云存储中基于SKP-ABE的访问控制方法

刘 鸿 李谢华 杨 波
(湖南大学信息科学与工程学院 长沙 410082)

摘 要 针对云存储中的密文访问控制,提出一种基于SKP-ABE的高效访问控制方法EACS。该方法在保持密文信息保密的前提下,将部分私钥组件和相关密文组件的重加密过程转移到云端,以大大降低数据属主的计算代价;并结合RMIA进行用户权限撤销,以实现高效、灵活的密文访问控制。

关键词 云存储,SKP-ABE,EACS,RMIA,用户权限撤销
中图法分类号 TP393 **文件标识码** A

SKP-ABE Based Access Control Method in Cloud Storage

LIU Hong LI Xie-hua YANG Bo
(School of Information Sciences and Engineering, Hunan University, Changsha 410082, China)

Abstract For protection of data in cloud storage server, this paper proposed a simple KP-ABE to make the access control more flexible and efficient. An efficient access control scheme was proposed for transferring some of the re-encryption and key module to cloud storage server so that the computational expenses will be reduced dramatically. Moreover, this paper used the strategy of revoke minimum influence attribute, which will only influence limited attributes and make attribute revocation easily.

Keywords Cloud storage, SKP-ABE, EACS, RMIA, User revocation

1 引言

由于云服务中心能够提供强大的运算能力和海量的存储空间,越来越多的企业和个人将商业数据、个人信息等敏感内容放在云端,进而减少对用户终端的计算负荷和存储空间的要求。然而,云存储服务提供商CSP(Cloud Storage Provider)如何为用户提供安全可靠的访问控制成为云服务推广所必需解决的关键问题。

针对云存储中存在的各类安全问题[1],研究人员进行了深入的研究,并取得了一定的成果。Sahai等人[2]提出了最早的基于属性加密(attribute-based encryption, ABE)的算法。Goyal和Pandey等人[3]提出了基于密钥策略的属性加密方案(key-policy attribute-based encryption, KP-ABE)。而后,Bethencourt等人[4]提出了基于密文策略的属性加密方案(ciphertext-policy attribute-based encryption, CP-ABE)。Shucheng等人在文献[5]中把CP-ABE算法和代理重加密结合起来,将部分工作转移到云端运行;在文献[6]中将KP-ABE、代理重加密[10]、懒惰重加密3种技术结合起来进行访问控制,但该方法没有提及KP-ABE随机参数的更改,其安全性很难保证。洪澄等人[7]提出了AB-ACCS:一种云存储密文访问控制方法,它利用CP-ABE算法和公钥密码系统来实现密文访问控制。吕志泉等人[8]在云存储密文访问控制方案中提出了基于CP-ABE的改进算法HCCP-ABE,同时引入了密钥分割技术和代理重加密技术,但是HCCP-ABE仍然没有解决CP-ABE中存在的问题。洪澄等人在基于CP-ABE的基础上又提出了一种密文访问控制方法HCRB[9],其主要是基于SCP-ABE[9]算法,大大提高了算法的效率,但在用户权限撤销时对复杂访问结构的处理仍缺乏有效的手段。

为了实现云存储服务的安全性和高效性,本文提出了一种简化的KP-ABE(SKP-ABE)接入控制方法,其目的是通过将部分私钥组件和相关密文组件转移到云端来减少数据属主的计算量,降低由于数据属主运算能力有限而造成的网络"瓶颈"。此外,为了保障解密密钥和密文的安全,密文和密钥组件都使用加密传输,而由于云服务器端仅掌握部分密钥组件,因此无法恢复解密密钥,所以在服务器不完全可信的基础上,仍能够实现数据的安全性。

2 安全假定

本文以KP-ABE算法为基础,根据实际情况,假定云存储提供商CSP是"Honest but Curious"[9]。即CSP会忠实地执行运行程序和相关访问协议,但是又会根据输入尽可能地获取数据。在本文中,数据的传送是通过对通信信道的加密

到稿日期:2012-10-16 返修日期:2013-03-18 本文受中国博士后科学基金(20110490193),中央高校基本科研业务费项目资助。
刘 鸿(1988—),女,硕士,主要研究方向为云存储访问控制,E-mail:dhbliuhong@163.com;李谢华(1977—),女,博士后,硕士生导师,主要研究方向为网络安全;杨 波(1987—),男,硕士,主要研究方向为网络安全。

实现的，因此能保证传输数据的安全性。

3 EACS 的实现

在原有的 KP-ABE 中，并没有明确地提出用户权限撤销的过程，相关背景知识参见文献[3]。而在文献[6]中，并没有对 *CA* 进行更新，同时也没有提及 KP-ABE 的随机参数更改，安全性很难保证。在文献[9]中用户权限撤销是直接撤销用户所对应的访问子树，但是属性是由多个用户共享的。针对这一系列的问题，本文提出了一种高效的访问控制方法 EACS(Efficient Access Control Scheme)。

3.1 EACS 访问控制原理

在本文中，仍沿用 KP-ABE 和 CP-ABE 算法中直接对明文加密的算法。不同的是，本文使用 SKP-ABE 对数据文件进行加密，生成数据密文 *E*(为了保证属性和数据的一致性，引入版本号 *VN*，且初始值被设为 1，被存储在 DO 和 CSP 中，当密文更新一次，*VN* 加 1)。同时结合代理重加密及 RMIA 对用户进行权限撤销来实现高效的访问控制方案 EACS。在 EACS 中几乎所有的私钥组件和密文组件的重加密工作都将转移到云服务器端进行，同时使用 RMIA 方法进行重加密，这些都将大大降低对数据属主的存储空间和计算性能的要求。

3.2 相关定义

DO 为每一个数据文件分配一个属性集 *I*，且 *I* 中每一个属性都有自己的版本号，便于更新，同时要求包括一个关键属性 *CA*(Critical Attribute)，以防已撤销用户和 CSP 的联合攻击，*CA* 将由 DO 直接进行更新并将其通过安全信道发送给合法用户。DO 还为每一个用户 *u* 分配一个访问结构 *T* 及用户身份标识号 ID_u，而云服务端建立一个用户列表 UL(User List)和一个属性更新列表 AUL(Attribute Update List)。UL 存储授权用户的 *ID* 号及除 *CA* 外用户的所有叶子节点所对应的属性及其私钥组件。

定义 1(属性更新列表，AUL，attribute update list) 用于记录每一个属性对应的所有更新的版本号及使用的 PRE 密钥，同时存放了引用属性 *i* 的用户集 U_i 及引用该属性的次数 N_i。

在本文中，DO 为每一个用户建立的访问结构都是一棵简单的访问结构树 SAST(simple access structure tree)。

定义 2(简单访问结构树，SAST，simple access structure tree) 有以下特征：

1)必须包含一个 *CA*；

2)内部节点只包括 and 和 or 关系；

3)and 节点仅有 2 个分支，而 or 节点不限制分支数；

4)析取范式；

5)为根节点随机选择 $y \in Z_p$，对于关系节点 *i*，为其随机选值 y_i，如果 *i* 节点所对应的关系为 or，则其子孩子的值与父节点 *i* 的值相同；如果 *i* 节点所对应的关系为 and，对于其子孩子，先随机选择一个值 y_i' 赋给左孩子，则其右孩子的值为 $y_i' + y_i$。

整棵树的结构：根节点必须是一个与门 and，且只有两个分支，一个分支是一个叶子节点，且与之相关联的属性为 *CA*；另一个分支是一棵子 SAST 树。图 1 用一个例子描述了 SAST 树的定义。

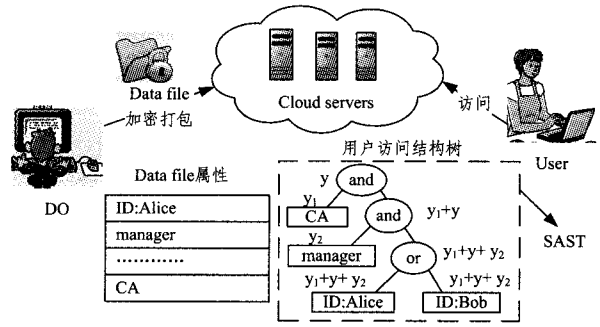

图 1 一个公司访问云端的例子

定义 3(撤销最小影响属性集，RMIA，revoke minimum influence attribute) 决定用户权限撤销时所要撤销的最小属性集。其核心思想是：因为属性是被多个用户共享的，所以对使用频率较高的属性不进行更新，从而降低由于属性撤销对用户所造成的影响。

由于 SAST 树中只有 and 和 or 关系的节点，因此对 RMIA 要分以下情况进行计算：

1)对于 and 关系的节点(无论其是否为根节点)，因为在其下只有 2 个分支，所以只需改变其中一个分支上的某一个属性或某一属性集即可。先查询 AUL，得到每个属性的 *N* 值。对于 and 关系节点，只需比较其两条分支上最终得到的 *N*，选择 *N* 最小的属性集作为更改的对象。

2)因为 or 关系的节点可以拥有多个分支，所以多个连续的 or 关系节点可以合并成一个，则其父节点和子节点的关系不再为“or”，因此 or 关系节点的上、下关系节点只能为“and”。

综上，将 SAST 中的节点关系分为 3 种情况：or 关系节点的父节点为 and，简记为 or 上 and 或 and 下 or；还有 and 上 and；and 上 or。

a)当 or 关系节点为子根节点时，其分支下的直接叶子节点的属性要全部更新。

b)当 or 关系节点不为子根节点时，有两种情况：1)因为 or 上 and，当 or 关系节点下只有直接叶子节点时，先将其分支上的所有 *N* 相加得到 $N_{总}$，再和 and 关系父节点的另一条分支上已算出的较小 *N* 或 $N_{总}$ 相比较，选择 *N* 最小的属性集作为更改的对象。2)当 or 关系节点下存在 and 关系的子节点时，由于 or 上 and，则将 or 关系节点下的直接叶子节点的 *N* 和 and 关系子节点下已计算出的较小的 *N* 相加得到 $N_{总}$，再和 and 关系父节点的另一分支上的 *N* 相比较，选择 *N* 最小的属性集作为更改的对象。可以用图 2 的 SAST 描述整个 RMIA 的计算过程，如图 3 和图 4 所示。

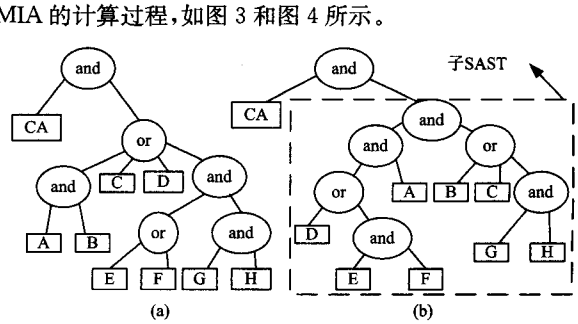

图 2 SAST

```
    / * generate the minimum influence attributes * /
    change the attribute of A and B;
if (NC>ND)
    change the D's attribute;
  else change the C's attribute;
if (NG>NH && NH>NE+NF)
    change the H's attribute;
  else if (NG>NH && NH>NE+NF)
    change the attribute of E and F;
    else
      change the G's attribute;
```

图 3 (a)中 RMIA 的计算过程

```
/ * generate the minimum influence attributes * /
if (NG>NH)
  compute NB+NC+NH;
else
  compute NB+NC+NG;
if (NE<NF)
  compute ND+NE;
        if (NA<ND+NE)
      change the A's attribute;
    else
    change the attribute of E and D;
  else compute ND+NF;
if(NA<ND+NF)
      change the A's attribute;
    else
    change the attribute of F and D;
```

图 4 (b)中 RMIA 的计算过程

定义 4(简单的基于密钥策略的属性加密，SKP-ABE，simple key-policy attribute-based encryption) 和 KP-ABE 相似，包括以下几个算法：

Setup 定义属性域 $u=\{0,1,2,\cdots,N\}$，对 $\forall i\in u$，为其随机选择 $t_i\in Z_p$，且随机选择 $y\in Z_p$。这个算法主要用来生成公共参数 PK 和主密钥 MK。

$$PK=(T_0=g^{t_0},\cdots,T_n=g^{t_N},Y=e(g,g)^y,VN)$$

$$MK=(t_0,t_1,\cdots,t_N,y,VN)$$

其中 PK 是公开的，而 MK 是由数据属主 DO(Data Owner)自行保存。其中 $i=0$ 是关键属性 CA。

Encryption(M,I,PK) 为了在数据文件的属性集 I 和公开参数 PK 下对消息 M 进行加密，随机选择 $s\in Z_p$，生成如下密文：

$$E=(I,\tilde{E}=MY^s,\{E_i=T_i^s\}_{i\in I},VN)$$

此外，数据属主还要对密文生成如下签名：

$$\delta_o=Sig(E,VN)$$

数据属主 DO 会将 E 及其签名 δ 发送给云存储服务器。

Key Generation(T,MK) 该算法用来生成用户私钥 SK。首先，在确保根节点的值为 y 的情况下按照 SAST 的方式为每一个叶子节点赋值$\{y_i\}_{i\in L}$，其中 L 为 T 的叶子节点所对应的属性集。生成如下私钥：

$$SK=\{sk_i=g^{\frac{y_i}{t_i}},VN\}_{i\in L}$$

生成私钥 SK 后，DO 会将 SK 及其签名 $\delta=Sig(SK,VN)$通过安全信道发送给用户；将除 CA 所对应的私钥组件外的其它所有私钥组件和签名发送到云端。

Decryption(E,SK) 指定解密过程为一个递归算法。同时假设访问结构树被内嵌于私钥中，只有当数据文件的属性满足用户的访问结构且密文版本号值和私钥组件版本号值相同时，才能成功进行解密。首先，定义一个递归函数 $DecryptNode(i)$。如果节点 i 为叶子节点：

$$DecryptNode(i)=\begin{cases}e(E_i,SK_i)=e(g,g)^{y_i s}, & i\in I\\ \perp, & \text{其它}\end{cases}$$

对于内部节点 n，则有：

$$DecryptNode(n)=\begin{cases}DecryptNode(RChild(n)), & n\text{ 为'or'}\\ \underline{DecryptNode(RChild(n))}\\ DecryptNode(LChild(n)), & n\text{ 为'and'}\end{cases}$$

由上递归最终可以得到：

$DecryptNode(r)=e(g,g)^{ys}$

最终得到明文 M：

$$Decryption(E,SK)=\frac{E}{DecryptNode(r)}=\frac{Me(g,g)^{ys}}{e(g,g)^{ys}}=M$$

3.3 用户授权

当 DO 为新用户 u 授予访问权限时，要为其分配相关的私钥及建一棵 SAST。

1)DO 先为新用户 u 分配一个 ID_u 并建立一棵访问结构树 T_u；

2)且用 Key Generation 生成私钥 SK；

3)将$(T_u,ID_u,SK,\delta_o=Sig(T_u,ID_u,SK))$通过安全信道 SSL 发送给新用户 u；

4)令 $w=(ID_u,\{j,sk_j\}_{j\in L\setminus CA})$，并将$(w,\delta_o=Sig(w))$发送到云端。

当云服务端收到(w,δ_o)，先验证 δ_o，如果正确则将 w 存储到 UL 中，并将新出现的属性版本存到 AUL 中，并刷新 AUL 中属性 i 引用的次数 N_i 和 U_i，完成用户的授权。

3.4 用户撤销

3.4.1 用户权限撤销

在整个用户权限撤销过程中，为达到高效性，其核心思想是：首先，DO 通过 RMIA 得到撤销用户 u 所需改变的最小影响属性集 I_1，同时结合代理重加密将私钥组件和密文组件的重加密计算转移到云端，大大减低了 DO 的计算量，同时因为引进了 RMIA，也减少了云端的计算。

用户 u 权限撤销过程具体描述如下：

/ * revoke user * /

DO:

$y\leftarrow y'$, $i\leftarrow i'$, $t_i\leftarrow t_i'$, $y_i\leftarrow y_i'$

$rk_{i1}:\frac{t_i'}{t_i}$, $rk_{i2}:\frac{y_i'}{y_i}$, $rk_r:\frac{y'}{y}$

DO generates new CA′ and y_o, sk_o;

$V=(ID_u,rk_{i1},rk_{i2},rk_r,\delta_o)$, $\delta_o=Sig(ID_u,rk_{i1},rk_{i2},rk_r)$;

DO sends V to Cloud Servers.

Cloud Servers:

Once received V, cloud servers deletes u from UL; updates attributes' version, U_i, N_i, rk in AUL; updates VN;

/ * private key SK update * /

$sk_i'=(sk_i)^{rk_{i1}^{-1}}rk_{i2}=(g^{\frac{y_i}{t_i}})^{\frac{t_i}{t_i'}\frac{y_i'}{y_i}}=g^{\frac{y_i'}{t_i'}}$;

/ * ciphertext update * /

$E_i'=(E_i)^{rk_{i1}}=(g^{t_i s})^{\frac{t_i'}{t_i}}=g^{t_i' s}$

$\tilde{E}'=(\tilde{E})^{rk_r}=(Me(g,g)^{ys})^{\frac{y'}{y}}=M^{\frac{y'}{y}}e(g,g)^{y's}$

Order $M'=M^{\frac{y'}{y}}$;

图 5 用户权限撤销时的相关计算

DO 随机选择新的参数 $y' \in Zp$ 时，在 SAST 树中，对于 and 关系节点，要保持不要更新的属性的 y_i 值不变。一旦用户对数据文件进行访问，云服务端就会把密文当前的版本号 VN 发送给 DO，数据属主会查看收到的版本号值是否和自己的相同，如果不同，则将已生成的最新版本的代理重加密密钥 rk 发送给云服务端(因为 DO 的 VN 的版本号值是最大的，只要有用户撤销，DO 就会更新 rk)，云服务端根据代理重加密密钥 rk，生成私钥组件和密文组件的最新版本，并将生成的最新版本的密文组件保存到密文 E' 中。

当一个用户访问数据文件时，首先看数据文件的属性是否满足用户的访问结构来判定其是否为合法用户，如果是，云服务端再查 UL，得到其相关的私钥组件，检查其除 CA 外所有属性的版本是否是最新的，如果不是，则更新其版本号，并将私钥组件和密文组件的最新版本、最新版本的 rk_r 发送给用户。在用户端，一旦收到最新版本的私钥和密文，则用上文提到的 SKP-ABE 算法进行解密，得到：

$M' = M_y^{y'}$；

最终得到明文：

$(M_y^{y'})^{rk_r^{-1}} = (M_y^{y'})^{y^{-1}} = M$

3.4.2 *用户属性级撤销*

在本文中，把对同一文件组(具有相同属性集的文件集合，记为 $Files_i$)有访问权限的用户划分到同一个用户组(记为 $Users_i$)，对于不同的文件组有不同的用户组，并且有不同的关键属性 CA。对用户的撤销，应该是基于属性级的而不是基于系统级的。一个用户可以访问多个文件组，在本文中当一个用户从一个文件组中被撤销权限时，只要其访问结构能继续满足其他文件的属性，还是可以继续进行访问，并没有撤出整个系统。

4 安全性分析

将从以下几个方面对本文的访问控制策略进行安全性分析：

1)数据的保密性。数据密文是用简化的 KP-ABE 进行加密生成的，要从密文 E 中解密得到明文则要破译 KP-ABE 密码体制；对于云端，CSP 只拥有部分的私钥组件，并不拥有 CA 所对应的私钥组件，因此并不能成功进行解密；而对于访问结构不满足文件属性的用户，不能成功进行解密，因为没有足够的私钥组件对密文组件进行匹配，且没有正确的 CA。因此，本方案具有数据保密性。

2)抗联合攻击性。用户是进行立即撤销的，用户一旦从该文件组中被撤销，就不能再访问该文件组；对于用户和已撤销用户之间的联合攻击，由于用户是进行立即撤销的，当一个用户撤销时，通过 RMIA 得到撤销用户所需改变的最小影响属性集 I_1，对于 I_1 中属性对应的私钥将会进行更新，版本号值 VN 将会加 1，而这些更新好的私钥组件将会发送给未被撤销的用户，所以即使用户和已撤销的用户进行联合攻击也不能成功进行解密；对于已撤销用户和 CSP 的联合攻击，由于部分的私钥组件和相关密文组件的重加密是在云端进行的，因此 CSP 只要再拥有正确版本号 CA 所对应的私钥组件就能成功进行解密，但是已撤销用户只拥有原来的 CA 版本所对应的私钥组件，所以即使云端和已撤销用户进行联合攻击也不能成功进行解密。所以在将几乎所有私钥组件和密文组件转移到云端进行重加密的情况下，方案是安全的。

5 性能分析

本节将从用户权限撤销和 SKP-ABE 算法两方面进行性能分析。

5.1 用户权限撤销的性能分析

在文献[4]中，用户权限撤销是通过给每一个属性贴上终结日期来实现的，而日期是用树来表示的，当日期过大时，树将会很庞大。同时附有终结日期的属性必须和加密数据当局和密钥发行当局进行协同。

在文献[9]中，用户权限撤销采用的是直接删除该用户在访问结构树中所对应的子树，而在子树中有些属性是共享的。在其所采用的例子中，只考虑了仅有一个属性的用户的撤销。

在本文中，用户权限撤销所采用的是 RMIA 算法，因为属性是被多个用户共享，并不能直接删除整棵子树，所以本文对使用频率较高的属性不进行更新，从而减低由于属性撤销对用户所造成的影响，同时也将减少云端的计算量。

5.2 SKP-ABE 的性能

Setup　这个算法的计算量是因 PK 和 MK 的生成而产生的，要在 G_0 上进行$(N+1)$次的群乘法运算，是由 DO 来完成。

Encryption　这个算法用来进行加密，实际上就是有新的文件产生。本文是直接对数据文件进行加密，要在 G_0 上进行$|I|$次的群乘法运算和在 G_1 上进行一次群乘法运算，还要进行一次签名，是由 DO 来完成。

用户权限授予时，DO 为用户建立一棵 SAST 树并用 Key Generation 算法为用户生成私钥，要在 G_0 上进行$|L|$次的群乘法运算，还要进行两次签名。云服务端要进行一次认证，用户也要进行一次认证。并且在私钥生成过程中，不再为树的每一个节点生成多项式，而是产生随机数。

用户权限撤销过程中，计算量主要是因为属性和密文的更新而产生。DO 对密文进行重加密，要在 G_1 上进行一次群乘法运算。由 RMIA 决定最小影响属性集$|I_1| < |L| - 1$，DO 要生成 $2|I_1|+1$ 个代理重加密密钥；而云服务端要在 G_0 上进行 $2|I_1|$次的群乘法运算，如表 1 所列。SKP-ABE 相对于 SCP-ABE[9]：在 SCP-ABE 中由于数据文件和访问结构相关联，在用户权限授予和撤销时，要对整个文件的访问结构树进行更改；而在 SKP-ABE 中由于用户对应的是访问结构树，数据文件对应的是属性，在用户权限更新和撤销时，只要更新属性即可。

表 1　EACS 的复杂度

Operation \ Complexity	DO	Cloud Servers
Setup	O(N+1)	0
Encryption	O(|I|+1)	0
User Authorization	O(|L|)	Almost negligible
User Revocation	O(1)	O(2|I_1|)

SKP-ABE 相对于 KP-ABE[6]：在 KP-ABE 中是通过多项式对树中的节点进行赋值；而在 SKP-ABE 中是直接用随机数进行赋值。

结束语　本文提出了一种高效的、灵活的访问控制方法 EACS。该方案基于 SKP-ABE 算法，并在用户权限撤销时，先用 RMIA 计算得到最小影响属性集，并结合代理重加密将

几乎所有的私钥组件和相关的密文组件转移到云端，大大降低了 DO 的计算量，达到高效性；并可以灵活地对用户进行权限授予和权限撤销；同时，由于 RMIA 的引入，也减少了云服务端的计算量，但仍没有解决计算量随属性数量线性增长的问题，同时，KP-ABE 在判断访问权限上也存在很大问题，当文件的属性范围少于用户属性范围时（即用户在满足文件属性的前提下，增加其它属性），用户却不能访问该文件。由于 DO 计算代价和 CSP 的计算代价是相关的，要想使 DO 的计算代价降低则需在保证安全的前提下将部分的计算转移到云端，这样势必会增加云端的计算量，所以在以后的研究中，在降低 DO 的计算量的同时，也使云端的计算量降低，这些都是亟需要解决的问题。

参 考 文 献

[1] Cachin C, Keidar I, Shraer A. Trusting the cloud [J]. ACM SIGACT News, 2009, 40(2): 81-86

[2] Sahai A, Waters B. Fuzzy identity based encryption [C]// Proceedings of the Advances in Cryptology (EUROCRYPT). Aarhus, Denmark, Berlin, Heidelberg: Springer-Verlag, 2005: 457-473

[3] Goyal V, Pandey O, Sahai A, et al. Attribute based encryption for fine-grained access control of encrypted data [C] // Proceedings of the 13th ACM Conference on Computer and Communications Security (CCS '06). New York, NY, USA: ACM, 2006: 89-98

[4] Bethencourt J, Sahai A, Waters B. Ciphertext-policy attribute-based encryption [C] // Proceedings of the 2007 IEEE Symposium on Security and Privacy. Oakland, California, USA, Washington, DC, USA: IEEE Computer Society, 2007: 321-334

[5] Yu S, Wang C, Ren K, et al. Attribute based data sharing with attribute revocation [C] // Proceedings of the 5th International Symposium on Information, Computer and Communications Security (ASIACCS 2010). New York, NY, USA: ACM, 2010: 261-270

[6] Yu S, Wang C, Ren K, et al. Achieving secure, scalable, and fine-grained data access control in cloud computing [A] // Proceedings of IEEE INFOCOM 2010 [C]. San Diego, CA, 2010

[7] 洪澄，张敏，冯登国. AB-ACCS：一种云存储密文访问控制方法 [J]. 计算机研究与发展，2010，47（增刊）：259-265

[8] 吕志泉，张敏，冯登国. 云存储密文访问控制方案 [J]. 计算机科学与探索，2011(09)

[9] 洪澄，张敏，冯登国. 面向云存储的高效动态密文访问控制方案 [J]. 通信学报，2011(07)

[10] Blaze M, Bleumer G, Strauss M. Divertible protocols and atomic proxy cryptography [C] // Proc. of EUROCRYPT '98. 1998

（上接第 37 页）

系统的性能；而路预测算法在单核处理器中已被证明能够很大程度地降低系统功耗。本文将基于公平性的 Cache 划分策略与路预测算法相结合，运用于多核处理器下的共享 L2 Cache 中，在保证系统性能的同时，很好地降低了 L2 Cache 的动态能耗。

众核处理器是未来处理器发展的趋势，随着片上集成的核数增多，对低功耗的要求也会越来越显著。在未来的工作中，将会进一步优化基于 Cache 划分的路预测 L2 Cache 算法，使之在核数更多的处理器系统下仍保持良好的低功耗效果。

参 考 文 献

[1] Ijaykr V, Ishman N, Kandemir M, et al. Energy-driven integrated hardware-software optimizations using simple-power [A] // Proceedings of the 27th Annual International Symposium on Computer Architecture, 2000 [C]. California: IEEE, 2000: 95-106

[2] Lu Jun-yang, Guo Yao. Energy-Aware Fixed-Priority Multi-core Scheduling for Real-Time Systems [A] // The 17th IEEE International Conference on Embedded and Real-Time Computing Systems and Applications, 2011 [C]. California: IEEE, 2011: 277-281

[3] Wang Wei-xun, Mishra P, Ranka S. Dynamic Cache Reconfiguration and Partitioning for Energy Optimization in Real-Time Multi-core Systems [A] // Design Automation Conference. New Jersey, 2011 [C]. IEEE, 2011: 948-953

[4] Reddy R, Petrov P. Cache partitioning for energy-efficient and interference-free embedded multitasking [J]. ACM TECS, 2010, 9(3): 1-35

[5] Inoue K, Ishihara T, Murakami K. Way-Predicting Set-associative Cache for High Performance and Low Energy Consumption [A] // Proceedings on Low Power Electronics and Design. New Jersey, 1999 [C]. IEEE, 1999: 273-275

[6] Chen H-C, Chiang J-S. Low-power Way-predicting Cache Using Valid-bit Predecision for Parallel Architectures [A] // Proceedings of the 19th International Conference on Advanced Information Networking and Applications, 2005 [C]. California: IEEE, 2005: 28-30

[7] Chung C-M, Kim J. Low-power L2 cache design for multi-core processors [J]. Electronics Letters 29th, 2010, 46(9): 618-120

[8] Suh G E, Rudolph L, Devadas S. Dynamic Partitioning of Shared Cache Memory [J]. Journal of Supercomputing, 2004, 28(1): 7-26

[9] 所光，杨学军. 面向多线程多道程序的加权共享 Cache 划分 [J]. 计算机学报，2008，31(11)：1938-1947

[10] Muralidhara S P, Kandemir M, Raghavan P. Intra-Application Cache Partitioning [A] // IEEE International Parallel & Distributed Processing Symposium, 2010 [C]. New Jersey: IEEE, 2010: 1-12

[11] Kim S, Chandra D, Solihin Y. Fair cache sharing and partitioning in a chip multiprocessor architecture [A] // Proceedings of 13th International Conference on PACT, 2004 [C]. California: IEEE, 2004: 111-122

[12] Hsu L R, Reinhard S K, Iyer R. Communist utilitarian, and capitalist cache policies on CMPs: Caches as a shared resource [A] // Proc of PACT-15, 2006 [C]. California: IEEE, 2006: 13-22

[13] Gabor R, Weiss S, Mendelson A. Fairness and throughput in switch on event multithreading [A] // 39th IEEE/ACM International Symposium on Microarchitecture, 2006 [C]. California: IEEE, 2006: 149-160

[14] Iyer, Ravi. A framework for enabling QoS in shared caches of CMP platforms [A] // Proceedings of ICS-18, 2004 [C]. USA: ACM, 2004: 257-266

[15] 隋秀峰，吴俊敏，等. ARP：同时多线程处理器中共享 Cache 自适应运行划分机制 [J]. 计算机研究与发展，2008，45(7)：1269-1277