Some Analysis of Radix-r Representations

Dong-Guk Han and Tsuyoshi Takagi

FUTURE UNIVERSITY-HAKODATE, JAPAN
{christa,takagi}@fun.ac.jp

Abstract. We deal with the radix-r representation used for the scalar multiplication of pairing-based cryptosystems with characteristic r. Our goal of this paper is to present some invariant properties about the signed radix-r representation; (1) approximation formulae for the average significant length and the average hamming weight of gNAF and wrNAF representation, (2) some classification formulae of equivalent classes called as Cutting Lemma, Collision Lemma, and Search Space Theorem. We also analyze the security of signed radix-r representations in the sense of side channel attacks, and to this end we propose a secure countermeasure.

Keywords: *pairing-based cryptosystems, radix-r representation, gNAF, wrNAF, side channel attacks.*

1 Introduction

The bilinear pairings such as Weil pairing or Tate pairing have recently been used for applications in cryptography, for example Joux's three-party key agreement [Jou00], ID-based encryption [BF01], short signature [BLS01], and so on. In general, main operations in these systems using secret information are as follows;

(1) a bilinear pairing computation $e(Q, R)$,

(2) a scalar multiplication dP,

where P, Q, R are points on the underlying elliptic curve and d is an integer scalar. Either the point Q of $e(Q, R)$ or the scalar d of dP is used for the secret key.

These two operations are computationally expensive, specially the pairing computation is time and memory consuming. Thus Barreto et al. and Galbraith et al. showed efficient pairing and scalar multiplication algorithms over supersingular elliptic curves [BKL$^+$02,GHS02] with characteristic three. Several efficient scalar multiplications of elliptic curves with characteristic three have been investigated [BGK$^+$03,HPS02,PS02,SW02]. In addition, Duursma and Lee proposed a closed formula for the Tate pairing on the hyperelliptic curve in characteristic r [DL03]. The variants of Duursma-Lee algorithm is currently one of the fastest schemes [BGhE$^+$04]. In this case, the radix-r representation is utilized for the efficient implementation of the pairing-based cryptosystems. Recently, Takagi et al. proposed non-adjacent form of width-w radix-r representation (wrNAF) and efficient algorithms for generating wrNAF to achieve faster scalar multiplication of the pairing-based cryptosystems [TYW04].

Since pairing-based cryptosystems are considered as an ideal partner for identity aware devices which have scarce computational resources (like Smart cards or RFID) we have to make an effort to optimize the memory and efficiency. These memory and power constraint devices will be carried into and used in hostile environments and often house sensitive information, for example identity related tokens or financial information, the threat of attack is significant. This threat is magnified by both the potential pay-off and level of anonymity that side channel attacks (SCA) allow. Recently, side channel attacks on pairing $e(Q, R)$ was discussed by Page and Vercauteren [PV04]. They analyzed the security of Boneh-Franklin encryption scheme [BF01] and gave some countermeasures.

One of their countermeasures is to use the concept of point blinding to randomize the points, e.g. $e(aQ, bR)$ for randomly chosen integer a, b with $ab = 1 \bmod q$, where q is the order of underlying elliptic curve. However, its security depends on that of scalar multiplication aQ or bR. Thus it is important to investigate how to securely and efficiently implement the scalar multiplication dP for pairing-based cryptosystems.

1.1 Our Achievements

We assume that the standard radix-r scalar multiplication is utilized for computing dQ, which satisfies following conditions;

- if the associated digit is zero then only elliptic curve r-th powering is calculated,
- else both elliptic curve r-th powering and elliptic addition are both calculated.

This algorithm is called Standard_Radix-r_Method in this paper. We investigate the efficiency and security of some encoding algorithms of a input integer d.

Precise Estimation of Efficiency: We propose approximation formulae for the average significant length and the average Hamming weight of the radix-r generalized non-adjacent form (gNAF) [CL73] and wrNAF representation [TYW04]. Note that Cohen investigated some approximation formulae of the width-w NAF, namely in the case $w2$NAF [Coh05]. He assumed that the integer d used as a scalar in scalar multiplication ranges uniformly among the integers having exactly n bits, in other words $2^{n-1} \leq d < 2^n$. In this paper, we assume that the radix-r gNAF or wrNAF is generated from the randomly chosen n digits radix-r representation of d, i.e., $0 \leq d < r^n$, this is more natural assumption in the application of cryptography.

– In the case of radix-r gNAF representation, the average significant length and the average number of non-zero digits are $n - \frac{1}{(r-1)(r+1)}$ and $n\frac{r-1}{r+1} + \frac{r^2+5r-4}{r(r+1)^2}$ for non-small n, respectively.
– In the case of wrNAF representation, those are $n - \frac{w(w-2)(r-1)^2+(w-1)(r-1)+1}{2(w(r-1)+1)(r-1)}$ and $n\frac{r-1}{w(r-1)+1} + \frac{w^2(r-1)^2+w(3(r-1))-(r-1)+1}{2(w(r-1)+1)^2}$ for non-small n, respectively.

In order to confirm these estimation formulae we show simulation results for the practically used security level, e.g., 160 bits. Our simulation results match with the theoretical estimation for randomly chosen one million scalars. These experiment results are contained in Section 4.

Some Invariant Properties: Given an n-digit integer d with radix-r representation if the digit set is allowed to have a redundancy, the integer d can be expanded in different signed radix-r representations. In this paper, we assume the used digit set is $\{0, \pm 1, \ldots, \pm(r-1)\}$. Suppose that $\varepsilon^{(1)} = \sum_{j=0}^{n} \varepsilon_j^{(1)} r^j$ and $\varepsilon^{(2)} = \sum_{j=0}^{n} \varepsilon_j^{(2)} r^j$ are two different signed radix-r representations of d. We prove the following invariant properties.

– (*Cutting Lemma*) We propose cutting conditions among signed radix-r representations; if $\varepsilon_t^{(1)} = \varepsilon_t^{(2)} = 0$ or $\varepsilon_t^{(1)}$ and $\varepsilon_t^{(2)}$ are all non-zero for $0 < t < n$, then

$$\sum_{j=0}^{t-1} \varepsilon_j^{(1)} r^j = \sum_{j=0}^{t-1} \varepsilon_j^{(2)} r^j \textit{ and } \sum_{j=t}^{n} \varepsilon_j^{(1)} r^j = \sum_{j=t}^{n} \varepsilon_j^{(2)} r^j.$$

– (*Collision Lemma*) We can find collision pairs between the representations of unsigned radix-r and that of signed radix-r when we cut these two representations in $(t+1)$-digit position, e.g. $\sum_{j=0}^{t} \varepsilon_j^{(i)} r^j$ is exactly same to one of $\{\sum_{j=0}^{t} d_j r^j, \sum_{j=0}^{t} d_j r^j - r^{t+1}\}$.

– (*Search Space Theorem*) Assume that we have m signed radix-r representations $\{\varepsilon^{(i)}\}$ of an original integer d and know only the position of zero digit in $\varepsilon^{(i)}$. Our interest is to find the search space order to detect the exact value d. We propose a novel formula computing the exact search space order only with the information of the position of zero digit in $\varepsilon^{(i)}$. If we utilize *Cutting Lemma* then the required memory to find the search space is on average $(\frac{n+1}{x}) \cdot \left\{\{2(r-1)\}^{\frac{r-1}{r}\frac{n+1}{x}} \cdot x \cdot m\right\}$ length digits if the original $\{\varepsilon^{(i)}\}$ is cut into x parts. However, if *Cutting Lemma* is not used, i.e. we use the exhaustive search method (ESM), then $(n+1) \cdot \left\{\{2(r-1)\}^{\frac{r-1}{r}(n+1)} \cdot m\right\}$ length digits memory is required. Roughly saying, the memory complexity of the proposed one is x-th root of that of ESM.

These results are contained in Section 5.

Security against Side Channel Attacks: By using the above proposed properties we can conclude that Standard_Radix-r_Method with random recoding technique is analyzed by simple power analysis (SPA). Moreover, even though SPA countermeasure is added to the target algorithm it is proved that that is not secure either. Note that these consequences are very natural extension of [HOK+04,FMPV04,SPL04]. Last, we propose an optimal countermeasure against SCA.

This paper is organized as follows: In Section 2 we shortly review pairing-based cryptosystems. In Section 3 we review some properties related to the radix-r representation and define some notations. In Section 4 we precisely analyze the efficiency of radix-r gNAF and wrNAF. In Section 5 we investigate some invariant properties related to the signed radix-r representation and security analysis of Standard_Radix-r_Method against side channel attacks. In Section 6 we present an optimal countermeasure against SCA.

2 Pairing-Based Cryptosystems

Positive cryptographic applications based on pairings arose from the work of Joux [Jou00], who gave a simple one round tripartite Diffie-Hellman protocol on supersingular curves. Curve based pairings, such as the Weil pairing and Tate pairing, provide a good setting for the so called bilinear Diffie-Hellman problem. Many cryptographic schemes based on the pairings have been developed recently, such as Joux's three-party key agreement, ID-based encryption [BF01], short signature [BLS01], and so on.

Let G_1, G_2 be two groups of the same prime order q. We view G_1 as an additive group and G_2 as a multiplicative group. Let P be an arbitrary generator of G_1 and a mapping $e : G_1 \times G_1 \rightarrow G_2$ be a cryptographic bilinear map.

In general, main operations in pairing-based cryptosystems using secret information are as follows;

(1) $e(Q, R)$ for $Q, R \in G_1$: a bilinear pairing computation,

(2) dP for $d \in Z_q^*$: a scalar multiplication in an additive group.

Since pairings and scalar multiplication underpin cryptographic protocols such as Identity Based Encryption [BF01] and Short Signature [BLS01], one might see them as an ideal application for the same identity aware, ubiquitous computing devices that are vulnerable to side channel attacks. For example,

- $e(S_{ID}, U)$ where S_{ID} is the private key used in decryption step and U is a public data in ID-Based Encryption scheme [BF01],

- dP where d is a secret key for a signer in Boneh-Lynn-Shacham short signature scheme [BLS01].

Recently, side channel attacks on pairings was first proposed by Page-Vercauteren [PV04]. They analyzed the security of pairing $e(P, Q)$ in Boneh-Franklin encryption scheme [BF01] and gave some countermeasures. One of their countermeasures is to use the concept of point blinding to randomize the points fed into the pairings. By using the relationship $e(aP, bQ) = e(P, Q)^{ab}$ they randomized the points P and Q by selecting random values for a and b such that $ab = 1 \bmod q$. But the security of this countermeasure depends on that of scalar multiplication because if a or b is revealed during the computation of aP and bQ then the computation of $e(aP, bQ)$ is also no more secure against the Messerges style differential power analysis proposed in [PV04]. Thus, in order to achieve secure implementation of pairing-based cryptosystems, we have to investigate how to securely implement the scalar multiplication.

Therefore, in this paper we are mainly concerned with efficiency and security of the scalar multiplication of pairing-based cryptosystems over characteristic r. The following algorithm is a standard method to compute elliptic scalar multiplication.

Standard_Radix-r_Method

Input: A point P and $d = \sum_{j=0}^{n-1} d_j r^j$, d_j in digit set $\mathcal{D}$
Output: $Q = dP$

1. precompute $|a|P$ for all positive $a \in \mathcal{D}$.
2. $Q \leftarrow \mathcal{O}$ (point of infinity),
3. for $j = n-1$ downto 0
   3.1. $Q \leftarrow rQ$
   3.2. if $d_j > 0$ $Q \leftarrow Q + d_j P$
   3.3. if $d_j < 0$ $Q \leftarrow Q - |d_j| P$
4. Return Q

In general elliptic curves defined over fields with characteristic r have an efficient formula for computing rP [SW02]. Therefore, the scalar multiplication with the standard radix-r algorithm above is usually more efficient than that using a double-and-add method from binary representation [HPS02,PS02].

3 Efficient Signed Radix-r Representation

In this section we shortly review some efficient signed radix-r representation.

First we want to define a precise meaning of radix-r representation of n digits. If a positive integer d is called as the radix-r representation of n digits (or n-digit radix-r representation), then it satisfies following two conditions;
(1) d distributes uniformly in $\{0, 1, 2, ..., r^n - 1\}$, in other words $0 \le d < r^n$,
(2) $d = \sum_{j=0}^{n-1} d_j r^j := (d_{n-1}, \ldots, d_0)_r$, for $d_j \in \{0, 1, ..., r-1\}$.

This representation is unique and the density of non-zero digits of the radix-r representation is $\frac{1}{r}$. In this paper we are interested in cryptographic applications of the radix-r representation, so that we assume that n is not small and $n \gg r$.

If the digit set of the radix-r representation is allowed to have a redundancy, the integer d can be expanded in different signed radix-r representations. Assume we use the digit set $\Gamma_r = \{0, \pm 1, \pm 2, ..., \pm(r-1)\}$. When we consider the radix-r representation of an integer d as one of its signed radix-r representations, different signed radix-r representations of d can be obtained by using following local replacing rules of two consecutive digits;

$$
\begin{array}{rcl@{\qquad\qquad}rcl}
(0,1)_r & \Leftrightarrow & (1,\overline{r-1})_{sr} & (0,\overline{1})_{sr} & \Leftrightarrow & (\overline{1},r-1)_{sr} \\
(0,2)_r & \Leftrightarrow & (1,\overline{r-2}_{sr} & 0,\overline{2})_{sr} & \Leftrightarrow & (\overline{1},r-2)_{sr} \\
 & \vdots & & & \vdots & \\
(0,r-1)_r & \Leftrightarrow & (1,\overline{1})_{sr} & (0,\overline{r-1})_{sr} & \Leftrightarrow & (\overline{1},1)_{sr}
\end{array}
$$

For example, consider $d = (0,0,0,1,0,2,0,2,0,2)_3$ which is one of radix-3 representations of 10 digits. The different signed radix-3 representations of d are: $(1,0,2,0,2,1,\overline{1})_{s3}$, $(1,0,2,1,\overline{1},1,\overline{1})_{s3}$, $(1,0,2,1,0,\overline{2},\overline{1})_{s3}$, $(1,\overline{2},\overline{2},\overline{2},0,2,1,0,\overline{2},\overline{1})_{s3}$, and so forth. By using the above replacing rules we can make any signed radix-3 representations which has a wanted length of digits, e.g., if we want l (> 7) digits representation for $(1,0,2,0,2,0,2)_3$ then the most simple way is $(1,\underbrace{\overline{2},\overline{2},\cdots,\overline{2},\overline{2},}_{l-7\text{ times}}0,2,0,2,0,2)_{s3}$.

If the used digit set is changed, then the replacing rules and generated signed radix-r representations are changed.

3.1 rSD Representation and Radix-r gNAF Form

We define radix-r signed digit (rSD) representation in the following.

Definition 1. *For a given n-digit radix-r representation $d = (d_{n-1},\ldots,d_1,\ d_0)_r$, a signed radix-$r$ representation ε of d is called $(n{+}1)$-digit radix-r signed digit (rSD) representation if it satisfied the following conditions.*
(1) ε is represented within length $n+1$, i.e., $\varepsilon = (\varepsilon_n,\ldots,\varepsilon_1,\varepsilon_0)_{rSD}$,
(2) $\varepsilon_j \in \Gamma_r = \{0,\pm1,\pm2,...,\pm(r-1)\}$.

There exist infinitely many rSD representations of d if the length of extension is not fixed. In the definition of rSD we consider only the rSD representation of $n+1$ bits for a given n-digit integer d.

In general rSD representation is not unique, however, the generalized non-adjacent form (gNAF) is known as a class whose hamming weight is minimal in all rSD representations and can uniquely represent each integer [CL73]. Clearly, radix-r gNAF is one of rSD representations, which satisfies the following two conditions:

$$|\gamma_i + \gamma_{i+1}| < r \text{ for all } i, \qquad |\gamma_i| < |\gamma_{i+1}| \text{ if } \gamma_i\gamma_{i+1} < 0.$$

The radix-r gNAF is generated by the following algorithm [TYW04].

Integer to radix-r gNAF

Input: integer d and radix r.
Output: the radix r gNAF of d: $= (...,\gamma_1,\gamma_0)_{gNAF}$.

1. $i \leftarrow 0$
2. while $d > 0$ do the following
   - 2.1. $v \leftarrow d \bmod r^2$; let $v_0 \leftarrow v \bmod r$, $v_1 \leftarrow (v - v_0)/r$,
   - 2.1.1. if $v \bmod r = 0$ then $\gamma_i \leftarrow 0, d \leftarrow d/r, i \leftarrow i+1$,
   - 2.1.2. else if $v < r$ then $\gamma_{i+1} \leftarrow 0, \gamma_i \leftarrow v$, $d \leftarrow (d-\gamma_i)/r^2, i \leftarrow i+2$,
   - 2.1.3. else if $r < v < r^2 - r$
   - 2.1.3.1. if $(v_0 + v_1) \geq r$ then $\gamma_i \leftarrow v_0 - r$, $d \leftarrow (d-\gamma_i)/r, i \leftarrow i+1$,
   - 2.1.3.2. else $\gamma_i \leftarrow v_0$, $d \leftarrow (d-\gamma_i)/r, i \leftarrow i+1$,
   - 2.1.4. else if $r^2 - r < v$ then $\gamma_{i+1} \leftarrow 0, \gamma_i \leftarrow v_0 - r$, $d \leftarrow (d-\gamma_i)/r^2, i \leftarrow i+2$,
3. Return $(...,\gamma_1,\gamma_0)_{gNAF}$.

The average density of the non-zero digit of the radix-r gNAF is equal to $\frac{r-1}{r+1}$ for non-small n and $n \gg r$.

3.2 *wr*NAF Form

Another efficient class of the signed radix-r representation is the radix-r width-w non-adjacent form (wrNAF) [TYW04]. wrNAF is defined by
(1) there is at most 1 non-zero digit among any w adjacent digits
(2) $\delta_j \in D_{w,r} = \{0, \pm 1, \pm 2, ..., \pm\lfloor\frac{r^w-1}{2}\rfloor\} \setminus \{\pm 1r, \pm 2r, ..., \pm\lfloor\frac{r^{w-1}-1}{2}\rfloor r\}$.
(3) the most non-zero digit is positive.
Every integer can be uniquely represented using wrNAF. The hamming weight of wrNAF representation is minimal among all signed representations using digit set $D_{w,r}$. An algorithm for generating wrNAF is as follows:

Integer to wrNAF

Input: integer d, radix r, and width w.
Output: the wrNAF of d: $= (..., \delta_1, \delta_0)_{wrNAF}$.

1. $i \leftarrow 0$
2. While $d > 0$ do the following
   - 2.1. $v \leftarrow d \bmod r^w$,
   - 2.1.1. if $v \bmod r = 0$ then $\delta_i \leftarrow 0$,
   - 2.1.2. else if $v < r^w/2$ then $\delta_i \leftarrow v$, $d \leftarrow d - \delta_i$,
   - 2.1.3. else $\delta_i \leftarrow v - r^w$, $d \leftarrow d - \delta_i$,
   - 2.2. $d \leftarrow d/r, i \leftarrow i + 1$,
3. Return $(..., \delta_1, \delta_0)_{wrNAF}$.

The average density of the non-zero digit of wrNAF is $\frac{r-1}{w(r-1)+1}$ for non-small n and $n \gg r, w$.

4 Precise Efficiency of Radix-*r* gNAF and *wr*NAF

In this section we analyze precisely the behavior of Standard Radix-r Method with a scalar using radix-r gNAF or wrNAF.

As we remarked previous section, every integer can be uniquely represented using gNAF or wrNAF. The hamming weight of gNAF or wrNAF representation is minimal among all signed representations using digit set Γ_r or $D_{w,r}$. We want to estimate the average significant length and the average number of non-zero digits for gNAF and wrNAF. The following notations are used in this section. We assume that the radix-r gNAF or wrNAF is generated from the randomly chosen radix-r representation of n digits, i.e. the range of original input integer d is $0 \leq d < r^n$.

- Let $\mathcal{L}_g(r,n)$ and $\mathcal{W}_g(r,n)$ be the average significant length and the average number of non-zero digits for gNAF converted from the radix-r representation of n digits, respectively.
- Let $\mathcal{L}_r(r,n,w)$ and $\mathcal{W}_r(r,n,w)$ be the average significant length and the average number of non-zero digits for width-w rNAF converted from the radix-r representation of n digits, respectively.

If we compute the scalar multiplication using Standard Radix-r Method of gNAF (or wrNAF), then the number of underlying elliptic curve addition and r-th powering is equal to $\mathcal{W}_g(r,n)-1$ (or $\mathcal{W}_r(r,n,w)-1$) and $\mathcal{L}_g(r,n)-1$ (or $\mathcal{L}_r(r,n,w)-1$), respectively. Cohen investigated some approximation formulae of the width-w NAF, namely $\mathcal{L}_r(2,n,w)$ and $\mathcal{W}_r(2,n,w)$ with restricted condition $2^{n-1} \leq d < 2^n$ [Coh05].

We present other approximation formulae for general r, i.e to the arbitrary radix r. We can prove the following propositions. Their proofs can be found in the appendix.

Proposition 1. *The average significant length and the average number of non-zero digits for gNAF converted from the radix-r representation of* n *digits are*

$$\mathcal{L}_g(r,n) = n - \frac{1}{(r-1)(r+1)},$$
$$\mathcal{W}_g(r,n) = n\frac{r-1}{r+1} + \frac{r^2+5r-4}{r(r+1)^2},$$

for non-small n *and* $n \gg r$.

Proposition 2. *The average significant length and the average number of non-zero digits for width-w rNAF converted from the radix-r representation of* n *digits are*

$$\mathcal{L}_r(r,n,w) = n - \frac{w(w-2)(r-1)^2 + (w-1)(r-1) + 1}{2(w(r-1)+1)(r-1)},$$
$$\mathcal{W}_r(r,n,w) = n\frac{r-1}{w(r-1)+1} + \frac{w^2(r-1)^2 + w(3(r-1)) - (r-1) + 1}{2(w(r-1)+1)^2},$$

for non-small n *and* $n \gg r, w$.

In order to confirm the estimation in the propositions above we show simulation results for the practically used security level, namely 160 bits. We randomly generate one million radix-r integers with digit length $160, 101, 60, 57, 46$ for $r = 2, 3, 5, 7, 11$, respectively. In our experiment each digit of the radix-r representation is uniformly distributed, namely with probability $1/r$. Table 1 and Table 2 demonstrate that the theoretical estimations match the simulation results for the practical security size.

	$r=2, n=160$	$r=3, n=101$	$r=5, n=69$	$r=7, n=57$	$r=11, n=46$
$\mathcal{L}_g(r,n)$	159.67	100.88	68.96	56.98	45.99
Simulation	159.67	100.88	68.96	56.98	45.99
$\mathcal{W}_g(r,n)$	53.89	50.92	46.26	42.93	38.44
Simulation	53.77	50.88	46.27	42.97	38.48

Table 1. Experiment for gNAF

	$r=2, n=160$	$r=3, n=101$	$r=5, n=69$	$r=7, n=57$	$r=11, n=46$
$\mathcal{L}_r(r,n,2)$	159.67	100.85	68.93	56.96	45.97
Simulation	159.67	100.85	68.93	56.96	45.97
$\mathcal{W}_r(r,n,2)$	53.89	40.94	31.19	26.83	22.42
Simulation	53.77	40.86	31.14	26.79	22.39
$\mathcal{L}_r(r,n,3)$	159.25	100.39	68.45	56.47	45.48
Simulation	159.25	100.39	68.45	56.46	45.53
$\mathcal{W}_r(r,n,3)$	40.56	29.40	21.75	18.52	15.35
Simulation	40.44	29.32	21.71	18.48	15.34
$\mathcal{L}_r(r,n,4)$	158.80	99.92	67.96	55.98	44.99
Simulation	158.80	99.92	67.95	55.93	45.26
$\mathcal{W}_r(r,n,4)$	32.56	22.98	16.76	14.19	11.73
Simulation	32.44	22.91	16.71	14.15	11.78
$\mathcal{L}_r(r,n,5)$	158.33	99.43	67.47	55.48	44.49
Simulation	158.33	99.43	67.57	55.72	44.81
$\mathcal{W}_r(r,n,5)$	27.22	18.90	13.66	11.54	9.53
Simulation	27.11	18.83	13.64	11.57	9.58
$\mathcal{L}_r(r,n,6)$	157.86	98.94	66.98	54.98	43.99
Simulation	157.86	98.93	67.24	55.64	43.75
$\mathcal{W}_r(r,n,6)$	23.41	16.07	11.56	9.75	8.05
Simulation	23.31	16.01	11.57	9.84	7.99

Table 2. Experiment for wrNAF

5 Security Analysis of General rSD Representation

In this section we show and prove some classification formulae of equivalent classes called as Cutting Lemma, Collision Lemma, and Search Space Theorem, and investigate the security of Standard Radix-r Method using rSD representation against side channel attacks.

5.1 Summary of Notations

In this section we define some notations used for this paper. Assume that d is an original input integer, which ranges uniformly among the integers having n digits, in other words $0 \le d < r^n$. We denote it as $(d_{n-1}, \ldots, d_0)_r$.

- $\varepsilon = (\varepsilon_n, \ldots, \varepsilon_0)_{rSD}$: a recoded $(n{+}1)$-digit rSD representation of d.
- $\varepsilon^{(i)} := (\varepsilon_n^{(i)}, \ldots, \varepsilon_0^{(i)})_{rSD}$: the i-th[1] $(n{+}1)$-digit rSD representation of d. Say $\varepsilon^{(1)} = \varepsilon$. Clearly $d = \varepsilon^{(1)} = \varepsilon^{(2)} = \cdots = \varepsilon^{(i)} = \cdots$.
- $\varepsilon^{(i)}(s,t) := (\varepsilon_t^{(i)}, \varepsilon_{t-1}^{(i)}, \ldots, \varepsilon_{s+1}^{(i)}, \varepsilon_s^{(i)})_{rSD}$ where $0 \le s \le t \le n$. $s = t$ implies $\varepsilon^{(i)}(s,t)$ is the s-th digit $\varepsilon_s^{(i)}$ of $\varepsilon^{(i)}$. Thus we can see that $\varepsilon^{(i)} = \varepsilon^{(i)}(t,n) \cdot r^t + \varepsilon^{(i)}(0, t-1)$.
- N : set $\{0, 1, ..., n\}$.
- $\mathcal{Z}_{\varepsilon^{(i)}} = \{j \in N | \varepsilon_j^{(i)} = 0\}$: set of indices whose digit is zero in $\varepsilon^{(i)}$.
- $\mathcal{Z}^c_{\varepsilon^{(i)}} = N - \mathcal{Z}_{\varepsilon^{(i)}} = \{j \in N | \varepsilon_j^{(i)} \ne 0\}$: set of indices whose digit is non-zero in $\varepsilon^{(i)}$.
- $[\varepsilon^{(i)}]_\star$: $\star$ representation of $\varepsilon^{(i)}$ such that the information of digit whose index is contained in $\mathcal{Z}^c_{\varepsilon^{(i)}}$ is hidden by a symbol $\star$.
- $[\varepsilon^{(i)}]_\star^{List}$: a set of all possible rSD representations from $[\varepsilon^{(i)}]_\star$. Thus $\#[\varepsilon^{(i)}]_\star^{List} = \{2(r-1)\}^{\#\mathcal{Z}^c_{\varepsilon^{(i)}}}$. Here, for a set A, $\#A$ denotes the number of the elements in A.

Example Consider original radix-3 representation $d = (1, 0, 2, 0, 2, 0, 2)_3$ with 7 digits. $\varepsilon = (0, 1, 0, 2, 1, 0, \overline{2}, \overline{1})$ is one of the 3SD representations of d. Then $\mathcal{Z}_\varepsilon = \{2, 5, 7\}$ and $\mathcal{Z}^c_\varepsilon = \{0, 1, 3, 4, 6\}$. Then $[\varepsilon]_\star$=(0,$\star$,0,$\star$,$\star$,0,$\star$,$\star$). As the symbol $\star \in \{\pm 1, \pm 2\}$, $[\varepsilon]_\star^{List} = \{\pm 505, \pm 506, \ldots, \pm 1682\}$ and $\#[\varepsilon]_\star^{List} = 4^{\#\mathcal{Z}^c_\varepsilon} = 4^5 = 1024$.

5.2 Previous Results of BSD representation

In this section we introduce the previous results of side channel attacks (SCA) on some countermeasures using Binary Signed Digit (BSD) representation in elliptic curve cryptosystems (ECC). In ECC, a dominant computation is a scalar multiplication and an attacker's goal is to detect the secret scalar during scalar multiplication. Thus constructing an efficient computation method of scalar multiplication secure against SCA on ECC and analyzing its security are important research topics.

For this purpose, many countermeasures against SCA have been proposed. In particular, the countermeasure that utilizes several BSD representations of the secret scalar is a popular one. Its digit set is $\{0, \pm 1\}$. This type of countermeasure encodes the secret integer into BSD representation, then computes the scalar multiplied point using the representation. In addition, a different representation is used for each scalar multiplication. This thwarts the attacker's guess. This type of countermeasures includes Ha-Moon's countermeasure [HM02], Ebeid-Hasan's countermeasure [EH03], and the countermeasure of Agagliate et al [AGO03]. Note that the BSD representation is a special case of rSD representation when $r = 2$.

Whereas many countermeasures using BSD representations were proposed, unfortunately most of them have been broken under following assumptions.

[1] The order of rSD representation has no special meaning. We just give indexes to rSD representations of d for simplicity of the description of the following proposed properties.

1. Scalar multiplication dP is computed by using $\mathsf{Standard\_Radix\text{-}}r\mathsf{\_Method}$ where $r = 2$ and digit set $\mathcal{D} = \{0, \pm 1\}$. First the original input scalar d is recoded into BSD representation $\varepsilon^{(i)}$ and then $\varepsilon^{(i)}P$ is computed. Of course, it is also possible that the recoding and scalar multiplication are both operated simultaneously. Actually $dP = \varepsilon^{(i)}P$.
2. Elliptic addition and doubling are distinguishable by a single measurement of power consumption, whereas elliptic addition and subtraction are indistinguishable.
3. We can repeat to obtain the measurement of power consumption at the same device, all using same secret key d.
4. We know the plaintext-ciphertext pair (P, dP).

We call some countermeasures satisfying the first assumption above as **BSD type countermeasure**. Han et al. [HOK$^+$04] proposed sophisticated simple power analysis (SPA) against BSD type countermeasure. Since their sophisticated attacks are based on SPA, BSD type countermeasure should be combined with some SPA countermeasures such as Coron's dummy method [Cor99]. However, Fouque et al. [FMPV04] and Sim et al. [SPL04] proposed a collision-based attack and a usual differential power analysis (DPA), respectively, against BSD type countermeasure with dummy operation as a SPA countermeasure. Therefore, the security of BSD type countermeasure is very controversial.

Han et al.'s analysis utilized a relation among BSD representations. Even though Fouque et al. and Sim et al.'s analysis techniques are totally different, their main idea used in their analysis was exactly same. That is a relation between BSD representations and an original input integer.

Thus it is very natural motivation to extend those properties generated from BSD representation, i.e. 2SD representation, to rSD representation, for arbitrary prime r. Because if we can prove those properties used in the analysis of BSD type countermeasure is extended to rSD representation, then we can show that the security of rSD type countermeasure is also controversial.

5.3 Some Invariant Properties of rSD representation

We first propose some relations among rSD representations. Assume that $d = (d_{n-1}, \ldots, d_0)_r$ is an original input integer, which ranges uniformly among the integers having n digits, and $\varepsilon^{(1)}$ and $\varepsilon^{(2)}$ are rSD representations of d. Clearly $d = \varepsilon^{(1)} = \varepsilon^{(2)}$.

Lemma 1 **(*Cutting Lemma*).** *For* $t \in (\mathcal{Z}_{\varepsilon^{(1)}} \cap \mathcal{Z}_{\varepsilon^{(2)}}) \backslash \{0, n\}$ *or* $t \in (\mathcal{Z}^c_{\varepsilon^{(1)}} \cap \mathcal{Z}^c_{\varepsilon^{(2)}}) \backslash \{0\}$, *we have*

$$\varepsilon^{(1)}(0, t-1) = \varepsilon^{(2)}(0, t-1) \text{ and } \varepsilon^{(1)}(t, n) = \varepsilon^{(2)}(t, n).$$

By using this lemma we can cut rSD representations into two parts such that each left and right part are all same integers if the digits of cutting position are all zero or all non-zero. We call this as *Cutting Lemma* according to the digits of cutting position. It is an extended generalization of Proposition 1 in [HOK$^+$04] which is just *Non-Zero Cutting Lemma* when the radix $r = 2$.

We can make a lot of rSD representations from a given integer d represented by radix-r. However, there is an invariant relation between a rSD and the original radix-r representation. Note that $\varepsilon^{(i)}$ is the i-th rSD representation of d. Then we can prove the following lemma for any index $0 \le t \le n-1$:

Lemma 2 **(*Collision Lemma*).** *For any* i, $\varepsilon^{(i)}(t, n)$ *is either* $d(t, n-1)$ *or* $d(t, n-1)+1$. *This implies* $\varepsilon^{(i)}(0, t)$ *is either* $d(0, t)$ *or* $d(0, t) - r^{t+1}$.

This lemma shows that the collection of $\{\varepsilon^{(i)}(t,n) \mid 1 \le i \le m\}$ for any integer m is separated into two parts which have the same value $d(t,n-1)$ and $d(t,n-1)+1$. In other words, for any i, the value of $\varepsilon^{(i)}(t,n)$ is exactly same to one of $\{d(t,n-1), d(t,n-1)+1\}$. We call it as *Collision Lemma*. The case when radix $r = 2$ was discussed in [FMPV04,SPL04].

Motivation: Let $d = (1,0,2,0,2,0,2)_3$ be an original radix-3 representation with 7 digits. Consider following three 3SD representations; $\varepsilon^{(1)} = (0,1,1,\overline{1},0,2,0,2)_{3SD}$, $\varepsilon^{(2)} = (1,\overline{2},0,2,1,0,\overline{2},\overline{1})_{3SD}$, and $\varepsilon^{(3)} = (1,\overline{2},1,0,\overline{2},0,\overline{2},\overline{1})_{3SD}$. Then

$$[\varepsilon^{(1)}]_\star = (0,\star,\star,\star,0,\star,0,\star) \;\; \textit{and} \;\; \mathcal{Z}^c_{\varepsilon^{(1)}} = \{0,2,4,5,6\},$$

$$[\varepsilon^{(2)}]_\star = (\star,\star,0,\star,\star,0,\star,\star) \;\; \textit{and} \;\; \mathcal{Z}^c_{\varepsilon^{(2)}} = \{0,1,3,4,6,7\},$$

$$[\varepsilon^{(3)}]_\star = (\star,\star,\star,0,\star,0,\star,\star) \;\; \textit{and} \;\; \mathcal{Z}^c_{\varepsilon^{(3)}} = \{0,1,3,5,6,7\}.$$

Assume that we only have the information of $[\varepsilon^{(1)}]_\star$, $[\varepsilon^{(2)}]_\star$, and $[\varepsilon^{(3)}]_\star$. What we want to do is to find the integer d with those information $[\varepsilon^{(i)}]_\star$. The simplest way is as follows; first calculate $\cap_{i=1}^3[\varepsilon^{(i)}]_\star^{List} = \{\pm547, \pm548, \pm910, \pm911, \pm1276, \pm1277, \pm1639, \pm1640\}$ and then check all elements in $\cap_{i=1}^3[\varepsilon^{(i)}]_\star^{List}$. The search space order is 16.

We want to find a relation between $\#\big(\cap_{i=1}^3[\varepsilon^{(i)}]_\star^{List}\big)$ and the information of $\mathcal{Z}_{\varepsilon^{(i)}}$ or $\mathcal{Z}^c_{\varepsilon^{(i)}}$ for $1 \le i \le 3$. Interestingly

$$\#(\cap_{i=1}^3[\varepsilon^{(i)}]_\star^{List}) = \{2(r-1)\}^{\#[\cap_{i=1}^3 \mathcal{Z}^c_{\varepsilon^{(i)}}]} = 4^2 = 16$$

because $\cap_{i=1}^3 \mathcal{Z}^c_{\varepsilon^{(i)}} = \{0,6\}$ and $r = 3$.

We can prove that the search space order can be calculated with the information of radix-r and $\mathcal{Z}^c_{\varepsilon^{(i)}}$.

Theorem 1 (*Search Space Theorem*). *For given m rSD representations of d, if $x = \#(\cap_{i=1}^m \mathcal{Z}^c_{\varepsilon^{(i)}})$, then the search space order l is as follows:*

$$l = \#(\cap_{i=1}^m[\varepsilon^{(i)}]_\star^{List}) = \{2(r-1)\}^x.$$

This theorem show that we can find the search space order without actual constructing $[\varepsilon^{(i)}]_\star^{List}$, all possible rSD representations of $[\varepsilon^{(i)}]_\star$. We can drastically reduce the required memory of the search space if *Cutting Lemma* is utilized. Its detail description is contained in the next section. It is a generalization of Theorem 1 in [HOK+04] which is the case that radix $r = 2$.

5.4 Side Channel Attacks on rSD Type Countermeasure

In this section we briefly describe the main idea of generic analysis technique according to the target algorithms. Assume that $d = (d_{n-1}, \ldots, d_0)_r$ is an original input integer, i.e. secret key, which ranges uniformly among the integers having n digits. $\varepsilon^{(i)} = (\varepsilon_n^{(i)}, \ldots, \varepsilon_0^{(i)})_{rSD}$ is the i-th rSD representation of d

Sophisticated SPA on Standard_Radix-r_Method with rSD representation: To make the proposed analysis success we need some assumptions like as the analysis of BSD type countermeasure described in Section 5.2. We call some countermeasure as *rSD type countermeasure* if the first assumption below is satisfied.

1. Scalar multiplication dP is computed by using Standard_Radix-r_Method where the digit set $\mathcal{D} = \Gamma_r$. First the original input scalar d is recoded into rSD representation $\varepsilon^{(i)}$ and then $\varepsilon^{(i)}P$ is computed.

2. Elliptic addition and r-th powering are distinguishable by a single measurement of power consumption, whereas elliptic addition and subtraction are indistinguishable.
3. We can repeat to obtain the measurement of power consumption at the same device, all using same secret key d.
4. We know the plaintext-ciphertext pair (P, dP).

Remark 1. Since the standard implementation of elliptic addition is different from r-th powering [SW02] and elliptic subtraction of (Q, P) is usually implemented as elliptic addition of $(Q, -P)$, the second assumption is realistic. The third assumption is applicable to Boneh-Lynn-Shacham short signature scheme [BLS01] because the same secret key is used in the generation of signature.

Under Assumption 3, an attacker obtains recoded m rSD representations, say $\{\varepsilon^{(i)} \mid 1 \le i \le m\}$, generated from the original input integer d. Under Assumption 1 and 2, he/she can know the position of zero digit of $\varepsilon^{(i)}$, in other words he/she can determine $\varepsilon_j^{(i)}$ is zero or not, where $0 \le j \le n$.

Attack Algorithm based on *Cutting Lemma*

1. Find $[\varepsilon^{(i)}]_\star$ and $\mathcal{Z}^c_{\varepsilon^{(i)}}$ for $1 \le i \le m$: under above assumptions an attacker can determine $\star$ representation and the set of indices whose digit is non-zero in $\varepsilon^{(i)}$.
2. Novel method for computing $\cap_{i=1}^m [\varepsilon^{(i)}]_\star^{List}$: this step is to find all possible keys using *Cutting Lemma*.
   2.1. Compute $\cap_{i=1}^m \mathcal{Z}^c_{\varepsilon^{(i)}}$. Assume $\cap_{i=1}^m \mathcal{Z}^c_{\varepsilon^{(i)}} = \{t_1, \ldots, t_x\}$ where $0 \le t_1 \le \ldots \le t_x \le n$.
   2.2. By using *Cutting Lemma*, he/she splits $[\varepsilon^{(i)}]_\star$ into x parts, those are as follows;
   - $[\varepsilon^{(i)}(t_j, t_{j+1}-1)]_\star$ if $1 \le j \le x-1$,
   - $[\varepsilon^{(i)}(t_j, n)]_\star$ if $j = x$.

   2.3. For $j = 1$ to x make
   - $\cap_{i=1}^m [\varepsilon^{(i)}(t_j, t_{j+1}-1)]_\star^{List}$ for $1 \le j \le x-1$,
   - $\cap_{i=1}^m [\varepsilon^{(i)}(t_j, n)]_\star^{List}$ for $j = x$.

   2.4. Re-construct $\cap_{i=1}^m [\varepsilon^{(i)}]_\star^{List}$ with the results of split parts of Step 2.3.
3. Key testing: Using the known pair of plaintext and ciphertext, the attacker checks all possible keys obtained from Step 2.4.

The proposed attack method can be simplified by directly calculating $\cap_{i=1}^m [\varepsilon^{(i)}]_\star^{List}$ without using *Cutting Lemma*, i.e. the exhaustive search method (ESM). In this case to compute $\cap_{i=1}^m [\varepsilon^{(i)}]_\star^{List}$ on average we require a memory for $(n+1) \cdot \left\{\{2(r-1)\}^{\frac{r-1}{r}(n+1)} \cdot m\right\}$ length digits under assumption the average density of the non-zero digit of rSD representation of $n{+}1$ digits is equal to $(r-1)/r$. But if we use *Cutting Lemma* the required memory is reduced to on average $(\frac{n+1}{x}) \cdot \left\{\{2(r-1)\}^{\frac{r-1}{r}\frac{n+1}{x}} \cdot x \cdot m\right\}$ under assumption that

- $\#\mathcal{Z}^c_{\varepsilon^{(i)}}(t_j, t_{j+1}-1) = \frac{r-1}{r} \cdot \frac{n+1}{x}$ and $t_{j+1} - t_j = \frac{n+1}{x}$ for $1 \le j \le x-1$,
- $\#\mathcal{Z}^c_{\varepsilon^{(i)}}(t_j, n) = \frac{r-1}{r} \cdot \frac{n+1}{x}$ and $n - t_x + 1 = \frac{n+1}{x}$ for $j = x$,

in other words we assume each separated part has same length and same hamming weight. Roughly the memory complexity of Step 2. of the proposed algorithm is x-th root of that of ESM. The exact time complexity of Step 3. can be derived from the result of Theorem 1. Note that as the non-zero most significant digit is positive in the application of cryptography the search space of the last block $\cap_{i=1}^m [\varepsilon^{(i)}(t_x, n)]_\star^{List}$ can be reduced to half.

Analysis of r*SD type countermeasure* with SPA countermeasure: We assume that the Coron's dummy method [Cor99] or indistinguishable type operation method

[BJ02] is used as SPA countermeasure. Suppose an attacker already knows the highest bits $d_{n-1}, \ldots, d_{t+1}$ of the secret key d. His/her goal is to recover the next bit d_t.

From *Collision Lemma*, the point which is actually calculated at the Step 3.1. after $j = t$ digit (d_t) calculation for the r-th execution is as follows;

$$
\begin{array}{lcll}
\big(r^2 \cdot d(t+1, n-1)\big) \cdot P & \text{or} & \big(r^2 \cdot d(t+1, n-1) + r\big) \cdot P & \text{if } d_t = 0, \\
\big(r^2 \cdot d(t+1, n-1) + r\big) \cdot P & \text{or} & \big(r^2 \cdot d(t+1, n-1) + 2r\big) \cdot P & \text{if } d_t = 1, \\
\vdots & & \vdots & \vdots \\
\big(r^2 \cdot d(t+1, n-1) + l \cdot r\big) \cdot P & \text{or} & \big(r^2 \cdot d(t+1, n-1) + (l+1) \cdot r\big) \cdot P & \text{if } d_t = l, \\
\vdots & & \vdots & \vdots \\
\big(r^2 \cdot d(t+1, n-1) + (r-1) \cdot r\big) \cdot P & \text{or} & \big(r^2 \cdot d(t+1, n-1) + r^2\big) \cdot P & \text{if } d_t = r-1,
\end{array}
$$

Thus, the number of possible output after Step 3.1 is always two. It is a drawback of rSD representation. This property of $r = 2$ was used as main idea in the attack algorithms on BSD type countermeasure with dummy operation [FMPV04,SPL04]. Therefore we can conclude that the security of rSD type countermeasure with SPA countermeasure is controversial.

6 A Radix-r SPA Countermeasure with Fixed Pattern

In the previous section, we discussed the security of *rSD type countermeasure* which uses rSD representation as a countermeasure against SCA. But, unfortunately it was not secure against even SPA. Also the security of *rSD type countermeasure* with SPA countermeasure was controversial. Thus in order to make SCA-resistant scalar multiplication dP for pairing-based cryptosystems which is suitable to scarce computational resources devices, we need efficient secure countermeasure against SCA.

In ECC with characteristic 2, it is believed that Okeya-Takagi countermeasure against SPA is an optimal scheme in the trade-off between memory and efficiency [OT03]. The main idea is to generate a fixed pattern of signed radix-2 digits for a given integer. Thus the power consumption is independent from the secret bit information, and thus SPA becomes infeasible. As the immunity against DPA is easily obtained by combining with countermeasure of the data randomization technique such as randomized projective coordinates method [Cor99] we mainly consider the countermeasure against SPA in this section. Note that Okeya-Takagi countermeasure requires no dummy operation.

We explain how to extend Okeya-Takagi scheme to the radix-r representation in the following. We scan from the least significant digits. If we find a zero digit, the following conversion is performed: $(\underbrace{0, ..., 0}_{k}, x)_r = (1, \underbrace{\overline{r-1}, ..., \overline{r-1}}_{k-1}, \overline{r-x})_{rSD}$ for positive integer k and $x = 1, 2, ..., r-1$. If we apply the width-w right-to-left sliding window method to this chain, we obtain the radix-r analogue of Okeya-Takagi scheme, which has the fixed pattern

$$|\underbrace{0, ..., 0}_{w-1}, y|\underbrace{0, ..., 0}_{w-1}, y|...|\underbrace{0, ..., 0}_{w-1}, y| \qquad (1)$$

for $y \in \{\pm 1, \pm 2, ..., \pm(r^w - 1)\} \setminus \{\pm r, \pm 2r, ..., \pm(r^w - r)\}$. The size of non-trivial digits is equal to $(r-1)r^{w-1}$, and this size is minimal among all signed radix-r representations with fixed pattern $(\underbrace{0, ..., 0}_{w-1}, z)$ for some z.

The generation algorithm of this fixed pattern is as follows:

Integer to radix-r Fixed Pattern Representation

Input: integer d ($d \bmod r \neq 0$), radix r, width w

Output: the radix r fixed pattern representation of d: $= (..., \gamma_1, \gamma_0)$.
1. $i \leftarrow 0$
2. While $d > 1$ do the following
 2.1. $\gamma_{i+w-1} \leftarrow 0, ..., \gamma_{i+1} \leftarrow 0$, and $v \leftarrow d \bmod r^w$
 2.1.1. if $v < r^w/2$ then $\gamma_i \leftarrow v$,
 2.1.2. else $\gamma_i \leftarrow v - r^w$,
 2.2. $d \leftarrow (d - \gamma_i)/r^w$,
 2.3. if $d \bmod r = 0$ then $\gamma_i \leftarrow \gamma_i - r^w, d \leftarrow d + 1$
 2.4. $i \leftarrow i + w$
3. Return $(..., \gamma_1, \gamma_0)$.

Note that this algorithm works only for the integer d which is not divisible by r. If $d \bmod r = 0$ holds, we need an additional treatment in order to achieve a fixed pattern, e.g., randomization by addition a multiplier of the order of underlying elliptic curve.

Acknowledgements

Dong-Guk Han was supported by the Korea Research Foundation Grant. (KRF-2005-214-C00016)

References

[AGO03] S. Agagliate, P. Guillot, O. Orcière, "A Randomized Efficient Algorithm for DPA Secure Implementation of elliptic curve Cryptosystems," *WCC 2003*, pp.11-19, 2003.

[BF01] D. Boneh and M. Franklin, "Identity Based Encryption from the Weil Pairing," *SIAM J. of Computing*, Vol. 32, No. 3, pp.586-615, 2001.

[BGK+03] G. Bertoni, J. Guajardo, S. Kumar, G. Orlando, C. Paar, and T. Wollinger, "Efficient $GF(p^m)$ Arithmetic Architectures for Cryptographic Applications," *CT-RSA 2003*, LNCS 2612, pp.158-175, 2003.

[BJ02] E. Brier, M. Joye, "Weierstrass Elliptic Curves and Side-Channel Attacks," *PKC 2002*, LNCS 2274, pp.335-345, 2002.

[BKL+02] P. Barreto, H. Kim, B. Lynn, and M. Scott, "Efficient Algorithms for Pairing-Based Cryptosystems," *CRYPTO 2002*, LNCS 2442, pp.354-368, 2002.

[BGhE+04] P. Barreto, S. Galbraith, C. hEigeartaigh, and M. Scott, "Efficient Pairing Computation on Supersingular Abelian Varieties," Cryptology ePrint Archive: Report 2004/375, 2005.

[BLS01] D. Boneh, B. Lynn, and H. Shacham, "Short Signatures from the Weil Pairing," *ASIACRYPT 2001*, LNCS 2248, pp.514-532, 2001.

[Coh05] H. Cohen, "Analysis of the Sliding Window Powering Algorithm," Journal of Cryptology, 18(1), pp.63-76, 2005.

[Cor99] J.S. Coron, "Resistance against Differential Power Analysis for Elliptic Curve Cryptosystems," *CHES 1999*, LNCS1717, pp.292-302, 1999.

[CL73] W. Clark and J. Liang, "On Arithmetic Weight for a General Radix Representation of Integers," *IEEE Transaction on IT*, IT-19, pp.823-826, 1973.

[DL03] I. Duursma and H -S .Lee, "Tate Pairing Implementation for Hyperelliptic Curves $y^2 = x^p - x + d$," *ASIACRYPT 2003*, LNCS 2894, pp.111-123, 2003.

[EH03] N. Ebeid and M. Hasan, "On Randomizing Private Keys to Counteract DPA Attacks," *SAC 2003*, LNCS 3006, pp.58-72, 2004.

[FMPV04] P.A. Fouque, F. Muller, G. Poupard, and F. Valette, "Defeating Countermeasures Based on Randomized BSD Representations," *CHES 2004*, LNCS 3156, pp.312-327, 2004.

[GHS02] S. Galbraith, K. Harrison, and D. Soldera, "Implementing the Tate pairing," *ANTS V*, LNCS 2369, pp.324-337, Springer-Verlag, 2002.

[HM02] J. -C. Ha, S. -J. Moon, "Randomized Signed-Scalar Multiplication of ECC to Resist Power Attacks," *CHES 2002*, LNCS 2523, pp.551-563, 2002.

[HOK+04] D. -G. Han, K. Okeya, T.H. Kim, Y.S. Hwang, Y. -H. Park, S. Jung, "Cryptanalysis of the Countermeasures Using Randomized Binary Signed Digits," *ACNS 2004*, LNCS 3089, pp. 398-413, 2004.

[HPS02] K. Harrison, D. Page, and N. Smart, "Software Implementation of Finite Fields of Characteristic Three," *LMS Journal of Computation and Mathematics*, Vol.5, pp.181-193, 2002

[Jou00] A. Joux, "A one round protocol for tripartite Diffie-Hellman," *ANTS V*, LNCS 1838, pp.385-394, 2000.

[OT03] K. Okeya and T. Takagi, "The Width-w NAF Method Provides Small Memory and Fast Elliptic Scalar Multiplications Secure against Side Channel Attacks," *CT-RSA 2003*, LNCS2612, pp.328-342, 2003.

[PS02] D. Page and N. Smart, "Hardware Implementation of Finite Fields of Characteristic Three," *CHES 2002*, LNCS 2523, pp.529-539, 2002.

[PV04] D. Page and F. Vercauteren, "Fault and Side-Channel Attacks on Pairing Based Cryptography," *Cryptology ePrint Archive*, Report 2004/283, 2004.

[SPL04] S.G. Sim, D.J. Park, P.J. Lee, "New power analyses on the Ha-Moon algorithm and the MIST algorithm," *ICICS 2004*, LNCS3269, pp. 291-304, 2004.

[SW02] N. Smart, and J. Westwood, "Point Multiplication on Ordinary Elliptic Curves over Fields of Characteristic Three," *Applicable Algebra in Engineering, Communication and Computing*, Vol.13, No.6, pp.485-497, 2003.

[TYW04] T. Takagi, S.-M. Yen, B.-C. Wu, "Radix-r Non-Adjacent Form," ISC 2004, LNCS 3225, pp.99-110, 2004.

A Some Proofs of Section 4

Before starting the proof, we define some notations and describe some properties of the radix-r gNAF recoding algorithm.

Notation: Let $(v_1, v_0)_j$ be the j-th input of Step 2.1. in the radix-r gNAF recoding algorithm. Here, $v_0 = v \bmod r$, $v_1 = (v - v_0)/r$, where $v = d \bmod r^2$ in Step 2.1.

Assume that with input $\{(v_1, v_0)_{j-1}, \ldots, (v_1, v_0)_0\}$ all i digits $(\gamma_{i-1}, \ldots, \gamma_0)$ are determined. Thus the next recoded digit(s) γ_i or (γ_{i+1}, γ_i) is determined according to the condition of $(v_1, v_0)_j$. From the radix-r gNAF recoding algorithm we can derive the following property.

Property 1. The result of Step 2.1.3.1. and 2.1.4. can cause a carry $+1$ or carry propagation to the next step input $(v_1, v_0)_{j+1}$. However, only the result of Step 2.1.4. can cause an $(n+1)$-digit recoded number from an n-digit input integer.

In more detail, if $1 \le v_2 \le r-2$ and $v_2 + v_1 \ge r$ then Step 2.1.3.1. is operated and the recoded digit $\gamma_i = v_0 - r$ with a carry $+1$ to the next digit v_1. Because of the required condition of Step 2.1.3.1. if Step 2.1.3.1. is the final step of the recoding algorithm then $(n+1)$-digit length can not be occurred. In the case that $(v_1, v_0)_j = (r-1, x)$ where $x \in \{1, \ldots, r-1\}$ then Step 2.1.4. is operated. Its recoded outputs are $(\gamma_{i+1}, \gamma_i) = (0, x-r)$ and a carry $+1$ influences on the next input digit v_0 of $(v_1, v_0)_{j+1}$. In this case, if v_0 is $r-1$ then the previous carry $+1$ causes a carry propagation, i.e., $(v_1, v_0)_{j+1} = (v_1 + 1, 0)_{j+1}$ with a carry $+1$ to v_1.

Proposition 1. *The average significant length and the average number of non-zero digits for gNAF converted from the radix-r representation of n digits are*

$$\mathcal{L}_g(r,n) = n - \frac{1}{(r-1)(r+1)},$$
$$\mathcal{W}_g(r,n) = n\frac{r-1}{r+1} + \frac{r^2+5r-4}{r(r+1)^2},$$

for non-small n.

Proof. From the assumption, each digit of the original n-digit radix-r representation distributes (i) for $i = 0, 1, ..., r-1$ with probability $1/r$.

Under the above assumption and notation, the next input digits $(v_1, v_0)_j$ of gNAF are classified to the following five categories.

- $(0)_n$: denotes $v_0 = 0$ and there is no carry $+1$ to the next digit v_1,
- $(r{-}1)_c$: denotes $v_0 = r{-}1$ and there is a carry $+1$ to the next digit v_1,
- $(0, x)_n$: denotes $v_1 = 0$ and $v_0 = x$, and there is no carry $+1$ to the next digit v_0 of the next step $(v_1, v_0)_{j+1}$,
- $(r{-}1, x)_c$: denotes $v_1 = r{-}1$ and $v_0 = x$, and there is a carry $+1$ to the next digit v_0 of the next step $(v_1, v_0)_{j+1}$,
- (x) : denotes $v_0 = x$, and there may or may not be a carry $+1$ to the next digit v_1.

Here, $x \in \Gamma_r^+$ where Γ_r^+ stands for the positive integers in Γ_r. The state transitions for the radix-r gNAF are as follows;

- $(0)_n$: the recoded digit $\gamma_i = 0$. If $v_1 = 0$ then go to state $(0)_n$, (n) otherwise.
- $(r{-}1)_c$: the recoded digit $\gamma_i = 0$. If $v_1 = r-1$ then go to state $(r{-}1)_c$, (n) otherwise.
- $(0, x)_n$: the recoded digit $(\gamma_{i+1}, \gamma_i) = (0, x)$. If $v_0 = 0$ of the next step $(v_1, v_0)_{j+1}$ then go to state $(0)_n$, (n) otherwise.
- $(r{-}1, x)_c$: the recoded digit $(\gamma_{i+1}, \gamma_i) = (0, x - r)$. If $v_0 = r - 1$ of the next step $(v_1, v_0)_{j+1}$ then go to state $(r{-}1)_c$, (n) otherwise.
- (x) : There are three cases;
  - If $v_1 = 0$ then go to the next state $(0, x)_n$.
  - If $v_1 = r - 1$ then go to the next state $(r - 1, x)_c$.
  - $v_1 \in \Gamma_r \backslash \{0, r-1\}$. If $v_1 + v_0 \geq r$ then the recoded digit $\gamma_i = x - r$ and go to the next state (n), else $(v_1 + v_0 < r)$ the recoded digit $\gamma_i = x$ and go to the next state (n).

Note that the type $(r{-}1, x)_c$ is generated after Step 2.1.4. and the type $(r{-}1)_c$ is generated when $(v_1, v_0)_{j-1}$ is calculated at Step 2.1.4. and the next digit $v_0 = r - 1$ in $(v_1, v_0)_j$.

Here we simulate the next input digits $(v_1, v_0)_j$ of gNAF as Markov chain with these five statuses. The transit matrix of the Markov chain is as follows:

$$\begin{pmatrix} (0)_n & : \frac{1}{r} & 0 & 0 & 0 & \frac{r-1}{r} \\ (r-1)_c & : 0 & \frac{1}{r} & 0 & 0 & \frac{r-1}{r} \\ (0,x)_n & : \frac{1}{r} & 0 & 0 & 0 & \frac{r-1}{r} \\ (r-1,x)_c & : 0 & \frac{1}{r} & 0 & 0 & \frac{r-1}{r} \\ (x) & : 0 & 0 & \frac{1}{r} & \frac{1}{r} & \frac{r-2}{r} \end{pmatrix},$$

The stationary distribution of each status for non-small n is as follows:

$$((0)_n, (r-1)_c, (0,x)_n, (r-1,x)_c, (x))) = \left(\frac{1}{r(r+1)}, \frac{1}{r(r+1)}, \frac{r-1}{r(r+1)}, \frac{r-1}{r(r+1)}, \frac{r-1}{r+1} \right).$$

Now we estimate $\mathcal{W}_g(r, n)$ in the following. From the assumption, the average non-zero density of lower bits from these statuses is $p(r) = \frac{r-1}{r+1}$ for large n. As the last statuses in recoding algorithm is one of above five statuses we can estimate $\mathcal{W}_g(r, n)$ which depends on the last utilized statuses. We then know that $\mathcal{W}_g(r, n) = (n-1)p(r)$ for statuses $(0)_n$. There is a carry $+1$ at statuses $(r-1)_c$ and the number of non-zero digit increases one, namely $\mathcal{W}_g(r, n) = 1 + (n-1)p(r)$. Status $(0, x)_n$ contains one non-zero digit and the lower $(n-2)$ digit has the density $p(r)$, namely $\mathcal{W}_g(r, n) = 1 + (n-2)p(r)$. Similarly we know $\mathcal{W}_g(r, n) = 2 + (n-2)p(r)$ and $\mathcal{W}_g(r, n) = 1 + (n-1)p(r)$ for status $(r-1, x)_c$ and

(x), respectively. From this observation we obtain the following relationship for non-small n.

$$\begin{aligned}\mathcal{W}_g(r,n) &= \left((n-1)\frac{r-1}{r+1}\right)\frac{1}{r(r+1)} + \left(1+(n-1)\frac{r-1}{r+1}\right)\frac{1}{r(r+1)} \\ &+ \left(1+(n-2)\frac{r-1}{r+1}\right)\frac{r-1}{r(r+1)} + \left(2+(n-2)\frac{r-1}{r+1}\right)\frac{r-1}{r(r+1)} \\ &+ \left(1+(n-1)\frac{r-1}{r+1}\right)\frac{r-1}{r+1} \\ &= n\frac{r-1}{r+1} + \frac{r^2+5r-4}{r(r+1)^2}.\end{aligned}$$

Next we estimate $\mathcal{L}_g(r,n)$. We know that $\mathcal{L}_g(r,n) = n+1$ for status $(r-1)_c$ and $(r-1,x)_c$, $\mathcal{L}_g(r,n) = n-1$ for status $(0,x)_n$, and $\mathcal{L}_g(r,n) = n$ for status (x), respectively.

In the case of $(0)_n$, we should consider the case that the most k bits of gNAF become consecutive zeros for $k = 2, 3, ..., n-1$. Define $\Pr[A \curvearrowright B]$ as the probability that B event occurred before the event A whenever A event occurred. For example, $\Pr[(0)_n \curvearrowright (x)]$ is zero and $\Pr[(0,x)_n \curvearrowright (x)] = 1$. From the proposed transit matrix of the Markov chain, we can see that $\Pr[(0)_n \curvearrowright (0)_n]+\Pr[(0)_n \curvearrowright (0,x)_n] = 1$ and $\Pr[(0)_n \curvearrowright (0)_n] = 1/r$. Clearly $\Pr[(0)_n \curvearrowright (0,x)_n] = (r-1)/r$. Thus depending on the number of consecutive zeros the probability that the most k bits of gNAF becomes consecutive zeros is as follows;

- $k = 2$: $\Pr[(0)_n]\cdot\Pr[(0)_n \curvearrowright (0,x)_n] = \frac{1}{r(r+1)} \cdot \frac{r-1}{r}$,
- $k = 3$: $\Pr[(0)_n]\cdot\Pr[(0)_n \curvearrowright (0)_n]\cdot\Pr[(0)_n \curvearrowright (0,x)_n] = \frac{1}{r(r+1)} \cdot \frac{1}{r} \cdot \frac{r-1}{r}$,
- $\cdots$
- $k = n-1$: $\Pr[(0)_n]\cdot\Pr[(0)_n \curvearrowright (0)_n]^{k-2}\cdot\Pr[(0)_n \curvearrowright (0,x)_n] = \frac{1}{r(r+1)} \cdot (\frac{1}{r})^{k-2} \cdot \frac{r-1}{r}$.

Thus $\mathcal{L}_g(r,n) = \sum_{k=2}^{n-1}(n-k)\left(\frac{1}{r}\right)^{k-2}\frac{r-1}{r}$ for status $(0)_n$.

Consequently we have obtained the following equation for non-small n.

$$\begin{aligned}\mathcal{L}_g(r,n) &= \left(\sum_{k=2}^{n-1}(n-k)\left(\frac{1}{r}\right)^{k-2}\frac{r-1}{r}\right)\frac{1}{r(r+1)} + (n+1)\frac{1}{r(r+1)} \\ &+(n-1)\frac{r-1}{r(r+1)} + (n+1)\frac{r-1}{r(r+1)} + n\frac{r-1}{r+1} \\ &= n - \frac{1}{(r-1)(r+1)}.\end{aligned}$$ □

The proof of Proposition 2 is similar to that of Proposition 1. We first define some notations. Let $(v_{w-1},\ldots,v_0)_j$ be the j-th input of Step 2.1. in the wrNAF recoding algorithm, i.e $v = v_{w-1}r^{w-1} + \ldots + v_1 r + v_0$. Assume that with input $\{(v_{w-1},\ldots,v_0)_{i-1},\ldots,(v_{w-1},\ldots,v_0)_0\}$ all i digits $(\delta_{i-1},\ldots,\delta_0)$ are determined. Thus the next recoded digit δ_i is determined according to the condition of $(v_{w-1},\ldots,v_0)_i$. The wrNAF recoding algorithm has the following property.

Property 2. Only the result of Step 2.1.3. can cause a carry +1 or carry propagation to the next step input $(v_{w-1},\ldots,v_0)_{i+1}$.

Thus if $(v_{w-1},\ldots,v_0)_i$ satisfies $v \geq r^w/2$ then $\delta_i = v - r^w$ and a carry +1 influences on the next input digit v_0 of $(v_{w-1},\ldots,v_0)_{i+1}$. In this case, if v_0 is $r-1$ then the previous carry +1 causes a carry propagation to v_1.

Proposition 2. *The average significant length and the average number of non-zero digits for width-w rNAF converted from the radix-r representation of n digits are*

$$\begin{aligned}\mathcal{L}_r(r,n,w) &= n - \frac{w(w-2)(r-1)^2+(w-1)(r-1)+1}{2(w(r-1)+1)(r-1)}, \\ \mathcal{W}_r(r,n,w) &= n\frac{r-1}{w(r-1)+1} + \frac{w^2(r-1)^2+w(3(r-1))-(r-1)+1}{2(w(r-1)+1)^2},\end{aligned}$$

for non-small n.

Proof. We give a sketch of the proof. In the generation algorithm of wrNAF the next input $(v_{w-1}, \ldots, v_0)_i$ are classified to the following $(w+3)$ statuses.

- $(0)_n$: denotes $v_0 = 0$ and there is no carry $+1$ to v_1,
- $(r-1)_c$: denotes $v_0 = r-1$ and there is a carry $+1$ to v_1,
- $(\underbrace{0...0x}_{w})_n$: denotes $v = v_{w-1}r^{w-1} + \ldots + v_0 \neq 0$, $v < r^w/2$, and there is no carry $+1$ to v_0 of the next step $(v_{w-1}, \ldots, v_0)_{i+1}$, actually $x = v$,
- $(\underbrace{0...0x}_{w})_c$: denotes $v = v_{w-1}r^{w-1} + \ldots + v_0 \neq 0$, $v \geq r^w/2$, and there is a carry $+1$ to v_0 of the next step $(v_{w-1}, \ldots, v_0)_{i+1}$, actually $x = v$,
- $(\underbrace{0...0y}_{w-1})_n$: denotes $v = v_{w-2}r^{w-2} + \ldots + v_0 \neq 0$ and there is no carry $+1$ to v_{w-1} of $(v_{w-1}, \ldots, v_0)_i$, actually $y = v$,
- $\vdots$
- $(\underbrace{\ 0y\ }_{2})_n$: denotes $v = v_1 r + v_0 \neq 0$ and there is no carry $+1$ to v_2 of $(v_{w-1}, \ldots, v_0)_i$, actually $y = v$,
- $(y)_n$: denotes $v = v_0 \neq 0$ and there is no carry $+1$ to v_1 of $(v_{w-1}, \ldots, v_0)_i$, actually $y = v$.

Here, $x \in D^+_{w,r}$ and $y \in D^+_{w,r}$ with $y < r^{w-1}$ where $D^+_{w,r}$ stands for the positive integers in $D_{w,r}$. Then the transit matrix of the Markov chain for these statuses is as follows:

$$\begin{pmatrix}
(0)_n : & \frac{1}{r} & 0 & 0 & 0 & 0 & . & . & . & \frac{r-1}{r} \\
(r-1)_c : & 0 & \frac{1}{r} & 0 & 0 & 0 & . & . & . & \frac{r-1}{r} \\
(\underbrace{0...0x}_{w})_n : & \frac{1}{r} & 0 & 0 & 0 & 0 & . & . & . & \frac{r-1}{r} \\
(\underbrace{0...0x}_{w})_c : & 0 & \frac{1}{r} & 0 & 0 & 0 & . & . & . & \frac{r-1}{r} \\
(\underbrace{0...0y}_{w-1})_n : & 0 & 0 & \frac{1}{2} & \frac{1}{2} & 0 & . & . & . & 0 \\
(\underbrace{0...0y}_{w-2})_n : & 0 & 0 & 0 & 0 & 1 & . & . & . & 0 \\
 & . & . & . & . & . & . & 1 & . & . \\
 & . & . & . & . & . & . & . & 1 & . \\
(y)_n : & 0 & 0 & 0 & 0 & 0 & . & . & 1 & 0
\end{pmatrix}.$$

The stationary distribution of each status is as follows: $\left(\frac{1}{2(w(r-1)+1)}, \frac{1}{2(w(r-1)+1)}, \frac{r-1}{2(w(r-1)+1)}, \frac{r-1}{2(w(r-1)+1)}, \frac{r-1}{w(r-1)+1}, \ldots, \frac{r-1}{w(r-1)+1}\right)$

The average non-zero density of lower bits from these statuses is $p(r) = \frac{r-1}{(w-1)r+1}$ for non-small n [TYW04]. Therefore, $\mathcal{L}_r(r,n,w)$ and $\mathcal{W}_r(r,n,w)$ can be estimated as we have done in Proposition 1.

$$
\begin{aligned}
\mathcal{W}_r(r,n,w) = & \left((n-1)\frac{r-1}{w(r-1)+1}\right)\frac{1}{2(w(r-1)+1)} \\
& + \left(1+(n-1)\frac{r-1}{w(r-1)+1}\right)\frac{1}{2(w(r-1)+1)} \\
& + \left(1+(n-w)\frac{r-1}{w(r-1)+1}\right)\frac{r-1}{2(w(r-1)+1)} \\
& + \left(2+(n-w)\frac{r-1}{w(r-1)+1}\right)\frac{r-1}{2(w(r-1)+1)} \\
& + \left(1+(n-w+1)\frac{r-1}{w(r-1)+1}\right)\frac{r-1}{w(r-1)+1} \\
& + \left(1+(n-w+2)\frac{r-1}{w(r-1)+1}\right)\frac{r-1}{w(r-1)+1} \\
& \ldots \\
& + \left(1+(n-1)\frac{r-1}{w(r-1)+1}\right)\frac{r-1}{w(r-1)+1} \\
= & \, n\frac{r-1}{w(r-1)+1} + \frac{w^2(r-1)^2 + w(3(r-1)) - (r-1) + 1}{2(w(r-1)+1)^2}.
\end{aligned}
$$

In order to estimate $\mathcal{L}_r(r,n,w)$, the relationships $\Pr[(0)_n \curvearrowright (0)_n] + \Pr[(0)_n \curvearrowright (\underbrace{0...0x}_{w})_n] = 1$ and $\Pr[(0)_n \curvearrowright (0)_n] = 1/r$ are used. We can obtain the following formula.

$$
\begin{aligned}
\mathcal{L}_r(r,n,w) = & \sum_{k=1}^{n-w}(n-k-w+1)\left(\frac{1}{r}\right)^{k-1}\left(1-\frac{1}{r}\right)\frac{1}{2(w(r-1)+1)} \\
& +(n+1)\frac{1}{2(w(r-1)+1)} \\
& +(n-w+1)\frac{r-1}{2(w(r-1)+1)} \\
& +(n+1)\frac{r-1}{2(w(r-1)+1)} \\
& +(n-w+2)\frac{r-1}{w(r-1)+1} \\
& \ldots \\
& +n\frac{r-1}{w(r-1)+1} \\
= & \, n - \frac{w(w-2)(r-1)^2 + (w-1)(r-1)+1}{2(w(r-1)+1)(r-1)}.
\end{aligned}
$$

□

B Some Proofs of Section 5

Lemma 1 (*Cutting Lemma*). *For $t \in (\mathcal{Z}_{\varepsilon^{(1)}} \cap \mathcal{Z}_{\varepsilon^{(2)}}) \backslash \{0,n\}$ or $t \in (\mathcal{Z}^c_{\varepsilon^{(1)}} \cap \mathcal{Z}^c_{\varepsilon^{(2)}}) \backslash \{0\}$, we have*

$$\varepsilon^{(1)}(0,t-1) = \varepsilon^{(2)}(0,t-1) \text{ and } \varepsilon^{(1)}(t,n) = \varepsilon^{(2)}(t,n).$$

Proof. We know $\varepsilon^{(1)} = \varepsilon^{(1)}(t,n) \cdot r^t + \varepsilon^{(1)}(0,t-1)$ and $\varepsilon^{(2)} = \varepsilon^{(2)}(t,n) \cdot r^t + \varepsilon^{(2)}(0,t-1)$.

- The case $t \in (\mathcal{Z}_{\varepsilon^{(1)}} \cap \mathcal{Z}_{\varepsilon^{(2)}}) \backslash \{0,n\}$;

Without loss of generality (WLOG), assume $\varepsilon^{(1)}(0,t-1) > \varepsilon^{(2)}(0,t-1)$. As $\varepsilon^{(1)} = \varepsilon^{(2)}$,

$$\varepsilon^{(1)}(0,t-1) - \varepsilon^{(2)}(0,t-1) = r^{t+1} \cdot (\varepsilon^{(2)}(t+1,n) - \varepsilon^{(1)}(t+1,n)) \tag{2}$$

because of $\varepsilon_t^{(1)} = \varepsilon_t^{(2)} = 0$. Then the maximum of left hand side (LHS) of (2) is $2(r^t - 1)$ and r^{t+1} divides right hand side (RHS) of (2). It's a contradiction. Therefore, the assertion is true.

- The case $t \in (\mathcal{Z}^c_{\varepsilon^{(1)}} \cap \mathcal{Z}^c_{\varepsilon^{(2)}}) \backslash \{0\}$;

WLOG assume $\varepsilon^{(1)}(0,t-1) > \varepsilon^{(2)}(0,t-1)$. Then

$$\varepsilon^{(1)}(0,t-1) - \varepsilon^{(2)}(0,t-1) = r^t \cdot (\varepsilon^{(2)}(t,n) - \varepsilon^{(1)}(t,n)). \tag{3}$$

If $r=2$ then the maximum of LHS of (3) is $2^{t+1}-2$ and 2^{t+1} divides RHS of (3). It's a contradiction. If $r \geq 3$ then the maximum of LHS of (3) is $2(r^t-1)$ and r^t divides RHS of (3). It's a contradiction. Therefore, the assertion is true. □

Lemma 2 (*Collision Lemma*). *For any i, $\varepsilon^{(i)}(t,n)$ is either $d(t,n-1)$ or $d(t,n-1)+1$. This implies $\varepsilon^{(i)}(0,t)$ is either $d(0,t)$ or $d(0,t) - r^{t+1}$.*

Proof. $\varepsilon^{(i)} = \varepsilon^{(i)}(i,n) \cdot r^i + \varepsilon^{(i)}(0,i-1)$ and $d = d(i,n-1) \cdot r^i + d(0,i-1)$. As $\varepsilon^{(i)} = d$, $\big(\varepsilon^{(i)}(i,n) - d(i,n-1)\big) \cdot r^i = d(0,i-1) - \varepsilon^{(i)}(0,i-1)$.

As $-r^i < \big(-(r^i-1) <\big)\ d(0,i-1) - \varepsilon^{(i)}(0,i-1)\ \big(< 2\cdot(r^i-1)\big) < 2\cdot r^i$, $-1 < \frac{d(0,i-1)-\varepsilon^{(i)}(0,i-1)}{r^i} < 2$. Here, $\frac{d(0,i-1)-\varepsilon^{(i)}(0,i-1)}{r^i}$ must be an integer since it is equal to $\varepsilon^{(i)}(i,n) - d(i,n-1)$. Thus $\varepsilon^{(i)}(i,n) - d(i,n-1)$ is 0 or 1, i.e., $\varepsilon^{(i)}(i,n)$ is either $d(i,n-1)$ or $d(i,n-1)+1$. □

Theorem 1. *For given m rSD representations of d, if $x = \#(\cap_{i=1}^m \mathcal{Z}^c_{\varepsilon^{(i)}})$, then the search space order l is as follows:*

$$l = \#(\cap_{i=1}^m [\varepsilon^{(i)}]_\star^{List}) = \{2(r-1)\}^x.$$

rSD representations of d is represented within $n+1$ because the original input integer d is radix-r representation of n digits. We call $n+1$ as the length of rSD representation of n digits integer d and denote it $\mathcal{L}$, i.e. $\mathcal{L} = n+1$.

Proof of Theorem 1: We argue by induction on $\mathcal{L}$.
(0) When $\mathcal{L} = 0$: This is the case that the integer d is zero, so it is trivial.

(1) When $\mathcal{L} = 2$: This is the case that d has only one digit representation. Thus $\#(\cap_{i=1}^m \mathcal{Z}^c_{\varepsilon^{(i)}}) = 1$ or 2 and clearly $\#(\cap_{i=1}^m [\varepsilon^{(i)}]_\star^{List}) = 2(r-1)$ or $\{2(r-1)\}^2$, respectively.

(2) Suppose when $\mathcal{L} \leq s$, the assertion is true: That is, if $\varepsilon^{(i)}$ is the i-th rSD representation with length $\mathcal{L} = t$ digits, where for any $t \leq s$, then the search space is $\{2(r-1)\}^{\#(\cap_{i=1}^m \mathcal{Z}^c_{\varepsilon^{(i)}})}$.

(3) We must prove the assertion is true when $\mathcal{L} = s+1$: Let $\varepsilon^{(i)}$ be the i-th rSD representation with length $s+1$ digits and $x = \#(\cap_{i=1}^m \mathcal{Z}^c_{\varepsilon^{(i)}})$. We will prove that

$$\#(\cap_{i=1}^m [\varepsilon^{(i)}]_\star^{List}) = \{2(r-1)\}^x.$$

We will consider the following three cases;

1. **When** $s \in \cap_{i=1}^{m} \mathcal{Z}_{\varepsilon^{(i)}}$:
2. **When** $s \in \cap_{i=1}^{m} \mathcal{Z}^{c}_{\varepsilon^{(i)}}$:
3. **When** $s \in \mathcal{Z}_{\varepsilon^{(i_1)}}$ **and** $\mathcal{Z}^{c}_{\varepsilon^{(i_2)}}$ **for some** $1 \le i_1, i_2 \le m$: in this case there are only the following two sub-cases. By re-indexing if necessary, we assume that
   - 3.1.
     - $s-1 \in \mathcal{Z}_{\varepsilon^{(i)}}$ for $1 \le i \le t_1$,
     - $s-1 \in \mathcal{Z}^{c}_{\varepsilon^{(i)}}$ for $t_1+1 \le i \le m$,
     - $s \in \mathcal{Z}^{c}_{\varepsilon^{(i)}}$ for $1 \le i \le t_2-1$,
     - $s \in \mathcal{Z}_{\varepsilon^{(i)}}$ for $t_2 \le i \le m$, where $1 \le t_1 < t_2 \le m$. Refer to Fig. 1.
   - 3.2.
     - $s-1 \in \cap_{i=1}^{m} \mathcal{Z}^{c}_{\varepsilon^{(i)}}$ for $1 \le i \le m$,
     - $s \in \mathcal{Z}_{\varepsilon^{(i)}}$ for $1 \le i \le t$,
     - $s \in \mathcal{Z}^{c}_{\varepsilon^{(i)}}$ for $t+1 \le i \le m$, where $1 \le t < m$. Refer to Fig. 2.

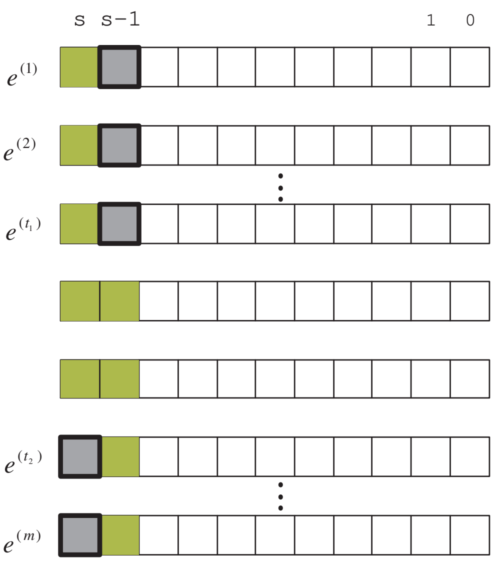

Fig. 1.

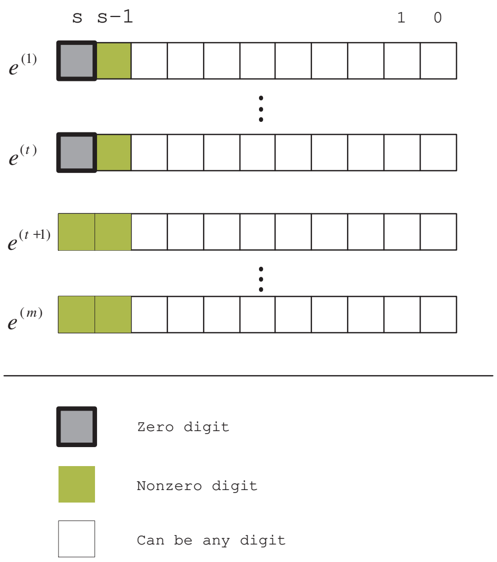

Fig. 2.

[Proof of Case 1.] As $s \in \cap_{i=1}^{m} \mathcal{Z}_{\varepsilon^{(i)}}$

$$\#(\cap_{i=1}^{m} [\varepsilon^{(i)}]_{\star}^{List}) = \#(\cap_{i=1}^{m} [\varepsilon^{(i)}(0, s-1)]_{\star}^{List}).$$

Thus by induction assumption $\#(\cap_{i=1}^{m} [\varepsilon^{(i)}(0, s-1)]_{\star}^{List}) = \{2(r-1)\}^{x}$ because $\varepsilon^{(i)}(0, s-1)$ can be regarded as i-th rSD representation with length s digits and $\#(\cap_{i=1}^{m} \mathcal{Z}^{c}_{\varepsilon^{(i)}(0,s-1)}) = x$.

[Proof of Case 2.] From Lemma 1, especially *Cutting Lemma*, $\varepsilon_s^{(1)} = \ldots = \varepsilon_s^{(m)}$ and $\varepsilon^{(1)}(0, s-1) = \ldots = \varepsilon^{(m)}(0, s-1)$. Thus

$$\#(\cap_{i=1}^{m} [\varepsilon^{(i)}]_{\star}^{List}) = 2(r-1) \cdot \#(\cap_{i=1}^{m} [\varepsilon^{(i)}(0, s-1)]_{\star}^{List}).$$

As $\varepsilon^{(i)}(0, s-1)$ can be regarded as i-th rSD representation with length s digits and $\#(\cap_{i=1}^{m} \mathcal{Z}^{c}_{\varepsilon^{(i)}(0,s-1)}) = x-1$, $\#(\cap_{i=1}^{m} [\varepsilon^{(i)}(0, s-1)]_{\star}^{List}) = \{2(r-1)\}^{x-1}$ by induction assumption. Thus the assertion is true.

[Proof of Case 3.1.] From the recoding rule of rSD representation,

$$\varepsilon_s^{(1)} = \ldots = \varepsilon_s^{(t_2-1)} = \begin{cases} 1 & \text{if } \varepsilon_{s-1}^{(t_2)} > 0, \\ -1 & \text{otherwise.} \end{cases}$$

Then we can make new m rSD representations with length s.

- When $\varepsilon_s^{(1)} = 1$,
  - for $1 \le i \le t_2 - 1$, $\varepsilon^{(i)}(0, s-1)$,
  - for $t_2 \le i \le m$, $(\varepsilon_{s-1}^{(i)} - r) \cdot r^{s-1} + \varepsilon^{(i)}(0, s-2)$, denote it $\varepsilon^{(i)'}(0, s-1)$,
  - then $\varepsilon^{(1)}(0, s-1) = \ldots = \varepsilon^{(t_2-1)}(0, s-1) = \varepsilon^{(t_2)'}(0, s-1) = \ldots = \varepsilon^{(m)'}(0, s-1)$. Note that we can easily obtain above equation by subtracting r^s from $r^s + \varepsilon^{(1)}(0, s-1) = \ldots = r^s + \varepsilon^{(t_2-1)}(0, s-1) = \varepsilon_{s-1}^{(t_2)} \cdot r^{s-1} + \varepsilon^{(t_2)}(0, s-2) = \ldots = \varepsilon_{s-1}^{(m)} \cdot r^{s-1} + \varepsilon^{(t_2)}(0, s-2)$.
- When $\varepsilon_s^{(1)} = -1$,
  - for $1 \le i \le t_2 - 1$, $\varepsilon^{(i)}(0, s-1)$,
  - for $t_2 \le i \le m$, $(\varepsilon_{s-1}^{(i)} + r) \cdot r^{s-1} + \varepsilon^{(i)}(0, s-2)$, denote it $\varepsilon^{(i)'}(0, s-1)$,
  - then $\varepsilon^{(1)}(0, s-1) = \ldots = \varepsilon^{(t_2-1)}(0, s-1) = \varepsilon^{(t_2)'}(0, s-1) = \ldots = \varepsilon^{(m)'}(0, s-1)$.

From above conversion, we can easily check that $\cap_{i=1}^{m}[\varepsilon^{(i)}]_\star^{List} = \Big(\cap_{i=1}^{t_2-1} [\varepsilon^{(i)}(0, s-1)]_\star^{List} \Big) \bigcap \Big(\cap_{i=t_2}^{m} [\varepsilon^{(i)'}(0, s-1)]_\star^{List} \Big)$. As $\varepsilon^{(i_1)}(0, s-1)$ and $\varepsilon^{(i_2)'}(0, s-1)$ can be regarded as i_j-th rSD representation with length s digits, $1 \le i_1 \le t_2 - 1$, $t_2 \le i_2 \le m$, and $\#\Big(\big(\cap_{i=1}^{t_2-1} \mathcal{Z}^c_{\varepsilon^{(i)}(0,s-1)} \big) \bigcap \big(\cap_{i=t_2}^{m} \mathcal{Z}^c_{\varepsilon^{(i)'}(0,s-1)} \big) \Big) = x$ the assertion is proved by using induction assumption.

[Proof of Case 3.2.] From the recoding rule of rSD representation,

$$\varepsilon_s^{(t+1)} = \ldots = \varepsilon_s^{(m)} = \begin{cases} 1 & \text{if } \varepsilon_{s-1}^{(1)} > 0, \\ -1 & \text{otherwise.} \end{cases}$$

Then we can make new m rSD representations with length s.

- When $\varepsilon_s^{(m)} = 1$,
  - for $1 \le i \le t$, $\varepsilon^{(i)}(0, s-1)$,
  - for $t+1 \le i \le m$, $(\varepsilon_{s-1}^{(i)} + r) \cdot r^{s-1} + \varepsilon^{(i)}(0, s-2)$, denote it $\varepsilon^{(i)'}(0, s-1)$,
  - then clearly $\varepsilon^{(1)}(0, s-1) = \ldots = \varepsilon^{(t)}(0, s-1) = \varepsilon^{(t+1)'}(0, s-1) = \ldots = \varepsilon^{(m)'}(0, s-1)$.
- When $\varepsilon_s^{(m)} = -1$,
  - for $1 \le i \le t$, $\varepsilon^{(i)}(0, s-1)$,
  - for $t+1 \le i \le m$, $(\varepsilon_{s-1}^{(i)} - r) \cdot r^{s-1} + \varepsilon^{(i)}(0, s-2)$, denote it $\varepsilon^{(i)'}(0, s-1)$,
  - then $\varepsilon^{(1)}(0, s-1) = \ldots = \varepsilon^{(t-1)}(0, s-1) = \varepsilon^{(t)'}(0, s-1) = \ldots = \varepsilon^{(m)'}(0, s-1)$.

From above conversion, we can easily check that $\cap_{i=1}^{m}[\varepsilon^{(i)}]_\star^{List} = \Big(\cap_{i=1}^{t} [\varepsilon^{(i)}(0, s-1)]_\star^{List} \Big) \bigcap \Big(\cap_{i=t+1}^{m} [\varepsilon^{(i)'}(0, s-1)]_\star^{List} \Big)$. As $\varepsilon^{(i_1)}(0, s-1)$ and $\varepsilon^{(i_2)'}(0, s-1)$ can be regarded as i_j-th rSD representation with length s digits, $1 \le i_1 \le t$, $t+1 \le i_2 \le m$, and $\#\Big(\big(\cap_{i=1}^{t} \mathcal{Z}^c_{\varepsilon^{(i)}(0,s-1)} \big) \bigcap \big(\cap_{i=t+1}^{m} \mathcal{Z}^c_{\varepsilon^{(i)'}(0,s-1)} \big) \Big) = x$ the assertion is proved by using induction assumption. $\square$