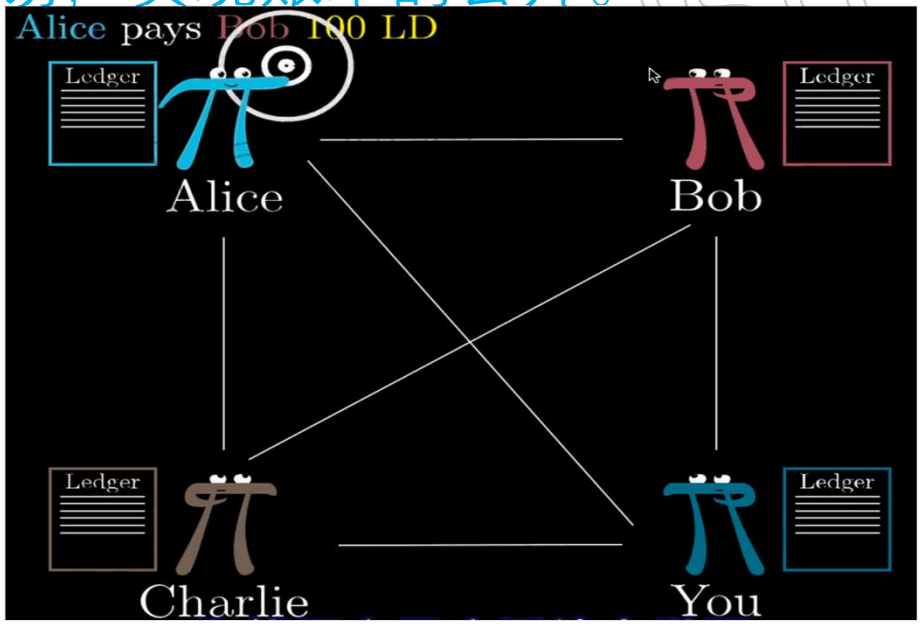
区块链隐私保护

网络空间安全学院 代炜琦

目录

区块链的公开透明与隐私保护

- 比特币：为了保证交易的真实性，比特币广播每一条交易，实现账本的公开。



账户模式

张三获得了12.5 枚比特币。过了几天，他把其中 2.5 枚支付给李四。
又过了几天，他和李四各出资 2.5 比特币凑成 5 比特币付给王五。

为了避免双花，传统系统往往需要较高的维护成本。

UTXO

张三挖到12.5 枚比特币。过了几天，他把其中 2.5 枚支付给李四。又过了几天，他和李四各出资 2.5 比特币凑成 5 比特币付给王五。

缺点：**隐私泄露**

- 交易行为（**张三在出2001块的时候交易了2.5比特币！**）
- 账户余额（**张三还有7.5比特币！**）
- 账户流水（**他和李四还有王五进行了交易！**）

比特币的隐私保护

比特币系统的匿名设置:

不足：**匿名性有限**

- 大多数用户只有一个地址
- 通过交易分析可以得出同一用户不同地址之间存在的联系

[Reid Martin 11] [Barber Boyen Shi Uzun 12] [Ron Shamir 12] [Meiklejohn PJLMVS 13]

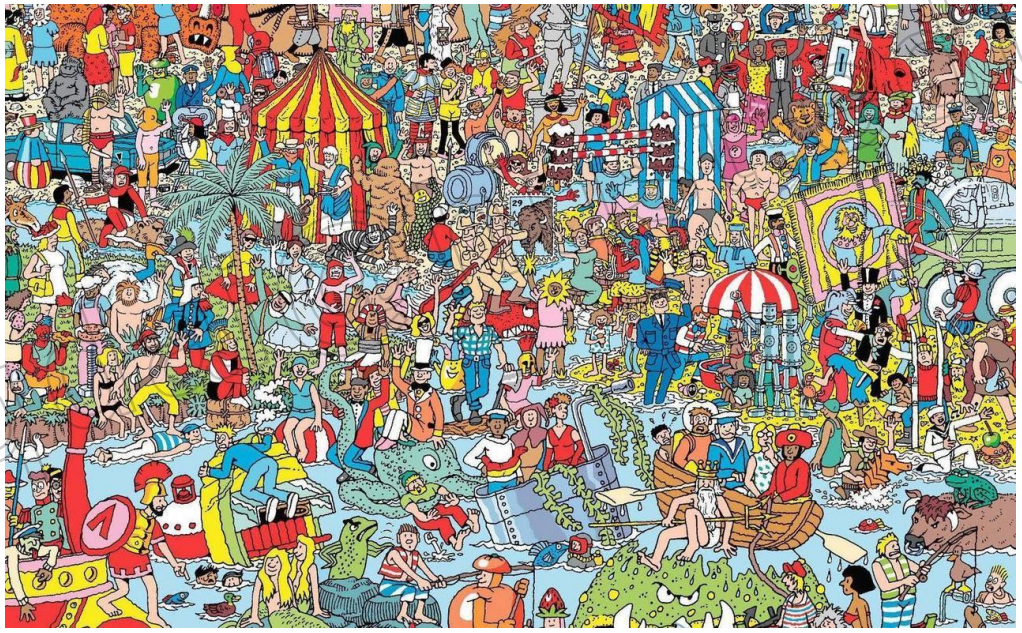
从用户交易的角度上看，比特币的匿名性是不能接受的

目录

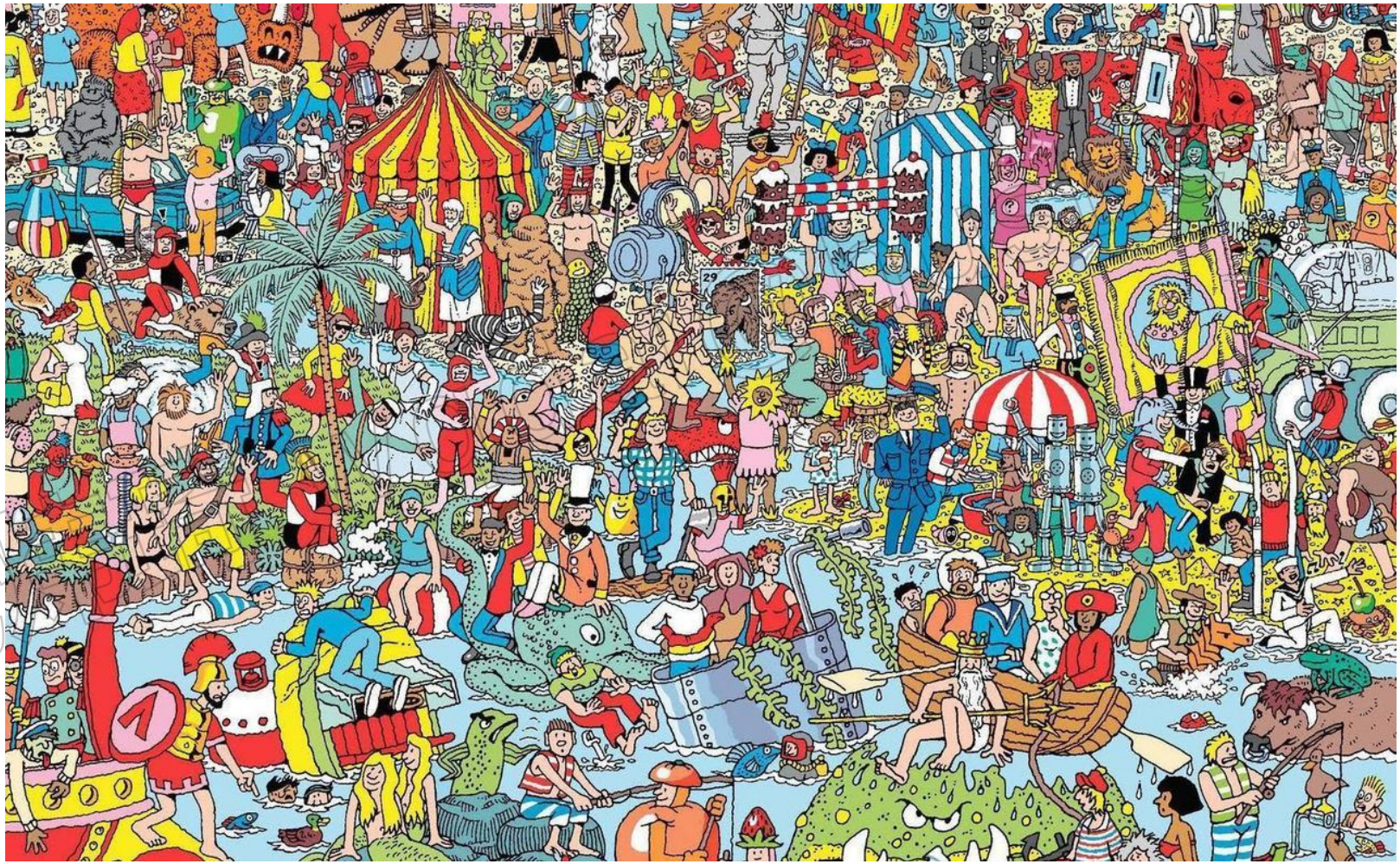
零知识证明--简介

- 寻找瓦尔多

怎么在**不暴露瓦尔多具体位置**的情况下向别人证明你找到了瓦尔多?



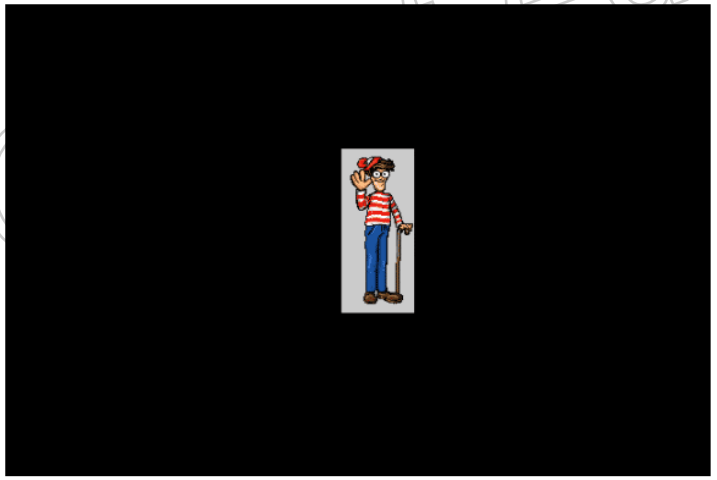
零知识证明--简介



零知识证明--简介

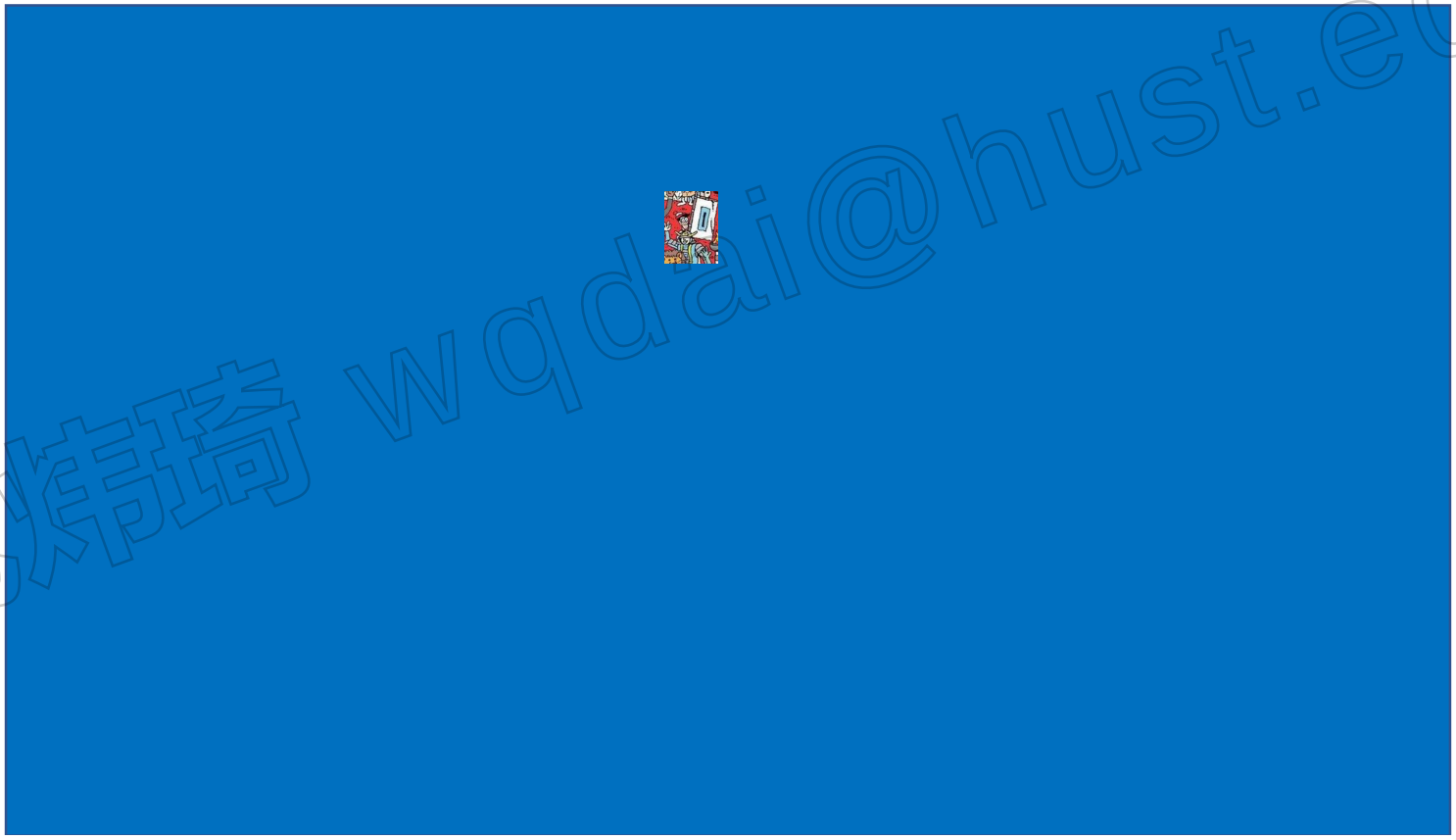
- 简易解决方案：

首先准备来一个大纸板，大约是游戏图片的两倍大小，然后在纸板上剪出一个**矩形小窗口**。接着，可以在确保 其他人没有偷看的时候，在游戏图上移动纸板，使得瓦尔多的图案正好出现在矩形窗口中。

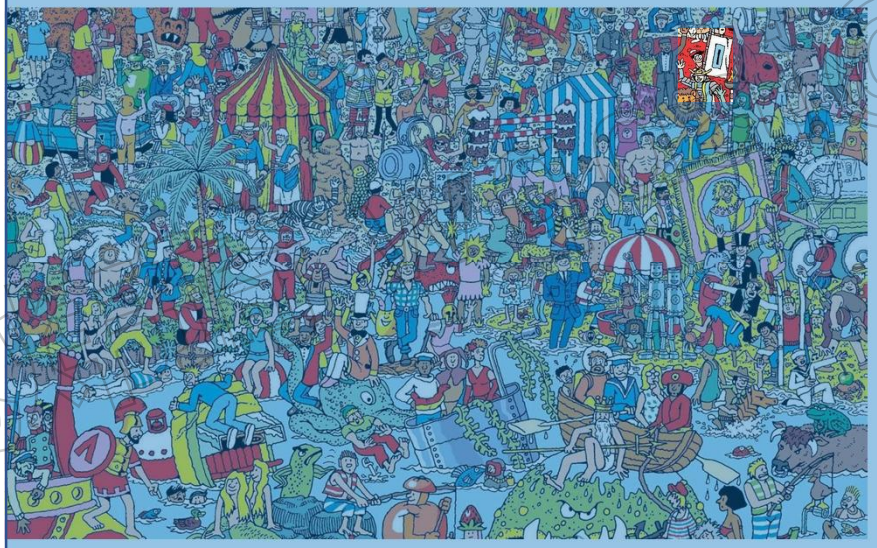


这样我们就在**没有暴露具体位置**的情况下向他人证明我们知道这个问题的答案。

零知识证明--简介



零知识证明--简介



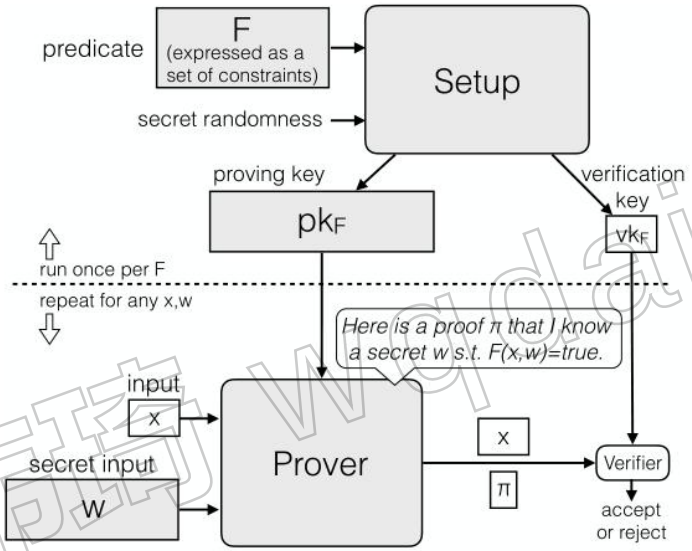
零知识证明--非正式定义

零知识证明:证明者能够在不向验证者提供任何有用的信息的情况下，使验证者相信某个论断是正确的。常被用来证明/验证如下NP语句(NP statement):

“对于某一**公开论断（predicate）**F和**公开输入**x，我知道**某一秘密输入**w,使得w是论断F关于x的**正确解**。”

“对于**寻找瓦尔多**这个问题以及**公开的图片**，我知道**瓦尔多的位置信息**，使得其他人在图片中**能够找到**瓦尔多””

零知识证明—zk-snark协议



zk-snark是" zero knowledge Succinct Non-interactive ARgument of Knowledge"的简写，是非交互式零知识证明（Non-interactive zero-knowledge proofs）中的一种，zk-snark将F和某一秘密随机数通过$setup$算法生成pk, vk 。

证明者，也就是知道F答案x的人，通过证明秘钥pk、 x和秘密输入w生成零知识证明π（ π长度固定），可以被任意验证者在有限时间内使用vk校验，但是证明者无法通过π的值获取w的有效信息。

目录

- 账户模式与UTXO
- 零知识证明
- Zcash
- 资产恢复方案

ZCash

比特币模型

$$UTXO_1 = (PK_1), UTXO_2 = (PK_2), UTXO_3 = (PK_3)$$

ZCash

Zcash 简要模型

$$UTXO_1 = (PK_1), UTXO_2 = (PK_2), UTXO_3 = PK_3$$

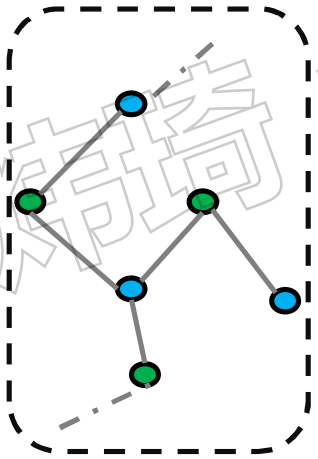
加入序列数r对UTXO进行混淆

$$UTXO_1 = (PK_1, r_1), UTXO_2 = (PK_2, r2), UTXO_3 = (PK_3, r_3)$$

链上不再记录UTXO，而是他们的摘要表

hash计入区块链网络

$$H_1 = HASH(UTXO_1), H_2 = HASH(UTXO_2), H_3 = HASH(UTXO_3)$$

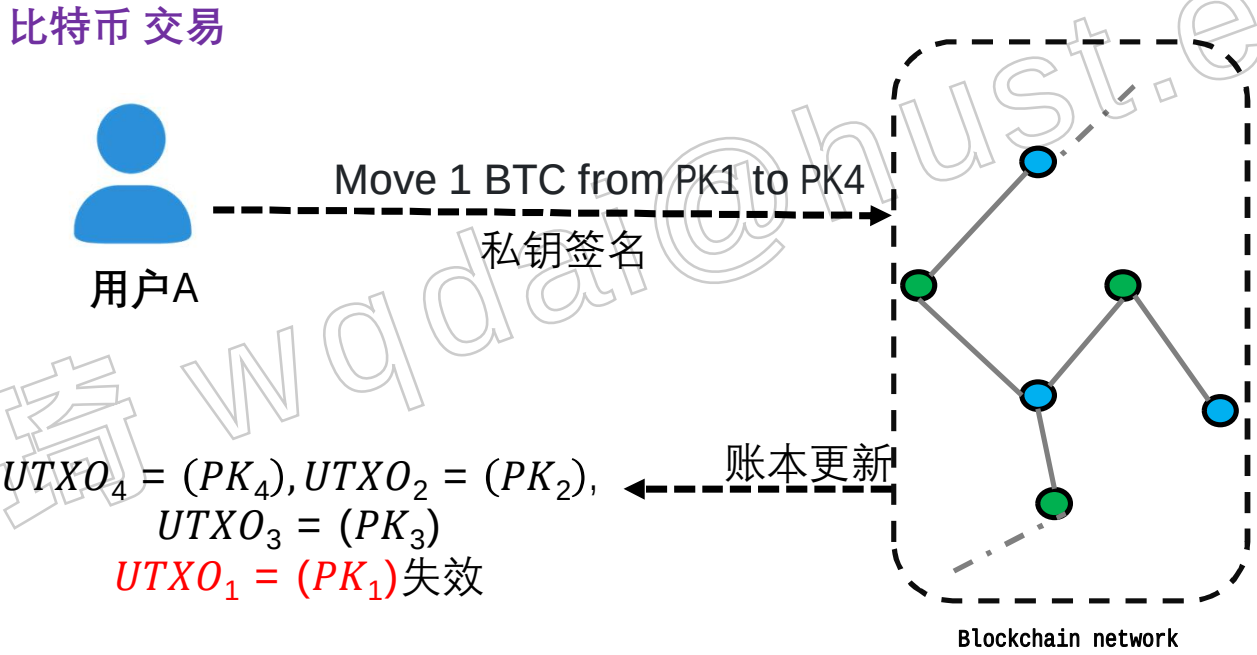


Blockchain network

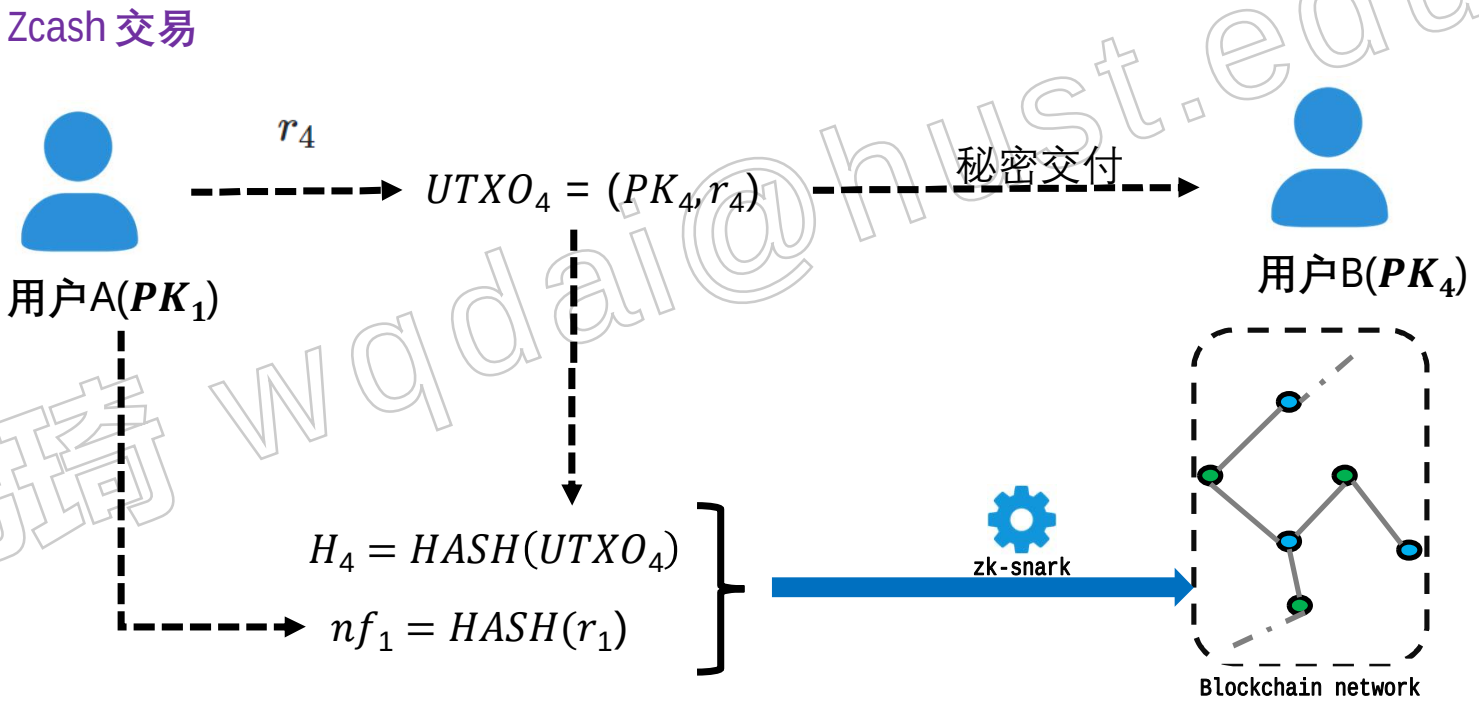
记录已经被消费的UTXO的随机数

Hashed UTXO	Nullifier set
$H_1 = HASH(UTXO_1),$	$nf_2 = HASH(r_2)$
$H_2 = HASH(UTXO_2)$	
$H_3 = HASH(UTXO_3)$	

ZCash



ZCash



ZCash

零知识证明内容

用户A知道公共输入 $UTXO_4 = (PK_4, v)$ 以及秘密输入 SK_1和r_1,需要向区块链网络证明以下几点:

1、 $UTXO_1 = (PK_1, v)$的hash值存在于区块链网络的hash表中
2、 用户A知道PK_1对应的私钥SK_1
3、 $nf_1 = HASH(r_1)$是关于r_1的hash，如果nf_1没有出现在nullifier set中，说明对应的$UTXO$还没被消费

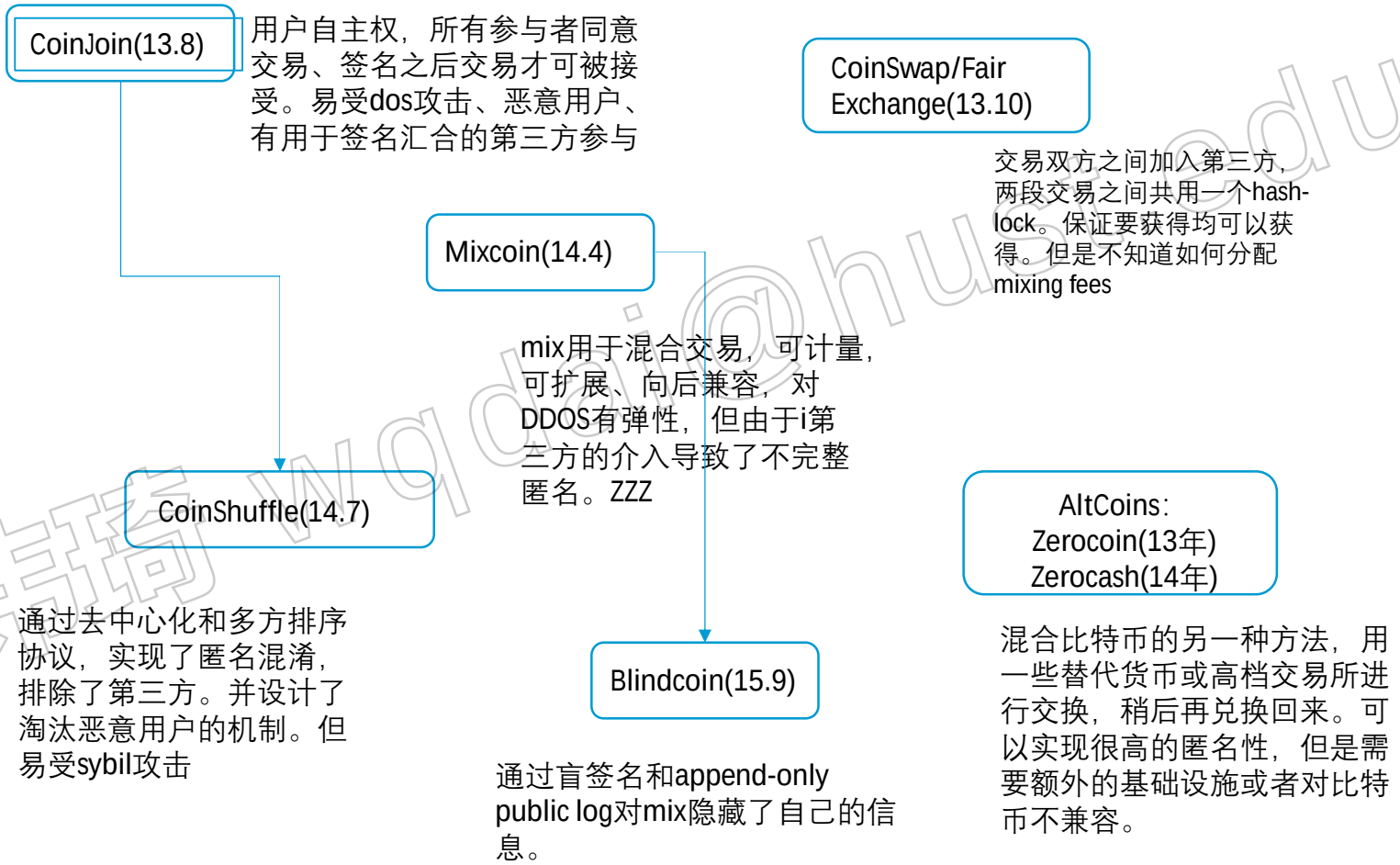
对应证明要点

- 证明pk1上拥有某一UTXO的hash
- 证明用户有权使用该UTXO进行消费
- 证明该UTXO还没有被用户消费

隐私保护技术

技术名称	定义	不足	创新点
链下方式（off-chain）（zeroCoin侧链）	这一分类包括链下信息传送、侧链（sidechain）以及区块链中隔离数据的交易通道。侧链和交易通道使用者可以在私有化控制下的链中进行交易，其中的资产在公链中也是有效的。	不同计算机中需数据复制，因此会破坏网络回弹力	多个隔离的交易通道，公私数据带并存。传播阶段
环签名（以太坊crytoNote环签名）	发送的交易通常捆绑了多个发送方的公钥，仅从环签名难以识别出交易发送方以及最终的签署方信息。实质为基于群签名的改进	部分观察值能够提高攻击者定位地址的准确度，三角分配的方法定位虚假地址	交易携带多个公钥。隐藏发送方私钥。（人群中隐藏）
零知识证明（ZeroCoin）	用于加密数据的密码学验证，可以不公开发送方以及交易金额，但同时又能做到证明这笔交易的合理性。	速度很慢（48秒），不适合大流量交易。且加密货币用此技术，还需其他一些密码学要素	验证方面的隐私性保护
隐身地址（bitcoinj\BIP）	是发选择 一个大随机数。这个数字用一系列公式（公式中因含着数据种子）进行计算后得到新公钥，其对应的私钥只能由收款人计算出来，而且与原来的数据种子不相关。 款人	不提供“100％匿名” 如果您了解一个交易或其中的一方，您可以推断出那些硬币来自哪里或去哪里。	收款方从多个不同的地址收入这些钱，所有交易都看起来毫无联系。同时付款方也只知道自己支付的这笔交易，无法知道其他人跟收款方的交易

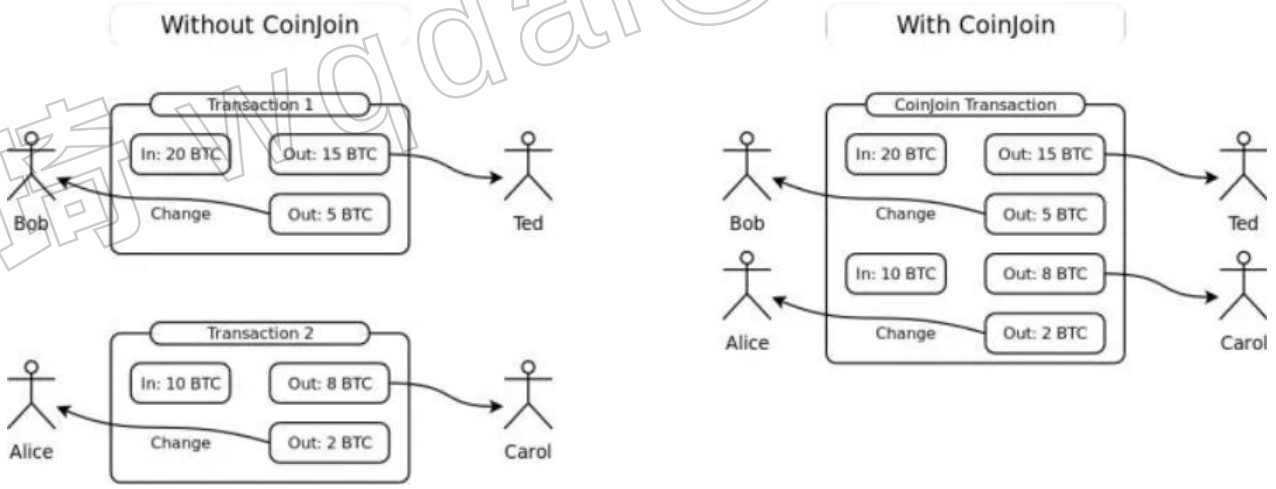
解决方案发展脉络



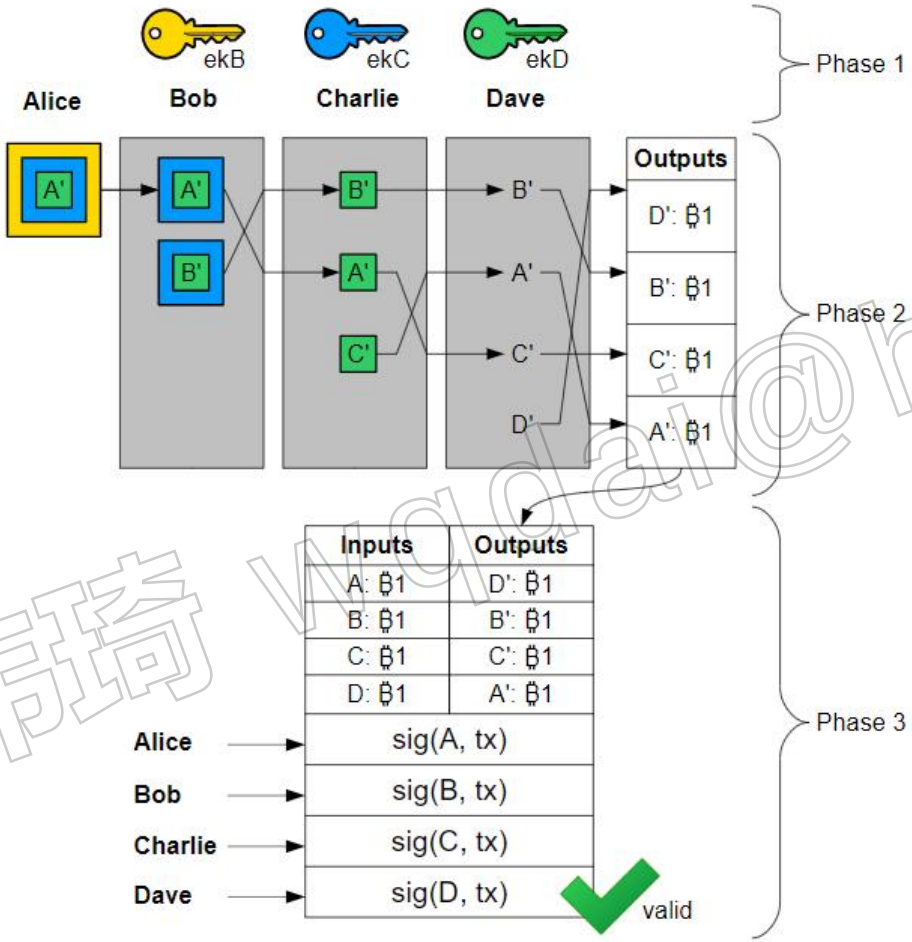
隐私保护方案--coinJoin

coinJoin:
背景：传统上，一个交易就只是一次转帐，不管交易双方是一对一、一对多还是多对多的方式，都很容易从区块链上得出交易双方的转账信息

而coinJoin提出的改进方案是**多个转账人**之间合作组成**一个群体**，将所有的转账打包到**一个交易**中，这样就无法知道输入和输出之间的关系了。



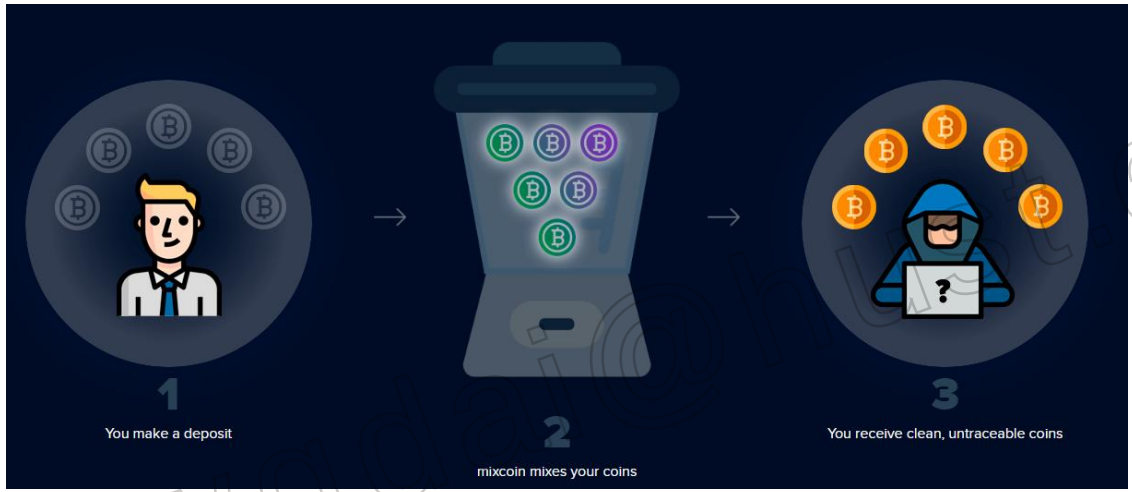
隐私保护方案--CoinShuffle



An overview over a successful run of CoinShuffle. [svg] [png]

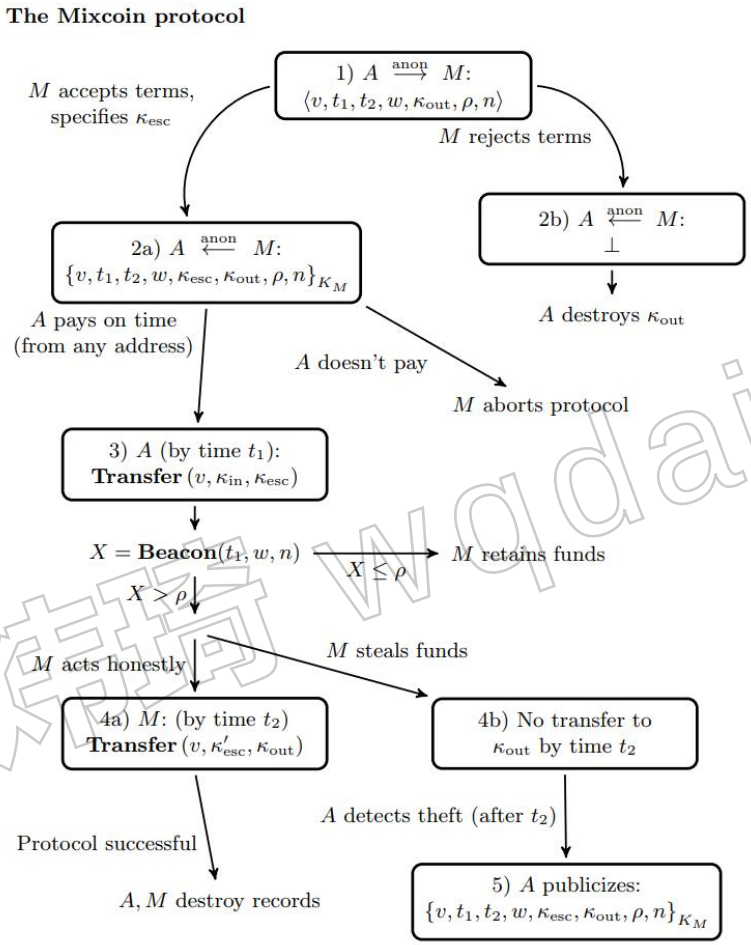
- 为了创建一个混合协议，同时确保输入地址不能与新的输出地址相连，参与者以一种遗忘的方式洗牌他们的输出地址，类似于解密混合网络

隐私保护方案--mixcoin



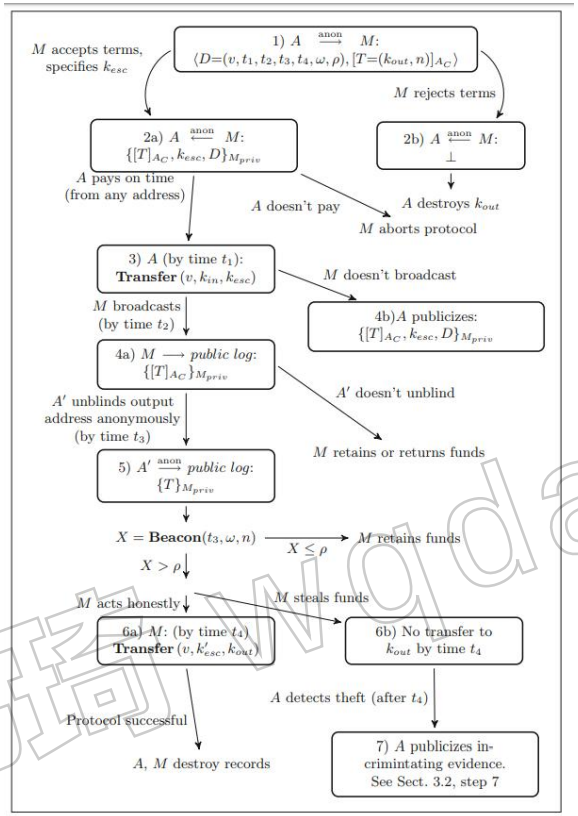
- 当用户向mix系统缴纳费用之后，他们就会受到保护，同时被提供一个保证，当mix系统作恶时，用户可以提供证据。
- 该保证包括一个签署的协议，如果用户在一段时间内将资金支付给组合指定的托管地址，那么组合将在达成协议的最后期限之前将等量的资金转移给用户指定的输出地址
- 匿名性的实现：收集混合费会激励mix行为诚实行事。如果mix费用足够高，那么寻求最大化利润的理性mix系统不会冒险作弊，以免被抓住。

隐私保护方案--mixcoin



v	the value (chunk size) to be mixed
t_1	the deadline[9] by which Alice must send funds to the mix
t_2	the deadline by which the mix must return funds to Alice
κ_{out}	the address where Alice wishes to transfer her funds
ρ	the mixing fee rate Alice will pay
n	a nonce, used to determine payment of randomized mixing fees
w	the number of blocks the mix requires to confirm Alice's payment

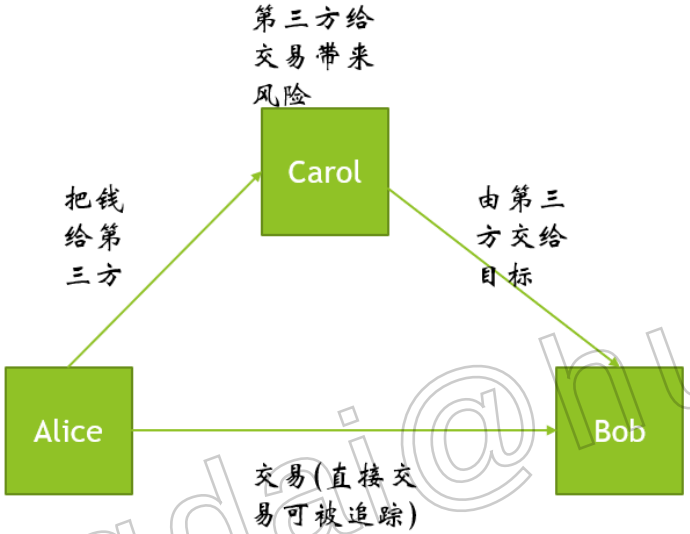
隐私保护方案--Blindcoin



k_{in}	the address from which the A pays, possibly linked to A's true identity
k_{out}	the address to which the user wishes funds transferred
k_{esc}	an escrow address, unique for each user, that M provides for A to pay
k'_{esc}	an escrow address that M uses to pay to k_{out}
A'	an anonymous identity that A can use to post to the public log
M_{pub}	the public key of M
M_{priv}	the private signing key of M
A_C	a secret commitment/encryption function of A
$A_{C'}$	the inverse of A_C
ω	the number of blocks M requires to confirm A's payment
n	a per-user nonce, used to determine payment of randomized mixing fees
v	the value (chunk size) to be mixed
ρ	the mixing fee rate A will pay
T	the token, which is the triple (k_{out},n)
t_1	the time by which A must v BTC to k_{esc} in order to participate in the mix
t_2	the time by which M must post the token T to the public log
t_3	the time by which A' must unblind the output address via the public log
t_4	the time by which the mix must transfer v BTC to k_{out}
D	the mix parameters, a tuple $\{t_1,t_2,t_3,t_4,v,\omega,\rho\}$

在Mixcoin协议中，从用户输入到输出地址的映射对mix服务器是可见的。Blindcoin修改Mixcoin协议，以确保对mix服务器隐藏任何用户的输入/输出地址映射。为了实现这一点，我们使用了一个盲签名方案和一个只进行追加操作的公共日志。该方案与比特币完全兼容，强制混合必须负责任，甚至在恶意混合的情况下也能保持用户匿名，对拒绝服务攻击很有弹性，而且很容易扩展到许多用户。

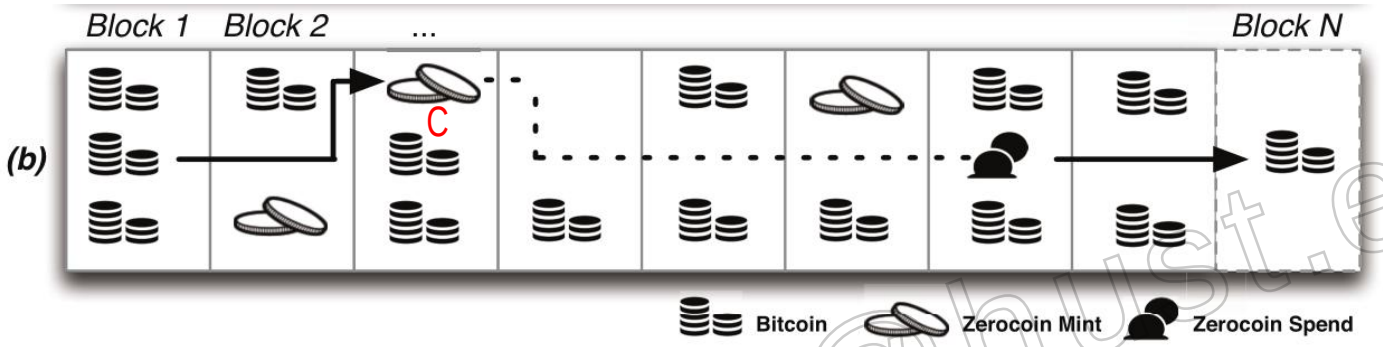
隐私保护方案--CoinSwap



CoinSwap [Ma13b]是格雷戈里•马克斯韦尔(Gregory Maxwell)提出的另一项通过第三方进行交易的建议。爱丽丝没有直接把硬币交给鲍勃，而是把硬币交给卡罗尔，卡罗尔又把硬币交给鲍勃。Alice和Carol以及Carol和Bob之间的事务是托管事务，可以使用受散列锁(hashlock)保护的赎回事务来使用这些事务。这就保证了爱丽丝和卡罗尔都不能偷币。

如今，CoinSwap在比特币上是可用的。它甚至可以用于跨不同链执行事务。然而，匿名性确实依赖于所有2of2e

隐私保护方案--zerocoin



假设Alice需要隐藏相关的交易，那么她可以选取一个**序列数（serial number）S**和**随意数r**，通过**$Hash$得到C**,同时花费$1来锻造一个ZeroCoin，网络中所有节点共同维护一个Clist.

当Alice需要花费这个ZeroCoin的时候，她就把（S，π）提交到区块链网络中，**π是非交互式零知识证明**，证明两点：

1）她知道一个C 属于$(c1, \ldots Cn)$，

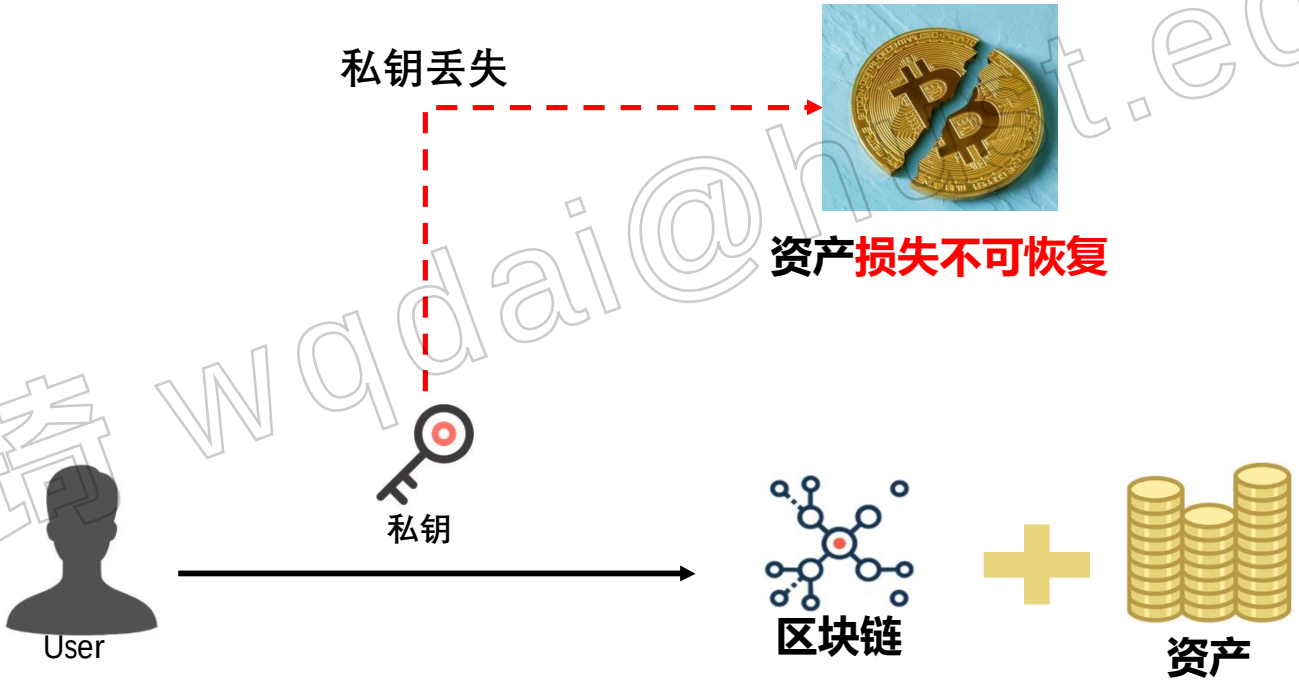
2）她知道一个r可以把S通过$hash$成C。

如果所有节点的零知识验证通过且之前没有任何一笔交易包含S,那么Alice可以以此花费$1.

目录

- 账户模式与UTXO
- 零知识证明
- Zcash
- **资产恢复方案**

区块链资产丢失问题

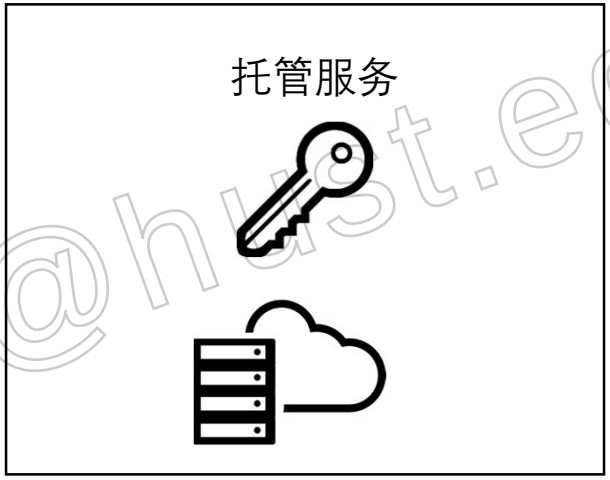


区块链资产丢失问题

据数字取证公司Chainalysis对比特币区块链进行的研究，大约有230万到370万个比特币已经永远消失，对应当时比特币市值的13％到 22％

2018年12月加拿大最大的数字货币交易平台Quadriga CX创始人兼CEO Gerald Cotton突然去世，也带走了平台上价值高达1.9亿加元（1.45亿美元）的数字货币，因为这些数字货币的密钥只有Gerald Cotton掌握。

现有私钥管理方式



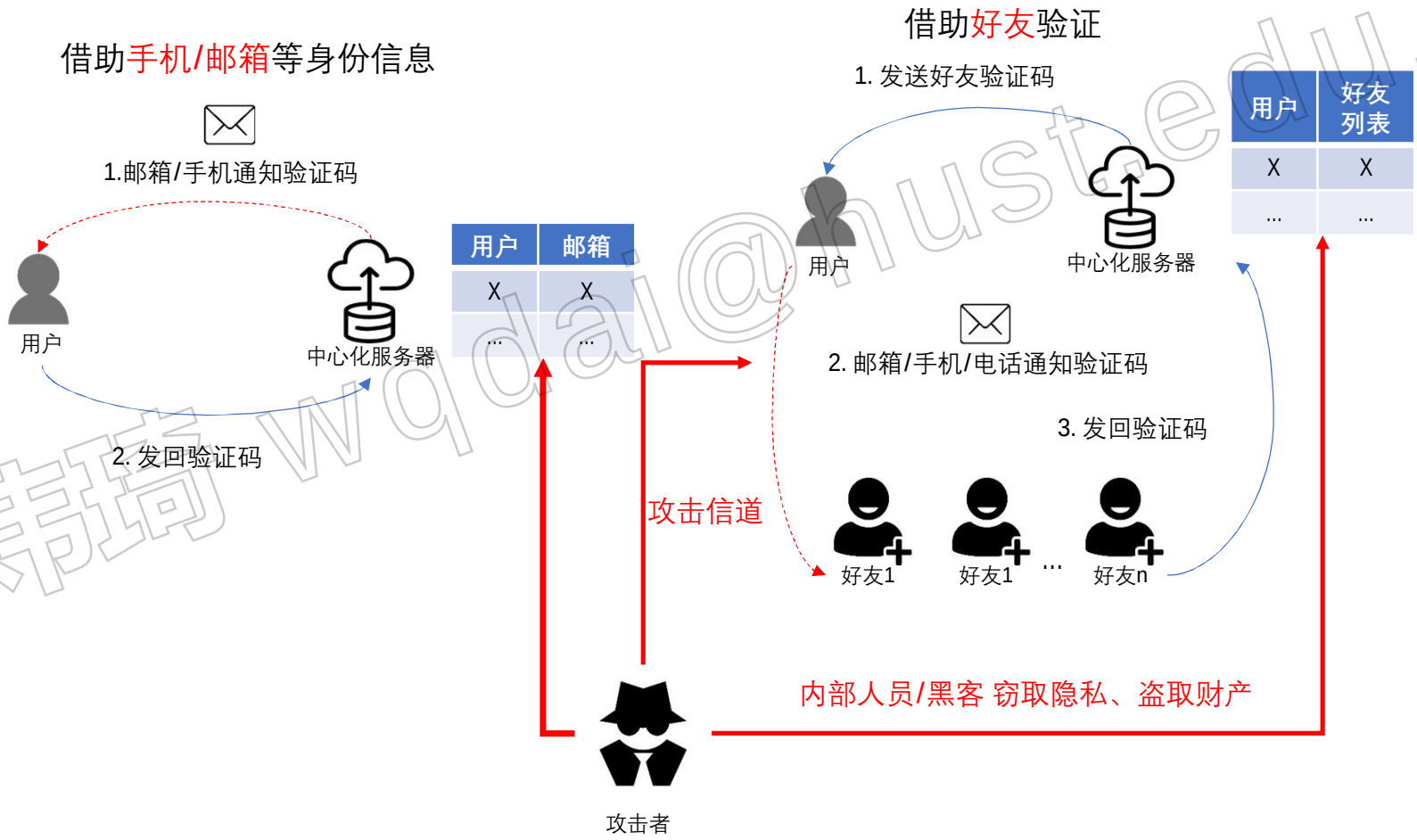
- 依赖存储载体的安全性
  - 物理损毁
  - 不慎丢失
  - 失窃

230~370万BTC已经永久丢失

- 依赖于中心化服务器
  - 监视隐私
  - 黑客攻击
  - 内部人员作恶

棱镜门

传统账户恢复手段



研究现状

资产管理与恢复	TrustZone-backed bitcoin wallet **（安全硬件架构）**
	Highly-efficient and composable password-protected secret sharing (or: How to protect your bitcoin wallet online) **（私钥保护）**
	A social-network-based cryptocurrency wallet-management scheme **（社交网络恢复私钥）**
	EOS-An introduction **（私钥丢失后通过预设的好友恢复）**
传统恢复方法	Web Password Recovery: A Necessary Evil **（基于邮件恢复）**
	Secure login by using One-time Password authentication based on MD5 Hash encrypted SMS （**基于**SMS**恢复**）
	It ‘ s not what you know, but who you know: a social approach to last-resort authentication **（基于社交关系恢复）**
	Codes v. people: A comparative usability study of two password recovery mechanisms **（基于备份代码和可信赖人员恢复）**
区块链身份认证	BCTrust: A decentralized authentication blockchain-based mechanism **（信任链）**
	An Authentication Scheme Using Identity-based Encryption & Blockchain **（身份与私钥绑定**）
	Beyond the hype: On using blockchains in trust management for authentication. **（区块链作为信任管理基础设施）**

1、研究主要集中在私钥管理与恢复，从系统层面进行资产恢复的研究比较少
2、传统恢复方法较为成熟，但无法直接套用在去中心化系统上

解决方案

- 基于多方验证的数字资产恢复方案
- 基于零知识证明的数字资产恢复方案

基于多方验证的数字资产恢复方案

❖ 目标

1. 实现一种基于**去中心化验证者**的身份认证模型，在私钥丢失的情况下，从系统层面恢复数字资产，不影响区块链原有功能。

2. 建立相关**分析模型**，验证设计的安全性。

3. 在以太坊上**实现**该系统，测试性能与存储开销。

基于多方验证的数字资产恢复方案

❖ 设计思路

设计需要考虑的2个关键点

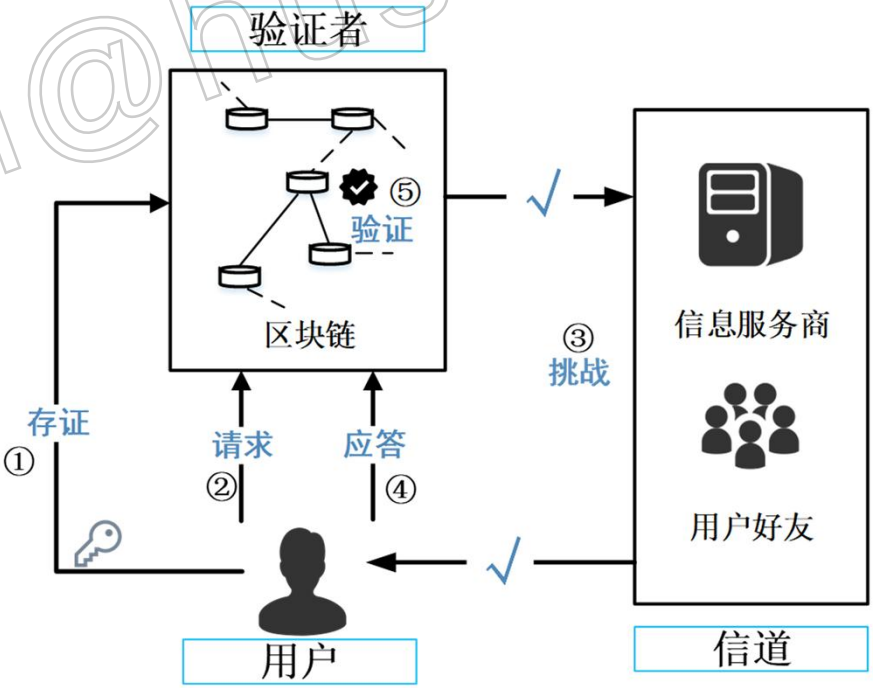
- 私钥丢失后的身份**标识**
- 系统如何**认证**身份标识

❖ 系统总览

主要角色：

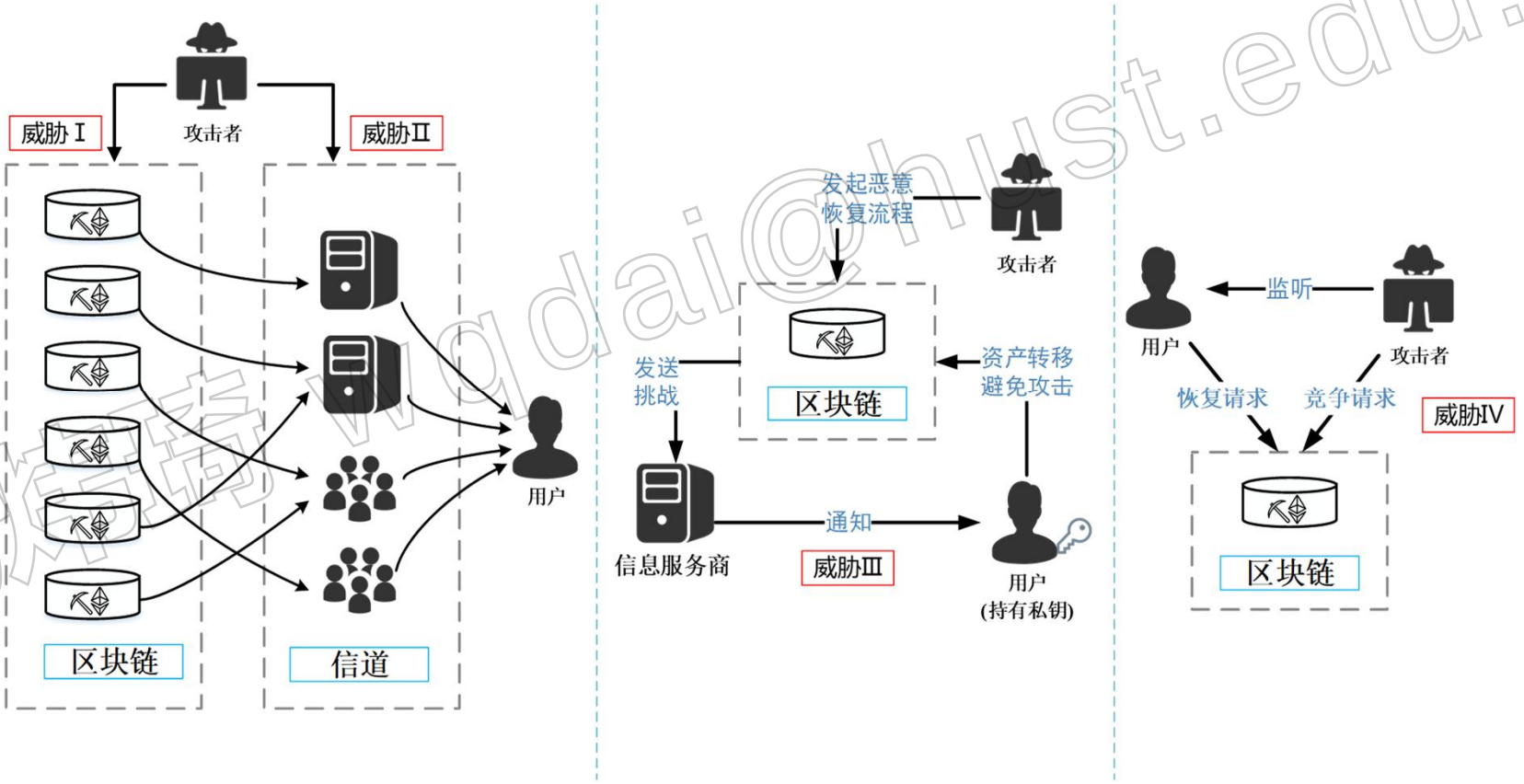
- **用户：**数字资产与私钥拥有者
- **验证者：**区块链上的矿工

信道：作为安全的消息传输通道，连接用户与验证者



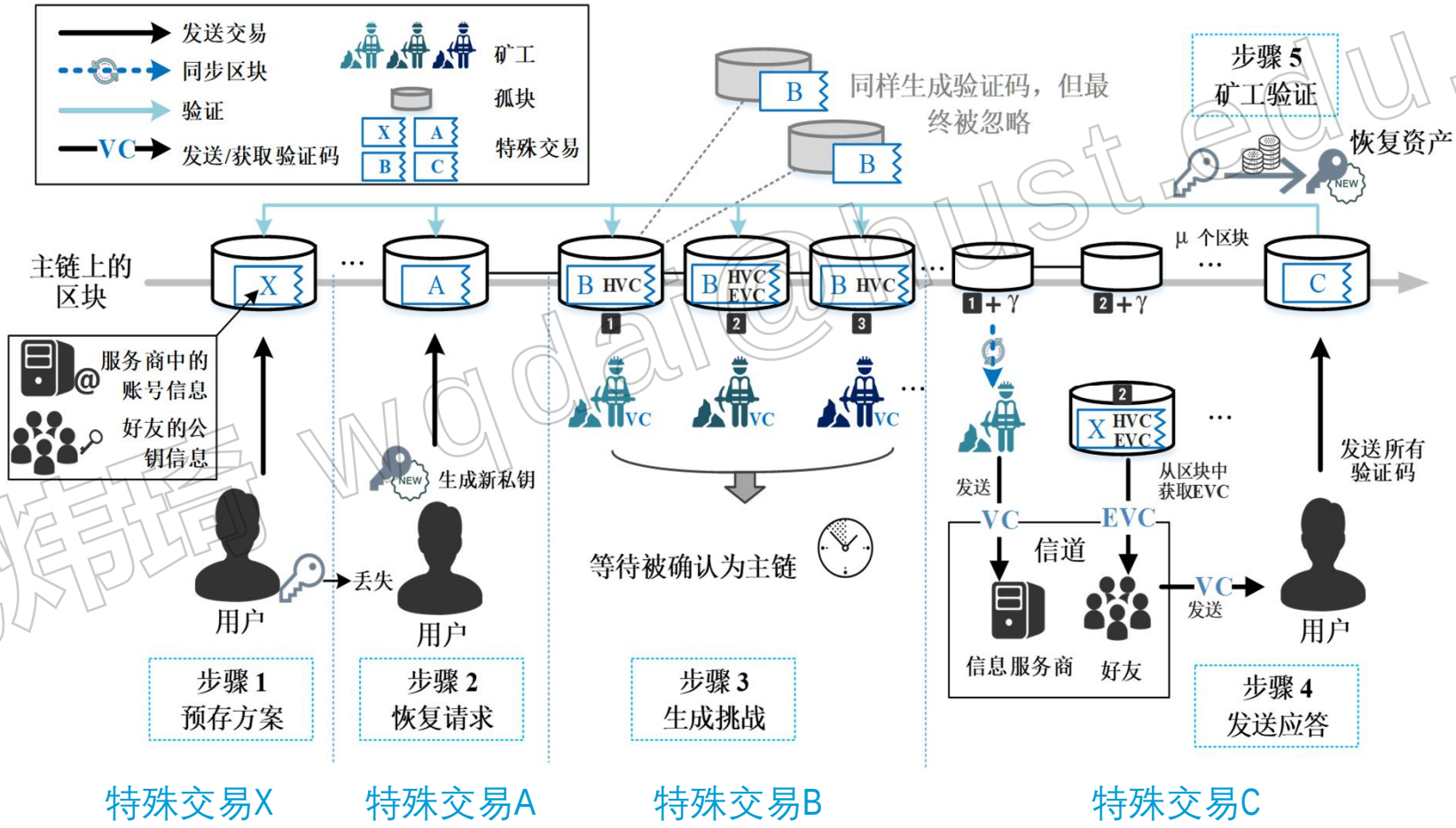
基于多方验证的数字资产恢复方案

❖ 威胁模型



基于多方验证的数字资产恢复方案

❖ 主要流程



基于多方验证的数字资产恢复方案

Step 1: Certifier

- 信道信息
- 信道数量(M)
- 验证码数量（N）
- 最少提交验证码数(n)

Step 2: Request

- 新地址
- 源地址
- 攻击状况

Step 3: Challenge

- 公平的生成策略
- 特殊交易的优势
- 根据序列数选择信道
- 生成HVC
- 生成EVC
- 延迟确认
- 避免重复生成

Step 4: Attestation

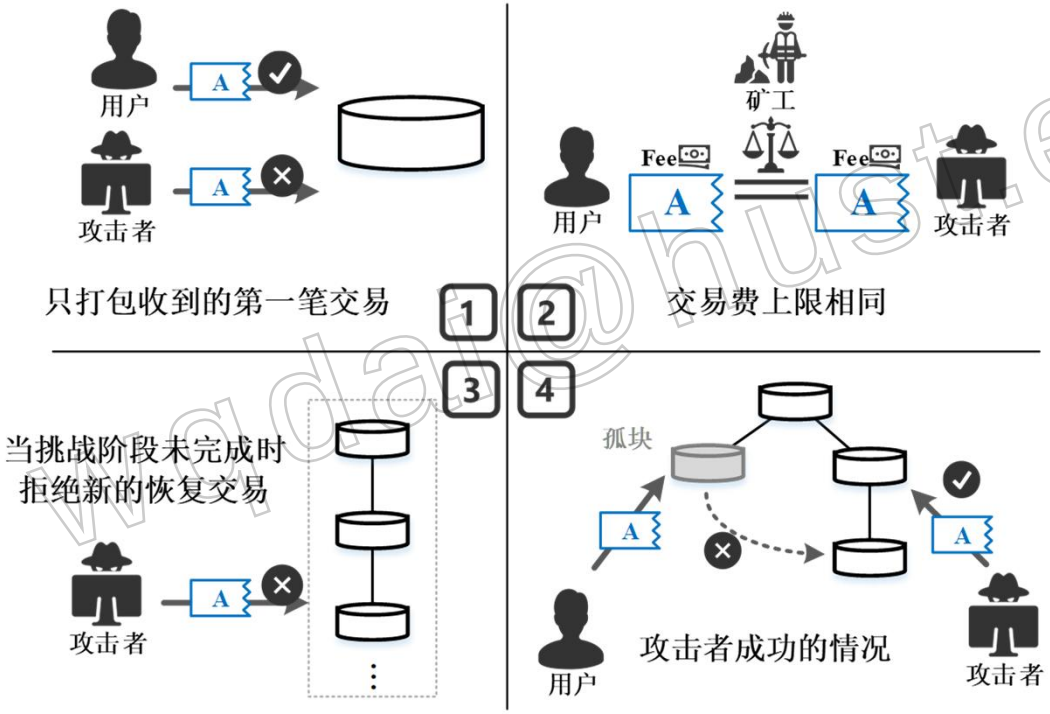
- 发送验证码的方式
- 收集验证码
- 意外收到验证码

Step 5: Verify

- 限制(6000 blocks)
- 失败

基于多方验证的数字资产恢复方案

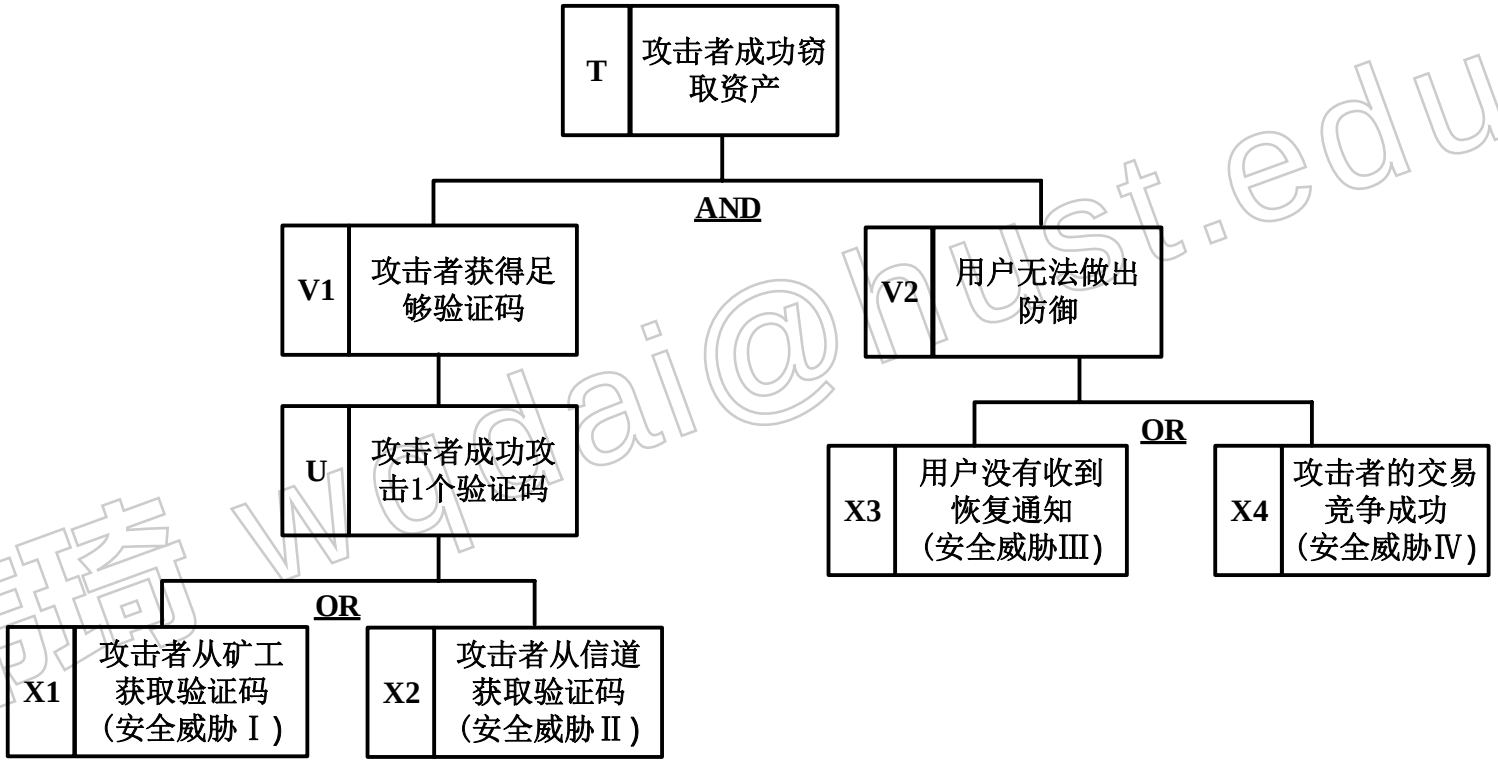
❖ 设计细节



设计了相应的策略，减少攻击者发送竞争交易带来的影响

基于多方验证的数字资产恢复方案

❖ 攻击树模型



需要同时满足左右子树才能达成攻击

基于多方验证的数字资产恢复方案

❖叶结点计算

	攻击原因	攻击概率	开销
X1	攻击者拥有足够算力掌握出块权	攻击者算力占全网百分比	租赁算力的开销
X2	攻击者成功攻击信道	控制信道的概率	控制信道的开销
X3	阻止用户获取所有通知	阻止用户从所有信道中收到通知	支付给矿工的奖励
X4	竞争交易成功	叔块概率	支付给矿工的奖励

攻击收益期望：当攻击成功时，攻击开销的期望值
（作为安全的用户资产上限参考）

$$E = \frac{C_T}{P_T}$$

基于多方验证的数字资产恢复方案

❖ 攻击树结点汇总

	Probability	Cost ($)
X1	$P_{X1} = K$	$C_{X1} = R \cdot N \cdot K$
X2	$P_{X2} = \alpha$	$C_{X2} = \beta$
X3	$P_{X3} = \left\{\left(\frac{1}{2} + \frac{1}{2} \cdot P_{X2}\right) + \left[1 - \left(\frac{1}{2} + \frac{1}{2} \cdot P_{X2}\right)\right] \cdot {P_{X1}}^{\frac{N}{M}}\right\}^{M}$	$C_{X3} = C_{attacker}$
X4	$P_{X4} = H$	$C_{X4} = C_{attacker}$
U	$\begin{cases} P_{Ua} = 1 - (1 - P_{X1}) \cdot (1 - P_{X2}), & N \le M \\ P_{Ub} = P_{Ua} + (1 - P_{Ua}) \cdot P_{X2}, & N > M \end{cases}$	$C_U = \frac{P_{X1} \cdot C_{X1} + \frac{M}{N} \cdot P_{X2} \cdot C_{X2}}{P_{X1} + P_{X2}}$
V1	$P_{V1} = \begin{cases} \sum_{k=0}^{N-n} (1 - P_{Ua})^k \cdot {P_{Ua}}^{N-k}, & N \le M \\ \sum_{i=0}^{M} \sum_{j=0}^{N-n} (1 - P_{Ua})^{M-i} \cdot {P_{Ua}}^{i} \cdot (1 - P_{Ub})^{(N-M)-(n+j-i)} \cdot {P_{Ub}}^{n+j-i}, & N > M \\ n + j - i \ge 0, \quad (N - M) - (n + j - i) \ge 0 \end{cases}$	$C_{V1} = N \cdot C_U$
V2	$P_{V2} = 1 - (1 - P_{X3}) \cdot (1 - P_{X3})$	$C_{V2} = \frac{P_{X3} \cdot C_{X3} + P_{X4} \cdot C_{X4}}{P_{X3} + P_{X4}}$
T	$P_T = P_{V1} \cdot P_{V2}$	$C_T = C_{V1} + C_{V2}$

基于多方验证的数字资产恢复方案

❖时间消耗

类型	原系统	交易X	交易A	交易B	交易C
时间	0.96ms	1.28ms	1.48ms	1.62ms	1.39ms

❖Gas消耗

交易类型	G_X	G_{Friend}	G_{Email}	G_A	G_B	G_C	G_{VC}
Gas计算	50104	14144	3672	68192	69552	59556	8160

1、额外流程增加的时间消耗在毫秒级别
2、Gas消耗为原交易的3倍，在用户可接受范围内

基于多方验证的数字资产恢复方案

❖ 安全验证参数取值

信道安全性的参考值 α = 1%、β = 20000

攻击者算力参考值 K_{max} = 0.3, K_{min} = 0.04

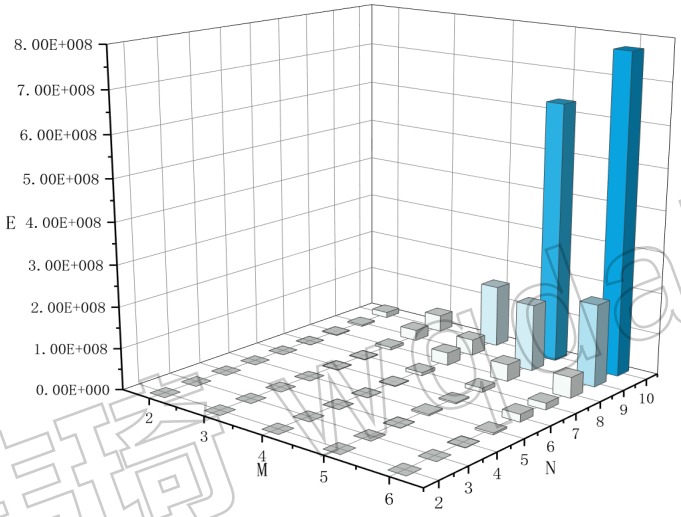
一次攻击中每1%算力的开销参考值 R = 316$

Gas Price参考值 GP = 50$Gwei$

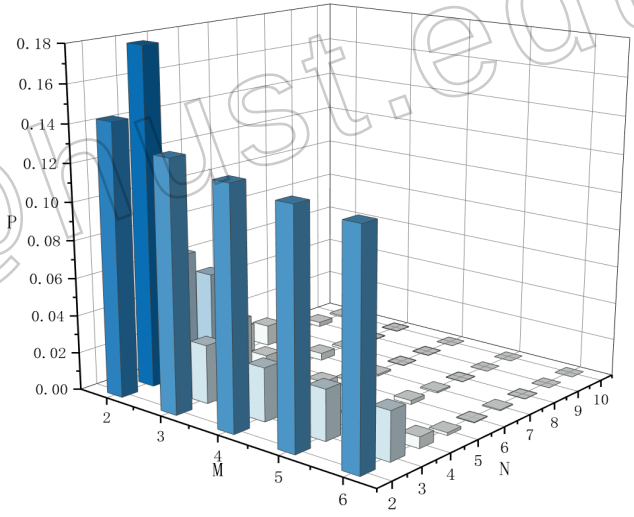
孤块概率参考值 H = 0.076

基于多方验证的数字资产恢复方案

- 信道(M)和验证码(N)变化对安全性的影响趋势



攻击收益期望（E）变化



攻击成功概率（P）变化

1、随着M和N增加，攻击收益期望（E）呈指数增长，成功的概率（P）呈指数下降
2、当N＝M＋1时，因为验证码冗余导致成功概率增加，预存方案阶段需避免该情况

基于多方验证的数字资产恢复方案

- 成功概率P_T为0.0001时的取值参考

$K = 0.3$

M	N	n	P_T	E ($)	User Cost ($)
2	13	6	1.16×10^{-4}	1.45×10^{8}	15.31
2	14	7	$\mathbf{3.64 \times 10^{-5}}$	$\mathbf{5.29 \times 10^{8}}$	16.36
3	10	6	2.80×10^{-4}	3.96×10^{7}	12.54
3	11	7	$\mathbf{8.70 \times 10^{-5}}$	$\mathbf{1.50 \times 10^{8}}$	13.59
4	9	6	2.66×10^{-4}	3.76×10^{7}	11.68
4	10	7	$\mathbf{8.10 \times 10^{-5}}$	$\mathbf{1.45 \times 10^{8}}$	12.73
5	8	6	2.31×10^{-4}	3.94×10^{7}	10.82
5	9	7	$\mathbf{2.06 \times 10^{-5}}$	$\mathbf{6.02 \times 10^{8}}$	12.92

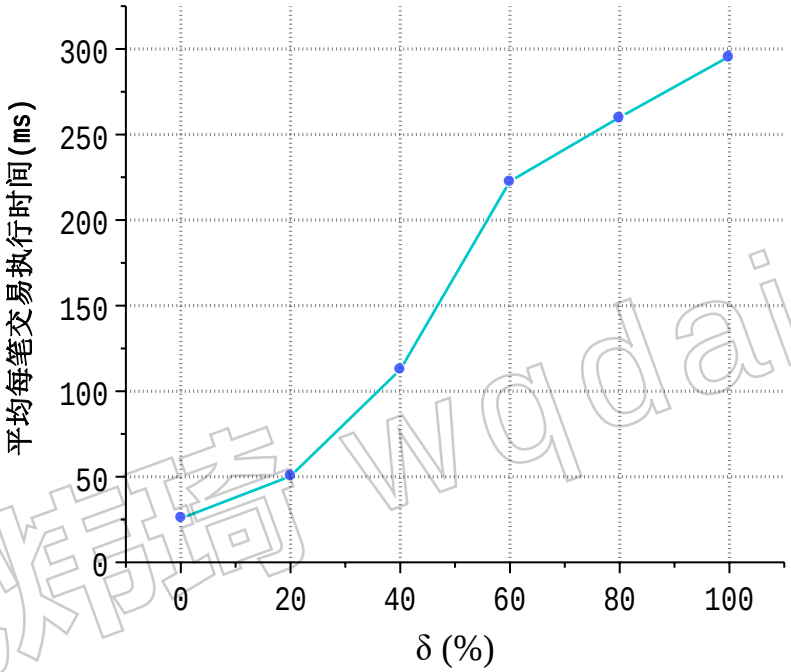
$K = 0.04$

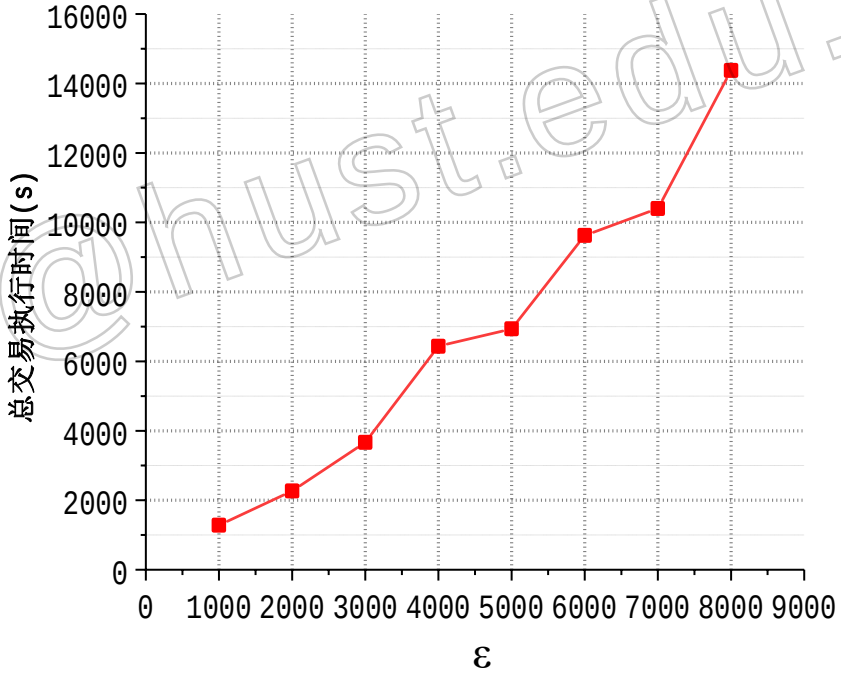
M	N	n	P_T	E ($)	User Cost ($)
2	6	3	1.47×10^{-4}	5.70×10^{7}	8.32
2	**8**	**4**	$\mathbf{8.18 \times 10^{-5}}$	$\mathbf{1.06 \times 10^{9}}$	**10.33**
3	3	2	5.16×10^{-4}	2.35×10^{7}	5.46
3	**5**	**3**	$\mathbf{8.06 \times 10^{-5}}$	$\mathbf{1.52 \times 10^{8}}$	**7.46**
4	3	2	3.94×10^{-4}	4.09×10^{7}	5.55
4	**4**	**3**	$\mathbf{1.78 \times 10^{-5}}$	$\mathbf{9.07 \times 10^{8}}$	**6.60**
5	3	2	3.32×10^{-4}	6.05×10^{7}	5.65
5	**4**	**3**	$\mathbf{1.46 \times 10^{-5}}$	$\mathbf{1.38 \times 10^{9}}$	**6.70**

1、无论M的取值，都可以通过设置更高的N值来实现足够的安全性
2、若用户资产总额小于E，则可以采用CRSMA的恢复方案，且用户开销较小

基于多方验证的数字资产恢复方案

❖ 交易并发测试





1、处理速度降低的主要原因是：交易体积增大，导致每个块中打包的交易量减小
2、实际恢复交易占比较低，对原系统交易处理能力的影响较小

基于多方验证的数字资产恢复方案

❖ 工作总结

- 设计了基于矿工验证者的**身份认证**模型，通过“信道”建立区块链与用户的联系，多方联合认证无私钥用户的身份，实现**资产恢复**。
- 使用**攻击树模型**分析系统安全点，评估整体安全性。最后通过实验测试得到的数据，计算出推荐参数选择。
- 在Geth中实现了系统并与原系统进行比较，分析了恢复流程的额外开销。

❖ 展望

- 研究更多类型的信道方案，并采用更加隐蔽的信道预设方法，使信息不暴露，例如零知识证明等。
- 应用到其他不同类型的区块链中，例如基于UTXO的比特币。
- 采用更详细的安全分析模型，符合现实攻击情况，提供更加精准的分析。

基于零知识证明的数字资产恢复方案

威胁	目标	关键点
威胁1：依赖存储载体安全性	恢复信息可记忆	口令：暴力猜解？ 好友验证：合谋？ 口令+好友验证 用户特有的知识
威胁2：依赖带外信道/中心化服务器	去中心化/去第三方信道	信息存在公开区块链上 公开可验证
威胁3：暴露身份/社交关系	隐私保护	零知识证明 不暴露秘密信息

实现手段：口令+好友 ✚ 区块链 ✚ 零知识证明

基于零知识证明的数字资产恢复方案

简洁非交互式零知识证明(zk-SNARKs):

Zero-Knowledge Succinct Non-interactive Arguments of Knowledge

“我可以证明我知道某个满足条件的秘密值x，并且所有人都可以验证，但是没有人知道x是什么”

口令+好友　　公开可验证　　不暴露秘密信息

零知识证明步骤:

1. Commit

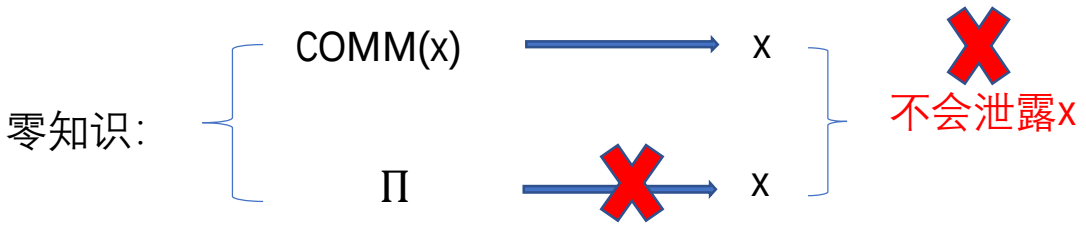
将x绑定到某个Commitment中:
COMM(x)

2. 证明

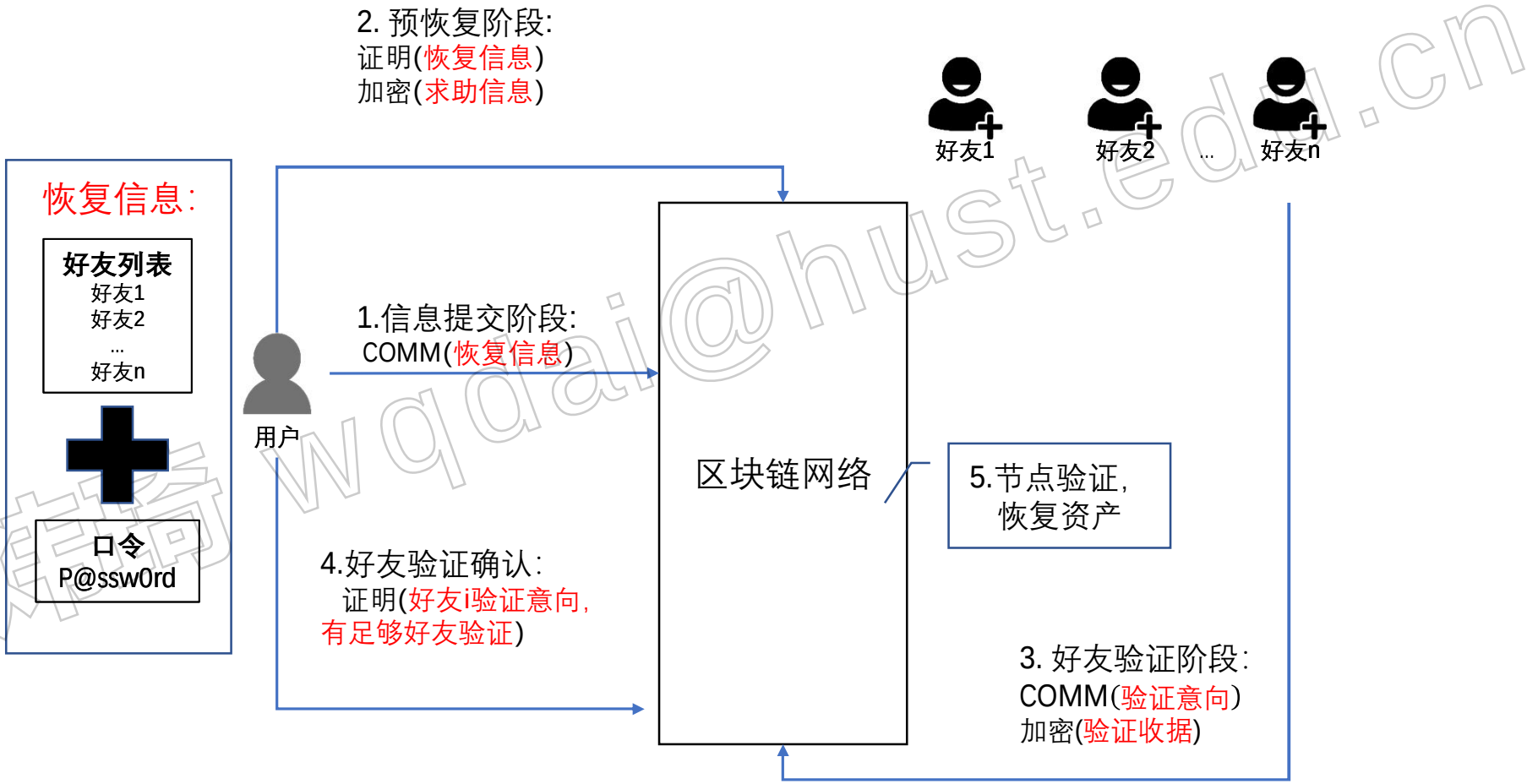
生成关于x的零知识证明:
Π

3. 验证

所有人均可验证:
生成**Π**的人知道COMM(x)绑定的x

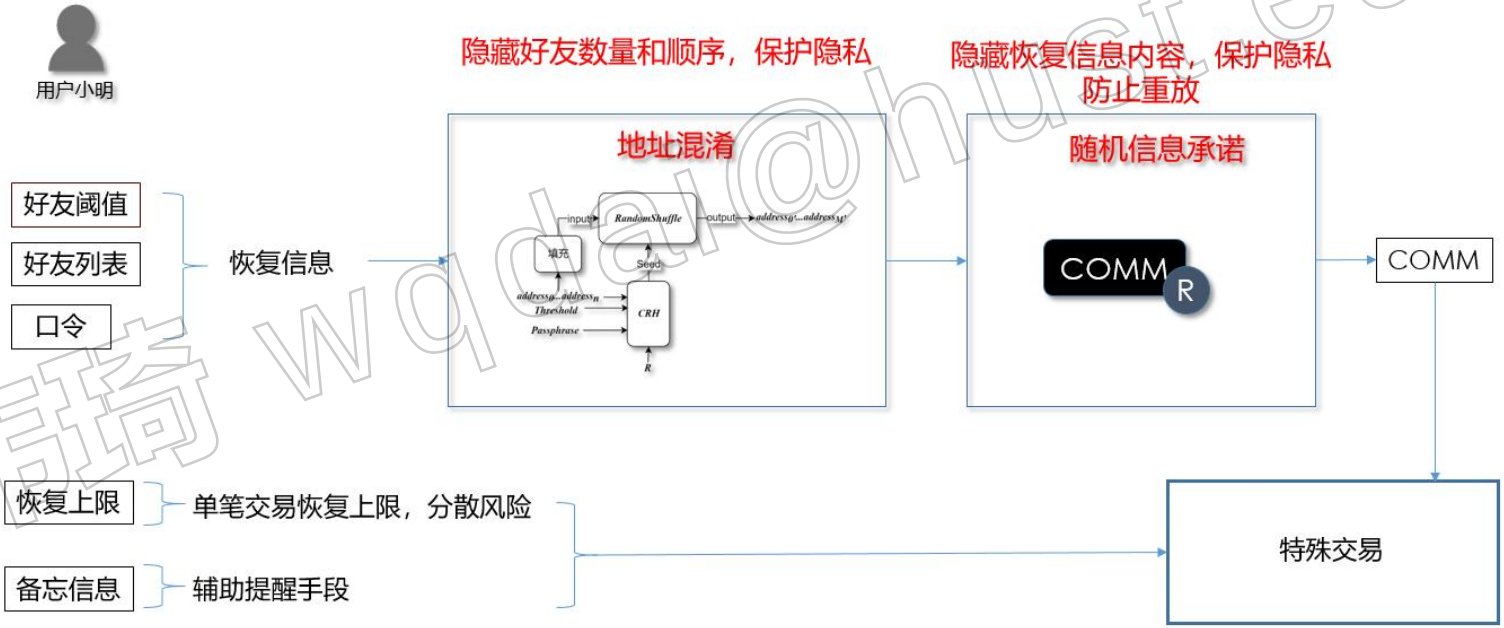


基于零知识证明的数字资产恢复方案



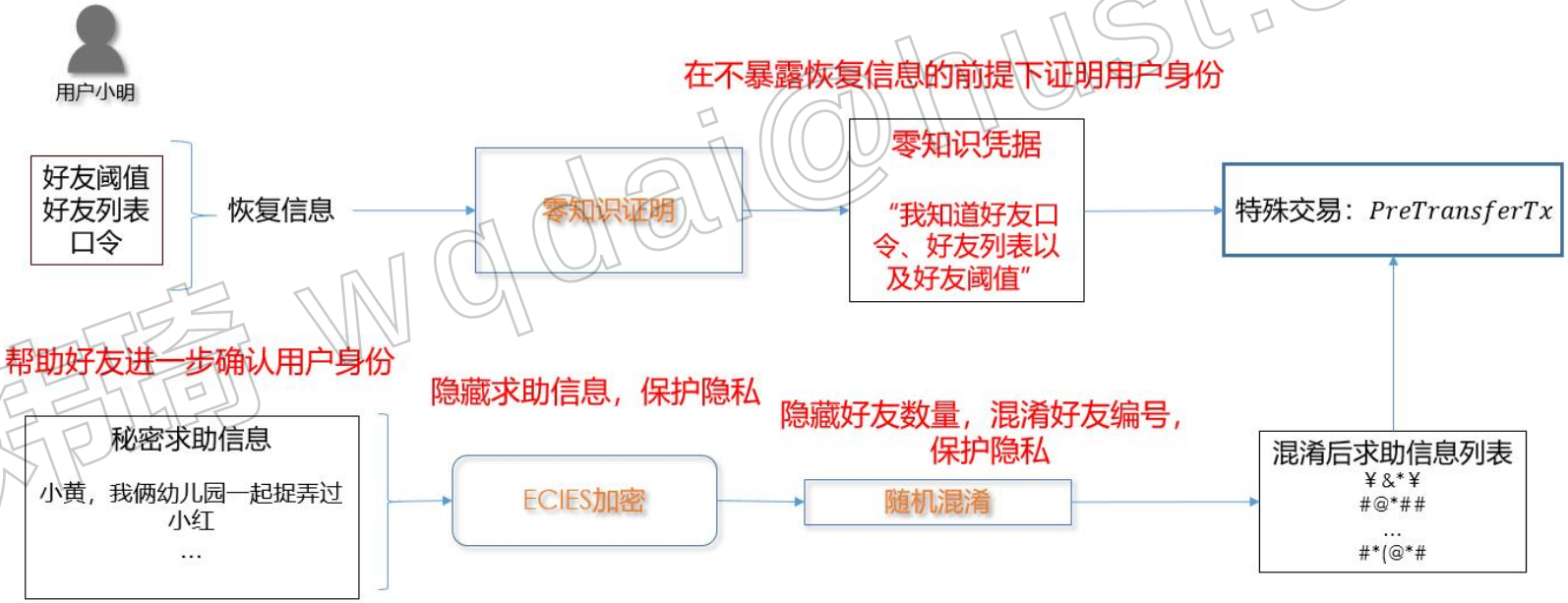
基于零知识证明的数字资产恢复方案

1.提交阶段：用户秘密地将口令及好友列表放置在区块链上



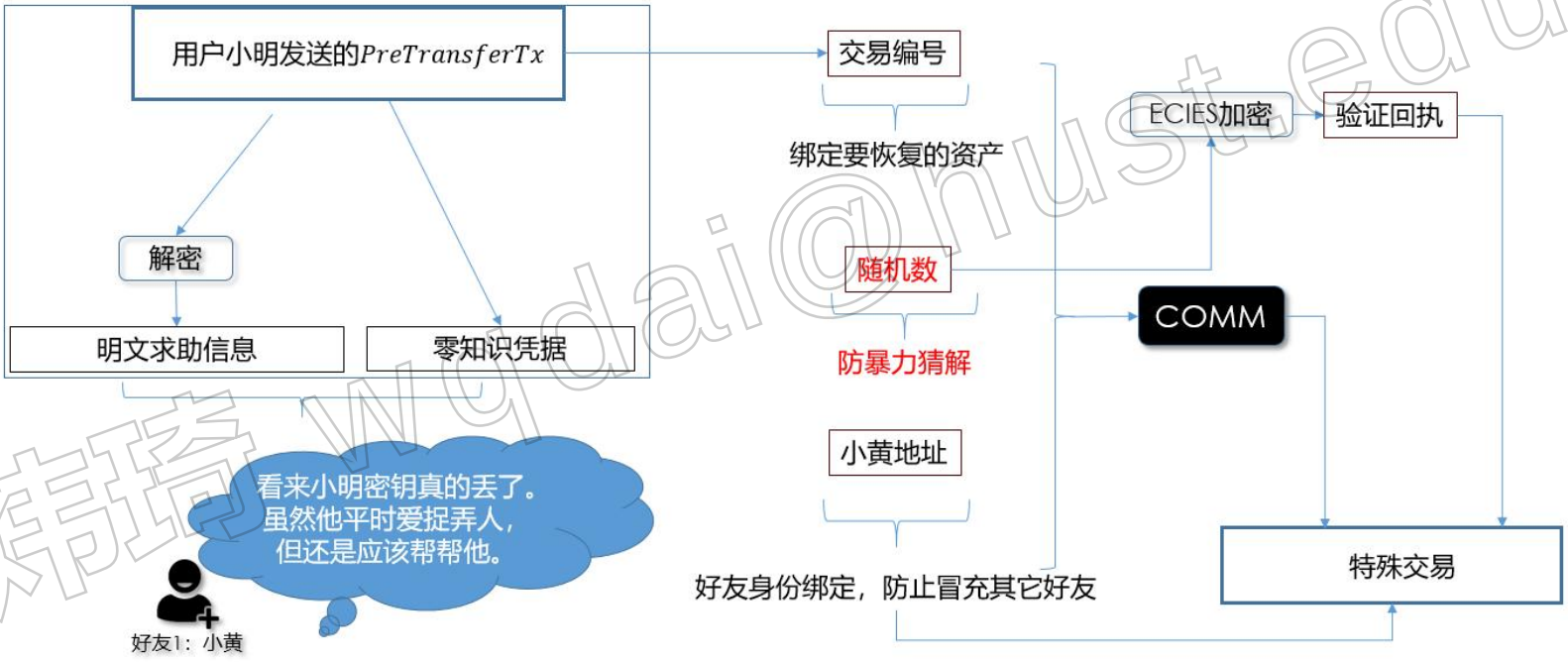
基于零知识证明的数字资产恢复方案

2.预恢复阶段：用户证明自己知道关键恢复信息，并向好友发出秘密求助信息



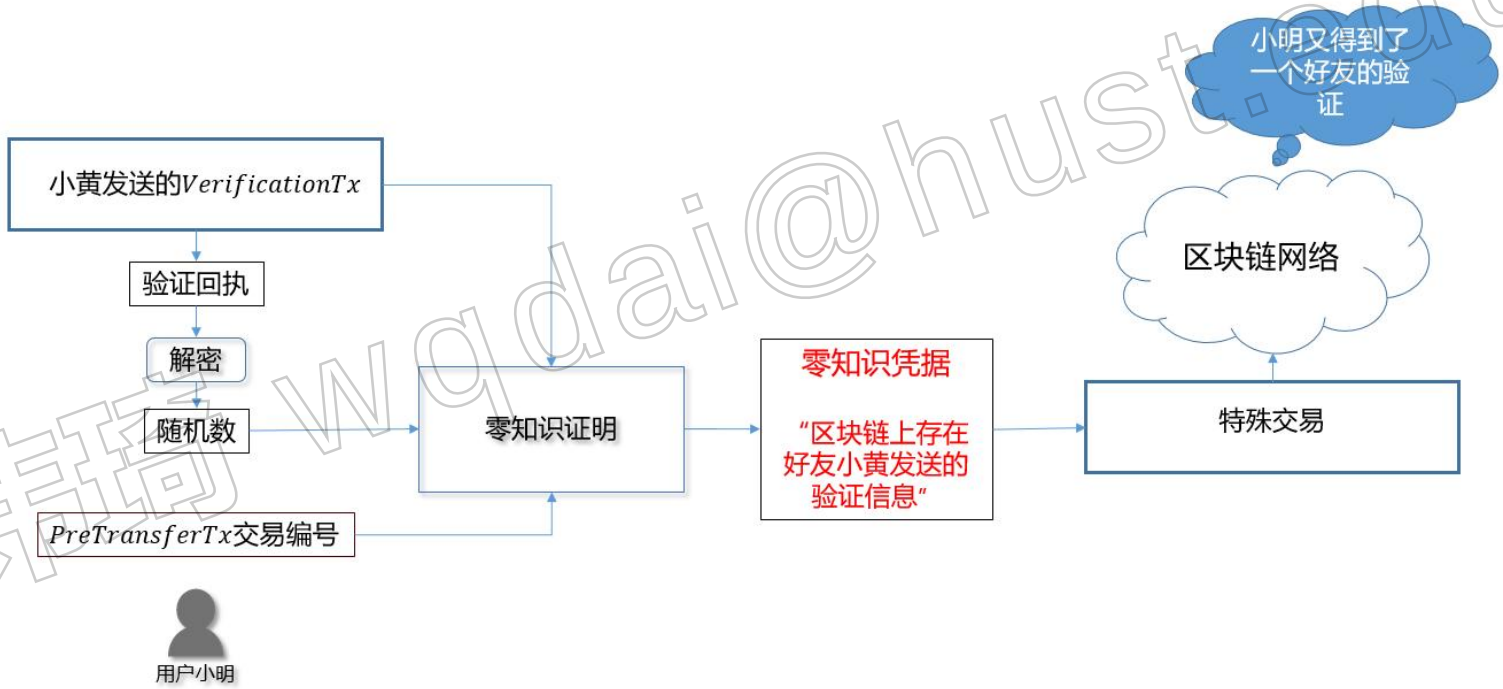
基于零知识证明的数字资产恢复方案

3.好友验证阶段：好友向链上发送一个验证信息，对用户身份进行确认



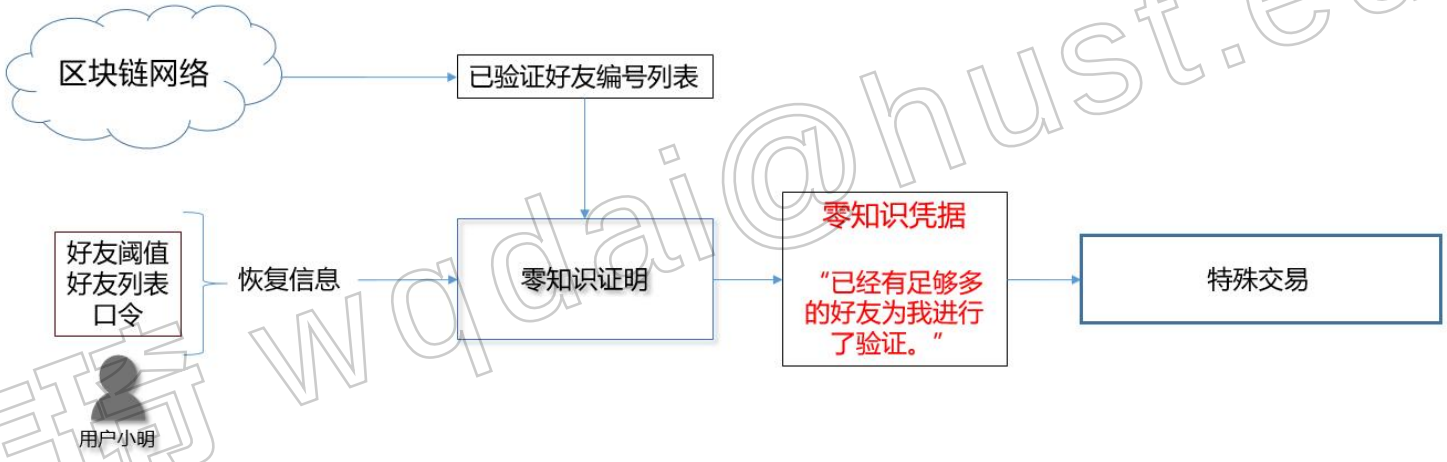
基于零知识证明的数字资产恢复方案

4.收集验证阶段：用户向所有节点证明某个好友已经进行验证



基于零知识证明的数字资产恢复方案

4.收集验证阶段：用户向所有节点证明有足够的好友进行验证

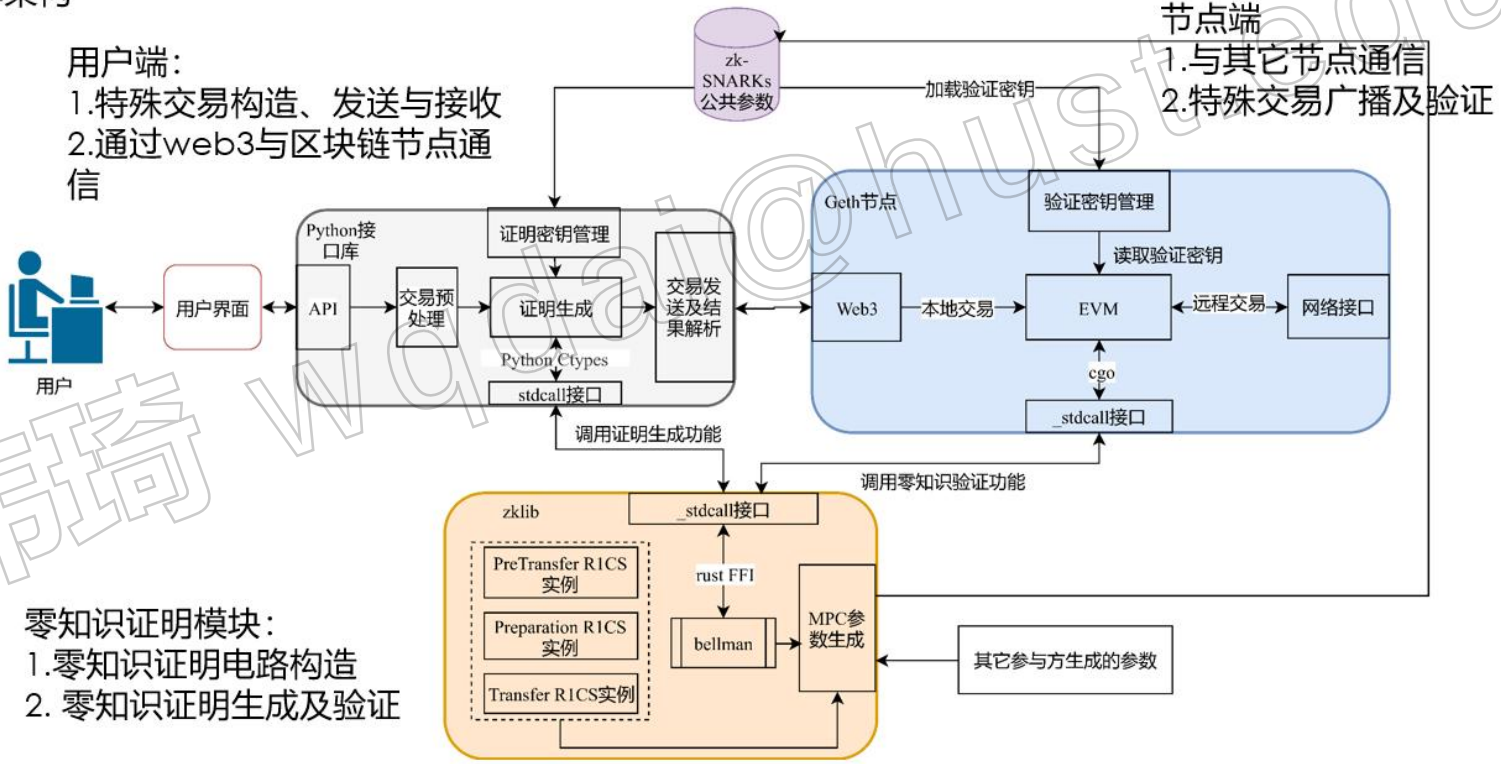


全程没有泄露恢复信息，也没有暴露用户的任何社交关系。

基于零知识证明的数字资产恢复方案

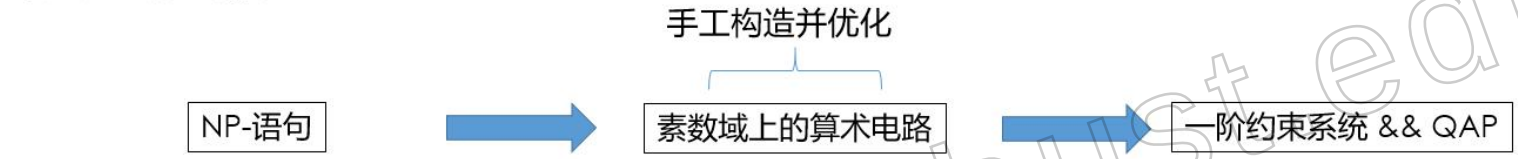
- 方案实现

整体架构



基于零知识证明的数字资产恢复方案

- 零知识证明电路构造

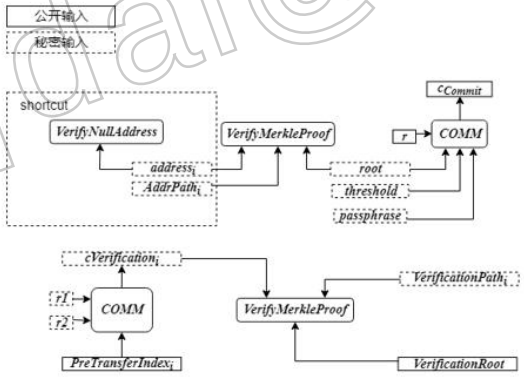


自然语言

“在区块链上存在某个VerificationTx对我的身份进行了验证。”

形式化描述

公开输入： $i, VerificationRoot, C_{Commit}, PreTransferIndex, R$
秘密输入：$Passphrase, Threshold, address_i,$ $AddressPath_i, C_{Verification_i}, VerificationPath_i, R1, R2$
内容：记$AddressRoot$为根据 $AddressPath_i$以及 $address_i$计算出的默克尔树树根。满足以下约束： 1. $C_{Commit} = COMM_R(AddressRoot \parallel Passphrase \parallel Threshold)$ 2. $address_i = NUL$，或者 $\begin{cases} C_{Verification_i} = COMM_{R2}(COMM_{R1}(PreTransferIndex \\ C_{Verification_i}$和$VerificationPath_i$在以$VerificationRoot$为

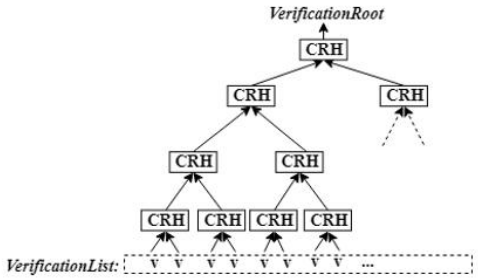


$$\left(\sum_{i=0}^{N} \boldsymbol{a}_{i,j} z_i\right) \cdot \left(\sum_{i=0}^{N} \boldsymbol{b}_{i,j} z_i\right) = \sum_{i=0}^{N} \boldsymbol{c}_{i,j} z_i, \quad j \in [M]$$

$$\left(\sum_{i=0}^{N} \boldsymbol{A}_i(X) z_i\right) \cdot \left(\sum_{i=0}^{N} \boldsymbol{B}_i(X) z_i\right) = \sum_{i=0}^{N} \boldsymbol{C}_i(X) z_i + \left(\sum_{i=0}^{M-2} h_i X^i\right) \cdot Z_D(X)$$

$$Z_D = \prod_{\alpha \in D} (X - \alpha)$$

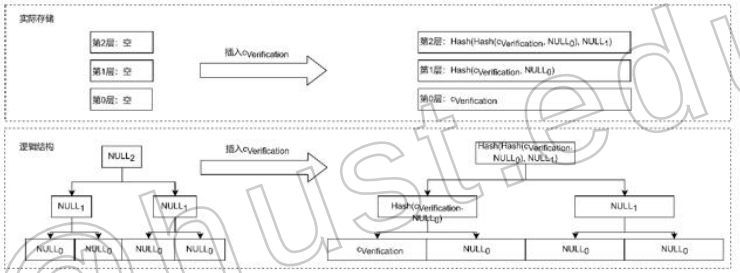
基于零知识证明的数字资产恢复方案



默克尔树存储好友列表及验证列表

零知识证明时间：$O(n) \rightarrow O(\log(n))$

零知识验证时间：$4800ms \rightarrow 400ms$



压缩默克尔树，合并重复节点

空间占用：$O(2^n) \rightarrow O(n)$

256-bit Vector → $x \in F_p$

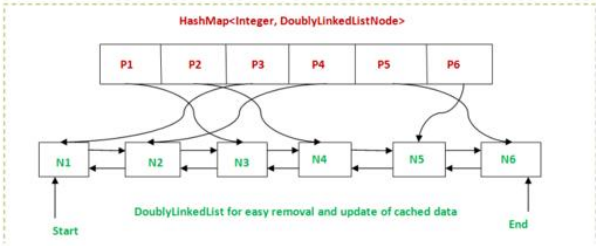
BN256 → BLS12-381

SHA256 → Pedersen Hash

压缩输入、更快的椭圆曲线、对算术电路
更友好的哈希函数

证明时间：$300s \rightarrow 1.5s$

内存占用：$3GB \rightarrow 40MB$



增加零知识证明LRU缓存

验证时间：$400ms \rightarrow 17ms$

基于零知识证明的数字资产恢复方案

❖ 测试

实验设置

节点配置：测试机器CPU为4核的Intel® Core™ i5-6300HQ CPU,主频为2.30GHz，内存为12GB。

网络配置：测试节点个数为4个，均运行在docker容器中

软件	版本
操作系统	Ubuntu18.04，内核版本5.0.7
Python	3.6.7
Golang	1.11
rustc	1.33.0-nightly
Geth	Github commit 38cce9ac333d674616047be14c270d7cfbd43641
Solidity	Solc版本0.4.25

测试方案

1.功能测试

测试1: 正常的恢复流程

测试2: 好友验证人数不足

测试3: 关键恢复信息不正确

2.性能测试

证明时间

验证时间

消耗的交易费用

基于零知识证明的数字资产恢复方案

❖ 测试结果

功能测试

```
before transfer:
old account balance: 192725849619000000000 wei
new account balance: 99998575800000000000 wei


transfer log:  {"amount": 1000000000000000000}

after transfer:
old account balance: 109788514106000000000 wei
new account balance: 199998411313000000000 wei

=========== all tests passed ============
```

```
send verification tx: trusting 0xE418DbBB73b84c8A0E59861dff23eAa44C0258bb as 0xBe949E848c145e4F00E90512c9
E8Ad4081074Dbb's new address
verification #1 content:
{
  "sender": "0x7C7ce408f8e126e31877870777e5866090831597",
  "verification": "211849c4ea914c130fbce2e659ffe401a17660f2df5d6d13dda6b3870218de1e",
  "pre_transfer_commitment": "77509e409c4f5427be3e062a71621e6c2f462a2635a2a6e2dce3a28c2d698f0f",
  "nonce": "ee69cd8475e58cc67f58550cf25a17609ba510c8b56196193567a82817bf8400"
}

preparation-tx for friend #1 sent
current verified friends(shuffled_indexes): [13]


before transfer:
old account balance: 238846969869000000000 wei
new account balance: 99999046142000000000 wei

generating transfer_proof...
thread '<unnamed>' panicked at 'assertion failed: groth16::verify_proof(&groth16::prepare_verifying_key(&
proof_params.vk),
                &proof, &inputs).unwrap()', src/lib.rs:299:5
```

测试1: 正常的恢复流程→恢复成功

测试2：验证好友人数不足→ 拒绝恢复，恢复失败

测试2：恢复信息不正确 → 拒绝恢复，恢复失败

性能测试

语句	单机公共参数生成时间	单机验证时间	单机证明时间
$statement_{PreTrasnfer}$	6.460s	14.711ms	0.744s
$statement_{Prepartion}$	14.517s	15.344ms	1.445s
$statement_{Transfer}$	16.418s	17.376ms	1.056s

交易类型	Gas（交易费用）消耗
$CommitTx$	132399
$PreTransferTx$	877605
$VerificationTx$	1148747
$PrepartionTx$	1490023
$TransferTx$	424423

验证时间**短**<18ms　　证明时间**短**<1.5s　　Gas消耗**低**