**中图分类号：TP311.5**

**论文编号：10006ZY1506221**



硕 士 学 位 论 文

**恶意域名检测技术研究**

作者姓名 王文博

学科专业 计算机技术

指导教师 兰雨晴 周渊

培养院系 计算机学院

**Research on malicious domain name detection technology**

A Dissertation Submitted for the Degree of Master

**Candidate：Wenbo Wang**

**Supervisor：Yuqing Lan**

School of Computer Science and Engineering

Beihang University, Beijing, China

**中图分类号：TP311.5**

**论文编号：10006ZY1506221**

硕 士 学 位 论 文

**恶意域名检测技术研究**

作者姓名 王文博 申请学位级别 工程硕士

指导教师姓名 兰雨晴 职 称 教授

学科专业 计算机技术 研究方向 网络空间安全

学习时间自 2015 年 09 月 01 日 起至 2018 年 03 月 04 日止

论文提交日期 2018 年 03 月 07 日 论文答辩日期 2018 年 03 月 04 日

学位授予单位 北京航空航天大学 学位授予日期 2018 年 月 日

关于学位论文的独创性声明

本人郑重声明：所呈交的论文是本人在指导教师指导下独立进行研究工作所取得的成果，论文中有关资料和数据是实事求是的。尽我所知，除文中已经加以标注和致谢外，本论文不包含其他人已经发表或撰写的研究成果，也不包含本人或他人为获得北京航空航天大学或其它教育机构的学位或学历证书而使用过的材料。与我一同工作的同志对研究所做的任何贡献均已在论文中作出了明确的说明。

若有不实之处，本人愿意承担相关法律责任。

学位论文作者签名：        日期： 年 月 日

学位论文使用授权书

本人完全同意北京航空航天大学有权使用本学位论文（包括但不限于其印刷版和电子版），使用方式包括但不限于：保留学位论文，按规定向国家有关部门（机构）送交学位论文，以学术交流为目的赠送和交换学位论文，允许学位论文被查阅、借阅和复印，将学位论文的全部或部分内容编入有关数据库进行检索，采用影印、缩印或其他复制手段保存学位论文。

保密学位论文在解密后的使用授权同上。

学位论文作者签名： 日期： 年 月 日

指导教师签名： 日期： 年 月 日

# 摘 要

这是摘要。

关键字：a；b；b；d

# **Abstract**

This is abstract.

**Key words:** a; b; b; d

目 录

[第一章 绪论 1](#_Toc476387908)

[1.1 研究背景 1](#_Toc476387909)

[1.1.1 快速发展的中小银行及其IT建设 1](#_Toc476387910)

[1.1.2 云计算及金融云 1](#_Toc476387911)

[1.1.3 同态加密 3](#_Toc476387912)

[1.2 问题的提出 3](#_Toc476387913)

[1.3 论文的主要内容 5](#_Toc476387914)

[1.4 论文的组织结构 6](#_Toc476387915)

[第二章 国内外研究现状分析 7](#_Toc476387916)

[2.1 同态加密基本技术研究现状 7](#_Toc476387917)

[2.1.1基于理想格的全同态加密方案 7](#_Toc476387918)

[2.1.2基于整数的全同态加密方案 10](#_Toc476387919)

[2.1.3基于容错学习问题的同态加密方案 11](#_Toc476387920)

[2.1.4无噪声的同态加密方案 12](#_Toc476387921)

[2.2 同态加密应用研究现状 13](#_Toc476387922)

[2.3 本章小结 16](#_Toc476387923)

[第三章 银行云计算中同态加密算法应用研究 17](#_Toc476387924)

[3.1 银行数据同态计算需求 17](#_Toc476387925)

[3.2 DGHV方案介绍 19](#_Toc476387926)

[3.2.1 部分同态加密方案 19](#_Toc476387927)

[3.2.2 重加密思想及实现思路 21](#_Toc476387928)

[3.2.3 全同态加密方案 24](#_Toc476387929)

[3.3 改进的单比特明文加密方案 25](#_Toc476387930)

[3.3.1 somewhat方案 25](#_Toc476387931)

[3.3.2 全同态方案 27](#_Toc476387932)

[3.3.3 基于二进制进位加法的重加密实现方法 29](#_Toc476387933)

[3.4 定精度实数同态算术运算方案 35](#_Toc476387934)

[3.4.1 数据标准化及加密 35](#_Toc476387935)

[3.4.2 基于二进制的实数同态运算规则 36](#_Toc476387936)

[3.5 本章小结 40](#_Toc476387937)

[第四章 银行云计算中同态加密应用原型系统设计与实现 41](#_Toc476387938)

[4.1 需求分析 41](#_Toc476387939)

[4.1.1 银行云计算中同态加密应用模型 41](#_Toc476387940)

[4.1.2 原型程序功能模块 43](#_Toc476387941)

[4.2 总体设计 44](#_Toc476387942)

[4.3 功能实现 45](#_Toc476387943)

[4.3.1 初始化参数 45](#_Toc476387944)

[4.3.2 同态加密 45](#_Toc476387945)

[4.3.3 同态计算 47](#_Toc476387946)

[4.4 本章小结 47](#_Toc476387947)

[第五章 实验结果与分析 48](#_Toc476387948)

[5.1 总体情况 48](#_Toc476387949)

[5.2参数对方案的影响分析 48](#_Toc476387950)

[5.2.1 安全参数的影响 48](#_Toc476387951)

[5.2.2 阶码的影响 49](#_Toc476387952)

[5.3 本章小结 54](#_Toc476387953)

[总结与展望 55](#_Toc476387954)

[研究工作总结 55](#_Toc476387955)

[未来工作展望 55](#_Toc476387956)

[参考文献 57](#_Toc476387957)

[攻读硕士学位期间取得的学术成果 61](#_Toc476387958)

[致谢 62](#_Toc476387959)

图 目

[图1 银行云计算平台示意图 4](#_Toc475557116)

[图2 增强型电路及密文刷新示意图 8](#_Toc475557117)

[图3 重加密实现原理示意图 21](#_Toc475557118)

[图4 多位二进制乘法示意图 22](#_Toc475557119)

[图5 “3-for-2 trick”原理示意图 23](#_Toc475557120)

[图6 多位二进制数加法示意图 23](#_Toc475557121)

[图7 ai矩阵及其列向量海明重量示意图 25](#_Toc475557122)

[图8 ai矩阵末尾列本位比特及进位比特示意图 31](#_Toc475557123)

[图9 基于海明重量多项式的二进制进位加法示意图 32](#_Toc475557124)

[图10 多比特密文加法运算规则示意图 36](#_Toc475557125)

[图11 多比特密文减法规则示意图 38](#_Toc475557126)

[图12 多比特密文乘法规则示意图 38](#_Toc475557127)

[图13 同态加密一般应用模型 41](#_Toc475557128)

[图14 银行云计算中同态加密应用模型 42](#_Toc475557129)

[图15 原型程序功能模块示意图 43](#_Toc475557130)

[图16 数据拥有方程序流程图 45](#_Toc475557131)

[图17 数据计算方程序流程图 45](#_Toc475557132)

[图18 安全参数对密钥生成时间的影响 49](#_Toc475557133)

[图19 阶码k对单一数据加解密误差的影响 50](#_Toc475557134)

[图20 阶码k对两数同态加结果误差的影响 50](#_Toc475557135)

[图21 对称阶码时不同k值对同态乘结果误差的影响 51](#_Toc475557136)

[图22 固定k1时k2对同态乘结果误差的影响趋势 52](#_Toc475557137)

[图23 固定k2时k1对同态乘结果误差的影响趋势 53](#_Toc475557138)

表 目

[表1 银行常见数据列表 17](#_Toc475386069)

[表2 本文SHE方案与DGHV方案的对比 29](#_Toc475386070)

[表3 本文FHE方案与DGHV方案的对比 29](#_Toc475386071)

[表4 临近取整真值表 30](#_Toc475386072)

[表5 模拟账户表结构 44](#_Toc475386073)

[表6 实验环境 48](#_Toc475386074)

[表7 不同安全参数下的密钥生成时间 49](#_Toc475386075)

[表8 对称阶码时不同k值对同态乘结果误差的影响 52](#_Toc475386076)

[表9 固定k1时k2对同态乘结果误差的影响 52](#_Toc475386077)

[表10 固定k2时k1对同态乘结果误差的影响 53](#_Toc475386078)

# 第一章 绪论

## 1.1 研究背景

### 1.1.1 恶意域名数量巨大

IP地址是由IP协议提供的数字型统一地址标识，作为一种逻辑地址来定义一台设备在网络之中的位置，网络设备逐渐增多IP地址的记忆困难显现出来，保罗·莫卡派乔斯 (Paul Mockapetris) 在1983年的第882和在南加州大学里资讯科学研究院所提出的883号因特网标准草案中提出DNS的架构，提议将其改进为分布式和动态的数据库域名系统，也就是我们今天所用的域名系统的雏形。从1985年Symbolics公司注册的第一个.com域名到如今仅中国域名总数增长至3698万个[1]，域名产业飞速发展，随之而来的安全问题也越来越多的暴露出来。高级持续性威胁常态化，移动互联网黑色产业链已经成熟，巨大的利益促使大量人进行相关活动。以僵尸网络控制端（通常使用域名来进行联系）为例，2012年木马和僵尸网络控制端数量高达36万余个，随着检测技术和安全意识的提高，至2014年数量有明显降低，仍有10万个左右的僵尸网络控制端，并且该数量稳定至今[3]。

### 1.1.2 恶意域名危害增大

我们将当前存在恶意行为或者被恶意使用的域名都视为恶意域名，这其中就包含了DGA、DNS放大攻击、钓鱼域名等等。正如1.1.1节中所述，尽管随着互联网安全监管的加强，恶意域名数量仍然庞大，并且恶意域名相关的技术在与检测技术的对抗之中不断进步，造成的安全威胁更加巨大。2017年活跃的Wannacry、

## 1.2 问题的提出

快速增长的中小银行尤其是小型银行受限于自身能力资源，必须依托金融云平台建设适应自身业务发展需求的IT系统，也那就意味着其所有用户数据都将放在云端存储和处理，并通过网络进行传输，数据安全是其必须首要考虑的问题。银行业因其事关用户切身利益的特殊性，对数据安全性、私密性的要求远高于其它行业。按照我国金融法律明确要求，对客户数据及信息金融企业应承担全部保密责任，同时《商业银行信息科技风险管理指引》也提出，金融机构需要承担在服务外包的过程中的风险管理责任。在传统的银行自建运营的系统中，所有数据由银行自身产生、存储、计算和管理，物理环境相对封闭，接触人员少，风险较小，不易出现泄露、窃取等情况，而基于云计算平台的银行系统，由于不再有物理上的隔离，通过一定的手段，相当数量的互联网用户均存在访问到系统数据的可能性，且系统运维管理人员相对行内来说，不完全可信可靠，因此存在较大风险。在云平台遭受攻击，或者存在管理漏洞、人为破坏安全管理制度等情况下，若银行信息数据发生丢失、窃取甚至被篡改等现象，对于用户而言是致命的。如何保证云计算系统中数据不被随意盗取、查看、拷贝和篡改是云服务商面临的重大挑战。云服务商除具备相应安全管理资质、建立安全管理制度之外，还应采取相应技术手段，从根本上杜绝银行所属数据的安全和私密，提高银行对云计算平台的信任等级。



图1 银行云计算平台示意图

采用数据加密技术是应对银行云计算中数据安全问题的重要途径。通常所称的数据加密技术，是指在加密密钥和加密函数的作用下，将一个明文信息转换无规则的密文形式，从而使得敌对方即使截获密文也无法获取有效信息，而接授权收方则可将此密文经解密函数和解密密钥还原成初始信息。

以字段值在系统中的变动情况为依据，银行数据大体可分为两类：静态数据和动态数据。所谓静态数据，是指某字段数据在其数据记录产生直至消失的整个生命周期里不再改变，如银行代码、定期账号等；动态数据，则是指某字段数据需要随各种操作发生相应变化。根据数据记录变化规律，动态数据又可分为与原记录无关的替换型数据和与原记录相关的运算型数据。替换型数据主要包括账户密码、联系方式、各类日期等，运算型数据主要是各类交易金额、利息、余额等。以上各种类别数据中，静态数据及替换型数据由于在密文状态下只有传输、存储、校验等操作，可根据安全需要选择以明文形式存在，或者用对称加密算法（DES、SM4等）、非对称加密算法（RSA、SM2等）、摘要算法（SHA1、SM3等）之类传统算法加密，而对于运算型数据，由于其涉及用户最核心的利益，是银行的关键业务数据，安全需求高，需要采用加密方式处理，但是现有的传统加密算法在密文状态下无法处理数据，需解密为明文再计算，基于传统加密技术处理此类数据可通过两条途径：一是解密操作和数据运算由银行自身完成，此时虽然风险可控，但是与银行自有系统相比较，徒然增加计算量，云计算就失去了意义；二是解密操作与数据运算由云计算平台完成，那么计算方必然掌握了密钥，加密也就失去了意义，由于计算方的非可信，风险将显著增大。对此类数据，需要采用一种允许数据在密文状态下进行计算的加密算法，加密数据运算全部交由计算方完成，这样能满足银行数据安全需求，也能充分利用云平台的计算能力。同态加密算法的性质决定了其完全满足此应用场景。

自1978年提出以来[1]，同态加密算法的研究从未间断过，先后产生了各种同态加密构造方案，这些方案基于不同的数学模型构建，在明文空间、算法复杂度、同态运算次数以及安全性上各有不同。由于同态加密算法研究多数偏重于理论模型的构造，在实用性方面涉及较少，所以现有的各种同态加密方案不能直接适用于现实场景，对于银行等各专业领域，需要依据现有理论构造更为合适的模型，研究相应的解决方案。

综上所述，银行业云计算中应用同态加密有如下几个问题需要解决：

1. 银行系统中需要同态运算的数据特点及加密需求；
2. 采用哪一类同态加密算法；
3. 如何根据银行数据特点对同态加密算法进行适应性修改和拓展。

## 1.3 论文的主要内容

本研究针对银行业云计算中应用同态加密的上述问题，首先探讨了银行云计算场景下同态加密应用模型，然后针对银行系统数据特点和同态加密算法研究现状，选择一种适应银行云计算场景的同态加密基础构造理论，最后，结合二进制运算法则，提出了密文状态下多位二进制算术运算方法，给出一种固定精度的定点数同态加密算术运算方案。实验证明，本文提出的同态加密算术运算方案能正确给出一定范围内固定精度定点数带密运算结果，满足银行系统运算型数据加密需求。具体而言，论文的主要内容包括四个方面，分别是：

1. 单比特二进制数上的同态加密算法及安全性分析；
2. 带固定精度小数的多位数上的同态加密运算方案，提出一种基于对称多项式的二进制密文向量海明重量计算方法；
3. 同态加密在银行云计算场景中的应用模型；
4. 定精度实数同态加密运算方案的代码实现与验证。

## 1.4 论文的组织结构

第一章：绪论，简单介绍了本文的研究背景和相关概念，提出了本文研究的问题。

第二章：同态加密领域的国内外研究现状分析，对现有同态加密方案进行分类，分别简述其特色和优缺点，并对同态加密应用情况进行介绍。

第三章：银行云计算环境中同态加密运算的应用方案，通过对经典方案的分析，结合银行系统数据实际特点，修改并拓展了DGHV方案，阐述了带小数的多位数同态运算方案的结构和实现方法，构造出使用银行云计算场景的固定精度实数同态加密算术运算方案。

第四章：银行云计算中应用同态加密技术的模型及相应原型程序的设计与实现

第五章：对方案进行了实验验证和结果分析。

总结与展望：对本文进行总结，并对下一步的研究工作进行了展望。

# 第二章 国内外研究现状分析

本章主要对国内外学者在同态加密技术以及应用两个方面的研究现状进行介绍和分析。在基础技术上，同态加密技术大致可分为四类：基于理想格的全同态加密方案、基于整数的全同态加密方案、基于容错学习的同态加密方案和无噪声的同态加密方案。同态加密的应用则主要有数字水印、隐私信息检索、电子投票等。

## 2.1 同态加密基本技术研究现状

同态加密技术在提出之后的三十年间，没有得到有效突破，直到2009年，Gentry第一次设计了全同态加密(FHE，Fully Homomorphic Encryption)的框架，并基于理想格构造了第一个理论上的FHE方案[2],[3]，同态加密技术才得以突破，从而进入新的发展阶段。在Gentry之后，研究人员也陆续的提出了一些改进技术和优化方案。同态加密方案的构造是同态加密技术发展及应用的基础，是当前同态加密技术研究的主题，国内外研究人员分别构造了多种全同态加密方案，其代数结构或计算困难问题各有不同。当前的全同态加密方案按照构造基础可分为三类：基于理想格、基于整数、基于环上容错学习的全同态加密方案，另外近年出现了一类不带噪声消减的全同态加密方案。

### 2.1.1基于理想格的全同态加密方案

2009年，IBM研究员Gentry基于理想陪集问题第一次提出了一个FHE方案[3]，由于计算效率过低，该方案并不实用，但它是第一个从理论上实现全同态加密的方案，实现了从无到有质的飞跃。

按照Gentry构想，首先构造一个基础方案，该方案可满足有限次密文的同态乘法和同态加法运算，具备这一同态属性的加密方案被称为部分同态加密(SWHE，Somewhat Homomoohic Encryption)方案。为保证部分同态加密的安全性，Gentry运用了概率加密思想，引入了“噪声”这一概念，可抵抗选择明文攻击，但同时，噪声也会随着同态计算的次数增加而增大。假设有多个密文，其初始噪声上界均为，进行次同态加操作之后，密文噪声的上界增加为，而进行次同态乘操作之后，密文噪声的上界将增大为。部分同态加密方案能正确对的前提条件是密文噪声必须小于某个阈值（该阈值由SWHE方案参数确定），若噪声超过阈值，则将导致解密错误，同态加密将失效，那么这里噪声的存在及其增长成了部分同态加密方案同态运算能力的限制，需要考虑噪声抑制，为此Gentry提出了重加密技术，如图2所示，每次同态运算前对密文进行重加密以减小输入密文的噪声，以期无限增加同态运算次数。密文重加密使得密文的噪声始终被保持在较低的水平，满足下一次同态计算的要求。Gentry将图2中包括了重加密及同态加/乘运算的电路称为增强型电路（Augmented Decryption Circuit）。如果初始设计构建的部分同态加密方案能够“同态”的运行其增强型电路，那么通过组合增强型电路就能实现任意的密文计算电路，或者说任意多项式函数都可以被同态计算，换言之，实现了真正的全同态计算。

增强型电路

新鲜密文

初始密文

同态加/

乘运算

重加密

重加密

新鲜密文

初始密文

重加密

刷新密文，降低噪声至新鲜密文状态

同态加/

乘运算

同态加/

乘运算

新鲜密文

初始密文

重加密

重加密

重加密

新鲜密文

初始密文

图2 增强型电路及密文刷新示意图

由上可知，Gentry方案实现全同态加密的关键在于其SWHE方案能否同态的运算自身的增强型电路，也即该方案是否具有自举（Bootstrappable，部分文献称为自启动）能力。若SWHE方案具备了自举能力，则称其为自举同态加密(BHE, Bootstrappable Homomorphic Encryption)方案，根据Gentry的自举定理，所有的BHE方案都可转化为FHE方案。但是，Gentry构建的SWHE方案不具备天然的自举能力，不能正确运行其增强型电路，关键原因在于其增强型电路主要组成部分也即解密电路的复杂度超过了SWHE方案的同态计算能力。为了解决这一问题实现自举，Gentry提出了压缩解密电路(Squashing the Decrypt Circuit)的方法以降低解密复杂度，从而将SWHE方案转化为BHE方案。

继Gentry提出基于重加密的全同态加密框架和方案之后，一些以之为基础架构的同态加密研究成果陆续诞生。Smart和Vercauteren[4]首次在实验的角度对Gentry方案做了研究，依然采用主理想格为其方案代数结构，且其格的判别式必须为素数。在参数选取时，要求格的维度不小于，过大的参数使得Gentry方案的仿真失败。另外当格的维度时，格判别式的素数要求也导致密钥生成失败。基于这一实验结果，Gentry和Halevi[5]进行了方案优化，去除了格判别式的素数限制，完成了FHE方案的仿真。该优化方案给出了不同维度的格对应的参数值，所生成密钥在格的维度最大时大小为2.3G，密钥生成算法需要运行2.24小时，加密时间3分钟，密文刷新也即重加密需时30分钟，解密时间相对很短，可忽略不计。Stehle和Steinfeld[6] 在其文章中对文献[3]中方案进行了两项改进，其一是对其中与稀疏子集和问题相关的困难问题假设的改进分析，其二是引入了概率解密算法（Probabilistic decryption algorithm），其可由低阶乘法代数运算电路实现。该方案要比Gentry方案效率高，单个初等加法/乘法门电路的比特复杂度降低到。

Smart和Vercauteren指出，Gentry的方案支持单指令多数据流(SIMD, Single Instruction Multiple Data) [7]，并提出一个能对密文进行SIMD运算的FHE方案[8]，通过选择适当的参数，将SIMD应用于重加密过程的执行，将有效提高方案运行速度，使方案运行时间降为Gentry方案的1/2.4，同时密文规模也缩减为Gentry方案的1/72。

在Gentry的设计框架下，解密函数需要计算一个整数模剩余，而这一计算的复杂度将高于SWHE方案同态运算能力，从而导致方案无法自举。为此，Gentry等在文献[9]提出，若该整数模数选取为一个的近似值，则可以绕过同态模剩余计算，降低解密复杂度，同时对私钥加密后可得到单一密文而非密文序列，从而避开稀疏子集和困难问题(SSSP，Sparse Subset Sum Problem)假设，降低公钥尺寸。该方案还可结合文献[8]中的SIMD同态计算技术以提高速度。

这类以理想格为基础架构的同态加密方案普遍存在以下三个缺陷：一是理想格这一代数结构过于复杂，对于大多数研究人员和使用这来说非常难以理解和分析；二是方案的可证明安全性都将规约到理想格困难问题上，但当前这一困难问题的复杂度还未得到可靠证明；三是此类方案由于密钥生成及重加密算法过于复杂，因此其算法效率非常低下，这也极大的限制其应用。

### 2.1.2基于整数的全同态加密方案

Dijk和Gentry等人在FHE架构基础做了有效突破，于2010年欧密会上提出了一个基于整数的FHE方案(DGHV方案)[10]，所采用的代数结构由理想格变为了整数环，其构造思路与Gentry方案基本一致。首先，设计出一个可支持有限次同态加和同态乘计算的SWHE方案，随后基于“压缩解密电路”技术实现同态解密从而使方案自举，最后利用自举转换定理得到一个FHE方案。出于安全考虑，方案的密钥规模设定为。对比基于整数和基于理想格两种FHE方案，前者构造基础所用的代数结构即整数环较为简单，使得该类方案更容易理解。DGHV方案开创了全同态加密研究的新领域，其后很多学者提出了基于该方案的优化方案。

Coron等人基于DGHV方案提出了一个改进的密钥生成算法[11]，将密钥规模从降低到，以解决DGHV方案中密钥规模过大的缺点。基本思路是：将公钥由线性形式改为二次形式，即在初始的密钥生成过程中仅生成公钥的一个子集，然后通过乘法组合子集元素以即时的生成完整公钥，这种处理方式仅是将密钥生成过程中的复杂运算转移到了加密计算中，解决了公钥存储问题，但并未降低整个方案的算法复杂度。对于一个较大的安全参数，该方案加密和重加密需时分别为3分钟和14分钟，密钥规模约为800M，解密时间可忽略不计。

上述DGHV方案及其改进方案都是单比特明文方案，也就是说此类方案只能将明文信息拆分成比特进行加密，且1比特明文可生成数百兆乃至上G的密文，若明文为多比特信息，则得到密文序列规模将非常可观，一般认为比特明文方案都会存在密文膨胀率过大的问题，在实际应用中密文需要通过网络传输，过大的密文将严重占用网络资源。为降低密文膨胀率这一指标，Cheon等人提出了的整数上批处理FHE方案[12]。这种方案可将多个比特的明文同时加密生成一个密文，同时还保持其同态性，方案的可证安全性规约为稀疏子集和问题。方案可将比特明文加密生成一个密文，由此，可将密文膨胀率由DGHV方案的降低到。利用中国剩余定理(CRT，China Remainder Theorem)的环同态性质， Kim等人提出了一个改进的整数同态加密方案[13]，可将明文空间由单比特扩展为，其中的。这一改进使得SWHE方案的计算负荷由降低至，FHE方案的计算负荷从降低至。基于CRT，Nuida等人提出了一个新的批处理FHE方案[14]，将明文空间扩展到上，并采用批处理技术降低密文膨胀率。

### 2.1.3基于容错学习问题的同态加密方案

容错学习 (LWE，Learning with Error)问题[15,16]由Oded Regev于2005年提出，作为一个困难假设被广泛用于公钥加密体系中。

Brakerski等基于LWE提出了构造全同态方案的新思路[17]，初步打破了Gentry于2009年提出的构造“蓝图”。首先构造一个可同态计算深度为D的算术电路的层次型同态加密(Leveled Fully Homomorphic Encryption)方案。该层次型同态加密方案在每次同态运算之后都需要使用重线性化(Relinearization)技术压缩结果密文的长度，并使用模转换(Modulus Switching)技术降低噪声。其次，使用重线性化和解密电路压缩技术改进方案，使方案可自举。最后，与Gentry方案类似的，使用自举定理即可将层次型同态加密方案转换为FHE方案。该方案第一次提出了一个无需同态解密即可构造层次型全同态加密方案的思路，其可证安全性规约为Ring—LWE问题，同时引入自举技术优化方案，将单门电路计算复杂度降低为准二次多项式，但若进一步提高其安全性至LWE问题，需要引入高斯分布的噪声，则其算法复杂度会有所提高。Gentry等人在文献[22]中提出，在Ring—LWE假设下，采用批处理技术可使单门电路同态计算复杂度由降低到，即算法复杂度只与安全参数有关，而与电路深度无关，从而也使得同态计算次数摆脱电路深度的限制。Brakerski和Vaikuntanathan基于这一思路，使用重线性化技术和消息依赖密钥(KDM，Key-dependent message)安全[19,20]假设构造出一个模数不变的FHE方案[18]，不再采用模数转换技术，提高了算法效率。该方案通过加密私钥的多项式函数实现私钥传递和重加密，解决了方案循环安全假设的限制，同时通过对秘钥向量在形式上做替换，即可达到压缩解密电路的目的，这将有效降低FHE方案的密钥规模。

Brakerski等人首次基于LWE问题提出了一个不需要解密电路压缩技术FHE方案[21]。首先构造一个基于重线性化技术的SWHE方案，该方案以LWE假设为基础，避免了理想格的复杂度问题和RLWE的安全性问题；其次利用密钥转换和模转换技术降低密文的长度和解密复杂度。方案中，解密是通过计算密文与密钥的内积实现，即，在保持解密结果不变的前提下，密钥转换技术可以把由密钥解密的密文转换为新密钥解密的密文，模转换可以将模值下的密文转换为另一个不同的模值下的密文，秘钥转换可降低密文维度，模转换可使密文噪声减小。重复此过程即可控制密文维度和噪声，继续参与运算，实现不需要压缩解密电路的同态运算。

### 2.1.4无噪声的同态加密方案

前述三类方案在算法效率上存在限制，主要原因在于方案产生密文的膨胀率过大，同时其噪声随着同态运算次数急速增长，需要对密文噪声进行控制，而密文刷新技术复杂度较高，因此部分学者试图研究构造一类无需噪声处理的方案。此类方案基本上以非交换环这类数学模型作为构建方案的基础，其共同特征是试图构造一种无需考虑噪声消减即可实现全同态加密的方案，以期不再受Gentry蓝图的限制，因为该类方案无需考虑自举，因此复杂度可有效降低。

Sharma Iti 在其文章[23]中提出了一种基于上矩阵（非交换环）的对称秘钥体制无噪声FHE构造思路。主要方法是利用中国剩余定理解出三个与明文线性同余的整数，连同明文数据作为对角线形成明文矩阵，再随机生成一个4阶可逆矩阵，加密方式为，利用线性变换避免直接的选择明文攻击，相应解密即为，同态加法及乘法运算分别对应矩阵加法和乘法，其正确性由矩阵性质极易得到验证，且构造方式较为简单，算法复杂度较低，但其未能解决明文与密文之间的线性关系，Jing Li和 Licheng Wang[24]对此方案进行了改进，将明文矩阵通过增加随机数扩展为上三角矩阵，对于通过求解线性方程组获取等价密钥的问题，Li- Wang认为此类方案无法避免选择明文攻击，但对于选择密文攻击，可以在明文矩阵中增设校验矩阵，判定敌手所列密文是否合法，从而达到抵御非适应性选择密文攻击的要求，实现IND-CCA1（Indistinguishability under Chosen Ciphertext Attack）安全。

Koji Nuida[25]首次提出了一种基于非交换群构造无噪声FHE的框架，该框架也仅考虑单比特数据的同态加密。通过群的换位子及其随机化定义群元素上的AND、NOT等运算规则，以对应明文的相应运算，但文中只是提出一种理论模型，并未具体实例化，因此其安全性不能确定，仅具有理论指导性。M. Yagisawa基于有限域上八元数环构造了一种无噪声控制（Noise-Free）的FHE方案[26,27]，方案中，Yagisawa定义了八元数环上的加法及乘法运算规则，利用八元数的共轭特性，将有限数域上的明文编码为八元数形式的中间密文，然后在中间密文前后各乘以多个八元数及其逆元构成的序列，生成最终密文，其同态加法定义为八元数逐元相加，同态乘法定义为密文的二重加密，根据八元数非可交换性及结合律等性质，同态运算的正确性可得到验证，由于八元数及其逆元在乘法过程中的消去，密文不会出现明显膨胀，且无需考虑噪声。

Dongxi Liu基于向量及矩阵乘法构造一种有限数域上的无噪声FHE方案[28]。其秘钥为多维向量及相关矩阵，密文在形式上与私钥相同，也为多维向量，同态加法定义为各向量分量之和组成的新向量，即，同态乘法定义为密文向量分量两两之积组成的新向量与公钥矩阵的乘积，通过公钥矩阵将其维度降低为初始状态，即，其中，Dongxi Liu称，该方案是紧凑的，因为密文的同态运算并不改变密文向量的维度，且运算使得密文向量分量不会超出一定界限。

Yongge Wang指出[29]，文献[26,27,28]在安全性上均不能达到作者预期指标。其中文献[27]方案对明文的0和非0在密文形式上存在较为明显的差异，易被敌手猜测破解，达不到惟密文攻击(Ciphtext-only attack)安全要求；方案[28]由于要求公钥中必须包含一组明文1的密文数据，那么敌手可通过同态乘法得到若干明文1对应的密文组，从而通过解线性方程组获取密钥。在此基础上，结合Koji Nuida和Yagisawa方案，他提出了一种基于八元数环和矩阵的无噪声FHE方案， 但也谨慎认为，此方案仅达到弱唯密文攻击安全模型（Weak ciphertext-only security model）标准。

## 2.2 同态加密应用研究现状

随着理论研究的逐渐深入，虽然全同态加密技术还存在一些固有问题未能完全解决，但其具备的优秀同态特性也吸引了大批学者先后开展应用研究。

1. 电子投票

电子投票有着比传统投票方式更加准确便捷的优点，也不可避免的面临一些问题，投票人隐私保护与计票人公正性等各种因素都将影响其应用，同态加密技术可有效解决电子投票系统上述问题。其基本原理如下。

一个投票系统中，投票方、计票方、公布方三权分立。投票方保有个人投票隐私，其他各方无权获取该隐私数据；计票方在数据加密的情况下，汇总统计得票数据，交由公布方宣告投票结果。基于同态加密技术的电子系统中投票方采用公钥加密，同时私钥由公布方持有，投票方将加密的选票送到计票方，计票方进行同态操作，得到汇总数据，公布方拿到该数据后以私钥解密即可得到最终得票结果，计票方由于没有私钥，无法查看票面信息，可以防止计票方从中作弊，公布方也仅知道最终得票情况，从而有效保护投票方隐私。

Cohen 和 Fischer 首次提出了在电子投票系统中加入同态加密的概念[30]，并于该方案中描述了电子投票相关流程。现代互联网投票系统也一直沿用此方案，Ben Adida 等人[31]开发的 Helios系列系统是其中具有代表意义的系统。

由于同态技术方便简捷、且易于验证的优点，在 Helios 2.0 系统中，Ben 就采用了同态计数代替了之前Helios系统中的 mixnet方法，他使用一种 El Gamal 的变种算法对加密的选票进行计数统计，该算法有效确保了投票方的隐私。在其后的Helios系列系统中，Ben Adida均采了同态计数方法。

1. 隐私信息检索

隐私信息检索是指为保障在公共网络平台上的用户隐私而采取的一种阻止数据库知晓用户检索信息的策略，检索执行方在不解密数据的前提下，完成对所请求信息的查询并返回结果，由请求方解密获取所需信息。当前隐私检索方案大多检索效率低下，且适仅用于小规模数据。若在隐私信息检索协议中采用同态加密技术，即可利用同态性质对加密数据进行处理以完成检索，与传统的加密信息检索算法（安全索引、公钥搜索和线性搜索等）相比，基于同态加密技术的隐私信息检索在不打乱其对应明文顺序的前提下，对需要检索的数据进行任意操作，完成更多检索任务，有效提高检索效率，且加强了隐私数据的安全性。文献[32,33,34]分别提出了基于同态加密的密文检索方案，谷歌自2010年就已经开始提供加密搜索服务。

1. 数字水印

数字水印是一种信息隐藏技术，该技术可有效保护数据完整性，多用于防伪溯源、保护版权、安全认证等。数字水印技术当前所面临的最大挑战就是对其的各类攻击。针对数字水印的主要安全攻击手段是非授权检测，即攻击者在未被授权的情况下对嵌入水印的载体进行检测，以确定是否存在水印，进而猜测或破解水印的含义，甚至去除水印并嵌入一个伪造水印。当前的数字水印一般都是未经加密的显式数据，在这种情况下，攻击者一旦攻击检测成功，就完全有能力对其破坏乃至伪造。Zhi Li等人提出了一种基于同态加密的新型数字水印方案[35]，通过加密在水印中引入非线性内容，以抵抗基于线性操作的常规检测攻击，检测水印前需要先行解密，这种方法使得攻击者即使检测到水印，也无法获取其信息，从而保证水印的安全。

1. 安全云计算

云计算在当前各个领域应用广泛，但其安全性始终是必须首先考虑的问题。同态加密技术从其诞生之初，就被认为是解决云计算安全问题的天然最佳手段，但目前同态加密技术在云计算领域应用还不够普遍。

在医疗系统方面中，Chase M等[36]提出了一种存储医疗记录的云系统，医疗服务提供者对病人的医疗记录首先进行加密，然后传输病储存到云系统。病人通过与该系统中的某一服务提供者分享私钥共享其医疗记录。然而，该系统仅提供云搜索服务，并未提供云计算服务。利用同态加密技术，该系统就可对加密的记录进行统计分析等的计算，为病人提供相应的云计算服务。随着科技发展，基于无线局域网的监视器或其他各种医疗设备将能够持续地向云存储服务器传输加密数据，利用同态加密体制，云服务器可以对加密数据进行分析计算，并根据计算结果给病人提出警告或者建议等。美国加州大学圣迭戈分校专家在2015年的一次研讨会上提出一种基因数据测试方案，该方案基于同态加密技术，可在10分钟内找到与疾病有关的基因变异，他们已在规划诸如“精准医学计划”（美国，由总统奥巴马发起）以及 “10万人基因工程”（英国）等工程以获取足够数量的数据，同态加密技术将在其中起到决定性作用。李雪松等[37]提出一种利用同态加密体制实现的对数据密文统计和数据挖掘方案，并用医疗数据做了模拟实验，最终得到了预期数据。李京钰等[38]设计实现了一种基于全同态加密体制的电子病例云存储系统。

Naehrig M等在[39]中以某化妆品公司为例描述了全同态加密在数据分析方面的一个应用。该公司希望能根据顾客的行为习惯、爱好等相关信息有针对性的发送特定广告给目标顾客。若顾客的手机能够持续地上传个人信息，包括实时方位、邮件内容、浏览记录等，则该公司可用相关函数进行计算，根据结果决定发送什么样的产品信息给顾客。作为顾客是需要保护个人隐私信息的，因此可用同态加密体制对加密隐私数据，所有顾客的信息加密后传输到服务器，同时广告商也将广告加密上传，服务器对所有加密数据进行相关性分析，精准投放广告给顾客，顾客收到加密广告后用其私钥解密即可获得内容。

在金融行业，刘明洁等[40]分析了金融行业全同态加密技术的应用潜在需求，探讨了应用流程和特点，但并未给出具体技术和实现方案。李顺吉[41]探讨了金融云平台下银行应用同态加密技术的可行性，并提出了金融云中同态加密技术应用的粗略模型。

## 2.3 本章小结

全同态加密由于实现时间较短，目前在研究方向上更多偏重理论，实践层面偏少，研究具体行业具体场景中同态加密技术的应用，一方面可加快技术向生产力的转化，另一方面也可促进基础理论研究的进一步深入。

全同态加密虽然在构造方法上一直在创新，抑或在现有方案基础上不断优化，其算法效率、安全性能均有着不同程度的提升，但现有的全同态加密方案无论是在理论层面还是实际应用上，仍然有很大的改进空间。基于格的FHE方案构造和理解都较为困难，算法复杂度过高，作为FHE的先驱方案，更多的意义在于理论指导；基于整数的FHE方案构造简单易懂，但目前该类算法仅关注比特（单比特或者比特序列）数据加密方法，其明文空间不能有效覆盖实际应用中的各类场景；基于LWE的FHE方案初衷在于设计可同态运算有限次的算法以满足部分场景需求，若要实现真正的全同态同样需要自举设计，而该类型的自举方案复杂度并不低于前面两种方案；无噪声的FHE方案复杂度一般较低，明文空间也足够大，但所有已知的无噪声FHE方案所述安全性均未得到有效验证，始终不能达到实用的要求。

# 第三章 银行云计算中同态加密算法应用研究

本章主要介绍适应银行云计算场景的定精度实数同态算术运算方案研究与设计。

## 3.1 银行数据同态计算需求

在1.2中我们已经提到，银行数据按照字段值变动情况可分为静态数据和动态数据两大类，其中的动态数据又可依据变化规律分为替换型和计算型两类，如下表所示。对于静态数据及替换性数据，传统加密技术即可满足其安全需求，而计算型数据则需要应用同态加密技术以解决其在密文状态下的计算需求。观察下列计算型数据，主要集中在金额及利率等数据上，该类数据虽然种类不多，但却是银行业务的核心数据，也是银行主要的私密数据。从数据类型上看，该类数据首先都是实数，对其计算即为实数的算术运算，其次按照实际业务运营要求，该类数据的精度是一固定数值。由此可知，银行在应用同态加密技术上的主要需求为一种可在固定精度实数上运行的同态算术运算方法。

表1 银行常见数据列表

| **类别** | **字段名称** | **数据类型** | **数据类别** |
| --- | --- | --- | --- |
| 账户信息 | 帐号 | varchar(20) | 静态数据 |
| 开户人姓名 | varchar(20) | 静态数据 |
| 账户密码 | char(6) | 替换型数据 |
| 身分证号 | varchar(20) | 静态数据 |
| 账户余额 | float | 运算型数据 |
| 开户日期 | datetime | 静态数据 |
| 开户地址 | varchar(30) | 静态数据 |
| 银行信息 | 名称 | varchar (50) | 静态数据 |
| 代号 | varchar (20) | 静态数据 |
| 地址 | varchar (50) | 替换型数据 |
| 联系电话 | varchar (50) | 替换型数据 |
| 银行职员 | 姓名 | varchar (24) | 静态数据 |
| 职称 | varchar (50) | 替换型数据 |
| 工作编号 | char(7) | 静态数据 |
| 联系电话 | varchar (50) | 替换型数据 |
| 银行代号 | varchar (20) | 静态数据 |
| 柜台 | 负责人姓名 | varchar(24) | 替换型数据 |
| 工作编号 | char(7) | 静态数据 |
| 柜台号 | char(4) | 静态数据 |
| 存折 | 存折号 | varchar(20) | 静态数据 |
| 账号 | varchar(20) | 静态数据 |
| 姓名 | varchar(24) | 静态数据 |

表1 银行常见数据列表（续）

|  |  |  |  |
| --- | --- | --- | --- |
| **类别** | **字段名称** | **数据类型** | **数据类别** |
| 存折 | 开户网点 | varchar(30) | 静态数据 |
| 开户日期 | datetime | 静态数据 |
| 活期操作 | 帐号 | varchar(20) | 静态数据 |
| 金额 | float | 运算型数据 |
| 类型(存入，支出) | boolean | 替换型数据 |
| 账户余额 | float | 运算型数据 |
| 利息 | float | 运算型数据 |
| 存入日期 | datetime | 替换型数据 |
| 定期存款 | 帐号 | varchar(20) | 静态数据 |
| 存款人姓名 | varhcar(10) | 静态数据 |
| 存款金额 | float | 运算型数据 |
| 存款日期 | datetime | 静态数据 |
| 存储年份 | int | 静态数据 |
| 存储利率 | float | 运算型数据 |
| 定期取款 | 帐号 | varchar(20) | 静态数据 |
| 取款人姓名 | varhcar(10) | 静态数据 |
| 取款金额 | float | 运算型数据 |
| 取款日期 | datetime | 替换型数据 |
| 定期历史  操作记录 | 帐号 | varchar(20) | 静态数据 |
| 存取款人姓名 | varhcar(10) | 替换型数据 |
| 类型(存入，支出) | boolean | 替换型数据 |
| 办理日期 | datetime | 替换型数据 |
| 存储年份 | int | 替换型数据 |
| 存储利率 | float | 替换型数据 |
| 转账 | 转账人卡号 | char(20) | 静态数据 |
| 收账人姓名 | varchar (24) | 静态数据 |
| 收账人卡号 | char(20) | 静态数据 |
| 转账金额 | Decimal(9) | 运算型数据 |
| 余额 | varchar (10) | 运算型数据 |
| 转账时间 | Datatime | 静态数据 |
| 汇款 | 汇款人 | varchar(24) | 静态数据 |
| 汇款帐号 | char(20) | 静态数据 |
| 汇款时间 | Datatime | 静态数据 |
| 金额 | Decimal(9) | 运算型数据 |
| 交易码 | Datatime | 静态数据 |
| 贷款 | 姓名 | varchar(24) | 静态数据 |
| 帐号 | char(20) | 静态数据 |
| 身份证号 | varchar(18) | 静态数据 |
| 日期 | Datatime | 静态数据 |
| 担保方式 | varchar() | 替换型数据 |
| 利率 | Declmal(5) | 运算型数据 |

表1 银行常见数据列表（续）

|  |  |  |  |
| --- | --- | --- | --- |
| **类别** | **字段名称** | **数据类型** | **数据类别** |
| 贷款 | 贷款金额 | Decimal(9) | 运算型数据 |
| 挂失 | 姓名 | varchar (24) | 静态数据 |
| 帐号 | char(20) | 静态数据 |
| 身份证号 | char(18) | 静态数据 |
| 密码 | char(6) | 替换型数据 |
| 挂失时间 | Date | 静态数据 |

Christof Paar在其著作[42]中强调，“除非拥有经验丰富的密码分析团队帮你检查设计，否则切勿开发你自己的加密算法”，“不要使用未经证明的加密算法或未经证明的协议”。作为一名研究生，所掌握的知识、资源以及时间不足以独力构建一种可靠的加密方案。当前已有的全同态加密方案中，研究人员或者着重单比特明文的加密运算，或者聚焦有限数域内整数的加密运算，而实践中的明文空间远比这两种情况复杂，因此这些方案并不适用。鉴于此，在实际场景中应用同态加密算法时，应根据明文空间、同态运算深度等需求，综合考虑现有方案，确定最佳选项，并针对场景实际情况，进行适当修改和综合运用。根据上述对银行数据类型的分析，结合当前全同态加密构造方案及其特点，本文选择相对简单易懂且成熟安全的单比特整数上的全同态加密方案作为基础，设计构造一种定精度实数的全同态加密计算方案，以适用于银行云计算场景。由于所选DGHV方案存在秘钥尺寸过大，重加密算法复杂度较高的问题，本文对其进行了适当改进，降低了运算复杂度。

## 3.2 DGHV方案介绍

### 3.2.1 部分同态加密方案

部分同态加密方案又称为somewhat方案，是指允许同态计算有限次的加密方案，同样包括密钥生成、加密、解密和密文计算算法四个部分。首先选定几个参数：

方案的安全参数；

方案公钥比特长度；

私钥比特长度（也即公钥的隐含近似最大公因子）；

噪声的比特长度；

公钥中的整数个数。

这些参数有如下约束条件：

，以抵抗对噪声的暴力攻击（若未加特别说明，本文中的一律代指）；

，为了确保somewhat方案能够同态运算足够高次的多项式函数以运行压缩的解密函数实现全同态；

，为了抵御潜在的对近似最大公因子问题的各种格攻击；

，为了在规约至近似最大公因子问题时应用剩余哈希引理。

另外引入第二个噪声参数。DGHV方案建议参数为：。

*。*私钥为一个比特的大素数，随机选取；公钥为，其中，，对重新排序以确保最大，且为奇数，为偶数，若不满足此条件则重新选择。输出公钥序列。

。对于明文，随机选择一个子集，以及一个随机数，计算输出密文（其中表示，下同）。

*。*输入私钥和密文，输出明文

*。*对于一个t输入的函数，密文为一个大整数，计算整数上的乘法和加法，返回结果也为整数形式。这里可被执行且解密结果正确的函数称为容许函数。

对于此方案，密文实质上可表示为，其正确性很容易得到验证。

令，

则

+； （3.1）

； （3.2）

显然可得，。

这里需要明确的是，为了达到语义安全要求，同态加密方案加密时增加了随机数，其和明文共同构成噪声，此方案中，要求噪声，否则在进行运算时噪声奇偶性发生变化，导致解密错误。

由式3.1及3.2可以看出，在进行密文同态加法和同态乘法运算时，噪声也会随之增大，且增长速度不同，计算同态加法时，结果密文噪声为两计算数密文噪声的叠加，而同态乘法中结果密文噪声为两计算数密文噪声的乘积。可以推断，当同态计算尤其是同态乘法计算达到一定次数之后，结果密文的噪声必将超出限制从而致使解密错误，也就是说SHE进行同态计算的次数是有限的，由私钥参数及噪声参数、共同决定，并不是真正意义上的全同态加密方案。

### 3.2.2 重加密思想及实现思路

噪声阻碍了全同态的实现，那么如何去除噪声影响，实现全同态，是亟待解决的问题。一个直接的思路是将密文解密后再计算加密，这样结果密文噪声就不会增大，但是由于私钥不能公开，计算方是不可能使用私钥解密的；第二种思路是既然同态加密方案可同态计算密文的函数，而解密本身也是一个函数，那么是否可以同态运行解密函数，实现同态解密？解密函数输入为密文及私钥，输出为明文，若解密函数输入为二次加密的密文及一次加密的私钥，那么理论上输出就应该是一次加密的明文，即新鲜密文。



图3 重加密实现原理示意图

若有

且同态加密方案可计算次数大于解密函数次数，即可同态运行解密函数，由同态加密性质，有，若令，则有

这里的和一样都是明文的对应密文，但是相当于直接对明文的加密，噪声是初始状态，而可能是经过一次同态计算的大噪声密文，通过这样的一次重加密操作，即可实现密文噪声的消减，此技术称为密文刷新，对此过程Gentry在其论文[1]中通过手套箱的比喻做了详细描述。如果此刷新密文技术可行，则在每次同态计算前或者计算后增加一次密文刷新操作，使得密文噪声始终不超过限值，即可保证同态计算无限次运行下去，从而实现全同态。

现在问题的关键点在于前文构建的SHE方案能否同态运行解密函数，或者说SHE方案同态运算次数是否高于解密函数的复杂度。根据Gentry设定的规则，由于刷新前的结果密文至少经历了一次同态计算，所以密文刷新过程中各种运算产生的密文噪声应小于，若解密函数可解析为次多项式，因为“新鲜”密文噪声最大为，则有

即有

（3.3）

实际上解密函数的次数是多少呢？

观察解密函数

式中表示对临近取整。显然函数复杂度主要源于计算，可以通过估计的次数确定解密函数的最高次数。，为小数，为确保取整之后的正确性，需要保留位二进制小数。对于二进制数来说，乘两个t位数相当于加t个数（移位相加），如

u3 u2 u1

× v3 v2 v1

u3v1 u2v1 u1v1

u3v2 u2v2 u1v2

u3v3 u2v3 u1v3

图4 多位二进制乘法示意图

而加t个数可以利用Gentry提出的“3-for-2 trick”进行计算，即3个数相加可以转化为2个数相加，输出二进制数的各位是输入各位的二次形式，如

u3 u2 u1

v3 v2 v1

＋ w3 w2 w1

(u3+v3+w3)2 (u2+v2+w2)2 (u1+v1+w1)2

(u3v3+u3w3+v3w3)2 (u2v2+u2w2+v2w2)2 (u1v1+u1w1+v1w1)2

图5 “3-for-2 trick”原理示意图

那么t个数相加经过应用这一技巧次后可以转化为两个数相加，输出位次数为。

再看两个t位数相加，以两个三位二进制数相加为例。

u3 u2 u1

＋ v3 v2 v1

[u3+v3+(u2v2+u2u1v1+v2u1v1)]2 [u2+v2+(u1v1)]2 [u1+v1]2

图6 多位二进制数加法示意图

上图中结果各位上( )内的式子代表为临近低位的进位数据。由图6可见两个三位二进制数相加其输出位上的最高次数为3，也就意味着两个t位数相加其输出各位上的最高次数为t。

结合上述两步可知，两个t位二进制数相乘，其输出各位次数最高为，而中是位，同时为确保结果正确其精度也是至少位，那么至少是次，由于，所以解密函数复杂度一定大于，可见somewhat方案不足以同态运行解密函数。

如果依然考虑这一方法解决噪声问题，就需要把解密函数的复杂度降低。解密函数主要复杂度集中在上，由于乘法运算相比加法运算复杂，可以考虑将这一部分表示为一个加法形式。这里考虑将表示为一个和的形式，同时，控制好的精度，即可实现乘法到加法的转换，降低计算复杂度。由于为私钥不可泄露，因此需要把隐藏到中，DGHV方案采用的是SSSP（Sparse Subset Sum Prob，稀疏子集和）这一陷门来实现隐藏。

### 3.2.3 全同态加密方案

在somewhat方案基础上，增加三个参数，分别是和，其中

，，

在原来私钥和公钥的基础上，增加一个公钥向量，其中的为[0,2)区间上带有-bit精度的随机实数，同时对于一个大小的子集，满足，此时私钥组合中增加一个。方案修改如下。

*。*如somewhat方案生成私钥和公钥，令，随机生成一个维的二进制（仅有0和1两种不同的元素）向量，且其海明重量为，令**，**随机选择大整数序列，，满足约束。令，从而得到，这里得到的都是小于2且带有位精度的二进制实数，且满足**。**输出私钥，公钥。

。按照somewhat方案加密单比特明文生成密文；之后令**，**保留小数点后个二进制精度位，得到输出密文。

*。*输入私钥和密文，输出明文。

对于修改后的方案，研究一下其解密函数的次数。整个解密过程可分为三步执行。

1.对所有的，令**，**也就是说，当时，，当时，**，**的精度与保持一致；

2.生成个实数，其中每个数均有小于的二进制精度，使得；

3.计算输出。

步骤1为一次乘法运算，次数为2；步骤2如果对直接应用“3-for-2 tricks”，其复杂度依然会超过somewhat方案计算能力，但是考虑到只有个非零实数，可以将其简化。

将的每个bit拆分，表示为，用矩阵表示，如图7所示。算法的核心是生成个整数，其中的每一个对应生成矩阵的“列向量”海明重量。由于只有个非零实数，所以不大于，那么的比特长度不大于，根据海明重量计算公式，的每个二进制位可以用（即该列向量所有元素）的多项式来表示，其最高次数不大于。

图7 ai矩阵及其列向量海明重量示意图

对于步骤3，由于的比特长度，其最高复杂度应小于个位二进制数相加，根据前文提到的“3-for-2 tricks”，其最高次数为

综上，三步操作的复杂度应小于，由式3.3，若取，可知somewhat方案可计算多项式次数大于压缩后的解密函数次数，即可同态的进行同态解密，完成密文刷新操作，实现方案的自举以形成真正的全同态加密方案。

## 3.3 改进的单比特明文加密方案

根据DGHV方案总体框架，对其部分参数进行修改，以在不影响安全性的前提下适当降低复杂度，使之更加实用。

### 3.3.1 somewhat方案

#### 3.3.1.1 方案设计

参数选取及设置：，为安全参数。

*。*随机选取一个比特的大素数作为私钥；随机选取一个比特的素数，计算，然后选取两个随机整数和，计算，输出公钥。

。明文，随机选择两个整数和，计算输出密文。

*。*输入私钥和密文，输出明文。

*。*对于一个t输入的函数和相应密文，将明文的模2运算映射为密文的模N运算，计算整数上模N的乘法和加法，返回整数形式的结果。

#### 3.3.1.2 同态性验证

若对应两个明文和的密文分别为，根据加密算法可知，存在两个整数和，使得，即有，那么显然是的有效加密形式，噪声两密文噪声之和。

同理，

也是的有效密文形式，噪声为两密文噪声之积，若该噪声依然满足小于的要求，即可正确解密得到

#### 3.3.1.2 可计算多项式次数

对照DGHV方案，可知本方案密文噪声为，为保证解密正确性，同样要求。

由于，与DGHV方案相等，参照DGHV方案，设为本方案可计算多项式次数，则有

#### 3.3.1.3 安全性分析

本文所述的改进的单比特明文同态加密方案，其安全性可规约至文献[43]中的部分近似最大公因子问题（PACDP，Partially Approximate Common Divisor Problem）。

**定义** 给定两个整数，其中均为某大素数的整数倍，且有。根据、，求素数。

类似DGHV方案[10]，有如下定理1。该定理证明过程与DGHV方案基本相同，在此只简要描述一下其证明过程，证明过程需要调用文献[10]中所述的算法，算法详情参见原文。

**定理** 若在上述同态方案攻击中所具有的优势为，那么通过调用，作为第二攻击者解决PACDP的优势为。

**证明** 挑战者首先生成一个近似最大公因子问题的实例，即按照上述SWHE方案，随机选取的素数和比特的素数、整数和，计算和。获取和后，以之为公钥，利用SWHE方案中加密算法产生多个对应明文0的密文，对每个而言，存在相应的两个整数，满足 。以从中随机选取的两个密文、调用算法，将输出，之后用和再次调用，输出 的比特串，最后通过计算可恢复。与文献[10] DGHV方案相同，得到的优势为。

### 3.3.2 全同态方案

#### 3.3.2.1 自举的实现

由于方案解密函数及SHE可计算多项式与DGHV方案相同，故本方案也不是自然的全同态方案，需要压缩密文完成同态解密以实现自举。

为同态解密增加三个参数，分别为，，，对应新增公钥的精度、扩展密文数量和有效密文的数量。

*。*按照3.2.1中的SHE方案生成私钥和公钥。令，随机生成一个维比特向量，确保其海明重量为，且要求最后一个元素，令其指标集，在上均匀随机选取，之后计算，令，生成新的公钥向量，输出私钥，公钥。

。按照SHE方案加密单比特明文生成密文；若同态计算需要进行密文刷新，则令**，**保留小数点前一位数字及小数点后个二进制精度位，得到扩展密文，输出密文。

*。*输入私钥和密文，输出明文。

*。*对于一个t输入的函数和相应密文，提取其中的主密文，将明文的模2运算映射为密文的模N运算，对其计算整数上模N的乘法和加法，在每次乘法计算后，将生成的结果密文进行扩展，并进行一次重加密刷新密文，以降低噪声。返回整数形式的密文计算结果及相应拓展密文。

参照DGHV方案，为降低解密函数复杂度，将解密过程分为三步进行。

步骤1 计算；

步骤2 根据步骤1的个实数生成个实数，使得；

步骤3 计算并输出。

显然，步骤1为一次乘法，次数为2，步骤2中为海明重量，其次数最高为，按照“3-for-2 trick”，步骤3所需次数为

整个解密算法的复杂度为，用加法或者乘法扩展解密函数，其最高次数为，所以本方案的自举条件是

当足够大时，上述不等式成立，即方案可实现自举，成为全同态加密方案。

#### 3.3.2.2与DGHV方案对比

本文所述之单比特明文全同态加密方案，与文献[10]中的DGHV方案相同，均为基于整数的方案，基本框架相似，这里基于前文在安全性、方案效率方面等简单对比下两种方案。采用了包括安全级别、困难问题等安全性的指标和密钥尺寸，加/解密复杂度等效率指标来衡量。

1.SHE方案对比。

表2 本文SHE方案与DGHV方案的对比

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **方案** | **安全级别** | **基于的困难问题** | **公钥尺寸** | **私钥尺寸** | **可计算多项式次数** |
| DGHV方案 | IND-CPA | 近似最大公因子问题 |  |  |  |
| 本文方案 | IND-CPA | 部分近似最大公因子问题 |  |  |  |

从表2可以看出，本文的SHE方案所基于的困难问题为部分近似最大公因子问题，达到了IND-CPA安全级别，该问题的困难度相比DGHV方案要弱一些，但本文方案在可计算多项式次数相同的情况下，公钥尺寸更短，算法效率要高些。

2.FHE方案对比。

从表3可以看出，本文扩展解密算法具有比DGHV方案更小的复杂度，更容易实现自举而达到全同态，同时密钥尺寸也更短。

表3 本文FHE方案与DGHV方案的对比

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **方案** | **基于的困难问题** | **公钥尺寸** | **私钥尺寸** | **扩展解密算法的多项式次数** |
| DGHV方案 | 近似最大公因子问题、稀疏子集和 |  |  | 约 |
| 本文方案 | 部分近似最大公因子问题、稀疏子集和 |  |  | 约 |

DGHV方案中，为控制密文规模，引入了模的运算，由于中包括了这一噪声分量，而在同态运算过程中，乘法和加法都需要进行模运算，由此引入新的噪声将使得方案的可计算多项式次数降低，因此文献[10]中另行设计了一个优化算法以降低这一噪声，而本文方案则采用了模运算，没有引入新的噪声，因此不再需要优化算法。

### 3.3.3 基于二进制进位加法的重加密实现方法

在文献[10]中，对重加密的可行性进行了论述，但并未涉及具体实现重加密的方法，本文在这方面进行了探究，综合文献[44]给出重加密实现的具体流程。

重加密核心就是同态的运行解密函数，以实现在不泄露私钥的情况下对密文进行刷新降噪，这里首先在明文私钥的基础上进行基于二进制的算法分析，之后将其映射到密文空间上。

抛出有关的几个引理。

**引理1**设为一维二进制向量，若该向量的海明重量用表示，且的二进制表达式为，即，那么第位的比特可用一个关于的次数为的齐次对称多项式表示，如下所示

即有。其中的是向量的指标集的子集，即。

**引理2**在本方案（整数上的单比特明文同态加密方案）中，明文空间的模2与运算、异或运算将分别映射到密文空间的模N运算乘法、加法运算。

**引理3** 在本方案（整数上的单比特明文同态加密方案）中，0和1是密文空间的有效元素，且分别是0和1的有效密文，即有，。

引理1可用实例验证，引理2、3根据同态加密方案性质可得到证明，在此不再给出证明过程。

**引理4** 对只有一个整数位的二进制实数，即，则有。

证明：根据、、和取值的对应关系，可得如下真值表。

表4 临近取整真值表

|  |  |  |
| --- | --- | --- |
|  |  |  |
| 0 | 0 |  |
| 0 | 1 |  |
| 1 | 0 |  |
| 1 | 1 |  |

根据表4，显然可得。

#### 3.3.3.1 二进制上的解密函数分析

全同态方案中，通过压缩密文降低解密函数复杂度，使得SHE方案可满足解密函数次数要求，为达到这一目的，将解密函数 分为三步进行分析，参照这一方法，将步骤二和步骤三合并分析。

步骤1 计算，精度保持为不变；

步骤2 计算； 将每个按照二进制位拆分成一个行向量，然后所有的构成一个维比特矩阵。

图8 ai矩阵末尾列本位比特及进位比特示意图

要求，可以按列求和然后把每列的和相加。逐列来看，对每一列求和相当于求该列向量的海明重量。海明重量的二进制形式最低位即为和的本位比特，那么次低位即为向临近位的进位比特，以此类推，海明重量的其余每一个二进制位为该列向其它对应高位的进位比特。先看最后一列，该列海明重量二进制最低位为，次低位记为该列(列)向列的进位比特，由引理1，，即列任意两个元素乘积和，以此类推，，为列任意四个元素乘积和，……。由于向量的海明重量为，那么对应所形成的比特矩阵中，最后一列至多有个1，也即该列的海明重量最多有位，进位到为止；其后计算第列，此时把看作该列元素，即该列有个元素，由于未知，故将其看做非零元素处理，即本列海明重量至多为，采用同计算列一样的方法计算本位比特和进位比特，最高进位至；计算列时后将两列的进位数据看做本列元素，如此进行下去，直至计算完所有小数列的本位及进位数据，由于解密函数执行模2运算，计算该矩阵时仅考虑第0列及以下的本位比特，故去除所有进位至第1及以上列的的比特。

图9 基于海明重量多项式的二进制进位加法示意图

根据引理4，即为第0列及第1列本位比特的模2和，即。

步骤3 计算，即完成解密。

#### 3.3.3.2 重加密实现步骤

在Gentry的论文[1]中，介绍了重加密步骤。

令，为单比特同态加密方案的两个密钥对，明文，密文，表示解密函数，表示重加密函数，通过运行，即可在不泄露任一私钥且不直接解密的情况下完成把下的密文转化为下的密文。算法的执行需要对私钥进行处理以防止泄露，即私钥在密钥下进行加密。算法的过程表示如下：

1.，其中的是指私钥的第个比特；

2.，其中的是密文的第个比特；

3.

在该算法中，方案在满足循环安全（Circular Securit）的前提下，两个密钥对可以相同，另外对私钥和密文比特的加密采用的是SHE方案。

具体到本方案，可按下述6个步骤进行。其中私钥的加密需要提前完成，步骤1计算主密文的再加密，步骤2计算扩展密文的再加密，步骤3到步骤5完成扩展密文的同态解密，步骤6生成刷新后的密文。

重加密算法的输入：FHE方案密文，公钥，私钥向量的加密=<，为私钥向量各比特元素在SHE下的密文。

步骤1 计算的密文。

步骤2 将扩展密文的每个按二进制比特展开成行向量，则所有的构成一个维比特矩阵A。

按照Gentry重加密设计思路，对矩阵A的每个0,1元素应进行SHE加密，生成对应密文，但为计算方便，此处可不进行加密，因为按照前述的引理3，本方案中0,1是密文空间的有效元素，且其与明文0,1是一一映射的，换言之，0即是0的有效加密，1也是1的有效加密。

步骤3 将乘以矩阵A的第行中的每一个元素，即，得到一个上的维矩阵，其中。

步骤4 按照3.2.3.1二进制基础上的步骤2，计算矩阵B的前两列和。

**计算第列**

由于私钥中只有个1，所以矩阵B中列对应明文的海明重量至多为，其海明重量的二进制形式至多有位，由该列向量产生至多个进位比特和1个本位比特密文，各比特数据的计算方法已在步骤2中明确，由引理2，对明文比特的模2运算将映射到密文上的模N运算，因此，将3.3.3.1中的步骤2计算本位比特及进位比特的二进制与、异或运算替换为模N的乘法、加法运算，同时将明文比特替换为对应密文（即矩阵B中相应位置的元素），即可得到列本位比特及进位比特的密文。即本位比特密文

进位比特密文分别为

**计算第列**

对列的计算与列相同，但由于列计算时产生了向其进位的密文数据，所以需要把作为列的元素一并计算，即列的维度增加一维变为，且其对应明文比特向量的海明重量至多也变为，产生至多个进位比特和1个本位比特的密文。本列本位比特密文为

进位比特密文分别为

重复运行这一方法计算其余各列，计算第列时注意，一定要将所有列在列产生的所有进位密文加入该列形成新的列向量来处理。计算完各列后，由于解密函数无需考虑二进制小数点前第二位及以上的数据，所以将所有进位密文中的数据舍弃，最后将每一列的本位比特密文按次序组成一个行向量，即。

步骤5 计算所对应的二进制小数取整后的的密文。

步骤6 计算重加密后的密文。

的输出：刷新后的密文。

## 3.4 定精度实数同态算术运算方案

本节针对银行数据特点，探讨一种在单比特明文同态加密的基础上构造的定精度实数同态加密方案，其基本思路是将带小数的实数二进制化并将小数点右移一定位数后逐位进行同态加密，然后设计多位密文的类二进制同态运算规则完成同态计算，解密时则逐位进行解密并左移相同位数进行还原，在整个运算过程中需考虑精度控制，以在可忽略解密误差的基础上保证密文规模不会过度膨胀。

### 3.4.1 数据标准化及加密

用于构建定精度实数同态算术运算方案的基础为单比特同态加密算法，因此常用的十进制实数需要转换为二进制形式，且由于同态加密运算方案时间复杂度和空间复杂度均为参与运算的数据长度的高次多项式，为控制复杂度及便于对齐运算，需要根据场景需求对参与运算的数据进行精度控制，本文统称为数据标准化。数据标准化包含两个方面，一是输入数据的二进制转换及精度的统一，二是二进制补码的表示。

对于十进制到二进制的转换很简单，不需要探讨，这里主要考虑数据精度问题。银行系统中需要同态运算的数据为金额类带一定小数位数的实数，其小数位数在输入时并不统一，为尽量保留有效数据，降低运算误差，本方案中对输入数据首先进行小数点在二进制意义上右移固定位数，即，其阶码由计算场景对数据精度的要求确定，一经确定，在整个方案运行过程中不再改变，然后临近取整后执行整数的二进制转换，其后的每个二进制位执行3.2.1中的SHE方案生成密文序列，例如，输入计算数，取阶码为，则，，其后对逐比特加密，还原数据时，解密得到，对应十进制为2647，然后，取两位有效小数位即得10.34。在参与多位密文同态运算时，小数点位置（即阶码）作为隐含数据不再体现在密文中，仅在执行乘法时用于多余尾数的舍弃和解密后还原数据初始小数精度。

参照银行业务，该方案中需要考虑同态减法的规则，对于单比特同态加密运算，加法与减法并没有区别，但多比特二进制数必须区分加减操作，计算机中采用的常规方法是用补码表示计算数，用加负的方法完成减法运算，这里参照这一模式处理，考虑运算数的补码表示。对负数而言，常规的补码表示方法需要预先确定数据类型以确定数据长度，在数据有效位数及符号位之间补1，而方案并未对数据长度做要求，因此需要另行考虑合适的表示方法。本文采用的方法为：负数在进行本节所述的精度控制和二进制转换后，直接逐位取反末位加1，形成不带符号位的二进制补码数，称之为类补码，然后直接按位进行3.3.1中的SHE加密，在同态减法规则中再考虑符号位的问题，其阶码等问题与前述方法一致，详见3.4.2.2。

### 3.4.2 基于二进制的实数同态运算规则

#### 3.4.2.1 多位同态加法

同态加法是方案的基础算法，本文所述的密文同态加法规则参考二进制基础上的加法规则制定。由于多个多位二进制数相加可首先计算前两个二进制数相加，将计算结果与第三个数相加，以此类推，直到完成所有数据的加和，因此这里只需考虑两个多位二进制数的加法运算规则即可。

**c3**= u3v3+u3c2+v3c2 **c2**= u2v2+u2c1+v2c1   **c1**=u1v1

u3 u2 u1

＋ v3 v2 v1

c3 [u3+v3+c2]2 [u2+v2+c1]2 [u1+v1]2

图10 多比特密文加法运算规则示意图

首先来看两个多位二进制明文的加法计算过程。两个多位二进制数相加是3.3.3节中所述方法的一个特例。以两个三位二进制数相加为例，末位直接进行模2加法得到本位比特，其进位比特只有一个。次低位需要考虑末位的进位数据，在明文计算时，进位可直接由最低位两个被加数判断出是否为零，以此决定计算次低位时是否需要计入，但在同态加密情况下，无法判断密文是否为零，因此该进位数据必须计入次低位的加法运算，即次低位相加时其计算数为三个，、和，其本位比特，由于其海明重量至多为3，故其进位比特也仅有个，为。以此类推，在计算第三及以上位时，由于其临近低位进位数据只有一个，故该位计算数至多三个，其本位比特计算及进位比特计算均与次低位相同，如此计算下去，直到完成所有位的加和，最高位前虽无计算数，但其进位数据依然有效，需直接放于前一位上，得到所有位的本位比特序列，也即计算结果的二进制形式。

根据3.3.3节引理2，将明文二进制运算映射到密文模N整数运算，用相应位的密文替换明文比特，同样将进位比特的密文纳入加法运算，如末位密文，末位进位比特密文，次低位密文，……，依次计算完所有密文序列，即可得到两个多位密文序列的加和结果密文序列。这里由于密文序列对应的是整数明文（标准化过的实数），所以在进行同态加法运算时需要右对齐，向左进位直至所有进位密文全部计入结果（即使最高位左侧已无本位计算数，但最高位的进位密文依然需要向左进位）。这里我们会发现，最高位可能会产生“虚进位”，即最高位的进位密文对应明文比特可能是0，这并不会使结果产生错误，但是会造成数据冗余，以致下一步计算的复杂度增加，需要进行优化处理。

需要注意的是，按照密文同态加法规则，在计算进位数据时多次出现了形如的单比特密文同态乘法，根据SHE方案性质，此类乘法操作会使得结果密文的噪声急速增大，对此乘法产生的结果密文，如果继续带入下一次的密文同态运算中去，则很快将出现噪声超过阈值导致解密错误的情况，因此在本方案中，所有的密文序列在计算过程中，一旦出现单比特密文的乘法运算，其相应的结果比特密文必须参照3.3.3节密文刷新操作的步骤更新一次，以控制噪声增长，每次刷新后的比特密文其噪声相当于新鲜密文，在有限次的乘法运算中不会出现解密错误。

#### 3.4.2.2 多位同态减法

在3.4.1节中已提及，在标准化时，若判断输入数据需要执行减法操作，则将减数直接标准化为原数据的类补码——移位取整后逐位取反，末位加1。与标准补码不同的是，为降低加密运算负担和取消方案对计算数位数的限制，本文所述的补码在加密前不再标示符号位，符号位与有效位之间也不再补1，而仅在密文序列进行同态运算时，直接将减数序列按照被减数序列的长度补齐，这里所说的序列长度指的是对应明文二进制位的长度。执行减法运算相当于对补码执行加法运算，为便于表述，此处仍以多位二进制明文减法为基础，分析密文序列相减的规则。

例如，被减数为21，对应二进制为，减数为7，对应-7的“补码”为，计算时，将按照被减数位数补齐为，执行二进制加法，如下图所示。

**舍弃**

图11 多比特密文减法规则示意图

图11中，由于减法不会产生比被减数大的结果，其最高位由于加法产生的进位数据需要舍弃，最终得到，可知结果正确。

多位密文的减法参照上述明文二进制补码运算进行。将上述“补码”逐比特执行3.3.1中的SHE方法加密。密文同态减法操作时，根据3.3.3节引理2，可在减数密文的最高位密文序列前直接补1（而非1的密文）至与被减数序列长度相同，然后执行3.4.2.1中所述同态加法运算，舍弃最高位的进位密文，即可得到结果密文序列。

进行密文同态减法操作时，若两操作数位数不等，即密文序列长度不同，可直接判断是否“够减”，需要注意的是，若两操作数位数相同，则无法在密文状态下直接判断被减数是否大于减数，即无法判断计算结果是否溢出造成错误，对此，需要同态计算前进行相应判断。

#### 3.4.2.3 多位同态乘法

首先仍以二进制明文乘法为基础进行分析。多位二进制乘法的基本运算规则是移位相加，如下图所示。

u2 u1• u0

**小数点**

× v2 v1• v0

u2v0 u1v0 u0v0

u2v1 u1v1 u0v1 0

u2v2 u1v2 u0v2 0 0

… **•**  u1v0+ u0v1 u0v0

**小数位数变为**

图12 多比特密文乘法规则示意图

两个多位二进制数相乘时，乘数的最低位（标记为）与被乘数各位相乘，得到一个二进制数，记为，次低位（标记为）与被乘数各位相乘，得到的二进制数左移一位，与相加，记为，同样的，第三位（标记为）与被乘数相乘得到的二进制数左移两位，与相加，记为，重复执行次步骤，将乘数序列按位次从低到高逐位与被乘数序列的各位相乘，将乘数每一位相乘得到的二进制序列左移所在位次次，然后相加，即得乘法运算最终结果。记输入数为**，**分别是、的二进制位，则计算结果可表示为。

由3.3.3中的引理2，将明文二进制乘法操作映射到密文空间上，替换为，每个乘数位密文与被乘数密文序列相乘产生的结果密文序列进行左移时，只需在该结果序列末补上（为该乘数位所在位次）个0，然后右对齐进行密文序列相加即可。这里需要调用3.4.2.1中同态加法对相乘移位产生的密文序列进行求和，注意所有对单比特密文的同态加和同态乘均需模N以降低密文长度。这里同样出现了单比特密文的乘法操作，也需要在每次比特密文乘法计算后对结果密文进行刷新操作。

由于方案是将一定精度小数转化为整数进行加密运算的，解密时将通过逆运算还原小数部分，而按照二进制乘法规则，在相乘后，产生结果的小数位数将与计算数的位数不同。在同态加法过程中，序列相加并不影响隐含小数点位置（也即阶码），但在同态密文乘法操作中，根据通用乘法规则，若计算数的二进制小数均为位，计算结果的小数位数即变为，如图12，此时再按照原来的解密还原方法获取明文结果，由于小数位的变化将造成数据错误，需要考虑结果密文值才能正确解密还原。一种方法每次同态相乘计算后保留所有结果密文序列，将结果密文的新值单独记录，解密时取用，但这样做将导致密文序列长度随着乘法运算的次数快速增长，为解决这一问题，采用将结果密文小数位数统一化的方法，即每次将结果密文按照被乘数的阶码进行统一，在误差率允许的情况下，在上述步骤的基础上对所生成的结果密文序列进行去尾，保证结果始终带有个小数位。按照四舍五入的法则，结合3.3.3中的引理4，直接舍弃序列末尾的个密文，然后将自低位起第个密文加到第位上，以保证对应明文小数点位置不变，且误差可控。

## 3.5 本章小结

本章主要分四个部分描述了适应银行在云计算环境下的数据同态加密及算术运算方案。首先，分析了银行数据特点及同态加密需求；其后，为便于理解和表述，对所参照的DGHV整数上同态加密方案进行描述，重点关注其参数设置和重加密思想；之后在DGHV方案基础上提出一种改进的单比特整数全同态加密方案；最后以单比特全同态加密方案为基础构造了定精度实数上的全同态加密方案。

# 第四章 银行云计算中同态加密应用原型系统设计与实现

本章是主要根据银行业务逻辑提出了其在云计算环境下应用全同态加密技术的模型，并设计实现了原型程序。

## 4.1 需求分析

### 4.1.1 银行云计算中同态加密应用模型

在同态加密的一般应用模型中，数据拥有方通过同态加密将隐私数据上传到非可信云存储平台，由非可信计算方对密文进行相应处理并返回存储平台，数据拥有方根据需求访问存储密文并解密输出预期计算结果。



图13 同态加密一般应用模型

全同态加密技术在银行云计算中的应用基于通用模型进行扩展，其模型如图134所示。

云计算场景下，银行IT系统及数据全部部署于云计算平台，本地中心保留数据加解密及通讯相关软硬件。作为数据拥有方，需要完成全同态加密计算方案下列任务。

密钥生成。私钥必须由数据拥有方掌握，故密钥生成算法必须由银行完成。根据需求确定安全参数，生成方案私钥，公钥，存入本地服务器密钥库，其中公钥传输给数据计算方（远程应用服务器）。由于银行系统数据量特点，更换密钥工作量很大，按照一般性加密要求，密钥一次生成在一定时间内多次使用，根据相关规定定期重新运行密钥生成算法，生成新的密钥并更新密文。

数据的标准化及加密。根据数据精度要求确定阶码，对交易输入数据进行标准化处理，并按照3.3.1中所述SHE方案逐位加密交易数据，将阶码、密文序列及交易类型（对应同态运算函数）传输存储至云计算平台远程数据服务器。



图14 银行云计算中同态加密应用模型

数据解密。1.根据客户需求读取远程数据服务器上相应密文数据，按照3.3.1中所述SHE方案解密数据并输出；2.根据3.4.2中同态运算规则，最终密文序列可能存在虚进位，应对此情况进行处理并返回处理后的数据至存储平台；3.根据远程应用服务器请求对操作数解密以判断同态减法是否合法。

云平台包括云存储及云计算两部分，分别对应银行的远程数据服务器和远程应用服务器，承担银行数据的存储和大部分计算任务。由于云平台非可信的特点，远程数据服务器和应用服务器均看做不可信计算方，主要完成以下工作。

数据存取。接收银行传入的密文数据并存储，根据银行及同态运算请求读取输出相应数据。

三方源数据的标准化及加密。为尽量减少银行端计算任务，将第三方发起的交易数据交由远程应用服务器进行加密。远程应用服务器接收并保存公钥和阶码，对三方明文数据进行标准化并调用SHE方案算法完成加密。

密文数据同态计算。根据交易类型选取对应同态运算函数，按照3.4.2所述规则完成密文序列的加法、乘法等操作，返回计算结果至远程数据服务器。

密文扩展和刷新。按照Gentry全同态加密框架，密文扩展是密文刷新的基础准备工作，随着密文刷新的运行而产生，因此在本文中不再直接对所有加密数据进行扩展，且将密文的扩展交由数据计算方直接完成，密文刷新完成则扩展密文用毕废止，以节约存储及网络传输负担。另按照Gentry框架，增强“门电路”需要在同态执行加法或乘法操作前即对密文进行刷新操作，这对算法效率不理，本文中只对密文乘法运算结果执行密文刷新操作，鉴于同态加法噪声增长缓慢，且在最不利情况下，SHE方案可满足数万次及以上同态加法操作而不至于解密错误，因此对加法运算结果不再刷新。在对密文序列进行同态运算过程中，每次调用单比特密文同态乘法后，均按照3.3.2中所述步骤扩展结果比特密文，并按照3.3.3中所述步骤对结果比特密文进行刷新，始终保持所有单比特密文均处于低噪声水平，满足后续运算要求。

### 4.1.2 原型程序功能模块



图15 原型程序功能模块示意图

根据第三章定精度实数同态加密计算方案和银行云计算同态加密应用模型，确定原型程序功能模块主要分为单比特同态加密和密文序列同态运算两大部分，其中单比特FHE以单比特SHE为基础，包含密文扩展和密文刷新两个主要模块，单比特SHE则由密钥生成，加密/解密三部分组成。同态运算部分主要分为密文序列的同态加减运算和乘法运算两个内容。另外程序还需要人机交互界面以输入计算数据和交易类型，并对数据进行标准化处理。

## 4.2 总体设计

整个程序以B/S模式展现，浏览器端主要实现数据的输入和回显，后台按照运行主体不同划分为两个部分，分别为数据拥有方（银行）后台和数据计算方（云平台）后台，且分别部署于代表可信中心和非可信云的两台不同电脑上，通过局域网通信。

为检验方案中同态运算的正确性，程序中以一个模拟银行账户的形式设计存款、取款和计息三种交易类型，分别对应同态加法、减法和乘法，另有余额查询功能，查看密文运算结果及检验解密正确性。为便于展示程序运行过程，后台增加了一个日志文件写入，按程序流程将产生的中间数据存储以查阅。

程序设计语言为java，并用mysql数据库设计一个简单表用于存储计算结果。表结构如下，其中balance字段为数组，存储密文序列计算结果，其余字段不参与同态加密和计算过程。

|  |  |  |  |
| --- | --- | --- | --- |
| **序号** | **字段名称** | **类型** | **说明** |
| 1 | no | varchar(18) | 账号 |
| 2 | name | varchar(30) | 账户名称 |
| 3 | balance | text | 账户余额，用字符串表示的密文序列 |

表5 模拟账户表结构

数据拥有方（银行本地服务器）后台服务执行流程图如下。



图16 数据拥有方程序流程图

数据计算方（远程应用服务器）后台服务流程图如下。



图17 数据计算方程序流程图

## 4.3 功能实现

### 4.3.1 初始化参数

初始化参数中数据拥有方需确定安全参数及其关联参数，阶码，该参数均设计为手动输入，数据计算方需建立数据表并定义初始数据。对于安全参数，为一个大于3的整数，全同态加密方案复杂度由决定，在原型程序中一般取值较小，避免密钥生成时间过长，输入后，其关联参数如私钥长度、公钥长度均可确定。阶码决定加密运算数据误差，由银行数据精度要求决定，由于银行余额精度一般取0.01，故即可，为进一步减小误差且控制计算复杂度，建议取值为9-12。

### 4.3.2 同态加密

#### 4.3.2.1 密钥生成

密钥生成部分用于生成整个加密方案的密钥，其中SHE方案的密钥核心是确定位数随机大素数的生成方法，FHE部分密钥的难点则在于高精度随机实数的生成。

随机大素数的生成用到了费马小定理：假如为质数，且，那么 。其基本流程为：

(1)随机产生一个 位的二进制数；

(2)分别设高位和低位为1以保证位数且过滤掉所有偶数；

(3)用常规的素数检测方法检查是否可被百位（或千位）以内的小素数；

(4)随机产生整数运行Rabix-Miller检测，若通过则产生新的随机数再测试，重复5 次Rabin-Miller 测试，即可判断大概率为素数；

(5)若测试失败则重复以上步骤。

为生成高精度实数，首先随机生成位范围内的大整数序列，并使其满足部分和约束，然后对所有大整数左移位，即可得到带有位小数精度的实数。

#### 4.3.2.2数据加密/解密

程序中所有数据在按照3.4.1中步骤进行标准化后，其单比特数据的加密及解密均按照SHE所述方法进行，扩展密文生成及以扩展密文解密的步骤不再使用。

BigInteger ciphertext = BigIntegerUtil.newBigInteger(m.toString())

.add(BigIntegerUtil.newBigInteger("2").multiply(r1))

.add(r2.multiply(publicKeys[1])).mod(publicKeys[0])

String resStr = c.mod(new BigInteger(privateKey.toString())).toString();

int result = (resStr.charAt(resStr.length() - 1) - '0') %2

在进行密文序列同态运算后，结果密文最高位可能是虚进位，在密文状态下数据计算方无法判断处理，对此在每次同态计算后，由数据拥有方读取结果密文，并从最高位开始逐位解密验证该位是否为0，若为0则舍弃该位，直至遇到1为止。虚进位的处理有利于降低密文运算复杂度，提高运算效率。

在进行同态减法运算时若出现减数与被减数位数相等，数据计算方同样无法判断该计算是否合法（余额出现负数则不合法），同样需要数据拥有方取回计算数进行解密，以明文方式判断计算并反馈结果。

#### 4.3.2.3密文扩展及刷新

密文的扩展完全为密文刷新服务，为节约存储空间和网络资源，在数据加密及传输时仅通过SHE产生主密文，需要进行密文刷新时再有数据计算法通过公钥生成扩展密文。密文刷新计算按照3.3.3.2所述步骤由数据计算方完成，其中步骤4计算密文矩阵前两列和过程中，由于前期的扩展密文按位展开后形成的0,1矩阵个元素未经加密，即其中0的个数约占整个矩阵元素的一半，即便与相乘，0元素个数依然保持不变，在进行按列求和尤其是计算进位比特密文过程中，需要对列元素取子集求积然后求和，而包含0元素的子集其积必为0，也就是说该子集对结果没有贡献，将0元素去掉可有效减少子集数量从而极大地降低计算复杂度且不影响最终结果。在实现该步骤时，首先对各列进行清理，去掉所有0元素，然后在按照海明重量计算公式求各列本位比特和进位比特密文。

### 4.3.3 同态计算

由于同态减法是以加法形式实现的，故同态减法并入加法，同态计算代码分为同态加/减和同态乘法两个部分。同态乘法需要调用同态加法代码实现，所有同态计算的过程均会产生单比特密文的乘积形式，因此需要调用密文刷新模块。核心代码如下。

同态加法：

for (i = 0; i < len3; i++) {

f1 = BigIntegerUtil.newBigInteger(lng.get(i)) ;

f2 = BigIntegerUtil.newBigInteger(shrt.get(i)) ;

result.add(f1.add(f2).add(carryBit).mod(sk.getPublicKeys()[0])) ;

carryBit=CombineAlgoUtil.newCombineAlgo(BigIntegerUtil.newBigIntegers(f1,f2,carryBit), 2)

.combineAndSum().mod(sk.getPublicKeys()[0]) ;

carryBit= FullyAlgo.refresh(carryBit,sk.getPublicKeys()); } //密文刷新

while(i < len4){

f1 = BigIntegerUtil.newBigInteger(lng.get(i)) ;

result.add(f1.add(carryBit).mod(sk.getPublicKeys()[0])) ;

carryBit = f1.multiply(carryBit) ;

i++ ; }

if(carryBit.signum() != 0)

result.add(carryBit) ;

return result ;

同态乘法：

for (int j = 0; j < param1.size(); j++) {

item.add(FullyAlgo.rencrypt(param2.get(i).multiply(param1.get(j)),sk.getPublicKey()), sk.getPublicKeys()) ; }

result = plus(result, item, pk) ;

## 4.4 本章小结

本章是第三章所述方案的应用与验证，主要是设计了银行云计算环境中应用全同态加密技术的模型，并设计实现了一个模拟银行账户余额管理的定精度全同态加密计算应用原型程序，以验证方案的正确性和可行性。

# 

# 第五章 实验结果与分析

本章主要对按照前文所述设计的原型系统所产生的实验结果进行分析。

## 5.1 总体情况

实验结果证明，基于整数上单比特同态加密方案设计的定精度实数同态算术运算方案可有效满足于银行云计算场景下的同态加密计算需求，在选择合适参数的情况下，能在允许误差范围内正确的输出预期计算结果。

表6 实验环境

|  |  |  |
| --- | --- | --- |
| 软件环境 | Java版本 | JDK1.8.0\_77 |
| IDE | Eclipse V4.6.2 |
| Web应用服务器 | Tomcat 8.0 |
| 数据库 | MySql 5.0 |
| 硬件环境 | CPU | Intel(R) Core(TM) i5-2320 @3.00GHz |
| 内存 | 8.00G |

前文所述的解决方案及针对银行云计算场景设计的原型系统中，主要关注两个关键指标，一是方案的运行速度，二是输出结果与实际计算结果的误差率，分别由方案的安全参数及阶码决定。在选择较大安全参数的情况下，上述系统以密钥生成为主要内容的初始化过程会运行缓慢，造成这一问题的主要原因是实验环境的硬件性能不足，在实验中可适当减小安全参数，而在实际应用中则可以根据安全需求选用性能较强的设备以提高运行速度，避免出错。在实验硬件条件下，密钥生成后，各输入数据均能在较短的时间内（毫秒级）给出正确计算结果，可满足实际场景要求。阶码主要影响系统输出结果的误差率，理论上值越大输出结果越精确，但由于方案复杂度与输入数据位数相关，值的增大将使得输入数据位数增长，实际应用中，银行可根据系统性能和精度要求选择适当的值。关于部分参数对方案的影响下文会做详细的说明。

## 5.2参数对方案的影响分析

### 5.2.1 安全参数的影响

安全参数用于生成方案的密钥，控制整个方案的安全性，原则上安全参数选取越大，方案越不容易被攻破。考察安全参数对方案的影响，主要考虑在不同的取值下密钥生成时间大小。根据硬件情况，选择[3,9]这一区间进行测试，相关结果如下。由于密钥规模是以的多项式为指数增长的，为便于显示，图中使用了时间的对数显示。

表7 不同安全参数下的密钥生成时间

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
|  | λ=3 | λ=4 | λ=5 | λ=6 | λ=7 |
| 密钥生  成时间t  单位：s | 1.65 | 54 | 2910 | 86434 | 2615732 |

图18 安全参数对密钥生成时间的影响

由表7数据可知，当安全参数逐渐增大时，相应的密钥生成时间以近30倍的等比速度增长，这是由方案决定的。在安全参数增大到一定程度时，将很快超出实验硬件的性能上限，可能致使内存溢出。实际应用场景中，由于硬件性能的提升，相应的密钥生成时间将会缩短，同时由于密钥生成属于系统初始化阶段的工作，一旦完成初始化，后期计算中不再需要该过程。程序正常运行过程中，同态运算的时间复杂度为，不再是安全参数的指数级多项式，对整体个程序运行而言，较长的密钥生成时间并不会直接影响方案的效率，银行在应用时，可根据硬件性能及安全需求指定值。

### 5.2.2 阶码的影响

阶码表示操作数的有效二进制小数位数，用于数据标准化时输入数据的精度和同态运算过程中的小数位数控制，多于位小数部分将被舍弃，会造成加解密及同态运算过程中产生一定误差。一般银行需同态运算的动态数据需要保留2位十进制小数位，为确保数据正确，相应的二进制小数位至少为7。实验的目的是找出由于取值不同所带来的误差规律，以期在实际应用中选择合适的值，尽量减小误差。为验证值的影响，分三种情况进行实验，分别是对单一数据的加解密误差的影响、对两数同态相加误差的影响和对两数同态相乘误差的影响。

#### 5.2.2.1 对单一数据加解密误差的影响

对应不同的值，随机选择一个带有2位十进制小数的实数，按照3.4.1方法进行数据标准化及加密，然后再行解密还原并与原数据作比较，重复运行100000次，计算其正确率，计算方法为（结果相等次数/100000）\*100%。

图19 阶码k对单一数据加解密误差的影响

由图19可以看出，当阶码时，可保证数据100%加解密正确，显然当时加解密正确率可确保为100%，按照这一情况，银行在应用方案时需确保。

#### 5.2.2.2 对两数同态加结果误差的影响

由于加法要求右对齐，故两操作数需取相同的值。对应不同的值，将两随机选取的带2位十进制小数的实数，按照3.4.1方法进行数据标准化及加密，然后按照进行3.4.2.1中的同态加法规则相加，将得到的结果解密并与原明文数据直接相加作比较，重复运行100000次，计算其正确率，计算方法为（结果相等次数/100000）\*100%。

图20 阶码k对两数同态加结果误差的影响

图20显示，由于舍尾造成的误差在两数相加时会叠加，因此的取值要比单一数据加解密时增大，才能确保计算输出正确结果。当阶码时，正确率下降到80%以下，只有当时才能保证完全正确，也即方案在应用中需要确保。

#### 5.2.2.3 对两数同态乘结果误差的影响

对于实数相乘，由于需保留的十进制小数位数始终为2位，因此不管明文数据还是密文数据相乘均需要做舍尾操作，也即始终存在误差，此时不能再以两种操作输出结果是否完全相同作为判断同态乘法输出是否正确的依据，这里采用平均相对误差作为判断指标。相对误差计算方法为（同态乘积-明文乘积）/明文乘积，而平均相对误差则是多次测试数据的平均值，即。

理论分析及实验均证明，输入数据在执行本文所述的数据标准化操作之后，由于舍尾操作会造成操作数本身的误差，且其绝对误差相对固定，但其相对误差则与数据本身大小相关，这将使得乘法结果相对误差对操作数大小比较敏感，因此这里仅按照银行业务中使用乘法的情况做针对性实验，即计算利息的场景，为一较大实数（余额）与较小实数（利息率）相乘。

首先考虑两操作数取相同阶码的情况，即。首先随机选取两个操作数，其取值范围分别是（0,109）和（0,2）。对应不同的值，将两随机选取的带2位十进制小数的实数，按照3.4.1方法进行数据标准化及加密，然后按照进行3.4.2.3中的同态加法规则相乘，并舍弃后个密文，然后将得到的结果解密并与原明文数据直接相乘并取两位有效数字作比较，计算其相对误差，重复运行100000次，计算其平均相对误差。

图21 对称阶码时不同k值对同态乘结果误差的影响

表8 对称阶码时不同k值对同态乘结果误差的影响

|  |  |  |  |
| --- | --- | --- | --- |
| **k=8** | **k=9** | **k=10** | **k=12** |
| -0.0129172 | -0.0064069 | -0.003236 | -0.00083388 |
| **k=14** | **k=16** | **k=18** | **k=20** |
| -0.00020312 | -0.00005018 | -0.00001267 | -0.00000316 |

表8数据及图21趋势显示，两输入数据采用的阶码相同时，其同态乘法计算结果的相对误差随着阶码的增大而以近1/2的比例递减。根据前文所述，至少取才能保证同态加法精度需求，但在乘法中，时计算结果的相对误差率达到0.64%，对于银行场景来说是比较高的，参照银行利率等数据大小，该误差率需要控制在0.1%以下，即取，且越大越精确。由于本文方案是按数据的比特位加密并同态计算的，而的取值直接影响计算数的位数，因此取较大的值会降低同态运算速度，需要根据实际需求确定。

我们注意到乘法计算时无需像加法那样考虑对齐操作，也就意味着两个操作数可以分别选取不同的阶码，因此以下考虑不同阶码情况下平均相对误差的变化规律。首先按照对称阶码误差率实验中的方法选择两随机操作数和，且要求，对应的阶码分别为和，分两次进行测试，先分别固定和，然后改变和的取值，观察平均相对误差率变化规律。

表9 固定k1时k2对同态乘结果误差的影响

|  |  |  |  |
| --- | --- | --- | --- |
| **k1=20、k2=8** | **k1=20、k2=9** | **k1=20、k2=10** | **k1=20、k2=12** |
| -0.0129084 | -0.0064417 | -0.0032418 | -0.00083476 |
| **k1=20、k2=14** | **k1=20、k2=16** | **k1=20、k2=18** | **k1=20、k2=20** |
| -0.00020316 | -0.00005037 | -0.00001263 | -0.00000316 |

图22 固定k1时k2对同态乘结果误差的影响趋势

表9数据及图22表明，固定大实数的阶码改变小实数的阶码，其同态相乘结果的相对误差与表8的数据变化规律相同，随着的逐渐增大以近半的规律逐渐减小，也即此情况下增大可有效降低结果误差。

表10 固定k2时k1对同态乘结果误差的影响

|  |  |  |  |
| --- | --- | --- | --- |
| **k1=8、k2=20** | **k1=9、k2=20** | **k1=10、k2=20** | **k1=12、k2=20** |
| -0.0000033503 | -0.0000033235 | -0.0000032246 | -0.0000031868 |
| **k1=14、k2=20** | **k1=16、k2=20** | **k1=18、k2=20** | **k1=20、k2=20** |
| -0.0000031604 | -0.0000031652 | -0.0000031598 | -0.0000031595 |

图23 固定k2时k1对同态乘结果误差的影响趋势

根据表10数据及图23结果所示，固定小实数的阶码改变大实数的阶码，其同态相乘结果的相对误差基本保持不变，换言之，在保证两操作数的阶码均大于自身的某个阈值后，再行增大大实数的阶码并不能对降低结果的相对误差产生有效贡献，为计算复杂度考虑，需将大实数的阶码控制在其阈值附近。由此可知，在应用同态乘法时，应预估操作数的取值范围，对相对较小的数取较大的阶码以降低运算结果相对误差率，对较大的数取较小的阶码（需大于某阈值，如上述实验中需保证），这样既可以满足计算误差的要求，也能尽量降低计算复杂度。

## 5.3 本章小结

根据前述的解决方案和模型程序，本章对实验结果进行了分析讨论。实验证明，选用适当的参数，可使得该方案的执行效率和误差都在银行许可的范围内，从而满足银行数据的同态加密和算术计算需求。

对于安全参数，由于密钥生成时间随着其增大而迅速增加，实际应用时，银行应根据自身硬件性能，综合考虑安全需求，确定该参数值；对于阶码，银行场景下应用本文所述的同态算术运算方案时，可对数据根据其不同的取值范围和计算需求选取不同的阶码。对于余额这类数值较大且以执行加法为主的数据而言，需要保证；对于利率这类较小且以执行乘法为主的数据，则需要保证。原则上来说阶码取值越大越好，但需要银行根据硬件性能及精度需求综合确定。

# 总结与展望

本文针对银行云计算场景下的数据同态加密运算问题，首先分析了银行数据特点和同态加密技术及应用发展现状，提出了银行云计算中的同态算术运算需求，然后根据经典的DGHV方案提出了改进的整数上单比特明文同态加密方案，最后基于二进制运算规则构建定精度实数同态运算方案。设计的原型系统运行情况表明，本文提出的定精度实数同态运算方案可有效应对银行部分数据在云计算环境下的同态计算问题，解决银行数据安全和云计算需求。具体而言，论文的主要研究成果包括三个方面，分别是：改进的基于整数的单比特明文同态加密方案，基于二进制的实数同态运算规则和银行云计算中同态加密应用模型。

## 研究工作总结

改进的单比特明文同态加密方案

以基于整数的DGHV方案为基础，通过参数的修改和加密算法的适当改变，有效降低密钥长度，提高计算效率，并利用Gentry的重加密技术，结合已有的理论研究，给出了基于二进制进位加法的密文刷新方法，降低了解密算法复杂度，实现了方案的全同态，为同态加密的应用打下基础。

基于二进制的实数同态运算规则

在基于整数的全同态加密方案基础上，结合多位二进制数算术运算方法，提出多比特密文同态算术运算规则，并利用移位等操作，实现了定精度实数的同态算术运算方案，有效应对银行等在云计算中的数据算术运算需求。

银行云计算中同态加密应用模型

根据银行业务逻辑和全同态加密技术的一般应用框架，考虑云计算场景，将同态加密技术应用到银行数据计算中，明确同态加密方案任务分工，确定各算法执行步骤和相关细节，并设计实现了一个原型程序，验证方案的可行性。

## 未来工作展望

本文对银行云计算场景下同态加密技术的应用进行了探讨，但由于个人能力、时间的限制，本文的研究与系统实现还存在许多不足，下一步将继续就相关工作进行研究和探讨。

如何进一步提高同态加密技术的实用性。文献[44]认为，现有及未来的同态加密技术出于安全性考虑，必然存在“fat”（效率不高）、“ugly”（不易理解）等情况，同态加密技术发展多年，基础架构改变不大，目前还未找到一种自然的全同态构造方案，在算法效率上也还有很大的发展空间。本文所述的整数上同态加密计算方案基于经典方案改进而来，但其算法效率依然不够低，在安全参数较大时，密钥生成耗时较长，且增长很快，导致方案执行时间过长，这将限制其应用空间。在现有全同态研究基础上，如何结合实际安全需求，综合考虑各类方案优缺点，选择并裁剪适用的全同态加密方案，进一步提高算法效率，是解决同态加密实用问题的关键。有两个途径，一是改变参数，根据理论研究和实际需求，设置合理的参数，避免算法复杂度过高的情况，二是选择更合适的构建方案，利用更自然的数学结构达到同态计算的目的，并强化其安全性。

如何拓展同态加密技术在银行等行业云计算场景下的应用范围。云计算是未来发展的趋势，当前已有且将来必将有更多的行业投入使用云计算，作为一种可执行密文计算的加密技术，同态加密具有先天优势，也必将在云计算发展中获得极大发展。本文所述的定精度实数同态加密计算方案虽然可满足部分银行数据的同态运算需求，也可在其它需要进行同态算术运算的场景下发挥作用，但适用范围依然较窄。下一步需要继续深入了解银行等行业业务逻辑，探讨同态加密技术的更多应用，以实践应用推动技术的进一步发展。

# 参考文献

1. 第38次中国互联网络发展状况统计报告. CNNIC，2016.
2. Bilge, L., Kirda, E., Kruegel, C., Balduzzi, M., & Antipolis, S. (2011). EXPOSURE : Finding Malicious Domains Using Passive DNS Analysis. Ndss, 1–17.
3. 2016中国互联网网络安全报告, 2017
4. Gentry C. A fully homomorphic encryption scheme[D]. Stanford University, 2009.
5. Gentry C. Fully homomorphic encryption using ideal lattices[C]//STOC. 2009, 9(2009): 169-178.
6. Smart N P, Vercauteren F. Fully homomorphic encryption with relatively small key and ciphertext sizes[C]//International Workshop on Public Key Cryptography. Springer Berlin Heidelberg, 2010: 420-443.
7. Gentry C, Halevi S. Implementing Gentry’s fully-homomorphic encryption scheme[C]//Annual International Conference on the Theory and Applications of Cryptographic Techniques. Springer Berlin Heidelberg, 2011: 129-148.
8. Stehlé D, Steinfeld R. Faster fully homomorphic encryption[C]//International Conference on the Theory and Application of Cryptology and Information Security. Springer Berlin Heidelberg, 2010: 377-394.
9. Smart N P, Vercauteren F. Fully homomorphic encryption with relatively small key and ciphertext sizes[C]//International Workshop on Public Key Cryptography. Springer Berlin Heidelberg, 2010: 420-443.
10. Smart N P, Vercauteren F. Fully homomorphic SIMD operations[J]. Designs, codes and cryptography, 2014: 1-25.
11. Gentry C, Halevi S, Smart N P. Better bootstrapping in fully homomorphic encryption[C]//International Workshop on Public Key Cryptography. Springer Berlin Heidelberg, 2012: 1-16.
12. Van Dijk M, Gentry C, Halevi S, et al. Fully homomorphic encryption over the integers[C]//Annual International Conference on the Theory and Applications of Cryptographic Techniques. Springer Berlin Heidelberg, 2010: 24-43.
13. Coron J S, Mandal A, Naccache D, et al. Fully homomorphic encryption over the integers with shorter public keys[C]//Annual Cryptology Conference. Springer Berlin Heidelberg, 2011: 487-504.
14. Cheon J H, Coron J S, Kim J, et al. Batch fully homomorphic encryption over the integers[C]//Annual International Conference on the Theory and Applications of Cryptographic Techniques. Springer Berlin Heidelberg, 2013: 315-335.
15. Cheon J H, Kim J, Lee M S, et al. CRT-based fully homomorphic encryption over the integers[J]. Information Sciences, 2015, 310(C):149-162.
16. Nuida K, Kurosawa K. (Batch) fully homomorphic encryption over integers for non-binary message spaces[C]//Annual International Conference on the Theory and Applications of Cryptographic Techniques. Springer Berlin Heidelberg, 2015: 537-555.
17. Regev O. On lattices, learning with errors, random linear codes, and cryptography[C]//Proceedings of the thirty-seventh annual ACM symposium on Theory of computing. ACM, 2005: 84-93.
18. Lyubashevsky V, Peikert C, Regev O. On ideal lattices and learning with errors over rings[C]//Annual International Conference on the Theory and Applications of Cryptographic Techniques. Springer Berlin Heidelberg, 2010: 1-23.
19. Brakerski Z, Gentry C, Vaikuntanathan V. (Leveled) fully homomorphic encryption without bootstrapping[C]// Innovations in Theoretical Computer Science Conference. ACM, 2012:309-325.
20. Brakerski Z, Vaikuntanathan V. Fully homomorphic encryption from ring-LWE and security for key dependent messages[C]//Annual cryptology conference. Springer Berlin Heidelberg, 2011: 505-524.
21. Boneh D, Halevi S, Hamburg M, et al. Circular-secure encryption from decision diffie-hellman[C]//Annual International Cryptology Conference. Springer Berlin Heidelberg, 2008: 108-125.
22. Malkin T, Teranishi I, Yung M. Efficient circuit-size independent public key encryption with KDM security[C]//Annual International Conference on the Theory and Applications of Cryptographic Techniques. Springer Berlin Heidelberg, 2011: 507-526.
23. Brakerski Z, Vaikuntanathan V. Efficient fully homomorphic encryption from (standard) LWE[J]. SIAM Journal on Computing, 2014, 43(2): 831-871.
24. Gentry C, Halevi S, Smart N P. Fully homomorphic encryption with polylog overhead[C]//Annual International Conference on the Theory and Applications of Cryptographic Techniques. Springer Berlin Heidelberg, 2012: 465-482.
25. Sharma I. Fully Homomorphic Encryption Scheme with Symmetric Keys[J]. Computer Science, 2013, 2011(4):1-4.
26. Li J, Wang L. Noise-free Symmetric Fully Homomorphic Encryption based on noncommutative rings[J]. IACR Cryptology ePrint Archive, 2015, 2015: 641.
27. Nuida K. A Simple Framework for Noise-Free Construction of Fully Homomorphic Encryption from a Special Class of Non-Commutative Groups[J]. IACR Cryptology ePrint Archive, 2014, 2014: 97.
28. Yagisawa M. Fully Homomorphic Encryption without bootstrapping[J]. IACR Cryptology ePrint Archive, 2015, 2015: 474.
29. Yagisawa M. Fully Homomorphic Encryption on Octonion Ring[J]. IACR Cryptology ePrint Archive, 2015, 2015: 733.
30. Liu D. Practical Fully Homomorphic Encryption without Noise Reduction[J]. IACR Cryptology ePrint Archive, 2015, 2015: 468.
31. Wang Y. Octonion Algebra and Noise-Free Fully Homomorphic Encryption (FHE) Schemes[J]. IACR Cryptology ePrint Archive, 2016, 2016: 68.
32. Cohen J D, Fischer M J. A robust and verifiable cryptographically secure election scheme[C]// Symposium on Foundations of Computer Science. IEEE Computer Society, 1985:372-382.
33. Adida B. Helios: Web-based Open-Audit Voting.[C]// Usenix Security Symposium, July 28-August 1, 2008, San Jose, Ca, Usa. 2008:335-348.
34. Li J, Wang Q, Wang C, et al. Fuzzy keyword search over encrypted data in cloud computing[C]//INFOCOM, 2010 Proceedings IEEE. IEEE, 2010: 1-5.
35. HU H, XU J, REN C, et al. Processing Private Queries Over Untrusted Data Cloud Through Privacy Homomorphism[C]// 2011 IEEE 27th International Conference on Data Engineering (ICDE), 2011: 601-612.
36. 郭璐璐, 许春根. 云存储密文检索方法的研究[J]. 信息网络安全, 2013(9):6-9.
37. Li Z, Zhu X, Lian Y, et al. Constructing secure content-dependent watermarking scheme using homomorphic encryption[C]//Multimedia and Expo, 2007 IEEE International Conference on. IEEE, 2007: 627-630.
38. Chase M, Lauter K, Benaloh J, et al. Patient Controlled Encryption: patient privacy in electronic medical records[J]. Ccsw’09 Acm Cloud Computing Security Workshop, 2009.
39. 李雪松, 彭长根, 李明伟,等. 基于同态加密的统计数据处理[J]. 信息安全与技术, 2015, 6(7).
40. 李京钰, 马春光, 李增鹏. 基于全同态加密的云存储电子病历系统的研究与实现[J]. 保密科学技术, 2015(3):42-45.
41. Naehrig M, Lauter K, Vaikuntanathan V. Can homomorphic encryption be practical?[C]// ACM Cloud Computing Security Workshop, Ccsw 2011, Chicago, Il, Usa, October. 2011:113-124.
42. 刘明洁, 王安. 全同态加密研究动态及其应用概述[J]. 计算机研究与发展, 2014, 51(12):2593-2603.
43. 李顺吉. 隐私同态加密技术在金融云安全上的应用探讨[J]. 中国金融电脑, 2014(11):52-54.
44. Paar C, Pelzl J. Understanding Cryptography: A Textbook for Students and Practitioners[M]. Springer Publishing Company, Incorporated, 2010.
45. Howgrave-Graham N.Approximate integer common divisors[C]//Volume 2146 of Lecture Notes in ComputerScience：CaLC’01.[S.l.]：Springer，2001：51-66.
46. 陈智罡, 王箭, 王梅娟. 整数上的全同态加密分析[C]// 中国密码学会2012年会. 2012.
47. Gjøsteen K, Strand M. Fully homomorphic encryption must be fat or ugly?[J]. IACR Cryptology ePrint Archive, 2016, 2016: 105.

# 攻读硕士学位期间取得的学术成果

(1) 发表的学术论文

[1] Bai L, Zhao Q, Lan Y. A Fully Homomorphic Encryption Scheme over Finite Prime Field[C]// International Conference on Machinery, Materials, Environment, Biotechnology and Computer. 2016.

[2] Bai L, Lan Y. A homomorphic arithmetic scheme on real number with fixed precision. 已录用。

(2) 申请专利

[1] 一种传送和处理数据的方法及系统，申请号201610383960X，发明人：兰雨晴、白立旺、赵琪琦。

# 致谢

在毕业论文将成之际，我谨向在硕士学习期间帮助和支持我的各级组织、老师和朋友们表达诚挚的谢意。

首先，我要感谢的是部队，培养了我多年，并给了我这次求学的机会，圆了我的硕士研究生梦，让我开拓了眼界，学到了很多知识，同时也增加了社会阅历，个人能力得到了很大的提高。

其次，感谢导师兰雨晴。我的学习期间，老师在各个方面都给予了无微不至的关怀，不仅关心关注我的学习和学术研究，在生活以及未来规划上也都不遗余力，使我在这两年多的研究生生活中倍感温暖。尤其是毕业论文撰写期间，老师不仅在论文选题和开题上给予了悉心的指导，在论文写作及实验中也经常督促我并提出改进意见，经常在百忙中过问研究进度，并在文章结构，语言表述等方面提出了很多有益的建议。没有老师的耐心指导和宝贵意见，这次的毕业论文将不可能如此顺利完成。在此我怀着感恩的心情，向兰老师致以最衷心的感谢！

再次，我要感谢实验室师兄师弟。实验室同学的积极协作配合，确保了我的论文的进度。能够在团结和谐的G308实验室与同学们一起学习、一起研究探讨，我深感荣幸，也使得我的研究生生活丰富多彩，你们在我的人生轨迹中抹下了浓墨重彩的一笔。在此，我衷心的感谢夏庆新、韩涛、王洋、伯为翰等师兄师姐，感谢解刚、运明纯同学，感谢于思民、王铖成、房大梁、任冠华等师弟，尤其感谢于思民同学，感谢你们的倾情相助，使我顺利地取得了当前的研究成果。

最后，感谢我的父母和妻子，是他们的支持使我能够安心学业，不为家事分心，尤其是妻子，一个人承担所有责任，照顾一家老小，从无怨言，对此我甚为感激！