논문 2005-42SD-9-7

경량화 시스템에 적합한 유한체 GF(2^m)에서의 고속 역원기

(A Fast Inversion for Low-Complexity System over GF(2^m))

김 소 선*, 장 남 수**, 김 창 한***

(Sosun Kim, Nam su Chang, and Chang Han Kim)

요 약

효율적인 암호 시스템의 설계는 환경에 적합한 유한체 연산이 뒷받침되어야 한다. 특히 유한체에서의 역원 연산은 다른 연산에 비해 가장 많은 수행시간을 소비하므로, 개선에 대한 연구가 활발히 진행되고 있다. 본 논문에서는 다항식 기저를 기반으로 Extended binary gcd algorithm (EBGA)를 이용한 유한체 $GF(2^m)$에서의 고속 역원 알고리즘을 제안한다. 제안된 역원 알고리즘은 EBGA보다 18.8%, Montgomery inverse algorithm (MIA)보다 45.9% 적은 수행횟수를 가진다. 또한 기존에 제안된 시스톨릭 어레이 구조 (Systolic array structure)는 유한체 차수 m이 증가하는 경우 많은 하드웨어 리소스가 요구된다. 따라서 스마트 카드나 모바일 폰 등과 같은 경량화와 저전력이 요구되는 환경에는 적용하기 힘들다. 본 논문에서는 경량화된 암호 시스템 환경을 바탕으로 공간복잡도가 적으면서 동기화된 연산을 수행하는 새로운 하드웨어 구조를 제시한다. 본 논문에서 제안된 하드웨어 구조는 유한체 $GF(2^m)$에서의 역원을 계산하기 위해 기존의 알고리즘보다 적은 덧셈 연산과 모듈러 감산 연산을 포함하고 있으며, 유한체 $GF(2^m)$와 $GF(p)$에 적용이 가능한 통합된 역원기이다.

Abstract

The design of efficient cryptosystems is mainly appointed by the efficiency of the underlying finite field arithmetic. Especially, among the basic arithmetic over finite field, the multiplicative inversion is the most time consuming operation. In this paper, a fast inversion algorithm in finite field $GF(2^m)$ with the standard basis representation is proposed. It is based on the Extended binary gcd algorithm (EBGA). The proposed algorithm executes about 18.8% or 45.9% less iterations than EBGA or Montgomery inverse algorithm (MIA), respectively. In practical applications where the dimension of the field is large or may vary, systolic array structure becomes area-complexity and time-complexity costly or even impractical in previous algorithms. It is not suitable for low-weight and low-power systems, i.e., smartcard, the mobile phone. In this paper, we propose a new hardware architecture to apply an area-efficient and a synchronized inverter on low-complexity systems. It requires the number of addition and reduction operation less than previous architectures for computing the inverses in $GF(2^m)$. Furthermore, the proposed inversion is applied over either prime or binary extension fields, more specially $GF(2^m)$ and $GF(p)$.

Keywords: Finite field arithmetic, Inversion, Extended binary gcd algorithm, Cryptography

I. 서 론

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 이 연산에 따라 암호 시스템의 효율성이 결정된다. 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, 이 중 역원 연산은 곱셈에 비해 시간 복잡도가 크므로 많은 연구의

* 정회원, ** 학생회원, 고려대학교 정보보호대학원
(Center for Information and Security Technologies (CIST), Korea Univ.)
*** 정회원, 세명대학교 정보보호학과
(Dept. of Information and Security, Semyumg Univ.)
※ 본 연구는 정보통신부 및 정보통신연구진흥원의 대학 IT연구센터 육성·지원사업의 연구결과로 수행되었음.
접수일자: 2005년3월25일, 수정완료일: 2005년8월22일

대상이 되고 있다.

유한체의 기저는 유한체 $GF(2^m)$의 연산과 밀접한 연관을 가지며 이는 정규기저 (Normal basis)와 다항식 기저 (Polynomial basis)로 나뉜다. 유한체의 기저가 정규기저인 경우 원소의 제곱 연산은 간단한 쉬프트 연산으로 표현 가능하다. 유한체 $GF(2^m)$의 임의의 원소 $A(x)$의 역원은 페르마 정리를 이용하여 $A(x)^{-1} = A(x)^{2^m-2}$로 계산할 수 있으나, 지수승 연산의 효율성이 적은 경우에는 적용하기 힘들다. 다항식기저를 사용하는 유한체의 역원 연산은 주로 Extended euclidean algorithm (EEA)과 Extended binary gcd algorithm (EBGA), Montgomery inverse algorithm (MIA)을 기반으로 한다. 이 알고리즘들은 입력된 값에 따라 반복문을 수행하면서 주어진 유한체에서의 쉬프트 연산, 덧셈이나 뺄셈 연산을 수행하면서 역원을 계산한다.

EEA와 EBGA는 입력된 값에 따라 연산의 반복 수행횟수가 고정되지 않으므로 하드웨어 구현상에 어려움이 있다. 따라서 유한체 $GF(2^m)$에서 다항식기저를 기반으로 VLSI 구현에 적합하도록 반복문이 $2m$번으로 고정된 역원 알고리즘과 시스톨릭 어레이 구조가 제안되었다[1],[2],[3],[4]. 그러나 큰 유한체 차수 m이 요구되는 시스템의 경우, 시스톨릭 어레이 구조는 하드웨어 리소스와 공간 요구량이 증가한다. 따라서 스마트 카드나 모바일 폰 등과 같은 경량화와 저전력이 요구되는 환경에는 적용하기 힘들다.

본 논문에서는 유한체 $GF(2^m)$에서 다항식기저를 바탕으로 EBGA를 수정하여 기존의 알고리즘의 수행횟수인 $2m$보다 적은 수행횟수를 가진 고속 역원 알고리즘을 제시한다. 본 논문에서 제안하는 알고리즘은 적은 공간복잡도를 가진 시스템에 적용이 가능하도록 이에 적합한 새로운 하드웨어 구조를 제안한다. 본 논문에서 제안된 하드웨어 구조는 유한체 $GF(2^m)$에서의 역원을 계산하기 위해 기존의 알고리즘보다 적은 덧셈 연산과 감산 연산을 포함하고 있으며 동기화된 연산을 수행한다. 또한 유한체 $GF(2^m)$와 $GF(p)$에 적용이 가능한 통합된 역원기이다.

본 논문의 Ⅱ장에서는 다항식기저를 사용하는 유한체 $GF(2^m)$에서 임의의 원소의 역원을 계산하는 기존의 역원 알고리즘인 EBGA을 소개한다. Ⅲ장에서는 본 논문에서 제안하는 역원 알고리즘에 대해 설명하고, Ⅳ장에서는 경량화된 암호 시스템에 적합한 하드웨어 구조를 제시한다. Ⅴ장에서는 제안하는 역원 알고리즘과 기존의 역원 알고리즘에 다양한 유한체 차수를 적용하여 테스트한 결과를 나타낸다. Ⅵ장에서는 제안하는 알고리즘의 적용 범위에 대해 살펴보고, Ⅶ장에서 결론을 맺는다.

Ⅱ. 유한체 GF(2^m)에서 기존의 역원 알고리즘

1. 유한체 GF(2^m)에서의 역원

유한체 $GF(2^m)$은 $GF(2)$에서의 m차 기약다항식 $G(x) = x^m + \sum_{i=0}^{m-1} g_i x^i$, $g_i \in GF(2)$에 의해 생성된다. 다항식기저에 의해 $GF(2^m)$의 임의의 원소 $A(x)$는

$$A(x) = a_{m-1}x^{m-1} + a_{m-2}x^{m-2} + \cdots + a_1 x + a_0,$$

$a_i \in GF(2)$로 표현되며, 벡터로 표현하면 다음과 같다.

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

유한체 $GF(2^m)$의 임의의 다항식 와 $B(x)$에 대해 덧셈과 뺄셈 연산은 XOR 연산과 동일하다. 다항식의 곱셈 연산은 $A(x) \cdot B(x) = A(x) \times B(x) \bmod G(x)$이며 나눗셈은 $A(x) \div B(x) = A(x) \times B(x)^{-1} \bmod G(x)$로 표현된다. 이 때 $B(x) \times B(x)^{-1} = 1 \bmod G(x)$인 경우 $B(x)^{-1}$을 $B(x)$의 역원이라 한다.

다항식기저를 사용하는 유한체의 역원 연산은 대부분 EEA을 기본으로 한다. 유한체 $GF(2^m)$의 임의의 원소 $A(x)$의 역원은 식 (1)을 만족하는 다항식 $S(x)$와 다항식 $T(x)$가 유일하게 존재함을 이용하여 계산한다.

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

식 (1)에서 다항식 $D(x)$는 $GCD(A(x), G(x))$이며, $D(x) = 1$이면 $S(x) \times A(x) + T(x) \times G(x) = 1$로 $S(x) \times A(x) \equiv 1 \bmod G(x)$이다. 따라서 $S(x) \equiv A(x)^{-1} \bmod G(x)$이므로, $S(x)$은 $A(x)$의 역원이다.

2. 유한체 GF(2^m)에서의 EBGA

EBGA는 이진 연산에 적합하며 나눗셈이 없는 쉬프

알고리즘 1.GF(2m)에서의 EBGA [1]

Input : $A(x) \in GF(2^m)$, $G = G(x)$.
Output : $A(x)^{-1} \bmod G(x)$.

1. $U = A(x)$, $V = G(x)$, $R = 1$, $S = 0$.
2. While $U \neq 1$ do the following:
3. If $u_0 = 0$ then
4. If $r_0 = 0$ then
5. $U = U/x$, $R = R/x$.
6. Else
7. $U = U/x$, $R = (R+G)/x$.
8. Else If $v_0 = 0$ then
9. If $s_0 = 0$ then
10. $V = V/x$, $S = S/x$.
11. Else
12. $V = V/x$, $S = (S+G)/x$.
13. Else
14. If deg(U)>deg(V) then
15. $U = U+V$, $R = R+S$.
16. Else
17. $V = U+V$, $S = R+S$.
18. Return (R).

트 연산과 같은 간단한 연산을 통해 역원을 계산한다. EBGA는 유한체 $GF(2^m)$의 임의의 다항식 $U(x)$와 다항식 $V(x)$에 대해 다음 성질을 이용하여 구성한다.

- 만약 다항식 $U(x)$와 $V(x)$가 x로 나누어지면, $GCD(U(x), V(x)) = x \times GCD(U(X)/x, V(x)/x)$이다.
- 만약 다항식 $U(x)$는 x로 나누어지나 다항식 $V(x)$는 x로 나누어지지 않으면, $GCD(U(x), V(x)) = GCD(U(X)/x, V(x))$이다.
- 만약 다항식 $U(x)$와 $V(x)$가 x로 나누어지지 않으면, $GCD(U(x), V(x)) = GCD((U(X)+V(x))/x, V(x))$이다.

알고리즘 1은 $A(x)^{-1} \bmod G(x)$을 계산하기 위한 EBGA를 나타낸 것이다. 알고리즘 1은 식 (2)와 식 (3) 사이의 연산을 기본으로 하며, 각 변수는 $U = A(x)$, $V = G(x)$, $R = 1$, $S = 0$로 초기화된다. 알고리즘 1은 반복수행의 각 단계마다 식 (2)와 식 (3)을 만족한다.

$$R \times A(x) \equiv U \bmod G(x) \quad (2)$$

$$S \times A(x) \equiv V \bmod G(x) \quad (3)$$

식 (1)에 따라 $GCD(A(x), G(x)) = 1$이면 식 (2)에서 $U = 1$로 $R \times A(x) \equiv 1 \bmod G(x)$이며 변수 R은 $R = A(x)^{-1} \bmod G(x)$로 $A(x)$의 역원이다.

알고리즘 1의 $G(x)$의 차수는 항상 m으로 변수 U와 V의 차수는 m을 넘지 않는다. 따라서 알고리즘 1의 반복문 실행횟수는 $2m$번을 넘지 않는다.

알고리즘 1은 나눗셈 연산이 없는 고정된 쉬프트 연산과 덧셈 연산, 비교 연산으로 구성되므로, 연산이 고정되어 있지 않은 EEA보다 EBGA가 하드웨어 구현에 더 적합하다. 그러나 $deg(U)$와 $deg(V)$의 비교 결과에 따라 다음 연산이 결정되므로 이 연산에 따른 알고리즘 지연이 발생할 수 있다.

III. 유한체 GF(2^m)에서 수정된 역원 알고리즘

유한체 $GF(2^m)$에서의 EBGA는 입력된 값에 따라 알고리즘 수행횟수가 결정되어 하드웨어 구현 시 어려운 점이 있다[2],[3],[4]. Wu et. al.는 [2], [3]에서 알고리즘 1을 기반으로 수행횟수를 $2m$으로 고정시킨 역원 알고리즘을 제안하고, 하드웨어로 시스톨릭 어레이 구조로 설계하였다. Watanabe et. al.은 [4]에서 알고리즘 1의 비교 연산을 효율적으로 처리하기 위해 쉬프트 레지스터와 이 레지스터의 최하위 비트 (LSB)을 이용한다. 그러나 큰 유한체 차수 m이 요구되는 시스템 환경인 경우 시스톨릭 어레이 구조의 하드웨어 공간복잡도는 증가하며, 더 많은 하드웨어 리소스가 요구된다. 따라서 이 구조는 경량화와 저전력이 요구되는 환경에 적합하지 않다.

본 논문에서는 유한체 $GF(2^m)$에서의 알고리즘 1을 기반으로 적은 공간복잡도를 요구하는 환경에 적합한 구조를 가지는 수정된 고속 역원 알고리즘을 제시한다. 본 논문에서 제시하는 알고리즘은 기존의 알고리즘의 수행횟수인 $2m$번보다 적은 수행횟수를 가지며, 기존의 알고리즘보다 적은 덧셈과 감산 연산으로 역원을 계산한다. 알고리즘 2는 본 논문에서 제시하는 유한체 $GF(2^m)$에서의 수정된 역원 알고리즘이다.

알고리즘 2는 식 (2)과 식 (3) 사이의 연산에 의해 역원이 결정되며, 식 (1)에 따라 $GCD(A(x), G(x)) = 1$이면 $U = 1$로, 변수 R이 $R = A(x)^{-1} \bmod G(x)$으로 $A(x)$의 역원이다.

알고리즘 2의 입력 $G(x)$의 차수는 항상 m이고, 입력 $A(x)$의 차수는 최대 $(m-1)$이다. 기존의 역원 알고리즘은[1],[2],[3],[4] 반복수행의 각 단계에서 다항식의 차수를 1씩 감소시키므로, 변수 U나 V는 1비트씩 감소한다. 따라서 $A(x)$의 역원을 계산하기 위한 알고리

알고리즘 2.GF(2m)에서의 수정된 고속 역원 알고리즘

Input : $A(x) \in GF(2^m), kG = kG(x), k = 0, 1, x, 1+x.$
Output : $A(x)^{-1} \bmod G(x)$

STEP 1. $U = A(x)$, $V = G(x)$, $R = 1$, $S = 0$,
$state = 0$, $\delta = -1$
STEP 2. $t = u_1 - v_1$, $V = t \cdot (xV)$, $S = S + t \cdot (xS)$
STEP 3.
1. While $U \neq 1$ do the following:
2. If $state = 0$ then
3. If $[(u_1, u_0) = (0,0)]$ then
4. $.U = U/x^2$, $R = (R + kG)/x^2$, $\delta = \delta - 2$
5. Else If $u_0 = 0$ then
6. $.U = U/x$, $R = (R + kG)/x$, $\delta = \delta - 1$
7. Else
8. If $\delta < 0$ then
9. $U \leftrightarrow V$, $R \leftrightarrow S$, $\delta = -\delta$
10. $U = U + V$
11. If $[t = 0]$ and$[(r_1, r_0) = (s_1, s_0)]$ then
12. $R = R + S$, $state = 0$
13. Else
14. $.R = R + kG$, $state = 1$
15. Else /* state = 1 */
16. $.U = U + V$, $R = R + S$, $state = 0$
17. Return (R).

즘 반복 수행횟수는 최악의 경우$2m$이 소요된다. 알고리즘 2는 다항식의 차수를 최대 2씩 감소시켜 변수 U나 V를 최대 2비트씩 줄여가면서, 연산에 따른 알고리즘 지연을 방지하며 경량화 암호 시스템에 적합한 하드웨어를 구성하도록 한다.

만약 $u_0 = 0$이면, 알고리즘 2는 알고리즘 1과 동일한 연산을 수행하여 변수 U는 1비트 감소한다. 그러나 $(u_1, u_0) = (0,0)$인 경우, 변수 U는 알고리즘 2의 STEP 3.4를 통해 2비트 감소된다. 그 외의 경우는 알고리즘 2의 덧셈 연산을 수행하여 그 결과의 최하위 두 비트가 항상 $(0,0)$이 되도록 한다. 표 1은 덧셈의 결과인 변수 U의 (u_1, u_0)가 항상 $(0,0)$의 값을 가지도록 필요한 덧셈 연산을 나타낸 것이다. 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. $(x \cdot v_1, x \cdot v_0) = (1,0)$이므로, 표 1에 따라 $(u_1 + v_1 + x \cdot v_1, u_0 + v_0 + x \cdot v_0)$은 $(0,0)$이다. 따라서 알고리즘 2는 $2m$보다 적은 반복 수행횟수로 임의의 원소 $A(x)$에 대한 역원 $A(x)^{-1} \bmod G(x)$을 계산한다. 표 1에서 제시된 덧셈 연산은 알고리즘 2의 STEP 3.7~STEP 3.16에서 수행된다.

표 1. 알고리즘 2에서 수행되는 덧셈연산
Table 1. Addition operations in Algorithm 2.

(u_1, u_0)	(v_1, v_0)	덧셈 연산	
(0, 1)	(0, 1)	$U = U + V$	$R = R + kG + S$
(0, 1)	(1, 1)	$U = U + V + xV$	$R = R + kG + S + xS$
(1, 1)	(0, 1)	$U = U + V + xV$	$R = R + kG + S + xS$
(1, 1)	(1, 1)	$U = U + V$	$R = R + kG + S$

표 2. 알고리즘 2의 STEP 3.14의 kG 결정 예
Table 2. An Example of determining kG in STEP 3.14 of Algorithm 2.

(r_1, s'_1)	(r_0, s'_0)	kG
(1, 0)	(0, 0)	xG
(1, 0)	(0, 1)	$(1+x)G$
(1, 0)	(1, 0)	0
(1, 0)	(1, 1)	G

알고리즘 2는 STEP 1을 통해 알고리즘 2의 각 변수를 초기화한다. STEP 3에서 연산의 반복수행으로 입력에 대한 역원을 계산한다. STEP 3에서 덧셈 연산은 변수 U와 V의 조건에 따라 연산한다. 알고리즘 2의 STEP 2는 STEP 3과 독립적으로 덧셈 연산을 수행하며 알고리즘 2의 STEP 3.16에서 이 결과를 이용한다. 각 STEP의 연산과 STEP 3의 조건판단은 병렬적으로 수행된다. 또한 STEP 3의 kG는 u_1, v_1, (r_1, r_0)와 (s'_1, s'_0)의 조건에 따라 사전에 결정되어 STEP 3.4나 STEP 3.6의 덧셈 연산을 수행한다. 표 2는 제시된 조건에 따라 STEP 3.14의 kG의 결정을 예로 나타낸 것이다.

알고리즘 2의 변수 U, V와 변수 R, S는 식 (2)과 식 (3)에 의해 밀접한 관계를 가지므로, 표 1에서 볼 수 있듯이 이 변수들은 동일한 연산이 적용된다. 그러나 알고리즘 1의 하드웨어 구현을 고려하면 변수 U, V의 값은 쉽게 결정이 되지만, 변수 R, S의 연산 과정에 지연이 발생할 수 있다. 즉 r_0의 값에 따라 감산 연산에 따른 추가적인 덧셈 연산이 필요하므로, 이 연산을 수행하는 동안 변수 U, V는 휴지 상태가 된다. 본 논문에서는 변수 U, V의 연산과 변수 R, S의 감산 연산의 비동기화에 따른 알고리즘 수행 지연을 방지하기 위해 동기화된 연산을 수행하도록 알고리즘 2를 구성한다. 동기화된 연산은 알고리즘 2의 상태 (State)에 따라 두 단계의 덧셈 연산을 수행으로 이루어진다. 알고리즘 2의 State 0이면 변수 U는 쉬프트 연산이나 덧셈 연산을 수행하며, u_1, v_1, (r_1, r_0)와 (s'_1, s'_0)의 조건에

표 3. 알고리즘 2의 STEP 3.4의 $\delta=\delta-2$ 연산
Table 3. $\delta=\delta-2$ in STEP 3.4 of Algorithm 2.

1. If $f=0$ and $d_0=0$ then
2. $D=D/x^2$
3. Else
4. $D=D\cdot x^2$, $f=0$

따라 변수 R의 덧셈 연산이 결정된다. 즉 State 0에서 변수 R은 감산 연산에 따른 덧셈 연산이나 변수 S와의 연산을 수행한다. 알고리즘 2의 State 0은 u_1와 v_1의 값에 따라 State 1로 변하며, State 1은 표 1에 제시된 덧셈 연산을 마무리한다. 알고리즘 2의 STEP 3.16에서 볼 수 있듯이 STEP 2에서 계산된 덧셈의 결과를 이용하며 State 0으로 변화한다. 따라서 감산 연산에 따른 덧셈 연산과 표 1에 제시된 덧셈 연산을 포함하여 알고리즘 2의 총 덧셈 연산의 수는 기존의 알고리즘에 비해 감소하였고, 알고리즘 2는 동기화된 연산을 수행한다.

알고리즘 1은 $\deg(U)$와 $\deg(V)$을 비교하는 연산이 필요하므로, 하드웨어 구현 시에 변수 U와 V의 차수를 반드시 저장하는 레지스터가 요구된다. 또한 m이 증가하면 이 비교 연산에 따른 알고리즘 수행의 지연이 발생한다. 이를 개선하기 위해 본 논문에서는 $|U|\le x^{\alpha}$와 $|V|\le x^{\beta}$을 만족하는 α와 β를 통해 α와 β의 차인 변수 δ을 이용한다[5]. 여기서 변수 δ가 $\delta<0$이면 $\deg(U)<\deg(V)$을 의미한다. 변수 δ는 $|\delta\le m$이므로, δ의 크기와 부호를 나타내는 변수들로 표현할 수 있다. 따라서 $(m+1)$비트 1-hot counter인 D와 1비트인 f를 이용한다. 여기서 1-hot counter는 한 비트만 1이고 나머지 비트는 모두 0의 값을 가진다[5].

f는 δ의 부호를 의미하는 것으로, $\delta>0$이면 $f=0$이고, 그 외의 경우 $f=1$이다. d의 최하위 비트를 d_0라고 하면, $\delta=0$인 경우 $d=1$이고 $d_0=1$이며, $\delta\neq0$인 경우 $d>1$이고 $d_0=0$이다. 따라서 $\delta<0$인지 판단하려면 $d_0=0$와 변수 $f=1$인지 확인한다. 알고리즘 2의 $\delta=-\delta$는 f=0으로 처리하며, 알고리즘 2의 STEP 3.4의 $\delta=\delta-2$은 표 3와 같이 제어한다.

표 4은 $GF(2^4)$에서의 $m=4$, $G(x)=x^4+x+1$, $A(x)=x^3+x$을 예제로 본 논문에서 제안한 알고리즘 2를 나타낸 것이다. 표 3에 따르면 6번 반복문 수행 후 변수 R이 (x^3+x^2)로 $A(x)$의 역원이다.

Ⅳ. 제안하는 역원 알고리즘의 하드웨어 구조

Wu et. al., Kim et. al., Takagi et. al.은 [2], [3], [4], [5]에서 $2m$으로 반복횟수가 고정된 유한체 $GF(2^m)$에서의 역원 알고리즘을 기반으로 시스톨릭 어레이 구조가 제안되었다. 그러나 유한체 차수 m이 증가하는 경우 하드웨어 복잡도는 증가하므로 경량화가 요구되는 환경에는 적합하지 않다. Lorenzo et. al.은 [6]에서 유한체 $GF(p)$에서의 새로운 역원 알고리즘을 제안하고, 이에 적합한 하드웨어를 제시하였다. Lorenzo가 제안한 역원 알고리즘은 평균 $2m$번의 반복 수행을 하며 알고리즘의 제어를 위해 비트 연산이 요구된다. 또한 하드웨어 구조는 감산 연산에 따른 알고리즘 수행 지연이 발생하며 제어 로직이 복잡하다는 단점을 가진다. 본 장에서는 알고리즘 2를 바탕으로 경량화된 암호 시스템 적합하도록 적은 복잡도를 가지면서, 동기화된 연산을 수행하는 새로운 하드웨어 구조를 제안한다. 본 논문에서 제안한 역원기는 유한체 $GF(p)$와 $GF(2^m)$에서 모두 수행이 가능한 통합된 구조이며, 이는 Ⅵ장에서 자세히 살펴본다.

알고리즘 2의 연산은 덧셈 연산과 오른쪽 쉬프트 연산, 교환 연산으로 구성된다. 변수 U와 V, 변수 R와 S는 독립된 연산을 수행하므로 분리하여 하드웨어를 설계한다. 또한 제어 로직은 변수 U의 비트인 u_1과 u_0에 따라 덧셈기 (Adder)가 덧셈 연산을 수행하도록 제어하며, δ의 조건에 따라 교환 연산을 결정한다.

그림 1은 본 논문에서 제안하는 적은 복잡도를 가지는 알고리즘 2의 하드웨어 블록 구조이다. 그림 1과 같이 알고리즘 2의 하드웨어 구조는 UV-Block과 RS-Block, 그리고 제어 로직 블록 (Control logic block)으로 구성된다. 그림 1의 UV-Block은 알고리즘 2의 변수 U와 V를, RS-Block은 알고리즘 2의 변수

표 4. 알고리즘 2의 GF(2^4)에서의 예제
Table 4. An Example of Inversion in GF(2^4) by Algorithm 2.

index	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^4+x^3+x	x^2+1	x^3	x^3+1	2
3	x^2+x+1	x^2+1	x	x^3+1	0
4	x^3	x^2+1	x^3+x	x^3+1	0
5	x	x^2+1	x^3+x+1	x^3+1	-2
6	1	x^2+1	x^3+x^2	x^3+1	-3

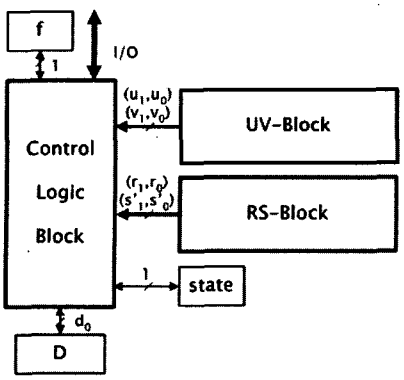

그림 1. 알고리즘 2의 하드웨어 구조
Fig. 1. Hardware Architecture of Algorithm 2.

R와 S에 대한 연산을 수행한다. 제어 로직 블록은 $(m+1)$비트인 1-hot counter D와 1비트 f에 의해 교환 연산을 수행하는 등 변수의 값에 따라 알고리즘 2의 연산을 제어한다. 또한 그림 1에서와 같이 제어 로직 블록은 1-hot counter D와 1비트 f, 1비트 *state*에 의해 제어하므로, [6]에 제안된 제어 로직에 비해 단순하고 효율적으로 수행할 수 있다.

그림 2와 그림 3은 각각 UV-Block과 RS-Block의 하드웨어 구조이다. 그림 2와 그림 3에서와 같이 $n=(m+1)$비트인 레지스터들인 $RegU$, $RegV$, $RegR$과 $RegS$을 사용한다. 각 레지스터는 알고리즘 2의 변수 U와 V, 변수 R과 S를 의미한다. 따라서 임의의 원소 $A(x) \in GF(2^m)$에 대한 역원을 계산하기 위해 각 레지스터는 다음과 같이 초기화된다.

$$RegU=A(x),\ RegV=G(x),\ RegR=1,\ RegS=0$$

$RegU$의 하위 2비트인 (u_1, u_0)에 의해 알고리즘 2를 수행한다. 알고리즘 2의 STEP 3.7에서 STEP 3.16까지 수행을 살펴보면, $state=0$이고 u_0와 v_0가 모두 1이면 그림 2의 UV-Block에서 $RegU$과 $RegV$의 덧셈 연산이 시작된다. 그림 3의 RS-Block의 $RegR$은 (r_1, r_0)와 (s'_1, s'_0)에 따라 V절에서 고려한 바와 같이 감산 연산을 수행하는 덧셈 연산이나 $RegS$을 더하는 덧셈 연산이 수행된다. 식 (4)와 식 (5)는 그림 2와 그림 3에서 수행하는 덧셈 연산을 나타낸 것이다.

$$U=(U+V+xV)/x^2=\{(U+V)/x+V\}/x \quad (4)$$

$$R=(R+kG+S+xS)/x^2=\{(R+kG)/x+(S/x+S)\}/x \quad (5)$$

그림 2의 UV-Block는 $state=0$일 때 식 (4)의 $(U+V)/x$을 수행하여 그 결과를 $RegU$에 저장하고, 식 (4)에서 $\{(U+V)/x+V\}/x$를 수행하기 위해 $state=1$이 된다. 따라서 $state=1$일 때 $RegU$에 저장되어 있는 부분합과 $RegV$의 V를 더하고 오른쪽 쉬프트 연산을 실행하여 $RegU$에 저장한다. 그림 2의 UV-Block의 연산을 수행하는 동안 그림 3의 RS-Block은 감산 연산에 따른 덧셈 연산을 수행한다. 그림 3에서 볼 수 있듯이 사전 계산된 kG, $k=0, 1, x, (x+1)$을 이용한다. 변수 k는 (r_1, r_0)와 (s'_1, s'_0)의 값에 따라 그림 1의 제어 로직 블록에서 결정한다. 즉 그림 3의 RS-Block은 $state=0$일 때 식 (5)의 $(R+kG)/x$을 연산한 후 $RegR$에 저장한다. 또한 알고리즘 2의 STEP 2는 STEP 3와 병렬적으로 수행하므로, $state=0$의 덧셈 연산을 수행하면서 Adder Ⅲ는 식 (5)의 $(S/x+S)$을 연산한다. $state=1$일 때 $RegR$의 부분합과 이미 계산된 $(S/x+S)$를 더한 후 오른쪽 쉬

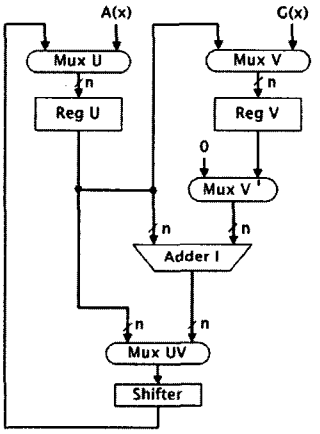

그림 2. 그림 1의 UV-Block
Fig. 2. UV-Block in Fig. 1.

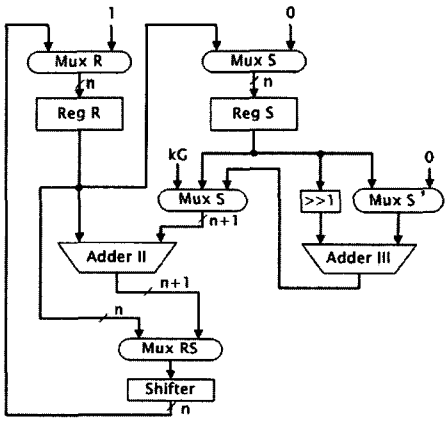

그림 3. 그림 1의 RS-Block
Fig. 3. RS-Block in Fig. 1.

프트 연산을 실행하여 식 (5)가 연산된다.

기존에 제안된 하드웨어 구조는 $2m$번의 알고리즘을 수행하면서 RS-Block의 감산 연산을 위해 추가적인 덧셈 연산을 실행하였다. 즉 RS-Block에서 2개 이상의 덧셈기를 통해 반복문 수행 시 감산 연산을 실행하거나 1개의 덧셈기를 이용하여 2번의 덧셈 연산을 수행하였다. 전자의 경우인 [7]에서는 최대지연경로가 증가하며 RS-Block 연산 시 매 회 3번의 덧셈 연산이 발생한다. 후자의 경우인 [6]에서는 RS-Block에서 감산 연산의 수행으로 UV-Block과의 동기화된 연산을 실행할 수 없으며 알고리즘 수행에 따른 지연이 발생한다. 본 논문에서 제안하는 역원기는 $2m$보다 적은 알고리즘을 수행하면서 감산 연산에 따른 알고리즘 지연을 방지하며, UV-Block와 RS-Block의 동기화된 연산 수행이 가능하다. 그림 3과 같이, [6]에 비해 덧셈기가 추가되었으나, 기존의 것보다 감산 연산의 수는 줄어든다. 또한 알고리즘 수행 시 감산 연산이 필요한 경우에만 추가적인 덧셈을 수행하므로 감산 연산을 포함한 총 덧셈 연산수는 감소하였다.

V. 시뮬레이션 결과

본 논문에서는 [8]의 타원곡선 환경과 기약 다항식을 기반으로 각 역원 알고리즘에 대해 $GF(2^m)$의 임의의 원소에 대한 역원 연산을 1,000,000번 테스트하였다. 표 5는 테스트한 결과를 나타낸 것이다. 본 논문에서 수행한 테스트는 감산 연산까지 포함한 평균 덧셈 연산 횟수, 평균 감산 연산 횟수, 평균 알고리즘 수행횟수 그리고 비트당 평균 알고리즘 수행횟수이다. 또한 MIA는 Phase I과 Phase II를 모두 고려하여 연산 횟수를 나타내었다.

알고리즘 2의 다항식 U의 차수는 알고리즘 한 번 수행 시 최대 2씩 감소하므로, 표 5에서 볼 수 있듯이 알고리즘 2는 알고리즘 1의 평균 수행횟수에 비해 약 18.8% 감소하여 역원 연산을 수행한다. 알고리즘 2의 평균 수행횟수는 [2], [3]에서 제안된 알고리즘에 비해 약 9.4%, MIA보다 약 45.9% 감소하였다. 또한 표 4에서 제시된 바와 같이 알고리즘 2는 비트당 평균 수행횟수를 비교하면 최소 1.78이며 최대 1.81이다.

알고리즘 2의 평균 덧셈 연산 횟수는 [2], [3]보다 약 34.7%, MIA보다 약 37.3% 감소하였다. 또한 알고리즘 2의 평균 감산 연산 횟수는 [2], [3]보다 약 37.3% 감소하였다. 따라서 알고리즘 2는 [6]에 비해 하드웨어 구현상 덧셈기가 추가되나, 총 덧셈 연산수는 기존의 알고리즘에 비해 감소한다.

VI. GF(p)와 GF(2^m)에서의 통합 역원기

역원 연산은 유한체 $GF(p)$와 $GF(2^m)$에서 정의된다. 유한체 $GF(2^m)$에서의 역원 구조는[2],[3],[9],[10],[11] 유한체 $GF(p)$에서 수행하는 어플리케이션에 적용할 수 없으므로, 유한체 $GF(p)$에서 수행하는 새로운 역

표 5. 역원 알고리즘 수행에 따른 시뮬레이션 결과
Table 5. The Results of the average computational cost in Inversions.

유한체	알고리즘	평균덧셈 연산횟수	평균감산 연산횟수	평균 수행횟수	평균수행 횟수/bit
$GF(2^{113})$	Algo. 2	144.1	68.4	201.6	**1.78**
	Algo. 1	184.4	91.3	277.6	2.46
	[2]	219.7	108.6	222.4	1.97
	MIA	186.6	93.5	372.6	3.30
$GF(2^{131})$	Algo. 2	168.8	81.2	234.6	**1.79**
	Algo. 1	216.8	108.7	324.6	2.48
	[2]	258.5	129.4	256.7	1.96
	MIA	216.6	108.4	432.5	3.30
$GF(2^{163})$	Algo. 2	210.9	101.4	293.2	**1.80**
	Algo. 1	269.7	135.0	404.6	2.48
	[2]	322.0	161.2	320.5	1.97
	MIA	269.9	135.1	539.1	3.31
$GF(2^{193})$	Algo. 2	248.9	117.0	348.8	**1.81**
	Algo. 1	314.9	155.0	474.6	2.46
	[2]	376.8	185.6	383.0	1.98
	MIA	319.9	160.1	639.1	3.31
$GF(2^{233})$	Algo. 2	293.4	124.3	421.5	**1.81**
	Algo. 1	351.5	158.8	544.3	2.34
	[2]	427.2	196.1	459.3	1.97
	MIA	385.8	193.1	770.6	3.31
$GF(2^{239})$	Algo. 2	289.7	102.4	433.0	**18.1**
	Algo. 1	319.6	121.8	517.3	2.16
	[2]	397.2	160.1	477.0	2.00
	MIA	395.8	198.1	790.3	3.31
$GF(2^{283})$	Algo. 2	369.0	176.8	513.6	**1.81**
	Algo. 1	468.7	234.2	703.2	2.48
	[2]	561.8	280.6	565.4	2.00
	MIA	469.3	234.8	938.8	3.32
$GF(2^{409})$	Algo. 2	523.8	231.4	746.8	**1.83**
	Algo. 1	638.5	298.9	977.0	2.39
	[2]	772.3	365.6	818.1	2.00
	MIA	679.9	339.8	1360.2	3.33
$GF(2^{571})$	Algo. 2	750.7	358.8	1039.5	**1.82**
	Algo. 1	950.4	475.0	1420.8	2.49
	[2]	1139.7	569.6	1146.8	2.01
	MIA	950.2	474.8	1903.4	3.33

표 6. 역원기에 대한 하드웨어 복잡도와 제어 로직에 대한 비교
Table 6. Comparison Hardware Complexity and Control Logics with Previously Architectures and the Proposed Inverter.

	제안된 역원기	[6]	[7]	[12]
유한체	$GF(p)$, $GF(2^m)$	$GF(p)$	$GF(p)$, $GF(2^m)$	$GF(p)$
최대지연 경로	$T_{RShift}+T_{Add}+T_{Mux_3}+2T_{Mux_2}$	$T_{RLShift}+T_{AddSub}+T_{Mux_3}+T_{Mux_2}$	$T_{RShift}+T_{AddSub}+T_{Mux_3}+3T_{Mux_2}+T_{Demux}$	$T_{RShift}+T_{AddSub}+T_{Mux_5}+T_{Mux_3}$
기본 하드웨어 구성 및 개수	Right Shifter : 2 $(m+1)$-bit Register : 4 2-input Add : 3 2-to-1 Mux : 7 3-to-1 Mux : 1	Right/Left Shifter : 2 $(m+1)$-bit Register : 3 $(m+2)$-bit Register : 1 2-input Add/Sub : 2 2-to-1 Mux : 2 3-to-1 Mux : 3	Right Shifter : 3 m-bit Register : 4 2-input Add : 1 2-input Sub: 2 2-to-1 Mux : 9 3-to-1 Mux : 4	Right Shifter : 2 $(m+1)$-bit Register : 4 3-input Add/Sub : 1 2-to-1 Mux : 2 3-to-1 Mux : 2 5-to-1 Mux : 2
제어 로직의 구성 및 개수	1-hot Counter : 1 Flag Register : 1	Sign Bit : 2 Operation Bit : 1 $(\log_2(m+1))$-bit Counter : 2 $(\log_2 m)$-bit Register : 1 Single-bit Flip-Flop : 2	$(\log_2 m)$-bit Counter : 1	Sign Bit : 2 Operation Bit : 1 Carry Bit : 1

T_{Add} : 덧셈기에서 발생하는 지연시간
T_{AddSub} : 덧셈/뺄셈기에서 발생하는 지연시간
T_{Mux_2} : 2개 입력 먹스 (2-input mux)에서 발생하는 지연시간
T_{Mux_3} : 3개 입력 먹스 (3-input mux)에서 발생하는 지연시간
T_{Mux_5} : 5개 입력 먹스 (5-input mux)에서 발생하는 지연시간
T_{Demux} : 디먹스 (Demux)에서 발생하는 지연시간
T_{RShift} : 오른쪽 쉬프터 (Right shifter)에서 발생하는 지연시간
$T_{RLShift}$: 오른쪽/왼쪽 쉬프터 (Right/Left shifter)에서 발생하는 지연시간

원 알고리즘이 요구된다. 따라서 유한체 $GF(p)$와 $GF(2^m)$에서 모두 수행이 가능한 통합된 역원 알고리즘과 하드웨어 구조가 필요하다. 알고리즘 2는 유한체 $GF(p)$에서 효율적으로 수행하는 덧셈기 (예. Carry-save adder, Carry look-ahead adder 등)를 이용하면 유한체 $GF(p)$에서의 역원기로 확장이 가능하다. 즉 알고리즘 2는 유한체 $GF(p)$와 $GF(2^m)$의 통합된 역원 알고리즘으로 확장이 가능하며 두 유한체에서의 역원 연산을 수행한다. 표 6는 기존의 역원기와 본 논문에서 제안한 역원기의 복잡도 및 제어 로직에 대해 비교한 것이다. 본 논문에서 제안한 역원기의 최대 지연 경로는 $T_{RShift}+T_{Add}+T_{Mux_3}+2T_{Mux_2}$로 기존의 역원기와[6], [11] 거의 동일한 지연 경로를 가진다. 본 논문에서 제안한 역원기는 [6]에서 제안한 구조에 비해 단순한 제어 로직을 가지고 있다. 또한 [11]에서 제안한 역원기는 1개의 덧셈기를 이용하여 역원 연산을 수행하나 본 논문에서 제안한 역원기는 Ⅳ절의 그림 2와 그림 3에서 볼 수 있듯이 병렬 처리가 가능하며 동기화된 연산을 수행한다.

Ⅶ. 결 론

본 논문에서는 알고리즘 1을 기반으로 유한체 $GF(2^m)$에서의 수정된 고속 역원 알고리즘을 제시하고, 경량화된 암호 시스템에 적용이 가능한 새로운 하드웨어 구조를 제안하였다. 본 논문에서 수정된 역원 알고리즘은 덧셈 연산과 쉬프트 연산을 이용하여 역원 연산을 계산하며, 기존의 알고리즘의 수행횟수인 $2m$보다 적게 수행한다. 본 논문에서 제시한 역원 알고리즘은 EBGA보다 18.8%, MIA보다 45.9% 적은 수행횟수를 가진다. 또한 본 논문에서 제안한 하드웨어 구조는 경

량화와 저전력이 요구되는 암호 시스템에 적합하도록 적은 복잡도를 가지며 [6]과 비교하면 동기화된 연산을 수행한다. 또한 유한체 $GF(2^m)$에서의 역원을 계산하기 위해 기존의 역원 구조보다[1],[2],[3],[4] 적은 덧셈 연산과 감산 연산을 포함하고 있다.

본 논문에서 제안된 역원 알고리즘은 유한체 $GF(p)$와 $GF(2^m)$에서의 통합된 역원 알고리즘으로 확장이 가능하여 공개키 암호 시스템의 효율성을 높일 수 있다. 또한 본 논문에서 제안된 하드웨어 구조는 처리 능력과 에너지 가용 능력이 제한된 환경인 스마트 카드, 토큰 하드웨어, 휴대용 단말기 등의 장비에 효과적으로 적용될 수 있다.

참 고 문 헌

[1] Y. Watanabe, N. Takagi, and K. Takagi, "A VLSI Algorithm for Division in GF(2m) Based on Extended Binary GCD Algorithm", IEICE Trans. Fundamentals, vol. E85-A, May 2002, pp. 994-999

[2] C. H. Wu, C. M. Wu, M. D. Shieh, and Y. T. Hwang, "Systolic VLSI Realization of a Novel Iterative Division Algorithm over GF(2^m): a High-Speed, Low-Complexity Design", 2001 IEEE International Symposium on Circuits and Systems, May 2001, pp.33-36.

[3] C. H. Wu, C. M. Wu, M. D. Shieh, and Y. T. Hwang, "An Area-Efficient Systolic Division Circuit over GF(2^m) for Secure Communication, 2002 IEEE International Symposium on Circuits and Systems, August 2002, pp.733-736

[4] C. H. Kim, S. H. Kwon, J. J. Kim, C. P. Hong, "A Compact and Fast Division Architecture for a Finite Field", ICCSA, LNCS 2667, 2003, pp.855-864.

[5] N. Takagi, C. K. Koc, "A VLSI Algorithm for Modular Division Based on the Binary GCD Algorithm", IEICE Trans. Fundamentals, vol. E81-A, May 1998, pp. 724-728.

[6] R. Lorenzo, "New Algorithm for Classical Modular Inverse", Cryptographic Hardware and Embedded Systems, CHES'02, LNCS 2523, 2002, pp.57-70.

[7] A. Gutub, A. F. Tenca, E. Savas, C. K. Koc, "Scalable and unified hardware to compute Montgomery inverse in GF(p) and GF(2^m)", CHES 2002, LNCS 2523, August 2002, pp.484-499.

[8] Certicom Research, "SEC 2: Recommanded Elliptic Curve Domain Parameters", version 1.0, September 2000.

[9] Z. Yan, D. V. Sarwate, "New Systolic architect-ures for Inversion and Division in GF(2m)", IEEE Trans. on Computers, vol. 52, no. 11, ,November 2003, pp.1514-1519.

[10] J. H. Guo, C. L. Wang, "Hardware-efficient systolic architecture for inversion and division in GF(2m)", IEE Proc. Comput. Digital Tech. vol. 145, no. 4, 1998, pp.272-278.

[11] J. H. Guo, C. L. Wang, "Systolic Array Implem-entation of Euclid's Algorithm for Inversion and Division in GF(2^m)", IEEE Transactions on Computers, vol. 47, no.10, October 1998, pp.1161-1167.

[12] T. Zhou, X. Wu, G. Bai and H. Chen, "Fast GF(p) modular inversion algorithm suitable for VLSI implementation", Electronics Letters, Vol. 38, No 14, 2002, pp.706-707.

[13] D. Hankerson, J. L. Hernandez, A. Menezes, "Software Implementation of Elliptic Curve Cryptography Over Binary Fields", Cryptographic Hardware and Embedded Systems, CHES'00, 2000, pp.1-24.

[14] B. S. Kaliski Jr., "The Montgomery Inverters and Its Applications", IEEE Trans. on Computers, vol. 44, no. 8, August 1995, pp.1064-1065.

[15] E. Savas, C. K. Koc, "The Montgomery Modular Inverse-Revisited", IEEE Trans. on Computers, vol. 49, No. 7, July 2000, pp.763-766.

[16] A. Daly, W. Marnane, T. Kerins, E. Popovici, "Fast Modular Division for Application in ECC on Reconfigurable logic", FPL 2003, LNCS 2778, 2003, pp.786-795.

저 자 소 개

김 소 선(정회원)
2003년 이화여자대학교 수학과 학사.
2005년 고려대학교 정보보호 대학원 석사.
<주관심분야 : 공개키 암호, 암호칩 설계 기술>

김 창 한(정회원)
1985년 2월 고려대학교 수학과 학사.
1987년 2월 고려대학교 수학과 석사.
1992년 2월 고려대학교 수학과 박사.
2002년 2월~현재 세명대학교 정보보호학과 부교수
<주관심분야 : 정수론, 공개키 암호, 암호프로토콜>

장 남 수(학생회원)
2002년 서울시립대학교 수학과 학사.
2005년 고려대학교 정보보호 대학원 석사.
2005년~현재 고려대학교 정보보호대학원 박사과정.
<주관심분야 : 공개키 암호, 암호칩 설계 기술, 부채널 공격 방법론>