Hindawi
Security and Communication Networks
Volume 2021, Article ID 1615117, 10 pages
https://doi.org/10.1155/2021/1615117

*Research Article*

# Publicly Verifiable $M+1$st-Price Auction Fit for IoT with Minimum Storage

**Po-Chu Hsu** [1] **and Atsuko Miyaji** [1,2]

[1]*Graduate School of Engineering, Osaka University, Suita, Japan*
[2]*Japan Advanced Institute of Science and Technology, Nomi, Japan*

Correspondence should be addressed to Atsuko Miyaji; miyaji@comm.eng.osaka-u.ac.jp

Received 20 May 2021; Revised 21 July 2021; Accepted 28 September 2021; Published 30 November 2021

Academic Editor: David Megias



In an $M+1$st-price auction, all bidders submit their bids simultaneously, and the $M$ highest bidders purchase $M$ identical goods at the $M+1$st bidding price. Previous research is constructed based on trusted managers such as a trusted third party (TTP), trusted mix servers, and honest managers. All of the previous auctions are not fit for edge-assisted IoT since they need TTP. In this paper, we formalize a notion of commutative bi-homomorphic multiparty encryption and achieve no-TTP $M+1$-st auction based on blockchain with public verifiability. Our $M+1$st auction guarantees financial fairness, robustness, and correctness without TTP and is secure under a malicious model for the first time. Our $M+1$st auction can be executed over a distributed network and is thus fit for edge-assisted IoT. Furthermore, our formalized commutative bi-homomorphic multiparty encryption can be used in various applications for edge-assisted IoT, which needs to protect privacy and correctness.

## 1. Introduction

Since the introduction of Ethereum [1], a smart contract has become a robust decentralized execution environment for many applications. A variety of applications have been built on Ethereum. For example, cryptocurrency wallets, decentralized markets, and games are just a few examples. In this paper, we study secure auctions as an example of smart contracts.

Vickrey auction is also called second-price auction. In a second-price auction, the bidder who placed the highest bid only needs to pay the second-highest bidding price for the goods. Except for the identity of the highest bidder and the second-highest price, all other information is kept a secret. A 2nd-price auction is a type of sealed-bid auction. In a sealed-bid auction, bidders submit written bids without knowing other bidders' bids.

$M+1$st-price auction is an extension of second-price auction. In $M+1$st-price auction, there are $M$ identical goods. The bidders who placed top $M$ bids only need to pay the $M+1$st bidding price for the goods. Ideally, all information except the $M+1$st price and the bidders who placed the $M$ highest bids are kept as a secret. Figure 1 shows an example of $M+1$st auction for $M=2$ goods. The top two bidders $B_4$ and $B_2$ can buy the goods by \$500. Except the boxed information, all other information is kept as a secret. Table 1 shows the public and secret information in a $M+1$st-price auction.

There are many applications using IoT devices such as smart agriculture [2], smart grid [3], and raw materials [4], all of which will use price decision procedures. In the price decision procedures, the second-price auction is said to be a method that reflects market prices and is used in various situations. The $M+1$st auction is the second-price auction that deals with $M$ products. This is why $M+1$st auctions are exactly one of the important applications since they can use $M$ products at once.

To achieve a secure $M+1$st auction, there are two main issues: correctness and public verifiability. Usually, correctness and public verifiability are realized by using trusted third party (TTP) as an auctioneer and mix and match [5]. Abe and Suzuki [6] introduced one of the first auction protocols based

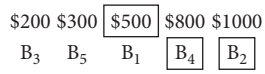


Figure 1: All but boxed information should be secret (example of $M$ + 1st auction for $M = 2$).

Table 1: Public and secret information in $M$ + 1st-price auction.

| | Other bidders | Top $M$ + 1st bidder | Top $M$ winning bidders |
|---|---|---|---|
| Identity | *Secret* | *Secret* | Public |
| Price | *Secret* | Public | *Secret* |

on homomorphic encryption. The bidding price is a bounded range. Bidders can only choose one bid from this pricing list. Many related research studies [7, 8] have a similar design such that the time or communication complexity is related to the length of the bidding price list $P$. To solve this problem, Mitsunaga et al. [9] proposed a scheme that uses a binary format to represent bidding vectors. This improvement reduced the time complexity to log $P$. In this research, instead of further improving complexity, we focus on removing the trusted parties, such as manager and mix servers. Based on Abe and Suzuki's scheme [6], we removed TTP by letting bidders collaboratively act as managers and mix servers. This strategy can also be used in follow-up research studies [7–9]. In our preliminary version of [10], we construct a verifiable $M$ + 1st-price auction without manager. In the journal version, we implement the smart contract and reduce the computation cost by using elliptic curve cryptography.

Auctioneer in a sealed-bid auction is usually a trusted person since he can see the secret information. To construct a scheme that no party, such as manager, bidder, smart contract, and trusted third party (TTP), knows secret information, a typical method is to divide a function of manager into two independent parties as in [7]. In their scheme, no single managers or bidders know this secret information. However, this scheme is based on the $p$th root problem, which causes high time complexity and requires massive memory usage by public parameters and bid values.

Mix and match [5] is widely used in auction protocols such as bit-slice-based auction [9, 11, 12] as well as [6] to verify if each encrypted bit is either $E(0)$ or $E(1)$. Mix and match can be divided into mix phase and match phase. The mix phase is based on mix net [13, 14] to shuffle a set of encrypted messages. The match phase is usually based on zero-knowledge proofs to prove that the encrypted message is in the shuffled set. Mix net not only relied on TTP but also increased communication costs. In summary, previous schemes need TTP.

*1.1. Our Contribution.* We proposed a secure $M$ + 1st auction, which satisfies the following features:

(1) *Multiparty encryption:* our multiparty encryption is asynchronous.

The order of decryption does not depend on the order of encryption, which makes the decryption asynchronous for the decryptors. So, decryptors do not need to wait for each other.

(2) *Publicly verifiable $M$ + 1st auction:* our $M$ + 1st auction does not use mix and match in bid verification phase and does not use TTP in the whole protocol.

(a) *No mix and match protocol in bid verification:* our protocol does not rely on mix and match unlike [6, 9, 11, 12]. Our new approach can also be used on bit-slice-based auction protocols [9, 11, 12] and all other auction protocols, which is based on encrypted bits.

(b) *No TTP*: in the $M$ + 1st-price decision phase, we modify mix and match by letting bidders act as mix servers, and thus no trusted mix server is used. Our $M$ + 1st-price auction is secure under a malicious model that guarantees financial fairness, robustness, and correctness without any other party such as manager and TTP. We remark that our scheme can be operated without TTP, and thus any bidder can start auction freely by using their IoT devices. Our decentralized auction exactly fits for IoT environment.

This paper is organized as follows. We introduce the related studies on smart contract auction protocols in Section 2 and explain cryptographic backgrounds in Section 3. We propose an efficient and secure $M$ + 1st-price auction protocol in Section 4 and compare our work with the related works in Section 6. Finally, we conclude our work in Section 7.

## 2. Related Work

This section reviews previous auction protocols which use a smart contract.

*2.1. Overview of Auction.* We define necessary features for auction protocol

(i) Bid binding: bidders cannot change their bidding price after the bid submission phase is closed.

(ii) Bid secrecy: all bidding prices except the $M$ + 1st-price should be kept secret.

(iii) Bidder anonymity: except for the winning bidder, the identity of other bidders, including the bidder who placed the $M$ + 1st-price, is kept secret.

(iv) Manager-based secrecy and anonymity: except those explained above, all secrets are also disclosed to the manager.

(v) Posterior secrecy and anonymity: all secrets can be a secret even after the auction.

(vi) Robustness: malicious behaviors in each phase can be found immediately. Timeouts are set to make sure the auction ends within a determined time.

(vii) Public verifiability: all procedures can be publicly verified.

(viii) Correctness: the auction has correctly proceeded as defined in the auction protocol.

(ix) Financial fairness: malicious parties need to compensate other parties. Usually, all parties need to deposit some Ether in the smart contract as a stake.

*2.2. Hawk.* Hawk executes smart contracts while keeping secrets [15] when a bidder and a manager exchange data in the following way. (1) When the protocol starts, the bidders and the manager deposit some Ether to the smart contract as a stake. (2) Zero-knowledge proof (zk-SNARK) is attached with all messages sent by the manager and bidders. (3) Smart contract verifies the zero-knowledge proof and the time limit in each phase. In the Hawk protocol, since the bid amount of all bidders is leaked to the manager by design, it is necessary to assume that the manager does not leak any information inappropriately or collude with any party. In other words, the manager has to be a trusted third party. The Hawk protocol achieves bid binding, bid secrecy, bidder anonymity, manager-based secrecy and anonymity, posterior secrecy and anonymity, robustness, correctness, public verifiability, and financial fairness.

*2.3. Verifiable Sealed-Bid Auction Protocol.* Another verifiable sealed-bid auction protocol [8] was proposed by using a smart contract and Pedersen commitment scheme [16]. In this protocol, there are bidders and managers, just as in Hawk. The manager and a bidder pay a deposit to the smart contract, and the bidder simultaneously sends a bid commitment to the smart contract. Next, the bidder encrypts the value by the manager's public key and opens the commitment to the smart contract. The manager determines the winning bid and winner, sends the commitment of the winning bid to the smart contract, and proves to the smart contract that the winning bid is higher than any other bid. This scheme achieves bid binding, bid secrecy, manager-based secrecy, posterior secrecy, robustness, correctness, and financial fairness.

*2.4. Other Related Works.* Wu et al. proposed a smart contract-enabled collusion-resistant e-auction [17], which is well organized and provides many experimental data. However, bidder anonymity is revealed when commitments have been opened. Blockchain-based smart contract for bidding system [18] has the same problem as [17] although they used a blockchain called MinerGate. It also reveals secrets at the end of the auction. Galal also proposed a full privacy-preserving Vickrey auction on top of Ethereum [19], whose design is similar to Hawk. All bidders encrypt their bids by managers of public key. The manager uses Intel SGX to decrypt each bids and determine a winner. Thus, it only satisfies manager-based secrecy and anonymity.

## 3. Preliminaries

*Definition 1* (DDH assumption). *Let $t$ be a security parameter. A decisional Diffie–Hellman (DDH) parameter generator $\mathcal{IG}$ is a probabilistic polynomial time (PPT) algorithm that takes an input $1^k$ and outputs the description of a finite field $\mathbb{F}_p$ and a basepoint $g \in \mathbb{F}_p$ with the prime order $q$. We say that $\mathcal{IG}$ satisfies the DDH assumption if $\epsilon = |p_1 - p_2|$ is negligible (in K) for all PPT algorithms A, where $p_1 = PR[(\mathbb{F}_p, g) \longleftarrow \mathcal{IG}(1^K); y_1 = g^{x_1}, y_2 = g^{x_2} \longleftarrow \mathbb{F}_p: A(\mathbb{F}_p, g, y_1, y_2, g^{x_1 x_2}) = 0]$ and $p_2 = \Pr[(\mathbb{F}_p, g) \longleftarrow \mathcal{IG}(1^K); y_1 = g^{x_1}, y_2 = g^{x_2}, z \longleftarrow \mathbb{F}_p: A(\mathbb{F}_p, g, y_1, y_2, z) = 0]$.*

*Definition 2* (ElGamal encryption [20]). Let $p$ and $q$ be large primes. Let $\langle g \rangle$ denote a prime subgroup of $\mathbb{Z}_p^*$ generated by $g$ whose order is $q$. Given a message $m \in \mathbb{Z}_p^*$, we define ElGamal [20] encryption as $Enc_y(m) = (g^r, m \cdot y^r)$, where $y$ is the public key and $r \in \mathbb{Z}_q^*$. Given a ciphertext $c = (g^r, m \cdot y^r)$, decryption is defined as $Dec_x(c) = m$, where $x$ is the private key.

**Theorem 1** (plaintext equivalence proof of ElGamal ciphertext). *Given an ElGamal ciphertext $(g^r, my^r)$ and a plaintext $m'$, the encryptor can prove the value of $m = m'$ without revealing $r$. This type of proof is based on the proof of equality of two discrete logarithms. i.e., $g^r$ and $my^r/m'$ share the same discrete logarithm $r$.*

**Theorem 2** (proof of knowledge of a discrete logarithm). *Given $g \in \mathbb{Z}_p$ and $x \in \mathbb{Z}_q^*$, the knowledge of $g^x$'s discrete logarithm $x$ can be proved without revealing $x$.*

**Theorem 3** (proof of equality of two discrete logarithms). *Given $g_1, g_2 \in \mathbb{Z}_p$ and $x_1, x_2 \in \mathbb{Z}_q^*$, the proof of $g_1^{x_1}$ and $g_2^{x_2}$ has same discrete logarithm $x_1 = x_2$ that can be proved without revealing $x_1$ and $x_2$. This type of proof is a variation of the proof of knowledge of a discrete logarithm (Theorem 2).*

*Protocol 1* (mix and match). Mix and match [5] is a technique used to examine whether the decryption of a ciphertext $Dec(c)$ belongs to a set of plaintexts. It is based on mix net, which can perform verifiable secret shuffle. The mix server only knows their own permutation, and they do not know the plaintext. $MixMatch[c, S]$:

Input: ciphertext $c$ and a set of plaintexts $S = \{m_1, m_2, \ldots, m_n\}$.

Output: true if $Dec(c) \in S$. Otherwise, output false.

(1) Construct $n$ ciphertext by $C = \{c/E(m_1), \ldots, c/E(m_n)\} = \{C_1, \ldots, C_n\}$.

(2) For each mix server

  (a) Calculate $C' = \{C_1^{r_1}, \ldots, C_n^{r_n}\}$ where $r_i$ is a secret random value.
  (b) Generate a random permutation $pm$.
  (c) Shuffle these ciphertexts: $Shuffle[C', pm] = (C'', \pi)$.
  (d) Send $C''$ to next mix server and publish $\pi$.

(3) Use a publicly verifiable way to decrypt the output $C''$ of the last mix server.

(4) If there exists one plaintext 1, then output true. Otherwise, output false.

| $price = [$ | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | $]$ |
|---|---|---|---|---|---|---|---|---|---|
| $V = [$ | E (0) | E (0) | E (0) | E (0) | E (0) | E (0) | E (0) | E (0) | $]$ |

Figure 2: Example of bit-slice bidding vector.

| $price = [$ | $2^0$ | $2^1$ | $2^2$ | $2^3$ | $]$ |
|---|---|---|---|---|---|
| $V = [$ | E (0) | E (1) | E (1) | E (0) | $]$ |

Figure 3: Example of bit-slice bidding vector in binary format.

*Protocol 2* (verifiable secret shuffle). Verifiable secret shuffle is a technique for mix and match, which is designed to use multiple mix servers (TTP) to secretly shuffle ciphertexts. $Shuffle[C, pm] = (C', \pi)$:

Input: $n$ ciphertexts $C = \{Enc(m_1), \ldots, Enc(m_n)\}$.

Output: shuffled ciphertexts $C' = \{Enc(m_{pm(1)}), \ldots, Enc(m_{pm(n)})\}$ and the proof $\pi$, where $pm(\cdot)$ is a secret permutation.

*Protocol 3* (bit-slice). Bit-slice is a common technique used in many auction protocols. Assume that the maximum bidding price is 8 dollars. If bidder wants to buy the goods by 6 dollars, the bidder can compose a bidding vector $V$ by $E(0)$ and $E(1)$. Figure 2 gives an example of a bit-slice bidding vector. The bidding vector can also be presented by a binary format [9, 12] to reduce its size. Figure 3 gives an example of a binary format bidding vector.

*Algorithm 1* (binary search). Binary search [21] is an algorithm used to search an ordered list; it can reduce the time complexity to $O(\log(N))$, where $N$ is the number of elements; $BiSearch[A, cmp] = i$:

Input: given an ordered list $A$ of $N$ elements and a compare function $cmp$.

Output: the index $j$ of the target element $A[j]$. If there is no element that satisfies the condition, output the largest $i$ where $cmp(i) = -1$.

## 4. Our Scheme

Mix servers in mix and match perform verifiable secret shuffle. However, the mix servers can leak the secret permutation to the bidder or the manager. In an auction, let bidder act as mix server that can still be secure because each bidder can only know their permutation. As bidders are counterparties, they have no incentives to leak their secret permutation to others. The design of this protocol is simple, and no additional parties such as manager and TTP are needed.

A notion of commutative encryption is extended to multiparty encryption, which is fit for encryption schemes in smart contract protocol. During decryption, each secret key holder can send their decryption message to the smart contract concurrently. Without this, if the ciphertext is not decrypted before timeout, the responsibility of failure is hard to define. The last honest decryptor might be punished simply because the previous decryptor takes too much time. Also, transaction settlement takes time. It can be time costly for decryptors to decrypt things one by one.

*4.1. Multiparty Encryption.* A concept of commutative encryption was introduced in [22]. However, to better fit the asynchronous nature of smart contract protocol, we reformalize it as multiparty encryption here. Without losing generality, we use $n = 2$ case to explain.

*Definition 3* (multiparty encryption). Let the public and private key pairs of the public key cryptosystem of the party be $(y_1, x_1)$ and $(y_2, x_2)$, respectively. The ciphertext of plaintext using the public key $y_i$ is $Enc_{y_i}(m)$, and the decryption is $Dec_{y_i}(Enc_{y_i}(m))$. When public key cryptosystem satisfies the following, the public key cryptosystem is called a multiparty encryption cryptosystem.

(i) *Full Encryption. We* define *the encryption using the public key $y_2$ for* the *ciphertext encrypted with public key $y_1$ for a plaintext $m$ through the following operation.*

$$Enc_{y_1 \cdot y_2}(m) = Enc_{y_2}\left(Enc_{y_1}(m)\right). \tag{1}$$

(ii) *Partial Decryption. We define the decryption performed by each party for the ciphertext $Enc_{y_1 \cdot y_2}(m)$ encrypted with the public keys $y_1$, $y_2$ for the plaintext $m$ through the following operation.*

$$\begin{aligned} Dec_{y_1}\left(Enc_{y_1 \cdot y_2}(m)\right) &= Enc_{y_2}(m), \\ Dec_{y_2}\left(Enc_{y_1 \cdot y_2}(m)\right) &= Enc_{y_1}(m). \end{aligned} \tag{2}$$

(iii) *Full Decryption. We define the decryption performed by both parties for the ciphertext $Enc_{y_1 \cdot y_2}(m)$encrypted with the public keys$y_1$, $y_2$ for the plaintext $m$ through the following operation.*

$$Dec_{y_1}\left(Dec_{y_2}\left(Enc_{y_1 \cdot y_2}(m)\right)\right) = m. \tag{3}$$

*Definition 4* (commutative). For a given encryption $Enc_{y_1 \cdot y_2}(m)$, if

$$\begin{aligned} &Dec_{y_1}\left(Dec_{y_2}\left(Enc_{y_1 \cdot y_2}(m)\right)\right) \\ &\quad = Dec_{y_2}\left(Dec_{y_1}\left(Enc_{y_1 \cdot y_2}(m)\right)\right) = m, \end{aligned} \tag{4}$$

is satisfied, then the encryption is a commutative.

*Definition 5* (bi-homomorphic). When a encryption satisfies the following properties, we say that the encryption is bi-homomorphic.

(i) $Enc_{y_1 \cdot y_2}(m) \cdot Enc_{y_1}(\tilde{m}) = Enc_{y_1 \cdot y_2}(m \cdot \tilde{m})$

(ii) $Enc_{y_1 \cdot y_2}(m) \cdot Enc_{y_2}(\tilde{m}) = Enc_{y_1 \cdot y_2}(m \cdot \tilde{m})$

(iii) $Enc_{y_1 \cdot y_2}(m) \cdot Enc_{y_1 \cdot y_2}(\tilde{m}) = Enc_{y_1 \cdot y_2}(m \cdot \tilde{m})$

(iv) $(Enc_{y_1 \cdot y_2}(m))^{\omega} = Enc_{y_1 \cdot y_2}(m^{\omega})$

The multiparty encryption based on the ElGamal encryption is defined as follows, which will be proved to be bihomomorphism.

(i) *Full Encryption*. The encryption using the public keys $y_2$ and $y_1$ is as follows.

$$Enc_{y_1 \cdot y_2}(m) = (u_1, u_2, c), where, \quad u_1 = g^{r_1}, u_2 = g^{r_2} \text{ and } c = m \cdot y_1^{r_1} y_2^{r_2}. \tag{5}$$

(ii) *Partial Decryption*. The partial decryption for a ciphertext $Enc_{y_1 \cdot y_2}(m)$ by each party $y_1$ or $y_2$ is as follows.

$$\begin{aligned} Dec_{y_1}\left(Enc_{y_1 \cdot y_2}(m)\right) &= \left(u_2, c \cdot u_1^{-x_1}\right) = \left(u_2, m y_2^{r_2}\right) \\ &= Enc_{y_2}(m), \\ Dec_{y_2}\left(Enc_{y_1 \cdot y_2}(m)\right) &= \left(u_1, c \cdot u_2^{-x_2}\right) = \left(u_1, m y_1^{r_1}\right) \\ &= Enc_{y_1}(m). \end{aligned} \tag{6}$$

(iii) *Full Decryption*. The decryption performed by both parties for the ElGamal ciphertext $Enc_{y_1 \cdot y_2}(m)$ encrypted with the public keys $y_1$ and $y_2$ for the plaintext $m$ is defined as follows.

$$Dec_{y_1}\left(Dec_{y_2}\left(Enc_{y_1 \cdot y_2}(m)\right)\right) = Dec_{y_1}\left(Enc_{y_1}(m)\right) = m. \tag{7}$$

Thus, the multiparty ElGamal encryption scheme satisfies partial decryption, full decryption, and commutative encryption. We also show that the multiparty ElGamal encryption satisfies the commutative encryption with bihomomorphism as follows.

**Theorem 4.** *The multiparty ElGamal encryption satisfies the commutative encryption with bi-homomorphism.*

*Proof.* For two given ciphertexts $Enc_{y_1 \cdot y_2}(m) = (u_1, u_2, c)$ and $Enc_{y_1}(\tilde{m}) = (\tilde{u_1}, \tilde{c})$ with $u_1 = g^{r_1}, u_2 = g^{r_2}, \tilde{u_1} = g^{\tilde{r}_1}, c = m \cdot y_1^{r_1} y_2^{r_2}, \tilde{c} = \tilde{m} \cdot y_1^{\tilde{r}_1}$, the multiplication of the ciphertexts is as follows.

$$\begin{aligned} Enc_{y_1 \cdot y_2}(m) \cdot Enc_{y_1}(\tilde{m}) &= (u_1, u_2, c) \cdot (\tilde{u_1}, \tilde{c}) \\ &= \left(u_1 \tilde{u_1}, u_2, c\tilde{c}\right) = Enc_{y_1 \cdot y_2}(m \cdot \tilde{m}). \end{aligned} \tag{8}$$

For two given ciphertexts $Enc_{y_1 \cdot y_2}(m) = (u_1, u_2, c)$ and $Enc_{y_2}(\tilde{m}) = (\tilde{u_2}, \tilde{c})$with $u_1 = g^{r_1}, u_2 = g^{r_2}, \tilde{u_2} = g^{\tilde{r}_2}, c = m \cdot y_1^{r_1} y_2^{r_2}, \tilde{c} = \tilde{m} \cdot y_2^{\tilde{r}_2}$, the multiplication of the ciphertexts is as follows.

$$\begin{aligned} Enc_{y_1 \cdot y_2}(m) \cdot Enc_{y_2}(\tilde{m}) &= (u_1, u_2, c) \cdot (\tilde{u_2}, \tilde{c}) \\ &= \left(u_1, u_2 \tilde{u_2}, c\tilde{c}\right) = Enc_{y_1 \cdot y_2}(m \cdot \tilde{m}). \end{aligned} \tag{9}$$

For two given ciphertexts $Enc_{y_1 \cdot y_2}(m) = (u_1, u_2, c)$ and $Enc_{y_1 \cdot y_2}(\tilde{m}) = (\tilde{u_1}, \tilde{u_2}, \tilde{c})$ with $u_1 = g^{r_1}, u_2 = g^{r_2}, c = m \cdot y_1^{r_1} y_2^{r_2}, \tilde{u_1} = g^{\tilde{r}_1}, \tilde{u_2} = g^{\tilde{r}_2}, \tilde{c} = \tilde{m} \cdot y_1^{\tilde{r}_1} y_2^{\tilde{r}_2}$, the multiplication of the ciphertexts is as follows.

$$\begin{aligned} Enc_{y_1 \cdot y_2}(m) \cdot Enc_{y_1 \cdot y_2}(\tilde{m}) &= (u_1, u_2, c) \cdot (\tilde{u_1}, \tilde{u_2}, \tilde{c}) \\ &= \left(u_1 \tilde{u_1}, u_2 \tilde{u_2}, c\tilde{c}\right) \\ &= Enc_{y_1 \cdot y_2}(m \cdot \tilde{m}). \end{aligned} \tag{10}$$

For a given ciphertext $Enc_{y_1 \cdot y_2}(m) = (u_1, u_2, c)$ with $u_1 = g^{r_1}, u_2 = g^{r_2}, c = m \cdot y_1^{r_1} y_2^{r_2}$, the exponentiation of the ciphertext is as follows.

$$\begin{aligned} \left(Enc_{y_1 \cdot y_2}(m)\right)^{\omega} &= \left(u_1^{\omega}, u_2^{\omega}, c^{\omega}\right) \\ &= \left(g^{r_1 \omega}, g^{r_2 \omega}, m^{\omega} \cdot y_1^{r_1 \omega} y_2^{r_2 \omega}\right) \\ &= Enc_{y_1 \cdot y_2}\left(m^{\omega}\right). \end{aligned} \tag{11}$$

By using Theorem 1, we can make the verifiable partial decryption as follows. □

**Theorem 5** (verifiable partial decryption). *A decryptor (party 2) can prove that a decrypted ciphertext $Enc_{y_1} = (u_1, c_1)$ is a partial decryption of $Enc_{y_1 \cdot y_2}(m) = (u_1, u_2, c)$ without revealing $x_2$ by showing a same discrete logarithm proof that $y_2 = g^{x_2}$ and $u_2^{-x_2} = (u_2^{-1})^{x_2}$ has same discrete log $x_2$.*

$$SPK\left[(\alpha): c/c_1 \equiv y_2^{\alpha} \wedge u_2 \equiv g^{\alpha}\right](m). \tag{12}$$

*4.2. Our Auction Protocol.* Our protocol consists of a seller, who sells $M$ goods at a fixed set of prices $\{1, \ldots, P\}$, and bidders, who bid a price for the goods. The top $M$ bidders can buy the goods by the $M + 1$st price. Remark that no other party except the seller of bidders is required in our protocol.

Our protocol consists of 7 phases as follows.

(1) *Smart contract deployment:* the seller sets necessary parameters and initialization.

(2) *Bidder initialization:* bidders who are interested in the auction submit their stake and join the auction.

(3) *Submission of bids by bidders:* bidders decide their bid and submit it to the smart contract.

(4) *Bid verification:* verify all bidders' bid.

(5) $M + 1$*st-price decision*: find out the $M + 1$st bidding price.

(6) *Winner decision:* find out the highest $M$ bidders who are winners of the auction.

(7) *Payment:* the winning bidders pay the seller the $M + 1$st price to buy the goods.

The protocol is described in detail as follows.

*4.2.1. Smart Contract Deployment.* The seller decides the following parameter and deploys the smart contract (SC) to start the auction.

(i) Cryptographic parameters: large prime $p$, a base-point $g$ with prime order $q$, and an auction base $z \longleftarrow \mathbb{Z}_p/\{0,1\}$.

(ii) Seller initialization:

(1) Bidding price list $\{p_1, \ldots, p_P\}$.

(2) Timeouts for each phase $T_1, \ldots, T_6$: if a bidder failed to submit valid messages to smart contract within the timeout, the bidder will be treated as a malicious bidder and be financially penalized.

(3) Stake requirement $d$: bidders are required to submit $d$ amount of Ether as stake to join the auction.

*4.2.2. Phase 1: Bidder Initialization.* Bidders submit $(y_i, \pi_i^1)$ to SC within time $T_1$ as follows. We assume that there are more than $M+1$ bidders.

(i) Each bidder $B_i, i \in \{1, \ldots, B\}$ computes the following and submits it to SC:

(1) Compute a public key $y_i = g^{x_i}$ for the ElGamal encryption, where $x_i \longleftarrow \mathbb{Z}_q$ is a randomly chosen secret key.

(2) Compute a proof $\pi_i^1$ of $y_i = g^{x_i}$ as a conventional way of a proof of knowledge of $y_i$'s discrete logarithm $x_i$ described in Section 3 and $d$ amount of Ether as stake.

(ii) After this phase ends, an aggregated public key $Y = \prod_{i=1}^{B} y_i$ can be calculated by SC.

*4.2.3. Phase 2: Submission of Bids by Bidders.* All bidders submit $(V_i, V_i', \pi_i^2)$ to SC within time $T_2$ as follows: see Figure 4 for $V_i$ and $V_i'$. Bidders $B_2$ and $B_3$ win the auction and pay $p_{j_{M+1}} = p_2$ for the goods.

(i) $B_i, i \in \{1, \ldots, B\}$, computes the below and submits it to SC:

(1) Bidding vector $V_i = \{V_{i1}, \ldots, V_{iP}\}$.

(2) Shuffled bidding vector $V_i' = \{V_{i,pm(1)}, \ldots, V_{i,pm(P)}\}$.

(3) A proof of secure shuffle $\pi_i^2$.

$$V_{ij} = \begin{cases} \mathrm{Z}^1, & \text{if } j = b_i, \\ \mathrm{Z}^0, & \text{if } j \neq b_i. \end{cases} \tag{13}$$

A bidder $B_i$ first chooses a bidding point $b_i \in \{1, \ldots, P\}$ corresponding to bidding price $p_{b_i}$, constructs a bidding vector $V_i$, and shuffles $V_i$ to $V_i'$, while making a proof of the secure shuffle:

$$\left(\pi_i^2, V_i'\right) = shuffle\left(V_i, pm_i\right), \tag{14}$$

where $pm_i$ is a secret permutation generated by $B_i$. This $V_i'$ is used for the verification of $V_i$. In phase 3, we can decrypt $V_i'$ to verify $V_i$ without leaking $V_i$.

(i) SC calculates the following array $\{a_i\}$ for $B_i$ and $c$ for bidding price after receiving all $(V_i, pm_i)$ (see Figure 4 for $\{a_i\}$ and $c$).

$$\begin{array}{lll} & & p_1 \quad p_2 \quad p_3 \quad p_4 \quad p_5 \\ b_1 = 2, & V_1 & = [\ \mathrm{Enc}_\gamma(\mathrm{z}^0), \mathrm{Enc}_\gamma(\mathrm{z}^1), \mathrm{Enc}_\gamma(\mathrm{z}^0), \mathrm{Enc}_\gamma(\mathrm{z}^0), \mathrm{Enc}_\gamma(\mathrm{z}^0)\ ], \\ b_2 = 4, & V_2 & = [\ \mathrm{Enc}_\gamma(\mathrm{z}^0), \mathrm{Enc}_\gamma(\mathrm{z}^0), \mathrm{Enc}_\gamma(\mathrm{z}^0), \mathrm{Enc}_\gamma(\mathrm{z}^1), \mathrm{Enc}_\gamma(\mathrm{z}^0)\ ], \\ b_3 = 3, & V_3 & = [\ \mathrm{Enc}_\gamma(\mathrm{z}^0), \mathrm{Enc}_\gamma(\mathrm{z}^0), \mathrm{Enc}_\gamma(\mathrm{z}^1), \mathrm{Enc}_\gamma(\mathrm{z}^0), \mathrm{Enc}_\gamma(\mathrm{z}^0)\ ], \\ & & \\ \text{(shuffled)} & V_1' & = [\ \mathrm{Enc}_\gamma(\mathrm{z}^0), \mathrm{Enc}_\gamma(\mathrm{z}^0), \mathrm{Enc}_\gamma(\mathrm{z}^0), \mathrm{Enc}_\gamma(\mathrm{z}^1), \mathrm{Enc}_\gamma(\mathrm{z}^0)\ ], \\ \text{(shuffled)} & V_2' & = [\ \mathrm{Enc}_\gamma(\mathrm{z}^0), \mathrm{Enc}_\gamma(\mathrm{z}^0), \mathrm{Enc}_\gamma(\mathrm{z}^0), \mathrm{Enc}_\gamma(\mathrm{z}^1), \mathrm{Enc}_\gamma(\mathrm{z}^0)\ ], \\ \text{(shuffled)} & V_3' & = [\ \mathrm{Enc}_\gamma(\mathrm{z}^0), \mathrm{Enc}_\gamma(\mathrm{z}^0), \mathrm{Enc}_\gamma(\mathrm{z}^0), \mathrm{Enc}_\gamma(\mathrm{z}^0), \mathrm{Enc}_\gamma(\mathrm{z}^1)\ ], \\ & & \\ & a_1 & = [\ \mathrm{Enc}_\gamma(\mathrm{z}^1), \mathrm{Enc}_\gamma(\mathrm{z}^1), \boxed{\mathrm{Enc}_\gamma(\mathrm{z}^0)}, \mathrm{Enc}_\gamma(\mathrm{z}^0), \mathrm{Enc}_\gamma(\mathrm{z}^0)\ ] \\ & a_2 & = [\ \mathrm{Enc}_\gamma(\mathrm{z}^1), \mathrm{Enc}_\gamma(\mathrm{z}^1), \boxed{\mathrm{Enc}_\gamma(\mathrm{z}^1)}, \mathrm{Enc}_\gamma(\mathrm{z}^1), \mathrm{Enc}_\gamma(\mathrm{z}^0)\ ] \text{ win} \\ & a_3 & = [\ \mathrm{Enc}_\gamma(\mathrm{z}^1), \mathrm{Enc}_\gamma(\mathrm{z}^1), \boxed{\mathrm{Enc}_\gamma(\mathrm{z}^1)}, \mathrm{Enc}_\gamma(\mathrm{z}^0), \mathrm{Enc}_\gamma(\mathrm{z}^0)\ ] \text{ win} \\ & & \\ & c & = [\ \mathrm{Enc}_\gamma(\mathrm{z}^3), \mathrm{Enc}_\gamma(\mathrm{z}^3), \mathrm{Enc}_\gamma(\mathrm{z}^2), \mathrm{Enc}_\gamma(\mathrm{z}^1), \mathrm{Enc}_\gamma(\mathrm{z}^0)\ ] \\ & \text{MixMatch} & = [\ \underset{j_{M+1}}{\mathit{False}}, \quad \mathit{False}, \quad \mathit{True}, \quad \mathit{True}, \quad \mathit{True}\ ] \end{array}$$

Figure 4: Example of $M = 2$ with 3 bidders and 5 bidding prices.

(1) Compute array $a_i = (a_{i1}, \ldots, a_{iP})$ for each bidder $B_i$:

$$a_{ij} = \prod_{k=j}^{P} V_{ik} = \mathrm{Z} \sum_{k=j}^{P} t_{ik} \quad (1 \leq j \leq p). \tag{15}$$

For any $j$, a value of $a_{ij}$ is equal to $\mathrm{Z}^1$ if $b_i \geq j$, where $b_i$ is $B_i$'s bidding point. Otherwise, $a_{ij} = \mathrm{Z}^0$. The time complexity is $O(P)$, by computing $a_{ij} = a_{i(j+1)} \cdot V_{ij}$.

(2) Compute array $c = (c_1, \ldots, c_P)$ for each bidding price $p_1, \ldots, p_P$:

$$c_j = \prod_{i=1}^{B} a_{ij} = \mathrm{Z} \sum_{i=1}^{B} \sum_{j=1}^{P} t_{ij} \quad (1 \leq j \leq p). \tag{16}$$

The value of $c_j$ is the number of bidders whose bid $b_i$ is larger than or equal to $j$.

*4.2.4. Phase 3: Bid Verification.* A bidder can only bid on one price. So, correct bidding vectors $V_i$ consists of $P-1$ times of $\mathrm{Z}^0$ and one $\mathrm{Z}^1$. Since bidder $B_i$ also submitted $V_i'$, a shuffled $V_i$, we can simply decrypt $V_i'$ to verify if it consists of $P-1$ times of $\mathrm{Z}^0$ and one $\mathrm{Z}^1$ within time $T_3$. The decryption will not leak any secret since the shuffled bidding vector $V_i'$ does not contain any secret.

(i) SC verifies all bidder $B_i$'s bid $V_i$:

(1) Decrypt the shuffled bidding vector $V_i'$:

$$\{m_{i1}, \ldots, m_{iP}\} = \{Dec(V_{i1}'), \ldots, Dec(V_{iP}')\}. \tag{17}$$

(2) Verify if there is exactly $P-1$ times of $z^0$ and one $z^1$ in $\{m_{i1}, \ldots, m_{iP}\}$.

*4.2.5. Phase 4: $M+1$st-Price Decision.* By using mix and match, SC and all $B_i$ determine the $M+1$st price within $T_4$. In array $c = (c_1, \ldots, c_P)$, the value of $c_j$ is the number of bidders $B_i$ whose bid $b_i$ is larger than or equal to bidding point $j$. Thus, we can use mix and match to find out the bidding point $j$ where $Dec(c_j) \notin \{z^0, z^1, \ldots, z^M\}$, but $Dec(c_{j+1}) \in \{z^0, z^1, \ldots, z^M\}$. This $j$ is the $M+1$st price's

bidding point. Since the $c$ array is a decrementing array, binary search can speed up the search. We use $j_{M+1st}$ as a symbol to this $j$.

(i) By all bidders' help, SC finds the bidding point $j$ that $MixMatch[c_j, \{z^0, z^1, \ldots, z^M\}] = False$ but $MixMatch[c_{j+1}, \{z^0, z^1, \ldots, z^M\}] = True$. Assume $cmp$ is the compare function of binary search (Algorithm 1). $BiSearch$ will return the $j$ where $cmp(c_j) = -1$ but $cmp(c_{j+1}) = 1$. This $j$ is the $M+1$st price.

$$j_{M+1st} = BiSearch[c, cmp],$$
$$cmp(c_j) = \begin{cases} 1, & \text{if } MixMatch[c_j, \{z^0, z^1, \ldots, z^M\}] = True, \\ -1, & \text{otherwise.} \end{cases} \tag{18}$$

*4.2.6. Phase 5: Winner Decision.* In this phase, by decrypting all bidders' $a_{i,j_{M+1st}+1}$ within time $T_5$, the winners can be easily found. In this phase, bid secrecy holds because the bid $b_i$ can be any number between $j_{M+1st}+1$ and $P$.

(i) By all bidders' help, SC decrypts $a_{i,j_{M+1st}+1}$ for all $i \in \{1, \ldots, B\}$.

$$\delta_i = DEC(a_{i,j_{M+1st}+1}) = z^{\sum_{k=j}^{P} t_{ik}}. \tag{19}$$

Bidder $B_i$ with $\delta_i = z^1$ wins the auction.

*4.2.7. Phase 6: Payment.* The bidders who win the auction send $d_{p_{j_{M+1st}}}$ amount of Ether (in Wei) to the seller through SC.

(i) All winning bidders $B_i$ send $d_{p_{j_{M+1st}}}$ amount of Ether (in Wei) to SC.

(ii) SC sends $d_{p_{j_{M+1st}}} M$ amount of Ether (in Wei) to the seller and refunds the stakes to all bidders.

*4.3. Features and Security.* In this section, we discuss the features and security of our protocol.

(i) *Bid binding:* according to our implementation, the functions in SC will not allow bidders to change their bidding point after the bid submission phase is closed.

(ii) *Bid secrecy:* the bidding vectors $V_1, \ldots, V_B$ are encrypted by all bidders' public keys $Y = y_1 \cdots y_B$. Without all bidders' collaboration, the bid is kept a secret. Even though $V'$ is decrypted, it is a permutation of $V$. Thus, decrypting $V'$ will not leak any secret. The secrecy of top $M$ bids is protected in phase 5. If $a_{i,j_{M+1st}+1}$ is an encrypted $z^1$, all bids from $a_{i,j_{M+1st}+1}$ to $a_{iP}$ can contain $z^1$.

Unless all bidders collude, the bid secrecy will be satisfied in the mix and match used in Phase 4. In the mix part, ciphertexts are re-randomized and secretly shuffled by all bidders. Without all bidders colluding, the permutation of shuffled ciphertexts cannot be identified. In the match part, the shuffled ciphertexts can be decrypted by only all bidders' collaboration.

(iii) *Bidder anonymity:* since $c_1, \ldots, c_P$ are products of ciphertexts generated by all bidders, i.e., $c_j = \prod_{i=1}^{B} a_{ij}$. The proof of $Dec(c_j) = z^M$ will not leak $a_{ij}, i = 1, \ldots, B$. Thus, the identity of the $M+1$st bidder is still a secret.

(iv) *Posterior secrecy and anonymity:* the bidding points $b_i, i = 1, \ldots, B$ and bidding vectors $V_i, i = 1, \ldots, B$ are still secrets even after the auction. Thus, posterior secrecy and anonymity still hold.

(v) *Robustness:* since all messages sent to the smart contract are attached with a proof, malicious behaviors in each phase can be found immediately. The timeouts $T_1, \ldots, T_6$ are also set for each phase to ensure that the auction ends within a determined time. Therefore, this protocol provides robustness even in malicious models.

(vi) *Public verifiability:* all messages sent to SC are attached with a publicly verifiable non-interactive proof, which can be verified by smart contract immediately. Therefore, the correctness of the protocol is publicly verifiable.

(1) *Phase 1: Bidder Initialization.* A public key $y_i = g^{x_i}$ is submitted by each bidder $B_i$ to SC, $i = 1, \ldots, B$. The proof of knowledge of $x_i$ is publicly verifiable. Thus, the correctness of the public key is publicly verifiable.

(2) *Phase 2: Submission of Bids by Bidders.* A secure shuffled bidding vector $V_i'$ is submitted by each bidder $B_i$ to SC, $i = 1, \ldots, B$. The proof that $V_i'$ is a secure shuffle of $V_i$ is publicly verifiable.

(3) *Phase 3: Bid Verification.* Let $V_i' = (V_{i1}', \ldots, V_{iP}') = ((u_{i1}, v_{i1}), \ldots, (u_{iP}, v_{iP}))$. $BP$ decryption messages $u_{ij}^{x_i}; i = 1, \ldots, B; j = 1, \ldots, P$ are submitted by each bidder $B_i$. The proof of the same discrete logarithm that $u_{ij}^{x_i}$ and public key $y_i = g^{x_i}$ have the same discrete logarithm $x_i$ is publicly verifiable. If $V_i'$ is valid, then $V_i$ is a valid bidding vector. Thus, the verification on bidding vectors $V_i, i = 1, \ldots, B$ is publicly verifiable.

(4) *Phase 4: $M+1$st-Bid Decision.* A verifiable mix and match [5] is performed in this phase. The validity of the $M+1$st bidding point $j$ where $MixMatch[c_j, \{z^0, z^1, \ldots, z^M\}] = False$ but $MixMatch[c_{j+1}, \{z^0, z^1, \ldots, z^M\}] = True$ is publicly verifiable. Thus, the $M+1$st bidding price is publicly verifiable.

(5) *Phase 5: Winner Decision.* Let $a_{ij} = (u_{ij}, v_{ij})$. $B$ decryption messages $u_{i,j_{M+1st}+1}^{x_i}; i = 1, \ldots, B$ are submitted by each bidder $B_i$. Therefore, the

decryption on $a_{i,j_{M+1st}+1}$ is verifiable. The decision of winners is publicly verifiable.

(vii) *Correctness:*

(1) *Phase 1: Bidder Initialization.* All public keys $y_i = g^{x_i}, i = 1, \ldots, B$ are publicly verified. Thus, the correctness holds.

(2) *Phase 2: Submission of Bids by Bidders.* All secure shuffled bidding vectors $V'_i, i = 1, \ldots, B$ are publicly verified. $V'_i$ is a secure shuffle of $V_i$. The correctness holds.

(3) *Phase 3: Bid Verification.* A correct bidding vector $V_i$ should contain $P - 1$ amount of $Z^0$ and one $Z^1$. Since $V'_i$ is a secure shuffle of $V_i$, $V_i$ is correct if and only if $V'_i$ also contains $P - 1$ amount of $Z^0$ and one $Z^1$. Thus, the correctness holds.

(4) *Phase 4: $M + 1st$-Bid Decision.* The bidding point $j_{M+1st}$ is correct since $MixMatch[c_{j_{M+1st}}, \{z^0, z^1, \ldots, z^M\}] = False$, but $MixMatch[c_{j_{M+1st}+1}, \{z^0, z^1, \ldots, z^M\}] = True$. There are less than $M$ bidders' bids that are larger than or equal to $j_{M+1st} + 1$, but there are more than $M$ bidders' bids that are larger than or equal to $j_{M+1st} + 1$. Thus, $j_{M+1st}$ is the $M + 1$st bidding point. The correctness holds.

(5) *Phase 5: Winner Decision.* If $a_{i,j_{M+1st}+1} = Z^1$, then the bidder $B_i$'s bidding point $b_i$ is larger than or equal to $j_{M+1st} + 1$. Thus, this bidder is a winning bidder. The correctness holds.

(viii) *Financial fairness:* all bidders deposit $d$ amount Ether (in Wei) in the smart contract as a stake in phase 1. In the failure condition part, malicious bidders will compensate other bidders by their stake.

## 5. Implementation and Optimization

In this section, we introduce our implementation, optimization, and benchmarks. This protocol consists of smart contract (https://github.com/tonypottera24/m-1st_auction_dlp_sol) and web3 clients (https://github.com/tonypottera24/m-1st_auction_dlp_py). The smart contract is implemented by Solidity 0.7.x with experimental ABIEncoderV2. We also used the "solidity-BigNumber" library (https://github.com/zcoinofficial/solidity-BigNumber) for big number computation. In the client part, we use Python library web3.py (https://github.com/ethereum/web3.py) for better big number support. In terms of the simulation environment, we use Ganache (https://www.trufflesuite.com/ganache), a testnet built by Truffle, to execute and estimate the gas usage.

The computational costs of discrete logarithm problem (DLP) based algorithms are usually significantly affected by the key size. Figure 5 shows the gas usage by using 1024-, 2048-, and 3072-bit DLP and ECC P256. A common way to solve this problem is to use elliptic curve cryptography (ECC). Our ECC contract used the "elliptic-curve-solidity" (https://github.com/witnet/elliptic-curve-solidity) library, which is well tested and provides common NIST series curves such as secp256r1 (P256). As an Ethereum virtual machine always uses 256-bit integers, the gas consumption should not have significant differences between P192, P224, and P256. The ECC P256 version can save 69% gas on average compared with 3072-bit DLP. Table 2 shows implementation results of AS [6] and our scheme. Without using full mix and match and applying some optimization (see Section 6), we can reduce 33% gas on a 3-bidder and 6-bidding price setting.

## 6. Comparison

In this section, we compared the performance of our scheme with those of the AS [6], OM [7], GY [8], and MMO [9], as shown in Table 3; compared with other schemes, we do not need a trusted manager. The only role in our scheme is bidder. This design matches the ultimate goal of smart contract protocol, decentralized and trustless. As one of our main contributions in the bid verification phase, we use verifiable shuffle, i.e., the mix part, instead of the whole mix and match protocol as other related works do.

Detailed comparisons of phases 1 and 3 are as follows:

(1) *Phase 1: Bidder Initialization.* In previous research [6, 9, 11, 12], the bids are encrypted only by managers' public key. The bid secrecy, bidder anonymity, and many other properties all relied on the trusted manager. On the other hand, in our scheme, the ciphertexts are encrypted by all bidders' public keys. Without all bidders' collaboration, no one can break the bid secrecy and bidder anonymity.

(2) *Phase 3: Bid Verification.* A valid bidding vector $V_i = (V_{i1}, \ldots, V_{iP})$ should contain exactly $P - 1$ amounts of $Z^0$ and one $Z^1$. In previous research [6, 9, 11, 12], bidding vector verification contains two parts: (1) $V_{i1} \in \{Z^0, Z^1\}$; (2) $\sum_{j=1}^{P} V_{ij} = Z^1$. The first part is accomplished by mix and match [5]. This requires $T$ (trusted) mix servers to perform mix (secure shuffle [23–25]) and match (zk equality proof). The second part is accomplished by asking trusted manager to decrypt $\sum_{j=1}^{P} V_{ij}$.

In our scheme, we only use the match part (secure shuffle) of the mix and match protocol to prove these two parts simultaneously. By all bidders' collaboration, we decrypt $V'_i$ (a publicly verifiable shuffle of $V_i$). SC can verify if there is exactly $P - 1$ amounts of $Z^0$ and one $Z^1$ in $V'_i$ without leaking any secret.

Table 4 compares our protocol with previous protocols. In previous research, trusted managers need to use mix and match on $M$ values to verify $P$ ciphertexts in bidding vector $V_i$ for all $B$ bidders. Therefore, the time complexity of manager is $O(BPM)$ [6–8]. The time complexity and storage complexity on SC are proportional to the number of trusted manager $T$, i.e., $O(TBPM)$ overall. By removing the manager and letting bidders act as managers, the time complexity of our bidder is as

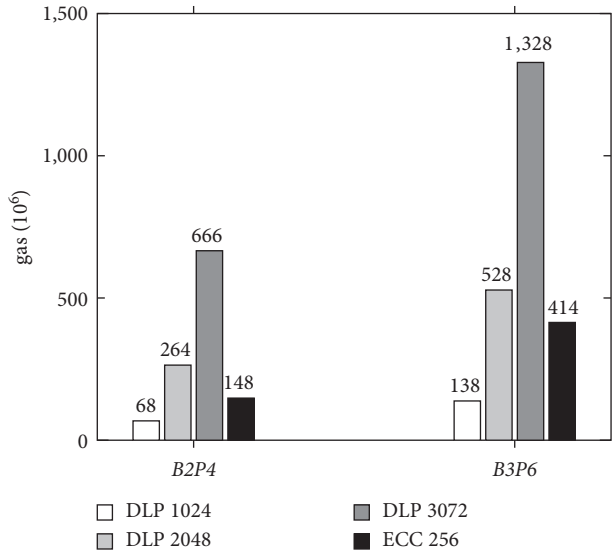


Figure 5: Gas usage of 1024-, 2048-, and 3072-bit DLP and 256-bit ECC (B2P4: 2 bidders and 4 bidding prices; B4P6: 4 bidders and 6 bidding prices).

Table 2: Comparison of AS [6] gas usage ($10^6$) and our scheme (ECC 256: 3 bidders and 6 bidding prices).

| | AS [6] | | This scheme | |
|---|---|---|---|---|
| | Manager | Bidder | Bidder | |
| (1) Initialization | 1 | — | 1 | (−0%) |
| (2) Submission of bids by bidders | 5 | 5 | 3 | (−40%) |
| (3) Bid verification | 520 | — | 350 | (−33%) |
| (4) $M$ + 1st-price decision | 79 | — | 52 | (−34%) |
| (5) Winner decision | 9 | — | 8 | (−11%) |
| (6) Payment | 4 | 4 | 2 | (−50%) |
| Overall | 618 | 9 | 416 | (−33%) |

Table 3: Comparison of related works and our scheme (TM: trusted manager).

| | Trusted manager | Bid verification | Posterior secrecy and anonymity | Public verifiability | Robustness | Financial fairness |
|---|---|---|---|---|---|---|
| AS [6] | Yes | Mix and match | TM based | No | TM based | No |
| OM [7] | Yes | Mix and match | TM based | No | TM based | No |
| MMO [9] | Yes | Mix and match | TM based | Interactive | TM based | No |
| GY [8] | Yes | Commitment | TM based | No | TM based | Yes |
| Our scheme | No | Verifiable shuffle | Yes | Yes | Yes | Yes |

Table 4: Comparison of the complexity with related works ($B$: the number of bidders, $P$: the number of bidding prices, $M$: number of goods, and $T$: the number of trusted managers and trusted mix servers).

| | Time per manager | Time per bidder | Time SC | Storage SC |
|---|---|---|---|---|
| AS [6] | $O(BPM)$ | $O(P)$ | $O(TBPM)$ | $O(TBPM)$ |
| OM [7] | $O(BPM)$ | $O(P)$ | $O(TBPM)$ | $O(TBPM)$ |
| MMO [9] | $O(B \log PM)$ | $O(\log PM)$ | $O(TB \log PM)$ | $O(TB \log PM)$ |
| GY [8] | $O(BPM)$ | $O(P)$ | $O(TBPM)$ | $O(TBPM)$ |
| Our scheme | — | $O(BPM)$ | $O(B^2PM)$ | $O(BPM)$ |

good as the time complexity of the manager in previous research. On the other hand, as one of our contributions, we applied optimization to reduce the storage complexity from $O(B^2PM)$ to $O(BPM)$. In previous research, the multiparty ElGamal ciphertext is in a format of $(g^{r_1}, \ldots, g^{r_T}, g^m y_1^{r_1} \cdots y_T^{r_T})$. However, it is meaningless to use different randomness $r_1, \ldots, r_T$ since the ciphertexts are created by the same bidder. Only $(g^r, g^m (y_1 \cdots y_T)^r)$ is enough.

## 7. Conclusion

We propose an efficient and secure $M$ + 1st-price auction protocol without a trusted manager under a malicious model. Out protocol satisfies anonymity, robustness, correctness, public verifiability, and financial fairness.

## Data Availability

The data used to support the findings of this study are included within the article.

## Conflicts of Interest

The authors declare that they have no conflicts of interest.

## Acknowledgments

This study was partially supported by enPiT (Education Network for Practical Information Technologies) of MEXT and Innovation Platform for Society 5.0 of MEXT.

## References

[1] G. Wood, "Ethereum: a secure decentralised generalised transaction ledger," *Ethereum project yellow paper*, vol. 151, no. 2014, pp. 1–32, 2014.

[2] J. Song, Q. Zhong, W. Wang, C. Su, Z. Tan, and Y. Liu, "FPDP: flexible privacy-preserving data publishing scheme for smart agriculture," *IEEE Sensors Journal*, vol. 21, no. 16, pp. 17430–17438, 2020.

[3] W. Wang, H. Huang, L. Zhang, and C. Su, "Secure and efficient mutual authentication protocol for smart grid under blockchain," *Peer-to-Peer Networking and Applications*, vol. 14, no. 5, pp. 2681–2693, 2020.

[4] W. Wang, H. Xu, M. Alazab, T. R. Gadekallu, Z. Han, and C. Su, "Blockchain-based reliable and efficient certificateless signature for IIOT devices," *IEEE Transactions on Industrial Informatics*, p. 1, 2021.

[5] M. Jakobsson and A. Juels, "Mix and match: secure function evaluation via ciphertexts," in *Proceedings of the Advances in Cryptology - ASIACRYPT 2000. In: International Conference on the Theory and Application of Cryptology and Information Security*, pp. 162–177, Springer, Kyoto, Japan, December 2000.

[6] M. Abe and K. Suzuki, "M + 1-st price auction using homomorphic encryption," in *Proceedings of the Key Cryptography. In: International Workshop on Public Key Cryptography*, pp. 115–124, Springer, Paris, France, February 2002.

[7] K. Omote and A. Miyaji, "A second-price sealed-bid auction with verifiable discriminant of p 0-th root," in *Proceedings of the International Conference on Financial Cryptography*, pp. 57–71, Springer, Southampton, Bermuda, March 2002.

[8] H. S. Galal and A. M. Youssef, "Verifiable sealed-bid auction on the ethereum blockchain," in *Proceedings of the International Conference on Financial Cryptography and Data Security*, pp. 265–278, Springer, Nieuwpoort, Curaçao, March 2018.

[9] T. Mitsunaga, Y. Manabe, and T. Okamoto, "A secure M + 1st price auction protocol based on bit slice circuits," *IEICE-Transactions on Fundamentals of Electronics, Communications and Computer Sciences*, vol. E99.A, no. 8, pp. 1591–1599, 2016.

[10] P. C. Hsu and A. Miyaji, "Verifiable m+ lst-price auction without manager," in *Proceedings of the IEEE Conference on Dependable and Secure Computing (DSC)*, pp. 1–8, IEEE, Aizuwakamatsu, Japan, January 2021.

[11] K. Kurosawa and W. Ogata, "Bit-slice auction circuit," in *Proceedings of the Computer Security - ESORICS 2002. In: European Symposium on Research in Computer Security*, pp. 24–38, Springer, Zurich, Switzerland, October 2002.

[12] T. Mistunaga, Y. Manabe, and T. Okamoto, "A secure M + 1st price auction protocol based on bit slice circuits," in *Proceedings of the Advances in Information and Computer Security: International Workshop on Security*, pp. 51–64, Springer, Tokyo, Japan, November 2011.

[13] M. Abe, "Universally verifiable mix-net with verification work independent of the number of mix-servers," in *Proceedings of the International Conference on the Theory and Applications of Cryptographic Techniques*, pp. 437–447, Springer, Espoo, Finland, May 1998.

[14] D. Wikström, "A universally composable mix-net," in *Proceedings of the Theory of Cryptography Conference*, pp. 317–335, Springer, Cambridge, MA, USA, February 2004.

[15] A. Kosba, A. Miller, E. Shi, Z. Wen, and C. Papamanthou, "Hawk: the blockchain model of cryptography and privacy-preserving smart contracts," in *Proceedings of the IEEE symposium on security and privacy (SP)*, pp. 839–858, IEEE, San Jose, CA, USA, May 2016.

[16] A. Juels and M. Wattenberg, "A fuzzy commitment scheme," in *Proceedings of the 6th ACM Conference on Computer and Communications Security*, pp. 28–36, Singapore, November 1999.

[17] S. Wu, Y. Chen, Q. Wang, M. Li, C. Wang, and X. Luo, "Cream: a smart contract enabled collusion-resistant e-auction," *IEEE Transactions on Information Forensics and Security*, vol. 14, no. 7, pp. 1687–1701, 2018.

[18] Y. H. Chen, S. H. Chen, and I. C. Lin, "Blockchain based smart contract for bidding system," in *Proceedings of the IEEE International Conference on Applied System Invention (ICASI)*, pp. 208–211, IEEE, Taiwan, China, April 2018.

[19] H. S. Galal and A. M. Youssef, "Trustee: full privacy preserving vickrey auction on top of ethereum," arXiv preprint, 2019, https://arxiv.org/pdf/1905.06280.

[20] T. ElGamal, "A public key cryptosystem and a signature scheme based on discrete logarithms," *IEEE Transactions on Information Theory*, vol. 31, no. 4, pp. 469–472, 1985.

[21] J. L. Bentley, "Multidimensional binary search trees used for associative searching," *Communications of the ACM*, vol. 18, no. 9, pp. 509–517, 1975.

[22] K. Huang and R. Tso, "A commutative encryption scheme based on elgamal encryption," in *Proceedings of the 2012 International Conference on Information Security and Intelligent Control*, pp. 156–159, IEEE, Yunlin, Taiwan, August 2012.

[23] C. A. Neff, "A verifiable secret shuffle and its application to e-voting," in *Proceedings of the 8th ACM conference on Computer and Communications Security*, pp. 116–125, Philadelphia, PA, USA, November 2001.

[24] J. Furukawa and K. Sako, "An efficient scheme for proving a shuffle," in *Proceedings of the Annual International Cryptology Conference*, pp. 368–387, Springer, Santa Barbara, California, USA, August 2001.

[25] J. Groth and S. Lu, "A non-interactive shuffle with pairing based verifiability," in *Proceedings of the International Conference on the Theory and Application of Cryptology and Information Security*, pp. 51–67, Springer, Kuching, Malaysia, December 2007.