

零知识证明在区块链应用中的 安全和隐私问题



彭峙酿 360 高级安全专家
2019区块链开发者大会

大纲

区块链和ZKP介绍

zkSNARKs

零知识证明在区块链应用中的安全和隐私问题

- 实现漏洞

- 信任风险

- 信息泄露

- 密码方案风险

- 其他风险

总结

比特币的隐私问题

比特币：去中心化数字货币

防止双花：广播交易到公开账本（区块链）

匿名性？

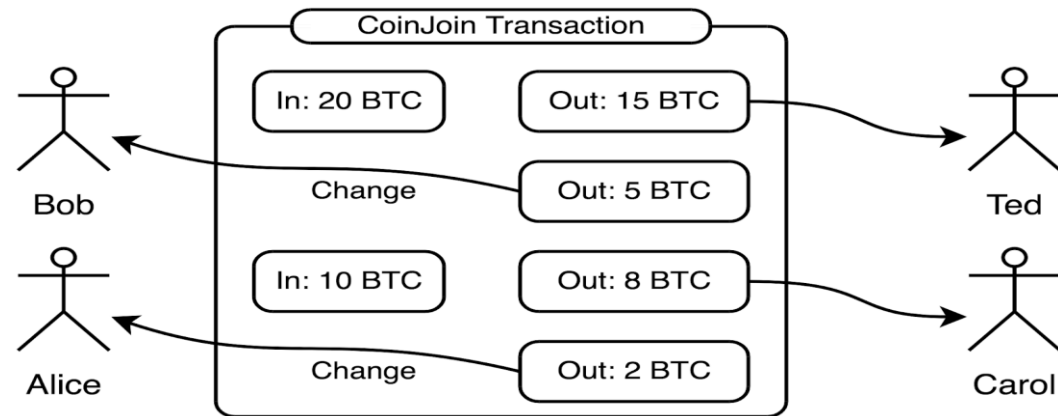
假名≠匿名

隐私问题：


个人交易记录/账户余额/商家现金流

钱的等价性：

黑钱/收藏币



隐私 VS 可审计性




From	Alice
To	Bob
Amount	1

From	Scrooge
To	Donald
Amount	2

...	...
...	...
...	...

From	Bob
To	Eve
Amount	1

保护隐私？ → 数据加密



From	Enc(A)
To	Enc(B)
Amount	Enc(1)

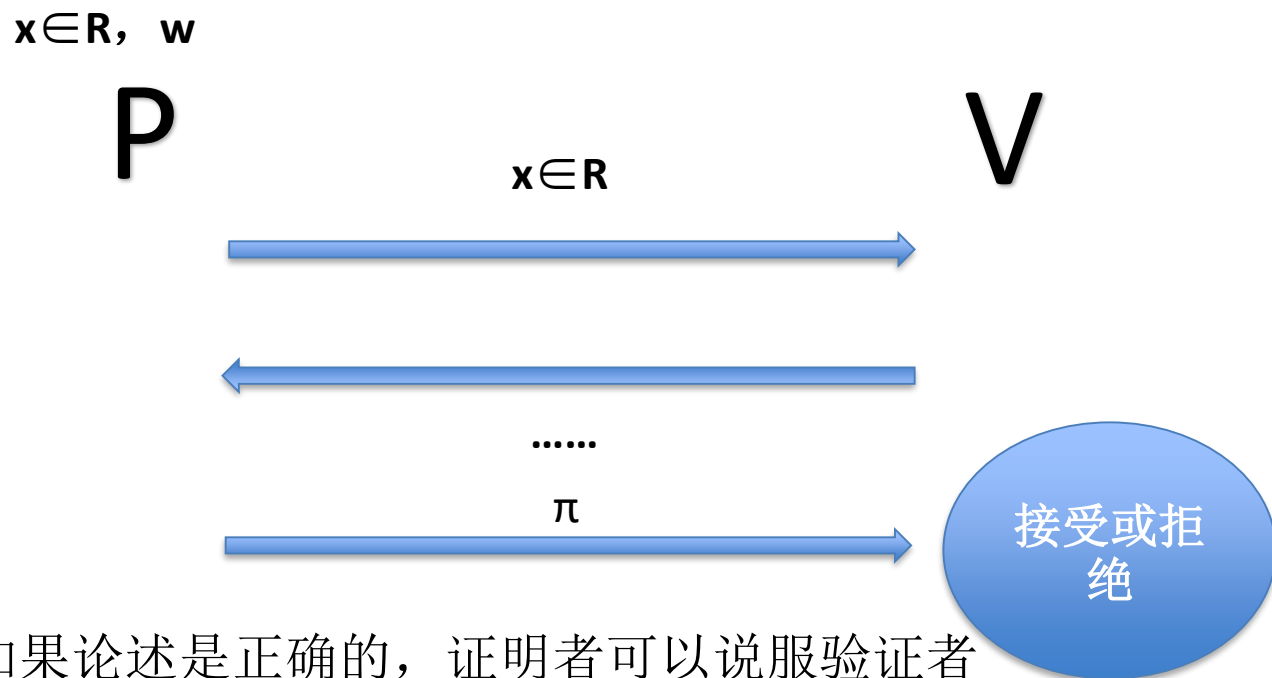
From	Enc(S)
To	Enc(D)
Amount	Enc(2)

...	...
...	...
...	...

From	Enc(B)
To	Enc(E)
Amount	Enc(1)

隐私与可审计性存在冲突

零知识证明



完备性: 如果论述是正确的, 证明者可以说服验证者

正确性: 如果论述是错误的, 证明者无法说服验证者 (可忽略概率说服)

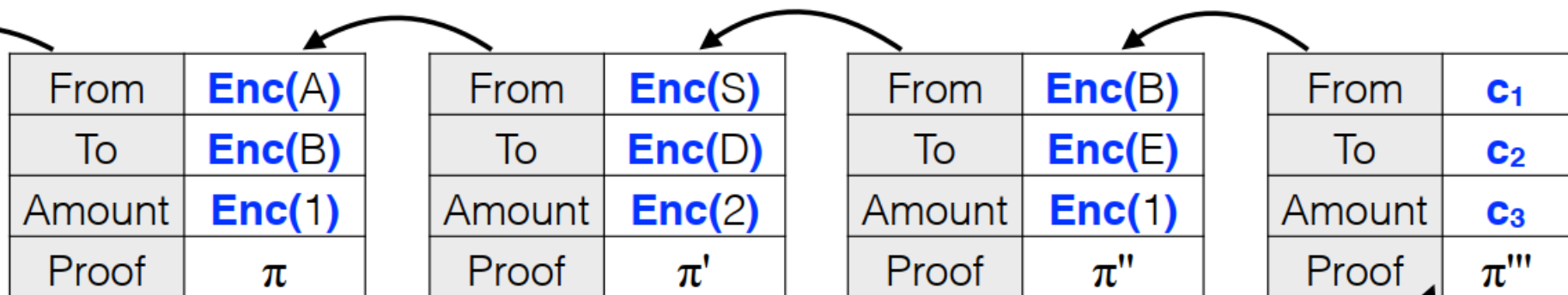
零知识性: 验证者除了论述是正确的之外, 无法获得任何其他信息

知识证明:

非交互式零知识证明(NIZK)

如果因式分解困难, 对任意NP语言都存在NIZK

零知识证明



3个密文： c_1 c_2 c_3

c_1 :发送者账户 c_2 :接收者账户 c_3 :转账金额
满足一定的规则（某个函数关系）
发送的钱没有被双花

证明 π ： 非交互式零知识证明， 以上要求满足

zkSNARKs

区块链上有额外的性能要求

zkSNARKs

Succint

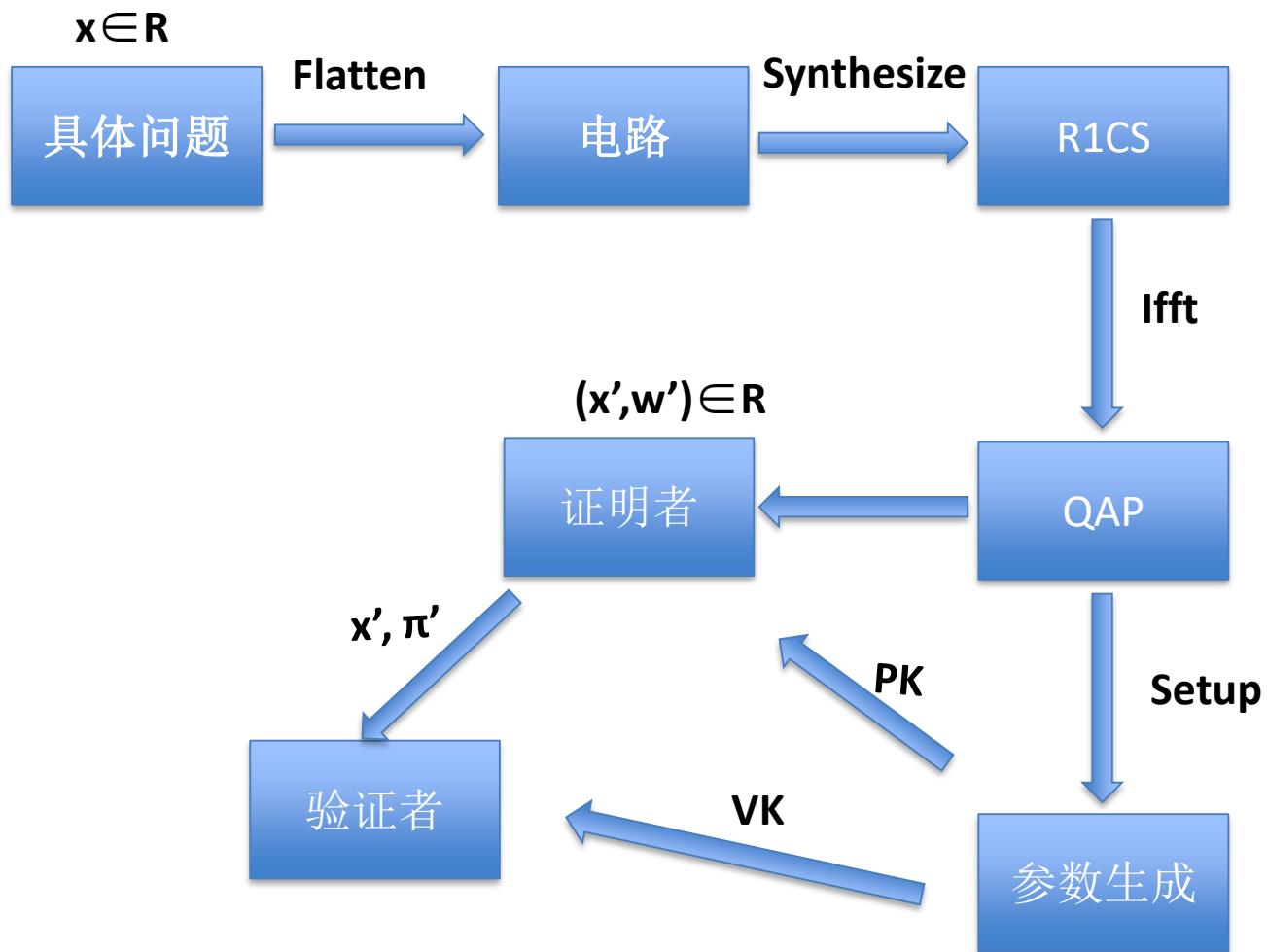
Non-interactive

Argument

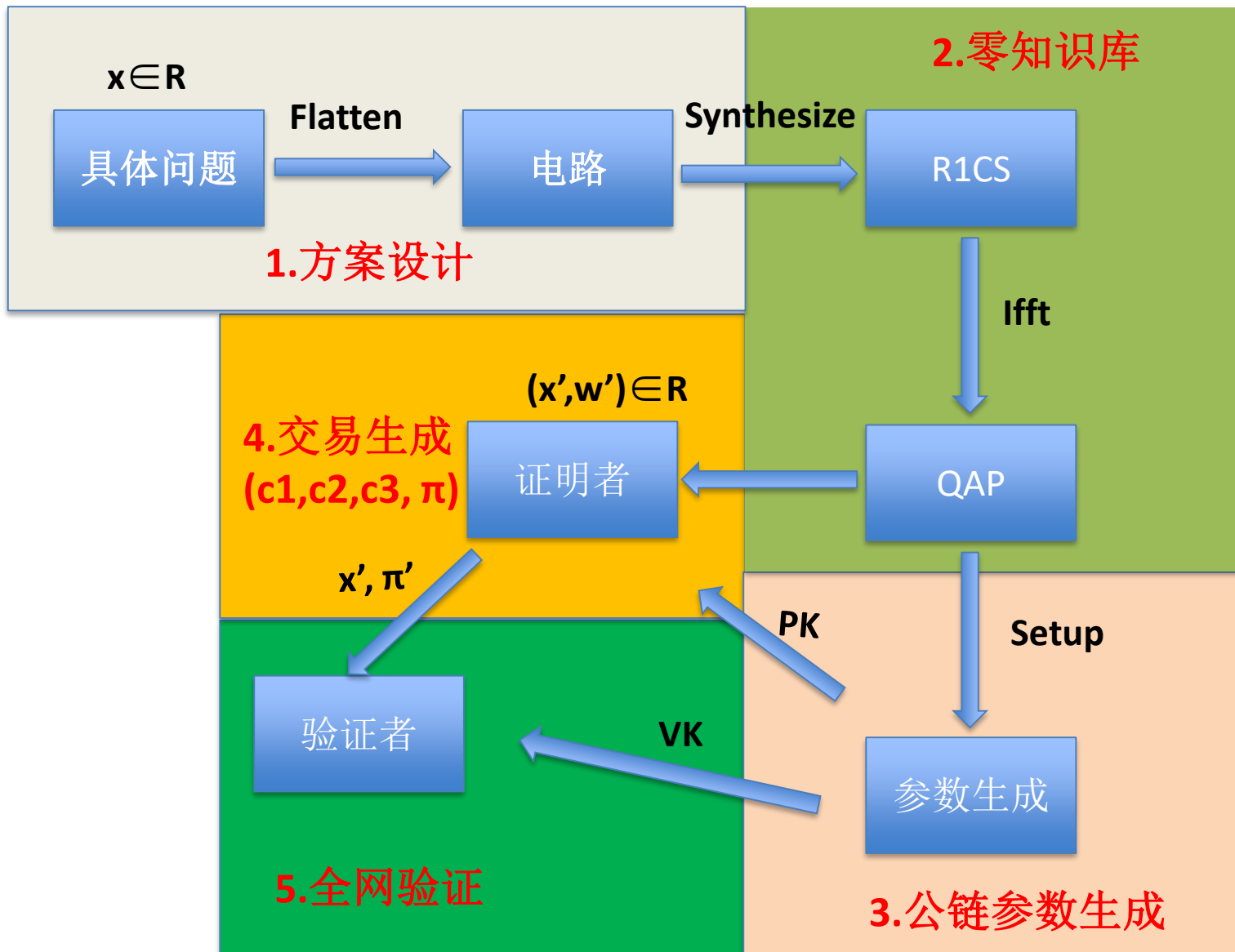
of Knowledge

Zero-knowledge

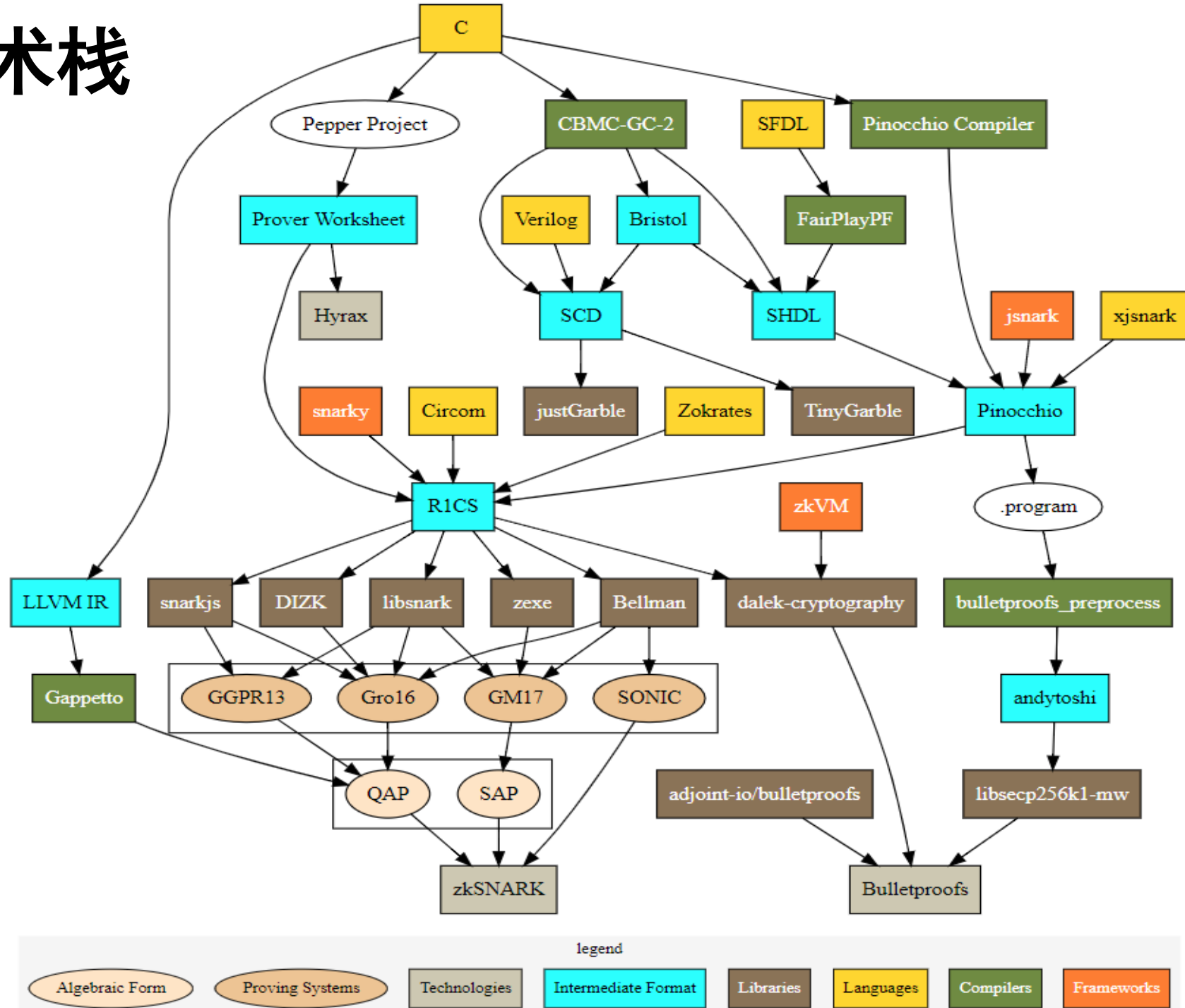
zkSNARKs



区块链上zkSNARKs的应用流程



技术栈



零知识证明在区块链应用中的 安全和隐私问题

实现漏洞

信任风险

信息泄露

密码方案风险

其他风险

实现漏洞

实现中的漏洞

逻辑漏洞

电路设计复杂

应用层逻辑

密码实现漏洞

新的密码原件

电路设计

电路设计非常复杂

大量密码算法实现

大量约束、大量优化技巧

审计需要高超的密码技巧

以zcash为例

Zcash屏蔽交易

输入

锚 rt

唯一值 nf

根节点rt

使用签名

值承诺 cv

证明

输出

输出承诺 cmu

临时密钥 epk

密文1

密文2

值承诺 cv

证明

输入

输出

.....

.....

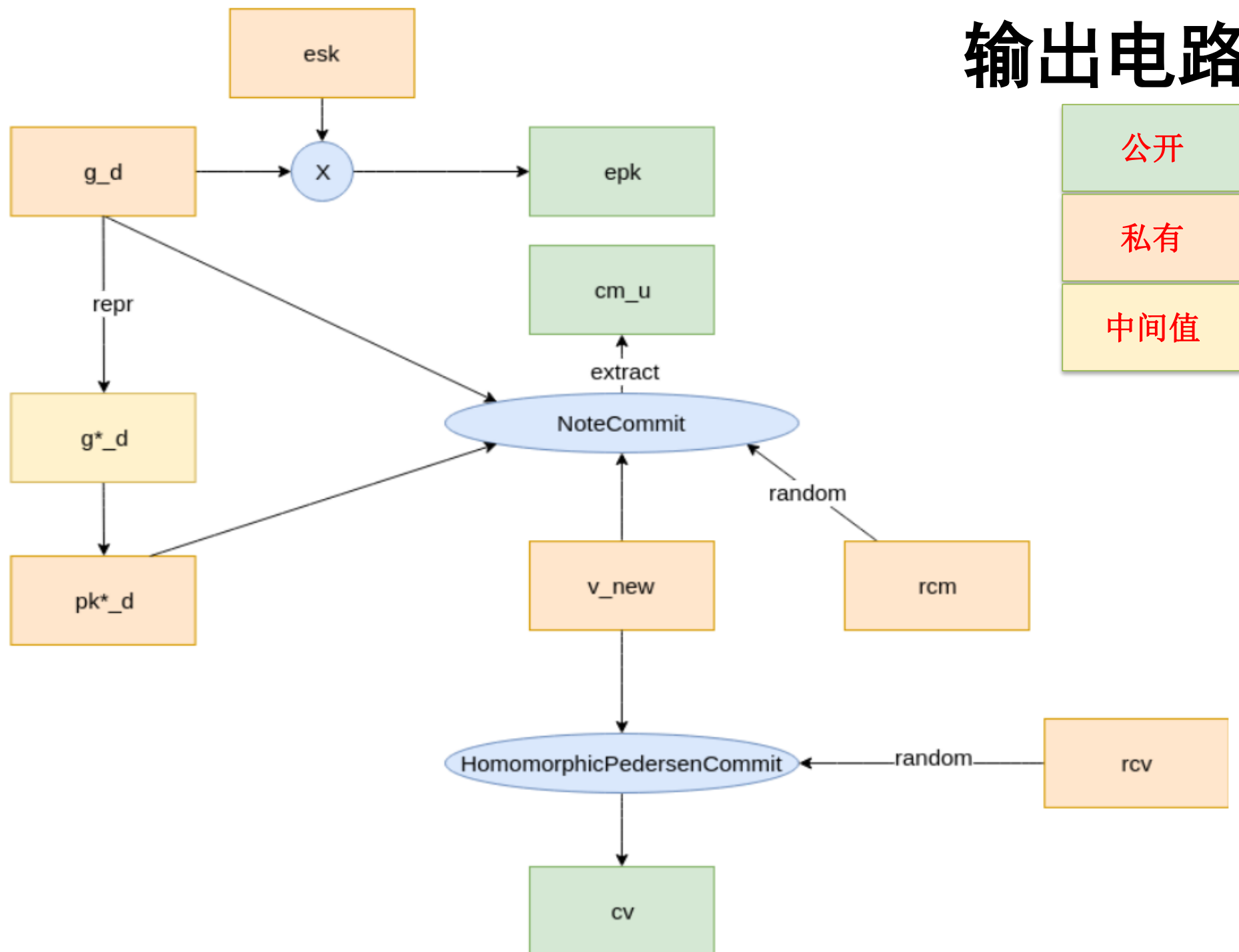
输入

输出

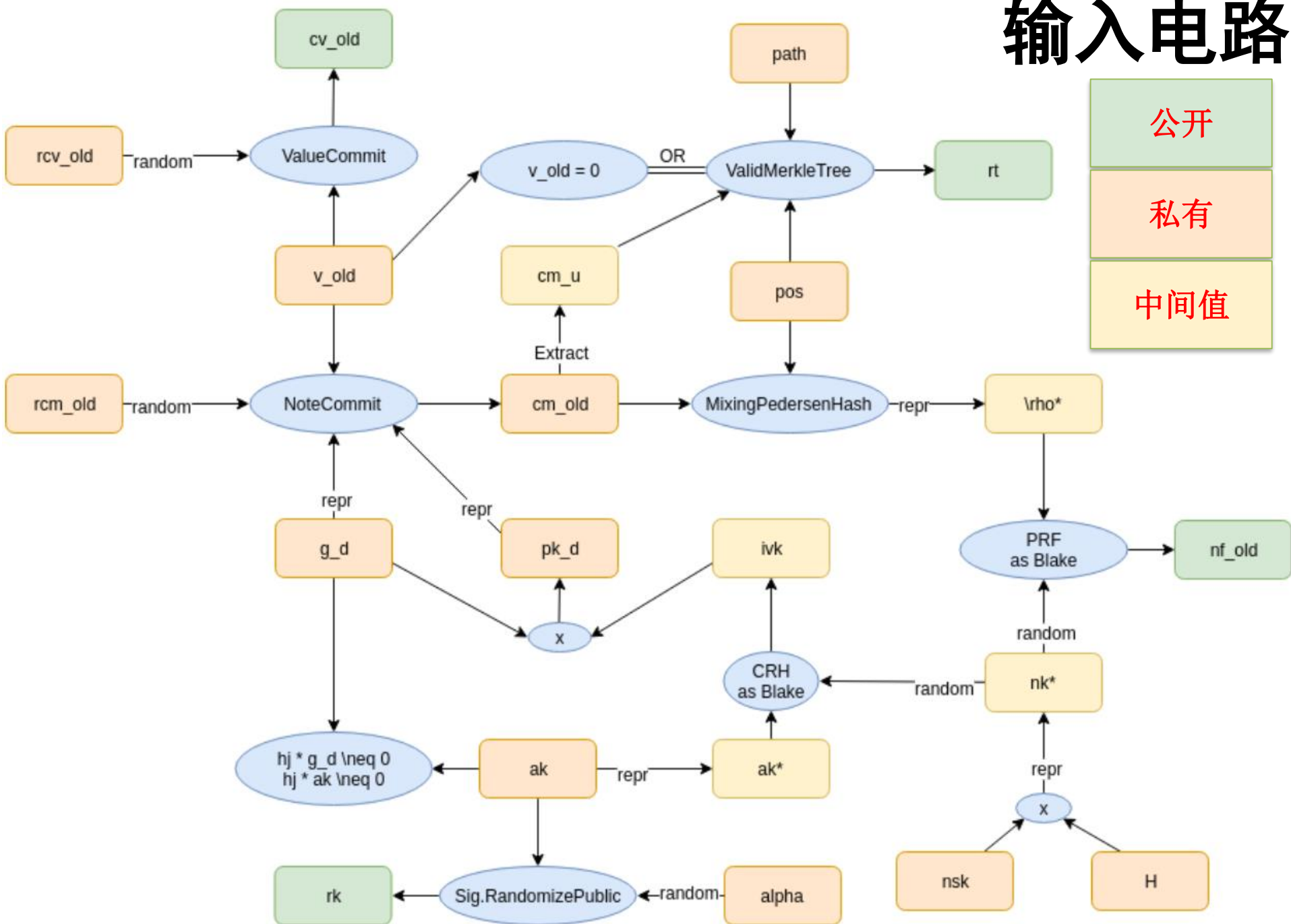
绑定签名

证明输入的总额与输出的总额相等

输出电路



输入电路



Zcash电路问题

Faerie Gold攻击

原Zcash电路：rho选择不受限制，攻击者可选择相同的rho。造成接收者无法使用收到的钱。

现Zcash电路：rho由CRHF生成。

标准与实现不一致

Version 2019.0.1 [Overwinter+Sapling]

Page 139: $cm^{\{old\}}$ 与cm未绑定，恶意攻击者可对相同note选择多个 $cm^{\{old\}}$ ，产生多个nf，造成双花

LibrustZcash中实现没问题

电路与非电路实现不一致

密码实现安全问题

LibSNARK:

Pinocchio协议的修改版实现

无可证明安全

安全性依赖于一个缺乏证据的引理

<https://eprint.iacr.org/2015/437.pdf>

R1CS-to-QAP 规约漏洞

只有QAP中多项式是线性独立的，才能保证soundness
增加冗余约束修复问题

应用逻辑安全问题

应用开发者调用ZKP库来实现ZKP应用

对底层ZKP缺乏足够理解，代码容易产生安全漏洞

Semaphore双花问题

未限定nullifier长度，造成双花

Vulnerability allowing double spend #16

 Closed poma opened this issue on 26 Jul · 2 comments



poma commented on 26 Jul • edited ▼



Looks like in [Semaphore.sol#L83](#) we don't check that nullifier length is less than field modulus. So `nullifier_hash + 21888242871839275222246405745257275088548364400416034343698204186575808495617` will also pass snark proof verification if it fits into uint256, allowing double spend.

Example of 2 transactions:

<https://kovan.etherscan.io/tx/0x5e8bf35ad76a086b98698f9d20bd7b6397ccc90aa6f85c1c5debc0262be5458a>


<https://kovan.etherscan.io/tx/0x9a47cc8daec9d0a5e9a860ada77730190124f9864a5917dcb8f41773d94cf c1a>



21

Tron nullifier双花

未校验一笔交易多个输入的nullifier是否不同，造成双花

src/main/java/org/tron/core/actuator/ShieldedTransferActuator.java 

```
@@ -103,9 +103,14 @@ private void executeTransparentTo(byte[] toAddress, long amount) throws Contract  
    }
```

```
    //record shielded transaction data.
```


```
- private void executeShielded(List<SpendDescription> spends, List<ReceiveDescription> receives) {  
+ private void executeShielded(List<SpendDescription> spends, List<ReceiveDescription> receives)  
+     throws ContractExeException {  
    //handle spends  
    for (SpendDescription spend : spends) {  
+        if (dbManager.getNullfierStore().has(  
+            new BytesCapsule(spend.getNullifier().toByteArray()).getData())) {  
+            throw new ContractExeException("double spend");  
+        }  
        dbManager.getNullfierStore().put(new BytesCapsule(spend.getNullifier().toByteArray()));  
    }
```


Tron零知识参数未校验

未校验参数直接使用Librustzcash，造成多种安全问题(如双花)

itHub, Inc. [US] | github.com/tronprotocol/java-tron/commit/c4729f1e7131e81786f95f7d3694ff6bed87749e?diff=unified


Add constraints on some variables.



 develop (#2352)

 skipjack8 committed on 4 Jul

1 parent [2653e1d](#)

commit [c4729f1e7131e81](#)

 Showing 2 changed files with 31 additions and 7 deletions.

▼ 35  src/main/java/org/tron/common/zksnark/LibrustzcashParam.java 

@@ -49,6 +49,18 @@ public static void validVoucherPath(byte[] voucherPath) throws ZksnarkException

49 49 }

50 50 }

51 51

52 + public static void validValueParams(long value) throws ZksnarkException {

53 + if(value < 0){

54 + throw new ZksnarkException("Value should be non-negative.");

55 + }

56 + }

57 +

58 + public static void validPositionParams(long value) throws ZksnarkException {

59 + if(value < 0){

60 + throw new ZksnarkException("Position should be non-negative.");

61 + }

62 + }

63 +

52 64 interface ValidParam {

53 65

信任风险

可信setup问题

实用NIZK生成思路:

提前生成验证者挑战

加密该挑战、丢弃明文挑战（有毒废料）

利用有毒废料伪造证明:

通过验证的本质是要构造A、B、C满足:

$$A \cdot B = \alpha \cdot \beta + \frac{\sum_{i=0}^n a_i (\beta u_i(x) + \alpha v_i(x) + w_i(x))}{\gamma} \cdot \gamma + C \cdot \delta$$

知道有毒废料可伪造任意证明

不可检测的后门

MPC1 : Power of tau

The screenshot shows the GitHub repository page for ZcashFoundation / powersoftau-attestations. The repository has 7 watches, 16 stars, and 17 forks. The main branch is master. The file list includes README.md (Updating mailing list URLs, 2 years ago) and report.asc (Andrew Miller's attestation, 2 years ago). The README.md content is displayed below the file list.

Branch: master ▾ powersoftau-attestations / 0001 /

Create new file Upload files Find file History

pera Updating mailing list URLs Latest commit 79ee8b1 on 3 Jan 2018

..

README.md	Updating mailing list URLs	2 years ago
report.asc	Andrew Miller's attestation.	2 years ago

README.md

Andrew Miller

- Assistant Professor at the University of Illinois, Urbana-Champaign
- Chair of the Zcash Foundation Board
- <https://soc1024.com/>
- Mailing list post: <https://lists.z.cash.foundation/pipermail/zapps-wg/2017/000007.html>
- See `./report.asc` for the signed attestation.

Response file:

- <https://powersoftau-transcript.s3-us-west-2.amazonaws.com/15729e0edc4201dc5ee6241437d926f614cb4214ff1b9c6fbd73daf401639f7a4238cf04bc94edac9f2ad037003daab9a4408ba7c62a4413dc2a0ddd683bd719>
- https://s3.amazonaws.com/socrates1024_a/response-2017-11-10-amiller

生成参数(可通用给任何电路)

公开可验证、至少一人诚实、密码学安全

MPC2: Sapling MPC

github.com/zcash-hackworks/sapling-mpc/wiki

Participants:

- Sean Bowe
 - 4caed5dfc4fd6959b5181c4adc31fe013c58b438c360d30af5a40fe02cc19a9ad56344679524e07fbad6166cc44ee35e1f6e70e5267b46ce93d059cad3ab7ed5
- Ian Munoz
 - dd8ca189564354d5c9405b16b32f8dd0ed4644e4184e9fbc9feaccebd732ad8276033318b0b70fa5b1f59f49ce3ed42f3f575b9e6219c11d3e28a82ed24ca990
- Jason Davies
 - 851b81478d1ce92c1f7116c8f355c999447e36a151b97f36ff8d5cbf9b98a5b00431b2a087da2038d201669dd3a8eba8bbd2d92ea8e348b92e1b51b09fe5bb5f
- Larry Ruane
 - 82641343d6840bcb218a7cef8f46d6f6678fff3692e8ec6b9bfc44245737d0c047dd441fd3608ea1463302a8c9521dcd716df3b114741bcd4da5add68bf35d3
- Gabor Losonci
 - be9fe35787982dda14b81a796dbab4a5ecfe1bf2d56c6b0186e4552b5a942d61a5378c4c903aa3087226b7bdf0fcf7808eaa6a2b1a1ae0d42793bb9836718a23
- Michael Perklin
 - ec782c5b9b33d98750012e3d1d62f34b490acd2d392a30629e98596198143e414536f30bd9414ee01dc26fc7d132ca245e400c738073a68cad9e439a5acb83a6
- Alexey Ermishkin
 - ce07ea6e537f106c0b9b70a5aff9dcc8105dfc1f5f69e32d272f7749e2b41ba728e3ef860d57ab933bdfb458c675bd5cd46c08429a1229dc8b21dc8e95f5815b

电路相关、
两个MPC阶段缺一不可

MPC信任风险

Ethereum、Tron都在进行MPC

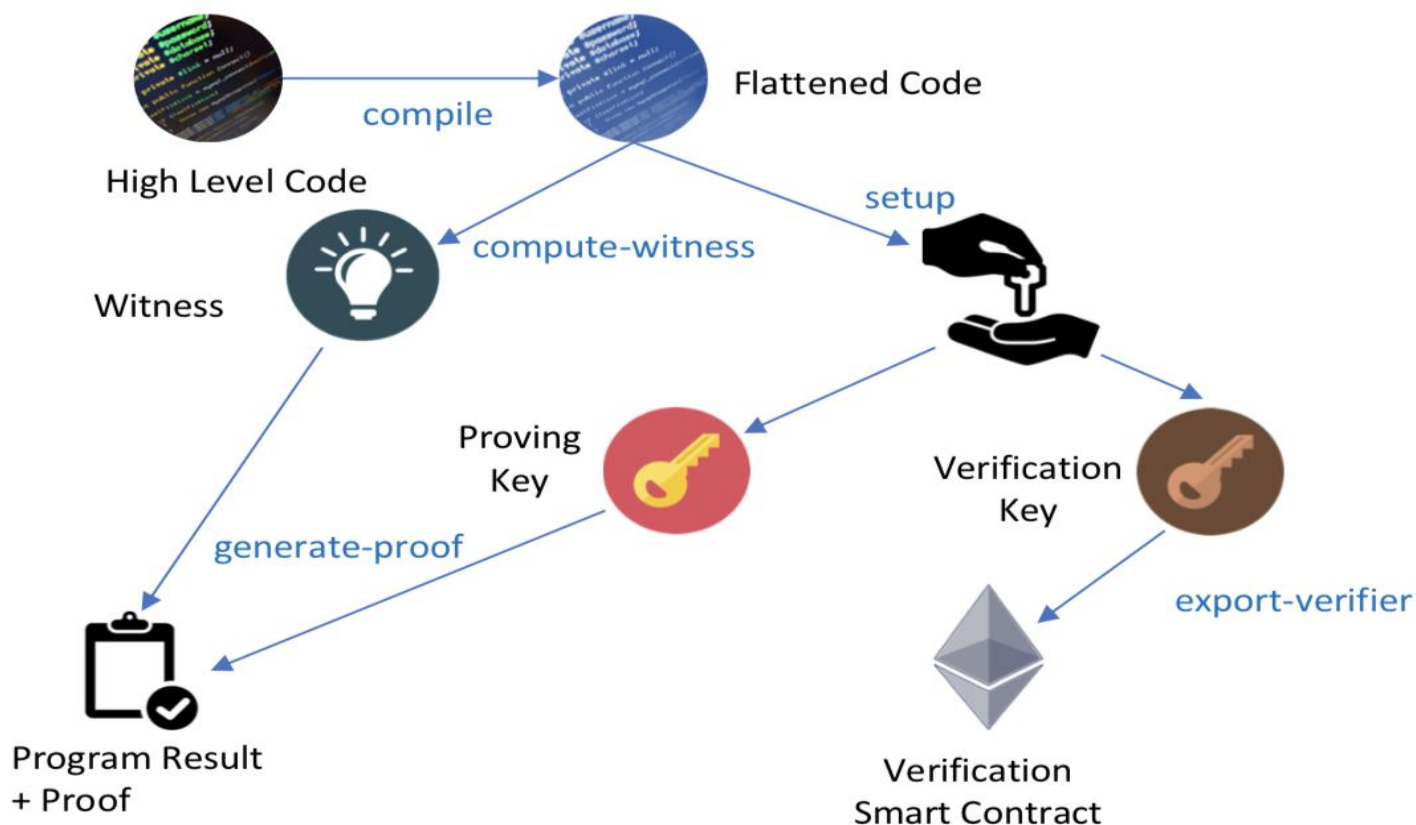
但任然有很多项目并没有真正可信的setup

MPC过程通常为黑盒

如果你没有参与，你不能100%确认安全

还没有考虑软件实现问题

例子：ZoKrates



ZoKrates: 零知识证明智能合约编译器

参数生成由合约方控制

难以审计

证明伪造、信息泄露

信息泄露

An Empirical Analysis of Anonymity in Zcash

George Kappos, Haaroon Yousaf, Mary Maller, and Sarah Meiklejohn

On the linkability of Zcash transactions s.meiklejohn@ucl.ac.uk

Jeffrey Quesnelle
University of Michigan-Dearborn

An Empirical Analysis of Traceability in the Monero Blockchain

Malte Möser, Kyle Soska, Ethan Heilman, Kevin Lee, Henry Heffan, Shashvat Srivastava,
Kyle Hogan, Jason Hennessey, Andrew Miller, Arvind Narayanan, Nicolas Christin

PETS 2018: The 18th Privacy Enhancing Technologies Symposium

轻客户端隐私矛盾

比特币：

SPV+布隆过滤器 → 普通用户的隐私保护

零知识证明Token：

解密后才能知道交易归属

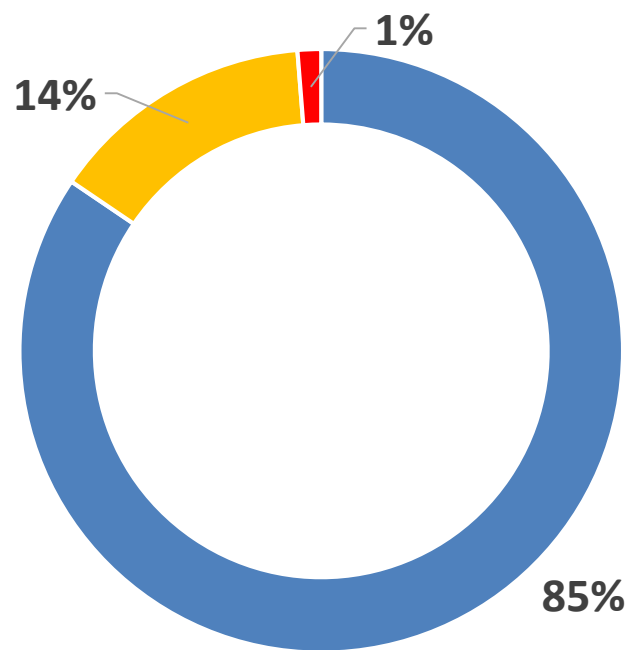
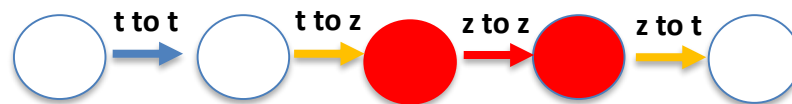
解密交易：轻客户端需将解密密钥交给全节点

隐私与易用性存在矛盾

Tron的例子

```
@Override
public void scanNoteByIvk(GrpcAPI.IvkDecryptParameters request,
    StreamObserver<GrpcAPI.DecryptNotes> responseObserver) {
    long startNum = request.getStartBlockIndex();
    long endNum = request.getEndBlockIndex();
    try {
        DecryptNotes decryptNotes = wallet
            .scanNoteByIvk(startNum, endNum, request.getIvk().toByteArray());
        responseObserver.onNext(decryptNotes);
    } catch (BadItemException | ZksnarkException e) {
        responseObserver.onError(getRuntimeException(e));
    }
    responseObserver.onCompleted();
}
```

Zcash 交易分类

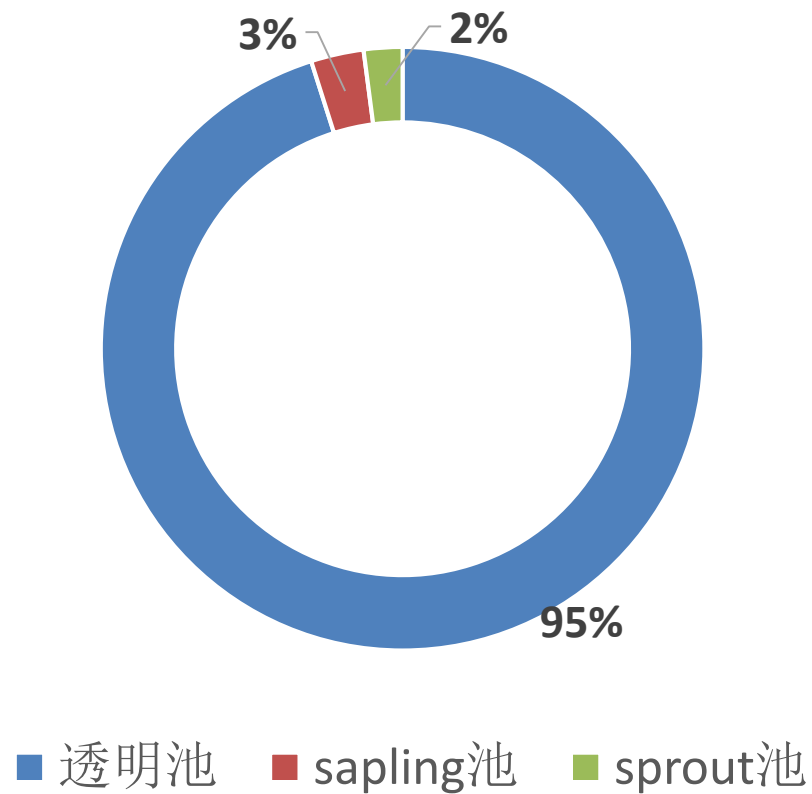


2019-09

■ 透明交易 ■ 部分频闭交易 ■ 完全频闭交易

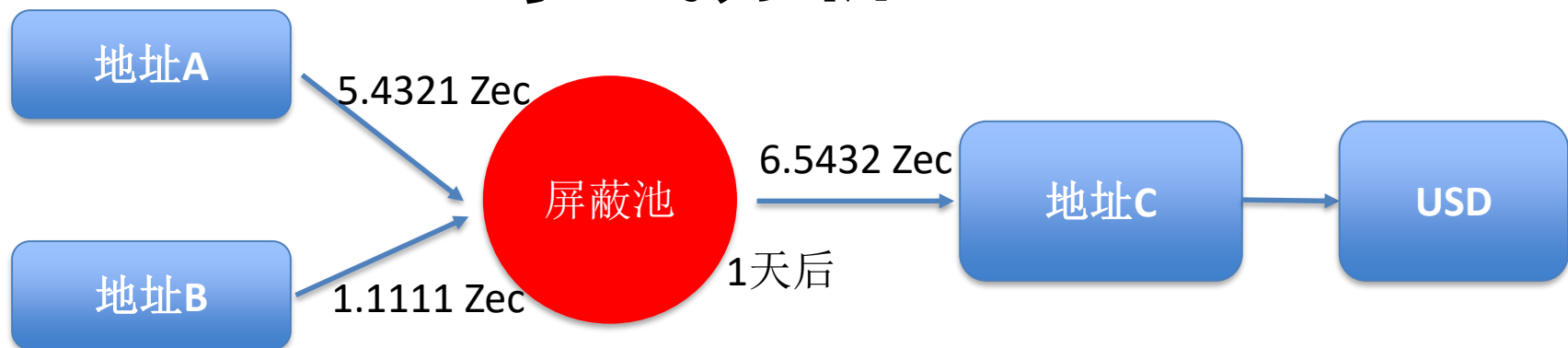
大多数的交易可被追溯

Zcash 货币池



2019-09

经验式分析



信息泄露

用户的使用习惯:

相关的地址: 地址聚类分析

相关的金额: 金额匹配分析

相关的时间: 时间相关分析

案例分析: Shadowbrokers

默认并不安全

因政策原因，交易所仅支持透明交易
轻节点难以进行屏蔽交易
默认行为会泄露信息

隐私应对：

使用新地址

金额不要相同

等待足够长的时间

密码方案风险

密码方案风险

ZKP技术相对教新

16年论文，17年大规模使用
有待时间考研

参数选择和优化比较激进

可证明安全

有的困难问题并不标准
依赖太多安全问题
缺乏足够的审计

Zcash伪造漏洞

CVE-2019-7167

2018年3月1日发现(Ariel Gabizon)，2019年公开
零知识证明密码方案[BCTV14]漏洞
任意人可伪造证明，凭空创造Zcash
影响多个Zcash的分叉项目
没人知道漏洞是否被利用，因为他是零知识的

大概8个月时间才完成修复

改变整个证明方案，升级全网

Zcash官方认为漏洞没有被利用：

很少有人拥有的高水平密码技术发现该漏洞
未发现zcash总额出现明显问题

Zcash伪造漏洞

CVE-2019-7167

[BCTV14]参数生成存在冗余元素

可被利用生成伪造证明

[BCTV14]并不存在可证明安全

原理很简单(阅读分析报告后)

[BCTV14] 升级为[Groth16]

可证明安全

并非第一次出现漏洞

Bryan Parno发现过另一个漏洞

是否会最后一次?

其他风险

完美的零知识？

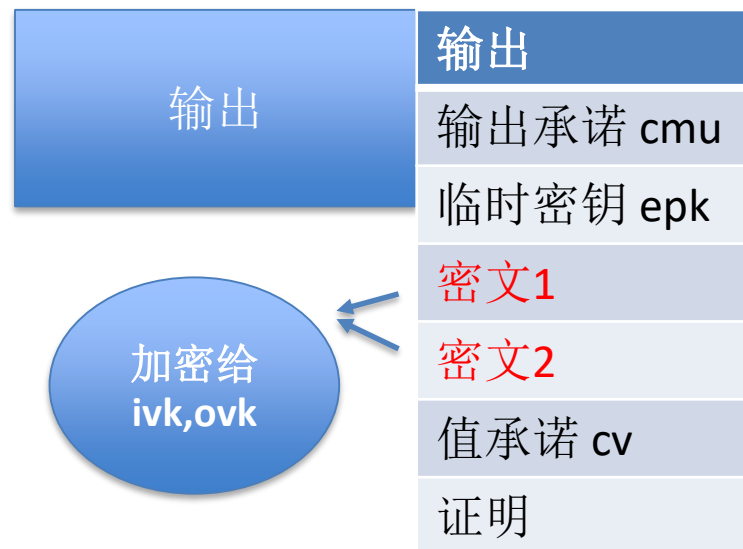
数学上完美的零知识方案

[Groth16]

实际应用会有额外的信息泄露

猜密钥，解密

打破了完美的零知识性



屏蔽地址不可链接性

屏蔽地址不可链接性：

安全性依赖于Jubjub群上的DDH问题

Jubjub群并非常用的ECC群，而是为了ZKP性能特制的
DDH未来可能会被攻破

交易信息在链上不可被抹去

目前不可链接，不代表未来不可链接

侧信道问题

Timing侧信道:

证明过程: 随机选择两个参数 r 和 s , 计算 $\pi = \Pi\sigma = (A, B, C)$

$$A = \alpha + \sum_{i=0}^m a_i u_i(x) + r\delta$$

$$B = \beta + \sum_{i=0}^m a_i v_i(x) + s\delta$$

$$C = \frac{\sum_{i=\ell+1}^m a_i (\beta u_i(x) + \alpha v_i(x) + w_i(x)) + h(x)t(x)}{\delta} + As + rB - rs\delta$$

A、B、C的计算时间，直接与秘密 a_i 相关

其他侧信道: Cache

总结

ZKP应用的安全风险

实现漏洞

信任风险

信息泄露

密码方案风险

其他风险

ZKP是一项新技术

仍然存在很多问题

越来越成熟

Thanks



360
WWW.360.CN

SAFETY FIRST