(19)대한민국특허청(KR)
(12) 등록특허공보(B1)

(51) 。Int. Cl.
G06F 17/10 (2006.01)

(45) 공고일자 2006년08월01일
(11) 등록번호 10-0606435
(24) 등록일자 2006년07월21일

(21) 출원번호 10-2005-0010772
(22) 출원일자 2005년02월04일

(65) 공개번호
(43) 공개일자

(73) 특허권자 고려대학교 산학협력단
서울 성북구 안암동5가1 고려대학교 내

(72) 발명자 김소선
서울 성북구 안암동5가 103-149 208호

장남수
서울 마포구 성산2동 200-84

김창한
서울 서초구 반포동 20-9 주공A 328-402호

임종인
서울 종로구 내수동 95 삼성파크팰리스 101-901

(74) 대리인 현종철

(56) 선행기술조사문헌
KR100368204 B1
KR1020020086005 A
KR100416291 B1
KR1020030083236 A
* 심사관에 의하여 인용된 문헌

심사관 : 김수섭

(54) 유한체에서의 역원 연산 장치 및 방법

요약

유한체에서의 역원 연산 장치 및 방법이 개시된다. 제1연산부는 유한체 GF(2^m)의 원소인 제1다항식 A(x)와 소정의 m차 기약다항식인 제2다항식 G(x)를 입력으로 하고, 제1다항식과 제2다항식을 각각 소정의 제1내부변수 및 제2내부변수의 초기값으로 설정한 후 교환연산제어변수, 제1내부변수의 최하위비트값, 및 제1내부변수와 제2내부변수의 최하위비트의 인접비트값의 동일여부에 따라 제1내부변수가 1이 될 때까지 제1내부변수와 제2내부변수에 대한 교환연산, 쉬프트연산, 및 덧셈연산을 선택적으로 반복수행한다. 제2연산부는 교환연산제어변수, 제1내부변수의 최하위비트값, 및 제1내부변수와 제2내부변수의 최하위비트의 인접비트값의 동일여부에 따라 제1내부변수가 1이 될 때까지 소정의 초기값이 설정된 제3내부변수 및 제4내부변수에 대한 교환연산, 쉬프트연산, 및 덧셈연산을 선택적으로 반복수행하며, 상기 덧셈연산을 수행

하는 동안 제2다항식에 대해 G(x)/x 및 (G+Gx)/x 연산을 수행하여 얻어진 G(x), G(x)/x, 및 (G+Gx)/x 중에서 선택된 제3다항식과 제4내부변수의 값을 합산하여 모듈러 감산 연산을 수행한다. 제어부는 제1내부변수가 1이 되면 제3내부변수를 제1다항식의 역원으로 출력한다.

대표도

도 1

명세서

도면의 간단한 설명

도 1은 본 발명에 따른 유한체에서의 역원 연산 방법에 대한 일 실시예의 수행과정을 도시한 흐름도, 그리고,

도 2는 본 발명에 따른 유한체에서 역원 연산 장치에 대한 바람직한 실시예의 상세한 구성을 도시한 블록도이다.

발명의 상세한 설명

발명의 목적

발명이 속하는 기술 및 그 분야의 종래기술

본 발명은 유한체에서의 역원 연산 장치 및 방법에 관한 것으로, 보다 상세하게는, 경량화된 암호 시스템에 적합한 유한체에서의 역원 연산 장치 및 방법에 관한 것이다.

최근 유비쿼터스 환경의 정보보호에 대한 관심이 증가되고 경량화된 암호 시스템의 설계가 요구되고 있다. 이에 따라 동일한 안전도를 유지하면서 다른 공개키 암호 시스템보다 적은 길이의 키를 가진 타원곡선을 이용한 암호 시스템이 주목받고 있다. 타원곡선 암호 시스템의 연산은 기본적으로 유한체 연산을 포함하고 있으며 구현이 용이하다. 처리능력, 에너지 가용 능력이 제한된 환경인 스마트 카드, 휴대용 단말기 등의 디바이스에 적용이 가능하다.

유한체 GF(2^m)는 암호이론, 에러정정코드와 같은 어플리케이션에 많이 적용된다. 따라서 유한체 GF(2^m)에서 정의된 덧셈, 뺄셈, 곱셈 및 역원 연산의 고속화는 매우 중요하다. 공개키 암호 시스템의 수행시간은 주로 곱셈 및 역원 연산의 수행시간을 기반으로 하며, 이중 역원 연산이 곱셈에 비해 시간 복잡도가 크므로 많은 연구의 대상이 되고 있다.

유한체의 GF(2^m)의 연산은 유한체의 기저와 밀접한 연관을 가지며 이는 정규 기저(Normal Basis)와 다항식 기저(Polynomial Basis)로 나뉜다. 정규 기저인 경우 원소의 제곱 연산이 쉬프트 연산으로 수행가능하므로, GF(2^m)의 임의의 원소 A(x)의 역원은 페르마 정리에 의해 $A(x)^{-1} = A(x)^{2^m-2}$이다. 그러나 확장체의 차수가 홀수이거나 지수승 연산의 효율성이 지는 경우에는 적용할 수 없다. 다항식 기저를 사용하는 유한체의 역원 연산은 주로 Extended Euclid's Algorithm (EEA)와 Extended Binary GCD Algorithm(EBGA), Almost Inversion Algorithm(AIA)을 기반으로 한다. EEA와 EBGA는 모듈러 연산을 포함하며, 입력된 값에 따라 필요한 연산이 바뀌고 역원 연산의 반복 수행횟수가 고정되지 않아 하드웨어 구현에 어려움이 있다. AIA는 $R(x) \times A(x) \equiv x^k$을 만족시키는 R(x)와 k를 계산하는 수정된 EBGA로, Almost Montgomery Inverse인 Phase I과 모듈러 감산 연산을 하는 Phase II로 구성되어 있다.

J. H. Guo와 C. L. Wang은 "유한체 GF(2^m)에서 역원과 나눗셈을 위한 하드웨어 효율적인 시스톨릭 구조"와 "유한체 GF(2^m)에서 역원과 나눗셈을 위한 유클리드 알고리즘의 시스톨릭 어레이 구현"이라는 제목의 논문을 통해 수행횟수가 2m번으로 고정된 EEA을 제안한 바 있고, 시스톨릭 어레이(Systolic Array) 하드웨어 구조를 제시하였다. 한편, Y. Watanabe, N. Takagi, 그리고, K. Takagi는 "확장된 이진 GCD 알고리즘에 기반한 유한체 GF(2^m)에서 나눗셈을 위한 VLSI 알고리즘"이라는 제목의 논문을 통해 EBGA에서 비교 연산을 효율적으로 수행하는 역원 알고리즘을 제시하였다. 또한, C. H. Wu, C. M. Wu, M. D. Shieh, 그리고, Y. T. Hwang은 "보안통신을 위한 유한체 GF(2^m)에 대한 면적 효율적인 시스톨릭 나눗셈 회로"라는 제목의 논문을 통해, 그리고, Z. Yan과 D. V. Sarwate는 "유한 필드 역원 및 나눗셈을 위한 시스톨릭 구조"와

"유한체 GF(2^m)에서 역원 및 나눗셈을 위한 새로운 시스톨릭 구조"라는 논문을 통해 수행횟수가 2m번으로 고정된 EBGA을 제안하고, 시스톨릭 어레이 구조를 제시하였다. 그러나 이러한 구조는 유한체 차수 m이 커지는 경우 많은 하드웨어 공간을 요구하므로 공간이 제약되어 있는 환경에는 적합하지 않다.

이하에서는 유한체 GF(2^m)에서 기존의 역원 알고리즘을 살펴본다.

유한체 GF(2^m)은 다음과 같은 GF(2)에서의 m차 기약다항식에 의해 생성된다.

수학식 1

$$G(x)=x^m+\sum_{i=0}^{m-1} g_i x^i, \quad g_i \in GF(2)$$

다항식 기저에 의해 GF(2^m)의 임의의 원소 A(x)는 다음의 수학식과 같이 표현된다.

수학식 2

$$A(x)=a_{m-1}x^{m-1}+a_{m-2}x^{m-2}+\cdots+a_1x+a_0, \quad a_i \in GF(2)$$

이 때, 수학식 2를 벡터로 표현하면 다음과 같다.

수학식 3

$$A=(a_{m-1},a_{m-2},\cdots,a_1,a_0), a_i \in \{0,1\}$$

유한체 GF(2^m)에서의 다항식 덧셈 연산은 XOR 연산과 동일하며, 다항식의 곱셈인 $A(x)\cdot B(x)=A(x)\times B(x) \bmod G(x)$과 나눗셈인 $A(x)\div B(x)=A(x)\times B(x)^{-1} \bmod G(x)$은 기약다항식에 의한 모듈러 감산 연산이 이루어진다. 다항식의 나눗셈에서 A(x)=1인 경우 $B(x)^{-1}$을 B(x)의 역원이라 한다.

다항식 기저를 사용하는 유한체의 역원 연산은 대부분 Euclid's Algorithm을 기본으로 한다. 유한체 GF(2^m)의 임의의 원소 A(x)의 역원은 다음식을 만족하는 S(x)와 T(x)가 유일하게 존재함을 이용하여 계산한다.

수학식 4

$$S(x)\times A(x)+T(x)\times G(x)=1$$

수학식 4에서 $S(x)\times A(x)\equiv \bmod G(x)$이고, $S(x)\equiv A(x)^{-1} \bmod G(x)$이므로, S(x)는 A(x)의 역원이다.

한편, 유한체 GF(2^m)의 확장된 이진 GCD(Extended Binary GCD) 알고리즘(EBGA)은 EEA와 동일하게 다음의 수학식들 사이의 연산을 기본으로 한다.

수학식 5

$$R(x)A(x)\equiv U(x) \bmod G(x)$$

수학식 6

$$S(x)A(x)\equiv V(x) \bmod G(x)$$

EBGA는 이진 연산에 적합한 알고리즘이므로, 다음 성질을 이용하여 다항식의 역원 알고리즘으로 수정한다.

1. $U(x) \mid x$이고, $V(x) \mid x$이면, $GCD(U(x),V(x))=x\times GCD(U(X)/x,V(x)/x)$이다.

2. $U(x) \mid x$이고, $V(x) \nmid x$이면, $GCD(U(x),V(x))=x\times GCD(U(X)/x,V(x)/x)$이다.

3. $U(x) \nmid x$이고, $V(x) \nmid x$이면, $GCD(U(x),V(x))=x \times GCD(U(X)/x,V(x)/x)$이다.

위와 같이 수정된 EBGA는 다음과 같이 표현된다.

입력: $A(x) \in GF(2^m)$, $G(x)$

출력: $A(x)^{-1} \mod G(x)$

1.U=A(x), V=G(x), R=1, S=0

2. While V≠0 do the following:

2.1 While u_0=0 do:

2.1.1 U=U/x, R=R/x mod G(x)

2.2 End While.

2.3 While v_o=0 do:

2.3.1 V=V/x, S=S/x mod G(x)

2.4 End While.

2.5 If deg(U) > deg(V) then

2.5.1 U=U+ V, R=R+ S

2.6 Else

2.6.1 V=U+ V, S=R+ S

2.7 End If.

3. End While.

4. Return (S)

위의 수정된 EBGA에서 V=0이면 U=1로 반복문은 종료되고, R(x)A(x)≡1 mod G(x)를 의미한다. S는 $S(x)=A(x)^{-1} \mod G(x)$로 A(x)의 역원이다. 수정된 EBGA의 연산은 쉬프트 연산과 XOR 연산, 비교 연산으로 구성된다. 수정된 EBGA의 2.2.1단계에서 U=U/x은 오른쪽 쉬프트 연산을 이용하여 U=U≫1로 표현된다. 따라서 고정된 쉬프트 연산을 사용하므로, 쉬프트 연산이 고정되어 있지 않은 EEA보다 EBGA가 하드웨어 구현에 더 적합하다.

수정된 EBGA에서 U, V와 R, S는 수학식 4에 의해 밀접한 관계를 가진다. 따라서 U와 V의 연산을 수행하면 R과 S도 동일한 연산을 수행해야 한다. 수정된 EBGA의 2.2.1단계와 같이 $R(x) \mid x$인 경우 쉬프트 연산을 수행한다. $R(x) \nmid x$인 경우 (R(x)+ G(x))| x이므로, EEA와 다르게 모듈러 연산이 필요하다.

수정된 EBGA의 입력 G(x)의 차수는 항상 m이며, 입력 A(x)의 차수는 최대 (m-1)이다. 수정된 EBGA는 수행횟수가 고정되어 있지 않으며 최악의 경우 2m번 반복문을 실행한다. 또한, 하드웨어 구현시 비교 연산을 하기 위해 입력값의 차수를 가진 레지스터가 추가된다. 나아가, 이상과 같은 종래의 유한체 GF(2^m)에서의 EBGA는 입력된 값에 따라 알고리즘 수행 횟수가 결정되어 하드웨어로의 구현이 어려운 문제가 있다.

한편, C. H. Wu, C. M. Wu, M. D. Shieh, 그리고, Y. T. Hwang은 "보안통신을 위한 유한체 GF(2^m)에 대한 면적 효율적인 시스톨릭 나눗셈 회로"라는 제목의 논문을 통해, 그리고, Z. Yan과 D. V. Sarwate는 "유한 필드 역원 및 나눗셈을 위한 시스톨릭 구조"라는 제목의 논문을 통해 수행횟수가 고정된 EBGA를 제안하고, 시스톨릭 어레이 구조를 제시하였다. 한편, Y. Watanabe, N. Takagi, 그리고, K. Takagi는 "확장된 이진 GCD 알고리즘에 기반한 유한체 GF(2^m)에서 나눗셈을 위한 VLSI 알고리즘"이라는 제목의 논문을 통해 VLSI 구현시 비교 연산에 따른 알고리즘 지연시간을 줄이기 위한 방법을 제안하였다. 그러나, 이러한 방법들 역시 m이 커지는 경우 하드웨어 면적이 증가되므로 경량화가 요구하는 환경에는 적합하지 않다는 문제가 있다.

발명이 이루고자 하는 기술적 과제

본 발명이 이루고자 하는 기술적 과제는 다항식 기저를 바탕으로 종래의 알고리즘에 비해 적은 수행횟수를 가지는 경량화된 암호 시스템에 적합한 유한체에서의 역원 연산 장치 및 방법을 제공하는 데 있다.

본 발명이 이루고자 하는 다른 기술적 과제는 다항식 기저를 바탕으로 종래의 알고리즘에 비해 적은 수행횟수를 가지는 경량화된 암호 시스템에 적합한 유한체에서의 역원 연산 방법을 컴퓨터에서 실행시키기 위한 프로그램을 기록한 컴퓨터로 읽을 수 있는 기록매체를 제공하는 데 있다.

발명의 구성 및 작용

상기의 기술적 과제를 달성하기 위한, 본 발명에 따른 유한체에서의 역원 연산 장치는, 유한체 GF(2^m)의 원소인 제1다항식 A(x)와 소정의 m차 기약다항식인 제2다항식 G(x)를 입력으로 하고, 상기 제1다항식과 상기 제2다항식을 각각 소정의 제1내부변수 및 제2내부변수의 초기값으로 설정한 후 교환연산제어변수, 상기 제1내부변수의 최하위비트값, 및 상기 제1내부변수와 상기 제2내부변수의 최하위비트의 인접비트값의 동일여부에 따라 상기 제1내부변수가 1이 될 때까지 상기 제1내부변수와 상기 제2내부변수에 대한 교환연산, 쉬프트연산, 및 덧셈연산을 선택적으로 반복수행하는 제1연산부; 상기 교환연산제어변수, 상기 제1내부변수의 최하위비트값, 및 상기 제1내부변수와 상기 제2내부변수의 최하위비트의 인접비트값의 동일여부에 따라 상기 제1내부변수가 1이 될 때까지 소정의 초기값이 설정된 제3내부변수 및 제4내부변수에 대한 교환연산, 쉬프트연산, 및 덧셈연산을 선택적으로 반복수행하며, 상기 덧셈연산을 수행하는 동안 상기 제2다항식에 대해 G(x)/x 및 (G+ Gx)/x 연산을 수행하여 얻어진 G(x), G(x)/x, 및 (G+ Gx)/x 중에서 선택된 제3다항식과 상기 제4내부변수의 값을 합산하여 모듈러 감산 연산을 수행하는 제2연산부; 상기 교환연산제어변수, 상기 제1내부변수의 최하위비트값, 및 상기 제1내부변수와 상기 제2내부변수의 최하위비트의 인접비트값의 동일여부에 따라 상기 제1연산부 및 상기 제2연산부의 동작을 제어하며, 상기 제1내부변수가 1이 되면 상기 제3내부변수를 상기 제1다항식의 역원으로 출력하는 제어부; 및 상기 교환연산제어변수가 저장되는 제어값저장부;를 구비한다.

상기의 다른 기술적 과제를 달성하기 위한, 본 발명에 따른 유한체에서의 역원 연산 방법은, 유한체 GF(2^m)의 원소인 제1다항식 A(x)와 소정의 m차 기약다항식인 제2다항식 G(x)를 각각 소정의 제1내부변수 및 제2내부변수의 초기값으로 설정하는 단계; 소정의 제3내부변수 및 제4내부변수의 초기값을 각각 1과 0으로 설정하는 단계; 소정의 교환연산제어변수가 0이면, 상기 제1내부변수를 하위비트 방향으로 1비트 이동시키고, 상기 제3내부변수를 하위비트 방향으로 1비트 이동시킨 후 상기 제2다항식과의 모듈러 연산을 수행하는 단계; 상기 교환연산제어변수가 0이 아닌 경우에 상기 제1내부변수와 상기 제2내부변수의 최하위비트의 인접비트값이 동일하면, 상기 제1내부변수를 상기 제1내부변수와 상기 제2내부변수의 합산값으로 변경하고, 상기 제3내부변수를 상기 제3내부변수와 상기 제4내부변수의 합산값으로 변경하는 단계; 상기 교환연산제어변수가 0이 아닌 경우에 상기 제1내부변수와 상기 제2내부변수의 최하위비트의 인접비트값이 상이하면, 상기 제1내부변수를 상기 제1내부변수, 상기 제2내부변수, 및 상기 제2내부변수와 x의 곱의 합산값으로 변경하고, 상기 제3내부변수를 상기 제3내부변수, 상기 제4내부변수, 및 상기 제4내부변수와 x의 곱의 합산값으로 변경하는 단계; 상기 제1내부변수를 하위비트 방향으로 2비트 이동시키고, 상기 제3내부변수를 하위비트방향으로 2비트 이동시킨 후 상기 제2다항식과의 모듈러 연산을 수행하며, 상기 교환연산제어변수의 값을 변경하는 단계; 및 상기 제1내부변수가 1이면 상기 제3내부변수를 상기 제1다항식의 역원으로 출력하는 단계;를 갖는다.

이에 의해, 종래의 역원 연산 알고리즘에 비해 1.5배 빠르게 연산을 수행할 수 있고, 종래의 역원 연산 알고리즘의 수행횟수의 65%정도로 수행횟수를 줄일 수 있다.

이하에서 첨부된 도면들을 참조하여 본 발명에 따른 유한체에서의 역원 연산 장치 및 방법의 바람직한 실시예에 대해 상세하게 설명한다.

도 1은 본 발명에 따른 유한체에서의 역원 연산 방법에 대한 일 실시예의 수행과정을 도시한 흐름도이다.

도 1을 참조하면, 본 발명에 따른 유한체에서의 역원 연산 장치에 다항식 A(x)와 G(x)가 입력으로 주어진다. 이 때, A(x)는 유한체 GF(2^m)의 원소이다. 또한, G(x)의 차수는 항상 m이고, A(x)의 차수는 C. H. Wu, C. M. Wu, M. D. Shieh, 그리고, Y. T. Hwang가 "보안통신을 위한 유한체 GF(2^m)에 대한 면적 효율적인 시스톨릭 나눗셈 회로"라는 제목의 논문에서 제안한 바와 같이 최대 (m-1)이다. 한편, 본 발명에 따른 유한체에서의 역원 연산 장치의 출력은 $A(x)^{-1} \bmod G(x)$로 정의된다. 다항식 A(x)와 G(x)가 입력되면, 변수 U, V, R, 및 S의 값으로 각각 A(x), G(x), 1, 및 0이 설정된다(S100).

다음으로, U의 값을 확인한다(S110). 만약, U=1이면, GCD(A(x),G(x))=1이므로 $R(x)A(x) \equiv 1 \bmod G(x)$이다. 이 때, 수학식 5와 6에 의해 역원이 결정된다. 따라서, $R(x)=A(x)^{-1} \bmod G(x)$으로 A(x)의 역원이므로, U=1이면 R(x)의 값을 출력하고 연산은 종료된다(S180).

이와 달리, U≠1이면, 반복계수 δ값을 변화시키면서 U=1이 될 때까지 역원을 구하는 연산을 수행한다. 먼저, u_0값을 확인한다(S120). u_0값이 1이면 변수 U와 R에 대해 각각 U=U/x와 R=R/x mod G(x) 연산을 수행하여 U의 최하위 비트를 제거함으로써 U의 차수를 1씩 감소시키고(S125), 반복계수 δ를 1만큼 감한 후(S170) S110단계로 진행한다. 이와 달리, u_0값이 1이 아니면, 반복계수 δ값이 음수인지 확인한다(S130). 만약, 반복계수 δ가 음수이면 U와 V, 그리고, R과 S를 교환하고 δ의 부호를 변경한다(S135). S135단계를 수행한 후 u_0와 v_0는 항상 1이며, u_1 과 v_1의 값은 0 또는 1이다.

반복계수가 양수로 확인되거나 S135단계를 수행하여 반복계수 δ를 양수로 변경한 후, u_1값과 v_1값이 동일한지 확인한다(S140). 만약, u_1값과 v_1 값이 동일하면, 변수 U와 R에 대해 각각 U=U+ V와 R=R+ S 연산을 수행하여 값을 변경한다(S145). 이와 달리, u_1값과 v_1값이 다르면, 변수 U와 R에 대해 각각 U=U+ V+ x와 R=R+ S+ x·S 연산을 수행하여 값을 변경한다(S150). 다음으로, 변수 U와 R에 대해 각각 U=U/x2, R=R/x2 mod G(x) 연산을 수행하여 U의 최하위 2비트를 제거하고, 반복계수 δ를 1만큼 감한 후(S160), S170단계로 진행한다.

U의 하위 2비트를 (0,0)으로 항상 제거하기 위해 표 1에 기재된 바와 같이 덧셈 연산을 수행한다. 예를 들어, (u_1,u_0)=(1,1)와 (v_1, v_0)=(0,1)인 경우, (u_1+v_1, u_0+v_0)=(1,0)이다. 또한, $(x \cdot v_1, x \cdot v_0)$=(1,0)이고, $(u_1+v_1+x \cdot v_1, u_0+v_0+x \cdot v_0)$=(0,0)이므로, S160단계를 수행한 후 U의 차수는 2만큼 감소한다.

[표 1]

(u_1,u_0)	(v_1, v_0)	덧셈 연산
(0,1)	(0,1)	U=U+V
(0,1)	(1,1)	U=U+V+x · V
(1,1)	(0,1)	U=U+V+x · V
(1,1)	(1,1)	U=U+V

도 1에 도시된 본 발명에 따른 유한체에서의 역원 연산 방법을 컴퓨터에서 실행시키기 위한 프로그램을 작성하기 위한 전단계로서의 알고리즘으로 나타내면 다음과 같다.

입력: $A(x) \in GF(2^m),\ \ G(x)$

출력: $A(x)^{-1} \mod G(x)$

1.U=A(x), V=G(x), R=1, S=0

2. While U≠1 do the following:

2.1 If u_0=0 then

2.1.1 U=U/x, R=R/x mod G(x)

2.2 Else

2.2.1 If δ<0 then

2.2.1.1 U↔V, R↔S, δ=-δ

2.2.2 End If.

2.2.3 If u_1=v_1 then

2.2.3.1 U=U+ V, R=R+ S

2.2.4 Else

2.2.4.1 U=U+ V+ x·V, R=R+ S+ x·S

2.2.5 End If.

2.2.6 U=U/x^2, R=R/x^2 mod G(x), δ=δ-1

2.3 End If.

2.4 δ=δ-1

3. End While.

4. Return (R)

상기의 본 발명에 따른 유한체에서의 역원 연산 방법에 대한 알고리즘은 반복문 수행시 최대 2비트씩 감소하여 1.5m번 반복문을 수행하여 역원을 계산할 수 있다. 이와 같은 본 발명에 따른 유한체에서의 역원 연산 방법에 대한 알고리즘의 수행 횟수는 C. H. Wu, C. M. Wu, M. D. Shieh, 그리고, Y. T. Hwang가 "보안통신을 위한 유한체 GF(2^m)에 대한 면적 효율적인 시스톨릭 나눗셈 회로"라는 제목의 논문에서 제안한 알고리즘의 수행횟수의 65%로 확인되었다.

한편, 종래의 EBGA는 deg(U)와 deg(V)을 비교하는 연산이 필요하므로, 하드웨어로 구현할 때, U와 V의 차수를 저장하는 레지스터가 필수적으로 요구된다. 이 경우 m이 커지면 비교 연산에 따른 알고리즘 지연이 발생하므로, 본 발명에 따른 유한체에서의 역원 연산 방법은 반복계수 δ값으로 C. H. Wu, C. M. Wu, M. D. Shieh, 그리고, Y. T. Hwang가 "보안통신

을 위한 유한체 GF(2^m)에 대한 면적 효율적인 시스톨릭 나눗셈 회로"라는 제목의 논문에서 제시한 값을 이용한다. 즉, deg(U)와 deg(V)가 존재하므로 $|U| \leq x^{\alpha}$와 $|V| \leq x^{\beta}$를 만족하는 α와 β가 존재한다. 따라서 α와 β를 저장하는 대신 δ=α-β를 사용한다. δ<0이면 deg(U)<deg(V)를 의미한다.

본 발명에 따른 유한체에서의 역원 연산 방법에서 δ는 $|\delta| \leq m$이므로, δ의 크기를 나타내는 d와 δ의 부호를 나타내는 f로 나타낸다. δ<0이면 f=1이고, δ>0이면 f=0으로 표시한다. 또한 d는 (m+ 1)비트인 벡터로 다음과 같이 정의한다.

수학식 7

$$d=2^{|\delta|}=2^{\delta(-1)^f}$$

여기서, f∈{0,1}이고, d∈{1, 2, 2^2, …2^m}이다.

d의 최하위 비트를 d_0라고 하면, δ=0인 경우 d=1이고 d_0=1이다. δ≠0이면 d>1이고 d_0=0이다. 따라서 다음과 같이 d_0와 f를 이용하여 δ의 값을 표현할 수 있다.

* 만약 δ<0이면, d_0=0이고 f=1이다.

* 만약 δ=0이면, d_0=0이다.

* 만약 δ>0이면, d_0=0이고 f=0이다.

이러한 방법을 본 발명에 따른 유한체에서의 역원 연산 방법에 적용하면, S135단계에서 수행되는 δ의 부호를 변경하는 과정은 f=0으로 설정하는 과정으로 변경된다. 또한 S170단계에서 수행되는 δ에서 1을 감하는 과정은 다음의 알고리즘에 의해 제어된다.

1. If (d0,f)=(0,0) then

1.1 d=d/x

2. Else

2.1 d=d·x, f=1

3. End If.

표 2에는 GF(2^4)에서의 입력 G(x)와 A(x)가 각각 $G(x)=x^4+x+1$이고 $A(x)=x^3+x$인 경우에 본 발명에 따른 유한체에서의 역원 연산 방법에 의한 역원 연산 과정이 기재되어 있다.

[표 2]

인덱스	U	V	R	S	δ
0	x^3+x	x^4+x+1	1	0	-1
1	x^2+1	x^4+x+1	x^3+1	0	-2
2	x^2+x+1	x^2+1	x	x^3+1	0
3	x	x^2+1	x^3+x+1	x^3+1	-2
4	1	x^2+1	x^3+x^2	x^3+1	-3

도 2는 본 발명에 따른 유한체에서 역원 연산 장치에 대한 바람직한 실시예의 상세한 구성을 도시한 블록도이다.

도 2를 참조하면, 본 발명에 따른 유한체에서 역원 연산 장치(200)는 제1연산부(210), 제2연산부(230), 제어부(250), 제1제어값저장부(260), 및 제2제어값저장부(265)로 구성된다.

제1연산부(210)는 제1레지스터(212), 제2레지스터(214), 제1가산기(216), 제1먹스(218), 및 제1쉬프터(220)로 구성된다.

제1레지스터(212) 및 제2레지스터(214)에는 각각 변수 U와 V의 값이 저장된다. 변수 U와 V의 차수는 최대 m이므로, 제1레지스터(212) 및 제2레지스터(214)로는 비트수 n이 m+1인 레지스터가 사용된다. 이 때, 제1레지스터(212) 및 제2레지스터(214)는 각각 본 발명에 따른 유한체에서 역원 연산 장치(200)의 입력값인 A(x)와 G(x)로 초기화된다.

제1가산기(216)는 변수 U의 최하위 비트인 u_0의 값에 따라 변수 U와 V에 대한 연산을 수행한다. 즉, 제1가산기(216)는 u_0가 1이고 u_1과 v_1의 값이 동일한 경우에 제1레지스터(212)로부터 입력되는 U와 제2레지스터(214)로부터 입력되는 V를 합산하여 출력한다.

제1먹스(218)는 δ값에 따라 제1레지스터(212)로부터 입력되는 U와 제2레지스터(214)로부터 입력되는 V와 제1가산기(216)로부터 입력되는 값을 선택적으로 출력한다. 즉, 제1먹스(218)는 $\delta<0$이면 제1레지스터(212)로부터 입력되는 U와 제2레지스터(214)로부터 입력되는 V를 출력하고, $\delta\geq0$이면 제1가산기(216)로부터 입력되는 값을 출력한다.

제1쉬프터(220)는 제1먹스(218)로부터 입력되는 값을 오른쪽으로 한비트 천이시킨 후 출력한다. 제1쉬프터(220)의 출력값은 제1레지스터(212), 제2레지스터(214), 및 제어부(250)로 입력된다.

제2연산부(230)는 제3레지스터(232), 제4레지스터(234), 제2가산기(236), 제3가산기(238), 제2먹스(240), 제3먹스(242), 제4먹스(244), 및 제2쉬프터(246)로 구성된다.

제3레지스터(232) 및 제4레지스터(234)에는 각각 변수 R과 S의 값이 저장된다. 변수 R과 S의 차수는 최대 m-1이므로, 제3레지스터(232) 및 제4레지스터(234)로는 비트수 n이 m+1인 레지스터가 사용된다. 이 때, 제3레지스터(232) 및 제4레지스터(234)는 각각 1과 0으로 초기화된다.

제2먹스(240)는 R의 하위 2비트의 조건에 따라 사전에 계산된 G, G/x, 및 (G+xG)x의 값 중에서 하나를 선택하여 출력한다. 제2가산기(236)는 제2먹스(240)으로부터 입력되는 값과 제4레지스터(234)로부터 입력되는 변수 S의 값을 합산하여 출력한다. 제3먹스(242)는 제4레지스터(234)로부터 입력되는 변수 S의 값, 제2가산기(236)로부터 입력되는 값, 그리고, 사전에 계산된 G값 중에서 하나의 값을 선택하여 출력한다. 제3가산기(238)는 제3레지스터(232)로부터 입력되는 변수 R의 값과 제3먹스(242)로부터 입력되는 값을 합산하여 출력한다. 제4먹스(244)는 제3레지스터(232)로부터 입력되는 변수 R의 값, 제4레지스터(234)로부터 입력되는 변수 S의 값, 그리고, 제3가산기(238)로부터 입력되는 값 중에서 하나를 선택하여 출력한다. 제2쉬프터(246)는 제4먹스(244)로부터 입력되는 값을 오른쪽으로 한비트 천이시킨 후 출력한다. 제2쉬프터(246)의 출력값은 제3레지스터(232), 제4레지스터(234), 및 제어부(250)로 입력된다.

제어부(250)는 제1연산부(210) 및 제2연산부(230)의 연산동작을 제어한다. 제어부(250)의 제어동작은 U의 최하위 비트값인 u_0의 값과 제1제어값저장부(260) 및 제2제어값저장부(265)에 저장되어 있는 제어값에 의해 구동된다. 제1제어값저장부(260)에는 도 1을 참조하여 설명한 바와 같이 δ의 크기를 나타내며, (m+1)비트의 벡터로 정의되는 d의 최하위 비트값인 d_0가 저장된다. 또한, 제2제어값저장부(265)에는 도 1을 참조하여 설명한 바와 같이 δ의 부호를 나타내는 f가 저장된다.

먼저, 제어부(250)에 의한 제1연산부(210)의 제어동작부터 설명한다. 만약 u_0가 0이면 제어부(250)는 제1연산부(210)의 제1먹스(218)가 제1레지스터(212)로부터 입력되는 변수 U의 값을 제1쉬프터(220)로 출력하도록 제어한다. 이와 달리, $u_0\neq0$, 제1제어값저장부(260)에 저장되어 있는 d의 최하위 비트값인 $d_0=0$, 그리고, 제2제어값저장부(265)에 저장되어 있는 f=1이면, 제어부(250)는 제1레지스터(212)와 제2레지스터(214)에 저장되어 있는 변수 U값과 변수 V값을 교환하도록 제어하고, δ의 부호를 변경한다.

한편, $u_0 \neq 0$이고, u_1과 v_1의 값이 동일하면, 제어부(250)는 제1먹스(218)가 제1가산기(216)로부터 입력되는 값을 제1쉬프터(220)로 출력하도록 제어하고, 제1쉬프터(220)가 제1먹스(218)로부터 입력받은 값을 2비트 오른쪽으로 쉬프트한 후 제1레지스터(212)에 저장하도록 제어한다. 또한, $u_0 \neq 0$이고 u_1과 v_1의 값이 상이하면, 제1연산부(210)는 다음의 수학식으로 표현되는 두 단계의 덧셈연산을 수행하여야 한다.

수학식 8

$$U=(U+V+x \cdot V)/x^2$$

수학식 8은 다음의 식과 같다.

수학식 9

$$U=\{(U+V)/x+V\}/x$$

수학식 9는 부분합 (U+ V)/x을 수행하여 제1레지스터(212)에 결과를 저장하는 제1단계 연산과 제1레지스터(212)에 저장되어 있는 부분합과 제2레지스터(214)에 저장되어 있는 변수 V의 값을 합산하여 오른쪽 쉬프트 연산을 수행하여 제1레지스터(212)에 저장하는 제2단계 연산으로 구분할 수 있다. 따라서, 제어부(250)는 수학식 9에 따른 연산을 수행하도록 제1연산부(210)를 제어한다.

즉, 제어부(250)는 1단계로 제1먹스(218)가 제1가산기(216)로부터 입력되는 값을 제1쉬프터(220)로 출력하도록 제어하고, 제1쉬프터(220)가 제1먹스(218)로부터 입력받은 값을 1비트 오른쪽으로 쉬프트한 후 제1레지스터(212)에 저장하도록 제어한다. 이어서, 제어부(250)는 2단계로 제1먹스(218)가 제1가산기(216)로부터 입력되는 값을 제1쉬프터(220)로 출력하도록 제어하고, 제1쉬프터(220)가 제1먹스(218)로부터 입력받은 값을 1비트 오른쪽으로 쉬프트한 후 제1레지스터(212)에 저장하도록 제어한다. 이상가 같은 제어부(250)에 의한 제1연산부(210)의 제어동작은 제1레지스터(212)에 저장되는 변수 U의 값이 1이 될때까지 반복적으로 수행된다.

다음으로, 제어부(250)에 의한 제2연산부(230)의 제어동작부터 설명한다. 만약 $u_0 \neq 0$, 제1제어값저장부(260)에 저장되어 있는 d의 최하위 비트값인 $d_0=0$, 그리고, 제2제어값저장부(265)에 저장되어 있는 f=1이면, 제어부(250)는 제3레지스터(232)와 제4레지스터(234)에 저장되어 있는 변수 R값과 변수 S값을 교환하도록 제어하고, δ의 부호를 변경한다.

한편, $u_0 \neq 0$이고, u_1과 v_1의 값이 동일하면, 제어부(250)는 제4먹스(244)가 제3가산기(238)로부터 입력되는 값을 제1쉬프터(220)로 출력하도록 제어한다. 이 때, 제3가산기(238)는 제3레지스터(232)로부터 입력되는 변수 R의 값과 제2먹스(242)를 통해 제4레지스터(234)로부터 입력되는 변수 S의 값을 합산하여 제4먹스(244)에 제공한다.

이와 달리, $u_0 \neq 0$이고 u_1과 v_1의 값이 상이한 경우에 제어부(250)는 제1연산부(210)에 대한 제어동작과는 상이하게 제2연산부(230)를 제어한다. 즉, 앞서 설명한 본 발명에 따른 역원 연산 방법을 구현한 알고리즘의 2.1.1단계와 2.2.6단계에서 변수 U와 변수 R에 대해 동일한 연산을 적용할 수 없으므로, 제2연산부(230)는 종래의 EBGA에 수행되는 모듈러 감산 연산을 수행하여야 한다. 모듈러 감산 연산을 하드웨어로 구현할 때 알고리즘의 지연시간에 영향을 주므로 효율적인 연산구성이 요구된다. 따라서, 효율적인 연산구성을 위해 본 발명에 따른 역원 연산 장치(200)는 사전에 계산된 G, G/x, 및 (G+ xG)/x를 이용하여 모듈러 감산연산을 수행한다. R의 하위 2비트의 조건에 따라 제2먹스(240)는 사전에 계산된 G, G/x, 및 (G+ xG)/x 중 하나의 값을 선택하여 출력한다. 그리고, 제2가산기(236)는 제2먹스(240)로부터 입력되는 값과 제4레지스터(234)로부터 입력되는 변수 S의 값을 합산하여 모듈러 감산 연산을 수행한다.

제2연산부(230)는 제1연산부(210)와 동일하게 두 단계의 덧셈 연산을 수행하며, 이를 수학식으로 표현하면 다음과 같다.

수학식 10

$$\{(R+S)/x+S+k\}/x, \quad k=G, G/x, (G+xG)/x$$

수학식 10에서 덧셈 연산의 1단계로 제3가산기(238)는 제3레지스터(232)로부터 입력되는 변수 R의 값과 제4레지스터(234)로부터 입력되는 변수 S의 값을 가산하여 제4먹스(244)로 출력한다. 제2쉬프터(246)는 제4먹스(244)를 통해 제3가산기(238)로부터 입력되는 값을 오른쪽으로 한비트 쉬프트하여 산출된 부분합 (R+ S)/x를 제3레지스터(232)로 출력한다. 한편, 부분합(R+ S)/x를 계산하는 동안 제2먹스(240)와 제2가산기(236)가 (S+ k) 연산을 수행하면, 모듈러 연산에 따른

알고리즘 지연이 발생하지 않는다. 이 때, k=G, G/x, 및 (G+ xG)/x 중에서 선택된다. 이어서, 덧셈 연산의 2단계로 제3가산기(238)는 제2가산기(236)로부터 입력되는 값과 제3레지스터(232)에 저장되어 있는 부분합을 가산하여 제4먹스(244)로 출력한다. 제2쉬프터(246)는 제4먹스(244)를 통해 제3가산기(238)로부터 입력되는 값을 오른쪽으로 한비트 쉬프트한 후 제3레지스터(232)에 저장한다.

임의의 A(x)∈GF(2^m)의 역원은 제1레지스터(212)에 저장되어 있는 변수 U의 값이 1일 때, 제3레지스터(232)에 저장되어 있는 변수 R의 값이므로, 제어부(250)는 변수 U가 1일 때 제3레지스터(232)에 저장되어 있는 변수 R의 값을 결과값으로 출력한다.

이하에서 WTLS #3번 타원곡선의 환경에서 다음과 같은 기약 다항식 G(x)을 기반으로 GF(2^m)의 임의의 원소에 대한 역원 연산을 1,000,000번 테스트한 결과를 제시한다.

수학식 11

$$G(x) = x^{163} + x^7 + x^6 + x^3 + x^2 + 1$$

본 발명에 따른 역원 연산 방법 및 종래의 EBGA에 의해 WTLS #3번 타원곡선의 환경에서 다음과 같은 기약 다항식 G(x)을 기반으로 GF(2^m)의 임의의 원소에 대한 역원 연산의 수행결과는 표 3에 기재되어 있다.

[표 3]

알고리즘	덧셈수			리덕션수			루프수		
	최소	최대	평균	최소	최대	평균	최소	최대	평균
본 발명	118	220	165	53	115	82	184	251	220
EBGA	119	204	161	117	206	161	286	325	321

표 3을 참조하면, 본 발명에 따른 역원 연산 방법은 반복문을 한 번 수행할 때 쉬프트 연산이 최대 2번 발생하나, EBGA는 쉬프트 연산이 매회 1번 수행된다. 따라서 모듈러 감산 연산의 평균 수행횟수에 따르면, 본 발명에 따른 역원 연산 방법이 종래의 EBGA보다 1.5배 빠르게 수행한다.

덧셈 연산의 평균 수행횟수를 비교하면 본 발명에 따른 역원 연산 방법은 165번이고, 종래의 EBGA는 161번으로 나타났다. 즉 하드웨어 구조 관점에서 본 발명에 따른 역원 연산 방법이 종래의 EBGA보다 덧셈기의 수는 증가하였으나, 두 알고리즘의 덧셈 연산 평균 수행횟수에는 차이를 보이지 않음을 알 수 있다.

루프(Loop)의 평균 수행횟수를 비교하면 본 발명에 따른 역원 연산 방법은 220번이고, 종래의 EBGA는 321번이다. 결론적으로 표 4에 기재된 테스트 결과에 따르면, 본 발명에 따른 역원 연산 방법은 종래의 EBGA에 비해 알고리즘의 수행횟수를 평균 약 31.5%를 감소시켰으며, 기존의 EBGA의 65%인 수행횟수를 가진다.

표 4에는 본 발명에 따른 역원 연산 방법을 하드웨어로 구현한 장치와 R. Lorenzo의 "New Algorithm for Classical Modular Inverse"라는 제목의 논문에 개시된 알고리즘 및 T. Zhou, X. Wu, G. Bai, 그리고 H. Chen의 "Fast GF(p) modular inversion algorithm suitable for VLSI implementation"라는 제목의 논문에 개시된 알고리즘을 구현한 하드웨어의 리소스가 기재되어 있다.

[표 4]

	본 발명	R. Lorenzo	T. Zhou 등
최대지연경로	$4T_{Mux3}+2T_{Add2}+2T_{RShift}$	$2T_{Mux2}+2T_{Add2}+T_{Mux3}+T_{LRShift}$	$4T_{Mux3}+4T_{Add3}+4T_{Mux5}+T_{Rshift}$
레지스터 갯수	6	8	4
덧셈기 갯수	3	2	1
먹스 갯수	4	3	6

쉬프터	오른쪽 쉬프터	왼쪽/오른쪽 쉬프터	오른쪽 쉬프터
	2	2	4
카운터 갯수	0	2	0

표 4를 참조하면, GF(p)에서의 역원 알고리즘은 GF(2^m)에서의 역원 알고리즘과 동일한 연산을 수행하며, 동일한 하드웨어 구조를 가진다. 또한 본 발명에 따른 역원 연산 방법은 GF(p)에서의 역원 알고리즘으로 확장 가능하므로 R. Lorenzo와 T. Zhou 등이 제시한 알고리즘을 구현한 하드웨어의 구조를 비교 대상으로 채택하였다.

표 4를 참조하여 본 발명에 따른 역원 연산 장치와 R. Lorenzo가 제시한 알고리즘을 구현한 하드웨어의 구조를 비교하면, 본 발명에 따른 역원 연산 장치는 R. Lorenzo가 제시한 알고리즘을 구현한 하드웨어보다 덧셈기 1개와 Mux 1개가 더 사용된다. 한편, 3-to-1 곱셈기를 이용하므로 본 발명에 따른 역원 연산 장치의 최대지연경로는 $4T_{Mux3}+2T_{Add2}+2T_{RShift}$ 이며, R. Lorenzo가 제시한 알고리즘을 구현한 하드웨어의 최대지연경로는 $2T_{Mux2}+2T_{Add2}+T_{Mux3}+T_{LRShift}$이다. 그러나 R. Lorenzo가 제시한 알고리즘을 구현한 하드웨어는 알고리즘 제어를 위해 제어 레지스터가 요구되어 레지스터 개수가 증가한다. 또한 카운터와 값의 조건에 따라 하드웨어를 제어하므로 제어로직이 복잡하다.

한편, 본 발명에 따른 역원 연산 장치는 Right Shifter를 이용하나, R. Lorenzo가 제시한 알고리즘을 구현한 하드웨어는 Left/Right Shifter를 사용하므로 이에 따른 연산 지연이 발생할 수 있다. 또한, T. Zhou 등이 제시한 알고리즘을 구현한 하드웨어는 역원 알고리즘을 순차적으로 수행한다. 즉, 모듈러 감산 연산을 실행한 후 U와 V에 대해 처리한다. 그리고, V의 조건에 따라 R과 S에 대한 연산을 실행한다. 따라서 T. Zhou 등이 제시한 알고리즘을 구현한 하드웨어는 1개의 덧셈기를 이용하나, 본 발명에 따른 역원 연산 장치에 비해 Mux를 2개 더 사용한다. T. Zhou 등이 제시한 알고리즘을 구현한 하드웨어는는 5-to-1 먹스를 이용하므로 본 발명에 따른 역원 연산 장치보다 큰 수행시간지연이 발생한다.

본 발명은 또한 컴퓨터로 읽을 수 있는 기록매체에 컴퓨터가 읽을 수 있는 코드로서 구현하는 것이 가능하다. 컴퓨터가 읽을 수 있는 기록매체는 컴퓨터 시스템에 의하여 읽혀질 수 있는 데이터가 저장되는 모든 종류의 기록장치를 포함한다. 컴퓨터가 읽을 수 있는 기록매체의 예로는 ROM, RAM, CD-ROM, 자기 테이프, 플로피디스크, 광데이터 저장장치 등이 있으며, 또한 캐리어 웨이브(예를 들어 인터넷을 통한 전송)의 형태로 구현되는 것도 포함한다. 또한 컴퓨터가 읽을 수 있는 기록매체는 네트워크로 연결된 컴퓨터 시스템에 분산되어 분산방식으로 컴퓨터가 읽을 수 있는 코드가 저장되고 실행될 수 있다.

이상에서 본 발명의 바람직한 실시예에 대해 도시하고 설명하였으나, 본 발명은 상술한 특정의 바람직한 실시예에 한정되지 아니하며, 청구범위에서 청구하는 본 발명의 요지를 벗어남이 없이 당해 발명이 속하는 기술분야에서 통상의 지식을 가진 자라면 누구든지 다양한 변형 실시가 가능한 것은 물론이고, 그와 같은 변경은 청구범위 기재의 범위 내에 있게 된다.

발명의 효과

본 발명에 따른 유한체에서의 역원 연산 장치 및 방법에 의하면, 종래의 역원 연산 알고리즘에 비해 1.5배 빠르게 연산을 수행할 수 있고, 종래의 역원 연산 알고리즘의 수행횟수의 65%정도로 수행횟수를 줄일 수 있다. 또한, 본 발명에 따른 유한체에서의 역원 연산 장치는 종래의 역원 연산 알고리즘을 구현한 하드웨어보다 모듈러 감산 연산을 효율적으로 처리하며, 덧셈기의 수는 증가하였으나 덧셈 연산 수행횟수는 종래의 역원 연산 알고리즘을 구현한 하드웨어와 거의 비슷하다. 또한, 본 발명에 따른 유한체에서의 역원 연산 장치는 종래의 역원 연산 알고리즘을 구현한 하드웨어와 거의 유사한 최대지연경로를 가지고 있으며, 제어 부분이 종래의 역원 연산 알고리즘을 구현한 하드웨어보다 단순하다. 이와 같은 장점으로 인해 본 발명에 따른 유한체에서의 역원 연산 장치 및 방법은 GF(p)에서의 역원 알고리즘으로 확장이 가능하여, 공개키 암호 시스템의 효율성을 높일 수 있으며, 나아가, 처리 능력과 에너지 가용 능력이 제한된 환경인 스마트 카드, 토큰 하드웨어, 휴대용 단말기 등의 장비에 효과적으로 적용될 수 있다.

(57) 청구의 범위

청구항 1.

유한체 GF(2^m)의 원소인 제1다항식 A(x)와 소정의 m차 기약다항식인 제2다항식 G(x)를 입력으로 하고, 상기 제1다항식과 상기 제2다항식을 각각 소정의 제1내부변수 및 제2내부변수의 초기값으로 설정한 후 교환연산제어변수, 상기 제1내부변수의 최하위비트값, 및 상기 제1내부변수와 상기 제2내부변수의 최하위비트의 인접비트값의 동일여부에 따라 상기 제1내부변수가 1이 될 때까지 상기 제1내부변수와 상기 제2내부변수에 대한 교환연산, 쉬프트연산, 및 덧셈연산을 선택적으로 반복수행하는 제1연산부;

상기 교환연산제어변수, 상기 제1내부변수의 최하위비트값, 및 상기 제1내부변수와 상기 제2내부변수의 최하위비트의 인접비트값의 동일여부에 따라 상기 제1내부변수가 1이 될 때까지 소정의 초기값이 설정된 제3내부변수 및 제4내부변수에 대한 교환연산, 쉬프트연산, 및 덧셈연산을 선택적으로 반복수행하며, 상기 덧셈연산을 수행하는 동안 상기 제2다항식에 대해 G(x)/x 및 (G+ Gx)/x 연산을 수행하여 얻어진 G(x), G(x)/x, 및 (G+ Gx)/x 중에서 선택된 제3다항식과 상기 제4내부변수의 값을 합산하여 모듈러 감산 연산을 수행하는 제2연산부;

상기 교환연산제어변수, 상기 제1내부변수의 최하위비트값, 및 상기 제1내부변수와 상기 제2내부변수의 최하위비트의 인접비트값의 동일여부에 따라 상기 제1연산부 및 상기 제2연산부의 동작을 제어하며, 상기 제1내부변수가 1이 되면 상기 제3내부변수를 상기 제1다항식의 역원으로 출력하는 제어부; 및

상기 교환연산제어변수가 저장되는 제어값저장부;를 포함하는 것을 특징으로 하는 유한체에서의 역원 연산 장치.

청구항 2.

제 1항에 있어서,

상기 제1연산부는,

상기 제1내부변수가 저장되는 제1레지스터;

상기 제2내부변수가 저장되는 제2레지스터;

상기 제1레지스터 및 상기 제2레지스터로부터 입력받은 상기 제1내부변수 및 상기 제2내부변수를 가산하여 출력하는 제1가산기;

상기 제1레지스터로부터 입력받은 상기 제1내부변수, 상기 제2레지스터로부터 입력받은 상기 제2내부변수, 및 상기 제1가산기로부터 입력받은 값 중에서 하나의 값을 선택하여 출력하는 제1먹스; 및

상기 제1먹스로부터 입력되는 값을 하위비트방향으로 1비트 이동시켜 출력하는 제1쉬프터;를 포함하며,

상기 제어부는 상기 제1내부변수의 최하위비트값이 0이면, 상기 제1먹스가 상기 제1내부변수를 선택하여 출력하도록 제어하고, 상기 제1내부변수의 최하위비트값이 0이 아니면, 상기 제1먹스가 상기 제1가산기로부터 입력되는 값을 선택하여 출력하도록 제어하며,

상기 제어부는 상기 제1내부변수의 최하위비트값이 0이 아닌 경우에 상기 제1내부변수와 상기 제2내부변수의 최하위비트의 인접비트값이 동일하면 상기 제1쉬프터가 상기 제1먹스로부터 입력되는 값을 하위비트방향으로 2비트 이동시킨 후 상기 제1레지스터에 저장하도록 제어하고, 상기 제1내부변수와 상기 제2내부변수의 최하위비트의 인접비트값이 상이하면 상기 제1쉬프터가 상기 제1먹스로부터 입력되는 값을 하위비트방향으로 1비트 이동시킨 후 상기 제1레지스터에 저장하도록 제어한 후 연속적으로 상기 제1쉬프터가 상기 제1먹스로부터 입력되는 값을 하위비트방향으로 1비트 이동시킨 후 상기 제1레지스터에 저장하도록 제어하는 것을 특징으로 하는 유한체에서의 역원 연산 장치.

청구항 3.

제 2항에 있어서,

상기 제어부는 상기 교환연산제어변수가 0보다 작으면 상기 제1내부변수값 및 상기 제2내부변수값을 교환하여 각각 상기 제2레지스터 및 상기 제1레지스터에 저장하도록 제어하고, 상기 교환연산제어변수의 부호를 변경하며,

상기 교환연산제어변수는 다음과 같이 정의되는 것을 특징으로 하는 유한체에서의 역원 연산 장치:

$\delta=\alpha-\beta$,

여기서, δ는 상기 교환연산제어변수이고, α 및 β는 각각 $|$제1내부변수$| \leq x^{\alpha}$와 $|$제2내부변수$| \leq x^{\beta}$를 만족하는 값이다.

청구항 4.

제 1항 또는 제 2항에 있어서,

상기 제2연산부는,

상기 제3내부변수가 저장되는 제3레지스터;

상기 제4내부변수가 저장되는 제4레지스터;

상기 제3내부변수의 하위 2비트값에 따라 상기 G(x), G(x)/x, 및 (G+ Gx)/x 중에서 하나를 상기 제3다항식으로 선택하여 출력하는 제2먹스;

상기 제4레지스터로부터 입력받은 상기 제4내부변수와 상기 제2먹스로부터 입력받은 상기 제3다항식을 합산하여 출력하는 제2가산기;

상기 제2가산기로부터 입력받은 값, 상기 제2다항식, 및 상기 제4레지스터로부터 입력받은 상기 제4내부변수 중에서 하나의 값을 선택하여 출력하는 제3먹스;

상기 제3먹스로부터 입력받은 값과 상기 제3레지스터로부터 입력받은 상기 제3내부변수를 합산하여 출력하는 제3가산기;

상기 제3레지스터로부터 입력받은 상기 제3내부변수, 상기 제4레지스터로부터 입력받은 상기 제4내부변수, 및 상기 제3가산기로부터 입력받은 합산값 중에서 하나의 값을 선택하여 출력하는 제4먹스; 및

상기 제4먹스로부터 입력되는 값을 하위비트방향으로 1비트 이동시켜 출력하는 제2쉬프터;를 포함하며,

상기 제어부는 상기 제1내부변수의 최하위비트값이 0이면, 상기 제4먹스가 상기 제3내부변수를 선택하여 출력하도록 제어하고, 상기 제1내부변수의 최하위비트값이 0이 아니면, 상기 제4먹스가 상기 제4가산기로부터 입력되는 값을 선택하여 출력하도록 제어하고,

상기 제어부는 상기 제1내부변수의 최하위비트값이 0이 아닌 경우에 상기 제1내부변수와 상기 제2내부변수의 최하위비트의 인접비트값이 동일하면 상기 제2쉬프터가 상기 제4먹스로부터 입력되는 값을 하위비트방향으로 2비트 이동시킨 후 상기 제3레지스터에 저장하도록 제어하고, 상기 제1내부변수와 상기 제2내부변수의 최하위비트의 인접비트값이 상이하면 상기 제2쉬프터가 상기 제4먹스로부터 입력되는 값을 하위비트방향으로 1비트 이동시킨 후 상기 제3레지스터에 저장하도록 제어한 후 연속적으로 상기 제2쉬프터가 상기 제4먹스로부터 입력되는 값을 하위비트방향으로 1비트 이동시킨 후 상기 제3레지스터에 저장하도록 제어하는 것을 특징으로 하는 유한체에서의 역원 연산 장치.

청구항 5.

제 4항에 있어서,

상기 제어부는 상기 제1내부변수의 최하위비트값이 0이 아닌 경우에 상기 제1내부변수와 상기 제2내부변수의 최하위비트의 인접비트값이 상이하면 상기 제2쉬프터가 상기 제4먹스로부터 입력되는 값을 하위비트방향으로 1비트 이동시키는 과정을 수행하는 동안에 상기 제3가산기가 상기 제2먹스로부터 입력되는 값과 상기 제4레지스터로부터 입력되는 상기 제4내부변수값을 합산하도록 제어하는 것을 특징으로 하는 유한체에서의 역원 연산 장치.

청구항 6.

제 4항에 있어서,

상기 제어부는 상기 교환연산제어변수가 0보다 작으면 상기 제3내부변수값 및 상기 제4내부변수값을 교환하여 각각 상기 제4레지스터 및 상기 제3레지스터에 저장하도록 제어하고, 상기 교환연산제어변수의 부호를 변경하며,

상기 교환연산제어변수는 다음과 같이 정의되는 것을 특징으로 하는 유한체에서의 역원 연산 장치:

$\delta=\alpha-\beta$,

여기서, δ는 상기 교환연산제어변수이고, α 및 β는 각각 $|$제1내부변수$|\leq x^{\alpha}$와 $|$제2내부변수$|\leq x^{\beta}$를 만족하는 값이다.

청구항 7.

유한체 $GF(2^m)$의 원소인 제1다항식 A(x)와 소정의 m차 기약다항식인 제2다항식 G(x)를 각각 소정의 제1내부변수 및 제2내부변수의 초기값으로 설정하는 단계;

소정의 제3내부변수 및 제4내부변수의 초기값을 각각 1과 0으로 설정하는 단계;

소정의 교환연산제어변수가 0이면, 상기 제1내부변수를 하위비트 방향으로 1비트 이동시키고, 상기 제3내부변수를 하위비트 방향으로 1비트 이동시킨 후 상기 제2다항식과의 모듈러 연산을 수행하는 단계;

상기 교환연산제어변수가 0이 아닌 경우에 상기 제1내부변수와 상기 제2내부변수의 최하위비트의 인접비트값이 동일하면, 상기 제1내부변수를 상기 제1내부변수와 상기 제2내부변수의 합산값으로 변경하고, 상기 제3내부변수를 상기 제3내부변수와 상기 제4내부변수의 합산값으로 변경하는 단계;

상기 교환연산제어변수가 0이 아닌 경우에 상기 제1내부변수와 상기 제2내부변수의 최하위비트의 인접비트값이 상이하면, 상기 제1내부변수를 상기 제1내부변수, 상기 제2내부변수, 및 상기 제2내부변수와 x의 곱의 합산값으로 변경하고, 상기 제3내부변수를 상기 제3내부변수, 상기 제4내부변수, 및 상기 제4내부변수와 x의 곱의 합산값으로 변경하는 단계;

상기 제1내부변수를 하위비트 방향으로 2비트 이동시키고, 상기 제3내부변수를 하위비트방향으로 2비트 이동시킨 후 상기 제2다항식과의 모듈러 연산을 수행하며, 상기 교한연산제어변수의 값을 변경하는 단계; 및

상기 제1내부변수가 1이면 상기 제3내부변수를 상기 제1다항식의 역원으로 출력하는 단계;를 포함하는 것을 특징으로 하는 유한체에서의 역원 연산 방법.

청구항 8.

제 7항에 있어서,

상기 교환연산제어변수가 0보다 작으면 상기 제1내부변수값 및 상기 제2내부변수값을 교환하고, 상기 제3내부변수값 및 상기 제4내부변수값을 교환하며, 상기 교환연산제어변수의 부호를 변경하는 단계를 더 포함하며,

상기 교환연산제어변수는 다음과 같이 정의되는 것을 특징으로 하는 유한체에서의 역원 연산 방법:

$\delta=\alpha-\beta$,

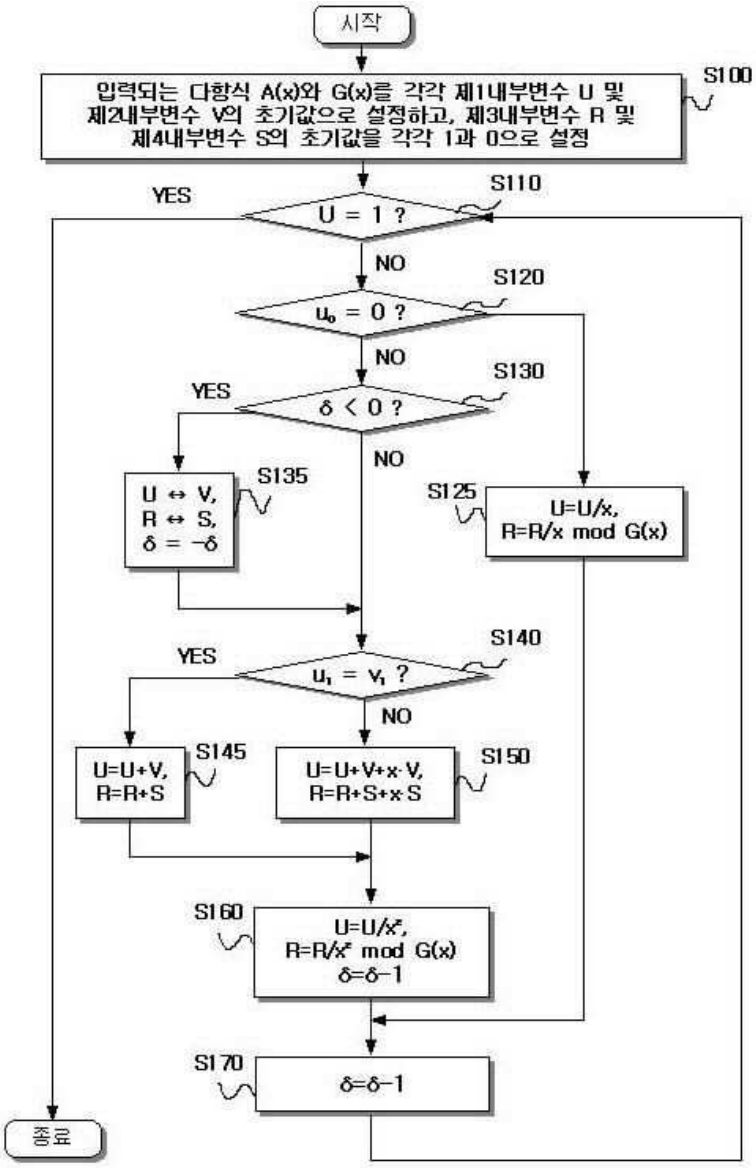
여기서, δ는 상기 교환연산제어변수이고, α 및 β는 각각 $|$제1내부변수$|\leq x^{\alpha}$와 $|$제2내부변수$|\leq x^{\beta}$를 만족하는 값이다.

청구항 9.

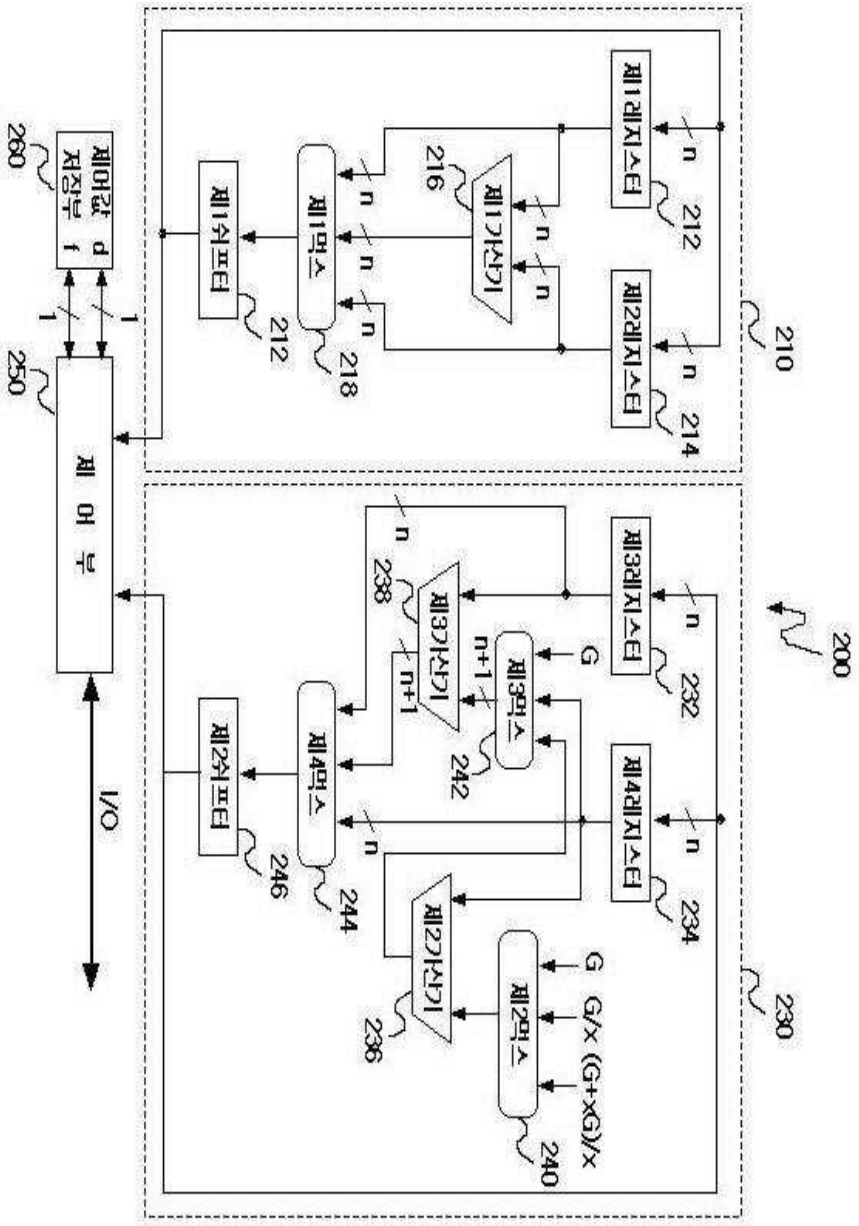
제 7항 또는 제 8항에 기재된 유한체에서의 역원 연산 방법을 컴퓨터에서 실행시키기 위한 프로그램을 기록한 컴퓨터로 읽을 수 있는 기록매체.

도면

도면1



도면2


200
210
제1레지스터
212
제2레지스터
214
제1가산기
216
제1멕스
218
제1쉬프터
212
230
제3레지스터
232
제4레지스터
234
제3가산기
238
제3멕스
242
제4멕스
244
제2쉬프터
246
제2가산기
236
제2멕스
240
G
G/X (G+XG)/X
제어부
250
I/O
제어값 저장부
260
d
f
1
n
n+1