doi:10.12068/j.issn.1005－3026.2024.04.001

基于LU分解的安全外包求解线性代数方程组方法

冯 达，周福才，吴淇毓，李 鲍
（东北大学 软件学院，辽宁 沈阳 110169）

摘 要：由于现有协议的安全性为基于某种安全假设的计算安全，依赖于敌手的计算能力，因此，本文针对恶意敌手模型，使用矩阵伪装技术对方程的系数矩阵进行隐藏，结合矩阵的LU分解（lower-upper decomposition）算法，提出一种新的信息论安全外包求解线性代数方程组（information-theoretically secure outsourcing of linear algebraic equations，ITS-OutsLAE）方法．与之前的研究相比，在保持计算和通信复杂度与现有最优方案保持一致的同时，首次将方程组唯一解的安全性提升至信息论安全（完美保密）．给出了形式化的安全性证明，并通过理论分析和实验证明了所提方法的实用性．

关 键 词：线性代数方程组；信息论安全；安全外包；LU分解；恶意敌手模型
中图分类号：TP 311.1 **文献标志码**：A **文章编号**：1005-3026(2024)04-0457-08

Secure Outsourcing Method for Solving Linear Algebraic Equations Based on LU Decomposition

FENG Da, *ZHOU Fu-cai*, *WU Qi-yu*, *LI Bao*
(School of Software, Northeastern University, Shenyang 110169, China. Corresponding author: ZHOU Fu-cai, E-mail: fczhou@mail.neu.edu.cn)

Abstract: All existing protocols are computationally secure, each of which is based on a certain security assumption and depends on the computational power of the adversary. This paper uses the matrix masking technique to hide the coefficient matrix. Combined with the LU (lower-upper) decomposition algorithm, a new information-theoretically secure outsourcing of linear algebraic equations method is proposed in malicious adversarial model. Compared with the previous protocol, the security of the unique solution is improved to information-theoretic security (perfect secrecy) for the first time, without sacrificing the complexity of computation and communication. A formal security proof is provided, and the practicality is proved theoretically and experimentally.

Key words: linear algebraic equations; information-theoretic security; secure outsourcing; LU decomposition; malicious adversarial model

求解线性代数方程组$\boldsymbol{Ax}=\boldsymbol{b}$是线性代数学科的核心问题[1]，其具有非常广泛的应用场景[2]，如椭圆或抛物线型偏微分方程（partial differential equation, PDE）的离散化和线性化，以及电路、电网、化工过程、经济模型等一些非PDE系统的设计与分析过程．大量学者深入研究了传统的求解线性代数方程组方法，代表性的方法如高斯消去法[3]、克拉默法则[4]等已被广泛应用多年．

许多现实问题会对应超大规模且系数非常密集的线性代数方程组．当客户端计算能力有限时，可以考虑将问题外包至云计算服务器[5-7]或者分布式代理网络[8-11]．然而，近年来随着对数据隐私的愈发重视，使用传统的方法会面临严重的安全问题：当客户端将明文数据$\boldsymbol{A}$，$\boldsymbol{b}$直接交给云计算服务器或分布式代理网络时，会泄露所有的方程组系数$\boldsymbol{A}$，$\boldsymbol{b}$以及方程组的唯一解

收稿日期：2022-11-14
基金项目：国家自然科学基金资助项目（62072090，62202090，62173101）；辽宁省自然科学基金医工交叉联合基金资助项目（2022-YGJC-24）；中央高校基本科研业务费专项资金资助项目（N2217009）．
作者简介：冯 达（1993-），男，辽宁沈阳人，东北大学博士研究生；周福才（1964-），男，辽宁沈阳人，东北大学教授，博士生导师．

x^*,这对于涉密问题或极具价值的工业或金融数据来说是不可接受的.Gennaro等[12]首先正式提出了可验证计算(verifiable computation)的概念和模型,并将全同态加密技术和混淆电路技术相结合,提出了一种安全外包计算的通用机制.近年来,一些学者利用同态加密技术或矩阵伪装技术解决了该问题[13-14].由于全同态加密算法的效率较慢,通常被认为是不可接受的.近些年,许多学者利用基于线性变换的矩阵加密技术构建矩阵相关计算的安全外包方案.这些方案使用稀疏矩阵作为加密某些明文矩阵的密钥,该技术也被称为基于稀疏矩阵的伪装(sparse matrix-based masking,SMM)技术.随着基于稀疏矩阵的矩阵伪装算法[15-17]的兴起,基于部分同态加密算法的方案也由于效率相对较低逐渐被遗弃.现有的基于矩阵伪装技术的外包求解线性代数方程组方法可在保证较高计算效率的同时,实现近似或严格的唯密文攻击安全(security against ciphertext-only attacks,COA Security).此安全性属于基于某种安全假设的计算安全(computational security),依赖于敌手的计算能力,也就是说当敌手计算能力足够强大时仍可以破坏加密算法的安全性.

本文研究求解大规模线性代数方程组的安全外包问题.针对方程组唯一解的安全性依赖于敌手计算能力这一问题,提出一种新型信息论安全外包求解线性代数方程组计算协议,在不牺牲计算效率的同时,使得方程组$\boldsymbol{Ax}=\boldsymbol{b}$的唯一解$\boldsymbol{x}^*$的安全性达到信息论安全,即$\boldsymbol{x}^*$对云计算服务器是完美保密的,且该安全性不依赖于云计算服务器的计算能力.具体来说,本文的主要贡献在如下三个方面:

1)在完全恶意敌手模型中,利用线性变换伪装方程组的系数矩阵,结合LU分解算法,设计了一种新的信息论安全外包求解线性代数方程组的计算协议.该协议允许客户端在不泄露数据隐私的条件下将复杂的求解线性代数方程组计算问题外包至云计算服务器,同时实现了结果的可验证性,即客户端可以验证云计算服务器是否篡改了协商好的计算过程.

2)将LU分解与传统的安全外包求解方法相结合,实现了方程组系数矩阵$\boldsymbol{b}$对云计算服务器完美保密的同时仍可使客户端得到正确的方程唯一解,为线性代数方程组$\boldsymbol{Ax}=\boldsymbol{b}$的唯一解$\boldsymbol{x}^*$的安全性达到信息论安全提供了可能.最终给出了$\boldsymbol{x}^*$是信息论安全的严格证明过程.

3)本文将唯一解$\boldsymbol{x}^*$的安全性由之前的唯密文攻击安全提升至信息论安全的同时,保证了两方外包计算协议的计算复杂度和通信复杂度均为$O(n^2)$,与现有最优方案保持一致.

1 相关知识

1.1 矩阵的LU分解

LU分解属于矩阵分解的一种,是高斯消去(Gaussian elimination)算法的矩阵形式.令$\boldsymbol{A}$为非奇异方阵,LU分解可以将$\boldsymbol{A}$分解为一个下三角矩阵和一个上三角矩阵的乘积,即$\boldsymbol{A}=\boldsymbol{L}\cdot\boldsymbol{U}$. LU分解是最常见的求解线性代数方程组的方法,一般思路是将$\boldsymbol{A}$分解为$\boldsymbol{L}\cdot\boldsymbol{U}$,将方程转化为$\boldsymbol{L}\cdot\boldsymbol{U}\cdot\boldsymbol{x}=\boldsymbol{b}$,通过$\boldsymbol{L}\cdot\boldsymbol{y}=\boldsymbol{b}$对$\boldsymbol{y}$求解,再通过$\boldsymbol{U}\cdot\boldsymbol{x}=\boldsymbol{y}$对$\boldsymbol{x}$求解.

考虑到数值的稳定性,LU分解在处理某些矩阵时会出现条件数很少但计算误差很大的情况.要解决该问题需要在高斯消元过程中加入一些行置换操作,具体如下.

定义1 LUP分解(LU decomposition with partial pivoting).LUP分解是仅添加一些行置换操作的LU分解:

$$\boldsymbol{P}_{\mathrm{LU}}\boldsymbol{A}=\boldsymbol{L}\cdot\boldsymbol{U}.$$

其中:$\boldsymbol{L}$为下三角矩阵;$\boldsymbol{U}$为上三角矩阵;$\boldsymbol{P}_{\mathrm{LU}}$为置换矩阵,将它右乘$\boldsymbol{A}$以重新排列其行.

LUP分解是LU分解中最常用的方法,适用于任何非奇异矩阵,且该方法在实际应用中是数值稳定的.

1.2 基于稀疏矩阵的矩阵伪装算法

直观地,对矩阵做适当的线性变换可以对矩阵中的信息进行伪装,而线性变换相当于左乘或右乘1个矩阵.设$\boldsymbol{A}$,$\boldsymbol{B}$为一般的密集矩阵,则计算$\boldsymbol{A}\cdot\boldsymbol{B}$的复杂度为$O(n^3)$;而若$\boldsymbol{A}$为稀疏矩阵,$\boldsymbol{B}$为密集矩阵,计算$\boldsymbol{A}\cdot\boldsymbol{B}$和$\boldsymbol{B}\cdot\boldsymbol{A}$的复杂度可降至$O(n^2)$.利用该特性,左乘或右乘稀疏矩阵可高效地对1个矩阵进行伪装,与之对应的恢复过程即左乘或右乘对应稀疏矩阵的逆矩阵.

最近,大量基于稀疏矩阵的相关方法在安全计算领域被提出,这些方法虽不尽相同,但在效率方面非常接近.本文选用具有代表性的文献[15]中的方法,给定安全参数λ,以及矩阵规模n,客户端随机选取2个$n\times n$大小的对角矩阵$\boldsymbol{D}_1,\boldsymbol{D}_2$和1个$n\times n$置换矩阵$\boldsymbol{P}_\pi$,令密钥$k=(\boldsymbol{P}_\pi,\boldsymbol{D}_1,\boldsymbol{D}_2)$.矩阵伪装算法以线性代数方程组$\boldsymbol{Ax}=\boldsymbol{b}$的系数矩阵$\boldsymbol{A}$,$\boldsymbol{b}$和密钥$k=(\boldsymbol{P}_\pi,\boldsymbol{D}_1,\boldsymbol{D}_2)$为输入,输出结果为

伪装后的系数矩阵$\boldsymbol{A}'$,$\boldsymbol{b}'$.算法具体描述见表1.

表1　矩阵伪装算法

Table 1　Matrix masking algorithm

Algorithm: $\text{Masking}(k, \boldsymbol{A}, \boldsymbol{b}) \rightarrow (\boldsymbol{A}', \boldsymbol{b}')$
Input: $k, \boldsymbol{A}, \boldsymbol{b}$
Output: $\boldsymbol{A}', \boldsymbol{b}'$
1　Compute $\boldsymbol{A}' = \boldsymbol{D}_1 \cdot (\boldsymbol{P}_\pi \cdot \boldsymbol{A}) \cdot \boldsymbol{D}_2$
2　Compute $\boldsymbol{b}' = \boldsymbol{D}_1 \cdot (\boldsymbol{P}_\pi \cdot \boldsymbol{b})$
3　Output $\boldsymbol{A}', \boldsymbol{b}'$

综合文献[15]和文献[17]中的安全性分析,有以下结论成立.

引理1　在不考虑矩阵密度泄露信息的前提下,上述矩阵伪装算法输出的$\boldsymbol{A}'$,$\boldsymbol{b}'$对原系数矩阵$\boldsymbol{A}$,$\boldsymbol{b}$的信息泄露是可忽略的.

2　安全模型及定义

2.1　安全模型

安全外包求解线性代数方程组协议涉及两方实体,分别为客户端和云计算服务器,如图1所示.客户端根据安全参数生成密钥,并对方程组系数进行伪装,生成1个伪装版本的线性代数方程组,交由云计算服务器进行求解.云计算服务器在进行适当计算后将结果返回给客户端.最后客户端对返回结果进行验证,并由此结果求得原方程组的唯一解.

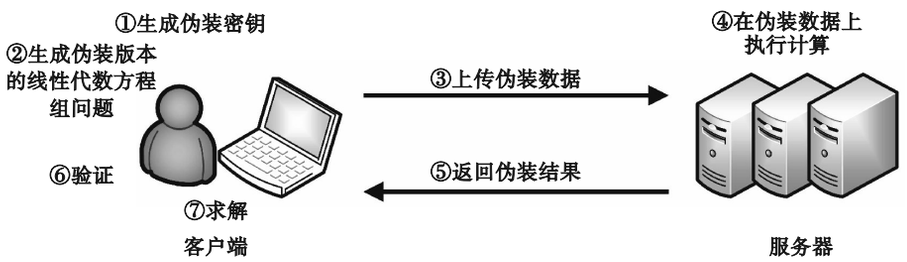

图1　安全计算协议架构

Fig. 1　Architecture of secure computing protocol

根据敌手能力不同,安全外包计算中通常分为两种敌手模型——半诚实(诚实但好奇)敌手模型和恶意敌手模型.在半诚实敌手模型中,敌手正确执行协议规定的计算过程,但会试图从它能得到的所有信息中推测并获取机密信息;在恶意敌手模型中,敌手具有更强的能力,可以任意篡改协议的计算过程,因此在该模型中,计算结果的可验证性尤为重要.

本文针对恶意敌手模型,设计安全外包求解线性代数方程组协议.客户端被视为诚实的一方,正确执行所有的计算并期望得到方程组的唯一解;云计算服务器被视为潜在的恶意敌手,可能希望得到客户端的机密信息,也可能偏离协议规定的计算过程,故意返回无法区分但不可用的结果(虚假的唯一解).简言之,云计算服务器可能既好奇又不诚实.

2.2　形式化定义

本文提出的安全外包求解线性代数方程组协议ITS-OutsLAE包含5个算法:密钥生成算法、问题生成算法、服务器计算算法、验证算法和求解算法,即ITS-OutsLAE=

(KeyGen, ProbGen, Compute, Verify, Solve).

1)密钥生成算法$\text{KeyGen}(\lambda, n) \rightarrow k$.该算法由客户端执行,给定1个安全参数$\lambda$和矩阵规模$n$作为输入,该随机密钥生成算法得到密钥$k$,由客户端保密存储.

2)问题生成算法$\text{ProbGen}(k, \boldsymbol{A}, \boldsymbol{b}) \rightarrow (\boldsymbol{A}', \boldsymbol{b}')$.该算法由客户端执行,给定密钥$k$和系数矩阵$\boldsymbol{A}$,$\boldsymbol{b}$作为输入,问题生成算法输出伪装后的系数矩阵$\boldsymbol{A}'$,$\boldsymbol{b}'$,客户端仅将$\boldsymbol{A}'$发送给服务器进行计算.

3)服务器计算算法$\text{Compute}(\boldsymbol{A}') \rightarrow (\boldsymbol{L}, \boldsymbol{U}, \boldsymbol{P}_{\text{LU}})$.该算法由服务器执行,给定矩阵$\boldsymbol{A}'$作为输入,输出对$\boldsymbol{A}'$的LUP分解结果$\boldsymbol{L}, \boldsymbol{U}, \boldsymbol{P}_{\text{LU}}$,并返回至客户端.

4)验证算法$\text{Verify}(\boldsymbol{L}, \boldsymbol{U}, \boldsymbol{P}_{\text{LU}}, \boldsymbol{A}') \rightarrow 1 \text{ or } 0$.该算法由客户端执行,以矩阵$\boldsymbol{L}, \boldsymbol{U}, \boldsymbol{P}_{\text{LU}}, \boldsymbol{A}'$为输入,判断服务器是否正确执行计算,输出1(接受)或0(拒绝).

5)求解算法$\text{Solve}(\boldsymbol{L}, \boldsymbol{U}, \boldsymbol{P}_{\text{LU}}, \boldsymbol{A}', \boldsymbol{b}') \rightarrow \boldsymbol{x}^*$.若验证算法输出为"1",则客户端继续执行该算法,以矩阵$\boldsymbol{L}, \boldsymbol{U}, \boldsymbol{P}_{\text{LU}}, \boldsymbol{A}', \boldsymbol{b}'$为输入,输出原线性代数方程组$\boldsymbol{Ax} = \boldsymbol{b}$的唯一解$\boldsymbol{x}^*$.

下面给出安全性定义.

定义2　(COA安全).对给定实验

$\text{Exp}_{\text{A}}^{\text{COA}}[\text{ITS} - \text{OutsLAE}, \lambda, n]$:

$((\boldsymbol{A}_0,\boldsymbol{b}_0),(\boldsymbol{A}_1,\boldsymbol{b}_1))\leftarrow \mathrm{A}$;
$b\leftarrow\{0,1\}$;
$k\leftarrow \mathrm{KeyGen}(\lambda,n)$;
$(\boldsymbol{A}_b',\boldsymbol{b}_b')\leftarrow \mathrm{ProbGen}(\boldsymbol{A}_b,\boldsymbol{b}_b,k)$;
$b'\leftarrow \mathrm{A}(\boldsymbol{A}_b')$.

定义敌手A对方程组系数矩阵的隐私性的优势为 $\mathrm{Adv}_{\mathrm{A}}^{\mathrm{COA-coef}}(\mathrm{LAE},\lambda)=\left|P[b=b']-\frac{1}{2}\right|$,若对任意多项式时间(probabilistic polynomial time, PPT)敌手,满足 $\mathrm{Adv}_{\mathrm{A}}^{\mathrm{COA-coef}}(\mathrm{LAE},\lambda)\leqslant \mathrm{negl}(\lambda)$,则称系数矩阵是COA安全的.

定义3 (完美保密)设方程组的解满足概率分布X.对任意概率分布X以及"解-系数矩阵"对$(\boldsymbol{x}_0,\boldsymbol{A}_0)$,有 $P[\boldsymbol{x}=\boldsymbol{x}_0|\boldsymbol{A}=\boldsymbol{A}_0]=P[\boldsymbol{x}=\boldsymbol{x}_0]$成立,则称方程组的解(对云服务器)是完美保密的.

2.3 协议设计目标

本文旨在设计一种新的安全外包求解线性代数方程组协议,在不泄露系数矩阵和方程组唯一解隐私的前提下,将超大规模的复杂方程组问题的大部分计算任务交给云计算服务器进行计算,使得云计算服务器能够有效地帮助客户端进行求解.具体来说,该协议应当满足以下特性:

1) 正确性.如果服务器和客户端正确执行计算协议,则客户端应当得到正确的方程组$\boldsymbol{Ax}=\boldsymbol{b}$的唯一解$\boldsymbol{x}^*$.

2) 隐私性.在整个计算协议过程中,服务器可能获得的隐私相关信息是可忽略的.计算过程泄露的有关方程组的系数矩阵$\boldsymbol{A}$,$\boldsymbol{b}$以及方程组唯一解$\boldsymbol{x}^*$的信息应当是可忽略的.特别地,本文要求$\boldsymbol{x}^*$对服务器完美保密.

3) 可验证性.诚实的服务器返回的计算结果一定会通过客户端的验证;反之,任何错误的或伪造的结果都不能以不可忽略的概率通过验证.

4) 效率.在安全外包计算场景中,要求客户端的计算复杂度尽可能低,且一定要低于客户端直接求解原问题的计算复杂度.在本文协议中,客户端的计算复杂度与现有最优方案保持一致.

3 ITS-OutsLAE协议

3.1 协议构建

Shannon[18]证明了一个密码系统若要达到完美保密,必须满足密钥空间大于等于明文空间,即$|K|\geqslant|M|$.考虑到计算效率,该条件在一般密码系统中是很难做到的.然而,通过观察可以发现,在线性代数方程组的安全外包问题中,方程组的解$\boldsymbol{x}$与系数矩阵$\boldsymbol{b}$均是n维向量.若将$\boldsymbol{x}\in X$看作需要保密的"明文",将$\boldsymbol{b}\in B$对云服务器隐藏,其中B是系数矩阵空间,并看作"密钥",就可实现$(|B|=|\mathbf{R}^n|)\geqslant(|X|=|\mathbf{R}^n|)$.进一步地,将求解问题转化为对伪装矩阵$\boldsymbol{A}'$的LU分解过程,并将其交给云计算服务器进行计算,可将客户端的求解复杂度降至$O(n^2)$,与已有的最优方案效率保持一致;同时无须将$\boldsymbol{b}'$交给云计算服务器,实现对系数矩阵$\boldsymbol{b}$的完全隐藏,从而使方程组唯一解的安全性达到完美保密.

具体地,结合表1矩阵伪装算法,原方程$\boldsymbol{Ax}=\boldsymbol{b}$可转化为

$$\boldsymbol{D}_1\cdot(\boldsymbol{P}_\pi\boldsymbol{A})\cdot\boldsymbol{D}_2(\boldsymbol{D}_2^{-1}\boldsymbol{x})=\boldsymbol{D}_1\cdot\boldsymbol{P}_\pi\cdot\boldsymbol{b}.$$

设$\boldsymbol{y}=\boldsymbol{D}_2^{-1}\boldsymbol{x}$,则有

$$\boldsymbol{A}'\boldsymbol{y}=\boldsymbol{b}'.$$

由LUP分解$\boldsymbol{P}_{\mathrm{LU}}\boldsymbol{A}'=\boldsymbol{L}\cdot\boldsymbol{U}$可知,

$$\boldsymbol{L}\cdot\boldsymbol{U}\cdot\boldsymbol{y}=\boldsymbol{P}_{\mathrm{LU}}\cdot\boldsymbol{b}'.$$

设

$$\boldsymbol{z}=\boldsymbol{U}\cdot\boldsymbol{y}, \tag{1}$$

有

$$\boldsymbol{L}\cdot\boldsymbol{z}=\boldsymbol{P}_{\mathrm{LU}}\cdot\boldsymbol{b}'. \tag{2}$$

所以,在客户端得知$\boldsymbol{L},\boldsymbol{U},\boldsymbol{P}_{\mathrm{LU}}$的值时,只需先对方程(2)求解得到$\boldsymbol{z}=\boldsymbol{z}^*$,再对方程(1)求解得到$\boldsymbol{y}=\boldsymbol{y}^*$,最后令$\boldsymbol{x}^*=\boldsymbol{D}_2\boldsymbol{y}^*$.

对于上述外包过程,验证方法有两种:其一是在服务器返回LUP分解结果后,立刻验证分解结果是否正确,若正确则继续对方程进行求解;若不正确则认定服务器计算错误或恶意篡改计算过程或伪造结果;其二是根据服务器返回的LUP分解结果,首先对方程组进行求解,得到唯一解之后再进行验证,判断唯一解是否正确.这两种方法均使协议实现可验证性,即服务器的恶意行为会被以接近于1的概率检测出来.不同之处在于第二种方法验证过程在求解过程之后,若服务器计算错误或恶意篡改计算过程或伪造结果,则求解过程是没有意义的.为了避免不必要的计算,本协议选择第二种验证方法,即在求解前先对LUP分解结果进行验证.

3.2 ITS-OutsLAE协议详细描述

基于以上讨论,本节对ITS-OutsLAE协议中的5个算法——密钥生成算法、问题生成算法、服务器计算算法、验证算法和求解算法——给出详细描述.

1）密钥生成算法 KeyGen$(\lambda, n)\to k$:该算法由客户端执行,给定1个安全参数λ和矩阵规模n作为输入,算法利用伪随机数生成器(pseudo-random number generator, PRG)生成2个$n\times n$大小的随机对角矩阵$\boldsymbol{D}_1, \boldsymbol{D}_2$和1个$n\times n$大小的随机置换矩阵$\boldsymbol{P}_\pi$,令密钥$k=(\boldsymbol{P}_\pi, \boldsymbol{D}_1, \boldsymbol{D}_2)$,算法输出密钥$k$,用于伪装系数矩阵以及恢复方程的唯一解,由客户端秘密保管.

2）问题生成算法 ProbGen$(k, \boldsymbol{A}, \boldsymbol{b})\to(\boldsymbol{A}', \boldsymbol{b}')$:该算法由客户端执行,以密钥生成算法输出的密钥$k$和原线性代数方程组的系数矩阵$\boldsymbol{A}$,$\boldsymbol{b}$作为输入,调用矩阵伪装算法 Masking$(k, \boldsymbol{A}, \boldsymbol{b})$,输出伪装后的系数矩阵$\boldsymbol{A}'$,$\boldsymbol{b}'$,其中$\boldsymbol{A}'=\boldsymbol{D}_1\cdot(\boldsymbol{P}_\pi\cdot\boldsymbol{A})\cdot\boldsymbol{D}_2$,$\boldsymbol{b}'=\boldsymbol{D}_1\cdot(\boldsymbol{P}_\pi\cdot\boldsymbol{b})$. 客户端仅将$\boldsymbol{A}'$发送给服务器进行计算,$\boldsymbol{b}'$由客户端秘密保管,用于恢复明文解.

3）服务器计算算法 Compute$(\boldsymbol{A}')\to(\boldsymbol{L}, \boldsymbol{U}, \boldsymbol{P}_{\mathrm{LU}})$:该算法由服务器执行,以伪装后的系数矩阵$\boldsymbol{A}'$作为输入,对$\boldsymbol{A}'$执行LUP分解算法,输出对$\boldsymbol{A}'$的LUP分解结果$\boldsymbol{L}, \boldsymbol{U}, \boldsymbol{P}_{\mathrm{LU}}$,并返回至客户端.对$\boldsymbol{A}'$的LUP分解算法步骤如表2所示.

表2　对$\boldsymbol{A}'$的LUP分解算法

Table 2　LUP decomposition algorithm for $\boldsymbol{A}'$

Algorithm: LUP$(\boldsymbol{A}')\to(\boldsymbol{L}, \boldsymbol{U}, \boldsymbol{P}_{\mathrm{LU}})$

Input: $\boldsymbol{A}'$
Output: $\boldsymbol{L}, \boldsymbol{U}, \boldsymbol{P}_{\mathrm{LU}}$
Initialize $\boldsymbol{U}=\boldsymbol{A}', \boldsymbol{L}=\boldsymbol{I}, \boldsymbol{P}_{\mathrm{LU}}=\boldsymbol{I}$
for $j=1:n-1$ **do**
　Select i ($\geqslant j$) that maximizes $|u_{ij}|$
　Interchange rows of $\boldsymbol{U}$: $u_{(j,j:n)}\leftrightarrow u_{(i,j:n)}$
　Interchange rows of $\boldsymbol{L}$: $l_{(j,1:(j-1))}\leftrightarrow l_{(i,1:(j-1))}$
　Interchange rows of $\boldsymbol{P}_{\mathrm{LU}}$: $p_{(j,:)}\leftrightarrow p_{(i,:)}$
　for $i=j+1$ **do**
　　$l_{ij}=u_{ij}/u_{jj}$
　　for $k=j:n$ **do**
　　$u_{ik}=u_{ik}-l_{ij}u_{jk}$
　　end for
　end for
end for
Output $\boldsymbol{L}, \boldsymbol{U}, \boldsymbol{P}_{\mathrm{LU}}$

4）验证算法 Verify$(\boldsymbol{L}, \boldsymbol{U}, \boldsymbol{P}_{\mathrm{LU}}, \boldsymbol{A}')\to 1$ or 0:该算法由客户端执行,以LUP分解计算结果$\boldsymbol{L}, \boldsymbol{U}, \boldsymbol{P}_{\mathrm{LU}}$和原伪装后的系数矩阵$\boldsymbol{A}'$为输入,计算$\boldsymbol{A}''=\boldsymbol{P}_{\mathrm{LU}}\boldsymbol{A}'$,对矩阵$\boldsymbol{A}''$每行任取2个位置,验证$\boldsymbol{A}''=\boldsymbol{L}\cdot\boldsymbol{U}$是否成立,若成立则输出1(接受),表示该结果正确,否则输出0(拒绝),表示服务器计算错误或恶意篡改计算过程或伪造结果.验证算法步骤如表3所示.

表3　验证算法

Table 3　Verification algorithm

Algorithm: Verify$(\boldsymbol{L}, \boldsymbol{U}, \boldsymbol{P}_{\mathrm{LU}}, \boldsymbol{A}')\to 1$ or 0

Input: $\boldsymbol{L}, \boldsymbol{U}, \boldsymbol{P}_{\mathrm{LU}}, \boldsymbol{A}'$
Output: 1 or 0
Initialize $c=1$
Compute $\boldsymbol{A}''=\boldsymbol{P}_{\mathrm{LU}}\boldsymbol{A}'$
for $i=1:n$ **do**
　Randomly pick s indices $j_1\neq j_2\in\{1,\cdots,n\}$
　if $(\sum_{k=1}^{n}l_{ik}u_{kj_1}\neq a''_{ij_1}$ or $\sum_{k=1}^{n}l_{ik}u_{kj_2}\neq a''_{ij_2})$
　　$c=0$
　　break
　end if
end for
Output c

为了进一步提高验证算法的准确性,减少伪造结果通过验证的概率,可要求第4步选取的n组(2n个)索引遍历1~n.

5）求解算法 Solve$(\boldsymbol{L}, \boldsymbol{U}, \boldsymbol{P}_{\mathrm{LU}}, \boldsymbol{A}', \boldsymbol{b}')\to \boldsymbol{x}^*$:若验证算法输出为“1”,则客户端继续执行该算法,以LUP分解算法的输出矩阵$\boldsymbol{L}, \boldsymbol{U}, \boldsymbol{P}_{\mathrm{LU}}$,伪装后的系数矩阵$\boldsymbol{A}'$和向量$\boldsymbol{b}'$为输入,首先对方程组$\boldsymbol{L}\cdot\boldsymbol{z}=\boldsymbol{P}_{\mathrm{LU}}\cdot\boldsymbol{b}'$求解得$\boldsymbol{z}=\boldsymbol{z}^*$,再对方程组$\boldsymbol{U}\cdot\boldsymbol{y}=\boldsymbol{z}$求解得$\boldsymbol{y}=\boldsymbol{y}^*$,最后令$\boldsymbol{x}^*=\boldsymbol{D}_2\boldsymbol{y}^*$.输出原线性代数方程组$\boldsymbol{A}\boldsymbol{x}=\boldsymbol{b}$的唯一解$\boldsymbol{x}^*$.

4　理论分析及实验

4.1　理论分析

4.1.1　协议正确性

定理1　若服务器和客户端正确地执行安全计算协议ITS-OutsLAE,则$\boldsymbol{A}\boldsymbol{x}^*=\boldsymbol{b}$.

证明　假设协议中所有算法都得到正确执行,则有

$$\boldsymbol{x}^*=\boldsymbol{D}_2\boldsymbol{y}^*,$$
$$\boldsymbol{z}=\boldsymbol{z}^*.$$

由方程(1)得

$$\boldsymbol{U}\cdot\boldsymbol{D}_2^{-1}\boldsymbol{x}^*=\boldsymbol{z}^*.$$

由方程(2)得

$$\boldsymbol{L}\cdot\boldsymbol{U}\cdot\boldsymbol{D}_2^{-1}\boldsymbol{x}^*=\boldsymbol{P}_{\mathrm{LU}}\cdot\boldsymbol{b}'.$$

又$\boldsymbol{P}_{\mathrm{LU}}\boldsymbol{A}'=\boldsymbol{L}\cdot\boldsymbol{U}$,则

$$\boldsymbol{A}'\boldsymbol{D}_2^{-1}\boldsymbol{x}^*=\boldsymbol{b}'.$$

再由伪装算法可知

$$\boldsymbol{D}_1\cdot(\boldsymbol{P}_\pi\cdot\boldsymbol{A})\cdot\boldsymbol{D}_2\boldsymbol{D}_2^{-1}\boldsymbol{x}^*=\boldsymbol{D}_1\cdot(\boldsymbol{P}_\pi\cdot\boldsymbol{b}),$$

所以

$$\boldsymbol{A}\boldsymbol{x}^*=\boldsymbol{b}.$$

4.1.2 数据隐私性

定理2 设线性代数方程组$\boldsymbol{Ax}=\boldsymbol{b}$的唯一解$\boldsymbol{x}_i\in X$,对服从任意概率分布的$X$以及任意"解-系数矩阵"对$(\boldsymbol{x}_0,\boldsymbol{A}_0)$,有

$$P[\boldsymbol{x}=\boldsymbol{x}_0|\boldsymbol{A}=\boldsymbol{A}_0]=P[\boldsymbol{x}=\boldsymbol{x}_0].$$

证明 对任意$\boldsymbol{A}_0\xleftarrow{R}\mathbf{R}^{n\times n}$,满足$|\boldsymbol{A}_0|\neq 0$,$\boldsymbol{b}_0\xleftarrow{R}\mathbf{R}^n$,其对应的解为$\boldsymbol{x}_0\in\mathbf{R}^n$.由$\boldsymbol{A}_0'=\boldsymbol{D}_1(\boldsymbol{P}\cdot\boldsymbol{A}_0)\cdot\boldsymbol{D}_2$,$\boldsymbol{A}_0=\boldsymbol{P}^{-1}(\boldsymbol{D}_1^{-1}\cdot\boldsymbol{A}_0'\cdot\boldsymbol{D}_2^{-1})$可知,$\boldsymbol{A}'=\boldsymbol{A}_0'$是$\boldsymbol{A}=\boldsymbol{A}_0$的充要条件.于是有

$$P[\boldsymbol{x}=\boldsymbol{x}_0|\boldsymbol{A}'=\boldsymbol{A}_0']=P[\boldsymbol{x}=\boldsymbol{x}_0|\boldsymbol{A}=\boldsymbol{A}_0].$$

由贝叶斯定理可知

$$P[\boldsymbol{x}=\boldsymbol{x}_0|\boldsymbol{A}=\boldsymbol{A}_0]=\frac{P[\boldsymbol{x}=\boldsymbol{x}_0,\boldsymbol{A}=\boldsymbol{A}_0]}{P[\boldsymbol{A}=\boldsymbol{A}_0]}=\frac{P[\boldsymbol{x}=\boldsymbol{x}_0]\cdot P[\boldsymbol{A}=\boldsymbol{A}_0|\boldsymbol{x}=\boldsymbol{x}_0]}{\sum_{\boldsymbol{x}_i\in X}(P[\boldsymbol{x}=\boldsymbol{x}_i]\cdot P[\boldsymbol{A}=\boldsymbol{A}_0|\boldsymbol{x}=\boldsymbol{x}_i])}.$$

因为

$$P[\boldsymbol{A}=\boldsymbol{A}_0|\boldsymbol{x}=\boldsymbol{x}_0]=P[\boldsymbol{b}=\boldsymbol{b}_0]=\frac{1}{|\mathbf{R}^n|},$$

同理

$$P[\boldsymbol{A}=\boldsymbol{A}_0|\boldsymbol{x}=\boldsymbol{x}_i]=\frac{1}{|\mathbf{R}^n|},$$

所以有

$$P[\boldsymbol{x}=\boldsymbol{x}_0|\boldsymbol{A}'=\boldsymbol{A}_0']=\frac{P[\boldsymbol{x}=\boldsymbol{x}_0]\cdot\frac{1}{|\mathbf{R}^n|}}{\sum_{\boldsymbol{x}_i\in X}\left(P[\boldsymbol{x}=\boldsymbol{x}_i]\cdot\frac{1}{|\mathbf{R}^n|}\right)}=\frac{P[\boldsymbol{x}=\boldsymbol{x}_0]}{\sum_{\boldsymbol{x}_i\in X}P[\boldsymbol{x}=\boldsymbol{x}_i]}=P[\boldsymbol{x}=\boldsymbol{x}_0].$$

4.1.3 可验证性

由LUP分解算法可知,诚实的服务器返回的计算结果一定会通过客户端的验证;反之,由验证算法可知,验证过程遍历了矩阵$\boldsymbol{L},\boldsymbol{U}$的每一个位置,且验证每一行的两个位置时,索引$j_1$和$j_2$是随机选取的,服务器伪造结果几乎是不可能完成的任务.定理3给出了ITS-OutsLAE协议中返回结果的不可伪造性.

定理3 ITS-OutsLAE协议中服务器端的计算结果是不可伪造的,即

$$P[\mathrm{Verify}(\boldsymbol{L}',\boldsymbol{U}')\to 1\,|\,(\boldsymbol{L}',\boldsymbol{U}')\neq(\boldsymbol{L},\boldsymbol{U})]\leqslant\mathrm{negl}(\lambda),$$

其中$\mathrm{negl}(\cdot)$为可忽略函数.

证明 假设诚实地对$\boldsymbol{A}'$执行LUP分解算法得到的结果为$(\boldsymbol{L},\boldsymbol{U})$,服务器伪造了结果$(\boldsymbol{L}',\boldsymbol{U}')\neq(\boldsymbol{L},\boldsymbol{U})$.设概率事件$F$为"存在索引$(i',j')$使得$l'_{i'j'}\neq l_{i'j'}$或$u'_{i'j'}\neq u_{i'j'}$".若$F$成立,则必可以找到验证项$\Sigma_1=\sum_{k=1}^{n}l'_{i'k}u'_{kj}$以及$\Sigma_2=\sum_{k=1}^{n}l'_{ik}u'_{kj'}$.于是有

$$P[\Sigma_1=a''_{i'j}\text{and}\,\Sigma_2=a''_{ij'}\,|\,F]=\frac{1}{2^{\lambda}}.$$

所以

$$P[\mathrm{Verify}(\boldsymbol{L}',\boldsymbol{U}',\boldsymbol{P}_{\mathrm{LU}},\boldsymbol{A}')\to 1\,|\,F]\leqslant\frac{1}{2^{\lambda}}.$$

又因为

$$P[F\,|\,(\boldsymbol{L}',\boldsymbol{U}')\neq(\boldsymbol{L},\boldsymbol{U})]=1,$$

所以

$$P[\mathrm{Verify}(\boldsymbol{L}',\boldsymbol{U}',\boldsymbol{P}_{\mathrm{LU}},\boldsymbol{A}')\to 1\,|\,(\boldsymbol{L}',\boldsymbol{U}')\neq(\boldsymbol{L},\boldsymbol{U})]\leqslant\frac{1}{2^{\lambda}}.$$

4.1.4 复杂度分析

本节从计算复杂度和通信复杂度两方面对ITS-OutsLAE协议进行分析.

本文提出的ITS-OutsLAE协议通过将复杂的大规模矩阵LU分解计算任务交给云服务器来完成,实现了将客户端计算复杂度由$O(n^3)$降低至$O(n^2)$.该协议中密钥生成算法的计算任务为调用$O(n)$次伪随机数生成器;问题生成算法需执行$O(n^2)$次乘法;验证算法同样需执行$O(n^2)$次乘法.为达到方程组唯一解的完美保密性质,本文在隐藏系数矩阵$\boldsymbol{b}$的同时要求客户端在求解算法阶段进行一些额外的求解计算过程,即先对方程(2)进行求解,再对方程(1)进行求解.由于方程(1)和(2)中的系数矩阵均为三角矩阵,所以该过程的计算复杂度为$O(n^2)$.因此,ITS-OutsLAE协议在客户端的计算复杂度为$O(n^2)$.此外,服务器端的计算任务为执行LUP分解算法,其复杂度为$O(n^3)$.可以看出,原本需要客户端完成的复杂度为$O(n^3)$的计算已由服务器协助降低至$O(n^2)$.

此外,ITS-OutsLAE协议中的问题生成算法仅仅是对矩阵问题作线性变换处理,并未改变矩阵的形状和大小,这使得客户端和服务器间的通信开销保持在$O(n^2)$不变.需要注意的是,直观上完美保密似乎相较于COA安全需要更长的密钥,然而本方案是通过完全隐藏系数矩阵$\boldsymbol{b}$(如3.1节所述)来实现唯一解的完美保密.也就是说,ITS-OutsLAE协议并未增加密钥长度,密钥仍为2个$n\times n$大小的稀疏矩阵,当安全参数$\lambda=1\,024$时,密钥大小为$1\,024n^2$ b,由客户端持有.简言之,本方案提高安全性的同时并未增加密钥长度,通信开销与现有最优方案保持一致.

综上所述,ITS-OutsLAE协议在客户端的计算复杂度为$O(n^2)$,协议总体通信复杂度为$O(n^2)$,均与现有的最优方案的性能保持一致.

4.1.5 方案对比

本节将ITS-OutsLAE协议与已有的代表性方案(文献[11]、文献[15])从外包类型、计算复

杂度、通信复杂度、安全性和可验证性进行对比,结果见表4.

表4　方案对比

Table 4　Comparisons with prior protocols

方法	外包类型	计算复杂度	通信复杂度	安全性	可验证性
文献[11]	分布式代理节点	$O(n^2)$	$O(n^2)$	COA	√
文献[15]	云计算服务器	$O(n^2)$	$O(n^2)$	COA	√
ITS-OutsLAE	云计算服务器	$O(n^2)$	$O(n^2)$	COA + ITS	√

可以看出本文提出的ITS-OutsLAE协议在安全性方面实现了突破,首次将方程组唯一解的安全性提升至信息论安全(完美保密),并且在效率(计算复杂度、通信复杂度)、可验证性方面与已有最优方案保持一致.

4.2　实验结果

通过4.1节的理论分析,说明ITS-OutsLAE协议相较于其他方案在安全性上有优势.由于基于稀疏矩阵伪装技术的安全外包协议利用稀疏矩阵乘法的性质降低了客户端的计算复杂度,导致此类协议在客户端的计算复杂度均为$O(n^2)$,所以本节省去了不同方案在算法效率上的横向对比,而仅通过测试ITS-OutsLAE协议中各算法的效率,表明ITS-OutsLAE协议的实用性.

下面对该协议中的各阶段算法效率进行测试,测试工具为MATLAB R2021b,所有测试均在2.60 GHz Core (TM) i7-9750H CPU 和 24 GB RAM环境下实现.针对ITS-OutsLAE协议,首先测试客户端的4个算法——密钥生成算法、问题生成算法、验证算法和求解算法,测试每个算法对应不同方程组规模n所需的执行时间,实验结果见图2.

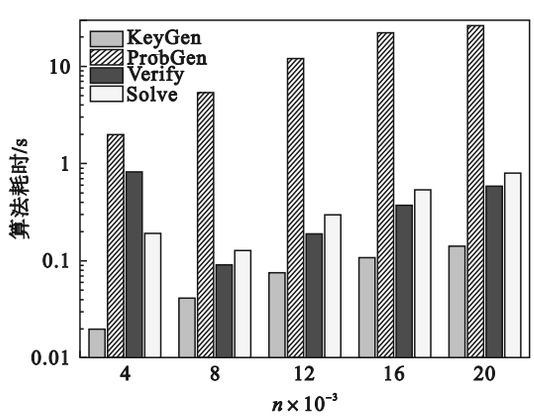

图2　客户端算法效率测试

Fig. 2　Algorithm efficiency test on the client side

由图2可以看出,需要在客户端执行的密钥生成算法、问题生成算法、验证算法和求解算法耗时均较短,即使矩阵规模扩大至20 000,客户端也仅需二十几秒的执行时间即可完成全部协议计算过程.然后对服务器端的服务器计算算法进行测试,实验结果见图3.由图3可以看出,当n=2 000时,在当前实验环境需执行约90 s,这对于计算能力强大的云计算服务器来说可以更高效地执行服务器计算算法.

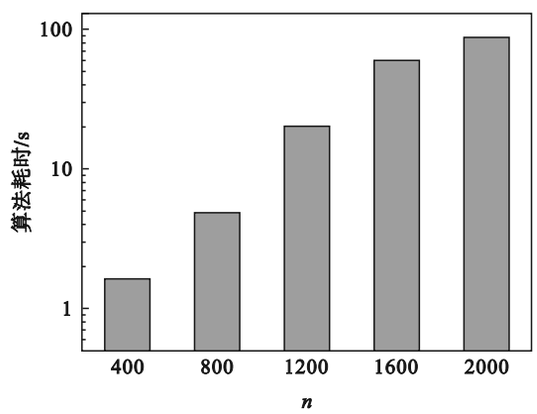

图3　服务器端算法效率测试

Fig. 3　Algorithm efficiency test on the server side

5　结　　语

本文研究了大规模线性代数方程组的安全外包求解问题.针对现有方案安全性依赖于敌手计算能力问题,提出了一种新的安全外包求解线性代数方程组方法.该方法在传统的基于线性变换的矩阵伪装算法基础上,结合矩阵的LU分解算法,减少客户端需要交给服务器的信息,在保证计算协议高效的同时,实现了对方程组唯一解的完全隐藏,即方程组唯一解的安全性达到了信息论安全(完美保密).总体来说,本研究提出的ITS-OutsLAE协议的安全性高于已有方案,对于安全外包求解大规模线性代数方程组问题有一定理论创新和实际应用价值.

参考文献:

[1] Strang G. Introduction to linear algebra [M]. 5th ed. Wellesley: Wellesley-Cambridge Press, 2021.

[2] Benzi M. Preconditioning techniques for large linear systems: a survey [J]. *Journal of Computational Physics*, 2002, 182(2): 418-477.

(下转第506页)