申报	系列：教师系列
	专业：计算机科学与技术
	职称：副教授

业绩成果材料

单　位（二级单位）　数学与信息学院

姓　名　　李宏博

材料核对人：　　单位盖章：

核对时间：

华南农业大学制

目　录

一、教学研究业绩

无

二、科研项目

1 主持：国家自然科学基金青年（C 类）项目

项目批准号	62502164
申请代码	F0206
归口管理部门	
依托单位代码	51064208A0499-0932

625021641015890

国家自然科学基金

资助项目计划书

（包干制项目）

资助类别：青年科学基金项目（C类）[原青年科学基金项目]

亚类说明：

附注说明：

项目名称：面向数据共享的动态可搜索加密算法研究

资助经费：30万元　　执行年限：2026.01-2028.12

负 责 人：李宏博　　BRID：07883.00.89552

通讯地址：广州市天河区五山路483号华南农业大学数学与信息(软件)学院603室

邮政编码：510642　　电　　话：

电子邮件：hongbo@scau.edu.cn

依托单位：华南农业大学

联 系 人：郑雪宜　　电　　话：020-85280070

填表日期：2025年08月29日

国家自然科学基金委员会制

Version：1.015.890

国家自然科学基金资助项目计划书填报说明

（包干制项目）

一、项目负责人收到《国家自然科学基金资助项目批准通知》（以下简称《批准通知》）后，请认真阅读本填报说明，参照国家自然科学基金相关项目管理办法和《国家自然科学基金资助项目资金管理办法》（以下简称《资金管理办法》，请查阅国家自然科学基金委员会门户网站首页“政策法规”栏目），按《批准通知》的要求认真填写和提交《国家自然科学基金资助项目计划书》（以下简称《计划书》）。

二、填写《计划书》时要科学严谨、实事求是、表述清晰、准确。《计划书》经国家自然科学基金委员会相关项目管理部门审核批准后，将作为项目研究计划执行、检查和验收的依据。

三、《计划书》各部分填写要求如下：

（一）简表：由系统自动生成。

（二）摘要及关键词：各类获资助项目都应当填写中、英文摘要及关键词。

（三）正文：

1. 青年科学基金项目（C类）、青年学生基础研究项目：如果《批准通知》所附“项目评审意见及修改意见表”中“修改意见”栏目没有修改要求的，只需选择“研究内容和研究目标按照申请书执行”即可；如果《批准通知》中上述栏目明确要求调整研究期限或研究内容等的，须选择“根据研究方案修改意见更改”并填报相关修改内容。

2. 青年科学基金项目（A类）和青年科学基金项目（B类）按下列提纲撰写：

（1）研究方向；

（2）结合国内外研究现状，说明研究工作的学术思想和科学意义（限两个页面）；

（3）研究内容、研究方案及预期目标（限两个页面）；

（4）年度研究计划；

四、资助经费相关要求：

1. 资助经费批准时不再区分直接费用和间接费用。

2. 项目负责人在提交计划书时需签署承诺书，承诺尊重科研规律，弘扬科学家精神，遵守科研伦理道德和作风学风诚信要求，认真开展科学研究工作；承诺项目经费全部用于与本项目研究工作相关的支出，不得用于与本项目研究无关的支出。

3. 项目负责人提交计划书时，无需编制项目预算。项目资金由项目负责人自主决定使用，按照《资金管理办法》第九条规定的开支范围列支。有关管理费用的补助支出，由依托单位根据实际管理需要，在充分征求项目负责人意见基础上合理确定。绩效支出由项目负责人根据实际科研需要和相关薪酬标准自主确定，依托单位按照工资制度进行管理。对于青年学生基础研究项目，支付给项目负责人本人的劳务费用，应符合相关比例要求。其余用途经费无额度限制，由项目负责人根据实际需要自主决定使用。

4. 项目结题时，项目负责人根据实际使用情况编制项目经费决算，经依托单位财务、科研管理部门审核后，报自然科学基金委。依托单位应当在单位内部公开非涉密项目立项、主要研究人员、资金使用（重点是间接费用、外拨资金、结余资金使用等）、决算、大型仪器设备购置以及项目研究成果等情况，接受内部监督。

5. 自然科学基金委结合项目管理，对经费使用情况和依托单位管理情况定期开展抽查。

五、其他事项

（一）根据有关要求，国家自然科学基金资助项目研究形成的代表性论文中发表在我国科技期刊上的应占20%以上。

（二）国家自然科学基金资助项目研究形成的专利申请应按照《建立财政资助科研项目形成专利的声明制度实施方案》要求进行声明。

简表

项目负责人信息	姓　　名	李宏博	性 别	男	出生年月	1991年02月	民 族	汉族
	学　　位	博士			职称	讲师		
	是否在站博士后	否		电子邮件		hongbo@scau.edu.cn		
	电　　话			个人网页		https://www.scholat.com/hongbo		
	工 作 单 位	华南农业大学						
	所 在 院 系 所	数学与信息(软件)学院						
依托单位信息	名　　称	华南农业大学					代码	51064208A0499
	联 系 人	郑雪宜		电子邮件		kycjhk@scau.edu.cn		
	电　　话	020-85280070		网站地址		http://kjc.scau.edu.cn/		
合作单位信息	单 位 名 称							
项目基本信息	项目名称	面向数据共享的动态可搜索加密算法研究						
	资助类别	青年科学基金项目（C类）[原青年科学基金项目]			亚类说明			
	附注说明							
	申请代码	F0206:信息安全						
	执行年限	2026.01-2028.12						
	资助经费	30万元						

项目摘要

中文摘要：

本项目聚焦数字经济时代数据安全共享的核心需求，针对传统公钥可搜索加密技术存在的密钥静态性、权限控制僵化及量子安全威胁等关键瓶颈问题，拟开展支持动态密钥轮换与多用户权限协同控制的可搜索加密理论创新研究。①设计时变密钥驱动的动态可搜索加密模型，建立基于时间序列的密钥轮换框架，实现动态权限控制机制；②探索多用户协同检索场景下的密钥聚合方法，消除密钥分片关联性导致的安全性降阶效应，构建支持动态多用户权限撤销与密钥同步轮换的多项式算法；③运用格密码理论探索动态格基优化方法，设计误差补偿方程，抑制密钥轮换过程中的维度膨胀现象，消除误差累积效应，建立量子随机预言机模型下可证明安全的动态多密钥聚合可搜索加密理论框架。研究成果预期形成具有动态权限管理能力的可搜索加密基础理论，发展抗量子攻击的新型密码学方法体系，为云计算环境下数据安全共享相关应用提供理论支撑，推进可搜索加密基础理论在数据要素背景中的实际应用。

Abstract：

This project focuses on the core demand for secure data sharing in the digital economy era. It addresses critical bottleneck issues of traditional public-key searchable encryption technology, such as key static nature, rigid access controls, and quantum security threats. The research aims at innovative theoretical developments in searchable encryption that support dynamic key rotation and collaborative control of multi-user permissions. ① A time-varying key-driven dynamic searchable encryption model will be designed, establishing a key evolution framework based on time series to realize a dynamic access control mechanism. ② The research will explore key aggregation methods in multi-user collaborative retrieval scenarios to eliminate the security degradation effect caused by key shard correlation. A polynomial algorithm will be constructed to support dynamic revocation of multi-user permissions and synchronized key updates. ③ Leveraging lattice-based cryptography, innovative methods for dynamic lattice basis optimization will be developed, along with error compensation equations to mitigate dimensional expansion during key rotation and eliminate error accumulation effects. This will result in a theoretically secure dynamic multi-key aggregation searchable encryption framework under a quantum random oracle model. The anticipated research outcomes aim to establish a fundamental theory of searchable encryption with dynamic permission management capabilities, develop a novel cryptographic methodology resistant to quantum attacks, provide theoretical support for secure data sharing in cloud computing environments, and advance the foundational theoretical application of searchable encryption technology in the context of data elements.

关键词(用分号分开)：公钥加密；可搜索加密；关键词检索；可更新加密；格基加密

Keywords(用分号分开)： Public-key Encryption; Searchable Encryption; Keyword Search; Updatable Encryption; Lattice-based Encryption

报告正文

研究内容和研究目标按照申请书执行。

本项目研究形成的代表性论文中发表在我国科技期刊上的将占 20%以上。

本项目研究形成的专利申请将按照《建立财政资助科研项目形成专利的声明制度实施方案》要求进行声明。


国家自然科学基金委员会
NATIONAL NATURAL SCIENCE FOUNDATION OF CHINA
科学基金网络信息系统
在线咨询
欢迎您，李宏博
项目负责人
退出
首页
申请与受理
项目批准
在研与结题
查询与统计
管理
项目计划书
计划调整书
在研管理
结题管理
项目变更
成果研究报告
填写项目计划书
计划书填写列表
项目批准号 / 项目名称 / 依托单位 / 资助类别 / 起止时间 / 项目经费 (万元)
报告年度
状态 / 最后提交时间
操作 / 截止日期
62502164，面向数据共享的动态可搜索加密算法研究
华南农业大学 / 数学与信息(软件)学院
青年科学基金项目（C类）[原青年科学基金项目]，2026-01-01至2028-12-31，30
2025
基金委已审核
批准通知书
2025-08-29 12:31:10
查看计划书
下载申请书

2 主持：广东省自然科学基金青年项目

受理编号：c232019102400000999

项目编号：2023A1515110618

文件编号：粤基金字〔2024〕4号

广东省基础与应用基础研究基金项目

任务书

项目名称：密文可更新的公钥认证可搜索加密研究

项目类别：区域联合基金-青年基金项目

项目起止时间：2023-11-01 至 2026-10-31

管理单位（甲方）：广东省基础与应用基础研究基金委员会

依托单位（乙方）：华南农业大学

通讯地址：广东省广州市天河区五山路483号

邮政编码：510642 单位电话：020-85283435

项目负责人：李宏博 联系电话：18819267316

（广东科技微信公众号） （查看任务书信息） （受理纸质材料二维码）

广东省基础与应用基础研究
基金委员会
二〇二〇年制

填写说明

一、项目任务书内容原则上要求与申报书相关内容保持一致，不得无故修改。

二、项目承担单位通过广东省科技业务管理阳光政务平台下载项目任务书，按要求完成签名盖章后扫描上传到广东省科技业务管理阳光政务平台。

三、签名盖章说明。请分别在单位工作分工及经费分配情况页、人员信息页、签约各方页等地方按要求签字或盖章，签章不合规或错漏将不予受理。其中，人员信息页要求所有参与人员本人亲笔签名，代签或印章无效，漏签将不予受理。

四、本任务书自签字并加盖公章之日起生效，各方均应负本任务书的法律责任，不应受机构、人事变动影响。

五、根据《广东省科学技术厅广东省财政厅关于深入推进省基础与应用基础研究基金项目经费使用“负面清单+包干制”改革试点工作的通知》(粤科规范字〔2022〕2号)，2022年度及以后立项资助的全部省基金项目(包括省自然科学基金、省市联合基金、省企联合基金项目等)均适用“负面清单+包干制”，项目提交申请书和任务书时无需编制费用明细科目预算。

一、主要研究内容和要达到的目标

研究内容

(1)在单服务器设置下构造PAUKS方案的理论方法。

目前的PAUKS方案能够抵抗来自服务器内部敌手的离线关键词猜测攻击，支持复杂度为的快速检索，并享有复杂度为的通信开销。但目前PAUKS的安全性基于完全可信的代理服务器的假设，现实场景中，用户通常难以寻找到完全可信的代理服务器。当没有完全可信的代理服务器时，目前的PAUKS方案不能满足预期的安全性。

因此，本项目拟重构PAUKS的系统模型，将代理服务器从系统模型中移除，研究在仅存云服务器的单服务器设置下，如何构造安全且高效的PAUKS方案。

(2)构造支持密文迭代的PAUKS方案的理论方法。

目前的PAUKS方案利用一次性的密文更新，降低了陷门的计算与通信复杂度，实现了密文快速检索，并具有较强的可扩展性，能支持如多用户密文分享等功能。但目前的PAUKS方案仅支持一次性的密文更新，不支持安全的密文迭代更新。在构建密码系统时，定期的密文更新可以降低密钥泄漏带来的风险，有助于实现长期安全。

因此，本项目拟研究PAUKS中实现密文安全迭代更新的理论方法与技术。

(3)研究PAUKS中验证密文更新有效性的理论方法。

目前的PAUKS方案中，接收者利用更新密钥生成更新令牌，将更新令牌提交给代理服务器，委托代理服务器更新云服务器存储的密文。由于代理服务器具有完全可信性，且云服务器未获得更新令牌的信息，密文更新的方案满足安全模型中的安全性。然而，对于云服务器，检索令牌及更新密钥相关的信息是随机且未知的，这导致云服务不能验证密文更新的有效性。特别地，在移除代理服务器的新系统模型中，验证密文更新的有效性将具有更重要的理论与实践意义。

因此，本项目拟研究PAUKS方案满足密文更新可验证性的充分条件和验证密文更新有效性的理论方法。

研究目标

(1)重构PAUKS系统模型，提出单服务器设置下的PAUKS新系统模型；解决对代理服务器的依赖性问题，构造出不依赖代理服务器的、单服务器设置下的PAUKS方案，满足IKGA攻击下的安全性，支持复杂度分别为和的密文快速检索和陷门高效通信。

(2)寻找到实现密文多次安全迭代的理论方法，进而寻找到支持密文多次安全更新、支持密文快速检索和陷门高效传输和能抵抗IKGA攻击的PAUKS构造方法；基于该预期方法，首先在基于代理的模型中构造出PAUKS具体方案，再构造出单服务器设置下的PAUKS具体方案。

(3)归纳总结出满足更新可验证性的充分条件，寻找到支持密文更新有效性验证的PAUKS构造方法；分别在基于代理服务器模型和单服务器模型下构造出可验证密文更新有效性的PAUKS具体方案。

二、项目预期获得的研究成果及形式

<table>
<tr><td rowspan="3">论文及专著情况</td><td colspan="2">国家统计源刊物以上刊物发表论文（篇）</td><td colspan="2">3</td><td colspan="2">科技报告（篇）</td><td colspan="2">0</td></tr>
<tr><td colspan="2">其中被SCI/EI/ISTP收录论文数（篇）</td><td colspan="2">3</td><td colspan="2">培养人才（人）</td><td colspan="2">2</td></tr>
<tr><td colspan="2">专著（册）</td><td colspan="2">0</td><td colspan="2">引进人才（人）</td><td colspan="2">0</td></tr>
<tr><td rowspan="3">专利情况(项)</td><td colspan="2">发明专利</td><td colspan="2">实用新型专利</td><td colspan="2">外观设计专利</td><td colspan="2">国外专利</td></tr>
<tr><td>申请</td><td>授权</td><td>申请</td><td>授权</td><td>申请</td><td>授权</td><td>申请</td><td>授权</td></tr>
<tr><td>2</td><td>1</td><td>0</td><td>0</td><td>0</td><td>0</td><td>0</td><td>0</td></tr>
<tr><td>其他</td><td colspan="8"></td></tr>
</table>

三、项目进度和阶段目标

（一）项目起止时间： 2023-11-01 至 2026-10-31		
（二）项目实施进度及阶段主要目标:		
开始日期	结束日期	主要工作内容
2023-11-01	2024-10-31	本项目将开展单服务器设置下PAUKS方案的构造方法研究，设计出不依赖代理服务器的PAUKS方案，在国内外高水平期刊/会议发表学术论文1篇，申请发明专利1件；组织学术交流研讨1次。
2024-11-01	2025-10-31	本项目将对PAUKS中密文的安全迭代更新问题开展研究，提出支持密文多次安全更新的PAUKS方案，在国内外高水平期刊/会议发表学术论文1篇；拟组织学术交流研讨1次；拟举办或协助举办国内/国际学术会议1次，邀请国内外专家讲学，促进合作交流，保证产出成果的前沿性。
2025-11-01	2026-10-31	本项目将研究PAUKS中密文更新后的可验证性问题，提出可验证密文有效性的PAUKS方案，在国内外高水平期刊/会议发表学术论文1篇，申请发明专利1件；组织学术交流研讨1次。

四、项目总经费及省基金委经费预算

（一）省基金委经费下达总额： （大写）壹拾万圆整；（小写 ）10万元；					
（二）省基金委经费年度下达计划：					
年度	2023 年	年	年	年	年
经费(万元)	10.00				

五、人员信息

项目负责人								
姓名	证件号码	年龄	性别	职称	学历	在项目中承担的任务	所在单位	签名
李宏博	370213199102085234	33	男	未取得	博士研究生	项目负责人	华南农业大学	李宏博

六、工作分工及财政经费分配

承担/参与单位名称（盖章）	工作分工	省级财政科技资金分配（万元）
华南农业大学		10
	合计	10

七、任务书条款

第一条 甲方与乙方根据《中华人民共和国民法典》及国家有关法规和规定，按照《广东省科学技术厅关于广东省基础与应用基础研究基金（省自然科学基金、联合基金等）项目管理的实施细则（试行）》《省级科技计划项目任务书管理细则》《广东省省级科技计划项目验收结题工作规程（试行）》等规定，为顺利完成（2023）年密文可更新的公钥认证可搜索加密研究专项项目（项目编号：2023A1515110618）经协商一致，特订立本任务书，作为甲乙双方在项目实施管理过程中共同遵守的依据。

第二条 甲方的权利义务：

1.按任务书规定进行经费核拨的有关工作协调。

2.根据甲方需要，在不影响乙方工作的前提下，定期或不定期对乙方项目的实施情况和经费使用情况进行检查或抽查。

3.根据《广东省科研诚信管理办法(试行)》等规定对乙方进行科技计划信用管理。

第三条 乙方的权利义务：

1.确保落实自筹经费及有关保障条件。

2.按任务书规定，对甲方核拨的经费实行专款专用，单独列账，并随时配合甲方进行监督检查。

3.经费使用按照广东省级财政科研项目经费使用等有关规定进行管理。

4.项目依托单位应制定经费使用“负面清单+包干制”内部管理制度并报甲方备案。

5.使用财政资金采购设备、原材料等，按照《广东省实施〈中华人民共和国招标投标法〉办法》有关规定，符合招标条件的须进行招标。

6.项目任务书任务完成后，或任务书规定的任务、指标及经费投入等提前完成的，乙方可提出验收结题申请，并按甲方要求做好项目验收结题工作。

7.若项目发生需要终止结题的情况，乙方须提出终止结题申请，并按甲方要求做好项目终止结题工作。

8.在每年规定时间内向甲方如实提交上年度工作情况报告，报告内容包含上年度项目进展情况、经费决算和取得的成果等。

9.按照国家和省有关规定，提交科技报告及其他材料。

10.利用甲方的经费获得的研究成果，项目负责人和参与者应当注明获得“广东省基础与应用基础研究基金（英文：Guangdong Basic and Applied Basic Research Foundation）（项目编号）”资助或作有关说明。

11.乙方要恪守科学道德准则，遵守科研活动规范，践行科研诚信要求，不得抄袭、剽窃他人科研成果或者伪造、篡改研究数据、研究结论；不得购买、代写、代投论文，虚构同行评议专家及评议意见；不得违反论文署名规范，擅自标注或虚假标注获得科技计划（专项、基金等）等资助；不得弄虚作假，骗取科技计划（专项、基金等）项目、科研经费以及奖励、荣誉等；不得有其他违背科研诚信要求的行为。

12.确保本项目开展的研究工作符合我国科研伦理管理相关规定。

第四条 在履行本任务书的过程中，如出现广东省相关政策法规重大改变等不可抗力情况，甲方有权对所核拨经费的数量和时间进行相应调整。

第五条 在履行本任务书的过程中，当事人一方发现可能导致项目整体或部分失败的情形时，应及时通知另一方，并采取适当措施减少损失，没有及时通知并采取适当措施，致使损失扩大的，应当就扩大的损失承担责任。

第六条 本项目技术成果的归属、转让和实施技术成果所产生的经济利益的分享，除双方另有约定外，按国家和广东省有关法规执行。

第七条 根据项目具体情况，经双方另行协商订立的附加条款，作为本任务书正式内容的一部分，与本任务书具有同等效力。

第八条 本任务书一式三份，各份具有同等效力。甲、乙方及项目负责人各执一份，三方签字、盖章后即生效，有效期至项目结题后一年内。各方均应负任务书的法律责任，不应受机构、人事变动的影响。

第九条 乙方必须接受甲方聘请的本项目任务书监理单位的监督和管理。监理单位按照甲方赋予的权利对本项目任务书的履行进行审核、进度调查，对项目任务书变更、经费使用情况进行监督管理及组织项目验收。

说明：1.本任务书中，凡是当事人约定无需填写的内容，应在空白处划（/）。

2.委托代理人签订本任务书的，应出具合法、有效的委托书。

八、本任务书签约各方

管理单位（甲方）：　　广东省基础与应用基础研究基金委员会　（盖章）

法定代表人（或法人代理）：　曾路　（签章）

2024 年　04 月　08 日

依托单位（乙方）：　华南农业大学　（盖章）

法定代表人（或法人代理）：　薛红卫　（签章）

联系人（项目主管）姓名：　倪慧群　（签章）

Email：　kjcgxk@scau.edu.cn

电话：　020-85283435　/　15920301530

开户单位名称：　华南农业大学

开户银行名称：　广东广州工行五山支行

开户银行账号：　3602002609000310520

2024 年 4 月 8 日

联系人（项目负责人）姓名：　李宏博　（签名）

Email：　hongbo@scau.edu.cn

电话：　18819267316

2024 年 4 月 8 日

3 主参：横向项目 高可靠密文数据细粒度授权检索研究

课题编号：YNSC24114

云南省服务计算重点实验室
开放课题合同书

课题名称：高可靠密文数据细粒度授权检索研究

委托方（甲方）：云南省服务计算重点实验室

受托方（乙方）：华南农业大学

甲方依托单位（丙方）：云南财经大学

课题负责人：肖媚燕　　手机：18675871082

课题起止时间：2025 年 1 月 1 至 2025 年 12 月 31 日

云南省服务计算重点实验室

2024 年

合 同 条 款

本合同甲方委托乙方就云南省服务计算重点实验室开放课题“高可靠密文数据细粒度授权检索研究”进行科学研究，并支付相应的开放课题经费。三方经过友好协商，在真实、充分地表达各自意愿的基础上，根据相关规定，达成如下合同条款，三方共同恪守。

第一条：甲方委托乙方进行科学研究的内容如下：

乙方研究内容及科技成果应覆盖申请书主要部分或与申请书主要内容一致。

第二条：丙方向乙方转账划拨开放课题经费金额及方式如下：

一、开放课题总额为：￥10000.00 元（壹万圆整）人民币。

二、开放课题经费由丙方 一次性 转账划拨给乙方。

三、需“先票后款”，即乙方先开票据之后，丙方再拨付款项。

四、时间要求：合同签订后 5 个工作日内，乙方向甲方提供事业单位往来收据或增值税普通发票（需盖章）；甲方收到乙方票据后 15 个工作日内，丙方以银行转账的方式向乙方拨付开放课题经费。

五、乙方开户银行名称、户名和账号为：

1、开户银行：广州工行五山支行

2、户　　名：华南农业大学

3、帐　　号：3602002609000310520

六、丙方开户银行名称、户名和账号为：

1、开户银行：中国农业银行股份有限公司昆明龙泉路支行

2、户　　名：云南财经大学

3、帐　　号：24013601040000374

七、乙方应当确保上述账户信息真实、合法、有效，因乙方提供的账户信息错误，导致的一切后果和责任，由乙方自行承担。

第三条：课题成果要求：

1、开放课题成果主要包括论文、发明专利、专著等，结项验收时须至少提

交1份研究报告及1篇以上SCI/SSCI期刊论文或CCF C类及以上会议论文或CSCD或北大中文核心期刊论文（不含预警期刊、负面清单期刊、增刊等）。

2、开放课题取得的研究成果和知识产权，由重点实验室与申请人共同所有和共享。鼓励与重点实验室专职人员开展合作研究。同时，申请人须在所取得的研究成果和知识产权中对重点实验室进行署名，否则不计入结题成果。署名方式如下：

（1）把重点实验室作为工作单位之一标注，中文单位信息："云南财经大学云南省服务计算重点实验室"，英文单位信息："Yunnan Key Laboratory of Service Computing, Yunnan University of Finance and Economics"。

（2）对于论文、专著等研究成果，除了署名工作单位信息外，还应在适当位置标明"云南省服务计算重点实验室开放课题"和资助编号。（中文为：云南省服务计算重点实验室开放课题（编号：YNSC24114），英文为：Supported by the Foundation of Yunnan Key Laboratory of Service Computing (No. YNSC24114)。

注：（1）（2）须同时满足。

第四条：课题验收：

一、乙方按本合同第一、三条要求完成开放课题的成果，论文须发表见刊（包括online）。

二、验收时间于2025年12月31日前，乙方应至少提前2周向甲方提交相关材料。

第五条：乙方若未能在合同期内取得本合同要求的成果，甲方有权作撤项处理，丙方有权追回课题经费。

第六条：本合同未尽事宜，按《云南省重点实验室建设与运行管理办法》《云南省服务计算重点实验室开放课题管理办法》《2024年度云南省服务计算重点实验室开放课题申报通知》等有关规定执行。

第七条：合同书一式3份，甲、乙、丙方各1份，本合同经三方签字盖章后生效。

附录 1 课题经费预算表

金额单位：万元（保留两位小数）

预算科目	财政经费	用途说明
合计	1.0	
1. 材料费	0.1	采购实验材料用于方案测试
2. 差旅费/会议费/国际合作交流费	0.2	用于项目调研、学术交流等
3. 出版/文献/信息传播/知识产权事务费	0.14	用于论文和知识产权等费用
4. 劳务费	0.4	用于支付研究生劳务费
5. 专家咨询费	0.1	用于邀请专家进行项目咨询
6. 管理费	0.06	用于学校项目管理费

合同书各责任方签字签章

委托方（甲方）	云南省服务计算重点实验室
经办人（签字）： 实验室主任（签章）： 单位公章 年　月　日	
受托单位（乙方）	华南农业大学
课题负责人（签字）： 法定代表人（签章）： 单位公章 2024 年 11 月 26 日	
甲方依托单位（丙方）	云南财经大学
经办人（签字）： 法定代表人（签章）： 单位公章 2024 年 11 月 26 日	

4 主参：横向项目猪业二部药物只能仓储数字化项目

技术开发（委托）合同

项目名称：猪业二部药物智能仓储数字化项目

委托方（甲方）：温氏食品集团股份有限公司

受托方（乙方）：华南农业大学

签订时间：2025 年 11 月 30 日

签订地点：广东省云浮市

有效期限：2025 年 11 月 30 日至
2026 年 6 月 30 日

中华人民共和国科学技术部印制

填 写 说 明

一、本合同为中华人民共和国科学技术部印制的技术开发（委托）合同示范文本，各技术合同认定登记机构可推介技术合同当事人参照使用。

二、本合同书适用于一方当事人委托另一方当事人进行新技术、新产品、新工艺或者新材料及其系统的研究开发所订立的技术开发合同。

三、签约一方为多个当事人的，可按各自在合同关系中的作用等，在“委托方”、“受托方”项下（增页）分别排列为共同委托人或共同受托人。

四、本合同书未尽事项，可由当事人附页另行约定，并可作为本合同的组成部分。

五、当事人使用本合同书时约定无需填写的条款，应在该条款处注明“无”等字样。

技术开发（委托）合同

委托方（甲方）：温氏食品集团股份有限公司

住　所　地：广东云浮新兴

法定代表人：温志芬

项目联系人：谢启钊

联系方式 ：13826862926

通讯地址：广东省云浮市新兴县新城镇东堤北路 9 号

受托方（乙方）：华南农业大学

住　所　地：广州市天河区五山路 486 号

法定代表人：薛红卫

项目联系人：黄立峰

联系方式 ：13929500478

通讯地址：广州市天河区五山路 486 号 数学与信息学院 514

电话：13929500478

电子信箱：huanglf6@scau.edu.cn

本合同甲方委托乙方研究开发 猪业二部药物智能仓储数字化项目 项目，并支付研究开发经费和报酬，乙方接受委托并进行此项研究开发工作。双方经过平等协商，在真实、充分地表达各自意愿的基础上，根据《中华人民共和国民法典》的规定，达成如下协议，并由双方共同恪守。

第一条 本合同研究开发项目的要求如下：

1．技术目标：智能仓储推荐算法通过“数据驱动 + 智能优化”的方式，实现仓储作业的精细化和智能决策，从而显著提升仓储体系的空间利用率与准确性，降低执行时间、缺货率与积压率。

2．技术内容：（1）方案设计，包括需求分析与方案设计；（2）算法实现，包括同批次合并优先模块、历史高度偏好分析（人因工程优化）、距离最短优先（动线优化）、空货位分配策略、多货位拆分与结果输出；（3）系统联调测试；（4）上线准备与相关操作。

3．技术方法和路线：（1）同批次合并优先模块。实现先进先出（FIFO），防止混批，便于质量追溯。通过查找当前仓库中是否存在与本次入库物料编码相同、生产日期相同、保质期相同的已有存储单元；筛选出其中可用容量 ≥ 本次入库数量的候选货位。（2）历史高度偏好分析。多个候选货位均满足“同批次+容量足够”条件。统计该物料在历史入库记录中，出现在高区、中区、低区三个垂直层级的频次；构建频率分布向量，如：[低区: 60%, 中区: 30%, 高区: 10%]；优先推荐位于历史出现频率最高的高度层级的货位；若多个货位处于同一高频层级，则选择物理高度更低者（如低区优于中区）。（3）距离最短优先。从多个等效候选货位中推荐距离最短者。（4）空货位分配策略。系统通过抓取该物料过去一段时间的出入库记录，综合考虑它被入库和出库的频繁程度、每次操作的数量大小等因素，计算出一个“热度得分”，通过“热度”定义“黄金分区”、“冷区”。系统不仅关注单个物料的热度，还会分析哪些物料经常一起出现，从而让“好伙伴”尽量住得近一些。这里同时考虑两种协同关系，即通过分析历史入库单据与历史出库单据，识别出哪些物料经常在同一张入库单和出库单中出现。例如，某批原材料 A 和 B 总是由一同送达，原材料 B 和 C 一同出库，说明它们在供应端高度关联。这样在库位分配时，生成日期不同的 A/B/C 物料都相近分配。最终形成一个统一的“物料亲和

关系网络”。（5） 多货位拆分与结果输出。如果单个空货位无法容纳本次全部入库数量，系统会循环执行上述推荐逻辑，依次为剩余数量分配次优、再次优的货位，直到全部入库量分配完毕，并输出最终的货位分配清单。

第二条 乙方应在本合同生效后 30 日内向甲方提交研究开发计划。研究开发计划应包括以下主要内容：

1. 总体设计方案、开发计划、详细原型设计；
2. 项目开发过程、分阶段开发清单和进度；
3. /。

第三条 乙方应按下列进度完成研究开发工作：

1. 2025.12-2026.02 算法实现，包括同批次合并优先模块、历史高度偏好分析（人因工程优化）、距离最短优先（动线优化）、空货位分配策略、多货位拆分与结果输出；
2. 2026.02-2026.03 系统联调测试、上线准备。

第四条 甲方应向乙方提供的技术资料及协作事项如下：

1. 技术资料清单：整体方案、详细原型设计、技术开发接口文档。
2. 提供时间和方式：双方协商。
3. 其他协作事项：无。

本合同履行完毕后，上述技术资料按以下方式处理：交还甲方。

第五条 甲方应按以下方式支付研究开发经费和报酬：

1. 研究开发经费和报酬（含税）总额为 216500 元（大写：人民币贰拾壹万陆仟伍佰元整）。

2. 研究开发经费由甲方 分期 （一次、分期或提成）支付乙方。具体支付方式和时间如下：

（1） 合同签订后 15 个工作日内，出具相应项目方案设计通过甲

方确认后，由甲方向乙方一次性支付人民币壹拾万捌仟元整（￥108000）

（2）项目结题验收通过后15个工作日内，由甲方向乙方一次性支付人民币壹拾万捌仟伍佰元整（￥108500）

乙方开户银行名称、地址和帐号为：

开户银行：中国工商银行广州五山支行

地址：广州市天河区五山路483号

帐号：3602002609000310520

3．双方确定，甲方以实施研究开发成果所产生的利益提成支付乙方的研究开发经费和报酬的，乙方有权以双方协商确认的方式查阅甲方有关的会计帐目。

第六条 本合同的研究开发经费由乙方以双方协商确认的方式使用。甲方有权以双方协商确认的方式检查乙方进行研究开发工作和使用研究开发经费的情况，但不得妨碍乙方的正常工作。

第七条 本合同的变更必须由双方协商一致，并以书面形式确定。但有下列情形之一的，一方可以向另一方提出变更合同权利与义务的请求，另一方应当在15日内予以答复；逾期未予答复的，视为同意：

1．无；

2．无。

第八条 未经甲方同意，乙方不得将本合同项目部分或全部研究开发工作转让第三人承担。但有下列情况之一的，乙方可以不经甲方同意，将本合同项目部分或全部研究开发工作转让第三人承担：

1．无；

2．无。

乙方可以转让研究开发工作的具体内容包括：无。

第九条 在本合同履行中，因出现在现有技术水平和条件下难以克服的技术困难，导致研究开发失败或部分失败，并造成一方或双方损失的，

双方按如下约定承担风险损失：双方协商确认。

双方确定，本合同项目的技术风险按双方协商确认的方式认定。认定技术风险的基本内容应当包括技术风险的存在、范围、程度及损失大小等。认定技术风险的基本条件是：

1. 本合同项目在现有技术水平条件下具有足够的难度；

2. 乙方在主观上无过错且经认定研究开发失败为合理的失败。

一方发现技术风险存在并有可能致使研究开发失败或部分失败的情形时，应当在15日内通知另一方并采取适当措施减少损失。逾期未通知并未采取适当措施而致使损失扩大的，应当就扩大的损失承担赔偿责任。

第十条 在本合同履行中，因作为研究开发标的的技术已经由他人公开（包括以专利权方式公开），一方应在15日内通知另一方解除合同。逾期未通知并致使另一方产生损失的，另一方有权要求予以赔偿。

第十一条 双方确定因履行本合同应遵守的保密义务如下：

甲方：

1. 保密内容（包括技术信息和经营信息）：涉及本合同的技术文件、资料和商业秘密。

2. 涉密人员范围：项目组成员。

3. 保密期限：本合同项下乙方的保密义务在保密信息合法公开前持续有效。未征得甲方事先的书面同意，乙方不得以任何形式向任何第三方披露全部或部分保密信息。

4. 泄密责任：双方协商解决。

乙方：

1. 保密内容（包括技术信息和经营信息）：涉及本合同的技术文件、资料和商业秘密。

2. 涉密人员范围：项目组成员。

3. 保密期限：合同期内。

4. 泄密责任：双方协商解决。

第十二条 乙方应当按以下方式向甲方交付研究开发成果：

1．研究开发成果交付的形式及数量：（1）设计方案1份；（2）入库推荐算法1个；（3）配合进行系统联调测试；（4）配合上线准备。

2．研究开发成果交付的时间及地点：按照甲方确定的项目详细技术开发规定履行。

第十三条 双方确定，按以下标准及方法对乙方完成的研究开发成果进行验收：乙方按照合同约定提交成果且甲方验收，即视为验收通过。

第十四条 乙方应当保证其交付给甲方的研究开发成果不侵犯任何第三人的合法权益。如发生第三人指控甲方实施的技术侵权，乙方应当退还研发经费。

第十五条 双方确定，因履行本合同所产生的研究开发成果及其相关知识产权权利归属，按下列第1种方式处理：

1．双（甲、乙、双）方享有申请专利的权利。

专利权取得后的使用和有关利益分配方式如下：双方协商约定。

2．按技术秘密方式处理。有关使用和转让的权利归属及由此产生的利益按以下约定处理：

（1）技术秘密的使用权：甲方所有

（2）技术秘密的转让权：甲方所有

（3）相关利益的分配办法：双方协商约定

。

双方对本合同有关的知识产权权利归属特别约定如下：双方所有

。

第十六条 乙方不得在向甲方交付研究开发成果之前，自行将研究开发成果转让给第三人。

第十七条 乙方完成本合同项目的研究开发人员享有在有关技术成果文件上写明技术成果完成者的权利和取得有关荣誉证书、奖励的权利。

第十八条 乙方利用研究开发经费所购置与研究开发工作有关的

设备、器材、资料等财产，归 乙 （甲、乙、双）方所有。

第十九条 双方确定，乙方应在向甲方交付研究开发成果后，根据甲方的请求，为甲方指定的人员提供技术指导和培训，或提供与使用该研究开发成果相关的技术服务。

1．技术服务和指导内容： 项目相关算法技术 。

2：地点和方式： 远程指导或现场指导 。

3．费用及支付方式： 免费 。

第二十条 双方确定：任何一方违反本合同约定，造成研究开发工作停滞、延误或失败的，按以下约定承担违约责任：

1． 甲 方违反本合同第 四 条约定，应当 尽快支付研发经费。如因此导致乙方交付时间延迟，乙方不承担责任 （支付违约金或损失赔偿额的计算方法）。

2． 甲 方违反本合同第 五 条约定，应当 尽快支付研发经费。如因此导致乙方交付时间延迟，乙方不承担责任 （支付违约金或损失赔偿额的计算方法）。

3． 乙 方违反本合同第 二 条约定，应当 尽快采取补救措施，向甲方支付因此造成的损失 （支付违约金或损失赔偿额的计算方法）。

4． 乙 方违反本合同第 三 条约定，应当 尽快采取补救措施，向甲方支付因此造成的损失 （支付违约金或损失赔偿额的计算方法）。

第二十一条 双方确定，甲方有权利用乙方按照本合同约定提供的研究开发成果，进行后续改进。由此产生的具有实质性或创造性技术进步特征的新的技术成果及其权属，由 甲 （甲、乙、双）方享有。具体相关利益的分配办如下： 甲方享有 100% 。

乙方有权在完成本合同约定的研究开发工作后，利用该项研究开发成果进行后续改进。由此产生的具有实质性或创造性技术进步特征的新的技术成果，归 乙 （甲、乙、双）方所有。具体相关利益的分配办法如下：

乙方享有 100%。

第二十二条 双方确定，在本合同有效期内，甲方指定 谢启钊 为甲方项目联系人，乙方指定 黄立峰 为乙方项目联系人。项目联系人承担以下责任：

1. 对项目实施过程有关疑难问题进行沟通协调；

2. 对相关技术范围内难以解决的事宜，进行汇报并督促落实。

一方变更项目联系人的，应当及时以书面形式通知另一方。未及时通知并影响本合同履行或造成损失的，应承担相应的责任。

第二十三条 双方确定，出现下列情形，致使本合同的履行成为不必要或不可能的，一方可以通知另一方解除本合同：

1. 因发生不可抗力或技术风险；

2. _____

第二十四条： 双方因履行本合同而发生的争议，应协商、调解解决。协商、调解不成的，确定按以下第 1 种方式处理：

1. 提交 中国云浮新兴 仲裁委员会仲裁；

2. 依法向人民法院起诉。

第二十五条 双方确定：本合同及相关附件中所涉及的有关名词和技术术语，其定义和解释如下：

1. _____

第二十六条 与履行本合同有关的下列技术文件，经双方确认后，无 为本合同的组成部分：

1. 技术背景资料：无；

2. 可行性论证报告：无；

3. 技术评价报告：无；

4. 技术标准和规范：无；

5. 原始设计和工艺文件：无；

6. 其他：无。

第二十七条 双方约定本合同其他相关事项为：无。

第二十八条 本合同一式肆份，甲乙双方各执贰份，具有同等法律效力。

第二十九条 本合同经双方签字盖章后生效。

甲方：温氏食品集团股份有限公司（盖章）

法定代表人/委托代理人：（签名）

年 月 日

乙方：华南农业大学（盖章）

法定代表人/委托代理人：薛红卫（签名）

年 月 日

印花税票粘贴处：

三、论文、著作等

1 检索证明

SCAULIB202626225

检索证明

根据委托人提供的论文材料，委托人华南农业大学数学与信息学院 李宏博(学科类型:自然科学) 3 篇论文收录情况如下表。

序号	论文名称	发表刊物及发表的年月卷期/页码等	作者排名	论文等级	作者文中单位	收录情况	影响因子	中科院大类分区
1	Public-Key Authenticated Encryption With Keyword Search Supporting Constant Trapdoor Generation and Fast Search	IEEE TRANSACTIONS ON INFORMATION FORENSICS AND SECURITY 出版年：2023 卷期：18 页码：396-410 文献类型：Article	第一作者	T2 类	华南农业大学	SCI	IF2-year=6.3 IF5-year=7.3 （2023）	计算机科学 1 区 Top 期刊：是 OA 期刊：否 （2023）
2	A Secure Cloud Data Sharing Protocol for Enterprise Supporting Hierarchical Keyword Search	IEEE Transactions on Dependable and Secure Computing 出版年：2022 卷期：19 3 页码：- 文献号： 文献类型：Article	第一作者	A 类	华南农业大学	SCI	IF2-year=7.3 IF5-year=7.2 （2022）	计算机科学 2 区 Top 期刊：否 OA 期刊：否 （2022）
3	Proxy-Free Public-Key Authenticated Updatable and Searchable Encryption for Cloud Storage	IEEE Transactions on Dependable and Secure Computing 出版年：2026 出版日期：JAN 2026	第一作者	A 类	华南农业大学	SCI	IF2-year=7.5 IF5-year=7.1 （2024）	计算机科学 2 区 Top 期刊：是 OA 期刊：否 （2025）

		卷期：23 1 页码：- 文献号： 文献类型：Article						

说明：论文等级和中科院大类分区按《华南农业大学学术论文评价方案（试行）》划分。

报告免责声明:如未盖章,报告无效

检索员：尹银怀

华南农业大学图书馆

2026-03-16

SCAULIB202626227

检索证明

根据委托人提供的论文材料，委托人华南农业大学数学与信息学院 李宏博(学科类型:自然科学) 2 篇论文收录情况如下表。

序号	论文名称	发表刊物及发表的年月卷期/页码等	作者排名	论文等级	作者文中单位	收录情况	影响因子	中科院大类分区
1	A more efficient public-key authenticated encryption scheme with keyword search	JOURNAL OF SYSTEMS ARCHITECTURE 出版年：2023 出版日期：APR 卷期：137 页码：- 文献号：102839 文献类型：Article	唯一通讯作者	A 类	华南农业大学	SCI	IF2-year=3.8 IF5-year=3.6 （2023）	计算机科学 2 区 Top 期刊：否 OA 期刊：否 （2023）
2	Secure channel free public key authenticated encryption with multi-keyword search on healthcare systems	FUTURE GENERATION COMPUTER SYSTEMS-THE INTERNATIONAL JOURNAL OF ESCIENCE 出版年：2023 出版日期：AUG 卷期：145 页码：511-520 文献类型：Article	唯一通讯作者	A 类	华南农业大学	SCI	IF2-year=6.2 IF5-year=5.9 （2023）	计算机科学 2 区 Top 期刊：是 OA 期刊：否 （2023）

说明：论文等级和中科院大类分区按《华南农业大学学术论文评价方案（试行）》划分。

报告免责声明:如未盖章,报告无效

检索员：尹银怀


华南农业大学图书馆
2026-03-16
华南农业大学图书馆
信息检索专用章

2 以第一作者发表本专业论文

2.1 代表作：Public-Key Authenticated Encryption With Keyword Search Supporting Constant Trapdoor Generation and Fast Search

发表于 IEEE Transactions on Informatics and Security (IF5-year-7.3)
中科院 1 区 CCF-A Top 期刊
论文等级：T2 类

Public-Key Authenticated Encryption With Keyword Search Supporting Constant Trapdoor Generation and Fast Search

Hongbo Li, Qiong Huang, Jianye Huang, and Willy Susilo, *Fellow, IEEE*

***Abstract*— To improve the quality of medical care and reduce unnecessary medical errors, electronic medical records (EMRs) are widely applied in hospital information systems. However, rapidly increasing EMRs bring heavy storage burden to hospitals. Professional data management service provided by cloud server can save the hospital local storage, and meanwhile, realize EMRs sharing among external researchers. However, the risk of leaking information of patients discourages hospitals to outsource patients' EMRs to the remote cloud server. In this paper, a secure and efficient cloud storing and sharing method can be achieved by applying the proposed public key authenticated encryption with ciphertext update and keyword search (PAUKS). The proposed PAUKS scheme enables EMRs to be encrypted and queried without decryption, and is secure against inside keyword guessing attacks. Compared with the recently proposed PAEKS in literature, the PAUKS scheme enjoys smaller computation and communication overheads. The required number of trapdoors per query is constant in PAUKS scheme, instead of the linearly expanding as the number of senders increases in PAEKS. Furthermore, an inverted index can be built safely in PAUKS scheme to accelerate the query procedure. Experiment results show that our PAUKS scheme owns a comparable running overhead, but enjoys a higher query efficiency after ciphertexts update.**

***Index Terms*— Searchable encryption, keyword guessing attacks, electronic medical record, light overhead, fast search.**

I. Introduction

ELECTRONIC medical record (EMR) plays an important role in the hospital information system, which managing information of patients including name, age, gender, allergic history, social security information, admission diagnosis, operation note, etc. EMR not only facilitates technical exchanges and medical case studies between patients, families and medical researchers, but also provides faster and more convenient services to patients, reduces the workload of medical staff, and reduces medical risks. However, the two problems in electronic EMR system, namely insufficient storage space and information security issues, arouse people's attention.

Although each electronic medical record takes up a small amount of storage space, the accumulated electronic medical records also take up a large amount of space due to the large number of patients. Especially in some famous hospitals, the number of patients is very large, and it is more difficult for hospitals to manage electronic medical records; On the other hand, once the hospital's network system is threatened by viruses, it is very likely to cause the network system to collapse or even cause the electronic medical records to be leaked, greatly threatening patients privacy.

To solve the above problems, an effective solution is to use sophisticated cloud computing and cloud storage technologies to ensure data consistency and loss prevention. Hospitals can outsource encrypted electronic medicals records to the cloud. The procedure of this solution is shown in Figure 1(a). At the beginning, the attending doctor encrypts patients' EMRs with the public key of the hospital data administrator and sends the encrypted EMRs to the administrator. The administrator checks the validity of the electronic medical records and outsources them to the cloud. The administrator is responsible for the data access control and grants relevant doctors or medical researchers access to the outsourced encrypted data. The drawback of this solution is that utilizing general public key encryption schemes cannot meet the demand of frequent queries from doctors and medical researchers.

Public key encryption with keyword search (PEKS) firstly proposed by Boneh et al. [1] probably satisfies the requirement. As shown in figure 1(b), a user is able to generate a trapdoor with a keyword and let the cloud server test whether there are ciphertexts matching with the trapdoor, i.e. ciphertexts embedded with the same keyword as the trapdoor. However, Boneh et al.'s PEKS scheme and a number of PEKS schemes were pointed out insecure under the special attacks named off-line keyword guessing attacks (KGAs) [2]. Once the keyword space is not large, a probabilistic polynomial time (PPT) adversary could detect which keyword is queried. Unfortunately, the keyword space in real world is small enough for a PPT adversary to launch KGAs.

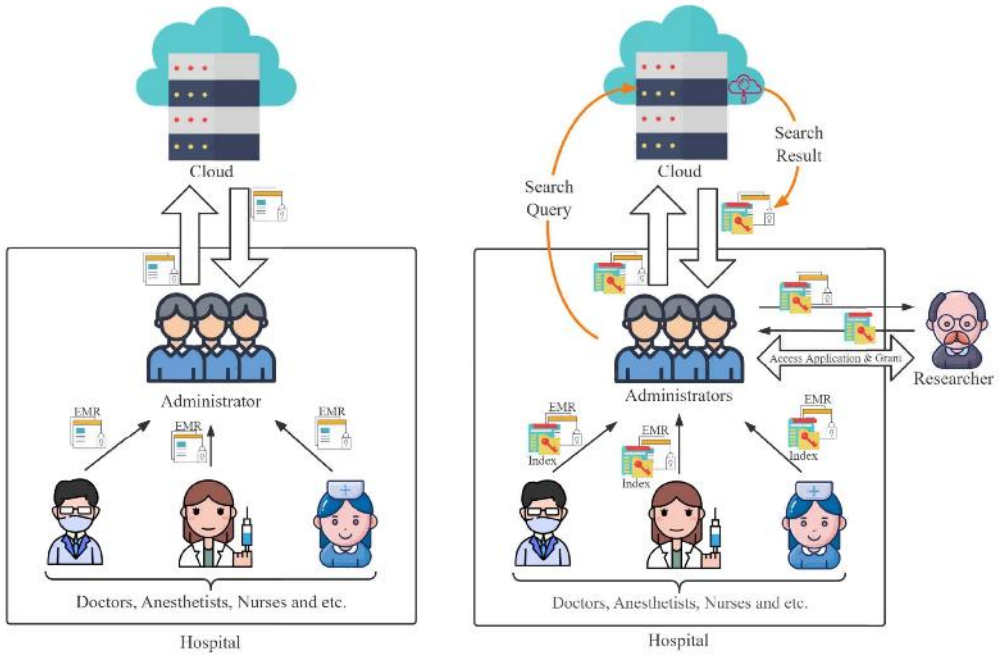
Manuscript received 5 March 2022; revised 11 September 2022; accepted 12 November 2022. Date of publication 23 November 2022; date of current version 7 December 2022. This work was supported in part by the Major Program of Guangdong Basic and Applied Research under Grant 2019B030302008, in part by the National Natural Science Foundation of China under Grant 61872152 and Grant 62272174, and in part by the Science and Technology Program of Guangzhou under Grant 201902010081. The associate editor coordinating the review of this manuscript and approving it for publication was Mr. Frederik Armknecht. *(Corresponding author: Qiong Huang.)*

Hongbo Li is with the College of Mathematics and Informatics, South China Agricultural University, Guangzhou 510642, China (e-mail: hongbo@scau.edu.cn).

Qiong Huang is with the College of Mathematics and Informatics and Guangzhou Key Laboratory of Intelligent Agriculture, South China Agricultural University, Guangzhou 510642, China (e-mail: qhuang@scau.edu.cn).

Jianye Huang and Willy Susilo are with the School of Computing and Information Technology, University of Wollongong, Wollongong, NSW 2522, Australia (e-mail: jh207@uow.edu.au; wsusilo@uow.edu.au).

Digital Object Identifier 10.1109/TIFS.2022.3224308



(a) System of Encrypted EMR System (b) System of Encrypted EMR System with PEKS Index

Fig. 1. Electronic medical record system based on cloud service.

Dual-servers based PEKS schemes [3], [4], [5] were proposed to counter the KGAs, but limited by the application scenario and with higher cost of cloud service. Secure channel-free PEKS schemes [6] resist outside adversaries from launching KGAs, but are unable to intercept KGAs from inside adversaries, e.g. attacks from staff of the CSP. Public key authenticated encryption with keyword search (PAEKS) [7] could be a safer alternative of PEKS to counterattack KGAs regardless of whether from outside or inside PPT adversaries. A number of references [8], [9], [10], [11], [12], [13], [14], [15], [16], [17], [18], and [19] on PAEKS demonstrate its scalability, however, a main shortcoming of PAEKS remains unresolved. The number of required trapdoors is linearly expanded as the number of senders increases. With each query, the user needs to generate multiple trapdoors for the same keyword. In the EMR system, the more doctors there are, the more trapdoors are required, which hinders the application and has motivated our work.

A. Motivation, Challenge and Contributions

This work aims to build a provably secure and practically efficient searchable public key encryption scheme for cloud computing and cloud storage applications with privacy protection requirements, represented by the hospital EMR system.

1) Motivation: The public key authenticated encryption with keyword search (PAEKS) [7] satisfies the security requirements and counterattacks the off-line keyword guessing attacks launched by semi-trusted cloud server, but are still some drawbacks to be concerned.

- The spatial complexity of the required trapdoor for retrieval is linear with the number of senders, which is not practical in scenarios of multiple senders. To query a keyword, the receiver needs to generate multiple trapdoors, one per sender, in case there are multiple senders. It is thus not efficient and of heavy trapdoor communication overhead if the number of senders is large. Besides, it is complex to generate different trapdoors corresponding to different senders in each query and for the same keyword. Overall, it is important to implement constant-size trapdoor generation for PAEKS to reduce the communication overhead and management complexity.
- The searching time per query is linear with the number of encrypted keywords as well, which is not efficient if there are a large number of files to be encrypted and each file contains multiple keywords. Therefore, it is reasonable and necessary to further improve the retrieval efficiency.

PAEKS counterattacks the inside KGAs benefiting from the necessary of the sender's secret key during the encryption.

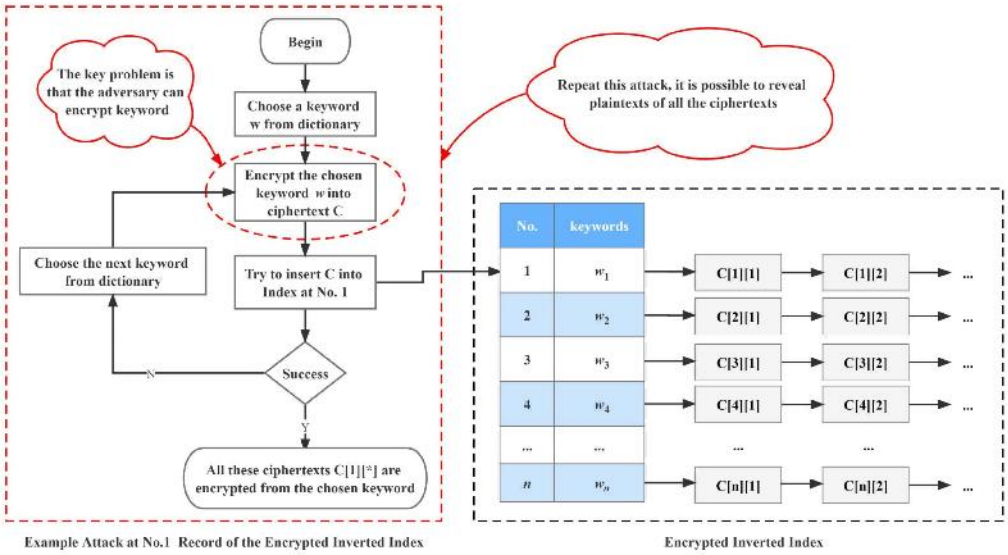


Fig. 2. Ciphertext inserting attacks on PEKS with inverted index.

However, it decreases the computation efficiency of trapdoor generation and increases the communication overhead of trapdoor transmission.

To the best of our knowledge, existing PAEKS schemes in literature fail to support constant size trapdoor generation for multiple senders. Han et al. [20] proposed a PAEKS scheme which supports logarithmic searching efficiency, but does not support constant size trapdoor generation. It is interesting and important to design a PAEKS scheme to support constant size trapdoor and sub-linearly fast retrieval, which motivates this work.

2) Challenge: In the original definition of PAEKS [7], the trapdoor generation algorithm takes as input both the receiver's secret key and the sender's public key. Thus, it is an inherent issue of PAEKS that the trapdoor size per query would linearly expand as the number of senders increases. The issue does not exist in those PEKS schemes which are vulnerable to inside KGAs, because neither encryption nor trapdoor generation include the sender's information in the algorithm, and the trapdoor is generated independently of the sender. On the other hand, although inverted index is often used in symmetric-key based secure data searching schemes, it is still challenging to achieve sub-linear searching efficiency in PEKS via applying inverted index.

- To reduce the communication overhead of trapdoor in PAEKS, it is natural to consider how to transform ciphertexts from different senders into a unified form so that the searching process does not need to consider the sender's information of a ciphertext. In the meanwhile, the transform should reserve the security of inside KGAs resistance.
- While applying the inverted index into PEKS, it would be vulnerable to the ciphertext inserting attacks, as illustrated in Figure 2.

Hence, it is not a trivial job to come up with a PAEKS scheme achieving the aforementioned goals.

3) Contributions: In this paper, we present an efficient solution to the above problem. The main idea is to add a non-collusion proxy to update the received ciphertexts and convert ciphertexts from different senders into a unified form. The contributions of this paper are summarized as follows.

- **PAUKS.** We introduce a new notion called *public key authenticated encryption with ciphertext update and keyword search* (PAUKS) in order to reduce the trapdoor communication overhead. The proposed PAUKS meets the needs of secure electronic medical record system of hospitals.
- **Security.** We give the threat model and the formally defined security model for PAUKS. The security defined in this paper requires a PAUKS scheme should counterattacks attacks launched by both the outside adversary and the inside adversary. Besides, the proxy in PAUKS is also considered as a threatening party and can be covered in the security model with the restriction that the proxy cannot collude with senders nor the cloud server.
- **Concrete Scheme.** We give a concrete PAUKS scheme based on the CDH assumption and a variant DLIN assumption. The proposed PAUKS scheme inherits features of PAEKS and meanwhile supports constant trapdoor communication overhead per query. We give formal security analysis of the proposed PAUKS scheme

shown in appendix and prove it to be secure in the proposed security model.

- **Efficiency.** It is worth noting that the proposed PAUKS scheme supports sub-liner searching efficiency, which is a rare property of PEKS or PAEKS schemes in the literature. The updated ciphertext can be securely sorted and inserted into an inverted index, searching based on inverted index will be much faster than searching directly over the whole encrypted-keyword space. Note that, leveraging inverted index to accelerate searching process is a natural method in scope of plaintext search, but challenging in PEKS or PAEKS, because it may face ciphertext inserting attacks. The proposed PAUKS scheme is proved secure with the inverted index.
- **Experimental Evaluation.** We evaluate the performance of our PAUKS scheme and HL-PAEKS scheme. The running efficiency of **Enc, Trapdoor** and **Test** algorithms of our PAUKS scheme are comparable with that of HL-PAEKS scheme. Differently, our PAUKS scheme enables a proxy to update the received ciphertexts. Before update, the running time of trapdoor generation per query is linear with the number of senders in both HL-PAEKS and our proposed PAUKS scheme. After update, the overhead of generating trapdoor will be decreased to constant which is no longer related to the number of senders. Experiments show that, when the number of senders is greater than a value (approximately 20), the running time of trapdoor generation before update is significantly greater than the time after update. Moreover, in our PAUKS scheme, the server can build a secure inverted index for fast search after update. With the inverted index, the search overhead is just linear with the size of keyword space and sub-linear with the total number of received searchable ciphertexts. Once the keyword space is much smaller than the received ciphertexts, the searching time based on inverted index will be significantly faster than the directly searching time.

B. Related Work

Boneh et al. [1] firstly introduced the notion of searchable encryption into the public key settings, and proposed the first public key encryption with keyword search (PEKS) scheme denoted by BDOP-PEKS. Byun et al. [2] pointed out the recent PEKS schemes were vulnerable against the proposed offline keyword guessing attacks (KGAs). Baek et al. [21] revisited the BDOP-PEKS scheme and gave a secure-channel free PEKS scheme which resists the outsider adversaries launching KGAs Xu et al. [22] proposed a fuzzy keyword search scheme based on PEKS and resists KGAs from outsid adversaries. However, the aforementioned schemes are still insecure under KGAs by insider adversaries.

Chen et al. [4] applied two server to counterattack the inside KGAs. In their scheme, the Keyword Server (KS) is separated from the Storage Server (SS). Based on the deterministic blind signature, Chen et al. [4] achieved the security against KS and SS. However, in their scheme, senders and receivers have to interact with the KS before storing ciphertexts or requesting searching results. Chen et al. [3] proposed another dual-server PEKS scheme which is secure against KGAs from both of the two servers. Based on a new proposed linear-and-homomorphic smooth projective hash function, Chen et al.'s scheme [3] lets a front server to preprocess the trapdoor and ciphertexts and deliver an internal test state to the back server. The back server finally returns the testing results to the receiver.

Huang and Li [7], based on PEKS, proposed a new primitive named public authenticated encryption with keyword search (PAEKS), which is secure against inside KGAs in the single-server setting. Noroozi and Eslami [23] pointed out their PAEKS scheme [7] fails to resist inside KGAs. Qin et al. [11] revisited the PAEKS scheme and proposed new security definitions, e.g. multi-ciphertext indistinguishability and multi-trapdoor privacy. They also provided a concrete scheme meeting the new definitions. Pan and Li [24] followed Qin et al.'s work [11] and proposed another PAEKS scheme. Chen et al., [25] proposed a PAEKS scheme to apply in specific scenarios, which is based on dual servers and enjoys a higher security. Lu et al. [26] proposed a secure-channel-free PAEKS scheme to prevent outside adversaries from obtaining any information about the search pattern of users. Guo et al. [27] improved the security of secure-channel-free PAEKS scheme and achieved multi-ciphertext indistinguishability.

Li et al. [10] introduced PAKES into the identity-based cryptographic settings to resolve the complex public-key certification problem and proposed an identity-based authenticated encryption scheme with keyword search (IBAEKS). Liu et al. [28] proposed another IBAEKS scheme which enjoys a higher running efficiency. He et al. [8] introduced PAEKS into certificateless public key setting and proposed a CLPAEKS scheme, removing the barrier of complex key management. Liu et al. [29] reviewed He et al.'s CLPAEKS scheme and improved the security. Wu et al. [9] proposed an improved CLPAEKS scheme which only allows the designated server to do the test and enjoys a better running efficiency than the preceding CLPAEKS scheme. Yang et al. [16] and Lu et al. [30] proposed PAEKS schemes without using bilinear pairings, suitable for Industrial Internet of Things scenarios.

Behnia et al. [31] proposed an efficient lattice-based PEKS scheme to resist quantum computing attacks. However, it does not secure against inside KGAs. Liu et al. [32] recently proposed a PAEKS scheme based on lattice assumption to resist inside KGAs and quantum computing attacks.

Proxy re-encryption (PRE), introduced by Blaze et al. [33] in 1998, is an important technique used in our scheme. It allows a proxy to re-encrypt a ciphertext for receiver Alice into another ciphertext for receiver Bob without decryption. Shao et al. [34] introduced PRE into public key searchable encryption settings. Chen et al. [35] designed access control strategy for the PRE based PEKS scheme. Hoang et al. [36] leveraged a secure enclave on the cloud server to securely perform the proxy work and minimize the bottleneck of network. Hoang et al. [37] proposed a searchable based the secure enclave supporting oblivious search and update.

Xu et al. [38] applied the PRF based PEKS scheme into the EMR system. Deng et al. [39] achieved a secure-channel-free PEKS scheme supporting proxy re-encryption.

II. Preliminaries

Before introducing our proposed searchable encryption scheme, in this section, we some necessary preliminaries used in this paper.

A. Bilinear Pairing

Assume p is a large prime and $\mathbb{G}, \mathbb{G}_T$ are two groups with order p. Bilinear pairing [1] is a map $\hat{e} : \mathbb{G} \times \mathbb{G} \rightarrow \mathbb{G}_T$ satisfying the following conditions:

- Bilinearity: $\forall x, y \subset \mathbb{Z}_p, g \subset \mathbb{G}, \hat{e}(g^x, g^y) = \hat{e}(g, g)^{xy}$;
- Non-degeneracy: $\forall g \neq 1, h \neq 1 \in \mathbb{G}, \hat{e}(g, h) \neq 1$;
- Computability: $\forall g, h \in \mathbb{G}$, $\hat{e}(g, h)$ is efficiently computable.

B. CDH Assumption

Let $\mathbb{G}$ be a group with a large prime order p. Given a CDH tuple (g, g^a, g^b), the CDH problem [40], [41] is to compute g^{ab}, where $g \in \mathbb{G}$ is a generator and $(a, b) \in \mathbb{Z}_p^2$ are randomly selected [40], [41].

CDH Assumption: The above CDH problem is intractable for any probabilistic polynomial time (PPT) adversary.

C. DLIN Assumption

Let $\hat{e} : \mathbb{G} \times \mathbb{G} \rightarrow \mathbb{G}_T$ be a bilinear pairing. Given a tuple $(g, g^x, g^y, g^{x \cdot r}, g^{y \cdot s}, Z \in \mathbb{G})$, the DLIN problem [40], [41] is to distinguish whether $Z = g^{r+s}$ or is a random element of $\mathbb{G}$, where $g \in \mathbb{G}$ is a generator and $(x, y) \in \mathbb{Z}_p^2$ are randomly selected.

DLIN Assumption: The above DLIN problem is intractable for any PPT adversary.

D. A Variant of DLIN Assumption

Let $\hat{e} : \mathbb{G} \times \mathbb{G} \rightarrow \mathbb{G}_T$ be a bilinear pairing. Given a modified DLIN tuple $(g, g^x, g^y, g^{x \cdot r}, g^{r+s}, Z \in \mathbb{G})$, the modified DLIN problem [40], [41] is to distinguish whether $Z = g^{r \cdot s}$ or not, where $g \in \mathbb{G}$ is a generator and $(x, y) \in \mathbb{Z}_p^2$ are randomly selected.

mDLIN Assumption: The above modified DLIN problem is intractable same as the DLIN problem.

III. The Public Key Authenticated Encryption With Ciphertext Update and Keyword Search

In this section, we propose a new cryptography encryption primitive named *public key authenticated encryption with ciphertext update and keyword search* (PAUKS) and introduce the framework, system model, threat model and security model of the proposed PAUKS.

A. PAUKS Framework

A basic PAUKS scheme contains ten algorithms as follows. The first six algorithms are inherited from the PAEKS scheme which guarantee any inside PPT adversary cannot successfully break the scheme via off-line keyword guessing attacks; The last four algorithms are designed to update ciphertexts and to support constant trapdoor communication overhead.

- **Setup**(1^λ): Given the security parameter 1^λ, this algorithm initializes the system and generates the public parameter $\mathbb{PP}$.
- **KeyGen$_\mathbf{R}$**$(\mathbb{PP})$: Given the public parameter $\mathbb{PP}$, this algorithm returns a receiver's public/secret key pair $(pk_\mathbf{R}, sk_\mathbf{R})$.
- **KeyGen$_\mathbf{S}$**$(\mathbb{PP})$: Given the public parameter $\mathbb{PP}$, this algorithm returns a sender's public/secret key pair $(pk_\mathbf{S}, sk_\mathbf{S})$.
- **Enc**$(\mathbb{PP}, sk_\mathbf{S}, pk_\mathbf{R}, w)$: Given the public parameter $\mathbb{PP}$, a sender's secret key $sk_\mathbf{S}$, a receiver's public key $pk_\mathbf{R}$ and a keyword w, this algorithm returns a PAEKS ciphertext C.
- **Trapdoor**$(\mathbb{PP}, sk_\mathbf{R}, pk_\mathbf{S}, w')$: Given the public parameter $\mathbb{PP}$, a receiver's secret key $sk_\mathbf{R}$, a sender's public key $pk_\mathbf{S}$ and a keyword w', this algorithm returns a trapdoor $T_{w'}$.
- **Test**$(\mathbb{PP}, pk_\mathbf{S}, C, T_w)$: Given the public parameter $\mathbb{PP}$, a sender's public key $pk_\mathbf{S}$, the ciphertext C and trapdoor T_w generated with $pk_\mathbf{S}$, this algorithm returns 1 if C and T_w are generated from the same keyword, 0, otherwise.
- **UpdKeyGen**$(\mathbb{PP}, sk_\mathbf{R}, pk_\mathbf{S})$: Given the public parameter $\mathbb{PP}$, the receiver's secret key $sk_\mathbf{R}$ and a candidate sender's public key $pk_\mathbf{S}$, this algorithm returns an updating key $uk_\mathbf{S}$ attached to the sender.
- **UpdEnc**$(\mathbb{PP}, C_\mathbf{S}, uk_\mathbf{S})$: Given the public parameter $\mathbb{PP}$, a ciphertext $C_\mathbf{S}$ sent from sender $\mathbf{S}$, and the updating key $uk_\mathbf{S}$ grant for $\mathbf{S}$, this algorithm updates $C_\mathbf{S}$ and returns an updated ciphertext $\hat{C}$.
- **ConstTrapdoor**$(\mathbb{PP}, sk_\mathbf{R}, w')$: Given the public parameter $\mathbb{PP}$, the receiver's secret key $sk_\mathbf{R}$ and a keyword w', this algorithm returns a constant trapdoor $\hat{T}_{w'}$.
- **UpdTest**$(\mathbb{PP}, \hat{C}, \hat{T}_{w'})$: Given the public parameter $\mathbb{PP}$, an updated ciphertext $\hat{C}$ and a constant $\hat{T}_{w'}$, this algorithm returns 1 if $\hat{C}$ and $\hat{T}_{w'}$ contain the same keyword, 0, otherwise.

B. System Model

The system model of the PAUKS, as shown in Figure 3, contains five types of parties: multiple data senders (Alice, Bob, Cindy, etc.), a data receiver (an administrator of a hospital), a proxy (a sub-administrator), a cloud server and an external data user (e.g. researchers).

1) Ability of Parties: We demonstrate the ability of each party of PAUKS as follows.

- **Data senders**: There are multiple senders, e.g. Alice, Bob, Cindy shown in Figure 3.
  - Each sender can encrypt keywords with the secret key of itself and the public key of the receiver.
  - ach sender is able to monitor the channel and capture ciphertexts sent from other senders and trapdoors submitted by the receiver.

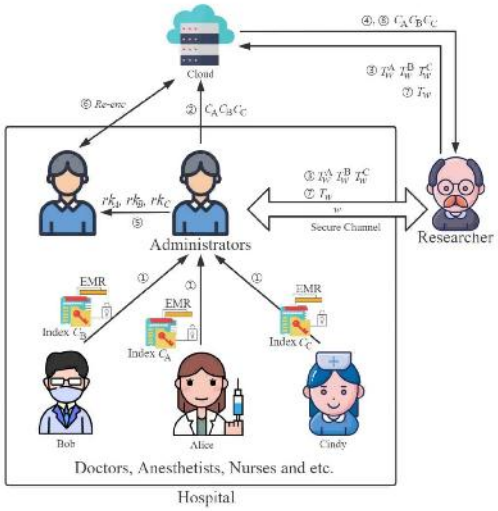


Fig. 3. System model of PAUKS.

- **Data receiver**: We consider only one receiver in PAUKS system for simplicity.
  - The receiver is the only user who can generate trapdoor using the secret key of itself and the public keys of senders.
  - The receiver in the proposed scenario could be an administrator of the hospital, who roles a fully trusted party.
- **Cloud server**: There is a cloud server in PAUKS system.
  - The server provides almost unlimited storage space and stores all the received ciphertexts.
  - The server will test whether there is any ciphertext matching with the submitted trapdoor.
  - The server is semi-trusted. It may try to reveal the information behind ciphertexts and trapdoors.
- **Proxy**: There is a proxy in PAUKS system to update the ciphertexts stored in the cloud.
  - The proxy could be a sub-administrator of the hospital and it help the administrator to update the ciphertexts.
  - The proxy could be semi-trusted. It means that the proxy will honestly perform the given task but may be curious about the plaintext information behind ciphertexts or trapdoors.
  - The restriction is that the proxy cannot collude with any sender nor the cloud server.
- **External user**: There could be one or more external users trying to utilize the EHR system to do medical research or data analysis. The external users are given limited authorization to access the data on most occasions.
  - The external user could be a researcher who would like to access some data.
  - The external user will be given a trapdoor containing the candidate keywords to search.
  - The external user can delegate the server to search files with the trapdoor and move on the next process, e.g. querying decryption keys of the files from the administrator, which is out of scope of this work.

2) System Flow: The PAUKS system runs after setup procedure supervised by a trusted certification authority (CA). All users' public keys are certificated and broadcasted by the CA. The system flow is shown as follows.

- At the beginning of the PAUKS system, a sender, e.g. Alice, encrypts patient EMRs with some kind of encryption scheme and encrypts index with the **Enc** algorithm of PAUKS. Alice concatenates the encrypted file and the encrypted keywords as a ciphertext, and finally sends the ciphertext to the administrator. Same with Alice, more senders send ciphertexts to the administrator (step ①). The administrator uploads ciphertexts to the cloud (step ②), to save local storage space.
- The administrator can generate trapdoors, e.g. T_w^A, T_w^B, T_w^C, with any candidate keyword w. Doctors in the hospital can query ciphertexts stored in cloud given trapdoors by the administrator. Researcher who was granted access of the ciphertexts can also obtain trapdoors from the administrator and query the matching ciphertexts stored in the cloud (step ③ ④). However, as shown in step ③, the overhead of generating trapdoors and the number of trapdoors per query per keyword is linear with the number of senders.
- The administrator can generate updating key for each sender (step ⑤) and delegate a sub-administrator to update the stored ciphertexts in the cloud (step ⑥). The update process will increase search efficiency and reduce communication overhead, meanwhile, it reserves the format of the original ciphertexts and still supports PAEKS-like test.
- After update, the administrator can generate constant size trapdoor, e.g. T_w, per query per keyword, and the server still returns the matching ciphertexts (step ⑦,⑧). The communication and computation overhead of search queries will be theoretically reduced.
- As an extension, after update, it is possible to classify the ciphertexts by the embedded keywords without decryption and against keyword guessing attacks. It means that an inverted index could be build to accelerate the search process.

C. Threat Model

Based on the system model of PAUKS, we analyse the threats from different adversaries.

1) Adversaries: The receiver is assumed honest naturally, and the external user with limited access permission can be considered as low risky. Three types of adversaries as follows are considered in the PAUKS.

- **Outside adversary**: it could be one of the senders trying to reveal plaintexts from trapdoors or ciphertexts encrypted by other senders.
- **Inside adversary**: it could be an adversary who can access in the inside network of the cloud storage service.

The inside adversary would try to reveal plaintexts from trapdoors, ciphertexts or updated ciphertexts.

- **Non-collusion proxy**: It could be a sub-administrator and should be a semi-trusted and non-collusion party. The proxy will honestly execute the proposed protocol but try to reveal information from ciphertexts or trapdoors; The proxy cannot collude with senders nor the cloud server, which is a reasonable as a sub-administrator in the real-life world.

2) Security Requirements: According to the threat model, the security requirements are as follows.

IKGA Resistance: We consider security against keyword guessing attacks launched by an inside adversary or the semi-trusted collusion-free proxy. Notice that security against KGA from an outside adversary could be implied by that from an inside adversary. Therefore, we do not discuss it in the rest of the paper.

- For any PPT adversary who does not have the knowledge of the secret keys and updating keys of the corresponding senders and receiver, it is of a negligible advantage to distinguish whether or not two ciphertexts (before update) contain the same keyword unless the matching trapdoors are given.
- For any PPT adversary who does not have knowledge of the updating keys and the receiver's secret key, the probability of revealing keyword from a trapdoor is negligible, even under the off-line keyword guessing attacks.

Non-collusion Proxy: Another security requirement is that the proxy with the updating keys cannot obtain more information than outside adversaries from the ciphertexts or trapdoors. That is, for any PPT adversary who does not have knowledge of the receiver's secret key, the probability of revealing the plaintext from the updated ciphertexts is negligible.

Remark: The secure enclave proposed by Hoang et al. [36] could be an alternative of a trusted proxy residing on the server side. To weaken the security assumption, it is enough to employ a semi-honest proxy who will not collude with the server and any sender for the proposed system. It is also possible to delegate the semi-honest cloud server as the proxy in the future work.

D. Security Model

Based on the threat model and security requirements, we define the security of PAUKS as follow.

Definition 1: A PAUKS scheme is semantically secure against the off-line keyword guessing attacks if for any PPT adversary $\mathcal{A}$, the advantage to win the following games is negligible.

Security Game 1 (Ciphertext-Indistinguishability Without Update, IND-CCA): In this game, the adversary $\mathcal{A}$ tries to distinguish ciphertexts without update.

- **Setup**: Given the security parameter 1^λ, return the public parameter $\mathbb{PP}$ and the public keys $pk_{\mathbf{S}}$, $pk_{\mathbf{R}}$ of a random sender and the receiver.
- **Phase 1**: Given the public parameter $\mathbb{PP}$, the adversary $\mathcal{A}$ can do the following queries.
  - *Ciphertext Query*: Given a keyword, return a corresponding PAUKS ciphertext encrypted with $sk_{\mathbf{S}}$ and $pk_{\mathbf{R}}$.
  - *Trapdoor Query*: Given a keyword, return a corresponding PAUKS trapdoor generated with $pk_{\mathbf{S}}$ and $sk_{\mathbf{R}}$.
- **Challenge**: $\mathcal{A}$ submits two keywords w_0^*, w_1^* which have not been queried for trapdoors. The simulator tosses a coin $b \leftarrow \{0,1\}$, encrypts w_b^* with $sk_{\mathbf{S}}$ and $pk_{\mathbf{R}}$, and returns the ciphertext C_b^*.
- **Phase 2**: $\mathcal{A}$ can also do the queries same as in Phase 1, except that w_0^*, w_1^* could not appear in trapdoor queries.
- **Guess**: Finally, $\mathcal{A}$ returns a bit $b' \leftarrow \{0,1\}$ and wins the game if $b' = b$. The advantage that $\mathcal{A}$ wins the game is defined as $Adv_{\mathcal{A}}^{C} = \left|\mathbf{Pr}[\mathcal{A}_{\text{win}}] - \frac{1}{2}\right| = \left|\mathbf{Pr}[b' = b] - \frac{1}{2}\right|$.

Security Game 2 (Updated-Ciphertext-Indistinguishability, IND-U-CCA): In this game, the adversary $\mathcal{A}$ tries to reveal plaintexts from the updated ciphertexts.

- **Setup**: Same as above.
- **Phase 1**: Given the public parameter $\mathbb{PP}$, the adversary $\mathcal{A}$ could issue ciphertext and trapdoor queries as in the above game.
- **Challenge**: $\mathcal{A}$ submits two keywords w_0^*, w_1^* which have not been queried for trapdoors. The simulator tosses a coin $b \leftarrow \{0,1\}$, encrypts w_b^* with $sk_{\mathbf{S}}$ and $pk_{\mathbf{R}}$ to get a ciphertext C_b^*. With the updating key $uk_{\mathbf{S}}$, the simulator updates the ciphertext C_b^*, and returns the updated ciphertext $\hat{C}_b^*$.
- **Phase 2**: $\mathcal{A}$ continues to query as in Phase 1, except that w_0^*, w_1^* cannot appear in trapdoor queries.
- **Guess**: Finally, $\mathcal{A}$ returns a bit $b' \leftarrow \{0,1\}$ and wins the game if $b' = b$. The advantage that $\mathcal{A}$ wins the game is defined as $Adv_{\mathcal{A}}^{\hat{C}} = \mathbf{Pr}[\mathcal{A}_{\text{win}}] - \frac{1}{2} = \mathbf{Pr}[b' = b] - \frac{1}{2}$.

Security Game 3 (Trapdoor Privacy, IND-TP-CCA): In this game, the adversary $\mathcal{A}$ tries to reveal plaintexts from the trapdoors (both with and without update).

- **Setup**: Same as above.
- **Phase 1**: Given the public parameter $\mathbb{PP}$, the adversary $\mathcal{A}$ can do the ciphertext and trapdoor queries same the above game. Differently, the returned ciphertexts have been updated.
- **Challenge**: $\mathcal{A}$ submits two keywords w_0^*, w_1^* which have not been queried for ciphertexts nor trapdoors. The simulator tosses a coin $b \leftarrow \{0,1\}$, returns two trapdoors $T_{w_b^*}, \hat{T}_{w_b^*}$.
- **Phase 2**: $\mathcal{A}$ continues to query as in Phase 1, except that w_0^*, w_1^* could not appear in both ciphertext and trapdoor queries.
- **Guess**: Finally, $\mathcal{A}$ returns a bit $b' \leftarrow \{0,1\}$ and wins the game if $b' = b$. The advantage that $\mathcal{A}$ wins the game is defined as $Adv_{\mathcal{A}}^{T} = \left|\mathbf{Pr}[\mathcal{A}_{\text{win}}] - \frac{1}{2} = \mathbf{Pr}[b' = b] - \frac{1}{2}\right|$.

IV. Our PAUKS Scheme

- **Setup**(1^λ) : Given the security parameter 1^λ, output the public parameter $\mathbb{PP} = \{p, g, \mathbb{G}, \mathbb{G}_T, \hat{e}, H, H_1, H_2, H_3, H_4\}$ shown as follows:
  - p is a large prime,
  - $\mathbb{G}, \mathbb{G}_T$ are groups with order p,

– $\hat{e}: \mathbb{G} \times \mathbb{G} \to \mathbb{G}_T$ is a bilinear pairing,
– H, H_1, H_2, H_3, H_4 are hash functions:
  * $H: \mathbb{G} \times \mathbb{G} \times \mathbb{G} \times \mathbb{G} \to \mathbb{G}$,
  * $H_1: \mathbb{G} \times \{0,1\}^* \to \mathbb{Z}_p$,
  * $H_2: \mathbb{G} \to \mathbb{Z}_p$,
  * $H_3: \mathbb{G} \to \mathbb{Z}_p$,
  * $H_4: \{0,1\}^* \to \mathbb{G}$.

- **KeyGen$_\mathbf{R}$**($\mathbb{PP}$) : Given $\mathbb{PP}$, the algorithm randomly selects x_1, x_2, x_3, x_4 from $\mathbb{Z}_p$ and outputs the receiver's public key $pk_\mathbf{R} = (pk_{\mathbf{R}_1}, pk_{\mathbf{R}_2}, pk_{\mathbf{R}_3})$ and secret key $sk_\mathbf{R} = (sk_{\mathbf{R}_1}, sk_{\mathbf{R}_2}, sk_{\mathbf{R}_3}, sk_{\mathbf{R}_4})$ as follows:
  – $pk_{\mathbf{R}_1} = g^{x_1}$, $pk_{\mathbf{R}_2} = g^{x_2}$, $pk_{\mathbf{R}_3} = g^{x_3}$,
  – $sk_{\mathbf{R}_1} = x_1, sk_{\mathbf{R}_2} = x_2, sk_{\mathbf{R}_3} = x_3, sk_{\mathbf{R}_4} = x_4$.
- **KeyGen$_\mathbf{S}$**($\mathbb{PP}$) : Given the public parameter $\mathbb{PP}$, the algorithm randomly selects $y \xleftarrow{\$} \mathbb{Z}_p$ and outputs the sender's public key $pk_\mathbf{S} = g^y$ and secret key $sk_\mathbf{S} = y$.
- **Enc**($\mathbb{PP}, sk_\mathbf{S}, pk_\mathbf{R}, w$) : Given the public parameter $\mathbb{PP}$, a sender's secret key $sk_\mathbf{S}$ and the receiver's public key $pk_\mathbf{R}$, the algorithm randomly selects $r_1, r_2 \leftarrow \mathbb{Z}_p$ and outputs the ciphertext $C = (C_1, C_2, C_3, C_4)$ as follows:

$$C_1 = \left(pk_{\mathbf{R}_2}^{H_1(pk_{\mathbf{R}_1}^{sk_\mathbf{S}}, w)} \cdot pk_{\mathbf{R}_3} \right)^{r_1}, \quad C_2 = g^{r_1},$$
$$C_3 = \left(pk_{\mathbf{R}_2}^{H_1(pk_{\mathbf{R}_1}^{sk_\mathbf{S}})} \cdot pk_{\mathbf{R}_3} \right)^{r_2} \cdot g^{H_3(pk_{\mathbf{R}_1}^{sk_\mathbf{S}}) \cdot r_1},$$
$$C_4 = H_4(w)^{r_2}, \quad C_5 = H(C_1, C_2, C_3, C_4)^r.$$

- **Trapdoor**($\mathbb{PP}, sk_\mathbf{R}, pk_\mathbf{S}, w'$) : Given the public parameter $\mathbb{PP}$, the receiver's secret key $sk_\mathbf{R}$ and a sender's public key $pk_\mathbf{S}$, the algorithm randomly selects $r_3 \leftarrow \mathbb{Z}_p$ and returns a trapdoor $T_w = (T_{w,1}, T_{w,2})$ as follows:

$$T_{w,1} = g^{\frac{r_3}{sk_{\mathbf{R}_2} \cdot H_1(pk_\mathbf{S}^{sk_{\mathbf{R}_1}}, w') - sk_{\mathbf{R}_3}}}, \quad T_{w,2} = g^{r_3}.$$

- **Test**($\mathbb{PP}, C, T_w$) : Given the public parameter $\mathbb{PP}$, a ciphertext C and a trapdoor T_w, the algorithm returns 1 if the following equation holds, otherwise, returns 0:

$$\hat{e}(C_1, T_{w,1}) = \hat{e}(C_2, T_{w,2}).$$

- **UpdKeyGen**($\mathbb{PP}, sk_\mathbf{R}, pk_\mathbf{S}$) : Given $\mathbb{PP}$, the receiver's secret key $sk_\mathbf{R}$ and a sender's public key $pk_\mathbf{S}$, the algorithm parses $sk_\mathbf{R}$ as $(sk_{\mathbf{R}_1}, sk_{\mathbf{R}_2}, sk_{\mathbf{R}_3}, sk_{\mathbf{R}_4})$ and generates an updating key $uk_\mathbf{S} = (uk_{\mathbf{S}_1}, uk_{\mathbf{S}_2})$ as follows:

$$uk_{\mathbf{S},1} = H_3(pk_\mathbf{S}^{sk_{\mathbf{R}_1}}),$$
$$uk_{\mathbf{S},2} = \frac{sk_{\mathbf{R}_4}}{sk_{\mathbf{R}_2} \cdot H_2(pk_\mathbf{S}^{sk_{\mathbf{R}_1}}) + sk_{\mathbf{R}_3}}.$$

- **UpdEnc**($\mathbb{PP}, C, uk_\mathbf{S}$) : Given $\mathbb{PP}$, a ciphertext C sent from sender $\mathbf{S}$ and the updating key $uk_\mathbf{S}$ assigned for the sender $\mathbf{S}$, the algorithm parses C as $(C_1, C_2, C_3, C_4, C_5)$ and returns $\perp$ if the following equation (1) does not hold, which means the ciphertext was tampered by adversaries.

$$\hat{e}(H(C_1, C_2, C_3, C_4), C_2) = \hat{e}(C_5, g). \tag{1}$$

Otherwise, this algorithm returns an updated ciphertext $\hat{C} = (C, C_6)$, where

$$C_6 = (C_3 / C_2^{uk_{\mathbf{S}_1}})^{uk_{\mathbf{S}_2}} = g^{r_2 \cdot sk_{\mathbf{R}_4}}.$$

Fig. 4. Header $\mathcal{H}$ of the empty index $\mathcal{I}$.

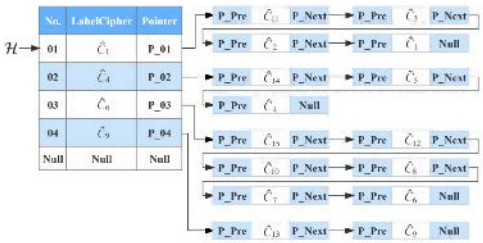


Fig. 5. Encrypted fast searching index.

- **ConstTrapdoor**($\mathbb{PP}, sk_\mathbf{R}, w'$) : Given $\mathbb{PP}$, the receiver's secret key $sk_\mathbf{R}$ and a keyword w', the algorithm randomly selects $r \leftarrow \mathbb{Z}_p$ and returns a constant trapdoor $\hat{T}_{w'} = (\hat{T}_{w',1}, \hat{T}_{w',2})$ applicable to matching ciphertexts sent from different senders, where

$$\hat{T}_{w',1} = g^{sk_{\mathbf{R}_4} \cdot r}, \quad \hat{T}_{w',2} = H_4(w')^r.$$

- **UpdTest**($\mathbb{PP}, \hat{C}, \hat{T}_{w'}$) : Given $\mathbb{PP}$, an updated ciphertext $\hat{C}$ and a constant trapdoor $\hat{T}_{w'}$, the algorithm returns 1 if the following equation holds, otherwise, returns 0:

$$\hat{e}(C_4, \hat{T}_{w',1}) = \hat{e}(C_6, \hat{T}_{w',2}).$$

A. Fast Searching Index for PAUKS

The updated ciphertexts can be inserted into an encrypted inverted index without decryption.

- **EqualityTest**($\mathbb{PP}, \hat{C}_1, \hat{C}_2$): Given $\mathbb{PP}$, and two updated ciphertexts $\hat{C}_1, \hat{C}_2$, this algorithm parses $\hat{C}_1, \hat{C}_2$ as $(C_{1,\dots}^{(1)} C_6^{(1)})$ and $(C_{1,\dots}^{(2)} C_6^{(2)})$, and returns 1 if the following equation holds, and 0 otherwise:

$$\hat{e}(C_4^{(1)}, C_6^{(2)}) = \hat{e}(C_4^{(2)}, C_6^{(1)}).$$

- **InitIndex**($\mathbb{PP}$): Given $\mathbb{PP}$, this algorithm initializes an empty index $\mathcal{I}$ with header

$$\mathcal{H} = (\text{No.}, \text{LabelCipher}, \text{Pointer}),$$

illustrated in Figure 4.
- **InsertIndex**($\mathbb{PP}, \hat{C}, \mathcal{H}$): Given $\mathbb{PP}$, an updated ciphertext $\hat{C}$ and the header $\mathcal{H}$ of the fast searching index, this algorithm inserts $\hat{C}$ into the index illustrated in Algorithm 1. After several rounds of insertion, the index $\mathcal{I}$ can be exampled in Figure 5.
- **FastSearch**($\mathbb{PP}, \mathcal{H}, \hat{T}_{w'}$): Given $\mathbb{PP}$, the header $\mathcal{H}$ of index $\mathcal{I}$ and a constant trapdoor $\hat{T}_{w'}$, this algorithm returns the a pointer (Pointer) if the ciphertexts pointed by the Pointer contain the same keyword as $\hat{T}_{w'}$, and $\perp$ otherwise. The detail algorithm is shown in Algorithm 2.

Algorithm 1 InsertIndex

Input: $\mathbb{PP}$, $\hat{C}$, $\mathcal{H}$ = (No., LabelCipher, Pointer)
Output: $\mathcal{H}$

```
ptr = H;
isEqual = 0;
while ptr ≠ Null && isEqual ≠ 0 do
    isEqual = EqualityTest(PP, ptr.LabelCipher, Ĉ);
    if isEqual then
        insert Ĉ into ptr.Pointer;
        return H;
    end if
    ptr = ptr.NextNode;
end while
if !isEqual then
    insert Ĉ into ptr.Pointer;
end if
return H;
```

Algorithm 2 FastSearch

Input: $\mathbb{PP}$, $\hat{T}_{w'}$, $\mathcal{H}$ = (No., LabelCipher, Pointer)
Output: Pointer or $\perp$

```
ptr = H;
isMatch = 0;
while ptr ≠ Null && isMatch ≠ 0 do
    isMatch = UpdTest(PP, ptr.LabelCipher, T̂_{w'});
    if isMatch then
        return ptr.Pointer;
    end if
    ptr = ptr.NextNode;
end while
if !isMatch then
    return ⊥;
end if
```

B. Correctness of the PAUKS Scheme

Assuming the hash function is collision resistant, the **Test** algorithm returns 1 if and only if the hash inputs are the same. The correctness can be proved by the following equations:

$$\hat{e}(C_1, T_{w,1}) = \hat{e}\left((pk_{\mathbf{R}_2}^{H_1(pk_{\mathbf{R}_1}^{sk_{\mathbf{S}}}, w)} \cdot pk_{\mathbf{R}_3})^{r_1}, g^{sk_{\mathbf{R}_2} \cdot H_1(pk_S^{r_2 sk_{\mathbf{R}_1}}, w') + sk_{\mathbf{R}_3}}\right) \xrightarrow{\text{iff}(w=w')} \hat{e}(g^{r_1}, g^{r_2}) = \hat{e}(C_2, T_{w,2}).$$

Parse $\hat{T}_{w',1}, \hat{T}_{w',2}$ and C_4, C_6 as follows:

$$\hat{T}_{w',1} = g^{sk_{\mathbf{R}_4} \cdot r}, \quad \hat{T}_{w',2} = H_2(w)^r, \quad C_4 = H_4(w)^{r_2},$$

$$C_6 = \left(pk_{\mathbf{R}_2}^{H_2(pk_{\mathbf{R}_1}^{sk_{\mathbf{S}}})} \cdot pk_{\mathbf{R}_3}\right)^{sk_{\mathbf{R}_2} \cdot H_2(pk_{\mathbf{S}}^{r_2 sk_{\mathbf{R}_4}}, pk_{\mathbf{R}_1}^{r_2}) + sk_{\mathbf{R}_3}} = g^{r_2 \cdot sk_{\mathbf{R}_4}}.$$

The **UpdTest** algorithm returns 1 if and only if the keyword embedded in the constant trapdoor is identical to that in the updated encrypted ciphertext, which can be proved by the following equations:

$$\hat{e}(C_4, \hat{T}_{w',1}) = \hat{e}(H_4(w)^{r_2}, g^{sk_{\mathbf{R}_4}}) \xrightarrow{\text{iff}(w'=w)} \hat{e}(g^{r_2 \cdot sk_{\mathbf{R}_4}}, H_4(w')^r) = \hat{e}(C_6, \hat{T}_{w,2}).$$

C. Security of the PAUKS Scheme

Theorem 1 (IKGA): The proposed PAUKS scheme is semantically secure against off-line keyword guessing attacks from any PPT adversary in the random oracle model if the CDH and mDLIN assumptions hold.

Similar to the HL-PAEKS scheme [7], the proposed PAUKS scheme prevents the adversary from encrypting keywords on behalf of other senders, which counterattacks the inside keyword guessing attacks. After update, the ciphertexts can only be tested with the constant trapdoor generated by the receiver, and the original ciphertexts without update cannot be tested with the constant trapdoor. Theorem 1 follows from the lemma 1, 2, and 3, which are given the formal proof in Appendix A.

TABLE I
COMMUNICATION OVERHEAD EVALUATION

Schemes	C	T_w	$\hat{C}$	$\hat{T}_w$
HL-PAEKS [7]	$2\lvert\mathbb{G}\rvert$	$m \cdot \lvert\mathbb{G}_T\rvert$	—	—
Our PAUKS	$5\lvert\mathbb{G}\rvert$	$m \cdot (2\lvert\mathbb{G}\rvert)$	$6\lvert\mathbb{G}\rvert$	$2\lvert\mathbb{G}\rvert$

$\lvert\mathbb{G}\rvert$: the size of an element in group $\mathbb{G}$.
$\lvert\mathbb{G}_T\rvert$: the size of an element in group $\mathbb{G}_T$.
m: the number of senders.

V. PERFORMANCE EVALUATION

We evaluate the communication and computation overhead of our PAUKS scheme compared with HL-PAEKS scheme [7], as shown in Table I and Table II. Before ciphertext update, the proposed PAUKS scheme has a higher but still comparable communication overhead in terms of the ciphertext and the trapdoor. However, after ciphertext update, the trapdoor communication overhead of our PAUKS scheme reduces to constant complexity and is significantly lower in comparison with [7] when the number of senders increases. On the other hand, although our PAUKS scheme has slightly larger computation overhead than HL-PAEKS scheme in terms of **Enc**, **Trapdoor** and **Test** algorithms, it enjoys a lower computation complexity when using **ConstTrapdoor** and **UpdTest** algorithms to realize ciphertext retrieval.

We perform experiments on a desktop computer with an Intel Core i7-6700 CPU (3.40 GHz), 8GB Memory. The operating system is the Ubuntu 20.04.3 LTS. We instantiated our scheme using C programming language with GNU Multiple Precision Arithmetic (GMP) library and Pairing-Based Cryptography (PBC) library. The initial HL-PAEKS scheme [7] was instantiated for comparison. The results are shown in Table III, Table IV, Table V, Figure 6 and Figure 7.

TABLE II
COMPUTATION OVERHEAD EVALUATION

Schemes	Enc	Trapdoor	Test	UpdEnc	ConstTrapdoor	UpdTest
HL-PAEKS [7]	3Exp + Hash	$m \cdot$ (Exp + Pair + Hash)	$n \cdot$ (2Exp + Mul)	—	—	—
Our PAUKS	9Exp + 5Hash +2Mul	$m \cdot$ (3Exp + Mul −Div + Add)	$2n \cdot$ Exp	$n \cdot$ (2Pair + Hash −2Exp − Div)	2Exp + Hash + Mul	2Pair

Exp: computation overhead of an exponential operation.
Hash: computation overhead of a hash operation.
Div: computation overhead of a division operation.
n: the number of received ciphertexts.
Pair: computation overhead of a bilinear pairing.
Mul: computation overhead of a multiplication operation.
Add: computation overhead of an addition operation.
m: the number of senders.

TABLE III
STORAGE COST OF KEYS, CIPHERTEXTS, AND TRAPDOORS

Schemes	Public Keys (Bytes)		Secret Keys (Bytes)		UpdKey (Bytes)	Ciphertexts (Bytes)		Trapdoors (Bytes)	
	Sender	Receiver	Sender	Receiver		Enc	UpdEnc	Trapdoor	ConstTrapdoor
HL-PAEKS [7]	128	128	20	20	—	256	—	128	—
Our PAUKS	128	384	20	80	40	640	768	256	256

TABLE IV
AVERAGE RUNNING TIME OF SETUP AND KEY GENERATION

Schemes	Setup Time (ms)	KeyGen Time (ms)		
		Sender	Receiver	UpdKeyGen
HL-PAEKS [7]	4.546	1.243	1.153	—
Our PAUKS	4.546	1.117	3.457	1.025

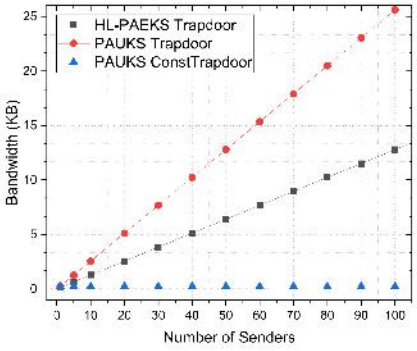


Fig. 6. Trapdoor bandwidth cost comparison.

Table III shows the storage cost of each key, ciphertext and trapdoor. The storage cost of the sender's keys in both HL-PAEKS and our PAUKS are the same. Due to extensional functions, our PAUKS scheme needs larger storage space than HL-PAEKS scheme in terms of the receiver's keys, ciphertexts and trapdoors. However, as shown in Figure 6, our PAUKS scheme supports constant trapdoor transmission after ciphertext update and needs much less bandwidth when the number of senders is large.

Table IV shows the running time of system initialization. The **Setup** algorithms are the same in both HL-PAEKS and our PAUKS, thus the running time are the same. The key generation algorithms for senders and receivers in HL-PAEKS scheme and for senders in our PAUKS scheme enjoy similar overhead; The key generation algorithm for receivers in our PAUKS scheme approximately runs three times as long. The **ReKeyGen** algorithm in our PAUKS scheme is as efficient as the key generation algorithm in HL-PAEKS scheme. Overall, compared with HL-PAEKS scheme, the system initialization overhead of our PAUKS scheme is higher, fortunately, each algorithm only needs to be run once and the total running time is negligible for users.

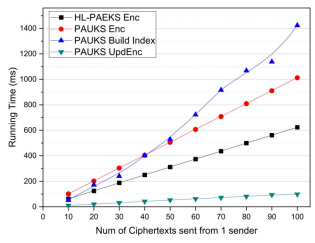
Table V shows the average running time of **Enc**, **Trapdoor**, **Test**, **UpdEnc** and **ConstTrapdoor** algorithms in the aforementioned two schemes. The data related to **UpdEnc** and **ConstTrapdoor** of HL-PAEKS is empty due that there is no such algorithms in HL-PAEKS scheme. In the comparison of **Enc** algorithms, the HL-PAEKS scheme enjoys a higher operation efficiency; In the comparison of **Trapdoor** algorithms, our PAUKS scheme leads a slight advantage. In the comparison of **Test** algorithms, the two schemes are almost tied. In our PAUKS scheme, the average running time of **UpdEnc** and **ConstTrapdoor** algorithms are close to that of **Enc** and **Trapdoor** algorithms, respectively.

From the experiment results shown in Figure 7(a)(b), the overhead of running **Enc** and **Trapdoor** algorithms are comparable between HL-PAEKS scheme and our PAUKS scheme. The overhead of the **Enc** algorithm in our PAUKS scheme is higher than but comparable with that in HL-PAEKS scheme because of longer ciphertext length and more functionality. The newly added algorithm **UpdEnc** enjoys a high running efficiency compared with the **Enc** algorithms. The PAUKS Build Index in Figure 7(a) shows the total overhead of building fast searching index and inserting index procedure, which is a bit higher than the **Enc** algorithms but reasonable.

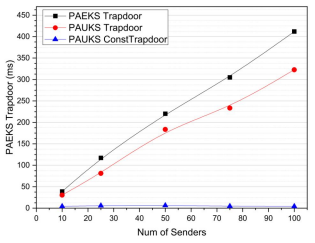
The running time of the **Trapdoor** algorithm in our PAUKS scheme is less than that in HL-PAEKS scheme. The **Trapdoor** running time in the two schemes are both linearly increase as the number of senders increases. The **ConstTrapdoor** algorithm is much more efficient and enjoys an almost constant running overhead as the number of senders increases.

TABLE V
ALGORITHMS AVERAGE RUNNING TIME COMPARASION

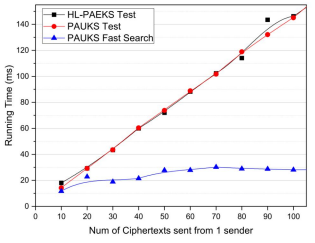
Schemes	**Enc** (ms)	**Trapdoor** (ms)	**Test** (ms)	**UpdEnc** (ms)	**ConstTrapdoor** (ms)
HL-PAEKS [7]	6.236	4.120	1.411	—	—
Our PAUKS	10.122	3.295	1.424	9.8514	4.302



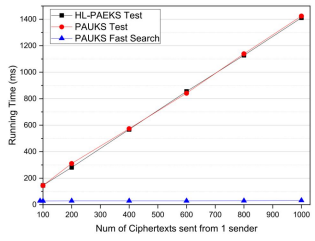
(a) Comparison of Encryption Time



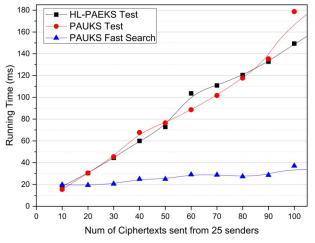
(b) Comparison of Trapdoor Generation Time



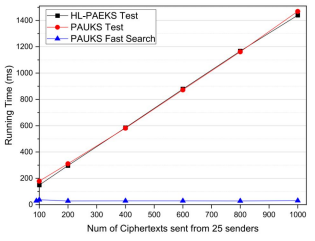
(c) Test Time with only 1 Sender



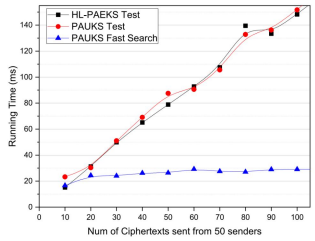
(d) Test Time with 1 Sender (more ciphertexts)



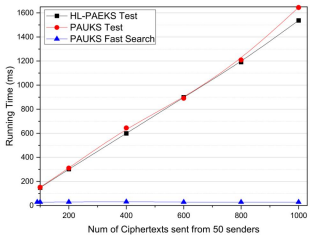
(e) Test Time with 25 Senders



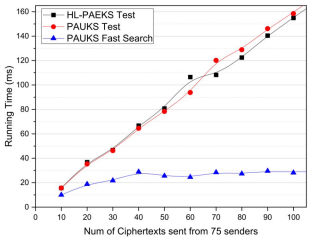
(f) Test Time with 25 Senders (more ciphertexts)



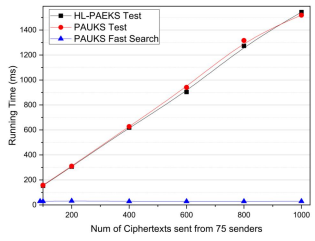
(g) Test Time with 50 Senders



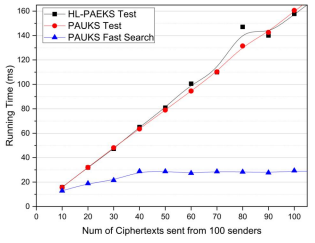
(h) Test Time with 50 Senders (more ciphertexts)



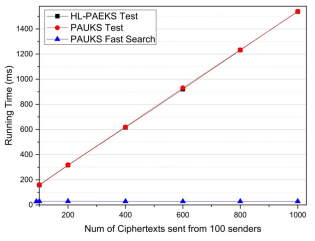
(i) Test Time with 75 Senders



(j) Test Time with 75 Senders (more ciphertexts)



(k) Test Time with 100 Senders



(l) Test Time with 100 Senders (more ciphertexts)

Fig. 7. Computation overhead comparison: **Enc**, **Trapdoor** and **Test**.

Figure 7(c)-(l) show the comparison of searching overhead of IIL-PAEKS scheme and our PAUKS scheme. To ensure the impartiality, the keywords in each encryption and query are randomly selected from the experimental keyword space; The sender in each encryption is also randomly chosen. Subfigure (c)(d), (e)(f), (g)(h), (i)(j) and (k)(l) show the test time varies with the total number of received ciphertexts, when there are 1, 25, 50, 75 and 100 senders, respectively. Each ciphertext is labeled with the sender's public key, hence, the running time of the **Test** algorithms in the two schemes are linear with the total received ciphertexts shown in subfigure (d)(f)(h)(j)(l) but not entirely linear fit shown in subfigure (c)(e)(g)(i)(k). This is because the randomness of the senders and plaintexts leads to noteworthy random errors in a small sample space but no influence in a larger sample space.

Observing subfigure (c)-(l), with the same number of ciphertexts, there is no significant difference in the running time of **Test** algorithms. This is because all ciphertexts have been classified based on the label of senders and the common operations caused by different number of senders are much more efficient than the **Test** algorithms, which have minimal effect on the final experimental results.

Illustrated in Figure 7(c)-(l), compared with the "linear-cost" **Test** algorithms, the fast search of our PAUKS scheme enjoys an almost constant running overhead as the number of ciphertexts increases. The experimental results show that the "constant" fast search of our PAUKS scheme has a huge advantage in search efficiency when the number of stored ciphertexts is greater than some threshold (50, approximately).

Remark: The source code of the experiment has been uploaded to the public website, which can be found at https://github.com/PAEKSMAKER/Crypto_PAUK.git.

VI. Conclusion and Future Work

In this paper, we proposed an EMR system by introducing a public key authenticated encryption with ciphertext update and keyword search (PAUKS) scheme. Different from preceding PAEKS schemes, the proposed PAUKS scheme not only resists KGAs from both outside and inside adversaries, but also reduces trapdoor communication overhead from linear to sub-linear with the help of a non-collusion proxy. As a side result, the proposed PAUKS scheme is compatible with a secure inverted index, which achieves a fast searching efficiency.

In the system model of our PAUKS, the proxy should not collude with senders nor the cloud server. It would be an interesting work to propose a PAUKS scheme to remove the proxy from the system model and delegate the cloud computing server as the proxy. Another future work is to design a long-term secure PAUKS scheme. For many scenarios, it is important to guarantee the long-term security and support key update or forward security.

With the development of quantum computing technique, pairing based schemes might be vulnerable under the quantum computing in the future. It is important to design schemes based on quantum-resistant hard problems, e.g. LWE and SIS problems. In addition, quantum-secure lattice-based schemes might be faster than pairing in certain cases, even though larger space might be needed for trapdoors and keys. Our PAUKS scheme proves the feasibility of achieving both KGA security and lightweight in application. In the future we consider to design lattice-based PAEKS schemes supporting constant trapdoor generation and fast retrieval efficiency.

Appendix I
Security Analysis of the PAUKS Scheme

Lemma 1 (IND-CCA): For any PPT adversary $\mathcal{A}$, the advantage $\mathbf{Adv}^{C}_{\mathcal{A}}(1^{\lambda})$ to break the IND-CCA security of our PAUKS scheme is negligible if the CDH and mDLIN assumptions hold.

Proof: Let $\mathcal{A}$ be a PPT adversary to break the trapdoor privacy of the proposed PAUKS scheme. There is an algorithm $\mathcal{B}$ to solve the CDH problem based on the following game played with $\mathcal{A}$. The advantage of winning the following game for $\mathcal{A}$ is negligible if the CDH and mDLIN assumptions hold.

Game 1: The algorithm $\mathcal{B}$ takes as input a CDH instance, e.g. (g, g^{x}, g^{v}), and then runs $\mathcal{A}$ as a subroutine and plays the following game with $\mathcal{A}$.

- **Setup**: Given the security parameter 1^{λ}, the algorithm $\mathcal{B}$ sets the public parameter parameter $\mathbb{PP} = (\mathbb{G}, \mathbb{G}_T, \hat{e}, p, g)$ and sets a pair of users (U^*_S, U^*_R) as the target sender and target receiver. Let $pk^*_S = (g^{v})$ be the public key of U^*_S and $pk^*_R = (g^{x}, g^{x_2}, g^{x_3})$ be the public key of U^*_R, where x_2, x_3 are randomly selected from $\mathbb{Z}_p$.
- **Phase 1**: Given the public key parameter $\mathbb{PP}$, the adversary $\mathcal{A}$ is able to query the following oracles. The oracles will log all the inputs and outputs, and preferentially returns the output value if the input appears in the log. If the input is a new record, then the oracles return as follows.
  - **Hash Oracles**:
    - $\mathcal{O}_{H_1}$: Given an input (T, w), this oracle randomly selects $h_1 \leftarrow \mathbb{Z}_p$ and returns $H_1(T, w) = h_1$;
    - $\mathcal{O}_{H_2}$: Given an input (T), this oracle randomly selects $h_2 \leftarrow \mathbb{Z}_p$ and returns $H_2(T) = h_2$;
    - $\mathcal{O}_{H_3}$: Given an input (T), this oracle randomly selects $h_3 \leftarrow \mathbb{Z}_p$ and returns $H_3(T) = h_3$.
  - **Ciphertext Oracle** $\mathcal{O}_C$: Given a keyword w, this oracle returns ciphertext $C = C_1, C_2, C_3, C_4, C_5$ as follows:

$$C_1 = \left((g^{x_2})^{H_1(T^*, w)} \cdot g^{x_3}\right)^{r_1}, \quad C_2 = g^{r_1}$$
$$C_3 = \left((g^{x_2})^{H_2(T^*)} \cdot g^{v_2}\right)^{r_2} \cdot C_2^{H_2(T^*)},$$
$$C_4 = H_4(w)^{r_2}, C_5 = H(C_1, C_2, C_3, C_4)^{r_1},$$

    where r_1 and r_2 are randomly selected from $\mathbb{Z}_p$, and $H_1(T^*, w)$, $H_2(T^*)$, $H_3(T^*)$ are generated randomly by the aforementioned hash oracles. Note that even though the input $T^* = g^{xv}$ is unknown by $\mathcal{B}$, the hash values are known, which are randomly selected.

– **Trapdoor Oracle** $\mathcal{O}_T$: Given a keyword w, this oracle returns trapdoor $T_w = T_{w,1}, T_{w,2}$ as follows:

$$T_{w,1} = g^{\frac{r_3}{x_2 \cdot H_1(T^w, w') \cdot x_3}}, \quad T_{w,2} = g^{r_3},$$

where $H_1(T^w, w')$ is randomly generated by oracle $\mathcal{O}_H$ and r_2 is randomly selected from $\mathbb{Z}_p$.

- **Challenge**: $\mathcal{A}$ submits two keywords w_0^*, w_1^* which have not been queried to oracle $\mathcal{O}_C$. The algorithm $\mathcal{B}$ tosses a coin to decide a bit $b \leftarrow \{0, 1\}$, and returns the following ciphertext $C^* = C_1^*, C_2^*, C_3^*, C_4^*, C_5^*$:

$$C_1^* = \left(g^{x_2 \cdot H_1^* + x_3}\right)^{r_1^*}, \quad C_2^* = g^{r_1^*},$$
$$C_3^* = \left(g^{x_2 \cdot H_2(T^*) + x_3}\right)^{r_2^*} \cdot C_2^{*H_3(T^w)}, \quad C_4^* = H_4(w)^{r_2^*},$$
$$C_5^* = H(C_1, C_2, C_3, C_4)^{r_1^*},$$

where r_1^*, r_2^*, H_1^* are random selected from $\mathbb{Z}_p$, and $H_2(T^w)$ is generated from the hash oracle $\mathcal{O}_{H_2}$.

- **Phase 2**: $\mathcal{A}$ can also query the oracles in phase 1, except that w_0^*, w_1^* cannot appear in $\mathcal{O}_T$.
- **Guess**: Finally, $\mathcal{A}$ returns a bit $b' \leftarrow \{0, 1\}$ and wins the game iff $b' = b$.

We denote by $\mathbf{E}_1$ the event that $\mathcal{A}$ has queried (g^{xy}, w_b^*) to the hash oracle $\mathcal{O}_{H_1}$. In case $\mathbf{E}_1$ happens, the algorithm $\mathcal{B}$ aborts the game and solves the CDH problem based on $\mathcal{A}$'s input. In other case, the game is identical to the game in the security model in $\mathcal{A}$'s view and the ciphertext is indistinguishable if the mDLIN assumption holds of which the proof is omitted here limited by space. Hence, the probability of $\mathcal{A}$ to win this game is:

$$\begin{aligned}
\mathbf{Pr}[\mathcal{A}_{\text{win}}^{\mathbf{G}_1}] &= \left|\mathbf{Pr}[\mathcal{A}_{\text{win}}^{\mathbf{G}_1} \wedge \mathbf{E}_1] + \mathbf{Pr}[\mathcal{A}_{\text{win}}^{\mathbf{G}_1} \wedge (\mathbf{E}_1)]\right| \\
&= \left|\mathbf{Pr}[\mathcal{A}_{\text{win}}^{\mathbf{G}_1} | \overline{\mathbf{E}_1}] \cdot \mathbf{Pr}[\overline{\mathbf{E}_1}] + \mathbf{Pr}[\mathcal{A}_{\text{win}}^{\mathbf{G}_1} | \mathbf{E}_1] \cdot \mathbf{Pr}[\mathbf{E}_1]\right| \\
&= \frac{1}{2} \cdot (1 - negl(1^\lambda)) + \mathbf{Pr}[\mathcal{A}_{\text{win}}^{\mathbf{G}_1} | \overline{\mathbf{E}_1}] \cdot negl(1^\lambda) \\
&\leq \frac{1}{2} + negl(1^\lambda).
\end{aligned}$$

■

Lemma 2 (IND-U-CCA): The proposed PAUKS scheme IND-U-CCA secure in the random oracle if the CDH and mDLIN assumptions hold.

Proof: The proof is based on the following game played between a PPT adversary $\mathcal{A}$ and an algorithm $\mathcal{B}$.

Game 2: The algorithm $\mathcal{B}$ is given the CDH tuple (g, g^x, g^y) and to solve the CDH problem if $\mathcal{A}$ is able to break the security of the proposed PAUKS scheme within a PPT time.

- **Setup**: Given the security parameter 1^λ, the algorithm $\mathcal{B}$ sets the public parameter parameter $\mathbb{PP} = \{\mathbb{G}, \mathbb{G}_T, \hat{e}, p, g\}$ and sets a pair of users (U_S^*, U_R^*) as target receiver. Let $pk_S^* = (g^x)$ be the public key of U_S^* and $pk_R^* = (g^{x_2}, g^{x_3}, g^{x_4})$ be the public key of U_R^*, where the private key x_2, x_3, x_4 are randomly selected from $\mathbb{Z}_p$.
- **Phase 1**: In this phase, the adversary $\mathcal{A}$ can query the following oracles.
  – Hash Oracles:
    * $\mathcal{O}_{H_1}$: Given an input (T, w), this oracle randomly selects $h \xleftarrow{\$} \mathbb{Z}_p$ and returns $H_1(T, w) = h_1$.
    * $\mathcal{O}_{H_2}$: Given an input (T), this oracle randomly selects $h_1 \xleftarrow{\$} \mathbb{Z}_p$ and returns $H_2(T) = h_2$.
    * $\mathcal{O}_{H_3}$: Given an input (T), this oracle randomly selects $h_2 \xleftarrow{\$} \mathbb{Z}_p$ and returns $H_3(T) = h_3$.
  – Encryption Oracle $\mathcal{O}_C$: Given a keyword w, this oracle randomly selects $r_1, r_2 \xleftarrow{\$} \mathbb{Z}_p$ and returns a ciphertext $C = C_1, C_2, C_3, C_4, C_5$ as follow.

$$C_1 = \left(pk_{\mathbf{R}_2}^{H_1(T^w, w)} \cdot pk_{\mathbf{R}_3}\right)^{r_1}, \quad C_2 = g^{r_1},$$
$$C_3 = \left(pk_{\mathbf{R}_2}^{H_2(T^w)} \cdot pk_{\mathbf{R}_3}\right)^{r_2} \cdot C_2^{H_3(T^w)},$$
$$C_4 = H_4(w)^{r_2}, C_5 = H(C_1, C_2, C_3, C_4)^{r_1}.$$

  – Trapdoor Oracle $\mathcal{O}_{\hat{T}}$: Given a keyword w, this oracle randomly selects $r_3 \xleftarrow{\$} \mathbb{Z}_p$ and returns a trapdoor $\hat{T}_w = (\hat{T}_{w,1}, \hat{T}_{w,2})$ as follow.

$$\hat{T}_{w,1} = g^{x_4 \cdot r_3}, \hat{T}_{w,2} = H_2(w)^{r_3}.$$

- **Challenge**: $\mathcal{A}$ submits two keywords w_0^*, w_1^* which have not been queried to oracle $\mathcal{O}_{\hat{T}}$. The algorithm $\mathcal{B}$ randomly selects $b \xleftarrow{\$} \{0, 1\}$ and $r_1^*, r_2^* \xleftarrow{\$} \mathbb{Z}_p$, and returns the challenged ciphertext $C^* = C_1^*, C_2^*, C_3^*, C_4^*, C_5^*, C_6^*$ as follow:

$$C_1^* = \left(pk_{\mathbf{R}_2}^{H_1^*} \cdot pk_{\mathbf{R}_3}\right)^{r_1^*}, \quad C_2^* = g^{r_1^*},$$
$$C_3^* = \left(pk_{\mathbf{R}_2}^{H_2(T^*)} \cdot pk_{\mathbf{R}_3}\right)^{r_2^*} \cdot C_2^{*H_3(T^*)}, C_4^* = H_1(w^*)^{r_2^*},$$
$$C_5^* = H(C_1, C_2, C_3, C_4)^{r_1^*}, \quad C_6^* = g^{x_4 \cdot r_2^*}$$

where r_1^*, r_2^*, H_1^* are random selected from $\mathbb{Z}_p$, and $H_2(T^*)$ is generated from the hash oracle $\mathcal{O}_{H_2}$.

- **Phase 2**: The adversary $\mathcal{A}$ can query the oracles in phase 1, except that the keywords w_0^*, w_1^* cannot appear in the trapdoor oracle $\mathcal{O}_{\hat{T}}$.
- **Guess**: $\mathcal{A}$ returns a keyword w' and wins the game iff $w' = w^*$.

We denote the event that the adversary $\mathcal{A}$ queries g^{xy} to the hash oracle $\mathcal{O}_{H_1}$, $\mathcal{O}_{H_2}$ or $\mathcal{O}_{H_3}$ by $\mathbf{E}_2$. In case $\mathbf{E}_2$ happens, $\mathcal{B}$ solves the CDH problem by retrieving the log of the hash oracles. In other case, the game is identical to the game in the security model in $\mathcal{A}$'s view and the ciphertext is indistinguishable if the mDLIN assumption holds of which the proof is similar to that in game 1 and omitted here due to the space limitation. Hence, The probability that $\mathcal{A}$ breaks the ciphertext semantic security of the proposed PAUKS scheme is:

$$\begin{aligned}
\left|\mathbf{Pr}[\mathcal{A}_{\text{win}}^{\mathbf{G}_2}]\right| &= \left|\mathbf{Pr}[\mathcal{A}_{\text{win}}^{\mathbf{G}_2} \wedge \overline{\mathbf{E}_2}] + \mathbf{Pr}[\mathcal{A}_{\text{win}}^{\mathbf{G}_2} \wedge \mathbf{E}_2]\right| \\
&= \left|\mathbf{Pr}[\mathcal{A}_{\text{win}}^{\mathbf{G}_2} | \overline{\mathbf{E}_2}] \cdot \mathbf{Pr}[\overline{\mathbf{E}_2}] + \mathbf{Pr}[\mathcal{A}_{\text{win}}^{\mathbf{G}_2} | \mathbf{E}_2] \cdot \mathbf{Pr}[\mathbf{E}_2]\right| \\
&= \frac{1}{2} \cdot (1 - negl(1^\lambda)) + \mathbf{Pr}[\mathcal{A}_{\text{win}}^{\mathbf{G}_2} | \mathbf{E}_2] \cdot negl(1^\lambda) \\
&\leq \frac{1}{2} + negl(1^\lambda).
\end{aligned}$$

■

Lemma 3 (IND-TP-CCA): The proposed PAEK scheme satisfies the IND-TP-CCA security in the random oracle if the CDH assumption holds.

The proof of lemma 3 is similar to that of lemma 1 and 2, which is omitted here.

REFERENCES

[1] D. Boneh, G. Di Crescenzo, R. Ostrovsky, and G. Persiano, "Public key encryption with keyword search," in *Advances in Cryptology—EUROCRYPT*, C. Cachin and J. L. Camenisch, Eds. Berlin, Germany: Springer, 2004, pp. 506–522.

[2] J. W. Byun, H. S. Rhee, H. A. Park, and D. H. Lee, "Off-line keyword guessing attacks on recent keyword search schemes over encrypted data," in *Secure Data Management*, 3rd ed. Seoul, (South) Korea: SDM, Sep. 2006.

[3] R. Chen, Y. Mu, G. Yang, F. Guo, and X. Wang, "Dual-server public-key encryption with keyword search for secure cloud storage," *IEEE Trans. Inf. Forensics Security*, vol. 11, no. 4, pp. 789–798, Apr. 2016.

[4] R. Chen et al., "Server-aided public key encryption with keyword search," *IEEE Trans. Inf. Forensics Security*, vol. 11, no. 12, pp. 2833–2842, Dec. 2016.

[5] Y. Ling, S. Ma, Q. Huang, and X. Li, "A general two-server framework for ciphertext-checkable encryption against offline message recovery attack," in *Cloud Computing and Security* (Lecture Notes in Computer Science), X. Sun, Z. Pan, and E. Bertino, Eds. Cham, Switzerland: Springer, 2018, pp. 370–382.

[6] L. Fang, W. Susilo, C. Ge, and J. Wang, "Public key encryption with keyword search secure against keyword guessing attacks without random Oracle," *Inf. Sci.*, vol. 238, pp. 221–241, Jul. 2013.

[7] Q. Huang and H. Li, "An efficient public-key searchable encryption scheme secure against inside keyword guessing attacks," *Inf. Sci.*, vols. 403–404, pp. 1–14, Sep. 2017.

[8] D. He, M. Ma, S. Zeadally, N. Kumar, and K. Liang, "Certificateless public key authenticated encryption with keyword search for industrial Internet of Things," *IEEE Trans. Ind. Informat.*, vol. 14, no. 8, pp. 3618–3627, Aug. 2018.

[9] L. Wu, Y. Zhang, M. Ma, N. Kumar, and D. He, "Certificateless searchable public key authenticated encryption with designated tester for cloud-assisted medical Internet of Things," *Ann. Telecommun.*, vol. 74, nos. 7–8, pp. 423–434, Aug. 2019.

[10] H. Li, Q. Huang, J. Shen, G. Yang, and W. Susilo, "Designated-server identity-based authenticated encryption with keyword search for encrypted emails," *Inf. Sci.*, vol. 481, pp. 330–343, May 2019.

[11] B. Qin, Y. Chen, Q. Huang, X. Liu, and D. Zheng, "Public-key authenticated encryption with keyword search revisited: Security model and constructions," *Inf. Sci.*, vol. 516, pp. 515–528, Apr. 2020.

[12] N. Paknia, D. Shiraly, and Z. Eslami, "Certificateless authenticated encryption with keyword search: Enhanced security model and a concrete construction for industrial IoT," *J. Inf. Secur. Appl.*, vol. 53, Aug. 2020, Art. no. 102525.

[13] M. Noroozi and Z. Eslami, "Public-key encryption with keyword search: A generic construction secure against online and offline keyword guessing attacks," *J. Ambient Intell. Humanized Comput.*, vol. 11, no. 2, pp. 879–890, Feb. 2020.

[14] B. Wu, C. Wang, and H. Yao, "Security analysis and secure channel-free certificateless searchable public key authenticated encryption for a cloud-based Internet of Things," *PLoS ONE*, vol. 15, no. 4, Apr. 2020, Art. no. e0230722.

[15] J. Li, M. Wang, Y. Lu, Y. Zhang, and H. Wang, "ABKS-SKGA: Attribute-based keyword search secure against keyword guessing attack," *Comput. Standards Interfaces*, vol. 74, Feb. 2021, Art. no. 103471.

[16] N. Yang, Q. Zhou, and S. Xu, "Public-key authenticated encryption with keyword search without pairings," *J. Comput. Res. Develop.*, vol. 57, no. 10, p. 2125, 2020.

[17] X. Liu, K. He, G. Yang, W. Susilo, J. Tonien, and Q. Huang, "Broadcast authenticated encryption with keyword search," in *Proc. Australas. Conf. Inf. Secur. Privacy*. Cham, Switzerland: Springer, 2021, pp. 193–213.

[18] Y. Lu and J. Li, "Lightweight public key authenticated encryption with keyword search against adaptively-chosen-targets adversaries for mobile devices," *IEEE Trans. Mobile Comput.*, vol. 21, no. 12, pp. 4397–4409, Dec. 2022.

[19] B. Qin, H. Cui, X. Zheng, and D. Zheng, "Improved security model for public-key authenticated encryption with keyword search," in *Proc. Int. Conf. Provable Practical Secur.* Cham, Switzerland: Springer, 2021, pp. 19–38.

[20] L. Han, J. Guo, G. Yang, Q. Xie, and C. Tian, "An efficient and secure public key authenticated encryption with keyword search in the logarithmic time," *IEEE Access*, vol. 9, pp. 151245–151253, 2021.

[21] J. Baek, R. Safavi-Naini, and W. Susilo, "Public key encryption with keyword search revisited," in *Proc. Int. Conf. Comput. Sci. Appl.*, Perugia, Italy, 2008, pp. 1249–1259.

[22] P. Xu, H. Jin, Q. Wu, and W. Wang, "Public-key encryption with fuzzy keyword search: A provably secure scheme under keyword guessing attack," *IEEE Trans. Comput.*, vol. 62, no. 11, pp. 2266–2277, Nov. 2013.

[23] M. Noroozi and Z. Eslami, "Public key authenticated encryption with keyword search: Revisited," *IET Inf. Secur.*, vol. 13, no. 4, pp. 336–342, Jul. 2019.

[24] X. Pan and F. Li, "Public-key authenticated encryption with keyword search achieving both multi-ciphertext and multi-trapdoor indistinguishability," *J. Syst. Archit.*, vol. 115, May 2021, Art. no. 102075.

[25] B. Chen, L. Wu, S. Zeadally, and D. He, "Dual-server public-key authenticated encryption with keyword search," *IEEE Trans. Cloud Comput.*, vol. 10, no. 1, pp. 322–333, Jan. 2022.

[26] Y. Lu, J. Li, and Y. Zhang, "Secure channel free certificate-based searchable encryption withstanding outside and inside keyword guessing attacks," *IEEE Trans. Services Comput.*, vol. 14, no. 6, pp. 2041–2054, Nov. 2021.

[27] J. Guo, L. Han, G. Yang, X. Liu, and C. Tian, "An improved secure designated server public key searchable encryption scheme with multi-ciphertext indistinguishability," *J. Cloud Comput.*, vol. 11, no. 1, pp. 1–12, Dec. 2022.

[28] Z.-Y. Liu, Y.-F. Tseng, R. Tso, Y.-C. Chen, and M. Mambo, "Identity-certifying authority-aided identity-based searchable encryption framework in cloud systems," *IEEE Syst. J.*, vol. 16, no. 3, pp. 4629–4640, Sep. 2022.

[29] X. Liu, H. Li, G. Yang, W. Susilo, J. Tonien, and Q. Huang, "Towards enhanced security for certificateless public-key authenticated encryption with keyword search," in *Provable Security* (Lecture Notes in Computer Science), R. Steinfeld and T. H. Yuen, Eds. Cham, Switzerland: Springer, 2019, pp. 113–129.

[30] Y. Lu, J. Li, and F. Wang, "Pairing-free certificate-based searchable encryption supporting privacy preserving keyword search function for IIoTs," *IEEE Trans. Ind. Informat.*, vol. 17, no. 4, pp. 2696–2706, Apr. 2021.

[31] R. Behnia, M. O. Ozmen, and A. A. Yavuz, "Lattice-based public key searchable encryption from experimental perspectives," *IEEE Trans. Dependable Secure Comput.*, vol. 17, no. 6, pp. 1269–1282, Nov. 2020.

[32] Z.-Y. Liu, Y.-F. Tseng, R. Tso, M. Mambo, and Y.-C. Chen, "Public-key authenticated encryption with keyword search: Cryptanalysis, enhanced security, and quantum resistant instantiation," in *Proc. ACM Asia Conf. Comput. Commun. Secur.*, May 2022, pp. 423–436.

[33] M. Blaze, G. Bleumer, and M. Strauss, "Divertible protocols and atomic proxy cryptography," in *Proc. Int. Conf. Theory Appl. Cryptograph. Techn.* Cham, Switzerland: Springer, 1998, pp. 127–144.

[34] J. Shao, Z. Cao, X. Liang, and H. Lin, "Proxy re-encryption with keyword search," *Inf. Sci.*, vol. 180, no. 13, pp. 2576–2587, Jul. 2010.

[35] Z. Chen, S. Li, Y. Guo, Y. Wang, and Y. Chu, "A limited proxy re-encryption with keyword search for data access control in cloud computing," in *Proc. Int. Conf. Netw. Syst. Secur.* Cham, Switzerland: Springer, 2015, pp. 82–95.

[36] T. Hoang, R. Behnia, Y. Jang, and A. A. Yavuz, "MOSE: Practical multi-user oblivious storage via secure enclaves," in *Proc. 10th ACM Conf. Data Appl. Secur. Privacy*, Mar. 2020, pp. 17–28.

[37] T. Hoang, M. O. Ozmen, Y. Jang, and A. A. Yavuz, "Hardware-supported ORAM in effect: Practical oblivious search and update on very large dataset," *Proc. Privacy Enhancing Technol.*, vol. 2019, no. 1, pp. 172–191, Jan. 2019.

[38] L. Xu, Z. Sun, W. Li, and H. Yan, "Delegatable searchable encryption with specified keywords for EHR systems," *Wireless Netw.*, pp. 1–13, Jul. 2020, doi: 10.1007/s11276-020-02410-3.

[39] M. Deng, W. Song, M. Ma, H. Li, and M. Israr, "Efficient designated-server proxy re-encryption with keyword search for big data," in *Proc. Int. Conf. Artif. Intell. Secur.* Cham, Switzerland: Springer, 2022, pp. 364–377.
[40] X. Boyen, "The uber-assumption family," in *Proc. Int. Conf. Pairing-Based Cryptogr.* Cham, Switzerland: Springer, 2008, pp. 39–56.
[41] F. Bao, R. H. Deng, and H. Zhu, "Variations of Diffie-Hellman problem," in *Proc. Int. Conf. Inf. Commun. Secur.* Cham, Switzerland: Springer, 2003, pp. 301–312.

Hongbo Li received the Ph.D. degree from South China Agricultural University, Guangzhou, China. He currently holds a post-doctoral position with the College of Mathematics and Informatics, South China Agricultural University. His research interests include applied cryptography and cloud security.

Qiong Huang received the Ph.D. degree from the City University of Hong Kong in 2010. He is currently a Professor with the College of Mathematics and Informatics, South China Agricultural University, Guangzhou, China. He has published more than 150 research papers in international conferences and journals. His research interests include cryptography and information security, in particular, cryptographic protocols design and analysis. He served as a program committee member in many international conferences.

Jianye Huang received the B.S. and M.S. degrees from South China Agricultural University, Guangzhou, China, in 2015 and 2018, respectively. He is currently pursuing the Ph.D. degree with the University of Wollongong, Australia. His research interests include cryptography: in particular, leakage-resilient signature, searchable encryption, and k-times anonymous authentication.

Willy Susilo (Fellow, IEEE) is currently a Distinguished Professor with the Faculty of Engineering and Information Sciences, School of Computing and Information Technology, University of Wollongong (UOW), Australia, where he is also the Director of the School of Computing and Information Technology, Institute of Cybersecurity and Cryptology. He has been the Head of the School of Computing and Information Technology, UOW, since 2015. He has also served as the program committee member of several international conferences. He was awarded the prestigious Australian Research Council Future Fellowship in 2009. In 2016, he was awarded the Researcher of the Year at UOW, due to his research excellence and contributions. He is the Editor-in-Chief of the *Computer Standards and Interfaces* (Elsevier) and the *Information* journal (MDPI). He is currently an Associate Editor of IEEE TRANSACTIONS ON DEPENDABLE AND SECURE COMPUTING, *ACM Computing Surveys*, and *Computers and Security* (Elsevier).

2.2 代表作：A Secure Cloud Data Sharing Protocol for Enterprise Supporting Hierarchical Keyword Search

发表于 IEEE Transactions on Dependable and Secure Computing (IF5-year-7.2)
中科院 2 区 CCF-A Top 期刊
论文等级：A 类

A Secure Cloud Data Sharing Protocol for Enterprise Supporting Hierarchical Keyword Search

Hongbo Li, Qiong Huang, and Willy Susilo, *Senior Member, IEEE*

Abstract—Cloud storage becomes the priority for storing and sharing data for enterprise users. Encrypting prior to uploading data to the cloud is the best way to protect business secrets, however, it hinders the convenient operations on plaintexts, such as searching over the cloud data. In addition, employees in an enterprise have multiple layer structures and a higher layer employee should have the privilege to monitor the lower layer employees' data to check if these users violate the regulation without letting the employees be aware of. Public key encryption with keyword search (PEKS) is a well-known cryptographic primitive suitable for secure cloud storage, which supports keyword search without decryption in public key encryption settings. Unfortunately, no existing PEKS scheme supports the monitoring function without authorization from the sender. To address this issue, we propose a variant of PEKS named Hierarchical Public Key Encryption with Keyword Search (HPEKS) and provide a semi-generic construction utilizing a public key tree (PKTree) and a PEKS scheme. To better suit for the enterprise secret data sharing, we build an advanced HPEKS scheme, named designated-tester decryptable hierarchical public key encryption with keyword search (dDHPEKS), which enjoys stronger security and integrates the public key and symmetric key encryptions. We prove our dDHPEKS scheme secure under the security definition in the random oracle model. Particularly, it satisfies the security against outside offline keyword guessing attacks and furthermore, enjoys the *transparency* property so that the sender does not need to know the internal hierarchy structure of an enterprise in order to share encrypted data to the enterprise. Theoretical evaluation and concrete experiments show that our dDHPEKS scheme has comparable running efficiency with existing PEKS schemes.

Index Terms—Public key encryption, keyword search, keyword guessing attacks, secret data sharing, cloud storage

1 Introduction

With the rapid development of computing and communication technology, data generated by enterprises or firms expand exponentially with very high speed. Due to the low cost of data management, the public cloud service (PCS) becomes the top priority for most enterprises. Statistics show that most enterprises used at least one PCS [1]. It allows users to access the mass of data everywhere using storage limited mobile devices via the internet. One of the services supplied by the PCS is to store and manage messages and documents sent to users. However, the risk of a security breach also limits some enterprises to use the PCS [1]. According to the statistics, security concern is the most important issue for respondents [1]. Encrypting prior to uploading is a direct measure to counter-attack this risk, but it limits the application on the data.

1.1 Motivation - The Need for Hierarchical Structure

The Office Automation (OA) system is one of the applications of the cloud computing. Thousands of messages are generated everyday from an enterprise or the government. In the real world, the OA services are often outsourced to the cloud service provider, because it owns a huge advantage in costs of data storage and management. Messages in the OA system are often stored as plaintexts, which make it possible for adversaries to access and accordingly would lead to economic loss or unfairness for the public.

For instance, a project undertaken by an enterprise would be obstructed by its business competitor if the related information leaks in advance. In another example, the leakage of some government-scheduled policy would give someone a head start, which is unfair to the rest of the public.

To avoid this situation, there should be a reasonable access control strategy, which allows authorized users to access the corresponding messages and denies the access from unauthorized user or the less privileged users. For example, in an enterprise, the chief executive officer (CEO) supervises various other employees, such as chief operating officer (COO), chief financial officer (CFO), chief information officer (CIO) and chief technology officer (CTO). The COO, CFO, CIO and CTO supervise several employees, respectively. To allow the CEO's access to the message sent from COO to CIO helps the

- *Hongbo Li is with the College of Mathematics and Informatics, South China Agricultural University, Guangzhou 510642, China. E-mail: hongbo@scau.edu.cn.*
- *Qiong Huang is with the College of Mathematics and Informatics, South China Agricultural University, Guangzhou 510642, China, and also with the Guangzhou Key Laboratory of Intelligent Agriculture, Guangzhou, China. E-mail: qhuang@scau.edu.cn.*
- *Willy Susilo is with the Institute of Cybersecurity and Cryptology, School of Computing and Information Technology, University of Wollongong, Wollongong, NSW 2522, Australia. E-mail: wsusilo@uow.edu.au.*

Manuscript received 5 July 2019; revised 26 Aug. 2020; accepted 25 Sept. 2020. Date of publication 29 Sept. 2020; date of current version 13 May 2022. (Corresponding author: Qiong Huang and Willy Susilo.) Digital Object Identifier no. 10.1109/TDSC.2020.3027611

CEO effectively manage the enterprise. To deny the CFO's access to the message sent from CTO to CEO can reduce the risk of information leakage.

Even though the OA system has an access control strategy,[1] once the system is broken in, all the information leaks. Hence, the ciphertext-level access control strategy is a reasonable and important way to protect data privacy. The basic idea of the ciphertext-level access control strategy is to encrypt messages with different users' public key, in which way even if the OA system is broken in, only ciphertexts are exposed, which leaks less information. For this scenario, the encryption scheme shell meet the following requirements.

1) Searchability. It is inefficient to download and decrypt all the ciphertexts to find some specific documents, such as documents corresponding to the keyword "contract". To efficiently find these documents without leaking information about the content of the document, a method of searching over ciphertexts without decryption is needed.
2) Access control based on user priority. A message or a task in the real world often needs to be sent to and handled by different users. The encrypted message should only be decrypted by the user who has the corresponding or higher access-priority. Considering the structure of the receivers, it is reasonable to apply a tree-like access control strategy to decrypt the ciphertexts.
3) Transparency. It is not necessary for a sender Alice to know the internal structure of the organization the receiver is in. When Alice sends an encrypted message to a receiver Bob, she only needs to know Bob's valid public key. In other words, Alice does not need to know Bob's priority in the enterprise neither who in the enterprise has a higher priority than Bob.

To the best of our knowledge, there is no corresponding cryptographic primitive meeting the above requirements simultaneously. The public key encryption with keyword search (PEKS) supports searching some keyword over ciphertexts without decryption, but it lacks the mechanism of access control. Attribute-based encryption (ABE) and Hierarchical identity-based encryption (HIBE) support ciphertext-level access control. It seems feasible to combine ABE or HIBE with PEKS to construct a suitable searchable encryption scheme, however, in ABE and HIBE, the sender needs to know the receiver's internal organization structure.

Therefore, it is essential to construct a new cryptographic primitive suitable for the aforementioned cloud data sharing scenario.

1.2 Contributions

1.2.1 Public Key Tree

To resolve the problem above, we introduce the public key tree (PKTree) into the searchable encryption and propose a new notion named *hierarchical public key encryption with keyword search* (HPEKS). In HPEKS, it allows users to search over ciphertexts encrypted with their public keys. Specially, if there is a group of users with a hierarchical structure, the user with higher access permission can search over ciphertexts sent to users with lower access permission. For example, if Alice supervises Bob and Carlos, Alice can perform searching operations over ciphertexts encrypted with Bob's and Carlos' public keys.

1.2.2 Semi-Generic HPEKS Construction From PEKS

To demonstrate the feasibility, we give a semi-generic construction HPEKS leveraging the proposed PKTree technique together with an existing bilinear-pairing based PEKS scheme in the literature.

1.2.3 Advanced HPEKS Scheme dDHPEKS

To resist outside offline keyword guessing attacks, we propose an advanced HPEKS scheme, named *designated-tester decryptable hierarchical public key encryption with keyword search* (dDHPEKS), in which only the designated server can search for users. Moreover, our proposal integrates PEKS and PKE, which means it supports not only keyword search but also decryption. Our dDHPEKS scheme enjoys an interesting property,*transparency*, meaning that the sender does not need to know the internal hierarchy structure of the organization that the receiver is in before encrypting a keyword for the receiver. Instead, the sender needs to encrypt the keyword under the public key of the intended receiver only. In contrast, HIBE or ABE integrated with PEKS does not support this property.

1.2.4 Security and Efficiency

We give formal security definition for dDHPEKS, including secret key security, keyword privacy and plaintext privacy, and prove our dDHPEKS scheme secure under the given security definitions. We analyze the running overhead of dDHPEKS theoretically and implement it utilizing C language and PBC library [2]. The analysis and experiment results show that our dDHPEKS scheme has comparable running overhead with existing PEKS schemes.

1.3 Related Works

The notion of searchable symmetric encryption (SSE) and the first searchable encryption scheme was proposed by Song *et al.* [3] in 2000. It allows a user who has the searching key to search some keyword over ciphertexts without decryption. However, Song *et al.*'s scheme and other schemes [4], [5], [6] based on it have a common restriction that it is hard to share the searching key with others, thus are only suitable for data owner itself to search over the ciphertexts. Boneh *et al.* [7] introduced SE into the public key settings and proposed the first public key SE scheme named Public Key Encryption with Keyword Search, which breaks the barrier of data sharing limit. In PEKS, a data owner (encrypter) performs the encryption algorithm using a receiver's (searcher's) public key, and in reverse, the receiver generates a trapdoor to search some keyword over ciphertexts using its secret key.

Other issues in PEKS have also engaged researchers' attention. Park *et al.* and Golle *et al.* proposed schemes named public key encryption with conjunctive keyword search (PECKS) [8], [9] which support receivers to search

1. The system-level access control strategy is out of our research scope and is thus omitted in this paper.

for documents containing all of several keywords in only a single query. To enable a ranked list of the searching result, a new notion called multi-keyword ranked search (MRSE) was proposed [10], [11], [12]. Bao *et al.* [13] and others [14], [15], [16] proposed searchable encryption schemes in multi-receiver settings. To support fine-access control in PEKS, attribute-based encryption with keyword search (ABEKS) was proposed [14], [17]. Byun *et al.* [18] and Yau *et al.* [19] pointed out that Boneh *et al.*'s PEKS scheme and other existing PEKS schemes are vulnerable under the new attack named offline keyword guessing attacks (KGA). They also emphasized that almost all the existing PEKS cannot resist the offline keyword guessing attacks given by inside adversaries (IKGA), e.g., KGA given by searching server. Fang *et al.* [20] proposed a secure-channel free PEKS scheme witch resists the KGA efficiently, however, cannot resist IKGA. Huang and Li proposed a new notion called public key authenticated encryption with keyword search (PAEKS) [21], [22] which resists IKGA by denying adversaries the ability to encrypt keywords. He *et al.* [23] and Li *et al.* [24] introduce PAEKS into Certificateless public key encryption and identity based encryption settings, respectively. Chen *et al.* [25] and Chen *et al.* [26] also solved the IKGA problem by utilizing two server in the system model.

1.4 Paper Organization

In the next section, we briefly introduce some preliminaries and the system model of our HEPKS. We introduce the public key tree structure and its construction in Section 3. We then propose our HPEKS schemes in Section 4, and analyze the security in Section 5. We discuss about the efficiency of our scheme and compare our scheme with some other related schemes in Section 6, and finally conclude the paper in Section 7.

2 Preliminaries

2.1 Bilinear Pairing

Let $\mathbb{G}_1, \mathbb{G}_2$ and $\mathbb{G}_T$ be three groups with same prime order p, g and h be any two generators of $\mathbb{G}_1$ and $\mathbb{G}_2$, respectively. There is a map $\hat{e}: \mathbb{G}_1 \times \mathbb{G}_2 \rightarrow \mathbb{G}_T$ from $\mathbb{G}_1$ and $\mathbb{G}_2$ to $\mathbb{G}_T$. We say $\hat{e}$ is a bilinear pairing map [27] if the following conditions are satisfied:

- Bilinearity: $\forall x, y \in \mathbb{Z}_p$, $\hat{e}(g^x, h^y) = \hat{e}(g, h)^{xy}$.
- Non-degeneracy: $\hat{e}(g, h) \neq 1$.
- Computability: $\hat{e}(g, h)$ is easily computable.

2.2 Computational Diffie-Hellman (CDH) Assumption

Definition 1 (CDH Problem). *Let $\hat{e}: \mathbb{G}_1 \times \mathbb{G}_2 \rightarrow \mathbb{G}_T$ be a bilinear pairing map. Given $g, g^x, g^y \in \mathbb{G}_1$, the CDH problem is to calculate g^{xy}.*

Definition 2 (CDH Assumption [28]). *The CDH assumption is that the probability that any probabilistic polynomial time (PPT) adversary solves the CDH problem is negligible.*

2.3 Decisional Linear (DLIN) Assumption

Let $\hat{e}: \mathbb{G}_1 \times \mathbb{G}_2 \rightarrow \mathbb{G}_T$ be a bilinear pairing. Given $u, u^x \in \mathbb{G}_1, v, v^y, h \in \mathbb{G}_2$, the DLIN assumption is that the advantage of distinguishing h^{x+y} from a random $h^r \in \mathbb{G}_2$ is negligible for any PPT adversary [29]

$$Adv_{\mathcal{A}}^{DLIN} = |\Pr[\mathcal{A}(u, u^x, v, v^y, h, h^{x+y}) = 1] - \Pr[\mathcal{A}(u, u^x, v, v^y, h, h^r) = 1]| \leq negl(\lambda).$$

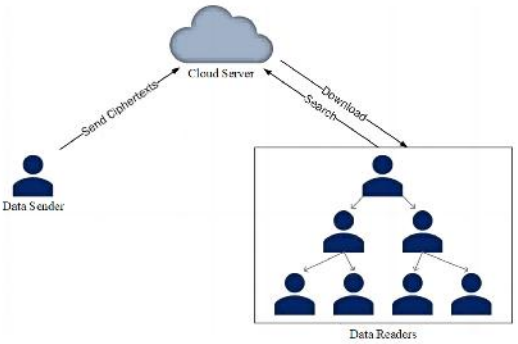


Fig. 1. System model of tree-based hierarchical multi-receiver searchable encryption.

2.4 Decisional Bilinear Diffie-Hellman (DBDH) Assumption

Definition 3 (DBDH Problem). *Let $\hat{e}: \mathbb{G}_1 \times \mathbb{G}_2 \rightarrow \mathbb{G}_T$ be a bilinear pairing map. Given $g, g^x, g^y \in \mathbb{G}_1, h, h^y, h^z \in \mathbb{G}_2$, the DBDH problem is to distinguish $\hat{e}(g, h)^{xyz}$ from a random element $R \in \mathbb{G}_T$, where $x, y, z \in \mathbb{Z}_p$ are unknown.*

Definition 4 (DBDH Assumption). *The DBDH assumption is that the advantage of solving the DBDH problem for any PPT adversary is negligible [28]*

$$Adv_{\mathcal{A}}^{DBDH} = \Pr[\mathcal{A}(g, g^x, g^y, h, h^y, h^z, \hat{e}(g, h)^{xyz}) = 1] - \Pr[\mathcal{A}(g, g^x, g^y, h, h^y, h^z, R) = 1]| \leq negl(\lambda).$$

2.5 System Model

Here we present the system model of the HPEKS system (Fig. 1). There are three entities in the system, denoted by data sender, data receivers and cloud server, respectively.

- *Data sender*: It encrypts data with a receiver's public parameter and sends the ciphertext to the receiver via the cloud server.
- *Data receivers*: The data receivers have a tree-based hierarchical structure. Each data receiver is denoted by a node in the tree. A data receiver can search over ciphertexts sent to itself and to all its children, not vice versa.
- *Cloud server*: It provides the storing and computing services for data receivers. The public parameters of the group of receivers and the received ciphertexts will be stored by the cloud server. It also supports receivers to run algorithms to search over the ciphertexts.

3 The Public Key Tree

To build HPEKS scheme, we introduce a new notion named *public key tree*, which is is a multi-way tree. Each node of a

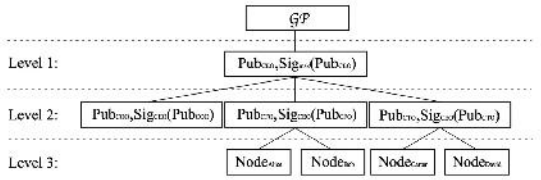


Fig. 2. An example of public key tree.

PKTree contains a user's public information, and we say the user who has the corresponding secret key is the owner of the node. As mentioned above, there is a group of data receivers. There is a supervisor **Rt** of the group, which initially setups the system and holds the master secret key msk. Using msk, **Rt** generates the secret key $\mathsf{sk}_{\mathbf{Rt}}$ of itself and the root node containing **Rt**'s public information. After the initialization, **Rt** should add at least one child node. For instance, **Rt** will add a node named Node_i for receiver R_i. First, **Rt** gathers the public information $\overline{\mathbf{ID}}_i$ of R_i and generates a secret key sk_i and a public parameter Pub_i for R_i using $(\mathbf{ID}_i, \mathsf{sk}_{\mathbf{Rt}})$. Second, **Rt** signs Pub_i to generate a signature $\sigma_{\mathbf{Rt}}(\mathsf{Pub}_i)$. Finally, **Rt** sends sk_i to R_i via a secure channel and adds the node Node_i, containing Pub_i and $\sigma_{\mathbf{Rt}}(\mathsf{Pub}_i)$, into the PKTree. Notice that, in order to resist adversaries from forging a child node, a signature for the content is essential. The work of generating Node_i's child nodes is then transferred to R_i. In this way, each receiver can add its own child nodes itself, which shares the workload of building a PKTree. Another feature of the PKTree is that a receiver can also add a grandchild node directly. Specially, the supervisor **Rt** can add a child node for any of its descendant nodes. However, owners of two nodes in different branch cannot add a child node for each other, no matter which node is closer to the root than the other.

Fig. 2 shows an example of the PKTree. There is an enterprise and the owner of the enterprise supervises this PKTree. The owner setups the system, publishes the system parameter $\mathcal{GP}$ and holds the key msk secretly. The root node of the PKTree will be assigned to the CEO of the enterprise (Level 1 in Fig. 2). The root node contains the CEO's public information $\mathsf{Pub}_{\mathrm{CEO}}$ and a signature $\mathsf{Sig}_{\mathsf{msk}}(\mathsf{Pub}_{\mathrm{CEO}})$ produced by the enterprise owner. Notice that, $\mathsf{Pub}_{\mathrm{CEO}}$ is generated by the enterprise owner and includes the CEO's identity information, and the $\mathsf{Sig}_{\mathsf{msk}}(\mathsf{Pub}_{\mathrm{CEO}})$ serves as a proof on the validity of $\mathsf{Pub}_{\mathrm{CEO}}$.

Similar to the CEO, the COO, CFO and CTO will be assigned with nodes by the CEO (Level 2 in Fig. 2). Each node in the PKTree contains an employee's public information. For instance, the CTO's public information is generated and signed by the CEO. The child nodes of the CTO's node will be generated by CTO instead of CEO. In this way, it allows every employee of the enterprise to add nodes into the PKTree and share the workload of building the tree. As mentioned above, the CFO can add child nodes for $\mathsf{Node}_{\mathrm{Alice}}$ and $\mathsf{Node}_{\mathrm{Bob}}$, but not for $\mathsf{Node}_{\mathrm{Carter}}$ nor $\mathsf{Node}_{\mathrm{David}}$.

3.1 Definition of PKTree

A public key tree consists of the following four algorithms.

- $\mathsf{Setup}(1^\lambda)$: Given the security parameter 1^λ, this algorithm outputs a master secret key for the system msk and the global parameter $\mathcal{GP}$.
- $\mathsf{BuildTree}(\mathcal{GP}, \mathsf{msk}, \mathrm{ID}_{\mathbf{Rt}}, \tau)$: Given the global parameter $\mathcal{GP}$, the master secret key msk, the identity $\mathrm{ID}_{\mathbf{Rt}}$ of the root receiver **Rt**, and a time stamp τ, this algorithm outputs the root node Node_{root} of a public key tree Tree and the secret key sk_{root} of the node. The Node_{root} contains the identity $\mathrm{ID}_{\mathbf{Rt}}$ and a signature σ_{msk} signed with the master secret key msk. The signature will be used to verify if the Node_{root} is valid and bound to the identity of the root receiver **Rt**.
- $\mathsf{AddNode}(\mathcal{GP}, \mathsf{Tree}, \mathsf{sk}_i, \mathrm{ID}_j, \tau_j)$: Given $\mathcal{GP}$, Tree, the secret key sk_i of node Node_i, an identity ID_j and a time stamp τ, this algorithm outputs a child node Node_j of Node_i and Node_j's corresponding secret key sk_j. Notice that the time stamp included in a child node is valid if and only if it is newer than that in the parent node. Similar to the BuildTree algorithm, the newly added child node Node_j contains the identity ID_j and a signature $\sigma_{i,j}$ signed with the parent node's secret key sk_i.
- $\mathsf{DeleteNode}(\mathcal{GP}, \mathsf{Tree}, \mathsf{sk}_i, \tau_i')$: Given $\mathcal{GP}$, Tree, the secret key sk_i of node Node_i and a new time stamp τ_i', this algorithm deletes all the child nodes of Node_i and renew the Node_i using the new time stamp τ_i'. Because the τ_i' will be newer than any of the child nodes, all the old child nodes will be invalid. Hence, it deletes the child nodes.
- $\mathsf{VerifyTree}(\mathcal{GP}, \mathsf{Tree})$: Given $\mathcal{GP}$ and Tree, this algorithm outputs 1 if the time stamp of every node is newer than its parent node and the signature of every node is validly signed by its parent node (the root node is signed by the master key msk), and outputs 0 otherwise.

3.2 The Proposed PKTree

Based on the bilinear pairing, we give a concrete PKTree which will be utilized to build the HPEKS schemes.

- $\mathsf{Setup}(1^\lambda)$: Select three groups $\mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T$ with the same prime order p and a bilinear pairing $\hat{e}: \mathbb{G}_1 \times \mathbb{G}_2 \to \mathbb{G}_T$. Randomly select a master secret key $\mathsf{msk} = k \in \mathbb{Z}_p$ and compute the master public key $\mathsf{MPK} = g^k \in \mathbb{G}_1$, where g is a generator of $\mathbb{G}_1$. Finally, this algorithm outputs the global parameter $\mathcal{GP} = \{\mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T, p, g, h, \hat{e}, \mathsf{MPK}, H^*, H_1, H_2\}$ in which h is a generator of $\mathbb{G}_2$ and $H^*: \{0,1\}^* \to \mathbb{G}_2, H_1: \{0,1\}^* \to \mathbb{G}_2, H_2: \{0,1\}^* \to \mathbb{Z}_p$ are cryptographic hash functions.
- $\mathsf{BuildTree}(\mathcal{GP}, \mathsf{msk}, \mathrm{ID}_{\mathbf{Rt}}, \tau)$: Given $\mathcal{GP}$, msk, **Rt**'s identity $\mathrm{ID}_{\mathbf{Rt}}$ and a time stamp τ, **Rt** generates the first node of the tree as follow.
  - Randomly select a salt ς, and generate **Rt**'s public parameter $\mathsf{Pub}_{\mathbf{Rt}} = (\mathsf{pk}_{\mathbf{Rt}}, \overline{\mathbf{ID}}_{\mathbf{Rt}})$ and secret key $\mathsf{sk}_{\mathbf{Rt}} = H_1(\mathbf{ID}_{\mathbf{Rt}})^{\mathsf{msk}}$, where $\mathsf{pk}_{\mathbf{Rt}} = g_1^{H_2(\mathsf{sk}_{\mathbf{Rt}})}$ and $\overline{\mathbf{ID}}_{\mathbf{Rt}} = (\mathrm{ID}_{\mathbf{Rt}}, \varsigma)$.
  - Sign $\overline{\mathbf{ID}}_{\mathbf{Rt}}$ with the master secret key msk to generate a signature $\sigma_{0,\mathbf{Rt}} = \mathsf{Sig}_{\mathsf{msk}}(\mathsf{Pub}_{\mathbf{Rt}}) = (\sigma_{0,\mathbf{Rt},1}, \sigma_{0,\mathbf{Rt},2}) = (H^*(\mathsf{Pub}_{\mathbf{Rt}} \| \tau)^{\mathsf{msk}}, \tau)$.
  - Set the root node of the PKTree as $\mathsf{Node}_{\mathbf{Rt}} = (\mathsf{Pub}_{\mathbf{Rt}}, \sigma_{\mathbf{Rt}})$ and upload $\mathsf{Tree} = \{(\mathsf{Node}_{\mathbf{Rt}}, \mathrm{P}_0)\}$ to the cloud. Here P_0 indicates the position of the $\mathsf{Node}_{\mathbf{Rt}}$, i.e. the root, in the Tree.

Remark. The salt ς is used to randomize the receiver's public parameter. Users can verify the validity of the root node by check whether the following equation holds:

$$\hat{e}(g, \sigma_{0,\mathbf{Rt},1}) \quad \hat{e}(\mathsf{MPK}, H^*(\mathsf{Pub}_{\mathbf{Rt}} \| \sigma_{0,\mathbf{Rt},2})).$$

- AddNode($\mathcal{GP}$, Tree, sk_i, ID_j, τ_j): This algorithm is run by a receiver i, who owns a public parameter in the Tree, to generate a child node for a new receiver j. Given $\mathcal{GP}$, Tree, the receiver i's secret key sk_i, the lower-layer receiver j's identity ID_j and the time stamp τ_j, the algorithm runs the following steps and outputs the new node Node_j and its corresponding secret key sk_j:
  - Randomly select a salt ς_j and generate the public parameter $\mathsf{Pub}_j = (\mathsf{pk}_j, \overline{\mathbf{ID}}_j)$ and secret key $\mathsf{sk}_j = H_1(\overline{\mathbf{ID}}_j)^{H_2(\mathsf{sk}_i)}$, where $\mathsf{pk}_j = g^{H_2(\mathsf{sk}_j)}$ and $\overline{\mathbf{ID}}_j = (\mathrm{ID}_j, \varsigma_j)$;
  - Sign $\overline{\mathbf{ID}}_j$ with sk_i and generate a signature

$$\sigma_{i,j} \quad \mathsf{Sig}_{\mathsf{sk}_i}(\mathsf{Pub}_j) \quad (\sigma_{i,j,1}, \sigma_{i,j,2}) = (H^*(\mathsf{Pub}_j \| \tau_j)^{H_2(\mathsf{sk}_i)}, \tau_j);$$

  - Set the child node as $\mathsf{Node}_j = (\mathsf{Pub}_j, \sigma_{i,j})$.

Remark. The time stamp τ_j of the new node Node_j must be newer than that of the parent node Node_i. Users can verify the validity of this node by check whether the following equation holds:

$$\hat{e}(g, \sigma_{i,j,1}) = \hat{e}(\mathsf{pk}_i, H^*(\mathsf{Pub}_j \| \sigma_{i,j,2})).$$

- DeleteNode($\mathcal{GP}$, sk_i, Node_j, τ'_j): This algorithm revokes all the child nodes of Node_j. Given $\mathcal{GP}$, the receiver's secret key sk_i, the child node Node_j, this algorithm runs as follows:
  - Catch the current time stamp τ'_j and generate a new signature $\sigma'_{i,j} \quad \mathsf{Sig}_{\mathsf{sk}_i}(\mathsf{Pub}_j) \quad ((H^*(\mathsf{Pub}_j \| \tau'_j)^{H_2(\mathsf{sk}_i)}, \tau'_j))$;
  - Generate a new node $\mathsf{Node}'_j = (\mathsf{Pub}_j, \sigma'_{i,j})$;
  - Replace Node_j with Node'_j in the Tree and delete all the child nodes of Node_j. Note that, the secret key of the Node'_j is not changed and same as that of Node_j.

Remark. A naive method to revoke the child nodes is to delete them from Node_j directly. However, it cannot be proved that the child nodes have been indeed deleted, because the signatures attached to them and the time stamps of the child nodes are still valid, which may mislead other users. A better solution to the problem is to update the time stamp of the parent node (e.g., Node_j). According to our design of PKTree, time stamps of child nodes should be newer than that of the parent node. Therefore, if we update the parent node's time stamp with the latest one (as well as the corresponding signature on the new time stamp), all the child nodes are invalidated immediately. However, in many cases we do not need to delete all the child nodes. Hence, we design a two-layer-per-receiver structure, i.e., each receiver has two layer nodes. The upper-layer node is assigned by the receiver's parent node and the lower-layer nodes are generated by the receiver itself. Each lower-layer node is associated with a child node. In case the receiver needs delete one of the child nodes, it simply updates the associated lower-layer node, to make the time stamp of the lower-layer node newer than the child node. In such a way, it completes the deletion of the to-be-deleted child node, and keeps the other child nodes unaffected.

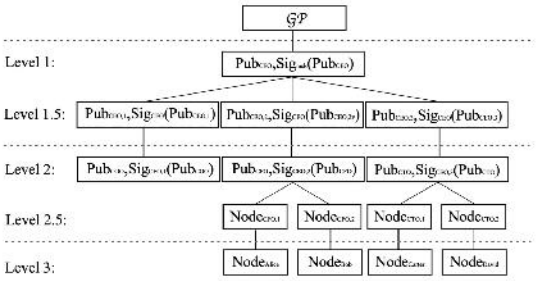


Fig. 3. The concrete PKTree.

- VerifyTree($\mathcal{GP}$, Tree): The Tree is valid if and only if, for all nodes in the tree, the equations

$$\hat{e}(g, \sigma_{0,\mathbf{Rt},1}) = \hat{e}(\mathsf{MPK}, H^*(\mathsf{Pub}_{\mathbf{Rt}} \| \sigma_{0,\mathbf{Rt},2}))$$
$$\hat{e}(g, \sigma_{i,j,1}) = \hat{e}(\mathsf{pk}_i, H^*(\mathsf{Pub}_j \| \sigma_{i,j,2})),$$

  hold, where **Rt** is the subscript of the root node and i, j indicate the subscripts of each pair of parent-child nodes.

3.3 Application of PKTree

Fig. 3 shows the construction of the PKTree in the real-world scenario. Before building the PKTree, the global parameter $\mathcal{GP}$ should be given by the enterprise owner. Slightly different from the aforementioned example, a receiver could own multiple nodes, e.g., the CEO owns a Level-1 node (assigned by the enterprise owner) and three Level-1.5 nodes (assigned by itself) in the PKTree.

To delete a node, e.g., the CTO's node, as shown in Fig. 4, the CEO runs the DeleteNode algorithm to update the Level-1.5 node $\mathsf{Node}_{\mathsf{CEO},3}$ which is the parent node of the CTO's node. The public parameter $\mathsf{Pub}_{\mathsf{CEO},3}$ in the updated node $\mathsf{Node}_{\mathsf{CEO},3}$ is not changed but the corresponding signature $\mathsf{Sig}_{\mathsf{CEO}}{}^{new}$ is updated with the latest time stamp $\tau_{\mathsf{CEO},3}{}^{new}$. Because the $\tau_{\mathsf{CEO},3}{}^{new}$ is newer than the time stamps of its child nodes, all its child nodes will be invalid, in which way the child nodes are revoked. Notice that the rest branches of the $\mathsf{Node}_{\mathsf{CEO}}$ are not effected.

In many cases, the CEO does not want to revoke all of the CTO's child nodes. Thus, after deleting the CTO's node, the CEO should assign a new node to an employee to manage the CTO's child nodes. For example, as shown in Fig. 5, to replace the CTO with the employee "*David*", the CEO runs the AddNode algorithm to replace the CTO's node with David's new node and sends David the corresponding secret key $\mathsf{sk}_{\mathsf{David}}$ via a secure channel. Finally, David and his child nodes use the AddNode algorithm to build up this branch iteratively.

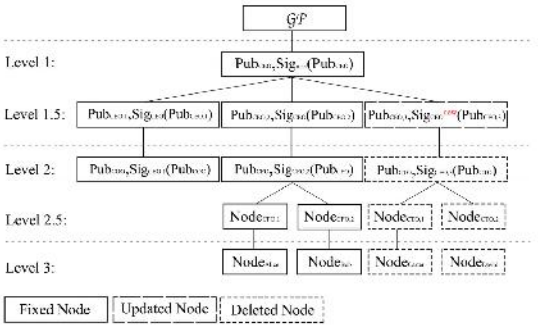


Fig. 4. Delete nodes from the PKTree.

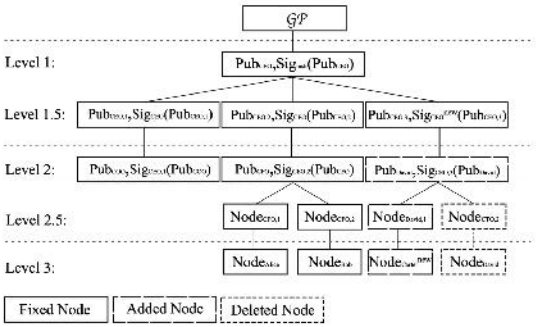


Fig. 5. Add nodes into the PKTree.

4 The Proposed HPEKS Schemes

In this section, based on the proposed PKTree, we introduce how to build an HPEKS scheme by giving a semi-generic HPEKS scheme. To keep the integrality of the encrypted keywords and plaintext, we give an enhanced concrete HPEKS scheme which supports decrypting ciphertexts.

4.1 The Semi-Generic HPEKS Scheme From PEKS

Thanks to the compatibility of PKTree, we can construct a semi-generic HPEKS scheme from pairing-based PEKS schemes. Let $\mathsf{PEKS} = \{\mathsf{KeyGen}', \mathsf{Encrypt}', \mathsf{Trapdoor}', \mathsf{Test}'\}$ be a pairing-based PEKS scheme in which the secret key sk is an element of $\mathbb{Z}_p$ and the public key is of the form g^{sk}.

- $\mathsf{Encrypt}(\mathcal{GP}, \mathsf{Tree}, \mathsf{Node}_j, w)$: Parse the receiver's node Node_j as $(\mathsf{pk}_j, \overline{\mathsf{ID}}_j, \sigma_{i,j})$ and verify the validity of the public parameter $\mathsf{pk}_j, \overline{\mathsf{ID}}_j$ and the signature $\sigma_{i,j}$. Run the PEKS $\mathsf{Encrypt}'(\mathcal{GP}, \mathsf{pk}_j, w)$ algorithm and return the generated ciphertext C_w.
- $\mathsf{Trapdoor}(\mathcal{GP}, \mathsf{sk}_j, w)$: This algorithm returns the trapdoor T_w generated by PEKS $\mathsf{Trapdoor}'(\mathcal{GP}, H_2(\mathsf{sk}_j), w)$.
- $\mathsf{Test}(\mathcal{GP}, C, T_w)$: This algorithm returns the trapdoor T_w generated by PEKS $\mathsf{Test}'(\mathcal{GP}, C, T_w)$.

Remark. The Setup, BuildTree, AddNode, DeleteNode and VerifyTree algorithms of the semi-generic HPEKS scheme are the same as those in Section 3.2 and thus are omitted here.

4.2 Designated-Tester Decryptable HEPKS Scheme

The semi-generic HPEKS scheme supports a group of users to delegate the cloud server to search some keyword over the received ciphertexts without decryption. Specially, a receiver can search over ciphertexts sent to receivers supervised by him. However, it is based on the existing PEKS schemes and the traditional PEKS schemes do not support the decryption function. Although the PKE/PEKS scheme proposed by Baek *et al.* [30] supports decryption of ciphertexts, there is a restriction that the plaintext in the PKE/PEKS scheme is length-limited. To resolve this problem, we propose an advanced scheme named *designated-tester decryptable hierarchical public key encryption with keyword search* (dDHPEKS), in which plaintext of variable length can be encrypted and the corresponding ciphertext is decryptable. Additionally, only the designated tester can test whether a ciphertext contains the same keyword as that in the queried trapdoor.

- $\mathsf{Setup}(1^\lambda)$: Given a security parameter λ, this algorithm setups the system via running the following steps.
  - Pick three groups $\mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T$ of same prime order p with a bilinear pairing $\hat{e} : \mathbb{G}_1 \times \mathbb{G}_2 \to \mathbb{G}_T$.
  - Randomly select two generators $g \in \mathbb{G}_1$, $h \in \mathbb{G}_2$.
  - Randomly select $\mathsf{msk} = k \in \mathbb{Z}_p$ and compute $\mathsf{MPK} = g^k$.
  - Select hash functions $H^* : \{0,1\}^* \to \mathbb{G}_2$, $H : \{0,1\}^* \to \mathbb{Z}_p$, $H_1 : \{0,1\}^* \to \mathbb{G}_2$, $H_2 : \mathbb{G}_2 \to \mathbb{Z}_p$, $H_3 : \{0,1\}^* \to \{0,1\}^{\ell_1}$, $H_4 : \{0,1\}^* \to \mathbb{G}_2$, $H_5 : \{0,1\}^* \to \{0,1\}^{\ell_2}$, where ℓ_1, ℓ_2 are two fixed lengths.
  - Output the msk and the $\mathcal{GP} = (\mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T, p, \hat{e}, g, h, \mathsf{MPK}, H^*, H, H_1, H_2, H_3, H_4, H_5)$.
- $\mathsf{KeyGen}_{server}(\mathcal{GP})$: Given the global parameter $\mathcal{GP}$, this algorithm randomly selects $\mathsf{sk}_s \in \mathbb{Z}_p$ as the designated test server's secret key and computes the corresponding public key $\mathsf{pk}_s = h^{\mathsf{sk}_s}$. Output the server's public/secret key pair $(\mathsf{pk}_s, \mathsf{sk}_s)$.
- $\mathsf{Encrypt}(\mathcal{GP}, \mathsf{Tree}, \mathsf{Node}_j, w, M)$: Randomly select $r, s \in \mathbb{Z}_p, K \in \{0,1\}^{\ell_1}$ and encrypt the keyword w and message M as below

$$C_1 = \hat{e}(\mathsf{pk}_j^{r \cdot H(w)}, H_1(\overline{\mathsf{ID}}_j)), C_2 = g^r, C_3 = h^r,$$
$$C_4 = K \oplus H_3(\mathsf{pk}_j^s), C_5 = g^s, C_7 = \mathsf{AESEnc}_K(M),$$
$$C_6 = H_4(C_1, C_2, C_3, C_4, C_5, H_5(C_7))^s.$$

  Output the ciphertext $C_{w,M} = (C_1, C_2, C_3, C_4, C_5, C_6, C_7)$.
- $\mathsf{Decrypt}(\mathcal{GP}, \mathsf{sk}_j, C)$: Parse C as $C = (C_1, \ldots, C_7)$. Return $\perp$ if either of the equations

$$\hat{e}(C_5, H_4(C_1, \ldots, C_5, H_5(C_7))) = \hat{e}(g, C_6),$$
$$\hat{e}(C_2, h) = \hat{e}(g, C_3),$$

  does not hold, and return the plaintext M otherwise, where

$$M = \mathsf{AESDec}_K(C_7), \quad K = C_4 \oplus H_3(C_5^{H_2(\mathsf{sk}_j)}).$$

- Trapdoor($\mathcal{GP}, \mathsf{sk}_j, \mathsf{pk}_s, w$): Randomly select $t \in \mathbb{Z}_p$ and generate trapdoor as follows:

$$T_1 = h^{t_1} \cdot H_1(\overline{\mathrm{ID}}_j)^{H_2(\mathsf{sk}_j) \cdot H(w)},$$
$$T_2 = \mathsf{pk}_s^t, T_3 = g^{t_2}.$$

- Test($\mathcal{GP}, C, T, \mathsf{sk}_s$): Parse C as $(C_1, \ldots, C_7)$ and parse T as (T_1, T_2, T_3). The algorithm outputs 1 if the equation

$$\hat{e}(C_2, T_1 / T_2^{\overline{\mathsf{sk}_s}}) \quad C_1 \cdot \hat{e}(T_3, C_3),$$

holds, and 0 otherwise.

Remark. The BuildTree, AddNode and ReplaceChild algorithms are the same as those in the Semi-Generic HPEKS scheme, and thus omitted here.

Our scheme is advantageous over HIBE (with keyword search) in the sense that it enjoys the property of *transparency*. To share an encrypted record to multiple receivers in a hierarchy, a sender in our scheme does not need to know the hierarchy of these receivers. It only needs to encrypt the record under the public key of the receiver at the lowest level. In contrast, a sender in HIBE does need to know the identity hierarchy.

5 Security Analysis

In this section, we provide the threats model of HPEKS, then give the formal security definitions, and finally prove our proposal is secure under the security definitions.

5.1 Security Models

5.1.1 Secret Key One-Way Security Against Collusion

Here we consider the key privacy of the proposed PKTree structure. The following two games are given to prove the semantic security of PKTree against low layer collusion and same layer collusion attacks, respectively.

Game K1: Security against low layer collusion attacks:

- Setup: The challenger $\mathcal{C}$ sends the adversary $\mathcal{A}$ the global parameter $\mathcal{GP}$ and the challenged node Node_i.
- Phase 1: $\mathcal{A}$ can issue at most q_E queries to extract oracle $\mathcal{O}_E$ to obtain Node_i's child nodes secret keys $\{\mathsf{sk}_j\}_1^{q_E}$:
  - $\mathcal{O}_E(\mathrm{ID}_j)$: Given an identity, the oracle returns the corresponding node Node_j and secret key sk_j.
- Guess: $\mathcal{A}$ outputs a secret key $\mathsf{sk}_{\mathcal{A}}$ and wins the game if $\mathsf{sk}_{\mathcal{A}} = \mathsf{sk}_i$.

Let $Adv_{\mathcal{A}}^{K1} = \Pr[\mathsf{sk}_{\mathcal{A}} = \mathsf{sk}_i]$ denote the advantage of $\mathcal{A}$ to win Game K1.

Game K2: Security against same layer collusion attacks:

- Setup: The challenger $\mathcal{C}$ sends the adversary $\mathcal{A}$ the global parameter $\mathcal{GP}$ and the parent node Node_i.
- Phase 1: $\mathcal{A}$ can issue at most q_E queries to extract oracle $\mathcal{O}_E$ to obtain Node_i's child nodes secret keys $\{\mathsf{sk}_j\}_1^{q_E}$:
  - $\mathcal{O}_E(\mathrm{ID}_j)$: Given an identity, the oracle returns the corresponding node Node_j and secret key sk_j.
- Challenge: $\mathcal{A}$ chooses an identity ID_c as the challenge identity and sends it to $\mathcal{C}$. The constraint is that ID_c cannot be submitted to $\mathcal{O}_E$ in Phase 1. $\mathcal{C}$ returns the corresponding child node $\mathsf{Node}_{\mathrm{ID}_c}$ of Node_i to $\mathcal{A}$.
- Phase 2: $\mathcal{A}$ issues queries to the oracle same as in Phase 1 with the constraint that ID_c cannot appear in $\mathcal{O}_E$.
- Guess: $\mathcal{A}$ outputs a secret key $\mathsf{sk}_{\mathcal{A}}$ and wins the game if $\mathsf{sk}_{\mathcal{A}} = \mathsf{sk}_{\mathrm{ID}_c}$ where sk_i is the secret key of $\mathsf{Node}_{\mathrm{ID}_c}$.

Let $Adv_{\mathcal{A}}^{K2} = \Pr[\mathsf{sk}_{\mathcal{A}} = \mathsf{sk}_{\mathrm{ID}_c}]$ denote the advantage of $\mathcal{A}$ to win Game K2.

Definition 5. *A dDHPEKS scheme is secure against key collusion attacks if for any PPT adversary $\mathcal{A}$, $Adv_{\mathcal{A}}^{K1}$ and $Adv_{\mathcal{A}}^{K2}$ are both negligible.*

5.1.2 Keyword Privacy Against Chosen Keyword Attacks

We give the following security game to define the keyword privacy against chosen keyword attacks.

Game W1: Trapdoor privacy:

- Setup: The challenger $\mathcal{C}$ sends the adversary $\mathcal{A}$ the global parameter $\mathcal{GP}$ and the parent node Node_i.
- Phase 1: $\mathcal{A}$ can issue at most q_E and q_T queries to extract oracle $\mathcal{O}_E$ and trapdoor oracle $\mathcal{O}_T$, respectively.
  - $\mathcal{O}_E(\mathrm{ID}_j)$: Given an identity ID_j, the oracle returns the corresponding node Node_j and secret key sk_j.
  - $\mathcal{O}_T(\mathrm{ID}_j, w)$: Given an identity ID_j and a keyword w, the oracle returns a corresponding trapdoor $T_{\mathrm{ID}_j, w}$.
- Challenge: $\mathcal{A}$ chooses an identity ID_c and two keywords w_0, w_1. The constraint is that ID_c cannot be submitted to $\mathcal{O}_E$ in Phase 1. $\mathcal{C}$ generates the node $\mathsf{Node}_{\mathrm{ID}_c}$ and randomly selects a bit $b \in \{0, 1\}$. In the next step, $\mathcal{C}$ generates a trapdoor T_{w_b}. Finally, $\mathcal{C}$ returns $\mathsf{Node}_{\mathrm{ID}_c}$ and T_{w_b} to $\mathcal{A}$.
- Phase 2: $\mathcal{A}$ issues queries to the oracles same as in Phase 1.
- Guess: $\mathcal{A}$ outputs a bit $b_{\mathcal{A}}$ and wins the game if $b' = b$.

Let $Adv_{\mathcal{A}}^{W1} = \Pr[w_{\mathcal{A}} = w_c]$ denote the advantage of $\mathcal{A}$ to win Game W1.

Definition 6. *A dDHPEKS scheme is IND-TW-CPA secure if for any PPT adversary $\mathcal{A}$, the advantage $Adv_{\mathcal{A}}^{W1}$ is negligible.*

Game W2: Ciphertext indistinguishability against chosen keyword attacks:

- Setup: Same as that in Game W1.
- Phase 1: Same as that in Game W1.
- Challenge: $\mathcal{A}$ chooses an identity ID_c, two keywords w_0, w_1 and a plaintext M. The constraint is that ID_c cannot appear in $\mathcal{O}_E$ in Phase 1. $\mathcal{C}$ generates the node $\mathsf{Node}_{\mathrm{ID}_c}$ and randomly selects a bit $b \in \{0, 1\}$. In the next step, $\mathcal{C}$ generates a ciphertext $C_{[w_b, M]}$. Finally, $\mathcal{C}$ returns $\mathsf{Node}_{\mathrm{ID}_c}$ and $C_{[w_b, M]}$ to $\mathcal{A}$.
- Phase 2: $\mathcal{A}$ still can issue queries to the oracle same as in phase 1 with the constraint that ID_c cannot appear in $\mathcal{O}_E$.

- Guess: $\mathcal{A}$ outputs a bit b' and wins the game if b' b.

Let $Adv_{\mathcal{A}}^{W2} = |\Pr[b' = b] - \frac{1}{2}|$ denote the advantage of $\mathcal{A}$ to win Game W2.

Definition 7. *A dDHPEKS scheme is IND-CW-CPA secure if for any PPT adversary $\mathcal{A}$, the advantage $Adv_{\mathcal{A}}^{W2}$ is negligible.*

5.1.3 Plaintext Privacy Against Chosen Ciphertext Attacks

The following game is to define the semantic security against chosen ciphertext attacks.

Game M: Ciphertext indistinguishability against chosen plaintext attacks:

- Setup: Same as that in Game W1.
- Phase 1: $\mathcal{A}$ can issue at most q_E queries to the Extract Oracle $\mathcal{O}_E$ below.
  - $\mathcal{O}_E(\mathrm{ID}_j)$: Given an identity ID_j, the oracle returns the corresponding node Node_j and secret key sk_j.
- Challenge: $\mathcal{A}$ chooses an identity ID_c, a keyword w and two plaintext M_0, M_1. The constraint is that ID_c cannot be submitted to $\mathcal{O}_E$ in Phase 1. $\mathcal{C}$ generates the node $\mathsf{Node}_{\mathrm{ID}_c}$ and randomly selects a bit $b \subset \{0, 1\}$. In the next step, $\mathcal{C}$ generates a ciphertext C_{w,M_b}. Finally, $\mathcal{C}$ returns $\mathsf{Node}_{\mathrm{ID}_c}$ and C_{w,M_b} to $\mathcal{A}$.
- Phase 2: $\mathcal{A}$ issues queries to the oracle same as in Phase 1 with the constraints that ID_c cannot be submitted to $\mathcal{O}_E$ and $C_{|w,M_b}$ cannot appear in $\mathcal{O}_D$.
- Guess: $\mathcal{A}$ outputs a bit b' and wins the game if $b' = b$.

Let $Adv_{\mathcal{A}}^{M}$ $\Pr[b'$ $b] - \frac{1}{2}|$ denote the advantage of $\mathcal{A}$ to win Game M.

Definition 8. *A dDHPEKS scheme is IND-CCA secure if for any PPT adversary $\mathcal{A}$, the advantage $Adv_{\mathcal{A}}^{M}$ is negligible.*

5.2 Security Analysis of dDHPEKS Scheme

We prove our dDHPEKS scheme is secure under the above security model.

Theorem 1. *Our proposed dDHPEKS scheme is secure against key collusion attacks if the CDH assumption holds.*

The proof of Theorem 1 is straightforward. The structure of our PKTree consists of multiple layers of public identity information. The corresponding secret keys of each layer is generated by their parent node, which is the same as the key extract process of the pairing based IBE schemes. Because the pairing based IBE scheme is secure against collusion attacks if the CDH assumption holds, our proposed dDHPEKS scheme enjoys security against key collusion attacks as well.

Theorem 2. *Our dDHPEKS scheme is IND-TW-CPA secure if the DLIN assumption holds.*

Proof. The challenger $\mathcal{C}$ is given a DLIN instance $\mathrm{Ins}_{\mathrm{DLIN}} = (\mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T, p, \hat{e}, u_1, u_2, u_1^x, u_2^y, v, Z)$. To distinguish whether Z is equal to $v^{x+y} \in \mathbb{G}_2$ or a random element $v^r \in \mathbb{G}_2$, the challenger $\mathcal{C}$ plays the following game with the adversary $\mathcal{A}$ who tries to break the security of our dDHPEKS scheme.

Game W1:

- Setup: $\mathcal{C}$ randomly selects $\mathsf{msk} = k \in \mathbb{Z}_p$, computes the master public key $\mathsf{MPK} = u_2^k$ and builds the PKTree. Given an identity ID_i, $\mathcal{C}$ generates the corresponding node Node_i and secret key sk_i. Finally, $\mathcal{C}$ returns the global parameter $\mathcal{GP} = \{\mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T, p, \hat{e}, g = u_2, h = v, \mathsf{MPK}, H, H_1, H_2, H_3, H_4, H_5\}$. Additionally, $\mathcal{C}$ returns the server's public key $\mathsf{pk}_s = u_1$ and the node Node_i.
- Phase 1: $\mathcal{A}$ can issue at most q_E, q_T, q_H and q_{H_1} queries to the following oracles, respectively.
  - Extract Oracle $\mathcal{O}_E$: Given an identity ID_j, it returns the corresponding result in records if ID_j has been asked before, otherwise, returns the corresponding results as follows:

$$\begin{aligned}\mathsf{Node}_j &= (\mathsf{Pub}_j, \sigma_{i,j}) = ((\mathsf{pk}_j \| \overline{\mathbf{ID}}_j), \sigma_{i,j}) \\ &= ((g_1^{H_2(\mathsf{sk}_j)} \| \mathrm{ID}_j \| \varsigma_j), \mathsf{Sig}_{\mathsf{sk}_i}(\mathsf{Pub}_j)), \\ \mathsf{sk}_j &= H_1(\overline{\mathbf{ID}}_i)^{H_2(\mathsf{sk}_i)}.\end{aligned}$$

  - Trapdoor Oracle $\mathcal{O}_T$: Given an identity ID_j and a keyword w, this oracle randomly selects $t_1, t_2 \subset \mathbb{Z}_p$ and returns $T_{w,\mathrm{ID}_j} = (T_1, T_2, T_3)$ as follows:

$$\begin{aligned}T_1 &= h^{t_1+t_2} \cdot H_1(\mathbf{ID}_j)^{H_2(\mathsf{sk}_j)\cdot H(w)}, \\ T_2 &= \mathsf{pk}_s^{t_1} = u_1^{t_1}, T_3 = g^{t_2}.\end{aligned}$$

- Challenge: $\mathcal{A}$ sends $\mathcal{C}$ two keywords w_0, w_1 and a node Node_r, parsed as $(\mathsf{pk}_r \| \overline{\mathbf{ID}}_r, \sigma_{i,r})$. $\mathcal{C}$ reveals the secret key sk_r of Node_r and returns a trapdoor $T_{w_b} = (T_1^*, T_2^*, T_3^*)$, where b is a randomly chosen bit to decide which keyword is encrypted

$$\begin{aligned}T_1^* &= Z \cdot H_1(\overline{\mathbf{ID}}_r)^{H_2(\mathsf{sk}_r)\cdot H(w_b)}, \\ T_2^* &= u_1^x, T_3^* = u_2^y.\end{aligned}$$

- Phase 2: $\mathcal{A}$ issues queries as in Phase 1.
- Guess: $\mathcal{A}$ returns a bit b' and wins the game if $b' = b$.

$Adv_{\mathcal{A}}^{W1} = \Pr[b' = b]$ $\frac{1}{2} = \epsilon$ denotes the advantage of $\mathcal{A}$'s winning the game. When $\mathcal{A}$ wins the game, $\mathcal{C}$ returns $Z = v^{x+y}$ to DLIN challenger, otherwise, Z is equal to a random element $R \overset{\$}{\leftarrow} \mathbb{G}_2$. $\mathcal{C}$ will break the DLIN assumption with the advantage $\epsilon_{\mathcal{C}} = \epsilon$. Hence, $Adv_{\mathcal{A}}^{W1}$ is negligible if the DLIN assumption holds. □

Theorem 3. *Our proposed dDHPEKS scheme is IND-CW-CPA secure if the DBDH assumption holds.*

Proof. The challenger $\mathcal{C}$ is given a DBDH instance $\mathrm{Ins}_{\mathrm{DBDH}} = (\mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T, p, \hat{e}, g, g^x, g^y, h, h^y, h^z, Z)$. To distinguish whether Z is equal to $\hat{e}(g, h)^{xyz}$ or Z is a randomly chosen element in $\mathbb{G}_T$, the challenger $\mathcal{C}$ plays the following game with the adversary $\mathcal{A}$ who tries to break the security of our dDHPEKS scheme.

Game W2:

- Setup: $\mathcal{C}$ initializes the system by giving the global parameter $\mathcal{GP} = (\mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T, p, \hat{e}, g, h, \mathsf{MPK}, H, H_2, H_3, H_4, H_5)$.
- Phase 1: $\mathcal{A}$ can issue at most q_H, q_k and q_T queries to the hash oracle $\mathcal{O}_{H_i}$, extract oracle $\mathcal{O}_E$ and the trapdoor oracle $\mathcal{O}_T$, respectively.

- Random Oracle $\mathcal{O}_{H_1}$: Given some public information $\overline{\mathbf{ID}}_j$, it randomly selects $r_j \in \mathbb{Z}_p$ and returns the hash value $H_1(\overline{\mathbf{ID}}_j) = (h^z)^{r_j}$.
- Extract Oracle $\mathcal{O}_E$: Same as that in Game W1.
- Trapdoor Oracle $\mathcal{O}_T$: Same as that in Game W1.

- **Challenge:** $\mathcal{A}$ sends $\mathcal{C}$ a new identity ID_c, which did not appear in $\mathcal{O}_E$, two keywords w_0, w_1 and a plaintext M. $\mathcal{C}$ adds a child node Node_c corresponding with this identity into the tree and generates a trapdoor $C_{w_b,M}$, where the the random bit b decides which keyword is encrypted in this ciphertext. Finally, $\mathcal{C}$ randomly selects $s \in \mathbb{Z}_p$ and returns the node Node_c and the Ciphertext $C_{w_b,M} = (C_1^*, \ldots, C_7^*)$ to $\mathcal{A}$

$$\begin{aligned}
\mathsf{Node}_c &= (\mathsf{Pub}_c, \sigma_{i,c}) = ((\mathsf{pk}_c, \overline{\mathbf{ID}}_c), \sigma_{i,c}) \\
&= ((g^x, (h^z)^{r_c}), \mathsf{Sig}_{\mathsf{sk}_i}(\mathsf{Pub}_c)), \\
C_1^* &= Z^{rc \cdot H(w)},\ C_2^* = g^y,\ C_3^* = h^y, \\
C_4^* &= K \oplus H_3(g^{rc \cdot s}),\ C_5^* = g^s,\ C_7^* = \mathsf{AESEnc}_K(M), \\
C_6^* &= H_4(C_1, \ldots, C_5, H_5(C_7))^s,
\end{aligned}$$

  where $\overline{\mathbf{ID}}_c = (h^z)^{rc}$ and r_c is recalled from records of the hash oracle $\mathcal{O}_{H_1}$.
- **Phase 2:** $\mathcal{A}$ still can issue queries to the oracles same as in phase 1 except that the tuples $\langle \mathrm{ID}_c, w_0 \rangle$ and $\langle \mathrm{ID}_c, w_1 \rangle$ cannot appears in the trapdoor oracle $\mathcal{O}_T$.
- **Guess:** $\mathcal{A}$ returns a bit b' and wins the game if $b' = b$.

Assume the advantage $Adv_{\mathcal{A}}^{W2} = \epsilon$ of $\mathcal{A}'$ winning the game is non-negligible, $\mathcal{C}$ can break the DBDH assumption with the non-negligible advantage $\epsilon_C = \epsilon$. Hence, $Adv_{\mathcal{A}}^{W2}$ is negligible if the DBDH assumption holds. □

Theorem 4. *Our proposed dDHPEKS scheme is IND-CPA secure if AES encryption is IND-CPA secure and the CDH assumption holds.*

Proof. The dDHPEKS scheme leverages the AES to encrypt the plaintext M and hides the session key K into C_4. Hence, if C_4 does not leak any information about the encryption key K, security of our dDHPEKS will be based on that of AES. The following game is played between a PPT adversary $\mathcal{A}$ and the challenger $\mathcal{C}$. Given a CDH instance $\mathrm{Ins}_{\mathrm{CDH}} = (\mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T, p, \hat{e}, g, g^x, g^y)$, $\mathcal{C}$ works as follows.

Game M:

- **Setup:** $\mathcal{C}$ initializes the system by giving the global parameter $\mathcal{GP} = (\mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T, p, \hat{e}, g, h, \mathsf{MPK}, H, H_1, H_2, H_3)$.
- **Phase 1:** $\mathcal{A}$ can issue at most q_H and q_E queries to the hash oracles $\mathcal{O}_{H_3}, \mathcal{O}_{H_5}$ and extract oracle $\mathcal{O}_E$, respectively.
  - Random Oracle $\mathcal{O}_{H_3}$: Given an element $g^a \in \mathbb{G}_1$, it returns an ℓ_1-bit random number h^* as the hash value $H_3(g^a)$.
  - Random Oracle $\mathcal{O}_{H_5}$: Given an input T, it randomly selects $r_T \in \mathbb{Z}_p$ and returns $H_5(T) = (g^y)^{r_T}$.
  - Extract Oracle $\mathcal{O}_E$: Same as that in Game W1.
- **Challenge:** $\mathcal{A}$ sends $\mathcal{C}$ a new identity ID_c, a keyword w and two plaintexts M_0, M_1. $\mathcal{C}$ adds a child node Node_c corresponding with this identity into the tree and generates a trapdoor C_{w,M_b}, where the the random bit b decides which plaintext is encrypted in this ciphertext. Finally, $\mathcal{C}$ returns the node Node_c and the Ciphertext $C_{w,M_b} = (C_1^*, \ldots, C_7^*)$ to $\mathcal{A}$

$$\begin{aligned}
\mathsf{Node}_c &= (\mathsf{Pub}_c, \sigma_{i,c}) = ((\mathsf{pk}_c, \overline{\mathbf{ID}}_c), \sigma_{i,c}) \\
&= ((g^x, \overline{\mathbf{ID}}_c), \mathsf{Sig}_{\mathsf{sk}_i}(\mathsf{Pub}_c)), \\
C_1^* &= \hat{e}(\mathsf{pk}_c^{r \cdot H(w)}, H_1(\overline{\mathbf{ID}}_c)),\ C_2^* = g^r,\ C_3^* = h^r, \\
C_4^* &= K \oplus h_c^*,\ C_5^* = g^{y \cdot s},\ C_7^* = \mathsf{AESEnc}_K(M), \\
C_6^* &= H_4(C_1, C_2, C_3, C_4, C_5, H_5(C_7))^s,
\end{aligned}$$

  where r, s are randomly chosen from $\mathbb{Z}_p$ and h_c^* is an ℓ_1-bit random number.
- **Phase 2:** $\mathcal{A}$ still can issue queries to the oracles same as in phase 1 except that the ciphertext C_{w,M_b} cannot appears in the decrypt oracle $\mathcal{O}_D$.
- **Guess:** $\mathcal{A}$ returns a bit b' and wins the game if $b' = b$.

We define an event, denoted by **E**, that $\mathcal{A}$ issues $g^a = g^{xy+s}$ to the hash oracle $\mathcal{O}_{H_3}$. In case **E** happens, the challenger $\mathcal{C}$ solves the CDH problem via computing $g^{xy} = (g^a)^{\frac{1}{s}}$. If the CDH assumption holds, **E** happens with a negligible probability. In another case, **E** does not happen, the ciphertext C_4 is random in $\mathcal{A}$'s view and the session key K can be revealed with a negligible probability. Hence, in this game, $\mathcal{A}$'s winning advantage $Adv_{\mathcal{A}}^{M}$ is equal to or less than a negligible probability if AES encryption is IND-CPA secure and the CDH assumption holds. □

6 Comparison and Experiments

To evaluate the running efficiency of the proposed dDHPEKS scheme, we compare it with previous searchable encryption schemes [7], [30], [31]. Table 1 shows that our proposal has the comparable running efficiency with the compared schemes. To achieve higher level security and more functionality, it is reasonable to sacrifices certain running efficiency. Among the four schemes, only our dDHPEKS scheme can not only resist outside keyword guessing attacks but also achieve integration of PKE and PEKS. Moreover, based on the PKTree, our proposal supports node supervision, which cannot be achieved in previous PEKS schemes.

We implemented our dDHPEKS scheme and the compared schemes denoted by BDOP-PEKS [7], BSS-PKE/PEKS [30] and HL-PEKS [31], respectively, utilizing the C language and PBC library [2] in Ubuntu OS 19.04 on a laptop with a 2.3 GHz Intel Core i5 CPU and 8 GB 2133 MHz LPDDR3 memory. A Type-A pairing was chosen and used to initialize the system, which owns the same security level as a 1024-bit RSA encryption. To apply the keyword search to some dataset, the first step is to extract keywords. Each record in the dataset should be associated with a keyword. The keyword will be encrypted with the proposed encryption scheme and the record would be encrypted with some other encryption scheme, such as AES. Inverted index is

TABLE 1
Comparisons With PEKS Schemes in Literature

Scheme	Encrypt	Trapdoor	Test	KGA	INT	INT/L	HMSE
BDOP-PEKS [7]	$2Exp_{\mathbb{G}_1}$ + 2Hash + 1Pairing	$1Exp_{\mathbb{G}_1}$ + 1Hash	1Hash + 1Pairing	No	No	No	No
BSS-PKE/PEKS [30]	$3Exp_{\mathbb{G}_1}$ + 4Hash + 1Pairing	$1Exp_{\mathbb{G}_1}$ + 1Hash	1Hash + 1Pairing	No	Yes	No	No
HL-PEKS [31]	$2Exp_{\mathbb{G}_1}$ + 2Hash + 3Pairing	$3Exp_{\mathbb{G}_1}$ + 1Hash	$1Exp_{\mathbb{G}_1}$ + 1Hash + 1Pairing	Yes	No	No	No
Our dDHPEKS	$4Exp_{\mathbb{G}_1}$ +$2Exp_{\mathbb{G}_2}$ + 5Hash + 1Pairing + 1AES	$1Exp_{\mathbb{G}_1}$ + $3Exp_{\mathbb{G}_2}$ + 3Hash	$1Exp_{\mathbb{G}_2}$ + 2Pairing	Yes	Yes	Yes	Yes

KGA: The schemes can resist outside keyword guessing attacks;
INT: The schemes integrate the encrypted keyword and plaintext;
INT/L: The schemes are adaptable of any length of the integrated plaintexts;
HMSE: The schemes support hierarchical multi-user keyword search.

usually used to support efficient encrypted data search. Basically, inverted index contains a list of (encrypted) keywords, and each keyword is associated with a queue of records which contains the keyword. In this system, the keywords can be found in a common dictionary with a high probability and different bit-length of keywords theoretically do not affect the efficiency of the program, because each keyword will be hashed into a fixed-length string before encryption. Hence, we choose the Oxford dictionary as the keyword space in the implementation and all the words in it are taken as input of the related algorithms of the scheme.

As shown in Fig. 6, compared with other three schemes, the encryption algorithm of our dDHPEKS scheme is slower. The reason is that nearly half of the computing overhead is to encrypt the session key K, which does not exist in BDOP-PEKS nor HL-PEKS schemes.

Notice that the overhead of AES encryption algorithm of our dDHPEKS scheme was not included in the experiment results, because it is indeterminate and affected by the length of the plaintext.

Fig. 7 shows that our dDHPEKS scheme owns an efficient trapdoor algorithm, which is similar to the BDOP-PEKS and the BSS-PKE/PEKS schemes and more efficient than the HL-PEKS scheme.

The overhead of test algorithms of the four schemes are shown in Fig. 8. To search a keyword over 1,000 ciphertexts, the average overhead of using the BDOP-PEKS and the BSS-PKE/PEKS schemes are nearly 0.75 second, the HL-PEKS scheme is 2.00 seconds and our dDHPEKS scheme is about 2.85 seconds. Even though the former two schemes are faster, they cannot resist the outside offline keyword guessing attacks. With the same security level, the searching efficiency of our dDHPEKS scheme is comparable with HL-PEKS.

Consider the scenario in which the sender sends an encrypted keyword to multiple receivers in a hierarchy, i.e., receiver R_2 is the child-node of R_1, receiver R_3 is the child-node of R_2, and so on. If we use a traditional PEKS scheme, the sender has to encrypt the keyword multiple times (or use a re-encryption mechanism to re-encrypt an existing ciphertext to new ciphertexts under other receivers' public keys), and sends ciphertexts to the respective receivers. Hence the encryption time she has to spend is linear in the number of receivers. However, if we use the proposed dDHPEKS scheme to implement the scenario, the sender needs only to encrypt the keyword for the receiver at the lowest level among all the receivers, and then all the receivers would be able to test the encrypted keyword, since

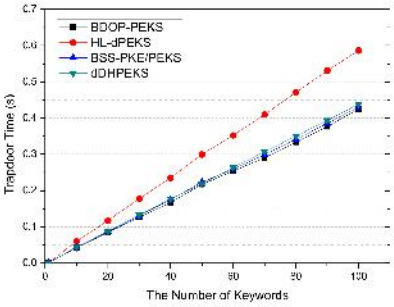


Fig. 7. Comparison of trapdoor algorithms.

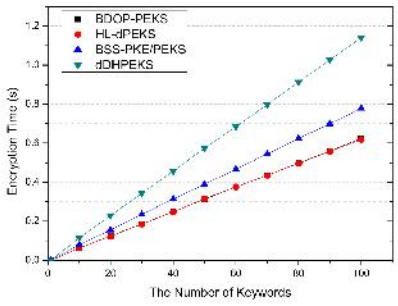


Fig. 6. Comparison of encryption algorithms.

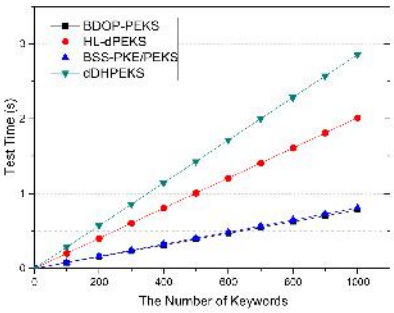


Fig. 8. Comparison of test algorithms.

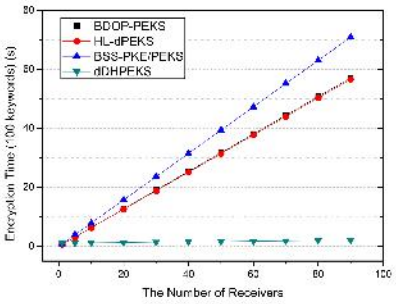


Fig. 9. Encryption time along with receiver number increasing.

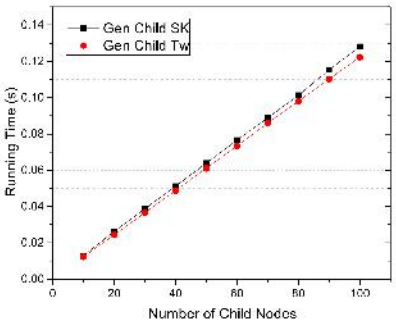


Fig. 10. Time of generating child-node's secret keys and trapdoors.

a parent node in our scheme has the privilege of accessing its child-node's encrypted data. Hence, the encryption time that the sender needs to spend in this case is constant, e.g., independent of the number of receivers. Fig. 9 shows the encryption time comparison in this scenario.

In order to search over ciphertexts for a receiver at a low level (say, level 5), a receiver at a high level (say, level 1) needs to generate secret key (and the trapdoor) for each node along the path from the high level receiver to the low level receiver, and the computational cost is thus linearly with the number of levels between the two receivers. Our implementation demonstrates that the additional running cost is much smaller than the encryption time. Fig. 10 shows that the time of generating the additional secret keys and trapdoors are both close to 0.13s in case there are 100 levels between the two receiver nodes, which is much smaller than the 55s~70s encryption time in the compared schemes. Consider that there are usually not many levels in an enterprise/organization, e.g., less than 10 levels, the cost of generating the trapdoor could thus be neglected.

7 Conclusion

In this paper, we proposed a new protocol named hierarchical public key encryption with keyword search, which supports a user to monitor its child users. We introduced a public key tree structure and used it to build a semi-generic HPEKS construction. We also proposed an advanced HPEKS scheme dDHPEKS, which integrates PEKS and PKE, and can resist outside offline keyword guessing attacks. Experiments show that our dDHPEKS scheme has comparable efficiency with existing related PEKS schemes.

Acknowledgments

This work was supported by National Natural Science Foundation of China (61872152), Guangdong Major Project of Basic and Applied Basic Research (No. 2019B030302008), Guangdong Natural Science Funds for Distinguished Young Scholar (2014A030306021), Science and Technology Program of Guangzhou (No. 201902010081), and Guangdong Program for Special Support of Top-notch Young Professionals (2015TQ01X796).

References

[1] W. Kim, "Cloud computing trends: 2018 state of the cloud survey," 2018. [Online]. Available: https://www.rightscale.com/blog/cloud-industry-insights/cloud-computing-trends-2018-state-cloud-survey
[2] B. Lynn *et al.*, "Pairing-based cryptography library," 2013. [Online]. Available: https://crypto.stanford.edu/pbc/
[3] D. X. Song, D. Wagner, and A. Perrig, "Practical techniques for searches on encrypted data," in *Proc. IEEE Symp. Security Privacy*, 2000, pp. 44–55.
[4] Y.-C. Chang and M. Mitzenmacher, "Privacy preserving keyword searches on remote encrypted data," in *Proc. Int. Conf. Appl. Cryptography Netw. Secur.*, 2005, pp. 442–455.
[5] R. Curtmola, J. Garay, S. Kamara, and R. Ostrovsky, "Searchable symmetric encryption: Improved definitions and efficient constructions," *J. Comput. Secur.*, vol. 19, no. 5, pp. 895–934, 2011.
[6] K. Kurosawa and Y. Ohtaki, "UC-secure searchable symmetric encryption," in *Proc. Int. Conf. Financial Cryptogr. Data Secur.*, 2012, pp. 285–298.
[7] D. Boneh, G. Di Crescenzo, R. Ostrovsky, and G. Persiano, *Public Key Encryption With Keyword Search*. Berlin, Germany: Springer, 2004, pp. 506–522.
[8] D. J. Park, K. Kim, and P. J. Lee, "Public key encryption with conjunctive field keyword search," in *Proc. Int. Workshop Inf. Secur. Appl.*, 2004, pp. 73–86.
[9] P. Golle, J. Staddon, and B. Waters, "Secure conjunctive keyword search over encrypted data," in *Proc. 2nd Int. Conf. Appl. Cryptogr. Netw. Secur.*, 2004, pp. 31–45.
[10] N. Cao, C. Wang, M. Li, K. Ren, and W. Lou, "Privacy-preserving multi-keyword ranked search over encrypted cloud data," *IEEE Trans. Parallel Distrib. Syst.*, vol. 25, no. 1, pp. 222–233, Jan. 2014.
[11] Z. Fu, X. Sun, Z. Xia, L. Zhou, and J. Shu, "Multi-keyword ranked search supporting synonym query over encrypted data in cloud computing," in *Proc. IEEE 32nd Int. Perform. Comput. Commun. Conf.*, 2013, pp. 1–8.
[12] H. Li, D. Liu, Y. Dai, T. H. Luan, and X. S. Shen, "Enabling efficient multi-keyword ranked search over encrypted mobile cloud data through blind storage," *IEEE Trans. Emerg. Topics Comput.*, vol. 3, no. 1, pp. 127–138, Mar. 2015.
[13] F. Bao, R. H. Deng, X. Ding, and Y. Yang, "Private query on encrypted data in multi-user settings," in *Proc. Int. Conf. Inf. Secur. Pract. Experience*, 2008, pp. 71–85.
[14] F. Zhao, T. Nishide, and K. Sakurai, "Multi-user keyword search scheme for secure data sharing with fine-grained access control," in *Proc. Int. Conf. Inf. Secur. Cryptol.*, 2011, pp. 406–418.
[15] Y. Yang, H. Lu, and J. Weng, "Multi-user private keyword search for cloud computing," in *Proc. IEEE 3rd Int. Conf. Cloud Comput. Technol. Sci.*, 2011, pp. 264–271.
[16] C. Van Rompay, R. Molva, and M. Önen, "Multi-user searchable encryption in the cloud," in *Proc. Int. Inf. Secur. Conf.*, 2015, pp. 299–316.
[17] M. Hattori *et al.*, "Ciphertext-policy delegatable hidden vector encryption and its application to searchable encryption in multi-user setting," in *Proc. IMA Int. Conf. Cryptogr. Coding*, 2011, pp. 190–209.
[18] J. W. Byun, H. S. Rhee, H.-A. Park, and D. H. Lee, "Off-line keyword guessing attacks on recent keyword search schemes over encrypted data," in *Proc. Workshop Secure Data Manage.*, 2006, pp. 75–83.

[19] W.-C. Yau, S.-H. Heng, and B.-M. Goi, "Off-line keyword guessing attacks on recent public key encryption with keyword search schemes," in *Proc. Int. Conf. Auton. Trusted Comput.*, 2008, pp. 100–105.
[20] L. Fang, W. Susilo, C. Ge, and J. Wang, "A secure channel free public key encryption with keyword search scheme without random oracle," in *Proc. Int. Conf. Cryptol. Netw. Secur.*, 2009, pp. 248–258.
[21] Q. Huang and H. Li, "An efficient public-key searchable encryption scheme secure against inside keyword guessing attacks," *Inf. Sci.*, vol. 403/404, pp. 1–14, 2017.
[22] Q. Huang and H. Li, "An efficient public-key searchable encryption scheme secure against inside keyword guessing attacks," 2018. [Online]. Available: [Online]. Available: http://eprint.iacr.org/2018/007
[23] D. He, M. Ma, S. Zeadally, N. Kumar, and K. Liang, "Certificateless public key authenticated encryption with keyword search for industrial Internet of Things," *IEEE Trans. Ind. Informat.*, vol. 14, no. 8, pp. 3618–3627, Aug. 2018.
[24] H. Li, Q. Huang, J. Shen, G. Yang, and W. Susilo, "Designated-server identity-based authenticated encryption with keyword search for encrypted Emails," *Inf. Sci.*, vol. 481, pp. 330–343, 2019.
[25] R. Chen, Y. Mu, G. Yang, F. Guo, and X. Wang, "Dual-server public-key encryption with keyword search for secure cloud storage," *IEEE. Trans. Inf. Forensics Security*, vol. 11, no. 4, pp. 789–798, Apr. 2016.
[26] R. Chen *et al.*, "Server-aided public key encryption with keyword search," *IEEE. Trans. Inf. Forensics Security*, vol. 11, no. 12, pp. 2833–2842, Dec. 2016.
[27] D. Boneh and M. Franklin, "Identity-based encryption from the weil pairing," in *Proc. Annu. Int. Cryptol. Conf.*, 2001, pp. 213–229.
[28] J. Katz and Y. Lindell, *Introduction to Modern Cryptography, 2nd ed.* Boca Raton, FL, USA: CRC Press, 2014.
[29] D. Boneh, X. Boyen, and H. Shacham, "Short group signatures," in *Proc. Annu. Int. Cryptol. Conf.*, 2004, pp. 41–55.
[30] J. Baek, R. Safavi-Naini, and W. Susilo, *On the Integration of Public Key Data Encryption and Public Key Encryption With Keyword Search*. Berlin, Germany: Springer, 2006.
[31] C. Hu and P. Liu, "A secure searchable public key encryption scheme with a designated tester against keyword guessing attacks and its extension," in *Proc. Int. Conf. Comput. Sci. Environ. Ecoinformatics Educ.*, 2011, pp. 131–136.

Hongbo Li received the BS, MS, and PhD degrees from South China Agricultural University, China, and currently is a postdoc student at College of Mathematics and Informatics, South China Agricultural University, Guangzhou, China. His research interests include applied cryptography and cloud security.

Qiong Huang received the PhD degree from the City University of Hong Kong, Hong Kong, in 2010. He is currently a professor with the College of Mathematics and Informatics, South China Agricultural University, Guangzhou, China. His research interests include cryptography and information security, in particular, cryptographic protocols design and analysis. He has published more than 100 research papers in international conferences and journals, and served as a programme committee member in many international conferences.

Willy Susilo (Senior Member, IEEE) is a senior professor with the School of Computing and Information Technology, Faculty of Engineering and Information Sciences, University of Wollongong, Australia. He is the director of Institute of Cybersecurity and Cryptology, School of Computing and Information Technology, University of Wollongong, Australia. Currently, he is the head of School of Computing and Information Technology at University of Wollongong, Australia (2015 - now). He has published more than 500 papers in journals and conference proceedings in cryptography and network security. Since 2016, he has served as the program committee member of several international conferences. He is currently an associate editor of the *IEEE Transactions on Dependable and Secure Computing*. He is the editor-in-chief of the Elsevier's *Computer Standards and Interface* and the MDPI's *Information* journal.

3 以通讯作者发表本专业论文情况

3.1 A more efficient public-key authenticated encryption scheme wit h keyword search

发表于 Journal of Systems Architecture (IF-5-year-3.6)
中科院 2 区
论文等级：A 类

Journal of Systems Architecture 137 (2023) 102839

Contents lists available at ScienceDirect

Journal of Systems Architecture

journal homepage: www.elsevier.com/locate/sysarc

A more efficient public-key authenticated encryption scheme with keyword search

Qiong Huang [a,b], Peisen Huang [a], Hongbo Li [a,*], Jianye Huang [c], Hongyuan Lin [a]

[a] College of Mathematics and Informatics, South China Agricultural University, Guangzhou 510642, China
[b] Guangzhou Key Laboratory of Intelligent Agriculture, Guangzhou 510642, China
[c] School of Computing and Information Technology, University of Wollongong, NSW 2500, Australia

ARTICLE INFO

Keywords:
Keyword search
Inside keyword guessing attacks
Low storage overhead
Inverted index
Fast search

ABSTRACT

To realize ciphertext retrieval without decryption in the public key settings, Boneh et al. in 2004 proposed a cryptographic primitive named *Public-key Encryption with Keyword Search* (PEKS) and proposed the first PEKS scheme. Following Boneh et al.'s work, various PEKS schemes were proposed; however, most schemes are vulnerable to inside keyword guessing attacks (IKGA). Huang and Li proposed a new primitive named *Public-key Authenticated Encryption with Keyword Search* (PAEKS) in 2017, which resists the IKGA in the setting of only one test server. Their main idea to resist IKGA is to require the data sender to authenticate the ciphertext while encrypting a keyword. However, in the era of big data, as the number of encrypted files increases, huge storage overhead and heavy retrieval costs become problems in PAEKS. In this paper, we proposed a new PAEKS scheme that costs low storage space and supports fast search. We introduced the ciphertext deduplication into PAEKS to reduce storage space and leveraged the inverted index to accelerate the retrieval process. We simulated the proposed scheme and several related schemes with a desktop computer. The experiment results show that our proposed scheme is of lower storage overhead and higher retrieval efficiency compared with related schemes.

1. Introduction

Nowadays, an increasing number of industries choose to store data on cloud servers. However, data security issues have also received a lot of attention from users because data leakage accidents frequently occur. Data owners worry that cloud service providers may illegally collect and sell their data. Moreover, the data in plaintext form stored in the cloud will be completely exposed if the cloud is hacked. Encrypting the data before uploading to the cloud is an effective method to deal with these problems, but it brings other issues, such as the ciphertext retrieval problem.

In 2004, Boneh et al. [1] proposed a primitive named *Public-key Encryption with Keyword Search* (PEKS), which enables retrieval of encrypted data. Following Boneh et al.'s work, a series of PEKS schemes for improvement was proposed by researchers. However, most PEKS schemes are threatened by keyword guessing attacks (KGA) because of the low entropy of the keyword space in practice. Roughly speaking, the adversary encrypts each possible keyword and tests the encrypted keyword with the given trapdoor to reveal the encrypted keyword.

The KGA can be classified into online KGA and offline KGA. In online KGA, the adversary uploads the specially crafted ciphertext of the selected keywords to the cloud and knows the receiver's queried keyword when a crafted ciphertext is tested matching. In offline KGA, the adversary does not upload crafted ciphertexts but only grabs a queried trapdoor. Via traversing keyword space, the adversary reveals the keyword of the trapdoor. The offline KGA belongs to passive attacks and is more tricky compared with the online KGA.

The offline KGA can be classified into inside keyword guessing attacks (IKGA) and outside keyword guessing attacks (OKGA), which means offline KGA is launched by internal adversaries inside of the cloud server and external adversaries outside of the cloud server, respectively. The external adversary could be a malicious user of the cloud or other hackers outside of the cloud server; The internal adversary is generally the cloud server itself, which has the privilege of testing whether the queried trapdoor matches with some ciphertexts. Most PEKS schemes are vulnerable against the IKGA because it allows any user, including the cloud server, to generate valid ciphertexts.

Li et al. [2] proposed a PEKS scheme in which the data sender shares a common session key with the data receiver to prevent the internal adversary from forging ciphertexts in terms of the sender, which is effective in resisting IKGA. Noroozi et al. [3] point out that

* Corresponding author.
E-mail address: hongbo@scau.edu.cn (H. Li).

https://doi.org/10.1016/j.sysarc.2023.102839
Received 13 September 2022; Received in revised form 8 January 2023; Accepted 11 February 2023
Available online 14 February 2023

Li et al.'s scheme is insecure against IKGA. Huang and Li [4] in 2017 proposed a new primitive named public-key authenticated encryption with keyword search (PAEKS) to counterattack the IKGA and proposed a concrete scheme (HL-PAEKS). The main idea of PAEKS is that the data sender and the data receiver generate a common key without interaction, and ciphertexts encrypted from the common key will be secure against the IKGA because it is intractable for an adversary to forage a valid ciphertext in terms of the sender without the common key. Noroozi et al. [5] pointed out that the HL-PAEKS [4] is still not secure against IKGA, and proposed a modified scheme to achieve security against IKGA. Qin et al. [6] revisited HL-PAEKS and proposed a new PAEKS scheme with higher security.

PAEKS received lots of attention due to the higher security compared with PEKS. A series of PAEKS schemes were proposed in the literature, which can be classified into the following categories: (1) Multi-keyword Security in PAEKS [5–8], (2) Simplifying the management of the data sender's public key in PAEKS [6,9], (3) Avoiding the certificate management and key escrow problems in PAEKS [10–17], (4) Quantum-resistant PAEKS [18,19].

However, existing PAEKS schemes still have some deficiencies. PAEKS schemes would suffer from the insufficient storage space problem when the number of encrypted files increases dramatically. In the era of big data, users have a great number of files that need to be stored in the cloud. Another problem is that the retrieval time will be too long to tolerate when the number of encrypted keywords is larger than some threshold value. Even though Xu et al. [20] proposed a fast search PEKS scheme, it is still vulnerable to inside keyword guessing attacks. Hence, designing a secure PAEKS scheme with lower storage overhead and higher retrieval efficiency is reasonable. It is desirable to realize almost constant storage overhead and retrieval time as the number of encrypted keywords increases.

1.1. Related works

In recent years, ciphertext retrieval in public key settings raised concerns for researchers. Boneh et al. [1] introduced the first searchable encryption scheme into the public key settings in 2004 to realize public key ciphertext retrieval. Byun et al. [21] proposed an attack named offline keyword guessing attacks and pointed out that Boneh et al.'s PEKS is vulnerable to facing this attack. To resolve this problem, security-channel-free PEKS schemes were proposed [22–24], in which the cloud server is designated as the only tester and any other party without the server's secret key does not own the ability to test whether matching or not. Rhee et al. [23] proved that trapdoor indistinguishability is equivalent to security against KGA. Yau et al. [25] pointed out that security-channel-free PEKS schemes are not secure against inside keyword guessing attacks (IKGA) launched by adversaries inside of the cloud. Li et al. [2] tried to resist IKGA by leveraging a shared common session key. Huang and Li [4] proposed a primitive of public-key authenticated encryption with keyword search (PAEKS) to resist IKGA, and implemented the first concrete PAEKS scheme. Even though the first PAEKS was proposed insecure [5], it raised the attention of researchers. Qin et al. [6] proposed multi-ciphertext indistinguishability (MCI) security and multi-trapdoor indistinguishability (MTI) model for PAEKS. Pan et al. [7] tried to achieve both MCI security and MTI security. However, it fails to achieve MCI security [26]. Li et al. [9] introduced the PAEKS into the identity-based cryptosystem. He et al. [10] proposed a certificateless PAEKS scheme. A series of improved certificateless PAEKS schemes [11–17] were proposed following He et al.'s work. To resist quantum computing attacks, post-quantum PAEKS schemes have been proposed [18,19] in recent years.

Achieving expressive functionalities in PEKS is another popular research field apart from enhancing the security of PEKS. Conjunctive, disjunctive and fuzzy PEKS schemes [27–31] were proposed to support multiple keyword search. Yang et al. [32] proposed a new primitive named public-key encryption with equality test (PKEET) to realize equality test between ciphertexts encrypted from different public keys. Xu et al. [20], leveraging a hidden-star structure, proposed a fast search PEKS scheme.

1.2. Our contributions

The main purpose of this work is to realize low storage overhead and high retrieval efficiency while resisting IKGA in PEKS. The contributions of this paper are listed as follows:

(1) We propose a new PAEKS scheme that supports to check of duplicate keywords embedded in ciphertexts. Via deduplicating ciphertexts, it costs constant storage space for ciphertexts encrypted from the same keyword. The restriction is that deduplication succeeds only if the ciphertexts are sent from the same sender.
(2) We give a method of accelerating ciphertext retrieval by combining the inverted index with ciphertexts. The accelerating retrieval method is discussed in Section 4.2 following the proposed scheme.
(3) We give a formal security proof of our proposed scheme based on the intractability of Computational Diffie–Hellman (CDH) problem in the random oracle model.
(4) We implement the proposed scheme and some related schemes in the literature using a desktop computer. The experiment results show that our proposed scheme costs almost constant storage overhead and enjoys a higher efficiency in terms of ciphertext retrieval.

1.3. Paper organization

In the next section, some necessary preliminaries are introduced briefly. In Section 3, we recall the definition of PAEKS and its security models. In Section 4, we describe the details of our proposed scheme, including the PAEKS construction, efficiency-improving method and security proof. In Section 5, we show the experimental results compared with some related schemes. The last section concludes the work of this paper and gives future work.

2. Preliminaries

2.1. Bilinear pairing

Let cyclic groups G and G_T be with the same prime order p. The map $\hat{e} : G \times G \to G_t$ is a bilinear pairing map if the following properties hold [33].

Bilinearity. $\forall g, h \in G$ and $x, y \in Z_p$, $\hat{e}(g^x, h^y) = \hat{e}(g, h)^{xy}$.

Non-degeneracy. For any generators $g_1, g_2 \in G$, it holds $\hat{e}(g_1, g_2) \neq 1_{G_T}$, where 1_{G_T} is the identity of G_t.

Computability. $\forall g, h \in G$, there is an efficient algorithm to compute $\hat{e}(g, h)$.

2.2. Computational Diffie–Hellman (CDH) assumption

Give a generator $g \in G$ and elements g^x, g^y in G, where x, y are randomly chosen from Z_p, the CDH problem is to compute g^{xy}.

Definition 1 (*CDH Assumption*). The CDH assumption is that for any probabilistic-polynomial-time (PPT) algorithm $\mathcal{A}$ given a CDH tuple (g, g^x, g^y), the following probability holds:

$$|\Pr[\mathcal{A}(g, g^x, g^y) = g^{xy}]| \leq negl(1^\lambda),$$

where $negl(\cdot)$ is a negligible function.

2.3. Inverted index

The inverted index is a key data structure for querying by keywords in retrieval systems [34]. An inverted index consists of multiple inverted lists. An inverted list L_{w_i} corresponds to a keyword w_i, which includes all the documents containing w_i.

3. Public-key authenticated encryption with keyword search

In this section, we recall the definition and security models of public-key authenticated encryption with keyword search (PAEKS).

3.1. Definition

There are five probabilistic-polynomial-time (PPT) algorithms in a PAEKS scheme:

- $Setup(1^\lambda) \rightarrow gp$: Inputting the security parameter 1^λ, it returns a global parameter gp.
- $KeyGen(gp) \rightarrow (pk_U, sk_U)$: Given gp as input, it returns a key pair(pk_U, sk_U).
- $PAEKS(w, sk_S, pk_R) \rightarrow C$: Given a sender's secret key sk_S, a receiver's public key pk_R and a keyword w as input, it returns a keyword's ciphertext C.
- $Trapdoor(w, pk_S, sk_R) \rightarrow T$: Given an input consisting of a sender's public key pk_S, a receiver's secret key sk_R and a keyword w, it returns a corresponding trapdoor T.
- $Test(C, T) \rightarrow result$: Given an input consisting of T and C, it returns **1** indicating T and C have the same keyword, otherwise, 0 indicating different keywords.

3.2. Security models

To resist IKGA, the ciphertext indistinguishability (CI) and trapdoor indistinguishability (TI) are introduced as follows.

Definition 2 (*CI-Security*). For any PPT adversary $\mathcal{A}$ in the security parameter 1^λ, a PAEKS scheme achieves **CI**-security if the advantage $ADV_{\mathcal{A}}^{CI}(1^\lambda)$ in **Game** CI is negligible.

Game CI.

1. Given a security parameter 1^λ, a global parameter gp would be generated by executing the $Setup(1^\lambda)$ algorithm. It runs $KeyGen(gp)$ to generate (pk_S, sk_S) and (pk_R, sk_R) as public/secret key pairs of a sender and a receiver, respectively. C gives (gp, pk_S, pk_R) to the adversary.
2. For adversary $\mathcal{A}$, the following two oracles can be adaptively queried polynomially many times.
   - Ciphertext Oracle $\mathcal{O}_C$: Inputting a receiver's public key pk and a keyword w, it computes the ciphertext $C \leftarrow PAEKS(w, sk_S, pk)$ and outputs C.
   - Trapdoor Oracle $\mathcal{O}_T$: Inputting a sender's public key pk and a keyword w, it computes the trapdoor $T \leftarrow Trapdoor(w, sk_R, pk)$ and outputs T.
3. At this step, $\mathcal{A}$ chooses two keywords w_0, w_1. The requirement here is that $\mathcal{A}$ should have not queried any (w_0, pk_S) or (w_1, pk_S) in $\mathcal{O}_T$, and any (w_0, pk_R) or (w_1, pk_R) in $\mathcal{O}_C$. Then $\mathcal{A}$ sends these two challenge keywords to C. The C randomly chooses a bit $b \in \{0, 1\}$, generates challenge ciphertext $C \leftarrow PAEKS(w_b, sk_S, pk_R)$, and outputs C to $\mathcal{A}$.
4. The oracles $\mathcal{O}_C$ and $\mathcal{O}_T$ can be queried by $\mathcal{A}$. The requirement here is that $\mathcal{A}$ cannot do any query on two challenge keywords as in step 3.
5. Finally, $\mathcal{A}$ outputs its guess $b' \in \{0, 1\}$. It wins the game if and only if $b = b'$.

The advantage that $\mathcal{A}$ wins **Game** CI is defined as:

$$ADV_{\mathcal{A}}^{CI}(1^\lambda) = \left| \Pr[b' = b] - \frac{1}{2} \right|.$$

Definition 3 (*TI-Security*). For any PPT adversary $\mathcal{A}$ in the security parameter 1^λ, a PAEKS scheme achieves TI-security if the advantage $ADV_{\mathcal{A}}^{TI}(\lambda)$ in **Game** TI is negligible.

Game TI.

1. Given a security parameter 1^λ, a global parameter gp would be generated by executing the $Setup(1^\lambda)$ algorithm. It runs $KeyGen(gp)$ to generate (pk_S, sk_S) and (pk_R, sk_R) as public/secret key pairs of a sender and a receiver, respectively. C gives (gp, pk_S, pk_R) to the adversary.
2. For adversary $\mathcal{A}$, the following two oracles can be adaptively queried polynomially many times.
   - Ciphertext Oracle $\mathcal{O}_C$: Inputting a receiver's public key pk and a keyword w, it computes the ciphertext $C \leftarrow PAEKS(w, sk_S, pk)$ and outputs C.
   - Trapdoor Oracle $\mathcal{O}_T$: Inputting a sender's public key pk and a keyword w, it computes the trapdoor $T \leftarrow Trapdoor(w, sk_R, pk)$ and outputs T.
3. At this step, $\mathcal{A}$ chooses two keywords w_0, w_1. The requirement here is that $\mathcal{A}$ should not have queried any (w_0, pk_S) or (w_1, pk_S) in $\mathcal{O}_T$, and any (w_0, pk_R) or (w_1, pk_R) in $\mathcal{O}_C$. Then $\mathcal{A}$ sends these two challenge keywords to C. The C randomly chooses a bit $b \in \{0, 1\}$, generates $T \leftarrow Trapdoor(w_b, pk_S, sk_R)$ as challenge trapdoor, and outputs T to $\mathcal{A}$.
4. The oracles $\mathcal{O}_C$ and $\mathcal{O}_T$ can be queried by $\mathcal{A}$. The requirement here is that $\mathcal{A}$ cannot do any query on two challenge keywords as in step 3.
5. Finally, $\mathcal{A}$ outputs a bit $b' \in \{0, 1\}$. It wins the game if and only if $b = b'$.

The advantage that $\mathcal{A}$ wins **Game** TI is defined as:

$$ADV_{\mathcal{A}}^{TI}(1^\lambda) = \left| \Pr[b' = b] - \frac{1}{2} \right|.$$

4. Our proposed PAEKS scheme

4.1. Construction

Based on the CDH assumption, we propose the following PAEKS scheme. Our scheme uses a bilinear pairing $\hat{e}$ and a collision-resistant hash function H. The algorithms of our scheme are as follows.

- $Setup(1^\lambda) \rightarrow gp$: Inputting a security parameter 1^λ, it selects a bilinear map $\hat{e} : G \times G \rightarrow G_T$ where both cyclic groups G and G_T are with the same prime order p. Then it chooses a random element $g \in G$ as a generator and sets the collision-resistant hash function $H : G \times G \times G \times \{0,1\}^* \rightarrow G$. Finally, it outputs $gp = \{p, g, G, G_T, \hat{e}, H\}$ as the global parameter.
- $KeyGen(gp) \rightarrow (pk_U, sk_U)$: Inputting the global parameter gp, it generates (pk_U, sk_U) as a public/secret key pair of user **U**. Without loss of generality, $(pk_S = g^x, sk_S = x)$ denotes the public/secret key pair of the data sender and $(pk_R = g^y, sk_R = y)$ denotes the public/secret key pair of the data receiver, where $x, y \in Z_p$ are randomly selected.
- $PAEKS(gp, w, sk_S, pk_R) \rightarrow C$: Inputting the global parameter gp, a keyword w, the secret key of a data sender sk_S and the public key of a data receiver pk_R, it chooses a random number $r \in Z_p$ and outputs ciphertext $C = (C_1, C_2)$ as the ciphertext:

$$C_1 = H(pk_S, pk_R, {pk_R}^{sk_S}, w)^r, \quad C_2 = {pk_R}^r.$$

- $Trapdoor(gp, w, pk_S, sk_R) \rightarrow T$: Inputting the global parameter gp, a keyword w, the public key of a data sender pk_S and the secret key of a data receiver sk_R, it chooses a random number $s \in Z_p$ and outputs the trapdoor $T = (T_1, T_2)$:

$$T_1 = H(pk_S, pk_R, {pk_S}^{sk_R}, w)^{\frac{s}{sk_R}}, \quad T_2 = g^s.$$

- $Test(gp, C, T) \rightarrow result$: Inputting the global parameter gp, a ciphertext C and a trapdoor T, it returns 1 1 if $\hat{e}(C_1, T_2) = \hat{e}(C_2, T_1)$, and 0, otherwise.

Correctness. Let $(pk_R, sk_R) = (g^x, x)$ and $(pk_S, sk_S) = (g^y, y)$ be the key pairs of the receiver and sender, respectively. Let w and w' be the keywords of C and T, respectively. For any valid $C = (C_1, C_2)$ and $T = (T_1, T_2)$, we have

$$\hat{e}(C_1, T_2) = \hat{e}(H(pk_S, pk_R, pk_R^{sk_S}, w)^r, g^x) = \hat{e}(H(g^y, g^x, g^{xy}, w), g)^{rx}.$$

$$\hat{e}(T_1, C_2) = \hat{e}(H(pk_S, pk_R, pk_S^{sk_R}, w')^s, g^{xr}) = \hat{e}(H(g^y, g^x, g^{xy}, w'), g)^{rx}.$$

The equation $\hat{e}(C_1, T_2) = \hat{e}(C_2, T_1)$ holds if and only if $w = w'$.

4.2. Overhead reduction

Let $C_1 = (C_{11}, C_{12})$ and $C_2 = (C_{21}, C_{22})$ be two ciphertexts sent from a sender to a receiver. Let w_1 and w_2 be keywords contained in C_1 and C_2, respectively. Once w_1 is equal to w_2, we have $\hat{e}(C_{11}, C_{22}) = \hat{e}(C_{21}, C_{12})$. Thus, the cloud server can build an inverted index and insert these two ciphertexts into the same inverted list. Taking the inverted index as a key pool, once C_1 is matched with a trapdoor, C_2 will be directly returned without the extra overhead of running the $Test$ algorithm.

Saving Storage Space.

To save storage space, the duplicate ciphertext C_2 could be deleted, and the document link pointer of C_2 could be inserted into the inverted list same as C_1. Because the document link pointer of some documents is much smaller than a ciphertext, it can deduplicate ciphertexts and significantly save storage space.

Improving Retrieval Efficiency.

Classifying all the ciphertext at leisure, the cloud server will build an inverted index for each sender. When the receiver submits a trapdoor, the server will traverse the inverted indices of all the senders. The search time will be related to the total number of inverted lists. More exactly, the average searching time will be linear with half the number of all the inverted lists. Because different ciphertexts often contain the same keyword in practice, the total number of the ciphertexts is theoretically larger than the number of inverted lists in order of magnitude. Hence, applying the inverted index can significantly improve retrieval efficiency.

Remark. Once the ciphertext has no repeated keywords, the efficiency of our scheme is identical to most related schemes. However, in practical applications, the keyword space is usually relatively small. Ciphertexts often contain the same keywords when the volume of data is large. Assuming that the keyword space size is 500, then for the same sender, the upper limit of the number of ciphertexts to be stored and the length of the inverted index is 500. Thus, no matter how large the number of ciphertexts is, in our solution, the cloud server only needs to use the $Test$ algorithm to perform at most 500 comparisons on the inverted index to retrieve all matching ciphertexts. Meanwhile, the larger the number of ciphertexts, the more advantageous the storage overhead of this scheme is compared with other schemes.

4.3. Security analysis

Theorem 1. *Our scheme satisfies the CI security under the random oracle model if the CDH assumption holds.*

Proof. Under the CI security model, if there is an adversary $\mathcal{A}$ with the ability to (t, ε) break our scheme, we can build a simulator $\mathcal{B}$ to deal with the CDH problem. Let (g, g^x, g^y) be a problem instance over the cyclic group (G, g, p). Given the instance, $\mathcal{B}$ performs the following works by controlling the random oracle and running $\mathcal{A}$.

1. Let $gp = \{p, g, G, G_T, \hat{e}\}$ be the global parameter and H be a random oracle. The simulator can control H. $\mathcal{B}$ sets $pk_R = g^x$ and $pk_S = g^y$ as the public keys of a receiver and a sender, respectively, and sends (gp, pk_S, pk_R) to $\mathcal{A}$.
2. For adversary $\mathcal{A}$, the following oracles can be adaptively queried polynomially many times.
   - Hash Oracle $\mathcal{O}_H$: $\mathcal{B}$ uses list L_H to record all queries and responses as follows. Upon receiving $\{pk_{Si}, pk_{Ri}, K_i, w_i\}$, if $\hat{e}(pk_x, pk_y) = \hat{e}(g, K_i)$, $\mathcal{B}$ gets the answer of the problem instance (g, g^x, g^y) and quits. If the tuple $\langle pk_{Si}, pk_{Ri}, K_i, w_i, h_i, a_i\rangle$ is already in L_H, $\mathcal{B}$ simply responds with $h_i = H(pk_{Si}, pk_{Ri}, K_i, w_i)$. Otherwise, $\mathcal{B}$ randomly chooses $a_i \leftarrow Z_p$, computes $h_i = pk_{Ri}{}^{a_i} = H(pk_{Si}, pk_{Ri}, K_i, w_i)$, then adds the tuple $\langle pk_{Si}, pk_{Ri}, K_i, w_i, h_i, a_i\rangle$ to L_H and outputs h_i.
   - Ciphertext Oracle $\mathcal{O}_C$: Inputting a receiver's public key pk_i and a keyword w_i, if L_H already contains the tuple $\langle pk_S, pk_i, w_i, K_i, w_i, h_i, a_i\rangle$, $\mathcal{B}$ retrieves h_i. Otherwise, $\mathcal{B}$ randomly selects $a_i \leftarrow Z_p$, computes $h_i = pk_i{}^{a_i}$ and adds $\langle pk_S, pk_i, w_i, *, h_i, a_i\rangle$ to L_H. Then, $\mathcal{B}$ randomly chooses $r_i \leftarrow Z_p$, computes

     $$C_{i1} = h_i{}^{r_i}, C_{i2} = pk_i{}^{r_i},$$

     and returns $C_i = (C_{i1}, C_{i2})$ to $\mathcal{A}$.
   - Trapdoor Oracle $\mathcal{O}_T$: Inputting a sender's public key pk_i and a keyword w_i, if L_H already contains the tuple $\langle pk_i, pk_R, w_i, K_i, w_i, h_i, a_i\rangle$, $\mathcal{B}$ retrieves h_i. Otherwise, $\mathcal{B}$ randomly selects $a_i \leftarrow Z_p$, computes $h_i = pk_R{}^{a_i}$ and adds $\langle pk_i, pk_R, w_i, *, h_i, a_i\rangle$ to L_{H1}. Then, $\mathcal{B}$ randomly chooses $s_i \leftarrow Z_p$, computes

     $$T_{i1} = h_i{}^{\frac{s_i}{sk_R}} = pk_R{}^{\frac{a_i w_i}{sk_R}} = g^{a_i s_i}, T_{i2} = g^{s_i},$$

     and returns $T_i = (T_{i1}, T_{i2})$ to $\mathcal{A}$.
3. $\mathcal{A}$ submits two keywords w_0, w_1. The restriction here is that $\mathcal{A}$ should not have queried any (w_0, pk_S) or (w_1, pk_S) in $\mathcal{O}_T$, and any (w_0, pk_R) or (w_1, pk_R) in $\mathcal{O}_C$ before. $\mathcal{B}$ randomly selects $a, r \leftarrow Z_p$ and sets the challenge ciphertext C^* as

   $$C^* = (g^{ra}, pk_R{}^r) = (g^{ra}, g^{xr}).$$

   The C^* is an encryption of the keyword $w_c \in \{w_0, w_1\}$ if $H(g^y, g^x, g^{xy}, w_c) = g^{xa}$:

   $$C^* = (g^{ra}, pk_R{}^r) = (H(g^y, g^x, g^{xy}, w_c)^r, g^{xr}).$$

   Therefore, if there is no hash query on (g^y, g^x, g^{xy}, w_c), the C^* is a correct ciphertext in the adversary's perception.
4. $\mathcal{A}$ can continue to query $\mathcal{O}_C$ and $\mathcal{O}_T$ with the same limitation as in step 3.
5. $\mathcal{A}$ outputs a guess.

These are the all steps of the simulation. It can be analyzed as follows.

Indistinguishable simulation. All random numbers in the simulation indicate randomness. These numbers are

$$x, y, h_1, h_2, \ldots, h_{q_H}, r, s.$$

Because of the random selection of these numbers, the randomness property holds. As a result, the real attack and the simulation are indistinguishable.

Advantage of breaking the challenge ciphertext. If $H(g^y, g^x, g^{xy}, w_0) = g^{xa}$, the challenge ciphertext C^* can be regarded as one of the encryption results of w_0. If $H(g^y, g^x, g^{xy}, w_1) = g^{xa}$, the C^* is an encryption of w_1. If the query (g^y, g^x, g^{xy}, w_c) is not made, $H(g^y, g^x, g^{xy}, w_c)$ is random for the adversary. Therefore, in breaking the challenge

ciphertext, it could not have any advantage when the query is not made. When the query is made, it has a full advantage.

Probability of finding solution. According to the breaking assumption, if the advantage that the adversary guesses the chosen message is ϵ, the probability that the adversary will query g^{xy} to the random oracle is ϵ. Therefore, K is equal to g^{xy} with probability ϵ.

Advantage and time cost. The simulator will deal with the CDH problem with $(t + T_s, \epsilon)$ where $T_s = O(1)$ represents the time cost of the simulation. □

Theorem 2. *Our scheme satisfies TI-security under the random model if the CDH assumption holds.*

Proof. The trapdoor structure in our scheme is similar to the ciphertext. Therefore, the TI security proof of our scheme is also similar to the CI security proof.

Under the TI security model, if there is an adversary $\mathcal{A}$ with the ability to (t, ϵ)-break our scheme, we can build a simulator $\mathcal{B}$ to deal with the CDH problem. Let (g, g^x, g^y) be a problem instance over the cyclic group (G, g, p). Given the instance, $\mathcal{B}$ performs the following works by controlling the random oracle and running $\mathcal{A}$.

1. Let $gp = \{p, g, G, G_T, \hat{e}\}$ be the global parameter and H be a random oracle. The simulator can control H. $\mathcal{B}$ sets $pk_R = g^x$ and $pk_S = g^y$ as the public keys of a receiver and a sender, respectively, and sends (gp, pk_S, pk_R) to $\mathcal{A}$.
2. For adversary $\mathcal{A}$, the following oracles can be adaptively queried polynomially many times.
   - Hash Oracle $\mathcal{O}_H$: $\mathcal{B}$ uses list L_H to record all queries and responses as follows. Upon receiving $\{pk_{Si}, pk_{Ri}, K_i, w_i\}$, if $\hat{e}(pk_x, pk_y) = \hat{e}(g, K_i)$, $\mathcal{B}$ gets the answer of the problem instance (g, g^x, g^y) and quits. If the tuple $\langle pk_{Si}, pk_{Ri}, K_i, w_i, h_i, a_i\rangle$ is already in L_H, $\mathcal{B}$ simply responds with $h_i = H(pk_{Si}, pk_{Ri}, K_i, w_i)$. Otherwise, $\mathcal{B}$ randomly chooses $a_i \leftarrow Z_p$, computes $h_i = pk_{Ri}^{a_i} = H(pk_{Si}, pk_{Ri}, K_i, w_i)$, then adds the tuple $\langle pk_{Si}, pk_{Ri}, K_i, w_i, h_i, a_i\rangle$ to L_H and outputs h_i.
   - Ciphertext Oracle $\mathcal{O}_C$: Inputting a receiver's public key pk_i and a keyword w_i chosen by $\mathcal{A}$, if L_H already contains the tuple $\langle pk_S, pk_i, w_i, K_i, w_i, h_i, a_i\rangle$, $\mathcal{B}$ retrieves h_i. Otherwise, $\mathcal{B}$ randomly selects $a_i \leftarrow Z_p$, computes $h_i = pk_i^{a_i}$ and adds $\langle pk_S, pk_i, w_i, *, h_i, a_i\rangle$ to L_H. Then, $\mathcal{B}$ randomly chooses $r_i \leftarrow Z_p$, computes

     $$C_{i1} = h_i^{r_i}, C_{i_2} = pk_i^{r_i},$$

     and returns $C_i = (C_{i1}, C_{i2})$ to $\mathcal{A}$.
   - Trapdoor Oracle $\mathcal{O}_T$: Inputting a sender's public key pk_i and a keyword w_i chosen by $\mathcal{A}$, if L_H already contains the tuple $\langle pk_i, pk_R, w_i, K_i, w_i, h_i, a_i\rangle$, $\mathcal{B}$ retrieves h_i. Otherwise, $\mathcal{B}$ randomly selects $a_i \leftarrow Z_p$, computes $h_i = pk_R^{a_i}$ and adds $\langle pk_i, pk_R, w_i, *, h_i, a_i\rangle$ to L_H. Then, $\mathcal{B}$ randomly chooses $s_i \leftarrow Z_p$, computes

     $$T_{i1} = h_i^{\frac{s_i}{pk_R}} = pk_R^{\frac{s_i a_i}{pk_R}} = g^{a_i s_i}, T_{i2} = g^{s_i},$$

     and returns $T_i = (T_{i1}, T_{i2})$ to $\mathcal{A}$.
3. $\mathcal{A}$ submits two keywords w_0, w_1. The restriction here is that $\mathcal{A}$ should not have queried any (w_0, pk_S) or (w_1, pk_S) in $\mathcal{O}_T$, and any (w_0, pk_R) or (w_1, pk_R) in $\mathcal{O}_C$ before. $\mathcal{B}$ randomly chooses $a, s \leftarrow Z_p$ and sets the challenge trapdoor T^* as

   $$T^* = (g^{ar}, g^s).$$

   The challenge trapdoor can be seen as a trapdoor of the keyword $w_c \in \{w_0, w_1\}$ if $H(g^y, g^x, g^{xy}, w_c) = g^{xa}$:

   $$T^* = (g^{ar}, g^s) = (H(g^y, g^x, g^{xy}, w_c)^{\frac{s}{pk_R}}, g^s).$$

   Therefore, if there is no hash query on $\langle g^y, g^x, g^{xy}, w_c\rangle$ to the random oracle, the challenge trapdoor is a correct trapdoor in the adversary's perception.
4. $\mathcal{A}$ can continue to query $\mathcal{O}_C$ and $\mathcal{O}_T$ with the same limitation as in step 3.
5. $\mathcal{A}$ outputs a guess.

These are the all steps of the simulation. It can be analyzed as follows.

Indistinguishable simulation. All random numbers in the simulation indicate randomness. These numbers are

$$x, y, h_1, h_2, \ldots, h_{q_H}, r, s.$$

Because of the random selection of these numbers, the randomness properly holds. As a result, the real attack and the simulation are indistinguishable.

Advantage of breaking the challenge trapdoor. If $H(g^y, g^x, g^{xy}, w_0) = g^{xa}$, the challenge trapdoor T^* is can be regarded as one of the encryption results of w_0. If $H(g^y, g^x, g^{xy}, w_1) = g^{xa}$, the T^* is an encryption of w_1. If the query (g^y, g^x, g^{xy}, w_c) is not made, $H(g^y, g^x, g^{xy}, w_c)$ is random for the adversary. Therefore, in breaking the challenge trapdoor, it could not have any advantage when the query is not made. When the query is made, it has a full advantage.

Probability of finding solution. According to the breaking assumption, if the advantage that the adversary guesses the chosen message is ϵ, the probability that the adversary will query g^{xy} to the random oracle is ϵ. Therefore, K is equal to g^{xy} with probability ϵ.

Advantage and time cost. The simulator will deal with the CDH problem with $(t + T_s, \epsilon)$ where $T_s = O(1)$ represents the time cost of the simulation. □

Table 1
Notations in the performance comparison.

Abbreviation	Description
Exp	The computing overhead of an exponentiation in G.
Exp_1	The computing overhead of an exponentiation in G_T.
H	The computing overhead of a hash-to-point operation.
H_p	The computing overhead of a hash-to-$\{0,1\}^*$ operation.
P	The computing overhead of a bilinear pairing operation.
$O(\cdot)$	Time complexity or space complexity.
nc	The number of all the received ciphertexts.
nd	The number of ciphertexts after deduplication.
$\lvert Z_p\rvert$	The length of an element in Z_p.
$\lvert G\rvert$	The length of an element in group G.
$\lvert G_1\rvert$	The length of an element in group G_1.

5. Comparison

We analyze our scheme with some related schemes in terms of computing efficiency and communication overhead. The descriptions of abbreviations used in performance comparison are listed in Table 1.

As shown in Table 2, our scheme is comparable with the compared schemes in terms of the computing overhead of each algorithm and the communicating overhead of keys, single ciphertext and single trapdoor. However, the storage and retrieval overhead of our scheme is much lower than the compared schemes as shown in Table 3. The required storage space of our scheme is reduced from $O(nc)$ to $O(nd)$ because of the ciphertext deduplication. The retrieval time of our scheme is also reduced from $O(nc)$ to $O(nd)$ with the help of the inverted index. The value (nc) means the number of all the received ciphertexts and (nd) means the number of ciphertexts after deduplication. The latter is much smaller in practice because different files often contain the same keywords. Table 4 shows the security comparison of our scheme and the compared schemes. Similar to the compared schemes, our scheme satisfies the acknowledged security to counterattack IKGA.

We implemented these schemes invoking the jPBC [35] library on a desktop computer with i7-10875H 2.3 HZ processor, 32 GB memory

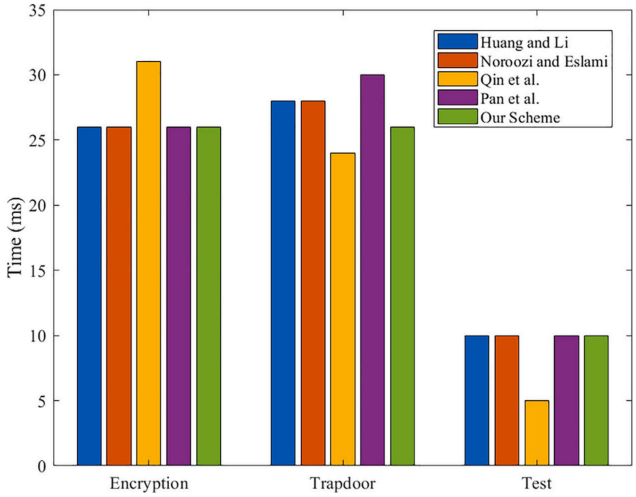


Fig. 1. Average computation cost of each algorithm.

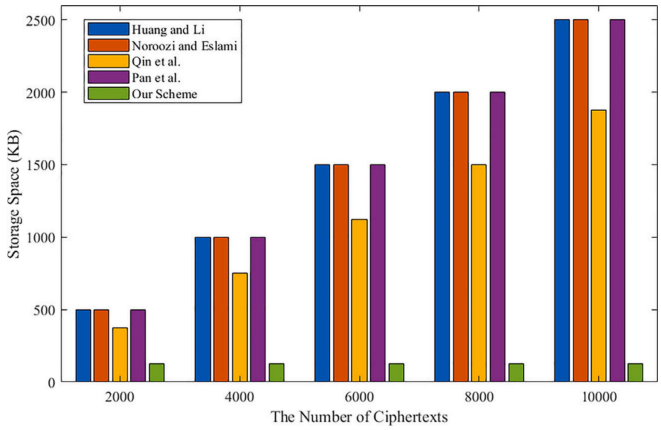


Fig. 2. Storage cost.

Table 2
Comparison analysis of computing and communicating overhead.

Scheme	Computing overhead			Communicating overhead			
	Encryption	Trapdoor	Test	$\|pk\|$	$\|sk\|$	$\|C\|$	$\|T_w\|$
[4]	3Exp+H	Exp+H+P	2P	$\|G\|$	$\|G\|$	$2\|G\|$	$\|G_T\|$
[5]	3Exp+H	Exp+H+P	2P	$\|G\|$	$\|G\|$	$2\|G\|$	$\|G_T\|$
[6]	3Exp+H+H_p+P	2Exp+H	H_p+P	$\|G\|$	$\|G\|$	$\|G\| + \|Z_p\|$	$\|G\|$
[7]	3Exp+H	2Exp+Exp_T+H+P	2P	$\|G\|$	$\|G\|$	$2\|G\|$	$2\|G\| + \|G_T\|$
Ours	3Exp+H	3Exp+H	2P	$\|G\|$	$\|G\|$	$2\|G\|$	$2\|G\|$

and Windows 10 operating system. The curve chosen in the experiment is the default Type-A curve in jPBC library [35], i.e. $y^2 = x^3 + x$ over the field F_q for prime $q = 3 \mod 4$. The order of Z_p and G is 160 and 512 bits, respectively. The size of the keyword space is set as 500 in our experiments.

The average computation cost of our scheme and the compared schemes is shown in Fig. 1 in terms of Encrypt, Trapdoor and Test algorithms. It is consistent with the theoretical analysis that our scheme is comparably efficient with the compared schemes.

Fig. 2 shows the storage cost of ciphertexts of each scheme. When the number of ciphertexts is larger than 2000, the storage cost of our scheme is much lower than the compared schemes. Furthermore, as the number of ciphertexts increases, the storage cost of our scheme is almost constant compared with the increasing linear cost of the compared schemes.

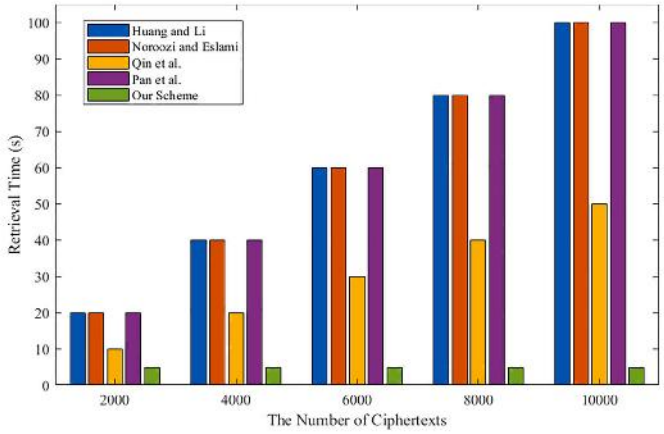


Fig. 3. Retrieval cost.

Table 3
Storage and retrieval overhead.

Scheme	Storage	Retrieval
[4]	$O(nc)$	$O(nc)$
[5]	$O(nc)$	$O(nc)$
[6]	$O(nc)$	$O(nc)$
[7]	$O(nc)$	$O(nc)$
Ours	$O(nd)$	$O(nd)$

Table 4
Security comparison.

Scheme	CI	TI	IKGA	Assumption
[4]	Yes	Yes	Yes	DBDH & mDLIN
[5]	Yes	Yes	Yes	DBDH & mDLIN
[6]	Yes	Yes	Yes	CBDH
[7]	Yes	Yes	Yes	BDHI & DLP
Ours	Yes	Yes	Yes	CDH

Fig. 3 shows the retrieval cost of each scheme. Due to the inverted index, the retrieval time cost of our scheme is almost constant compared to that of other schemes. Hence, our scheme owns a significant advantage in terms of ciphertext retrieval when the number of ciphertexts is larger than 2000.

6. Conclusion and future work

In this paper, we focused on the issues of storage and retrieval overhead of public-key authenticated encryption with keyword search (PAEKS). We proposed a new PAEKS scheme supporting ciphertexts deduplication and inverted-index-based fast search. Our scheme satisfies the CI-security and TI-security against inside keyword guessing attacks. We analyzed our scheme and some related schemes in terms of computing overhead and communicating overhead. Theoretical analysis and experiment results show that our scheme enjoys advantage in terms of storage overhead and retrieval efficiency. In future work, we consider to design efficient PAEKS schemes based on lattice to resist quantum computing attacks.

Declaration of competing interest

The authors declare the following financial interests/personal relationships which may be considered as potential competing interests:

Qiong Huang reports financial support was provided by National Natural Science Foundation of China. Qiong Huang reports financial support was provided by Major Program of Guangdong Basic and Applied Research. Qiong Huang reports financial support was provided by Science and Technology Program of Guangzhou.

Data availability

Data will be made available on request.

Acknowledgments

This work is supported by National Natural Science Foundation of China (No. 61872152, 62272174), Major Program of Guangdong Basic and Applied Research (No. 2019B030302008), and Science and Technology Program of Guangzhou, China (No. 201902010081).

References

[1] D. Boneh, G.D. Crescenzo, R. Ostrovsky, G. Persiano, Public key encryption with keyword search, in: International Conference on the Theory and Applications of Cryptographic Techniques, Springer, 2004, pp. 506–522.
[2] C.-T. Li, C. W. Lee, J.-J. Shen, An extended chaotic maps-based keyword search scheme over encrypted data resist outside and inside keyword guessing attacks in cloud storage services, Nonlinear Dynam. 80 (3) (2015) 1601–1611.
[3] M. Noroozi, Z. Eslami, N. Pakniat, Comments on a chaos-based public key encryption with keyword search scheme, Nonlinear Dynam. 94 (2) (2018) 1127–1132.
[4] Q. Huang, H. Li, An efficient public-key searchable encryption scheme secure against inside keyword guessing attacks, Inform. Sci. 403 (2017) 1–14.
[5] M. Noroozi, Z. Eslami, Public key authenticated encryption with keyword search: Revisited, IET Inf. Secur. 13 (4) (2019) 336–342.
[6] B. Qin, Y. Chen, Q. Huang, X. Liu, D. Zheng, Public-key authenticated encryption with keyword search revisited: Security model and constructions, Inform. Sci. 516 (2020) 515–528.
[7] X. Pan, F. Li, Public-key authenticated encryption with keyword search achieving both multi-ciphertext and multi-trapdoor indistinguishability, J. Syst. Archit. 115 (2021) 102075.
[8] L. Han, J. Guo, G. Yang, Q. Xie, C. Tian, An efficient and secure public key authenticated encryption with keyword search in the logarithmic time, IEEE Access 9 (2021) 151245–151253.
[9] H. Li, Q. Huang, J. Shen, G. Yang, W. Susilo, Designated-server identity-based authenticated encryption with keyword search for encrypted emails, Inform. Sci. 481 (2019) 330–343.
[10] D. He, M. Ma, S. Zeadally, N. Kumar, K. Liang, Certificateless public key authenticated encryption with keyword search for industrial Internet of Things, IEEE Trans. Ind. Inform. 14 (8) (2017) 3618–3627.

[11] L. Wu, Y. Zhang, M. Ma, N. Kumar, D. He, Certificateless searchable public key authenticated encryption with designated tester for cloud-assisted medical Internet of Things, Ann. Telecommun. 74 (7) (2019) 423–434.
[12] Y. Zhang, L. Wen, Y. Zhang, C. Wang, Designated server certificateless deniably authenticated encryption with keyword search, IEEE Access 7 (2019) 146542–146551.
[13] X. Liu, H. Li, G. Yang, W. Susilo, J. Tonien, Q. Huang, Towards enhanced security for certificateless public-key authenticated encryption with keyword search, in: International Conference on Provable Security, Springer, 2019, pp. 113–129.
[14] N. Pakniat, D. Shiraly, Z. Eslami, Certificateless authenticated encryption with keyword search: Enhanced security model and a concrete construction for industrial IoT, J. Inf. Secur. Appl. 53 (2020) 102525.
[15] B. Wu, C. Wang, H. Yao, Security analysis and secure channel-free certificateless searchable public key authenticated encryption for a cloud-based Internet of Things, PLoS One 15 (4) (2020) e0230722.
[16] T. Chi, B. Qin, D. Zheng, An efficient searchable public-key authenticated encryption for cloud-assisted medical Internet of Things, Wirel. Commun. Mob. Comput. 2020 (2020).
[17] D. Shiraly, N. Pakniat, M. Noroozi, Z. Eslami, Pairing-free certificateless authenticated encryption with keyword search, J. Syst. Archit. (2022) 102390.
[18] Z.-Y. Liu, Y.-F. Tseng, R. Tso, M. Mambo, Y.-C. Chen, Public-key authenticated encryption with keyword search: A generic construction and its quantum-resistant instantiation, Cryptol. EPrint Archive (2020).
[19] Z.-Y. Liu, Y.-F. Tseng, R. Tso, M. Mambo, Y.-C. Chen, Public-key authenticated encryption with keyword search: Cryptanalysis, enhanced security, and quantum-resistant instantiation, Cryptol. EPrint Archive (2021).
[20] P. Xu, Q. Wu, W. Wang, W. Susilo, J. Domingo-Ferrer, H. Jin, Generating searchable public-key ciphertexts with hidden structures for fast keyword search, IEEE Trans. Inf. Forensics Secur. 10 (9) (2015) 1993–2006.
[21] J.W. Byun, H.S. Rhee, H.-A. Park, D.H. Lee, Off-line keyword guessing attacks on recent keyword search schemes over encrypted data, in: Workshop on Secure Data Management, Springer, 2006, pp. 75–83.
[22] J. Baek, R. Safavi-Naini, W. Susilo, Public key encryption with keyword search revisited, in: Computational Science and Its Applications, ICCSA 2008, Springer Berlin Heidelberg, Berlin, Heidelberg, 2008, pp. 1249–1259.
[23] H.S. Rhee, W. Susilo, H.-J. Kim, Secure searchable public key encryption scheme against keyword guessing attacks, IEICE Electron. Express 6 (5) (2009) 237–243.
[24] L. Fang, W. Susilo, C. Ge, J. Wang, Public key encryption with keyword search secure against keyword guessing attacks without random oracle, Inform. Sci. 238 (2013) 221–241.
[25] W.-C. Yau, S.-H. Heng, B.-M. Goi, Off-line keyword guessing attacks on recent public key encryption with keyword search schemes, in: International Conference on Autonomic and Trusted Computing, Springer, 2008, pp. 100–105.
[26] L. Cheng, F. Meng, Security analysis of Pan et al.s public-key authenticated encryption with keyword search achieving both multi-ciphertext and multi-trapdoor indistinguishability, J. Syst. Archit. 119 (2021) 102248.
[27] P. Golle, J. Staddon, B. Waters, Secure conjunctive keyword search over encrypted data, in: International Conference on Applied Cryptography and Network Security, Springer, 2004, pp. 31–45.
[28] Z. Jiang, K. Zhang, L. Wang, J. Ning, Forward secure public-key authenticated encryption with conjunctive keyword search, Comput. J. (2022).
[29] Y. Zhang, S. Lu, POSTER: Efficient method for disjunctive and conjunctive keyword search over encrypted data, in: Proceedings of the 2014 ACM SIGSAC Conference on Computer and Communications Security, 2014, pp. 1535–1537.
[30] X. Liu, G. Yang, W. Susilo, J. Tonien, X. Liu, J. Shen, Privacy-preserving multi-keyword searchable encryption for distributed systems, IEEE Trans. Parallel Distrib. Syst. 32 (3) (2020) 561–574.
[31] M. Chuah, W. Hu, Privacy-aware bedtree based solution for fuzzy multi-keyword search over encrypted data, in: 2011 31st International Conference on Distributed Computing Systems Workshops, IEEE, 2011, pp. 273–281.
[32] G. Yang, C.H. Tan, Q. Huang, D.S. Wong, Probabilistic public key encryption with equality test, in: Cryptographers' Track at the RSA Conference, Springer, 2010, pp. 119–131.
[33] D. Boneh, M. Franklin, Identity-based encryption from the weil pairing, in: J. Kilian (Ed.), Advances in Cryptology, CRYPTO 2001, Springer Berlin Heidelberg, Berlin, Heidelberg, ISBN: 978-3-540-44647-7, 2001, pp. 213–229.
[34] D.E. Knuth, Fundamental algorithms, in: The Art of Computer Programming, Vol. 1, Addison-Wesley, 1997, pp. 261–268.
[35] A. De Caro, V. Iovino, jPBC: Java pairing based cryptography, in: 2011 IEEE Symposium on Computers and Communications, ISCC, IEEE, 2011, pp. 850–855.

3.2 Security channel free public key authenticated encryption with multi-keywod search on healthcare systems

发表于 Future Generation Computer Systems-The International Journal of Escience (IF5-year-5.9)
中科院 2 区
论文等级 A 类

Future Generation Computer Systems 145 (2023) 511–520

Contents lists available at ScienceDirect

Future Generation Computer Systems

journal homepage: www.elsevier.com/locate/fgcs

Secure channel free public key authenticated encryption with multi-keyword search on healthcare systems

Pan Yang [a], Hongbo Li [a,*], Jianye Huang [b], Hao Zhang [c], Man Ho Allen Au [d], Qiong Huang [a,e]

[a] College of Mathematics and Informatics, South China Agricultural University, Guangzhou 510642, China
[b] University of Wollongong, Wollongong, NSW, Australia
[c] The 5th Electronics Research Institute of the Ministry of Industry and Information Technology, Guangzhou 510610, China
[d] The Hong Kong Polytechnic University, Hong Kong Special Administrative Region of China
[e] Guangzhou Key Lab of Intelligent Agriculture, Guangzhou 510642, China

ARTICLE INFO

Article history:
Received 16 August 2022
Received in revised form 15 January 2023
Accepted 2 March 2023
Available online 30 March 2023

Keywords:
Public key authenticated encryption
Multi-keyword search
Secure channel free
Multi-ciphertext indistinguishability
Multi-trapdoor privacy
Keyword guessing attack

ABSTRACT

A significant amount of electronic health records (EHRs) have been kept in the cloud due to the quick development of healthcare information systems. However, storing EHRs in the cloud may raise security and data privacy issues. Because of the privacy of a patient's medical data, storing it on a cloud server requires encryption. Public-key encryption with keyword search (PEKS) solves the problem of retrieval on ciphertext, thus avoiding the problem of plaintext information leakage. However, most PEKS schemes are susceptible to inside keyword guessing attacks (KGA). Besides, transmitting the trapdoor to the cloud server requires a secure channel, which is impractical in healthcare information systems. Roughly speaking, PEKS does not protect the privacy of trapdoor and it is expensive to establish secure channels. In this paper, we propose a *secure-channel free public key authenticated encryption with multi-keyword search* (SCF-PAEMKS) scheme. The proposed SCF-PAEMKS scheme supports conjunctive keyword search, meanwhile satisfying multi-ciphertext indistinguishability (MCI) and multi-trapdoor privacy (MTP) simultaneously. The efficiency analysis and experimental results show that our SCF-PAEMKS scheme enjoys a better performance compared with related schemes.

1. Introduction

Nowadays, modern hospitals use cloud-assisted e-health systems to provide effective treatments. A large number of electronic health records (EHRs) have been stored in e-health systems. To moderate the hospital's local storage, outsourcing the e-health system to the remote cloud server is a sensible plan. Meanwhile, cloud storage has the advantage of data sharing compared with traditional local storage. Human medical research benefits from the convenient EHRs sharing of the cloud. Medical research organizations, such as MediSphere Medical Research Center [1], are now using cloud storage for research. However, cloud storage faces the problem of information leakage. Once the cloud server is breached, it will cause patient privacy disclosure. Furthermore, insiders of the cloud service provider, who have access to cloud storage data, have the opportunity to sell user data for profit. Hence, EHRs should be encrypted before being stored on the cloud server, but losing the searchability. According to recent studies, Public-key encryption with keyword search (PEKS) [2] can solve the problem of ciphertext retrieval in cloud storage.

We show the workflow of PEKS in Fig. 1. The encrypted file and any relevant keyword ciphertexts are initially uploaded to the cloud server by the data owner. When a data user wants to search for a file with the relevant keyword, the user generates a trapdoor by using the private key. Once the cloud server has verified the trapdoor the user provided, they give the corresponding encrypted file and keyword to the user. This solution allows users to search for ciphertext keywords quickly and effectively in the cloud server without revealing any data.

PEKS combined with another encryption scheme can be smoothly applied in the e-health system shown in Fig. 2. When patients have a consultation or surgery at the hospital, they can voluntarily upload their EHRs and illness symptoms to the cloud server. The doctors generate the trapdoor of selected illness symptoms using the private key, and submit it to the cloud server. Eventually, the cloud server verifies if the doctor's trapdoor correspond to the encrypted disease symptoms, and returns the matching EHRs and relevant keywords to the doctor.

Even though the symmetric searchable encryption (SSE) [3] also supports ciphertext retrieval, it does not have more advantages than PEKS in the cloud settings. In SSE, the secret key

* Corresponding author.
E-mail address: hongbo@scau.edu.cn (H. Li).

https://doi.org/10.1016/j.future.2023.03.002
0167-739X/

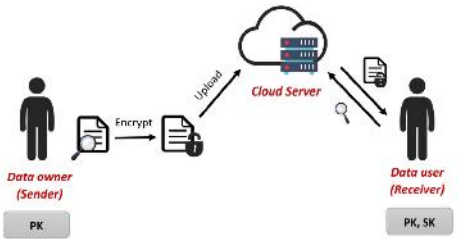


Fig. 1. The workflow of PEKS.

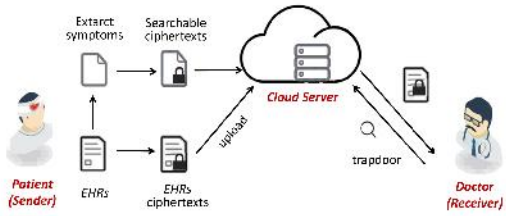


Fig. 2. EHRs system model.

for generating search-token is the same as that for encryption and the user should share their private key with other users. It leads to complex key management and could result in private key leakage. However, in PEKS, the keys for generating search-token (trapdoor) and for encryption are different and the user only needs to disclose its public key. For this reason, PEKS are better suited for ciphertext retrieval tasks in the cloud, although SSE is generally faster and there still are some drawbacks to existing PEKS schemes.

Most existing PEKS schemes cannot resist the inside keyword guessing attacks (IKGA) [4]. The IKGA is the keyword guessing attack (KGA) launched by inside adversaries who have access to the data stored in the cloud. The inside adversary can easily obtain a trapdoor submitted by a receiver. Then the adversary can encrypt all the keywords in a dictionary and test whether one of these ciphertexts is matched or not with the trapdoor. The adversary directly knows the keyword embedded in the trapdoor once the matched ciphertext exists. The IKGA attack is feasible because real-life keywords are limited and the keyword space is small enough for current computers. To address the problem, Huang and Li [5] proposed a new cryptographic primitive named public-key authenticated encryption with keyword search (PAEKS). In PAEKS, the data sender authenticates the keyword while encrypting to prevent inside adversaries from forging the valid ciphertext of the sender.

However, the majority of the PAEKS schemes in literature only support single-keyword searches, the search accuracy is inaccurate and some irrelevant results may be returned. To increase search accuracy and save bandwidth, it is necessary to build PAEKS schemes supporting multi-keyword search. Although some PEKS schemes [6–8] support multi-keyword searches, some security issues still should be a concern. For example, Zhang et al. [8] proposed a PEKS scheme supporting multi-keyword search and applied it to cloud-based smart grids. But Zhang et al.'s scheme [8] cannot resist the *selective keywords and chosen keyword attacks*. The detailed attack on Zhang et al.'s scheme [8] is described in Appendix. Besides, most multi-keyword public key searchable encryption schemes cannot resist IKGA and require deploying a secure channel to achieve the applicable security. Because of the costly establishment of a secure channel in practice, it is reasonable to consider removing the barrier of the secure channel in PEKS schemes.

Therefore, we studied the existing problems in PEKS and tried to build a secure PAEKS scheme supporting multi-keyword search without relying on a secure channel.

1.1. Related works

The notion of searchable encryption in the public key settings was proposed by Boneh et al. [9] in 2004. They named the searchable encryption in public key settings as *Public key Encryption with Keyword Search* (PEKS). In their work, they defined the security of Ciphertext Indistinguishability (CI) which ensures an adversary outside of the cloud server without knowledge of the keyword trapdoor cannot reveal any information of the keyword embedded in the trapdoor. In 2010, Rhee et al. [10] proposed a designated-server PEKS (dPKES) scheme. In their scheme, the public and private keys of the server are essential in the encryption algorithm, such that it resists keyword guessing attacks performed by an outside attacker. Li et al. [11] proposed a PEKS scheme based on chaos in which the keywords should be authenticated by the data owner during encryption. However, Noroozi et al. [12] proved Li et al.'s scheme [11] is insecure. Wang and Tu [13] introduced the concept of multi-server into public key searchable encryption for the first time in 2014. Subsequently, Chen et al. [14] proposed a dual-server public-key searchable encryption scheme, which can resist IKGA on the premise of no collusion between the dual servers. Huang and Li [5] proposed a new primitive named public-key authenticated encryption with keyword search (PAEKS), which could resist IKGA under the condition of hiring only one cloud server. Inspired by Huang et al.'s work, a series of PAEKS schemes with the property of resisting IKGA have been proposed [15–21]. In 2017, He et al. [16] present certificateless public key authenticated encryption with keyword search technique that could satisfy the security against IKGA. Later, Qin et al. [22] proposed multi-ciphertext indistinguishability (MCI) and multi-trapdoor privacy (MTP) as improved security for PAEKS. In the MCI security model, the adversary can submit two same-length sets of keywords, rather than two keywords in the CI security model, to the challenger. The adversary under the MCI security model owns more information than that under the CI security model. Hence, MCI security enjoys a higher security level than CI security. Analogously, MTP security enjoys a higher security level than TP security. Pan et al. [19] proposed a new PAEKS scheme in 2021 and declared that their scheme is the first PAEKS satisfying MTP security. However, Cheng et al. [23] proved the insecurity of Pan et al.'s scheme [19]. Qin et al. [20] came out with a new CI security model for PAEKS of multi-user. In their security model, an attacker may obtain cipher-keywords of any keyword (even a challenge keyword) of his choice. They proved that a PAEKS scheme satisfying their CI security indicates that this PAEKS scheme satisfies both MCI security and security against fully chosen keyword to cipher-keyword attacks (CKC). Shiraly et al. [21] constructed a pairing-free certificateless public key authenticated encryption with keyword search (CLAEKS) scheme and showed that it was safe using the multiple-KGC model of security.

In the study of multi-keyword PEKS, Golle et al. [6] proposed the first public-key encryption with conjunctive keyword search (PECK). Park et al. [7] improved the security model of PECK. Zhang et al. [8] proposed a public-key encryption with multi-keyword search (PEMKS) scheme supporting multi-keyword search and applied it to cloud-based smart grids, but this scheme could not resist selective keywords and chosen keyword attack. In addition,

the public-key encryption with multi-keyword search has also been studied a lot over the years [24–29].

In the study of secure-channel-free PEKS, Wang et al. [30] proposed an efficient secure channel-free public key encryption with keyword search scheme without random oracle. A concrete certificate-based searchable encryption (CBSE) system was created by Lu et al. [31], who demonstrated that it was resistant to KGA attacks and satisfied the criteria for ciphertext indistinguishability, ciphertext unforgeability, and trapdoor indistinguishability. Recently, many schemes [32–34] have been developed based on SCF-PEKS.

Remark. Achieving CI security guarantees an outside adversary without the matched trapdoor cannot obtain any information about the keyword contained in the encrypted file. A TP-secure PAEKS scheme guarantees that the inside adversary (possibly one with access to the cloud data) cannot obtain any information about the keyword from a trapdoor. From CI security to MCI security, the adversary is allowed to submit extra keywords in the challenge phase, indeed from two keywords (w_0, w_1) to two same-length sets of keywords (W_0, W_1). Intuitively, the adversary is given more information under the MCI security model than that under the CI security model. Similarly, from TP security to MTP security, the adversary owns more information. To achieve MTP security, the trapdoor algorithm must be probabilistic. Otherwise, the inside adversary can easily obtain the frequency of the keyword queried by the receiver. In the security model of CKC, the adversary can know the ciphertext of any keywords, while in the security model of MCI and CI, the adversary can only know part of the ciphertext of challenge keywords.

1.2. Our contributions

The following is a summary of the contributions of this work.

- We propose a new notion called secure-channel free public key authenticated encryption with multi-keyword search (SCF-PAEMKS). Our system eliminates the assumption of a secure channel between the server and the receiver, in addition to providing support for conjunctive keyword search. Even if the data receiver's trapdoor was recovered by the attacker, they would not be able to run the test algorithm since the attacker would not know the private key of the cloud server that was assigned.
- We formally define SCF-PAEMKS and present the security models. Then we propose a concrete SCF-PAEMKS scheme based on XDH and DLIN assumptions. We prove it to be secure against inside keyword guessing attacks (IKGA) and outside keyword guessing attacks (OKGA). It is shown that the scheme satisfies the multi-keyword ciphertext indistinguishability (MCI) and multi-trapdoor privacy (MTP).
- With comparable schemes, we contrast the proposed scheme's performance in terms of its function. The comparison shows that our framework has better properties than the comparable schemes. Moreover, the security proofs are more detailed and the scheme provides a stronger security guarantee.
- When it comes to both computing and communication efficiency, we compare our scheme to a few other similar ones. To show how effective our plan is, we also conduct several experiments. Our scheme is significantly efficient in the encryption phase and trapdoor phase and is comparable with experimental schemes in the test phase.

1.3. Organization

The remaining content of this paper is organized as follows. We offer the necessary preliminary knowledge and the difficult assumptions that must be used in Section 2. We suggest a novel SCF-PAEMKS method and security models in Section 3. The specific structure of the system is described in Section 4 along with the correctness verification. Its security proof is demonstrated in Section 5. We analyze its performance and comparison efficiency in Section 6. Finally, we conclude the paper in Section 7 and describe an attack on Zhang's scheme in Appendix.

2. Preliminaries

In this section, we review some background knowledge that contributes to the understanding of the SCF-PAEMKS construction and security analysis.

2.1. Asymmetric bilinear pairing

A prime p is the order of $\mathbb{G}_1$, $\mathbb{G}_2$, and $\mathbb{G}_T$. Let g_1, g_2 be generators of $\mathbb{G}_1$ and $\mathbb{G}_2$, respectively. a and b are integers in $\mathbb{Z}_p$. An asymmetric bilinear pairing mapping $\hat{e} : \mathbb{G}_1 \times \mathbb{G}_2 \to \mathbb{G}_T$ satisfies the following properties, where $\mathbb{G}_1 \neq \mathbb{G}_2$ and there is no computable homomorphism between $\mathbb{G}_1$ and $\mathbb{G}_2$ [35]:

- Bilinearity: $\hat{e}(g_1^a, g_2^b) = \hat{e}(g_1, g_2)^{ab}$;
- Non-degeneracy: $\hat{e}(g_1, g_2) \neq 1$;
- Computability: $\hat{e}(g_1, g_2)$ can be computed.

2.2. Hardness assumptions

We briefly introduce some hardness assumptions used in the design of our scheme as follows.

Definition 1 (*eXternal Diffie–Hellman(XDH) Problem [36]*)**.** Given the asymmetric bilinear pairing $\hat{e} : \mathbb{G}_1 \times \mathbb{G}_2 \to \mathbb{G}_T$, the XDH problem is to distinguish $(g_1, g_1^a, g_1^b, g_1^{ab})$ and (g_1, g_1^a, g_1^b, Z), where $a, b \in \mathbb{Z}_p^*$, $g_1 \in \mathbb{G}_1$, Z is randomly selected from $\mathbb{G}_1$.

Definition 2 (*XDH Assumption*)**.** For any adversary, the algorithm can correctly distinguish g_1^{ab} and Z with negligible probability ϵ_1 in the probability polynomial time, namely:

$$\Pr[\mathcal{A}(g_1, g_1^a, g_1^b, g_1^{ab}) = 1] - \Pr[\mathcal{A}(g_1, g_1^a, g_1^b, Z) = 1] \leq \epsilon_1.$$

Definition 3 (*The Decision Linear (DLIN) Problem [37]*)**.** Given $\{g, g^a, g^b, g^{ac_1}, g^{bc_2}, Z\} \in \mathbb{G}_r$, the DLIN assumption is to distinguish $g^{c_1+c_2}$ or $Z \in \mathbb{G}_r$, where $(a, b, c_1, c_2) \in \mathbb{Z}_p^*$.

Definition 4 (*DLIN Assumption*)**.** For any adversary, the algorithm can correctly distinguish $g^{c_1+c_2}$ and Z with negligible probability ϵ_2 in probability polynomial time, namely:

$$\Pr[\mathcal{A}(g, g^a, g^b, g^{ac_1}, g^{bc_2}, g^{c_1+c_2}) = 1] - \Pr[\mathcal{A}(g, g^a, g^b, g^{ac_1}, g^{bc_2}, Z) = 1] \leq \epsilon_2.$$

3. System and security models

We detail the security and system models in this section.

3.1. System model

As can be seen in Fig. 3, there is a detailed system model of SCF-PAEMKS. The patient, the doctor, and the server execute the key generation algorithm to generate their public and private keys, respectively. The patient encrypts an electronic medical record (EHR) into C_{EHR} using an encryption scheme and encrypts a corresponding keyword set $\mathbf{W}$ into $C_{\mathbf{W}}$ using the $\mathbf{SCF-PAEMKS}$ algorithm. Then the patient uploads C_{EHR} combined with $C_{\mathbf{W}}$ to the cloud server directly without a secure channel. Once the

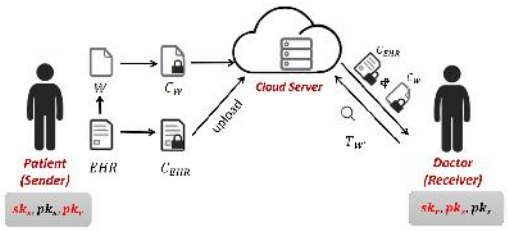


Fig. 3. System model of SCF-PAEMKS.

doctor needs to search EHRs corresponding to the keyword set $\mathbf{W}'$, s/he generates a trapdoor $T_{W'}$ using the **Trapdoor** algorithm. The cloud server runs the **Test** algorithm to search the matched ciphertexts $\{C_W\}$ and sends them with the combined $\{C_{\text{EHR}}\}$ to the doctor.

(1) **Setup**(1^λ): It requires a security parameter 1^λ as input, and the algorithm outputs the public parameters PP.

(2) **KeyGen**$_s(PP)$: It requires PP as input, and the algorithm outputs the public and private keys (pk_s, sk_s) of the sender.

(3) **KeyGen**$_r(PP)$: It requires PP as input, and the algorithm outputs the public and private keys (pk_r, sk_r) of the receiver.

(4) **KeyGen**$_v(PP)$: It requires PP as input, and the algorithm outputs the public and private keys (pk_v, sk_v) of the server.

(5) **SCF − PAEMKS**$(PP, sk_s, pk_r, \text{EHR}, \mathbf{W})$: It requires PP, the sender's secret key sk_s, the receiver's public key pk_r, the EHR and relevant keyword set $\mathbf{W}$ as input, and the algorithm outputs a searchable ciphertext of C_W.

(6) **Trapdoor**$(PP, sk_r, pk_s, pk_v, \mathbf{W}')$: It requires PP, the receiver's secret key sk_r, the sender's public key pk_s, the server's public key pk_v, the keyword set $\mathbf{W}'$ as input, and the algorithm outputs a trapdoor $T_{W'}$.

(7) **Test**$(PP, sk_v, C_W, T_{W'})$: It requires PP, the server's secret key sk_v, the ciphertext C_W, the trapdoor $T_{W'}$ as input, and the algorithm outputs 1 if $\mathbf{W}' \subseteq \mathbf{W}$, 0 otherwise.

Definition 5 (*Correctness*). For any pairs (pk_r, sk_r) of the receiver, (pk_s, sk_s) of the sender, (pk_v, sk_v) of the server, and any two sets of keywords $\mathbf{W} = (w_1, w_2, \ldots, w_{l_1})$, $\mathbf{W}' = (w_1, w_2, \ldots, w_{l_2})$, where l_2 is less than or equal to l_1. Calculating C_W and $T_{W'}$. If $\mathbf{W}' \subseteq \mathbf{W}$, we have

$$\Pr[\text{Test}(PP, sk_v, C_W, T_{W'}) = 1] = 1.$$

3.2. Threat model

In our SCF-PAEMKS scheme, we consider three types of adversaries.

When the adversary is a semi-trusted server, we consider two types of adversaries $\mathcal{A}_1$, $\mathcal{A}_2$ corresponding to two security models of multi-ciphertext indistinguishability and multi-trapdoor privacy, which capture (outside) chosen multi-keyword attacks and (inside) keyword guessing attacks, respectively. A semi-trusted server should not be able to distinguish which keyword corresponds to a given keyword ciphertext without the trapdoor from the receiver.

When the adversary is an outside adversary, we consider types of adversaries $\mathcal{A}_3$. The adversary $\mathcal{A}_3$ has access to the receiver's private key. A malicious adversary should not be able to distinguish which keyword corresponds to a target ciphertext without the server's private key even if s/he has the trapdoor of the keyword.

3.3. Security model

Our security model has three games. We define them as follows:

Game 1 (MCI): $\mathcal{A}_1$ is regarded to be a semi-trusted server.

(1) **Setup Phase.** The challenger performs algorithm Setup to generate the global public parameter PP. Next, it involves algorithms KeyGen$_s(PP)$, KeyGen$_r(PP)$ and KeyGen$_v(PP)$ respectively to generate (pk_s, sk_s), (pk_r, sk_r) and (pk_v, sk_v). Finally, $\mathcal{A}_1$ obtains (PP, pk_r, pk_s, sk_v).

(2) **Phase 1.** $\mathcal{A}_1$ can adaptively issue the following queries.

- **Ciphertext-Query $\mathcal{O}_C$.** Given $\mathbf{W}$, the challenger computes the ciphertext $C_W \leftarrow$ **SCF − PAEMKS**$(PP, sk_s, pk_r, \mathbf{W})$ and returns C_W to $\mathcal{A}_1$.
- **Trapdoor-Query $\mathcal{O}_1$.** Given $\mathbf{W}'$, the challenger computes the trapdoor $T_{W'} \leftarrow$ Trapdoor$(PP, sk_r, pk_s, pk_v, \mathbf{W}')$ and returns $T_{W'}$ to $\mathcal{A}_1$.

(3) **Challenge Phase.** $\mathcal{A}_1$ outputs two challenge keyword sets $\mathbf{W}_0^*, \mathbf{W}_1^*$ $(i.e \mid \mathbf{W}_0^* \mid = \mid \mathbf{W}_1^* \mid = l$, and $l \leq l_1)$. With the restriction that they should not be queried in phase 1. The challenger selects $\delta \in \{0, 1\}$ and performs $C_{W_\delta^*} \leftarrow$ SCF-PAEMKS $(PP, sk_s, pk_r, \mathbf{W}_\delta^*)$. The challenger generates the challenge ciphertext $C_{W_\delta^*}$ and returns it to $\mathcal{A}_1$.

(4) **Phase 2.** $\mathcal{A}_1$ can query Ciphertext-Query $\mathcal{O}_C$ and Trapdoor-Query $\mathcal{O}_T$, with the restriction that $\mathcal{A}_1$ could neither query any subset of $\mathbf{W}_0^*$ nor $\mathbf{W}_1^*$.

(5) **Guess.** $\mathcal{A}_1$ returns $\delta' \in \{0, 1\}$. $\mathcal{A}_1$ wins the game if $\delta' = \delta$. The advantage is $Adv_{\mathcal{A}_1}^{\text{MCI}}(\lambda) = |\Pr[\delta = \delta'] - \frac{1}{2}|$.

Game 2 (MTP): $\mathcal{A}_2$ is regarded to be a semi-trusted server.

(1) **Setup Phase.** Same as Game MCI.

(2) **Phase 1.** Same as Game MCI.

(3) **Challenge Phase.** $\mathcal{A}_2$ outputs two challenge keyword sets of equal length $\mathbf{W}_0^*, \mathbf{W}_1^*$ $(i.e \mid \mathbf{W}_0^* \mid = \mid \mathbf{W}_1^* \mid = l$, and $l \leq l_1)$. With the restriction that they should not be queried in phase **1**. The challenger chooses $\delta \in \{0, 1\}$, performs $T_{W_\delta^*} \leftarrow$ Trapdoor$(PP, sk_r, pk_s, pk_v, \mathbf{W}_\delta^*)$, to generate challenge trapdoor $T_{W_\delta^*}$ and return it to $\mathcal{A}_2$.

(4) **Phase 2.** $\mathcal{A}_2$ can query Ciphertext-Query $\mathcal{O}_C$ and Trapdoor-Query $\mathcal{O}_T$, except that $\mathcal{A}_2$ could neither query any subset of $\mathbf{W}_0^*$ nor $\mathbf{W}_1^*$.

(5) **Guess.** $\mathcal{A}_2$ returns $\delta' \in \{0, 1\}$. $\mathcal{A}_2$ wins the game if $\delta' = \delta$. The advantage is $Adv_{\mathcal{A}_2}^{\text{MTP}}(\lambda) = |\Pr[\delta = \delta'] - \frac{1}{2}|$.

Game 3: $\mathcal{A}_3$ is regarded to be a malicious user.

In this scheme, the key of the receiver is a random number in the integer group selected by the receiver, so the adversary can also randomly select the private key to generate a trapdoor. Assuming an outside adversary intercepts the trapdoor, it replaces it with a trapdoor it generated and sends it to the server for testing. Second, the server testing algorithm does not bind the public keys of the sender and receiver, so as long as the trapdoor is given to the designated server for testing, outside adversaries can also know the results. There is a certain probability that the server matches, and then the outside adversary succeeds. Thus, $\mathcal{A}_3$ can be simulated as a malicious outside adversary.

(1) **Setup Phase.** The challenger performs algorithm Setup to generate the global public parameter PP. Next, it involves algorithms KeyGen$_s(PP)$, KeyGen$_r(PP)$ and KeyGen$_v(PP)$ respectively to generate (pk_s, sk_s), (pk_r, sk_r) and (pk_v, sk_v). Then, $\mathcal{A}_3$ obtains (PP, pk_r, pk_v, sk_r).

(2) **Phase 1.** $\mathcal{A}_3$ can adaptively issue the following query.

- **Ciphertext-Query $\mathcal{O}_C$.** Given $\mathbf{W}$, the ciphertext $C_W \leftarrow$ SCF-PAEMKS$(PP, sk_s, pk_r, \mathbf{W})$ is returned to $\mathcal{A}_3$.

(3) **Challenge Phase**. $\mathcal{A}_3$ outputs two challenge keyword sets $\mathbf{W}_0^*, \mathbf{W}_1^*$ ($i.e$ $|\mathbf{W}_0^*| = |\mathbf{W}_1^*| = l$, and $l \leq l_1$). With the restriction that they should not be queried in phase 1. The challenger chooses $\delta \in \{0, 1\}$, performs $C_{W_\delta^*} \leftarrow$ SCF-PAEMKS(PP, sk_s, pk_r, $\mathbf{W}_\delta^*$), to generate the challenge ciphertexts $C_{W_\delta^*}$ and return it to $\mathcal{A}_3$.

(4) **Phase 2**. $\mathcal{A}_3$ can query Ciphertext-Query $\mathcal{O}_C$, except that $\mathcal{A}_3$ could neither query any subset of $\mathbf{W}_0^*$ nor $\mathbf{W}_1^*$.

(5) **Guess**. $\mathcal{A}_3$ returns $\delta' \in \{0, 1\}$. $\mathcal{A}_3$ wins the game if $\delta' = \delta$. The advantage is $Adv_{\mathcal{A}_3}^{ext}(\lambda) = |\Pr[\delta = \delta'] - \frac{1}{2}|$.

Remark. Our security model has a major difference from other PEKS schemes. In our model, the adversary is given access to not only the trapdoor oracle, but also the ciphertext oracle. In our scheme, the trapdoor is public, but the trapdoor is encrypted under the public key of the sender and the public key of the designated server, so only the designated server can use the trapdoor and its private key to search for encrypted data. The security problem caused by key transmission in secure channel is avoided.

4. Our construction

In this section, we propose our SCF-PAEMKS scheme that is secure against both chosen multi-keyword attacks and keyword guessing attacks simultaneously. We construct the keywords ciphertexts using the roots and coefficients of the polynomial. The scheme is described as follows.

4.1. SCF-PAEMKS

Our scheme is based on the XDH ($\mathbb{G}_1 \neq \mathbb{G}_2$) and DLIN assumption. Our SCF-PAEMKS scheme works as follows.

- **Setup** (1^λ). p is the prime order of $\mathbb{G}_1$, $\mathbb{G}_2$, and $\mathbb{G}_T$ respectively. $g_1 \in \mathbb{G}_1$, $g_2 \in \mathbb{G}_2$. The mapping $\hat{e} : \mathbb{G}_1 \times \mathbb{G}_2 \to \mathbb{G}_T$ is an asymmetric bilinear pairing. Choose two hash functions $h : \mathbb{G}_T \to \mathbb{Z}_p^*$, $H : \{0, 1\}^* \times \{0, 1\}^l \to \mathbb{Z}_p^*$, where l is denotes the binary length of a hash value. Return parameter $PP = (\mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T, p, g_1, g_2, \hat{e}, h, H)$.
- **KeyGen**$_s$(PP). Randomly choose $\alpha_1 \leftarrow \mathbb{Z}_p^*$ and output $sk_s = \alpha_1$, $pk_s = g_1^{\alpha_1}$.
- **KeyGen**$_r$(PP). Randomly choose $\alpha_2 \leftarrow \mathbb{Z}_p^*$ and output $sk_r = \alpha_2$, $pk_r = g_1^{\alpha_2}$.
- **KeyGen**$_u$(PP). Randomly choose $y_1, y_2 \leftarrow \mathbb{Z}_p^*$ and set $sk_u = (y_1, y_2)$, $pk_u = (Y_1, Y_2) = (g_2^{y_1}, g_2^{y_2})$. Output (pk_u, sk_u).
- **SCF − PAEMKS**(PP, pk_r, sk_s, EHR, $\mathbf{W}$). The data owner encrypts the EHR utilizing another encryption scheme, such as AES encryption, which is not the main focus of this work. Then the data owner combines the encrypted EHR with the searchable ciphertexts described as follows, and uploads them together to the cloud.

  1. The data owner generates $l_1 + 1$ degree polynomial with keywords set $\mathbf{W}$.

  $$r(x) = \prod_{i=1}^{l_1}(x - H(w_i \,||\, pk_r^{sk_s}))(x - k) + 1 = \sum_{i=0}^{l_1+1} r_i x^i,$$

  The $l_1 + 1$ roots of the equation $r(x) = 1$ are $H(w_1 \,||\, pk_r^{sk_s})$, $H(w_2 \,||\, pk_r^{sk_s})$, $\ldots$, $H(w_{l_1} \,||\, pk_r^{sk_s})$ and k.

  2. Chooses a random nonce at random $\rho \in \mathbb{Z}_p^*$, then computes the two sections of the ciphertexts:

  $$I = g_1^{\rho}, C_i = g_1^{\rho r_i} \text{ for } i = 0, \ldots, l_1 + 1.$$

  The ciphertexts are $C_W = \{C_{EHR}, I, C_i (i = 0, \ldots, l_1 + 1)\}$.

  Remark. Once $W = \emptyset$ ($l_1 = |W| = 0$), the encrypted ciphertext does not contain any keyword and it cannot be retrieved. The application program stops running.

- **Trapdoor**(PP, pk_s, pk_u, sk_r, $\mathbf{W}'$). The user wants to search a file with the keyword set $\mathbf{W}' = \{w_1, w_2, \ldots, w_{l_2}\}$, it selects two random numbers $\beta_1, \beta_2 \in \mathbb{Z}_p^*$, computes

  $$D_1 = Y_1^{\beta_1}, D_2 = Y_2^{\beta_2},$$

  $$T_i = g_2^{\frac{1}{2}(\beta_1+\beta_2)\sum_{j=1}^{l_2} H(w_j || pk_s^{sk_r})^j} \text{ for } i = 0, \ldots, l_1 + 1.$$

  The trapdoors are $T_{W'} = \{D_1, D_2, T_i (i = 0, \ldots, l_1 + 1)\}$.
- **Test**(PP, sk_u, C_W, $T_{W'}$). The cloud server receives the trapdoors $T_{W'} = \{D_1, D_2, T_i (i = 0, \ldots, l_1 + 1)\}$, and performs the following algorithm with its own private key sk_u to match the correct ciphertexts $C_W = \{I, C_i (i = 0, \ldots, l_1 + 1)\}$. Test the equation:

$$\prod_{i=0}^{l_1+1} \hat{e}(C_i, T_i) = \hat{e}(I, D_1^{\frac{1}{y_1}} D_2^{\frac{1}{y_2}}).$$

4.2. Correctness

Get $(pk_r, sk_r) = (g_1^{\alpha_2}, \alpha_2)$ and the $(pk_s, sk_s) = (g_1^{\alpha_1}, \alpha_1)$. So we have :

$$\begin{aligned}
\prod_{i=0}^{l_1+1} \hat{e}(C_i, T_i) &= \prod_{i=0}^{l_1+1} \hat{e}\left(g_1^{\rho r_i}, g_2^{\frac{1}{2}(\beta_1+\beta_2)\sum_{j=1}^{l_2} H(w_j || pk_s^{sk_r})^j}\right) \\
&= \hat{e}(g_1, g_2)^{\frac{1}{2}\rho(\beta_1+\beta_2)\sum_{i=0}^{l_1+1} r_i \sum_{j=1}^{l_2} H(w_j || pk_s^{sk_r})^j} \\
&= \hat{e}(g_1, g_2)^{\frac{1}{2}\rho(\beta_1+\beta_2)\sum_{j=1}^{l_2}\sum_{i=0}^{l_1+1} r_i H(w_j || pk_s^{sk_r})^j} \\
&= \hat{e}(g_1, g_2)^{\frac{1}{2}\rho(\beta_1+\beta_2)\sum_{j=1}^{l_2} r(H(w_j || pk_s^{sk_r}))} \\
&= \hat{e}(g_1, g_2)^{\frac{1}{2}\rho(\beta_1+\beta_2)\sum_{j=1}^{l_2} r(H(w_j || pk_r^{sk_s}))} \\
&= \hat{e}(g_1, g_2)^{\rho(\beta_1+\beta_2)} \\
&= \hat{e}(I, D_1^{\frac{1}{y_1}} D_2^{\frac{1}{y_2}}).
\end{aligned}$$

When $\mathbf{W}' \subseteq \mathbf{W}$, then $H(w_i \,||\, pk_s^{sk_r}) = H(w_i' \,||\, pk_r^{sk_s})$. We have $\prod_{i=0}^{l_1+1} \hat{e}(C_i, T_i) = \hat{e}(I, D_1^{\frac{1}{y_1}} D_2^{\frac{1}{y_2}})$, the relevant encrypted file could be obtained by the user.

5. Security proof

Our SCF-PAEMKS scheme satisfies the security against KGA in the random oracle model. Applying XDH assumption and DLIN assumption to verify MCI-security and MTP-security respectively.

To prove the theorem, we should show that the advantage of any polynomial-time adversary playing three games is negligible. In this section, we describe the security proof of the SCF-PAEMKS scheme. The proofs are described as follows.

Lemma 1. *For any adversary $\mathcal{A}_1$ with polynomial time, $Adv_{\mathcal{A}_1}^{MCI}(\lambda)$ is negligible.*

Proof. Given a public parameter group $(\hat{e}, p, g_1, g_2, \mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T)$ as the input and an instance of XDH problem $(g_1, g_1^a, g_1^b, Z) \in \mathbb{G}_1$. Note that $a, b \in \mathbb{Z}_p^*$, $Z = g_1^{ab}$ or $Z \in \mathbb{G}_1$. Algorithm $\mathcal{B}$ simulates the challenger of the game.

- **Setup Phase**. B sets $pk_s = g_1^{\alpha_1}$, $pk_r = g_1^{\alpha_2}$, $pk_v = (Y_1, Y_2) = (g_2^{y_1}, g_2^{y_2})$ and then sets $sk_s = \alpha_1$ $sk_r = \alpha_2$ and $sk_v = (y_1, y_2)$. Finally, B sends $PP = (\mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T, p, g_1, g_2, \hat{e}, H)$, pk_s, pk_r, sk_v to $\mathcal{A}_1$.
- **Phase 1**. The following oracle can be adaptively queried polynomially many times by $\mathcal{A}_1$.
  - **Ciphertext Oracle** $\mathcal{O}_C$. Given $\mathbf{W}$ is a queried set with l_1 keywords. $\mathcal{B}$ selects random number $\rho' \in \mathbb{Z}_p^*$, and computes $I' = g_1^{\rho'}$, $C_i' = g_1^{\rho' r_i'}$ for $i = 0, \ldots, l_1 + 1$. Then, it gives $C_W' = \{I', C_i'(i = 0, \ldots, l_1 + 1)\}$ to $\mathcal{A}_1$.
  - **Trapdoor Oracle** $\mathcal{O}_T$. Given $\mathbf{W}'$ is a queried set with l_2 keywords. $\mathcal{B}$ selects random number $\beta_1', \beta_2' \in \mathbb{Z}_p^*$, then
    $$D_1' = Y_1^{\beta_1'}, D_2' = Y_2^{\beta_2'}, T_i' = g_2^{\frac{1}{l_2}(\beta_1' + \beta_2') \sum_{j=1}^{l_2} H(w_j \| pk_s^{sk_r} j)}.$$
    $\mathcal{B}$ returns $T_{W'} = \{D_1', D_2', T_i'(i = 0, \ldots, l_1 + 1)\}$.
- **Challenge**. $\mathcal{A}_1$ outputs $(\mathbf{W}_0^*, \mathbf{W}_1^*)$. With the restriction that they have not been queried in phase 1. $\mathcal{B}$ chooses $\delta \in \{0, 1\}$, and computes the challenge ciphertext. For the roots$(i = 1, \ldots, l_1)$ coefficients of $r(x)$ are $\prod_{i=1}^{l_1} r_{(i)}$, the coefficient with random root k is $r_{(1)}$, then $I_\delta^* = g_1^{\rho'} = g_1^a$, $C_{\delta i}^* = g_1^{\rho' r_i'} = (g_1^\rho)^{\| \prod_{i=1}^{l_1} r_{(i)}} \cdot (g_1^{\rho k})^{r_{(1)}} = (g_1^a)^{\| \prod_{i=1}^{l_1} r_{(i)}} \cdot (g_1^{ab})^{r_{(1)}}$. In this equation, $\prod_{i=1}^{l_1} r_{(i)}$ and $r_{(1)}$ can be computable. So when $\rho = a, b = k$, g_1^a, g_1^b, g_1^{ab} can be simulated, and the oracle can answer correctly. For example, suppose that the adversary $\mathcal{A}_1$ query $\mathbf{W}_\delta^* = \{m, n\}$, and the $(m \| pk_r^{sk_s}) = M$, $(n \| pk_r^{sk_s}) = N$, $\mathcal{B}$ constructs a polynomial over x of degree three:

$$\begin{aligned}
&(x - M)(x - N)(x - k) - 1 \\
&= (x^2 - (M + N)x + MN)(x - k) + 1 \\
&= x^3 - (M + N + k)x^2 + (MN + Mk + Nk)x - MNk + 1.
\end{aligned}$$

It can be seen from the last formula: $r_0 = 1 - MNk$, $r_1 = MN + Mk + Nk$, $r_2 = -M - N - k$, $r_3 = 1$. From this it can be deduced:

$$\begin{aligned}
I^* &= g_1^\rho, \\
C_0^* &= g_1^{\rho(1 - MNk)} = g_1^\rho \cdot (g_1^{\rho k})^{-MN}, \\
C_1^* &= g_1^{\rho(MN - Mk + Nk)} = (g_1^\rho)^{MN} \cdot (g_1^{\rho k})^{M + N}, \\
C_2^* &= g_1^{-\rho(M + N + k)} = (g_1^\rho)^{-M - N} \cdot (g_1^{\rho k})^{-1}, \\
C_3^* &= g_1^\rho.
\end{aligned}$$

When $\rho = a, k = b$, it can be seen that the coefficients of the three keywords asked by $\mathcal{A}_1$ are linearly related to the root k. Thus, r_i is computable, g_1^a, g_1^b, g_1^{ab} can be simulated, and the oracle can correctly answer the adversary's query:

$$\begin{aligned}
I^* &= g_1^a, \\
C_0^* &= g_1^a \cdot (g_1^{ab})^{-MN}, \\
C_1^* &= (g_1^a)^{MN} \cdot (g_1^{ab})^{M - N}, \\
C_2^* &= (g_1^a)^{-M - N} \cdot (g_1^{ab})^{-1}, \\
C_3^* &= g_1^a.
\end{aligned}$$

Note that if $Z = g_1^{ab}$, the adversary $\mathcal{A}_1$ cannot distinguish between Z and g_1^{ab}. From $\mathcal{A}_1$'s vision, $C_{W_\delta^*} = \{I_\delta^*, C_{\delta i}^*(i = 0, \ldots, l_1 + 1)\}$ is the correct ciphertext. Thus $\mathcal{B}$ breaks the XDH problem on $\mathbb{G}_1$. Then $\mathcal{B}$ gives $C_{W_\delta^*} = \{I_\delta^*, C_{\delta i}^*(i = 0, \ldots, l_1 + 1)\}$ to $\mathcal{A}_1$.

- **Phase 2**. Same as in phase 1, except that $\mathcal{A}_1$ could neither query any subset of $\mathbf{W}_0^*$ nor $\mathbf{W}_1^*$.
- **Guess**: Eventually, $\mathcal{A}_1$ outputs δ'. If $\delta' = \delta$, $\mathcal{A}_1$ wins the game.

From the perspective of the adversary $\mathcal{A}_1$, the $C_{W_\delta^*}$ is a standard ciphertext if the Z for the XDH problem is true. It indicates that the advantage of $\mathcal{A}_1$ is greater than a non-negligible value ϵ_1. $\mathcal{A}_1$ might not receive any information about the random bit δ if Z is evenly distributed in $\mathbb{G}_T$. As a result, $\mathcal{A}_1$ could only guess at random. We have that

$$\begin{aligned}
&Adv_{\mathcal{A}_1}^{MCI}(\lambda) = \left| \Pr[\mathcal{A}_1(g_1, g_1^a, g_1^b, g_1^{ab}) = 1] - \right. \\
&\left. \Pr[\mathcal{A}_1(g_1, g_1^a, g_1^b, Z) = 1] \right| \leq \left| \frac{1}{2} + \epsilon_1 - \frac{1}{2} \right| = \epsilon_1. \quad \square
\end{aligned}$$

Lemma 2. *For any adversary $\mathcal{A}_2$ with polynomial-time, $Adv_{\mathcal{A}_2}^{MTP}(\lambda)$ is negligible.*

Proof. Given the parameter $(\hat{e}, p, g_1, g_2, \mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T)$ as the input and a instance of DLIN assumption $\{g_2, g_2^a, g_2^b, g_2^{ac_1}, g_2^{bc_2}, Z\}$. Note that $(a, b, c_1, c_2) \in \mathbb{Z}_p^*$, and $T = g_2^{c_1 + c_2}$ or $T = Z \in \mathbb{G}_2$.

- **Setup Phase**. B sets $pk_s = g_1^{\alpha_1}$, $pk_r = g_1^{\alpha_2}$, $pk_v = (Y_1, Y_2) = (g_2^a, g_2^b)$ and then sets $sk_s = \alpha_1$ $sk_r = \alpha_2$ and $sk_v = (y_1, y_2) = (a, b)$. Finally, B sends $PP = (\mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T, p, g_1, g_2, \hat{e}, H)$, pk_s, pk_r, sk_v to $\mathcal{A}_2$.
- **Phase 1**. $\mathcal{A}_2$ can adaptively issue the following queries.
  - **Ciphertext-Query** $\mathcal{O}_C$. The same as the Game MCI.
  - **Trapdoor Oracle** $\mathcal{O}_T$. The same as the Game MCI.
- **Challenge**. $\mathcal{A}_2$ outputs $(\mathbf{W}_0^*, \mathbf{W}_1^*)$ with the restriction that they have not been queried as in phase 1. $\mathcal{B}$ chooses $\delta \in \{0, 1\}$ and generates the trapdoor

  $$D_{1W_\delta^*} = Y_1^{\beta_1} = g_2^{a\beta_1} = g_2^{ac_1}, D_{2W_\delta^*} = g_2^{b\beta_2} = g_2^{bc_2},$$

  $$T_{iW_\delta^*} = g_2^{\frac{1}{l_2}(\beta_1 + \beta_2) \sum_{j=1}^{l_2} H(w_j \| pk_s^{sk_r} j)} = g_2^{\frac{1}{l_2}(c_1 + c_2) \sum_{j=1}^{l_2} H(w_j \| pk_s^{sk_r} j)}$$

  for $i = 0, \ldots, l_1 + 1$. Note that if $Z = g_2^{c_1 + c_2}$, explain that the adversary $\mathcal{A}_2$ cannot distinguish between

  $$g_2^{\frac{1}{l_2}(\beta_1 + \beta_2) \sum_{j=1}^{l_2} H(w_j \| pk_s^{sk_r} j)} \text{ and } Z^{\frac{1}{l_2} \sum_{j=1}^{l_2} H(w_j \| pk_s^{sk_r} j)}.$$

  Hence $\mathcal{B}$ breaks the DLIN problem. $\mathcal{B}$ gives $T_{W_\delta^*} = \{D_{1W_\delta^*}, D_{2W_\delta^*}, T_{iW_\delta^*}(i = 0, \ldots, l_1 + 1)\}$ to $\mathcal{A}_2$.
- **Phase 2**. Same as in phase 1, except that $\mathcal{A}_2$ could neither query any subset of $\mathbf{W}_0^*$ nor $\mathbf{W}_1^*$.
- **Guess**: $\mathcal{A}_2$ outputs $\delta' \in \{0, 1\}$. if $\delta' = \delta$, $\mathcal{A}_2$ wins the game.

From the perspective of the adversary $\mathcal{A}_2$, the $T_{W_\delta^*}$ is a typical trapdoor if the Z is true for the DLIN assumption. $\mathcal{A}_2$ might not receive any information on the random bit δ if Z is evenly distributed in $\mathbb{G}_2$. Therefore, $\mathcal{A}_2$ could only guess at random. We have that

$$\begin{aligned}
&Adv_{\mathcal{A}_2}^{MTP}(\lambda) = \left| \Pr[\mathcal{A}_2(g_2, g_2^a, g_2^b, g_2^{ac_1}, g_2^{bc_2}, g_2^{c_1 + c_2}) = 1] - \right. \\
&\left. \Pr[\mathcal{A}_2(g_2, g_2^a, g_2^b, g_2^{ac_1}, g_2^{bc_2}, Z) = 1] \right| \leq \left| \frac{1}{2} + \epsilon_2 - \frac{1}{2} \right| = \epsilon_2. \quad \square
\end{aligned}$$

Lemma 3. *For any adversary $\mathcal{A}_3$ with polynomialtime, $Adv_{\mathcal{A}_3}^{ext}(\lambda)$ is negligible.*

Proof. Given the parameter $(\hat{e}, p, g_1, g_2, \mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T)$ as the input and an instance of XDH problem $(g_1, g_1^a, g_1^b, Z) \in \mathbb{G}_1$, the XDH problem is on group $\mathbb{G}_1$. Note that $a, b \in \mathbb{Z}_p^*$, $Z = g_1^{ab}$ or $Z \in \mathbb{G}_1$.

Table 1
Framework comparisons.

Scheme	KGA[a]	MCI[b]	MTP[c]	Conjunctive[d]	SCF[e]
BDOP-PEKS [9]	No	No	No	No	No
HL-PAEKS [5]	Yes	No	No	No	No
Qin-PAEKS [22]	Yes	Yes	No	No	No
Qin-IPAEKS [20]	Yes	Yes	No	No	No
SCF-PECKS [38]	Yes	No	No	Yes	Yes
SCF-PEKS [39]	Yes	No	No	No	Yes
Miao-VCSE [41]	Yes	No	No	Yes	Yes
SPE-SMKS [42]	Yes	No	No	Yes	No
Wu-PEMKS [40]	Yes	No	No	Yes	No
PAEMKS [33]	Yes	No	No	No	Yes
PEMKS [8]	No	No	No	Yes	No
FS-PAEKS [29]	Yes	No	No	Yes	No
Our Scheme	Yes	Yes	Yes	Yes	Yes

[a]Resist keyword guessing attacks.
[b]Satisfy multi-ciphertext indistinguishability.
[c]Satisfy multi-trapdoor privacy.
[d]Support conjunctive keyword search.
[e]Secure channel free.

- **Setup**. B sets $pk_s = g_1^{\alpha_1}$, $pk_r = g_1^{\alpha_2}$, $pk_v = (Y_1, Y_2) = (g_2^{y_1}, g_2^{y_2})$ and then sets $sk_s = \alpha_1$ $sk_r = \alpha_2$ and $sk_v = (y_1, y_2)$. Finally, B sends $PP = (\mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T, p, g_1, g_2, \hat{e}, H)$, pk_s, pk_r, sk_r to $\mathcal{A}_3$.
- **Phase 1.** $\mathcal{A}_3$ can adaptively issue the following queries.
  - **Ciphertext** $\mathcal{O}_C$. The same as in Game MCI.
- **Challenge**. $\mathcal{A}_3$ outputs sets $(\mathbf{W}_0^*, \mathbf{W}_1^*)$, Which have never been queried in phase 1. The rest is the same as that in Game MCI.
- **Phase 2**. Same as in phase 1, except that $\mathcal{A}_3$ could neither query any subset of $\mathbf{W}_0^*, \mathbf{W}_1^*$.
- **Guess**: Eventually, $\mathcal{A}_3$ outputs δ' if $\delta' = \delta$, $\mathcal{A}_3$ wins the game.

If the Z is true for the XDH problem, the $C_{W_\delta^*}$ is a standard ciphertext from the view of the adversary $\mathcal{A}_3$. It indicates that the advantage of $\mathcal{A}_3$ is greater than a non-negligible value ϵ_3. We have that

$$Adv_{\mathcal{A}_3}^{ext}(\lambda) = \left| \Pr[\mathcal{A}_3(g_1, g_1^a, g_1^b, g_1^{ab}) = 1] - \Pr[\mathcal{A}_3(g_1, g_1^a, g_1^b, Z) - 1] \le \frac{1}{2} + \epsilon_3 - \frac{1}{2} \right| - \epsilon_3.$$

Game 3 is similar to Game 1 (MCI), with the restriction that the receiver's private key sk_v is given to the $\mathcal{A}_3$ during the Setup phase. Game 3 is played according to the security model, where $\mathcal{A}_3$ is only allowed to query the Ciphertext-Query Oracle $\mathcal{O}_C$ in phase 1. The challenge phase is still the same as Game 1 (MCI), where two challenge keyword sets $(\mathbf{W}_0^*, \mathbf{W}_1^*)$ are submitted. The rest of the phases are also similar and are not much explanation here. So for any polynomial-time adversary $\mathcal{A}_3$, $Adv_{\mathcal{A}_3}^{Rec}(\lambda)$ is still negligible. □

6. Comparison and experiments

The SCF-PAEMKS technique described in this work can resist keyword guessing attacks (KGA). We compared our SCF-PAEMKS framework with the related schemes (including BDOP-PEKS [9], HL-PAEKS [5], Qin-PAEKS [22], Qin-IPAEKS [20], SCF-PECKS [38], SCF-PEKS [39], Wu-PEMKS [40], Miao-VCSE [41], SPE-SMKS [42], PAEMKS [33], PEMKS [8] and FS-PAEKS [29]). Table 1 shows the properties of all compared frameworks. The comparison shows that our framework has better properties than other related schemes.

Considering that our scheme shares several similar features with some schemes. For computation and communication costs, we make a comparison with some recent schemes. Table 2 gives a comparison with schemes (CLPAEKS [16], PAEMKS [33], SCF-MPEKS [30], PEMKS [8]) in the encryption phase, the trapdoor phase, and the testing phase respectively. $|\mathbb{G}_1|$ and $|\mathbb{G}_T|$ are the bit length of an element in $\mathbb{G}_1$ and $\mathbb{G}_T$, respectively. Sum all time-consuming processes to get the algorithm's time cost. As usual, cryptographic hash costs are neglected. For example, to encrypt $W = \{w_1, w_2, \ldots, w_n\}$. The number of keywords in the file is n, the keyword encryption algorithm in our SCF-PAEMKS scheme needs to calculate $n + 2$ exponentiation in $\mathbb{G}_1$. So the time cost of the encryption algorithm is $(n + 2)e$. Due to the schemes [16,30,33] are computed from a single keyword in the trapdoor phase and test phase. For a more convenient comparison, we changed the trapdoor phases to be computed with n keywords in the same way as ours. As shown in Table 2, our scheme is more efficient in the encryption algorithm compared with other schemes [8,16,30,33]. In the trapdoor algorithm, our scheme is also similar to the scheme [8] and more efficient than the schemes [16,30,33].

Nevertheless, the scheme [8] does not satisfy the security requirements, which we have specified in Appendix. Scheme [8] is also a public key searchable encryption scheme that supports conjunctive keyword search. However, it cannot resist KGA, and its security does not satisfy MTI and MTP. Moreover, our scheme is more functional in assigning public keys and private keys to all three roles of the sender, receiver, and server, eliminating the assumption of a secure channel between server and receiver, and having authentication capabilities.

Moreover, the total number of all group elements and hash values constitute the communication cost of the ciphertext/trapdoor. For example, our ciphertexts have $n+2$ group elements in $\mathbb{G}_1$. Thus, the bit-lengths of the ciphertext are $(n+2)|\mathbb{G}_1|$ bits. For the communication cost, n is not much bigger than m, and the storage cost of $|Z_p^*|$ is small than that of $|G_1|$. Our scheme is the most space-saving in terms of ciphertext storage.

We evaluate our scheme's effectiveness and contrast it with previous analogous schemes in order to demonstrate the performance more intuitively (CLPAEKS [16], SCF-MPEKS [30], PEMKS [8], PAEMKS [33]) in Jave language. We use the Type F curves in the Pairing Based Cryptography (PBC) library [43] to evaluate the performance of our proposed system and [8], Type A in PBC library to the schemes [16,30,33] on a computer running Windows with 3.0 GHz Intel Core i5 CPU and 16 GB 3000MHz Gigabyte memory. In the setup phase, we have preprocessed some calculations $(pk_r^{sk_r}, pk_s^{sk_r}, D_1 = Y_1^{\beta_1}, D_2 = Y_2^{\beta_2})$ that will not be counted in the encryption, trapdoor, and testing phases. The number of document keywords n is from 1 to 100.

In Fig. 4, our scheme demonstrates that the time required by the ciphertext algorithm grows linearly with the number of keywords in the index. Our SCF-PAEMKS scheme presents a better computational performance on the ciphertext generation with [8,16,30,33], because our encryption does not use bilinear pairs.

Fig. 5 shows that the time of trapdoor generation changes with the number of keywords in the index, which is because T_i in the trapdoor algorithm is related to the number of document keywords t_1. The trapdoor generation phase of the schemes [16,30,33] is to search only a single keyword at a time, here we use m keywords for m runs of the trapdoor generation algorithm. From the experimental results, it can be seen that our scheme is a bit slower than the scheme [8] because our scheme has one more modal exponential operation than theirs. And other schemes are slower.

The overheads of test algorithms are shown in Fig. 6. Here we use m keywords for m runs of the [16,30,33] test generation algorithm. Even though the PEMKS [8] is faster in the test phase,

Table 2
Efficiency comparisons.

Scheme	Computation complexity			Storage cost	
	Encryption	Trapdoor	Test	Ciphertext	Trapdoor
CLPAEKS [16][a]	$3nh$ $(2 + n)e$	$(m + 2)h + mp$	$2m(h$ $p)$	$(m$ $1)\|G_1\|$	$m\|G_2\|$
PAEMKS [33][a]	$(2e + p)n$	$(e + p)m$	em	$\|G_1\| - \|G_T\|n$	$\|G_T\|m$
SCF-MPEKS [30][a]	$n(h + p) + e$	$me + mh$	$n(h + p)$	$\|G_1\|$ $n\|G_T\|$	$m\|G_1\|$
PEMKS [8][b]	$p + (n + 1)e$	$2e + mh$	$2p + ne$	$(n - 1)\|G_2\| + \|G_T\|$	$2\|G_1\| - n\|Z_p^*\|$
Our Scheme[b]	$(n + 2)e$	$3e + mh$	$(n + 3)p + 2e$	$(n - 2)\|G_1\|$	$3\|G_2\| - m\|Z_p^*\|$

[a]Symmetric bilinear pair.
[b]Asymmetric bilinear pair.
e: The modal exponential operation. h: The general hash operation. p: A bilinear pair operation. m: The number of search keywords. n: The number of keywords in files. $|G_1|$: The size of an element in group G_1. $|G_2|$: The size of an element in group G_2. $|G_T|$: The size of an element in group G_T. $|Z_p^*|$: The size of an element in group Z_p^*.

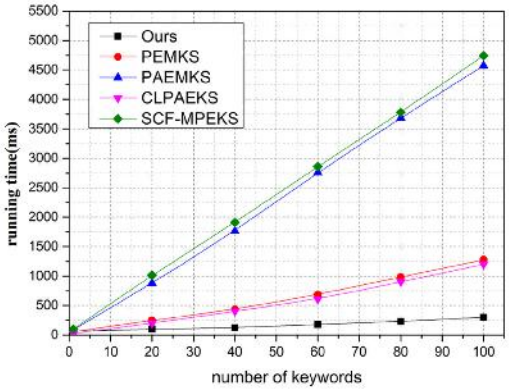


Fig. 4. Running time of encryption.

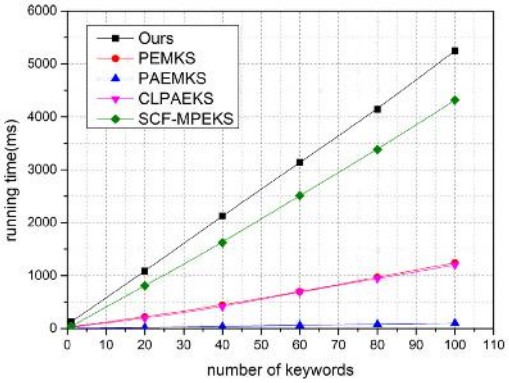


Fig. 6. Running time of test.

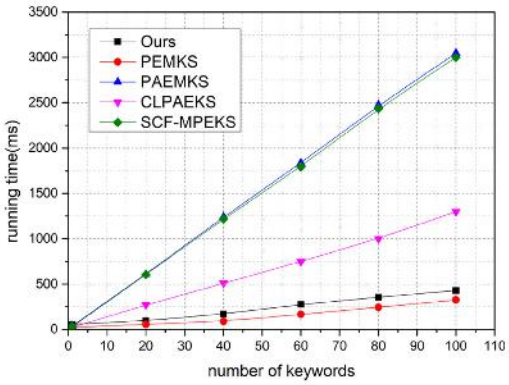


Fig. 5. Running time of trapdoor.

they are not as well as we do in terms of functionality and cannot satisfy the security requirements. The detailed attacks against [8] are shown in Appendix. The searching efficiency of our SCF-PAEMKS scheme is comparable with the schemes [16,30,33].

7. Conclusion and future work

In this paper, we proposed the notion of SCF-PAEMKS on e-health systems and formalized its security models. Moreover, we designed a concrete SCF-PAEMKS scheme and demonstrated that it satisfies the required security requirements. In addition, the proposed SCF-PAEMKS scheme supports conjunctive keyword search and removes the secure channel assumption between the server and the receiver. Moreover, it satisfies multi-ciphertext indistinguishability (MCI) and multi-trapdoor privacy (MTP) simultaneously. Meanwhile, our scheme achieves security against keyword guessing attacks (KGA). The efficiency analysis and experimental results show that our SCF-PAEMKS scheme enjoys a better performance compared with the related schemes. We leave designing a more secure and more efficient public-key authenticated searchable encryption with multi-keyword and apply disjunction keyword search to our future work.

CRediT authorship contribution statement

Pan Yang: Conceptualization, Methodology, Writing – original draft, Data curation, Software. **Hongbo Li:** Conceptualization, Methodology, Writing – review & editing, Supervision. **Jianye Huang:** Software, Visualization, Investigation. **Hao Zhang:** Validation. **Man Ho Allen Au:** Writing – review & editing. **Qiong Huang:** Supervision, Writing – review & editing.

Data availability

Available when querying.

Acknowledgments

This work is supported by National Natural Science Foundation of China (No. 61872152, 62272174), Major Program of Guangdong Basic and Applied Research (No. 2019B030302008), Guangdong Basic and Applied Basic Research Foundation (No. 2023A1515011194), and Science and Technology Program of Guang-zhou (No. 201902010081).

Appendix. Review on Zhang et al.'s scheme [8]

The construction and security proof of Zhang et al.'s scheme [8] are briefly reviewed in this section. After that, we show the security model's flaws and demonstrate an attack on their semantic security of keywords using selective keywords and chosen keyword assaults (SS-sCKA).

A.1. Zhang et al.'s PEMKS scheme

- **Setup**(1^λ). The prime p is the order of $\mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T$. g_1, g_2 are the generators in $\mathbb{G}_1$ and $\mathbb{G}_2$, respectively. $H : \{0,1\}^* \to \mathbb{Z}_q^*$, $h : \mathbb{G}_T \to \mathbb{Z}_q^*$ are cryptographic hash functions. There is a map $\hat{e} : \mathbb{G}_1 \times \mathbb{G}_2 \to \mathbb{G}_T$. Return parameter $PP = (\mathbb{G}_1, \mathbb{G}_2, q, g_1, g_2, \hat{e})$.
- **KeyGen**$(1^\epsilon, PP)$. Each user randomly selects $\alpha \leftarrow \mathbb{Z}_q^*$ and computes $g_1^\alpha, h \leftarrow g_2^\alpha$, sets $sk \leftarrow g_1^\alpha, pk \leftarrow \{g_1, g_2, h\}$. Return (pk, sk).
- **PEMKS**$(PP, \mathbf{W})$. The data owner extracts keywords set $\mathbf{W} = \{w_1, w_2, \ldots, w_{l_1}\}$.
  1. The data owner generates l_1 degree polynomial with keywords set to $\mathbf{W}$.
     $$r(x) = \sum_{i=0}^{l_1} r_i x^i$$
     The l_1 roots of the equation $r(x) = 1$ are $H(w_1)$, $H(w_2), \ldots, H(w_{l_1})$.
  2. select $\rho \in \mathbb{Z}_q^*$ and computes ciphertexts: $C_1 = \hat{e}(g_1, h)^\rho, C_2 = g_2^\rho, C_i = g_2^{\rho r_i}$ for $i = 0, \ldots, l_1$. The ciphertexts are $C_W = \{C_1, C_2, C_i (i = 0, \ldots, l_1)\}$.
- **Trapdoor**$(PP, sk, \mathbf{W}')$. An user selects $r, \beta \in \mathbb{Z}_q^*$, computes $T_1 = g_1^{r/\beta}, T_2 = g_1^\alpha g_1^{l_2 r}, T_i = \sum_{j=1}^{l_2} H(w_j)^i \beta$ for $i = 0, \ldots, l_1$. We have $\mathbf{W}' = \{w_1, w_2, \ldots, w_{l_2}\}$. The keywords trapdoor is $T_{W'} = \{T_1, T_2, T_i (i = 0, \ldots, l_1)\}$.
- **Test**$(PP, C_W, T_{W'})$. The cloud server receives the trapdoor $T_{W'}$ from the user, and the ciphertexts C_W from the data owner. Test the equation
  $$\hat{e}(\prod_{i=0}^{l_1} C_i^{T_i}, T_1) \cdot C_1 = \hat{e}(T_2, C_2)$$

A.2. Security analysis

In this section, we will elaborate on the insecurity of PEMKS [8] from three aspects: security model, ciphertext generation phase and trapdoor generation phase, respectively.

First, in the security model defined by zhang's scheme [8], the challenge keyword sets are submitted by the adversary $\mathcal{A}$ in the Setup phase, and the subsequent phase 1 $\mathcal{A}$ can query the trapdoor oracle and there is no rule that it cannot query the challenge keyword sets submitted in the Setup phase. And in phase 2 there is no restriction on the range of keyword sets queried by $\mathcal{A}$. $\mathcal{A}$ can ask for a subset of the set of two challenge keywords sets by asking in phase 1 ($w \subseteq \mathbf{W}_0^*$ or $\mathbf{W}_1^*$). Therefore, $\mathcal{A}$ can easily guess whether the challenge keywords set returned by the challenger are encrypted by $\mathbf{W}_0^*$ or $\mathbf{W}_1^*$.

Second, the ciphertexts consist of three parts in the ciphertext generation phase: $C_1 = \hat{e}(g_1, h)^\rho, C_2 = g_2^\rho, C_i = g_2^{\rho r_i}$ for $i = 0, \ldots, l_1$, where r_i is the polynomial coefficient for $r(x) = 1$. From our previous description in the main text, the adversary $\mathcal{A}$ can likewise compute r_i. The adversary submits the set of challenge keywords $\mathbf{W}_0^*$, $\mathbf{W}_1^*$ to the simulator, and the challenger selects $\mathbf{W}_0^*$ or $\mathbf{W}_1^*$ for encryption and returns the result to the adversary. However, any character can encrypt here, the adversary can also encrypt $\mathbf{W}_0^*$, $\mathbf{W}_1^*$ by himself and compare the result with the one sent by the challenger to know which keywords set the challenger encrypted. Thus the adversary can break the game.

Third, the trapdoor phase generates a trapdoor consisting of three parts: $T_1 = g_1^{r/\beta}, T_2 = g_1^\alpha g_1^{l_2 r}, T_i = \sum_{j}^{l_2} H(w_j)^i \beta$ for $i = 0, \ldots, l_1$. When $i = 0$, the adversary can calculate the value of β and the hash of the keyword by T_i. The private key of the user is g_1^α, and the private key is not used in T_i, so the adversary can calculate T_i for all keywords after calculating α. Similarly, the adversary queries the trapdoor oracle O_T for the set of keywords $\mathbf{W}_0^*$, $\mathbf{W}_1^*$, and at the same time the adversary itself can calculate T_i' of $\mathbf{W}_0^*/\mathbf{W}_1^*$, compare T_i' with the answer T_i of the O_T approximately. So the O_T has no role here because the random number β can be computed. Therefore, the trapdoor algorithm is not safe either.

We formally analyzed that [8] cannot achieve semantic security of keywords under selective keywords and chosen keyword attacks (SS-sCKA), the adversary can guess correctly and break the (SS-sCKA). Thus, Zhang et al.'s scheme [8] is not secure.

References

[1] MediSphere Medical Research Center, 2023, www.medisphereresearch.com.
[2] Z. Shen, W. Xue, J. Shu, Survey on the research and development of searchable encryption schemes, J. Softw. 25 (4) (2014) 880–895.
[3] D.X. Song, D. Wagner, A. Perrig, Practical techniques for searches on encrypted data, in: Proceeding 2000 IEEE Symposium on Security and Privacy. S&P 2000, IEEE, 2000, pp. 44–55.
[4] J.W. Byun, H.S. Rhee, H.-A. Park, D.H. Lee, Off-line keyword guessing attacks on recent keyword search schemes over encrypted data, in: Workshop on Secure Data Management, Springer, 2006, pp. 75–83.
[5] Q. Huang, H. Li, An efficient public-key searchable encryption scheme secure against inside keyword guessing attacks, Inform. Sci. 403 (2017) 1–14.
[6] P. Golle, J. Staddon, B. Waters, Secure conjunctive keyword search over encrypted data, in: International Conference on Applied Cryptography and Network Security, Springer, 2004, pp. 31–45.
[7] D.J. Park, K. Kim, P.J. Lee, Public key encryption with conjunctive field keyword search, in: International Workshop on Information Security Applications, Springer, 2004, pp. 73–86.
[8] W. Zhang, B. Qin, X. Dong, A. Tian, Public-key encryption with bidirectional keyword search and its application to encrypted emails, Comput. Stand. Interfaces 78 (2021) 103542.
[9] D. Boneh, G. Di Crescenzo, R. Ostrovsky, G. Persiano, Public key encryption with keyword search, in: International Conference on the Theory and Applications of Cryptographic Techniques, Springer, 2004, pp. 506–522.
[10] H.S. Rhee, J.H. Park, W. Susilo, D.H. Lee, Trapdoor security in a searchable public-key encryption scheme with a designated tester, J. Syst. Softw. 83 (5) (2010) 763–771.
[11] C.-T. Li, C.-W. Lee, J.-J. Shen, An extended chaotic maps-based keyword search scheme over encrypted data resist outside and inside keyword guessing attacks in cloud storage services, Nonlinear Dynam. 80 (3) (2015) 1601–1611.
[12] M. Noroozi, Z. Eslami, N. Paknia, Comments on a chaos-based public key encryption with keyword search scheme, Nonlinear Dynam. 94 (2) (2018) 1127–1132.
[13] C.-h. Wang, T.-y. Tu, Keyword search encryption scheme resistant against keyword-guessing attack by the untrusted server, J. Shanghai Jiaotong Univ. (Sci.) 19 (4) (2014) 440–442.

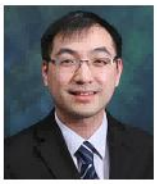
[14] R. Chen, Y. Mu, G. Yang, F. Guo, X. Wang, Dual-server public-key encryption with keyword search for secure cloud storage, IEEE Trans. Inf. Forensics Secur. 11 (4) (2015) 789–798.
[15] K. Emura, Generic construction of public-key authenticated encryption with keyword search revisited: Stronger security and efficient construction, Cryptol. ePrint Arch. (2022).
[16] D. He, M. Ma, S. Zeadally, N. Kumar, K. Liang, Certificateless public key authenticated encryption with keyword search for industrial internet of things, IEEE Trans. Ind. Inform. 14 (8) (2017) 3618–3627.
[17] Y. Lu, J. Li, Lightweight public key authenticated encryption with keyword search against adaptively-chosen-targets adversaries for mobile devices, IEEE Trans. Mob. Comput. (2021).
[18] M. Noroozi, Z. Eslami, Public key authenticated encryption with keyword search: revisited, IET Inf. Secur. 13 (4) (2019) 336–342.
[19] X. Pan, F. Li, Public-key authenticated encryption with keyword search achieving both multi-ciphertext and multi-trapdoor indistinguishability, J. Syst. Archit. 115 (2021) 102075.
[20] B. Qin, H. Cui, X. Zheng, D. Zheng, Improved security model for public-key authenticated encryption with keyword search, in: International Conference on Provable Security, Springer, 2021, pp. 19–38.
[21] D. Shiraly, N. Pakniat, M. Noroozi, Z. Eslami, Pairing-free certificateless authenticated encryption with keyword search, J. Syst. Archit. 124 (2022) 102390.
[22] B. Qin, Y. Chen, Q. Huang, X. Liu, D. Zheng, Public-key authenticated encryption with keyword search revisited: Security model and constructions, Inform. Sci. 516 (2020) 515–528.
[23] L. Cheng, F. Meng, Security analysis of Pan et al.'s "Public-key authenticated encryption with keyword search achieving both multi-ciphertext and multi-trapdoor indistinguishability", J. Syst. Archit. 119 (2021) 102248.
[24] Z. Guan, X. Liu, L. Wu, J. Wu, R. Xu, J. Zhang, Y. Li, Cross-lingual multi-keyword rank search with semantic extension over encrypted data, Inform. Sci. 514 (2020) 523–540.
[25] H. Li, Q. Huang, S. Ma, J. Shen, W. Susilo, Authorized equality test on identity-based ciphertexts for secret data sharing via cloud storage, IEEE Access 7 (2019) 25409–25421.
[26] H. Li, Q. Huang, W. Susilo, A secure cloud data sharing protocol for enterprise supporting hierarchical keyword search, IEEE Trans. Dependable Secure Comput. (2020).
[27] X. Liu, G. Yang, W. Susilo, J. Tonien, X. Liu, J. Shen, Privacy-preserving multi-keyword searchable encryption for distributed systems, IEEE Trans. Parallel Distrib. Syst. 32 (3) (2020) 561–574.
[28] H. Yin, Z. Qin, J. Zhang, L. Ou, F. Li, K. Li, Secure conjunctive multi-keyword ranked search over encrypted cloud data for multiple data owners, Future Gener. Comput. Syst. 100 (2019) 689–700.
[29] Z. Jiang, K. Zhang, L. Wang, J. Ning, Forward secure public-key authenticated encryption with conjunctive keyword search, Comput. J. (2022).
[30] T. Wang, M.H. Au, W. Wu, An efficient secure channel free searchable encryption scheme with multiple keywords, in: International Conference on Network and System Security, Springer, 2016, pp. 251–265.
[31] Y. Lu, J. Li, Y. Zhang, Secure channel free certificate-based searchable encryption withstanding outside and inside keyword guessing attacks, IEEE Trans. Serv. Comput. (2019).
[32] W. Guangbo, L. Feng, F. Liwen, L. Haicheng, An efficient SCF-PEKS without random oracle under simple assumption, Chin. J. Electron. 30 (1) (2021) 77–84.
[33] Y. Ma, H. Kazemian, A secure and efficient public key authenticated encryption with multi-keywords search scheme against inside keyword guessing attack, Int. J. Cyber-Secur. Digit. Forensics 9 (2) (2020) 90–102.
[34] J. Wang, Z. Zhao, L. Sun, Z. Zhu, Secure and efficient conjunctive keyword search scheme without secure channel, KSII Trans. Internet Inf. Syst. (TIIS) 13 (5) (2019) 2718–2731.
[35] S.D. Galbraith, K.G. Paterson, N.P. Smart, Pairings for cryptographers, Discrete Appl. Math. 156 (16) (2008) 3113–3121.
[36] L. Ballard, M. Green, B. De Medeiros, F. Monrose, Correlation-resistant storage via keyword-searchable encryption, Cryptol. ePrint Arch. (2005).
[37] D. Boneh, X. Boyen, H. Shacham, Short group signatures, in: Annual International Cryptology Conference, Springer, 2004, pp. 41–55.
[38] M.-S. Hwang, S.-T. Hsu, C.-C. Lee, A new public key encryption with conjunctive field keyword search scheme, Inf. Technol. Control 43 (3) (2014) 277–288.
[39] L. Guo, W.-C. Yau, Efficient secure-channel free public key encryption with keyword search for EMRs in cloud storage, J. Med. Syst. 39 (2) (2015) 1–11.
[40] Y. Wu, J. Hou, J. Liu, W. Zhou, S. Yao, Novel multi-keyword search on encrypted data in the cloud, IEEE Access 7 (2019) 31984–31996.
[41] Y. Miao, J. Ma, F. Wei, Z. Liu, X.A. Wang, C. Lu, VCSE: Verifiable conjunctive keywords search over encrypted data without secure-channel, Peer To Peer Netw. Appl. 10 (4) (2017) 995–1007.
[42] Y. Zhang, Y. Wang, Y. Li, Searchable public key encryption supporting semantic multi-keywords search, IEEE Access 7 (2019) 122078–122090.
[43] A. De Caro, V. Iovino, jPBC: Java pairing based cryptography, in: 2011 IEEE Symposium on Computers and Communications, ISCC, IEEE, 2011, pp. 850–855.

Pan Yang is currently pursuing the M.S. degree with the College of Mathematics and Informatics from South China Agricultural University. Her research interests include applied cryptography and security in cloud computing.

Hongbo Li received the Ph.D. degree from South China Agricultural University, Guangzhou, China. He currently holds a post-doctoral position with the College of Mathematics and Informatics, South China Agricultural University. His research interests include applied cryptography and cloud security.

Jianye Huang received the B.S. and M.S. degrees from South China Agricultural University, Guangzhou, China, in 2015 and 2018. He is currently pursuing a Ph.D. degree with the University of Wollongong, Australia. His research interests include cryptography; in particular, leakage-resilient signature, searchable encryption and k-times anonymous authentication.

Hao Zhang received his master degree from South China University of Technology. Now he works as a senior engineer at the 5th Electronics Research Institute of the Ministry of Industry and Information Technology. His main research directions and areas of concern include network security, network reliability and commercial cryptography.



Man Ho Au received the B.Eng. and M.Phil. degrees from the Department of Information Engineering, The Chinese University of Hong Kong, in 2003 and 2005, respectively, and the Ph.D. degree from the School of Computer Science and Software Engineering, University of Wollongong, in 2009. He is currently an Associate Professor with the Department of Computer Science, The University of Hong Kong (HKU). Prior to joining HKU, he was an Associate Professor with the Department of Computing, The Hong Kong Polytechnic University. He has published over 160 refereed articles in top journals and conferences, including CRYPTO, ASIACRYPT, ACM CCS, ACM SIGMOD, NDSS, IEEE TIFS, TC, and TKDE. His research interests include applied cryptography, information security, blockchain technology, and related industrial applications. He was a recipient of the 2009 PET Runner-Up Award for outstanding research in privacy enhancing technologies. His digital signature technology has been adopted by the Hyperledger Fabric Project. He is also an Expert Member of the China delegation of ISO/IEC JTC 1/SC 27 Working Group 2: Cryptography and Security Mechanisms and a Committee Member of the Research and Development Division, Hong Kong Blockchain Society.

Qiong Huang received the Ph.D. degree from the City University of Hong Kong in 2010. He is currently a Professor with the College of Mathematics and Informatics, South China Agricultural University, Guangzhou, China. He has published more than 120 research papers in international conferences and journals, and served as a program committee member in many international conferences. His research interests include cryptography and information security, in particular, cryptographic protocols design and analysis.

四、知识产权

1 发明专利：密文相似度计算方法、装置、系统及存储介质

发明人：黄琼；王元昊；李宏博；刘文博；苗莹
专利号：ZL 2021 1 0094766.0
授权公告日：2022 年 8 月 12 日

证书号第5381638号

发明专利证书

发明名称：密文相似度计算方法、装置、系统及存储介质

发明人：黄琼;王元昊;李宏博;刘文博;苗莹

专利号：ZL 2021 1 0094766.0

专利申请日：2021年01月25日

专利权人：华南农业大学

地址：510642 广东省广州市天河区五山路483号

授权公告日：2022年08月12日　　授权公告号：CN 112887089 B

国家知识产权局依照中华人民共和国专利法进行审查，决定授予专利权，颁发发明专利证书并在专利登记簿上予以登记。专利权自授权公告之日起生效。专利权期限为二十年，自申请日起算。

专利证书记载专利权登记时的法律状况。专利权的转移、质押、无效、终止、恢复和专利权人的姓名或名称、国籍、地址变更等事项记载在专利登记簿上。

局长
申长雨

国家知识产权局

2022年08月12日

第 1 页 (共 2 页)

其他事项参见续页

2 发明专利：一种多关键词搜索功能的安全无信道公钥认证可搜索加密方法及相关装置

发明人：黄琼；杨潘；李宏博；陆经纬；王庭安
专利号：ZL 2022 1 0916507.6
授权公告日：2024 年 3 月 12 日

证书号第6776449号

发明专利证书

发明名称：一种多关键词搜索功能的安全无信道公钥认证可搜索加密方法及相关装置

发　明　人：黄琼;杨潘;李宏博;陆经纬;王庭安

专　　利　　号：ZL 2022 1 0916507.6

专利申请日：2022年08月01日

专利权人：华南农业大学

地　　址：510640 广东省广州市天河区五山路483号

授权公告日：2024年03月12日　　授权公告号：CN 115333811 B

国家知识产权局依照中华人民共和国专利法进行审查，决定授予专利权，颁发发明专利证书并在专利登记簿上予以登记。专利权自授权公告之日起生效。专利权期限为二十年，自申请日起算。

专利证书记载专利权登记时的法律状况。专利权的转移、质押、无效、终止、恢复和专利权人的姓名或名称、国籍、地址变更等事项记载在专利登记簿上。

局长
申长雨

国家知识产权局
2024年03月12日

第 1 页 (共 2 页)

其他事项参见续页

3 发明专利：具有分级可扩展访问策略的属性基加密方法、装置及截止

发明人：黄琼；肖媚燕；苗莹；李宏博；陈新坚
专利号：ZL 2021 1 1523644.5
授权公告日：2024 年 7 月 12 日

证书号第7191311号

专利公告信息

发明专利证书

发 明 名 称：具有分级可扩展访问策略的属性基加密方法、装置及介质

专 利 权 人：华南农业大学

地　　址：510642 广东省广州市天河区五山路483号

发 明 人：黄琼;肖媚燕;苗莹;李宏博;陈新坚

专 利 号：ZL 2021 1 1523644.5　　授权公告号：CN 114218604 B

专利申请日：2021年12月14日　　授权公告日：2024年07月12日

申请日时申请人：华南农业大学

申请日时发明人：黄琼;肖媚燕;苗莹;李宏博;陈新坚

国家知识产权局依照中华人民共和国专利法进行审查，决定授予专利权，并予以公告。

专利权自授权公告之日起生效。专利权有效性及专利权人变更等法律信息以专利登记簿记载为准。

局长
申长雨

国家知识产权局

2024年07月12日

第 1 页 (共 1 页)

五、其他业绩

1 指导学生学科竞赛

1.1 第十六届蓝桥杯全国软件和信息技术专业人才大赛总决赛 C/C++ 程序设计大学 B 组优秀奖

蓝桥杯大赛

获奖证书

华南农业大学李宏博：

指导林业茂荣获第十六届蓝桥杯全国软件和信息技术专业人才大赛全国总决赛C/C++程序设计大学B组优秀奖，被评为优秀指导教师。

特发此证，以资鼓励。

证书编号：1602114916

证件号码：610422198012090518

工业和信息化部人才交流中心

蓝桥杯大赛组委会

2025年6月23日

1.2 第十六届蓝桥杯全国软件和信息技术专业人才大赛广东赛区 C/C++程序设计大学 B 组一等奖

蓝桥杯大赛

获奖证书

华南农业大学李宏博：

指导林业茂荣获第十六届蓝桥杯全国软件和信息技术专业人才大赛广东赛区C/C++程序设计大学B组一等奖，被评为优秀指导教师。

特发此证，以资鼓励。

证书编号：1602057438

证件号码：610422198012090518

工业和信息化部
人才交流中心

蓝桥杯大赛组委会

2025年5月26日

1.3 第十六届蓝桥杯全国软件和信息技术专业人才大赛广东赛区 C/C++程序设计大学 B 组二等奖

蓝桥杯大赛

获奖证书

华南农业大学李宏博：

指导林业茂荣获第十六届蓝桥杯全国软件和信息技术专业人才大赛广东赛区C/C++程序设计大学B组一等奖，被评为优秀指导教师。

特发此证，以资鼓励。

证书编号：1602057438

证件号码：610422198012090518

工业和信息化部人才交流中心

蓝桥杯大赛组委会

2025年5月26日

1.4 第十六届蓝桥杯全国软件和信息技术专业人才大赛广东赛区 C/C++程序设计大学 B 组三等奖

蓝桥杯大赛

获奖证书

华南农业大学李宏博：

指导林业茂荣获第十六届蓝桥杯全国软件和信息技术专业人才大赛广东赛区C/C++程序设计大学B组一等奖，被评为优秀指导教师。

特发此证，以资鼓励。

证书编号：1602057438

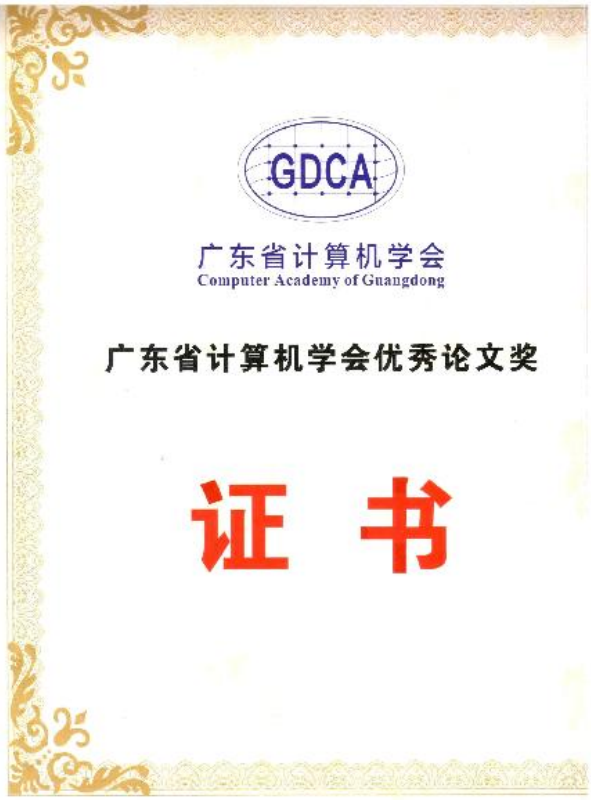
证件号码：610422198012090518

工业和信息化部人才交流中心

蓝桥杯大赛组委会

2025年5月26日

2 广东省计算机学会优秀论文奖



GDCA

广东省计算机学会
Computer Academy of Guangdong

广东省计算机学会优秀论文奖

证 书

为表彰2023年度广东省计算机学会
优秀论文奖获奖者，特颁发此证书。

项目名称：Attribute-Based Hierarchical Access Control With Extendable Policy

奖励等级：一等奖

获奖单位：华南农业大学

获 奖 者：肖媚燕 李宏博 黄琼 ShuiYu Willy Susilo

粤计学证：[2023]57号 二〇二三年十二月

项目编号：2023-J1098