以患者为中心的细粒度访问控制，通过双区块链实现安全的电子医疗记录共享

1. Learn

1.1 电子病历（Electronic Medical Records, EMRs）

- 方便存储、共享能力
- 电子病历共享——医生可从不同医院访问患者的历史EMRs。有助于医生做出更精确的诊断，并帮助研究人员开发新药或疫苗

1.2 中间人攻击（Man-in-the-Middle Attack，MITM 攻击）

中间人攻击是一种网络安全攻击，攻击者在**通信的两端**之间插入自己，从而能够监视、篡改或劫持通信流量。这种攻击通常在用户和目标服务器之间进行，攻击者能够获取敏感信息、窃取凭据，或者干扰通信内容。中间人攻击通常包括以下步骤：

1. 拦截通信：攻击者在通信路径上插入自己，可能是通过劫持公共Wi-Fi网络、DNS欺骗、ARP 欺骗等方式。
2. 监视流量：一旦攻击者成功插入中间，他们就可以监视通信双方之间的数据流量，获取敏感信息，如登录凭据、信用卡信息等。
3. 篡改数据：攻击者可以修改传输的数据，导致通信双方接收到被篡改的信息，这可能会引发误解、混淆，甚至是恶意操作。
4. 劫持连接：攻击者可能冒充通信双方之一，与另一方建立连接。这使得他们能够操纵通信并传递虚假信息。

为了防范中间人攻击，可以采取以下措施：

1. 使用加密通信：通过使用SSL/TLS等加密协议，可以确保通信内容在传输过程中是加密的，攻击者无法轻易获取明文信息。
2. 验证证书：在使用HTTPS等加密连接时，确保浏览器或应用程序验证服务器的数字证书，以防止受到伪造证书的攻击。
3. 公共网络谨慎使用：避免在公共Wi-Fi网络上传输敏感信息，因为这些网络容易受到中间人攻击。
4. 使用虚拟专用网络（VPN）：通过使用VPN可以加密你的互联网连接，提供额外的安全保障。
5. 定期更改密码：定期更改账户密码，即使信息被攻击者获取，他们也不能长时间使用。

1.3 双线性映射-建立认证密钥协议

什么是群

一、定义

定义1 设G是定义了一个二元运算+的集合，如果这个运算满足下列性质：

（1）封闭性——如果a和b都属于G，则a+b也属于G。

（2）结合律——对于G中的任意元素a、b和c，都有（a+b）+c=a+（b+c）成立。

（3）单位元——G中存在元素e，对于G中任意元素a，都有a+e=e+a=a成立。

（4）逆元——对于G中任意元素a，G中都存在元素a'，使得a+a'=a'+a=e成立。G就叫作一个群，记为（G，+）。

什么是群

（G，+）

如果这里的运算+是加法运算，则称G为加法群；如果这里的运算+是乘法运算，则称G为乘法群。

如果一个群中的元素是有限的，则称这个群是一个有限群；否则称这个群是一个无限群。有限群中元素的个数称为群的阶。

例：集合 {0，1} 关于xor运算是群，阶为2

封闭性：0 xor 1 = 1属于该群

结合律：（0 xor 1）xor 0 = 1 = 0 xor（1 xor 0）

单位元为0：0 xor 0 = 0，0 xor 1 = 1

逆元：0 xor 0 = 0，1 xor 1 = 0

又如：自然数集合N= {1，2，3...} 对于通常的加法封闭且满足结合律，但不存在左单位元和左逆元，因此对于加法不是群。

如果群（G，+）中的运算+还满足交换律，即对G中的任意元素a和b，都有a+b=b+a成立，则称G为一个交换群，例如整数关于加法的运算（Z，+）就为交换群。

在群中定义求幂运算为重复使用群中的运算，如$a^4 = a + a + a + a$。

规定$a^0 = e$为单位元。如果一个群的所有元素都是a的幂a^k，则称这个群是一个循环群，这里的k是整数。a也被称为这个群的生成元。

例：整数加法群是一个循环群，1是生成元，每一个元素都是1的幂，如：

$4=1^4=1+1+1+1$

$-3=1^{-3}=(-1)+(-1)+(-1)$

而且规定$0=1^0$，即0为0个1相加。

（注：定义中的“+”并不代表具体的加法，而是抽象的加法——代表一种代数运算）

定义2 给定群G中元素a，称满足$a^i=e$的最小正整数i为元素a的阶。

二、群的基本性质

（1）左逆元同时也是右逆元，即对于a，$b\in G$，$b+a=e$，则$a+b=e$。

（2）左单位元同时也是右单位元，即如果对于所有的$a\in G$有$ea=e$，则对于所有的$a\in G$也有$ae=e$。

（3）单位元是唯一的。

（4）逆元是唯一的。

双线性映射

1. 双线性映射

设 V , W 和 X 是在同一个基础域F上的三个向量空间。双线性映射是函数：

$$\mathrm{B}:\mathrm{V}\times\mathrm{W}\to\mathrm{X}$$

使得对于任何 W 中 w，映射

$$\mathrm{v}\mapsto\mathrm{B}(\mathrm{v},\mathrm{w})$$

是从 V 到 X 的线性映射，并且对于任何 V 中的 v，映射

$$\mathrm{w}\mapsto\mathrm{B}(\mathrm{v},\mathrm{w})$$

是从 W 到 X 的线性映射。

换句话说，如果保持双线性映射的第一个参数固定，并留下第二个参数可变，结果的是线性算子，如果保持第二个参数固定也是类似的。

双线性映射

抽象意义的双线性映射描述如下：

设G_1、G_2都是阶为p的循环群，p是素数。如果映射$e: G_1 \times G_1 \rightarrow G_2$满足以下性质：

（1）双线性性。

对于任意a，$b \in Z_p$和R，$S \in G_1$，有$e(R^a, S^b) = e(R, S)^{ab}$；

（2）非退化性。

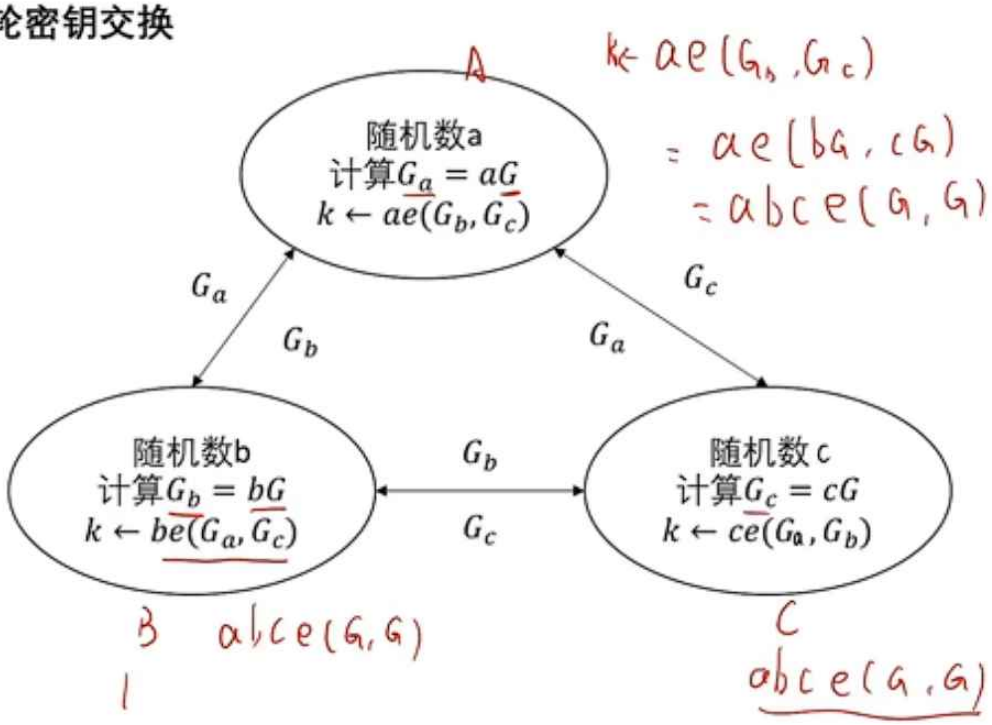
存在R，$S \in G_1$，使得$e(R, S) \neq 1_{G2}$。这里1_{G2}代表G_2群的单位元；

（3）可计算性。

存在有效的算法对任意的R，$S \in G_1$，计算$e(R, S)$的值。

那么称e是一个双线性映射。

三方一轮密钥交换



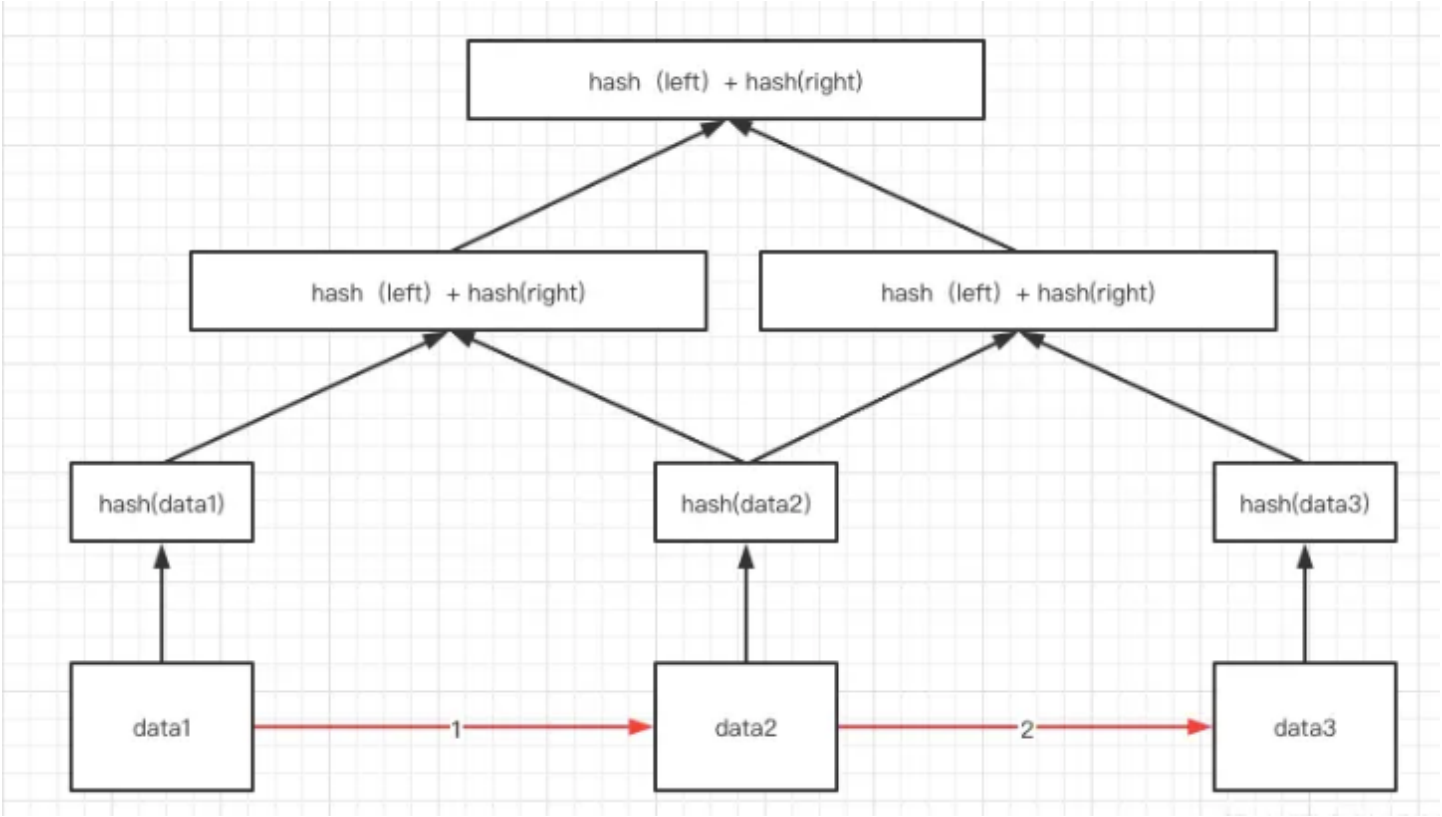
- 双线性映射函数是一种特殊的映射函数，可以将**两个向量映射到一个标量值**。
- 用途：
  a. 加密和解密：将明文和密钥映射到一个标量值，实现对数据的加密和解密操作。
  b. 签名和验证：通过将消息和私钥映射到一个标量值，可以生成数字签名。而将消息、公钥和签名映射到一个标量值，可以验证数字签名的有效性。
  c. 身份验证：通过将用户的身份信息和私钥映射到一个标量值，可以生成身份证明。而将用户的身份信息、公钥和身份证明映射到一个标量值，可以验证用户的身份。

d. 零知识证明：通过将证明者的证明和验证者的挑战映射到一个标量值，可以实现零知识证明的过程。

1.4 Merkle tree

Merkle tree

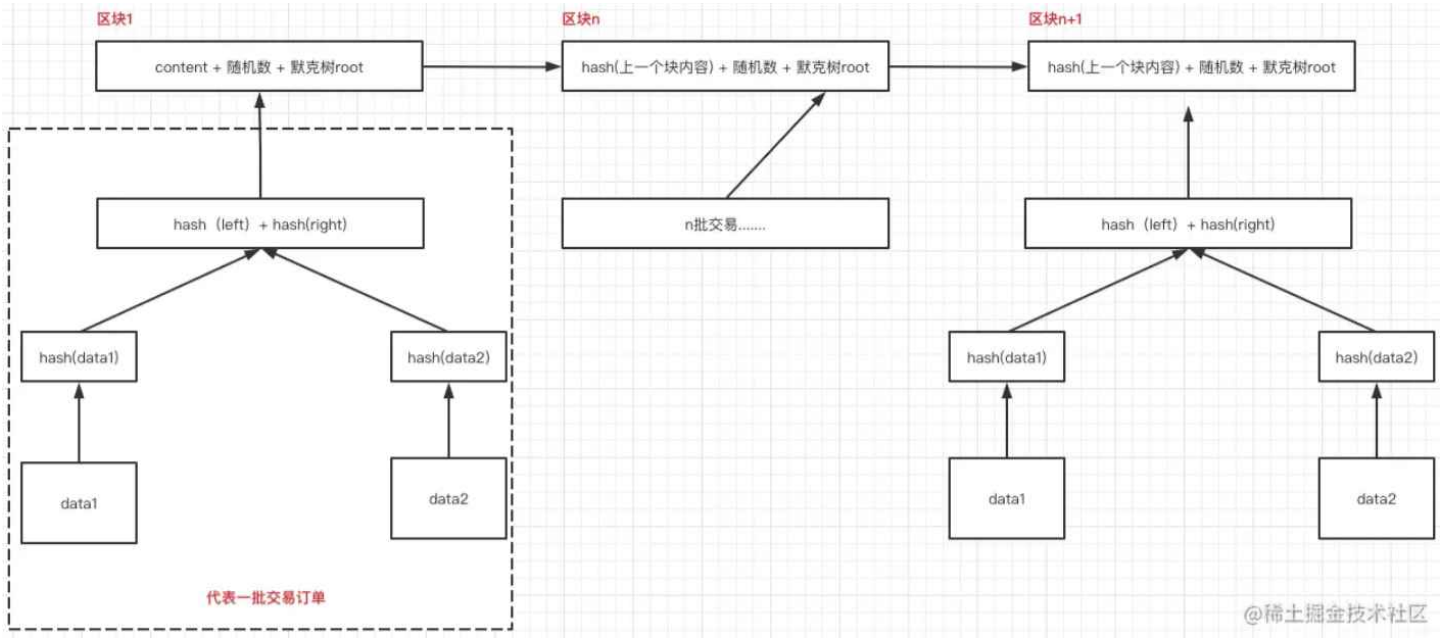
下图是一个简单的Merkle tree，可以看到除最底层的数据外，其他节点都是左右两个子节点的hash值组成。（注：红线代表左右顺序）



Hash链表

传统链表是使用下一个节点地址作为指向（上一个节点知道下一个节点地址，因此能指向下一个节点），但是Merkle tree中的链表和区块链中的链表一样，使用上一个节点的hash值作为指向（下一个节点知道上一个节点的hash值？），如图：

Hash链表+Merkle tree=防篡改



假设我们更改了左边虚框内那一批已经存在的事务数据，例如data1，那区块1的**Merkle tree** root值就一定会改变，区块1的hash值也一定会变，这种变化会产生新的链，当发现这条新链在区块1后的所有区块值与各个节点原本记录的值不一致，就会认为有人修改了链上的旧数据。

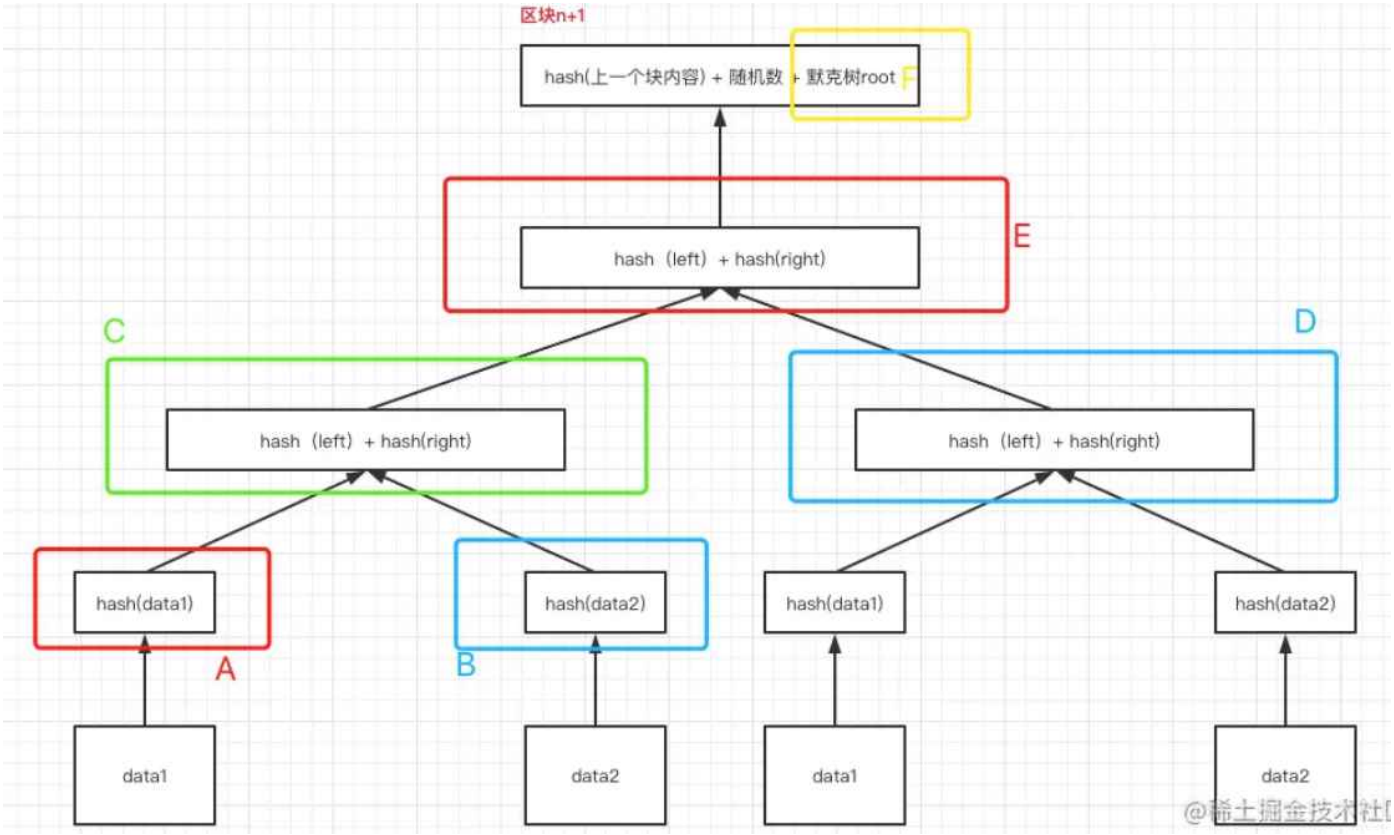
而且我们使用的是hash值作为指向，只要大家手上的最后一个值没问题，在回溯时必然无法回溯到被篡改的数据，甚至回溯对比后还可以知道哪里发生了篡改。

既然无法指向我们篡改的数据，那我们把后面的所有区块以及其数据也篡改了行不行？可以的，但是区块有无数个，而且并不是简单的遍历修改本地数据就ok了，还需要所有节点的共识，你能黑光所有的节点，让他们都直接放弃手中的数据，认可你这新的链吗？

所以在对账时，就很容易知道账目是否正确，由于是直接比较hash值，使用**Merkle tree**去判断内容是否被篡改是很快的！

判断某个事务是否被记录（是否存在）

你怎么保证你手中的数据和链上一致？怎么证明你的数据在链上呢？



例子：你在银行存了50万，银行怎么证明它给你存了50万呢？

假设要判断红色框data1的数据是否是A=“我在银行存了50万”

1. 首先向信任节点获取蓝色框和黄色框的值。
2. 然后假定我们要判断的数据的已经被记为A，假定值A与B进行hash,得到C
3. 将C与D进行hash，得到E
4. 判断E是否等于 F
    - 等于，说明data1存储的数据确实是A，事务已被记录
    - 不等于，说明data1存储的数据不是A，事务没有被记录。

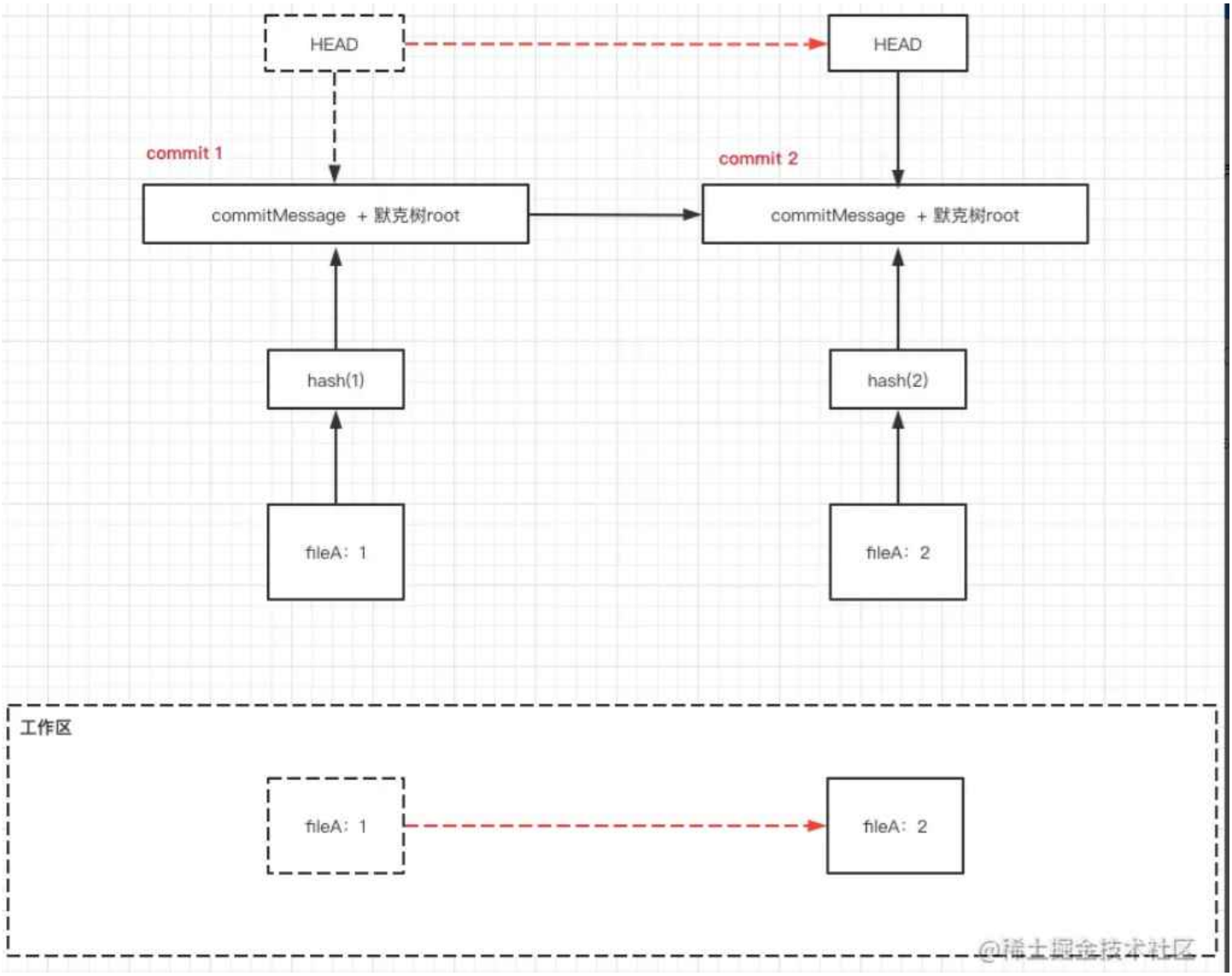
常见应用

git

我们切换commit时，git是怎么实现不同commit文件数量和文件内容的切换的？

git会记录所有版本的文件，例如文件a在第一个commit中内容是1，第二次commit中内容是2，此时git本地仓库中会分别有：内容为1的文件a，内容为2的文件a。

git中每一个commit就相当于一个区块，这个区块有对应的Merkle tree，而Merkle tree的hash值又指向了对应的文件，所以切换一个commit其实就相当于将当前区块切换，如下图



分布式数据存储的数据校验

我们将成千上万个文件存在互联网上的任意服务器，任何一个能上网的终端都可以作为我们的存储器。假设我们为了保证性能，不通过中介服务器，直接p2p连接，并且不校验这些存储器的身份。那如何保证我们从这些不受信任的存储器中下载的数据，是我们存入时的样子（没有被篡改）？

步骤：

1. 这些任意的服务器都要拥有其存储文件的Merkle tree。
2. 终端下载这个服务器中存储的Merkle tree，向值得信任的服务器取得这个Merkle tree对应区块的值，计算并判断Merkle tree顶部的hash值是否等于区块记录的值，等于说明这个服务器记录的Merkle tree没有问题。

1.5 Hotstuff共识机制

https://blog.csdn.net/ganzr/article/details/110879463

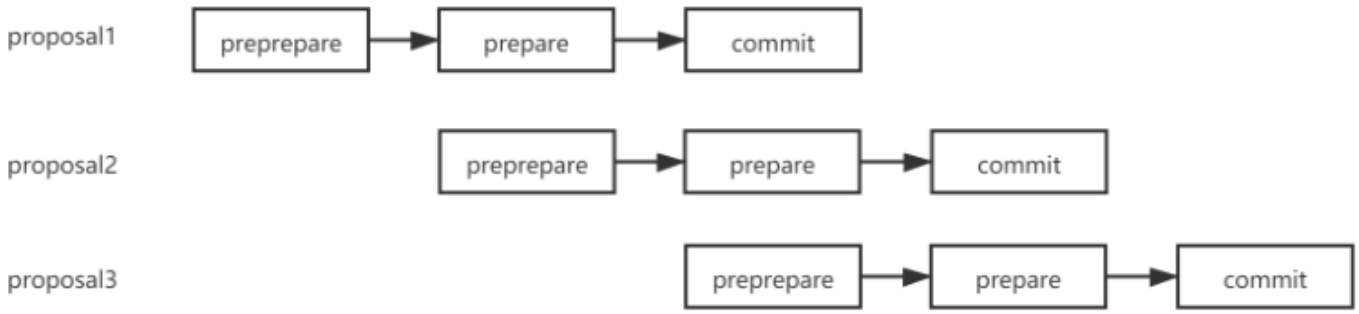
算法复杂度为O(n)

优化点：

1. 将**节点广播投票**改成**leader节点收集投票然后再广播**，从而将 $O(n^2)$ 的复杂度降低为 $O(n)$

> 在PBFT中，由于每个节点都会广播自己的投票并收集其他人的投票（因为有拜占庭节点的存在，节点只相信自己所获取到的关于某个节点的投票信息），导致每一个节点都做了大量重复的工作。
>
> HotStuff只需要一个节点负责收集，将收集到的结果广播给每一个节点，避免其他节点重复收集

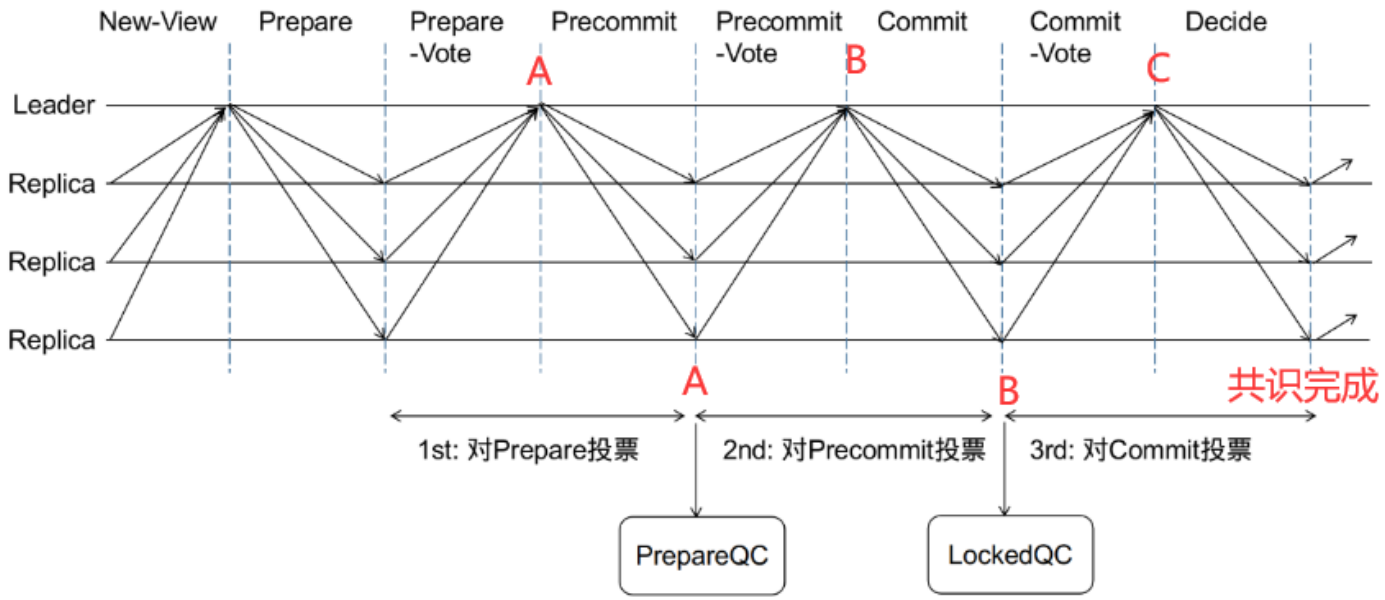
2. 每一个轮的投票除了是对本轮的请求的共识，**也是对上一轮请求的共识**。

> pbft预准备、准备、提交**三个阶段**的内容都有**收集投票**：
>
> - preprepare：收集leader的提议
> - prepare：收集所有节点对于该提议的投票信息；
> - commit：收集所有节点对于prepare阶段中“自身节点收集到2/3的其他节点的反对/赞成投票”的信息；
>
> 将提议的不同阶段重合，实现**并行处理**提议的投票
>
> 
>

3. **每一轮投票更换一次leader节点**，尽可能减少leader出错时更换leader所需要的时间。

> 在pbft中，有一个专门的**view-change**阶段来更换出问题的leader，但是这个view-change实质上就是一个特殊的proposal，这个proposal不需要leader来发出，而是节点本身在判断当前leader出问题时，就会直接广播一个针对这个特殊的proposal的投票。那么这里的leader更换就需要走至少一轮完整的共识。**耗费了一个共识的时间**
>
> 在HotStuff中，为了减少更换leader所需要的走的流程，主动在每次发送完proposal之后就更换leader。

非流水线HotStuff算法流程：

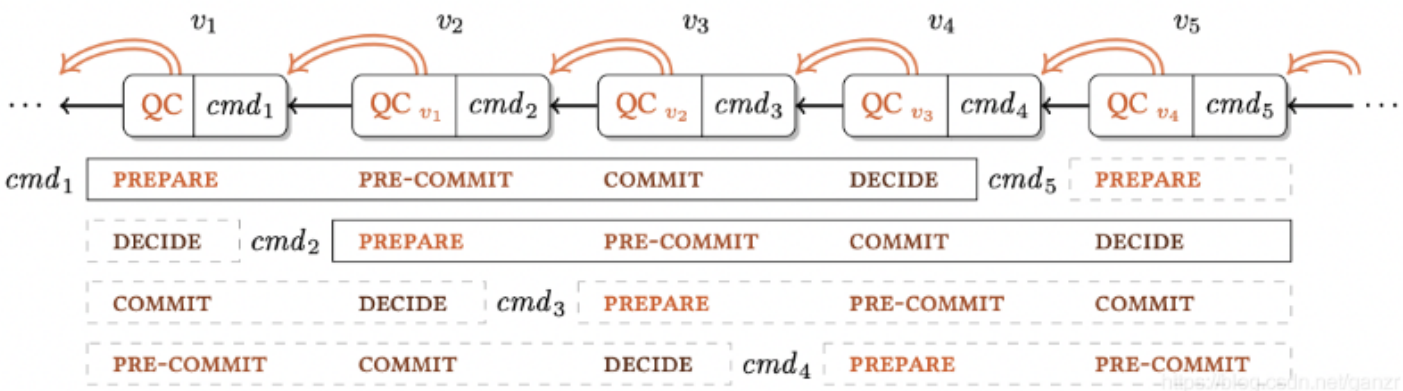


leader做收集投票的事情：

1. 收集至少2/3节点的投票，即leader节点**发生了**事件A——收到至少2/3个投票，此时leader节点把这个时间的证据保留下来，广播给其他节点，那么其他节点也就相当于发生了事件A；但是还是不知道其他节点是否有**接收到**leader广播的事件A
2. 此时**接收到**事件A的节点会广播一个投票给leader；leader会收集这些投票，即leader发生了事件B——至少2/3的节点收到至少2/3个投票，同理，将这个事件B广播给其他节点，其他的节点在收到时也相当于发生了事件B，但是还是不知道其他节点是否有**收到**leader广播的事件B
3. 跟第2步一样，收到事件B的节点也会广播一个投票给leader，leader收集，此时leader节点上发生了事件C，同理，将这个事件广播给其他节点，其他节点受到后，就确认了共识已经达成。

流水线HotStuff算法流程

在非流水线的hotstuff中，节点的步骤有4个，分别是pre-prepare，prepare，pre-commit，commit，在**每一阶段中都会有一次广播**，因此，可以将**不同的proposal的不同的阶段重合**，然后复用每一轮的广播来完成这些重叠的proposal的阶段的变换。



QC即上述的事件发生的“证据”——至少2/3节点的投票

1. 每一个cmd（即proposal）在prepare被提出，经历后续三个阶段后达成共识；

2. 每一个QC，既是所指向的cmd达到下一个阶段的证明，也是上一个cmd的的证明，同时还是上上个cmd的证明，且分别表示不同的cmd到达了不同阶段的证明。

总结

- **leader完成**本来每个节点都需要重复做的事情——收集投票
- 将整个共识的阶段抽象成**流水线**模型，每一个QC同时证明三个cmd状态
- 每一个leader在广播完一个（QC，cmd）之后就会**更换另一个leader**

2. Why?

问题

- 无法保证**安全性**：EMR共享存在未经授权访问、数据泄漏风险——无法维护患者病历隐私
- 无法保证**不可篡改性**：不完整、或被篡改的EMR会误导医生的治疗
- 无法保证**隐私性**：患者需要对EMR进行细粒度的访问控制，防止医生访问不相关的EMR

目前基于云、结合多种密码算法的方案问题

提高了效率，确保了数据安全

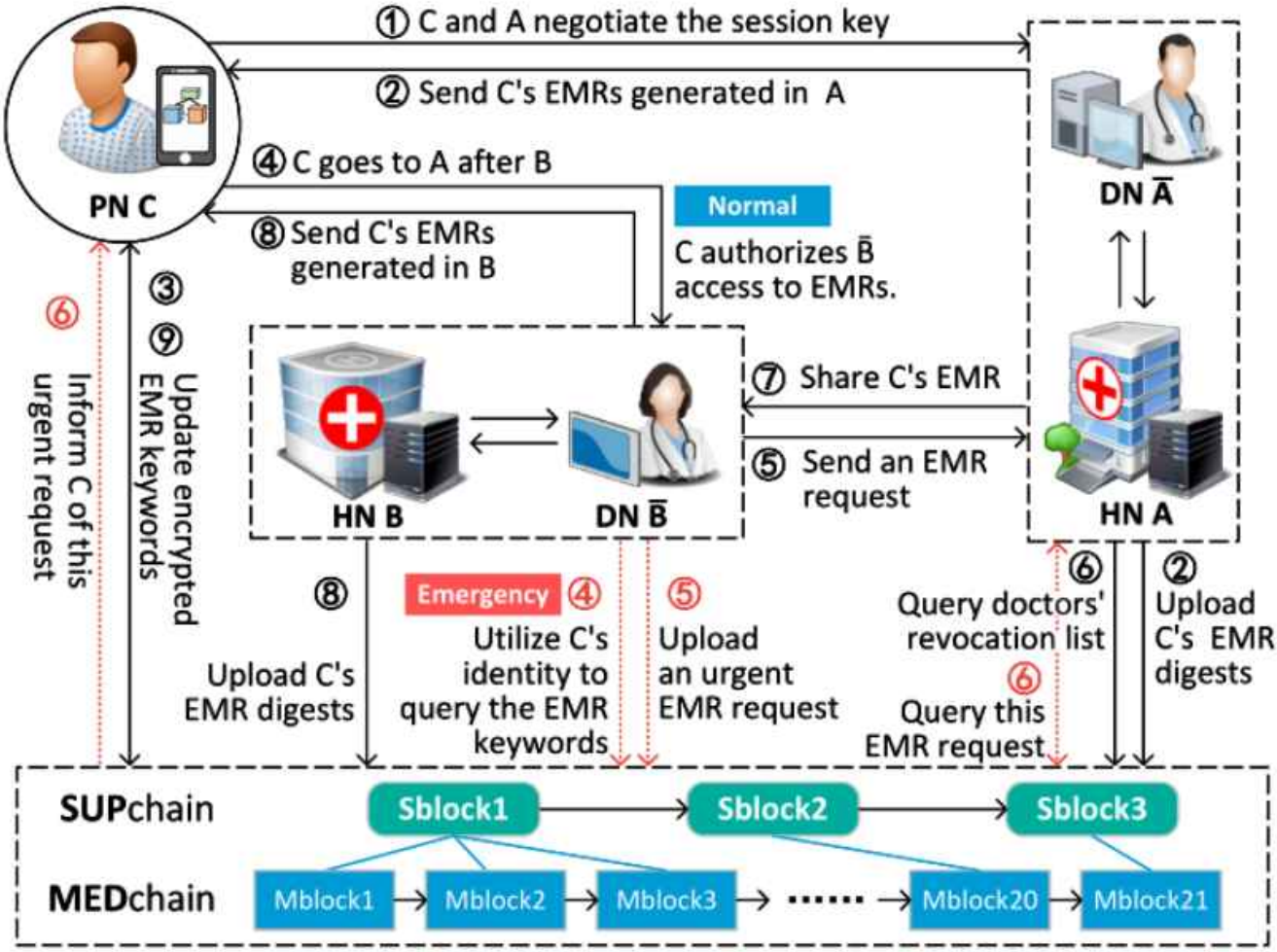
- 容易出现单点故障
- 无法在医院之间建立信任

目前基于区块链的方案问题

实现医院间的信任、跨医院的EMR不变存储、允许每个实体验证EMR的真实性(无篡改)、完整性(无损失)

- 恶意的医生可能滥用病人的授权，查阅不相关的电子病历，损害病人的隐私
- 昏迷患者没有能力授权医生访问自己先前的EMR
- 对区块链共识机制的某些攻击可能会篡改上传的事务，从而破坏实体之间的信任。

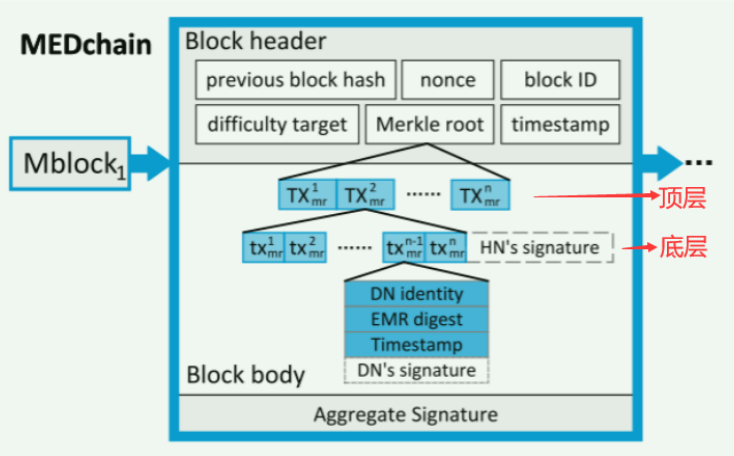
3. What?

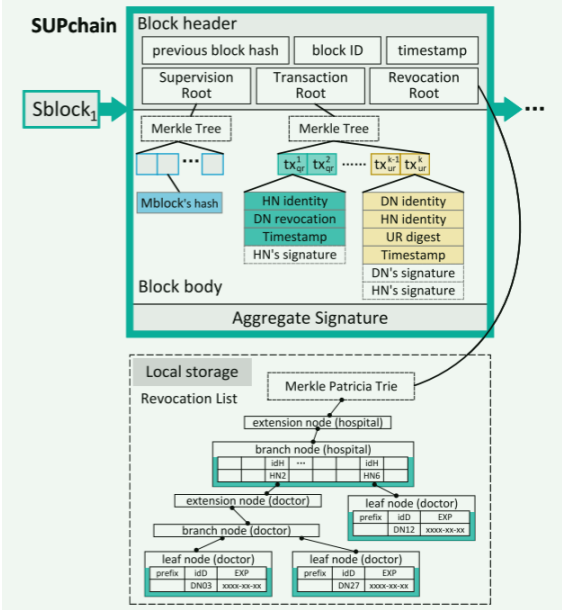


3.1 双区块链架构

MED医疗区块链（公有链）：

- 使用**工作量证明机制**（以建立实体间的信任关系）生成新的医疗区块Mblocks
- 存储：时间不敏感的EMR相关事务（tx_{mr}）
- Mblock不存储事务的单个签名，而是保留最终的聚合签名——节省MED存储空间
- 其他实体通过验证聚合签名，从而轻松验证Mblock中所有事务的真实性——降低计算开销
- Mblock中的**事务**记录在两层结构中：
  - HN从DNs收集事务，然后继续验证、聚合DNs各自的签名——形成底层
  - 矿工节点从HNs获得这些事务，对HNs的签名进行进一步验证和**第二次聚合**——形成顶层





SUP监管区块链（联盟链）：

- 使用**Hotstuff共识机制**（以快速生成）生成监管区块（Sblock）
- 存储：时间敏感的事务（如**紧急EMR请求** tx_{ur} 、**医生许可操作请求** tx_{qr} -监管医生资格、**Mblock的哈希**-使患者昏迷时可紧急访问EMR）、医生撤销列表Merkle树的根节点
- 专门在**医院之间**运行
- 采用Merkle Patricia Trie，MRT 在每个HN的本地存储中维护一致的医生撤销列表。该列表可随着医生证书的改变，动态扩展或收缩
- 实现聚合签名方案——提高事务验证的效率
- 监督公有链-医疗区块链，以防止自私挖矿攻击
- 监管身份过期的医生

3.2 三类型节点

- 医院节点HN：
  - 同时参与MED链、SUP链，**维护MED链、SUP链**
  - 职责：
    - 存储EMR
    - 验证请求真实性
    - 参与合法医生节点DN的协商，以共享EMR
    - 聚合事务签名，以节省区块链存储资源
    - 维护医生撤销列表
    - 通过SUP链监管MED链

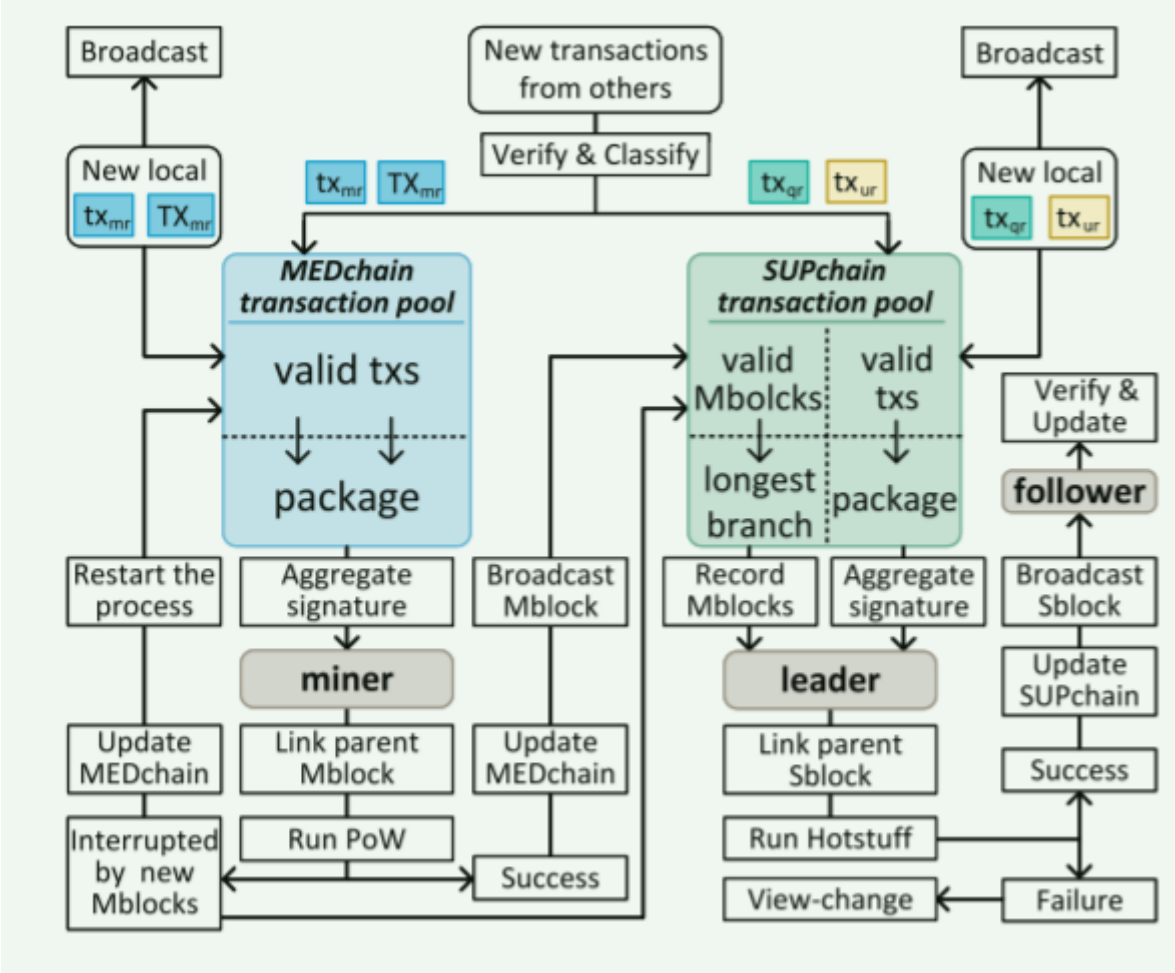
- 医生节点DN：
  - 同时参与MED链、SUP链
  - 视为HN的代理，并持有其颁发的许可证
  - 职责：
    - 代表HN向其他机构请求EMR（通常使用TAKA协议获得EMR）
    - 生成EMR
    - 在MED链上发布EMR相关事务
    - 在紧急情况下，在MED链上查询EMR**关键字**，并在SUP链发出紧急请求，以实现可追溯、安全的EMR共享
- 患者节点PN：
  - 仅维护MED链，可以从SUP链读取数据
  - 作为EMR访问控制器
  - 作为MED链中的**轻量级节点**，可自由加入或退出系统
  - 职责：
    - 基于TAKA协议，授权DN访问存储在其他HN中的特定EMR
    - 利用生物特征信息加密EMR关键字，存储在MED链上

3.3 时间敏感性事务

- 基于时间及时性的事务分类，有助于在紧急情况下对紧急电子病历做出迅速反应
- 时间敏感：存储在SUP链中
  - 紧急EMR请求 tx_{ur}
  - 医生资格撤销 tx_{qr}
- 时间不敏感：存储在MED链中
  - EMR相关事务 tx_{mr}

3.4 实体与链的关系

- 如果一个实体只维护MED链，则必须进行MED链事务池相关操作
- 如果一个实体只维护SUP链，则需要向SUP链事务池添加新Mblock、和添加事务
- 如果一个实体同时维护MED、SUP，则除以上操作外，还必须识别、分类来自其他实体的事务



3.5 TAKA协议

基于身份的三方认证密钥协议（Tripartite Authentication Key Agreement, TAKA）：现患者对自己EMR的细粒度访问控制

- TAKA为不同EMR生成唯一的会话密钥——防止医生访问不相关的EMR
- TAKA可以扩展为一次共享多个EMR

3.6 以患者为中心

让病人对他们EMR访问有细颗粒度的控制

- 患者精确控制其他人对自己的EMR访问权限
- 患者授权特定医生访问特定的EMR——根据个人偏好
- 即使患者处于无意识状态，紧急EMR共享也需要患者的生物识别信息

4. How?

4.1 相关工作

4.1.1 基于区块链的EMR共享

- Yu [3]设计了一个基于联盟区块链的EMR共享平台，并引入假名机制保护患者隐私

- Amofa [16]提出了一种区块链架构，将智能合约与预先指定的策略相匹配，以保证EMR共享的安全性。
- Zhuang [17]允许患者控制授权请求者列表，隐藏他们不想共享的EMR。特别地，如果患者无意识，则以患者为中心的医疗保健系统可以安全地共享EMR
- Xu [18]创新性地提出了双区块链结构，将患者健康数据和医生诊断分开管理，减少了通信和计算开销。
- Yazdinejad [19]提出了一种基于区块链的医院网络的有效分散身份验证方案。但该方案要求患者将其公钥和私钥发送给参与转移的医院，这增加了医疗数据泄露的风险。
- Chen [20]提出了医院之间的安全EMR共享系统，以避免医疗资源的浪费，使用智能合约的程序化授权来确保EMR的安全性。
- Liu [21]提出了一种基于医院私有区块链的EMR共享方案，医生能够在保护患者隐私的同时存储或访问EMR。该方案允许实体生成自己的公钥和私钥，以确保数据的机密性。然而，参与者和系统管理之间的频繁通信可能导致单点故障漏洞。
- Sun [22]采用基于属性的加密（ABE）来实现EMR的细粒度访问控制，其中患者和医生共同制定EMR访问策略。医生对数据请求者进行身份验证，降低患者操作难度。然而，该方案依赖于安全信道来发送密钥，这降低了可用性。
- an [23]提出了一种基于区块链的EMR共享方案，基于密文策略属性加密（CP-ABE），可以跟踪并直接撤销满足属性集的恶意用户。
- Wu [24]提出了一种具有DiffieHellman密钥交换的细粒度EMR访问控制解决方案。它引入区块链来提高EMR存储和共享的效率。然而，密钥交换机制不能认证身份并且容易受到中间人攻击。

上述相关工作缺陷：

- 没有以患者为中心的细粒度EMR访问控制
- 某些方案需要部署访问策略、与分发密钥，从而限制了患者调整EMR请求者许可的灵活性

4.1.2 认证密钥协议

- Zhang [11]提出了云与用户之间的匿名认证，以防止用户的身份在认证过程中被暴露。
- Feng [26]提出了一种基于双方计算的协作认证协议，恶意实体无法使用该协议通过仅使用其中一个部分私钥对医疗服务器的验证。

这两种方案都不是基于区块链架构，都有传统集中式EMR共享的弊端。

本文设计了一个基于身份的三方认证密钥协议：

- 允许医生在患者的授权下，与另一家医院安全地协商特定的会话密钥以进行EMR共享。
- 使用两个**反向哈希链**来生成加密因子。再将加密因子集成到会话密钥中，实现患者对EMR精确的访问控制

4.2 系统框架

见上

4.3 威胁模型

- 恶意医生会滥用、伪造患者的授权，访问不相关的EMRs
- 每个实体都有动机操纵双区块链的数据以谋私利
- 不到三分之一的医院会恶意串通

4.4 安全目标

- 授权安全性：患者生成访问许可，供医生访问患者在其他医院存储的EMRs。攻击者应该无法伪造患者的授权书，无法通过医院验证
- 数据保密性：未经授权的实体不得访问EMR，或不得解密任何EMR密文
- 身份验证密钥协商：医生与医院之间的身份认证应该
  - 仅在患者的授权下完成
  - 在协商会话密钥时完成
- 前向和后向保密：过去和未来，共享EMR应该一直保持安全，即使参与者不知道被泄漏的长期密钥和当前会话密钥
- 抵抗自私挖矿攻击：在MED链中，恶意矿工可能会使用高计算能力设备篡改MED链的原始数据，本系统应该可以抵御自私挖矿攻击

4.5 方案细节

4.5.1 方案概览

- 当医生需要查看患者在其他医院的EMRs时，患者给该医生授权
  - 如果患者清醒，使用**TAKA方案**直接给医生授权，MED链用于验证EMR完整性
  - 如果患者昏迷，医生**在SUP链发布紧急请求**，以安全访问患者EMR
- 医院管理医生许可证
  - 医院为雇佣的医生发放许可证，并设定有效期
  - 利用**SUP链**维护被撤销医生列表
- SUP链监督MED链，防止发生自私挖矿攻击

4.5.2 系统初始化

全局可信的PKG(私钥生成器)初始化协议所需系统参数：

- 初始化双线性映射参数：
  a. 选择加法群G1和乘法群G2，两者具有**相同素数阶**q的。P是G1的生成元。
  b. 选择一个双线性映射G1 × G1 →G2，设置 $g = \hat{e}(P, P)$ 。

c. 选择随机数 $s \in Z_q^*$ 作为**主私钥**，输出 $P_{pub} = sP$ 作为主公钥，在G1中

- 选择安全哈希函数：
  - H_0：$\{0，1\}^* \to Z_q^*$
  - H_1：$\{0，1\}^* \to G_1^*$
  - H_2：$\{0，1\}^* \times G_1^* \times G_1^* \times G_1^* \to Z_q^*$
  - H_3：$G_2 \to \{0，1\}^*$
  - H_4：$G_1^* \to Z_q^*$
  - H_5：$Z_q^* \to Z_q^*$
  - H_6：$\{0，1\}^* \to \{0，1\}^*$
  - H_7：$\{0，1\}^* \times G_1^* \times G_1^* \to Z_q^*$
  - H_{sk}：$\{0，1\}^* \to \{0，1\}^k$ （sk表示会话密钥，k =|sk|）.
- 发布系统参数：params = $(G_1, G_2, \hat{e}, q, g, P, P_{pub}, H_0, H_1, H_2, H_3, H_4, H_5, H_6, H_7, H_{sk})$

4.5.3 注册

总述：

1. 每个实体通过安全信道向PKG发送注册请求
2. PKG为每个实体提取两个密钥：一个用于签名消息；一个用于协商会话密钥
3. 每个实体在双区块链中注册一个系统账户

详细步骤：

假设实体U进行登记

1. U向PKG发出登记请求：U选择随机值 $x_U \in Z_q^*$ ， ID_U 为U已有的唯一标识
   a. $L_{U1} = x_U P$ ，在G1中
   b. $Q_U = H_0(ID_U)$ ， $\hat{Q}_U = H_1(ID_U)$ ，将两者存于本地（H_0：$\{0，1\}^* \to Z_q^*$ ，H_1：$\{0，1\}^* \to G_1^*$）

   发送 ID_U和L_{U1} 给PKG——登记请求
2. PKG提取密钥，响应U：选择随机数 $y_U \in z_p^*$
   a. $L_{U2} = y_U P$ ，在G1中
   b. $L_U = L_{U1} + L_{U2}$ ，由群的封闭性可知，在G1中
   c. $f = H_7(ID, P_{pub}, L_U)$ ， （H_7：$\{0，1\}^* \times G_1^* \times G_1^* \to Z_q^*$）
   d. $d_U = (y_U + sf) mod\ q$ ，s是PKG的主私钥，结果在 z_q^* 中
   e. $\hat{Q}_U = H_1(ID_U), \hat{S}_U = s\hat{Q}_U$ ，两者都在G1中

发送 $(L_U, d_U, \hat{S}_U)$ 给U

3. U安全存放 $x_U, L_U, d_U, \hat{S}_U$
   - 可以使用 (x_U, L_U, d_U) 签名消息
   - 可以使用 Q_U 授权或验证授权
   - 可以使用 $(\hat{Q}_U, \hat{S}_U)$ 与其他实体协商会话密钥
   - 协议采用Gentry[30]的聚合签名方案，来对双区块链上的事务进行签名
4. 医生实体在PKG注册后，还要在医院注册

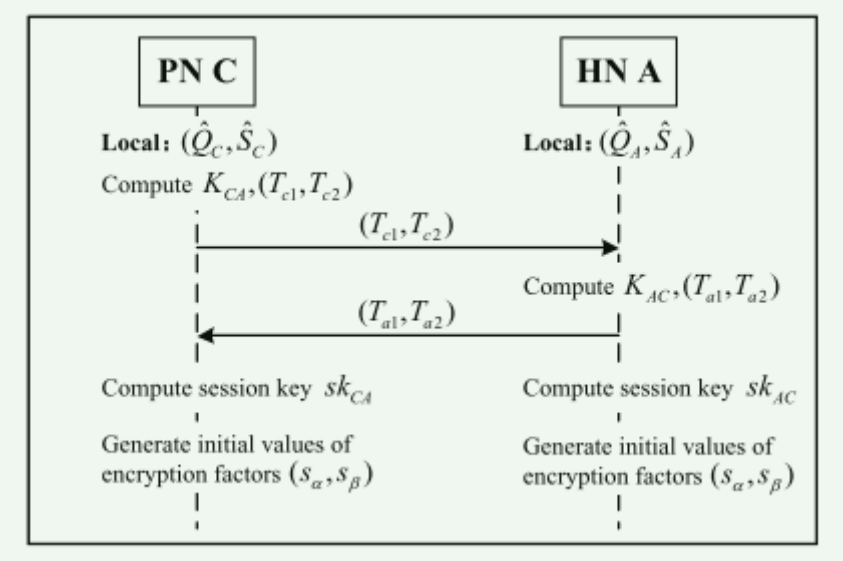
   当医院雇佣医生时，医院在 $\rho = (ID_{HN}, ID_{DN}, EXP)$ 中指定医生许可证的到期日期EXP，使用DU[32]的方案签名ρ，并签发给医生

4.5.4 EMR管理

使用HNs管理PNs的EMR——HNs总是在线的

协商密钥、生成初始因子

当患者（PN C）第一次访问医院（HN A）时，C和A协商会话密钥，并生成两个初始因子 S_α和S_β（初始因子生成加密因子，用于C的EMR细粒度访问控制）



详细步骤：

1. C
   a. 计算 $K_{CA} = e(\hat{S}_C, \hat{Q}_A)$
   b. 选择两个随机数 $c_1, c_2 \in Z_q^*$，计算 $T_{c1} = c_1 P, T_{c2} = c_2 P$
   c. 发送 T_{c1}, T_{c2} 给A
2. A
   a. 计算 $K_{AC} = e(\hat{S}_A, \hat{Q}_C)$

b. 选择两个随机数 $a_1, a_2 \in Z_q^*$ ，计算 $T_{a1} = a_1 P, T_{a2} = a_2 P$

c. 发送 T_{a1}, T_{a2} 给C

d. 生成对称会话密钥 $sk_{AC} = H_{sk}(ID_C||ID_A||a_1 T_{c1}||a_2 T_{c2}||K_{AC}||T_{c1}||T_{c2}||T_{a1}||T_{a2})$

3. C

a. 生成对称会话密钥 $sk_{CA} = H_{sk}(ID_C||ID_A||c_1 T_{a1}||c_2 T_{a2}||K_{CA}||T_{c1}||T_{c2}||T_{a1}||T_{a2})$

4. A C

根据 $T_{a1}, T_{a2}, T_{c1}, T_{c2}$ ，A C都产生两个相同的初始因子

$s_\alpha = c_1 T_{a2} = a_2 T_{c1} = a_2 c_1 P, s_\beta = c_2 T_{a1} = a_1 T_{c2} = a_1 c_2 P$

EMR和相关事务 tx_{mr} 的生成、上链

1. 生成：就诊医生（DN $\bar{A}$）对患者进行诊断后，生成C的EMR、包含C的EMR哈希值的医疗记录事务 tx_{mr}

$tx_{mr} = \{\lambda_0, ID_{\bar{A}}, hEMR, ts_1, htxM, \sigma_{\bar{A}}\}$

- λ_0 ： 事务类型
- $ID_{\bar{A}}$ ： DN唯一标识
- $hEMR$ ： EMR哈希值
- ts_1 ： 时间戳
- $htxM = H(\lambda_0, ID_{\bar{A}}, hEMR, ts_1)$ ，前四项哈希值
- $\sigma_{\bar{A}} = Sign_{\bar{A}}(htxM)$ ： $\bar{A}$ 对 $htxM$ 的签名

2. 发送：医生 $\bar{A}$ 将EMR和 tx_{mr} 发送到该医院A

3. 上链：医院A**存储**C的EMR和 tx_{mr} ，并**聚合事务签名**，将这些事务**传到MED链**。具体步骤如下：

a. 校验有效性：A使用 tx_{mr} 的 $hEMR$ 验证EMR的完整性，使用 $(htxM, \sigma_{\bar{A}})$ 验证 tx_{mr} 的有效性。之后，A根据每个EMR的 $hEMR$ 生成顺序递增的EMR标识符

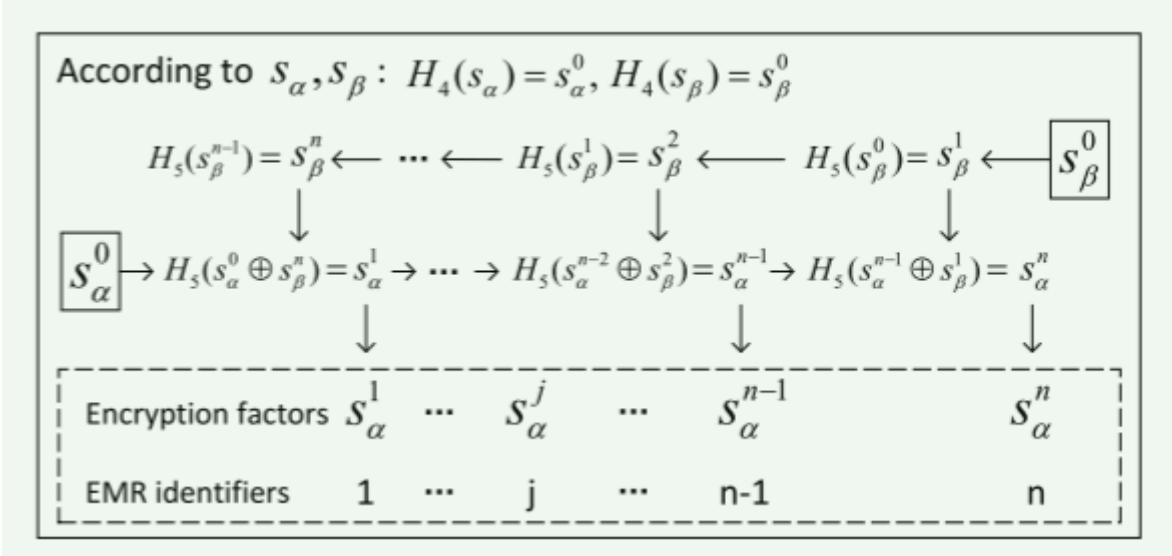
b. 初始因子生成加密因子：A根据有效EMR的个数，利用 S_α和S_β 生成相等数量的加密因子，**加密因子与每个EMR一对一对应**。

i. A计算 $s_\alpha^0 = H_4(s_\alpha)$ 、 $s_\beta^0 = H_4(s_\beta)$ ， H_4： $G_1^* \to Z_q^*$

ii. 假设C的EMR数量为n，A计算hash链表： $s_\beta^1 = H_5(s_\beta^0)$ ， $s_\beta^2 = H_5(s_\beta^1) \cdots\cdots$ ， $s_\beta^n = H_5(s_\beta^{n-1})$ 。 H_5： $Z_q^* \to Z_q^*$

iii. A将这些值反向插入另一个hash链表中： $s_\alpha^1 = H_5(s_\alpha^0 \oplus s_\beta^n)$ ， $s_\alpha^2 = H_5(s_\alpha^1 \oplus s_\beta^{n-1})$ $\cdots\cdots$ ， $s_\alpha^n = H_5(s_\alpha^{n-1} \oplus s_\beta^1)$

iv. 最后，A得到加密因子集合 $\{s_\alpha^1, s_\alpha^2, \cdots, s_\alpha^j, \cdots, s_\alpha^n\}$ ，其中j是EMR标识符



c. 将事务组合成Merkle tree：基于 $\{htxM_1, \cdots, htxM_n\}$，A将n个事务 $\{tx_{mr}^1, \cdots, tx_{mr}^n\}$ 组合到Merkle tree中，并计算Merkle tree root值rootM，生成关于rootM的事务 TX_{mr}

$$TX_{mr} = \{\lambda_0, ID_A, rootM, ts_2, hTXM, \sigma_A\}$$

- ID_A：HN唯一标识

d. 聚合签名：A将事务集合 $T = \{tx_{mr}^1, \cdots, tx_{mr}^n, TX_{mr}\}$ 中的**签名**聚合成一个聚合签名 σ_{agg}，再移除T中所有签名以得到事务 T/σ

e. 发送给信息：

i. To 矿工：A发出(T/σ, σ_{agg})

ii. To C：A用 sk_{AC} 加密EMR和相应的htxM，并将它们发送给C

f. 矿工上链：

i. 每个矿工接收到(T/σ, σ_{agg})后，通过聚合签名 σ_{agg} 验证事务集合 T/σ 的有效性

ii. 基于 $\{htxM_1, \cdots, htxM_n\}$，矿工生成Merkle root，并进一步将 $\{\sigma_{agg}^1, \cdots, \sigma_{agg}^n\}$ 聚合成一个聚合签名

iii. 通过PoW共识机制将新的Mblock上传到MED链上。

g. C加密EMR生成EEE：

i. 解密A传来的数据

ii. 对于每个EMR，使用相关htxM来访问**MED链上**的相关 tx_{mr}，tx_{mr} 中的hEMR用于检查EMR的完整性

iii. 从EMR中提取关键字W，创建 EMR条目EE

$$EE = \{W||htxM||ID_A||j\}$$

- ID_A：存储EMR的**HN**唯一标识
- j：EMR编号

iv. 使用C自己的生物信息（如：指纹fp）加密EMR条目 $EEE = H_6(fp) \oplus EE$ （H_6：$\{0，1\}^* \to \{0，1\}^*$）

v. 调用智能合约将EEE存储在MED链上

vi. C再执行与A相同的操作以获得相同的加密因子 $\{s_\alpha^1, s_\alpha^2, \cdots, s_\alpha^j, \cdots, s_\alpha^n\}$ ——因为C在协商密钥阶段有和A相同的初始因子

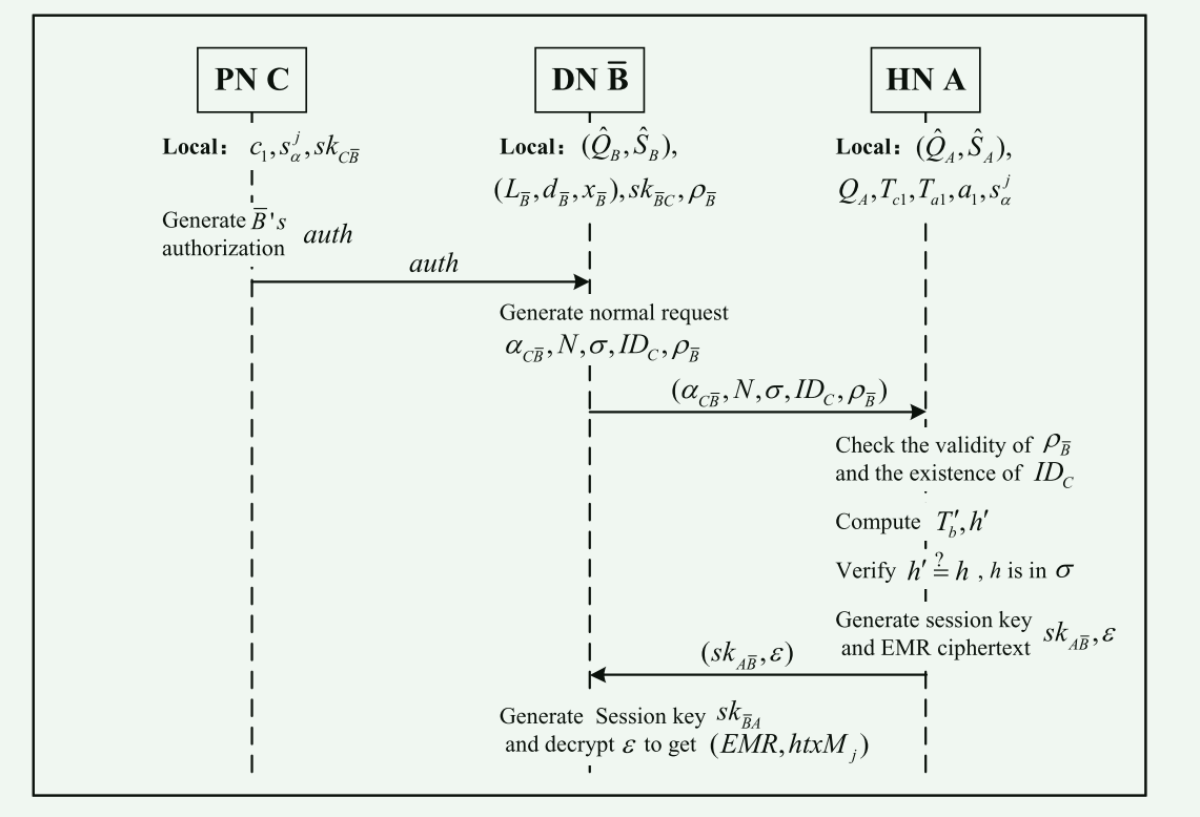
4.5.5 EMR共享

正常情况EMR共享：患者授权医生访问EMR

紧急情况EMR共享：昏迷患者不能直接授权医生，医生需要利用双区块链搜索、访问患者EMR

正常情况EMR共享

总览：医生 $\bar{B}$ 先向患者C获取授权，再拿着授权向医院A获取EMR，再从MED链找到对应hEMR验证EMR完整性



1. C给予 $\bar{B}$ 所需访问EMR的**标识符j和加密因子**——即获取EMR的授权
   a. 协商会话密钥： $\bar{B}$ 和C协商认证的会话密钥 $sk_{C\bar{B}} = sk_{\bar{B}C}$ ，（协商步骤间2.4.4.1）
   b. 计算 α_C ：C计算 $Q_A = H_0(ID_A)$ ， $Q_{\bar{B}} = H_0(ID_{\bar{B}})$ ， $\alpha_C = \dfrac{1}{c_1 + Q_A + Q_{\bar{B}}} P$ ， $\alpha_C \in G_1$
   c. 发送授权消息给医生：加密并发送 $(\alpha_C, c_1 s_\alpha^j P, j)$ 给 $\bar{B}$ ，其中 s_α^j 为 标识符为j的EMR 所对应的加密因子）
2. $\bar{B}$ 通过C给予的授权信息生成EMR请求，向A请求EMR

a. 使用授权的 α_C 计算 $\alpha_{C\bar{B}}$： $\bar{B}$ 选择随机数 $b \in Z_q^*$，计算 $T_b = bP, g_b = g^b, \alpha_{C\bar{B}} = b\alpha_C$，$\alpha_{C\bar{B}} \in G_1$，其中双线性映射 $g = \hat{e}(P, P)$

b. 生成请求：**正常请求** $\theta = (ID_C||j)$，**加密请求** $N = H_3(g_b) \oplus \theta$，（$H_3$：$G_2 \rightarrow \{0，1\}^*$

c. 请求签名：根据登记阶段获得的 $(L_{\bar{B}}, d_{\bar{B}}, x_{\bar{B}})$，计算 $h = H_2(\theta||ID_{\bar{B}}||P_{pub}||L_{\bar{B}}||T_b)$，$v = b + h(d_{\bar{B}} + x_{\bar{B}}) mod q$，签名 $\sigma = (L_{\bar{B}}, h, v)$

d. 发送请求给医院：发送 $(\alpha_{C\bar{B}}, N, \sigma, ID_C, \rho_{\bar{B}})$ 给医院A，其中 $\rho_{\bar{B}}$ 是 $\bar{B}$ 的许可

3. 医院A检查医生发出的请求是否合法、被授权

a. 检查医生身份合法性：如果 $\bar{B}$ 不在医生撤销列表中，则认为 $\rho_{\bar{B}}$ 是合法的

b. 检查授权的真实性：如果C的EMR存在，A计算

i. $Q_B = H_0(ID_B)$,

ii. $\alpha'_{CB} = T_{c1} + (Q_A + Q_B)P$, $\alpha'_{CB} \in G_1$

iii. $g'_b = \hat{e}(\alpha'_{CB}, \alpha_{CB})$,

iv. $\theta' = H_3(g'_b) \oplus N$ ——验证量

v. $f' = H_7(ID_B||P_{pub}||L_B)$,

vi. $T'_b = vP - h(L_B + f'P_{pub})$ ——验证量

vii. $h' = H_2(\theta'||ID_B||P_{pub}||L_B||T_B||T'_b)$.

如果等式 $h' = H_2(\theta'||ID_B||P_{pub}||L_B||T_B||T'_b) == h$ 成立，则A可以**确认C授权给了** $\bar{B}$，**且正常请求θ没有被篡改。**

正确性分析：

$$
\begin{aligned}
g'_b &= \hat{e}\left(T_{c1} + Q_A P + Q_{\bar{B}} P, \alpha_{C\bar{B}}\right) \\
&= \hat{e}\left((c_1 + Q_A + Q_{\bar{B}})P, b\frac{1}{c_1 + Q_A + Q_{\bar{B}}}P\right) \\
&= \hat{e}\left(P, P\right)^{(c_1+Q_A+Q_{\bar{B}})\cdot\frac{b}{c_1+Q_A+Q_{\bar{B}}}} \\
&= \hat{e}\left(P, P\right)^{b} = g_b \\
\theta' &= H_3\left(g'_b\right) \oplus N = H_3\left(g'_b\right) \oplus \left(H_3(g_b) \oplus \theta\right) \\
&= H_3\left(g_b\right) \oplus \left(H_3(g_b) \oplus \theta\right) = \theta
\end{aligned}
$$

$$
\begin{aligned}
T'_b &= vP - h\left(L_{\bar{B}} + f'P_{pub}\right) \\
&= \left(b + h(d_{\bar{B}} + x_{\bar{B}})\right)P - h\left(L_{\bar{B}} + f'P_{pub}\right) \\
&= \left(b + h(d_{\bar{B}} + x_{\bar{B}})\right)P - h\left(x_{\bar{B}}P + y_{\bar{B}}P + f'P_{pub}\right) \\
&= \left(b + h(d_{\bar{B}} + x_{\bar{B}})\right)P - h\left(x_{\bar{B}}P + y_{\bar{B}}P + f'sP\right) \\
&= \left(b + h(d_{\bar{B}} + x_{\bar{B}}) - h(x_{\bar{B}} + y_{\bar{B}} + f's)\right)P \\
&= \left(b + h((y_{\bar{B}} + fs) + x_{\bar{B}}) - h(x_{\bar{B}} + y_{\bar{B}} + f's)\right)P \\
&= bP = T_b \\
h' &= H_2\left(\theta' \| ID_B \| P_{pub} \| L_B \| T'_b\right) \\
&= H_2\left(\theta \| ID_B \| P_{pub} \| L_B \| T_b\right) = h
\end{aligned}
$$

4. A 根据 θ 中的 EMR 标识符 j，将指定EMR和htxM加密返回给医生 $\bar{B}$

a. $K_{AB-C} = \hat{e}(T_b, c_1 P)^{s_\alpha^j a_1}$

b. $sk_{AB} = H_{sk}(ID_B||ID_A||a_1 T'_b||K_{AB-C}||T_b||T_{a1})$

c. 加密EMR消息：用 sk_{AB} 加密 $(EMR, htxM_j)$，得到ε

d. 返回给医生：发送 (T_{a1}, ε) 给 $\bar{B}$

5. $\bar{B}$ 解密EMR消息，并通过MED链上的hEMR验证其完整性

   a. 计算 $K_{BA-C} = \hat{e}(T_{a1}, c_1 s_\alpha^j P)^b$ ，如果 $K_{BA-C} == K_{AB-C}$ ，则确定 T_{a1} 的完整性

$$
\begin{aligned}
K_{A\bar{B}-C} &= \hat{e}(T_b, c_1 P)^{s_\alpha^j a_1} = \hat{e}(bP, c_1 P)^{s_\alpha^j a_1} \\
&= \hat{e}(P, P)^{bc_1 s_\alpha^j a_1} \\
K_{\bar{B}A-C} &= \hat{e}(T_{a1}, c_1 s_\alpha^j P)^b = \hat{e}(a_1 P, c_1 s_\alpha^j P)^b \\
&= \hat{e}(P, P)^{a_1 c_1 s_\alpha^j b}
\end{aligned}
$$

   b. 计算密钥： $sk_{BA} = H_{sk}(ID_B || ID_A || bT'_{a1} || K_{BA-C} || T_b || T_{a1})$

   c. 解密EMR消息：用 sk_{BA} 解密ε，获得目标EMR和htxM的明文

   d. 从MED链获取hEMR以检验医院发来的EMR的完整性：通过htxM在MED链上找到 tx_{mr} 的 $hEMR$ ，验证EMR完整性

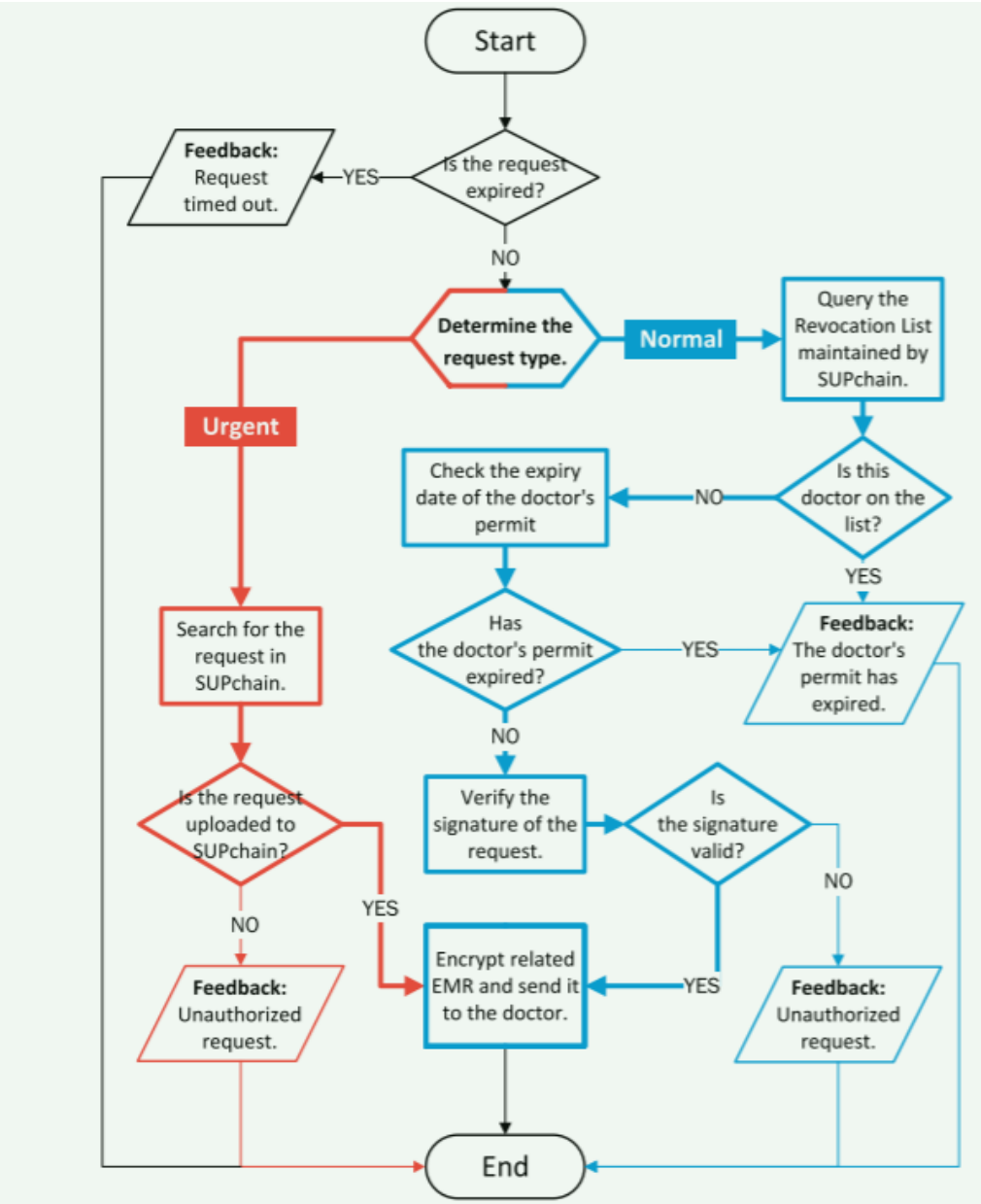


Fig. 8. The process for hospital A to check doctor $\bar{B}$'s request.

协议支持一次共享多个EMR。例如，B和A计划共 享 m标识符为j－1，j+1，j+2的三个EMR。然后将上述协议中的 s_α^j 替换为 （$s_\alpha^{j-1}, s_\alpha^{j+1}, s_\alpha^{j+2}$） ，B和A生成三个EMR的特定会话密钥

紧急情况EMR共享

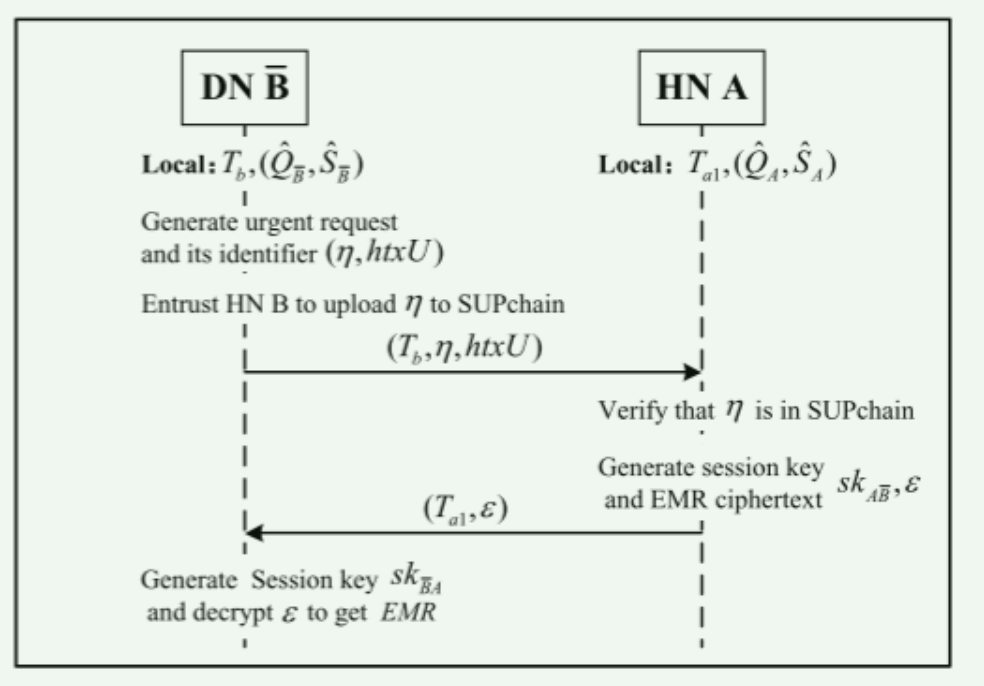
总述：医生 $\bar{B}$ 使用C的生物信息从MED链获取EMR条目EE，再请求所属医院B为医生作安全担保，即医院B在SUP链发出紧急EMR共享请求—— tx_{ur} 上SUP链，医生 $\bar{B}$ 再发送紧急请求给医院A（存有C相关EMR），A检查SUP链是否有记录 tx_{ur} ，检验 $\sigma_{\bar{B}B}$ 有效性，加密相关EMR信息给医生 $\bar{B}$ ，医生用EE的htxM从MED链获取hEMR,验证EMR完整性

治疗后，C访问双区块链可以查看C在昏迷期间的EMR共享历史

1. 医生 $\bar{B}$ 从MED链获取C的EMR条目EE，提前关键字W（后面第三步会使用W从医院A获取所需EMR）：
   a. 搜索EEE： $\bar{B}$ 调用具有 ID_C 和指纹fp的MED链智能合约，来搜索C的加密EMR条目EEE
   b. 自动解密为EE：如果 ID_C 和fp有效，智能合约自动通过 $EEE = H_6(fp) \oplus EE$ 解密得到EE
   c. 提取W： $\bar{B}$ 从 $EE = \{W||htxM||ID_A||j\}$ 中基于关键字W确定存储在A中的C的 EMR_j 是有用的
2. 医生紧急请求**所属医院B**为其做**安全担保**，将 tx_{ur} 存入SUB链，以便下一步向存有EMR的医院A请求EMR
   a. 医生生成紧急请求并发给所属医院：医生 $\bar{B}$ 生成紧急请求 $\eta = (ID_C||j)$ ，和元组$\{\lambda_1, ID_{\bar{B}}, ID_B, hUR, ts_3, htxU, \sigma_{\bar{B}}\}$

   > 事务类型、医生唯一标识、医院唯一标识
   > - $hUR = H(\eta)$：紧急请求的哈希值
   > - $htxU = H(\lambda_1, ID_{\bar{B}}, ID_B, hUR, ts_3)$：事务前五项的哈希值
   > - $\sigma_{\bar{B}} = Sign_{\bar{B}}(\lambda_1, ID_{\bar{B}}, hUR, ts_3)$

   将这些数据发送给医院B
   b. 医院B检查医生 $\bar{B}$ 合法性（撤销列表）
   c. 医院B聚合签名：医院B签署事务 $\sigma_B = Sign_B(htxU)$ 。聚合 σ_B和$\sigma_{\bar{B}}$得到$\sigma_{\bar{B}B}$。
   d. 医院B生成 $tx_{ur} = \{\lambda_1, ID_{\bar{B}}, ID_B, hUR, ts_3, htxU, \sigma_{\bar{B}B}\}$
   e. 医院B将 tx_{ur} 上SUP链：如果 $\sigma_{\bar{B}B}$ 有效，则SUP链通过HotStuff共识将 tx_{ur} 记录在区块中
3. 医生 $\bar{B}$ 紧急请求医院A以获取所需EMR，A根据η从SUP链获取 tx_{ur} ，告知患者紧急请求，计算密钥，将EMR加密发送给医生 $\bar{B}$



a. 医生发送紧急请求给医院A：医生 $\bar{B}$ 选择随机数 $b \in Z_q^*$ ，计算 $T_b = bP$ ，发送 $(T_b, \eta, htxU)$ 给医院A（因为医院A才保存着患者C的完整EMR）

b. 医院A查看SUP链，根据η获取 tx_{ur} ：A收到 $\bar{B}$ 的紧急请求，使用η和htxU调用SUP的智能合约，确认η是否在SUP链中，如果 tx_{ur} 存在，则智能合约返回 tx_{ur} 给A，并告知患者C这个紧急请求。

c. 检查 tx_{ur} 中 $\sigma_{\bar{B}B}$ 的有效性——紧急请求是否是由这个医生发出： $\sigma_{\bar{B}B}$ 有效，则A相信 $\bar{B}$ 是合法的，η有效。

d. A计算与医生的密钥： $K_{A\bar{B}} = \hat{e}(\hat{S}A, \hat{Q}_{\bar{B}}), sk_{A\bar{B}} = H_{sk}(ID_{\bar{B}}||ID_A||a_1T_b||K_{A\bar{B}}||T_b||T_{a1}).$

e. 加密EMR信息：用 sk_{AB} 加密 $(EMR, htxM_j)$ ，得到ε

f. 发送：将 $(T_{a1}$ ， $\varepsilon)$ 发送给医生 $\bar{B}$

4. 医生 $\bar{B}$

a. 计算密钥： $K_{\bar{B}A} = \hat{e}(\hat{S}_{\bar{B}}, \hat{Q}_A),\ sk_{\bar{B}A} = H_{sk}(ID_{\bar{B}}||ID_A||bT_{a1}||K_{\bar{B}A}||T_b||T_{a1}).$

b. 解密EMR消息：使用 $sk_{\bar{B}A}$ 解密ε，获得目标EMR和htxM的明文，

c. 从MED链获取hEMR以检验医院发来的EMR的完整性：使用 $EE = \{W||htxM||ID_A||j\}$ 中的htxM，在MED链上找到 tx_{mr}^j ，验证EMR的完整性、有效性。

4.5.6 监督

SUP链有两个功能：

监督MED链的Mblocks

- 当HN生成或接收到新的有效Mblocks时，HN将其哈希值复制到SUP链事务池中
- 一个Sblock存储一段Mblocks的哈希

- 当前共识的主节点HN从SUP事务池中选择新的Mblock的最长分支，以Merkle树的形式记录在新的Sblock中
- 选择MED链分支的具体规则（由上到下）：
  a. 规则1：新MED链分支的**父区块**必须是上一个Sblock中记录的最新Mblock
  b. 规则2：**最长**分支
  c. 规则3：如果多分支等长，选择M**erkle root哈希值最小**的分支
- 例如：HN将要生成新的Sblock，检查S3
  a. 寻找**父区块是S3中最新Mblock**的MED链分支
    - 情况a的MED链分支父区块是M11
    - 情况b的MED链分支父区块是M9
    - 情况e的MED链分支父区块是M2
  b. 寻找**最长**的MED链分支——情况d最长
  c. d的MED链分支被记录在S4区块中

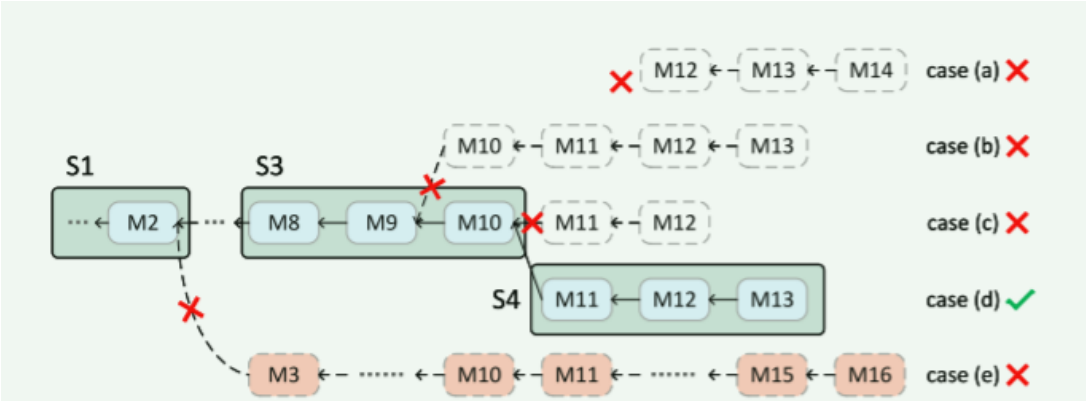


Fig. 10. SUPchain supervise MEDchain.

- 如果生成了Sblock，则说明**所有HN已经确认MED链新分支的有效性**，MED链中记录的EMR可以在SUP监督下，由医疗机构安全地交叉使用

管理过期DN

HN可以**撤销**或**重新授权**DN的许可（在DN的许可自然到期之前）

例如：HN E通过SUP链发出关于医生的资格撤销请求 tx_{qr}

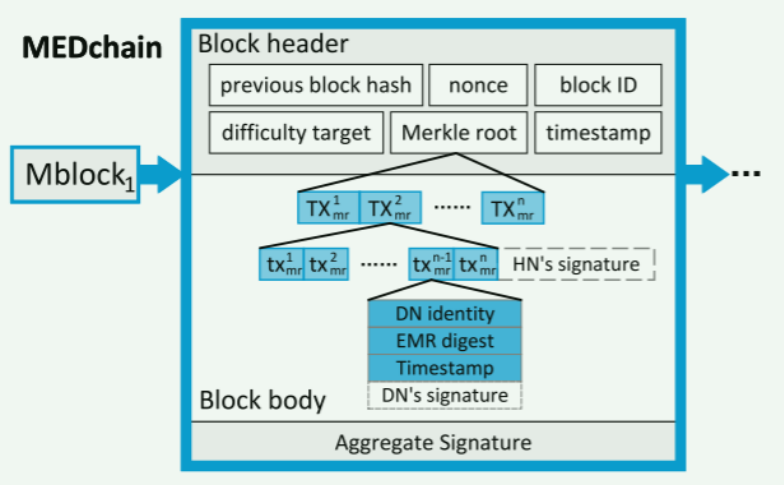
1. 医院E生成资格撤销通知 $QR = \{ID_{\bar{E}}，EXP，AoD\}$

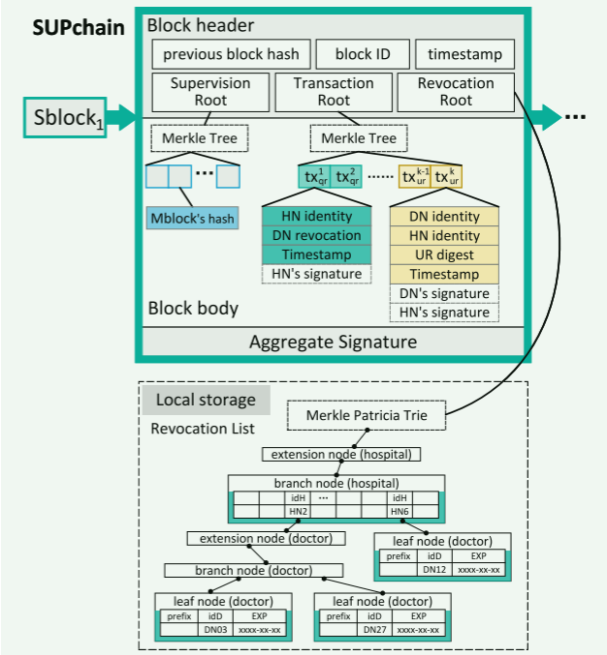
> $ID_{\bar{E}}$：医生的唯一标识
>
> EXP：医生的许可自然到期时间
>
> AoD：添加/删除标记；0-将医生添加到撤销列表；1-将医生从撤销列表删除

2. 医院E生成 $tx_{qr} = \{\lambda_2, ID_E, QR, ts_4, htxQ, \sigma_E\}$

> λ_2：事务类型
>
> ID_E：医院E的唯一标识
>
> $htxQ = H(\lambda_2, ID_E, QR, ts_4)$：$tx_{qr}$ 的标识符
>
> $\sigma_E = SignE\ (htxQ)$ ：医院E的签名

3. 每个HN将撤销列表本地存储在MPT结构中，基于 tx_{qr} 从列表中添加/删除DN——如果**本地撤销列表的root**与**新Sblock中的root**值相同，则表明HN维护与其他HN相同的撤销列表





4.6 取得了什么效果?

- 安全性分析证实，协议满足指定的目标。
- 性能分析表明，该协议具有较高的计算效率和较低的通信开销。
- 安全性分析，以证明我们的协议可以确保数据安全和保护用户隐私。我们还提出了一个详细的证据的安全性的TAKA协议。我们的安全性比较和性能评估表明，我们的方案具有较低的计算开销，略低的通信开销

4.7 收获

1. MED区块链存储 tx_{mr} 而不是整个EMR，既降低了区块链存储负担，又通过htxM、hEMR保证了EMR的分布式安全和隐私保护
2. SUP链通过存储MED链区块的哈希，实现区块链监督另一个区块链
3. 节点本地存储医生撤销列表的Merkle，SUP链只存储Merkle树根值，降低了区块链负担、又能保证撤销列表一致性
4. 使用聚合签名降低存储成本、验证成本

5. 使用Hotstuff共识机制，相较于PBFT共识算法，减少了节点间的通信开销，更适用于大规模节点的区块链网络