软件学报 ISSN 1000-9825, CODEN RUXUEW
Journal of Software, [doi: 10.13328/j.cnki.jos.006129]

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

全委托的公共可验证的外包数据库方案*

周搏洋[1], 陈春雨[2], 王 强[1], 周福才[1]

[1](东北大学 软件学院,辽宁 沈阳 110169)

[2](中国科学院沈阳自动化研究所,辽宁 沈阳 110016)

通讯作者: 周福才, E-mail: fczhou@mail.neu.edu.cn

摘 要: 为解决可验证外包数据库方案存在的预处理阶段开销较大及不支持公共可验证的问题,提出了一个全委托的公共可验证的外包数据库模型.给出了模型的架构及交互流程,对模型进行了形式化定义并给出了模型的正确性定义和安全性定义.利用双线性映射及可验证外包模幂运算协议构建了一个全委托的公共可验证外包数据库方案,且给出了各个算法的详细描述,证明了方案的正确性和安全性.其安全性可规约为 BDHE(Bilinear Diffie-Hellman Exponent)难题.与现有方案及不进行全委托计算的方案相比,该方案基于可验证外包模幂运算,将大量模幂运算外包给云处理,减小了数据拥有者的开销.理论与实验分析表明:该方案数据拥有者在预处理阶段所需的代价更低,效率更高,适于实际应用.此外,验证过程无需私钥参与,实现了公共可验证.

关键词: 可验证数据库;可验证计算;公共可验证;全委托;双线性映射

中图法分类号: TP311

中文引用格式: 周搏洋,陈春雨,王强,周福才.全委托的公共可验证的外包数据库方案.软件学报. http://www.jos.org.cn/1000-9825/6129.htm

英文引用格式: Zhou BY, Chen CY, Wang Q, Zhou FC. Publicly Verifiable Outsourced Database with Full Delegation. Ruan Jian Xue Bao/Journal of Software, (in Chinese). http://www.jos.org.cn/1000-9825/6129.htm

Publicly Verifiable Outsourced Database with Full Delegations

ZHOU Bo-Yang[1], CHEN Chun-Yu[2], WANG Qiang[1], ZHOU Fu-Cai[1]

[1](Software College, Northeastern University, Shenyang 110169, China)

[2](Shenyang Institute of Automation Chinese Academy of Sciences, Shenyang 110016, China)

Abstract: To solve the problem of high preprocessing cost and public verifiability in the verifiable outsourced database schemes, a publicly verifiable outsourced database with full delegation is proposed. We present the architecture and the definition of security and correctness of the model. Based on the bilinear map and verifiable outsourced modular exponentiations protocol, we construct a publicly verifiable outsourced database scheme with full delegation, and design each algorithm in detail. We present the rigorous security proof under the Bilinear Diffie-Hellman Exponent problem. Compared with performing the protocol without full delegation scheme and the existing schemes, the data owner in PVDVD scheme outsources more operations to the cloud because of the application of the verifiable outsourced modular exponentiation operation. The theoretical analysis and simulation confirm that the cost of our scheme is lower in the preprocessing phase, which makes it more efficient and practical. In the verification phase, any user can verify the result since the verification algorithm does not take any secret key as input. Therefore, our proposed scheme achieves public verifiability.

Key words: verifiable database; verifiable computing; public verification; full delegation; bilinear pairing

* 基金项目: 国家自然科学基金(61872069); 中央高校基本科研业务费专项基金项目(N171704005)

Foundation item: National Natural Science Foundation of China (61872069); Fundamental Research Funds for the Central Universities (N171704005).

收稿时间: 2020-04-11; 修改时间: 2020-05-24; 采用时间: 2020-08-13; jos 在线出版时间: 2021-05-20

随着信息技术的高速发展,智能手机等小型终端设备成为人们生活中不可或缺的一部分,然而这些终端设备受限于较弱的计算和存储能力,无法承受庞大的数据量所带来的昂贵代价.伴随着云计算[1-3]的高速发展,这些弱计算能力的终端可以将数据外包给云服务提供商[4]管理,即用户将庞大的数据通过外包的形式让拥有强大计算能力的云服务器来存储和计算.通过这种将数据外包的方式,用户可以随时随地发出查询请求,云服务器则会根据不同的请求执行查询操作并将结果返回给用户.因此,用户本地只需负责数据传输等简单操作即可,使得其持有轻量级设备即可执行查询操作.然而,由于用户丧失了对数据的直接控制[5],云服务器可能会伪造查询结果来减少响应时间和查询代价.这种资源租用的模式也带来了查询的安全性、正确性和隐私性等诸多问题,在这种情况下如何使得用户能够检测和验证查询结果的完整性成为了亟待解决的问题.

针对上述问题,Benabbas 等人于 2011 年首次提出可验证数据库(verifiable database, VDB)的概念[6],包括三方实体:数据拥有者、云服务器和用户.其中数据拥有者持有数据库,并将其外包给云服务器存储.用户则可根据索引对数据库进行查询.由于具有可验证性,当用户对索引进行查询时,云服务器返回查询结果和一个能够验证结果完整性的证据.用户便可根据该证据来验证结果是否被篡改,即得到查询结果的同时又能验证其完整性.而其中需要满足的一个关键条件是:用户查询验证的开销要远远小于其在本地执行查询计算的开销,否则便失去了外包的意义.对于持有小型终端设备的数据拥有者而言,由于计算能力的限制,方案能实际应用的关键是其使用轻量级设备在可接受的时间内能完成协议中的步骤.在可验证数据库方案中,数据拥有者的工作量主要由外包预处理阶段的开销构成,包括初始化和密钥生成算法.

近年来,国内外研究人员针对可验证数据库做了大量的研究[7-25].其中的很多方案在预处理阶段的开销超出了持有小型终端的数据拥有者的计算能力,以至于无法在现实中使用.如文献[18]利用向量承诺[26]的信息隐藏性和位置绑定性构建了一个可验证数据库方案,但由于向量承诺所需开销较大,数据拥有者在预处理阶段要执行 $O(\ell^2)$ 量级的指数运算才能实现初始化和密钥生成等操作,其中 ℓ 为数据库中数据集的大小.当数据库很庞大时,这样的开销对于轻量级的设备而言是不可接受的.为解决这一问题,文献[19]提出了一种分摊模型,将昂贵的预处理开销分摊到方案后面的查询步骤中,从而对每次查询而言“减小”其预处理阶段的开销.然而实际上,对数据拥有者而言,分摊的方式并没有减少其在整个方案中的工作量.因此,分摊模型依然不满足数据拥有者的关键要求.此外,在当前很多方案中,如文献[6]和文献[21],用户的验证过程需要私钥的参与,也就是说只有持有私钥的用户才能验证查询结果的完整性.而在实际场景中,用户可能希望通过代理来验证查询结果,公共可验证则恰好满足此要求.在公共可验证的方案中,任何一个被数据拥有者授权即持有验证密钥的用户均可以对外包数据库的查询结果进行验证.综上,在实际的不可信云环境中,如何在实现外包数据库完整性验证的前提下,又能够降低数据拥有者在预处理阶段的开销成为了关键,现有方案未能同时解决数据拥有者预处理开销大及公共可验证的问题.

针对上述要求,本文提出了一个全委托的公共可验证的外包数据库(Publicly Verifiable Outsourced Database with Full Delegation, PVDFD)模型,该模型具有以下 2 个特点:

1) 数据拥有者预处理开销小.利用可验证外包模幂运算,将预处理阶段大量的模幂运算委托给云服务器,使得数据拥有者与云服务器交互获得验证密钥的工作量小于自己生成验证密钥的工作量,同时保证验证密钥的机密性.这里的全委托是指除了将数据库外包给云服务器来检索外,还将预处理阶段生成验证密钥的部分运算交给云来完成;

2) 公共可验证.本方案支持公共可验证,即任何持有验证密钥的用户均可根据任意用户发送的查询请求及云服务器返回的结果和证据进行验证.

1 相关知识

本文所用符号,如表 1 所示.

Table 1 Notations

表 1 符号说明

符号	描述	符号	描述
λ	安全参数	x	授权用户的查询索引
pp	公共参数	y	索引 x 的查询结果
DB	外包的数据库	π_y	结果 y 的证据
ℓ	数据库 DB 的大小	$ME.EK$	$MExp$ 协议的求值密钥
$\mathcal{EK}_{DB}$	数据库 DB 的查询密钥	$ME.VK$	$MExp$ 协议的验证密钥
$\mathcal{VK}_{DB}$	数据库 DB 的验证密钥	$ME.y$	$MExp$ 协议的模幂运算结果
$\mathcal{EK}_{pp}$	验证密钥 $\mathcal{VK}_{DB}$ 的计算密钥	$ME.\sigma_y$	编码后的结果 $ME.y$
$\mathcal{VK}_{pp}$	验证密钥 $\mathcal{VK}_{DB}$ 的恢复密钥	$ME.\pi_y$	结果 $ME.y$ 对应的证据
$\mathcal{RK}_{pp}$	验证密钥 $\mathcal{VK}_{DB}$ 的检索密钥	$\mathcal{F}$	数据库 DB 插值后形成的多项式
$\sigma_{\mathcal{VK}_{DB}}$	编码后的验证密钥 $\mathcal{VK}_{DB}$	PRF	伪随机函数
$\pi_{\mathcal{VK}_{\mathrm{DB}}}$	编码 $\sigma_{\mathcal{VK}_{DB}}$ 对应的证据	α	伪随机函数 PRF 的密钥

1.1 双线性映射

双线性映射[27-29]指的是 2 个循环群之间相对应的线性映射关系.$\mathcal{G}_1$ 和 $\mathcal{G}_2$ 均为 p 阶乘法循环群,g 为群 $\mathcal{G}_1$ 的生成元,定义 2 个群上的双线性映射为 $e:\mathcal{G}_1\times\mathcal{G}_1\to\mathcal{G}_2$,满足以下性质:

(1) 双线性.对于任意的 $a,b\in\mathcal{Z}_p$ 和 $u,v\in\mathcal{G}_1$,均满足 $e\left(u^a,v^b\right)=e\left(u,v\right)^{ab}$.

(2) 非退化性.$e\left(g,g\right)\neq 1_{\mathcal{G}_2}$.

(3) 可计算性.对于任意的 $u,v\in\mathcal{G}_1$,都存在有效的算法计算出 $e\left(u,v\right)$.

1.2 双线性Diffie-Hellman指数难题(BDHE)

双线性 Diffie-Hellman 指数难题[30](Bilinear Diffie-Hellman Exponent, BDHE),$\mathcal{G}_1$ 和 $\mathcal{G}_2$ 均为 p 阶乘法循环群,g,u 为群 $\mathcal{G}_1$ 的两个生成元,群上的双线性映射为 $e:\mathcal{G}_1\times\mathcal{G}_1\to\mathcal{G}_2$.随机选择 $\alpha\in\mathcal{Z}_p$,给定一个元组 $\left(U_0,U_1,\ldots,U_n,U_{n+2},\ldots,U_{2n}\right)$,使得对于任意的 $i\in\left\{0,\ldots,n,n+2,\ldots,2n\right\}$ 均满足 $U_i=u^{\alpha^i}$.计算 $e\left(g,U_{n+1}\right)=e\left(g,u^{\alpha^{n+1}}\right)\in\mathcal{G}_2$ 是困难的.

2 PVDFD 模型

本节主要介绍了全委托的公共可验证的外包数据库 PVDFD 模型.首先给出了模型的架构及交互流程,然后给出了模型的形式化定义和安全性定义.

2.1 PVDFD模型架构

全委托的公共可验证的外包数据库模型 PVDFD 中包含 4 方实体,分别是可信第三方、数据拥有者、云服务器以及授权用户,其中云服务器是不可信的,其他实体是可信的.模型架构如图 1 所示,可信第三方首先执行初始化算法生成公共参数,并将其分发给数据拥有者、授权的用户以及云服务器.数据拥有者在外包数据库之前,利用可信第三方生成的公共参数执行密钥生成算法生成密钥.包括数据库的查询密钥、验证密钥的计算密钥、验证密钥的恢复密钥和验证密钥的检索密钥,并将验证密钥的计算密钥发送给云服务器.云服务器执行验证密

钥计算算法,返回编码后的验证密钥及相应的证据.数据拥有者利用验证密钥恢复算法验证,在验证通过的情况下完成验证密钥的恢复操作,将其发送给授权的用户,并将数据库的查询密钥发给云服务器.接着授权的用户便可以向云服务器发送查询索引.由于云服务器是不可信的,其执行查询算法返回结果及证据,授权的用户执行验证算法实现验证操作,若算法输出 1 则表示云服务器执行了正确的查询.由于持有验证密钥的用户均可对查询结果进行验证,因此该方案支持公共可验证.

注:本文中的数据库为 NoSQL 类型,表示为 $DB=\{(i,v_i)\mid i\in\{0,\ldots,n\}\}$,数据库中数据集大小为 $\ell=n+1$.

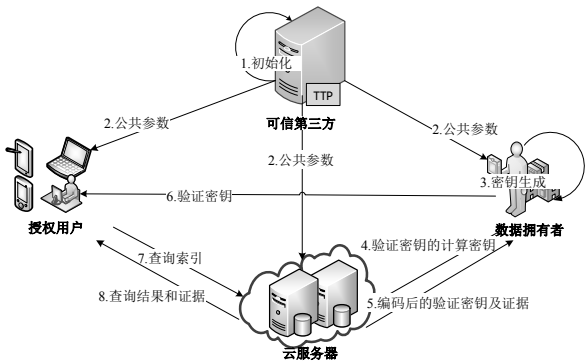

Fig.1 The architecture of PVDFD

图 1 PVDFD 模型架构

2.2 PVDFD的形式化定义

该模型可由概率多项式时间算法六元组 PVDFD = (*Setup*,*KeyGen*,*VKeval*,*VKrec*,*Query*,*Verify*) 表示,其中每项都代表一个多项式时间算法.6 个算法具体描述如下:

1) $Setup(1^{\lambda})\to pp$:初始化算法.初始化算法由可信第三方执行,算法根据输入的安全参数 λ 生成公共参数 pp,并发送给方案中的其他实体.

2) $KeyGen(pp,DB)\to(\mathcal{EK}_{DB},\mathcal{EK}_{pp},\mathcal{VK}_{pp},\mathcal{RK}_{pp})$:密钥生成算法.密钥生成算法由数据拥有者执行,算法输入数据库 DB 和公共参数 pp,生成数据库的查询密钥 $\mathcal{EK}_{DB}$、验证密钥的计算密钥 $\mathcal{EK}_{pp}$、验证密钥的恢复密钥 $\mathcal{VK}_{pp}$ 和验证密钥的检索密钥 $\mathcal{RK}_{pp}$.其中 $\mathcal{EK}_{DB}$ 用来查询数据库,$\mathcal{EK}_{pp}$、$\mathcal{VK}_{pp}$、$\mathcal{RK}_{pp}$ 分别用来计算、恢复和检索数据库的验证密钥 $\mathcal{VK}_{DB}$.(数据库的验证密钥 $\mathcal{VK}_{DB}$ 在下文验证密钥恢复算法 *VKrec* 中定义).

3) $VKeval(pp,\mathcal{EK}_{pp})\to(\sigma_{\mathcal{VK}_{DB}},\pi_{\mathcal{VK}_{DB}})$:验证密钥计算算法.验证密钥计算算法由云服务器执行,算法输入公共参数 pp 和验证密钥的计算密钥 $\mathcal{EK}_{pp}$,输出编码后的数据库的验证密钥 $\sigma_{\mathcal{VK}_{DB}}$ 和相关的证据 $\pi_{\mathcal{VK}_{DB}}$.

4) $VKrec(pp,\mathcal{VK}_{pp},\sigma_{\mathcal{VK}_{DB}},\pi_{\mathcal{VK}_{DB}},\mathcal{RK}_{pp})\to\{\mathcal{VK}_{DB},\perp\}$:验证密钥恢复算法.验证密钥恢复算法由数据拥有者执行,算法输入公共参数 pp、验证密钥的恢复密钥 $\mathcal{VK}_{pp}$、验证密钥的检索密钥 $\mathcal{RK}_{pp}$ 和云服务器返回的编码后的数据库的验证密钥 $\sigma_{\mathcal{VK}_{DB}}$ 及证据 $\pi_{\mathcal{VK}_{DB}}$.若验证通过,则输出解码后的数据库的验证密钥 $\mathcal{VK}_{DB}$,否则输出 $\perp$.

5) $Query\left(\mathcal{EK}_{DB},x\right)\to\left(y,\pi_y\right)$:查询算法.查询算法由云服务器执行,算法输入数据库的查询密钥 $\mathcal{EK}_{DB}$ 及授权用户发来的查询索引 $x\in domain\left(DB\right)$,输出对应的查询结果和证据对 $\left(y,\pi_y\right)$.

6) $Verify\left(pp,\mathcal{VK}_{DB},x,y,\pi_y\right)\to\{0,1\}$:验证算法.验证算法由持有数据库的验证密钥的用户执行,算法输入公共参数 pp、验证密钥 $\mathcal{VK}_{DB}$、查询索引 x 和云服务器返回的查询结果 y 证据 π_y.验证通过输出 1,否则输出 0.

2.3 PVDFD的正确性及安全性定义

2.3.1 正确性

PVDFD 方案的正确性即若模型中的每个算法都被正确地执行,则永远不会拒绝正确的结果.其形式化定义如下:

$$\Pr\left[\begin{array}{l}Setup\left(1^{\lambda}\right)\to pp,\\ KeyGen\left(pp,DB\right)\to\left(\mathcal{EK}_{DB},\mathcal{EK}_{pp},\mathcal{VK}_{pp},\mathcal{RK}_{pp}\right),\\ VKeval\left(pp,\mathcal{EK}_{pp}\right)\to\left(\sigma_{\mathcal{VK}_{DB}},\pi_{\mathcal{VK}_{DB}}\right):\\ VKrec\left(pp,\mathcal{VK}_{pp},\sigma_{\mathcal{VK}_{DB}},\pi_{\mathcal{VK}_{DB}},\mathcal{RK}_{pp}\right)\to\mathcal{VK}_{DB}\end{array}\right]\ge 1-negl\left(\lambda\right),\text{且 }\Pr\left[\begin{array}{l}Query\left(\mathcal{EK}_{DB},x\right)\to\left(y,\pi_y\right):\\ Verify\left(pp,\mathcal{VK}_{DB},x,y,\pi_y\right)\to 1\end{array}\right]\ge 1-negl\left(\lambda\right).$$

其中 $negl(\lambda)$ 为关于安全参数 λ 的可忽略函数.

2.3.2 安全性

1) 验证密钥的机密性

验证密钥的机密性是针对不可信的云服务器而言的.简单来说,云服务器无法获取外包数据库的验证密钥.下面通过安全性实验来定义该模型中的验证密钥的机密性.定义敌手 $\mathcal{A}\left(\cdot\right)=\left(\mathcal{A}_0,\mathcal{A}_1\right)$,其中 $\mathcal{A}_0,\mathcal{A}_1$ 为两个概率多项式时间模拟器,设计安全实验如下:

$Exp_{\mathcal{A}}^{Privacy}\left[\text{PVDFD},\lambda\right]$:
$pp\leftarrow Setup\left(1^{\lambda}\right)$;
$\left(\mathcal{EK}_{DB_0},\mathcal{EK}_{DB_1}\right)\leftarrow\mathcal{A}_0\left(pp,DB_0,DB_1\right)$;
$b\in_R\{0,1\}$;
$\left(\sigma_{\mathcal{VK}_{DB_b}},\pi_{\mathcal{VK}_{DB_b}}\right)\leftarrow VKeval\left(pp,\mathcal{EK}_{DB_b}\right)$;
$\hat{b}\leftarrow\mathcal{A}_1\left(\mathcal{EK}_{DB_0},\mathcal{EK}_{DB_1},\sigma_{\mathcal{VK}_{DB_b}},\pi_{\mathcal{VK}_{DB_b}}\right)$;
if $b=\hat{b}$:
  output 1;
else
  output 0;

对于任意的 $\lambda\in N$,定义敌手 $\mathcal{A}$ 在 PVDFD 中的优势如下:

$$Adv_{\mathcal{A}}^{Privacy}\left(\text{PVDFD},\lambda\right)=\left|\Pr\left[Exp_{\mathcal{A}}^{Privacy}\left[\text{PVDFD},\lambda\right]=1\right]-1/2\right|$$

定义 1. 若 $Adv_{\mathcal{A}}^{Privacy}\left(\text{PVDFD},\lambda\right)\le negl\left(\lambda\right)$,则 PVDFD 方案实现了验证密钥的机密性.其中 $negl\left(\lambda\right)$ 为可忽略函数.

2) 验证密钥的不可伪造性

验证密钥的不可伪造性是针对不可信的云服务器而言的.即云服务器伪造一个验证密钥始终不能通过数

据拥有者的验证.下面通过安全性实验来定义该模型中的验证密钥的不可伪造性.定义敌手 $\mathcal{A}(\cdot)=(\mathcal{A}_0,\mathcal{A}_1,\mathcal{A}_2)$,其中 $\mathcal{A}_0,\mathcal{A}_1,\mathcal{A}_2$ 为三个概率多项式时间模拟器,设计安全实验如下:

$Exp_{\mathcal{A}}^{VKU}[\mathrm{PVDFD},\lambda]$:
$pp \leftarrow Setup(1^{\lambda})$;
For $i=1$ *to* $q=poly(\lambda)$:
  $DB_i \leftarrow \mathcal{A}_0(1^{\lambda},pp)$;
  $(\mathcal{VK}_{pp_i},\mathcal{RK}_{pp_i},\mathcal{EK}_{DB_i},\mathcal{EK}_{pp_i}) \leftarrow KeyGen(pp,DB_i)$;
  $(\sigma_{\mathcal{VK}_{DB_i}},\pi_{\mathcal{VK}_{DB_i}}) \leftarrow VKeval(pp,\mathcal{EK}_{pp_i})$;
$\mathcal{EK}_{pp_i} \leftarrow \mathcal{A}_1(pp,\mathcal{EK}_{pp_1},\ldots,\mathcal{EK}_{pp_q},$
  $\sigma_{\mathcal{VK}_{DB_1}},\ldots,\sigma_{\mathcal{VK}_{DB_q}},\pi_{\mathcal{VK}_{DB_1}},\ldots,\pi_{\mathcal{VK}_{DB_q}})$;
$(\hat{\sigma}_{\mathcal{VK}_{DB_i}},\hat{\pi}_{\mathcal{VK}_{DB_i}}) \leftarrow \mathcal{A}_2(pp,\mathcal{EK}_{pp_i},\sigma_{\mathcal{VK}_{DB_i}},\pi_{\mathcal{VK}_{DB_i}})$;
$\hat{b} \leftarrow VKrec(pp,\mathcal{VK}_{pp_i},\hat{\sigma}^*_{\mathcal{VK}_{DB_i}},\hat{\pi}^*_{\mathcal{VK}_{DB_i}},\mathcal{RK}_{pp_i})$;
if $\hat{\sigma}_{\mathcal{VK}_{DB_i}} \neq \sigma_{\mathcal{VK}_{DB_i}}$ *and* $\hat{b}=\mathcal{VK}_{DB_i}$:
  output 1;
else
  output 0;

对于任意的 $\lambda \in N$,定义敌手 $\mathcal{A}$ 在 PVDFD 中的优势如下:

$$Adv_{\mathcal{A}}^{VKU}(\mathrm{PVDFD},\lambda)=\Pr\left[Exp_{\mathcal{A}}^{VKU}[\mathrm{PVDFD},\lambda]=1\right]$$

定义 2. 若 $Adv_{\mathcal{A}}^{VKU}(\mathrm{PVDFD},\lambda) \le negl(\lambda)$,则 PVDFD 方案实现了验证密钥的不可伪造性.其中 $negl(\lambda)$ 为可忽略函数.

3) 查询结果的不可伪造性

查询结果的不可伪造性是针对不可信的云服务器而言的.即云服务器伪造一个查询结果和证据始终不能通过授权用户的验证.下面通过安全性实验来定义该模型中的查询结果的不可伪造性.定义敌手 $\mathcal{A}(\cdot)=(\mathcal{A}_0,\mathcal{A}_1)$,其中 $\mathcal{A}_0,\mathcal{A}_1$ 为两个概率多项式时间模拟器,设计安全实验如下:

$Exp_{\mathcal{A}}^{RU}[\mathrm{PVDFD},DB,\lambda]$:
$pp \leftarrow Setup(1^{\lambda})$;
$(\mathcal{RK}_{pp},\mathcal{VK}_{pp},\mathcal{EK}_{pp},\mathcal{EK}_{DB}) \leftarrow KeyGen(pp,DB)$;
$(\sigma_{\mathcal{VK}_{DB}},\pi_{\mathcal{VK}_{DB}}) \leftarrow VKeval(pp,\mathcal{EK}_{pp})$;
$\mathcal{VK}_{DB} \leftarrow VKrec(pp,\sigma_{\mathcal{VK}_{DB}},\pi_{\mathcal{VK}_{DB}},\mathcal{VK}_{pp},\mathcal{RK}_{pp})$;
For $i=1$ *to* $q=poly(\lambda)$:
  $(y_i,\pi_{y_i}) \leftarrow Query(\mathcal{EK}_{DB},x_i)$;
$x^* \leftarrow \mathcal{A}_0(pp,x_1,\ldots,x_q,y_1,\ldots,y_q,\mathcal{EK}_{DB})$;
$(y^*,\pi^*_y) \leftarrow \mathcal{A}_0(pp,\mathcal{EK}_{pp},x^*)$;
$(\hat{y}^*,\hat{\pi}^*_y) \leftarrow \mathcal{A}_1(pp,x^*,x_1,\ldots,x_q,y_1,\ldots,y_q,\mathcal{EK}_{DB})$;
$\hat{b}^* \leftarrow Verify(pp,\mathcal{VK}_{DB},\mathcal{RK}_{pp},\hat{y}^*,\hat{\pi}^*_y,x^*)$;
If $\hat{y}^* \neq y^*$ *and* $\hat{b}^*=1$:
  output 1;
else
  output 0;

对于任意的 $\lambda \in N$,定义敌手 $\mathcal{A}$ 在 PVDFD 中的优势如下:

$$Adv_{\mathcal{A}}^{RU}\left(\text{PVDFD}, DB, \lambda\right) = \Pr\left[Exp_{\mathcal{A}}^{RU}\left[\text{PVDFD}, DB, \lambda\right] = 1\right]$$

定义 3. 若 $Adv_{\mathcal{A}}^{RU}\left(\text{PVDFD}, DB, \lambda\right) \le negl(\lambda)$,则 PVDFD 方案实现了结果的不可伪造性.其中 $negl(\lambda)$ 为可忽略函数.

3 PVDFD 方案的描述

本节首先介绍了方案构建中用到的一个子协议 *MExp* [31],然后详细介绍了 PVDFD 方案中的各个算法,并给出了正确性证明.由于本方案的安全性归约为 BDHE 难题,在此难题中需要 $O(\ell)$ 量级的幂运算,因此数据拥有者在预处理阶段要生成验证密钥所需的开销很大.而 *MExp* 协议为可验证的外包模幂运算协议,数据拥有者可以利用此协议,将这些幂运算外包给云服务器计算,同时又能验证结果的完整性.在此过程中需要注意的是,云服务器不能获取验证密钥的信息,否则其可以伪造结果和证据而不被验证者察觉.

3.1 子协议:可验证外包模幂运算 *MExp*

PVDFD 方案在预处理阶段的云服务器和数据拥有者交互生成验证密钥的过程中用到了可验证外包模幂运算协议 *MExp* [31] 作为子协议.为简单起见,令 p 为大素数, $u_i \in \mathcal{Z}_p$ 作为底, $c_i \in \mathcal{Z}_p$ 作为幂,其中 $i \in \{0,\ldots,n\}$. *MExp* 协议由 Ding 等人于 2017 年提出,协议中存在两方实体:诚实的客户端和不可信的服务器.客户端希望将乘法模幂计算 $u_0^{c_0}u_1^{c_1}\cdots u_n^{c_n} \bmod p$ 外包给计算能力强但不可信的服务器来计算,而客户端能够验证结果的完整性.简单地提取该协议中的系统模型,表示为以下三个步骤,其中均省略 mod p 操作.

1) $ME.Setup(u_0,\ldots,u_n,c_0,\ldots,c_n,p) \to (ME.EK, ME.VK)$

此步骤由客户端执行,生成关于模幂计算 $u_0^{c_0}u_1^{c_1}\cdots u_n^{c_n}$ 的求值密钥 *ME.EK* 和验证密钥 *ME.VK* .具体如下:

① 生成六组绑定的对 (k_0, g^{k_0}) , (k_1, g^{k_1}) , (k_2, g^{k_2}) , (k_3, g^{k_3}) , (k_4, g^{k_4}) , (k_5, g^{k_5}) ,同时令 $s_0 = g^{k0}$, $s_1 = g^{k1}$, $v_0 = g^{k_4}$, $v_1 = g^{k_5}$.

②计算 b_i 和 t_0h_0 ,使其满足 $k_2 = k_4(c_0 + c_1 + \ldots + c_n) - k_0t_0h_0$,其中 $i \in \{0,\ldots,n\}$.

③计算 $w_i = u_i / v_0$,其中, $i \in \{0,\ldots,n\}$ 并生成逻辑划分如下:

$$\begin{aligned} u_0^{c_0}\cdots u_n^{c_n} &= (v_0w_0)^{c_0}\cdots(v_0w_n)^{c_n} = v_0^{c_0+c_1+\cdots+c_n}w_0^{c_0}\cdots w_n^{c_n} \\ &= g^{k_4(c_0+c_1+\cdots+c_n)-k_0t_0h_0}g^{k_0t_0h_0}\cdot(w_0\cdots w_n)^{t_0h_0}w_0^{b_0}\cdots w_n^{b_n} = g^{k_2}(s_0w_0\cdots w_n)^{t_0h_0}w_0^{b_0}\cdots w_n^{b_n} \end{aligned}$$

④选择随机数 $r \in \mathcal{Z}_p$,计算 $w_i^{'} = u_i^{'} / v_1$,其中, $i \in \{0,\ldots,n\}$.将 $\left(u_0^{c_0}\cdots u_n^{c_n}\right)^r$ 进行逻辑划分如下:

$$\begin{aligned} \left(u_0^{c_0}\cdots u_n^{c_n}\right)^r &= \left(v_1w_0^{'}\right)^{rc_0}\cdots\left(v_1w_n^{'}\right)^{rc_n} = v_1^{r(c_0+c_1+\cdots c_n)}\left(w_0^{'c_0}\cdots w_n^{'c_n}\right)^r \\ &= g^{k_5r(c_0+c_1+\cdots c_n)-k_1t_1h_1}g^{k_1t_1h_1}\cdot\left(w_0^{'}\cdots w_n^{'}\right)^{t_1h_1}w_0^{'b_0^{'}}\cdots w_n^{'b_n^{'}} = g^{k_3}\left(s_1w_0^{'}\cdots w_n^{'}\right)^{t_1h_1}w_0^{'b_0^{'}}\cdots w_n^{'b_n^{'}} \end{aligned}$$

⑤计算满足 $k_3 = k_5 r\left(c_0 + c_1 + \ldots + c_n\right) - k_1 t_1 h_1$ 和 $rc_i = b_i^{'} + t_1 h_1$ 的 $b_i^{'}$ 和 $t_1 h_1$,其中 $i \in \{0,\ldots,n\}$.

⑥令验证密钥为 $\left\{g^{k_2}, g^{k_3}, r, t_0, t_1\right\}$,求值密钥为 $\left\{\left(h_0, s_0 w_0 \cdots w_n\right), \left(h_1, s_1 w_0^{'} \cdots w_n^{'}\right), \left(b_i, w_i\right)_{i\in[n]}, \left(b_i^{'}, w_i^{'}\right)_{i\in[n]}\right\}$.

2) $ME.Compute\left(MExp.EK\right) \to \left(ME.\sigma_y, ME.\pi_y\right)$

此步骤由服务器生成编码后的结果 $ME.\sigma_y$ 和证据 $ME.\pi_y$.具体如下:

①将 $ME.EK$ 分解为 $\left(h_0, s_0 w_0 \cdots w_n\right)$, $\left(h_1, s_1 w_0^{'} \cdots w_n^{'}\right)$, $\left\{\left(b_i, w_i\right)_{i\in[n]}\right\}$ 和 $\left\{\left(b_i^{'}, w_i^{'}\right)_{i\in[n]}\right\}$.

②对于 $i \in \{0,\ldots,n\}$,计算 $w_i^{b_i}$ 和 $w_i^{'b_i^{'}}$.

③令 $R_0 = \left(s_0 w_0 \cdots w_n\right)^{h_0}$ 和 $R_1 = \left(s_1 w_0^{'} \cdots w_n^{'}\right)^{h_1}$.

④返回结果和证据,分别为 $ME.\sigma_y = \left\{\left\{w_i^{b_i}\right\}_{i\in[n]}, R_0\right\}$ 和 $ME.\pi_y = \left\{\left\{w_i^{'b_i^{'}}\right\}_{i\in[n]}, R_1\right\}$.

3) $ME.Verify\left(ME.VK, ME.\sigma_y, ME.\pi_y\right) \to \left\{ME.y, \perp\right\}$

此步骤为客户端验证计算结果的正确性,具体如下:

①将 $ME.VK$ 分解为 $\left\{g^{k_2}, g^{k_3}, r, t_0, t_1\right\}$, $ME.\sigma_y$ 分解为 $\left\{\left\{w_i^{b_i}\right\}_{i\in[n]}, R_0\right\}$, $ME.\pi_y$ 分解为 $\left\{\left\{w_i^{'b_i^{'}}\right\}_{i\in[n]}, R_1\right\}$.

②计算是否满足等式

$$\left(g^{k_2} R_0^{t_0} w_0^{b_0} w_1^{b_1} \cdots w_n^{b_n}\right)^r = g^{k_3} R_1^{t_1} w_0^{'b_0^{'}} \cdots w_n^{'b_n^{'}}.$$

若不满足则终止协议并输出 $\perp$,否则恢复计算结果 $ME.y = g^{k_2} R_0^{t_0} w_0^{b_0} w_1^{b_1} \cdots w_n^{b_n}$.

MExp 协议有以下的性质:

1) 正确性:若服务器是诚实地遵循上述过程, 协议可确保客户端总是可以输出 $ME.y$.

2) 零知识性:协议可确保一个不遵守协议的恶意服务器无法获得任何秘密输入和输出(运算结果)的信息.

3) α-可检查性:协议确保检测到恶意服务器返回的错误结果的概率不小于 α.

3.2 方案详细设计

本节首先详细介绍了 PVDFD 方案中各个算法的具体构造,然后给出了正确性证明.下面分别进行描述.

1) 初始化算法.

该算法用于为方案中的所有实体生成公共参数.算法的主要步骤如下:

①给定安全参数 λ,可信第三方首先选择阶为 $p \in poly(\lambda)$ 的两个群 $\mathcal{G}_1$ 和 $\mathcal{G}_2$ 满足双线性映射 $e: \mathcal{G}_1 \times \mathcal{G}_1 \to \mathcal{G}_2$.

②随机选取两个生成元 $g, u \in \mathcal{G}_1$.

③令公共参数 $pp = \left(p, g, u, \mathcal{G}_1, \mathcal{G}_2, e\right)$ 并发送给云服务器、数据拥有者和授权用户.

2) 密钥生成算法.

该算法用于生成方案所需密钥.算法的主要步骤如下:

①数据库 $DB=\{(i,v_i)\mid i\in\{0,\ldots,n\}\}$,其中数据集大小为 $\ell=n+1$.数据拥有者先将数据库 DB 插值为多项式 $\mathcal{F}(x)=\sum_{i=0}^{n}c_i x^i$ 的形式,使得对于任意的 $i\in\{0,\ldots,n\}$ 均满足 $\mathcal{F}(i)=v_i$.提取多项式 $\mathcal{F}$ 的系数集合为 $C=\{c_0,c_1,\ldots,c_n\}$.

②选择两个随机数 $\gamma,k\in\mathcal{Z}_p$.令验证密钥的检索密钥 $\mathcal{RK}_{pp}=k$,数据库的查询密钥 $\mathcal{EK}_{DB}=(C,\gamma)$.

③选择随机数 $\alpha\in\mathcal{Z}_p$ 作为伪随机函数(PRF)的密钥,并建立伪随机函数 $PRF(\alpha,t)$ 为: $PRF(\alpha,t)=g^{k\alpha^{t+1}}$.

④记 j 为模幂操作的索引,对于 $\forall j\in[2n+1]\backslash\{n+1\}$,执行 $MExp$ 协议的初始化算法 $ME.Setup(u,\alpha^{j+1},p)$ 生成模幂计算求值密钥 $ME.EK[j]$ 和验证密钥 $ME.VK[j]$.其中 u 和 α^{j+1} 分别作为底和幂.当 $j=2n+2$ 时,执行算法 $ME.Setup(F,C,p)$ 生成求值密钥 $ME.EK[j]$ 和验证密钥 $ME.VK[j]$,其中 $F=\{PRF(\alpha,i)\mid i=0,1,\ldots,n\}$ 为底.

⑤令验证密钥的恢复密钥 $\mathcal{VK}_{pp}=\{ME.VK[j]|_{0\le j\le 2n+2\wedge j\ne n+1}\}$、验证密钥的计算密钥 $\mathcal{EK}_{pp}=\{ME.EK[j]|_{0\le j\le 2n+2\wedge j\ne n+1}\}$.其中,$ME.VK[j]=\{g^{k_{j,2}},g^{k_{j,3}},r_j,t_{j,0},t_{j,1}\}$,$ME.EK[j]=\{(b_{j,i},w_{j,i})_{i\in[n]},(b'_{j,i},w'_{j,i})_{i\in[n]},(h_{j,0},s_{j,0}w_{j,0}\cdots w_{j,n}),(h_{j,1},s_{j,1}w'_{j,0}\cdots w'_{j,n})\}$.

⑥最终,数据拥有者保持 PRF 密钥 α,$\mathcal{EK}_{DB}$,$\mathcal{VK}_{pp}$ 和 $\mathcal{RK}_{pp}$ 私有,并将 $\mathcal{EK}_{pp}$ 发送给云服务器.值得注意的是,数据拥有者在执行算法 $ME.Setup$ 时只选择一次随机的六元组,在后面的操作中均都使用该六元组.也就是说,对于所有的 $i\in[5]$ 和 $j\in[2n+2]\backslash\{n+1\}$ 均满足 $k_{j,i}=k_{0,i}$.

3) 验证密钥计算算法.

该算法根据接收到的验证密钥的计算密钥来计算编码后的验证密钥.算法的主要步骤如下:

①云服务器收到验证密钥的计算密钥 $\mathcal{EK}_{pp}$ 后,将其分解为 $\{ME.EK[j]|_{0\le j\le 2n+2\wedge j\ne n+1}\}$.

②云服务器执行算法 $ME.Compute(ME.EK[j])$ 生成编码后的结果 $ME.\sigma_{y[j]}$ 和证据 $ME.\pi_{y[j]}$.

③令编码后的数据库验证密钥 $\sigma_{\mathcal{VK}_{DB}}$、证据 $\pi_{\mathcal{VK}_{DB}}$ 分别为 $\{ME.\sigma_{y[j]}|_{0\le j\le 2n+2\wedge j\ne n+1}\}$,$\{ME.\pi_{y[j]}|_{0\le j\le 2n+2\wedge j\ne n+1}\}$,其中,$ME.\sigma_{y[j]}=\{\{w_{j,i}^{\,b_{j,i}}\},R_{j,0}\}$,$ME.\pi_{y[j]}=\{\{w'^{\,b'_{j,i}}_{j,i}\},R_{j,1}\}$.

④最终云服务器将 $\sigma_{\mathcal{VK}_{DB}}$ 和 $\pi_{\mathcal{VK}_{DB}}$ 发给数据拥有者.

4) 验证密钥恢复算法.

该算法用于对云服务器返回的结果进行验证并恢复出数据库的验证密钥.算法的主要步骤如下:

①数据拥有者将 $\mathcal{VK}_{pp}$、$\sigma_{\mathcal{VK}_{DB}}$ 和 $\pi_{\mathcal{VK}_{DB}}$ 分别解析为: $\{ME.VK[j]|_{0\le j\le 2n+2\wedge j\ne n+1}\}$,$\{ME.\sigma_{y[j]}|_{0\le j\le 2n+2\wedge j\ne n+1}\}$ 和

$\left\{ME.\pi_{y[j]}\mid_{0\le j\le 2n+2\wedge j\ne n+1}\right\}$.

②执行算法 $ME.Verify\left(ME.VK[j],ME.\sigma_{y[j]},ME.\pi_{y[j]}\right)$.

③当 $j\in[2n+2]\setminus\{n+1\}$ 时,数据拥有者验证是否满足如下等式:

$$\left(g^{k_{j,2}}R_{j,0}^{t_{j,0}}w_{j,0}{}^{b_{j,0}}\right)^{r_j}=\left(g^{k_{j,3}}R_{j,1}^{t_{j,1}}w'_{j,1}{}^{b'_{j,1}}\right). \quad (1)$$

若不满足则输出 $\perp$ 并丢弃,否则令

$$U_j=ME.y[j]=\left(g^{k_{j,2}}R_{j,0}^{t_{j,0}}w_{j,0}{}^{b_{j,0}}\right)=u^{\alpha^{j+1}}.$$

当 $j=2n+2$ 时,验证是否满足等式

$$\left(g^{k_{j,2}}R_{j,0}^{t_{j,0}}\prod_{i=0}^{n}w_{j,i}{}^{b_{j,i}}\right)^{r_j}=\left(g^{k_{j,3}}R_{j,1}^{t_{j,1}}\prod_{i=0}^{n}w'_{j,i}{}^{b'_{j,i}}\right). \quad (2)$$

若不满足则输出 $\perp$ 并丢弃,否则令

$$\sigma_{DB}=g^{\gamma}ME.y[j]^{\mathcal{RK}_{pp}^{-1}}=g^{\gamma}\left(g^{k_{j,2}}R_{j,0}^{t_{j,0}}\prod_{i=0}^{n}w_{j,i}{}^{b_{j,i}}\right)^{\mathcal{RK}_{pp}^{-1}}=g^{\gamma}\prod_{i=0}^{n}g^{c_i\alpha^{i+1}}.$$

④数据拥有者将 $\left\{U_j\right\}_{j\in[2n+2]\setminus\{n+1\}}$ 添加到数据库的查询密钥 $\mathcal{EK}_{DB}$ 中,并将更新后的该值发送给云服务器,此时 $\mathcal{EK}_{DB}=\left\{C,\gamma,\left\{U_j\right\}_{j\in[2n+2]\setminus\{n+1\}}\right\}$.

⑤数据拥有者最终令 $\mathcal{VK}_{DB}=\left\{\sigma_{DB},\left\{U_j\right\}_{j\in[2n+2]\setminus\{n+1\}},\mathcal{RK}_{pp},PRF(\alpha,0)\right\}$ 并通过安全的信道发送给授权的用户.

5) 查询算法.

该算法用于查询数据库即执行多项式求值计算.算法的主要步骤如下:

①当云服务器收到数据拥有者发来的 $\mathcal{EK}_{DB}$ 以及授权用户发来的查询索引 x 后,首先将 $\mathcal{EK}_{DB}$ 分解为 $\left\{C,\gamma,\left\{U_j\right\}_{j\in[2n+2]\setminus\{n+1\}}\right\}$,并令 $X=\left\{1,x,x^2,\ldots,x^n\right\}$.

②计算结果 $y=\mathcal{F}(x)=\sum_{i=0}^{n}c_ix^i$.

③计算相关的证据 $\pi_y=\prod_{i=0}^{n}W_i^{x^i}$,其中 $W_i=U_{n-i}^{\gamma}\cdot\prod_{j=0,j\ne i}^{n}U_{n+1+j-i}^{c_j}$.

④将结果 y 和证据 π_y 发回给授权的用户.

6) 验证算法.

该算法用于对云服务器返回的查询结果进行验证.算法的主要步骤如下:

①授权用户收到结果 y 和证据 π_y 后,首先将 $\mathcal{VK}_{DB}$ 分解为 $\left\{\sigma_{DB},\left\{U_j\right\}_{j\in[2n+2]\setminus\{n+1\}},\mathcal{RK}_{pp},PRF(\alpha,0)\right\}$.

②计算 $\prod_{i=0}^{n}U_{n-i}^{x^i}$.

③验证以下等式是否成立:

$$e\left(\sigma_{DB},\prod_{i=0}^{n}U_{n-i}^{x^i}\right)=e\left(g,\pi_y\right)\cdot e\left(PRF\left(\alpha,0\right),U_n\right)^{y\cdot\mathcal{RK}_{pp}^{-1}}\ .(3)$$

若成立则接受并输出 1,否则拒绝并输出 0.

注: 本方案亦可直接用于外包多项式的可验证计算,此时多项式的更新效率为常量级.假设 $\mathcal{F}$ 为 n 阶的原外包多项式, $\mathcal{F}'$ 为系数更新后的外包多项式, $\sigma_{\mathcal{F}}$ 对应上文中的 σ_{DB} , $\mathcal{VK}_{\mathcal{F}}$ 对应上文中的 $\mathcal{VK}_{DB}$ 表示外包多项式的验证密钥, *upd* 为更新信息.具体更新算法为:

$$Update\left(\mathcal{VK}_{\mathcal{F}},\mathcal{RK}_{pp},\mathcal{F}'\right)\rightarrow\left(upd,\mathcal{VK}_{\mathcal{F}'}\right)$$

该算法在保证多项式的阶不变的情况下, 可高效地对多项式的任意系数进行更新.算法的主要步骤如下:

①假设更新的系数位置为 $i\in\{0,\ldots,n\}$,令 $upd=\left(i,c_i,c_i'\right)$.其中 c_i 为系数更改前的值, c_i' 为更改后的值.

②计算 $\sigma_{\mathcal{F}'}=\sigma_{\mathcal{F}}PRF\left(\alpha,i\right)^{\left(c_i'-c_i\right)/\mathcal{RK}_{pp}}$ 并将 $\mathcal{VK}_{\mathcal{F}}$ 更新为 $\mathcal{VK}_{\mathcal{F}'}$.

③外包多项式的用户将 *upd* 发送给云服务器,并将更新后的验证密钥 $\mathcal{VK}_{\mathcal{F}'}$ 重新发送给授权用户即可实现外包多项式的更新.

3.3 正确性证明

本节将基于公式 1,2,3 的正确性来证明整个方案的正确性. 公式 1,2 可根据文献[31]来证明, 而对于公式 3, 其等式的左边(left-hand side, LHS)有:

$$LHS=e\left(\sigma_{DB},\prod_{i=0}^{n}U_{n-i}^{x^i}\right)=\prod_{i=0}^{n}e\left(\sigma_{DB},U_{n-i}^{x^i}\right)=\prod_{i=0}^{n}e\left(\sigma_{DB},U_{n-i}\right)^{x^i}$$

$$\begin{aligned}e\left(\sigma_{DB},U_{n-i}\right)^{x^i}&=e\left(g^{\gamma}\cdot\prod_{j=0}^{n}g^{c_j\alpha^{j+1}},U_{n-i}\right)^{x^i}\\&=\left(e\left(g^{\gamma},U_{n-i}\right)\cdot e\left(\prod_{j=0,j\neq i}^{n}g^{c_j\cdot\alpha^{j+1}},U_{n-i}\right)\cdot e\left(g^{c_i\cdot\alpha^{i+1}},U_{n-i}\right)\right)^{x^i}\\&=\left(e\left(g,U_{n-i}^{\gamma}\right)\cdot e\left(g,\prod_{j=0,j\neq i}^{n}U_{n-i}^{c_j\cdot\alpha^{j+1}}\right)\cdot e\left(g^{c_i\cdot\alpha^{i+1}},U_{n-i}\right)\right)^{x^i}\\&=\left(e\left(g,U_{n-i}^{\gamma}\right)\cdot e\left(g,\prod_{j=0,j\neq i}^{n}u^{\alpha^{n-i+1}c_j\cdot\alpha^{j+1}}\right)\cdot e\left(g^{c_i\cdot\alpha^{i+1}},u^{\alpha^{n-i+1}}\right)\right)^{x^i}\\&=\left(e\left(g,U_{n-i}^{\gamma}\right)\cdot e\left(g,\prod_{j=0,j\neq i}^{n}u^{c_j\cdot\alpha^{n+j-i+2}}\right)\cdot e\left(g^{\alpha^{i+1}},u^{\alpha^{n-i+1}}\right)^{c_i}\right)^{x^i}\end{aligned}$$

,

$$\begin{aligned}LHS&=\prod_{i=0}^{n}e\left(\sigma_{DB},U_{n-i}\right)^{x^i}\\&=\prod_{i=0}^{n}\left(e\left(g,U_{n-i}^{\gamma}\right)\cdot e\left(g,\prod_{j=0,j\neq i}^{n}u^{c_j\cdot\alpha^{n+j-i+2}}\right)\cdot e\left(g^{\alpha^{i+1}},u^{\alpha^{n-i+1}}\right)^{c_i}\right)^{x^i}\\&=\prod_{i=0}^{n}\left(e\left(g,U_{n-i}^{\gamma}\right)\cdot\left(e\left(g,\prod_{j=0,j\neq i}^{n}U_{n+j-i+1}^{c_j}\right)\cdot e\left(g^{\alpha},u^{\alpha^{n+1}}\right)^{c_i}\right)\right)^{x^i}\\&=\prod_{i=0}^{n}\left(e\left(g,U_{n-i}^{\gamma}\right)\cdot\left(e\left(g,\prod_{j=0,j\neq i}^{n}U_{n+j-i+1}^{c_j}\right)\cdot e\left(g^{\alpha},U_n\right)^{c_i}\right)\right)^{x^i}\\&=\prod_{i=0}^{n}\left(e\left(g,U_{n-i}^{\gamma}\cdot\prod_{j=0,j\neq i}^{n}U_{n+j-i+1}^{c_j}\right)\cdot e\left(g^{\alpha},U_n\right)^{c_i}\right)^{x^i}\\&=\prod_{i=0}^{n}\left(e\left(g,W_i\right)\cdot e\left(g^{\alpha},U_n\right)^{c_i}\right)^{x^i}\end{aligned}$$

又由公式 3 等式的右边(right-hand side, RHS)可知:

$$RHS=e\left(g,\prod_{i=0}^{n}W_i^{x^i}\right)\cdot e\left(PRF\left(\alpha,0\right),U_n\right)^{\mathcal{RK}_{pp}^{-1}\cdot\sum_{i=0}^{n}c_ix^i}=e\left(g,\prod_{i=0}^{n}W_i^{x^i}\right)\cdot e\left(PRF\left(\alpha,0\right)^{\mathcal{RK}_{pp}^{-1}},\prod_{i=0}^{n}U_n^{c_i}\right)^{x^i}$$

$$=\prod_{i=0}^{n}\left(e\left(g,W_i\right)\cdot e\left(g^{k\alpha\mathcal{RK}_{pp}^{-1}},U_n^{c_i}\right)\right)^{x^i}=\prod_{i=0}^{n}\left(e\left(g,W_i\right)\cdot e\left(g^{\alpha},U_n\right)^{c_i}\right)^{x^i}$$

即 $LHS=RHS$,公式 3 成立.因此,若方案步骤都是正确执行,产生的结果都是未被篡改的,则结果 $\left(\sigma_{\mathcal{VK}_{DB}},\pi_{\mathcal{VK}_{DB}}\right)$ 和 (y,π_y) 总能分别通过数据拥有者及授权用户的验证.

4 安全性证明

定理 1. 若存在一个概率多项式时间的敌手 $\mathcal{A}$,满足 $Adv_{\mathcal{A}}^{\mathrm{Privacy}}(\mathrm{PVDFD},\lambda)\geq negl(\lambda)$,则他可以创建一个有效的算法 $\mathcal{B}$ 来打破可验证外包模幂计算的零知识性.

证明.根据文献[31],若敌手 $\mathcal{A}$ 已知公共参数信息,且能够构建一个算法 $\mathcal{B}$ 区分计算结果和哪一个输入相关,即打破了可验证外包模幂计算的零知识性.

敌手 $\mathcal{A}$ 的挑战过程如下,已知公共参数为 pp ,敌手通过模拟器随机构造两个数据库的计算密钥 $\mathcal{EK}_{DB_0}$ 和 $\mathcal{EK}_{DB_1}$,并将其发送给挑战者.挑战者随机选择其中一个密钥 $\mathcal{EK}_{DB_b}$,调用算法 $VKeval\left(pp,\mathcal{EK}_{DB_b}\right)$ 执行模幂运算,并将结果 $\sigma_{\mathcal{VK}_{DB_b}}$ 和证据 $\pi_{\mathcal{VK}_{DB_b}}$ 返回给敌手.敌手收到结果和证据后,利用算法 $\mathcal{B}$ 猜测出 b 的值,即区分出计算结果与哪一个计算请求相关.由此可知,敌手使用了有效的算法打破了可验证外包模幂计算的零知识性. 证毕.

定理 2. 若存在一个概率多项式时间的敌手 $\mathcal{A}$,满足 $Adv_{\mathcal{A}}^{VKU}(\mathrm{PVDFD},\lambda)\geq negl(\lambda)$,则他可以创建一个有效的算法 $\mathcal{B}$ 来打破可验证外包模幂计算的 α -可检查性.

证明.根据文献[31],若敌手 $\mathcal{A}$ 伪造了结果和证据,分别将其中的 R_0 和 R_1 替换为为 R_0^* 和 R_1^* ,且满足 $\left(g^{k_2}R_0^{*t_0}w_0^{b_0}\cdots w_n^{b_n}\right)^r=g^{k_3}R_1^{*t_1}w_0^{'b_0'}\cdots w_n^{'b_n'}$,则伪造成功.敌手 $\mathcal{A}$ 利用伪造信息和真实的值可以构建一个算法 $\mathcal{B}$ 来打破可验证外包模幂计算的 α -可检查性.

具体构建过程如下,敌手 $\mathcal{A}$ 随机选择值 R 满足 $R^{t_0r}=R^{t_1}$.令 $R_0^*=R\cdot R_0$, $R_1^*=R\cdot R_1$,根据等式

$$\left(g^{k_2}w_0^{b_0}\ldots w_n^{b_n}\left(R\left(s_0w_0\ldots w_n\right)^{h_0}\right)^{t_0}\right)^r=g^{k_3}w_0^{'b_0'}\ldots w_n^{'b_n'}\left(R\left(s_1w_0'\ldots w_n'\right)^{h_1}\right)^{t_1}$$

可知敌手用伪造的结果和证据,满足了等式 $\left(g^{k_2}R_0^{*t_0}w_0^{b_0}\cdots w_n^{b_n}\right)^r=g^{k_3}R_1^{*t_1}w_0^{'b_0'}\cdots w_n^{'b_n'}$,通过了客户端的验证.因此敌手使用有效的算法打破了可验证外包模幂计算的 α -可检查性. 证毕.

定理 3. 若存在一个概率多项式时间的敌手 $\mathcal{A}$,满足 $Adv_{\mathcal{A}}^{RU}(\mathrm{PVDFD},DB,\lambda)\geq negl(\lambda)$,则他可以创建一个有效的算法 $\mathcal{B}$ 来打破 BDHE 难题.

证明.查询索引 x^* 对应的值为 y^* ,敌手 $\mathcal{A}$ 伪造了该值和证据分别为 $\hat{y}^*$, $\hat{\pi}_{y^*}$.若伪造的值满足 $\hat{y}^*\neq y^*$ 且 $Verify\left(params,\mathcal{VK}_{DB},x^*,\hat{y}^*,\hat{\pi}_{y^*}\right)=1$,则伪造成功.敌手 $\mathcal{A}$ 利用伪造信息和真实的查询结果及证据可以构建一个算法 $\mathcal{B}$ 来打破 BDHE 难题.

具体构建过程如下,令 $c=\hat{y}^*-y^*\neq 0\in\mathcal{Z}_p$,即真值与伪造值的差值.由于验证通过,则一定满足以下等式:

$$e\left(\sigma_{DB},\prod_{i=0}^{n}U_{n-i}^{x^i}\right)=e\left(g,\pi_{\hat{y}^*}\right)\cdot e\left(PRF(\alpha,0),U_n\right)^{\hat{y}^*\cdot\mathcal{RK}_{pp}^{-1}} \quad (4)$$

而对于真实结果和证据$\left(y^*,\pi_{y^*}\right)$,满足以下等式:

$$e\left(\sigma_{DB},\prod_{i=0}^{n}U_{n-i}^{x^i}\right)=e\left(g,\pi_{y^*}\right)\cdot e\left(PRF\left(\alpha,0\right),U_n\right)^{y^*\cdot\mathcal{RK}_{pp}^{-1}} \quad (5)$$

根据公式 4 和公式 5,可以推出:

$$e\left(g,\pi_{y^*}\right)\cdot e\left(PRF\left(\alpha,0\right),U_n\right)^{y^*\cdot\mathcal{RK}_{pp}^{-1}}=e\left(g,\pi_{\hat{y}^*}\right)\cdot e\left(PRF\left(\alpha,0\right),U_n\right)^{\hat{y}^*\cdot\mathcal{RK}_{pp}^{-1}}$$

$$e\left(g,\pi_{y^*}\right)\cdot e\left(g^{k\alpha},u^{\alpha^{n+1}}\right)^{y^*\cdot\mathcal{RK}_{pp}^{-1}}=e\left(g,\pi_{\hat{y}^*}\right)\cdot e\left(g^{k\alpha},u^{\alpha^{n+1}}\right)^{\hat{y}^*\cdot\mathcal{RK}_{pp}^{-1}}$$

$$e\left(g,\pi_{y^*}\right)\cdot e\left(g^{\alpha},u^{\alpha^{n+1}}\right)^{y^*}=e\left(g,\pi_{\hat{y}^*}\right)\cdot e\left(g^{\alpha},u^{\alpha^{n+1}}\right)^{\hat{y}^*}$$

$$e\left(g,\pi_{y^*}\right)\cdot e\left(g,u^{\alpha^{n+2}}\right)^{y^*}=e\left(g,\pi_{\hat{y}^*}\right)\cdot e\left(g,u^{\alpha^{n+2}}\right)^{\hat{y}^*}$$

$$e\left(g,\pi_{y^*}\right)\cdot e\left(g,U_{n+1}\right)^{y^*}=e\left(g,\pi_{\hat{y}^*}\right)\cdot e\left(g,U_{n+1}\right)^{\hat{y}^*}$$

$$e\left(g,U_{n+1}\right)=\left(\frac{e\left(g,\pi_{y^*}\right)}{e\left(g,\pi_{\hat{y}^*}\right)}\right)^{\frac{1}{\hat{y}^*-y^*}}=\left(\frac{e\left(g,\pi_{y^*}\right)}{e\left(g,\pi_{\hat{y}^*}\right)}\right)^{e^{-1}} \quad (6)$$

由公式 6 可知,敌手使用了有效的算法打破了 BDHE 难题. 证毕.

5 分析

5.1 方案对比

本节将本方案 PVDFD 与 Miao 等人的方案[24](MVDB)、Chen 等人的方案[18](CVDB)、Benabbas 等人的方案[6](BVDB)、Backes 等人的方案[21](BFVDB)和 Wang 等人的方案[23](WVDB)进行了功能上的对比分析.主要比较了方案是否需要较大的预处理开销、是否支持公共可验证和计算难题的假设.结果如表 2 所示.

从表 2 中可以看出,MVDB 方案、CVDB 方案和 BVDB 方案在预处理阶段均需要数据拥有者执行昂贵的计算.这对于持有小型终端的数据拥有者而言,代价过于昂贵.而 BFVDB 方案和 WVDB 方案虽不存在预处理开销大的问题,却不能实现公共可验证,BVDB 方案同样存在不支持公共可验证的问题.本文的 PVDFD 方案通过采用将预处理开销委托给云计算的方式,不需要数据拥有者过大的预处理开销,且支持公共可验证.

5.2 复杂度分析

BVDB 方案、BFVDB 方案和 WVDB 方案不能实现公共可验证,本文的 PVDFD 方案在功能方面优于三者,因此在本节中不做对比.本节主要对比了本方案和 CVDB 方案、MVDB 方案及不进行全委托计算的方案(Without Full Delegation Scheme)分别在预处理阶段、查询阶段和验证阶段的计算复杂度.其中,不进行全委托计算的方案(Without Full Delegation Scheme)是指将本方案中由云服务器和数据拥有者交互生成验证密钥的过程由数据拥有者独立完成,即数据拥有者独立计算验证密钥,而其他步骤不改变所形成的方案.由于插值操作、双线性映射和指数运算的计算开销较大,因此忽略其他的运算,仅对比三方案中的插值操作开销、双线性映射开销和指数运算开销,结果如表 3 所示.在表 3 中,ℓ 、 *Interplt* 、 *Pairing* 、 *Exp* 分别代表数据库中的数据集大小、插值操作的开销、双线性映射的开销以及指数运算的开销.

PVDFD 方案在预处理阶段,数据拥有者首先需要将数据库进行插值操作,并执行 6 次指数运算生成 *MExp* 协议中 6 个隐藏的对.其次执行 *VKrec* 算法,执行 2ℓ 次指数运算以恢复验证密钥.因此预处理阶段数据拥有者共需$\left(2\ell+6\right)$次指数运算和一次插值操作.若不进行全委托计算,在本地执行全部的密钥生成操作则需要进行一次

插值操作及 3ℓ 次指数运算.而 CVDB 方案基于向量承诺实现,MVDB 方案基于向量承诺和布隆过滤器实现,二者在预处理阶段所需代价均为$O(\ell^2)$.昂贵的平方量级的指数运算操作远远超出了数据拥有者的计算能力,尤其是当数据库中的数据集较大时,其开销是不可接受的.显然本文的 PVDFD 方案在预处理阶段的工作量最低,证明了全委托方案的优越性.而在用户验证阶段,CVDB 方案和 MVDB 方案的开销较小为常量级,本方案和不进行全委托的方案的开销均要执行与数据库大小呈线性量级的指数运算,但线性量级的开销是用户可以接受的.CVDB 方案和 MVDB 方案虽然在验证过程中开销较小,但在预处理阶段开销过大,使其仍无法实际应用.此外,由于云服务器可以认为是拥有海量计算资源的实体,拥有强大的计算能力.在 PVDFD 方案中,云服务器执行的任务量最大,是对其强大计算能力的充分利用.CVDB 方案、MVDB 方案和不进行全委托计算的方案对云服务器的利用较小,不能使其充分发挥作用以减小数据拥有者本地的开销.

Table 2 Comparison Among Six Schemes

表 2 方案对比

性能	MVDB[24]	CVDB[18]	BVDB[6]	BFVDB[21]	WVDB[23]	PVDFD
昂贵的预处理操作	√	√	√	×	×	×
公共可验证	√	√	×	×	×	√
计算假设	CDH	CDH	DDH	CDH	q-SDH	BDHE

Table 3 Comparison of Computation Cost

表 3 计算复杂度对比

实体	阶段	CVDB[18]	MVDB[24]	Without Full Delegation Scheme	PVDFD	
					算法	开销
数据拥有者	预处理	$\frac{\ell^2+3\ell+4}{2}Exp$	$\ell^2+3\ell+2\ Exp$	$3\ell Exp+Interplt$	*KeyGen*	$6Exp+Interplt$
					VKrec	$2\ell Exp$
					总计	$(2\ell+6)Exp+Interplt$
授权用户	验证	$1Exp+4Pairing$	$2Exp+4Pairing$	$(\ell+1)Exp+3Pairing$	*Verify*	$(\ell+1)Exp+3Pairing$
云服务器	查询	$(\ell-1)Exp$	$(\ell-1)Exp$	$\ell^2\ Exp$	*VKeval*	$(4\ell^2+4\ell)Exp$
					Query	$\ell^2\ Exp$
					总计	$(5\ell^2+4\ell)Exp$

5.3 效率分析

本节通过实验对比分析了本方案与 CVDB 方案、MVDB 方案及不进行全委托计算(Without FD Scheme,即上文所述数据拥有者独立计算验证密钥形成的方案)三种方案在预处理阶段、验证密钥生成过程、验证阶段和查询阶段的开销.实验采用 PBC 库实现双线性群的初始化,采用 NTL 库实现数据库的插值操作.运行实验程序的计算机配置为:Intel(R) Core(TM) i7-6700 CPU @ 3.40GHz 3.41GHz 主频的处理器和 8GB 内存.

1) 比较本方案与 CVDB 方案、MVDB 方案及不进行全委托计算的方案在预处理阶段的时间开销.预处理阶段包括初始化操作和密钥生成操作.实验结果如图 2 所示,其中横坐标表示外包数据库中的数据集大小,纵坐标表示三种方案在预处理阶段的执行时间.实验结果表明,本方案和不进行全委托计算的方案在预处理阶段时间开销较低.且当数据集越大时,两方案的时间开销越接近.这是由于数据集越大,数据库插值的时间则越长,指数运算的时间可以被忽略.而 MVDB 方案和 CVDB 方案的时间开销均远远高于二者,在数据集大小为 10000 时的开销甚至达到了约 100000 秒.其原因是 MVDB 方案和 CVDB 方案在预处理阶段需要过于昂贵的平方量级的指数运算操作,另两方案则只需线性量级的指数运算即可完成.当数据集较大时,MVDB 方案和 CVDB 方案的时间开销是持有小型终端的数据拥有者所不能接受的.

2) 比较本方案和不进行全委托计算的方案生成验证密钥的时间开销.实验结果如图 3 所示,其中横坐标表

示外包数据库中的数据集大小,纵坐标表示两种方案生成验证密钥的执行时间.实验结果表明,两方案的执行时间几乎都与数据集大小呈线性关系,而本方案生成验证密钥所需时间更低.由上文, 要计算验证密钥中的 σ_{DB} 和 $\{U_j\}$,不进行全委托计算的方案需要进行 3ℓ 次指数运算, ℓ 为数据库大小.而本方案执行验证密钥恢复算法,进行 2ℓ 次指数运算即可求得验证密钥.因此,本方案的开销更小,尤其当数据集大小较大时优势更加明显.

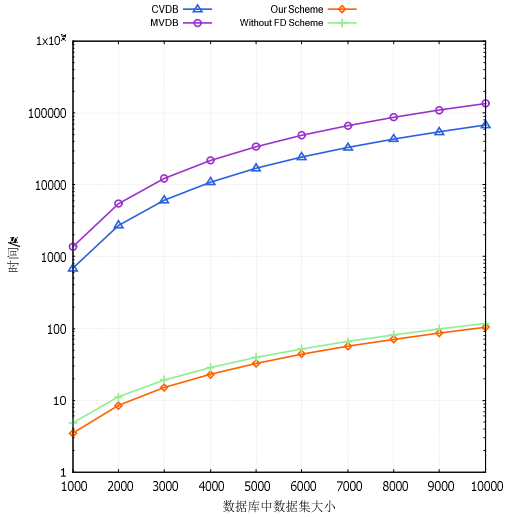

Fig.2 Comparison of Time Cost in Pre Phase
图 2 预处理阶段开销对比图

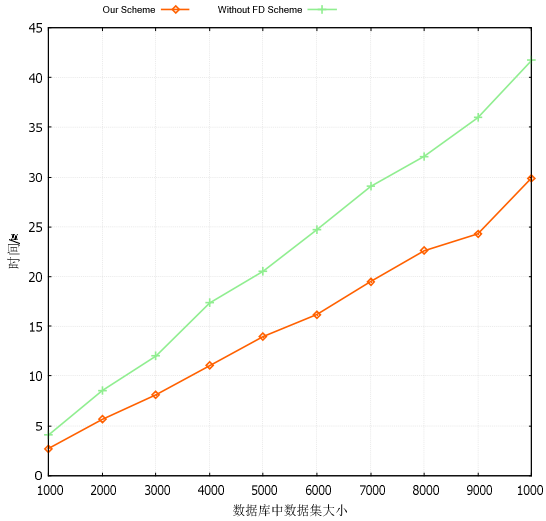

Fig.3 Comparison of Time Cost in VKEval Phase
图 3 计算验证密钥的开销对比图

3) 比较本方案与 CVDB 方案、MVDB 方案及不进行全委托计算的方案在验证阶段的时间开销.实验结果如图 4 所示,其中横坐标表示外包数据库中的数据集大小,纵坐标表示三种方案在用户验证阶段的执行时间.实验结果表明,随着数据集大小的增长,本方案和不进行全委托计算的方案所需的时间开销随之线性增长.而 CVDB 方案和 MVDB 方案中验证操作的时间开销则为常量级,独立于数据集的大小.这是由于本方案为了实现公共可验证性,使得用户执行线性量级的指数运算操作,但由图 4 中可看出,当数据集大小为 10000 时,时间开销约为 14 秒,这对于用户而言是完全可以接受的.而 CVDB 方案和 MVDB 方案由于将大部分开销都放在预处理阶段执行,因此在验证阶段开销较小,然而其在预处理操作中过于昂贵的开销使其难以实际应用.

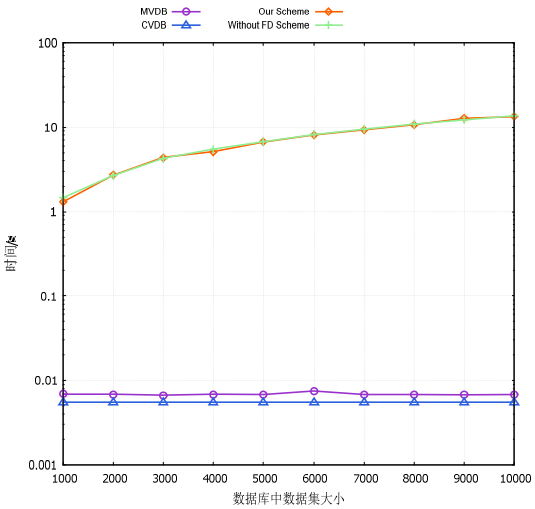

Fig.4 Comparison of Time Cost in Verify Phase
图 4 验证阶段开销对比图

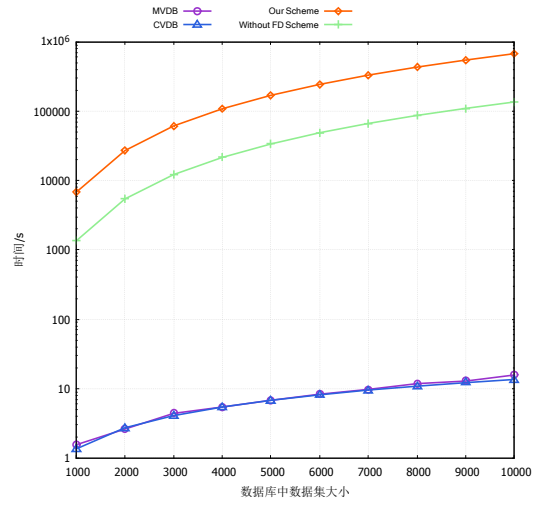

Fig.5 Comparison of Time Cost in Query Phase
图 5 查询阶段开销对比图

4) 比较本方案与 CVDB 方案、MVDB 方案及不进行全委托计算的方案在查询阶段的时间开销.实验结果如图 5 所示,其中横坐标表示外包数据库中的数据集大小,纵坐标表示三种方案在云服务器查询阶段的执行时间.实验结果表明,CVDB 方案和 MVDB 方案对云服务器的利用率最低,其次是不进行全委托计算的方案,而本方案对云服务器的利用率最高.这是由于 CVDB 方案和 MVDB 方案将大部分的开销放在了预处理阶段,数据拥有者执行昂贵的预处理操作,使得其查询和验证的开销较小.然而,云服务器的计算能力很强,数据拥有者的计算能力却是有限的,仅仅使用云服务器进行数据库的查询操作,是对其计算能力的浪费.此外,持有小型终端的数据拥有者执行如此昂贵的预处理开销是不现实的.因此,CVDB 和 MVDB 两方案没有利用好云服务器的特性减小数据拥有者的开销.而不进行全委托的计算方案同样没有利用好云服务器的计算能力.本文把预处理阶段全委托给云服务器计算,合理地利用了云服务器强大的计算能力,大大减少了数据拥有者在预处理阶段的时间开销,虽然使得用户的验证开销略有增长,但该增长是在用户接受范围内的,这使得本方案更适于实际应用.

以上实验结果表明,PVDFD 方案很好地利用了云服务器具有强大计算能力的特性,使得数据拥有者在预处理操作的开销上有很大的优势.尤其当数据集较大时,优势更加明显.其原因是,在本方案中,数据拥有者将预处理操作全委托给云计算,而本地只需执行密钥生成操作及验证密钥恢复操作即可,两个操作的执行时间之和仍小于对比文献及不进行全委托计算的方案执行预处理操作的时间,因此本方案实用性更强.

6 总结

本文提出了一个全委托的公共可验证的外包数据库方案 PVDFD.通过将预处理阶段的工作量委托给云服务器计算减少了数据拥有者的计算成本,且云无法获得任何与验证密钥相关的信息.此外,本方案支持公共可验证,使得任何持有验证密钥的授权用户均可向云服务器发出查询请求, 并可以根据云服务器返回的计算结果和证据进行验证, 实现了公共可验证.此外,本方案亦可用于外包多项式的可验证计算,且其更新的代价为常量级.本文还证明了 PVDFD 方案的正确性和安全性,最后通过理论和实验分析表明 PVDFD 方案解决了可验证数据库方案中数据拥有者预处理阶段开销过高及不能公共可验证的问题.

References:

[1] Mohammad B, Qahtan S, Yahya T. Cloud computing service models: A comparative study. In: 2016 International Conference on Computing for Sustainable Global Development. Piscataway, NJ: IEEE, 2019.

[2] Feng DG, Zhang M, Zhang Y, Xu Z. Study on Cloud Computing Security. Ruan Jian Xue Bao/Journal of Software, 2011, 22(1):71-83(in Chinese with English abstact).

[3] Wang Y, Li J, Wang HH. Cluster and cloud computing framework for scientific metrology in flow control. Cluster Computing, 2019, 22(1): 1189-1198.

[4] Baranwal G, Vidyarthi DP. A framework for selection of best cloud service provider using ranked voting method. In: 2014 IEEE International Advance Computing Conference (IACC). IEEE, 2014: 831-837.

[5] Kellaris G, Kollios G, Nissim K, O'neill, A. Generic attacks on secure outsourced databases. In: Proceedings of the 2016 ACM SIGSAC Conference on Computer and Communications Security. 2016: 1329-1340.

[6] Benabbas S, Gennaro R, Vahlis Y. Verifiable delegation of computation over large datasets. In: Annual Cryptology Conference. Berlin, Heidelberg: Springer, 2011: 111-131.

[7] Chen Z, Chen L, Zhong H. Towards secure and verifiable database-driven spectrum sharing. In: 2017 47th Annual IEEE/IFIP International Conference on Dependable Systems and Networks (DSN). IEEE, 2017: 285-296.

[8] Wang Q, Zhou FC, Chen CY, Wu QY. Secure Collaborative Publicly Verifiable Computation. IEEE Access, 2017, 5(99):2479-2488.

[9] Zhang Y, Katz J, Papamantho u C. IntegriDB: Verifiable SQL for outsourced databases. In: Proceedings of the 22nd ACM SIGSAC Conference on Computer and Communications Security. 2015: 1480-1491.

[10] Zhang Z, Chen X, Li J, Ma J. HVDB: a hierarchical verifiable database scheme with scalable updates. Journal of Ambient Intelligence and Humanized Computing, 2019, 10(8): 3045-3057.

[11] Kim KS, Jeong IR. Efficient verifiable data streaming. Security and Communication Networks, 2015, 8(18): 4013-4018.

[12] Guo Z, Li H, Cao C, Wei Z. Verifiable algorithm for outsourced database with updating. Cluster Computing, 2019, 22(3): 5185-5193.

[13] Song W, Wang B, Wang Q, Shi CL, Lou WJ, Peng ZY. Publicly verifiable computation of polynomials over outsourced data with multiple sources. IEEE Transactions on Information Forensics and Security, 2017, 12(10): 2334-2347.

[14] Chen X, Li H, Li J, Wang Q, Huang XY, Susilo W, Xiang Y. Publicly Verifiable Databases with All Efficient Updating Operations. IEEE Transactions on Knowledge and Data Engineering, 2020.

[15] Miao M, Ma J, Huang X, Wang Q. Efficient verifiable databases with insertion/deletion operations from delegating polynomial functions. IEEE Transactions on Information Forensics and Security, 2017, 13(2): 511-520.

[16] Zhang A, Huang S, Wu X, Wang SL, Li JH. An effective verifiable database protocol in severe untrusted environment. In: 2018 5th IEEE International Conference on Cyber Security and Cloud Computing (CSCloud)/2018 4th IEEE International Conference on Edge Computing and Scalable Cloud (EdgeCom). IEEE, 2018: 5-11.

[17] Eltayesh F. Verifiable Outsourced Database Model: A Game-Theoretic Approach. Montreal: Concordia University, 2017.

[18] Chen X, Li J, Huang X, Ma J, Lou W. New publicly verifiable databases with efficient updates. IEEE Transactions on Dependable and Secure Computing, 2014, 12(5): 546-556.

[19] Gennaro R, Gentry C, Parno B. Non-interactive verifiable computing: Outsourcing computation to untrusted workers. In: Annual Cryptology Conference. Berlin, Heidelberg: Springer, 2010: 465-482.

[20] Su Y, Sun J, Qin J, Hu J. Publicly Verifiable Shared Dynamic Electronic Health Record Databases with Functional Commitment Supporting Privacy-Preserving Integrity Auditing. IEEE Transactions on Cloud Computing, 2020.

[21] Backes M, Fiore D, Reischuk R M. Verifiable delegation of computation on outsourced data. In: Acm Sigsac Conference on Computer & Communications Security. New York: ACM, 2013:863-874.

[22] Wang Q, Xuan PK, Wang HW, Zhou FC. A Publicly Verifiable Outsourced Database Scheme with Full Operations. Journal Of Northeastern University Natural Science. 2018, v.39；No.335(08):37-41+52. (in Chinese with English abstact).

[23] Wang JF, Chen XF, Sun SF, Liu JK, Au MH, Zhan ZH. Towards Efficient Verifiable Conjunctive Keyword Search for Large Encrypted Database. In: European Symposium on Research in Computer Security. Springer, Cham, 2018.

[24] Miao M , Wang J , Wen S , Ma J. Publicly verifiable database scheme with efficient keyword search. Information Sciences, 2019, 475:18-28.

[25] Chen CY. Research and Implementation of Verifiable Outsourcing Polynomial Computation. Northeastern University, 2017.

[26] Catalano D , Fiore D . Vector Commitments and their Applications. In: International Workshop on Public Key Cryptography. Springer, Berlin, Heidelberg, 2013.

[27] Nguyen L. Accumulators from bilinear pairings and applications to id-based ring signatures and group membership revocation. Topics in Cryptology-CT-RSA 2005, 275-292.

[28] Au MH, Tsang PP, Susilo W, Mu Y. Dynamic universal accumulators for DDH groups and their application to attribute-based anonymous credential systems. In: Cryptographers' Track at the RSA Conference. Berlin, Heidelberg: Springer, 2009: 295-308.

[29] Dan B, Boyen X. Short Signatures Without Random Oracles and the SDH Assumption in Bilinear Groups. Journal of Cryptology, 2008, 21(2): 149-177.

[30] Boneh D, Boyen X, Goh E.J. Hierarchical identity based encryption with constant size ciphertext. In: Annual International Conference on the Theory and Applications of Cryptographic Techniques. Berlin, Heidelberg: Springer, 2005:440-456

[31] Ding Y, Xu Z, Ye J, Choo KKR. Secure outsourcing of modular exponentiations under single untrusted programme model. Journal of Computer and System Sciences, 2017, 90(C): 1-13.

附中文参考文献:

[2] 冯登国,张敏,张妍,徐震.云计算安全研究.软件学报,2011, 22(1):71-83.

[22]王强,玄鹏开,王红伟,周福才.支持全操作的公共可验证外包数据库方案.东北大学学报:自然科学版,2018, v.39; No.335(08):37-41+52

[25]陈春雨. 可验证的外包多项式计算的研究与实现. 东北大学, 2017.