

# 公开云环境中身份基代理远程数据完整性证明

王 新,秦敬源

(大连海洋大学 信息工程学院,辽宁 大连 116023)

**摘 要:**云计算作为一种传统计算的替代方式,发展迅速。因为它可以提供给和商业环境一种灵活的、动态的、有弹性的架构。在公共云环境中,客户端将其数据上传到公有云服务器 PCS 中,无法控制其远程数据,因此检查远程数据的完整性是至关重要的工作。在原始数据不需下载的情况下,确保用户能够检测其外包数据完整性。在某些情况下,客户没有能力检查其远程数据的完整性,不得不将远程数据完整性检测任务委托给其代理。基于双线性对和计算 Diffie-Hellman 问题的困难性,设计了身份基代理远程数据完整性证明协议。在该协议中采用身份基的密码体系,减少了对公钥证书的验证,降低了用户和云服务器的储存、通信开销。通过安全性分析和性能分析,表明该协议是可证安全的和高效的。

**关键词:**公开云;身份基;代理;完整性检验

**中图分类号:**TP39

**文献标识码:**A

**文章编号:**1673-629X(2017)12-0093-05

**doi:**10.3969/j.issn.1673-629X.2017.12.021

## Remote Data Integrity Checking of Identity-based Proxy in Public Cloud Environment

WANG Xin, QIN Jing-yuan

(School of Information Engineering, Dalian Ocean University, Dalian 116023, China)

**Abstract:** Cloud computing as an alternative to conventional computing has a rapid development because it can provide a flexible, dynamic and resilient infrastructure for both academy and business. In public cloud environment, the clients move their data to Public Cloud Server (PCS). Since remote data are uncontrollable, checking its integrity is of crucial importance. It enables the clients to check whether their outsourced data have been kept intact without downloading the original data. In some cases, the clients are failed to check the integrity of remote data so that they have to delegate their proxy for checking of remote data integrity. Based on the bilinear pairing technique and the hardness of computational Diffie-Hellman problem, a protocol of remote data integrity checking of identity-based proxy is designed where identity-based cryptography system is used to reduce authentication of public key certificates and lower the storage and communication costs of user and public cloud server. Through analysis of security and performance, the protocol proposed is provably secure and efficient.

**Key words:** public cloud; identity-based; proxy; integrity checking

## 1 概 述

### 1.1 研究背景

云存储是云计算的一种重要服务,允许数据所有者将数据存储在云服务器中,并通过云服务器向用户提供数据访问。

这种数据的外置服务在给数据所有者带来优势和便利的同时,也带来如下问题<sup>[1-2]</sup>:

首先,数据所有者将数据存储在云中后,就失去了对数据的物理控制,导致数据的安全性高度依赖于云服务提供商。所以并不能保证数据所有者数据的完

整性。

其次,虽然云存储的访问没有物理地点的限制,但在某些情况下,数据所有者会被限制访问互联网而没有能力检查其远程数据。这时,数据所有者将远程数据检查的任务委托给代理,代理根据授权执行远程数据完整性检测协议。当云服务器的响应无法通过验证时,代理将联系云服务提供商,评估损失并根据损失的严重程度讨论赔偿。

由于身份基公钥密码学具有效率方面的优势,不需要进行公钥证书验证,研究人员将身份基公钥密码

收稿日期:2016-11-30

修回日期:2017-04-05

网络出版时间:2017-08-01

基金项目:辽宁省自然科学基金(20102042)

作者简介:王 新(1984-),男,硕士研究生,研究方向为密码学。

网络出版地址: <http://www.cnki.net/kcms/detail/61.1450.TP.20170801.1554.056.html>

学引入云计算安全的研究中。综合国内外对云计算、远程数据完整性检测、身份基公钥密码学的研究现状<sup>[3-5]</sup>,不论从理论上还是从应用前景上看,开展公共云环境中身份基代理数据完整性检测的研究都具有重要的学术价值以及广阔的应用前景。

## 1.2 相关工作

2007 年,Ateniese 等<sup>[6]</sup>首次提出了“数据拥有性证明”(Provable Data Possession, PDP) 方案。在该方案中,用户可以以一个很高的概率发现远程服务器上的数据存储错误。方案的设计基于 RSA 公钥加密算法,因此计算开销很大,同时,方案仅支持静态数据检验。后来,他们扩展了原有方案<sup>[7]</sup>,提出了一种动态版本的 PDP 方案,但新版本也不支持完整的动态操作,如数据插入操作。2008 年,Sebe 等<sup>[8]</sup>提出了一种可以进行无限次验证的远程数据完整性检验方案,支持在设置时间与用户存储开销间进行权衡。2009 年,Erway 等<sup>[9]</sup>提出了两种动态的 PDP 方案,其中基于 RSA 树的认证哈希字典结构的方案通过更大的计算开销提高了错误检验概率。

2010 年,Wang 等<sup>[10]</sup>提出了实现隐私保护的公共审计方案。该方案利用同态密钥随机掩码技术保证第三方审计不能从验证过程中获得用户的任何有用信息。2012 年,Zhu 等<sup>[11]</sup>将 PDP 的使用环境从单云扩展到多云。2013 年,Wang<sup>[12]</sup>提出一种代理 PDP 方案。在该方案中,用户将授权证据嵌入到数据块标签的生成中,以达到只有拥有授权的代理才能完成数据完整性验证的目的,但是方案没有考虑授权的撤销问题。2015 年,Wang<sup>[13]</sup>将基于身份加密体制引入到 PDP 模型中,设计了一种多云环境下的数据完整性证明方案。

代理公共密钥加密技术已经研究了多年。Mambo 等<sup>[14]</sup>提出了代理签名的概念。之后 Hwang 等<sup>[15]</sup>提出了一个基于 RSA 的门限代理签名方案。Libert 等<sup>[16]</sup>提出了单向密文选择安全代理重加密。Pack 等<sup>[17]</sup>引入了代理缓存。由于代理有大量的应用领域,所以在云计算领域研究代理密码学很重要。

## 2 预备知识

### 2.1 双线性对

设  $G$  和  $G_T$  是两个阶为素数  $q$  的乘法群,  $g$  是  $G$  的生成元,双线性映射  $e: G \times G \rightarrow G_T$  满足下述性质:

(1) 可计算性:对任意的  $u, v \in G$ , 存在有效的算法计算  $e(u, v)$ ;

(2) 双线性:对于任意的  $a, b \in \mathbb{Z}_q$ , 都满足  $e(g^a, g^b) = e(g, g)^{ab}$ ;

(3) 非退化性:存在  $u, v \in G$ , 满足  $e(u, v) \neq 1$ , 其中 1 代表群  $G_T$  的单位元。

双线性映射  $e$  可以通过有限域上椭圆曲线的 Weil 配对<sup>[18]</sup>或者 Tate 配对<sup>[19]</sup>来构造。

### 2.2 计算 Diffie-Hellman 问题

$g$  是  $G$  的生成子,给出  $g, g^a, g^b \in G$ , 其中  $a, b \in \mathbb{Z}_q^*$  未知,计算  $g^{ab}$  的概率是可忽略的。该问题称为计算 Diffie-Hellman 问题。

## 3 构造身份基代理远程数据完整性检测方案

提出的协议包含 7 个步骤: Setup, Extract, TagGen, SignVerify, CheckTag, GenProof, CheckProof。

首先,引入一些符号。假设存储的块-标签对的最大数为  $n$ ,  $f$  和  $\Omega$  是两个伪随机函数,  $\pi$  是一个伪随机置换函数,  $h$  是一个加密哈希函数,  $H$  是一个有两个输入的加密哈希函数,详细表述如下:

$$\begin{aligned} f: \mathbb{Z}_q^* \times \{1, 2, \dots, n\} &\rightarrow \mathbb{Z}_q^* \\ \Omega: \mathbb{Z}_q^* \times \{1, 2, \dots, n\} &\rightarrow \mathbb{Z}_q^* \\ \pi: \mathbb{Z}_q^* \times \{1, 2, \dots, n\} &\rightarrow \{1, 2, \dots, n\} \\ H: G_T \times \{0, 1\} &\rightarrow \mathbb{Z}_q^* \\ h: \mathbb{Z}_q^* &\rightarrow G^* \end{aligned}$$

$f_z(i)$  定义为有  $z$  和  $i$  输入的函数  $f$ ,  $\Omega_z(i)$  定义为  $z$  和  $i$  输入的函数  $\Omega$ ,  $g$  作为  $G$  的生成子。假设文件  $F$  被分成  $n$  块  $(m_1, m_2, \dots, m_n)$ , 其中  $m_i \in \mathbb{Z}_q^*$ ,  $q$  是  $G$  和  $G_T$  的阶。不失普遍性,定义  $F = (m_1, m_2, \dots, m_n)$ 。PCS 选取一个随机数  $y \in \mathbb{Z}_q^*$  作为私钥,计算出  $Y = g^y$  作为公钥。Proxy 选取一个随机数  $z \in \mathbb{Z}_q^*$ , 计算出  $Z = g^z$ , 把  $(z, Z)$  作为 Proxy 的私钥-公钥对。

方案的详细步骤如下:

(1) Setup: PKG (Private Key Generator) 选取一个随机数  $x \in \mathbb{Z}_q^*$ , 计算出  $X = g^x$ 。然后, PKG 选择一个随机元素  $u \in G^*$ 。PKG 将  $\{G, G_T, e, q, g, X, Y, Z, u, f, \Omega, \pi, H, h\}$  作为系统公开参数,将  $x$  作为主密钥保存。

(2) Extract: Client 将它的身份 ID 发送给 PKG。PKG 选取一个随机数  $r \in \mathbb{Z}_q^*$ , 然后计算  $R = g^r$ ,  $\sigma = r + xH(\text{ID}, R) \bmod q$ 。PKG 将密钥  $\text{sk}_{\text{ID}} = (R, \sigma)$  通过安全通道发送给 Client, Client 通过计算  $g^\sigma = R \cdot X^{(\text{ID}, R)}$  是否成立来确定密钥的正确性。如果等式成立, Client 就接受该密钥,并将  $R$  作为公钥公开;反之则拒绝。

Client 通过发送一个证书来授予 Proxy 进行远程数据完整性检测。这个证书由远程数据完整性检测授权的证据  $(C, w)$  (其中  $C = X^{(\text{ID}, R)}$ ) 和在 Client 的私钥  $\sigma$  下用  $(C, w)$  签名的证书 Sign 组成。一旦被指定, Proxy 可以通过自己的私钥  $z$  和  $((C, w), \text{Sign})$  来检查 Client 的远程数据。远程数据完整性检测涉及了与  $R$  相关的  $(C, w)$  的有效性验证,同时也包含了检查 Proxy 是否符

合指定的  $(C, w)$ 。另一方面,  $(\text{Sig}, \text{SignVerify})$  是一个签名-验证算法对, 并且  $\text{Sign} = \text{Sig}_R(C, w)$ 。Client 发送这个证书-证明对  $((C, w), \text{Sign})$  给 Proxy。

(3) TagGen: 对于  $F = (m_1, m_2, \dots, m_n)$ , Client 生成每个块  $m_i$  的标签  $T_{m_i}$ , 步骤如下:

- ① Client 计算  $t = H(e(Y, Z)^\sigma, w)$ ,  $\hat{W}_i = \Omega_i(i)$ ;
- ② Client 计算  $T_{m_i} = (h(\hat{W}_i) u^{m_i})^\sigma$ ;
- ③ Client 输出  $(T_{m_i}, m_i)$ ;
- ④ Client 计算关于  $w$  的签名  $\text{Sign} = \text{Sig}_R(C, w)$ 。

执行了  $n$  次以上步骤之后, 所有的块标签都生成了, 将它们表示成一个集合  $\Sigma = (T_{m_1}, T_{m_2}, \dots, T_{m_n})$ 。然后 Client 将块-标签对集合  $\{(m_i, T_{m_i}), 1 \leq i \leq n\}$  和证书  $(C, w)$  发送给 PCS。PCS 存储这些块标签对和证书, 之后 Client 在其本地储存中删除这些块-标签对。

(4) SignVerify: 根据收到的 Client 关于证书  $(C, w)$  的签名  $\text{Sign} = \text{Sig}_R(C, w)$ , Proxy 执行验证算法  $\text{SignVerify}((C, w), \text{Sign}, R)$ 。如果有效, Proxy 就接受证书  $(C, w)$ ; 否则 Proxy 就拒绝, 并且向 Client 请求一个新的证书-证明对。

(5) CheckTag: 对于收到的  $\{(m_i, T_{m_i}), 1 \leq i \leq n\}$ , 执行如下步骤:

- ① PCS 计算  $t = H(e(R \cdot C, Z)^\gamma, w)$ ,  $\hat{W}_i = \Omega_i(i)$ ;
- ② PCS 验证  $e(T_i, g) = e(h(\hat{W}_i) u^{m_i}, R \cdot C)$  是否成立;
- ③ 如果上式成立, PCS 接受这个块-标签对; 否则 PCS 拒绝接受, 并且向 Client 请求一个新的块-标签对。

(6) GenProof: 设定一个挑战  $\text{chal} = (c, k_1, k_2)$ , 其中  $1 \leq c \leq n, k_1 \in Z_q^*, k_2 \in Z_q^*$ 。在这一阶段, Proxy 向 PCS 请求  $c$  个远程数据持有证明文件块, 这些文件块由为每个挑战随机刷新关键字的伪随机置换函数随机选出。这样可以防止被检测的块在任何一次挑战中被预知。 $k_1$  是伪随机置换函数  $\pi$  的随机关键字,  $k_2$  是伪随机函数  $f$  的随机关键字。

Proxy 将  $((C, w), \text{Sign})$  发送给 PCS, PCS 验证签名  $\text{Sig}_R(C, w)$  是否有效。如果有效, 则 PCS 比较这个证书与其存储的  $(C, w)$  是否一致。当  $(C, w) = (C, w)'$  并且 Proxy 的请求符合其证书  $(C, w)$  时, PCS 执行如下步骤 (否则 PCS 拒绝 Proxy 的请求):

① 对于  $1 \leq j \leq c$ , PCS 计算选取文件块的参数, 通过函数:  $i_j = \pi_{k_1}(j), a_j = f_{k_2}(j)$ ;

② PCS 计算  $T = \prod_{j=1}^c T_{m_{i_j}}^{a_j}, m = \sum_{j=1}^c a_j m_{i_j}$ ;

③ PCS 输出  $V = (m, T)$ , 将  $V$  作为  $\text{chal}$  请求的回复发送给 Proxy。

(7) CheckProof: 根据从 PCS 收到的回复  $V$ , Proxy 执行如下步骤:

① Proxy 计算  $t = H(e(R \cdot C, Y)^z, w)$ ;

② Proxy 验证  $e(T, g) = e(\prod_{i=1}^c h(\Omega_i(\pi_{k_1}(i)))^{f_{k_2}(i)})^{f_{k_2}(i)} u^m, R \cdot C)$  是否成立;

③ 如果上式成立, Proxy 输出 “success”, 否则输出 “failure”;

④ 当 PCS 的回复不能通过 Proxy 的验证时, Proxy 将会多次重复执行同样的挑战。如果回复仍然不能通过验证, Proxy 会联系 PCS 报告情况。PCS 会检查 Client 存储的数据, 并且通过离线备份来恢复丢失的数据。如果 PCS 失败了, Proxy 和 PCS 将不得不评估损失, 以及根据损失程度来讨论赔偿。

## 4 方案分析

### 4.1 正确性分析

定理 1: 如果 Client 和 PCS 都是诚实的, 而且能够遵守上文的步骤, 那么任何块-标签对都可以通过 PCS 的标签检查。

证明: 根据 TagGen 和 CheckTag 两个步骤, 可知:

$$\begin{aligned} t &= H(e(R \cdot C, Z)^\gamma, w) = H(e(g, g)^{\gamma\sigma}, w) = \\ &= H(e(Y, Z)^\sigma, w) = t \end{aligned}$$

$$\hat{W}_i = \Omega_i(i) = \Omega_i(i) = W_i$$

那么

$$\begin{aligned} e(T_i, g) &= e((h(\hat{W}_i) u^{m_i})^\sigma, g) = \\ &= e((h(\hat{W}_i) u^{m_i})^\sigma, g) = e(h(\hat{W}_i) u^{m_i}, g^\sigma) = \\ &= e(h(\hat{W}_i) u^{m_i}, g^{r+\gamma H(\text{ID}, R)}) = e(h(\hat{W}_i) u^{m_i}, R \cdot C) \end{aligned}$$

定理成立。

定理 2: 如果 Proxy 和 PCS 都是诚实的, 而且能够遵守上文的步骤, 那么回复  $V$  可以通过 Proxy 的数据持有检查。

证明: 设定一个挑战为  $\text{chal} = (c, k_1, k_2)$ , 根据 TagGen 和 GenProof 两个步骤, 可知:

$$\begin{aligned} t &= H(e(R \cdot C, Z)^\gamma, w) = H(e(g, g)^{\gamma\sigma}, w) = \\ &= H(e(Y, Z)^\sigma, w) = t \end{aligned}$$

$$\hat{W}_i = \Omega_i(i) = \Omega_i(i) = W_i$$

那么

$$\begin{aligned} e(T, g) &= e(\prod_{j=1}^c T_{m_{i_j}}^{a_j}, g) = e(\prod_{j=1}^c (h(\hat{W}_{i_j})^{a_j})^\sigma u^{\sum_{j=1}^c a_j m_{i_j}}, g) = \\ &= e(\prod_{j=1}^c (h(\hat{W}_{i_j})^{a_j})^\sigma u^{\sum_{j=1}^c a_j m_{i_j}}, g^\sigma) = \\ &= e(\prod_{j=1}^c h(\Omega_i(i_j))^{f_{k_2}(j)} u^m, g^{r+\gamma H(\text{ID}, R)}) = \end{aligned}$$

$$e(\prod_{j=1}^c h(\Omega_i(\pi_{k_i}(j)))^{f_{k_i}(j)} u^m, R \cdot C)$$

定理成立。

定理 3: 如果计算 Diffie-Hellman 问题在  $G$  上成立, 在随机预言模型下, 本协议是不可被欺骗的。

证明: 如果 PCS 存储的某些被挑战的文件数据和标签被破坏, 而 PCS 仍然能够生成通过  $e(T, g) = e(\prod_{i=1}^c h(\Omega_i(\pi_{k_i}(i)))^{f_{k_i}(i)} u^m, R \cdot C)$  验证的证明, 则需要成功伪造标签。然而, 如果 PCS 或其他实体能够在没有 Client 私钥的情况下生成可通过验证的标签, 这需要成功计算 Diffie-Hellman 问题。由计算 Diffie-Hellman 问题的困难性可知, 成功的概率可以忽略, 定理成立。

定理 4: 当 PCS 存储了  $n$  个块-标签对  $((m_1, T_1), (m_2, T_2), \dots, (m_n, T_n))$ ,  $d$  为篡改了的块-标签对的个数, 挑战为  $\text{chal} = (c, k_1, k_2)$  时, 发现篡改的概率  $P_R$  满足:

$$1 - (\frac{n-d}{n})^c \leq P_R \leq 1 - (\frac{n-c+1-d}{n-c+1})^c$$

则本协议是  $(\frac{d}{n}, 1 - (\frac{n-d}{n})^c)$  安全的。

证明: 假定 PCS 在  $n$  个块-标签对中有  $d$  个被篡改了。挑战  $\text{chal} = (c, k_1, k_2)$  中代理请求的块数量为  $c$ 。设  $R$  是一个离散随机变量, 定义为被 PCS 篡改的块-标签对和由代理选择的块匹配的数量。将  $P_R$  定义为 PCS 篡改的块-标签对和由代理选择的块相匹配至少为一个的概率。于是有:

$$P_R = P\{R \geq 1\} = 1 - P\{R = 0\} = 1 - \frac{n-d}{n} \frac{n-1-d}{n-1} \dots \frac{n-c+1-d}{n-c+1}$$

因此, 可得:

$$1 - (\frac{n-d}{n})^c \leq P_R \leq 1 - (\frac{n-c+1-d}{n-c+1})^c$$

概率曲线如图 1 所示。

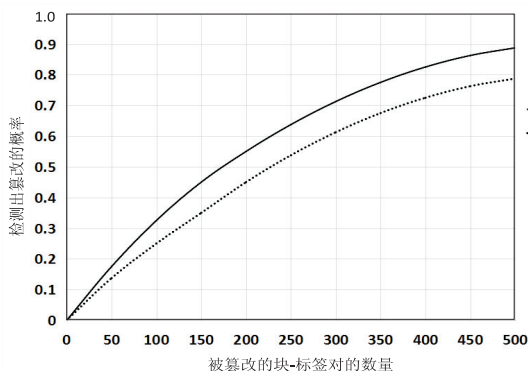


图 1 概率曲线

定理 5: 如果 PCS 事先对所有可能的挑战计算出响应后删除数据, 那么 PCS 将会消耗更多的存储空间。

对于  $n$  个块, 对所有可能的挑战的响应包含了  $2^n - 1$  个块-标签对。因为在  $n \geq 2$  的情况下,  $2^n - 1$  始终大于  $n$ , 对于理性的 PCS 是不可行的。

证明: 假设 Client 储存了  $n$  个块-标签对在 PCS 上。挑战为  $\text{chal} = (c, k_1, k_2)$ , 其中  $c (1 \leq c \leq n)$  是块-标签对的请求数。因为  $\pi$  是一个具有陷门  $k_1$  的随机置换函数, 则存在  $C_n^c$  种不同的块-标签挑战组合。因为随机数  $c$  取值为  $\{1, 2, \dots, n\}$ , 那么可能的块-标签挑战组合数量是  $C_n^1 + C_n^2 + \dots + C_n^n = 2^n - 1$ 。如果 PCS 事先对所有可能的挑战计算出响应后删除数据, PCS 需要储存  $2^n - 1$  个响应, 也就是  $2^n - 1$  个聚合块-标签对。因为在  $n \geq 2$  的情况下,  $2^n - 1$  始终大于  $n$ , PCS 需要花费比存储原始块-标签对更多的空间来存储这些聚合块-标签对。需要的空间随块-标签对的数量  $n$  呈指数增长, 因此对于理性的 PCS 是不可行的。

## 4.2 效率分析

### 4.2.1 计算效率

在文中方案 (ID-PPDP), 假设有  $n$  个消息块。在 Extract 阶段, PKG 进行一次幂运算, Client 进行两次幂运算, 一次  $G$  上的点乘运算。在 TagGen 阶段, Client 进行一次双线性对运算、 $2n$  次幂运算、 $n$  次  $G$  上的点乘运算和一次证书  $(C, w)$  的签名运算。在 SignVerify 阶段, Proxy 进行一次验证签名 Sign 的运算。在 CheckTag 阶段, PCS 进行  $2n + 1$  次双线性对运算、 $n$  次幂运算、 $n$  次  $G$  上的点乘运算。在 GenProof 和 CheckProof 阶段, PCS 进行  $c$  次幂运算、 $c - 1$  次  $G$  上的点乘运算, Proxy 进行 2 次双线性对运算、 $c + 1$  次幂运算、 $c$  次  $G$  上的点乘运算。其他类似于 hash, 置换一类的运算可以忽略, 因为它们对计算量的影响微不足道。

### 4.2.2 通信效率

与公钥加密算法 (RSA) 相比, 椭圆曲线密码体制 (ECC) 在相同的安全等级上有更短的密钥长度<sup>[20-22]</sup>。160 位的 ECC 密钥可以提供与 1 024 位的 RSA 密钥相同的安全等级。同样, 224 位的 ECC 密钥可以提供与 2 048 位的 RSA 密钥相同的安全等级。ID-PPDP 的通信开销主要发生在请求和响应阶段。在请求阶段, Proxy 需要将取自  $Z_q^*$  中的三个元素发送给 PCS。在响应阶段, PCS 需要将取自  $G$  的一个元素和取自  $Z_q^*$  的一个元素发送给 Proxy。总的通信量大约为 960 位。这种通信开销在现在的科技条件下是完全可容许的。在同样的过程中, Ateniese 等提出的方案需要 6 144 ( $6 * 1 024$ ) 位的通信量。在那两种方案中, 不考虑存储在 PCS 上数据的通信开销。而在存储过程中, 通信开销是相似的。因此, 在通信开销方面, ID-PPDP 比 Ateniese 等提出的方案<sup>[6]</sup> 更加高效。不同方案的比较如表 1 所示。



表 1 方案对比

方案	请求	响应	本地储存
Ateniese 等提出 的方案 <sup>[6]</sup>	$4Z_N^*$ (4 096 bit)	$2Z_N^*$ (2 048 bit)	$O(1)$
Sebe 提出的 方案 <sup>[8]</sup>	$2Z_N^*$ (2 048 bit)	$1Z_N^*$ (1 024 bit)	$O(n)$
ID-PPDP	$3Z_q^*$ (480 bit)	$1G + 1Z_q^*$ (480 bit)	$O(1)$

5 结束语

文中提出了身份基代理远程数据持有证明,给出了系统模型和安全模型,然后设计了基于双线性对的高效的协议,并且通过安全和性能分析证明了该协议的安全性和高效性。

参考文献:

[1] 冯登国,张敏,张妍,等. 云计算安全研究[J]. 软件学报,2011,22(1):71-83.

[2] 张玉清,王晓菲,刘雪峰,等. 云计算环境安全综述[J]. 软件学报,2016,27(6):1328-1348.

[3] 冯朝胜,秦志光,袁丁云. 数据安全存储技术[J]. 计算机学报,2015,38(1):150-163.

[4] 胡德敏,余星. 云存储服务中支持动态数据完整性检测方案[J]. 计算机应用研究,2014,31(10):3056-3060.

[5] 沈文婷,于佳,杨光洋,等. 具有私钥可恢复能力的云存储完整性检测方案[J]. 软件学报,2016,27(6):1451-1462.

[6] Ateniese G, Burns R, Curtmola R, et al. Provable data possession at untrusted stores[C]//Proceedings of 14th ACM conference on computer and communication security. [s. l.]: ACM, 2007:598-609.

[7] Ateniese G, Dipietro R, Mancini L V, et al. Scalable and efficient provable data possession[C]//Proceedings of fourth international conference on security and privacy in communication networks. [s. l.]: [s. n.], 2008.

[8] Sebe F, Domingo-Ferrer J, Martinez-Balleste A, et al. Efficient remote data integrity checking in critical information infrastructures[J]. IEEE Transactions on Knowledge and Data Engineering, 2008, 20(8):1034-1038.

[9] Erway C C, Kucpau A, Papamanthou C, et al. Dynamic provable data possession[C]//Proceedings of 16th ACM conference on computer and communication security. [s. l.]: ACM, 2009;

213-222.

[10] Wang C, Wang Q, Ren K, et al. Privacy-preserving public auditing for data storage security in cloud computing[C]//29th IEEE international conference on computer communications. [s. l.]: IEEE, 2010:1-9.

[11] Zhu Y, Hu H, Ahn G J, et al. Cooperative provable data possession for integrity verification in multicloud storage[J]. IEEE Transactions on Parallel and Distributed Systems, 2012, 23(12):2231-2244.

[12] Wang H. Proxy provable data possession in public clouds[J]. IEEE Transactions on Services Computing, 2013, 6(4):551-559.

[13] Wang H. Identity-based distributed provable data possession in multi-cloud storage[J]. IEEE Transactions on Services Computing, 2015, 8(2):328-340.

[14] Mambo M, Usuda K, Okamoto E. Proxy signatures for delegating signing operation[C]//Proceedings of third ACM conference on computer and communications security. [s. l.]: ACM, 1996:48-57.

[15] Hwang M, Lu J, Lin E. A practical (t, n) threshold proxy signature scheme based on the RSA cryptosystem[J]. IEEE Transactions on Knowledge and Data Engineering, 2003, 15(6):1552-1560.

[16] Libert B, Vergnaud D. Unidirectional chosen-ciphertext secure proxy re-encryption[J]. IEEE Transactions on Information Theory, 2011, 57(3):1786-1802.

[17] Pack S, Rutagemwa H, Shen X, et al. Proxy-based wireless data access algorithms in mobile hotspots[J]. IEEE Transactions on Vehicular Technology, 2008, 57(5):3165-3177.

[18] Boneh D, Franklin M. Identity-based encryption from the weil pairing[C]//Proceedings of 21st annual international cryptology conference. [s. l.]: [s. n.], 2001:213-229.

[19] Miyaji A, Nakabayashi M, Takano S. New explicit conditions of elliptic curve traces for fr-reduction[J]. IEICE Transactions on Fundamentals Electronics Communications & Computer Sciences, 2001, 84(5):1234-1243.

[20] Kumanduri R. Number theory with computer applications[M]. [s. l.]: Prentice Hall, 1998.

[21] Miller V. Uses of elliptic curves in cryptography[C]//Proceedings of fifth annual international cryptology conference. [s. l.]: [s. n.], 1985:417-426.

[22] Vanstone S. Responses to NISTs proposal[J]. Communication of ACM, 1992, 35:50-52.

(上接第 92 页)

[11] 李明,刘士仪,年福忠. 基于时序描述逻辑的 Web 服务本体语言过程模型语义[J]. 计算机应用,2013,33(1):266-269.

[12] 荣先球. 基于描述逻辑的 UML 行为图的形式化研究[D]. 兰州:兰州理工大学,2012.

[13] Baader F, Bauer A, Lippmann M. Runtime verification using a temporal description logic[C]//Proceedings of 7th international conference on frontiers of combining systems. Berlin;

Springer-Verlag, 2009:149-164.

[14] Vesely W E, Goldberg F F, Roberts N H, et al. Fault tree handbook[M]. United States: U. S. Nuclear Regulatory Commission, 1981.

[15] Schamann J M. Automated theorem proving in software engineering[M]. Berlin: Springer-Verlag, 2001.

[16] Huth M, Ryan M. Logic in computer science: modeling and reasoning about systems[M]. Cambridge, UK: Cambridge University Press, 2004.