Blockchain Technologies and Applications 2021 homework assignments

Version 1.0

Contact: Imre Kocsis Kocsis.imre@vik.bme.hu

General remarks

- You have to choose a homework assignment from the below set and decide whether you want to solve the homework using ***Solidity*** or ***Hyperledger Fabric*** (using one of the supported chaincode languages).
- We will make available a form to apply for the homework assignments of your liking. After that, we will publish the assignments determined by us (based on your preferences and the already determined assignments).
- All assignment are partially underdefined. You are expected to interpret the assignment and "fill in the blanks".
- You are expected to submit (as a single zip file):
    - Written documentation of
        - the design decisions you took,
        - the API of the smart contract,
        - the important implementation details (if any),
        - the definition and implementation of test cases.
    - The smart contract in a compilable and deployable form, with instructions to run the implemented test cases.
    - Test cases.
- You are NOT REQUIRED to create a full "DApp" (full distributed application which also has a client front-end implementation) – the homework focuses on the smart contract layer.
- Please be mindful of the fact that most assignments require some form of participant management. The usual "style" of this differs on the different implementation platforms.
- The same applies for time – e.g., in Solidity, you have only "block time" (time passed is computed – roughly - from block depth difference and the roughly known block time of the consensus).
- We will publish development environment suggestions shortly. You are ***not*** required to deploy your solution "in production"; the development environment suggestions describe the expected style of execution for testing.
- Naturally, you can submit your homework in Hungarian – but we do accept English, too. (The reason that all our material is in English is obviously so that we don't have to translate everything every time when it's used in other contexts.)

Solidity specifics

For Solidity, we recommend submitting a Truffle project (see guide), which includes the contract(s) and the test cases and can be easily executed. The guide we created for the course: ftsrg.mit.bme.hu/blockchain-ethereumlab/guide.html#development-and-test-environments

Hyperledger Fabric specifics

For Fabric chaincodes, we recommend using the Visual Studio Code environment (https://code.visualstudio.com/), with an installed IBM Blockchain Platform extension (https://marketplace.visualstudio.com/items?itemName=IBMBlockchain.ibm-blockchain-platform). The extension also contains embedded tutorials for various topics on its home page (in VS Code). If you have trouble installing the environment, reach out to Attila Klenik.

HF1 Safe crossings for autonomous cars

Unguarded level crossings will pose a special challenge for autonomous vehicles; from the safety point of view, relying only on the on-board sensor packages and intelligence to decide whether it is safe to cross (no train is approaching) will be problematic. At the same time, the railway infrastructure – commonly partitioned into blocks https://en.wikipedia.org/wiki/Railway_signalling#Block_signalling – "knows" where the trains are; e.g., safety systems ensure that no train can enter a block already (or still) occupied by a train.

The task is to design and implement a train crossing smart contract with the following features:

1. The railroad infrastructure must periodically signal the crossing to be in a "FREE TO CROSS" state. This state has a preset validity time; if the last update is older than that, the crossing must be assumed to be in a "LOCKED" state. This can happen either on a train approaching or a failure of the infrastructure.
2. Autonomous vehicles wanting to cross must request permission to do so.
3. Permission may be granted only if the intersection is not in a "LOCKED" state.
4. Additionally, an intersection comprises of one or more lanes. A single lane can accommodate a fix number of crossing cars, predetermined by the railroad infrastructure managing authority.
5. Autonomous vehicles must explicitly release their permission after leaving the crossing. Failure to do so will later involve legal action; for this purpose, their identity must be recorded on the ledger, but in a privacy-preserving way (as much as possible).
6. As an additional safety measure, using a means of communication independent from the one used by the above mentioned infrastructure, approaching trains also explicitly request the crossing to get into the "LOCKED" state (and release this signal only when they passed it).
7. However, if the intersection is still occupied, it transitions into a special state (also signaling this to the train) where the train will have priority to request crossing, i.e., no more cars are granted crossing permission, until the train crosses.
8. If the train can't gain permission in a predetermined time since the original try, it is assumed that there is an obstacle in the crossing, and the train can break or halt.

Note: this exercise does reflect some of the concepts used in safety critical engineering, but falls far from a full, real-life safety strategy. (I.e., don't build a real system from this specification. If you happen to be a rail fan, https://arxiv.org/abs/1901.06236 and https://www.deutschebahn.com/en/Digitalization/technology/New-Technology/blockchain-3520362 are good reads.)

Homework owner: Attila Klenik (attila.klenik@edu.bme.hu)

HF2 AutoQuarantine

Design and implement a smart contract which helps the authorities to check whether citizens adhere to quarantine orders.

1. Citizens under quarantine are registered in the contract by the designated health authorities with their quarantine GPS coordinates.
2. After registration, the police force can randomly request citizens to “check in” (but at most 4 times a day).
3. Check-in involves the smart phone of the citizen recording its GPS coordinates on the ledger with a timestamp (naturally, the smart phone app functionality is out of the scope of the homework; mobile network attested cell information and mandatory fingerprint-based authentication can provide security that’s far from being bullet proof, but statistically most probably “good enough”).
4. The check-in involves a time limit; citizens who fail to check in within the time limit or leave their quarantine zone (with some tolerance threshold) are automatically flagged by the smart contract.
5. If a citizen had to leave the quarantine due to some emergency, he/she can check in within the time limit, attaching the emergency reason, flagging the citizen for further verification and suspending his/her check-ins for the remainder of the day.
6. Only the police force can lift the flagged status and only the health authorities can remove a citizen from the quarantine.
7. GPS coordinates should be handled in a privacy-protecting way. (In this regard, you can specify additional functionality for the smart phone app, but not out of band communication w.r.t. the distributed ledger.)

Homework owner: Attila Klenik (attila.klenik@edu.bme.hu)

HF3 L(a)unch codes

A high security facility always houses a shift of two soldiers. It must also provide regular access to low security clearance staff (food delivery, cleaning, …). All entries and exits are tracked and authorized by a distributed ledger (a tamper-proof electronic lock on the main entrance continuously monitors the ledger and decides whether it should open or close; requests and authorizations are supported by smart cards and electronic terminals).

1. Entry is requested at the outside and must be authorized by both soldiers on duty.
2. Successful entry must be logged inside by the entering party (after the door was closed).
3. The protocol for exits is the same in the reverse.
4. Shift changes happen in two phases (first soldier1' replaces soldier1 in a full entry-exit cycle, then soldier2' replaces soldier2).
5. Guard duty is transferred inside by a mutual "acknowledgement" of the involved two soldiers.
6. There must not be more than 3 persons in the facility at any time.
7. Shift change must take place when the facility is empty, and no entry is allowed until it is over.
8. A soldier cannot enter or exit the facility while he/she is on guard duty.

Design and implement a smart contract supporting the above access management protocol.

Homework owner: Attila Klenik (attila.klenik@edu.bme.hu)

HF4 Food chain

Traditional food supply chains often suffer huge losses when it turns out that the origin of the product is contaminated, because they must annihilate tons of potentially affected products, not to mention the costs of tracing them.

Your task is to design and implement a smart contract that allows tracking the route of each product from farms through factories and wholesalers to retailers. The system shall consist of the following parties:

- farmers who register the resources on the blockchain,
- shippers who register product transitions (e.g., from farm to factory, etc.),
- factories that register products linked to the resources they are made from,
- wholesalers who can split the chunks of products arriving from a factory into smaller units,
- and retailers who can sell the final products.

If any parties mark one of their products spoiled, the system must trace and mark all other assets that may have been affected (typically the ones that originate from the same source, e.g., the ones that were shipped together). Parties can only mark products that are either produced by them or are currently in their possession. The contamination/defect report may also provide information about the type (location) of the defect: farm-related, factory-related, shipment-related, etc. If such information is available, the system should only trace the routes starting from the source of the defect (e.g., if a factory is the source of a defect, not all products shipped together from the same farm should be marked).

Note: This assignment requires on-chain computation of transitive closure. Be aware of its implications (e.g., related to gas usage)! Naturally, we would most likely *not* do this directly on the Ethereum mainnet (where computation costs “money”), but the applicable techniques (one party computing the forward traces off-chain and providing a succinct proof of correctness, reaching agreement on the affected products, …) are beyond the scope of a homework. For consortial networks, this should be a lesser concern, but do provide an estimate (and possibly tests) for the problem sizes which are deemed manageable.

Homework owner: Balázs Prehoda

HF5 Vaccination slot

Design and implement a “vaccination slot” utility token for a (single) medical station.

Vaccination slots are issued for specific occasions (at specific dates) to specific patients by the doctors. Patients are allowed to swap their slots with others, but only those patients can take part in these transactions who already own a valid token. Doctors “burn” the tokens as the vaccination of the owner occurs. No patient can hold more than one vaccination slot for any occasion at any time. Provide facilities for the patients to be able to “swap” vaccination slots in an atomic way. Take into account that the doctors use multiple types of vaccines, some of which have to be administered multiple times, within some (vaccine-dependent) time interval after the first shot.

Note: while this token *is* kind of a non-fungible one, a full ERC-721 implementation would be an overkill. That being said, it is worthwhile to look at the various Ethereum token standards at least for inspiration.

Homework owner: Balázs Prehoda

HF6 Electric scooter sharing

Design and implement an ERC20 compliant usage right utility token for a university-wide electric scooter sharing program. (If your platform is not Solidity, adapt the ERC20 specification accordingly.)

New scooters get into the system by their owners registering them (requires the attestation of at least three existing scooter owners); we can assume that the cooperative running the program equips registered scooters with electronic gear locks which can monitor a distributed ledger wirelessly as well as communicate with a smart phone app over Bluetooth for periodic identity checking.

There are two kinds of users in the system: those who are also scooter owners and those who are not. Non-owners must pay a predetermined fee for the releasing party for the usage right, if she/he is an owner; otherwise, they pay “into the contract”. This unlocks the scooter for 30 minutes; for each additional 30 minutes of unlocking, they must pay some fee “into the contract”. Owners can get the usage right transferred to them fee-free and do not have to pay for unlocking the scooter. Each user must have at most one scooter usage right at any time; scooters that have not been used (have been being in a locked status) for an hour can be acquired without the original holder transferring them (the above fee scheme still applies).

In addition to the above fee scheme, non-owners may “sell” their usage rights to non-owners for cryptocurrency (or some additionally created unit of value, if the platform is not Solidity).

The collective of the owners may decide – by majority voting – to divide up among themselves and “transfer out” the fees collected by the contract.

Homework owner: Bendegúz Gyönki

HF7 Non-fungible token

Non-fungible tokens can represent ownership over digital or physical assets. These tokens are distinguishable, and you must track the ownership of each one separately.

Design and implement an ERC721 compliant digital image token. (You can find the associated Ethereum Improvement Proposal here: https://eips.ethereum.org/EIPS/eip-721 If your platform is not Solidity, adapt the ERC721 specification accordingly.)

Each digital image token shall have a unique identifier and an URL (the location of the image). Users can create new tokens and then buy and sell them among themselves (in Solidity, for Ether; otherwise, create some sort of “unit of value” on the ledger that we assume the participants accept). Each token should have one owner at any time. The number of tokens a user may own is not limited.

Homework owner: Bendegúz Gyönki

HF8 YeastBid

At certain times, producers of yeast cannot keep up with user demand. Yeast is produced in large blocks of predetermined size; the producer decides to ~~make extra profit~~ regulate the market by accepting bids for a quantity (in grams) of yeast at some bidder-determined unit price (HUF/gram) from each large block of yeast. Bidding is performed in three phases. First, bidders register the hashes of their bids (salt + quantity + price). In the second phase, bidders register their bids ("reveal" them by showing the original content they registered the hash of). Third, the yeast producer registers the bids they accept and the bids they refuse. The accepted bids together must be an optimal (cumulatively, maximum financial value) solution to the packing-like problem of maximally covering the volume of the block of yeast with the bid-volumes. If any bidder can show a higher-value valid bid set within a predetermined time frame (thus proving that the yeast producer is not behaving fairly), the results are automatically invalidated, and the process starts again.

Design and implement a smart contract that implements this bidding process.

Homework owner: Imre Kocsis (Kocsis.imre@vik.bme.hu)

HF9 Simple factoring

Design and implement a smart contract that emulates a simplistic form of factoring (a type of debtor finance): parties create invoices with due dates (with the digitally signed agreement of payer as well as payee) and the beneficiaries of the invoices are allowed to sell the invoices before their due dates at a discount. Further re-selling is also allowed. There should be mechanisms to split and merge invoices of the same debtor with the same due date. Also design a mechanism where a debtor may request due date extension for a proposed fee and the current holder of the debt can either accept or reject the proposal.

When the due date is reached, further operations on the invoice are not allowed, only their payment.

With respect to payment operations: for Solidity, use Ether; otherwise, create some sort of “unit of value” on the ledger that we assume the participants accept.

Homework owner: Imre Kocsis (Kocsis.imre@vik.bme.hu)