

	(19) 대한민국특허청(KR) (12) 공개특허공보(A)	(11) 공개번호 10-2012-0101243 (43) 공개일자 2012년09월13일
(51) 국제특허분류(Int. Cl.) G06F 19/00 (2011.01) G06F 17/11 (2006.01) H04L 9/28 (2006.01)		(71) 출원인 고려대학교 산학협력단 서울 성북구 안암동5가 1
(21) 출원번호	10-2011-0017994	(72) 발명자 홍석희
(22) 출원일자	2011년02월28일	서울특별시 도봉구 마들로 859-19, 한신아파트 103동 107호 (도봉동)
심사청구일자	2011년02월28일	조영인 서울특별시 강동구 올림픽로77길 46, 301호 (천호동) (뒷면에 계속)
		(74) 대리인 특허법인충현

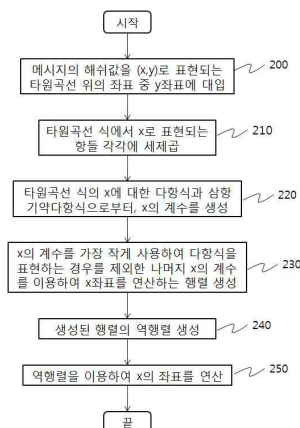
전체 청구항 수 : 총 11 항

(54) 발명의 명칭 GF (3^m)에서의 메시지 매핑 방법 및 이를 이용한 페어링 암호 시스템

(57) 요약

본 발명은 F_{3^m} 에서의 메시지 매핑 방법에 관한 것으로서, 메시지의 해쉬 값을 (x,y)로 표현되는 타원곡선 위의 좌표 중 y좌표에 대입하고, x좌표에 관한 방정식을 풀으므로써, 타원곡선 식을 만족하는 x좌표를 연산하되, 타원곡선 식에서 x로 표현되는 항들 각각에 세제곱을 하여 방정식을 푸는 것을 특징으로 하며, 메시지 매핑에 사용되는 행렬의 크기를 줄여 역행렬을 빠르게 계산할 수 있으며, 역행렬을 저장하는데 작은 메모리 공간이 필요하게 된다.

대표도 - 도2



(72) 발명자

장남수

서울특별시 마포구 월드컵북로27길 16, 2층 (성산동)

김창한

서울특별시 강남구 삼성로 212, 21동 507호 (대치동, 은마아파트)

이 발명을 지원한 국가연구개발사업

과제고유번호 NIPA-2011-C1090-1001-0004

부처명 지식경제부 정보통신산업진흥원

연구사업명 대학 IT연구센터 육성 지원사업

연구과제명 스마트 그리드 보안기술 개발

주관기관 고려대학교 산학협력단

연구기간 2011.01.01 ~ 2011.12.31

특허청구의 범위

청구항 1

메시지의 해쉬 값을 (x,y)로 표현되는 타원곡선 위의 좌표 중 y좌표에 대입하는 단계; 및

상기 해쉬 값을 상기 y좌표에 대입한 후, x좌표에 관한 방정식을 풀으로써, 상기 타원곡선 식을 만족하는 x좌표를 연산하는 단계를 포함하고,

상기 타원곡선 식에서 x로 표현되는 항들 각각에 세제곱을 하여 상기 방정식을 푸는 것을 특징으로 하는 F_{3^m} 에서의 메시지 매핑 방법.

청구항 2

제 1 항에 있어서,

상기 x좌표를 연산하는 단계는,

상기 타원곡선 식의 x에 대한 다항식과 삼항기약다항식으로부터 상기 x의 계수를 생성하는 단계;

상기 생성된 다항식의 계수 분포를 참조하여, x의 계수를 가장 작게 사용하여 다항식을 표현하는 경우를 제외한 나머지 x의 계수를 이용하여 상기 x좌표를 연산하는 행렬을 생성하는 단계; 및

상기 생성된 행렬로부터 상기 x의 좌표를 연산하는 단계를 더 포함하는 F_{3^m} 에서의 메시지 매핑 방법.

청구항 3

제 2 항에 있어서,

상기 x의 좌표를 연산하는 단계는,

상기 생성된 행렬의 역행렬을 생성하는 단계; 및

상기 생성된 역행렬을 이용하여 상기 x의 좌표를 연산하는 단계인 것을 특징으로 하는 F_{3^m} 에서의 메시지 매핑 방법.

청구항 4

제 2 항에 있어서,

상기 제외된 x의 계수들은 상기 연산된 x의 계수들로부터 덧셈 연산되는 것을 특징으로 하는 F_{3^m} 에서의 메시지 매핑 방법.

청구항 5

제 3 항에 있어서,

상기 생성된 역행렬은 저장되어, 입력되는 y좌표에 대응하는 x좌표의 연산에 사용되는 것을 특징으로 하는 F_{3^m} 에서의 메시지 매핑 방법.

청구항 6

임의의 메시지 스트링을 수신하는 메시지 수신부;

상기 수신된 메시지 스트링을 코드화하는 변환부;

상기 코드화된 메시지 스트링을 해싱하는 해싱부; 및

상기 해싱된 값을 타원곡선 위의 점으로 매핑하는 매핑부를 포함하고,

상기 매핑부는 상기 해싱된 값을 (x, y) 로 표현되는 타원곡선 위의 좌표 중 y 좌표에 대입하고, x 좌표에 관한 방정식을 풀으로써, 상기 타원곡선 식을 만족하는 x 좌표를 연산하되, 상기 타원곡선 식에서 x 로 표현되는 항들 각각에 세제곱을 하여 상기 방정식을 푸는 것을 특징으로 하는 F_{3^m} 에서의 페어링 암호 시스템.

청구항 7

제 6 항에 있어서,

상기 매핑부는,

상기 타원곡선 식의 x 에 대한 다항식과 삼항기약다항식으로부터 상기 x 의 계수를 생성하고, 상기 생성된 다항식의 계수 분포를 참조하여, x 의 계수를 가장 작게 사용하여 다항식을 표현하는 경우를 제외한 나머지 x 의 계수를 이용하여 상기 x 좌표를 연산하는 행렬을 생성한 다음, 상기 생성된 행렬로부터 상기 x 의 좌표를 연산하는 것을 특징으로 하는 F_{3^m} 에서의 페어링 암호 시스템.

청구항 8

제 7 항에 있어서,

상기 x 의 좌표를 연산하는 것은 상기 생성된 행렬의 역행렬을 생성하고, 상기 생성된 역행렬을 이용하여 상기 x 의 좌표를 연산하는 것을 특징으로 하는 F_{3^m} 에서의 페어링 암호 시스템.

청구항 9

제 7 항에 있어서,

상기 제외된 x 의 계수들은 상기 연산된 x 의 계수들로부터 덧셈 연산되는 것을 특징으로 하는 F_{3^m} 에서의 페어링 암호 시스템.

청구항 10

제 8 항에 있어서,

상기 생성된 역행렬은 저장되어, 입력되는 y 좌표에 대응하는 x 좌표의 연산에 사용되는 것을 특징으로 하는 F_{3^m} 에서의 페어링 암호 시스템.

청구항 11

제 1 항 내지 제 5 항 중에 어느 한 항의 방법을 컴퓨터에서 실행시키기 위한 프로그램을 기록한 컴퓨터로 읽을 수 있는 기록매체.

명세서

기술 분야

[0001]

본 발명은 F_{3^m} ($=GF(3^m)$)에서의 메시지 매핑 방법에 관한 것으로서, 더욱 상세하게는 메시지 매핑에 사용되는 행렬의 크기를 줄여 역행렬을 빠르게 계산할 수 있으며, 역행렬을 저장하기 위해 사용하는 메모리 역시 작은 공간을 사용하는 F_{3^m} 에서의 메시지 매핑 방법 및 이를 이용한 페어링 암호 시스템에 관한 것이다.

배경 기술

[0002]

유한체(finite field)는 원소의 개수가 유한한 체(field)를 뜻한다. 체는 0으로 나누는 것을 제외한 사칙연산(덧셈, 뺄셈, 곱셈, 나눗셈)을 자유롭게 할 수 있는 집합을 말한다. 따라서 자연수나 정수의 집합은 나눗셈의

결과가 원래의 집합에 항상 포함되는 것은 아니므로 체가 될 수 없고, 유리수, 실수는 체이기는 하지만 무한 집합이므로 유한체는 아니다.

[0003] 유한체는 표수(characteristic)에 따라서 그 특성이 결정된다. 표수는 유한체가 기본적으로 가지고 있는 모듈라 연산에서 법으로 사용되는 수로서 반드시 소수이다. 표수를 p 라고 할 때, 유한체의 원소의 개수는 표수의 거듭제곱의 꼴이 되며, $q=p^m$ ($m \geq 1$)개의 원소를 가지고 있는 유한체를 F_q 라고 나타낸다.

[0004] 가장 간단한 형태의 유한체는 $m=1$ 일때로서, $F_q=\{0,1,2,\dots,p-1\}$ 로 나타낼 수 있으며, 원소들 사이의 사칙 연산은 p 를 법으로 하는 모듈라 연산을 사용한다.

[0005] $m>1$ 인 경우는 원소를 정수로만 표현할 수 없을 때 다항식으로 표현한다. 유한체와 다항식의 집합 사이에는 다음과 같은 동치관계가 있다.

[0006]
$$F_{p^m} \simeq \frac{F_p[x]}{f(x)}$$

[0007] 여기서, $f(x)$ 는 F_p 의 원소를 계수로 갖는 m 차의 기약 다항식, $F_p[x]$ 는 계수가 F_p 의 원소인 모든 다항식의 무한 집합을 말하고, $\frac{F_p[x]}{f(x)}$ 는 다항식을 $f(x)$ 로 나눈 나머지를 대표원소로 하는 유한 집합이다. 그러면 위의 동치 관계에 따라서 F_{p^m} 의 원소는 $m-1$ 차 이하의 다항식과 일대일 관계에 있으며, 다항식으로 표현된 원소들 사이의 연산은 $f(x)$ 를 법으로 하는 모듈라 연산이다. 즉, m 차 이상의 다항식은 $f(x)$ 로 나눈 나머지를 유한체의 원소로 간주한다.

[0008] 한편, 페어링 암호 시스템(Pairing-based cryptosystem)은 최근 가장 활발하게 연구되는 분야로서 페어링 기반 암호 시스템은 타원곡선 위의 두 개의 점을 하나의 유한체 원소로 보내는 양방향 곱선형성(Bilinearity)을 가지는 페어링 함수를 기반으로 구성되는 암호 시스템이다.

[0009] 페어링 암호 연산 시간을 개선하려는 여러 노력에도 불구하고 페어링 암호의 구현은 페어링 기반 암호 시스템들을 구현하는데 있어 가장 큰 연산 부담을 주고 있다.

[0010] 페어링 함수는 소수 위수 q 를 갖는 타원곡선 점들의 그룹 G_1 과 G_2 에서 유한체 원소들의 그룹 G_T 로의 함수이다. P_1 과 P_2 를 각각 G_1 과 G_2 의 생성자라 할 때, G_1 , G_2 , 그리고 G_T 의 이산 대수 문제(DLP)를 풀기 어렵다고 가정한다. 그러면 아래의 두 조건을 만족하는 함수 $e : G_1 \times G_2 \rightarrow G_T$ 는 암호에 사용될 수 있는 곱선형 함수인 페어링 함수가 된다.

$$\text{곱선형성} : \forall g \in G_1, \forall h \in G_2, \forall \gamma, \delta \in \mathbb{Z}, e(g^\gamma, h^\delta) = e(g, h)^{\gamma\delta}$$

Non-degeneracy : $e(P_1, P_2) \neq 1$ 이고 G_T 의 생성자

[0011] 이진체 또는 소수체 위의 타원곡선만을 사용하는 타원곡선 암호와 달리 페어링 암호는 위수가 3인 삼진체 위의 타원곡선도 이용한다. 삼진체 위에서의 페어링 암호는 다른 유한체 위에서 보다 연산시간이 빠르기 때문에,

F_{3^m} ($=\text{GF}(3^m)$)에서의 효율적인 페어링 암호 연산 및 이에 사용되는 유한체 연산에 대한 연구가 주목을 받고 있다. 페어링 암호 기반의 암호 프로토콜 구현을 위하여 embedding degree가 상대적으로 작은 초특이 타원곡선이

매우 적합한 것으로 연구되었고, F_{3^m} 위의 페어링 연산을 위해서는 embedding degree가 6이며, $E^\pm : y^2 = x^3 - x \pm 1$ 로 정의된 초특이 타원곡선이 사용된다. 이 초특이 타원곡선을 이용하여 페어링 암호 연산을 수행하기 위해서는 먼저 주어진 메시지를 타원곡선 위의 점으로 작은 메모리를 이용하면서도 빠른 속도로 매핑하기 위한 과정이 필요하다.

[0013] F_{2^m} 위에서의 타원곡선 암호에서는 메시지의 해쉬(Hash) 값을 x 좌표에 대입하고 타원곡선 식을 만족하는 y 좌

표를 계산하는 것이 일반적이다. F_{3^m} 위에서는 세제곱이 선형연산으로 비교적 가벼운 연산이므로 메시지의 해쉬값을 y 좌표에 대입하고 타원곡선 식을 만족하는 x 좌표를 찾는 것이 연산량이 적다.

[0014] 문헌 1에서는 F_{3^m} 위에서의 삼차방정식을 풀으로써 해당 x 좌표를 찾을 수 있도록 하였다. 이 삼차방정식을 풀기 위해서는 F_3 의 원소로 이루어진 $(m-1) \times (m-1)$ 행렬의 역행렬이 필요하고 이 역행렬을 y^2-b 의 계수들과 행렬 곱셈을 수행하여 x 좌표를 찾을 수 있게 된다. 따라서 연산량은 역행렬을 구하는 과정과 이 역행렬의 행렬 곱셈을 하는 과정에서 결정된다. 일단 역행렬을 찾은 뒤에는 주어진 y 좌표에 따라 행렬 곱셈만으로 x 좌표를 계산할 수 있으므로 다른 메시지 스트링의 MapToPoint을 위해서 계산된 역행렬은 저장되는 것이 바람직하다.

[0015] 문헌 1을 참조하면, $\Gamma(x)=x^3-x$ 로 정의되는 매핑을 $\Gamma : F_{3^m} \rightarrow F_{3^m}$ 라 할 때, Γ 의 kernel이 F_3 이므로, Γ 의 랭크는 $m-1$ 이 된다. 문헌 1의 매핑방법은 알고리즘 1과 같다.

알고리즘 1. MapToPoint

- 1: $i=0$ 으로 세팅한다.
- 2: (i, M) 의 해쉬값 $h(i, M) = (y, t) \in F_{3^m} \times F_3$ 를 계산한다.
- 3: $c = y^2 - b$ 을 계산한다.
- 4: $\Gamma(x) = c$ 를 푼다.
- 5: 근이 없다면 i 를 증가시키고 step 2로 이동한다.
- 6: 근이 있다면 ${}^0x, {}^1x, {}^2x$ 중에서 t 를 이용하여 $({}^tx, y)$ 를 반환한다.

[0017] 문헌 1에서는 $\tau(c) = c_0, \tilde{c} = (c_0, \dots, c_{m-1})$ 으로 정의하고 마지막 스텝에서는 $\tau({}^0x) < \tau({}^1x) < \tau({}^2x)$ 을 바탕으로 근을 선택한다.

[0018] 정의 1. $a \in F_{3^m}$ 의 trace $a + a^3 + a^3 + \dots + a^{3^{m-1}}$ 를 $\text{Tr}(a)$ 라 한다.

[0019] 알고리즘 1의 스텝 4에서는 삼차방정식을 풀기 위해 $\text{Tr}(c)=0$ 을 확인함으로써 먼저 근이 있는지를 판단한다. 근이 있다면, Γ 의 랭크가 $m-1$ 이므로 $(m-1) \times (m-1)$ 크기의 행렬 A 를 생성하고 A 의 역행렬 A^{-1} 를 계산한다. 삼차방정식의 근은 $\tilde{x} = A^{-1}\tilde{c}$ 와 선택된 $x_0 \in F_3$ 으로 구한다.

[0020] BLS 스킴에서는 F_{2^m} 위에서의 동일한 방법을 사용하였기 때문에 MapToPoint의 연산량이 $O(m^3)$ 으로 비교적 컸으나 문헌 1에서는 MapToPoint의 연산량을 $O(m^2)$ 으로 줄였다. 그러나 타원곡선 위로 임의의 메시지를 매핑할 때, 메시지 매핑에 사용되는 행렬의 크기를 줄여 역행렬을 보다 더 빠르게 계산할 필요가 있다.

선행기술문헌

[0021] [문헌 1] P. Barreto and H.Y. Kim, "Fast hashing onto elliptic curves over fields of characteristic 3", Cryptology ePrint Archive, Report 2001/098.

[0022] [문헌 2] P. Barreto, "A note on efficient computation of cube roots in characteristic 3", Cryptology ePrint Archive: Report 2004/305, 2004.

[0023] [문헌 3] Young In Cho, Nam Su Chang, Chang Han Kim and Seokhie Hong, "Formulas for Cube roots in F_{3^m} with Reduced Hamming weight", submitted to Elsevier Discrete Applied Mathematics.

[0024] [문헌 4] K. Harrison, D. Page and N. Smart, "Software implementation of finite fields of characteristic three, for use in pairing-based cryptosystems", LMS Journal of Computation and Mathematics, vol. 5, pp. 181-193, 2002.

- [0025] [문헌 5] IEEE P1363. Standard specifications for public key cryptography, 2000.
http://grouper.ieee.org/groups/1363/index.html.

발명의 내용

해결하려는 과제

- [0026] 따라서, 본 발명이 해결하고자 하는 첫 번째 과제는 메시지 매핑에 사용되는 행렬의 크기를 줄여 역행렬을 빠르게 계산할 수 있는 F_{3^m} 에서의 메시지 매핑 방법을 제공하는 것이다.
- [0027] 본 발명이 해결하고자 하는 두 번째 과제는 메시지 매핑에 사용되는 행렬의 크기를 줄여 역행렬을 빠르게 계산할 수 있는 F_{3^m} 에서의 페어링 암호 시스템을 제공하는 것이다.
- [0028] 또한, 상기된 방법을 컴퓨터에서 실행시키기 위한 프로그램을 기록한 컴퓨터로 읽을 수 있는 기록 매체를 제공하는데 있다.

과제의 해결 수단

- [0029] 본 발명은 상기 첫 번째 과제를 달성하기 위하여, 메시지의 해쉬 값을 (x,y)로 표현되는 타원곡선 위의 좌표 중 y좌표에 대입하는 단계; 및 상기 해쉬 값을 상기 y좌표에 대입한 후, x좌표에 관한 방정식을 풀으로써, 상기 타원곡선 식을 만족하는 x좌표를 연산하는 단계를 포함하고, 상기 타원곡선 식에서 x로 표현되는 항들 각각에 세제곱을 하여 상기 방정식을 푸는 것을 특징으로 하는 F_{3^m} 에서의 메시지 매핑 방법을 제공한다.
- [0030] 본 발명의 일 실시예에 의하면, 상기 x좌표를 연산하는 단계는, 상기 타원곡선 식의 x에 대한 다항식과 삼항기약다항식으로부터 상기 x의 계수를 생성하는 단계; 상기 생성된 다항식의 계수 분포를 참조하여, x의 계수를 가장 작게 사용하여 다항식을 표현하는 경우를 제외한 나머지 x의 계수를 이용하여 상기 x좌표를 연산하는 행렬을 생성하는 단계; 및 상기 생성된 행렬로부터 상기 x의 좌표를 연산하는 단계를 더 포함할 수 있다.
- [0031] 또한, 상기 x의 좌표를 연산하는 단계는, 상기 생성된 행렬의 역행렬을 생성하는 단계; 및 상기 생성된 역행렬을 이용하여 상기 x의 좌표를 연산하는 단계를 포함할 수 있다.
- [0032] 또한, 상기 제외된 x의 계수들은 상기 연산된 x의 계수들로부터 덧셈 연산되는 것이 바람직하다.
- [0033] 본 발명의 다른 실시예에 의하면, 상기 생성된 역행렬은 저장되어, 입력되는 y좌표에 대응하는 x좌표의 연산에 사용되는 것이 바람직하다.
- [0034] 본 발명은 상기 두 번째 과제를 달성하기 위하여, 임의의 메시지 스트링을 수신하는 메시지 수신부; 상기 수신된 메시지 스트링을 코드화하는 변환부; 상기 코드화된 메시지 스트링을 해싱하는 해싱부; 및 상기 해싱된 값을 타원곡선 위의 점으로 매핑하는 매핑부를 포함하고, 상기 매핑부는 상기 해싱된 값을 (x,y)로 표현되는 타원곡선 위의 좌표 중 y좌표에 대입하고, x좌표에 관한 방정식을 풀으로써, 상기 타원곡선 식을 만족하는 x좌표를 연산하되, 상기 타원곡선 식에서 x로 표현되는 항들 각각에 세제곱을 하여 상기 방정식을 푸는 것을 특징으로 하는 F_{3^m} 에서의 페어링 암호 시스템을 제공한다.
- [0035] 상기 다른 기술적 과제를 해결하기 위하여, 본 발명은 상기된 F_{3^m} 에서의 메시지 매핑 방법을 컴퓨터에서 실행시키기 위한 프로그램을 기록한 컴퓨터로 읽을 수 있는 기록 매체를 제공한다.

발명의 효과

- [0036] 본 발명에 따르면, 메시지 매핑에 사용되는 행렬의 크기를 줄여 역행렬을 빠르게 계산할 수 있으며, 역행렬을 저장하는데 작은 메모리 공간이 필요하게 된다.

도면의 간단한 설명

[0037] 도 1은 본 발명의 바람직한 일 실시예에 따른 F_{3^m} 에서의 페어링 암호 시스템의 구성도이다.

도 2는 본 발명의 바람직한 일 실시예에 따른 F_{3^m} 에서의 메시지 매핑 방법의 흐름도이다.

도 3은 수학식 3의 차수에 따른 $\Gamma'(x)$ 의 도표이다.

도 3은 $m \equiv k \equiv 1(\text{mod } 3)$ 일 때의 차수에 따른 $\Gamma'(x)(=x-x^{1/3})$ 의 도표를 도시한 것이다.

도 4는 $m \equiv k \equiv 2(\text{mod } 3)$ 일 때의 차수에 따른 $\Gamma'(x)(=x-x^{1/3})$ 의 도표를 도시한 것이다.

발명을 실시하기 위한 구체적인 내용

[0038] 본 발명에 관한 구체적인 내용의 설명에 앞서 이해의 편의를 위해 본 발명이 해결하고자 하는 과제의 해결 방안의 개요 혹은 기술적 사상의 핵심을 우선 제시한다.

[0039] 본 발명의 일 실시예에 따른 F_{3^m} 에서의 메시지 매핑 방법은 메시지의 해쉬 값을 (x,y) 로 표현되는 타원곡선 위의 좌표 중 y 좌표에 대입하는 단계; 및 상기 해쉬 값을 상기 y 좌표에 대입한 후, x 좌표에 관한 방정식을 풀으로써, 상기 타원곡선 식을 만족하는 x 좌표를 연산하는 단계를 포함하고, 상기 타원곡선 식에서 x 로 표현되는 항들 각각에 세제곱을 하여 상기 방정식을 푸는 것을 특징으로 한다.

[0040] 이하 첨부된 도면을 참조하여 본 발명이 속하는 기술 분야에서 통상의 지식을 가진 자가 본 발명을 용이하게 실시할 수 있는 바람직한 실시 예를 상세히 설명한다. 그러나 이들 실시예는 본 발명을 보다 구체적으로 설명하기 위한 것으로, 본 발명의 범위가 이에 의하여 제한되지 않는다는 것은 당업계의 통상의 지식을 가진 자에게 자명할 것이다.

[0041] 본 발명이 해결하고자 하는 과제의 해결 방안을 명확하게 하기 위한 발명의 구성을 본 발명의 바람직한 실시예에 근거하여 첨부 도면을 참조하여 상세히 설명하되, 본 발명의 바람직한 실시 예에 대한 동작 원리를 상세하게 설명함에 있어 본 발명과 관련된 공지 기능 혹은 구성에 대한 구체적인 설명 그리고 그 이외의 제반 사항이 본 발명의 요지를 불필요하게 흐릴 수 있다고 판단되는 경우, 그 상세한 설명을 생략한다.

[0042] 덧붙여, 명세서 전체에서, 어떤 부분이 다른 부분과 '연결'되어 있다고 할때, 이는 '직접적으로 연결'되어 있는 경우뿐만 아니라, 그 중간에 다른 소자를 사이에 두고 '간접적으로 연결'되어 있는 경우도 포함한다. 또한 어떤 구성 요소를 '포함'한다는 것은, 특별히 반대되는 기재가 없는 한 다른 구성요소를 제외하는 것이 아니라, 다른 구성요소를 더 포함할 수 있는 것을 의미한다.

[0043] 도 1은 본 발명의 바람직한 일 실시예에 따른 F_{3^m} 에서의 페어링 암호 시스템의 구성도이다.

[0044] 도 1을 참조하면, 본 실시예에 따른 페어링 암호 시스템은 메시지 수신부(100), 변환부(110), 해싱부(120), 및 매핑부(130)로 구성된다.

[0045] 메시지 수신부(100)는 암호화하고자 하는 임의의 메시지 스트링을 수신한다.

[0046] 변환부(110)는 수신된 메시지 스트링을 타원곡선의 좌표값으로 입력할 수 있도록 숫자로 변환한다.

[0047] 해싱부(120)는 변환된 숫자를 해쉬 함수에 대입하여 해쉬값을 생성한다.

[0048] 매핑부(130)는 상기 해쉬값을 (x,y) 로 표현되는 타원곡선 위의 좌표 중 y 좌표에 대입하고, x 좌표에 관한 방정식을 풀으로써, 타원곡선 식을 만족하는 x 좌표를 연산한다. 이때, 타원곡선 식에서 x 로 표현되는 항들 각각에 세제곱을 하여 상기 방정식을 푸는 것이 바람직하다. 결과적으로 매핑부(130)는 메시지 수신부(100)가 수신한 메시지 스트링에 대응하는 타원곡선 위의 좌표값을 생성하게 된다.

[0049] 이하에서는 매핑부(130)가 페어링 연산을 위해 F_{3^m} 위에서 메시지 스트링을 G_1 의 원소로 매핑하는 과정을 MapToPoint이라 하고 이를 효율적으로 수행할 수 있는 방법을 설명하기로 한다.

[0050] 도 2는 본 발명의 바람직한 일 실시예에 따른 F_{3^m} 에서의 메시지 매핑 방법의 흐름도이다.

- [0051] 도 2를 참조하면, 본 실시예에 따른 메시지 매핑 방법은 도 1에 도시된 F_{3^m} 에서의 페어링 암호 시스템에서 시계열적으로 처리되는 단계들로 구성된다. 따라서, 이하 생략된 내용이라 하더라도 도 1에 도시된 페어링 암호 시스템에 관하여 이상에서 기술된 내용은 본 실시예에 따른 F_{3^m} 에서의 메시지 매핑 방법에도 적용된다.
- [0052] 200 단계에서 페어링 암호 시스템은 메시지의 해쉬값을 (x, y) 로 표현되는 타원곡선 위의 좌표 중 y 좌표에 대입한다.
- [0053] 210 단계에서 페어링 암호 시스템은 타원곡선 식에서 x 로 표현되는 항들 각각에 세제곱을 한다. 이때, y 로 표현되는 항과 상수항은 같이 계산된 후, 전체에 대하여 세제곱을 한다. 즉, 타원 곡선이 $y^2 = x^3 - x + b$ 로 표현되는 경우, $(y^2 - b)^{1/3} = (x^3 - x)^{1/3} = x - x^{1/3}$ 이 된다.
- [0054] 220 단계에서 페어링 암호 시스템은 타원곡선 식의 x 에 대한 다항식과 삼항기약다항식으로부터 x 의 계수를 생성한다.
- [0055] 230 단계에서 페어링 암호 시스템은 생성된 다항식의 계수 분포를 참조하여, x 의 계수를 가장 작게 사용하여 다항식을 표현하는 경우를 제외한 나머지 x 의 계수를 이용하여 상기 x 좌표를 연산하는 행렬을 생성한다.
- [0056] 240 단계에서 페어링 암호 시스템은 생성된 행렬의 역행렬을 생성한다.
- [0057] 250 단계에서 페어링 암호 시스템은 생성된 역행렬을 이용하여 수신된 y 좌표에 상응하는 x 의 좌표를 연산한다.
- [0058] 210 단계를 보다 상세하게 살펴보면 다음과 같다.
- [0059] 알고리즘 1의 방법과 달리 210 단계에서는 230 단계의 행렬 A 를 생성하기 위해서 x 의 세제곱근을 사용한다. 즉, 문헌 1에서는 $\Gamma(x) = (x^3 - x)$ 을 위해 x 의 세제곱을 사용했으나 210 단계에서는 $\Gamma'(x) = x - x^{1/3}$ 으로 정의되는 함수 $\Gamma' : F_{3^m} \rightarrow F_{3^m}$ 을 이용한다. F_{3^m} 에서는 $a \mapsto a^3$ 으로의 매핑이 permutation이므로 모든 원소는 유일한 세제곱근을 갖는다. 따라서 $(y^2 - b)^{1/3} = (x^3 - x)^{1/3} = x - x^{1/3}$ 을 만족하는 $d = (y^2 - b)^{1/3}$ 를 계산할 수 있으며 $d = \Gamma'(x)$ 을 풀으로써 x 좌표를 얻게 된다.
- [0060] $\Gamma'(x)$ 를 위해서 x 의 세제곱이 아닌 세제곱근을 이용하는 이유는 생성 삼항기약다항식이 $f(s) = s^m + \alpha s^k + \beta (\alpha, \beta \in F_3)$ 일 때, x 의 세제곱을 계산할 때는 m 과 k 의 값에 따라 x 의 계수가 서로 다르게 분포되지만 x 의 세제곱근을 계산할 때는 $m \equiv k \pmod{3}$ 인 경우에 같은 패턴으로 x 의 계수가 분포되기 때문이다. 이로 인해 $\Gamma'(x)$ 의 행렬 표현 A 를 보다 편리하게 축소시킬 수 있게 된다.
- [0061] 220 단계 내지 240 단계를 보다 상세하게 살펴보면 다음과 같다.
- [0062] 특히, 230 단계에서는 행렬 A 를 축소하는 과정을 거치게 된다.
- [0063] $f(s) = s^m + \alpha s^k + \beta (\alpha, \beta \in F_3)$ 가 $F_3[s]$ 의 생성 삼항기약다항식이라고 할 때, $F_{3^m} = F_3[s]/(f)$ 라 하자. F_{3^m} 에서의 세제곱근 연산을 살펴보면, 우선 임의의 자연수 u 에 대하여 $m = 3u$ 라 하자. ($m \equiv \pm 1 \pmod{3}$ 인 경우도 유사하게 적용할 수 있다.) 그러면 $x \in F_{3^m}$ 에 대하여 다음의 수학적 1과 같이 표현되고,

수학식 1

$$x = \sum_{i=0}^{3u-1} x_i s^i = \sum_{i=0}^{u-1} x_{3i} s^{3i} + s \sum_{i=0}^{u-1} x_{3i+1} s^{3i} + s^2 \sum_{i=0}^{u-1} x_{3i+2} s^{3i}$$

이고 이 식을 이용한 x의 세제곱 연산식은 다음의 수학식 2와 같다.

수학식 2

$$x^{1/3} = \sum_{i=0}^{u-1} x_i s^i = \sum_{i=0}^{u-1} x_{3i} s^i + s^{1/3} \sum_{i=0}^{u-1} x_{3i+1} s^i + s^{2/3} \sum_{i=0}^{u-1} x_{3i+2} s^i$$

$s^{1/3}$ 과 $s^{2/3}$ 가 사전계산 되었다면, 수학식 2를 이용하여 단지 두 번의 다항식 곱셈만으로 세제곱근을 계산할 수 있다. 이 다항식 곱셈을 할 때, $s^{1/3}$ 과 $s^{2/3}$ 의 다항식 표현이 작은 개수의 항을 갖는다면 보다 효율적으로 연산이 이루어진다. $m \equiv k \pmod{3}$ 인 경우 $s^{1/3}$ 과 $s^{2/3}$ 의 다항식 표현이 매우 작은 개수의 항을 가지므로 가장 효율적으로 세제곱 연산을 할 수 있다.

$m \equiv k \pmod{3}$ 인 형태의 삼항 기약다항식을 cube root-friendly trinomial이라 하며 실제로 대부분의 F_{3^m} 위의 페어링 암호 연산 연구에서 이 cube root-friendly trinomial을 고려하고 있다. 이를 이용할 경우, 같은 값에 대하여 세제곱 보다 세제곱근 연산이 더 적은 덧셈 연산을 필요로 하게 된다. 본 발명에서는 세제곱근 연산에 유리한 cube root-friendly trinomial을 대상으로 MapToPoint를 전개하나 이에 한정되지 아니하며, non-cube root friendly irreducible trinomial에 대해서도 MapToPoint 방법을 이용할 수 있다.

우선, 첫번째로 Cube root friendly irreducible trinomials($m \equiv k \pmod{3}$)를 이용하여 $\Gamma'(x)$ 의 행렬 표현 A를 축소시키는 방법과 이것의 역행렬을 구하는 방법을 살펴본다.

$m \equiv 1 \pmod{3}$ 또는 $m \equiv 2 \pmod{3}$ 인 각 경우에 대하여 축소된 $\Gamma'(x)$ 의 행렬 표현 $\hat{A}$ 를 구한다.

$m \equiv k \equiv 1 \pmod{3}$ 인 경우

$m \equiv 1 \pmod{3}$, $k \equiv 1 \pmod{3}$ 이라 하면 임의의 양수 u와 v에 대하여($u > v$) $m = 3u + 1$ 이고, $k = 3v + 1$ 라 하자.

F_{3^m} 에서 $x^{3u+1} + ax^{3v+1} + b = 0$ 이므로, 문헌 2에 의하여

$s^{1/3} = s^{2u+1} - a s^{u+v+1} + s^{2v+1}$ 과 $s^{2/3} = -\beta s^{u+1} - \alpha \beta s^{v+1}$ 를 구할 수 있다. 따라서 $\Gamma'(x)$ 는 다음의 수학식 3과 같이 표현된다.

수학식 3

$$\begin{aligned} \Gamma'(x) &= x - x^{1/3} = C - C_0 - s^{1/3} \cdot C_1 - s^{2/3} \cdot C_2 \\ &= C - C_0 - C_1(s^{2u+1} - \alpha s^{u+v+1} + s^{2v+1}) + C_2(\beta s^{u+1} + \alpha \beta s^{v+1}) \end{aligned}$$

이고, 이때, $C = \sum_{i=0}^{3u+1} x_i s^i$, $C_0 = \sum_{i=0}^u x_{3i} s^i$, $C_1 = \sum_{i=0}^{u-1} x_{3i+1} s^i$, $C_2 = \sum_{i=0}^{u-1} x_{3i+2} s^i$ 이다.

[0077] 예제를 통하여 $\hat{A}$ 를 생성하기 위한 사전계산과정을 설명하기로 한다.

[0078] 생성 삼항기약다항식이 $f(s)=s^{13}+s^4-1$ 이라 하면 $u=4$, $v=1$ 이고, $s^{1/3}=s^9-s^6+s^3$ 과 $s^{2/3}=s^5+s^2$ 를 구할 수 있다.

[0079] 표 1은 $f(s)=s^{13}+s^4-1$ 일 때, $d=\Gamma'(x)=x-x^{1/3}$ 의 다항식 표현을 나타낸 것이다.

표 1

차수	s^0	s^1	s^2	s^3	s^4	s^5	s^6	s^7	s^8	s^9	s^{10}	s^{11}	s^{12}
d	d_0	d_1	d_2	d_3	d_4	d_5	d_6	d_7	d_8	d_9	d_{10}	d_{11}	d_{12}
x	x_0	x_1	x_2	x_3	x_4	x_5	x_6	x_7	x_8	x_9	x_{10}	x_{11}	x_{12}
$-x^{1/3}$	$-x_0$	$-x_3$	$-x_6$	$-x_9$	$-x_{12}$	$-x_2$	$-x_5$	$-x_8$	$-x_{11}$	$-x_1$	$-x_4$	$-x_7$	$-x_{10}$

[0080]

[0081] 표 1에서 $d_0=0$ 이고, 각 d_1 , d_{10} , d_{11} , d_{12} 는 x 의 계수 중 두 원소의 뺄셈으로 이루어졌다. 이제 이들을 제거하여 행렬 A 를 축소하는 과정을 살펴본다.

[0082] x_1 은 d_1 , d_3 , d_6 , d_9 에 나타나고 $d_3'=d_3+d_1$, $d_6'=d_6-d_1$, 그리고 $d_9'=d_9+d_1$ 을 계산하고, (d_3, d_6, d_9) 를 (d_3', d_6', d_9') 으로 교체함으로써 표 1에서 x_1 을 삭제하고 d_1 을 제거할 수 있다. 다음으로 x_{10} 은 d_6 , d_9 , d_{10} , d_{12} 에 나타나고 x_{10} 을 삭제하기 위해 $d_6'=d_6'+d_{10}$, $d_9'=d_9'-d_{10}$, $d_{12}'=d_{12}+d_{10}$ 을 계산한다. 그리고 d_{12} 을 d_{12}' 으로 교체한다. 유사한 과정을 x_{11} 과 x_{12} 에 수행하면 다음의 수학적 4를 얻는다.

수학적 4

$$\begin{aligned} d_2' &= -x_6, & d_3' &= -x_5 - x_9, & d_4' &= -x_4 - x_8, \\ d_5' &= -x_2 + x_5 + x_7, & d_6' &= x_3 - x_4 - x_5 + x_6, & d_7' &= x_4 + x_7 - x_8, \\ d_8' &= -x_8, & d_9' &= -x_3 + x_4 + x_9. \end{aligned}$$

[0083]

[0084] 따라서 $(d_2', \dots, d_9')^T = \hat{A} \cdot (x_2, \dots, x_9)^T$ 을 만족하는 축소된 행렬 $\hat{A}$ 을 구할 수 있고 아래의 수학적 5와 같은 행렬 곱셈을 통하여 $(x_2, \dots, x_9)$ 을 구할 수 있다.

수학식 5

$$\widehat{A}^{-1} \cdot \widetilde{d}' = \begin{pmatrix} 0 & 0 & 0 & 2 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 2 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 2 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 2 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 2 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \end{pmatrix} \cdot \begin{pmatrix} d_2' \\ d_3' \\ d_4' \\ d_5' \\ d_6' \\ d_7' \\ d_8' \\ d_9' \end{pmatrix} = \begin{pmatrix} x_2 \\ x_3 \\ x_4 \\ x_5 \\ x_6 \\ x_7 \\ x_8 \\ x_9 \end{pmatrix}$$

[0085]

[0086] 마지막으로 $(x_1, x_{10}, x_{11}, x_{12})$ 을 복원하기 위하여 단지 $x_1=d_1+x_3$, $x_{10}=d_{10}+x_4$, $x_{11}=d_{11}+x_7$, $x_{12}=d_{12}+x_{10}$ 만을 계산하면 된다. 여기서 크기가 12×12 인 A 대신에 크기가 8×8 인 $\widehat{A}$ 을 사용함으로써 역행렬 연산 시간 및 저장 공간과 행렬 곱셈 연산량을 줄일 수 있게 된다.

[0087] 이제 일반적인 경우에 대하여 행렬 A 를 축소하는 방법을 살펴본다.

[0088] 도 3은 $m \equiv k \equiv 1 \pmod{3}$ 일 때의 차수에 따른 $\Gamma'(x)(=x-x^{1/3})$ 의 도표를 도시한 것으로, 수학식 3의 차수에 따른 $\Gamma'(x)$ 의 도표이다.

[0089] 도 3을 참조하면, Part 1과 Part 2의 각 d_i 들은 x 의 계수 중 두 원소의 뺄셈으로 이루어진다.

[0090] 다시 말해서 Part 1에는 $d_i=x_i-x_{3i}(i=0, \dots, v)$, Part 2에는 $d_{2u+v+j}=x_{2u+v+j}-x_{3v+3j+1}(j=1, \dots, u-v)$ 가 존재한다.

[0091] 앞서 보인 예제에서처럼 축소된 행렬 $\widehat{A}$ 을 생성하기 위한 사전 계산은 $\Gamma'(x)$ 의 행렬 표현인 A 에서 Part 1과 Part 2를 제거하는 것이다. 예를 들어 Part 1의 $d_1=x_1-x_3$ 이고 x_1 은 d_{2v+1} , d_{u+v+1} , 그리고 d_{2u+1} 에 존재한다.

[0092] $d_{2v+1}'=d_{2v+1}+d_1$, $d_{u+v+1}'=d_{u+v+1}-a \cdot d_1$, 그리고 $d_{2u+1}'=d_{2u+1}+d_1$ 을 계산함으로써, x_1 을 삭제하고 d_1 을 제거할 수 있다. 이러한 과정을 $(d_2, \dots, d_v, d_{2u+v+1}, \dots, d_{3u})$ 에 대하여 반복하고, $(d_2, \dots, d_v, d_{2u+v+1}, \dots, d_{3u})$ 을 $(d_2', \dots, d_v', d_{2u+v+1}', \dots, d_{3u}')$ 으로 교체한다. 알고리즘 2는 Part 1을 제거하는 과정을 알고리즘으로 표현한 것이다. Part 2 제거 또한 매우 유사하게 이루어진다.

알고리즘 2. $m \equiv k \equiv 1 \pmod{3}$ 일 때 사전계산(Part 1)

입력: $\tilde{d} = (d_0, \dots, d_{m-1})$
 출력: $\tilde{d}' = (d_{v+1}', \dots, d_{2u+v}')$
 1: $(d_0', \dots, d_{m-1}')$ 에 $\tilde{d}$ 를 대입한다.
 2: for $i = \lceil v/3 \rceil$ to 1 do
 3: if $3i \leq v$ then
 4: $d_i' = d_i + d_{3i}$
 5: end if
 6: end for
 7: for $i = 1$ to v do
 8: if $i \equiv 1 \pmod{3}$ then
 9: $d_{2u+i}' = d_{2u+i} + d_i'$
 10: $d_{u+v+i}' = d_{u+v+i} - \alpha \cdot d_i'$
 11: $d_{2v+i}' = d_{2v+i} + d_i'$
 12: else if $i \equiv 2 \pmod{3}$ then
 13: $d_{u-1+i}' = d_{u-1+i} - \beta \cdot d_i'$
 14: $d_{v-1+i}' = d_{v-1+i} - \alpha\beta \cdot d_i'$
 15: end if
 16: end for

[0093]

[0094]

위의 사전계산 전에 추가적인 연산이 필요할 수 있는데, 만약 $d_1 = x_1 - x_3$ 과 $d_3 = x_3 - x_9$ 가 모두 Part 1에 속해 있다면 x_1 을 삭제하는 과정에서 x_3 이 d_{2u+1} , d_{u+v+1} , 그리고 d_{2v+1} 에 더해지거나 빼지게 된다. 따라서 예상하지 않던 위치에 x_3 이 생기게 되고 그러면 d_3 을 제거하는 과정에서 x_3 의 삭제가 복잡하게 된다. 이런 상황을 방지하기 위하여 $d_1' = d_1 + d_3 = x_1 - x_9$ 을 계산하여 d_1 과 d_1' 을 교체한 후 사전계산을 시작한다. 사전 계산 전의 이러한 추가적인 연산은 알고리즘 2의 step 2 ~ 6에 해당한다.

[0095]

사전계산 후에 $\tilde{d}' = (d_{v+1}', \dots, d_{2u+v}')$ 과 이들의 해당 x 좌표 계수를 얻어 $\hat{A}$ 를 생성한다. 다음으로 $\hat{A}^{-1} \cdot \tilde{d}'$ 을 계산함으로써 $(x_{v+1}, \dots, x_{2u+v})$ 을 구한다. 마지막으로 알고리즘 3의 step 8 ~ 13에서처럼 $(x_1, \dots, x_v, x_{2u+v+1}, \dots, x_{3u})$ 을 복원함으로써 x 좌표의 전체 계수를 구한다. 이어서 $m \equiv k \equiv 2 \pmod{3}$ 인 경우에 대해서도 $d = \Gamma'(x)$ 의 근을 찾도록 한다. 이는 앞서 설명한 과정과 유사하다.

알고리즘 3. $m \equiv k \equiv 1 \pmod{3}$ 일 때 MapToPoint

입력: $M, y, \tilde{d}'$

출력: $\tilde{x} = (x_0, \dots, x_{m-1})$

1: $i=0$ 으로 세팅한다.

2: (i, M) 의 해쉬값 $h(i, M) = (y, t) \in F_{3^m} \times F_3$ 를 계산한다.

3: $d = (y^2 - b)^{1/3}$ 을 계산한다.

4: 사전계산 (알고리즘 2)

5: $\tilde{d}' = (d'_{v+1}, \dots, d'_{2u+v})$ 로 이루어진 행렬 $\hat{A}$ 를 생성한다.

6: $\hat{A}^{-1}$ 를 계산한다.

7: $\hat{A}^{-1} \cdot \tilde{d}' = (x_{v+1}, \dots, x_{2u+v})^T$ 를 계산한다.

8: for $i=1$ to v do

9: $x_i = d_i + x_{3i}$ (Part 1 복원)

10: end for

11: for $i=0$ to $u-v-2$ do

12: $x_{2u+v+1+i} = d_{2u+v+1+i} + x_{3v+1+3i}$ (Part 2 복원)

13: end for

14: 근이 없다면 i 를 증가시키고 step 2로 이동한다.

15: 근이 있다면 ${}^0x, {}^1x, {}^2x$ 중에서 t 를 이용하여 $({}^tx, y)$ 를 반환한다.

[0096]

[0097]

$m \equiv k \equiv 2 \pmod{3}$ 인 경우

[0098]

$m \equiv 2 \pmod{3}$, $k \equiv 2 \pmod{3}$ 이라 하면 임의의 양수 u 와 v 에 대하여($u > v$) $m=3u+2$ 이고, $k=3v+2$ 라 하자.

F_{3^m} 에서 $x^{3u+2} + \alpha x^{3v+2} + b = 0$ 이므로, 문헌 2에 의하여 $s^{1/3} = -\beta s^{u+1} - \alpha \beta s^{v+1}$ 과 $s^{2/3} = s^{2u+2} - \alpha s^{u+v+2} + s^{2v+2}$ 를 구할 수 있다. 따라서 $\Gamma'(x)$ 는 다음의 수학적식 6과 같이 표현된다.

수학적식 6

$$\begin{aligned} \Gamma'(x) &= x - x^{1/3} = C - C_0 - s^{1/3} \cdot C_1 - s^{2/3} \cdot C_2 \\ &= C - C_0 - C_1(\beta s^{u+1} + \alpha \beta s^{v+1}) + C_2(s^{2u+2} - \alpha s^{u+v+2} + s^{2v+2}) \end{aligned}$$

[0099]

$$\text{이고, 이때, } C = \sum_{i=0}^{3u+1} x_i s^i, \quad C_0 = \sum_{i=0}^u x_{3i} s^i, \quad C_1 = \sum_{i=0}^u x_{3i+1} s^i, \quad C_2 = \sum_{i=0}^{u-1} x_{3i+2} s^i \text{ 이다.}$$

[0100]

도 4는 $m \equiv k \equiv 2 \pmod{3}$ 일 때의 차수에 따른 $\Gamma'(x) (= x - x^{1/3})$ 의 도표를 도시한 것이다. 알고리즘 4는 Part1 제거 과정을 나타내고 알고리즘 5는 MapToPoint을 나타낸다.

[0101]

알고리즘 4. $m \equiv k \equiv 2 \pmod{3}$ 일 때 사전계산(Part 1)

입력: $\tilde{d} = (d_0, \dots, d_{m-1})$
 출력: $\tilde{d}' = (d'_{v+1}, \dots, d'_{2u+v+1})$
 1: $(d'_0, \dots, d'_{m-1})$ 에 $\tilde{d}$ 를 대입한다.
 2: for $i = \lceil v/3 \rceil$ to 1 do
 3: if $3i \leq v$ then
 4: $d'_i = d_i + d_{3i}$
 5: end if
 6: end for
 7: for $i = 1$ to v do
 8: if $i \equiv 1 \pmod{3}$ then
 9: $d'_{u+i} = d_{u+i} - \beta \cdot d'_i$
 10: $d'_{v+i} = d_{v+i} - \alpha \beta \cdot d'_i$
 11: else if $i \equiv 2 \pmod{3}$ then
 12: $d'_{2u+i} = d_{2u+i} + d'_i$
 13: $d'_{u+v+i} = d_{u+v+i} - \alpha \cdot d'_i$
 14: $d'_{2v+i} = d_{2v+i} + d'_i$
 15: end if
 16: end for

[0102]

알고리즘 5. $m \equiv k \equiv 2 \pmod{3}$ 일 때 MapToPoint

입력: $M, y, \tilde{d}'$
 출력: $\tilde{x} = (x_0, \dots, x_{m-1})$
 1: $i = 0$ 으로 세팅한다.
 2: (i, M) 의 해쉬값 $h(i, M) = (y, t) \in F_3^m \times F_3$ 를 계산한다.
 3: $d = (y^2 - b)^{1/3}$ 을 계산한다.
 4: 사전계산 (알고리즘 4)
 5: $\tilde{d}' = (d'_{v+1}, \dots, d'_{2u+v+1})$ 로 이루어진 행렬 $\hat{A}$ 를 생성한다.
 6: $\hat{A}^{-1}$ 를 계산한다.
 7: $\hat{A}^{-1} \cdot \tilde{d}' = (x_{v+1}, \dots, x_{2u+v+1})^T$ 를 계산한다.
 8: for $i = 1$ to v do
 9: $x_i = d_i + x_{3i}$ (Part 1 복원)
 10: end for
 11: for $i = 0$ to $u-v-1$ do
 12: $x_{2u+v+2+i} = d_{2u+v+2+i} + x_{3v+2+3i}$ (Part 2 복원)
 13: end for
 14: 근이 없다면 i 를 증가시키고 step 2로 이동한다.
 15: 근이 있다면 ${}^0x, {}^1x, {}^2x$ 중에서 t 를 이용하여 $({}^tx, y)$ 를 반환한다.

[0103]

[0104] 이상에서는 $m \equiv 1 \pmod{3}$ 또는 $m \equiv 2 \pmod{3}$ 인 각 경우에 대하여 축소된 $\Gamma'(x)$ 의 행렬 표현 $\hat{A}$ 를 구하는 과정에 대하여 살펴보았다.

[0105] 이하에서는 생성 삼항기약다항식이 cube root-friendly trinomial이 아닌 경우 $\Gamma'(x)$ 의 행렬 표현 A 를 축소시키는 방법을 대략적으로 설명한다.

[0106] 즉, 나머지 Irreducible Trinomials($m \neq k \pmod{3}$) 이용한 경우로서, $m \neq \pm k \pmod{3}$ 인 경우에 대하여 문헌 3에서는 Shifted polynomial basis(SPB)를 이용하여 cube root-friendly trinomial을 사용했을 때와 같은 연산량으로 세제곱근을 구할 수 있는 방법을 제안하였다. 문헌 3에 따르면 cube root-friendly trinomial을 사용했을 때와 같이 $s^{1/3}$ 과 $s^{2/3}$ 의 다항식 표현을 매우 작은 개수의 항으로 표현할 수 있으므로 이상에서 살펴본 행렬 생성과 축소 방법을 그대로 적용할 수 있다. $m \equiv -k \pmod{3}$ 인 경우에는 $s^{1/3}$ 과 $s^{2/3}$ 의 다항식 표현이 많은 개수

의 항을 가지므로 세제곱근 연산량이 비교적 크게 된다. 이는 행렬 A로부터 $\hat{A}$ 를 얻는 것이 매우 복잡하기 때문에 세제곱을 이용한 방법을 이용하는 것이 유리하다는 것을 의미한다. 세제곱을 이용한다고 하면 생성 삼항 기약다항식에 따라 사전계산이 달라져야 하는 단점이 있으나 마찬가지로 행렬을 축소할 수 있을 것이다.

[0107] 본 발명의 실시예들에서는 $\Gamma'(x)$ 의 행렬 표현을 축소하여 MapToPoint를 효율적으로 수행할 수 있는 방법을 제시하고 있다.

[0108] 이때, Part 1과 Part 2을 제거하기 위한 덧셈과 x 좌표의 일부를 복원하기 위한 덧셈은 매우 가벼운 연산이므로 MapToPoint의 연산량은 $\hat{A}$ 의 역행렬 계산과 $\hat{A}^{-1} \cdot \tilde{d}$ 의 행렬 곱셈에서 결정된다. 문헌 4에서는 F_{3^n} 에서의 유한체 연산을 구현하기 적합한 원소 표현 방법 두 가지를 제안하였다. 이중 Type II 표현은 F_{2^n} 에서의 원소 표현과 유사하며 각 $a_i \in F_3$ 을 $a_i = a_i^L - a_i^H$ 를 만족하는 (a_i^L, a_i^H) 두 비트로 유일하게 표현하였다. 여기서 a_i^L 과 a_i^H 는 동시에 1이 아니다. 본 발명의 실시예에서는 이 Type II 표현 방법을 이용하여 MapToPoint 방법을 구현하였다. $F_{3^{167}}$ 과 $F_{3^{193}}$ 위의 초특이 타원곡선 $E^- : y^2 = x^3 - x - 1$, $F_{3^{97}}$ 위의 초특이 타원곡선 $E^+ : y^2 = x^3 - x + 1$ 과 cube root-friendly trinomial : $f(s) = s^{97} - s^{16} + 1$, $f(s) = s^{167} - s^{71} + 1$, $f(s) = s^{193} - s^{64} + 1$ 을 이용하였다. 역행렬 연산은 일반적인 가우스 소거법보다 약 m/4배 빠른 LU 분해 방법을 사용하였다. 문헌 1의 결과와 비교했을 때, 역행렬은 3~5배 가량 빨리 연산 할 수 있으며 이 역행렬을 저장하기 위해서는 4/9(44%)의 메모리만이 필요하였다. 또한 역행렬이 주어졌을 때 MapToPoint를 위한 행렬 곱셈은 2~3배 정도 빠르다. 이들의 구현 결과는 표 3에 주어졌으며 각 연산 시간은 표 2의 연산이 수행되는 동안 측정되었다.

표 2

	Barreto 등	This work
Mat_Inv	A^{-1} 계산	Precomputation $\hat{A}^{-1}$ 계산
MapToPoint	$y^2 - b$ 계산 $A^{-1} \cdot \tilde{c}$ 계산	$y^2 - b$ 계산 $(y^2 - b)^{1/3}$ 계산 $\hat{A}^{-1} \cdot \tilde{d}'$ 계산 나머지 x 좌표의 계수 복원

[0109]

표 3

m	Barreto 등			This work		
	Mat_Inv (ms)	MapToPoint (μs)	Memory (byte)	Mat_Inv (ms)	MapToPoint (μs)	Memory (byte)
97	12.06	53.02	2304	2.58	19.65	1024
167	42.27	102.82	6889	13.79	58.46	3081
193	79.64	135.06	9216	22.58	63.00	4096

[0110]

[0111] 한편, 타원곡선위의 점을 전송할 때 대역폭 절약을 위해 (x,y)좌표 모두를 전송하는 대신 둘 중 하나만을 전송한 후 다른 하나는 수신자가 복원하는 방법을 Point Compression이라 한다. F_{2^n} 위의 타원곡선 암호에서는 수신자에게 160 비트 이상의 x 좌표와 1 비트의 y 좌표 부호가 주어지고 수신자는 타원곡선 이차식을 만족하는 y

좌표를 구하게 된다. F_P 위에서는 x 좌표와 1 비트의 $y'=y(\bmod 2)$ 가 주어진다. F_{2^n} 과 F_P 에서의 point compression 방법은 문헌 5에 자세하게 설명되어있다. 본 발명의 실시예에 따른 MapToPoint 방법을 F_{3^n} 에서의 point compression에 효과적으로 적용할 수 있는데 이때는 y 좌표와 x_0 및 복원할 x 좌표의 부호까지 모두 3 비트가 추가로 주어져야 하며, point decompression시에 적합한 x 좌표를 연산하는데 있어 가장 빠른 알고리즘을 제공할 수 있다.

[0112] 본 발명에서는 임의의 메시지 스트링을 F_{3^n} 위의 타원곡선 점으로 보다 효율적으로 매핑하는 방법을 제시하고 있다. 덧셈 연산이 다른 연산에 비해 훨씬 가벼우므로 그 횟수를 늘려 MapToPoint을 위한 행렬의 크기를 줄였다.

[0113] 구체적으로 문헌 1에서 사용된 행렬 크기를 $(m-1) \times (m-1)$ 에서 $\lfloor (2/3)(m-1) \rfloor \times \lfloor (2/3)(m-1) \rfloor$ 으로 줄임으로써, 역행렬을 3 ~ 5배 빠르게 계산할 수 있으며 역행렬의 크기가 기존에 비해 44%로 작아졌으므로 이를 저장하기 위한 메모리 또한 기존의 44%만 필요하게 된다. 또한 역행렬과 y^2-b 의 계수의 행렬 곱셈 연산량도 줄어들게 되므로 결과적으로 역행렬이 주어졌을 때 MapToPoint를 2 ~ 3배 정도 빠르게 할 수 있다. 따라서 본 발명에 따른 MapToPoint 방법은 유비쿼터스 환경 등의 제한된 저장 공간과 컴퓨팅 파워를 갖는 장비에 효과적으로 적용가능하다.

[0114] 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 media), 및 롬(ROM), 램(RAM), 플래시 메모리 등과 같은 프로그램 명령을 저장하고 수행하도록 특별히 구성된 하드웨어 장치가 포함된다. 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 동작을 수행하기 위해 하나 이상의 소프트웨어 모듈로서 작동하도록 구성될 수 있으며, 그 역도 마찬가지이다.

[0115] 이상과 같이 본 발명에서는 구체적인 구성 요소 등과 같은 특정 사항들과 한정된 실시예 및 도면에 의해 설명되었으나 이는 본 발명의 보다 전반적인 이해를 돕기 위해서 제공된 것일 뿐, 본 발명은 상기의 실시예에 한정되는 것은 아니며, 본 발명이 속하는 분야에서 통상적인 지식을 가진 자라면 이러한 기재로부터 다양한 수정 및 변형이 가능하다.

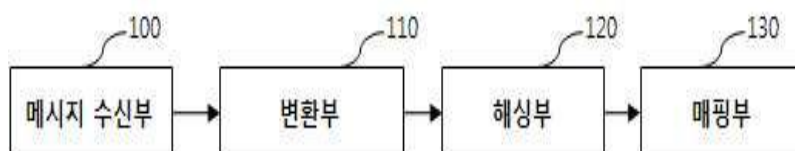
[0116] 따라서, 본 발명의 사상은 설명된 실시예에 국한되어 정해져서는 아니되며, 후술하는 특허청구범위뿐 아니라 이 특허청구범위와 균등하거나 등가적 변형이 있는 모든 것들은 본 발명 사상의 범주에 속한다고 할 것이다.

산업상 이용가능성

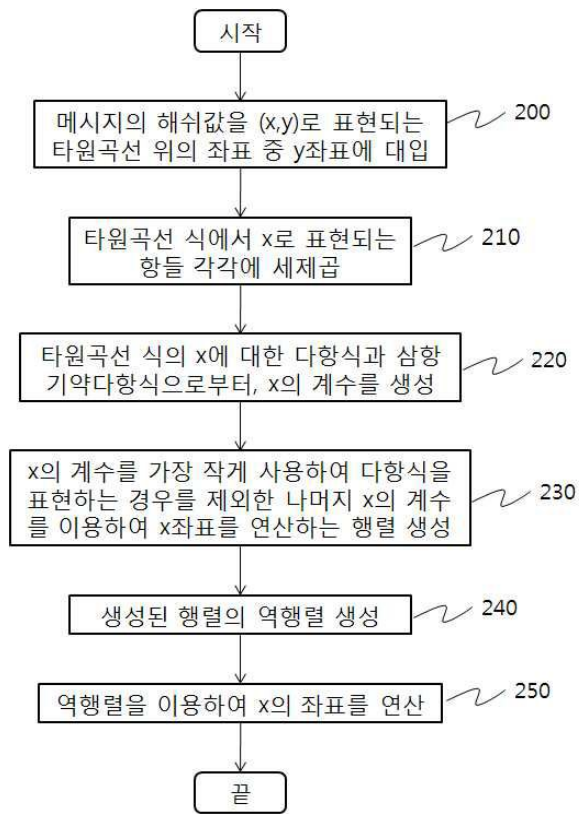
[0117] 공개키 암호 시스템, 페어링 암호 시스템 등에 사용될 수 있다.

도면

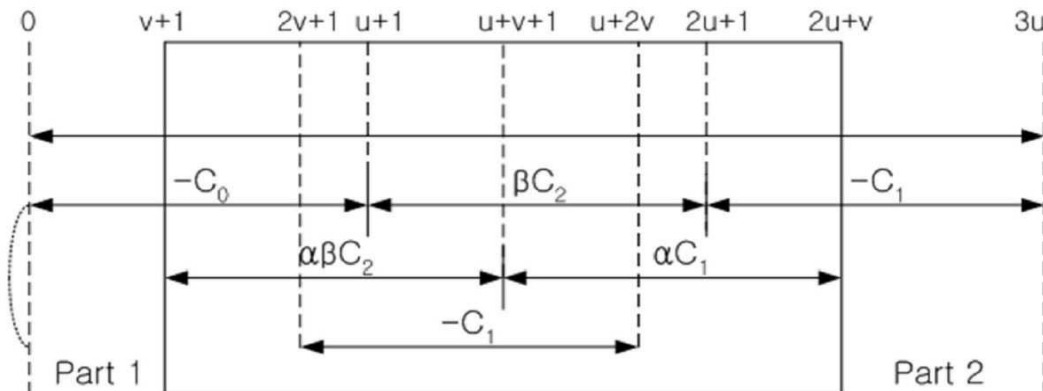
도면1



도면2



도면3



도면4

