Non-interactive Private Auctions

Olivier Baudron and Jacques Stern

École Normale Supérieure, LIENS
45, rue d’Ulm
F-75230 Paris Cedex 05, France

Abstract. We describe a new auction protocol that enjoys the following properties: the biddings are submitted non-interactively and no information beyond the result is disclosed. The protocol is efficient for a logarithmic number of players. Our solution uses a semi-trusted third party $T$ who learns no information provided that he does not collude with any participant. The robustness against active cheating players is achieved through an extra mechanism for fair encryption of a bit which is of independent interest. The scheme is based on homomorphic encryption but differs from general techniques of secure circuit evaluation by taking into account the level of each gate and allowing efficient computation of unbounded logical gates. In a scenario with a small numbers of players, we believe that our work may be of practical significance, especially for electronic transactions.

Keywords: auctions, bidding, homomorphic encryption, secure circuit evaluation.

1 Introduction

In web electronic commerce, the question of auctions has become a major issue. They offer a very flexible way to exchange goods while minimizing negotiation costs, and as expected, a variety of software architecture have been discussed [1,15]. Additionally, it is desirable to ensure privacy of each customer through cryptographic mechanisms. Ideally, at the end of the protocol, no information on the submitted bids should be disclosed. Depending on the auction settings, only the winner and the highest (or 2nd highest) bid should be revealed. So far, several approaches have been considered. Based on multi-party computation [2,6] and secret sharing [22], Harkavy, Tygar and Kikuchi [14] have described a distributed protocol, that ensures privacy but needs several rounds of interaction between the auctionners. In a novel direction, Cadin has proposed a non-interactive protocol [3] based on the so-called $\Phi$-hiding assumption that allows to secretly compare two numbers. However bidders have to interact in a direct manner and, furthermore, for a number of users greater than 2, it is necessary to consider two non-colluding third parties and partial order of bids is leaked to one of them. Finally, a more promising and efficient technique using two third parties has been introduced by Naor, Pinkas and Sumner [19]; it uses pseudo-randomness and oblivious transfer to securely compute arbitrary circuits.

Our solution uses a different approach which is built on a new one round secure circuit evaluation [23, 10, 9, 11] tailored for our specific problem. Although, it is more efficient than general techniques, it is limited to a logarithmic number of players. In practice, 5 or 6 participants keep the amount of network traffic at a reasonable level. We also require a semi-trusted third party $T$ (the server), who learns no information provided that he does not collude with any participant. To achieve robustness against active cheating players, the hardness of deciding composite residuosity classes is assumed. We point out that no
interaction between the bidders is required, which is a main achievement of our work. A high level description of the protocol is as follows.

1. Registration. Bidders who wish to participate publish their public encryption key.
2. Submission. Each bidder encrypts the figure of his choice under all participant’s public keys and sends the result to the server using a secure communication channel.
3. Results. The server publishes, in a encrypted way, whether each participant is the winner.
4. The winner reveals himself by proving to the server that he has actually won.
5. The server sends to each subscriber (or to the winner only) an encryption of the highest (or 2nd highest) bid.

The core of the problem is to decide whether a given participant has submitted the highest bid. This is accomplished in the next section. In section 3, we extend the submission scheme to withstand cheating players. In section 4, we propose some solutions to deal with a larger number of players. The conclusion comes in section 5.

2 Computing over encrypted bids

2.1 Preliminaries

We consider a protocol with $p$ participants, who submit $\ell$-bit numbers representing their bids. A probabilistic encryption scheme $E$ satisfying the following properties is fixed:

- The set of plaintext messages $\mathcal{M}$ is a group of order $N$ such that $1/N$ is a negligible function of the security parameter $k$. In the sequel we will use an additive notation.
- $E$ is self-randomizable: there exists a probabilistic polynomial time function $\mathcal{R}$ such that for any $m \in \mathcal{M}$, $\mathcal{R}(E(m))$ is uniformly distributed over the sets of encryptions of $m$.
- $E$ is homomorphic: for any $m_1, m_2 \in \mathcal{M}$, $E(m_1 + m_2) = E(m_1) \cdot E(m_2)$.
- $E$ is semantically secure against a chosen plaintext attack: no probabilistic polynomial time adversary can distinguish, with a non-negligible success, between encryptions of two plaintexts of its choice [12].
- There exists a full decryption algorithm $D$: for any pair of public key and secret key $(pk, sk)$, for any encryption $c$ of $m$ under $pk$, $D_{sk}(c)$ outputs $(m, r)$ such that $E_{pk}(m, r) = c$.

Known efficient schemes meeting these requirements are: Naccache-Stern [16], Okamoto-Uchiyama [17] and Paillier [18]. The latter will be used for the robust version of our protocol together with proofs of membership, so it should be given more attention.

Each participant encrypts $p$ times each bit of his bid using the $p$ candidates’ public key (including his own one). The output, of length $\ell \times p$ times the length of an encryption, is sent to the server through a private channel. The ultimate goal of this section is for the server, to compute, for each integer $i$ in $\{1, \ldots, n\}$ the predicate: $(P_i) : a_i \geq \max(a_1, \ldots, a_p)$ where the $a_j = (a_j^{\ell-1} a_j^{\ell-2} \cdots a_j^0)_2$ are the binary representations of the biddings. To perform the comparison of two numbers using logical bit operations, we observe that $a_i > a_j$ if and only if there exists an index $s$ in $\{0, \ldots, \ell - 1\}$ such that the following predicate is satisfied

$$Q_{i,s} : \bigwedge_{m=\ell-s}^{\ell-1} (a_i^m \iff a_j^m) \land a_i^{\ell-s-1} \land (-a_j^{\ell-s-1})$$

(1)
Namely, the first $s$ bits of $a_i$ match the first $s$ bits of $a_j$ and the $(s+1)^{th}$ bit of $a_i$ is greater than the $(s+1)^{th}$ bit of $a_j$. Observe that the predicate deciding the equality of $a_i$ and $a_j$ is given by $(Q_{i,j}) : \wedge_{m=0}^{t-1} (a_i^m \neq a_j^m)$. In the next stage, the existential quantifier is evaluated by OR-ing over the various boolean formulae. Finally, to decide whether a number $a_i$ is the maximum of a set of $p$ numbers $(a_1, ..., a_p)$, it remains to compute a logical AND of the $p-1$ subexpressions comparing $a_i$ with all others $a_j$. Consequently, the circuits representing the predicates $(P_i)$, using unbounded AND and OR gates, are given by

$$ (P_i) : \bigwedge_{j=1, j \neq i}^{t} Q_{i,j} $$

Considerable efforts have been made to provide general protocols that enable a third party to blindly compute each logical gate of a circuit with the help of the secret inputs' owners. However, efficient protocols require a number of interaction rounds linear in the depth of the circuit. As told in the introduction, it is essential from a practical viewpoint to perform the whole circuit evaluation non-interactively. Recently, Sander, Young and Yung showed how to compute in a single round any $NC^1$ circuit over encrypted data [21]. They recursively define structures allowing the computation of logical gates. However, it must be pointed out that an OR-gate inflates the length of the input datas by a factor of 8, and the same holds for AND-gates. Thus, considering our initial circuit of the max function, the algorithm would produce a string of length $\Theta(L \log L + \log (p-1)) = \Theta(L^2 (p-1)^3)$ encryptions.

Our solution differs from Sander et al. by applying different rules to a given gate according to its level in the circuit. Also, the use of a message space of order $N$ enables us to build an efficient method for computing unbounded gates directly, rather than considering the equivalent binary sub-circuit. Against a curious but honest server, the privacy of the submitted inputs is ensured throughout the semantic security of the encryption scheme $E$. Similarly, privacy towards curious participants is guaranteed provided the encryption of the result is independent of the posted data. This is achieved throughout the self-randomization of $E$.

### 2.2 Efficient computation of the max function

We now precisely describe our specific solution to compute the various predicates $P_i$. The security parameter $k$ is fixed. We denote by $\mathcal{C}$ the space of ciphertexts: $\mathcal{C} = \mathcal{E}(\mathcal{M})$. We define $(\text{Enc}_0^t)_{t \in \mathbb{N} \setminus \{0\}}$ and $(\text{Enc}_1^t)_{t \in \mathbb{N} \setminus \{0\}}$ two family of sets representing encryptions of bit 0 and bit 1 respectively. For each $t \in \mathbb{N} \setminus \{0\}$, $\text{Enc}_0^t$ is the set of $t$-coordinates vectors in $\mathcal{C}^t$ such that the decryption of any coordinate is non zero, and $\text{Enc}_1^t$ is the set of $t$-coordinates vectors in $\mathcal{C}^t$ such that there exists exactly one coordinate which encrypts zero. We also define $\text{Enc}_i$ to be the set $\text{Enc}_0^t \cup \text{Enc}_1^t$. In symbols:

$$ \text{Enc}_0^t = \{(c_1, \ldots, c_t) \in \mathcal{C}^t \mid \forall i \in \{1, \ldots, t\} D(c_i) \neq 0 \} $$

$$ \text{Enc}_1^t = \{(c_1, \ldots, c_t) \in \mathcal{C}^t \mid \exists i \in \{1, \ldots, t\} D(c_i) = 0 \} $$

Each logical gate $G$ takes as input elements from $\text{Enc}_{f(G)}$ and outputs an element in $\text{Enc}_{g(G)}$, where $f$ and $g$ are positive integer functions which only depend on the type and the level of the given gate $G$. Namely, $f(G)$ is the length of the inputs of $G$ and $g(G)$ is the length of its output, both in number of encryptions. The server propagates the cipher bits in the circuit by the following algorithm:

- Inputs: cipher bits are elements of the sets:
\[ \text{In}^0 = \{ c \in C \mid D(c) = 1 \} \subseteq \text{Enc}_i^0 \]

and
\[ \text{In}^1 = \{ c \in C \mid D(c) = 0 \} = \text{Enc}_i^1 \]

We note \( \text{In} = \text{In}^0 \cup \text{In}^1 \).

- Level 1, \( \neg \) gates: \( f = 1, g = 1 \)
  \( \text{NOT}_1 : \text{In} \rightarrow \text{Enc}_1 \)
  \[ E(x) \hookrightarrow \mathcal{R}([E(x)/E(1)]^r) = \mathcal{R}(E(r(x - 1))) \]
  where \( r \) is uniformly drawn in \( \mathbb{Z}_N \setminus \{0\} \).

- Level 1, \( \leftrightarrow \) gates: \( f = 1, g = 1 \)
  \( \text{EQUIV}_1 : \text{In} \times \text{In} \rightarrow \text{Enc}_1 \)
  \[ (E(x), E(y)) \hookrightarrow \mathcal{R}([E(x)/E(y)]^r) = \mathcal{R}(E(r(x - y))) \]
  where \( r \) is uniformly drawn in \( \mathbb{Z}_N \setminus \{0\} \).

- Level 2, \( \land \) gates: \( f = 1, g = 1 \)
  \( \text{AND}_2 : (\text{Enc}_1)^k \rightarrow \text{Enc}_1 \)
  \[ (E(x_1), \ldots, E(x_k)) \hookrightarrow \mathcal{R}([\prod_{i=1}^k E(x_i)]^r) = \mathcal{R}(E(r \sum_{i=1}^k x_i)) \]
  where \( r \) is uniformly drawn in \( \mathbb{Z}_N \setminus \{0\} \).

- Level 3, \( \lor \) gates: \( f = 1, g = \ell \)
  \( \text{OR}_3 : (\text{Enc}_1)^\ell \rightarrow \text{Enc}_\ell \)
  \[ (c_1, \ldots, c_\ell) \hookrightarrow (c_{\sigma(1)}, \ldots, c_{\sigma(\ell)}) \]
  where \( \sigma \) is a random permutation of \( \ell \) elements.

- Level 4, \( \land \) gate: \( f = \ell, g = \ell^{p-1} \)
  \( \text{AND}_4 : (\text{Enc}_1)^{p-1} \rightarrow \text{Enc}_{\ell^{p-1}} \)
  \[ (E(x_1), \ldots, E(x_{\ell^{p-1}}))_{1 \leq i \leq \ell-1} \hookrightarrow \left( \mathcal{R}(\prod_{i=1}^{\ell^{p-1}} E(x_{j_i}^i)) \right)_{(j_1, \ldots, j_i) \in [1, \ell-1]^i} \]
  coordinates of the final vector are randomly permuted.

The final result is a string of \( \Theta(\ell^{p-1}) \) encryptions. Although it is asymptotically exponential in the number of participants, it is better than what can be achieved by general techniques for a limited number of players. For example, considering 32-bit precision of bids and 4 participants, our scheme leads to strings of length \( 2^{15} \) whereas [21] would produce strings of length \( 2^{512} \). Now, we prove that our computation is correct and leaks no information on the inputs. First the following lemma results from the particular structure of the boolean circuit.

**Lemma 1.** For any \( s \) and \( s' \) such that \( s \neq s' \), the predicates \( Q_{i,s} \) and \( Q_{i,s'} \) are mutually exclusive.

**Proof.** Without loss of generality, assume that \( s < s' \). We focus on the particular bit position \( r = \ell - s - 1 \) in the integers \( a_i \) and \( a_j \). As \( s \leq \ell \), it follows that \( Q_{i,s} \) is a conjunction of terms including \( a_i^r \land \neg a_j^r \). Similarly, \( Q_{i,s'} \) is a conjunction of terms including \( a_i^m \iff a_j^m \) for each \( m \) in \( \{\ell - s', \ell - 1\} \). Since \( r \) lies in this interval, the conjunction includes \( a_i^r \iff a_j^r \). Consequently, either \( a_i^r = a_j^r \) and \( Q_{i,s} \) is false, or \( a_i^r \neq a_j^r \) and \( Q_{i,s'} \) is false.

Then, we prove correctness of the crypto computing algorithm.

**Theorem 1.** For any bit precision \( \ell \), for any number of participants \( p \), for any integer \( i \in \{1, \ldots, p\} \) the proposed algorithm correctly outputs with probability \( 1 - \mathcal{O}(\ell^{p-1}/N) \) a random element uniformly distributed in \( \text{Enc}_{\ell^{p-1}} \) (resp. \( \text{Enc}_{\ell^{p-1}}^0 \)) iff the predicate \( P_i \) is true (resp. false).
Proof. We will prove that the probabilistic computation is correct at each level of the circuit. For the input encrypted data, and the first level, the verification is obvious and true with probability 1. For the AND gates at the second level: if some bits are 0 then \( r \sum_{i=1}^{s} x_i \neq 0 \) holds with probability \((N - 1)/N\) and if all the bits are 1 then \( \sum_{i=1}^{s} x_i = 0 \) with probability 1. Since this layer includes \( \ell(p-1) \) such AND gates, the set of its output is correctly computed with probability \(( (N - 1)/N )^{\ell(p-1)} \). We now consider the OR gates at the third level. If each of the input bits is 0, then the sequence of such bits is also 0. Otherwise, from the previous Lemma, it follows that exactly one input is 1, so the output sequence lies in the correct space. In both cases, it is easily verified that the output is uniformly distributed in \( Enc_{\ell} \). Furthermore, assuming correct inputs, the whole computation of this \( \ell \)-gate layer is correct with probability 1. Finally, the AND gate at the fourth level outputs a sequence of encryptions that performs the product of each \((p - 1)\)-tuples of \( Enc_{\ell} \). Thus, if each input is 1, each input includes one encryption of \( \text{zero} \), and this combination leads to exactly one \( \text{zero} \). If there exists a \( \text{zero} \) input, then, assuming inputs are correct, it is encryptions of only non \( \text{zero} \) terms, and thus each sum is non-\( \text{zero} \) with probability \((N - 1)/N\). The conditional probability of correctness of this whole layer is then \(( (N - 1)/N )^{\ell(p-1)} \). The uniformity is easily seen. In conclusion, it results that the success of the computation holds with probability \((1 - 1/N )^{\ell(p-1) + \ell(p-1)}\) which is greater than \( 1 - (\ell(p - 1) + \ell(p-1))/N \). □

Having performed these computations, the server publishes a bulletin board containing the results of the predicates \( P_i \) encrypted under the public key of the \( i^{th} \) player. The amount of data is \( \Theta(p^{\ell(p-1)}) \) which is reasonably small for 4 to 5 players. Then each player decrypts its sequence of encrypted data, which either leads to a set of non-\( \text{zero} \) values, in case he has lost the auction, or to exactly one \( \text{zero} \) in case he is the winner. To prove his status, the winner sends the full decryption of the encrypted value of zero to the server. This transaction may occur publicly, or through a secure channel. If several players have submitted the same maximum bid, then they may all prove they did and an additional round can take place. It should be noticed that the initial input bits are only a small subset of the plaintext space. Therefore, dishonest bidders could encrypt values that are not real bits (ie: not 0 nor 1). Then the whole protocol would collapse, since circuit evaluation would produce only false value, for example if the leading bit is \("z\). The next section proposes an enhanced version of our scheme where each participant proves that he has only encrypted fair bits. Before doing so, we turn to the last part of the protocol.

### 2.3 End of the protocol

In the standard case, it is assumed that the winner makes himself known. Otherwise, if he remains silent, one may consider various solutions, e.g. asking other users to prove that they really lose the auction by decrypting each of the messages announcing the results. We underline that such a hiding player does not compromise privacy. Next, it remains to set the price. First, we consider a scenario where the transaction is done at the highest price (sealed-bid auctions). Since the server has no information on the initial plaintexts, the winner has to reveal the full decryptions of his bid and the \( \text{zero} \) message proving that he has won. Remind that a full decryption provides the cleartext message and the random coins that enables to check the validity of a given encryption. Using the homomorphism of \( E \), this phase is very efficient: from the initial data \( E(x_i) \) he has sent,
the winner computes the encrypted message

\[ E \left( \sum_{i=0}^{t-1} x_i 2^i \right) = \prod_{i=0}^{t-1} E(x_i)^{2^i} \]

and sends its full decryption to the server. Then this one checks the validity of the computation. In a second scenario, where we consider that the value of the transaction is set at the 2nd highest price (as is well known, this scheme is equivalent to a public “English” auction) further computation and an additional round of interaction are needed. Basically, once the winner has revealed himself by decrypting a zero from the bulletin board, the server withdraws this encrypted bid and computes the maximum predicates over the remaining bids. Then, it sends a random permutation of the predicates to the winner and asks him to provide the full decryption of the zero value it contains, together with the underlying \( \ell \)-bit bid. Then the server checks the decryption values he has received, and publicly announce the price of the transaction as in the previous case.

3 A robust protocol against cheating players

We now turn to a scenario where some dishonest players may send arbitrary data, possibly not encrypting fair bits or encrypting different bids under the different public keys. We have already observed that this could compromise the auction: if the \( j^{th} \) player submits the encryption of an \( \ell \)-bit integer \( a_j \) with an unfair leading “bit” then for each predicate \( (P_i) \), the evaluations of \( -a_j^{\ell-1} \) and \( a_j^{\ell-1} \land a_i^{\ell-1} \) would both leads to false. The same holds for the predicate \( (P_j) \) considering \( a_j^{\ell-1} \land a_i^{\ell-1} \). As a result, none of the participant could be declared the winner. It may be asked to each player to decrypt his own data, but contrary to the situation where we considered a fair but silent player, this would compromise privacy and is not acceptable here. Therefore, in order to achieve robustness, each participant adds a short proof of fairness to his encrypted bid. We will consider the specific homomorphic encryption scheme proposed by Pailler [18] at Eurocrypt’99 whose overview is given below. Using this system, we will design a proof of fair encryption of bits.

3.1 Overview of Paillier’s encryption scheme

**Key Generation.** Let \( N \) be a RSA modulus of \( k+1 \) bits, where \( k \) is a security parameter. Let \( g \) be an element of \( \mathbb{Z}_N^{*} \) whose order is a large multiple of \( N \). The public parameters are \( N \) and \( g \) whilst the factorization of \( N \), or equivalently \( \lambda(N) \), remains secret. Recall that in this case the Carmichael function \( \lambda \) is \( \lambda(N) = \text{lcm}(p-1, q-1) \).

**Encryption.** The space of plaintext messages \( \mathcal{M} \) is \( \mathbb{Z}_N \). The encryption of a message \( m \in \mathcal{M} \) is \( E(m) = g^m r^N \mod N^2 \) where \( r \) is randomly chosen in \( \mathbb{Z}_N^{*} \). \( m \) is called the \( N^{th} \) residuosity class of \( c \) with respect to \( g \).

**Decryption.** Let \( L \) be the function \( L(u) = (u-1)/N \) defined over the subgroup \( \mathcal{S}_N = \{ u < N^2 \mid u \equiv 1 \mod N \} \). For any ciphertext \( c = g^m r^N \mod N^2 \), using the trapdoor \( \lambda(N) \), it holds that \( m = \frac{L(c^{\lambda(N)} \mod N^2)}{L(g^{\lambda(N)} \mod N^2)} \). Full decryption is achieved by extracting the \( N^{th} \) root mod \( N \) of \( (cg^{-m} \mod N) \).

Assuming the hardness of deciding composite residuosity classes, this encryption scheme
is proven to be semantically secure against a chosen plaintext attack. Using appropriate optimizations, the workload for encryption and decryption is of the same order of magnitude as RSA. The required properties for our auction protocol are efficient and easily verified: self-randomization is achieved through a single modular exponentiation and the additive homomorphic property is obvious. Furthermore, the scheme enjoys the advantage of encrypting 0 in a $N^{th}$ residue. Therefore, using the additive homomorphic property, $c$ encrypts a fair bit if and only if either $c$ or $c/E(1)$ is a $N$-residue. This leads to an efficient proof of fair encryption described below.

### 3.2 Zero-knowledge proof of fair encryption of a bit

To prove that one correctly encrypted a plaintext in $\{0,1\}$, we combine a Guillou-Quisquater proof of knowledge of a $N^{th}$ root [13] with a proof of knowledge of one discrete log out of two $[7,8,4]$. Firstly, we propose a 3-round interactive protocol between a prover $P$ and a verifier $V$, then we turn it into an non interactive protocol using hash functions, as usual.

**Settings:** $k \in \mathbb{N}$ and $A$ are security parameters. $N$ is a RSA modulus of $k$ bits. $P$ owns a secret value $b \in \{0,1\}$ and publishes $c = r^b \mod N^2$ where $r$ is a random secret value in $\mathbb{Z}_N$. We note $c_0 = c$ and $c_1 = c/g$. The following 3 rounds of interaction is iterated $t$ times.

**1st round :** $P \rightarrow V$

$P$ picks at random two values $r_0$ and $r_1$ in $\mathbb{Z}_N$. He has to commit to $u_0$ and $u_1$, as if he was trying to prove in parallel that both $c_0$ and $c_1$ are $N$-residues. To this end, since only $c_b$ is an actual residue, further messages indexed by $b$ are fairly computed, whereas messages indexed by $1 - b$ take advantage of the malleability of the challenge. So, the prover chooses half of the challenge in advance, by picking at random $e_{1-b} \in \mathbb{Z}_A$. This knowledge enables him to choose at random the corresponding final answer $v_{1-b}$ in $\mathbb{Z}_N^{*2}$. Then he computes a fake commitment $u_{1-b}$ satisfying the verifier’s equality and a fair commitment $u_b$ such that

$$
\begin{align*}
  u_b & = r_b^N \mod N^2 \\
  u_{1-b} & = v_{1-b}^{e_{1-b}}/c_{1-b}^{e_{1-b}} \mod N^2
\end{align*}
$$

Finally he sends $u_0$ and $u_1$ to the prover.

**2nd round :** $V \rightarrow P$

$V$ picks a random a challenge $e$ in $\mathbb{Z}_A$ and sends it to $P$.

**3rd round :** $P \rightarrow V$

$P$ computes the regular challenge $e_b$ such that $e = e_0 + e_1 \mod A$. It also computes $v_b = r_b^{e_b}$. Then he sends $v_0, v_1, e_0, e_1$ to $V$.

$V$ verifies that

$$
\begin{align*}
  v_0^N & = u_0 c_0^e_0 \mod N^2 \\
  v_1^N & = u_1 c_1^e_1 \mod N^2 \\
  e & = e_0 + e_1 \mod A
\end{align*}
$$

**Remark 1.** In the last round of interaction, the prover may be asked not to send $e_1$ since it is deducible from $e$ and $e_0$. Also, the last test performed by the verifier may be discarded by using $e - e_0$ instead of $e_1$. This presentation is for convenience only. The figure shows the actual protocol.
Theorem 2. For any positive constants $\alpha$ and $\beta$, for any non-zero parameters $A$ and $t$ such that $A = O(k^\alpha)$ and $t = \Omega(\log^{1+\beta} k)$, it holds that $t$ iterations of the previous protocol is a perfect zero-knowledge proof of membership that $c$ is a fair encryption of a bit.

Proof. We note $L$ the language of fair encrypted bits:

$$L = \{g^{b r^N} \mod N^2 \mid b \in \{0, 1\}, r \in \mathbb{Z}^*_N\}$$

Completeness. Assume $c \in L$. Then either $c$ or $c/g$ is a $N^{th}$ residue. For this residue, the prover may answer to any challenge $e_b$. Thanks to the degree of freedom, he has the ability to fix in advance the challenge $e_{1-b}$ and forge the appropriate answer $v_{1-b}$. Therefore the prover is accepted with probability 1.

Soundness. Assume $c \notin L$. Suppose that a cheating prover $P^*$ successfully completes an iteration of the protocol. From the final verifying equations and the expression of $c$ we have

$$\begin{cases}
    v_0^N = u_0 \cdot g^{b e_0 r e_0 N} \mod N^2 \\
    v_1^N = u_1 \cdot g^{(b-1) e_1 r e_1 N} \mod N^2
\end{cases}$$

Taking the logarithms of each expression, it follows

$$\begin{cases}
    \log u_0 + b e_0 = 0 \mod N \\
    \log u_1 + (b-1) e_1 = 0 \mod N
\end{cases}$$

So we have the system of 3 equations in the variables $e_0$ and $e_1$

$$\begin{cases}
    b e_0 = - \log u_0 \mod N \\
    (b-1)e_1 = - \log u_1 \mod N \\
    e_0 + e_1 = e \mod A
\end{cases}$$
If $b$ is different from 0 and 1, it follows that $e_0$ and $e_1$ are functions of $b$ and the original commitment \( \{u_0, u_1\} \). Therefore, the third equation holds with probability at most \( 1/A \). If the protocol is iterated $t$ times, then standard arguments show that the probability that $P^*$ passes the protocol cannot significantly exceed \( 1/A^t \). Since $A$ is a positive integer and $t = \Omega(\log^{1+\beta} k)$ the probability of success is $O(k^{-1} \log A \log^\delta k)$ which is a negligible function of $k$.

**Simulation.** Fix any verifier $V^*$. First guess the challenge: pick $e'$ randomly in $[0, A]$. Then choose $e_0$ and $e_1$ such that $e' = e_0 + e_1$. Next compute $u_0 = v_0^N / c_0$ and $u_1 = v_1^N / (c/g)^{e_1}$ and send $u_0$ and $u_1$ (mod $N^2$) to $V^*$. If $V^*$ answers $e$ such that $e = e'$ then this iteration is successfully completed by sending $v_0, v_1, e_0$ and $e_1$. Otherwise, rewind the simulation to the beginning of the iteration. It results that the whole protocol is perfectly simulated in expected time $O(A \cdot t)$.

From a practical point of view, it may be desirable to perform a single iteration of the 3-round protocol. Then, since a large $A$ is required to ensure soundness of the protocol, the resulting scheme is not zero-knowledge anymore. However, no strategy is known to increase the probability of accepting a dishonest prover.

### 3.3 Equalities of bids under multiple encryptions

To achieve robustness of the submission protocol, it is also required that each bidder proves that he has encrypted the same bits under the different public keys. As shown in equation (3), the server learns an encryption of the $\ell$-bit integer submission. Therefore it remains to prove equality of $p$ discrete logs lying in a given interval [5]. Following the previous section, we first propose an interactive zero-knowledge proof between a prover $P$ and a verifier $V$.

**Settings:** $k, k'$ and $A$ are security parameters such that $2^\ell A < 2^{k+k'}$. The set \( \{N_i\}_{1 \leq i \leq p} \) are RSA moduli of $k + 1$ bits. $P$ owns a secret value $x \in [0, 2^\ell]$ and publishes \( \{c_i = g_i r_i^{N_i} \mod N_i^2\}_{1 \leq i \leq p} \) where the $r_i$ are $p$ random secret values in $\mathbb{Z}_{N_i}^*$.

**1st round:** $P \rightarrow V$

$P$ picks at random $\rho \in [0, 2^k]$, and $s_i \in \mathbb{Z}_{N_i}^*$ for each $i = 1, \ldots, p$. Then he commits to \( \{u_i = g_i^{\rho} r_i^{N_i} \mod N_i^2\}_{1 \leq i \leq p} \).

**2nd round:** $V \rightarrow P$

$P$ picks at random a challenge $e \in [0, A[$ and sends it to $P$.

**3rd round:** $P \rightarrow V$

$P$ computes $z = \rho + xe$, and \( \{v_i = s_i r_i^e \mod N_i^2\}_{1 \leq i \leq p} \) and sends them to $V$. Then $V$ verifies that $z < 2^k$ and $g_i^z v_i^{N_i} = u_i c_i^e \mod N_i^2$ for each $i = 1, \ldots, p$.

**Theorem 3.** For any positive constants $\alpha, \beta$ and $\gamma$, for any non-zero parameters $A$, $t$ and $\ell$ such that $A = O(k^\alpha)$, $t = \Omega(\log^{1+\beta} k)$ and $\ell = k - \Omega(\log^{1+\gamma} k)$, it holds that $t$ iterations of the previous protocol provides a statistical zero-knowledge proof of membership that elements \( \{c_1, \ldots, c_p\} \) encrypt the same $\ell$-bit message.

**Proof.**

**Completeness.** For any $i \in \{1, \ldots, p\}$, it holds that $g_i^z v_i^{N_i} = g_i^{\rho+xe} s_i^{N_i} r_i^{N_i} = u_i c_i^{e} \mod N_i^2$, with probability 1. Furthermore, since $z = \rho + xe$, the inequality $z < 2^k$ holds with probability at least $1 - 2^\ell A/2^k$. Thus, a honest prover successfully completes $t$ iterations of the protocol with probability at least $1 - 2^{\ell-k} At$. Since $t$ and $A$ are upper-bounded on $A$, the protocol is complete and zero-knowledge.
by polynomials and $2^{l-k} = O(k^{-\log^2 k})$, this probability is overwhelming.

**Soundness.** Assume there exists $i_1$ and $i_2$ in $\{1, ..., p\}$ such that $c_{i_1}$ decrypts $x_1$ and $c_{i_2}$ decrypts $x_2$ with $x_1 \neq x_2$. Then, from the equalities verified by $V$

$$
\begin{align*}
g_{i_1}^{v_{N_{i_1}}} &= u_{i_1} g_{t_1}^{v_{N_{i_1}}} \mod N_1 \\
g_{i_2}^{v_{N_{i_2}}} &= u_{i_2} g_{t_2}^{v_{N_{i_2}}} \mod N_2
\end{align*}
$$

Taking the logarithms it follows

$$
\begin{align*}
z &= \log u_{i_1} + e x_1 \mod N_1 \\
z &= \log u_{i_2} + e x_2 \mod N_2
\end{align*}
$$

Since $z < 2^k$ then $z - e < N_{i_1}, N_{i_2}$ and both equalities hold without the moduli. It results that $\log u_{i_1} + e x_1 = \log u_{i_2} + e x_2$ in the integers. So, if $x_1 \neq x_2$, $e = (\log u_{i_1} - \log u_0)/(x_2 - x_1)$, which occurs with probability at most $A$.

**Simulation.** Following the previous proof, the same resettable simulation works. However, since the simulator uniformly picks $z$ in $[0, 2^k]$ and not in $[xe, 2^k + xe]$, only a statistical indistinguishability can be achieved (see [20] for a complete proof).

We are now ready to design a robust auction protocol. The main operation is to replace the verifiers by a secure hash function such as SHA-1. This leads to non-interactive proofs that has to be stuck to the bit encryptions. To reduce the amount of data, the following trick may be used. The $lp$ proofs of fair encryption consist of their last two rounds $\{e; v_0, v_1, e_0\}$ and a hash of the parallel commitments. To check the proof, these commitments are first computed from the last predicates of equality, then the whole verifications are performed. As a result the total length these $lp$ proofs is no more than $3lp$ encryptions. In the same way, the proof of equality of logs consists of the last $p + 2$ messages from rounds 2 and 3. Thus its length is about $p + 2$ encryptions. One can also ask that the proofs are given only in the case that the server is unable to provide any winner. This makes an additional round of interaction, but still preserves the privacy of each bidder.

4 Dealing with many participants

To cope with real-life Internet business application, it is obvious that the number of total participants should be increased. Under the hypothesis that we accept a partial leak of information and reasonable interaction, we can substantially gain efficiency and deal with a polynomial number of players. A possible approach is to form small groups of users and perform the protocol to decide who owns the maximum bid inside of them. Assume that we allow $q$ participants in each group, then we can build a $q$-ary tree and achieve the whole protocol of bid submission in a number of rounds proportional to $\log_q(p)$. Next, if we have to find the second highest bid, we consider the following algorithm: form the path of the progression of the winner in the $q$-ary tree and select all the participants that are present in one of the winner’s subgroup along this path. This list contains at most $q \log_q p$ players where it remains to extract the highest bid.

5 Conclusion

We have proposed a practical protocol of auctions with a high degree of confidence and very few interaction. Compared to existing schemes, we focused on security. The drawback resides in the limited number of players that may simultaneously participate in a scenario where absolute privacy is needed. Nonetheless, we believe that in many scenarios these parameters meet real life applications.
\[
\begin{array}{c|c}
\text{Prover} & \text{Verifier} \\
\end{array}
\]

\[
\{N_i = p_i q_i\}_{1 \leq i \leq p}, \text{ such that each } |N_i| = k + 1 \\
A < 2^k : \text{ security parameter} \\
\{g_i \in \mathbb{Z}_{N_i}^* \text{ of order multiple of } N_i\}_{1 \leq i \leq p}
\]

\[
\begin{align*}
\text{secret} & : \{p_i, q_i\}_{1 \leq i \leq p} \\
x & \in [0, 2^k] \\
\{r_i \in \mathbb{Z}_{N_i}\}_{1 \leq i \leq p} \\
\text{public} & : \{c_i = g_i^r N_i \mod N_i^2\}_{1 \leq i \leq p} \\
\rho & \in [0, 2^k] \\
\{s_i \in \mathbb{Z}_{N_i}\}_{1 \leq i \leq p} \\
\{u_i = g_i^s N_i \mod N_i^2\}_{1 \leq i \leq p} & \overset{e}{\rightarrow} \{u_i \mod N_i^2\}_{1 \leq i \leq p} \\
z = \rho + xe \\
\{v_i = s_i r_i \mod N_i^2\}_{1 \leq i \leq p} & \overset{z, \{v_i \mod N_i^2\}_{1 \leq i \leq p}}{\rightarrow} z < 2^k \\
\{g_i v_i^N_i \mod N_i^2\}_{1 \leq i \leq p}
\end{align*}
\]

\[\text{Fig. 2. Zero-knowledge proof of equality of logs}\]

References

2. M. Ben-Or, S. Goldwasser, and A. Wigderson. Completeness theorems for non-


