Principles of Decentralized Ledgers
Courseguide 2020

Dr. Arthur Gervais

Part I
Course Layout

The course takes place in the second term, room HXLY 311, over a duration of 9 weeks, with 7 main lectures and 7 lectures supporting the course work. The main lectures are scheduled on Tuesdays 2-4pm, while the supporting lectures are scheduled on Wednesdays 9-11am. The inaugural lecture is scheduled for the 14th of January 2020.

The main lectures serve to primarily convey the main course contents. The supporting lectures serve to discuss more about smart contract programming, relevant scientific papers, discuss and advance on course works as well as to elaborate and repeat on the contents provided within the main lecture.

1 Schedule

You can find the tentative course schedule in Table 1.

2 Grading Scheme

We plan the following course works, exam and grading scheme for this year's course.

Blockchain Workbench 0% of the grade

Cryptomons 20% of the grade

Paper Reading 0% of the grade

Exam 80% of the grade

14.01	Introduction Blockchain
15.01	Blockchain Workbench 1
21.01	Blockchain Background
22.01	Blockchain Workbench 2
28.01	Smart Contracts
29.01	Paper Reading/Presentation by Lecturer Session 1
04.02	Consensus and P2P Networks
05.02	Blockchain Exam Exercises
11.02	Security
12.02	Paper Reading/Presentation by Lecturer Session 2 / Due date for Cryptomons
18.02	Scaling
19.02	Exercise Exam 2019 / Return marks for Cryptomons
25.02	Privacy
26.02	Presentations for Cryptomons (the 5 best projects only)
03.03	Summary Revision / Q&A (Optional)
04.03	Free for revision
10.03	Free for revision
11.03	Free for revision
16.03 - 20.03	Exam Week

Table 1: Tentative Course Schedule 2020.

3 Administrative Comments

3.1 External Students – Registration for DoC Courses

1. Apply at: `https://dbc.doc.ic.ac.uk/externalreg/`
2. Then,
   - Your department's endorser will approve/reject your application
3. If approved,
   - DoC's External Student Liaison will approve/reject your application
4. If approved (again!),
   - Students will get access to DoC resources (DoC account, CATE, materials, ...)
   - No access after a few days? Check status of approval and contact relevant person(s)

Key Dates

- Exams for DoC $3^{rd}/4^{th}$ yr. courses take place at the end of the Term in which the course is taught
- Registration for exams opens end January

If in doubt, read the guidelines available at the link above :)

Part II
Course works

In this part we discuss the course works.

4 Blockchain Workbench

To complement the lectures and courseworks, we will work through the blockchain workbench (`https://blockchainworkbench.com/`) during the supporting lectures. The blockchain workbench covers the blockchain basics and allows to learn and practice the development of solidity based smart contracts. I expect that we can cover the workbench modules (i) introduction, (ii) structure of a contract, (iii) functions, (iv) types and (v) contracts, **within the first two supporting lectures**.

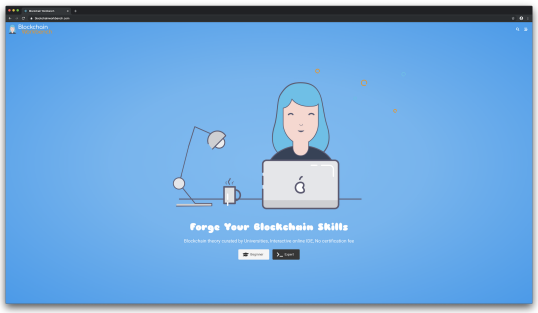

Figure 1: Blockchain Workbench landing page. Proceed as "Beginner".

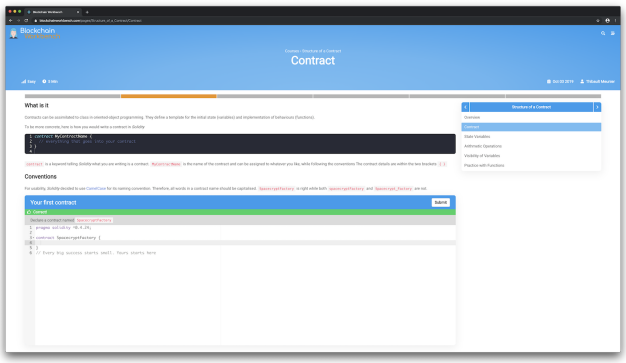

Figure 2: Smart contract IDE integrated into the web interface with editor, deploy and test scripts.

5 Cryptomons

Within the second (and the only graded) coursework, we will design an application that allows to purchase and sell cryptomon cards and let them fight and breed. You are expected to become more familiar and put into practice the advanced development of a smart contract. You can develop for the EVM (Ethereum Virtual Machine) or any other blockchain that you're interested in. For the EVM I would recommend to use Solidity, but you're free to use any language. Besides the tutorials of the blockchain workbench, you might find the following resource helpful: `https://solidity.readthedocs.io/en/develop/solidity-in-depth.html`.

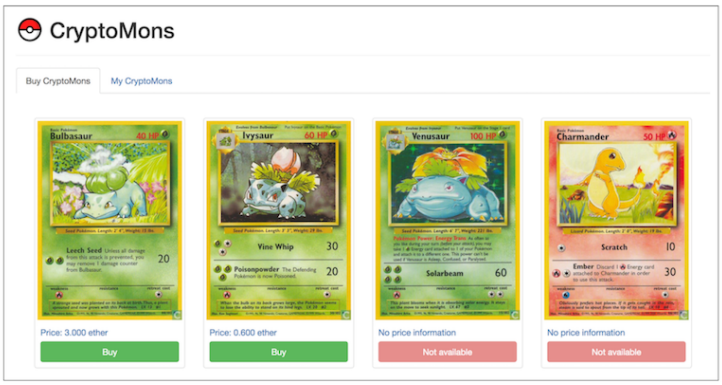

Figure 1: Screenshot of the marketplace where players can buy available CryptoMons

Figure 3: Cryptomons interface example.

5.1 Background

CryptoKitties, a once popular blockchain game, allows players to exchange cryptocollectibles and breed new kitties (`https://www.cryptokitties.co/`, `https://www.youtube.com/watch?v=jGfvkjzLrNw`). In this coursework, you can implement your own cryptomon game, where users can purchase and sell virtual cards. You're asked to specify and design a battle logic (i.e. fights between cards), in addition to the trading, sharing and ownership logic that is handled by the smart contract. Moreover, different cryptomons should be compatible to breed and create new cryptomons over time. While cryptomons are breeding, they cannot fight, and if they are attacked, their breeding might fail. You are free to specify the exact rules that your cryptomons are governed by — please be creative! Your rules must be transparent and clearly described within your specification.

5.2 Outline

We expect you to design the Cryptomon game in for example two main components, *(i)* the frontend and *(ii)* the Cryptomons smart contract (cf. Figure 4).

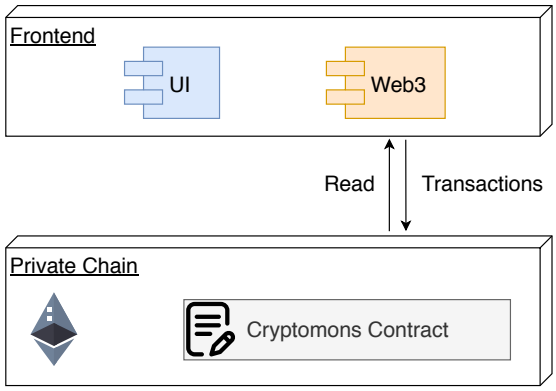

Figure 4: System Architecture.

You can add a backend server if you think that's required, but we do not see it as being necessary. If you chose to not use any backend please explain your architectural choices within the report.

5.2.1 Frontend

The frontend component provides a "user-friendly" graphical interface that allows users to perform blockchain-related operations. The basic requirements for the frontend component are as follow.

Frontend Requirements

1. The frontend manages the user's account (i.e. the private key is safely generated and stored; the account address and its balance can be properly shown on the web page).
2. The user is able to read the contents of the Cryptomons contract and issues signed transactions through the web page. The execution results of these operations should be appropriately presented to the user.
3. The front-end should allow a user to execute all functionality offered by your smart contract (i.e. trading, sharing, fighting, breeding of cryptomons).

You can choose any library or framework to develop and manage the web-based frontend. Some popular options may be:

- React (`https://reactjs.org/`)
- Angular (`https://angular.io/`)
- Vue (`https://vuejs.org/`)

Feel free to use Vanilla JavaScript if you think it's enough. We recommend to use web3.js (`https://github.com/ethereum/web3.js/`) to interact with the blockchain. For account management, you can try MetaMask (`https://metamask.io/`) or set up an in-browser wallet as *blockchainworkbench* does.

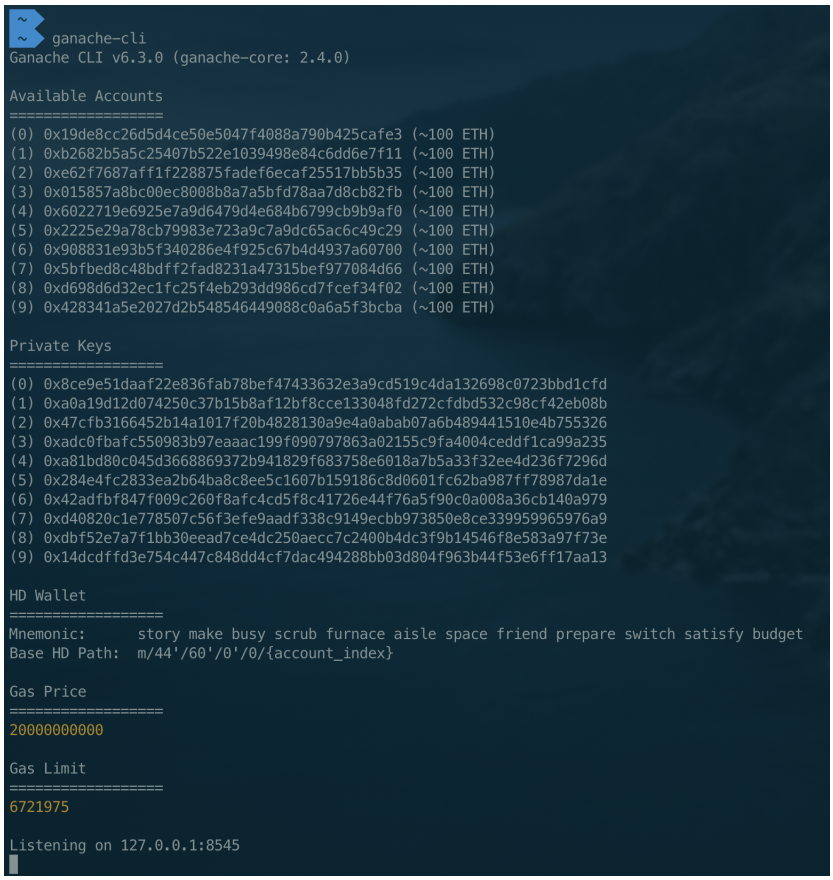

Figure 5: Running ganache-cli with default configurations.

5.2.2 Cryptomons Contract

It's easier to develop and test your Cryptomons contract in a private blockchain environment. Ganache (`https://www.trufflesuite.com/ganache`) is an easy-to-use tool that helps you quickly set up an Ethereum emulator environment.

Figure 5 shows how to run ganache with default configurations. By default, it sets up 10 accounts, and allocates to each a balance of 100 ETH. Ganache also shows you the corresponding private keys that allows you to operate sign transactions from these accounts. As shown in Figure 5, you can access the private chain on TCP port 8545. Ganache is quite flexible and allows you to set e.g. port, block interval as you like. We recommend you to read its documentation and configure your private chain in line with your requirements. Alternatively, you can also use other Ethereum clients, e.g. Parity (`https://www.parity.io/ethereum/`) and Geth (`https://geth.ethereum.org/`).

You should deploy your Cryptomons contract to your private chain. The requirements are listed as follow.

Cryptomons Contract Requirements

1. Every Cryptomon card should be uniquely identified and its ownership is clearly specified in the contract.
2. Users should be able to trade cards at an agreed price.
3. You should design a basic fighting mechanism. For example, you can assign two properties, HP and ATK, to each card. The HP of each card is reduced by the ATK of the opponent in a fighting round.
4. You should also design a breeding mechanism. Consider how properties are inherited to children.
5. Users should be able to verify the properties and states of any Cryptomon card.
6. The game logic including trading, breeding and fighting should be secure which implies that no malicious player can manipulate the game. For example, the trading, payment and ownership transfer should be atomic. In your report, please elaborate on your security considerations and how you guarantee them in your smart contract.

All the aforementioned requirements are the most basic features you can specify and implement. You are encouraged to use your creativity to make your game unique and add more features you deem interesting. As this is the only coursework for this course, there will be extra points given for independent extension and novel game solutions.

Security The smart contract needs to be resistant against an adversary that tries to attack the game and is able to read the blockchain's content. For instance, you should make sure that your contract is not vulnerable to reentrancy attacks, and describe what measures you have taken to secure the contract.

5.3 Schedule and Guidelines

The coursework is scheduled according to the following guidelines.

- The project is assessed individually.
- Write the complete specification of the application.
- Implement the specification.
- Write a short report (with screenshots and max 2000 words, up to 10 pages).
- Report + Code due by 12th of February 2020.
- The 5 best projects will present on the 19th of February 2020.

5.4 Grading

The grading will be mostly influenced by

- the quality of the code. Simplicity is preferred over complexity.
- secure programming considerations are required and the attacker model needs to be explained explicitly.
- building a web interface to interact with the game is required (a mobile, e.g. react native version would also be OK).
- the clarity and quality of your written presentation.

5.4.1 Detailed Grading Scheme

Each report is planned to be graded according to the following grading scheme:

Overall Report Quality (1-3) How easy the report can be read, ideally by someone with less blockchain experience.

Topic and Specification Description (1-3) How well the topic and the detailed specification is provided.

Threat Model Description (1-3) How accurate and real-world applicable the threat model is described.

All basic functionality working? (0-4) : trading, sharing, fighting and breeding

Screenshots (1-3) Quality and appropriateness of the final solution screenshots.

Code Quality (1-3) Efficiency over complexity wins, the code should be appropriately commented (possibly using netspec as seen in the blockchain-workbench).

Secure Coding Practices (1-3) Have security vulnerabilities such as reentrancy, missing modifiers, etc been considered?

Web Interface Usability (1-3) How approachable is the web interface for non-blockchain users?

Bonus points for creative additions (1-3)

Bonus points for the presentation of the top 5 projects (1-3)

Note: while beautiful graphics are great to look at, you won't be graded on the visual appearance of the game - rather on the functionality and user-friendliness. As such, we're not sharing any dataset for cryptomon cards, it's really up to you to be creative and build your own game.
The report should contain the following information:

1. Your architectural choice on how to implement the application logic.
2. Threat model description, i.e. what an adversary is supposed to be able to do.
3. At least one screenshot showcasing how you call your contract (with a transaction), and what the return result is.

5.5 Presentation

The 5 best cryptomon projects will have the honour to present their results to their fellow students. The presentations should last about 10-15 minutes, discuss the difficulties, the easy parts, and architectural choices.

6 Paper Reading/Presentation

The blockchain field is only 10 years old, and heavily driven by academic research results. To stay at the edge of this nascent technology we will review/discuss several academic, peer-reviewed papers (tentative list):

- Bitcoin: A Peer-to-Peer Electronic Cash System by Sathoshi Nakamoto (`https://bitcoin.org/bitcoin.pdf`)
- On the security and performance of proof of work blockchains by Gervais et al. (`https://eprint.iacr.org/2016/555.pdf`)
- Do you need a Blockchain? by Gervais et al. (`http://doyouneedablockchain.com/`)
- SoK: Off The Chain Transactions by Gudgeon et al. (`https://eprint.iacr.org/2019/360.pdf`)
- On the Privacy Provisions of Bloom Filters in Lightweight Bitcoin Clients by Gervais et al. (`https://ethz.ch/content/dam/ethz/special-interest/infk/inst-infsec/system-security-group-dam/research/publications/pub2014/acsac\_gervais.pdf`)

Reading the papers prior to the supporting lectures within which they are presented is optional but helpful for your understanding.

Part III
Exam

The exam will cover the topics and problems discussed within the lectures (i.e. not necessarily capture all contents of the papers that are being discussed/read, but the parts that are discussed within the lecture). We will have two lectures where we will work on (i) generic exam questions, and (ii) the exam of 2019. Those exercises are planned to be worked on together and discussed if there are any questions arising.