PAPER

Multiple Scalar-Multiplication Algorithm over Elliptic Curve*

Kunio KOBAYASHI†, **Hikaru MORITA**†, *and* **Mitsuari HAKUTA**††, ***Regular Members***

SUMMARY This paper proposes an extended variant of the method by Brickel et al. for multiple scalar multiplication over an elliptic curve. In smartcard environments, the proposed method is superior to conventional methods. In particular, when the typical number of bases $t = 2$, the proposed method is four times faster than the simultaneous multiple exponentiation method, a well known fast method for multiple scalar multiplication. Furthermore, the proposed method can change the amount of time and memory to fit various platform environment (e.g., personal computers as rich ones or mobile devices such as smartcards as poor ones) by adjusting the division bit width (division unit).

key words: *multiple scalar multiplication, elliptic curve*

1. Introduction

Since the Internet has gained popularity, the importance of entity and message authentication is increasing. Moreover, new services among multiple clients via networks have been created. We need not only public-key schemes to authenticate entities and messages, but also higher schemes such as multi-party protocols for multiple users. Although portable (mobile) devices such as smartcards are needed to implement public-key schemes, they are limited in terms of their storage and computational powers. Furthermore, multi-party protocols should be speeded-up because of the marked increase in their usage.

Public-key schemes and multi-party protocols are constructed on modular arithmetic (e.g., modular exponentiation for the Diffie-Hellman scheme [4] and the Rivest-Shamir-Adleman (RSA) scheme [13]). The security of these schemes is based on the difficulty of the discrete logarithm problem or the integer factorization problem respectively. Since it is known that the factoring problem is efficiently solved by using the number field sieve method [8], we should use at least 1024 and 2048 bits in future. The discrete logarithm problem faces a similar situation. While smartcards with 1000-bit modular-exponentiation co-processors are entering the market, it is said that 1500-bit-version smartcards will be available in the near future. However, we cannot obtain longer-bit-version smartcards at reasonable cost. This is because long-bit exponentiation is time and hardware consuming. Furthermore, since algorithms to solve the factoring or discrete logarithm problem have been improved day by day, these conventional schemes might be come useless.

It is usually said that conventional schemes will be replaced by various schemes over elliptic curves. This is because the security of 1000-bit RSA can be equaled by a 160-bit elliptic curve scheme such as EC-DSA [5]. The strength of even 2000-bit RSA is equivalent to that of at most 200-bit EC-DSA. Consequently, elliptic-curve schemes are superior to the conventional schemes based on modular arithmetic. This is because no efficient method to solve discrete logarithms over elliptic curves has been found yet. (While sub-exponential-time methods to solve the factoring and discrete logarithm problems for the conventional schemes are known, existing methods to solve the discrete logarithm problems over an elliptic curve consume exponential time in proportion to key length.) However, the time needed to encrypt, decrypt, or generate signatures for elliptic curves isn't greatly different from the time needed by the conventional schemes implemented on the particular hardware such as a smartcard co-processor. Thus, efficient algorithms to speed up elliptic-curve schemes tend to be the focus of information security studies.

While modular-exponentiation algorithms are important when implementing the conventional schemes, the elliptic-curve schemes need algorithms to speed up scalar multiplications. On the other hand, highly-secure schemes such as provable-secure digital signatures [11], [12] and multi-party protocols (e.g., [2]) have been proposed. These algorithms use multiple scalar multiplication whose formula is $\sum_{i=0}^{t-1} k_i G_i$ where k_i is a scalar variable, G_i shows a rational point on an elliptic curve and i is an integer in $[0,\ t-1]$ where $t \geq 2$. However, there isn't any efficient multiple scalar multiplication algorithm, only single-base scalar multiplication and poor multiple-base versions. Thus, an efficient algorithm to implement these high-secure schemes is needed.

In this paper, we don't discuss any addition-chain method. This is because the addition-chain method doesn't suit the case of variable exponents and fixed bases, only fixed exponents and variable bases. We need an algorithm of variable exponents which are equivalent

Manuscript received March 29, 2000.
Manuscript revised October 6, 2000.

†The authors are with NTT Information Sharing Platform Laboratories, Yokosuka-shi, 239–0847 Japan.
††The author is with NTT Software Corporation, Yokohama-shi, 231-8551 Japan.

*This research was done at NTT Information Sharing Platform Laboratories.

to a variable scalar multipliers over an elliptic curve.

While the idea of the popular modular exponentiation algorithm known as binary method (p.461 of [6]) can be extended to multiple scalar multiplication, the simultaneous multiple exponentiation method, SME method (p.618 of [9]) can also be applied to multiplication. However, while the former is slow, the latter needs certain-fixed-capacity memory. On the other hand, it is known that precomputational methods are effective for fixed-base calculation. One of the precomputational methods, the BGMW method, was invented by Brickell, et al. [3].

This paper proposes a new method to compute elliptic-curve scalar multiplication having multiple bases by expanding the BGMW method for single-base scalar multiplication. The paper shows this new method's efficiency in a comparison to a simple expansion of the binary method and the SME method. Section 2 defines our notation. Section 3 shows procedures of the conventional algorithms, the binary method's expansion and the SME method. The proposed algorithm is given in Sect. 4. A discussion of the tradeoffs between time and memory is given in Sect. 5. This paper is concluded in Sect. 6.

2. Preparation

The following notations are used in this paper.

b	:	Division bit width (Division unit).
n	:	Key bit length (The size of a prime finite field).
t	:	The number of bases.
E	:	An elliptic curve defined over a prime finite field.
G	:	A point on elliptic curve E.
$-G$	:	The inverse of point G.
$\mathcal{O}$	:	The elliptic curve point at infinity as an additive identity or zero element.
$X+Y$	:	Addition operation on points X and Y defined over E.
kG	:	The scalar multiplication of elliptic curve point G by a non-negative integer k (the result of adding k copies of G).
k_j	:	A scalar multiplier ($j = 0 \sim t-1$).
G_j	:	Multiple bases as points on E ($j = 0 \sim t-1$).
$\sum_{j=0}^{t-1} k_j G_j$	:	Multiple-base scalar multiplication.
$\lceil x \rceil$	:	The smallest integer greater than or equal to the real number x.
$\lfloor x \rfloor$	:	The largest integer less than or equal to the real number x.

Let $(x_{n-1}, x_{n-2}, \ldots, x_0)_m$ be the m-adic representation of x, where the most significant digit is x_{n-1}.

3. Conventional Algorithms

In this section, two conventional algorithms to compute multiple scalar multiplication are introduced as preparation for the following discussion.

3.1 Multiple-Base Binary Method (MBB Method)

While the original binary method is applied to single modular exponentiation, the binary method can be modified to support multiple scalar multiplication over an elliptic curve as follows:

[MBB Method]

- Input:
  $G_{t-1},\ G_{t-2}, \ldots, G_0$;
  $k_{t-1},\ k_{t-2}, \ldots, k_0$
  where $\begin{cases} k_j = (k_{j,n-1}\ k_{j,n-2} \cdots k_{j,0})_2 & \text{and} \\ 0 \le j \le t-1. \end{cases}$
- Output:
  $S = k_{t-1}G_{t-1} + k_{t-2}G_{t-2} + \cdots + k_0 G_0$.
- Procedure:

step1: $S \leftarrow \mathcal{O}$.
step2: For i from $n-1$ down to 0, do the following:
  step2.1: $S \leftarrow 2S$.
  step2.2: For each j, do:
    if $k_{j,i} = 1$ then $S \leftarrow S + G_j$.
step3: Return S.

3.2 Simultaneous Multiple Exponentiation Method (SME Method)

When multiple bases are fixed, a precomputed reference table can accelerate computational speed. While the original SME method with a precomputed reference table was applied to multiple modular exponentiation (see §14.88 of [9]), the SME method can be modified to support multiple scalar multiplication over an elliptic curve as follows:

[SME Method]

- Input:
  $G_{t-1}, G_{t-2}, \ldots, G_0$;
  $k_{t-1},\ k_{t-2} \ldots, k_0$
  where $\begin{cases} k_j = (k_{j,n-1}\ k_{j,n-2} \cdots\ k_{j,0})_2 & \text{and} \\ 0 \le j \le t-1. \end{cases}$
- Output:
  $S = k_{t-1}G_{t-1} + k_{t-2}G_{t-2} + \cdots + k_0 G_0$.
- Precomputation:
  For i from 0 to $2^t - 1$, do: $M[i] \leftarrow \sum_{j=0}^{t-1} i_j G_j$,
  where $i = (i_{t-1}, i_{t-2}, \ldots, i_0)_2$.
- Procedure:

step1: S $\leftarrow \mathcal{O}$.
step2: For i from $n-1$ down to 0, do the following:

step2-1: $S \leftarrow 2S$.
step2-2: $S \leftarrow S + M[(k_{t-1,i}\ k_{t-2,i} \cdots k_{0,i})_2]$.

step3: Return S.

4. Proposed Method

Section 3 introduced two methods to compute multiple scalar multiplication. They are more efficient than methods that sum up products of single scalar multiplication of each base. However, since both of them, the MBB and SME methods, don't have tradeoffs between computation speed and memory applied to computational environment, it's difficult to achieve design flexibility to match any given platform.

This section proposes a new efficient variant of the BGMW method (Original ones is given in Appendix A and B) for multiple scalar multiplication. Our proposed method offers a tradeoff between computational time and memory by adjusting division unit. Thus, optimal designs are possible for a variety of platforms.

We extend the BGMW method, which compute single-base scalar multiplication, to multiple scalar multiplication over elliptic curve. We compute the following multiple scalar multiplication.

$$S = k_{t-1}G_{t-1} + k_{t-2}G_{t-2} + \cdots + k_0G_0$$

where k_j is a scalar variable, G_j shows a rational point on an elliptic curve and j is an integer in $[0, t-1]$ where $t \geq 2$.

If k_j is represented as follows,

$$k_j = k_{j,m}2^{mb} + k_{j,m-1}2^{(m-1)b} + \cdots + k_{j,1}2^b + k_{j,0}2^0,$$

$$\text{where} \begin{cases} m = \lfloor (n-1)/b \rfloor, \\ -2^{b-1} \leq k_{j,i} \leq 2^{b-1}, \\ 0 \leq j \leq t-1 \quad \text{and} \\ 0 \leq i \leq m, \end{cases}$$

then we obtain the next equation.

$$\begin{aligned} S = {} & k_{t-1,m}2^{mb}G_{t-1} + k_{t-1,m-1}2^{(m-1)b}G_{t-1} \\ & + \cdots + k_{t-1,0}2^0G_{t-1} \\ & + k_{t-2,m}2^{mb}G_{t-2} + k_{t-2,m-1}2^{(m-1)b}G_{t-2} \\ & + \cdots + k_{t-2,0}2^0G_{t-2} \\ & \vdots \\ & + k_{0,m}2^{mb}G_0 + k_{0,m-1}2^{(m-1)b}G_0 \\ & + \cdots + k_{0,0}2^0G_0 \end{aligned}$$

Therefore, if we have the precomputation value $2^{ib}G_j$, we can compute the above S efficiently.

The procedure of the proposed method is as follows:

[Proposed Method]

- Input:
  $G_{t-1},\ G_{t-2}, \ldots, G_0$;
  $k_{t-1},\ k_{t-2}, \ldots, k_0$,
  $$\text{where} \begin{cases} k_j = (k_{j,m}\ k_{j,m-1} \cdots\ k_{j,0})_{2^b}, \\ -2^{b-1} \leq k_{j,i} \leq 2^{b-1}, \\ m = \lfloor (n-1)/b \rfloor, \\ 0 \leq j \leq t-1 \quad \text{and} \\ 0 \leq i \leq m. \end{cases}$$
- Output:
  $S = k_{t-1}G_{t-1} + k_{t-2}G_{t-2} + \cdots + k_0G_0$.
- Precomputation:
  $M[\,][\,] = \{$
  $\{2^{mb}G_{t-1},\ 2^{(m-1)b}G_{t-1}, \ldots, G_{t-1}\}$
  $\{2^{mb}G_{t-2},\ 2^{(m-1)b}G_{t-2}, \ldots, G_{t-2}\}$
  $\vdots$
  $\{2^{mb}G_0,\ 2^{(m-1)b}G_0, \ldots, G_0\}$
  $\}$.
- Procedure:

step1: $S \leftarrow \mathcal{O}$, $T \leftarrow \mathcal{O}$.
step2: For i from 2^{b-1} down to 1, do the following:

step2.1: For each j, l for which $|k_{j,l}| = i$ do:
$$T \leftarrow T + \text{sign}(k_{j,l}) \cdot M[t-1-j][l].$$
$$\text{where} \begin{cases} 1 \leq l \leq m, \\ \text{sign}(x) = \begin{cases} 1, & \text{for } x \geq 0 \\ -1, & \text{otherwise.} \end{cases} \end{cases}$$
step2.2: $S \leftarrow S + T$.

step3: Return S.

5. Tradeoffs between Time and Memory

This section describes the efficiency of the proposed method. When we discuss numerical values, the key bit length $n = 160$ is selected. This is because the security of a 160-bit elliptic curve cryptographic scheme such as EC-DSA is equal to the security of 1000-bit RSA which is widely used.

5.1 Time and Memory

In Table 1, equations show computational delay time and memory needed for the SME method given in Sect. 3 and the proposed algorithm in Sect. 4. Computational delay time corresponds to the average number of addition over elliptic curve, and memory needed corresponds to the number of elliptic curve points in a precomputation table. The MBB method doesn't need extra memory except working area to compute. In Table 1, 'Memory needed' doesn't include memory of t fixed bases.

5.2 Time Ratio

Figures 1 and 2 show time ratio of the proposed method

Table 1 Computational complexity and memory needed for the SME method and the proposed method.

Method	Computational complexity (the av. number of addition)	Memory needed (the number of points)
MBB	$(nt-1)/2+n-1$	————
SME	$(n-1)(2-1/2^t)$	2^t-t-1
Proposed	$\lceil n/b \rceil t+2^{b-1}-2$	$(\lceil n/b \rceil -1)\,t$

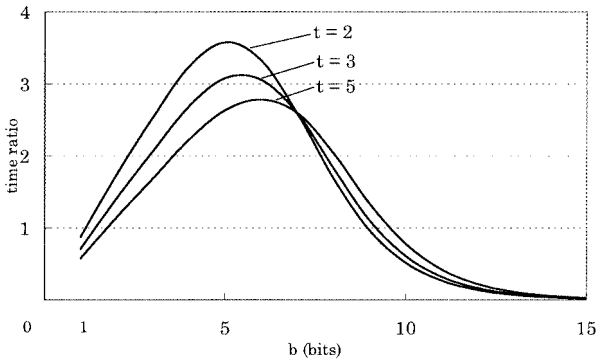


Fig. 1 Time ratio of the MBB method to the proposed method. ($n = 160$)

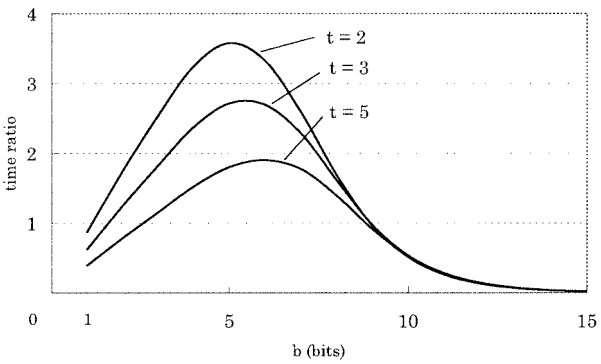


Fig. 2 Time ratio of the SME method to the proposed method. ($n = 160$)

to the MBB method and the SME method, respectively. Their horizontal coordinate shows division unit for the proposed algorithm. Their vertical coordinate represents the ratio of the conventional method's delay to the proposed one's delay. When the number of bases ranges from two through five, Figs. 1 and 2 show that the proposed algorithm is 2 through 3.5 times faster than the two conventional algorithms at the peaks of their curves.

5.3 Memory

We discuss the amount of precomputation memory consumed by each method. Since the MBB method is slow, we focus on more efficient methods that involve extra precomputation memory: the proposed and the SME methods. Table 2 shows the amount of memory needed by these methods for typical numbers of bases

Table 2 Memory needed by the proposed method and the SME method. ($n = 160$)

Number of bases	Proposed method	SME method
2	1240	40
3	3720	160
5	5200	1040

unit: Byte

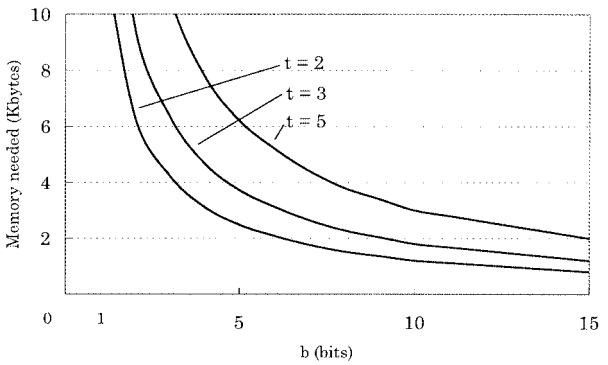


Fig. 3 Memory needed by the proposed method versus division unit. ($n = 160$)

"t": $t = 2$ through 5. In the same way, Fig. 3 shows the relation between division unit and the amount of memory needed by the proposed method for the same numbers of bases. As seen in Table 2 and Fig. 3, though the SME method consumes less precomputation memory than the proposed method, the proposed method is more efficient. Since the amount of memory used by the proposed method is acceptable in the real use (It is less than five Kbytes), the proposed method is superior to the SME method.

5.4 Results of Performance Estimation

When elliptic curve schemes are implemented, there are three options for finite fields: F_{2^n}, F_p and F_{q^r} where $n = \lceil \log(p) \rceil = r\lceil \log(q) \rceil$. (Refer to [5] for F_{2^n} and F_p types; and to [1], [7] for an F_{q^r} type.) F_p is selected for this experiment because smartcard co-processors in the market suit to compute an F_p-type elliptic curve.

Table 3 shows the relation between the number of bases and delay time. The delay time is estimated by using the time of addition and doubling operations by a real smartcard (We use delay time; addition: 1.584 ms, doubling: 1.933 ms as estimated on a smartcard specified in Appendix C). We can see that the proposed method has a definitive advantage to the SME equiva-

Table 3 Delay time by the proposed method and the SME method. ($n = 160$)

Number of bases	Proposed method	SME method
2	124	496
3	174	528
5	261	551

unit: msec

lent. In particular, when $t = 2$, the proposed method is four times faster.

6. Conclusion

This paper proposed an extended variant of the BGMW method for multiple scalar multiplication over an elliptic curve, and also showed this new method's efficiency in a comparison to a simple expansion of the binary method and the SME method. In smartcard environments, the proposed method is superior to the conventional methods. In particular, when the typical number of bases $t = 2$, the proposed method is four times faster than the SME method, which is well known as a fast method for multiple scalar multiplication. Furthermore, the proposed method can vary the amount of time and memory to fit various platform environment (e.g., personal computers as rich ones or mobile devices such as smartcards as poor ones) by adjusting the division bit width (division unit).

This method is considered to be an efficient algorithm to speed up elliptic-curve schemes that uses multiple scalar multiplication, which tend to be the focus of information security studies.

References

[1] D.V. Bailey and C. Paar, "Optimal extension fields for east arithmetic in public-key algorithms," Advances in Cryptology CRYPTO'98, Lecture Notes in Computer Science, vol.1462, pp.472–485, Springer-Verlag, 1998.
[2] M. Bellare, J.A. Garay, and T. Rabin, "Fast batch verification for modular exponentiation and digital signatures," Advances in Cryptology EUROCRYPT'98, Lecture Notes in Computer Science, vol.1403, pp.236–250, Springer-Verlag, 1998.
[3] E. Brickell, D. Gordon, K. McCurley, and D. Wilson, "Fast exponentiation with precomputation," Advances in Cryptology EUROCRYPT'92, Lecture Notes in Computer Science, vol.658, pp.200–207, Springer-Verlag, 1993.
[4] W. Diffie and M.E. Hellman, "New directions in cryptography," IEEE Trans. Inf. Theory, vol.IT-22, no.11, pp.644–654, Nov. 1976.
[5] IEEE P1363, "Standard specifications for public key cryptography," http://grouper.ieee.org/groups/1363/draft.html, D9 (Draft Version 9), Feb. 1999.
[6] D.E. Knuth, "Seminumerical algorithms," The Art of Computer Programming, vol.2, 3rd ed., Addison-Wesley, 1998.
[7] T. Kobayashi, H. Morita, K. Kobayashi, and F. Hoshino, "Fast elliptic-curve algorithm combining Frobenius map and table reference method to adapt to higher characteristic," Advances in Cryptology EUROCRYPT'99, Lecture Notes in Computer Science, vol.1592, pp.176–189, Springer-Verlag, 1999.
[8] A.K. Lenstra, H.W. Lenstra, Jr., M.S. Manasse, and J.M. Pollard, "The number field sieve," Proc. STOC, pp.564–572, 1990.
[9] A.J. Menezes, P.C. vanOorschot, and S.A. Vanstone, Handbook of applied cryptography, CRC Press, Boca Raton, 1997.
[10] NTT NEWS RELEASE, "NTT develops super-multi-purpose smart card," http://www.ntt.co.jp/news/news98e/981016a.html, Oct. 16. 1998.
[11] T. Okamoto, "Provably secure and practical identification schemes and corresponding signature schemes," Advances in Cryptology CRYPTO'92, Lecture Notes in Computer Science, vol.740, pp.31–53, Springer-Verlag, 1993.
[12] T. Okamoto, "Practical identification schemes as secure as the DL and RSA problems," http://grouper.ieee.org/groups/1363/addendum.html#Okamoto, March 1999.
[13] R.L. Rivest, A. Shamir, and L. Adleman, "A method for obtaining digital signatures and public-key cryptosystems," Comm. ACM, vol.21, no.2, pp.120–126, Feb. 1978.

Appendix A: BGMW Method

The original BGMW method is as follows:

- Input:
  g;
  n
  where $\begin{cases} n = \sum_{i=0}^{m-1} a_i x_i & \text{and} \\ 0 \le a_i \le h. \end{cases}$
- Output:
  $S = g^n$.
- Precomputation:
  g^{x_i} for each $0 \le i < m$.
- Procedure:

step1: $S \leftarrow 1$, $T \leftarrow 1$.
step2: For $i = h$ to 1 by –1

  step2.1: For each j such that $a_j = i$
    $T \leftarrow T * g^{x_j}$.
  step2.2: $S \leftarrow S * T$.

step3: Return S.

The number of multiplications performed by the algorithm can be counted as follows:
There are m digits, so the step2.1 gets executed at most m times. The step 2.2 gets executed h times. Finally, at least two of these multiplications are free since S and T are initially 1. Therefore, g^n can be computed with $m + h - 2$ multiplications in this procedure.

Appendix B: BGMW Method (Extension to an Elliptic Curve)

The application to single-base multiple scalar multiplication over an elliptic curve of the BGMW method is as follows:

- Input:
  G;
  k
  $$\text{where} \begin{cases} k = (k_m k_{m-1} \cdots k_0)_{2^b}, \\ -2^{b-1} \leq k_i \leq 2^{b-1}, \\ m = \lfloor n/b \rfloor \quad \text{and} \\ 0 \leq i \leq m. \end{cases}$$
- Output:
  $S = kG$.
- Precomputation: $M[\,] = \{2^{mb}G,\ 2^{(m-1)b}G, \ldots, G\}$.
- Procedure:

step1: $S \leftarrow \mathcal{O}$, $T \leftarrow \mathcal{O}$.
step2: For i from 2^{b-1} down to 1, do the following :

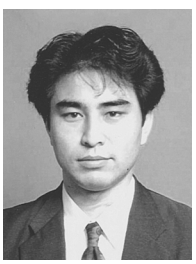
step2.1: For each l for which $|k_l| = i$ do:
$T \leftarrow T + sign(k_l) \cdot M[l]$.
$$\text{where} \begin{cases} 1 \leq m, \\ sign(x) = \begin{cases} 1, & \text{for } x \geq 0 \\ -1, & \text{otherwise.} \end{cases} \end{cases}$$
step2.2: $S \leftarrow S + T$.

step3: Return S.

Appendix C: Super-Multipurpose Smartcard

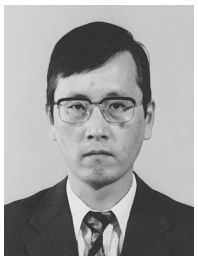
The following table gives the main functions and qualities of the Super–Multipurpose Smartcard [10].

CPU	16-bit unit, 15-MHz clock
Co-processor	1152-bit modular exponentiation
RAM	2 KBytes
Non-volatile erasable memory	512-KByte Flash memory, Rewritable 100,000 times
Input/Output interface	Transmission speed: 9.6 to 76 Kbps. Compatible with ISO7816. Functions for downloading programs.



Kunio Kobayashi received the B.E. and M.E. degrees from Waseda University, Tokyo, Japan, in 1995, and 1997, respectively. He is a researcher of NTT information Sharing Platform Laboratories. He was awarded SCIS'97 paper prize.

Hikaru Morita received the B.E., M.E. and Dr.E. degrees from Hokkaido University, Sapporo, Japan, in 1980, 1982 and 1993, respectively. He is a Senior Research Engineer, Supervisor of NTT Information Sharing Platform Laboratories and a Guest Professor, Graduate School of Information Systems, University of Electro-Communications. He is presently engaged in research and standardization on Information Security. Dr. Morita is a member of IEEE and IPSJ.



Mitsuari Hakuta received the B.E., M.E. and Dr.E. degrees in Electronics Engineering from Tokyo Institute of Technology, Tokyo, Japan, in 1971, 1973 and 1997, respectively. He had been a research engineer at NTT Laboratories since 1973. He has joined NTT Software Corporation since April 2000. His research interests include software engineering and security systems.