第 33 卷 第 22 期 计 算 机 工 程 2007 年 11 月
Vol.33 No.22 Computer Engineering November 2007
·博士论文· 文章编号：1000—3428(2007)22—0029—03 文献标识码：A 中图分类号：TP309.7

RSA/ECC 密码协处理器的硬件实现

何德彪，陈建华，孙金龙
(武汉大学数学与统计学院，武汉 430072)

摘 要：给出了一种公钥密码协处理器的结构，既可以计算定义在F_p上的椭圆曲线的点乘运算，也可以计算应用在RSA中的模幂运算，支持域长度不超过 256 比特的ECC，长度不超过 2 048 比特的RSA。该协处理器具有结构简单、实现方便、稍加调整即可满足用户对面积的要求等特点。

关键词：RSA；椭圆曲线密码；硬件实现；协处理器

Hardware Implementation of Cryptographic Processor for RSA and ECC

HE De-biao, CHEN Jian-hua, SUN Jin-long
(School of Mathematics & Statistics, Wuhan University, Wuhan 430072)

【Abstract】 This paper describes a hardware implementation of a cryptographic processor for RAS and ECC. The processor can not only compute modual exponent of RSA, which length is less than 2 048, but also compute point multiplication of ECC, which length is less than 256. The processor is implemented simply and can adjust area.

【Key words】 RSA; elliptic curve cryptography(ECC); hardware implementation; processor

1976 年，Diffie 和 Hellman 提出公钥密码的思想[1]和数字签名的概念，从此公钥密码得到迅速发展。目前应用最为广泛的公钥密码学是RSA[2]，随着破译技术的发展，RSA需要的密钥越来越长，运算代价越来越大。而椭圆曲线密码(ECC)[3~4]在密钥长度为 160 比特时与 1 024 比特密钥的RSA安全性相当，因此，与RSA相比具有明显的优势，得到越来越广泛的应用。虽然RSA被认为必然被ECC替代，但是这个替代过程需要很长的时间，到目前为止，RSA依然在很多地方得到应用。

随着技术的发展，需要处理的数据量越来越大，单纯依靠软件实现的密码机制已经不能满足工业的需要，而计算机本身又不支持超大整数的运算，设计公钥密码协处理器成为必然。现在又处于两种公钥密码的替代之际，因此支持两种公钥密码的协处理器很容易成为工业界关心的焦点。

文献[5~9]对 RSA 或者 ECC 协处理器进行了描述，大多利用 FPGA 进行实现，控制非常复杂。本文给出了一种结构简单、易于实现的协处理器结构。

1 数学基础

1.1 RSA 密码

下面给出 RSA 的简要介绍，具体参见文献[2]：

(1)公开的模 $M=P*Q$，其中，M 是长度为 $m=\lceil \log_2(M+1) \rceil$ 的数，是两个不公开的素数 P 和 Q 的乘绩。

(2)公钥 E 是一个公开的随机奇整数，有时为了加快加密速度可以取 E 为 3。

(3)私钥 D 由 $E*D=1 \bmod((P-1)*(Q-1))$确定。

(4)长度为 $n=km$ 的明文被划分为每块长度为 m 的明文块，对其中一块 A 加密，解密过程如下：

1)利用公钥加密

$P(A)=A^E \bmod M$

2)利用私钥解密

$S(A)=P(A)^E \bmod M$

显然，RSA 中最主要的运算是模幂运算，下面给出最基本的模幂算法：

算法 1 模幂运算

输入：X，E，M，设E的二进制表示为$(e_{l-1}, e_{l-2},\cdots,e_1,e_0)$

输出：$Y=X^E \bmod M$

(1)$Y \leftarrow 1$

(2)For i=l-1 down to 0 do

1)$Y \leftarrow Y^2 \bmod M$

2)If l_i=1 then $Y \leftarrow Y*X \bmod M$

(3)Return Y

1.2 椭圆曲线密码

1.2.1 椭圆曲线

有限域F_p(下面所有的有限域都是F_p，用K来表示)上的椭圆曲线定义如下：

E： $y^2=x^3+a\,x+b\ (a,b\in K,\ 4a^3+27b^2\neq 0)$

其中，K 的特征 $\neq 2$, 3，E 上的 K－有理点构成一个 Abelian 群，记做 $E(K)$，这里 $E(K)=\{(x,y)\in k^2 \mid y^2=x^3+a\,x+b\}\cup\{o\}$。

1.2.2 仿射坐标下的加法

在摄影坐标下，E 上的点 P 表示为：(x, y), $x, y\in K$，此时 Abelian 群 $E(K)$的加法定义如下：

(1)$O+O=O$；

(2) $\forall\ P=(x,y)\in E(K)\backslash\{O\}$, $P+O=P$；

(3) $\forall\ P=(x,y)\in E(K)\backslash\{O\}$, P 的逆元$-P=(x,-y)$，$P+(-P)=O$；

(4)点$P_1=(x_1,y_1)\in E(K)\backslash\{O\}$，$P_2=(x_2,y_2)\in E(K)\backslash\{O\}$，$P_3=P_1+P_2$，则

作者简介：何德彪(1980－)，男，博士，主研方向：数论与密码方向研究；陈建华，博士生导师；孙金龙，博士

收稿日期：2006-11-30 **E-mail：**hedebiao@163.com

$$\begin{cases} x_3=\lambda^2-x_1-x_2 \\ y_3=\lambda(x_1-x_3)-y_1 \end{cases}$$

其中，

$$\lambda=\begin{cases} \dfrac{y_2-y_1}{x_2-x_1}, \text{若} x_1\neq x_2 \\ \dfrac{3x_1+a}{2x_1}, \text{若} x_1=x_2, \text{且} P_1\neq -P_2 \end{cases}$$

1.2.3 仿射坐标下的点加

从上面公式很容易看出，在仿射坐标下面，需要求逆运算。而求逆运算相当困难，在摄影坐标下，可以避免求逆运算[4]。

摄影坐标有多种形式，采用了 Jacobi 摄影坐标。椭圆曲线上的点 P 在 Jacobi 摄影坐标下表示为：(X,Y,Z)。仿射点 $P(x, y)$按照如下公式转换为摄影坐标点 $P(X,Y,Z)$：

$X = x, Y = y, Z = 1$

摄影坐标点 $P(X,Y,Z)$按照如下公式转换为仿射坐标点 $P(x, y)$：

$x=X/Y^2$，$y = Y/Z^3$

下面给出在摄影坐标下点的加法公式：设$P = (X_0,Y_0,Z_0)$, $Q= (X_1,Y_1,Z_1)$, $P + Q = (X_2,Y_2,Z_2)$，如果$P = Q$，则

$X_2 =(3 X_1^2+ a Z_1^4)^2-8 X_1 Y_1^2$

$Y_2 = (3 X_1^2+ a Z_1^4) (4 X_1Y_1^2 -X_2)-8 Y_1^4$

$Z_2=2Y_1Z_1$

如果 $P \neq \pm Q$，则

$V= (X_0 Z_1^2+ X_1 Z_0^2) (X_0 Z_1^2- X_1 Z_0^2)-2X_2$

$X_2 = (X_0 Z_1^3-Y_1 Z_0^2)^2-(X_0 Z_1^2+ X_1 Z_0^2) (X_0 Z_1^2 -X_1 Z_0^2)$

$2Y_2 =V(X_0 Z_1^3-Y_1 Z_0^2)-(Y_0 Z_1^2+ Y_1 Z_0^2)(X_0 Z_1^2-X_1 Z_0^2)$

$Z_2 = Z_0 Z_1 (X_0 Z_1^2-X_1 Z_0^2)$

1.2.4 椭圆曲线的点乘运算

点乘运算的方法有很多种[10]，这里给出最基本的一种算法，其他的算法都是在这个算法的基础上进行了改进。

设 P 为椭圆曲线上的点，k 为正整数，k 的二进制表示为

$(k_{l-1}, k_{l-2},\cdots,k_1,k_0)$ $Q = kP$

算法 2 椭圆曲线点乘算法

输入：$k, P \notin E(K)$

输出：$Q = kP$

(1)$Q \leftarrow O$

(2)For i from l-1 down to 0 do

1)$Q \leftarrow 2Q$

2)If k_i=1 then $Q \leftarrow Q+P$

(3)Return Q

1.2.5 椭圆曲线数字签名和验证

定义 P 为椭圆曲线 E 上阶为大素数 n 的基点，d 为私钥，$Q = d P$ 为公钥，则椭圆曲线签名和验证算法如下：

算法 3 椭圆曲线签名算法

(1)产生随机数 k

(2)计算 $R=kP$

(3)计算$r=(R)_x \bmod n$，如果 r=0，则转到(1)

(4)计算 $s =(e+dr)/k \bmod n$，如果 s=0，则转到(1)

(5)输出签名 sig=(r, s)

算法 4 椭圆曲线签名验证算法

(1)如果 $r \notin [1,n-1]$或者 $s \notin [1,n-1]$，则输出“错误”

(2)计算 $1/s \bmod n$

(3)计算$(e/s \bmod n)P$

(4)计算$(r/s \bmod n)Q$

(5)计算 $r'=((e/s \bmod n)P+(r/s \bmod n)Q)_x$

(6)如果$r=r'$，则输出正确，否则输出错误

2 协处理器结构

从第 1 节对 RSA 和 ECC 的描述可以看出，这两种公钥密码机制都是建立在模乘和模加的基础上，因此，协处理器必须能进行模乘和模加运算。

2.1 公钥密码芯片的整体框架

公钥密码芯片整体框架如图 1 所示。

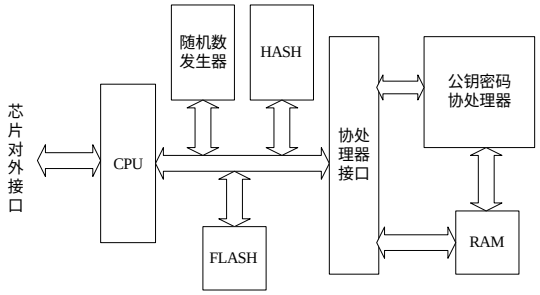

图 1 公钥密码芯片整体框架

为了降低实现代价，只把最低层的模乘和模加利用硬件实现，RSA 用到的模幂，ECC 用到的点加、点倍，都利用软件协处理器来实现，公钥密码机制也利用软件来实现。把实现公钥密码机制的代码，写在 FLASH 中，当 CPU 接到外部命令时，调用 FLASH 中的程序，与随机数发生器和公钥密码接口通信，完成公钥的加密、解密、签名或者验证。随机数发生器，FLASH，HASH，CPU 不是设计的重点所在，这里就不再赘述。

2.2 公钥密码协处理器

公钥密码协处理器是实现模乘、模加、模减运算，为了节省面积参与运算的数据都存储在 RAM 中，当参与运算的数据在 RAM 中存好以后，CPU 调用 FLASH 中的程序，不断对公钥密码协处理器发模乘，模加或模减的命令，从而完成需要进行的模幂或者点乘运算。运算结果仍然存储在 RAM 中，运算结束后 CPU 把 RAM 中的数据取走。模加和模减算法比较简单，其中主要是用到了全加器，在这里采用超前进位加法器，不再赘述，模乘算法如下：

算法 5 Montgomery 模乘算法

输入：A,B,M,e

输出：$D = A*B*R^{-1} \bmod M$ $(R = 2^{we})$, $D < M$

(1)D := 0

(2)For j = 0 to e-1 do

1)(C, S) := $A[0]B[j] + D[0]$

2)U := $S*MC \pmod{2^w}$

3)(C, S) := $(C, S)+M[0]U$

4)(C, S) >> w

5)For i = 1 to e-1 do

5.1)(C, S) := $(C, S) + A[i]B[j] + D[i] + D[i]U$

5.2)$D[i-1]$:= S

5.3)(C, S) >> w

6)$(C, S) = (C, S) + Dn$

7)$D[e-1] = S$

8)Dn = C 的第 0bit

(3)If$(D>M)$ then $D=D-M$

从算法很容易看出，关键点是实现 2.5.1，为了到达节省芯片面积的目的，把 A，B，M 都存储在 RAM 中，运算需要的时候才从 RAM 中取出 $A[i]$, $B[j]$, $D[i]$, $D[i]$，把这一步分成 2 次来做，第 1 次计算$(C, S) + A[i]B[j]$，第 2 次计算$(C, S) + D[i]$

$+D[i]U$，RAM 是三端口的，端口为两读一写，以 w 为 32 比特为例来说明，为了计算 2 048 比特的 RSA 需要的模幂运算，需要容量为 256×32 的 RAM，显然，在计算 2.5.1 中，也需要用到全加器，在实现过程中，把模乘运算，模加运算和模减运算用到的全加器复用，从而达到降低实现代价的目的。公钥密码协处理结构如图 2 所示。

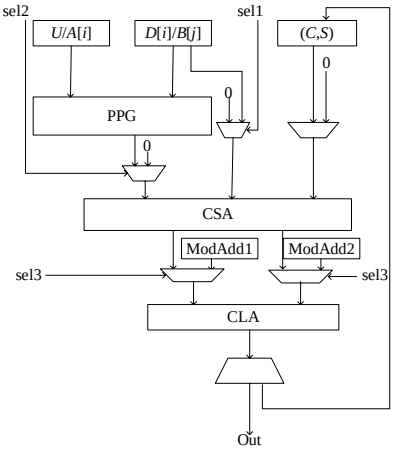

图 2　公钥密码协处理器结构

2.3　协处理器接口

协处理器接口模块，负责协处理器以及 RAM 和外部的数据交换，当接收的是读写命令时，接口模块直接对 RAM 操作，当接收到的为模乘、模加、模减命令时，在协处理空闲的时候，接口模块则把命令传递给协处理器，一般情况下，外部环境都是在检测到协处理空闲的时候再发送操作命令，这样从协处理器运算结束到下一次运算开始需要等待相当长的时间，会大大降低协处理的性能。为了克服这一弱点，在接口模块中加入了 16×4 的 FIFO，这样所有的模乘、模加、模减命令，都先存储在 FIFO 中，在 FIFO 非空的情况下，当接口模块检测到协处理空闲时就会从 FIFO 中取出一条指令送到协处理器模块，让协处理器始终处于运转状态，而外边不再检测协处理器是否空闲，只要检测到 FIFO 不是满的，就可以把需要的指令送到接口模块，所有的运算指令发送完毕以后，外部检测 FIFO 是否为空，当为空的时候再检测协处理器是否处于空闲状态，当协处理器处于空闲状态时，说明所有运算结束，可以从 RAM 中取出结果。协处理，接口模块，RAM 的数据交换如图 3 所示。

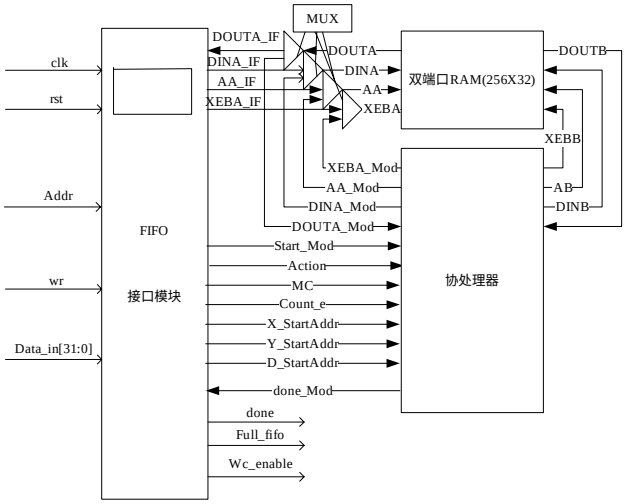

图 3　数据交换图

2.4　例子

为了说明协处理器的使用方法，以点倍运算为例，详细说明该处理器的。首先对指令结构进行说明，指令结构如表 1 所示。

表 1　指令结构

Bit15..Bit12	Bit11..Bit0
xxxx	X|Y|D

第 12 比特到第 15 比特为操作命令，0001 代表模乘运算，0010 代表模加运算，0011 代表模加运算，第 8 比特到第 11 比特为第 1 个操作数再 RAM 中的地址，第 4 比特到第 7 比特为第 2 个数在 RAM 中的地址，D 为结果存放的地址。

在进行点倍运算之前，先把数据写进 RAM 中，具体数据安排如图 4 所示。

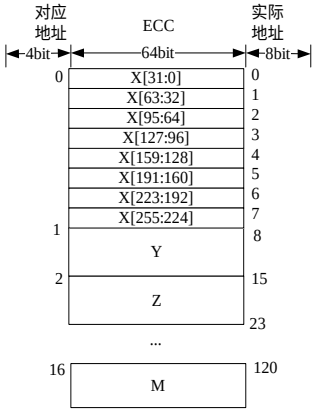

图 4　RAM 中的数据分布

该处理器最终要进行点乘运算，从算法 2 可以看出，每一次的点倍的结构要放在原来的位置，因此，可以连续放松如下命令就可以完成点倍运算。点倍运算命令为：16'h1115，16'h1056，16'h2666，16'h2666，16'h1557，16'h2775，16'h2555，16'h2555，16'h1227，16'h1778，16'h18e7，16'h1008，16'h2880，16'h2808，16'h2878，16'h1880，16'h3060，16'h3060，16'h3606，16'h1127，16'h2772，16'h1861，16'h3151。当然，可以根据需要设计不同的数据存储位置，对于 RSA，指令相当简单，这里不再赘述。

3　结束语

本文给出了一种同时支持RSA和F_p上ECC的公钥密码协处理器的实现方法，该处理器可以根据需要调整面积，对于最关键的模乘算法，采用Montgomery模乘算法。协处理器代码用Verilog编写，用Modelsim6.0 仿真，由Complier Design 综合得到网表。对于遇到的协处理器空闲时间过长问题，也给出了解决方案。最后结合具体例子，对协处理器的使用方法给出了说明。

参考文献

1 Diffie W, Hellman M. New Directions in Cryptography[J]. IEEE Transactions on Information Theory, 1976, 22(6): 644-654.

2 Rivest R, Shamir A, Adleman L. A Method for Obtaining Digital Signatures and Public Key Cryptosystems[J]. Communications of the ACM, 1978, 21(2): 120-126.

3 Koblitz N. Elliptic Curve Cryptosystem[J]. Math. Comp., 1987, 44(1): 203-209.

4 Blake I, Seroussi G, Smart N P. Elliptic Curves in Cryptography[M]. London: Cambridge University Press,1999.

(下转第 34 页)