对类型可修改的基于身份代理重加密方案的改进

张新鹏[1,2]，许春香[1]，张晓均[1]，邓 江[1]，黄 新[2]

(1. 电子科技大学计算机科学与工程学院 成都 611731；2. 成都军区联勤部信息中心 成都 610015)

【摘要】类型可修改的基于身份的代理重加密方案不仅具有传统代理重加密方案的核心功能，而且密文的拥有者可以随时修改密文的类型信息，能够满足实际云存储应用中，密文类型信息需要动态转换的应用场景。对类型可修改的基于身份代理重加密方案进行分析，发现该方案存在2个安全漏洞：1) 类型修改缺乏验证，攻击者可以随意修改类型标记；2) 类型修改引起了新的条件性选择明文攻击问题。在分析这两个安全漏洞的基础上，提出了改进方案，并给出了安全性分析。

关 键 词 云存储；可证明安全性；代理重加密；基于类型和身份的代理重加密

中图分类号 TP309 **文献标志码** A doi:10.3969/j.issn.1001-0548.2016.06.015

Further Improvement of a Dynamic Type and Identity-Based Proxy Re-Encryption Scheme

ZHANG Xin-peng[1,2], XU Chun-xiang[1], ZHANG Xiao-jun[1], DENG Jiang[1] and Huang Xin[2]

(1. School of Computer Science and Engineering, University of Electronic Science and Technology of China Chengdu 611731;

2. Logistic Information Center, Joint Logistics Department, Chengdu Military Region Chengdu 610015)

Abstract Dynamic type information of ciphertext can be modified properly so that it can be well applied in a practical cloud storage environment. In order to meet the application requirements, Liu *et al* proposed a dynamic type and identity-based proxy re-encryption (PRE) scheme based on Ibraimi *et al*'s scheme. Their scheme not only keeps the traditional core function of PRE scheme, but also makes sure that the owner of ciphertext can modify the type information at any time. However, after careful security analysis it found that Liu *et al*.'s scheme has two security flaws. Firstly, the dynamic type information lacks of verification, the adversary can modify the type tag. Secondly, the dynamic type information causes a conditional chosen plaintext attack. Thus we further improve Liu *et al*.'s scheme and give the security analysis.

Key words cloud storage; provable; proxy re-encryption; type and identity-based proxy re-encryption security

1 简述

云存储中，代理重加密(PRE)[1]技术可以保障用户数据在存储第三方的安全性和可共享性。该技术的核心思想是：数据拥有者以密文形式将数据存储在第三方；数据拥有者可以委托存储第三方对其存储的密文进行重加密并共享给其他用户。

针对现实应用，基于文献[2-3]的类型和身份的代理重加密方案，文献[4]提出了一种类型可修改的基于身份代理重加密方案[5-8]。该方案不但实现PRE中细粒度的密文共享，还解决了原方案密文类型信息静态，不能动态修改问题。文献[4]发现了密文类型信息动态修改的现实意义与应用场景。如数据拥有者(data owner, DO)为节省本地存储开销，或方便数据在不同终端的灵活使用，将标记为“不可共享”类型的加密数据存储于storage。DO根据现实需要希望将“不可共享”类型修改为“可共享”类型；一段时间后又需要将“可共享”类型恢复为“可共享”。类型信息的动态修改就可以很好的适应此类应用。

文献[4]虽然具有很高的现实应用价值，并且根据其安全性分析可知具有与文献[2]相同的安全性。但是在对其所提方案进行仔细研究后发现，在完成密文类型信息的修改后并没有对密文类型信息进行验证，如图1所示。其方案存在两个安全漏洞：1) 类型修改缺乏验证，攻击者可以随意修改类型标记，而不会被接收方发现。2) 由于类型修改引起了新的条件性选择明文攻击问题。图中实例所涉及的通信实体与文献[4]保持一致。DO、数据共享者

收稿日期：2015 – 03 – 23；修回日期：2015 – 06 – 05
基金项目：国家自然科学基金(61370203)；总后研究所项目(BS211L019-3)
作者简介：张新鹏(1978 –)，男，博士，主要从事信息安全密码学方面的研究.

(receiver, R)、密钥生成中心(KGC)、第三方存储器(storage)。

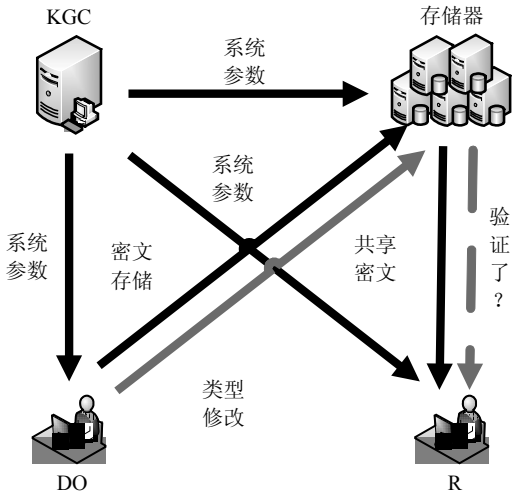

图1　文献[4]的方案基本工作流程

2　类型可修改的基于身份代理重加密方案的安全漏洞分析

2.1　类型修改缺乏验证

文献[4]方案在文献[2-3]的基础上，添加了密文类型的修改功能。在密文类型修改阶段中DO可以生成用于修改密文类型的密钥，并将其发送给storage; 根据接收到的密钥，storage可以修改密文类型的信息，并生成新的预共享密文。在经过密文共享阶段的重加密后再生成共享密文，最终实现数据的安全共享。仔细分析整个工作流程，在添加了密文类型修改阶段后，该方案直接使用了原来的密文共享步骤，并没有在后面重加密和解密步骤对修改后的密文类型的状态做出相应的验证。整个方案的完整性遭到了破坏。

由于类型修改缺乏验证，攻击者可以随意修改密文类型标记t'。将修改后t'代入后面的代理重加密过程和解密过程。如下面推导过程所示t'是否被修改，与这两个过程是否正确执行并不相关。相应的服务器、数据持有者、数据共享对象也无法及时地发现密文类型是否被攻击者所修改。

1) 攻击者截获算法8)[4]中，由数据拥有者执行算法$\mathrm{TKey}(t,t',\mathrm{SK}_{\mathrm{ID}_i})$生成的并传送于云存储服务器的修改密文类型密钥，有：

$$\mathrm{TK}=(t',\mathrm{SK}_{\mathrm{ID}_i}^{-H_2(\mathrm{SK}_{\mathrm{ID}_i},t)}\mathrm{SK}_{\mathrm{ID}_i}^{H_2(\mathrm{SK}_{\mathrm{ID}_i},t')})$$

将t'修改成t''后传于云存储服务器。

2) 云存储服务器执行算法9)[4]中的$\mathrm{TSet}(c,\mathrm{TK})$算法为：

$$c_1'=c_1 \qquad c_3'=t''$$

$$c_2'=c_2\hat{e}(c_1,\mathrm{SK}_{\mathrm{ID}_i}^{-H_2(\mathrm{SK}_{\mathrm{ID}_i},t)}\mathrm{SK}_{\mathrm{ID}_i}^{H_2(\mathrm{SK}_{\mathrm{ID}_i},t')})=$$

$$m\hat{e}(c_1,\mathrm{SK}_{\mathrm{ID}_i}^{H_2(\mathrm{SK}_{\mathrm{ID}_i},t')})$$

3) 云存储服务器在收到具有新的密文类型的密文$c'=(c_1',c_2',c_3')$。其数据的代理重加密过程和解密过程与正常情况一致。由数据拥有者执行算法5)[4]的RKey()算法为：

$$\mathrm{RK}_{\mathrm{ID}_i\to\mathrm{ID}_j}=$$

$$(t',\mathrm{SK}_{\mathrm{ID}_i}^{-H_2(\mathrm{SK}_{\mathrm{ID}_i},t')}H_1(X),g^{r'},X\hat{e}(H_1(\mathrm{ID}_j),\mathrm{Pub})^{r'})$$

攻击者截获后，仍将t'修改成t''后传于云存储服务器为：

$$\mathrm{RK}^*_{\mathrm{ID}_i\to\mathrm{ID}_j}=$$

$$(t'',\mathrm{SK}_{\mathrm{ID}_i}^{-H_2(\mathrm{SK}_{\mathrm{ID}_i},t')}H_1(X),g^{r'},X\hat{e}(H_1(\mathrm{ID}_j),\mathrm{Pub})^{r'})$$

4) 云存储服务器执行算法6)[4]中的$\mathrm{REnc}(c,\mathrm{RK}_{\mathrm{ID}_i\to\mathrm{ID}_j})$算法，得到$c'=(c_1',c_2',c_3',c_4')$；$c_2'=c_2\times\hat{e}(c_1,\mathrm{SK}_{\mathrm{ID}_i}^{-H_2(\mathrm{SK}_{\mathrm{ID}_i},t')}H_1(X))=m\hat{e}(g^r,H_1(X))$；$c_1'=c_1$；$c_3'=g^{r'}$；$c_4'=X\hat{e}(H_1(\mathrm{ID}_j),\mathrm{Pub})^{r'}$。

5) R执行算法7)[4]的$\mathrm{RDec}(c',\mathrm{SK}_{\mathrm{ID}_j})$算法，解密得到明文$m: X=\dfrac{c_4'}{\hat{e}(c_3',\mathrm{SK}_{\mathrm{ID}_j})}$；$m=\dfrac{c_2'}{\hat{e}(c_1',H_1(X))}$。

根据上面的推导，可以清晰地看到由于重加密过程与解密过程不涉及密文类型标识t'，因此可任意修改。攻击者只需要针对密文类型修改密钥和代理重加密密钥的传输过程做出相应的篡改，就可以将原本的可共享类型修改为不可共享类型，将不可共享类型修改为可共享类型。

2.2　由于类型修改引起了新的条件性选择明文攻击问题分析

在文献[4]中出现的PRE方案“即使storage不可信，storage也无法知道数据的内容”。但是该方案在提供数据类型可修改功能的过程中，实际上泄露了部分的数据信息内容，造成了新的选择明文攻击问题。

攻击者与服务器合谋，攻击者在storage处获取到在密文类型为t_1情况下的密文：$c_{m_1}=(c_{1,m_1},c_{2,m_1},c_{3,m_1})$，其攻击目标为恢复相应的明文$m_1$。攻击者与云服务器合谋攻击步骤如下。

首先攻击者采用选择明文攻击，获取到密文类型为t_2情况下的明密文对为：

$$c_{m_2}=(c_{1,m_2},c_{2,m_2},c_{3,m_2})$$

式中，$c_{2,m_2}=m_2\hat{e}(H_1(\mathrm{ID}_i),\mathrm{Pub})^{r_2H_2(\mathrm{SK}_{\mathrm{ID}_i},t_2)}$。

根据前面得到的密文 $c_{m_1}=(c_{1,m_1},c_{2,m_1},c_{3,m_1})$，$c_{2,m_1}=m_1\hat{e}(H_1(\mathrm{ID}_i),\mathrm{Pub})^{r_1H_2(\mathrm{SK}_{\mathrm{ID}_i},t_1)}$，则有：

$$c_{2,m_1}c_{2,m_2}=$$
$$m_1m_2\hat{e}(H_1(\mathrm{ID}_i),\mathrm{Pub})^{r_1H_2(\mathrm{SK}_{\mathrm{ID}_i},t_1)+r_2H_2(\mathrm{SK}_{\mathrm{ID}_i},t_2)}$$

从上面可以看出，如果密文类型不可修改，该方案是可以抵御选择明文攻击的。但是由于数据类型明文存放、传输并可修改，带来了新的问题。如果云服务器出于好奇，将历史的代理重加密密钥与密文类型转换密钥记录，其中密文类型转换记录如表1所示。

表1 文类型转换记录表

	t_1	t_2	$\cdots$	t_n
t_1	$t_1\to t_1$			$t_1\to t_n$
t_2		$t_2\to t_2$		$t_2\to t_n$
$\vdots$			$\ddots$	
t_n				$t_n\to t_n$

在表中除自身的转换外，每一项都对应了一次密文类型转换。这些项代表storage记录了相应的密文类型转换所需的密文类型转换密钥。分析算法8)[4]密文类型转换密钥 $\mathrm{TK}=(t',\mathrm{SK}_{\mathrm{ID}_i}^{-H_2(\mathrm{SK}_{\mathrm{ID}_i},t)}\mathrm{SK}_{\mathrm{ID}_i}^{H_2(\mathrm{SK}_{\mathrm{ID}_i},t')})$ 得知，其数据结构只与 $(\mathrm{SK}_{\mathrm{ID}_i},t',t)$ 相关。这说明storage对密文类型转换的操作是可以多次叠加的。将 TK = () 中的密文类型转换密钥 $\mathrm{SK}_{\mathrm{ID}_i}^{-H_2(\mathrm{SK}_{\mathrm{ID}_i},t)}\times\mathrm{SK}_{\mathrm{ID}_i}^{H_2(\mathrm{SK}_{\mathrm{ID}_i},t')}$ 记做 $\mathrm{RTK}_{t\to t'}$。如果密文类型转换记录表中存在一个 t_1 与 t_2 共同指向的密文类型 t^*，将由 t_1 转换到 t^* 中间所经历的密文类型记录为一个非空集合 $\{t_{\theta_1},t_{\theta_2},\cdots,t_{\theta_l}\}$，$1\leqslant l\leqslant n$。storage可以通过查询记录表获取所需的密文类型修改密钥组：$\{\mathrm{RTK}_{t_1\to t_{\theta_1}},\mathrm{RTK}_{t_{\theta_1}\to t_{\theta_2}},\cdots,\mathrm{RTK}_{t_{\theta_l}\to t_{t^*}}\}$。对密文类型修改密钥进行乘积处理，有：

$$\mathrm{RTK}_{t_1\to t_{\theta_1}}\mathrm{RTK}_{t_{\theta_1}\to t_{\theta_2}}\cdots\mathrm{RTK}_{t_{\theta_l}\to t_{t^*}}=$$
$$\mathrm{SK}_{\mathrm{ID}_i}^{-H_2(\mathrm{SK}_{\mathrm{ID}_i},t_1)}\mathrm{SK}_{\mathrm{ID}_i}^{H_2(\mathrm{SK}_{\mathrm{ID}_i},t_{\theta_1})}\mathrm{SK}_{\mathrm{ID}_i}^{-H_2(\mathrm{SK}_{\mathrm{ID}_i},t_{\theta_1})}\times$$
$$\mathrm{SK}_{\mathrm{ID}_i}^{H_2(\mathrm{SK}_{\mathrm{ID}_i},t_{\theta_2})}\cdots\mathrm{SK}_{\mathrm{ID}_i}^{-H_2(\mathrm{SK}_{\mathrm{ID}_i},t_{\theta_l})}\mathrm{SK}_{\mathrm{ID}_i}^{H_2(\mathrm{SK}_{\mathrm{ID}_i},t^*)}=$$
$$\mathrm{SK}_{\mathrm{ID}_i}^{-H_2(\mathrm{SK}_{\mathrm{ID}_i},t_1)}\mathrm{SK}_{\mathrm{ID}_i}^{H_2(\mathrm{SK}_{\mathrm{ID}_i},t^*)}=\mathrm{RTK}_{t_1\to t^*}$$

storage最终可以通过计算得到由类型 t_1 到共同类型 t^* 的密文类型转换密钥 $\mathrm{RTK}_{t_1\to t^*}$。再经过算法9)[4]进行密文类型转换为：

$$c'^*_{2,m_1}=c_{2,m_1}\hat{e}(c_{1,m_1},\mathrm{SK}_{\mathrm{ID}_i}^{-H_2(\mathrm{SK}_{\mathrm{ID}_i},t_1)}\mathrm{SK}_{\mathrm{ID}_i}^{H_2(\mathrm{SK}_{\mathrm{ID}_i},t^*)})=$$
$$m_1\hat{e}(H_1(\mathrm{ID}_i),\mathrm{Pub})^{r_1H_2(\mathrm{SK}_{\mathrm{ID}_i},t_1)}\times$$
$$\hat{e}(c_{1,m_1},\mathrm{SK}_{\mathrm{ID}_i}^{-H_2(\mathrm{SK}_{\mathrm{ID}_i},t_1)}\mathrm{SK}_{\mathrm{ID}_i}^{H_2(\mathrm{SK}_{\mathrm{ID}_i},t^*)})=$$
$$m_1\hat{e}(H_1(\mathrm{ID}_i),\mathrm{Pub})^{r_1H_2(\mathrm{SK}_{\mathrm{ID}_i},t^*)}$$

得到新类型 t^* 下的预共享密文 c'^*_{2,m_1}，同理可得由 t_2 转换到 t^* 情况下的预共享密文 c'^*_{2,m_2} 为：

$$c'^*_{2,m_2}=m_2\hat{e}(H_1(\mathrm{ID}_i),\mathrm{Pub})^{r_2H_2(\mathrm{SK}_{\mathrm{ID}_i},t^*)}$$

接下来在代理重加密过程中，服务器根据记录的代理重加密密钥为：

$$\mathrm{RK}^*_{\mathrm{ID}_i\to\mathrm{ID}_j}=$$
$$(t^*,\mathrm{SK}_{\mathrm{ID}_i}^{-H_2(\mathrm{SK}_{\mathrm{ID}_i},t^*)}H_1(X^*),$$
$$g^{r^*},X^*\hat{e}(H_1(\mathrm{ID}_j),\mathrm{Pub})^{r^*})$$

修改有：

$$c'^*_1=c'_{1,m_1}c'_{1,m_2}=g^{r_1}g^{r_2}=g^{r_1+r_2}$$
$$c'^*_3=c'_3=g^{r^*}$$

令 $c^*_2=c'^*_{2,m_1}c'^*_{2,m_2}$，攻击者最终将得到篡改过的共享密文为：

$$c'^*=(c'^*_1,c'^*_2,c'^*_3,c'^*_4)$$

其中，有：

$$c'^*_2=c^*_2\hat{e}(c'^*_1,\mathrm{SK}_{\mathrm{ID}_i}^{-H_2(\mathrm{SK}_{\mathrm{ID}_i},t^*)}H_1(X^*))=$$
$$m_1m_2\hat{e}(H_1(\mathrm{ID}_i),\mathrm{Pub})^{(r_1+r_2)H_2(\mathrm{SK}_{\mathrm{ID}_i},t^*)}\times$$
$$\hat{e}(g^{r_1+r_2},\mathrm{SK}_{\mathrm{ID}_i}^{-H_2(\mathrm{SK}_{\mathrm{ID}_i},t^*)}H_1(X^*))=$$
$$m_1m_2\hat{e}(g^{(r_1+r_2)},H_1(X^*))$$
$$c'^*_4=X^*\hat{e}(H_1(\mathrm{ID}_j),\mathrm{Pub})^{r^*}$$

攻击者最终通过合法解密过程7)[4]得到：

$$X^*=\frac{c'^*_4}{\hat{e}(c'^*_3,\mathrm{SK}_{\mathrm{ID}_j})}$$

$$m=\frac{c'^*_2}{\hat{e}(c'^*_1,H_1(X^*))}=\frac{c'^*_2}{\hat{e}(g^{(r_1+r_2)},H_1(X^*))}=m_1m_2$$

由 m 可得 $m_1=m/m_2$

攻击者攻击成功，由上可见在一定条件下，即攻击者与云服务器合谋时，密文类型修改会引起新的选择明文攻击问题。

3 对类型可修改的基于身份代理重加密方案的改进

针对上述两个安全漏洞，本文对文献[4]的方案做出了以下改进：1) 增加一个对类型 t 的完整性验证环节；2) 对数据块进行编号排序。

3.1 方案改进

在密文存储阶段，本文引入对数据块进行区分的编号信息 k ($m_1,\cdots,m_k,\cdots,m_n \in G$，$1 \leqslant k \leqslant n$)，并将算法3)[4]修改为 $\mathrm{Enc}(\mathrm{PK},\mathrm{ID},\mathrm{SK}_{\mathrm{ID}},t,m_k,k)$，其中，$c_2$ 由 $c_2 = m\hat{e}(H_1(\mathrm{ID}),\mathrm{Pub})^{rH_2(\mathrm{SK}_{\mathrm{ID}},t)}$ 改为 $c_2 = m_k \times \hat{e}(H_1(\mathrm{ID}),\mathrm{Pub})^{rH_2(\mathrm{SK}_{\mathrm{ID}},t,k)}$。

相应将算法4)[4]修改为：

$$m_k = \frac{c_2}{\hat{e}(\mathrm{SK}_{\mathrm{ID}},c_1)^{H_2(\mathrm{SK}_{\mathrm{ID}},c_3,k)}}$$

将算法5)[4]修改为 $\mathrm{RKey}(\mathrm{ID}_i,\mathrm{ID}_j,t,k,\mathrm{SK}_{\mathrm{ID}_i})$，其中有：

$$\mathrm{RK}_{\mathrm{ID}_i\to\mathrm{ID}_j} =$$

$$(t,\mathrm{SK}_{\mathrm{ID}_i}^{-H_2(\mathrm{SK}_{\mathrm{ID}_i},t,k)}H_1(X)^t, g^{r'}, X\hat{e}(H_1(\mathrm{ID}_j),\mathrm{Pub})^{r'})$$

算法6)[4]中 storage 将传输 $c' = (c_1',c_2',c_3',c_4',c_5')$，其中有：

$$c_2' = c_2\hat{e}(c_1,\mathrm{SK}_{\mathrm{ID}_i}^{-H_2(\mathrm{SK}_{\mathrm{ID}_i},t,k)}H_1(X)^t) =$$

$$m_k\hat{e}(g^r,H_1(X))^t$$

$$c_5' = t$$

算法7)[4]为：

$$X = \frac{c_4'}{\hat{e}(c_3',\mathrm{SK}_{\mathrm{ID}_j})} \qquad m_k = \frac{c_2'}{\hat{e}(c_1',H_1(X))^{c_5'}}$$

算法8)[4]为：

$$\mathrm{TK} = (t',\mathrm{SK}_{\mathrm{ID}_i}^{-H_2(\mathrm{SK}_{\mathrm{ID}_i},t,k)}\mathrm{SK}_{\mathrm{ID}_i}^{H_2(\mathrm{SK}_{\mathrm{ID}_i},t',k)})$$

算法9)[4]为：

$$c_2' = c_2\hat{e}(c_1,\mathrm{SK}_{\mathrm{ID}_i}^{-H_2(\mathrm{SK}_{\mathrm{ID}_i},t,k)}\mathrm{SK}_{\mathrm{ID}_i}^{H_2(\mathrm{SK}_{\mathrm{ID}_i},t',k)}) =$$

$$m_k\hat{e}(c_1,\mathrm{SK}_{\mathrm{ID}_i}^{H_2(\mathrm{SK}_{\mathrm{ID}_i},t',k)})$$

3.2 正确性验证

通过上述改进，可以看到R可正常解密共享密文，最终取得 m_k。

1) storage存储：

$$c_2 = m_k\hat{e}(H_1(\mathrm{ID}),\mathrm{Pub})^{rH_2(\mathrm{SK}_{\mathrm{ID}},t,k)}$$

2) DO解密正确性：

$$m_k = \frac{c_2}{\hat{e}(\mathrm{SK}_{\mathrm{ID}},c_1)^{H_2(\mathrm{SK}_{\mathrm{ID}},c_3,k)}} =$$

$$m_k\frac{\hat{e}(H_1(\mathrm{ID}),\mathrm{Pub})^{rH_2(\mathrm{SK}_{\mathrm{ID}},t,k)}}{\hat{e}(\mathrm{SK}_{\mathrm{ID}},g^r)^{H_2(\mathrm{SK}_{\mathrm{ID}},t,k)}} = m_k$$

3) storage存储预共享密文通过代理重加密后转换成最终共享密文：

$$c_2 \to c_2'$$

$$c_2' = c_2\hat{e}(c_1,\mathrm{SK}_{\mathrm{ID}_i}^{-H_2(\mathrm{SK}_{\mathrm{ID}_i},t,k)}H_1(X)^t) =$$

$$m_k\hat{e}(H_1(\mathrm{ID}),\mathrm{Pub})^{rH_2(\mathrm{SK}_{\mathrm{ID}},t,k)} \times$$

$$\hat{e}(g^r,\mathrm{SK}_{\mathrm{ID}_i}^{-H_2(\mathrm{SK}_{\mathrm{ID}_i},t,k)})\hat{e}(g^r,H_1(X)^t) =$$

$$m_k\hat{e}(g^r,H_1(X))^t$$

4) R进行最后的数据解密：

$$X = \frac{c_4'}{\hat{e}(c_3',\mathrm{SK}_{\mathrm{ID}_j})} = \frac{X\hat{e}(H_1(\mathrm{ID}_j),\mathrm{Pub})^{r'}}{\hat{e}(g^{r'},\mathrm{SK}_{\mathrm{ID}_j})} = X$$

$$m_k = m_k\frac{\hat{e}(g^r,H_1(X))^t}{\hat{e}(c_1',H_1(X))^{c_5'}} = m_k\frac{\hat{e}(g^r,H_1(X))^t}{\hat{e}(g^r,H_1(X))^{c_5'}} = m_k$$

3.3 安全性分析

文献[4]方案安全漏洞的产生是由于只添加了密文类型修改功能，而没有在其后的环节做出相应的验证，使方案整体的完整性遭到了破坏。针对密文类型修改所引起的两个安全漏洞，本文在其方案相应的位置做出了修改，下面通过对比来进行说明此修改是否会引起问题。

首先在密文存储阶段引入了对数据块进行区分的编号信息 k，并将 c_2 由 $c_2 = m\hat{e}(H_1(\mathrm{ID}),\mathrm{Pub})^{rH_2(\mathrm{SK}_{\mathrm{ID}},t)}$ 改为 $c_2 = m_k\hat{e}(H_1(\mathrm{ID}),\mathrm{Pub})^{rH_2(\mathrm{SK}_{\mathrm{ID}},t,k)}$。这样就对数据的密文与数据的编号信息进行了位置绑定，阻止了可能的指数项合并。在遭遇攻击时，云服务器无法将不同编号数据块所对应的预共享密文顺利的改变为相同的类型结构，从而抵御了选择明文攻击。

其次为解决密文类型修改的验证问题，本文在代理重加密阶段将 RKey() 中的 $\mathrm{RK}_{\mathrm{ID}_i\to\mathrm{ID}_j}$ 由 $(t,\mathrm{SK}_{\mathrm{ID}_i}^{-H_2(\mathrm{SK}_{\mathrm{ID}_i},t)}H_1(X), g^{r'}, X\hat{e}(H_1(\mathrm{ID}_j),\mathrm{Pub})^{r'})$ 变为 $(t,\mathrm{SK}_{\mathrm{ID}_i}^{-H_2(\mathrm{SK}_{\mathrm{ID}_i},t,k)}H_1(X)^t, g^{r'}, X\hat{e}(H_1(\mathrm{ID}_j),\ \mathrm{Pub})^{r'})$。

显然由于 RKey() 是由DO进行运算的，在 $H_1(X)$ 的指数部分增加 t，并不会损害方案的安全性，但是却可以在算法7)[4]的解密阶段与进行验证，如果 c_5' 被修改，则R无法正确解密获取数据信息。

根据文献[2]的安全性证明与文献[4]的安全性分析。本文的方案对比文献[4]方案只增加了密文与数据块明文编号信息的绑定，以及最后一阶段对明文传输密文类型信息 t 的验证，因此本文的方案具有与文献[2]一致的安全性。

4 结 束 语

本文针对文献[4]方案中两个安全漏洞进行了分析与改进，并在其基础上提出了对类型可修改的基于身份代理重加密方案的改进方案。在经过对该方案的正确性验证与安全性分析后表明，该方案与传统的基于类型和身份的代理重加密方案相比，具有相同的安全性。

参 考 文 献

[1] BLAZE M, BLEUMER G, STRAUSS M. Divertible protocols and atomic proxy cryptography[C]//Advances in Cryptology-EUROCRYPT'98, LNCS 1403. Heidelberg: Springer-Verlag, 1998.

[2] IBRAIMI L, TANG Q, HARTEL P, et al. A type-and-identity-based proxy re-encryption scheme and its application in healthcare[C]//Secure Data Management 2008, LNCS 5159. Heidelberg: Springer, 2008.

[3] IBRAIMI L, TANG Q, HARTEL P, et al. Exploring type-and-identity-based proxy re-encryption scheme to securely manage personal health records[J]. Innovations in Data Methodologies and Computational Algorithms for Medical Applications, 2012, 1(2):1-21.

[4] 刘志远, 崔国华. 类型可修改的基于身份代理重加密方案[J]. 电子科技大学学报, 2014, 43(3): 408-412.
LIU Zhi-yuan, CUI Guo-hua. A dynamic type and identity-based proxy re-encryption scheme[J]. Journal of University of Electronic Scienceand Technology of China 2014, 43(3): 408-412.

[5] GREEN M, ATENIESE G. Identity-based proxy re-encryption[C]//Applied Cryptography and Network Security (ACNS), LNCS 4521. Heidelberg: Springer-Verlag, 2007.

[6] CHU C K, TZENG W G. Identity-based proxy re-encryption without random oracles[C]//ISC 2007, LNCS 4779. Heidelberg: Springer, 2007.

[7] MATSUO T. Proxy re-encryption systems for identity-based encryption[C]//Pairing-Based Cryptography-Pairing 2007. Tokyo, Japan: Computer Science, 2007.

[8] LIBERT B, DAMIEN V. Unidirectional chosen-ciphertext secure proxy re-encryption[C]//Public Key Cryptography PKC 2008, LNCS 4939. Heidelberg: Springer-Verlag, 2008.

编 辑 黄 莘