**Department of Electrical Engineering**
**Sharif University of Technology**

**Foundations of Blockchain**
**by: Dr. Mohammad A. Maddah-Ali**
**Fall 1399(2020)**

---

**Problem Set 5**

---

If you have any questions about the problems please contact at this email address : kasraabbaszadeh@gmail.com

**Question 1:** Consider a longest-chain PoS protocol based on"VRF$(R, time, SK) < threshold$" inequality where VRF is a verifiable random function, SK is secret key of the miner and R is randomness. we call this protocol c-correlated when randomness is updated after each c blocks.
( e.g. In 1-correlated R can be the hash of previous block.)

**a)** What is "Nothing at Stake" and how this attack can increase the power of adversary to mine more blocks?

**b)** Assume the protocol is 1-correlated. Let $\lambda_h$ as mining power of honest nodes and $\lambda_a$ as mining power of adversary. Show that the security threshold under the private double spend attack is $\frac{1}{1+e}$ when adversary use "Nothing at Stake" to increase his mining rate. In other word, the length of the attacker's private chain will grow at most at $e\lambda_a$.
(Hint: Refer to the reference[1].)

**c)** We can protect the protocol against NaS attack by increasing c.(why?) What are the drawbacks of large c? Can you find a trade-off here?

**Question 2:** Three parties, A, B, and C, are constantly making pairwise payments and thus design a 3-party payment channel based on the revocable hashed time-lock contracts. At each step, A gets a revocable commitment that it can sign and submit with three outputs, one for B with a value of B's current balance , one for C with a value of C's current balance, and one for A with a value of A's current balance. The last one can be spent 48-hours after the transaction is mined, but either B or C can spend it immediately given a hash preimage initially known only to A (and released by A to invalidate the transaction). Similarly, B and C each gets a corresponding commitment transaction with an output that either of the other two parties can claim given a hash preimage. when one the party's wants to pay other party, first he will reveal his secret x to invalidate the transaction of the last step then he will get a new revocable commitment.

Explain how two colluding parties may be able to steal funds from the third.

**Question 3:** Suppose we have a number n, which we know is either a product of two distinct primes, or a product of three distinct primes. Alice wishes to prove to Bob (in zero knowledge) that it is a product of three primes. Consider the following protocol:

- Bob picks three elements at random (a, b, r) at random from $Z_n$. He computes the three numbers a, b,and $c = ar^2$. He then sends the three numbers a, b, c to Alice in a random order.
- Alice receives ($x_1$, $x_2$, $x_3$), finds two entries that are of the form $x_i = x_j r^2$ for some r (if there are multiple, pick one arbitrarily), and tells Bob $\{1, 2, 3\} - \{i, j\}$. That is, Alice tells Bob which index is not i or j.
- The above is repeated 100log n times. Each time in step 2, if the index sent to Bob corresponds to b, Bob gives Alice a point. If the total number of points Alice gets is more than 90log n, then Bob accepts. Otherwise, he rejects.

  Prove that for all large enough n, if n is a product of three primes, then if Alice and Bob follow the protocol above, with probability at least 2/3 Bob will Accept and if n is a product of two primes, Bob will accept with probability at most 1/3.
  (Hint: Think of the total number of quadratic residues that exist mod an n which is a product of two primes vs an n which is a product of three primes.)

**Question 4:** Let M be an $n * n$ matrix over a field F, and let $\lambda \in F$. Both the prover and verifier know M and $\lambda$. The prover wants to convince the verifier that $\lambda$ is an eigenvalue of M, that is, there exists a vector $v \in F^n$ n such that $Mv = \lambda v$. The verifier should be able to check the proof in constant time, independent of n. Let $C_M(\lambda, v)$ be an arithmetic circuit that outputs $0 \in F$ if and only if $Mv = \lambda v$ (the inner workings of CM are not important). Design a linear PCP(P, $V_1$, $V_2$) for $C_M$( not necessarily zero-knowledge), where $V_1$ issues only two linear queries. Recall that a linear PCP(probabilistic checkable proof) works as follows:

- the prover P outputs the proof $\pi := v$
- then $V_1$ issues two linear queries u, r where $u, r \in F^n$
- finally, $V_2$ gets back the query responses $a_u :=< u, \pi i >\in F$ and $a_r :=< r, \pi i >\in F$, and outputs yes or no.

  **a)** First, explain how $V_1$ chooses u, r and how $V_2$ decides when to output yes
  **b)** Then prove that a malicious prover cannot fool the verifier. That is, if

$Mv = \lambda v + \Delta$, where $\Delta \neq 0 \in F^n$, then the verifier will accept the proof with probability at most $\frac{1}{|F|}$ over the choice of $r \in F^n$
(Hint: $V_1$ will choose a random vector $r \in F^n$, and compute $u := r^T M \in F^n$. The first linear query from $V_1$ is $u \in F^n$ , and the second linear query is $r \in F^n$. Explain how $V_2$ works.)

**Question 5:** Let's do some hacking in solidity

**a)** Bob posts the following wallet contract to Ethereum to manage his personal finances. The function pay lets Bob send funds to anyone he wants. Suppose Alice can trick Bob into calling a method on a contract she controls. Explain how Alice can transfer all the funds out of Bob's wallet into her own account.

```
pragma solidity >=0.5.0 <0.7.0;

// THIS CONTRACT CONTAINS A BUG - DO NOT USE
contract TxUserWallet {
    address owner;

    constructor() public {
        owner = msg.sender;
    }

    function transferTo(address payable dest, uint amount) public {
        require(tx.origin == owner);
        dest.transfer(amount);
    }
}
```

**b)** Consider this simple auction smart contract. Explain how an attacker can always win the auction by using a special smart contract for bidding?

```
// INSECURE
contract Auction {
    address currentLeader;
    uint highestBid;

    function bid() payable {
        require(msg.value > highestBid);

        require(currentLeader.send(highestBid));

        currentLeader = msg.sender;
        highestBid = msg.value;
    }
}
```

# References

[1] L.Fan and H.S.Zhou *A Scalable Proof-of-Stake Blockchain in the Open Setting* , 2018.