支持浮点运算的高效并行全同态加密算法

史经启[1] 杨 庚[1,2] 孙彦珺[1] 白双杰[1] 闵兆娥[1]
(南京邮电大学计算机学院 南京 210003)[1] (江苏省大数据安全与智能处理重点实验室 南京 210023)[2]

摘 要 云计算的快速发展在给人们带来便利的同时,其隐私安全问题也备受关注。结合全同态加密算法,实现直接对密文的运算,是解决隐私安全问题的一种可行方案。但目前大多同态算法支持的数据类型有限,难以有效应用于实际环境。鉴于此,提出一种支持浮点运算的全同态加密算法,以及基于Spark环境的并行算法,并分析了算法的安全性和实际性能。实验结果表明,基于Spark的并行浮点数全同态加密算法支持整数和浮点同态运算,在4节点16核心的集群中能够达到3.9的整体加速比,能有效减少数据加密和密文运算的时间,满足云计算环境中对大规模浮点数据进行高效同态加密的需求。

关键词 全同态加密,浮点数加密,Spark,并行加密
中图法分类号 TP309.7 **文献标识码** A **DOI** 10.11896/j.issn.1002-137X.2018.05.020

Efficient Parallel Algorithm of Fully Homomorphic Encryption Supporting Operation of Floating-point Number

SHI Jing-qi[1] YANG Geng[1,2] SUN Yan-jun[1] BAI Shuang-jie[1] MIN Zhao-e[1]
(School of Computer Science, Nanjing University of Posts and Telecommunications, Nanjing 210003, China)[1]
(Jiangsu Key Laboratory of Big Data Security & Intelligent Processing, Nanjing 210023, China)[2]

Abstract The rapid development of cloud computing provide convenience to people, as well as security problems, such as privacy preserving. One of the major ways to solve this problem is to utilize fully homomorphic encryption(FHE) algorithm to support operations on the encrypted data directly. However, because most fully homomorphic encryption schemes only support limited data types for the time being, it is difficult to apply them to reality. In this paper, a fully homomorphic encryption algorithm supporting floating-point operations and a parallelization algorithm based on Spark were proposed. The security and performance of the parallel algorithm were analyzed in theory and experiments were conducted to demonstrate its practical performance. Experimental results show that the overall speed-up ratio of the given algorithm can reach 3.9 in a 4-node 16-core cluster and the encryption time and calculation time on encrypted data can be reduced effectively. The parallel fully homomorphic encryption algorithm can satisfy the encryption requirement of large-scale data in cloud environment.

Keywords Fully homomorphic encryption, Floating-point encryption, Spark, Parallel encryption

1 引言

随着计算机技术的快速发展,云计算、大数据等概念逐渐被人们所认识和接受,成为人们生活中的一部分;移动化浪潮的到来更是使用户通过手机等移动终端连接到云端,用户的个人隐私数据也更多地存储在应用服务提供商的服务器中。随着越来越多的数据和个人信息被汇集并存储到应用提供商的服务器中,以及云安全事故频发,大数据、云计算以及隐私保护等问题逐渐成为热门话题[1]。来自外部的攻击者或内部"不怀好意"的数据库管理员都可能造成用户隐私信息的泄露,给用户造成困扰和不便[2]。如何保证存储在不可信云端服务器中的数据的安全性,尤其是用户敏感的隐私数据,成为学术界研究的重点。

一种解决方案是将隐私数据进行加密存储,将所有的计算放在客户端进行,如SPORC[3],SUNDR[4],Depot[5]。但是这类方案只是将云服务提供商的资源当作存储设备来使用,这既降低了云端卓越的计算能力,又加重了客户端的计算压力;并且现有的应用服务模式并不适合这类方案,对各类应用进行改造使其适应这类方案将极为困难。

另一种解决方案是使用同态加密算法(homomorphic encryption)HOM,HOM是一种安全的概率加密方案(IND-CPA安全)。不同于以往秉承"数据存储安全"理念的加密算

到稿日期:2017-03-13 返修日期:2017-06-05 本文受国家自然科学基金资助项目(61572263,61502251),中国博士后科学基金资助项目(2016M601859),江苏省高校自然科学研究项目(14KJB520031)资助。

史经启(1990-),男,硕士生,主要研究方向为隐私保护、同态加密;**杨 庚**(1961-),男,博士,教授,CCF高级会员,主要研究方向为网络与信息安全、分布与并行计算、大数据隐私保护,E-mail:yangg@njupt.edu.cn(通信作者);**孙彦珺**(1992-),女,硕士生,主要研究方向为隐私保护、保序加密;**白双杰**(1992-),男,博士生,主要研究方向为信息安全、隐私保护;**闵兆娥**(1978-),女,博士,主要研究方向为信息安全、隐私保护、并行计算。

法，同态加密的关键之处在于其关注点为“数据处理安全”，允许人们直接对密文进行特定的数学运算操作，但是处理过程不会泄露任何原始内容[6]，只有持有密钥的用户才可以解密密文，获得期待的计算结果。同态加密技术为云计算技术提供了安全保障：用户将数据及对数据的操作权委托给第三方而不泄露自身隐私，保证了自身数据的安全性。HOM 理论上允许服务器端对加密后的密文数据进行任意操作，且服务器端永远存储密文，除了拥有解密密钥的用户，其他人员无法查看明文信息。这一特性适用于云计算场景，满足将计算托付给云端计算资源的需求，兼顾了数据的安全性和实用性。

1978 年，Rivest 等[7]首次提出同态加密的概念，其又被称为“隐私同态”。同态加密的主要思想为设计一套加密方案，使得能够在不进行解密操作的情况下对密文进行任意的运算，并且解密后得到的结果即为明文对应操作得到的结果。此后，国内外学者相继提出了多种同态加密方案，如具有乘法同态特性的 Elgamal[8]加密方案和具有加法同态特性的 Paillier 加密方案[9]，但它们都不具有全同态特性。

2009 年，Gentry 基于理想格问题，首次提出了真正实际可行的全同态加密方案[10]。Gentry 首先设计一种部分同态加密方案，之后利用压缩解密电路来降低解密函数的多项式次数，实现自举特性。在进行同态运算时，通过重加密等技术来抑制密文噪声的增加，实现全同态效果。Dijk[11]等于 2010 年提出了整数域范围内的全同态加密方案 DGHV，该方案基于近似最大公约数问题，将明文的比特加密为一个大整数，实现了加法和乘法的同态运算。2011 年和 2014 年，Brakerski 等[12-13]提出了基于误差学习 LWE 问题和 RLWE 问题的全同态加密方案，采用模交换技术来抑制噪声的增长。文献[13-15]中所提方案为近几年较为成熟的同态加密方案。文献[16]基于 LWE 和 RLWE 问题，借用矩阵的加法、乘法运算实现密文的同态计算，该方法被认为是目前较为自然的方案。

Liu[17]基于近似最大公约数问题提出了一种基于整数的全同态加密方案，其通过复杂代数方程实现全同态加密，具有较高的执行效率。文献[18-19]讨论了现有同态加密方案的可行性以及效率问题，并结合一些应用场景，如医疗信息存储、股票销售，分析了对同态算法的要求。虽然全同态算法效率低下，但是在实际应用场景中，部分同态加密方案已经可以经受实际环境的考验，能胜任现实工作。

目前已经有一些同态方案的应用实例。例如，Li 等[20]基于安全交集协议，为移动社交网络提出了支持隐私保护的个人数据匹配方案 FindU，并使用同态方案增强安全性，为用户提供安全且准确的好友推荐、匹配服务。Rahulamathavan 等[21]基于支持向量机提出了一种数据分类协议，并结合 Paillier 方案加强安全性。文献[22]在云存储环境中针对隐私保护问题，通过结合同态算法，满足了第三方审查且不泄露隐私信息。Liu 等[23-24]设计了一种支持隐私保护的计算框架和工具集，其支持多密钥加密并扩展到了有理数计算，但是由于现实中更多地涉及整数和浮点数运算，因此该方法仍有不足之处。

文献[25-27]将整数域的同态加密方案扩展到了定点数和浮点数部分，延伸了同态算法的使用范围。文献[28]从理论层面详细分析了上述同态加密方案的数学理论基础以及各自的特点，规范了同态方案中的各类名词、相关概念和定义，并运用数学知识对以上概念进行了统一描述。

现有的同态加密方案大多仅支持整数型数据的同态运算，不支持浮点型数据的同态运算，因此无法满足实际应用需求。本文结合云计算环境，提出了一种支持浮点运算的全同态加密算法，其目的是将加密算法从整数扩展到浮点数；同时结合 Spark 框架，设计了并行浮点数全同态加密算法，利用集群优势提高了算法的执行效率，实现了高效的加解密操作，并有效减少了同态操作的时间。理论分析和实验结果显示，并行同态加密算法支持浮点运算，对大量浮点数据能够进行快速高效的加解密操作，具有较好的安全性和实用性，能够很好地适用于云计算场景。

2　原始算法概述

对于现有的全同态加密（FHE）方案，噪声消减是保证全同态加密方案可行性和实用性的重要机制，即一旦密文中的噪声超过了某一界限值，那么该密文将不能被正确解密。由于同态操作，尤其是同态乘法，会大幅增加结果密文中的噪声，因此噪声消减机制必须与同态操作同步进行，以抑制结果密文中的累积噪声，如常用的自举、模变换方法。

文献[17]提出了一种对称的全同态加密方案，其安全性基于近似最大公约数问题，依赖于复杂代数运算，因此效率高于基于理想格问题的同态方案。加密方案主要由密钥生成算法 $KeyGen()$、加密算法 $Enc()$、解密算法 $Dec()$组成。

（1）密钥生成算法 $KeyGen()$

1）设方案中的质数为 q，在伽罗华域 $GF(q)^{n+1}$ 中选取一个随机整数向量并记为 $K=[k_0,\cdots,k_n]$，在 $GF(q)^l$ 中选取随机向量并记为 $\Theta=[\theta_0,\cdots,\theta_{l-1}]$；

2）设 $\Phi=[Enc(K,\theta_0),\cdots,Enc(K,\theta_{l-1}),Enc(K,1)]$；

3）私钥为 K 和 Φ；

4）公共评估密钥为 $PEK=\{p_{i,j}=Enc(K,K_iK_j):0\leqslant i,j\leqslant n\}$。

（2）加密算法 $Enc(K,V)$

给出待加密整数 V，生成随机整数 $r_0,\cdots,r_l\in GF(q)$，满足 $V=r_0\oplus\cdots\oplus r_l$，则整数 V 对应的密文为 $c_v=(r_0\cdot Enc(K,\theta_0)\oplus\cdots\oplus r_{l-1}\cdot Enc(K,\theta_{l-1})\oplus r_l\cdot Enc(K,1))$。

（3）解密算法 $Dec(K,c_v)$

对于密文 c_v，直接计算得出 $V=K\cdot c_v=c_0k_0+\cdots+c_nk_n$。

密文加法：密文 c_v 和 $c_{v'}$ 的加法为普通的向量对应项相加，即 $c_{v+v'}=c_v+c_{v'}=(c_0+c_0'\bmod q,\cdots,c_n+c_n'\bmod q)$。

密文乘法：密文 $c_v=[c_0,\cdots,c_n]$ 和 $c_{v'}=[c_0',\cdots,c_n']$ 的乘法操作被定义为 $c_{vv'}=\sum_{i,j=0}^{n}c_ic_j'p_{ij}$。

该方案使用伽罗华域上复杂的线性代数实现了整数的全同态加密方案，其安全性依赖于数学中的近似最大公约数问题。

3　面向浮点运算的全同态加密方案

原始 FHE 方案只能以整数为运算对象，虽然可以满足

部分日常生活需求，但是无法解决科学计算问题。实际应用场景中，用户多以实数进行日常计算，单纯的整数运算显然无法处理全部的日常事务，特别是医疗、电子商务等领域中的相关计算。在医疗系统中，传感器实时监测患者的心率、血压、血糖值、血清 C 肽浓度等指标，并上传至云服务器进行处理。当进行医疗决策时，用户将会频繁检索云端的健康信息以提供决策支持。决策参数通常包含浮点数，因此支持浮点运算的同态算法可以与云计算场景紧密结合，保护用户隐私不被侵犯。本文在原始 FHE 方案的基础上，对加密方案做出改进：调整参数，重新设置约束条件，并将明文从整数扩展到浮点数，从而实现了支持浮点运算的全同态加密方案。

3.1 加密方案

改进后的 FHE 方案由密钥生成算法 $KeyGen()$、加密算法 $Enc()$、解密算法 $Dec()$ 组成，支持同态加法和同态乘法。本文方案同时支持整数运算和浮点运算，下文均以扩展对象即浮点数为例进行说明。

(1)密钥生成算法 $KeyGen()$

生成两个维度为 n 的加密密钥向量 k 和 s，它们共同构成实数对集合 $K(n)=[(k_1,s_1),\cdots,(k_n,s_n)]$，$n>1$，并且满足浮点数 FHE 方案的约束条件：

$$\begin{cases} k_i\neq 0, & \forall\, 1\leqslant i\leqslant n \\ s_1+\cdots+s_{n-1}\neq 0, & s_n\neq 0 \end{cases} \tag{1}$$

(2)加密算法 $Enc()$

设 V 为待加密的浮点数(也可以是整数)，则加密算法 $Enc()$ 的具体步骤如下：

1)生成随机噪声集合 $P=[(r_1,p_1),\cdots,(r_{n-1},p_{n-1})]$，其中 $r_i,p_i\in R$。

2)计算顺序密文 C'，其中 C' 包含 n 个子密文。子密文的计算公式为：

$$c_i=\begin{cases} k_i*(s_i*V+p_i)+r_i, & 1\leqslant i\leqslant n-1 \\ k_n*s_n*\sum\limits_{i=1}^{n-1}\left(p_i+\dfrac{r_i}{k_i}\right), & i=n \end{cases} \tag{2}$$

3)定义映射函数 f：

$$f(i)=j,\ \forall\, 1\leqslant i,j\leqslant n \tag{3}$$

按照函数 f 的映射结果，将顺序密文 C' 的第 i 个子密文 c_i 映射为乱序密文 C'' 的第 j 个子密文，记为 c_{d_j}。因此，$d_j=i$，其中 j 表示 c_{d_j} 在密文 C'' 中的第 j 个位置。对于 $i\in[1,2,\cdots,n]$，将所有的映射结果 j 的集合定义为 J。

因此，子密文 c_i 和 c_{d_j} 满足：

$$\forall\, c_{d_j}\in C'',\exists\, c_i\in C',c_{d_j}=C''[j]=C'[i]=c_i \tag{4}$$

由式(4)可知，函数 f 维系着密文 C' 和 C'' 各自子密文之间的关系。函数 f 的映射结果是随机的，且不同密文 C' 的映射结果相互独立。因此，不同密文的子密文顺序互不影响，均为随机排列。

最后，使用确定性加密算法(如 AES 加密算法)将数组 J 加密后作为子密文 c_{n+1}。密文 C'' 和子密文 c_{n+1} 即为明文 V 最终的加密结果 $C=[c_{d_1},\cdots,c_{d_n},c_{n+1}]$。

(3)解密算法 $Dec()$

解密算法将密文向量 $C=[c_{d_1},\cdots,c_{d_n},c_{n+1}]$ 解密后，得出明文 V。具体步骤如下：

1)解密子密文 c_{n+1}，得到数组 J，通过式(4)确定子密文 c_i，从而建立 c_i 与密钥元素 k_i 和 s_i 的对应关系。

2)计算 S：$S=\sum\limits_{i=1}^{n-1}s_i$。

3)计算明文值 V：

$$V=\sum_{i=1}^{n-1}\frac{c_i}{k_i*S}-\frac{c_n}{k_n*s_n*S} \tag{5}$$

3.2 同态性证明

给定密钥 $K(n)$，对于任意的浮点数 V，执行加密算法后将得到 n 维密文向量 C'' 和子密文 c_{n+1}。乱序密文 C'' 的维度与密钥 $K(n)$ 中的 n 大小相同。本方案中的安全参数为 n，其决定了密文中子密文的数目，可由用户指定。在同态加法和乘法操作中，默认是密文 C'' 中的 n 个子密文参与计算，除非特别指出，子密文 c_{n+1} 仅作为维系映射关系的密文，不参与子密文的加法和乘法运算。

改进的 FHE 方案中的加密操作和解密操作可以表示为：

$$Enc(K(n),V)=[c_{d_1},\cdots,c_{d_n},c_{n+1}]$$

$$Dec(K(n),[c_{d_1},\cdots,c_{d_n},c_{n+1}])=V$$

3.2.1 同态加法

设对任意两个明文数据 V_1 和 V_2 加密后的密文数据分别为 C_1 和 C_2：

$$\begin{cases} C_1=[c_{1d_{11}},\cdots,c_{1d_{1n}},c_{1(n+1)}]=Enc(K(n),V_1) \\ C_2=[c_{2d_{21}},\cdots,c_{2d_{2n}},c_{2(n+1)}]=Enc(K(n),V_2) \end{cases} \tag{6}$$

密文 C_1 和 C_2 的同态加操作在本方案中被定义为向量加，但由于子密文已经被随机打乱，C_1 和 C_2 对应位置的子密文 $c_{1d_{1i}}$ 和 $c_{2d_{2i}}$ 不是由同一密钥对 k_i 和 s_i 加密而来，因此无法直接相加。

首先解密子密文 $c_{1(n+1)}$，$c_{2(n+1)}$，并通过式(4)访问到子密文 c_{1i} 和 c_{2i}。但是如果将密文 C 还原为最初的 $C'=[c_1,\cdots,c_i,\cdots,c_n]$，并通过密文 C_1' 和 C_2' 对应位置的子密文相加完成同态加法操作，则攻击者可以得知子密文 c_i 与密钥对 k_i 和 s_i 的对应位置关系，从而破解出密钥。本文利用映射函数 f 重新生成一组新的映射关系，记为 J_adj，并以 J_adj 为基准调节 C_1'' 和 C_2'' 的子密文的排列顺序。假设调整后的密文为 $C''_{1\_adj}=[c_{1d_1},\cdots,c_{1d_n}]$ 和 $C''_{2\_adj}=[c_{2d_1},\cdots,c_{2d_n}]$，调整方法为：

$$C''_{adj}[J\_adj[i]]=C''[J[i]] \tag{7}$$

将密文 C''_{1_adj} 和 C''_{2_adj} 的对应项相加，并将映射关系 J_adj 加密作为新的子密文 c_{n+1}，即：

$$C_1\oplus C_2=[c_{1d_1}+c_{2d_1},\cdots,c_{1d_n}+c_{2d_n},c_{n+1}] \tag{8}$$

对同态加法的密文结果进行解密：

$$\begin{aligned} Dec(K(n),C_1\oplus C_2)&=Dec(K(n),C_1)+Dec(K(n),C_2)\\ &=V_1+V_2 \end{aligned} \tag{9}$$

即为明文加法的对应结果。因此，本方案具有加法同态特性。

由同态加法属性可以推知本方案的另一属性：设 $d\in R$，$d\odot C=[d*c_{d_1},\cdots,d*c_{d_n},c_{n+1}]$，对上述密文结果进行解密后可得：$Dec(K(n),d\odot C)=d*Dec(K(n),C)=d*V$。

3.2.2 同态乘法

设任意两个明文数据为 V_1 和 V_2，两个密钥为 $K_1(n_1)$ 和 $K_2(n_2)$，密钥可以相同也可以不同，且密钥的维度 n_1 和 n_2 可以相同也可以不同。假设使用同一密钥 $K(n)$，通过加密算法

$Enc()$对明文进行加密，可以得到密文 C_1 和 C_2。密文 C_1 和 C_2 的乘积为向量的外积，子密文 c_{n+1} 不参与计算，得到 $n*n$ 的密文矩阵。具体表达式如下：

$$C_1 * C_2 = \begin{bmatrix} c_{1d_{11}} * c_{2d_{21}}, \cdots, c_{1d_{11}} * c_{2d_{2n}} \\ \cdots \\ c_{1d_{1n}} * c_{2d_{21}}, \cdots, c_{1d_{1n}} * c_{2d_{2n}} \end{bmatrix} \tag{10}$$

对密文乘积矩阵按照行或者列执行解密操作，此处按列解密，即

$$\begin{aligned} c^*_{2d_{2i}} &= Dec(K(n), [c_{1d_{11}} * c_{2d_{2i}}, \cdots, c_{1d_{1n}} * c_{2d_{2i}}, c_{1(n+1)}]) \\ &= c_{2d_{2i}} * Dec(K(n), [c_{1d_{11}}, \cdots, c_{1d_{1n}}, c_{1(n+1)}]) \\ &= c_{2d_{2i}} * V_1 \end{aligned} \tag{11}$$

最终得到结果密文：

$$\begin{aligned} C^* &= [c^*_{2d_{21}}, \cdots, c^*_{2d_{2n}}, c_{2(n+1)}] \\ &= [V_1 * c_{2d_{21}}, \cdots, V_1 * c_{2d_{2n}}, c_{2(n+1)}] \\ &= V_1 \odot C_2 \end{aligned} \tag{12}$$

密文 C^* 即为密文乘法的结果，其密文维度和子密文顺序与密文 C_2 一样。同理可知，如果按行解密，密文 C^* 的密文维度和子密文顺序与密文 C_1 保持一致。由上式可知，本方案在执行同态乘法操作后，不会造成密文数据膨胀，子密文数量依然维持不变。

综上所述，本方案具有乘法同态特性，且通过观察加法和乘法的密文结果，可知本方案的密文具有紧凑特性。

继续解密密文 C^* 即可得到明文乘积 $V_1 * V_2$。本方案不需要任何的噪声消减机制辅助同态操作来抑制噪声的急剧增长，且所有的同态操作均不改变结果密文的维度。

3.3　安全性分析

本方案中密钥恢复问题的困难模型为近似最大公约数(AGCD)问题。

AGCD 问题由 Howgrave-Graham[29] 提出，具体可以表述为：在给出任意数量的关于整数 p 的近似倍数 $a_i = p * q_i + r_i$（其中 q_i 和 r_i 都是整数，并且 q_i 和 r_i 的值根据 a_i 的不同而不同）的情况下，求出隐藏的公约数 p。

本文方案具有语义安全特性，可以证明其在选择性明文攻击模型下具有密文不可区分性。

在本方案中，攻击者选定明文 V_1，V_2 和经过密钥 $K(n)$ 加密的密文 C，但无法确定密文对应的是 V_1 和 V_2。攻击者能够在多项式时间内区分出 C 是由 V_1 和 V_2 加密得来的概率为 $\frac{1}{2}+\varepsilon$，其中 ε 可以忽略不计。

针对文献[17]中算法密钥泄露的问题，本文在提出的支持浮点运算的同态加密方案中加入了打乱密文顺序的操作，该操作消除了子密文和密钥对的关联性，使得攻击者无法通过选取 n 组子密文求解方程组来破解加密密钥，从而抵挡选择性明文攻击。

将包含 n 个子密文的顺序密文 C' 随机打乱排序，共有 $n!$ 种不同的排列方式。如果选取 n 组子密文破解密钥，则只有当 n 组子密文的排列方式完全一致时才可以求解出正确的解密算法 $Dec()$ 的系数 $\frac{1}{k_i * S}$ 和 $-\frac{1}{k_n * s_n * S}$。由于每一个密文 C' 都有 $n!$ 种不同的排列方式，因此 n 组密文共有 $n!^n$ 种组合方式，其中完全一致的组合共有 $n!$ 种，所以正确求解系数的可能性为 $\frac{n!}{n!^n}$。而求解出的正确系数 $\frac{1}{k_i * S}$ 和 $-\frac{1}{k_n * s_n * S}$ 有 $n!$ 种可能的排列顺序，因此能够求解出正确的解密系数并还原出原始相对顺序的可能性为 $\frac{1}{n!^n}$，时间复杂度为 $O(n!^n)$。因此，可以认为无法在线性时间内破解本方案，无法获取密钥的相关信息。

3.4　优缺点分析

原方案中的加密算法由低阶加密和高阶加密组成，借助复杂的线性代数运算实现，密钥 $K(n)$ 由长度不等的一维向量组合而成，整体方案的运算量大而且逻辑繁琐。本方案简化了加解密算法的实现公式和密钥生成方案，易于实现，方便与实际应用结合使用。

本方案在存储空间方面的需求与原始方案保持一致。算法在加密明文的过程中会生成 n 个子密文，用来保障加密算法的安全性和功能性，且 n 越大，算法的安全性越高。存储空间由明文对应的 1 扩展至密文对应的 n，增长快速，需要占用的存储空间是原始明文的 n 倍。在实际应用中，该方法需要根据实际需求兼顾安全性和存储空间，并在两者之间取得平衡。

本文的全同态加密方案通过打乱子密文顺序增强了算法的安全性，使得攻击者无法在线性时间内破解密钥，保障了密文数据的安全。

在同态加法执行过程中，解密子密文 $c_{1(n+1)}$ 和 $c_{2(n+1)}$ 以 J_adj 为基准进行调整时，数组 J 暴露在内存中，如果攻击者在此期间攻破服务器，可能造成部分隐私信息的泄露。

表 1 对比分析了本文方案与 DGHV 和文献[17]方案的特性。由表 1 可知，3 种全同态加密算法的困难模型均为近似最大公约数问题 AGCD。DGHV 仅支持整数，且密钥尺寸和密文膨胀率远大于文献[17]和本文的浮点同态方案，在运算效率和存储空间方面均处于劣势。本文在文献[17]的基础上精简了密钥生成算法，对密钥尺寸进行压缩，并通过简单代数公式实现了加解密算法，易于实现和应用。

表 1　同态方案的复杂性比较

Table 1　Comparison of complexity between homomorphic schemes

方案	困难模型	密钥尺寸	密文膨胀率
DGHV[11]	AGCD	公钥 $\tilde{O}(\lambda^{13})$，私钥 $\tilde{O}(\lambda^7)$	$\tilde{O}(\lambda^5)$
Liu[17]	AGCD	$\tilde{O}(n^2)$	$\tilde{O}(n)$
浮点同态	AGCD	$\tilde{O}(n)$	$\tilde{O}(n)$

4　并行算法的设计

分析本文提出的加密方案可知，改进后的浮点数 FHE 方案在加密明文数据时会生成一个维度为 $n+1$ 的向量，即包含 n 个子密文和一个映射关系子密文 c_{n+1} 的密文数据 C。其中，前 n 个子密文的计算相互独立，互不影响，可以同时进行。因此，本文结合基于内存计算的并行计算框架 Spark[30]，设计了并行浮点数全同态加密算法来提升加密算法在云环境中的运算性能，从而进一步提升加密方案的执行效率和实用性。

4.1　Spark 架构

Spark 是加州大学伯克利分校的 AMPLab 创立的大数据

处理和计算框架，它立足于内存计算，引入了弹性分布式数据集(Resilient Distributed Datasets，RDD)[31]的概念，允许用户将数据加载至内存进行反复计算或查询，非常适用于大数据和机器学习。Spark 提供了丰富的接口，易于使用，除了支持 Scala，Python，Java 等语言，还可以集成 Hadoop，运行在任意的 Hadoop 数据源上，如 HDFS，Hive，HBase 等。

Spark 的基本运作流程如图 1 所示。应用程序首先在 Driver 节点上初始化 SparkContext，并根据相关配置向 Cluster Manager 申请所需资源，然后在各个 Worker 节点初始化相应的 Executor。SparkContext 启动 DAGScheduler，将提交的作业划分成若干 Stage(TaskSet)，每个 Stage 包含多个 Task。底层调度器 TaskScheduler 和 TaskSetManager 管理接收到的 Stage，并将计算代码和数据资源发送至相应的Executor执行任务 Task，以加快程序的整体运行速度。

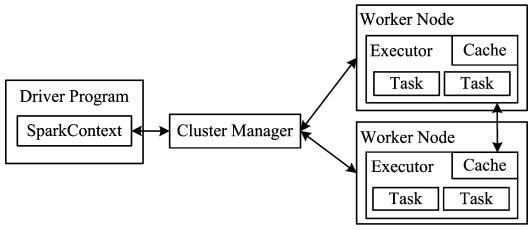


图 1 Spark 集群组件

Fig. 1 Components of spark cluster

本文以 Standalone 模式运行，从 HDFS 读取文件并将其初始化为 RDD。通过控制集群中可用核心数量和分区 Partition 的数量来初始化不同数量的 Task，并将其分发到集群内的所有工作节点(Worker Node)，由本节点的 Task 针对本地分区完成计算，从而达到设置不同并行度的效果。

4.2 基于 Spark 的并行浮点同态加密算法

Spark 并行计算框架仍然采用 Map 和 Reduce 的思想：Map 函数对由划分的数据块转换而来的 RDD 进行相应操作，Reduce 函数负责整合所有 Map 得到的结果。

支持浮点运算的并行同态算法的整体流程如图 2 所示。

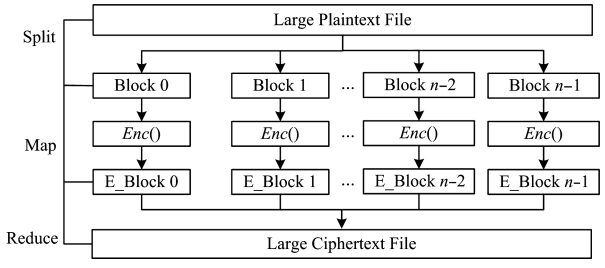


图 2 并行浮点同态方案的加密流程

Fig. 2 Encryption process of parallel floating-point homomorphic scheme

在 Split 阶段，通过设置 Block Size 将明文大文件划分成合适的小文件并上传至 HDFS 分布存储，在运行 Spark 程序时将其转换为 RDD 进行浮点加密运算。图 2 中明文小文件块的数量即为算法中分区 Partition 的数量，将其与集群可用核心数相结合来设定浮点同态算法的并行度。算法的并行执行由 Spark 内核负责调度，内核根据算法的并行度自动将数据集和计算任务分配至工作节点，集群节点同时进行浮点同态运算，最后由 Reduce 函数对各计算结果整合后得到最终结果。

具体的并行浮点数全同态加密算法如算法 1 所示。

算法 1 并行浮点同态算法

输入：明文文件

输出：密文文件

1. 串行生成加密密钥 K(n)：

   K(n)＝KeyGen()

2. 并行计算明文分块 Block 中数据 V 对应的密文 C：

   2.1 FOR i＝1 to n－1

   生成噪声 P

   计算子密文 c_i

   2.2 计算子密文 c_n

   2.3 将子密文打乱，并计算子密文 c_{n+1}

3. 通过 Reduce 函数汇集成密文大文件

5 并行算法的性能分析

5.1 串行性能分析

浮点数 FHE 方案的加密算法可以划分为两个阶段：准备阶段和加密阶段。前者主要进行加密密钥的生成和校验；后者主要进行具体的数据加密操作，是算法性能分析的主体部分。

在改进后的 FHE 方案中，加密算法的操作粒度为浮点数，设明文文件中包含 N 个浮点数。在计算机中，加法运算、移位运算和赋值运算的复杂度近似相等，本文定义一种 X 运算来统一表示上述 3 种运算。

本文将准备阶段定义为算法的计算残余量，表示为 R。准备阶段主要生成密钥 $K(n)$，包括两个 n 维向量的生成，即密钥 k 和 s，同时密钥 k 和 s 均需要满足密钥约束条件。两个 n 维密钥向量的生成涉及 $2n$ 个元素的初始化。由于存在约束条件，因此当生成的密钥不满足约束条件时，需要重新生成密钥。由于重新生成密钥会造成密钥生成算法的重复执行和程序步的不确定，因此可以采用辅助调节措施使得密钥生成一次即可满足约束条件，比如将 k 和 s 的取值范围限定为正数。在添加辅助条件的情况下，准备阶段由固定的 $2n$ 个赋值运算组成。计算残余量 R 可以表示为 $R=2n$。

数据加密阶段主要涉及加法运算、乘法运算和除法运算。生成随机噪声部分涉及 $2(n-1)$次赋值操作。对实数 V 执行加密算法，得到 n 维密文数组，其中前$(n-1)$个子密文对应于 $2(n-1)$次乘法运算和 $2(n-1)$次加法运算，共 $4(n-1)$个 X 运算。子密文 c_n 主要涉及 2 个乘法运算、$(n-1)$个加法运算和$(n-1)$个除法运算，可以表示为 $2(n-1)+2$ 个 X 运算。顺序打乱操作涉及映射函数 f 的 n 次映射和 n 次赋值操作，以及 x 次 X 运算的确定性加密操作。

在浮点数 FHE 方案中，设明文 V 的加密复杂度为 E_0，则 $E_0=2(n-1)+4(n-1)+2(n-1)+2+2n+x=10n-6+x$。

设对包含 N 个浮点数的明文文件执行加密运算的复杂度为 U，则 $U=NE_0+R=N(10n-6+x)+2n$。设 X 运算的时间消耗均为 T_{fc}，则浮点数 FHE 串行加密算法的耗时 T_s 为：

$$T_s = UT_{fc} = [N(10n-6+x)+2n]T_{fc} \tag{13}$$

5.2 并行性能分析

设明文文件中包含 N 个浮点数，这些浮点数被划分成 t 个分区后交由对应数量的 Task 处理，则每个分片包含 m 个浮点数明文，并且 $m=N/t$。设计算机集群中共有 u 个处理器，每个处理器平均处理 w 个数据分区，则 $t=u*w$。

基于 Spark 的并行浮点数全同态加密方案较为简单，主要分为 3 个阶段：前期准备阶段、一次 Transformation 操作对应的阶段 T 和一次 Action 操作对应的阶段 A。阶段 T 和阶段 A 在一个 Stage 中即可完成。准备阶段的复杂度与 4.1 节中讨论的串行算法中的准备阶段一样；阶段 T 包括文件生成 RDD 并划分 RDD 分片的操作，以及 RDD 的转换和一些operator操作，主要为功能数据加密；阶段 A 会触发 Spark 的作业提交，并返回各个节点的 Reduce 作业结果。由于本方案逻辑简单，在 Spark 中划分为一个 Stage，包括准备阶段和阶段 T，A，即可完成任务。首先分析阶段 T 的加速比，然后结合阶段 A 来分析加密算法的整体加速比。

5.2.1 阶段 T 的性能分析

阶段 T 需要加密 m 个明文。设每个阶段 T 的复杂度为 T_0 个 X 运算，则结合之前的分析可知：$T_0=mE_0=m(10n-6+x)$。整个阶段 T 需要加密 w 个分片，设其复杂度为 T_x 个 X 运算，则 $T_x=wT_0=mw(10_n-6+x)$。

阶段 T 中，每个子节点在执行任务的起始阶段和结束阶段都要与主节点进行通信，共有 t 个数据分片，因此至少需要 $2t$ 次通信开销。设通信开销为 $T_{ft}=\xi_1 tT_{fc}$，则阶段 T 的并行加密算法的耗时为 $T_{TP}=(T_x+R)T_{fc}+T_{ft}$。

综上所述，阶段 T 的加速比 S_T 为：

$$\begin{aligned} S_T &= \frac{T_s}{T_{TP}} = \frac{UT_{fc}}{(T_x+R)T_{fc}+T_{ft}} \\ &= \frac{[(10n-6+x)N+2n]T_{fc}}{[(10n-6+x)mw+2n)T_{fc}+\xi_1 tT_{fc}} \\ &= \frac{(10n-6+x)N+2n}{(10n-6+x)N+2nu+\xi_1 tu}u \\ &= [1-\frac{2n(u-1)+\xi_1 tu}{(10n-6+x)N+2nu+\xi_1 tu}]u \end{aligned} \tag{14}$$

在实际使用场景中，处理器个数 u、文件分区数 t 和密文中的子密文个数 n 都远远少于明文文件中浮点数的个数 N，即 $u,t,n \ll N$；并且通信时间可以忽略不计，即 $\xi_1 \to 0$。因此，阶段 T 的加速比 S_T 近似等于 u，接近理想加速比。

5.2.2 并行算法的整体性能分析

阶段 A 的时间消耗主要由对各个节点 Reduce 作业结果的合并排序耗时以及节点之间的通信耗时组成。设此时的通信时间消耗为 $T_{fa}=\xi_2 tT_{fc}$。对于排序算法，最好的时间复杂度为 $s(t)=O(t\log t)$，那么存在正数常量 t_0 和 δ，对于 $t>t_0$，满足 $0 \leqslant s(t) \leqslant \delta t\log t$。设阶段 A 中合并排序算法的复杂度为 A_x 个 X 运算，则 $A_x=\delta t\log t(\delta>0)$。

可以认为阶段 T 和阶段 A 的通信时间消耗近似相等，即 $\xi_1 \approx \xi_2$。

整体加速比为：

$$S_p = \frac{T_s}{T_p} \approx \frac{UT_{fc}}{(T_x+A_x+R)T_{fc}+T_{ft}+T_{fa}} < \frac{UT_{fc}}{(T_x+R)T_{fc}+T_{ft}} = \frac{T_s}{T_{TP}} = S_T \tag{15}$$

实验中，所有数据加密完成后，阶段 A 中的 Reduce 操作耗时比阶段 T 中加密数据时的 Map 操作耗时多，前者至少是后者的 2 倍。由式(15)可知，整体加速比小于阶段 A 的加速比 u，小于处理器核心数的 1/3。

6 实验

6.1 实验平台

Spark 集群硬件平台包括 1 个 Master 节点和 3 个 Slave 节点，同时 Master 节点在程序运行时也承担数据存储和计算任务，且配置方式为所有数据分片在各个节点均有冗余。所有节点的软硬件配置相同：CPU 为 Intel(R) Xeon E3-1225 v3，3.2 GHz/8 M 缓存；内存为 16 GB(2x8 GB) 1333 MHz Dual Ranked RDIM；硬盘为 1 TB 3.5-inch 7.2 k RPM SATA II Hard Drive。

软件平台：操作系统为 Centos 6.6，JDK 版本为 64 位 1.7.0_45，Hadoop 版本为 2.5.2，Spark 版本为 1.4.0，Scala 版本为 2.10.4。

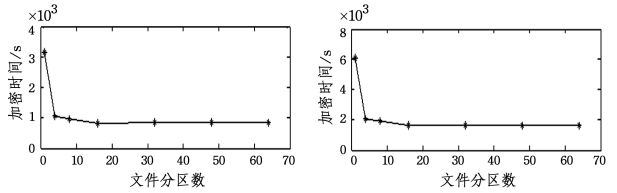
6.2 实验结果分析

对测试数据集分别进行浮点数全同态加密算法的串行和并行测试，通过控制集群可用 CPU 核心数和文件分区数量来设置集群的并行度。实验统计了集群在不同并行度时文件的总加密时间，计算出各自的整体加速比 S_p，结果如表 2、图 3、图 4 所示。

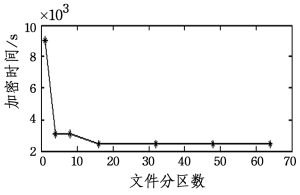
表 2　不同大小文件的测试结果

Table 2　Experimental results for files of different sizes

分片数量	加密时间/s			整体加速比 Sp		
	1 GB	2 GB	3 GB	1 GB	2 GB	3 GB
1	3161.1	6093.1	9037.5	1.0	1.0	1.0
4	1061.8	2066.5	3100.5	3.0	3.0	3.0
8	972.8	1920.1	3085.2	3.3	3.2	3.0
16	815.7	1612.7	2453.4	3.9	3.8	3.7
32	846.8	1637.2	2468.8	3.8	3.8	3.7
48	838.0	1635.5	2468.1	3.8	3.8	3.7
64	843.2	1646.5	2458.3	3.8	3.8	3.7



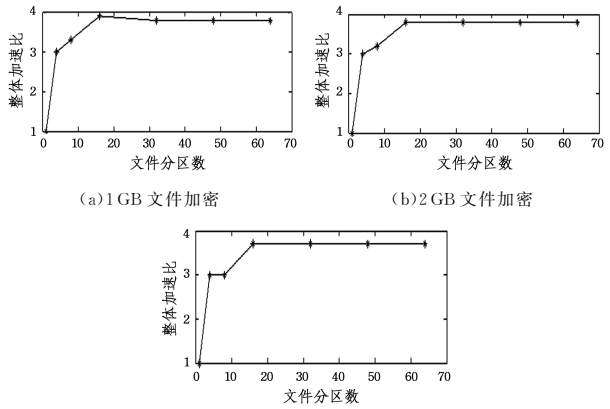
(a)1GB 文件加密　　(b)2GB 文件加密



(c)3GB 文件加密

图 3　文件分区数和加密时间

Fig. 3　File partition number and encryption time



(a)1 GB 文件加密 (b)2 GB 文件加密

(c)3 GB 文件加密

图 4 文件分区数和整体加速比

Fig. 4 File partition number and overall speedup ratio

观察表 2 和图 3、图 4 可知:1)随着集群可用核心数和文件分区数的增长,文件加密时间整体呈现下降趋势。前期随着可用核心数的增长,文件加密时间显著下降,算法性能显著提升,集群性能得到有效发挥;当集群的所有核心均用于计算时,文件分区数的增长对集群性能的影响较小,文件加密时间趋于稳定。2)随着文件分区数超过集群核心数 16,文件分区数量的逐渐增大可以消除集群机器之间的计算能力差异,系统可以将更多的计算负荷分配给计算能力更强大的机器,从而减少文件总的加密时间。但一味地增加分区数量,会导致通信和调度时间大幅增加,因此要根据具体算法制定优化方案。3)由于阶段 A 中 Reduce 操作耗时较多,因此实验中算法的整体加速比最大可以达到 3.9,小于集群核心数 u 的 1/3,即小于 16/3≈5.3,与前文理论分析的结论一致。4)浮点同态算法可以有效支持整数和浮点同态运算,扩展了算法的适用范围,可以为云计算场景中的很多商业实例提供安全、高效的加密服务,保障云端用户隐私数据的安全性和私密性。

结束语 本文提出了一种支持浮点运算的全同态加密方案,使用乱序密文操作增强了算法的安全性。基于 Spark 平台,利用其内存计算模型,设计并实现了并行化浮点数全同态加密算法。该算法在加密过程中将文件分割成不同数量的数据块,通过指定计算时的可用核心数和分区数来控制算法的并行度。本文通过实验验证了该并行算法在 Spark 集群中处理大数据文件时具有良好的加速比,相比于原始的线性加密算法,处理效率显著提升,整体加速比最高可以达到 3.9,所提算法提高了同态加密算法的执行效率,可以有效应用于云计算环境下的隐私保护等场景。

参 考 文 献

[1] FENG D G,ZHANG M,ZHANG Y,et al. Study on cloud computing security[J]. Journal of Software,2011,22(1):71-83. (in Chinese)
冯登国,张敏,张妍,等. 云计算安全研究[J]. 软件学报,2011,22(1):71-83.

[2] POPA R A,REDFIELD C,ZELDOVICH N,et al. CryptDB: protecting confidentiality with encrypted query processing[C]// Proc. of the 23rd ACM Symp. on Operating Systems Principles. New York:ACM,2011:85-100.

[3] FELDMAN A J,ZELLER W P,FREEDMAN M J,et al. SPORC:Group Collaboration using Untrusted Cloud Resources [C]// Proc. of the 9th USENIX Symp. on Operating Systems Design and Implementation. Berkeley,CA:USENIX,2010:337-350.

[4] LI J,KROHN M N,MAZIÉRES D,et al. Secure Untrusted Data Repository(SUNDR)[C]// Proc. of the 6th Symp. on Operating System Design and Implementation. Berkeley, CA: USENIX, 2004:121-136.

[5] MAHAJAN P,SETTY S,LEE S,et al. Depot:cloud storage with minimal trust[C]// Proc of the 9th USENIX Symp on Operating Systems Design and Implementation. Berkeley,CA:USENIX,2010:307-322.

[6] MICCIANCIO D. A first glimpse of cryptography's Holy Grail [J]. Communications of the ACM,2010,53(3):96-96.

[7] RIVEST R L,ADLEMAN L,DERTOUZOS M L. On data banks and privacy homomorphisms[J]. Foundations of Secure Computation,1978,4(11):169-180.

[8] ELGAMAL T. A public key cryptosystem and a signature scheme based on discrete logarithms[J]. IEEE Transactions on Information Theory,1985,31(4):469-472.

[9] PAILLIER P. Public-key cryptosystems based on composite degree residuosity classes[C]// Proc. of IntConf on the Theory and Applications of Cryptographic Techniques. Berlin:Springer, 1999:223-238.

[10] GENTRY C. A fully homomorphic encryption scheme[D]. Stanford:Stanford University,2009.

[11] DIJK M V,GENTRY C,HALEVI S,et al. Fully homomorphic encryption over the integers[M]// Advances in Cryptology—EUROCRYPT 2010. Berlin:Springer,2010:24-43.

[12] BRAKERSKI Z,VAIKUNTANATHAN V. Fully homomorphic encryption from ring-LWE and security for key dependent messages[C]// Cryptology Conference. Berlin:Springer,2011:505-524.

[13] BRAKERSKI Z,VAIKUNTANATHAN V. Efficient fully homomorphic encryption from(standard) LWE[J]. SIAM Journal on Computing,2014,43(2):831-871.

[14] BRAKERSKI Z,GENTRY C,VAIKUNTANATHAN V.(Leveled) fully homomorphic encryption without bootstrapping[C]// Innovations in Theoretical Computer Science Conference. New York:ACM,2012:309-325.

[15] KIM J,LEE M S,YUN A,et al. CRT-based Fully Homomorphic Encryption over the Integers [EB/OL]. [2016-12-15]. http://eprint.iacr.org/2013/057.pdf.

[16] GENTRY C,SAHAI A,WATERS B. Homomorphic encryption from learning with errors:Conceptually-simpler,asymptotically-faster,attribute-based[M]// Advances in Cryptology—CRYPTO 2013. Berlin:Springer,2013:75-92.

(下转第 130 页)