软件学报 ISSN 1000-9825, CODEN RUXUEW
Journal of Software,2015,26(5):1181−1195 [doi: 10.13328/j.cnki.jos.004656]

E-mail: jos@iscas.ac.cn
http://www.jos.org.cn
Tel: +86-10-62562563

云计算环境中支持关系运算的加密算法*

黄汝维[1,2], 桂小林[1], 陈宁江[2], 姚 婧[1]

[1](西安交通大学 电子与信息工程学院,陕西 西安 710049)

[2](广西大学 计算机与电子信息学院,广西 南宁 530004)

通讯作者: 黄汝维, E-mail: ruweih@126.com

摘 要: 随着云计算的深入发展,隐私安全成为云安全的一个关键问题.加密是一种常用的保护敏感数据的方法,但是它不支持有效的数据操作.为了提供云计算环境中的隐私保护,设计了一种随机数据结构——随机树,并构建了基于随机树的保序加密算法OPEART(order-preserving encryption based on random tree).OPEART通过引入随机性实现了对数据的加密,并支持加密数据的任何关系运算(>,<,>=等).安全分析和性能评估表明:OPEART 是 IND-DNCPA(indistinguishability under distinct and neighbouring chosen plaintext attack)安全的,并能高效地实现对加密数据的关系运算.

关键词: 云计算;隐私安全;保序加密;关系运算

中图法分类号: TP309

中文引用格式: 黄汝维,桂小林,陈宁江,姚婧.云计算环境中支持关系运算的加密算法.软件学报,2015,26(5):1181−1195. http://www.jos.org.cn/1000-9825/4656.htm

英文引用格式: Huang RW, Gui XL, Chen NJ, Yao J. Encryption algorithm supporting relational calculations in cloud computing. Ruan Jian Xue Bao/Journal of Software, 2015,26(5):1181−1195 (in Chinese). http://www.jos.org.cn/1000-9825/4656.htm

Encryption Algorithm Supporting Relational Calculations in Cloud Computing

HUANG Ru-Wei[1,2], GUI Xiao-Lin[1], CHEN Ning-Jiang[2], YAO Jing[1]

[1](School of Electronic and Information Engineering, Xi'an Jiaotong University, Xi'an 710049, China)

[2](School of Computer, Electronics and Information, Guangxi University, Nanning 530004, China)

Abstract: With the development of cloud computing, privacy has become the key problem of cloud security. While encryption is a well-established technology for protecting sensitive data, it makes effective data utilization a very challenging task. To solve the problem, this paper designs a randomized data structure—random tree (RT), and constructs an encryption scheme OPEART (order-preserving encryption algorithm based on RT). OPEART realizes the encryption of data by randomness, and supports relational calculations (>, <, >=, etc.) on encrypted data. Security analysis and performance evaluation show that OPEART is IND-DNCPA while achieving the goal of relational calculations on encrypted cloud data efficiently.

Key words: cloud computing; privacy security; order-preserving encryption; relational calculation

云计算作为一种新型的网络计算模式,以一种比传统 IT 更经济的方式向用户提供按需的 IT 服务(计算、存储和应用等).由于云计算的发展理念符合当前低碳经济与绿色计算的总体趋势,它得到了世界各国政府和企业的大力倡导与推动,正带来计算领域和商业领域的巨大变革.

但在已经实现的云计算服务中,隐私安全问题一直令人担忧,并已经成为阻碍云计算发展和推广的主要因

* 基金项目: 国家自然科学基金(61063012, 61363003); 教育部高等学校博士学科点专项科研基金(20120201110013); 广西自然科学基金(2013GXNSFBA019281, 2012GXNSFAA053222); 广西科学研究与技术开发计划(桂科攻 1348020-7); 广西教育厅科研基金(2013YB007); 广西大学科研基金(XBZ120257); 陕西省科技攻关项目(2012K06-30)

收稿时间: 2013-09-23; 修改时间: 2014-01-21; 定稿时间: 2014-05-21

素之一.用户的隐私数据包括可用来识别或定位个人的信息(例如电话号码、地址和信用卡号等)、敏感的信息(例如个人的健康状况、财务信息、公司的重要文件等).云计算的隐私安全问题源于云计算的数据外包和服务租赁的特点.用户数据存储到云环境中,人们失去了对数据的直接控制力,可能会导致个人隐私数据的泄露和滥用.而近年来发生的 Google,MediaMax 和 Salesforce.com 等云服务商泄露或丢失用户数据的事实证实了人们的担心[1].加密是一种常用的保护用户隐私数据的方法,但目前的大多数加密方案都不支持对密文的运算,如对加密的数据进行检索、对加密的公司财务信息进行排序等统计分析,因而严重妨碍了云服务商为用户提供更进一步的数据管理和运算服务,从而削弱了云计算的优势.针对上述问题,本文提出一种基于随机树的保序加密算法 OPEART(order-preserving encryption algorithm based on random tree).OPEART 通过树状结构组织数据,同时,通过随机地为树中节点分配大小不同的区域实现对数据的加密.该算法支持对加密数据的任何关系运算(==,>,<,<=,>=等),特别适用于云计算环境中的大型数据库,方便建立索引和快速检索.

本文第 1 节回顾相关研究的进展情况.第 2 节建立支持隐私保护的网络计算模型和攻击模型.第 3 节定义 OPEART 并介绍其具体实现.第 4 节和第 5 节分别就 OPEART 的安全性和性能进行评估.第 6 节做出总结.

1 相关工作

对于加密数据的关系运算问题,目前已有一些研究成果,可以大致分为以下几类:(1) 支持精确匹配的加密算法;(2) 保序的最小完美哈希函数;(3) 保序的加密算法;(4) 不保序的加密算法;(5) 借助索引的方法.

(1) 支持精确匹配的加密算法

Song 等人[2]提出了基于对称加密和数据异或运算的加密关键字检索算法;Boneh 等人[3]提出了基于双线性映射的加密关键字检索算法 PEKS;Ohtaki 等人[4]使用 BloomFilter 来存储关键字的各种布尔组合信息,从而实现了支持逻辑运算的密文检索;Liu 等人[5]提出了基于双线性映射的加密关键字检索算法 EPPKS,它是在 PEKS 的基础上增加了对外包数据(而不仅仅是外包数据的关键字)的加密部分,并让服务提供者参与了一部分解密工作,减轻了用户的计算负载.以上算法都只支持关系运算中的“==”比较,功能较为单一,不能完全解决云计算环境中对密文数据的关系运算问题.

(2) 保序的最小完美哈希函数

假设一个集合 S 有 m 个关键字,一个有 n 个桶的哈希函数 h 是保序的最小完美哈希函数需满足:

i) $h:S\rightarrow n$ 是单射函数;

ii) $m=n$;

iii) 如果 $x<y$,那么 $h(x)<h(y)$.

Belazzougui 等人[6]通过建立相关分级树和前缀匹配的方法实现了一个单调变化的最小完美哈希函数,该函数没有隐藏原数据的值,而是把该数据映射到与该值相近的桶中.Czech 等人[7,8]通过以两个任意函数的函数值作为顶点构造带权无环图实现了一个保序的最小完美哈希函数,该函数能够很好地隐藏原数据的值,但是要构造一个无环图可能需要多次尝试.以上的哈希函数都是针对静态数据域的,计算复杂度随着数据域中元素个数的增加而增长,当数据域很大或者动态变化时,保序的最小完美哈希函数的方法是不适用的.另外,要用哈希函数来实现对数据的加解密运算,需要保存一份映射表,这无疑增加了数据拥有者的存储负载.

(3) 保序的加密算法

Agrawal 等人[9]基于桶划分和分布概率映射的思想提出了一种保序对称加密算法 OPES,支持对加密数值数据的各种关系运算,其不足是:当定义域较大时,桶划分的计算负载较大;而且面对已知明文的攻击时,当输入分布的桶与对应的输出分布的桶中点的个数足够多时,可以通过解方程的方式破解.Boldyreva 等人[10,11]提出了一种基于折半查找和超几何概率分布的保序对称加密算法 OPSE,支持对加密数据的各种关系运算,但由于在计算超几何概率时需要进行多次组合运算,其计算负载较大.以上两种保序的加密方法的优点是保序性,即,密文反映了明文的大小关系,从而方便服务提供者对密文数据建立索引,加快查找速度,并易于实现各种关系运算;不足是,它们都不能同时满足高效性和安全性的要求.

(4) 不保序的加密算法

Wong 等人[12]设计了一个基于向量标量积的对称加密方案,该方案支持对加密数据库进行 KNN(*k*-nearest neighbor)计算,具有 IND-CCA 安全性,其不足是:

i) 只能对对象之间的距离进行比较,并不支持两对象本身的比较;

ii) 由于该算法是 IND-CCA 安全的,数据加密后具有不确定性和不保序性,因此要找到满足条件的数据,必须搜索整个数据库,这对于大型数据库来说是不现实的.

Boneh 等人[13,14]基于双线性映射理论实现了支持关系运算检索的加密方案,它们的优点是具有 IND-CCA 安全性,缺点是:

i) 计算复杂度高;

ii) 数据加密后具有不确定性和不保序性,服务提供者必须对表中的所有数据进行遍历才能找到符合条件的数据,效率很低.

Shi 等人[15]提出了一个 Interval tree 的概念,并基于双线性映射原理实现了支持关系运算检索的加密方案.该方案的优点是 IND-CCA 安全的,缺点是:

i) Interval tree 只能定义在静态数据域上;

ii) 计算复杂度高;

iii) SP 必须对表中的所有数据进行遍历才能找到符合条件的数据.

通过以上分析可知,不保序的加密算法的优点是安全性高(IND-CCA),缺点是:

i) 计算负载大;

ii) 要对整个表进行遍历才能找到满足要求的所有数据,这对于大型数据库来说是不现实的.

(5) 借助索引的方法

Wang 等人[16]针对 XML 数据库提出了一个安全的加密方案:

i) 采用任意一种确定的对称加密算法加密外包数据;

ii) 为了快速定位,构建可用于结构检索的结构索引和可用于数值比较的值索引.

该方案的不足是:

i) 需要服务提供者开辟额外的存储空间存放索引,且当数据发生变化时,数据拥有者要重新建立索引;

ii) 在建立值索引时采用了保序的加密算法,因此具有这类算法的不足.

Hacıgümüş 等人[17]将关系数据库中表的属性分为 5 类,并采用不同的加密方式:

i) 对于需要进行聚合型运算(SUM,AVG)的属性,采用隐私同态方案[18]加密;

ii) 对于需要进行精确匹配的属性,采用确定的对称加密算法来加密;

iii) 对于需要进行关系运算的属性,建立一个粗粒度的索引,将整个数据域分为若干段,每一段给一个编号,用编号来代替实际的属性值;

iv) 对于不会被单独访问的属性,对整个对象进行加密.

该方案中采用的粗粒度索引不能真正实现关系运算.

Wang 等人[19]通过建立索引实现了查找与关键字最相关(例如出现频率)的 k 个文件的功能,其索引主要包括用哈希函数加密的关键字和用任意一种对称加密算法加密的,包含该关键的文件编号和相关度,其中,为了隐藏真实的相关度,先用改进的 OPSE 对相关度进行了加密.该方法有以下不足:

i) 建立索引时需要对包含某个关键字的文件进行扫描,从而得到与该关键字相关的信息,例如出现频率,这增加了用户的计算负担;

ii) 索引占用了服务提供者额外的存储空间;

iii) 当增加或删除与某个关键字相关的文件时,用户必须重新设置并加密索引,这是非常麻烦的事.

以上方法都是借助索引来实现支持关系运算的检索,需要数据拥有者自己建立并维护索引,这增加了数据拥有者的负担,同时也占有了服务提供者额外的存储空间.

针对云计算环境中的大型数据库系统,我们权衡了以上加密算法的利弊,就已有的保序加密算法不能同时满足高效性和安全性的要求,设计了一种随机数据结构——随机树,并构建了基于随机树的保序加密算法OPEART(order-preserving encryption based on random tree),能够满足:

(1) 具有保序性,从而支持大规模数据库的快速检索,并支持各种关系运算;

(2) 安全性介于已有的保序加密算法和不保序的加密算法之间,即 IND-DNCPA 安全性;

(3) 不需要事先建立额外的索引,减轻 Owner 的计算和存储负担;

(4) 负载不会随着数据域的变化而变化.

2 问题描述

2.1 支持隐私保护的云计算模型

如图 1 所示,支持隐私保护的云计算模型反映了数据拥有者(owner)、用户(user)和服务提供者(service provider,简称 SP)之间的交互,具体过程如下:

(1) Owner 用加密算法 E 对敏感数据 $d_i(i\in[1,n],n\geqslant 1)$加密,得到 $E(d_i)$,然后存储到 SP 的服务器上;

(2) User 获得 Owner 的授权后,对敏感计算参数(*para*)加密,得到 $E(para)$,并将 $E(para)$和计算要求(*type*)提交给 SP;

(3) SP 验证 User 的权限,然后根据 User 的计算要求,对其权限范围的 $E(d_i)$和计算参数 $E(para)$进行计算,得到计算结果 $E(result)$,并将 $E(result)$返回给 User;

(4) User 对 $E(result)$进行解密,得到结果的明文 *result*.

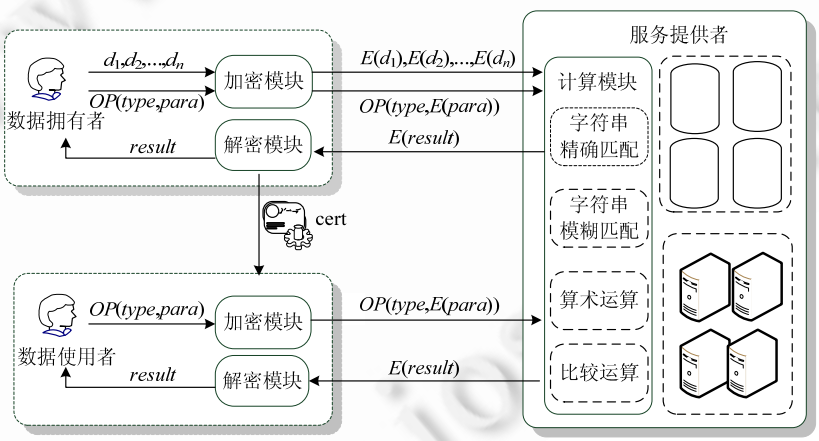


Fig.1 Privacy-Preserving calculation model of cloud

图 1 支持隐私保护的云计算模型

在这个过程中,由于 Owner 和 User 分别对敏感的外包数据和计算参数进行了加密处理,使得 Owner 和 User 的隐私得到了很好的保护.但同时也带来了一个新的问题:SP 如何对加密的数据进行计算呢?如果这个问题不能得到有效的解决,Owner 和 User 就不能利用云计算中的计算资源对敏感数据进行处理,从而削弱了云计算的优势.因此,本文提出的支持关系运算的加密算法是解决这一矛盾的关键技术之一.

2.2 威胁模型

根据图 1 可知,在用户交换的过程中可能存在以下几种隐私攻击:

(1) 数据从 Owner 传递到 SP 的过程中,外部攻击者可以通过窃听等方式盗取数据;

(2) 外部攻击者可以通过无授权的访问、木马和钓鱼软件来攻击 SP 从而获取数据;

(3) Owner 和 User 将数据提交给 SP 进行存储和运算,SP(内部攻击者)能够获得数据,因此,SP 要发起攻击(例如盗取、泄露数据、私自分析和统计数据等)更为容易.

在以上的 3 种攻击中,第(1)种和第(2)种是传统网络安全中涉及的问题;第(3)种是在云计算模式下出现的新的攻击形式,也是破坏性最大的一种隐私威胁.因此,我们设计的加密算法主要针对第 3 类攻击,同时能够有效地防御前两种攻击.

3 基于随机树的保序的加密算法 OPEART(order-preserving encryption algorithm based on random tree)

3.1 OPEART定义

本节将构造一种基于随机树的保序加密算法 OPEART.OPEART 在确保 Owner 和 User 数据安全的前提下,支持对加密数据进行任何关系运算.

定义 1(保序的加密算法). 假设加密算法 E 的定义域为 D,E 是保序的加密算法,满足:如果 $x_1>x_2(x_1,x_2\in D)$,那么 $E(x_1)>E(x_2)$.

定义 2(随机树). 随机树(random tree,简称 RT)是一棵多叉树,树中每个节点都对应定义域中的一点,并占据值域中的一段区间;树中的子节点区间是对父节点区间的一种随机划分,如图 2 所示.RT 的定义域为 $D=[0,4\times10^{12}]$,值域为 V 且 V 的大小$|V|\in[10^{16},2^{64}](2^{64}$ 为 long 类型的取值范围大小),$RT=(Gen,Div)$由以下两种算法组成:

(1) RT 生成算法 Gen:{min,max,$level$,num}$\leftarrow Gen(|V|,key)$,其中,key 表示用户的密钥;$|V|$表示值域的大小;min 和 max 定义了 RT 的针对 key 的值域 V=[min,max],且 $\min=|V|\times(-1)^{(key \bmod 2)}\times random(key)$,$\max=\min+|V|$,其中,$random(key)\in(0,1)$;$level$ 表示树的高度;num 定义了每个节点的子节点的数量,满足:

$$num^{level}\geqslant|D|;$$

(2) 区间划分算法 Div:$node_{x_i}$ ($i\in Z$ 且 $i\in[1,num]$)}$\leftarrow Div(node_x,key,i)$,其中,

- $node_x$ 是当前待划分区间的节点,它的 4 个属性分别是:$node_x.value$ 表示该节点对应的明文值,$node_x.level$ 表示该节点处于 RT 中的层数,$node_x$.min 和 $node_x$.max 分别表示其区间的最小值和最大值;
- i 是待求子节点 $node_{x_i}$ 的编号,即,$node_{x_i}$ 是 $node_x$ 的第 i 个子节点.

如图 2 所示,随机树首先将整个值域 V=[min,max]根据密钥 key 分为值域 vf_0 和间隔域 gf_0 两部分,然后对 vf_0 按照子节点的个数平均分为 num 份,再根据 key 将 gf_0 分为大小不等的 $m(m\in Z$ 且 $m\leqslant num)$份,并分给随机选择的 m 个子节点,每一层按照以上原理依次往下分裂.除 L=0 层外,以下各层分配的值域、间隔域大小以及间隔域的分裂份数和选择哪些子节点由 key 和该节点本身代表的值共同决定.

定义 3(OPEART). 设 OPEART 的定义域为 D,值域为 V,$OPEART=(Gen_RT,Enc,Dec)$由以下 3 种算法组成:

(1) RT 生成算法 Gen_RT:$RT\leftarrow Gen(|V|,key)$,其中,$key$ 表示用户的密钥;

(2) 确定的加密算法 Enc:$\forall m\in D,c\leftarrow Enc(RT,key,m)$且 $c\in V$;

(3) 确定的解密算法 Dec:对于密文 c,$m/\perp\leftarrow Dec(RT,key,c)$,$\perp$表示无解.

定义 4(IND-DNCPA). 一种加密算法 E 是 IND-DNCPA(indistinguishability under distinct and neighbouring chosen plaintext attack)安全的,如果攻击者 A 发起以下攻击:

(1) A 选择一组明文 $M=\{x_1,x_2,\ldots,x_k\}(x_i\in D,x_i\neq x_j$ 且 $i,j\in[1,k])$向 E 进行加密询问,E 将加密的结果 $C=\{c_1,c_2,\ldots,c_k\}$返回给 A,A 不能通过$(x_i,c_i)(i\in[1,k])$解出其他密文 $c(c\notin C)$对应的明文;

(2) A 选择两个数据 m_0,m_1 发给 E,其中 $m_1=m_0+1$,且 $m_0\notin M,m_1\notin M$;

(3) E 随机加密其中一个数据 $m_b(b\in\{0,1\})$,并将产生的密文 c_b 返回给 A;

(4) A 输出 b'作为对 b 的猜测.

A 的优势概率定义为

$$Adv(A)=|\Pr[b'=b]-0.5|=\varepsilon.$$

如果ε足够小,可忽略不计,则 E 是 IND-DNCPA 安全的.

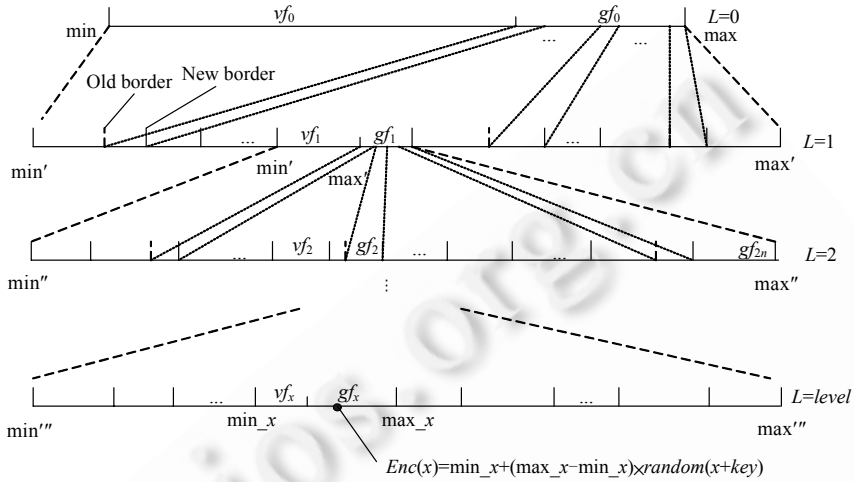


Fig.2 Schematic of RT

图 2 随机树原理的示例图

定义 5(随机性强度). OPEART 基于随机树和用户的密钥为定义域中的每个点分配了大小不等的密文区间,从而实现了对数据的加密,其随机性强度通过以下两个参数来衡量:

(1) 值参数(parameter of value,简称 pv):

假设任取连续的 k 个点 $\{x_1,x_2,\ldots,x_k\}(x_i\in D,i\in[1,k],k\geq 2)$,定义平均距离 $d=(Enc(x_k)-Enc(x_1))/(k-1)$,点 x 的参照值可以通过函数 $f(x)=Enc(x_1)+(x-x_1)\times d$ 计算得到,则有:

$$pv=\frac{k}{\sum_{i=1}^{k}\sqrt{|Enc(x_i)^2-f(x_i)^2|}} \tag{1}$$

pv 反映了 OPEART 产生的密文与参考值的差距大小,变化越大,pv 的值越小,攻击者通过密文猜测明文就越困难.

(2) 值变化参数(parameter of value change,简称 pvc):

假设任取连续的 k 个点 $\{x_1,x_2,\ldots,x_k\}(x_i\in D,i\in[1,k],k\geq 2)$,平均距离为 $d=(Enc(x_k)-Enc(x_1))/(k-1)$,用 $d'=(Enc(x_i)-Enc(x_{i-1}))(i\in[2,k])$来表示相邻两点间密文值的差距.假设 OPEAT 的值域为 V,其中共有 n 个点,则 $d'\in[1,V-n]$.下面将 d'的取值范围$[1,V-n]$分为 5 个区间,每个区间赋予一个权值:$(0,d/2)$的权值为 1,$(d/2,d)$的权值为 2,$(d,d\times 3/2)$的权值为 3,$(d\times 3/2,2\times d)$的权值为 4,$(2\times d,V-n)$的权值为 5.假设在以上的 k 个点对应的密文值的差距中,在以上 5 个区间中的比例分别为(p_1,p_2,p_3,p_4,p_5),下面定义值变化参数 pvc 如下:当 $p_i\neq 0(i\in[1,5])$时,

$$pvc=p_1+p_2\times 2+p_3\times 3+p_4\times 4+p_5\times 5 \tag{2}$$

由于平均距离为 d,所以平均值变化参数 pvc_{avg} 为

$$pvc_{avg}=2 \tag{3}$$

最大值变化参数 $pvc_{\max}$ 可通过下面的方程组求出:

$$\begin{cases} p_1+p_2+p_3+p_4+p_5=1 \\ p_1\times\left(\dfrac{0+\frac{d}{2}}{2}\right)+p_2\times\left(\dfrac{\frac{d}{2}+d}{2}\right)+p_3\times\left(\dfrac{d+\frac{3d}{2}}{2}\right)+p_4\times\left(\dfrac{\frac{3d}{2}+2d}{2}\right)+p_5\times\left(\dfrac{2d+\frac{5d}{2}}{2}\right)=d \end{cases} \tag{4}$$

解方程(4)可得:

$$pvc_{\max}=2.5 \tag{5}$$

因此,当 $pvc=pvc_{\max}$ 时,密文值之间的差距变化最大.注意,为了方便计算,方程(4)假设第 5 区间的范围为(2×d,5×d/2],因此 $pvc_{\max}$=2.5;当实际的连续密文距离超过 5×d/2 时,$pvc_{\max}$>2.5.这对安全性分析没有影响.

3.2 算法描述

本节先对 RT 的区域划分算法 $Div(\cdot)$进行描述,再对 OPEART 的加密算法 $Enc(\cdot)$和解密算法 $Dec(\cdot)$进行描述.

1) $Div(\cdot)$算法用于求出 $node_x$ 的第 i 个子节点 $node_{x_i}$ 的区间范围.其原理是:将节点 $node_x$ 的区间[$node_x$.min, $node_x$.max]随机地分为值域 vf 和间隔域 gf 两部分,其中,vf 部分被均匀的划分为 num 份;而 gf 部分则使用密钥 key 和本节点对应的值作为随机运算的种子,随机地划分为 m($m\in\mathbf{Z}$ 且 $m\leqslant num$)份并分给随机选择的 m 个子节点,第 j 份表示为 gf_j,其中的第 j 个子节点的区间大小($node_x$.max−$node_x$.min)/num+gf_j.

算法 1. RT 的区域划分算法 $Div(\cdot)$.

输入:待划分区域的节点 $node_x$、密钥 key、待求子节点的编号 i.

1: min=$node_x$.min; max=$node_x$.max;
2: If $node_x.level$==0 then
3: $ran\_root=key$; //产生随机数的种子
4: Else
5: $ran\_root=key+node_x.value$;
6: End If
7: 根据 ran_root 分割数据域(min,max)产生值域 vf 和间隔域 gf;
8: $d=|vf|/num$;
9: 根据 ran_root 产生间隔域 gf 的分裂份数 m 和选择 m 个子节点:
$Node=\{node'_{x_1},...,node'_{x_m}\}$, 满足:

$$\left(\sum_{j=1}^{m} random(key+node'_{x_j}.value)\right)\Big/base\leqslant 1, \text{且}\left(\sum_{j=1}^{m+1} random(key+node'_{x_j}.value)\right)\Big/base>1;$$

//$base$ 为间隔域的分割基数(见第 3.3.4 节),可调整值参数 pv 和值间隔参数 pvc;
10: For j=0 to i do
11: If $node_{x_j}\in Node$ then
12: $node_{x_j}.\min=\min$;
13: $node_{x_j}.\max=\min+d+|gf|\times random(key+node_{x_j}.value)$;
14: Else
15: $node_{x_j}.\min=\min$;
16: $node_{x_j}.\max=\min+d$;
17: End If
18: $\min=node_{x_j}.\max+1$;
19: End For
20: $node_{x_i}.value=node_x.value\times num+i$;
21: $node_{x_i}.level=node_x.level+1$;

2) 加密算法 $Enc(\cdot)$对数据进行加密,即,要寻找一条从随机树的根节点到加密数据对应的叶子节点的路径.

算法 2. OPEART 的加密算法 $Enc(\cdot)$.

输入:值域 V=[min,max]、随机树 RT、密钥 key、待加密数据 x.

1: 在 x 的前面补 0,使得补位后的 x'满足$|x'|=|\max-1|$;将 x'分解为 *RT.level* 段,第 i 段的值用 $x[i]$表示;
2: $node=node_{root}$,其中,$node_{root}$.min=min,$node_{root}$.max=max,$node_{root}.level=0$,$node_{root}.value=0$,
3: For L=0 to *RT.level* do
4: For j=0 to *RT.num* do //判断 $x[L]$对应 *node* 的哪个子节点
5: $node_j.value=node.value\times num+j$; //$node_j$ 表示 *node* 的第 j 个子节点
5: If $x[L]==node_j.value$ then
6: $i=j$; break;
7: End If
8: End For
9: $node=RT.Div(node,key,i)$;
10: If $L==RT.level$ then
11: $ciphertext=node.\min+(node.\max-node.\min)\times random(key+x[L])$;
12: End If
13: End For
14: return ciphertext;

3) 解密算法 $Dec(\cdot)$也是要寻找一条从随机树的根节点到加密数据对应的叶子节点的路径.

算法 3. OPEART 的解密算法 $Dec(\cdot)$.

输入:值域 V=[min,max]、随机树 RT、密钥 *key*、待解密数据 *val*.

1: $node=node_{root}$,其中,$node_{root}$.min=min,$node_{root}$.max=max,$node_{root}.level=0$,$node_{root}.value=0$,
2: For L=0 to *RT.level* do
3: For j=0 to *RT.num* do //判断 *val* 属于 *node* 的哪个子节点的区间
4: $node_j=RT.Div(node, key,j)$;
5: If $val\in[node_j.\min,node_j.\max]$ then
6: $i=j$; break;
7: End If
8: End For
9: $node= node_j$;
10: $plaintext=plaintext+node.value$; //字符串连接
11: End For

3.3 关键问题

3.3.1 密钥的值域和字符串的基本长度

OPEART 的密钥 *key* 的值域用 V_{key} 来表示.字符串的基本长度 *len* 定义了 OPEART 一次能处理的最长字符串长度.当一个字符串 S 的长度$|S|>len$ 时,在 S 后补充最少的随机字符得到 S',满足:$|S'| \bmod len==0$.然后,将 S'分割成(S'/len)段,每一段字符串作为一个基本字符串进行处理.

密钥的值域和字符串的基本长度直接影响到 OPEART 算法的安全性强度,分析如下:

攻击者 A 可能会发起以下两种袭击:

(1) 当 A 得到一对明密文对(x,c)时,可通过遍历法求出 *key*.假设 OPEART 加密一次基本字符串所需要的时间为 t,则 A 攻破密钥的平均时间 T 为

$$T=\left(t\times\frac{|x'|}{len}\right)^{|V_{key}|} \tag{6}$$

其中,x'是在 x 后补充随机字符后得到的字符串.根据公式(6)可知,指数越大,T 值越大,A 攻破密钥的难度就越大.

(2) 当 A 知道基本字符串长度 *len* 后,可以通过密文 c 的长度 len_c 判断出明文 x 的长度 len_x;然后,通过分析

密文的出现频率和长度为 len_x 的明文字符的各种组合的使用频率来推断明文 x.假设明文字符的集合为 D_x,长度为 len 的字符串加密后的密文长度为 len_b,则 A 要完成以上分析的计算复杂度 O_c 为

$$O_c = O\left(|D_x|^{\left(len\times\frac{len_c}{len_b}\right)}\right) \tag{7}$$

根据公式(7)可见,O_c 的最小值为 $O(D_x|^{len})$.因此,len 越长,A 推断明文的难度就越大.

OPEART 采用 long 类型作为密钥 key 的类型,即:$key\in[-9223372036854775808,9223372036854775807]$,$|V_{key}|=2^{64}$,字符串的基本长度 len=12(由 RT 的定义域可知).通过实验得出:当 $RT.level$=7,$RT.num$=100 时, OPEART 加密一次基本字符串所需要的时间 t=0.088ms.假设已知一个长度为 len 的明密文对,则根据公式(6), A 要破解密钥需要 2.9 亿年;根据公式(7),假设明文字符都是数字,A 要进行统计分析,其计算复杂度为 $O(10^{12})$.因此,OPEART 采用 long 类型作为密钥 key 的类型,字符串的基本长度 len=12 能够有效抵御以上两种攻击.

3.3.2 RT 的高度和宽度的平衡

RT 的高度指树的层数 $RT.level$,宽度表示每一个节点的子节点数 $RT.num$,假设 RT 的定义域为 D,则:

$$|D|=RT.num^{(RT.level-1)} \tag{8}$$

满足公式(8)的 $RT.num$ 和 $RT.level$ 的值并不是唯一的,例如,当$|D|=10^{12}$ 时,($RT.num$,$RT.level$)可以取以下几种值:v_1=(10,13),v_2=(100,7),v_3=(1000,5),v_4=(10000,4).图 3 反映了以上几种值对对 OPEART 加解密性能的影响,其中,$length$ 为明文的长度,"Enc_v_1"表示在 v_1 的情况下的加密时间,其他符号以此类推.如图 3 所示,($RT.num$, $RT.level$)的取值对加密和解密时间都有影响,对解密时间影响最大.其原因是,根据算法 $Div(\cdot)$的第 10 步,要判断一个子节点 $node_{x_j}$ 是否属于 $Node$,这是一个遍历的过程,对于加密操作,从根节点找到对应的叶子节点的计算复杂度为 $O(RT.num\times RT.level)$;对于解密操作,从根节点找到对应的叶子节点的计算复杂度为 $O(RT.num\times RT.num\times RT.level)$.因此在实现 RT 的时候,要根据 RT 的定义域,选择合适的高度和宽度,从而获得较好的效率.

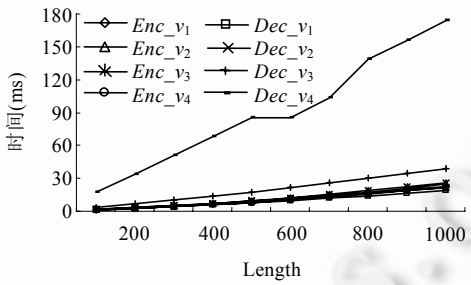


Fig.3 Effects of the height and width of RT on OPEART performance

图 3 RT 的高度和宽度对 OPEART 性能的影响

3.3.3 随机数的选取

在 RT 中,划分节点 $node_x$ 的区间 D_x=[$node_x$.min,$node_x$.max]时,以密钥 key 和 $node_x.value$ 作为选取随机数的根,从而生成随机数 ran_x($ran_x\in R$ 且 $ran_x\in(0,1)$),然后根据 ran_x 将 D_x 分为值域 vf 和间隔域 gf 两部分.假设 $node_x$ 的直接子节点和间接子节点的个数为 Sum_x,为使 vf 能够容纳下所有子节点且能够保持随机性,ran_x 应满足如下要求:

(1) 当 $Sum_x/D_x\leqslant 0.3$ 时,$ran_x\in(0.3,0.7)$;

(2) 当 $Sum_x/D_x\in(0.3,0.7)$时,$ran_x\in(Sum_x/D_x,0.7)$;

(3) 当 $Sum_x/D_x\geqslant 0.7$ 时,$ran_x\in(Sum_x/D_x,1)$.

3.3.4 间隔域的分割基数

间隔域的分割基数 $base$($base\in[1,RT.num]$)限定了分割间隔域的最小份数.根据算法 1,将间隔域 gf 根据 ran_root 随机分裂为大小不等的 m 份,再根据 ran_root 随机选择 m 个子节点 $Node=\{node'_{x_1},...,node'_{x_m}\}$,满足:

$$\left(\sum_{j=1}^{m} random(key+node'_{x_j}.value)\right)\Big/base\leqslant 1,且\left(\sum_{j=1}^{m+1} random(key+node'_{x_j}.value)\right)\Big/base>1.$$

$Node$ 中的子节点 $node'_{x_i}$ 可分配得到大小为 $|gf|\times random(key+node'_{x_i}.value)$ 的子间隔,因此,间隔域分裂时产生的子间隔大小是随机的,分裂份数 m 是不确定的,但 $m\geqslant base$.间隔域主要用于实现对数据分布的干扰,一方面,m 越大,受干扰的子节点就越多;另一方面,如果 m 太大,那么,虽然受干扰的子节点个数多,但对每个节点的干扰不大.因此,选择合适的分裂基数将直接影响到 RT 的随机性.

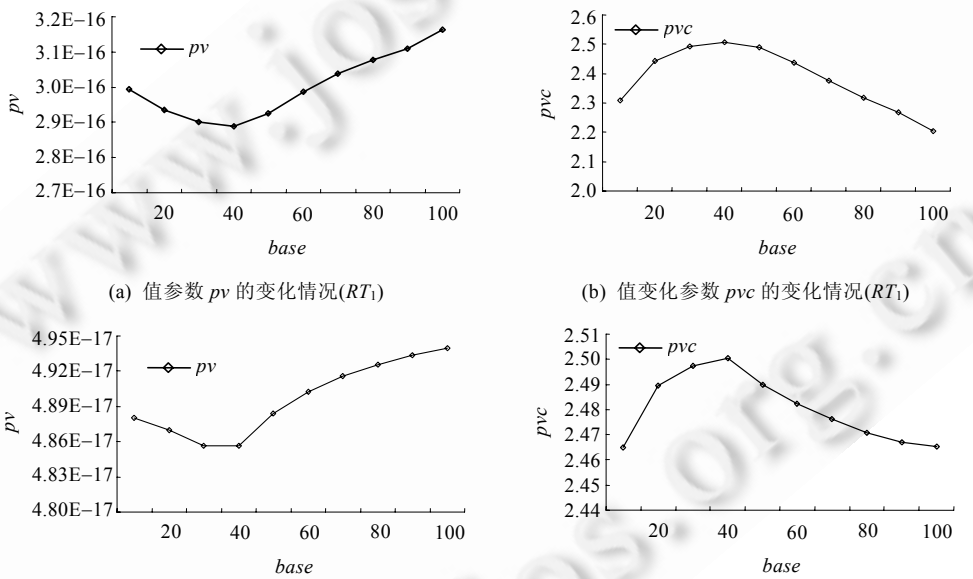
图 4 反映了间隔域分裂基数 $base$ 对 RT 随机性的影响,其中,

$$RT_1.level=7,RT_1.num=100;RT_2.level=5,RT_2.num=1000.$$

如图 4(a)和图 4(b)所示,对于 RT_1 来说:

- 当 $base$=40 时,pv 的值最小,pvc 的值最大.由 RT 的随机性强度定义可知,这时,RT_1 的随机性最好;
- 当 $base$<40 时,受干扰的子节点个数不够多,因此随机性不够强;
- 当 $base$>40 时,每个节点的干扰区域太小,因此随机性也不够强.

根据图 4(c)和图 4(d)所示,对于 RT_2 来说:当 $base$=400 时,pv 的值最小,pvc 的值最大.这时,RT_2 的随机性最好.



(a) 值参数 pv 的变化情况(RT_1)　(b) 值变化参数 pvc 的变化情况(RT_1)

(c) 值参数 pv 的变化情况(RT_2)　(d) 值变化参数 pvc 的变化情况(RT_2)

Fig.4 Effects of segmentation base on RT randomness

图 4 分割基数 base 对 RT 随机性的影响

通过以上分析可知,间隔域分割基数对 RT 的随机性是有直接影响的,当 $base$=$RT.num$×40%时,RT 的随机性最好.

3.3.5 拐 点

在 RT 的 $Div(\cdot)$算法中,对于 $node_x$ 节点,根据密钥 key 和 $node_x.value$ 随机地将 $node_x$ 的区域 D_x 分为值域 vf 和间隔域 gf 两部分,然后再以 key 和 $node_x.value$ 为根,随机地将间隔域 gf 分裂成 m 份,并随机地选择 m 个子节点构成子节点集合 $Node=\{node'_{x_1},...,node'_{x_m}\}$,满足:

$$\left(\sum_{j=1}^{m} random(key+node'_{x_j}.value)\right)\Big/base\leqslant 1,且\left(\sum_{j=1}^{m+1} random(key+node'_{x_j}.value)\right)\Big/base>1.$$

因此,在进行加解密时,当判断某一子节点 $node_{x_i}$ 是否属于 $Node$ 时,平均需要进行(m/2)次取随机数运算和

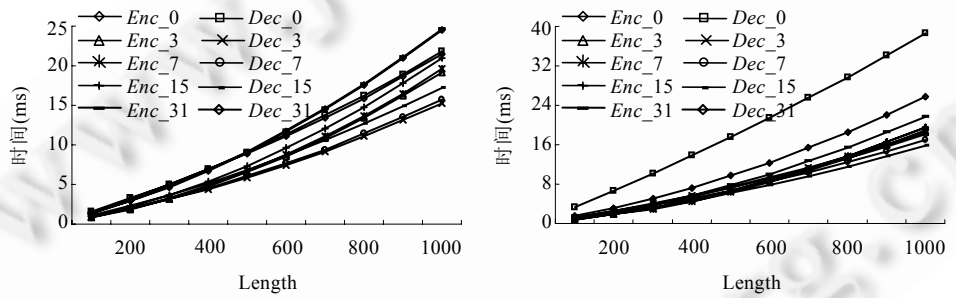
加法运算.当间隔域的分割基数为 *RT.num*×40%,OPEART 的随机性最好,则 $m \geqslant RT.num \times 40\%$.因此,加解密一个基本字符串 *S* 时,除最底层以外的各层都要进行上述判断,其平均计算复杂度 O_r 为

$$O_r = O\left(\frac{RT.num}{5} \times (RT.level - 1)\right) \tag{9}$$

例如,当 *RT.num*=1000,*RT.level*=5 时,一共需要进行 800 次随机运算和加法运算,这是一个较大的计算负载.因此,OPEART 引入了拐点的概念.

引入拐点的目的是对间隔域 $gf=[g_{\min},g_{\max}]$进行粗粒度的划分,得到多个大小不等的桶,每个桶的边界即为拐点.生成拐点的步骤如下:采用二叉树结构,在二叉树的第 1 层,根据 *key* 和 $node_x.value$ 取随机值 $i(i \in \mathbf{R}$,且 $i \in (0.3,0.7))$,拐点为 $x=g_{\min}+\lceil (g_{\max}-g_{\min}) \times i \rceil$,其中,限定 $i \in (0.3,0.7)$是为了尽可能平均分割 *gf*.于是,拐点 *x* 将 *gf* 分割成$[g_{\min},x]$和$(x,g_{\max})$两个桶.接下来,递归的对两部分进行划分,直至产生足够的拐点.假设有 *n* 个拐点,则 *gf* 被分成了(*n*+1)个桶.

判断某一子节点 $node_{x_i}$ 是否属于 *Node* 时,首先按照拐点生成的方法寻找一条从树根到叶子节点的路径,确定 $node_{x_i}$ 对应的桶,然后再对桶内的子节点逐一比较,判断它是否在桶内即可.实际上,拐点实现了对 *Node* 中的 *m* 个子节点的近似折半查找.合适的拐点个数 *n* 要依据 *RT.level* 和 *RT.num* 来确定.图 5 反映了拐点个数对 OPEART 加解密性能的影响,其中,*length* 为明文的长度,"*Enc*_0"表示在拐点个数 *n*=0 时的加密时间,其他符号以此类推.



(a) *RT.num*=100, *RT.level*=7 (b) *RT.num*=1000, *RT.level*=5

Fig.5 Effects of number of RT inflexion on OPEART performance

图 5 RT 拐点个数对 OPEART 性能的影响

如图 5(a)所示,在 *RT.num*=100,*RT.level*=7 的情况下,当拐点个数 *n*=3 时,OPEART 的加解密效率最高;其次是 *n*=7 的情况;当 *n*=32 时,与 *n*=0 的情况相似,OPEART 的加解密效率最低.如图 5(b)所示,在 *RT.num*=1000, *RT.level*=5 的情况下,当拐点个数 *n*=7 时,OPEART 的加解密效率最高;其次是 *n*=15 的情况;当 *n*=0 的时,即,没有拐点的情况下,OPEART 的加解密效率最低.

根据以上分析可知,拐点可以提高 OPEART 的加解密效率,但合适的拐点个数要依据 *RT.level* 和 *RT.num* 来确定.

4 OPEART 的正确性和安全性

定理 1. *OPEART*=(*Gen_RT*,*Enc*,*Dec*)是正确的,如果满足以下条件:

(1) $\forall m \in D, \exists Dec(RT,key,Enc(RT,key,m))=m$;

(2) 如果 $x_1>x_2(x_1,x_2 \in D)$,那么 $Enc(x_1)>Enc(x_2)$.

证明:

(1) 根据 RT 的结构,每个节点对应定义域中的一个点,每个子节点的值域是对父节点值域的任意划分且没有交集,因此保证了任何密文值只能属于一个叶子节点的值域空间.另外,如算法 2 和算法 3 所示,加密和解密的

过程都是寻找一条从 RT 的根节点到明文值对应的叶子节点的路径.不同的是,在加密时,由于知道明文 x,因此只需调用 $RT.Div(\cdot)$算法一次,根据由密钥 key 和 x 构成的 ran_root 直接求出分裂份数 m 和选择 m 个子节点集合 $Node$;在解密时,由于不知道 x,因此要逐一检查节点 $node_x$ 的每个子节点 $node_{x_j}$,根据由 key 和 $node_{x_j}.value$ 构成的 ran_root 产生 $node_{x_j}$ 的值域,然后判断密文 val 是否属于该子节点的值域,从而找到对应的明文,也就是说,要多次调用 $RT.Div(\cdot)$算法.因此,只要加密和解密的过程中使用相同的密钥 key,就能找到相同的一条路径.

因此,$\forall m\in D,\exists Dec(RT,key,Enc(RT,key,m))=m$.

(2) 根据 RT 的结构,每个子节点的值域是对父节点值域的任意划分且没有交集,假设 $node_x$ 的两个相邻的子节点为 $node_{x_j}$ 和 $node_{x_{j+1}}$,其中 $node_{x_{j+1}}.value=node_{x_j}.value+1$,则有 $node_{x_{j+1}}.\min=node_{x_j}.\max+1$.

因为 $Enc(node_{x_j}.value)=node_{x_j}.\min+Random(key+node_{x_j}.value)\times(node_{x_j}.\max-node_{x_j}.\min)$,

$Enc(node_{x_{j+1}}.value)=node_{x_{j+1}}.\min+Random(key+node_{x_{j+1}}.value)\times(node_{x_{j+1}}.\max-node_{x_{j+1}}.\min)$,

所以, $Enc(node_{x_j}.value)<Enc(node_{x_{j+1}}.value)$.

所以,OPEART 是保序的.

综上所述,OPEART 是正确的. □

定理 2. OPEART 是 IND-DNCPA 安全的.

证明:假设攻击者 A 发起以下攻击:

(i) 选择一组明文 $M=\{x_1,x_2,\ldots,x_k\}(x_i\in D,x_i\neq x_j$ 且 $i,j\in[1,k])$向 OPEART 进行加密询问,OPEART 将加密的结果 $C=\{c_1,c_2,\ldots,c_k\}$返回给 A,A 试图通过解方程的方式解出其他密文 $c(c\notin C)$对应的明文;

(ii) A 选择两个数据 m_0,m_1 发给 OPEART,其中,$m_1=m_0+1$,且 $m_0\notin M,m_1\notin M$;

(iii) OPEART 随机加密其中一个数据 $m_b(b\in\{0,1\})$,并将产生的密文 c_b 返回给 A;

(vi) A 输出 b'作为对 b 的猜测.

(1) OPEART 通过以下操作产生了随机性,从而使得从明文到密文的映射关系是无法通过解方程的方式来破解的:

(i) 对于非叶子节点 x_i,以密钥 key 和 x_i 来决定该点的值域和间隔域的大小,使得不同的点具有不同的值域和间隔域大小;

(ii) 对于非树根节点 x_j,以 key 和 x_j 作为随机数的根,将父节点的间隔域随机地分成大小不等的 m_j 份,并分配任选的 m_j 个子节点;

(iii) 对于叶子节点 x_k,以 key 和 x_k 产生随机数在$(\min_k,\max_k)$之间选择一点作为 x_k 的密文.

(2) 由于 OPEART 具有保序性,A 可能通过分析 M 中密文的差距以及与 m_0 或 m_1 最近的点的密文来推断 b.M 中密文的差距受 OPEART 的随机性的影响:

(i) 当 A 通过求出平均距离 $d=(Enc(x_k)-Enc(x_1))/(k-1)$来猜测 x 的值时,由于 $pv\to0$,说明由于 OPEART 的随机性,使得密文与参照函数 $f(x)=Enc(x_1)+(x-x_1)\times d$ 相差很大,因此,A 无法从密文直接猜测出对应的明文;

(ii) 当 A 通过分析密文之间的距离 d'来推测密文的值域时,由于 $pvc>pvc_{avg}$,说明相邻点密文之间的变化超过了 d,尤其是 $p_i\neq0(i\in[1,5])$且 $pvc\to pvc_{\max}$ 时,处于 5 个变化区间的百分比达到了最大值,d'的波动最大,即,$d'\in[1,V-n]$,使得 A 要判断值域的大小变得很困难.因此,A 的优势概率为

$$Adv(A)=|\Pr[b'=b]-0.5|=|(0.5+\varepsilon)-0.5|=\varepsilon,$$

其中,ε是一个足够小可忽略不计的数.

综上所述,OPEART 是 IND-DACPA 安全的. □

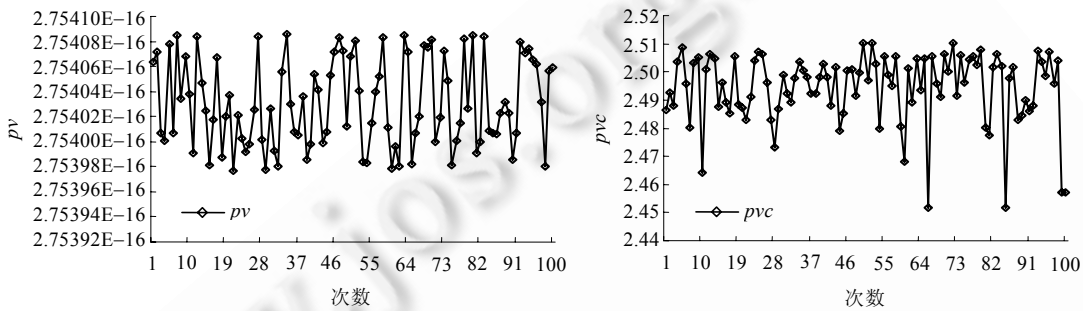
5 OPEART 的性能

本节将通过测试OPEART的随机性能判断其是否能够达到IND-DACPA安全性的要求;通过对比OPEART

和 OPES 的性能来分析 OPEART 的效率,性能指标包括加、解密性能,关系运算性能以及存储/通信负载等.实验部署在由本研究小组自行研发的青云实验平台 3.0 上,该平台是一个基于 Web 的面向校园的云计算环境,以 KVM 和 Hadoop 的 HDFS 为底层支撑技术,并部署在由 12 台服务器构成的集群上.

5.1 随机性能

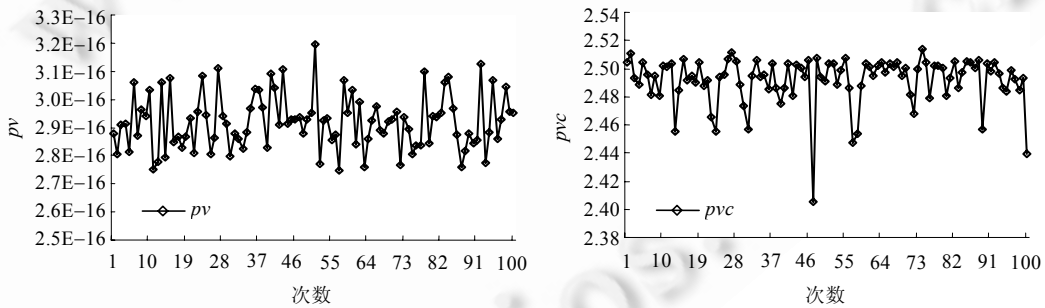
实验 1 测量 OPEART 在相同密钥下不同数据组的 *pv* 值和 *pvc* 值,每次随机抽取 10 000 个连续的数进行加密,重复 100 次,如图 6 所示.实验 2 测量 OPEART 在不同密钥下同一组数据的 *pv* 值和 *pvc* 值,每次随机生成密钥,对同一组由 10 000 个连续的数组成的数据组进行加密,重复 100 次,如图 7 所示.在测试 *pvc* 值时,当出现 p_i=0 ($i\in[1,5]$)时,*pvc*=0.



(a) 相同密钥下不同的数据组的值参数 *pv* (b) 在相同密钥下不同数据组的值变化参数 *pvc*

Fig.6 Randomness of OPEART under the same keys

图 6 OPEART 在相同密钥下的随机性



(a) 在不同密钥下同一组数据的值参数 *pv* (b) 在不同密钥下同一组数据的值变化参数 *pvc*

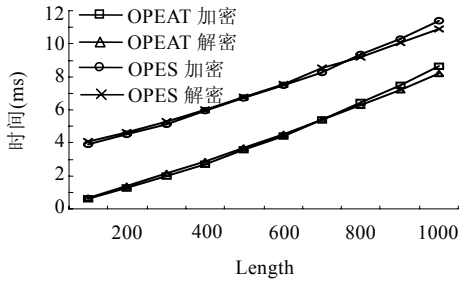
Fig.7 Randomness of OPEART under the different keys

图 7 OPEART 在不同密钥下的随机性

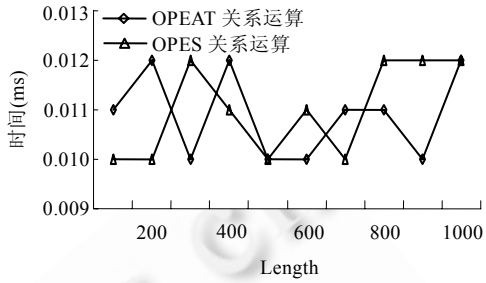
如图 6(a)和图 7(a)所示,*pv* 值均小于 10^{-15},是一个足够小的数,因此满足 $pv\to 0$ 的随机性要求;如图 6(b)和图 7(b)所示,*pvc* 值均大于 2.4,满足 $p_i\neq 0(i\in[1,5])$,$pvc>pvc_{avg}$ 且 $pvc\to pvc_{\max}$ 的要求.在某些点的 *pvc* 值大于 2.5,那是因为处于第五区间[(2×*d*,*V*−*n*)]的连续密文之间的差距的平均距离大于 9×*d*/4.实验 1 和实验 2 表明:OPEART 具有很好的随机性能,能够满足 IND-DACPA 的安全性要求.

5.2 OPEART的效率

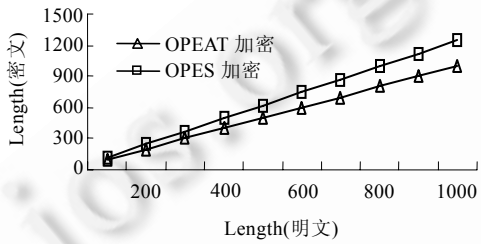
实验 3 比较了 OPEART 与 OPES 在加解密、关系运算以及存储/通信负载这 3 个方面的性能,其中,OPEART 的 *level*=7,*num*=100;OPES 具有和 OPEART 相同的数据域和值域,输入分布函数和输出函数的分桶数分别为 100 和 150.结果如图 8 所示.



(a) 加解密性能的比较



(b) 关系运算性能的比较



(c) 存储/通信负载的比较

Fig.8 Efficiency of OPEART

图 8 OPEART 的效率

如图 8(a)所示,OPEART 的加密和解密负载几乎是一样的,而且均比 OPES 的加解密负载低;在进行关系运算时,OPEART 与 OPES 的运算负载相差不大,基本上是在 0.01ms~0.013ms 之间;对于相同长度的明文,OPEART 的密文长度小于 OPES.总的说来,OPEART 比 OPES 具有更好的运行效率.

6 结束语

针对云计算中的隐私保护问题,本文提出了支持隐私保护的云计算模型,并设计了一种支持隐私保护的加密方案 OPEART.OPEART 是一种保序的、确定的对称加密方案,支持对加密数据的任何一种关系运算(=,>,<等).安全分析和性能评估证实:OPEART 具有 IND-DNCPA 安全性,并且具有很好的运行效率.

下一步,我们将从提高安全性的角度对 OPEART 进行改进,同时也计划研究新的加密技术来实现安全的密文计算.

References:

[1] Jessica EV. Google discloses privacy glitch (2009). 2011. http://blogs.wsj.com/ digits/2009/03/08/1214/

[2] Song DX, Wagner P, Perrig P. Practical techniques for searches on encrypted data. In: Proc. of the 2000 IEEE Symp. on Security and Privacy. 2000. 44−55. [doi: 10.1109/SECPRI.2000.848445]

[3] Bonech D, Crescenzo GD, Ostrovsky R, Persiano G. Public-Key encryption with keyword search. In: Proc. of the Eurocrypt 2004. 2004. 506−522.

[4] Ohtaki Y. Partial disclosure of searchable encrypted data with support for boolean queries. In: Proc. of the 3th Int'l Conf. on Availability, Reliability and Security (ARES 2008). 2008. 1083−1090. [doi: 10.1109/ARES.2008.80]

[5] Liu Q, Wang GJ, Wu J. An efficient privacy preserving keyword search scheme in cloud computing. In: Proc. of the 2009 Int'l Conf. on Computational Science and Engineering. IEEE, 2009. 715−720. [doi: 10.1109/CSE.2009.66]

[6] Boldyreva A, Chenette N, Lee Y, O'Neill A. Order-Preserving symmetric encryption. In: Proc. of the 28th Annual Int'l Conf. on Advances in Cryptology (Eurocrypt 2009). 2009. 224−241. [doi: 10.1007/978-3-642-01001-9_13]

[7] Belazzougui D, Boldi P, Pagh R, Vigna S. Monotone minimal perfect hashing. In: Proc. of the 20th Annual ACM-SIAM Symp. on Discrete Algorithms. 2009. 785−794.

[8] Fox EA, Chen QF, Daoud AM, Heath LS. Order-Preserving minimal perfect hash functions and information retrieval. ACM Trans. on Information Systems, 1991,9(3):281−308.

[9] Czech ZJ, Havas G, Majewski B. An optimal algorithm for generating minimal perfect hash functions. Information Processing Letters, 1992,43(5):257−264. [doi: 10.1016/0020-0190(92)90220-P]

[10] Agrawal R, Kiernan J, Srikant R, Xu YR. Order-Preserving encryptiion for numeric data. In: Proc. of the 2004 ACM SIGMOD Int'l Conf. on Management of Data (SIGMOD 2004). 2004. 563−574. [doi: 10.1145/1007568.1007632]

[11] Boldyreva A, Chenette N, O'Neil A. Order-Preserving encryption revisited: Improved security analysis and alternative solutions. In: Proc. of the 31st Annual Conf. on Advances in Cryptology. 2011. 578−595.

[12] Wong WK, Cheung DW, Kao B, Mamoulis N. Secure *k*NN computation on encrypted databases. In: Proc. of the 35th SIGMOD Int'l Conf. on Management of Data (SIGMOD 2009). 2009. 139−152. [doi: 10.1145/1559845.1559862]

[13] Boneh D, Sahai A, Waters B. Fully collusion resistant traitor tracing with short ciphertexts and private keys. In: Proc. of the Eurocrypt 2006. LNCS 4004, 2006. 573−592. [doi: 10.1007/11761679_34]

[14] Boneh D, Waters B. Conjunctive, subset, and range queries on encrypted data. In: Proc. of the TCC 2007. LNCS 4392, 2007. 535−554. [doi: 10.1007/978-3-540-70936-7_29]

[15] Shi E, Bethencourt J, Chan THH, Song D, Perrig A. Multi-Dimensional range query over encrypted data. In: Proc. of the 2007 IEEE Symp. on Security and Privacy. 2007. 350−364. [doi: 10.1109/SP.2007.29]

[16] Wang PH, Lakshmanan LVS. Efficient secure query evaluation over encrypted XML databases. In: Proc. of the 32nd Int'l Conf. on Very Large Databases. 2006. 127−138.

[17] Hacıgümüş H, Iyer B, Mehrotra S. Efficient execution of aggregation queries over encrypted relational databases. In: Proc. of the 9th Int'l Conf. on Database Systems for Advanced Applications (DASFAA 2004). 2004. 633−650. [doi: 10.1007/978-3-540-24571-1_10]

[18] Hacıgümüş H. Privacy in database-as-a-service model [Ph.D. Thesis]. Irivne: Department of Information and Computer Science, University of California, 2003.

[19] Wang C, Cao N, Li J, Ren K, Lou WJ. Secure ranked keyword search over encrypted cloud data. In: Proc. of the 30th Int'l Conf. on Distributed Computing Systems (ICDCS 2010). 2010. 253−262. [doi: 10.1109/ICDCS.2010.34]

黄汝维(1978－),女,广西岑溪人,博士,副教授,CCF 会员,主要研究领域为服务计算,云安全,同态加密.

陈宁江(1976－),男,博士,教授,CCF 高级会员,主要研究领域为软件工程,网络分布式计算,中间件技术.

桂小林(1966－),男,博士,教授,博士生导师,CCF 高级会员,主要研究领域为服务计算,网络安全.

姚婧(1987－),女,博士生,主要研究领域为云安全,同态加密.