타원곡선 암호시스템에서의 빠른 연산을 위한 새로운 덧셈/뺄셈 사슬 알고리즘†

홍 성민°,오 상엽,윤 현수
한국과학기술원 전산학과

A New Addition/Subtraction Chain Algorithm for Fast Computation over Elliptic Curve Cryptosystem

Hong Seong-Min°, Oh Sang-yeop, Yoon Hyun-Soo
Dept. of Computer Science, KAIST

요약

보다 짧은 길이의 덧셈/뺄셈 사슬(addition/subtraction-chain)을 찾는 문제는 정수론을 기반으로 하는 많은 암호시스템들에 있어서 중요한 문제이다. 특히, RSA에서의 모듈라 멱승(modular exponentiation)이나 타원곡선(elliptic curve)에서의 곱셈 연산시간은 덧셈사슬(addition-chain) 또는 덧셈/뺄셈 사슬의 길이와 정비례 한다. 본 논문에서는 덧셈/뺄셈 사슬을 구하는 새로운 알고리즘을 제안하고, 그 성능을 분석하여 기존의 방법들과 비교한다. 본 논문에서 제안하는 알고리즘은 작은윈도우(small-window) 기법을 기반으로 하고, 뺄셈을 사용해서 윈도우의 갯수를 최적화 함으로써 덧셈/뺄셈 사슬의 길이를 짧게 한다. 본 논문에서 제안하는 알고리즘은 512비트의 정수에 대해 평균길이 595.6의 덧셈/뺄셈 사슬을 찾는다.

1 서론

1976년 공개키 암호시스템 개념이 출현한 이래로, 정수론을 기반으로 하는 암호시스템들이 많이 개발되었다[1, 2]. 그러나, 이들은 매우 큰 수들을 다루기때문에 그 수행속도가 느려서 빠른 연산알고리즘을 필요로 한다. RSA 등에서 사용되는 모듈라 멱승도 이러한 연산들 중 하나이다[3]. 모듈라 멱승 연산은 모듈라 곱셈(modular multiplication)의 반복으로 이루어져 있다. 이러한 모듈라 곱셈의 연산순서를 나타낸 것이 덧셈사슬(addition-chain)로서 다음과 같이 정의된다.

정의 1.1 임의의 양의 정수 n에 대한 **덧셈사슬**은 다음과 같은 성질을 지닌 일련의 수열 $a_0, a_1, ..., a_l$이며, 이 때의 l을 덧셈사슬의 길이라고 한다.

1. $a_0 = 1$, $a_l = n$.
2. $a_i = a_j + a_k, 0 \leq j \leq k < i \leq l$.

보다 짧은 덧셈사슬을 찾는 것은 바로 모듈라 멱승연산을 빠르게 수행할 수 있음을 의미한다. 이러한 덧셈사슬은 타원곡선에서 정점의 배수 $d \cdot P(P$는 정점)를 구할 때에도 연산시간을 단축시키는 데에 사용된다. 즉, 타원곡선에서 정점의 배수를 계산 하는 것은 덧셈을 반복함으로써 이루어지는데, 이 때 덧셈의 연산순서가 덧셈사슬로 표현될 수 있다. 그런데,

†본 연구는 국민은행과 (주)한글과컴퓨터의 지원에 의해 수행중임

타원곡선에서는 한 정점과 다른 정점의 덧셈이 뺄셈과 같은 시간만큼 걸린다. 따라서 타원곡선에서는 뺄셈까지 포함해서 연산순서를 결정할 수 있다. 이 때의 연산순서를 덧셈/뺄셈 사슬이라고 하며 다음과 같이 정의된다.

정의 1.2 임의의 양의 정수 n에 대한 **덧셈/뺄셈 사슬**은 다음과 같은 성질을 지닌 일련의 수열 $a_0, a_1, ..., a_l$이며, 이 때의 l을 덧셈/뺄셈 사슬의 길이라고 한다.

1. $a_0 = 1$, $a_l = n$.
2. $a_i = \pm a_j \pm a_k, 0 \leq j \leq k < i \leq l$.

타원곡선에서는 덧셈사슬과 마찬가지로, 보다 짧은 덧셈/뺄셈 사슬을 구할수록 보다 빠르게 정점의 배수를 구할 수 있다.

이와같이 덧셈사슬과 덧셈/뺄셈 사슬은 암호학에서 필요한 연산들의 수행시간을 단축시키는 데에 중요한 역할을 하기때문에 그동안 많은 연구들이 있어왔다. [4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16, 17, 18]. 이러한 덧셈사슬과 덧셈/뺄셈 사슬에 대한 연구들 중 주목할 만한 것들이 몇 가지 있다. 먼저 이론적인 연구결과들을 살펴본다. Downey 등은 [10]에서 가장 짧은 덧셈사슬을 찾는 문제가 NP-complete임을 증명하였다. 그리고, 임의의 수 a에 대한 덧셈사슬 중 가장 짧은 것의 길이 $l(a)$의 최소한계(lower bound)는 $\log n + \log \nu(n) - 2.13$임이 밝혀졌다[18]($\nu(n)$은 n을 이진수로 표현했을 때 나타나는 "1"의 갯수). 다음으로 덧셈사슬 또는 덧셈/뺄셈 사슬을 구하는 알고리즘들을 살펴본다. 우선 가장 직관적인 이진방법(binary method)과 이를 개선한 방법들이 [4],[14], 그리고 [15]에 나타나 있다. 그리고, 큰윈도우(large-window) 기법에 사용되는 휴리스틱(heuristic) 알고리즘들이 [5]에 나타나 있다. 또한, Yacobi는 [7]에서 큰 수들에 대한 빠른 연산과 데이타 압축 사이의 유사성을 이용하는 알고리즘을 제안하였다.

그동안 이러한 많은 알고리즘들이 연구되었으나, 기본적으로 최소길이의 덧셈사슬을 구하는 문제가 NP-complete이므로 모든 알고리즘은 최소길이보다 긴 덧셈사슬과 덧셈/뺄셈 사슬을 구하게 되어 있다. 따라서, 항상 개선의 여지를 남기게 된다. 게다가, 모든 알고리즘들이 [4]의 작은윈도우 기법을 사용해서 얻어지는 덧셈사슬보다 길거나, 거의 비슷한 길이의 덧셈사슬 또는 덧셈/뺄셈 사슬을 찾는다. 본 논문에서는 작은윈도우 기법을 기본으로 하는 새로운 알고리즘을 제안한다. 이 알고리즘은 뺄셈을 이용해서 윈도우의 갯수를 최적화함으로써 덧셈/뺄셈 사슬의 길이를 짧게 한다. 본 논문에서 제안하는 알고리즘은 512비트의 정수에 대해 평균길이 595.6인 덧셈/뺄셈 사슬을 찾아낸다. 또한 본 논문에서 제안하는 알고리즘은 사용가능한 메모리가 적은 상황에서도 성능이 크게 떨어지지 않음을 보인다.

본 논문의 구성은 다음과 같다. 2절에서는 본 논문에서 제안하는 알고리즘에 대해 설명하고, 3절에서는 그 알고리즘을 사용하여 구할 수 있는 덧셈/뺄셈 사슬의 길이를 계산한다. 4절에서는 기존의 다른 덧셈/뺄셈 사슬 알고리즘들과의 비교를 수행하고, 마지막으로 5절에서는 결론을 맺는다.

2 알고리즘

본 논문에서 제안하는 알고리즘은 작은윈도우 기법을 기반으로 하므로, 먼저 작은윈도우 기법을 간략하게 기술하도록 한다.

2.1 작은윈도우 기법

작은윈도우 기법은 숫자를 이진법으로 표현하여, 이를 여러 개의 작은조각(이를 윈도우라 한다)들로 나누어 덧셈사슬을 구하는 방법이다. 윈도우가 가지게 되는 범위의 모든 수들은

미리 구해 놓는다. 예를 들어 "3584965235_{10}"에 대한 덧셈사슬을 작은윈도우 기법을 이용해서 구해본다. 먼저, "3584965235_{10}"를 이진수로 나타내면

$$11010101101011100011101001110011$$

이 된다. 이를 크기가 4인 조각(윈도우)으로 나누면 다음과 같다.

$$\underline{1101}\ 0\ \underline{1011}\ 0\ \underline{1011}\ \underline{1}\ 000\ \underline{111}\ 0\ \underline{1001}\ \underline{11}\ 00\ \underline{11}$$

이것을 가지고 덧셈사슬을 구하기 위해서는 우선 크기가 4인 윈도우가 가질 수 있는 모든 값들을 덧셈사슬에 포함시킨다. 즉, $\{1,2,3,5,7,...,13,15\}$가 덧셈사슬에 포함된다. 우선 가장 처음에 나오는 윈도우 "$\underline{1101}$"로부터 시작한다. "$\underline{1101}$"에 해당하는 13_{10}이 이미 덧셈사슬에 포함되어 있으므로 별도의 덧셈이 필요없다. 이제 "$\underline{1101}$"을 왼쪽으로 5번 이동(shift) 시키면 "$\underline{1101}$ 00000"이 되고, 그 과정에서 $\{13\times 2, 13\times 3, 13\times 4, 13\times 5\}$가 덧셈사슬에 포함된다. 그 후에 "$\underline{1101}$ 00000"에 두번째 윈도우 "$\underline{1011}$"을 더하면 "$\underline{1101}$ 0 $\underline{1011}$"이 되며, 이에 해당하는 10진수 $76_{10}(=13\times 5+11)$이 덧셈사슬에 포함된다. 이는 덧셈사슬을 구하고자 하는 수에 대한 이진표현 중에서 앞부분 9비트에 해당한다. 이러한 과정을 윈도우 접합(window merge)이라고 하며, "0"이 아닌 윈도우 8개를 모두 접합하고 나면 그 과정에서 덧셈사슬이 완성된다. 이러한 작은윈도우 기법으로 덧셈사슬을 구하는 방법이 알고리즘 2.1에 *C-like* 의 사코드(pseudo-code)로 정리되어 있다.

이 때에 만들어지는 덧셈사슬의 길이를 계산해 본다. 처음에 미리 만들어 놓는 덧셈사슬의 길이는 $2^{4-1}=8$이고, 필요한 이동의 횟수는 첫번째 윈도우가 윈도우 접합을 수행하는 동안 총 이동한 횟수와 같으므로 $32-4=28$회이다. 또한 각 윈도우를 더할 때마다 한 개씩의 숫자가 덧셈사슬에 더해지므로 윈도우의 갯수(8개)에서 하나 뺀 수가 덧셈사슬의 길이에 더해져야 한다. 따라서 덧셈사슬의 총 길이는 43(=8+28+7)이다.

알고리즘 2.1 덧셈사슬을 구하고자 하는 숫자를 $a(=e_{n-1}e_{n-2}\cdots e_1e_0, e_i=0|1, 0\le i<n)$라 하고 윈도우 크기를 k라고 한다. 또한 α는 덧셈사슬을 집합으로 표현한 것이며, w_i는 각 윈도우를 의미한다.

```
1   Small_Window( a,k )
2   {
3       α = {1,2,3,5,7,9,...,2^k − 1};
4       i = n − 1; p = 0;
5       while( i ≥ 0 ) {
6           w_p = e_i e_{i−1} e_{i−2} ··· e_{i−k+1};
7           i = i − k;
8           for( ;e_i == "0";i − − )    w_p = w_p|e_i;
9           p + +;
10      }

    /* 현재 a = w_{p−1} w_{p−2} ··· w_1 w_0, w_i = e_{i,s_i−1} e_{i,s_i−2} ··· e_{i,1} e_{i,0}, 0 ≤ i < p,k < s_i */

11      while( w_0 > 2^k )    w_0 = w_0 / 2;
12      α = α ∪ {w_0, 2^1 × w_0, 2^2 × w_0, ..., 2^{s_0 − ⌈log w_0⌉} × w_0};
13      ac = 2^{s_0 − ⌈log w_0⌉} × w_0
14      for( i = 1; i < p; i + + ) {
15          while( w_i == even )    w_i = w_i / 2;
```

```
16          α = α∪ {2^1 × ac, 2^2 × ac, ..., 2^⌈log w_i⌉ × ac};
17          ac = 2^⌈log w_i⌉ × ac + w_i;
18          α = α∪ {ac, 2^1 × ac, 2^2 × ac, ..., 2^(s_i − ⌈log w_i⌉) × ac};
19          ac = 2^(s_i − ⌈log w_i⌉) × ac
20      }
21  }
```

2.2 새로운 알고리즘

작은윈도우 기법을 수정한 새로운 덧셈/뺄셈 사슬 알고리즘을 설명한다.

윈도우 크기 k를 4로 하여 작은윈도우 기법으로 덧셈사슬을 구할 경우, 크기가 9인 비트열 "101110101"은 "$\underline{1011}\ \underline{1010}\ \underline{1}$"과 같이 세 개의 윈도우로 나뉘어진다. 즉 $(\underline{1011}\times 2^4+\underline{1010})\times 2^1+\underline{1}$로 표현되므로, 길이 8인 기본 덧셈사슬 $\{1,2,3,...,15\}$를 포함하여, 이동 5번과 윈도우 접합 2번이 필요하므로 덧셈사슬의 총길이가 15(=8+5+2)가 된다. 그러나, 여기에 뺄셈을 도입하게 되면, $\underline{1010}\ \underline{1} = 100000 - 1011$이므로 $(\underline{1011}+1)\times 2^5-\underline{1011}) = \underline{1100}\times 2^5-\underline{1011}$로 변형될 수 있다. 따라서, 윈도우 접합이 두 번에서 한 번으로 줄어 덧셈/뺄셈 사슬의 총길이는 14(=8+5+1)가 된다.

즉, 비트열을 윈도우로 분할할 때에, 서로 접하게 되는 윈도우의 경우 한 비트의 이득을 볼 수 있다는 점이 핵심 아이디어이다. 게다가 접하는 두 윈도우 중에서 뒤의 윈도우가 11,111,1111,... 이런식으로 상위 비트들이 1로만 이루어져 있으면 1의 갯수만큼 이득을 볼 수 있게 된다.

2.1절에서 사용한 예를 이용해서 덧셈/뺄셈 사슬을 구하는 과정을 살펴본다. 윈도우를 처음 분할하는 방법은 동일하므로 "3584965235_{10}"의 이진표현을 크기가 4인 윈도우로 나누어 다음과 같이 표현한다.

$$\underline{1101}\ 0\ \underline{1011}\ 0\ \underline{1011}\ \underline{1}\ 000\ \underline{111}\ 0\ \underline{1001}\ \underline{11}\ 00\ \underline{11}$$

여기서 보면 "0"이 아닌 윈도우 중 3번째와 4번째 윈도우가 서로 붙어있다. 그러므로 $\underline{1011}\ \underline{1}\ 000\ 1 = (\underline{1011}\times 2^1+\underline{1})\times 2^4+1 = \underline{1100}\times 2^5-\underline{1111}$이 되어 전체 이진표현의 윈도우 분할 모양새가 다음과 같이 변한다.

$$\underline{1101}\ 0\ \underline{1011}\ 0\ \underline{11}\ 00\ /\underline{1111}\ \underline{1101}\ 00\ \underline{111}\ 00\ \underline{11}$$

위의 식에서, "/"은 사선의 앞 윈도우에서 사선의 뒷 윈도우를 뺌을 의미한다. 위의 수정된 윈도우 분할에서 보면, "0"이 아닌 윈도우들 중 4번째와 5번째 윈도우가 맞붙어 있다. 따라서 이는 $/\underline{1111}\ \underline{1101}\ 00 = (-\underline{1111}\times 2^4+\underline{1101})\times 2^2 = (-\underline{1110}\times 2^4+\underline{11})\times 2^2$으로 변형될 수 있다. 이러한 식으로 마지막 연결된 윈도우까지 변형을 시켜나가면 최종적으로 다음과 같은 모양의 분할이 이루어진다.

$$\underline{1101}\ 0\ \underline{1011}\ 0\ \underline{11}\ 00\ /\underline{111}\ 000\ /\underline{1011}\ 000\ /\underline{1101}$$

이것을 가지고 덧셈/뺄셈 사슬을 구하는 방법은 2.1절에서 설명한 작은윈도우 기법과 유사하다. 다만, 앞에 "/"이 붙어있는 윈도우가 나타나면 그 값을 더하는 대신 빼면 된다. 지금까지 설명한 방법으로 덧셈/뺄셈 사슬을 구하는 절차가 알고리즘 2.2에 *C-like* 의사코드로 정리되어 있다. 그리고, 실제로 512비트의 정수에 대해서 덧셈/뺄셈 사슬을 구하는 과정이 본 논문의 부록에 수록되어 있다.

이러한 방법으로 구해지는 덧셈/뺄셈 사슬의 길이를 계산해 본다. 처음에 만들어 놓는 순열의 길이와 총 필요한 이동의 회수는 2.1절에서와 같이 각각 8과 28이다. 다른 것은 윈도

우의 갯수로서 작은윈도우 기법으로 덧셈사슬을 구할 때보다 2개가 줄어든 6개이다. 따라서, 위의 새로운 알고리즘으로 구해지는 덧셈/뺄셈 사슬의 총 길이는 41(=8+28+5)이다.

알고리즘 2.2 덧셈사슬을 구하고자 하는 숫자를 $a(= e_{n-1}e_{n-2}\cdots e_1e_0, e_i = 0|1, 0 \le i < n)$라 하고 윈도우 크기를 k라고 한다. 또한 α는 덧셈사슬을 집합으로 표한한 것이며, w_i는 각 윈도우를 의미한다.

```
1   Proposed( a,k )
2   {
3       α = {1, 2, 3, 5, 7, 9, ..., 2^k − 1};
4       i = n − 1; p = 0; sub[...] = FALSE;
5       while( i ≥ 0 ) {
6           w_p = e_i e_{i−1} e_{i−2} ··· e_{i−k+1};
7           i = i − k;
8           if( e_i == "1" ) {
9               w_p = w_p + 1;
10              if( w_p ≤ 2^{k−1} ) { w_p = 2's complement of w_p; sub[p] = TRUE; }
11              for( ;e_i == "1";i − − )    w_p = w_p|e_i;
12          }
13          else {
14              if( w_p < 2^{k−1} ) { w_p = 2's complement of w_p; sub[p] = TRUE; }
15              for( ;e_i == "0";i − − )    w_p = w_p|e_i;
16          }
17          p + +;
18      }

    /*현재 a = w_{p−1} w_{p−2} ··· w_1 w_0, w_i = e_{i,s_i−1} e_{i,s_i−2} ··· e_{i,1} e_{i,0}, 0 ≤ i < p, k < s_i*/

19      while( w_0 > 2^k )    w_0 = w_0 / 2;
20      α = α∪ {w_0, 2^1 × w_0, 2^2 × w_0, ..., 2^{s_0 − ⌈log w_0⌉} × w_0};
21      ac = 2^{s_0 − ⌈log w_0⌉} × w_0
22      for( i = 1; i < p; i + + ) {
23          while( w_i == even )    w_i = w_i / 2;
24          α = α∪ {2^1 × ac, 2^2 × ac, ..., 2^{⌈log w_i⌉} × ac};
25          ac = 2^{⌈log w_i⌉} × ac
26          if( sub[i] == TRUE )    ac = ac − w_i;
27          else    ac = ac + w_i;
28          α = α∪ {ac, 2^1 × ac, 2^2 × ac, ..., 2^{s_i − ⌈log w_i⌉} × ac};
29          ac = 2^{s_i − ⌈log w_i⌉} × ac
30      }
31  }
```

3 성능분석

본 논문에서 제안하는 알고리즘을 사용해서 구해지는 덧셈/뺄셈 사슬의 길이를 평균적인 경우(average case)와 최악의 경우(worst-case)에 대해 각각 계산한다. 이번 절에서 사용되는 a,k, 그리고 n은 각각 다음의 의미를 지닌다.

- a: 덧셈/뺄셈 사슬을 구하고자 하는 정수.
- n: a를 이진수로 표현하는 데에 필요한 비트수, 즉 $\lfloor \log a \rfloor + 1$
- k: 윈도우의 크기

3.1 평균적인 경우

본 논문에서 제안하는 알고리즘을 사용해서 구해지는 덧셈/뺄셈 사슬의 원소들은 다음의 세 가지 종류로 나누어진다.

1. 미리 구해놓는 수들$(2, 3, 5, 7, ..., 2^k - 1)$
2. 이미 덧셈/뺄셈 사슬에 들어있는 원소를 2배 함으로써 얻어지는 수들과, 미리 구해놓는 수들 중 두 개를 더함으로써 얻어지는 수
3. 미리 구해놓는 수들 중 하나와 위의 두번째 종류의 수들 중 하나의 덧셈, 또는 뺄셈으로 만들어지는 수들

위의 세 가지 종류의 원소들 중, 첫번째 종류의 원소들의 갯수는 2^{k-1}이 된다. 두 번째 종류의 원소들의 갯수는 덧셈/뺄셈 사슬을 구하는 동안에 필요한 총 이동 횟수와 같다. 그런데, a의 이진표현 중 최상위에서부터 k번째의 비트가 "1"이면 최상위 윈도우의 값은 2^k보다 작은 홀수가 되어 따로이 덧셈을 필요로 하지 않으므로 $n-k$번의 이동이 필요하다. 반면에, 그것이 "0"일경우에는 최상위 윈도우의 값이 2^k보다 작은 짝수가 되어 한 번의 덧셈이 더 필요하게 되어 $n-k+1$번의 이동이 필요하게 된다. a의 이진표현 중 최상위에서부터 k번째의 비트가 "0"일 확률과 "1"일 확률은 각각 $\frac{1}{2}$이므로 평균적으로 $n-k+0.5$번의 이동이 필요하고, 이것이 두 번째 종류의 원소들의 갯수이다. 이 두 가지 종류의 원소들의 갯수는 작은윈도우 기법에서와 동일하다. 다만 마지막 세 번째 종류의 원소들의 갯수만 달라지게 된다.

세 번째 종류의 원소들의 갯수는 윈도우의 갯수보다 하나 적으므로 평균 윈도우 갯수를 계산한다. 먼저 k비트인 윈도우 하나를 생성하고 난 후, 그 다음 윈도우가 할당되는 위치를 생각해 본다. 이미 생성된 윈도우에 이어져 오는 비트가 "0"이면 작은윈도우 기법에 의해서 다음 윈도우는 한 비트 물러나서 만들어진다. 또한 그 비트가 "1"이면 2.2절에서 설명한 방법에 의해서 다음 윈도우가 한 비트 물러나서 할당된다. 즉 방금 생성된 윈도우에 이어져 오는 비트가 "0"이든지 "1"이든지 다음 윈도우는 한 비트 물러나서 할당된다. 또한 이미 생성된 윈도우 뒤에 이어져 오는 비트들이 "00"이면 작은윈도우 기법에 의해서 다음 윈도우는 두 비트 물러나서 만들어진다. 그리고, 그 비트들이 "11"이면 2.2절에서 설명한 방법에 의해 그 다음 윈도우가 두 비트 물러나서 할당된다. 따라서 다음 윈도우가 두 비트 걸러서 만들어지게 되는 확률은 $\frac{1}{2}$이다. 이러한 방식으로, 이미 생성된 윈도우에 이어져 오는 비트열에 따라서 다음 윈도우와의 간격이 정해진다. 지금까지 설명한 것처럼 윈도우 간격이 한 비트일 확률은 1이고, 두 비트일 확률은 $\frac{1}{2}$이며, 세 비트일 확률은 $\frac{1}{4}$이다. 즉, 윈도우들 간의 간격이 i비트일 확률은 2^{1-i}이다. 따라서 윈도우들 간의 평균 간격은 $2(=\sum_{i=0} 2^{1-i})$ 비트이다. 그러므로, 본 논문에서 제안한 알고리즘에 의한 평균 윈도우 갯수는 $\frac{n}{k+2}$개이다. 지금까지 계산한 바에 의하면, 본 논문에서 제안한 알고리즘에 의해서 구해지는 덧셈/뺄셈 사슬의 평균길이는 다음과 같다.

$$2^{k-1} + n - k - 0.5 + \frac{n}{k+2}$$

3.2 최악의 경우

본 논문에서 제안한 알고리즘으로 덧셈/뺄셈 사슬을 구할 때 최악의 경우가 되는 입력을 생각한다. 3.1절에서 구분한 세 가지 종류 중, 첫번째 종류의 원소들은 입력에 관계가 없다. 두 번째 종류의 원소들의 갯수는 입력의 크기에만 좌우된다. 다만, 처음 만들어지는 윈도우의 값이 짝수일 경우에 하나의 원소가 더 필요하다. 입력에 좌우되는 것은 세번째 종류의 원소들로서, 윈도우가 가장 많이 필요한 경우가 최악의 경우이다. 그런데, 3.1절에서 설명했듯이 방금 만들어진 윈도우에 이어져 오는 비트가 "0"이든지 "1"이든지 다음 윈도우는 한 비트 물러나서 할당된다. 따라서, 모든 윈도우들이 한 비트씩을 사이에 두고 할당될 경우가 최악의 경우이다.

이러한 경우에 본 논문에서 제안한 알고리즘을 사용해서 얻어지는 덧셈/뺄셈 사슬의 길이를 3.1절에서 설명한 세 가지 종류로 나누어 계산한다. 먼저, 첫번째 종류의 갯수는 평균적인 경우에서와 동일하고, 두 번째 종류의 것은 덧셈이 한 번 필요한 경우이므로 $n-k+1$ 개이다. 마지막으로, 윈도우의 갯수는 $\frac{n}{k+1}$이다. 따라서, 본 논문에서 제안한 알고리즘을 사용해서 얻어지는 덧셈/뺄셈 사슬의 총 길이는 최악의 경우에 다음과 같다.

$$2^{k-1}+n-k+\frac{n}{k+1}$$

4 비교평가

본 논문에서 제안한 알고리즘을 사용해서 구해지는 덧셈/뺄셈 사슬의 길이를 기존의 여러 가지 방법들로 구해지는 덧셈사슬 또는 덧셈/뺄셈 사슬들의 길이와 비교한다. 이번 절에서 사용되는 a,n,그리고 k는 3절에서 사용된 것과 같은 의미로 사용된다.

우선, 기존의 방법들을 사용해서 덧셈사슬 또는 덧셈/뺄셈 사슬을 구할 때의 성능을 살펴본다. 이진방법[4]으로 구해지는 덧셈사슬의 길이는 평균 $\frac{3}{2}n-1.5$이고, 최악의 경우에 $2n-2$이다. 작은윈도우 기법[6]을 사용하면, 평균길이 $2^{k-1}+n-k-0.5+\frac{n}{k+1}$의 덧셈사슬을 얻을 수 있고 최악의 경우 얻어지는 덧셈사슬의 길이는 $2^{k-1}+n-k+\frac{n}{k}$이다. [5]에서 Bos 등이 제안한 휴리스틱을 사용해서 큰윈도우 기법으로 덧셈사슬을 구하면, 512비트의 정수에 대해 평균길이 605의 덧셈사슬을 얻을 수 있다. 그러나, 이는 휴리스틱을 사용하므로 정확한 분석이 불가능하다. Morain-Olivos[6, 14]의 방법을 사용해서 구해지는 덧셈/뺄셈 사슬의 길이는 $\frac{5}{3}n$을 보장하고, 평균 $\frac{4}{3}n$이 된다. 그리고 [7]에서 Yacobi가 제안한 알고리즘을 사용하면, 평균 길이 $n-(\log n-\log\log n)+1.5(\frac{n}{\log n}+o(\frac{n}{\log n}))$의 덧셈사슬이 구해진다. 마지막으로 Koyama 등이 [13]에서 제안한 방법을 사용해서 덧셈/뺄셈 사슬을 구할 경우 평균 길이가 $2^{k-1}+n-k+\frac{3}{4}+\frac{n+1/4}{k+3/2}$이다.

지금까지 살펴본 방법들과 본 논문에서 제안한 알고리즘을 수행하여 얻어지는 덧셈사슬 또는 덧셈/뺄셈 사슬들의 평균길이에 대한 그래프가 그림1에 나타나 있다. 가로축은 피연산자의 크기 n을 의미하고, 세로축은 덧셈사슬 또는 덧셈/뺄셈 사슬의 길이에서 n을 뺀 값을 나타낸다. 512비트의 정수에 대해, 위에서 설명한 알고리즘들을 사용해서 구해지는 덧셈사슬 또는 덧셈/뺄셈 사슬들의 길이가 표1에 비교되어 있다. 각각 평균길이와 최악의 상황에서의 길이가 나타나 있다. 표1에서, 알고리즘 열(column)의 괄호안의 숫자는 윈도우 크기를 나타낸다. 그리고 $l(a)$ 열에서 괄호 밖의 숫자는 평균적인 경우를 의미하고, 괄호 안의 숫자는 최악의 경우를 의미한다. 그림1과 표1에서 알 수 있듯이, 본 논문에서 제안한 알고리즘이 가장 좋은 성능을 보인다.

4.1 윈도우 크기에 따른 성능비교

작은윈도우 기법이나 큰 윈도우 기법에 의해서 덧셈사슬을 구하면, 사용한 윈도우의 크기

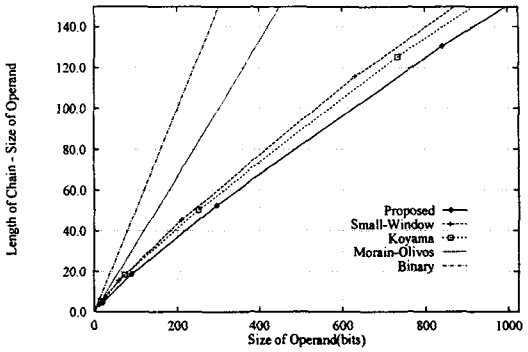

그림 1: 각 방법에 의해서 구해지는 덧셈사슬 또는 덧셈/뺄셈 사슬의 길이 비교

알고리즘	길이
이진방법[4]	766.5(1022)
Morain-Olivos'[14]	681.7(768)
Yacobi's[7][13]	635.1(−)
작은윈도우(5)[4]	607.8(625.4)
큰윈도우(11)[5]	605(−)
Koyama's(5)[13]	602.6(629)
제안된 알고리즘(5)	**595.6(608.3)**

표 1: 각 알고리즘으로 구해지는 덧셈사슬 또는 덧셈/뺄셈 사슬의 길이 비교, $n = 512$

에 따라서 구해지는 덧셈사슬의 길이가 달라진다. 이는 본 논문에서 제안한 알고리즘의 경우도 마찬가지이다. 그러나, 윈도우의 크기가 크다는 것은 메모리가 많이 필요하다는 것이다. 윈도우의 크기를 k라 하면, 필요한 메모리의 양은 2^k에 비례해서 증가한다. 따라서, 윈도우 크기가 작을수록 메모리에 있어서는 이득을 보게 된다.

512비트 정수에 대해 본 논문에서 제안한 알고리즘이 구하는 덧셈/뺄셈 사슬의 길이를 윈도우 크기에 따라 나타낸 그래프가 그림2에 나타나 있다. 또한 그림2에는 작은윈도우 기법을 사용해서 구할 때의 덧셈사슬의 길이 변화와 Koyama 등의 방법을 사용해서 구할 때의 덧셈/뺄셈 사슬의 길이 변화도 함께 나타나 있다. 그림2에서 볼 수 있듯이 본 논문에서 제안한 알고리즘의 경우 곡선의 기울기가 완만하다. 즉, 사용가능한 메모리의 양이 적은 경우에도 좋은 성능을 보임을 의미한다. 표2에 구체적인 수치들이 나타나 있다.

윈도우 크기	1	2	3	4	5	6	7	8
작은윈도우	766.5	682.2	640.5	617.9	607.8	610.6	632.5	688.4
Koyama's	717.2	659.1	627.6	609.9	602.6	607.1	630.2	686.7
새로운 알고리즘	681.7	639.5	614.9	600.8	595.6	601.5	625.4	682.7

표 2: 윈도우 크기에 따른 덧셈사슬 또는 덧셈/뺄셈 사슬의 길이 변화

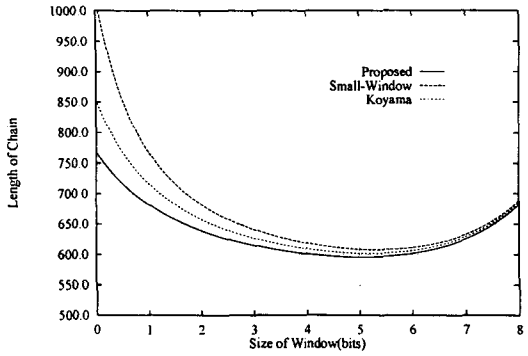

그림 2: 윈도우 크기에 따른 덧셈사슬 또는 덧셈/뺄셈 사슬의 길이 변화

5 결론

본 논문에서는 작은윈도우 기법을 응용한 새로운 덧셈/뺄셈 사슬 알고리즘을 제안하였다. 덧셈/뺄셈 사슬을 구하고자 하는 정수를 a라고 하고, 윈도우의 크기를 k라고 하면, 본 논문에서 제안한 알고리즘을 사용해서 구해지는 덧셈/뺄셈 사슬의 평균길이는 다음과 같다.

$$2^{k-1} + \lfloor \log a \rfloor - k + 0.5 + \frac{\lfloor \log a \rfloor + 1}{k+2}$$

또한 최악의 경우에도 다음과 같은 길이의 덧셈/뺄셈 사슬을 찾을 수 있다.

$$2^{k-1} + \lfloor \log a \rfloor - k + 1 + \frac{\lfloor \log a \rfloor + 1}{k+1}$$

이는 기존의 다른 어떠한 알고리즘을 사용할 때보다 짧은 길이이다. 본 논문에서 제안한 알고리즘의 또다른 장점은 사용가능한 메모리의 양이 적을 때에도 효과적으로 덧셈/뺄셈 사슬을 구할 수 있다는 점이다.

참고 문헌

[1] W.Diffie and M.E.Hellman, "New directions in cryptography," *IEEE Trans. Computers*, vol. IT-22, pp. 644–654, June 1976.

[2] W. Diffie, "The first ten years of public-key cryptography," in *Proceeding of The IEEE*, vol. 76,NO.5, pp. 560–576, May 1988.

[3] R.L.Rivest, A.Shamir, and L.Adleman, "A method for obtaining digital signatures and public key crytosystems," *CACM*, vol. 21, pp. 120–126, 1978.

[4] D.E.Knuth, *The art of computer programming.* Addition-Wesley,Inc., 1981.

[5] J. Bos and M. Coster, "Addition chain heuristics," in *Crypto'89*, pp. 400–407, 1989.

[6] M.J.Coster, *Some algorithms on addition chains and their complexity.* CWI Report CS-R9024, 1990.

[7] Y.Yacobi, "Exponentiating faster with addition chains," in *Eurocrypt'90*, 1991.

[8] Y.Tsai and Y.Chin, "A study of some addition chain problems," *Intern. J.Computer Math.*, vol. 22, pp. 117–134, 1987.

[9] E. Vegh, "A note on addition chains," *Journal of Combinatorial Theory(A)*, vol. 19, pp. 117–118, 1975.

[10] P. Downey, B. Leong, and R. Sethi, "Computing sequences with addition chains," *SIAM J. Comput.*, vol. 10, pp. 638–646, August 1981.

[11] H. Volger, "Some results on addition/subtraction chains," *Information Processing Letters*, vol. 20, pp. 155–160, 1985.

[12] J. Olivos, "On vectorial addition chains," *Journal of Algorithms*, vol. 2, pp. 13–21, 1981.

[13] K. Koyama and Y. Tsuruoka, "A signed binary window method for fast computing over elliptic curves," *IEICE Trans. Fundamentals*, vol. E76-A, pp. 55–62, 1993.

[14] J.Jedwab and C.J.Mitchell, "Minimum weight modified signed-digit representations and fast exponentiation," *Electronics Letters*, vol. 25, pp. 1171–1172, 1989.

[15] A.Selby and C.Mitcheil, "Algorithms for software implementations of rsa," *IEE Proceedings(E)*, vol. 136, pp. 166–170, MAY 1989.

[16] A. Yao, "On the evaluation of powers," *SIAM J. Comput.*, vol. 5, pp. 100–103, March 1976.

[17] P.Erdos, "Remarks on number theory iii," *ACTA Arithmetica*, vol. VI, pp. 77–81, 1960.

[18] A.Schonhage, "A lower bound on the length of addition chains," *Theoret.Comput.Sci.*, vol. 1, pp. 1–12, 1975.

부록

입력데이타 :

1122798133684174844058006430767237907378986585188019355510336830039477007189468012574248319638782630422113824439319436832022387346529226956954300630822016510

입력데이타의 이진표현 :

1101011001100001001110101100010001100111010011110110001000010110101000100110001100101000000010001111100001110101010100001100100100111001000111010111101010010100111001011010101100000101001101111101011101000100110010110110001010111110110011111110101101011101111111100111000001010010011100001111001011010000010100001100001000110011011000001100110110000010110011111001110010110001110111011010111000100100000101100000000011001111010010101000001100110010111111101011101100111111010101110001100111110100000101

작은윈도우 기법으로 윈도우 분할을 수행한 결과(86개의 윈도우) :

1101 0 11001 1 0000 10011 10101 10001 000 11001 1101 00 1111 0 11 000 1 0000 1011 0 101 000 10011 000 11001 0 1 0000000 10001 1111 0000 11101 0 10101 0000 11001 00 10011 10111 00 10001 1101 0 1111 0 101 00 101 00 111 00 1011 0 10101 1 00000 10101 0 10011 0 11111 0 10111 0 10001 00 11001 0 11011 000 10101 1111 0 11001 11111 0 10101 10101 11011 11111 00 111 00000 101 00 10011 1 0000 1111 00 1011 0 1 0000 10101 000 11 0000 10001 10011 0 11 00000 11001 1011 00000 1011 00 1111 00 111 00 1011 000 11 000 10111 0 1111 0 10111 000 1001 00000 1011 00000000 11001 11101 00 10101 0 1 00000 11001 1001 0 11111 10101 11011 00 11111 0 10101 11000 11001 11101 00000 101

뺄셈을 이용해서 윈도우 분할을 수정한 결과(73개의 윈도우) :

11011 00 /1101 0000 101 000 /10101 000 1001 00 /11001 0 101 0000 /101 000 10001 0 /10011 0 10001 00 11001 00 /11011 0000000 1001 00000 /1111 000 /10101 0 /11 0000 11001 00 101 000 /1001 00 1001 000 /101 0000 /1011 00 10101 000 /1101 00 /10101 00 /11111 0 /10101 0 /11001 00000 /101 000 /10111 0 /1101 00 10111 00 /11101 0 /1 00000 /10011 000000 /10101 00 /101 000 /1 0000000 /11001 00000 10101 0 /11001 0000 1111 00 10111 0 /11111 0 101 000 11 0000 1001 00 /11001 00 /11111 00 /11001 00 /11111 0 /10011 0000 /11001 00 1011 000 11001 0 /10001 0000 /101 000 /1111 00 1 00000 1011 00000000 1101 0000 /10111 0 10101 00000 1101 00 /1101 000000 /101 000 /10011 00000 /10101 000 /11101 00 11111 0 /11111 0 /11

< 덧셈/뺄셈 사슬(윈도우 크기 5,총 길이 594) >

1.초기에 구해놓는 원소들(16개)

$$2,3,5,7,9,11,13,15,17,19,21,23,25,27,29,31$$

2.다음의 연산순서를 따라 계산한다. "$\rightarrow d$"는 바로 직전의 연산을 통해서 얻어진 숫자를 2배함을 의미하고, "$\rightarrow \#$"는 바로 직전의 연산을 통해서 얻어진 숫자에 해당 "#"를 더함을 의미한다. 처음에 27부터 시작한다(길이 578).

$$
\begin{aligned}
&27 \rightarrow d \rightarrow d \rightarrow d \rightarrow d \rightarrow d \rightarrow d \rightarrow -13 \rightarrow d \rightarrow d \rightarrow d \rightarrow d \rightarrow d \rightarrow d \rightarrow d \rightarrow 5 \rightarrow d \rightarrow d \rightarrow d \\
&\rightarrow d \rightarrow d \rightarrow d \rightarrow d \rightarrow d \rightarrow d \rightarrow d \rightarrow 9 \rightarrow d \rightarrow d \rightarrow d \rightarrow d \rightarrow d \rightarrow d \rightarrow d \rightarrow -25 \rightarrow d \rightarrow d \\
&\rightarrow d \rightarrow d \rightarrow 5 \rightarrow d \rightarrow d \rightarrow d \rightarrow d \rightarrow d \rightarrow d \rightarrow d \rightarrow -5 \rightarrow d \rightarrow d \rightarrow d \rightarrow d \rightarrow d \rightarrow d \rightarrow d \\
&\rightarrow d \rightarrow 17 \rightarrow d \rightarrow d \rightarrow d \rightarrow d \rightarrow d \rightarrow -19 \rightarrow d \rightarrow d \rightarrow d \rightarrow d \rightarrow d \rightarrow d \rightarrow 17 \rightarrow d \rightarrow d \\
&\rightarrow d \rightarrow d \rightarrow d \rightarrow d \rightarrow d \rightarrow 25 \rightarrow d \rightarrow d \rightarrow d \rightarrow d \rightarrow d \rightarrow d \rightarrow -27 \rightarrow d \rightarrow d \rightarrow d \rightarrow d \\
&\rightarrow d \rightarrow d \rightarrow d \rightarrow d \rightarrow d \rightarrow d \rightarrow 9 \rightarrow d \rightarrow d \rightarrow d \rightarrow d \rightarrow d \rightarrow d \rightarrow d \rightarrow d \rightarrow -15 \\
&\rightarrow d \rightarrow d \rightarrow d \rightarrow d \rightarrow d \rightarrow d \rightarrow d \rightarrow -21 \rightarrow d \rightarrow d \rightarrow d \rightarrow -3 \rightarrow d \rightarrow d \rightarrow d \rightarrow d \rightarrow d \\
&\rightarrow d \rightarrow d \rightarrow d \rightarrow d \rightarrow 25 \rightarrow d \rightarrow d \rightarrow d \rightarrow d \rightarrow d \rightarrow 5 \rightarrow d \rightarrow d \rightarrow d \rightarrow d \rightarrow d \rightarrow d \rightarrow d \\
&\rightarrow -9 \rightarrow d \rightarrow d \rightarrow d \rightarrow d \rightarrow d \rightarrow d \rightarrow 9 \rightarrow d \rightarrow d \rightarrow d \rightarrow d \rightarrow d \rightarrow d \rightarrow -5 \rightarrow d \rightarrow d \rightarrow d \\
&\rightarrow d \rightarrow d \rightarrow d \rightarrow d \rightarrow d \rightarrow -11 \rightarrow d \rightarrow d \rightarrow d \rightarrow d \rightarrow d \rightarrow d \rightarrow d \rightarrow 21 \rightarrow d \rightarrow d \rightarrow d \rightarrow d \\
&\rightarrow d \rightarrow d \rightarrow d \rightarrow -13 \rightarrow d \rightarrow d \rightarrow d \rightarrow d \rightarrow d \rightarrow d \rightarrow d \rightarrow -21 \rightarrow d \rightarrow d \rightarrow d \rightarrow d \rightarrow d \rightarrow d \\
&\rightarrow d \rightarrow -31 \rightarrow d \rightarrow d \rightarrow d \rightarrow d \rightarrow d \rightarrow d \rightarrow -21 \rightarrow d \rightarrow d \rightarrow d \rightarrow d \rightarrow d \rightarrow d \rightarrow d \rightarrow -25 \rightarrow d \rightarrow d \\
&\rightarrow d \rightarrow d \rightarrow d \rightarrow d \rightarrow d \rightarrow d \rightarrow -5 \rightarrow d \rightarrow d \rightarrow d \rightarrow d \rightarrow d \rightarrow d \rightarrow d \rightarrow d \rightarrow -23 \rightarrow d \rightarrow d \\
&\rightarrow d \rightarrow d \rightarrow d \rightarrow -13 \rightarrow d \rightarrow d \rightarrow d \rightarrow d \rightarrow d \rightarrow d \rightarrow d \rightarrow 23 \rightarrow d \rightarrow d \rightarrow d \rightarrow d \rightarrow d \rightarrow d \\
&\rightarrow d \rightarrow -29 \rightarrow d \rightarrow d \rightarrow -1 \rightarrow d \rightarrow d \rightarrow d \rightarrow d \rightarrow d \rightarrow d \rightarrow d \rightarrow d \rightarrow d \rightarrow d \rightarrow -19 \rightarrow d \rightarrow d \\
&\rightarrow d \rightarrow d \rightarrow d \rightarrow d \rightarrow d \rightarrow d \rightarrow d \rightarrow d \rightarrow -21 \rightarrow d \rightarrow d \rightarrow d \rightarrow d \rightarrow d \rightarrow -5 \rightarrow d \rightarrow d \\
&\rightarrow d \rightarrow d \rightarrow -1 \rightarrow d \rightarrow d \rightarrow d \rightarrow d \rightarrow d \rightarrow d \rightarrow d \rightarrow d \rightarrow d \rightarrow d \rightarrow d \rightarrow d \rightarrow -25 \rightarrow d \rightarrow d \\
&\rightarrow d \rightarrow d \rightarrow d \rightarrow d \rightarrow d \rightarrow d \rightarrow d \rightarrow 21 \rightarrow d \rightarrow d \rightarrow d \rightarrow d \rightarrow d \rightarrow d \rightarrow -25 \rightarrow d \rightarrow d \\
&\rightarrow d \rightarrow d \rightarrow d \rightarrow d \rightarrow d \rightarrow d \rightarrow 15 \rightarrow d \rightarrow d \rightarrow d \rightarrow d \rightarrow d \rightarrow d \rightarrow d \rightarrow 23 \rightarrow d \rightarrow d \rightarrow d \\
&\rightarrow d \rightarrow d \rightarrow d \rightarrow -31 \rightarrow d \rightarrow d \rightarrow d \rightarrow d \rightarrow 5 \rightarrow d \rightarrow d \rightarrow d \rightarrow d \rightarrow d \rightarrow 3 \rightarrow d \rightarrow d \rightarrow d \\
&\rightarrow d \rightarrow d \rightarrow d \rightarrow d \rightarrow d \rightarrow 9 \rightarrow d \rightarrow d \rightarrow d \rightarrow d \rightarrow d \rightarrow d \rightarrow d \rightarrow -25 \rightarrow d \rightarrow d \rightarrow d \rightarrow d \\
&\rightarrow d \rightarrow d \rightarrow d \rightarrow -31 \rightarrow d \rightarrow d \rightarrow d \rightarrow d \rightarrow d \rightarrow d \rightarrow d \rightarrow -25 \rightarrow d \rightarrow d \rightarrow d \rightarrow d \rightarrow d \rightarrow d \\
&\rightarrow d \rightarrow -31 \rightarrow d \rightarrow d \rightarrow d \rightarrow d \rightarrow d \rightarrow d \rightarrow -19 \rightarrow d \rightarrow d \rightarrow d \rightarrow d \rightarrow d \rightarrow d \rightarrow d \rightarrow d \rightarrow d \\
&\rightarrow -25 \rightarrow d \rightarrow d \rightarrow d \rightarrow d \rightarrow d \rightarrow d \rightarrow 11 \rightarrow d \rightarrow d \rightarrow d \rightarrow d \rightarrow d \rightarrow d \rightarrow d \rightarrow d \rightarrow 25 \rightarrow d \\
&\rightarrow d \rightarrow d \rightarrow d \rightarrow d \rightarrow d \rightarrow -17 \rightarrow d \rightarrow d \rightarrow d \rightarrow d \rightarrow d \rightarrow d \rightarrow d \rightarrow -5 \rightarrow d \rightarrow d \rightarrow d \rightarrow d \\
&\rightarrow d \rightarrow d \rightarrow d \rightarrow -15 \rightarrow d \rightarrow d \rightarrow d \rightarrow 1 \rightarrow d \rightarrow d \rightarrow d \rightarrow d \rightarrow d \rightarrow d \rightarrow d \rightarrow d \rightarrow d \rightarrow 11 \\
&\rightarrow d \rightarrow d \rightarrow d \rightarrow d \rightarrow d \rightarrow d \rightarrow d \rightarrow d \rightarrow d \rightarrow d \rightarrow d \rightarrow 13 \rightarrow d \rightarrow d \rightarrow d \rightarrow d \rightarrow d \\
&\rightarrow d \rightarrow d \rightarrow d \rightarrow d \rightarrow -23 \rightarrow d \rightarrow d \rightarrow d \rightarrow d \rightarrow d \rightarrow d \rightarrow 21 \rightarrow d \rightarrow d \rightarrow d \rightarrow d \rightarrow d \rightarrow d \\
&\rightarrow d \rightarrow d \rightarrow d \rightarrow 13 \rightarrow d \rightarrow d \rightarrow d \rightarrow d \rightarrow d \rightarrow d \rightarrow -13 \rightarrow d \rightarrow d \rightarrow d \rightarrow d \rightarrow d \rightarrow d \rightarrow d \\
&\rightarrow d \rightarrow d \rightarrow -5 \rightarrow d \rightarrow d \rightarrow d \rightarrow d \rightarrow d \rightarrow d \rightarrow d \rightarrow d \rightarrow -19 \rightarrow d \rightarrow d \rightarrow d \rightarrow d \rightarrow d \rightarrow d \\
&\rightarrow d \rightarrow d \rightarrow d \rightarrow d \rightarrow -21 \rightarrow d \rightarrow d \rightarrow d \rightarrow d \rightarrow d \rightarrow d \rightarrow d \rightarrow d \rightarrow -29 \rightarrow d \rightarrow d \rightarrow d \rightarrow d \\
&\rightarrow d \rightarrow d \rightarrow d \rightarrow 31 \rightarrow d \rightarrow d \rightarrow d \rightarrow d \rightarrow d \rightarrow d \rightarrow -31 \rightarrow d \rightarrow d \rightarrow d \rightarrow -3
\end{aligned}
$$