Computer Science and Application 计算机科学与应用, 2021, 11(2), 314-325
Published Online February 2021 in Hans. http://www.hanspub.org/journal/csa
https://doi.org/10.12677/csa.2021.112032

Hans汉斯

基于云平台的密文数据离群点检测算法

陈政波[1]，程　清[1]，于鹏飞[2,3]

[1]国网浙江省电力有限公司信息通信分公司，浙江　杭州
[2]全球能源互联网研究院有限公司，江苏　南京
[3]信息网络安全国网重点实验室，江苏　南京
Email: yupengfei@geiri.sgcc.com.cn

收稿日期：2021年1月17日；录用日期：2021年2月12日；发布日期：2021年2月19日

摘　要

将数据和计算外包给云服务器为大规模数据存储和查询处理是一种经济高效的方法。但是由于安全和隐私问题，我们需要在使用云服务器计算过程中保护诸如医疗记录这种敏感数据。一种方法是将加密过的数据外包给云服务器，并让云服务器仅对加密数据执行查询处理，这样云服务器就不会获得有关数据、查询和查询结果的任何知识。在不进行数据云解密的情况下进行加密数据的查询任务是非常具有挑战性的。我们设计的任务是在加密数据条件下云服务器进行安全K近邻(KNN)查询方法进行离群点检测，即用户发送一条加密查询给具有加密数据集的云服务器，并得到该数据是否为离群点的信息。我们首先提出了一个简单方案，并证明该方案并非是完全安全的。之后我们提出了具有更高安全性的安全方案，它能够有效保护用户查询数据、服务器数据集和数据访问模式的机密性。

关键词

安全，加密数据，离群点检测

Outlier Detection Algorithm for Cryptographic Data Based on Cloud Platform

Zhengbo Chen[1], Qing Cheng[1], Pengfei Yu[2,3]

[1]State Grid Zhejiang Electric Power Corporation Information & Telecommunication Branch, Hangzhou Zhejiang
[2]Global Energy Interconnection Research Institute Co., Ltd., Nanjing Jiangsu
[3]State Grid Key Laboratory of Information & Network Security, Nanjing Jiangsu
Email: yupengfei@geiri.sgcc.com.cn

Received: Jan. 17th, 2021; accepted: Feb. 12th, 2021; published: Feb. 19th, 2021

文章引用: 陈政波, 程清, 于鹏飞. 基于云平台的密文数据离群点检测算法[J]. 计算机科学与应用, 2021, 11(2): 314-325. DOI: 10.12677/csa.2021.112032

Abstract

Outsourcing data and computing to cloud servers for large-scale data storage and query processing is an economical and efficient method. However, due to security and privacy issues, we need to protect sensitive data such as medical records in the process of using cloud server computing. One approach is to outsource the encrypted data to a cloud server and have the cloud server perform only query processing on the encrypted data, so that the cloud server does not gain any knowledge of the data, queries, and query results. The query task of encrypting data without data cloud decryption is very challenging. The task we designed is to detect outliers with the cloud server's secure k-nearest neighbor (KNN) query method under the condition of encrypted data, that is, the user sends an encrypted query to the cloud server with encrypted data set and obtains the information whether the data is an outlier or not. We first propose a simple scheme and prove that the scheme is not completely safe. Then we proposed a security scheme with higher security, which can effectively protect the confidentiality of user query data, server data set and data access mode.

Keywords

Security, Encrypted Data, Outlier Detection

1. 引言

云计算是近年来兴起的计算方式，它具有低成本，高灵活性，高效率的优势，吸引了工业界和许多研究人员的目光，将数据和计算外包给云服务器已经成为了很多组织/企业的选择。通用的云计算模型[1] [2]中，用户将自己的数据库放到云服务器上，同时进行数据的查询和管理操作。这样做对用户来说，一方面不用自己建数据库服务器并进行数据运算，降低了数据管理和服务器维护开销，节省了运算开销，减少了运营成本；另一方面，被外包到云服务器上的数据可能存在泄露的危险，数据的查询运算操作也可能会泄露用户隐私，这为云计算带来了安全上的问题。

为保护外包给云服务器的数据不会因为云服务器和黑客攻击而被泄露，一种简单方法是在数据外包给云服务器之前就将数据进行加密处理[3]。在云服务器的安全查询过程中，需要能够保证：1) 外包给云服务器数据的保密性；2) 用户进行查询数据的保密性；3) 计算过程中隐藏数据访问模式。许多文献提出了与加密数据查询处理相关的技术，例如范围查询[4] [5] [6]，以及聚合查询[7] [8]等等。但是，这些技术对于解决云服务器上加密数据的 KNN 查询来说要么不适合使用，要么效率不能满足用户的需求。

本文在一种经典的 SKNN 协议[9]基础上，进行修改以提高安全性，使服务器能够在对加密数据进行 KNN 算法检测离群点的同时，既能够保护数据机密性又能够保护查询数据隐私。在此协议中，数据拥有者一旦将数据加密并提交服务器，就不会再获取后续信息且不参与后续任何计算。特别地，该协议满足以下要求：

1) 数据机密性——数据库数据集 T 或任何中间结果的内容不应向服务器公开。

2) 查询隐私——输入查询数据 Q 不应该被泄露给服务器。

3) 正确性——输出应该只透露给客户端用户。并且，除了此信息没有其他信息透露给 Bob。

4) 用户上的计算开销尽可能低——将加密的查询记录发送到服务器后，我们的协议在客户端上的计算开销比现有的工作[1]低。

5) 隐藏的数据访问模式——对数据的访问模式，例如查询数据 Q 的 k 近邻对应的记录，不应该向数据拥有者和服务器公开(以防止任何推理攻击)。

应该强调的是，服务器在我们的协议中看到的中间结果要么是新生成的随机加密，要么是随机数。因此，服务器不知道哪些数据记录对应于查询数据 Q 的 k 近邻。此外，在将加密的查询记录发送到服务器之后，用户尽可能涉及最少计算过程使得用户的成本尽可能低。

2. 基础知识

在本文中，隐私安全与协议执行过程中披露的信息量密切相关。有许多方法可以用来定义可披露的信息。为了使数据隐私最大化或者说信息披露最小化，我们采用了之前文献中提出的安全多方计算(SMC)定义，该定义最早是由 Yao 通过 Millionaires 问题引入的，并且他们针对该问题开发了一个可证明安全的解决方案。在本文中，我们假设服务器是半诚实的，即诚实且好奇，具体来说，半诚实的服务器使用正确的操作流程遵守协议的规则，但是之后可以自由地使用在协议执行期间得到的中间内容来危害安全性比如数据机密性。一般来说，半诚实模型下的安全协议比恶意对手模型下的安全协议更有效，而文献中提出的实际 SMC 协议几乎所有都属于半诚实模型下的安全协议。具体来说，我们设置两个半诚实的参与方 P1 和 P2 进行交互计算。其中 P1 具有加密数据库 $E_{pk}(T)$ 和公钥 pk(public key)，P2 具有公钥 pk 和私钥 sk (secret key)，我们采用 paillier 同态加密算法作为加密方式，同时采用一系列安全子协议来构建我们的加密协议。

2.1. Paillier 加密

paillier 加密基于复合剩余类的难解性，是 1999 年 paillier 发明的概率公钥加密系统。该加密算法是一种同态加密，满足加法和数乘同态。其同态性质可描述为：

$$(1)\quad E_{pk}(a+b) \leftarrow E_{pk}(a) * E_{pk}(b) \bmod N^2$$

$$(2)\quad E_{pk}(a*b) \leftarrow E_{pk}(a)^b \bmod N^2$$

2.2. 安全乘法(SM)协议

P1 包含 $E_{pk}(a)$、$E_{pk}(b)$，且 P1 和 P2 不知道 a 和 b 的具体数值，经过 P1、P2 协同计算后将 $E_{pk}(a*b)$ 输出到 P1，并且在这个过程中，P1 和 P2 没有得到关于 a 和 b 的信息。输出 $E_{pk}(a*b)$ 只有 P1 知道。安全乘法(SM)协议的基本思想基于以下性质，对于任意给定的 $a,b \in Z_N$，有：

$$a*b = (a+r_a)*(b+r_b) - a*r_b - b*r_a - r_a*r_b$$

2.3. 安全平方欧几里得距离协议

P1 包含 $E_{pk}(X)$，$E_{pk}(Y)$，其与 P2 安全地协同计算向量 X 和 Y 之间欧氏距离的平方加密。这里 X 和 Y 是 m 维向量，其中：

$$(1)\quad E_{pk}(X) = \langle E_{pk}(X_1), \cdots, E_{pk}(X_m) \rangle$$

$$(2)\quad E_{pk}(Y) = \langle E_{pk}(Y_1), \cdots, E_{pk}(Y_m) \rangle$$

最后，输出 $E_{pk}\left(\left|X-Y\right|^2\right)$ 只有服务器 P1 知道。安全的欧几里德距离平方协议 SSED 的基本思想如下式所示：

$$\left|X-Y\right|^2=\sum_{i=1}^{m}\left(x_i-y_i\right)^2$$

2.4. 安全位分解(SBD)协议

P1 包含 $E_{pk}\left(z\right)$，其与 P2 安全地协同计算 z 的各个比特位的加密，其中 $0\le Z<2^l$。输出 $\left[z\right]=\left\langle E_{pk}\left(z_1\right),\cdots,E_{pk}\left(z_l\right)\right\rangle$ 只有 P1 知道。这里 z_1 和 z_l 分别表示整数 z 的最高有效位和最低有效位。

2.5. 安全最小(SMIN)协议

P1 包含([u]，[v])，服务器 P2 包含私钥 sk，两个服务器协同安全计算 u 和 v 之间小的那一个数的各个比特位的加密，即输出为[min(u,v)]，且该结果只有 P1 知道。在这个协议中，P1 和 P2 无法得到关于 u 和 v 的信息。

2.6. 安全最小 n 数字(SMINn)协议

P1 包含 n 个加密向量$\left(\left[d_1\right],\cdots,\left[d_n\right]\right)$，P2 包含私钥 sk，这里$\left[d_i\right]=\left\langle E_{pk}\left(d_{i,1}\right),\cdots,E_{pk}\left(d_{i,l}\right)\right\rangle$，其中 $d_{i,1}$ 和 $d_{i,l}$ 分别为整数 d_i 的最高和最低有效位，且$1\le i\le n$。P1 和 P2 协同计算输出$\left[\min\left(d_1,\cdots,d_n\right)\right]$。最后，$\left[\min\left(d_1,\cdots,d_n\right)\right]$仅为 P1 所知晓。并且在 SMIN_n 协议过程中，P1 和 P2 无法获取有关 d_i 的信息。

2.7. 安全位或(SBOR)协议

P1 包含$\left(E_{pk}\left(o_1\right),E_{pk}\left(o_2\right)\right)$，其与 P2 安全地协同计算 $E_{pk}\left(o_1\vee o_2\right)$，其中 o_1 和 o_2 是两个二进制数值。输出 $E_{pk}\left(o_1\vee o_2\right)$ 只有 P1 知道。

3. 方案设计

在本节中，我们首先介绍一个基本的 SkNN 离群点检测协议，并说明为什么这样一个简单的解决方案不安全。然后，我们讨论第二种方法，一个完全安全的 SkNN 离群点检测协议。这两个协议都是使用第三节中说明的安全子协议作为构建模块构建的。

我们假设数据拥有者 Alice 的数据库由 n 条记录组成，用 $T=\left\langle t_1,\cdots,t_n\right\rangle$ 表示，每条记录有 m 个属性，其中 $t_{i,j}$ 表示记录 t_i 的第 j 个属性值。Alice 首先对数据库属性进行 Paillier 加密，即计算 $E_{pk}(t_{i,j})$，范围为 $1\le i\le n$ 和 $1\le j\le m$，加密后的数据库记为 $E_{pk}(T)$。我们假设 Alice 将 $E_{pk}(T)$以及未来的查询处理服务发送给服务器。同时，我们假设所有的属性值及其欧氏距离都在$\left[0,2^l\right]$范围内。在我们使用的协议中，我们假设存在两个非合谋半诚实的服务提供者，分别用 C1 和 C2 表示，它们共同组成一个联合服务器。在此设置下，Alice 将加密数据库 $E_{pk}(T)$发送给服务器 C1，将密钥 sk 发送给服务器 C2。我们使用的协议的目标是以高效和安全的方式检索最接近用户查询的 k 条记录(KNN)并以此为基础进行离群点的检测。简单地说，假设一个授权用户 Bob，他想基于服务器 C1 中的 $E_{pk}(T)$，确认自己的输入 $Q=\left\langle q_1,\cdots,q_n\right\rangle$ 是否为离群点。Bob 最初将他的查询 Q(以加密的形式)发送到 C1。在此之后，C1 和 C2 使用一组子协议来安全地(以加密的形式)检索输入查询数据 Q 的 k 近邻数据集，之后 Bob 来确认自己的查询数据是否为离群点。我们强调，只有 Bob 才能获取到 Q 的 k 近邻的准确数据，具体协议设置如图 1 所示。

3.1. 基本安全协议

下面给出了 SkNNb 协议的主要步骤。Bob 首先加密查询数据 Q 的属性，即计算：

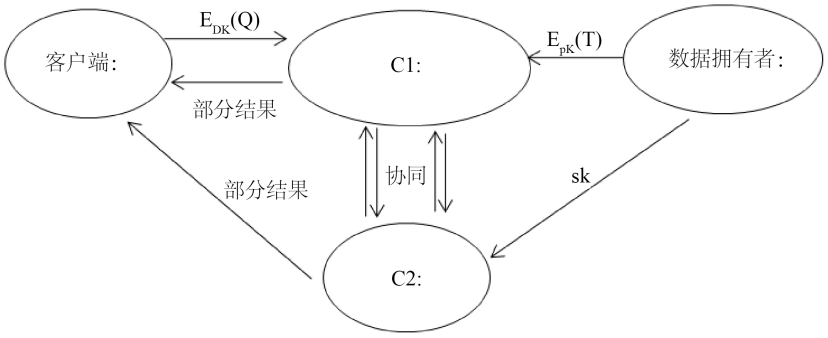

Figure 1. Parties setup of the agreement
图 1. 协议各方设置

$$E_{pk}(Q)=\left\langle E_{pk}(q_1),\cdots,E_{pk}(q_m)\right\rangle$$

然后将其发送到服务器 C1。从 Bob 处接收到 $E_{pk}(Q)$后，持有输入($E_{pk}(Q)$、$E_{pk}(t_i)$)的 C1 和持有私钥 sk 的 C2 共同参与到安全欧几里得距离 SSED 协议中，其中：

$$E_{pk}(t_i)=\left\langle E_{pk}(t_{i,1}),\cdots,E_{pk}(t_{i,m})\right\rangle$$

其中 i 范围为$1\le i\le n$，这一步的输出为 Q 与 t_i 欧氏距离的平方的 paillier 加密，即 $d_i=\left|Q-t_i\right|^2$，用 $E_{pk}(d_i)$ 表示。如前所述，$E_{pk}(d_i)$只有服务器 C1 知道。我们强调加密向量之间的精确欧氏距离计算是很困难的，因为它涉及到平方根的计算。之后，服务器 C1 发送$\left\{\left\langle 1,E_{pk}(d_1)\right\rangle,\cdots,\left\langle n,E_{pk}(d_n)\right\rangle\right\}$到 C2，其中条目$\left\langle i,E_{pk}(d_i)\right\rangle$对应数据记录$t_i,i$范围为$1\le i\le n$。之后服务器C2对每个条目进行解密，得到$d_i=D_{sk}\left(E_{pk}(d_i)\right)$。然后，C2 生成一个索引列表$\delta=\left\langle i_1,\cdots,i_k\right\rangle$，使得$\left\langle d_{i_1},\cdots,d_{i_k}\right\rangle$为$\left\langle d_1,\cdots,d_n\right\rangle$中 K 个最短距离的表示。在这之后，C2 发送 δ 到 C1。C1 收到 δ 之后，进行如下操作：

1) 选择加密记录$E_{pk}\left(t_{i_1}\right),\cdots,E_{pk}\left(t_{i_k}\right)$作为查询数据 Q 的 K 最近邻记录，并随机化它们的属性。具体而言，C1 计算$\gamma_{j,h}=E_{pk}\left(t_{i_j,h}\right)*E_{pk}\left(r_{j,h}\right)$，$1\le j\le k$ 且$1\le h\le m$。这里$r_{j,h}$是 Z_N 范围内的一个随机数，$t_{i_j,h}$表示列 h 中t_{i_j}属性值。然后 C1 将$\gamma_{j,h}$发送到 C2，$r_{j,h}$发送给 Bob，其中$1\le j\le k$ 且$1\le h\le m$。

2) 收到$\gamma_{j,h}$之后，服务器 C2 解密计算$\gamma'_{j,h}=D_{sk}\left(\gamma_{j,h}\right)$，并将其发送给 Bob。注意，因为 C1 之前将$\gamma_{j,h}$的加密数值随机化，所以$\gamma'_{j,h}$仍然是 Z_N 范围内的随机数。

3) Bob 收到$r_{j,h}$和$\gamma'_{j,h}$之后，使用$t'_{j,h}=\gamma'_{j,h}-r_{j,h}\bmod N$来计算查询数据 Q 的 K 近邻数据值，其中$1\le j\le k$ 且$1\le h\le m$，之后 Bob 根据得到的 K 个数值进行离群点的检测，我们采用距离方法顺序进行检测，即从这组数据的第一个开始进行数值欧式距离的判定，一旦存在某一个数值 t_i 与 Q 之间的欧式距离大于预设的距离 D，$1\le i\le k$，则说明在数据群 T 中与查询数据 Q 的欧式距离小于 D 的点不足 k 个，判定 Q 为离群点，否则，Q 不是离群点。

3.2. 完全安全协议

上述的 SkNNb 协议向服务器 C1 和 C2 泄露了数据访问模式。也就是说，对于任何给定的查询数据 Q，服务器 C1 和 C2 都知道哪些数据相当于 Q 的 k 近邻记录。然而，在例如医疗数据等隐私敏感的应用条件中，泄露此类信息可能是不可接受的。为此，我们使用了一个完全安全的协议，用 SkNN_m 表示(其中 m 代表最大安全)来检索 Q 的 k 近邻。

完全安全协议 $\text{SkNN}_\text{m}(E_{pk}(T),Q)$

条件：C1 持有加密数据库 $E_{pk}(T)$，随机置换函数 π，C2 持有私钥 sk，客户端 Bob 提出查询 Q1。

客户端 Bob 将 $E_{pk}(Q)=\langle E_{pk}(q_1),\cdots,E_{pk}(q_m)\rangle$ 发送至 C12.C1 和 C2：

(1): C1 自客户端 Bob 处接收 $E_{pk}(Q)$

(2): for i=1 to n:

$E_{pk}(d_i)=\mathrm{SSED}(E_{pk}(Q),E_{pk}(t_i))$

$[d_i]=\langle E_{pk}(d_{i,1}),\cdots,E_{pk}(d_{i,l})\rangle=\mathrm{SBD}(E_{pk}(d_i))$

for s=1 to k do:

(a): C1 和 C2:

$[d_{\min}]=\mathrm{SMIN}_n([d_1],\cdots,[d_n])$

(b): C1:

$E_{pk}(d_{\min})=\prod_{\gamma=0}^{l-1}E_{pk}(d_{\min,\gamma+1})^{2^{l-\gamma-1}}$

$E_{pk}(d_i)=\prod_{\gamma=0}^{l-1}E_{pk}(d_{i,\gamma+1})^{2^{l-\gamma-1}},s\neq 1,1\le i\le n$

for i=1 to n:

$\tau_i=E_{pk}(d_{\min})*E_{pk}(d_i)^{N-1}$

$\tau_i'=\tau_i^{r_i}=E_{pk}(r_i*(d_{\min}-d_i))$

$\beta=\pi(\tau')$，将 β 发送至 C2

(c): C2:

自 C1 处接收 β

$\beta_i'=D_{sk}(\beta_i),1\le i\le n$

for $1\le i\le n$:

if $\beta_i'=0$: $U_i=E_{pk}(1)$

else $U_i=E_{pk}(0)$

将 U 发送至 C1

(d): C1:

自 C2 处接收 U 并计算 $V=\pi^{-1}(U)$

$V_{i,j}'=\mathrm{SM}(V_i,E_{pk}(t_{i,j})),1\le i\le n,1\le j\le m$

$E_{pk}(t_{i,j}')=\prod_{i=1}^{n}V_{i,j}',1\le j\le m$

$E_{pk}(t_s')=\langle E_{pk}(t_{s,1}'),\cdots,E_{pk}(t_{s,m}')\rangle$

(e): C1 和 C2:

$E_{pk}(d_{i,\gamma})=\mathrm{SBOR}(V_i,E_{pk}(d_{i,\gamma})),1\le i\le n,1\le\gamma\le l$ 剩余算法步骤与 $\mathrm{SkNN_b}$ 的 4~6 步相同

我们使用的 $\mathrm{SkNN_m}$ 协议的主要步骤如上述算法所示。首先，Bob 发送他的属性加密查询数据 Q，即 $E_{pk}(Q)=\langle E_{pk}(q_1),\cdots,E_{pk}(q_m)\rangle$ 至服务器 C1。持有($E_{pk}(Q)$、$E_{pk}(t_i)$)的 C1 和持有私钥 sk 的 C2 共同参与到安全欧几里得距离 SSED 协议中。这一步的输出 $E_{pk}(d_i)=E_{pk}(|Q-t_i|^2)$ 只有 C1 知道，i 范围为 $1\le i\le n$。之后，持有 $E_{pk}(d_i)$的服务器 C1 与持有私钥 sk 的服务器 C2 使用安全位分解 SBD 协议安全计算 d_i 的各个比特位的加密。注意，这一步的输出 $[d_i]=\langle E_{pk}(d_{i,1}),\cdots,E_{pk}(d_{i,l})\rangle$ 只有 C1 知道，其中 $d_{i,1}$ 和 $d_{i,l}$ 分别是 d_i 的最高有效位和最低有效位。其中 $0\le d_i<2^l$，且 $1\le i\le n$。

之后，服务器 C1 和 C2 以迭代的方式计算最接近查询数据 Q 的 k 近邻加密记录。更具体地说，他们在第一个迭代中计算 $E_{pk}(t_1')$，在第二个迭代中计算 $E_{pk}(t_2')$，以此类推。其中 t_s' 表示距离 Q 第 s 近的数

据记录，s 范围为$1 \le s \le k$，在 k 次迭代结束时，只有服务器 C1 知道迭代结果$\langle E_{pk}(t'_1),\cdots,E_{pk}(t'_k)\rangle$。首先，在第一次迭代中，服务器 C1 和 C2 使用安全最小 n 个数字协议 SMIN_n 计算 $d_1,\cdots,d_n$ 中最小值的各个比特位的加密.我们令$[d_{\min}]$为 $d_1,\cdots,d_n$ 中最小值，该加密数值只有服务器 C1 才能知道。现在，服务器 C1 执行以下操作：

1) 使用下面的公式计算加密数值 $d_{\min}$：

$$E_{pk}(d_{\min})=\prod_{\gamma=0}^{l-1}E_{pk}(d_{\min,\gamma+1})^{2^{l-\gamma-1}}=E_{pk}(d_{\min,1}*2^{l-1}+\cdots+d_{\min,l})$$

其中 $d_{\min,1}$ 和 $d_{\min,l}$ 为 $d_{\min}$ 的最高有效位和最低有效位。

2) 计算加密数据 $d_{\min}$ 和每一个距离数据 d_i 的差别，即在$1 \le i \le n$范围内计算

$$\tau_i=E_{pk}(d_{\min})*E_{pk}(d_i)^{N-1}=E_{pk}(d_{\min}-d_i)$$

3) 将 τ_i 随机化，具体表示为 $\tau'_i=\tau_i^{r_i}=E_{pk}(r_i*(d_{\min}-d_i))$，$r_i$ 为 Z_N 范围内的随机数。注意，τ'_i 是一个加密过的 0 或一个随机数，其中$1 \le i \le n$。同时，使用随机置换函数 π(该函数只有 C1 知道)将其乱序，得到 $\beta=\pi(\tau')$ 并发送到 C2。

接到 β 之后，服务器 C2 将其解密获得 $\beta'_i=D_{sk}(\beta_i)$，其中$1 \le i \le n$。在这之后，C2 计算一个长度为 n 的加密向量 U，如果 $\beta'_i=0$ 那么 $U_i=E_{pk}(1)$，否则 $U_i=E_{pk}(0)$。在这里，我们假设 β 中只有一个元素为零，其他都是随机数。以此类推，U 中只有一个元素为加密之后的 1，其余元素为加密之后的 0。然而，我们强调，如果 β' 中含有不止一个元素为 0，那么服务器 C2 可以随机选择其中一个下标索引 i，将 $E_{pk}(1)$ 分配给相应的 U_i，并将其余的元素设为 $E_{pk}(0)$。然后，服务器 C2 把加密向量 U 发送到服务器 C1。收到 U 之后，C1 对其执行逆序，获得 $V=\pi^{-1}(U)$。注意，V 中只有一个元素为 $E_{pk}(1)$，其余的都是 $E_{pk}(0)$。此时，如果 $V_i=E_{pk}(1)$，则 t_i 是离查询数据 Q 最近的数据记录。然而，服务器 C1 和 C2 无法获知 V 中的哪一项是 $E_{pk}(1)$。最后，服务器 C1 计算与 Q 最接近的数据记录 $E_{pk}(t')$，并更新距离向量，方法如下：

1) 服务器 C1 和 C2 共同参与安全乘法(SM)协议，计算 $V'_{i,j}=V_i*E_{pk}(t_{i,j})$，范围为$1 \le i \le n$ 和$1 \le j \le m$。安全乘法 SM 协议的输出结果 V' 只有 C1 知道。然后，服务器 C1 通过使用 paillier 加密的同态属性，计算加密数据记录 $E_{pk}(t'_1)=\langle E_{pk}(t_{1,1}),\cdots,E_{pk}(t_{1,m})\rangle$，其中 $E_{pk}(t'_{i,j})=\prod_{i=1}^{n}V'_{i,j}$，其中$1 \le j \le m$，$t'_{1,j}$ 为 t'_1 的第 j 个属性值。

2) 我们需要注意，最接近查询数据 Q 的那一个元组应该被排除在后续的计算之外。但是，由于服务器 C1 不知道与 $E_{pk}(t'_1)$ 对应的数据记录，因此我们需要强制消除在下一次迭代运算中再次选择该数据记录的可能性。对此，我们让 C1 将 $E_{pk}(t'_1)$ 对应的距离更新为最大值，即 $2l-1$。更具体地说，C1 在 C2 的帮助下，使用安全位或 SBOR 协议在范围$1 \le i \le n$ 和$1 \le \gamma \le l$ 内更新距离向量 $E_{pk}(d_{i,\gamma})=\text{SBOR}(V_i,E_{pk}(d_{i,\gamma}))$。当 $V_i=E_{pk}(1)$ 时，相应的距离向量 d_i 设置为最大值。也就是说，在这种情况下，$[d_i]=\langle E_{pk}(1),\cdots,E_{pk}(1)\rangle$。但是，当 $V_i=E_{pk}(0)$ 时，OR 操作对 d_i 没有影响。

重复上述迭代过程 k 次，直到每次迭代的$[d_i]$当前对应的记录值均被设置为最大值。但是，由于服务器 C1 不知道更新的数据对应哪个$[d_i]$，所以每次迭代都必须使用对应的$[d_i]$重新计算 $E_{pk}(d_i)$，i 范围为$1 \le i \le n$。在第 s 次迭代中，$E_{pk}(t'_s)$ 只有服务器 C1 知道。

在迭代步骤的末尾，即算法的第 3 步，C1 拥有查询数据 Q 的 k 个最近邻的加密记录数据列表 $\langle E_{pk}(t'_1),\cdots,E_{pk}(t'_k)\rangle$。该过程的其余部分类似于 SkNNb 算法的步骤 4 到 6。简而言之，选择加密记录 $E_{pk}(t_{i_1}),\cdots,E_{pk}(t_{i_k})$ 作为查询数据 Q 的 K 最近邻记录，并随机化它们的属性。具体而言，C1 计算 $E_{pk}(\gamma_{j,h})=E_{pk}(t_{i_j,h})*E_{pk}(r_{j,h})$，$1 \le j \le k$ 且$1 \le h \le m$。这里 $r_{j,h}$ 是 Z_N 范围内的一个随机数，$t_{i_j,h}$ 表示列 h

中 t_{i_j} 属性值。然后 C1 将 $\gamma_{j,h}$ 发送到 C2，$r_{j,h}$ 发送给 Bob，其中 $1 \le j \le k$ 且 $1 \le h \le m$。收到 $\gamma_{j,h}$ 之后，服务器 C2 解密计算 $\gamma'_{j,h} = D_{sk}\left(\gamma_{j,h}\right)$，并将其发送给 Bob。注意，因为 C1 之前将 $\gamma_{j,h}$ 的加密数值随机化，所以 $\gamma'_{j,h}$ 仍然是 Z_N 范围内的随机数。

Bob 收到 $r_{j,h}$ 和 $\gamma'_{j,h}$ 之后，使用 $t'_{j,h} = \gamma'_{j,h} - r_{j,h} \bmod N$ 来计算查询数据 Q 的 K 近邻数据值，其中 $1 \le j \le k$ 且 $1 \le h \le m$，之后 Bob 根据得到的 K 个数值进行离群点的检测，我们采用距离方法顺序进行检测，即从这组数据的第一个开始进行数值欧式距离的判定，一旦存在某一个数值 t_i 与 Q 之间的欧式距离大于预设的距离 D，$1 \le i \le k$，则说明在数据群 T 中与查询数据 Q 的欧式距离小于 D 的点不足 k 个，判定 Q 为离群点，否则，Q 不是离群点。

4. 实验分析

4.1. 安全性分析

首先，由于查询数据 Q 经过具有语义安全性的 Paillier 密码系统加密，Bob 的输入查询 Q 在第四节提出的两种协议中都无法被数据拥有者 Alice、服务器 C1 和 C2 知晓。

在 SkNNb 协议中，步骤 3(b)的解密操作将欧式距离 d_i 值泄露给服务器 C2。此外，由于 C2 生成了 k 近邻索引列表(在算法 5 步骤 3(c)中)并将其发送给 C1，因此数据访问模式将被泄露给服务器 C1 和 C2。因此，如果我们不介意 di 值可以泄露给 C2，数据访问模式可以泄露给 C1 和 C2，我们的基本 SkNNb 协议可以认为是安全的。

另一方面，对 SkNNm 的安全性分析如下。在算法步骤 2 中，安全欧几里得距离 SSED 协议和安全位分解 SBD 协议的输出是经过加密的，只有服务器 C1 知道。此外，在安全欧几里得距离 SSED 协议中所有由服务器 C2 解密的中间结果在 Z_N 中都是一致随机的。此外，安全位分解 SBD 协议是安全的。因此，在算法步骤 2 中没有泄露任何信息。在每次迭代中，SMINn 的输出只有服务器 C1 知道，而服务器 C2 不知道任何信息。另外，C1 和 C2 不知道哪个记录对应当前的全局最小值。因此，数据访问模式无法被服务器 C1 和 C2 知晓。算法步骤 3(c)，β 向服务器 C2 泄露了某个解密的元组满足当前全局最小距离。但是，由于 C1 发送之前进行了乱序排列，C2 无法追溯到对应的数据记录。同时，注意 β 的解密数据为 0 或范围 Z_N 内的随机数进行加密后的数据。同样，由于 C2 返回给 C1 的 U 是加密的向量，C1 无法知道哪个加密元组对应当前全局最小距离。因此，在此步骤中，数据访问模式无法被 C1 知晓。此外，算法步骤 3(e)的更新过程不会向服务器 C1 和 C2 泄漏任何信息。总之，服务器 C1 和 C2 不知道哪个数据记录对应于输出集 $\langle t'_1, \cdots, t'_k \rangle$。

综上所述，很明显，本文使用的 SkNNm 协议能有效保护数据的机密性、用户输入查询数据的隐私，并隐藏数据访问模式。

4.2. 复杂性分析

SkNNb 的计算复杂度受加密、解密和求幂的限制为 $O(n*m+k)$。实际上 $k \ll n*m$，因此，受到加密和求幂的限制(假设在 Pallier 密码系统下加密和解密操作需要相同的时间)SkNNb 的计算复杂度为 $O(n*m)$。另一方面，SkNNm 的计算复杂度受加密和取幂的限制为 $O\left(n*\left(l+m+k*l*\log_2 n\right)\right)$。

根据密钥大小的不同，我们使用的 SkNNm 的总体计算成本(高于 SkNNb 协议)比非加密情况(本文提出的相关工作)高 2 到 3 个数量级。这是为了最大化数据保密性所需要付出必要的代价。然而，在客户端，我们使用的协议所用运行时间与非加密情况相当，因为用户只需执行非常少量的操作(受属性数量的限制)，这些操作将在不到一秒的时间内完成。我们的目标是将所有或大部分计算发送给服务器，这样用户就可以使用任何具有存储和有限计算能力的移动设备提出查询。

4.3. 数据对比

我们在一台 Windows 机器上进行了各种实验。其中我们使用 JetBrainsPyCharm 作为 python 编辑器，使用 Anaconda 作为 python 第三方库的管理软件，主要使用了 python-paillier 库进行 paillier 加密解密，所使用的 python 版本为 3.7。

在数据集选择方面，我们使用了著名的 UCI Heart Disease 数据集作为数据库数据 T，并选择其中十四个主要属性进行实验。但是由于原数据集数据量较少，且在实际数据集中很难控制参数，我们根据考虑的参数值随机生成合成数据集。利用这些综合数据集，我们可以对不同参数设置下的协议计算成本进行更详细的分析。我们在实验中使用密钥大小不同的 Paillier 加密技术对这些数据集进行了属性加密，加密后的数据存储在我们的机器上。然后，根据所提议的协议，我们对这个加密数据执行一个随机查询。对于本节的其余部分，我们不讨论数据拥有者 Alice 的性能，因为它是一次性成本。我们分别评估和分析了 SkNNb 和 SkNNm 的性能，然后，我们比较了这两种协议的性能。在我们的实验中，Paillier 加密密钥大小 S 被设置为 512 位或 1024 位。

1) SkNNb 协议测试

在本小节中，我们通过改变数据记录的数量(n)、属性的数量(m)、最近邻居的数量(k)和加密密钥大小(S)来分析 SkNNb 协议的计算成本。

首先，通过设置最近邻数量 $k = 5$，密钥大小 $S = 512$，我们评估不同 n 和 m 下 SkNNb 的计算成本。如图 2 所示，SkNNb 的计算成本呈线性增长趋势。例如，当属性数量 $m = 6$ 时，随着数据记录数量 n 从 2000 增加到 4000，SkNNb 计算时间从 44.01 秒增加到 88.02 秒。密钥大小 $S = 1024$ 时也有类似的趋势，如图 3 所示。对于任何固定参数，我们发现当 S 大小增加一倍时，SkNNb 的计算时间几乎增加了 7 倍。

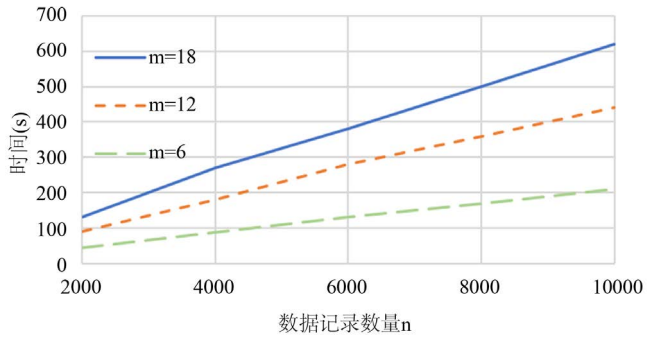

Figure 2. SkNNb trend chart in Case $k = 5$, $S = 512$
图 2. $k = 5$，$S = 512$ 情况下 SkNNb 趋势图

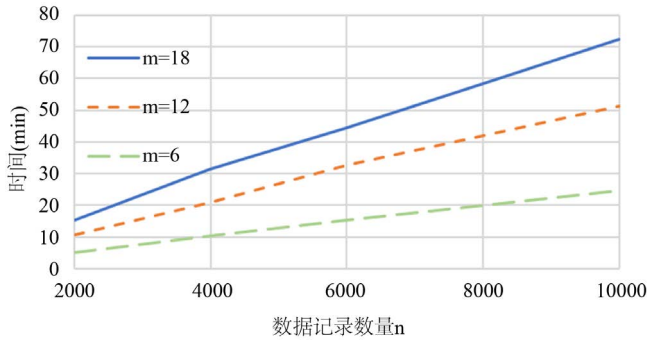

Figure 3. SkNNb Trend Chart in Case $k = 5$, $S = 1024$
图 3. $k = 5$，$S = 1024$ 情况下 SkNNb 趋势图

接下来，通过固定设置参数属性数量 $m = 6$ 和记录数量 $n = 2000$，我们评估了最近邻数量 k 和密钥大小 S 变化时 SkNNb 的运行时间，结果如图 4 所示。不管密钥大小 S 是多少，SkNNb 不会因最近邻数量 k 的改变而变化计算时间。这是因为 SkNNb 大部分的计算成本来自安全欧几里德 SSED 协议，该协议与最近邻数量 k 无关。例如，当 $S = 512$ 位，随着 k 从 5 变化到 25，SkNNb 计算时间从 44.01 变化到 44.11 秒。综上所述，可以看出，SkNNb 的运行时间主要取决于记录数量 n 和属性数量 m，这进一步证明了我们的复杂性分析是合理的。

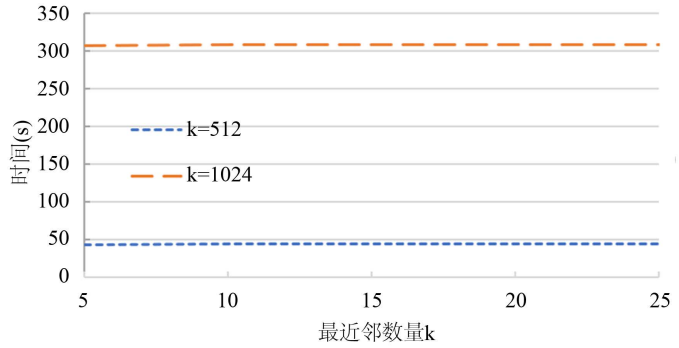

Figure 4. SkNNb Trend Chart in Case m = 6, $n = 2000$
图 4. $m = 6$，$n = 2000$ 情况下 SkNNb 趋势图

2) SkNNm 协议测试

我们还评估了最近邻数量 k、属性值大小(二进制位数)l 和密钥大小 S 值变化时 SkNNm 的计算成本。在本小节中，我们固定属性数量 $m = 6$ 且数据数量 $n = 2000$。我们发现 SkNNm 的运行时间也几乎呈线性增长。

当密钥大小 $S = 512$ 位时，变化最近邻数量 K 和属性二进制位数 l，产生的 SkNNm 计算量如图 5 所示。由图 5 可知，当 $l = 6$ 时，随着 k 从 5 变为 25，SkNNm 的运行时间从 11.91 变为 55.58 分钟。同样，对于 $l = 12$，SkNNm 的运行时间随着 k 从 5 变为 25，由 20.56 分钟变为 97.83 分钟。无论哪种情况，SkNNm 的计算成本几乎与 k 和 l 呈线性增长。

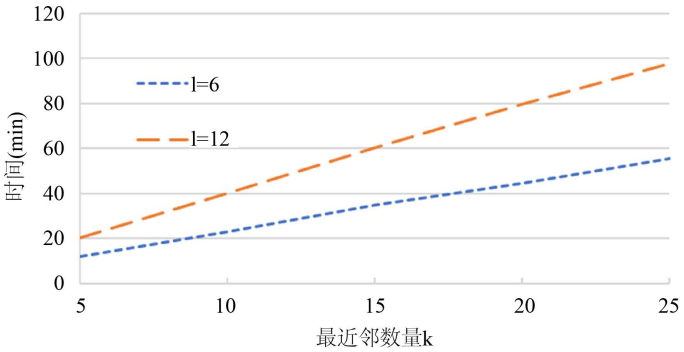

Figure 5. SkNNm trend chart in Case $n = 2000$, $S = 512$
图 5. $n = 2000$，$S = 512$ 情况下 SkNNm 趋势图

在 $S = 1024$ 时也观察到类似的趋势，如图 6 所示。特别地，对于任何给定的固定参数，我们发现当 S 增加一倍时，SkNNm 的计算成本几乎增加了 7 倍。例如，当 $k = 10$ 时，SkNNm 分别花费 22.83 分钟和 157.16 分钟生成 $S = 512$ 和 1024 位下查询数据 Q 的 10 个最近邻。此外，当 $k = 5$ 时，我们发现在 SkNNm

中约有 69.7%的成本是由于安全最小 n 个数字协议 SMINn 在 SkNNm 中启动了 k 次。同时，当 k 值从 5 增加到 25 时，SMINn 引起的成本从 69.7%增加到至少 75%。

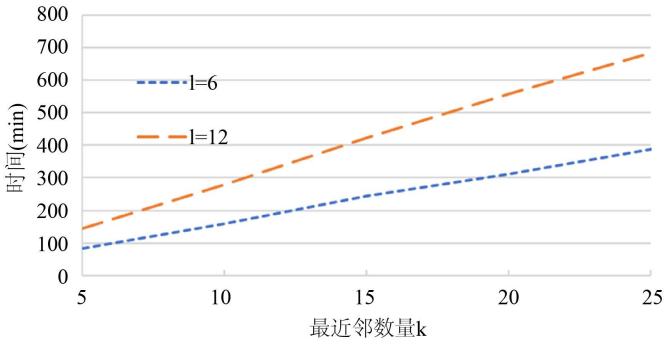

Figure 6. SkNNm trend chart in Case $n = 2000$, $S = 1024$
图 6. $n = 2000$，$S = 1024$ 情况下 SkNNm 趋势图

此外，通过设置记录数量 $n = 2000$，属性数量 $m = 6$，属性大小 $l = 6$ 和密钥位数 $S = 512$，我们比较两个协议在不同的 K 值下的运行时间，如图 7 所示，SkNNb 的运行时间一直约为 0.75 分钟，因为它几乎与 k 不相关。然而，当我们把 K 从 5 增加到 25，SkNNm 的运行时间从 11.91 分钟变化到 55.58 分钟。

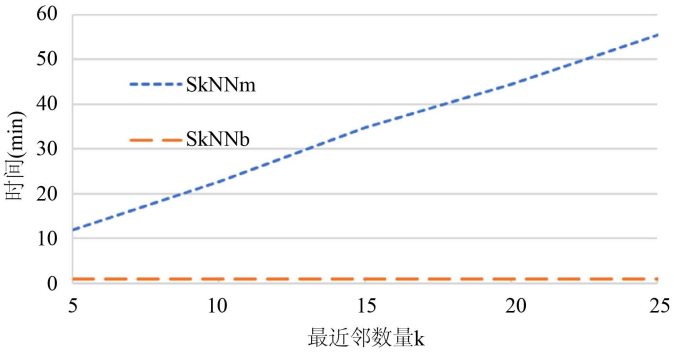

Figure 7. Trend comparison chart between SkNNm and SkNNb
图 7. SkNNm 与 SkNNb 趋势对比图

从以上结果可以看出，SkNNm 的计算成本明显高于 SkNNb。但是，我们强调 SkNNm 比 SkNNb 更安全。在 SkNNb 中，安全欧几里得距离 SSED 协议最耗费时间，而在 SkNNm 中，则是安全最小 n 个数字 SMINn 协议。另外，需要注意的是，Bob 的计算成本主要是由于其输入查询记录的加密和最终离群点的计算造成的。例如，对于 $m = 6$，当 K 分别为 512 位和 1024 位时，Bob 的计算成本分别为 10 和 37 毫秒。这进一步表明，从终端用户的角度来看，我们的协议非常有效。

5. 总结

k 近邻是离群点检测常用的查询方法之一。在将加密数据存储在数据库环境中，对加密数据的安全查询处理变得很有挑战性。现有的基于加密数据的 SkNN 技术是不安全的。在本文中，我们使用了两种新的基于加密数据的 SkNN 协议进行离群点的检测。作为基本解决方案的第一种协议向服务器泄漏了一些信息。但是，我们使用的第二个协议是完全安全的，能够保护数据、用户输入查询的机密性，并隐藏数据访问模式。然而，第二种协议比基本协议计算成本更高。此外，我们还评估了协议在不同参数设置

下的性能。

关于在客户端进行离群点最终检测问题，因为服务器 C1 和 C2 得到 K 近邻数据之后分别用随机数扰动再汇总到客户端。直接得出欧式距离后与固定距离 D 进行比较，在不将数据泄露给服务器 C2 的情况下是不可能进行的，所以我们还没有研究出如何直接在服务器上安全地进行离群点检测。其实我们可以在获取到 Q 的 k 近邻数据后将其加密并二次发送至服务器进行离群点检测，但是存在一个问题，就是解密之后才能继续判断是否为离群点，但这又会让 C2 得知离群点检测的信息。关于进一步降低客户端的计算成本，我们留待以后进行研究。

基金项目

文章研究成果由国家电网有限公司科技项目“面向数据中心的数据共享分发安全关键技术研究及应用”(项目编号 Grand No. 5700-202090192A-0-0-00)支持。

参考文献

[1] Oppliger, R. (2003) Microsoft.net Passport: A Security Analysis. *Computer*, **7**, 29-35. https://doi.org/10.1109/MC.2003.1212687

[2] Ellin, B. (2006) About openlD. http://www.openidenabled.com/openid/about-openid

[3] ShoCardBlockchain Identity Management White Paper. https://shocard.com/blockchain-identity-whitepapers/

[4] Erlingsson, U., Pihur, V. and Korolova, A. (2014) Rappor: Randomized Aggregatable Privacy-Preserving Ordinal Response. *ACM SIGSAC Conference on Computer and Communications Security*, 1054-1067. https://doi.org/10.1145/2660267.2660348

[5] Warner, S.L. (1965) Randomized Response: A Survey Technique for Eliminating Evasive Answer Bias. *Journal of the American Statistical Association*, **60**, 63-66. https://doi.org/10.1080/01621459.1965.10480775

[6] Dwork, C. (2006) Differential Privacy. *International Conference on Automata*, *Languages and Programming*, 1-12. https://doi.org/10.1007/11787006_1

[7] Dwork, C., McSherry, F., Nissim, K. and Smith, A. (2006) Calibrating Noise to Sensitivity in Private Data Analysis. *Theory of Cryptography Conference*, 265-284. https://doi.org/10.1007/11681878_14

[8] Duchi, J.C., Jordan, M.I. and Wainwright, M.J. (2013) Local Privacy and Statistical Minimax Rates. *IEEE FOCS*, 429-438. https://doi.org/10.1109/FOCS.2013.53

[9] Liu, J.F., Yang, J.C. and Li, X. (2018) Secure and Efficient Skyline Queries on Encrypted Data, TKDE.