정규 기저를 이용한 $GF((2^n)^m)$에서의 효율적인 역원 알고리즘

장구영*, 김호원*, 강주성*

*한국전자통신연구원 정보보호연구본부

A fast inversion algorithm in $GF((2^n)^m)$ using normal basis

Ku-Young Chang*, Howon Kim*, Ju-Sung Kang*

*Information Security Research Division,

Electronics and Telecommunications Research Institute

요 약

본 논문은 기존의 정규 기저를 이용한 역원 알고리즘인 IT 알고리즘과 TYT 알고리즘을 개선한 $GF(q^m)^*$ ($q=2^n$)에서의 효율적인 역원 알고리즘을 제안한다. 제안된 알고리즘은 작은 n에 대해 $GF(q)^*$의 원소에 대한 역원을 선행 계산으로 저장하고, $m-1$을 몇 개의 인수와 나머지로 분해함으로써 역원 알고리즘에 필요한 곱셈의 수를 줄일 수 있는 방법이다. 즉, 작은 양의 데이터에 대한 메모리 저장 공간을 이용하여, $GF(q^m)^*$에서의 역원을 계산하는 데 필요한 곱셈의 수를 줄일 수 있음을 보여준다.

I. 서론

유한체 $GF(2^m)$은 에러 정정 코드(error-correcting code)나 암호학 등과 같은 다양한 응용 분야에서 사용되고 있다. 특히 $GF(2^m)$에서 정의된 타원곡선에 기반한 공개키 암호 시스템은 보통 m이 160 이상인 큰 유한체 위에서 구성되어 있다. 이러한 응용 분야에서는 $GF(2^m)$에서의 덧셈, 곱셈, 제곱, 곱셈에 대한 역원과 같은 유한체 연산에 대한 효율적인 알고리즘이 필요하다. 일반적으로 $GF(2^m)^*$에서의 역원은 다른 유한체 연산에 비해 시간이 많이 걸리는 연산으로 알려져 있다.

$GF(2^m)^*$에서의 역원을 계산하는 알고리즘으로는 EEA(Extended Euclidean Algorithm), AIA(Almost Inverse Algorithm), MAIA(Modified Almost Inverse Algorithm)과 정규 기저(normal basis)에서 페르마의 정리를 이용하는 방법 등이 알려져 있다.[2,3,6] 페르마의 정리에 의하면 임의의 $\beta \in GF(2^m)^*$에 대해 $\beta^{-1}=\beta^{2^m-2}$를 만족시키기 때문에, $GF(2^m)^*$에서의 β에 대한 역원을 구하기 위해 β^{-1}를 직접 계산하는 대신에 β^{2^m-2}를 계산한다. β^{2^m-2}는 제곱과 곱셈을 반복함으로써 계산 가능하고, 이때 필요한 계산 량은 $m-1$번의 제곱과 $m-2$번의 곱셈이다.

정규 기저에서의 원소의 제곱은 단순한 한번의 순환 우측 쉬프트(cyclic right shift)로 수행할 수 있기 때문에 빠른 역원 연산을 수행하기 위해서는 곱셈의 수를 줄이는 것이 중요하다. Itoh와 Tsujii[3]는 이러한 곱셈의 수를 기존의 $m-2$에서 $O(m)$으로 줄였다. Chang 등[6]은 $m-1$을 두 개의 인수 m_1과 m_2로 분해함으로써 IT 알고리즘을 보다 효율적으로 향상시켰다. 이 알고리즘의 효율성은 $m-1$의 인수분해에 의존하고, 특히 $m-1$이 소수인 경우는 적용할 수 없다. 최근에

Takagi 등[6]은 그러한 $m-1$에 대해 적용할 수 있도록 최적 분해(optimal decomposition)라는 방법을 이용한 알고리즘을 제안하였다.

본 논문에서, 우리는 정규 기저를 이용한 $GF(q^m)^*$ ($q=2^n$)에서의 역원을 계산하는 효율적인 알고리즘을 제안한다. 제안한 알고리즘에서는 작은 n에 대해 $GF(q)^*$에서의 역원을 선행 계산으로 저장하고, A^{r-1}을 계산하기 위해 $m-1$을 몇 개의 인수들과 작은 나머지 값으로 분해함으로써 필요한 곱셈의 수를 효율적으로 줄인다. 즉, 본 논문에서 제안하는 방법은 작은 량의 데이터를 메모리에 저장함으로써 역원을 계산하는 데 필요한 곱셈의 수를 줄이는 방법이다.

$GF(q)^*$에 대해 미리 계산하여 저장할 공간은 $n(2^n-1)$비트로써, $n=8$이하인 경우는 하드웨어 구현 시 충분히 저장할 수 있는 양이다. 예를 들어, $GF(2^{384})^*$인 경우 IT 알고리즘[3]은 15번의 곱셈을 필요로 하고, TYT 알고리즘[6]은 13번의 곱셈이 필요하다. 한편, 제안된 알고리즘은 $q=2^4$, $m=96$으로 놓고, $m-1=95=19\times5$로 분해한다. 이때, 우리는 선행 계산으로 저장된 $4(2^4-1)$비트를 이용하여 곱셈의 수를 11번으로 줄일 수 있다.

한편 Smart[5]는 차수가 합성수인 어떤 확장체에서의 ECDLP 문제를 Weil Descent 방법을 이용하여, 기존의 Pollard rho 방법에 비해 효율적으로 해결할 수 있음을 보였다. 특히 차수가 4에 의해 나누어지는 확장체는 피하도록 권고하고 있다. 그러나, Smart에 의해 분석된 $GF(2^{155})^*$ 위에서의 ECDLP는 Pollard rho 방법에 비해 효율적으로 분석할 수 있지만, 현재의 기술 수준에서는 안전한 것으로 보인다. 이러한 결과 때문에 차수가 합성수인 유한체 위에서의 타원 곡선 암호는 그 안전성에 의심을 받게 되었고, 현재는 잘 사용하고 있지 않은 실정이다. 그러므로 본 논문에서 제안한 $GF(q^m)^*$에서의 역원 계산 알고리즘은 타원 곡선 암호에 적용하기는 적당하지 않은 것으로 사료된다. 하지만 향후 다양한 암호 응용에서 차수가 합성수인 유한체의 사용을 전혀 배제할 수는 없으며, 이에 따라 여기에서 제안한 $GF(q^m)^*$에서의 역원 계산 알고리즘에 대한 연구도 필요한 것으로 보인다.

II. 기존에 제안된 역원 알고리즘

정규 기저를 이용하여 $GF(2^m)^*$에서의 역원을 계산하는 기존의 알고리즘에 대해 간략하게 살펴보자.

[정의 1] $\alpha\in GF(2^m)^*$일 때, $GF(2)$ 위에서의 $GF(2^m)$에 대한 기저 $\{\alpha, \alpha^2, \alpha^{2^2}, \cdots, \alpha^{2^{m-1}}\}$를 정규 기저라고 부른다.

우리는 임의의 $x\in GF(2^m)$에 대해 $x=x_0\alpha^{2^0}+x_1\alpha^{2^1}+\cdots+x_{m-1}\alpha^{2^{m-1}}=\sum_{i=0}^{m-1}x_i\alpha^{2^i}$, $x_i\in GF(2)$로 표현할 수 있다. 이것을 벡터로 표현하면 $x=(x_0, x_1, \cdots, x_{m-1})$로 나타낼 수 있다. 그러면, $x^2=x_{m-1}\alpha^{2^0}+x_0\alpha^{2^1}+\cdots+x_{m-2}\alpha^{2^{m-1}}$이 되고, x^2을 벡터로 표현하면, $x^2=(x_{m-1}, x_0, x_1, \cdots, x_{m-2})$이기 때문에 제곱은 한 번의 우측 순환 쉬프트에 의해 계산할 수 있다. 우리는 앞으로 $q=2^n$으로 나타낸다.

[정리 1] $A\in GF(q^m)^*$라 하고 $t=1+q+q^2+\cdots+q^{s-1}$이라 하자. 그러면, A^t을 계산하기 위해 $len(s)+Hw(s)-2$번의 곱셈이 필요한 알고리즘이 존재한다. 여기에서 $len(s)$은 $\lfloor \log_2 s \rfloor+1$와 같고, $Hw(s)$는 s의 이진 표현에 대한 Hamming weight를 나타낸다.

[증명] [6]의 2장을 참고하면 쉽게 증명할 수 있다.

페르마의 정리에 의하면, 임의의 $\beta\in GF(2^m)^*$에 대해 $\beta^{-1}=\beta^{2^m-2}$을 만족한다. $2^m-2=2(1+2^1+2^2+\cdots+2^{m-2})$이기 때문에 $\beta^{-1}=(\beta^{2^0}\beta^{2^1}\beta^{2^2}\cdots\beta^{2^{m-2}})^2$을 만족한다. Itoh와 Tsujii[3]는 정리 1을 이용하여 $\beta^{1+2^1+2^2+\cdots+2^{m-2}}$을 계산하는 데 필요한 곱셈의 수가 $len(m-1)+Hw(m-1)-2$임을 보였다.

Chang 등[6]은 $m-1$을 두 개의 인수로 분해하여 각각의 인수 m_1, m_2에 정리 1을 적용시킴으로써 IT 알고리즘을 보다 효율적으로 향상시켰다. 이 때 필요한 곱셈의 수는 $\sum_{i=1}^{2}((len(m_i)+Hw(m_i)-2)$이다. 이 알고리즘은 $m-1$의 어떠한 값으로 인수분해 되느냐

에 의존하고, 특히 $m-1$이 소수인 경우는 적용할 수 없다. 최근에 Takagi 등[6]은 그러한 $m-1$에 대해 적용할 수 있는 알고리즘을 제안하였다. 이들이 제안한 TYT 알고리즘은 다음과 같은 사실에 기반한다:

$$2^m-2=2^{m-1}+2^{m-1}-2$$
$$=2^{m-1}+2^{m-2}+\cdots+2^{m-h}+2^{m-h}-2$$

이기 때문에, $\beta^{-1}=\beta^{2^m-2}$는 $\beta^{2^{m-1}}$부터 $\beta^{2^{m-h}-2}$까지의 h개의 곱셈으로 구할 수 있다. 그리고 $\beta^{2^{m-h}-2}$는 IT 알고리즘[3]이나 Chang 알고리즘[6]에 의해 계산할 수 있다. 또한, Takagi 등은 곱셈의 수를 최소로 하기 위한 h를 선택하기 위한 최적 분해라는 방법을 제시하였다. 이러한 최적의 h는 응용에 적용될 유한체 $GF(2^m)$이 선택되면 같이 결정되는 값으로 실제 역원 연산의 효율성을 감소시키는 요소가 아닐 뿐 아니라, $m-1$이 큰 값이 아니기 때문에 쉽게 찾을 수 있다.

III. 정규 기저를 이용한 $GF(q^m)$에서의 새로운 역원 알고리즘

[정리 2][1,3] $q=2^n$ 이라 하고, $A\in GF(q^m)^*$ 라고 하자. 그러면

$$A^{-1}=(A^r)^{-1}A^{r-1},\ A^r\in GF(q)$$

여기에서 $r=(q^m-1)/(q-1)=1+q+q^2+\cdots+q^{(m-1)}$ 이다.

정리 2는 임의의 $A\in GF(q^m)^*$에 대해 A^{-1}을 구하기 위해서, $GF(q^m)$에서 두 번의 곱셈 $A\cdot A^{r-1}$, $(A^r)^{-1}\cdot A^{r-1}$과 $GF(q^m)$에서의 지수승 연산 A^{r-1}, $GF(q)$에서의 역원 $(A^r)^{-1}$가 필요함을 나타낸다. 정리 2에 의해 $(A^r)^{-1}\in GF(q)$임을 알고 있다. 만약 n이 작다면, 우리는 $GF(q)^*$의 모든 원소들에 대한 역원을 미리 계산하여 look-up 테이블에 저장한 뒤 역원을 계산할 때마다 적당한 값을 호출하면 된다. 우리는 다음과 같은 방법[4]에 의해 $GF(q)^*$의 원소에 대한 역원을 저장할 수 있다.

$y=(y_0,y_1,\cdots,y_{nm-1})\in GF(q)\subset GF(q^m)$이면, $y^q=y^{2^n}=y$를 만족한다. 따라서 y는 $(y_0,y_1,\cdots,y_{n-1},y_0,y_1,\cdots,y_{n-1},\cdots,y_0,y_1,,\cdots,y_{n-1})$로 표현할 수 있다. 이것은 $GF(q)$에 있는 원소는 그 원소의 벡터 표현의 앞쪽 n비트만을 저장하면 충분함을 의미한다. 결국 $GF(q)^*$에 속해 있는 모든 원소들의 역원에 대해서 선행 계산하여 저장할 양은 $n(2^n-1)$비트이다. n이 8인 경우 저장량은 $8(2^8-1)$비트로써 255 바이트가 된다. 이 저장량은 하드웨어 구현 시에 충분히 저장할 수 있는 양이다. $A^{r-1}=A^{(1+q^2+\cdots+q^{m-2})q}$이 다 . $m-1=m_1m_2+h$라고 하자. 그러면

$$1+q+\cdots+q^{m-2}=$$
$$1+q+\cdots+q^{m_1m_2-1}+q^{m_1m_2}+\cdots+q^{m_1m_2+h-1}$$

이 된다. 따라서 $A^{1+q+\cdots+q^{m-2}}$는 $A^{q^{m_1m_2+h-1}}$부터 $A^{1+q^2+\cdots+q^{m_1m_2-1}}$까지 h번의 곱셈을 필요로 한다. 각각의 $A^{2^{m_1m_2+i}}(0\le i<h)$는 A의 순환 우측 쉬프트에 의해 수행할 수 있다. 이제 $A^{1+q^2+\cdots+q^{m_1m_2-1}}$을 고려하자.

$$1+q^2+\cdots+q^{m_1m_2-1}$$
$$=(1+q+q^2+\cdots+q^{m_1-1})+$$
$$q^{m_1}(1+q+q^2+\cdots+q^{m_1-1})+\cdots+$$
$$q^{m_1(m_2-1)}(1+q+q^2+\cdots+q^{m_1-1})$$
$$=(1+q+q^2+\cdots+q^{m_1-1})\times$$
$$(1+q^{m_1}+q^{2m_1}+\cdots+q^{m_1(m_2-1)})$$

따라서 정리 1에 의해 A^{r-1}은 $\sum_{i=1}^{2}((len(m_i)+Hw(m_i)-2)+h$번의 곱셈이 필요하다. 필요하다면 m_1, m_2의 양쪽 인수에 같은 과정을 반복할 수 있다. A^{r-1}에 필요한 곱셈의 수를 최소화하기 위한 h의 선택은 TYT 알고리즘[6]의 최적 분해 방법과 마찬가지로 할 수 있다. 결과적으로 $m-1=m_1m_2\cdots m_k+h$ 일 때, 제안한 알고리즘은 A^{-1}를 계산하기 위해서 $n(2^n-1)$비트의 선행 계산 공간과 $\sum_{i=1}^{k}((len(m_i)+Hw(m_i))+h+2$번의 곱셈

이 필요하다. 만약 $m-1$에 IT 알고리즘을 적용하면 $n(2^n-1)$비트의 선행 계산 공간과 $len(m-1)+Hw(m-1)-2+2$번의 곱셈이 필요하다.

한편, $n=1$인 경우에는 제안한 알고리즘이 TYT 알고리즘과 같음을 알 수 있다. 따라서 본 논문에서 제안한 알고리즘은 TYT 알고리즘의 일반화라고 볼 수 있다. 다음에 제시하는 두 개의 예들은 제안된 알고리즘의 효율성을 보여준다.

[예 1] $GF(2^{480})$에서의 역원을 고려하자. IT알고리즘은 15번의 곱셈을 필요로 한다. 반면에 TYT 알고리즘은 $480-1=479$를 $(34\times 7+1)\times 2+1$로 분해했을 때, 곱셈 양을 13번으로 줄일 수 있다. 제안된 알고리즘은 $q=2^4$, $m=120$로 놓고, $m-1=119=17\cdot 7$로 분해한다. 이때, 우리는 선행 계산으로 저장된 $4(2^4-1)$비트를 이용하여 곱셈의 수를 11번으로 줄일 수 있다. 만약 119에 IT 알고리즘을 적용하면, 선행 계산을 위한 $4(2^4-1)$비트 공간과 13번의 곱셈이 필요하다.

표 1. $n\le 8$일 때, $GF((2^n)^m)$에서의 곱셈의 수

nn	n	m-1	$m-1$의 분해	곱셈 수 (본 노문)	곱셈 수 (TYT)	곱셈 수 (IT)
128	8	15	3 × 5	8	10	12
256	8	31	10 × 3+1	9	10	14
320	8	39	3 × 13	9	12	14
384	4	95	19 × 5	11	13	15
416	8	51	17 × 3	9	12	14
448	8	55	11 × 5	10	12	15
480	4	119	17 × 7	11	13	15
512	8	63	9 × 7	10	12	16
608	8	75	17 × 5	10	13	15
640	8	79	26 × 3+1	11	13	16
704	8	87	29 × 3	11	13	16
736	8	91	13 × 7	11	12	16
768	8	95	19 × 5	11	14	17

[예 2] $GF(2^{2^{11}})$에서의 역원을 고려하자. IT알고리즘은 20번의 곱셈을 필요로 한다. 반면에 TYT 알고리즘은 $2^{11}-1=2047$을 $(10\times 3+1)\times 66+1$로 분해했을 때, 15번의 곱셈으로 줄일 수 있다. 제안된 알고리즘은 $q=2^8$, $m=2^8$로 놓고, $m-1=255=17\cdot 5\cdot 3$으로 분해한다. 이때, 우리는 선행 계산으로 저장된 $8(2^8-1)$비트를 이용하여 곱셈의 수를 12번으로 줄일 수 있다. 만약 255에 IT 알고리즘을 적용하면, 선행 계산을 위한 $8(2^8-1)$비트 공간과 16번의 곱셈이 필요하다.

표 1은 $GF((2^n)^m)=GF(2^{32k})$인 몇 개의 유한체에 대해 제안된 알고리즘에 필요한 최소의 곱셈 수와 이에 필요한 가장 작은 선행 계산 공간을 나타낸다. 표1의 n은 미리 계산해야 할 부분체 $GF(2^n)^*$를 나타낸다. 따라서 이 경우 우리는 $n(2^n-1)$비트를 저장한다.

IV. 결론

우리는 정규 기저를 이용한 $GF(q^m)$, $q=2^n$에서의 역원을 계산하기 위한 효율적인 방법을 제안하였다. 이 방법은 임의의 $A\in GF(q^m)^*$에 대하여 $A^{-1}=(A^r)^{-1}A^{r-1}$이라는 사실에 의존한다. 여기에서 $r=(q^m-1)/(q-1)=1+q+q^2+\cdots+q^{(m-1)}$이다. 이 논문에서는 두 개의 주요한 결과를 나타내고 있다. 우선 작은 n에 대해서 $A^r\in GF(q)$이기 때문에, 부분체 $GF(q)^*$에서의 역원에 대한 $n(2^n-1)$비트 크기를 갖는 look-up 테이블을 이용하여 $(A^r)^{-1}$를 계산할 수 있다. 두 번째로 정규 기저를 사용하여 $GF(q^m)$에서의 A^{r-1}을 계산하는 효율적인 방법을 제안하였다. 제안된 알고리즘은 표 1에서 알 수 있듯이 작은 양의 데이터에 대한 메모리 저장 공간을 이용하여, $GF(q^m)^*$에서의 역원 계산에 필요한 곱셈의 수를 줄일 수 있음을 보여준다.

참고문헌

[1] J. Guajardo and C. Paar, "Efficient Algorithms for Elliptic Curve Crypto systems", CRYPTO'97, LNCS 1294, pp. 342-356, Springer-Verlag, 1997.

[2] D. Hankerson, J. Lopez, and Alfred Menezes,

Software, " Software Implementation of Elliptic Curve Cryptography over Binary Fields", CHES, LNCS 1965, pp.1-24, 2000.

[3] T. Itoh and S. Tsujii, "A Fast Algorithm for Computing Multiplicative Inverse in $GF(2^m)$ Using Normal Basis" , Information and Computation, vol. 78, pp. 171-177, 1988.

[4] T. Itoh and S. Tsujii, "Effective Recursive Algorithm for Computing Multiplicative Inverse in $GF(2^m)$ ", Electronic Letters, vol. 24, no. 6, pp. 334-335, May 1988.

[5] Nigel P. Smart, "How secure are Elliptic curves over composite extension fields?", EUROCRYPT 2001, LNCS 2045, pp.30-39, Springer-Verlag, 2001.

[6] N. Takaki, J.-I. Yoshiki and K. Takaki, "A Fast Algorithm for Multiplicative Inversion in $GF(2^m)$ Using Normal Basis", IEEE Trans. Computers, vol. 50, no. 5, pp. 394-398, May 2001.