# 夜幕(Darkness)

I am a flyfire drift in the dark ----flyfire

声明1:本文档中技术仅限于正义和合法用途，如若采用文中技术进行违法犯罪活动，本人概不负责。

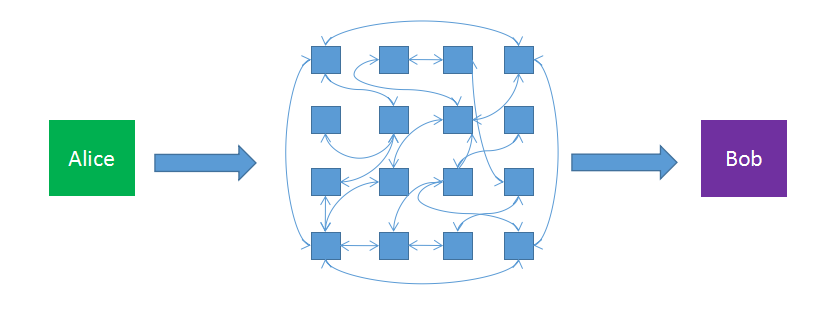
声明2:本文档中区块链技术基于Bismuth加密货币区块链进行构建，详情：www.bismuth.top

# 信条(Creed)

1. 自由(Freedom)
2. 平等(Equality)
3. 简洁即美(Simple is beautiful)
4. 不违反法律法规

# 夜幕的结构

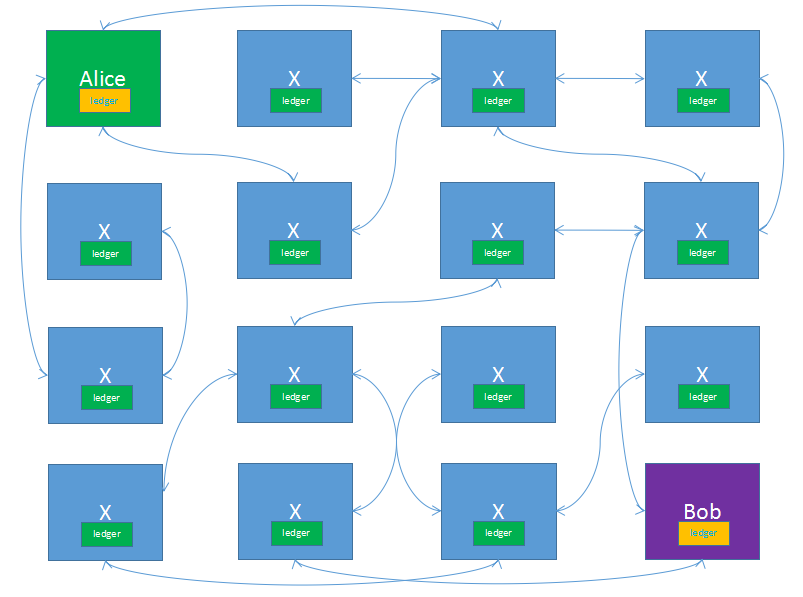
## 2.1 传统网络



P1: 传统网络

传统网络中数据借助路由机制在网络空间传输。

## 2.2 区块链网络



P2:区块链网络

区块链网络借助传统网络层构建，数据采用分布式存储，采用加密学算法进行数据核验，保证数据一致性和准确性。存在两个问题，一是分布式数据库一般只记录转账信息，且随着交易次数的增加数据库占用空间也会增加。二是网络同步时间问题，Alice在本地记录了一笔转账，同步到Bob的本地过程中数据的同步时间不确定，经过了多少层网络也不确定。

## 2.3 夜幕的网络

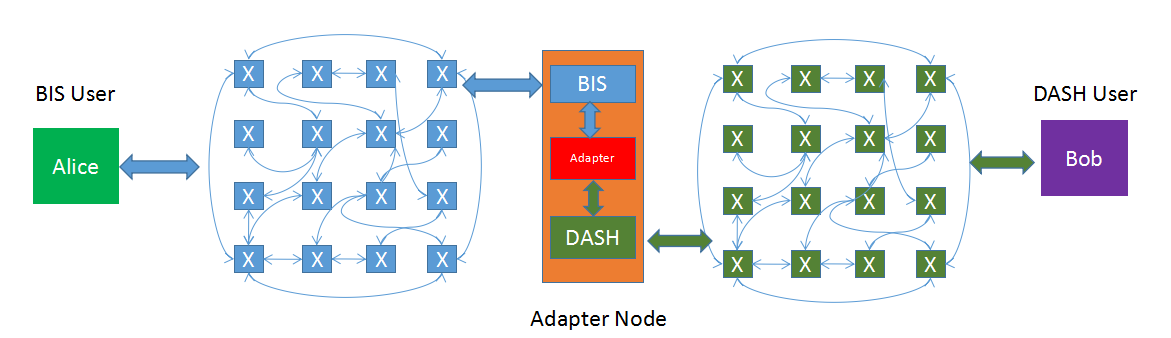


P3:基于区块链的暗网网络

Alice首先转账给Bob，转账信息中的openfield地段中设置”get ticket”字段（相当于买票），Bob收到转账后给Alice进行转账（给Alice进行找零），其中设置openfield地段设置为“ticket：IP\_list”，IP地址可以是一个或者多个。然后Alice从找零信息中获取出主节点的信息，然后向主节点注册，然后发消息给主节点告知主节点，我有消息要给Bob。然后主节点存储消息，与此同时Bob向主节点注册并获取自己的消息，这样Bob就收到了Alice发送的消息。如果Alice的消息过大，可以进行消息拆分操作，然后通过不同的主节点进行发送。

通过上述机制，可以减少ledger.db的大小，同时Alice和Bob之间的网络层次减少了。结合了区块链技术和传统网络系统的优点，且端到端进行加密，消息不可以被窃取。

## 2.4 夜幕的延伸



P4：夜幕的延伸网络

当夜幕在一个加密货币中形成后，接下来就是下一步，夜幕的延伸，借助Adapter Node将加密货币间的通信进行转换，这样不同的区块链就可以链接成一个巨大的夜幕网络。Adapter节点承担了汇率转换和信息转换的作用。这个依赖于具体的加密货币的实现方式，以及信息的加密方式。因为夜幕是端到端的加密，所以Adapter节点的功能不复杂。Adpater节点的具体实现以夜幕的发展为前提，在此不进行深入分析。

# 交互流程

主节点：在区块链中存在至关重要作用的节点，必须是公网IP，且必须稳定，主节点可以按照服务量获得一定的报酬。

其他节点：在区块链中不定时开启的节点或者不稳定的节点，如个人的钱包或者个人节点，可以没有公网IP。

## 3.1 主节点的流程

1. 启动监听，等待Alice或者Bob注册
2. Alice进行注册
3. 主节点生成一个挑战码，可以是随机数
4. 主节点查询Alice在ledger.db中的公钥，并用此公钥对挑战码进行加密，然后发送给Alice
5. 主节点接收Alice返回的挑战码验证信息，采用自己的私钥解密加密的挑战码，如果与发送的挑战码一致，则认为Alice注册成功，然后把结果发送给Alice
6. 如果Alice注册成功，则监听Alice的发送的其他消息，如果Alice注册失败，则发送失败结果并断开连接，流程结束
7. Alice如果发送过来的消息是‘sendmsg’则接受Alice的消息，消息中包含发送的地址、消息的sequence信息，消息的内容，其中消息的sequence和消息内容都是加密的。Sequence中格式如下：

消息分片的index/消息的总长度/消息的UUID

如果消息存储成功，则将成功结果发送给Alice，否则发送失败结果，然后断开连接，会话结束。

1. 第6步中如果接收到的是‘getmsg’消息，则把Alice的所有本地存储的消息都发送给Alice，然后接收响应，如果成功则断开连接，关闭会话。
2. 其他命令功能暂时还未添加，后续优化和补充。主要包含“查询Bob主节点信息”，“查询主节点状态的信息”等

## 3.2 非主节点的流程

**Alice侧的流程：**

1. Alice启动后，首先查询Bob转给自己的转账记录，如果存在ticket，则可以直接连接主节点，进行与Bob的消息交互，如果不存在ticket，则需要向Bob进行转账购买ticket。其中买票的记录中openfield字段为“get\_ticket”，Bob返回的转账（找零）信息中携带“ticket：IP\_list”。此处的IP\_list可以是一个，也可以是多个，建议为多个，可靠性考虑。
2. Alice获取到Bob使用的主节点信息后，连接主节点，发送“register”请求，按照主节点中描述的流程进行注册
3. 然后发送“sendmsg”命令给主节点，然后发送Bob的地址给主节点，告知主节点我要向Bob发送消息，同时从ledger.db中获取Bob的publick\_key对消息内容进行加密。然后按照大小或者其他规则对加密后的消息进行分片。每个分片的sequence格式如下：

消息分片的index/消息的总长度/消息的UUID

分片的sequence也需要使用Bob的publick\_key进行加密

因为Alice同时连接了多个主节点，所以不同的分片可以随机的通过多个节点进行发送。

1. 接收主节点返回的响应，如果成功，则断开连接，关闭会话。如果失败则换其他主节点进行发送，直到成功为止。

**Bob侧的流程：**

1. Bob启动后，首先查询谁给我发送了get\_ticket的转账，查看是否我给了他们门票，以及门票中的主节点信息是什么，然后连接主节点。
2. Bob按照服务端的流程向服务端进行注册
3. 然后发送“getmsg”命令给主节点，然后接收主节点发送消息，接收成功后返回成功，如果失败，再次执行2、3步骤
4. 从各个主节点接收到消息后进行消息的拼装和解码操作。根据每个消息的sequence进行拼装，因为每个消息都存在一个UUID，所以只要消息没有丢失则可以完整的组装成功。

## 3.3 异常场景

主节点存在的问题：

消息积压，连接数达到限制，消息发送超时，连接超时。

非主节点存在的问题：

连接主节点失败，Bob的所有主节点都连接失败，接收消息失败。

目前对异常场景的分析还在持续进行中。

## 3.4 常见问题

1、主节点如何获利？

目前Bismuth还没有主节点，但是后续会存在主节点计划，主节点可以通过服务量获得相应的报酬。因为每笔转账都会存在一定的费用，这些费用种的一部分可以贴补主节点的服务费。

1. 传送的消息是否安全？

完全安全，因为是端到端进行加密的，除了Bob的私钥，没有人可以解读消息内容。

1. 门票的价格？

门票的价格完全由被访问者制定，如果票价过低，则可以拒绝访问，被访问者退换访问者的转账，因为转账费的关系，这个值可能小于访问者起初给的票价，这部分损失访问者和被访问者都会承担一部分。所以只有明确知道被访问者的票价才敢购买，当然土豪请随意。

# 应用场景

## 4.1 项目的初衷

因为网上各种很黄很暴力的内容经常弹出，作为人父，我不希望我的小孩儿过早接触这些不健康的内容，如果采用夜幕这种架构，只有买了票才能上网查看内容，可以解决这个问题

首先，如果需要浏览其他网站则需要买票，这个密码一定在掌握在大人手中

其次，买票的网站上内容是可控的，不存在不健康的内容。如果存在，可以进行本地屏蔽，在夜幕的架构中后台会设计一个禁止访问列表。

我的初衷很简单，只是想净化网络上的垃圾信息，还网络一片净土。

## 4.2 夜幕适用的场景

**网站：**

这个是我的初衷

借助夜幕这种架构完全可以在任意节点搭建网站，且网站内容全部加密，而且是端到端加密。在这种架构下搜索引擎的作用很小，如果要构建搜索引擎，则高额的门票费用（门票费用由网站提供者制定）会让搜索引擎无法维持。其次搜索引擎不能保证访问的内容不是被针对修改过的，因为是端到端加密的，所以你看到的内容可能与我看到的内容是不一致的。导致搜索引擎的准确度降低。基于上述，搜索引擎基本不可用。

**聊天功能：**

这个功能虽然在Bismuth的转账中可以借助openfield字段实现，但是每笔转账都存在手续费，造成聊天费用成本过高。且这样会增加Bismuth的ledger.db的大小，对加密货币的同步不利。其次，加密货币的转账时间也不确定，因为不知道什么时候会同步到本地节点。基于上述问题，采用夜幕系统可以解决上述问题。

**物联网和云化应用：**

物联网是一个很火的话题，采用夜幕架构可以很方便的组织自己的公有云，只需要很少的投资即可，当然夜幕架构也可以用在私有云上。因为这种设计消息内容是通过主节点传递的，且消息传递中Alice与Bob中间只隔了一层主节点，所以速度上会快很多。

云化过程中有很多事网络配置相关的，但是基于夜幕架构后，IP可以随便更改，只要应用的地址和私钥存在则就是以前的节点，不用再去更改IP等配置文件，因为作者目前是从事云运维行业，深知里面的辛酸，采用这种架构对运维是一种很大的简便。

**其他应用场景：**

夜幕架构说白了提供了一种高层的消息传递机制，基于这种机制现有的网站、各种云服务都可以很轻松的移植过来，且移植后的安全性有质的飞跃，毕竟加密货币的算法已经存在了将近10年之久。

因为本人对AI需要的数据不是很了解，所以对于夜幕架构在AI上的应用场景不是很了解，如果谁了解可以进行补充。

# 未来规划(roadmap)

1. 2017年12月份（编写demo进行验证）-----已经完成，可以在

<https://github.com/flyfire100/Darkness>

中进行查看

1. 2018年3月份基于Bismuth完成初步框架，提供API接口，并公布应用开发手册
2. 2018年6月份基于夜幕架构开发适配的聊天程序，主要为安卓手机客户端
3. 2018年12月份基于夜幕架构开发适配的浏览器
4. 2019年集成Fask架构到夜幕框架中，提供网站部署能力，增加使用的便捷性
5. 2020年开始着手研究Apdater Node的实现方式

# 捐助信息

BTC: 1GuoUm1mqd8pwXRee22BkATnEMtpioRiRV

ETH: 0xef56e1429b1ff7d1885e4056e7da3477d84c0b13

BIS: f69b7621d402f4ca17a679adf692b98130003019d968c643700f625e