Parallel Calculation of Volcanoes for Cryptographic Uses *

Santi Martínez[1], Rosana Tomàs[2], Concepció Roig[1],
Magda Valls[1] and Ramiro Moreno[2]

[1]Universitat de Lleida
Dept. d'Informàtica i Enginyería Industrial
Jaume II, 69 25001 Lleida, Spain
{santi, roig, magda}@eps.udl.es

[2]Universitat de Lleida
Dept. de Matemàtica
Jaume II, 69 25001 Lleida, Spain
{rosana, ramiro}@eps.udl.es

Abstract

Elliptic curve cryptosystems are nowadays widely used in the design of many security devices. Nevertheless, since not every elliptic curve is useful for cryptographic purposes, mechanisms for providing good curves are highly needed. The generation of the volcano graph of elliptic curves can help to provide such good curves. However, this procedure turns out to be very expensive when performed sequentially. Hence, a parallel application for the calculation of such volcano graphs is proposed in this paper. In order to obtain high efficiency, a theoretical analysis is provided for obtaining an accurate granularity and for giving the appropriate number of tasks to be created. Experimental results show the benefits obtained in the speedup when executing the application in a cluster of workstations with message-passing for the generation of different volcano graphs. By the use of simulation, we study the scalability of the implementation and show that a speedup of more than 80 can be achieved in some cases.

1. Introduction

Nowadays, cryptographic systems are an essential tool to guarantee security in communications. Indeed, there are many devices in daily use that need to incorporate cryptographic functionalities. Examples of these devices are contactless cards, smart cards or devices for mobile communications [3, 11]. This type of devices must send short keys so that their reception and authentication are fast enough. In addition, the restrictions of memory space, capacity of computation and bandwidth of these devices discourage the use of long keys. Another problem that appears is the fact that a huge number of keys are needed, since they have to be changed frequently to prevent attackers from obtaining them and making fraudulent use.

Such problems can be solved using elliptic curves, one of the peak tendencies within the field of cryptography [1, 2, 7]. This is because cryptographic techniques based on elliptic curves allow the use of smaller keys (170 bits, instead of 1024 bits in conventional cryptography) to provide a similar level of security.

However, not every elliptic curve is cryptographically secure. Thus, it becomes necessary to know which of them are appropriate for cryptographic uses and which are not. Therefore, it would be useful to classify them according to their security and cryptographic properties. A structure that allows the curves to be classified in such a way is a graph called *volcano* [5].

Given an elliptic curve with good cryptographic characteristics, the construction of the volcano to which it belongs will give us new curves with the same security properties. Nevertheless, the construction of this structure is computationally hard [9], since the computation of the adjacent nodes of each node in the volcano is CPU-intensive. This fact has motivated us to confront the parallelization of the problem.

In this paper we present the parallelization strategy for the problem of generating volcanoes in a distributed environment based on the message passing paradigm. As far as we know, no parallelization of this algorithm has ever been reported in the literature. An experimentation process has been carried out both in a cluster of workstations in order to evaluate the speedup of the parallel application that has been implemented, and also in a simulation environment to evaluate the scalability. The results show that significant improvements in the speedup are achieved in both approaches,

*This work is supported by the project TIC2003-09188 of the MCyT.

1-4244-0054-6/06/$20.00

reaching a speedup of more than 5 in the volcanoes used in the cluster, and a speedup of more than 80 in the volcanoes used in the simulation.

An analytical study of the application has also been carried out to determine the parameters that affect the granularity of the parallelization and that have crucial influence in the performance of the execution.

The rest of the paper is structured as follows. In Section 2, we introduce the volcano graph of elliptic curves, as well as its properties. In Section 3, the parallelization strategy that was followed for the generation of the volcano graphs is presented, along with an analysis of the application that predicts a theoretical appropriate number of tasks to be used. Section 4 presents the experimentation results that were obtained when executing the parallel application on a cluster of workstations and in a simulation environment. Finally, Section 5 outlines the main conclusions.

2. Volcanoes

In this section, the structure of the volcano graph is introduced along with its main properties [5].

Given a prime number l, an l-volcano is defined as a graph that contains a unique cycle, called *crater*, and such that $(l-1)$ l-ary trees with the same height h hang from each node of that cycle. The leaves of these trees are called the *floor* of the l-volcano, while the rest of the nodes of every tree configure the *volcanoside*. Each node of the l-volcano, except those on the floor, has $l+1$ edges. The general structure of an l-volcano graph is sketched in Figure 1.

This volcano graph can be used to represent classes of elliptic curves on a field $\mathbb{F}_p$, with p prime. An elliptic curve consists of all the points of the form $P = (x, y) \in \mathbb{F}_p \times \mathbb{F}_p$ that satisfy an equation of the type:

$$y^2 = x^3 + ax + b, \text{ with } a, b \in \mathbb{F}_p$$

whose discriminant $\Delta = -16(27b^2 + 4a^3)$ is non null.

Each elliptic curve can be located in a unique node of its l-volcano. Then, the adjacent nodes of a given curve correspond to their l-isogeneous curves [1] .

The cryptographic security of an elliptic curve is directly related to the number of points of the curve, denoted by the *cardinal* of the curve. Since isogeneous curves have the same cardinal, if a given curve fulfils the requirements for cryptographic security, the curves obtained visiting the nodes of its l-volcano will have the same properties as the original.

[1] An isogeny between two elliptic curves E and E' is a rational morphism $\mathcal{I} : E \to E'$ [1]. Then, an l-isogeny is an isogeny whose kernel is an order l subgroup. Hence, two curves are called l-isogeneous if there exist an l-isogeny between them. Moreover, the isogenies between elliptic curves can be classified into horizontal, ascending or descending. This hierarchy can be straightforwardly represented through the levels of the volcano graph.

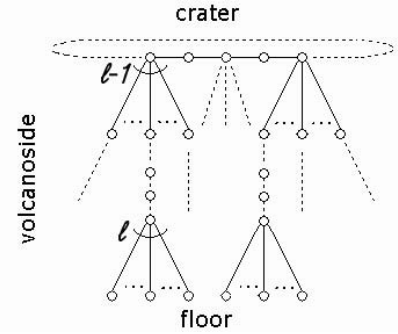

Figure 1. Volcano graph.

The structure of this volcano graph was firstly analyzed by M. Fouquet [4]. In general, the l-volcano graph is usually flat, since while the side of the l-volcano has few levels (i.e., the distance between a crater node and a floor node is frequently less than 10), the crater has a high number of nodes. In curves used in cryptographic applications the crater can have millions of nodes.

Therefore, it would be useful to have an algorithm that allows us to generate the l-volcano graph in order to have a set of useful elliptic curves available. However, the sequential computation of the nodes of the volcano turns out to be computationally hard [9], due to the fact that visiting each neighbour node is costly. Thus, we propose an approach that consists of providing a parallel algorithm that generates all the elliptic curves of the volcano by overlapping the generation of the different parts of the volcano graph.

3. Parallel implementation

Given a *good* (i.e. cryptographically secure) elliptic curve, the process of generating the nodes of its volcano graph consists of three main stages:

1. Find a path to the crater: The procedure ascends from the node where the curve is located (whose level, or distance to the floor, is not always known) until reaching the crater of the volcano.
2. Visit all the nodes of the crater.
3. Explore the trees hanging from the crater nodes.

On the one hand, reaching the crater is a computational simple task because the height of the l-volcanoes is usually small. On the other hand, the traversing of the crater and the descending of the trees of its

side can imply a calculation of the order of millions of nodes and, as a consequence, a high computational cost. Therefore, parallelization has been carried out to perform the second and third stages.

It is important to point out that the calculation of the adjacent nodes from a given node is costly [9], since obtaining each node implies the calculation of modular square roots for $l = 2$ or the resolution of a polynomial of degree $\frac{l^2-1}{2}$ in the finite field $\mathbb{F}_p$ for $l > 2$.

3.1. Parallel algorithm

The parallel algorithm was developed with the message passing interface MPI [8], in its implementation LAM 7.1. The parallelization strategy is based on the *master-worker* mechanism by exploiting a mixed functional and data parallelism approach [10].

The mechanism implemented in the algorithm consists of obtaining the nodes of the crater and, concurrently, generating the tree that hangs from each one of these nodes. To do that, three different tasks are defined: *(a)* Master task (T_M) that coordinates the global operation of the application, *(b)* Crater tasks (T_C^i) that obtain the nodes of the crater and, *(c)* Side tasks (T_S^j) that generate the nodes of the volcanoside.

The interaction structure of these tasks is shown in Figure 2. Notice that, in particular, the application consists of one master task T_M, two crater tasks T_C^i and n side tasks T_S^j. The appropriate number of T_S^j tasks to be generated depends on the characteristics of the volcano graph. The "a priori" choice of an accurate number of tasks is crucial to obtain a properly balanced parallelization, since it is one of the parameters that affects the granularity. An analytic study of this granularity is provided in the next subsection.

The functionality of each task is described below.

Master task (T_M). It finds the first two nodes of the crater, sends the information to be processed by the rest of tasks and composes the whole graph from the

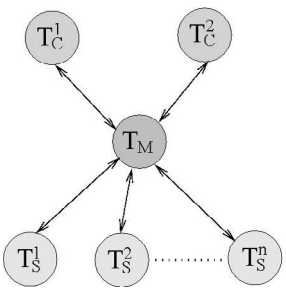

Figure 2. Task graph structure.

```
Initialize_parameters ()
send (Crater_info, T_C^i)
send (Side_info, T_S^j)
while not crater_explored
     receive (pack_nodes, source)
     if source == T_C^i
          send (crater_nodes, T_S^j)
     else
          Store (nodes)
     end if
end while
send (Finalization, T_C^i)
while T_S^j active
     receive (pack_nodes, T_S^j)
     Store (nodes)
end while
send (Finalization, T_S^j)
End
```

Figure 3. Pseudocode of Master Task (T_M).

nodes generated by the different tasks. The pseudocode of task T_M is shown in Figure 3.

It can be seen in the algorithm that task T_M sends the information to the crater and the side tasks so that they can begin their calculation. Then, it enters an iteration in which the crater nodes are received and sent back to the side tasks so that they compute the tree that hangs from each crater node.

Once the traversing of the crater is finished, the master task activates the finalization of the T_C^i tasks and continues until all the calculated nodes of the volcanoside are gathered, in which case it activates the finalization of the T_S^j tasks.

Crater task (T_C^i). Given two nodes calculated by the master task, the traversing of the crater is performed in two directions. Due to this fact, the creation of only two crater tasks T_C^1 and T_C^2 is considered. It is important to note that the calculation of each node depends on the previous one, therefore the creation of more crater tasks would give no benefit.

The pseudocode in Figure 4 shows that each T_C^i calculates a configurable number of crater nodes every time and then sends them to the master task. The length of the package of crater nodes (P_Len) is crucial for the performance of the application since it defines one of the factors that affects the granularity: the minimum amount of work assigned to each side task.

Since the crater is a cycle, the process of traversing the crater will finish when both tasks T_C^1 and T_C^2 are intersected (i.e., they have calculated a common node). This situation will be detected by task T_M, in which case it will activate the finalization of the T_C^i tasks.

The main function involved in this task is

```
receive (Crater_info, T_M)
while not finalization_T_C^i
     for n in 1 to P_Len
          node = Calculate_new_node
          Add (node, pack_nodes)
     end for
     send (pack_nodes, T_M)
end while
End
```

Figure 4. Pseudocode of Crater Task (T_C^i).

`Calculate_new_node`. This function takes as input the current and previous crater nodes and outputs the next crater node to be visited. Notice that the current node has l adjacent non-visited nodes, one of which is on the crater. Distinguishing the crater node among the others is not immediate. For such a purpose the algorithm considers l paths beginning at each of the adjacent nodes, and visits subsequent volcanoside nodes until reaching the floor. The length of the path starting at the crater node will be greater than the others. Hence, the crater node will be distinguished.

Side task (T_S^j). Each package of nodes of the crater calculated by a crater task is sent to a side task T_S^j, that computes the tree that hangs from each node of the received package. Once all the trees have been calculated, they are sent to task T_M. Figure 5 shows the pseudocode that corresponds to a T_S^j task.

Notice that the proposed parallel implementation does not fix the number of side tasks T_S^j to be created. The number of side tasks must be established by the programmer depending on the characteristics of the volcano graph. Again, an accurate tuning of this parameter would revert on a better efficiency. Thus, in the next subsection we carry out an analytical study in order to find, prior the execution for each volcano, the number of side tasks that provide an appropriate granularity of computation versus communication that guarantee a load balanced execution.

```
receive (Side_info, T_M)
while not finalization_T_S^j
     receive (crater_nodes, T_M)
     for each node ∈ crater_nodes
          Build_tree (node)
     end for
     send (trees, T_M)
end while
End
```

Figure 5. Pseudocode of Side Task (T_S^j).

3.2. Analysis of granularity

In order to obtain better results with the parallel application we need to appropriately adjust some parameters that affect the granularity. This granularity may change depending on the length of the package that the crater tasks send to the master task and on the number of side tasks T_S^j being generated.

Because of that, the core of an efficient implementation heavily lies on finding the appropriate package length as well as the number of side tasks which must be established at the beginning of the execution.

Moreover, since l and h are the only parameters of the volcano which are known a priori, the criteria to determine the package size and the number of side tasks should depend only on them.

Package length. A proper length of the package should be selected attending the number of nodes that would hang from each crater node. As it is logical, greater number of hanging nodes would imply taking shorter packages.

The number of hanging nodes from a crater node is $l^h - 1$ (see Table 1). The package length has been selected according to the expected maximum number of volcanoside nodes for each l value. Previous experimentation suggest that a suitabe amount of nodes to be generated for each side task should be between 10^6 and 10^7. For instance, in $l = 5$ the expected maximum number of volcanoside nodes is $390624 \cdot P\_Len$, so taking $P\_Len = 20$ the previous thresholds are satisfied.

Nevertheless, implementation issues suggest a minimum and maximum value for P_Len. On the one hand a minimum of 2 nodes is considered, since package of length 1 need some extra work to discriminate the isogenies of the crater. On the other hand a maximum of 200 nodes is established, because if the package length were larger there would be more overlapping at the intersection of the two crater tasks when computing the final nodes of the crater.

Table 1. Number of nodes hanging from each crater node.

h	$l=2$	$l=3$	$l=5$	$l=7$	$l=11$
3	7	26	124	342	1330
4	15	80	624	2400	14640
5	31	242	3124	16806	161050
6	63	728	15624	117648	1771560
7	127	2186	78124	823542	19487170
8	255	6560	390624	5764800	214358880
P_Len	200	200	20	2	2

Number of side tasks. The goal is to create a number of side tasks N_Ts in such a way that the work to be carried out for each crater task and each side task is load balanced.

The analysis of the computations that must be performed for the generation of volcano graphs allowed us to derive an analytical expression to calculate beforehand the appropriate number of side tasks to be created, in order to obtain the maximum efficiency in the execution. The reference computation unit is the necessary work for the generation of the adjacent nodes of a given node.

On the one hand, recall that, as commented before, determining the next node in the crater implies visiting l paths towards the floor. Hence hl nodes must be generated at each step. Then, taking into account that there are two crater tasks, each of them visiting $c/2$ crater nodes, the total amount of work assigned to each T_C^i task is $hlc/2$, where c is the number of crater nodes.

On the other hand, the amount of work performed by a side task should be evaluated. Firstly, notice that, visiting the nodes that hang from each crater node would imply the computation of the adjacent nodes of l^{h-1} nodes. Hence, the total work would be $l^{h-1}c$. So, a balanced distribution of the computation between the side tasks, suggests that the amount of computation to be carried out by each T_S^j is $l^{h-1}c/N\_Ts$.

Therefore, trying to equilibrate the amount of work assigned to the crater and side tasks, we obtain that the suitable number of side tasks to be created will be:

$$\begin{aligned} hlc/2 &= l^{h-1}c/N\_Ts \\ N\_Ts &= 2l^{h-2}/h. \end{aligned}$$

As can be observed, the N_Ts value depends only on l and h as required, which are two parameters that are known beforehand.

The experimentation will confirm the suitability of this number of tasks obtained analytically.

4. Experimentation results

In this section we conducted an experimentation process, aimed at evaluating the speedup that was achieved when executing the parallel application of generation of volcanoes. This has been performed in two different environments: *(a)* Message-passing platform with the implemented application for the specific case of $l = 2$ and, *(b)* Simulation environment using the modelled parallel application of calculation of volcanoes with greater values of l, in order to study the scalability of the proposed approach.

In the next subsection, we show the experimentation results that were obtained with both approaches.

4.1. Message passing platform

The system used for the executions was a cluster with 16 nodes, where each node consisted of a 3 GHz Pentium IV processor with 2 GB of RAM memory. The interconnection network was a Gigabit Ethernet.

The parallel application was implemented to generate *2-volcanoes* with different heights. Table 2 summarizes the characteristics of the volcanoes that were generated with the following parameters:

- Volcano height (h). Number of edges from the crater to the floor.
- Number of nodes of the crater (N_Cr).
- Total number of nodes of the volcanoside, including the floor (N_Sd).
- Sequential time (S_Time), that corresponds to the obtained time, in minutes, when executing the application on a single processor.

In order to evaluate the accurate number of side tasks, different executions were done for each volcano varying the number of side tasks T_S^j. The total number of tasks that the application had in each case corresponded to the number of side tasks plus three, since there was a master task and two crater tasks. Each execution was done assigning one task per processor.

In this case, it was chosen to use a crater package length of two hundred nodes (as was shown on Table 1) to provide the application with a suitable granularity to be executed in a cluster.

Figure 6 shows the evolution of the speedup based on the number of side tasks that were created for each of the tested volcanoes, and the efficiency obtained.

Firstly, it can be observed that, in general, until a certain threshold in the number of processors, the speedup increases, and then it stabilizes. The stabilization point depends on the characteristics of the volcano, so that a greater height and a longer crater, favour the obtaining of greater speedup.

Table 2. Description of the tested volcanoes.

Volcano name	h	N_Cr	N_Sd	S_Time
V_three	3	27104	189728	7.489 m
V_four	4	11121	166815	5.209 m
V_five	5	13164	408084	9.516 m
V_six	6	5457	343791	5.748 m
V_seven	7	41052	5213604	60.370 m
V_eight	8	788	200940	1.601 m

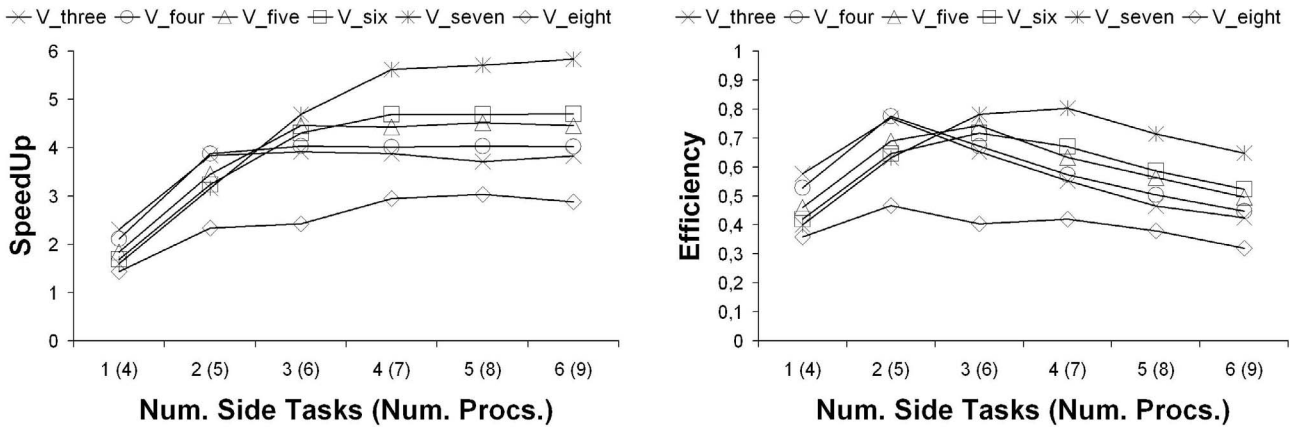

Figure 6. Speedup and efficiency for $l = 2$.

The volcano that presents higher speedup is V_seven, since it has considerable height and crater length. It is followed by volcanoes V_six to V_three in strict order of height. Nevertheless, height is not the only determining factor of the maximum speedup; as a counterexample we have V_eight, which in spite of being the one with the greatest height, it provides the poorest speedup due to its unusual small crater.

We can also observe that the efficiency peak appears close to the stabilization point of the speedup and that, as the height of the volcano increases, the efficiency graph smooths its slope. Thus, while in the case of V_three and V_four the peak is perfectly defined and there is a considerable slope near the efficiency peak, in cases V_six, V_seven and V_eight there are smoother slopes. This is due to the fact that when the stabilization point is reached at a higher number of tasks, which happens when the height is greater, adding or supressing a task has less influence in the efficiency.

In V_eight we can appreciate a poor efficiency, due to the fact that a too small crater (not much bigger than the crater package size) is more vulnerable to the overlapping problem previously commented in Section 3.2. Nevertheless, the length of the crater is not a factor that we can know a priori. Thus, the suitable choice of the number of side tasks should be based on the height of each volcano as we proposed in the analytical study of previous section.

The number of side tasks, N_Ts, that provided experimentally the best results of speedup and efficiency for each h value, was compared with the analytical value of N_Ts that was obtained before. Both values are shown, for each height, in Table 3.

It can be observed that the analytical value is quite accurate for heights varying from 3 to 6. For greater heights, the number of side tasks to be generated increases significantly. Thus, there are factors that influence more the execution as, contentions in communication, context switches, etc. that are not taken into account in the analytical expression. Due to that, as it could be expected in these cases the analytical value of N_Ts starts to differ more with the appropriate value of N_Ts obtained experimentally.

4.2. Simulation environment

The computation of the adjacent nodes in the generation of volcano graphs, implies the calculation of modular square roots for $l = 2$ or the resolution of a polynomial of degree $\frac{l^2-1}{2}$ in the finite field $\mathbb{F}_p$ for $l > 2$. Due to this fact, different values of l need different implementations of the application.

Before investing such huge efforts for the future implementations, it would be worth to study the scalability of the application when the l value increases. Thus, we carried out an experimentation based on simulation that we present below.

The execution of each application was carried out with the simulation framework *ESPPADA* [6], that works with message passing applications. The applications to be simulated were modelled based on the knowledge of periods of computation time of tasks and

Table 3. Analytic and experimental values of N_Ts for $l = 2$.

	h=3	h=4	h=5	h=6	h=7	h=8
Analytic	1.33	2	3.2	5.33	9.14	16
Experimental	2	2	3	4	6	5

communication volumes to be transferred among tasks, which can be obtained from the implementation of the real 2-volcano application adapted to the case of volcanoes with different l values. The underlying system was modelled in *ESPPADA* by defining a set of homogeneous nodes with the same characteristics of those used in the real cluster of the previous subsection.

The volcanoes tested had an l value of 3, 5, 7 and 11, and a height h of 3, 4, 5, 6 and 7. It has to be remarked that when the l value increases, the height of the volcanoes becomes lower. Thus, for each l value we have only tested heights that could be reasonably expected. The package size to be transferred between the crater tasks and the side tasks was established, in each case, based on the study shown in Subsection 3.2.

The speedup results obtained from the simulation are shown in Figure 7. Each graphic corresponds to a different l value, and reflects the speedup behaviour for each different h value of the volcanoes.

It can be seen in Figure 7 that when l is equal to 3, for the heights 3, 4, 5, 6 and 7 the speedup stabilizes at 2, 4, 9, 24 and 65 side tasks respectively, therefore the stabilization point is greater when the height is higher. Remember that a similar phenomenon of stabilization was also observed in the case of 2-volcanoes, so this is consistent with the experimental results obtained.

For $l = 5$, the height 3 stabilizes at 3 side tasks, heights 4 and 5 stabilize at 12 and 45 side tasks respectively, and height 6 stabilizes at 200 side tasks.

For $l = 7$, the stabilization for height 3 was reached at 4 side tasks, for height 4 it was reached at 24 side tasks, and for height 5 it was reached at 135 side tasks. This is not surprising, because not only a large height, but also a large l provides the volcano with a large side, that gives better parallelization results.

For $l = 11$, the stabilization phenomenon is also observable for height 3, which stabilizes at 7 side tasks, and height 4, which stabilizes at 55 side tasks.

Another fact that can be derived from the graphics is that for larger heights and higher l values, the speedup tends to the number of side tasks (number of processors minus three), until some point where it stabilizes. This is due to the fact that in these cases, the volcanoside is so big that the crater tasks have a negligible amount of work when compared to the side tasks.

The evolution of the speedup obtained in this simulation process is coherent with the results for the case of $l = 2$ that were shown in Subsection 4.1. The analytical values of N_Ts are close to the stabilization points obtained for lower heights or l values, as can be seen in Table 4. For greater heights or l values, the volcanoside packages become larger, so the communication and system overloads result in lower speedups, that translate into lower stabilization points. The experimental value of N_Ts is always lower than the analytical one, because the analysis does not take into account the overloads mentioned before.

5. Conclusions

The construction of volcano graphs of elliptic curves would agilize the generation of keys with safe cryptographic properties, because two curves in the same volcano provide equivalent levels of security.

The creation of volcanoes is an expensive computational task, since they can have several million nodes and the calculations of the adjacent nodes of a given node are CPU-intensive. For that reason, we propose a parallel approach for the construction of the volcano by exploiting a mixed, functional and data, parallelism.

From an applied point of view, determining the correct number of tasks is crucial for obtaining the best results. Since configuring the number of tasks must be done a priori, a theoretical analysis of the application was done to find an expression for this value.

Experimental results are presented for the described parallelization for the construction of *2-volcano* graphs with different heights and number of nodes. The results have been extended for the general case of *l-volcanoes* by simulation. These results confirm the study about the correct number of tasks as a good approach.

The empirical analysis shows that for a fixed l, height is the most influential factor in the maximum speedup. This is because one more level of height implies that the volcano has l times the number of nodes for craters of equal length.

A speedup of more than 5 is reached in the construction of *2-volcanoes*. The results obtained in the simulation show that significant improvements in the speedup can be reached for *l-volcanoes* with greater l,

Table 4. Analytic and experimental N_Ts.

l=3		h=3	h=4	h=5	h=6	h=7
	Analytic	2	4.5	10.8	27	69.4
	Experimental	2	4	9	24	65
l=5		h=3	h=4	h=5	h=6	
	Analytic	3.33	12.5	50	208.3	
	Experimental	3	12	45	200	
l=7		h=3	h=4	h=5		
	Analytic	4.66	24.5	137.2		
	Experimental	4	24	135		
l=11		h=3	h=4			
	Analytic	7.33	60.5			
	Experimental	7	55			

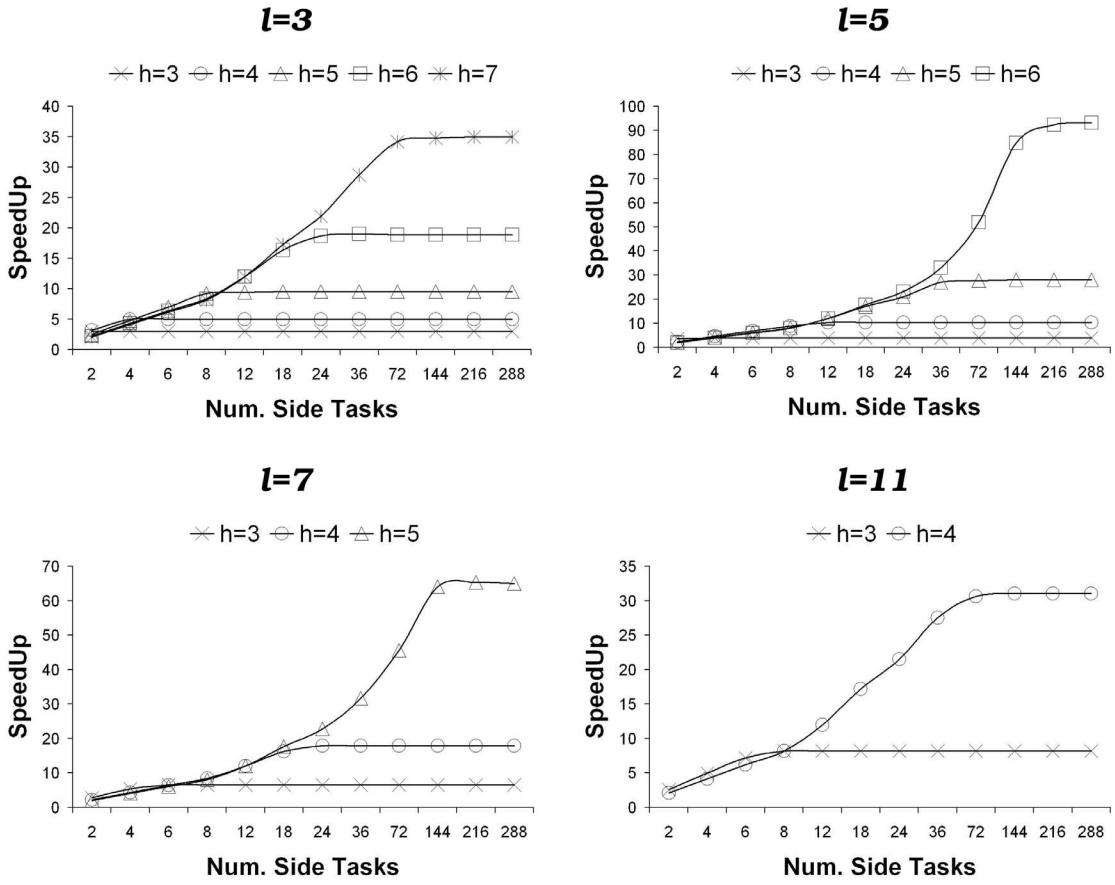

Figure 7. Speedup for $l = 3$, $l = 5$, $l = 7$ and $l = 11$.

obtaining speedups of more than 80. This confirms the feasibility of implementing them in the future.

References

[1] I. Blake, G. Seroussi, and N. Smart. *Elliptic Curves in Cryptography*. Cambridge University Press, London Math. Soc., LNS 265, 2000.

[2] Certicom. The elliptic curve cryptosystem for smart cards. *http://www.comms.scitech.susx.ac.uk/fft/crypto/ECC_SC.pdf, Certicom White Paper*, The seventh in a series of ECC white papers, 1998.

[3] M. W. David and K. Sakurai. Security issues for contactless smart cards. *Public Key Cryptography: First International Workshop on Practice and Theory in Public Key Cryptography, PKC'98, Pacifico Yokohama, Japan*, LNCS 1431, 1998.

[4] M. Fouquet. Anneau d'endomorphismes et cardinalité des courbes elliptiques: aspects algorithmiques. *PhD Thesis*, 2001.

[5] M. Fouquet and F. Morain. Isogeny volcanoes and the SEA algorithm. *Springer 2002*, Proc. ANTS-V, LNCS 2369:276–291, 2002.

[6] F. Guirado, A. Ripoll, C. Roig, and E. Luque. Performance prediction using an application oriented mapping tool. *Proc. Euromicro Conf. on Parallel, Distributed and Network-based Processing*, pages 184–191, 2004.

[7] J. K. Liu, V. K. Wei, C. Siu, R. L. Chan, and T. Choi. Multi-application smart card with elliptic curve cryptosystem certificate. *EUROCON'2001, International Conference on Trends in Communications*, 2:381–384.

[8] Message Passing Interface Forum. MPI: A message-passing interface standard. *Journal of Supercomputer Applications*, 8(3/4), 1994.

[9] J. Miret, R. Moreno, D. Sadornil, J. Tena, and M. Valls. An algorithm to compute volcanoes of 2-isogenies of elliptic curves over finite fields. *Applied Mathematics and Computation*, (in press), 2005.

[10] J. Subhlok and G. Vondram. Optimal use of mixed task and data parallelism for pipelined computations. *Journal of Parallel and Distributed Computing*, 60:297–319, 2000.

[11] N. T. Trask and M. V. Meyerstein. Smart cards in electronic commerce. *BT Technology Journal, Springer Science*, 17(3):57–66, 1999.