第 36 卷 第 6 期 计 算 机 工 程 2010 年 3 月
Vol.36 No.6 Computer Engineering March 2010
·安全技术· 文章编号：1000—3428(2010)06—0140—02 文献标识码：A 中图分类号：TP309

GF(2^m)上的快速模约减算法

段 斌，马自堂
(解放军信息工程大学电子技术学院，郑州 450004)

摘 要：针对 GF(2^m)上的模约减运算问题，在基于固定三(或五)项式(FTOP)算法的基础上提出一种改进的快速算法。该算法采用动态计算分组字序号和偏移量的方法，克服 FTOP 只适用于特定约减多项式的不足。实验结果表明，当约减多项式项数小于 123(m<719)时，该算法速度比一次一位的算法有较大提高，最大为 89%，平均为 30%左右，当约减多项式为任意三(或五)项式时，能达到与 FTOP 相同的速度。
关键词：有限域；模约减；约减多项式；快速算法

Fast Module Reduction Algorithm over GF(2^m)

DUAN Bin, MA Zi-tang
(Institute of Electronic Technology, PLA Information Engineering University, Zhengzhou 450004)

【Abstract】 Aiming at the problem of the module reduction operation, this paper proposes a fast algorithm based on Fixed Trinomial Or Pentanomial(FTOP) algorithm over GF(2^m). By dynamically counting the index and offset of grouping words, the proposed algorithm overcomes the disadvantage of FTOP that of only available for fixed reduction polynomial, the proposed algorithm is faster than the one-time-one-bit algorithm when reduction polynomial has less terms of 123(m<719), maximum 89%, average about 30%, and with arbitrary trinomial or petanomial, it has the same speed as the FTOP's.
【Key words】 finite field; module reduction; reduction polynomial; fast algorithm

1 概述

模约减运算是基于有限域及其运算的密码体制中调用频繁的一种基本运算，而且还常常结合在乘法、平方等其他域运算中[1-2]。因此，算法的优化和改进对于提高密码体制的速度十分重要。按有限域的不同，模约减算法分为素数域 GF(p)和二进制域 GF(2^m)2 类[3]。对于有限域 GF(2^m)，模约减算法主要有一次一位和基于固定三(或五)项式(Fixed Trinomial Or Pentanomial, FTOP)2 种算法。一次一位的算法支持任意约减多项式，但速度较慢；而基于固定三(或五)项式的算法虽然速度较快，但只适合约减多项式为特定三项式或五项式的情形。本文对这 2 种算法及其原理进行分析，总结其优缺点，然后在 FTOP 算法原理的基础上提出了一种能适合任意约减多项式的改进算法，并进行了理论分析和实验结果的比较。

2 现有的 2 种模约减算法

2.1 一次一位的模约减算法

对于约减多项式 $f(x)=x^m+r(x)$，其中，$r(x)$次数最多为 m-1，可以通过一次一位的方式，从最低位起逐位完成对要约减的多项式 $c(x)$的处理，其原理如下：

$$c(x)=c_{2m-2}x^{2m-2}+\cdots+c_mx^m+\cdots+c_1x+c_0\equiv$$
$$(c_{2m-2}x^{m-1}+\cdots+c_m)r(x)+$$
$$c_{m-1}x^{m-1}+\cdots+c_1x+c_0 \pmod{f(x)}$$

通过预计算 $x^kr(x)(0\le k\le W-1)$，能进一步提高速度，得到算法 1(在以下算法中，设实现平台的字长为 W=32，有限域为表示域元素所用 W 位字的个数为 L)。

算法 1 一次一位的模约减算法[4]

输入 次数最高为 2m-2 的二进制多项式 $c(x)$
输出 $c(x) \bmod f(x)$
(1)预计算 $u_k(x)=x^kr(x),0\le k\le W-1$
(2)对于 i 从 2m-2 到 m，重复执行
若 c_i=1，则
$j=\lfloor (i-m)/W \rfloor, k=(i-m)-Wj$
$u_k(x)+C\{j\}$
(3)返回($C[t$-1, t-2,…,C[1],C[0])

2.2 固定三(或五)项式的快速模约减算法

若约减多项式 $f(x)$是固定项数的，如三项式或五项式，则约减运算可以通过分组字的移位加来完成。对于 $f(x)=x^m+x^k+1$，有

$$x^{2m-2}\equiv x^{k+m-2}+x^{m-2} \pmod{f(x)}$$
$$x^{2m-2}\equiv x^{k+m-2}+x^{m-2} \pmod{f(x)}$$
$$\vdots$$
$$x^{m+2}\equiv x^{k+2}+x^2 \pmod{f(x)}$$
$$x^{m+1}\equiv x^{k+1}+x \pmod{f(x)}$$
$$x^m\equiv x^k+1 \pmod{f(x)}$$

则乘积

$$c(x)=c_{2m-2}x^{2m-2}+\cdots+c_mx^m+c_{m-1}x^{m-1}+\cdots+c_1x+c_0$$

的约减可通过将 $c_i(m\le i\le 2m-2)$ 按位加到如上公式给出的相应位置即可。而且由于这一算法是按照分组字的方式来处理的，因此能很好地发挥通用处理器固定字长的优点。文献[5]推荐了 m=163,233,283,409,571 时的算法，下面本文给出 m=359 时的算法。

算法 2 固定三(或五)项式的快速模约减算法

以下是当 $f(x)=x^{359}+x^{68}+1$ 时的情形：

输入 次数最高为 2m-2 的二进制多项式 $c(x)$
输出 $c(x) \bmod f(x)$
(1)对于 i 从 23 到 12，重复执行

作者简介：段 斌(1981－)，男，硕士研究生，主研方向：信息安全技术；马自堂，教授
收稿日期：2009-12-10 **E-mail：**duanbin236366@hotmail.com

T=C[i]
C[i-12]=C[i-12]^(T<<25)
C[i-11]=C[i-11]^(T>>7)
C[i-10]=C[i-10]^(T<<29)
C[i-9]=C[i-9]^(T>>3)
(2)T=C[11]>>7
(3)C[0]=C[0]^T
(4)C[2]=C[2]^(T<<4)
(5)C[11]=C[11]&0*x*7F
(6)返回(C[11],C[10],⋯,C[2],C[1],C[0])

2.3 2 种算法的优缺点

算法 1 处理速度较为稳定，适用于对任意约减多项式的模约减，但是需要预计算，必须逐位处理，速度较慢；算法 2 速度较快，不需要预计算，但是当约减多项式项数较多或变化时，在其算法中不可能列出所有分组字的移位操作或者重新改写代码，无法实现对任意约减多项式的约减。

3 一种改进的快速模约减算法

针对算法 2 只能适用特定约减多项式的不足，本文对它进行了改进，将其扩展到对任意多项式的约减运算上。同理如约减多项式为固定三项或五项式的情形，对于任意的约减多项式

$$f(x)=x^{k_0}+x^{k_1}+\cdots+x^{k_{t-1}}+x^{k_t}=x^m+x^{k_1}+\cdots+x^{k_{t-1}}+1$$

其中，$2\le t\le m-1; k_t\le k_{t-1}\le\cdots\le k_2\le k_1\le k_0, k_t=0, k_0=m$，有，

$$\begin{aligned}
x^{2m-2}&\equiv x^{k_1+m-2}+x^{k_2+m-2}+\cdots+x^{k_t+m-2}+x^{m-2}(\bmod f(x))\\
x^{2m-3}&\equiv x^{k_1+m-3}+x^{k_2+m-3}+\cdots+x^{k_t+m-3}+x^{m-3}(\bmod f(x))\\
&\vdots\\
x^{m+2}&\equiv x^{k_1+2}+x^{k_2+2}+\cdots+x^{k_t+2}+x^2(\bmod f(x))\\
x^{m+1}&\equiv x^{k_1+1}+x^{k_2+1}+\cdots+x^{k_t+1}+x(\bmod f(x))\\
x^m&\equiv x^{k_1}+x^{k_2}+\cdots+x^{k_t}+1(\bmod f(x))
\end{aligned}$$

则乘积

$$c(x)=c_{2m-2}x^{2m-2}+\cdots+c_mx^m+c_{m-1}x^{m-1}+\cdots+c_1x+c_0$$

的约减可通过将 $c_i(m\le i\le 2m-2)$ 按位加到如上公式给出的相应位置即可。

算法 3 按上述原理给出了改进后对于任意约减多项式的快速模约减算法：先对 $f(x)$进行预计算，用一个数组 k 保存 $f(x)$中各项的次数，再通过引入 2 个中间变量 *index* 和 *offset* 来动态计算分组序号和偏移量，实现了约减多项式的任意性。最后对分组中包括要约减位和不需要约减位 2 部分的分组字 $C[L\text{-}1]$进行处理，把要约减的部分位提取出来按步骤(2)的方式约减，然后对 $C[L\text{-}1]$清除这一部分位。

算法 3 任意约减多项式的快速模约减算法

输入 次数最高为 2*m*-2 的多项式 $c(x)$

输出 $c(x)$ mod $f(x)$

(1)对于 f(x)，预计算 $k[i]=k_i(1\le i\le t)$

(2)对于 i 从$\lfloor 2m/W\rfloor+1$到$\lceil m/W\rceil$，重复执行

    T=C[i]
    对于 s 从 1 到 t，重复执行
        用 k[s]确定 index 和 offset
        如果 offset 不等于 0，则
            C[index]=C[index]^(T<<(W-offset))
            C[index+1]=C[index+1]^(T>>offset)
        否则
            C[index+1]=C[index+1]^T

(3)提取 C[L-1]中要约减的位

(4)约减 C[L-1]中要约减的位

(5)清除 C[L-1]中要约减的位

(6)返回(C[L-1], C[L-2],⋯,C[2],C[1],C[0])

4 分析比较

4.1 对 3 种算法的分析

算法 1~算法 3 的基本原理是相同的，区别在于算法 3 把对 $r(x)$的移位加操作整合成对分组字的移位加操作。对于给定的域宽 m，若约减多项式不是以静态常量形式而是以参数形式给出，那么当约减多项式改变时，算法 1 就需要较多移位操作和存储空间来预计算。在算法 1 中，若 $r(x)$由 $p(2\le p\le L)$个 W 位字表示，则需要 pW 个 W 位字的存储空间，并且在预计算时要进行 pW 次分组字移位运算。与算法 1 相比，算法 3 的存储复杂度为 $O(t)(3\le t\le m+1)$，需要分组字至多 $2tL$ 次移位和异或操作。但由于该算法在步骤(1)~步骤(2)中循环执行的次数取决于所选用的任意不可约多项式的项数，因此该算法的时间复杂度为 $O(tm)$。当 t=m-1 即 $f(x)$有 m 项时，时间复杂度达到最大 $O(m^2)$，当 t=2 即 $f(x)$有 3 项时，时间复杂度达到最小为 $O(m)$。因此，当 t 较小即 $f(x)$项数较少时，算法 3 的速度比算法 1 要快。但是当 t 较大即 $f(x)$项数较多时，算法 3 速度会明显下降。

对于给定的域宽 m，算法 1 所用时间 $t1$ 随 p 呈线性单调递增，当项数满足 p=L 时有最大值并且基本保持在这一数值左右，而算法 3 所用时间 $t3$ 随项数 x 呈线性单调递增；对于项数最小为 3 的情况，2 个算法所用时间显然有 $t1(3)>t3(3)$，而对于项数接近于 m 的情况有 $t1(m)<t3(m)$。因此，算法对应的时间-项数曲线 $t1(x)$和 $t3(x)$必有交点，而且 $t1(x)$和 $t3(x)$的交点 $Q(x,y)$中的 x 会随 m 的增大而变小。

与算法 2 相比，当 m=359, t=3 时，算法 3 能以相同的速度实现对任意三项式 $x^m+x^k+1(1\le k\le m-1)$ 的约减。FTOP 相当于算法 3 在约减多项式为某一特定三项式的特殊情形。

4.2 实验比较

实验在 Pentium4 2.9 GB 处理器和 Visual Studio 2008 环境中使用 C#语言编译测试，对算法 1 和算法 3 在约减多项式任意时的处理时间-项数曲线进行了比较，结果如图 1 所示。对于 m=191, 359 和 719 的情形，当项数小于 181,141 和 123 时，算法 3 所用时间比算法 1 明显要少，速度最大能提高 89%(当项数为 3 时)，平均提高 30%。需要说明的是，由于约减多项式每一项的位置可以任意，因此把约减多项式限定在给定项数范围内，并不影响算法满足实际应用对约减多项式任意性的需要。

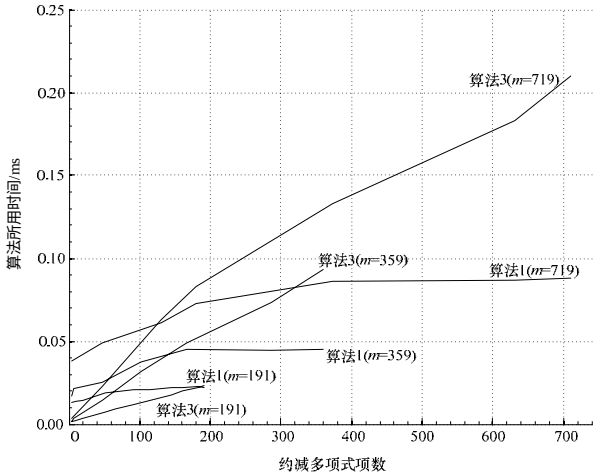

图 1 算法 1 和算法 3 的时间-项数曲线(*m*=191,359,719)

(下转第 145 页)