



(12)发明专利申请

(10)申请公布号 CN 105871552 A  
(43)申请公布日 2016.08.17

(21)申请号 201610428614.9

(22)申请日 2016.06.14

(71)申请人 天津大学

地址 300072 天津市南开区卫津路92号

(72)发明人 郭炜 郝中源 魏继增

(74)专利代理机构 天津市北洋有限责任专利代理  
事务所 12201

代理人 刘国威

(51)Int.Cl.

H04L 9/30(2006.01)

H04L 9/00(2006.01)

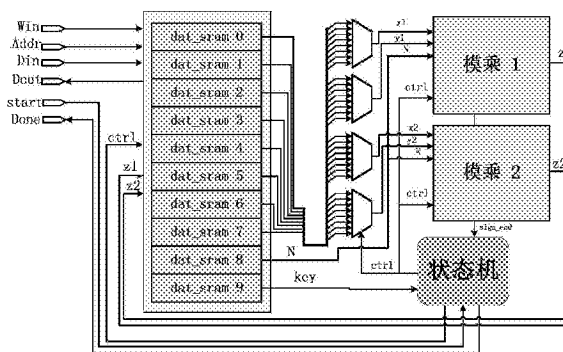
权利要求书2页 说明书8页 附图4页

(54)发明名称

双核并行RSA密码处理方法及协处理器

(57)摘要

本发明涉及到信息安全及微处理器设计领域,为实现通过FIOS模乘算法将模乘转换为简单的小数加法和乘法运算,充分降低模乘运算单元的面积,有效避免大量中间数据的写回过程。从硬件实现的角度提升算法的计算效率并进一步节省计算资源,从根本上缩减加解密时间和空间开销,有效提升RSA的加解密性能。本发明采用的技术方案是,双核并行RSA密码处理方法,在进行加密之前,需要借助证书颁发机构(CA)作为可信第三方,负责用户私钥和公钥证书的生成、保管、维护、撤销环节,加密时,用户B执行运算 $c=m^e \pmod{N}$ 并将加密信息c发送给用户A;解密时,用户A利用自己的私钥d对密文c执行运算从而恢复出明文。本发明主要应用于信息安全处理。



1. 一种双核并行RSA密码处理方法,其特征是,在进行加密之前,需要借助证书颁发机构(CA)作为可信第三方,负责用户私钥和公钥证书的生成、保管、维护、撤销环节,其密钥对生成过程为:

取两个大素数 $p$ 和 $q$ ,需要保密;

计算模数 $N=pq$ ,使 $N$ 的宽度等于密钥长度 $L$ ,公开;

计算 $n$ 的欧拉函数 $\phi(N)=(p-1)(q-1)$ ,保密;

随机选取整数 $e \in (1, \phi(N))$ ,并使最大公约数 $\text{GCD}(e, \phi(N))=1$ ,将 $e$ 作为公钥并公开;

计算私钥 $d$ ,使其满足 $d \times e \equiv 1 \pmod{\phi(N)}$ ,并 $d$ 将交给用户保密;

假设用户B要加密消息发送给用户A,那么用户B应提前获得用户A的公钥 $(e, N)$ ,并将信息数字化,使每段信息 $m$ 的长度不大于 $L$ ;

加密时,用户B执行运算 $c = m^e \pmod{N}$ 并将加密信息 $c$ 发送给用户A;

解密时,用户A利用自己的私钥 $d$ 对密文 $c$ 执行运算 $m = c^d \pmod{N}$ ,从而恢复出明文。

2. 如权利要求1所述的双核并行RSA密码处理方法,其特征是,加解密是不断地调用模乘操作以实现模幂运算的过程,具体使用基于蒙哥马利算法的高基FIOS算法,该算法将密钥长度 $L$ 分为 $s$ 个 $r$ -bit的字段,所述算法的所有运算总结为 $(t, z) = a + xy + b$ 一种,其中 $a, b, x, y$ 均为 $r$ -bit输入, $z$ 为结果的低 $r$ 位, $t$ 为结果的高 $r$ 位;同时利用该运算,通过将 $x, y$ 置1或0来完成素域上的模加和模减操作,计算模加或模减时,首先由超前进位加法器CLA完成加减操作,若结果越界则先由3-2压缩器执行 $a+b-N$ 或 $a-b+N$ ,再对压缩后的结果使用超前进位加法器相加即可完成。

3. 如权利要求1所述的双核并行RSA密码处理方法,其特征是,加解密是采用PoweringLadder模幂算法,所述算法是对二进制Montgomery阶梯算法的改进,执行时循环体从左到右对私钥各位依次扫描,无论0、1都需要执行一次模乘和模平方操作。

4. 一种双核并行RSA密码处理器,其特征是,由存储、控制和模乘运算单元三部分组成,工作时,两个模乘单元并行运行,并且总是一个执行模乘操作,另一个执行模平方操作;其中,存储单元共包括10块数据RAM,除RAM8和RAM9分别固定存储模数 $N$ 和密钥 $e$ 外,其余8块RAM将在每次模乘操作之前由控制信号选择3块作为两个模乘单元的输入,2块存放运算结果;控制单元负责控制内部RAM和数据通路的选择,包括状态机和数据选择器;空闲时,各块数据RAM均由外部端口访问,启动后再将读写控制权交给模乘运算单元;此时状态机按位扫描RAM9中的密钥 $e$ 并根据工作状态产生相应的控制信号,选出相应的RAM并将读写控制权交给模乘单元;运算结束后模乘单元将结果写回RAM,状态机将系统状态置为空闲,外部端口重新获得RAM控制权;

模乘模块由控制、运算和输入输出IO三部分构成,其中,控制单元包括FSM状态机和片选信号,控制着模乘器的工作状态和存储、运算的执行过程;IO单元包括了各数据和控制信号端口,负责根据模乘器的工作状态从外部存储器中读取或写入操作数和结果;运算单元由算数核构成,选择64bit的字作为基本运算单元,运算核 $(t, z) = a + xy + b$ 的硬件架构是,运算核的 $x \times y$ 部分使用Booth乘法器实现,操作数 $a, b$ 和部分积同时由3-2压缩器进行压缩,压缩后的结果通过超前进位加法器相加完成运算。

5. 如权利要求4所述的双核并行RSA密码处理器,其特征是,整个模乘模块对于 $s \times 64$ bit的操作数完成一次模加减运算只需 $8+s$ 个周期,完成一次模乘运算需要 $2s \times (s+1)$ 个

周期。

## 双核并行RSA密码处理方法及协处理器

### 技术领域

[0001] 本发明涉及到信息安全及微处理器设计领域,特别涉及一种基于RSA密码算法的双核并行密码协处理器设计方案。

### 背景技术

[0002] 公钥密码体制也称非对称密码,是目前保障信息安全的主要手段。RSA算法是第一个能被同时用于加密和数字签名的公钥加密算法。由于其安全性良好和易于理解和实现的特点,被认为是目前最有影响力的公钥加密算法之一。RSA算法基于一个十分简单的数学事实:将两个大素数相乘十分容易,但对其乘积进行因式分解却极其困难,因此可以将两数的乘积公开作为加密密钥,接收方只有使用自己的私钥才能解密。这就使得加密算法甚至加密密钥均可以公开,接收者只将解密密钥保密就可以,实现了信息的保密性。RSA算法从提出到现在的四十多年来,经历了各种攻击的考验,已经在网络通信的许多方面得到了广泛认可和应用。发展至今,在电子安全领域已经形成了较为完备的国际规范,在各领域的应用数不胜数。它能够抵抗到目前为止已知的绝大多数密码攻击,并且已被国际标准化组织(ISO)推荐为公钥数据加密的国际标准。

[0003] RSA密码系统通过使用一系列大数模幂运算完成加解密过程,而模幂运算由一系列大数模乘运算构成。对极大整数做因数分解的难度决定了RSA算法的可靠性。密钥长度越长,安全等级越高,同时计算量越大速度就越慢。为了保证密钥具有足够的安全等级,当前RSA加密协议的密钥长度普遍需要2048位甚至更长,限制了RSA算法的计算速度。RSA中的模幂运算性能主要依赖于大数模乘运算的速度且结构非常复杂,无法满足加密芯片对速度和面积日益苛刻的要求。模幂阶梯(Powering Ladder)算法是对二进制模幂算法的改进,具有一定的并行性,但也造成芯片面积的显著增加。基于FIOS(Finely Integrated Operand Scanning)方法的Montgomery模乘算法由于避免了长整数的比较并将复杂的除法操作转换为简单的移位操作,非常便于硬件实现的算法,目前已成为公钥密码系统应用最为广泛的模乘算法之一。该算法将模乘的两个操作数都划分为多个字逐字扫描,将模乘转换为简单的小数加法和乘法运算。这就使得RSA加密处理器可以利用FIOS模乘算法和Powering Ladder模幂算法结合实现快速并行处理。

### 发明内容

[0004] 为克服现有技术的不足,在Powering Ladder模幂算法的基础上,本发明旨在设计、提供一种实用的双核RSA密码协处理器。通过FIOS模乘算法将模乘转换为简单的小数加法和乘法运算,充分降低模乘运算单元的面积,有效避免大量中间数据的写回过程。从硬件实现的角度提升算法的计算效率并进一步节省计算资源,从根本上缩减加解密时间和空间开销,有效提升RSA的加解密性能。本发明采用的技术方案是,双核并行RSA密码处理方法,在进行加密之前,需要借助证书颁发机构(CA)作为可信第三方,负责用户私钥和公钥证书的生成、保管、维护、撤销环节,其密钥对生成过程为:

- [0005] 取两个大素数 $p$ 和 $q$ ,需要保密;
- [0006] 计算模数 $N=pq$ ,使 $N$ 的宽度等于密钥长度 $L$ ,公开;
- [0007] 计算 $n$ 的欧拉函数 $\phi(N)=(p-1)(q-1)$ ,保密;
- [0008] 随机选取整数 $e \in (1, \phi(N))$ ,并使最大公约数 $\text{GCD}(e, \phi(N))=1$ ,将 $e$ 作为公钥并公开;
- [0009] 计算私钥 $d$ ,使其满足 $d \times e \equiv 1 \pmod{\phi(N)}$ ,并 $d$ 将交给用户保密;
- [0010] 假设用户B要加密消息发送给用户A,那么用户B应提前获得用户A的公钥 $(e, N)$ ,并将信息数字化,使每段信息 $m$ 的长度不大于 $L$ ;
- [0011] 加密时,用户B执行运算 $c = m^e \pmod{N}$ 并将加密信息 $c$ 发送给用户A;
- [0012] 解密时,用户A利用自己的私钥 $d$ 对密文 $c$ 执行运算 $m = c^d \pmod{N}$ ,从而恢复出明文。
- [0013] 加解密是不断地调用模乘操作以实现模幂运算的过程,具体使用基于蒙哥马利算法的高基FIOS算法,该算法将密钥长度 $L$ 分为 $s$ 个 $r$ -bit的字段,所述算法的所有运算总结为 $(t, z) = a + xy + b$ 一种,其中 $a, b, x, y$ 均为 $r$ -bit输入, $z$ 为结果的低 $r$ 位, $t$ 为结果的高 $r$ 位;同时利用该运算,通过将 $x, y$ 置1或0来完成素域上的模加和模减操作,计算模加或模减时,首先由超前进位加法器CLA完成加减操作,若结果越界则先由3-2压缩器执行 $a+b-N$ 或 $a-b+N$ ,再对压缩后的结果使用超前进位加法器相加即可完成。
- [0014] 加解密是采用Powering Ladder模幂算法,所述算法是对二进制Montgomery阶梯算法的改进,执行时循环体从左到右对私钥各位依次扫描,无论0、1都需要执行一次模乘和模平方操作。
- [0015] 双核并行RSA密码协处理器,由存储、控制和模乘运算单元三部分组成,工作时,两个模乘单元并行运行,并且总是一个执行模乘操作,另一个执行模平方操作;其中,存储单元共包括10块数据RAM,除RAM8和RAM9分别固定存储模数 $N$ 和密钥 $e$ 外,其余8块RAM将在每次模乘操作之前由控制信号选择3块作为两个模乘单元的输入,2块存放运算结果;控制单元负责控制内部RAM和数据通路的选择,包括状态机和数据选择器;空闲时,各块数据RAM均由外部端口访问,启动后再将读写控制权交给模乘运算单元;此时状态机按位扫描RAM9中的密钥 $e$ 并根据工作状态产生相应的控制信号,选出相应的RAM并将读写控制权交给模乘单元;运算结束后模乘单元将结果写回RAM,状态机将系统状态置为空闲,外部端口重新获得RAM控制权;
- [0016] 模乘模块由控制、运算和输入输出IO三部分构成,其中,控制单元包括FSM状态机和片选信号,控制着模乘器的工作状态和存储、运算的执行过程;IO单元包括了各数据和控制信号端口,负责根据模乘器的工作状态从外部存储器中读取或写入操作数和结果;运算单元由算数核构成,选择64bit的字作为基本运算单元,运算核 $(t, z) = a + xy + b$ 的硬件架构是,运算核的 $x \times y$ 部分使用Booth乘法器实现,操作数 $a, b$ 和部分积同时由3-2压缩器进行压缩,压缩后的结果通过超前进位加法器相加完成运算。
- [0017] 整个模乘模块对于 $s \times 64\text{bit}$ 的操作数完成一次模加减运算只需 $8+s$ 个周期,完成一次模乘运算需要 $2s \times (s+1)$ 个周期。
- [0018] 本发明的特点及有益效果是:
- [0019] 本发明使用Powering Ladder模幂算法,实现了双核并行的RSA密码协处理器设计。对其中的单个运算模块,发明设计了基于FIOS蒙哥马利算法的硬件模乘器。模乘器核心

运算只有 $(t, z) = a + xy + b$ 一种,而且所有的运算可以用同一硬件完成,节省了芯片面积。系统提供多个RAM用以存储操作数和结果,避免了数据的连续搬运,节省了模幂运算时间。加密时每次循环中模乘和模平方操作均并行执行,不仅提高了RSA的运算效率,而且能有效抵抗时间攻击和简单功耗攻击,提高了系统的安全性。

#### 附图说明:

- [0020] 图1模乘模块硬件架构图。图中,
- [0021] X、Y、N、Z为操作数 Dout:数据输出端口 Mode:模式选择信号
- [0022] Start:启示信号 Ctrl:控制信号 State:状态信号
- [0023] sum\_l:地址低位 sum\_h:地址高位 sign\_end:完成信号。
- [0024] 图2乘加运算单元结构示意图。
- [0025] 图3 FIOS蒙哥马利算法流程图。
- [0026] 图4 Powering Ladder模幂算法流程图。
- [0027] 图5 RSA双核协处理器架构图。
- [0028] 符号说明:Din:数据输入端口 Dout:数据输出端口 Win:读写控制信号
- [0029] Start:启示信号 Ctrl:控制信号 Done:完成信号
- [0030] Dat\_sram:数据RAM N:模数 key:密钥。

#### 具体实施方式

[0031] 本发明设计是一种双核并行RSA加密协处理器。设计使用了基于64bit字的FIOS模乘算法作为单个运算核的执行算法,使系统在硬件结构上充分提高硬件的并行性,充分降低芯片的面积。同时利用Powering Ladder模幂算法中模乘与模平方之间无数据依赖关系的特性,保证两个核的运算独立运行,大大提升了RSA加密运算的执行效率。

[0032] 本发明为设计出高效安全的双核并行RAS密码协处理器,选择了Powering Ladder模幂算法作为RAS的加解密执行算法。该算法通过指数拆分将模幂运算转化为一系列的模乘和模平方运算,且两者之间无数据依赖关系,能够通过两个核独立运行提升模幂运算效率。由于模乘和模平方其实是一样的操作,可以通过同一硬件来完成。基于FIOS方法的Montgomery模乘算法通过对模乘的两个操作数都划分为多个字逐字扫描,将模乘转换为简单的小数加法和乘法运算,便于硬件实现。该方法将对大整数的取模转换为对 $2^r$ 的取模,因而能通过简单的移位操作实现,避免了长整数的比较和复杂的除法操作。同时算法具有较小的运算核心,能充分提高硬件的并行性并显著降低芯片的面积。

[0033] RSA加密是一种公钥密码体制,其安全性依赖于有限域上的离散对数问题(DLP),其数学基础是欧拉定理。在进行加密之前,用户需要借助证书颁发机构(CA)作为可信第三方,负责用户私钥和公钥证书的生成、保管、维护、撤销等环节,其密钥对生成过程为:

- [0034] 取两个大素数 $p$ 和 $q$ ,需要保密;
- [0035] 计算模数 $N = pq$ ,使 $N$ 的宽度等于密钥长度 $L$ ,公开;
- [0036] 计算 $n$ 的欧拉函数 $\phi(N) = (p-1)(q-1)$ ,保密;
- [0037] 随机选取整数 $e \in (1, \phi(N))$ ,并使最大公约数 $\text{GCD}(e, \phi(N)) = 1$ ,将 $e$ 作为公钥并公开;

[0038] 计算私钥 $d$ ,使其满足 $d \times e \equiv 1 \pmod{\phi(N)}$ ,并 $d$ 将交给用户保密;

[0039] 假设用户B要加密消息发送给用户A,那么用户B应提前获得用户A的公钥 $(e, N)$ ,并将信息数字化,使每段信息 $m$ 的长度不大于 $L$ ;

[0040] 加密时,用户B执行运算 $c = m^e \pmod{N}$ 并将加密信息 $c$ 发送给用户A;

[0041] 解密时,用户A利用自己的私钥 $d$ 对密文 $c$ 执行运算 $m = c^d \pmod{N}$ ,从而恢复出明文。

[0042] RSA算法的加解密其实就是不断地调用模乘操作以实现模幂运算的过程,针对加密速度有着关键性影响的模乘模块,本发明使用了基于蒙哥马利算法的高基FIOS算法如算法1所示,该算法的所有运算可被简单总结为 $(t, z) = a + xy + b$ 一种。同时利用该运算,系统还能通过将 $x, y$ 置1或0来完成素域上的模加和模减操作。计算模加或模减时,模块首先由超前进位加法器(CLA)完成加减操作,若结果越界则先由3-2压缩器执行 $a+b-N$ 或 $a-b+N$ ,再对压缩后的结果使用超前进位加法器相加即可完成。

[0043] 模乘模块由控制、运算和输入输出(I/O)三部分构成,其基本架构如图1所示。其中,控制单元包括FSM状态机和片选信号,控制着模乘器的工作状态和存储、运算的执行过程。I/O单元包括了各数据和控制信号端口,负责根据模乘器的工作状态从外部存储器中读取或写入操作数和结果。运算单元由算数核构成,是系统的主要运算部分。通过对比不同位宽的基导致的计算速度和时钟周期数的差异,本设计选择了64bit的字作为基本运算单元。图2为模乘器中运算核 $(t, z) = a + xy + b$ 的硬件架构图,运算核的 $x \times y$ 部分使用Booth乘法器实现,操作数 $a, b$ 和部分积同时由3-2压缩器进行压缩,压缩后的结果通过超前进位加法器相加完成运算。图3为本发明执行FIOS蒙哥马利算法的算法流程图,整个模乘模块对于 $s \times 64\text{bit}$ 的操作数完成一次模加减运算只需 $8+s$ 个周期,完成一次模乘运算需要 $2s \times (s+1)$ 个周期。由于该运算单元采用按字相乘实现,实现了计算速度的提升。

[0044] 算法1、FIOS蒙哥马利算法

[0045]

---

输入:  $X, Y, N, \eta = -N^{-1} \pmod{2^r}$ ,  $r$  为字长,  $s$  为字的个数

输出:  $Mont(X, Y, N) = X \times Y \times R^{-1} \pmod{N}$ , 式中  $R = 2^L, L = r \times s$  为密钥位宽

---

1:  $Z = 0; v = 0;$

2: **for** ( $i = 0$  to  $s-1$ )

3:  $(t_a, Z_0) = Z_0 + X_i Y_0; \mu_i = \eta \times Z_0 \pmod{2^r}; (t_b, Z_0) = Z_0 + \mu_i N_0;$

4: **for** ( $j = 1$  to  $s-1$ )

5:  $(t_a, Z_j) = Z_j + X_i Y_j + t_a;$

6:  $(t_b, Z_{j-1}) = Z_j + \mu_i N_j + t_b;$

7: **endfor**

8:  $(v, Z_{s-1}) = t_b + t_b + v;$

---

[0046]

9: **endfor**

10: **if** ( $Z > N$ )  $Z = Z - N;$

11: return  $Z$

---

[0047] 算法2为本发明为实现RSA加解密过程而采用的Powering Ladder模幂算法。该算法是对二进制Montgomery阶梯算法的改进。该算法的运算流程如图4所示,执行时循环体从左到右对私钥各位依次扫描,无论0、1都需要执行一次模乘和模平方操作。这就保证了模幂运算的时间是固定的,加解密时间不因私钥变化而变化,因而能有效地抵抗时间攻击(Timing Attack)和简单功耗攻击(SPA)。此算法也因其模乘与模平方之间无数据依赖关系,故可以采用并行的方法来实现。如果按私钥中0和1的个数相等来算,并且模乘和模平方的执行时间相同,则此并行后的执行时间将比传统的二进制Montgomery阶梯算法快了2.6倍。

[0048] 算法2、Powering Ladder模幂算法

[0049]

---

输入: 明文  $M$ , 模数  $N$ , 密钥  $e = (e_{L-1}, e_{L-2}, \dots, e_0)_2$ , 参数  $\lambda = 2^{2(L+2)} \bmod N$ ,

$L$  为密钥宽度

输出:  $M^e \bmod N$

---

```

1:   $f_0 = \text{Mont}(\lambda, 1, N); f_1 = \text{Mont}(\lambda, M, N);$ 
2:  for (i = L-1 down to 0)
3:      if ( $e_i == 0$ )
4:           $f_1 = \text{Mont}(f_0, f_1, N); f_0 = \text{Mont}(f_0, f_0, N);$ 
5:      else ( $e_i == 1$ )
6:           $f_0 = \text{Mont}(f_0, f_1, N); f_1 = \text{Mont}(f_1, f_1, N);$ 
7:  endfor
8:   $f_0 = \text{Mont}(f_0, 1, N);$ 
9:  return  $f_0$ 

```

---

[0050] 图5为本设计RSA加密协处理器的整体硬件架构图,整个芯片由存储、控制和模乘运算单元三部分组成。工作时,两个模乘单元可并行运行,并且总是一个执行模乘操作,另一个执行模平方操作。其中,存储单元共包括10块数据RAM(大小 $16 \times 64\text{bit}$ )。除RAM8和RAM9分别固定存储模数 $N$ 和密钥 $e$ 外,其余8块RAM将在每次模乘操作之前由控制信号选择3块作为两个模乘单元的输入,2块存放运算结果。这样模幂过程就避免了数据的连续搬运,节省了模幂运算时间。控制单元负责控制内部RAM和数据通路的选择,主要包括状态机和数据选择器。空闲时,各块数据RAM均由外部端口访问,系统启动后再将读写控制权交给模乘运算单元。此时状态机按位扫描RAM9中的密钥 $e$ 并根据工作状态产生相应的控制信号,选出相应的RAM并将读写控制权交给模乘单元。运算结束后模乘单元将结果写回RAM,状态机将系统状态置为空闲,外部端口重新获得RAM控制权。

[0051] RAS模块的硬件架构共有6个端口,其中Win为读写控制信号,Addr为地址信号,控制选择特定RAM中的特定字段作为输入,Din和Dout分别为数据输入和输出端口,Start为起始信号,Done为完成信号。工作时,首先将Win设为写模式,通过Addr和Din将操作数和参数传入存储模块,Start启动后系统开始工作,若Done置1则运算完成,再设置Addr和Win将结果读出。该架构模乘和模平方操作可并行执行,通过修改RAM的深度可支持512bit到2048bit不同长度信息的加密。对于位宽为 $s \times 64\text{bit}$ 的信息完成一次加密最大需要 $2s(64s+$



2)(s+1)个周期。设计通过双核并行执行不仅充分提高了RSA的运算效率,而且有效地抵抗时间攻击和简单功耗攻击,提高了系统的安全性。

[0052] 下面结合附图及实例,对本发明进行进一步详细说明。应该指出,此处所描述的具体实施实例仅仅用以解释本发明,并不用于限定本发明。为使本发明的目的、技术方案和优点更加清晰,本实例在以本发明技术方案为前提下进行实施,给出了详细的实施方式和具体的操作过程。

[0053] 本发明选择基于Powering Ladder模幂算法实现双核并行架构的RSA密码协处理器。该方案设计了多个RAM用以存储操作数和结果,避免了数据的连续搬运时间。为使本发明的目的、技术方案和优点更加清晰,这里给出一段Verilog代码来表示该双核RSA密码协处理器架构的具体方案:

```

`define Prm      64'h298bc7dc99178307//定义预计算参数
`define DAT_W    64 //定义数据位宽
`define ADDR_W   16 //定义数据深度
module pl_rsa(
    clk,          //输入端口声明
    rst_b,
    wen,
    addr,
    Din,
    start,
    Dout,        //输出端口声明
    Done
);
//状态机描述

//时序逻辑描述

Mont  mont1();//FIOS 蒙哥马利模乘模块
Mont  mont2();//双核

dat_sram  dat_sram0();//RAM 存储模块
dat_sram  dat_sram1();
.....
dat_sram  dat_sram9();//RAM9 中存储密钥e
endmodule

```

[0055] 加密时,系统首先将明文和参数传入存储模块,其中所有RAM的RTL代码均由

memory compiler工具生成。其中RAM0存放明文m, RAM0存放常数1, 初始化参数λ存入RAM1和RAM3。RAM8和RAM9分别保存密钥e和模数N, 并在计算过程中保持不变。加密过程中, 系统根据状态机信号在RAM0到RAM7中选择相应数据作为模乘和模平方的操作数和结果, 加密结束后经过多次循环最终结果仍保存在RAM0中。

[0057] 针对模乘模块, 发明设计了基于FIOS蒙哥马利算法的硬件模乘器。由于蒙哥马利算法的结果为 $X \times Y \times R^{-1} \bmod N$ , 还不是真正的模乘结果, 为此系统在真正运算之前都先将操作数转换到蒙哥马利域下, 其形式为:

[0058]  $X \rightarrow XR \bmod N; Y \rightarrow YR \bmod N$

[0059] 这就保证了利用FIOS算法可在蒙哥马利域下进行正常的模乘运算:

[0060]  $\text{Mont}(XR, YR, N) = XR \times YR \times R^{-1} \bmod N$

[0061]  $= (X \times Y)R \bmod N;$

[0062] 加密结束前再将运算结果由蒙哥马利域转换到正常域下:

[0063]  $X = \text{Mont}(XR, 1, N) = XR \times 1 \times R^{-1} \bmod N$

[0064] 以上各操作数的转化步骤均已在算法2中有所体现。根据对FIOS蒙哥马利算法的分析, 模乘器的核心运算只有 $(t, z) = a + xy + b$ 一种, 所有的运算可以用同一硬件完成, 能有效降低芯片面积, 模乘模块设计代码为:

```

module    Mont(
            clk,                //输入端口声明
            rst_n,
            start,
            Din_X, //操作数读入端口
            Din_Y,
            Din_N,
            Din_Z,
[0065]
            reg_addr,          //输出端口声明
            reg_wen, //外部 RAM 读写控制
            Dout_Z, //结果输出端口
            sign_end
        );
        //状态机描述
        //时序逻辑描述

```

```
//算术运算单元  $(t, z) = a + xy + b$   
    mult_add    mult_add(add_a,    //输入端口声明  
                        mul_x,  
[0066]         mul_y,  
                        add_b,  
                        sum_h,    //输出端口声明  
                        sum_l);  
  
    endmodule
```

[0067] 使用该代码在modelsim仿真平台下运行加密,对比得到的密文和magma高等代数仿真软件结果是否相同。代码中循环扫描部分也可以通过增加对密钥信息的盲化措施等方法加以改进,进一步提高系统的抗攻击性。

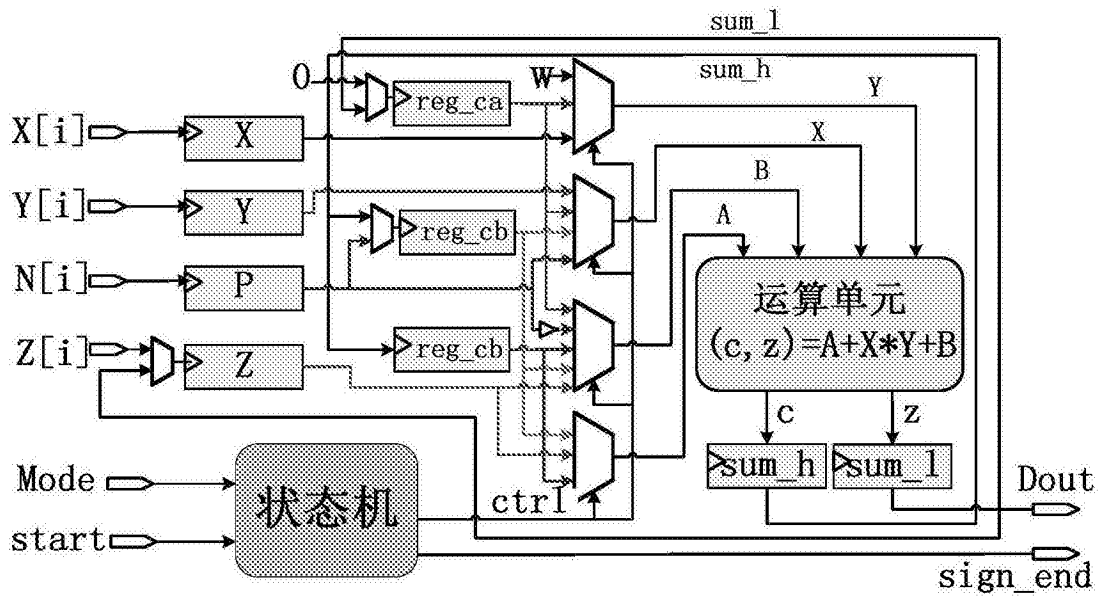


图1

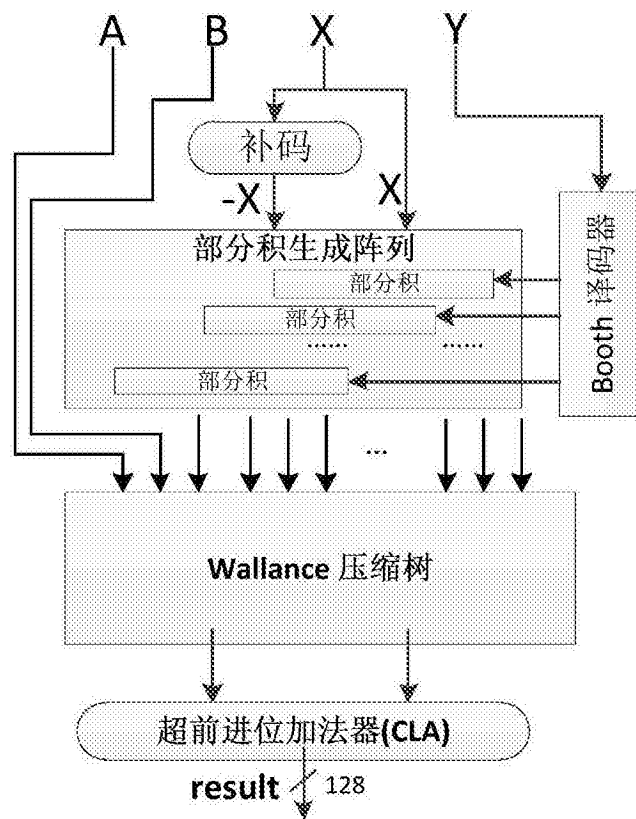


图2

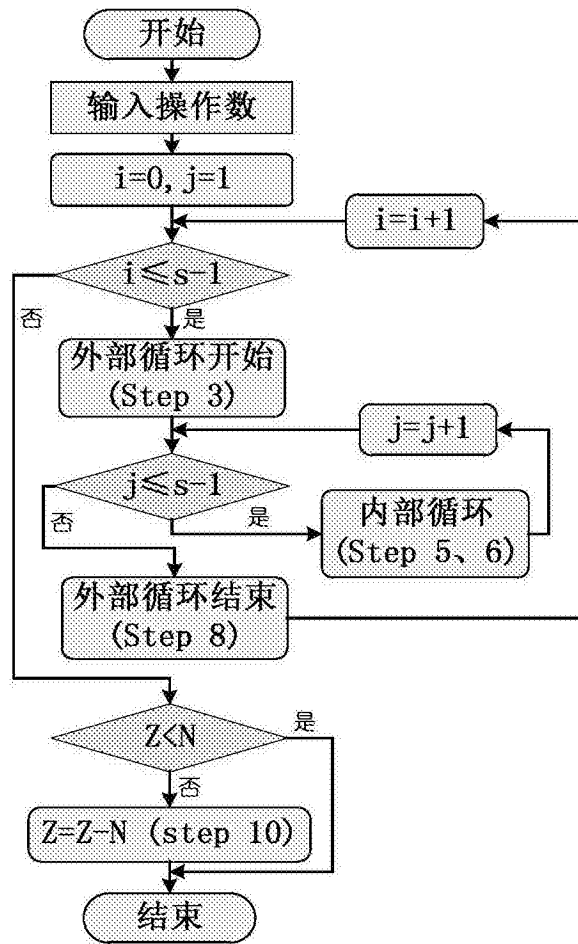


图3

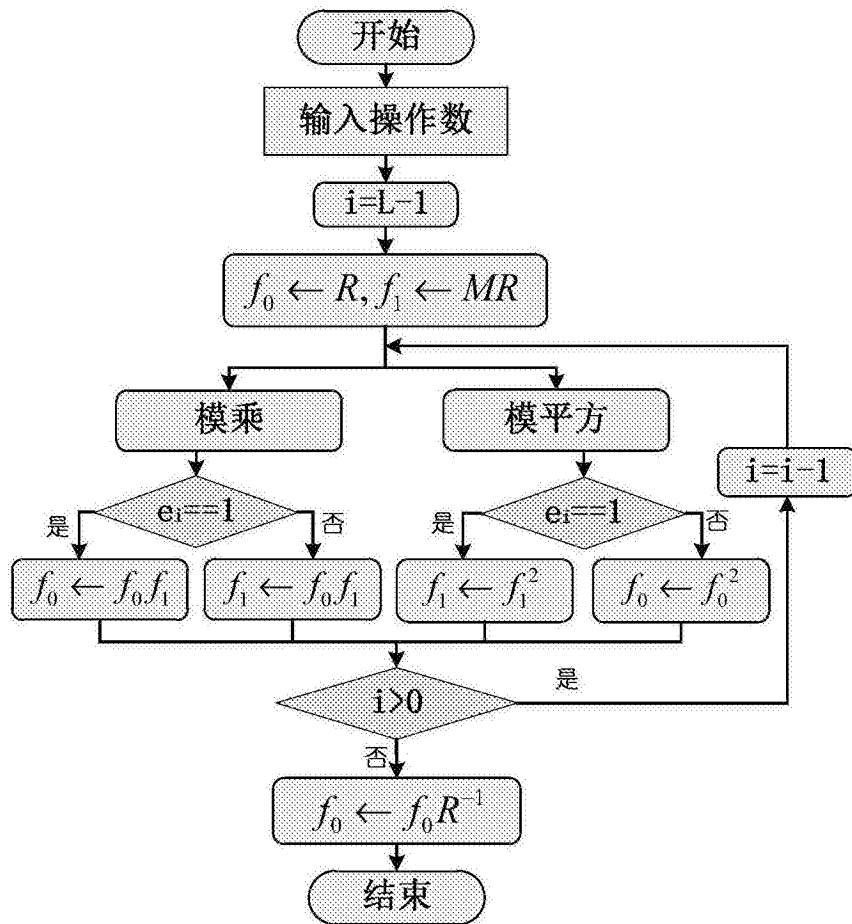


图4

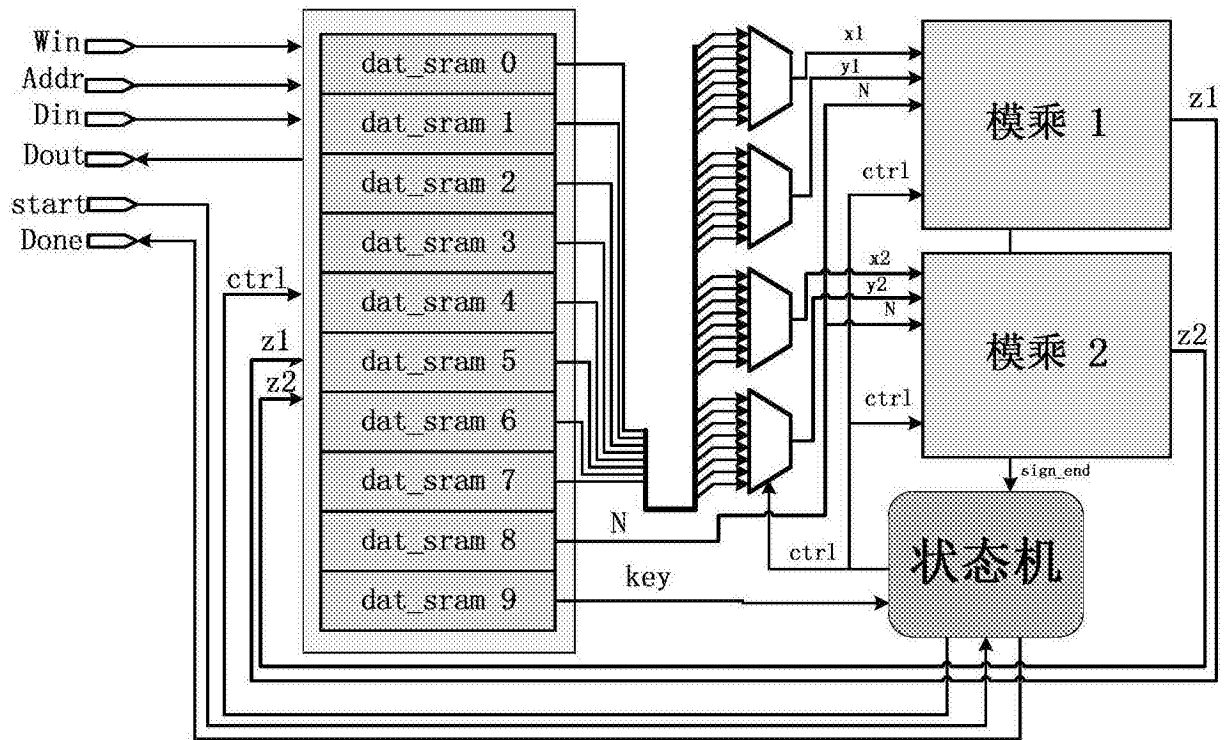


图5