基于信任值的云存储数据确定性删除方案

冯贵兰[1] 谭 良[1,2]

(四川师范大学计算机学院 成都 610101)[1] (中国科学院计算技术研究所 北京 100190)[2]

摘 要 数据确定性删除是云存储安全的研究热点。目前云存储中分散式的数据确定性删除方案在密钥分量存储时没有考虑DHT((Distributed Hash Table)节点可信性,使得用户在授权时间内也存在无法访问自己敏感数据的问题。为此提出了一种基于信任值的云存储数据确定性删除方案,该方案的核心是对DHT节点进行可信度评价,密钥分量的存储选择可信度较高的节点。与已有的确定性删除方案相比,该方案仍然是利用DHT网络的动态特性实现密钥的定期删除,使得用户敏感数据能够在特定的一段时间之后自动销毁。不同的是,该方案在将密钥分量分发到DHT网络中时倾向于选择可信度高的节点进行交互,使得在密钥的过期时间戳之前从DHT网络中得到足够多的密钥分量来恢复出密钥的可能性大大提高,降低了用户在授权时间内无法访问自己敏感数据的概率。实验结果表明,该方案不仅可以有效地抑制恶意节点,还可以提高密钥分量提取成功率,从而增加用户在授权时间内访问自己敏感数据的成功率。

关键词 云存储,数据删除,节点信任值,数据机密性

中图法分类号 TP311 **文献标识码** A

Data Assured Deletion Scheme Based on Trust Value for Cloud Storage

FENG Gui-lan[1] TAN Liang[1,2]

(College of Computer, Sichuan Normal University, Chengdu 610101, China)[1]

(Institute of Computing Technology, Chinese Academy of Sciences, Beijing 100190, China)[2]

Abstract The data assured deletion is a spot for cloud storage security. The data assured deletion approach which is distributed is not account for nodes' trust in the DHT, so it will lead user can't access data before the deadline. To solve this problem, a data assured deletion scheme based on trust value for cloud storage (DDTV) was proposed. The core of DDTV is evaluation of DHT nodes and choosing high nodes to store key shares. The dynamic property of DHT network makes keys to be deleted periodically, causing the sensitive data in cloud computing to be automatically destroyed after the expiration time. Compared with the data assured deletion approach which is distributed, the difference is that keys are pushed to DHT network based on trust value after partitioned by secret sharing scheme, especially the nodes which have high trust value will be chosen in the DDTV. This difference improves the possibility of decrypting the data before the deadline. The experiment results show that the method can not only identify the malicious node in the DHT, but also improve the success rate of key share extract. The high rate of key share extract can increase the rate of user access to sensitive data in the authorized time.

Keywords Cloud storage, Data deletion, Node trust value, Data confidentiality

1 引言

数据确定性删除是云存储安全的研究热点,因为在云存储模式下,数据的存储和处理都是在CSP的服务器上实现的,脱离了用户的控制范围,即使数据以密文状态保存在CSP处也可能会存在一些安全隐患。例如CSP为了提高服务的可靠性,数据会做多个备份[1]。用户要求CSP删除其数据时,CSP可能会恶意地保留此文件,或者由于技术原因并未删除所有副本。一旦攻击者得到密钥,并从不可信CSP得到未被删除的密文数据或备份,数据就面临着被泄露的风险。

目前云存储中分散式的数据确定性删除方案[2-5]可以使用户敏感数据在不需要任何用户或可信第三方进行干预的情况下,在特定的一段时间之后自动销毁。但这类数据确定性删除方案都蕴含了一个前提,即所有DHT节点都是诚实的。然而在实际网络中,可能会出现一些恶意的行为。例如,在密钥分发时,DHT节点直接将分发给自己的密钥分量删除或大面积地泄露;在密钥恢复时,DHT节点提供不正确的密钥分量,使得无法恢复正确的密钥;这些恶意行为会导致用户在授权时间内无法访问自己的敏感数据。为解决这个问题,本文提出了一种基于信任值的云存储数据确定性删除方案(A

到稿日期:2013-08-05 返修日期:2013-10-28 本文受国家自然科学基金(61373162),四川省教育厅青年基金项目(08zb02),四川师范大学校级项目(201314)资助。

冯贵兰(1988-),女,硕士,主要研究领域为云计算、信息安全,E-mail:fengguilan1016@sina.com;谭 良(1972-),男,博士,教授,主要研究领域为可信计算、网络安全。

Data Assured Deletion Scheme based on Trust Value for Cloud Storage，简称 DDTV)。在 DDTV 方案中，每个 DHT 节点都有一个可信度值，该值是根据该节点在与云用户的交互历史中的表现获得的。在进行密钥分发时，用户倾向于选择可信度高的节点存放密钥分量，使密钥分量存储得更安全。该方案不仅确保了用户敏感数据在授权时间内可以使用，还使用户获得较高的密钥分量提取成功率。

本文第 2 节回顾前人在该领域所做的相关工作；第 3 节具体介绍 DDTV 的总体模型和具体构造；第 4 节介绍 DDTV 实验及分析；最后给出全文的总结。

2 相关工作

当前的数据确定性删除方案主要集中在两个方面，其一是集中式的管理方式，主要有 Ephemerizer 机制[6]、基于策略的文件确定性删除[7]。文献[6]提出的 Ephemerizer 机制需要一个或者多个可信第三方来为用户保存数据解密密钥，并负责在一段用户指定的时间后销毁解密密钥，从而使得任何人都无法解密出数据明文。FADE 系统[7]在文献[6]的基础上提出了一种基于策略的文件确定性删除方法，其基本思想是由数据密钥对文件加密，再由与策略相关联的控制密钥加密数据密钥，删除控制密钥即可实现数据的确定性删除。然而，在这类集中式的方案中控制密钥是由第三方的密钥管理者管理的，存在密钥管理者不可信而未删除或泄漏控制密钥的安全隐患。

其二是分散式的管理方式。相对于集中式的管理方法，分布式密钥管理方法的安全性更高，更加适合云计算环境。典型的是 Vanish 机制[2]。该机制首先将用户敏感数据进行加密，然后把密钥经秘密共享方案处理后分发到 DHT 网络中，利用 DHT 网络的动态特性实现密钥的定期删除，使得加密数据在非授权时间内不能被解密和访问，从而实现了 CSP 服务器上的数据副本的自销毁。随后的文献[3,4]都是在 Vanish 的基础上进行了相应的改进。文献[3]认为只销毁密钥而不销毁数据存在着攻击者暴力破解密码算法的安全隐患，在 Vanish 系统的基础上，将密钥和部分密文数据一起分发到 DHT 网络中，使得攻击者暴力破解不完整密文数据所需的密钥空间增大，从而增加攻击的难度和代价。该方案将部分密文也分发到网络中，增加了网络通信开销。文献[4]通过对 Shamir 秘密共享算法进行改进，扩展密钥份数的长度来抵抗 Vanish 系统中存在的跳跃攻击。文献[5]认为单个密钥加密全部数据不适用海量数据规模的云存储，将 Vanish 系统中的单个密钥扩展到使用密钥派生树生成和管理密钥。但这类数据确定性删除方案都蕴含了一个前提，即所有 DHT 节点都是诚实的。然而，在实际网络中，DHT 节点可能会出现一些恶意的行为。例如，在密钥分发时，DHT 节点直接将分发给自己的密钥分量删除或大面积地泄露；在密钥恢复时，DHT 节点提供不正确的密钥分量，使得无法恢复正确的密钥。这些恶意行为会导致用户在授权时间内无法访问自己的敏感数据。

3 基于信任值的云存储数据确定性删除方案

3.1 技术背景

为了更好地描述本文提出的 DDTV 方案，在这一节中将详细地介绍一些关键技术。

3.1.1 DHT 网络

DHT 网络是指利用 DHT 表[8,9]存储数据和实现节点路由的 P2P 网络。DHT 网络的以下 3 个特性使得它能够恰好应用于本文提出的数据自毁机制中，这 3 个特性依次如下：

(1)可用性。多年的 DHT 网络研究已经使得如今的 DHT 网络具有良好的可用性，尤其是它提供了一个稳定的特定超时期限。各种 DHT 网络的超时期限各不相同，比如：Vuze[10]的超时期限是 8 个小时，而 OpenDHT[11]则允许用户在不大于一个星期的范围内自主选择超时期限。该特性使得被保护的数据在授权时间(称之为 timeout) 内是可用的，是 DHT 网络能够应用到本文研究的基础。

(2)规模大且地理分布广。研究表明在 Vuze 网中同时并发存在的活动节点超过了 100 万个，并且地理分布超过了 190 个国家。这种完全非集中的分布方式能够提供很健壮的抗攻击能力。

(3)周期性更新。DHT 网络有这样一个特性，它每经过一段特定的时间，就会丢弃旧的数据以腾出空间来存储新的数据[12]。DHT 的这个特性能够使用户敏感数据在不需要任何用户或第三方进行干预的情况下，在特定的一段时间之后自动销毁。

3.1.2 秘密共享方案

秘密共享方案(Secret Sharing Scheme，SSS)[13]是将秘密 k 以某种方式分成 n 份(shares)：$k_1, k_2, \cdots, k_n$，并且满足：

(1)通过任意 t 或 t 个以上 k_i 能够计算出 k；

(2)通过任意 $t-1$ 个或更少的 k_i 不能计算出 k。

这种方法也称为(t,n)门限方案，称 t 为门限阈值，n 为门限值，t/n 为门限率。

通过 SSS 将任意秘密 k 拆分成 n 份，且重构 k 需要至少知道任意 t 或 t 个以上 k_i，故泄露 $s<t$ 个 shares 也不会暴露秘密 k，而丢失 $s<n-t$ 个 shares 仍然可以重构出 k。SSS 是解决 DHT 网络因为动态性使得密钥丢失而不可用的一种有效方法。

3.2 FTrust 信任模型

为评估 DHT 节点的可信度，本节建立 DHT 节点的信任模型——FTrust。DHT 中每个节点都有一个可信度值，该值是根据该节点在与云用户的交互历史中的表现而获得的。云用户在每个节点的每一次交互之后，就对该节点做出一个评价，而综合所有云用户对该节点作出的评价，就是该节点的可信度值。其中涉及 3 个相关的方面，即信任值的定义与表示、信任值的存储和信任值的求解。

1. 信任值的定义与表示

首先给出满意度评价函数，然后给出了个体信任度的定义，最后引出了全局信任度的定义。

定义 1 满意度评价函数。其功能是用户 u 在和节点 v 进行第 n 次交互之后提交满意度的评价值，我们可将用户 u 对节点 v 第 n 次交互的满意度的评价定义为 Map 函数 $f_n(u,v)$：

$$f_n(u,v)=\begin{cases}1, & \text{完全满意}\\ 0, & \text{完全不满意}\\ e(\in(0,1)), & \text{其他}\end{cases} \tag{1}$$

我们采用概率可能性的方法来区分节点提供的不同服务质量，1 表示用户 u 对节点 v 第 n 次交互完全满意，0 表示用户 u 对节点 v 第 n 次交互完全不满意，值越大表示本次交互的满意度越高。

定义 2 个体信任度，即归一化的个体满意度。在时间区间 t（t 视具体的应用而定，如一个星期）内，假设该用户 u 和节点 v 之间交互的次数为 m，则个体信任度可定义为：

$$D_{uv}=\begin{cases}\dfrac{\sum_{n=1}^{m} f_n(u,v)}{m}, & m\neq 0\\ 0, & m=0\end{cases} \tag{2}$$

D_{uv} 是在时间区间 t 内用户 u 根据直接交易历史对节点 v 做出的信任评价，也即用户 u 对节点 v 提供的反馈。当 $m=0$ 时，表示用户 u 与节点 v 之间没有交互历史，将用户 u 对节点 v 的个体信任度设定为 0。D_{uv} 的取值范围如下，D_{uv} 的值越高代表个体信任度越高：

$$D_{uv}=\begin{cases}1, & \text{完全满意}\\ 0, & \text{完全不满意}\\ d(\in(0,1)), & \text{其他}\end{cases} \tag{3}$$

定义 3 全局信任度，即归一化的全局满意度。假设在网络 N 中对节点 v 提供过反馈评价的用户数为 k，则节点 v 的全局可信度 T_v 可定义为：

$$T_v=\begin{cases}\dfrac{\sum_{i=1}^{k} D_{iv}}{k}, & k\neq 0\\ 0, & k=0\end{cases} \tag{4}$$

当 $k=0$ 时，表示节点 v 还没有与任何用户进行交互，将节点 v 的全局信任度设定为 0。T_v 的取值范围如下，T_v 的值越高代表全局信任度越高：

$$T_v=\begin{cases}1, & \text{完全满意}\\ 0, & \text{完全不满意}\\ t(\in(0,1)), & \text{其他}\end{cases} \tag{5}$$

2. 信任值的存储

在云中设置一个代理 Agent 用于计算和存储 DHT 节点信任值，Agent 具有以下 4 项功能：

(1)存储节点 i 与全局可信度计算相关的数据；

(2)验证所存储的数据的合理性；

(3)向其他云用户提交 i 的全局可信度 T_i；

(4)计算它所管理的所有节点的信任值。

3. 全局信任值求解算法

(1)首先给出用到的原语及其语义

Put_Eval(ID_v, ID_u, $EVal_{uv}$)：用户 u 将对节点 v 的满意度评价 $EVal_{uv}$ 写入到代理 Agent 中。

Put_Num(ID_v, ID_u, Num_{uv}, $TNum_{uv}$)：用户 u 将在一段时间内与节点 v 交易的次数 Num_{uv} 及与其他用户交易的总次数 $TNum_{uv}$ 写入到代理 Agent 中。

Get_Num(ID_v, ID_u, N_u, TN_u)：从代理 Agent 中读取用户 u 与节点 v 交易的次数及其与其他用户交易的总次数信息，并分别存入本地变量 N_u 与 TN_u 中。

Get_Eval(ID_v, ID_u, $EVal_{uv}$)：从代理 Agent 中读取用户 u 对节点 v 的满意度评价信息。

Get_GlobalTrust(ID_v, T_v)：从代理 Agent 中读取节点 v 的全局信任度并写入本地变量 T_v 中。用户通过该原语获取任意节点 v 的当前全局信任度；

CalGlobalTrust(ID_v, ID_u, T_v)：计算任意节点 v 全局信任度的过程，并将最终结果存入本地变量 T_v 中；

TransFinish (ID_v)：节点每次与其他节点进行交易，最终通过该原语评估交易是否结束。ID_v 为交易对方（服务提供方），其值为 True 表示一次交易完成，否则表示未完成。

(2)用户和节点 v 交互完成后，对该节点的表现进行评价，评价算法如下：

```
Procedure Evaluate (IDv){
    If (TransFinish(IDv)==True){
    Evaluv← Call (1) //调用式(1)进行满意度评价
    Put_Eval(IDv,IDu,EValuv);
    Get_Num(IDv,IDu,Nu,TNu);
    Nu++;
    TNu++;
    Put_Num(IDv,IDu,Numuv,TNumuv);
    }
    CalGlobalTrust();
}
```

(3)代理 Agent 求解节点 v 的全局信任度，全局信任度计算算法如下：

```
Procedure CalGlobalTrust(){
    for (k∈FSet){
        Get_Eval(IDv,IDu,EValuv);
        Get_Num(IDv,Tv);
        Dkv←Call(2);  //调用式(2)计算个体信任值，即取很多用
        户对节点 v 的评价
    }
    Tv←Call (4);//调用式(4)计算全局信任值
}
```

本节建立的信任模型——FTrust 和文献[14]提出的信任模型理念比较相似，但我们的模型和此模型有两个方面的区别：1)我们的模型是从外部对 DHT 节点行为进行评价，由云用户对节点进行评价，而节点并不对云用户进行评价，此评价是单向的。而文献[14]提出的模型是 DHT 节点内部之间的评价行为，评价是双向的。2)在 FTrust 信任模型中节点信任值的存储和计算由 Agent 完成，而文献[14]提出的模型中节点信任值的存储和计算由该节点所对应的管理节点完成。

3.3 DDTV 方案描述

本节在秘密共享方案[13]、DHT 网络[8,9]和 FTrust 信任模型的基础上，建立了如图 1 所示的系统方案。该模型由数据拥有者 Owner、云服务器提供商 CSP 和授权用户 User 3 个实体组成。由于 CSP 是不可信的，Owner 将数据加密后再上传到 CSP 的服务器上。与其他加密方案不同的是，数据解密密钥不是由 Owner 保存在本地，而是将解密密钥经秘密共享方案处理后分发到基于信任值的 DHT 网络中，分发时倾向于选择可信度高的节点进行交互。User 通过授权信息从 CSP 处获取密文数据，通过密钥定位器恢复出解密密钥，解密

后得到明文数据。User 可以根据 DHT 节点的表现做出相应评价，即个体信任度，而综合所有 User 对该节点的评价，就是该节点的全局信任度。

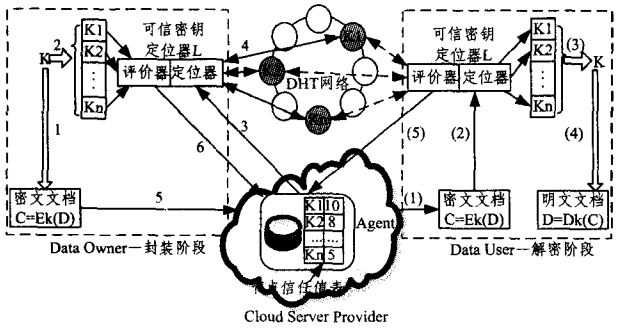

图 1　DDTV 系统方案

DDTV 模型主要包括封装阶段和解密阶段两个流程。封装阶段的主要功能是将数据加密后外包给 CSP，把密钥分发到基于信任值的 DHT 节点中。解密阶段是通过密钥定位器恢复出解密密钥，解密得到明文数据。在这两个阶段的最后一个步骤，都会进行节点评价。节点评价是当云用户和 DHT 节点一次交互完成后，根据节点的行为给出对应的评价。

3.4　DDTV 方案流程

3.4.1　封装阶段

由于存储文档的服务器是不可信的，如果将文档直接提交上传，那么将导致用户数据的安全和隐私受到威胁，因此需要进行加密处理从而保证信息在服务器端存储时的安全性。同时为了使文档一旦过了生存期限，任何人都将无法解密出这些敏感数据，需将解密密钥分发到基于信任值的 DHT 网络中。为满足以上要求，数据拥有者 Owner 在文档上传之前需对其进行封装，封装的具体步骤如图 1 左所示。

1. 文档加密。以一个足够大的系统安全参数 P 为输入，输出系统参数 $params$ 和一个随机加密密钥 K，系统参数 $params$ 为 (n,t,E)。以用户文档 D、随机加密密钥 K 和系统参数 $params$ 为输入，用随机密钥 K 来加密用户文档 D，最后输出密文 C。

2. 密钥分量生成。根据 3.1.2 节介绍的秘密共享方案将密钥 K 分解成 n 个密钥分量 $(K_1,\cdots,K_i,\cdots,K_n)$，其中至少需要 t 个密钥分量才能重构出解密密钥 K。

3. 可信节点选择。Owner 在密钥分量分发之前先和代理 Agent 进行交互，从 DHT 节点信任值表中选择前 n 个信任值高的 DHT 节点。

4. 密钥分量分发。将这 n 个节点作为 n 个密钥分量的存储索引 $I_1,\cdots,I_i,\cdots,I_n$，得到密钥定位器 L。对于 $i=1,\cdots,n$，将 K_i 存储到 DHT 网络中索引为 I_i 的节点上。值得指出的是，因为 DHT 网络的节点可以自由退出，所以之前选定的前 n 个可信节点会存在部分节点不在线的情况，假如这部分不在线的节点数为 x，则可信节点定位器还需要从云端代理 Agent 处重新顺延获取 x 个节点，直到密钥分量分发成功。

5. 数据上传。以密钥定位器 L、密文数据 C 和系统参数 $params$ 作为输入，输出可以自毁的加密文档 SDD(Self-Destruct Document)，$SDD=(L,C,n,t)$。将 SDD 上传至 CSP 的服务器上。

6. 节点评价。当用户 u 与 DHT 网络中的节点 v 交互之后，首先使用3.3.1 节节中设计的评价算法对节点进行满意度评价。若节点不在线，则评价取值为 0。若节点在线，评价取值在 0～1 之间，值越大表示满意度越高。然后代理 Agent 使用 3.3.1 节设计的全局信任度计算算法计算并保存更新后的节点 v 的全局可信度。任何的评价都有有效期，一旦超过此有效期限，评价就失效，不计入节点信任值之内，可以设定有效期限为一个星期。

3.4.2　解密阶段

当 User 从 CSP 处获得最符合自己查询需求的密文文件后，通过密钥定位器获得密钥分量，基于秘密共享方案将获得的密钥分量恢复出解密密钥，解密得到明文文档，并且根据 DHT 节点的表现做出相应评价，其具体步骤如图 1 右所示。

1. 文档获取。User 通过授权后从 CSP 处获取自毁密文文档 $SDD=(L,C,n,t)$。

2. 密钥分量获取。以 $SDD=(L,C,n,t)$ 为输入，通过密钥定位器 L 获得 n 个密钥存储索引 $I_1,\cdots,I_i,\cdots,I_n$，然后通过这些密钥存储索引从 DHT 网络中获取 t 个或者 t 个以上的密钥分量。

3. 密钥恢复。验证从 DHT 网络中所获得的任意的 t 个密钥分量是否是正确的，若正确，则执行密钥恢复算法重构出解密密钥 K，若不正确，则重新获取密钥分量。

4. 文档解密。使用重构出的解密密钥 K 来解密密文数据 C，从而得到用户敏感数据 D。

5. 节点评价。当用户 u 与 DHT 网络中的节点 v 交互之后，首先使用 3.3.1 节设计的评价算法对节点进行满意度评价。如果发现节点提供的密钥分量是错误的，则评价取值为 0。如果发现节点提供的密钥分量是正确的，根据响应时间，评价取值在 0～1 之间，值越大表示满意度越高。然后代理 Agent 使用 3.3.1 节设计的全局信任度计算算法计算并保存更新后的节点 v 的全局可信度。

4　实验及分析

本节的主要目的：(1)验证本文提出的 DDTV 方案的可行性；(2)从密钥分发/重构时间和密钥分量提取的成功率两个方面，将本文提出的 DDTV 方案与原 Vanish 方案进行比较。

4.1　实验环境

云存储环境：实验中的云存储系统由 4 台计算机构成，其中一台为 NameNode，其他 3 台为 DataNode，服务器的配置均为 Intel Core(TM) 3.40GHz，内存 4G，硬盘 800G，操作系统均为 Ubuntu 10.10，Hadoop 版本为 0.20.2。客户端配置为英特尔酷睿双核 E7200@2.53GHz 处理器，2GB 内存，运行 Windows XP 操作系统。客户端加/解密程序基于 java 语言编写，数据对称加密采用 DES 算法。

密钥分发和恢复环境：使用 Stanford 大学 P2P 研究组开发的查询周期仿真器 QCS(Query Cycle Simulator)[14] 来模拟 DHT 网络，选用开源项目 Vanish[15] 的源代码连接 QCS DHT

网络。QCS DHT 网络实验参数如表 1 所列。

表 1　QCS DHT 网络参数配置

参数名称	参数值	参数名称	参数值
DHT 网络规模	100	密钥分量数	200
恶意节点邻居数	5	节点密钥分量数	4
正常节点邻居数	3	查询消息 TTL	5

4.2　实验结果及分析

实验的基本思路为：首先在客户端上生成密钥，使用密钥加密文件，其次上传密文到由 Hadoop 搭建的云存储环境中，再次将密钥经秘密共享方案处理后分发到 QCS DHT 网络中，然后从网络中提取并重构密钥，解密得到明文，最后利用 DHT 网络的动态特性，使得分发到网络中的密钥在 timeout 达到后不可用。

4.2.1　DDTV 方案中的客户端对文档的加密和解密

首先，我们在 DDTV 方案的客户端对文档进行加密，文档大小的变化范围是 4MB～256MB，加密的时间开销如表 2 所列。其次，将密文从 DDTV 方案的云端下载到 DDTV 方案的客户端，用恢复出来的密钥对相应的文档密文进行解密，解密对应的时间开销如表 2 所列。

表 2　不同文件大小加密解密时间

File size	Time/s(Encrypt)	Time/s(Decrypt)
4MB	0.584	0.632
8 MB	0.786	0.836
16MB	1.214	1.255
32MB	1.982	2.137
64 MB	3.691	3.79
128 MB	7.007	7.393
256 MB	14.035	14.286

图 2 显示了加解密操作所需时间的变化情况，其中“Encrypt”表示数据加密，“Decrypt”表示数据解密。从图中可以看出，文件的加解密时间与文件大小基本上呈线性关系。当文件在 256MB 时，加解密时间为 14s 左右，在可接受范围内。从以上实验数据可以看出，在 DDTV 方案的客户端对文档进行加密/解密是可行的。

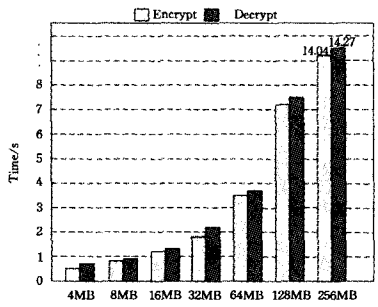

图 2　不同文件大小加密/解密时间

4.2.2　DDTV 方案中密文的上传和下载

在 DDTV 方案中，密文上传是指 DDTV 方案中的客户端将加密后的密文上传至 DDTV 方案的云端；密文下载是指将 DDTV 方案中云端的密文下载到 DDTV 方案的客户端。为了节省篇幅，只验证密文上传。

实验中，我们在 DDTV 方案中的客户端将加密后的密文上传至 DDTV 方案的云端，测试不同大小的密文文件传输时 DDTV 方案的云端与 DDTV 方案的客户端性能。

从图 3 和图 4 可以看出，云端和客户端的 CPU 占有率均随文件大小的增大而线性地增加。从以上实验可以看出，从 DDTV 方案的客户端上传密文到 DDTV 方案的云端是可行的。同理，从 DDTV 方案的云端将密文下载到 DDTV 方案的客户端也是可行的。

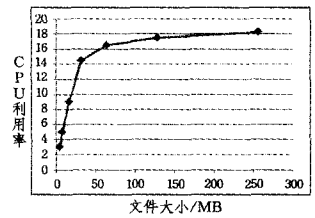

图 3　不同文件大小传输时云端 CPU 占用率

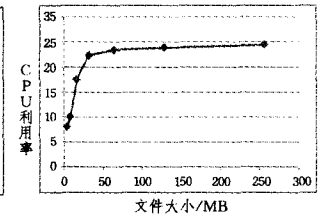

图 4　不同文件大小传输时客户端 CPU 占用率

4.2.3　DDTV 方案中密钥的分发与重构

在 DDTV 方案中，密钥分发是指 DDTV 方案中的客户端将密钥经秘密共享后分发至 DHT 网络中。在分发时会选择信任值高的节点存储。密钥重构是指从 DHT 网络中提取密钥分量后进行密钥恢复。在提取密钥时，会根据 DHT 节点信任值列表选最高全局信任值节点下载；若失败，从列表删除，重新选择最高信任值节点，直到成功为止。

(1)选择信任节点时间开销

相较于 Vanish 方案，DDTV 方案在密钥的分发和重构时多了从 DHT 节点信任值列表中选择信任值较高的节点的时间开销。DHT 节点信任值列表的生成首先需要获得所有节点的信任值，然后根据节点的信任值进行排序。测试不同节点数量时选择信任节点时间开销，如图 5 所示。

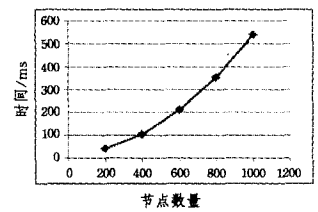

图 5　不同节点数量时选择信任节点时间开销

由图 5 可知，选择信任节点的时间开销与节点的数量成正比。节点数量越多，需要收集的节点信任值的时间越长，因此选择信任节点的时间开销越大。当 1000 个节点时，选择信任节点所用时间为 538ms，在可接受范围内。

(2)密钥分量提取成功的量

密钥分量提取成功的量是指在每个提取周期以后，100 个节点中提取密钥分量成功的节点数量(个数)。周期循环执行 100 次后密钥分量提取成功的量的对比如图 6 所示。

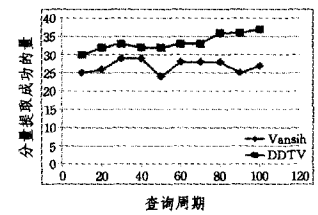

图 6　密钥分量提取成功的量

由图 6 可知，未使用信任模型的 Vanish 方案由于提供劣质服务的恶意节点无法被系统识别，服务请求节点随机选择

(下转第 154 页)

[16] Guo Wei, Su Wei, Lian Li, et al. MQL: A Mathematical Formula Query Language for Mathematical Search[C]// IEEE 14th International Conference on Computational Science and Engineering (CSE). IEEE, 2011: 245-250
[17] 景珂. 网络数学搜索中的数学查询语言与索引的研究[D]. 兰州: 兰州大学, 2009
[18] 崔林卫, 苏伟, 郭卫, 等. 基于 Nutch 的 Web 数学公式提取[J]. 广西师范大学学报: 自然科学版, 2011, 29(1)
[19] Srinivasan P, Menczer F, Pant G. A general evaluation framework for topical crawlers[J]. Information Retrieval, 2005, 8(3): 417-447
[20] Menczer F, Pant G, Srinivasan P. Topical web crawlers: Evaluating adaptive algorithms[J]. ACM Transactions on Internet Technology (TOIT), 2004, 4(4): 378-419
[21] 郑冬冬, 赵朋朋, 崔志明, 等. Deep Web 爬虫研究与设计[J]. 清华大学学报: 自然科学版, 2005, 45(9): 1896-1902
[22] 谭思亮. 聚焦爬行系统的设计—算法视角[D]. 成都: 中国科学院研究生院(成都计算机应用研究所), 2006
[23] Fuentes Sepúlveda J, Ferres L. Improving accessibility to mathematical formulas: the Wikipedia Math Accessor[J]. New Review of Hypermedia and Multimedia, 2012, 18(3): 183-204
[24] Abelson H, Dybvig R K, Haynes C T, et al. Revised report on the algorithmic language scheme[J]. ACM SIGPLAN Lisp Pointers, 1991, 4(3): 1-55

(上接第 112 页)

服务节点进行下载，因此密钥分量提取成功的量一直保持在较低水平，系统稳定性差。加入了信任模型的 DDTV 方案随着提取周期个数的增加，也就是提取次数的增加，节点之间的信任值计算逐渐准确，由此密钥分量提取成功的量也逐渐增加。

(3)密钥分量提取成功率

密钥分量提取成功率是整个系统的密钥分量提取成功次数在所有提取次数中所占的比例，它直观地反映了抵制恶意节点的抗攻击能力。为保证结果的准确性，每个方案的周期循环执行 20 次，采集每次循环得出的数据，并计算出平均值，结果如图 7 所示。

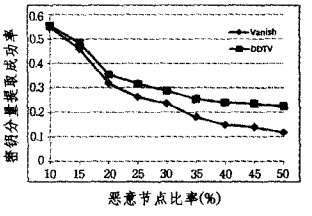

图 7 密钥分量提取成功率

由图 7 所示，当恶意节点比例特别低时，密钥分量提取成功率都接近于 58%。随着恶意节点比例的增加，在没有引入信任模型的情况下，节点缺乏抵制恶意节点的能力，表现为 Vanish 的仿真曲线迅速下降。相比 Vanish，DDTV 由于引入了信任模型，故能有效地抑制恶意节点，因而曲线的斜度更加平缓。

从以上的实验可以看出，DDTV 方案密钥的分发和重构虽然多了选择信任节点的时间，但时间开销在可接受的范围之内，而且同原来的 Vanish 方案相比，密钥分量提取成功率更高，使系统具有更高的可靠性。

结束语 本文设计了一种基于信任值的云存储数据确定性删除方案。在封装阶段，选择信任值较高的节点存放用户密钥分量。在解密阶段，从 DHT 网络中得到足够多的密钥分量后解密得到数据。在这两个阶段中均会根据节点的行为进行相应评价，综合这些评价就是该节点的信任值。引入信任值后，使用户密钥存放更安全，确保用户数据在授权时间内可用。实验结果表明，该方案是可行的，可以有效地抑制恶意节点，提高用户密钥分量提取成功率。接下来，将进一步完善我们的信任模型，使密钥分量提取成功率得到进一步提高。

参 考 文 献

[1] 武永卫, 黄小猛. 云存储[J]. 中国计算机学会通讯, 2009, 5(6): 44-52
[2] Kohno G T, Levy A, Levy H M. Vanish: Increasing data privacy with self-destructing data [C]// Proceedings of the 18th USENIX Seeurity Symposium. 2009
[3] Yue Feng-shun, Wang Guo-jun, Liu Qin. A secure self-destructing scheme for electronic data[C]// Proc of EUC2010. New York: IEEE Press, 2010: 651-658
[4] Zeng Ling-fang, Shi Zhan, Xu Sheng-jie, et al. Safevanish: An improved data self-destruction for protecting data privacy[C]// Proc of CloudCom 2010. New York: IEEE Press, 2010: 521-528
[5] 王丽娜, 任正伟, 余荣威. 一种适于云存储的数据确定性删除方法[J]. 电子学报, 2012(2): 266-273
[6] Perlman R. File System Design with Assured Delete [C]// SISW'05 Proceeding of the Third IEEE International Security in Storage Workshop. 2005: 83-88
[7] Tang Yang, Lee P P C, Lui J C S, et al. FADE: Secure overlay cloud storage with file assure ddeletion[C]// Proc of the SecureComm'10. New York: ACM Press, 2010. 380-397
[8] Stoica I, Morris R, Karger D, et al. Chord: A scalable peer-topeer lookup service for internet applications[C]// Proc of the SIGCOMM 2001. New York: ACM Press, 2001: 149-160
[9] Dabek F. A Distributed Hash Table [D]. Massachusetts: Massachusetts Institute of Technology, 2005
[10] Falkner J, Piatek M, John J, et al. Profiling a million user DHT [C]// Proc of the 7th ACM SIGCOMM Conference on Internet Measurement. New York: ACM Press, 2007: 129-134
[11] Rhea S, Godfrey B, Karp B, et al. OpenDHT: A public DHT service and its uses[C]// Proceedings of ACM SIGCOMM. 2005: 73-84
[12] Azureus[OL]. http://www.vuze.com/
[13] Shamir A. How to share a secret [J]. Communications of the ACM, 1979, 22(11): 612-613
[14] Dou W, Wang H M, Jia Y, et al. A recommendation-based Peer-to-Peer trust model[J]. Journal of Software, 2004, 15(4): 571-583
[15] The Stanford P2P sociology project[OL]. http://p2p.standford.edu/
[16] Vanish. [EB/OL]. http://vanish.cs.washington.edu/. 2011-07-29