一种有效的Batch RSA算法的研究

李云飞[1,2,3] 柳 青[1,4] 李 彤[1,4] 郝 林[3]
(云南大学软件学院 昆明 650091)[1] (云南空管分局技术保障部 昆明 650200)[2]
(云南大学信息学院 昆明 650091)[3] (云南省软件工程重点实验室 昆明 650091)[4]

摘 要 提出了一种改进的Batch RSA算法来提升Batch RSA算法的解密性能。该改进算法结合了负载转移技术和Multi-Power RSA技术,在Batch RSA算法的指数计算阶段提升Batch RSA算法的解密性能。实验结果和理论分析表明,该改进算法使得Batch RSA算法的解密性能得到显著提升,且易于并行实现,可使基于多核平台的Batch RSA算法的整体性能得到进一步提升。

关键词 Batch RSA,Multi-Power RSA,解密,加速,并行,多核
中图法分类号 TP309.7 **文献标识码** A

Research of an Efficient Variant of Batch RSA Algorithm

LI Yun-fei[1,2,3] LIU Qing[1,4] LI Tong[1,4] HAO Lin[3]
(School of Software, Yunnan University, Kunming 650091, China)[1]
(Technical Support Department, Yunnan Air Traffic Management Branch, Kunming 650200, China)[2]
(School of Information Science and Engineering, Yunnan University, Kunming 650091, China)[3]
(Key Laboratory in Software Engineering of Yunnan Province, Kunming 650091, China)[4]

Abstract This paper aimed at speeding up Batch RSA decryption. An efficient variant of Batch RSA was proposed to improve the Batch RSA decryption performance. The improved Batch RSA variant speeds up decryption by combining the load transferring technique and multi-power RSA technique in the exponentiation phase. The experimental result and the theoretical values show that the speed of the decryption is substantially improved and the variant can be efficiently implemented in parallel and parallel implementation of the variant on multi-core devices can further improve the overall performance of Batch RSA algorithm.

Keywords Batch RSA, Multi-power RSA, Decryption, Accelerate, Parallel, Multi-core

1 引言

RSA是由Rivest,Shamir和Adleman于1978年提出的一种公开密钥体制[1],它是目前应用最为广泛的公钥密码算法,现已成为公钥密码体制的国际标准。RSA密码算法不仅可以用于数据加密,还可用于数字签名和身份认证。由于RSA密码算法计算模正整数次幂时需要耗费大量的计算时间和资源,从而降低了RSA密码算法的执行效率。特别是进行解密和数字签名时,需要进行大数的模幂运算,会耗费更多的计算资源,所以如何提升RSA密码算法的解密性能并使其尽可能少耗费资源,是密码学界一直研究的问题。其中一种加快RSA解密过程的方法是使进行模幂运算的模尽可能小。Multi-Power RSA[2-4]就是一种通过改变标准RSA大数模的结构使得模幂运算的模减小,从而加速RSA解密过程的RSA改进算法。该算法使得标准RSA算法的模结构由$N=pq$变为$N=p^{l-1}q(l>2)$,解密时进一步运用中国剩余定理[5],使得解密时的模幂运算的模由标准RSA的两个$n/2$位的素数变为Multi-power RSA时的两个n/l位的素数[2],很大程度上减少了模的位数,加快了模幂运算的速度。另一种加速RSA解密的方法是将RSA解密时的一些操作转移到RSA加密过程中,这种思想源于RSA算法的加密和验证签名的过程要比解密和进行数字签名的过程快得多[2]。目前,通过负载转移技术来提升RSA算法解密性能的方案有Rebalanced RSA[2]和RSA-S2系统[6]等。另一种加速RSA解密的方法是通过使用批处理技术来加速RSA解密。Fiat[7]第一次提出了称为Batch RSA[2,7]的批处理RSA解密加速算法。Fiat发现,若使用两个小公钥指数e_1和e_2对两个明文进行加密,则对得到的两个密文进行解密处理时,只需消耗解密一个密文的代价。通过这样的批处理技术,RSA算法的解密性能得到了提升。但文献[8]提到,当Batch RSA密钥长度相

到稿日期:2010-07-04 返修日期:2010-11-11 本文受国家自然科学基金(60963007),云南省自然科学基金(2007F008M),云南大学软件学院学科建设基金(2010KS01),云南省软件工程重点实验室开放基金(2010KS01),云南大学中青年骨干教师培养计划(21132014)资助。
李云飞(1986-),男,硕士生,主要研究方向为密码学与信息安全,E-mail:liyunfei67@163.com;柳 青(1963-),男,教授,主要研究方向为软件工程与信息安全;李 彤(1963-),男,教授,博士生导师,主要研究方向为软件工程与密码技术;郝 林(1955-),男,教授,主要研究方向为组合数学与密码学。

对较小时，其解密加速并不明显。本文针对这种情况，提出了一种改进的 Batch RSA 算法，它有效地结合了 Multi-Power RSA 的减小模幂运算模的技术和 RSA-S2 系统的负载转移技术来加速 Batch RSA 算法的解密过程，提升 Batch RSA 算法的解密性能。

理论分析和实验测试结果显示，这种改进算法使得 Batch RSA 算法解密性能得到较大提升。实验数据显示，相对于使用了中国剩余定理的标准 Batch RSA 算法，改进算法的解密平均加速比可达到 3.79(1792－3072 位)。近年来，多核处理器的计算机设备已经普及，该种处理器通过多核架构中的一整套 CPU 的并行工作来获取更高的性能[9]。多核设备为应用程序提供了并行运行的硬件平台，但现今大多数通用应用程序，特别是密码应用程序都是串行的，不能有效发挥当前已经普及的多核设备的性能。本文为了充分发挥当前多核设备的性能，从数据分解分析和功能分解分析[10]入手，发掘改进 Batch RSA 算法的并行性，使算法具有明显的并行结构，易于进行并行实现，能有效部署到当前多核处理器的计算机设备上，使多核平台上的 RSA 密码系统整体性能得到进一步提升，充分发挥当前多核计算机设备的并行处理能力。

2 RSA 改进算法

本文将改进 Batch RSA 算法称为 BEAPORSA(Batch Encrypt Assistant Multi-power RSA)算法。它结合了 Multi-power RSA 和 RSA-S2 两种算法的优点。文献[3]对 Multi-power RSA 算法进行了详细的描述。文献[11]对 RSA-S2 系统进行了改进。文献[12,13]基于改进的 RSA-S2 系统提出了 EARSA 算法。接下来，本文采用的大数模结构为 $N=p^2q$，即 l 为 3 时的 Multi-power RSA 算法的模结构。

2.1 BEAPORSA 算法

BEAPORSA 算法的整个处理过程可通过构造一棵拥有 b 个叶子节点的完全二叉树来引导。该完全二叉树内部节点必须拥有左右两个孩子节点，且树中的每个节点包含了两个值，一个是公钥指数值，一个是密文计算值，这里分别用字母 E 和字母 V 来表示。BEAPORSA 算法包括 4 个阶段：初始化阶段、Batch 树构造阶段、指数计算阶段和 Batch 树分解阶段。

1. 初始化：安全参数 n 和附加参数 b,k,c 和 l 作为密钥生成算法的输入。密钥生成算法的输出是改进算法的公钥和私钥，具体生成操作如下：

(1)产生两个互不相同且长度为⌊n/l⌋位的素数。本文中 l 取值 3，所以产生⌊$n/3$⌋位的素数 p 和 q，计算 $N=p^2q$，进一步计算 $\phi(N)=(p-1)(q-1)$。

(2) 输入 b 个两两不同且互素的公钥加密指数 $e_1,\cdots,e_b$，且 e_i 要与 $\phi(N)$ 互素，并计算 $d_i=e_i^{-1} \bmod \phi(N)$，其中 $1\leqslant i\leqslant b$ 并计算 $d=d_1\cdot d_2\cdot\cdots\cdot d_b \bmod \phi(N)$ 和 $e=\prod_{i=1}^{b}e_i$，此时 d 和 e 满足关系 $d\cdot e=1 \bmod \phi(N)$。与此同时计算 $e\_inv\_p=e^{-1} \bmod p$ 和 $p^2\_inv\_q=(p^2)^{-1} \bmod q$。

(3) 计算 $d_p=d \bmod (p-1)$ 和 $d_q=d \bmod (q-1)$，并将这两个等式转换为以下两个表达式：

$$d_p=d_{1,1}f_{1,1}+d_{1,2}f_{1,2}+\cdots+d_{1,k}f_{1,k} \bmod (p-1)$$

$$d_q=d_{2,1}f_{2,1}+d_{2,2}f_{2,2}+\cdots+d_{2,k}f_{2,k} \bmod (q-1)$$

式中，$d_{i,j}$ 和 $e_{i,j}$ 为二进制向量，$1\leqslant i\leqslant 2$ 和 $1\leqslant j\leqslant k$。这里的 $d_{i,j}$ 是 c 位的随机数，$f_{i,j}$ 是 $|n/l|$ 位的随机数。通过以上步骤改进 Batch RSA 算法的公钥是〈$N,e,e_1,\cdots,e_b,f_{1,1},\cdots,f_{1,k},f_{2,1},\cdots,f_{2,k}$〉，私钥为〈$N,d_{1,1},\cdots,d_{1,k},d_{2,1},\cdots,d_{2,k}$〉。

(4) b 个明文 $m_1,\cdots,m_b$ 分别使用相对应的公钥指数 e_i 进行加密，通过 $v_i=m_i^{e_i} \bmod N$ 计算得到密文 $v_1,\cdots,v_b$。

2. Batch 树构造：构造 Batch 树的过程是一个自底向上的过程，首先用 b 个公钥指数来标注树的 b 个叶子节点。这里使用 E 来表示公钥指数：$E\leftarrow e_i$，其中内部节点的公钥指数的值是其左右孩子公钥指数值的乘积 $E\leftarrow E_L\cdot E_R$。这样 Batch 树的根节点的公钥指数 E 值是 b 个公钥指数的乘积，即 $E\leftarrow e=\prod_{i=1}^{b}e_i$。接下来把 b 个密文信息 v_i 作为相应公钥指数叶子节点的输入，对密文信息 v 进行计算：$v\leftarrow v_L^{E_R}\cdot v_R^{E_L}$。改进 Batch RSA 算法通过使用 RSA-S2 系统将解密时的一些计算量转移到加密方，所以加密方在构造 Batch 树时，还需要计算解密方转移来的工作。因此改进 Batch RSA 算法的 Batch 树构造阶段有两个任务，第一个任务是将各个独立的密文消息 v_i 结合成为一个整体，在树的根部得到 $V=(\prod_{i=1}^{b}v_i^{1/e_i})^e$，其中 $e=\prod_{i=1}^{b}e_i$，并存储相应的公钥值 E 和密文值 V，为 Batch 树的分解做准备，以上 Batch 树的构造过程完成了第一个任务。第二个任务是计算解密方在指数计算阶段转移过来的计算量。本文为了清晰地描述改进算法的第二个任务，当参数 k 取值 2 时进行阐述。$k=2$ 使得 d_p 值变为 $d_p=d_{1,1}f_{1,1}+d_{1,2}f_{1,2} \bmod (p-1)$，$d_q$ 的值变为 $d_q=d_{2,1}f_{2,1}+d_{2,2}f_{2,2} \bmod (q-1)$，则改进算法需要计算出针对 d_p 的 $V_1=V^{f_{1,1}}=(\prod_{i=1}^{b}v_i^{1/e_i})^{e\times f_{1,1}}$ 和 $V_2=V^{f_{1,2}}=(\prod_{i=1}^{b}v_i^{1/e_i})^{e\times f_{1,2}}$ 以及针对 d_q 的 $V_3=V^{f_{2,1}}=(\prod_{i=1}^{b}v_i^{1/e_i})^{e\times f_{2,1}}$ 和 $V_4=V^{f_{2,2}}=(\prod_{i=1}^{b}v_i^{1/e_i})^{e\times f_{2,2}}$，其中 $V=(\prod_{i=1}^{b}v_i^{1/e_i})^e$ 已在 Batch 树构造的第一个任务中求出。图 1 显示了 $e_1=3$，$e_2=5$，$e_3=7$ 和 $e_4=11$ 时 4 个公钥指数($b=4$)进行 Batch 树构造得到 $V_1=(\prod_{i=1}^{b}v_i^{1/e_i})^{e\times f_{1,1}}$ 的过程。其余 V 值的计算以此类推。通过以上步骤，解密方的指数计算阶段的大部分计算量被转移到了加密方。通过这一阶段操作之后，产生的数值包括标准密文值 V 和密文向量〈V_1,V_2,V_3,V_4〉以及相应的公钥值和初始密文值，最后加密方将这些值发送到解密方作为解密方的输入值。

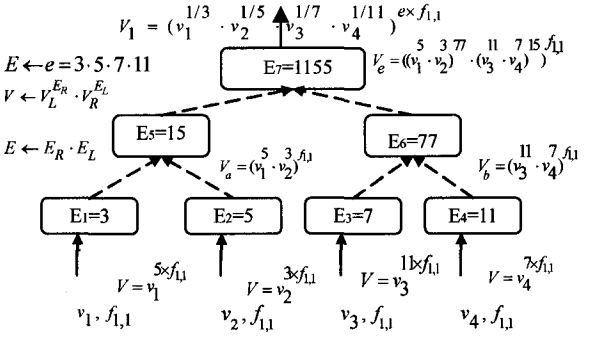

图 1 BEAPORSA 构造 Batch 树的过程

3. 指数计算：标准 Batch RSA 算法中，在 Batch 树构造阶段完成之后，根节点得到了密文值 $V=(\prod_{i=1}^{b}v_i^{1/e_i})^e$，而指数计算阶段的任务是对该密文值 V 进行 e 次根求解，即对 V 进行 d 次模幂运算 $V^d \bmod N$。最后通过该运算，得到 m 的值 $m=$

$V^{1/e}=V^d=\prod_{i=1}^{b}v_i^{1/e_i}$。然而该运算是 Batch RSA 算法中最耗时的操作，所以标准 Batch RSA 算法对该模幂运算的计算使用了中国剩余定理来加速。提出的改进算法有效地利用了负载转移技术和 Multi-power RSA 技术的优点，从两方面对 Batch RSA 算法的解密过程进行了加速。本改进算法对指数计算阶段的模幂运算也使用了中国剩余定理来进行加速，算法的整个模幂运算的过程是围绕中国剩余定理展开的。以下是改进算法通过中国剩余定理计算得到 m 值的具体过程：

(1)通过 $M_q=V^{d_q}=(V_3)^{d_{2,1}}\cdot(V_4)^{d_{2,2}} \bmod q$ 计算 M_q 值，其中 V_3 和 V_4 的值已在加密方构造 Batch 树的过程中求出，而 $d_{2,1}$ 和 $d_{2,2}$ 在初始化阶段已经计算出，保存在改进算法的私钥中。这样解密方指数计算阶段大部分计算量被转移到了加密方。该阶段得到 M_q 值只需计算两个较小的模幂运算。

(2)通过 $A_0=V^{d_p}=(V_1)^{d_{1,1}}\cdot(V_2)^{d_{1,2}} \bmod p$ 计算 A_0，其中 V_1 和 V_2 的值已在 Batch 树构造阶段中求出，而 $d_{1,1}$ 和 $d_{1,2}$ 在初始化阶段已经计算出。接下来通过 $M_p'=(V\_inv\_P)\cdot A_0$ 计算出 M_p'，其中 $V\_inv\_P=V^{-1} \bmod p$。

(3)通过 Hensel lifting 定理[14]计算出 $M_p=V^{d_p} \bmod p^2$。文献[4,15]对该定理进行了改进，使得通过该定理求解 M_p 值的过程不需一次模逆运算操作。本文的改进算法是基于改进的 Hensel lifting 定理来求解 M_p 的，通过改进定理计算 M_p 的步骤如下：

输入值：A_0,P,V,e_inv_p,M_p'；输出值：M_p

(a) $p^2=p\times p$

(b) $F=A_0^5 \bmod p^2$

(c) $E=V-F \bmod p^2$

(d) $B=E\times M_p'\times(e\_inv\_p) \bmod p^2$

(e) $M_p=A_0+B \bmod p^2$

通过以上 5 个步骤可以求出 M_p 值，对于该求解过程的正确性将在下节进行分析。

(4)根据中国剩余定理，计算出 $W=M_q-M_p \bmod q$ 并进一步计算 $W=W\times(p^2\_inv\_q) \bmod q$。

(5)最后通过中国剩余定理合并 M_p 值和 M_q 值，最终得到 m 值，$m=M+M_p \bmod N$，其中 $M=W\cdot p^2 \bmod N$。改进算法通过以上 5 个步骤能高效率地计算出 m 值。

4. Batch 树分解：该阶段是把 m 分解成为左右孩子的乘积并且最终在 Batch 树的叶子节点上得到各个密文 V 的明文。Batch 树的分解是一个自顶向下的过程，具体的分解方法见文献[7]。当分解过程结束时，每个叶子节点的明文 m 都被解密出来。

图 2 显示了改进 Batch RSA 算法的加密和解密的处理流程。如图 2 所示，在加密方完成的阶段为 Batch 树构造阶段，在解密方完成的阶段包括指数计算阶段和 Batch 树分解阶段。从图中可以看出，BEAPORSA 算法的核心部分为指数计算阶段。整个指数计算阶段围绕中国剩余定理展开。从图中可以看出整个阶段由 3 部分组成：第 1 部分计算 M_q，第 2 部分计算 M_p，第 3 部分使用中国剩余定理合并 M_q 和 M_p 得到 m。通过使用 RSA-S2 技术将解密方的指数计算阶段的大部分计算量转移到了加密方的 Batch 树构造阶段，提升了 Batch RSA 算法的解密性能。通过以上各个部分，不难看出改进算法 BEAPORSA 算法有效地结合了 Multi-power RSA 算法和 RSA-S2 系统的优点，既减小了大数模幂运算的模，又将解密部分的一些计算工作转移到加密方，从两方面加速了 Batch RSA 算法的解密过程，使 Batch RSA 算法解密性能得到显著提升。

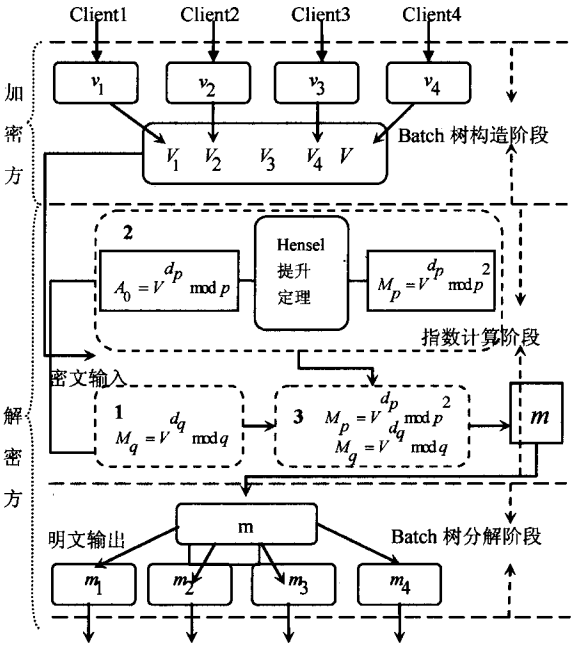

图 2 BEAPORSA 算法加密与解密处理流程示意

2.2 改进算法解密正确性分析

本节将对 BEAPORSA 算法解密过程的正确性进行分析和证明。首先本文将结合文献[7,15]和图 2 对 BEAPORSA 算法处理过程中的各个阶段进行总结：

1) 初始化阶段和 Batch 树的构造阶段可以总结为以下两个式子：

$$e=\prod_{i=1}^{b}e_i,V=\prod_{i=1}^{b}v_i^{e/e_i} \bmod N \tag{1}$$

图 2 显示了 Batch 树构造阶段为加密方的阶段。该阶段需要计算指数计算阶段转移来的计算量。

2) 指数阶段主要完成了

$$m=V^{1/e} \bmod N \tag{2}$$

该阶段是 BEAPORSA 算法改进的关键部分，该阶段处在解密方。改进算法有效利用了 RSA-S2 系统和 Multi-power RSA 的优点对Batch RSA 算法的解密过程进行了加速。

3) 最后 Batch 树的分解可总结为以下式子[2,15]：

$$m_i=\frac{V^{\delta_i}}{v_i^{(\delta_i-1)/e_i}\cdot\prod_{j=1,j\neq i}^{b}v_j^{\delta_i/e_j}} \bmod N \tag{3}$$

从以上总结及图 2 所示的改进算法的解密部分，改进算法的解密过程的正确性集中在 2)和 3)两部分的正确性分析上。本文将对指数计算阶段和 Batch 树分解阶段的正确性进行分析。若保证了这两部分的正确性，则改进算法的整个解密过程的正确性也将得到保证。具体分析过程如下：

(1)改进算法指数计算阶段得到 m 值的正确性分析

选取参数 $k=2$ 时的改进算法进行分析。整个改进算法的指数计算阶段是围绕中国剩余定理展开的，本节将按照图 2 所示的改进算法指数计算阶段的流程图，从 3 个部分对改进算法解密过程的正确性进行分析：改进 Batch RSA 算法指数计算阶段是基于 $M_q=V^{d_q} \bmod q$ 和 $A_0=V^{d_p} \bmod p$ 同余式

展开的。根据以上的参数值，N 由 p,q 两个素数组成，每个素数与私钥 d 进行运算得到 d_p 和 d_q，并使用多元一次不定方程[5]进行划分得到以下等式：

$$d_p=d_{1,1}f_{1,1}+d_{1,2}f_{1,2} \bmod (p-1)$$

式中，$d_p \equiv d \bmod p-1$。

$$d_q=d_{2,1}f_{2,1}+d_{2,2}f_{2,2} \bmod (q-1)$$

式中，$d_q \equiv d \bmod q-1$。

改进 Batch RSA 算法在 Batch 树构造阶段得到密文 $V=(\prod_{i=1}^{b} v_i^{1/e_i})^e$，其中 $e=\prod_{i=1}^{b} e_i$，并且得到密文向量 Z：$\langle V_1,V_2,V_3,V_4\rangle$，其中 $V_1=V^{f_{1,1}}$，$V_2=V^{f_{1,2}}$，$V_3=V^{f_{2,1}}$，$V_4=V^{f_{2,2}}$。

1）输入 Z 向量得到 $M_q=V^{d_q} \bmod q$ 正确性分析

改进算法的加密操作得到 $V_3=V^{f_{2,1}}$ 和 $V_4=V^{f_{2,2}}$ 两个密文并作为解密方的指数计算阶段的输入。Batch RSA 改进算法在指数计算阶段中首先计算 $M_q=V_3^{d_{2,1}} \cdot V_4^{d_{2,2}} \bmod q$。根据以上的输入值可得 $M_q=(V^{f_{2,1}\cdot d_{2,1}} \bmod q) \cdot (V^{f_{2,2}\cdot d_{2,2}} \bmod q)$，进一步推出 $M_q=V^{d_q}=V^{d_{2,1}f_{2,1}+d_{2,2}f_{2,2}} \bmod q$，其中 $d_q=d_{2,1}f_{2,1}+d_{2,2}f_{2,2} \bmod (q-1)$ 已在密钥生成算法中定义。所以通过以上密文向量输入得到 $M_q=V^{d_q} \bmod q$ 值的过程是正确的。该部分对应图 2 中 BEAPORSA 指数计算阶段的第 1 部分。

2）BEAPORSA 算法得到 $M_p=V^{d_p} \bmod p^2$ 正确性分析

Batch RSA 改进算法在求 $M_p=V^{d_p} \bmod p^2$ 首先计算 $A_0=V^{d_p}=(V_1)^{d_{1,1}} \cdot (V_2)^{d_{1,2}} \bmod p$，接下来计算出 $M'_p=(V\_inv\_P) \cdot A_0$，其中 $V\_inv\_P=V^{-1} \bmod p$。根据同余运算的性质，V_p 和密文 V 以及指数计算阶段结果 $m=\prod_{i=1}^{b} v_i^{1/e_i}$ 的关系是 $V_p=(\prod_{i=1}^{b} v_i^{1/e_i})^e=m^e \bmod p^2$，其中 $e=\prod_{i=1}^{b} e_i$；M_p 和 m 的关系是 $M_p=m \bmod p^2$，密文 V，V_p 和明文 M_p 的关系是 $V \bmod p^2=V_p=M_p^e \bmod p^2$。接下来对通过 Hensel lifting 定理得到 M_p 的正确性进行证明。

证明：首先用 P 进制表达式来描述 $M_p \bmod p^2$ 得到 M_p

$$M_p=A_0+P \cdot A_1 \bmod p^2 \tag{4}$$

若 $B=P \cdot A_1$，则式(4)可写为 $M_p=A_0+B \bmod p^2$。由于 $V \bmod p^2=M_p^e \bmod p^2$，则由式(4)可推出：

$$V \bmod p^2=(A_0+P \cdot A_1)^e \bmod p^2 \tag{5}$$

将式(5)右部分展开可进一步推出：

$$V=\sum_{j=0}^{e}\binom{e}{i} \cdot A_0^i \cdot P^{e-j} \cdot A_1^{e-j} \bmod p^2 \tag{6}$$

式中，所有 p 的指数大于等于 2 的项模上 p^2 都等于 0，则由式(6)可进一步推出：

$$V=A_0^e+e \cdot P \cdot A_0^{e-1} \cdot A_1 \bmod p^2 \tag{7}$$

将式(7)模上 p 则可得到以下等式：

$$V=A_0^e \bmod p \tag{8}$$

由此可计算出 A_0。由于 $ed=1 \bmod \phi(N)$，则可推出 $A_0=V^d \bmod p$。由于 $d_p \equiv d \bmod (p-1)$，则必存在整数 k，满足 $d=d_p+k(p-1)$，可推出 $A_0 \equiv V^{d_p} \cdot (V^{p-1})^k \bmod p$。根据 Fermat 小定理[5]和同余定理[5]可以推出 $V^{p-1} \equiv 1 \bmod p$，则有 $A_0 \equiv V^{d_p} \cdot (1)^k \bmod p$，所以可以得到：

$$A_0 \equiv V^{d_p} \bmod p \tag{9}$$

改进算法的关键之处在于对式(9)的改进。d_p 值被划分为 $d_p=d_{1,1}f_{1,1}+d_{1,2}f_{1,2} \bmod (p-1)$，其中在加密方已得到 $V_1=V^{f_{1,1}}$ 和 $V_2=V^{f_{1,2}}$，可推出 $V_1 \bmod p \equiv V^{f_{1,1}} \bmod p$，同时也可得到 $V_2 \bmod p \equiv V^{f_{1,2}} \bmod p$。根据同余关系，进一步计算 $A_0=(V_1^{d_{1,1}} \bmod p) \cdot (V_2^{d_{1,2}} \bmod p)$，该式进一步等价于 $A_0=(V^{f_{1,1}\cdot d_{1,1}} \bmod p) \cdot (V^{f_{1,2}\cdot d_{1,2}} \bmod p)$，进一步推出 $A_0=V^{f_{1,1}\cdot d_{1,1}+f_{1,2}\cdot d_{1,2}} \bmod p=V^{d_p} \bmod p$，所以改进后的算法得到 A_0 是正确的。接下来计算 $M_p'=(V\_inv\_P) \cdot A_0$。

此时由(7)式可进一步推出：

$$V-A_0^e=e \cdot P \cdot A_0^{e-1} \cdot A_1 \bmod p^2 \tag{10}$$

式中，$F=A_0^e \bmod p^2$，$E=V-F \bmod p^2$，则可以推出 $E=e \cdot A_0^{e-1} \cdot P \cdot A_1 \bmod p^2$。根据同余定理，该式可进一步推出：

$$B=P \cdot A_1 \equiv E \cdot (e \cdot A_0^{e-1})^{-1} \bmod p^2 \tag{11}$$

由于 e 和 $A_0 \equiv V^{d_p} \bmod p$ 与 p^2 互素，可推出 $e \cdot A_0^{e-1} \bmod p^2$ 存在逆元，所以式(11)成立。由式(11)可进一步推出：

$$P \cdot A_1=E \cdot e^{-1} \cdot A_0^{1-e} \bmod p^2 \tag{12}$$

式中，A_0^{1-e} 可通过以下式子计算得到：

$$A_0^{1-e}=A_0 \cdot (A_0^e)^{-1} \bmod p=V^{d_p} \cdot (V^{-1}) \bmod p=M_p' \tag{13}$$

式中，$M_p'=(V\_inv\_P) \cdot A_0=V^{d_p} \cdot (V^{-1}) \bmod p$ 已经计算出。将式(13)和 e_inv_p 的值代入到式(12)中，则可以求出 B 的值。根据式(4)可最终得出 M_p 的值 $M_p=A_0+B \bmod p^2$。所以由以上证明可知得到 M_p 的过程是正确的。该部分证明对应图 2 中指数计算阶段的第 2 部分的内容正确性分析。

3）BEAPORSA 算法使用中国剩余定理得到 m 正确性分析

通过以上 1)和 2)两个部分得到 M_p 和 M_q 值。现可以通过中国剩余定理将这两个值进行合并，计算 $V=M_q-M_p \bmod q$ 和 $V=V\times(p^2\_inv\_q) \bmod q$ 的值。最后通过 $M=V \cdot p^2 \bmod N$ 和 $m=M+M_p \bmod N$ 得到 m。该部分对应了图 2 中指数计算阶段的第 3 部分的内容正确性的分析。

(2)Batch 树分解得到各个明文的正确性分析

本小节将结合文献[7,15]对改进 Batch RSA 算法解密得到明文的正确性进行分析。在 Batch 树分解阶段，各个公钥值存在关系 $\delta_i=1 \bmod e_i$ 和 $\delta_i=0 \bmod e_j$，则可进一步推出，存在整数 k_i 和 $k_{i,j}$，可推出关系 $\delta_i=1+k_i \cdot e_i$ 和 $\delta_i=k_{i,j} \cdot e_j$。将这两个结果代入到式(6)中，可得

$$\frac{V^{\delta_i}}{v_i^{(\delta_i-1)/e_i} \cdot \prod_{j=1,j\neq i}^{b} v_j^{\delta_i/e_j}}=\frac{V^{\delta_i}}{v_i^{k_i} \cdot \prod_{j=1,j\neq i}^{b} v_j^{k_{i,j}}} \bmod N \tag{14}$$

并且由式(4)可计算出 $V^{\delta_i}=(\prod_{j=1}^{b} v_j^{1/e_j})^{\delta_i} \bmod N$，进一步得到 $V^{\delta_i}=v_i^{\delta_i/e_i} \cdot \prod_{j=1,j\neq i}^{b} v_j^{\delta_i/e_j} \bmod N$。再次使用 δ_i 的关系式，可以得到 $\delta_i/e_i=1/e_i+k_i$ 和 $\delta_i/e_j=k_{i,j}$。将这两个式子代入上式，可以得到 $V^{\delta_i}=v_i^{1/e_i} \cdot v_i^{k_i} \cdot \prod_{j=1,j\neq i}^{b} v_j^{k_{i,j}} \bmod N$。将该式代入到式(14)可得

$$\frac{V^{\delta_i}}{v_i^{(\delta_i-1)/e_i} \cdot \prod_{j=1,j\neq i}^{b} v_j^{\delta_i/e_j}}=\frac{v_i^{1/e_i} \cdot v_i^{k_i} \cdot \prod_{j=1,j\neq i}^{b} v_j^{k_{i,j}}}{v_i^{k_i} \cdot \prod_{j=1,j\neq i}^{b} v_j^{k_{i,j}}} \bmod N$$

由上式可以进一步推出：

$$m_i = v_i^{1/e_i} = \frac{v_i^{1/e_i} \cdot v_i^{k_i} \cdot \prod_{j=1,j\neq i}^{b} v_j^{k_i,j}}{v_i^{k_i} \cdot \prod_{j=1,j\neq i}^{b} v_j^{k_i,j}} \bmod N$$

所以通过以上批处理解密的方式得到明文是正确的。通过以上两部分的正确性分析，可得到 Batch RSA 改进算法的解密过程是正确的。

2.3 安全性和参数选取分析

BEAPORSA 算法的安全性依赖于 3 个方面：第 1 是大数模 N 的素数因子的安全性；第 2 是多个小私钥 $d_{i,j}$ 的安全性，其中 $1\leqslant i\leqslant 2$ 和 $1\leqslant j\leqslant k$；第 3 是 Batch RSA 算法的多个小公钥指数的安全性。

首先分析大数模 $N=p^{l-1}q$ 的素数因子的安全性。改进算法的安全性依赖于对模形式为 $N=p^{l-1}q$ 的整数因式分解的难度，最快的因式分解算法对 N 的这种特殊结构并没有有效的搜索、破解方法[2]。然而，必须确保 N 的素数因子不能在椭圆曲线方法[16]和改进椭圆曲线方法[17]的范围之内。当前 256 位的素因子被认为是在椭圆曲线方法的范围内[2]，所以针对 1024 位的模，l 值至多等于 3，即 $N=p^2q$。当 N 为 1024 位时，文献[2]注意到 LFM[18]方法也不能有效地分解形式为 $N=p^2q$ 的模。所以 BEAPORSA 改进算法在 l 参数值的选取上要保证 N 的每个素数因子位数都必须大于 256 位。本文在接下来的实验中，各测试算法的 l 的取值都远大于 341 位，保证了改进算法的高安全性。

在确保 l 取值使得 BEAPORSA 算法不在椭圆曲线范围的前提下，还需要保证通过暴力搜索得到 d_p 和 d_q 的值是不可行的。若存在某种搜索方法能通过公钥向量提供的信息 $\langle N,e,f_{1,1},\cdots,f_{1,k},f_{2,1},\cdots,f_{2,k}\rangle$ 搜索到所有可能的 $d_{i,j}$ 值，其中 $1\leqslant i\leqslant 2$ 和 $1\leqslant j\leqslant k$，则这种方法可以还原出 d_p 和 d_q 的值。由于 BEAPORSA 算法一个 d_p 有 k 个 c 位向量元素，所以要保证 $2^{k\times c}$ 有足够大的搜索空间来阻止穷尽搜索，保证改进 Batch RSA 算法的安全性[11]。而要在改进算法中穷尽出 $2^{k\times c}$ 的所有可能值等价于在对称密码系统搜索所有可能的密钥值[11,19]，所以改进 Batch RSA 算法的密钥长度提供的安全强度要与相应对称密码系统的密钥长度提供的安全强度等价[19]。文献[19]提出了相应对称密码系统密钥长度的公式。根据文献[19]中的公式，RSA 密码算法 1024 位和 1536 位的安全强度等价于对称密码系统中密钥值为 72 位和 80 位的强度[11,19]。所以 BEAPORSA 算法在选取参数 c 值和 k 值时，不要低于相应的对称码系统的密钥值长度。本文在接下来的实验测试中，从 1024 位到 3072 位范围内，k 取值为 2，c 取值为 64 位、128 位以及 256 位，从而确保了 $k\times c$ 的取值远远大于文献[19]提供公式计算的对应对称密码系统密钥长度的强度，保证了改进 Batch RSA 算法的高安全性。

在确保以上两方面安全性的同时，还需考虑 Batch RSA 算法的多个小公钥指数的安全性。Batch RSA 算法使用小公钥指数会带来某些问题：如果一个相同明文被不同的并且互素的公钥指数模上相同的大数模 N 进行加密得到密文，则通过某些方法能够从密文中重构出明文[7]。所以在进行批处理 RSA 加解密处理时，需要保证处理的各个明文信息内容不同。因此若能在需要处理的信息中加入随机值，则能有效阻止各种小公钥指数攻击[7]。所以 BEAPORSA 算法若能满足以上 3 个方面的安全需求和安全参数选取，则能有效保证改进 Batch RSA 算法的高安全性。

2.4 改进算法并行性分析

随着多核处理器时代的到来，绝大多数计算机都拥有了运行并行程序的硬件环境。但当前大部分的应用程序都是串行的，不能充分发挥多核设备的并行处理能力。为了充分发挥当前多核设备的并行处理能力和提升程序运行的性能，必须开发并行程序。而仅当问题具有可并行性时，即多个活动或任务能够在同一时间内同时执行时，该程序才能被并行处理。所以在设计并行程序时，首先要寻找相关程序的并行性[10,12,20,21]。而寻找并行性的第一步是将问题分解为多个能够并发执行的元素。分解可以从两方面去分析：数据分解和任务分解[10]。本文从数据分解和任务分解两方面对 BEAPORSA 算法进行并行性分析，充分挖掘算法的并行特征。

首先进行数据分析。改进 Batch RSA 算法的主要数据包括公钥、私钥、明文和密文。和标准 Batch RSA 算法的不同之处在于，改进算法的公钥变为了 $\langle N,e,e_1,\cdots,e_b,f_{1,1},\cdots,f_{1,k},f_{2,1},\cdots,f_{2,k}\rangle$，私钥变为了 $\langle N,d_{1,1},\cdots,d_{1,k},d_{2,1},\cdots,d_{2,k}\rangle$，之前每个加密方的单个 e 值被划分为多个值，私钥也被划分为多个更小的值。从私钥和公钥的数值中可以看出，整个公钥和私钥数据可划分为一个 $2\times k$ 的矩阵[12]，其中包括 d_p 和 d_q 的值及其 k 个划分值，每个数据单元包括了私钥和公钥值。其中的每个数据单元之间相互独立，为以下功能分解奠定了基础。仅当数据块能被相对独立地操作时，与数据相关的计算才能够高效地执行。改进 Batch RSA 算法的分解和并行的粒度可以通过 k 值来设定。通过分析可知改进算法的数据有很强的独立特性，每个数据单元之间互不影响，易于并行实现。

接下来进行任务分解分析。该分解是将问题看作是一个指令流，指令流中的指令能够被分解为多个称为任务的序列[10]。改进 Batch RSA 算法主要任务都集中在加密和解密两个操作上，其中加密主要在 Batch 树构造阶段完成，而解密主要在指数计算阶段和 Batch 树分解阶段完成。标准 Batch RSA 算法每个客户端的加密方进行加密时只有一次大数模幂运算，而在指数计算阶段利用中国剩余定理来加速 Batch RSA 指数运算的方法有两次大数模幂运算，运算较为单一，所以指令流很难被分解为多个任务的序列，不易进行并行实现。而改进 Batch RSA 算法的每个加密方需进行 $2\times k+1$ 次模幂运算，解密方的指数计算阶段需进行 $2\times k$ 次模幂运算。标准 Batch RSA 算法的大数模幂运算被分解为多个较小的模幂运算，加快了解密速度，且指令流被分为多个任务，易于进行并行实现。图 3 显示了拥有 4 个加密客户端的改进 Batch RSA 算法构造 Batch 树即加密数据的并行操作示意图。从图中可以看出，4 个加密客户端可独立地分别进行并行处理。以上数据分解为功能分解奠定了基础。各个输入数据值的加密操作能够同时执行，各个任务的操作都独立于其他任务的操作。通过加密方的加密操作得到一个密文向量，并且将其作为解密方的输入数据。解密方在指数计算阶段输入密文向量以及对应的密钥向量进行模幂运算。各个解密的模幂运算操作都独立于其它任务的操作，可并行进行处理，且可按行并行和按列并行。由以上任务分解分析可以看出，改进算法在任务操作上也具有很强的并行性。

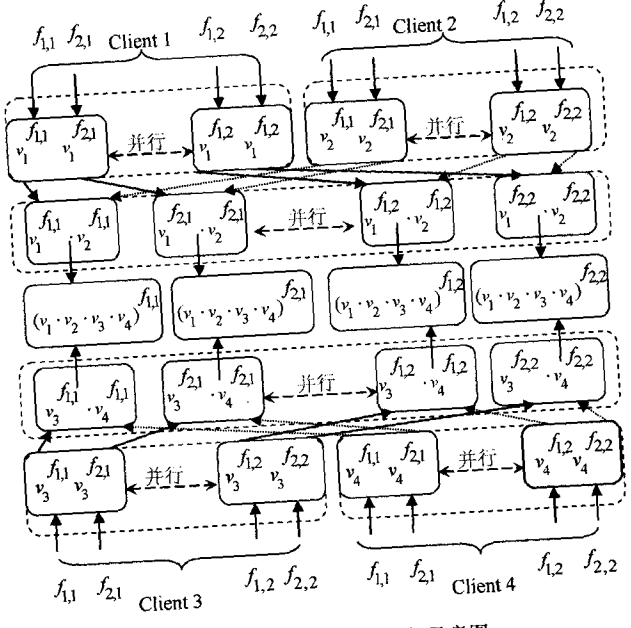

图 3　改进算法加密方并行示意图

通过以上数据分解分析和任务分解分析，可知改进 Batch RSA 算法具有很强的并行特征，并且可以根据当前的硬件平台处理器的核数，调整并行粒度，以达到最佳并行效果。对于改进算法的并行实现，可通过 OpenMP[10,20] 和 OpenSSL[21] 加密库相结合来实现。

2.5　性能分析和相关问题讨论

接下来对改进 Batch RSA 算法相对于使用了中国剩余定理加速解密的标准 Batch RSA 算法的理论加速比进行分析。标准 RSA 算法使用中国剩余定理计算 $C^d \bmod N$ 要耗时 $O(\log^3 N/2)$[13]，则改进算法相对于标准 Batch RSA 理论加速比约为：

$$\frac{2 \cdot \log^3 N/2}{2 \cdot k \cdot 2 \cdot c \cdot \log^2 N/l} = \frac{l^2 \cdot \log N}{16 \cdot k \cdot c}$$

式中，k 是指 d_p 和 d_q 被划分后的向量的元素个数，c 是解密向量中向量元素的位数，l 是构成模 $N(N=p^{l-1}q)$ 的参数。如 $n=\log N$ 为 2048 位，$l=3$，$k=2$ 和 $c=128$ 位，改进算法的理论解密加速比为 4.50，在以下实验测试中得到的解密加速比是 3.88。

BEAPORSA 算法在 Batch RSA 解密和产生数字签名方面有较好的性能。改进算法从两个方面优化了 Batch RSA 系统的性能，但由于算法的改进是通过将解密方的一些计算量转移到加密方来完成，会使得加密端的计算量增加。但本文认为，当前计算机设备性能都比较优秀，加密耗时是可承受或忽略的。对于高安全的服务，可能会受到一些影响，但由于当前计算机设备大部分已是多核，且改进算法易于进行并行实现，可充分发挥多核设备的性能来加速加密的过程，使得加密耗时降到最低。

3　实验结果及分析

Batch RSA 改进算法和相应测试算法是基于 OpenSSL (0.9.8k)库和 OpenMP 库实现的。实验测试平台为：操作系统 Windows XP，处理器 Intel Pentium 双核 1.73GHz，内存 1GB。在接下来的实验测试中改进算法 BEAPORSA 的参数取值，$l=3$，$k=2$，$b=4$。本文将参数 c 取值为 128 位的 BEAPORSA 算法称为 BEA1PORSA，将 c 取值为 64 位和 256 PORSA。BEAPPORSA 表示并行优化后的改进算法。长度为 1792，2048，2304，2560，2816 和 3072 位的密钥被用来测试改进 Batch RSA 算法 BEAPORSA 在当前和未来安全强度下的性能状况。

由于改进算法在密钥长度 1792 位之前进行解密，其解密时间都不到 1 毫秒。为了易于和其它类型的 RSA 算法进行比较，本文从 1792 位开始比较。表 1 显示了 4 种不同类型的 Batch RSA 算法、6 种不同长度模时相应的解密密文的平均耗时情况，可知 BEA6PORSA 算法平均耗时最少。

表 1　各类型 Batch RSA 解密消耗情况(ms)

算法类型	密钥长度(位/bit)					
	1792	2048	2304	2560	2816	3072
BEA6PORSA	15	16	16	25	28	31
BEA1PORSA	16	16	22	29	31	39
BEA2PORSA	24	27	31	38	44	47
Batch RSA	41	62	70	94	110	125

表 2 显示了各个类型的改进 Batch 算法从 1792 位到 3072 位范围内相对于标准 Batch RSA(使用中国剩余定理进行解密)的解密时的加速比。从表 2 中可以看出，BEA6PORSA 算法的加速比最高，其平均加速比为 3.79。

表 2　改进算法解密时相对于标准 Batch RSA 的加速比

算法类型	密钥长度(位/bit)					
	1792	2048	2304	2560	2816	3072
BEA6PORSA	2.73	3.88	4.38	3.76	3.93	4.03
BEA1PORSA	2.56	3.88	3.18	3.24	3.55	3.21
BEA2PORSA	1.71	2.30	2.26	2.47	2.50	2.66

图 4 显示了参数值不同的改进算法解密时密钥长度和加速比的变化关系。从图中可以看出，3 条曲线在密钥长度不算太大时各算法的解密加速比呈现上升的趋势，解决了之前文献[8]中提出的当 Batch RSA 算法密钥长度小时解密性能和效率不高的问题。而当密钥长度变得较大时，各算法解密加速比提升趋势变得比较平缓。这种现象出现的原因是本文提出的改进算法是通过对算法解密中最耗时的指数运算操作进行改进和加速来提升 Batch RSA 算法的解密性能，没有考虑那些耗时较少的阶段(Batch 树分解阶段)。当密钥长度不太大时，Batch 树分解阶段耗时会很小。但当密钥长度很大时，改进算法的 Batch 树分解阶段也会耗费很多时间，所以出现了图 4 所示的密钥长度和加速比的变化关系。但就在这种情况下，改进算法($c=128$，64 位)也获得了超过 3 的平均加速比的提升。从图 4 中还可以看出，当 k 值固定时，改进算法的 c 值越小，其获得的加速比越高。

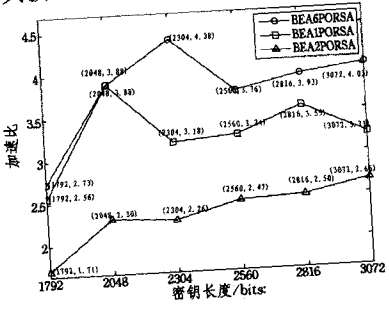

图 4　各类型改进算法解密时密钥长度和加速比变化关系

表 3 显示了并行前和并行后 BEAPORSA 算法加密时的

(下转第 139 页)

Heidelberg:Springer-Verlag,2005:359-367

[7] Wu W,Mu Y,Susilo W,et al. Identity-based proxy signature from pairings[C]// Proceedings of the 4th International Conference on Autonomic and Trusted Computing. Berlin/Heidelberg:Springer-Yerlag,2007:22-31

[8] Bellare M,Rogaway P. Random oracles are practical:a paradigm for designing efficient protocols[C]// Proceedings of the First ACM Conference on Computer and Communications Security. New York:ACM,1993:62-73

[9] 冯登国. 可证明安全性理论与方法研究[J]. 软件学报,2005,16(10):1743-1756

[10] Paterson K G,Schuldt J C N. Efficient identity-based signatures secure in the standard model[C]// Proceedings of the 11th Australasian Conference on Information Security and Privacy. Berlin/Heidelberg:Springer-Vedag,2006:207-222

[11] 李明祥,韩伯涛,朱建勇,等. 在标准模型下安全的基于身份的代理签名方案[J]. 华南理工大学学报:自然科学版,2009,37(5):118-129

[12] Waters B. Efficient identity-based encryption without random oracles[C]// Proceedings of Eurocrypt. Berlin/Heidelberg:Springer-Verlag,2005:114-127

[13] 李继国,姜平进. 标准模型下可证安全的基于身份的高效签名方案[J]. 计算机学报,2009,32(11):2131-2136

(上接第 132 页)

耗时情况和相应的加速比。从表 3 中可以看出,BEAPORSA 算法加密并行后,在双核平台上可得到 1.95 的平均加速比值。加密方通过并行处理之后很好地处理了解密方转移过来的计算负载,使 Batch RSA 算法的性能得到整体提升。

表 3 改进算法并行加密前和并行加密后的时间和加速比

算法类型	密钥长度(位/bit)					
	1792	2048	2304	2560	2816	3072
BEAPORSA	468	797	984	1309	1685	2157
BEAPPORSA	250	406	500	672	859	1082
加速比	1.87	1.96	1.97	1.95	1.96	1.99

结束语 本文提出的 Batch RSA 改进算法通过利用 Multi-power RSA 算法的优点,并且将解密时的一些计算量转移到加密端的方式来提升 Batch RSA 的解密性能。该算法可以在保证系统安全性的情况下使系统获得很好的加速比。且改进算法易于并行实现,能在多核平台上有效实现,可通过并行处理的方式使加密端的负载大幅度降低和解密端解密性能进一步提升。改进算法充分发挥了当前多核处理器设备的优势,使 Batch RSA 系统的整体性能得到进一步提升。下一步研究的重点是:如何在多核平台上更加有效地结合 OpenMP 和 OpenSSL 库来实现并行的密码系统以及将改进算法应用到安全套接层协议的握手过程中,进一步研究改进算法在该协议中的性能。

参 考 文 献

[1] Rivest R,Shamir A,Aldeman L. A Methoed for Obtaining DigitalSignatures and Public-key Cryptosystems [J]. Communications of the ACM,1978,21(2):120-126

[2] Boneh D,Shacham H. Fast Variants of RSA [R]. RSA Laboratories Cryptobytes,2002

[3] Takagi T. Fast RSA-type cryptosystem modulo pkq [C]// Krawczyk H,eds. CRYPTO,volume 1462 of Lecture Notes in Computer Science. 1998:318-326

[4] Takagi T. A fast RSA-type public-key primitive modulo pkq using Hensel lifting [J]. IEICE Transactions,2004,87(1):94-101

[5] 闵嗣鹤,严士健. 初等数论(第三版)[M]. 北京:高等教育出版社,2003

[6] Matsumoto T,Kato K. Speeding up secret computations with insecure auxiliary device [C]// Proc of the 8th Annual International Crypto Conference on Advances in Cryptology. London:Springer-Verlag,1988:497-506

[7] FIAT A. Batch RSA[C]// Proc of Crypto '89,LNCS435. Berlin:Springer-Verlag,1989:175-185

[8] Shacham H,Boneh D. Improving SSL Handshake Performance via Batching[C]// Proceedings of 2001'RSA. 2001:28-43

[9] Paxson V,Sommer R. An architecture for exploiting multi-core processors to parallelize network intrusion prevention [C]// Proceedings of the IEEE Sarnoff Symposium. 2007:1-7

[10] Timothy G, Beverly A. Patterns for Parallel Programming [M]. Boston,MA:Addison-Wesley,2005

[11] Castelluccia C,Mykletun E,Tsudik G. Improving secure server performance by re-balancing SSL/TLS handshakes [C]// Proc of the 2006 ACM Symposium on Information,Computer and Communications Security. New York:ACM,2006:26-34

[12] Li Yun-fei,Liu Qing,Li Tong. Design and Implementation of an Improved RSA Algorithm [C]// Proc of the 2010 e-Health Networking,Digital Ecosystems and Technologies. Shenzhen,2010:390-393

[13] 李云飞,柳青,郝林,等. 一种有效的 RSA 算法改进方案的研究[J]. 计算机应用,2010,30(9):2393-2397

[14] Cohen H. A Course in Computational Algebraic Number Theory [D]. vol 138 of Graduate Texts in Mathematics. Springer-Verlag,1996

[15] Vuillaume C. Efficiency comparison of several RSA variants [D]. Darmstadt University of Technology,2003

[16] Silverman R,Wagstaff Jr S. A Practical Analysis of the Elliptic Curve Factoring Algorithm [J]. Math. Comp,1993,61(203):445-462

[17] Okamoto E,Peralta R. Faster Factoring of Integers of a Special Form [J]. IEICE Transactionson Fundamentals of Electronics,Communications,and Computer Sciences,1996,79(4):489-493

[18] Boneh D,Durfee G,Howgrave-Graham N. Factoring N=prq for Large r [C]// Proceedings of Crypto '99. vol. 1666 of LNCS. Springer-Verlag,1999:326-337

[19] Lenstra A K, Verheul E R. Selecting cryptographic key sizes [J]. The Journal of the International Association for Cryptologic Research,2001,14(4):255-293

[20] 李云飞,柳青,李彤,等. 基于多核的批处理 RSA 的并行加速方法[J]. 云南大学学报:自然科学版,2011,33(1):22-26

[21] Liu Qing,Li Yun-fei,Li Tong,et al. The Research of the Batch RSA Decryption Performance[J]. Journal of Computational Information Systems,2011,7(3):948-955

[22] Chandra R,Menon R,Dagum L,et al. Parallel Programming in OpenMP [M]. New York:Morgan Kaufmann,2000

[23] Viega J,Messier M,Chandra P. Network Security with OpenSSL[M]. O'Reilly,2002