IEICE TRANS. FUNDAMENTALS, VOL.E83–A, NO.1 JANUARY 2000

PAPER *Special Section on Cryptography and Information Security*

New Methods for Generating Short Addition Chains

Noboru KUNIHIRO[†] ***and*** **Hirosuke YAMAMOTO**[††], ***Members***

SUMMARY Power exponentiation is an important operation in modern cryptography. This operation can be efficiently calculated using the concept of the addition chain. In this paper, two new systematic methods, a Run-length method and a Hybrid method, are proposed to generate a short addition chain. The performance of these two methods are theoretically analyzed and it is shown that the Hybrid method is more efficient and practical than known methods. The proposed methods can reduce the addition chain length by 8%, in the best case, compared to the Window method.

key words: *addition chain, hamming weight, extended window method, run-length method, hybrid method*

1. Introduction

The encryption and decryption in the RSA scheme [9] consist of the power exponentiation: $M^e \bmod n$. This operation is very important and is also used in prime testing algorithms, integer factoring algorithms, and so on. Fast calculation of this operation is realized by reducing the number of multiplications as much as possible and/or speeding up the operation of multiplication. In this paper, we consider the former problem, i.e., how to reduce the number of multiplications.

It is well known that the power exponentiation can be efficiently calculated using the addition chain [3]. Suppose that M^e can be calculated as $M^1 \to M^{a_1} \to M^{a_2} \to \cdots \to M^{a_r}$ $(= M^e)$ based on a rule:

$$M^{a_i} = M^{a_j} \times M^{a_k}, \;\; j,k < i. \tag{1}$$

The sequence of exponents a_i, $\langle a_0 \;(= 1), a_1, a_2, \cdots, a_r$ $(= e)\rangle$, is called an addition chain since this sequence satisfies the relation

$$a_i = a_j + a_k, \;\; j,k < i. \tag{2}$$

Note that the chain length r is equal to the number of multiplications used to calculate M^e.

Finding a short addition chain for a given e leads to the fast calculation of the power exponentiation and, hence, fast encryption and decryption in the RSA scheme. However, obtaining the shortest chain is an NP-hard problem [2]. Therefore, many algorithms have been proposed to obtain a sub-optimal chain, for example, the Bos-Coster method [1], the Yacobi method [11], the m-ary or 2^κ-ary method [3], the Lou-Chang method [7], the Window method [3], and the Extended Window method with the Tunstall-like algorithm [6], etc. In [6], we showed that the Window method is optimal under the following two conditions. One is that a power exponent e is uniformly randomly distributed, and the other is that only the doubling rule $a_i = 2a_{i-1}$ and the star chain rule $a_i = a_{i-1} + a_k, \; k < i$, are allowed in Eq. (2).

However, non-uniform cases have not been analyzed in details. In this paper, we mainly treat non-uniform cases. In order to derive a short addition chain, we propose two methods, a Run-length method* and a Hybrid method, which are different from the known methods. We theoretically show that the Hybrid method is as efficient as the Extended Window method, which is optimal even for non-uniform cases under the above second condition, and the Run-length method is efficient when the Hamming weight of the binary representation of the power exponent e is large. It is also shown that the performance of the Hybrid method is better than the other known methods.

The Hybrid method and Run-length method are also applicable to elliptic curve cryptosystems [4], [8]. However, the scalar multiplication over elliptic curves can use an addition-subtraction chain instead of an addition chain since an inverse element can be easily obtained. The computation rule of the addition-subtraction chain is given by $a_i = a_j \pm a_k, \; j,k < i$ instead of Eq. (2). An efficient method for obtaining a short addition-subtraction chain based on the canonical signed binary representation of e is described in [6].

In this paper, the base of logarithm is always 2. Let $Seq = s_1 s_2 \cdots s_L, s_l \in \{0,1\}$, be the ordinary binary representation of a positive integer e, and $(Seq)_{10} = e$. $|Seq|$ stands for the length of Seq, i.e., L, which is equal to $\lfloor \log e \rfloor + 1$. $w(e)$ represents the Hamming weight of e, in other words, $w(e)$ is the number of "1" in $Seq = s_1 s_2 \cdots s_L$, i.e., $w(e) = \sum_{i=1}^{L} s_i$. We assume that the bits "0" and "1" occur with the probability of p and q, respectively, i.e., $q = \Pr\{s_l = 1\} = w(e)/L$ and $p = \Pr\{s_l = 0\} = 1 - q$. The addition chain length r must satisfy $r \geq L + \log(qL) - 2.13$ for any e [10].

Manuscript received March 24, 1999.
Manuscript revised July 2, 1999.

[†]The author is with NTT Communication Science Laboratories, Kyoto-fu, 619–0237 Japan.
[††]The author is with the School of Engineering, The University of Tokyo, Tokyo, 113–8656 Japan.

*Run-length method is firstly proposed in [5].

2. Window Methods

In this section, we briefly review the Window method [3] and the Extended Window method with the Tunstall-like algorithm [6].

2.1 Window Method

The addition of the Window method is restricted by the following two rules[†]:

$$\text{doubling rule: } b_i = b_{i-1} + b_{i-1} = 2b_{i-1}, \tag{3}$$

$$\text{star chain rule: } b_i = b_{i-1} + a_k,\ a_k \in \mathcal{D}, \tag{4}$$

where $\mathcal{D}$ is a set of all odd integers less than 2^κ. Hereafter, we call $\mathcal{D}$ as "dictionary." Note that the operations based on the doubling rule $b_i = 2b_{i-1}$ and star chain rule $b_i = b_{i-1} + a_k$ correspond to $M^{b_i} = (M^{b_{i-1}})^2$ and $M^{b_i} = M^{b_{i-1}} \cdot M^{a_k}$ in the power exponents, respectively.

The Window method is formulated as follows.

Input an integer e (or the binary representation of e, *Seq*).

Output an addition chain for e, $\langle c_0\ (= 1), c_1, \cdots, c_r = e\rangle$.

Step 1 Determine the window size κ appropriately from L and $w(e)$.

Step 2 Make the shortest addition chain for the dictionary $\mathcal{D}$. In this case, it is given by $\langle 1, \underline{2}, 3, 5, \cdots, 2^\kappa - 1\rangle$, where the underline represents the number not included in the dictionary.

Step 3 [Main Procedure]
Process the sequence *Seq* from the most significant bit in the following two phases.

Phase 1 Read κ bits from *Seq*. Let the κ bits be $1s_1 \cdots s_{\kappa-l-1}\underbrace{0\cdots0}_{l}$, where $s_{\kappa-l-1} = 1$ and $(1s_1 \cdots s_{\kappa-l-1})_{10} = a$. First apply the doubling rule $b_i = 2b_{i-1}$, $\kappa - l$ times. Next, apply the star chain rule $b_i = b_{i-1} + a$ and finally apply the doubling rule l times. Remove the κ bits from *Seq*.

Phase 2 If $Seq = \underbrace{0\cdots0}_{l_0}1\cdots$, then apply the doubling rule $b_i = 2b_{i-1}$, l_0 times. Remove the l_0 bits from *Seq* and return to Phase 1.

Step 4 Concatenate $\langle 1, 2, 3, 5, \cdots, 2^\kappa - 1\rangle$ in Step 2 and $\{b_i\}$ in Phases 1 and 2 of Step 3 to obtain the whole addition chain $\langle c_0, c_1, \ldots, c_r\rangle$.

Note 1: After applying the doubling rule or the star chain rule, the index i is increased one. All methods described in Sects. 2 and 3 obey this rule.

Note 2: The sequence is parsed by κ bits in Phase 1 of Step 3, but the parse length in Phase 2 depends on the sequence.

Note 3: When $\kappa = 1$, the above method is equivalent to the binary method [3].

Note 4: The average chain length is given as follows [6].

$$\left(L - \left(\kappa - \frac{p - p^\kappa}{1-p}\right)\right) + \frac{L}{\kappa + \frac{p}{1-p}} + 2^{\kappa-1}, \tag{5}$$

where the first and second terms are the average numbers of the applied doubling and star chain rules in Step 3, respectively. The third term is the chain length for the dictionary $\mathcal{D}$ in Step 2. When $L = 512$ and $p = 1/2$, the optimal window size κ is equal to 5 and the average chain length is 609.

Note 5: The 2^κ-ary method [3] is similar to the Window method. This method can be easily implemented, but it is less efficient than the Window method. For instance, the chain length obtained by the 2^κ-ary method is $L - \kappa + L/\kappa + 2^\kappa$, which becomes 640 for $\kappa = 5$ in the above case. The average chain length obtained by its variant, the Lou-Chang method [7], is 613 in the same case.

2.2 Extended Window Method with Tunstall-Like Algorithm

As we have seen in the previous subsection, the set of all odd positive integers less than 2^κ is adopted as the dictionary in the Window method. On the other hand, a more flexible dictionary can be used in the Extended Window Method (EWM) proposed in [6]. Moreover, the "Tunstall-like algorithm" was introduced to make the optimal dictionary in the same paper. In this subsection, we review the EWM and the Tunstall-like algorithm.

The EWM is formulated as follows.

Input an integer e (or the binary representation of e, *Seq*).

Output an addition chain for e, $\langle c_0\ (= 1), c_1, \cdots, c_r = e\rangle$.

Step 1 Determine the primitive dictionary[††] $\mathcal{D}_s = \{Seq_0, Seq_1, \cdots Seq_K\}$ appropriately from L and $w(e)$ and obtain the dictionary $\mathcal{D} = \{a_0\ (= 1), a_1, \ldots, a_K\} = \{(Seq'_j)_{10}\}_{j=0}^{K}$, where Seq'_j is the sequence obtained by removing trailing zeros, in Seq_j, ending at the least significant bit position[†††].

Step 2 Make the shortest addition chain for $\mathcal{D}$.

[†]All methods described in Sects. 2 and 3 use these two rules. However, only the concatenation point of the addition chains obtained by Steps 2 and 3 described later can break this rule in all methods.

[††]The most significant bit of each Seq_j is "1."

[†††]If $Seq_j = 1s_1s_2\cdots s_l0\cdots0$, then $Seq'_j = 1s_1s_2\cdots s_l$.

Step 3 [Main Procedure]
Process the sequence Seq from the most significant bit in the following two phases.
Phase 1

(a) Find Seq_j that is equal to the prefix of Seq.
(b) Apply the doubling rule $b_i = 2b_{i-1}$, $|Seq'_j|$ times.
(c) Apply the star chain rule $b_i = b_{i-1} + a_j$.
(d) Again apply the doubling rule $|Seq_j| - |Seq'_j|$ times.
(e) Remove the prefix of Seq.

Phase 2

(f) If the prefix of Seq is $\underbrace{0\cdots0}_{l_0}1$, then apply the doubling rule $b_i = 2b_{i-1}$, l_0 times.
(g) Remove the prefix $\underbrace{0\cdots0}_{l_0}$ from Seq and return to Phase 1.

Step 4 Concatenate the addition chains obtained in Steps 2 and 3.

Note 6: The primitive dictionary $\mathcal{D}_s$ is a set such that $\mathcal{D}_s$ and 0^*, zero sequences with arbitrary length, uniquely parse any Seq.

Note 7: In Step 2, it is possible to make the shortest addition chain for $\mathcal{D}$ unless K is very large. But it becomes harder as K becomes larger.

Note 8: In general, finding Seq_j in Phase 1 (a) of Step 3 is executed by using a tree obtained from the primitive dictionary determined in Step 1, and the optimal primitive dictionary is obtained by the Tunstall-like algorithm described later. Hence, the implement of the EWM is a little complicated than other methods, which can be implemented without a tree structure, such as the Window method, the Run-length method (described in Sect. 3.1) and the Hybrid method (described in Sect. 3.3).

Note 9: At the first parsing of Seq, (b) of Phase 1 can be skipped.

The optimal primitive dictionary $\mathcal{D}_s$ is constructed as follows.

Tunstall-like algorithm [6]

Input a dictionary size K, bit length of e (i.e. L), and the Hamming weight of e (i.e. $w(e)$).
Output a primitive dictionary $\mathcal{D}_s$.
Step 1 Make the root of a tree with weight 1. Set $q = w(e)/L$ and $p = 1 - q$.
Step 2 While the number of leaves is less than $K + 1$, repeat the following.
Let l and $weight(l)$ be the leaf with largest weight and its weight, respectively. Create two children with weight $p \times weight(l)$ and $q \times weight(l)$, and connect them to l via edges labeled with "0" and "1," respectively.
Step 3 Get binary sequences by reading along all paths from the root to all leaves. Then each element of $\mathcal{D}_s$ is obtained by concatenating "1" toeach sequence as the most significant bit.

Note that the optimal dictionary depends on the Hamming weight.

In [6], the Tunstall-like algorithm is derived from the duality between the minimization problem of the chain length in the EWM and the optimization problem of variable-to-fixed length codes in data compression theory. We showed in [6] that the EWM reduces to the Window method when $p = 0.5$. Furthermore, it is also proved that the average chain length asymptotically converges $L + H(p)\frac{L}{\log L}$, where $H(p) \equiv -p\log p - (1-p)\log(1-p)$. This result and Eq. (5) imply that in case of $p < 0.5$, the addition chains of the EWM and the Window method become shorter and longer, respectively, as the Hamming weight becomes larger. Hence, the Window method is not always optimal, especially when e has a large Hamming weight. In the next section, we propose two new algorithms that can be more easily implemented than the EWM. In addition, the two methods are as efficient as the EWM in the case of a large Hamming weight, i.e., small p.

3. New Methods

In this section, we propose the Run-length method and the Hybrid method. The former can make a shorter addition chain than the Window method when p is very small.The latter is a hybrid of the Window method and the Run-length method.

3.1 Run-Length Method

First, consider a case with small p. The optimal primitive dictionary obtained by the Tunstall-like algorithm, for example when $K = 512$ and $p = 0.1$, becomes

$$\mathcal{D}_s = \{10, 110, 1110, \cdots, \underbrace{1\cdots1}_{13}0, \underbrace{1\cdots1}_{14}\}.$$

If the above $\mathcal{D}_s$ is adopted, the rule of parsing in Phase 1 can be described as follows.

Read Seq until bit "0" appears or the run-length of bit "1" becomes t, where t is the maximal run-length of "1."

The above example corresponds to the case of $t = 14$.

The EWM with the above primitive dictionary $\mathcal{D}_s = \{10, 110, \cdots, \underbrace{1\cdots1}_{t-1}0, \underbrace{1\cdots1}_{t}\}$, which gives the dictionary $\mathcal{D} = \{2^i - 1\}_{i=1}^{t}$, can be simplified as follows. The "Run-length method" is named from its similarity to the "Run-length method" in data compression theory.

Run-length method

Input and Output are the same as the EWM.

Step 1 Determine the maximal run-length t appropriately from L and $w(e)$.

Step 2 Make the shortest addition chain for $\mathcal{D} = \{2^i - 1\}_{i=1}^{t}$. In this case, it is given by $\langle 1, \underline{2}, 3, \cdots, 2^i - 1, \underline{2^{i+1}-2}, 2^{i+1}-1, \cdots, \underline{2^t - 2}, 2^t - 1\rangle$. Note that the underlines represent the numbers not included in $\mathcal{D}$. Simply speaking, the shortest chain for $\mathcal{D}$ is obtained by repeating, $t-1$ times, the operations of doubling and adding 1.

Step 3 [Main Procedure]

Phase 1 Read Seq until bit "0" appears or the run-length of "1" becomes t. Let l be the run-length of "1." First apply the doubling rule l times. Next, apply the star chain rule $b_i = b_{i-1} + (2^l - 1)$. Remove the l bits from the prefix of Seq.

Phase 2 Read Seq until bit "1" appears. Let l_0 be the run-length of "0." Apply the doubling rule l_0 times. Remove the l_0 bits from the prefix of Seq and return to Phase 1.

Step 4 Concatenate the addition chains obtained in Steps 2 and 3.

As a simple example, we treat the case of $e = 445$ and $Seq = 110111101$. In Step 1, we set $t = 3$. In Step 2, the addition chain $\langle 1, \underline{2}, 3, \underline{6}, 7\rangle$ is obtained for $\mathcal{D} = \{1\ (= 2^1 - 1), 3\ (= 2^2 - 1), 7\ (= 2^3 - 1)\}$. In Step 3, the binary sequence Seq is parsed as follows. 11/0//111//1/0//1//, where "/" means the parsing of Phase 1 and "//" stands for the end of Phase 2. These parsed sequences lead to the addition chain, $\langle 3, 6, 12, 24, 48,_{(+7)} 55, 110,_{(+1)} 111, 222, 444,_{(+1)} 445\rangle$, where "," and "$,_{(+a)}$" mean that the doubling and star chain rules are applied, respectively. Hence, the final addition chain for 445 becomes

$$\langle 1, 2, 3, 6, 7, 12, 24, 48, 55, 110, 111, 222, 444, 445\rangle$$

in Step 4, and the addition chain length is 13. Hence, x^{445} can be calculated by 13 multiplications if we use the Run-length method with $t = 3$.

3.2 Average Chain Length of the Run-Length Method

In this subsection, we evaluate the average chain length attained by the Run-length method. We showed in [6] that the average chain length obtained by the "EWM" is given by

$$\left(L - \sum_{i=0}^{K} |Seq_i'| P_i\right) + \frac{L}{\sum_{i=0}^{K} |Seq_i| P_i + \frac{p}{1-p}} + (\text{chain length for } \mathcal{D}), \qquad (6)$$

where P_i is the probability of Seq_i in Phase 1 of Step 3. The first and second terms are the average numbers of the doubling and star chain rules applied in Step 3, respectively. The third term is the chain length for the dictionary $\mathcal{D}$ in Step 2.

By letting $K = t-1, Seq_i = \underbrace{1\cdots1}_{i+1}0,\ P_i = p \cdot q^i\ (0 \le i < t-1), Seq_{t-1} = \underbrace{1\cdots1}_{t},\ P_{t-1} = q^{t-1}$, and the third term $= 2(t-1)$, we obtain the following theorem.

Theorem 1: The average chain length for a positive integer e obtained by the Run-length method is given as

$$\left(L - \frac{1-q^t}{p}\right) + \frac{L}{\frac{1-q^{t-1}}{p} + \frac{1}{q}} + 2(t-1), \qquad (7)$$

where $L = \lfloor \log e \rfloor + 1$, $q = w(e)/L$, $p = 1-q$, and t is a parameter to be optimized.

When $L = 512, p = 0.1$, and $t = 14$, the above value becomes 590.0 while the chain length obtained by the Window method ($\kappa = 6$) is 621.9.

3.3 Hybrid Method

The optimal primitive dictionary obtained by the Tunstall-like algorithm, for instance when $L = 512$ and $p = 0.15$, becomes

$$\mathcal{D}_s = \{100, 1010, 1100, 10110, 10111, 11010, 11011, 11100, 11101, 11110, \underbrace{1\cdots1}_{5}0, \ldots, \underbrace{1\cdots1}_{14}0, \underbrace{1\cdots1}_{15}\}. \qquad (8)$$

The average chain length attained by the above primitive dictionary is only 602.9 while the chain length obtained by the Window method ($\kappa = 6$) is 621.0. However, the EWM with this primitive dictionary has two demerits compared with the Window method. One is Step 3, especially Phase 1 (a), which is more complicated than that in the Window method. The reason is that the EWM needs to use a tree structure in order to quickly find Seq_j since it is a little cumbersome to find Seq_j from many dictionary sequences. The other is the difficulty of obtaining the shortest addition chain for the dictionary in Step 2. In order to remove these defects, we introduce a simple primitive dictionary $\widetilde{\mathcal{D}_s}$, which is similar to the above $\mathcal{D}_s$.

$$\widetilde{\mathcal{D}_s} = \{1\{s_1 s_2 \cdots s_{\kappa-1}\}^*, \underbrace{1\cdots1}_{\kappa}0, \underbrace{1\cdots1}_{\kappa+1}0, \ldots, \underbrace{1\cdots1}_{t-1}0, \underbrace{1\cdots1}_{t}\}, \qquad (9)$$

where $\{s_1 \cdots s_{\kappa-1}\}^*$ are all $(2^{\kappa-1} - 1)$ sequences with length $\kappa - 1$ excluding $\underbrace{1\cdots1}_{\kappa-1}$.

Note 10: When $\kappa = 5, t = 15$, $\widetilde{\mathcal{D}_s}$ of (9) becomes "similar" to $\mathcal{D}_s$ given by (8).

Note 11: The primitive dictionary $\widetilde{\mathcal{D}_s}$ and 0^* can uniquely parse any sequences.

If $\widetilde{\mathcal{D}_s}$ is used instead of $\mathcal{D}_s$, the EWM's defects described above are removed in the following way. Step 3 can be simplified because the parsing rule with $\widetilde{\mathcal{D}_s}$ can be described as follows.

> Read κ bits from Seq. If all κ bits are "1," read Seq until "0" appears or the run-length of "1" becomes $t - \kappa$ and parse Seq there.

Note that this parsing rule can be easily implemented without a tree structure.

Furthermore, the shortest optimal addition chain for the dictionary is easily obtained. Note that the dictionary $\widetilde{\mathcal{D}}$ for $\widetilde{\mathcal{D}_s}$ is given by

$$\{1, 3, 5, \ldots, 2^\kappa - 3, 2^\kappa - 1, 2^{\kappa+1} - 1, 2^{\kappa+2} - 1, \ldots 2^t - 1\},$$

or, in short, $\widetilde{\mathcal{D}} = \{\{2i - 1\}_{i=1}^{2^{\kappa-1}}, \{2^{\kappa+i} - 1\}_{i=1}^{t-\kappa}\}$. The optimal addition chain for $\widetilde{\mathcal{D}}$ is easily obtained as $\langle 1, \underline{2}, 3, 5, \ldots, 2^\kappa - 3, 2^\kappa - 1, \underline{2^{\kappa+1} - 2}, 2^{\kappa+1} - 1, \ldots, \underline{2^t - 2}, 2^t - 1\rangle$, where the underlined numbers are not included in $\widetilde{\mathcal{D}}$. This chain length is $2^{\kappa-1} + 2(t - \kappa)$.

The above dictionary is equivalent to the dictionary of the Window method when $\kappa = t$. Furthermore, it is equivalent to the dictionary of the Run-length method when $\kappa = 2$. Hence, the following method is a hybrid of the Window method and the Run-length method.

Hybrid method

Input and Output are the same as the EWM.

Step 1 Determine the parameters κ and t appropriately from L and $w(e)$.

Step 2 Make the shortest addition chain for $\mathcal{D} = \{\{2i - 1\}_{i=1}^{2^{\kappa-1}}, \{2^{\kappa+i} - 1\}_{i=1}^{\kappa-t}\}$. In this case, it is given by $\langle 1, 2, 3, 5, \cdots, 2^\kappa - 3, 2^\kappa - 1, 2^{\kappa+1} - 2, \cdots, 2^t - 2, 2^t - 1\rangle$.

Step 3 [Main Procedure]

Phase 1 Read κ bits from Seq. If bit "0" is included in the κ bits, execute (a). Otherwise, execute (b).

(a) Assume that the κ bits be $1s_1 \cdots s_{\kappa-l-1}\underbrace{0 \cdots 0}_{l}$, where $s_{\kappa-l-1} = 1$ and $(1s_1 \cdots s_{\kappa-l-1})_{10} = a$. Then, apply the doubling rule $\kappa - l$ times, the star chain rule $b_i = b_{i-1} + a$ once, and the doubling rule l times in this order. Remove the κ bits from the prefix of Seq.

(b) Read Seq until bit "0" appears or the run-length of bit "1" becomes $t - \kappa$. Let l be the total run-length of "1." Then, apply the doubling rule l times, and the star chain rule $b_i = b_{i-1} + (2^l - 1)$. Remove the l bits from the prefix of Seq.

Phase 2 Read Seq until bit "1" appears. Let l_0 be the run-length of "0." Then, apply the doubling rule l_0 times. Remove the l_0 bits from the prefix of Seq and return to Phase 1.

Step 4 Concatenate the addition chains obtained in Steps 2 and 3.

Note that (a) and (b) correspond to Phase 1 of Step 3 in the Window and Run-length methods, respectively. As a simple example, consider $e = 75064310$, $Seq = 100011110010110001111110110$. In Step 1, we set $\kappa = 3$ and $t = 5$. In Step 2, the addition chain $\langle 1, \underline{2}, 3, 5, 7, \underline{14}, 15, \underline{30}, 31\rangle$ is derived for $\mathcal{D} = \{1, 3, 5, 7, 15, 31\}$. In Step 3, Seq is parsed as follows.

100/0//1111/00//101//100/0//11111//101//10//,

where "/" and "//" are the same as the example treated in Sect. 3.1. These parsed sequences lead to the following chain, $\langle 1, 2, 4, 8, 16, 32, 64, 128,_{(+15)} 143, 286, 572, 1144, 2288, 4576,_{(+5)} 4581, 9162,_{(+1)} 9163, 18326, 36652, 73304, 146608, 293216, 586432, 1172864, 2345728,_{(+31)} 2345759, 4691518, 9383036, 18766072,_{(+5)} 18766077, 37532154,_{(+1)} 37532155, 75064310\ (= e)\rangle$. In Step 4, the whole addition chain is obtained by concatenating the chains obtained in Steps 2 and 3. Hence, the total chain length for e becomes 8 (Step 2) +32 (Step 3) = 40 in this case.

Note 12: When $\kappa = 1$ and $t = 1$, Hybrid method is equivalent to the binary method.

3.4 Average Chain Length of the Hybrid Method

The average chain length obtained by the Hybrid method can also be derived from Eq. (6). Let $K = 2^{\kappa-1} + t - \kappa - 1$. In case of $Seq_i = 1\{s_1 \cdots s_{\kappa-1}\}^*$, $0 \leq i \leq 2^{\kappa-1} - 2$, where $\{s_1 \cdots s_{\kappa-1}\}^*$ represents all $(2^{\kappa-1} - 1)$ sequences excluding $\underbrace{1 \cdots 1}_{\kappa-1}$, P_i in Eq. (6) is given by $P_i = p^{w_1} \cdot q^{w_2}$ where w_1 and w_2 are the numbers of "0" and "1" included in $s_1 \cdots s_{\kappa-1}$, respectively. Furthermore, $P_i = p \cdot q^{i - 2^{\kappa-1} + \kappa}$ if $Seq_i = \underbrace{1 \cdots 1}_{\kappa \cdots t-1}0$, $2^{\kappa-1} - 1 \leq i \leq 2^{\kappa-1} + t - \kappa - 2\ (= K - 1)$, and $P_K = q^{t-1}$ if $Seq_K = \underbrace{1 \cdots 1}_{t}$. Finally, since the third term is equal to $2^{\kappa-1} + 2(t - \kappa)$, we obtain the following theorem.

Theorem 2: The average chain length for positive integer e obtained by the Hybrid method is given by

$$\left(L - (\kappa - \frac{p - p^\kappa}{1 - p} + \frac{q^\kappa - q^t}{p})\right)$$

Table 1 Performance of four methods.

p	$w(e)$	EWM		Window		Run-length		Hybrid	
		length	$\#\mathcal{D}$	length	κ	length	t	length	(κ, t)
0.95	25	536.6	1	536.6	1	536.6	1	536.6	$(1,1)$
0.90	51	554.2	6	557.4	3	559.5	2	557.4	$(3,3)$
0.85	77	567.4	7	571.2	4	579.6	2	571.2	$(4,4)$
0.80	102	576.8	10	582.0	4	597.3	3	582.0	$(4,4)$
0.75	128	585.2	14	589.1	5	612.2	3	589.1	$(5,5)$
0.70	154	591.7	15	594.6	5	625.0	4	594.6	$(5,5)$
0.65	179	598.0	21	599.2	5	634.7	4	599.2	$(5,5)$
0.60	205	602.8	21	603.1	5	642.5	4	603.1	$(5,5)$
0.55	230	606.0	17	606.4	5	647.3	5	606.4	$(5,5)$
0.50	256	609.2	17	609.3	5	650.1	5	609.3	$(5,5)$
0.45	282	613.1	22	611.8	5	650.2	6	611.8	$(5,5)$
0.40	307	613.6	22	614.0	5	648.0	7	613.9	$(5,6)$
0.35	333	615.1	21	616.0	5	643.5	8	615.0	$(5,6)$
0.30	358	614.3	19	617.7	5	636.8	9	614.9	$(5,7)$
0.25	384	611.7	19	619.2	6	628.0	10	613.2	$(5,8)$
0.20	410	608.6	22	620.2	6	617.0	11	609.4	$(5,10)$
0.15	435	602.9	21	621.1	6	604.4	12	601.2	$(4,12)$
0.10	461	590.0	14	621.9	6	590.0	14	589.2	$(3,14)$
0.05	486	574.2	17	622.6	6	574.2	17	573.9	$(3,17)$

$$+\frac{L}{\kappa+\frac{q^{\kappa-1}-q^{t-1}}{p}+\frac{p}{1-p}}+2^{\kappa-1}+2(t-\kappa), \quad (10)$$

where $L = \lfloor \log e \rfloor + 1$, $q = w(e)/L$, and $p = 1 - q$. κ and t are parameters to be optimized.

When $L = 512, p = 0.15, \kappa = 5,$ and $t = 15$, the above value is 605.0. This value is a little larger than the chain length 602.9, which is attained by the EWM with the primitive dictionary given by (8). However, the parsing in the Hybrid method is much easier than the EWM as described in Sect. 3.3. In addition, the case of $\kappa = 5$ and $t = 15$ is not optimal for the Hybrid method. In fact, the optimal parameters are $\kappa = 4$ and $t = 12$ and the minimum value is 601.2. See Table 1.

Although the optimal parameters (κ, t) must be chosen in Step 1, they can easily be determined by evaluating Eq. (10). It is worth noting that Eq. (10) is equivalent to Eq. (5), which is the average chain length by the Window method, for $\kappa = t$, and Eq. (10) is equal to Eq. (7), which is the average chain length by the Run-length method, for $\kappa = 2$.

4. Numerical Results

In this section, we show some numerical results for $L = 512$. Table 1 and Fig. 1 represent the minimum values of average addition chain length obtained by the EWM with the Tunstall-like algorithm, the Window method, the Run-length method, and the Hybrid method, which are calculated from Eqs. (6), (5), (7), and (10), respectively. For each method, the optimal size of $\mathcal{D}$ and parameters κ, t, or (κ, t) that minimize the theoretical average chain length are used for each p.

In order to confirm the validity of the above theoretical results, we have some simulation results, which are obtained in the following way. Random sequences are generated with a given $p = \Pr\{s_l = 0\}$ and addition chains are constructed by the above four methods. The optimal parameters are searched for each method to attain the minimum value of the average chain length.

These simulation values coincide with Table 1 very well except that each values are larger than Table 1 by about 0.4. This difference is caused from the fact that the last parsing is stopped by the end of a sequence.

From Table 1 and Fig. 1, the compared three methods have the following properties.

1. The Hybrid method can generate an addition chain whose length is almost equal to that of the EWM with the Tunstall-like algorithm for any p.
2. In $0 < p < 0.2$, the average chain length obtained by the Run-length method is almost equal to the EWM and much smaller than the Window method.
3. In $0.2 \leq p \leq 0.4$, the Hybrid method can generate a shorter addition chain than both the Window method and the Run-length method.
4. In $0.4 < p < 1$, the performance of the Hybrid method is almost equal to the Window method.

Note that when $p = 0.45$, 0.35, 0.15, 0.10, and 0.05, the Hybrid method generates a shorter chain than the EWM with the Tunstall-like algorithm. This is caused from the fact that the EWM includes more extra numbers in the addition chain for $\mathcal{D}$ than the Hybrid method in the above cases.

Next, we compare the Window method and the Hybrid method for the case of a large Hamming weight ($p = 0.05$) and larger exponents ($L = 1024, 2048,$ and 4096). Table 2 shows the average chain length obtained by these two methods. In all cases, the Hybrid method can generate a shorter addition chain than the Window method by 7–8%.

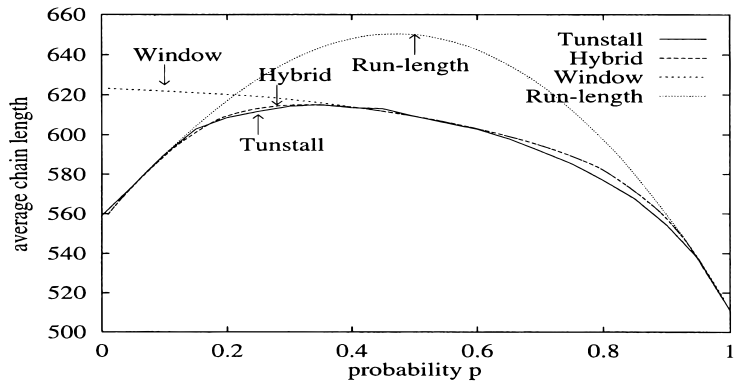


Fig. 1 Comparison of four methods.

Table 2 Performance of the Window method and the Hybrid method when $p = 0.05$.

L	Window method		Hybrid method		Speedup
	length	κ	length	(κ, t)	(%)
512	622.6	6	573.9	$(3, 17)$	7.8
1024	1219.2	6	1124.0	$(3, 22)$	7.8
2048	2395.4	7	2213.7	$(3, 30)$	7.6
4096	4724.7	8	4379.0	$(3, 39)$	7.3

From the above observation and our theoretical analysis in Sect. 3.3, it follows that the Hybrid method is more efficient and practical compared to the Window method, the EWM with the Tunstall-like algorithm, and the Run-length method.

Finally, we compare the Hybrid method with two other methods: the Bos-Coster method [1] and the Yacobi method [11]. The Bos-Coster method is similar to the Window method. However, their method uses a large window (for example, $\kappa = 10$) and must make a short addition chain for a large dictionary $\mathcal{D}$ with many elements, which is a hard task. Hence, this method is not practical and not suited for implementation. The Yacobi method uses an adaptive dictionary such as the EWM. However, since his method is based on the LZ78 code in data compression theory and the primitive dictionary $\mathcal{D}_s$ is created unboundedly by the incremental parsing, the chain length cannot become short for practical size exponents, e.g. $L = 512, 1024,$ and 2048.

These results lead us to the conclusion that the Hybrid method is more efficient than the known methods.

5. Conclusion

In this paper, we have proposed the Run-length method and the Hybrid method. We theoretically analyzed these two methods and showed that the Hybrid method is more efficient and practical, especially in the case of the large Hamming weight, than known methods. Roughly speaking, the Hybrid method can attain 8% reduction, in the best case, in the average addition chain length compared with the Window method.

References

[1] J. Bos and M. Coster, "Addition chain heuristics," Advances in Cryptology – CRYPTO'89, vol.LNCS 435, pp.400–407, 1989.

[2] P. Downey, B. Leong, and R. Sthi, "Computing sequences with addition chains," SIAM J. Computing, vol.10, no.3, pp.638–646, 1981.

[3] D.E. Knuth, Seminumerical algorithm (arithmetic) The art of Computer Programming vol.2, Addison Wesley, 1981.

[4] N. Koblitz, "Elliptic curve cryptosystems," Mathematics of Computation, vol.48, no.177, pp.203–209, 1987.

[5] N. Kunihiro and H. Yamamoto, "Optimal addition chain classified by Hamming weight," IEICE Technical Report, ISEC96-74, 1997.

[6] N. Kunihiro and H. Yamamoto, "Window and extended window methods for addition chain and addition-subtraction chain," IEICE Trans. Fundamentals, vol.E81-A, no.1, pp.72–81, Jan. 1998.

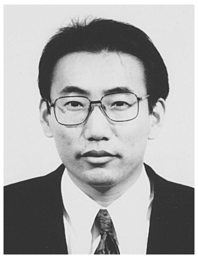
[7] D.C. Lou and C.C. Chang, "An adaptive exponentiation method," The Journal of Systems and Software, vol.42, pp.59–69, 1998.

[8] V.S. Miller, "Use of elliptic curves in cryptography," Proc. of Crypto'85, vol.LNCS 218, pp.417–426, Springer-Verlag, 1985.

[9] R.L. Rivest, A. Shamir, and L. Adleman, "A method for obtaining digital signature and public key cryptosystems," Comm. of ACM, vol.21, no.2, pp.120–126, 1978.

[10] A. Schönhage, "A lower bound for the length of addition chains," Theoretical Computer Science, vol.1, pp.1–12, 1975.

[11] Y. Yacobi, "Exponentiating faster with addition chains," Advances in Cryptology – EUROCRYPT'90, vol.LNCS 473, pp.222–229, 1990.



Noboru Kunihiro was born in Aichi, Japan on June 26, 1971. He received the B.E. and M.E. in mathematical engineering and information physics from the University of Tokyo, Tokyo, Japan in 1994 and 1996, respectively. Since 1996 he has been with NTT Communication Science Laboratories and engaged in research for cryptography and information security. His research interest includes elliptic curve theory and cryptography. He was awarded the SCIS'97 paper prize.

Hirosuke Yamamoto was born in Wakayama, Japan, on November 15, 1952. He received the B.E. degree from Shizuoka University, Shizuoka, Japan, in 1975 and the M.E. and Dr.E. degrees from the University of Tokyo, Tokyo, Japan, in 1977 and 1980, respectively, all in electrical engineering. In 1980 he joined Tokushima University, Tokushima, Japan. He was an Associate Professor at Tokushima University, University of Electro-Communications, and University of Tokyo, during 1983–1987, 1987–1993, and 1993–1999, respectively. Since March 1999, he is a professor in the Department of Mathematical Engineering and Information Physics, Graduate School of Engineering, University of Tokyo. In 1989–90, he was a Visiting Scholar at the Information Systems Laboratory, Stanford University. His research interests are in Shannon theory, coding theory, cryptology, and communication theory.