Final Study Guide

Short Essay Questions will be from:

1. Explain a zero knowledge proof to a layman.

Read the section "Ali Baba Cave" in Wikipedia on Zero Knowledge Proof. https://en.wikipedia.org/wiki/Zero-knowledge_proof. Be able to reproduce the cave diagram and write a short explanation of it.

1. Explain zk-SNARK - What is the "SNARK" part of zk-SNARK.

"zk" is for zero knowledge.

SNARK is (S, N, AR, K)

- Succinct – the sizes of the hash functions (proofs) are tiny in comparison to the length of the actual process required to create them.
- Non-interactive – there is no or only little interaction. For zk-SNARKs, there is usually a setup phase and after that a single point of contact between the prover and the verifier. Furthermore, SNARKs often have the so-called "public verifier" property meaning that anyone can verify without interacting anew, which is important for blockchains.
- ARguments – the verifier is only protected against computationally limited provers. Provers with enough computational power can create proofs/arguments about wrong statements. (Note that with enough computational power, any public-key encryption can be broken). This is also called "computational soundness," as opposed to "perfect soundness."
- of Knowledge – it is not possible for the prover to construct a proof/argument without knowing a certain so-called witness. (For example the address he wants to spend from, or the path to a certain Merkle-tree node).

1. How can blockchain change accounting?

Double entry accounting was first recorded in 1340 in Genoa, Italy. It has remained basically the same since. We have automated the same system that has been used for hundreds of years. Blockchain provides a mathematically provable way of implementing accounting so that the most common form of accounting fraud is impossible to do. This is the first real "technological" advance in accounting in 500+ years.

1. When was double entry accounting invented?

Double entry accounting was first recorded in 1340 in Genoa Italy.

1. What is a blockchain? - Why is blockchain important? - What are blockchains used for?

A blockchain is a set of blocks that are written over time, and stored in a publicly accessible way. Each block is cryptographically signed and the signatures are used to chain the blocks together. The blocks are distributed so that every computer in the network has a copy of the blocks. Blockchain is the underlying technology behind BitCoin and other Cryptocurrencies.

1. Why is blockchain important?

Blockchain creates trust between un-trusting parties. It is a system for verifying that data that is entered has not changed over time. It creates an immutable data store that can be verified mathematically.

1. What is a smart contract?

A smart contract is a program that runs on a blockchain. It is run by the “miners” that run and maintain the chain.

1. What is “mining” in the context of a blockchain system?

Mining is the process of transferring funds from account to account and running of smart contracts. It also implements the digital signatures, (hashes), that provide security for a blockchain.

1. What are the good properties of a blockchain?

- Immutable data
- Distributed data
- Verifiable data
- Distributed source of trust

1. What are the bad properties of a blockchain?

- Slowest database around.
- Most expensive database around.

1. How do blockchains create trust between non-trusting parties?

The data is distributed and verifiable by all parties. Trust is no longer a part of an institution, it is a mathematically verifiable system.

1. Why is a blockchain an expensive database?

Ethereum has 12,000 nodes, (computers), that have to all update the storage of any data. It is not cheap to update 12,000 computers every time.

1. Why are blockchains a so slow?

Ethereum has 12,000 nodes, (computers). It takes time to update a large set of computers. Every computer has to see every block of data and append every verified block to the chain. Every new block of data has to be verified. Mining also takes time.

1. What is Ethereum “gas” and how does this relates to Turing-Complete?
2. Explain the Byzantine Generals problem and how Threshold-ECDSA (Honey Badger Byzantine Generals) solves it.

Imagine that Byzantine empire has decided to capture a city. There is fierce resistance from within the city. The Byzantine army has completely encircled the city. The army has many divisions and each division has a general. The generals communicate between each as well as between all lieutenants within their division only through unreliable messengers.

All the generals or commanders have to agree upon one of the two plans of action. Exact time to attack all at once or if faced by fierce resistance then the time to retreat all at once. If the attack or retreat is without full strength then it means only one thing—horrible defeat.

If all generals and/or messengers were trustworthy then it is a very simple solution. However, some of the messengers and even a few generals/commanders are traitors. There is a very high chance that they will not follow orders or pass on the incorrect message.

In message passing the desired result is that they all either agree or all disagree. Previous implementations (Lamport 82) of this required a *lot* of computational work to get the desired result.

BGP is applicable to every distributed network.

t-ECDSA solves this by having in its last stage encrypted data that thru key sharing can only be decrypted when you have a threshold number of participants agree on the private key information. At the moment when you have the threshold number then you can decrypt the data (the transactions in a blockchain) and the data is finalized and accepted.

Honey Badger sits on top of t-ECDSA.

Coding:

1. Solidity - a simple contract to store the hash of a document and a name for the document.

- When is the contract constructor run?
- Who can call the functions?
- What do the 2 functions do?

Finance:

1. What is the Yield Curve?

A yield curve is a line that plots the interest rates, at a set point in time, of bonds having equal credit quality but differing maturity dates. The most frequently reported yield curve compares the three-month, two-year, five-year, 10-year and 30-year U.S. Treasury debt. This yield curve is used as a benchmark for other debt in the market, such as mortgage rates or bank lending rates, and it is used to predict changes in economic output and growth.

1. An inverted or down-sloped yield curve suggests yields on longer-term bonds may continue to fall, corresponding to periods of economic recession.
2. A Mutual Fund is an investment vehicle made up of a pool of money collected from many investors for the purpose of investing in securities such as stocks, bonds, money market instruments and other assets. ... A mutual fund's portfolio is structured and maintained to match the investment objectives stated in its prospectus.

Solidity Code

```
pragma solidity >=0.4.21 <0.6.0;

import "openzeppelin-solidity/contracts/ownership/Ownable.sol";

contract DocumentReg is Ownable {

  mapping(bytes32 => bool) infoSet;
  mapping(bytes32 => address) infoOwner;
  mapping(bytes32 => string) infoData;
  mapping(bytes32 => string) infoName;
  uint256 minPayment;

  event DocumentSet(string, bytes32, string);

  constructor() public {
    minPayment = 0;
  }

  function setPayment ( uint256 p ) public onlyOwner {
    minPayment = p;
  }

  function newInfo (  string name, bytes32 infoHash, string info )
           public payable {
    require(!infoSet[infoHash], "already set, already has owner.");
    require(msg.value >= minPayment, "insufficient payment to set data.");
    infoSet[infoHash] = true;
    infoOwner[infoHash] = msg.sender;
    infoData[infoHash] = info;
    infoName[infoHash] = name;
    emit DocumentSet(name, infoHash, info);
  }

  function updateInfo (  bytes32 infoHash, string info )
           public payable {
    require(infoOwner[infoHash] == msg.sender, "not correct owner of this data.");
    require(msg.value >= minPayment, "insufficient payment to update data.");
    string storage name;
    name = infoName[infoHash];
    infoData[infoHash] = info;
    emit DocumentSet(name, infoHash, info);
  }

}
```