**基于区块链的访问控制创新性研究**

**区块链的核心技术：**

分布式账本

多个节点共同完成

非对称加密

存储在区块链上的交易信息是公开的，但是账户身份信息是高度加密的，只有在数据拥有者授权的情况下才能访问到，从而保证了数据的安全和个人的隐私。

共识机制

少数服从多数

智能合约

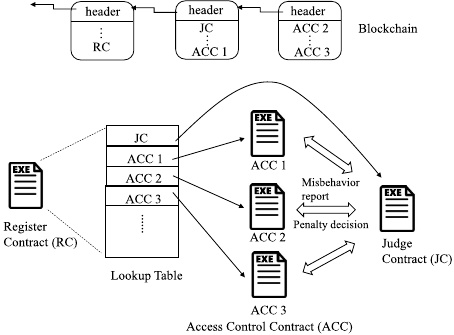
既定的套路

**基于智能合约的区块链技术，实现物联网分布式可信接入控制**

提出一种基于智能合约的访问控制框架，该框架由多个访问控制合约（ACCs）、一个裁判合约（JC）和一个注册合约（RC），实现对物联网系统的分布式和可信访问控制。

在这个框架中，每一个ACC为每一对主客体提供一个访问控制方法，实现基于预定义访问控制策略的静态访问权限验证和基于主体行为的动态访问权限验证。一旦被一个主体调用进行访问控制，ACC将由系统中的大多数参与者运行和验证，确保访问控制的可信性。ACCs还提供增加、更新和删除访问控制策略的功能。为了促进ACCs的动态验证，JC提供了一种不当行为判断方法，通过接收来自ACCs的受试者不当行为报告，判断不当行为并返回相应惩罚。为了管理访问控制和不当行为判断策略，RC不仅能注册方法信息（比如名称、主体、客体、智能合约等），还提供注册新方法和更新或删除现有方法的功能。

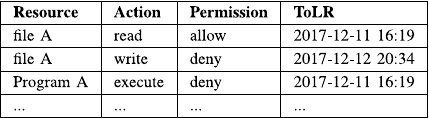
**访问控制框架结构**



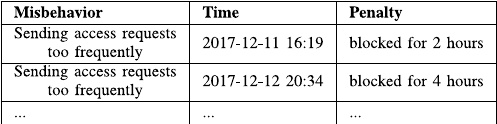
ACC：

一对主客体可以与多个ACC关联，但是一个ACC能且只能与一对主客体相关联。

ACC策略事例如下：



ACC还有对于不当行为的策略报告和相应惩罚（为JC判断不当行为提供便利）：



ACC还提供以下主要ABIs来管理策略和实现访问控制：

policyAdd()

policyUpdate()

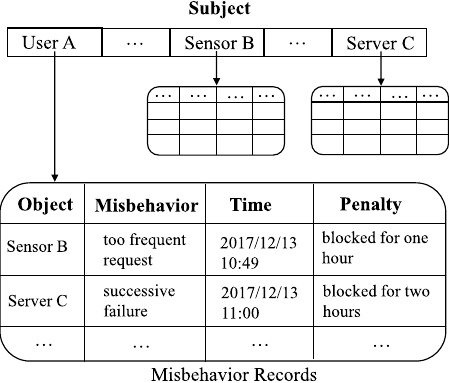
policyDelete()

accessControl：接收访问控制所需的信息，并返回访问结果和惩罚。实现静态和动态验证。

setJC()：执行JC的判决，接收JC地址并设置JC

deleteACC()

JC：



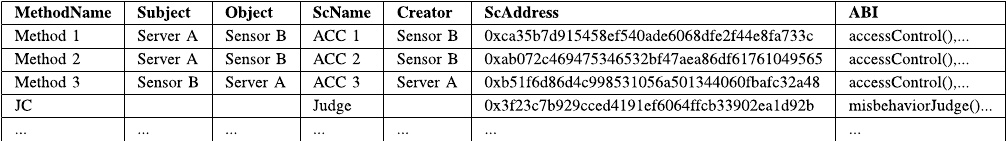
JC接收ACCs的不当行为，根据历史记录做判定，返回惩罚结果给ACCs

JC主要提供以下主要ABIs，用于判断不当行为、确定处罚、管理JC：

misbehaviorJudge()

deleteJC()

RC：



RC提供主要以下ABIs：

methodRegister()

methodUpdate()

methodDelete()

getContract()：接收MethodName，返回ACCs和JC的address和ABIs。

**该框架功能：**

实现主客体之间的访问控制

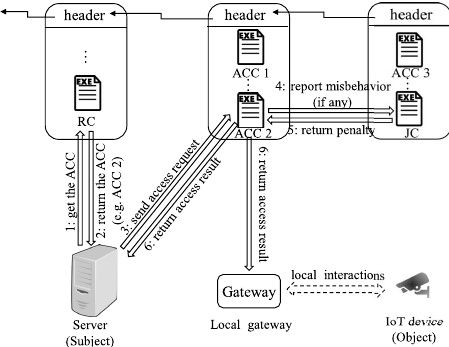
添加、更新、删除ACC策略

登记和更新不良行为的判断方法

注册、更新、删除访问控制方法

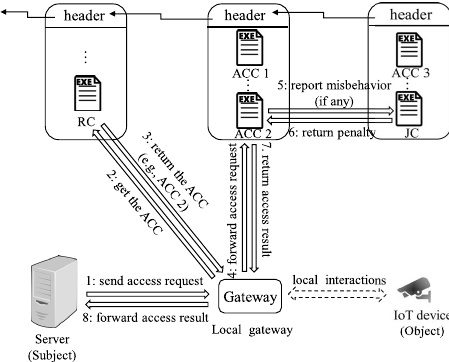
功能实现流程举例：

主体调用ACC的事例，服务器（主体）希望访问物联网设备（客体）资源。需要增设一个访问控制的过程。



1. 服务器用RC的getContract ABI来检索ACC
2. RC向服务器返回ACC的地址和ABI
3. 服务器发送访问请求，包含访问控制所需的信息，调用ACC的accessControl ABI。
4. 在访问控制过程中，如果检测到服务器（主体）的一些不当行为，ACC将发送消息调用JC的misbehaviorJudge ABI。
5. 一旦misbehaviorJudge ABI完成了对错误行为的判断并决定了惩罚，它将惩罚返回给ACC。
6. 访问控制过程结束后，访问结果将被返回给主客体。

客体访问主体，服务器给设备发送请求，客体回复服务器（访问控制体现在客体设备回复主体的过程中）



本地网关作为物联网设备的代理，通过安全的本地交互，实时通知物联网设备访问控制状态。

**框架实现**

以太坊平台

account/address：外部账户、智能合约账户

smart contract：提供功能或者应用程序二进制接口

transaction and message

blockchain：包含交易和智能合约

mining：挖掘是一个过程，它通过称为矿工的节点将新块包含到区块链中。

系统配置

为了利用以太坊平台实现访问控制框架，需要对系统进行以下配置：

每对主客体必须与一个以太坊账户关联，部署智能合约，并在访问控制过程中识别自己

以太坊客户端运行在除物联网设备以外的系统所有对等点上

物联网设备有当地网关做代理

**案例研究**

**新思路**

基于属性的访问控制合约

attribute

subject

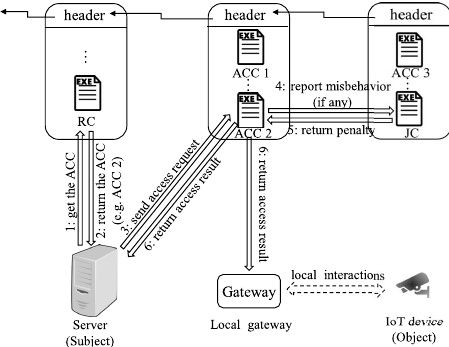
object

operation

policy

environment conditions

实际举例



ACC维护列表：（这里可以将属性单设一个列表）

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| Subject’s attribute | Object’s attribute | Resource | Action/operation | Environment condition | Rules | Decision | ToLR |
| Name/affiliation/clearance | Type/owner/classification | File A | Read | 当前时间/安全等级/生产or测试环境 | 服务器白天在安全的生产环境可以读取文件A | Allow | 2022年5月19日10:35 |

ACC维护的不当行为列表：

|  |  |  |
| --- | --- | --- |
| Misbehavior | Time | Penalty |
| 服务器夜间在非生产环境下频繁读取文件A | 2022年5月19日22:35 | 中断 |

JC维护列表：

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| Subject | Object | Rules | Misbehavior | Decision | Time | Penalty |

参考文献

**基于智能合约的分类分级属性访问控制方法**

算法一

访问控制策略生成（既定）

算法二

属性信息查询

算法三

访问控制策略查询

算法四

策略判决

**基于区块链的细粒度物联网访问控制模型**

ABAC-B

属性访问控制算法

访问控制实施点向权限裁决点发送一个三元组请求的消息 (s.id，o.id，s.p)。分别是主客体的唯一id，主体和客体所请求的操作s.p。当收到访问控制请求时权限裁决点通过调用GetAttribute()，GetStrategy()分别获得用户的属性s.a，客体属性o.a。依据属性条件函数ACF()判断属性是否合规，同时结合计算所得的信任度 Tt 确认用户设备身份、类型和信任评价。使用角色条件函数AFC()根据角色属性ra找到与设备相关访问策略合约Decision()的地址。调用裁决合约Decision()利用获得o.a，s.a，s.p做出决策,判断是否授予权限。

算法：

输入：(s.id，o.id，s.p)

输出：token

(1)RequestAccess(s.id，o.id，s.p)

(2)if o=msg.sender∧s. a≠NULL∧o≠NULL then

(3) s.a←GetAttributes(s)；ra，pa←s.a；

(4) o.a←GetAttributes(o)；

(5) If AFC(s.a)∈ACR∧Tt>Tr

(6) getDecision()←AFC(ra,pa);

(7) If Decision(o.a,s.a,s.p)=true then

(8) emit request event;

(9) token←CreateToken(s.a,o.a,s.p);

(10) return token;

(11) else return Access denied;

(12) endif

(13) else return Fail verification;

(14)end if

**基于区块链的大数据访问控制机制**

定义：

定义一.属性项是表示属性的基本单元，用{xAttrName=attrValue},(xAttrName∈attrSet,attrValue∈Range(xAttrName),x∈{s,r,a,e})表示，xAttrName表示属性名，attrValue表示属性值，为了对不同类型的属性表示方便，用x表示属性类型，s,r,a,e分别表示主体属性、资源属性、动作属性和环境属性。

定义二.属性元组是同类型属性项的集合，用xAttrTuple,x∈{s,r,a,e}表示，即：xAttrTuple:{(xAttrName1=attrValue1)∧(xAttrName2=attrValue2)∧…(xAttrNamei=attrValuei)}

定义三.属性访问请求由一组主体属性、资源属性、动作属性、环境属性组成用AAR:{sAttrTuple∧rAttrTuple∧aAttrTuple∧eAttrTuple}表示，AAR的含义是：属性为sAttrTuple的请求者在环境为eAttrTuple条件下，对资源rAttrTuple请求进行操作aAttrTuple。

定义四.访问控制策略：针对客体资源的访问控制规则，体现了资源拥有者的授权行为，规定了访问受保护资源所需要具有的属性集合，记为Policy:result(R,action,pid)←Θ{xAattrTupleSet}signature\_owner,x∈{s,r,a,e}。其中，Θ{xAattrTupleSet}表示由属性项集合xAattrTupleSet中的属性通过合取、析取等逻辑关系构成的逻辑表达式，pid表示策略id。当请求方所拥有的的属性使Θ{xAattrTupleSet}为真时，请求方能够被允许或拒绝对资源R进行action操作，result∈{Permit,Deny}。另外，策略需要被资源拥有着或策略发行方签名后在区块链中保存，保证发布策略的真实性。

访问控制合约层

PAP contract：访问控制策略管理，类似于ACC维护列表

PDP contract：访问控制请求判决，类似于JC维护列表

AA contract：实体属性管理（专门开的一个列表）

RC：注册列表

PAP contract

1. 解析AAR得到所请求的资源属性信息rAttrTuple；
2. 遍历区块链各区块内策略类事务数据块；
3. 获取BF（策略管理方法）；
4. 根据该BF所对应的哈希函数，计算rAttrTuple所对应的哈希值；
5. 若计算rAttrTuple得到的哈希值与BF中对应位全部为1，则在RELEVANT\_POLICY\_SET中添加该数据块内策略；否则，该数据块中无rAttrTuple的相关策略；
6. 当对所有区块内策略类事务遍历完成后，向PDP返回资源相关策略集RELEVANT\_POLICY\_SET用于权限判决。

输入：属性策略相关请求AAR，区块链blocks；

输出：RELEVANT\_POLICY\_SET.

AttrTuple=attributeParser(AAR);this=currentBlock;mark=null;

for i=1 to blocks.length do

{for j=1 to this.policy\_datablock.length do

{BF=getBloomFilter(this.policy\_datablock[i]);mark=1;

For k=1 to this.datablock.hashfunction.length do

{key[k]=hash(k,AttrTuple);

If (BF.NotContain(key[k]))mark=0;

}

If(mark=1) then

{RELEVANT\_POLICY\_SET.increase(this.datablock.transaction.policy);continue;}

Else continue;

}

}

Return RELEVANT\_POLICY\_SET;

PDP contract

1. 将所有待判决策略放入UNKNOW\_SET，作为预判决策略集；
2. 遍历预判决策略，分别得到4个结果集PERMINT\_RESULT\_SET，DENY\_RESULT\_SET，UNSATISFY\_RESULT\_SET，UNKNOWN\_RESULT\_SET；
3. 根据判决结果集，得出AAR请求的最终判决结果。（若存在冲突的判决结果，则根据优先级进行冲突消解）。

输入：属性访问请求AAR，访问控制策略集POLICY\_SET；

输出：策略判决结果PERMIT，DENY，UNKNOWN，UNSATISFY

UNKNOWN\_SET=POLICY\_SET；PERMINT\_RESULT\_SET=null；

DENY\_RESULT\_SET=null；UNSATISFY\_RESULT\_SET=null；

For i=1 to UNKNOWN\_SET.length do

{result=PolicyDecide(UNKNOWN\_SET[i]);

If (result=permit) then

{UNKNOWN\_SET.delete(UNKNOWN\_SET[i]);

PERMINT\_RESULT\_SET.add(UNKNOWN\_SET[i].PID);

}

Eles If(result=deny)then

{UNKNOWN\_SET.delete(UNKNOWN\_SET[i]);

DENY\_RESULT\_SET.add(UNKNOWN\_SET[i].PID);

}

Else If(result=unsatisfy)then

{UNKNOWN\_SET.delete(UNKNOWN\_SET[i]);

UNSTAISFY\_RESULT\_SET.add(UNKNOWN\_SET[i].PID);

}

}

if (PERMINT\_RESULT\_SET≠null && DENY\_RESULT\_SET==null) then

return PERMIT;

else if (DENY\_RESULT\_SET≠null && PERMINT\_RESULT\_SET==null) then

return DENY;

else if (PERMINT\_RESULT\_SET≠null && DENY\_RESULT\_SET≠null) then

return conflict\_handle(·);

else

return UNKNOWN;

AA contract

系统中可能存在多个AA，AA中存储主体属性、客体属性、动作属性和环境属性的属性值和属性关系列表。以事务的形式存储在区块链中。AA contract相当于提供AA查询服务的代理。

1. 接收属性查询请求attributeRequest;
2. 遍历数据块
3. 获取BF
4. 根据BF对应的哈希函数，计算对应属性的哈希值
5. 若计算哈希值与BF中对应位全部为1，则在RELEVANT\_ATTRIBUTE\_SET中添加该数据块内属性信息；否则无相关属性信息
6. 遍历完成后，将RELEVANT\_ATTRIBUTE\_SET作为响应进行返回。

输入：属性相关请求AAR，区块链blocks；

输出：RELEVANT\_ATTRIBUTE\_SET.

AttrTuple=attributeParser(attributeRequest);this=currentBlock;mark=null;

for i=1 to blocks.length do

{for j=1 to this.attribute\_datablock.length do

{BF=getBloomFilter(this.attribute\_datablock[i]);mark=1;

For k=1 to this.datablock.hashfunction.length do

{key[k]=hash(k,AttrTuple);

If (BF.NotContain(key[k]))mark=0;

}

If(mark=1) then

{RELEVANT\_ATTRIBUTE\_SET.increase(this.datablock.transaction.policy);continue;}

Else continue;

}

}

Return RELEVANT\_ATTRIBUTE\_SET;

实验过程

熟悉XACML提供标准策略一致性测试包，包含属性集和策略集。

寻找PAP、PDP、AA三个合约代码

下一步工作

[Blockchain-ABAC](https://github.com/etsrohan/blockchain-ABAC)

基于区块链的属性访问源码找到，接下来做的工作

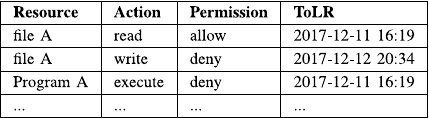
1. 实验。分析源码融入原文代码。
2. 报告。分析源码写思路写算法（最好是可以和看过的伪代码相似）

AccessControlContract、ObjectAttributeContract、SubjectAttributeContract、PolicyManagementContract

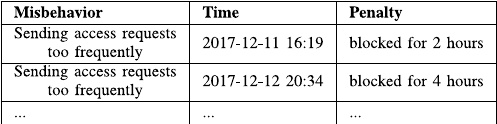
**下一步的工作（紧迫）**

对异常访问节点的惩罚机制

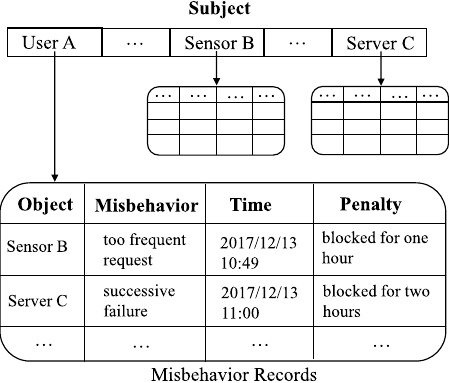
ACC策略事例如下：



ACC还有对于不当行为的策略报告和相应惩罚（为JC判断不当行为提供便利）：

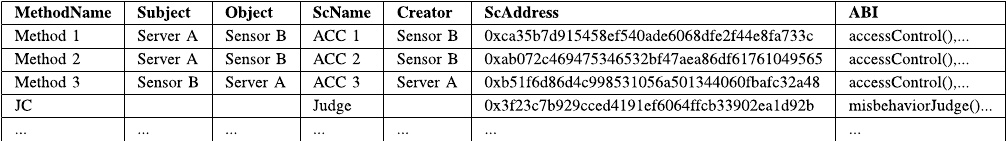


JC：



JC接收ACCs的不当行为，根据历史记录做判定，返回惩罚结果给ACCs

RC：



增加一步，信誉积分机制

以防访问主体屡教不改，增加信誉积分机制。

两部分：

如何判定访问主体非法

如何加入信誉积分机制，非法一次，扣除相应积分，直至扣到阈值以下，剔除非法节点。

[添加恶意行为检测机制](https://flywh.github.io/2020/credit-based-mechanism/#jump)

添加恶意行为检测（动态访问控制）

对原文合约架构进行调整

ACC不变，