annual IEEE Sympos. Field-Programmable Custom Computing Machines, 2002, pp. 13–21.

[9] OOMMEN, B.: *Constrained string editing*, Inform. Sci. **40** (1986), 267–284.

[10] LIPTON, R.—LOPRESTI, D.: *A systolic array for rapid string comparisons*, Chapel Hill Conference on VLSI, Computer Science Press, 1985, pp. 363–376.

[11] ANDREWS, G.: *The Theory of Partitions*, Addison-Wesley, New York, 1976.

[12] HIRSCHBERG, D.: *Serial computations of Levenshtein distances*, in: Pattern Matching Algorithms, (A. Apostolico, Zvi Galil, eds.), Oxford University Press, Oxford, 1997, pp. 123–141.

Received October 14, 2003

*S. Bojanić, G. Caffarena, O. Nieto-Taladriz*
*Dpto. de Ingeniería Electrónica*
*Universidad Politécnica de Madrid*
*E.T.S.I. de Telecomunicación*
*Ciudad Universitaria s/n*
*28040 Madrid*
*SPAIN*

*E-mail*: slobodan@die.upm.es
gabriel@die.upm.es
nieto@die.upm.es

*S. Petrović*
*Insituto de Física Aplicada*
*Consejo Superior de Investigaciones Científicas*
*Serrano 144*
*28006 Madrid*
*SPAIN*

*E-mail*: slobodan@iec.csic.es