Blockchain-Based Auction Management

Authors: Filipe Pires (85122) & João Alegria (85048)
Date: 15/11/2018

Message Protection

RSA Asymmetric Key-Pair Strategy, both sides have a key-pair (public and private) - autogenerated (on start/restart) by the server and accessed from the CC by the client (see Author Identification)- accessible only by themselves.
Client signs request message and sends signed message with signature public key all encrypted with server's public key. Server decrypts confidential message and is the only one with access to client's public key. Server sends challenge to client using client's public key. Client responds to the challenge encrypting the answer with the server's public key. Secure communication channel is now established.

Bid Protection (until end of auction)

AES-128-CBC Cipher Strategy, where each bid is encrypted and xored with previous bid (or with IV). IV is generated and stored by the Manager (for each auction). Each auction has its own AES private symmetric key, stored in a database, accessible only by the Repository. This allows for the creation of the secure bid blockchain (see Blockchain Creation).

Author Identification (with a CC)

CC is crucial in the message encryption process. In order for the client to use the server, it needs to have access to his own CC – the SmartCard's Authentication Key-Pair – to help with the encriptions needed to communicate with the server.

Bid Exposure (at the end of auction)

In order to expose the necessary bids at the end of an auction, the storage of the auction's IV (by the Manager) and private key (by the Repository) must be assured. This allows for the decryption of the bid-sequence cryptogram (see Bid Protection) from the end to the desired bids, using the stored elements previously mentioned.

Bid Validation (with Dynamic Code)

When a client decides to create an auction, he may choose to include a validation function. This function goes inside a string with a specific format and has the purpose of limiting the bids allowed and their characteristics. When receiving a bid, the Repository asks the Manager to validate it – this may including running the auction's author's validation function.
Concerning the security of the executable string, if the creation of the auction comes with it, the string is lexically analyzed right from the start (assuring its correct structure – 1 function, called "validate", with no unauthorized access to the environment).

Valid Bid Modification (with Dynamic Code)

The idea behind valid bid modification lies in the creation of automatic bids. This means that a client has the ability of defining a margin of bid values to which he is willing to go if other clients outbid his current one. The way the autor wants to create this auto generated bids and the conditions he wants to take in consideration must be pass through a string, similar to the string for the validation of the bids, but now the function must be called “generate”. The assurance that this string is valid is made exactly the same way as the validation string (see Bid Validation (with Dynamic Code)).

Blockchain Construction

Regarding the blockchain, each block will contain a bid, the blockchain’s hash until then and (optionally) the auction’s validation and modification functions. At the end of the auction, a closing block is added to the chain, making sure that no more blocks can be added.

When a client does any operation that envolves updating the blockchain, the server sends him a copy of the blockchain until that moment (without giving the decryption private key). This is useful for the client to validate the entire process of the auction once this is finished (see Bid Receipts).

Cryptopuzzle Deployment

A cryptopuzzle is a necessary step in the bid validation process. This sort of puzzle consists of making an concatenation over the received hash of the blockchain’s current state, the client’s bid and a random number, this operation is done by the bidder and needs to be tried until the hash of that concatenation follows the cryptopuzzle’s conditions(changing the random number each time the hash is calculated). These conditions are: the hash string must start with X (X=integer number) characters equal to a given sequence of characters of that size (e.g. X=3, sequence=”abc”, hash string=”abck3jif94fk49 (...)”).

Bid Receipts (production and validation)

When a bid is considered valid and is introduced into an auction, the Repository generates a receipt and sends it through the communication channel of the respective client. This receipt has a specific and secure format: it contains the auction’s id, the bid’s id, the value and the digest of the bid’s author’s name with his public key, all of these digitally signed by the server.

The user validates this receipt by decrypting with his private key, then decrypting again with the server’s public key, analysing the result and checking if the digest result is equal to his digest with the same arguments. The digest function is known by both sides.

Closed Auction Validation (by client)

All bids are visible by any client, once the auction is closed and finished. This is done by sending the entire blockchain to the client along with the decryption key and IV. This allows for the participants to check if the results are correct.

Notes

Exporting a CC’s public key will be crucial in this implementation, and the private key will be used for security assurance (without ever being exported, this being in fact impossible to do).
Only the client can validate whether a key is from a CC or not. He does this with the help of the respective Certificate Authority.