SIMD상에서의 이차선별법을 사용한 병렬 소인수분해 알고리즘

김 양 희†

요 약

본 논문에서는 첫째로 큰 정수의 소인수 분해를 위한 병렬 이차선별법(parallel quadratic sieve) 알고리즘을 제시한다. 이 알고리즘을 반복적으로 사용하여, 분산 메모리 모델(DMM)을 갖는 SIMD구조의 병렬 컴퓨터 상에서 분할정복기법을 사용하는 병렬 소인수 분해(parallel factoring) 알고리즘을 제시한다. 또한 이러한 알고리즘이 시간과 프로세서의 곱의 관점에서 최적화 알고리즘임을 보인다.

Parallel Factorization using Quadratic Sieve Algorithm on SIMD machines

Yang-Hee Kim†

ABSTRACT

In this paper, we first design a parallel quadratic sieve algorithm for factoring method. We then present parallel factoring algorithm for factoring a large odd integer by repeatedly using the parallel quadratic sieve algorithm based on the divide-and-conquer strategy on SIMD machines with DMM. We show that this algorithm is optimal in view of the product of time and processor numbers.

키워드 : 이차선별 소인수분해 알고리즘(Quadratic Sieve factoring algorithm), SIMD 컴퓨터(SIMD computer), 병렬 알고리즘(parallel algorithm)

1. Introduction

SIMD(Single Instruction stream, Multiple Data stream) computer consists of several processing elements(PEs) which are synchronized and operate under the control of a single instruction stream by a control unit. From the view point of memory distribution, SIMD can be classified into two classes : distributed memory model(DMM) and shared memory model(SMM). The processing elements(PEs) of DMM are connected through interconnection network(IN) and those of SMM are connected through common memory via bus or multistage interconnection network. In this paper we address a parallel factoring problem on SIMD machines with DMM.(See Figure 1.)

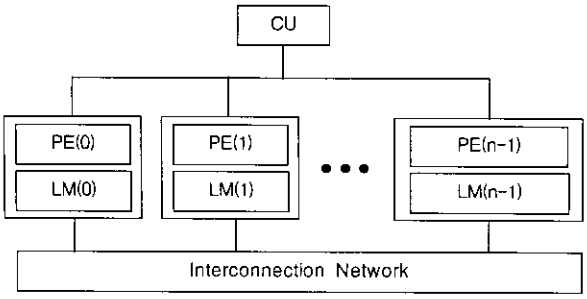
In the last few years, the intractability of factoring problem has been used in several new public-key cryptosystems, signature schemes and discrete logarithm problems for their security, including the RSA(Rivest-Shamir-Adleman) public-key cryptosystems. Since the time complexity of known factoring algorithm is exponential, the parallel processing is adequate [2].

Generally speaking, there are two types of factoring algorithms, the general factoring algorithm which solely depends on the size of the number N to be factored and the special factoring algorithm which depends on properties of the unknown factor of N. In this paper, we first design an optimal parallel algorithm for factoring a number into two factors using the quadratic sieve factoring method which is one of the general purpose factoring schemes. We also present an optimal parallel algorithm for factoring a large composite integer by repeatedly using the parallel quadratic sieve algorithm. This algorithm can be carried out efficiently using divide-and-conquer strategy on SIMD machines by partitioning the problem into two subproblems each of which can be mapped onto a subblock of equal size and run in parallel.

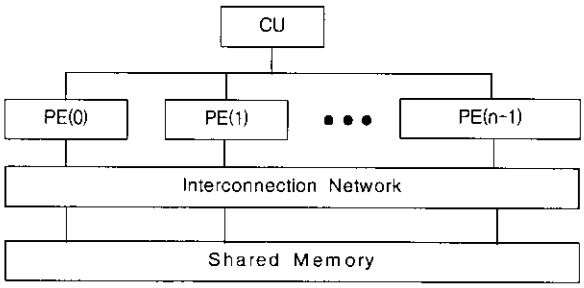
※ 본 논문은 2000년도 한세대학교 교내 연구비(자유공모) 지원에 의해 수행되었음.

† 정 회 원 : 한세대학교 컴퓨터정보통신공학부 교수

논문접수 : 2000년 10월 18일, 심사완료 : 2001년 3월 15일



(a) SIMD/DMM



(b) SIMD/SMM

CU : Control Unit PE : Processing Element
SMM : Shared Memory Model LM : Local Memory
DMM : Distributed Memory Model

(Figure 1) SIMD computer organization

2. The Quadratic Sieve Factoring Method

Suppose N is a composite number which we would like to factor. The quadratic sieve factoring method belongs to a family of factoring algorithms known as combination of congruences. So the basic goal in factoring N is to find a solution to the congruence

$$X^2 \equiv Y^2 \pmod{N}. \tag{1}$$

If such a pair (X, Y) is found, then with probability 1/2, the greatest common divisor $(X-Y, N)$ becomes a nontrivial factor of N. To solve equation (1), we use the polynomial

$$f(x)=(x+\lfloor\sqrt{N}\rfloor)^2-N. \tag{2}$$

Since $f(x)$ is a quadratic polynomial and if x is an integer, then

$$(x+\lfloor\sqrt{N}\rfloor)^2 \equiv f(x) \pmod{N}. \tag{3}$$

Suppose we could find a set of distinct integers $x_1, \cdots, x_k$ such that $f(x_1)\cdots f(x_k)$ is a square, i.e.

$$f(x_1)\cdots f(x_k)=Y^2.$$

Then let $X=(x_1+\lfloor\sqrt{N}\rfloor)\cdots(x_k+\lfloor\sqrt{N}\rfloor)$. From (3), we have $X^2 \equiv Y^2 \pmod{N}$. This pair (X, Y) is a solution of (1). To be specific, we try to factor $f(x_i)$ using the first B primes $p_1, \cdots, p_B$ with Legendre symbol $\left(\frac{N}{p_j}\right)=1$, $j=1, \cdots, B$, that is $f(x_i) = \Pi_{j=1}^{B} p_j^{a_{ji}}$, where the exponents a_{ji} are nonnegative integers. The set of small B primes is called the "factor base" and B is a parameter to be chosen later. Let $\vec{a_i}=(a_{1i}, a_{2i}, \cdots, a_{Bi})$, then finding set of indices I where $\Pi_{i\in I} f(x_i)$ is a square is equivalent to finding I where $\sum_{i\in I} \vec{a_i} \equiv \vec{0} \pmod{2}$. Thus finding I is reduced to a problem of finding a nontrivial linear dependency among the vector $\vec{a_i}$ over the finite field with two elements.

The heuristic running time for quadratic sieve factoring method [11] is $L(N)=\exp((1+o(1))\sqrt{\log N \log\log N})$. And heuristics suggest we choose $B=\sqrt{L(N)}$ and $2M$ is between B and B^2, where $[-M, M]$ is the sieve interval [13]. The quadratic sieve algorithm is given below.

2.1 Algorithm Quadratic Sieve

2.1.1 (Precomputation stage)

(1.1) Select a factor base $FB=\{p_i \mid \left(\frac{N}{p_i}\right)=1,\ p_i$ is prime, $i=1, \cdots, B\}$ for the given value B and $p_0=-1$ for the sign.

(1.2) Solve the quadratic congruence $f(x) \equiv 0 \pmod{p_i}$ for all $p_i \in FB$. Say its two roots a_{1p_i} and a_{2p_i} for all $p_i \in FB$.

2.1.2 (Sieving stage)

(2.1) Initialize a sieve array S to $\lfloor \log_2 f(x) \rfloor$ over the sieve interval $[-M, M]$ for given value M.

(2.2) For all odd prime $p_i \in FB$ subtract the value $\lfloor \log p_i \rfloor$ from the sieve array at locations x where $x \equiv a_{1p_i}$ or $a_{2p_i} \pmod{p_i}$.

(2.3) Find the location x_i's of sieve array whose values are closed to 0. For these locations, construct the exact factorization via division, that is, $f(x_i) = \Pi_{j=1}^{B} p_j^{a_{ji}}$, $p_j \in FB$.

2.1.3 (GEM stage)

Use Gaussian Elimination over $GF(2)$ to find a set of $f(x_1), \cdots, f(x_k)$ whose product is a square, that is, $f(x_1)\cdots f(x_k)=Y^2$ for some Y.

2.2 End Algorithm

Lemma 1 *Given a positive large odd integer N with $N \geq 2^{2^6}$, the sequential factoring algorithm*

using divide-and-conquer strategy can be computed in $O(L(N))$.

Proof : Let $T(N)$ be the overall time complexity of the sequential factoring algorithm. Since the time complexity of the sequential algorithm Quadratic Sieve is $L(N)$, we have the following recurrence relation :

$$T(N) = \begin{cases} b & \text{if } N=2 \\ 2T(\sqrt{N}) + O(L(N)) & \text{if } N>2, \end{cases}$$

where b is a constant. If we remove the recursion from the above relation,

$$T(N)=2^kT(N^{\frac{1}{2^k}})+2^{k-1}L(N^{\frac{1}{2^{k-1}}})+2^{k-2}L(N^{\frac{1}{2^{k-2}}})+\cdots+L(N)$$

Let $N=2^{2^k}$. Then we have

$$\sum_{j=1}^{k}2^{k-j}L(N^{\frac{1}{2^{k-j}}})=\sum_{j=1}^{k}2^{k-j}L(2^{2^j})=\sum_{j=1}^{k}2^{-j}(\,e^{\sqrt{j2^j(\ln 2)^2}}\cdot 2^k)$$

$$=\sum_{j=1}^{k}2^{-j}(\,e^{\sqrt{j2^j(\ln 2)^2}+k\ln 2}).$$

Since $\sqrt{j2^j(\ln 2)^2}+k\ln 2 \le \sqrt{k2^k(\ln 2)^2}\left(2^{\frac{j-k}{2}}\sqrt{\frac{j}{k}}+\sqrt{\frac{k}{2^k}}\right)$ and $\left(2^{\frac{j-k}{2}}\sqrt{\frac{j}{k}}+\sqrt{\frac{k}{2^k}}\right)\le 1$ for $j\le k$ and $k\ge 6$, we obtain

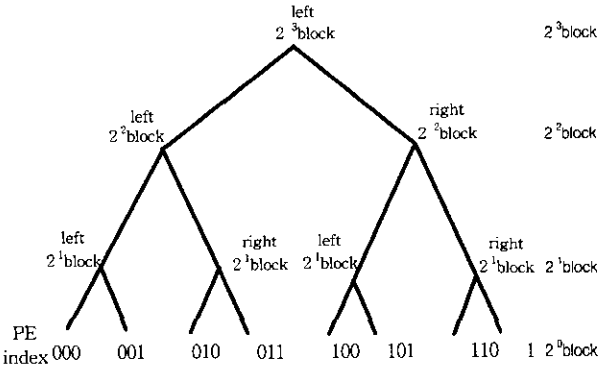
$$\sum_{j=1}^{k}2^{-j}\,e^{\sqrt{j2^j(\ln 2)^2}+k\ln 2} \le \sum_{j=1}^{k}2^{-j}L(N) = O(L(N)).$$

Therefore, we have $T(N)=O(L(N))$.

The main difficulty with the quadratic sieve factoring method is that we must obtain as many fully factored quadratic residues as there are primes in the factor base. So parallel processing is adequate in finding these quadratic residues quickly.

3. Model of parallel computation

We assume that our model of parallel computation consists of n *PEs* from $PE[1]$ to $PE[n]$ each of which has a local memory. We shall use a divide-and-conquer strategy to factorize a large odd integer N. Before describing the strategy, let us introduce the concept of blocks on SIMD machines. If we represent each PE with index i by its binary representation $i_{q-1}\,i_{q-2}\cdots i_1\,i_0$ where $1\le i\le n$ and $n=2^q$, a 2^k-block of *PE*s consists of 2^k consecutive *PE*s whose indices differ only in the least significant k bits. A 2^k-block is called a left 2^k-block if it contains only *PE*s with indices i whose $i_k=0$ and a right 2^k-block if it contains only *PE*s with indices i whose $i_k=1$. A 2^{k+1}-block consists of two sibling blocks, a left 2^k-block and a right 2^k-block. Thus each PE is a 2^0-block, and the whole n *PE*s itself is a left 2^q-block, where $2^q=n$.(See Figure 2.)



(Figure 2) A block with 8 PEs

The divide-and-conquer can be briefly described as follows : First the given problem is divided into two subproblems, stored in two left and right sibling blocks respectively. We then recursively solve in parallel the subproblems in each block, and then combine two subsolutions obtained in the left and right blocks, producing the final result. The overall time complexity is given by the recurrence relation $T(n)=T(n/2)+f(n)$, where $f(n)$ is the time taken for the merging of two subsolutions in the left and right blocks and n is the number of *PE*s.

4. Parallel Algorithms

In this section, we design two parallel algorithms. We first design a parallel algorithm for solving the quadratic sieve factoring and then for factoring a large number based on divide-and-conquer strategy on SIMD machines.

4.1 Parallel quadratic sieve algorithm

We design the parallel quadratic sieve algorithm in this section. Each PE has one of $\frac{2M}{k}$ equal size subintervals of the sieve interval $[-M, M]$, where k is the number of *PE*s. The parallel quadratic sieve algorithm has three stages : (stage 1) precomputation, (stage 2) sieving, and (stage 3) GEM(Gaussian Elimination). The precomputation and sieving stage can be parallelized on any number of *PE*s by restricting each of them to a unique interval disjoint from

the intervals assigned to other PEs. In (stage 1), each of PEs selects a factor base FB and solves the quadratic congruence

$$f(x) \equiv 0 \pmod{p} \text{ for any primes } p \text{ in } FB$$

so that two roots of the equation lie in its subinterval of the sieve array. We denote the collection of all such primes by $P(x)$. In (stage 2), each of PEs initializes its subinterval S and then for all primes in $P(x)$, subtracts $[\log p]$ from the sieve array at location x in parallel. Finally, if the residual number at this location are close to 0, then each of PEs found residues. For parallel Gaussian Elimination for sparse matrix, we assign each of PEs to rows alternatively. For details, we refer to [6].

Parallel Algorithm Quadratic Sieve

Input. A positive large odd integer N. Each PE has one of $\frac{2M}{k}$ equal size subintervals of the sieve interval $[-M, M]$, where k is the number of PEs.

Output. Two integers a and b such that $N = a \cdot b$.

1. For $j=1$ *to* k do the following in parallel.
   (1) (Precomputation stage)
   (1.1) Select a factor base $FB = \{p_i \mid \left(\frac{N}{p_i}\right) = 1,\ p_i$ is prime$, i = 1, \cdots, B\}$ for the given value B and $p_0 = -1$ for the sign.
   (1.2) For each x in the subinterval in PE [j].
   For $i=1$ to B do
   Add prime p_i in FB to $P(x)$ if $f(x) \equiv 0 \pmod{p_i}$.
   (2) (Sieving stage)
   For each x in the subinterval in PE [j].
   (2.1) Initialize a sieve array S to $[\log_2 f(x)]$ for x in the subinterval in PE [j].
   (2.2) Subtract the value of $[\log p_i]$ from the sieve array at location x for all primes in $P(x)$.
   (2.3) Find the location j's of sieve array whose values are close to 0.
   For these locations, factorization of $f(x)$ is the product of all primes p_i's in $P(x)$.

2. (GEM stage)
   Apply Parallel Gaussian Elimination over $GF(2)$[6] to find a set of $f(x_1), \cdots, f(x_k)$ whose product is a square, that is, $f(x_1) \cdots f(x_k) = Y^2$ for some Y.

End Parallel Algorithm

Theorem 1 *Given a positive large odd integer N, the parallel algorithm Quadratic Sieve can be computed in $O\left(\frac{B^\alpha}{k}\right)$ with k PEs, where $2 \le \alpha \le 3$.*

Proof : Since the number of quadratic residues and that of nonresidues are the same, step(1.1) takes $O(B)$. Step(1.2) takes time $O\left(B \cdot \frac{M}{k}\right)$, step(2.1) and (2.3) take $O\left(\frac{M}{k}\right)$. Since the number of prime factors of $f(x)$ are much smaller than the number of factor base, step(2.2) takes $O\left(B^{o(1)} \cdot \frac{M}{k}\right)$ where $o(1) \to 0$ as $N \to \infty$. Finally step 2 take $O\left(\frac{B^\beta}{k}\right)$ for $2 \le \beta \le 3$. Since $2M$ is chosen to be between B and B^2, the overall time complexity becomes $O\left(\frac{B^\alpha}{k}\right)$.

If we use $O(B^2)$ PEs, the sieving stage can be computed in $O(B^{o(1)})$ and the parallel algorithm Quadratic Sieve takes time $O(B^\gamma)$ where $0 \le \gamma \le 1$. On the other hand, if we use $O(B)$ PEs, the parallel algorithm Quadratic Sieve can be computed in $O(B^s)$ where $1 \le s \le 2$. Since the sequential algorithm takes $O(B^\alpha)$ for $2 \le \alpha \le 3$, our algorithm is optimal in view of the product of time and processor numbers.

4.2 Parallel Factoring Algorithm

In this section, we present a parallel factoring algorithm based on divide-and-conquer strategy on SIMD machines. Let $[-M_h, M_h]$ be the sieve interval, B_h be the factor base, and let N_h be a composite number which we wish to factor in 2^h block. Then in each 2^h block, if $N_h > e^{16}$, then we factor N_h by N_{h-1}^L and N_{h-1}^R using the parallel algorithm Quadratic Sieve. Now transfer N_{h-1}^L and N_{h-1}^R to the left and right 2^{h-1} block respectively and update its sieve interval $[-M_{h-1}, M_{h-1}]$ and factor base B_{h-1}. In 2^0 block, each $PE[j]$ has either a prime number or a small composite number which we can be factored using trial division. The parallel algorithm Factorization is given below.

Parallel Algorithm Factorization

Input. A positive large odd integer N.

Output. A complete factorization of N. That is $N = p_1^{a_1} \cdots p_s^{a_s}$ where p_i is prime.

1. Set $N_q = N$, $M_q = M$ and $B_q = B$, where $[-M, M]$ is the sieve interval, B is the number of factor base and $k = 2^q$ with k is the number of PEs.
2. For $h = q$ down to 1 do the following for each 2^h-block in parallel.
   (2.1) Determine the sieve interval $[-M_h, M_h]$ and the number of factor base B_h heuristically.
   (2.2) If $N_h > e^{16}$ then
   factor $N_h = N_{h-1}^L \cdot N_{h-1}^R$ using the parallel algorithm Quadratic Sieve.
   Else
   factor N_h using trial division.
   (2.3) Transfer N_{h-1}^L and N_{h-1}^R to the left and right 2^{h-1}-block respectively.
3. Do the following for each 2^0-block in parallel.
   Apply trial division if the number in 2^0-block is not prime.

4.2.2 End Parallel Algorithm

Lemma 2 *Let N be a positive large integer and $N > e^{16}$. Then*

$$2\ln 4 + \sqrt{-\frac{1}{2}\ln 2 \ln N + \frac{1}{2}\ln N \ln\ln N} \le \sqrt{\ln N \ln\ln N}.$$

Proof : Let $t = \ln N$ and let $N > e^{16}$, then $t > e^4$ and $\ln t > (\ln 4)^2$. Therefore,

$$t > \frac{16(\ln 4)^2 \ln t}{(\ln t)^2} > \frac{16(\ln 4)^2 \ln t}{(\ln t + \ln 2)^2}. \quad (4)$$

So equation (4) can be written as $\frac{1}{2}t(\ln t + \ln 2) > 2\ln 4\sqrt{t \ln t}$. It then follows that $(\sqrt{t\ln t} - 2\ln 4)^2 > -\frac{1}{2}t\ln 2 + \frac{1}{2}t\ln t$.

Therefore, we have $2\ln 4 + \sqrt{-\frac{1}{2}\ln 2 \ln N + \frac{1}{2}\ln N \ln\ln N} \le \sqrt{\ln N \ln\ln N}$.

Lemma 3 *Let B_q and B_{q-1} be the number of factor base in 2^q and 2^{q-1} blocks respectively. Let k be the number of PEs in 2^q blocks with $k = B_q^\gamma$, where $1 \le \gamma \le 2$. Then for a positive large odd integer N, if $N > e^{16}$, then $\frac{k}{2} > B_{q-1}^\gamma$.*

Proof : Let $t = \ln N$ and let $N > e^{16}$. Then by Lemma 2, we have

$$4B_{q-1}^2 = 4e^{\sqrt{\ln\sqrt{N}\ \ln\ln\sqrt{N}}} = e^{\ln 4 + \sqrt{-\frac{1}{2}t\ln 2 + \frac{1}{2}t\ln t}} \le e^{\sqrt{t\ln t}} = B_q^2$$

So, for $1 \le \gamma \le 2$, $B_{q-1}^\gamma < \left(\frac{1}{2}B_q\right)^\gamma = \left(\frac{1}{2}\right)^\gamma \cdot B_q^\gamma < \frac{1}{2} \cdot B_q^\gamma$

As a consequence, we have $B_{q-1}^\gamma < \frac{1}{2} \cdot B_q^\gamma = \frac{k}{2}$.

By Lemma 3, the number of PEs in 2^h block is not less than B_{h-1}^γ during the execution of the parallel algorithm for $0 \le h \le q$ with $N > e^{16}$.

Theorem 2 *Let k be number of PEs with $k = B^\beta$ for given value $B = \sqrt{L(N)}$ where $1 \le \beta \le 2$ and $L(N) = \exp((1+o(1))\sqrt{\log N \ \log\log N}$. Then for a positive large odd integer N with $N > e^{16}$, the parallel factoring algorithm can be computed in $O(\log B)$ if $\frac{\alpha}{\beta} - 1 = 0$ and $O(B^{\alpha-\beta})$ if $0 < \frac{\alpha}{\beta} - 1 \le 2$. Furthermore, our algorithm is optimal.*

Proof : Let $T(k)$ be the overall time complexity of the parallel factoring algorithm. Then by Theorem 1 and Lemma 3, we have the following recurrence relation :

$$T(k) = \begin{cases} b & \text{if } k = 2 \\ T\left(\frac{k}{2}\right) + O\left(\frac{B^\alpha}{k}\right) & \text{if } k > 2, \end{cases}$$

where b is a constant. Since $k = B^\beta$ $O\left(\frac{B^\alpha}{k}\right) = O(B^{\alpha-\beta})$. Therefore we have

$$T(k) = \begin{cases} O(\log k) & \text{if } \frac{\alpha}{\beta} - 1 = 0 \\ O\left(k^{\frac{\alpha}{\beta}-1}\right) & \text{if } 0 < \frac{\alpha}{\beta} - 1 \le 2. \end{cases}$$

But $O(\log k) = O(\log B)$ and $O(k^{\frac{\alpha}{\beta}-1}) = O(B^{\alpha-\beta})$, we proved our theorem. Since if we use $O(B^\beta)$ PEs, time complexity of our algorithm is computed in $O(B^{\alpha-\beta})$ where $1 \le \alpha - \beta \le 2$, our algorithm is optimal.

If we use $O(B^\beta)$ PEs with $1 \le \beta \le 2$, the parallel factoring algorithm can be computed in $O(B^s)$ where $1 \le s \le 2$. Since the sequential factoring algorithm takes $O(B^\alpha)$ for $2 \le \alpha \le 3$, our algorithm is optimal in view of the product of time and processor numbers.

5. Conclusion

In this paper, we have presented a parallel factoring algorithm based on quadratic sieve method on SIMD machines with DMM. First we have described the parallel algorithm Quadratic Sieve and we have shown that it can be carried out in $O\left(\frac{B^{\alpha}}{k}\right)$ optimal time, where k is the number of PEs and $2 \leq \alpha \leq 3$. Our algorithm is optimal in the sense of the product of time and processor numbers. The parallel algorithm Quadratic sieve can factorize a given positive large odd integer into two subfactors of almost equal size and the number of processors in 2^h block is not less than B_{h-1}^{r}, we could partition the factoring problem into two subproblems each of which can be mapped onto a subblock of equal size respectively, which enable us to obtain an optimal parallel factoring algorithm using divide-and-conquer strategy on SIMD machines.

References

[1] T. R. Caron and R. D. Silverman, *Parallel implementation of the quadratic sieve*, in J. Super-computing, Vol.1, pp.273-290, 1988.

[2] A. Bosselaers, R. Govaerts and J. Vanderwalle, SHA : *A Design for Parallel Architectures?*, in Eurocrypt '97, pp. 348-362, 1997.

[3] D. Coppersmith, *Specialized integer factorization*, in Eurocrypt '98, pp.542-545, 1998.

[4] B. Dixon and A. K. Lenstra, *Massively parallel elliptic curve factoring*, in Eurocrypt '92, pp.183-193, 1992.

[5] B. Dixon and A. K. Lenstra, *Factoring integers using SIMD sieves, in Eurocrypt* '93, pp.28-39, 1993.

[6] C. S. Jeong and H. D. Kim, *Parallel Gaussian Elimination on SIMD Machines*, in Technical Report, Dept. of Electronics Eng., Korea University, 1993.

[7] N. Koblitz, A course in number theory and cryptography, Springer-verlag, 1987.

[8] P. Nguyen and J. Stern, *A Cryptanalysis of the Qu-Vanstone Cryptosystem Based on Group Factorizations*, in Crypto '97, pp.198-212, 1997.

[9] T. Okamoto and S. Uchiyama, *A New Public-Key Cryptosystem as Secure as Factoring*, in Eurocrypt '98, pp.308-318, 1998.

[10] Rene Peralta, *A Quadratic Sieve on the n-Dimensional Cube*, Advances in Cryptology-Crypto '92, pp.324-332, 1992.

[11] C. Pomerance, *Factoring, in proceedings of Symposia in Applied Mathematics*, Vol.42, pp.27-47, 1990.

[12] C. Pomerance, *A pipeline architecture for factoring large integers with the quadratic sieve algorithm*, in SIAM J. Computing 17, pp.387-403, 1988.

[13] C. Pomerance, *The quadratic sieve factoring algorithm*, in Eurocrypt '84, Paris 1984, Lecture Notes in Computer Science Vol.209, pp.169-182, 1985.

[14] R. D. Silverman, *The Multiple polynomial quadratic sieve*, in Math. Comp. Vol.48, pp.329-339, 1987.

김 양 희

e-mail : yanghk@hansei.ac.kr

1982년 이화여자대학교 수학과 졸업(이학사)

1984년 서울대학교 수학과 졸업(이학석사)

1988년 University of Wisconsin Madison 전자계산학과(이학석사)

1989년 University of Wisconsin Madison 전자계산학과(박사과정 이수)

1997년 고려대학교 전자공학과 컴퓨터 공학전공 졸업(공학박사)

1990년~1998년 수원과학대학 전자계산과 전임강사, 조교수

1998년~현재 한세대학교 컴퓨터정보통신공학부 전임강사, 조교수

관심분야 : 분산 및 병렬 알고리즘 및 시스템, 정보 보호, 공간 데이터베이스, 데이터베이스 보안 등