(19) 대한민국특허청(KR)
(12) 등록특허공보(B1)

(45) 공고일자 2016년07월07일
(11) 등록번호 10-1636809
(24) 등록일자 2016년06월30일

(51) 국제특허분류(Int. Cl.)
G06F 7/494 (2006.01)
(21) 출원번호 10-2014-0194809
(22) 출원일자 2014년12월31일
심사청구일자 2014년12월31일
(56) 선행기술조사문헌
KR1020100025403 A
KR1020090090881 A
KR1020100026358 A

(73) 특허권자
고려대학교 산학협력단
서울특별시 성북구 안암로 145, 고려대학교 (안암동5가)
(72) 발명자
홍석희
서울특별시 도봉구 마들로 859-19, 103동 107호 (도봉동, 한신아파트)
권희택
경기도 수원시 권선구 동수원로58번길 21, 101동 302호 (곡반정동, 한솔아파트)
(뒷면에 계속)
(74) 대리인
특허법인충현

전체 청구항 수 : 총 3 항 심사관 : 지정훈

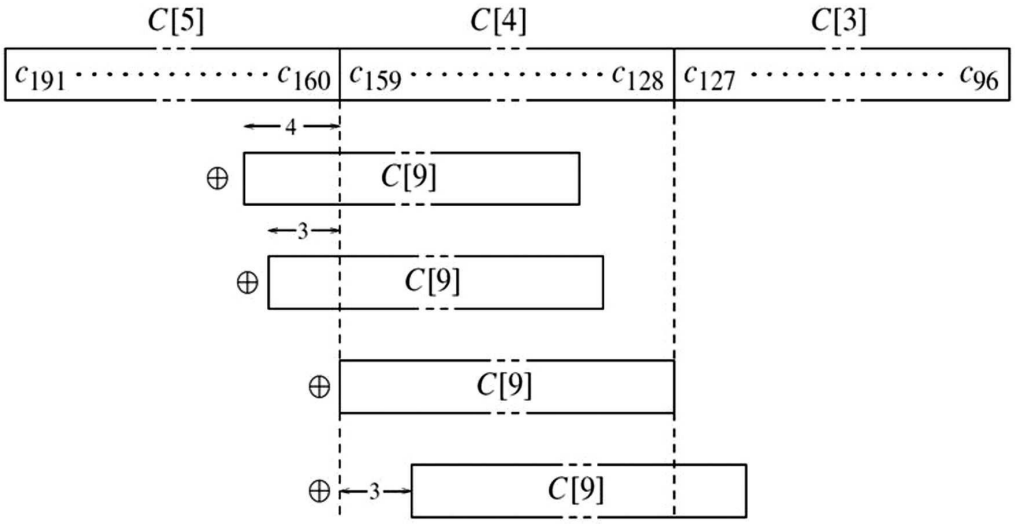
(54) 발명의 명칭 **타원 곡선 암호용 이진체의 감산 방법**

(57) 요 약

본 발명은 타원 곡선 암호용 이진체의 감산 방법에 관한 것으로서, 보다 바람직하게는 입력부가 이진체의 차수를 나타낸 복수 개의 워드를 메모리부로부터 입력받아 레지스터에 로드하는 단계; 및 감산부가 상기 레지스터에 로드된 복수 개의 워드에 대하여 삼항 또는 오항 다항식으로 이루어지는 기약다항식으로 감산 연산하는 단계;를 포함한다.

이러한 구성에 의해, 본 발명의 타원 곡선 암호용 이진체의 감산 방법은 메모리에 저장된 데이터를 레지스터로 로드하고, 감산 연산에 따른 결과값을 다시 메모리에 저장하여 메모리 로드와 저장을 최소한으로 설계하여 어셈블리 언어로 구현함으로써, 감산 연산 향상율이 증가할 수 있는 효과가 있다.

대 표 도 - 도1



(72) 발명자

박동원

서울특별시 서초구 명달로9길 61-12 (방배동)

조성민

서울특별시 성북구 화랑로41길 9 2층 201호 (장위동)

이 발명을 지원한 국가연구개발사업

과제고유번호	R1008585
부처명	미래창조과학부
연구관리전문기관	정보통신산업진흥원
연구사업명	ITRC
연구과제명	제어시스템보안연구
기 여 율	1/1
주관기관	고려대학교 정보보호대학원
연구기간	2014.01.01 ~ 2014.12.31

명 세 서

청구범위

청구항 1

입력부가 이진체의 차수를 나타낸 복수 개의 워드를 메모리부로부터 입력받아 레지스터에 로드하는 단계; 및

감산부가 상기 레지스터에 로드된 복수 개의 워드에 대하여 삼항 또는 오항 다항식으로 이루어지는 기약다항식으로 감산 연산하는 단계;를 포함하되,

상기 기약다항식으로 감산 연산하는 단계는,

상기 복수 개의 워드를 1-비트 레프트 시프트 및 워드 시프트 연산을 통해 감산하는 것을 특징으로 하는 타원곡선 암호용 이진체의 감산 방법.

청구항 2

제1항에 있어서,

상기 메모리부가 감산된 상기 복수 개의 워드를 갱신하여 내부에 저장하는 단계;

를 더 포함하는 것을 특징으로 하는 타원 곡선 암호용 이진체의 감산 방법.

청구항 3

삭제

청구항 4

제1항 내지 제2항 중 어느 한 항에 따른 방법을 컴퓨터로 실행하기 위한 프로그램이 기록된 컴퓨터 판독가능 기록매체.

발명의 설명

기 술 분 야

[0001] 본 발명은 타원 곡선 암호용 이진체의 감산 방법에 관한 것으로, 특히 8-비트 ATmega 128 프로세서 환경에서 효율적으로 이진체를 감산할 수 있는 타원 곡선 암호용 이진체의 감산 방법에 관한 것이다.

배 경 기 술

[0002] 각종 전자 상거래의 급격한 증가로 인하여, 전자 상거래 시스템뿐만 아니라, 보안 시스템에서도 다양한 암호 방식이 널리 연구되고 있다.

[0003] 그 중에서도 타원 곡선 시스템을 이용한 공개키 암호방식(ECC: Elliptic Curve Cryptosystem, 이하 ECC라고 한다.)은 이산대수의 난해성에 기반을 둔 공개키 방식의 암호 알고리즘으로서, RSA 알고리즘에 이어 전자 상거래의 핵심기술로 최근 주목받고 있다. 이산대수의 난해성에 기반을 둔 타원곡선 암호체계는 DSA 알고리즘과 소인수분해 기반의 RSA 등과 함께 공개키 방식의 일종이다. 상기 ECC는 지난 85년 N.Koblitz와 V.S.Miller 가 RSA 암호화방식에 대한 대안으로 처음 제안하였으며, 현재 보안시스템 및 전자상거래 솔루션업체들에 의해 폭넓게 채용되고 있는 추세이다.

[0004] 특히, 상기 ECC는 타원곡선을 이용하여 유한체위에서 새로운 공개키 암호알고리즘을 만들지는 않지만 기존의 공개키 알고리즘을 타원곡선을 이용하여 구현했다는 특징을 지니고 있다. 보다 정확하게는 상기 ECC는 특정 암호

알고리즘이 아니라 암호알고리즘을 구현해 볼 수 있는 수학적인 장소를 제공하고 있는 것으로 RSA, 디피 헬만(Diffie-Hellman), 엘가말(Elgamal) 등의 알고리즘을 기존의 정수공간이 아닌 타원 곡선위에서 구현할 수 있게 한다. 타원 곡선위에서의 암호 구현은 수학적 복잡도 때문에 동일한 키 크기의 정수위에서 구현하는 것 보다 훨씬 강도가 강하므로 향후 암호 알고리즘의 대부분이 ECC로 바뀔 가능성이 매우 높다.

[0005] 이러한 타원곡선 암호시스템은 유한체의 곱셈 군에 근거한 시스템으로서 다양한 암호 시스템을 설계할 수 있고, 안전한 특성을 갖는다.

[0006] 특히, 군(group)을 제공할 수 있는 다양한 타원곡선을 활용할 수 있다. 즉, 다양한 암호시스템 설계가 용이하다. 또한, 모든 사용자가 같은 기저체 K를 사용한다 할 지라도 각 사용자가 다른 곡선을 선택할 수 있다는 것이다. 즉, 모든 사용자는 연산을 수행하기 위해 같은 하드웨어를 사용할 수 있으며, 추가 보안을 위해 주기적으로 곡선을 바꿀 수 있다.

[0007] 또한, 공개키 암호시스템의 이론적 안전도를 조사하기 위해서는 먼저 시스템을 공격하는 데에 있어 그 시스템의 기반이 되는 수학적 문제를 푸는 것이 어느 정도 요구되는가를 분석하는 것이다. 실제 소인수문제에 기반을 둔 공개키 암호시스템 RSA, 이산대수문제에 기반을 둔 공개키 암호시스템 DSA 및 타원곡선 암호시스템 모두 수년간 정밀한 분석이 이루어져 왔고, 그러한 시스템 공격방법은 그 기반이 되는 수학적 분제를 푸는 것이라고 알려져 있다. 따라서, 이제 "기반이 되는 어려운 문제"가 무엇인지를 분석하여야 한다. 타원곡선 이산대수문제는 타원곡선의 작은 클래스인 초 특이 타원곡선일 때 상대적으로 쉽다는 것이 알려져 있다. 그러나 이러한 취약점은 쉽게 확인될 수 있으며, 그러므로 쉽게 피할 수도 있다. 모듈러 p에서의 소인수 분해 문제와 이산대수문제는 일반적으로 Sub-exponential time 알고리즘이 알려져 있다. Sub-expoinential time 알고리즘이란 여전히 어려운 문제로 알려져 있으나, 완전지수복잡도(fully expoenetail time) 알고리즘만을 허용하는 문제보다는 쉽다는 것을 의미한다. 이 알고리즘의 수행시간은 상수 c에 대하여 그러나(초 특이 타원곡선을 피하면서) 이 군에서의 sub-exponential time algorithms 이 존재하지 않는다. 즉, 안전한 암호시스템을 설계하는 것이 용이하다. RSA와 DSA가 1024비트의 모듈러를 사용하는 반면, ECC는 160비트 모듈러로도 충분함을 보여준다. 더욱이 키 크기가 증가할수록 그 비율은 더욱 증가한다.

[0008] 이러한 ECC에 적용하기 위한 이진체 감산 방법을 수행하기 위해서는 자원이 제한된 환경에서 중복되는 메모리 접근에 대한 부하가 상대적으로 크다는 문제점이 발생했다.

선행기술문헌

특허문헌

[0009] (특허문헌 0001) KR 10-2010-0025403 (다항식 기저 기반의 이진체 병렬 곱셈 장치 및 방법과 이를 이용한 마이크로프로세서, 고려대학교 산학협력단) 2010.03.09.

발명의 내용

해결하려는 과제

[0010] 상기와 같은 종래 기술의 문제점을 해결하기 위해, 본 발명은 8 비트 기반 ATmega 128 프로세서와 같이 자원이 제한된 환경에서는 이진체의 감산 방법을 통해 레지스터를 효율적으로 사용하여 메모리 접근 중복에 대한 부하를 감소시킬 수 있는 타원 곡선 암호용 이진체의 감산 방법을 제공하고자 한다.

과제의 해결 수단

[0011] 위와 같은 과제를 해결하기 위한 본 발명의 한 실시 예에 따른 타원 곡선 암호용 이진체의 감산 방법은 입력부가 이진체의 차수를 나타낸 복수 개의 워드를 메모리부로부터 입력받아 레지스터에 로드하는 단계; 및 감산부가 상기 레지스터에 로드된 복수 개의 워드에 대하여 삼항 또는 오항 다항식으로 이루어지는 기약다항식으로 감산 연산하는 단계;를 포함한다.

[0012] 보다 바람직하게는 상기 메모리부가 감산된 상기 복수 개의 워드를 갱신하여 내부에 저장하는 단계;를 더 포함할 수 있다.

[0013] 보다 바람직하게는 상기 복수 개의 워드를 1-비트 레프트 시프트 및 워드 시프트 연산을 통해 감산하는 감산부가 상기 레지스터에 로드된 복수 개의 워드를 상기 기약다항식으로 감산 연산하는 단계를 포함할 수 있다.

발명의 효과

[0014] 본 발명의 타원 곡선 암호용 이진체의 감산 방법은 메모리에 저장된 데이터를 레지스터로 로드하고, 감산 연산에 따른 결과값을 다시 메모리에 저장하여 메모리 로드와 저장을 최소한으로 설계하여 어셈블리 언어로 구현함으로써, 감산 연산 향상율이 증가할 수 있는 효과가 있다.

도면의 간단한 설명

[0015] 도 1은 32 비트 프로세서 기반의 이진체 감산 방법을 나타낸 도면이다.

도 2는 32 비트 프로세서에 적용된 종래 기술에 따른 이진체의 감산 방법을 나타낸 도면이다.

도 3은 본 발명의 일 실시 예에 따른 타원 곡선 암호용 이진체의 감산 방법의 순서도이다.

도 4는 8비트 기반 ATmega128 프로세서의 범용 레지스터를 나타낸 도면이다.

도 5는 8 비트 프로세서 기반 타원 곡선 암호용 이진체의 감산 방법을 나타낸 도면이다.

도 6은 감산 연산에 대한 루프 카운터별 감산과정의 동작과정을 나타낸 도면이다.

도 7은 감산 과정에서 각 워드가 감산 연산되어 최종적으로 이동하는 워드를 나타낸 테이블이다.

도 8은 종래 기술과 본 발명간의 효율성을 비교를 나타낸 표이다.

발명을 실시하기 위한 구체적인 내용

[0016] 이하, 본 발명을 바람직한 실시 예와 첨부한 도면을 참고로 하여 본 발명이 속하는 기술 분야에서 통상의 지식을 가진 자가 용이하게 실시할 수 있도록 상세히 설명한다. 그러나 본 발명은 여러 가지 상이한 형태로 구현될 수 있으며, 여기에서 설명하는 실시 예에 한정되는 것은 아니다.

[0017] 이하에서는 차세대 공개키 암호로 대두되고 있는 타원 곡선 암호(ECC: Elliptic Curve Cryptography)에 사용되는 이진체 감산 방법에 대하여 자세히 살펴보도록 한다.

[0018] 먼저, 본 발명이 사용되는 타원 곡선 암호에 간략히 살펴보면, 타원곡선암호체계는 이산대수의 난해성에 기반을 둔 공개키 방식의 암호 알고리즘으로 RSA 알고리즘에 이어 전자 상거래의 핵심기술로 최근 주목받고 있다. 이산대수의 난해성에 기반을 둔 타원곡선 암호체계는 DSA 알고리즘과 소인수분해 기반의 RSA 등과 함께 공개키 방식의 일종이다. ECC는 지난 85년 N.Koblitz와 V.S.Miller 가 RSA 암호화방식에 대한 대안으로 처음 제안하였으며, 현재 보안시스템 및 전자상거래 솔루션업체들에 의해 폭넓게 채용되고 있는 추세이다. ECC는 타원곡선을 이용, 유한체위에서 새로운 공개키 암호알고리즘을 만들지는 않지만 기존의 공개키 알고리즘을 타원곡선을 이용해 구현했다는 특징을 지니고 있다.

[0019] 또한 이진체는 임의 m(유한체의 차수)에 대한 2m 원소를 갖는 유한체 GF(2m)이다. 이 유한체의 원소는 길이 m의 비트 스트링이며, 유한체 산술은 비트의 연산 용어로 구현된다. 이러한 이진체에 적용 가능한 타원 곡선은 y2 + xy = x3 + ax2 + b 형식을 취한다.

[0020] 방정식을 만족하는 가능한 곡선의 수는 실제로 무한대라고 할 수 있지만, ECC에 관련해서는 소수의 곡선만 사용된다. 이 곡선들은 FIPS 186에 명시되어 있으며, NIST 권장 타원 곡선이라고 한다. 각각의 곡선은 각 이름과 소수 모듈로 p, 소수 위수 n, 계수 a, 계수 b 및 곡선상의 기저점 G(x, y)의 x와 y 좌표로 이루어진 영역 매개변수 세트로 구성된다.

[0021] 이진체 $GF(2^m)$의 원소들을 비트(bit)의 나열하여 기약다항식으로 표현할 수 있다. 이때, 각 비트들은 최고 차항이 $(m-1)$ 이하인 다항식의 계수들을 나타낼 수 있다. 상기 이진체의 원소들은 기약다항식에 따라 구성 방식이 달라지며 각 원소들끼리의 곱셈 연산은 기약다항식을 이용한 감산 연산을 통해서 수행 될 수 있다.

[0022] 이하에서는 종래기술에 따라 이진체에서 기약다항식에 의존하는 감산 방법인 Fast reduction 방법에 대하여 살펴보도록 한다.

[0023] 도 1은 32 비트 프로세서 기반의 이진체 감산 방법을 나타낸 도면이다.

[0024] 도 1에 도시된 바와 같이, 종래 기술에 따른 이진체 감산 방법은 루프 한 번당 한 워드씩 감산을 진행한다. 예를 들어, 32 비트 프로세서 환경에서 기약다항식이 $f(x)=x^{163}+x^7+x^6+x^3+1$ 일 때, 입력 값의 10번째 워드 $(C[9] = x^{319} + \cdots + x^{288})$를 하기의 수학식 1과 같이 표현할 수 있다.

수학식 1

[0025]
$$
\begin{aligned}
x^{288} &\equiv x^{132}+x^{131}+x^{128}+x^{125} \mod(f(x)) \\
x^{289} &\equiv x^{133}+x^{132}+x^{129}+x^{126} \mod(f(x)) \\
&\vdots \\
x^{319} &\equiv x^{163}+x^{162}+x^{159}+x^{156} \mod(f(x))
\end{aligned}
$$

[0026] 상기 합동식의 오른쪽 4열을 고려해볼 때, C[9]의 감산 연산은 해당하는 하위 워드 C[5], C[4], C[3]에 4번의 배타적 논리합(XOR) 연산으로 처리가 가능하다. C[9]의 최하위 비트는 각각 132, 131, 128, 125번째 비트들에 배타적 논리합(XOR) 연산으로 처리된다.

[0027] 도 2는 32 비트 프로세서에 적용된 종래기술에 따른 이진체 감산 방법을 나타낸 도면이다.

[0028] 도 2에 도시된 바와 같이, 32 비트 프로세서 환경에서 곱셈 연산이나 제곱 연산의 결과로 워드 크기가 2배로 늘어난 값을 입력값을 수신하여 기약다항식 $f(x)=x^{163}+x^7+x^6+x^3+1$ 에 대한 Fast reduction 방법을 수행한다.

[0029] 이하, 도 3을 참조하여 본 발명의 일 실시 예에 따른 타원 곡선 암호용 이진체의 감산 방법에 대하여 살펴보도록 한다.

[0030] 도 3은 본 발명의 타원 곡선 암호용 이진체의 감산 방법의 순서도이다.

[0031] 도 3에 도시된 바와 같이, 본 발명의 타원 곡선 암호용 이진체의 감산 방법은 입력부가 이진체의 차수를 나타낸 복수 개의 워드를 메모리부로부터 입력받아 레지스터에 로드한다(S110).

[0032] 감산부가 상기 레지스터에 로드된 복수 개의 워드에 대하여 선택된 상기 기약다항식으로 감산 연산한다(S120).

[0033] 특히, 상기 다항식은 삼항 또는 오항 다항식으로 이루어지는데, 이진체 연산을 위해, 삼항다항식 x^m+x^a+1 또는 오항다항식 $x^m+x^a+x^b+x^c+1$을 기약다항식으로 선택한다. 예를 들어, 만약 삼항 기약다항식이 존재한다면 그 중에서 가장 차수가 낮은 a를 선택하고, 삼항 기약다항식이 존재하지 않으면 오항 기약다항식에서 차수가 가장 낮은 a를 선택한다. 이어서, b, c도 차수가 가장 낮은 것으로 기

약다항식을 선택한다. 위와 같은 방식으로 모든 m 에 대해 기약다항식을 선택할 수 있다.

[0034] 또한, 상기 복수 개의 워드를 1-비트 레프트 시프트 및 워드 시프트 연산을 통해 감산할 수 있다.

[0035] 상기 메모리부가 감산된 상기 복수 개의 워드를 갱신하여 내부에 저장한다(S130).

[0036] 상술한 본 발명은 32 비트 기반 프로세서 환경이 아닌 8 비트 기반 ATmega128 프로세서 환경에서 Fast reduction 알고리즘을 기약다항식 $f(x)=x^{271}+x^{207}+x^{175}+x^{111}+1$ 적용하는 경우를 살펴보기에 앞서, 본 발명이 적용되는 8 비트 기반의 ATmega128 프로세서에 대하여 간략히 살펴보도록 한다.

[0037] 이러한 ATmega128 프로세서는 무선 센서네트워크 환경에서 대표적으로 사용되는 MICA2/MICAz 센서모트에 탑재된 8 비트 기반의 마이크로프로세서이다. 이러한 ATmega128 프로세서는 성능과 병렬성을 최대화하기 위해, 프로그램과 데이터를 위한 메모리와 버스를 구분하는 하버드(Havard) 구조이며 프로그램 메모리 내의 명령어들은 하나의 명령어가 수행될 때, 다음에 수행될 명령어가 프로그램 메모리로부터 프리페치(prefetch)되는 한 단계의 파이프라인으로 처리된다. 이러한 처리 과정을 통하여 ATmega128 프로세서는 매 클럭 사이클마다 명령어를 수행할 수 있다.

[0038] 이러한 ATmega128 CPU 내의 범용 레지스터 파일은 도 4에 도시된 바와 같이, ALU가 1 클럭 사이클 안에 레지스터 파일로부터 두 개의 데이터에 대하여 명령어가 수행되고, 그 결과값이 다시 레지스터 파일에 저장된다. 32개의 레지스터 중 6개의 레지스터(R31:R30, R29:R28, R27:R26)는 세 개의 16-bit 간접주소 레지스터(X, Y, Z)로 사용되어 효율적인 주소 계산이 가능하다. 나머지 26개의 레지스터들을 이용하여 유한체 연산을 수행할 수 있다.

[0039] 이러한 8 비트 기반 ATmega128 프로세서 환경에서 Fast reduction 알고리즘을 기약다항식 $f(x)=x^{271}+x^{207}+x^{175}+x^{111}+1$ 에 적용하면 도 5와 같이 나타낼 수 있다. 특히, 8 비트 프로세서 환경에서 271 비트 원소는 34개의 8 비트 워드로 구성되는데, 유한체의 감산 연산 시에는 유한체 곱셈 연산과 유한체 제곱 연산의 결과를 입력받기 때문에 최대 68개의 워드를 입력받는다. 따라서 C[67]부터 C[0]까지의 입력을 C[33]부터 C[0]으로 감산 연산이 진행된다.

[0040] 도 6은 본 발명에 따른 타원 곡선 암호용 이진체의 감산 방법을 수행하는데 있어서, 감산 연산에 대한 루프 카운터의 동작과정을 나타낸 도면이다.

[0041] 도 6에 도시된 바와 같이, 유한체 곱셈 연산과 유한체 제곱 연산의 결과값을 기약다항식 $f(x)=x^{271}+x^{207}+x^{175}+x^{111}+1$ 으로 감산하는 루프 카운터 i 가 66에서 62에 대하여 동작할 때를 나타낸 것이다.

[0042] 도 6에서 한 줄 밑줄로 표기된 연산은 데이터가 메모리로부터 중복 로드하거나 결과값을 중복 저장하는 부분을 표시한 것이다. 이처럼 한줄 밑줄로 표기된 C[32], C[30], C[29], C[28]은 5번의 루프를 수행할 때 각각 두 번씩 메모리로부터 로드하고 감산 연산을 수행하여 갱신한 후에 두 번씩 다시 저장한다. 이와 같이 중복 로드하거나 저장하는 값들을 레지스터에서 재사용하여 메모리 접근으로 인한 부하를 줄일 수 있다.

[0043] 또한 루프 카운터 i 가 66일 때와 i 가 62일 때 두줄 밑줄로 표기된 C[54]를 각각 로드하여 감산 연산 루프를 수행하여 다시 저장하게 된다. 이처럼 루프 카운터가 진행됨에 따라서 동일한 워드를 다시 호출하는 경우가 발생하게 된다. 따라서 이를 효율적으로 처리할 수 있는 방법이 필요하다.

[0044] 감산하고자하는 대상의 상위 워드를 살펴보면 x^{272} 부터 차례로 8 비트씩 각각 하나의 워드를 형성한다. 여기서 각 워드들은 오직 1-비트 레프트 시프트(1-bit left shift)와 워드 시프트 연산을 통해 감산을 수행한다.

[0045] 예를 들어, $x^{272} = x^{208} + x^{176} + x^{112} + x \pmod{f(x)}$ 이고 272와 208, 176, 112의 차는 각각 64, 32, 64로 모두 8의 배수이므로 워드 시프트 연산으로 처리 가능하고 x 는 1-비트 레프트 시프트로 처리하여 감산이 가능하다.

[0046] 도 7은 감산 과정에서 각 워드가 감산되어 최종적으로 이동하는 워드를 나타낸 테이블이다.

[0047] 도 7에 도시된 바와 같이, 감산된 1번째 워드 C[0]에는 C[34], C[42], C[46], C[50], C[54], C[62] 워드가 각각 1 비트 레프트 시프트된 상태로 배타적 논리합(XOR) 연산이 발생한다. 또한 15 번째 워드 C[14]에는 C[48], C[56], C[60], C[64]가 1비트 레프트 시프트된 상태와 C[34], C[42], C[46], C[50], C[54], C[62]가 배타적 논리합 연산이 발생한다.

[0048] 이러한 특성을 고려하여 효율적으로 워드를 레지스터로 로드하여 재사용하고 감산 연산이 끝난 후에만 메모리에 저장함으로써 메모리 접근 연산을 최소화 할 수 있다.

[0049] 이에 따라, 본 발명에 따른 감산 연산을 수행 시 메모리에서 로드하는 연산은 C[0]부터 C[41]까지 1회, C[42]부터 C[67]까지는 2회가 필요하며 연산이 끝난 후 메모리로 저장하는 연산은 각 워드의 연산이 종료된 후에만 저장하기 때문에 C[0]부터 C[33]까지 1회만 사용하게 된다.

[0050] 특히, 본 발명의 타원 곡선 암호용 이진체의 감산 방법은 레지스터 내부의 데이터에 대하여 수행되는 연산에는 1클럭 사이클만이 소요되는데 비하여, 메모리에 저장된 데이터를 레지스터로 로드하거나 연산의 결과값을 다시 메모리에 저장하는 데는 2클럭 사이클이 소모된다. 따라서 감산 알고리즘에 메모리 로드와 저장을 최소한으로 설계하여 어셈블리 언어로 구현하였다.

[0051] 도 8은 종래 기술에 따른 Fast reduction 방법과 본 발명에 따른 감산 방법을 각각 어셈블리 언어로 구현 시, 성능을 클럭 사이클 단위로 비교한 표이다.

[0052] 도 8에 도시된 바와 같이, $GF(2^{271})$ 에서 기약다항식 $f(x) = x^{271} + x^{207} + x^{175} + x^{111} + 1$ 에 Fast reduction을 구현 시 1,084 클럭 사이클이 소모되었고, 제안하는 감산 연산 알고리즘을 사용하여 구현 시 482 클럭 사이클이 소모되어 약 56% 정도 속도가 향상된 것을 알 수 있다.

[0053] 다만 이러한 수치는 본 발명인 이진체의 감산 방법에 대한 향상율이고, 실제 공개키 암호시스템의 구현을 위해서는 부가적인 처리들이 필수적으로 추가된다. 예를 들어, 입력 또는 출력 메모리의 주소 값이 들어 있는 레지스터는 사용하기 전에 미리 push 명령어로 스택에 넣었다가 감산이 종료된 후에 다시 pop 명령어로 주소 값을 받아와야 다음 연산을 진행하는데 지장을 주지 않는다. 추가 연산으로 인해 종래 기술에 따른 Fast reduction은 1,084 클럭 사이클에서 1,126 클럭 사이클로 늘어나고, 본 발명에서는 482 클럭 사이클에서 524 클럭 사이클로 늘어난다. 추가된 클럭 사이클은 전체적인 향상율을 저하시키게 된다. 따라서 실질적인 감산 연산 향상율은 약 53%로 나타났다.

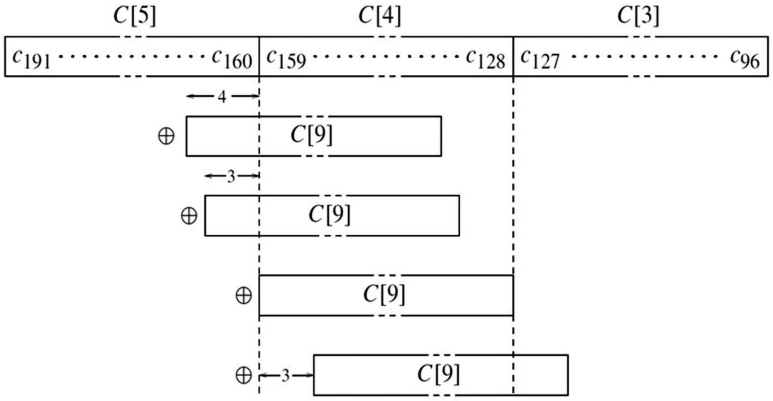
[0054] 본 발명의 실시예들은 다양한 컴퓨터 수단을 통하여 수행될 수 있는 프로그램 명령 형태로 구현되어 컴퓨터 판독 가능 매체에 기록될 수 있다. 상기 컴퓨터 판독 가능 매체는 프로그램 명령, 데이터 파일, 데이터 구조 등을 단독으로 또는 조합하여 포함할 수 있다. 상기 매체에 기록되는 프로그램 명령은 본 발명을 위하여 특별히 설계되고 구성된 것들이거나 컴퓨터 소프트웨어 당업자에게 공지되어 사용 가능한 것일 수도 있다. 컴퓨터 판독 가능 기록 매체의 예에는 하드 디스크, 플로피 디스크 및 자기 테이프와 같은 자기 매체(magnetic media), CD-ROM, DVD와 같은 광기록 매체(optical media), 플롭티컬 디스크(floptical disk)와 같은 자기-광 매체(magneto-optical), 및 롬(ROM), 램(RAM), 플래시 메모리 등과 같은 프로그램 명령의 예에는 컴파일러에 의해 만들어지는 것과 같은 기계어 코드뿐만 아니라 인터프리터 등을 사용해서 컴퓨터에 의해서 실행될 수 있는 고급 언어 코드를 포함한다. 상기된 하드웨어 장치는 본 발명의 일 실시예들의 동작을 수행하기 위해 적어도 하나의 소프트웨어 모듈로서 작동하도록 구성될 수 있으며, 그 역도 마찬가지이다.

[0055] 본 발명의 타원 곡선 암호용 이진체의 감산 방법은 메모리에 저장된 데이터를 레지스터로 로드하거나 연산의 결과값을 다시 메모리에 저장하는 데는 2 클럭 사이클이 소모됨에 따라, 감산 알고리즘에 메모리 로드와 저장을 최소한으로 설계하여 어셈블리 언어로 구현함으로써, 감산 연산 향상율이 증가할 수 있는 효과가 있다.

[0056] 상기에서는 본 발명의 바람직한 실시 예에 대하여 설명하였지만, 본 발명은 이에 한정되는 것이 아니고 본 발명의 기술 사상 범위 내에서 여러 가지로 변형하여 실시하는 것이 가능하고 이 또한 첨부된 특허청구범위에 속하는 것은 당연하다.

도면

도면1



도면2

Input: $c(x)$ **which of degree is at most 324**

Output: $c(x) \pmod{x^{163}+x^7+x^6+x^3+1}$

1: for i **from 10 downto 6 do**

1.1: T ← C[i].

1.2: C[i-6] ← C[i-6]⊕(T<<29).

1.3: C[i-5]←C[i-5]⊕(T<<4)⊕(T<<3)⊕(T>>3).

1.4: C[i-4] ← C[i-4]⊕(T>>28)⊕(T>>29).

2: T ← C[5]>>3.

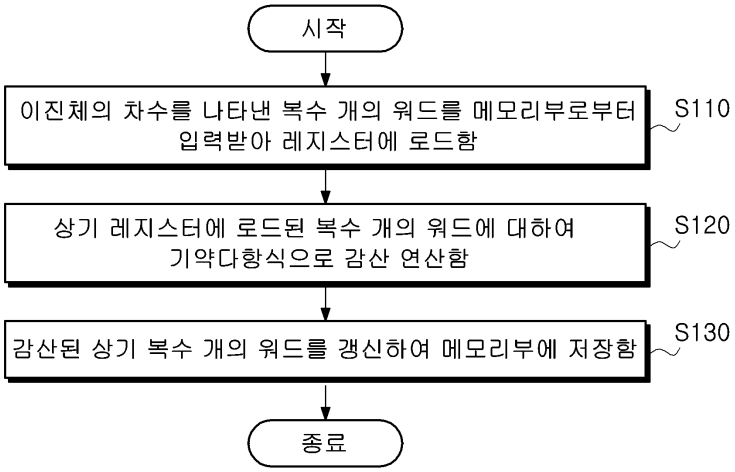
3: C[0] ← C[0]⊕(T<<7)⊕(T<<6)⊕(T<<3)⊕T.

4: C[1] ← C[1]⊕(T>>25)⊕(T>>26).

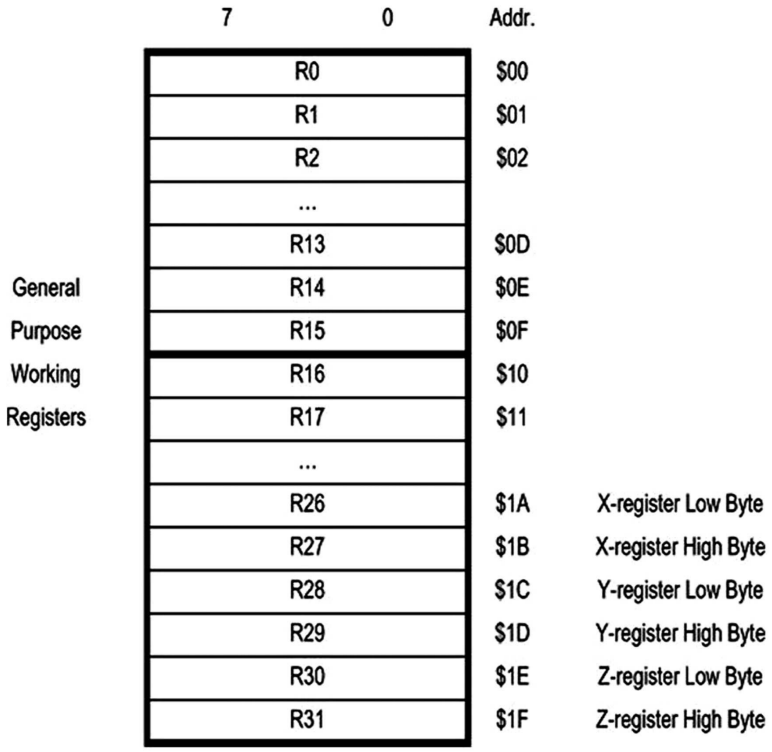
5: C[5] ← C[5]&0x7.

6. Return (C[5],···,C[0])

도면3


시작
이진체의 차수를 나타낸 복수 개의 워드를 메모리부로부터 입력받아 레지스터에 로드함
S110
상기 레지스터에 로드된 복수 개의 워드에 대하여 기약다항식으로 감산 연산함
S120
감산된 상기 복수 개의 워드를 갱신하여 메모리부에 저장함
S130
종료

도면4


7
0
Addr.
R0 $00
R1 $01
R2 $02
...
R13 $0D
General
R14 $0E
Purpose
R15 $0F
Working
R16 $10
Registers
R17 $11
...
R26 $1A X-register Low Byte
R27 $1B X-register High Byte
R28 $1C Y-register Low Byte
R29 $1D Y-register High Byte
R30 $1E Z-register Low Byte
R31 $1F Z-register High Byte

도면5

Output: $c(x)\ (\mathrm{mod}\ x^{271}+x^{207}+x^{175}+x^{111}+1)$

1: T ← C[67].
2: C[59] ← C[59]⊕T.
3: C[55] ← C[55]⊕T.
4: C[47] ← C[47]⊕T.
5: C[33] ← C[33]⊕(T<<1).
6: for i from 66 downto 34 do
  6.1: T ← C[i].
  6.2: C[i-8] ← C[i-8]⊕T.
  6.3: C[i-12] ← C[i-12]⊕T.
  6.4: C[i-20] ← C[i-20]⊕T.
  6.5: C[i-33] ← C[i-33]⊕(T>>7).
  6.6: C[i-34] ← C[i-34]⊕(T<<1).
7: T ← C[33]&0x80.
8: C[25] ← C[25]⊕T.
9: C[21] ← C[21]⊕T.
10: C[13] ← C[13]⊕T.
11: C[0] ← C[0]⊕(T>>7).
12: C[33] ← C[33]&0x7F.
13. Return (C[33],···,C[0])

도면6

루프 카운터	동작
i = 66	T = C[66] C[58] = C[58]⊕T C[54] = C[54]⊕T C[46] = C[46]⊕T C[33] = C[33]⊕(T>>7) C[32] = C[32]⊕(T<<1)
i = 65	T = C[65] C[57] = C[57]⊕T C[53] = C[53]⊕T C[45] = C[45]⊕T C[32] = C[32]⊕(T>>7) C[31] = C[31]⊕(T<<1)
i = 64	T = C[64] C[56] = C[56]⊕T C[52] = C[52]⊕T C[44] = C[44]⊕T C[31] = C[31]⊕(T>>7) C[30] = C[30]⊕(T<<1)
i = 63	T = C[63] C[55] = C[55]⊕T C[51] = C[51]⊕T C[43] = C[43]⊕T C[30] = C[30]⊕(T>>7) C[29] = C[29]⊕(T<<1)
i = 62	T = C[62] C[54] = C[54]⊕T C[50] = C[50]⊕T C[42] = C[42]⊕T C[29] = C[29]⊕(T>>7) C[28] = C[28]⊕(T<<1)

도면7

Word No.	1-bit left shift	Just moving
0	34,42,46,50,54,62	
1	35,43,47,51,55,63	
2	36,44,48,52,56,64	
3	37,45,49,53,57,65	
4	38,46,50,54,58,66	
5	39,47,51,55,59,67	
6	40,48,52,56,60	
7	41,49,53,57,61	
8	42,50,54,58,62	
9	43,51,55,59,63	
10	44,52,56,60,64	
11	45,53,57,61,65	
12	46,54,58,62,66	
13	47,55,59,63,67	
14	48,56,60,64,	34,42,46,50,54,62
15	49,57,61,65,	35,43,47,51,55,63
16	50,58,62,66,	36,44,48,52,56,64
17	51,59,63,67,	37,45,49,53,57,65
18	52,60,64	38,46,50,54,58,66
19	53,61,65	39,47,51,55,59,67
20	54,62,66	40,48,52,56,60
21	55,63,67	41,49,53,57,61
22	56,54	34,46,58
23	57,65	35,47,59
24	58,66	36,48,60
25	59,67	37,49,61
26	60	34,38,42,46,54
27	61	35,39,43,47,55
28	62	36,40,44,48,56
29	63	37,41,45,49,57
30	64	38,42,46,50,58
31	65	39,43,47,51,59
32	66	40,44,48,52,60
33	67	41,45,49,53,61

도면8

유전체		종래기술 (클럭주기)	본 발명 (클럭주기)	향상률
$GF(2^{271})$	감산비용	1084	482	56%
	실행결과	1126	524	53%