Integrated MSc Course on Informatics Engineering, DI/FCT/UNL
Computer Networks and Systems Security / Semester 1, 2019-2020

Practical Evaluation: Work Assignment #2

Secure Blocchain-Enabled Auction Management System (version 1.0)

Submission period: From 9 to 15 December/2019
Deadline for submission: 15/December /2019, 23h59

Summary

The objective of this project is to develop a system enabling users to create and participate in auctions, with the bids enabled on a Blockchain ecosystem. The system will be composed by an auction manager, an auction repository that can be leveraged by a decentralized blochchain repository and the client applications. The system should be designed to support the following base security features:

- *Bids' confidentiality, integrity and authentication: Bids may contain secret material which can only be disclosed in special occasions, cannot be modified once accepted and cannot be forged on someone else's behalf. Peer-authenticity requirements of such bids is required.*

- *Bid acceptance control and confirmation: bids can only be accepted if fulfilling special criteria (in terms of bids' structure). A source quenching mechanism must be used to reduce the amount of submitted bids, and accepted bids are unequivocally confirmed as such;*

- *Bid author identity under anonymity criteria: Bids should be linked to subjects using public-key certificates, which can be managed anonymously (wilt e possible use of pseudonyms for example), but with some kind of bindind proof between such pseudonym identifiers and real identifiers, if and when required for some reason (ex., in a dispute). However, submitted bids should remain anonymous until the end of the auction.*

- *Honesty assurance: The auction repository must provide public access to all auctions and their bids, either finished or still active, providing evidences of its honesty on each answer to a client request. Furthermore, the auction repository cannot have access to any kind of information that may enable it to act differently to different clients.*

Beyond the above security guarantees, students are free to propose and implement other security features considered useful, as well as, extended security features from the base required security properties

1. System model assumptions

1.1 System Components

As far as messaging systems go, we can consider the existence of two main components:

Client (with the existence of possible multiple clients, through which users interact; Two servers, which serve as rendezvous service points for all clients to connect.

1.1.1 Auction Server

This server will expose a REST connection endpoint, protected by TLS, through which clients can exchange structured requests/responses with it. The TLS in this interactions can be configured in the server endpoint in order to support server-only TLS authentication or mutual authentication.

The Auction Server is the system management component that creates an auction upon a client request. When a request is done, the Action Server instantiates a new auction in the Auction Repository.

The Auction Server is also the component that may perform special bid validations requested by the auction creator. For instance, the auction creator may limit the number of bidders to a given set of identities, restrict the number of bids performed by each identity and other validations suggested and implemented by students as relevant validations. Ideally, such validations should be performed by dynamic code uploaded to the Auction Manager at the time of the auction creation, a feature that will be interesting in the evaluation of the found solution.

The bid validation process implemented by the Auction Server may change or add some fields of the bid, namely encrypt them. In any case, the Auction Server cannot, in any case, modify the original intents of a bid (in its essential data), and all encrypted fields must, at the end of the auction, be publicly exposed.

1.1.2 Auction Repository Server

This server will expose a connection endpoint through which clients can exchange structured requests/responses with it. The Auction Repository will store a list of auctions. Each auction is identified by a (possibly short) name, a unique serial number, a time limit for accepting new bids and a description. Each auction must be implemented by a blockchain, with a bid per block. A blockchain ihere is nothing more than a sequence of blocks were the last one ”seals” the previous sequence of blocks, making them immutable thereafter. A possible solution for this data-structure can be a Merckle

Tree, in a similar way as real Blockchains also implement.

In practice, a blockchain can also be implemented as an ordered linked list of blocks where each block contains a cryptography hash of the previous block- chain (i.e., the one existing before its insertion).

The Auction Repository Server closes an active auction upon a request made by the Auction Manager or upon reaching the auction's time limit.

Clients send new bids directly to the Auction Repository. The rate at which bids are sent can be controlled by a mechanism called cryptopuzzle, or proof- of-work. A cryptopuzzle is a task that is hard to perform, while the result of such task is easy to validate. A client willing to send a bid first asks for a cryptopuzzle, then solves it, incorporates the solution in the bid and sends it to the Auction Repository. This checks the solution of the cryptopuzzle, optionally sends the bid to the Action Repository for being validated, adds the bid to the auction blockchain and sends a receipt proving that the bid was added to the auction.

1.1.3 Auction Client

An Auction Client is a front-end application that interacts with a user, enabling they to create and participate in auctions. This application needs to interact with the user Citizen Card in order to authenticate auction creation/termination requests or bids.

For each bid added to an auction, the Auction Client must store its receipt in non-volatile memory for an `a posteriori validation. This is fundamental for preventing both servers from cheating by manipulating the sequence of bids in a auction.

1.2 Enabled Processes

There are several critical processes that must be supported. Students are free to add other processes as deemed required.

> • Create/terminate an auction;• List open/closed auctions;• Display all bids of an auction;• Display all bids sent by a client:• Check the outcome of an auction where the client participated; • Validate a receipt.
>
> Several types of auctions may be predefined. The following ones are mandatory, but others are also possible:
>
> > • Open ascending price auction, a.k.a English auction. Each bid must overcome the value of the previous one. The minimum (or maximum) extra amount of a new bid can be enforced by dynamic code uploaded to the Auction Manager when the auction is

created.

Bids for this kind of auction should have a cleartext value and an encrypted identity, which can only be revealed by the Action Manager (upon the end of the auction). Everyone, at the end of the auction may be able to check if the Auction Manager did not cheat.

• Sealed first-price auction or blind auction. Bid amounts are encrypted, while identities may either be exposed or not. At the end of the auction all bids are decrypted by the Auction Manager, yielding the auction winner.

1.3 Messages in the system

It strongly suggested to structure all exchanged messages as JSON objects. JSON is a very user-friendly textual format and there are many libraries for building and analysing JSON objects. Binary content can be added to JSON objects by converting them to a textual format, such as Base-64.

2 Additional Suggestions

A simple, while effective way of implementing cryptopuzzles is the one used in the Bitcoin blockchain. Each new block must contain the hash of the current blockchain and its hash must belong to a given set of values (e.g. lower than a threshold). The set of values if defined by the blockchain keepers in order to enforce a maximum update rate in the blockchain. The solution for this kind of cryptopuzzle requires brute-forcing for a solution using a random value (a counter is enough) in the block structure.

Both the Auction Manager and the Auction Repository are easier to im- plement with UDP/IP requests/responses (very much like DNS servers and remote file systems do) or, alternatively, with HTTP requests/responses. In both cases, they permanently wait on a single communication endpoint for client requests, handle them and send back an answer before tackling a new request. Multiple requests may be handled in parallel using threads, but that requires synchronization to deal with concurrency and complicates debugging.

Both servers should keep their status in permanent storage, to overcome failures. However, this is not mandatory.

3 Functionalities and evaluation criteria

The following functionalities, and their grading, are to be implemented:

- (2 points) Protection (encryption, authentication, etc.) of the messa- ges exchanged;
- (2 points) Protection of the bids until the end of their auction;
- (2 points) Identification of the bid author with a Citizen Card;
- (2 points) Exposure of the necessary bids at the end of an auction;
- (2 points) Validation of bids using dynamic code;
- (2 points) Modification of validated bids by dynamic code;
- (2 points) Construction of a blockchain per auction;
- (2 points) Deployment of cryptopuzzles;
- (2 points) Production and validation of bid receipts;
- (2 points) Validation of a closed auction (by a user client); To simplify the implementation, you may:

Initially, students can assume the use of well-established, fixed cryptographic algorithms. In other words, for each cryptographic transformation you do not need to describe it (i.e., what you have used) in the data exchanged (encrypted messages, receipts, etc.). However a bonus of 2 points may be given if the complete system is able to use alternative algorithms. And to have the maximum flexibility in supporting the configuration of all the involved cryptographic constructions.

Evaluation bonus can take place for good and original ideas from student or highlights reported and implemented, with the design foundations argued on consistent argumentation. Students can also discuss their ideas with the Professor, checking also for challenges and realism.

It can be assumed that each server has a non-certified, asymmetric key pair (Diffie-Hellman, RSA, etc) with a well-know public component. Grading will also take into consideration the elegance of both the design and actual implementation. Up to 2 (two) bonus points will be awarded if the solution correctly imple- ments interesting security features not referred above. A report should be produced addressing:

- the studies performed, the alternatives considered and the decisions taken;
- the functionalities implemented; and

- all known problems and deficiencies.Grading will be focused in the actual solutions, with a strong focus in the text presented in the report (4 points), and not only on the code produced.

It is strongly recommended that this report clearly describes the designed and implemented solution.

Using materials, code snippets, or any other content from external sources without proper reference (e.g. Wikipedia, colleagues, StackOverflow), will imply that the entire project will not be considered for grading or will be strongly penalized. External components or text where there is a proper reference will not be considered for grading, but will still allow the remaining project to be graded.

4. Delivery Instructions

The delivery instructions will be the same as presented for option TP2a.