USI Blockchain Winter School: Writing Smart Contracts

Projects

Peter H. Gruber

General rules

The projects are deliberately formulated in an open way. A detailed formulation of the problem and a project plan is already part of the project. This includes the decision, which features to implement beyond the minimum requirements and how to document your project.

The main goal is to produce one working and secure solution. Complexity is rewarded, but everything that you hand in has to work. Document all steps of your project such that it is reproducible. This includes all code and all commands. All your submissions will be tested for possible vulnerabilities.

Deliverables

You should deliver all of the following for your project.

- Write the necessary smart contract(s) and deploy them on the Algorand testnet.
- Provide one Jupyter notebook with all the code for *creating* and *deploying* your the smart contract(s)
- Provide a second Jupyter notebook for *using* the smart contract(s). Include several use cases that work. This should include ways to calculate the right amounts that should be requested to obtain one's payoff (if applicable).
- Provide a third Jupyter notebook for *testing* the smart contract(s). Include several attacks that result in an error message.
- You may create an additional web page, if useful. Document the URL in the PDF.
- Provide a documentation in PDF form that contains the following
  - Brief description of your project
  - Addresses of all smart contracts
  - Explanation for end users on how to *use* the smart contract(s)
  - Do not include your credentials. Rather explain how and where credentials must be added and which accounts must be funded by how much ALGOs.

Grading criteria

1. Correctness: Does your solution work? Is it secure?
2. Problem solution: Did you implement the requirements completely?
3. Documentation: quality and detail of documentation
4. Complexity/comprehensiveness: extra points for going beyond the "minimum requirements"
5. Coding: structure and style of your program(s)

Grading criteria are in decreasing order of importance, e.g. a good programming style cannot compensate a solution that does not work.

Material from the web or from other courses

Use of foreign code (libraries, tutorials from stackexchange or others) is permitted. Foreign code (=anything that you did not write yourself for this project) needs to be cited correctly. Foreign code is ignored when assessing your contribution (I.e. it counts neither positively nor negatively towards the contribution. This implies that if you used only foreign code, there is no contribution from your part). It is absolutely forbidden to use one project in two courses.

1 Quiz Coin

Implement all of the following steps: Create a "Quiz Coin". Quiz enthusiasts obtain the "Quiz Coin" for free from a "Dispenser" Smart Contract. They are then offered quizzes that have either textual answers (e.g. "In which country is USI located?" – "Switzerland") or numeric answers (e.g. "If you calculate the square of x, the result is 25. What is x?" – "5"). To participate, the quiz enthusiast can send his/her guess for the answer together with one Quiz Coin. (Submissions without Quiz Coin should be rejected). The first one who find the correct answer obtains a fixed price of 50 ALGOs. After a quiz has been correctly solved, no more submissions should be accepted.

Extensions. (1) Implement a Smart Contract that sells the Quiz Coin for 1 ALGO. (2) Implement a rebate scheme such that the quiz enthusiast may buy 10 Quiz Coins for only 9 ALGOs and 50 Quiz Coins for only 40 ALGOs. (3) If no correct answer has been submitted after 1 week, the prize money (50 ALGOs) goes back to the quiz company and no more submissions are possible.

2 Mini lottery

Implement all of the following steps: six numbers out of 45 are secretly drawn (without repetition). Drawing the numbers can be implemented off-chain. A player can submit one guess of the six numbers for 1 ALGO. If the player has guessed correctly, she receives 50 ALGOs. The remainder goes to the lottery company and the game is closed.

Extensions. (1) If the player has guessed correctly, she receives *half* of the money submitted so far. The other half goes to the lottery company and the game is closed. (2) If no player submits the correct numbers after 1 week, the prize pot goes back to the lottery company and no more submissions are possible. (3) Implement the drawing of the numbers on-chain. (4**) Do not pay out immediately when a correct submission has been made, but allow submissions for a period of time (1 week) and split the pot such that one half is distributed among all the winners and one half goes to the lottery company.

3 Call option

An american call option gives the buyer the right, but not the obligation, to buy a certain asset at at any time within a certain time frame for a fixed price (the so-called strike price). Recreate a call option as an atomic swap. The owner of the option can redeem by paying a certain number of ALGOs in return for an ASA. If the option is not redeemed, the ASA goes back to the seller of the option.

Extensions. (1) Create a new contract that represents a European call option. A european call option gives the owner the right, but not the obligation, to buy a certain asset for a fixed price only at a certain point in time. In practise, there is an exercise window which is 1 week from the "expiry date". (2) Create yet a new contract that represents a cash-settled Euopean put option. In this case, the seller obtains a cash payoff that is "strike price minus current price", if the current price is below the strike. Use an oracle to determine the current price.

Hint: You may use the oracle from the course. *Hint 2:* This contact must be structured a bit differently than the call options.

4 Staged vesting

Write a simple time locked contract so that team members get 1/4 of their tokens after 3 months and then 1/12 of the tokens after each month so that the overall tokens are available out after one year.

5 Incentive-based vesting with oracle

Start with a simple time locked contract so that team members get 1/4 of their tokens after 3 months and then 1/12 of the tokens after each month so that the overall tokens are available out after one year.

Now extend your contract with an incentive component. If the price of the token is currently 10% larger than at the beginning, then 10% of the tokens are unlocked, regardless of time, if the price is currently 20% higher, then 20% are unlocked and so forth.

If both conditions apply, then tokens are unlocked only once. Example: if after 2 months the price has risen by 20%, then 20% of the tokens are unlocked. After the end of month 3, $1/4 = 25\%$ would have been unlocked, but as 20% have already been unlocked, only an extra 5% are unlocked at this point in time. If a few days later the price has risen by a total of 30%, then an extra 5% of tokens are unlocked.

Hint: You may use the oracle that we have created in class.

Extensions. (1) Create your own oracle. (2) Change the incentive condition so that it does not depend on the current price but on the maximum of the historic price. (There are several ways how this can be done).

6 Voting

A club has a membership list with the public keys of all members. Members receive one free vote coin for the elections of the new president. Write a smart contract that allows members to vote for one of three candidates (Rossi, Smith or Meier).

Extensions. (1) Ensure that voting is only possible in a certain time frame. (2) If you have not yet done so, add a counting and a selction of the majority candidate on-chain.

7 Social Security

Write a Smart Contract that everyone who has less than 100 ALGOs in his/her account receives 100 ALGOs for free.

Extensions. (1) Subtract the current holding from the free transfer, so that someone who hold 20 ALGOs would only get 80 ALGOs for free. (2) Only eligible citizens, who own the "Social Coin", can receive the transfer.

8 Time coin*

Create a coin called Timecoin that only you own. Create a Python script that obtains the UNIX timestamp for UTC and that adjusts the free float of Timecoin so that it matches the UNIX timestamp. Run a cron job on a server of your choice to regularly adjust time coin.

Extensions. Produce sample code how time coin could be used as well as documentation and publish it on Github.

9 Blockchain analytics*

Create a web site that performs some analysis of the Algorand blockchain (You choose which analysis exactly. The Analysis can be limited to one specific aspect.) Create a coin called Marketcoin and create a smart contract that sells 10 Marketcoins for 10 Algos. You business model is that whoever possesses 10 Marketcoins has access to your web site. Allow the user to link their wallet to your web site, which automatically gets the amount of Marketcoins in the user's wallet and grants (or does not grant) access to the website.

Extensions. (1) Set up your own indexer node so that you are not depending on the Purestake API. (2) Create a second tier of more valuable information that is only available to users who have 100 Marketcoins in their wallet.