计算机研究与发展
Journal of Computer Research and Development
DOI:10.7544/issn1000-1239.2019.20190351
56(10): 2170-2182, 2019

云环境下支持可更新加密的分布式数据编码存储方案

严新成[1] 陈 越[1] 巴 阳[1] 贾洪勇[2] 朱 彧[1]
[1](战略支援部队信息工程大学 郑州 450001)
[2](郑州大学软件与应用科技学院 郑州 450001)
(imtodshine@163.com)

Distributed Data Encoding Storage Scheme Supporting Updatable Encryption in Cloud

Yan Xincheng[1], Chen Yue[1], Ba Yang[1], Jia Hongyong[2], and Zhu Yu[1]
[1](*Strategic Support Force Information Engineering University*, *Zhengzhou* 450001)
[2](*School of Software and Applied Technology*, *Zhengzhou University*, *Zhengzhou* 450001)

Abstract Due to the long-term immutability of the ciphertext stored in the cloud, key compromise becomes an important factor affecting the security of stored data. Data re-encryption is an effective way to deal with key leakage, but the corresponding computational overhead and communication overhead of data uploading and downloading increase the burden on users and storage systems. In addition, for data storage based on distributed coding, ciphertext update needs to be performed on the basis of decrypting ciphertext, and the ciphertext merging also increases the communication and computational overhead of the system. Aiming at the above problems, a distributed data encoding storage scheme supporting updatable encryption (DDES-UE) in cloud environment is proposed. By constructing the updatable encryption scheme with key homomorphic pseudo-random functions, the heavy calculation and communication overhead of ciphertext update can be avoided; ciphertext segmentation and improved functional minimum storage regenerated code (FMSR) are used for achieving distributed data storage, which ensures high availability for storage data and direct data update of each storage node. Security proofs and performance analysis show that the proposed scheme can support secure and efficient data recoverability in the case of node corruption and the integrity verification of decrypted data while guaranteeing the security of data storage. Compared with traditional data re-encryption, DDES-UE can avoid the computation and communication overhead for data re-encryption, uploading, downloading, decoding, and ciphertext merging as well, which is of great significance for building secure and efficient cloud storage system with direct data update. In addition, the periodic key update can effectively increase the time cost for an attacker to crack the ciphertext by acquiring the key, which also enhance the active security defense capability of the system.

Key words distributed cloud storage; key compromise; updatable encryption; FMSR encoding; periodic key update

摘 要 由于云存储密文的静态性特征,密钥泄露成为影响存储数据安全性的重要因素.数据重加密是应对密钥泄露的有效手段,但相应的计算开销以及上传下载的通信开销增加了用户和存储系统的负担.

收稿日期:2019-06-10;**修回日期**:2019-08-05
基金项目:国家自然科学基金项目(61702549);河南省科技攻关计划基金项目(172102210017)
This work was supported by the National Natural Science Foundation of China (61702549) and the Key Technologies Research and Development Program of Henan Province (172102210017).
通信作者:巴阳(zzu_bayang@163.com)

此外,对基于分布式编码的数据存储而言,密文更新需要在解密密文的基础上进行,密文合并过程同样增加了系统的通信及计算开销.针对上述问题,提出一种云环境下支持可更新加密的分布式数据编码存储方案(distributed data encoding storage scheme supporting updatable encryption, DDES-UE).通过利用密钥同态伪随机函数构造可更新加密方案,可避免密文更新的计算与通信开销过大问题;基于密文分割与改进 FMSR 编码实现数据分布式存储,保证存储数据的高可用性和各存储节点的直接数据更新.安全性证明及性能分析表明:提出的方案在保证数据存储安全性的同时,能够支持部分存储节点损坏时安全高效的数据可恢复性以及解密数据的完整性验证.与传统的数据重加密相比,DDES-UE 能够避免数据重加密及数据上传、下载、解码、合并带来的计算和通信开销,对于构建支持直接数据更新的安全高效云存储系统有重要意义.此外,周期性密钥更新可有效增加攻击者通过获取密钥破解密文的时间成本,从而增强了系统的主动安全防御能力.

关键词 分布式云存储;密钥泄露;可更新加密;FMSR 编码;周期性密钥更新

中图法分类号 TP309

随着大数据与云计算技术的发展,越来越多的数据可以在云端进行存储、计算以及共享.数据加密有效保证了存储与通信的机密性.而根据美国国家标准与技术研究院(National Institute of Standards and Technology, NIST)发布的 Special Publication[1] SP 800-57,基于密码学的密文密钥具有严格的生命周期,保留时间越长,密钥泄露风险越大.

我们知道,在加解密服务中,非对称加密主要实现的是对称密钥的保护和共享,也就是对实际存储数据的访问权限的一种控制.就存储的用户敏感数据而言,对称加密提供的是基本的安全性保障.云服务器的存储数据资源丰富,其分为"冷数据"和"热数据",不同类型的数据其价值相差甚大.对于云中长期存储的数据而言,尤其是重要性较高的敏感密文数据,由于密文的硬破解是困难的[2],因此密钥存储与使用的安全性直接影响了密文存储的安全性.现代密码学都假定用户密钥对可能的攻击来说是完全隐藏的,但在实际中通过边信道攻击,例如时间攻击、电源耗损、冷启动攻击以及频谱分析等,都能从保密密钥或加密系统内部获得有关密钥的部分信息[3],极大损害了存储密文的安全性.因此,对于长期存储的加密敏感数据,为防止用户私钥泄露,需要周期性地轮换密钥或根据实际情况撤销部分用户访问权限.

比如 NIST[4] 建议定期轮换密钥,开放式 Web 应用程序安全项目 OWASP[5] 也是如此,支付卡行业[6] 定期要求轮换客户数据.谷歌[7] 和亚马逊[8] 现在为其长期存储服务中的此类操作提供部分支持,因此被授权密钥更新的客户可以这样做.然而,正如 Everspaugh 等人[9] 所指出的那样,Google 和 Amazon 所使用的技术存在可疑的和未定义的安全性,且容易受到密钥搜索攻击(key-scraping attacks).因此,密钥更新是实践中的难题之一.

此外,虽然更新共享文件很常见,但很少关注如何有效地更新分布式存储系统中的大量加密的文件.事实上,要修改文件中的任何一个比特位,当前的解决方案需要通过简单且昂贵的方式下载和解密文件[10].数据重加密是实现密文更新的简单方式,但是极大增加了客户端的计算开销以及存储系统的通信开销,同时客户端需要更大的缓存空间来存储从服务器端返回的额外数据块.混合加密虽然实现了用户私钥(密钥加密密钥)的更新,但内部的数据加密密钥并未更新.对于之前能访问该数据密钥的被撤销权限的用户而言,数据的存储仍存在安全风险.我们知道,数据的存储结构包括 2 类:集中式存储和分布式存储.集中式存储使得数据在管理上较为直观方便,但难以满足数据灾备的需求.而将数据重加密或混合加密直接应用到分布式数据存储时,通信及计算开销过大:上一个版本的密文块更新需要经历从各个数据节点"下载-还原-解密-重加密-重新分块上传"的过程,客户端计算开销及系统的通信开销均可能成为制约系统运行效率的关键[11].因此,就云存储的实际应用而言,需要同时考虑数据的机密性、完整性、可用性及单节点的直接更新,这对支持加密的分布式数据存储提出了更高的要求,即能够在各数据节点存储的编码后的密文数据的基础上直接进行数据更新,从而避免数据上传下载带来的通信开销以及还原密文和重加密带来的计算开销,为进一步构建安全实用的加密云存储系统提供有效的技术支撑.

1 相关工作

从当前研究来看,密钥更新面临的最大挑战仍然是密文更新的计算开销问题.本文整理并归纳了近几年关于可更新加密的研究工作,并结合分布式云存储及数据安全更新的场景分析可更新加密在实际应用中的问题与挑战.

云计算技术的大规模应用使得存储数据的隐私保护问题日益凸显,加密环境下密钥泄露的威胁场景也无处不在.Sahai 等人[12]在 CRYPTO 2012 上提出这样一个担心:在云存储系统中,当用户证书变化时,必须同时考虑私钥的撤销以及密文的更新,使得被撤销权限的用户无法访问之前可解密的数据,而合法用户不受影响;Sahai 等人定义了可撤销存储,基于密文代理技术,利用公开信息直接对给定策略加密的密文进行"重加密",使得该密文转换为原明文信息对应的更具有约束性策略的新的密文,保证新加密的数据无法被已撤销私钥的用户访问;通过对 Lewko 等人[13]的方案进行改进,分别构建了支持 KPABE(key-policy attribute-based encryption)和 CPABE(ciphertext-policy attribute-based encryption)的解决方案,均满足上述定义的"分段密钥产生"特性,并在标准模型下证明是安全的.

自更新加密(self-updatable encryption, SUE)由 Lee 等人[14]在 ASIACRYPT 2013 提出,是一个新开发的密码学原语,它实现了作为内置功能的密文更新,从而提高了云管理中密钥撤销和时间演进的效率.在 SUE 中,当且仅当用户拥有与密文的时间相同或未来某个时间对应的私钥时,用户可以解密与特定时间相关联的密文.此外,可以只使用公共信息将附加到某个时间的密文更新为附加到未来时间的新密文.Datta 等人[15]在标准假设下给出了素数阶双线性群中的第 1 个完全安全的 SUE 方案,即决策线性假设和决策双线性 Diffie-Hellman 假设;但 Freeman[16]和 Lewko[17]指出,素数阶双线性群的通信和存储以及计算效率比具有相当安全级别的复合阶双线性群要高得多.因此,文献[15]提出的 SUE 方案比现有的完全安全的 SUE 具有高度的成本效益.Lee[18]在素数阶双线性群中提出了一种 SUE 方案,该方案公共参数较短,并将其先前的方案扩展到支持密文时间间隔的 TI-SUE(time-interval SUE)方案;Aono 等人[19]提出了密钥可轮换和安全可更新同态加密的概念,其中任何密文密钥都可以轮换和更新,同时仍然保持底层明文的完整和未透露;Lee 等人[20]提出了第 1 个 SUE 和 RS-ABE 方案来解决新的威胁,即被撤销权限的用户仍然可以访问由存储服务器提供的过去的密文.

Rodriguez 等人[21]考虑使用计数器(counter, CTR)模式加密来保护数据安全.CTR 由 Diffie 和 Hellman[22]提出,并在 2001 年被 NIST 通过,作为加密的安全操作码[23].有关如何生成这些"计数器"的规范可以在 NIST 发布的标准中找到.但是,他们提出的方案缺乏用户私钥的定义和标准的安全证明.

然而,上述可更新数据加密方案并未考虑实际云存储中的应用.如引言所述,集中式存储难以满足有效的数据灾备需要.也就是说,构建支持可更新加密的分布式数据存储方案可以满足更高的安全性和可靠性要求.分布式存储架构是用于在授权用户之间存储和共享大数据的云计算系统的最重要组件之一.分布式存储系统通过分散在单个不可靠节点上的冗余提供对数据的可靠访问,应用场景包括数据中心、点对点存储系统和无线网络中的存储[24].现代分布式存储系统将冗余编码技术应用于存储数据,能够解决集中式存储的单点失效问题以及避免多副本存储的巨大开销;Hu 等人[25]研究了功能最小存储再生码 FMSR,它可以在没有幸存节点的编码要求的情况下实现无编码修复,同时保留最小的修复带宽保证并最小化磁盘读取;Rawat 等人[26]研究分布式存储系统的安全性和本地可修复性,并重点介绍能够实现最佳局部修复的编码方案;Li 等人[27]提出了一种通用转换以实现分布式存储系统中 MDS 编码的最佳修复;Su[28]引入了一种称为柔韧 FR 编码的新型 FR 编码,它可以解决现有 FR 编码不够灵活,无法充分适应分布式存储系统变化的缺点;唐英杰等人[29]针对基于纠删码的分布式存储中数据恢复开销过大问题,提出一种基于网络计算的高效故障重建方案来减少网络流量,有效解决了客户端的网络瓶颈.

2 预备知识

2.1 密钥同态伪随机函数

考虑一个高效可计算的函数 $R:\mathcal{K}\times\mathcal{X}\to G$,其中 $(\mathcal{K},\oplus)$ 和 $(G,\otimes)$ 都是群.我们称 $(R,\oplus,\otimes)$ 是密钥同态的伪随机函数(key homomorphic pseudorandom functions, KHPRF),如果函数 R 是安全的,并且对于每个 $k_1,k_2\in\mathcal{K}$ 以及元素 $x\in\mathcal{X}$,都有 $R(k_1,x)\otimes$

$R(k_2,x)=R(k_1\oplus k_2,x)$成立.函数 R 的安全性定义如下:

定义 $PRF1_R^{\mathcal{A}}$ 为随机选择一个密钥 $k\in\mathcal{K}$ 然后运行攻击者 $\mathcal{A}$ 的游戏,该敌手可以自适应地查询谕言,返回应用于查询消息的 R_k.敌手输出一个比特位.在游戏 $PRF0_R^{\mathcal{A}}$ 中,敌手 $\mathcal{A}$ 可以自适应地查询谕言并返回一个从 G 中随机抽取的值.同样敌手输出一个比特.我们假设在任一游戏中 $\mathcal{A}$ 都不会向其谕言查询 2 次相同的值.我们定义敌手 $\mathcal{A}$ 的 PRF 优势为 $Adv_R^{\mathrm{prf}}(\mathcal{A})=|pr[PRF1_R^{\mathcal{A}}\Rightarrow 1]-pr[PRF0_R^{\mathcal{A}}\Rightarrow 1]|$.

若 $Adv_R^{\mathrm{prf}}(\mathcal{A})$的值是可忽略的,则称函数 R 是安全的.随机预言模型中的安全密钥同态 PRF 的简单示例是函数 $R(k,x)=H(x)^k$,其中 G 是加法群,判定性 Diffie Hellman 假设成立.这种构建最初归功于 Naor,Pinkas 和 Reingold[30].

2.2 FMSR 码

(n,k)FMSR 码将大小为 M 的原始文件切分成 $k(n-k)$个固定大小的数据块,再将它们编码成 $n(n-k)$个编码块然后上传到 n 个数据节点,每个节点存储相邻的 $n-k$ 个编码块.数据读取时,首先从随机挑选的 k 个数据节点中下载 $k(n-k)$个编码块进行译码操作,然后将还原后的数据块合并成原始文件.

当某节点意外失效时,为保证数据的安全性和服务的连续性,数据修复过程如下:从正常工作的 $n-1$ 个数据节点上各取一个数据块重新进行编码,用生成的 $n-k$ 个编码块来替代失效节点存储的数据.(4,2)FMSR 码的修复过程如图 1 所示:

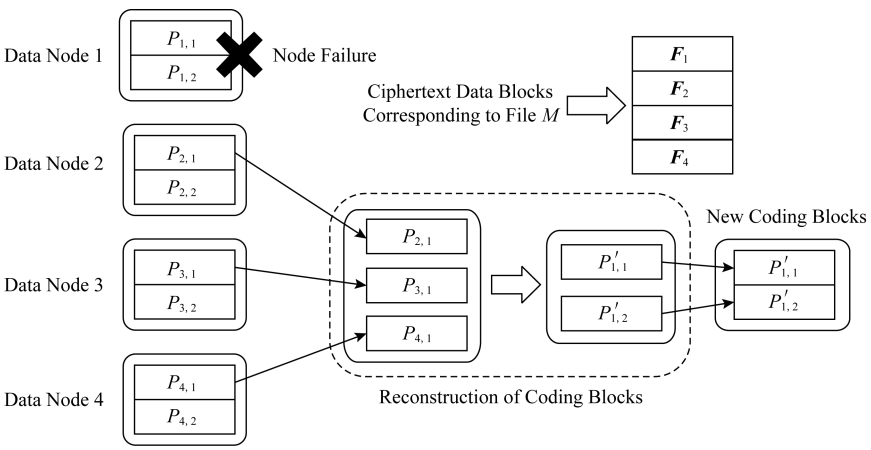


Fig. 1 Data recovery of (4,2)FMSR encoding for corrupted nodes

图 1 (4,2)FMSR 码的损坏节点数据恢复

2.3 DDH 假设

以元组(G,g,p)为输入,其中群 G 表示阶为 p 生成元为 g 的循环群,判定 Diffie-Hellman(DDH)问题关于 λ 是困难的,即 p 是 λ 位素数.更确切地说,如果对于任何高效的敌手 $\mathcal{A}$,都有概率:

$$|Pr[\mathcal{A}(G,p,g,g^a,g^b,g^{ab})]-Pr[\mathcal{A}(G,p,g,g^a,g^b,g^c)]|.$$

在 λ 中可忽略不计,则称群(G,g,p)满足 DDH 假设,其中概率超过 p,g 的随机选择,$a,b,c\in\mathbb{Z}_p$的随机选择以及 $\mathcal{A}$ 的硬币投掷.

3 DDES-UE 方案设计

3.1 系统架构

DDES-UE 的系统架构如图 2 所示.它主要分为 2 部分:客户端和云存储系统.客户端系统通过 Internet 连接到各种数据存储服务器(data storage server,DSS)和数据管理服务器(data management server,DMS).云存储系统由分布在网络中的一系列数据存储服务器和数据管理服务器组成,通过网络连接形成分布式存储系统.待存储的数据首先在客户端基于密钥同态伪随机函数进行分块加密,然后将分块

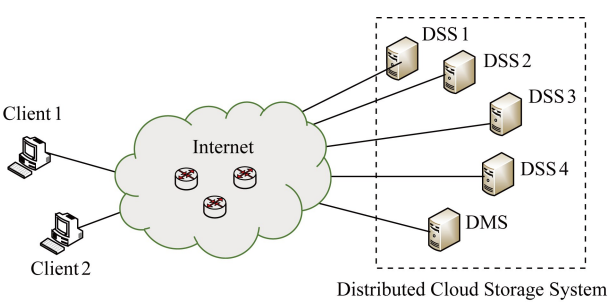


Fig. 2 System architecture of DDES-UE

图 2 DDES-UE 方案系统架构

密文进行编码,生成的编码数据块分布式存储在不同的数据节点(即数据存储服务器)中,关于密文的相关信息和密钥由数据管理服务器管理.具体的数据存储过程见 3.2.4 节.

3.2 方案构建

3.2.1 数据加密

在数据加密过程中使用密钥同态伪随机函数 $R:\mathcal{K}\times\mathcal{X}\rightarrow G$ 和 Hash 函数 $h:\{0,1\}^{*}\rightarrow G$.假设消息 M 可以由群 G 中的元素序列 $m_1,m_2,\cdots,m_l$ 表示.集合 $\mathcal{X}$ 中的元素都可以用整数来表示.注意 $\mathcal{KG}$ 和 $\mathcal{K}$ 虽然都产生参数,但是前者表示的是密钥生成算法,生成的是对称密钥 k;后者是群$(\mathcal{K},+)$,从中随机选取 2 个元素.客户端对数据进行加密的过程如下.

此外,图 3 给出了一个简单的数据加密和密文分块的例子,其中参数 $n=4,k=2,r=5$.数据加密前首先调用密钥产生算法 $\mathcal{KG}$ 生成一个对称密钥 k.数据加密过程的描述如算法 1 所示.

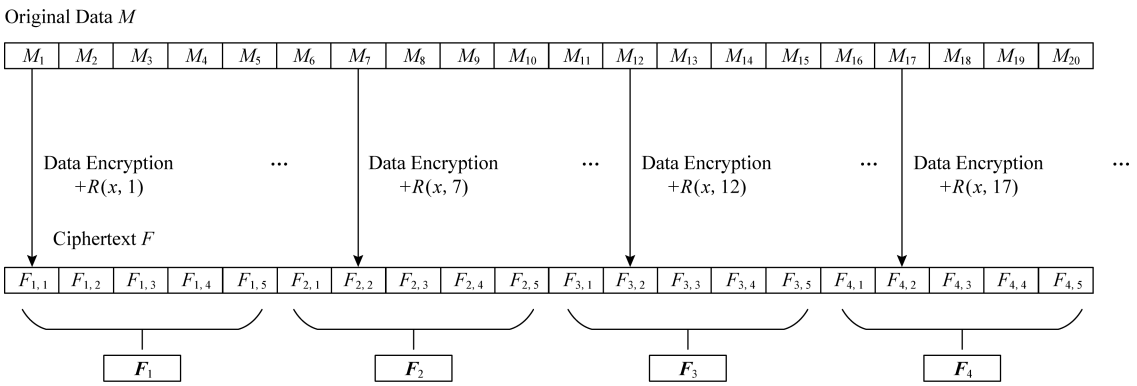


Fig. 3 Data encryption and ciphertext segmentation

图 3 数据加密与密文分割

算法 1. 数据加密算法.

输入:对称密钥 k、待加密的数据 m;

输出:密文头 $\widetilde{C}$,$\overline{C}=(y,\overline{C}_1,\cdots,\overline{C}_\eta)$.

① $x,y\leftarrow_{\$}\mathcal{K}$; /* 首先从群 $\mathcal{K}$ 选出 2 个重要参数 x,y */

② $\chi=x+y$;

③ $\tau=h(m)+R(x,0)$;

④ $\widetilde{C}=\varepsilon(k,(\chi,\tau))$; /* 使用上一阶段产生的对称密钥 k 来加密(χ,τ)生成密文头 $\widetilde{C}$ */

⑤ for $1\leqslant i\leqslant\eta$ /* 接下来计算密文体 $\overline{C}_i$ */

⑥ $\overline{C}_i=m_i+R(x,i)$;

⑦ end for

⑧ return $(\widetilde{C},\overline{C}=(y,\overline{C}_1,\cdots,\overline{C}_\eta))$.

最后,算法返回$(\widetilde{C},\overline{C}=(y,\overline{C}_1,\cdots,\overline{C}_\eta))$.其中密文头 $\widetilde{C}$ 由数据管理服务器保存.

3.2.2 数据分割

在数据分割阶段,客户端首先将加密后的文件(即密文体部分)切分成块.假设密文大小为 F,将之切分成 $k(n-k)$ 个固定大小的数据块(每个数据块称为 Macro-block),记作 $\boldsymbol{F}=(\overline{C}_1,\overline{C}_2,\cdots,\overline{C}_{k(n-k)})$,每个数据块 Macro-block 的大小为 $F/k(n-k)$,包含 r 个子数据块,记为 Mini-block(在阶为素数 p 的 Z_p 中).即对于 $i=1,2,\cdots,k(n-k)$,$\boldsymbol{F}_i=(F_i^1,F_i^2,\cdots,F_i^r)$.

在实际部署中,用户端可通过运行客户端程序与云存储系统进行交互,后者除了负责数据的上传和下载,还负责维护编码参数和加密密钥等信息.

3.2.3 数据编码

客户端对原始密文数据(即密文体 F)进行编码,得到 $n(n-k)\times r$ 个编码数据块,记作$(P_{ij})_{i=1,2,\cdots,n(n-k),j=1,2,\cdots,r}$,每个编码块都是 $k(n-k)$ 个初始密文数据块 Macro-blocks 所包含的所有子数据块 Mini-block 的线性组合.

具体的编码过程为:

1) 首先在客户端构造一个 $n(n-k)\times k(n-k)$ 的编码矩阵 $\boldsymbol{EM}=(\alpha_{i,j})$,其中元素 $\alpha_{i,j}$ 均是从有限域$GF(2^w)$中随机产生,$\boldsymbol{EM}$ 必须满足 MDS 性质.

2) 客户端使用过程 1)中产生的编码矩阵与初始密文块(即 $k(n-k)$ 个 Macro-block)进行乘法运算,得到 $n(n-k)\times r$ 个编码数据块.编码矩阵中每个行向量称之为一个编码向量,其形式为 $\boldsymbol{P}_i=(P_{i,1},P_{i,2},\cdots,P_{i,r})$,对应于一个编码块.对初始密文块的每个 Macro-block 进行编码后得到的编码块矩阵为

$$\boldsymbol{P}_{i,x}=\boldsymbol{ECV}_i\times\boldsymbol{F}=\sum_{j=1}^{k(n-k)}\alpha_{i,j}F_j^x, \tag{1}$$

其中,$i=1,2,\cdots,n(n-k)$,$x=1,2,\cdots,r$.

3.2.4 数据存储

客户端将 $n(n-k)\times r$ 个编码数据块上传给 n 个数据节点,每个节点存储相邻的 $(n-k)\times r$ 个数据块,编码矩阵由客户端持有.同时将密文头上传至数据管理服务器.

取 $n=4,k=2$ 时,单个文件的上传过程如图 4 所示.注意在编码块的产生过程中,我们以 Macro-block 为单位进行表述.

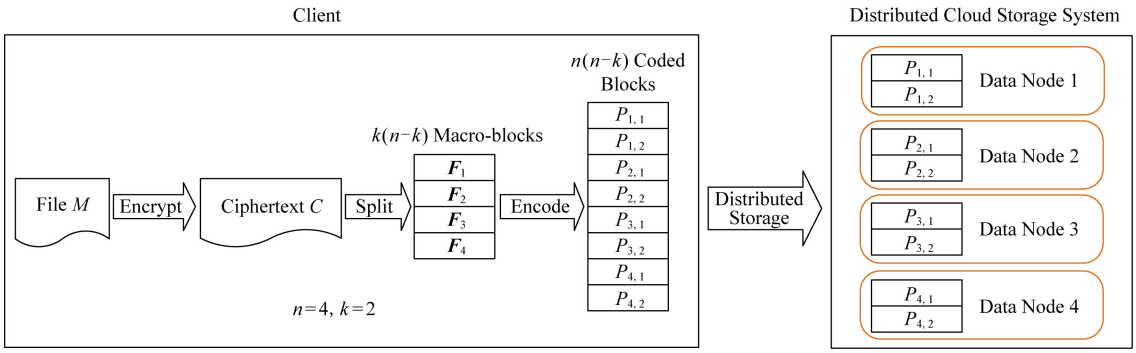


Fig. 4 Process of data chunking storage

图 4 数据分块存储过程

3.2.5 密文更新

数据更新包括 2 个阶段:更新令牌生成和密文重新加密.我们分别描述它们,然后在不同的分布式数据节点中给出编码块的独立更新过程.更新令牌生成过程的描述如算法 2 所示.

算法 2. 更新令牌产生算法.

输入:当前使用的对称密钥 k_i 及密文头 $\widetilde{C}$,以及需要更新至下个周期使用的对称密钥 k_j;

输出:更新令牌 $\Delta_{i,j,\bar{C}}$.

① $(\chi,\tau)=\mathcal{D}(k_i,\widetilde{C})$;/* 使用 k_i 解密当前使用的密文头 $\widetilde{C}$ */

② if $(\chi,\tau)=\perp$ then

③ return $\perp$;

④ end if

⑤ $x',y'\leftarrow_{\$}\mathcal{K}$;/* 从群 $\mathcal{K}$ 中随机选取 2 个新的参数 x',y' */

⑥ $\chi'=\chi+x'+y'$;

⑦ $\tau'=\tau+R(x',0)$;

⑧ $\widetilde{C}'\leftarrow_{\$}\varepsilon(k_j,(\chi',\tau'))$;/* 使用新的对称密钥 k_j 来加密(χ',τ')生成密文头 $\widetilde{C}'$ */

⑨ return $\Delta_{i,j,\bar{C}}=(\widetilde{C}',x'-x,y')$.

基于算法 2 生成的更新令牌,密文重加密过程 $ReEnc(\Delta_{i,j,\bar{C}},(\widetilde{C},\bar{C}))$描述如算法 3 所示:

算法 3. 密文重加密算法.

输入:当前版本的密文$(\widetilde{C},\bar{C})$,以及更新令牌 $\Delta_{i,j,\bar{C}}$;

输出:新产生的密文$(\widetilde{C}',\bar{C}')$.

① $(\widetilde{C}',x'-x,y')=\Delta_{i,j,\bar{C}}$;

② $y=\bar{C}_0$;

③ for $1\leqslant i\leqslant\eta$

④ $\bar{C}_i'=\bar{C}_i+R(x'-x,i)$;

⑤ end for

⑥ return $(\widetilde{C}',\bar{C}'=(y',\bar{C}_1',\cdots,\bar{C}_\eta'))$.

第 i 行第 υ 列的编码块的更新过程可表示为

$$\boldsymbol{P}'_{i,\upsilon}=\boldsymbol{ECV}_i\times\boldsymbol{F}'=\sum_{j=1}^{k(n-k)}\alpha_{i,j}F_j'^{\upsilon}=\sum_{j=1}^{k(n-k)}\alpha_{i,j}(F_j^{\upsilon}+(\Delta_F)_{\upsilon})=\sum_{j=1}^{k(n-k)}\alpha_{i,j}(F_j^{\upsilon}+R(x'-x,(i-1)\times r+\upsilon))=\boldsymbol{P}_{i,\upsilon}+\sum_{j=1}^{k(n-k)}\alpha_{i,j}\times R(x'-x,(i-1)\times r+\upsilon).\quad(2)$$

并将 $\widetilde{C}',\bar{C}'=y'$发送至数据管理服务器存储,替换上个版本的$(\widetilde{C},\bar{C})$.

3.2.6 数据还原

1) 数据解码与合并

根据 2.2 节中基于 FMSR 码的数据读取过程,客户端从 n 个数据节点中任取 k 个负载较小的节点并下载其所有的编码块,共计 $k(n-k)\times r$ 个编码块,然后从 3.2.3 节构造的 $\boldsymbol{EM}$ 中取出上述编码数据对应的行向量,组成一个 $k(n-k)\times k(n-k)$阶方阵,记作 $\boldsymbol{EM}'$.需要注意的是,新产生的方阵 $\boldsymbol{EM}'$中的数据来源于 $\boldsymbol{EM}$,其各个行向量线性无关,因此逆矩阵 $\boldsymbol{EM}'^{-1}$必然存在;

然后客户端使用 $\boldsymbol{EM}'^{-1}$乘以下载的编码数据块即可得到 $k(n-k)\times r$ 个原始块,将其合并组装即可得到初始的密文数据块.

取 $n=4, k=2$ 时,文件 M 的下载过程如图 5 所示,其中每个数据节点存储的数据块代表编码块矩阵对应的行向量,即 $\boldsymbol{P}_{i,j}=(P_{i,1},P_{i,2},\cdots,P_{i,r})$.

2) 数据解密

客户端从数据管理服务器下载$(\widetilde{C}',\overline{C}')$,执行数据解密过程如算法 4 所示.

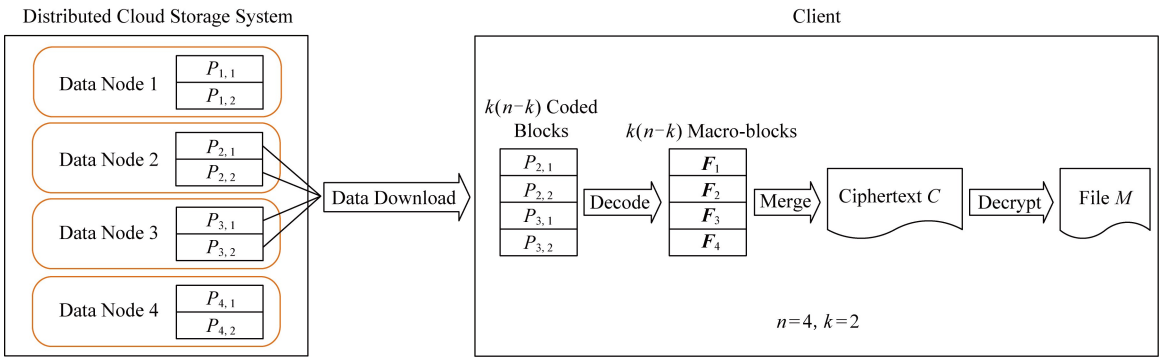


Fig. 5 Process of data reduction

图 5 数据还原过程

算法 4. 数据解密算法.

输入:当前使用的对称密钥 k 及当前版本的密文$(\widetilde{C},\overline{C})$;

输出:原始明文 M 或 $\perp$.

① $(\chi,\tau)\leftarrow_{\$}\mathcal{D}(k,\widetilde{C})$; /* 使用 k 解密当前版本的密文头 $\widetilde{C}$ */

② if $(\chi,\tau)=\perp$ then

③ return $\perp$;

④ end if

⑤ $y=\overline{C}_0$;

⑥ for $1\leqslant i\leqslant\eta$

⑦ $m_i=\overline{C}_i-R(\chi-y,i)$;

⑧ end for

⑨ if $h(m)+R(\chi-y,0)=\tau$ then

⑩ return $m=(m_1,m_2,\cdots,m_\eta)$;

⑪ else

⑫ return $\perp$;

⑬ end if

4 安全性及数据可用性分析

4.1 安全性分析

根据 Everspaugh 等人[9]的方案证明,这里从可更新加密不可区分性 UE-IND 和重加密安全性 UE-REENC 两个方面给出 DDES-UE 中可更新加密的安全性证明.

定理 1. 令 $\pi=(\mathcal{KG},\varepsilon,\mathcal{D})$为认证加密方案[9],$R:\mathcal{K}\times\mathcal{X}\rightarrow G$ 是一个密钥同态伪随机函数,$h:\{0,1\}^*\rightarrow G$ 是一个 Hash 函数.Π 是第 3 节描述的可更新加密方案.那么对于针对 Π 的任意敌手 $\mathcal{A}$,对于所有的 $\kappa\geqslant 0, t\geqslant 1$,存在敌手 $\mathcal{B},\mathcal{C}$,满足:

$$Adv^{\text{ue-ind}}_{\Pi,\kappa,t}(\mathcal{A})\leqslant 2t\times Adv^{\text{ae}}_{\pi}(\mathcal{B})+2\times Adv^{\text{prf}}_{R}(\mathcal{C}).$$

证明. UE-IND 安全游戏选取 2 个参数 t 和 κ,并初始化 $t+\kappa$ 秘密密钥,其中 κ 个给予敌手,t 个保留用于在谕言中使用.我们用 $k_1,k_2,\cdots,k_t$ 表示未泄露的密钥,$k_{t+1},k_{t+2},\cdots,k_{t+\kappa}$表示已泄露的密钥.在安全游戏中,我们要求至少一个未泄露的密钥,但对已泄露的密钥数量不进行要求,即 $t\geqslant 1$, $\kappa\geqslant 0$.敌手的目标是猜测比特位 b,成功意味着方案泄露了关于明文的部分信息.

我们将证明分为 2 部分:1)使用 π 的 AE 安全性来表明用于密钥同态 PRF 输入的 x 的值对敌手而言是隐藏的;2)基于 R 是一个密钥同态伪随机函数这一事实来说明不可区分性成立,考虑到敌手无法了解到 x.

UE-IND 安全游戏描述:

$b\leftarrow_{\$}\{0,1\}$;

$k_1,k_2,\cdots,k_{t+\kappa}\leftarrow_{\$}KeyGen()$;

$b'\leftarrow_{\$}\mathcal{A}^{O}(k_{t+1},k_{t+2},\cdots,k_{t+\kappa})$;

return $(b'=b)$.

$Enc(i,m)$:

return $Enc(k_i,m)$.

$ReKeyGen(i,j,\widetilde{C})$:

if $Invalid_{RK}(i,j,\widetilde{C})$ then

return $\perp$;

end if

$\Delta_{i,j,\widetilde{C}}\leftarrow_{\$}ReKeyGen(k_i,k_j,\widetilde{C})$;

return $\Delta_{i,j,\widetilde{C}}$.

$ReEnc(i,j,(\overline{C},\widetilde{C}))$:
$\Delta_{i,j,\widetilde{C}} \leftarrow_{\$} ReKeyGen(k_i,k_j,\widetilde{C})$;
$C'=(\widetilde{C}',\overline{C}') \leftarrow ReEnc(\Delta_{i,j,\widetilde{C}},(\widetilde{C},\overline{C}))$;
if $Invalid_{RE}(i,j,\widetilde{C})$ then
return $\widetilde{C}'$;
else return C';
end if
$LR(i,m_0,m_1)$:
if $i>t$ then
return $\perp$;
end if
$C \leftarrow_{\$} \mathrm{Enc}(k_i,m_b)$;
return C.

设 G_0 为 UE-IND 游戏.对于 $1\leqslant i\leqslant t$,游戏 G_i 与 G_{i-1} 等同,除了我们用随机字符串 r 替换 $\varepsilon(k_i,(\chi,h(m)+R(x,0)))$ 加密并存储元组 (x,y,m) 作为由随机字符串 r 索引的表 C 中的查找.对于 $ReKeyGen(i,j,\widetilde{C})$ 的查询,如果某些先前返回的 r 满足 $\widetilde{C}=r$,则正常计算 x',y'.

—若 $j\leqslant i$,为随机值 r' 返回 (r',x',y'),并存储 $(x+x',y+y',m)$,或者

—若 $j>i$,返回 $(\varepsilon(k_j,(\chi',h(m)+R(x+x',0))),x',y')$.

如果 $\widetilde{C}$ 先前没有出现,则返回 $\perp$.这些修改描述了用于 DDES-UE 方案可更新加密的安全性证明的游戏 G_i 中的替换算法:

$Enc(i,m)$:
$x,y \leftarrow_{\$} \mathcal{K}$;
$\chi=x+y$;
$r \leftarrow_{\$} \{0,1\}^{|\widetilde{C}|}$;
$C_i[r]=(x,y,m)$;
$\overline{C}_0=y$;
for $1\leqslant i\leqslant \eta$
$\overline{C}_i=m_i+R(x,i)$;
end for
return $(\widetilde{C}=r,\overline{C})$.
$ReKeyGen(i,j,\widetilde{C})$:
$(x,y,m) \leftarrow C[\widetilde{C}]$;
if $(x,y,m)=\perp$ then
return $\perp$;
end if
$x',y' \leftarrow_{\$} \mathcal{K}$;
$\chi'=x+y+x'+y'$;
if $j\leqslant i$ then
$r \leftarrow_{\$} \{0,1\}^{|\widetilde{C}|}$;
$C_j[r]=(x+x',y+y',m)$;
$\widetilde{C}'=r$;
else
$\tau=h(m)+R(x+x',0)$;
$\widetilde{C}' \leftarrow_{\$} \varepsilon(k_j,(\chi',\tau))$;
end if
return $\Delta_{i,j,\widetilde{C}}=(\widetilde{C}',x',y')$.
$Dec(i,(\widetilde{C},\overline{C}))$:
$(x,y,m) \leftarrow C_i[\widetilde{C}]$;
if $(x,y,m)=\perp$ then
return $\perp$;
end if
$y'=\overline{C}_0$;
$x'=x+y-y'$;
for $1\leqslant i\leqslant \eta$
$m'_i=\overline{C}_i-R(x',i)$;
end for
$\tau=h(m)+R(x,0)$;
if $\tau-R(x',0)=h(m')$ then
return $m=(m_1,m_2,\cdots,m_\eta)$;
else
return $\perp$;
end if

令 S_i 为敌手 $\mathcal{A}$ 在游戏 G_i 中输出正确的比特这一事件.然后我们可以构造敌手 $\mathcal{B}$,使得 $|Pr[S_{i-1}]-Pr[S_i]|\leqslant Adv^{ae}_{\pi}(\mathcal{B})$.通过调用 Π 的 AE 游戏的一个实例来替换 $\varepsilon(k_i,\cdot)$ 和 $\mathcal{D}(k_i,\cdot)$ 函数,可以在实际中得到 G_{i-1},在随机场景下得到 G_i.

假设攻击者在游戏 G_t 中查询 LR 谕言机以接收挑战密文 C.那么密文头 $\widetilde{C}$ 将等于某个随机字符串 r,而密文体 $\overline{C}$ 由参数 y 和基于输入 x 的伪随机函数 R 加密的消息组成.请注意,$i\leqslant t$ 时敌手被要求使用 AE 只能提交一次先前看到的密文头给谕言机.从 re-keygen 查询中,攻击者可以获得 x',y',r' 的新值.但是,这些不足以了解用于加密消息的 x 的值.

类似地,re-encryption 查询,其中目标密钥 j 未泄露,即 $j\leqslant t$,敌手接收在 x' 下加密的新密文,但仍无法获得 x' 的信息.另一方面,如果 $j>t$,$Invalid_{\mathrm{RE}}=\mathrm{true}$,因此攻击者能够学习到密文头.其形式为:$\varepsilon(k_j,(x+y+x'+y',h(m)+R(x+x',0)))$.

由于攻击者拥有 k_j,他们可以解密密文头.但是 x 仍被 y 所隐藏.

参考 Boneh 等人[11]给出的原始证明大纲,我们构造一个安全游戏 G_{t+1},使得等式 $|Pr[S_{t+1}]-Pr[S_t]|\leqslant Adv_R^{\text{prf}}(\mathcal{C})$ 成立.当 $i\leqslant t$ 时,我们用 PRF 挑战谕言替换 LR 谕言机中 $R(x,i)$ 的使用,表示为 $f(i)$.假设 K 是 PRF 挑战者使用的密钥.当 PRF 挑战来自现实世界时,计算的加密形式可以表述为 $m_i+R(x,i)+f(i)=m_i+R(K+x,i)$.而密文头包括值 $\tau=h(m)+R(K+x,0)$.综上分析,x 的值对于挑战密文的敌手是未知的,因此这种变化能够完美模拟 G_t.

如果密钥 j 是未泄露的,那么我们可以计算 $\overline{C}_i+R(x',i)=m_i+R(x+x'+K,i)$;否则,$Invalid_{\text{RE}}$ 或 $Invalid_{\text{RK}}$ 将为 false,并且对于密文体而言谕言机返回 $\perp$.此外,攻击者还可以获得包含加密$(x+x',\tau=h(m)+R(K+x+x',0))$的密文头.令 G_{t+1} 对应于 PRF 挑战返回随机值的事件.然后消息总是被 $f(i)$ 的输出完美掩盖,即使在计算重新加密之后该值也存在.因此,消息和散列都被 PRF 值隐藏,并且敌手无法以大于 1/2 的概率获胜.因此我们得出:

$$Adv_{\Pi,\kappa,t}^{\text{ue-ind}}(\mathcal{A})=|2\times Pr[S_0]-1|\leqslant 2t\times Adv_\pi^{\text{ae}}(\mathcal{B})+2\times Adv_R^{\text{prf}}(\mathcal{C}),$$

对于所有的 $\kappa\geqslant 0,t\geqslant 1$ 均成立.

定理 2. 令 $\pi=(\mathcal{KG},\varepsilon,\mathcal{D})$ 为认证加密方案[9],$R:\mathcal{K}\times\mathcal{X}\to G$ 是一个密钥同态伪随机函数,Π 是第 3 节描述的可更新加密方案.那么对于针对 Π 的任意敌手 $\mathcal{A}$,存在敌手 $\mathcal{B},\mathcal{C}$,对于所有的 $\kappa\geqslant 0,t\geqslant 1$,使得:

$$Adv_{\Pi,\kappa,t}^{\text{ue-reenc}}(\mathcal{A})\leqslant 2t\times Adv_\pi^{\text{ae}}(\mathcal{B})+2\times Adv_R^{\text{prf}}(\mathcal{C}),$$

成立.

证明. 在该定理证明中,应用与定理 1 证明中相同的 t 个步骤.然后我们用随机字符串替换密文头,并通过记录先前看到的输入来模拟 re-keygen 过程.

因此,在游戏 G_t 中,由于所有值都被存储在密文体中的值隐藏,敌手无法获得挑战中使用的 x 的任何信息,密文头也不会泄露任何被泄露的密钥相关的信息.接下来构建一个 PRF 敌手 $\mathcal{C}$ 来完成游戏.假设在游戏 G_t 中我们将挑战中的重加密过程替换为$(r,\overline{C}_b+f(i)+R(x'+y',i))$.也就是说密文头 $\widetilde{C}$ 是随机串,并且我们隐含地使用密钥 $\chi+K+x'$ 来重新加密密文体 $\overline{C}_b$,其中 K 是 PRF 挑战者使用的密钥.更新令牌为$(r,K+x',y')$,而敌手无法获得 $K+x'$.对于敌手任何进一步的查询 re-encryption 或 re-keygen,攻击者将保留 $f(i)$ 作为掩码的使用.

从 G_{t+1} 提供的 PRF 挑战者返回随机值的场景我们可以得出结论,敌手无法获悉有关比特位 b 的任何信息.由此可以得出敌手 $\mathcal{A}$ 的优势:

$$Adv_{\Pi,\kappa,t}^{\text{ue-reenc}}(\mathcal{A})\leqslant 2t\times Adv_\pi^{\text{ae}}(\mathcal{B})+2\times Adv_R^{\text{prf}}(\mathcal{C}).$$

4.2 数据可用性分析

本节通过对受损节点的数据重构过程描述来说明 DDES-UE 方案能够保证部分节点失效情形下的数据可用性.在 2.2 节有简要描述,这里进行详细说明.在数据节点 $N_1,N_2,\cdots,N_n$ 中,假设对数据节点 N_x 上的数据进行恢复,步骤为

1)客户端在除了 N_x 以外的每个节点上随机挑选一个编码块(共有$(n-k)^{n-1}$种不同的可能),在存储的编码矩阵中取出这$(n-1)$个编码块所对应的编码向量,记作 $\boldsymbol{ECV}_{i_1},\boldsymbol{ECV}_{i_2},\cdots,\boldsymbol{ECV}_{i_{n-1}}$.

2)在客户端构造$(n-k)\times(n-1)$的变换矩阵 $\boldsymbol{TM}=(\gamma_{i,j})$,其中 $i=1,2,\cdots,n-k,j=1,2,\cdots,n-1$,矩阵中的元素 $\gamma_{i,j}$ 均从有限域$GF(2^w)$中随机产生.

3)客户端用 $\boldsymbol{TM}$ 乘以步骤 1)中选取的编码向量,得到 $n-k$ 个新的编码块对应的编码向量 $\boldsymbol{ECV}'_1,\boldsymbol{ECV}'_2,\cdots,\boldsymbol{ECV}'_{n-k}$,该过程可表示为

$$(\boldsymbol{ECV}'_1,\boldsymbol{ECV}'_2,\cdots,\boldsymbol{ECV}'_{n-k})^{\text{T}}=\boldsymbol{TM}\times(\boldsymbol{ECV}_{i_1},\boldsymbol{ECV}_{i_2},\cdots,\boldsymbol{ECV}_{i_{n-1}})^{\text{T}},\quad(3)$$

其中每个新的编码向量可表示为

$$\boldsymbol{ECV}'_i=\sum_{j=1}^{n-1}\gamma_{i,j}\boldsymbol{ECV}_{i_j},i=1,2,\cdots,n-k.\quad(4)$$

4)客户端用$(\boldsymbol{ECV}'_1,\boldsymbol{ECV}'_2,\cdots,\boldsymbol{ECV}'_{n-k})$替换掉原来编码矩阵 $\boldsymbol{EM}$ 中的数据节点 N_x 所占部分,形成一个新的编码矩阵 $\boldsymbol{EM}'$.

5)客户端判断 $\boldsymbol{EM}'$ 是否满足 MDS 性质,即判断 $\boldsymbol{EM}'$ 的中任意 $k(n-k)$ 个行向量组成的方阵是否满秩.若满足 MDS 性质,执行步骤 6),否则跳到步骤 1),进入下一轮迭代,重新挑选新的编码向量和变换系数.

6)客户端将 $\boldsymbol{TM}$,以及步骤 1)中挑选出的编码块序号发送给数据节点 N_x.

7)数据节点 N_x 从相应的数据节点上下载步

骤1)中的挑选出的编码块,并使用 ***TM*** 对这些编码块进行编码得到 $n-k$ 个数据块,覆盖自身的原有数据,完成一次数据重构.

5 DDES-UE 方案的性能分析

DDES-UE 方案在存储结构、数据加密、密钥轮换以及安全性证明4个方面的对比如表1所示.

本节主要从数据上传、数据下载、污染数据恢复以及密文更新4个环节对 DDES-UE 方案的性能进行验证与分析,分别选取 10~100 MB(每次以 10 MB 递增)的文件大小,对每个环节中各部分所涉及的关键计算(如数据读取、数据写入、群元素加法运算、群元素加法运算、群元素幂运算、群元素映射等)的时间开销进行了综合测试,并使用 Java 编程语言构建了我们的参考实现.实验环境描述为:Windows 64 b 操作系统,Intel® Core™ i7-4770 CPU(3.4 GHz),内存 12 GB.

Table 1 Comparison of Advantages in Different Schemes

表1 方案优势对比

Schemes	Storage Architecture	Data Encryption is Supported or Not	Key Rotation is Supported or Not	Security Proofs
Google[7]	Distributed Storage	Partial Data Encryption	No	/
Amazon[8]	Ddistributed Storage	Partial Data Encryption	No	/
Ref [9]	/	Yes	Yes	Yes
Ref [11]	/	Yes	Yes	/
Ref [14]	/	Yes	No	Yes
Ref [19]	/	Yes	Yes	Yes
Ref [21]	Distributed Data Coding	Yes	Yes	/
Ref [24]	Distributed Data Coding	No	No	/
Ref [25]	Distributed Data Coding	No	No	/
Ref [26]	Distributed Data Coding	No	No	/
Ref [33]	Distributed Data Coding	Yes	No	/
Ours	Distributed Data Coding	Yes	Yes	Yes

Note: "/" indicates that this scheme does not show the corresponding explanation.

在仿真实验中首先需要实例化 DDES-UE 使用的密钥同态 PRF.这里使用 $R(x,i)=H(i)^x$,其中 H 被建模为随机预言 $H:X\to G$.构建 H 的自然方法是采用常规 Hash 函数 $h:\{0,1\}^*\to\{0,1\}^n$.X 是任何合适的集合(例如,某些固定长度的位串),并且 $(G,+)$是判定性 Diffie Hellman 假设成立的群.

在方案实现的过程中我们还需要解决消息编码的问题.根据 $\overline{C}_i=m_i+R(x,i)$,加密是按块完成的.等式成立前提是消息 m 已经表述为群 G 上的元素序列.为实现加密数据到椭圆曲线点的映射,需要构造一个针对数据块的长度为 n 的编码函数 σ_m:$\{0,1\}^n\to G$.根据解密过程 $m_i=\overline{C}_i-R(\chi-y,i)$,生成的 m_i 是群 G 上的元素,还需要还原为比特序列.因此我们要求消息编码函数的正反向都能够高效计算.本文使用 Elligator2 函数[31]来实现 σ_m,即构造为 KHPRF 中的映射函数.Elligator2 使用 Curve25519 曲线进行函数设计.因此在实现中使用特定曲线 Curve25519[32],其中 $p=2^{255}-19$.参照 Everspaugh 等人的方案[9],我们使用 SHA256 作为散列函数 h.Elligator2 能够实现 Z_p 中的群元素到群 G 中元素的双向映射,而 m_i 表示的是划分后的文件块,需要通过 Hash 函数 $h:\{0,1\}^*\to\{0,1\}^n$ 实现文件块到 Z_p 群元素的映射,此时可以得出 $H(i)=\sigma_m(h(i))$.

其中数据上传(选取数据加密、数据编码、数据传输、数据写入4个部分)与数据下载(选取数据读取、数据传输、数据解码、数据解密4个部分)过程仿真实验结果如图6、图7所示.结合 Hu 等人[25]以及陈越等人[33]中的性能分析可以看出,本文构造的 DDES-UE 方案在分布式数据存储与恢复(如在数据上传与下载过程中涉及的数据读取、数据传输、数据编码解码以及数据合并等运算和在污染数据恢复时的变换矩阵计算、下载编码块、数据重构等)上的性能开销在可接受范围内.

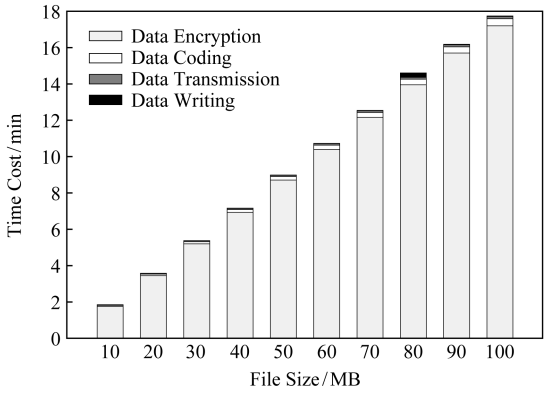


Fig. 6 Time cost in different stages of data uploading

图 6 数据上传过程计算开销

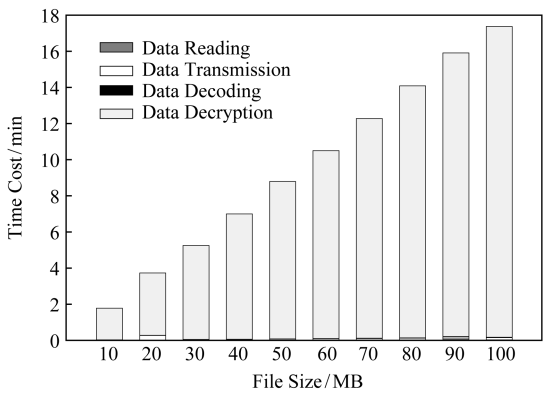


Fig. 7 Time cost in different stages of data downloading

图 7 数据下载过程计算开销

污染数据修复(选取计算转换矩阵、下载编码数据块、数据重构、数据写入 4 个部分)与数据更新过程仿真实验结果分别如图 8、图 9 所示.随着文件大小的增长,污染数据修复过程及数据更新的计算开销均呈线性增长趋势,但前者的时间开销较小,能够保证高效的数据可恢复性;而文件更新的计算开销

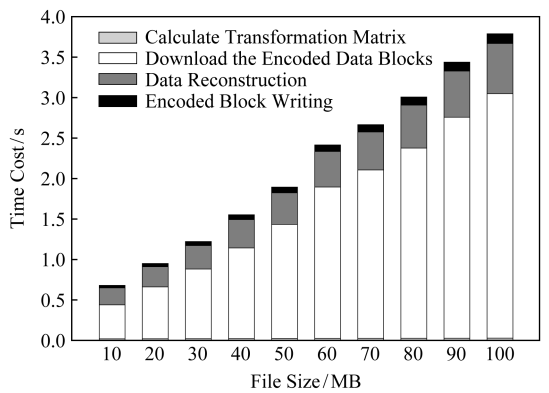


Fig. 8 Time cost in different stages of data recovery

图 8 数据修复过程计算开销

过大.接下来对仿真实验中涉及数据加解密的计算开销进行详细分析.

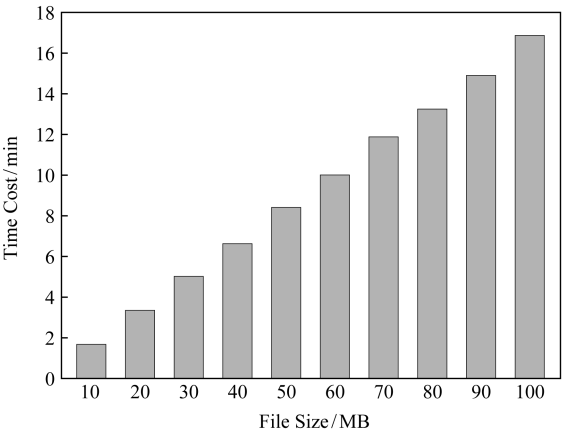


Fig. 9 Time cost of one data update

图 9 一次数据更新过程计算开销

根据 Everspaugh 等人[9]的测试结果(Rust 编程语言实现),其提出的 ReCrypt 方案加密 1 个数据块(≤31 B),1 KB,1 MB 和 1 GB 的明文分别需要 253 μs,8.5 μs,8.9 s 和 2.5 h 内.同样.我们对数据加密中涉及的计数器模式加密运算和解密过程的运算进行测试,使用 Elligator2 算法进行加法群元素到椭圆曲线群元素的映射及其逆映射以及一次椭圆曲线群元素加法运算耗时分别为 92 μs,157.5 μs 和 23.5 μs.一方面,使用 Elligator2 时数据块的大小等于群 G 中元素大小,这使得实际上使用 31 B 的块大小,而以 MB 为单位的文件将产生较多的数据块,极大影响了加密的性能;另一方面,椭圆曲线点的倍数 x 是从 Z_p 中随机选取的群元素,进行倍数运算时候开销不容忽视.根据等式 $\overline{C}_i=m_i+R(x,i)$,$m_i=\overline{C}_i-R(\chi-y,i)$ 及 $R(x,i)=H(i)^x=(\sigma_m(h(i)))^x$,可看出 DDES-UE 方案的性能很大程度上取决于计算伪随机函数所需的标量乘法次数和 Elligator2 中元素映射运算的次数.这是制约 DDES-UE 中数据加解密及密文更新的效率瓶颈.当前有很多方案(如 Sasdrich 等人[34]和 Zhe 等人[35]等)针对椭圆曲线点的标量乘法效率展开研究,如何提高方案中可更新加密的计算效率也是下一步需要解决的问题.

当然,正如 Everspaugh 等人[9]描述的那样,我们提出的方案 DDES-UE 当前最适用于体量小但非常有价值的数据,比如大型企业的重要客户资料备份等.该方案既能够通过分布式存储满足数据灾备及污染数据恢复的需求,同时也支持分布式架构下存储数据的直接更新,有效避免了数据恢复与重加

密更新的计算开销和数据上传下载造成的网络通信开销.

6 结束语

本文提出 DDES-UE 方案,支持分布式云存储中不同数据节点持有的编码密文块的直接更新.基于密钥同态伪随机函数,可以将当前密文直接更新为新版本的密文,且更新后的密文与当前版本的密文是一致分布的,即等同于用新密钥加密的密文,在保证数据安全性的同时节约了通信资源成本.

此外,结合改进的 FMSR 编码技术,提出的 DDES-UE 方案能够确保分布式存储系统中部分数据节点受损时的数据可用性,且支持数据恢复时的完整性验证.同时,可更新加密中的密钥更新具有时变性,压缩了攻击者发起攻击的时间窗口.假设在较长时间内攻击者通过某种攻击方式获取到存储密文及用户私钥及编码矩阵,若三者不在同一个时间周期,则无法完成解密,从而使得攻击结果无效.因此,当云存储中加密数据的密钥周期性更新时,系统能够通过增加攻击者的时间成本,有效降低攻击者通过获取密文密钥相关信息来破解密文的概率.

安全性证明和性能分析表明:DDES-UE 方案可以实现安全有效的密文更新,以应对密钥泄露的风险;避免了数据重加密时数据上传下载带来的巨大通信开销,并支持数据恢复以保证数据的持续可用性.然而,以群元素大小为单位的数据更新导致计算伪随机函数所需的标量乘法次数以及 Elligator2 函数中元素映射的次数成为制约密文更新效率的关键因素.这使得我们在有效解决了降低网络带宽负载及客户端重加密计算开销问题之后,仍需考虑如何降低执行密文更新操作的云存储服务器的计算开销.因此,未来工作将深入研究分布式云环境下以数据块为单位的直接密文更新方法以及针对椭圆曲线点的标量乘法效率改进,为进一步构建更具普适性的加密云存储系统提供技术支持.

参 考 文 献

[1] National Institute of Standards and Technology. Recommendation for key management [OL]. 2016 [2019-06-01]. http://dx.doi.org/10.6028/NIST.SP.800-57pt1r4

[2] James N, Elaine B, Lawrence B, et al. Report on the development of the advanced encryption standard (AES)[J]. Journal of Research of the National Institute of Standards & Technology, 2001, 106(3): 511-577

[3] Zhang Mingwu, Yang Bo, Tsuyoshi T. Master-key leakage-resilient and continue leakage-resilient functional encryption in dual affine spaces [J]. Chinese Journal of Computers, 2012, 35(9): 1856-1867 (in Chinese)
(张明武, 杨波, Tsuyoshi T. 抗主密钥泄露和连续泄露的双态仿射函数加密[J]. 计算机学报, 2012, 35(9): 1856-1867)

[4] Myers S, Shull A. Practical revocation and key rotation [C] //Proc of Cryptographers' Track at the RSA Conf (CT-RSA 2018). Berlin: Springer, 2018: 157-178

[5] The OWASP Foundation. Open Web application security project: Cryptographic storage cheat sheet [OL]. 2016 [2019-06-01]. https://www.owasp.org/index.php/Cryptographic_Storage_Cheat_Sheet

[6] Payment Card Industry Security Standards Council (PCI SSC). DSS: Payment card industry data security standard [OL]. 2018 [2019-06-01]. https://en.wikipedia.org/wiki/Payment_Card_Industry_Data_Security_Standard

[7] Google. Managing data encryption [OL]. 2017 [2019-06-01]. https://goo.gl/5UidnU

[8] Amazon Web Services. Rotating customer master keys [OL]. 2017 [2019-06-01]. https://goo.gl/Ym9WeM

[9] Everspaugh A, Paterson K, Ristenpart T, et al. Key rotation for authenticated encryption [G] //LNCS 10403: Advances in Cryptology (CRYPTO 2017). Berlin: Springer, 2017

[10] Zhao Yongjun, Chow S S M. Updatable block-level message-locked encryption [C] //Proc of the 2017 ACM on Asia Conf on Computer and Communications Security (ASIACCS 2017). New York: ACM, 2017: 449-460

[11] Boneh D, Lewi K, Montgomery H, et al. Key homomorphic PRFs and their applications [G] //LNCS 8042: Proc of CRYPTO 2013. Berlin: Springer, 2013: 410-428

[12] Sahai A, Seyalioglu H, Waters B. Dynamic credentials and ciphertext delegation for attribute-based encryption [G] //LNCS 7417: Proc of CRYPTO 2012. Berlin: Springer, 2012: 199-217

[13] Lewko A, Okamoto T, Sahai A, et al. Fully secure functional encryption: Attribute-based encryption and (hierarchical) inner product encryption [G] //LNCS 6110: Proc of EUROCRYPT 2010. Berlin: Springer, 2010: 62-91

[14] Lee K, Choi S G, Dong H L, et al. Self-updatable encryption: Time constrained access control with hidden attributes and better efficiency [G] //LNCS 8269: Proc of ASIACRYPT 2013. Berlin: Springer, 2013: 235-254

[15] Datta P, Dutta R, Mukhopadhyay S. Fully secure self-updatable encryption in prime order bilinear groups [G] //LNCS 8783: Proc of Int Conf on Information Security. Berlin: Springer, 2014: 1-18

[16] Freeman D M. Converting pairing-based cryptosystems from composite-order groups to prime-order groups [G] //LNCS 6110: Proc of EUROCRYPT 2010. Berlin: Springer, 2010: 44-61

[17] Lewko A. Tools for simulating features of composite order bilinear groups in the prime order setting [G] //LNCS 7237: Proc of EUROCRYPT 2012. Berlin: Springer, 2012: 318-335

[18] Lee K. Self-updatable encryption with short public parameters and its extensions [J]. Designs, Codes and Cryptography, 2016, 79(1): 121-161

[19] Aono Y, Hayashi T, Phong L T, et al. Efficient key-rotatable and security-updatable homomorphic encryption [C] //Proc of the 5th ACM Int Workshop on Security in Cloud Computing. New York: ACM, 2017: 35-42

[20] Lee K, Lee D H, Park J H, et al. CCA security for self-updatable encryption: Protecting cloud data when clients read/write ciphertexts [J]. The Computer Journal, 2019, 62(4): 545-562

[21] Parra J R, Chan T H, Ho S W. Updatable encryption in distributed storage systems using key homomorphic pseudorandom functions [J]. International Journal of Information and Coding Theory, 2016, 3(4): 365-391

[22] Diffie W, Hellman M E. Privacy and authentication: An introduction to cryptography [J]. Proceedings of the IEEE, 1979, 67(3): 397-427

[23] Dworkin M. Recommendation for block cipher modes of operation: Methods and techniques [J]. National Institute of Standards & Technology, 2001, 5(6): 669-675

[24] Dimakis A G, Godfrey P B, Wu Y, et al. Network coding for distributed storage systems [J]. IEEE Transactions on Information Theory, 2010, 56(9): 4539-4551

[25] Hu Yucong, Lee P P C, Shum K W. Analysis and construction of functional regenerating codes with uncoded repair for distributed storage systems [C] //Proc of the 32nd INFOCOM Conf. Piscataway, NJ: IEEE, 2013: 2355-2363

[26] Rawat A S, Koyluoglu O O, Silberstein N, et al. Optimal locally repairable and secure codes for distributed storage systems [J]. IEEE Transactions on Information Theory, 2014, 60(1): 212-236

[27] Li Jie, Tang Xiaohu, Tian Chao. A generic transformation to enable optimal repair in MDS codes for distributed storage systems [J]. IEEE Transactions on Information Theory, 2018, 64(9): 6257-6267

[28] Su Yisheng. Pliable fractional repetition codes for distributed storage systems: Design and analysis [J]. IEEE Transactions on Communications, 2018, 66(6): 2359-2375

[29] Tang Yingjie, Wang Fang, Xie Yanwen. An efficient failure reconstruction based on in-network computing for erasure-coded storage systems [J]. Journal of Computer Research and Development, 2019, 56(4): 767-778 (in Chinese)
(唐英杰, 王芳, 谢燕文. 纠删码存储系统中基于网络计算的高效故障重建方法[J]. 计算机研究与发展, 2019, 56(4): 767-778)

[30] Naor M, Pinkas B, Reingold O. Distributed pseudo-random functions and KDCs [G] //LNCS 1592: Proc of EUROCRYPT 1999. Berlin: Springer, 1999: 327-346

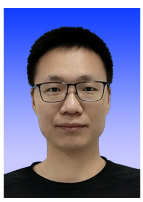
[31] Bernstein J, Hamburg M, Krasnova A, et al. Elligator: Elliptic-curve points indistinguishable from uniform random strings [C] //Proc of the 2013 ACM SIGSAC Conf on Computer & Communications Security. New York: ACM, 2013: 967-980

[32] Bernstein J. Curve25519: New Diffie-Hellman speed records [G] //LNCS 3958: Proc of PKC 2006. Berlin: Springer, 2006: 207-228

[33] Chen Yue, Wang Longjiang, Yan Xincheng, et al. Mimic storage scheme based on regenerated code [J]. Journal on Communications, 2018, 39(4): 21-34 (in Chinese)
(陈越, 王龙江, 严新成, 等. 基于再生码的拟态数据存储方案[J]. 通信学报, 2018, 39(4): 21-34)

[34] Sasdrich P, Tim G. Efficient elliptic-curve cryptography using Curve25519 on reconfigurable devices [G] //LNCS 8405: Proc of the 10th Int Symp on Applied Reconfigurable Computing. Berlin: Springer, 2014: 25-36

[35] Liu Zhe, Groszschaedl J, Hu Zhi, et al. Elliptic curve cryptography with efficiently computable endomorphisms and its hardware implementations for the Internet of things [J]. IEEE Transactions on Computers, 2017, 66(5): 773-785



Yan Xincheng, born in 1991. PhD candidate. His main research interests include cloud data privacy protection and secure data sharing based on cryptography.

Chen Yue, born in 1965. PhD, professor and PhD supervisor. His main research interests include network and information security.

Ba Yang, born in 1989. PhD candidate. His main research interests include blockchain and data security, applied cryptography.

Jia Hongyong, born in 1978. PhD, lecturer. His main research interests include applied cryptography and hierarchical data access control.

Zhu Yu, born in 1992. Master candidate, assistant engineer. His main research interests include cloud computing and data security.