类型可修改的基于身份代理重加密方案

刘志远[1,2]，崔国华[1]

(1. 华中科技大学计算机科学与技术学院 武汉 430074; 2. 湖北理工学院计算机学院 湖北 黄石 435003)

【摘要】云存储中，代理重加密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。该文提出了一种类型可修改的基于身份代理重加密方案，该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信息类型。在实际应用中，该方案比基于类型和身份的代理重加密方案具有更加广泛的应用场景。

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.1001-0548.2014.03.016

A Dynamic Type and Identity-Based Proxy Re-Encryption Scheme

LIU Zhi-yuan[1,2] and CUI Guo-hua[1]

(1. College of Computer Science and Technology, Huazhong University of Science and Technology Wuhan 430074;
2. College of Computer Science, HuBei Institute of Technology Huangshi Hubei 435003)

Abstract Cloud storage is drawing tremendous attention in IT field. In it, an open challenge is to keep the privacy of users' outsourced data and securely share users' data. To address this challenge, proxy re-encryption (PRE) scheme is one of promising method. A dynamic type and identity-based PRE scheme will be proposed. This scheme not only obtains the traditional functions of PRE, but also allows a user to update the original types of his outsourced ciphertexts. Compared with the previous PRE scheme, the new feature makes the proposed PRE scheme more useful in practical applications.

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

目前，由于云存储的兴起，用户数据的安全分布式存储研究受到了广泛的关注。为了解决该研究中面临的核心挑战，即用户数据的安全分布式存储和共享，代理重加密(proxy re-encryption, PRE)方案成为倍受关注的方法之一。

基于代理重加密方案，实现用户数据安全分布式存储和共享的基本工作流程包括4个角色和3个功能。4个角色分别是数据拥有者(data owner, DO)、数据共享者(receiver, R)、密钥生成中心(key generator center, KGC)和第三方存储器(storage)。3个功能分别是密码系统参数生成、数据加密存储和密文数据共享。在密码系统参数生成阶段，KGC生成PRE的系统参数，并将可公开部分发送给所有的用户和storage。在数据加密存储阶段，DO采用自身的公钥信息将其拥有的数据加密并存储在storage。在密文数据共享阶段，DO根据数据共享者R的公钥信息生成重加密密钥并发送给storage。根据该密钥，storage对预共享的密文进行重加密，并将生成的密文发送给R，从而实现密文的共享。在整个数据的存储和共享过程中，PRE可以保证即使storage不可信，也不可能知道数据的任意内容。

代理重加密最早由文献[1]提出，随后得到了广泛的研究，涉及可多次重加密的PRE方案[2-3]、基于身份的PRE方案[4-5]、单向PRE方案[6-8]和可追踪代理重加密方案[9]等。目前多数PRE方案仅能实现全密文共享，即当storage收到重加密密钥后可以把DO拥有的所有密文都共享给共享者。因此，为了实现PRE中细粒度的密文共享，文献[10]提出了基于类型和身份的PRE方案。在该方案中每个密文被赋予了信息类型，因此DO可以选择共享指定的信息类型给共享者。但是在该方案中，密文的信息类型是静态的，因此无法实现密文信息类型的动态修改。另一方面，

收稿日期：2013－06－21； 修回日期：2014－03－12
基金项目：国家自然科学基金(60903196)；湖北省自然科学基金(2013CFB039)
作者简介：刘志远(1972－)，男，博士，副教授，主要从事密码学与网络安全的研究.

密文信息类型的可动态修改性可以实现新的应用。例如，令密文信息类型为“可共享”和“不可共享”。DO为了节省自身本地存储空间的开销，或者为了实现自身数据在不同终端的灵活使用，它也会将“不可共享”类型的数据以加密的方式安全地存储在storage。但是在实际应用中，DO可能也希望“不可共享”类型的密文可以改变其信息类型为“可共享”，因此实现密文信息类型的动态性非常适应此类应用。

1 基于类型和身份的代理重加密

1.1 双线性映射

设 G 和 G_T 是两个大素数 p 阶群，g 是 G 的生成元。$\hat{e}:G\times G\to G_T$ 是一个有效可计算的双线性映射，并且满足如下性质：

1) 双线性性：对任意的 $(a,b)\in Z_p^*$，有 $\hat{e}(g^a,g^b)=\hat{e}(g,g)^{ab}$；

2) 非退化性：$\hat{e}(g,g)\neq 1$。

双线性映射最早由文献[11]提出，并用于求解椭圆曲线群的离散对数问题。后因其特有的双线性性，被广泛应用于各类密码方案的设计中。有关双线性映射更多的细节可查阅文献[12]。

1.2 基于类型和身份的代理重加密方案

文献[10]提出的基于类型和身份的代理重加密方案描述如下：

1) Setup(1^k) 算法：该算法由KGC执行并生成系统用的主公开参数和主秘密参数。具体过程如下：

①生成两个大素数 p 阶群 G 和 G_T，G 的生成元 g 和双线性映射 $\hat{e}:G\times G\to G_T$；

②选择主秘密参数 $\alpha\in Z_p^*$ 和Hash函数 $\mathrm{H}_1:\{0,1\}^*\to G$、$\mathrm{H}_2:\{0,1\}^*\to Z_p^*$；

③生成主公开参数 $\mathrm{PK}=(G,G_T,g,\mathrm{Pub}=g^{\alpha},\hat{e},\mathrm{H}_1,\mathrm{H}_2)$ 和主秘密参数 $\mathrm{MK}=\alpha$。

2) Extract(MK, ID) 算法：该算法由KGC执行并为身份信息ID生成对应的私钥，身份信息ID对应的私钥为 $\mathrm{SK}_{\mathrm{ID}}=\mathrm{H}_1(\mathrm{ID})^{\alpha}$。

3) Enc($\mathrm{PK},\mathrm{ID},\mathrm{SK}_{\mathrm{ID}},t,m$) 算法：该算法由明文 m 的拥有者采用自己的身份信息ID生成密文 $c=(c_1,c_2,c_3)$。其中，c_1、c_2 和 c_3 的计算分别如下：

① 随机选取 $r\in Z_p^*$，$c_1=g^r$；

② 计算 $c_2=m\hat{e}(\mathrm{H}_1(\mathrm{ID}),\mathrm{Pub})^{r\mathrm{H}_2(\mathrm{SK}_{\mathrm{ID}},t)}$；

③ $c_3=t$。

值得注意的是，在 c_2 的生成过程中，明文 m 的信息类型 t 和用户的私钥 $\mathrm{SK}_{\mathrm{ID}}$ 作为Hash函数 H_2 的输入，并且将输出值作为指数部分参与双线性映射 $\hat{e}$ 的计算。根据Hash函数 H_2 的散列性，攻击者不能合法地修改函数 $\mathrm{H}_2(\mathrm{SK}_{\mathrm{ID}},t)$ 中的 t 值。这样的处理方式也使得后面重加密的过程必须采用与目标密文具有相同信息类型的重加密密钥才能转换为一个正确的重加密密文。虽然 $c_3=t$ 使得攻击者可以改变 c_3 的值，但是这种能力并不会使攻击者具有改变 c_2 中的 t 值的能力。这里将 t 值作为密文的一部分并且直接给出，主要目的是：当第三方需要对具有指定信息类型的密文做重加密时，能够快速地找到具有该类型值的密文，然后进行重加密过程。当然，如果想进一步限制对 c_3 的修改，可以直接采用签名技术实现，使得攻击者对 c_3 的修改可以很容易检测出来。

4) Dec($c,\mathrm{SK}_{\mathrm{ID}}$) 算法：该算法由私钥的拥有者对密文 c 进行解密，即 $m=\dfrac{c_2}{\hat{e}(\mathrm{SK}_{\mathrm{ID}},c_1)^{\mathrm{H}_2(\mathrm{SK}_{\mathrm{ID}},c_3)}}$。

5) RKey($\mathrm{ID}_i,\mathrm{ID}_j,t,\mathrm{SK}_{\mathrm{ID}_i}$) 算法：该算法由数据拥有者 ID_i 执行，并为预共享者 ID_j 和预共享数据的类型 t 生成代理重加密密钥 $\mathrm{RK}_{\mathrm{ID}_i\to\mathrm{ID}_j}$。具体的执行过程如下：

①随机选择 $X\in G$ 和 $r'\in Z_p^*$；

②计算：

$$\mathrm{RK}_{\mathrm{ID}_i\to\mathrm{ID}_j}=(t,\mathrm{SK}_{\mathrm{ID}_i}^{-\mathrm{H}_2(\mathrm{SK}_{\mathrm{ID}_i},t)}\cdot\mathrm{H}_1(X),$$
$$g^{r'},X\cdot\hat{e}(\mathrm{H}_1(\mathrm{ID}_j),\mathrm{Pub})^{r'}\cdot\hat{e}(\mathrm{H}_1(\mathrm{ID}_j),\mathrm{Pub})^{r'}) \quad (1)$$

值得注意的是，在式(1)中，$\mathrm{SK}_{\mathrm{ID}_i}^{-\mathrm{H}_2(\mathrm{SK}_{\mathrm{ID}_i},t)}$ 的计算分两步完成，即先计算 $\mathrm{H}_2(\mathrm{SK}_{\mathrm{ID}_i},t)$，然后再计算得到 $\mathrm{SK}_{\mathrm{ID}_i}^{-\mathrm{H}_2(\mathrm{SK}_{\mathrm{ID}_i},t)}$ 的值。因此根据Hash函数 H_2 的散列性，攻击者即使得到了 $\mathrm{RK}_{\mathrm{ID}_i\to\mathrm{ID}_j}$ 的值，也不可能改变式(1)中 $\mathrm{SK}_{\mathrm{ID}_i}^{-\mathrm{H}_2(\mathrm{SK}_{\mathrm{ID}_i},t)}$ 指数部分的 t 值，并使得 $\mathrm{RK}_{\mathrm{ID}_i\to\mathrm{ID}_j}$ 变为其他信息类型的重加密密钥。信息类型 t 作为一部分直接在 $\mathrm{RK}_{\mathrm{ID}_i\to\mathrm{ID}_j}$ 中给出的目的是为了方便第三方快速地找到具有指定信息类型的密文，并进行重加密过程。当然可以直接采用签名的方法来防止攻击者对 t 的修改。在后面的重加密部分，可以看到具有只有当重加密密钥与目标密文相同的信息类型的时候，才能生成正确的重加密密文。

6) REnc($c,\mathrm{RK}_{\mathrm{ID}_i\to\mathrm{ID}_j}$) 算法：该算法由预共享密文的存储第三方storage执行，用于实现预共享密文 c 的所有权从其拥有者 ID_i 到共享者 ID_j 的转换，并生成可由共享者 ID_j 解密的密文 $c'=(c_1',c_2',c_3',c_4')$。具体的执行过程如下：

① $c_1'=c_1$；

② 计算 $c_2'=c_2\cdot\hat{e}(c_1,\mathrm{SK}_{\mathrm{ID}_i}^{-\mathrm{H}_2(\mathrm{SK}_{\mathrm{ID}_i},t)}\cdot\mathrm{H}_1(X))$

$=m\cdot\hat{e}(g^{r},\mathrm{H}_1(X))$；

③计算 $c_3'=g^{r'}$；

④计算 $c_4'=X\cdot\hat{e}(\mathrm{H}_1(\mathrm{ID}_j),\mathrm{Pub})^{r'}$。

值得注意的是，c_2' 的计算需要目标密文中的 c_2 和重加密密钥的 $\mathrm{SK}_{\mathrm{ID}_i}^{-\mathrm{H}_2(\mathrm{SK}_{\mathrm{ID}_i},t)}\cdot\mathrm{H}_1(X)$ 部分，且 $c_2=m\cdot\hat{e}(\mathrm{H}_1(\mathrm{ID}),\mathrm{Pub})^{r\cdot\mathrm{H}_2(\mathrm{SK}_{ID},t)}$ 已知。可以看出只有当这两部分中的信息类型 t 具有相同的取值时，才能够计算生成 $m\cdot\hat{e}(g^{r},\mathrm{H}_1(X))$。反之，即如果信息类型 t 具有不相同的取值，那么根据Hash函数 H_2 的散列性是不可能得到 $c_2'=m\cdot\hat{e}(g^{r},\mathrm{H}_1(X))$ 的。进一步地，只有当 c_2' 具有这种取值时，重加密密文的解密部分才能正确地解密出明文。简单地说，只有当目标密文与重加密密钥具有相同的信息类型时才能重加密为正确的密文。

7) $\mathrm{RDec}(c',\mathrm{SK}_{\mathrm{ID}_j})$ 算法：该算法由共享者 ID_j 执行，并解密出共享的明文 m。具体的执行过程如下：

①计算 $X=\dfrac{c_4'}{\hat{e}(c_3',\mathrm{SK}_{\mathrm{ID}_j})}$；

②计算 $m=\dfrac{c_2'}{\hat{e}(c_1',\mathrm{H}_1(X))}$。

从上述算法可以看出，文献[10]提出的方案并没有实现密文类型的动态修改，即当密文 c 的拥有者想修改该密文的类型时，该方案并没有给出具体的解决方法。

2　类型可修改的基于身份的代理重加密方案

本文类型可修改的基于身份的代理重加密方案主要由9个算法组成，前面7个算法和文献[10]提出的基于类型和身份的代理重加密方案相同，在此不再描述。不同的两个算法描述如下：

8) $\mathrm{TKey}(t,t',\mathrm{SK}_{\mathrm{ID}_i})$ 算法：该算法由密文的拥有者执行，并生成可用于storage修改密文类型的密钥。具体的执行过程如下：

$$\mathrm{TK}=(t',\mathrm{SK}_{\mathrm{ID}_i}^{-\mathrm{H}_2(\mathrm{SK}_{\mathrm{ID}_i},t)}\cdot\mathrm{SK}_{\mathrm{ID}_i}^{\mathrm{H}_2(\mathrm{SK}_{\mathrm{ID}_i},t')}) \tag{2}$$

根据上述算法可看出，攻击者是无法修改式(2)中 $\mathrm{SK}_{\mathrm{ID}_i}^{-\mathrm{H}_2(\mathrm{SK}_{\mathrm{ID}_i},t)}\cdot\mathrm{SK}_{\mathrm{ID}_i}^{\mathrm{H}_2(\mathrm{SK}_{\mathrm{ID}_i},t')}$ 的信息类型值的。

9) $\mathrm{TSet}(c,\mathrm{TK})$ 算法：该算法由密文的storage执行并修改密文 $c=(c_1,c_2,c_3)$ 的类型，并生成具有新类型的密文 $c'=(c_1',c_2',c_3')$。具体的执行过程如下：

① $c_1'=c_1$；

②计算 $c_2'=c_2\cdot\hat{e}(c_1,\mathrm{SK}_{\mathrm{ID}_i}^{-\mathrm{H}_2(\mathrm{SK}_{\mathrm{ID}_i},t)}\cdot\mathrm{SK}_{\mathrm{ID}_i}^{\mathrm{H}_2(\mathrm{SK}_{\mathrm{ID}_i},t')})$

$=m\cdot\hat{e}(c_1,\mathrm{SK}_{\mathrm{ID}_i}^{\mathrm{H}_2(\mathrm{SK}_{\mathrm{ID}_i},t')})$；

③ $c_3'=t'$。

通过以上的算法描述可以看出，本文提出的方案并没有修改原有方案中关于私钥和加解密部分，因此在这两个部分本方案依然具有一致性或正确性。对于修改密文类型后的新密文也可看出新密文依然具有一致性或正确性，其原因是新密文与旧密文除了类型不同外，依然具有完全相同的数据结构，新密文的拥有者依然可对其解密，也可以用于由重加密密钥生成算法 $\mathrm{RKey}(\mathrm{ID}_i,\mathrm{ID}_j,t,\mathrm{SK}_{\mathrm{ID}_i})$ 和代理重加密算法 $\mathrm{REnc}(c,\mathrm{RK}_{\mathrm{ID}_i\to\mathrm{ID}_j})$ 共同完成的密文共享过程。

3　安全性分析

根据文献[10]的结论，基于类型和身份的代理重加密方案实现了适应性选择身份和明文攻击下的密文不可区分性。本文方案除了增加TKey算法和TSet算法外，其他部分文献[10]的方案完全相同，因此在不考虑TKey算法和TSet算法的情况下，本文提出的方案依然具有适应性选择身份和明文攻击下的密文不可区分性的安全特性。下面主要讨论新方案增加的 $\mathrm{TKey}(t,t',\mathrm{SK}_{\mathrm{ID}_i})$ 算法和 $\mathrm{TSet}(c,\mathrm{TK})$ 算法是否会引发其他新的安全性问题。

首先对比重加密密钥和用于修改密文类型的密钥，即对比式(1)中的 $\mathrm{RK}_{\mathrm{ID}_i\to\mathrm{ID}_j}$ 和式(2)中的TK。由式(2)可以看出：如果TK会引发安全性问题，那么必然是由 $\mathrm{SK}_{\mathrm{ID}_i}^{-\mathrm{H}_2(\mathrm{SK}_{\mathrm{ID}_i},t)}\cdot\mathrm{SK}_{\mathrm{ID}_i}^{\mathrm{H}_2(\mathrm{SK}_{\mathrm{ID}_i},t')}$ 造成的，因为信息类型 t' 在文献[10]的方案和本文的方案中均是以明文出现的，而且信息类型并不是隐私信息。那么 $\mathrm{SK}_{\mathrm{ID}_i}^{-\mathrm{H}_2(\mathrm{SK}_{\mathrm{ID}_i},t)}\cdot\mathrm{SK}_{\mathrm{ID}_i}^{\mathrm{H}_2(\mathrm{SK}_{\mathrm{ID}_i},t')}$ 部分是否会引发安全性问题，这一点可以从 $\mathrm{RK}_{\mathrm{ID}_i\to\mathrm{ID}_j}$ 中找到答案。$\mathrm{RK}_{\mathrm{ID}_i\to\mathrm{ID}_j}$ 中包含有 $\mathrm{SK}_{ID_i}^{-\mathrm{H}_2(\mathrm{SK}_{ID_i},t)}\cdot\mathrm{H}_1(X)$ 部分，该部分与TK中的 $\mathrm{SK}_{\mathrm{ID}_i}^{-\mathrm{H}_2(\mathrm{SK}_{\mathrm{ID}_i},t)}\cdot\mathrm{SK}_{\mathrm{ID}_i}^{\mathrm{H}_2(\mathrm{SK}_{\mathrm{ID}_i},t')}$ 部分有部分相同，它们的主要差别在于 $\mathrm{H}_1(X)$ 与 $\mathrm{SK}_{\mathrm{ID}_i}^{\mathrm{H}_2(\mathrm{SK}_{\mathrm{ID}_i},t')}$。由于在随机预言机模型下的安全性证明过程中，Hash函数 H_1 和 H_2 会被模型化为随机预言机，即 $\mathrm{H}_1(X)$ 与 $\mathrm{SK}_{\mathrm{ID}_i}^{\mathrm{H}_2(\mathrm{SK}_{\mathrm{ID}_i},t')}$ 均具有相同的均匀分布，因此若 $\mathrm{RK}_{\mathrm{ID}_i\to\mathrm{ID}_j}$ 可以保证安全性，那么TK也可以保证安全性。由于文献[10]已经证明了 $\mathrm{RK}_{\mathrm{ID}_i\to\mathrm{ID}_j}$ 具有安全性，所以本文方案的TKey算法也具有安全性。

另一方面，在新方案的 $\mathrm{TSet}(c,\mathrm{TK})$ 算法中，$\mathrm{TSet}(c,\mathrm{TK})$ 算法生成的新密文与旧密文具有完全相同的数据结构，它们仅仅是信息类型不同。进一步，由于旧密文的安全性在文献[10]中已经证明，因此由 $\mathrm{TSet}(c,\mathrm{TK})$ 算法生成的新密文依然具有安全性。

4 结束语

本文提出了一种类型可修改的基于身份代理重加密方案，实现了密文信息类型的可动态修改性。同时，方案的安全性分析表明，本文的方案与传统的基于类型和身份的代理重加密方案具有相同的安全性。

参 考 文 献

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

[2] WANG Hong-bing, CAO Zhen-fu, WANG Li-cheng. Multi-use and unidirectional identity-based proxy re-encryption schemes[J]. Information Sciences, 2010, 180(20): 4042-4059.

[3] SHAO Jun, LIU Peng, CAO Zhen-fu, et al. Multi-use unidirectional proxy re-encryption[C]//IEEE International Conference on Communications(ICC). Kyoto; 2011.

[4] GREEN M, ATENIESE G. Identity-based proxy re-encryption[C]//Proceedings of the 5th international conference on Applied Cryptography and Network Security (ACNS'07) 2007, LNCS 4521. Berlin: Springer-Verlag, 2007.

[5] CHU Cheng-Kang, TZENG Wen-Guey. Identity-based proxy re-encryption without random oracles[C]// Proceedings of information security conference (ISC 2007), LNCS 4779. Berlin: Springer-Verlag, 2007.

[6] LIBERT B, VERGNAUD D. Unidirectional chosen-ciphertext secure proxy re-encryption[C]//Public Key Cryptography PKC 2008, LNCS 4939. Berlin: Springer-Verlag, 2008.

[7] SHERMAN S M Chow, WENG Jian, YANG Yan-jiang, et al. Efficient unidirectional proxy re-encryption[C]//Progress in Cryptology Africacrypt 2010, LNCS 6055. Berlin: Springer-Verlag, 2010.

[8] WENG Jian, CHEN Ming-rong, YANG Yan-jiang, et al. CCA-secure unidirectional proxy re-encryption in the adaptive corruption model without random oracles[J]. Science China(Information Sciences),2010, 53(3):593-606.

[9] LIBERT B, VERGNAUD D. Tracing malicious proxies in proxy re-encryption[C]//Pairing-Based Cryptography 2008, LNCS 5209. Berlin: Springer-Verlag, 2008.

[10] IBRAIMI L, TANG Qiang, 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. Berlin: Springer-Verlag, 2008.

[11] Menezes A J, OKAMOTO T, Vanston S A. Reducing elliptic curve logarithms to logarithms in a finite field[C]// IEEE Transactions on Information Theory. IEEE, 1993.

[12] BONEH D, FRANKLIN M. Identity-based encryption from the weil pairing[C]//Advances in Cryptology- Crypto 2001, LNCS 2139. Berlin: Springer-Verlag, 2001.

编 辑 叶 芳