

	(19) 대한민국특허청(KR) (12) 공개특허공보(A)	(11) 공개번호 10-2013-0024487 (43) 공개일자 2013년03월08일
(51) 국제특허분류(Int. Cl.) H04L 9/28 (2006.01) G06F 7/52 (2006.01) (21) 출원번호 10-2011-0087960 (22) 출원일자 2011년08월31일 심사청구일자 2011년08월31일	(71) 출원인 고려대학교 산학협력단 서울 성북구 안암동5가 1 (72) 발명자 홍석희 서울특별시 도봉구 마들로 859-19, 한신아파트 103동 107호 (도봉동) 김창한 서울특별시 강남구 삼성로 212, 21동 507호 (대치동, 은마아파트) (뒷면에 계속) (74) 대리인 특허법인충현	

전체 청구항 수 : 총 9 항

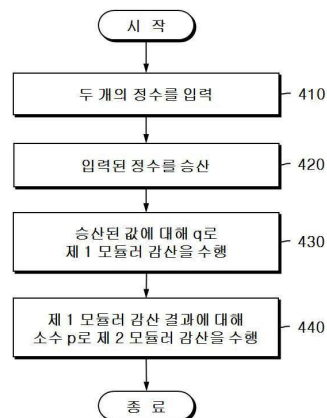
(54) 발명의 명칭 확장된 NIST 소수를 이용한 모듈러 곱셈 및 모듈러 지수승 방법

(57) 요약

본 발명은 확장된 NIST 소수를 이용한 모듈러 곱셈, 모듈러 지수승 방법 및 그 방법을 기록한 기록매체에 관한 것으로, 본 발명에 따른 모듈러 곱셈 방법은, 두 개의 정수를 입력받아 입력된 정수를 승산한 후, 승산된 값에 대해 q로 제 1 모듈러 감산을 수행하고, 제 1 모듈러 감산 결과에 대해 소수 p로 제 2 모듈러 감산을 수행하며,

q와 p는 양의 홀수 h에 대하여 $q=p \cdot h$ 를 만족한다.

대표도 - 도4



(72) 발명자

장남수

서울특별시 마포구 월드컵북로27길 16, 2층 (성산동)

조영인

서울특별시 강동구 올림픽로77길 46, 301호 (천호동)

이 발명을 지원한 국가연구개발사업

과제고유번호 ITAC1090111100020001000100100

부처명 정보통신산업진흥원

연구사업명 대학IT연구센터육성지원사업

연구과제명 스마트 그리드 보안기술 개발

주관기관 고려대학교 산학협력단

연구기간 2011.01.01 ~ 2011.12.31

특허청구의 범위

청구항 1

적어도 하나의 프로세서(processor)를 이용한 모듈러 곱셈 방법에 있어서,

두 개의 정수를 입력받는 단계;

상기 입력된 정수를 승산하는 단계;

상기 승산된 값에 대해 q 로 제 1 모듈러(modular) 감산을 수행하는 단계; 및

상기 제 1 모듈러 감산 결과에 대해 소수 p 로 제 2 모듈러 감산을 수행하는 단계를 포함하고,

상기 q 와 p 는 양의 홀수 h 에 대하여 $q=p \cdot h$ 를 만족하는 것을 특징으로 하는 방법.

청구항 2

제 1 항에 있어서,

상기 소수 p 는 소수 인수를 갖는 NIST 소수 형태의 양수인 것을 특징으로 하는 방법.

청구항 3

제 1 항에 있어서,

상기 두 개의 정수는 각각 $A=(a_{n-1}, \dots, a_0)_b$ 및 $B=(b_{n-1}, \dots, b_0)_b$ ($b=2^{32}$)이고,

상기 제 1 모듈러 감산은 $U=A \cdot B \bmod q$ 이며,

상기 제 2 모듈러 감산은 $S=U \bmod p$ 인 것을 특징으로 하는 방법.

청구항 4

제 1 항에 있어서,

상기 제 1 모듈러 감산은,

상기 승산된 값에 대한 모듈러 q 의 연산을 소정 개수의 비트 정수의 덧셈으로 변환하여 연산하는 것을 특징으로 하는 방법.

청구항 5

제 4 항에 있어서,

상기 제 1 모듈러 감산은,

$q=2^{288}+2^{96}-1$ ($0 \leq c < p_{285}^2$) 및 기저 2^{32} 인 $c=(c_{17}, \dots, c_0)$ 를 입력받고,

288비트의 정수 $s_1=(c_8, c_7, c_6, c_5, c_4, c_3, c_2, c_1, c_0),$

$s_2=(-c_{14}, -c_{14}, -c_{12}, -c_{11}, -c_{10}, -c_9, 0, 0, 0),$

$s_3=(0, 0, 0, c_{14}, c_{13}, c_{12}, c_{11}, c_{10}, c_9),$

$s_4 = (c_{17}, c_{16}, c_{15}, c_{17}, c_{16}, c_{15}, -c_{17}, -c_{16}, -c_{15})$ 를 정의하며,

상기 정의된 정수에 대한 $(s_1 + s_2 + s_3 + s_4 \bmod q)$ 를 수행하여 반환하는 것을 특징으로 하는 방법.

청구항 6

제 1 항에 있어서,

상기 제 2 모듈러 감산은,

상기 제 1 모듈러 감산 결과와 상기 소수 p를 입력받고,

상기 제 1 모듈러 감산 결과가 상기 소수 p보다 큰 만큼 상기 소수 p를 왼쪽 시프트(shift)하고,

상기 시프트 결과를 1 만큼 오른쪽 시프트하고,

상기 제 1 모듈러 감산 결과에서 상기 시프트 결과를 감산하고,

만약 상기 감산 결과가 상기 소수 p보다 크다면 상기 과정을 반복하며,

이상의 연산 결과 산출된 감산 결과를 반환하는 것을 특징으로 하는 방법.

청구항 7

적어도 하나의 프로세서(processor)를 이용한 모듈러 지수승 방법에 있어서,

$g = (g_n g_{n-1} \cdots g_1 g_0)_b$ 및 양수 $e = (e_n e_{n-1} \cdots e_1 e_0)_2$ 를 입력받는 단계;

A에 대해 2의 지수승하고, q로 모듈러 감산을 수행하는 단계;

만약 $e_i = 1$ 이면, A와 g를 승산하고, q로 모듈러 감산을 수행하는 단계;

상기 모듈러 감산을 t회 만큼 반복 수행하는 단계; 및

상기 수행 결과에 대해 p로 모듈러 감산을 수행하여 그 결과를 반환하는 단계를 포함하고,

상기 q와 p는 양의 홀수 h에 대하여 $q = p \cdot h$ 를 만족하고,

상기 p는 소수 인수를 갖는 NIST 소수 형태의 양수인 것을 특징으로 하는 방법.

청구항 8

제 7 항에 있어서,

상기 q는 상기 p와 동일한 수의 워드 블록을 사용하며,

상기 h는 상기 워드 사이즈의 절반 이하를 사용하는 것을 특징으로 하는 방법.

청구항 9

제 1 항 내지 제 8 항 중에 어느 한 항의 방법을 컴퓨터에서 실행시키기 위한 프로그램을 기록한 컴퓨터로 읽을 수 있는 기록매체.

명세서

기술분야

본 발명은 모듈러 연산에 관한 것으로, 특히 NIST 소수에 기초하여 RSA, 타원 곡선 암호 시스템, 페어링 기반

[0001]

암호 시스템 등의 스마트 그리드(smart grid) 보안 분야에서 활용될 수 있는 모듈러 곱셈, 모듈러 지수승 방법 및 이를 기록한 기록매체에 관한 것이다.

배경 기술

[0002] 모듈러 연산의 효율적인 구현은 RSA, 타원 곡선 암호 시스템, 페어링 기반 암호시스템 등의 고속화를 위해 매우 중요하므로 많은 연구가 진행되어왔다. 그 중 모듈러 지수승 연산에 가장 빠른 방법은 이하에서 제시되는 비특허문헌1에 따른 몽고메리 지수승 알고리즘(Montgomery Exponentiation Algorithm)이며 몽고메리 지수승 알고리즘은 특수한 표현 방법을 이용하여 나눗셈 연산을 비트 시프트 연산으로 대체한다. 이 때, 시프트 연산은 나눗셈 연산에 비해 훨씬 가벼우므로 몽고메리 지수승 알고리즘은 일반적인 지수승 알고리즘에 비하여 빠르게 지수승 연산을 수행할 수 있다.

[0003] 또한, 모듈러스 p 가 특수한 형태일 경우에도 모듈러 연산의 효율성을 높일 수 있다. 비특허문헌2와 같이 $2^n - 1$ 의 형태를 가지는 메르센 수(Mersenne numbers)가 잘 알려진 예이다. 그러나 암호학적으로 유용한 메르센 수는 많지 않다. 이에 J. Solinas는 보다 일반적인 메르센 수를 제안하였다. Solinas 수는 $p=f(t)$ 의 형태로 나타나며 t 는 2의 지수승 형태이다. 이러한 표현법을 이용하면 덧셈 연산만으로 모듈러 감산 연산을 수행할 수 있다.

[0004] NIST 소수는 Solinas 수의 특수한 형태로서, t 의 지수가 워드 사이즈(Wordsize)의 배수인 소수들이다. 표준 문서 FIPS 186-2에서는 NIST 소수(NIST Primes)를 이용하여 소수체 기반의 타원 곡선 암호 알고리즘을 구현하도록 권장한다. NIST 소수는 모듈러 감산 알고리즘에 특화된 형태이기 때문에 일반 소수에 비하여 빠른 모듈러 연산을 수행할 수 있다. 그러나 그 수가 많지 않아 타원 곡선 및 페어링 암호 알고리즘에 적용하기에 충분한 소수체를 제공하고 있지 않다.

선행기술문헌

비특허문헌

[0005] (비특허문헌 0001) P. L. Montgomery, "Modular multiplication without trial division", Math. Comp, 44, 1985.

(비특허문헌 0002) R. Crandall, "Method and apparatus for public key exchange in a cryptographic system", U. S. Patent number 5159632, 1985.

발명의 내용

해결하려는 과제

[0006] 본 발명이 해결하고자 하는 기술적 과제는 종래의 NIST 소수의 수가 많지 않다는 문제로 인해 타원 곡선 및 페어링 암호 알고리즘에 적용하기에 충분한 소수체를 제공하지 못하는 제약을 해소하고, 몽고메리 곱셈 및 지수승 알고리즘 이상의 성능과 효율을 갖는 모듈러 연산에 제시되고 있지 못하다는 암호화 분야의 한계를 극복하고자 한다.

과제의 해결 수단

[0007] 상기 기술적 과제를 해결하기 위하여, 본 발명의 일 실시예에 따른 적어도 하나의 프로세서(processor)를 이용한 모듈러 곱셈 방법은, 두 개의 정수를 입력받는 단계; 상기 입력된 정수를 승산하는 단계; 상기 승산된 값에 대해 q 로 제 1 모듈러(modular) 감산을 수행하는 단계; 및 상기 제 1 모듈러 감산 결과에 대해 소수 p 로 제 2 모듈러 감산을 수행하는 단계를 포함하고, 상기 q 와 p 는 양의 홀수 h 에 대하여 $q=p \cdot h$ 를 만족한다.

[0008] 상기된 모듈러 곱셈 방법에서 상기 소수 p 는 소수 인수를 갖는 NIST 소수 형태의 양수이다.

[0009] 또한, 상기된 모듈러 곱셈 방법에서 상기 제 1 모듈러 감산은, 상기 승산된 값에 대한 모듈러 q 의 연산을 소정

개수의 비트 정수의 덧셈으로 변환하여 연산하는 것이 바람직하다. 나아가 상기된 모듈러 곱셈 방법에서 상기 제 2 모듈러 감산은, 상기 제 1 모듈러 감산 결과와 상기 소수 p를 입력받고, 상기 제 1 모듈러 감산 결과가 상기 소수 p보다 큰 만큼 상기 소수 p를 왼쪽 시프트(shift)하고, 상기 시프트 결과를 1 만큼 오른쪽 시프트하고, 상기 제 1 모듈러 감산 결과에서 상기 시프트 결과를 감산하고, 만약 상기 감산 결과가 상기 소수 p보다 크다면 상기 과정을 반복하며, 이상의 연산 결과 산출된 감산 결과를 반환하는 것이 바람직하다.

[0010] 상기 기술적 과제를 해결하기 위하여, 본 발명의 다른 실시예에 따른 적어도 하나의 프로세서(processor)를 이용한 모듈러 지수승 방법은, $g=(g_n g_{n-1} \cdots g_1 g_0)_b$ 및 양수 $e=(e_n e_{n-1} \cdots e_1 e_0)_2$ 를 입

력받는 단계; A에 대해 2의 지수승하고, q로 모듈러 감산을 수행하는 단계; 만약 $e_i=1$ 이면, A와 g를 승산하고, q로 모듈러 감산을 수행하는 단계; 상기 모듈러 감산을 t회 만큼 반복 수행하는 단계; 및 상기 수행 결과에 대해 p로 모듈러 감산을 수행하여 그 결과를 반환하는 단계를 포함하고, 상기 q와 p는 양의 홀수 h에 대하여 $q=p \cdot h$ 를 만족하고, 상기 p는 소수 인수를 갖는 NIST 소수 형태의 양수이다.

[0011] 또한, 상기된 모듈러 지수승 방법에서, 상기 q는 상기 p와 동일한 수의 워드 블록을 사용하며, 상기 h는 상기 워드 사이즈의 절반 이하를 사용하는 것이 바람직하다.

[0012] 한편, 이하에서는 상기 기재된 모듈러 곱셈 방법과 모듈러 지수승 방법을 컴퓨터에서 실행시키기 위한 프로그램을 기록한 컴퓨터로 읽을 수 있는 기록매체를 제공한다.

발명의 효과

[0013] 본 발명의 다양한 실시예들은 확장된 NIST 소수를 이용한 모듈러 곱셈, 모듈러 지수승을 제안함으로써, 타원 곡선 및 페어링 암호 알고리즘에서 활용할 수 있는 충분한 수의 소수체를 제공할 수 있으며, 몽고메리 곱셈 및 지수승 알고리즘보다 효율적인 모듈러 연산을 수행할 수 있으므로, 소수체 기반의 타원 곡선 암호 시스템의 페어링 기반 암호 시스템의 성능을 향상시킬 수 있다.

도면의 간단한 설명

[0014] 도 1은 본 발명의 실시예들이 채택하고 있는 모듈러스 q를 이용한 모듈러 감산을 설명하기 위한 예시도이다.
 도 2는 본 발명의 실시예들이 채택하고 있는 모듈러 q 감산 알고리즘을 도시한 도면이다.
 도 3은 본 발명의 실시예들이 채택하고 있는 모듈러 p 감산 알고리즘을 도시한 도면이다.
 도 4는 본 발명의 일 실시예에 따른 적어도 하나의 프로세서를 이용한 모듈러 곱셈 방법을 도시한 흐름도이다.
 도 5는 도 4에 따른 본 발명의 실시예들이 채택하고 있는 확장된 NIST 소수를 이용한 모듈러 곱셈 알고리즘을 도시한 도면이다.
 도 6은 본 발명의 다른 실시예에 따른 적어도 하나의 프로세서를 이용한 모듈러 지수승 알고리즘을 도시한 도면이다.
 도 7은 본 발명의 실시예들이 채택하고 있는 확장된 NIST 소수를 이용한 모듈러 곱셈 및 모듈러 지수승 알고리즘의 실행 시간을 종래의 연산 방법과 비교하여 도시한 도면이다.
 도 8은 본 발명의 실시예들이 채택하고 있는 확장된 NIST 소수를 이용한 모듈러 연산의 평균 실행 시간을 종래의 연산 방법과 비교하여 도시한 도면이다.
 도 9는 본 발명의 실시예들이 채택하고 있는 확장된 NIST 소수와 종래의 NIST 소수의 개수를 비교하여 도시한 도면이다.

발명을 실시하기 위한 구체적인 내용

[0015] 본 발명의 실시예들을 설명하기에 앞서, 우선 본 발명의 실시예들이 구현되는 암호화 기술의 특징에 대해 소개하고, 이러한 실시예들이 구현되는 환경의 특징으로부터 안출될 수 있는 본 발명의 기본적인 아이디어를 제시하고자 한다.

[0016] 대용량 암호 키를 사용하는 정보 보호 기술 분야에서는 모듈러 곱셈을 고속화하기 위한 연구들이 지속적으로 이루어져 왔다. 특히, 암호화된 정보에 대한 해독 기술이 발달함에 따라 이에 대응하여 많은 암호화 기술들이 암호 알고리즘의 빈도를 강화하기 위해 모듈러 곱셈을 반복적으로 사용하는 연산 구조를 채택하고 있다. 따라서, 암호 알고리즘의 처리 속도를 향상시키기 위해서는 모듈러 곱셈 연산을 고속화시키는 것이 무엇보다 중요하다.

[0017] 이러한 모듈러 곱셈 연산과 관련하여, FIPS 186-2 표준은 다음의 수학적 식 1과 같은 NIST 소수를 이용하여 타원 곡선 암호 시스템을 구현하도록 권장한다.

수학적 식 1

$$\begin{aligned} p_{192} &= 2^{192} - 2^{64} - 1, \\ p_{224} &= 2^{224} - 2^{96} + 1, \\ p_{256} &= 2^{256} - 2^{224} + 2^{192} + 2^{96} - 1, \\ p_{384} &= 2^{384} - 2^{128} - 2^{96} + 2^{32} - 1, \\ p_{521} &= 2^{521} - 1 \end{aligned}$$

[0018]

[0019] p_{521} 을 제외한 NIST 소수는 모두 지수가 32의 배수인 항들의 합으로 이루어져 있다. 이러한 형태를 이용하여 32비트 워드사이즈를 사용하는 컴퓨터에서 효율적인 모듈러 감산 연산이 수행될 수 있다.

[0020] 앞서 언급한 바와 같이 이러한 NIST 소수는 Solinas 수의 특수한 형태로서, t 의 지수가 워드 사이즈(Wordsize)의 배수인 소수들이다. 특히, NIST 소수는 모듈러 감산 알고리즘에 특화된 형태이기 때문에 일반 소수에 비하여 빠른 모듈러 연산이 가능하다는 장점을 갖는다. 그러나, NIST 소수는 그 수가 많지 않아 타원 곡선 및 페어링 암호 알고리즘에 적용하기에 충분한 소수체를 제공하지 못한다는 점이 약점으로 지적되었다.

[0021] 따라서, 이하에서 제시될 본 발명의 실시예들은 이러한 NIST 소수의 형태에 착안한 확장된 NIST 소수를 제안하고자 한다. 이러한 확장된 NIST 소수를 이용하여 모듈러 연산을 수행함으로써 종래의 몽고메리 곱셈이나 몽고메리 지수승 연산에 비해 향상된 성능의 암호화 연산이 가능하다.

[0022] 이제, 확장된 NIST 소수를 제안하고, 이의 연산 효율성에 대하여 논한다. 먼저 확장된 NIST 소수를 정의하기 위

해 소수 p 를 이루는 2의 지수승 항의 개수를 $wt(p)$ 라 한다. 즉,

$p = 2^{\alpha} + 2^{\beta} + 1 (\alpha \neq 0, \beta \neq 0, \alpha \neq \beta)$ 일 때, $wt(p)$ 는 3이다. 확장된 NIST 소수는 다음과 같이 정의된다.

[0023] [정의] Solinas 수 $q = p \cdot h$ 를 만족하는 양의 홀수 h 가 존재할 때, 소수 p 를 확장된 NIST 소수(Extended NIST Prime : ENP)라 한다.

[0024] 즉, 이상의 정의와 같이 소수 인수를 갖는 NIST 소수 형태의 양수를 찾아 이 소수 인수를 확장된 NIST 소수(ENP)로 사용한다. 이하에서는, 이상과 같이 정의된 확장된 NIST 소수를 이용한 모듈러 연산 방법에 대해 설명한다.

[0025] (1) 모듈러 감산

[0026] 본 발명의 실시예들은 정수 c 에 대해 통상적인 NIST 소수와 같은 방법으로 $c \bmod q$ 를 효율적으로 연산한

후, 그 결과 값을 $\text{mod } p$ 에 대하여 연산한다.

[0027]

설명의 편의를 위해, 285 비트 확장된 NIST 소수(ENP) $p=p_{285}$ 를 $q=2^{288}+2^{96}-1$, $h=11$ 인

예를 사용하여 모듈러 감산 연산을 수행한다. $0 \leq c < p_{285}^2$ 인 정수인 c 를 선택하면 다음의 수학적 2와 같이 나타낼 수 있다.

수학적 2

$$\begin{aligned} 2^{288} &\equiv -2^{96} + 1 \pmod{q}, \\ 2^{320} &\equiv -2^{128} + 2^{32} \pmod{q}, \\ 2^{352} &\equiv -2^{160} + 2^{64} \pmod{q}, \\ 2^{384} &\equiv -2^{192} + 2^{96} \pmod{q}, \\ 2^{416} &\equiv -2^{224} + 2^{128} \pmod{q}, \\ 2^{448} &\equiv -2^{256} + 2^{160} \pmod{q}, \\ 2^{480} &\equiv 2^{192} + 2^{96} - 1 \pmod{q}, \\ 2^{512} &\equiv 2^{224} + 2^{128} - 2^{32} \pmod{q}, \\ 2^{544} &\equiv 2^{256} + 2^{160} - 2^{64} \pmod{q} \end{aligned}$$

[0028]

[0029]

이상의 수학적 2의 표현 방법을 이용하면, $c \text{ mod } q$ 연산은 수학적 3와 같이 10개의 288 비트 정수의 덧셈으로 연산 가능하다.

수학적 3

$$\begin{aligned} &c_{17}2^{256} + c_{17}2^{160} - c_{17}2^{64}, c_{16}2^{224} + c_{16}2^{128} - c_{16}2^{32}, \\ &c_{15}2^{192} + c_{15}2^{96} - c_{15}, -c_{14}2^{256} + c_{14}2^{160}, -c_{13}2^{224} + c_{13}2^{128}, \\ &-c_{12}2^{192} + c_{12}2^{96}, -c_{11}2^{160} + c_{11}2^{64}, -c_{10}2^{128} + c_{10}2^{32}, -c_92^{96} + c_9, \\ &c_82^{256} + c_72^{224} + c_62^{192} + c_52^{160} + c_42^{128} + c_32^{96} + c_22^{64} + c_12^{32} + c_0 \end{aligned}$$

[0030]

[0031]

도 1은 본 발명의 실시예들이 채택하고 있는 모듈러스 q 를 이용한 모듈러 감산을 설명하기 위한 예시도로서, 이상의 모듈러스 q 감산의 10 개의 288 비트 정수가 어떻게 도출되었는지를 도시하고 있다.

[0032]

도 2는 본 발명의 실시예들이 채택하고 있는 모듈러 q 감산 알고리즘을 도시한 도면으로서, 도 2에 도시된 알고

리즘 1은 모듈러스 ENP p_{285} 에 대한 $\text{mod } q$ 연산 알고리즘이다.

- [0033] 도 2를 참조하면, 모듈러 q 감산 알고리즘은 그 입력값으로서 $q=2^{288}+2^{96}-1$, $0 \leq c < p_{285}^2$ 이고, 기저 2^{32} 인 $c=(c_{17}, \dots, c_0)$ 를 입력받는다.
- [0034] 입력 값에 기초하여 우선, 288비트의 정수를 다음과 같이 정의한다.
- $$s_1=(c_8, c_7, c_6, c_5, c_4, c_3, c_2, c_1, c_0),$$
- $$s_2=(-c_{14}, -c_{14}, -c_{12}, -c_{11}, -c_{10}, -c_9, 0, 0, 0),$$
- $$s_3=(0, 0, 0, c_{14}, c_{13}, c_{12}, c_{11}, c_{10}, c_9),$$
- $$s_4=(c_{17}, c_{16}, c_{15}, c_{17}, c_{16}, c_{15}, -c_{17}, -c_{16}, -c_{15})$$
- [0035] 그런 다음, 정의된 정수에 대한 $(s_1+s_2+s_3+s_4 \bmod q)$ 를 수행하여 결과로서 반환한다. 이상과 같은 모듈러 q 감산을 통해 $c \bmod q$ 의 결과값을 얻을 수 있다.
- [0036] 이제, $c \bmod q$ 의 결과값으로부터 확장된 NIST 소수 p 에 의한 모듈러 감산을 수행한다.
- [0037] 도 3은 본 발명의 실시예들이 채택하고 있는 모듈러 p 감산 알고리즘을 도시한 도면으로서, $c \bmod q$ 의 결과를 $\bmod p$ 에 대하여 연산하는 알고리즘 2를 예시하고 있다. 특히, 도 3은 $c \bmod q$ 의 결과값과 모듈러스 ENP p_{285} 를 입력받아 $c \bmod p_{285}$ 를 결과값으로 산출한다.
- [0038] 알고리즘 2는 입력값에 대해 앞서 알고리즘 1에 의한 모듈러 감산 결과($c \bmod q$ 의 연산 결과를 의미한다.)가 확장된 NIST 소수 p (p_{285} 를 의미한다.)보다 큰지 여부를 검사한 후, 해당 조건을 만족하는 동안 왼쪽 시프트(shift)를 수행한다. 즉, 알고리즘 1에 의한 모듈러 감산 결과가 확장된 소수 p 보다 큰 만큼 소수 p 를 왼쪽 시프트한다. 이어서, 이상의 시프트 결과를 1 만큼 오른쪽 시프트한다.
- [0039] 다음으로, 알고리즘 2는 알고리즘 1에 의한 모듈러 감산 결과에서 이상의 시프트된 결과를 감산하여 감산 결과를 저장한다. 이 때, 감산 결과가 확장된 NIST 소수 p 보다 크다면 이상의 알고리즘 2의 전체 과정을 반복하게 된다.
- [0040] 마지막으로, 이상의 연산 결과 산출된 감산 결과를 결과값으로 반환한다.
- [0041] (2) 모듈러 곱셈
- [0042] 본 발명의 실시예들은 확장된 NIST 소수를 이용하여 모듈러 곱셈 연산을 수행함에 있어, 이상의 알고리즘 1 및 알고리즘 2에 의한 모듈러 감산을 순차적으로 진행한다.
- [0043] 도 4는 본 발명의 일 실시예에 따른 적어도 하나의 프로세서를 이용한 모듈러 곱셈 방법을 도시한 흐름도로서, 이상의 연산 과정을 간략화하여 도식화하였다. 본 실시예에 따른 모듈러 곱셈 방법에서, q 와 p 는 양의 홀수 h 에 대하여 $q=p \cdot h$ 를 만족함은 기본 전제로 한다. 즉, 확장된 NIST 소수에 대한 이상의 모듈러 곱셈 연산은 이상의 전제가 있기 때문에 가능한 것이며, 단지 2 회의 모듈러 감산을 순차적으로 수행함으로써 그 결과값을

연을 수 있다.

[0044] 410 단계에서, 본 발명의 실시예에 따른 모듈러 곱셈기는 두 개의 정수를 입력받는다.

[0045] 420 단계에서, 모듈러 곱셈기는 410 단계를 통해 입력된 정수를 승산한다.

[0046] 430 단계에서, 모듈러 곱셈기는 420 단계를 통해 승산된 값에 대해 q 로 제 1 모듈러(modular) 감산을 수행한다. 이러한 제 1 모듈러 감산은 앞서 도 2를 통해 설명한 바 있는 알고리즘 1에 대응한다.

[0047] 440 단계에서, 모듈러 곱셈기는 430 단계를 통해 산출된 제 1 모듈러 감산 결과에 대해 소수 p 로 제 2 모듈러 감산을 수행한다. 이러한 제 2 모듈러 감산은 앞서 도 3을 통해 설명한 바 있는 알고리즘 2에 대응한다.

[0048] 본 실시예에서 모듈러 곱셈기는 이상에서 기술될 일련의 연산을 수행하기 위한 적어도 하나의 프로세서를 구비한다. 이러한 프로세서는 입력값에 대해 승산과 모듈러 감산을 연속적으로 수행함으로써 최종적으로 모듈러 곱셈의 결과값을 반환한다. 구현의 관점에서 이러한 모듈러 곱셈기는 상기된 프로세서가 일련의 연산을 수행하는 과정에서 연산 그 자체 또는 임시 저장을 위한 공간으로서 사용되는 저장 공간(memory) 내지 레지스터(register), 그리고 승산 및 제산을 간편히 수행할 수 있는 시프터(shifter)를 더 포함할 수 있다. 물론, 이들 연산들을 상기된 프로세서와 메모리를 통해 수행하기 위해 부가적인 소프트웨어 코드(software code)가 활용될 수 있음은 당연하다. 본 발명이 속하는 기술 분야에서 통상의 지식을 가진 자는 이상의 실시예가 구현될 수 있는 하드웨어 구성을 자유롭게 구성할 수 있으므로, 여기서는 새로운 모듈러 곱셈의 방법론과 연산의 순차적인 구성에 집중하여 소개하도록 한다.

[0049] 도 5는 도 4에 따른 본 발명의 실시예들이 채택하고 있는 확장된 NIST 소수를 이용한 모듈러 곱셈 알고리즘을 도시한 도면으로, 모듈러스 ENP에 대한 정수 A 와 B 의 곱은 알고리즘 3과 같이 세 단계에 걸쳐 수행된다.

[0050] 알고리즘 3은 우선 입력값으로 두 개의 정수 $A=(a_{n-1}, \dots, a_0)_b$ 및 $B=(b_{n-1}, \dots, b_0)_b$

(이 때, $b=2^{32}$ 이다.)를 입력받는다.

[0051] 1 단계에서, 모듈러 곱셈기는 입력된 두 개의 정수에 대한 곱을 연산한다.

[0052] 2 단계에서, 모듈러 곱셈기는 곱 연산의 결과로부터 제 1 모듈러 감산을 수행한다. 즉, $\bmod q$ 연산을 수행한다. 이러한 과정은 앞서 소개한 알고리즘 1에 따른다.

[0053] 3 단계에서, 모듈러 곱셈기는 제 1 모듈러 감산의 결과에 대해 제 2 모듈러 감산을 수행한다. 즉, $\bmod p$ 연산을 수행한다. 이러한 과정은 앞서 소개한 알고리즘 2에 따른다.

[0054] 기존의 NIST 소수를 이용한 알고리즘과 비교할 때, 본 발명의 실시예들이 채택하고 있는 확장된 NIST 소수를 이용함에 따라 새롭게 추가된 연산은 알고리즘 3의 세 번째 단계에 해당한다. 이상과 같은 알고리즘 3에 따라 본 발명의 실시예들은 보다 많은 수의 확장된 NIST 소수를 활용할 수 있으면서도 여전히 간단한 모듈러 곱셈 방법에 따라 연산을 수행할 수 있다.

[0055] 이제, 소프트웨어로 구현한 모듈러 곱셈의 연산의 효율성을 분석하기 위해 마이크로프로세서의 워드 곱셈 연산 시간 t_m 과 나눗셈 연산 시간 t_d 를 정의한다. 큰 수의 곱셈, 제곱과 나눗셈 연산의 수행 시간은 다음의 수학적 식 4와 같다고 알려져 있다.

수학적 식 4

$$M(n)=n^2 t_m, S(n)=(n^2+n)/2 t_m, \\ D(n_1, n_2)=(n_1-n_2)(n_2+3) t_m+(n_1-n_2) t_d$$

[0056]

- [0057] 수학식 4에서 $M(n), S(n)$ 은 각각 큰 수의 곱셈과 제곱의 수행 시간이며, 피승수는 n 워드이다.
 $D(n_1, n_2)$ 는 큰 수의 나눗셈 수행 시간이며, 피제수는 n_1 워드이며 제수는 n_2 워드이다.
- [0058] (3) 모듈러 지수승
- [0059] 확장된 NIST 소수를 이용한 지수승 연산에서는 $\text{mod } p$ 연산이 전체 연산의 작은 부분을 차지하므로 매우 효율적으로 연산을 수행할 수 있다. 도 6을 참조하여 확장된 NIST 소수를 이용한 지수승 연산에 대해 설명하도록 한다.
- [0060] 도 6은 본 발명의 다른 실시예에 따른 적어도 하나의 프로세서를 이용한 모듈러 지수승 알고리즘을 도시한 도면으로서, 본 실시예에서는 left-to-right binary 지수승 알고리즘에 기반한 알고리즘 4을 이용하였다. 이미 모듈러 곱셈 과정을 통해 설명한 바와 같이, 알고리즘 4를 통해 q 와 p 는 양의 홀수 h 에 대하여 $q = p \cdot h$ 를 만족하고, p 는 소수 인수를 갖는 NIST 소수 형태의 양수인 것은 당연하다.
- [0061] 알고리즘 4는 입력값으로 $g = (g_n g_{n-1} \dots g_1 g_0)_b$ 및 양수 $e = (e_n e_{n-1} \dots e_1 e_0)_2$ 를 입력받는다.
- [0062] 우선, 알고리즘 4는 A 에 초기값을 할당하고, A 에 대해 2의 지수승(즉, $A \cdot A$ 를 연산한다.)한 후, q 로 모듈러 감산($\text{mod } q$)을 수행한다. 그런 다음, 만약 $e_i = 1$ 이면, A 와 g 를 승산하고, q 로 모듈러 감산($\text{mod } q$)을 수행한다.
- [0063] 이상의 모듈러 감산 과정을 일정 횟수(t 회) 만큼 반복 수행한 후, 수행 결과에 대해 p 로 모듈러 감산($\text{mod } p$)을 수행하여 그 결과를 반환한다.
- [0064] 본 실시예를 통해 모듈러 지수승을 수행하는 장치는 이상에서 기술될 일련의 연산을 수행하기 위한 적어도 하나의 프로세서를 구비한다. 이러한 프로세서는 입력값에 대해 승산과 q 모듈러 감산을 반복적으로 수행하고, 이에 연속하여 p 모듈러 감산을 수행함으로써 최종적으로 모듈러 지수승의 결과값을 반환한다. 구현의 관점에서 본 장치는 상기된 프로세서가 일련의 연산을 수행하는 과정에서 연산 그 자체 또는 임시 저장을 위한 공간으로서 사용되는 저장 공간(memory) 내지 레지스터(register), 그리고 승산 및 제산을 간편히 수행할 수 있는 시프터(shifter)를 더 포함할 수 있다. 물론, 이들 연산들을 상기된 프로세서와 메모리를 통해 수행하기 위해 추가적인 소프트웨어 코드(software code)가 활용될 수 있음은 당연하다.
- [0065] 한편, 이상의 알고리즘 4에서, q 가 p 보다 많은 워드 블록을 사용한다면 과정 4와 7역시 p 보다 많은 워드 블록을 차지하게 된다. 이에 따라 단계 3과 6에서 p 보다 큰 워드 블록을 갖는 수들의 곱셈 연산이 수행되어야 한다. 이 경우, p 와 같은 워드 블록을 갖는 수들의 곱셈 연산을 하는 경우에 비하여 복잡도가 증가하게 된다. 또한 h 가 너무 크면 과정 10의 모듈러 감산에 소모되는 비용이 증가한다. 따라서, 이하의 [Remark 1]에서는 효율적인 모듈러 연산을 위해 q 와 h 를 제한할 수 있음을 제안한다.
- [0066] [Remark 1] 효율적인 모듈러 연산을 위해 q 는 p 와 동일한 수의 워드 블록을 사용하며 h 는 워드 사이즈의 절반 이하를 사용하도록 제한한다.
- [0067] 이러한 제한에 의해 본 발명의 실시예들에 따른 모듈러 연산의 효율이 향상될 수 있다.
- [0068] 도 7은 본 발명의 실시예들이 채택하고 있는 확장된 NIST 소수를 이용한 모듈러 곱셈 및 모듈러 지수승 알고리즘의 실행 시간을 종래의 연산 방법과 비교하여 실시한 실험 결과를 도시한 도면이다. 도 7에서는 모듈러스

통상적인 NIST 소수를 이용한 모듈러 연산과 확장된 NIST 소수를 이용한 모듈러 연산의 실험 결과를 비교한다.

[0069]

본 실험에서 q 와 h 는 상기된 [Remark 1]을 따르고, $wt(q)=2 \sim 5$ 를 만족하는 32~2048 비트 소수 중에서 확장된 NIST 소수를 찾는 실험을 수행하였다. 모듈러 연산 실험 결과는 발견된 확장된 NIST 소수 중 가장 나

쁜 경우를 기준으로 수행하였다. 이 때의 P_{1649} 는 $h=65531$ 과 $q=2^{1664}+2^{1184}+2^{896}-2^{288}-1$ 로 이루어지며, 1649 비트의 임의의(random) 소수를 생성하여 일반적인 모듈러 연산 알고리즘과 몽고메리 알고리즘을 수행하였다.

[0070]

도 7을 통해 확인할 수 있듯이, 본 발명의 실시예들이 채택하고 있는 확장된 NIST 소수(ENP)를 이용한 모듈러 연산에 대하여, 모듈러 곱셈 알고리즘의 경우, 일반적인 모듈러 곱셈 및 몽고메리 모듈러 곱셈에 비해 향상된 연산 효율을 나타낸다. 또한, 모듈러 지수승 알고리즘의 경우에도 일반적인 모듈러 지수승 연산뿐만 아니라 몽고메리 모듈러 지수승 연산과 비교하여도 그 효율의 우수함을 확인할 수 있다.

[0071]

도 8은 본 발명의 실시예들이 채택하고 있는 확장된 NIST 소수(ENP)를 이용한 모듈러 연산의 평균 실행 시간을 종래의 연산 방법과 비교하여 도시한 도면으로서, 도 7의 조건하에서 실제로 실험을 통해 조사된 연산 결과를 제시하고 있다. 도 8의 실험을 통해, 모듈러 연산은 각각의 경우에 100만 번 실행하였으며, 각 알고리즘의 평균 실행 시간을 측정하였다. 또한, 모듈러 연산을 적용함에 있어 OpenSSL 암호 라이브러리를 사용하였으며, Intel(R) Core(TM) i7 CPU 860 @ 2.80GHz processor 에 구현하여 실행 시간을 측정하였다.

[0072]

도 8에서 확인할 수 있듯이, 확장된 NIST 소수를 사용한 모듈러 곱셈 및 모듈러 지수승 연산의 경우 양자 모두 일반적인 연산 방법 및 몽고메리 알고리즘의 경우에 비해 상대적으로 더 짧은 수행 시간을 나타내고 있다.

[0073]

이상에서 설명한 바와 같이, 확장된 NIST 소수는 통상적인 NIST 소수가 갖는 장점(빠른 모듈러 감산이 가능함을 의미한다.)을 그대로 가질 뿐만 아니라, 통상적인 NIST 소수에서 제공하지 못하는 다수의 비트 크기의 소수를 제공할 수 있다. 이하에서는 도 9를 참조하여 확장된 NIST 소수의 개수에 대해 설명한다.

[0074]

도 9는 본 발명의 실시예들이 채택하고 있는 확장된 NIST 소수와 종래의 NIST 소수의 개수를 비교하여 도시한 도면으로서, $wt(q)=2 \sim 5$ 를 만족하는 32~2048 비트의 확장된 NIST 소수의 개수를 측정한 결과이다.

도 9를 참조할 때, 본 발명이 속하는 기술 분야에서 통상의 지식을 가진 자는 $wt(q)$ 에 따라 비교할 때 확장된 NIST 소수의 개수가 종래의 NIST 소수에 비해 월등히 많은 것을 확인할 수 있으며, 다양한 크기의 소수체를 위해 확장된 NIST 소수를 적용할 수 있음을 알 수 있다.

[0075]

상기된 본 발명의 다양한 실시예들에 따르면, 통상적인 NIST 소수의 형태를 변형하여 암호학적으로 유용한 확장된 NIST 소수를 정의하고, 확장된 NIST 소수를 이용한 모듈러 곱셈, 모듈러 지수승을 제안함으로써, 타원 곡선 및 페어링 암호 알고리즘에서 활용할 수 있는 충분한 수의 소수체를 제공할 수 있으며, 몽고메리 곱셈 및 지수승 알고리즘보다 효율적인 모듈러 연산을 수행할 수 있으므로, 소수체 기반의 타원 곡선 암호 시스템의 페어링 기반 암호 시스템의 성능을 향상시킬 수 있다.

[0076]

한편, 본 발명의 실시예들은 컴퓨터로 읽을 수 있는 기록 매체에 컴퓨터가 읽을 수 있는 코드로 구현하는 것이 가능하다. 컴퓨터가 읽을 수 있는 기록 매체는 컴퓨터 시스템에 의하여 읽혀질 수 있는 데이터가 저장되는 모든 종류의 기록 장치를 포함한다.

[0077]

컴퓨터가 읽을 수 있는 기록 매체의 예로는 ROM, RAM, CD-ROM, 자기 테이프, 플로피디스크, 광 데이터 저장장치 등이 있으며, 또한 캐리어 웨이브(예를 들어 인터넷을 통한 전송)의 형태로 구현하는 것을 포함한다. 또한, 컴퓨터가 읽을 수 있는 기록 매체는 네트워크로 연결된 컴퓨터 시스템에 분산되어, 분산 방식으로 컴퓨터가 읽을 수 있는 코드가 저장되고 실행될 수 있다. 그리고 본 발명을 구현하기 위한 기능적인(functional) 프로그램, 코드 및 코드 세그먼트들은 본 발명이 속하는 기술 분야의 프로그래머들에 의하여 용이하게 추론될 수 있다.

[0078]

이상에서 본 발명에 대하여 그 다양한 실시예들을 중심으로 살펴보았다. 본 발명에 속하는 기술 분야에서 통상의 지식을 가진 자는 본 발명이 본 발명의 본질적인 특성에서 벗어나지 않는 범위에서 변형된 형태로 구현될 수 있음을 이해할 수 있을 것이다. 그러므로 개시된 실시예들은 한정적인 관점이 아니라 설명적인 관점에서 고려되

어야 한다. 본 발명의 범위는 전술한 설명이 아니라 특허청구범위에 나타나 있으며, 그와 동등한 범위 내에 있는 모든 차이점은 본 발명에 포함된 것으로 해석되어야 할 것이다.

도면

도면1

$$\begin{aligned}
 c \equiv & c_{17}2^{256} + c_{16}2^{224} + c_{15}2^{192} + c_{14}2^{160} + c_{13}2^{128} + c_{12}2^{96} + c_{11}2^{64} + c_{10}2^{32} + c_9 + c_0 \pmod{q} \\
 & - c_{14}2^{256} - c_{13}2^{224} - c_{12}2^{192} - c_{11}2^{160} - c_{10}2^{128} - c_92^{96} + c_22^{64} + c_12^{32} + c_0 \pmod{q} \\
 & + c_82^{256} + c_72^{224} + c_62^{192} + c_52^{160} + c_42^{128} + c_32^{96} + c_22^{64} + c_12^{32} + c_0 \pmod{q}
 \end{aligned}$$

도면2

Algorithm 1 모듈로 q 감산

Input: $q = 2^{288} + 2^{96} - 1$, $0 \leq c < p_{285}^2$,
 기저 2^{32} 인 $c = (c_{17}, \dots, c_0)$

Output: $c \bmod q$

1. 288비트 정수 정의 :

$$\begin{aligned}
 s_1 &= (c_8, c_7, c_6, c_5, c_4, c_3, c_2, c_1, c_0), \\
 s_2 &= (-c_{14}, -c_{14}, -c_{12}, -c_{11}, -c_{10}, -c_9, 0, 0, 0), \\
 s_3 &= (0, 0, 0, c_{14}, c_{13}, c_{12}, c_{11}, c_{10}, c_9), \\
 s_4 &= (c_{17}, c_{16}, c_{15}, c_{17}, c_{16}, c_{15}, -c_{17}, -c_{16}, -c_{15}).
 \end{aligned}$$

2. $(s_1 + s_2 + s_3 + s_4 \bmod q)$ 반환

도면3

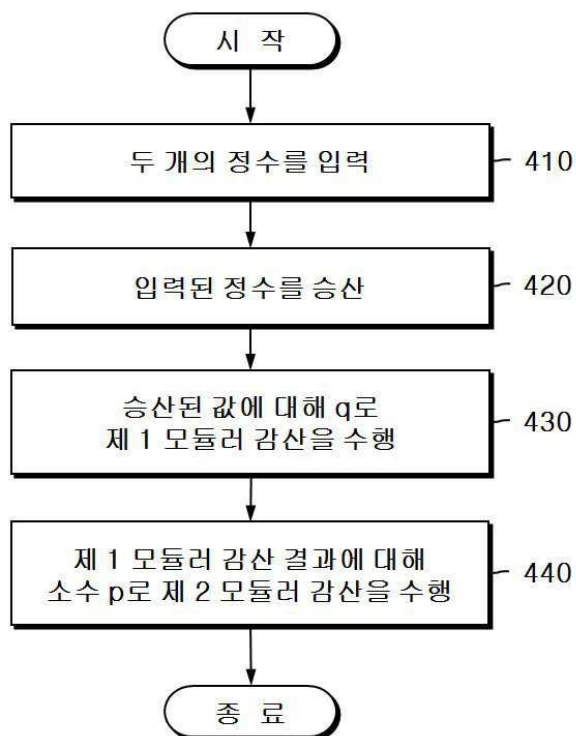
Algorithm 2 모듈로 p_{285} 감산

Input: $c \bmod q$, 모듈러스 ENP p_{285}

Output: $c \bmod p_{285}$

1. $A \leftarrow c \bmod q$
2. $P \leftarrow p_{285}$
3. **while** $A > P$
4. $P \leftarrow P \ll 1$
5. **end while**
6. $P \leftarrow P \gg 1$
7. $A \leftarrow A - P$
8. **if** $A > p_{285}$
9. **goto** 2
10. **end if**
11. A 반환

도면4



도면5

Algorithm 3 모듈러스 ENP의 모듈러 곱셈

Input: $A = (a_{n-1}, \dots, a_0)_b$, $B = (b_{n-1}, \dots, b_0)_b$,
 $b = 2^{32}$

Output: $S = (s_{n-1}, \dots, s_0)_b$, $S = AB \bmod p$

1. $A \cdot B$ 연산
2. $U = A \cdot B \bmod q$ 연산
3. $S = U \bmod p$ 연산

도면6

Algorithm 4 모듈러 지수승

Input: $g = (g_n g_{n-1} \dots g_1 g_0)_b$,
 양수 $e = (e_n e_{n-1} \dots e_1 e_0)_2$,

Output: $g^e \bmod p$,

1. $A \leftarrow 1$
2. **for** $i = t$ **to** 0 **do**
3. $A \leftarrow A \cdot A$
4. $A \leftarrow A \bmod q$
5. **if** $e_i = 1$ **then**
6. $A \leftarrow A \cdot g$
7. $A \leftarrow A \bmod q$
8. **end if**
9. **end for**
10. $A \leftarrow A \bmod p$
11. A 반환

도면7

모듈러 곱셈 알고리즘	연산 효율	추가 연산
일반	$(2n^2 + 3n)t_m + nt_d$	-
몽고메리	$(4n^2 + 4n)t_m$	사전계산 $((2n^2 + 3n)t_m + nt_d + t_f)$
ENP	$n^2 t_m$	시프트 및 뺄셈 연산
모듈러 지수승 알고리즘	연산 효율	추가 연산
일반	$(3/2n^2 t + 1120t)t_m + (32t)t_d$	-
몽고메리	$3n(n+1)(t+1)t_m$	사전계산 $((2n^2 + 3n)t_m + nt_d + t_f)$
ENP	$(3/2n^2 t)t_m$	시프트 및 뺄셈 연산

t_f : 워드 역원 연산 시간

도면8

알고리즘	모듈러 곱셈(μs)	모듈러 지수승(μs)
일반	18	36,757
몽고메리	30	12,383
ENP	10	11,546

도면9

2의 지수승 개수 소수	2	3	4	5	적용가능 소수체
NIST 소수	11	45	472	23,119	130
ENP	84	91	911	88,799	1,006