基于分层结构的匹配量隐藏加密多重映射方案

陈晶，李瀚星，何琨，加梦，李雨晴，杜瑞颖
（武汉大学国家网络安全学院，湖北 武汉 430040）

摘 要：匹配量隐藏的加密多重映射（EMM）方案可以防止攻击者利用匹配量泄露推理搜索的明文，但是现有方案存在查询计算开销较大的问题。基于被检索数据的匹配量往往服从齐夫定律的特性，设计了一种分层结构的匹配量隐藏 EMM 方案。相对将全部键值匹配量填充至相等的朴素设计，所提方案将对整体数据的填充转为对多块子数据的填充，减少了存储开销，并实现了常数复杂度的查询开销。安全性分析表明，所提方案能够在查询结果无损的情况下实现匹配量隐藏。仿真结果表明，与当前最高效的方案 XorMM 相比，所提方案能够以增加 10%的存储开销为代价，减小 90%的查询计算开销，显著提高查询效率。

关键词：云存储；可搜索加密；匹配量隐藏；结构化加密；分层结构
中图分类号：TP309
文献标志码：A
DOI: 10.11959/j.issn.1000−436x.2024002

Volume-hiding encrypted multi-map scheme based on hierarchical structure

CHEN Jing, LI Hanxing, HE Kun, JIA Meng, LI Yuqing, DU Ruiying
School of Cyber Science and Engineering, Wuhan University, Wuhan 430040, China

Abstract: The volume-hiding encrypted multi-map (EMM) scheme is able to prevent attackers from inferring plaintext through the volume leakage, but existing solutions have problems with large storage and search computational costs. Based on the characteristic that the volume of the searched data often follows Zipf's law, a hierarchical volume-hiding EMM scheme was proposed. Compared to the plain scheme that filled all keys' volumes equally, converting the filling of the overall data into the filling of multiple sub-data blocks could reduce storage costs, and also achieved constant level complexity query overhead. Security analysis shows that the proposed scheme can achieve hidden matching volume without loss of query results. Simulation results show that compared with the most efficient scheme XorMM, the proposed scheme can increase the storage overhead by 10%, reduce the search computing overhead by 90%, and significantly improve the search efficiency.

Keywords: cloud storage, searchable encryption, volume-hiding, structured encryption, hierarchical structure

---

收稿日期：2023−07−14；**修回日期：**2023−10−19
通信作者：何琨，hekun@whu.edu.cn
基金项目：国家重点研发计划基金资助项目（No.2021YFB2700200）；中央高校基本科研业务费专项资金资助项目（No.2042022kf1195, No.2042022kf0046）；国家自然科学基金资助项目（No.62076187, No.62172303）；湖北省重点研发计划基金资助项目（No.2022BAA039）；山东省重点研发计划基金资助项目（No.2022CXPT055）
Foundation Items: The National Key Research and Development Program of China (No.2021YFB2700200), The Fundamental Research Funds for the Central Universities (No.2042022kf1195, No.2042022kf0046), The National Natural Science Foundation of China (No.62076187, No.62172303), The Key Research and Development Program of Hubei Province (No.2022BAA039), The Key Research and Development Program of Shandong Province (No.2022CXPT055)

0　引言

随着云计算和云存储服务日渐成熟，大量企业与个人用户习惯将数据的计算与存储外包给云服务器。通过降低本地设备的计算与存储开销，可以构建更加灵活与高效的数据应用，但随之而来的是用户数据的隐私问题。为了防止好奇的服务器获取明文数据，简单的解决方法是将数据在本地加密后再上传至云服务器。但用户使用数据时，需要下载并解密全部的加密数据，之后才能进行检索，期间的网络带宽与计算开销显然无法应用于一般场景。因此，如何高效地从海量密文中快速定位到用户感兴趣的部分又成为新的难题。

结构化加密（STE, structured encryption）[1]技术令数据结构的拥有者可以将加密后的结构外包至不受信任的云服务器，同时还能保留原本数据结构的特定功能。加密多重映射（EMM, encrypted multi-map）作为一种特殊的 STE，允许用户对一组多重映射（MM, multi-map）结构进行加密后上传至云服务器，其中 MM 以键−值对的形式保存数据，支持用户通过键值检索所有相关的数据。

EMM 的主要应用领域包括可搜索加密和加密数据库。可搜索加密技术由 Song 等[2]提出，主要针对文本数据中的关键词检索，基于 EMM 的可搜索加密方案[3-4]可以实现次线性复杂度的查询计算开销；加密数据库则着眼于在加密数据下支持关系型数据库中常用的结构查询语言（SQL）操作，基于 EMM 的加密数据库[5]方案具有更高的安全性与查询效率。

虽然用户加密了 EMM 执行查询时的输入以及外包于服务器的数据，但是出于实用性的考虑，通常会允许交互过程泄露部分信息给服务器。在 EMM 的安全性研究中，使用泄露函数 $\mathcal{L}$ 描述方案的泄露。如果除事先确认的“可接受”的泄露外，方案没有泄露任何其他信息，就认为方案达到了预设的安全目标。常见的泄露包括搜索模式、访问模式等。一系列攻击方案[6-9]聚焦于常见的泄露，试图通过数据挖掘的思路，恢复用户的查询以及加密数据明文。攻击目标通常是恢复用户查询的键值明文，也称“查询恢复攻击”。因为一旦攻击者了解了加密文件与哪些关键词相关，则意味着加密文件的内容暴露。

最近的研究[10-13]尝试利用方案的匹配量泄露进行攻击，匹配量是指对一次查询返回的匹配结果数量。基于一些较强的假设或者特殊的场景，攻击者可以在了解方案匹配量泄露和部分先验知识的条件下恢复大部分查询对应的明文。如果能减少方案的匹配量泄露，则可以提高方案的安全性。传统的基于 ORAM 的访问模式隐藏方案[14]固然可以避免匹配量泄露，但通信与计算开销过大，难以实用；而允许部分比特泄露[15]、基于差分隐私混淆访问模式[16-17]或基于 SGX（software guard extension）[18]的方案对实际攻击的防御效果则仍需进一步评估。

为了对抗此类基于匹配量泄露的攻击，Kamara 等[19]最早提出了匹配量隐藏的 EMM 方案。匹配量隐藏要求方案的查询操作不会向服务器暴露特定查询的匹配量，具体表现为对于任意搜索，服务器观察到的匹配量都与最大匹配量 l 相等。

最朴素的匹配量隐藏 EMM 方案通过直接填充令每个键值在查询时匹配的数据个数都和匹配量最大的键值相同。这一方案的缺陷在于存储开销的空间开销为 $O(ml)$，其中 m 为可查询键值的总数。对于大规模数据，其匹配量分布通常符合齐夫分布[20-21]，即绝大多数键值的匹配量远小于最大匹配量，填充产生的额外存储开销会数十倍于原本的数据体积，因此该方法难以实用。Kamara 等[19]提出了基于伪随机变换和最密子图变换的 2 种匹配量隐藏方案，但前者不能保证结果的完整性，后者查询开销过高。Patel 等[20]提出了基于布谷鸟哈希的匹配量隐藏 EMM 方案 dprfMM，将存储开销降低至约 $2n$，其中 n 为数据个数，且该方案在查询时需要调用 $O(l)$轮可委派伪随机函数（PRF, pseudo random function）[22]和哈希操作，查询效率低；Wang 等[23]使用 Xor 过滤器[24]替换布谷鸟哈希优化方案效率，将存储开销降低至$1.23n$，但查询复杂度仍为 $O(ml)$，效率较低。

已有的匹配量隐藏 EMM 均不支持动态增删，因此本文假设用户能够在方案初始化阶段获取数据的统计信息，通常为齐夫分布。本文针对现有方案查询效率较低的问题，提出了一种针对齐夫分布数据的分层结构匹配量隐藏 EMM（HEMM, hierarchical EMM）方案。对于其中的分层设计，本文首先提出了一种基础方法，指出其存储开销偏大的问题，进一步设计了使用遗传算法（GA, genetic algorithm）确定分层结构的优化方法。改进后的 HEMM 方案能够实现对齐夫分布下数据匹配量隐藏的效果，同时能以较低的存储开销达到更优的

查询效率。本文主要贡献如下。

1) 本文提出了一种分层结构的匹配量隐藏 EMM 方案，能够在匹配结果无损且存储开销较低的情况下，实现更低的查询计算开销。

2) 本文提出了一种高效的基于遗传算法的最优分层方案查找方法，并对其进行了理论分析与实验验证。

3) 实验结果表明，相较 XorMM 方案[23]，本文提出的 HEMM 方案能够在满足已知匹配量分布的匹配量隐藏安全的前提下降低 90%的查询计算开销。

1 相关工作

云存储服务的兴起令用户可以轻松地将数据外包至服务器，从而减小本地设备的存储负担，但随之产生了用户数据的隐私安全问题。EMM 作为一种特殊的 STE，支持用户以键值的形式执行查询操作，并获取匹配的数据。EMM 可以应用于多种数据外包场景下的隐私保护方案设计，如支持高效查询的可搜索加密方案[3-4]、加密数据上的范围查询功能[25]、加密数据库中的 SQL 支持[5]等。

尽管 EMM 能够保证服务器无法直接获取用户数据与查询键值的明文数据，但服务器可以使用数据挖掘的思路分析方案的泄露信息：结合一定的先验知识，推测用户查询的键值明文，进而推测匹配数据的内容。此类攻击手段可以称为“推理攻击”[26]。Islam 等[6]利用可搜索加密方案的访问模式和搜索模式泄露，在实验中以高成功率恢复了用户查询的关键词明文。之后相似的攻击思路得到了广泛的研究[6-9,12,27]。Kellaris 等[8]提出了第一个仅根据查询匹配量泄露实现的针对加密数据库的推理攻击，但要求攻击者了解用户的查询分布以及 $O(n^4\log n)$次查询的泄露信息。Grubbs 等[12]和 Gui 等[28]对其进行改进，令攻击者不需要了解用户数据或查询分布先验知识，同时只需要观察到 $O(n^2\log n)$次查询的泄露。Blackstone 等[10]提出了 2 种针对关键词查询的基于匹配量泄露的推理攻击，令攻击者能在了解部分数据的情况下恢复大部分查询的关键词。

为了削弱此类针对匹配量泄露的攻击手段，Kamara 等[19]开始了匹配量隐藏 EMM 方案的研究，通过隐藏用户查询的匹配量泄露达到削弱相关推理攻击的目的。该方案可以分为基于伪随机变换和基于最密子图变换 2 种，其中前者着眼于查询效率，但会导致结果有损，即有一定概率不能返回全部的与键值匹配的数据；后者可以在保证结果无损的情况下实现 $O(n)$的存储开销，但查询的计算开销为 $O(l\log n)$。Patel 等[20]提出了基于布谷鸟哈希的匹配量隐藏 EMM 方案，每次查询时返回 $2l$ 个结果，利用布谷鸟哈希的碰撞特性隐藏实际的映射关系，能够在结果无损的同时实现 $O(2(1+\alpha)n)$的存储开销、$O(2l)$的查询计算开销和 $O(2l)$的查询通信开销，其中 α 是小于 1 的常数。之后 Wang 等[23]使用 Xor 过滤器[24]进一步优化方案效率，将存储开销降低至 $O(1.23n)$，查询的计算和通信开销为 $O(l)$。

本文方案针对齐夫分布数据设计，在匹配结果无损的同时，能够以与目前存储开销最优的 XorMM 方案相近的存储开销，实现常数复杂度的查询计算开销，同时查询通信开销相同，避免了带宽浪费的问题。方案开销对比如表 1 所示。

表 1 方案开销对比

方案	存储开销	通信开销	查询开销
直接填充	$O(ml)$	$O(l)$	$O(1)$
AVLH[19]	$O(n)$	$O(l\log n)$	$O(l\log n)$
dprfMM[20]	$O(2(1+\alpha)n)$	$O(2l)$	$O(2l)$
XorMM[23]	$O(1.23n)$	$O(l)$	$O(l)$
HEMM	$O(1.35n)$	$O(l)$	$O(1)$

2 匹配量隐藏的加密多重映射

本节将介绍匹配量隐藏的加密多重映射的系统模型、威胁模型、算法定义以及安全模型。

2.1 系统模型

一个匹配量隐藏的加密多重映射方案包含数据拥有者和云服务器两类实体，其系统模型如图 1 所示。

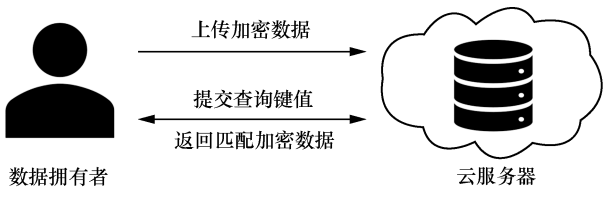


图 1 系统模型

数据拥有者同时也是数据的用户，保有一组密钥，将数据加密后外包至云服务器，同时可以向服务器提交查询键值，获取与键值匹配的加密数据。

云服务器有着丰富的存储与计算资源，向数据拥有者提供数据存储以及查询计算服务。

2.2 威胁模型

加密多重映射方案通常以半诚实的云服务器作为敌手：敌手会诚实地执行搜索协议，但会因“好奇”而利用搜索协议的信息泄露挖掘用户的隐私信息，同时可能具备用户数据集相关的先验知识。本文假设敌手能力如下。

1) 能够观察到用户与服务器间的全部交互信息。

2) 能够观察到协议执行期间的全部计算过程。

3) 能够随意访问用户存储的加密数据。

4) 了解用户数据集对不同键值匹配量的分布。

2.3 算法定义

加密多重映射的概念由常用数据结构多重映射衍生而来。

2.3.1 多重映射

MM 是一种常用的数据结构，以键值对的形式保存数据，每个键值 key 与若干数据 $\boldsymbol{v}$ 相关联，支持 Get 和 Put 这 2 种主要功能。

MM.Put (key,$\boldsymbol{v}$)：输入一组键值对，存入 MM，令数据 $\boldsymbol{v}$ 与键值 key 相关联，即 MM[key] = $\boldsymbol{v}$。

MM.Get (key) $\rightarrow \boldsymbol{v}$：输入键值 key，返回相关联的全部数据 $\boldsymbol{v}$。

2.3.2 加密多重映射

EMM 将不同类型的数据查询操作统一抽象为对一个多重映射结构的键值查询操作，并基于这一思路构建 STE 方案，将可搜索加密的应用场景从对文本数据的关键词检索拓展到了一般的数据查询操作。一个 EMM 方案可以形式化表示为初始化函数 Setup 和查询协议 Query。

$\text{Setup}(1^{\lambda},\text{MM};\perp)\rightarrow(\text{sk};\text{EMM})$：用户输入安全参数 λ 和原始数据集 MM，输出密钥 sk 和包含加密数据的 EMM，用户保存密钥 sk 并将 EMM 上传至服务器。

$\text{Query}(\text{sk},\text{key};\text{EMM})\rightarrow(\boldsymbol{v};\perp)$：用户输入密钥 sk 和要查询的键值 key，服务端输入初始化中用户上传的 EMM，最终输出一个匹配结果集合 $\boldsymbol{v}$ 发送给用户。

2.4 安全模型

在描述 EMM 方案的安全性时，使用泄露函数 $\mathcal{L}=\{\mathcal{L}_{\text{Setup}},\mathcal{L}_{\text{Query}}\}$ 表示方案向敌手泄露的信息，其中 $\mathcal{L}_{\text{Setup}}$ 和 $\mathcal{L}_{\text{Query}}$ 分别表示 Setup 和 Query 协议执行时的泄露信息。

定义 1　$\mathcal{L}$-适应性安全的 EMM[20]。给定一个 EMM 方案 Σ=(Setup, Query)，如果对于任意的半诚实概率多项式时间（PPT, probabilistic polynomial time）敌手 A，在进行 poly(λ) 次的查询游戏中，存在模拟器 S 满足

$$\left|\Pr(\text{Real}_{\text{A}}^{\Sigma}(\lambda)=1)-\Pr(\text{Ideal}_{\text{A,S},\mathcal{L}}^{\Sigma}(\lambda)=1)\right|\leqslant\text{negl}(\lambda)\quad(1)$$

其中，negl(λ) 为可忽略函数，则称该 EMM 方案是 $\mathcal{L}$-适应性安全的。概率游戏 $\text{Real}_{\text{A}}^{\Sigma}(\lambda)$ 和 $\text{Ideal}_{\text{A,S},\mathcal{L}}^{\Sigma}(\lambda)$ 的定义如下。

$\text{Real}_{\text{A}}^{\Sigma}(\lambda)$。敌手 A 任意选择一组 MM 输入 Setup 函数得到对应的 EMM，之后 A 自适应性地指定 poly(λ) 个查询；对于每个指定的查询，挑战者 C 与 A 之间执行 Query 协议；最后 A 返回结果 $b\in\{0,1\}$。

$\text{Ideal}_{\text{A,S},\mathcal{L}}^{\Sigma}(\lambda)$。由模拟器 S 根据 $\mathcal{L}_{\text{Setup}(\lambda,\text{MM})}$ 生成 EMM 后发送给敌手 A；针对 A 指定的一系列查询 $\text{key}_1,\cdots,\text{key}_{\text{poly}(\lambda)}$，查询结果由模拟器 S 根据 $\mathcal{L}_{\text{Query}(\lambda,\text{MM},\text{key}_1,\cdots,\text{key}_{\text{poly}(\lambda)})}$ 生成；最后 A 返回结果 $b\in\{0,1\}$。

EMM 中通常被认为可接受的泄露信息包括以下几种。

1) 搜索模式。该泄露使敌手能够确认两次查询操作是否对应相同 key。对于用户执行的 q 个查询 $\text{key}_1,\cdots,\text{key}_q$，产生的搜索模式泄露可以表示为一个 $q\times q$ 大小的矩阵 **sp**，当且仅当 $\text{key}_i=\text{key}_j$ 时，**sp**[i][j]=1。

2) 数据总量。MM 中存储的全部 key 对应的匹配量之和，即键值对数据总数，用 n 表示。

3) 键值数量。MM 中可以用于查询的 key 的个数，用 m 表示。

4) 匹配量。该泄露使敌手能够了解到与一次查询实际相关的数据个数，对于查询操作 MM[key]=$\boldsymbol{v}$，匹配量泄露表示为 $l(\text{key})=|v|$。

5) 最大匹配量。MM 中全部 key 匹配量的最大值，用 $l=\max\left|\text{MM}[\text{key}_i]\right|_{i=1,2,\cdots,m}$ 表示。

如果一个 EMM 的泄露函数中不包含特定查询的匹配量信息，而只泄露最大匹配量，则该泄露函数满足匹配量隐藏泄露函数的定义，对该定义的描述首先要定义一个概率游戏 $\text{Game}_{\eta}^{\text{A},\mathcal{L}}(n,m,l,D)$。

$\mathrm{Game}_{\eta}^{\mathrm{A},\mathcal{L}}(n,m,l,D)$。敌手 A 选择两组数据 MM_0 和 MM_1 发送给挑战者 C，满足以下条件。

1) $\Sigma_{\mathrm{key}\in\mathrm{MM}_0} l(\mathrm{key}) = \Sigma_{\mathrm{key}\in\mathrm{MM}_1} l(\mathrm{key}) = n$。

2) $\left|\{\mathrm{key}\}_{\mathrm{key}\in\mathrm{MM}_0}\right| = \left|\{\mathrm{key}\}_{\mathrm{key}\in\mathrm{MM}_1}\right| = m$。

3) $\max_{\mathrm{key}\in\mathrm{MM}_0} l(\mathrm{key}) = \max_{\mathrm{key}\in\mathrm{MM}_1} l(\mathrm{key}) = l$。

4) $\{l(\mathrm{key})\}_{\mathrm{key}\in\mathrm{MM}_0} \sim D$ 且 $\{l(\mathrm{key})\}_{\mathrm{key}\in\mathrm{MM}_1} \sim D$。

挑战者收到数据后，选取 MM_{η} 并计算 $\mathcal{L}_{\mathrm{Setup}}(\mathrm{MM}_{\eta})$ 发送给敌手；之后敌手适应性地选取键值 $\mathrm{key}_1,\cdots,\mathrm{key}_{\mathrm{poly}(\lambda)}$，对其中每个 key_i，挑战者计算 $\mathcal{L}_{\mathrm{Query}(\mathrm{MM}_{\eta},\mathrm{key}_1,\cdots,\mathrm{key}_i)}$ 发给敌手；最终敌手输出 $b\in\{0,1\}$。

将敌手在游戏 $\mathrm{Game}_{\eta}^{\mathrm{A},\mathcal{L}}(n,m,l,D)$ 中输出 1 的概率记为 $p_{\eta}^{\mathrm{A},\mathcal{L}}(n,m,l,D)$，有如下定义。

定义 2 匹配量隐藏泄露函数[20]。当且仅当对于任意敌手 A，任意 $n\geqslant l\geqslant 1$、任意 $m\geqslant 1$ 和任意分布 D 满足

$$p_0^{\mathrm{A},\mathcal{L}}(n,m,l,D) = p_1^{\mathrm{A},\mathcal{L}}(n,m,l,D) \tag{2}$$

则称一个 EMM 方案 Σ=(Setup, Query)的泄露函数 $\mathcal{L}$ 是匹配量隐藏泄露函数。

定义 3 匹配量隐藏 EMM[20]。当且仅当一个 EMM 方案 Σ=(Setup, Query)的泄露函数 $\mathcal{L}=\{\mathcal{L}_{\mathrm{Setup}},\mathcal{L}_{\mathrm{Query}}\}$ 满足：1) Σ 是一个 $\mathcal{L}$−适应性安全的 EMM 方案；2) $\mathcal{L}$ 是一个匹配量隐藏泄露函数，则称该方案是匹配量隐藏的。

3 HEMM 方案设计

本节详细介绍了 HEMM 方案设计。首先介绍了方案的设计思路以及一个基础方案，然后提出了一个基于遗传算法的分层方法。

3.1 设计思路

本文方案的设计目标是在实现匹配量隐藏的同时，以较低的存储开销实现更高的查询效率。现有的基于布谷鸟哈希或 Xor 过滤器的方案虽然对敌手隐藏了匹配量，但也导致服务器需要对每次查询执行 l 次哈希计算，耗时较高。

本文方案的解决思路是采用分层结构替代哈希计算，对每次查询只需要计算一次就可以定位到相关数据的位置，查询开销较小。同时根据数据满足齐夫分布的特性对 MM 中不同键值关联的数据进行层次划分，将朴素填充方案中全部键值的匹配量填充为 l 变为将不同分层中的数据分别填充至相同数量，以此减小总体的空间开销。

齐夫定律最开始出现于自然语言研究中，是表示语料库数据分布的实验定律，可以指出关键词在文本中的出现频率服从幂律分布。它可以表述为在自然语言语料库里，一个单词出现的频率与它在频率表里的排名成反比。之后这一定律被发现同样适用于非语言学场景下的大规模数据[14,16]，可以作为数据分布中的通用假设。在查询场景下，如果一个 MM 符合齐夫分布 $\mathcal{Z}_{a,b}$，则其中出现频率从高到低排名第 r 的 key 对应的匹配量为

$$\frac{r^{-b}}{H_{a,b}}n \tag{3}$$

其中，$H_{a,b}$ 是调和级数，即

$$H_{a,b} = \sum_{i=1}^{a} i^{-b} \tag{4}$$

层次划分是指将每个键值对应的一组加密数据分为若干个不相交子集，简称为片段；之后将这若干个片段放入不同的层中，保证同一层中不会包含 2 个对应同一键值的加密数据片段；每一层中放置片段的位置通过随机生成；将每层中与一个键值相关的数据片段收集起来即可恢复出键值对应的全部加密数据。如果令每个键值都可以在每一层以服务器不可区分的方式查找到一个对应的片段，虽然这个片段中的数据可能与该键值并不匹配，那么服务器就不能获取到键值的匹配量信息，因此能够实现匹配量隐藏的目的。

3.2 基础方案

本文符号表示如下。对于一个集合 S，用 $|S|$ 表示其中元素的个数，用 $s\leftarrow S$ 表示从集合中随机抽取元素。数据集中的数据总量表示为 n，共包含 m 个可用于查询的键值 $\boldsymbol{K}=\{\mathrm{key}_1,\cdots,\mathrm{key}_m\}$，与特定键值关联的数据表示为 $\boldsymbol{v}$，键值对数据的集合表示为 MM，其中所有键值匹配量的最大值记为 $l=\max\left|\mathrm{MM}[\mathrm{key}_i]\right|_{i=1,2,\cdots,m}$；算法中使用 F 表示伪随机函数 PRF，Enc/Dec 表示满足 IND-CPA 安全的对称加密算法的加解密操作。

通过表 2 所示的一组样例数据处理，来展示基础方案中层次划分的具体思路。

表 2　　样例数据

键值	相关数据
key_1	v_1, v_2, v_3, v_4, v_5
key_2	v_6, v_7
key_3	v_8, v_9, v_{10}, v_{11}
key_4	v_{12}

假设一个数据集中键值与数据之间的对应关系如表 2 所示，依照本文的思路，对数据划分后可以生成如图 2 所示的逻辑结构。其中，圆角方块表示一个分层结构 Level，一个 Level 中的方块表示若干个数据片段 f，每个片段中可以保存若干个对应同一个键值 key 的数据；同一 Level 下的各个片段大小相同，即保存的数据个数相同，可能存在通过填充补齐的情况；一个 Level 中每个片段能容纳的数据数量也称为当前 Level 的高度 height。本文方案的整体思路是将每个键值相关的数据分为若干个数据片段，随机地分散在从下至上的若干个连续 Level 中，因此下层的 Level 会包含更多的片段，最下层 Level 包含的片段个数等于关键词总数；每次查询时，服务器根据用户传来的加密键值从每个 Level 中取一个片段加入结果集合中，最后将结果集返回给用户，因此每次查询操作得到的结果个数均为 l，能够达到匹配量隐藏的目的。对于包含查询对应数据的 Level，返回的片段内容是有意义的；对于上层 Level 不包含对应数据的情况，本文方案也会在初始化的时候在这一 Level 中随机规定一个固定的片段与查询对应。由于同一 Level 中片段的位置随机分布，因此对于不同的键值查询，每个 Level 中的片段选取互相独立且同分布，即实际匹配量不同的查询对应的结果分布相同，此时服务器就不能区分不同的键值实际对应的数据个数。由于采用了分层结构，相对于朴素地将全部键值的匹配量对齐为最大长度，HEMM 只需要在每层 Level 中进行少量的填充即可，因此节约了空间开销。

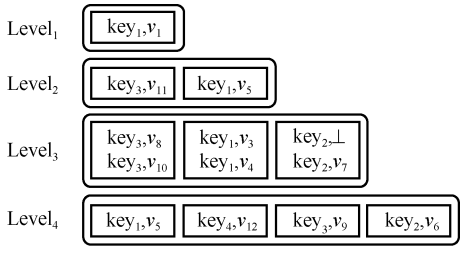


图 2　样例数据分层结构

HEMM 的主要算法包括初始化函数 Setup 和查询协议 Query。

$\text{Setup}(1^\lambda, \text{MM}; \perp) \to (\text{sk}; \text{EMM})$。初始化算法具体内容如算法 1 所示，输入安全参数 λ 和 MM。输出密钥 sk 和 EMM，其中 EMM 包含 PosSet 和分层结构 H 两部分，PosSet 中记录了不同键值对应片段的位置信息，可供之后查询时使用。首先根据 λ 生成所需密钥，再从 MM 中提取全部的键值形成列表 $\boldsymbol{K}$。之后循环通过 NextHeight 函数获取下一 Level 的高度 height，该函数初始版本如算法 2 所示；对每一 Level 进行填充时，先随机地打乱 $\boldsymbol{K}$ 中的键值顺序，按照打乱后的顺序遍历其中的键值 key_i，i 表示该键值在列表中所处的位置；构造片段 f 时，循环将 $\text{MM}[key_i]$的数据用 K_e 加密后放入，如果数据不足则使用随机数据，直到将片段中的数据个数填充至 height，其中，pop 表示返回列表末尾的元素并删除原本列表中的数据，push 表示顺序地向列表的末尾追加数据；之后对该片段的位置加密后放入结构 PosSet，使用计数器保证相同的位置信息有不同的密文值；不断将片段组合成 Level 结构，再将多个 Level 组合为分层结构 H；最后将 PosSet 和 H 组合为 EMM 保存在服务器，用户保留密钥 sk 用于后续的查询操作。

算法 1　初始化算法 Setup

输入　1^λ，MM

输出　密钥 sk，EMM

1) $K_e \coloneqq \text{Gen}(1^\lambda), K_d \leftarrow \{0,1\}^\lambda$;
2) $\text{PosSet} \coloneqq \{\}; H \coloneqq \{\}; \text{cnt} \coloneqq 0$;
3) $\text{st} \coloneqq \varnothing$;
4) $\boldsymbol{K} \coloneqq \{\text{key}\}_{\text{key} \in \text{MM}}$;
5) $\text{height} \coloneqq \text{NextHeight}(\text{st}, \text{MM})$;
6) while height $\neq \perp$:
7)　Level:={};
8)　随机打乱 $\boldsymbol{K}$ 中键值的顺序;
9)　for key_i in $\boldsymbol{K}$:
10)　$f \coloneqq \{\}$;
11)　$\boldsymbol{v} \coloneqq \text{MM}[key_i]$;
12)　for $j \coloneqq 1$ to height:
13)　if $\boldsymbol{v} \neq \varnothing$:
14)　val $\coloneqq \boldsymbol{v}$.pop();
15)　else val := 随机数据;
16)　end if

17) f.push(Enc(K_e, val || cnt));
18) end for
19) ek := F_{K_d}(key$_i$ || 1);
20) fk := F_{K_d}(key$_i$ || 0);
21) PosSet[ek]∪ = {Enc(fk, i || cnt)};
22) cnt = cnt + 1;
23) Level.push(f);
24) if $\boldsymbol{v} = \varnothing$:
25) $\boldsymbol{K}$.pop(key$_i$);
26) end if
27) H.push(Level);
28) height = NextHeight(st, MM);
29) end for
30) end while
31) sk := {K_e, K_d};
32) EMM := {PosSet, H};
33) return {sk; EMM};

NextHeight(st, MM) → height。算法 1 生成结构 H 时会多次调用 NextHeight 函数获取下一 Level 高度，初始方案如算法 2 所示。状态 st 为空则为初次调用；初始化时将 MM 全部键值的匹配量按升序保存至列表 **vol**，c 为哨兵值，记录上一次对 **vol** 遍历到的位置；H 结构为空时，第一层 Level 高度为 **vol** 中的首个元素，即 MM 中最小匹配量；之后每次调用会通过步骤 12)～步骤 16)找到下一个不同的匹配量，减去调用开始时第 c 个匹配量得到下一层高度；遍历结束则返回 ⊥。该算法可以描述为：假设 MM 有 x 个不同的匹配量，$x \leqslant m$，可划分出 x 个层次，此时结构 PosSet 的空间复杂度为 $O(m^2)$，结构 H 的空间复杂度为 $O(n)$，EMM 的总体空间开销为 $O(n+m^2)$。在键值个数 m 远小于 n 的情况下，可以视为 $O(n)$，但对于一般性情况，$O(n+m^2)$ 的空间开销仍过高，需要设计更优的层次划分方法。

算法 2 NextHeight
输入 状态 st，MM
输出 下一层高度 height

1) if st = ∅:
2) **vol** := {}; c := 0;
3) for key, $\boldsymbol{v}$ in MM:
4) **vol**.push(| $\boldsymbol{v}$ |);
5) end for
6) sort(**vol**); // 从小到大排序
7) st = {**vol**, c};
8) end if
9) {**vol**, c} := st;
10) height := ⊥;
11) if $c = 0$:
12) height = **vol**[0];
13) else
14) while c+1<|**vol**|:
15) if **vol**[c+1]=**vol**[c]:
16) $c = c + 1$;
17) else break;
18) end if
19) end while
20) height = **vol**[$c+1$] − **vol**[c];
21) end if
22) $c = c + 1$;
23) return height;

Query(sk,key; EMM) → ($\boldsymbol{v}$; ⊥)。查询协议如算法 3 所示，用户输入密钥 sk 和查询键值 key，生成陷门 trapdoor 后发送给服务器；服务器用陷门中解析的密钥从 PosSet 获取对应一组片段位置，即每个 Level 中与键值关联的片段下标集合；算法中变量 epos$_j$ 指的是 PosSet[ek]中第 j 个元素，解密后的值表示 H 中第 j 个 Level 中当前检索对应数据片段的位置；之后服务器根据这些下标获取所有的片段组合为加密结果集 **ev** 返回给用户，用户使用本地密钥 sk 对 **ev** 解密后，即可得到查询的结果 $\boldsymbol{v}$。

算法 3 查询协议 Query
输入 sk，key，EMM
输出 $\boldsymbol{v}$，⊥

用户：
1) (K_e, K_d):=sk;
2) ek := F_{K_d}(key || 1);
3) fk := F_{K_d}(key || 0);
4) trapdoor := {ek, fk};
5) 将 trapdoor 发送给服务器;

服务器：
6) (PosSet, H) := EMM;
7) (ek, fk) := trapdoor;
8) **ev** := {};
9) for epos$_j$ in PosSet[ek]:
10) (i || cnt) := Dec(fk, epos$_j$);

11)　$\mathbf{ev} \cup = \{f_i \text{ in Level}_j\}$;

12) end for

13) 将 **ev** 发送给用户;

用户：

14)使用 K_e 对 **ev** 解密后得到 $\boldsymbol{v}$;

3.3　基于遗传算法的分层方法

本文使用遗传算法进行最优分层方案中参数的求解。遗传算法是一种借鉴生物进化规律的启发式算法，被广泛应用于数值优化、组合优化等优化问题。

本文的优化思路为使用遗传算法权衡 Level 结构的数量，减少分层个数就能缩小 PosSet 结构的空间开销，但会导致 H 结构中需要额外填充，用于补齐实际长度小于 Level 高度的片段，如图 3 所示。本文尝试找到一个使整体开销最优的平衡点。

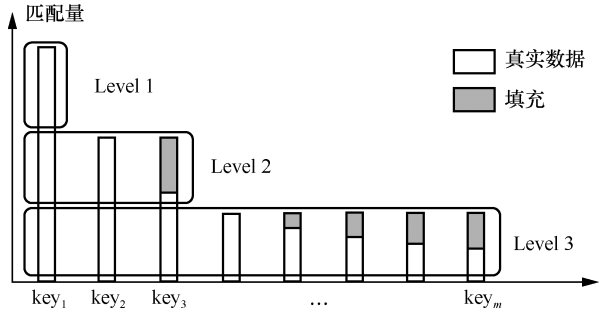


图 3　齐夫分布下数据的分层与填充

考虑划分 Level 时，首先将键值按照各自匹配量从大到小排序，在本文的研究场景中，这组数据符合齐夫分布 $\mathcal{Z}_{a,b}$；如果要对全部的数据划分出 k 个 Level，相对于在一组齐夫分布数据上选取 k–1 个断点，其中第 $i(1\leqslant i<k)$个断点对应排序中第 x_i 个位置的数据，Level_i 的高度为 $l(\text{key}_{x_i-1})-l(\text{key}_{x_i})$，其中数据片段的个数为 MM 中匹配量大于 $l(\text{key}_{x_i})$ 的键值的个数。此时结构 H 的空间开销为

$$\left(x_1\frac{1}{H_{m,b}}+x_2^{-b}\frac{x_2-x_1}{H_{m,b}}+\cdots+x_{k-1}^{-b}\frac{m-x_{k-1}}{H_{m,b}}\right)l \tag{5}$$

将式(3)中 l 的系数记为 X，考虑此时固定划分 k 个层次，因此方案整体空间复杂度为

$$O(Xl+mk) \tag{6}$$

此时可以转化为优化问题：对于一个包含 m 个键值的 MM，选择一组数据 $\{k,x_1,\cdots,x_{k-1}\}$ 使式(5)的值达到最小，其中 $1\leqslant x_1\leqslant\cdots\leqslant x_{k-1}<m$。显然，每增加一个分层，都会单调地减少结构 H 的填充开销；同时从直觉上看，随着分层数量不断增加，每新增一个分层带来的收益会快速减少，因此可以判断只需要少量的分层就可以让 H 的空间开销快速下降到一个可接受的范围，进而达到整体最优。

之后使用遗传算法对系数 X 进行最优值求解，目标函数即式(5)，令 b=1，得到适应度函数为

$$\text{fitness}_{x,k,m}=\left(x_1+\cdots+\frac{x_{k-1}-x_{k-2}}{x_{k-2}}+\frac{m-x_{k-1}}{x_{k-1}}\right)^{-1} \tag{7}$$

适应度越高，目标函数值越小，对应的存储开销越小。对于不同的数据键值个数和结构分层数，实验中迭代得到的系数 X 最优值如表 3 所示。第一行的 m 个分层表示无填充下的期望开销，可以看出只需要常数级的分层，就能将填充后 H 的存储开销下降为接近无填充的状态，当键值个数为 1×10^5 时，只需要 20 层 Level 就可以将存储开销控制为 1.35n；当键值个数增加到 1×10^7 时，20 层索引下的存储开销也只增长到约 1.52n。考虑到百万级别的键值个数已经足够满足大多数场景，因此本文默认采用 20 层的常数 Level 划分，即默认采用 k=20。

表 3　系数 X 在不同键值个数和分层数下的最优值

分层数	键值个数				
	1×10^3	5×10^3	3×10^4	1×10^5	1×10^7
m	7.49	9.09	10.90	12.09	16.69
3	19.51	30.65	49.64	68.13	221.93
5	13.98	19.82	28.46	35.88	83.07
10	10.63	13.89	18.13	21.33	37.67
15	9.71	12.23	15.31	17.82	28.81
20	9.26	11.54	14.36	16.35	25.33
25	9.07	11.20	14.05	15.63	23.35
30	8.94	10.96	13.40	15.17	22.21
40	8.80	10.70	12.95	14.54	20.97

使用 GA 表示遗传算法对最优分层方法的求解过程，具体步骤如算法 4 所示。实验中每一轮迭代的种群规模固定包含 50 个个体，每个个体的值最初为随机生成，即不重复的 2～m 中的 k 个值；经过实验验证，固定迭代轮次为 100，即可实现与 10 000 轮迭代近似的结果，每一轮中将个体两两交叉，并以 10%的概率发生变异，即随机修改个体 p 中的一个 x 值，同时要求每次交叉和变异产生的新数据要符合 x 值互不相等并且不影响原本的大小顺序；之后只保留适应度前 50 的个体，并淘汰其他数据个体，最终返回最后一轮迭代中适应度最好的个体。

算法 4　分层求解算法 GA

输入　键值数 m，分层数 k

输出 最优个体 **bp**

1) 种群 $P \coloneqq \{p_i\}_{i=1,2,\cdots,50}$;
2) for p_i in P:
3) $p_i = \{x_1, x_2, \cdots, x_{k-1}\} \leftarrow \{2,3,\cdots,m\}$;
4) 满足 $x_1 < x_2 < \cdots < x_{k-1}$
5) end for
6) for round $\coloneqq$ 1 to 100: // 100 轮迭代
7) for $(p_i, p_j)_{i \neq j}$ in P:
8) $np = p_i p_j$; // 交叉
9) if rand[0,1]<0.1:
10) 随机改变 np 中一个 x 值;
11) end if
12) P.push(np);
13) end for
14) sort p in P with fitness$_{p,k,m}$; // 对 P 中个体按适应度从大到小进行排序
15) $P = P[0:50]$; // 保留适应度最大的 50 个个体
16) end for
17) **bp** $\coloneqq P$.top;
18) return **bp**;

采用这一思路，可得优化后的 NextHeight'函数如算法 5 所示。在第一次调用的初始化阶段通过 GA 得到一组最优的分层选点；reverse 函数修改列表元素为逆序；哨兵 c 为 0 时会选取 **vol** 中排在第 $\boldsymbol{x}[0]$个的匹配量作为下一层高度，否则返回 **vol** 中排在第 $\boldsymbol{x}[c]$与第 $\boldsymbol{x}[c-1]$个匹配量的差值。

算法 5 NextHeight'

输入 状态 st，MM

输出 下一层高度 height

1) if st $= \varnothing$:
2) $m \coloneqq$ MM 键值数;
3) $\boldsymbol{x} \coloneqq$ GA(m,20);
4) reverse($\boldsymbol{x}$);
5) **vol** $\coloneqq \{\}$; $c \coloneqq 0$;
6) for key,$\boldsymbol{v}$ in MM:
7) **vol**.push($|\boldsymbol{v}|$);
8) end for
9) sort(**vol**);// 从小到大排序
10) st $= \{$**vol**,$\boldsymbol{x}$,$c\}$;
11) end if
12) $\{$**vol**,$\boldsymbol{x}$,$c\} \coloneqq$ st;
13) height $\coloneqq \perp$;
14) if $c<|\boldsymbol{x}|$:
15) if $c = 0$ height = **vol**[$\boldsymbol{x}$[0]];
16) else height = **vol**[$\boldsymbol{x}$[c]] − **vol**[$\boldsymbol{x}$[$c-1$]];
17) end if
18) end if
19) $c = c + 1$;
20) return height;

4 安全性分析

本节给出 HEMM 方案的安全性分析，最终证明与目前最新的 XorMM 相比，HEMM 仅增加了键值个数和键值匹配量分布的信息泄露。

首先考虑方案的泄露函数。在 Setup 函数阶段，服务器收到加密后的 EMM，包括 H 和 PosSet 这 2 个结构；服务器可以从结构 PosSet 中分析得到可查询键值的数量 m 和 MM 的最大匹配量 l，从结构 H 中学习 MM 中键值对数据的个数 n、键值个数 m，以及不同键值匹配量的分布 $\mathcal{Z}$；在查询协议 Query 执行阶段，服务器可以得到加入噪声后匹配量保持为 l 的搜索模式 sp，其中每次查询从 PosSet 获取的位置信息均表现为一组随机排列。综上，HEMM 的泄露函数可以表示为 $\mathcal{L}_{\text{HEMM}} = \{(m,l,n,\mathcal{Z}),\text{sp}\}$。

定理 1 HEMM 是满足 $\mathcal{L}_{\text{HEMM}}$−适应性安全的 EMM 方案。

证明 可以针对方案的 Setup 函数和 Query 协议以如下方式构造高效模拟器 S

$$\text{S.SimSetup}(1^{\lambda},(m,l,n,\mathcal{Z})) \rightarrow (\text{state};\text{EMM})$$

首先构造一个空的结构 H，包含 20 层 Level 结构，其中最下层 Level 包括 m 个片段；根据匹配量分布 $\mathcal{Z}$ 向 H 中填入 n 个数据，满足从全部 Level 中任取一个片段并组合后均包含 l 个数据，其中的数据通过均匀随机生成得到；构造 PosSet 时，对每次查询匹配的片段位置使用随机选择的方法生成，且优先选择还未被选取过的片段，保证所有片段的出现概率相等且不为 0，加密位置信息时使用随机生成的密钥，保存在 state 中；最后将 2 个结构组合为 EMM，返回 (state;EMM)。

$\text{S.SimQuery}(1^{\lambda},\text{sp}(\text{key}_i)) \rightarrow (\text{state};\text{res})$。通过搜索模式泄露 $\text{sp}(\text{key}_i)$，可以判断是否存在键值 $\text{key}_j = \text{key}_i (j<i)$ 已被查询过，如果未被查询过，则从 state 中随机选取一个密钥与当前查询绑定，使用改名密钥解析 PosSet 中对应的位置信息，从 H

中找到这组位置对应的片段 $(f_1,\cdots,f_{20})$ 保存至 state[key$_i$]；之后返回(state; res=state[key$_i$])。

在 $\mathrm{Ideal}^{\Sigma}_{\mathrm{A,S},\mathcal{L}}(\lambda)$ 中，使用模拟器 S 生成返回结构，基于对称加密算法 Enc 的安全性，模拟器生成的 EMM 与 $\mathrm{Real}^{\Sigma}_{\mathrm{A}}(\lambda)$ 中的 EMM 不可区分；在查询阶段，由于相同查询匹配的片段位置由模拟器随机生成，与 $\mathrm{Real}^{\Sigma}_{\mathrm{A}}(\lambda)$ 中片段位置的生成过程相同，因此不可区分；综上，$\mathrm{Ideal}^{\Sigma}_{\mathrm{A,S},\mathcal{L}}(\lambda)$ 与 $\mathrm{Real}^{\Sigma}_{\mathrm{A}}(\lambda)$ 对敌手不可区分，因此 HEMM 满足 $\mathcal{L}_{\mathrm{HEMM}}$ −适应性安全。证毕。

定理 2　泄露函数 $\mathcal{L}_{\mathrm{HEMM}}=\{(m,l,n,\mathcal{Z}),\mathrm{sp}\}$ 是一个匹配量隐藏泄露函数。

证明　考虑 2 个 MM，其有相同的键值数 m、最大匹配量 l、数据总量 n 且匹配量符合同分布 $\mathcal{Z}$，则 2 个 MM 在方案中 Setup 阶段的泄露函数相等；由于查询阶段的搜索模式泄露与输入的 MM 无关，在游戏 $\mathrm{Game}^{\mathrm{A},\mathcal{L}}_0(n,m,l,\mathcal{Z})$ 和 $\mathrm{Game}^{\mathrm{A},\mathcal{L}}_1(n,m,l,\mathcal{Z})$ 中，不同 MM 产生的输入相等，因此产生相同输出的概率相等。综上，泄露函数 $\mathcal{L}_{\mathrm{HEMM}}=\{(m,l,n,\mathcal{Z}),\mathrm{sp}\}$ 是一个匹配量隐藏泄露函数。证毕。

定理 3　HEMM 是匹配量隐藏的 EMM 方案。

证明　基于定理 2 和定理 3，可得 HEMM 是一个匹配量隐藏的 EMM 方案。证毕。

5　仿真实验

本文设计了齐夫分布下匹配量隐藏 EMM 方案，考虑自然语言天然符合齐夫定律[21]，采用 Enron 数据集提取关键词生成的 MM 作为测试数据集。实验主要将本文方案与 dprfMM 方案[20]和 XorMM 方案[23]进行对比。硬件环境与软件环境如下：CPU 为 AMD Ryzen 7 5800H，主频为 3.2 GHz，运行内存为 16 GB，操作系统为 64 位 Windows 10 操作系统下的 WSL1 中的 Ubuntu 20.04，编程语言为 C++；实验中使用的伪随机函数和对称加密函数基于开源软件库 OpenSSL 1.1.1 中的国密算法 SM3 和 SM4 实现。

首先在考虑停用词的情况下，对全部文件进行关键词提取；每轮实验随机抽取一定数量的文件构造 MM 作为方案输入；实验主要对比了在不同规模数据量下，3 种方案的初始化时间、存储开销以及查询计算开销，并验证了不同分层数对本文方案分层算法的运行效率以及存储开销的影响。

初始化时间如图 4 所示。dprfMM 方案和 XorMM 方案中初始化阶段的主要开销产生于哈希结构的构建过程，会通过多次哈希运算确认数据存储位置，计算开销较大；HEMM 方案初始化阶段的主要开销产生于数据加密以及通过遗传算法查找最优分层方法的过程，实验中设置每一轮保留 50 个数据个体，迭代 100 轮即可使结果与 10 万轮迭代只相差 5%以内，可以认为找到了最优解，因此不需要大量计算。从实验结果可以发现，HEMM 的初始化时间大约为 dprfMM 初始化时间的 10%，XorMM 初始化时间的 17%。

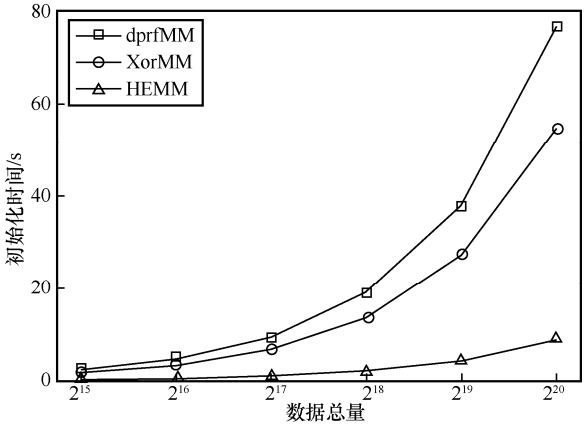


图 4　初始化时间

在不同数据量下，HEMM 方案分层算法运行时间如图 5 所示。从图 5 可以看出，分层算法作为初始化算法的主要部分之一，未出现计算瓶颈。

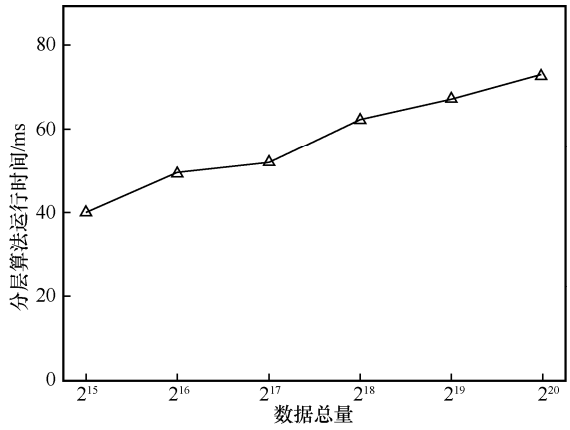


图 5　HEMM 方案分层算法运行时间

存储开销对比如图 6 所示。方案中的被查询加密数据统一为 128 bit，用 Base64 编码后落盘的文件大小即方案的存储开销。方案的存储开销均与数据总量 n 保持线性增长，HEMM 的存储开销基本保持为 dprfMM 的 60%，但比 XorMM 增加大约 10%，小于理论结果，分析后确认原因是实际数据分布 $\mathcal{Z}_{a,b}$ 中的参数 b<1，说明实际数据的分布特征对方案的存储开销有一定的影响。

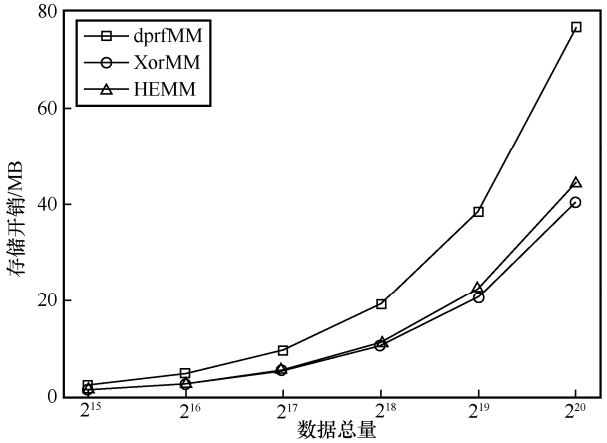


图 6　存储开销对比

HEMM 方案分层数对存储开销的影响如图 7 所示。实验中取 10 万个文件在不同分层数下的平均存储开销，从图 7 可以看出，超过 20 层的分层已经几乎不能再起到减少存储开销的效果了。

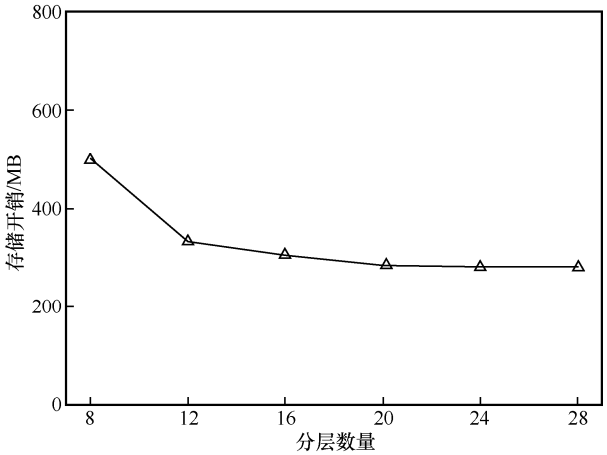


图 7　HEMM 方案分层数对存储开销的影响

不同最大匹配量和不同数据总量下的查询时间如图 8 所示。对于 HEMM，查询任意键值时的计算复杂度相等，且只与最大匹配量线性相关，查询时不需要执行多轮哈希寻址，开销为 $O(1)$，相对 dprfMM 和 XorMM 方案的查询开销 $O(l)$提升明显。从图 8 中可以看出，在控制最大匹配量相等和数据总量相等的 2 种情况下，HEMM 的查询计算时间相对 dprfMM 和 XorMM 均减少 90%以上。

6　结束语

本文基于数据匹配量服从齐夫分布的通用假设，构造了一种分层结构的匹配量隐藏加密多重映射方案，支持常数复杂度的查询开销，并设计了基于遗传算法的改进分层方法。通过令各个分层中需填充数据总量的期望值尽可能小，降低了方案的存储开销。通过安全性分析，证明了本文所提方案满足匹配量隐藏 EMM 的定义。与其他方案的实验对比表明，本文方案具有较低的存储开销和初始化时间，查询性能相对已有方案优势明显。

参考文献：

[1] CHASE M, KAMARA S. Structured encryption and controlled disclosure[C]//International Conference on the Theory and Application of Cryptology and Information Security. Berlin: Springer, 2010: 577-594.

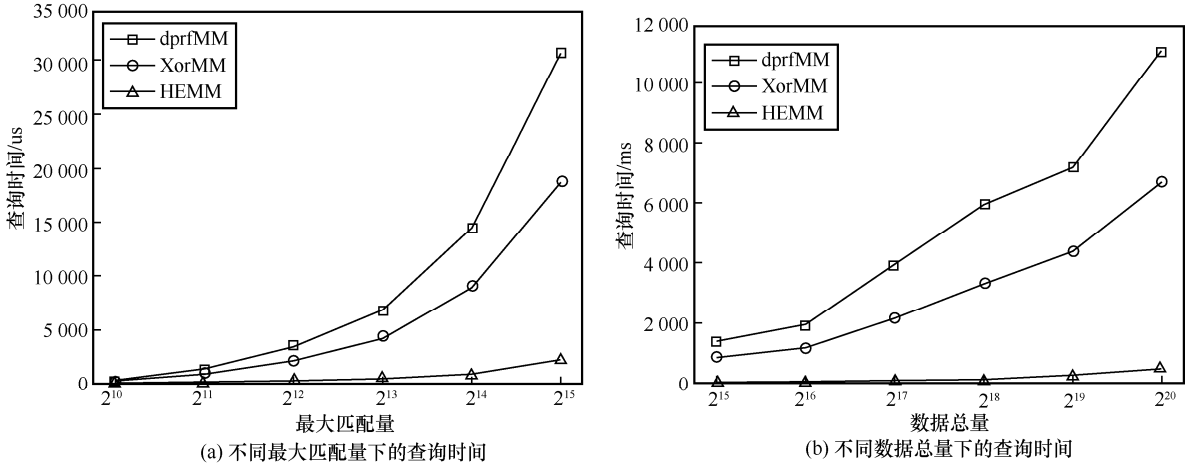
[2] SONG D X, WAGNER D, PERRIG A. Practical techniques for searches on encrypted data[C]//Proceedings of IEEE Symposium on Security and Privacy. Piscataway: IEEE Press, 2002: 44-55.

[3] CURTMOLA R, GARAY J, KAMARA S, et al. Searchable symmetric encryption: improved definitions and efficient constructions[C]//Proceedings of the 13th ACM conference on Computer and communications security. New York: ACM Press, 2006: 79-88.

[4] KAMARA S, MOATAZ T, OHRIMENKO O. Structuredencryption and leakage suppression[C]//Annual International Cryptology Conference. Cham: Springer, 2018: 339-370.

[5] KAMARA S, MOATAZ T. SQL on Structurally-encrypted databases[C]//International Conference on the Theory and Application of Cryptology and Information Security. Cham: Springer, 2018: 149-180.

[6] ISLAM M S, KUZU M, KANTARCIOGLU M. Access pattern disclo-



(a) 不同最大匹配量下的查询时间　　(b) 不同数据总量下的查询时间

图 8　不同最大匹配量和不同数据总量下的查询时间

sure on searchable encryption: ramification, attack and mitigation[C]//Network and Distributed System Security Symposium.Piscataway: IEEE Press, 2012: 1-15.

[7] CASH D, GRUBBS P, PERRY J, et al. Leakage-abuse attacks against searchable encryption[C]//Proceedings of the 22nd ACM SIGSAC Conference on Computer and Communications Security. New York: ACM Press, 2015: 668-679.

[8] KELLARIS G, KOLLIOS G, NISSIM K, et al. Generic attacks on secure outsourced databases[C]//Proceedings of the ACM SIGSAC Conference on Computer and Communications Security. New York: ACM Press, 2016: 1329-1340.

[9] OYA S, KERSCHBAUM F. Hiding the access pattern is not enough: exploiting search pattern leakage in searchable encryption[J]. arXiv Preprint, arXiv: 2010.03465, 2020.

[10] BLACKSTONE L, KAMARA S, MOATAZ T. Revisiting leakage abuse attacks[C]//Network and Distributed System Security Symposium. Piscataway: IEEE Press, 2020: 1-18.

[11] PODDAR R, WANG S, LU J N, et al. Practical volume-based attacks on encrypted databases[C]//Proceedings of 2020 IEEE European Symposium on Security and Privacy (EuroS&P). Piscataway: IEEE Press, 2020: 354-369.

[12] GRUBBS P, LACHARITE M S, MINAUD B, et al. Pump up the volume: practical database reconstruction from volume leakage on range queries[C]//Proceedings of the 2018 ACM SIGSAC Conference on Computer and Communications Security. New York: ACM Press, 2018: 315-331.

[13] GUI Z C, PATERSON K G, PATRANABIS S. Rethinking searchable symmetric encryption[C]//Proceedings of 2023 IEEE Symposium on Security and Privacy (SP). Piscataway: IEEE Press, 2023: 1401-1418.

[14] GARG S, MOHASSEL P, PAPAMANTHOU C. TWORAM: efficient oblivious ram in two rounds with applications to searchable encryption[C]//Annual International Cryptology Conference. Berlin: Springer, 2016: 563-592.

[15] DEMERTZIS I, PAPADOPOULOS D, PAPAMANTHOU C, et al. SEAL: attack mitigation for encrypted databases via adjustable leakage[C]//Proceedings of the 29th USENIX Conference on Security Symposium. New York: ACM Press, 2020: 2433-2450.

[16] CHEN G X, LAI T H, REITER M K, et al. Differentially private access patterns for searchable symmetric encryption[C]//Proceedings of IEEE Conference on Computer Communications. Piscataway: IEEE Press, 2018: 810-818.

[17] SHANG Z, OYA S, PETER A, et al. Obfuscated access and search patterns in searchable encryption[J]. arXiv Preprint, arXiv: 2102.09651, 2021.

[18] REN K, GUO Y, LI J Q, et al. HybrIDX: new hybrid index for volume-hiding range queries in data outsourcing services[C]//Proceedings of 2020 IEEE 40th International Conference on Distributed Computing Systems (ICDCS). Piscataway: IEEE Press, 2021: 23-33.

[19] KAMARA S, MOATAZ T. Computationally Volume-Hiding Structured Encryption[C]//Annual International Conference on the Theory and Applications of Cryptographic Techniques. Cham: Springer, 2019: 183-213.

[20] PATEL S, PERSIANO G, YEO K, et al. Mitigating leakage in secure cloud-hosted data structures: volume-hiding for multi-maps via hashing[C]//Proceedings of the 2019 ACM SIGSAC Conference on Computer and Communications Security. New York: ACM Press, 2019: 79-93.

[21] JOOS M, ZIPF G K. The psycho-biology of language[J]. Language, 1936, 12(3): 196.

[22] GOLDREICH O, GOLDWASSER S, MICALI S. How to construct random functions[J]. Journal of the ACM, 1986, 33(4): 792-807.

[23] WANG J F, SUN S F, LI T C, et al. Practical volume-hiding encrypted multi-maps with optimal overhead and beyond[C]//Proceedings of the 2022 ACM SIGSAC Conference on Computer and Communications Security. New York: ACM Press, 2022: 2825-2839.

[24] GRAF T M, LEMIRE D. Xor filters: faster and smaller than bloom and cuckoo filters[J]. ACM Journal of Experimental Algorithmics, 2020, 25: 1-16.

[25] DEMERTZIS I, PAPADOPOULOS S, PAPAPETROU O, et al. Practical private range search revisited[C]//Proceedings of the 2016 International Conference on Management of Data. New York: ACM Press, 2016: 185-198.

[26] 刘文心, 高莹. 对称可搜索加密的安全性研究进展[J]. 信息安全学报, 2021, 6(2): 73-84.
LIU W X, GAO Y. A survey on security development of searchable symmetric encryption[J]. Journal of Cyber Security, 2021, 6(2): 73-84.

[27] NING J T, HUANG X Y, POH G S, et al. LEAP: leakage-abuse attack on efficiently deployable, efficiently searchable encryption with partially known dataset[C]//Proceedings of the 2021 ACM SIGSAC Conference on Computer and Communications Security. New York: ACM Press, 2021: 2307-2320.

[28] GUI Z C, JOHNSON O, WARINSCHI B. Encrypted databases: new volume attacks against range queries[C]//Proceedings of the 2019 ACM SIGSAC Conference on Computer and Communications Security. New York: ACM Press, 2019: 361-378.

[作者简介]

陈晶（1981− ），男，湖北武汉人，博士，武汉大学教授、博士生导师，主要研究方向为网络安全、应用密码学、分布式系统安全等。

李瀚星（1999− ），男，天津人，武汉大学硕士生，主要研究方向为网络安全、应用密码学等。

何琨（1986− ），男，湖北武汉人，博士，武汉大学副教授、博士生导师，主要研究方向为应用密码学、网络安全、云计算安全、人工智能安全、区块链安全等。

加梦（1996− ），女，湖北随州人，武汉大学博士生，主要研究方向为区块链、应用密码学等。

李雨晴（1991− ），女，江苏徐州人，博士，武汉大学副研究员、硕士生导师，主要研究方向为隐私计算、分布式学习、网络安全等。

杜瑞颖（1964− ），女，河南新乡人，博士，武汉大学教授、博士生导师，主要研究方向为网络安全、隐私保护等。