IEICE TRANS. FUNDAMENTALS, VOL.E98–A, NO.6 JUNE 2015

LETTER *Special Section on Discrete Mathematics and Its Applications*

Improving Width-3 Joint Sparse Form to Attain Asymptotically Optimal Complexity on Average Case[*]

Hiroshi IMAI[†] ***and*** **Vorapong SUPPAKITPAISARN**[††,†††a)], ***Members***

SUMMARY In this paper, we improve a width-3 joint sparse form proposed by Okeya, Katoh, and Nogami. After the improvement, the representation can attain an asymptotically optimal complexity found in our previous work. Although claimed as optimal by the authors, the average computation time of multi-scalar multiplication obtained by the representation is $563/1574n + o(n) \approx 0.3577n + o(n)$. That number is larger than the optimal complexity $281/786n + o(n) \approx 0.3575n + o(n)$ found in our previous work. To optimize the width-3 joint sparse form, we add more cases to the representation. After the addition, we can show that the complexity is updated to $281/786n + o(n) \approx 0.3575n + o(n)$, which implies that the modified representation is asymptotically optimal. Compared to our optimal algorithm in the previous work, the modified width-3 joint sparse form uses less dynamic memory, but it consumes more static memory.

key words: *analysis of algorithms, number representation, elliptic curve cryptography, multi-scalar multiplication, width-3 joint sparse form*

1. Introduction and Notations

In this work, we analyze the average efficiency of multi-scalar multiplication, when it is implemented using width-3 joint sparse form. The operation is the bottleneck operation of many elliptic curve cryptographic protocols, including elliptic curve digital signature algorithm (ECDSA) [2]. Our goal is to compute $S = r_1 P_1 + r_2 P_2$ where r_1, r_2 are natural numbers and P_1, P_2 are points on the elliptic curve.

Denote $n := \lfloor \log_2 \max\{r_1, r_2\} \rfloor$. Let D_S be a finite set. The *representations* of r_1, r_2 using D_S can be defined as follows.

Definition 1 (Representation Using D_S)**.** *Let $R_1 := \langle r_{1,t} \rangle_{t=0}^{n-1}$ and $R_2 := \langle r_{2,t} \rangle_{t=1}^{n-1}$, when $r_{1,t}, r_{2,t} \in D_S$. If $r_1 = \sum_{t=0}^{n} r_{1,t} 2^t$ and $r_2 = \sum_{t=0}^{n} r_{2,t} 2^t$, then (R_1, R_2) is the **representation** of (r_1, r_2) using D_S.*

If (R_1, R_2) is the representation of (r_1, r_2) using $\{0, 1\}$, we call (R_1, R_2) as the binary representation of (r_1, r_2). If $r_1 = 12$ and $r_2 = 21$, the only binary representation of (r_1, r_2) is $\left(R_1^{(b)}, R_2^{(b)}\right)$, when $R_1^{(b)} := \langle 0, 1, 1, 0, 0 \rangle$ and $R_2^{(b)} = \langle 1, 0, 1, 0, 1 \rangle$ respectively. One of the representations of $(12, 21)$ using $D_S = \{0, \pm 1\}$ is $\left(R_1^{(c)}, R_2^{(c)}\right)$, when $R_1^{(c)} = \langle 1, 0, \bar{1}, 0, 0 \rangle$, $R_1^{(c)} = \langle 1, 0, 1, 0, 1 \rangle$, and -1 is denoted as $\bar{1}$.

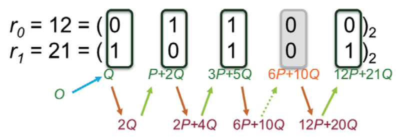

Fig. 1 The computation of multi-scalar multiplication using Shamir's trick with binary representation.

```
Input: natural numbers $r_1, r_2$,
representations of $r_1, r_2$ using $D_S$, $\langle r_{1,t} \rangle_{t=0}^{n-1}, \langle r_{2,t} \rangle_{t=0}^{n-1}$,
elliptic points $P_1, P_2$, and $d_1 P_1 + d_2 P_2$ for all $d_1, d_2 \in D_S$
Output: elliptic points $S = r_1 P_1 + r_2 P_2$
1 $S \leftarrow r_{1,n-1} P_1 + r_{2,n-1} P_2$
2 for $t \leftarrow n-2$ to $0$ do
3     $S \leftarrow 2S$
4     if $(r_{1,t}, r_{2,t}) \neq (0,0)$ then
5         $S \leftarrow S + (r_{1,t} P_1 + r_{2,t} P_2)$
6 end
```

Algorithm 1: A multi-scalar multiplication algorithm using Shamir's trick

In Algorithm 1 and Fig. 1, we show a method to compute $r_1 P_1 + r_2 P_2$ using a representation of (r_1, r_2) and Shamir's trick. The bottleneck of Algorithm 1 are point doubles in Line 3 and point additions in Line 5. While the number of point doubles is always equal to n, the number of point additions equals the number of non-zero $(r_{1,t}, r_{2,t})$ minus one. The number of non-zero $(r_{1,t}, r_{2,t})$ can be formally defined as a *joint hamming weight* in the following definition.

Definition 2 (Joint Hamming Weight)**.** *The **joint Hamming weight** of $(R_1, R_2) = \left(\langle r_{1,t} \rangle_{t=0}^{n-1}, \langle r_{2,t} \rangle_{t=0}^{n-1}\right)$, denoted as $|(R_1, R_2)|$, is equal to $\sum_{t=0}^{n-1} w_t$, when $w_t = 0$ if $(r_{1,t}, r_{2,t}) = (0, 0)$ and $w_t = 1$ otherwise.*

Recall the notation $R_1^{(b)}, R_2^{(b)}, R_1^{(c)}, R_2^{(c)}$. From the previous definition, we get $\left|\left(R_1^{(b)}, R_2^{(b)}\right)\right| = 4$ and $\left|\left(R_1^{(c)}, R_2^{(c)}\right)\right| = 3$. By that, the multi-scalar multiplication using $R_1^{(c)}, R_2^{(c)}$ is faster than the calculation using $R_1^{(b)}, R_2^{(b)}$. If $D_S \supsetneq \{0, 1\}$, we can represent some natural numbers by more than one

Manuscript received September 17, 2014.
Manuscript revised November 27, 2014.

[†]The author is with the Graduate School of Information Science and Technology, The University of Tokyo, Tokyo, 113-0033 Japan.
[††]The author is with the Global Research Center for Big Data Mathematics, National Institute of Informatics, Tokyo, 101-8430 Japan.
[†††]The author is also with ERATO Kawarabayashi Large Graph Project, JST, Tokyo, 101-8430 Japan.

[*]The abstract of this paper is appeared in [1].

a) E-mail: vorapong@nii.ac.jp

DOI: 10.1587/transfun.E98.A.1216

ways, and we want to find a way to minimize the computation time. That leads us to the following definition.

Definition 3 (Minimum Weight Representation)**.** *Let $\left(R_1^*, R_2^*\right)$ be a representation of (r_1, r_2) using D_S. If there does not exist a representation of (r_1, r_2) using D_S, (R_1, R_2), such that $|(R_1, R_2)| < \left|(R_1^*, R_2^*)\right|$, then (R_1^*, R_2^*) is* ***the minimum weight representation*** *of (r_1, r_2).*

We denote the joint Hamming weight of the minimum weight representation of r_1, r_2, $|(R_1^*, R_2^*)|$, as $\mathcal{M}_{D_S}(r_1, r_2)$.

Given r_1, r_2, D_S, the minimum weight representation (R_1^*, R_2^*) can be computed using a dynamic programming algorithm proposed in [3]. By the same algorithm, we can also get the minimum joint weight $\mathcal{M}_{D_S}(r_1, r_2)$ in polynomial time.

Next, we will analyze the average performance of multi-scalar multiplication using the minimum weight representation. The metric that we use is defined in the following definition.

Definition 4 (Average joint Hamming weight)**.** *The* ***average joint Hamming weight*** *of digit set D_S, $\mathcal{A}_{D_S}(n)$, can be defined as*

$$\mathcal{A}_{D_S}(n) := \sum_{r_1=0}^{2^n-1} \sum_{r_2=0}^{2^n-1} \frac{\mathcal{M}_{D_S}(r_1, r_2)}{2^{2n}}.$$

It can be easily seen that $\mathcal{A}_{\{0,1\}}(n) \in 0.75n + o(n)$. Also, Solinas shows that $\mathcal{A}_{\{0,\pm 1\}}(n) \in 0.5n + o(n)$ in [4]. One of the open problems in the paper is the value of $\mathcal{A}_{\{0,\pm 1,\pm 3\}}(n)$. While our previous work shows that $\mathcal{A}_{\{0,\pm 1,\pm 3\}}(n) \in 281/786n + o(n) \approx 0.3575n + o(n)$ in [5], Okeya, Katoh, and Nogami independently report that $\mathcal{A}_{\{0,\pm 1,\pm 3\}}(n) \in 563/1574n + o(n) \approx 0.3577n + o(n)$ in [6], which is slightly larger than our value.

1.1 Contributions in This Letter

In [6], the authors claim that a representation is a minimum weight representation, if it satisfies a set of condition called width-3 joint sparse form. Then, they devise an algorithm which finds a representation that satisfies the conditions, and use Markov chain to prove that the average joint Hamming weight is $0.3577n + o(n)$. After we briefly explain those ideas, we will show that there are representations in the width-3 joint sparse form that are not minimum weight representations in Sect. 2.

To handle those missing cases, we will modify conditions for width-3 joint sparse form in Sect. 3. Then, we modify their algorithms to handle the conditions modified, and add more states to their Markov chain. After those modification, we can show that the average joint Hamming weight of the algorithm is updated to $281/786n + o(n) \approx 0.3575n + o(n)$. That number matches our result in [5].

By the modification, we get two asymtotically optimal algorithms for representations using D_S. We will compare the performance of those two algorithms in Sect. 4. While our algorithm in [5] consumes more dynamic memory than the algorithm modified in this paper, ours is faster, simpler, and use less static memory than the other. Because both optimal algorithms have different advantages, we can choose one that fits our computational environment more.

It is very important to note that we do not show the optimality of the modified representation by correcting the proof in [6]. Instead, we found an incorrect statement in the early state of the proof. Then, we try to correct that statement by adding more conditions to the representation, and use Markov chain to prove the average joint Hamming weight. We can claim that the modified representation is asymtotically optimal, because the average weight obtained from the Markov chain matches the result in our previous work, which is proved to be optimal.

Although the result in those two papers are very close, knowing the exact value of $\mathcal{A}_{\{0,\pm 1,\pm 3\}}(n)$ can help us predicting more general results easier. Currently, we are aiming to find an explicit closed form between h and $\mathcal{A}_{\{0,\pm 1,\pm 3 \ldots,\pm(2h+1)\}}(n)$. A small discrepancy can lead us to an incorrect formula and a large discrepancy for large h. Another reason is the fact that we want to verify our general results for $h \leq 5$ in [3]. By the discrepancy with the result in [6], we could not be sure that the results were correct. As we can improve the result in [6] to match our result in this paper, we can gain more confidence on our proof in [3]. As a result, we can also gain more confidence on our general results for $h \leq 5$.

2. Missing Cases in [6]

In this section, we will briefly explain the results in [6], and point out the case missing in their proof. Let $(R_1, R_2) = \left(\langle r_{1,t}\rangle_{t=0}^{n-1}, \langle r_{2,t}\rangle_{t=0}^{n-1}\right)$ be a representation of (r_1, r_2) using $\{0, \pm 1, \pm 3\}$. By those notations, we get the following definitions.

Notations: $i \in \{1, 2\}$, $0 \leq t < n$, and $0 \leq t_2 < t_1 < n$. $r_{i,[t_1,t_2]} := \sum_{t=t_2}^{t_1} r_{i,t} 2^{t-t_2}$; $R_{i,[t_1,t_2]} := \langle r_{i,t}\rangle_{t=t_1}^{t_2}$;
$u_t := (r_{1,t}, r_{2,t})$; $U_{[t_1,t_2]} := \left(R_{1,[t_1,t_2]}, R_{2,[t_1,t_2]}\right)$; $O = (0, 0)$; $\bar{i} = 2$ when $i = 1$, and $\bar{i} = 1$ when $i = 2$.

$C_1(i,t)$: $r_{\bar{i},t} = 0$, $u_{t+1} = u_{t+2} = 0$.
$C_2(i,t)$: $r_{\bar{i},t} = 0$, $u_{t+1} = O$, $r_{1,t+2} r_{2,t+2} \neq 0$.
$C_3(i,t)$: $r_{\bar{i},t} = 0$, $r_{1,t+1} r_{2,t+1} \neq 0$. If $r_{i,[t+3,t]} \in \{\pm 7, \pm 9\}$, then $u_{t+3} = O$ and $r_{0,j+4} \equiv r_{1,j+4} \pmod 2$.
$C_4(i,t)$: $r_{\bar{i},t} \neq 0$, $u_{t+1} = u_{t+2} = O$, at least one of u_{t+3} and u_{t+4} are equal to O.
$C_5(i,t)$: $r_{\bar{i},t} \neq 0$, $u_{t+1} = O$, $r_{1,t+2} r_{2,t+2} \neq 0$, and $C_4(i, t+2)$ is satisfied. Furthermore, there are no representation (R_1', R_2') of (r_1, r_2) such that $(R'_{1,[n-1,t]}, R'_{2,[n-1,t]}) \neq (R_{1,[n-1,t]}, R_{2,[n-1,t]})$ and (R_1', R_2') satisfies $C_4(i,t)$.
$C_6(i,t)$: $r_{\bar{i},t} \neq 0$, $u_{t+1} = u_{t+2} = O$. There exists $i' \in \{1, 2\}$ such that $r_{i',[t+6,t+3]} \in \{\pm 7, \pm 9\}$ Furthermore, there are no representation (R_1', R_2') of (r_1, r_2) such that $(R'_{1,[n-1,t]}, R'_{2,[n-1,t]}) \neq (R_{1,[n-1,t]}, R_{2,[n-1,t]})$ and (R_1', R_2') satisfies $C_5(i,t)$.
$C_7(i,t)$: $r_{\bar{i},t} \neq 0$, $u_{t+1} = O$, $r_{0,t+2} r_{1,t+2} \neq 0$. One of the conditions $C_5(i, t+2), C_6(i, t+2), C_7(i, t+2)$ is satisfies. Furthermore, there are no representation (R_1', R_2') of (r_1, r_2) such that $(R'_{1,[n-1,t]}, R'_{2,[n-1,t]}) \neq (R_{1,[n-1,t]}, R_{2,[n-1,t]})$ and (R_1', R_2') satisfies $C_4(i,t)$, $C_5(i,t)$, and $C_7(i,t)$.

Fig. 2 Conditions for width-3 joint sparse form.

Definition 5 (Width-3 Joint Sparse Form)**.** *A representation (R_1, R_2) is a representation in* ***width-3 joint sparse form****, if it satisfies one of the conditions $C_1(i,t), \ldots, C_7(i,t)$ for any i, t such that $r_{i,t} \neq 0$.*

The result that the authors of [6] got from the previous definition can be concluded as follows:

Claim 1 (Appendix A of [6])**.** *For each $(r_1, r_2) \in \mathbb{Z}_+^2$, there is exactly one representation of (r_1, r_2) that is in width-3 joint sparse form.*

Claim 2 (Appendix B of [6])**.** *Let (R_1, R_2) and (R_1', R_2') be representations of (r_1, r_2). If (R_1, R_2) is in width-3 joint sparse form while (R_1', R_2') is not, then $|(R_1, R_2)| \leq |(R_1', R_2')|$.*

From the previous statements, the authors claim that (R_1, R_2) is a minimum weight representation of (r_1, r_2) using $\{0, \pm1, \pm3\}$, if it is in width-3 joint sparse form. The authors also propose Algorithm 2 that can find that minimum weight representation in linear time.

We found cases missing in the proof of Claim 1. Actually, the representations in width-3 joint sparse form is not unique. When $n = 8$ and $(r_1, r_2) = (10, 33)$, a minimum weight representation using $\{0, \pm1, \pm3\}$ is

$$(R_1^*, R_2^*) = (\langle 0,0,0,0,1,0,1,0\rangle, \langle 0,0,0,0,3,0,3,3\rangle).$$

However, the representation

$$(R_1', R_2') = (\langle 0,0,0,0,1,0,1,0\rangle, \langle 0,1,0,0,\bar{3},0,\bar{3},\bar{1}\rangle)$$

is also a representation in width-3 joint sparse form, if $\bar{1}, \bar{3}$ denote $-1, -3$.

We know that (R_1', R_2') is not a minimum weight representation, since $4 = |(R_1', R_2')| > |(R_1^*, R_2^*)| = 3$. Although the proof of Claim 2 might be correct, we cannot guarantee that the algorithm, which always find a representation in the form, will give us an optimal solution. That is because there are many representations in the form, and some of them are not a minimum weight representation.

Without our modification, Algorithm 2 unfortunately outputs the sub-optimal solution (R_1', R_2'), when its input is $(10, 33)$. Other than $(10, 33)$, we found infinitely many cases that Algorithm 2 cannot output an optimal solution, such as $(10 \cdot 2^h, 33 \cdot 2^h)$ for $h \geq 1$.

3. Modification

Although we show that there are cases missing in those conditions, our task is not only to modify those conditions. Recall that Okeya, Katoh, and Nogami proposes conditions for width-3 joint sparse form. Then, they propose an algorithm based on those conditions, and provide a Markov chain based on the algorithm. Because of that, we have to fix all of them. We will modify the conditions in Sect. 3.1, the algorithms in Sect. 3.2, and the Markov chain in Sect. 3.3.

3.1 Modification on Conditions

To modify the conditions, we replace the condition $C_3(i,t)$

$C_3'(i,t)$: $r_{\bar{i},t} = 0$, $r_{1,t+1} r_{2,t+1} \neq 0$. If $r_{i,[t+3,t]} \in \{\pm7, \pm9\}$, then $U_{t+3} = O$ and $r_{0,j+4} \equiv r_{1,j+4} \pmod 2$. **If** $r_{i,[t+6,t]} \in \{\pm33\}$ **and** $r_{\bar{i},[t+6,t+1]} \in \{\pm5, \pm7, \pm9, \pm11\}$**, then** $u_{t+2} = u_{t+4} = u_{t+5} = u_{t+6} = O$**. If** $r_{i,[t+6,t]} \in \{\pm31\}$ **and** $r_{\bar{i},[t+6,t+1]} \in \{\pm21, \pm23, \pm25, \pm27\}$**, then** $u_{t+2} = u_{t+4} = u_{t+5}$ **and** $r_{1,t+6}, r_{2,t+6} \in D_S \backslash \{0\}$**.**

Fig. 3 A modified condition for width-3 joint sparse form.

by the condition $C_3'(i,t)$ shown in Fig. 3. In the figure, our modifications are marked with bold letters. Obviously, the representation (R_1', R_2') in the previous section does not satisfy the conditions after that replacement.

3.2 Modification on Algorithm

In Algorithm 2, we show an algorithm by Okeya, Katoh, and Nogami, together with our modification marked with bold letters. The notation used in the algorithm can be found in Fig. 5. From that algorithm, we get the following claims.

Lemma 1. *If the input is the Booth recording of (r_1, r_2), the modified version of Algorithm 2 outputs a representation of (r_1, r_2) using $\{0, \pm1, \pm3\}$ in the modified width-3 joint sparse form.*

Proof. We know that the output of the algorithm is the representation of (r_1, r_2), since all operations in the algorithm, such as SW, ZF, or DI, do not change the value that $\left(\left\langle r_{1,t}^{(b)}\right\rangle_{t=0}^{n-1}, \left\langle r_{2,t}^{(b)}\right\rangle_{t=0}^{n-1}\right)$ represents.

Before our modification, it is proved in [6] that Algorithm 2 outputs representations in width-3 joint sparse form. Since it is obvious that our modifications in Lines 12-17 are corresponding to the modifications in $C_3'(i,t)$, we know that the modified algorithm outputs representations in the modified form. □

3.3 Modification on Markov Chain

In this subsection, we will prove the following theorem.

Theorem 1. *Let $\mathcal{M}_j(r_1, r_2)$ be a joint Hamming weight of a representation for (r_1, r_2) in the modified width-3 joint sparse form. We get*

$$\sum_{r_1=0}^{2^n-1} \sum_{r_2=0}^{2^n-1} \frac{\mathcal{M}_j(r_1, r_2)}{2^{2n}} = \frac{281}{786} n + o(n).$$

Proof. To prove this theorem, we will modify Markov chain proposed in [6]. The result of that modification is shown in Fig. 4. While the Markov chain in [6] contains five nodes, A, B, C, D, E, we add more nodes, B', C', E', in this modification. In [6], each node represents a value of the variable $STATE$ in the algorithm, but we use two nodes, B, B', to represents the case when $STATE = B$ in this proof. Similarly, both C, C' represents the case when $STATE = C$, and both E, E' represents the case when $STATE = E$. We will explain the reason why we need to split those three nodes

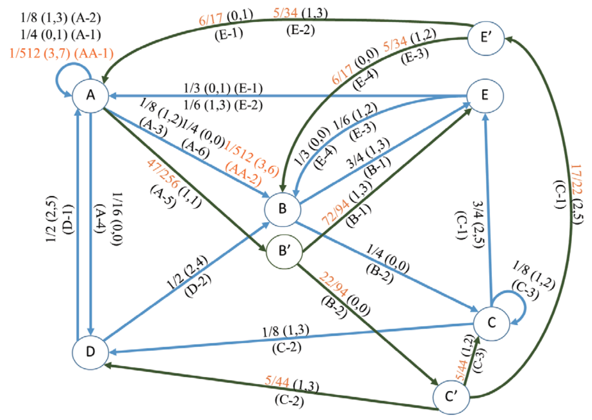

Fig. 4 Markov chain used for finding an average joint Hamming weight of Algorithm 2.

later in this proof. The label of each transition T represents the transition probability $p(T)$, the joint Hamming weight increased by that transition $w(T)$, the value of t after the transition minus the value before the transition $\ell(T)$, and the case in the algorithm corresponding to that transition.

We add the transition (AA-1) and (AA-2), which are the case added to improve the average joint Hamming weight in Algorithm 2. By that addition, some transition probabilities of the existing transition have to be modified. Besides, we found that the distribution of the input bits in the case (A-5) is not uniform after the distribution. We need to add nodes B', C', E' into the Markov chain to take care of those cases. We will give the detail of this non-uniformity, explain more detail on this Markov chain modification, and calculate the transition probabilities for each newly-added nodes in Appendix.

All probabilities in this Markov chain are confirmed to be correct by simulations done in the similar way as in [6]. The simulations are done on 1000 random 10^7-bit integers. For all of those integers, the difference between the transition probability found in the simulation and our calculation results is at most 10^{-6}.

The stationary distribution of the Markov chain is

$$\begin{aligned}\pi &= [\pi(A), \pi(B), \pi(B'), \pi(C), \pi(C'), \pi(D), \pi(E), \pi(E')]\\ &= \frac{1}{6745}\left[1792, 1792, 329, 522, 77, \frac{3975}{2}, \frac{119}{2}\right].\end{aligned}$$

Let $V := \{A, B, C, D, E\}$, $\mathcal{T}_v$ be a set of transition starting from $v \in V$, $w(v) = \sum_{T\in\mathcal{T}_v} p(T)\cdot w(T)$, and $\ell(v) = \sum_{T\in\mathcal{T}_v} p(T)\cdot \ell(T)$. Using the same argument as in [6], we know that the expected Hamming weight after n transitions is $W := \sum_{v\in V} w(v)\cdot\pi(v)\cdot n = \frac{4496}{6745}n$. The expected length processed after n transitions is $L := \sum_{v\in V} \ell(v)\cdot\pi(v)\cdot n = \frac{12576}{6745}n$. Then, the average joint Hamming weight is

$$\frac{W}{L}\cdot n + o(n) = \frac{281}{786}n + o(n).$$

□

Let $\left(\left\langle r^{(b)}_{1,t}\right\rangle^{n-1}_{t=0}, \left\langle r^{(b)}_{2,t}\right\rangle^{n-1}_{t=0}\right)$ be a binary representation of (r_1, r_2). A Booth recording of (r_1, r_2), $\left(\left\langle r^{(d)}_{1,t}\right\rangle^{n}_{t=0}, \left\langle r^{(d)}_{2,t}\right\rangle^{n}_{t=0}\right)$ when $r^{(d)}_{i,n} := r^{(b)}_{i,n-1}$, $r^{(d)}_{i,t} := r^{(b)}_{i,t-1} - r^{(b)}_{i,t}$ for $0 < t < n$, and $r^{(d)}_{i,0} := -r^{(b)}_{i,0}$ [7].
Recall notations in Fig. 3. Let $0 \le t_1 < t_2 < n$ and $t_1 \le t'_i \le t_2$ be the smallest number such that $r_{i,t'_i} \neq 0$. $SW(U_{[t_1,t_2]}) := \left(SW\left(R_{1,[t_1,t_2]}\right), SW\left(R_{2,[t_1,t_2]}\right)\right)$, where $SW\left(R_{i,[t_1,t_2]}\right) := \langle r'_{i,t}\rangle^{t_2}_{t=t_1}$, $r'_{i,t} = 0$ if $t \neq t'_i$, and $r'_{i,t'_i} = r_{i,[t_1,t_2]}$.
Let $t_1 \le t'_i \le t_2$ be the largest number such that $r_{i,t'_i} \neq 0$, $s := r_{i,t'_i}/|r_{i,t'_i}|$, and $\bar{s} := -s$. $ZF(U_{[t_1,t_2]}) := \left(ZF\left(R_{1,[t_1,t_2]}\right), ZF\left(R_{2,[t_1,t_2]}\right)\right)$, where $ZF\left(R_{i,[t_1,t_2]}\right) := \langle r'_{i,t}\rangle^{t_2}_{t=t_1}$, $r'_{i,t} = r_{i,t}$ if $t < t'_i$, $r'_{i,t} = r_{i,t} - 2s$ if $t = t'_i$, $r'_{i,t} = \bar{s}$ if $t'_i < t < t_2$, and $r'_{i,t} = s$ if $t = t_2$.
$DI(\langle x, 0, 0, 3, 1\rangle) := DI(\langle x+1, 0, 0, -3, -3\rangle)$ and
$DI(\langle x, 0, 0, -3, -1\rangle) := DI(\langle x-1, 0, 0, 3, 3\rangle)$
Let a, b be natural numbers such that $a < b$, and $\left(\langle r'_{1,t'}\rangle^{t+b}_{t'=t+a}, \langle r'_{2,t'}\rangle^{t+b}_{t'=t+a}\right) = SW(U_{[t+a,t+b]})$. Denote $\omega(a,b) := \left|\left\{t' : (r'_{1,t'}, r'_{2,t'}) \neq O \text{ for } t+a \le t' \le t+b\right\}\right|$.
Let $\left(\langle r''_{1,t}\rangle^{t_2}_{t=t_1}, \langle r''_{2,t}\rangle^{t_2}_{t=t_1}\right) := ZF(U_{[t_1,t_2]})$, and $\left(\langle r'_{1,t}\rangle^{t_2-1}_{t=t_1}, \langle r'_{2,t}\rangle^{t_2-1}_{t=t_1}\right) := SW\left(\left(\langle r''_{1,t}\rangle^{t_2-1}_{t=t_1}, \langle r''_{2,t}\rangle^{t_2-1}_{t=t_1}\right)\right)$. $\overline{SW}\left(U_{[t_1,t_2]}\right) := \left(\langle r_{1,t}\rangle^{t_2}_{t=t_1}, \langle r_{2,t}\rangle^{t_2}_{t=t_1}\right)$ when $r_{i,t} = r'_{i,t}$ when $t < t_2$ and $r_{i,t_2} = r''_{i,t_2}$.
* $\mathcal{R}_{33} = \langle 0,0,0,3,0,3,3\rangle$, $\mathcal{R}_{31} = \langle 1,0,0,-3,0,-3,-3\rangle$,
$\mathcal{R}_{10} = \langle 0,0,0,1,0,1,0\rangle$, $\mathcal{R}_{14} = \langle 0,0,0,1,0,3,0\rangle$,
$\mathcal{R}_{18} = \langle 0,0,0,3,0,-3,0\rangle$, $\mathcal{R}_{22} = \langle 0,0,0,3,0,-1,0\rangle$,
$\mathcal{R}_{42} = \langle 1,0,0,-3,0,1,0\rangle$, $\mathcal{R}_{46} = \langle 1,0,0,-3,0,3,0\rangle$,
$\mathcal{R}_{50} = \langle 1,0,0,-1,0,-3,0\rangle$, $\mathcal{R}_{54} = \langle 1,0,0,-1,0,-1,0\rangle$.
Denote $-R_{i,[t_1,t_2]} = \langle -r_{i,t}\rangle^{t_2}_{t=t_1}$. $\mathcal{R}_{-y} = -\mathcal{R}_y$ for $y \in \mathbb{Z}_+$.

Fig. 5 Notations in Algorithm 2. The notations added in this letter are marked with asterisks (*).

Table 1 Comparison between Algorithm 2 and the algorithm in [5].

	Algorithm 2	Algorithm in [5]
Algorithmic scheme	If-Case Based	Dynamic Programming Based
#Lines of Codes	453 Lines	58 Lines
Computation Time	1.8 ms	0.7 ms
Dynamic Memory Usage (bits)	$6n + o(n)$	$105n + o(n)$

4. Comparison between Two Algorithms

By the result in this paper, we have two optimal algorithms for the minimum weight representation using $\{0, \pm 1, \pm 3\}$, Algorithm 2 and our algorithm in [5]. In this section, we will compare those two algorithms in several aspects, and give suggestions for researchers who will implement the algorithms.

We summarize our comparison results in Table 1. Those results are obtained by implementing both algorithms using Java in a personal computer with Intel Core i7-3770 CPU @3.40 GHz, 16.0 GB RAM, Windows 8 64 bits.

First, we compare the computation time of both algorithms. We randomly select 10,000 pairs of (r_1, r_2) such that $0 \le r_1, r_2 < 2^{512}$. Then, we insert those (r_1, r_2) as an input of the algorithms, and calculate average computations time that they use. For this aspect, the algorithm in [5] is clearly faster. While the algorithm spends 0.7 ms on average, Algorithm 2 spends 1.8 ms on average.

```
Input: A representation of $(r_1, r_2)$ in Booth recording
       $\left(\langle r_{1,t}\rangle_{t=0}^{n-1}, \langle r_{2,t}\rangle_{t=0}^{n-1}\right)$, such that $r_{i,t} = 0$ for $n - 10 \le t < n$
Output: A representation of $(r_1, r_2)$ in modified width-3 joint sparse
        form $\left(\langle r_{1,t}\rangle_{t=0}^{n-1}, \langle r_{2,t}\rangle_{t=0}^{n-1}\right)$
1  $STATE \leftarrow A; t \leftarrow 0$
2  if $STATE = A$ then
3      if $u_j = O$ (A-1) then $t \leftarrow t + 1$ ;
4      else if there exists $i \in \{1, 2\}$ such that $r_{i,t} \ne 0$ and $r_{\bar{i},[t,t+2]} = 0$ (A-2)
       then
5          $U_{[t,t+2]} \leftarrow SW(U_{[t,t+2]}); t \leftarrow t + 3$
6      else if there exists $i \in \{1, 2\}$ such that $r_{i,t} \ne 0$ and $r_{\bar{i},[t,t+1]} = 0$ (A-3)
       then
7          $U_{[t,t+2]} \leftarrow \overline{SW}(U_{[t,t+2]}); t \leftarrow t + 2; STATE \leftarrow B$
8      else if there exists $i \in \{1, 2\}$ such that $r_{i,t} \ne 0$, $r_{\bar{i},t} = 0$, and
       $r_{i,[t,t+3]} \in \{\pm 7\}$ (A-4) then
9          $U_{[t,t+2]} \leftarrow \overline{SW}(U_{[t,t+2]}); t \leftarrow t + 2; STATE \leftarrow B$
10     else if there exists $i \in \{1, 2\}$ such that $r_{\bar{i},t} = 0$ and $r_{i,[t,t+3]} \in \{\pm 7\}$
       (A-4) then
11         $STATE \leftarrow D$
12     else if there exists $i \in \{1, 2\}$ such that $r_{i,[t,t+6]} \in \{\pm 33\}$ and
       $r_{\bar{i},[t,t+6]} \in \{\pm 10, \pm 14, \pm 18, \pm 22\}$ (*AA-1) then
13         $R_{i,[t,t+6]} \leftarrow \mathcal{R}_{r_{i,[t,t+6]}}; R_{\bar{i},[t,t+6]} \leftarrow \mathcal{R}_{r_{\bar{i},[t,t+6]}}$
14         $t \leftarrow t + 7$;
15     else if there exists $i \in \{1, 2\}$ such that $r_{i,[t,t+6]} \in \{\pm 31\}$ and
       $r_{\bar{i},[t,t+6]} \in \{\pm 42, \pm 46, \pm 50, \pm 54\}$ (*AA-2) then
16         $R_{i,[t,t+6]} \leftarrow \mathcal{R}_{r_{i,[t,t+6]}}; R_{\bar{i},[t,t+6]} \leftarrow \mathcal{R}_{r_{\bar{i},[t,t+6]}}$
17         $t \leftarrow t + 6; STATE \leftarrow B$
18     else if there exists $i \in \{1, 2\}$ such that $r_{i,t} \ne 0$ and $r_{\bar{i},t} = 0$ (A-5)
       then
19         $U_{[t,t+1]} \leftarrow ZF(U_{[t,t+1]}); t \leftarrow t + 1; STATE \leftarrow B.$
20     else (A-6) $STATE \leftarrow B$;
21 else if $STATE = B$ then
22     if $\omega(3, 4) \le 1$ (B-1) then
23         $U_{[t,t+2]} \leftarrow SW(U_{[t,t+2]}); t \leftarrow t + 3; STATE \leftarrow E$
24     else (B-2) $STATE \leftarrow C$;
25 else if $STATE = C$ then
26     if $\omega(5, 6) \le 1$ (C-1) then
27         $U_{[t,t+2]} \leftarrow \overline{SW}(U_{[t,t+2]}); U_{[t+2,t+4]} \leftarrow SW(U_{[t+2,t+4]})$;
28         $t \leftarrow t + 5; STATE \leftarrow E$
29     else if there exists $i \in \{1, 2\}$ such that $r_{i,[t+3,t+6]} \in \{\pm 7\}$ (C-2) then
30         $U_{[t,t+2]} \leftarrow SW(U_{[t,t+2]}); t \leftarrow t + 3; STATE \leftarrow D$
31     else (C-3) $U_{[t,t+2]} \leftarrow \overline{SW}(U_{[t,t+2]}); t \leftarrow t + 2$;
32 else if $STATE = D$ then
33     Let $i \in \{1, 2\}$ be an integer such that $r_{i,[t,t+3]} \in \{\pm 7\}$.
34     $U_{[t,t+1]} \leftarrow ZF(U_{[t,t+1]}); U_{[t+3,t+1]} \leftarrow SW(U_{[t+3,t+1]})$;
       $R_{i,[t+4,t]} \leftarrow DI(R_{i,[t+4,t]})$ if $r_{i,t+4} \ne 0$;
35     if $r_{\bar{i},t+4} = 0$ (D-1) then
36         $t \leftarrow t + 5; STATE \leftarrow A$
37     else (D-2) $t \leftarrow t + 4; STATE \leftarrow B$;
38 else
39     if $U_t = 0$ (E-1) then $t \leftarrow t + 1; STATE \leftarrow A$;
40     else if there exists $i \in \{1, 2\}$ such that $r_{i,t} \ne 0$ and $r_{\bar{i},[t,t+2]} = 0$ (E-2)
       then
41         $U_{[t,t+2]} \leftarrow SW(U_{[t,t+2]}); t \leftarrow t + 2; STATE \leftarrow A$
42     else if there exists $i \in \{1, 2\}$ such that $r_{i,t} \ne 0$ and $r_{\bar{i},[t,t+1]} = 0$ (E-3)
       then
43         $U_{[t,t+2]} \leftarrow \overline{SW}(U_{[t,t+2]}); t \leftarrow t + 2; STATE \leftarrow B$
44     else (E-4) $STATE \leftarrow B$;
45 Go to Line 2, if there is more bits to process
```

Algorithm 2: An algorithm for width-3 joint sparse form with our modifications marked by asterisks (*)

Algorithm 2 is clearly a better algorithm, when we compare dynamic memory usage. The algorithm can finish a computation without using additional working memory. Because of that, the authors of [6] claim that Algorithm 2 is suitable for an environment with limited computation resource, such as smart cards. However, this if-case-based algorithm is complicated. Although we try our best to shorten our java code, the algorithm needs around 300 lines before our modification in Sect. 2. After our modification, the code length is 453 lines. To store the program in the smart card, we have to store it in its static memory such as ROM. Since it is discussed in [8] that the smart card in our current technology does not contain a static memory larger than a few hundred Kilobytes, we strongly believe that it is hard to deploy Algorithm 2 to that environment.

While the algorithm in [5] use more dynamic memory, the length of the code for the algorithm is around 9 times shorter than Algorithm 2. Because of that, we suggest to use Algorithm 2 in the computation environment with limited dynamic memory, and the algorithm in [5] in the computation environment with limited static memory.

5. Conclusion

Although our algorithm in [3] is not the best algorithm in all aspects, the results can be applied to several classes of number representations (e.g. [9]). Because of that, the results can be useful, when researchers develop an efficient algorithm for a minimum weight representation. They can compare their results with ours to check the difference between their solutions and the optimal solutions. They can also use our solutions to find the way to improve their algorithm, in the similar way we have done in this letter.

Acknowledgement

The authors would like to thank an anonymous reviewer who kindly give us several useful comments during the course of this research.

References

[1] V. Suppakitpaisarn and H. Imai, "Yet another proof for optimal joint average hamming weight of width-3 sparse form," Proc. AAAC'13, 2014.

[2] D. Johnson, A. Menezes, and S. Vanstone, "The elliptic curve digital signature algorithm (ECDSA)," Int. J. of Information Security, vol.1, no.1, pp.36–63, 2001.

[3] V. Suppakitpaisarn, M. Edahiro, and H. Imai, "Fast elliptic curve cryptography using minimal weight conversion of d integers," Proc. AISC'12, pp.15–26, 2012.

[4] J. Solinas, "Low-weight binary representations for pairs of integers," Tech. Rep., CORR 2001-41, Centre for Applied Cryptographic Research, University of Waterloo, 2001.

[5] V. Suppakitpaisarn and M. Edahiro, "Fast scalar-point multiplication using enlarged digit set on integer pairs," Proc. SCIS'09, vol.14, 2009.

[6] K. Okeya, H. Kato, and Y. Nogami, "Width-3 joint sparse form," Proc. ISPEC'10, pp.67–84, 2010.

[7] A.D. Booth, "A signed binary multiplication technique," The Quarterly Journal of Mechanics and Applied Mathematics, vol.4, no.2, pp.236–240, 1951.

[8] H. Ko and R.D. Caytiles, "A review of smartcard security issues," J. Security Engineering, vol.8, no.3, pp.359–370, 2011.

[9] F. Heigl and C. Heuberger, "Analysis of digital expansions of minimal weight," DMTCS Proceedings, no.01, pp.399–412, 2012.

Appendix

In this section, we will explain how we construct the Markov chain in Fig. 4 in detail. When we reach Line 45 of Algorithm 2 and there are more bits to process, we will loop back to Line 2 of the algorithm. We will consider the value of variable $STATE$ at the beginning of each loop. By denoting the value of $STATE$ at τth iteration of the algorithm as $STATE_\tau$, the transition probabilities can be written as $\Pr\{STATE_\tau = s' | STATE_{\tau-1} = s\}$ for $s, s' \in \{A, B, B', C, C', D, E\}$. For simplicity, we refer to $\Pr\{STATE_\tau = s' | STATE_{\tau-1} = s\}$ as $p_{s,s'}$.

For most transitions, the probabilities that we will get into each condition, (A-1), ..., (E-4), does not depend on how we process the inputs in the previous iteration. In other words, each condition does apply on different bits of inputs. Because of that, we can calculate the transition probabilities by assuming that the input distribution is uniform. The transitions that we can assume that uniformity are edges departing from nodes A, B, C, D, E. By the uniformity, we can use the transition probabilities calculated in [6] for those edges.

Unfortunately, we cannot assume that uniformity for the edges departing from B', C', E'. We will explain the reason, and calculate the transition probability in the following lemmas and corollary. In the proof of those lemmas and corollary, we sometimes refer to state B', C', E' as B, C, E if it is clear from the context.

Lemma 2 (Probability when $STATE_{\tau-1} = B'$).

$$p_{B',C'} = 22/94, p_{B',E} = 72/94,$$

and $p_{B',s} = 0$ *for* $s \in \{A, B, C, D, E'\}$.

Proof. Since we can assume that the input is uniform when $STATE = A$, we can fall into conditions (AA-1), (AA-2), and (A-5) with probabilities $1/512, 1/512$, and $47/256$ respectively. Those three conditions represent the case when $r_{i,t} \neq 0$, $r_{\bar{i},t} = 0$. The probability that $STATE = A$, $r_{i,t} \neq 0$, $r_{\bar{i},t} = 0$, and $\Omega(4,5) \leq 1$ is $48/256 \times 3/4 = 36/256$, but $\Omega(4,5)$ is always equal to 2 in the case (AA-1) and (AA-2). Therefore, the probability that (A-5) will be selected with $\Omega(4,5) \leq 1$ is $36/256$, while the probability for $\Omega(4,5) = 2$ is $48/256 \times 1/4 - 1/256 = 11/256$.

Since t is incremented by 1 at Line 19, $\Omega(4,5)$ becomes $\Omega(3,4)$ there. By the fact that the only incoming edge to B' is (A-5), we know that the probability that $\Omega(3,4) \leq 1$ when $STATE = B'$ is $36/(36+11) = 36/47$, and the probability for $\Omega(3,4) = 2$ is $11/47$.

Since the conditions (AA-1) and (AA-2) are invoked only if $\Omega(3,4) = 2$, the distribution on the larger bits is uniform if $\Omega(3,4) \leq 1$. Because of that, we can move to a node E, when the condition (B-1) is satisfied. □

Lemma 3 (Probability when $STATE_{\tau-1} = C'$).

$$p_{C',C} = 5/44, p_{C',D} = 5/44, p_{C',E'} = 17/22,$$

and $p_{C',s} = 0$ *for* $s \in \{A, B, C', E\}$.

Proof. From the previous proof, the probability that $r_{i,t} \neq 0$, $r_{\bar{i},t} = 0$, $\Omega(4,5) = 2$ is $12/256$ when $STATE = A$. Since the distribution is uniform, we know that those conditions with $\Omega(6,7) \leq 1$ are satisfies with probability $12/256 \times 3/4 = 9/256$. By that, the probability for $\Omega(6,7) = 2$ is $3/256$. The probability that (AA-1) or (AA-2) is satisfied, $\Omega(4,5) = 2$, and $\Omega(6,7) \leq 1$ is $1/512$. Therefore, the probability that (A-5) is satisfied, $\Omega(4,5) = 2$, and $\Omega(6,7) \leq 1$ is $9/256 - 1/512 = 17/512$. The probability is $3/256 - 1/512 = 5/512$ for the case where $\Omega(6,7) = 2$.

By the fact that the only incoming edge to C' is from (A-5), (B'-2) and t is incremented by 1 at (A-5), we know that $\Omega(5,6) \leq 1$ when $STATE = C'$ is $17/(5+17) = 17/22$. The probability for $\Omega(5,6) = 2$ is $5/22$.

When $\Omega(6,7) = 2$, $\Omega(4,5) = 2$, and (AA-1) or (AA-2) are satisfied, the probability that (C-2) is satisfied is $1/2$. Therefore, the probability that the condition in (C-2) is satisfied after (A-5) does not change by the newly-added (AA-1) and (AA-2). The probability for (C-2) is $5/22 \times 1/2 = 5/44$. From there we know that the probability for (C-3) is also $1 - 17/22 - 5/44 = 5/44$.

By (A-5) and (C-2), t is incremented by 4. The condition when $STATE = D$ is imposed on bit $t+4$, which is $t+8$ when $STATE = A$. Since the newly-added conditions are imposed from t to $t+7$, they cannot interfere the condition for $STATE = D$. We do not need to add a node D', and the transition from C' by (C-2) move us to state D. Similarly, we increase t by 3 at (A-5) and (C-3), and the condition when $STATE = C$ is imposed on $t+5$ and $t+6$. Since they are $t+8$ and $t+9$ when $STATE = A$, they are uniform when we get from C' by (C-3). Therefore, that transition move us to state C. □

Lemma 4. *When* $STATE = E'$, *the probability that the condition (E-1) is satisfied is* $6/17$.

Proof. From the previous proof, we know that the probability that $r_{i,t} \neq 0$, $r_{\bar{i},t} = 0$, $\Omega(4,5) = 2$, and $\Omega(6,7) \leq 1$ is $9/256$ when $STATE = A$. By the proof in [6], we know that $U_{t+6} = 0$ with probability $1/3$ under those conditions. Hence, the probability that $r_{i,t} \neq 0$, $r_{\bar{i},t} = 0$, $\Omega(4,5) = 2$, $\Omega(6,7) \leq 1$, and $U_{t+6} \neq 0$ is $9/256 \times 2/3 = 6/256$.

When $STATE = A$ and (AA-1) or (AA-2) are satisfied, U_{t+6} is always not equal to 0. The probability that (AA-1) or (AA-2) is satisfed, $\Omega(4,5) = 2$, and $\Omega(6,7) \leq 1$ is equal to $1/512$. By that, the probability that (A-5) is satisfed, $\Omega(4,5) = 2$, $\Omega(6,7) \leq 1$, and $U_{t+6} = 0$ is equal to $6/256 - 1/512 = 11/512$. We also know that the probability that (A-5) is satisfed, $\Omega(4,5) = 2$, $\Omega(6,7) \leq 1$, and $U_{t+6} \neq 0$ is equal to $3/256 = 6/512$.

By the fact that the only incoming edge to E' is from (A-5), (B'-2), (C'-1) and t is incremented by 6 at (A-5) and (C'-1), we know that $U_t = 0$ with probability $6/(6+11) = 6/17$ when $STATE = E'$. □

Lemma 5. *When $STATE = E'$, the probability that the condition (E-4) is satisfied is* $6/17$.

Proof. It is shown in [6] that $\Omega(0,1) \leq 1$ when $STATE = E$, and it is obvious that the property is also hold for E'. Therefore, the condition (E-4) is satisfied, when $r_{i,t} \neq 0$ and $r_{\bar{i},t} \neq 0$.

From the previous proof and the proof in [6], we know that the probability that $r_{i,t} \neq 0$, $r_{\bar{i},t} = 0$, $\Omega(4,5) = 2$, $\Omega(6,7) \leq 1$, and $r_{1,t+6}r_{2,t+6} = 0$ is $9/256 \times 2/3 = 6/256$.

When $STATE = A$ and (AA-1) or (AA-2) are satisfied, $r_{1,t+6}r_{2,t+6}$ is always equal to 0. The probability that (AA-1) or (AA-2) is satisfed, $\Omega(4,5) = 2$, and $\Omega(6,7) \leq 1$ is equal to $1/512$. By that, the probability that (A-5) is satisfed, $\Omega(4,5) = 2$, $\Omega(6,7) \leq 1$, and $r_{1,t+6}r_{2,t+6} = 0$ is equal to $6/256 - 1/512 = 11/512$. We also know that the probability that (A-5) is satisfed, $\Omega(4,5) = 2$, $\Omega(6,7) \leq 1$, and $r_{1,t+6}r_{2,t+6} \neq 0$ is equal to $3/256 = 6/512$.

By the fact that the only incoming edge to E' is from (A-5), (B'-2), (C'-1) and t is incremented by 6 at (A-5) and (C'-1), we know that $r_{1,t}r_{2,t} \neq 0$ with probability $6/(6+11) = 6/17$ when $STATE = E'$. □

Lemma 6. *When $STATE = E'$, the probability that the condition (E-2) is satisfied is* $5/34$. *The probability that the condition (E-3) is satisfied is also* $5/34$.

Proof. Consider the condition statement of (E-2) and (E-3). The only difference between those two conditions is $r_{\bar{i},t+2}$. To proof this lemma, we will show that $\Pr\{r_{\bar{i},t+2} = 0\} = \Pr\{r_{\bar{i},t+2} \neq 0\} = 0.5$. That statement is true by the fact that the only incoming edge to E' is from (A-5), (B'-2), (C'-1) and t is incremented by 6 at (A-5) and (C'-1). The bit $t+2$ when $STATE = E'$ is $t+8$ when $STATE = A$. Since the newly-added condition (AA-1) and (AA-2) do impose the non-uniformity on bit t to $t+6$, we can assume that the bit $t+8$ is uniform when $STATE = E'$. □

Corollary 1 (Probability when $STATE_{\tau-1} = E'$)**.**

$$p_{E',A} = 6/17 + 5/34 = 1/2,$$
$$p_{E',B} = 5/34 + 6/17 = 1/2,$$

and $p_{E',s} = 0$ *for* $s \in \{B', C, C', D, E, E'\}$.

Proof. When (E-1), (E-2), or (E-3) is satisfied, we increase the value of t by at least 1. After the increment, the value is larger than the value when $STATE = A$ by at least 7. Because of the non-uniformity caused by (AA-1) and (AA-2) will affect only 7 bits, the distribution of input becomes uniform again. We can move to state A and B after the transition.

We do not increase t when (E-4) is satisfied. However, the bit t is not used in the condition of the next state, state B, and the following state, state C. After we reach those states, the value t is increased, and the distribution becomes uniform. Hence, we can move to state B after the condition (E-4) is selected. □