一种抵抗内存泄漏的可验证密文检索方案*

文◆漳州城市职业学院 邱林冰
阳光学院 刘燕江
漳州城市职业学院 郭亚峰

引言

近年来，云计算技术的迅速发展为学术界和IT行业带来了更多机遇。通过数据外包，资源有限的用户能够享受大规模数据存储和高效处理的便利。然而，云服务器具有半诚实且好奇的特点，且设备在互联网中时刻面临信息泄漏的安全威胁。此外，数据托管在云端使得用户在一定程度上失去了对数据的直接控制，从而引发数据安全性的担忧。为确保信息的机密性，越来越多的企业和个人采取加密措施，将敏感数据以密文形式存储在云服务器上。但加密后的数据特性使得传统明文检索方法不再适用。因此，在保障数据安全的前提下，如何实现高效密文检索成为亟须解决的重要技术难题。

在传统“半诚实且好奇”的云服务器模型中，服务器会严格遵循协议执行搜索任务，但在执行过程中可能会额外推断出与数据或搜索流程相关的信息。与此同时，为降低计算、存储以及网络开销，部分云服务器可能会采用自利策略。在这种环境中，云服务器不仅可能试图窃取更多隐私信息，还可能只返回部分检索结果。此外，数据拥有者为确保密文本身及其检索的安全性，通常将私密信息保存在非易失性内存中。在互联网环境中，用户设备时刻面临着敌手窃取私密数据的安全威胁。为防范此类风险，提出一种支持结果验证的抵抗内存泄漏的密文检索方案。该方案要求服务器必须证明其返回的搜索结果既完整又准确。为此，利用物理不可克隆函数和模糊提取器构造出的加密索引结构和新型验证结构，能够同时满足更高安全性要求和高效检索性能，确保用户获取完整准确的目标数据。

1 相关工作

2000年，Song等[1]提出了可搜索加密概念。他们在方案中实现了非交互式访问私密数据的方法，但面对庞大数据时其访问效率低下。2003年，Goh[2]利用布隆过滤器（Bloom Filter, BF）构建安全索引，但其方案的搜索存在一定错误率。2006年，Curtmola等[3]提出的加密检索方案SSE-1和SSE-2可以满足更高安全性和效率的自适应和非自适应模型。Boneh等[4]在其公钥加密方案中提出关键词检索，但由于搜索是利用双线性对和群元素之间的计算而实现，导致搜索时间开销远高于对称加密方案。因此，为实现高效密文检索性能，采用对称加密方法构造加密倒排索引。

2012年，Chai等[5]提出了“半诚实且好奇”的服务器模型。在该模型中，云服务器仅执行部分操作并返回部分搜索结果，给搜索工作带来安全挑战。Wang等[6]在模糊关键词检索方案中使用BF和消息认证码，而BF由于错误率缺陷，使得难以满足验证安全的基本要求。2015年，Li等[7]对可搜索加密相关工作进行了综述。2017年，陈等[8]利用哈希函数构造出支持验证结果的Merkle哈希树结构。2024年，在Wu等[9]的多客户端和多关键字可搜索对称加密方案中，利用同态加密技术实现了前向和后向隐私要求。2025年，Ji等[10]提出利用多项式编码和加性对称同态加密实现可验证方案，但较高的计算开销和有限的应用场

*【基金项目】2024年度福建省中青年教师教育科研项目（科技类）（JAT241341）；数智时代高等职业教育信息技术与数字素养课程改革项目（KSTZ2024）
【作者简介】邱林冰（1988—），男，福建漳州人，研究生，助教，研究方向：云存储安全。

景导致方案具有一定的局限性。以上方案虽然都支持验证服务器的搜索结果，但都是在数据拥有者需要妥善保管私密数据的前提下实现的。当面对云存储服务系统中的敌手通过侧信道攻击窃取内存中私密数据时，这些方案就显得无所适从。

陈等在 Sun 等 [11] 方案的索引树结构基础上，结合 Merkle[12-13] 的认证数据签名结构，并利用 Hash 函数构造出 Merkle 哈希树（Merkle Hash Tree，MHT）验证结构。在 MHT 构造过程中，所有节点都是其孩子节点数据的哈希值，这使得验证云服务器是否存在篡改、删除、伪造等非法行为成为可能。为应对日益激烈的云存储服务安全挑战，在 MHT 结构基础上，利用物理不可克隆函数 [14-17]（Physically Unclonable Function, PUF）和模糊提取器 [18]（Fuzzy Extractor, FE）构造出新型验证结构——物理不可克隆验证树（Physically Unclonable Verifiable Tree, PUVT）。在此基础上，提出的基于 PUF 和 FE 的抵抗内存泄漏攻击的可验证密文检索方案（Verifiable Ciphertext Search Scheme against Memory Leakage, VCSS-ML）满足了抵抗非易失性内存 [19] 泄漏攻击的更高安全性要求。

表 1 本文使用的符号

符号	说明
$D=\{d_1,d_2,\cdots,d_n\}$	明文文档集
$C=\{c_1,c_2,\cdots,c_n\}$	密文文档集
w_i	查询关键词
$W=\{w_1,w_2,\cdots,w_m\}$	关键词集
I	加密倒排索引
π	验证路径
idc	加密文档标识符
$C_{wi}=\{c_{w1},c_{w2},\cdots,c_{wl}\}$	包含检索关键词 w 的加密文档集
$D_{wi}=\{d_{w1},d_{w2},\cdots,d_{wl}\}$	包含检索关键词 w 的明文文档集
$IDC_{wi}=\{id'_1,id'_2,\cdots,id'_l\}$	包含检索关键词 w 的加密文档标识符集
$rv_{i,j}$	PUVT 叶子节点值
$rv_{i,o}$	PUVT 分支节点值
α	认证符，即包含检索关键词 w_i 的 PUVT
r	PUVT 根节点值
r'	验证执行时计算得到的 PUVT 新根节点值
v_r	验证请求，即随机选取的加密文档标识符
$B=(r^1_{v\,1,o},r^2_{v\,2,o},\cdots\cdots,r^q_{v\,h,o})$	验证节点至根节点路径上所有兄弟节点值，h 为 α 的深度

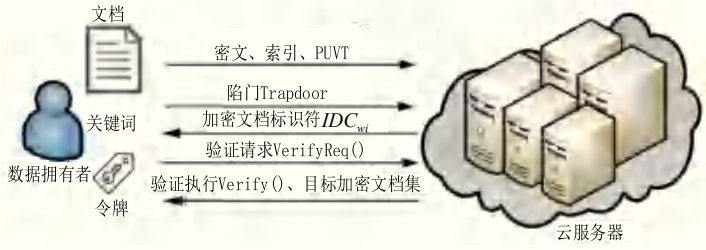

图 1 系统模型

2 方案形式化描述

2.1 符号定义

相关研究指出，一个 PUF 族由一个算法对（*Sample* 和 *Eval*）构成，*Sample* 算法以安全参数作为输入，输出具有 PUF 族特征的索引标识符 *idp*。*Eval* 算法以激励 *s* 为输入，输出响应 *r*。PUF 是一种噪声函数，会出现有界噪声的现象，因此需要利用 FE 恢复有用的私密信息。根据相关文献的定义，FE 是由两个高效算法（*Gen*, *Rep*）构成，其中 *Gen* 为生成算法，计算输入数据生成一个随机串和辅助数据；*Rep* 为重现算法，在辅助数据的参与下，将输入的有界噪声随机串恢复出 *Gen* 算法生成的随机串。

下文中将用到的符号定义见表 1。

2.2 系统模型

在云存储服务环境中，设定方案由数据拥有者和云服务器的两个实体构成，系统模型如图 1 所示。

图 1 中，数据拥有者（Data Owner, DO）会进行密文检索操作。云服务器本身具备半诚实且好奇的特点，对外提供数据存储、检索和验证等服务。考虑到数据的安全要求，数据拥有者首先要加密文档集，构建加密倒排索引，再利用物理不可克隆函数构造出新型验证树 PUVT 实现搜索结果的完整性验证。其次，数据拥有者将密文、加密索引和 PUVT 上传至云服务器，本地只负责保存密钥和 PUVT 根节点值等私密信息。云服务器执行加密索引和 PUVT

关键词	倒排记录指针	文档列表
关键词1	指针1	文档1信息、文档2信息、文档3信息
关键词2	指针2	文档2信息、文档4信息
关键词3	指针3	文档3信息
关键词4	指针4	文档4信息
.....	.....	.....
关键词n	指针n	文档3信息、文档5信息.....

图 2 传统明文倒排索引

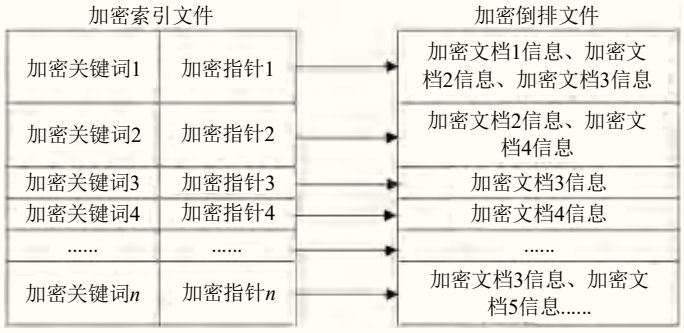

图 3 加密倒排索引

的遍历操作，将搜索结果和验证信息返回给数据拥有者。数据拥有者再根据返回的加密文档标识符和验证信息进行验证操作。最后，云服务器根据经数据拥有者确认的加密文档标识符集返回相应密文集，数据拥有者解密密文集得到包含查询关键词的明文文档集。

3 方案详细设计

VCSS-ML 方案由构建索引、构造 PUVT、检索密文和验证结果 4 个过程构成。

3.1 构建索引

传统明文倒排索引如图 2 所示，由关键词词典、倒排记录指针和文档列表构成。其中，关键词词典是从明文文档中提取的关键词集合；倒排记录指针是指向所有包含关键词文档的地址；文档列表为倒排文件，记录关键词在文档中的相关信息（如相关文档地址集）。

为实现快速且安全的密文检索性能，构造出加密倒排索引（见图 3）。具体做法是，对传统明文倒排索引中的关键词进行加密，将原有关键词替换为加密关键词；对倒排记录指针和明文文档标识符加密后，替换原有的指针集和文档信息集。通过这样的操作，最终分别得到由加密关键词、加密指针和加密倒排文件构成的加密倒排索引。

为构造出符合方案安全要求的加密倒排索引，DO 需要先执行密钥生成算法 Keygen(1^{λ})：输入参数 λ，具体步骤如下。

（1）选取满足 PUF 条件的 3 组参数，即（p+log$_2n$,d_1,δ_1）PUF_1、（t,d_2,δ_2）PUF_2 和（t',d_3,δ_3）PUF_3。

（2）输出密钥 K=（PUF_1,PUF_2,PUF_3），其中，PUF_1 用于构造关键词查找表，PUF_2 用于加密文档集，PUF_3 用于构造 PUVT。

DO 再执行索引构建算法 BuildIndex(K,D)，具体步骤如下。

（1）初始化

1）遍历明文文档集 D，提取关键字，创建去重关键字的词典 W'。

2）词典 W 是由向 W' 中插入若干虚拟关键字构成的集合。

3）为 W 中的每个关键字 w_i 创建链表 $D(w_i)$。

（2）构造关键字查找表 T

1）对于 W 中的每个关键字 w_i，计算 $ut_{i,j}$=$PUF_1(w_i||j)$ 和 ($rt_{i,j}$, $adt_{i,j}$)=$FE_1.Gen(ut_{i,j})$，其中 j 表示 $D(w_i)$ 中的文档序号。

2）对于 W 中的每个关键字 w_i 和取值为 [1, |$D(w_i)$|] 中的变量 j，设置 T [$rt_{i,j}$]=$idc_{i,j}$。如果 v<$\tilde{v}$，则将 $\tilde{v}$–v 个的 m_1 位随机字符串随机插入 T 中的空闲位置。

（3）构造辅助查找表 T'

1）对于取值为 [1,m] 中的变量 i 和取值为 [1,|$D(w_i)$|] 中的变量 j，设置 T' [w_i]=$adt_{i,j}$。

2）对于取值为 [m+1,|Ω|] 中的变量 i 和取值为 [1,n] 中的变量 j，设置 T' [w_i]=$adrt_{i,j}$，其中 $adrt_{i,j}$ 长度与 $adt_{i,j}$ 相等的随机字符串。

（4）生成密文 C_j

1）对于明文文档集 D 中的每个文档 D_j（$1 \leqslant j \leqslant n$），计算 us_j= $PUF_2(id(D_j))$、(rc_j,adc_j)=$FE_2.Gen(us_j)$ 和 C_j=$Enc(rc_j,D_j)$。

2）对于取值为 [1,n] 中的变量 j，设置密文查找表 T_c[id_j]=adc_j。

（5）将输出的索引 I=T 和加密文档集 C=(C_1,C_2,⋯,C_n) 发送给服务器。

3.2 构造 PUVT

陈等构造的 MHT 是一棵完全二叉树。MHT 的所有节点都存储着子节点的哈希值。MHT 结构如图 4 所示。为获得目标安全性，在 MHT 的基础上结合 PUF 构造出 PUVT。PUVT 构造过程如下。

将 MHT 中节点计算所使用的哈希函数替换成 PUF。PUVT 生成算

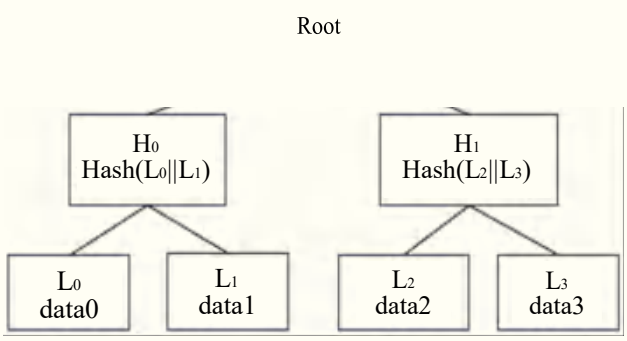

图 4 MHT 结构

法 BuildPUVT(I,K) 由 DO 负责执行，具体步骤如下。

（1）对于 W 中的每个关键字 w_i，遍历索引 I，得到含关键字 w_i 的所有加密文档标识符集 IDC_{wi}= {$idc_{wi,j}$,1 ⩽ i,j ⩽ n}。

（2）计算 $uv_{i,j}$=PUF_3($idc_{wi,j}$) 和 ($rvl_{i,j}$,$advl_{i,j}$)=FE_3.Gen(uv_j)，其中 ifc_{wi}^{j} 为包含密态关键词 wc_i 的第 j 个密文文档标识符，$advl_{i,j}$ 作为查找验证树叶子节点的辅助数据。

（3）构造叶子节点辅助查找表 T_{vl}。

1）为取值在 [1,|D(w_i)|] 中的变量 j，设置 T_{vl}[$idc_{wi,j}$]=$advl_{i,j}$。

2）对于取值为 [m+1,|Ω|] 中的变量 i 和取值为 [|D(w_i)|+1,n] 中的变量 j，设置 T_{vl}[$idc_{wi,j}$]=$advlr_{i,j}$，其中 $advlr_{i,j}$ 长度与 $advl_{i,j}$ 相等的随机字符串。

（4）PUVT 叶子节点定义为 L_o=<w_i,$rv_{i,j}$>，双亲节点定义为 $P_{2,o}$；内部节点定义为 $I_{l,o}$；内部节点的孩子节点定义为 $CI_{l-1,o}$。其中，l 表示节点所在层次，o 表示在当前层节点顺序。PUVT 节点值计算如下。

1）串接两个叶子节点并利用 PUF_3 计算 $P_{2,o}$=PUF_3(L_o ‖ L_{o+1}) 和 ($rv_{2,o}$,$adv_{2,o}$)=FE_3.Gen($P_{2,o}$)，得到其双亲节点值 $rv_{2,o}$；

2）对于其余内部节点，计算 $I_{l,o}$=PUF_3($CI_{l-1,o}$ ‖ $CI_{l-1,o+1}$) 和 ($rvi_{l,o}$,$adv_{l,o}$)=FE_3.Gen($I_{l,o}$)，得到内部节点值 $rv_{l,o}$。

3）构造内部节点辅助查找表 T_{vi}。

①为取值在 [1,|D(w_i)|] 中的变量 j，设置 T_{vi}[w_i]=$advi_{i,j}$。

②对于取值为 [m+1,|Ω|] 中的变量 i 和取值为 [|D(w_i)|+1,n] 中的变量 j，设置 T_{vi}[$CI_{l-1,o}$ ‖ $CI_{l-1,o+1}$]=$advir_{i,j}$，其中 $advir_{i,j}$ 长度与 $advi_{i,j}$ 相等的随机字符串。

经过上述步骤 1）~步骤 3）迭代计算，最终得到包含关键词 w_i 的 PUVT（即认证符 a_i）及其根节点值 r_i=$rv_{l,o}$。

（5）经过步骤（1）~步骤（4），得到验证辅助查找表 T_v=(T_{vl},T_{vi})、词典 W 的认证符集 A=(α_1,α_2,⋯,α_m) 和根节点值集 R=(r_1,r_2,⋯,r_m)，随后 DO 将 A 提交给云服务器，将 R 发送给用户。

3.3 检索密文

检索密文包括两个过程，分别是陷门生成算法和检索算法。陷门生成算法 TrapdoorGen(K,w_i) 由 DO 负责执行，具体步骤如下。

（1）对于查询关键字 w_i 和取值为 [1,n] 中的变量 j，计算 $\hat{u}t_{i,j}$=PUF_1(w_i‖j) 和 $rt_{i,j}$=FE_1.Rep($\hat{u}t_{i,j}$,$adt_{i,j}$)，其中 adtj 是辅助查找表 T' 中的数据。

（2）输出陷门 T_{wi}=$rt_{i,j}$。

检索算法 Search(I,T_{wi}) 由云服务器负责执行。根据陷门 T_{wi} 遍历索引 I，对于取值为 [1,n] 中的变量 j，如果 T[$rt_{i,j}$] 不为无效符号，那么将加密文档标识符 $idc_{i,j}$ 添加到加密文档标识符集 IDC_{wi}= {$idc_{i,j}$,1 ⩽ j ⩽ k} 中。

3.4 验证结果

验证结果包括 4 个过程，分别是验证请求算法、验证接受算法、验证执行算法和解密算法。

当 DO 希望判断云服务器是否真实可信时，首先执行验证请求算法 VerifyReq(IDC_w)，选取 IDC_w 中任一加密文档标识符 $idc_{i,j}$，计算 $\hat{u}t_{i,j}$=PUF_3($idc_{i,j}$) 和 $rv_{i,j}$=FE_3.Rep($\hat{u}t_{i,j}$,$advl_{i,j}$)，向服务器发起验证请求 vr=$rv_{i,j}$。

接着，由云服务器执行验证接受算法 RouteGen(vr,α)，具体步骤如下。

（1）根据验证请求 vr，遍历所有的 PUVT，找出属于 w_i 的认证符 α_i。

（2）遍历 α_i，记录从叶子节点至根节点路径上的所有兄弟节点值 B=($rv^1_{1,o}$,$rv^2_{2,o}$,⋯⋯,$rv^q_{h,o}$)，其中，h 表述 PUVT 深度，q 表述遍历到的兄弟节点数。

（3）输出验证路径 π={$B_{r,o}{}^u$: 1 ⩽ r ⩽ h, 1 ⩽ u ⩽ q}。

DO 根据收到的 π 执行验证执行算法 Verify(π, R, T_v)，具体步骤如下。

（1）对于 1 ⩽ r ⩽ h, 1 ⩽ u ⩽ q，遍历辅助验证数据表 T_v 获取匹配查询关键字 w_i 的 PUVT 节点辅助验证数据 $advl_{i,j}$ 和 $advi_{i,j}$。

（2）对于 1 ⩽ j ⩽ k 和验证路径 π，计算 $\hat{u}v_{i,j}$=PUF_3($idc_{i,j}$)、$rvl_{i,j}$=FE_3.Rep($\hat{u}v_{i,j}$,$advl_{i,j}$) 和 $I_{l,o}$=PUF_3($CI_{l-1,o}$ ‖ $CI_{l-1,o+1}$)、$rvi_{l,o}$=$FE3$.Rep($I_{l,o}$,$advi_{i,j}$)，直至得到匹配查

询关键字 w_i 的根节点值 r_i'。

（3）比较 r_i 与 r_i'。

1）若 $r_i=r_i'$，则输出 1，说明服务器返回检索结果是完整且正确的，再执行解密算法。

2）若 $r_i \neq r_i'$，则输出 0，要求服务器重新执行检索算法。

最后，DO 对通过验证的返回结果执行解密算法 Decrypt(K, IDC_{wi},T_c,C_w)，具体步骤如下。

（1）云服务器返回匹配查询关键字 w_i 的加密文档集 C_{wi} 给数据拥有者。

（2）遍历加密文档标识符集 IDC_{wi} 和密文查找表 T_c，先计算 $\hat{us}_j=PUF2(idc_j)$ 和 $rc_j=FE2.Rep(\hat{us}_j, adc_j)$，再利用 AES 算法和密钥 K 计算 $\mathrm{Dec}(rc_{i,j},C_{wi,j}), j \in [1,k]$，得到匹配查询关键字 w_i 的明文文档集 D_{wi}。

4 安全性与性能分析

4.1 安全性分析

鉴于半诚实且好奇的云服务器特点，本文侧重于索引安全和验证安全两个方面的安全性分析。

4.1.1 索引安全

索引的安全性主要取决于关键词、密文文档和索引信息的私密性及其关联性的安全。首先，对文档集中提取的关键词和文档标识符加密，再利用加密关键词和加密文档标识符构建加密倒排索引。索引中的信息都以密态存储，不存在明文语义上的顺序，所以攻击者在未获取明文相关信息的前提下，可以满足已知密文索引词的语义攻击和统计攻击的抵抗要求。其次，在构建加密索引时，不仅对关键词和链表的节点地址加密，还在索引文件中存储加密文档信息，而非文档地址。这样，一方面可以确保索引词频的私密性，另一方面进一步切断了用户查询关键词与密文信息的联系，从而最大程度地保证检索关键词、密文文档和索引之间的零关联性。

4.1.2 验证安全

验证部分的安全性威胁主要来源于云服务器可能利用 PUF 和 FE 计算出的随机字符串值欺骗用户执行验证算法，进而获取整个 PUVT，致使用户检索错误信息。在算法描述中，由于 PUF 本身的特性，认证符和验证请求的执行过程可能存在安全隐患。PUF 具有有界噪声的特点，因此可以利用 FE 克服 PUF 的噪声缺陷。

此外，PUF 面临着建模攻击、侧信道攻击、物理入侵以及环境因素等安全威胁。这些威胁因素的存在意味着云服务器可以通过收集大量“挑战—响应对（CRP）”，并利用机器学习算法（如逻辑回归、支持向量机和神经网络）建立数学模型，从而预测未知挑战的响应；利用 PUF 工作时泄露的功耗、电磁辐射、时延等物理信息，通过对这些数据的精密测量和分析，试图推导出内部结构或关键参数；通过解封、显微观察等物理手段，试图重构 PUF 内部结构。

由于 PUF 依赖于芯片制造中不可控的微小工艺差异，每个 PUF 实例都独一无二且不可预测，这一特性使得即使云服务器获得部分 CRP 数据，也难以重建完整模型。设计 PUF 电路时，一方面通过均衡功耗设计、屏蔽和随机掩码等手段，减少功耗、电磁等侧信道信息的泄露；另一方面对芯片进行防篡改封装、物理隔离和防逆向工程设计，以对抗云服务器的侧信道攻击，有效阻止云服务器直接获取 PUF 内部结构信息。PUF 本身具备的安全特性，使得云服务器即使获得了非易性内存中的辅助数据，也无法获取关于文档与关键词、PUVT 和验证过程的任何信息，从而进一步提高整体方案的安全性。

由上述分析可知，本方案在索引安全和验证安全两个方面都能够很好地抵抗现有可能的安全性攻击，并能够准确识别服务器是否存在篡改、删除等不良行为，实现搜索结果的正确性与完整性验证。

4.2 性能分析

根据相关研究可知，PUF 的性能与哈希函数接近，因此实验中采用哈希函数替代 PUF 进行 VCSS–ML 方案的性能分析。VCSS–ML 实验是在配置为 Intel Core i9–14900HX 2.2GHz、64G RAM 的 PC 环境中进行。实验中，随机生成文档集，分成 500、1000、2000、4000、8000 和 16000 篇等 6 组文档集进行各项实验。接下来，从存储开销和计算开销两个方面与陈等的 MHT 方案进行对比分析。

4.2.1 存储开销

VCSS–ML 的存储开销主要体现在由云服务器负责存储的索引和 PUVT。

（1）索引

从上述 VCSS–ML 的详细介绍可知，索引的存储开销主要受文档个数的影响。此外，明文索引受文档标识符、关键词个数及其长度的影响，而两个方案的加密索引都是利用加密函数对文档标识符、关键词加密计算得到固定长度的输出，所以明文索引相比于加密索引需要更多的存储空间。

两个方案的索引大小都随文档个数呈线性增长（见图 5）。在构造

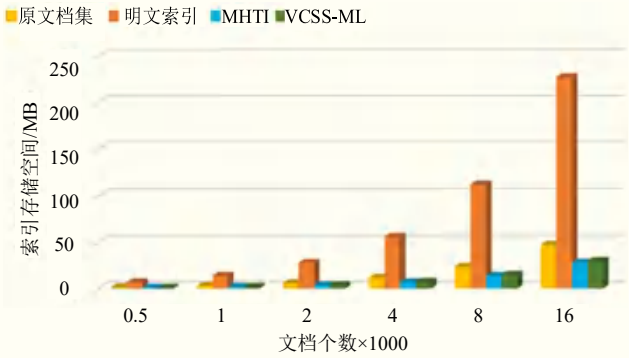

图 5 索引存储空间

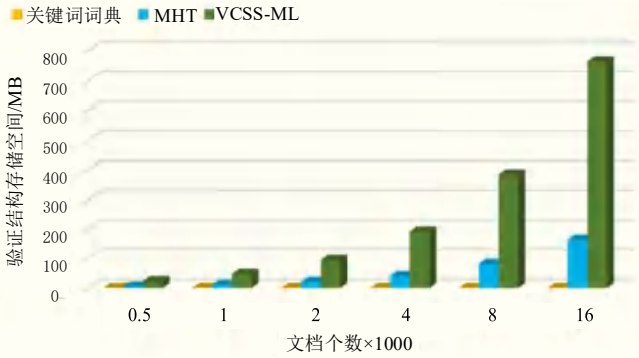

图 6 验证结构存储空间

加密索引过程中，VCSS–ML 相比于 MHT 需要存储额外的查找辅助表，但是查找表本身只负责存储加密关键词，只需较小的存储空间，因此 VCSS–ML 的索引开销与 MHT 接近。

（2）验证结构

从上述分析可知，VCSS–ML 验证结构 PUVT 和 MHT 验证结构的存储开销主要受到文档个数的影响。与索引构造不同的是，在构造 PUVT 过程中，不仅要为经 FE 计算每个叶子节点间 PUF 值得到的辅助数据创建表格，还需要在分支节点间计算时构造辅助数据表，故而 PUVT 的存储开销相比于 MHT 会更大一些，且对大规模文档集更加敏感（见图 6）。

4.2.2 **计算开销**

VCSS–ML 的计算开销主要体现在构造索引、构造验证树、生成陷门、执行检索和执行验证等 5 个方面。

（1）构造索引

在 VCSS–ML 中，构造索引分成两个步骤。先根据关键词词典生成文档向量和查询向量。再利用 PUF 和 FE 加密文档向量和查询向量，并生成查找辅助表。而 MHT 的索引构造方法仅仅在加密函数上有所不同，且不生成查找辅助表。因此，构造索引的时间开销与文档和关键词个数密切相关。索引构造时间如图 7 所示，当相同词典大小时，文档个数为 500 和 16000 时，VCSS–ML 索引构造时长分别为 37.704ms 和 860.288ms，而 MHT 索引构造时长分别为 11.235ms 和 328.884ms。两个方案的索引构造开销都随文档个数的增加而线性增长，由于 VCSS–ML 的索引构造涉及查找辅助表的生成，因此 VCSS–ML 相较于 MHT 需要更多的索引构造时间。然而，当文档集规模越大时，它们之间的时长差距将会逐渐缩小。

（2）构造验证树

验证树结构构造时如图 8 所示，本方案 VCSS–ML 和 MHT 方案在构造验证树所需的时间都会随叶子节点个数的增加而呈线性增加的趋势。两个方案的验证树构造过程相似，但在构造 PUVT 时，叶子节点值和分支节点值的计算都涉及辅助表的构造和存储，所以构造 PUVT 所需时间会相对多一些，而 MHT 方案并未涉及辅助表的生成和存储，但当文档规模增大时，其验证树构造时间也明显增大。

（3）生成陷门

陷门生成时间如图 9 所示，在文档个数分别为 500 和 16000 时，VCSS–ML 陷门生成时间分别为 56us 和 88.8us，而 MHT 陷门生成时间分别为 21us 和 28.2us。因此，随文档个数的增加，陷门生成时间增加不明显。VCSS–ML 在

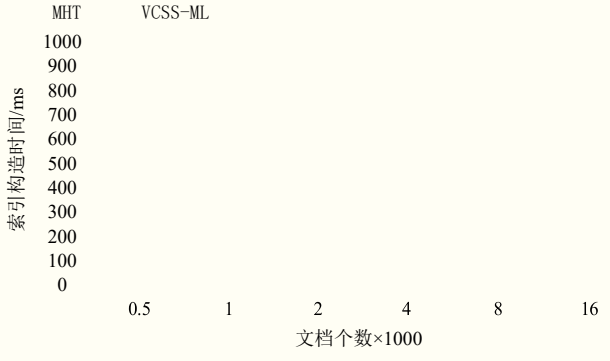

图 7 索引构造时间

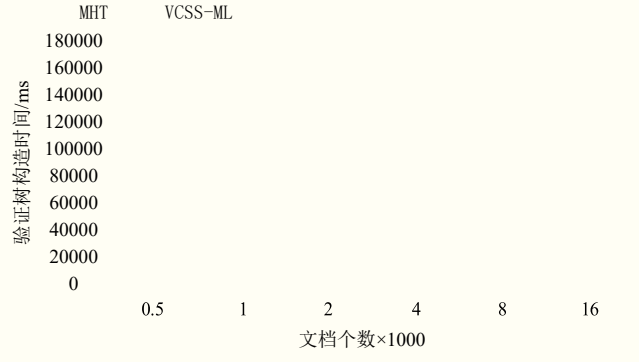

图 8 验证树结构构造时间

生成陷门时，同样涉及 FE 的计算，由于 PUF 和 FE 计算的高效性，因此相比于 MHT，VCSS-ML 陷门生成时间会有所增加，但增加的幅度不大。

（4）执行检索

检索时间如图 10 所示，文档个数为 500 和 16000 时，VCSS-ML 检索时间开销分别为 126.899us 和 94.299us，而 MHT 检索时间开销分别为 36.1us 和 34us。两个方案的检索时间开销并未受到文档规模的明显影响。检索过程中，VCSS-ML 除了要经过 PUF 和 FE 计算，还需要遍历查找辅助表和数据查找表。因此，相比于 MHT 的检索时间，VCSS-ML 检索时间相对较高。

（5）执行验证

在 MHT 方案中，验证过程是用户利用服务器返回的验证路径进行一系列哈希计算，最终将得到 MHT 的根节点与本地存储的根节点进行比较。而 VCSS-ML 不仅涉及计算效率与哈希函数接近的 PUF 计算，还需要 FE 和辅助表的遍历。这种方式的计算开销上远小于将所有检索结果取回再进行计算的方式。因此，VCSS-ML 方案的验证时间随着叶子节点个数的增加而增加（见图 11）。相比于 MHT，VCSS-ML 验证时间对文档个数更加敏感。

从上述对比分析可以看出，在构造索引、构造验证树、生成陷门、执行检索和执行验证等 5 个方面，VCSS-ML 的计算开销相对于 MHT 方案略有增加。但是，VCSS-ML 方案能够在保持较高的存储和计算效率前提下，为用户提供更高地抵抗内存泄漏的安全性。

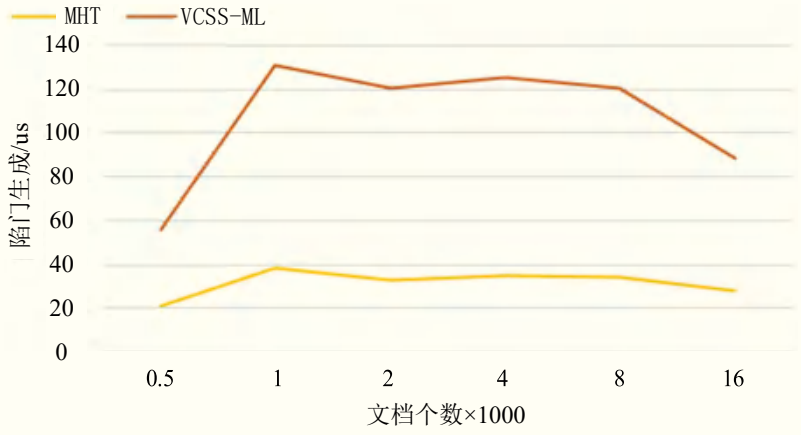

图 9 陷门生成时间

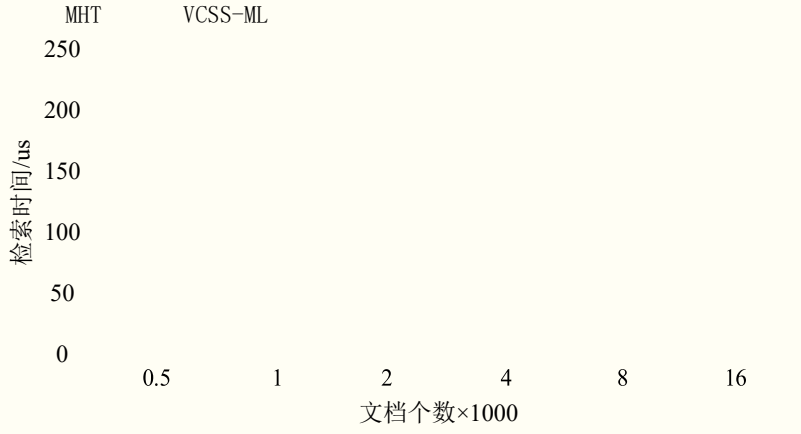

图 10 检索时间

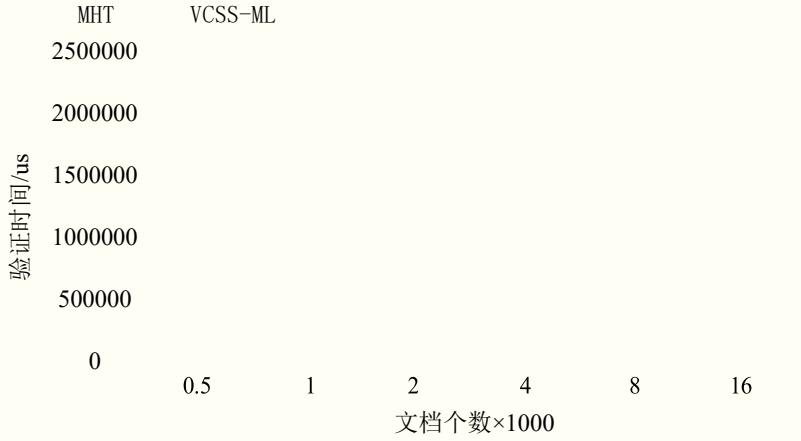

图 11 验证时间

结语

本文针对“半诚实且好奇”的服务器对搜索结果进行攻击和联网设备时刻面临信息泄露的可能性，提出了一个支持抵抗内存泄漏的可验证服务器行为的密文检索方案。一方面，通过利用 PUF 和 FE 构建安全索引、关键词陷门和生成密文等增强用户数据的隐私安全性能，实现高效检索；另一方面，通过利用 PUF 和 FE 对 MHT 进行改进，构造出的 PUVT 验证树结构实现对可能存在的篡改、删除和伪造等未授权的非法

行为进行验证，同时满足用户无需担心非易失性内存泄漏的更高安全要求。理论和实验分析表明，本方案在实现高效检索外包数据的基础上，能够提供更高地抵抗内存泄漏的可验证安全性能。

引用

[1] Song D X,Wagner D,Perrig A.Practical Techniques for Searches on Encrypted Data[C]//Security and Privacy,2000(S&P 200).Berkeley:IEEE Press,44-55.

[2] Goh E J.Secure Indexes[J].IACR Cryptology ePrint Archive,2003:216-235.

[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] Boneh D,Di Crescenzo G,Ostrovsky R,et al.Public Key Encryption with Keyword Search[C]//International Conference on the Theory and Applications of Cryptographic Techniques.Berlin,Heidelberg:Springer Berlin Heidelberg,2004:506-522.

[5] Chai Q,Gong G.Verifiable Symmetric Searchable Encryption for Semi-honest-but-curious Cloud Server[C]//Proceeding of 2012 IEEE International Conference on Communications(ICC).Ottawa:IEEE Press,2012:917-922.

[6] Wang J F,Ma H,Tang Q,et al.A New Efficient Verifiable Fuzzy Keyword Search Scheme[J].Journal of Wireless Mobile Networks,Ubiquitous Computing and Dependable Applications,2012,3(4):61-71.

[7] 李经纬,贾春福,刘哲理,等.可搜索加密技术研究综述[J].软件学报,2015,26(1):20.

[8] 陈兰香,邱林冰.基于Merkle哈希树的可验证密文检索方案[J].信息网络安全,2017,(4):1-8.

[9] Wu P,Shen J,Cao Z,et al.MMKFB:Multi-client and Multi-keyword Searchable Symmetric Encryption with Forward and Backward Privacy[J].Frontiers of Computer Science,2024,19(3):193804-193804.

[10] Ji L,Li J,Zhang Y,et al.Verifiable Searchable Symmetric Encryption Over Additive Homomorphism[J].IEEE Transactions on Information Forensics and Security, 2025,20:1320-1332.

[11] Sun W H,Wang B,Cao N,et al.Verifiable Privacy-preserving Multi-keyword Text Search in the Cloud Supporting Similarity-based Ranking[J].IEEE Transactions on Parallel and Distributed Systems,2013,99(11):1-11.

[12] Merkle R C.Method of Providing Digital Signatures:U.S.Patent 4,309,569[P]. 1982-1-5.

[13] Merkle R C.A Certified Digital Signature[C].Advances in Cryptology—CRYPTO'89 Proceedings.Springer Berlin/Heidelberg,1990:218-238.

[14] Ravikanth P S.Physical One-Way Functions[J].Science,2002,297(5589): 2026-2030.

[15] Willers O,Huth C,Guajardo J,et al.MEMS Gyroscopes as Physical Unclonable Functions[C]//ACM Sigsac Conference on Computer and Communications Security. ACM, 2016:591-602.

[16] Park S Y,Lim S,Jeong D,et al. PUFSec:Device Fingerprint-based Security Architecture for Internet of Things[C]//IEEE INFOCOM. IEEE,2017.

[17] Brzuska C,Fischlin M,Der H,et al.Physically Uncloneable Functions in the Universal Composition Framework[M]// Advances in Cryptology–CRYPTO 2011.Springer Berlin Heidelberg, 2011:51-70.

[18] Dodis Y,Reyzin L.Fuzzy Extractors:How to Generate Strong Keys from Biometrics and Other Noisy Data[C]//International Conference on the Theory and Applications of Cryptographic Techniques.Springer,Berlin,Heidelberg,2004:523-540.

[19] Armknecht F,Maes R, Sadeghi A R,et al.Memory Leakage-Resilient Encryption Based on Physically Unclonable Functions[C]//International Conference on the Theory and Application of Cryptology and Information Security:Advances in Cryptology.Springer-Verlag, 2009:685-702.