AMRITA
VISHWA VIDYAPEETHAM
University Established u/s 3 of UGC Act 1956
Amritapuri Campus

Computing the Modular Inverse of a Polynomial Function over GF(2^P) Using Bit Wise Operation

Seshagiri Prabhu N
Guide: Raphael Fourquet

April 27, 2013

Outline

AMRITA VISHWA VIDYAPEETHAM
University Established u/s 3 of UGC Act 1956
Amritapuri Campus

Introduction

1. Relevance of Modulo arithmetic in public key crypto system
2. The use of **E**xtended **E**uclidean **A**lgorithm (**EEA**) to evaluate the multiplicative inverse

Contribution of this paper

AMRITA VISHWA VIDYAPEETHAM
University Established u/s 3 of UGC Act 1956
Amritapuri Campus

Contribution of this paper

Computerized algorithm for the determination of the multiplicative inverse of a polynomial over GF(2^P) using simple bit wise shift and XOR operations.

Problem Description

EEA

Let $A(x)$ and $B(x)$ be polynomials.
EEA gives U and V such that
$\gcd(A, B) = U * A + V * B$

Note

If A is irreducible, then its gcd is 1, and we are only interested in V, which is the inverse of $B[modA]$

Polynomial representation

The finite field is a representative of a polynomial function with respect to one variable x:
$GF(2^p) = x^{p-1} + x^{p-2} + \ldots + x^2 + x^1$

Example

Finite field $GF(2^8) = x^8 + x^4 + x^3 + x + 1$
$53_{10} \rightarrow 1010011_2 \rightarrow (x^6 + x^4 + x + 1)$
The EEA of 53 on $GF(2^8)$ is $x^7 + x^6 + x^3 + x$

Proposed Algorithm

AMRITA VISHWA VIDYAPEETHAM
University Established u/s 3 of UGC Act 1956
Amritapuri Campus


```
procedure MULTIPLICATIVE INVERSE(A3[ ], B3[ ])
    C1 = A2 = B2 = 0;
    while (B3>1) do                                        ▷ Step 1 do
        Q = 0;
        Temp = B3;
        while (A3 > Temp || BitSize(C) ≥ BitSize(Temp)) do
            Q1 = 1;
            while (A3MSB == B3MSB) do
                B3 = B3 << LinearLeftShift ;
                Q1 = Q1 * 2;
            end while
            Q = Q + Q1;
            A3 = A3[ ] ⊕ B3[ ];
            B3 = Temp;
        end while
        A2 = B2; B3 = A3; A3 = Temp;
        N = BitSize(Q);                                    ▷ Binary Bit Size of Q
        Temp = B2; C2 = 0;                                 ▷ Step2
        while (N ≥ 1) do
            C2 = 0d;
            if (QN == 1) then                              ▷ Testing if Nth bit of Q is 1
                C1 = B2 << N − 1;                          ▷ Linear left shift by N-1 times
                C2 = C2 ⊕ C1;
            end if
            N − −;
        end while
        B2 = C2; A2 = Temp; B2 = B2 ⊕ A2;                  ▷ Multiplicative Inverse
    end while
```

Implementation of Algorithm

Let's apply EEA to $A = 283$ and $B = 42$

i	Operation	Binary	U	V
0	A	100011011	1	0
1	B	000101010	0	1
	$3 << B$	101010000	0	1000
2	$A \leftarrow A \oplus (3 << B)$	001001011	1	1000
	$1 << B$	001010100	0	0010
3	$A \leftarrow A \oplus (1 << B)$	000011111	1	1010
	$A < B \; A \rightleftharpoons B$			
	A	000101010	00	00001
	B	000011111	01	01010
	$1 << B$	000111110	10	10100
4	$A \leftarrow A \oplus (1 << B)$	000010100	10	10101
	$A < B \; A \rightleftharpoons B$			
	A	000011111	01	01010
	B	000010100	10	10101
5	$A \leftarrow A \oplus B$	000001011	11	11111
	$A < B \; A \rightleftharpoons B$			
	A	000010100	010	010101
	B	000001011	011	011111
	$1 << B$	000010110	110	111110
6	$A \leftarrow A \oplus (1 << B)$	000000010	100	101011
	$A < B \; A \rightleftharpoons B$			
	A	000001011	00011	00011111
	B	000000010	00100	00101011
	$2 << B)$	000001000	10000	10101100
7	$A \leftarrow A \oplus (1 << B)$	000000011	10011	10110011
8	$A \leftarrow A \oplus B$	000000001	10111	10011000

Conclusion

1. This algorithm can be easily extended for determining the elements of the S-Box used in *AES*.
2. This algorithm is efficient for determining the multiplicative inverse of polynomial over $GF(2^P)$

Future works

Possible future works

1. Optimize the algorithm
2. Comparative study with many existing algorithm
3. Implementation in hardware for real time applications

Questions ?

Seshagiri Prabhu
seshagiriprabhu@gmail.com