CS620 Assignment 3

Due date: November **1**, 2019

Objective: Build a solidity smart contract for distribution of licensed media on Ethereum private network with at least 3 different geth instance, each acting as a user

Overview:
In this assignment you will build a Smart Contract for licensed Media (Digital Copies of songs or movies) distribution over Ethereum (Private network). The smart contract must have the following features for licensed media distribution

- Users
  - Smart contract can be accessed by the set of users. Every user accessing the contract can act as a Creator and/or a Consumer.
  - Two types of consumers are available: Individual or Company.
- Licensed media.

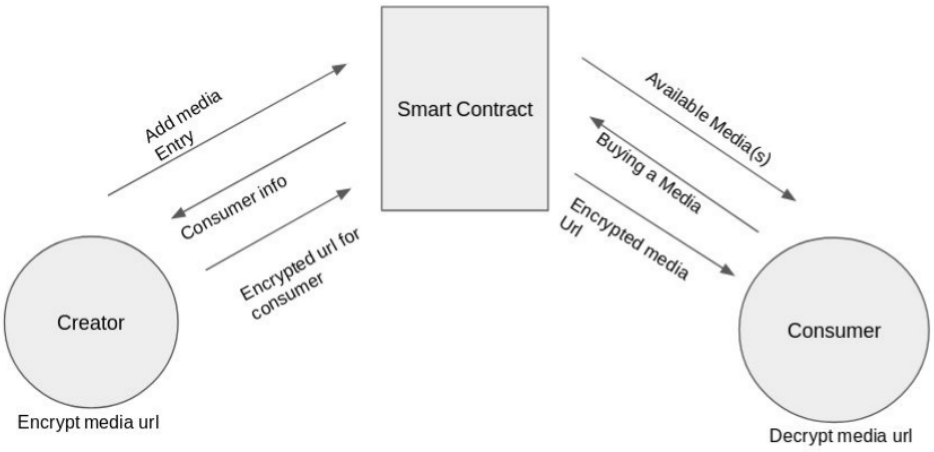

Figure 1: Data flow description

Working

- Every user will register themselves as a Creator and/or Consumer. The consumer can be a company or an Individual.

- A creator should be able to **add a licensed media to the contract** (You have to come up with an appropriate data structure to represent a media in the contract.). The creator should also set the cost for the media license for individual consumers and a Company. The media or media url is **not accessible** by the consumers until they pay for the license. The creator should also be able to specify stakeholders and the percentage of profit they are going to gain. The stakeholder can be one of the registered users.
- The consumers should be able to view all the media entries available to buy. Whenever a consumer wants to buy some media, he/she should initiate a transaction, and the contract should internally invoke a function to transfer the license amount to the creator. You should ensure that the amount paid for the license depends upon the type of consumer(individual or company), where the **payment should be in Ether**. You should also ensure that the consumer has sufficient funds to purchase the media.
- Whenever a consumer buys any media, the creator should **create a new encrypted url** (encrypted using consumer's public key) of the media. The creator will add the encrypted url in the contract (assume creator to be honest). Now, for a consumer to access the media, he needs to **decrypt the url** and use it. Once paid for a particular media, a buy option should not appear for the consumer anymore, for that media from that particular consumer. Please refer Figure 1 to clarify the data flow.
- The revenue obtained from the media license by the creator should be divided among other stakeholders(ex. Production company) of the media. (Number of stakeholders can range from 0 to maximum of 5). The creator should be able to specify the stakeholders and corresponding share of each stakeholder while creating the media entry. The division of revenue (generated from each media license selling) among the stakeholders should happen on-chain by a contract function.

Supporting files and Code skeleton

To make your life simpler, you have been provided with a skeleton of solidity code. Skeleton consists of a list of functions with the description that appears in the comment on top of it. Also, we have provided python scripts to deploy the smart contract(**deployContract.py**) and to send the transaction by calling the exposed functionality of the smart contract(**sendTransaction.py**). You have to modify the deployContract.py by passing appropriate variables to the function. For example, value 42 is passed to the *constructor* in *deployMediaContract* function. You have to modify the python script to pass the required variable to initialize your smart contract.

We have provided a function to call register user (*sendRegisterTransaction*), in the sendTransaction.py. You are expected to write other functions to fire the transaction like to add the media, to purchase the media, etc. Also, *registerUser* call in *sendRegisterTransaction* is passed with value 1. You have to modify the function call to pass the appropriate user parameter required for the registration.

User should be able to take the following actions by firing the **Transactions**. The list below contains a list of functions in the smart contract and description of what each function should do.

- *registerUser* - Register the user in the smart contract (DAPP). This function is accessible to both creator and consumer.

- *addMedia* - Add the media to smart contracts with the list of stakeholders, the owner(creator) info, stakeholder shares, the price they are offering for an individual and a company, etc. This function should be accessible to the **creator** only.
- *purchaseMedia* - It transfers the purchase amount from the consumer to the creator account. Please make sure to distinguish between the individual consumer and the company while charging the amount. Also, this function should ensure the availability of balance in the consumer's account. This function should be exposed to the **consumer** only.
- *sendMediaLink* - It randomly generate the url and send it to the consumer after encrypting it with the consumer's public key. Before sending the url to the consumer, you should ensure that the payment made by him is confirmed (payment transaction is encapsulated in some block). This function should be accessible to the **creator** only. (**Clue:** you can monitor the status of payment by maintaining a variable for each consumer).

You should provide the user with the following get APIs to access the data. Please note that these are **not** the transactions. Solidity skeleton has provided the *getMedia*, likewise you have to add other get functions. Also, you have to fill the *getMedia* function with appropriate logic to return the list of available media with each creator.

- *getCreator* - View the list of available creators - Only accessible to consumer.
- *getConsumer* - View the list of available consumers - Only accessible to creator.
- *getMedia* - List the media offered by each creator - Only accessible to consumer
- *getMediaDetails* - Details of each media - Accessible to Both consumer and creator

sendTransaction.py provide a call to solidity get function *getMedia*(printSendGetMedia); likewise, you have to add functions to call the other get functions listed above. Also, please note that the path to the geth client and solidity contract is hardcoded in both the python files, you may have to change it while running the scripts at your local machines.

To run provided python scripts, your system should have few system packages. We have provided the file *installPackages.sh*, which will do the task of installation. You can run the file by "sh installPackages.sh". If you think a GUI is required, you can have a GUI. However GUI is not essential.

Program Output
You should submit your code on moodle along with readMe file and modified python scripts (deployContract.py and sendTransaction.py)(submit a zip file).

Useful Links You can follow the links below to improve your knowledge about solidity and smart contract.

1. `https://medium.com/@mvmurthy/full-stack-hello-world-voting-ethereum-dapp-tutorial-part-1-40d2d0d807c2`.
2. `https://medium.com/@mvmurthy/full-stack-hello-world-voting-ethereum-dapp-tutorial-part-2-30b3d335aa1f`
3. `https://medium.com/@mvmurthy/full-stack-hello-world-voting-ethereum-dapp-tutorial-part-3-331c2712c9df`

Submission Instructions:

1. The assignment should be done in a group consisting of a maximum of two students.
2. The code should follow programming etiquette with appropriate comments.
3. Add a **README** file which includes a description of the code and gives detailed steps to compile and run the code.
4. Zip all your code and the readMe as a single file with the name **rollno1-rollno2.tar.gz** and upload it to Moodle. Only one group member should upload the file.
5. You should be able to **demonstrate** the code on a machine provided by us.