Batch Verification of Validity of Bids in Homomorphic E-auction

Kun Peng, Colin Boyd and Ed Dawson

Level 7, 126 Margaret St, Brisbane QLD4000, Information Security Institute, Queensland University of Technology, Australia

Abstract

Bid opening in e-auction is efficient when a homomorphic secret sharing function is employed to seal the bids and homomorphic secret reconstruction is employed to open the bids. However, this high efficiency is based on an assumption: the bids are valid (e.g. within a special range). An undetected invalid bid can compromise correctness and fairness of the auction. Unfortunately, validity verification of the bids is ignored in the auction schemes employing homomorphic secret sharing (called homomorphic auction in this paper). In this paper, an attack against the homomorphic auction in the absence of bid validity check is presented and a necessary bid validity check mechanism is proposed. Then a batch cryptographic technique is introduced and applied to improve the efficiency of bid validity check.

Key words: homomorphic e-auction, bid validity check, batch verification, oblivious transfer

1 Introduction

In a sealed-bid auction scheme, each bidder chooses his evaluation from a number of biddable prices and submits it to some auctioneers, who then open the bids and determine the winning price and winner(s) according to a predefined auction rule. The commonly applied auction rules include first bid auction (the bidder with the highest bid wins and pays the highest bid), Vickrey auction (the bidder with the highest bid wins and pays the second highest bid) and the ρ^{th} bid auction (the bidders with the $\rho - 1$ highest bids win, pay the ρ^{th} highest bid and each get an identical item). The first-bid auction and Vickrey auction can be regarded as special cases of the ρ^{th} bid auction, which is a general solution. An auction must be correct, namely the auction result is strictly determined according to the auction rule. Fairness is necessary in any auction such that no bidder can take advantage over other bidders. Usually, bid privacy must be kept in an auction scheme, which means in the course of bid opening no losing bid is revealed.

When bid privacy must be kept in a non-interactive auction [1] , an efficient bid opening function is homomorphic secret reconstruction [8,10,9,15]. To adopt this bid opening function, one-selection-per-price principle and homomorphic bid sharing mechanism must be employed. Each bidder has to submit a bidding selection at every biddable price to indicate whether he is willing to pay that price ("YES" or "NO"). Every selection is sealed with a homomorphic secret sharing function, so that the auctioneers can use a homomorphic secret reconstruction function to determine whether the number of bidders willing to pay a price is over ρ without revealing any bidding selection. When this homomorphic bid opening mechanism is applied together with binary search strategy, the winning bid can be determined very efficiently.

In homomorphic e-auction, each bidding selection must be in some special range (certain values standing for "YES" or "NO") to guarantee correctness and fairness of the auction. So validity of the bids must be proved by the bidders and verified publicly. However, all the existing homomorphic auction schemes [8,10,9,15] ignore bid validity check. An attack to compromise correctness and fairness in the absence of bid validity check is presented in this paper to demonstrate necessity of bid validity check. Then implementation of bid validity check in homomorphic auction is proposed. As proof and verification of bid validity is highly inefficient, a batch cryptographic technique is proposed and applied to improve the efficiency of bid validity check. With the help of a new 1-out-of-w oblivious transfer technique, validity of bids can be efficiently proved and verified.

2 Symbols and Parameters

The following symbols and parameters will be used in this paper.

- w represent the number of biddable prices in auction.
- $E()$ denotes encryption.
- $D()$ denotes decryption.
- $\langle x \rangle$: the bit length of integer x.
- $ExpCost(x)$ denotes the number of multiplications needed to calculate an exponentiation with an x-bit exponent. $ExpCost^y(x)$ denotes the number of multiplications needed to calculate the product of y exponentiations with x-bit exponents.

[1] An auction is non-interactive if no communication between the bidders and the auctioneers is needed after the bids are submitted.

- Two large primes p and q are chosen, such that $p = 2q + 1$ and $w < q$. Integer g_0 is a generator of Z_p^*. Integers g and h are generators of G, the subgroup with order q of Z_p^*, such that $\log_g h$ is unknown.
- **Definition 1** $|\ |$ *is the absolute-value function from* Z_p^* *to* G *defined by*

$$|\sigma| = \begin{cases} \sigma & \text{if } \sigma \in G \\ g_0^q \sigma & \text{if } \sigma \in Z_p^* \setminus G \end{cases}$$

- RSA encryption: $N = p'q'$ where p' and q' are large primes. The public key is e where $e \in Z_N^*$. The private key is d, such that $ed = 1 \bmod \phi(N)$.
- $KN(x)$ denotes knowledge of x; $KN(x|Cond)$ denotes knowledge of x satisfying condition $Cond$.

3 Related Work

Homomorphic auction and two related cryptographic tools, batch verification and oblivious transfer are recalled in this section. The two tools will be improved and then employed later in this paper to optimize homomorphic auction.

3.1 Bid Sharing

Let $p_1, p_2, \ldots, p_w$ stand for the biddable prices in an auction. A bid in a homomorphic auction [8,10,9,15] is a vector $m = (m_1, m_2, \ldots, m_w)$. Each component in m is either "YES" or "NO", respectively represented by a non-zero integer and zero. Bid m is shared among the auctioneers. A threshold secret sharing function with a threshold t is employed to share the bid, such that the bid can be recovered if t shares are available. Due to additive homomorphism of the secret sharing algorithm, bid opening at any price can be implemented using homomorphic secret reconstruction without revealing any selection. The bid sharing must be verifiable, so it can be verified that the shares can be used to recover a unique secret bid. In the usual sharing verification mechanism, the secret holder hides the secret in a commitment, so that each share holder can verify that his share can be used to recover the secret hidden in the commitment. The commitment function must be hiding (the secret is not revealed) and binding (the secret cannot be changed). In a bid sharing, the commitment function must be information-theoretically hiding as each bidding selection can only be chosen from two values standing for "YES" and "NO" respectively. So general purpose VSS schemes, such as [4] and [13] are not suitable as they do not employ information-theoretically hiding commitment. As suggested in

[15], a VSS scheme with information-theoretically hiding commitment like [14] must be employed. In [15], the most advanced homomorphic auction scheme, every m_j is shared as follows where $1 \leq j \leq w$.

(1) A polynomial $f_j(x) = m_j + \sum_{k=1}^{t} a_{j,k}x^k \bmod q$ is chosen and $a_{j,k}$ for $k = 1, 2, \ldots, t$ are random integers.
(2) A random integer u_j is chosen.
(3) A polynomial $h_j(x) = u_j + \sum_{k=1}^{t} b_{j,k}x^k \bmod q$ is chosen where $b_{j,k}$ for $k = 1, 2, \ldots, t$ are random integers.
(4) Share $s_{j,l} = (m_{j,l}, u_{j,l}) = (f_j(l), h_j(l))$ is encrypted and sent to auctioneer A_l for $l = 1, 2, \ldots, M$.
(5) As the secret sharing is homomorphic, homomorphic secret reconstruction (see [15,14] for details) can be performed at any price to recover the sum of selections at that price in the bid opening phase.
(6) The bidder publishes commitments $e_{j,k} = g^{a_{j,k}}h^{b_{j,k}} \bmod p$ for $k = 1, 2, \ldots, t$ and $e_{j,0} = g^{m_j}h^{u_j} \bmod p$.
(7) Correctness of the sharing can be verified against the commitments. $s_{j,l}$ is a correct share to recover the secret bid committed in $e_{j,0}$ if and only if $g^{m_{j,l}}h^{u_{j,l}} = \prod_{k=0}^{t} e_{j,k}^{j^k}$. See [14] for correctness and soundness of this sharing verification.

3.2 Batch Verification Techniques

Suppose it is required to verify $y_j = g^{x_j} \bmod p$ where $x_j \in z_q$, $y_j \in z_p^*$, for $j \in 1, 2, \ldots, w$. The easiest solution is to calculate g^{x_j} for $j = 1, 2, \ldots, w$ and compare the results with y_j for $j = 1, 2, \ldots, w$, the computation cost of which is $w \times ExpCost(log_2 q)$. An intuitive idea to batch verify the w equations is to test $\prod_{j=1}^{w} y_j = g^{\sum_{j=1}^{w} x_j}$. Harn [6] used this idea to construct a batch verification for DSA. However this method is not reliable since it is easy to pass the verification with incorrect pairs $y_j \neq g^{x_j}$ as pointed out by Bellare *et al* [2].

Bellare *et al* [2] proposed three batch verification schemes for common base exponentiation: RS (random subset) test, SE (small exponent) test and Bucket test. In SE test, the batch verification equation is $\prod_{j=1}^{w} y_j^{s_j} = g^{\sum_{j=1}^{w} x_j s_j}$ where $\langle s_j \rangle = L$ for $j = 1, 2, \ldots, w$ and L is a security parameter. RS test is a special case of SE test when = 1. Bucket test is a variant of SE test and more efficient when the batch is of great size (all the pairs (x_j, y_j) for $j \in 1, 2, \ldots, w$ are divided into 2^T buckets and SE test is performed in each the bucket). In this paper we focus on SE test.

Bellare *et al* proved that if $y_j \in G$ for $j = 1, 2, \ldots, w$ and q is a prime, SE test costs $w + L + wL/2 + ExpCost(log_2 q)$ multiplications, while Bucket test costs

$\lceil \frac{L}{T-1} \rceil (w + T + 2^{T-1}(T+2) + ExpCost(log_2 q))$ multiplications. They use SE test or Bucket test together with a slightly modified DSS scheme to achieve efficient batch signature verification. Bellare proved that the probability that an incorrect batch can pass the batch verification (SE test or bucket test) is no more than 2^{-L}. Although SE test and Bucket test are sound with an overwhelmingly large probability, their soundness is based on an assumption: $y_j \in G$ for $j \in 1, 2, \ldots, w$. Otherwise incorrect pair $((x_j, y_j)$ with $y_j \neq g^{x_j})$ can pass the batch verification with a non-negligible probability. Bellare *et al* seem to ignore the impact of this assumption.

Boyd *et al* [3] pointed out that the theorems in [2] are correct, but their application to DSS verification is inappropriate because there is no efficient way to verify $y_j \in G$ for $j = 1, 2, \ldots, w$. If $y_j \in G$ is verified for $j = 1, 2, \ldots, w$, w exponentiations with full-length exponents are costed in the verification, which compromise the efficiency improvement of the batch verification. When $y_j \notin G$, the probability for an incorrect batch to pass the verification can be much greater than 2^{-L} (When $y_j = g_0^q g^{x_j}$, the probability is 0.5). This observation greatly restricts the application of Bellare's batch verification techniques. However, it is illustrated in [1] that carefully-designed SE test is still useful in some special applications.

3.3 Oblivious Transfer

Oblivious transfer is a function involving two parties: a sender and a chooser. The sender possesses a few secrets. The chooser chooses one of the secrets and gets it from the sender. The sender does not know which secret the chooser gets and the chooser has no knowledge of any other secret.

When there are only two secrets to choose from in an oblivious transfer, it is called 1-out-of-2 oblivious transfer [12]. When there are more than two secrets to choose from in an oblivious transfer, it is called 1-out-of-n oblivious transfer [11,17]. Most oblivious transfer schemes are based on ElGamal encryption. Juels and Szydlo proposed an 1-out-of-2 oblivious transfer based on RSA encryption [7], which is more efficient than those based on ElGamal encryption.

4 Necessity and Implementation of Bid Validity Check

Necessity and implementation of bid validity check in homomorphic auction are discussed in this section.

4.1 Necessity of Bid Validity Check

Bid validity check is necessary in homomorphic auction although it is not adopted in the existing homomorphic auction schemes [8,10,9,15]. In a Vickrey auction or the ρ^{th}-bid auction, a constant non-zero integer must be chosen to represent the "YES" selection, so validity of bid must be checked. Otherwise, it is impossible to test whether the number of "YES" selections at a price is at least 2 (in Vickrey auction) or ρ (in the ρ^{th}-bid auction). Although bid validity has never been mentioned in any first-bid auction scheme, an invalid bid can compromise correctness and fairness of a first-bid auction scheme.

For example, three colluding bidders B_1, B_2 and B_3 may perform the following attack against first bid auction where in a bidding choice no-zero integer Y and 0 stand for "YES" and "NO" respectively.

- B_1, B_2 and B_3 estimate that the other bidders' bids are lower than p_μ while their own evaluation is p_ν, which is higher than p_μ. They try to win the auction and pay as low as possible.
- B_1 bids Y at prices no higher than p_μ and zero at other prices; B_2 bids Y at prices no higher than p_ν and zero at other prices; B_3 bids $-Y$ at prices higher than p_μ but no higher than p_ν and zero at other prices.
- If all other bidder submits a bid lower than p_μ as expected, the sum of choices at p_μ is non-zero and the sum of choices at prices higher than p_μ is 0. So p_μ is the winning price and there is a tie between B_1 and B_2. One of them gives up and the other wins at p_μ.
- If other bidders' highest bid, p_H is no lower than p_μ but lower than p_ν, the sum of choices at p_H is larger than zero and the sum of choices at prices higher than p_H is 0. So some other bidder wins the auction at p_H together with B_2. B_2 disputes the tie and publishes his bid to win the auction at p_ν.
- If other bidders' highest bid is p_ν, the sum of choices at p_ν is larger than zero and the sum of choices at prices higher than p_ν is 0. So some other bidder draws with B_2 at p_ν. B_2 still has a chance to win the auction in the following tie-breaking operation.

With this attack, either B_1 or B_2 win unless another bidder submits a bid higher than the attackers' evaluation. The attackers can pay a price lower than their evaluation if the other bids are as low as the attackers expect. So, no matter what auction rule is applied, bid validity check is always necessary.

4.2 Implementation of Bid Validity Check

Bid validity check in homomorphic auctions can be implemented using zero knowledge proof techniques. When the verifiable bid sharing mechanism in

[15] as described in Section 3.1 is employed, the bidder has to prove that the bid committed in $(e_{1,0}, e_{2,0}, \ldots, e_{w,0})$ is valid. For simplicity, integer 1 is used to represent a "YES" selection. So when the bid is the δ^{th} biddable price, zero must be committed in $e_{1,0}, e_{2,0}, \ldots, e_{\delta-1,0}$ while one must be committed in $e_{\delta,0}, e_{\delta+1,0}, \ldots, e_{w,0}$. Validity of the bid can be proved using the following zero-knowledge proof.

$$KN(\log_h e'_{j,0}) \vee \ KN(\log_h(e'_{j,0}/g)) \text{ for } j = 1, 2, \ldots, w \tag{1}$$

and

$$KN(\log_h((\prod_{j=1}^{w} e'_{j,0})/g)) \tag{2}$$

where $e'_{j,0} = e_{j,0}/e_{j-1,0}$ for $j = 2, 3, \ldots, w$ and $e'_{1,0} = e_{1,0}$.

Theorem 1 *If and only if (1) and (2) can be proved, the bid committed in* $(e_{1,0}, e_{2,0}, \ldots, e_{w,0})$ *is valid.*

Proof: Proof (1) indicates that either zero or one is committed in $e'_{j,0}$ for $j = 1, 2, \ldots, w$. Proof (2) indicates that the sum of the committed bidding selections in $(e'_{1,0}, e'_{2,0}, \ldots, e'_{w,0})$ is 1 mod q. As $w < q$, the sum of the committed bidding selections in $(e'_{1,0}, e'_{2,0}, \ldots, e'_{w,0})$ is 1. So one 1 and $w-1$ zeros are committed in $(e'_{1,0}, e'_{2,0}, \ldots, e'_{w,0})$ if and only if (1) and (2) can be proved. As $e'_{j,0} = e_{j,0}/e_{j-1,0}$ for $j = 2, 3, \ldots, w$ and $e'_{1,0} = e_{1,0}$, zero is committed in $e_{1,0}, e_{2,0}, \ldots, e_{\delta-1,0}$ and one is committed in $e_{\delta,0}, e_{\delta+1,0}, \ldots, e_{w,0}$ for a δ in $\{1, 2, \ldots, w\}$ if and only if (1) and (2) can be proved. □

The normal method to verify equation (1) is to perform w instances of proof of 1-out-of-2 knowledge of logarithms based on [16] and [5]. This method is highly costly. The computational cost for a bidder is $e'_{j,0}$ for $3wExpCost(|q|) + 2w$ multiplications. The computational cost of a verifier (auctioneer or observer) is $(4ExpCost(|q|) + 2)nw$ multiplications where n is the number of bidders.

5 Efficiency Improvement of Bid Validity Check

The bid validity check mechanism in Section 4.2 is so inefficient, that it becomes an efficiency bottleneck. So, it is optimised in efficiency in this section. The efficiency improvement is based on two new cryptographic primitives: a new batch verification primitive and a new 1-out-of-w oblivious transfer technique, which are extensions of the existing schemes described in Section 3.

Theorem 2 is applied to verify knowledge of $\log_g y_i$ where $y_i \in G$ for $i = 1, 2, \ldots, w$. Unless specified, any multiplicative computation in Theorem 2 and its proof occurs in G with a modulus p.

Theorem 2 *Suppose $y_i \in G$ for $i = 1, 2, \ldots, w$. Let L be a security parameter such that $2^L < q$. If there is an efficient deterministic algorithm, which with a probability bigger than 2^{-L} can calculate $\log_g \prod_{i=1}^{w} y_i^{t_i}$ with random t_i for $i = 1, 2, \ldots, w$ where $t_i < 2^L$, then there is an efficient deterministic algorithm to calculate $\log_g y_i$ for $i = 1, 2, \ldots, w$.*

Proof: Given any integer v in $\{1, 2, \ldots, w\}$, there must exist integers $t_1, t_2, \ldots, t_w$ and $\hat{t}_v$ in $\{0, 1, \ldots, 2^L - 1\}$ such that there is an efficient deterministic algorithm to calculate $\log_g \prod_{i=1}^{w} y_i^{t_i}$ and $\log_g((\prod_{i=1}^{v-1} y_i^{t_i}) y_v^{\hat{t}_v} \prod_{i=v+1}^{w} y_i^{t_i})$ where $t_v \neq \hat{t}_v$.

Otherwise, given any $t_1, t_2, \ldots, t_{v-1}, t_{v+1}, \ldots, t_w$, any efficient deterministic algorithm can calculate $\log_g \prod_{i=1}^{w} y_i^{t_i}$ for at most one t_v. This deduction implies among the 2^{wL} possible combinations of $t_1, t_2, \ldots, t_w$, only $2^{(w-1)L}$ of them enable an efficient deterministic algorithm to calculate $\log_g \prod_{i=1}^{w} y_i^{t_i}$, which leads to a contradiction: the probability that there exists an efficient deterministic algorithm to calculate $\log_g \prod_{i=1}^{w} y_i^{t_i}$ with random $t_1, t_2, \ldots, t_w$ is no more than 2^{-L}.

So there exists an efficient deterministic algorithm to calculate

$$\log_g \prod_{i=1}^{w} y_i^{t_i} - \log_g((\prod_{i=1}^{v-1} y_i^{t_i}) y_v^{\hat{t}_v} \prod_{i=v+1}^{w} y_i^{t_i}) = \log_g y_v^{t_v - \hat{t}_v} = (t_v - \hat{t}_v) \log_g y_v \bmod q$$

Note that $t_v \neq \hat{t}_v$, $t_v, \hat{t}_v < 2^L < q$, so $t_v - \hat{t}_v \neq 0 \bmod q$. So there is an efficient deterministic algorithm to calculate $log_g y_v$. Therefore, there is an efficient deterministic algorithm to calculate $\log_g y_i$ for $i = 1, 2, \ldots, w$ as v can be any integer in $\{1, 2, \ldots, w\}$. □

According to Theorem 2 verification of knowledge of $log_g y_i$ for $i = 1, 2, \ldots w$ can be batched to verification of knowledge of $log_g \prod_{i=1}^{w} y_i^{t_i}$. The probability that $\log_g \prod_{i=1}^{w} y_i^{t_i}$ is known while $log_g y_i$ for some i in $\{1, 2, \ldots, w\}$ is unknown is no more than 2^{-L}.

5.2 New 1-out-of-w Oblivious Transfer

A 1-out-of-w oblivious transfer protocol must be employed in the optimised bid validity verification mechanism. In the application of oblivious transfer in this paper, only one value is transferred, so the 1-out-of-w oblivious transfer protocol is performed only once and security requirement for multiple transfers need not be considered. However, the employed 1-out-of-w oblivious transfer protocol must be very efficient to achieve high efficiency in the bid validity verification mechanism. So a new 1-out-of-w oblivious transfer based on RSA encryption is designed by extending [7] to 1-out-of-w circumstance while keeping its high efficiency. After the extension, the new protocol only provides one-time security, but is very efficient.

The sender has w secrets s_1, s_2, $\dots s_w$ and the chooser wants to know $s_\delta \in \{1, 2, \dots, w\}$. They run the following protocol where the chooser's operation is denoted as $OT1(s_\delta)$ and the sender's operation is denoted as $OT2(s_\delta)$.

(1) **Initialisation**: The sender sets up RSA encryption, keeps private key d and publishes public key e and N where $ed = 1 \bmod \phi(N)$. He randomly selects $\epsilon_{i,j}$ from Z_N for $i = 1, 2, \dots, \log_2 w$ and $j = 0, 1$ and publishes $s'_l = s_l - (\prod_{i=1}^{\log_2 w} \epsilon_{i,b_{l,i}}) \bmod N$ for $l = 1, 2, \dots w$ where $b_{l,i}$ denotes the i^{th} bit of l. He chooses randomly $c_i \in Z_N$ for $i = 1, 2, \dots \log_2 w$ and sends them to the chooser.

(2) $OT1(s_\delta)$**: choosing a secret**
The chooser chooses secrets $\tau_i \in N$ for $i = 1, 2, \dots \log_2 w$. He calculates $y_{i,b_{\delta,i}} = \tau_i^e \bmod N$ for $i = 1, 2, \dots, \log_2 w$. Then he calculates $y_{i,1 \oplus b_{\delta,i}} = y_{i,b_{\delta,i}} c_i \bmod N$ if $b_{\delta,i} = 0$ or $y_{i,1 \oplus b_{\delta,i}} = y_{i,b_{\delta,i}} / c_i \bmod N$ if $b_{\delta,i} = 1$ for $i = 1, 2, \dots \log_2 w$ where $\oplus$ stands for XOR. He sends $y_{i,0}$ and $y_{i,1}$ for $i = 1, 2, \dots \log_2 w$ to the sender in correct order.

(3) $OT2(s_\delta)$**: sending the secret**
The sender verifies $y_{i,1} = c_i y_{i,0} \bmod N$ for $i = 1, 2, \dots \log_2 w$ and sends $E_{i,0} = y_{i,0}^d \epsilon_{i,0} \bmod N$ and $E_{i,1} = y_{i,1}^d \epsilon_{i,1} \bmod N$ for $i = 1, 2, \dots \log_2 w$ to the chooser.

(4) **Obtaining the secret**: The chooser can only get $s_\delta = s'_\delta + (\prod_{i=1}^{\log_2 w} E_{i,b_{\delta,i}}) / \prod_{i=1}^{\log_2 w} \tau_i \bmod N$.

Properties of this 1-out-of-w oblivious transfer protocol are as follows.

(1) Correctness
If the sender and the chooser follow the protocol, the chooser can obtain s_δ as

$$s'_\delta + (\prod_{i=1}^{\log_2 w} E_{i,b_{\delta,i}}) / \prod_{i=1}^{\log_2 w} \tau_i \bmod N$$

$$= s'_\delta + (\prod_{i=1}^{\log_2 w} y^d_{i,b_{\delta,i}} \epsilon_{i,b_{\delta,i}}) / \prod_{i=1}^{\log_2 w} \tau_i \bmod N$$

$$= s'_\delta + (\prod_{i=1}^{\log_2 w} \tau_i^{ed} \epsilon_{i,b_{\delta,i}}) / \prod_{i=1}^{\log_2 w} \tau_i \bmod N$$

$$= s'_\delta + \prod_{i=1}^{\log_2 w} \epsilon_{i,b_{\delta,i}} \bmod N = s_\delta$$

(2) Privacy of the chooser
As $y_{i,0}$ and $y_{i,1}$ for $i = 1, 2, \ldots \log_2 w$ are distributed uniformly, the sender has no knowledge of δ. Namely, information-theoretic privacy of the chooser is achieved.

(3) Privacy of the sender
It is widely believed that composite factorization is intractable and without the knowledge of the factorisation of N it is intractable to find d given e and N. So the chooser can get only one of $\epsilon_{i,0}$ and $\epsilon_{i,1}$ for every i in $\{1, 2, \ldots, \log_2 w\}$. Therefore, the chooser does not know any other secret than s_δ.

(4) High efficiency
- The cost to the sender is $2\log_2 w$ exponentiations and $n(\log_2 w - 1) + 2\log_2 w$ multiplications;
- The cost to the chooser is $(\log_2 w)/2 + 1$ divisions and $1.5\log_2 w + 1$ multiplications on average if e is a small integer as suggested in [7].

In this paper, this 1-out-of-w oblivious transfer protocol will be applied to transfer L-bit integers, which are smaller than N.

5.3 Batch Verification of Bid Validity

A method to improve efficiency of bid validity check is to apply the batch ZK proof-verification technique in Section 5.1 to batch prove and verify validity of a bid. For example, according to Theorem 2, Proof (1) and Proof (2) can be batched into

$$KN(\delta \mid KN(\log_h(\prod_{j=1}^{w} e'^{t_j}_{j,0})/g^{t_\delta}),\ 1 \le \delta \le w) \tag{3}$$

where the bid is the δ^{th} biddable price and $t_1, t_2, \ldots, t_w$ are random L-bit integers. A simple implementation of Proof (3) is presented in Figure 1, where A and B represent a verifier (auctioneer) and a prover (bidder) respectively and t_j is much smaller than q.

$B \to A:$ $a = h^r \bmod p$ where r is randomly chosen from Z_q.
$A \to B:$ $t_1, t_2, \ldots, t_w$ where $t_j \in \{0,1\}^L$ for $j = 1, 2, \ldots, w$
$B \to A:$ $z = s - r \bmod q$ where $s = \sum_{j=1}^{w} u'_j t_j \bmod q$
where $u'_j = u_j - u_{j-1}$ for $j = 2, 3, \ldots, w$ and $u'_1 = u_1$.
Verification: $\prod_{j=1}^{w} {e'}_{j,0}^{t_j} = ah^z g^{t_\delta} \bmod p$

Fig. 1. Simple Batch Proof-Verification of Bid Validity

Although this approach is efficient, the verification equation $\prod_{j=1}^{w} {e'}_{j,0}^{t_j} = ah^z g^{t_\delta} \bmod p$ reveals that the bid is the δ^{th} biddable price. So the proof in Figure 1 is not a ZK proof of (3). Instead, it is a ZK proof of $KN(\log_h(\prod_{j=1}^{w} {e'}_{j,0}^{t_j})/g^{t_\delta})$, which reveals δ. Therefore, the batch proof-verification protocol in Figure 1 must be modified, so that the bid is not revealed. The following is a modified four-move protocol.

- In the first move (the added move) the prover and the verifier perform an oblivious transfer. Firstly, the verifier (auctioneer as the sender in the oblivious transfer) chooses challenges $t_1, t_2, \ldots, t_w$ and keep them secret. Then the prover (bidder as the chooser in the oblivious transfer) use the 1-out-of-w oblivious transfer protocol in Section 5.2 to obtain t_δ from the verifier while he obtains no knowledge of other secret challenges.
- The prover's commitment is the second move, which is a commitment of two secrets: t_δ and a random integer r.
- The last two moves (challenge and response) are the same as the last two moves in Figure 1.
- The verification equation is slightly modified so that t_δ does not appear explicitly in the verification equation and the bid is not revealed.

The protocol is described in detail in Figure 2. A and B represent a verifier (auctioneer) and a prover (bidder) respectively. According to Theorem 2 and the privacy property for the sender in the 1-out-of-w transfer, the protocol in Figure 2 proves

$$KN(\log_h e'_{j,0}) \text{ for } w-1 \text{ instances of } j \in \{1, 2, \ldots, w\} \quad (4)$$

Note that in his proof of the binding property of his VSS in [14], Pedersen assumed that $e_{j,0}$ is in the cyclic subgroup G. So although it is not explicitly described, $e_{j,0}$ for $j = 1, 2, \ldots, w$ must have been verified to be in G for the sake of verifiability of secret sharing. Therefore, $e'_{j,0} \in G$ for $j = 1, 2, \ldots, w$ and the condition in Theorem 2 is satisfied. This proof is much more efficient than Proof (1) in Section 4.2, but is as effective as it. In addition to Proof (2) (which can be proved and verified using a standard Schnnor ZK proof protocol [16]), Proof (4) guarantees that $(1, 0, 0 \ldots, 0)$ is committed in $e'_{1,0}, e'_{2,0}, \ldots, e'_{w,0}$ in an unknown order. The position of commitment of 1 (bidding information)

$B \rightarrow A:$	$OT_1(t_\delta)$
$A \rightarrow B:$	$OT_2(t_\delta)$,
$B \rightarrow A:$	$a = h^r g^{t_\delta} \bmod p$ where r is randomly chosen from Z_q.
$A \rightarrow B:$	$t_1, t_2, \ldots, t_w$ where $t_j \in \{0,1\}^L$ for $j = 1, 2, \ldots, w$
$B \rightarrow A:$	$z = s - r \bmod q$ where $s = \sum_{j=1}^{w} u'_j t_j$
	where $u'_j = u_j - u_{j-1}$ for $j = 2, 3, \ldots, w$ and $u'_1 = u_1$.
Verification: $\prod_{j=1}^{w} {e'}_{j,0}^{t_j} = ah^z \bmod p$	

Fig. 2. Batch Verification with Secret Sharing

is not revealed in the proof.

This new proof and verification cost $O(\log_2 w)$ full-length exponentiations and is much more efficient than before batch verification is applied.

6 Analysis

In this section we show that the optimised bid validity check mechanism using batch verification is correct, sound, zero knowledge and efficient.

6.1 Security Analysis

Security properties of batch verification of bid validity is analysed in this section. Correctness of the batch verification is obvious, namely when the bidders are honest their validity proof can successfully pass the verification. Other security properties like soundness and zero knowledge are not so obvious.

Theorem 3 *The proof protocol in Figure 2 is honest-verifier ZK.*

Proof: Without any help from the prover, the verifier can generate a proof transcript as follows. He randomly selects $OT_1(t_\delta)$,$OT_2(t_\delta)$, z from Z_q and $t_1, t_2, \ldots, t_w$ from $\{0,1\}^L$. Then he calculates $a = \prod_{j=1}^{w} {e'}_{j,0}^{t_j}/h^z$. Thus he independently gets a proof transcript $(OT_1(t_\delta), OT_2(t_\delta), a, t_1, t_2, \ldots, t_w, z)$. In this transcript, every integer is uniformly distributed. If the verifier is honest and randomly selects his challenge $t_1, t_2, \ldots, t_w$, the proof protocol in Figure 2 also generates a proof transcript $(OT_1(t_\delta), OT_2(t_\delta)a, t_1, t_2, \ldots, t_w, z)$, in which every integer is uniformly distributed. These two transcripts are indistinguishable. □

Theorem 4 *The proof protocol in Figure 2 is sound. More precisely, if the*

$B \to A:\ \ \delta$
$A \to B:\ \ t_\delta$
$B \to A:\ \ a = h^r g^{t_\delta} \bmod p$ where r is randomly chosen from Z_q
$A \to B:\ \ t_1, t_2, \ldots, t_{\delta-1}, t_{\delta+1}, \ldots, t_w$ where $t_j \in \{0,1\}^L$ for $j = 1, 2, \ldots, w$
$B \to A:\ \ z = s - r \bmod q$ where $s = \sum_{j=1}^{w} u'_j t_j$
Verification: $\prod_{j=1}^{w} e'^{t_j}_{j,0} = ah^z \bmod p$

Fig. 3. A protocol used in proof of Theorem 4

verifier is honest and the proof passes the verification in the protocol in Figure 2 with a non-negligible probability, the proof together with Proof (2) guarantees that $(1, 0, 0 \ldots, 0)$ *is committed in* $e'_{1,0}, e'_{2,0}, \ldots, e'_{w,0}$ *after being permuted.*

To prove Theorem 4, the following two lemmas are proved first.

Lemma 1 *If a prover can pass the protocol in either Figure 3 or Figure 2, then he can pass the other as well.*

Proof: As the oblivious transfer protocol guarantees privacy of sender, in the protocol in Figure 2 the prover only gets t_δ in the first two steps and has no information about $t_1, t_2, \ldots, t_{\delta-1}, t_{\delta+1}, \ldots, t_w$ until the fourth step. So in protocols in Figure 2 and Figure 3 the prover tries to prove the same statement while given the same knowledge. Therefore, he can pass the other as well if he can pass one of them. $\square$

Lemma 2 *The protocol in Figure 3 is sound. More precisely, if the verifier is honest and the prover can pass the verification in the protocol in Figure 3 with a non-negligible probability, then he knows* $\log_h e'_{1,0}, log_h e'_{2,0}, \ldots, log_h e'_{\delta-1,0}$, $log_h e'_{\delta+1,0} \ldots, log_h e'_{w,0}$.

Proof: As the prover can pass the verification in the protocol in Figure 3 with a non-negligible probability, the prover must be able to give two responses z and z' to two different challenges $t_1, t_2, \ldots, t_{\delta-1}, t_{\delta+1}, \ldots, t_w$ and $t'_1, t'_2, \ldots, t'_{\delta-1}, t'_{\delta+1}, \ldots, t'_w$ to the same commitment a, such that

$$\prod_{j=1}^{w} e'^{t_j}_{j,0} = ah^z \bmod p \tag{5}$$

$$\left(\prod_{j=1}^{\delta-1} e'^{t'_j}_{j,0}\right) e'^{t_\delta}_{\delta,0} \prod_{j=\delta+1}^{w} e'^{t'_j}_{j,0} = ah^{z'} \bmod p \tag{6}$$

with a probability larger than 2^{-L}. Otherwise, the prover can give correct response to at most one challenge with a non-negligible probability, while the probability that he can give a correct response to any other challenge is no larger than 2^{-L}. This deduction implies a contradiction: when an honest verifier chooses a random challenge from all the 2^{wL} possible challenges the probability that the prover can pass the verification is no more than $1/2^{wL} + 2^{-L}(2^{wL}-1)/2^{wL}$, which is negligible.

Dividing (5) with (6) yields

$$\prod_{j=1}^{\delta-1} {e'}_{j,0}^{t_j-t'_j} \prod_{j=\delta+1}^{w} {e'}_{j,0}^{t_j-t'_j} = h^{z-z'} \bmod p,$$

which is correct with a probability larger than 2^{-L}. So, the prover knows $\log_h(\prod_{j=1}^{\delta} {e'}_{j,0}^{t_j-t'_j} \prod_{j=\delta+1}^{w} {e'}_{j,0}^{t_j-t'_j}) = z - z'$ with a probability larger than 2^{-L}.

As the verifier is honest and chooses $t_1, t_2, \ldots, t_{\delta-1}, t_{\delta+1}, \ldots, t_w$ and $t'_1, t'_2, \ldots, t'_{\delta-1}, t'_{\delta+1}, \ldots, t'_w$ randomly, $t_1 - t'_1, t_2 - t'_2, \ldots, t_{\delta-1} - t'_{\delta-1}, t_{\delta+1} - t'_{\delta+1}, \ldots, t_w - t'_w$ are random. So according to Theorem 2, the prover knows $\log_h e'_{1,0}, log_h e'_{2,0}, \ldots, log_h e'_{\delta-1,0}, log_h e'_{\delta+1,0} \ldots, log_h e'_{w,0}$. □

Proof of Theorem 4: According to Lemma 1, if the proof protocol in Figure 2 passes the verification with a non-negligible probability when the verifier is honest, the prover can also pass the protocol in Figure 3 with the same non-negligible probability when the verifier is honest.

So according to Lemma 2, the prover knows $\log_h e'_{1,0}, log_h e'_{2,0}, \ldots, log_h e'_{\delta-1,0}, log_h e'_{\delta+1,0} \ldots, log_h e'_{w,0}$. Proof (2) guarantees that the prover knows $\log_h(\prod_{j=1}^{w} e'_{j,0})/g$. So the prover knows $\log_h e'_{\delta,0}/g$. As the prover does not know $\log_g h$ and the commitment of shares in [14] is binding, 1 is committed in $e'_{\delta,0}$ and 0 is committed in $e'_{1,0}, e'_{2,0}, \ldots, e'_{\delta-1,0}, e'_{\delta+1,0} \ldots, e'_{w,0}$ Therefore, $(1, 0, 0 \ldots, 0)$ is committed in $e'_{1,0}, e'_{2,0}, \ldots, e'_{w,0}$ after being permuted. □

6.2 Efficiency Analysis

Suppose an inverse with modulus has the same cost of an exponentiation with the same modulus and e, the RSA public key in the oblivious transfer, is a small integer. In Table 1, the cost of bid validity check with normal method and batch solution respectively are compared. An example is given in the table for this comparison, where $w = 1024$, $|N| = |q| = 1024$ and $L = 20$. Suppose $ExpCost(x) = 1.5x$ and $ExpCost^y(x) = x + 0.5xy$.

The batch solution is much more efficient in computation, while the probability that an invalid bid is accepted is no more than $2^{-L} = 2^{-20}$. The larger w

Table 1
Comparison of Computation Efficiency

	prover		verifier	
	cost	example	verifier	example
normal	$w(3ExpCost(\|q\|)+2)$ $+ExpCost(\|q\|)+1$	4722177	$w(4ExpCost(\|q\|)+2)+$ $2ExpCost(\|q\|)+1$	6296577
batch	$(0.5\log_2 w+1)ExpCost(\|n\|)+$ $2ExpCost(\|q\|)+ExpCost^w(L)$ $+w+1.5\log_2 w+3$	23590	$2(ExpCost(\|n\|)+1)\log_2 w+$ $3ExpCost(\|q\|)+ExpCost^w(L)$ $+w\log_2 w+2$	55850

is, the greater advantage the batch solution has. In auction, the number of biddable prices must be large enough to avoid tie situation, especially when many bidders take part in the auction. So computational improvement by batch verification is very helpful in large-scale auctions.

7 Conclusion

Bid validity check is necessary in homomorphic auction schemes to prevent attack against correctness and fairness. However, bid validity check is usually costly. Batch verification technique and oblivious transfer can be combined to significantly improve efficiency of bid validity check in homomorphic auction schemes without compromising bid privacy.

Acknowledgements

We acknowledge the support of the Australian Research Council through ARC Discovery Grant No. DP0345458.

References

[1] Riza Aditya, Kun Peng, Colin Boyd, and Ed Dawson. Batch verification for equality of discrete logarithms and threshold decryptions. In *Second conference of Applied Cryptography and Network Security, ACNS 04*, volume 3089 of *Lecture Notes in Computer Science*, pages 494–508, Berlin, 2004. Springer-Verlag.

[2] M Bellare, J A Garay, and T Rabin. Fast batch verification for modular exponentiation and digital signatures. In *EUROCRYPT '98*, pages 236–250, Berlin, 1998. Springer-Verlag. Lecture Notes in Computer Science 1403.

[3] Colin Boyd and Chris Pavlovski. Attacking and repairing batch verification schemes. In *ASIACRYPT '00*, pages 58–71, Berlin, 2000. Springer-Verlag. Lecture Notes in Computer Science 1976.

[4] J. Cohen Benaloh. Secret sharing homomorphisms: keeping shares of a secret secret. In *CRYPTO '86*, pages 251–260, Berlin, 1986. Springer-Verlag. Lecture Notes in Computer Science Volume 263.

[5] R. Cramer, I. B. Damgård, and B. Schoenmakers. Proofs of partial knowledge and simplified design of witness hiding protocols. In *CRYPTO '94*, pages 174–187, Berlin, 1994. Springer-Verlag. Lecture Notes in Computer Science Volume 839.

[6] L Harn. Batch verifying multiple DSA-type digital signatures. In *Elecrronics Letters, 34,9*, pages 870–871, 1998.

[7] A. Juels and M. Szydlo. An two-server auction protocol. In *Proc. of Financial Cryptography*, pages 329–340, 2002.

[8] H Kikuchi, Michael Harkavy, and J D Tygar. Multi-round anonymous auction. In *Proceedings of the First IEEE Workshop on Dependable and Real-Time E-Commerce Systems*, pages 62–69, June 1998.

[9] Hiroaki Kikuchi. (m+1)st-price auction. In *The Fifth International Conference on Financial Cryptography 2001*, pages 291–298, Berlin, February 2001. Springer-Verlag. Lecture Notes in Computer Science Volume 2339.

[10] Hiroaki Kikuchi, Shinji Hotta, Kensuke Abe, and Shohachiro Nakanishi. Distributed auction servers resolving winner and winning bid without revealing privacy of bids. In *proc. of International Workshop on Next Generation Internet (NGITA2000), IEEE*, pages 307–312, July 2000.

[11] Moni Naor and Benny Pinkas. Efficient oblivious transfer protocols. In *Twelfth Annual Symposium on Discrete Algorithms, January 7-9, 2001, Washington, DC, USA. ACM/SIAM*, pages 448–457, 2001.

[12] Moni Naor, Benny Pinkas, and Reuben Sumner. Privacy perserving auctions and mechanism design. In *ACM Conference on Electronic Commerce 1999*, pages 129–139, 1999.

[13] Torben P. Pedersen. Distributed provers with applications to undeniable signatures. In *EUROCRYPT '91*, pages 221–242, Berlin, 1991. Springer-Verlag. Lecture Notes in Computer Science Volume 547.

[14] Torben P. Pedersen. Non-interactive and information-theoretic secure verifiable secret sharing. In *EUROCRYPT '91*, pages 129–140, Berlin, 1991. Springer-Verlag. Lecture Notes in Computer Science Volume 547.

[15] Kun Peng, Colin Boyd, Ed Dawson, and Kapali Viswanathan. Robust, privacy protecting and publicly verifiable sealed-bid auction. In *ICICS*, volume 2513 of *Lecture Notes in Computer Science*, pages 147 – 159. Springer, 2002.

[16] C Schnorr. Efficient signature generation by smart cards. *Journal of Cryptology, 4, 1991*, pages 161–174, 1991.

[17] Wen-Guey Tzeng. Efficient 1-out-n oblivious transfer schemes. In *Public Key Cryptography, 5th International Workshop on Practice and Theory in Public Key Cryptosystems, PKC 2002, Paris, France, February 12-14, 2002, Proceedings*, volume 2274 of *Lecture Notes in Computer Science*, pages 159–171. Springer, 2002.