Acceleration of the Processing of Linear Equations in Characteristic 2 for RSA Decryption

Project Representative
Hidehiko Hasegawa Faculty of Library, Information and Media Science, University of Tsukuba

Authors
Yasunori Ushiro Department of Mathematics, School of Education, Waseda University
Hidehiko Hasegawa Faculty of Library, Information and Media Science, University of Tsukuba

The authors tuned the processing of linear equations, which is the second most time-consuming process in RSA decryption. As this process only tests whether the exponent part of equations are odd or even, binary numbers are used. The 64 binary numbers are stored as 64-bit integers and only the locations of non-zero elements of a sparse binary matrix are stored. The processing of linear equations in characteristic 2 requires fewer floating-point operations but an enormous amount of integer operation and list-vector processing.

The list-vector processing consumes almost of all computation time, but can be accelerated by loop unrolling. Hybrid parallelization with automatic parallelization by compiler at each node and MPI parallelization among nodes is effective for the Earth Simulator 2 (ES2). The processing of linear equations of dimension 8×10^7 in characteristic 2 required 130 hours on 4 nodes, and their parallel speedups were estimated as 3.5 for 4 nodes and 5.8 for 8 nodes. The computing time for linear equations of dimension 2×10^8 (equivalent to RSA-768) was estimated 800 hours on 4 nodes, whict is a performance equivalent to that of a PC cluster with 1,700 CPUs. The computing time for linear equations of dimension 8×10^8 (equivalent to RSA-898) was estimated as 5,000 hours on 16 nodes.

Keywords: RSA cryptosystem, factorization, Characteristic 2, Block Lanczos method, GNFS

1. Introduction

The RSA cryptosystem is the most important technology for using the Internet safely; however; currently used 1,024-bit RSA code will not be safe for use in the near future [5]. The RSA cryptosystem is based on the difficulty of the factorization of long digits of a composite number, and the decryption time a of 1,024-bit RSA code is several tens of years even if the fastest supercomputer is used. For a RSA code with certain number of bits to be considered safe, its decryption time using the fastest algorithm and on the fastest supercomputer must be more than 10 years.

The present world record of RSA decryption, for RSA-768 (768 bits, 232 digits) is 1,677 CPU-year [7]. This means that if only one core in CPU (AMD64 2.2GHz) is used, then decryption takes 1,677 years. All reported world records for RSA decryption were achieved by PC clusters; no report has yet been made regarding a vector supercomputer. Therefore, a test on a vector supercomputer is necessary for a precise evaluation/discussion of the safety of 1,024-bit RSA cryptography code.

The present project intends to obtain some basic information for carrying out RSA decryption on the Earth Simulator 2 (ES2), as a representative of vector supercomputer. The decryption processing consists of three parts: the first step is “sieve processing”, the second step is the processing of linear equations in characteristic 2, and the third step is the computation of algebraic square roots. In 2011, the second year of our project, the authors tuned the processing of linear equations in characteristic 2 for decryption software for use on the ES2.

2. RSA code

The common key cryptosystem and the public key cryptosystem are basic cryptosystems. The common key cryptosystem has only one key. It is simple and fast to process, but there is a problem to send the key in secret via the internet. The public key cryptosystem has two different keys for the encryption and decoding. The key for encryption is opened to the public, and the key for decoding is able to keep in secure, because it is not necessary to send the decoding key. A set of keys for the public key cryptosystem is based on the RSA code, which is innovated by R. L. Rivest, A. Shamir and L. M. Adlman in 1978. The RSA code uses two long digits prime numbers P and Q, and a prime number e, then computes $n = P\times Q$, $F = (P-1)\times(Q-1)$, and $D = e^{-1} \pmod F$. Numbers n and e are used for the key for encryption, and the number D is used for the key for decoding. The safeness of this is based on a result that, for a given long digits number n, the factorization

Table 1 Computational complexity of RSA-768 (768 bits, 232 digits).

	PC-year	Ratio (%)
Exploration of polynomial	20	1
Sieve processing	1500	90
Processing of linear equations	155	9
Algebraic square root	1	0
Others	1	0
Total	1677	100

algorithm of n to P and Q has high computational complexity and consumes enormous computation time.

3. Decryption of RSA code

The sieve method is a factorization method for a composite number N which obtains a relationship a^2-b^2=0 (mod N) for some a, b. Because natural numbers a and b are constructed as the products of prime numbers provided by the sieve, the exponent of each prime number must be an even number.

For a composite number N, we assume X is the nearest integer to $N^{1/2}$ and calculate $(X+k)^2$-N = A_k, k=0,1,2…. Then, we collect A_k that can be factorized using only prime numbers in factor base P. We can factorize N into a product of prime numbers using a combination of A_k whose exponent part is even. This provides the squared numbers for a^2-b^2=0 (mod N).

The processing of linear equations in characteristic 2 involves choosing a and b that satisfy a^2-b^2=0 (mod N). Since only the exponent part is tested for whether it is even or odd, we can use 0-1 binary computation. First, we create a matrix A whose elements are 0 or 1 by modulo operation from the sieved result, and then apply Gaussian Elimination to A+I, setting U equal to the result. If one row in the left part of U has all zero then non-zero columns in the right part of U indicate dependent rows. For these rows we test whether a^2-b^2=0 (mod N) can be satisfied. As the probability of being factored correctly is 50%, enormous rows having dependent columns must be listed.

Table 1 shows the computational complexity of RSA-768 in PC-year of AMD64 (2.2GHz).

4. Processing of linear equations in characteristic 2

Because the dimension of the matrix for the decryption of RSA code is more than 10^8 and each row has nearly a hundred non-zero elements, on average, for solving this system of linear equations, iterative methods are preferred over Gaussian elimination. Using an iterative method, the least squares solution of the A^Tx=0 is computed by the following steps:

(1) Set y_0 equal to a random 0-1 vectors,

(2) Compute the 0-1 vector b = AA^Ty_0,

(3) Solve the system of linear equations AA^Ty = b by iterative method,

(4) x = y - y_0 is a least squares solution of A^Tx=0.

The 64-bit integer can store 64 binary (0-1) numbers. The arrays for x, b, x_0, and y can each store 64 binary vectors, and can be computed at the same time by 64 parallel computation. In this study, the block Lanczos method is used as the iterative method. Binary vectors give an inner product equal to zero with probability 1/2, which is significant since division by zero in a simple Lanczos algorithm interrupts the iteration step in which it occurs. However, in the block Lanczos method with a block size of 64, a scalar in the simple Lanczos method becomes a 64×64 binary matrix in the processing of linear equations in characteristic 2, and the computation will not terminate if the matrix is regular. The 64 systems of linear equations with the same coefficient matrix are solved by the block Lanczos method; in this computation 64 solutions are computed simultaneously, and the iteration count is less than the dimension, N/64. The computation time of q= AA^Tp, α=p^Tq, and q=pα exceeds 99% in the block Lanczos method calculation, where A and A^T are binary (0-1) sparse matrices whose dimension are more than 10^8, p and q each consist of 64 binary (0-1) vectors, and α is a 64×64 binary matrix. The vectors p, q, and w and their arrays represent 64 binary vectors because a 64-bit integer can store 64 binary numbers.

The matrices and vectors are distributed among the nodes. To compute w=A^Tp and q=Aw in multiple nodes, an enormous amount of communication is needed. Since the vector p and w are computed in parallel and stored in distributed manner, the full values of p and w are copied to each node before the sparse matrix computation. There are eight CPUs in each node of the ES2 and they have a common memory space. Each node must store all portion of p and w for the hybrid parallelization, but each CPU must hold all of p and w for the pure MPI parallelization. Thus, pure MPI parallelization requires much more memory than the hybrid parallelization, and the difference between the memory requirements for the pure MPI parallelization and the hybrid parallelization is particularly significant when using many nodes.

The following considerations and facts are used:

4.1. Sparse matrix computation: q=AA^Tp

The full vector p must be stored in each node, whereas matrices and other vectors are distributed among nodes.

(1) Hybrid parallelization

Hybrid parallelization is applied to reduce the required memory. Automatic parallelization by the compiler is applied to each node, and MPI is used for parallelization among nodes.

(2) Both A and A^T are stored

To use long vector computation, matrices A and A^T are stored in different arrays on the ES2. Since the elements of binary matrices are 0 or 1, the locations of non-zero elements are stored rather than the values themselves. At each node, the number of non-zero elements in A and A^T set to be the same, and they are sorted at the beginning of the computation to construct long vectors on the ES2.

(3) AA^T is not computed

The matrix multiplication AA^T is not computed for the large

sparse matrices A and A^T. To obtain $q=AA^Tp$, the computations $w=A^Tp$ and $q=Aw$ are performed. Because nodes require the full value of p and w for this computation, the communications among nodes is necessary.

(4) Loop unrolling for the list-vector processing

Because the speed of list-vector processing on the ES2 strongly depends on the number of loop unrollings, the 2, 4, 8, and 16 unrolling by hand coding are tested.

4.2. Inner-product computation: $\alpha=p^Tq$

The following method A with loop unrolling is chosen because, although method B has less computational complexity, it is not vectorized.

Method A:

The 64 times 64 (64^2) inner-products are computed simultaneously. Since a 64-bit integer can hold 64 binary numbers, the result of 64^2 inner-products is stored in 64 arrays: TH[0] to TH[63]. The outmost loop k is unrolled by 8, which means eight parallel executions take place per node. The parallelization among nodes is done by dividing the dimension N by the number of nodes.

```
for (k=0; k<64; k++)
  { S = 0;
    for (i=0; i<N; i++)
      { Wk = (P[i] >> k) & 1;
        S ^= Wk*Q[i]; }
    TH[k] = S;
  }
```

Method B:

The amount of computation is one eighth of that of method A, but this method is not vectorized because of the dependency of W[ir]. This method is appropriate for a PC, but not for the ES2.

```
for (k=0; k<8; k++)
  { for (i=0; i<256; i++)  W[i] = 0;
    for (i=0; i<N; i++) { Bit = 8*k;  <-- Main part
      ir = (P[i] >> Bit) &255;  W[ir] ^= Q[i]; }
    for (i=0; i<8; i++)     { S = 0;
      for (j=0; j<256; j++)
        { is = (j >> i) & 1;  S ^= is*W[j];  }
          TH[k*8+i] = S;
}  }
```

4.3. Multiply-and-add computation: $q=p\alpha$

Method A:

TH[k] stores 64 times 64 (64^2) binary numbers for scalar α, and Q[i] stores the result of 64 times 64 vectors. This has approximately the same performance as method A for the inner-product computation on the ES2.

```
for (k=0; k<64; k++)
  { for (i=0; i<N; i++)
      { Wk = (P[i] >> k) & 1;
        Q[i] ^= Wk*TH[k]; }
  }
```

Method B:

The amount of computation is one eighth of that of method A, and this computation can be vectorized. The outmost loop j is unrolled by 8, and then 8 times 256 W[i] are computed simultaneously. This means eight parallel executions take place per node. The parallelization among nodes is done by dividing the dimension N.

```
for (j=0; j<8; j++)
  { j8 = j*8;
    for (i=0; i<256; i++)
      { k = 0;  id = i;  S = 0;
        while (id) { S ^= (id & 1)*TH[k+j8];
                     id  = id >> 1;  k++;  }
        W[i] = S;  }
    for (i=0; i<N; i++)                    <-- Main part
      { Q[i] ^= W[(P[i] >> j8) & 255];  }
  }
```

5 Performance of the processing of linear equations in characteristic 2

Figure 1 shows the performance of a transposed binary sparse matrix vector multiplication $w=A^Tp$ on one node of the ES2. The length of vector is sufficiently long but low performance is observed compared to having continuous memory access

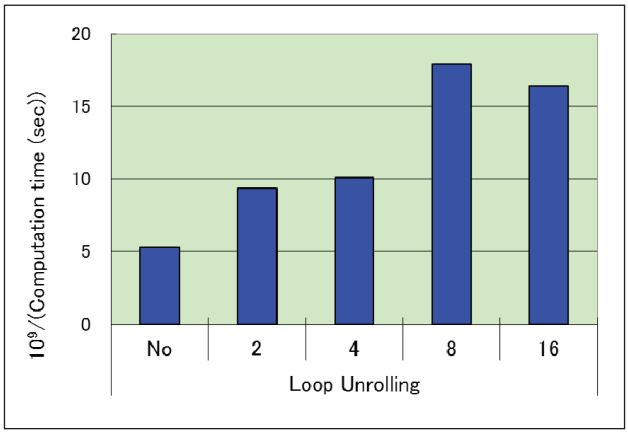

Fig. 1 Performance of binary sparse matrix multiplication $w=A^Tp$ on one node.

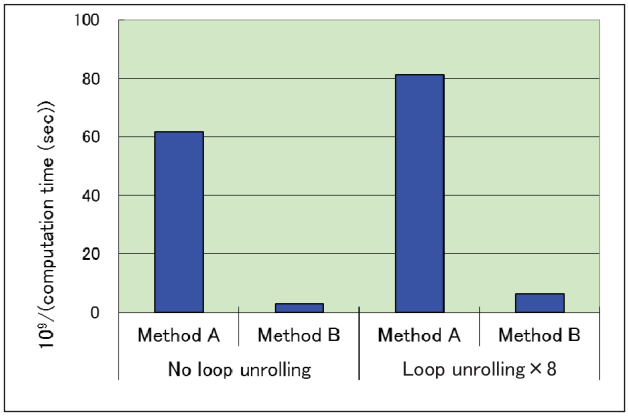

Fig. 2 Performance of inner-product $\alpha=p^Tq$ on one node.

because the use of a list-vector. The vector operation ratio exceeds 99%. The performance of a matrix vector multiplication q=Aw is nearly equal to that of $w=A^Tp$.

Figure 2 shows the performance of an inner-product $\alpha=p^Tq$ on one node of the ES2. A computational complexity of method B is assumed to be the same as that of method A. The real operation cost of method B is 1/8 times that of method A; however, method B has very poor performance because of the lack of vector computation. The vector operation ratio of the method A exceeds 99%.

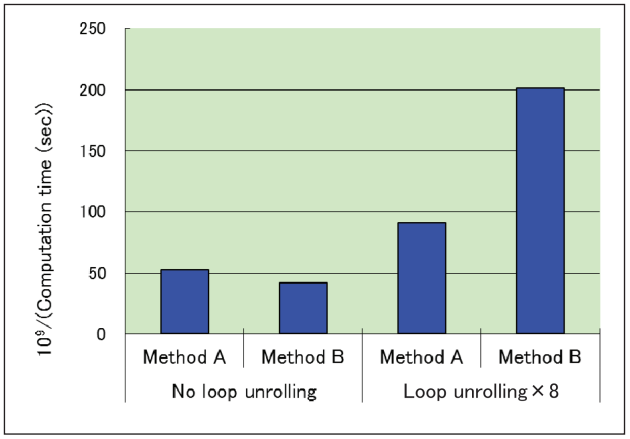

Fig. 3 Performance of multiply-and-add q=pα on one node.

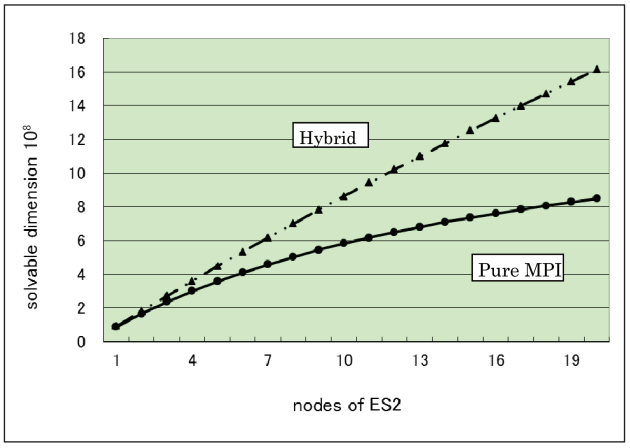

Fig. 4 Potential size problem when processing of linear equations in characteristic 2.

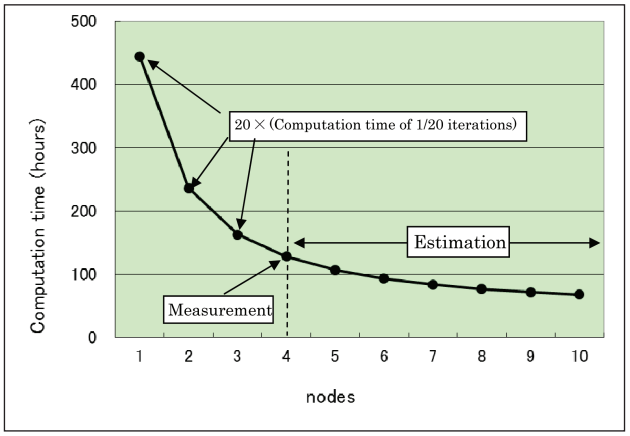

Fig. 5 Computation time of processing linear equations of dimension 8×10^7.

Figure 3 shows the performance of a multiply-and-add q=pα on one node of the ES2. The computational complexity of method B is assumed to be the same as that of method A. In this case, method B can be vectorized, after which it has a high performance value. The vector operation ratios of both methods exceed 99%.

Figure 4 shows a potential size problem when processing of the linear equations in characteristic 2. In the case of 20 nodes (160 CPUs) of the ES2, the dimension size of the hybrid parallelization is twice that of the pure MPI parallelization.

Figure 5 shows the computation time of processing of linear equations in characteristic 2 whose dimension is 8×10^7. The computation required 130 hours on 4 nodes of the ES2. The iteration count does not depend on the number of nodes. The computation time shown for 4 nodes is real measurement, whereas the values for 1, 2, 3 nodes are calculated by multiplying 20 the computation time for 1/20 of the total required iterations. The case of 5 to 10 nodes are estimated based on the other results and the communication cost among nodes. The dimension 8×10^7 was selected because it is the largest problem size that can be stored in the memory on one node of the ES2.

Figure 6 shows the speedups of the processing of linear equations in characteristic 2 whose dimension is 8×10^7 based on the result shown in Figure 5. The speedup is estimated to be 3.5 on 4 nodes and 5.8 on 8 nodes of the ES2. Herein, speedup of

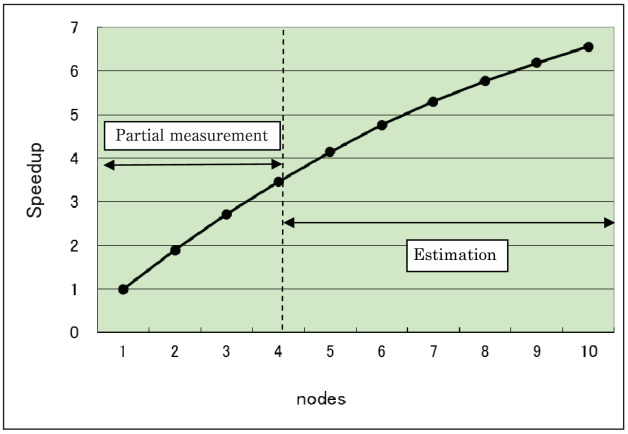

Fig. 6 Speedup of processing of linear equations of dimension 8×10^7.

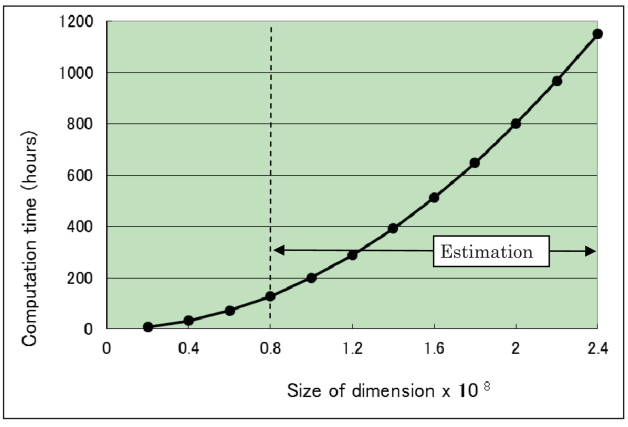

Fig. 7 Processing time of linear equations in characteristic 2 on 4 nodes.

an ES2 computation is defined as the time measured for a single node divided by the time measured for multiple nodes.

Figure 7 shows the estimation of computation time for the processing of linear equations in characteristic 2 on 4 nodes of the ES2. As the iteration count is proportional to the square of the dimension, an estimation of the iteration counts is calculated for measurement for dimensions of 2×10^7 to 8×10^7. An estimation of the computation time is calculated from the measurement of 1,000 iterations.

The computation time of the processing of linear equations in characteristic 2 whose dimension is 2×10^8 is estimated to be 800 hours on 4 nodes of the ES2. Four nodes of the ES2 are equivalent to 1,700 (AMD64 2.2GHz) CPUs, based on that the present world record RSA-768 is 150 CPU-year (AMD64 2.2GHz).

6. Summary

The authors tuned the processing of the linear equations in characteristic 2 in the RSA decryption processing. The following basic information on the ES2 were obtained:

1) This portion of the decryption processing is more than 99% vectorized, and has less floating-point operations. The list-vector processing consumes almost of all computation time and has accelerated by loop unrolling. The maximum improvement obtained by loop unrolling is 4.8-fold in the case of the multiply-and-add operation ($q=p\alpha$) .
2) Binary sparse matrices computation and multiply-and-add computation ($q=p\alpha$) have sufficient performance; however, the inner-product computation ($\alpha=p^Tq$) does not realized a good speedup relative to scalar performance because the faster method (method B) was not applicable.
3) Hybrid parallelization with automatic parallelization by compiler on one node and MPI parallelization among nodes is effective for the ES2.
4) The processing of the linear equations in characteristic 2 whose dimension is 8×10^7 requires 130 hours on 4 nodes of the ES2. The speedup of the processing of linear equations in characteristic 2 whose dimension is 8×10^7 is estimated to be 3.5 on 4 nodes and 5.8 on 8 nodes of the ES2.
5) The computation time of the processing of linear equations in characteristic 2 whose dimension is 2×10^8 (equivalent to RSA-768) is estimated 800 hours on 4 nodes of the ES2. This is equivalent to 1,700 (AMD64 2.2GHz) CPUs. The computation time of the processing of linear equations in characteristic 2 whose dimension is 8×10^8 (equivalent to RSA-898) is estimated 5,000 hours (7 months) on 16 nodes of the ES2.

Acknowledgement

The authors thank to Dr. Yoshinari Fukui and Mr. Tadashi Kai of JAMSTEC, who gave us precious advice on speeding up the ES2.

References

[1] Yasunori Ushiro, “RSA Decryption using Earth Simulator”, Annual Report of Earth Simulator Center, 167-171, 2011.
[2] Yasunori Ushiro, “A high speed sieve method for RSA cipher using the vector computer”, RIMS Kokyuroku 1733, 101-117, March 2011. (in Japanese)
[3] Richard A. Mollin, *“RSA and PUBLIC-KEY CRYPTOGRAPHY”*, CRC Press, 2002.
[4] Yuji Kida, “Prime factoring by General Number Field Sieve”, 2003. http://www.rkmath.rikkyo.ac.jp/~kida/nfs_intro.pdf (in Japanese)
[5] Junichi Yamazaki and Souta Kamaike, “The 2010 problem of the Cryptography”, ASCII technologies, 75-93, Sept. 2010. (in Japanese)
[6] Neal Koblitz, translate by Kouichi Sakurai, “*A Course in Number Theory and Cryptography*”, Springer, 1997. (in Japanese)
[7] Kazumaro Aoki, “Advances in Integer Factoring Technique: The Way to Factor RSA-768”, IPSJ Magazine, 51(8), 1030-1038, Aug. 2010. (in Japanese)

RSA 暗号の解読処理向け標数 2 の線形計算の高速化

プロジェクト責任者
長谷川 秀彦　　筑波大学　図書館情報メディア系
著者
後　　保範　　早稲田大学　教育学部　数学科
長谷川秀彦　　筑波大学　図書館情報メディア系

RSA 暗号の解読処理で「ふるい処理」の次に時間を占める「標数 2 の線形計算」を ES2 向けに高速化した。0-1 の 2 値データを扱うため、64 ビット整数ではサイズ 64 のブロックが自然に扱え、ブロックランチョス法が利用できる。0-1 の疎行列では値を格納する必要はなく、非ゼロ要素の位置だけを記憶すればよい。課題は 64 ビット整数の演算とリストベクトル処理の高速化であり、2 億次元 (RSA-768 相当) の標数 2 の線形計算は ES2 の 4 ノードで約 800 時間と推定され、約 1,700 台の PC クラスタと同程度の性能が達成できた。

RSA 暗号はインターネットを安全に使ううえで欠かせない技術である。RSA 暗号には桁数の多い合成数の因数分解の困難性が利用されており、現在使用されている 1,024 ビット RSA 暗号の安全性はスーパーコンピュータを数年使用しても解読されないという仮定のもとで成り立っている。これまでの RSA 暗号解読の世界記録はすべて PC クラスタで達成されており、ベクトル方式のスーパーコンピュータによる RSA 暗号の解読実験は全く報告されていない。そこで、本研究では代表的なベクトル方式のスーパーコンピュータである地球シミュレータ（ES2）において RSA 暗号の解読実験を行う。

一般的な RSA 暗号解読は、ふるい処理、標数 2（0-1 データ）の線形計算、代数的平方根の計算の 3 段階からなり、RSA 暗号解読プログラムはすべて PC 向きに開発されている。プロジェクト 2 年目となる本年度は「ふるい処理」の次に時間を占める「標数 2 の線形計算」と ES2 の相性を評価した。

標数 2 の線形計算にはブロックランチョス法を使用するため、疎行列とベクトルの乗算、内積計算および積和計算が必要となった。これらは PC とは異なる最適化が必要であるが、ES2 との相性は良いことが明らかになった。実際、64 ビット整数を用いると 0-1 データ 64 個分が一括計算できる。リストベクトル処理に対するループアンローリングの効果が大きいこと、疎行列 A および A^T とベクトルの乗算ではベクトル長を長くするため A と A^T を別に保持する必要があることなどが明らかになった。標数 2 の内積計算と積和計算は演算量を約 1/8 に削減する方法が知られているが、内積計算では演算を削減する方法のデータ依存性を取り除くことができなかった。ES2 では、内積計算は演算を削減しない方法、積和計算は演算を削減する方法を採用した。

この結果、8,000 万次元の標数 2 の線形計算が ES2 の 4 ノードを使用して約 130 時間で計算できる。8,000 万次元での並列性能は、4 ノードで 1 ノードの 3.5 倍、8 ノードで 5.8 倍と推定できた。浮動小数点数の演算はないが、ベクトル化率は 99% 以上で、ふるいと異なり並列化率は通信性能に依存する。これから、2 億次元（RSA-768 相当）の標数 2 の線形計算は ES2 の 4 ノードで約 800 時間と推定され、これは 2.2GHz の PC 1,700 台のクラクラスタと同程度の性能になる。

キーワード:RSA 暗号, 因数分解, 標数2, ブロックランチョス法, GNFS

謝辞

地球シミュレータ（ES2）における高速化について貴重なご意見を頂いた海洋研究開発機構地球シミュレータセンターの福井義成氏及び甲斐恭氏に謹んで感謝の意を表する。

参考文献

[1] Y. Ushiro, “RSA Decryption using Earth Simulator”, Annual Report of Earth Simulator Center, 167-171, 2011
[2] 後 保範 , “ベクトル計算機による RSA 暗号ふるいの高速化”, 京大数理解析研講究録 1733, 101-117, 2011 年 3 月
[3] Richard A. Mollin, “RSA and PUBLIC-KEY CRYPTOGRAPHY”, CRC Press, 2002
[4] 木田 祐司 , “数体ふるい法による素因数分解”, 2003; http://www.rkmath.rikkyo.ac.jp/~kida/nfs_intro.pdf
[5] 山崎 潤一 , 釜池 聡太 , “暗号の 2010 年問題”, ASCII Technologies, 75-93, 2010 年 9 月
[6] N. コブリッツ , 桜井 幸一訳 , “数論アルゴリズムと楕円暗号理論入門”, シュプリンガー・フェアラーク東京 , 1997 年
[7] 青木和磨呂 , “素因数分解技術の進展：RSA-768 の分解達成への道のり”, 情報処理 , 51(8), 1030-1038, 2010 年 8 月