第36卷 第12期 北京工业大学学报 Vol.36 No.12
2010年 12月 JOURNAL OF BEIJING UNIVERSITY OF TECHNOLOGY Dec. 2010

嵌入式椭圆曲线加密算法性能的研究与改进

易小琳，杨 峰，鲁鹏程
（北京工业大学 计算机学院，北京 100124）

摘 要：为了解决应用于嵌入式系统的椭圆曲线加密算法存在的加密速度较慢、系统开销过大等问题，改善其算法性能，提高加密速度，减少系统开销，对实际研发的嵌入式密码器中应用的椭圆曲线加密算法做了深入分析，并在此基础上提出从大数模幂子算法和模乘子算法2级改善整体算法性能的方案. 实验结果证明算法的改进效果明显.

关键词：椭圆曲线加密算法；EU-KEY；大数模幂运算；BR算法；大数模乘
中图分类号：TP301.6 **文献标志码**：A **文章编号**：0254-0037(2010)12-1722-07

1985年，Neal Koblitz[1]和Victor Miller[2]将椭圆曲线引入密码学，提出了椭圆曲线公钥密码体制，即利用椭圆曲线群上定义的离散对数系统，构造出基于离散对数的公钥体制——椭圆曲线离散对数密码体制(ECDLC). 椭圆曲线公钥密码体制安全性的理论基础是椭圆曲线离散对数问题的困难性[3]. 正因为这种安全性的保障，椭圆曲线加密(elliptic curve cryptography，ECC)算法才可能作为一种良好的公钥密码体制加密算法广泛地应用于计算机安全领域. 目前，还没有找到解决椭圆曲线离散对数难题的亚指数时间算法，这就标志着ECC算法在目前环境下的安全性是可以保障的.

另外，对于给定的安全级别，ECC算法比经典的公钥密码体制算法RSA和DL有更小的参数，而对于更高的安全级别，参数大小的差异更加明显. 这样，使用ECC计算速度更快，密钥更短，密钥证书更小，因此时间复杂度和空间复杂度更优[4]. 特别是应用到速度相对较低且资源相对稀缺的嵌入式设备中，ECC的优势更加明显.

作者提出了应用当下最为先进的椭圆曲线加密算法设计的一款嵌入式密码器(在此取名为EU-KEY)的实现方法，并重点讨论了基于EU-KEY的椭圆曲线加密算法的性能改进方法.

1 基于ECC算法密码器的设计简介

目前，应用于网络安全的加密系统种类很多. 传统的安全体系大都采用对称密码体制(例如DES、AES等)相关算法对传输的文件信息进行加密，但是随着相关破解技术的提高以及对称密码体制自身的弱点，这种基于对称密码体制的安全体系的不足日益凸显. 人们更加青睐于安全性更高、保密性更强的公钥密码体制.

公钥密码体制克服了对称密码体制中通信双方必须使用一个安全信道预先约定密码的缺点. 通信双方使用一个可公开的公钥对信息进行保密，于是私钥的保护便成为一个至关重要的问题. EU-KEY就是基于目前最为先进的公钥体制椭圆曲线算法设计而成的，并使用公钥密码体制中的加密模型. 在完整的一套加密/解密的流程中，EU-KEY的作用有2点：一是负责保存和管理私钥，这样就杜绝了因为将私钥保存在PC上而导致的木马攻击、私钥流失等危险；二是对文件进行加密和解密操作，与加密/解密有关的程序和数据都独立于PC机单独运行，最大限度地保证了信息的安全，同时也节省了PC上的系统资源.

收稿日期：2009-03-10.
基金项目：北京工业大学博士科研启动资金资助项目(52007013200501)；北京工业大学教育教学研究资助项目(007000514121).
作者简介：易小琳，女，北京人，高级工程师.

嵌入式密码器 EU-KEY 硬件电路主要包括 3 部分：ARM 核作为密码器的 CPU，负责程序的执行和运算；外部 Flash 作为拓展存储区；USB 作为通讯接口，支持 USB2.0 协议，负责与 PC 机交换数据，可以实现数据的高速传输. EU-KEY 的硬件设计简图如图 1 所示.

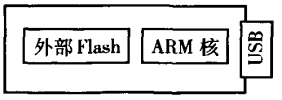

图 1　EU-KEY 的硬件设计简图

Fig. 1　EU-KEY hardware design schematic diagram

本嵌入式密码器硬件系统由艾易电子技术有限公司设计，在此基础上进行了软件系统的设计与开发. 由于本密码器属于相对低速的、资源稀缺的嵌入式系统，因此，系统开发过程中重点需要解决的问题是如何基于本设备最大限度地提高 ECC 算法的性能.

2　ECC 算法瓶颈的研究

椭圆曲线密码体制算法以其绝佳的安全性能和种种优势得到广泛应用和青睐，但是与传统的对称密码体制相比，ECC 的时间复杂度要高得多，特别是在嵌入式设备中运行 ECC 算法程序，速度问题成为制约加密器性能的最主要方面；因此，改进 ECC 的性能，提高算法的速度势在必行.

使用 ECC 算法对文件进行加密和解密，其中文件的加密时间就要占据全部时间（加密时间 + 解密时间）的 80% 以上，这主要因为 ECC 算法的加密过程包含着一个最为费时的子过程：将明文信息转化为椭圆曲线代码，在此简称为明文嵌入过程. 明文嵌入过程采用了 Kobliz 建议的一种概率式算法编码[4]，无法进行信息编码出现的可能性仅为 2^{-30}，这是绝对的小概率事件，可以通过异常处理程序加以解决[5]，因此应用 Kobliz 建议概率式编码算法是可行的.

考虑椭圆曲线

$$E: y^2 \equiv x^3 + ax + b \pmod{p} \tag{1}$$

明文信息 m 可以对应为 $E(F_p)$ 的某个 x 坐标值，但是 $m^3 + am + b$ 为二次剩余的概率仅为 1/2.

要构造 $m^3 + am + b$ 的二次剩余，可以通过在 m 上添加若干位的方法，即

$$m' = mk + j \quad (j = 0, 1, \cdots, k-1) \tag{2}$$

使得 $m^3 + am + b$ 为二次剩余，这样便可以得到 $E(F_p)$ 上的一点 (x, y). 这个过程中失败的概率，也就是 m' 无法对应到 $E(F_p)$ 上的一点的概率为 2^{-k}，其中 $(m+1)k < p$. 接收方解密后的信息 $P_m = (x, y)$ 只要通过运算

$$m = [x/k] \tag{3}$$

就可以恢复真正的明文信息. 其中“[ ]”运算为取整数运算[6].

在这个过程中，判断大数 $m^3 + am + b$ 是否是二次剩余是一个关键问题. 由欧拉准则可知：设 p 是奇素数，$\gcd(a, p) = 1$，Q_p 为模 p 的二次剩余集合，则对任意 $a \in Z_p^*$ 满足 $a \in Q_p$（即 a 为模 p 的二次剩余）的充分必要条件为[7-8]

$$a^{(p-1)/2} \equiv 1 \pmod{p} \tag{4}$$

$a \notin Q_p$（即 m 为模 p 的非二次剩余）的充分必要条件为

$$a^{(p-1)/2} \equiv -1 \pmod{p} \tag{5}$$

而如果 $p \equiv 7 \pmod 8$，那么整数 $a \in Q_p$ 的一个平方根为

$$a^{(p+1)/4} \pmod{p} \tag{6}$$

由于作者在实际课题中采用了 NIST 推荐的素数，例如 $p^{192} = 2^{192} - 2^{64} - 1 \equiv 7 \pmod 8$，所以对于所有 $a \in Q_p$，都可以通过式(6)求其平方根. 由于本设计采用雅可比-仿射坐标系下的曲线加法，所以采用欧拉定理比采用勒让德/雅可比符号在判定二次剩余方面更加适合.

由式(4)、(5)可知，要判断一个数 a 是否是二次剩余，必须用到大数的模幂运算来实现，而且如果应用式(2)来进行二次剩余的构造，那么这样的二次剩余的判断就不止 1 次. 由式(6)可知，在通过式(2)构造出一个二次剩余后，计算其平方根也要采用大数的模幂运算来实现. 然而，在实际的算法中 p 的取值为

192 位的大素数,明文 x 对应的椭圆曲线上的纵坐标值 y 也是 192 位的信息,这样 2 个 192 位的大数进行模幂运算十分耗时,而且每加密 192 位的明文信息都要进行至少 2 次(将明文 1 次嵌入椭圆曲线成功,再加上 1 次求平方根),至多 $k+1$ 次(将明文 k 次嵌入椭圆曲线,再加上 1 次求平方根)的大数模幂运算(暂不考虑嵌入失败的情况). 因此,完全有理由怀疑大数的模幂运算是影响算法加密速度、制约算法性能的瓶颈. 通过实验测试也证实了这一点,如图 2 所示.

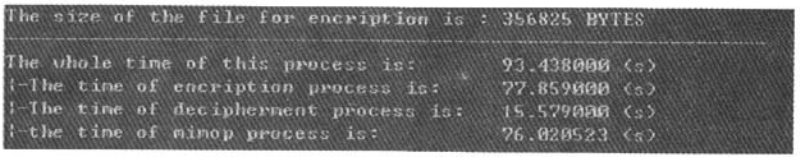

图 2 ECC 算法性能测试

Fig. 2 ECC algorithm performance testing

图 2 中,加密文件大小为 356.825 kB,加密/解密的总耗费时间为 93.438 s,加密时间 77.859 s,模幂时间 76.021 s. 不难看出,加密的时间耗费主要花在大数的模幂运算上,因此,改进 ECC 的性能关键在于改进大数的模幂运算.

3 快速椭圆曲线加密算法的实现

由于确定了 ECC 算法的瓶颈是明文嵌入过程中大数模幂运算的时间复杂度过高,因此针对此问题,本文提出了一种从大数模幂算法以及模乘算法两级改善整体算法性能的方案.

3.1 传统的大数模幂算法——BR 算法

一般最常使用的大数模幂算法是经典的 BR 算法(binary redundant algorithm). 使用 BR 算法进行模幂计算要比循环乘法快得多. 计算 x^n 只需要 $\log_2 n$ 次的乘法运算,而使用一般的循环乘法则要执行($n-1$)次的乘法运算.

BR 算法的基本思想是:在进行运算 x^e 时,将指数 e 二进制化,即将指数 e 表示为一个二元整数

$$e = e_0 + e_1 2^1 + e_2 2^2 + \cdots + e_{n-1} 2^{n-1} = \sum_{i=0}^{n-1} e_i 2^i, e_i \in (0,1) \tag{7}$$

这样计算 x^e 实际上就变为计算

$$x^e = x^{e_0 + e_1 2^1 + e_2 2^2 + \cdots + e_{n-1} 2^{n-1}} = x^{e_0} x^{e_1 2^1} \cdots x^{e_{n-1} 2^{n-1}} \tag{8}$$

从式(8)可以看出,在计算 x^e 时可从最高位为 1(二进制,即 e_i 为 1)开始进行 x 的迭代相乘(平方运算),指数位依次移向低位. 当指数为 1(二进制,即 e_i 为 1)时,除 x 自身平方外,再乘以底数 x,否则当指数为 0(二进制,即 e_i 为 0)时,只作 x 的平方运算,直到指数移到最低位为止. 采用这种方法计算 x^e,其中 e 为 n 位的二进制指数,最少只要迭代相乘($n-1$)次,最多迭代相乘 $2(n-1)$ 次. 可以归纳出以下的算法.

算法 1 计算 x^e 的 BR 算法.

1) 将指数 e 二进制化, $e = (e_l, \cdots, e_0)_2$;

2) $d \leftarrow x$;

3) 如果指数的最高位为 1, $c \leftarrow d$;否则 $c \leftarrow 1$;

4) 从次高位起:

① $c \leftarrow c * c \bmod p$;

② 如果该位为 1, $c \leftarrow c * d$;

直到最低位;

5) $x^e \leftarrow c$.

通过算法 1 可以看出，BR 算法实现大数模幂运算的效率要比循环相乘的方法高很多. 例如，对于指数和底数都是 192 位的大数，如果采用 BR 算法，每进行一次模幂计算，最多调用 382(191 ×2)次的大数模乘计算. 如果用循环相乘的方法最多要进行 $2^{192}-1$ 次的模乘计算，而 $2^{192}-1$ 是个难以想象的天文数字.

3.2 对 BR 算法的改进

传统的 BR 算法的实现过程存在很多局限. 在本研究课题中，调用大数模幂过程的次数是巨大的，然而大数模幂计算的指数却只有 2 个：一个是用来进行二次剩余判断的欧拉函数所使用的指数，另一个是用来计算整数平方根时所使用的指数. 这 2 个指数中含有较多的 1，如果使用传统的 BR 算法进行模幂运算，那么执行算法 1 的 4)的第 2 步的次数将会很多，于是执行 1 次模幂运算时就要调用更多次数的大数模乘运算.

对于指数中 1 的位数较多的情形可以采用 BR 算法的改进方案. 以往的运算都是将指数二进制化后迭代求模乘，这样指数的比特序列长度变得很大，因此要在运算中作很多次的迭代[9]. 如果 $e_i=0$，则要作 1 次平方运算和 1 次模运算；如果 $e_i=1$，则要作 1 次乘法运算加上 1 次平方运算和 1 次模运算，导致其计算效率低下. 如果将指数进行 2^k 进制化，指数 e 转换为

$$e=e_0+e_1(2^k)^1+e_2(2^k)^2+\cdots+e_{n-1}(2^k)^{n-1}=\sum_{i=0}^{n-1}e_i(2^k)^i,e\in(0,1,2,\cdots,2^k-1) \tag{9}$$

这样计算 x^e 实际上就变为计算

$$x^e=x^{e_0+e_1(2^k)^1+e_2(2^k)^2+\cdots+e_{n-1}(2^k)^{n-1}}=x^{e_0}x^{e_1(2^k)^1}\cdots x^{e_{n-1}(2^k)^{n-1}} \tag{10}$$

由式(10)可以看出，指数的长度大大减小. 在进行模幂运算时，从最高位为非 0(2^k 进制，即 e_i 非 0)开始进行 x 的迭代相乘(计算 x^{2^k})，指数位依次移向低位. 当指数为非 0(2^k 进制，即 e_i 非 0)时，除计算 x^{2^k} 外，再乘以 x^{e_i}，直到指数移到最低位为止. 可以归纳出以下的算法.

算法 2　计算 x^e 的改进的 BR 算法.

1) 将指数 e 进行 2^k 进制化，$e=(e_l,\cdots,e_0)_{2^k}$，$e_l\neq 0$；

2) $d_1\leftarrow x^{p_1}$，$d_2\leftarrow x^{p_2}$，$\cdots$，$d_n\leftarrow x^{p_n}$，$n=2^k-1$，$p_i=i$，$i=1,2,\cdots,2^k-1$，$c\leftarrow x^{e_l}$；

3) 从最高位 e_{l-1} 到最低位 e_0 重复执行

① $c\leftarrow(c*c\cdots*c) \bmod p$ (2^k 个 c 相乘)；

② 如果 $e_i=j$，$j=1,2,\cdots,2^k-1$，则 $c\leftarrow(c*d_j) \bmod p$；

4) $x^e\leftarrow c$.

从算法 2 不难看出，改进的 BR 算法之所以效率较高，原因就在于算法 2 的第 2)步. 因为第 2)步事先计算出了底数的 1，2，3，$\cdots$，(2^k-1)次方的所有值，在进行算法 2 的 3)的第②步的计算中只要 1 次乘法运算即可. 相比算法 1 的 4)的第②步的执行次数，算法 2 的乘法次数大大减少，而算法 1 的 4)的第①步与算法 2 的 3)的第①步的执行效率是一样的. 因此，当指数中 1 的位数较多时，就要反复执行算法 1 的 4)的第②步及算法 2 的 3)的第②步，执行的次数越多，算法 2 相对于算法 1 调用大数模乘的次数就越少，优势就越明显.

3.3 大数平方算法的改进

在对模幂运算进行改进后，还应进一步考虑改进大数模乘算法的效率. 通过分析可以看出，模幂运算的时间主要耗费在调用大数模乘上，改进大数模幂算法本质上就是要减少大数模乘的调用次数. 如果同时将大数模乘算法的性能提高，大数模幂的性能也会随之提高，那么整个算法的效率自然就会得以提高.

通常大数平方运算的实现采用传统的运算数扫描方式进行整数相乘，这种方法简单直观，易于实现，但是时间复杂度相对较高. 进行 n 位乘 n 位的整数乘法时间复杂度为 $O(n^2)$. 在此试图应用一种由

Karatsuba和 Ofman 提出的分而治之的算法,这种算法可将复杂性减少到 $O(n^{\log 3})$[10],但是该算法实现起来要用到大数的分治与递归调用,空间复杂度较高,在进行递归调用时也会开销大量的时间. 由于嵌入式设备的空间资源有限,因此该算法并不十分适用于嵌入式设备的椭圆加密.

在加密系统的实现过程中,利用大数模乘实现大数平方的地方很多. 如果应用运算数扫描方式来实现大数的平方固然可以,但是有些大材小用了. 因为在实现大数的平方时,有很多的交叉相乘是重复的,图 3 可以形象地说明这一点.

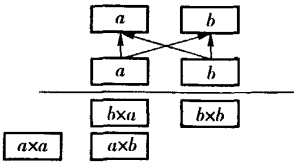

图 3 大数相乘示意(未考虑进位情况)

Fig. 3 Multiplication of large numbers schematic diagram

图 3 表明在进行大数平方时,按照传统的交叉相乘移位相加的方法,会有一些计算上的重复. 其实,$b \times a$ 和 $a \times b$ 的运算只要计算 1 次乘法即可,完全没有必要像运算数扫描方式算法那样都计算出来. 在此提出采用专门用于整数平方的算法 3. 算法 3 也可看作是运算数扫描方式算法的一个特例.

算法 3 整数的平方.

输入:整数 a;

输出:$C = a^2$;

1) $R_0 \leftarrow 0$, $R_1 \leftarrow 0$, $R_2 \leftarrow 0$.

2) 对于 k 从 0 到 $2t-2$ 重复执行:

① 对于集合$\{(i,j) \mid i+j=k,\ 0 \leqslant i \leqslant j \leqslant t-1\}$中的每个元素,重复执行:

$(UV) \leftarrow A[i] * A[j]$;

若 $i<j$, 则$(\varepsilon, UV) \leftarrow (UV) * 2$, $R_2 \leftarrow R_2 + \varepsilon$;

$(\varepsilon, R_0) \leftarrow R_0 + V$;

$(\varepsilon, R1) \leftarrow R_1 + U + \varepsilon$;

$R_2 \leftarrow R_2 + \varepsilon$.

② $C[k] \leftarrow R_0, R_0 \leftarrow R_1, R_1 \leftarrow R_2, R_2 \leftarrow 0$.

3) $C[2t-1] \leftarrow R_0$.

4) 返回 C.

算法 3 中,形如"整数 w 的赋值操作 $(\varepsilon, z) \leftarrow w$" 表示

$z \leftarrow w \bmod 2^w$,若 $w \in [0, 2w)$,则 $\varepsilon \leftarrow 0$,否则 $\varepsilon \leftarrow 1$

算法 3 并不是机械地将 2 个大整数的每一位(段)交叉相乘,再移位求和,而是限定了一个运算的范围$\{i+j=k,\ 0 \leqslant i \leqslant j \leqslant t-1\}$,这样的整数平方算法比应用运算数扫描方式进行整数平方减少了大约一半的单精度乘法. 由于程序中所采用的大整数模幂运算要反复调用大数平方运算,因此改进大整数平方算法的性能对提高整个算法的速度性能有着重要的意义.

4 性能的比较

加密系统采用算法 1 所示的大数模幂算法进行大数幂计算以及采用传统的运算数扫描方式进行大数的平方运算的性能测试结果如图 2 所示. 实践证明加密的速度不十分理想.

将算法 1 替换为算法 2,采用改进的 BR 算法进行大数的模幂运算,这时将指数进行十六进制化,测试结果如图 4 所示.

文件大小 356.825 kB 字节,加密时间 59.312 s,模幂时间 57.886 s. 加密时间加快了 18.547 s,速度提高了约 1/3. 改进的效果比较明显.

再将传统的运算数扫描方式替换为算法 3 来计算大数的平方. 测试结果如图 5 所示.

文件大小为 356.825 kB 字节,加密时间 51.922 s,模幂时间 50.565 s. 与第 1 次改进相比,加密速度又

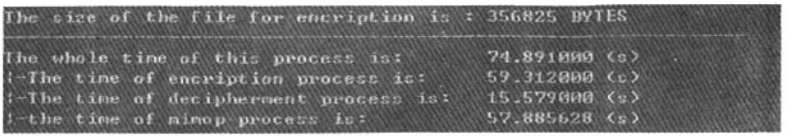

图 4　改进模幂算法后的运算结果

Fig. 4　The result of improving module exponentiation algorithm

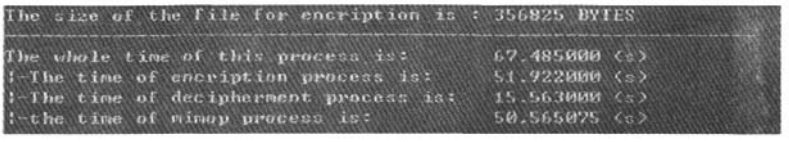

图 5　改进模幂算法和大数平方算法后的运算结果

Fig. 5　The result of improving module exponentiation algorithm & modular multiplication algorithm

加快了 7.390 s.

算法改进的比较结果见表 1.

表 1　椭圆曲线加密算法 2 次改进后的性能比较

Table 1　The comparison of ECC algorithm for twice improvement　　s

比较项目	全部流程时间	文件加密时间	模幂执行时间
最初的算法	93.438	77.859	76.021
改进的模幂算法	74.891	59.312	57.886
改进的模幂算法和大数平方算法	67.485	51.922	50.565

如表 1 所示，经过 2 次算法改进，加密 356.825 kB 大小的文件时间总共加快了 25.953 s. 相比之下，模幂算法的优化对整个程序的效率提高得更加明显，这是因为改进的模幂算法中调用大数模乘过程的次数较改进前大大减少.

5　结束语

椭圆曲线加密算法以其可靠的安全性、相对短小的密钥长度和相对低的系统开销已成为密码学研究的热点，受到国际上广泛关注，但相对于对称密码体制而言它仍避免不了公钥密码体制运算复杂度过高，系统开销过大的瓶颈. 针对目前研究开发的 EU-KEY，将椭圆曲线算法中的大数模幂算法和大数平方算法进行改进，可以大大提高整个系统的性能，降低整体算法的时间复杂度. 推而广之，将这种改进方案应用到其他的相对低速的、资源稀缺的嵌入式设备开发中也具有实际的意义.

参考文献：

[1] KOBLITZ Neal. Elliptic curve cryptosystems[J]. Mathematics of Computation, 1987, 48: 203-209.

[2] MILLER V. Use of elliptic curves in cryptography[C]// WILLIAMS H. Advances in Cryptology-CRYPTO'85, Volume 218 of Lecture Notes in Comput Sci. Berlin: Springer-Verlag, 1986: 417-428.

[3] HANKERSON Darrel, MENEZES Alfred, VANSTONE Scott. Guide to elliptic curve cryptography[M]. Heidelberg: Springer-Verlag, 2002.

[4] 张福泰，李继国，王晓明，等. 密码学教程[M]. 武汉：武汉大学出版社，2006：1-10.

[5] 邓安文. 密码学-加密演算法[M]. 北京：中国水利水电出版社，2006：189-192.

[6] KOBLITZ N. Elliptic curve cryptosystems[J]. Math Comp, 1987, 1(48): 203-209.

[7] STALLINGS William. Cryptography and network security principles and practices[M]. Beijing: Publishing House of

Electronics Industry, 2006: 210-224.

[8] 侯爱琴，高宝建，辛小龙. 信息明文嵌入椭圆曲线的改进算法及实现[J]. 计算机应用与软件，2008，25(7)：63-65.
HOU Ai-qin, GAO Bao-jian, XIN Xiao-long. An improved algorithm and its implementation for the embedding of plaintext into elliptic curve[J]. Computer Applications and Software, 2008, 25(7): 63-65. (in Chinese)

[9] 谢建全. 一种大数模幂的快速实现方法[J]. 信息安全与通信保密，2006(8)：21-27.
XIE Jian-quan. A fast calculating method for large data mode power[J]. Information Security and Communications Privacy, 2006(8): 21-27. (in Chinese)

[10] 王晓东. 计算机算法设计与分析[M]. 北京：电子工业出版社，2005：13-15.

Research and Improvement of the Performance of Embedded System-based Elliptic Curve Cryptography Algorithm

YI Xiao-lin, YANG Feng, LU Peng-cheng

(College of Computer Science, Beijing University of Technology, Beijing 100124, China)

Abstract: In order to solve problems of the low computing speed and the expensive system cost in embedded system-based elliptic curve cryptography (ECC) algorithm applications, and improve the performance of ECC algorithm, the authors conduct researches and analyses of the real encryption algorithms in an actual R & D embedded encryption device. Based on the researches and analyses, and provide a method, which modifies the module exponentiation sub-algorithm and modular multiplication sub-algorithm to speed up the whole algorithm computation. Test results show an obvious improvement.

Key words: elliptic curve cryptography algorithm; EU-KEY; module exponentiation; BR algorithm; modular multiplication

(责任编辑 梁 洁)