一个具有前向安全和后向安全的可验证多关键字可搜索加密方案

宋翔飞 王化群

（南京邮电大学计算机学院 南京 210046）

摘 要 随着现代社会软硬件技术的快速发展，云计算技术对于客户端的数据存储和计算提供了巨大的帮助，为用户节省了大量成本.而在存储的同时，用户可以通过可搜索加密（SE，Searchable Encryption）技术，对存储数据进行加密搜索，同时确保数据的安全性和搜索隐私性.本方案使用公钥加密实现了数据拥有者和数据使用者之间的多关键字可搜索加密，数据使用者搜索时以出现频率最少的关键字作为主关键字进行加密搜索，并且云服务器对每个索引密文只需要一次计算即可得到准确的多关键字搜索结果，最大程度降低了无关文件的访问，节约了大量时间成本.而在存储和搜索过程中，我们认为云服务器是诚实且好奇的系统，它会诚实地为用户存储数据，并正确执行存储和计算过程，但是其会对存储的数据产生好奇感，即窥探用户的数据，并且对每次的搜索结果很感兴趣.为减少多关键字可搜索加密方案的信息泄露，提高安全性，提出的方案具有前向安全和后向安全的特性，在动态更新时为每次更新的状态创建一种隐式结构，使得云服务器只需要保存最新一次的更新状态就能保证对所有数据进行搜索，并且每个更新状态只保存前一次更新状态的信息，实现了前向安全性；通过将每次对文件的更新操作以密文方式存储，使服务器无法分辨插入和删除的文件，确保了方案的后向安全性.在可搜索加密过程中，云服务器可能会因为需要减少算力和带宽消耗而返回不完整的密文，所以需要对云服务器返回密文的完整性进行验证，相对于传统方案中使用第三方可信机构进行密文完整性的验证，本方案采用区块链中的智能合约进行验证.当搜索结束时为了确保密文的完整性，要求云服务器将密文及认证签名发送至智能合约触发验证算法进行完整性验证.根据安全性分析，本文方案可以抵抗关键字猜测攻击，在搜索过程中的信息泄露方面实现了前向安全，并且比同类方案增加了搜索模式的安全以及更高类型的后向安全，减少了搜索时向服务器的信息泄露，安全性更符合现代可搜索加密要求.效率方面，本文方案在陷门生成阶段和搜索阶段相较于同类方案减少大量计算，效率提高显著，最后使用5000条数据进行实验分析，根据实验表明，本方案的可搜索加密方案比同类方案更加高效和实用.

关键词 可搜索加密；智能合约；前向安全；后向安全；多关键字；可验证

中图法分类号 TP309 **DOI号** 10.11897/SP.J.1016.2023.00727

Verifiable Multi-Keyword Searchable Encryption with Forward and Backward Security

SONG Xiang-Fei WANG Hua-Qun

（*School of Computer*, *Nanjing University of Posts and Telecommunications*, *Nanjing* 210046）

Abstract With the rapid development of software and hardware technology in modern society, cloud computing technology provides great help for data storage and calculation on the client side, and saves a lot of costs for users. At the same time, users can use Searchable Encryption (SE) technology to encrypt and search the stored data, while ensuring data security and search privacy. This scheme uses public key encryption to realize multi-keyword searchable encryption between data owner and data user. When data user searches, the keyword with the least frequency is used

收稿日期：2021-12-06；在线发布日期：2022-11-04. 本课题得到国家自然科学基金（62272238）的资助.**宋翔飞**，硕士研究生，主要研究方向为可搜索加密. E-mail：songxiangfei0801@163.com. **王化群**（通信作者），博士，教授，中国计算机学会（CCF）会员，主要研究方向为应用密码学、区块链技术、未来移动通信安全等. E-mail：whq@njupt.edu.cn.

as the main keyword to encrypt and search, and the cloud server only needs to calculate each index ciphertext once to get accurate multi-keyword search results, which minimizes the access to irrelevant files. It saves a lot of time cost. In the process of storage and search, we think that the cloud server is an honest and curious system. It will store data for us honestly and perform the storage and calculation process correctly, but it will be curious about the stored data, that is, snoop on our data and be interested in our search results each time. In order to reduce the information leakage of the multi-keyword searchable encryption scheme and improve the security, the proposed scheme has the characteristics of forward security and backward security, and creates an implicit structure for each updated state during the dynamic update, so that the cloud server only needs to save the latest updated state to ensure that all the data can be searched. And each updated state only saved the information of the previous updated state, which achieved forward security. By storing each update operation to the file in the form of ciphertext, the server can not distinguish between inserted and deleted files, which ensures the backward security of the scheme. In the process of searchable encryption, the cloud server may return incomplete ciphertext due to the need to reduce computing power and bandwidth consumption. At this time, it is necessary to verify the integrity of the ciphertext returned by the cloud server. At the end of the search, in order to ensure the integrity of the ciphertext, the cloud server is required to send the ciphertext and the authentication signature to the smart contract to trigger the verification algorithm for integrity verification. According to the security analysis, the proposed scheme can resist the keyword guessing attack and achieve forward security in terms of information leakage during the search process. Compared with the similar schemes, the proposed scheme increases the security of search mode and a higher type of backward security, reduces the information leakage to the server during the search, and the security is more in line with the requirements of modern searchable encryption. In terms of efficiency, this scheme reduces a lot of calculations in the trapdoor generation stage and search stage compared with similar schemes, and the efficiency is significantly improved. Finally, 5000 data are used for experimental analysis. According to the experiments, the searchable encryption scheme of this scheme is more efficient and practical than similar schemes.

Keywords searchable encryption; smart contracts; forward privacy; fackward privacy; multi-keyword; verifiable

1 引 言

随着云计算和云存储技术日益成熟的发展,大部分企业和个人用户更愿意将私有数据存储在云服务器中,并使用云服务完成大量计算.这种方式极大地便利了人们的生活,如节省经济成本、灵活性高、部署速度快、高效的计算能力、弹性存储的带宽资源,以及随需应变的高质量服务.但是享受便利的同时,数据隐私安全问题也得到了广泛的关注[1].由于存储的数据中含有很多敏感数据,如姓名、身份证号、电话等信息,导致数据一旦泄露就会造成严重的后果.于是人们开始考虑将隐私数据以密文方式存储[2],这样可以有效地保护存储在云服务器上的数据.然而,如何搜索加密数据又成了一个难题,于是可搜索加密技术被提出.

2000年,Song等人首次提出了对称可搜索加密方案[3](SSE,Searchable Symmetric Encryption),将文件分成若干关键字,使用对称密钥和伪随机函数为每一个关键字生成加密密钥,并使用给定加密算法将每一个关键字加密再与流密码产生的随机值/伪随机置换对进行异或产生密文上传到服务器.搜索时只需要给定关键字的密文和加密的伪随机置换密钥即可,该方案保证了搜索时服务器得不到任何搜索关键字的信息,但是该方案的搜索效率比较低,必须遍历每个文件的每一个关键字.后来人们就开

始对SSE方案的高效性开始了研究[4-6]，2006年Curtmola等人提出了一种更高效的SSE方案[4]，该方案搜索时的计算量都是固定的，不需要将所有文件中的关键字进行一一计算，该方案同时实现了多用户搜索；2014年Cash等人提出了一种面向大型数据库的可搜索加密方案[5]，使用字典数据结构加标签的方式实现了对具有百亿记录/关键字对的数据集进行加密和搜索，并进行扩展实现了动态可搜索加密方案(DSSE，Dynamic Searchable Symmetric Encryption)，但是该方案是通过增加信息泄露作为前提的，并且该方案并不适合纯动态索引(即初始为空的索引)；2017年Miers等人提出了一种不经意更新索引方案[6](OUI，Obliviously Updatable Index)，该方案大幅度降低了I/O成本，但是不可避免地增加了信息的泄露.

对称可搜索加密虽然高效，但是只适合数据拥有者(DO，Data Owner)本身进行存和取的操作，对于数据拥有者和数据使用者(DU，Data User)场景却需要交互获得密钥. 于是，2004年Boneh等人提出了公钥可搜索加密[7](PEKS，Public-key Encryption with Keyword Search)，数据拥有者可以使用数据使用者公钥对数据进行加密上传至服务器，数据使用者使用自己的私钥生成关键字陷门对密文进行搜索. 但是其方案只能对单关键字进行搜索，在搜索时可能会返回大量不需要的文件，为了提高效率，可搜索加密应该支持多关键字搜索.

2004年，Golle等人首次提出了根据连接关键字进行可搜索加密的方案[8]，该方案只能对连接关键字进行搜索，做不到对非连接关键字进行可搜索加密，而且其搜索时通信开销较高，实用性较低. 后来，Cash等人设计了一种同时支持连接关键字和非连接关键字的方案[5]. Zhang等人设计了一种基于公钥的多关键字可搜索加密方案[9]，但是该方案在搜索时，服务器必须进行$n \times m$次双线性对运算，该方案的计算消耗是非常昂贵的. Zhang等人设计了基于困难性假设的多关键字可搜索加密方案[10]，且具有完整性验证，能够防止半诚实服务器为减少计算开销而返回不完整的密文.

以上方案实现了多关键字可搜索加密，但是都没有同时满足前向安全和后向安全属性. 为了达到对密文搜索的目的，往往会向服务器泄露一些信息，比如哪些搜索涉及同一个关键字的搜索模式以及为搜索返回了哪些文件的搜索模式，前向安全能够确保在服务器不能利用已有的搜索陷门的前提下，推测出新插入的文件的任何信息，后向安全确保服务器无法访问已经被删除的文件. 2016年，Bost等人提出了一种高效的前向安全可搜索加密方案(Sophos)[11]，其中客户端为每个关键字保留一个状态，并在更新服务器上包含该关键字的文件时更改该状态，当为更新的文件生成加密索引时，状态被用作输入，这意味着以前的陷门已经过时，不能用于匹配新的索引. Sophos使用陷门置换来更改状态，使得不知道私钥的服务器无法预测未来的状态，当发生新的更新时，服务器没有获得任何信息，从而实现了前向安全.

2018年Song等人引入了两种更高效的前向安全可搜索加密方案FAST和FASTIO[12]. FAST使用单链表这种数据结构，每次更新就像单链表中的一个结点，包含一个随机生成的临时密钥，通过临时密钥加密的方式来生成新的状态，给定当前密钥和当前状态，服务器可以通过解密计算之前的状态. 解密状态将帮助服务器获得之前的更新和之前的密钥，而由于之后更新的密钥都是由客户随机生成的，所以服务器是无法获得的，从而实现了前向安全. FASTIO是在FAST的基础上让服务器存储搜索结果实现的.

以上方案只实现了前向安全，但是都忽略了后向安全的属性.

2017年Bost等人引入了后向安全的正式定义[13]，并提出了多个有效的后向安全可搜索加密方案，其中包括第一个非交互的后向安全可搜索加密方案Janus. Janus中数据索引通过穿刺加密进行加密，对于被穿刺的密钥，服务器只能解密和检索匹配的未删除索引. 然而该方案中的穿刺加密部署起来是十分困难的，所以文献[13]中的方案都是具有高通信和高计算消耗的.

2018年Sun等人利用对称穿刺加密的变体提出了一种实用的非交互后向安全方案[14]. 该方案使用简单的加密方式代替文献[13]中的穿刺加密方案，形式化地定义了对称穿刺加密(SPE，Symmetric Puncturable Encryption)，并提出了SPE的一种变体增强SPE，使其附带一个属性，能够在客户进行删除操作时，将部分穿刺的密钥外包给服务器，从而在本地存储效率较低且不变的情况下逐步删除数据.

在可搜索加密中，往往会出现第三方可信中心来验证完整性[10][15-16]，但近些年区块链技术得到了发展，人们可以在区块链上部署智能合约进行计算，并且其不可篡改性得到了人们的青睐，将完整性验

证的过程转移到了智能合约上.Li等人将可搜索加密技术与区块链相结合[17-18],但是这些方案都是将大量数据存储在区块链上,如方案[17]使用区块链代替服务器,直接将加密数据存储在区块链上;方案[18]利用区块链完成加密数据的完整性证明,却导致每次搜索都需要在区块链上进行多次交易,而交易所需的大量数据都需要存储在区块链中,这在现实中还是有困难的,但是使用区块链中的智能合约代替第三方机构设计简单的完整性验证方案却是可行的[19-20].

以上方案均未同时满足前向安全、后向安全以及完整性验证等属性,或者不符合多关键字搜索的功能,而本方案在满足可搜索加密基础安全性之上实现了多关键字搜索,并且具有前向安全、后向安全和完整性验证的属性,具体比较如表1所示.

表1 相关文献对比表

方案	多关键字搜索	前向安全	后向安全	完整性验证
方案[6]	×	×	×	×
方案[10]	√	×	×	√
方案[12]	×	√	×	×
方案[14]	×	×	√	×
方案[18]	×	×	×	√
本文方案	√	√	√	√

本文在方案[10]的基础上设计了具有前向安全和后向安全的多关键字可搜索加密方案,同时满足搜索模式的隐私保护、完整性验证,主要贡献如下:

(1) 多关键字搜索:本方案提出了一种多关键字可搜索加密方案,该方案对文件中的所有关键字进行计算并生成该关键字的多关键字验证标识,当授权用户需要进行搜索时使用自己的私钥生成多关键字搜索陷门,云服务器匹配文件时对每个文件只需要一次计算即可确定该文件是否符合搜索条件,大大减少了多关键字搜索时的计算次数.

(2) 前向安全性:本方案使用隐式结构实现了前向安全,使用安全的哈希函数将关键字的索引更新分解为验证密文、更新状态密文和文件索引密文三部分,每次更新生成的更新状态密文都可以隐式地计算出之前的更新状态,使得更新算法不会泄露任何关于关键字的信息,确保服务器根据已有搜索陷门无法推测出新插入的文件信息.

(3) 后向安全性:本方案的可搜索加密方案在动态更新文件时将对文件的更新操作op(插入或删除)与关键字一起以密文形式保存,因此服务器无法分辨文件进行的更新操作,使得更新算法不会泄露插入之后又删除的文件信息,实现了多关键字可搜索加密的后向安全.

(4) 完整性验证:为防止半诚实的服务器为减少带宽消耗而返回不完整的密文数据,本方案考虑在不借助第三方可信机构的情况下进行完整性验证,最终选择使用区块链的智能合约完成该操作,其公开性和不可篡改性可以保证验证结果的可信度.

(5) 可搜索加密的安全性和可行性:本方案形式化地分析了该方案的安全性,证明其在标准模型下抗关键字猜测攻击(KGA, Keyword Guessing Attack).最后对本文方案以及其他方案进行实现,并通过大量实验数据的验证证明本方案能够在保护隐私安全的前提下改善搜索效率,实现多关键字的搜索要求.

2 预备知识

2.1 符号说明

方案中符号如表2所示.

2.2 可忽略函数

设一个函数$\mu:\mathbb{N}\to\mathbb{R}$,对于任何多项式$poly(\cdot)$,存在$N$,当$n>N$时,下式成立:

$$\mu(n)<\frac{1}{poly(n)}$$

则称函数μ是可忽略的.

2.3 伪随机置换函数和双线性映射

伪随机置换函数:其产生的置换结果与真正随机生成的序列是无法区分的.伪随机置换函数$F:\{0,1\}^L\times\{0,1\}^\lambda\to\{0,1\}^L$满足以下特点:

(1) 任意给定$K\leftarrow\{0,1\}^\lambda$,$F$是$\{0,1\}^L\to\{0,1\}^L$的一个双射.

(2)对于所有概率多项式敌手$\mathcal{A}$,$\left|\Pr\left[\mathcal{A}^{F_K}(1^L)=1\right]-\Pr\left[\mathcal{A}^{f}(1^L)=1\right]\right|<\varepsilon$,其中$K\leftarrow\{0,1\}^\lambda$,$f$是一个$L$比特串上的随机置换,$\varepsilon$是可忽略的概率.

(3) 给定$K\leftarrow\{0,1\}^\lambda$,$x\leftarrow\{0,1\}^L$,存在有效的算法计算$F_K(x)$.

除此之外,逆置换$F^{-1}:\{0,1\}^L\times\{0,1\}^\lambda\to\{0,1\}^L$是伪随机置换$F$的逆运算,即如果$F_K(x)=y$,则$F_K^{-1}(y)=x$.

双线性映射:设置两个阶为素数p的乘法循环

表2　符号说明

符号	含义
λ	安全参数
(G_1,G_2,e,p,g_1)	双线性对参数
$(H,H_{0,1,2,3},h_{(1,2,3,4)})$	安全哈希函数
r_0	多关键字验证使用的随机数
a_i	多关键字验证的随机点
(PK_{do},SK_{do})	数据拥有者公私钥对
(PK_{du},SK_{du})	数据使用者公私钥对
(PK_c,SK_c)	云服务器公私钥对
$Para$	公开参数
DB	文件-关键字数据集
Σ	数据拥有者维护的键值映射
W	关键字集和
l	关键字总个数
C	密文数据
Ind	索引集
Sig	文件签名集
(e_i,s_i)	第i个文件的签名
CT	密文/签名集
EV	更新版本号
$st_c^{w_i}$	w_i第c次更新的状态
(v,T,π_0,π_1,π_2)	多关键字验证参数
$I^{ind_j^{w_i}}$	文件多关键字加密标识
$(add_{st_{c+1}^{w_i}},val_{st_{c+1}^{w_i}})$	w_i第$c+1$次更新的状态密文
m_{w_i}	包含w_i的文件个数
$ind_j^{w_i}$	包含w_i的第j个文件的索引
$EI_{ind_j^{w_i}}$	文件$ind_j^{w_i}$的加密索引
$(add_{ind_j^{w_i}},val_{ind_j^{w_i}})$	文件$ind_j^{w_i}$的索引密文
$(add_{t_{w_i}},val_{t_{w_i}})$	关键字w_i的验证密文
EDB	加密数据库
$T_{W'}$	搜索陷门
MEI	匹配的索引密文集
θ	数据拥有者和使用者共同维护的随机数
op	更新操作add/del

群G_1和G_2，g_1是G_1的一个生成元. 存在一个双线性映射e: $G_1\times G_1\rightarrow G_2$，且其满足以下性质：

1) 双线性：对于任意$u,v\in G_1$，任意$a,b\in Z_p$都有$e(u^a,v^b)=e(u,v)^{ab}$.

2) 非退化性：存在$u,v\in G_1$满足$e(u,v)\neq 1$.

3) 可计算性：对于任意$u,v\in G_1$，$e(u,v)$可有效计算.

2.4　困难假设

*DBDH*假设：根据系统安全参数选择阶为素数p的乘法循环群G_1和G_2，g_1是G_1的生成元，给定两个元组$(g_1,g_1^a,g_1^b,g_1^c,e(g_1,g_1)^{abc})$和$(g_1,g_1^a,g_1^b,g_1^c,Z)$，其中$a,b,c\in Z_p$，$Z\in G_2$，算法$\mathcal{A}$判断$Z=e(g_1,g_1)^{abc}$是否成立，若成立输出1，否则输出0. 如果对于任意多项式时间算法$\mathcal{A}$，区分上述两个元组的优势：

$$Adv^{DBDH}=\left|\Pr\left[\mathcal{A}\left(g_1,g_1^a,g_1^b,g_1^c,e(g_1,g_1)^{abc}\right)\right]\right|-\left|\Pr\left[\mathcal{A}(g_1,g_1^a,g_1^b,g_1^c,Z)\right]\right|$$

是计算可忽略的，则*DBDH*假设在群G_1，G_2中是成立的.

$q-ABDHE$假设：根据系统安全参数选择阶为素数p的乘法循环群G_1和G_2，g_1,g_1'是G_1的生成元. 给定两个$q+4$元组$\left(g_1',g_1'^{q+2},g_1,g_1^{\alpha},\cdots,g_1^{\alpha^q},e(g_1,g_1')^{\alpha^{q+1}}\right)$和$\left(g_1',g_1'^{q+2},g_1,g_1^{\alpha},\cdots,g_1^{\alpha^q},Z\right)$，其中$Z\in G_2$，算法$\mathcal{A}$判断$Z=e(g_1,g_1')^{\alpha^{q+1}}$是否成立，若成立，输出1，否则输出0. 如果对于任意多项式时间算法$\mathcal{A}$，区分上述两个元组的优势：

$$Adv^{q-ABDHE}=\left|\Pr\left[\mathcal{A}\left(g_1',g_1'^{q+2},g_1,g_1^{\alpha},\cdots,g_1^{\alpha^q},e(g_1,g_1')^{\alpha^{q+1}}\right)\right]\right|-\left|\Pr\left[\mathcal{A}\left(g_1',g_1'^{q+2},g_1,g_1^{\alpha},\cdots,g_1^{\alpha^q},Z\right)\right]\right|$$

是计算可忽略的，则$q-ABDHE$假设在群G_1，G_2中是成立的.

2.5　前向安全

前向安全表示服务器不能利用已有的陷门来搜索新更新的文件，换句话说，更新算法不应该泄露关于关键字的有用信息. 前向安全的形式化定义如下：

定义1[21]. 若一个方案具有前向安全性，则方案中更新操作的泄露函数$\mathcal{L}_{\text{update}}$满足：

$$\mathcal{L}_{\text{update}}(w_i)=\mathcal{L}'(\text{op},\text{ind}_j)$$

其中$\mathcal{L}'$为无状态泄露函数.

2.6　后向安全

后向安全是指服务器不能访问已经被删除的文件，同时文献[13]正式定义了三种强度的后向安全：

Ⅰ型　插入模式的后向安全：

泄露与w匹配的文件和插入时间，以及总的更新次数；

Ⅱ型　更新模式的后向安全：

泄露与w匹配的文件和插入时间，以及w所有的更新发生时间；

Ⅲ型　弱后向安全：

泄露与w匹配的文件和插入时间，以及w所有的更新发生时间，和某个删除更新取消了某个插入更新形式化定义如下：

定义2[13]. 若一个方案具有后向安全性，则方案中更新操作的泄露函数$\mathcal{L}_{\text{update}}$和搜索操作的泄露函数$\mathcal{L}_{\text{search}}$需要满足：

Ⅰ型：

$$\mathcal{L}_{update}(op, w, id) = \mathcal{L}'(\text{op})$$

$$\mathcal{L}_{\text{search}}(w) = \mathcal{L}''(\text{TimeDB(w)}, N)$$

Ⅱ型：

$$\mathcal{L}_{\text{update}}(op, w, id) = \mathcal{L}'(\text{op}, w)$$

$$\mathcal{L}_{\text{search}}(w) = \mathcal{L}''(\text{TimeDB(w)}, Update(w))$$

Ⅲ型：

$$\mathcal{L}_{\text{update}}(op, w, id) = \mathcal{L}'(\text{op}, w)$$

$$\mathcal{L}_{\text{search}}(w) = \mathcal{L}''(\text{TimeDB}(w), DelHist(w))$$

其中$\mathcal{L}'$, $\mathcal{L}''$为无状态泄露函数，$TimeDB()$为表示所有当前在数据集中的含有关键字w的文档及其插入时间，$Update(w)$表示所有与w有关的更新操作的时间戳，$DelHist(w)$表示关于w的插入/删除对.

2.7 搜索模式安全

设$QList$为查询列表，列表元素形式为(t, w_i)，表示在时间t查询了关键词w_i，则关键词w_i的搜索模式为：

$$sp(w_i) = \{t | (t, w_i) \in QList\}$$

定义3[22]. 若一个可搜索加密方案具有搜索模式的隐私安全，则在搜索阶段不会泄露任何搜索模式信息.

2.8 智能合约

智能合约是一些部署在区块链上特殊的脚本程序，其可以自动执行，并且永远不能被修改，包括创建程序的人也不能对其进行修改. 理想情况下，智能合约支持完整脚本的特性可以完成多种复杂的函数. 已部署的智能合约被存储在区块链中的一个块上，其不依赖于任何权限执行. 所有存储在智能合约上的数据以及智能合约的执行都是公开透明的，因此，智能合约的正确性是可信赖的[20].

3 系统模型及设计目标

3.1 系统模型

在本文方案中存在四个实体，分别为：数据拥有者(DO)、数据使用者(DU)、云服务器(CS，Cloud Server)和区块链系统(BS，Blockchain System). 数据拥有者(DO)是数据的上传者，其将加密数据上传至云服务器，并对加密数据建立索引上传至云服务器等待数据使用者搜索使用；数据使用者(DU)搜索获取需要的数据，使用关键词生成搜索陷门发送至云服务器，利用云计算获得索引密文；云服务器(CS)存储数据，并提供一定的计算能力；区块链系统(BS)是由多个结点维护的区块链网络来支持智能合约的运行. 系统模型如图1所示：

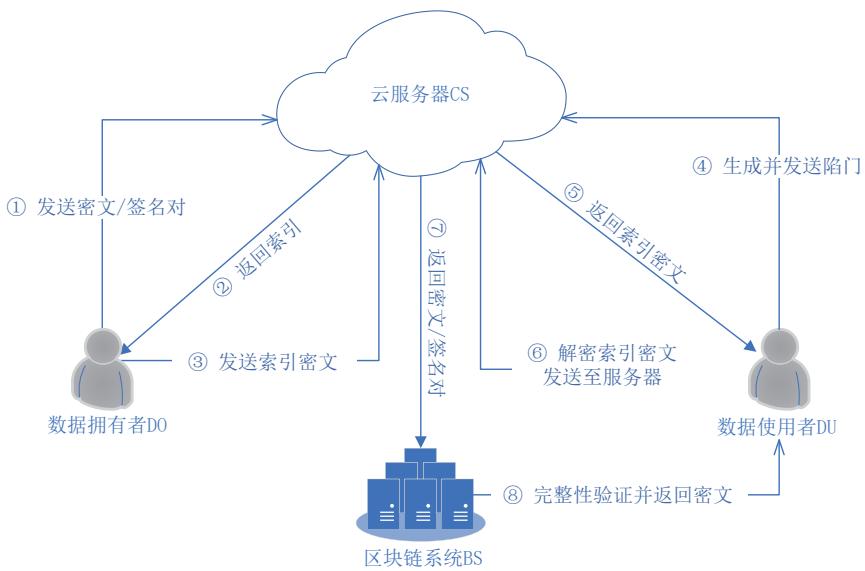

图1 系统模型

(1) 数据拥有者使用事先沟通好的对称加密(如：AES)密钥为数据生成密文，并为密文生成验证签名，将密文/签名对存入云服务器中.

(2) 云服务器收到密文/签名对并对其进行存

储，生成索引（如存储地址）返回给数据拥有者.

（3）数据拥有者收到云服务器生成的索引，使用数据使用者的公钥对索引进行加密，产生多关键字可搜索加密密文发送至云服务器.

（4）数据使用者使用自己的私钥生成多关键字搜索陷门并发送至云服务器，等待云服务器返回搜索结果.

（5）云服务器收到数据使用者的搜索陷门并触发搜索算法得到搜索结果返回给数据使用者.

（6）数据使用者对返回结果解密，得到搜索数据的索引并发送给云服务器.

（7）云服务器根据索引将数据密文/签名对发送至智能合约并触发验证算法.

（8）智能合约进行验证，若验证成功则将密文返回给数据使用者，否则返回验证失败.

本文多关键字可搜索加密方案包括以下算法：

$Setup(\lambda)\rightarrow\{Para\}$：这是一个确定性算法，输入一个安全参数$\lambda$，输出系统公开参数$Para$.

$KeyGen(Para)\rightarrow\{PK_c, SK_c, PK_{do}, SK_{do}, PK_{du}, SK_{du}, \theta\}$：该算法是一个概率算法，输入系统公开参数$Para$，生成数据拥有者公私钥对$(PK_{do}, SK_{do})$，以及数据使用者公私钥对$(PK_{du}, SK_{du})$，服务器公私钥对$(PK_c, SK_c)$，和一个随机值$\theta$.

$Update(Para, DB, SK_{do}, PK_{du}, PK_c, \Sigma, C, Ind)\rightarrow\{Sig, EDB\}$：输入公共参数Para，数据拥有者先根据密文和自己的私钥SK_{do}生成签名集Sig，再使用数据使用者公钥PK_{du}，服务器公钥PK_c，数据拥有者自己维护的一组映射Σ和服务器返回的索引集Ind生成索引密文键值集合EDB.

$Trapdoor(Para, PK_{du}, SK_{du}, W')\rightarrow\{T_{W'}\}$：该算法是一个概率算法，由数据使用者执行，输入系统参数、数据使用者公私钥对和关键字集$W'=\{w_1, w_2, \cdots, w_\lambda\}$，数据使用者通过该算法获得搜索陷门$T_{W'}$.

$Search(Para, T_{W'}, SK_c, PK_{du}, EDB)\rightarrow\{MEI\}$：该算法是一个概率算法，由云服务器执行，输入公共参数$Para$，搜索陷门$T_{W'}$，云服务器私钥SK_c，数据使用者公钥PK_{du}，以及索引密文键值集合EDB，云服务器生成匹配索引密文集MEI并发送给数据使用者.

$Decrypt(Para, SK_{du}, w_i, MEI)$：该算法由数据使用者执行，以公共参数$Para$，数据使用者私钥$SK_{du}$，以及主关键字$w_i$和加密索引集$MEI$作为输入，数据使用者获得搜索结果的索引明文并发送给云服务器.

$Verify(Para, C, PK_{do}, Sig)\rightarrow C/false$：该算法由智能合约执行，输入公共参数$Para$，密文/签名对$(C, Sig)$，以及数据使用者公钥$PK_{do}$，验证成功则将密文集$C$发送给数据使用者，否则验证失败，输出$false$.

3.2　设计目标

根据多关键字可搜索加密的高效性及安全性，本文设计目标如下：

（1）多关键字搜索：本方案实现了使用多个关键字进行精确搜索的目的，并在文章最后验证了其安全性和抗关键字猜测攻击的安全性能.

（2）完整性验证：为了防止云服务器为减少计算量而返回不完整的密文数据，在得到数据之前对返回数据进行完整性验证.

（3）搜索模式的隐私安全：在云服务器得到搜索陷门后无法推测出任何有关搜索关键词的信息.

（4）前向安全：使得云服务器根据现有的搜索陷门无法推测出之后插入的数据信息.

（5）后向安全：减少更新数据时的信息泄露，使删除操作之后，云服务器无法获得任何有关删除数据的信息.

4　具体方案构造

4.1　初始化阶段

$Setup(\lambda)\rightarrow\{Para\}$：这是一个确定性算法，输入一个安全参数$\lambda$，输出系统公开参数$Para$. 输入安全参数$\lambda$，首先生成双线性对参数$(G_1, G_2, e, p, g_1)$，其中$G_1, G_2$都是阶为$p$的乘法循环群，$p$是一个大素数，$e: G_1\times G_1\rightarrow G_2$是一个双线性对映射. g_1是群G_1的随机生成元. 其次生成一些安全的哈希函数$(H, H_0, H_1, H_2, H_3, h_1, h_2, h_3, h_4)$和一个可逆伪随机置换$F$，其中：

$$H:\{0,1\}^*\rightarrow\{0,1\}^*, H_0:\{0,1\}^*\rightarrow G_1,$$
$$H_1: G_1\times\{0,1\}^*\rightarrow\{0,1\}^{\lambda+1},$$
$$H_2, H_3: G_2\rightarrow\{0,1\}^{2\lambda},$$
$$h_1:\{0,1\}^{\lambda}\rightarrow\{0,1\}^{2\lambda},$$
$$h_2:\{0,1\}^{\lambda}\rightarrow\{0,1\}^{\lambda+\log N_{max}},$$
$$h_3:\{0,1\}^{\lambda}\times\{0,1\}^{\log N_{max}}\rightarrow\{0,1\}^{2\lambda}$$
$$h_4:\{0,1\}^{\lambda}\times\{0,1\}^{\lambda+\log N_{max}}\rightarrow\{0,1\}^{\lambda+\log N_{max}+1},$$
$$F:\{0,1\}^{\lambda}\times\{0,1\}^{\lambda}\rightarrow\{0,1\}^{\lambda}$$

随机选择$r_0\in Z_p$,$a_i\in G_1(i=0,1,\cdots,l)$,$l$表示一个文件能够含有的最大关键字个数.

4.2 密钥生成阶段

$KeyGen(Para)\rightarrow\{PK_c,SK_c,PK_{do},SK_{do},PK_{du},SK_{du},\theta\}$:该算法是一个概率算法,输入系统公开参数$Para$,输出公私钥对.数据拥有者随机选择$x\in Z_p$作为其私钥$SK_{do}$,公钥为$PK_{do}=g_1^x$.数据使用者随机选择$d\in Z_p$作为其私钥$SK_{du}$,公钥$PK_{du}=g_1^d$.云服务器也随机选择一个随机数$c\in Z_p$作为其私钥$SK_c$,公钥$PK_c=g_1^c$.最后数据拥有者选择一个随机数$\theta\in Z_p$,并通过安全信道发送给数据使用者,两者将$\theta$秘密保存起来.

4.3 更新阶段

$Update(Para,DB,SK_{do},PK_{du},PK_c,\Sigma,C,Ind)\rightarrow\{Sig,EDB\}$:数据拥有者输入公共参数Para,使用数据使用者公钥PK_{du},服务器公钥PK_c,更新文件的密文集C,为每一个文件随机选择$k_i\in Z_p$,计算$e_i=H\left(g_1^{k_i},c_i\right)$,$s_i=SK_{do}*e_i+k_i$,令$sig_i=(e_i,s_i)$,将$CT=\left(sig_i,c_i\right)_{i=1,\cdots,N}$上传至云服务器,其中$N$为本次更新文件数量.接下来为索引创建密文:

(1)选择一个随机数$r\in Z_p$,并计算当前加密数据的版本信息$EV=g_1^r\in G_1$作为关键字的最新更新标志.

(2)对于每一个$w_i\in W$:

① 检索$(st_c^{w_i},c)\leftarrow\sum[w_i]$.如果$\left(st_c^{w_i},c\right)=\perp$,则设$st_c^{w_i}\leftarrow\{0,1\}^\lambda$,$c\leftarrow 0$,$k_{c+1}^{w_i}\leftarrow\{0,1\}^\lambda$并计算:

$$st_{c+1}^{w_i}=F\left(k_{c+1}^{w_i},st_c^{w_i}\right)$$

然后更新映射$\sum[w_i]=\left(st_{c+1}^{w_i},c+1\right)$;

② 数据拥有者随机选择$x_1,x_2\in Z_p$,计算:

$$v=e\left(PK_{du},\ g_1\right)^{x_1\cdot st_{c+1}^{w_i}}$$

$$T=PK_c^{x_1}$$

$$\pi_0=PK_{du}^{x_2},\ \pi_1=v\cdot e\left(g_1,a_0\right)^{x_2},\ \pi_2=e\left(g_1,g_1\right)^{x_2}$$

其中(T,π_0,π_1,π_2)在搜索阶段辅助判断多关键词的验证.

③ 设每个文件包含关键字数为l,对$ind_j^{w_i}$下的每一个关键词$w_s(s=1,\cdots,l)$:

$$I_s^{ind_j^{w_i}}=a_s^{w_s'x_2}$$

并计算:

$$I^{ind_j^{w_i}}=\prod_{s=1}^{l}I_s^{ind_j^{w_i}}$$

其中$w_s'=H(w_s\|\theta)$,$I^{ind_j^{w_i}}$表示文件$ind_j^{w_i}$的多关键字加密标识,搜索阶段配合(T,π_0,π_1,π_2)来验证文件是否符合多关键字搜索条件.

④ 计算

$$add_{st_{c+1}^{w_i}}=h_1\left(st_{c+1}^{w_i}\right)$$

$$val_{st_{c+1}^{w_i}}=[(m_{w_i}\|k_{c+1}^{w_i})\oplus h_2(st_{c+1}^{w_i})]\|(T,\pi_0,\pi_1,\pi_2)$$

其中$m_{w_i}=|DB(w_i)|$,至此生成了关键字w_i的第$c+1$次更新状态密文$(add_{st_{c+1}^{w_i}},val_{st_{c+1}^{w_i}})$,每次更新的所有文件都与该状态密文关联,只有计算出该状态密文才能搜索本次更新的所有文件索引密文.

⑤ 对于$DB(w_i)$下的每一个文件索引$ind_j^{w_i}$,计算

$$EI_{ind_j^{w_i}}=H_1(PK_{du}^r,w_i\|j)\oplus(op\|ind_j^{w_i})$$

$$add_{ind_j^{w_i}}=h_3\left(st_{c+1}^{w_i},j\right)$$

$$val_{ind_j^{w_i}}=[(j\|EI_{ind_j^{w_i}})\oplus h_4(st_{c+1}^{w_i},j)]\|I^{ind_j^{w_i}}$$

其中$j=[1,m_{w_i}]$,每一个文件索引密文都是一个$(add_{ind_j^{w_i}},val_{ind_j^{w_i}})$,数据使用者可以使用自己的私钥对其进行解密从而得到数据的索引.

⑥ 计算

$$t_{w_i}=e\left(H_0(w_i'),PK_{du}^r\right)=e\left(H_0(w_i'),g_1^{dr}\right)\in G_2$$

最后计算

$$add_{t_{w_i}}=H_2\left(t_{w_i}\right)$$

$$val_{t_{w_i}}=H_3\left(t_{w_i}\right)\oplus st_{c+1}^{w_i}$$

每一个关键字w_i只有一个验证密文$(add_{t_{w_i}},val_{t_{w_i}})$,搜索阶段只有通过该验证才能进行后续搜索.

⑦ 将加密数据库

$$EDB=\{(add_{t_{w_i}},val_{t_{w_i}}),(add_{st_{c+1}^{w_i}},val_{st_{c+1}^{w_i}}),(add_{ind_j^{w_i}},vla_{ind_j^{w_i}})\}$$

存入云服务器等待数据使用者搜索.

这种方式生成的索引密文能够保证所有状态之间具有一种隐式结构.如图2所示,将文件集中的关键字分解,为每一个关键字w_i创建索引链表.如图3所示,关键词索引链表包括三部分,分别是关键字w_i的验证密文$(add_{t_{w_i}},val_{t_{w_i}})$,更新文件集时的更新状态密文$(add_{st_c^{w_i}},val_{st_c^{w_i}})$和文件索引密文$(add_{ind_j}^{w_i},val_{ind_j}^{w_i})$,密文之间具有隐式结构的关系,其中使用验证密文$(add_{t_{w_i}},val_{t_{w_i}})$进行$w_i$存在性的验证,在$val_{t_{w_i}}$中隐式存储$w_i$的最新更新状态$st_c^{w_i}$,更新状态密文中的$val_{st_c}^{w_i}$可以计算出所有状态密文,并且其又隐式地存储了计算前一个状态$st_{c-1}^{w_i}$的关键参数$k_c^{w_i}$,使得最终可以通过隐式结构计算出所有更新状态下的文件索引密文并进行多关键字匹配.

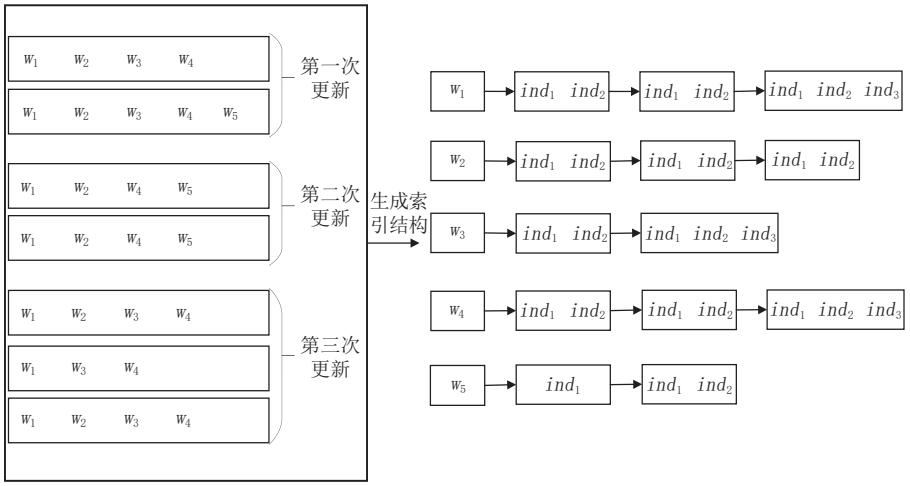

图2　文件索引结构

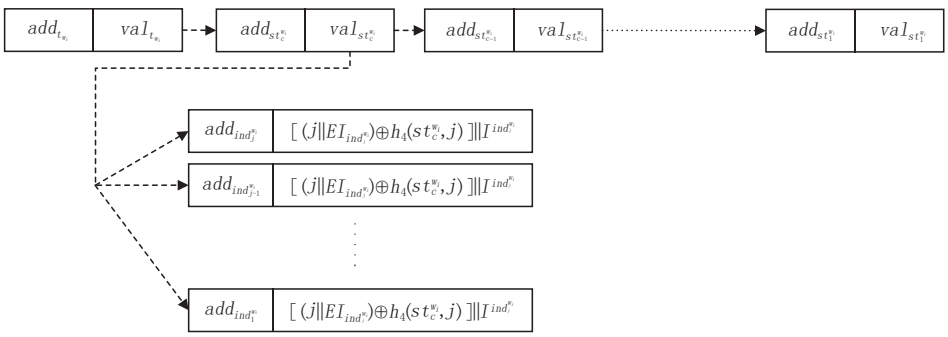

图3　索引的隐式结构

4.4　陷门生成阶段

$Trapdoor(Para, PK_{du}, SK_{du}, W')\rightarrow\{T_{W'}\}$：该算法是一个概率算法，数据使用者输入系统参数、数据使用者公私钥对和关键字集 $W'=\{w_1, w_2, \cdots, w_\lambda\}$，输出搜索陷门 $T_{W'}$.

(1) 首先从版本信息中获得最新版本号 $EV=g^r$，并选择一个关键字 $w_i\in W'$ 作为主关键字，计算

$$T_{w_i}=e\left(H_0(w_i')^{SK_{du}}, EV\right)=e\left(H_0(w_i')^d, g_1^r\right)$$
$$=e\left(H_0(w_i'), g_1^{dr}\right)$$

T_{w_i} 用于搜索阶段计算正确的 w_i 验证密文 $(add_{t_{w_i}}, val_{t_{w_i}})$.

(2) 随机选择 $\gamma\in Z_p$ 计算：

$$d_0=\left(a_0 g_1^{-r_0}\right)^{\frac{1}{d}}\left(\prod_{k=1}^{\lambda}a_k^{w_k'}\right)^{\gamma}$$
$$d_1=PK_{du}^{\gamma}$$

其中 $w_k'=H(w_k\|\theta)$，以 $T_{W'}=(T_{w_i}, d_0, d_1)$ 作为关键字集 W' 的搜索陷门发送给云服务器，其中 (d_0, d_1) 用于判断 $ind_j^{w_i}$ 表示的文件是否符合多关键集 W'.

4.5　搜索阶段

$Search(Para, T_{W'}, SK_c, PK_{du}, EDB)\rightarrow\{MEI\}$：

(1) 初始化一个索引集和 $MEI(W')\leftarrow\varnothing$，并计算

$$add_{t_{w_i}}^*=H_2(T_{w_i})=H_2\left(e\left(H_0(w_i'), g_1^{dr}\right)\right)=H_2(t_{w_i})$$

(2) 若陷门 T_{w_i} 不正确，则 $add_{t_{w_i}}^*$ 检索不到，停止搜索并返回 $MEI(w_i)\leftarrow\varnothing$. 否则，检索得到

$$val_{t_{w_i}}^*=EDB(add_{t_{w_i}}^*)$$

并计算

$$st_c^{w_i}=val_{t_{w_i}}^*\oplus H_3(T_{w_i})=H_3(T_{w_i})\oplus st_c^{w_i}\oplus H_3(T_{w_i})$$

(3) 计算 $add_{st_c^{w_i}}=h_1(st_c^{w_i})$，若计算得到的 $add_{st_c^{w_i}}$ 检索不到 $val_{st_c^{w_i}}$，即 $EDB\left[add_{st_c^{w_i}}\right]=\perp$，则表示 w_i 关键字链表的所有更新状态都已经搜索完毕，停止搜索并返回 $MEI(w_i)$. 否则检索得到

$$val_{st_c^{w_i}}=EDB(add_{st_c^{w_i}})$$

并计算

$$(m_{w_i}\|k_c^{w_i})=\left[(m_{w_i}\|k_c^{w_i}\oplus h_2(st_c^{w_i})\right]\oplus h_2(st_c^{w_i})$$

对于 $j\in[1, m_{w_i}]$ 计算：

$$add_{ind_j^{w_i}}=h_3(st_c^{w_i},j)$$

$$val_{ind_j^{w_i}}=EDB\left[add_{ind_j^{w_i}}\right]$$

$$=[(j\|EI_{ind_j^{w_i}})\oplus h_4(st_c^{w_i},j)]\|I^{ind_j^{w_i}}$$

得到一个文件的索引密文后，还要通过$val_{ind_j^{w_i}}$中的$I^{ind_j^{w_i}}$和(T,π_0,π_1,π_2)进行多关键字验证.

(4) 令$t=st_c^{w_i}$计算：

$$v'=e(T,PK_{du})^{SK_c^{-1}t}$$

取$I^{ind_j^{w_i}}$进行匹配，如果：

$$\pi_1\cdot\frac{e\left(d_1,I^{ind_j^{w_i}}\right)\cdot\pi_2^{-r_0}}{e(\pi_0,d_0)}=v'$$

则表示该索引密文中的加密索引符合多关键字搜索条件，计算

$$(j\|EI_{ind_j^{w_i}})=[(j\|EI_{ind_j^{w_i}})\oplus h_4(st_c^{w_i},j)]\oplus h_4(st_c^{w_i},j)$$

将$(j\|EI_{ind_j^{w_i}})$插入到索引集$MEI(W')$. 若不符合多关键字搜索条件，则令$j=j+1$循环执行第四步，计算下一个索引是否符合多关键字条件，直到$j>m_{w_i}$

(5) 计算状态

$$st_{c-1}^{w_i}=F^{-1}(k_c^{w_i},st_c^{w_i})$$

返回第三部继续执行.

最终在第三步得到MEI集合返回给数据使用者.

4.6 解密阶段

$Decrypt(Para,SK_{du},w_i,MEI)$：数据使用者以公共参数$Para$，数据使用者私钥$SK_{du}$，以及主关键字$w_i$和加密索引集$MEI$作为输入，输出搜索结果的索引明文. 对加密索引集中MEI中的每一项计算：

$$(op\|ind_j^{w_i})=H_1(EV^{SK_{du}},w_i\|j)\oplus EI_{ind_j^{w_i}}$$

如果$op=deletion$，则表示该索引是已经被删除的. 最后将索引提交给服务器，服务器返回相应密文/签名对CT给区块链.

4.7 验证阶段

$Verify(Para,C,PK_{do},Sig)\rightarrow C/false$：服务器以密文/签名对集合$(C,Sig)$作为参数，发送至智能合约，智能合约进行密文完整性验证：

(1) 通过密文集(C)和签名集(Sig)获取所有密文/签名对$(c_i,sig_i)=(c_i,(s_i,e_i))$

(2) 对每对密文/签名对计算：

$$r'=g_1^{s_i}PK_{do}^{-e_i}$$

(3) 验证：

$$H(r',c_i)=e_i$$

若所有密文/签名对都能够完成验证，则将密文/签名集(C,Sig)发送给数据使用者，否则返回$false$表示验证失败.

5 安全性证明

5.1 正确性证明

(1) 陷门匹配的正确性

以关键字W'为目标进行多关键字搜索时，以w_i为主关键字生成搜索陷门(T_{w_i},d_0,d_1)搜索，服务器接收到陷门后计算：

$$\begin{aligned}H_2(T_{w_i})&=H_2(e(H_0(w_i)^{SK_{du}},EV))\\&=H_2(e(H_0(w_i),EV^{SK_{du}}))\\&=H_2(e(H_0(w_i),EV^{d}))\\&=H_2(e(H_0(w_i),g_1^{dr}))\\&=H_2(e(H_0(w_i),PK_{du}^{r}))\\&=H_2(t_{w_i}^*)\end{aligned}$$

(2) 多关键字验证的正确性

若$H_2(t_{w_i}^*)$能够成功计算出$val_{ind_j^{w_i}}$，则可以对$ind_j^{w_i}$进行多关键字验证，计算

$$\pi_1\cdot\frac{e\left(d_1,I^{ind_j^{w_i}}\right)\cdot\pi_2^{-r_0}}{e(\pi_0,d_0)}$$

$$=v\cdot e(g_1,a_0)^{x_2}\cdot\frac{e\left(PK_D^{\gamma_1},\prod_{s=1}^{l}I_s^{ind_j^{w_i}}\right)\cdot e(g_1,g_1)^{-r_0x_2}}{e\left(\pi_0,(a_0g_1^{-r_0})^{1/d}\cdot\left(\prod_{k=1}^{\lambda}a_k^{\vec{w}'}\right)^{\gamma_1}\right)}$$

$$=v\cdot e(g_1,a_0)^{x_2}\cdot\frac{e(g_1,g_1)^{-r_0x_2}}{e\left(\pi_0,(a_0g_1^{-r_0})^{1/d}\right)}$$

$$=v=e(T,PK_{du})^{c^{-1}\cdot t}$$

根据计算过程可以发现，若$ind_j^{w_i}$的关键字符合搜索关键字集W'，则最终服务器可计算得到

$$v'=e(T,PK_{du})^{c^{-1}\cdot t}$$

然后计算$(j\|EI_{ind_j^{w_i}})$并入集合$MEI(W')$，接下来另$j=j+1$继续进行计算.

(3) 完整性验证的正确性

智能合约得到密文/签名集后对每一个$(c_i,sig_i)=(c_i,(s_i,e_i))$进行验证：

$$\begin{aligned}H(r',c_i)&=H(g_1^{s_i}PK_{do}^{-e_i},c_i)=H(g_1^{SK_{do}*e_i+k_i}g_1^{-xe_i},c_i)\\&=H(g_1^{xe_i+k_i-xe_i},c_i)=H(g_1^{k_i},c_i)=e_i\end{aligned}$$

若对所有密文/签名对都能够验证成功则证明密文是完整的.

5.2　安全性

(1) 保密性

保密性包括两部分，数据保密性和索引保密性.

数据保密性指在数据外包到云服务器之前使用安全对称加密算法(如AES)对其进行加密，而本文主要针对索引进行加解密搜索. 在向服务器CS存储并搜索内容时，一定会向服务器泄露一些信息，索引的保密性就是保证不会泄露不必要的信息. 为了证明保密性，我们使用理想-现实模拟环境. 把泄露的信息参数化为 $\mathcal{L}=(\mathcal{L}_{Setup}, \mathcal{L}_{KeyGen}, \mathcal{L}_{Update}, \mathcal{L}_{Trapdoor}, \mathcal{L}_{Search})$ 来描述方案泄露给对手的信息.

定义4.　设我们的方案为 $\Pi=(Setup, KeyGen, Update, Trapdoor, Seach)$，$S$ 是一个模拟器并且 A 是一个敌手. 定义两个游戏：

① $Real_A^{\Pi}(\lambda)$：这个游戏运行 $Setup(\lambda)$ 算法和 $KeyGen(Para, \lambda)$ 算法生成 $(Para, \Sigma, \mathrm{PK}_{du}, PK_c, SK_c)$，然后发布公共参数 $(Para, PK_{du}, PK_c)$ 并保证 SK_{du}, SK_c 的私密性. A 基于 $(Para, PK_{du}, PK_c)$ 对更新查询操作创建一个数据库 DB. 接着游戏运行 $Update(Para, PK_{du}, PK_c, DB, \Sigma)$ 并返回 EDB 给 A. A 选择一个关键词集 $W'=\{w_1, w_2, \cdots, w_\lambda\}$ 生成询问陷门，该游戏运行 $Trapdoor(Para, PK_{du}, SK_{du}, W')=T$ 算法并返回 MEI 给 A. 最终，A 输出一比特 $b\in\{0,1\}$.

② $Ideal_{A,S}^{\Pi}(\lambda)$：模拟器 S 使用泄露函数产生公共参数 $(Para, PK_{du}, PK_c, PK_{du})\leftarrow S(\mathcal{L}_{Setup}, \mathcal{L}_{KeyGen})$. 模拟器S对 A 的陷门询问创建一个加密数据库 $EDB\leftarrow S(\mathcal{L}_{Trapdoor})$ 作为回应. 然后模拟器对 A 的搜索询问创建 $MEI\leftarrow\mathcal{L}_{Search}$ 作为响应. 最终，敌手 A 输出一比特 $b\in\{0,1\}$.

定理.　如果 F 是一个安全的伪随机函数，哈希函数是抗碰撞的，并且 $DBDH$ 假设和 $q-ABDHE$ 假设成立，则说明该方案是一个 $\mathcal{L}$ -适应性安全的可搜索加密方案.

证明.　通过一系列游戏来证明，每一个游戏都与前一个略有不同，但是对于敌手都是不可区分的. 第一个游戏 G_1 是 $Real_A^{\Pi}(\lambda)$，最后一个游戏是 $Ideal_{A,S}^{\Pi}(\lambda)$. 我们利用不可区分性的传递性质来完成证明，证明过程如图4所示.

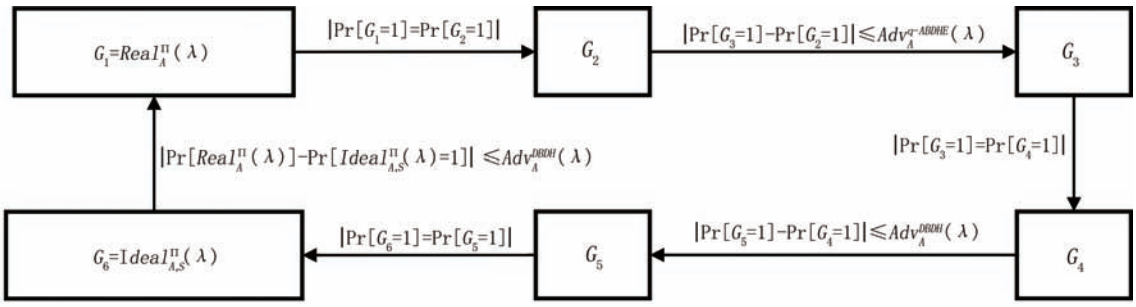

图4　保密性证明

在第二个游戏 G_2 中，我们维护一个内容为 $SList[w,c]=st_c$ 的状态列表 $SList$，用它代替 $st_c=F(k_c, st_{c-1})$. 当在更新阶段需要 st_c 时，该游戏随机选择 $st_c\leftarrow\{0,1\}^{\lambda}$. 很明显，当伪随机置换是安全的，游戏 G_1 和 G_2 是不可区分的.

$$|\Pr[G_1=1]=\Pr[G_2=1]|$$

在第三个游戏 G_3 中，每次更新中计算用来验证一个文件是否含有关键词集 W' 的 $(T, \pi_0, \pi_1, \pi_2, I^{ind_j^{w_i}})$ 元组在 G_3 中随机产生为 $\left(T^*, \pi_0^*, \pi_1^*, \pi_2^*, \left(I^{ind_j^{w_i}}\right)^*\right)$，并维护一个列表 $DList$ 存储 (d_0, d_1, W') 来回应敌手 A 的陷门询问. 产生的 $\left(T^*, \pi_0^*, \pi_1^*, \pi_2^*, \left(I^{ind_j^{w_i}}\right)^*, d_0, d_1\right)$ 元组可以使用挑战者敌手安全模型基于 $q-ABDHE$ 假设被证明是安全的，该工作已被文献[10]证实，所以游戏 G_3 和 G_2 是不可区分的.

$$\left|\Pr[G_3=1]-\Pr[G_2=1]\right|\leqslant Adv_A^{q-ABDHE}(\lambda)$$

在第四个游戏 G_4 中，我们模拟所有哈希函数为随机原语，其中每一个原语维护一个列表存储输入输出对. 例如，对 h_1 输入一个 x，该原语随机选择一个字符串 $y=h_1(x)$ 作为输出并向 h_1-List 中插入 (x,y) 对. 因此，如果所有哈希函数是抗碰撞的，那么 G_3 和 G_4 是不可区分的.

$$|\Pr[G_3=1]=\Pr[G_4=1]|$$

在下一个游戏 G_5 中，使用随机生成的 $t_{w_i}^*\leftarrow\mathbb{G}_2$

代替 $t_{w_i}=e\left(H_0(w_i'),PK_{du}^r\right)$. 同时该游戏需要维护一个映射列表 $TList$ 用来存储 $(w_i,EV,t_{w_i}^*)$ 来回应 A 的陷门询问. 其中 $(g_1,g_1^r,PK_{du},H_0(w_i),t_{w_i})$ 元组是 $DBDH$ 元组，$(g_1,g_1^r,PK_{du},H_0(w_i),t_{w_i}^*)$ 是随机元组. 如果敌手可以区分 G_4 和 G_5，就意味着该敌手可以解决 $DBDH$ 问题，这与假设相矛盾，因此我们有：

$$|\Pr[G_5=1]-\Pr[G_4=1]|\leqslant Adv_A^{DBDH}(\lambda)$$

在最后的游戏 G_6 中，模拟器 S 使用泄露函数 $\mathcal{L}$（包含搜索模式和更新历史纪录）模拟敌手环境. 从对手角度来看 G_6 和 G_5 行为完全一样. 因此他们是不可区分的：

$$|\Pr[G_6=1]=\Pr[G_5=1]|=Ideal_{A,S}^{\Pi}(\lambda)$$

具体形式化的证明参考文献[12].

因此. 最终我们有

$$|\Pr\left[Real_A^{\Pi}(\lambda)\right]-\Pr[Ideal_{A,S}^{\Pi}(\lambda)=1]|\leqslant Adv_A^{DBDH}(\lambda)$$

因为 $Adv_A^{DBDH}(\lambda)$ 是计算可忽略的，本文方案是 $\mathcal{L}$－适应性安全的可搜索加密方案.

(2) 完整性验证

本文完整性验证方案是基于 Schnorr 签名[23]方案，该签名方案具有计算的高效性且该签名方案已在安全模型下被证明是安全的，并且本文方案是对密文进行签名验证，所以在智能合约公开透明的情况下也能保证数据的安全性.

(3) 搜索模式的隐私安全

本文所提的方案，服务器无法从陷门中猜测出有关关键字的信息. 在搜索阶段，云服务器收到陷门 $T_{W'}=(T_{w_i},d_0,d_1)$，其中

$$T_{w_i}=e\left(H_0(w_i')^{SK_{du}},EV\right)=e\left(H_0(w_i')^d,g_1^r\right)=e\left(H_0(w_i'),g_1^{dr}\right)$$

$$d_0=\left(a_0g_1^{-r_0}\right)^{\frac{1}{d}}\left(\prod_{k=1}^{\lambda}a_k^{w_k'}\right)^{\gamma}$$

$$d_1=PK_{du}^{\gamma}$$

其中 $w_i'=H(w_i\|\theta)$，θ 只由 DO 和 DU 两者维护，服务器无法获得 $H(w_i\|\theta)$，从而无法通过陷门查询的验证：

$$add_{t_{w_i}}^*=H_2(T_{w_i})=H_2\left(e\left(H_0(w_i'),g_1^{dr}\right)\right)=H_2(t_{w_i})$$

来猜测与之对应的关键字，达到了搜索模式的隐私保护.

(4) 前向安全

本文的前向安全由 $st_c^{w_i}$ 更新状态实现. 搜索陷门中的 $T_{w_i}=e\left(H_0(w_i')^{SK_{du}},EV\right)$ 是使用最新版本号生成的，由此陷门可计算得到 $val_{t_{w_i}}$，从而求得 $st_c^{w_i}$，而一个陷门对应一个更新状态 $st_c^{w_i}$，根据 $st_c^{w_i}$ 可计算出之前的更新状态 $st_{c-1}^{w_i}=F^{-1}(k_c^{w_i},st_c^{w_i})$，直到计算得到 $EDB\left(add_{st_0^{w_i}}\right)=\perp$，至此可以求出所有更新过的文件索引密文. 并且由于伪随机置换是安全的，所以根据 $st_c^{w_i}$ 是无法通过伪随即置换得到下一个状态的. 因此获得之前的某一个陷门无法获得后来更新的密文，达到了前向安全.

根据形式化定义证明：设更新泄露函数为 $\mathcal{L}_{\text{update}}(w)$，则根据文献[21]定义的前向安全，若存在无状态泄露函数 $\mathcal{L}'$ 使得

$$\mathcal{L}_{\text{update}}(w_i)=\mathcal{L}'(\text{op},\text{ind}_j)$$

即更新阶段不会泄露任何有关关键字的信息，在保密性证明中已得到证明，当哈希函数是安全的，$DBDH$ 以及 $q-ABDHE$ 困难性假设成立时，则有：

$$\mathcal{L}_{\text{update}}(w_i)=\perp$$

所以本文方案具有前向安全性.

(5) 后向安全

Bost 在文献[13]中正式定义了三种不同泄露类型的后向安全. 后向安全即在搜索过程中不会泄露之前插入但是后来又删除的信息. 而在本文方案中索引及操作以密文

$$EI_{ind_j^{w_i}}=H_1(PK_{du}^r,w_i\|j)\oplus(op\|ind_j^{w_i})$$

的形式存在，其中使用了最新版本号和数据使用者的公钥，在没有 SK_{du} 的情况下是无法获得索引和操作的，服务器无法获悉该文件是插入还是删除.

根据形式化定义证明：设更新泄露函数为 $\mathcal{L}_{\text{update}}(w)$，搜索泄露函数为 $\mathcal{L}_{\text{search}}(w)$，根据前向安全证明可得更新泄露泄露函数为：

$$\mathcal{L}_{\text{update}}(w_i)=\perp$$

而由于服务器无法获悉该文件是插入还是删除，所以本文方案满足

$$\mathcal{L}_{\text{search}}(w)=\mathcal{L}''(\text{TimeDB(w)},Update(w))$$

即搜索只泄露与 w 匹配的文件和插入时间，以及 w 所有的更新发生时间，因此本文方案为 Ⅱ 型后向安全.

6 性能分析

6.1 功能与效率分析

本节将对本文方案的功能及效率与 Wang[24]和

Cui[15]的方案进行比较. 为了简化分析,效率方面对Update、Trapdoor、Search、Verify阶段进行分析. 表3显示了本文方案和另外两个方案安全性能之间的差别. 三者都实现了对密文的多关键字搜索功能,使得更有效率地搜索到包含多关键字的数据,避免多余数据的返回. 而在此基础上另外两个方案都没有实现对搜索模式的安全保护. Wang的方案同时忽略了前向安全和后向安全的实现,使得在搜索数据时,云服务器会得到一些不必要的泄露信息. 在后向安全方面,Cui的方案是使用穿透加密方式实现的,在删除文件后服务器并不能得到删除之后的数据,但是该方案向云服务器泄露了何时发生了删除,所以该方案只满足弱后向安全. 而本文方案在保证云服务器无法访问被删除信息的同时确保了不会泄露给服务器删除操作的时间,使服务器无法区分删除和插入操作,因此本文方案满足Ⅱ型后向安全. Cui的方案和本文方案都具有可验证密文完整性的功能,不同的是本文使用区块链上的智能合约取代了传统的第三方可信任中心,根据其不可篡改性,我们随时可验证其正确性.

表3　功能性比较

方案	多关键字搜索	搜索模式安全	前向安全	后向安全	完整性验证	加密方式
Wang's	√	×	×	×	×	非对称
Cui's	√	×	√	Ⅲ型	√	对称
Our	√	√	√	Ⅱ型	√	非对称

表4显示了三种方案在更新、陷门生成、搜索以及验证时的效率,其中F/F^{-1}表示伪随机置换及其逆置换,e表示双线性对运算,pow、mul、div分别表示群上的幂次运算、乘法运算和除法运算,xor、sub、h分别表示异或、减法和哈希运算,Enc/Dec表示加密和解密运算,M表示所有关键词个数,m表示搜索关键词个数,d为Cui的方案中的一个参数.

表4　效率比较

	更新	陷门产生	搜索	验证
Wang's	$F+M(mul+sub)+Enc+M^2mul$	$(M+m)(mul+sub)+Enc+M^2mul$	$(M^2+1)mul+MEnc$	
Cui's	$(d+M+3)h+(d+M+Md+1)(xor+F)+(M+2)Enc$	$mEnc$	$(d+m+3)h+(d+m+md)(xor+F)+xor+Dec$	e
Our	$8h+4xor+F+4e+7pow+(M+2)mul$	$e+h+2pow+mmul$	$5h+3xor+2pow+3e+2mul+div+F^{-1}$	$2pow+mul+h$

在更新和陷门生成阶段,Wang的方案中用到了二维矩阵,需要进行矩阵向量的乘法运算,所以乘法运算的次数过多导致其在更新阶段需要耗费的时间最长,Cui的方案中虽然用到了加密算法,但由于使用的是时间耗费较短的对称加密算法且其他运算都是异或、哈希等运算,所以更新阶段的时间成本最短,本文方案要保证搜索模式安全以及Ⅱ型后向安全的功能,所以使得加密阶段以及陷门生成阶段的时间耗费比*Cui*的方案大一些.

在搜索阶段,由于Wang的方案中仍然用到了向量与矩阵的乘法,所以时间耗费最长,Cui的方案的搜索时间耗费受两点影响,第一点,在Cui的方案中需要用到主密钥中的一个参数d作为循环异或的次数,而该参数取值较小时安全性不高,取值较大时计算时间耗费较大,在接下来的实验分析中我们折中取值与我们的方案进行比较;第二点,Cui的方案本质上还是每解密出一个文件,再把每一个所要搜索的关键字进行计算并与其进行比较,即:

$$t=F(e_j,ind_i)$$

$$v=\bigoplus_{s=1}^{l}F.Eval(psk_{e_{j_s}},t)\oplus\bigoplus_{m=l}^{d}F(sk_{e_{j_m}},t)$$

虽然只是异或和伪随机置换运算,但是由于计算次数过多,大大增加了其时间成本. 而本文方案则是在得到一个文件时,仅需要一次运算,即:

$$\pi_1\cdot\frac{e\left(d_1,I^{ind_j^{w_i}}\right)\cdot\pi_2^{-r_0}}{e(\pi_0,d_0)}=v'$$

仅包含2e+2mul+pow+div次运算就可以判断其是否满足搜索条件. 所以本文方案的搜索效率是最高的.

验证阶段由于Wang等方案中没有完整性验证的功能所以不予比较,在Cui的方案中,验证过程为使用双线性对计算:

$$e\left(sig_i,g\right)=e(H_5(ind_i)\cdot g^{H_5(C_i)},pk)$$

虽然只用到了一次双线性对的运算,但是其计算消耗比较大. 本文方案的验证方法使用了幂次、乘法以及哈希运算,计算:

$$r'=g_1^{s_i}PK_{do}^{-e_i}$$

$$H(r', c_i) = e_i$$

其运算消耗要比双线性对运算低，所以在完整性验证方面的效率，本文方案要比Cui的方案较好.

综上所述，本文方案在安全性方面比Wang和Cui的方案更好. 效率方面，本文方案在更新和陷门生成阶段时间较长，但也低于Wang的方案，要比Cui的方案较差一些，但是在搜索阶段则大大降低了搜索时间，搜索效率最高，并且验证阶段的效率也要好于Cui的方案. Cui的方案中使用的是对称加密，在加密搜索过程中会有大量的数据使用者和数据拥有者的交互，这在现实中安全性及效率是比较差的. 在验证方面本文使用了智能合约进行完整性验证，使得验证结果更加透明以及不可篡改. 所以本文的搜索效率和安全属性是最高的.

6.2 实验分析

为了测试本文方案的实际性能，我们在主频为2.3 GHz、16 GB内存的Win10系统中，使用Python3.5实现了本文及文献[15][24]的方案，智能合约选择建立在以Ganache CLI为主的私有以太坊测试链上，智能合约使用*Go*(1.15.6)语言编译器运行，每次算法执行100次并取其平均值得到以下数据.

更新阶段的计算消耗如图5所示，x轴表示文件数量，y轴表示更新时的计算时间消耗，本实验采用5000个文件数据进行加密实验，并在不同文件数量时记录下更新时间，三个方案的计算消耗都随文件数量或关键字个数的增长而线性增长. Wang的方案在更新阶段使用了矩阵的乘法运算，所以计算消耗最大，本文方案中，需要进行关键字个数的乘法运算，而Cui的方案虽然进行了$d+M+Md+1$次的运算，但由于xor和F运算的时间较短，所以在更新阶段，计算消耗和文件数呈线性关系，当文件数增加到5000时，Cui的方案比本文方案快40%.

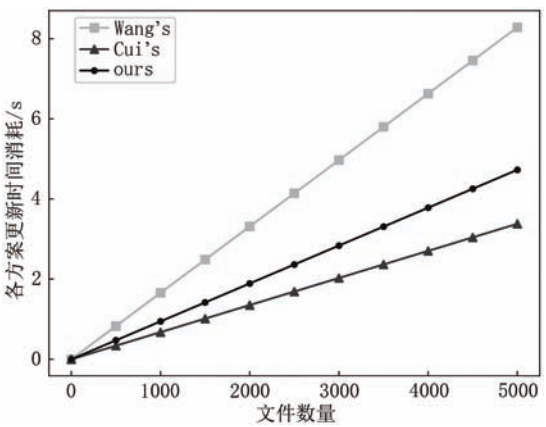

图5 更新阶段计算开销

陷门生成阶段的计算消耗如图6所示，x轴表示搜索关键字个数，y轴表示不同搜索关键字个数下的陷门生成时间消耗，本实验最多以50个关键词作为搜索关键词集. Wang的方案仍然涉及到矩阵乘法，所以时间消耗仍是最大的. Cui的方案只对所搜索关键词进行加密操作，所以陷门生成时间消耗非常小，当加密关键词个数为50时，仅需要0.05 ms，但是却会使得搜索阶段的计算消耗变得比较大. 本文方案由于使用了m次乘法运算，所以时间消耗比Cui的方案大，当加密关键字个数为50时，需要0.8ms，但是仍比Wang的方案计算消耗少很多.

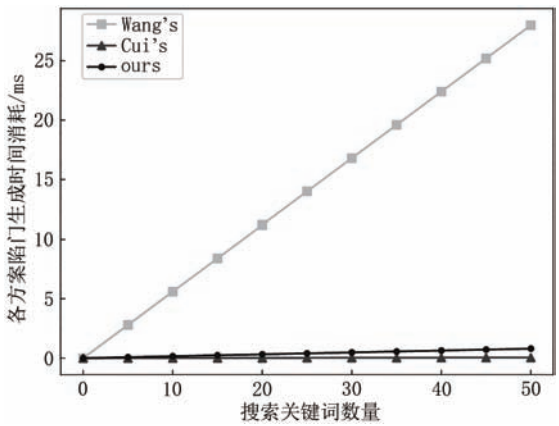

图6 陷门生成阶段计算开销

搜索阶段的计算消耗如图7所示，x轴表示文件数量，y轴表示不同文件数量下搜索时的计算时间消耗，本实验最多采用5000个文件数据进行加密搜索实验. Wang的方案仍涉及矩阵运算，能一次性计算得到所有文件，但是时间消耗仍然是巨大的. Cui的方案陷门生成很简单也预示着其搜索阶段的时间消耗比较大，其搜索过程即搜索到一个文件接着将文件的加密关键字与所有搜索关键字密文对比，所以其搜索时间复杂度应为$O(Nm)$，N为包含关键字w_i的文件数量，m为搜索关键字个数，当文件数量增加到5000时，其搜索时间为0.97 s. 本文方案搜索过程中只对搜索到的文件进行一次计算比较，所以本文方案的搜索时间复杂度为$O(N)$，当搜索文件数为5000时，本文方案仅需0.24 s即可搜索全部内容，比Cui的方案提高了约75%.

验证阶段的计算消耗如图8所示，x轴表示文件数量，y轴表示不同文件数量下验证时的计算消耗，本实验最多采用5000个文件数据进行验证实验. Wang的方案中没有验证密文完整性的功能，所以不予比较. Cui的方案中验证阶段虽然只对每一个签名

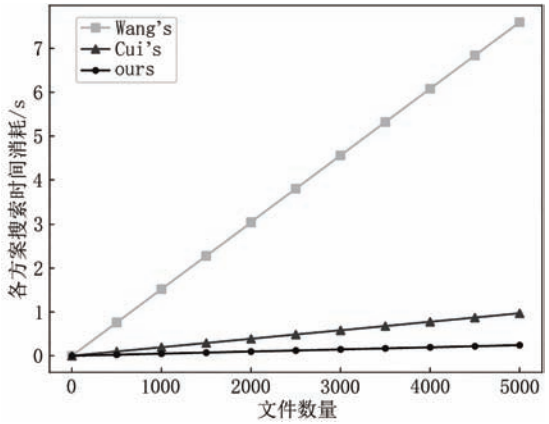

图7　搜索阶段计算开销

进行一次双线性对运算，但是双线性对运算的时间消耗相较乘法和哈希运算的时间消耗还是比较大的，当文件数量达到5000时，其计算消耗约为187 ms. 而本文的验证阶段使用了两次幂运算，一次乘法运算和一次哈希运算，时间消耗相比双线性对运算要小很多，当文件数量为5000时，计算消耗为104 ms，所以本文的验证方案效率要比Cui的方案好一些.

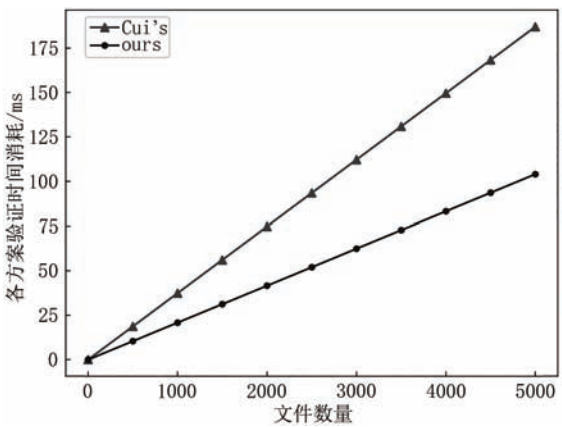

图8　验证阶段计算开销

7　总　结

本文提出了一种基于公钥加密的具有前向安全和后向安全的多关键字可搜索加密方案，通过在生成加密索引及陷门时为关键词加随机数取哈希值，保护了搜索模式的安全，在云服务器返回密文时使用区块链进行验证返回密文的完整性，在数据拥有者和数据使用者的一对一场景中具有较高的搜索效率和安全性能，本文通过理论分析和实验验证了方案的高效性和实用性. 在未来的研究中，可搜索加密中的前向安全和后向安全仍然是一个被重视的安全属性，考虑使用区块链确保该安全属性以获得更加高效安全的可搜索加密方案.

参　考　文　献

[1] Mather T，Kumaraswamy S，Latif S.Cloud security and privacy：an enterprise perspective on risks and compliance. New York，USA：O'Reilly Media，2009

[2] Kamara S，Lauter K.Cryptographic Cloud Storage//Proceedings of the Financial Cryptography and Data Security. Canary Islands，Spain ，2010：136-149

[3] SONG D X，WAGNER D，PERRIG A. Practical techniques for searches on encrypted data//Proceedings of the 2000 IEEE Symposium on Security and Privacy. Berkeley，USA，2000：44-55

[4] Curtmola R，Garay J，Kamara S，Ostrovsky R. Searchable symmetric encryption：Improved definitions and efficient constructions//Proceedings of the AM Conference Computer and Communications Security 2006. Alexandria Virginia，USA，2006：79-88

[5] Cash D，Jaeger J，Jarecki S，Jutla C S，Krawczyk H，Rosu M C，Steiner M. Dynamic searchable encryption in very-large databases：Data structures and implementation//Proceedings of the Network and Distributed System Security Symposium. San Diego，USA，2014：853-885

[6] Miers I，Mohassel P. IO-DSSE：Scaling dynamic searchable encryption to millions of indexes by improving locality//Proceedings of the Network and Distributed System Security Symposium (NDSS'17) (2017. California，USA，2017：34-46

[7] Boneh D，Crescenzo G，Ostrovsky R，et al. Public key encryption with keyword search//Proceedings of the Theory and Applications of Cryptographic Techniques. Interlaken，Switzerland，2004：506-522

[8] Golle P，Staddon J，Waters B. Secure conjunctive keyword search over encrypted data//Proceedings of the 2nd International Conference on Applied Cryptography and Network Security. Yellow Mountain，China，2004：31-45

[9] Zhang B，Zhang F G. An efficient public key encryption with conjunctive subset keywords search. Journal of network and computer applications，2011，34(1)：262-267

[10] Zhang J H，Wu M L，Wang J，Liu P，Jiang Z T，Peng C G. Secure and verifiable multi-keyword searchable encryption in cloud. Journal on Communications，2021，42(04)：139-149(in Chinese)
(张键红，武梦龙，王晶，刘沛，姜正涛，彭长根. 云环境下安全的可验证多关键词搜索加密方案. 通信学报，2021，42(04)：139-149)

[11] Bost R，Σοφος：Forward secure searchable encryption//Proceedings of the ACM SIGSAC Conference on Computer and Communications Security. Vienna，Austria ，2016：1143-1154

[12] Song X F，Dong C Y，Yuan D D，Xu Q L，Zhao M H. Forward private searchable symmetric encryption with optimized i/o efficiency. IEEE Transactions on Dependable and Secure Computing，2020，17(5)：912-927

[13] Bost R，Minaud B，Ohrimenko O. Forward and backward

private searchable encryption from constrained cryptographic primitives//Proceedings of the ACM SIGSAC Conference on Computer and Communications Security. New York, USA, 2017:1465-1482

[14] Sun S F, Yuan X L, Liu J K, Steinfeld R, Sakzad A. Practical backward-secure searchable encryption from symmetric puncturable encryption//Proceedings of the ACM SIGSAC Conference on Computer and Communications Security. Toronto, Canada, 2018: 763-780

[15] Cui J, Sun Y, Xu Y, Tian M M, Zhong H. Forward and backward secure searchable encryption with multi-keyword search and result verification. SCIENCE CHINA Information Sciences, 2022, 65(5): 159102

[16] Guo Y, Zhang C, Jia X H. Verifiable and forward-secure encrypted search using blockchain techniques//Proceedings of the IEEE International Conference on Communications. Dublin, Ireland, 2020: 1321-1329

[17] Li H G, Zhang F G, He J J, Tian H B. A searchable symmetric encryption scheme using blockchain. arXiv preprint: 1711.01030, 2017

[18] Li H G, Tian H B, Zhang F G, He J J. Blockchain-based searchable symmetric encryption scheme. Computers & Electrical Engineering, 2019, 73: 32-45

[19] Li H. Zhang C, Huang H J, Guo Y, et al. Algorithm for encrypted search with forward secure updates and verification. Journal of Xidian University, 2020, 47(5): 48-56 (in Chinese)
(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)

[20] Hu S S, Cai C J, Wang Q, Wang C, Luo X Y, Ren K. Searching an encrypted cloud meets blockchain: A decentralized, reliable and fair realization//Proceedings of the IEEE Conference on Computer Communications. Honolulu, USA, 2018: 792-800

[21] Stefanov E, Papamanthou C, Shi E. Practical Dynamic Searchable Encryption with Small Leakage//Proceedings of the Network and Distributed System Security Symposium (NDSS' 14) (2014. California, USA, 2014

[22] Zheng Y D, Lu R X, Shao J, Yin F, Zhu H. Achieving practical symmetric searchable encryption with search pattern privacy over cloud. IEEE Transactions on Services Computing, 2022, 15 (3): 1358-1370

[23] Schnorr C P. Efficient signature generation by smart cards. Journal of Cryptology, 1991, 4(3): 161-174

[24] Wang B, Song W, Lou W J, Hou T. Inverted index based multi-keyword public-key searchable encryption with strong privacy guarantee//Proceedings of the IEEE Conference on Computer Communications. Hong Kong, China, 2015: 2092-2100

SONG Xang-Fei, M. S. candidate. His main research interests is searchable encryption.

WANG Hua-Qun, Ph. D., professor. His main research interests include cryptography, blockchain technology, future mobile communication security etc.

Background

Nowadays, cloud servers are favored by enterprises and private individuals because of their high storage capacity and ease of operation. We can store private data encrypted on cloud servers. However, we need to obtain all the data to select what we need whenever we take data, which has a huge demand on bandwidth. Therefore, people start to study searchable encryption, using keywords to search ciphertext, which can not only quickly obtain the needed ciphertext data, but also avoid unnecessary information leakage. Nowadays, searchable encryption technology has developed to multi-keyword searchable encryption, which can search ciphertext with multiple keywords to avoid the return of redundant data and improve the search efficiency.

Although searchable encryption technology is very mature, there are still some shortcomings in some security attributes, of which preventing information leakage to semi-honest cloud servers is the main one. In this paper, forward and backward privacy security and search mode privacy protection and integrity verification of returned ciphertext are added on the basis of multi-keyword search. We use the method of encrypted status update to achieve forward privacy security, so that the cloud server can not predict the data information updated later through the previous search trap. We encrypt the operation and index together to achieve backward privacy security, so that the cloud server cannot distinguish between insert and delete operations. In terms of efficiency, compared with similar programs, our program is more efficient.

In future research, forward and backward privacy security in searchable encryption is still an important security attribute, and blockchain is considered to ensure this security attribute in order to obtain more efficient and secure searchable encryption.

This paper was supported by the National Natural Science Foundation of China (No. 62272238).