

# Speeding Up Evaluation of Powers and Monomials

(Extended Abstract)

Hatem M. Bahig and Hazem M. Bahig

Computer Science Division, Department of Mathematics,  
Faculty of Science, Ain Shams University, Cairo, Egypt.  
hmbahig@asunet.shams.edu.eg hbahig@asunet.shams.edu.eg

**Abstract**— An addition sequence problem is given a set of numbers  $X = \{n_1, n_2, \dots, n_m\}$ , what is the minimal number of additions needed to compute all  $m$  numbers starting from 1? Downey et al. [9] showed that the addition sequence problem is NP-complete. This problem has application in evaluating the monomials  $y^{n_1}, y^{n_2}, \dots, y^{n_m}$ . In this paper, we present an algorithm to generate an addition sequence with minimal number of elements. We generalize some results on addition chain ( $m = 1$ ) to addition sequence to speed up the computation.

**keywords:** addition chain, addition sequence, vectorial addition chain, monomials evaluation, branch and bound algorithm.

## I. INTRODUCTION

An *addition chain* [11][13] for a natural number  $n$  is a strictly increasing sequence of natural numbers  $1 = a_0, a_1, \dots, a_r = n$  such that for each  $0 < i \leq r$ ,  $a_i = a_j + a_k$  with some  $0 \leq k \leq j < i$ .

Clear that, to ensure that an addition chain for  $n$  does not contain superfluous elements,  $a_j$  ( $0 \leq j < r$ ) should be used to construct some  $a_i$  with  $j < i \leq r$ .

The integer  $r$  is called the *length* of the addition chain for  $n$ . The shortest length of an addition chain for  $n$  is denoted by  $\ell(n)$ .

Let  $\lambda(n) = \lfloor \log_2 n \rfloor$  and  $\nu(n)$  be the number of 1's in the binary representation of  $n$ . The  $i^{\text{th}}$  step  $a_i = a_j + a_k$  ( $0 \leq k \leq j < i$ ) is called *doubling* if  $j = k = i - 1$ ; *star* if  $j = i - 1$ ; *small* if  $\lambda(a_i) = \lambda(a_{i-1})$ ; and *big* if  $\lambda(a_i) = \lambda(a_{i-1}) + 1$ .

The length of an addition chain can be expressed as

$$r = \lambda(n) + S(a_0, a_1, \dots, a_r = n),$$

where  $S(a_0, a_1, \dots, a_i)$  denotes the number of small steps in the chain up to  $a_i$ .

The problem of computing  $y^n$  with a minimal number of multiplications is equivalent to find a shortest addition chain for  $n$ . Computing  $y^n$  is widely used in cryptography, such as [17].

Addition chain is also defined for nonempty finite set of numbers. Without loss of generality, we assume that  $X = \{n_1, n_2, \dots, n_m\}$  be a set of  $m$ -numbers such that  $2 \leq n_i < n_{i+1}$ , for  $1 \leq i < m$ . An addition chain for the set  $X$  (in this case it is called an *addition sequence* for  $X$ ) is an addition chain for the maximum element of  $X$  containing every element of  $X$ . We will denote the length of addition sequence of  $X$  by  $\ell(X)$ . Note that every element,  $a_i \neq n_j$  ( $1 \leq j \leq m$ ), in the addition sequence of  $X$  should be used to construct some  $a_k$ ,  $i < k \leq r$ .

The problem of generating a shortest addition sequence for  $X = \{n_1, n_2, \dots, n_m\}$  is equivalent to compute simultaneously  $m$  power monomials  $y^{n_1}, y^{n_2}, \dots, y^{n_m}$  with minimum number of multiplications [13]. This problem is NP-complete [9].

In case of we have different  $y$ 's:  $(y_1, y_2, \dots, y_m)$ , the monomial can be evaluated by constructing a sequence of monomials where each monomial is a product of two previous ones. Since the exponent of a monomial can be represented by a  $m$ -dimension vector, therefore the exponents of the monomial sequence can be considered as sequence of vectors, where each vector is sum of two previous vectors. This sequence of vectors is called vector addition chain [13][16]. Olivos [16] proved that the problem of generating a shortest addition sequence for  $\{n_1, n_2, \dots, n_m\}$  is equivalent to generating a shortest vector addition chain for the vector  $(n_1, n_2, \dots, n_m)$ .

Speeding up generation of a shortest addition sequence is important in cryptography and number theory. For examples, the efficiency of some cryptosystems and protocols [10][14][15] is based on developing a fast algorithm to compute a monomial with large integers.

Many papers studied generating (short or shortest) addition chains (see [12] and [20] for examples), while a few papers studied generating addition sequences. Most of the papers which studied generating addition sequences concentrate on generating a short addition sequence, see for examples [3], [4], [8] and [21]. For generating a shortest addition sequence, Chen *et al.* [6] generated a shortest addition sequence for some special sets. Bleichenbacher and Flammenkamp [5] proposed a subroutine to search for a shortest addition sequence for  $X$  with length  $k$  provided that we have  $\ell(\alpha)$  for all numbers  $\alpha$  less than the maximum of  $X$  and  $\ell(\alpha) < k$ .

In this paper, we develop an algorithm to generate a shortest addition sequence by

- 1) extending Thurber's method [20] to shortest addition sequence, taking into consideration the differences between addition chains and addition sequences.
- 2) generalizing some results on addition chains [1] to addition sequence. We use the new results to speed up the generation.
- 3) using a subroutine from [5] to compute a lower bound of  $X$ .

The paper is organized as follows. Section II includes some prior results. In Section III, we present a depth-first search algorithm to generate a shortest addition sequence. The algorithm is a generalization of Thurber algorithm [7][20]. In Section IV, we generalize some results on a shortest addition chain to improve the algorithm. Section V summarize the differences between shortest addition chains and addition sequences. The implementation of the proposed algorithm and the ordinary one is presented in Section VI. Finally, Section VII includes the conclusion.

## II. PRELIMINARIES

In this section, we mention some prior results.

*Theorem 1:* [18]

$$\ell(n) \geq \lceil \log_2 n + \log_2(\nu(n)) - 2.13 \rceil.$$

In case of  $\nu(n) \leq 16$ , Thurber [19] proved that

$$\ell(n) \geq \lambda(n) + \lceil \log_2 \nu(n) \rceil.$$

*Lemma 1:* [6]

$$\ell(\{n_1, n_2, \dots, n_m\}) \geq \max\{\ell(n_1), \dots, \ell(n_m)\} + m - 1.$$

*Theorem 2:* [20] Let  $n$  be a natural number. Then the partial chain  $a_0, a_1, \dots, a_i$  cannot lead to a chain of length  $lb$  for  $n$ , if  $a_i < b_i$ , where the bounding sequence  $\{b_i\}_{i=1}^{lb}$  is defined as follows:

$$b_i = \lceil n/2^{lb-i} \rceil, \quad i = 0, \dots, lb. \quad (1)$$

*Theorem 3:* [20] Let  $n$  be a natural number. Then the partial chain  $a_0, a_1, \dots, a_i$  cannot lead to a chain of length  $lb$  for  $n$ , if  $a_i + a_{i-1} < b_{i+1}$ , where  $b_i$  is defined in Eq(1).

## III. GENERATING A SHORTEST ADDITION SEQUENCE

In this section, we present a depth-first search algorithm to generate a shortest addition sequence. The algorithm is a generalization of generating an addition chain ( $m = 1$ ) which is given in [20][1].

### A. Outline of the algorithm

Let  $X = \{n_1, n_2, \dots, n_m\}$  be a set of  $m$ -numbers such that  $n_i < n_{i+1}$ , for  $1 \leq i < m$ . Our algorithm for generating a shortest addition sequence uses a stack to hold the possible children of  $a_i$  at each stage. The algorithm starts by finding a short addition sequence for  $X$  using good heuristic methods (see Section III-B). Then it computes the lower bound  $lb$  of  $\ell(\{n_1, \dots, n_m\})$  (see Section III-C). The algorithm tries to find an addition sequence for  $X$  with length  $lb$  by taking the path from 1 down to level  $lb$  for the appropriate occurrence of  $n_m$  including the occurrence of  $n_1, \dots, n_{m-1}$ . While no addition sequence is found with length,  $lb$ , less than the length of the generated short addition sequence  $ub$ , the following steps will be repeated:

- 1) Determining a lower bound,  $b_i$ , of each  $a_i$  in any addition sequence of  $X$  with length  $lb$ . The set  $\{b_i\}_{i=1}^{lb}$  is called the bounding sequence. Thurber [20] proposed three bounding sequences and two types of pruning bounds when  $X = \{n\}$ . If for some step  $i$ ,  $a_i$  can be pruned (formally  $a_i < b_i$  or  $a_{i+1} + a_i < b_{i+2}$ ), then the partial sequence  $a_0, a_1, \dots, a_i$  cannot lead to addition sequence for  $X$  with length  $lb$ . In other words, no need to go down in the search tree.
- 2) The stack holds initially the number 1 and its level 0. Then the algorithm is looking for a sequence  $a_0, a_1, \dots, a_{lb} = n_m$  containing  $n_1, \dots, n_{m-1}$ . At each stage, the possible children  $a_{i+1}$  of  $a_i$  (and their level  $i + 1$ ) are put

in the stack. The values of  $a_{i+1}$  are all sums  $a_{i+1} = a_{k_1} + a_{k_2}$  for  $k_1 \leq k_2 \leq i$ , such that:

- a)  $a_i < a_{i+1} \leq n_j$ ; and
- b)  $a_{i+1} \geq b_{i+1}$ , and  $a_{i+1} + a_i \geq b_{i+2}$ , where  $\{b_i\}_{i=1}^{lb}$  is the bounding sequence.

- 3) If there is no children of  $a_i$ , then the partial sequence  $a_0, a_1, \dots, a_i$  cannot lead to an addition sequence for  $X$  with length  $lb$ . Therefore, the stack is popped. Since there is a possibility that the element at the top of stack is not brother of  $a_i$ , i.e., not children of  $a_{i-1}$ , it follows that we should first set  $i$  to the level of the element at the top of the stack, and then set  $a_i$  to the top of the stack.
- 4) If  $a_i$  has a children, then the stack is popped and the children  $a_{i+1}$  is added to the partial sequence  $a_0, a_1, \dots, a_i$ . In case of  $i = lb - 1$ ,  $a_i$  has either no children or  $n_m$ . If  $a_{lb} = n_m$ , then the algorithm finds a shortest addition sequence if the path contains  $n_1, \dots, n_{m-1}$ . If the path does not contain all  $n_j$ , then the stack is popped.
- 5) If no addition sequence of length  $lb$  is found, then the algorithm sets  $lb$  to  $lb + 1$ , and repeats the process. In case of  $lb = ub - 1$  and no addition sequence is found, then the algorithm sets the shortest addition sequence to the generated short sequence with length  $ub$ .

The algorithm is as follows:

**Algorithm: GSAS**

**Input:**  $X = \{n_1, n_2, \dots, n_m\}$ , where  $2 \leq n_i < n_{i+1}$  for  $1 \leq i < m$ .

**Output:** a shortest addition sequence for  $X$ .

**Begin**

find a short addition chain for  $X$  (its length =  $ub$ ).  
compute the lower bounds  $lb$  of  $\ell(\{n_1, \dots, n_m\})$ .

**while**  $lb < ub$  **loop**

Determine the bounding sequence  $B = \{b_i\}_{i=1}^{lb}$ .

$i \leftarrow 0$ .

$a_i \leftarrow 1$ .

Push the number  $a_0 = 1$  at level 0 in the stack.

**repeat**

**if**  $i < lb$  **then**

Push in the stack the possible children  $a_{i+1}$  of  $a_i$  and their levels  $i + 1$ .

**if** there are children  $a_{i+1}$  **then**

$i \leftarrow$  the level of the element at top of the stack ( $= i + 1$ ).

$a_i \leftarrow$  the element on top of the stack.

**if**  $a_{lb} = n_m$  **then**

**if** the path contains all  $n_j$  **then**

**return**  $a_0, a_1, \dots, a_{lb} = n_m$ .

**end if**

**end if**

**else**

$i \leftarrow$  the level of the element at top of the stack.

$a_i \leftarrow$  the element on top of the stack.

**end if**

**else**

$i \leftarrow$  the level of the element at top of the stack.

$a_i \leftarrow$  the element on top of the stack.

**end if**

**until**  $i = 0$ .

$lb \leftarrow lb + 1$ .

**end while**

**return** the generated short AS.

**End.**

### B. Generating a short addition sequence

The advantage of generating a short addition sequence in our algorithm is that, if no addition sequence for  $X$  is found with length  $lb_m = ub - 1$ , then fast generating a short addition sequence (where its length is  $ub$ ) is less time consuming than finding a shortest addition sequence with length  $ub = \ell(X)$ . One of the good methods to generate a short addition sequence for a set of numbers is described in [3] using continued fractions.

### C. Lower bound of $\ell(\{n_1, \dots, n_m\})$

If we have a lower bound close to the shortest length of addition sequence, then the search tree can be pruned much and the running time will be significantly improved. Unfortunately, except for very special set of numbers [6], all lower bounds of  $\ell(\{n_1, n_2, \dots, n_m\})$  depends on  $\ell(n_i)$   $1 \leq i \leq m$ . We use a subroutine from [5] to compute the lower bound of  $\ell(\{n_1, \dots, n_m\})$ .

**Algorithm: lower bounds**

**Input:**  $X = \{n_1, n_2, \dots, n_m\}$  where  $n_i < n_{i+1}$  for  $1 \leq i < m$ .

**Output:** lower bounds  $lb$  of  $\ell(\{n_1, \dots, n_m\})$ .

**Begin**

$lb_1 \leftarrow$  lower bound of  $n_1$  using Theorem 1;  
(or compute  $\ell(n_1)$  using [20][1]).

**for**  $i$  **from** 2 **to**  $m$  **loop**

temp1  $\leftarrow$  lower bound of  $n_i$  (Theorem 1);  
(or compute  $\ell(n_i)$  using [20][1]).

temp2  $\leftarrow lb_{i-1} + \lceil \log_2(n_i/n_{i-1}) \rceil + 1$ .

**if** temp1  $\geq$  temp2 **then**

$lb_i \leftarrow$  temp1.

**else**

$lb_i \leftarrow$  temp2.

**end if**

**end for**

**return**  $lb$ .

**End.**

#### D. Bounding Sequence

In this section we propose how to define the bounding sequence for the set  $X = \{n_1, n_2, \dots, n_m\}$ .

Thurber [20] proposed three bounding sequences and two types of pruning bounds when  $X = \{n\}$ . Two of the proposed bounding sequences may not applicable (directly) in addition sequence, because their proofs depend on the fact “the last step in a shortest addition chain must be star”. This fact is not true in a shortest addition sequence (see Section V). Therefore, we restrict our study to one bounding sequence and two types of pruning bounds as in Theorems 2 and 3.

We define the bounding sequence of  $X = \{n_1, n_2, \dots, n_m\}$  to be the bounding sequence of  $n_m$  with length  $lb$ , where  $lb$  is the lower bound computed by Section III-C. It is not difficult, from Eq(1) and the construction of the lower bound  $lb$  to show that the bounding sequence  $\{b_i\}_{i=1}^{lb}$  will be a bounding sequence for the partial sequence  $a_0, a_1, \dots, n_i$ . Therefore, we can cut some branches cannot lead to a shortest addition sequence for  $X$  if one of (or both) the conditions in Theorems 2 and 3 is satisfied.

#### IV. PROPERTIES OF STAR AND NONSTAR IN A SHORTEST ADDITION SEQUENCE

Generating all children  $a_{i+1}$  of  $a_i$  requires  $(i+1)(i+2)/2$  steps. It follows that if we have sufficient conditions for a step to be star, then no need to search for nonstar steps. Therefore, the sufficient conditions for stars omit  $i(i+1)/2$  steps to generate  $a_{i+1}$ . Also, if we have a lower bound for  $k$  or  $j$  in a nonstar step  $a_i = a_j + a_k$ , ( $k \leq j$ ), then the generation of nonstar steps can be restricted to the lower bound. We define the lower bounds of  $j$  and  $k$  to be the lower bound of nonstar. In this section we present three sufficient conditions for star steps and a lower bound of nonstar steps. These results are generalization of similar results in addition chains [1][2].

*Theorem 4:* Let  $a_0, a_1, \dots, a_i, \dots, a_r = n_m$  be an addition sequence for  $\{n_1, n_2, \dots, n_m\}$ . If one of the following conditions is true, then  $a_{i+1}$  is star.

- 1)  $a_i$  is double.
- 2)  $a_{i-1}$  is star and  $a_i = a_{i-1} + a_{i-2}$ .
- 3)  $S(a_0, a_1, \dots, a_{i-2}) = 0$ ,  $a_{i-1} = a_{i-2} + a_k$ ,  $k \leq i-3$  and  $a_i = 2a_{i-2}$ .

*Proof:*

- 1) From the definition.
- 2) Note that  $a_{i+1}$  cannot be  $a_{i-1} + a_p$ ;  $p \leq i-2$  or  $a_p + a_q$ ;  $p, q \leq i-2$ . Thus, the possibilities

for  $a_{i+1}$  are drawn from the set

$$\{a_i + a_j, j \leq i\} \cup \{2a_{i-1}\}.$$

Let  $a_{i-1} = a_{i-2} + a_k$ ,  $k \leq i-2$ . If  $a_{i+1} = 2a_{i-1}$ , then  $a_{i+1}$  is star since  $a_{i+1} = a_{i-1} + a_{i-1} = a_{i-1} + a_{i-2} + a_k = a_i + a_k$ ,  $k \leq i-2$ .

- 3) Note that  $a_{i+1}$  cannot be  $a_{i-1} + a_p$ ;  $p \leq i-3$  or  $a_p + a_q$ ;  $p, q \leq i-2$ . Thus, the possibilities for  $a_{i+1}$  are drawn from the set

$$\{a_i + a_j, j \leq i\} \cup \{2a_{i-1}, a_{i-1} + a_{i-2}\}.$$

If  $a_{i+1} = 2a_{i-1}$ , then  $a_{i+1}$  is star since  $2a_{i-1} = 2a_{i-2} + 2a_k = a_i + a_{k+1}$ , where  $a_{k+1} = 2a_k$  belongs to the partial sequence  $a_0, a_1, \dots, a_{i-2}$  since  $S(a_0, a_1, \dots, a_{i-2}) = 0$  and  $k \leq i-3$ .

If  $a_{i+1} = a_{i-1} + a_{i-2}$ , then  $a_{i+1}$  is star since  $a_{i-1} + a_{i-2} = a_i + a_k$ . This proves that  $a_{i+1}$  is star. ■

Now, we compute the lower bounds of nonstar steps.

*Theorem 5:* Let  $1 = a_0, a_1, \dots, a_r = n_m$  be a shortest addition sequence for  $\{n_1, n_2, \dots, n_m\}$  with length

$$r = \lambda(n_m) + S(a_0, a_1, \dots, a_r = n_m).$$

The lower bounds of a nonstar step  $a_i = a_j + a_k$ , ( $k \leq j$ ) are

$$k \geq 2 \text{ and } j \geq i - S(a_0, a_1, \dots, a_r = n_m) - 1.$$

*Proof:* Suppose that  $k \leq j$ . Since  $a_i - a_{i-1} \geq 3$  for  $a_i$  is to be nonstar; and there is no nonstar step at  $i = 1, 2$  and 3, for every number (see [20, Fig 3.1]); it follows that  $2 \leq k \leq j$ , and  $2 \leq j \leq i-2$ .

Now, we prove that  $j \geq i - S(a_0, a_1, \dots, a_r = n_m) - 1$ . Let  $a_i = a_j + a_k$ ,  $k \leq j \leq i - (S(a_0, a_1, \dots, n_m) + 2)$  be a nonstar step in a shortest addition sequence for  $\{n_1, n_2, \dots, n_m\}$  with length  $r = \lambda(n_m) + S(a_0, a_1, \dots, n_m)$ . Then  $S(a_0, a_1, \dots, a_r = n_m) \geq S(a_0, a_1, \dots, a_i) \geq S(a_0, a_1, \dots, a_j) + S(a_0, a_1, \dots, a_r = n_m) + 1$  which is a contradiction. ■

#### V. DIFFERENCES BETWEEN ADDITION CHAINS AND ADDITION SEQUENCES

In this section we summarize the differences between addition chains and addition sequences.

- 1) The last step in shortest addition chains is star, while it is not necessary to be star in shortest addition sequences.
- 2) In addition chains, each element  $a_i \neq a_r = n$  should be used to construct some  $a_k$  ( $i < k \leq$

$r$ ), while in addition sequences  $a_i \neq n_j$  ( $1 \leq j \leq m$ ).

- 3) The complexity of computing a shortest addition chain for  $n$  remains open [13], while computing a shortest addition sequence is NP-complete [9].
- 4) In addition chains, the lower bounds of nonstar steps  $a_i = a_j + a_k$  are

$$k \geq 2 \text{ and } j \geq i - \ell(n) - \lambda(n),$$

while in addition sequences

$$k \geq 2 \text{ and } j \geq i - \ell(\{n_1, \dots, n_m\}) - \lambda(n_m) - 1.$$

## VI. IMPLEMENTATION

In this section, we present results of timing experiments we conducted to determine how well the proposed algorithm perform in practice. We implemented **GSAS** along the ordinary depth-first algorithm (without bounding sequence, sufficient conditions for stars and lower bounds for nonstar) in the C language. We used gcc compiler with standard level of optimization. Our platform was Pentium IV 2.4 GHz with Linux operating system.

Each algorithm was timed for (number of elements in  $X$ )  $m = 2, 4, 6, 8$ . For each  $m$  we used a common set of 50 pseudo-random numbers from the interval  $[2, 2^{15}]$ . The data reported in Table I are averages.

TABLE I  
AVERAGE RUNNING TIME IN CPU SECONDS

Algorithm	m			
	2	4	6	8
Ordinary	44.03	5425.24	>> 8000	>> 10000
Proposed	3.28	739.59	2715.37	6087.18

## VII. CONCLUSION AND FUTURE WORKS

We have presented an algorithm to generate a shortest addition sequence. Then we have improved it by generalizing some results on addition chain to addition sequence.

One can improve the generation by

- 1) proposing a better bounding sequence.
- 2) making a check point at the level  $lb_j$ , where  $lb_j$  is the lower bound of  $\ell(n_1, \dots, n_j)$ , ( $1 \leq j \leq m$ ).

## REFERENCES

- [1] H. Bahig. Improved generation of minimal addition chains. *submitted*.
- [2] H. Bahig and K. Nakamura. Some properties of nonstar steps in addition chains and new cases where the Scholz conjecture is true, *J. Algorithms* **42**, pp. 304–316, 2002.
- [3] F. Bergeron, J. Berstel and S. Brlek, Efficient computation of addition chains, *J. de Theorie Nombres de Bordeaux* **6**, pp. 21–38, 1994.
- [4] F. Bergeron, J. Berstel, S. Brlek and C. Duboc, Addition chains using continued fractions, *J. Algorithms* **10**, pp. 403–412, 1989.
- [5] D. Bleichenbacher and A. Flammenkamp, An efficient algorithm for computing shortest addition chains. *submitted*
- [6] Y. Chen, C. Chang and W. Yang, Some properties of vectorial addition chains, *Intern. J. Computer Math.* **54**, pp. 185–196, 1994.
- [7] Y. H. Chin and Y. H. Tsai. Algorithms for Finding the Shortest Addition Chain. Proceedings of National Computer Symposium, Kaoshiung, Taiwan, 1398–1414 Dec. 20-22 (1985).
- [8] P. de Rooij. Efficient exponentiation using precomputation and vector addition chains. Advances in cryptology—EUROCRYPT '94 (Perugia), Lecture Notes in Computer Science **950**, 389–399, 1995.
- [9] P. Downey, B. Leong and R. Sethi, Computing sequences with addition chains, *SIAM J. Computing* Vol.10, No.3, pp. 638–646, 1981.
- [10] T. ElGamal, A public key cryptosystem and a signature scheme based on discrete logarithms. *IEEE Transaction on Information Theory* IT-31. No. 4, pp. 469–472, 1985.
- [11] A. Flammenkamp, Integers with a small number of minimal addition chains. *Discrete Math.* **205**, (1999) 221–227.
- [12] D. M. Gordon. A survey of fast exponentiation methods. *J. Algorithms* **122**, pp. 129–146, 1998.
- [13] D. E. Knuth, *The Art of Computer Programming: Seminumerical Algorithms*. Vol.2, 3rd ed., Addison-Wesley, Reading MA, pp. 461–485, 1997.
- [14] C. Laih, and S. Yen. Secure addition sequence and its applications on the server-aided secret computation protocols. in *Advances in Cryptology-AUSCRYPT'92*, Lecture Notes in Computer Science **718**, pp. 219–229, 1992
- [15] C. Laih, S. Yen, and L. Harn. Two efficient server-aided secret computation protocols based on the addition sequence. in *Advances in Cryptology-ASIACRYPT'91*, pp. 450–459, 1991.
- [16] J. Olives. On vectorial addition chains. *J. Algorithms* **2**, pp. 13–21, 1981.
- [17] R. Rivest, A. Shamir and L. Adleman. A method for obtaining digital signatures and public-key cryptosystems. *Communications of the ACM*, **21**, No.2, 120–126 (1978).
- [18] A. Schönhage, A lower bound for the length of addition chains, *Theoretical Computer Science* **1**, pp. 1-12, 1975.
- [19] E.G. Thurber. The Scholz-Brauer problem on addition chains. *Pacific J. Math.* **49**, 229–242 (1973).
- [20] E.G. Thurber, Efficient generation of minimal length addition chains. *SIAM J. Computing* **28**, pp. 1247–1263, 1999.
- [21] S. Yen and C. Laih, The fast cascade exponentiation algorithm and its applications on cryptography. Lecture Notes in Computer Science **718**, Proceeding of Auscrypt'92, Australia, pp. 447–456, 1992.