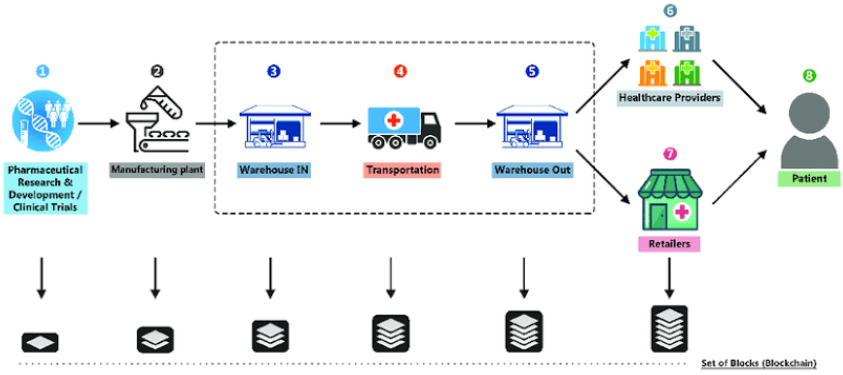
# BLOCKCHAIN FOR PHARMACEUTICAL SUPPLY CHAINS

*BITS F463 Cryptography*

*Naren Surampudi - 2016AAPS0206H*

## INTRODUCTION:

Blockchain technology as become increasingly popular in the current age, with many companies and businesses adopting the technology for making sensitive transaction more secure. This has also opened up avenues in applications that do not strictly use a crypto-currency. One such example is the application of blockchain for stemming the presence of counterfeit products in the market, especially in the pharma industry. This project aims to build a simple blockchain application for the pharma supply chain, making use of some basic features to show how blockchain can be used for solving the problem of counterfeit drugs.



## THE APPLICATION:

*INSTRUCTIONS FOR RUNNING THE APPLICATION:*

*To run the application, navigate to Blockchain/codebase within the RAR file through the terminal of choice. From here, run the Python script blockchain.py to start the application*

*Additionally, if on Windows, the application can be started by launching the blockchain.exe file present in Blockchain/dist. In summary:*

- *Codebase is present in Blockchain/codebase*
- *On Windows, application can be run by executing Blockchain/dist/blockchain.exe*
- *Application can be run by executing Blockchain/codebase/blockchain.py from the terminal using either the python or python3 command*

The application is divided into various scripts, with each containing various methods relevant to the application and its usage. This document gives a description of the implementation.

*Class block_script.py*

This class implements the "block" of the blockchain. The current implementation of the block hold information of the product ID, location ID, current hash, previous hash and time of creation of block. The *create_block()* method is used to create a block, which in turn call the *create_hash()* function which calculates the hash of the block using the previous hash, nonce, product ID and location ID. The *mine_block()* method in this class is used for mining the newly created block. This function uses preset values- difficulty hash and difficult decimal. The idea of this method is to generate proof of work. The method takes as it's parameters the set of miners available for mining. These miners then compete with each other to solve the proof of work, which in this case is to generate a hexadecimal hash that is greater than or equal to the difficulty hash. The work done is indicated by the nonce count of every miner, and the current hash of the block is updated after every iteration according to the nonce count before finally adding the block onto the chain. This makes it more secure, since the nonce count cannot be pre-determined.

*Class chain_script.py*

This class implements the “chain” of the blockchain and is used for chaining together the blocks. The *add_block()* method is used for adding the block onto the chain. The *verify_block()* is a simple verification method for verifying the validity of the chain by tracing through the entire chain of blocks. The *verify_transaction()* method is used for verifying the transaction initiated by the user. In the current use case, a transaction is the process of recording the presence of a drug or medicine at some part of the supply chain. This is achieved using Zero Knowledge Proof (ZKP).

*Class user_script.py*

This class implements the typical user or employee who is part of the supply chain. This class works in tandem with the *verify_transaction()* method using it’s various methods to complete the ZKP. The application currently uses just one round for ZKP for demonstration purposes, and uses a prime value of 7919 and generator value of 7 for calculations. This can be scaled up to further rounds to improve verification security. The *view_user()* method is used to view all the transactions that were completed by this particular user in the entire history of the blockchain.

*Class miners_script.py*

This class implements the methods required for creating a miner and building a miner network. The *add_miner()* method creates a miner using a provided name and the amount of CPU units the miner can use. The amount of CPU units is directly proportional to the computing power of the miner. The *create_miner_network()* creates a network of miners using the information of the miners provided. Its main function is to shuffle the computing power of the provided miners to emulate the real world, without clustering all the power together. This ensures that the chance of a miner completing the proof of work is only dependent on the computing power and not on the order in which the miner was added.

*Script blockchain.py*

This script is the entry point of the application that calls all the relevant methods to build the blockchain as well as perform the relevant functions. Running the

script gives a console window that can be used for performing various operations on the blockchain. The script creates three miners- Alice, Bob and Charlie with computing power of 5, 3, 9 respectively everytime the script is run. In the real world, this can be viewed as the company ensuring the running of the blockchain independent of the presence of any miners i.e. these miners are employed by the company. More temporary miners can be added through the console however to emulate a bigger network. The script also makes use of two files *chain_file.pickle* and *user_file.pickle*. These files contain the information of the blockchain and the user transactions respectively and are loaded everytime the script is run, updated after every successful transaction and saved before exiting the console. Various commands are available to the user for operations to the console and their respective descriptions can be made available by running the *help* command in the console. Some terms that have been widely used in the application are:

- Product ID: The ID of the medicine or drug. Every medicine has a unique ID and every such medicine has a blockchain of its own
- Chain ID: The ID of the blockchain of some product. Every product can have multiple blockchains. Every blockchain is thus uniquely identified by the Product ID and the Chain ID
- User ID: The ID of the employee. Used for verifying a transaction
- Location ID: The ID of the location where a user in the supply chain can initiate a transaction

## FURTHER UPDATES

The application has been built as a simple demonstration of how blockchain can be used as part of the pharma supply chain for combating counterfeit drugs. Below are listed updates that can be added to the application in the future:

- Tracking of product. This can be accomplished by having a database of locations and their respective IDs, and traversing the blockchain for some product accordingly. This introduces more transparency in the functioning of the supply chain
- Adding more rounds to ZKP and using a bigger prime value. This will make the verification of the users more secure and will take negligible time on faster computers
- Easier verification of the product. The current implementation expects the user to enter the entire hash code of a product to verify whether the product fits in the supply chain or not. This can be made easier by replacing it with scanning technologies like barcodes or QR codes to make this easier