面向云存储的数据加密系统与技术研究

韩培义[1,2]，刘川意[2,3]，王佳慧[4]，段少明[2]，潘鹤中[1]，方滨兴[2,3]

（1. 北京邮电大学网络空间安全学院，北京 100876；
2. 哈尔滨工业大学（深圳）计算机科学与技术学院，广东 深圳 518055；
3. 鹏城实验室网络空间安全研究中心，广东 深圳 518040；
4. 国家信息中心信息与网络安全部，北京 100045）

摘 要：针对云存储数据安全问题，提出了面向浏览器云存储应用的自动化数据加密系统。该系统采用 JavaScript 动态程序分析技术，可自动化识别与适配各类云应用，确保了对各类云应用敏感数据的加密保护，并集成了基于安全网关执行的密文搜索功能，在实现数据加密保护的同时还可最大限度地保持云应用原有功能。实验结果表明，该系统可自动化适配加密各类云应用，支持密文搜索功能，且花费较小的性能代价。

关键词：云安全；数据安全；数据加密

中图分类号：TP309.2

文献标识码：A

doi: 10.11959/j.issn.1000−436x.2020140

Research on data encryption system and technology for cloud storage

HAN Peiyi[1,2], LIU Chuanyi[2,3], WANG Jiahui[4], DUAN Shaoming[2], PAN Hezhong[1], FANG Binxing[2,3]

1. School of Cyberspace Security, Beijing University of Posts and Telecommunications, Beijing 100876, China
2. School of Computer and Technology, Harbin Institute of Technology (Shenzhen), Shenzhen 518055, China
3. Cyberspace Security Research Center, Peng Cheng Laboratory, Shenzhen 518040, China
4. Department of Information and Security, The State Information Center, Beijing 100045, China

Abstract: To order to address the problem of cloud storage data security, the generic proxy-based data protection system was proposed, which could automatically and transparently secure sensitive data in browser-based cloud storage applications. A novel dynamic program analysis technique was adopted based on JavaScript API function hooking for automatically extending to various cloud applications. And a novel proxy executed searchable encryption solution was presented so that it could achieve data encryption while maintaining the original functions of cloud applications. Experimental results show that the system can support a variety of typical cloud services, effectively protect sensitive data, and bring a relatively low overhead.

Key words: cloud security, data security, data encryption

1 引言

当前，随着云计算的飞速发展，越来越多的个人或企业选择使用云存储应用。与此同时，云存储数据安全事件频频发生。2019 年，云存储服务商 MEGA 泄露 87 GB 数据，其中含 7.7 亿个邮箱。亚

收稿日期：2020−03−20；**修回日期**：2020−06−04

通信作者：刘川意，liuchuanyi@hit.edu.cn

基金项目：广东省重点研发计划基金资助项目（No.2019B010136001）；国家重点研发计划基金资助项目（No.2018YFB1004005）；国家自然科学基金资助项目（No.61872110）

Foundation Items: The Key Research and Development Program of Guangdong (No.2019B010136001), The National Key Research and Development Program of China (No.2018YFB1004005), The National Natural Science Foundation of China (No.61872110)

马逊云存储服务曾泄露大量商业数据、军事敏感信息以及多达 150 万名公民的医疗数据[1]。针对云存储数据安全问题，一个最直接有效的方法是将用户隐私数据加密后上传至云端。这样，云服务商只能看到密文，数据控制权完全掌握在用户手中。然而，在实际应用中针对基于浏览器的云存储应用进行加密面临着如下挑战。

1) 自动化适配云服务应用

网络安全公司 Skyhigh Networks 和 Cipher Cloud[2]提出了云访问安全代理（CASB, cloud access security broker）[3]技术，该技术通过逆向分析云服务应用协议来适配各类云服务，进而对上云数据进行加密。面对众多软件即服务（SaaS, software as a service），目前的技术需要人工逐个分析云服务协议进行适配，不仅工作量巨大而且容易出错。这种方式耗时耗力，并且一旦协议发生变动，数据加密功能就会失效。因此，如何实现自动化地识别和适配云服务是一大技术挑战。

2) 数据加密的同时确保云服务功能

实现数据加密和确保云服务原有功能是一对矛盾体。如果只是将加密数据上传到云中，云平台将沦为一个仅支持数据上传和下载的数据池，无法发挥云服务对数据计算、管理和挖掘的优势。传统的端到端加密客户端如 PGP（pretty good privacy）需要用户在上传文件之前使用它来加密文件。尽管该方法简单有效，但用户需要不断切换应用和加密客户端，难以管理密钥。安全中间层技术[4]通过提供 UI（user interface）中间层用原始文件上传功能替换可加密的文件上传功能，使云应用成为“哑巴存储”，导致原有大规模存储和文档搜索功能失效。

针对上述挑战，本文提出了面向浏览器云存储应用的自动化数据加密系统——CloudCrypt。面对类型繁多的云服务应用，CloudCrypt 需要尽可能地减少工作量，同时支持适配成千上万的云应用。因此，CloudCrypt 采用 JavaScript 动态程序分析技术来监听各类云应用客户端程序的文件上传行为，识别其文件操作请求并加密文件内容。该技术可将敏感数据安全隔离，并从根本上解决云应用适配难的问题。针对数据加密影响云服务原有功能问题，CloudCrypt 集成了基于安全网关执行的密文搜索功能（PESE, proxy executed searchable encryption）。PESE 不需要云提供商配合以及对云服务进行改造，在实现数据加密保护的同时最大限度地保持云应用原有的搜索功能。

本文主要创新点总结如下。

1) 提出了面向浏览器云存储应用的自动化数据加密系统——CloudCrypt。该系统可自动化适配各类云存储应用，具有良好的扩展性。

2) CloudCrypt 系统集成的基于安全网关执行的密文搜索功能不需要对云应用进行修改，在保证敏感数据安全的同时最大限度地保持云应用原有的搜索功能。

3) 实现了 CloudCrypt 系统并将其应用于邮件、存储、办公等 10 个典型云应用，并通过实验对其性能、可扩展性等进行了详细的测试评估。实验表明，CloudCrypt 系统既可透明地加密数据，又可支持云应用原有的搜索功能，且引入性能代价较小。

2 相关工作

本节主要介绍针对面向云存储服务的数据加密系统相关工作。目前，在工业界与学术界均推出了一系列面向云存储服务的敏感数据加密系统。本节将讨论各个数据加密系统的优势与劣势，详情如表 1 所示。

1) 文件加密工具

文件加密工具是使用传统密码算法如 AES-256 对文件进行加解密。用户需要在文件上云前使用文件加密工具如 PGP 对自己的文件进行加密，确保敏感数据以加密形式存储在不可信的云存储服务商中。然而，这种方法导致云服务无法处理加密文件，丧失其重要功能如搜索、文档编辑、文档预览等。同时，用户在使用云端的文件时需要将整个文件下载并解密，还需要维护每个文件对应的密钥，由于

表 1 各个数据加密系统对比

数据加密系统	加密位置	支持透明加密	支持通用应用	密钥管理位置	实际应用
文件加密工具	客户端	×	√	用户	PGP、Encryption Dog
安全中间层	浏览器	√	×	密钥管理服务器	MessageGuard、Virtru
云访问安全代理	网关	√	×	网关	Skyhigh Networks、CipherCloud
CloudCrypt	浏览器/网关	√	√	网关	—

管理密钥繁杂，因此大大影响了用户的使用习惯。

2) 安全中间层

安全中间层[5]是一种可以给用户提供安全中间层来查看和交互敏感数据的技术。具体来说，该方法使用 UI 中间层如 iFrame、Shadow DOM 等来覆盖应用程序的原始 UI 界面，进而在中间层上实现保护数据的功能来覆盖和替换原始功能。

MessageGuard[6-7]和 Virtru[8]利用浏览器中的 iFrames 组件，实现了一个可替代原有应用文件上传功能的 UI 中间层，来保护敏感文件数据。尽管可对文件数据进行加密，却需要一个额外的文件存储服务器来存储加密数据，也丧失了原有的文件搜索功能。ShadowCrypt[9]作为一个浏览器插件，基于浏览器的 Shadow DOM 机制构建了安全隔离的文本输入/输出环境。ShadowHPE[10]在 ShadowCrypt 的基础上利用 Shadow DOM 与 DOM 树技术，提出了可支持格式保全加密且支持文本加密功能更稳定的方法。然而，ShadowCrypt 与 ShadowHPE 均只支持文本数据的加密，无法支持文件数据。M-Aegis[11]通过创建一个名为 Layer 7.5 (L-7.5)的透明 UI 中间层来截获并加解密用户输入文本数据，在不改变用户使用习惯的同时确保用户敏感信息的安全性。M-Aegis 同样只支持文本数据。

3) 云访问安全代理

云访问安全代理（CASB）[3]受到工业界的广泛追捧，连续多年被 Gartner 评为网络安全十大创新性技术之首[12]。该技术被以 Skyhigh Networks、CipherCloud 为代表的“独角兽”创新企业所采用，并推出了基于 CASB 架构的数据加密产品。CASB 位于云服务用户与云服务提供商之间，可对上云数据进行加密保护。然而，该技术需要开发者对云服务应用协议进行逆向分析和适配，耗时耗力。同时，协议一旦发生更新，CASB 将无法识别和解析最新协议，导致数据加密功能失效。在实际中，该技术需要大量的人工维护工作，且数据加密功能不稳定。

4) 其他数据加密系统

近年来，工业界也涌现出一些针对云应用数据加密系统的研究工作。CryptDB[13]是一个部署在服务器端和数据库服务器之间的代理服务器，将数据加密后存储到数据库，并对加密数据执行查询操作，可有效防范内部恶意数据库管理员。ARX[14]提出了针对 MongoDB 的数据库加密系统，可支持丰富的复杂密文计算功能。然而，CryptDB 与 ARX 均无法阻止恶意云服务器窃取用户敏感数据行为。Mylar[15]提供了一种基于 Meteor JavaScript 的 Web 框架，由云服务开发者调用该框架编程接口来实现用户数据的加密。Mylar 假设云服务商是可信的，因此难以防范云服务商。DataBlinder[16]针对软件服务开发商提供分布式数据访问中间件的方式，确保其开发的软件服务上云数据的安全性。另外，该方法需要对云服务应用进行改造。即使云服务商提供了数据保护，对于用户来说依然是不可信的。

另外，在密文搜索研究方面，Song 等[17]提出了第一个密文搜索方案，该方案通过对密文进行顺序扫描，进而对搜索关键字与密文进行比较，返回包含关键字的密文文件，但搜索准确率较低且计算开销较大。Goh[18]提出了基于 Bloom Filter 的安全索引机制的密文搜索方法，但搜索准确率较差。Curtmola 等[19]提出了基于反向索引的密文搜索方案，提升了搜索效率和准确率，然而，该方案只支持精准关键字搜索。Xia 等[20]提出了基于树的多关键字排序密文搜索方案，可达到亚线性搜索时间，但存在一定的敏感信息泄露。Li 等[21]采用了提前预设基于通配符的关键字模糊集合的密文搜索方案，在建立索引和搜索过程中时间消耗较大。

3 CloudCrypt 系统

3.1 设计目标

面对越来越多的云服务，本文期望得到一个通用的可自动化适配支持各类云应用的数据加密系统，其需要满足的设计目标如下。

1) 提供一个安全隔离数据加密环境，以抵御恶意或被攻陷的云提供商。

2) 支持透明加密并保留丰富的功能如搜索。同时，尽可能地不改变用户习惯。

3) 尽可能以最小代价扩展适配不同的云应用。

3.2 威胁模型

在本文威胁模型中，从以下 3 个方面进行威胁分析。

1) 云服务商

云服务商被认为是诚实且好奇的，存在偷窥或窃取用户隐私的动机，如通过分析用户数据实现商业目的，被黑客攻陷控制来窃取用户隐私数据。

2) 网络中间件

部署在企业内网与云服务器之间的网络中间件

在接收用户上传敏感数据时，可以被攻击成为恶意中间人，进而在通信链路上拦截和解析获取敏感数据。

3) CloudCrypt

在本文威胁模型中，CloudCrypt 部署在企业内部网络，CloudCrypt 的加密密钥存储在可信模块（TPM, trusted platform module）[22]中。TPM 被认为是可信的，因此，恶意内部攻击者将无法获取存储在 CloudCrypt TPM 中对应的密钥，难以解密密文。另外，CloudCrypt 不针对侧信道攻击提供防御和保护能力。

3.3　CloudCrypt 系统架构

CloudCrypt 部署在企业与云服务提供商之间来保护上云的敏感数据。CloudCrypt 采用基于 JavaScript 动态程序分析技术透明加密用户敏感文件数据。如图 1 所示，CloudCrypt 由如下几个重要组件构成。

1) 安全网关

CloudCrypt 基于安全网关开发，安全网关作为中间人可截获浏览器和云服务应用之间的 HTTP/HTTPS 网络流量。对于企业来说，通过安全网关截获 HTTP/HTTPS 请求是合理的，因为企业需要保证企业内部员工上传至云服务的企业敏感数据的安全性。

2) 协议解析器

协议解析器主要用于识别 HTTP/HTTPS 等协议，解析请求内容格式如 HTML、键值对、multipart 等。它通过解析协议和内容，提取重要文件内容，传至 JavaScript 注入模块或数据加解密模块等。

3) 请求转发器

请求转发器是安全网关的一个重要组件，主要

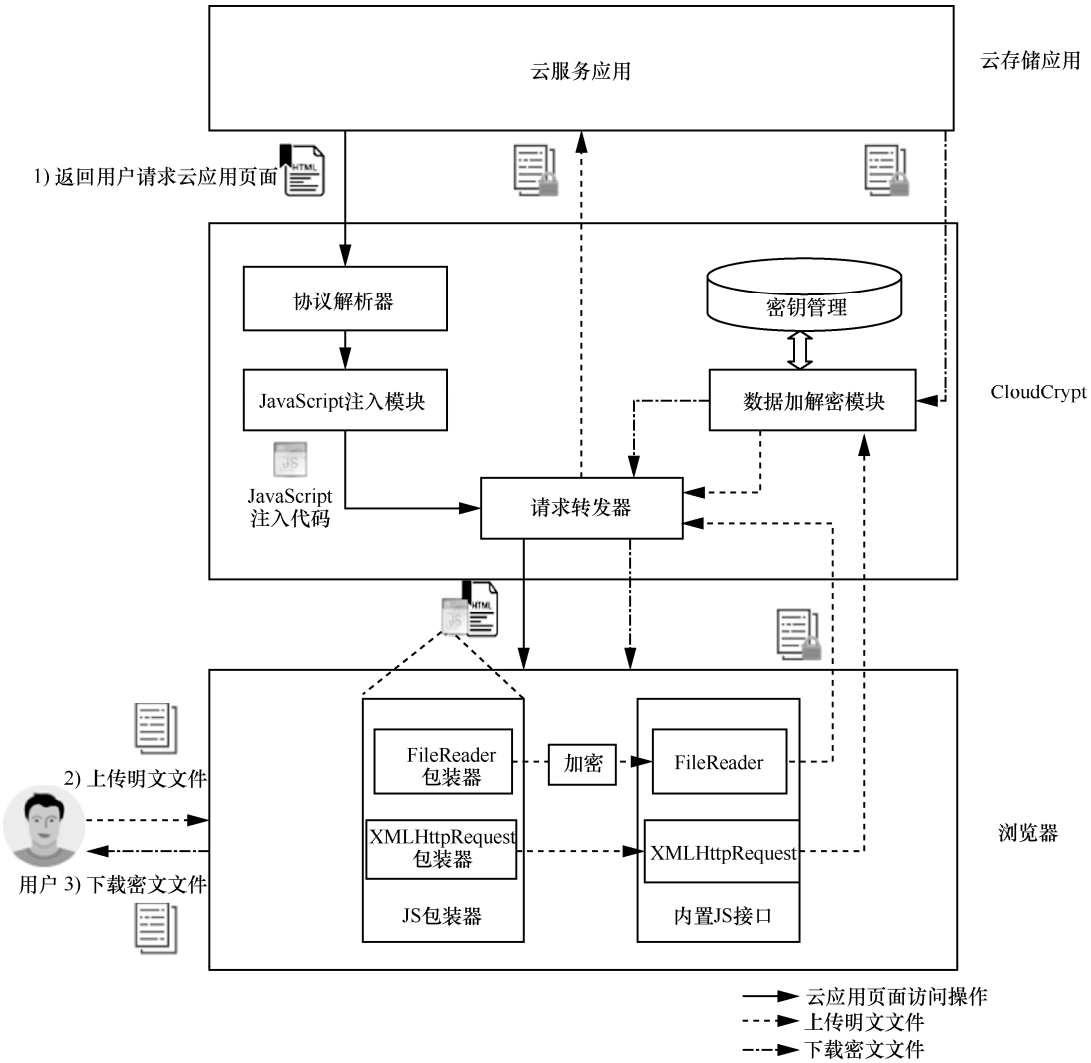


图 1　系统架构

作为代理转发各类请求。

4) JavaScript 注入模块

JavaScript 注入模块接收到云应用 Web 页面后，将 JavaScript 代码片段注入网页的头部。JavaScript 代码片段用来针对 JavaScript 执行环境中的 API 函数设置钩子函数，进而实现程序的动态分析。

5) 数据加解密模块

数据加解密模块用于对敏感文件数据进行对称加密，每个加密数据块均会带有加密标识和密钥 ID，便于识别密文数据和快速获取密文对应的密钥。

6) 密钥管理模块

密钥管理模块基于属性加密的密钥管理机制来管理加密文件密钥，详见第 3.6 节。

7) JavaScript 包装器

JavaScript 包装器由 XMLHttpRequest 包装器和 FileReader 包装器组成，均是针对 JavaScript API 如 XMLHttpRequest[23]和 FileReader[24]设置钩子函数的 JavaScript 脚本代码。其中，XMLHttpRequest 包装器可截获客户端 JavaScript 程序发起的网络请求并对文件上传请求进行识别。FileReader 包装器通过捕获读取文件 I/O 来加密文件。

举例如下。Alice 通过浏览器访问一云存储应用页面，页面访问请求经过 CloudCrypt 时，CloudCrypt 会将 JavaScript 包装器代码注入页面的头部。注入代码后的页面会在浏览器中执行，JavaScript 包装器会首先执行，提前安装钩子函数来修改原始 JavaScript API。它将捕获云应用客户端对 JavaScript API 函数和对象的调用。算法 1 描述了 CloudCrypt 的工作流。

算法 1　CloudCrypt 工作流

输入　浏览器 *B*，云存储服务器 *C*，CloudCrypt 系统 *F*

1) while 接收浏览器发起 HTTP/HTTPS 连接 c do
2)　if c 请求部分包含 HTML 页面 then
3)　　c.response⇐inject_js_wrapper(c.response)
4)　　if c.request 为文件上传请求 then
5)　　c.request ⇐proxy_encryption(request)
6)　　if c.response 包含密文 then
7)　　c.response ⇐ proxy_decryption (response)
8)　　转发请求(c)

算法 1 首先接收一个 HTTP/HTTPS 来自客户端的连接（步骤 1)）。访问云服务的初始请求往往包含应用页面，inject_js_wrapper 函数主要用于注入 JavaScript 包装器到网页的头部，JavaScript 包装器在浏览器中执行并重写 JavaScript API（步骤 3)）。其中，XMLHttpRequest 包装器用来截获网络流量，而 FileReader 包装器用来捕获所有的文件读写操作。proxy_encryption 函数是网关加密函数。CloudCrypt 将会根据应用请求中的加密标记来决定是否需要加密请求中的文件内容，若存在加密标记，则提取出文件内容并加密，再将密文重写至应用请求内容中（步骤 6)）。proxy_decryption 函数是网关解密函数。当应用响应请求中包含有加密文件标识时，安全网关将会从加密文件标识中提取出加密文件的密钥 ID，进而得到对应的密钥，利用密钥对密文解密并将解密后的明文内容重写至响应请求内容中。最后返回给客户端（步骤 7)）。

3.4　自动化数据加密技术实现

CloudCrypt 中最重要的设计目标是自动化适配各类云应用。CloudCrypt 需要自适应地识别各类云服务的文件上传请求。本节首先讨论 2 个可行方案，然后再介绍本文使用的方案：基于 JavaScript 动态程序分析技术实现自动识别文件上传请求。

方案 1　简单地使用正则表达式来匹配各类云服务的文件上传请求。但该方案需要分析每个云服务应用协议请求，形成对应的匹配规则。然而，一旦云服务协议更新，数据保护功能失效，这些匹配规则均需要及时更新。

方案 2　根据文献[25]提出的程序内字符串分析方法对 JavaScript 代码进行解析，进而提取文件上传请求的 URL 字符串、HTTP 方法和对应的请求内容数据。由于云服务往往会将 JavaScript 代码压缩和混淆，这种方法提取到字符串的准确率很低，很难应用于实际需求。

本文针对 JavaScript 采用动态程序分析技术来监听它的文件上传行为，捕获其文件上传请求和文件内容。CloudCrypt 针对 JavaScript 执行环境中的 API 函数设置钩子函数来实现程序的动态分析。当浏览器加载云存储应用的客户端 JavaScript 代码时，首先会提前加载 CloudCrypt 注入的 JavaScript 包装器代码。JavaScript 包装器代码通过设置 API 钩子函数来修改浏览器中 JavaScript 执行环境，之后便可捕获到 JavaScript API 函数、对象方法等调用。

XMLHttpRequest 是客户端 JavaScript 程序向云

服务应用发起网络请求的基础 API。CloudCrypt 基于动态程序分析技术设置 XMLHttpRequest API 钩子函数，可拦截到云服务客户端 JavaScript 程序发起的所有网络请求。而文件上传请求与其他网络请求最大的区别在于 XMLHttpRequest API 函数调用对象类型。文件数据对象往往由 Blob、File、FormData、ArrayBuffer 表示。因此，CloudCrypt 通过设置调用对象类型白名单，检查每个网络请求的调用对象类型，进而自动识别出云应用的文件操作请求。图 2 代码片段描述了 CloudCrypt 基于动态程序分析技术设置 XMLHttpRequest API 钩子函数。

```
var xhr = XMLHttpRequest();
// new a original XMLHttpRequest object
var NativeXMLHttp = XMLHttpRequest;
XHookHttpRequest = function () {
 // hooked method and member
}
XMLHttpRequest = XHookHttpRequest;
var newxhr = XMLHttpRequest();
// new a hooked XMLHttpRequest object
```

图 2 JavaScript 包装器代码片段

3.5 密钥管理技术实现

为了确保在不同用户之间可实现透明加密和共享加密文件，CloudCrypt 引入了基于属性加密的密钥管理机制。在 CloudCrypt 中，每个用户拥有一个专门用于保护文件密钥的主密钥(pku,sku)，其中 pku 是公钥，sku 是私钥，文件密钥加密后称为密钥锁。在实际应用中，用户所拥有的密钥存储和管理在 CloudCrypt 中进行，为了安全考虑，CloudCrypt 将密钥存储在 TPM。

每个企业拥有每个员工的公钥，本文需要一个机制来管理企业用户的密钥信息。公钥机制（PKI, public key infrastructure）[26]在传统密码应用领域常常用来管理密钥，而这种方式需要第三方密钥管理中心。为了避免使用 PKI，本文采用了 IBE（identity-based encryption）[27]机制使 Alice 可以直接用 Bob 的属性（如邮件地址）作为公钥加密数据，而不需要引入第三方来管理公钥。

3.6 密文搜索技术实现

数据加密与数据可用本身是一对矛盾，数据加密后往往会牺牲一些云应用功能，其中典型的功能为搜索功能。密文搜索在一定程度上平衡了功能、性能和安全性 3 个方面。大多数学术研究关注基于加密索引的密文搜索，即用户将文档加密，然后生成可搜索的加密索引并上传至云服务器，再利用搜索关键字对应的搜索陷门（trapdoor）来对云服务器上的加密索引进行搜索，最后得到对应的加密文档[17-21,28-31]。本文将这种方式称为基于云端执行的密文搜索（CESE, cloud executed searchable encryption）算法[32]。然而 CESE 算法需要服务商的配合并对云服务应用做较大改动，显然在实际中很难实行。本文提出了一种基于安全网关执行的密文搜索（PESE, proxy executed searchable encryption）算法。PESE 算法在安全网关处建立索引并关联存储在云服务器上的加密文档标识符。

PESE 算法如算法 2 所示，具体步骤介绍如下。

算法 2 PESE 算法

定义 文档 D，对称密钥 K，对称加密算法 $E_k(D)$，文档 D 标识符 $\mathrm{ID}(D)$，关键字 w 与索引 $L(w)$ 之间的映射函数 $f(w_i)$

建立索引

1) 利用对称密钥 K 对文档 D 加密得到密文文档 $D'=E_k(D)$，并上传至云存储服务器 C

2) 将对称密钥 K 进行加密保护得到 $K'=E_{\mathrm{pku}}(K)$

3) 云存储服务器 C 返回上传成功后的密文文档标识符 $\mathrm{ID}(D')$

4) for 文档 D 中的每个关键字 w_i do

5) if w_i 在关键字集合 W 中 then

6) 根据 $f(w_i)$ 得到对应索引 $L(w_i)$，将 $\mathrm{TF}(w_i,\mathrm{ID}(D'))$ 新增到索引 $L(w_i)$ 中

7) else

8) 将 w_i 加入关键字集合 W 中

9) 根据 $f(w_i)$ 新增对应索引 $L(w_i)$，并将 $\mathrm{TF}(w_i,\mathrm{ID}(D'))$ 新增到索引 $L(w_i)$ 中

10) 返回索引 $I=(W,L)$

搜索

1) 发起搜索请求，输入搜索关键字 $\{w_1,w_2,\cdots\}$

2) 使用搜索关键字 $\{w_1,w_2,\cdots\}$ 对索引进行查询，得到包含关键字 w 的文档标识符 $\{\mathrm{ID}(D_1'),\mathrm{ID}(D_2'),\cdots\}$

3) 根据搜索结果返回文档标识符列表，获取存储在云服务器上的密文文档 $\{D_1',D_2',\cdots\}$

步骤 1 用户上传一个文档 D 到云存储服务器 C_1。

步骤 2 CloudCrypt 拦截上传请求，提取出文档内容，使用一个密钥 K 对文档内容进行加密，然后上传加密文档 D' 到云存储服务器 C_1，并得到云

存储服务器 C_1 返回的加密文档标识符 ID(D')。

步骤 3　CloudCrypt 针对缓存在网关上的文档 D 提取出关键字 ($\{w_1, w_2, \cdots, w_n\}$) 并建立搜索索引 $I = \{t_1, t_2, \cdots, t_n\}$。CloudCrypt 将搜索索引 I 加密后通过云存储接口存储至另一个云存储服务器 C_2。

步骤 4　CloudCrypt 将索引 I 与索引文件标识符 ID(I) 以及加密文档标识符 ID(D') 关联起来。

步骤 5　用户输入搜索关键字 w 并发起查询请求，CloudCrypt 接收到查询请求并将搜索关键字 w 替换为密文搜索词 T_{kw}，并向云存储服务器 C_2 发起搜索查询请求。

步骤 6　云存储服务器 C_2 返回对应的索引文件标识符 ID，之后 CloudCrypt 根据索引文件标识符与加密文档标识符的映射关系表得到对应的加密文档标识符 ID′。

步骤 7　CloudCrypt 使用加密文档标识符 ID′ 向云存储服务器 C_1 发起请求，得到对应的加密文档 D'。

假设有 m 个文档，每个文档 D_i 含有 n 个关键字 ($\{w_1, w_2, \cdots, w_n\}$)，一次查询可搜索关键字数量为 k 个，每个关键字的平均长度为 l，每个关键字 w_i 对应的索引倒排表长度为 $|D(w_i)|$，一次索引更新变化的关键字个数为 $|W_{\text{update}}|$，那么文档集索引倒排表总长度为 $L = \sum w|D(w)|$。由于 PESE 算法采用了在本地构建倒排索引的方式，因此，PESE 算法建立索引的时间复杂度为 $O(L)$，建立索引的空间复杂度为 $O(m+n)$，搜索时间复杂度为 $O(k)$。文献[29]仅支持单关键字搜索功能。文献[30]支持多关键字搜索，但无法支持模糊搜索，且建立索引的时间复杂度为 $O(mn^2)$，建立索引的空间复杂度为 $O(mn)$，搜索时间复杂度为 $O(\theta n \log(m))$，其中 θ 为搜索结果中返回的文件数目。文献[31]支持模糊搜索功能，其建立索引的时间与空间复杂度以及搜索时间复杂度均为 $O(mwl^e)$，其中 e 为编辑距离。如表 2 所示，PESE 算法在安全网关处构建索引并执行搜索操作，可支持丰富的搜索功能，且在建立索引、搜索等维度上的时间与空间复杂度较低，具备较高的性能。

4　系统实现与评价

本节重点讨论 CloudCrypt 系统实现与实验评估结果。实验主要内容和目的包括在 5 个典型云应用中测试评估 CloudCrypt 引入的额外开销、PESE 算法的密文搜索功能创建索引开销与搜索性能效率以及对云应用功能的影响。

4.1　CloudCrypt 系统实现

本文的 CloudCrypt 系统已在多家企业开始实际试用，后续考虑将 CloudCrypt 开源。CloudCrypt 是基于 Squid[33]实现的，其中 Squid 是一款支持 HTTP/HTTPS/FTP 等协议的开源内容缓存网关。而 JavaScript 包装器是基于 XMLHttpRequest 和 FileReader 接口实现的。CloudCrypt 采用 CTR 模式的 AES 标准加密算法[34]来对敏感数据进行加密。CTR 模式的 AES 加密算法确保明文与密文的长度相同。JavaScript 包装器中采用了斯坦福实现的 JavaScript 加密算法库（SJCL, Stanford JavaScript Crypto Library）[35]进行加解密。在网关处采用 OpenSSL 加密算法库[36]来进行加解密。加密后的文件名称为“YAB-098…7F6”，包括加密标志符和密钥锁 ID。‘YAB’是加密标志符，用来表示该文件被 CloudCrypt 已加密。密钥锁 ID 是文件密钥加密后的密文编码。CloudCrypt 采用 IBE 算法库[37]来保护文件加密密钥，详见第 3.6 节。对于基于安全网关执行的密文搜索方案，本文实验采用了常用的搜索引擎 Elasticsearch 在安全网关处来构建索引并关联云存储上的密文文件。同时，本文也采用了 HMAC-SHA-256[38]算法来生成搜索关键字。

4.2　实验设置

本文实验装置主要由一个运行浏览器的虚拟机和一个运行 CloudCrypt 系统的虚拟机组成。运行浏览器的虚拟机配置为英特尔 i7 2.20 GHz 型号 CPU 4 核，内存为 16 GB。运行 CloudCrypt 的虚拟机配置为英特尔 i72.20 GHz CPU 双核，内存为 4 GB。

表 2　**PESE 算法与相关工作搜索效率的对比**

密文搜索方案	建立索引的时间复杂度	建立索引的空间复杂度	搜索时间复杂度	索引更新时间复杂度
文献[29]	$O(L)$	$O(m+n)$	$O(\theta n \log(m))$	$O(\lvert W_{\text{update}}\rvert)$
文献[30]	$O(mn^2)$	$O(mn)$	$O(\theta n \log(m))$	$O(n^2 \log(m))$
文献[31]	$O(mwl^e)$	$O(mwl^e)$	$O(mwl^e)$	不支持
PESE 算法	$O(L)$	$O(m+n)$	$O(k)$	$O(\lvert W_{\text{update}}\rvert)$

4.3 性能测试

CloudCrypt 的核心功能为针对敏感数据的加解密。因此，本文首先评估加解密操作带来的性能开销。加解密操作阶段主要发生在 JavaScript 包装器和安全网关处。JavaScript 包装器在客户端代码读取数据之前进行加密，安全网关在发送数据至云服务器之前对数据进行加密。本文实验通过调用 CloudCrypt 加解密接口来加密明文和解密密文，其文本数据大小从 1 KB 递增至 100 MB，从而评估出数据加解密操作带来的性能损耗。图 3 为在 JavaScript 包装器和安全网关分别调用数据加解密操作 100 次的平均时长。JavaScript 包装器对一个 100 MB 的文件进行加密总耗时为 5 089.3 ms，因为大多数云服务应用的上传文件服务均采用最大不超过 10 MB 的文件分块来上传文件。而针对大小为 100 KB 的文件的加密带来的额外开销仅为 5.5 ms，因此，这个微小时间差对用户体验来讲影响甚微。安全网关加密一个 100 MB 的文件需要花费 520.311 ms，解密一个 100 MB 的文件需要花费 1 631.370 ms。显而易见，与上传或下载一个文件所需总时长相比，加解密操作引入的时间开销较小。

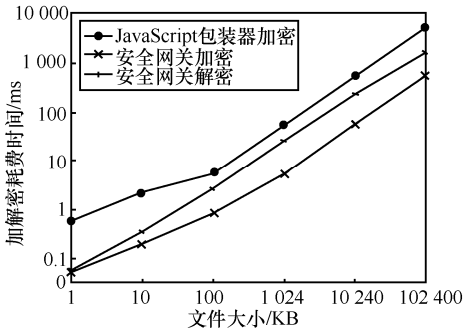


图 3 JS 包装器与安全网关的加解密性能

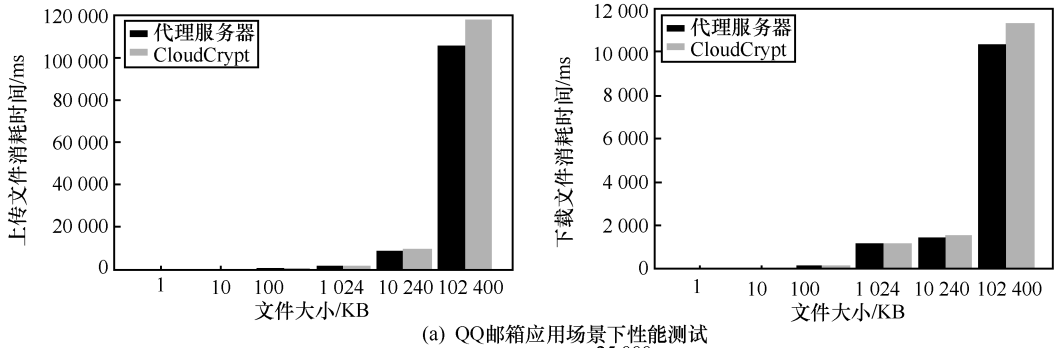
在实验中，本文选择了 5 个主流并被广泛应用的云服务应用来评估 CloudCrypt 带来的实际性能开销。其中，QQ 邮箱是目前主流的电子邮件服务，提供在邮件中传送附件的功能；Box 和 Dropbox 为大量的个人用户和企业客户提供了存储、访问和共享文件的功能；Google Docs 和 Salesforce 是在企业客户中最受欢迎的云办公服务。由于在企业中使用网关来保护内部网络安全较普遍，因此本文直接对比普通网关与 CloudCrypt 这 2 个场景下的性能测试结果。实验中分别在普通网关与 CloudCrypt 这 2 个场景下，将大小从 1 KB～100 MB 不等的文件上传至上述云服务应用，再从云服务器下载已被加密的文件重复 10 次，最后计算平均耗时。图 4 显示了系统性能测试结果，这些耗时通常在毫秒量级。通过比较在普通网关与 CloudCrypt 场景下执行同样的加解密操作，观察到 CloudCrypt 应用在真实云服务应用时带来的性能开销较小。加解密操作造成的时间开销不到 11%。CloudCrypt 对于 1 KB 文件的下载任务引入了 10.52%的开销，对于 100 MB 的文件下载任务引入了 9.23%的开销，而其他文件大小的下载任务带来的开销要低得多。这种显著的差异可能由于网络环境的实时动态变化，导致对小文件的影响更大。此外，随着文件数据大小的增加，大量的内存操作发生在代码中，从而导致性能降低。因此，持续的代码优化将会大大提高 CloudCrypt 的性能。

在 PESE 算法中，CloudCrypt 在本地网关处对上传的明文文档构建明文索引，再将明文索引与存储在云服务器上的密文文档标识符进行关联。本文实验对 PESE 算法与其他密文搜索相关工作支持的搜索功能进行了详细分析，如表 3 所示。与其他相关工作相比，PESE 密文搜索方案不需要云服务商配合，也不需要对云服务应用进行修改，并可支持丰富的搜索功能如多关键字、模糊搜索、结果排序等。

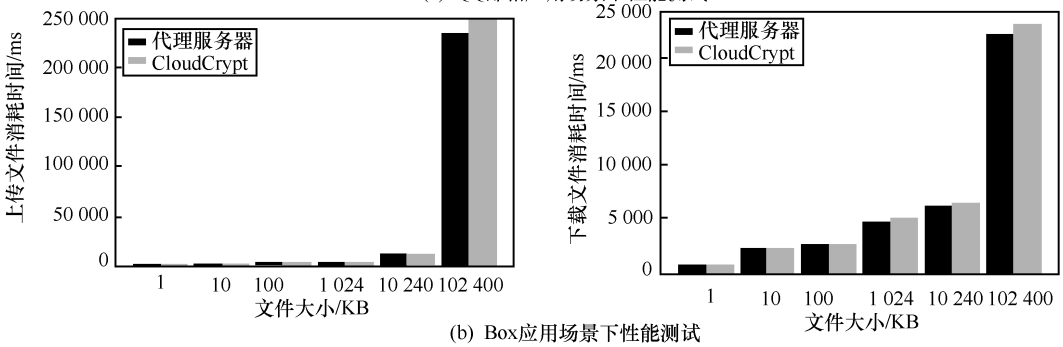
表 3 PESE 算法与相关工作的密文搜索支持功能对比

密文搜索	单关键字	多关键字	模糊搜索	结果排序	动态更新
SWP[20]	√	×	×	×	×
Goh[21]	√	×	×	×	×
LWW[22]	√	×	√	√	×
XWS[23]	√	√	×	√	×
KPR[36]	√	×	×	×	√
PESE	√	√	√	√	√

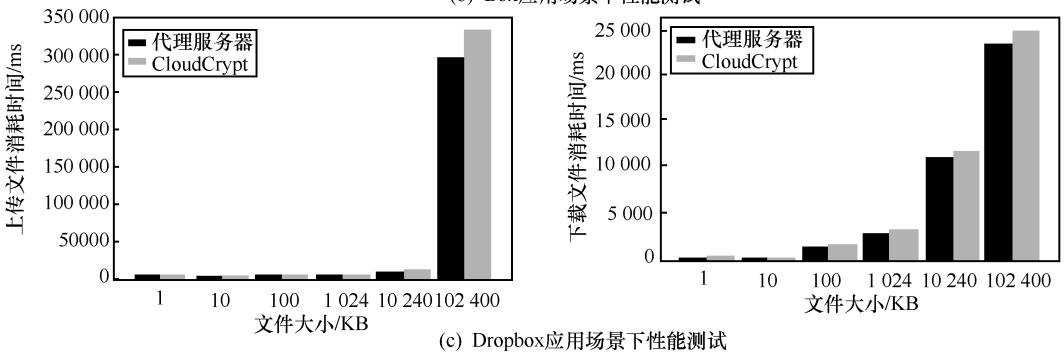
本文实验还着重针对基于安全网关执行的密文搜索方案进行了详细的性能测试。实验采用 esrally[39]对 5 个文件集合来评测搜索性能的好坏。其中，这 5 个文件集合分别为文件集 1（8 697 882 个文件，总大小为 1.21 GB）、文件集 2（10 716 760 个文件、总大小为 1.34 GB）、文件集 3（11 961 342 个文件、总大小为 1.48 GB）、文件集 4（13 053 463 个文件、总大小为 1.62 GB）和文件集 5（17 647 279 个文件、总大小为 2.2 GB）。如图 5 所示，当文件集大小和文件数目增大时，对应的索引大小缓慢增大，搜索性能缓慢下降。当文件集大小超过 2 GB 时，其搜索性能可达



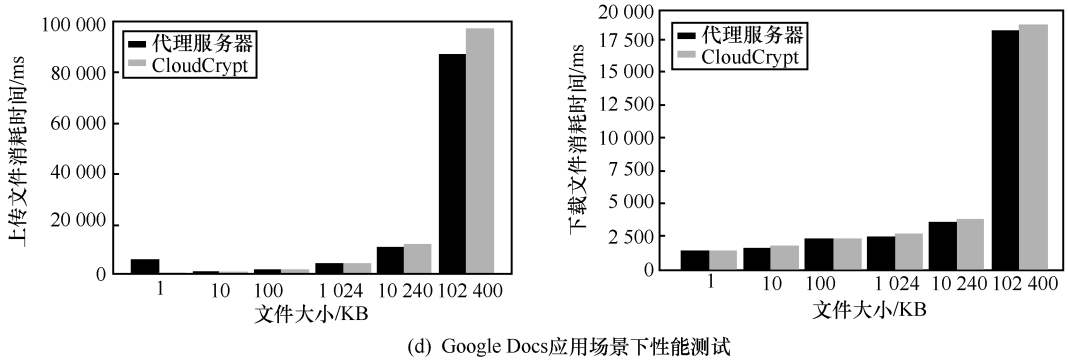
(a) QQ邮箱应用场景下性能测试



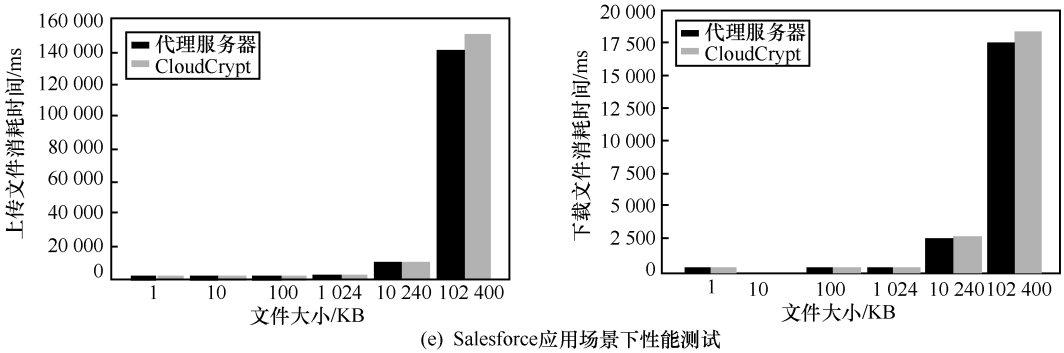
(b) Box应用场景下性能测试



(c) Dropbox应用场景下性能测试



(d) Google Docs应用场景下性能测试



(e) Salesforce应用场景下性能测试

图 4　系统性能测试结果

11.09 ops/s，完全满足实际使用需求。

4.4 案例研究

本节主要讨论 CloudCrypt 在加密敏感数据时对云服务应用功能的影响程度。QQ 邮箱是腾讯公司开发的电子邮件应用，该应用提供邮件内容输入区域，包括邮件主题输入区、邮件正文输入区和邮件附件上传区。QQ 邮箱的文件上传功能在本文实验中的云应用程序中是最复杂的，它会严格检查文件大小和上传文件块大小，很难通过协议分析和代码分析实现提取并加密文件的目的。CloudCrypt 通过 JavaScript 动态程序分析技术注入的 JavaScript 包装器可在客户端代码加载文件之前就对文件进行加密，从而确保上传文件计算的 MD5 校验值与加密文件的 MD5 校验值一致。对于复杂应用 QQ 邮箱的无缝适配，充分表明 CloudCrypt 具有可扩展性。然而，由于云服务器无法解析加密文件的格式内容，用户将无法预览和编辑存储在云中的 PDF 和 WORD 等类型文件。CloudCrypt 应用在 Gmail 时，由于 Gmail 需要解析正常原始文档才能提供文档编辑和预览功能，因此加密邮件附件影响到了 Gmail 的文档编辑功能。尽管丧失了文档预览和编辑功能，但却阻止了云服务商偷窥和泄露用户敏感文档数据的恶意行为。

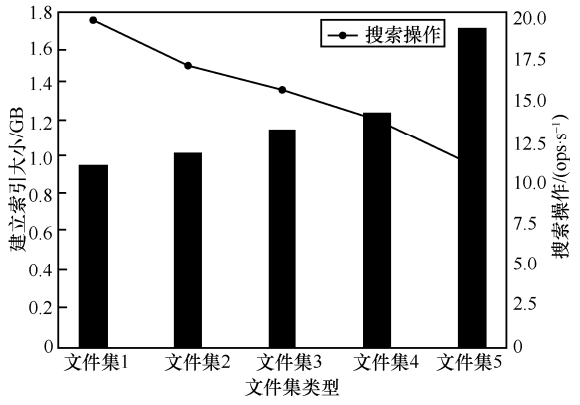


图 5 基于安全网关执行的密文搜索性能

本文实验还将 CloudCrypt 应用于典型的云存储应用如 Dropbox、Box、OneDrive、Google Drive、Mega.nz 上。CloudCrypt 支持对上传至云存储应用文件的加解密，它的引入导致文档预览和分享功能在一定程度有所丧失。企业更希望敏感数据流出时被加密，且更倾向于将 CloudCrypt 应用在云办公应用如 Salesforce、Google Docs 和 Slack 中，可支持对 Salesforce 和 Google Docs 的上传文件进行加密。而 Salesforce 的文件导入功能由于无法解析加密文件而导致功能失效。Slack 通过 WebSocket 连接来传输数据，在未来的工作中，本文方法可扩展基于 WebSocket 的数据加密。

5 结束语

本文提出了一个通用的可自动化实现对基于浏览器云存储应用数据保护的系统 CloudCrypt。该系统不需要云提供商配合，在对敏感数据加密的同时还可最大限度地保持云服务原有的重要功能，在不影响用户体验的前提下确保用户云上数据的机密性。本文实验表明，CloudCrypt 引入代价较小，可支持大部分云服务应用。

参考文献：

[1] FREEBUF. 2019 年网络安全事件回顾（国际篇）[R]. (2020-02-10)[2020-03-08].
FREEBUF. Review of cybersecurity incidents in 2019 (International)[R]. (2020-02-10)[2020-03-08].

[2] SAN JOSE. Ciphercloud: cloud services adoption while ensuring security, compliance and control[R].(2017-06-19)[2020-03-08].

[3] NEIL M, BRIAN L, CRAIG L, et al. Market guide for cloud access security brokers[R]. (2016-10-24)[2020-03-08].

[4] SCOTT R, NATHAN K, BEN B, et al. When automatic encryption leads to confusion and mistakes[C]//Symposium on Usable Privacy and Security. [S.n.:s.l.], 2013: 1-5.

[5] SOARERSOFT SOFTWARE. Folder encryption dog: encrypt your folder and maintain the privacy of your confidential data[R]. (2012-03-19)[2020-03-08].

[6] RUOTI S, ANDERSEN J, MONSON T, et al. Messageguard: a browser-based platform for usable, content-based encryption research[J]. arXiv Preprint, arXiv: 1510.08943, 2015.

[7] RUOTI S, KENT S, DANIEL Z. Layering security at global control points to secure unmodified software[C]//2017 IEEE Cybersecurity Development. Piscataway: IEEE Press, 2017: 42-49.

[8] VIRTRU CORPORATION TERMS. Virtru: email encryption and data security for business privacy[R]. (2017-01-24)[2020-03-08].

[9] WARREN H, DEVDATTA A, SUMEET J, et al. Shadowcrypt: encrypted web applications for everyone[C]//ACM SIGSAC Conference on Computer and Communications Security. New York: ACM Press, 2014: 1028-1039.

[10] GUO X, HUANG Y, YE J, et al. ShadowFPE: new encrypted Web application solution based on shadow DOM[J]. Mobile Networks and Applications, 2020(4): 1-14.

[11] BILLY L, SIMON C, CHENGYU S, et al. Mimesis aegis: a mimicry privacy shield–a systems approach to data privacy on public cloud[C]//23rd USENIX Security Symposium. Berkeley: USENIX Association, 2014: 33-48.

[12] GARTNER. Top 10 security projects for 2019[R]. (2019-03-11)[2020-03-08].

[13] RALUCA P, REDFIELD, NICKOLAI Z, et al. Cryptdb: protecting confidentiality with encrypted query processing[C]//23rd ACM Symposium on Operating Systems. New York: ACM Press, 2011: 85-100.

[14] PODDAR R, TOBIAS B, RULUCA A. Arx: an encrypted database using semantically secure encryption[J]. Proceedings of the VLDB Endowment, 2019, 12(11): 1664-1678.
[15] RALUCA P, EMELIE S, STEVEN V, et al. Building Web applications on top of encrypted data using mylar[C]//11th USENIX Symposium on Networked Systems Design and Implementation. Berkeley: USENIX Association, 2014: 157-172.
[16] BENI E H, LAGAISSE B, JOOSEN W, et al. DataBlinder: a distributed data protection middleware supporting search and computation on encrypted data[C]//The 20th International Middleware Conference Industrial Track. [S.n.:s.l.], 2019: 50-57.
[17] SONG D, DAVID W, ADRIAN P. Practical techniques for searches on encrypted data[C]//2000 IEEE Symposium on Security and Privacy. Piscataway: IEEE Press, 2000: 44-55.
[18] GOH E J. Secure indexes[J]. IACR Cryptology ePrint Archive, 2003 (216): 1-19.
[19] CURTMOLA R, GARAY J, KAMARA S, et al. Searchable symmetric encryption: improved definitions and efficient constructions[J]. Journal of Computer Security, 2011, 19(5): 895-934.
[20] XIA Z, WANG X, SUN X, et al. A secure and dynamic multi-keyword ranked search scheme over encrypted cloud data[J]. IEEE Transactions on Parallel and Distributed Systems, 2016, 27(2): 340-352.
[21] LI J, WANG Q, WANG C, et al. Fuzzy keyword search over encrypted data in cloud computing[C]//Proceedings of IEEE INFOCOM. Piscataway: IEEE Press, 2010: 1-5.
[22] MARGARET ROUSE. Trusted platform module (TPM)[R]. (2019-03-11)[2020-03-08].
[23] MDN CONTRIBUTORS. JavaScript API XMLHttpRequest[R]. (2019-03-23) [2020-03-08].
[24] MDN CONTRIBUTORS. JavaScript API file[R]. (2019-03-23) [2020-03-08].
[25] WITTERN E, ANNIE Y, YUNHUI Z, et al. Statically checking Web API requests in JavaScript[C]//39th International Conference on Software Engineering. Piscataway: IEEE Press, 2017: 244-254.
[26] VACCA J. Public key infrastructure[M]. Public Key Infrastructure: Building Trusted Applications and Web Services, 2004.
[27] ADI S. Identity-based cryptosystems and signature schemes[J]. Lecture Notes in Computer Science, 1985, 196(2): 47-53.
[28] DAN B, GIOVANNI D, RAFAIL O. Public key encryption with keyword search[C]//International Conference on the Theory and Applications of Cryptographic Techniques. Piscataway: IEEE Press, 2004: 506-522.
[29] KAMARA S, CHARALAMPOS P, TOM R. Dynamic searchable symmetric encryption[C]//ACM Conference on Computer and Communications Security. New York: ACM Press, 2012: 965-976.
[30] CURTMOLA R, GARAY J, KAMARA S, et al. Searchable symmetric encryption: improved definitions and efficient constructions[J]. Journal of Computer Security, 2011, 19(5): 895-934.
[31] CHANG Y, MICHAEL M. Privacy preserving keyword searches on remote encrypted data[C]//International Conference on Applied Cryptography and Network Security. Berlin: Springer, 2005: 442-455.
[32] 王国峰, 刘川意, 韩培义, 等. 基于访问代理的数据加密及搜索技术研究[J]. 通信学报, 2018, 39(7): 1-14.
WANG G F, LIU C Y, HAN P Y, et al. Research on technology of data encryption and search based on access broker[J]. Journal on Communications, 2018, 39(7): 1-14.
[33] CLARKE J, ALEX B. The SQUID handbook[M]. Weinheim: Wiley-Vch, 2004.
[34] DWORKIN M. Recommendation for block cipher modes of operation: methods and techniques[J]. National Institute of Standards and Technology Gaithersburg MD Computer Security, 2001, 5(6): 669-675.
[35] STANFORD. Stanford javascript crypto library[R]. (2013-11-17) [2020-03-08].
[36] OPENSSL SOFTWARE FOUNDATION. OpenSSL cryptography and SSL/TLS toolkit[R]. (2009-10-07)[2020-03-08].
[37] DAN B, MATT F, BEN L, et al. Stanford IBE library[R]. (2011-09-20) [2020-03-08].
[38] KRAWCZYK H, RAN C, MIHIR B. HMAC: keyed-hashing for message authentication[R]. (1997-02-01)[2020-03-08].
[39] RALLY. Macrobenchmarking framework for elasticsearch[R]. (2015-12-22) [2020-03-08].

[作者简介]

韩培义（1992− ），男，山西吕梁人，北京邮电大学博士生，主要研究方向为数据安全、云安全。

刘川意（1982− ），男，四川乐山人，哈尔滨工业大学（深圳）副教授，主要研究方向为云计算与云安全、大规模存储系统、数据保护与数据安全。

王佳慧（1983− ），女，山西大同人，国家信息中心博士生，主要研究方向为数据安全、云安全、云取证安全、大数据安全。

段少明（1994− ），男，湖南邵阳人，哈尔滨工业大学（深圳）博士生，主要研究方向为数据安全、机器学习。

潘鹤中（1991− ），男，辽宁本溪人，北京邮电大学博士生，主要研究方向为数据安全、云安全。

方滨兴（1960− ），男，江西上饶人，中国工程院院士，哈尔滨工业大学（深圳）教授，主要研究方向为网络与信息安全、内容安全。