

第一周作业

1 环签名与零知识证明比较

环签名：这个方式解决了对签名者完全匿名的问题，允许一个成员代表一组人进行签名而不泄漏签名者的信息。

零知识证明：允许证明者和验证者来证明某个提议是真实的，而且无需泄露除了其是真实的之外的任何信息。在密码学货币和区块链中，这通常是指交易信息数据。

环签名本质上是一种工程应用，零知识证明本质上是一种密码学技术。但两者都提高了支付匿名性的保护。零知识证明可达成任意等级强隐私保护,这是环签名无法达到的。环签名依旧需要与其他用户的公钥进行混合,可能会遭遇恶意用户从而暴露隐私,而零知识证明避免了这个问题

环签名的执行效率是与其 decoy 的数目有关联的，也就是说与它的成员数目有关，成员数越多，运行时间，空间存储呈线性增长。零知识证明目前的性能较差，计算复杂度主要来自于底层依赖的椭圆曲线配对运算,运行效率与比较低。

环签名的验证程序 ring_sig.py

环境 ubuntu 16.04 64 python2.7

验证的成员数由 1 到 100，运行结果时间与存储呈线性增长：

```
('成员数','---',签名,'---', 运行时间(微秒))
('1','---', 1211173223857594755206029412869193419210348971700L,'---', '33680')
('2','---', 691272497158154961393271889715361140943734655898L,'---', '24240')
('3','---', 254865827643538308749543751552943251790722241029L,'---', '64301')
('4','---', 654363812916278941124822008808793344415733363157L,'---', '105726')
('5','---', 1345375983086672812494496852841871254539845588007L,'---', '111759')
('6','---', 554660862872180359008314489992228283033137738845L,'---', '100158')
('7','---', 3902413806639282123040295200982441078668544280L,'---', '138593')
('8','---', 714706511469630967269051779686391623786156244023L,'---', '132773')
('9','---', 1460206101762743038423806984241516091142221927632L,'---', '147354')
('10','---', 338663760879732091010580326964579059349628217271L,'---', '226506')
.....
('93','---', 973933033409757588355382538146640436298131109121L,'---', '375584')
('94','---', 1226648796952879948933569891349627530411034036857L,'---', '312810')
('95','---', 111197558574071981536748911016344001920731116642L,'---', '370561')
('96','---', 692208874240998732441126699847183850751372597779L,'---', '406028')
('97','---', 1182477223916958539255330044952742704082015035404L,'---', '471258')
('98','---', 1306489804157969789944532142001579169623949955158L,'---', '351492')
('99','---', 1357219220786762331370496462947601238259666440302L,'---', '425623')
('100','---', 1426143933167032953566813242544670197897465121785L,'---', '439466')
```

零知识验证程序 <https://github.com/howardwu/libsnark-tutorial>

环境 ubuntu 16.04 64

计算复杂度主要来自于底层依赖的椭圆曲线配对运算。

调整测试函数 test_r1cs_gg_ppzksnark(size_t num_constraints, size_t input_size)

对参数 num_constraints, input_size 进行调整。

目前只能运行起来，还有一些理论需要补充,目前卡在 R1CS->QAP

```
cookieswolf@cookieswolf-virtual-machine:~/zkSNARK/libsnark-tutorial/build$ ./src/main
(enter) Call to generate_r1cs_example_with_binary_input [ ] (1537930126.9776s x0.00 from start)
(leave) Call to generate_r1cs_example_with_binary_input [0.0013s x1.00] (1537930126.9789s x0.00 from start)

=====
R1CS GG-ppzksNARK Generator
=====

(enter) Call to r1cs_gg_ppzksnark_generator [ ] (1537930126.9796s x0.00 from start)
(enter) Call to r1cs_constraint_system::swap_AB_if_beneficial [ ] (1537930126.9798s x0.00 from start)
(enter) Estimate densities [ ] (1537930126.9798s x0.00 from start)
* Non-zero A-count (estimate): 492
* Non-zero B-count (estimate): 505
(leave) Estimate densities [0.0000s x1.00] (1537930126.9798s x0.00 from start)
(enter) Perform the swap [ ] (1537930126.9799s x0.00 from start)
(leave) Perform the swap [0.0000s x1.02] (1537930126.9799s x0.00 from start)
(leave) Call to r1cs_constraint_system::swap_AB_if_beneficial [0.0001s x0.99] (1537930126.9799s x0.00 from start)
(enter) Call to r1cs_to_qap_instance_map_with_evaluation [ ] (1537930126.9800s x0.00 from start)
(enter) Compute evaluations of A, B, C, H at t [ ] (1537930126.9801s x0.00 from start)
(leave) Compute evaluations of A, B, C, H at t [0.0059s x1.00] (1537930126.9800s x0.00 from start)
(leave) Call to r1cs_to_qap_instance_map_with_evaluation [0.0061s x1.00] (1537930126.9801s x0.00 from start)
* QAP number of variables: 1100
* QAP pre degree: 1000
* QAP degree: 1152
(enter) Compute query densities [ ] (1537930126.9801s x0.00 from start)
(leave) Compute query densities [0.0000s x1.02] (1537930126.9801s x0.00 from start)
```

2 Bitcoin,Eth,Menero,Zcash,Eos 交易

	交易属性	交易数	块大小
Bitecoin	Pow	容纳 1500-2000 条交易	上限 1M
Eth	Pow	用 gaslimit 来限制	大小不固定
Menero	Pow	以区块大小为上限	自适应区块大小，根据交易量 计算区块大小。最大 2 *最近 100 个区块中位数
Zcash	Pow, zk-SNARK	以区块大小为上限	上限 2M
Eos	DPos	以区块大小为上限	1M，可扩展

不太理解交易属性，如果是交易字段可以列出来。个人认为交易属性就是共识算法