DOI:10.13718/j.cnki.xsxb.2018.10.016

基于身份的并行增量移动云存储方案①

柯 钢

东莞职业技术学院 计算机工程系,广东 东莞 523808

摘要:针对移动云计算环境中的数据存储问题,提出了一种基于身份的并行增量移动云存储方案.该方案引用了并行计算思想,使用密文聚合技术,充分挖掘了现有移动客户端的性能,在效率和能耗之间达到了平衡;使用增量代理重加密算法,在将部分计算迁移至云计算中心的同时,提高其整体性能;不使用传统公钥证书,既减少了密钥管理压力,也实现了信息的保密性和完整性.实验表明,该方案提高了 CPU 的使用率,具有较强的可用性.

关 键 词:移动云计算;云存储;并行加密;代理重加密

中图分类号:TP309.2 **文献标志码**:A **文章编号**:1000-5471(2018)10-0087-10

作为一种潜力巨大、前景广阔的计算模式,云计算[1]得到产业界、学术界的广泛关注.随着移动终端技术、无线网络的发展,云计算与之逐渐融合,形成了移动云计算.通常在移动云计算环境中,移动客户端安装有轻量的应用,通过无线网络(如 4G,3G 或 GPRS)与云计算服务商通信.这样资源受限的移动客户端,可利用云计算服务商提供的计算资源、存储资源,有效减少自身的能量消耗,通过备份等技术扩展了移动应用的服务功能,提高了其可用性和可靠性.然而,移动云计算也面临着诸多问题[2]:1)安全和隐私保护:用户的数据迁移至云计算服务商,脱离了用户的管控,有信息泄露风险;2)资源受限:通常移动客户端使用电池,无法长时间工作,与常规计算设备相比,存储空间和计算能力也不足;3)现有的密码算法无法直接适用于移动云计算环境,需要结合移动云计算特点,取得安全和效率的平衡.

针对移动云计算环境中的数据存储问题,许多学者展开了研究.Hsueh 等人[3]提出了一个安全移动云计算框架,但是该框架忽视了移动客户端的资源受限性,难以承受大量的密码学计算.在移动云计算环境下,传统的密码方案无法直接应用,我们必须提出适用于移动云计算环境的密码方案.在 2005 年,Ateniese 等人提出了代理重加密的概念,并运用于分布式安全存储中.从此,代理重加密方案引起了人们的极大关注.在代理重加密[4-5]中,代理节点(Proxy)不是完全可信的,只是按照约定将发送方的密文转化为接收方的密文,但是不能获取明文的任何信息.这样用户可以将繁重的数据访问操作迁移至云服务商.Tysowski[6]基于代理重加密的思想,提出了一种云存储方案,同时避免了可信第三方的问题.文献[7-9]提出了基于代理重加密的安全云存储密钥管理方案.Khan 等人[10]提出了一种基于增量代理重加密的移动云存储方案,在保证保密性和完整性的基础上,提高了云存储中心文件修改的效率.Son 等人[11]提出一个新的适用于移动云存储环境的条件代理重加密方案,实现了数据的组安全和隐私保护,同时还支持外包计

① 收稿日期:2017-11-04

基金项目:广东省教育厅 2017 年度青年创新人才类项目(自然科学,2017GKQNCX116),东莞职业技术学院示范建设专项资金资助项目(政 201721).

作者简介:柯 钢(1983-),男,硕士,讲师,主要从事网络安全技术、机器学习等方面的研究.

算. 但是这些移动云计算方案都是基于传统公钥的代理重加密方案，这些方案依赖一个可信中心，可信中心拥有每个用户的私钥，掌握用户的秘密信息，存在着密钥管理代价大的缺点. 针对这些问题，基于身份的密码体制被提出. 它将用户的公开信息作为公钥，比如个人的电子邮件、手机号码等. 由于其他用户知道对方的公开信息，这种方法不再需要从CA查找用户的公钥，省去了密钥管理的麻烦，但是方案一般效率不高.

因此，本文提出一种基于身份的并行增量移动云存储方案，保证了信息的保密性和完整性. 该方案引用了并行计算思想，使用密文聚合技术，充分发挥现有移动客户端的性能，在效率和能耗之间达到了平衡；使用增量代理重加密算法，在将繁重的计算迁移至云计算中心的同时，提高其整体性能；不使用传统公钥证书，减少了密钥管理压力.

1 并行加密

目前密码算法的并行化鲜有研究，大多数仍然沿用传统的串行思维，而没有意识到硬件的迅速发展带给密码算法的挑战和变化. 另外，目前的大多数并行密码方案[12-14]仅仅是将传统的串行方案并行化，并未在算法层面进一步优化. 仅有少数并行密码方案在算法层面优化，如Han等人[15]针对多用户组播通信环境，借助签密本身所具有的特性，提出了并行的多接收方签密算法的框架，并在此基础上提出了一个并行多接收方签密的方案. Zhang等人[16]提出了一个并行多接收方代理重加密方案，解决无线环境中的安全组播问题. 柯钢[17]提出了一种并行无证书加密方案，解决了云存储中的安全问题.

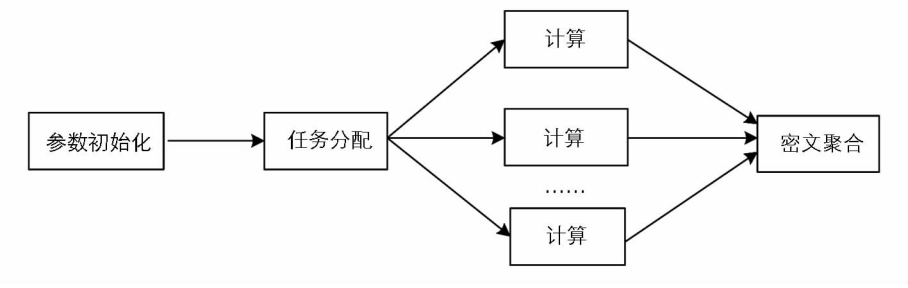

图1 并行计算结构图

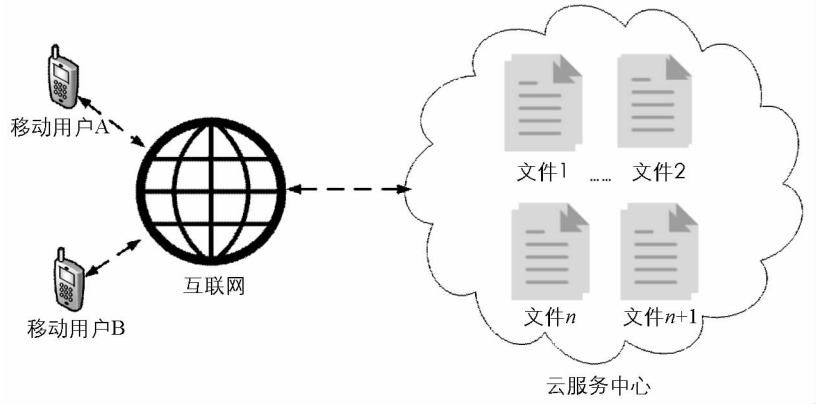

图2 网络模型图

本文提出的并行计算方案分为4个阶段：公共参数计算、任务分配、任务计算和消息整合，如图1所示. 在第一个阶段，计算方案所需的共有参数，为后续的并行计算准备复用参数；在第二个阶段，将繁重的计算任务分解，交给各个计算组件；在第三个阶段，每个计算组件独立地完成计算任务，彼此没有依赖关系；在最后阶段，将每个计算组件的输出消息进行有效整合，去除共有部分，形成最终的长度更短的有效消息.

2　并行增量移动云存储方案

2.1　网络模型

如图 2 所示，本文方案参与方由云计算中心和用户组成，其中用户根据收发数据不同，可分为数据拥有者和数据接收者. 数据拥有者(如移动用户 A)，将数据加密上传至云计算中心(又被称为代理节点)，并通过云计算中心将数据分享给数据接收者(如移动用户 B). 因云计算中心内的数据频繁被修改，数据拥有者增量加密原始明文数据，并产生代理重加密密钥. 云计算中心收到代理重加密密钥后，根据访问控制列表，进行重加密操作，产生重加密密文. 数据接收者在下载属于自己的密文后，使用自己的私钥就能解密，从而得到明文信息.

2.2　基于身份的并行增量代理重加密算法

该算法是一种基于身份的并行增量代理重加密算法，它有 6 个步骤组成，即系统参数生成、私钥生成、加密、代理重密钥生成、代理重加密、解密.

1) 系统参数生成：密钥产生中心选取 2 个 q 阶循环群 G_1，G_2，满足 e：$G_1 \times G_1 \rightarrow G_2$，其中 g 是 G_1 的生成元；选择 2 个 Hash 函数：$H_1:\{0,1\}^* \rightarrow G_1$ 和 $H_2: G_2 \rightarrow G_1$；选择随机数 $s \in Z_p^*$ 作为主密钥，那么共有参数为(G_1，H_1，H_2，g，g^s).

2) 私钥生成：假设用户身份为 $id \in \{0,1\}^*$，该算法输出私钥 $sk_{id} = H_1(id)^s$. 令用户 A 和 B 的身份标识为 id_A 和 id_B.

3) 加密：假设移动用户 A 要上传文件 F 至云服务中心中，为了保证保密性，做如下操作：

① 将 F 划分为 d 块，即 $F = \sum_{j=1}^{d} F_j$ 并且满足公式：

$$Length(F_j) = \begin{cases} \lfloor Length(F)/d \rfloor = t & 1 \leqslant j \leqslant (d-1) \\ Length(F) - t \cdot (d-1) & j = d \end{cases}$$

其中 F_j 代表第 j 个文件块，$Length(F_j)$ 代表第 j 个文件块的文件长度；

② 选取随机数 $r_i \in Z_p^*$；

③ 对于 d 个明文块，划分 N 个分组，分配到对应的 CPU 核心中，进行并行计算. 这里 N 代表着 CPU 的核数. 过程如下：

For $i = 0,\cdots,N-1$

　For $t = 0,\cdots,\lceil d/N \rceil - 1$：

　　计算 $C_A(m_{t+i*N}) = m_{t+i*N} \cdot e(g^s, H_1(id_A))_i^r$；

　　计算 $MAC_{t+i*N} = Hash(m_{t+i*N})$

　End For

End For

④ 计算 $MAC_{final} = Hash(\sum_{j=1}^{d} MAC_j)$，输出密文 $C(m) = \langle C_A(m_1),\cdots,C_A(m_d)\rangle$ 和消息块哈希值 $mac(A) = (MAC_1, \cdots, MAC_d)$；

⑤ 上传密文和哈希值$\langle g^r, C(m), mac(A), MAC_{\text{final}}, d\rangle$ 至云服务中心.

4) 代理重密钥生成：为了将解密权限代理给用户 B，用户 A 选择随机数 $x \in G_2$ 和 $r'_i \in Z_p^*$，计算 $R_1 = H_1(id_A)^{-s} \cdot H_2(x)$，$R_2 = (g^{r'_i}, x \cdot e(g^s, H_1(id_2))^{r'_i})$，返回代理重加密密钥为 $rk_{A\rightarrow B} = (R_1, R_2, sk_{id_A}^{-1} \cdot H_2(x))$.

5) 代理重加密：

如果用户B下载用户A上传的文件，在经过用户A的授权后，云服务中心得到代理重加密密钥 $rk_{A\to B}$. 这里假设云服务中心为半可信的第三方，它对存储信息不感兴趣，只是忠实地执行各种运算. 为了对密文 $C(m)=\langle C_A(m_1),\cdots,C_A(m_d)\rangle$ 进行代理重加密，代理节点使用 $rk_{A\to B}=(R_1,R_2,R_3)$ 进行并行计算：

For $i=0,\cdots,N-1$

For $t=0,\cdots,\lceil d/N\rceil-1$：

计算 $C_B(m_{t+i*N})=C_A(m_{t+i*N})\cdot e(g^{r_i},R_3)$；

End For

End For

并行计算结束后，输出 d 个密文 $\langle g^{r_i},C_B(m_j),R_1,R_2\rangle(j=1,\cdots,d)$，云服务中心聚合密文，得到最终的密文为 $\langle g^{r_i},C_B(m_1),\cdots,C_B(m_d),R_1,R_2\rangle$，接着将聚合的密文、$MAC_{\text{final}}$ 和 d 转发给用户B.

6) 解密：

在收到密文后，用户B执行如下计算：

① 由 (R_1,R_2)，计算 $R_2/e(R'_1,H_2(x))$ 得到 x；

② 执行并行计算：

For $i=0,\cdots,N-1$

For $t=0,\cdots,\lceil d/N\rceil-1$：

计算 $m_{t+i*N}=(C_2\cdot e(g^{r_i},R_3))/e(g^{r_i},H_2(x))$；

计算 $MAC'_{t+i*N}=Hash(m_{t+i*N})$

End For

End For

③ 聚合得到明文 $m=\langle m_1,\cdots,m_d\rangle$；

④ 计算 $MAC'_{\text{final}}=Hash(\sum_{j=1}^{d}MAC'_j)$.

这时，用户B比较 MAC_{final} 和 MAC'_{final}，如果相等，那么解密得到的明文具有完整性，否则已被篡改.

2.2.1 明文块插入

当数据拥有者用户A在位置 loc 上，将 d_{insert} 个新明文块 m_{insert} 插入到上传文件中. 将 $\langle g^r,C(m),mac(A),MAC_{\text{final}},d\rangle$ 下载至本地，用户A执行计算：

1) 选取随机数 r，加密 $C_A(m_{\text{insert}_i})=m_{\text{insert}_i}\cdot e(g^s,H_1(id_A))^r$，$1\le i\le d_{\text{insert}}$；

2) 更新消息认证码，计算：

$$MAC_{\text{insert}_i}=Hash(m_{\text{insert}_i}),\ 1\le i\le d_{\text{insert}}$$

$$MAC_{\text{final}}=Hash(\sum_{i=1}^{loc}MAC_i\ \|\ \sum_{j=1}^{d_{\text{insert}}}MAC_{\text{insert}_j}\ \|\ \sum_{k=loc+1}^{d}MAC_k)$$

3) 发送更新申请至云服务中心，并附带信息包括新明文块认证码 MAC_{insert}、整体消息认证码 $MAC_{\text{final}}(m)$、位置信息 loc 和新加密密文 $C_A(m_{\text{insert}})$.

待收到用户A发送的信息后，云服务中心将密文插入到相应的文件中，更新消息认证码，执行如下操作：

1) 更新消息认证码，计算 $MAC=\sum_{i=1}^{loc}MAC_i\ \|\ \sum_{j=1}^{d_{\text{insert}}}MAC_{\text{insert}_j}\ \|\ \sum_{k=loc+1}^{d}MAC_k$；

2) 更新密文，计算 $C=\sum_{i=1}^{loc}C_i\ \|\ \sum_{i=1}^{d_{\text{insert}}}C_A(m_{\text{insert}_i})\ \|\ \sum_{k=loc+1}^{d}C_k$；

3) 更新 $d_{\text{new}} = d + d_{\text{insert}}$.

2.2.2　明文块删除

当数据拥有者用户 A 在位置 loc 上将 d_{delete} 个明文块删除时，下载每个消息块的认证码 MAC_i，并更新整体消息认证码，计算 $MAC_{\text{final}} = Hash(\sum_{i=1}^{loc} MAC_i \parallel \sum_{k=loc+1+d_{\text{delete}}}^{d} MAC_k)$. 接着向云服务中心发送删除申请，并附带整体消息认证码 $MAC_{\text{final}}(m)$、位置信息 loc 和删除消息块数.

当收到相应的信息后，云服务中心删除相应的消息块，并作如下更新：

1) $MAC = Hash(\sum_{i=1}^{loc} MAC_i \parallel \sum_{k=loc+1+d_{\text{delete}}}^{d} MAC_k)$;

2) $C = \sum_{i=1}^{loc} C_i \parallel \sum_{k=loc+1+d_{\text{delete}}}^{d} C_k$;

3) $d = d - d_{\text{delete}}$.

2.2.3　明文块修改

当数据拥有者用户 A 在位置 loc 上将 d_{modify} 个明文块更新时，将$\langle g^{r_i}, C(m)\rangle$ 和每个消息块的认证码 MAC_i 下载至本地，用户 A 执行计算：

1) 加密 $C_A(m_{\text{modify}_i}) = m_{\text{modify}_i} \cdot e(g^s, H_1(id_A))^r$, $1 \leqslant i \leqslant d_{\text{modify}}$;

2) 更新 $MAC_{\text{modify}_i} = Hash(m_{\text{modify}_i})$, $1 \leqslant i \leqslant d_{\text{modify}}$;

3) 计算 $MAC_{\text{final}} = \sum_{i=1}^{loc} MAC_i \parallel \sum_{j=1}^{d_{\text{modify}}} MAC_{\text{modify}_j} \parallel \sum_{k=loc+1}^{d} MAC_k$.

用户 A 向云服务中心发送更新申请，并附带更新信息块的认证码 MAC_{modify}、更新密文 $C_A(m_{\text{modify}_i})$、整体消息认证码 $MAC_{\text{final}}(m)$、位置信息 loc 和更新消息块数 d_{modify}.

同样，收到这些信息后，云服务中心更新相应的消息块：

1) 更新消息认证码 $MAC = \sum_{i=1}^{loc} MAC_i \parallel \sum_{j=1}^{d_{\text{modify}}} MAC_{\text{modify}_j} \parallel \sum_{k=loc+1}^{d} MAC_k$;

2) 更新密文 $C = \sum_{i=1}^{loc} C_i \parallel \sum_{j=1}^{d_{\text{modify}}} C_A(m_{\text{modify}_j}) \parallel \sum_{k=loc+1}^{d} C_k$.

3　实验与分析

为了测试方案效率，本文建立了移动云计算环境，其中云客户端程序运行在小米 note 1s Android 手机上，详细参数如表 1 所示. 为了加解密明文信息，采用 JPBC (Pairing Based Cryptography Library)密码函数库，进行 Android 编程，分别实现了算法 IPRE、串行基于身份的增量代理重加密方案 I-IPRE 和并行基于身份的增量代理重加密方案 PI-IPRE. 双线性对参数选择 Type A 类型，即素数域 F_q 上的椭圆曲线 $y^2 = x^3 + x$，其中 q 是 512 位. 每次加密的明文信息最大不超过 64 个字节. 同时为了确保上传文件的完整性，选择 SHA256 作为哈希算法，选择安卓 Thread Pool Executor 线程池进行多线程并行计算[19]，因为安卓构建于 Linux Kernel 上，它自动将多线程运算分配到各个核心上，执行多核并行计算.

表 1　移动客户端参数

项　目	参　数
CPU	高通 骁龙 801(MSM8974)2.5 GHz
RAM	3 GB
ROM	16 GB
操作系统	Android 4.4
电池容量	3000mAh

选用 Emmagee 作为移动端的性能测试工具，它能实时监测每个 app 运行时的 CPU 使用率、所占内存、流量、消耗电量等相关信息，并能生成统计表.

3.1 加解密效率实验

在本实验中，主要评估方案对数据的加密和解密效率，分别加密和解密数据集中的5种类型数据，大小分别为51 200 bit、102 400 bit、153 600 bit、204 800 bit和256 000 bit的文件. 围绕加解密时间、能量消耗、CPU使用率和内存占用情况进行比较.

1) 加解密时间

加解密实验结果如图3所示，X轴代表明文的大小(单位：bit)，Y轴代表在移动客户端的运行时间(单位：s). 从图3可以看出，I-IPRE方案在加密时，因使用了基于身份的加密体制，其复杂度比IPRE要高；而本方案PI-IPRE采用了并行加密的思想，在4核CPU上进行优化，所使用的时间最少，有效地提高了计算效率.

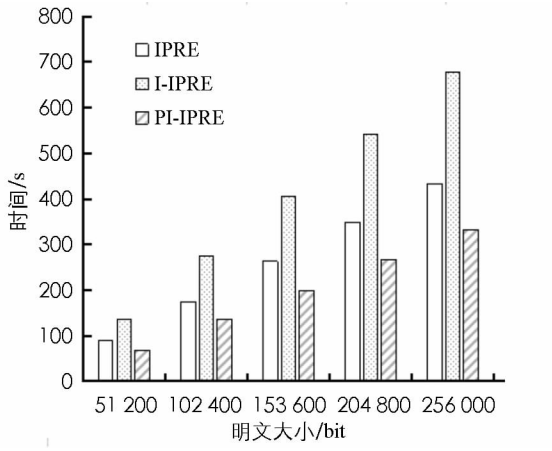

图3 明文加密消耗时间

为了表示方案b相对于方案a的效率提升，我们定义提升效率公式为：$\eta=(T_b-T_a)/T_a$，其中T_a代表a方案的运算时间，T_b代表b方案的运算时间. 该实验用η_1和η_2表示PI-IPRE相对于IPRE和I-IPRE的提升效率，如表2所示. I-IPRE采用了基于身份的代理重加密算法，与文献[10]的代理重加密算法相比，以公开信息作为用户的公钥，降低了密钥管理的复杂度，但存在效率低的缺点. 本文提出的PI-IPRE方案，与IPRE方案相比较，方案提升效率稳定在24%左右.

表2 PI-IPRE方案提升效率对比

占用内存/bit	提升效率 η/%	
	η_1	η_2
51 200	25.10	51.02
102 400	22.65	50.69
153 600	24.12	50.80
204 800	23.54	50.75
256 000	23.28	50.90

同时，根据Amdahl法则[20]，如果程序在N个处理器或核心上运行，其全部加速效果可以这样计算：加速比$Speedup=1/(s+p/N)$. 其中，s表示程序串行部分的时间总和，$p(p=1-s)$表示程序的并行运算部分的运行时间总和. 在本实验中，如果串行部分时间可以被忽略，那么简单并行的理论加速是4，理论提升效率$\eta=75\%$. 详细结果如表2所示，虽然Android手机CPU并行的优化效果无法达到理论的提升效率，但PI-IPRE方案与串行I-IPRE方案相比也有明显提高，基本稳定在51%左右.

2) 能量消耗

如图4所示，电量消耗图X轴代表明文的大小，Y轴代表算法在运行期间消耗的电量. 从图可以看出，本文所提方案PI-IPRE在电量消耗方面介于IPRE和I-IPRE两者之间.

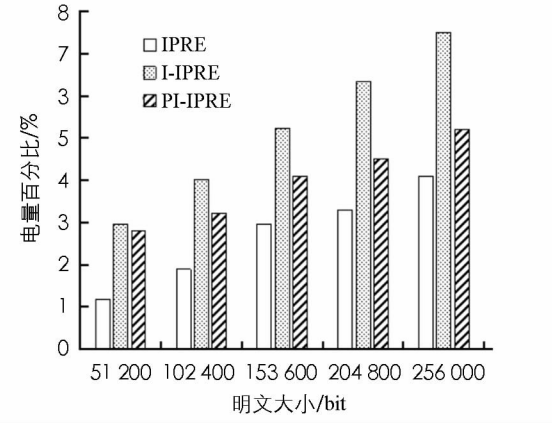

图4 电量消耗图

3) CPU使用率

该部分实验测试串行算法和并行算法对CPU使用率的影响，分别使用I-IPRE和PI-IPRE方案对51 200 bit的

文件进行算法初始化、产生密钥、加密、解密等操作，测试整个过程中 CPU 使用率的变化情况，实验结果如图 5 和图 6 所示.

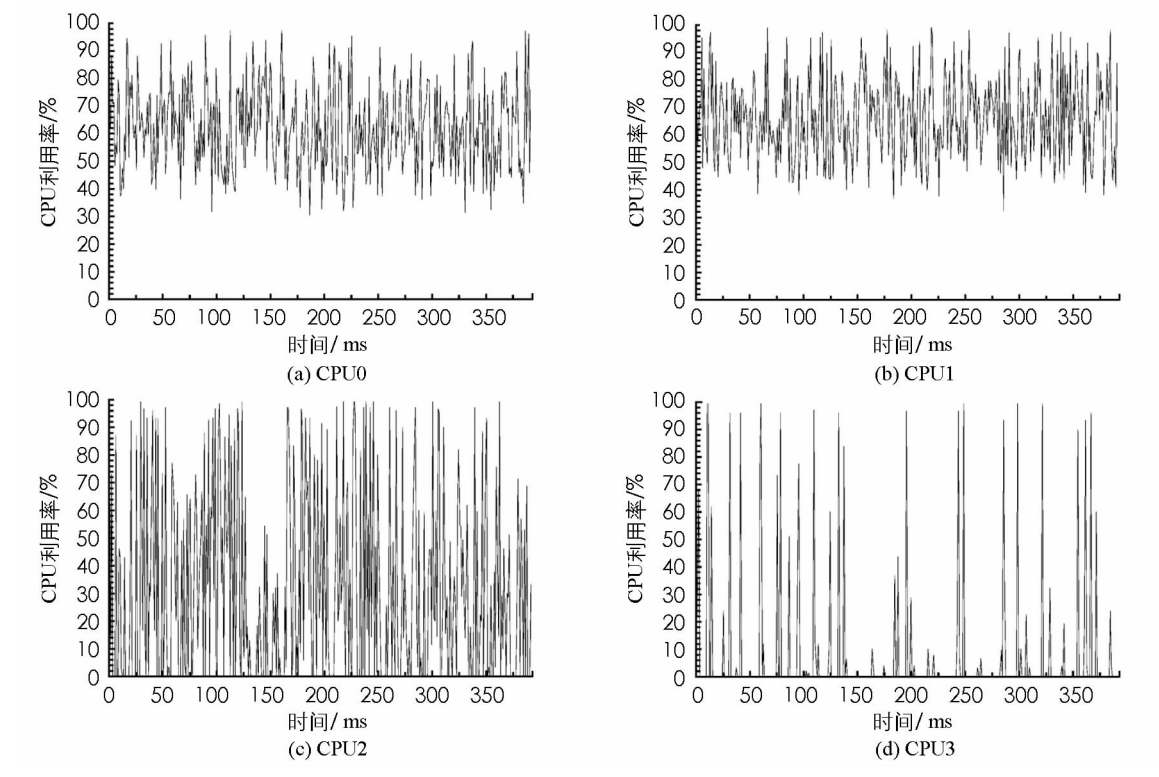

图 5　I-IPRE 方案串行加密 51 200 bit 文件时的 CPU 利用率

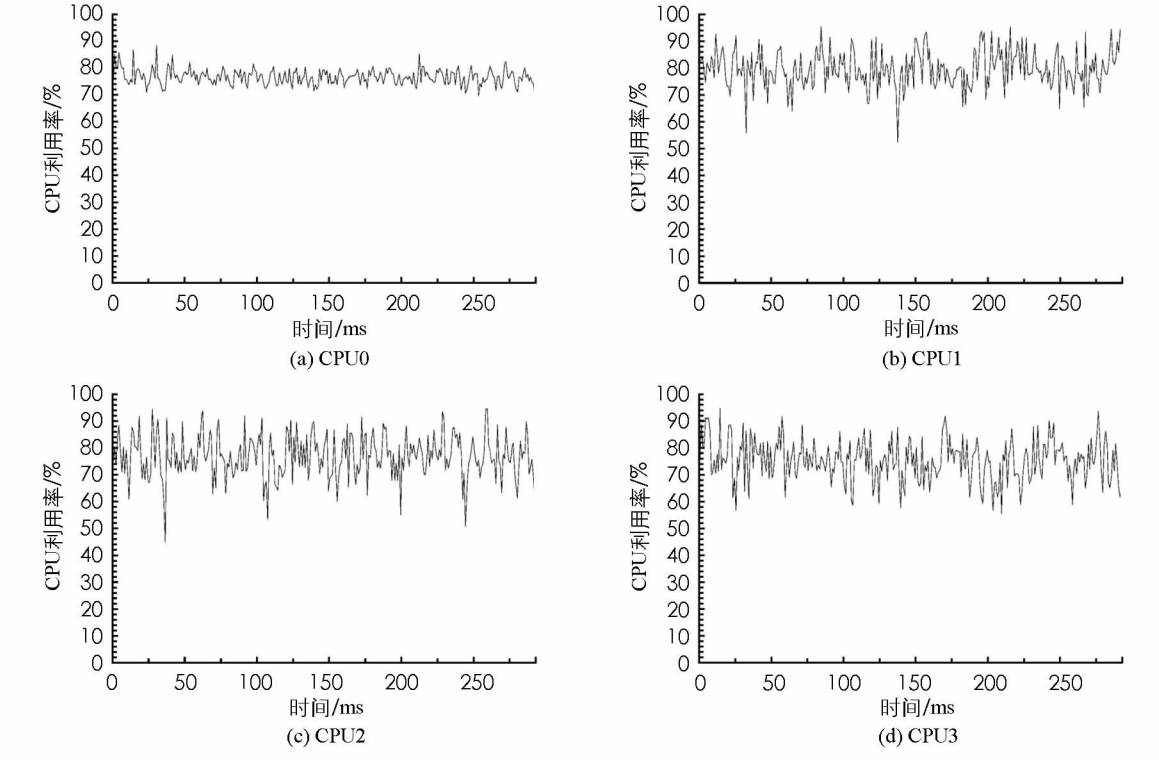

图 6　PI-IPRE 方案并行加密 51 200 bit 文件时的 CPU 利用率

从图5可以看出，I-IPRE运行时，CPU的核心CPU0和CPU1的使用率保持在40%到90%之间，而到CPU2和CPU3，CPU使用率逐渐减少，甚至CPU3大部分时间其使用率为0. 从图6可以看出，并行PI-IPRE算法运行时，CPU的4个核心均保持了60%～90%的CPU使用率，充分利用了硬件资源，降低了算法的运行时间.

4) 内存占用情况

该部分实验测试各方案在加密文件时，所消耗的内存. 实验结果如表3所示，各方案占用内存基本保持在20～21 MB.

表3 各方案占用内存PSS(MB)

占用内存/bit	IPRE方案	I-IPRE方案	PI-IPRE方案
51 200	20.36	20.19	20.80
102 400	20.65	20.57	20.81
153 600	21.14	20.56	20.64
204 800	20.21	20.38	20.35
256 000	20.53	20.19	20.96

3.2 明文块修改实验

用户在移动云计算中，经常要修改存储于云服务器内的文件，频繁进行明文块插入、删除、修改等操作. 由于明文块的插入和修改操作相同，它们所消耗的时间也一样. 本实验采用PI-IPRE方案只对文件修改操作进行验证，要修改的明文大小为204 800 bit，分别划分8,16,32块，那么每块明文大小为25 600 bit,12 800 bit,6 400 bit.

修改每块所消耗的时间如图7所示，X轴代表明文块的大小(单位：bit)，Y轴代表在移动客户端的运行时间(单位：s). 由于本实验多了SHA256等完整性验证操作，所消耗时间有所增加.

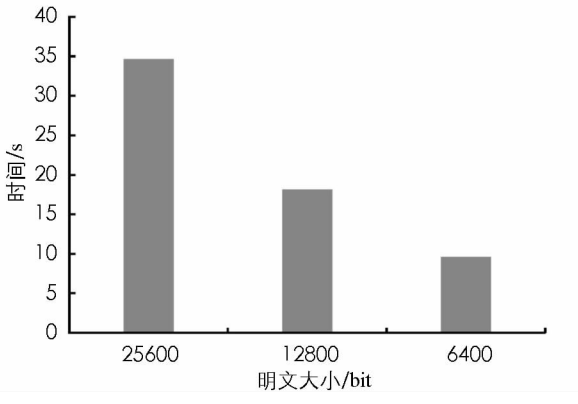

图7 明文块修改消耗时间图

4 安全性分析

根据Buonanno等人的方案[21]定义，增量加密方案安全性指的是：如果攻击者A在时间t内，执行了q_e轮的加密预言机询问和q_{inc}轮的增量操作预言机询问，而它获胜的概率小于可忽略的概率ε，那么方案$(t,q_e,\mu,q_{inc},\iota;\varepsilon)$是不可区分性安全的，其中$\iota$是分块数，$\mu$是增量操作的块数.

该增量加密方案采用的加密方案为Matthew等人[18]提出的身份类加密算法，它的安全性可规约为Bilinear Diffie Hellman困难性问题，且在随机预言机下满足选择明文攻击的不可区分性. 对于给定的消息块$\sigma_1,\sigma_2,\cdots,\sigma_n$，选择随机数$r_i$，密文为$\langle g^{r_i},\sigma_i \cdot e(g^s,H_1(id_A))^{r_i}\rangle$. 攻击者如果以不可忽略的概率攻破该方案，那么只能以$o(\iota^2\mu^2/2^{2b})$概率攻破文献[18]的方案. 当攻击者做增量操作时，如替换或插入，每次新的随机数r_i被选择，攻击者这时候攻破文献[18]方案的概率为$o(\mu/2^b)$. 这时$\varepsilon = o(\iota^2\mu^2/2^{2b}+\mu/2^b)$；该增量方案在随机预言机下，具备不可区分性.

5 结 论

针对移动云计算环境中的数据迁移和存储问题，提出了一种基于身份的并行增量移动云存储方案. 为了在效率和能耗之间达到平衡，该方案引入了并行计算思想，充分挖掘了现有移动客户端的性能，在效率

和能耗之间达到平衡；使用基于身份的增量代理重加密算法，在将繁重的计算迁移至云计算中心的同时，也减少了密钥管理的压力. 最后，通过实验证明了本方案的有效性.

参考文献：

[1] MEHTA M, AJMERA I, JONDHALE. Mobile Cloud Computing [J]. International Journal of Electronics and Communication Engineering & Technology , 2013, 4(5): 152－160.

[2] FERNANDO N, LOKE S W, RAHAYU W. Mobile Cloud Computing: A Survey [J]. Future Generation Computer Systems, 2013, 29(1): 84－106.

[3] HSUEH S C, LIN J Y, LIN M Y. Secure Cloud Storage for Convenient Data Archive of Smart Phones [C]// 2011 IEEE 15th International Conference on Consumer Electronics. Singapore: IEEE, 2011: 156－161.

[4] AI-RIYAMI S S, PATERSON K G. Certificateless Public Key Cryptography [C]// International Conference on the Theory and Application of Cryptology and Information Security. Advances in Cryptology-ASIACRYPT 2003. Taipei, Taiwan: Springer, 2003: 452－473.

[5] BLAZE M, BLEUMER G, STRAUSS M. Divertible Protocols and Atomic Proxy Cryptography [C]// International Conference on the Theory and Application of Crytographic Techniques: Advances in Cryptology-EUROCRYPT'98. Espoo, Finland: Springer, 1998: 127－144.

[6] TYSOWSKI P K, HASAN M A. Re-Encryption-Based Key Management Towards Secure and Scalable Mobile Applications in Clouds [J]. IACR Cryptology ePrint Archive, 2011, 668: 1－10.

[7] ITANI W, KAYSSI A, CHEHAB A. Energy-Efficient Incremental Integrity for Securing Storage in Mobile Cloud Computing [C]// 2010 International Conference on Energy Aware Computing (ICEAC 2010), Cairo, Egypt: IEEE, 2010: 1－2.

[8] JIA W, ZHU H, CAO Z, et al. SDSM: A Secure Data Service Mechanism in Mobile Cloud Computing [C]// 2011 IEEE Conference on Computer Communications Workshops (INFOCOM WKSHPS), Shanghai, China: IEEE, 2011: 1060－1065.

[9] YANG J, WANG H, WANG J, et al. Provable Data Possession of Resource-Constrained Mobile Devices in Cloud Computing [J]. Journal of Networks, 2011, 6(7): 1033－1040.

[10] KHAN A N, KIAH M L M, MADANI S A, et al. Incremental Proxy Re-Encryption Scheme for Mobile Cloud Computing Environment [J]. Journal of Supercomputing, 2014, 68(2): 624－651.

[11] SON J, KIM D, BHUIYAN M Z A, et al. A New Outsourcing Conditional Proxy Re-Encryption Suitable for Mobile Cloud Environment [J]. Concurrency & Computation Practice & Experience, 2017, 29(14): e3946

[12] NISHIKAWA N, AMANO H, IWAI K. Implementation of Bitsliced AES Encryption on CUDA-Enabled GPU [C]// International Conference on Network and System Security: NSS 2017. Helsinki, Finland: Springer, 2017: 273－287.

[13] LIU B, BASS B M. Parallel AES Encryption Engines for Many-Core Processor Arrays [J]. IEEE Transactions on Computers, 2013, 62(3): 536－547.

[14] HUANG R, RHEE K H, UCHIDA S. A Parallel Image Encryption Method Based on Compressive Sensing [J]. Multimedia Tools and Applications, 2014, 72(1): 71－93.

[15] HAN Y, GUI X, WU X. Parallel Multi-Recipient Signcryption for Imbalanced Wireless Networks [J]. International Journal of Innovative Computing Information & Control, 2010, 6(8): 3621－3630.

[16] ZHANG M, WU X, HAN Y. Parallel Multi-Recipient Proxy Re-Encryption for Secure Group Communication in Wireless Networks [J]. ICIC Express Letters, 2013, 7(8): 2203－2209.

[17] 柯　钢. 适用于云存储的并行无证书代理重加密方案 [J]. 西南师范大学学报(自然科学版), 2016, 41(7): 61－67.

[18] GREEN M, ATENIESE G. Identity-Based Proxy Re-Encryption [C]// International Conference on Applied Cryptography and Network Security. Zhuhai, China: ACNS, 2007: 288－306.

[19] GOETZ B, PEIERLS T, BLOCH J, et al. Java Concurrency in Practice [M]. Boston: Addison-Wesley, 2006.

[20] KRISHNAPRASAD S. Uses and Abuses of Amdahl's Law [J]. Journal of Computing Sciences in Colleges, 2001, 17(2): 288－293.

[21] BUONANNO E, KATZ J, YUNG M. Incremental Unforgeable Encryption [J]. Lecture Notes in Computer Science, 2002: 109－124.

Parallel Incremental Mobile Cloud Data Storage Scheme on Identity-Based Cryptography

KE Gang

Department of Computer Engineering, Dongguan Polytechnic, Dongguan Guangdong 523808, China

Abstract: Aiming at data storage issue in mobile cloud computing environment, a parallel incremental mobile cloud storage scheme based on identity-based proxy re-encryption has been proposed. Parallel computing is brought into the scheme, which aggregates ciphertexts, and the existing mobile client performance is full excavated for the balance between efficiency and energy consumption. Incremental proxy re-encryption is used to migrate the part computing to the cloud computing center, improving the overall performance at the same time. Certificates in traditional public key cryptosystem are not need anymore, which not only reduce the pressure of key management, and also ensure the confidentiality and integrity of the information. Experimental results show that the scheme improves the utilization of CPU and has considerable availability.

Key words: mobile cloud computing; cloud storage; parallel encryption; proxy re-encryption

责任编辑 崔玉洁