

# Multiplication trick in GF(2<sup>m</sup>)

---

*Source:* <http://www.derkeiler.com/Newsgroups/sci.crypt/2007-06/msg33345.html>

---

- *From:* Manuel Pancorbo Castro <[mpancorbo@xxxxxxxx](mailto:mpancorbo@xxxxxxxx)>
  - *Date:* Wed, 27 Jun 2007 00:25:13 +0200
- 

-----BEGIN PGP SIGNED MESSAGE-----

Hash: SHA1

I present a trick to perform efficiently the operation

$$R(x) = A(x)*B(x) \bmod P(x)$$

where A,B,R are polynomials in GF(2<sup>m</sup>)/P and P is the primitive trinomial

$$P = x^m + x^n + 1$$

where "m" and "n" are odd.

The goal is not to overflow the actual word size. The actual algorithms calculates  $R' = A*B$  and then reduces the result  $R = R' \bmod P$ ; so it is needed a double-sized record. On the other side, it can be made at bit level without overflow, but to the cost of too much shifts and xors. This method does the things within single-sized records and without extensive bit-level operations. Moreover, the performing speed can be also increased as a side effect.

The key of the method is the square root of "x". For trinomials with "m" and "n" odd it can be easily computed:

$$\begin{aligned} \text{sqrt}(x) \bmod P &= x^s + x^t \\ \text{with } s &= (m+1)/2, t = (n+1)/2 \end{aligned}$$

Then we split A and B as follows:

$$\begin{aligned} A &= a_0 + a_1*\text{sqrt}(x) \\ B &= b_0 + b_1*\text{sqrt}(x) \end{aligned}$$

where a<sub>0</sub>, b<sub>0</sub> are the remainders [A mod sqrt(x)] and [B mod sqrt(x)]; and a<sub>1</sub>, b<sub>1</sub> the quotients [A / sqrt(x)], [B / sqrt(x)]. These quotient/remainder operations can be done very efficiently because sqrt(x) is a binomial. All these polynomials have maximum degree (m-1)/2.

So we have:

## Multiplication trick in GF(2<sup>m</sup>)

$$A*B = a_0*b_0 + x*(a_1*b_1) + C*\text{sqrt}(x)$$

where C follows from the known Karatsuba's trick

$$C = (a_0 + a_1)*(b_0 + b_1) + a_0*b_0 + a_1*b_1$$

C is again splitted:

$$C = c_0 + c_1*\text{sqrt}(x)$$

so finally

$$R' = a_0*b_0 + x*(a_1*b_1 + c_1) + c_0*\text{sqrt}(x)$$

Notice that multiplication by  $\text{sqrt}(x)$  is very quick (two shifts and one xor).

R' has degree at most "m" so:

if  $\text{deg}(R') \leq m$  then  $R = R' + P$ ;  
else  $R = R'$ .

Squaring is also trivially done:

$$A^2 \bmod P = (a_0 + \text{sqrt}(x)*a_1)^2 = a_0^2 + x*a_1^2$$

The method applies only to this kind of trinomial primitives with "m" and "n" odd, but I think them very often in cryptographic algorithms; "m" uses to be prime and you can choose in most cases "n" as odd.

After somehow extensive google survey about optimal GF(2<sup>m</sup>) multiplication, I didn't find this matter documented anywhere; thus perhaps it is original. Comments and references to previous similar works are welcome.

---

Manuel Pancorbo Castro

<http://sks.merseine.nu/>

-----BEGIN PGP SIGNATURE-----

Version: GnuPG v1.4.6 (GNU/Linux)

Comment: SKS pocket ECC. <http://sks.merseine.nu/>

iD8DBQFGgZJSWmnkWA8sPKsRAiNvAKDucc33HLgIG7dIMQKx+toH28OkSQCeM7IJ

wUL7uQjpWrUf0rQVeBDSAc8=

=qDt4

-----END PGP SIGNATURE-----

.