计算机研究与发展
Journal of Computer Research and Development
DOI:10.7544/issn1000-1239.2018.20170415
55(6):1222-1235,2018

基于零知识验证的密文去重与密钥传递方法

何司蒙 杨 超 姜 奇 杨 力 马建峰
(西安电子科技大学网络与信息安全学院 西安 710071)
(陕西省网络与系统安全重点实验室(西安电子科技大学) 西安 710071)
(simenghe@foxmail.com)

Deduplication on Encrypted Data Based on Zero-Knowledge Proof and Key Transmission

He Simeng, Yang Chao, Jiang Qi, Yang Li, and Ma Jianfeng
(*School of Cyber Engineering, Xidian University, Xi'an* 710071)
(*Shaanxi Key Laboratory of Network and System Security (Xidian University), Xi'an* 710071)

Abstract Data deduplication has been widely used in cloud storage servers to reduce bandwidth and save resource effectively. At present, the key chosen to encrypt the file is always the convergent key in the client-based deduplication, so when parts of the file are revealed or the file is poor in entropy, convergent encryption cannot guarantee the semantic security. As for ownership of the file, now the way in some protocols is to check certain numbers of the file blocks to response the challenge of the server, so it cannot prove the whole ownership of the file. In another word, this way is only in a certain probability condition to ensure the ownership of the file. Apart from above, some protocols choose a third party server to participate in the program. Through this way, we need higher security assumption, and it is not suitable for the reality scenes. In this paper, we propose a scheme to deduplicate encrypted data stored in cloud based on zero-knowledge proof and hidden credential retrieval. It uses zero-knowledge proof to achieve the proof of ownership of the file and hidden credential retrieval to transmit the encrypted key to file owners who have proved their ownership of the file. The result shows that our protocol is more efficient and effective. It is easy to be implemented. Meanwhile it improves the security of the ownership authentication and proposes a new key transmission method.

Key words deduplication; proof of ownership (PoW); key transmission; zero-knowledge proof; hidden credential retrieval

摘 要 文件去重技术已广泛运用于云服务器中,有效地减少带宽并提高资源利用率.目前大部分客户端密文去重方案中,文件加密密钥均采用收敛加密,当文件部分信息泄露或文件熵值较小时,收敛加密不能保证语义安全;部分方案中文件所有权认证采取挑战一定数量的文件数据块进行所有权认证,仅能在一定概率条件下通过所有权认证;部分方案中加入可信第三方,需要更高安全假设,不适用于现实场景.针对上述不足,该方案提出了一种新的密文去重场景下所有权认证与密钥传递方法,利用零知识验

收稿日期:2017-06-09;**修回日期**:2017-12-19
基金项目:国家自然科学基金面上项目(61672415,61672413,61671360);陕西省自然科学基础研究计划基金项目(2017JM6054);111基地专项基金项目(B16037)
This work was supported by the General Program of the National Natural Science Foundation of China (61672415, 61672413, 61671360), the Natural Science Foundation of Shaanxi Province (2017JM6054), and the 111 Project (B16037).
通信作者:杨超(chaoyang@xidian.edu.cn)

证方法,通过不损失熵的文件大摘要实现文件所有权认证,利用隐藏凭据恢复方法实现密钥安全传递.该方案具有密钥与文件分离、完整所有权认证、不使用第三方传递密钥等特点.安全性分析理论证明本方案所有权认证及密钥传递达到了可证明的安全强度,实际云平台测试数据表明:该方案减少了密文去重运算量,使用户可以更高效地使用云服务.

关键词 去重;所有权认证;密钥传递;零知识验证;隐藏凭据恢复方法

中图法分类号 TP391

随着互联网的快速发展,网络用户数量激增,产生海量数据,然而用户本地计算和存储能力有限,因此,云计算服务给数据计算和存储提供了新的解决方案,用户通过云服务器,可以享受到快速高效的计算资源,可以更便捷地共享存储资源.然而,云存储服务器面临了2种矛盾的需求:1)云服务器需要减少数据存储空间的开销;2)用户出于数据安全和隐私的考虑,往往希望自己的数据加密存储.如果用户拥有相同的文件数据,那么云服务器可以通过对待上传的文件进行重复性检测来判断是否需要上传该文件,实现相同文件去重工作;并且云服务器不需要重复存储,有效提高云服务器存储利用率.研究表明:去重工作减少了90%~95%的后台备份数据的存储[1],减少了68%的标准文件系统存储[2],这些系统资源提升工作为云计算发展带来了明显的经济价值.

用户在上传文件至云服务器前,为了确保数据隐私安全,会选择加密文件后再进行上传[3-5].由于加密密钥选择的多样性,即使拥有相同的明文,也会计算得到不同的密文[6],这就使得拥有相同明文的不同用户在上传自己加密数据后,服务器无法识别其重复性,从而造成大量相同数据重复存储,导致云存储利用率低.因此,如何在文件加密场景下完成文件所有权认证、高效地进行去重工作、提高云存储的利用率成为了当前的研究热点.

早在2011年,Halevi等人[7]就提出了"所有权认证"(proof of ownership, PoW)的概念,PoW策略提出服务器应验证用户是否真正拥有某个文件,而不仅是与文件相关的短消息.在文件所有权认证步骤中,客户端在原始明文基础上建立Hash树,云服务器要求客户端返回随机选定的叶子节点及其到根节点路径上的节点集,若所有叶子节点及其到根节点路径上的节点与服务器端运算结果相同,那么服务器通过该客户端的文件所有权认证,否则所有权认证失败.但是,在该方案中客户端需要消耗大量资源建立Hash树,对客户端计算能力要求较高;并且该方案建立在明文数据上,不利于保护客户端数据安全.

收敛加密由Douceur等人[8]首先提出,在该策略中,将文件Hash值作为文件加密密钥,利用对称加密算法对文件进行加密.文献[9]提出了客户端密文文件级去重方法,该方法利用收敛加密与文件所有权认证方法相结合,这样相同的明文加密后将得到相同的密文,有利于服务器识别文件的重复性.但收敛加密是不安全的,因为它的加密密钥是通过确定的算法由文件本身生成,因此容易被泄露.某种程度上,收敛加密所受到的攻击和"Hash as a proof"方法所受到的攻击是一致的[7].并且由于加密密钥由文件Hash值生成,那么低熵文件的安全在该方法中无法保证.

此外,文献[10]提出了一种用于用户敏感数据的客户端去重方案.该方案能够在诚实而好奇的服务器场景下保护数据的隐私.在该方案中,用于加密文件的Hash值作为文件所有权认证的证据,这种方法仍然是用短消息表示文件,所受到的攻击和"Hash as a proof"方法所受到的攻击是一致的[7].

虽然收敛加密已被广泛应用,但由于它没有达到语义安全,因此,文献[11]提出消息锁定式加密(message-locked encryption, MLE)框架,规范了去重方案的机密性和完整性定义,保证数据在不可预测情况下做到语义安全,收敛加密即为MLE的一个特例.但该方案中使用凭据T作为拥有文件的凭证,若凭据T泄露则密文泄露,无法保证文件的机密性.

由于收敛加密算法中密钥与明文强关联,无法抵抗暴力破解攻击,攻击者可以利用暴力破解方式根据Hash值生成的加密密钥得到原始明文,因此文献[12]提出了DupLESS方案.在该方案中,增加了一个密钥服务器(key-server)用来生成文件的加密密钥,而不是直接用明文Hash值作为加密密钥,并且密钥服务器生成的密钥也与原始文件相关,这样可以防止由于数据可预测导致的MLE不安全问

题.然而该方案引入了负责密钥生成的第三方密钥服务器,这样做需要更高的安全假设前提,在现实中难以保证和实现.

文献[13]提出了一种对文件预处理后客户端去重的方案,用户对文件利用纠删码扩展加工后分块,在文件数据块上建立 Hash 树,将 Hash 树的根节点作为摘要,与服务器进行所有权认证.该方法虽然与原始 Hash 树所有权认证方法相比有所改进,但是在该方法验证所有权时,随机选择一定数量的文件块进行所有权认证是概率条件下的认证,不能完整验证文件所有权,无法保证所有权认证的安全性.

文献[14]提出了一种基于(n,k,r)-RSSS(ramp secret sharing scheme)[15-16]的收敛密钥管理方案 Dekey,该方案中首位上传者生成并保存共享秘密信息(secret shares),对于 n 个共享秘密信息的实体,其中任 k 个可以恢复秘密信息,任 r 个不能推理出秘密信息,因此在该方案中需要利用(n,k,r)-RSSS 机制将密钥存储在多个密钥管理服务器上,恢复密钥时需要获得一定的先验知识后通过多个密钥管理服务器进行密钥恢复.该方案在密钥管理方面有所改进,但由于方案中需要多个密钥管理服务器参与,整个方案的安全性假设仍然较高,不易实现.

文献[17]提出了一种支持文件级和数据块级去重的客户端去重方案,由于该方案建立在两级密钥上,需要利用主密钥和标签值共同加密文件或文件数据块,其中标签值用于文件所有权认证,因此在客户端运算过程中需要计算大量的中间数据,并且客户端需要保留主密钥;所有权认证建立在低熵的标签值上,在所有权认证过程中会泄露原始明文信息,无法保证文件的安全性.

在此基础上,文献[18]提出了一种利用第三方服务器代理重加密的方法实现文件密钥传递及椭圆曲线密码算法实现加密文件所有权认证的方案.在该方案中,需要利用第三方代理服务器实现密钥信息传递,因此对代理服务器提出更高的安全假设,增加了第三方交互运算量,在实际中需要更高的假设前提.

由上述加密场景去重方法的分析可知以下 3 点:

1) 大部分方案为了能够实现加密场景下去重,均采用收敛加密方法,这些方案认为若不使用收敛加密方法,由于加密密钥选择的多样性,会阻止拥有相同明文的密文去重;然而当文件部分信息泄露或文件熵值较小时,收敛加密方法不能保证语义安全,攻击者可利用收敛加密密钥恢复原始明文,导致明文信息泄露.

2) 部分去重方案中,文件所有权认证仅基于部分文件的挑战应答,而非整体文件信息;这种基于概率条件下的所有权认证,不能完整确认文件所有权,当文件信息泄露时,攻击者在一定概率条件下可猜测文件信息,从而通过所有权认证导致文件信息泄露.

3) 如何将首位上传者的文件密钥安全传递至后续上传者是密文去重场景下的一个重要环节,但目前除了使用收敛加密方法外,其他方案均需使用第三方服务器辅助分发密钥,这种方式需要更高的安全假设支持,并且方案性能受第三方服务器影响,不适用于具体实际场景.

因此,目前的方案还未能同时满足密钥与文件内容分离、完整所有权认证及不使用第三方传递密钥这 3 个条件,如何建立一个具有密钥与文件内容分离、完整所有权认证、不使用第三方服务器传递密钥等特点的密文去重方案是当前亟待解决的问题.

综合上述问题,本文提出了一种新的密文去重场景下所有权认证与密钥传递方法,其主要思路是:文件首位上传者对文件进行预处理,利用独立成对 Hash 方法生成不损失熵的文件大摘要,利用隐藏凭据恢复方法注册阶段,对文件密钥进行保护处理生成凭据,并将凭据与文件密文上传至云服务器.后续上传者与服务器利用零知识验证方法,通过不损失熵的文件大摘要进行文件所有权认证交互,若后续上传者所有权认证成功,则服务器将后续上传者标记为文件拥有者,通知后续上传者删除本地文件,实现客户端密文去重;否则文件所有权认证失败.通过所有权认证的文件拥有者可利用隐藏凭据恢复方法传递阶段恢复文件密钥,后续可利用该密钥解密密文.本方案具有密钥与文件内容分离、完整所有权认证、不使用第三方服务器传递密钥等特点,更适用于现实场景.

通过安全性分析理论证明本方案所有权认证及密钥传递达到了可证明的安全强度,实际云平台的测试数据表明,本方案时间效率良好,减少了密文去重的运算量,使用户可以更方便高效地使用云服务.

1 预备知识

1.1 隐藏凭据恢复方法

隐藏凭据恢复方法(hidden credential retrieval, HCR)[19]是一种在重复使用用户口令的情况下,将用户的秘密信息安全地存储在不可靠的服务器上,

并能安全地恢复秘密信息的协议方法,常运用于云存储中基于口令的加密系统. HCR 协议建立在 Diffie-Hellman 签名模式上[20],在 HCR 协议方法中,用户仅需要保存一个秘密口令 pwd,服务器即使是恶意的、不可信的,也可以实现将用户信息 msg 安全存储在服务器端.

隐藏凭据恢复方法分为 2 个阶段:注册阶段和传递阶段,通过这 2 个阶段的运算,从而实现对加密信息的安全传递.

1) 注册阶段

① 用户选择随机数 v 和随机数 S;

② 客户端计算中间变量 $h=v^S$;

③ 客户端计算口令传递值 $D=(hash(pwd))^S$;

④ 客户端计算口令传递保护值 $r=msg\times D^{-1}$;

⑤ 用户将(v,h,r,S)发送至服务器并存储.

2) 传递阶段

① 服务器检索后发送(v,h,r)至用户;

② 客户端选择一个随机数 R,由此计算口令证据值 $U=v^R hash(pwd)$,并将口令证据值 U 发送至服务器;

③ 服务器计算口令证据验证值 $B=U^S$,将口令证据验证值 B 发送至客户端;

④ 客户端计算口令传递值 $D=B\times h^{-R}$;

⑤ 客户端计算 $msg=r\times D$,恢复得到原始信息 msg.

1.2 零知识验证协议

零知识验证协议[21]是交互系统中常用的一种认证协议,证明者能够在交互多次后让验证者相信自己声称拥有某一秘密的正确性. 零知识验证一般分为 3 步:承诺、挑战、应答.

1) 利用系统参数,证明者选择随机数,生成承诺值 $commit$,将承诺值 $commit$ 发送至验证者;

2) 利用系统参数,验证者选择随机数,生成挑战值 $chal$,将挑战值 $chal$ 发送至证明者;

3) 证明者根据挑战值 $chal$ 及自己拥有的秘密信息,生成证据值 $Proof$,将证据值 $Proof$ 发送至验证者完成应答.

1.3 独立成对 Hash 方法

独立成对 Hash 方法(pairwise independent Hash)[22]是一种安全的 Hash 族方法,包含一个 Hash 族群,其中每一个 Hash 函数都是独立成对的,即对于任意的 2 个不同的输入值以及任意 2 个输出值,其结果相对应的概率仅与 Hash 族群的长度有关,与输入值输出值无关,且不损失信息熵.

根据不同的输入值,利用独立成对 Hash 方法,可生成对应不损失熵的输出值.

2 系统模型与敌手模型

2.1 系统模型

本文方案的系统模型如图 1 所示,主要包括云存储服务器和客户端 2 个实体.

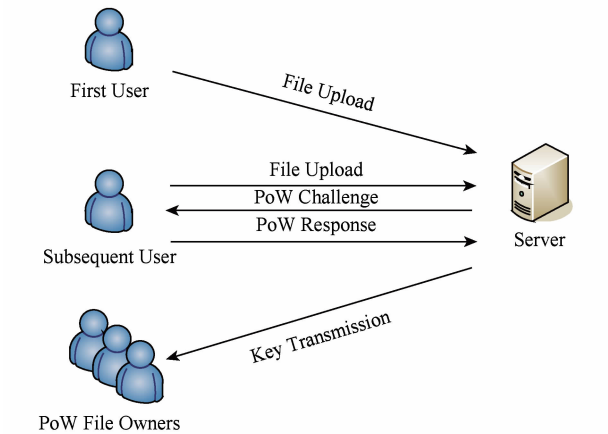

Fig. 1 System model of our scheme

图 1 本文方案系统模型图

1) 云存储服务器. 拥有大量存储空间,可提供数据存储服务,用户通过购买或租赁一定大小的云存储空间成为云存储用户,在提供云存储服务过程中,安全传递文件加密密钥,参与用户所有权认证交互,完成密文去重工作,提高存储利用率.

2) 用户客户端. 在密文去重场景中,用户分为首位上传者、后续上传者以及拥有某一文件的用户. 用户通过上传文件至云存储服务器完成文件数据的存储备份,其中首位上传者上传密文并利用隐藏凭据恢复方法保护加密密钥的传递;后续上传者与服务器利用不损失熵的文件大摘要及零知识验证方法进行所有权认证交互,若通过认证,则不再需要上传文件,实现客户端密文去重;通过所有权认证的文件拥有者可利用隐藏凭据恢复方法安全获取文件加密密钥,后续可解密密文得到原始文件.

2.2 敌手模型

对应于 2.1 节所述系统模型的敌手模型如下:

1) 外部攻击者. 外部攻击者可能获取了用户的部分数据,或是在网络传输中截获到部分数据,从而发起重放攻击,破坏密文去重中所有权认证交互.

2) 内部攻击者. 内部攻击者是指能够直接访问到存储数据的攻击者,包括云存储服务器及客户端恶意软件. 诚实而好奇的服务器可能会非法访问数据,客户端恶意软件可能会对密钥进行字典攻击等.

系统安全性假设:

1) 假设云存储服务器与文件拥有者相互独立且不共谋;

2) 假设文件拥有者上传文件正确匹配的不损失熵的文件摘要值及 Hash 值.

3 ZHCR-Dup 方案设计实现

3.1 方案设计思想

本文方案实现了一种新的密文去重场景下所有权认证与密钥传递方法,用于解决现有技术中所有权认证安全性低的问题,并实现密文去重场景下密钥的安全传递. 其设计思想是:首位上传者对文件进行预处理,利用独立成对 Hash 方法生成不损失熵的文件大摘要,利用隐藏凭据恢复方法注册阶段,对文件密钥进行保护生成凭据,并将凭据与文件密文上传至云服务器. 后续上传者与服务器利用零知识验证方法,通过不损失熵的文件大摘要进行文件所有权认证交互,若后续上传者所有权认证成功,则服务器将后续上传者标记为文件拥有者,通知后续上传者删除本地文件,实现客户端密文去重;否则文件所有权认证失败. 通过所有权认证的文件拥有者可以利用隐藏凭据恢复方法传递阶段,恢复文件密钥,后续可以利用该密钥解密密文. 本文方案整体协议框架如图 2 所示:

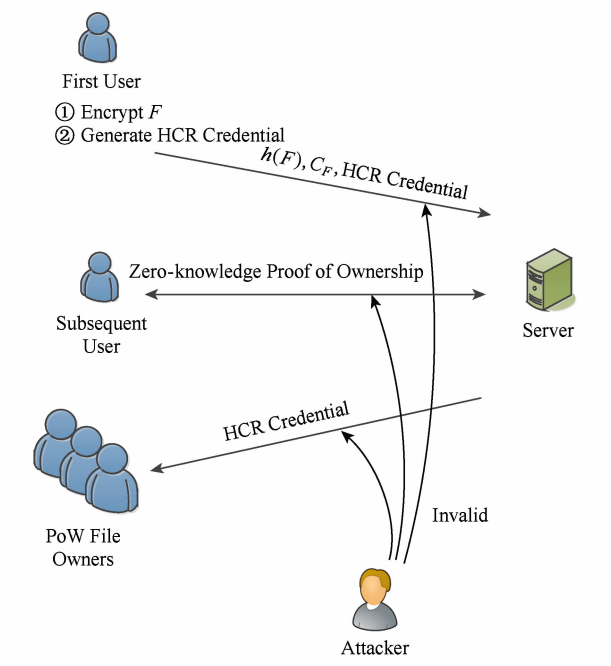

Fig. 2 Protocol framework of our scheme

图 2 本文方案协议框架示意图

3.2 方案详细设计

本方案包括用户客户端和云服务器 2 个实体,用户通过服务器实现文件所有权认证及加密密钥传递,本文方案的详细设计分别如图 3~5 所示.

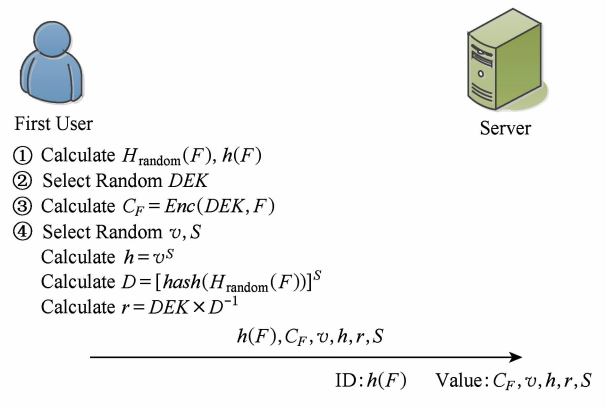

Fig. 3 First user operation in the protocol

图 3 首位上传者交互流程图

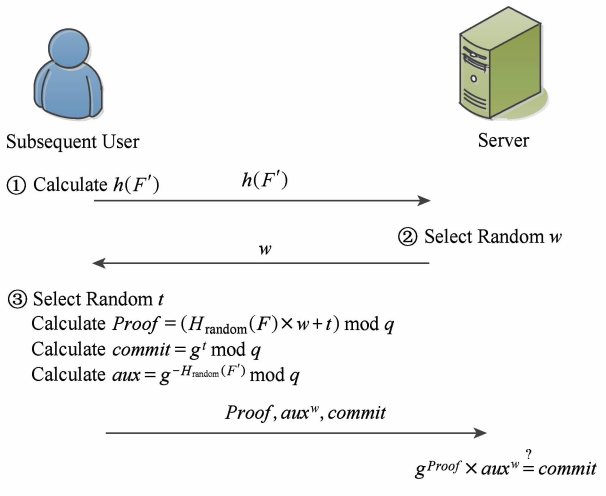

Fig. 4 Subsequent user operation in the protocol

图 4 后续上传者交互流程图

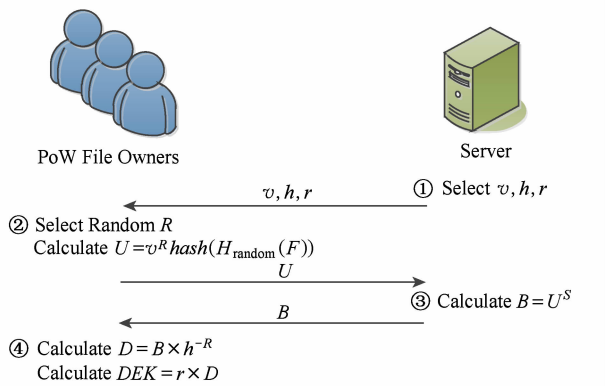

Fig. 5 Key transmission operation in the protocol

图 5 密钥传递交互流程图

本方案系统初始化定义为

p 与 q 是 2 个素数,满足关系 $q \mid p-1$;

G 是以素数 p 为阶的乘法循环群,其中 g 是 G 上的一个生成元,$G=\{g^0,g^1,\cdots,g^{p-1}\}$;

$h()$为 md5Hash 函数;

$hash()$为 SHA1Hash 函数;

$H_{\text{random}}()$为独立成对 Hash 方法;

DEK 为首位上传者随机选择的文件加密密钥;

$Enc()$为对称加密方法;

C_F 为加密得到的密文文件.

步骤 1. 文件首位上传者 U_1 对明文 F 进行预处理,并将预处理结果上传至服务器,详细设计如图 3 所示,具体过程如下:

步骤 1.1. 首位上传者 U_1 利用 md5Hash 函数 $h()$,计算明文 F 的索引值 $h(F)$;

步骤 1.2. 首位上传者 U_1 利用独立成对 Hash 方法,计算明文 F 的不损失熵的文件大摘要 $H_{\text{random}}(F)$,其中:

步骤 1.2.1. 首位上传者 U_1 对明文 F 进行分块,得到明文 $F=\{w_1,w_2,\cdots,w_i,\cdots,w_l\}$,其中 w_i 表示明文 F 中的第 i 块,l 表示块的数量,且 $i\in[1,l]$;

步骤 1.2.2. 首位上传者 U_1 利用 SHA1Hash 函数 $hash()$,计算明文 F 中每一块 w_i 的 Hash 值 $y_i=hash(w_1\parallel w_2\parallel\cdots\parallel w_i)$,并将所有块的 Hash 值联结,得到明文 F 正向 Hash 值 $Y=\{y_1\parallel y_2\parallel\cdots\parallel y_l\}$;

步骤 1.2.3. 首位上传者 U_1 对明文 F 进行逆序处理,得到明文 F 逆序文件 $\bar{F}$,并对明文 F 逆序文件 $\bar{F}$ 进行分块,得到逆序文件 $\bar{F}=\{\bar{w}_1,\bar{w}_2,\cdots,\bar{w}_i,\cdots,\bar{w}_l\}$,其中 $\bar{w}_i$ 表示逆序文件 $\bar{F}$ 中的第 i 块,l 表示块的数量,且 $i\in[1,l]$;

步骤 1.2.4. 首位上传者 U_1 利用 SHA1Hash 函数 $hash()$,计算明文 F 逆序文件 $\bar{F}$ 中每一块 $\bar{w}_i$ 的 Hash 值 $z_i=hash(\bar{w}_1\parallel\bar{w}_2\parallel\cdots\parallel\bar{w}_i)$,并将所有块的 Hash 值联结,得到明文 F 逆向 Hash 值 $Z=\{z_1\parallel z_2\parallel\cdots\parallel z_l\}$,最终对明文 F 逆向 Hash 值 Z 进行逆序处理,得到明文 F 逆向 Hash 转换值 $\bar{Z}$;

步骤 1.2.5. 首位上传者 U_1 通过明文 F 正向 Hash 值 Y 和明文 F 逆向 Hash 转换值 $\bar{Z}$,计算明文 F 的不损失熵的文件大摘要 $H_{\text{random}}(F)=Y\oplus\bar{Z}$,其中⊕表示异或操作;

步骤 1.3. 首位上传者 U_1 随机生成文件对称加密密钥 DEK,并利用该对称加密密钥 DEK 对明文 F 进行加密,得到文件密文 $C_F=Enc(DEK,F)$,其中 $Enc()$为对称加密方法;

步骤 1.4. 首位上传者 U_1 对文件对称加密密钥 DEK 的保护:首位上传者 U_1 选择随机数 v 和随机数 S,利用隐藏凭据恢复方法,通过随机数 v 和随机数 S,计算中间变量 $h=v^S$,并通过随机数 S 和文件大摘要 $H_{\text{random}}(F)$,计算文件对称加密密钥 DEK 传递值 D,再通过传递值 D 和对称密钥 DEK,计算文件对称加密密钥 DEK 传递保护值 r,详细过程如下:

步骤 1.4.1. 首位上传者 U_1 通过明文 F 的不损失熵的文件大摘要 $H_{\text{random}}(F)$和随机数 S,计算文件对称加密密钥 DEK 传递值 $D=(hash(H_{\text{random}}(F)))^S$,其中 $hash()$为 SHA1Hash 函数;

步骤 1.4.2. 首位上传者 U_1 计算文件对称加密密钥 DEK 传递保护值 $r=DEK\times D^{-1}$;

步骤 1.5. 首位上传者 U_1 将随机数 v、随机数 S、中间变量 h 和文件对称加密密钥 DEK 传递保护值 r 发送至服务器并存储,实现对文件对称加密密钥 DEK 的安全传递,同时将明文 F 的索引值 $h(F)$ 和文件密文 C_F 发送至服务器并存储.

步骤 2. 后续上传者 U_2 与服务器进行文件所有权认证交互,交互流程如图 4 所示,具体过程如下:

步骤 2.1. 后续上传者 U_2 利用 md5Hash 函数 $h()$,计算明文 F' 的索引值 $h(F')$,并将索引值 $h(F')$发送至服务器;

步骤 2.2. 服务器判断明文 F'索引值 $h(F')$与存储的明文 F 索引值 $h(F)$是否相等,若是,选择随机数 w,并将随机数 w 发送至后续上传者 U_2,否则,结束运算;

步骤 2.3. 后续上传者 U_2 利用独立成对 Hash 方法,计算明文 F'的不损失熵的文件大摘要 $H_{\text{random}}(F')$,同时选择随机数 t,并利用零知识验证方法,通过明文 F'的不损失熵的文件大摘要 $H_{\text{random}}(F')$、随机数 w 和随机数 t,在生成元为 g 的 p 阶乘法循环群 G 中计算所有权认证的证据值 $Proof$、承诺值 $commit$、辅助值 aux 和辅助验证值 aux^w,最终将所有权认证的证据值 $Proof$、承诺值 $commit$ 和辅助验证值 aux^w 发送至服务器,运算过程如下:

步骤 2.3.1. 后续上传者 U_2 计算所有权认证的证据值 $Proof=(H_{\text{random}}(F')\times w+t)\bmod q$;

步骤 2.3.2. 后续上传者 U_2 计算所有权认证的承诺值 $commit=g^t\bmod q$;

步骤 2.3.3. 后续上传者 U_2 计算所有权认证的

辅助值 $aux=g^{-H_{\text{random}}(F')} \bmod q$，并通过所有权认证的辅助值 aux 和随机数 w，计算所有权认证的辅助验证值 aux^w，其中，q 是一个素数，且 $q|p-1$；

步骤 2.4. 服务器利用零知识验证方法，通过所有权认证的证据值 $Proof$，在生成元为 g 的 p 阶乘法循环群 G 中计算所有权认证的证据验证值 g^{Proof}，并判断 $g^{Proof}\times aux^w$ 与承诺值 $commit$ 是否相等，若相等，后续上传者 U_2 文件所有权认证成功，将文件所有权认证成功的后续上传者 U_2 标记为文件拥有者，并通知后续上传者 U_2 删除本地明文 F'，从而实现客户端密文去重，否则，后续上传者 U_2 文件所有权认证失败，结束运算；

步骤 3. 利用服务器实现密钥传递，交互流程如图 5 所示，具体过程如下：

步骤 3.1. 服务器将随机数 v、中间变量 h 和对称密钥 DEK 传递保护值 r 发送至通过所有权认证的文件拥有者；

步骤 3.2. 文件拥有者选择随机数 R，并通过随机数 R、文件大摘要 $H_{\text{random}}(F)$ 和随机数 v，计算文件大摘要 $H_{\text{random}}(F)$ 证据值 $U=v^R hash(H_{\text{random}}(F))$，其中 $hash()$ 为 SHA1Hash 函数，再将证据值 U 发送至服务器；

步骤 3.3. 服务器通过证据值 U 和随机数 S，计算文件大摘要 $H_{\text{random}}(F)$ 证据验证值 $B=U^S$，并将证据验证值 B 发送至文件拥有者；

步骤 3.4. 文件拥有者通过证据验证值 B、中间变量 h 和随机数 R，计算文件对称加密密钥 DEK 传递值 $D=B\times h^{-R}$，并通过对称密钥 DEK 传递保护值 r 和传递值 D，计算文件对称加密密钥 $DEK=r\times D$.

文件拥有者首先利用不损失熵的大摘要通过零知识验证方法完成所有权认证，再利用隐藏凭据恢复方法通过云服务器安全获取到文件加密密钥，实现了客户端密文去重，并成功传递文件密钥，整体协议结束.

4 安全性分析

4.1 基于零知识验证的所有权认证安全性分析

定理 1. 假设离散对数问题的困难性，那么本文设计的所有权认证方案是一个零知识验证方案.

证明.

具有零知识验证特性的交互协议方案具有完备性、正确性、零知识性这 3 个属性，离散对数问题的困难性意味着攻击者根据所有权认证的证据验证值 g^{Proof} 无法推导得到证据值 $Proof$.

1) 完备性. 对于协议方法中的证明者和验证者，如果诚实地遵守协议方法流程，那么诚实的验证者接受该证明者的证明.

完备属性意味着只有当客户端证明者拥有原始明文文件，诚实的证明者与验证者按照协议规定的交互流程进行，那么验证者通过证明者所有权认证的概率为 1. 验证过程为

$$\begin{aligned}&(g^{Proof}\times aux^w)\bmod q=\\&(g^{H_{\text{random}}(F)\times w+t}\times g^{-H_{\text{random}}(F)\times w})\bmod q=\\&g^t \bmod q=commit.\end{aligned}\tag{1}$$

2) 正确性. 对于一个不正确的协议方案，通过调整其协议交互策略，从而能够使验证者通过证明者所有权认证，该事件发生的概率可以忽略.

正确属性意味着当攻击者没有原始文件时，无论攻击者使用什么方式，服务器端都不会通过其所有权认证.

假设攻击者没有正确完整的原始文件，那么攻击者不能计算出正确的证据值 $Proof$ 和证据辅助值 aux，因此攻击者也无法计算得到正确的承诺值 $commit$，当攻击者将不正确的证据值 $Proof$、承诺值 $commit$ 和辅助验证值 aux^w 发送至服务器后，服务器判断服务器端存储的 $g^{Proof}\times aux^w$ 与承诺值 $commit$ 不相等，因此攻击者所有权认证失败，保证了协议的正确属性.

3) 零知识性. 证明者和验证者在整个协议的交互过程中，生成的证明副本可以由平均拟多项式时间(pseudo-polynomial time, PPT)算法生成，并且由该算法生成的证明副本与真实的证明副本是不可区分的，即通过模拟可以实现无差别的证明过程.

零知识属性意味着一个诚实而好奇的服务器，在本协议方法中，除了能获知客户端是否拥有原始文件外，不能获知与原始文件相关的其他任何信息. 这种零知识属性被定义为“诚实验证者零知识”，在本协议方案中能够提供足够的安全性.

假设存在一个模拟器，它与服务器之间进行交互，能够替代拥有原始文件的客户端生成一个证明副本，并且该证明副本与拥有原始文件的客户端所生成的证明副本在计算复杂度上无差别.

在整个协议的辅助值 aux 基础上，模拟器可以在平均拟多项式时间 PPT($\text{in}|q|$)内计算证明副本，步骤如下：

1) 初始化证明副本为空串;

2) 选择随机数 w,t;

3) 利用文件大摘要 $H_{\text{random}}(F)$ 计算证据值 $Proof=(H_{\text{random}}(F)\times w+t)\bmod q$;

4) 计算承诺值 $commit=(g^{Proof}\times aux^{w})\bmod q$;

5) 证明副本 $copy\leftarrow commit$.

假设证明者和验证者诚实地按照协议方法流程交互,那么可以描述一个模拟器对证明交互进行模拟,生成证明副本 $copy$. 模拟器可以在拟多项式时间 PPT 内生成该证明副本,在其运算过程中没有原始明文参与运算,不会泄露原始明文的信息,并且在每一次协议过程中都使用随机数,防止攻击者进行重放攻击.

综上,在诚实的云服务器场景中,本协议基于零知识验证的所有权认证方法具有完备的零知识验证特性. 证毕.

4.2 基于隐藏凭据的密钥传递安全性分析

定理 2. 如果对于随机密钥 DEK,任一攻击者在拟多项式时间 PPT 内,挑战猜测密钥信息 DEK 成功的概率是可以忽略不计,那么基于隐藏凭据的密钥传递是安全的.

证明.

为了能够更好地量化攻击者挑战成功的概率,我们需要补充定义符号:

t_1 和 t_2 分别是独立合法的挑战 f_1 和 f_2 的数目,n_1 和 n_2 分别是这些挑战中被拒绝的响应数,其中 $n_1\leqslant t_1$,$n_2\leqslant t_2$,f_1 表示离线挑战,f_2 表示在线挑战.

为了更好地简化算法,我们假设对于每一个 f_2 中的挑战,都是在每一个 f_1 中的挑战之前,并且 f_1 与 f_2 中的挑战相互独立,即如果 f_1 中的挑战失败,不会影响到 f_2 中的挑战结果. 并且,我们假设挑战 f_1 与 f_2 是与攻击者猜测得到的信息 DEK' 相关.

我们同时假设文件大摘要 $H_{\text{random}}(F)$ 是独立一致于字典 Dic 的,并且在攻击者角度上看来,原始的密钥信息 DEK 是一个在 $\{0,1\}^k$ 内不可区分的序列,攻击者除了知道原始的密钥信息 DEK 是集合 M 的一个子集之外,没有其他任何先验知识,其中 $M\subseteq\{0,1\}^k$,k 表示密钥信息 DEK 的长度.

基于隐藏凭据的密钥传递方法有可能遭受的攻击分为外部攻击和内部攻击,这 2 种攻击涉及到的参数有:安全参数 λ,$|D|$ 是文件大摘要 $H_{\text{random}}(F)$ 集合的长度,$negl[\lambda]$ 表示对于任意非零常数 c,存在一个有限非零数 L,对于任意 $\lambda>L$,都有 $negl[\lambda]<\lambda^{-c}$.

1) 外部攻击. 外部攻击者 A_{out} 可能获取到用户的部分数据,或者是在网络传输过程中截获到部分数据,模拟用户与云服务器进行密钥传递交互. 若外部攻击者 A_{out} 通过了密钥传递交互,获取到文件加密密钥 DEK,则外部攻击者 A_{out} 挑战成功.

假设外部攻击者 A_{out} 挑战成功的概率是 $Pr[A_{\text{out}}^{f_1,f_2}\text{ wins}]$,在基于隐藏凭据的密钥传递方法中,外部攻击者 A_{out} 攻击场景如下:

① 外部攻击者 A_{out} 模拟用户生成一个文件大摘要 $H_{\text{random}}(F)'$,然后向云服务器发起密钥传递请求,云服务器根据索引 $h(F)$ 向外部攻击者 A_{out} 发送随机数,然而由于攻击者 A_{out} 没有真正完整的 $H_{\text{random}}(F)$ 由算法的特性可知,攻击者 A_{out} 挑战成功的概率为

$$Pr[A_{\text{out}}^{f_1,f_2}\text{ wins}]\leqslant\frac{\min\{q,t_2\}}{|D|-n_1}+\frac{t_2}{2^k-n_1}+negl[\lambda];$$

② 外部攻击者 A_{out} 绕过文件大摘要 $H_{\text{random}}(F)$,直接对文件加密密钥 DEK 进行猜测,则攻击者 A_{out} 猜测得到 DEK,挑战成功的概率为

$$Pr[A_{\text{out}}^{f_1,f_2}\text{ wins}]\leqslant\frac{1}{2^k}.$$

综上,外部攻击者 A_{out} 挑战成功的概率如式(2)所示,所以基于隐藏凭据的密钥传递方法具有外部安全性.

$$Pr[A_{\text{out}}^{f_1,f_2}\text{ wins}]\leqslant \max\left(\frac{\min\{q,t_2\}}{|D|-n_1}+\frac{t_2}{2^k-n_1}+negl[\lambda],\frac{1}{2^k}\right). \quad (2)$$

2) 内部攻击. 内部攻击者 A_{in} 可能控制了诚实而好奇的服务器,由于好奇而非法访问用户数据,并发起离线字典攻击,恢复得到文件加密密钥 DEK,则内部攻击者 A_{in} 挑战成功.

假设内部攻击者 A_{in} 挑战成功的概率是 $Pr[A_{\text{in}}^{f_1,f_2}\text{ wins}]$,在基于隐藏凭据的密钥传递方法中,内部攻击者 A_{in} 攻击场景如下:

① 恶意好奇的云服务器对存储的密钥传递保护值发起字典攻击,生成密钥 DEK',利用密钥 DEK' 解密密文,由 HCR 的安全性可知,内部攻击者 A_{in} 在解密过程中获取到文件大摘要 $H_{\text{random}}(F)$ 信息的概率为

$$Pr[A_{\text{in}}^{f_1,f_2}\text{ wins}]\leqslant\frac{t_2}{|D|-n_1}+\frac{t_2}{2^k-n_1};$$

② 内部攻击者 A_{in} 模拟外部攻击者 A_{out} 行为,对文件大摘要 $H_{\text{random}}(F)$ 猜测攻击,由外部攻击者挑战成功的概率说明,即使一个外部攻击者 A_{out} 足够强大,本协议中基于隐藏凭据的密钥传递方法依然具有外部安全性.

综上,内部攻击者 A_{in} 挑战成功的概率为

$$Pr[A_{in}^{f_1,f_2}\text{ wins}]\leqslant\frac{t_2}{|D|-n_1}+\frac{t_2}{2^k-n_1},$$

所以基于隐藏凭据的密钥传递方法具有内部安全性.

在实际计算场景中,外部攻击者 A_{out} 会有更强的约束条件,参数 q 与复杂的中间人攻击相关联,并且相比于任何 σ 随机预言模型 Hash 函数计算而言,参数 q 非常小. 值得注意的是,基于隐藏凭据的密钥传递方法中参数 p 和 R 并没有参与到攻击者挑战成功的概率计算中. 由于参数 p 有可能会被外部攻击者 A_{out} 被动窃听截获,参数 R 有可能会在协议会话过程中被内部攻击者 A_{in} 获取,但服务器不会从用户协议中获得任何反馈信息,因此不会对攻击者挑战成功的概率计算有所影响.

综上,在诚实的云服务器场景中,本协议基于隐藏凭据的密钥传递方法是安全的. 证毕.

5 测试与结果分析

本节给出对本方案运行性能的评估,主要测试其在密文去重所有权认证与密钥传递过程中产生的通信开销.

5.1 测试方案与场景

为了使测试场景更贴近实际,在本方案中我们租用了位于青岛的阿里云服务器作为实验中服务器端,加上位于西安的用户客户端,共同完成文件所有权认证与密钥传递工作. 由百度地图测得西安—青岛图上距离为 1 058.6 km,如图 6 所示:

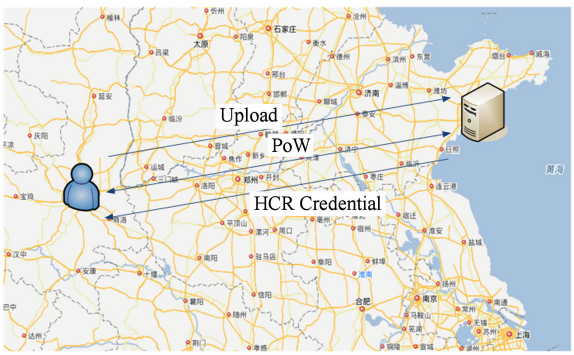

Fig. 6 The test plan diagram

图 6 测试方案示意图

本方案的实验设备为云服务器 1 台、用户客户端 2 台,实验设备具体参数如下:

搭建的服务器实验环境为阿里云服务器,部署于青岛,CPU 单核 3.00 GHz、内存 2 GB、操作系统为 Windows Server 2008 标准版 32 b、带宽 5 Mbps.

本地客户端位于西安,使用 Win7(32 b)操作系统、双核 Intel™ Core™ 2 Duo CPU E8400 3.00 GHz、内存 4 GB、500 GB 硬盘.

测试程序由 C++编写,其中 AES、MD5、零知识证明的指数运算部分用到了 cryptopp5.6.3 库[23].

5.2 首位上传者测试数据

针对不同大小文件,分别测试 1 MB,5 MB,10 MB,50 MB,100 MB,200 MB,300 MB,400 MB,500 MB,600 MB,800 MB,1 000 MB 文件,记录首位上传者计算文件 Hash 值、文件大摘要、文件加密、生成 HCR 凭据并将凭据与密文共同上传至服务器的各部分运行时间,实验数据结果如下.

生成大摘要时间开销如表 1 所示:

Table 1 Comparison of Calculation Time Between $H_{random}(F)$ and $h(F)$

表 1 生成大摘要与文件 Hash 值的时间对比

File Size/MB	Calculation Time/s	
	$H_{random}(F)$	$h(F)$
1	0.033 262	0.020 263
5	0.152 663	0.109 276
10	0.348 185	0.160 544
50	0.461 597	0.407 581
100	1.204 216	0.998 312
200	2.405 528	2.017 546
300	3.615 234	3.113 277
400	4.885 774	4.101 824
500	6.089 996	5.201 672
600	7.293 108	6.349 598
800	8.640 098	7.580 164
1 000	10.568 139	9.649 251

在本方案中,统一采用 AES-128 加密算法对文件数据加密,文件加密时间如表 2 所示:

Table 2 File Encryption Time

表 2 文件加密时间

File Size/MB	AES-128 Encryption Time/s
1	0.086 000
5	0.439 000
10	0.897 000
50	4.314 000
100	9.243 000
200	19.168 000
300	27.671 000
400	36.395 000
500	46.071 000
600	53.337 000
800	68.838 000
1 000	83.627 000

首位上传者利用 HCR 协议注册阶段对密钥进行保护,生成 HCR 凭据,并将凭据与密文一并上传至云服务器,总体时间关系如表 3 所示:

Table 3 First User HCR Protection and Upload Time

表 3 首位上传者 HCR 密钥保护及上传时间

File Size/MB	Time/s		
	HCR Credential Protection	HCR Upload with C_F	HCR Protection and Upload
1	0.054 512	1.71	1.764 512
5	0.171 898	3.65	3.821 898
10	0.358 021	5.57	5.928 021
50	0.515 663	10.49	11.005 663
100	1.242 185	17.86	19.102 185
200	2.605 579	34.44	37.045 579
300	3.816 562	54.09	57.906 562
400	5.085 716	66.95	72.035 716
500	6.201 399	78.14	84.341 399
600	7.381 247	96.10	103.481 247
800	8.709 573	122.24	130.949 573
1 000	10.534 681	166.87	177.404 681

5.3 后续上传者测试数据

针对不同大小文件,分别测试 1 MB,5 MB,10 MB,50 MB,100 MB,200 MB,300 MB,400 MB,500 MB,600 MB,800 MB,1 000 MB 文件,记录后续上传者利用零知识验证方法实现所有权认证,并与利用 MHT(Merkel-Hash tree)方法实现所有权认证的时间进行比较,实验数据结果如表 4 所示:

Table 4 Comparison of the Total Time Between Zero-Knowledge Proof and MHT

表 4 零知识验证与 MHT 的总时间对比

File Size/MB	Total Time/s	
	Zero-Knowledge Proof	MHT
1	0.192 225	0.029 225
5	0.649 492	0.115 362
10	1.148 233	0.214 823
50	4.782 209	0.810 326
100	6.920 186	1.570 733
200	10.229 651	3.115 476
300	13.536 723	7.235 495
400	19.332 291	12.813 769
500	26.182 607	19.019 434
600	32.032 861	24.703 428
800	43.648 090	31.518 936
1 000	55.007 976	42.019 975

5.4 文件拥有者测试数据

针对不同大小文件,分别测试 1 MB,5 MB,10 MB,50 MB,100 MB,200 MB,300 MB,400 MB,500 MB,600 MB,800 MB,1 000 MB 文件,记录通过所有权认证的文件拥有者利用 HCR 协议传递阶段恢复文件密钥,针对不同大小文件,HCR 传递阶段测试数据如表 5 所示:

Table 5 File Owner HCR Transmission Time

表 5 文件拥有者 HCR 传递时间

File Size/MB	Transmission Time/s					
	Server Sends v,h,r	User Calculates U	User Uploads U	Server Calculates B	Server Sends B	HCR Total Transmission
1	1.062	0.085 279	1.98	0.061 186	1.095	4.283 465
5	1.081	0.209 121	2.05	0.160 072	1.039	4.539 193
10	1.069	0.491 827	2.00	0.335 794	1.108	5.004 621
50	1.072	0.972 529	3.07	0.785 482	1.216	7.116 011
100	1.088	1.359 395	3.47	1.241 139	1.114	8.272 534
200	1.086	2.633 672	2.88	2.308 871	1.118	10.026 543
300	1.079	3.933 106	3.09	3.758 021	1.073	12.933 127
400	1.086	5.112 433	3.44	5.115 763	1.081	15.835 196
500	1.092	6.201 392	3.35	6.081 074	1.107	17.831 466
600	1.077	7.192 108	3.12	6.910 035	1.096	19.395 143
800	1.101	9.227 497	3.91	8.604 908	1.103	23.946 405
1 000	1.094	11.162 379	3.77	10.561 893	1.134	27.722 272

5.5 性能分析

针对不同大小文件，分别分析 1 MB，5 MB，10 MB，50 MB，100 MB，200 MB，300 MB，400 MB，500 MB，600 MB，800 MB，1 000 MB 文件测试数据. 首位上传者计算大摘要的时间与文件大小相关，测试的 12 组数据如图 7 所示. 由表 1 及图 7 可知. 计算大摘要的时间与计算文件 Hash 值的时间相比，时间性能上差别不大. 但计算的大摘要不损失文件熵值，可以利用大摘要作为后续的所有权认证凭据，而直接计算的文件 Hash 值只能作为文件标识，不能无损地代表文件本身.

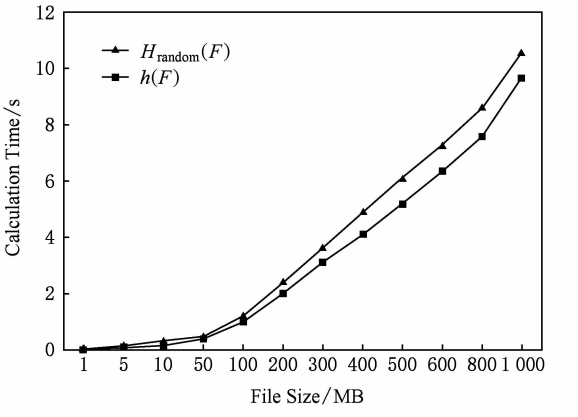

Fig. 7 Compare between $H_{random}(F)$ and $h(F)$

图 7 生成大摘要时间与文件 Hash 值时间对比图

首位上传者对文件数据进行加密，由表 2 及图 8 可知，随着文件大小的增加，文件加密时间也在增量变化，但文件加密这一操作，仅需要首位上传者实现，对于同一明文文件，仅需要一次文件加密上传，因此这一步骤对整体协议方案影响较小.

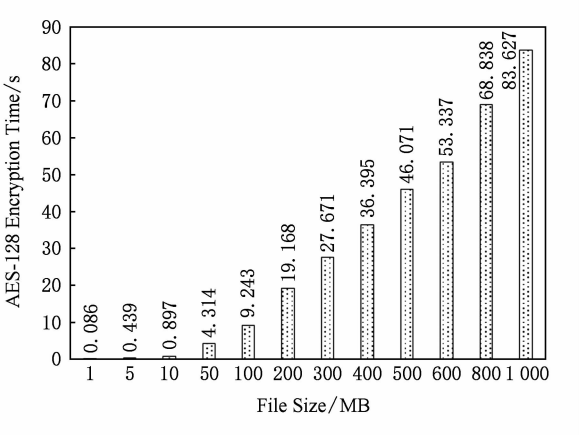

Fig. 8 File encryption time

图 8 文件加密时间图

由表 3、表 5 及表 6 可知，本方案中首位上传者利用 HCR 协议注册阶段对密钥进行保护，生成 HCR 凭据，通过所有权认证的文件拥有者利用 HCR 协议传递阶段恢复文件密钥，整体传递恢复时间与云服务器带宽有一定关系. 文献[18]中使用代理重加密方法实现首位上传者密钥传递至后续通过所有权认证的文件拥有者，这一过程由于第三方代理服务器参与，整体密钥传输时间受第三方服务器影响较大. 由图 9 比较 HCR 协议整体运算时间和代理重加密整体运算时间可知，本方案密钥传递方法相较于代理重加密密钥传递方法性能更优.

Table 6 Comparison of Transmission Time Between HCR and Proxy Re-encryption

表 6 HCR 与代理重加密密钥传递时间对比

File Size/MB	Transmission Time/s					
	HCR Credential	HCR Transmission	Total HCR	Rekey	Key Share	Total Proxy Re-encryption
1	0.054512	4.283465	4.337977	0.746	0.92004	1.66604
5	0.171898	4.539193	4.711091	0.749	3.22991	3.97891
10	0.358021	5.004621	5.362642	0.744	6.85279	7.59679
50	0.515663	7.116011	7.631674	0.752	22.54627	23.29827
100	1.242185	8.272534	9.514719	0.749	46.38131	47.13031
200	2.605579	10.026543	12.632122	0.753	94.64308	95.39608
300	3.816562	12.933127	16.749689	0.746	125.44356	126.18956
400	5.085716	15.835196	20.920912	0.743	169.84565	170.58865
500	6.201399	17.100117	23.301516	0.749	208.37324	209.12224
600	7.381247	19.395143	26.77639	0.751	252.90814	253.65914
800	8.709573	23.946405	32.655978	0.744	337.01452	337.75852
1 000	10.534681	27.722272	38.256953	0.749	418.29714	419.04614

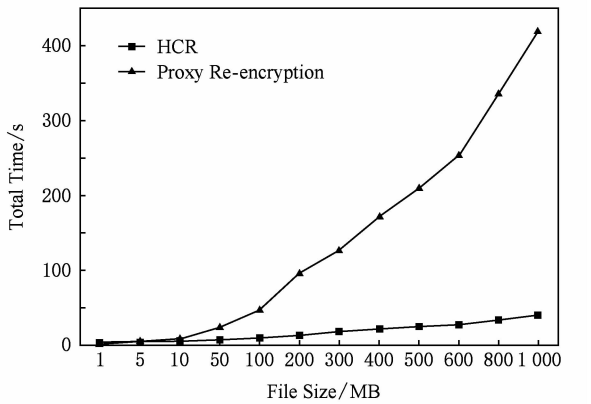

Fig. 9 Compare between HCR and proxy re-encryption

图 9 HCR 与代理重加密总时间对比图

本方案中后续上传者利用零知识验证方法实现所有权认证,由表 4 及图 10 可知,针对不同大小的文件,本方案与采用 MHT 方法实现所有权认证时间相差较小,相较于整体文件上传及下载时间而言,时间差影响可忽略,有效实现文件所有权认证,完成去重工作.

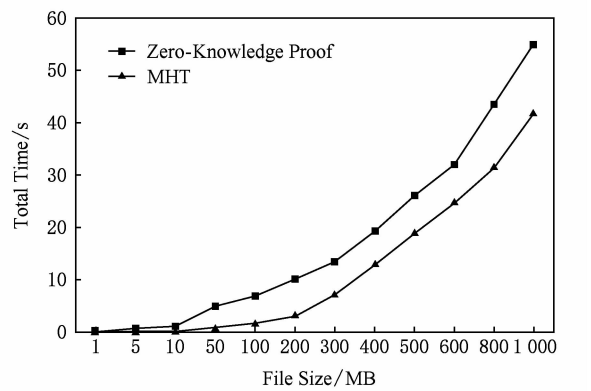

Fig. 10 Compare between zero-knowledge proof and MHT

图 10 零知识验证与 MHT 时间对比图

综合本节上述性能分析,以 1 000 MB 文件为例,对比文献[9]、文献[18]及本文方案,其中文献[9]是当前典型的利用收敛加密及 MHT 方法实现密文去重的方案,文献[18]是当前典型的利用椭圆曲线加密及代理重加密方法实现密文去重及密钥传递的方案.假设用户均采用 AES-128 加密算法,并且实验环境网络条件相同,那么对于相同的密文文件上传时间相同.在本实验环境中,1 000 MB 文件平均加密时间是 83.63 s,密文文件上传至云服务器的平均时间是 166.82 s,共计 250.45 s,对于这部分时间,3 种方案取相同值,不再进行讨论.对于 1 000 MB 文件,统计 3 种方案中首位上传者与后续上传者平均计算时间,本方案中首位上传者统计生成文件 Hash 值、文件大摘要值及 HCR 凭据值时间,后续上传值统计零知识验证时间,并计算首位上传者及后续上传者运算总时间和,对比数据结果如表 7 所示:

Table 7 Compare Among 3 Protocols in 1 000 MB File

表 7 1 000 MB 文件 3 种协议时间对比 s

Protocol	First User	Subsequent User	Total Time
Ref[9]	36.071	42.019	78.090
Ref[18]	150.400	4.910	155.310
Ours	30.751	55.007	85.758

由表 7 及图 11 可知,本文方案与文献[9]中首位上传者及后续上传者计算总时间相差较小,远小于文献[18]中计算总时间.由测试数据可知,首位上传者及后续上传者计算总时间远远小于加密文件并上传至云服务器时间,由于文件加密并上传总时间与文件大小成正比,那么采用客户端密文去重方案可以大大减少文件重复加密上传时间,提高云服务器存储利用率,具有良好的应用价值.

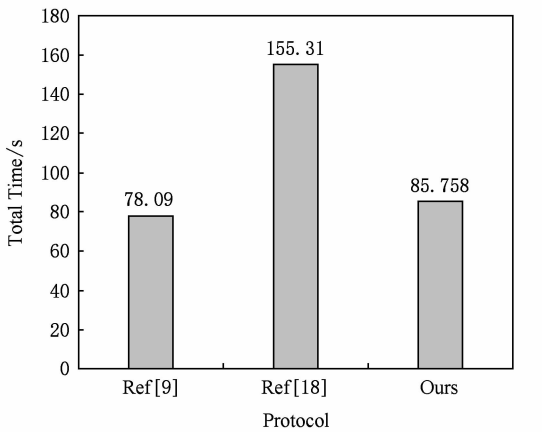

Fig. 11 Compare among 3 protocols in 1 000 MB file

图 11 1 000 MB 文件 3 种协议总时间对比图

6 总 结

本文提出了一种新的密文去重场景下所有权认证与密钥传递方法.利用独立成对 Hash 方法生成不损失熵的文件大摘要,同时利用零知识验证方法完成文件所有权认证,验证过程具有零知识性,保护数据隐私,提高文件所有权认证过程的安全性.利用隐藏凭据恢复方法,加密密钥与文件无关,同时隐藏凭据恢复方法可以建立在服务器不可信的两方密钥传递过程中,不需要第三方服务器参与,在密钥传递过程中不会泄露密钥信息,保证密钥传递的安全性.

最后通过安全性分析理论证明本方案所有权认证及密钥传递达到了可证明的安全强度,实际云平台的测试数据表明,本方案时间效率良好,减少了密文去重的运算量,使用户可以更方便高效地使用云服务,具有很高的应用价值.

参 考 文 献

[1] Opendedup. In-line deduplication to a cloud storage backend—SDFS can send all of its data to AWS, AZURE, or Google [OL]. [2016-01-06]. http://opendedup.org

[2] Meyer D, Bolosky W. A study of practical deduplication [J]. ACM Trans on Storage, 2012, 7(4): 1-20

[3] Tsai Chunwei, Lai Chinfeng, Chao Hanche, et al. Big data analytics: A survey [J]. Journal of Big Data, 2015, 2(1): 1-32

[4] Wei Lifei, Zhu Haojin, Cao Zhenfu, et al. Security and privacy for storage and computation in cloud computing [J]. Information Science an International Journal, 2014, 258(3): 371-386

[5] Wen Mi, Lu Rongxing, Zhang Kuan, et al. PaRQ: A privacy-preserving range query scheme over encrypted metering data for smart grid [J]. IEEE Trans on Emerging Topics in Computing, 2013, 1(1): 178-191

[6] Wen Mi, Lu Kejie, Lei Jingsheng, et al. BDO-SD: An efficient scheme for big data outsourcing with secure deduplication [C] //Proc of the 22nd IEEE Conf on Computer Communications Workshops. Piscataway, NJ: IEEE, 2015: 214-219

[7] Halevi S, Harnik D, Pinkas B, et al. Proofs of ownership in remote storage systems [C] //Proc of the 18th ACM Conf on Computer and Communications Security. New York: ACM, 2011: 491-500

[8] Douceur J, Theimer M, Bolosky W. Encryption systems and methods for identifying and coalescing identical objects encrypted with different keys [OL]. [2007-09-04]. http://www.freepatentonline.com

[9] Xu Jia, Chang Ee-Chien, Zhou Jianying. Weak leakage-resilient clientside deduplication of encrypted data in cloud storage [C] //Proc of the 8th ACM SIGSAC Symp on Information, Computer and Communications Security. New York: ACM, 2013: 195-206

[10] Xu Jia, Zhou Jianying, Chang Eechien. Leakage-resilient client-side deduplication of encrypted data in cloud storage [OL]. 2011 [2012-05-01]. http://eprint.iacr.org

[11] Bellare M. Message-locked encryption and secure deduplication [G] //LNCS 7881: Proc of the 32nd Annual Int Conf on the Theory and Applications of Cryptographic Techniques. Berlin: Springer, 2013: 296-312

[12] Bellare M, Keelveedhi S, Ristenpart T. DupLESS: Server-aided encryption for deduplicated storage [C] //Proc of the 22nd USENIX Security Symp. Berkeley, CA: USENIX Association, 2013: 179-194

[13] Wee Keongng, Wen Yonggang, Zhu Huafei. Private data deduplication protocols in cloud storage [C] //Proc of the 27th Annual ACM Symp on Applied Computing. New York: ACM, 2012: 172-191

[14] Li Jin, Chen Xiaofeng, Li Mingqiang, et al. Secure deduplication with efficient and reliable convergent key management [J]. IEEE Trans on Parallel and Distributed Systems, 2014, 25(6): 1615-1625

[15] Huang Wentao, Langberg M, Kliewer J, et al. Communication efficient secret sharing [J]. IEEE Trans on Information Theory, 2016, 62(12): 7195-7206

[16] Li Jin, Chen Xiaofeng, Huang Xinyi, et al. Secure distributed deduplication systems with improved reliability [J]. IEEE Trans on Computers, 2015, 64(12): 3569-3579

[17] Chen Rongmao, Mu Yi, Yang Guomin, et al. BL-MLE: Block-level message-locked encryption for secure large file deduplication [J]. IEEE Trans on Information Forensics and Security, 2015, 10(12): 2643-2652

[18] Yan Zheng, Ding Wenxiu, Yu Xixun, et al. Deduplication on encrypted big data in cloud [J]. IEEE Trans on Big Data, 2016, 2(2): 138-150

[19] Boyen X. Hidden credential retrieval from a reusable password [C] //Proc of the 4th Conf on Computer and Communications Security. New York: ACM, 2009: 228-238

[20] Alexandra B. Threshold signatures, multisignatures and blind signatures based on the gap-Diffie-Hellman-group signature scheme [G] //LNCS 2567: Proc of the 6th Advances in Public Key Cryptography (PKC2003). Berlin: Springer, 2003: 31-46

[21] Cash D, Kupcu A, Wichs D. Dynamic proofs of retrievability via oblivious RAM [J]. Journal of Cryptology, 2017, 30(1): 22-57

[22] Barak B, Dodis Y, Krawczyk H, et al. Leftover Hash Lemma, revisited [C] //Proc of the 31st Int Cryptology Conf. New York: ACM, 2011: 1-20

[23] Wei Dai. Crypto++5.6.3 [OL]. [2015-11-20]. https://www.cryptopp.com

He Simeng, born in 1993. Master candidate in Xidian University. Her main research interests include cloud computing and storage security.

Yang Chao, born in 1979. Received his PhD degree in computer science and technology from Xidian University in 2008. Professor in Xidian University. His main research interests include cryptography and information & network security.

Jiang Qi, born in 1983. Received his PhD degree in computer science and technology from Xidian University in 2011. His main research interests include security protocols and wireless network security.

Yang Li, born in 1977. Received his PhD degree in computer science and technology from Xidian University in 2009. His main research interests include security protocols and wireless network security.

Ma Jianfeng, born in 1963. Professor and PhD supervisor in Xidian University. His main research interests include network and information security, cryptography.

2018 年《计算机研究与发展》专题(正刊)征文通知

——分布式安全与区块链技术研究

计算机网络已经渗透于万物之中,我们正处在一个万物互联的时代,无人驾驶汽车、家庭机器人以及家庭安保系统等各种网络、智能设备,都需要分布式网络系统的支撑.但是,随着分布式网络给人们带来越来越多便利的同时,安全性威胁也越来越严重.区块链正在以让人们始料未及的速度和规模迅速发展,其去中心化、可验证、防篡改的特性以及进行价值传递的功能,也许会引起人们社会生活的深刻变革,改变社会某些领域的管理模式.区块链是以密码技术构造的新型信息系统体系,从设计到运行都涉及大量密码学与信息安全问题.

为推动我国学者在分布式安全及区块链技术领域的研究,促进信息安全在基础理论、重要应用领域、新兴安全技术的发展,及时报道我国学者的最新研究成果,《计算机研究与发展》将于 2018 年 10 月出版网络与信息安全专题——分布式安全及区块链技术研究,主要聚焦分布式安全及智能安全、区块链技术等领域,欢迎相关领域的专家学者和科研人员踊跃投稿.

征文内容

本专辑包括(但不限于)下列主题:

1) 分布式密码算法与协议;
2) 分布式安全理论与智能安全;
3) 分布式网络攻防;
4) 物联网安全;
5) 电子货币与区块链技术;
6) 安全多方计算与隐私保护.

投稿要求

1) 论文应属于作者的科研成果,数据真实可靠,具有重要的学术价值或推广应用价值,未在国内外公开发行的刊物或会议上发表过,不存在一稿多投问题.作者在投稿时,需向编辑部提交版权转让协议.
2) 论文应包括题目、作者信息、摘要、关键词、正文和参考文献,论文一律用 Word 排版,论文格式体例格式请参考《计算机研究与发展》近期文章.
3) 论文需附通信作者的联系地址、电话或手机及 E-mail 地址.
4) 论文请通过期刊网站(http://crad.ict.ac.cn)进行投稿,并在作者留言中注明"网络与信息安全 2018 专题"(否则按自由来搞处理).

重要日期

征文截止日期: 2018 年 6 月 11 日
录用通知日期: 2018 年 7 月 20 日
作者修改稿提交日期:2018 年 8 月 6 日
出版日期:2018 年 10 月

专题特邀编委

徐秋亮 教授 山东大学 xql@sdu.edu.cn
张玉清 教授 中国科学院大学、国家计算机网络入侵防范中心 zhangyq@ucas.ac.cn
董晓蕾 教授 华东师范大学 dongxiaolei@sei.ecnu.edu.cn

联系方式

联 系 人:徐秋亮 xql@sdu.edu.cn
编 辑 部:crad@ict.ac.cn, 010-62620696, 010-62600350
通信地址:北京 2704 信箱《计算机研究与发展》编辑部
邮政编码:100190