MIECT: Security 2018-19

Blockchain-based auction management

Presentation of final project: 1st and 2nd weeks of Jan, 2019
Deadlines: Dec 31 (final project)

André Zúquete, João Paulo Barraca

Changelog

- v1.0 - Initial Version.

1 Project Description

The objective of this project is to develop a system enabling users to create and participate in auctions. The system is composed by an auction manager, an auction repository and client applications. The system should be designed to support the following 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;
- **Bid acceptance control and confirmation**: bids can only be accepted if fulfilling special criteria (in terms of 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 and anonymity**: Bids should be linked to subjects using their Portuguese Citizen Card. 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, and provide 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.

2 Project Description

2.1 System Components

As far as messaging systems go, we can consider the existence of two main components: (i) multiple clients, through which users interact, and (ii) two servers, which serve as rendezvous points for all clients to connect.

2.1.1 Auction Manager

This server will expose a connection endpoint through which clients can exchange structured requests/responses with it.

The Auction Manager is the system component that creates an auction upon a client request. Upon such request, the Action Manager instantiates a new auction in the Auction Repository.

The Auction Manager 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, etc. Ideally, such validations should be performed by dynamic code uploaded to the Auction Manager at the time of the auction creation.

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

2.1.2 Auction Repository

This server will expose a connection endpoint through which clients can exchange structured requests/responses with it.

This component 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 is a sequence of blocks were the last one "seals"the previous sequence of blocks, making them immutable thereafter.

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

The Auction Repository 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.

2.1.3 Auction Client

An Auction Client is an 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 à posteriori validation. This is fundamental for preventing both servers from cheating by manipulating the sequence of bids in a auction.

2.2 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.

3 Messages

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.

4 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 implement 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.

5 Functionalities to implement

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

- (2 points) Protection (encryption, authentication, etc.) of the messages 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:

- 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.). **A bonus of 2 points** may be given if the complete system is able to use alternative algorithms.
- 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 implements 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 solution proposed for each functionality.

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.

The detection of a fraud in the development of a project (e.g. steal code from a colleague, get help from an external person to write the code, or any other action taken for having the project developed by other than the responsible students) will lead to a grade of 0 (zero) and a communication of the event to the University academic services.

6 Project phases

The security features should be fully specified prior to start its implementation.

We recommend the following steps for a successful project development:

- Develop a complete, non-secure client and server applications. This step can start immediately.
- Produce a draft report of the security specification.
- Involve the Citizen Card in the auction creation and biding processes and use its certificates to extract user identity information.
- Add security to bid receipts.
- Implement bid encryption procedures performed by the Auction Manager.
- Protect the communication between all applications.
- Add support to dynamic code for validating bids.
- Implement cryptopuzzles.

There will be a **project milestone in November, 15 and 16**. In this milestone students should get feedback about a **written overview** (through their CodeUA project, see Section 7) of the security mechanisms they have designed (not implemented!) for their project. This milestone **does not involve any grading**, and will serve to assess the progress of students and to give them some feedback. Since the feedback will be given based on the

written contents, they must be stored in CodeUA a couple of days before the milestone date. Please send an **email to the course teachers** upon the delivery of the overview.

7 Delivery Instructions

You should deliver all code produced and a report before the deadline. That is, 23.59 of the delivery date, December 30, 2018.

In order to deliver the project you should create a project in the CodeUA[1] platform. The project should be named after the course name (e.g. `security2018`), the practical class name (e.g. `p2`) and the group number in the class (e.g. `g5`), with yields the following complete format for the examples given before: `security2018-p2g5`. **Please commit to this format to simplify the evaluation of the projects! Outliers will be penalized!** Each project should be given access to all the course professors (André Zúquete).

Each CodeUA project should have a `git` or `svn` repository. The repository can be used for members of the same group to synchronize work. After the deadline, and unless otherwise requested by students, the content of the repository will be considered for grading.

[1] `https://code.ua.pt`