软件学报 ISSN 1000-9825, CODEN RUXUEW
Journal of Software,2023,34(11):5392−5407 [doi: 10.13328/j.cnki.jos.006685]

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

基于区块链的动态可验证对称可搜索加密方案*

徐万山[1,2], 张建标[1,2], 袁艺林[1,2]

[1](北京工业大学 信息学部 计算机学院, 北京 100124)

[2](可信计算北京市重点实验室 (北京工业大学), 北京 100124)

通信作者: 张建标, E-mail: zjb@bjut.edu.cn

摘　要: 对称可搜索加密 (symmetric searchable encryption, SSE) 能实现密文数据的检索而不泄露用户隐私, 在云存储领域得到了广泛的研究与应用. 然而, 在 SSE 方案中, 半诚实或者不诚实的服务器可能篡改文件中的数据, 返回给用户不可信的文件, 因此对这些文件进行验证是十分必要的. 现有的可验证 SSE 方案大多是用户本地进行验证, 恶意用户可能会伪造验证结果, 无法保证验证的公平性. 基于以上考虑, 提出一种基于区块链的动态可验证对称可搜索加密方案 (verifiable dynamic symmetric searchable encryption, VDSSE); VDSSE 采用对称加密实现动态更新过程中的前向安全; 在此基础上, 利用区块链实现搜索结果的验证, 验证过程中, 提出一种新的验证标签——*Vtag*, 利用 *Vtag* 的累积性实现验证信息的压缩存储, 降低验证信息在区块链上的存储开销, 并能够有效支持 SSE 方案的动态验证. 由于区块链具有不可篡改的性质, 验证的公平性得以保证. 最后, 对 VDSSE 进行实验评估和安全性分析, 验证方案的可行性和安全性.

关键词: 对称可搜索加密; 可验证; 区块链; 动态更新

中图法分类号: TP309

中文引用格式: 徐万山, 张建标, 袁艺林. 基于区块链的动态可验证对称可搜索加密方案. 软件学报, 2023, 34(11): 5392–5407. http://www.jos.org.cn/1000-9825/6685.htm

英文引用格式: Xu WS, Zhang JB, Yuan YL. Verifiable Dynamic Searchable Symmetric Encryption Based on Blockchain. Ruan Jian Xue Bao/Journal of Software, 2023, 34(11): 5392–5407 (in Chinese). http://www.jos.org.cn/1000-9825/6685.htm

Verifiable Dynamic Searchable Symmetric Encryption Based on Blockchain

XU Wan-Shan[1,2], ZHANG Jian-Biao[1,2], YUAN Yi-Lin[1,2]

[1](School of Computer Science and Technology, Faculty of Information Technology, Beijing University of Technology, Beijing 100124, China)

[2](Beijing Key Laboratory of Trusted Computing (Beijing University of Technology), Beijing 100124, China)

Abstract: Symmetric searchable encryption (SSE) can retrieve encrypted data without disclosing user privacy and has been widely studied and applied in cloud storage. However, in SSE schemes, semi-honest or dishonest servers may tamper with the data in files and return the untrusted files to users, so it is necessary to verify these files. Most existing verifiable SSE schemes are verified by the users locally, and malicious users may forge verification results, which cannot ensure verification fairness. To this end, this study proposes a verifiable dynamic symmetric searchable encryption scheme based on blockchain, VDSSE). VDSSE employs symmetric encryption to achieve forward security in the dynamic updating, and on this basis, the blockchain is utilized to verify the search results. During the verification, a new verification tag, *Vtag*, is proposed. The accumulation of *Vtag* is leveraged to compress the verification information, reduce the storage cost of verification information on the blockchain, and effectively support the dynamic verification of SSE schemes. Finally, experimental evaluation and security analysis are conducted on VDSSE to verify the feasibility and security of the scheme.

Key words: symmetric searchable encryption (SSE); verifiable; blockchain; dynamic update

* 基金项目: 北京市自然科学基金 (M21039)
收稿时间: 2021-05-04; 修改时间: 2021-12-02, 2022-03-11; 采用时间: 2022-03-29; jos 在线出版时间: 2023-04-27
CNKI 网络首发时间: 2023-04-28

随着云计算的不断发展, 越来越多的用户将数据外包至云服务器上; 借助云计算资源海量、可移动接入等优势, 用户既节省了本地存储和管理开销, 又提高了数据访问的便捷性. 为了保护数据的安全和隐私, 用户一般先将数据加密, 然后将密文上传至云服务器. 在此过程中, 尽管加密在一定程度上保护了数据的安全, 但却破坏了数据之间的关系, 使得对密文的检索成为一个难题.

为解决密文检索问题, Song 等人[1]提出了可搜索加密技术, 实现了在不泄露文件信息的前提下对密文进行检索, 有效地保护了用户隐私. 相比于安全多方计算、同态加密等其他密文检索方案, 可搜索加密技术计算代价低, 实现了搜索效率和安全性的平衡, 优势更显著, 因而得到了广泛的研究与应用.

可搜索加密分为对称可搜索加密 (SSE)[2]和公钥可搜索加密 (PEKS)[3], 对称可搜索加密在文件加密、解密时采用对称加密算法, 相比于公钥可搜索加密, 效率更高. 现有的大多数 SSE 方案中[4−7], 服务器是诚实且好奇 (honest-but-curious) 的, 即密文检索过程中, 服务器能够诚实的进行文件检索, 但是可能会猜测文件的内容信息, 甚至对文件进行解密. 然而, 在公有云存储场景下, 服务器可能是半诚实的或者不诚实的 (semi-honest-but-curious), 云存储服务器可能为了节省计算资源而向用户返回不正确的搜索结果. 同时, 密文检索过程中, 由于网络的不稳定性以及可能的软硬件故障, 也可能导致用户收到不正确的搜索结果. 因此, 对密文检索的结果进行验证十分必要.

在实际应用中, 云服务器上存储的数据始终保持静态是不现实的, 还需要对数据进行动态更新. 对此, 一些学者提出了动态对称可搜索加密方案 (DSSE), 实现了动态的密文检索. 动态密文检索过程中, 非前向安全模型下存在因文件更新而产生隐私泄露的问题, 因此动态密文检索要保证前向安全. 另外, 传统的对文件进行单次验证的静态验证方案无法满足动态密文检索的需求. 因此, 如何设计实现高效的、动态的密文检索验证方案成为新的挑战. 针对以上问题, 本文提出了一种基于区块链的动态可验证对称可搜索加密方案 (verifiable dynamic symmetric searchable encryption, VDSSE), 本文的主要贡献如下.

(1) 基于区块链技术, 构建了一种可验证 SSE 方案 VDSSE. VDSSE 在保证前向安全的基础上, 利用服务器实现密文检索, 并由区块链实现搜索结果验证. 数据拥有者将安全索引上传到服务器, 同时将验证信息上传到区块链; 服务器执行密文检索后, 将搜索文件发送到区块链进行验证, 最后用户取回区块链验证结果和搜索文件. VDSSE 利用区块链不可篡改的性质保证了验证结果的正确性和公平性, 降低了用户本地计算开销; 同时由于文件检索由服务器执行, 保证了检索的高效性.

(2) 提出了一种新的验证标签——*Vtag*, 利用 *Vtag* 累积的性质, 可以实现文件更新过程中验证信息的动态更新, 实现了 SSE 方案的动态验证, 提高了验证效率; 同时将同一关键词对应的任意多个文件的验证信息压缩存储到固定大小, 降低了区块上验证信息的存储开销.

(3) 利用对称加密实现了前向安全的搜索索引, 保证了方案更新过程中的前向安全性. 我们对 VDSSE 进行了安全性分析和性能对比, 并基于真实数据集在以太坊网络中实现了本文方案; 实验结果表明, 本文提出的 VDSSE 方案在检索效率、存储空间等方面具有显著的优势.

1 相关工作

可搜索加密技术提出后, 众多学者围绕着更丰富的查询范式、动态更新以及更高的安全性等 3 个方面对可搜索加密技术进行了广泛的研究. Song 等人[1]提出的可搜索加密方案中, 数据加密和查询都是基于对称加密的, 因而是一种对称可搜索加密技术, 然而该方案是基于全文检索的, 搜索时间与文件大小线性相关. 为了提高效率, Goh[8] 构建了一种基于布隆过滤器的搜索索引, 在此基础上, Curtmola 等人[9]提出一种基于倒排索引的 SSE 方案, 实现了亚线性搜索, 提高了搜索效率.

初始的可搜索加密技术是一种静态的可搜索加密技术, Kamara 等人[2]首次提出了支持高效更新的动态可搜索加密方案. 动态可搜索加密方案允许用户远程对文件进行更新, 但也带来了前向安全等问题. Cash 等人[10]研究表明, 被动攻击者利用 SSE 方案中即使很小的泄露也能猜测出文件的敏感信息, 甚至恢复全文. Zhang 等人[11]对 SSE 动态更新过程中的文件注入攻击进行了研究, 结果表明在适应性攻击和非适应性攻击两种场景下, 敌手可以

通过 SSE 更新时注入特定的文件从而推断出文件中包含的关键词等信息, 而前向安全的 SSE 方案能够有效抵御该攻击. 此后, 一系列前向安全的 SSE 方案被提出, Stefanov 等人[12]和 Garg 等人[13]提出了基于 ORAM 的前向安全 SSE 方案, 但是该方案通信开销较大, 效率较低. Bost[14]利用陷门置换构建了高效的前向安全 SSE 方案, 取得了很好的效果, 但是由于陷门置换采用公钥加密实现, 当数据频繁更新时会降低效率. Wei 等人[15]改进了 Bost 的方案, 利用对称加密实现陷门置换, 并基于 keyed-blockchain 构建了前向安全 SSE 方案, 进一步提高了效率. 此外, 最近也有一些研究[16,17]采用可信硬件技术 (如 Intel SGX 等) 保障 SSE 的前向安全性, 但是这些方案需要专用的硬件组件, 不具有通用性.

上述方案主要是基于诚实且好奇的服务器, 然而, 实际中有些服务器是半诚实的, 可能只返回部分搜索文件或者对文件进行篡改, 因此需要对云服务器返回给用户的文件进行验证. Chai 等人[18]第 1 次提出了可验证 SSE 的概念并基于单词树构建了可验证 SSE 方案. Kurosawa 等人[19]提出了一种 UC-Secure (通用可组合) 的可验证 SSE 方案, 方案满足非适应性安全. Wang 等人[20]提出了一种支持连接关键词查询的可验证 SSE 方案. 以上几种方案仅支持静态的 SSE 验证, 为了实现 SSE 方案的动态验证, Sun 等人[21]基于双线性映射累加器以及累加器树构建了支持动态连接词查询的可验证 SSE 方案, Zhu 等人[22]提出了一种基于倒排索引的动态模糊关键词可验证 SSE 方案, Liu 等人[23]提出了一种支持搜索结果排序的可验证 SSE 方案. 这些动态可验证方案采用 RSA 累加器实现结果验证, 采用公钥加密, 效率不高. Zhang 等人[24]基于多集哈希函数构建了一个高效的动态可验证 SSE 方案, Ge 等人[25]基于对称加密实现了可验证的 SSE 方案, 提高了方案执行效率. 以上方案均是由用户本地进行验证, 而在用户不诚实或者恶意抵赖的情况下, 可能会伪造验证结果, 无法实现搜索结果的公平验证.

区块链是一种分布式的加密数据库, 具有去中心化、不可篡改的特点, 利用区块链作为可信第三方, 实现加密数据搜索结果的验证, 能够有效解决服务器和用户之间密文检索的公平性验证问题. Hu 等人[26]基于区块链技术, 利用智能合约替代中心服务器, 构建了一个分布式的、可验证的、公平的密文检索方案. Cai 等人[27]实现了分布式存储中的动态关键词检索, 利用区块链技术对检索结果进行验证, 保证客户端和服务器之间的公平. Li 等人[28]将区块链与 SSE 相结合, 实现了服务器和用户之间的公平验证, 但是该方案基于验证过程基于消息验证码, 是静态验证. 李涵等人[29]基于区块链技术, 利用双索引结构, 构建了一种分布式场景下支持前向安全更新和验证的加密搜索算法. 闫玺玺等人[30]提出了一种基于区块链且支持验证的属性基搜索加密方案, 解决了一对多模型下密钥共享和搜索结果正确性验证问题. 这些方案密文检索和验证两个过程都建立在区块链上, 需要通过智能合约的交易完成, 效率不高.

针对以上问题, 本文利用对称加密实现了前向安全的对称可搜索加密, 保证了动态对称可搜索加密的安全更新, 在此基础上, 利用区块链实现了搜索结果的公平性验证, 在验证过程中将密文检索建立在服务器上, 仅将验证过程建立在区块链上, 减少了交易次数. 同时, 采用创新的验证标签, 能实现 SSE 方案的动态高效验证, 并节省存储空间.

2 方案概述

2.1 系统模型

表 1 为本文主要符号说明. 本文提出的 VDSSE 方案模型如图 1 所示, 系统共分为 4 个实体: 数据拥有者 (data owner, DO)、云服务器 (cloud server, CS)、数据用户 (data user, DU) 和区块链 (blockchain, BC). 对于文件集合 D, 数据拥有者 DO 提取文件中的关键词 $W_{d=1}^{n}$, 并产生文件列表 DB, 对 DB 加密生成前向安全索引 $\mathcal{I}$, 对集合 D 中的文件加密, 生成密文集合 C, 利用 C 中的加密文件生成验证列表 $\mathcal{L}$, 将 ($\mathcal{I}$, C) 上传到服务器 CS, 将 $\mathcal{L}$ 上传到区块链 BC. 有新用户 DU 需要共享数据时, 首先向数据拥有者注册, 获取系统公共参数 δ, 然后利用 δ 和查询关键词 w_i ($w_i \in W_{d=1}^{n}$) 生成搜索令牌 $TK_{i,Q}$, 发送到服务器. 服务器接收到 $TK_{i,Q}$ 后, 利用索引 $\mathcal{I}$ 在密文集合 C 上进行检索, 将满足搜索条件的文件集合 R 返回给区块链 BC, BC 对 R 中的文件进行验证. 验证完成后, 用户 DU 取回验证结果及文件.

表 1 符号表

名称	描述
n	关键词个数
m	文件个数
$W_{d=1}^{n}$	关键词列表, $d=1,2,\ldots,n$
DB	文件标识符-关键词对 $(id_i,w_i)_{i=1}^{d}$ 列表
w_i	文档 D_i 的关键词
$DB(w_i)$	关键词 w_i 对应的文档标识符的集合
$x \xleftarrow{\$} X$	从有限集X中均匀、随机的选取一个元素 x
D	文件集合, $D=\{D_1,D_2,\ldots,D_m\}$
D_{id}	文件标识为 id 的文件, $id=1,2,\ldots,m$
C	F对应的密文集合, $C=\{C_{D_1},C_{D_2},\ldots,C_{D_m}\}$
$C_{D_{id}}$	文件 D_{id} 对应的密文

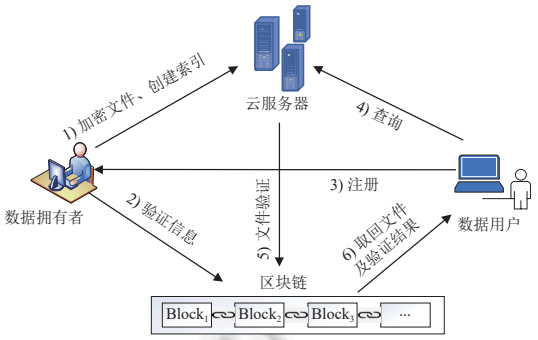

图 1 VDSSE 方案模型

2.2 威胁模型

本系统中, 服务器是不可信的, 密文检索过程中, 服务器可能为了节省经济利益而返回部分搜索结果, 有些恶意服务器可能会篡改用户存储的数据, 同时服务器上存储的数据也可能会被攻击者篡改, 因此服务器返回的搜索结果不可信. 此外, 用户可能为了避免付费而伪造验证结果, 因此, 用户也是不可信的. 系统中的其他实体 (数据拥有者、区块链) 是可信的.

2.3 算法定义

定义 1. 一个动态可验证 SSE 方案 Π 能够对 SSE 方案的检索结果进行验证, 并支持文件的动态更新. Π 包括 9 个多项式算法, 即 $\Pi=\{KeyGen,EDBSetup,ClientAuth,TokenGen,Search,Verify,Dec,TokenUp,Update\}$, 算法定义如下.

• 密钥生成算法, $K \leftarrow KeyGen(1^{\lambda})$, 数据拥有者 DO 输入秘密参数 λ, 输出系统密钥 K.

• 初始化算法, $(C,\mathcal{I},\mathcal{L}) \leftarrow EDBSetup(D, W, DB, K, \mathbb{U})$, 数据拥有者 DO 输入文件集合 D, 关键词集合 W, 文件列表 DB, 密钥 K, 属性集合 $\mathbb{U}$, 生成加密文件集合 U, 前向安全索引 $\mathcal{I}$ 以及验证列表 $\mathcal{L}$, 将 C、$\mathcal{I}$ 上传到服务器 CS, 将 $\mathcal{L}$ 上传到区块链 BC.

• 用户注册算法, $(K_w,\Sigma,sk_A^i) \leftarrow ClientAuth(u_i)$, 数据用户 DU 输入个人属性信息 u_i, 发送给数据拥有者 DO 进行注册, 注册完成后得到密钥 K_w、系统状态 Σ 以及属性私钥 sk_A^i.

• 搜索令牌生成算法, $TK_{i,Q} \leftarrow TokenGen(K_w,\Sigma,w_i)$, 数据用户 DU 输入密钥 K_w、状态 Σ 以及要搜索的关键词 w_i, 生成搜索令牌 $TK_{i,Q}$, 发送给服务器 CS.

• 搜索算法, $R \leftarrow Search(C,\mathcal{I},TK_{i,Q})$, 服务器 CS 输入搜索令牌 $TK_{i,Q}$, 加密文件集合 C 以及安全索引 $\mathcal{I}$, 执行密文检索, 输出搜索结果 R, 并把 R 发送给区块链 BC 进行完整性验证.

• 验证算法, $(R,proof) \leftarrow Verify(R,TK_{i,Q},\mathcal{L})$, 区块链 BC 利用验证合约, 将搜索结果 R、搜索令牌 $TK_{i,Q}$、验证列表 $\mathcal{L}$ 作为输入, 对搜索结果进行完整性验证, 输出验证信息 $proof$, 用户 DU 取回 $(R, proof)$.

• 解密算法, $D \leftarrow Dec(R,proof,sk_A^i)$, 数据用户 DU 根据区块链返回的验证信息 $proof$ 决定对搜索结果 R 接受或者拒绝, 如果接受文件则利用属性私钥 sk_A^i 解密出对称密钥 K, 从而对文件进行解密, 得到明文 D.

• 更新令牌生成算法, $(TK_{i,I},TK_{i,L},C_{D_{id}}) \leftarrow TokenUp(K,D_{id})$, 数据拥有者 DO 以系统私钥 K、要更新的文件 D_{id} 作为输入, 输出更新令牌 $TK_{i,I}$, $C_{D_{id}}$ 和 $TK_{i,L}$, 分别发送给服务器 CS 和区块链 BC.

• 更新算法, $(C',\mathcal{I}',\mathcal{L}') \leftarrow Update(C,C_{D_{id}},\mathcal{I},\mathcal{L},TK_{i,I},TK_{i,L})$, 服务器 CS 以密文集合 C、要更新的密文 $C_{D_{id}}$, 安

全索引$\mathcal{I}$、更新令牌$TK_{i,I}$为输入, 对文件动态更新 (添加或删除), 输出更新后的数据库C'和安全索引$\mathcal{I}'$; 区块链 BC 以验证列表$\mathcal{L}$、更新令牌$TK_{i,L}$为输入, 输出更新后的验证列表$\mathcal{L}'$.

2.4 安全定义

定义 2. 可验证性. 如果对于任意 $(D,W,\mathcal{I})$ 以及令牌$TK_{i,Q}$, 一个任意多项式时间 (*PPT*) 敌手$\mathcal{A}$能够篡改搜索结果的概率是可忽略的, 则称 VDSSE 满足可验证性.

由于云服务器是不可信的, 因此它返回的搜索结果可能是不正确的, 区块链应该能够识别这种错误从而保证搜索结果的可信性. 假设一个搜索令牌对应的密文集合是$R(w)$, 区块链存储的验证信息是$\mathcal{L}(w)$, 如果敌手$\mathcal{A}$利用伪造的 $(R'(w), \mathcal{L}'(w))$ 能够通过验证算法 *Verify*, 则敌手$\mathcal{A}$获胜.

定义 3. 更新安全性. VDSSE 能够实现更新安全, 即能够抵抗敌手$\mathcal{A}$在多项式时间内的重放攻击, 对于任意 $(D,W,\mathcal{I})$ 以及令牌$TK_{i,Q}$, 更新之前的搜索结果能够通过验证的概率是可忽略的.

由于云服务器 CS 是不可信的, 因此, 对于数据拥有者的更新请求, 云服务器可能不执行或者部分执行, VDSSE 利用算法 *Verify* 能验证文件是不是最新的. 给定一个搜索令牌对应的已更新且有效的密文集合$D'(w)$以及对应的验证信息$\mathcal{L}'(w)$, 如果敌手$\mathcal{A}$利用未更新的 $(D(w), \mathcal{L}(w))$ 能够通过验证算法 *Verify*, 则敌手$\mathcal{A}$获胜.

定义 4. 自适应选择关键字攻击安全 (CKA2-security). 对于动态可验证的对称可搜索加密方案$\Pi=\{KeyGen, EDBSetup, ClienAuth, TokenGen, Search, Update, Verify, Dec\}$, 定义泄漏函数$\mathcal{L}=\{\mathcal{L}_{\text{setup}}, \mathcal{L}_{\text{search}}, \mathcal{L}_{\text{update}}\}$, 对于敌手$\mathcal{A}$和模拟器$\mathcal{S}$, 通过进行两个实验$Real_{\mathcal{A}}(\lambda)$和$Ideal_{\mathcal{A},\mathcal{S}}(\lambda)$定义方案的 CKA2-security.

• $Real_{\mathcal{A}}(\lambda)$: 挑战者运行$KeyGen(1^{\lambda})$产生系统密钥$K$, 敌手$\mathcal{A}$输出文件集合$D$和关键字集合$W$, 挑战者运行$EDBSetup(DB,K)$, 生成索引$\mathcal{I}$和密文$C$, 并发送给敌手$\mathcal{A}$. 敌手$\mathcal{A}$进行多项式数量的自适应查询$Q=\{q_1,q_2,\ldots,q_t\}$, 对于每个查询$q_i$, 挑战者产生对应的搜索令牌和更新令牌并发送给敌手, 最后敌手返回一个比特b作为游戏的输出.

• $Ideal_{\mathcal{A},\mathcal{S}}(\lambda)$: 敌手$\mathcal{A}$输出文件集合$D$和关键字集合$W$, 给定泄露函数$\mathcal{L}_{\text{setup}}(D,W)$, 模拟器$\mathcal{S}$产生安全索引$\mathcal{I}'$和密文$C'$, 并发送给敌手$\mathcal{A}$. 敌手$\mathcal{A}$进行多项式数量的自适应查询$Q=\{q_1,q_2,\ldots,q_t\}$, 对于每个查询$q_i$, 给定泄露函数$\mathcal{L}_{\text{search}}(w)$、$\mathcal{L}_{\text{update}}(D)$, 模拟器$\mathcal{S}$产生对应的搜索令牌和更新令牌并发送给敌手, 最后敌手返回一个比特b作为游戏的输出.

如果对于任何 *PPT* 时间的敌手$\mathcal{A}$, 存在一个 *PPT* 时间的模拟器$\mathcal{S}$, 使得:

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

我们就说Π对于自适应选择关键字攻击是$\left(\mathcal{L}_{\text{setup}}, \mathcal{L}_{\text{search}}, \mathcal{L}_{\text{update}}\right)$安全的. 其中, *negl* 是可忽略函数.

定义 5. 前向安全性. 对于一个$\mathcal{L}$– 自适应安全 SSE 方案, 如果更新泄露函数$\mathcal{L}_{\text{update}}$具有如下形式:

$$\mathcal{L}_{\text{update}}(op,in)=\mathcal{L}'(op,\{(ind_i,u_i)\}),$$

则称该 SSE 方案满足前向安全性, 其中$\{(ind_i,u_i)\}$是与更新文档 ind_i 中修改的u_i个关键字对应的更新文档的集合, *op* 表示更新操作.

3 基于区块链的动态可验证对称可搜索加密方案

3.1 验证标签 *Vtag*

为了实现搜索结果验证, 文献 [30] 将每个文件的消息验证码 (MAC) 形成验证列表上传到区块链, 由区块链验证合约对文件进行验证, 假设关键词w_i对应 m 个文件 $D_1,D_2,\ldots,D_m$, 则验证列表$\mathcal{L}=\{MAC(D_1),MAC(D_2),\ldots,MAC(D_m)\}$, 验证列表占用的存储空间与文件数量呈线性关系, 文件数量很大时, 将占用很多的存储空间. 此外, 在文件动态更新时, 验证列表$\mathcal{L}'$需要重新上传到区块链, 由此会占用更大的存储空间. 由于区块链只增加不删除的性质, 存储空间是十分宝贵的, 存储在区块链上的数据应该尽可能少. 对此, 本文提出累积验证标签 *Vtag*:

$$Vtag_{w_i}=F(w_i)\times F(V)\times\prod_{j=1}^{m}prime(H(D_j)) \tag{1}$$

其中, F 是伪随机函数, 用来对关键词 w_i 和文件更新计数器 V 进行加密; V 用来统计系统中文件的更新次数, 用来抵抗重放攻击; $prime$ 是文件的哈希值到一个素数的映射, m 是关键词 w_i 对应的文件数量, H 为哈希函数. F, H 均是多项式函数, 在多项式时间内与随机函数不可区分.

验证标签 $Vtag$ 具有如下性质.

(1) 累积性. 对于关键词 w_i, 假设存在 n 个对应文件, 则 $Vtag_{w_i}=F(w_i)\times F(V)\times\prod_{j=1}^{m}prime(H(D_j))$, 此时如果再增加一个文件 D_{j+1}, 则对应的新的验证标签为: $Vtag'_{w_i}=F(w_i)\times F(V)\times prime(H(D_{j+1}))\times\prod_{j=1}^{m}prime(H(D_j))$, 即 $Vtag'_{w_i}=Vtag_{w_i}\times prime(H(D_{j+1}))$.

(2) 更新性. 更新包括文件添加、删除以及修改, 文件添加可以利用 $Vtag$ 本身的累加性实现, 下面主要讨论文件的删除以及修改.

文件删除: 假设要删除关键词 w_i 对应的文件 D_m, 则 $Vtag'_{w_i}=Vtag_{w_i}\times prime^{-1}(H(D_m))$.

文件修改: 假设将文件 $D_t(t\in n)$ 修改为 D'_t, 则 $Vtag'_{w_i}=Vtag_{w_i}\times prime^{-1}(H(D_t))\times prime(H(D'_t))$.

(3) 不可伪造性. 对于关键词 w_i, 假设该关键词对应的正确的加密文件的集合是 $E=\{D_{j_1},D_{j_2},\ldots,D_{j_z}\}$, 正确的验证标签是 $Vtag_E$. 如果敌手 $\mathcal{A}$ 能够伪造一个集合 $E'(E'\neq E)$, 使得 $Vtag_{E'}=Vtag_E$, 则 $\mathcal{A}$ 违背了伪随机函数的安全性.

令 $E'=\{D'_{j_1},D'_{j_2},\ldots,D'_{j_{z'}}\}$, 并假设 $z'\leqslant z$. 令 Δ 为 E 中文件的 id 的集合, Δ' 为 E 和 E' 中相同文件的 id 的集合. 当 $j_k\in\Delta'$, $D'_{j_k}=D_{j_k}$; 否则 $D'_{j_k}\neq D_{j_k}$ $(1\leqslant k\leqslant z')$, 或者 $D_{j_k}\in E$ $(z'<k\leqslant z)$. 为了易于表示, 假设 $\Delta'=\{j_1,j_2,\ldots,j_r\}(r\leqslant z')$, $\Delta/\Delta'=\{j_{r+1},j_{r+2},\ldots,j_z\}$. 由于 $Vtag_{E'}=Vtag_E$, 所以:

$$Vtag_E/Vtag_{E'}=prime(H(D_{r+1}))\times\ldots\times prime(H(D_z))=1 \tag{2}$$

敌手 $\mathcal{A}$ 可以分别对预言机 H 进行 $z-1$ 次随机询问, 不失一般性, 假设敌手已经从预言机询问了 $prime(H(D_{j_1}))$, $prime(H(D_{j_2})),\ldots,prime(H(D_{j_{z-1}}))$, 则根据公式 (2) 可知:

$$prime(H(D_z))=1/prime(H(D_{r+1}))\times\ldots\times prime(H(D_{z-1})).$$

显然 $prime(H(D_{r+1}))\times\ldots\times prime(H(D_{z-1}))$ 等于 0 的概率可以忽略, 因此存在敌手 $\mathcal{A}$ 在不知道函数 H 的密钥的情况下可以伪造 $prime(H(D_z))$, 这与伪随机函数 H 的安全性是相冲突的, 因此, 验证标签具有不可伪造性.

利用验证标签 $Vtag$ 累加性和更新性的性质, 可以实现文件的高效动态更新, 同时, 对于具有 m 个文件的关键词 w_i, 验证列表 $\mathcal{L}=Vtag$, 占用的存储空间为 $O(1)$, 验证列表占用的空间是固定的, 不会随着文件数量的增加而线性增加. 相比于文献 [30] 的 $O(m)$ 的存储空间, 本方案验证列表极大地节省了存储空间, 减轻了区块链的负担, 提高了空间利用率.

3.2 方案构造

本文提出的 VDSSE 方案, 框架如图 2 所示, 数据拥有者 DO 从文件中提取出关键词 $W_{d=1}^{n}$ ($W_{d=1}^{n}=\{w_1,w_2,w_3,\ldots,w_n\}$) 后, 生成索引 $\mathcal{I}$ 和验证列表 $\mathcal{L}$, 为了减少区块链的交易次数, 提高计算效率, 数据拥有者 DO 将索引 $\mathcal{I}$ 上传到服务器 CS, 由服务器 CS 执行密文检索. 同时, 为了实现密文检索结果可靠、公平的验证, 数据拥有者 DO 将验证列表 $\mathcal{L}$ 上传到区块链 BC, 服务器 CS 将满足查询条件的搜索结果 R 上传到区块链 BC, 由区块链实现对搜索结果 R 的验证. 验证完成后用户 DU 取回验证结果 $proof$ 和搜索结果 R, 用户 DU 根据验证结果 $proof$ 接受或拒绝搜索结果 R. VDSSE 方案包括 9 个多项式算法: $KeyGen, EDBSetup, ClientAuth, TokenGen, Search, Verify, Dec, TokenUp, Update$, 主要算法设计如下.

● 密钥生成算法: $K\leftarrow KeyGen(1^{\lambda})$, 数据拥有者 DO 利用安全参数 λ 生成系统密钥 $K=(K_{id}K_w,mpk,msk)$, 其中 K_{id} 是用于对文件加密的对称密钥, K_w 是伪随机函数 F 的随机密钥, mpk 是系统公钥, msk 是系统主密钥, $(mpk,msk)\leftarrow ABE.Setup(1^{\lambda})$.

● 初始化算法: $(C,\mathcal{I},\mathcal{L})\leftarrow EDBSetup(D, W, DB, K, \mathbb{U})$, 数据拥有者 DO 提取文件集合 D 所有关键词, 构建关键词集合 W; 利用对称密钥 K_{id} 对文件集合 D 中的文件 D_{id} 进行加密, 形成密文集合 C; 利用伪随机函数生成前向

安全索引 $\mathcal{I}$; 利用 C 中的加密文件生成验证列表 $\mathcal{L}$, 并将 $(\mathcal{I}, C)$ 上传到服务器用于执行密文检索, 将验证列表 $\mathcal{L}$ 发送到区块链, 用于验证. 同时, 为了实现多客户端管理, 由 DO 根据属性集合 $\mathbb{U}$ 指定访问策略, 利用属性加密 *ABE.Enc* 对文件对称加密密钥 K_{id} 进行加密, 只有用户属性满足 *ABE* 访问策略时才能解密对称密钥 K_{id}, 从而解密文件, 过程如算法 1 所示. 其中, $H_i: \{0,1\}^* \to \{0,1\}^{\lambda}(i=1,2,3)$, 是防碰撞哈希函数, *prime* 是哈希散列值到大素数的映射函数.

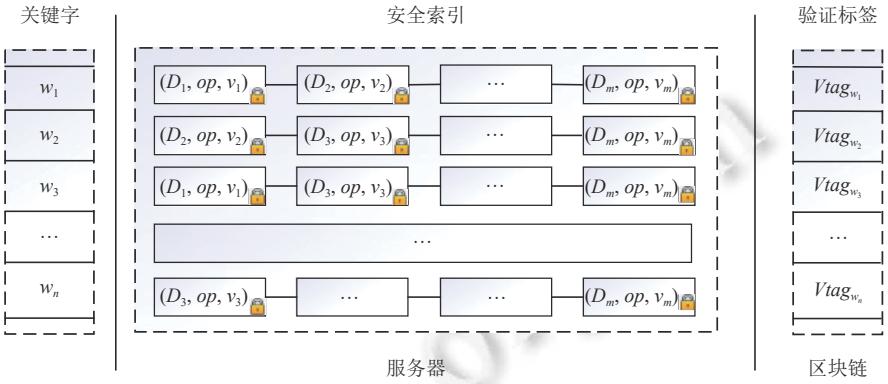

图 2 VDSSE 框架图

算法 1. *EDBSetup*.

输入: D, W, DB, K, $\mathbb{U}$;
输出: C, $\mathcal{I}$, $\mathcal{L}$.

1. function $EDBSetup(D, W, DB, K, \mathbb{U})$
2. $\mathcal{I} \leftarrow \{\}; \mathcal{L} \leftarrow \{\}; \Sigma \leftarrow \{\}; C \leftarrow \{\}; Opp \leftarrow 1$
3. for $w \in W$ do
4. $stag_w \leftarrow F(K_w, w)$
5. $st_v \xleftarrow{\$} \{0,1\}^{\lambda}; v \leftarrow 0$
6. for $id \in DB[w]$ do
7. $K_{v+1} \xleftarrow{\$} \{0,1\}^{\lambda}; C_k \leftarrow ABE.Enc(mpk, K_{id}, \mathbb{U}); st_{v+1} \leftarrow P(K_{v+1}, st_v)$
8. $e \leftarrow (id \| op \| v \| K_{v+1} \| C_k) \oplus H_1(stag_w \| st_{v+1})$
9. $C_{D_{id}} \leftarrow Enc(K_{id}, D_{id}); C \leftarrow C \cup C_{D_{id}}$
10. $\ell \leftarrow H_2(stag_w \| st_{v+1}); \mathcal{I}[\ell] = e$
11. $v \leftarrow v+1; Opp \leftarrow Opp \times prime(H_3(C_{D_{id}}))$
12. end for
13. $\Sigma[w] \leftarrow (st_{v+1}, v+1); Vtag_w \leftarrow stag_w \times F(v) \times Opp$
14. $\mathcal{L}[stag_w] \leftarrow Vtag_w$
15. end for
16. 发送安全索引 $\mathcal{I}$ 和密文集合 C 到服务器 CS
17. 发送验证列表 $\mathcal{L}$ 到区块链 BC
18. end function

● 用户注册算法: $(K_w, \Sigma, sk_A^i) \leftarrow ClientAuth(u_i)$, 用户 DU 加入系统时, 首先向数据拥有者 DO 提交个人属性信息 u_i, 进行注册, 获取注册后的密钥 K_w, 状态 Σ 及属性私钥 sk_A^i, 用于生成搜索令牌及文件解密.

● 搜索令牌生成算法: $TK_{i,Q} \leftarrow TokenGen(K_w,\Sigma,w_i)$, 对于一次关键词查询 Q, 用户 DU 利用密钥 K_w、状态 Σ 和关键词 w_i 生成搜索令牌 $TK_{i,Q}$, 发送给服务器 CS 进行搜索, 过程如算法 2 所示.

算法 2. *TokenGen*.

输入: w_i, K, Σ;
输出: $TK_{i,Q}$.

1. function *TokenGen*(K, w_i)
2. $TK_{i,Q}=\{\}$
3. $stag_w \leftarrow F(K_w,w)$; $(st_v,v) \leftarrow \Sigma[w]$
4. if $(st_v,v) = \perp$ then
5. return $\varnothing$
6. else
7. return $TK_{i,Q} = (stag_w, st_v, v)$
8. 发送 $TK_{i,Q}$ 到服务器 CS
9. end function

● 搜索算法: $R \leftarrow Search(C,\mathcal{I},TK_{i,Q})$, 服务器 CS 接收到用户 DU 发送的搜索令牌 $TK_{i,Q}$ 后, 利用安全索引 $\mathcal{I}$ 执行关键词检索, 找到符合查询条件的文件 id, 然后利用 id 找到对应的密文 $C_{D_{id}}$, 生成文件集合 R, 服务器将 R 和搜索令牌 $TK_{i,Q}$ 返回给区块链 BC 进行验证, 过程如算法 3 所示.

算法 3. *Search*.

输入: C, $\mathcal{I}$, $TK_{i,Q}$;
输出: R.

1. function Search (C, $\mathcal{I}$, $TK_{i,Q}$)
2. $R \leftarrow \{\}$; $c \leftarrow 0$
3. $(stag_w, st_v, v) \leftarrow TK_{i,Q}$; $c \leftarrow v$
4. for $c = v$ to 1 do:
5. $\ell \leftarrow H_2(stag_w \| st_v)$
6. if ($\mathcal{I}[\ell] = \varnothing$)
7. return $\varnothing$;
8. else
9. $e \leftarrow \mathcal{I}[\ell]$; $(id\|op\|v\|K_{id}\|C_k) \leftarrow e \oplus H_1(stag_w \| st_v)$
10. if (op="add")
11. $R \leftarrow R \cup C_{D_{id}} \cup C_k$
12. else
13. $R \leftarrow R - C_{D_{id}} - C_k$
14. $st_{v-1} \leftarrow P^{-1}(K_{id}, st_v)$
15. end for
16. return R
17. end function

● 验证算法: $(R, proof) \leftarrow Verify(R,TK_{i,Q},\mathcal{L})$, 如算法 4 所示, 区块链 BC 接收到服务器 CS 发送的验证文件 R 后, 计算文件的验证标签 $Vtag_w$, 并与验证列表中的标准验证标签 $Vtag'_w$ 比较, 如果 $Vtag'_w = Vtag_w$, 表示验证通

过, $proof=1$, 如果$Vtag'_w \neq Vtag_w$, 则表明 R 中文件不完整, 验证不通过, $proof=0$. 最后, 用户 DU 取回文件 R 和验证结果 $proof$.

算法 4. *Verify*.

输入: R, $TK_{i,Q}$, $\mathcal{L}$;
输出: R, *proof*.

1. function *Verify*(R, $TK_{i,Q}$, $\mathcal{L}$)
2. $proof \leftarrow 0; Opp \leftarrow 1; id \leftarrow 0$
3. $(stag_w, st_v, v) \leftarrow TK_{i,Q}; Vtag'_w \leftarrow \mathcal{L}[stag_w]$
4. for $C_{D_{id}} \in R$ do
5. $Opp \leftarrow Opp \times prime(H_3(C_{D_{id}}))$
6. end for
7. $Vtag_w \leftarrow stag_w \times F(v) \times Opp$
8. if ($Vtag'_w = Vtag_w$) $proof = 1$
9. else $proof = 0$
10. return (R, $proof$)
11. end function

● 解密算法: $D \leftarrow Dec(R, proof, sk_A^i)$, 数据用户 DU 首先判断区块链返回的验证信息 $proof$, 如果 $proof=0$, 则表示验证失败, 服务器返回的文件不完整, 拒绝本次搜索结果; 如果 $proof=1$, 表明服务器返回的文件是完整的, 接受文件并利用属性私钥 sk_A^i 对 R 中的 C_k 进行解密, 得出文件加密密钥 K_{id}, 从而对文件进行解密, 得到明文 D.

● 更新令牌算法: $(TK_{i,I}, TK_{i,L}, C_{D_{id}}) \leftarrow TokenUp(K, D_{id})$, 数据拥有者 DO 利用系统密钥 K、关键词 w_i 对应的文件 D_{id} 分别生成索引更新令牌 $TK_{i,I}$ 和验证列表更新令牌 $TK_{i,L}$, 分别用于对文件 D_{id} 进行更新 (更新操作用 op 表示, $op=$ "*add*" 文件添加、 $op=$ "*del*" 文件删除) 以及验证列表更新.

算法 5. *TokenUp*.

输入: K, D_{id};
输出: $TK_{i,I}$, $TK_{i,L}$, $C_{D_{id}}$.

1. function *TokenUp*(K, D_{id})
2. $Opp \leftarrow 1; \alpha \leftarrow 0$
3. $(st_v, v) \leftarrow \Sigma[w]; stag_{tw} \leftarrow F(K_w, w)$
4. $Vtag_w \leftarrow \mathcal{L}[stag_w]; \alpha \leftarrow f(v)$
5. if $(st_v, v) = \perp$ then
6. $st_0 \xleftarrow{\$} \{0,1\}^\lambda; v \leftarrow 0$
7. end if
8. $K_{v+1} \xleftarrow{\$} \{0,1\}^\lambda; C_k \leftarrow ABE.Enc(mpk, K_{id}, \mathbb{U}); st_{v+1} \leftarrow P(K_{v+1}, st_v)$
9. $e \leftarrow (id \| op \| v \| K_{v+1} \| C_k) \oplus H_1(stag_w \| st_{v+1})$
10. $C_{D_{id}} \leftarrow Enc(K_{id}, D_{id}); \ell \leftarrow H_2(stag_w \| st_{v+1})$
11. $TK_{i,I} \leftarrow (\ell, e); v \leftarrow v+1$
12. $Opp \leftarrow prime(H_3(C_{D_{id}})); \alpha \leftarrow \alpha^{-1} \times f(v)$
13. if (op="*add*")

14. $Vtag'_w \leftarrow Vtag_w \times \alpha \times Opp$
15. else
16. $Vtag'_w \leftarrow Vtag_w \times \alpha \times Opp^{-1}$
17. $TK_{i,L} \leftarrow (stag_w, Vtag'_w)$
18. 发送$(TK_{i,I}, C_{D_{id}})$到服务器 CS
19. 发送$TK_{i,L}$到区块链 BC
20. end function

● 更新算法: $(C', \mathcal{I}', \mathcal{L}') \leftarrow Update(C, C_{D_{id}}, \mathcal{I}, \mathcal{L}, TK_{i,I}, TK_{i,L})$, 云服务器 CS 接收到更新令牌$TK_{i,I}$后解析为$(\ell, e)$, 添加到索引$\mathcal{I}$, 形成新的索引$\mathcal{I}'$, 同时更新加密文件集合 C 为C'. 区块链 BC 接收到更新令牌$TK_{i,L}$后解析为$(stag_w, Vtag'_w)$, 将验证列表$\mathcal{L}$中位置$stag_w$对应的值更改为$Vtag'_w$, 实现验证列表从$\mathcal{L}$到$\mathcal{L}'$的更新.

3.3 安全性分析

定理 1. VDSSE 满足可验证性.

证明: 假设一个多项式时间敌手$\mathcal{A}$利用伪造的$(R'(w), \mathcal{L}'(w))$能够通过验证算法*Verify*, 而正确的搜索结果和验证信息是$(R(w), \mathcal{L}(w))$, 我们将证明不存在这样的敌手$\mathcal{A}$使得$(R'(w), \mathcal{L}'(w))=(R(w), \mathcal{L}(w))$.

令 t 表示$R(w)$中文件的 id 的集合, $C_{D_j} \in R(w)$; t' 表示$R'(w)$中文件的 id 的集合, $C'_{D_j} \in R'(w)$.

我们将讨论下述情况:

① $R'(w) = R(w)$且$\mathcal{L}'(w) \neq \mathcal{L}(w)$. 根据前文所述, 可知: $\mathcal{L}(w) = Vtag_w = f(\pi(w)) \times f(v) \times \prod_{j=1}^{t} prime(H(C'_{D_j}))$, $\mathcal{L}'(w) = Vtag_w = f(\pi(w)) \times f(v) \times \prod_{j=1}^{t'} prime(H(C'_{D_j}))$, 由于$R'(w) = R(w)$, 可以得出$\mathcal{L}'(w) = \mathcal{L}(w)$, 这与$\mathcal{L}'(w) \neq \mathcal{L}(w)$是矛盾的, 因此这种情况不成立.

② $R'(w) \neq R(w)$且$\mathcal{L}'(w) = \mathcal{L}(w)$. 这种情况表明敌手$\mathcal{A}$可以伪造验证标签, 这与第 3.1 节描述的验证标签的不可伪造性相矛盾, 因此这种情况不成立.

③ $R'(w) \neq R(w)$且$\mathcal{L}'(w) \neq \mathcal{L}(w)$. 这种情况与验证标签的不可伪造性证明过程类似, 与伪随机函数的安全性相矛盾, 因此这种情况不成立.

定理 2. VDSSE 满足更新安全性.

证明: 对于一次数据更新请求, 假设存在多项式时间敌手$\mathcal{A}$没有对文件进行更新, 对于$\mathcal{A}$返回的搜索结果, 我们将证明 VDSSE 验证信息$proof = 0$.

对于关键词w_i, 假设当前查询返回的搜索结果$D(w_i)$ $(D(w_i) = \{C_{D_1}, C_{D_2}, \ldots, C_{D_t}\}, t \in m)$, 更新次数是$v$, 对应的验证标签$Vtag_{w_i}$:

$$Vtag_{w_i} = f(\pi(w_i)) \times f(v) \times \prod_{j=1}^{t} prime(H(C_{D_j})) \tag{3}$$

经过k $(k \geqslant 1)$次更新后, $v' = v + k$, 此时验证标签$Vtag'_{w_i}$:

$$Vtag'_{w_i} = f(\pi(w_i)) \times f(v') \times \prod_{j=1}^{t} prime(H(C_{D_j})) \tag{4}$$

如果更新前的验证标签通过了*Verify*, 则对于敌手$\mathcal{A}$, $Vtag_w = Vtag'_w$, 对比公式 (3) 和公式 (4), 可以得出$f(v) = f(v')$, 即敌手$\mathcal{A}$能够找到一个v和一个v', 使得$f(v) = f(v')$, 这与伪随机函数 PRF f 的安全性是违背的, 因此更新前的文件不能通过算法*Verify*, $proof = 0$, 所以 VDSSE 满足更新安全性.

定理 3. 如果函数f、π、F是伪随机的, 对称可搜索加密方案是选择明文攻击安全的 (CPA-secure), 那么 VDSSE 对于自适应选择关键字攻击是$\mathcal{L} = \left(\mathcal{L}_{\text{setup}}, \mathcal{L}_{\text{search}}, \mathcal{L}_{\text{update}}\right)$安全的.

证明: 我们构建了一个多项式时间的模拟器$\mathcal{S}$, 对于任意多项式时间敌手$\mathcal{A}$, 通过证明$Real_{\mathcal{A}}(\lambda)$和$Ideal_{\mathcal{A},\mathcal{S}}(\lambda)$输出的不可区分实现定理的证明. 在$Real_{\mathcal{A}}(\lambda)$中, 敌手$\mathcal{A}$收到安全索引$\mathcal{I}$、密文集合$C$、搜索令牌$ST$和更新令牌

$TK_{i,I}$、$TK_{i,L}$, 在$Ideal_{\mathcal{A},\mathcal{S}}(\lambda)$中, 敌手$\mathcal{A}$收到安全索引$\mathcal{I}'$、密文集合$C'$、搜索令牌$ST'$和更新令牌$TK'_{i,I}$、$TK'_{i,L}$, 证明$Real_{\mathcal{A}}(\lambda)$和$Ideal_{\mathcal{A},\mathcal{S}}(\lambda)$输出的不可区分即要证明($\mathcal{I}$, C, ST, $TK_{i,I}$, $TK_{i,L}$)和($\mathcal{I}'$, C', ST', $TK'_{i,I}$, $TK'_{i,L}$)的不可区分.

● 模拟安全索引$\mathcal{I}'$: 模拟器$\mathcal{S}$初始化$|W|$个列表$\mathcal{I}_{w_i(1\leqslant i\leqslant|W|)}$, 在$\{0,1\}^{\lambda}$中均匀随机的选择$|W|$个元素$p$, 在$Real_{\mathcal{A}}(\lambda)$中安全索引$\mathcal{I}$由伪随机函数$F(K_w,w)$、$F(stag_w,v)$生成, 而在$Ideal_{\mathcal{A},\mathcal{S}}(\lambda)$中, 模拟器$\mathcal{S}$利用输出长度相同的、均匀随机选择的元素$p$拟生成安全索引$\mathcal{I}'$, 由伪随机函数的安全性可知, 在无法知道伪随机函数密钥K_w的情况下, 敌手$\mathcal{A}$无法区分$\mathcal{I}$和$\mathcal{I}'$.

● 模拟密文C': 对于模拟的密文文档C' $\left(C'=\{C'_1,C'_2,\ldots,C'_m\}\right)$, 对每个文件$C'_i$ $(1\leqslant i\leqslant m)$, 模拟器$\mathcal{S}$生成$|C'_i|$长度的零字符串替换文件, 由于本方案中对称加密是选择明文攻击安全的, 因此, 在给定泄漏函数$\mathcal{L}_{\text{setup}}(\mathcal{F},W)$下, 敌手$\mathcal{A}$无法区分$Real_{\mathcal{A}}(\lambda)$中密文$C$和$Ideal_{\mathcal{A},\mathcal{S}}(\lambda)$中密文$C'$.

模拟搜索令牌ST': 给定泄漏函数$\mathcal{L}_{\text{search}}(w)$, 模拟器$\mathcal{S}$计算搜索令牌$ST'$, 由于$ST'$由伪随机函数$F(K_w,w')$生成, 由伪随机函数的安全性可知, 敌手$\mathcal{A}$无法区分$ST$和$ST'$.

● 模拟更新令牌UT': 给定泄漏函数$\mathcal{L}_{\text{update}}(F)$, 模拟器$\mathcal{S}$计算更新令牌$TK_{i,I}$和$TK_{i,L}$, 安全索引更新令牌$TK_{i,I}$和验证列表更新令牌$TK_{i,L}$均是基于伪随机函数实现, 利用相同长度的随机字符串替换$TK_{i,I}$和$TK_{i,L}$对应的输出, 根据伪随机函数的安全性可知, 敌手$\mathcal{A}$无法区分$TK_{i,I}$和$TK'_{i,I}$以及$TK_{i,L}$和$TK'_{i,L}$.

综上所述, 对于任意多项式敌手$\mathcal{A}$, $Real_{\mathcal{A}}(\lambda)$和$Ideal_{\mathcal{A},\mathcal{S}}(\lambda)$输出的不可区分的, 即:

$$\left|\Pr[Real_{\mathcal{A}}(\lambda)=1]-\Pr[Ideal_{\mathcal{A},\mathcal{S}}(\lambda)=1]\right|\leqslant negl(\lambda) \tag{5}$$

因此, VDSSE 满足自适应选择关键字攻击安全, 证毕.

定理 4. VDSSE 满足前向安全性.

证明: 根据前向安全的定义: $\mathcal{L}_{\text{update}}(op,in)=\mathcal{L}'(op,\{(ind_i,u_i)\})$, 我们利用模拟器$\mathcal{S}$模拟更新令牌, 从而证明 VDSSE 的前向安全性. 由于前向安全主要发生在文件更新过程中, 对文件验证过程几乎没有影响, 因此, 本证明中略去了验证部分, 而集中于文件更新部分.$\mathcal{S}$模拟更新令牌算法如算法 6 所示.

算法 6. $\mathcal{S}.Tokenup$.

输入: ind_i, u_i;
输出: $TK'_{i,I}$.

1. function $TokenGen(ind_i, u_i)$
2. $TK'_{i,I}=\{\}$
3. for $j=1$ to u_i do
4. 随机生成(ℓ,e)对;
5. 将$(ind_i,(\ell,e))$添加到字典 E;
6. end for
7. $TK_{i,I}\leftarrow(\ell,e)$
8. 发送$TK_{i,I}$到服务器 CS
9. end function

在模拟算法中, 我们利用满足随机预言机模型的随机字符串来代替哈希函数H的输出, 生成(ℓ,e)对, 并存储在字典E中, 当再次需要使用该哈希函数的输出时, 直接从E中读取, 而不用再重新生成. 随机字符串的大小与真实字符串的大小完全相同, 模拟算法模拟的令牌与真实的令牌格式和大小完全相同, 因此敌手$\mathcal{A}$无法区分, 证明了在$\mathcal{L}_{\text{update}}(op,in)=\mathcal{L}'(op,\{(ind_i,u_i)\})$前提下, 就可以模拟更新令牌, 从而证明了方案的前向安全性.

4 实验分析

4.1 功能分析

将本文方案与同样基于区块链实现 SSE 验证的相关文献 [27,28,30] 进行对比分析, 如表 2 所示. 文献 [27,28,30] 和本文提出的方案都是基于区块链实现对 SSE 方案的搜索结果的验证, 都能实现验证结果的正确性和验证的公平性. 但是, 文献 [27] 利用多集哈希函数实现了文件的动态更新, 方案中没有实现文件更新的前向安全; 文献 [28] 实现了服务器和用户之间的公平性验证, 然而验证过程是基于消息验证码的单次验证, 是一种静态验证; 文献 [30] 通过验证文件的消息验证码实现了搜索结果的验证, 并且基于属性加密实现了一对多场景下的 SSE 方案的验证, 但是缺少对文件更新后的验证, 同样是一种静态的验证. 本文通过对称加密技术保证了文件动态更新过程中的前向安全性, 在此基础上利用区块链实现了 SSE 搜索结果的验证, 保证了验证的公平性, 同时利用验证标签 $Vtag_w$ 实现了 SSE 的动态验证, 同时本方案支持多客户端的场景, 具有更好的实用性.

表 2 方案对比

方案	可验证	基于区块链	动态验证	前向安全	多客户端
文献[27]	√	√	√	×	×
文献[28]	√	√	×	×	×
文献[30]	√	√	×	×	√
本文方案	√	√	√	√	√

4.2 性能分析

(1) 性能对比

从验证信息占用的存储空间、安全索引生成、加密搜索、结果验证 4 个方面对本文提出的 VDSSE 方案与文献 [27,28,30] 进行性能对比分析, 结果如表 3 所示. 其中, N 表示关键词个数, M 表示关键词 w_i ($i \in [1,N]$) 对应的文件个数, T_h 表示哈希运算的时间, T_f 表示伪随机函数运算时间, T_m 表示模乘运算时间, T_a 表示属性加密时间, T_c 表示区块链一次交易时间, $|H|$ 表示哈希运算的长度.

表 3 计算量和存储代价对比

方案	索引生成阶段	搜索阶段	验证阶段	存储空间
文献[27]	$2M\cdot T_h + N\cdot T_f + 2N\cdot T_h$	$T_c + 2M\cdot T_h$	$3M\cdot T_h + T_m$	$N\cdot M\cdot \lvert H\rvert$
文献[28]	$3N\cdot T_f + N\cdot T_h$	$6T_c$	$(M+3)T_h + 2T_c$	$3N\cdot \lvert H\rvert$
文献[30]	$T_m + T_a + N\cdot T_f$	T_c	$M\cdot T_h$	$N\cdot M\cdot \lvert H\rvert$
本文方案	$M\cdot T_a + (2M+N)\cdot T_f + 2M\cdot T_h + M\cdot T_m$	$M\cdot T_f + M\cdot T_h$	$2M\cdot T_h + (M+1)T_m$	$2N\cdot \lvert H\rvert$

(2) 实验分析

为准确评估本方案的性能, 本文基于真实数据集 Enron Email Dataset 对本文提出的 SSE 方案进行了实验分析, 并与具有相似功能的文献 [30] 方案进行了对比. 我们用 Python 实现了本文提出的 SSE 的算法, 采用 Solidity 构建以太坊智能合约. 为了更好地模拟普通的客户端环境, 我们的实验部署在 Intel Core i7 CPU、8 GB RAM 的笔记本电脑上, 电脑采用 Linux 系统, 我们采用以太坊区块链的本地网络模拟环境 Ganache 对智能合约进行评估. 实验过程中, 首先对数据集 Enron Email Dataset 提取关键词, 本试验中共提取关键词 1 672 878 个, 并根据关键词生成倒排索引, 利用倒排索引构建 SSE 方案. 实验中, 对称加密函数采用 AES-128, 伪随机函数采用 HMAC-256.

由于区块链只附加的性质, 链上的存储空间十分宝贵, 因此保存在区块链上的验证信息应该占用尽可能少的空间. 从图 3 可以看出, 文献 [30] 在进行 SSE 方案验证时占用的区块链的存储空间大小随着文件数量增加呈线性

增长, 文件数量从 10k 到 160k 时, 存储空间从 0.678 MB 到 9.76 MB, 这是因为文献 [30] 的 SSE 方案采用文件 MAC 值进行验证, 区块链上需要存储所有文件的 MAC 值, 而在本文提出的 VDSSE 方案中, 验证信息只包括关键词标识和验证标签, 与文件数量无关, 因此文件数量从 10k 到 160k 时, 存储空间从 0.013 MB 到 0.095 MB, 占用区块链存储空间显著减小.

图 4 显示了构建关键词索引花费的时间与文档数量的关系, 其中关键词数量从 115 到 1 235, 从中可以看出, 本文提出的方案比文献 [30] 提出的方案花费时间更多, 这是因为本文提出的方案采用倒排索引, 并实现了前向安全, 因此构建索引需要的时间与文档数量呈线性关系, 而在文献 [30] 中, 没有实现前向安全, 索引采用键值对的形式, 建立索引的时间仅与关键词的数量相关, 在实际的数据集中, 一个关键词可能对应多个文件, 因此本文提出的方案在索引建立阶段比文献 [30] 花费的时间多.

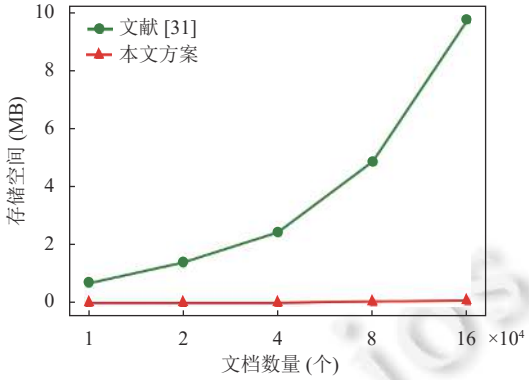

图 3 存储空间成本

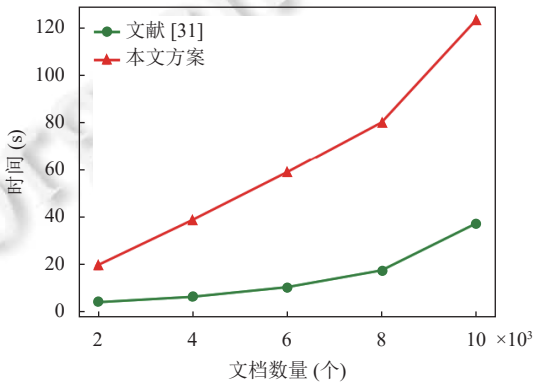

图 4 构建关键词索引

在关键词搜索阶段, 文献 [30] 通过建立搜索合约、依靠区块链实现关键词搜索. 用户进行关键词搜索时需要与区块链进行交互, 产生一次搜索交易, 会额外产生一次交易时间并需要花费一定的 Gas, 而本文提出的 VDSSE 方案在关键词搜索时用户直接向服务器发起请求, 由服务器进行关键词检索, 不需要区块链参与计算, 从图 5 可以看出, 本文方案搜索效率明显高于文献 [30].

如图 6 所示, 文件验证阶段, 文献 [30] 采用静态验证方式, 通过对比文件的 MAC 值实现文件验证. 本文采用动态验证方式, 相比于文献 [30] 的静态验证, 增加了一次模乘运算, 因此验证花费的时间略大. 虽然我们的方案验证时间比文献 [30] 略大, 但却可以实现文件的动态验证, 也能抵御文件重放攻击, 有更好的安全性.

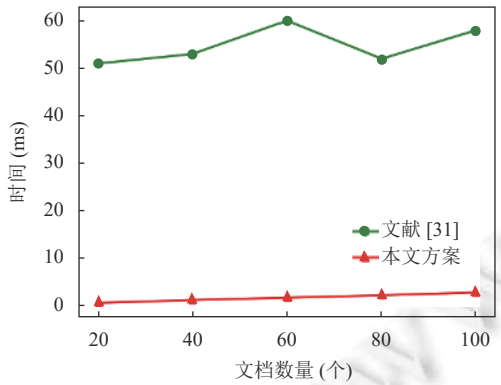

图 5 关键词搜索时间

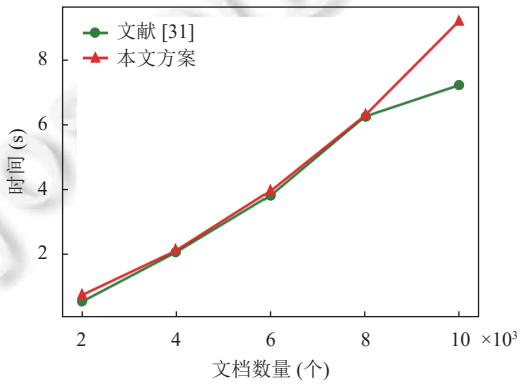

图 6 验证时间

图 7 显示了验证过程中的 Gas 消耗情况, 其中每个文件的大小为 1 KB. 根据公式 (1) 可以得出, 验证过程中 Gas 消耗主要包含两部分: 对文件进行哈希计算的 Gas 消耗和基本运算 (乘法、模幂等) 的 Gas 消耗. 从图 7 中可

以看出文件数量越多, 消耗的 Gas 越多, 这是因为文件数量越多, 文件的规模越大, 同时执行的基本运算次数越多, 因此需要更多 Gas. 同时, 由图 7 可以看出, 文件计算代价 Gas 消耗是验证过程中 Gas 消耗的主要部分. 当前以太坊系统中, 区块大小取决于 GasLimit, 当验证过程中消耗的 Gas 小于 GasLimit 时, 一次交易就可以完成搜索结果的验证, 而当验证过程中消耗的 Gas 大于 GasLimit 时, 就需要多次交易完成搜索结果的验证, 验证算法的效率就取决于系统的吞吐量, 吞吐量越高, 验证效率越高.

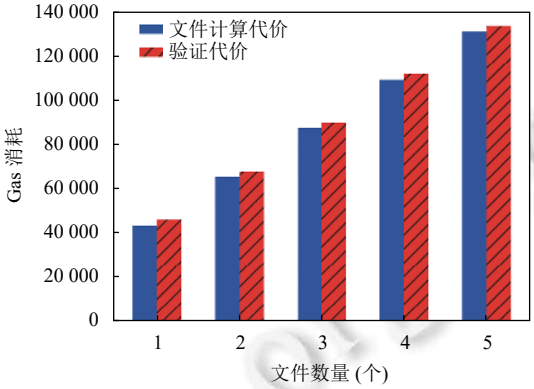

图 7 验证过程中的 Gas 消耗

5 结 论

为了解决密文检索过程中半诚实或不诚实服务器模型下搜索结果的公平性验证问题, 本文提出了基于区块链的动态可验证对称可搜索加密方案 VDSSE. 在保证前向安全的基础上, 利用区块链实现了搜索结果的完整性验证, 由于区块链具有不可篡改的性质, 既能防止不可信服务器恶意篡改数据也能防止恶意用户伪造验证结果, 保证了服务器和用户之间验证的公平性. 验证过程中, 本文方案利用累积性的验证标签 *Vtag*, 实现了验证信息的压缩, 降低了区块链存储开销, 同时实现了加密检索的动态验证. 安全性分析和实验结果表明, 本文方案满足更新安全, 能够抵御自适应选择关键字攻击, 同时在检索效率、存储空间开销等方面与现有的方案相比有着显著优势. 下一步, 我们将探索实现多关键词查询验证, 使得查询条件更加灵活.

References:

[1] Song DX, Wagner D, Perrig A. Practical techniques for searches on encrypted data. In: Proc. of the 2000 IEEE Symp. on Security and Privacy. Berkeley: IEEE, 2000. 44−55. [doi: 10.1109/SECPRI.2000.848445]

[2] Kamara S, Papamanthou C, Roeder T. Dynamic searchable symmetric encryption. In: Proc. of the 2012 ACM Conf. on Computer and Communications Security. Raleigh: ACM, 2012. 965–976. [doi: 10.1145/2382196.2382298]

[3] Boneh D, Di Crescenzo G, Ostrovsky R, Persiano G. Public key encryption with keyword search. In: Proc. of the 2004 Int'l Conf. on the Theory and Applications of Cryptographic Techniques. Interlaken: Springer, 2004. 506–522. [doi: 10.1007/978-3-540-24676-3_30]

[4] Tahir S, Ruj S, Rahulamathavan Y, Rajarajan M, Glackin C. A new secure and lightweight searchable encryption scheme over encrypted cloud data. IEEE Trans. on Emerging Topics in Computing, 2019, 7(4): 530–544. [doi: 10.1109/TETC.2017.2737789]

[5] Li HW, Yang Y, Dai YS, Yu S, Xiang Y. Achieving secure and efficient dynamic searchable symmetric encryption over medical cloud data. IEEE Trans. on Cloud Computing, 2020, 8(2): 484–494. [doi: 10.1109/TCC.2017.2769645]

[6] Sun SF, Zuo C, Liu JK, Sakzad A, Steinfeld R, Yuen TH, Yuan XL, Gu DW. Non-interactive multi-client searchable encryption: Realization and implementation. IEEE Trans. on Dependable and Secure Computing, 2022, 19(1): 452–467. [doi: 10.1109/TDSC.2020.2973633]

[7] Zhang MW, Chen Y, Huang JJ. SE-PPFM: A searchable encryption scheme supporting privacy-preserving fuzzy multikeyword in cloud systems. IEEE Systems Journal, 2021, 15(2): 2980–2988. [doi: 10.1109/JSYST.2020.2997932]

[8] Goh EJ. Secure indexes. Technical Report, Stanford: Stanford University, 2003.

[9] Curtmola R, Garay J, Kamara S, Ostrovsky R. Searchable symmetric encryption: Improved definitions and efficient constructions. In:

Proc. of the 13th ACM Conf. on Computer and Communications Security. Alexandria: ACM, 2006. 79–88. [doi: 10.1145/1180405.1180417]

[10] Cash D, Grubbs P, Perry J, Ristenpart T. Leakage-abuse attacks against searchable encryption. In: Proc. of the 22nd ACM SIGSAC Conf. on Computer and Communications Security. Los Angeles: ACM, 2015. 668–679. [doi: 10.1145/2810103.2813700]

[11] Zhang YP, Katz J, Papamanthou C. All your queries are belong to us: The power of file-injection attacks on searchable encryption. In: Proc. of the 25th USENIX Conf. on Security Symp. Austin: USENIX Association, 2016. 707–720.

[12] Stefanov E, Papamanthou C, Shi E. Practical dynamic searchable encryption with small leakage. In: Proc. of the 21st Network & Distributed System Security Symp. (NDSS). San Diego: The Internet Society, 2014. 1–15.

[13] Garg S, Mohassel P, Papamanthou C. TWORAM: Roundoptimal oblivious RAM with applications to searchable encryption. Cryptology ePrint Archive, 2015: Paper 2015/1010.

[14] Bost R. ∑οφος: Forward secure searchable encryption. In: Proc. of the 2016 ACM SIGSAC Conf. on Computer and Communications Security. Vienna: ACM, 2016. 1143–1154. [doi: 10.1145/2976749.2978303]

[15] Wei Y, Lv SY, Guo XJ, Liu ZL, Huang YY, Li B. FSSE: Forward secure searchable encryption with keyed-block chains. Information Sciences, 2019, 500: 113–126. [doi: 10.1016/j.ins.2019.05.059]

[16] Amjad G, Kamara S, Moataz T. Forward and backward private searchable encryption with SGX. In: Proc. of the 12th European Workshop on Systems Security. Dresden: ACM, 2019. 4. [doi: 10.1145/3301417.3312496]

[17] Jiang Q, Qi Y, Qi SY, Zhao WJ, Lu YS. Pbsx: A practical private boolean search using Intel SGX. Information Sciences, 2020, 521: 174–194. [doi: 10.1016/j.ins.2020.02.031]

[18] Chai Q, Gong G. Verifiable symmetric searchable encryption for semi-honest-but-curious cloud servers. In: Proc. of the 2012 IEEE Int'l Conf. on Communications. Ottawa: IEEE, 2012. 917–922. [doi: 10.1109/ICC.2012.6364125]

[19] Kurosawa K, Ohtaki Y. UC-secure searchable symmetric encryption. In: Proc. of the 16th Int'l Conf. on Financial Cryptography and Data Security. Kralendijk: Springer, 2012. 285–298. [doi: 10.1007/978-3-642-32946-3_21]

[20] Wang JF, Chen XF, Sun SF, Liu JK, Au MH, Zhan ZH. Towards efficient verifiable conjunctive keyword search for large encrypted database. In: Proc. of the 23rd European Symp. on Research in Computer Security. Barcelona: Springer, 2018. 83–100. [doi: 10.1007/978-3-319-98989-1_5]

[21] Sun WH, Liu XF, Lou WJ, Hou YT, Li H. Catch you if you lie to me: Efficient verifiable conjunctive keyword search over large dynamic encrypted cloud data. In: Proc. of the 2015 IEEE Conf. on Computer Communications. Hong Kong: IEEE, 2015. 2110–2118. [doi: 10.1109/INFOCOM.2015.7218596]

[22] Zhu XY, Liu Q, Wang GJ. A novel verifiable and dynamic fuzzy keyword search scheme over encrypted data in cloud computing. In: Proc. of the 2016 IEEE Trustcom/BigDataSE/ISPA. Tianjin: IEEE, 2016. 845–851. [doi: 10.1109/TrustCom.2016.0147]

[23] Liu Q, Nie XH, Liu XH, Peng T, Wu J. Verifiable ranked search over dynamic encrypted data in cloud computing. In: Proc. of the 25th IEEE/ACM Int'l Symp. on Quality of Service. Vilanova i la Geltrú: IEEE, 2017. 1–6. [doi: 10.1109/IWQoS.2017.7969156]

[24] Zhang ZJ, Wang JF, Wang YL, Su YP, Chen XF. Towards efficient verifiable forward secure searchable symmetric encryption. In: Proc. of the 24th European Symp. on Research in Computer Security. Luxembourg: Springer, 2019. 304–321. [doi: 10.1007/978-3-030-29962-0_15]

[25] Ge XR, Yu J, Zhang HL, Hu CY, Li ZP, Qin Z, Hao R. Towards achieving keyword search over dynamic encrypted cloud data with symmetric-key based verification. IEEE Trans. on Dependable and Secure Computing, 2021, 18(1): 490–504. [doi: 10.1109/TDSC.2019.2896258]

[26] Hu SS, Cai CJ, Wang Q, Wang C, Luo XY, Ren K. Searching an encrypted cloud meets blockchain: A decentralized, reliable and fair realization. In: Proc. of the 2018 IEEE Conf. on Computer Communications. Honolulu: IEEE, 2018. 792–800. [doi: 10.1109/INFOCOM.2018.8485890]

[27] Cai CJ, Weng J, Yuan XL, Wang C. Enabling reliable keyword search in encrypted decentralized storage with fairness. IEEE Trans. on Dependable and Secure Computing, 2021, 18(1): 131–144. [doi: 10.1109/TDSC.2018.2877332]

[28] Li HG, Tian HB, Zhang FG, He JJ. Blockchain-based searchable symmetric encryption scheme. Computers & Electrical Engineering, 2019, 73: 32–45. [doi: 10.1016/j.compeleceng.2018.10.015]

[29] Li H, Zhang C, Huang HJ, Guo Y. Algorithm for encrypted search with forward secure updates and verification. Journal of Xidian University, 2020, 47(5): 48–56 (in Chinese with English abstract). [doi: 10.19665/j.issn1001-2400.2020.05.007]

[30] Yan XX, Yuan XH, Tang YL, Chen YL. Verifiable attribute-based searchable encryption scheme based on blockchain. Journal on Communications, 2020, 41(2): 187–198 (in Chinese with English abstract). [doi: 10.11959/j.issn.1000-436x.2020011]

附中文参考文献:

[29] 李涵, 张晨, 黄荷姣, 郭宇. 一种支持前向安全更新和验证的加密搜索算法. 西安电子科技大学学报, 2020, 47(5): 48–56. [doi: 10.19665/j.issn1001-2400.2020.05.007]

[30] 闫玺玺, 原笑含, 汤永利, 陈艳丽. 基于区块链且支持验证的属性基搜索加密方案. 通信学报, 2020, 41(2): 187–198. [doi: 10.11959/j.issn.1000-436x.2020011]

徐万山(1988－), 男, 博士生, 主要研究领域为信息安全, 可信计算, 密文检索, 区块链.

袁艺林(1991－), 女, 博士生, 主要研究领域为信息安全, 云计算, 云存储.

张建标(1969－), 男, 博士, 教授, 博士生导师, 主要研究领域为可信计算, 网络安全, 区块链.