Implementing a Secure Verifiable Auction

Mark Bun, Yi-Hsiu Chen, and Tom Morgan

May 14, 2015

1 Introduction

An online auction system provides a platform that allows users to exchange items. Such a platform raises a number of interesting security concerns that are in tension with each other. On one hand, each bidder wishes to keep her bid private so that other bidders do not gain an advantage over her in the present or subsequent auctions. On the other hand, bidders and regulating authorities want to be able to verify that the result of an auction is correct, even in the face of a cheating auctioneer. Thus we are interested in designing *sealed-bid auctions* that are *secure* and verifiable in that they achieve the following properties simultaneously.

- Correctness: The auction result is deterministic and strictly follows the auction rules.
- Bid Privacy: The confidentiality of the bids should be maintained during and after the auction. (This requirement implies the fairness of the auction. Namely one cannot change their bid after the submission.)
- Public Verifiability: Both the validity of the bids and the results of the auction can be verified by any third party.

Of course, with an eye toward making such an auction practical, we want to make both the auction and verification procedure computationally efficient.

Building on ideas of Parkes et al. [PRST08], Rabin, Servedio, and Thorpe [RST07] designed and implemented such a highly efficient auction protocol. The idea of the protocol is as follows. Each bidder determines his bid and publicly posts a commitment to this bid to a public bulletin board. After the auction has ended, the bidders open their commitments to the auctioneer over private channels. The auctioneer can then determine the results of the auction, and then engage in a zero-knowledge proof with a verifier to convince her of the correctness of the outcome. The zero-knowledge property of such a proof guarantees protects the privacy of the bids.

Using the work of [RST07] as a starting point, we make the following contributions.

1. Through more careful bookkeeping, we reduce the proof size and verification time of the protocol of [RST07] by constant factors.
2. We implement and test the protocol using our theoretical optimizations.
3. We extend the techniques of [RST07] to handle more general classes of auctions.

1.1 Related work

Prior to the work of [RST07], Parkes, Rabin, Shieber, and Thorpe [PRST08] gave a solution based on additively homomorphic encryption, and in particular implemented a secure verifiable auction using Pailler encryption. Their C++ implementation is somewhat practical (running in less than 24 hours) for about 100 users. The work of Rabin, Servedio, and Thorpe [RST07] gave a more efficient solution using ideas from secret sharing, which bypasses the need to use homomorphic encryption. More specifically (see Section 2 for details), they represent every number x in a field $\mathbb{F}_p$ as a random representation (u, v) such that $u + v = x$ in $\mathbb{F}_p$. This representation enables a "cut-and-choose" style interactive proof, where opening a commitment to either u or v enables verification while revealing nothing about x. The auction system of [RST07] offers about a 60-fold speed up, at the expense of having a much larger proof for verification (i.e. communication complexity).

Subsequently, Micali and Rabin [MR14] extended the techniques of [RST07] to prevent collusion in Vickrey auctions and to verify the correctness of solutions to matching problems (e.g. for medical resident matching). Similar random representation techniques were also used by Rivest and Rabin [RR] to implement verifiable electronic voting schemes.

2 The Protocol

We give a complete description of our verifiable auction protocol, which builds heavily on the protocol of [RST07]. In Section 2.7 we discuss the improvements we make over their protocol.

2.1 Model and Tools

Infrastructure. In our model, we assume there is a secure private channel between each player and the auctioneer. There is also a public bulletin board that every participant can post messages to. We assume that all parties (buyers, sellers, auctioneers, and verifiers) are users of a common public key infrastructure. In particular, each party has private digital signature key corresponding to a public verification key with a trustworthy link to that party's identity. Without mention in the following, we assume that whenever a user posts or sends a message, she will include a digital signature of that message to ensure its authenticity.

Commitment scheme. We assume a perfectly (i.e. information-theoretically) hiding and computationally binding commitment scheme COM.

Definition 1 (Commitment)**.** *A perfectly hiding and computationally binding commitment scheme is a function* $\mathrm{COM} : \mathbb{F}_p \times [0, m-1] \to R$ *where* $|R| = m$ *that is*

- *(Perfectly hiding) For any* $u \in \mathbb{F}_p$, $\mathrm{COM}(u, \cdot)$ *is a bijection (i.e.* $\mathrm{COM}(u, \cdot)$ *is one-to-one and* $\{\mathrm{COM}(u, r)\}_{r \in [0,m-1]} = R$*)*
- *(Computationally binding) It is infeasible for a polynomial time algorithm to find a pair* $(u_1, r_1), (u_2, r_2)$ *such that* $\mathrm{COM}(u_1, r_1) = \mathrm{COM}(u_2, r_2)$*.*

Random representation. To enable cut-and-choose proofs, we define a random representation for points $x \in \mathbb{F}_p$.

Definition 2 (Random representation)**.** *A random representation of x is a vector $X = (u, v)$ where* $u \overset{R}{\longleftarrow} \mathbb{F}_p$ *and* $v = x - u \bmod p$*.*

Note that for any x, the marginal random variables u and v are uniformly distributed on $\mathbb{F}_p$. Namely,

$$\forall a \in \mathbb{F}_p, \mathbb{P}[u = a \mid x] = \mathbb{P}[v = a \mid x] = \frac{1}{p}.$$

The commitment of a random representation is simply the commitment of both components. Namely, $\mathrm{COM}(X) = (\mathrm{COM}(u), \mathrm{COM}(v))$.

2.2 Overview

Before the auction closes, a buyer can post a commitment to her bid to the public bulletin board. By the hiding property of the commitment scheme, no other party can determine the contents of the bid. After the auction has concluded, she can then send the decommitment value to the auctioneer (through a secure channel). The binding property of the commitment scheme prevents her, or anyone else, from changing the underlying bid. After receiving every buyer's decommitment, the auctioneer can decide the winner(s) of the auction and announce the winner(s) and the winning bids.

Now if another party wants to check the correctness of the auction result, he can request to engage in a verification protocol with the auctioneer. The auctioneer first generates a "proof" that the result is correct. Next, the verifier sends a random challenge. The auctioneer responds to the challenge, and finally the verifier can check if the auctioneer resolved the auction correctly. To convince the verifier, we want the verification protocol to satisfy completeness and soundness properties. Moreover, to protect bidder privacy, we want it to be zero-knowledge.

To prove the correctness of an auction outcome, the auctioneer must supply zero-knowledge proofs that the committed values on the bulletin board compare correctly. While we only need to prove the correctness of comparisons, the techniques underlying such proofs require the ability to prove the correctness of additions and multiplications – making this proof system quite general. In the following 3 subsections, we present the protocols for addition, multiplication and comparison respectively. We will then show how to put these protocols together.

2.3 Addition

Addition proof generation. Suppose the random representations of $x_1, \cdots, x_k$ are $X_i = (u_i, v_i)$ for $i = 1, \cdots, k$. Assume these commitments $\mathrm{COM}(X_i)$s are posted on the public bulletin board. The auctioneer wishes to prove that $x_1 + \cdots + x_k = x$ without revealing anything else about the individual x_i's.

First, the auctioneer generates a random representation of x, $X = (u, v)$. He then posts $\mathrm{COM}(X)$ to the bulletin board. (If the verifier needs to know x as part of the correctness guarantee, the auctioneer can simply reveal $\mathrm{COM}(X)$. But if the verifier is not allowed to learn x, the auctioneer can still prove the equality). The auctioneer calculates t such that

$$(u_1, v_1) + (u_2, v_2) + \cdots + (u_k, v_k) = (u, v) + (t, -t) \tag{1}$$

and posts $\mathrm{COM}(t)$ on the public board as well.

Addition verification. Next, a verifier can randomly challenge the auctioneer to open either all of the u-components or all of the v-components. The auctioneer will then open the requested commitments and also reveal t. If the challenge was for the u-components, the verifier checks that

$$u_1 + u_2 + \cdots + u_k = u + t$$

Otherwise, she checks that

$$v_1 + v_2 + \cdots + v_k = v - t$$

Properties. The protocol has *perfect completeness* since an honest auctioneer can always find t such that equalities hold for both components in equation (1)). If he honestly opens the commitments, the verifier will accept in either challenge case. On the other hand, X is not correctly calculated and posted on the public board, then there is no t such that equation (1)) holds. Since the commitments are computationally binding, the verifier will discover the inconsistency with probability at least $1/2 - \mathrm{negl}$ against a computationally bounded auctioneer. Therefore, the protocol achieves *soundness* of 1/2. The last desired property is zero-knowledge. Intuitively, the protocol is zero-knowledge because by only seeing u_i (or v_i) component of a vector, the verifier learns nothing about x_i. (Perfect zero-knoweldge can be proved formally from the perfect hiding property of the commitment scheme.)

2.4 Multiplication

Multiplication proof generation. Suppose the random representations of x, y, z are $X = (u_x, v_x)$, $Y = (u_y, v_y)$ and $Z = (u_z, v_z)$. Commitments to all three representations are posted on the public board. The auctioneer wishes to show that $x \cdot y = z$.

The auctioneer prepares three more auxiliary random representation below, where r_1, r_2 are randomly chosen from $\mathbb{F}_p$.

$$\begin{cases} Z_0 &= (u_0, v_0) = (u_x u_y, v_x v_y) \\ Z_1 &= (u_1, v_1) = (u_x v_y + r_1, -r_1) \\ Z_2 &= (u_2, v_2) = (v_x u_y + r_2, -r_2) \end{cases}$$

As in the addition protocol, the auctioneer first calculates t such that

$$(u_z, v_z) = (u_0, v_0) + (u_1, v_1) + (u_2, v_2) + (t, -t)$$

and post the commitment COM(t) as well.

Multiplication verification. A verifier randomly challenges the auctioneer to open one of four following aspects.

1. All the u components and t.
2. All the v components and t.
3. u_x, v_y and both components of Z_1.
4. v_x, u_y and both components of Z_2.

The auctioneer then opens the corresponding aspects. Based on the challenge requested, the verifier checks the following.

1. $u_z = u_0 + u_1 + u_2 + t$ and $u_0 = u_x \cdot u_y$.
2. $v_z = v_0 + v_1 + v_2 - t$ and $v_0 = v_x \cdot v_y$.
3. $u_x v_y = u_1 + v_1$.
4. $v_x u_y = u_2 + v_2$.

Properties. As before, the *perfect completeness* of the protocol is trivial: if $x \cdot y = z$, honest prover can always convince the verifier. On the other hand, $x \cdot y \neq z$, we can see that no matter how the prover generates Z_0, Z_1, Z_2 and w, at least 1 of 4 aspects that checked by the verifier fails. Thus the protocol has 1/4-*soundness.*

For the *zero-knowledge* property, we first examine the first two aspects. As with addition, individual components of Z_0, X, Y reveal nothing about X, Y and Z. And for Z_1, Z_2, since the auctioneer added the new randomness r_1 and r_2, these are also just a uniformly random numbers from $\mathbb{F}_p$. Now we consider the third and fourth aspects. Here, the verifier only learns one of two components of X and Y, which also contains no information about x, y.

2.5 Comparison

In $\mathbb{F}_p$, if we know that $0 \leq x < p/2$, $0 \leq y < p/2$ and $0 \leq x - y < p/2$, then we can conclude that $x > y$. Therefore, we will focus on proving that $0 \leq x < p/2$.

First, by Lagrange's four-square theorem, we can represent x as $x = a^2 + b^2 + c^2 + d^2$. If we can show that $-q \leq a, b, c, d \leq 2q$ (In $\mathbb{F}_p$, this really means $0 \leq a \leq 2q$ or $p - q \leq a < p$) where $q = \sqrt{p/32}$, then $0 \leq x < 16q^2 = p/2$. Below we will show how to prove that $-q < a < 2q$ based on the fact that $0 \leq a < q$.

The auctioneer first randomly picks w_1, w_2 from $[-q, q)$ such that $|w_1 - w_2| = q$. Then it is easy to see that one of $w_1 + a$ and $w_2 + a$ lies in the range $[0, q)$. Let the one in range be $r = w_b + a$ where $b \in \{1, 2\}$. Now the auctioneer posts the random representations of all the above quantities, which are

$$W_1 = (u_1, v_1),\ W_2 = (u_2, v_2),\ A = (u_a, v_a),\ R = (u_r, v_r)$$

Verification of $-q \leq a < 2q$. First, the auctioneer posts t such that

$$(u_b, v_b) + (u_a, v_a) = (u_r, v_r) + (t, -t)$$

Then as in multiplication verification, the verifier will send a random challenge for opening one of the following four aspects.

1. All the u components and t.
2. All the v components and t.
3. Both W_1 and W_2.
4. R.

Then for each challenge, the verifier checks the following for the corresponding aspects.

1. $u_b + u_a = u_r + t$.
2. $v_b + v_a = v_r - t$.
3. $|w_1 - w_2| = q$ and they both lie in the range $[-q, q)$.
4. r is in the range $[0, q)$.

Properties. Under the condition that $a \in [0, q)$, the auctioneer can post items that validate all four aspects, so we again, we have *perfect completeness*. On the other hand, if in fact $a \notin [-q, 2q)$, then it is impossible for all four aspects to true. No matter what w_b auctioneer gives, either $w_b \notin [-q, q)$ or $r \notin [0, q)$. Thus we also have the 1/4-*soundness*. Finally, the *zero-knowledge* property is also obvious. As before, the first two aspects reveals nothing about a. And for aspect 3 and 4, the distribution of (w_1, w_2) and r are also independent of a (separately).

Being able to prove that $a \in [-q, -2q]$, together with the protocols for proving addition and multiplication, allows us to prove that $0 \leq x < p/2$. In the next subsection, we will see how we put things together.

2.6 Putting Things Together

In this section, we present the complete protocol for proving $0 \leq x < p/2$. (Suppose $X = (u_x, v_x)$ is on the board). As in the previous section, we assume that $x \in [0, q)$ where $q = \sqrt{p/32}$. We follow the convention that a capital letter is the random representation of the value of the corresponding lower-case letter.

2.6.1 Proof Generation

Values. First, use the algorithmic version of the four-square theorem to find a, b, c, d such that $x = a^2 + b^2 + c^2 + d^2$.

For a, we randomly pick $w_0^a, w_1^a \in [-q, q)$ such that $|w_0^a - w_1^a| = q$. We let $r_a = a + w_h^a$ where $h = 0$ or q such that $r_a \in [0, q)$. We do the same thing for b, c and d, yielding tuples (w_0^b, w_1^b, r_b), (w_0^c, w_1^c, r_c) and (w_0^d, w_1^d, r_d).

Random Representations. We need the following random representations as part of the proof.

- The random representations for a^2, b^2, c^2 and d^2.

$$A^2 = (u_a^2, v_a^2)\ , \quad B^2 = (u_b^2, v_b^2)\ , \quad C^2 = (u_c^2, v_c^2)\ , \quad D^2 = (u_d^2, v_d^2)$$

- Two random representations for each of a, b, c and d.

$$A = (u_a, v_a)\ , \quad B = (u_b, v_b)\ , \quad C = (u_c, v_c)\ , \quad D = (u_d, v_d)$$
$$A' = (u'_a, v'_a)\ , \quad B' = (u'_b, v'_b)\ , \quad C' = (u'_c, v'_c)\ , \quad D' = (u'_d, v'_d)$$

- The auxiliary random representation for the showing $AA' = A^2$ (and similarly for B, C, D).

$$Z_0^a = (u_0^a, v_0^a) = (u_a u'_a, v_a v'_a)\ , \quad Z_1^a = (u_1^a, v_1^a) = (u_a v'_a + r_1^a, -r_1^a)\ , \quad Z_2^a = (u_2^a, v_2^a) = (v_a u'_a + r_2^a, -r_2^a)$$

- Random representations for proving $a \in [-q, 2q]$ (and similarly for b, c, d).

$$W_0^a = (u_a^{w0}, v_a^{w0})\ , \quad W_1^a = (u_a^{w1}, v_a^{w1})\ , \quad R_a = (u_a^r, v_a^r)$$

“t values” for proving addition. The auctioneer also calculates random t values that satisfy following equations.

- t for $X = A^2 + B^2 + C^2 + D^2$:
$$(u_x, v_x) = (u_a^2, v_a^2) + (u_b^2, v_b^2) + (u_c^2, v_c^2) + (u_d^2, v_d^2) + (t, -t)$$
- t_a, t_b, t_c, t_d for $A = A'$ (and B, C, D)
$$(u_a, v_a) = (u'_a, v'_a) + (t_a, -t_a)$$
- $t_a^2, t_b^2, t_c^2, t_d^2$ for showing $A^2 = Z_0^a + Z_1^a + Z_2^a$ (and B, C, D).
$$(u_a^2, v_a^2) = (u_0^a, v_0^a) + (u_1^a, v_1^a) + (u_2^a, v_2^a) + (t_a^2, -t_a^2)$$
- $t_a^w, t_b^w, t_c^w, t_d^w$ for $R_a = W_h^a + A$ (and B, C, D.
$$R_a = W_h^a + A + (t_a^w, -t_a^w)$$

All the above representations and t values are committed then posted on the public board.

2.6.2 Verification

The verifier sends a random challenge to ask the auctioneer to open one of four following commitments.

1. All the u-components and t values.
2. All the v-components and t values.
3. W_0^a, W_1^a and $Z_2^a, Z_2^b, Z_2^c, Z_2^d$.
4. R^a, R^b, R^c, R^d and $Z_2^a, Z_2^b, Z_2^c, Z_2^d$.

Then the verifier can check the corresponding aspects

1. Check
$$\begin{aligned} u_x &= u_a^2 + u_b^2 + u_c^2 + u_d^2 + t \\ u_a &= u'_a + t_a \\ u_a^2 &= u_0^a + u_1^a + u_2^a + t_a^2 \\ u_a^2 &= u_a \cdot u'_a \\ u_a^r &= u_a^{wh} + t_a^w \end{aligned}$$
2. As in previous aspect, but replace all u's by v's and t by $-t$.
3. Check $u_a v'_a = u_1^a + v_1^a$, $|w_0^a - w_1^a| = q$ and $w_0^a, w_1^a \in [-q, q)$ (Also for b, c, d)
4. Check $v_a u'_a = u_2^a + v_2^a$ and $r_a \in [-q, q]$ (Also for b, c, d)

2.7 Our Improvements

The protocol we just described makes two improvements in comparison to [RST07].

Reducing the number of aspects. The protocol of [RST07] creates 16 aspects for comparison, so they only achieve 1/16-soundness in each round. In Section 2.6, we adopt the idea from [RMMY12] of wrapping the verification into only 4 aspects while maintain the zero-knowledge. (Note that the description given by [RST07] does not specify a wrapping). Therefore, to achieve the same level of soundness, we need $\log(15/16)/\log(3/4) \approx 4.45$ times fewer proofs as in [RST07].

Reducing the commitments. In order to prove $x < p/2$, the authors of [RST07] count a cost of 101 commitments. However, the protocol we just described above only uses 87 commitments. In our implementation, we use 91 commitments to make the implementation more concise and structured, but we still get about a 10% improvement in all time and space complexities.

3 Extension to Double Auctions

In a double auction there are m sellers each trying to sell an item, and n buyers each trying to buy an item. All items are indistinugishable. The ith seller submits a price s_i to the auctioneer which is the least they would be willing to accept for their item, and the jth buyer submits a price b_j to the auctioneer which is the most they would be willing to pay for the item. The auctioneer takes all of the submitted prices and by some mechanism arrives at a price p for item to be exchanged at.

One simple mechanism that is commonly used is the average mechanism, which operates as follows. First, the auctioneer orders the prices from the buyers and sellers in increasing order of constraint. For the buyers, this means decreasing order of price so that $b_1 \geq b_2 \geq \ldots \geq b_n$. For the sellers, this means increasing order of price so that $s_1 \leq s_2 \leq \ldots s_m$. The auctioneer then finds the largets index $k \in \{1, \ldots, \min(n, m)\}$ such that $b_k \geq s_k$. This is the largest k for which there exists a price satisfying both the kth buyer and the kth seller. The auctioneer then sets the price at $p = (b_k + s_k)/2$.

We will use the comparison proof primitive that we discussed in Section 2.5 to prove the correctness of a double auction using the average mechanism. The auctioneer will publicly release the price p, the identities of the participants who submitted prices b_k and s_k, and the identities of those whose prices were in $\{b_1, \ldots, b_{k-1}\}$ and $\{s_1, \ldots, s_{k-1}\}$. Note that for those sets of participants, the ordering is not revealed, just the set membership, and none of the participants have their submitted prices revealed.

In order to prove the correctness of the output, we most prove the following conditions:

1. $p = (b_k + s_k)/2$.
2. b_k is the kth largest buyer price.
3. s_k is the kth smallest seller price.
4. $b_k \geq s_k$.
5. $b_{k+1} < s_{k+1}$.

Condition 1 is easily proven with an addition proof, and Condition 4 is easily proven with a comparision proof. We prove Condition 2 using $n - 1$ comparison proofs, by proving $b_i \geq b_k$ for each $b_i \in \{b_1, \ldots, b_{k-1}\}$ and $b_i \leq b_k$, and then proving that $b_i \leq b_k$ for each $b_i \notin \{b_k\} \cup \{b_1, \ldots, b_{k-1}\}$. Condition 3 is analagously proven with $m - 1$ comparison proofs.

We could easily prove Condition 5 with a single comparison proof if the identities of the participants who submitted prices b_{k+1} and s_{k+1} were revealed, however do not need to. Instead we do a little more work. It is equivalent to prove that $\max\{b_{k+1}, \ldots, b_n\} < \min\{s_{k+1}, \ldots, s_m\}$. To prove this, the auctioneer sets $w = s_{k+1} = \min\{s_{k+1}, \ldots, s_m\}$, and commits to a pair representation of w (without revealing which participant it came from). The auctioneer then uses comparison proofs to prove that $b_i < w$ for all $b_i \notin \{b_k\} \cup \{b_1, \ldots, b_{k-1}\}$, and $s_i \geq w$ for all $s_i \notin \{s_k\} \cup \{s_1, \ldots, s_{k-1}\}$. This gives us that $\max\{b_{k+1}, \ldots, b_n\} < w \leq \min\{s_{k+1}, \ldots, s_m\}$, implying Condition 5.

In total we proved the result using one addition proof, and $2(n + m - k) - 1 \leq 2n + 2m - 1$ comparison proofs.

4 Implementation

We implemented an end-to-end secure verifiable auction system according to the protocol described in Section 2. Our implementation was done in C++ using the NTL number theory library[1] built on top of the GNU

[1]`http://www.shoup.net/ntl/`

Multiple Precision Arithmetic library.[2] Our implementation code is attached as a zip file. In the rest of this section, we will overview the design of implementation, and discuss the algorithms we used to implement the different pieces of the protocol.

4.1 Design

Our implementation to run on a single machine, and does not have any networking infrastructure. However, a key design principle we adhered to was to make our implementation sufficiently modular and "honest" in its application of the protocol so that a network layer could easily be added to it if desired. By this we mean that we have a different class for each participant in an auction (auctioneer, buyer, seller and verifier) and each class only receives data as input into its functions that the corresponding participant in the protocol would be allowed to see.

Here is a sketch of an example run of our implementation:

1. We initialize an instance of a `PublicBoard` (the public bulletin board), an `Auctioneer`, a `Seller` and n instances of `Buyer`.
2. The `Seller` asks the `Auctioneer` to start an auction for a given `Article`, in response to which the `Auctioneer` creates a new `Auction` and posts its `PublicAuctionInfo` to the `PublicBoard`.
3. Each of the n `Buyer`s selects a bid and creates a BidPackage which it sends to the `Auctioneer`. A `BidPackage` is created by making a series of `PairReps` (pair representations) of the bid, and then creating a pair of `SignedCommitment`s for each one. In response to each `BidPackage`, the `Auctioneer` verifies that all commitments are consistent and the signatures are valid before and adding it to the `Auction`, and adding the public information from it as a `BidInfo` object to the `PublicAuctionInfo`.
4. The `Auctioneer` resolves the auction, by computing the winner and updating `Auction` and `PublicAuctionInfo` accordingly.
5. A `Player` (which is a super-class of `Buyer` and `Seller`) requests a proof of the correctness of the auction from the `Auctioneer`. The `Auctioneer` responds by construction a series of `ComparisonProof` objects (which consist of `Commitment`s and PairReps) in accordance with the protocol. The `Auctioneer` extracts just the commitment outputs from these proofs in `ComparisonProofCommitments` objects, and returns the set of these to the `Player`.
6. The `Player` picks a random number in $\{1, 2, 3, 4\}$ for each translation of the proof, and sends these numbers to the `Auctioneer` (each corresponding to a challenge from Section 2.6.2) to challenge the provided proof. In response to these, the `Auctioneer` opens the corresponding commitments in the proof and sends back a series of `Challenge` objects, each containing a set of `Commitment` objects from the earlier `ComparisonProof` objects. The `Player` then iterates through all of these `Challenge` objects, as well as the signed commitments in `BidInfo` to check that they accord with the protocol in Section 2.6.2.

4.2 Cryptographic Functions

The primary cryptographic tool used in the protocol described in Section 2 is the commitment scheme. In particular, the vast majority of communication done between prover and verifier is taken up by commitments. The commitment scheme we chose for our implementation was Olivier Gay's implementation of the SHA-256 cryptographic hash function.[3] To commit to a value x, we randomly create a 5 byte (40 bit) string r, concatenate r with a string representation of x (padded as needed to a fixed length), and then apply SHA-256 to the result. The output of the hash function is the commitment string that is broadcast, and the commitment is opened by revealing x and r.

[2] `https://gmplib.org/`
[3] `http://www.ouah.org/ogay/sha2/`

The only other cryptographic primitive required by the protocol is a digital signature scheme. For this, we used our own implementation of the Digital Signature Algorithm.[4] Our implementation uses DSA key lengths (L, N) of 2048 bits and 224 bits respectively.

4.3 Four-Square Theorem and the Algorithm

As we have seen, the algorithm for four square theorem plays an important role in proving the comparison result. Rabin and Shallit [RS86] proposed three randomized algorithms to represent a number into four squares. The fastest one runs in expected time $O(\log^2 N)$ where N is the number. This algorithm is still the state of the art and is which we implemented. Although the correctness relies on the extended Riemann hypothesis. Without any hypothesis, the best known algorithm runs in expected time $O(\log^2 N \log\log N)$.

Here we sketch the algorithm we used in the protocol. First, we need an algorithm for representing a prime number $p = 4k + 1$ as sum of two squares.

Two square algorithm: Find a, b such that $a^2 + b^2 = p$ where $p \equiv 1 \bmod 4$

1. Find the solution of the equation $x^2 + 1 = 0$ in $\mathbb{Z}_p$ in the following way.
   (a) Randomly guess $b \in \mathbb{Z}_p$, let $f_b(x) = x^2 - 2bx + (b^2 + 1)$.
   (b) Find the greatest common divisor $(f_b(x), x^{2k} - 1)$. With probability 1/2 (over the choice of b), the greatest common divisor is $x - u - b$ where u is the root of $x^2 + 1 = 0 \pmod p$.
2. Once we get $u^2 + 1 = mp$ for some m. Compute the greatest common divisor in Gaussian integer $x + yi = (u + i, p)$, then $x^2 + y^2 = p$.

Once we have the algorithm for finding two squares, we have the following four-square algorithm for any odd number n

Four square algorithm: For given odd n, find a, b, c, d such that $a^2 + b^2 + c^2 + d^2 = n$.

1. Randomly pick m which is less than n^3 and $m \equiv 2 \bmod 4$.
2. Let $q = mn - 1$. If q is not a prime, then repick m in step 1.
3. Once q is a prime (and $q \equiv 1 \bmod 4$), use the tow square algorithm to find u, v such that $u^2 + v^2 = q$. Basically, we have found $u^2 + v^2 + 1^2 + 0^2 = mn$ for some m.
4. W.L.O.G, let $-n/2 < u, v < n/2$ Calculate the greatest common divisor in "integral quaternion". That is $a + bi + cj + dk = (u + vi + j + 0k, n)$, then $a^2 + b^2 + c^2 + d^2 = n$.

When we express the solution in integral quaternion. If we have solutions for n_1, n_2, we can get the solution of $n_1 n_2$ by multiplying the solutions in integral quaternion ring. It's trivial to get the solution for $n = 2^k$. Therefore, by knowing how to find the representation of odd numbers, we know how to find a solution for any natural number.

Remark 1. *Under the settings that the price limit is 1 million. 60% of the proof preparing time is from the four square theorem algorithm. And if we raise the price limit, the portion of running four square theorem algorithm increases. That is because the running time of four square theorem is quadratic to the bit length, while other operations (mainly the commitments) are linear. Therefore, the four square theorem algorithms is the most significant bottleneck in the protocol.*

[4] `http://nvlpubs.nist.gov/nistpubs/FIPS/NIST.FIPS.186-4.pdf`

5 Experimental Results

There are two kinds of time complexity we care about: the proof generation time and the verification time. Meanwhile, the communication complexity is dominated by the size of the proof. In each experiment, we will measure these three kinds of complexity.

The parameters we are interested in that will affect the complexity time are (1) The number of buyers and (2) The price limit.

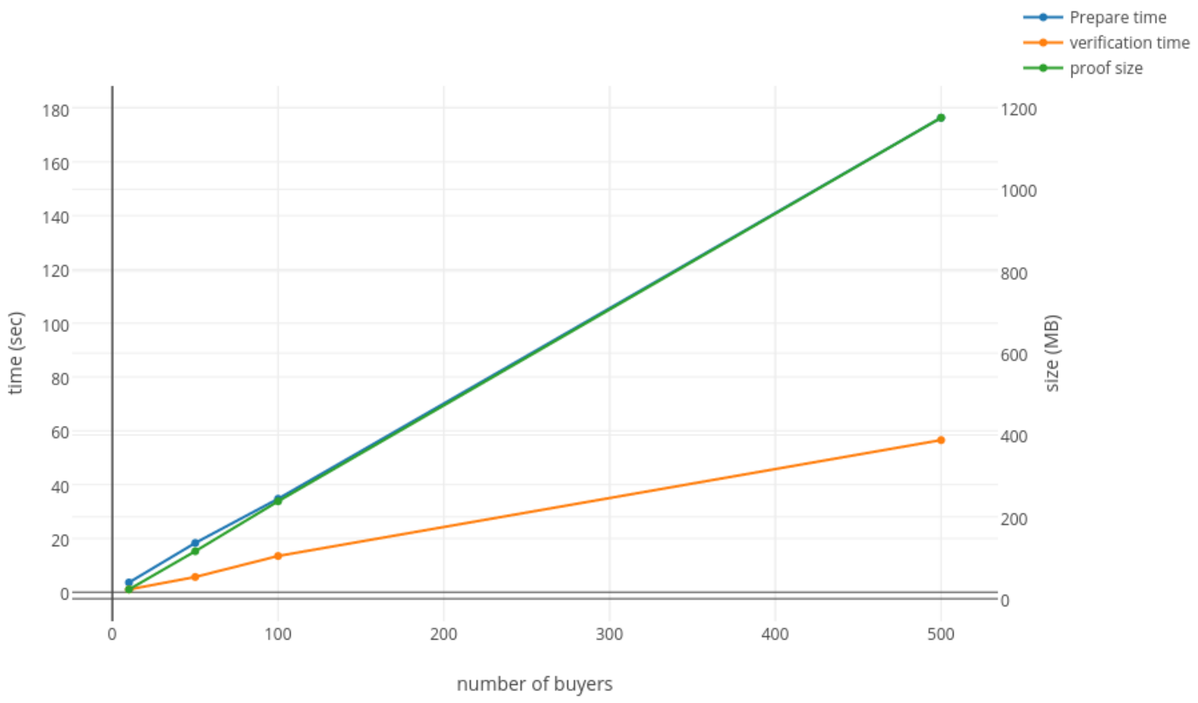
The following are the results we found by modifying those parameters. We set the soundness to be less than 0.03 meaning we repeated the proof 100 times. Note that we can reduce the soundness error to 10^{-12} by repeat 10 multiple times, which is 1000 times. Then all the time complexity and space complexity simply scales up by a factor of 10. All the tests were run on a machine with a 1.4 GHz Intel Core i5 processor.

5.1 Results

Performance If we set the price limit at about 1 million, and we want the soundness to be 10^{-12}. Under the setting of 100 buyers then

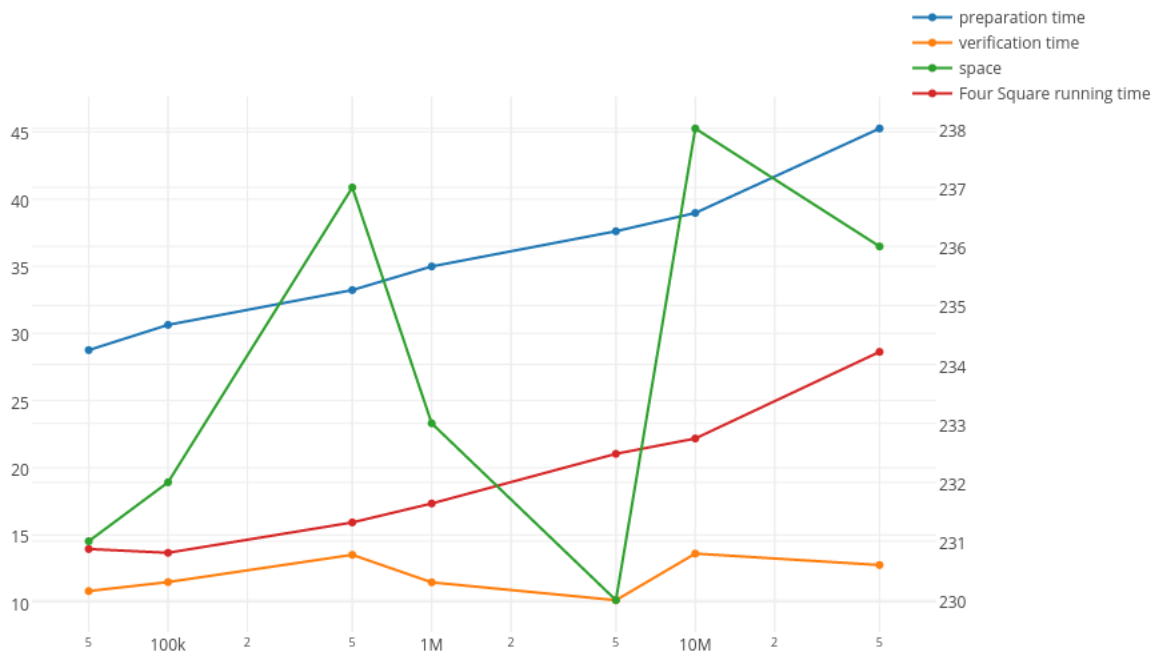
- Proof preparation time: 347 secs
- Verification time: 135 secs
- Proof size: 2.38 GB

The number of buyers



As expected, The running time is basically linear in the number of the buyers.

The Price Limit



The verification time and the proof size does not really grow with the size of the price limit. This is because raising the price limit does not really change the size of commitment. However, increasing the price limit does increase the proof preparation time. Moreover, we can see that if we subtract the running time of four square algorithm, the running time is like a constant, which is also as expected. The four square running time grows quadratically as in the analysis of the algorithm.

6 The Paillier-Based Scheme of Parkes et al.

For the sake of experimental comparison, we also implemented the verifiable auction of Parkes et al. [PRST08]. Before describing the protocol, we recall Paillier's additively homomorphic encryption scheme. Paillier's scheme is a public key encryption scheme with encryption key $n = p \cdot q$ for large (e.g. 1024 or 2048-bit) primes p and q. The secret decryption key is given by $\phi = \varphi(n) = (p-1)(q-1)$. To encrypt a message $x \in \mathbb{Z}_n$, we compute a random help value $r \in \mathbb{Z}_n^*$ and evaluate $E(x;r) = (1+n)^x r^n \pmod{n}^2$. It is easy to see that Paillier encryption enables the addition of encrypted values as $E(x,y) = E(x)E(y)$.

We now give a brief description of the verifiable auction protocol of [PRST08]. The bidding process is as described in Section 2. Each bidder posts a signed commitment to her bid to a public bulletin board, and at the conclusion of the auction, reveals her bid to the auctioneer. Commitments are done using Paillier encryption for keys generated once and for all by the auctioneer: the commitment to a value x is $E(x;r)$ for a random opening r.

What remains is to describe how the auctioneer can prove the correctness of comparisons between the encrypted bids posted to the bulletin board. Recall that it suffices for the auctioneer to be able to prove that a commitment contains a value x with $0 < x < 2^t < n/2$ for some parameter t. To do this, the auctioneer generates a "test set" $T = \{E(u_1; s_1), \ldots, E(u_{2t}; s_{2t})\}$ containing encryptions of t copies of 0 and each of the t powers of 2: $1, 2, \ldots, 2^{t-1}$. To prove that $0 < x < 2^t$, the prover writes x as a sum $2^{t_1} + \cdots + 2^{t_\ell}$ of distinct powers of 2, and selects from T the corresponding set of encryptions $G_{j_1}, \ldots, G_{j_\ell}$. Moreover, it selects $t - \ell$ encryptions of 0: $G_{j_{\ell+1}}, \ldots, G_t$. By the additive homomorphism of Paillier encryption, we have that $E(x;r)^{-1} \cdot G_{j_1} \cdot \cdots \cdot G_{j_t} \equiv E(0;s) \pmod{n}^2$ where $s = r^{-1} \cdot s_{j_1} \cdot \cdots \cdot s_{j_t} \pmod{n}$. Thus, the auctioneer can reveal $G_{j_1}, \ldots, G_{j_t}$ and the help value s to the verifier to convince her that x is in range. Moreover, the protocol reveals nothing about which powers of t or how many copies of 0 are included in the revealed sets.

A few additional details need to be addressed, including how the verifier can check that the test set T is valid. For brevity, we omit these details, but they can be found in [PRST08]. Experimental results are below for 100 bidders, a price limit of 10^6, and soundness 10^{-12} with 1024 and 2048-bit key sizes: These

Figure 1: Complexity of the Paillier-Based Scheme

	1024-bit	2048-bit
Proof generation	140 min	2.64 min
Proof verification	990 min	17.7 min
Proof size	4.33 MB	8.66 MB

results corroborate those of [RST07]: the scheme based on homomorphic encryption enables much smaller proof sizes at the expense of much longer verification times.

7 Future Directions

One direction we are interested in is whether we can further decrease the amount of information that is revealed about the participants in an auction. For example, in the case of a double auction we reveal exactly the identities of the buyer and seller who bid b_k and s_k respectively. Naively this may seem necessary in order to prove the correctness of the price (and it is consistent with what is done for second price auctions in [RST07]) however it may be possible to avoid this. We would presumably commit to values $x = b_k$ and $y = s_k$, prove everything as before in our protocol except using these values instead, and then somehow prove that there exists some bid (without revealing which one) that equals x and there is some bid (without revealing which one) that equals y. One way to do this may be to select commit to a random permutation of random representations of the bids, and through one set of challenges prove that it is a valid permutation, and through another set of challenges prove that one of the items in the permutation is equal to our target.

References

[MR14] Silvio Micali and Michael O. Rabin. Cryptography miracles, secure auctions, matching problem verification. *Commun. ACM*, 57(2):85–93, February 2014.

[PRST08] David C. Parkes, Michael O. Rabin, Stuart M. Shieber, and Christopher Thorpe. Practical secrecy-preserving, verifiably correct and trustworthy auctions. *Electronic Commerce Research and Applications*, 7(3):294–312, 2008.

[RMMY12] Michael O Rabin, Yishay Mansour, S Muthukrishnan, and Moti Yung. Strictly-black-box zero-knowledge and efficient validation of financial transactions. In *Automata, Languages, and Programming*, pages 738–749. Springer, 2012.

[RR] Michael O. Rabin and Ronald L. Rivest. Efficient end to end verifiable electronic voting employing split value representations. To appear in Proc. EVOTE 2014 (Bregenz, Austria).

[RS86] Michael O Rabin and Jeffery O Shallit. Randomized algorithms in number theory. *Communications on Pure and Applied Mathematics*, 39(S1):S239–S256, 1986.

[RST07] Michael O. Rabin, Rocco A. Servedio, and Christopher Thorpe. Highly efficient secrecy-preserving proofs of correctness of computations and applications. In *22nd IEEE Symposium on Logic in Computer Science (LICS 2007), 10-12 July 2007, Wroclaw, Poland, Proceedings*, pages 63–76. IEEE Computer Society, 2007.