Carnegie Mellon University

Private Computation on the Blockchain

Anirudh Baddepudi   Supervised by: Vipul Goyal

Introduction and Motivation

- Blockchain is an innovative technology with the potential of transforming cryptography and computation. Currently, everything stored on a blockchain, such as an Ethereum network, is public. However, Prof Goyal at CMU is working on a way to also store/retrieve private data, such as encrypted keys and encrypted data on a blockchain network.
- In this project, we aim to implement Blockchains with private comptuation capability. This will enable miners (individual machines interacting with the blockchain) to jointly store data in secret shared form.
- This technology has a wealth of applications in various domains. In this project, we work with an engineering lab at CMU and the Department of Energy to design and implement a secure private blockchain architecture. Our main focus is on storing and retrieving private data hidden from an adversarial miner.

Secret Sharing

- We implement a modified version of Shamir's secret sharing scheme. We use this to distribute shares of a secret key, which is used to encrypt the data/file we want to store on the blockchain.
- As some miners can be completely dishonest, the secret sharing scheme also includes error correction properties, similar to Reed-Solomon codes.
- This helps prevention from the situation where some nodes are actively corrupted and release incorrect shares during secret reconstruction.

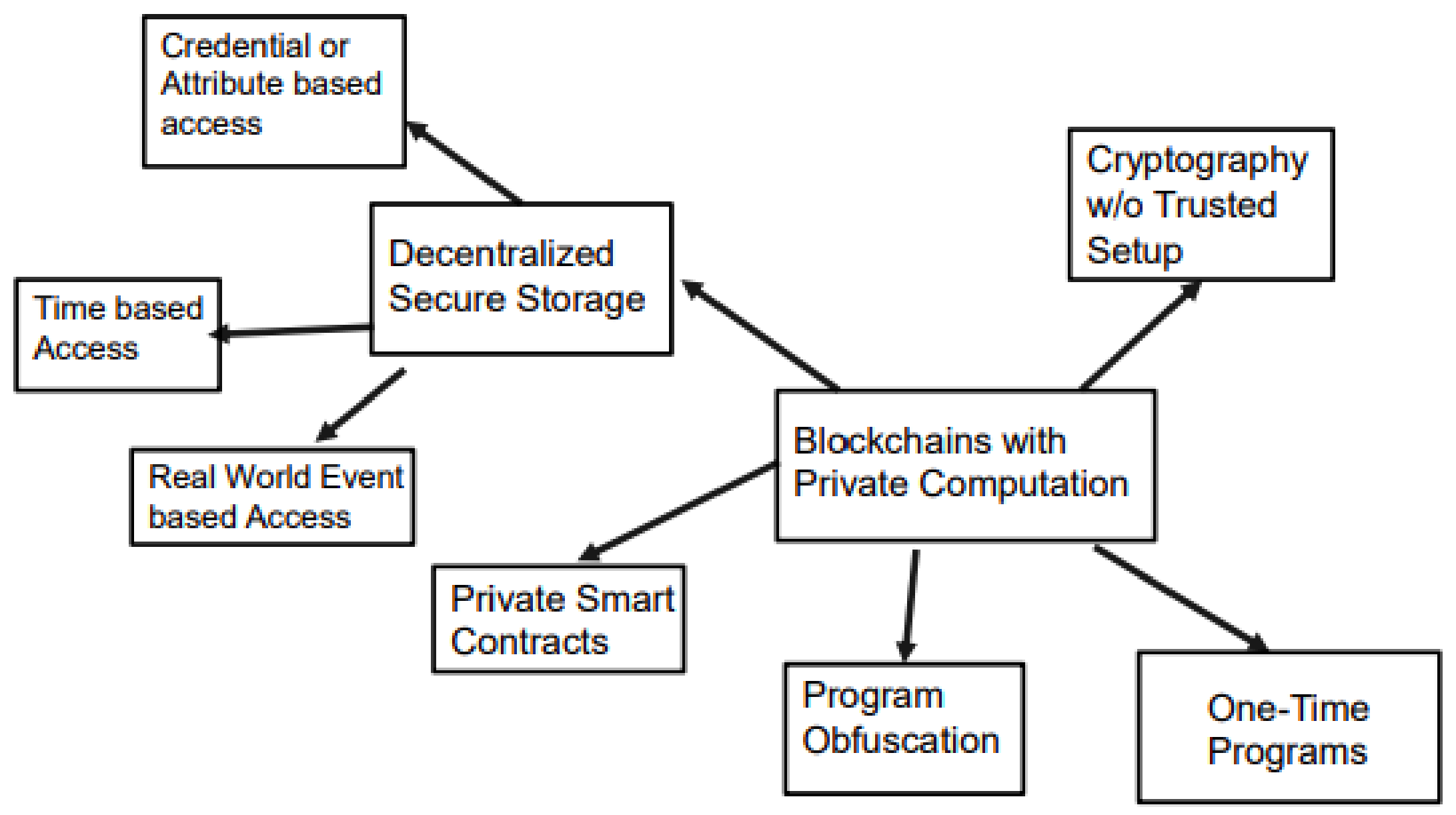
Shamir's Secret Sharing Scheme:

- We implement an $(\frac{n}{2}, n)$ Shamir secret sharing scheme.
- Let the secret be S. We then construct a random polynomial $f(x) = S + S_1 x + S_2 x^2 + ... + S_{\frac{n}{2}-1} x^{\frac{n}{2}-1}$, where the secret S is the constant term. A share is defined as a tuple $(i, f(i))$, where for us $i \in \mathbb{Z}$.
- As the degree of the random polynomial is $\frac{n}{2} - 1$, with access to $\frac{n}{2}$ shares, we are able to reconstruct the secret. This is done by reconstructing the polynomial using Lagrange interpolation.

Private Ledger Overview

We first enable storage of secret data on the blockchain (Ethereum) by performing the following:

- Given classified data M, generate a secret key K and encrypt M under K. The ciphertext of M is posted publicly on the blockchain.
- Then securely distribute shares of K to the n miners.
- We utilize the public keys associated with each miner from their associated blocks mined (we choose the latest N) in order to encrypt the secret key shares.
- K can be reconstructed if greater than $\frac{N}{2}$ of the shares are revealed, by Lagrange Interpolation. This parameter can be changed.


Credential or Attribute based access
Cryptography w/o Trusted Setup
Decentralized Secure Storage
Time based Access
Blockchains with Private Computation
Real World Event based Access
Private Smart Contracts
Program Obfuscation
One-Time Programs

System Design

The overall system is designed in two major components - **storing** and **retrieving** data. We use geth (Go) Ethereum as the blockchain instance.

Storing Data:

- For the purpose of this project, data is given as an xls file. This is first converted to bytecode.
- An implemented program encrypts this bytecode using a randomly generated 32-byte secret key. Shares of this secret key are then created locally.
- We then use solidity and the Remix IDE to construct and deploy/test smart contracts in which the encrypted data and secret key shares are stored.

Retrieving Data:

- Miners interact with the contract and decrypt the shares associated with them (this is implemented by some arbitrary numbering on the miner blocks) and then release these on the blockchain.
- When a release condition is posted and the minimum number of shares are posted, then the secret sharing reconstruction takes place locally on a device.