计算机研究与发展
Journal of Computer Research and Development
DOI:10.7544/issn1000-1239.20210153
59(11):2635-2647, 2022

支持双向验证的动态密文检索方案

杜瑞忠[1,2] 王 一[1,2] 李明月[3]
[1](河北大学网络空间安全与计算机学院 河北保定 071002)
[2](河北省高可信信息系统重点实验室(河北大学) 河北保定 071002)
[3](南开大学计算机学院 天津 300350)
(limingyue@hbu.edu.cn)

Dynamic Ciphertext Retrieval Scheme with Two-Way Verification

Du Ruizhong[1,2], Wang Yi[1,2], and Li Mingyue[3]
[1](*School of Cyber Scurity and Computer, Hebei University, Baoding, Hebei* 071002)
[2](*Key Laboratory on High Trusted Information System in Hebei Province (Hebei University), Baoding, Hebei* 071002)
[3](*College of Computer Science, Nankai University, Tianjin* 300350)

Abstract The dynamic searchable encryption technology realizes the dynamic update of data, which can cope with more flexible application challenges, but the problem of privacy leakage and the dishonesty between users and cloud servers during data update have not been solved. In order to solve the above problem, a dynamic ciphertext retrieval scheme with two-way verification is proposed to achieve two-way verification between users and cloud servers. First, the introduction of bitmap index and homomorphic addition symmetric encryption technology, the use of bitmap index can represent all document identifiers involved in each update of a single keyword, reduce the number of cloud server searches and local index encryption times, thereby improve search and update efficiency, and the use of homomorphic addition symmetric encryption to encrypt the bitmap index can effectively protect the safe update of data. Secondly, the clients upload the aggregate MACs to the blockchain, and use the blockchain to verify the correctness of the results returned by the cloud server to prevent fraudulent behaviors between users and the cloud servers. Finally, the experimental results and security analysis show that the solution meets forward security and backward security, and improves efficiency in index building, search, update, and verification.

Key words two-way verification; dynamic searchable encryption; forward security; backward security; blockchain

摘 要 动态可搜索加密技术实现了数据动态更新,可以应对更加灵活多变的应用挑战,但是对于数据更新时产生的隐私泄露以及用户与云服务器的不诚实性问题并没有解决.为了解决上述问题,提出了一种支持双向验证的动态密文检索方案,实现用户与云服务器之间的双向验证.首先,引入位图索引以及同态加法对称加密技术,使用位图索引表示单个关键字每次更新涉及的所有文档标识符,减少了云服务

收稿日期:2021-02-26;**修回日期**:2022-03-18
基金项目:国家自然科学基金项目(61972073);河北省重点研发计划项目(22340701D);河北省自然科学基金项目(F2022201005)
This work was supported by the National Natural Science Foundation of China (61972073), the Key Research and Development Program of Hebei Province of China (22340701D), and the Natural Science Foundation of Hebei Province of China (F2022201005).

器搜索次数和本地索引加密次数,从而提高了搜索效率以及更新效率,并且利用同态加法对称加密对位图索引进行加密,可以有效地保护数据的安全更新.其次,将聚合消息认证码上传到区块链中,利用区块链对云服务器返回的结果进行正确性验证,防止用户和云服务器发生欺骗行为.最后,实验结果和安全分析表明,方案满足前向安全与后向安全,并且在索引生成、搜索、更新以及验证方面提高了效率.

关键词 双向验证;动态搜索加密;前向安全;后向安全;区块链

中图法分类号 TP309

云存储的按需索取和高效便捷等特点,使其在全世界范围内得到普及.为了保护数据的隐私性,在数据上传到云服务器之前,需要对其进行加密.然而,加密操作大大限制了外包数据的可用性.如何在保证数据隐私的前提下对加密数据进行搜索,成为亟待解决的问题.

可搜索加密技术支持用户在密文中进行关键字检索,且不会泄露关于原文的任何信息.早期的方案需要扫描整个加密数据集合才能返回结果,时长随着文件增加而呈现线性增长.为了提高搜索效率,许多学者对可搜索加密技术进行了改进,通过提取关键词构建索引,使得方案复杂度只与文件集中所包含的关键词相关.

早期的可搜索加密方案由于没有考虑实际的应用情况,都是基于静态环境下设计的.但是在现实情况中,存储在云服务器的数据并不是一成不变的,可能会对云服务器中存储的数据具有动态更新的要求.为了满足这些需求,动态可搜索加密技术应运而生.然而由于攻击者可以在更新期间观察数据库的变化,从而发现查询关键字与文件之间的关系,使得动态搜索加密技术的安全分析更加复杂.随着学者对其安全性的研究,前向安全与后向安全的概念被提出.前向安全保证了新添加的文档不会泄露之前搜索过的关键字信息;后向安全保证了删除的文档不会因为后续的搜索泄露信息.

除了保证方案的安全性以外,还需要对服务器的结果的正确性进行验证.目前,已经有很多可验证搜索加密方案被提出,这些方案均假定用户为诚实可信的实体,将验证操作交由用户去执行.但是恶意用户出于自私的目的,否认云服务器返回的正确结果,进而拒绝向其支付服务费.考虑到上述问题,本文提出了一种双向验证的动态密文检索方案,旨在解决动态环境下对用户与云服务器双向验证的问题.本文的主要贡献包括3个方面:

1) 利用位图技术构建索引,将文件标识符映射为位图中的项,并引入同态加密技术对索引进行加密,在满足前向安全与后向安全的同时,提高了搜索以及更新效率.

2) 将搜索与验证过程进行分离,由云服务器进行链下搜索,将验证过程交由区块链去处理.利用聚合消息认证码对关键字对应的加密结果进行验证,在保证用户与云服务器诚实操作外,使得区块链的存储安全且轻量.

3) 通过将本方案部署到本地私有测试网络Ganache中,并且对数据进行分梯度实验.实验结果分析表明,本方案在索引生成、搜索、验证以及更新效率方面均优于其他方案.

1 相关工作

Song等人[1]提出对称可搜索加密技术,但由于其搜索过程需要扫描整个数据集合才可以返回结果,其时长也随文件增加而呈现出线性增长的情况.Goh[2]通过布隆过滤器构建索引,在进行搜索时可以直接对索引进行匹配.虽然提升了搜索效率,但是由于布隆过滤器本身的特性,导致返回结果存在误差.Curtmola等人[3]提出一种倒排索引的搜索加密方案,这种方法提升了搜索效率,使得方案复杂度只与文件集中所包含的关键词相关.

随着动态可搜索加密技术[4]的提出,学者将注意力转移到了这个更加灵活的方案中.但是由于数据的更新可能带来更多信息泄露,为了防止这种情况发生,Stefanov等人[5]提出了关于前向安全和后向安全的概念.2016年,Bost[6]对前向安全进行了正式定义,并设计了第1个满足前向安全的方案.2017年,Bost等人[7]提出了同时满足前向安全和后向安全的可搜索加密方案,并在文中对后向安全等级从低到高分为Type-Ⅲ,Type-Ⅱ,Type-Ⅰ这3种类型.2018年,Sun等人[8]构造一种新型加密方式——对称刺穿加密,该方式使得服务器丧失对删除文档包含关键字的搜索能力,达到了后向安全.同年,Chamani等人[9]提出了3种方案,分别满足后向安

全的 3 种类型,对其效率和安全性进行了分析对比.Zuo 等人[10]提出一种满足更强后向安全的方案,并在补充方案中进行了分块操作,实现了大量数据更新.Vo 等人[11]利用 SGX 硬件结合批处理数据和压缩状态技术,减少了 SGX 与服务器的通信开销.He 等人[12]与 Demertzis 等人[13]就减少客户端存储开销问题,分别提出自己的方案.Zuo 等人[14]将文献[10]拓展到范围查询,在保证安全性的前提下丰富了动态搜索加密技术的功能.

在实际应用中发现,云服务器可能会因为外部攻击或者内部配置错误,变成恶意服务器返回给用户错误结果或者不完整结果.鉴于这种情况,Chai 等人[15]提出第 1 个可验证搜索加密方案,基于单词构造树为索引,对结果进行验证.随后,Soleimanian 等人[16]利用伪随机函数和单向函数,完成了对结果验证的公开化.Zhang 等人[17]在方案中引入多值 Hash 函数,提高了其方案的验证效率.Liang 等人[18]利用改进的 k 近邻算法技术对文档与关键词进行相关度分数计算,最后通过向量内积的方法对文档是否包含关键词进行验证.2020 年,Tong 等人[19]将 Merkle Hash Tree 与 k 均值聚类相结合,提出了 2 种方案,在提升验证效率的同时也提高了安全性.Miao 等人[20]将公共审计技术引入到方案中,由可信第三方进行审计并如实报告给用户.2020 年,Yang 等人[21]通过将验证过程转化为线性编程任务,解决了语义环境下的相关性验证问题.

以上方案均假定用户是可信的,会诚实地执行验证过程并发布验证结果.但是有些用户可能否认云服务器返回的正确结果,达到拒绝支付服务费的目的.随着区块链的出现,将可搜索加密技术与区块链相结合,确保了返回结果的正确性,从而无需用户进行验证.Shamshad 等人[22]将私有链与联盟链结合,将索引存储在联盟链中,而将密文存储在私有链中.Zhang 等人[23]将基于属性的索引上传到区块中,当用户满足属性策略时会获得区块地址,从而获得密文数据.Hoang 等人[24]通过创建隐藏数据列表,使得用户在获取数据时不会泄露身份以及交易信息.Li 等人[25]将二叉树存储在区块链中,并对其进行分块处理,实现了区块链中多关键字的搜索.但是以上方案都是在区块链中存储加密索引,由于区块链的燃料限制以及价格昂贵,在实际应用中受到了很多限制.最近,Guo 等人[26]将索引以及搜索过程交由链下处理,区块链仅对结果进行验证,并且实现了前向安全.Li 等人[27]分别对关键字和文档标识符构造索引,实现了对文档的链下删除.虽然以上方案在不同程度解决了云服务器与用户的验证问题,但是对动态更新安全并没有进行深入的考虑.

2 预备知识

2.1 位图索引技术

以往方案都是基于倒排索引构造,对于一次性更新大量数据效率低下且繁琐.基于上述问题的考虑,位图索引技术由此诞生.每个关键字对应一个字符串,字符串的长度代表数据库支持的最大文件数.其中若该关键字存在于文档 f_i,则将其对应第 i 位变为 1,否则设置为 0.为了方便阐述其构造,我们用图 1 进行举例说明.如图 1(a)是一个长度为 6 的字符串,此时表示关键字存在 f_4 和 f_2 两篇文档.如图 1(b)所示,现在我们想增加一篇文档 f_0 包含该关键字,需要构造其对应的字符串为 $2^0=000001$,然后与原始字符串进行加法.如图 1(c)所示,如果想删除文档 f_4,首先构造其对应的字符串为 $-2^4=-010000$.由于以上索引计算都需要将结果与 2^6 进行

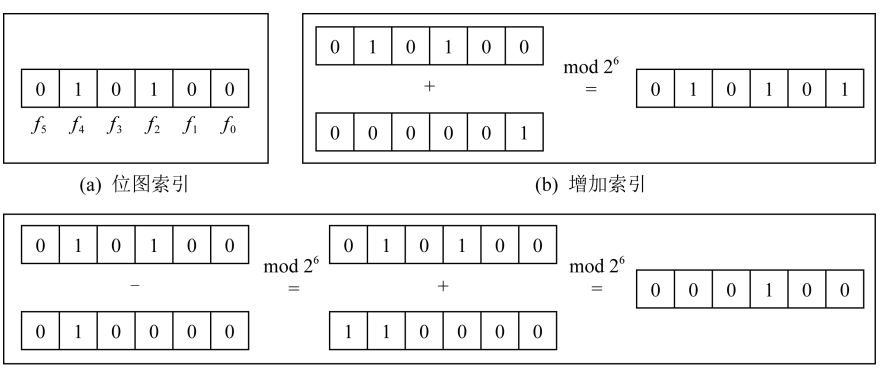

Fig. 1 Bitmap index and basic operations

图 1 位图索引及基本操作

模运算,所以可以将-2^4转化为$-2^4=2^6-2^4=48$,其对应二进制字符串为110000,随后与原始字符串相加,得到最终结果.通过上述运算,我们可以发现无论删除增加都可以利用模加法完成.

2.2 同态加法对称加密

由于位图索引的特殊构造,无论删除还是增加文档,都可以使用加法来实现,我们可以采用同态加法对称加密技术对数据进行处理,这样可以保证云服务器上加密数据的安全更新,保证了后向安全.同态加法对称加密方案Ⅱ主要由4个算法组成.

1) $n\leftarrow Setup(\lambda)$.输入安全参数$\lambda$,输出公共参数$n$,其中$n=2^m$代表整个明文消息空间大小,$m$为整个方案支持的最大文件数目.

2) $ct\leftarrow Enc(msg,sk,n)$.输入公共参数$n$,明文消息$msg(0\leqslant msg<n)$以及密钥$sk(0\leqslant sk<n)$.然后通过$ct=sk+msg \bmod n$计算密文,并且对于每次加密,密钥需要重新生成并保存在本地.

3) $msg\leftarrow Dec(ct,sk,n)$.输入公共参数$n$,密文$ct$以及密钥$sk$,然后通过计算$msg=ct-sk \bmod n$计算明文消息.

4) $\hat{ct}\leftarrow Add(ct_0,ct_1,n)$.输入密文$ct_0$与$ct_1$,计算$\hat{ct}\leftarrow ct_0+ct_1 \bmod n$,其中$ct_0\leftarrow Enc(msg_0,sk_0,n)$,$ct_1\leftarrow Enc(msg_1,sk_1,n)$.

正确性:对于计算明文$msg_0+msg_1 \bmod n$,需要2个密文之和$\hat{ct}=ct_0+ct_1 \bmod n$进行解密.首先计算一次性密钥$\hat{sk}\leftarrow sk_0+sk_1 \bmod n$,然后进行解密操作算法$Dec(\hat{ct},\hat{sk},n)$:

$$\hat{ct}-\hat{sk} \bmod n=msg_0+msg_1 \bmod n. \quad (1)$$

完美安全性[28]:如果对于任何敌手A,在完美安全游戏中的优势可以忽略,或者存在以下不等式.

$$Adv_{II,A}^{PS}=|Pr[A(Enc(msg_1,sk_1,n))]|-|Pr[A(Enc(msg_2,sk_2,n))]|\leqslant negl(\lambda), \quad (2)$$

其中$n\leftarrow Setup(\lambda)$,$negl(\lambda)$可忽略不计的,则本方案具有完美安全性.由于篇幅限制,同态加法对称加密可以在文献[28]中找到详细的完美安全性证明,本文不再赘述.

2.3 符号定义

符号的定义如表1所示.

2.4 区块链及智能合约

以太坊[29]是一种基于区块链的分布式可计算平台,其支持图灵完备的编程语言.以太坊的安全性主要依靠解决困难问题(或者说是区块),矿工会通过解决困难问题来获得奖励,保证了以太坊中的任何操作都是透明且可靠的.而智能合约作为以太坊中的一部分,也随之提出.

Table 1 Systematic Parameters

表1 系统参数

符号	意义
cs	搜索次数
c	更新次数
bs	当前存在文档对应的字符串
$S\in\{Y,N\}$	当前搜索状态
UT_c	最新的索引指针
l_c	当前检查列表标记
E_c	字符串bs对应的密文
Sum_e	加密字符串的总和
K_1,K_1^{-1}	当前搜索令牌/之前搜索令牌
Sk_c	一次性密钥
Sum_{sk}	一次性密钥的总和
F,H_1,H_2,H_3,H_4	伪随机函数

智能合约[30]是一种伴随着状态变化的应用,存在于区块链中.发布智能合约本质上就是将一笔交易发布到区块链,区块链中所有矿工会进行工作,当挖出包含此智能合约的区块时,智能合约也会有自己对应的地址,并且区块链中的所有节点都会保存此智能合约.而矿工的也会得到奖励,奖励由$gaslimit\times gasprice$所决定,其中$gaslimit$为发送方可以支出的最多燃料,$gasprice$为发送方为每个燃料支付的费用.正是由于这种特性,保证了智能合约的工作可靠性.对于所有需要在可信环境下进行的操作,理论上都可以使用智能合约来执行.

2.5 聚合消息认证码

聚合消息认证码(aggregate message authentication code, aggregate MAC)是一种通过减少数据消息认证码数量,达到高效地验证多条数据的技术.举例而言,给定2个数据消息认证码对(m_1,δ_1)和(m_2,δ_2),其中$\delta_i=Auth(K,m_i)$,$i\in\{1,2\}$.将2个消息认证码进行异或操作,得到聚合消息认证码$\delta^*=\delta_1\oplus\delta_2$,当对这2条数据$(m_1,m_2)$进行验证时,只需对聚合消息认证码$\delta^*$验证即可.利用这种技术,可以大大减少以太坊中数据量,避免超过$gaslimit$的限制.

3 系统模型

3.1 系统简介

系统模型如图2所示,分为3个实体部分,分别为客户端、云服务器以及区块链.

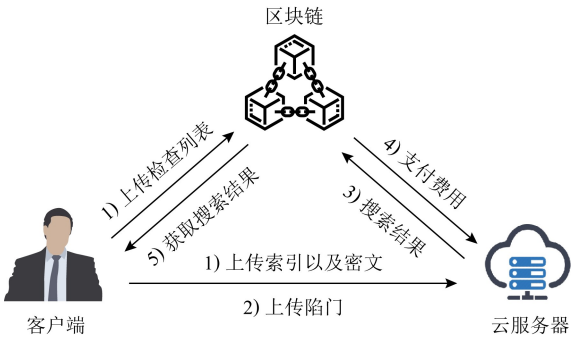

Fig. 2 Scheme model

图 2 方案模型图

1) 客户端.客户端既是数据拥有者也是用户,主要负责对数据以及索引进行加密,并将索引进行认证处理得到消息认证码.密文以及索引上传到云服务器,消息认证码上传到区块链中.同时,也可以对云服务器发送搜索请求,或者对原有数据进行动态更新.

2) 云服务器.云服务器主要负责存储索引和密文数据,并且会向客户端提供搜索服务,将结果发送到区块链中进行验证.

3) 区块链.区块链系统主要负责对云服务器上传的结果进行验证,如果结果正确并通过验证,说明结果是正确的,将搜索服务费支付给云服务器,并将结果永久保存.随后,客户端从区块链中获取加密结果.

为了保证系统内所有操作不可篡改且无法否认,所有的操作都是以交易的形式发布.区块链中的任何节点都可以对交易中的内容进行验证,使得客户端不可谎称结果错误而拒绝支付服务费,云服务器必须诚实执行搜索服务.

3.2 威胁模型

本文为了应对更加复杂的情况,假设整个方案置于不可信的环境进行设计,存在 3 种强有力的攻击者:

1) 由于区块链的公开透明特性,对于存储的数据以及进行操作都是公开的,可能存在潜在攻击者会对数据以及操作进行分析,获取各自之间的联系,进而对数据安全以及用户隐私产生威胁.

2) 客服端在请求服务器进行搜索服务后,可能会出于自私的目的,否认服务器返回的正确结果,进而拒绝向其支付服务费.

3) 云服务器可能会对存储本地的数据进行分析,获取一些敏感信息,还可能会出于某些自私的原因,提供给用户不可信的操作,其中主要涉及到更新以及搜索服务.当客户端提出请求后,云服务器拒绝提供服务并返回错误结果.

3.3 安全目标

基于 3.2 节所述的威胁模型,本方案应遵循以下安全目标,以达到保护用户隐私以及数据安全的目的.

1) 隐私性.由于云服务器和其他潜在的攻击者可能会对数据和索引进行分析,所以需要预先对数据以及索引进行加密处理,避免攻击者获取任何敏感信息.

2) 前向安全.前向安全指的是客户端上传了一篇新文档,这篇文档中包含之前搜索过的关键字,攻击者无法通过之前的陷门搜索到该文档,也就无法得到关键字与文档之间的关系.

3) 后向安全.后向安全指的是先前添加的文档被删除后,后续的搜索操作不会泄露该文档的信息.其中后向安全在文献[7]被分为 3 种类型,本方案比 Type-Ⅰ类型还要安全,即攻击者除了更新时间以及最后结果外,无法获得任何信息.

4) 验证性.需要对恶意用户以及恶意云服务器进行双向验证,将验证过程交由区块链去执行,保证了各个实体的诚实操作.

4 系统描述

4.1 设计理念

前向安全与后向安全作为动态搜索加密方案中的 2 个重要的安全属性,在更新时需要生成新的关键字令牌,并且保证攻击者无法知晓结果涉及的文档插入时间.而结果完整性验证则需要知道结果是否涉及同一关键字,这与前向安全和后向属性产生冲突.由于存储空间以及计算能力的限制,将结果交由用户验证是不切实际的.此外,用户还可能谎称错误结果,进而拒绝支付服务费.因此,本方案需要解决 2 个问题:1)前向安全、后向安全与结果验证之间的冲突;2)结果验证的可信问题.

根据后向安全的简单定义,只需要保证云服务器在搜索某个关键字时,无法搜索到先添加后删除的文档即可.我们可以改变索引以及加密方式,使得云服务器无法识别删除操作与增加操作的区别来达到后向安全.由于位图索引的增加和删除操作都可以使用模加法表示,然后使用同态加法对称加密技术对位图索引进行加密,使得在云服务器看来都是在添加加密数据,也就无法知道关键字与删除的

文档之间的关系.通过以上方法,将问题简化为前向安全与结果验证之间的冲突以及结果验证的可信问题.

首先,为了解决结果验证的可信问题,引入区块链系统作为第三方可信实体,客户端将预先定义的检查列表上传到智能合约中,对云服务器返回的加密结果进行验证.由于区块链的公开验证以及不可否认性,保证了验证结果的绝对可信,使得云服务器无法返回错误结果,客户端也无法谎称正确结果为错误的.

接下来需要解决前向安全与结果验证之间的冲突问题.对于本方案,我们需要保证云服务器存储的索引和区块链中存储的检查列表满足前向安全.根据前向安全的简单定义,我们可以简化为该加密数据在搜索过后进行了更新,需要生成新的标志指向该加密数据,而没有搜索过就进行更新的数据,则不需要生成新的标志.检查列表在每次更新时都需要生成新的标志,主要原因是:1)如果该关键字是搜索之后的首次更新,需要生成新的标志指向新的检查列表,防止智能合约搜索到之前的检查列表对结果进行验证,产生错误的验证结果.2)如果该关键字不是搜索之后的首次更新,需要搜索之前的检查列表,将新的加密索引对应的 MAC 与之前检查列表进行聚合,这种情况也需要生成新的标志指向新的检查列表.并且对于以上 2 种原因,为了防止攻击者测试从未查询过的加密结果与检查列表之间的关系,每次更新后检查列表需要插入一个新的随机数 α.同样,对于存储在云服务器中的加密索引,索引标志只有在搜索之后才会更新.

最后,对于验证通过的加密结果将被永久保存在区块链中,在该关键字没有发生更新的情况下,客户端可以直接从区块链中获取加密结果,避免了云服务器的重复搜索操作,提高了搜索效率.

如图 3 所示,其中 T_w^j 是关键字 w 对应的搜索令牌,L_w^j 为关键字 w 对应的检查列表的标志,bs_i 代表本次更新的位图索引,V_i 代表 bs_i 对应的 Hash 认证表示.可以看出一个完整的搜索过程是在搜索之后进行验证,在验证通过之后就会记录在区块链中.从图 3 可以看出 2 次更新之间的区别,第 1 次更新由于是在搜索之后首次更新,所以采用了新的搜索令牌 T_w^2,而第 2 次更新期间并没有进行搜索,所以也不需要更新搜索令牌,仍然使用搜索令牌 T_w^2,但是检查列表标志在每次更新期间都需要使用新的标志 L_w^j.

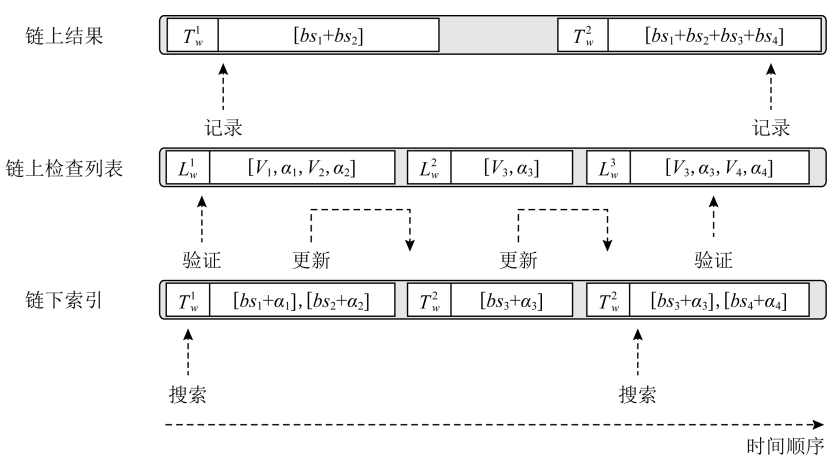

Fig. 3 Design structure

图 3 设计结构

4.2 具体方案

1) $Setup(1^\lambda)$.客户端在本地初始化系统,输入参数 λ,输出主密钥 $K\leftarrow\{0,1\}^\lambda$ 和一个整数 $n=2^m$,其中 m 为此方案可以容纳的最大文件数.初始化空集 Map 和 EDB,其中 Map 用来记录更新以及搜索状态,EDB 用来存储加密数据集.

2) $Buildindex(w,bs,Map,EDB)$.输入关键字 w 及其对应的字符串 bs.将更新次数 c 以及搜索次数 cs 置为 0,搜索状态 S 置为 N 表示没有被搜索.首先生成搜索令牌 K_1、密钥令牌 K_2 和检查列表标志 l_c.利用搜索令牌 K_1 生成新的索引指针 UT_c,由于初始化时该指针为首个指针,并不存在以前的指针,所以将之前的指针置为空,并进行异或操作得到密文 CT_c.随后,用密钥令牌 K_2 与当前更新次数 c 生成一次性密钥 Sk_c,对字符串 bs 加密.客户端生成随机数 α_c,与密文 E_c 进行 Hash 认证.最后,将加密数据库 EDB 以及检查列表 LT 分别发送到云服务器和区块链,客户端本地更新 Map.

该算法中搜索状态 S 和搜索次数 cs 是为了实现前向安全而设置的参数,通过对搜索状态 S 判断云服务器是否执行过搜索算法,进而考虑搜索次数 cs 是否需要更新,而更新次数 c 负责对更新字符串的加密,所以只要发生更新过程,更新次数就需要加 1.

算法 1. 构建索引算法.

输入:需要构建索引的关键字 w、关键字对应的位图索引 bs、本地状态映射 Map 以及加密数据库 EDB;

输出:检查列表 LT 以及加密数据库 EDB.

客户端:

① 生成一个随机数 α_c;

② $\{c,cs\}\leftarrow 0$, $S\leftarrow$"N", $l_c\leftarrow F(K,cs\|c)$;

③ $K_1\leftarrow F(K,w\|cs)$, $K_2\leftarrow F(K,w\|\perp)$;

④ $UT_c\leftarrow H_1(K_1,c)$;

⑤ $CT_c\leftarrow H_2(K_1,UT_c)\oplus\perp$; /* 初始化时并不存在之前的指针,$UT_{c-1}$ 置为空 */

⑥ $Sk_c\leftarrow H_3(K_2,c)$;

⑦ $E_c\leftarrow Enc(Sk_c,bs,n)$;

⑧ $V_c\leftarrow H_4(E_c,\alpha_c)$;

⑨ 将$[UT_c:CT_c,E_c,\alpha_c]$发送到服务器,将$[l_c:V_c]$发送到区块链;

⑩ 更新 $Map[w]\leftarrow\{c,cs,S,UT_c\}$;

云服务器:

⑪ 服务器更新 $EDB[UT_c]\leftarrow\{CT_c,E_c,\alpha_c\}$;

区块链:

⑫ 区块链将更新检查列表 $LT[l_c]\leftarrow\{V_c\}$.

3) $Search(w,Map,EDB)$.客户端获取关键字 w 的本地状态,计算搜索令牌 K_1.然后需要对 S 进行判断,如图 4 所示:如果 S 等于 Y,说明该关键字在搜索过后没有发生更新,并且搜索结果验证通过之后记录在区块链上,只需将搜索令牌 K_1 发送到区块链中,获取之前结果 Sum_e;否则说明该关键字

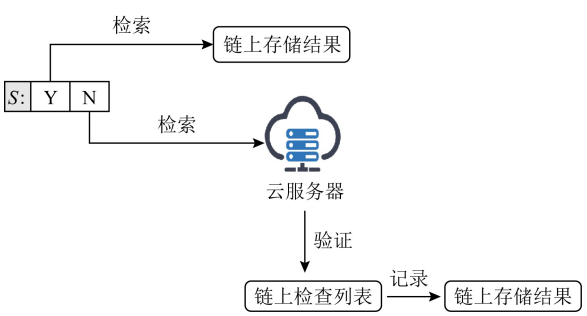

Fig. 4 Search status

图 4 搜索状态

有更新后的索引还没有被搜索,需要计算最新索引指针 UT_c、检查列表标志 l_c 以及之前的搜索令牌 K_1^{-1} 发送到云服务器进行搜索.

云服务器搜索所有与关键字 w 有关的密文,直到索引指针对应空值,将搜索令牌和检查列表标志 $\{K_1,K_1^{-1},l_c\}$ 与所有密文和随机数结果 $\{E_c,\cdots,E_{c-i}\}$,$\{\alpha_c,\cdots,\alpha_{c-i}\}$ 发送到区块链中.

智能合约将密文与随机数进行异或,与检查列表 LT 匹配,验证通过后搜索之前结果 $MEI[K_1^{-1}]$,如果结果不存在,说明本次为首次搜索,需要对 Sum_e 初始化,所有结果利用同态加法对称加密获得最终结果,将最终结果记录在区块链.

客户端得到密文结果后,本地计算密钥 Sum_{sk},解密获得最终字符串 bs 以及匹配文档标识符.

算法 2. 搜索算法.

输入:需要搜索的关键字 w、本地状态映射 Map 以及加密数据库 EDB;

输出:关键字 w 对应的最终结果 bs.

客户端:

① 获取$\{c,cs,S,UT\}\leftarrow Map[w]$, $K_1\leftarrow F(K,w\|cs)$;

② if $S==$"Y" then

③ 将 K_1 发送到区块链中;

④ 区块链进行搜索$\{Sum_e\}\leftarrow MEI[K_1]$;

⑤ return Sum_e;

⑥ else

⑦ $UT_c\leftarrow H_1(K_1,c)$, $l_c\leftarrow F(K,cs\|c)$, $K_1^{-1}\leftarrow F(K,w\|cs-1)$, $S\leftarrow$"Y";

⑧ 将$\{UT_c,K_1,K_1^{-1},l_c\}$广播到区块链中;

⑨ end if

服务器:

⑩ for $i=0$ until $EDB[UT_i]=\perp$ do

⑪ $\{CT_c,E_c,\alpha_c\}\leftarrow EDB[UT_c]$;

⑫ $UT_{c-1}\leftarrow CT_c\oplus H_2(K_1,UT_c)$;

⑬ end for

⑭ 将$\{K_1,K_1^{-1},l_c\}$,$\{E_c,\cdots,E_{c-i}\}$,$\{\alpha_c,\cdots,\alpha_{c-i}\}$发送到区块链;

区块链:

⑮ $V_w\leftarrow H_4(E_c,\alpha_c)\oplus\cdots\oplus H_4(E_{c-i},\alpha_{c-i})$;

⑯ if $V_w==LT[l_c]$ then

⑰ $\{Sum_e\}\leftarrow MEI[K_1^{-1}]$;

⑱ if $MEI[K_1^{-1}]==\perp$ then /* 如果不存在,则需要初始化 */

⑲ $Sum_e\leftarrow 0$;

⑳ end if
㉑ for $j=c$ to $c-i$ do
㉒ $\{Sum_e\}\leftarrow MEI[K_1^{-1}]$;
㉓ $Sum_e\leftarrow Add(Sum_e,E_j,n)$;
㉔ end for
㉕ end if
㉖ 将结果 $MEI[K_1]\leftarrow Sum_e$ 保存在区块链;
客户端:
㉗ $Sum_{sk}\leftarrow 0$;
㉘ for $i=0$ to c do
㉙ $Sk_i\leftarrow H_3(K_2,i)$;
㉚ $Sum_{sk}\leftarrow Sum_{sk}+Sk_i \bmod n$;
㉛ end for
㉜ $bs\leftarrow Dec(Sum_{sk},Sum_e,n)$.

4) $Update(w,bs,Map,EDB)$.客户端获取关键字 w 对应的更新状态,对字符串进行加密和 Hash 认证,生成新的检查列表标志 l_{c+1}.接下来对 S 进行判断:如果 S 等于 Y,表示该关键字搜索之后的首次更新,需要将搜索次数 cs 更新,生成新的密钥 K_1^*,最后将置 S 为 N;否则代表该关键字在更新后并没有被搜索,可以使用之前的密钥 K_1,新的索引指针与之前指针进行连接.向区块链获取之前的检查列表,与新的消息认证码 V_{c+1} 进行异或,得到新的检查列表.最后将加密数据库 EDB 以及新的检查列表 LT 发送到云服务器与区块链,本地更新 Map.

算法 3. 更新算法.

输入:需要更新的关键字 w、关键字对应的位图索引 bs、本地状态映射 Map 以及加密数据库 EDB;

输出:更新后的检查列表 LT 和加密数据库 EDB.

客户端:
① 获取$\{c,cs,S,UT\}\leftarrow Map[w]$;
② $l_{c+1}\leftarrow F(K,cs\|c+1)$;
③ $K_2\leftarrow F(K,w\|\perp)$,生成一个随机数 α_{c+1};
④ $Sk_{c+1}\leftarrow H_3(K_2,c+1)$;
⑤ $E_{c+1}\leftarrow Enc(Sk_{c+1},bs,n)$;
⑥ $V_{c+1}\leftarrow H_4(E_{c+1},\alpha_{c+1})$;
⑦ if $S==$"Y" then /* 对当前关键字对应的搜索状态进行判断 */
⑧ $cs=cs+1$;
⑨ $K_1^*\leftarrow F(K,w\|cs)$; /* 生成新的密钥 */
⑩ $UT_{c+1}\leftarrow H_1(K_1^*,c+1)$;
⑪ $CT_{c+1}\leftarrow H_2(K_1^*,UT_{c+1})\oplus\perp$; /* 此索引指针为更新后的首个索引指针 */
⑫ 更新 $LT[l_{c+1}]=V_{c+1}$;
⑬ $S\leftarrow$"N";
⑭ else
⑮ $l_c\leftarrow F(K,cs\|c)$;
⑯ $K_1\leftarrow F(K,w\|cs)$;
/* 仍然使用之前的密钥 */
⑰ $UT_{c+1}\leftarrow H_1(K_1,c+1)$;
⑱ $CT_{c+1}\leftarrow H_2(K_1,UT_{c+1})\oplus UT_c$; /* 此索引指针连接到之前的索引指针上 */
⑲ 向区块链请求之前的检查表 $L[l_c]$;
⑳ 更新 $LT[l_{c+1}]=L[l_c]\oplus V_{c+1}$;
㉑ end if
㉒ 更新 $Map[w]\leftarrow\{c,cs,S,UT\}$;
㉓ 将$[UT_{c+1}:CT_{c+1},E_{c+1},\alpha_{c+1}]$发送到服务器;
㉔ 将$[l_{c+1}:L[l_c]\oplus V_{c+1}]$发送到区块链;
客户端:
㉓ 服务器更新 $EDB[UT_{c+1}]$;
区块链:
㉔ 区块链将更新检查列表 $LT[l_{c+1}]$.

5 安全分析

5.1 安全模型

本方案采用文献[10]安全模型,通过现实模型 $Real$ 和理想模型 $Ideal$ 完成.$Real$ 模型与本方案行为一致;而 $Ideal$ 模型则反映了模拟器 S 的行为,模拟器 S 以泄露函数 $L=(L^{Setup},L^{Update},L^{Search})$ 作为输入.需要特别说明,由于 $Buildindex$ 算法与 $Update$ 算法只是在客户端存在差异,但是对于敌手 A 是相同的行为,所以二者的泄露函数没有区别.然后,我们给出 $Real$ 模型与 $Ideal$ 模型的定义.

$Real_A(\lambda)$:首先运行 $Setup$ 算法输出 EDB,敌手 A 执行搜索查询 sr(或者更新查询(op,in)),随后 A 输出结果 $b\in\{0,1\}$.

$Ideal_{A,S}(\lambda)$:模拟器 S 输入泄露函数 L^{Setup},敌手 A 进行搜索查询 sr(或者更新查询(op,in)),模拟器 S 输入泄露函数 L^{Search} 或者 L^{Update} 作为回答,敌手 A 输出结果 $b\in\{0,1\}$.

如果对于任何概率多项式敌手 A 存在高效模拟器 S 以及输入 L,使得:

$$|Pr[Real_A(\lambda)=1]-Pr[Ideal_{A,S}(\lambda)]|\leqslant negl(\lambda), \tag{3}$$

其中 $negl(\lambda)$ 是可以忽略的函数,则说明本方案满足 L-自适应安全.

5.2 前向安全

对于任何敌手而言,前向安全保证了更新不会泄露新添加的文档与之前搜索查询的匹配信息.如果更新泄露函数 L^{Update} 可以写成

$$L^{Update}(op,in)=L'(op,\{(f_i,\mu_i)\}),\quad(4)$$

$\{(f_i,\mu_i)\}$是 1 组发生过更改的关键字-文件标识符对,μ_i 是文档 f_i 包含的关键词数量,op 为具体更新操作.在本文中,泄露函数可以写成

$$L^{Update}(op,w,bs)=L'(op,bs),\quad(5)$$

其中 w 为更新过的关键字,bs 为新添加的字符串.

5.3 后向安全

后向安全保证了当之前添加的文档被删除后,在添加之前的搜索和删除后的搜索不会泄露关于这篇文档的信息.文献[7]定义了 3 种不同级别的后向安全,从低到高分为 Type-Ⅲ,Type-Ⅱ,Type-Ⅰ.

本方案采用位图索引,达到了更强的后向安全 Type-Ⅰ$^-$,下面给出 Type-Ⅰ$^-$与 Type-Ⅰ的区别.

Type-Ⅰ$^-$:对于关键字 w 的 2 次搜索,只会泄露该关键字最终匹配文档结果,以及该关键字的更新次数和更新时间.

Type-Ⅰ:对于关键字 w 的 2 次搜索,除了泄露该关键字最终匹配文档结果和更新次数,还会泄露结果中文档的添加时间.

在正式定义 Type-Ⅰ$^-$之前,需要构建一个新的泄露函数 $Time$.对于关键字 w 的一次搜索查询,$Time(w)$会列出其对应的每次更新时间戳 t.对于一系列的更新查询 Q':

$$Time(w)=(t:(t,op,(w,bs))\in Q').\quad(6)$$

如果搜索泄露函数以及更新泄露函数可以写成下面 2 个公式:

$$\begin{aligned}&L^{Update}(op,w,bs)=L'(op),\\&L^{Search}(w)=L''(sp(w),rp(w),Time(w)),\end{aligned}\quad(7)$$

即满足 Type-Ⅰ$^-$后向安全.其中 $sp(w)$为搜索模式且 $sp(w)=\{t:(t,op)\in Q\}$,Q 为一系列的搜索查询;$rp(w)=bs^*$ 为结果模式,bs^* 代表当前 w 匹配的所有文档标识符;L'与 L''表示 2 个无状态的函数.

5.4 具体安全分析

定理 1. 如果伪随机函数 F 是安全的,所有 Hash 函数具有抗碰撞性质,$\Pi=(Setup,Enc,Dec,Add)$是具有完美安全性的同态加法对称加密算法,则定义本方案泄露函数 $L=(L^{Update},L^{Search})$,其中 $L^{Update}(op,w,bs)=\perp$,$L^{Search}(w)=L''(sp(w),rp(w),Time(w))$,那么本方案为满足自适应安全、前向安全以及 Type-Ⅰ$^-$后向安全的定义.本文与文献[10]中方案类似,由此给出以下安全分析:从模型 $Real$ 到 $Ideal$ 设置一系列游戏,每个游戏都与上一个游戏有所不同,但是对于敌手 A 无法区分 2 个游戏,最后使用定理 1 中泄露函数模拟 $Ideal$,从而得到敌手 A 无法区分模型 $Real$ 和 $Ideal$.在游戏中敌手 A 会尝试破坏本方案的安全,挑战者 C 负责生成搜索令牌以及密文,模拟器 S 则从始至终模拟 A 与 C 的交互.

游戏 G_0 与现实模型游戏 $Real_A(\lambda)$相同,所以有

$$Pr[Real_A(\lambda)=1]=Pr[G_0=1].\quad(8)$$

在游戏 G_1 中,当查询使用 F 生成关键字 w 的密钥时,如果该密钥之前没有被搜索过,挑战者 C 则选择一个新的随机密钥返回给敌手 A,并保存在表 $KeyL$ 中,否则返回 $KeyL$ 表中关键字 w 对应的密钥.如果敌手 A 可以区分 G_0 与 G_1 的优势,则

$$Pr[G_0=1]-Pr[G_1=1]\leqslant Adv^{prf}_{F,A}(\lambda).\quad(9)$$

在游戏 G_2 中,对于 $Update$ 算法我们采用随机字符串作为索引指针 UT_c,并将其保存在表 UTL 中,即 $UTL[w,c]=UT_c$.随后在搜索算法中,我们将这些随机字符串处理为随机预言机 H_1 的输出,其中 $H_1(K_1,c)=UTL[w,c]$.当敌手 A 把(K_1,c)输入 H_1 进行查询时,挑战者 C 会输出 $UTL[w,c]$并保存在 HL 表中以应对接下来的查询.如果$(K_1,c+1)$已经存在 HL 表中,那么就不会输出 $UTL[w,c+1]$并且游戏中止.由于 UT_c 为挑战者 C 随机选择的字符串,所以敌手 A 可以猜测到正确的索引指针的概率为 $1/2^\lambda$,假设 A 可以进行多项式 q 次查询,那么其概率为 $q/2^\lambda$,所以得出:

$$Pr[G_2=1]-Pr[G_1=1]\leqslant q/2^\lambda.\quad(10)$$

在游戏 G_3 中,我们将 Hash 函数 H_2 也转化为随机预言机,由于 G_3 与 G_2 类似,可以得出:

$$Pr[G_3=1]-Pr[G_2=1]\leqslant q/2^\lambda.\quad(11)$$

在游戏 G_4 中,同样将 Hash 函数 H_3 转化为随机预言机.由于敌手 A 不知道密钥 K_2,所以其猜测正确一次性密钥的概率为 $1/2^\lambda$.假设 A 可以进行多项式 q 次查询,那么其概率为 $q/2^\lambda$,所以得出:

$$Pr[G_4=1]-Pr[G_3=1]\leqslant q/2^\lambda.\quad(12)$$

在游戏 G_5 中,我们使用一个全为 0 的字符串 $bs^\sharp$ 去代替字符串 bs,并且二者的长度相同均为 m.如果敌手 A 可以区分 G_5 与 G_4,则其获得的优势概率为

$$Pr[G_4=1]-Pr[G_5=1]\leqslant Adv^{PS}_{\Pi,A}(\lambda).\quad(13)$$

接下来,我们将使用模拟器 S 从敌手 A 的视角进行模拟,使用泄露函数 $sp(w)$ 代替关键字 w.使用第 1 个时间戳 $w^* \leftarrow minsp(w)$,并且移除对于敌手 A 没有影响的部分.对于 $Update$ 算法很明显看出,在 G_5 游戏中每次更新都会选取新的随机字符串.在 $Search$ 算法中,S 会从当前的索引指针 UT 开始,并选取随机字符串作为之前的索引指针,然后通过 H_2 得出更新密文 CT,将 bs^* 转化为索引指针 UT,并将剩余所有 0 字符 H_3 转为剩下的一次性密钥.随后,我们将 (w,i) 映射到全局计数器 t 上,然后将索引指针表 UTL、更新密文表 CT 以及 $Update$ 算法中随机选取的一次性密钥 sk,作为 $Search$ 算法 (w,i) 对应的值.可以得出:

$$Pr[G_5=1]=Pr[Ideal_{A,S}(\lambda)=1]. \quad (14)$$

最后,我们可以总结为如下等式

$$Pr[Real_A(\lambda)=1]-Pr[Ideal_{A,S}(\lambda)=1]\leqslant Adv_{F,B}^{prf}(\lambda)+Adv_{H,B1}^{PS}(\lambda)+3q/2^{\lambda}. \quad (15)$$

由于本方案采用伪随机函数是安全的,Hash 函数具有抗冲突性质且同态加法对称加密具有完美安全性,所以可以得出敌手 A 区分出 $Real$ 模型和 $Ideal$ 模型的优势为可以忽略的函数 $negl(\lambda)$.综上所述,本方案满足自适应安全以及前向安全和后向安全定义.

6 方案分析

6.1 功能分析

将本文方案与文献[10,17,26-27]进行了功能上的对比,如表 2 所示.文献[10]达到了前向安全与后向安全,但是对于云服务器返回的结果并没有验证其正确性,并且无法保证云服务器是否如实地删除了之前状态的索引,是否将最终结果诚实记录在最新状态下.文献[17]支持对结果正确性的验证,但是其安全性仅支持前向安全,每次更新状态只对应一个关键字-文档标识符对,造成了很大的存储以及计算开销.

Table 2 Comparison of Function

表 2 功能对比

方案	区块链	前向安全	后向安全	结果验证	恶意用户检测
文献[10]方案	×	√	√	×	×
文献[17]方案	×	√	×	√	×
文献[26]方案	√	√	×	√	√
文献[27]方案	√	√	×	√	√
本文方案	√	√	√	√	√

注:"×"表示不具有特定功能或未使用某种技术;"√"表示具有特定功能或未使用某种技术.

文献[26-27]均基于区块链平台,将搜索过程与验证过程进行分离,达到了对恶意用户和云服务器的双向验证,但是二者均不支持后向安全.文献[26]中的方案,只给出了增加文档的具体算法,对于删除操作并没有进行详细说明.文献[27]方案中的删除算法需要在搜索算法执行之后才能进行,所以可能导致最后结果中包含已经删除的文档,使其变成了错误结果.

从功能上说,本方案实现了前向安全与后向安全,对于用户与云服务器进行了双向验证,保证了整个方案的诚实执行.将最终结果保存在区块链中,用户在获取结果后本地进行解密,保证了删除文档不会被搜索.本方案在提高安全性的同时减少了用户本地的计算存储开销,提高了云服务器的搜索以及更新效率.

6.2 理论分析

在理论分析方面,我们给出了本文方案与其他方案在计算开销复杂度方面的对比,为了简单明了地展示对比结果,对比参数均设定在一次更新的情况下进行,其中 N_D 表示关键字 w 需要更新的文档数量.如表 3 所示,由于本方案采用了位图索引,所以不需要对所有的包含关键字 w 的更新文档进行操作,复杂度仅为 $O(1)$.

Table 3 Comparison of Computational Complexity

表 3 计算复杂度对比

方案	*Buildindex*	*Search*	*Update*	*Verify*
文献[26]方案	$O(N_D)$	$O(N_D)$	$O(N_D)$	$O(N_D)$
文献[27]方案	$O(N_D)$	$O(N_D)$	$O(N_D)$	$O(N_D)$
本文方案	$O(1)$	$O(1)$	$O(1)$	$O(1)$

注:*Verify* 表示验证操作.

6.3 实现分析对比

本方案实验环境参数为 64 b Windows 操作系统,Intel® Core™ i5-4570 CPU 3.20 GHz、内存

16.00 GB.使用 Enron Email 数据集作为原始数据集,提取出子集作为测试数据集.实验中智能合约使用 Solidity 语言,与智能合约交互语言为 Python 语言.加密算法基于 SHA-256 构建,MAC 生成算法与伪随机函数均采用 HMAC-SHA256.最后将智能合约部署到 Ganache 网络.由于文献[10,17]并非基于区块链构建的方案,且不能对用户和云服务器进行双向验证,所以我们将本方案与文献[26-27]分别在索引生成、搜索、更新以及验证 4 个方面进行了性能对比.

如图 5 所示,随着关键字数量的增加,构建索引时间呈现线性增长趋势.但是本方案的索引构建时间远小于其他方案,主要因为文献[26-27]需要对每个关键字-文档标识符对进行加密操作,随后将该关键字对应的所有文档标识符密文进行连接处理,而本方案只需要对关键字对应的字符串进行加密即可,减少了计算开销.

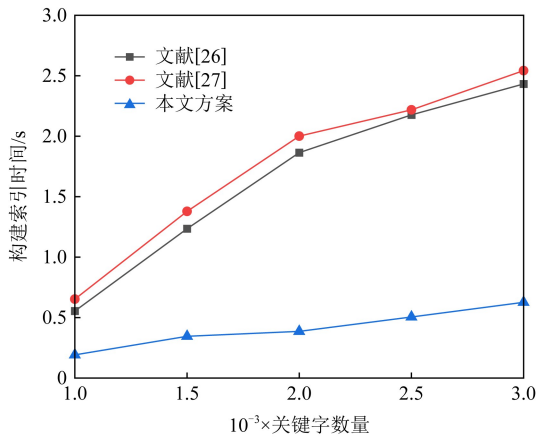

Fig. 5 Build index time

图 5 构建索引时间

对于搜索方面的性能对比,将关键字涉及的更新次数设置为参数,即在不同方案中关键字 w 取相同的更新次数,每次更新的关键字-文档标识符对数量也相同.如图 6 所示,本方案的搜索时间成本低于其他方案,主要原因在于本方案的搜索时间与更新次数呈线性相关,每次搜索过程中只需对位图索引进行匹配即可,而其他方案则需要对关键字 w 涉及到的所有加密数据进行匹配.并且,由于本方案会将验证通过后的位图索引加密结果记录在区块链中,客户端可以直接对其进行搜索,减少了重复验证的过程,进而提高了搜索效率.

对于更新方面,我们以关键字 w 的 1 次更新涉及的文档数量为参数,即在对比试验中不同方案都

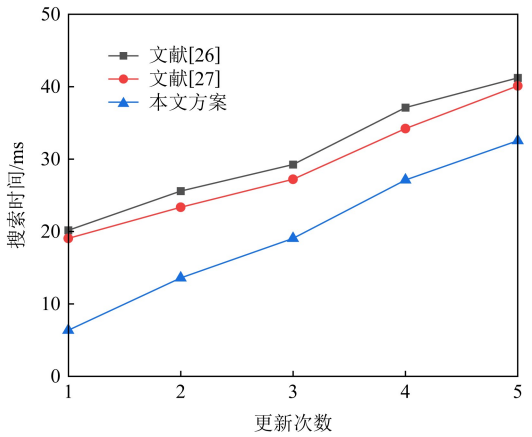

Fig. 6 Search time

图 6 搜索时间

只更新 1 次,在更新中涉及到的关键字-文档标识符对数量相同.如图 7 所示,文献[26-27]中方案的更新时间随着文档数量的增加而增加,造成这种现象的主要原因在于,这 2 个方案的索引采用链式结构,在更新过程中,需要对每个文档标识符重复加密 2 次,并且为了最后可以对其进行验证,还需要对每个文档标识符进行 Hash 运算.而本方案由于其特殊的索引构造,在每次更新过程中只需要对索引进行 1 次加密和 Hash 运算即可,与文档标识符数量无关,导致更新时间不会发生剧烈变化,且更新时间远低于其他方案.

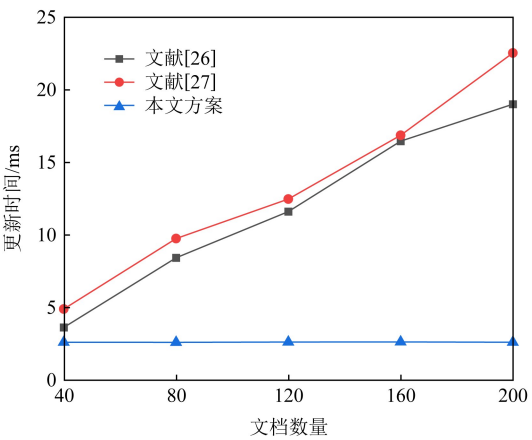

Fig. 7 Update time

图 7 更新时间

如图 8 所示,验证时间随着更新次数的增加而增加.本方案在进行验证时只需要对返回的加密字符串进行 Hash 运算即可,而其他方案需要对所有文档标识符进行计算,导致了大量的计算开销.最后区块链需要将加密字符串进行加法操作,从而得到

最终的加密结果,虽然在一定程度上影响了本方案的验证时间,但是对比其他方案仍具有一定优势.

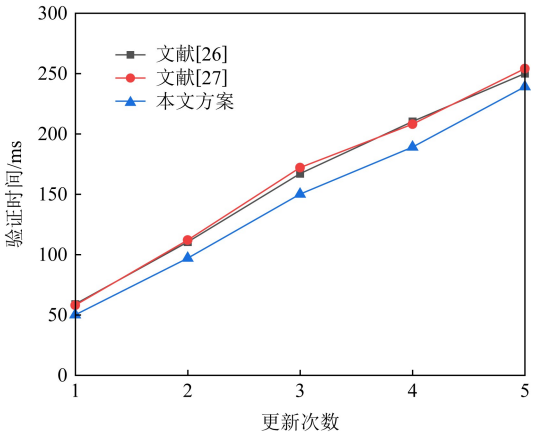

Fig. 8 Verification time

图 8 验证时间

7 总 结

本方案将区块链与可搜索加密技术进行结合,提出了一种双向验证的动态密文检索方案.本方案主要应用于需要确保数据绝对安全的私有云环境,例如机密机构的数据库.由于一个位图索引可以表示多个文件标识符,减少了关键词对应的加密数据,提高了搜索效率以及更新效率,并使用同态加法对称加密技术,实现了云服务器对数据的安全更新.将验证过程交由区块链去处理,实现了对用户与云服务器的双向验证,保证了结果不可否认.同时针对方案中索引生成、搜索、更新以及验证 4 个方面进行了实验测试,测试结果显示本方案具有较高的效率.

未来工作将会针对可搜索加密如何在区块链环境下实现 1 对多用户模型的双向验证,并且安全地实现数据拥有者对数据用户的访问控制.

作者贡献声明:杜瑞忠负责论文方案的构思以及论文的撰写与分析,并对论文整体进行把关;王一负责论文的撰写、修改;李明月对方案提出了改进意见.

参 考 文 献

[1] Song Xiaodong, Wagner D, Perrig A. Practical techniques for searches on encrypted data [C] //Proc of the 21st IEEE Symp on Security and Privacy. Piscataway, NJ: IEEE, 2000: 44-55

[2] Goh E J. Secure indexes, 2003/216 [R/OL]. IACR ePrint Cryptography Archive, 2003 [2021-01-30]. http://eprint.iacr.org/2003/216

[3] Curtmola R, Garay J, Kamara S, et al. Searchable symmetric encryption: Improved definitions and efficient constructions [J]. Journal of Computer Security, 2011, 19(5): 895-934

[4] Kamara S, Papamanthou C, Roeder T. Dynamic searchable symmetric encryption [C] //Proc of the 19th ACM CCS. New York: ACM, 2012: 965-976

[5] Stefanov E, Papamanthou C, Shi E. Practical dynamic searchable encryption with small leakage, 2013/832[R/OL]. IACR ePrint Cryptography Archive, 2013 [2021-01-30]. http://eprint.iacr.org/2013/ 832

[6] Bost R. Σ οφος: Forward secure searchable encryption [C] //Proc of the 23rd ACM CCS. New York: ACM, 2016: 1143-1154

[7] Bost R, Minaud B, Ohrimenko O. Forward and backward private searchable encryption from constrained cryptographic primitives [C] //Proc of the 24th ACM CCS. New York: ACM, 2017: 1465-1482

[8] Sun Shifeng, Yuan Xingliang, Liu J K, et al. Practical backward-secure searchable encryption from symmetric puncturable encryption [C] //Proc of the 25th ACM CCS. New York: ACM, 2018: 763-780

[9] Chamani J G, Papadopoulos D, Papamanthou C, et al. New constructions for forward and backward private symmetric searchable encryption [C] //Proc of the 25th ACM CCS. New York: ACM, 2018: 1038-1055

[10] Zuo Cong, Sun Shifeng, Liu J K, et al. Dynamic searchable symmetric encryption with forward and stronger backward privacy [C] //Proc of the 24th ESORICS. Berlin: Springer, 2019: 283-303

[11] Vo V, Lai Shangqi, Yuan Xingliang, et al. Accelerating forward and backward private searchable encryption using trusted execution [C] //Proc of the 18th ACNS. Berlin: Springer, 2020: 83-103

[12] He Kun, Chen Jing, Zhou Qinxi, et al. Secure dynamic searchable symmetric encryption with constant client storage cost [J]. IEEE Transactions on Information Forensics and Security, 2020, 16: 1538-1549

[13] Demertzis I, Chamani J G, Papadopoulos D, et al. Dynamic searchable encryption with small client storage, 2019/1227 [R/OL]. IACR ePrint Cryptography Archive, 2019 [2021-01-30]. http://eprint.iacr.org/2019/1227

[14] Zuo Cong, Sun Shifeng, Liu J K, et al. Forward and backward private DSSE for range queries, 2019/1240 [R/OL]. IACR ePrint Cryptography Archive, 2019 [2021-01-30]. http://eprint.iacr.org/2019/1240

[15] Chai Qi, Gong Guang. Verifiable symmetric searchable encryption for semi-honest-but-curious cloud servers [C] // Proc of the 25th ICC. Piscataway, NJ: IEEE, 2012: 917-922

[16] Soleimanian A, Khazaei S. Publicly verifiable searchable symmetric encryption based on efficient cryptographic components [J]. Designs, Codes and Cryptography, 2019, 87(1): 123-147

[17] Zhang Zhongjun, Wang Jianfeng, Wang Yunling, et al. Towards efficient verifiable forward secure searchable symmetric encryption [C] //Proc of the 24th ESORICS. Berlin: Springer, 2019: 304-321

[18] Liang Yanrong, Li Yanping, Cao Qiqang, et al. VPAMS: Verifiable and practical attribute-based multi-keyword search over encrypted cloud data [J]. Journal of Systems Architecture, 2020, 108: 101741

[19] Tong Qiuyun, Miao Yinbin, Liu Ximeng, et al. VPSL: Verifiable privacy-preserving data search for cloud-assisted Internet of things [J/OL]. IEEE Transactions on Cloud Computing, 2020 [2021-01-30]. https://ieeexplore.ieee.org/document/9224165

[20] Miao Yinbin, Tong Qiuyun, Deng R, et al. Verifiable searchable encryption framework against insider keyword-guessing attack in cloud storage [J/OL]. IEEE Transactions on Cloud Computing, 2020 [2021-01-30]. https://ieeexplore.ieee.org/document/9075374

[21] Yang Wenyuan, Zhu Yuesheng. A verifiable semantic searching scheme by optimal matching over encrypted data in public cloud [J]. IEEE Transactions on Information Forensics and Security, 2020, 16: 100-115

[22] Shamshad S, Mahmood K, Kumari S, et al. A secure blockchain-based e-health records storage and sharing scheme [J]. Journal of Information Security and Applications, 2020, 55: 102590

[23] Zhang Qikun, Li Yongjiao, Wang Ruifang, et al. Data security sharing model based on privacy protection for blockchain-enabled industrial Internet of things [J]. International Journal of Intelligent Systems, 2021, 36(1): 94-111

[24] Hoang V H, Lehtihet E, Ghamri-Doudane Y. Privacy-preserving blockchain-based data sharing platform for decentralized storage systems [C] //Proc of the 2020 IFIP Networking Conf. Piscataway, NJ: IEEE, 2020: 280-288

[25] Li Mingyue, Jia Chunfu, Shao Wei. Blockchain based multi-keyword similarity search scheme over encrypted data [C] // Proc of the 16th SecureComm 2020. Berlin: Springer, 2020: 350-371

[26] Guo Yu, Zhang Chen, Jia Xiaohua. Verifiable and forward-secure encrypted search using blockchain techniques [C/OL] //Proc of the 33rd ICC. 2020 [2021-01-30]. https://ieeexplore.ieee.org/document/9148612

[27] Li Han, Zhou Hongliang, Huang Hejiao, et al. Verifiable encrypted search with forward secure updates for blockchain-based system [C] //Proc of the 14th WASA. Berlin: Springer, 2020: 206-217

[28] Castelluccia C, Mykletun E, Tsudik G. Efficient aggregation of encrypted data in wireless sensor networks [C] //Proc of the 2nd Annual Int Conf on Mobile and Ubiquitous Systems: Networking and Services. Piscataway, NJ: IEEE, 2005: 109-117

[29] Wood G. Ethereum: A secure decentralised generalised transaction ledger [EB/OL]. [2021-01-30]. http://ethereum.github.io/yellowpaper/ paper. pdf

[30] Atzei N, Bartoletti M, Cimoli T. A survey of attacks on Ethereum smart contracts (SoK) [C] //Proc of the 6th POST. Berlin: Springer, 2017: 164-186

Du Ruizhong, born in 1975. PhD, professor, PhD supervisor. His main research interests include reliable computing and network security.

杜瑞忠,1975 年生.博士,教授,博士生导师.主要研究方向为可信计算与网络安全.

Wang Yi, born in 1995. Master. His main research interests include searchable encryption, trusted computing and network technology.

王　一,1995 年生.硕士.主要研究方向为可搜索加密、可信计算和网络技术.

Li Mingyue, born in 1993. PhD candidate. Her main research interests include network security and searchable encryption.

李明月,1993 年生.博士研究生.主要研究方向为网络安全和可搜索加密.