韓國SI學會誌
第3卷 第2號
2004年 11月

Java 기반의 D-클래스 계산 패키지 구현에 대한 연구

임범준* · 한재일*

A Study on the Implementation of a D-Class Computation Package based on Java

Bum Jun Lim* · Jae-il Han*

■ Abstract ■

Conventional and public-key cryptography has been widely accepted as a base technology for the design of computer security systems. D-classes have the potential for application to conventional and public-key cryptography. However, there are very few results on D-classes because the computational complexity of D-class computation is NP-complete. This paper discusses the design of algorithms for the efficient computation of D-classes and the Java implementation of them. In addition, the paper implements the same D-class computation algorithms in C and shows the performance of C and Java programming languages for the computation-intensive applications by comparing their execution results.

Keyword : Cryptography, D-Class, Security, NP-Complete, Boolean Matrix, Algorithm

1. 서 론

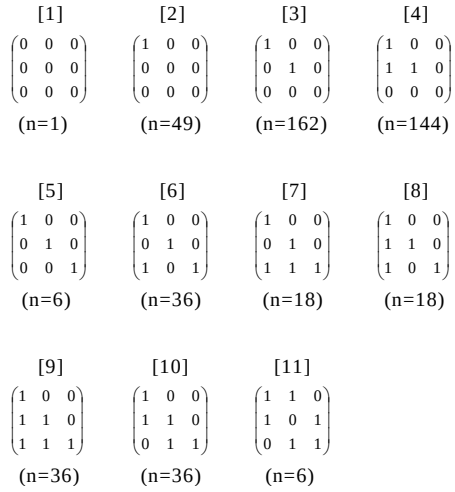
현재 널리 사용되고 있는 암호 기술로 대칭키나 공개키 기술을 들 수 있다. D-클래스[1]는 동치(equivalent) 클래스를 이용하여 대칭키나 공개키 암호 기술에서 암·복호 키를 더욱 안전하게 생성하는데 응용할 수 있는 가능성을 가지고 있다. 그러나 계산 복잡도가 NP-완전문제로서 현재 크기가 극히 제한된 행렬에 대해서만 결과가 알려져 있으며, 응용방법을 연구하기 위해서는 D-클래스에 대한 보다 많은 결과가 필요하다.

D-클래스 계산은 원소가 0과 1 값만을 갖는 불

* 국민대학교 컴퓨터학부

리언 행렬에서 D-클래스 조건을 만족시키는 관계식에 대하여 효율적인 연산을 할 수 있는 알고리즘이 필요하다. 그러나 현재 D-클래스 계산에 대한 이론적인 연구만이 있을 뿐이며 D-클래스 계산 알고리즘에 대한 연구는 아직 보이지 않고 있다. 일반적인 행렬 연산이나 특수한 행렬 연산에 대하여 매우 많은 알고리즘이 연구되었으나[2 - 6], 적용할 수 있는 문제 범위가 한정되어 있어 불리언 행렬에서 D-클래스 조건을 만족시키는 관계식에 대하여 최적화된 연산을 할 수 있는 특수 목적 알고리즘에 적용하기에는 부적합하다. 따라서 D-클래스를 효율적으로 계산할 수 있는 알고리즘의 설계 및 구현이 필요하며 본 논문은 효율적인 D-클래스 계산 알고리즘을 설계하고 이를 Java 언어를 사용하여 패키지로 구현하였다.

D-클래스의 정의는 다음과 같다[1]. 집합 F는 0과 1의 두 원소로 구성되며, $M_n(F)$는 행렬의 원소가 F의 원소를 값으로 갖는 모든 n×n 불리언 행렬의 집합이다. $M_n(F)$에 속하는 임의의 두 행렬 A, B에 대하여 AX=C, CY=A, UC=B, VB=C를 만족시키는 C, X, Y, U, V가 $M_n(F)$에 존재할 때 A와 B는 R_D관계가 성립한다. R_D는 반사성(reflexive), 대칭성(symmetric), 전이성(transitive)을 만족하는 동치 관계이며, D-클래스는 R_D에 의하여 동치 관계에 놓여 있는 n×n 불리언 행렬의 집합으로 정의한다. 따라서, $M_n(F)$에 속한 임의의 행렬 A에 대한 D-클래스는 다음과 같이 정의된다.



[그림 1] $M_3(F)$에 대한 D-클래스 예

[정의 1]

$$D_A = \{B \in M_n(F) : \exists C, X, Y, U, V \in M_n(F) \; such \; that \; AX = C, CY = A, UC = B, VB = C\}$$

[그림 1]은 $M_3(F)$로부터 얻을 수 있는 11개의 D-클래스, 각 클래스에 속하는 하나의 행렬, 그리고 괄호에 동치관계에 있는 3×3 행렬의 개수를 보이고 있다.

본 논문의 구성은 다음과 같다. 2장과 3장은 각각 기본적인 D-클래스 계산 순차 알고리즘과 개선된 순차 알고리즘을 논하며, 4장은 D-클래스 계산 병렬 알고리즘과 Java 언어의 성능에 대하여 기술한다. 5장은 결론 및 향후 과제에 대하여 논한다.

2. D-클래스 계산 순차 알고리즘

순수하게 [정의 1]에 따라 D-클래스를 계산하는 순차 알고리즘은 [그림 2]와 같다. 이 순차 알고리즘은 5개의 모든 순환문에서 $M_n(F)$에 속한 모든 행렬에 대하여 명시된 행렬 곱셈을 수행한다.

```
D_A ← φ

for each A in M_n(F)
  for each X in M_n(F)
    C ← AX
    for each Y in M_n(F)
      if A = CY
        for each U in M_n(F)
          B ← UC
            for each V in M_n(F)
              if C ← VB
                insert B to D_A
```

[그림 2] 순차 알고리즘

따라서 행렬의 원소 개수가 m일 때 이 알고리즘의 계산복잡도는 $O(2^{5m})$, 즉 $O(2^m)$이 되어 D-클래스 계산 순차 알고리즘의 계산복잡도가 NP-완전문제라는 것을 보인다[2]. 이 계산복잡도로 인해 현재 D-클래스는 극히 제한된 크기의 행렬에 대해서만 알려져 있다.

위에서 언급하였듯이 일반 행렬이나 불리언 행렬 계산에 대한 연구는 많이 있으나[2 - 6] 적용할 수 있는 문제 범위가 한정되어 있어 D-클래스의 효율적인 계산을 위해서는 D-클래스 계산에 최적화된 알고리즘이 필요하다. 본 논문은 이를 위해 D-클래스를 효율적으로 계산할 수 있도록 개선된 순차 알고리즘과 병렬 알고리즘을 제시한다.

3. 개선된 순차 알고리즘

효율적인 행렬의 곱셈을 위하여 [정의 1]의 VB=C 식을 CY=A 식에 대입하고 AX=C 식을 UC=B의 식에 대입하여 UAX=B, VBY=A의 식을 얻을 수 있다. 따라서 D-클래스는 이 변형된 행렬 곱셈식을 사용하여 다음과 같이 재정의 할 수 있다.

[정의 2]

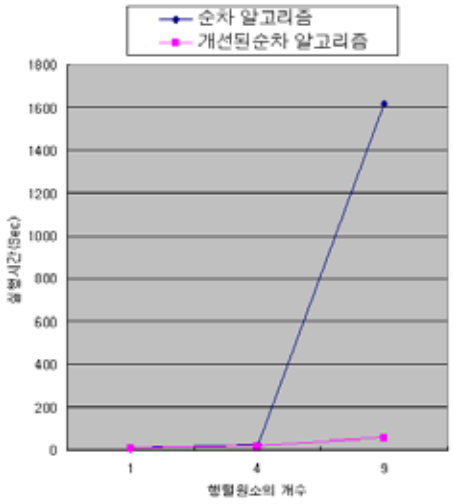
$$D_A = \{B \in M_n(F) : \exists X, Y, U, V \in M_n(F)$$
$$such\ that\ UAX = B, VBY = A\}$$

```
DA, SB←ϕ
for each A in Mn(F)
  for each A in Mn(F)
    T←UA
    for each X in Mn(F)
      if TX in Mn(F)
        insert TX to SB
  for each B in SB
    for each V in Mn(F)
      T1←VB
      for each Y in Mn(F)
        if A =T1Y
          insert B to DA
          remove B from Mn(F)
```

[그림 3] 개선된 순차 알고리즘

[그림 3]은 [정의 2]를 기반으로 하여 설계한 개선된 D-클래스 계산 순차 알고리즘을 보이고 있다. 이 알고리즘은 [정의 1]에 나타난 네개의 행렬 곱셈식을 연속된 중첩 순환문으로 처리하는 대신 [정의 2]에 나타난 두개의 행렬 곱셈식을 수행한다. 특히 네번째 내부 순환문에서 반복 계산해야 할 행렬의 개수는 두번째와 세번째 내부 순환문에서 계산되는 UAX 행렬의 수로 제한되며, 마지막 순환문에서 A와 동치관계인 행렬을 발견하는 대로 $M_n(F)$ 집합에서 제거하여 전체 순환문의 반복 회수를 크게 개선하였다.

UAX를 계산하는 첫번째 내부 순환문의 계산복잡도는 $O(2^{2m})$이고 VBY를 계산하는 두번째 내부 순환문의 계산복잡도는 $O(2^{3m})$이다. 따라서 외부 순환문의 계산복잡도는 A가 2^m개가 있으므로 $O(2^m(2^{2m} + 2^{3m}))$이 되어 개선된 순차 알고리즘의 계산복잡도는 $O(2^{4m})$이 되어 결국 $O(2^m)$이 된다. 그러나 각 알고리즘의 계산복잡도가 유도과정에서 $O(2^{5m})$과 $O(2^{4m})$으로 나타났던 것처럼 두 알고리즘의 실제 계산시간은 [정의 1]에 의한 순차 알고리즘보다 [정의 2]에 의한 순차 알고리즘에서 개선되었으며 [그림 4]는 두 알고리즘의 실행 결과를 보인다.



[그림 4] 알고리즘 실행결과

4. D-클래스 계산 병렬 알고리즘

본 논문에서는 앞의 결과를 보다 향상시키기 위해 3개의 쓰레드 사용하여 병렬 알고리즘을 설계하였다. $M_n(F)$에 속하는 행렬의 인덱스를 다음 식에 의하여 주었다고 가정하면,

$$i = \sum_{i,j=1}^{n} a_{ij} 2^{(i-1)n+(j-1)}$$

$M_n(F)$는 다음과 같이 정의할 수 있다.

$$M_n(F) = \{M_i \mid 0 \le i \le 2^n - 1\}$$

따라서 다음과 같이 $M_n^k(F)$을 정의할 수 있다.

$$M_n^k(F) = \{M_i \mid i = k \pmod 3,\ 0 \le i \le 2^n - 1\}$$

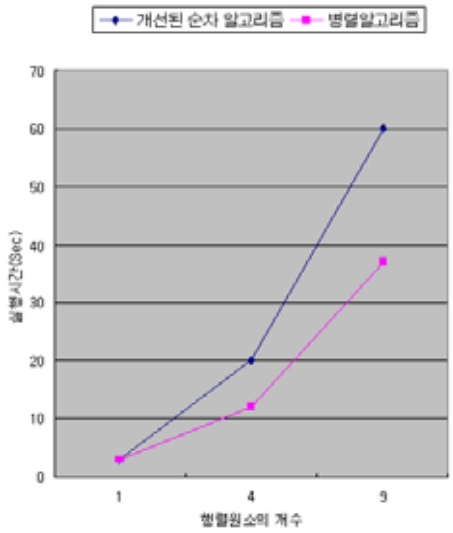
[그림 5]는 이 정의를 이용하여 기술한 D-클래스 계산 병렬 알고리즘이다. 첫번째 병렬문은 UAX를 계산하기 위해 세개의 쓰레드를 이용하며, 각 쓰레드는 하나의 행렬 A에 대하여 $M_n(F)$의 행렬을 1/3씩 나누어 계산한 결과를 S_B에 저장한다.

```
D_A, S_B ← φ
for each A in M_n(F)
  do in parallel
    thread i=0, 1, 2 :
      for each U in M_n^i(F)
        T ← UA
        for each X in M_n(F)
          if TX in M_n(F)
            insert TX to S_B
  end do
  do in parallel
    thread i=0, 1, 2 :
      for each B in M_n^i(F)
        if B is in S_B
          for each V in M_n(F)
            T_1 ← VB
            for each Y in M_n(F)
              if A = T_1 Y
                insert B to D_A
                remove B from M_n(F)
  end do
```

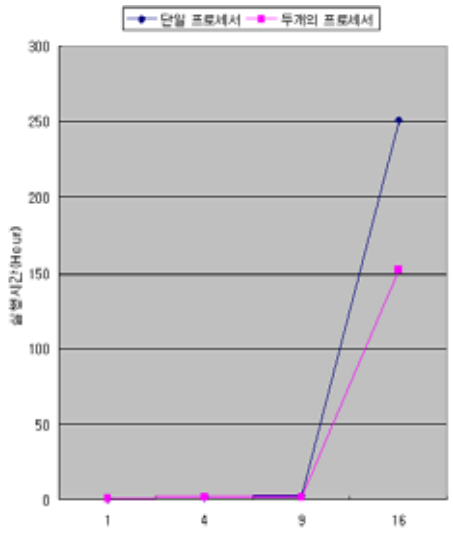
[그림 5] 병렬 알고리즘

UAX 계산이 모두 끝난 후 두번째 병렬문에서도 첫번째 병렬문과 유사하게 VBY의 계산을 위해 세개의 쓰레드를 이용하며, 각 쓰레드는 S_B의 각 행렬에 대하여 $M_n(F)$의 행렬을 1/3씩 나누어 VBY를 계산한다.

[그림 6]은 하나의 프로세서를 사용하여 얻은 D-클래스 계산 병렬 알고리즘의 실행결과를 나타

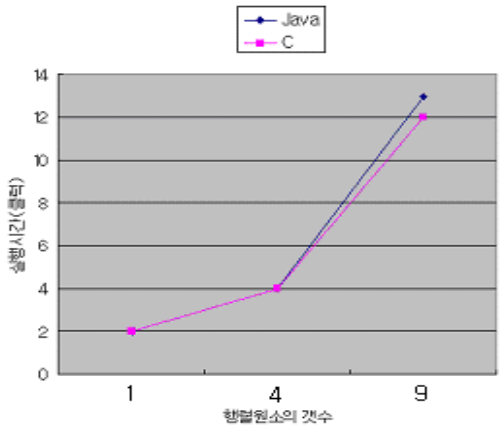


[그림 6] 병렬 알고리즘 실행결과



[그림 7] 프로세서 수에 따른 성능 비교

내며 순차 알고리즘보다 약 40%정도로 실행시간이 개선되는 것을 보여주고 있다. 또한 병렬 알고리즘은 [그림 7]에서 보듯이 2개의 프로세서를 사용하였을 때 그리고 행렬의 크기가 클수록 보다 좋은 결과를 나타내었다. 본 논문에서는 병렬 알고리즘을 Java와 C 언어로 구현하여 각 언어에 대한 실행시간을 측정하여 성능 비교를 하였다. [그림 8]과 <표 1>은 두 언어의 실행 결과를 보이고 있으며 Java 언어가 계산 위주의 응용에 대해서도 다른 프로그래밍 언어의 성능과 거의 같은 수준임을 나타내고 있다.



[그림 8] 두 언어의 실행결과

〈표 1〉 두 언어의 실행 결과 비교

테스	설	C++ (초)	Java (초)
정수 나눗세	정수 나눗셈 1000만 번 수행	1.8	1.8
실수 나눗셈	실수 나눗셈 1000만 번 수행	1.6	1.6
정적 함수	정수 나눗셈을 하는 정적 함수 1000만 번 호추	1.8	1.8
멤버 함수	정수 나눗셈을 하는 멤버 함수를 1000만 번 호출	1.8	1.8
가상 함수	정수 나눗셈을 하는 가상 함수를 1000만 번 호출	1.8	1.8

5. 결 론

D-클래스는 대칭키나 공개키 암호 기술에 사용될 수 있는 가능성을 가지고 있으나 계산복잡도가 NP-완전문제로서 현재 극히 제한된 결과만이 알려져 있다. 본 논문은 이러한 D-클래스의 계산을 효율적으로 할 수 있는 알고리즘의 설계와 Java 기반 D-클래스 계산의 패키지 구현에 대하여 논하였다. 또한 D-클래스 계산 알고리즘을 C 언어로 구현하여 Java 패키지와 실행결과를 비교함으로써 높은 계산복잡도를 가지는 응용에 대한 Java 언어의 성능을 검증하였다.

본 논문의 알고리즘은 많은 개선이 요구되며 이를 위해 D-클래스를 정의한 수식 변형, 개선된 병렬 알고리즘 등에 대한 많은 연구가 필요하다. 또한 본 논문은 간단한 병렬 알고리즘을 설계하고 구현 하였으나 앞으로 병렬 컴퓨터나 클러스터 컴퓨팅 등의 환경에서 D-클래스 계산 병렬 알고리즘에 대한 연구가 요구된다[7].

참 고 문 헌

[1] Rim, D. S. and Kim, J. B., "Tables of D-Classes in the semigroup Bn of the binary relations on a set X wit n-elements," *Bull. Korea Math. Soc.*, Vol.20, No.1(1983), pp.9-13.

[2] Gunnels, J. et al., A *Flexible Class of Parallel Matrix Multiplication Algorithms*, Department of Computer Sciences The University of Texas at Austin, 1995.

[3] Golub, G. H. and Van Loan, C. F., *Matrix Computation*, The Johns Hopkins' University Press, 1983.

[4] Butler, K. K., "On (0, 1)-matrix semigroups," *Semigroup Forum*, Vol.3(1971), pp.74-79

[5] Fox, G., Otto S., and Hey A., "Matrix algorithms on a hypercube I : matrix Multiplication," *Parallel Computing*, Vol.3(1987), pp.17-31.

[6] Leighton F. T., *Parallel Algorithms And Architectures : Arrays·Trees·Hypercubes*, Morgan Kaufmann, 1992.

[7] Wilkinson, B. and Allen M., *Parallel Programming with MPI*, Prentice Hall, 1999.

◈ 저 자 소 개 ◈

임 범 준 (limbj6518@cs.kookmin.ac.kr)

관동대학교 컴퓨터학과에서 학사를 졸업하고, 국민대학교 컴퓨터학과에서 석사를 재학 중이다. 현재는 분산처리, 병렬 알고리즘 등을 연구 중이다. 관심분야는 분산 시스템, 웹 서비스, 전자상거래 보안, 유비쿼터스 컴퓨팅 등이다.

한 재 일 (jhan@kookmin.ac.kr)

연세대학교에서 이학사, 미국 Syracuse University에서 전산학 석사와 박사학위를 취득하고, 국민대학교 컴퓨터학부 부교수로 재직 중이다. 현재 분산처리, 객체지향 시스템과 감정시스템을 연구 중이다. 관심분야는 분산 시스템, 객체지향 시스템, 컴퓨터 및 네트워크 보안, 지능형 시스템 등이다.