基于区块链的车辆数据监管和任务协同平台

**项目团队：林怡静、王苗苗、吴孟晓、柴泽**

**所属院校：北京邮电大学**

**指导教师：高志鹏**

摘要

该部分由林怡静补充

目录

TODO List（20210905）

1. 最重要的事情放在前面强调，每个部分都要留痕，即写文档、写注释，包括区块链的部署、合约的编写、合约的测试以及预言机的设计与开发等，以方便整个项目团队更加快速理解。

2. 9.10提交PPT和部分的项目文档，以PPT为主，如果项目文档可以写完的话，提交两项；因此，9.6开始需要制作PPT和撰写项目文档。

PPT由王苗苗结合“火币和直播产品运营”的两个PPT制作，大概在9.9之前发送给林怡静一起修改；

项目文档由林怡静、柴泽、王苗苗结合“火币和数字湾区”的两个文档制作，大概在9.9之前完成项目背景和技术内容的撰写，发送给王苗苗一起修改；

3. 9.6开始团队一起讨论文档的需求分析和详细设计部分，并进行修改；如有必要，9.7号讨论根据9.6的修改后的版本；该部分讨论完之后，进行合约编程；

合约编程的跨域共享、计算共享和数据共享主要由吴孟晓完成，需要完成的合约分别包括计算预言机、跨链预言机、数据预言机3个部分，共6个合约（每个部分2个合约，一个是代理合约，一个是应用合约）；合约编程可参考中兴跨链项目的代码，设计和实现请看下文的技术路线和系统研发部分；时间节点为9.18之前完成合约的编写。

这里需要说明的一点是，在完成跨链预言机开发的过程中，需要考虑两种情况，一种是同构链的开发，一种是异构链的开发；同构链即xuperchain之间，异构链可在xuperchain和其他区块链之间共享，这里的“其他区块链”由吴孟晓同学确定。

合约编程的授权认证由王苗苗完成，需要完成的部分包括合约的编写以及链下的调用，该部分内容可参考王苗苗的上一篇投稿论文。

合约测试主要由柴泽完成，需要完成上述合约的单元测试和功能测试。通常来说测试方法有两种，一种是直接用合约编程语言测试，一种是用SDK测试。具体采用哪种测试，需要去比赛官网查看比赛的要求；时间节点为9.26之前。

4. 9.27开始，由林怡静和王苗苗完成预言机部分的开发，时间节点为10.10之前完成。具体的分工如下，王苗苗负责完成域内共享和授权认证的链下预言机部分，林怡静负责完成跨域共享和计算共享的链下预言机部分。这里的四个部分都需要建立在合约编程无bug、有测试用例的情况，并且时间比较紧张，因此希望前述两个步骤最好可以在时间节点之前的两三天完成，为后续整个项目的完成预留时间。

5. 最后，假如BSN的比赛进入决赛，需要有前端的页面展示；xuperchain的开放创新比赛无需前端。在完成上述内容并且进入决赛之后再进行探讨。

1.项目介绍

该部分由柴泽补充

背景

特斯拉女车主维权事件

滴滴低调赴美上市引发的数据安全争议

近日出台的关于车辆数据的相关法律法规

存在的问题

针对我们的三个技术路线，有针对性地指出当前车辆数据监管和任务协同存在的问题

竞争对手

市场上对于车辆数据的使用还处于野蛮生长时代，迫使国家不得不出台数据安全法等一系列措施进行调控，以保护数据安全。（也就是说，目前市场上还没有竞争对手）

提一下，我们的解决方案还可以适配到其他的领域

解决方案

1. 针对xxx问题，提出xxx
2. 针对xxx问题，提出xxx
3. 针对xxx问题，提出xxx
4. 进行系统验证，实现了功能

2.技术方案

需求分析

技术路线

技术方案和系统研发的区别是，该部分讲系统设计的原理，即为什么要这么设计这个系统。

授权认证

该部分由王苗苗补充

域内共享

该部分由王苗苗补充

跨域共享

该部分由林怡静补充

计算共享

该部分由林怡静补充

集成上述四个研究点的平台

该部分由林怡静补充

创新点

系统研发

最重要的再次写在前面，与中兴的跨链项目不同，本次的三个共享不在链上聚合，即在链下选择一个节点进行聚合之后，返回到链上验证聚合签名是否有效。这里的签名采用bls签名，由于不知道某些合约语言是否支持该签名，如若不支持，则底层采用固定的写法即可。

同时，为方便吴孟晓同学的理解，当前版本的系统研发部分内容包含了技术方案的内容，等到后续可以删除。下文给的数据类型只是参考，具体实现根据情况而定。

该部分内容写系统的数据结构和功能函数。

授权认证

该部分由王苗苗补充

域内共享

域内共享，即大量的敏感车辆数据在区块链网络共享。主要的特点是，数据接口、数据文件等多源异构数据存储于链下某个数据中心（可采用文件读写的方式实现），链上存储的只是元数据（例如，数据存储位置、数据摘要等）。链上对数据的调用需要利用预言机根据元数据向数据中心请求数据，由预言机返回到链上。除此以外，域内共享还可做到，对外界实时数据的可信共享。区块链智能合约的确定性特性使得其无法共享链下的实时数据。以航班信息为例，区块链无法获取实时的航班信息。

数据预言机

数据预言机属于链下部分，是一个去中心化的p2p网络，其中的节点通过区块链的身份与区块链网络交互。数据预言机的功能是利用事件event机制监听链上触发的事件，随后转发给数据预言机的其他节点，其他节点根据event内携带的请求到指定的数据源获取数据，发送给leader节点，leader节点聚合各个节点的签名以及数据，并返回到链上。

应用合约

有域内共享需求的客户端节点可调用应用合约的数据获取请求。

/\*\*

from: 存在域内共享需求的客户端节点

datasource: 指定数据源获取数据

metadata: 元数据，可以是数据摘要等

expiration: 请求的终止时间

\*\*/

func Request(from string, datasource string, metadata string, expiration string) {

判断expiration是否合理；

调用代理合约的Accept功能；

}

应用合约接收代理合约的回调

func Callback(taskid string, data string) {

接收数据并修改状态；

}

应用合约中还可包含一些功能性函数，以车联网为例，可修改部分车辆的配件信息，具体内容可参考经典的fabcar代码；

域内共享代理合约

域内共享合理合约首先接收应用合约的请求，并触发链下事件；

func Accept(from string, datasource string, metadata string) {

将当前请求加入队列，并生成任务id；

触发域内共享的事件；

}

数据预言机执行请求；

随后链下预言机返回聚合结果之后进行回调；

func Callback(taskid string, data []string, asig []string, pks []string ) {

判断当前的区块时间是否在expiration之内；

根据taskid获取from；

验证链下的预言机leader节点的签名是否正确

参考代码(如果受限于区块链本身，那么这部分内容可以固定写死)

聚合部分

for i := 0; i < N; i++ {

pks[i] = node.blslog.pks[i].Marshal()

msgs[i] = node.blslog.msgs[i]

sigs[i] = node.blslog.sigs[i]

}

asig := Aggregate(sigs)

验签部分

//不能直接json.unmarshal，需要采用bn256自己的函数

asig,\_ := new(bn256.G1).Unmarshal(verified.Asig)

length := len(verified.Msgs)

pks := make([]\*bn256.G2,length,length)

for i:=0;i<length;i++{

pks[i],\_= new(bn256.G2).Unmarshal(verified.Pks[i])

}

ok := AVerify(asig,verified.Msgs,pks)

if ok {

fmt.Println("request success!!")

回调data给应用合约；

应用合约回调给from；

将当前的跨域共享请求从队列中删除；

} else {

fmt.Println("request fail!!")

等待；

}

}

跨域共享

跨域共享的前提条件是，网络中存在多条区块链，多条区块链之间的数据共享即为跨域共享。与普通的跨链有所区分的是，这里需要考虑到跨域共享的监管问题。因此，跨域预言机中将会有超级节点，当跨域预言机网络中超过2/3节点投票时方可完成共识。

通常来说，跨链需要实现双向的，但是为方便省事，项目只需要完成xuperchain到其他链的即可。

跨链预言机

跨域预言机属于链下部分，是一个去中心化的p2p网络，其中的节点通过区块链的身份与区块链网络交互。跨域预言机的功能是利用事件event机制监听链上触发的事件，随后转发给数据预言机的其他节点，其他节点根据event内携带的请求进行链下签名共识，将共识后的结果发送给leader节点，leader节点聚合各个节点的签名以及数据，并返回到另一条区块链上。

应用合约

源区块链上有跨域共享需求的客户端节点可调用应用合约的数据获取请求。

/\*\*

from: 源区块链的应用合约地址

to: 目的区块链的应用合约地址

metadata: 元数据，跨域共享的数据内容

expiration: 请求的终止时间

\*\*/

func Request(from string, to string, metadata string, expiration string) {

判断expiration是否合理；

调用代理合约的Accept功能；

}

应用合约接收代理合约的回调

func Callback(taskid string, data bool) {

接收数据并修改状态；这里的data指跨链是否成功

}

跨域共享代理合约

源区块链上代理合约接收应用合约的请求

func Accept(from string, datasource string, metadata string) {

将当前请求加入队列，并生成任务id；

触发跨域共享的事件；

}

跨域预言机向目的区块链转发请求

目的区块链上代理合约接收源区块链的数据请求，目的区块链上的代理合约调用目的区块链上的应用合约执行相应的状态修改。

func AcceptCrossChain(taskid string, data []string, asig []string, pks []string ) {

判断当前的区块时间是否在expiration之内；

根据taskid获取from；

验证链下的预言机leader节点的签名是否正确

调用目的区块链的应用合约执行状态修改；

触发跨链回调事件；

}

跨域预言机接收到跨链回调事件之后，调用源区块链的代理合约进行跨链回调

func Callback(taskid string, data []string, asig []string, pks []string ) {

判断当前的区块时间是否在expiration之内；

根据taskid获取from；

验证链下的预言机leader节点的签名是否正确

回调data给应用合约；

应用合约回调给from；

将当前的跨域共享请求从队列中删除；

}

计算共享

计算共享的假设条件是，节点的计算资源有限，需要将计算任务外包出去，共享的过程包括发布计算任务、链下运算、链上回调。但是，外包可能产生的问题包括，怎么验证外包的结果？我们将采用联邦学习的方式进行计算外包。在该过程中，可能出现两种情况，第一是，客户端节点作恶，可结合区块链+信誉机制解决；第二是，聚合节点作恶，可采用阈值签名解决。上述作恶情况在许多论文中已有解决方案。我们的关注重点是，真正实现计算资源的共享。

节点将链下计算需要的模型model、数据dataset、通信轮数round、本地迭代次数epoch等写入应用合约，应用合约调用代理合约，代理合约将请求写入队列后触发计算共享事件。

计算预言机监听到该事件后，转发给网络中的有能力的节点，这部分节点计算之后，将参数所在的地址、参数摘要、正确率等进行阈值签名之后回调给leader节点，leader节点回调到代理合约中；

代理合约验证签名之后，调用应用合约回调计算结果（参数所在地址、参数摘要、正确率）给应用合约；将当前的计算共享请求从队列中删除。

系统验证

展示合约的单元测试和功能测试

应用场景

项目团队

核心团队

顾问团队

合规团队

项目基础

结束语