情 報 保 護 學 會 誌
第 16 卷 第 4 號, 2006. 8

인수분해 전용 하드웨어 연구 동향*

이 상 진**, 김 창 한***, 장 남 수**, 윤 택 영**

요 약

NP-Hard 문제인 정수의 소인수분해 알고리즘의 연구와 구현은 1978년 RSA 암호의 개발과 함께 암호학에서 중요한 문제로 부각되었으며 지난 25년간 이 분야에서 많은 발전이 이룩되었다. QS 인수분해 알고리즘과 NFS 인수분해 알고리즘이 최근까지도 RSA-challenge를 분석하기 위한 도구로 사용되었고, NFS가 가장 효율적인 것으로 알려져 있다. 그러나 인수분해 대상 정수의 크기가 커짐에 따라 기존의 소프트웨어 기반의 접근 방법으로 분석하는 것은 점차 어려워지고 있다. 99년도 CHES Rump Session에서 Shamir에 의해 제안된 TWINKLE은 인수분해 알고리즘의 연구에 새 지평을 마련하였다. TWINKLE는 기존과는 근본적으로 다른 접근 방법으로 수행되는 인수분해 전용 하드웨어 장비이다. TWINKLE이 발표된 이후 TWIRL와 SHARK 등 다양한 인수분해 전용 하드웨어들이 제안되었고, 이는 인수분해 방법론 연구에서 새로운 방향이 되고 있다. 본 논문에서는 이와 같은 인수분해 전용 하드웨어 연구 동향에 대해 살펴보고, 각 장비들의 효율성을 비교·분석하도록 한다.

Ⅰ. 서 론

암호 기법은 정보의 상품성을 안전하게 지켜주는 기법의 하나로 정보화 사회에서 유용하게 사용되고 있다. 현재 정보화 시대에서 정보의 상품성을 제공하기 위해 사용하는 많은 프로토콜들은 RSA[12]를 기본 도구로 사용하고 있다. RSA 외에도 ElGamal 암호화 기법이나 Diffie-Hellman 키 교환 프로토콜도 오랫동안 연구되어왔고 ECC와 같은 새로운 암호시스템이 무선 통신 등의 분야에서 많이 사용되고 있긴 하지만 기존에 구축되어있는 인프라에서는 RSA가 많이 사용되고 있고 앞으로도 계속적으로 사용될 것이다. 이와 같이 정보화 사회를 구현하는 도구로 널리 사용되고 있는 RSA 암호화 기법은 인수분해 문제를 기반으로 설계되어 있고, 이에 따라 NP-Hard 문제인 정수의 소인수분해 알고리즘의 연구와 구현은 암호학에서 가장 중요한 문제의 하나가 되어왔으며 지난 25년간 이 분야에서 많은 발전이 이룩되었다.

기존에 알려져 있는 인수분해 알고리즘으로는 ECM (Elliptic curve method)[4], QS (Quadratic Sieve) 인수분해 알고리즘[11], MPQS (Multiple Polynomial Quadratic Sieve) 인수분해 알고리즘[16] 그리고 GNFS (General Number Field Sieve) 인수분해 알고리즘[2]이 있는데 그 중에서 GNFS 인수분해 알고리즘이 큰 수를 대상으로 구현되는 경우 가장 효율적인 것으로 알려져 있다. 과거에 비해 효율적인 알고리즘이 개발되고 연산환경이 좋아짐에 따라 인수분해가 가능한 정수의 크기는 점점 커지고 있다. 최근의 인수분해 기록은 RSA-challenge 인 RSA-576으로 독일의 연구원들과 세계 각국의 협력을 통해 수행되었다. 그동안 수행된 RSA-security사의 RSA-challenge[8] 인수분해 기록은 표 1에서 확인할 수 있다.

그 동안 인수분해 알고리즘에 대한 연구에서 괄목할 만한 결과 중 하나는 시빙 단계를 위해서 인수분해 전용 하드웨어가 제안되었다는 것이다. GNFS와 같이 시빙을 기반으로 구성된 인수분해 알고리즘을 구성하는 단계 중에서 시빙을 수행하기 위한 과정의 연산이 시간복잡도가 가장 높으므로 이 단계를 효율적으로 구성하는 것이 중요하다. 인수분해를 위한 전용 하드웨

* 본 연구는 정보통신부 및 정보통신연구진흥원의 대학 IT연구센터 육성.지원사업의 연구결과로 수행되었음.
** 고려대학교 정보보호대학원 (sangjin@korea.ac.kr, {nschang,taekyoung}@cist.korea.ac.kr)
*** 세명대학교 정보보호학과 (chkim@semyung.ac.kr)

〔표 1〕 인수분해 완료된 RSA-challenge

대상 정수	수행된 년도	인수분해 방법
RSA-100	1991	MPQS
RSA-110	1992	MPQS
RSA-120	1993	MPQS
RSA-129	1994	MPQS
RSA-130	1996	NFS
RSA-140	1999	NFS
RSA-155	1999	NFS
RSA-160	2003	GNFS
RSA-576	2003	GNFS
RSA-200	2005	GNFS

어는 이와 같은 시빙 단계를 대상으로 구성된다. 최초의 시빙 전용 하드웨어 장비인 TWINKLE〔13〕은 Adi Shamir에 의해 1999년에 제안되었다. 광전자 기술을 기반으로 설계된 TWINKLE은 기존의 소프트웨어 기반 시빙에 비해 500-1000배 정도 효율적인 것으로 분석되었다. 이후 분석된 결과에서 실제 효율성이 제안되었을 당시 주장된 것보다 작다는 것이 밝혀졌지만 효율적인 인수분해를 구현하기 위한 새로운 방향을 제시했다는 면에서 큰 의미를 지닌다.

TWINKEL이 제안된 이후 다양한 방법으로 인수분해 전용 하드웨어 장비 설계 기술에 대한 연구가 수행되었다. 2003년에는 TWINKEL을 제안한 Adi Shamir과 Eran Tromer가 Parallel processing pipeline 구조를 기반으로 TWIRL〔14〕을 제안하였다. 최근 2005년에는 Jens Franke 등이 저렴한 비용으로 현실적인 구현이 가능한 인수분해 전용 하드웨어인 SHARK〔5〕를 제안하였는데, 이는 GNFS 에 가장 효율적으로 적용되는 시빙 방법인 lattice 시빙 방법에 적합하게 설계되었으며 1024비트 정수 인수분해를 수행하기 위한 목적에 적합하게 설계되었다. 이러한 시빙 전용 하드웨어들의 개발은 단순히 PC의 성능에 의존해 분석되던 인수분해 문제의 안전성이 크게 낮추는 결과를 낳았다. 즉, 시빙 하드웨어의 개발은 현재 가장 널리 사용하는 1024비트의 합성수를 인수분해 하기 위한 비용의 감소를 가져왔다. 따라서 시빙 하드웨어 장비의 개발은 정수의 인수분해에 있어서 새로운 지평을 열었다.

본 논문에서는 인수분해 연구에서 새로운 방향을 제시한 시빙 하드웨어 설계 기술 관련 연구 동향에 대해 살펴보고 각 장비의 효율성을 비교한다.

II. 시빙 기반 인수분해 알고리즘

시빙 전용 하드웨어 장비에 대한 논의를 진행하기에 앞서 시빙을 기반으로 수행되는 인수분해 알고리즘에 대해 간략히 살펴보도록 하자.

1. 시빙 기반 인수분해 알고리즘의 개괄

시빙 기반 인수분해 알고리즘은 인수분해를 수행하기 위한 합성수 n을 모듈로 값으로 고려했을 때 다음의 관계식을 만족하는 두 정수 x, y를 찾기 위한 목적으로 수행된다.

$$x^2 = y^2 \mod(n).$$

위의 관계를 만족하는 x,y를 찾으면 $\gcd(x \pm y, n)$를 계산함으로써 n의 인수를 찾을 수 있다. 시빙 과정은 위의 조건을 만족하는 두 정수를 구성하기 위한 값들을 찾기 위한 과정을 의미한다.

어떤 정수가 주어진 인수기저 집합의 원소들로 인수분해가 되는 경우에 그 정수를 smooth 하다고 하자. 본 논문에서는 특정 알고리즘을 대상으로 시빙 단계를 설명하지 않고 일반적인 개념만 설명하도록 한다. 우선 B를 smoothness의 기준이 되는 경계값(boundary)보다 작은 소수로 구성된 인수기저라고 하자. B의 원소들로 완전히 나누어지는 정수는 smoothness를 만족한다고 정의된다. 시빙 과정은 B에 대해서 v_i와 $s_i = v_i^2 \mod(n)$가 smooth인 순서쌍 (v_i, s_i)을 찾는 연산을 수행한다. 시빙 과정에서 다음과 같이 m개의 순서쌍을 찾았다고 하자:

$$s_1 = v_1^2 \mod(n), s_2 = v_2^2 \mod(n),$$
$$\cdots, s_m = v_m^2 \mod(n)$$

m개의 순서쌍 중에서 어떤 정수 S에 대해

$$\prod_{i \in T} s_{v_i} = S^2 \mod(n)$$

를 만족하는 집합 $T \subseteq 1, 2, \cdots,$ m가 존재하면 다음과 같은 관계식을 통해 서로 다른 제곱수가 되는 두 정수를 생성할 수 있다.

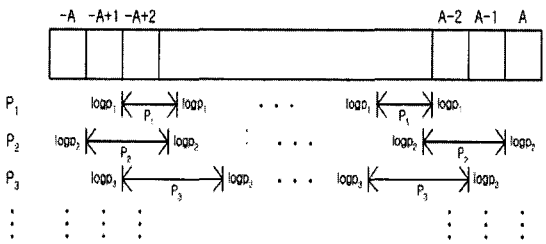
$$S^2=\prod_{i\in T}s_{v_i}=\prod_{i\in T}v_i^2=(\prod_{i\in T}v_i)^2 \mod(n).$$

두 정수는 $x=S$와 $y=\prod_{i\in T}v_i$로 계산되고, 두 정수는 의도된 결과식 $x^2=y^2 \mod(n)$을 만족한다. QS 알고리즘와 GNFS 알고리즘도 시빙을 기반으로 수행되므로 위에서 설명된 방법으로 서로 다른 제곱수를 구성하는 x, y를 찾기 위한 과정이 수행된다.

2. 시빙 기법

시빙 기법에는 선(line) 시빙[2]과 래티스(lattice) 시빙[7]이 있는데 시빙 전용 하드웨어 장비는 시빙 기법의 종류에 상관없이 수행될 수 있으므로 선 시빙을 기반으로 논의를 진행한다.

시빙이 수행되기 위해서는 인수기저 B가 정의되고, smoothness를 판단하게 될 대상을 명시하는 시빙구간(sieving interval)이 정의된다. 시빙 대상 a의 구간은 $-A<a<A$ 라고 하자. 선 시빙의 기본적인 아이디어는 다음과 같다. 우선 시빙 구간 값으로 계산되어 smoothness가 판단되어야 하는 원소를 표현할 수 있는 값이 배열에 저장된다. 배열에는 각 원소의 로그값이 저장된다. smoothness를 판단해야 하는 값이 주어지면, 각 인수기저의 원소로 나누어지는지 확인한 뒤 해당 값을 나누는 것으로 판단되는 인수의 로그값을 해당 값의 로그값이 저장된 배열에 있는 값에서 뺀다. 이와 같은 과정을 모든 인수에 대해서 수행한다. 배열에 저장된 값이 특정 범위의 값보다 작으면, 해당 위치에 저장된 값이 인수기저의 원소들로 나누어진다는 것을 의미하므로 찾고자 하였던 smoothness를 만족하는 값을 찾게 되는 것이다. 선 시빙의 수행 과정은 그림 1에서 확인할 수 있다.



[그림 1] 선 시빙의 수행 과정에 대한 모식도.

그림의 상단에 위치한 사각형은 시빙 대상이 저장된 배열을 의미한다. 시빙의 수행 과정은 P_1, P_2 그리고 P_3 순으로 나누어지는 여부를 확인한다. P_1에 대해 나누어지는 시빙 대상을 찾는 과정이 가장 먼저 수행되는데, 시빙 구간의 값 중에서 $-A+1$에 의해 생성되는 시빙 대상이 P_1에 의해 나누어지고, 이에 따라 P_1를 간격으로 동일한 소수에 의해 나누어지므로 해당 위치의 배열에서 $\log P_1$에 해당하는 값을 빼거나 더하는 연산을 수행하게 된다. P_3에 대한 처리가 완료된 후에 $-A+1$에 대응되는 배열에는 $\log P_1$와 $\log P_3$가 더해지거나 빼지게 된다. 이와 같은 과정을 모든 인수기저의 원소인 소수들에 대해 수행한 뒤, 해당 배열에 저장된 값이 특정 범위보다 크거나 작으면 smooth인 것으로 판단한다.

선 시빙에서는 배열의 특정 위치가 인수기저의 원소 p로 나누어지는 경우, p를 주기로 동일한 인수로 나누어지는 특성을 사용하여 시빙을 수행한다. 동일한 인수 p의 로그값을 p를 주기로 위치한 배열에서 차감함으로써 하나의 인수에 대해 나누어지는 모든 배열을 한 번에 처리한다. 이와 같이 구성하지 않는 경우 각 배열에 저장된 값이 인수기저의 원소들로 나누어지는지를 확인하기 위해 인수기저의 원소 개수만큼의 나눗셈을 수행해야 하므로, 선 시빙 방법이 효율적으로 smooth인 값을 찾을 수 있음을 알 수 있다.

Ⅲ. 시빙 하드웨어

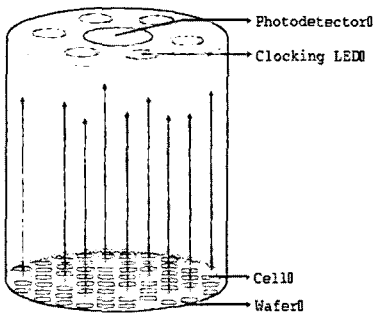
본 절에서는 NFS에서 가장 큰 시간과 비용을 필요로 하는 시빙 연산부의 하드웨어 구성에 관하여 논한다. 시빙 하드웨어로는 대표적으로 TWINKLE, TWIRL, SHARK 등이 있다.

1. TWINKLE

Adi Sharmir는 CHES'99 Rump Session에서 TWINKLE(The Weizmann Institute Key Locating Engine)이라는 시빙을 위한 전용 하드웨어 디바이스를 발표했다[9,13]. TWINKLE은 광전자 기술을 기반으로 설계된 시빙 전용 장비로 최대 10GHz에서 동작할 수 있는 웨이퍼 기반의 하드웨어 장비이며 0.01초안에 200,000개의 인수기저에 대해 100,000,000개의 정수의 smoothness를 확인할 수 있는 매우 효율적인 시빙 장비이다.

시빙 과정에서 smoothness를 확인하는데 걸리는

시간만을 기준으로 비교하면 기존의 시빙 방법에 비해 500배에서 1,000배 효율적인 결과이다. 하드웨어의 특성상 장비를 설계하기 위한 비용을 포함한 초기 생산 비용은 많이 요구되지만, 설계가 이루어진 이후 각 장비를 생산하기 위한 비용은 $5,000 정도로 고성능 PC나 워크스테이션과 비슷하다.



〔그림 2〕 TWINKLE의 구조

TWINKLE이 올바르게 동작하기 위해 여러 대의 PC가 필요하게 되어 비용이 증가되었기 때문에 사용자원대비 효율성이 처음 고려된 수준에 비해 낮아졌다. 그러나 TWINKLE의 개발은 효율성의 향상이라는 측면 외에도 기존에 진행되었던 소프트웨어적인 접근이나 하드웨어로 단순히 인수분해 알고리즘을 구현하는 방식과는 달리 시빙 단계의 효율성을 증가시키는 것을 목적으로 새로운 하드웨어적인 접근을 시도하였다는 점에서 큰 의미를 가진다.

1.1 TWINKLE의 구성

TWINKLE은 효율적인 시빙 과정을 위해 설계된 광학장비로 직경이 6인치, 높이가 10인치인 불투명한 실린더 관으로 둘러싸인 단순한 형태의 광전자장비이다. 클럭 단위로 시빙 구간의 원소를 한 개씩 처리하는데, 대상 원소의 smoothness는 장비에 장착된 LED의 발광에 의존한다. LED는 실린더 관의 하단에 위치하고 있고 각각 다른 주기로 발광한다. 각 LED는 인수기저의 원소를 표현해야 하므로 발광 주기와 발광 강도는 인수의 크기와 관련되어 있다. 실린더의 상단에는 하단을 구성하고 있는 LED에서 발광된 빛을 수합하는 광검출기와 하단의 LED들이 동작하는 시점을 동일하게 구성하기 위한 클럭 LED로 구성되어 있다. 하단의 LED들은 할당된 인수의 특성에 따라 일정 시점에서 발광하게 되는데, 상단의 광검출기는 이를 수합하여 빛의 총합을 계산한다. 광검출기는 수합된 빛의 합이 정해진 기준보다 큰 경우에 해당 단계에서 검사된 시빙 대상이 smoothness integer가 될 가능성이 높은 것으로 판단한다. 각 LED에서 발광되는 빛의 크기는 LED에 할당된 인수의 로그값의 크기를 나타내는 값이므로 수합된 빛의 합이 특정 바운드 값보다 크다는 것은 해당 대상을 나누는 인수들의 로그값의 합이 인수분해 알고리즘에서 고려하는 경계값보다 크다는 것을 의미한다. 이와 같은 과정으로 smoothness integer가 될 가능성이 높은 시빙 대상을 선정하게 되면 연결된 PC에 알리고, PC에서는 trial division을 수행함으로써 smoothness를 명확하게 판단한다. 시빙 단계에서 smooth integer가 나타나는 빈도는 매우 낮기 때문에 PC에서 수행되는 trial division은 전체 연산을 수행함에 있어 병목구간으로 작용하지 않는다. 이와 같은 과정으로 수집된 smooth integer들은 기존의 QS 인수분해 알고리즘이나 GNFS 인수분해 알고리즘에서 시빙 단계 이후에 처리한 것과 마찬가지로 행렬 연산과정을 수행하기 위한 데이터로 사용한다.

PC에서 구현된 시빙에서는 smoothness를 확인할 대상들을 배열에 저장하고, 각 인수기저의 원소들에 대한 반복적인 검사를 통해 smoothness를 만족할 가능성이 높은 대상들을 선별한다. 따라서 검사대상들을 저장하기 위한 배열을 위한 공간이 필요하고, 각 인수기저의 원소로 나누어지는지 검사하고 해당 원소의 로그 값을 더하거나 빼는 연산을 수행하기 위한 시간적 비용이 필요하다. TWINKLE의 경우에는 각 LED가 인수기저의 원소인 소수의 역할을 수행한다. 그리고 클럭 단위로 반복되는 루프는 smoothness를 확인하고자 하는 대상값에 대응되어 매 클럭마다 하나의 대상 값에 대한 smoothness를 확인한다. 즉, 기존에는 시간적 자원을 사용하여 인수기저의 원소들에 의해 나누어지는지를 검사하였다면, TWINKLE은 시간적 자원을 사용해서 주어진 값의 smoothness를 확인한다. TWINKLE의 이와 같은 구성은 기존에 구현되던 시빙의 구현과 설정이 반대가 된다.

TWINKLE과 기존 PC기반 인수분해 알고리즘 사이의 차이점을 정리하면 다음과 같다.

- 일반적인 PC에 탑재된 실리콘 RAM chip은 약 100 MHz에서 동작하는 반면에 GaAs (Gallium Arsenide) wafer로 구현하면 10 GHz이상의 속도의 클럭으로 기능을 수행할 수 있다.

〔표 2〕 기존의 PC 기반 장비와 TWINKLE의 768비트 정수 인수분해 시 필요한 자원 및 시간

구 분	TWINKLE	기존의 PC기반 인수분해 방법
시빙기법	Line Sieving	Line Sieving
사용자원	TWINKLE 2,500대+ 10GB 메모리의 PC 40,000대	5GB 메모리의 PC 90,000대
비 용	$ 8M	$ 13M
소요시간	1년	1년

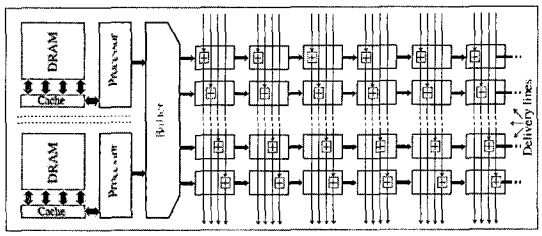
- 각 LED가 발광하는 빛을 광 검출기가 합산함으로써 한 클럭안에 주어진 대상을 나누는 인수의 로그값을 더할 수 있다.
- TWINKLE은 시빙을 수행하는 과정에서 큰 저장 공간을 요구하지 않는다는 특징을 가지고, 이 특징은 효율성 및 수행 기간의 측면에서 장점이 된다.
- TWINKLE은 클럭 단위로 대상 값들을 처리한다는 특성에 의해 배열을 초기화 하는 등의 추가 비용이 발생하지 않는다.

본 절의 서두에서 TWINKLE이 기존의 결과와 비교하여 500배에서 1,000배 이상 빠르다고 하였으나 이는 단지 시빙 장비의 속도만을 고려한 경우이다. TWINKLE의 경우 PC와 연동해야하므로 클럭 및 추가 시간이 늘어나 실질적으로 큰 효율성을 기대할 수 없었다.

2. TWIRL

1999년에 수백 개의 워크스테이션을 포함한 분산 계산으로 수개월에 걸쳐 512 비트 RSA 키를 인수분해 하는데 성공했으나 1024 비트 크기의 키는 향후 15년에서 20년 동안 안전할 것이라 당시 믿어졌다.

본 소절에서는 TWIRL(The Weizmann Institute Relation Locator) 하드웨어 설계 기법에 관하여 기술한다[14].(표준 0.13μm 공법, 1GHz 실리콘 VLSI) 이는 이전에 소개된 디자인(opto-electronic TWINKLE과 mesh-based sieving)에 비하여 3~4배 비용 면에서 효율적이다.(실제 구현물은 없음), $10M 비용으로 TWIRL을 설계한 경우 RSA 512 비트 키를 완전히 인수분해 하는데 10분 미만의 시간이 소요될 것으로 사료된다. 또한 1024 비트 키에 대하여 파라미터 분석 결과를



〔그림 3〕 Largish Prime 영역의 TWIRL 구조

적용한 경우 $10M 비용의 장비로 1년 이내에 시빙이 가능할 것이라 판단된다.

2.1 TWIRL의 구성

NFS의 PC 기반 설계는 실질적으로 512bit, 530 bit가 한계이며 이보다 더 확장하는 것은 불가능하다. 특히 1024 비트 합성수에 대한 비용을 고려할 때 이는 〔표 4〕와 같다.

〔표 3〕 기존의 PC 기반 장비의 1024bit 합성수 인수분해 비용

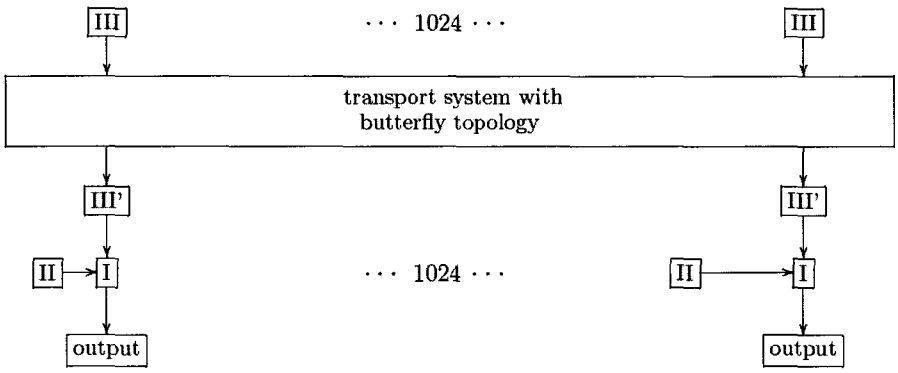
사용자원	10GB 메모리의 PC $5x10^7$대
비 용	$ 10^{11}
소요시간	1년

TWINKLE은 한번에 하나씩 시빙 영역을 조사하는 반면에, TWIRL은 매 클럭마다 수천 개의 시빙 영역을 다루며 더 작고 더 쉽게 구성된다. 512 합성수의 경우 30cm 하나의 실리콘 웨이퍼(Silicon Wafer)에 79개의 독립 시빙 장비가 위치한다. 반면에 TWINKLE은 GaAs 웨이퍼 전부를 필요로 한다.

그림 3에서 수열의 p_i 값은 s에 비하여 매우 큰 값이므로 $\lfloor \log p_i \rfloor$ 를 매우 드물게 방출한다. Largish 소수 $(p_i > 5.2x10^5)_R$, $(p_i > 4.2x10^6)_A$은 많은 수열을 다루지만 각각의 수열은 저장 공간이 콤팩트하므로 비싼 로직 회로를 이용하는 것이 유용하다. 각각의 수열은 3개의 구성을 가지는 수열(트리플릿 수열)로 표현되며 콤팩트 DRAM을 사용한 메모리 뱅크에 저장된다. 트리플릿 수열은 특수 목적의 프로세서에 의하여 주기적으로 점검 및 갱신된다. 방출 트리플릿 수열은 버퍼(buffer)를 통과하면서 몇몇 프로세서의 출력값과 병합되며, 시간의 미세 조정과 전송쌍(delivery pair)을 생성한다. 전송쌍은 파이프라인된 전송라인을 지나가며, 전송 셀의 체인으로 구성된 적당한 버스라인을 통과하며 $\lfloor \log p_i \rfloor$ 를 더한다.

〔표 4〕 TWIRL의 사용자원 및 소요시간 비교

대 상	768 Bit		1024 Bit	
파라미터	Rational Factor Base	1×10^8	Rational Factor Base	3.5×10^9
	Algebraic Factor Base	1×10^9	Algebraic Factor Base	2.6×10^{10}
	Sieve Line Width	3.4×10^{13}	Sieve Line Width	1.1×10^{15}
사용자원 (1cluster)	$(1{,}330mm^2 \times 4)_R + (4{,}430mm^2 \times 1)_A$ { $(4)_R + (1)_A$ = 1/6 wafer }		$(15{,}960mm^2 \times 4)_R + (65{,}900mm^2 \times 1)_A$ { $(1/4$ wafer $\times 8)_R + (1$ wafer$)_A$ }	
Frequence	1 GHz		1 GHz	
비 용 (1cluster)	$ 830 (NRE 제외)		$ 15,000 (NRE 제외)	
소요시간 (1cluster)	1.6 년		194 년	
1년 이내 인수분해 비용	$ 3,400 (NRE 제외)		$ 2.9M (NRE 제외)	



〔그림 4〕 SHARK 시빙 장비의 구조

3. SHARK

본 소절에서는 1년 이내에 1024Bit 합성수를 인수분해하는데 필요한 시빙을 할 수 있는 특수 목적 하드웨어 SHARK에 대하여 기술한다[5].

SHARK는 2,300개의 독립 장비로 구성되며 하나의 비용은 $70,000 정도 소요된다. 시빙 방법은 lattice 시빙을 사용하며 실제 시빙은 매우 빠른 캐시 메모리를 이용한다. 그러나 실제 캐시 메모리는 매우 비싸므로 32 MB 정도만 시빙 영역에서 사용한다. 시빙 영역은 많은 부분 영역으로 분할되며 각각의 영역에 캐시 메모리가 위치한다. 시빙 단계의 출력 값은 ECM(Elliptic Curve Method) 하드웨어 장비를 통하여 빠르게 합성수 여부를 판단한다. 이때 ECM 장비는 비교적 적은 비용으로 구성이 가능하다.

3.1 SHARK의 구성

SHARK는 I, II, III 세 가지의 부분과 전송 시스템으로 구성되며 시빙 영역은 2^{14}개의 lattice 점으로 구성된 작은 부분으로 분할된다. 시빙 진행을 위하여 각각의 점마다 1 바이트의 $\lfloor \log p_i \rfloor$ 을 더하는 것으로 충분하므로 하나의 시빙 부분은 16kB의 캐시메모리로 구성된다. SHARK는 small, medium, large 소수의 영역으로 인수 기저를 분할하여 각각을 다르게 구성한다(I, II, III).

III 영역은 큰 소수를 다루며 시빙 진행을 위해 필요한 데이터를 butterfly 전송시스템으로 전달한다. 이때 1,024개의 작은 유닛이 병렬로 동작하며 각각은 1/1,024의 시빙 영역을 다룬다. 따라서 전송시스템은 1,024개의 입력을 받는다.

II 영역은 중간 크기의 소수를 다루며 lattice들은 비교적 조밀한 소수의 분포에 대응한다. 따라서 모두 전송하지 않으며 부분적으로 정렬된다. 〔그림 4〕에서 1024개의 작은 부분으로 구성되며 각각은 시빙 영역의 일부를 다룬다. 그러나 이들 각각은 서로 통신하지 않는다.

I 영역은 작은 크기의 소수를 다루며 II와 마찬가지로 1,024개의 유닛 각각은 서로 통신하지 않는다. 인

수 기저에서 작은 소수는 분포가 매우 조밀하며 2^{14}개의 lattice 점을 가진다. 또한 I은 II와 III으로 부터의 출력 값을 수집하는 역할을 하며 이때 I에서 출력되는 값은 잠재적으로 시빙이 수행된 결과 값이다. 이때 출력 값은 ECM 장비를 통하여 smoothness를 점검한다.

2,300개의 장비는 $1.7x10^{14}$개의 리포트를 출력하며 이는 ECM을 통하여 smoothness를 점검한다. 물론 이는 일반적인 PC를 통하여 구성될 수도 있다. 그러나 파라미터를 적당히 조절하고 ECM 장비를 설계하는 것이 50%정도의 비용을 절감한다.

〔표 5〕 SHARK의 1024bit 합성수 인수분해의 시빙영역 설계비용

사용자원 (1cluster)	1/4 wafer + DRAM
비 용 (1cluster)	$ 40,000 (NRE 제외)
소요시간 (1cluster)	2,300년
Frequence	1 GH
1년 이내 인수분해 비용	$ 92 M (NRE 제외)

Ⅳ. 결 론

현재 사용하고 있는 상당수의 정보보호 서비스들은 RSA를 기본 도구로 설계된 프로토콜을 통해 제공되고 있다. 따라서 RSA의 안전성을 판단하는 기준이 되는 인수분해 알고리즘에 대한 연구는 중요하다. 인수분해 알고리즘들은 크기가 매우 큰 정수를 대상으로 수행되기 때문에 알고리즘의 시간 복잡도가 높고, 특히 시빙 연산의 시간 복잡도가 가장 높다. 이를 효율적으로 구현하기 위해 TWINKLE, TWIRL, SHARK 등의 인수분해 전용 하드웨어 구조가 제안되었으며, 이를 사용하면 1024비트 합성수에 대한 인수분해에 소요되는 비용이 감소된다.

외국에서는 국제적인 협력을 통해 RSA-challenge를 인수분해 하는 작업이 꾸준히 이루어지고 있고, 인수분해를 위한 새로운 접근 방법인 시빙 전용 하드웨어의 설계 기술에 대한 연구가 활발하게 진행되고 있지만 국내에서는 인수분해에 대한 이론의 어려움과 필요성에 대한 인식부족 등으로 인하여 외국에 비해 연구가 활발하게 수행되지 않고 있다. 따라서 현재 알려져 있는 가장 효율적인 알고리즘인 NFS 의 효율성을 높이기 위한 시빙 방법과 이를 효율적으로 구현하기 위한 시빙 전용 하드웨어 설계 기술에 대한 연구는 보안 서비스를 제공하는 기반 암호의 안전성을 분석하기 위한 도구로써의 의미를 가질 뿐 아니라 기반 기술의 확보차원에서 중요한 연구이다.

참 고 문 헌

〔1〕 정창성, 김양희, *암호응용 문제를 위한 병렬 소인수분해 알고리즘에 관한 연구*, 통신정보보호학회지 제 3장 제 2절, 1993

〔2〕 D J Bernstein and A K Lenstra, *A General Number Field Sieve Implementation*, The Development of the Number Field Sieve, LNM 1554 (1993) pp 103-125.

〔3〕 D J. Bernstein, *Circuits for integer factorization: a proposal*, manuscript, 2001, http://cr.yp.to /papers.html.

〔4〕 R P Brent, *Some Integer Factorization Algorithms using Elliptic Curves*, Aus. Comp. Sci. Communications 8 (1986) pp 149-163.

〔5〕 J Franke, T Kleinjung, C Paar, J Pelzl, C Priplata, C Stahlke, *A Realizable Special Hardware Sieving Device for Factoring 1024-bit Integers*, SHARC'06, pp.189-199, 2006.

〔6〕 Willi Geiselmann, Rainer Steinwandt, *A dedicated sieving hardware*, proc. PKC 2003, LNCS 2567 254.266, Springer-Verlag, 2002.

〔7〕 R A Golliver, A K Lenstra and K S McCurley, *Lattice Sieving and Trial Division.*, Algorithmic Number Theory - ANTS 94, LNCS 877 (1994) pp 18-27.

〔8〕 RSA Laboratorie, *Information on RSA Challenge*, http://www.rsa.com/rsalabs/html/challenges.html.

〔9〕 Arjen K. Lenstra, Adi Shamir, *Analysis and Optimization of the TWINKLE Factoring Device*, proc. Eurocrypt 2002, LNCS 1807 35.52, Springer-Verlag, 2000.

〔10〕 Arjen K. Lenstra, Adi Shamir, Jim Tomlinson, Eran Tromer, *Analysis of*

Bernstein's factorization circuit, proc. Asiacrypt 2002, LNCS 2501 1.26, Springer- Verlag, 2002.

[11] C Pomerance, *The Quadratic Sieve Factoring Algorithm.*, Advances in Cryptology - EUROCRYPT 84, LNCS 209 (1985) pp 169-182.

[12] R L Rivest, A Shamir and L Adleman, A Method for Obtaining Digital Signatures and Public-Key Cryptosystems.}, Communications of the ACM, 21(2) (1978) pp 120-126.

[13] A Shamir, *Factoring Large Numbers with the TWINKLE Device.*, Extended Abstract (1999).

[14] Adi Shamir and Eran Tromer. *Factoring Large Numbers with the TWIRL Device*. Draft, February 9, 2003. Available via http://www.wisdom.weizmann.ac.il/~tromer.

[15] A Shamir, E Tromer, *Special-Purpose Hardware for Factoring: the NFS Sieving Step*, Invited talk at SHARCS'05, 2005.

[16] R D Silverman, *The multiple polynomial quadratic sieve.*, Math.Comp., Vol. 84 (1987) pp. 327-339.

〈著 者 紹 介〉

이 상 진 (Sangjin Lee)

1987년 2월 : 고려대 수학과 학사
1989년 2월 : 고려대 수학과 석사
1994년 2월 : 고려대 수학과 박사
1989년 2월~1999년 2월 : 한국전자통신연구원 선임연구원
1999년 2월~2001년 8월 : 고려대학교 자연과학대학 조교수
2001년 9월~현재 : 고려대학교 정보보호대학원 교수
관심분야 : 대칭키 암호의 분석 및 설계, 정보은닉이론, 컴퓨터 포렌식

김 창 한 (Chang Han Kim)

1985년 2월 고려대 수학과 학사.
1987년 2월 고려대 수학과 석사.
1992년 2월 고려대 수학과 박사.
1992년 3월~현재 : 세명대학교 정보보호학과 교수
관심분야 : 정수론, 공개키 암호, 암호 프로토콜

장 남 수 (Nam Su Chang)

2002년 2월 : 서울시립대학교 수학과 학사
2004년 8월 : 고려대학교 정보보호대학원 석사
2005년 3월~현재 : 고려대학교 정보보호대학원 박사과정
관심분야 : 공개키 암호, 암호칩 설계 기술, 부채널 공격 방법론

윤 택 영 (Taek-Young Youn)

2003년 2월 : 고려대학교 수학과 학사
2005년 2월 : 고려대학교 정보보호대학원 석사
2005년 3월~현재 : 고려대학교 정보보호대학원 박사과정
관심분야 : 공개키 암호 설계 및 분석, 부채널 공격 방법론, 정보보호 프로토콜