Design and implementation of a privacy preserving decentralized application

Karamoustou Vasiliki, Panagiotou Dimitra, Papadouli Vasiliki
Electrical and Computer Engineering, University of Thessaly, Volos, Greece

***Abstract*—During this course we design and implement a privacy preserving decentralized application, upon the Ethereum network. Main focus of this implementation is to protect private data of entities (both consumers and producers) participating in a double auction. For this purpose we design a double auction smart contract coupled with cryptographic schemes to ensure privacy during trading process.**

***Index Terms*—Uniform Price Double Auction, Ring Encryption, Smart Contract Privacy**

I. Introduction

The distributed ledger technology satisfies the need of transparency, openness with high audit degree and enables users to trade commodities in a decentralized manner. Smart contracts especially, can be used as communication channels between multiple producers and consumers by establishing basic communication rules. In such an implementation a double auction smart contract is used, where the demand and offer from consumers and producers, respectively, is stored publicly in the blockchain ledger as transactions.

However, the transparency of the above described implementation may form a security breach. The consumers' data, which are stored publicly, are sensitive and rather private information so, the need for an alternative, privacy-preserving approach is raised. In the implementation proposed by the current work, we used cryptographic schemes such as rings and commitments to mask the quantity and price that both consumers and producers enclose in their bid requests.

Considering the fact that each node has a full record of the data that has been stored on the blockchain, it is highly demanded to apply effective methods to conceal private data.

The rest of this report is organised as follows: Section 2 describes the related work in the area of blockchain, Section 3 and 4 present basic background concepts for this implementation such as the Uniform Price definition and the Cryptographic Preliminaries. Section 5 presents our implementation and Section 6 present the results of our approach. Lastly, at section 7 we discuss a possible configuration for a foreseen application.

II. Related Work

In literature there are lots of related studies concerning applications with privacy preserving schemes build upon public ledgers.

Li, Dong, Cao and Shen in [2] implement a blockchain-based house rental system that uses Bulletproof zero knowledge proof protocol along with Pedersen commitment to hide the bid price for every bidder participating in the auction. Moreover in [1] a blockchain-based systems for car sharing using Zero-Knowledge Protocols was built. In this approach, they leverage zero knowledge proofs and anonymous credentials to tackle security breaches in car sharing market. Both approaches make use of a third-party authority, to cooperate in order to ensure users' privacy.

Concerning the energy market, the majority of e-auctions deal with smart grids and interactions between consumers and generators so as to balance the electricity network. In [3] paper's approach, the authors create a secure privacy-preserving protocol to satisfy double auction implementations. In this paper, they propose a scheme to preserve user's anonymity by assigning a pseudo-identity to every participant on the contract while all bids and offers, from both consumer and generator's party, are encrypted using the Paillier cryptosystem. Finally they apply Pedersen commitment to verify which participant has been honest and correct throughout the auction process.

In our proposed sealed-bid double auction scheme, all auction data were encrypted and stored in the blockchain ledger. The identity of each participant remains unchanged and public, however only the bid offer and the demanded or offered hours are encrypted. Our approach uses a third party authority (smart contract) that focus on collecting the data and performing the calculations for the uniform price and the amount of the commodity that is, finally, offered in the trading process.

III. Uniform price

Consumers and producers are permitted to take part in the market in a way that they can influence the price via a uniform-price double auction mechanism. The way uniform price auction works is the following. After various offers from sellers to the contract are given, it covers buyer's demands starting from the one with the lowest cost, with ascending manner, until supplies run out.This procedure is named "clearing the market". The market price is been set by the last seller selected, called "marginal unit". All sellers with lower offers than the uniform price are going to be satisfied. On the other hand, all buyers asking for prices higher than the market price will be satisfied.

IV. Cryptographic preliminaries

A. Commitment Scheme

A commitment scheme is a basic cryptographic tool that enables one to commit to a specific value or statement without

revealing it to others in case he wants to disclose it in a later time. An important characteristic of commitment schemes is that no one can modify value or statement after somebody has committed to it. This property is called "binding".

A commitment scheme (CS) contains the two algorithms explained below (CS-Com,CS-Ver):

1) Commitment algorithm (CS-Com) with input a bit-string s and output its committal-decommittal pair.
   (com, dec) ← **CS-Com**(s).
2) The commitment verification algorithm CS-Ver with inputs the committal(com) - decommittal(dec) pair and a bit-string s and output a bit.
   b ← **CS-Ver**(com, dec, s).

B. Ring Signature

A ring R is a list consisting of a number of ring members R[i], each one linked with a pair of public and secret keys(pk[i], sk[i]). The ring can be identified by the tuple pk := (pk[1],...,pk[n]). The basic security characteristic of a ring signature is the impossibility of finding the specific key used to generate the signature. In other words, the anonymity of a ring signature cannot be revoked. Additional, ring signatures offer unforgeability, which means that an adversary should not be able to forge a signature on behalf of an honest ring of signers. A ring signature scheme RS contains a bunch of algorithms:

1) The ring generation RS-Gen with input a security parameter k, structures a ring R including the signer as a member and outputs the tuple of public keys and the signer's secret-key.
2) The signature generation RS-Sig with input the ring's public-key, the signer's secret-key and a message m and output its signature ← RS-Sig(pk, sk[s],m).
3) The ring signature verification RS-Ver with input the public-key pk of the ring, a message m and a signature and output a bit b ← RS-Ver(pk,m,).

C. Public-Key Encryption Scheme

A public-key cryptographic algorithm is based on pairs of keys. Every pair contains a public key, known to others, and a secret key, known only by the owner. These pairs of keys are being produced by cryptographic algorithms with mathematical background. Therefore,it is worth mentioning that it offers semantic security and recipient anonymity. In a scheme like this, anyone can encrypt information based on the receiver's public key so, that it can be decrypted only by using the receiver's secret key.

A public-key encryption scheme PKE contains a set of algorithms:

1) The key generation algorithm PK-Gen with input the security parameter k and output the pair of secret and public keys.
   (pk, sk) ← PK-Gen(k)
2) The encryption algorithm PK-Enc with input the receiver's public-key and a message m and output the encryption.
   enc ← PK-Enc(pk, m)
3) The decryption algorithm PK-Dec with input the receiver's secret-key sk and the previous enc and outputs the message.
   m ← PK-Dec (sk, enc)

Fig. 1. Uniform Price Double Auction

V. Implementation

The approach we are going to present below is an extension of the implementation proposed in [4] and [5]. The proposed method implements a smart contract that ensures the privacy of every participant, an auctioneer, a bidder, a participant and an auction class. The auctioneer class is responsible for collecting and decrypting the data. In this implementation, we consider auctioneer as a third party authority that is honest and trusted to enforce the flow of the auction process. The same class also performs the calculations for the commodity quantity that is satisfied during the trading and the uniform price establishing for this specific double auction. The bidder class performs every single calculation which refers to the bidders, such as the commitment or the ring calculation. The participant class is a superclass and it is used to inherit the key generation ability to the auctioneer and bidder classes. And the auction class that stimulates a complete trading cycle, starting from the bid phase till the announcement of the results. Below we describe in detail the operating parts of this implementation.

Smart Contract:The proposed smart contract is separated in three phases: the Place Bid Phase, the Open Bid Phase and the Announce Result Phase. The auction process starts by the auctioneer who sends a deploy transaction to initiate the contract, then every participant in this contract is allowed to make a bid offer, by sending the encrypted commitment of the bid value and the corresponding quantity. For every bidder participating in the auction, is demanded a deposit of 1 eth along with the data to complete its enrollment in the contract. When every participant has placed its bid, the contract continues with the Open Bid Phase. During this phase every participant sends a transaction with the opening token information, which the auctioneer will use to open the encrypted data offered in previous step. The opening tokens

are send in future step to ensure that the opening combination, that finally is used to open the commitments, will remain as little as possible in the public ledger. In the final phase, after the completion of the calculations, the auctioneer sends to the contract the price and the quantity that "clears the market". Participants who had lied during the place bid phase, and their opening tokens did not verify the corresponding commitments are being punished by losing the initial offered deposit. The flow of the trading process is ensured by the smart contract, however all the off-chain calculations are implemented by the 3 classes presented above, the auctioneer, the bidder and the participant class.

Participant: This class generates a pair of keys, the public and the private key for the auctioneer and the bidders. It makes use of the Crypto.PublicKey python library which practicly uses the RSA encryption algorithm to produce this keys. These features are inherited to auctioneer and bidder class.

Bidder: Bidder class contains methods to handle all bidder's operations. In this class the ring generation is issued. Also the bidder class is responsible for the calculation of bid and quantity commitments along with the opening tokens. The basic method inside bidder class is accomplishing a bid offer, as described above and it is presented in the following pseudocode:

```
bid(): All the variables are attributes of the class Bidder
    c1, d1 = commit(quantity)
    c2, d2 = commit(value)
    sigma1 = sign(ring, s, c1) #s stands for bidder's secret key
    sigma2 = sign(ring, s, c2) #s stands for bidder's secret key
    msg1 = concatenate(c1, σ1, quantity, d1)
    msg2 = concatenate(c2, σ2, value, d2)
    C1 = encrypt(msg1, auctioneer_pub_key)
    C2 = encrypt(msg2, auctioneer_pub_key)
    c11, d11 = commit(C1)
    c21, d21 = commit(C2)
    sig = concatenate(σ1,σ2, c11,c21)
    tau_1 = concatenate(C1, d11)
    tau_2 = concatenate(C2, d21)
```

This algorithm outputs the commitment to the bid **c1**, the commitment to the quantity **c2**, the modified signature **sig** = $\sigma 1||\sigma 2||c1||c2$ and the bid opening tokens $\tau 1 = C1||d1$, $\tau 2 = C2||d2$.The modified signature sig is used to obtain a proof that all the required information about the bid value and the identity of the bidder are embedded and that they cannot be changed at later time. The bidder needs to send the bid and quantity opening tokens for the verifier to be able to retrieve the bid value and the quantity, respectively. The ring is used to sign the constructed commitments and to ensure a higher level security.

Auctioneer: Auctioneer class handles every calculation for the decryption of the data and the final estimation of clearing price, clearing quantity and clearing type for a certain trading session. The basic method Auctioneer is accomplishing is bid_opening. It is presented by the submission algorithm bidopening, which is depicted in the following pseudocode:

```
bid_opening( address, ring: List[RSA.RsaKey], c, sig, tau_1, bid_value, bidder_type): Opens the bid value for bidder at address and stores it.
param bidder_type
param bid_value
param address: Address of the bidder.
param ring: Ring of public keys used by the bidder for the Ring Signature.
param c: Commitment to the bid.
param sig: Ring Signature to the bid.
param tau_1: Bid opening token.
return: Whether the bid opening was successful.
status = False

#initialise bidders attribute
self.bidders[address] = { 'quantity': 0, 'bid_value': 0, 'bidder_type': -1, 'status': status }

sigma1,sigma2, c11,c21 = parse(sig)
if verify(c1, sigma1, ring) and verify(c2, sigma2, ring):
    C1, d11 = parse(tau_1)
    C2, d21 = parse(tau_2)
    if commit_verify(C1, d11, c11) and verify(C2, d21, c21) :
        m1 = decrypt(C1)
        m2 = decrypt(C2)
        c1_tilde, sigma1_tilde, quantity, d1 = parse(m1)
        c2_tilde, sigma2_tilde, value, d2 = parse(m2)
        if c1_tilde == c1 and sigma1_tilde == sigma1 and if c2_tilde == c2 and sigma2_tilde == sigma2:
            if commit_verify(quantity, d1, c1) and commit_verify(value, d2, c2):
            self.bidders[address] = {'quantity': quantity, 'bid_value': value, 'bidder_type': bidder_type, 'status': True }
            status = True
return status
```

The bid opening algorithm is used to verify that the bid is valid, and to open its value. The bid opening token generated by the bidder is required. If the signatures and the commitments are successfully verified, the verifier locally stores the bid valuealong with its corresponding opening value d and the quantity along with its opening value.

Auction: Auction class contains the proof of concept for this implementation. Initially it generates a json text file that contains bidder's information (bid value, quantity and bidder type). At the same time the connection with a local ethereum testing network is established, using the Truffle and the Ganache software. The auctioneer sends a deploy transaction to initiate the contract. Then for every bidder participating in

the contract, auction class calls the bid method from Bidder class to generate the commitments and the opening tokens. After every calculation has been completed, all bidders send transactions to the contract containing the commitments along with a predetermined deposit. Later the participants send the opening token for the previously sent commitments. The auctioneer uses all collected data to decrypt the quantity and the bid value in order to calculate the quantity and price that "clear the market". During this process the auctioneer also checks the integrity of the participants and if there are participants who had trade dishonestly, they are submitted to corresponding punishment (established by the smart contract rules, in this case withdraw of the initial deposit). Finally the auctioneer announces the results and a full trading process has been completed. It is important to note that the address of consignor in the transactions of each bidder with the contract, it is not concealed, however it is extremely difficult for malicious users to link this address with a person since the data with which he/she interacts with the smart contract are encrypted.

The source code pertaining to our implementation can be seen at our github page: https://github.com/DimitraPanagiotou/sealed-double-auction

VI. Results

In this section we present the ability of the application to handle a growing number of users. The graph shows that this implementation can manage a considerable number of bidders with no important effect on the time needed for the auction to be completed. Generally, the addition of 10 extra bidders seems to double the total performance time. Particularly, having 100 users, interacting in the auction process, the required time is approximately 4510 seconds. In respect of this fact, our approach can be considered efficient for challenging applications. The scalability of the implementation is shown in figure 2.

The keys generated in our approach are 2048 bits long.

The code has been tested on Dell-inspiron -5559 GL -Processor core i7 -6500U -2.50GHz -8GB Ram -1 TB HDD -VGA 4GB RADEON -DOS -15.6 inch, with Python 3.8.0, Solc 0.7.4 and Ganache 2.5.4.

VII. A Foreseen Application

The energy power forms a combination of retail and wholesale markets. The definition of retail market involves the electricity sale process to consumers while the wholesale market refers to electricity trade process between utility providers and traders. In our approach we refer to energy market as a collection of both consumers and generators. Generally energy market comprises one of the most lucrative and expanding market. Statistics show that during 2018 , the global energy market generated around 1,341.6 dollars billion, with a growth of 2.4% in four years. During next years this percentage is expected to rise at 3.5% by the end of 2022 for a total of around 1,537.5 dollars billion.

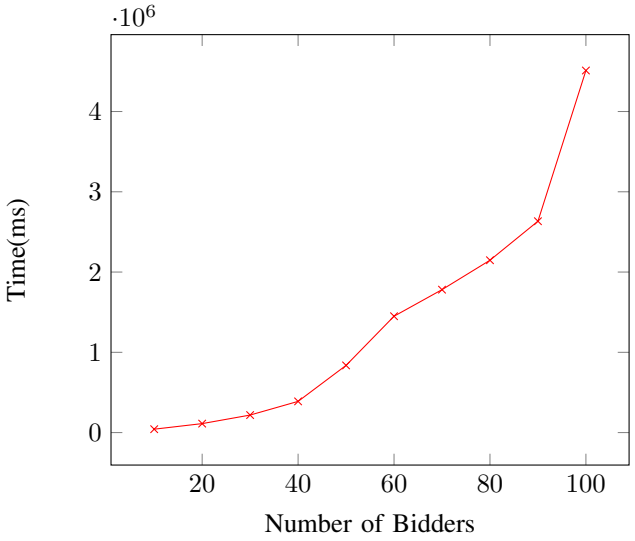

Fig. 2. Time to complete an auction with varying number of bidders.

Nowadays there is hot discussion about decentralized management of energy systems. These systems aim to facilitate the generators to engage in demand response programs (DR) that allow a more efficient management of electricity resources. Moreover the automation that is offered from DR approaches will allow distributed system operators (DSOs) to handle in a cost-effective way the electricity sale without compromising the reliability and security of the network.

References

[1] Ivan Gudymenko, Asadullah Khalid, Hira Siddiqui, Mujtaba Idrees, Sebastian Claub, Andre Luckow, Manuel Bolsinger, and Daniel Miehle. Privacy-Preserving Blockchain-Based Systems for Car Sharing Leveraging Zero-Knowledge Protocols. In *Proceedings - 2020 IEEE International Conference on Decentralized Applications and Infrastructures, DAPPS 2020*, pages 114–119. Institute of Electrical and Electronics Engineers Inc., 8 2020.

[2] Mingchong Li, Xiaolei Dong, Zhenfu Cao, and Jiachen Shen. PPHR: Blockchain-based Privacy Protection House Rental System. pages 145–149. Institute of Electrical and Electronics Engineers (IEEE), 10 2021.

[3] Roozbeh Sarenche, Mahmoud Salmasizadeh, Mohammad Hassan Ameri, and Mohammad Reza Aref. A secure and privacy-preserving protocol for holding double auctions in smart grid. *Information Sciences*, 557:108–129, 5 2021.

[4] Gaurav Sharma, Denis Verstraeten, Vishal Saraswat, Jean Michel Dricot, and Olivier Markowitch. Anonymous Fair Auction on Blockchain. *2021 11th IFIP International Conference on New Technologies, Mobility and Security, NTMS 2021*, 4 2021.

[5] Gaurav Sharma, Denis Verstraeten, Vishal Saraswat, Jean Michel Dricot, and Olivier Markowitch. Anonymous Sealed-Bid Auction on Ethereum. *Electronics 2021, Vol. 10, Page 2340*, 10(19):2340, 9 2021.