**5 Experimental Evaluation**

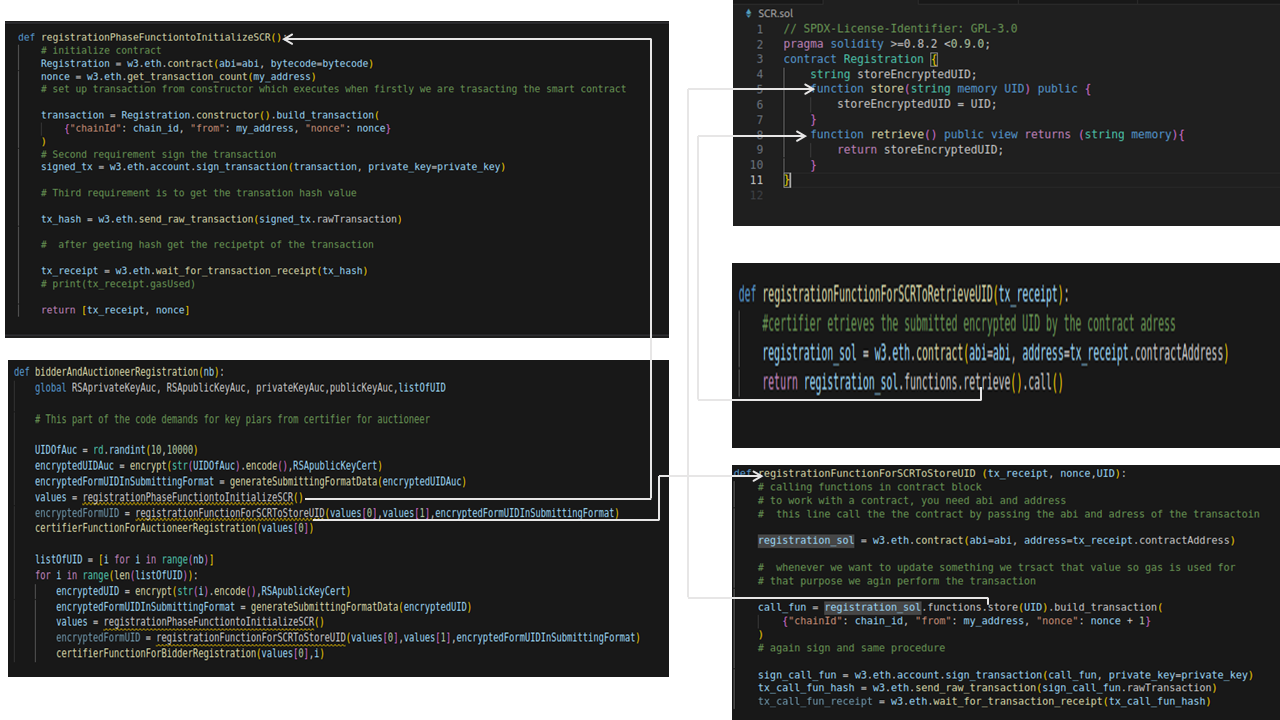
The experimental evaluation is categorized in two subgroups in first group we look at the different function with their input, output and descriptions and in other groups we look at the implementation of different algorithms with smart contracts. The implementation and testing of the proposed hybrid privacy preserving auction scheme has been performed over the intel i7-12700 12 gen processor with 32GB ram and with max clock speed of 4900MHz. The software suite that we have used is Ganache[reference] for simulating Ethereum blockchain on local node together with web3.py[reference] library to call and run smart contracts inside the python code. The off-chain computation is simulated in python programming language on local node.

**Registration phase on-chain and off-chain data handling**

On registration of any new bidder a new call to the smart contract is sent with the of the bidder encrypted with the certifier public key. This section shows the working of the code and its way of execution and handling of the smart contract, how the off-chain and on-chain data is computed and transferred.

From the function bidder and auctioneer submit the encrypted and this encryption is performed offchain in their own node, and after that initialize the contract by the function , later the tx\_recipet and nonce value is passed as the parameter to the and this function further calls the smart contract and stress the encrypted UID as a form of input to smart contract. One key thing t notice is that the encryption of on the node is generating the ciphertext of the size given as the key length of the encryption and this value can’t be stores directly to the byte variable for that purpose the preprocessing needs to be done of the ciphertext and is converted to the string format and that passed to smart contract otherwise the size limit for byte datatype is only of 32 bytes which is too far short than the length to the ciphertext as in our case it is 1024 bits(128 bytes).

Certifier in returns provides the public and private key value pairs and makes the public key value pairs of all the participants public on the blockchain.

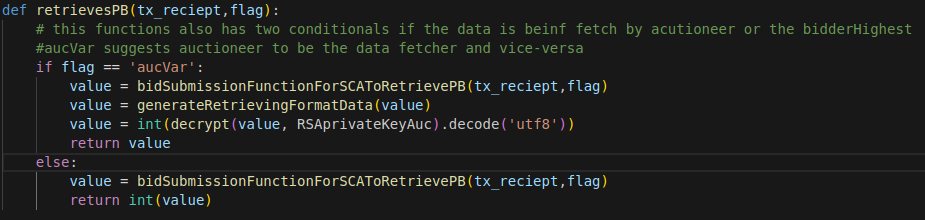
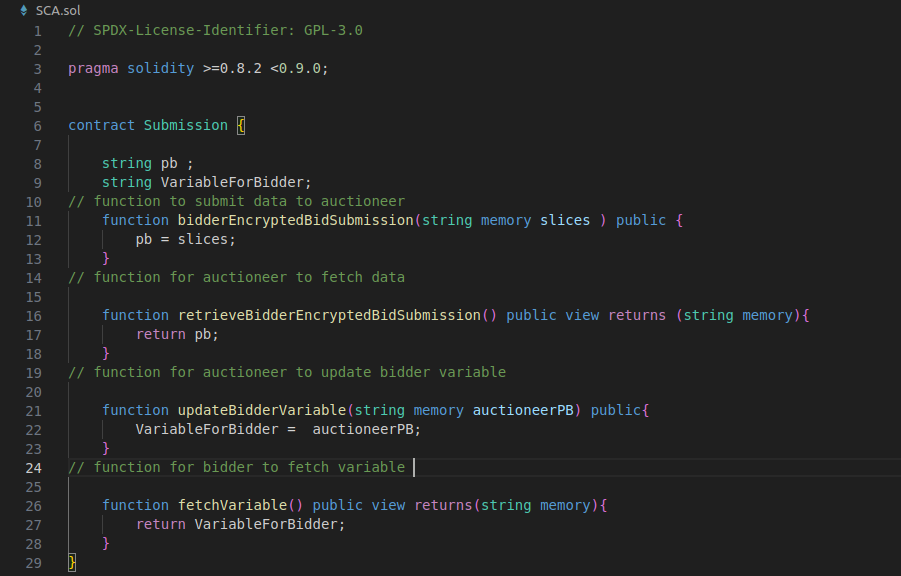
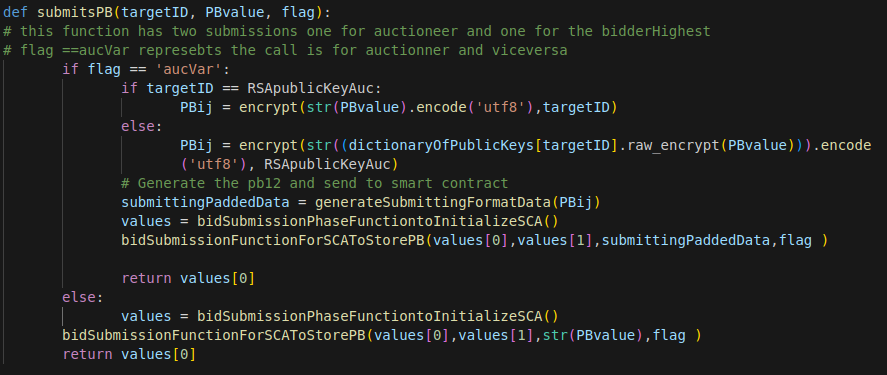


Following figure represents the functions of the certifier to generate the public and private key value pairs for and.



**Bid submission phase off-chain and on-chain data handling**

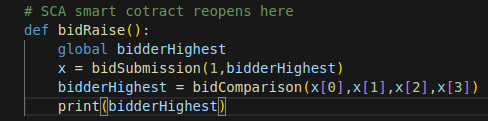
In this section we look at the calling, computations and data transfer related to the smart contract .In this phase the smart contract is considered as the entry point to transfer the encrypted data between the bidders and the auctioneer and vice-versa. consists of four functions. First two functions and are used for transferring data to auctioneer and bidder respectively. Similarly, there are two functions which are used to fetch back the data from the which are and for auctioneer and bidder respectively. This smart contract is also handled by the web3.py library, we have defined that function that directly interact with the functions of SCA and our simulation setup has two different such functions that submit and fetch the data from the smart contract which are and for submitting and retrieving data respectively.



The method that we have used to share the data among the bidder and auctioneer is by sharing the transaction receipt of every transaction to recipient if we are submitting the data to it.

**Bid comparison and winner declaration**

This phase has been implemented with the help of the function that simulated bidding phase for a given number of times named as This functions calls the bid submission phase n number of times with each times highest bidder and a new random bidder being the input , to perform bid comparison.



This function and whole submission is called by the function with the parameter the number of bid raise and the number of bidders to participate.

